aws

SQLReferensi

AWS Clean Rooms

AWS Clean Rooms: SQLReferensi

Table of Contents

Ikhtisar SQL di AWS Clean Rooms

Selamat datang di Referensi AWS Clean Rooms SQL.

AWS Clean Rooms dibangun di sekitar standar industri Structured Query Language (SQL), bahasa query yang terdiri dari perintah dan fungsi yang Anda gunakan untuk bekerja dengan database dan objek database. SQL juga memberlakukan aturan mengenai penggunaan tipe data, ekspresi, dan literal.

Topik berikut memberikan informasi umum tentang konvensi, aturan penamaan, dan tipe data:

Topik

Untuk mendapatkan pemahaman tentang perintah SQL, jenis fungsi SQL, dan kondisi SQL yang dapat Anda gunakan AWS Clean Rooms, tinjau topik berikut:

- Perintah SQL di AWS Clean Rooms
- Fungsi SQL di AWS Clean Rooms
- Kondisi SQL di AWS Clean Rooms

Untuk informasi selengkapnya AWS Clean Rooms, lihat Panduan AWS Clean Rooms Pengguna dan Referensi AWS Clean Rooms API.

Konvensi referensi SQL

Bagian ini menjelaskan konvensi yang digunakan untuk menulis sintaks untuk ekspresi SQL, perintah, dan fungsi.

Karakter	Deskripsi
PENUTUP	Kata-kata dalam huruf kapital adalah kata kunci.

Karakter	Deskripsi
[ ]	Kurung menunjukkan argumen opsional. Beberapa argumen dalam tanda kurung menunjukkan bahwa Anda dapat memilih sejumlah argumen. Selain itu, argumen dalam tanda kurung pada baris terpisah menunjukkan bahwa parser mengharapkan argumen untuk berada dalam urutan bahwa mereka terdaftar dalam sintaks.
{ }	Tanda kurung menunjukkan bahwa Anda diminta untuk memilih salah satu penjelasan di dalam kurung kurawal.
|	Pipa menunjukkan bahwa Anda dapat memilih di antara argumen.
huruf miring	Kata-kata yang dicetak miring menunjukkan placehold er. Anda harus memasukkan nilai yang sesuai di tempat kata yang dicetak miring.
...	Elipsis menunjukkan bahwa Anda dapat mengulangi elemen sebelumnya.
'	Kata-kata dalam tanda kutip tunggal menunjukkan bahwa Anda harus mengetikkan tanda kutip.

Aturan penamaan SQL

Bagian berikut menjelaskan aturan penamaan SQL diAWS Clean Rooms.

Dikonfigurasi nama asosiasi tabel dan kolom

Anggota yang dapat query menggunakan dikonfigurasi nama asosiasi tabel sebagai nama tabel dalam query. Dikonfigurasi nama asosiasi tabel dan kolom tabel dikonfigurasi dapat alias dalam query.

Aturan penamaan berikut berlaku untuk nama asosiasi tabel yang dikonfigurasi, nama kolom tabel yang dikonfigurasi, dan alias:

- Mereka hanya harus menggunakan karakter alfanumerik, garis bawah (_), atau tanda hubung (-) tetapi tidak dapat memulai atau diakhiri dengan tanda hubung.
  - (Hanya aturan analisis khusus) Mereka dapat menggunakan tanda dolar ($) tetapi tidak dapat menggunakan pola yang mengikuti konstanta string yang dikutip dolar.

    Konstanta string yang dikutip dolar terdiri dari:
    - tanda dolar ($)
    - opsional “tag” dari nol atau lebih karakter
    - tanda dolar lain
    - urutan sewenang-wenang karakter yang membentuk konten string
    - tanda dolar ($)
    - tag yang sama yang memulai kutipan dolar
    - tanda dolar

    Misalnya: `$$invalid$$`
- Mereka tidak dapat berisi tanda hubung berturut-turut (-) karakter.
- Mereka tidak dapat memulai dengan salah satu awalan berikut:

  `padb_`, `pg_`, `stcs_`, `stl_`, `stll_`, `stv_`, `svcs_`, `svl_`, `svv_`, `sys_`, `systable_`
- Mereka tidak dapat berisi karakter garis miring terbalik (\), tanda kutip ('), atau spasi yang tidak dikutip ganda.
- Jika mereka mulai dengan karakter non-abjad, mereka harus dalam tanda kutip ganda (” “).
- Jika mereka mengandung tanda hubung (-) karakter, mereka harus dalam tanda kutip ganda (” “).
- Panjangnya harus antara 1 dan 127 karakter.
- Kata-kata yang dipesanharus dalam tanda kutip ganda (” “).
- Nama kolom berikut dicadangkan tidak dapat digunakan dalamAWS Clean Rooms(bahkan dengan kutipan):
  - oid
  - tableoid
  - xmin
  - cmin

- xmax
- cmax
- ctid

Literal

Sebuah literal atau konstan adalah nilai data tetap, terdiri dari urutan karakter atau konstanta numerik.

Aturan penamaan berikut adalah untuk literal diAWS Clean Rooms:

- Numerik, karakter dan tanggal, waktu, dan literal timestamp didukung.
- HanyaTAB,CARRIAGE RETURN(CR), danLINE FEED(LF) Karakter kontrol Unicode dari kategori umum Unicode (Cc) didukung.
- Referensi langsung ke literal dalam daftar proyeksi tidak didukung dalam pernyataan SELECT.

  Misalnya:

```
SELECT 'test', consumer.first_purchase_day
FROM consumer
INNER JOIN provider2
ON consumer.hashed_email = provider2.hashedemail
```

Kata-kata yang dipesan

Berikut ini adalah daftar kata-kata reserved diAWS Clean Rooms.

AES128	DELTA32KDESC	LEADING	PRIMARY
AES256ALL	DISTINCT	LEFTLIKE	RAW
ALLOWOVER WRITEANALYSE	DO	LIMIT	READRATIO
ANALYZE	DISABLE	LOCALTIME	RECOVERRE FERENCES

AND	ELSE	LOCALTIMESTAMP	REJECTLOG
ANY	EMPTYASNU LLENABLE	LUN	RESORT
ARRAY	ENCODE	LUNS	RESPECT
AS	ENCRYPT	LZO	RESTORE
ASC	ENCRYPTIONEND	LZOP	RIGHTSELECT
AUTHORIZATION	EXCEPT	MINUS	SESSION_USER
AZ64	EXPLICITFALSE	MOSTLY16	SIMILAR
BACKUPBETWEEN	FOR	MOSTLY32	SNAPSHOT
BINARY	FOREIGN	MOSTLY8NATURAL	SOME
BLANKSASN ULLBOTH	FREEZE	NEW	SYSDATESYSTEM
BYTEDICT	FROM	NOT	TABLE
BZIP2CASE	FULL	NOTNULL	TAG
CAST	GLOBALDICT256	NULL	TDES
CHECK	GLOBALDIC T64KGRANT	NULLSOFF	TEXT255
COLLATE	GROUP	OFFLINEOFFSET	TEXT32KTHEN
COLUMN	GZIPHAVING	OID	TIMESTAMP
CONSTRAINT	IDENTITY	OLD	TO
CREATE	IGNOREILIKE	ON	TOPTRAILING
CREDENTIA LSCROSS	IN	ONLY	TRUE

CURRENT_DATE	INITIALLY	OPEN	TRUNCATEC OLUMNSUNION
CURRENT_TIME	INNER	OR	UNIQUE
CURRENT_T IMESTAMP	INTERSECT	ORDER	UNNEST
CURRENT_USER	INTERVAL	OUTER	USING
CURRENT_U SER_IDDEFAULT	INTO	OVERLAPS	VERBOSE
DEFERRABLE	IS	PARALLELP ARTITION	WALLETWHEN
DEFLATE	ISNULL	PERCENT	WHERE
DEFRAG	JOIN	PERMISSIONS	WITH
DELTA	LANGUAGE	PIVOTPLACING	WITHOUT

Jenis Data

Setiap nilai yang AWS Clean Rooms menyimpan atau mengambil memiliki tipe data dengan set tetap properti terkait. Tipe data dideklarasikan saat tabel dibuat. Tipe data membatasi kumpulan nilai yang dapat berisi kolom atau argumen.

Tabel berikut mencantumkan tipe data yang dapat Anda gunakan dalam AWS Clean Rooms tabel.

Jenis data	Alias	Deskripsi
ARRAY	Tidak berlaku	Tipe data bersarang array
BIGINT	Tidak berlaku	Bilangan bulat delapan byte bertanda
BOOLEAN	BOOL	Logis Boolean (benar/salah)

Jenis data	Alias	Deskripsi
CHAR	KARAKTER	String karakter dengan panjang tetap
DATE	Tidak berlaku	Tanggal kalender (tahun, bulan, hari)
DECIMAL	NUMERIC	Numerik persis dari presisi yang dapat dipilih
DOUBLE PRECISION	MENGAPUNG8, MENGAPUNG	Angka floating-point presisi ganda
INTEGER	INT	Bilangan bulat empat byte bertanda
PETA	Tidak berlaku	Memetakan tipe data bersarang
REAL	FLOAT4	Angka floating-point presisi tunggal
SMALLINT	Tidak berlaku	Bilangan bulat dua byte bertanda
STRUCT	Tidak berlaku	Struct tipe data bersarang
SUPER	Tidak berlaku	Tipe data superset yang mencakup semua jenis skalar AWS Clean Rooms termasuk tipe kompleks seperti ARRAY dan STRUCTS.
TIME	Tidak berlaku	Waktu hari
JADWAL	Tidak berlaku	Waktu hari dengan zona waktu

Jenis data	Alias	Deskripsi
VARBYTE	VARBINARY, BINER BERVARIASI	Nilai biner dengan panjang variabel
VARCHAR	KARAKTER BERVARIASI	String karakter panjang variabel dengan batas yang ditentukan pengguna

> **Note**
>
> Tipe data bersarang ARRAY, STRUCT, dan MAP saat ini hanya diaktifkan untuk aturan analisis kustom. Untuk informasi selengkapnya, lihat Jenis bersarang.

Karakter multibyte

Tipe data VARCHAR mendukung karakter multibyte UTF-8 hingga maksimal empat byte. Karakter lima byte atau lebih lama tidak didukung. Untuk menghitung ukuran kolom VARCHAR yang berisi karakter multibyte, kalikan jumlah karakter dengan jumlah byte per karakter. Misalnya, jika string memiliki empat karakter Mandarin, dan setiap karakter panjangnya tiga byte, maka Anda memerlukan kolom VARCHAR (12) untuk menyimpan string.

Tipe data VARCHAR tidak mendukung titik kode UTF-8 yang tidak valid berikut ini:

`0xD800 – 0xDFFF`(Urutan byte: `ED A0 80` —) `ED BF BF`

Tipe data CHAR tidak mendukung karakter multibyte.

Jenis numerik

Topik

- Jenis bilangan bulat
- Jenis DESIMAL atau NUMERIK
- Catatan tentang menggunakan kolom DESIMAL atau NUMERIK 128-bit
- Jenis Floating-point

- Perhitungan dengan nilai numerik

Tipe data numerik termasuk bilangan bulat, desimal, dan angka floating-point.

Jenis bilangan bulat

Gunakan tipe data SMALLINT, INTEGER, dan BIGINT untuk menyimpan seluruh nomor dari berbagai rentang. Anda tidak dapat menyimpan nilai di luar rentang yang diizinkan untuk setiap jenis.

Nama	Penyimpanan	Kisaran
SMALLINT	2 byte	-32768 ke +32767
INTEGER atau INT	4 byte	-2147483648 ke +2147483647
BIGINT	8 byte	-92233720368547758 08 ke 922337203 6854775807

Jenis DESIMAL atau NUMERIK

Gunakan tipe data DECIMAL atau NUMERIK untuk menyimpan nilai dengan presisi yang ditentukan pengguna. Kata kunci DECIMAL dan NUMERIK dapat dipertukarkan. Dalam dokumen ini, desimal adalah istilah yang disukai untuk tipe data ini. Istilah numerik digunakan secara umum untuk merujuk pada tipe data integer, desimal, dan floating-point.

Penyimpanan	Kisaran
Variabel, hingga 128 bit untuk tipe DECIMAL yang tidak terkompresi.	Bilangan bulat bertanda 128-bit dengan presisi hingga 38 digit.

Tentukan kolom DECIMAL dalam tabel dengan menentukan presisi dan skala:

```
decimal(precision, scale)
```

presisi

Jumlah total digit signifikan dalam seluruh nilai: jumlah digit di kedua sisi titik desimal. Misalnya, angka tersebut `48.2891` memiliki presisi 6 dan skala 4. Presisi default, jika tidak ditentukan, adalah 18. Presisi maksimum adalah 38.

Jika jumlah digit di sebelah kiri titik desimal dalam nilai input melebihi presisi kolom dikurangi skalanya, nilai tidak dapat disalin ke kolom (atau dimasukkan atau diperbarui). Aturan ini berlaku untuk setiap nilai yang berada di luar rentang definisi kolom. Misalnya, rentang nilai yang diizinkan untuk `numeric(5,2)` kolom adalah `-999.99` untuk`999.99`.

skala

Jumlah digit desimal di bagian pecahan nilai, di sebelah kanan titik desimal. Bilangan bulat memiliki skala nol. Dalam spesifikasi kolom, nilai skala harus kurang dari atau sama dengan nilai presisi. Skala default, jika tidak ditentukan, adalah 0. Skala maksimum adalah 37.

Jika skala nilai input yang dimuat ke dalam tabel lebih besar dari skala kolom, nilainya dibulatkan ke skala yang ditentukan. Misalnya, kolom PRICEPAID dalam tabel PENJUALAN adalah kolom DECIMAL (8,2). Jika nilai DECIMAL (8,4) dimasukkan ke dalam kolom PRICEPAID, nilainya dibulatkan ke skala 2.

```
insert into sales
values (0, 8, 1, 1, 2000, 14, 5, 4323.8951, 11.00, null);

select pricepaid, salesid from sales where salesid=0;

pricepaid | salesid
-----------+---------
4323.90 |       0
(1 row)
```

Namun, hasil pemeran eksplisit nilai yang dipilih dari tabel tidak dibulatkan.

Note

Nilai positif maksimum yang dapat Anda masukkan ke dalam kolom DECIMAL (19,0) adalah `9223372036854775807` (2 63 -1). Nilai negatif maksimum adalah`-9223372036854775807`. Misalnya, upaya untuk memasukkan nilai `9999999999999999` (19 nines) akan menyebabkan kesalahan

overflow. Terlepas dari penempatan titik desimal, string terbesar yang AWS Clean Rooms dapat mewakili sebagai angka DESIMAL adalah. `9223372036854775807` Misalnya, nilai terbesar yang dapat Anda muat ke kolom DECIMAL (19,18) adalah. `9.223372036854775807`
Aturan-aturan ini adalah karena hal-hal berikut:

- Nilai DESIMAL dengan 19 atau kurang digit presisi signifikan disimpan secara internal sebagai bilangan bulat 8-byte.
- Nilai DESIMAL dengan 20 hingga 38 digit presisi signifikan disimpan sebagai bilangan bulat 16-byte.

Catatan tentang menggunakan kolom DESIMAL atau NUMERIK 128-bit

Jangan sewenang-wenang menetapkan presisi maksimum ke kolom DECIMAL kecuali Anda yakin bahwa aplikasi Anda memerlukan presisi itu. Nilai 128-bit menggunakan ruang disk dua kali lebih banyak daripada nilai 64-bit dan dapat memperlambat waktu eksekusi kueri.

Jenis Floating-point

Gunakan tipe data REAL dan DOUBLE PRECISION untuk menyimpan nilai numerik dengan presisi variabel. Tipe ini adalah tipe yang tidak tepat, artinya beberapa nilai disimpan sebagai perkiraan, sehingga menyimpan dan mengembalikan nilai tertentu dapat mengakibatkan sedikit perbedaan. Jika Anda memerlukan penyimpanan dan perhitungan yang tepat (seperti untuk jumlah uang), gunakan tipe data DECIMAL.

REAL mewakili format floating point presisi tunggal, menurut IEEE Standard 754 untuk Floating-Point Arithmetic. Ini memiliki presisi sekitar 6 digit, dan kisaran sekitar 1E-37 hingga 1E+37. Anda juga dapat menentukan tipe data ini sebagai FLOAT4.

DOUBLE PRECISION mewakili format floating point presisi ganda, menurut IEEE Standard 754 untuk Binary Floating-Point Arithmetic. Ini memiliki presisi sekitar 15 digit, dan kisaran sekitar 1E-307 hingga 1E+308. Anda juga dapat menentukan tipe data ini sebagai FLOAT atau FLOAT8.

Perhitungan dengan nilai numerik

Dalam AWS Clean Rooms, komputasi mengacu pada operasi matematika biner: penjumlahan, pengurangan, perkalian, dan pembagian. Bagian ini menjelaskan jenis pengembalian yang

diharapkan untuk operasi ini, serta rumus spesifik yang diterapkan untuk menentukan presisi dan skala saat tipe data DECIMAL terlibat.

Ketika nilai numerik dihitung selama pemrosesan kueri, Anda mungkin mengalami kasus di mana perhitungan tidak mungkin dan kueri mengembalikan kesalahan luapan numerik. Anda mungkin juga mengalami kasus di mana skala nilai yang dihitung bervariasi atau tidak terduga. Untuk beberapa operasi, Anda dapat menggunakan casting eksplisit (jenis promosi) atau parameter AWS Clean Rooms konfigurasi untuk mengatasi masalah ini.

Untuk informasi tentang hasil perhitungan serupa dengan fungsi SQL, lihat. Fungsi SQL di AWS Clean Rooms

Jenis pengembalian untuk perhitungan

Mengingat kumpulan tipe data numerik yang didukung AWS Clean Rooms, tabel berikut menunjukkan jenis pengembalian yang diharapkan untuk operasi penambahan, pengurangan, perkalian, dan pembagian. Kolom pertama di sisi kiri tabel mewakili operan pertama dalam perhitungan, dan baris atas mewakili operan kedua.

	BERKULIT KECIL	BILANGAN BULAT	BIGINT	DESIMAL	FLOAT4	FLOAT8
BERKULIT KECIL	SMALLINT	INTEGER	BIGINT	DECIMAL	FLOAT8	FLOAT8
BILANGAN BULAT	INTEGER	INTEGER	BIGINT	DECIMAL	FLOAT8	FLOAT8
BIGINT	BIGINT	BIGINT	BIGINT	DECIMAL	FLOAT8	FLOAT8
DESIMAL	DECIMAL	DECIMAL	DECIMAL	DECIMAL	FLOAT8	FLOAT8
FLOAT4	FLOAT8	FLOAT8	FLOAT8	FLOAT8	FLOAT4	FLOAT8
FLOAT8	FLOAT8	FLOAT8	FLOAT8	FLOAT8	FLOAT8	FLOAT8

Presisi dan skala hasil DECIMAL yang dihitung

Tabel berikut merangkum aturan untuk menghitung presisi dan skala yang dihasilkan ketika operasi matematika mengembalikan hasil DECIMAL. Dalam tabel ini, `p1` dan `s1` mewakili presisi dan

skala operan pertama dalam perhitungan dan p2 dan s2 mewakili presisi dan skala operan kedua. (Terlepas dari perhitungan ini, presisi hasil maksimum adalah 38, dan skala hasil maksimum adalah 38.)

Operasi	Ketepatan dan skala hasil
+ atau -	Skala = `max(s1,s2)` presisi = `max(p1-s1,p2-s2)+1+scale`
*	Skala = `s1+s2` presisi = `p1+p2+1`
/	Skala = `max(4,s1+p2-s2+1)` presisi = `p1-s1+ s2+scale`

Misalnya, kolom PRICEPAID dan KOMISI dalam tabel PENJUALAN keduanya adalah kolom DECIMAL (8,2). Jika Anda membagi PRICEPAID dengan KOMISI (atau sebaliknya), rumusnya diterapkan sebagai berikut:

```
Precision = 8-2 + 2 + max(4,2+8-2+1)
= 6 + 2 + 9 = 17

Scale = max(4,2+8-2+1) = 9

Result = DECIMAL(17,9)
```

Perhitungan berikut adalah aturan umum untuk menghitung presisi dan skala yang dihasilkan untuk operasi yang dilakukan pada nilai DECIMAL dengan operator yang ditetapkan seperti UNION, INTERSECT, dan EXCEPT atau fungsi seperti COALESCE dan DECODE:

```
Scale = max(s1,s2)
Precision = min(max(p1-s1,p2-s2)+scale,19)
```

Misalnya, tabel DEC1 dengan satu kolom DECIMAL (7,2) digabungkan dengan tabel DEC2 dengan satu kolom DECIMAL (15,3) untuk membuat tabel DEC3. Skema DEC3 menunjukkan bahwa kolom menjadi kolom NUMERIK (15,3).

```
select * from dec1 union select * from dec2;
```

Pada contoh di atas, rumus diterapkan sebagai berikut:

```
Precision = min(max(7-2,15-3) + max(2,3), 19)
= 12 + 3 = 15

Scale = max(2,3) = 3

Result = DECIMAL(15,3)
```

Catatan tentang operasi divisi

Untuk operasi divisi, divide-by-zero kondisi mengembalikan kesalahan.

Batas skala 100 diterapkan setelah presisi dan skala dihitung. Jika skala hasil yang dihitung lebih besar dari 100, hasil pembagian diskalakan sebagai berikut:

- presisi = `precision - (scale - max_scale)`
- Skala = `max_scale`

Jika presisi yang dihitung lebih besar dari presisi maksimum (38), presisi dikurangi menjadi 38, dan skala menjadi hasil dari: `max(38 + scale - precision), min(4, 100))`

Kondisi luapan

Overflow diperiksa untuk semua perhitungan numerik. Data DESIMAL dengan presisi 19 atau kurang disimpan sebagai bilangan bulat 64-bit. Data DESIMAL dengan presisi yang lebih besar dari 19 disimpan sebagai bilangan bulat 128-bit. Ketepatan maksimum untuk semua nilai DECIMAL adalah 38, dan skala maksimum adalah 37. Kesalahan overflow terjadi ketika nilai melebihi batas ini, yang berlaku untuk set hasil menengah dan akhir:

- Casting eksplisit menghasilkan kesalahan runtime overflow ketika nilai data tertentu tidak sesuai dengan presisi atau skala yang diminta yang ditentukan oleh fungsi cast. Misalnya, Anda tidak dapat mentransmisikan semua nilai dari kolom PRICEPAID di tabel PENJUALAN (kolom DECIMAL (8,2)) dan mengembalikan hasil DECIMAL (7,3):

  ```
  select pricepaid::decimal(7,3) from sales;
  ```

```
ERROR:  Numeric data overflow (result precision)
```

Kesalahan ini terjadi karena beberapa nilai yang lebih besar di kolom PRICEPAID tidak dapat dilemparkan.

- Operasi perkalian menghasilkan hasil di mana skala hasil adalah jumlah dari skala masing-masing operan. Jika kedua operan memiliki skala 4, misalnya, skala hasilnya adalah 8, hanya menyisakan 10 digit untuk sisi kiri titik desimal. Oleh karena itu, relatif mudah untuk mengalami kondisi luapan ketika mengalikan dua angka besar yang keduanya memiliki skala signifikan.

Perhitungan numerik dengan tipe INTEGER dan DECIMAL

Ketika salah satu operan dalam perhitungan memiliki tipe data INTEGER dan operan lainnya adalah DECIMAL, operan INTEGER secara implisit dilemparkan sebagai DECIMAL.

- SMALLINT dilemparkan sebagai DECIMAL (5,0)
- INTEGER dilemparkan sebagai DECIMAL (10,0)
- BIGINT berperan sebagai DECIMAL (19,0)

Misalnya, jika Anda mengalikan SALES.COMMISSION, kolom DECIMAL (8,2), dan SALES.QTYSOLD, kolom SMALLINT, perhitungan ini dilemparkan sebagai:

```
DECIMAL(8,2) * DECIMAL(5,0)
```

Jenis karakter

Tipe data karakter termasuk CHAR (karakter) dan VARCHAR (karakter bervariasi).

Penyimpanan dan rentang

Jenis data CHAR dan VARCHAR didefinisikan dalam hal byte, bukan karakter. Kolom CHAR hanya dapat berisi karakter single-byte, sehingga kolom CHAR (10) dapat berisi string dengan panjang maksimum 10 byte. VARCHAR dapat berisi karakter multibyte, hingga maksimal empat byte per karakter. Misalnya, kolom VARCHAR (12) dapat berisi 12 karakter single-byte, 6 karakter dua-byte, 4 karakter tiga byte, atau 3 karakter empat byte.

Nama	Penyimpanan	Rentang (lebar kolom)
CHAR atau KARAKTER	Panjang string, termasuk trailing blank (jika ada)	4096 bita
VARCHAR atau KARAKTER BERVARIASI	4 byte+total byte untuk karakter, di mana setiap karakter dapat 1 sampai 4 byte.	65535 byte (64K -1)

CHAR atau KARAKTER

Gunakan kolom CHAR atau KARAKTER untuk menyimpan string dengan panjang tetap. String ini dilapisi dengan blanko, sehingga kolom CHAR (10) selalu menempati 10 byte penyimpanan.

```
char(10)
```

Kolom CHAR tanpa spesifikasi panjang menghasilkan kolom CHAR (1).

VARCHAR atau KARAKTER BERVARIASI

Gunakan kolom VARCHAR atau CHARACTER VARY untuk menyimpan string panjang variabel dengan batas tetap. String ini tidak dilapisi dengan blanko, sehingga kolom VARCHAR (120) terdiri dari maksimum 120 karakter single-byte, 60 karakter dua-byte, 40 karakter tiga byte, atau 30 karakter empat byte.

```
varchar(120)
```

Signifikansi trailing blanko

Baik tipe data CHAR dan VARCHAR menyimpan string hingga n byte panjangnya. Upaya untuk menyimpan string yang lebih panjang ke dalam kolom jenis ini menghasilkan kesalahan. Namun, jika karakter tambahan adalah semua spasi (kosong), string terpotong hingga panjang maksimum. Jika string lebih pendek dari panjang maksimum, nilai CHAR dilapisi dengan kosong, tetapi nilai VARCHAR menyimpan string tanpa kosong.

Trailing blank dalam nilai CHAR selalu tidak signifikan secara semantik. Mereka diabaikan ketika Anda membandingkan dua nilai CHAR, tidak termasuk dalam perhitungan PANJANG, dan dihapus saat Anda mengonversi nilai CHAR ke tipe string lain.

Spasi trailing dalam nilai VARCHAR dan CHAR diperlakukan sebagai tidak signifikan secara semantik ketika nilai dibandingkan.

Perhitungan panjang mengembalikan panjang string karakter VARCHAR dengan spasi tambahan yang termasuk dalam panjangnya. Trailing blank tidak dihitung panjangnya untuk string karakter dengan panjang tetap.

Jenis Datetime

Tipe data datetime termasuk DATE, TIME, TIMETZ, TIMESTAMP, dan TIMESTAMPTZ.

Topik

Penyimpanan dan rentang

Nama	Penyimpanan	Kisaran	Penyelesaian
DATE	4 byte	4713 SM hingga 294276 M	1 hari
TIME	8 byte	00:00:00 hingga 24:00:00	1 mikrodetik
JADWAL	8 byte	00:00:00 +1459 hingga 00:00:00 +1459	1 mikrodetik

Nama	Penyimpanan	Kisaran	Penyelesaian
TIMESTAMP	8 byte	4713 SM hingga 294276 M	1 mikrodetik
TIMESTAMPTZ	8 byte	4713 SM hingga 294276 M	1 mikrodetik

DATE

Gunakan tipe data DATE untuk menyimpan tanggal kalender sederhana tanpa cap waktu.

TIME

Gunakan tipe data TIME untuk menyimpan waktu dalam sehari.

Kolom TIME menyimpan nilai hingga maksimum enam digit presisi untuk detik pecahan.

Secara default, nilai TIME adalah Coordinated Universal Time (UTC) di tabel pengguna dan tabel AWS Clean Rooms sistem.

JADWAL

Gunakan tipe data TIMETZ untuk menyimpan waktu hari dengan zona waktu.

Kolom TIMETZ menyimpan nilai hingga maksimum enam digit presisi untuk detik pecahan.

Secara default, nilai TIMETZ adalah UTC di kedua tabel pengguna dan tabel AWS Clean Rooms sistem.

TIMESTAMP

Gunakan tipe data TIMESTAMP untuk menyimpan nilai stempel waktu lengkap yang menyertakan tanggal dan waktu hari.

Kolom TIMESTAMP menyimpan nilai dengan presisi maksimal enam digit selama pecahan detik.

Jika Anda menyisipkan tanggal ke kolom TIMESTAMP, atau tanggal dengan nilai stempel waktu sebagian, nilai tersebut secara implisit diubah menjadi nilai stempel waktu penuh. Nilai stempel waktu penuh ini memiliki nilai default (00) untuk jam, menit, dan detik yang hilang. Nilai zona waktu dalam string input diabaikan.

Secara default, nilai TIMESTAMP adalah UTC di tabel pengguna dan AWS Clean Rooms tabel sistem.

TIMESTAMPTZ

Gunakan tipe data TIMESTAMPTZ untuk memasukkan nilai stempel waktu lengkap yang mencakup tanggal, waktu hari, dan zona waktu. Ketika nilai input mencakup zona waktu, AWS Clean Rooms gunakan zona waktu untuk mengonversi nilai ke UTC dan menyimpan nilai UTC.

Untuk melihat daftar nama zona waktu yang didukung, jalankan perintah berikut.

```
select my_timezone_names();
```

Untuk melihat daftar singkatan zona waktu yang didukung, jalankan perintah berikut.

```
select my_timezone_abbrevs();
```

Anda juga dapat menemukan informasi terkini tentang zona waktu di Database Zona Waktu IANA.

Tabel berikut memiliki contoh format zona waktu.

format	Contoh
dd mon hh:mi:ss yyyy tz	17 Des 07:37:16 1997 PST
mm/dd/yyyy hh: mi: ss.ss tz	12/17/1997 07:37:16.00 PST
mm/dd/yyyy hh: mi: ss.ss tz	12/17/1997 07:37:16.00 US/Pasifik
yyyy-mm-dd hh:mi: ss+/- tz	1997-12-17 07:37:16-08
dd.mm.yyyy hh:mi:ss tz	17.12.1997 07:37:16.00 PST

Kolom TIMESTAMPTZ menyimpan nilai hingga maksimum enam digit presisi untuk detik pecahan.

Jika Anda menyisipkan tanggal ke kolom TIMESTAMPTZ, atau tanggal dengan stempel waktu sebagian, nilainya secara implisit diubah menjadi nilai stempel waktu penuh. Nilai stempel waktu penuh ini memiliki nilai default (00) untuk jam, menit, dan detik yang hilang.

Nilai TIMESTAMPTZ adalah UTC dalam tabel pengguna.

Contoh dengan tipe datetime

Contoh berikut menunjukkan cara bekerja dengan tipe datetime yang didukung oleh. AWS Clean Rooms

Contoh tanggal

Contoh berikut menyisipkan tanggal yang memiliki format berbeda dan menampilkan output.

```
select * from datetable order by 1;

start_date |  end_date
-----------------------
2008-06-01 | 2008-12-31
2008-06-01 | 2008-12-31
```

Jika Anda memasukkan nilai stempel waktu ke kolom DATE, bagian waktu diabaikan dan hanya tanggal yang dimuat.

Contoh waktu

Contoh berikut menyisipkan TIME dan TIMETZ nilai yang memiliki format yang berbeda dan menampilkan output.

```
select * from timetable order by 1;
start_time |  end_time
------------------------
 19:11:19  | 20:41:19+00
 19:11:19  | 20:41:19+00
```

Contoh cap waktu

Jika Anda memasukkan tanggal ke kolom TIMESTAMP atau TIMESTAMPTZ, waktu defaultnya adalah tengah malam. Misalnya, jika Anda memasukkan literal`20081231`, nilai yang disimpan adalah`2008-12-31 00:00:00`.

Contoh berikut menyisipkan stempel waktu yang memiliki format berbeda dan menampilkan output.

```
timeofday
--------------------
2008-06-01 09:59:59
```

```
2008-12-31 18:20:00
(2 rows)
```

Tanggal, waktu, dan literal stempel waktu

Berikut ini adalah aturan untuk bekerja dengan literal tanggal, waktu, dan stempel waktu yang didukung oleh. AWS Clean Rooms

Tanggal

Tabel berikut menunjukkan tanggal masukan yang merupakan contoh valid dari nilai tanggal literal yang dapat Anda muat ke dalam AWS Clean Rooms tabel. `MDY DateStyle`Mode default diasumsikan berlaku. Mode ini berarti bahwa nilai bulan mendahului nilai hari dalam string seperti `1999-01-08` dan. `01/02/00`

> **Note**
>
> Tanggal atau stempel waktu literal harus dilampirkan dalam tanda kutip saat Anda memuatnya ke dalam tabel.

Tanggal masukan	Tanggal penuh
8 Januari 1999	8 Januari 1999
1999-01-08	8 Januari 1999
1/8/1999	8 Januari 1999
01/02/00	2 Januari 2000
2000-Jan-31	31 Januari 2000
Jan-31-2000	31 Januari 2000
31-Jan-2000	31 Januari 2000
20080215	15 Februari 2008
080215	15 Februari 2008

Tanggal masukan	Tanggal penuh
2008.366	31 Desember 2008 (bagian tiga digit tanggal harus antara 001 dan 366)

Kali

Tabel berikut menunjukkan waktu masukan yang merupakan contoh valid dari nilai waktu literal yang dapat Anda muat ke dalam AWS Clean Rooms tabel.

Waktu masukan	Deskripsi (bagian waktu)
04:05:06.789	4:05 AM dan 6.789 detik
04:05:06	4:05 AM dan 6 detik
04:05	4:05 AM tepatnya
040506	4:05 AM dan 6 detik
04:05AM	4:05 AM tepatnya; AM adalah opsional
04:05 SORE	4:05 PM tepatnya; nilai jam harus kurang dari 12
16:05	16:05 PM tepatnya

Stempel waktu

Tabel berikut menunjukkan cap waktu masukan yang merupakan contoh valid dari nilai waktu literal yang dapat Anda muat ke dalam AWS Clean Rooms tabel. Semua literal tanggal yang valid dapat digabungkan dengan literal waktu berikut.

Masukan stempel waktu (tanggal dan waktu gabungan)	Deskripsi (bagian waktu)
20080215 04:05:06.789	4:05 AM dan 6.789 detik

Masukan stempel waktu (tanggal dan waktu gabungan)	Deskripsi (bagian waktu)
20080215 04:05:06	4:05 AM dan 6 detik
20080215 04:05	4:05 AM tepatnya
20080215 040506	4:05 AM dan 6 detik
20080215 04:05AM	4:05 AM tepatnya; AM adalah opsional
20080215 04:05PM	4:05 PM tepatnya; nilai jam harus kurang dari 12
20080215 16:05	16:05 PM tepatnya
20080215	Tengah malam (secara default)

Nilai datetime khusus

Tabel berikut menunjukkan nilai-nilai khusus yang dapat digunakan sebagai literal datetime dan sebagai argumen untuk fungsi tanggal. Mereka membutuhkan tanda kutip tunggal dan dikonversi ke nilai stempel waktu biasa selama pemrosesan kueri.

Nilai khusus	Deskripsi
`now`	Mengevaluasi waktu mulai transaksi saat ini dan mengembalikan stempel waktu dengan presisi mikrodetik.
`today`	Mengevaluasi ke tanggal yang sesuai dan mengembalikan stempel waktu dengan nol untuk bagian waktu.
`tomorrow`	Mengevaluasi ke tanggal yang sesuai dan mengembalikan stempel waktu dengan nol untuk bagian waktu.

Nilai khusus	Deskripsi
`yesterday`	Mengevaluasi ke tanggal yang sesuai dan mengembalikan stempel waktu dengan nol untuk bagian waktu.

Contoh berikut menunjukkan bagaimana now dan `today` bekerja dengan fungsi DATEADD.

```
select dateadd(day,1,'today');

date_add
---------------------
2009-11-17 00:00:00
(1 row)

select dateadd(day,1,'now');

date_add
----------------------------
2009-11-17 10:45:32.021394
(1 row)
```

Literal interval

Berikut ini adalah aturan untuk bekerja dengan literal interval yang didukung oleh AWS Clean Rooms.

Gunakan interval literal untuk mengidentifikasi periode waktu tertentu, seperti `12 hours` atau`6 weeks`. Anda dapat menggunakan literal interval ini dalam kondisi dan perhitungan yang melibatkan ekspresi datetime.

> **Note**
>
> Anda tidak dapat menggunakan tipe data INTERVAL untuk kolom dalam AWS Clean Rooms tabel.

Interval dinyatakan sebagai kombinasi kata kunci INTERVAL dengan kuantitas numerik dan bagian tanggal yang didukung, misalnya `INTERVAL '7 days'` atau`INTERVAL '59 minutes'`. Anda dapat menghubungkan beberapa kuantitas dan unit untuk membentuk interval yang lebih tepat,

misalnya:`INTERVAL '7 days, 3 hours, 59 minutes'`. Singkatan dan bentuk jamak dari setiap unit juga didukung; misalnya:`5 s`,`5 second`, dan `5 seconds` merupakan interval yang setara.

Jika Anda tidak menentukan bagian tanggal, nilai interval mewakili detik. Anda dapat menentukan nilai kuantitas sebagai pecahan (misalnya:`0.5 days`).

Contoh-contoh

Contoh berikut menunjukkan serangkaian perhitungan dengan nilai interval yang berbeda.

Contoh berikut menambahkan 1 detik ke tanggal yang ditentukan.

```
select caldate + interval '1 second' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 00:00:01
(1 row)
```

Contoh berikut menambahkan 1 menit ke tanggal yang ditentukan.

```
select caldate + interval '1 minute' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 00:01:00
(1 row)
```

Contoh berikut menambahkan 3 jam dan 35 menit ke tanggal yang ditentukan.

```
select caldate + interval '3 hours, 35 minutes' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 03:35:00
(1 row)
```

Contoh berikut menambahkan 52 minggu ke tanggal yang ditentukan.

```
select caldate + interval '52 weeks' as dateplus from date
```

```
where caldate='12-31-2008';
dateplus
---------------------
2009-12-30 00:00:00
(1 row)
```

Contoh berikut menambahkan 1 minggu, 1 jam, 1 menit, dan 1 detik ke tanggal yang ditentukan.

```
select caldate + interval '1w, 1h, 1m, 1s' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2009-01-07 01:01:01
(1 row)
```

Contoh berikut menambahkan 12 jam (setengah hari) ke tanggal yang ditentukan.

```
select caldate + interval '0.5 days' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 12:00:00
(1 row)
```

Contoh berikut mengurangi 4 bulan dari 15 Februari 2023 dan hasilnya adalah 15 Oktober 2022.

```
select date '2023-02-15' - interval '4 months';

?column?
---------------------
2022-10-15 00:00:00
```

Contoh berikut mengurangi 4 bulan dari 31 Maret 2023 dan hasilnya adalah 30 November 2022. Perhitungan mempertimbangkan jumlah hari dalam sebulan.

```
select date '2023-03-31' - interval '4 months';

?column?
---------------------
2022-11-30 00:00:00
```

Jenis Boolean

Gunakan tipe data BOOLEAN untuk menyimpan nilai true dan false dalam kolom single-byte. Tabel berikut menjelaskan tiga kemungkinan status untuk nilai Boolean dan nilai literal yang menghasilkan keadaan itu. Terlepas dari string input, kolom Boolean menyimpan dan mengeluarkan “t” untuk true dan “f” untuk false.

Status	Nilai literal yang valid	Penyimpanan
True	`TRUE 't' 'true' 'y' 'yes' '1'`	1 byte
False	`FALSE 'f' 'false' 'n' 'no' '0'`	1 byte
Tidak Diketahui	`NULL`	1 byte

Anda dapat menggunakan perbandingan IS untuk memeriksa nilai Boolean hanya sebagai predikat dalam klausa WHERE. Anda tidak dapat menggunakan perbandingan IS dengan nilai Boolean dalam daftar SELECT.

Contoh-contoh

Anda dapat menggunakan kolom BOOLEAN untuk menyimpan status “Aktif/Tidak Aktif” untuk setiap pelanggan dalam tabel PELANGGAN.

```
select * from customer;
custid | active_flag
-------+-------------
   100 | t
```

Dalam contoh ini, kueri berikut memilih pengguna dari tabel USERS yang menyukai olahraga tetapi tidak menyukai teater:

```
select firstname, lastname, likesports, liketheatre
from users
```

```
where likesports is true and liketheatre is false
order by userid limit 10;

firstname |  lastname  | likesports | liketheatre
----------+------------+------------+-------------
Alejandro | Rosalez    | t          | f
Akua      | Mansa      | t          | f
Arnav     | Desai      | t          | f
Carlos    | Salazar    | t          | f
Diego     | Ramirez    | t          | f
Efua      | Owusu      | t          | f
John      | Stiles     | t          | f
Jorge     | Souza      | t          | f
Kwaku     | Mensah     | t          | f
Kwesi     | Manu       | t          | f
(10 rows)
```

Contoh berikut memilih pengguna dari tabel USERS yang tidak diketahui apakah mereka menyukai musik rock.

```
select firstname, lastname, likerock
from users
where likerock is unknown
order by userid limit 10;

firstname | lastname | likerock
----------+----------+----------
Alejandro | Rosalez  |
Carlos    | Salazar  |
Diego     | Ramirez  |
John      | Stiles   |
Kwaku     | Mensah   |
Martha    | Rivera   |
Mateo     | Jackson  |
Paulo     | Santos   |
Richard   | Roe      |
Saanvi    | Sarkar   |
(10 rows)
```

Contoh berikut mengembalikan kesalahan karena menggunakan perbandingan IS dalam daftar SELECT.

```
select firstname, lastname, likerock is true as "check"
```

```
from users
order by userid limit 10;

[Amazon](500310) Invalid operation: Not implemented
```

Contoh berikut berhasil karena menggunakan perbandingan yang sama (=) dalam daftar SELECT alih-alih IS perbandingan.

```
select firstname, lastname, likerock = true as "check"
from users
order by userid limit 10;

firstname | lastname  | check
----------+-----------+------
Alejandro | Rosalez   |
Carlos    | Salazar   |
Diego     | Ramirez   | true
John      | Stiles    |
Kwaku     | Mensah    | true
Martha    | Rivera    | true
Mateo     | Jackson   |
Paulo     | Santos    | false
Richard   | Roe       |
Saanvi    | Sarkar    |
```

Tipe SUPER

Gunakan tipe data SUPER untuk menyimpan data atau dokumen semi-terstruktur sebagai nilai.

Data semi-terstruktur tidak sesuai dengan struktur kaku dan tabular dari model data relasional yang digunakan dalam database SQL. Tipe data SUPER berisi tag yang mereferensikan entitas berbeda dalam data. Tipe data SUPER dapat berisi nilai kompleks seperti array, struktur bersarang, dan struktur kompleks lainnya yang terkait dengan format serialisasi, seperti JSON. Tipe data SUPER adalah seperangkat array tanpa skema dan nilai struktur yang mencakup semua jenis skalar lainnya. AWS Clean Rooms

Tipe data SUPER mendukung hingga 1 MB data untuk bidang atau objek SUPER individu.

Tipe data SUPER memiliki properti berikut:

- Nilai AWS Clean Rooms skalar:

  - Sebuah null
  - Sebuah boolean
  - Angka, seperti smallint, integer, bigint, desimal, atau floating point (seperti float4 atau float8)
  - Nilai string, seperti varchar atau char
- Nilai yang kompleks:
  - Array nilai, termasuk skalar atau kompleks
  - Struktur, juga dikenal sebagai tuple atau objek, yang merupakan peta nama dan nilai atribut (skalar atau kompleks)

Salah satu dari dua jenis nilai kompleks mengandung skalar atau nilai kompleksnya sendiri tanpa batasan keteraturan.

Tipe data SUPER mendukung persistensi data semi-terstruktur dalam bentuk skema. Meskipun model data hierarkis dapat berubah, versi data lama dapat hidup berdampingan di kolom SUPER yang sama.

Jenis bersarang

AWS Clean Rooms mendukung kueri yang melibatkan data dengan tipe data bersarang, khususnya tipe kolom AWS Glue struct, array, dan peta. Hanya aturan analisis kustom yang mendukung tipe data bersarang.

Khususnya, tipe data bersarang tidak sesuai dengan struktur tabel yang kaku dari model data relasional database SQL.

Tipe data bersarang berisi tag yang mereferensikan entitas berbeda dalam data. Mereka dapat berisi nilai-nilai kompleks seperti array, struktur bersarang, dan struktur kompleks lainnya yang terkait dengan format serialisasi, seperti JSON. Tipe data bersarang mendukung hingga 1 MB data untuk bidang atau objek tipe data bersarang individu.

Contoh tipe data bersarang

Untuk `struct<given:varchar, family:varchar>` tipe, ada dua nama atribut:`given`, dan`family`, masing-masing sesuai dengan `varchar` nilai.

Untuk `array<varchar>` tipe, array ditentukan sebagai daftar`varchar`.

`array<struct<shipdate:timestamp, price:double>>`Tipe mengacu pada daftar elemen dengan `struct<shipdate:timestamp, price:double>` tipe.

Tipe `map` data berperilaku seperti `array` dari`structs`, di mana nama atribut untuk setiap elemen dalam array dilambangkan dengan `key` dan dipetakan ke a. `value`

Example

Misalnya, `map<varchar(20), varchar(20)>` tipe diperlakukan sebagai`array<struct<key:varchar(20), value:varchar(20)>>`, di mana `key` dan `value` merujuk ke atribut peta dalam data yang mendasarinya.

Untuk informasi tentang cara AWS Clean Rooms mengaktifkan navigasi ke dalam array dan struktur, lihatNavigasi.

Untuk informasi tentang cara AWS Clean Rooms mengaktifkan iterasi melalui array dengan menavigasi array menggunakan klausa FROM dari kueri, lihat. Kueri yang tidak bersarang

Jenis VARBYTE

Gunakan kolom VARBYTE, VARBINARY, atau BINARY VARY VARY untuk menyimpan nilai biner panjang variabel dengan batas tetap.

```
varbyte [ (n) ]
```

Jumlah maksimum byte (n) dapat berkisar antara 1 — 1.024.000. Defaultnya adalah 64.000.

Beberapa contoh di mana Anda mungkin ingin menggunakan tipe data VARBYTE adalah sebagai berikut:

- Menggabungkan tabel pada kolom VARBYTE.
- Membuat tampilan terwujud yang berisi kolom VARBYTE. Penyegaran inkremental tampilan terwujud yang berisi kolom VARBYTE didukung. Namun, fungsi agregat selain COUNT, MIN, dan MAX dan GROUP BY pada kolom VARBYTE tidak mendukung penyegaran tambahan.

Untuk memastikan bahwa semua byte adalah karakter yang dapat dicetak, AWS Clean Rooms gunakan format hex untuk mencetak nilai VARBYTE. Misalnya, SQL berikut mengubah string heksadesimal menjadi nilai biner. 6162 Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal6162.

```
select from_hex('6162');

 from_hex
----------
 6162
```

AWS Clean Rooms mendukung casting antara VARBYTE dan tipe data berikut:

- CHAR
- VARCHAR
- SMALLINT
- INTEGER
- BIGINT

Pernyataan SQL berikut melemparkan string VARCHAR ke VARBYTE. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal616263.

```
select 'abc'::varbyte;

 varbyte
---------
 616263
```

Pernyataan SQL berikut memberikan nilai CHAR dalam kolom ke VARBYTE. Contoh ini membuat tabel dengan kolom CHAR (10) (c), menyisipkan nilai karakter yang lebih pendek dari panjang 10. Cast yang dihasilkan melapisi hasil dengan karakter spasi (hex'20') ke ukuran kolom yang ditentukan. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal.

```
create table t (c char(10));
insert into t values ('aa'), ('abc');
select c::varbyte from t;
          c
----------------------
 6161202020202020202020
 6162632020202020202020
```

Pernyataan SQL berikut melemparkan string SMALLINT ke VARBYTE. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal`0005`, yang merupakan dua byte atau empat karakter heksadesimal.

```
select 5::smallint::varbyte;

 varbyte
---------
 0005
```

Pernyataan SQL berikut melemparkan INTEGER ke VARBYTE. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal`00000005`, yaitu empat byte atau delapan karakter heksadesimal.

```
select 5::int::varbyte;

 varbyte
----------
 00000005
```

Pernyataan SQL berikut melemparkan BIGINT ke VARBYTE. Meskipun nilai yang dikembalikan adalah nilai biner, hasilnya dicetak sebagai heksadesimal`0000000000000005`, yaitu delapan byte atau 16 karakter heksadesimal.

```
select 5::bigint::varbyte;

     varbyte
------------------
 0000000000000005
```

Keterbatasan saat menggunakan tipe data VARBYTE dengan AWS Clean Rooms

Berikut ini adalah batasan saat menggunakan tipe data VARBYTE dengan AWS Clean Rooms:

- AWS Clean Rooms mendukung tipe data VARBYTE hanya untuk file Parket dan ORC.
- AWS Clean Rooms editor kueri belum sepenuhnya mendukung tipe data VARBYTE. Oleh karena itu, gunakan klien SQL yang berbeda saat bekerja dengan ekspresi VARBYTE.

  Sebagai solusi untuk menggunakan editor kueri, jika panjang data Anda di bawah 64 KB dan kontennya valid UTF-8, Anda dapat mentransmisikan nilai VARBYTE ke VARCHAR, misalnya:

```
select to_varbyte('6162', 'hex')::varchar;
```

- Anda tidak dapat menggunakan tipe data VARBYTE dengan Python atau Lambda yang ditentukan pengguna (UDF).
- Anda tidak dapat membuat kolom HLLSKETCH dari kolom VARBYTE atau menggunakan PERKIRAAN COUNT DISTINCT pada kolom VARBYTE.

Ketik kompatibilitas dan konversi

Diskusi berikut menjelaskan cara kerja aturan konversi tipe dan kompatibilitas tipe data AWS Clean Rooms.

Kompatibilitas

Pencocokan tipe data dan pencocokan nilai literal dan konstanta dengan tipe data terjadi selama berbagai operasi database, termasuk yang berikut:

- Operasi bahasa manipulasi data (DHTML) pada tabel
- UNION, INTERSECT, dan EXCEPT query
- Ekspresi CASE
- Evaluasi predikat, seperti LIKE dan IN
- Evaluasi fungsi SQL yang melakukan perbandingan atau ekstraksi data
- Perbandingan dengan operator matematika

Hasil operasi ini bergantung pada aturan konversi tipe dan kompatibilitas tipe data. Kompatibilitas menyiratkan bahwa one-to-one pencocokan nilai tertentu dan tipe data tertentu tidak selalu diperlukan. Karena beberapa tipe data kompatibel, konversi implisit, atau paksaan, dimungkinkan. Untuk informasi selengkapnya, lihat Jenis konversi implisit. Ketika tipe data tidak kompatibel, terkadang Anda dapat mengonversi nilai dari satu tipe data ke tipe data lainnya dengan menggunakan fungsi konversi eksplisit.

Kompatibilitas umum dan aturan konversi

Perhatikan aturan kompatibilitas dan konversi berikut:

- Secara umum, tipe data yang termasuk dalam kategori tipe yang sama (seperti tipe data numerik yang berbeda) kompatibel dan dapat dikonversi secara implisit.

  Misalnya, dengan konversi implisit Anda dapat menyisipkan nilai desimal ke dalam kolom integer. Desimal dibulatkan untuk menghasilkan bilangan bulat. Atau Anda dapat mengekstrak nilai numerik, seperti`2008`, dari tanggal dan memasukkan nilai itu ke dalam kolom integer.

- Tipe data numerik memberlakukan kondisi luapan yang terjadi saat Anda mencoba menyisipkan nilai. out-of-range Misalnya, nilai desimal dengan presisi 5 tidak cocok dengan kolom desimal yang didefinisikan dengan presisi 4. Sebuah integer atau seluruh bagian dari desimal tidak pernah terpotong. Namun, bagian pecahan desimal dapat dibulatkan ke atas atau ke bawah, sebagaimana mestinya. Namun, hasil pemeran eksplisit nilai yang dipilih dari tabel tidak dibulatkan.

- Berbagai jenis string karakter kompatibel. String kolom VARCHAR yang berisi data byte tunggal dan string kolom CHAR sebanding dan dapat dikonversi secara implisit. String VARCHAR yang berisi data multibyte tidak sebanding. Selain itu, Anda dapat mengonversi string karakter ke tanggal, waktu, stempel waktu, atau nilai numerik jika string adalah nilai literal yang sesuai. Setiap spasi utama atau belakang diabaikan. Sebaliknya, Anda dapat mengonversi tanggal, waktu, stempel waktu, atau nilai numerik menjadi string karakter dengan panjang tetap atau panjang variabel.

  > **Note**
  >
  > String karakter yang ingin Anda transmisikan ke tipe numerik harus berisi representasi karakter angka. Misalnya, Anda dapat mentransmisikan string `'1.0'` atau `'5.9'` ke nilai desimal, tetapi Anda tidak dapat mentransmisikan string `'ABC'` ke jenis numerik apa pun.

- Jika Anda membandingkan nilai DECIMAL dengan string karakter, AWS Clean Rooms mencoba untuk mengubah string karakter ke nilai DECIMAL. Saat membandingkan semua nilai numerik lainnya dengan string karakter, nilai numerik dikonversi ke string karakter. Untuk menegakkan konversi yang berlawanan (misalnya, mengubah string karakter menjadi bilangan bulat, atau mengubah nilai DECIMAL menjadi string karakter), gunakan fungsi eksplisit, seperti. Fungsi CAST

- Untuk mengonversi nilai DECIMAL atau NUMERIK 64-bit ke presisi yang lebih tinggi, Anda harus menggunakan fungsi konversi eksplisit seperti fungsi CAST atau CONVERT.

- Saat mengonversi DATE atau TIMESTAMP ke TIMESTAMPTZ, atau mengonversi TIME ke TIMETZ, zona waktu diatur ke zona waktu sesi saat ini. Zona waktu sesi adalah UTC secara default.

- Demikian pula, TIMESTAMPTZ dikonversi ke DATE, TIME, atau TIMESTAMP berdasarkan zona waktu sesi saat ini. Zona waktu sesi adalah UTC secara default. Setelah konversi, informasi zona waktu dijatuhkan.
- String karakter yang mewakili stempel waktu dengan zona waktu yang ditentukan dikonversi ke TIMESTAMPTZ menggunakan zona waktu sesi saat ini, yang merupakan UTC secara default. Demikian juga, string karakter yang mewakili waktu dengan zona waktu yang ditentukan dikonversi ke TIMETZ menggunakan zona waktu sesi saat ini, yang merupakan UTC secara default.

Jenis konversi implisit

Ada dua jenis konversi implisit:

- Konversi implisit dalam tugas, seperti menyetel nilai dalam perintah INSERT atau UPDATE
- Konversi implisit dalam ekspresi, seperti melakukan perbandingan dalam klausa WHERE

Tabel berikut mencantumkan tipe data yang dapat dikonversi secara implisit dalam tugas atau ekspresi. Anda juga dapat menggunakan fungsi konversi eksplisit untuk melakukan konversi ini.

Dari tipe	Untuk mengetik
BIGINT	BOOLEAN
	CHAR
	DESIMAL (NUMERIK)
	PRESISI GANDA (FLOAT8)
	INTEGER
	NYATA (FLOAT4)
	SMALLINT
	VARCHAR
CHAR	VARCHAR
DATE	CHAR

Dari tipe	Untuk mengetik
	VARCHAR
	TIMESTAMP
	TIMESTAMPTZ
DESIMAL (NUMERIK)	BIGINT
	CHAR
	PRESISI GANDA (FLOAT8)
	INTEGER INT)
	NYATA (FLOAT4)
	SMALLINT
	VARCHAR
PRESISI GANDA (FLOAT8)	BIGINT
	CHAR
	DESIMAL (NUMERIK)
	BILANGAN BULAT (INT)
	NYATA (FLOAT4)
	SMALLINT
	VARCHAR
BILANGAN BULAT (INT)	BIGINT
	BOOLEAN
	CHAR

Dari tipe	Untuk mengetik
	DESIMAL (NUMERIK)
	PRESISI GANDA (FLOAT8)
	NYATA (FLOAT4)
	SMALLINT
	VARCHAR
NYATA (FLOAT4)	BIGINT
	CHAR
	DESIMAL (NUMERIK)
	BILANGAN BULAT (INT)
	SMALLINT
	VARCHAR
SMALLINT	BIGINT
	BOOLEAN
	CHAR
	DESIMAL (NUMERIK)
	PRESISI GANDA (FLOAT8)
	BILANGAN BULAT (INT)
	NYATA (FLOAT4)
	VARCHAR
TIMESTAMP	CHAR

Dari tipe	Untuk mengetik
	DATE
	VARCHAR
	TIMESTAMPTZ
	TIME
TIMESTAMPTZ	CHAR
	DATE
	VARCHAR
	TIMESTAMP
	JADWAL
TIME	VARCHAR
	JADWAL
JADWAL	VARCHAR
	TIME

Note

Konversi implisit antara TIMESTAMPTZ, TIMESTAMP, DATE, TIME, TIMETZ, atau string karakter menggunakan zona waktu sesi saat ini.
Tipe data VARBYTE tidak dapat secara implisit dikonversi ke tipe data lainnya. Untuk informasi selengkapnya, lihat Fungsi CAST.

Perintah SQL di AWS Clean Rooms

Perintah SQL berikut didukung di AWS Clean Rooms:

Topik

SELECT

Perintah SELECT mengembalikan baris dari tabel dan fungsi yang ditentukan pengguna.

Perintah SELECT SQL berikut didukung di AWS Clean Rooms:

Topik

SELECT list

SELECT listNama-nama kolom, fungsi, dan ekspresi yang Anda ingin kueri untuk kembali. Daftar ini mewakili output kueri.

Sintaksis

```
SELECT
```

```
[ TOP number ]
[ DISTINCT ] | expression [ AS column_alias ] [, ...]
```

Parameter-parameter

TOP *nomor*

TOPmengambil bilangan bulat positif sebagai argumennya, yang mendefinisikan jumlah baris yang dikembalikan ke klien. Perilaku dengan TOP klausa sama dengan perilaku dengan LIMIT klausa. Jumlah baris yang dikembalikan adalah tetap, tetapi himpunan baris tidak tetap. Untuk mengembalikan serangkaian baris yang konsisten, gunakan TOP atau LIMIT bersama dengan ORDER BY klausa.

DISTINCT

Opsi yang menghilangkan baris duplikat dari set hasil, berdasarkan nilai yang cocok dalam satu atau beberapa kolom.

ekspresi

Ekspresi yang terbentuk dari satu atau lebih kolom yang ada di tabel yang direferensikan oleh kueri. Ekspresi dapat berisi fungsi SQL. Sebagai contoh:

```
coalesce(dimension, 'stringifnull') AS column_alias
```

AS column_alias

Nama sementara untuk kolom yang digunakan dalam set hasil akhir. ASKata kunci adalah opsional. Sebagai contoh:

```
coalesce(dimension, 'stringifnull') AS dimensioncomplete
```

Jika Anda tidak menentukan alias untuk ekspresi yang bukan nama kolom sederhana, set hasil akan menerapkan nama default ke kolom tersebut.

> **Note**
>
> Alias dikenali tepat setelah didefinisikan dalam daftar target. Anda tidak dapat menggunakan alias dalam ekspresi lain yang ditentukan setelahnya dalam daftar target yang sama.

Catatan penggunaan

TOPadalah ekstensi SQL. TOPmemberikan alternatif untuk LIMIT perilaku. Anda tidak dapat menggunakan TOP dan LIMIT dalam kueri yang sama.

DENGAN klausa

Klausa WITH adalah klausa opsional yang mendahului daftar SELECT dalam kueri. Klausa WITH mendefinisikan satu atau lebih common_table_expressions. Setiap ekspresi tabel umum (CTE) mendefinisikan tabel sementara, yang mirip dengan definisi tampilan. Anda dapat mereferensikan tabel sementara ini di klausa FROM. Mereka hanya digunakan saat kueri milik mereka berjalan. Setiap CTE dalam klausa WITH menentukan nama tabel, daftar opsional nama kolom, dan ekspresi kueri yang mengevaluasi tabel (pernyataan SELECT).

Dengan subquery klausa adalah cara yang efisien untuk mendefinisikan tabel yang dapat digunakan selama eksekusi query tunggal. Dalam semua kasus, hasil yang sama dapat dicapai dengan menggunakan subquery di bagian utama pernyataan SELECT, tetapi dengan subquery klausa mungkin lebih mudah untuk ditulis dan dibaca. Jika memungkinkan, subkueri klausa WITH yang direferensikan beberapa kali dioptimalkan sebagai subexpressions umum; yaitu, dimungkinkan untuk mengevaluasi subquery WITH sekali dan menggunakan kembali hasilnya. (Perhatikan bahwa subexpressions umum tidak terbatas pada yang didefinisikan dalam klausa WITH.)

Sintaks

```
[ WITH  common_table_expression [, common_table_expression , ...] ]
```

dimana common_table_expression bisa non-rekursif. Berikut ini adalah bentuk non-rekursif:

```
CTE_table_name AS ( query )
```

Parameter

common_table_expression

Mendefinisikan tabel sementara yang dapat Anda referensikan di Klausa FROM dan hanya digunakan selama eksekusi kueri yang dimilikinya.

CTE_TABLE_NAME

Nama unik untuk tabel sementara yang mendefinisikan hasil subquery klausa WITH. Anda tidak dapat menggunakan nama duplikat dalam satu klausa WITH. Setiap subquery harus diberi nama tabel yang dapat direferensikan di. Klausa FROM

kueri

Setiap kueri SELECT yang AWS Clean Rooms mendukung. Lihat SELECT.

Catatan penggunaan

Anda dapat menggunakan klausa WITH dalam pernyataan SQL berikut:

- PILIH, DENGAN, UNION, INTERSECT, dan KECUALI

Jika klausa FROM dari kueri yang berisi klausa WITH tidak mereferensikan salah satu tabel yang ditentukan oleh klausa WITH, klausa WITH diabaikan dan kueri berjalan seperti biasa.

Sebuah tabel yang didefinisikan oleh subquery klausa WITH dapat direferensikan hanya dalam lingkup kueri SELECT bahwa klausa WITH dimulai. Misalnya, Anda dapat mereferensikan tabel tersebut dalam klausa FROM dari subquery dalam daftar SELECT, klausa WHERE, atau HAVING. Anda tidak dapat menggunakan klausa WITH dalam subquery dan mereferensikan tabelnya di klausa FROM dari kueri utama atau subquery lainnya. Pola kueri ini menghasilkan pesan kesalahan formulir `relation table_name doesn't exist` untuk tabel klausa WITH.

Anda tidak dapat menentukan klausa WITH lain di dalam subquery klausa WITH.

Anda tidak dapat meneruskan referensi ke tabel yang ditentukan oleh subkueri klausa WITH. Misalnya, query berikut mengembalikan kesalahan karena referensi forward ke tabel W2 dalam definisi tabel W1:

```
with w1 as (select * from w2), w2 as (select * from w1)
select * from sales;
ERROR:  relation "w2" does not exist
```

Contoh-contoh

Contoh berikut menunjukkan kasus yang paling sederhana dari query yang berisi klausa WITH. Query WITH bernama VENUECOPY memilih semua baris dari tabel VENUE. Kueri utama pada

gilirannya memilih semua baris dari VENUECOPY. Tabel VENUECOPY hanya ada selama durasi kueri ini.

```
with venuecopy as (select * from venue)
select * from venuecopy order by 1 limit 10;
```

```
 venueid |         venuename          |    venuecity    | venuestate | venueseats
---------+----------------------------+-----------------+------------+------------
1 | Toyota Park                | Bridgeview      | IL         |          0
2 | Columbus Crew Stadium      | Columbus        | OH         |          0
3 | RFK Stadium                | Washington      | DC         |          0
4 | CommunityAmerica Ballpark  | Kansas City     | KS         |          0
5 | Gillette Stadium           | Foxborough      | MA         |      68756
6 | New York Giants Stadium    | East Rutherford | NJ         |      80242
7 | BMO Field                  | Toronto         | ON         |          0
8 | The Home Depot Center      | Carson          | CA         |          0
9 | Dick's Sporting Goods Park | Commerce City   | CO         |          0
v     10 | Pizza Hut Park             | Frisco          | TX         |          0
(10 rows)
```

Contoh berikut menunjukkan klausa WITH yang menghasilkan dua tabel, bernama VENUE_SALES dan TOP_VENUES. Tabel WITH query kedua memilih dari yang pertama. Pada gilirannya, klausa WHERE dari blok kueri utama berisi subquery yang membatasi tabel TOP_VENUES.

```
with venue_sales as
(select venuename, venuecity, sum(pricepaid) as venuename_sales
from sales, venue, event
where venue.venueid=event.venueid and event.eventid=sales.eventid
group by venuename, venuecity),

top_venues as
(select venuename
from venue_sales
where venuename_sales > 800000)

select venuename, venuecity, venuestate,
sum(qtysold) as venue_qty,
sum(pricepaid) as venue_sales
from sales, venue, event
where venue.venueid=event.venueid and event.eventid=sales.eventid
and venuename in(select venuename from top_venues)
group by venuename, venuecity, venuestate
```

```
order by venuename;
```

```
        venuename       |   venuecity   | venuestate | venue_qty | venue_sales
------------------------+---------------+------------+-----------+-------------
August Wilson Theatre   | New York City | NY         |      3187 |  1032156.00
Biltmore Theatre        | New York City | NY         |      2629 |   828981.00
Charles Playhouse       | Boston        | MA         |      2502 |   857031.00
Ethel Barrymore Theatre | New York City | NY         |      2828 |   891172.00
Eugene O'Neill Theatre  | New York City | NY         |      2488 |   828950.00
Greek Theatre           | Los Angeles   | CA         |      2445 |   838918.00
Helen Hayes Theatre     | New York City | NY         |      2948 |   978765.00
Hilton Theatre          | New York City | NY         |      2999 |   885686.00
Imperial Theatre        | New York City | NY         |      2702 |   877993.00
Lunt-Fontanne Theatre   | New York City | NY         |      3326 |  1115182.00
Majestic Theatre        | New York City | NY         |      2549 |   894275.00
Nederlander Theatre     | New York City | NY         |      2934 |   936312.00
Pasadena Playhouse      | Pasadena      | CA         |      2739 |   820435.00
Winter Garden Theatre   | New York City | NY         |      2838 |   939257.00
(14 rows)
```

Dua contoh berikut menunjukkan aturan untuk ruang lingkup referensi tabel berdasarkan subquery klausa WITH. Kueri pertama berjalan, tetapi yang kedua gagal dengan kesalahan yang diharapkan. Kueri pertama memiliki subquery klausa WITH di dalam daftar SELECT dari kueri utama. Tabel yang ditentukan oleh klausa WITH (HOLIDAYS) direferensikan dalam klausa FROM subquery dalam daftar SELECT:

```
select caldate, sum(pricepaid) as daysales,
(with holidays as (select * from date where holiday ='t')
select sum(pricepaid)
from sales join holidays on sales.dateid=holidays.dateid
where caldate='2008-12-25') as dec25sales
from sales join date on sales.dateid=date.dateid
where caldate in('2008-12-25','2008-12-31')
group by caldate
order by caldate;

caldate   | daysales | dec25sales
----------+----------+------------
2008-12-25 | 70402.00 |   70402.00
2008-12-31 | 12678.00 |   70402.00
(2 rows)
```

Kueri kedua gagal karena mencoba mereferensikan tabel HOLIDAYS di kueri utama serta dalam subquery daftar SELECT. Referensi kueri utama berada di luar cakupan.

```
select caldate, sum(pricepaid) as daysales,
(with holidays as (select * from date where holiday ='t')
select sum(pricepaid)
from sales join holidays on sales.dateid=holidays.dateid
where caldate='2008-12-25') as dec25sales
from sales join holidays on sales.dateid=holidays.dateid
where caldate in('2008-12-25','2008-12-31')
group by caldate
order by caldate;

ERROR:  relation "holidays" does not exist
```

Klausa FROM

Klausa FROM dalam kueri mencantumkan referensi tabel (tabel, tampilan, dan subkueri) tempat data dipilih. Jika beberapa referensi tabel terdaftar, tabel harus digabungkan, menggunakan sintaks yang sesuai baik dalam klausa FROM atau klausa WHERE. Jika tidak ada kriteria gabungan yang ditentukan, sistem memproses kueri sebagai cross-join (produk Cartesian).

Topik

- Sintaks
- Parameter
- Catatan penggunaan
- JOIN contoh

Sintaks

```
FROM table_reference [, ...]
```

di mana table_reference adalah salah satu dari berikut ini:

```
with_subquery_table_name | table_name | ( subquery ) [ [ AS ] alias ]
table_reference [ NATURAL ] join_type table_reference [ USING ( join_column [, ...] ) ]
table_reference [ INNER ] join_type table_reference ON expr
```

Parameter

dengan_subquery_table_name

Sebuah tabel didefinisikan oleh subquery di. DENGAN klausa

table_name

Nama tabel atau tampilan.

alias

Nama alternatif sementara untuk tabel atau tampilan. Alias harus disediakan untuk tabel yang berasal dari subquery. Dalam referensi tabel lainnya, alias bersifat opsional. Kata AS kunci selalu opsional. Alias tabel menyediakan pintasan yang nyaman untuk mengidentifikasi tabel di bagian lain dari kueri, seperti klausa WHERE.

Sebagai contoh:

```
select * from sales s, listing l
where s.listid=l.listid
```

Jika Anda mendefinisikan alias tabel didefinisikan, maka alias harus digunakan untuk referensi tabel dalam query.

Misalnya, jika kueri adalah`SELECT "tbl"."col" FROM "tbl" AS "t"`, kueri akan gagal karena nama tabel pada dasarnya diganti sekarang. Kueri yang valid dalam kasus ini adalah`SELECT "t"."col" FROM "tbl" AS "t"`.

column_alias

Nama alternatif sementara untuk kolom dalam tabel atau tampilan.

subkueri

Ekspresi kueri yang mengevaluasi ke tabel. Tabel hanya ada selama durasi kueri dan biasanya diberi nama atau alias. Namun, alias tidak diperlukan. Anda juga dapat menentukan nama kolom untuk tabel yang berasal dari subquery. Penamaan alias kolom penting saat Anda ingin menggabungkan hasil subkueri ke tabel lain dan saat Anda ingin memilih atau membatasi kolom tersebut di tempat lain dalam kueri.

Subquery mungkin berisi klausa ORDER BY, tetapi klausa ini mungkin tidak berpengaruh jika klausa LIMIT atau OFFSET tidak juga ditentukan.

ALAMI

Mendefinisikan gabungan yang secara otomatis menggunakan semua pasangan kolom bernama identik dalam dua tabel sebagai kolom bergabung. Tidak diperlukan kondisi gabungan eksplisit. Misalnya, jika tabel CATEGORY dan EVENT keduanya memiliki kolom bernama CATID, gabungan alami dari tabel tersebut adalah gabungan di atas kolom CATID mereka.

> **Note**
>
> Jika gabungan NATURAL ditentukan tetapi tidak ada pasangan kolom bernama identik yang ada di tabel yang akan digabungkan, kueri default ke cross-join.

join_type

Tentukan salah satu jenis join berikut:

- [BATIN] BERGABUNG
- KIRI [LUAR] BERGABUNG
- KANAN [LUAR] BERGABUNG
- PENUH [LUAR] BERGABUNG
- CROSS JOIN

Cross-join adalah gabungan yang tidak memenuhi syarat; mereka mengembalikan produk Cartesian dari dua tabel.

Gabungan dalam dan luar adalah gabungan yang memenuhi syarat. Mereka memenuhi syarat baik secara implisit (dalam gabungan alami); dengan sintaks ON atau USING dalam klausa FROM; atau dengan kondisi klausa WHERE.

Gabungan bagian dalam mengembalikan baris yang cocok saja, berdasarkan kondisi gabungan atau daftar kolom yang bergabung. Gabungan luar mengembalikan semua baris yang akan dikembalikan oleh gabungan dalam yang setara ditambah baris yang tidak cocok dari tabel “kiri”, tabel “kanan”, atau kedua tabel. Tabel kiri adalah tabel yang terdaftar pertama, dan tabel kanan adalah tabel kedua yang terdaftar. Baris yang tidak cocok berisi nilai NULL untuk mengisi celah di kolom output.

PADA join_condition

Jenis spesifikasi gabungan di mana kolom bergabung dinyatakan sebagai kondisi yang mengikuti kata kunci ON. Sebagai contoh:

```
sales join listing
on sales.listid=listing.listid and sales.eventid=listing.eventid
```

MENGGUNAKAN (join_column [,...])

Jenis spesifikasi gabungan di mana kolom bergabung tercantum dalam tanda kurung. Jika beberapa kolom bergabung ditentukan, mereka dibatasi oleh koma. Kata kunci USING harus mendahului daftar. Sebagai contoh:

```
sales join listing
using (listid,eventid)
```

Catatan penggunaan

Kolom yang bergabung harus memiliki tipe data yang sebanding.

Gabungan ALAMI atau MENGGUNAKAN hanya mempertahankan satu dari setiap pasangan kolom penggabungan dalam kumpulan hasil perantara.

Gabungan dengan sintaks ON mempertahankan kedua kolom yang bergabung dalam kumpulan hasil perantara.

Lihat juga DENGAN klausa.

JOIN contoh

Klausa SQL JOIN digunakan untuk menggabungkan data dari dua atau lebih tabel berdasarkan bidang umum. Hasilnya mungkin atau mungkin tidak berubah tergantung pada metode gabungan yang ditentukan. Untuk informasi selengkapnya tentang sintaks klausa JOIN, lihat. Parameter

Kueri berikut adalah gabungan dalam (tanpa kata kunci JOIN) antara tabel LISTING dan tabel PENJUALAN, di mana LISTID dari tabel LISTING adalah antara 1 dan 5. Kueri ini cocok dengan nilai kolom LISTID dalam tabel LISTING (tabel kiri) dan tabel PENJUALAN (tabel kanan). Hasilnya menunjukkan bahwa LISTID 1, 4, dan 5 sesuai dengan kriteria.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing, sales
where listing.listid = sales.listid
```

```
and listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

Kueri berikut adalah gabungan luar kiri. Gabungan luar kiri dan kanan mempertahankan nilai dari salah satu tabel yang digabungkan ketika tidak ada kecocokan yang ditemukan di tabel lainnya. Tabel kiri dan kanan adalah tabel pertama dan kedua yang tercantum dalam sintaks. Nilai NULL digunakan untuk mengisi “celah” di set hasil. Kueri ini cocok dengan nilai kolom LISTID dalam tabel LISTING (tabel kiri) dan tabel PENJUALAN (tabel kanan). Hasilnya menunjukkan bahwa LISTID 2 dan 3 tidak menghasilkan penjualan apa pun.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing left outer join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     2 | NULL   | NULL
     3 | NULL   | NULL
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

Kueri berikut adalah gabungan luar kanan. Kueri ini cocok dengan nilai kolom LISTID dalam tabel LISTING (tabel kiri) dan tabel PENJUALAN (tabel kanan). Hasilnya menunjukkan bahwa LISTID 1, 4, dan 5 cocok dengan kriteria.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing right outer join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;
```

```
listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

Kueri berikut adalah gabungan penuh. Gabungan penuh mempertahankan nilai dari tabel yang digabungkan ketika tidak ada kecocokan yang ditemukan di tabel lainnya. Tabel kiri dan kanan adalah tabel pertama dan kedua yang tercantum dalam sintaks. Nilai NULL digunakan untuk mengisi “celah” di set hasil. Kueri ini cocok dengan nilai kolom LISTID dalam tabel LISTING (tabel kiri) dan tabel PENJUALAN (tabel kanan). Hasilnya menunjukkan bahwa LISTID 2 dan 3 tidak menghasilkan penjualan apa pun.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing full join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     2 | NULL   | NULL
     3 | NULL   | NULL
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

Kueri berikut adalah gabungan penuh. Kueri ini cocok dengan nilai kolom LISTID dalam tabel LISTING (tabel kiri) dan tabel PENJUALAN (tabel kanan). Hanya baris yang tidak menghasilkan penjualan apa pun (LISTID 2 dan 3) yang ada di hasil.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing full join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
and (listing.listid IS NULL or sales.listid IS NULL)
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     2 | NULL   | NULL
```

```
     3 | NULL   | NULL
```

Contoh berikut adalah gabungan batin dengan klausa ON. Dalam hal ini, baris NULL tidak dikembalikan.

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from sales join listing
on sales.listid=listing.listid and sales.eventid=listing.eventid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

Kueri berikut adalah gabungan silang atau gabungan Cartesian dari tabel LISTING dan tabel PENJUALAN dengan predikat untuk membatasi hasil. Kueri ini cocok dengan nilai kolom LISTID dalam tabel PENJUALAN dan tabel LISTING untuk LISTID 1, 2, 3, 4, dan 5 di kedua tabel. Hasilnya menunjukkan bahwa 20 baris cocok dengan kriteria.

```
select sales.listid as sales_listid, listing.listid as listing_listid
from sales cross join listing
where sales.listid between 1 and 5
and listing.listid between 1 and 5
order by 1,2;

sales_listid | listing_listid
-------------+---------------
1            | 1
1            | 2
1            | 3
1            | 4
1            | 5
4            | 1
4            | 2
4            | 3
4            | 4
4            | 5
5            | 1
```

```
5            | 1
5            | 2
5            | 2
5            | 3
5            | 3
5            | 4
5            | 4
5            | 5
5            | 5
```

Contoh berikut adalah gabungan alami antara dua tabel. Dalam hal ini, kolom listid, sellerid, eventid, dan dateid memiliki nama dan tipe data yang identik di kedua tabel dan digunakan sebagai kolom gabungan. Hasilnya dibatasi hingga lima baris.

```
select listid, sellerid, eventid, dateid, numtickets
from listing natural join sales
order by 1
limit 5;

listid | sellerid  | eventid | dateid | numtickets
-------+-----------+---------+--------+-----------
113    | 29704     | 4699    | 2075   | 22
115    | 39115     | 3513    | 2062   | 14
116    | 43314     | 8675    | 1910   | 28
118    | 6079      | 1611    | 1862   | 9
163    | 24880     | 8253    | 1888   | 14
```

Contoh berikut adalah gabungan antara dua tabel dengan klausa USING. Dalam hal ini, kolom listid dan eventid digunakan sebagai kolom gabungan. Hasilnya dibatasi hingga lima baris.

```
select listid, listing.sellerid, eventid, listing.dateid, numtickets
from listing join sales
using (listid, eventid)
order by 1
limit 5;

listid | sellerid | eventid | dateid | numtickets
-------+----------+---------+--------+-----------
1      | 36861    | 7872    | 1850   | 10
4      | 8117     | 4337    | 1970   | 8
5      | 1616     | 8647    | 1963   | 4
5      | 1616     | 8647    | 1963   | 4
```

```
6      | 47402    | 8240    | 2053   | 18
```

Query berikut adalah gabungan dalam dari dua subquery dalam klausa FROM. Kueri menemukan jumlah tiket yang terjual dan tidak terjual untuk berbagai kategori acara (konser dan pertunjukan). Subquery klausa FROM adalah subquery tabel; mereka dapat mengembalikan beberapa kolom dan baris.

```
select catgroup1, sold, unsold
from
(select catgroup, sum(qtysold) as sold
from category c, event e, sales s
where c.catid = e.catid and e.eventid = s.eventid
group by catgroup) as a(catgroup1, sold)
join
(select catgroup, sum(numtickets)-sum(qtysold) as unsold
from category c, event e, sales s, listing l
where c.catid = e.catid and e.eventid = s.eventid
and s.listid = l.listid
group by catgroup) as b(catgroup2, unsold)

on a.catgroup1 = b.catgroup2
order by 1;

catgroup1 |  sold  | unsold
----------+--------+--------
Concerts  | 195444 |1067199
Shows     | 149905 | 817736
```

Klausa WHERE

Klausa WHERE berisi kondisi yang menggabungkan tabel atau menerapkan predikat ke kolom dalam tabel. Tabel dapat bergabung dalam dengan menggunakan sintaks yang sesuai baik dalam klausa WHERE atau klausa FROM. Kriteria gabungan luar harus ditentukan dalam klausa FROM.

Sintaks

```
[ WHERE condition ]
```

ketentuan

Setiap kondisi pencarian dengan hasil Boolean, seperti kondisi gabungan atau predikat pada kolom tabel. Contoh berikut adalah ketentuan gabungan yang valid:

```
sales.listid=listing.listid
sales.listid<>listing.listid
```

Contoh berikut adalah kondisi yang valid pada kolom dalam tabel:

```
catgroup like 'S%'
venueseats between 20000 and 50000
eventname in('Jersey Boys','Spamalot')
year=2008
length(catdesc)>25
date_part(month, caldate)=6
```

Kondisi bisa sederhana atau kompleks; untuk kondisi kompleks, Anda dapat menggunakan tanda kurung untuk mengisolasi unit logis. Dalam contoh berikut, kondisi bergabung diapit oleh tanda kurung.

```
where (category.catid=event.catid) and category.catid in(6,7,8)
```

Catatan penggunaan

Anda dapat menggunakan alias dalam klausa WHERE untuk referensi ekspresi daftar pilih.

Anda tidak dapat membatasi hasil fungsi agregat dalam klausa WHERE; gunakan klausa HAVING untuk tujuan ini.

Kolom yang dibatasi dalam klausa WHERE harus berasal dari referensi tabel dalam klausa FROM.

Contoh

Kueri berikut menggunakan kombinasi batasan klausa WHERE yang berbeda, termasuk kondisi gabungan untuk tabel PENJUALAN dan EVENT, predikat pada kolom EVENTNAME, dan dua predikat pada kolom STARTTIME.

```
select eventname, starttime, pricepaid/qtysold as costperticket, qtysold
```

```
from sales, event
where sales.eventid = event.eventid
and eventname='Hannah Montana'
and date_part(quarter, starttime) in(1,2)
and date_part(year, starttime) = 2008
order by 3 desc, 4, 2, 1 limit 10;

eventname    |      starttime      |   costperticket   | qtysold
---------------+---------------------+-------------------+---------
Hannah Montana | 2008-06-07 14:00:00 |     1706.00000000 |       2
Hannah Montana | 2008-05-01 19:00:00 |     1658.00000000 |       2
Hannah Montana | 2008-06-07 14:00:00 |     1479.00000000 |       1
Hannah Montana | 2008-06-07 14:00:00 |     1479.00000000 |       3
Hannah Montana | 2008-06-07 14:00:00 |     1163.00000000 |       1
Hannah Montana | 2008-06-07 14:00:00 |     1163.00000000 |       2
Hannah Montana | 2008-06-07 14:00:00 |     1163.00000000 |       4
Hannah Montana | 2008-05-01 19:00:00 |      497.00000000 |       1
Hannah Montana | 2008-05-01 19:00:00 |      497.00000000 |       2
Hannah Montana | 2008-05-01 19:00:00 |      497.00000000 |       4
(10 rows)
```

Klausa GROUP BY

Klausa GROUP BY mengidentifikasi kolom pengelompokan untuk kueri. Kolom pengelompokan harus dideklarasikan saat kueri menghitung agregat dengan fungsi standar seperti SUM, AVG, dan COUNT. Jika fungsi agregat hadir dalam ekspresi SELECT, kolom apa pun dalam ekspresi SELECT yang tidak dalam fungsi agregat harus berada dalam klausa GROUP BY.

Untuk informasi selengkapnya, lihat Fungsi SQL di AWS Clean Rooms.

Sintaks

```
GROUP BY group_by_clause [, ...]

group_by_clause := {
    expr |
        ROLLUP ( expr [, ...] ) |
        }
```

Parameter

expr

Daftar kolom atau ekspresi harus cocok dengan daftar ekspresi non-agregat dalam daftar pilih kueri. Misalnya, pertimbangkan kueri sederhana berikut.

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by listid, eventid
order by 3, 4, 2, 1
limit 5;

listid | eventid | revenue | numtix
-------+---------+---------+--------
89397  |      47 |   20.00 |      1
106590 |      76 |   20.00 |      1
124683 |     393 |   20.00 |      1
103037 |     403 |   20.00 |      1
147685 |     429 |   20.00 |      1
(5 rows)
```

Dalam kueri ini, daftar pilih terdiri dari dua ekspresi agregat. Yang pertama menggunakan fungsi SUM dan yang kedua menggunakan fungsi COUNT. Dua kolom yang tersisa, LISTID dan EVENTID, harus dinyatakan sebagai kolom pengelompokan.

Ekspresi dalam klausa GROUP BY juga dapat mereferensikan daftar pilih dengan menggunakan nomor urut. Misalnya, contoh sebelumnya dapat disingkat sebagai berikut.

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by 1,2
order by 3, 4, 2, 1
limit 5;

listid | eventid | revenue | numtix
-------+---------+---------+--------
89397  |      47 |   20.00 |      1
106590 |      76 |   20.00 |      1
124683 |     393 |   20.00 |      1
```

```
103037 |     403 |   20.00 |      1
147685 |     429 |   20.00 |      1
(5 rows)
```

ROLLUP

Anda dapat menggunakan ekstensi agregasi ROLLUP untuk melakukan pekerjaan beberapa operasi GROUP BY dalam satu pernyataan. Untuk informasi selengkapnya tentang ekstensi agregasi dan fungsi terkait, lihatEkstensi agregasi.

Ekstensi agregasi

AWS Clean Rooms mendukung ekstensi agregasi untuk melakukan pekerjaan beberapa operasi GROUP BY dalam satu pernyataan.

SET PENGELOMPOKAN

Menghitung satu atau lebih kumpulan pengelompokan dalam satu pernyataan. Kumpulan pengelompokan adalah kumpulan klausa GROUP BY tunggal, satu set kolom 0 atau lebih yang dengannya Anda dapat mengelompokkan kumpulan hasil kueri. GROUP BY GROUPING SETS setara dengan menjalankan query UNION ALL pada satu set hasil yang dikelompokkan berdasarkan kolom yang berbeda. Misalnya, GROUP BY GROUPING SETS ((a), (b)) setara dengan GROUP BY a UNION ALL GROUP BY b.

Contoh berikut mengembalikan biaya produk tabel pesanan dikelompokkan sesuai dengan kategori produk dan jenis produk yang dijual.

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY GROUPING SETS(category, product);

       category       |       product        | total
----------------------+----------------------+-------
 computers            |                      |  2100
 cellphones           |                      |  1610
                      | laptop               |  2050
                      | smartphone           |  1610
                      | mouse                |    50
```

```
(5 rows)
```

ROLLUP

Mengasumsikan hierarki di mana kolom sebelumnya dianggap sebagai orang tua dari kolom berikutnya. ROLLUP mengelompokkan data berdasarkan kolom yang disediakan, mengembalikan baris subtotal tambahan yang mewakili total di semua tingkat kolom pengelompokan, selain baris yang dikelompokkan. Misalnya, Anda dapat menggunakan GROUP BY ROLLUP ((a), (b)) untuk mengembalikan kumpulan hasil yang dikelompokkan terlebih dahulu oleh a, kemudian oleh b sambil mengasumsikan bahwa b adalah ayat dari a. ROLLUP juga mengembalikan baris dengan seluruh hasil yang ditetapkan tanpa pengelompokan kolom.

GROUP BY ROLLUP ((a), (b)) setara dengan GROUP BY GROUPING SETS ((a, b), (a), ()).

Contoh berikut mengembalikan biaya produk tabel pesanan dikelompokkan pertama berdasarkan kategori dan kemudian produk, dengan produk sebagai subdivisi kategori.

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY ROLLUP(category, product) ORDER BY 1,2;

       category       |       product        | total
----------------------+----------------------+-------
 cellphones           | smartphone           |  1610
 cellphones           |                      |  1610
 computers            | laptop               |  2050
 computers            | mouse                |    50
 computers            |                      |  2100
                      |                      |  3710
(6 rows)
```

KUBUS

Kelompokkan data berdasarkan kolom yang disediakan, mengembalikan baris subtotal tambahan yang mewakili total di semua tingkat kolom pengelompokan, selain baris yang dikelompokkan. CUBE mengembalikan baris yang sama dengan ROLLUP, sambil menambahkan baris subtotal tambahan untuk setiap kombinasi kolom pengelompokan yang tidak dicakup oleh ROLLUP. Misalnya, Anda dapat menggunakan GROUP BY CUBE ((a), (b)) untuk mengembalikan kumpulan hasil yang dikelompokkan terlebih dahulu oleh a, kemudian oleh b sambil mengasumsikan bahwa b adalah subbagian dari a, lalu oleh b saja. CUBE juga mengembalikan baris dengan seluruh hasil yang ditetapkan tanpa pengelompokan kolom.

GROUP BY CUBE ((a), (b)) setara dengan GROUP BY GROUPING SETS ((a, b), (a), (b), ()).

Contoh berikut mengembalikan biaya produk tabel pesanan dikelompokkan pertama berdasarkan kategori dan kemudian produk, dengan produk sebagai subdivisi kategori. Berbeda dengan contoh sebelumnya untuk ROLLUP, pernyataan mengembalikan hasil untuk setiap kombinasi kolom pengelompokan.

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY CUBE(category, product) ORDER BY 1,2;

       category       |       product        | total
----------------------+----------------------+-------
 cellphones           | smartphone           |  1610
 cellphones           |                      |  1610
 computers            | laptop               |  2050
 computers            | mouse                |    50
 computers            |                      |  2100
                      | laptop               |  2050
                      | mouse                |    50
                      | smartphone           |  1610
                      |                      |  3710
(9 rows)
```

Klausa HAVING

Klausa HAVING menerapkan kondisi untuk kumpulan hasil dikelompokkan perantara yang dikembalikan oleh kueri.

Sintaks

```
[ HAVING condition ]
```

Misalnya, Anda dapat membatasi hasil fungsi SUM:

```
having sum(pricepaid) >10000
```

Kondisi HAVING diterapkan setelah semua kondisi klausa WHERE diterapkan dan operasi GROUP BY selesai.

Kondisi itu sendiri mengambil bentuk yang sama dengan kondisi klausa WHERE.

Catatan penggunaan

- Setiap kolom yang direferensikan dalam kondisi klausa HAVING harus berupa kolom pengelompokan atau kolom yang mengacu pada hasil fungsi agregat.
- Dalam klausa HAVING, Anda tidak dapat menentukan:
  - Nomor urut yang mengacu pada item daftar pilih. Hanya klausa GROUP BY dan ORDER BY yang menerima nomor urut.

Contoh-contoh

Kueri berikut menghitung total penjualan tiket untuk semua acara berdasarkan nama, kemudian menghilangkan peristiwa di mana total penjualan kurang dari $800.000. Kondisi HAVING diterapkan pada hasil fungsi agregat dalam daftar pilih:`sum(pricepaid)`.

```
select eventname, sum(pricepaid)
from sales join event on sales.eventid = event.eventid
group by 1
having sum(pricepaid) > 800000
order by 2 desc, 1;

eventname     |    sum
------------------+-----------
Mamma Mia!       | 1135454.00
Spring Awakening |  972855.00
The Country Girl |  910563.00
Macbeth          |  862580.00
Jersey Boys      |  811877.00
Legally Blonde   |  804583.00
(6 rows)
```

Query berikut menghitung set hasil yang sama. Namun, dalam kasus ini, kondisi HAVING diterapkan ke agregat yang tidak ditentukan dalam daftar pilih:`sum(qtysold)`. Acara yang tidak menjual lebih dari 2.000 tiket dihilangkan dari hasil akhir.

```
select eventname, sum(pricepaid)
from sales join event on sales.eventid = event.eventid
group by 1
having sum(qtysold) >2000
order by 2 desc, 1;
```

```
eventname     |    sum
------------------+-----------
Mamma Mia!       | 1135454.00
Spring Awakening |  972855.00
The Country Girl |  910563.00
Macbeth          |  862580.00
Jersey Boys      |  811877.00
Legally Blonde   |  804583.00
Chicago          |  790993.00
Spamalot         |  714307.00
(8 rows)
```

Tetapkan operator

Operator set UNION, INTERSECT, dan EXCEPT digunakan untuk membandingkan dan menggabungkan hasil dari dua ekspresi kueri terpisah. Misalnya, jika Anda ingin mengetahui pengguna situs web mana yang merupakan pembeli dan penjual tetapi nama pengguna mereka disimpan dalam kolom atau tabel terpisah, Anda dapat menemukan persimpangan kedua jenis pengguna ini. Jika Anda ingin tahu pengguna situs web mana yang merupakan pembeli tetapi bukan penjual, Anda dapat menggunakan operator EXCEPT untuk menemukan perbedaan antara dua daftar pengguna. Jika Anda ingin membuat daftar semua pengguna, apa pun perannya, Anda dapat menggunakan operator UNION.

> **Note**
>
> Klausa ORDER BY, LIMIT, SELECT TOP, dan OFFSET tidak dapat digunakan dalam ekspresi kueri yang digabungkan oleh operator set UNION, UNION ALL, INTERSECT, dan EXCEPT.

Topik

- Sintaks
- Parameter
- Urutan evaluasi untuk operator yang ditetapkan
- Catatan penggunaan
- Contoh kueri UNION
- Contoh UNION ALL query

- Contoh pertanyaan INTERSECT
- Contoh KECUALI kueri

Sintaks

```
query
{ UNION [ ALL ] | INTERSECT | EXCEPT | MINUS }
query
```

Parameter

kueri

Ekspresi kueri yang sesuai, dalam bentuk daftar pilihannya, dengan ekspresi kueri kedua yang mengikuti operator UNION, INTERSECT, atau EXCEPT. Kedua ekspresi harus berisi jumlah kolom keluaran yang sama dengan tipe data yang kompatibel; jika tidak, dua set hasil tidak dapat dibandingkan dan digabungkan. Operasi set tidak mengizinkan konversi implisit antara berbagai kategori tipe data; untuk informasi selengkapnya, lihatKetik kompatibilitas dan konversi.

Anda dapat membuat kueri yang berisi ekspresi kueri dalam jumlah tak terbatas dan menautkannya dengan operator UNION, INTERSECT, dan EXCEPT dalam kombinasi apa pun. Misalnya, struktur kueri berikut valid, dengan asumsi bahwa tabel T1, T2, dan T3 berisi kumpulan kolom yang kompatibel:

```
select * from t1
union
select * from t2
except
select * from t3
```

UNION

Mengatur operasi yang mengembalikan baris dari dua ekspresi query, terlepas dari apakah baris berasal dari satu atau kedua ekspresi.

BERPOTONGAN

Mengatur operasi yang mengembalikan baris yang berasal dari dua ekspresi query. Baris yang tidak dikembalikan oleh kedua ekspresi akan dibuang.

KECUALI | MINUS

Mengatur operasi yang mengembalikan baris yang berasal dari salah satu dari dua ekspresi query. Agar memenuhi syarat untuk hasil, baris harus ada di tabel hasil pertama tetapi bukan yang kedua. MINUS dan EXCEPT adalah sinonim yang tepat.

SEMUA

Kata kunci ALL mempertahankan setiap baris duplikat yang dihasilkan oleh UNION. Perilaku default saat kata kunci ALL tidak digunakan adalah membuang duplikat ini. INTERSECT ALL, KECUALI ALL, dan MINUS ALL tidak didukung.

Urutan evaluasi untuk operator yang ditetapkan

Operator set UNION dan EXCEPLE adalah asosiatif kiri. Jika tanda kurung tidak ditentukan untuk mempengaruhi urutan prioritas, kombinasi dari operator set ini dievaluasi dari kiri ke kanan. Misalnya, dalam kueri berikut, UNION T1 dan T2 dievaluasi terlebih dahulu, kemudian operasi EXCEPT dilakukan pada hasil UNION:

```
select * from t1
union
select * from t2
except
select * from t3
```

Operator INTERSECT lebih diutamakan daripada operator UNION dan EXCEPT ketika kombinasi operator digunakan dalam kueri yang sama. Misalnya, kueri berikut mengevaluasi persimpangan T2 dan T3, lalu menyatukan hasilnya dengan T1:

```
select * from t1
union
select * from t2
intersect
select * from t3
```

Dengan menambahkan tanda kurung, Anda dapat menerapkan urutan evaluasi yang berbeda. Dalam kasus berikut, hasil penyatuan T1 dan T2 berpotongan dengan T3, dan kueri kemungkinan akan menghasilkan hasil yang berbeda.

```
(select * from t1
```

```
union
select * from t2)
intersect
(select * from t3)
```

Catatan penggunaan

- Nama kolom yang dikembalikan dalam hasil kueri operasi set adalah nama kolom (atau alias) dari tabel dalam ekspresi kueri pertama. Karena nama kolom ini berpotensi menyesatkan, karena nilai dalam kolom berasal dari tabel di kedua sisi operator set, Anda mungkin ingin memberikan alias yang berarti untuk kumpulan hasil.
- Ketika kueri operator yang disetel mengembalikan hasil desimal, kolom hasil yang sesuai dipromosikan untuk mengembalikan presisi dan skala yang sama. Misalnya, dalam kueri berikut, di mana T1.REVENUE adalah kolom DECIMAL (10,2) dan T2.REVENUE adalah kolom DECIMAL (8,4), hasil desimal dipromosikan ke DECIMAL (12,4):

  ```
  select t1.revenue union select t2.revenue;
  ```

  Skala ini 4 karena itu adalah skala maksimum dari dua kolom. Ketepatannya adalah 12 karena T1.REVENUE membutuhkan 8 digit di sebelah kiri titik desimal (12 - 4 = 8). Promosi jenis ini memastikan bahwa semua nilai dari kedua sisi UNION sesuai dengan hasilnya. Untuk nilai 64-bit, presisi hasil maksimum adalah 19 dan skala hasil maksimum adalah 18. Untuk nilai 128-bit, presisi hasil maksimum adalah 38 dan skala hasil maksimum adalah 37.

  Jika tipe data yang dihasilkan melebihi AWS Clean Rooms presisi dan batas skala, kueri mengembalikan kesalahan.
- Untuk operasi set, dua baris diperlakukan sebagai identik jika, untuk setiap pasangan kolom yang sesuai, dua nilai data sama atau keduanya NULL. Misalnya, jika tabel T1 dan T2 keduanya berisi satu kolom dan satu baris, dan baris itu adalah NULL di kedua tabel, operasi INTERSECT di atas tabel tersebut mengembalikan baris itu.

Contoh kueri UNION

Dalam query UNION berikut, baris dalam tabel PENJUALAN digabungkan dengan baris dalam tabel LISTING. Tiga kolom yang kompatibel dipilih dari setiap tabel; dalam hal ini, kolom yang sesuai memiliki nama dan tipe data yang sama.

```
select listid, sellerid, eventid from listing
union select listid, sellerid, eventid from sales


listid | sellerid | eventid
--------+----------+---------
1 |    36861 |    7872
2 |    16002 |    4806
3 |    21461 |    4256
4 |     8117 |    4337
5 |     1616 |    8647
```

Contoh berikut menunjukkan bagaimana Anda dapat menambahkan nilai literal untuk output dari query UNION sehingga Anda dapat melihat ekspresi query yang dihasilkan setiap baris dalam set hasil. Kueri mengidentifikasi baris dari ekspresi kueri pertama sebagai “B” (untuk pembeli) dan baris dari ekspresi kueri kedua sebagai “S” (untuk penjual).

Kueri mengidentifikasi pembeli dan penjual untuk transaksi tiket yang harganya $10.000 atau lebih. Satu-satunya perbedaan antara dua ekspresi kueri di kedua sisi operator UNION adalah kolom bergabung untuk tabel PENJUALAN.

```
select listid, lastname, firstname, username,
pricepaid as price, 'S' as buyorsell
from sales, users
where sales.sellerid=users.userid
and pricepaid >=10000
union
select listid, lastname, firstname, username, pricepaid,
'B' as buyorsell
from sales, users
where sales.buyerid=users.userid
and pricepaid >=10000

listid | lastname | firstname | username |   price   | buyorsell
--------+----------+-----------+----------+-----------+-----------
209658 | Lamb     | Colette   | VOR15LYI |  10000.00 | B
209658 | West     | Kato      | ELU81XAA |  10000.00 | S
212395 | Greer    | Harlan    | GXO71KOC |  12624.00 | S
212395 | Perry    | Cora      | YWR73YNZ |  12624.00 | B
215156 | Banks    | Patrick   | ZNQ69CLT |  10000.00 | S
215156 | Hayden   | Malachi   | BBG56AKU |  10000.00 | B
```

Contoh berikut menggunakan operator UNION ALL karena baris duplikat, jika ditemukan, perlu dipertahankan dalam hasilnya. Untuk serangkaian ID peristiwa tertentu, kueri mengembalikan 0 atau lebih baris untuk setiap penjualan yang terkait dengan setiap acara, dan 0 atau 1 baris untuk setiap daftar acara tersebut. ID peristiwa unik untuk setiap baris dalam tabel LISTING dan EVENT, tetapi mungkin ada beberapa penjualan untuk kombinasi ID acara dan daftar yang sama di tabel PENJUALAN.

Kolom ketiga dalam set hasil mengidentifikasi sumber baris. Jika berasal dari tabel PENJUALAN, itu ditandai “Ya” di kolom SALESROW. (SALESROW adalah alias untuk SALES.LISTID.) Jika baris berasal dari tabel LISTING, itu ditandai “Tidak” di kolom SALESROW.

Dalam hal ini, set hasil terdiri dari tiga baris penjualan untuk daftar 500, acara 7787. Dengan kata lain, tiga transaksi berbeda terjadi untuk daftar dan kombinasi acara ini. Dua daftar lainnya, 501 dan 502, tidak menghasilkan penjualan apa pun, jadi satu-satunya baris yang dihasilkan kueri untuk ID daftar ini berasal dari tabel LISTING (SALESROW = 'Tidak').

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union all
select eventid, listid, 'No'
from listing
where listid in(500,501,502)

eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
7787 |    500 | Yes
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
```

Jika Anda menjalankan kueri yang sama tanpa kata kunci ALL, hasilnya hanya mempertahankan satu transaksi penjualan.

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union
select eventid, listid, 'No'
```

```
from listing
where listid in(500,501,502)

eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
```

Contoh UNION ALL query

Contoh berikut menggunakan operator UNION ALL karena baris duplikat, jika ditemukan, perlu dipertahankan dalam hasilnya. Untuk serangkaian ID peristiwa tertentu, kueri mengembalikan 0 atau lebih baris untuk setiap penjualan yang terkait dengan setiap acara, dan 0 atau 1 baris untuk setiap daftar acara tersebut. ID peristiwa unik untuk setiap baris dalam tabel LISTING dan EVENT, tetapi mungkin ada beberapa penjualan untuk kombinasi ID acara dan daftar yang sama di tabel PENJUALAN.

Kolom ketiga dalam set hasil mengidentifikasi sumber baris. Jika berasal dari tabel PENJUALAN, itu ditandai “Ya” di kolom SALESROW. (SALESROW adalah alias untuk SALES.LISTID.) Jika baris berasal dari tabel LISTING, itu ditandai “Tidak” di kolom SALESROW.

Dalam hal ini, set hasil terdiri dari tiga baris penjualan untuk daftar 500, acara 7787. Dengan kata lain, tiga transaksi berbeda terjadi untuk daftar dan kombinasi acara ini. Dua daftar lainnya, 501 dan 502, tidak menghasilkan penjualan apa pun, jadi satu-satunya baris yang dihasilkan kueri untuk ID daftar ini berasal dari tabel LISTING (SALESROW = 'Tidak').

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union all
select eventid, listid, 'No'
from listing
where listid in(500,501,502)

eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
7787 |    500 | Yes
7787 |    500 | Yes
```

```
6473 |    501 | No
5108 |    502 | No
```

Jika Anda menjalankan kueri yang sama tanpa kata kunci ALL, hasilnya hanya mempertahankan satu transaksi penjualan.

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
```

Contoh pertanyaan INTERSECT

Bandingkan contoh berikut dengan contoh UNION pertama. Satu-satunya perbedaan antara kedua contoh adalah operator set yang digunakan, tetapi hasilnya sangat berbeda. Hanya satu baris yang sama:

```
235494 |    23875 |    8771
```

Ini adalah satu-satunya baris dalam hasil terbatas dari 5 baris yang ditemukan di kedua tabel.

```
select listid, sellerid, eventid from listing
intersect
select listid, sellerid, eventid from sales

listid | sellerid | eventid
--------+----------+---------
235494 |    23875 |    8771
235482 |     1067 |    2667
235479 |     1589 |    7303
235476 |    15550 |     793
235475 |    22306 |    7848
```

Pertanyaan berikut menemukan peristiwa (yang tiketnya terjual) yang terjadi di tempat-tempat di New York City dan Los Angeles pada bulan Maret. Perbedaan antara dua ekspresi kueri adalah kendala pada kolom VENUECITY.

```
select distinct eventname from event, sales, venue
where event.eventid=sales.eventid and event.venueid=venue.venueid
and date_part(month,starttime)=3 and venuecity='Los Angeles'
intersect
select distinct eventname from event, sales, venue
where event.eventid=sales.eventid and event.venueid=venue.venueid
and date_part(month,starttime)=3 and venuecity='New York City';

eventname
----------------------------
A Streetcar Named Desire
Dirty Dancing
Electra
Running with Annalise
Hairspray
Mary Poppins
November
Oliver!
Return To Forever
Rhinoceros
South Pacific
The 39 Steps
The Bacchae
The Caucasian Chalk Circle
The Country Girl
Wicked
Woyzeck
```

Contoh KECUALI kueri

Tabel CATEGORY dalam database berisi 11 baris berikut:

```
 catid | catgroup |  catname  |                 catdesc
-------+----------+-----------+--------------------------------------------
   1   | Sports   | MLB       | Major League Baseball
   2   | Sports   | NHL       | National Hockey League
   3   | Sports   | NFL       | National Football League
   4   | Sports   | NBA       | National Basketball Association
   5   | Sports   | MLS       | Major League Soccer
```

```
   6   | Shows    | Musicals  | Musical theatre
   7   | Shows    | Plays     | All non-musical theatre
   8   | Shows    | Opera     | All opera and light opera
   9   | Concerts | Pop       | All rock and pop music concerts
  10   | Concerts | Jazz      | All jazz singers and bands
  11   | Concerts | Classical | All symphony, concerto, and choir concerts
(11 rows)
```

Asumsikan bahwa tabel CATEGORY_STAGE (tabel pementasan) berisi satu baris tambahan:

```
 catid | catgroup |  catname  |                 catdesc
-------+----------+-----------+--------------------------------------------
   1   | Sports   | MLB       | Major League Baseball
   2   | Sports   | NHL       | National Hockey League
   3   | Sports   | NFL       | National Football League
   4   | Sports   | NBA       | National Basketball Association
   5   | Sports   | MLS       | Major League Soccer
   6   | Shows    | Musicals  | Musical theatre
   7   | Shows    | Plays     | All non-musical theatre
   8   | Shows    | Opera     | All opera and light opera
   9   | Concerts | Pop       | All rock and pop music concerts
  10   | Concerts | Jazz      | All jazz singers and bands
  11   | Concerts | Classical | All symphony, concerto, and choir concerts
  12   | Concerts | Comedy    | All stand up comedy performances
(12 rows)
```

Kembalikan perbedaan antara dua tabel. Dengan kata lain, kembalikan baris yang ada di tabel CATEGORY_STAGE tetapi tidak di tabel CATEGORY:

```
select * from category_stage
except
select * from category;

catid | catgroup | catname |             catdesc
-------+----------+---------+----------------------------------
  12  | Concerts | Comedy  | All stand up comedy performances
(1 row)
```

Kueri setara berikut menggunakan sinonim MINUS.

```
select * from category_stage
minus
```

```
select * from category;

catid | catgroup | catname |             catdesc
-------+----------+---------+----------------------------------
  12  | Concerts | Comedy  | All stand up comedy performances
(1 row)
```

Jika Anda membalikkan urutan ekspresi SELECT, kueri tidak mengembalikan baris.

Klausa ORDER BY

Klausa ORDER BY mengurutkan kumpulan hasil kueri.

> **Note**
>
> Ekspresi ORDER BY terluar harus hanya memiliki kolom yang ada di daftar pilih.

Topik

- Sintaks
- Parameter
- Catatan penggunaan
- Contoh dengan ORDER BY

Sintaks

```
[ ORDER BY expression [ ASC | DESC ] ]
[ NULLS FIRST | NULLS LAST ]
[ LIMIT { count | ALL } ]
[ OFFSET start ]
```

Parameter

ekspresi

Ekspresi yang mendefinisikan urutan pengurutan hasil query. Ini terdiri dari satu atau lebih kolom dalam daftar pilih. Hasil dikembalikan berdasarkan urutan UTF-8 biner. Anda juga dapat menentukan yang berikut:

- Nomor urut yang mewakili posisi entri daftar pilih (atau posisi kolom dalam tabel jika tidak ada daftar pilih)
- Alias yang menentukan entri daftar pilih

Ketika klausa ORDER BY berisi beberapa ekspresi, kumpulan hasil diurutkan menurut ekspresi pertama, maka ekspresi kedua diterapkan ke baris yang memiliki nilai yang cocok dari ekspresi pertama, dan seterusnya.

ASC | DESC

Opsi yang mendefinisikan urutan pengurutan untuk ekspresi, sebagai berikut:

- ASC: naik (misalnya, rendah ke tinggi untuk nilai numerik dan 'A' ke 'Z' untuk string karakter). Jika tidak ada opsi yang ditentukan, data diurutkan dalam urutan menaik secara default.
- DESC: turun (tinggi ke rendah untuk nilai numerik; 'Z' ke 'A' untuk string).

NULLS PERTAMA | NULLS TERAKHIR

Opsi yang menentukan apakah nilai NULL harus diurutkan terlebih dahulu, sebelum nilai non-null, atau terakhir, setelah nilai non-null. Secara default, nilai NULL diurutkan dan diberi peringkat terakhir dalam urutan ASC, dan diurutkan dan diberi peringkat pertama dalam urutan DESC.

BATASAN nomor | SEMUA

Opsi yang mengontrol jumlah baris yang diurutkan yang dikembalikan kueri. Bilangan LIMIT harus berupa bilangan bulat positif; nilai maksimumnya adalah`2147483647`.

LIMIT 0 tidak mengembalikan baris. Anda dapat menggunakan sintaks ini untuk tujuan pengujian: untuk memeriksa apakah kueri berjalan (tanpa menampilkan baris apa pun) atau mengembalikan daftar kolom dari tabel. Klausa ORDER BY berlebihan jika Anda menggunakan LIMIT 0 untuk mengembalikan daftar kolom. Defaultnya adalah LIMIT ALL.

OFFSET mulai

Opsi yang menentukan untuk melewati jumlah baris sebelum memulai sebelum mulai mengembalikan baris. Angka OFFSET harus berupa bilangan bulat positif; nilai maksimumnya adalah. `2147483647` Saat digunakan dengan opsi LIMIT, baris OFFSET dilewati sebelum mulai menghitung baris LIMIT yang dikembalikan. Jika opsi LIMIT tidak digunakan, jumlah baris dalam kumpulan hasil dikurangi dengan jumlah baris yang dilewati. Baris yang dilewati oleh klausa

OFFSET masih harus dipindai, jadi mungkin tidak efisien untuk menggunakan nilai OFFSET yang besar.

Catatan penggunaan

Perhatikan perilaku yang diharapkan berikut dengan klausa ORDER BY:

- Nilai NULL dianggap “lebih tinggi” dari semua nilai lainnya. Dengan urutan urutan menaik default, nilai NULL mengurutkan di akhir. Untuk mengubah perilaku ini, gunakan opsi NULLS FIRST.
- Ketika kueri tidak berisi klausa ORDER BY, sistem mengembalikan set hasil tanpa urutan baris yang dapat diprediksi. Kueri yang sama dijalankan dua kali mungkin mengembalikan set hasil dalam urutan yang berbeda.
- Opsi LIMIT dan OFFSET dapat digunakan tanpa klausa ORDER BY; namun, untuk mengembalikan serangkaian baris yang konsisten, gunakan opsi ini bersama dengan ORDER BY.
- Dalam sistem parallel seperti AWS Clean Rooms, ketika ORDER BY tidak menghasilkan urutan yang unik, urutan baris adalah nondeterministik. Artinya, jika ekspresi ORDER BY menghasilkan nilai duplikat, urutan pengembalian baris tersebut mungkin berbeda dari sistem lain atau dari satu proses AWS Clean Rooms ke yang berikutnya.
- AWS Clean Rooms tidak mendukung literal string dalam klausa ORDER BY.

Contoh dengan ORDER BY

Kembalikan semua 11 baris dari tabel CATEGORY, diurutkan berdasarkan kolom kedua, CATGROUP. Untuk hasil yang memiliki nilai CATGROUP yang sama, urutkan nilai kolom CATDESC dengan panjang string karakter. Kemudian urutkan berdasarkan kolom CATID dan CATNAME.

```
select * from category order by 2, 1, 3;

catid | catgroup |  catname  |                 catdesc
-------+----------+-----------+--------------------------------------
10 | Concerts | Jazz      | All jazz singers and bands
9 | Concerts | Pop       | All rock and pop music concerts
11 | Concerts | Classical | All symphony, concerto, and choir conce
6 | Shows    | Musicals  | Musical theatre
7 | Shows    | Plays     | All non-musical theatre
8 | Shows    | Opera     | All opera and light opera
5 | Sports   | MLS       | Major League Soccer
1 | Sports   | MLB       | Major League Baseball
```

```
2 | Sports   | NHL       | National Hockey League
3 | Sports   | NFL       | National Football League
4 | Sports   | NBA       | National Basketball Association
(11 rows)
```

Kembalikan kolom yang dipilih dari tabel PENJUALAN, diurutkan berdasarkan nilai QTYSOLD tertinggi. Batasi hasilnya ke 10 baris teratas:

```
select salesid, qtysold, pricepaid, commission, saletime from sales
order by qtysold, pricepaid, commission, salesid, saletime desc

salesid | qtysold | pricepaid | commission |      saletime
---------+---------+-----------+------------+---------------------
15401 |       8 |    272.00 |      40.80 | 2008-03-18 06:54:56
61683 |       8 |    296.00 |      44.40 | 2008-11-26 04:00:23
90528 |       8 |    328.00 |      49.20 | 2008-06-11 02:38:09
74549 |       8 |    336.00 |      50.40 | 2008-01-19 12:01:21
130232 |       8 |    352.00 |      52.80 | 2008-05-02 05:52:31
55243 |       8 |    384.00 |      57.60 | 2008-07-12 02:19:53
16004 |       8 |    440.00 |      66.00 | 2008-11-04 07:22:31
489 |       8 |    496.00 |      74.40 | 2008-08-03 05:48:55
4197 |       8 |    512.00 |      76.80 | 2008-03-23 11:35:33
16929 |       8 |    568.00 |      85.20 | 2008-12-19 02:59:33
```

Kembalikan daftar kolom dan tidak ada baris dengan menggunakan sintaks LIMIT 0:

```
select * from venue limit 0;
venueid | venuename | venuecity | venuestate | venueseats
---------+-----------+-----------+------------+------------
(0 rows)
```

Contoh subquery

Contoh berikut menunjukkan cara yang berbeda di mana subquery cocok dengan kueri SELECT. Lihat JOIN contoh contoh lain dari penggunaan subquery.

PILIH daftar subquery

Contoh berikut berisi subquery dalam daftar SELECT. Subquery ini adalah skalar: ia mengembalikan hanya satu kolom dan satu nilai, yang diulang dalam hasil untuk setiap baris yang dikembalikan dari

query luar. Kueri membandingkan nilai Q1SALES yang dihitung subquery dengan nilai penjualan untuk dua kuartal lainnya (2 dan 3) pada tahun 2008, seperti yang didefinisikan oleh kueri luar.

```
select qtr, sum(pricepaid) as qtrsales,
(select sum(pricepaid)
from sales join date on sales.dateid=date.dateid
where qtr='1' and year=2008) as q1sales
from sales join date on sales.dateid=date.dateid
where qtr in('2','3') and year=2008
group by qtr
order by qtr;

qtr  |  qtrsales   |   q1sales
-------+-------------+-------------
2     | 30560050.00 | 24742065.00
3     | 31170237.00 | 24742065.00
(2 rows)
```

Subquery klausa WHERE

Contoh berikut berisi subquery tabel dalam klausa WHERE. Subquery ini menghasilkan beberapa baris. Dalam hal ini, baris hanya berisi satu kolom, tetapi subquery tabel dapat berisi beberapa kolom dan baris, sama seperti tabel lainnya.

Kueri menemukan 10 penjual teratas dalam hal tiket maksimum yang terjual. Daftar 10 teratas dibatasi oleh subquery, yang menghapus pengguna yang tinggal di kota di mana ada tempat tiket. Kueri ini dapat ditulis dengan cara yang berbeda; misalnya, subquery dapat ditulis ulang sebagai gabungan dalam kueri utama.

```
select firstname, lastname, city, max(qtysold) as maxsold
from users join sales on users.userid=sales.sellerid
where users.city not in(select venuecity from venue)
group by firstname, lastname, city
order by maxsold desc, city desc
limit 10;

firstname | lastname  |      city      | maxsold
-----------+-----------+----------------+---------
Noah       | Guerrero | Worcester      |       8
Isadora    | Moss     | Winooski       |       8
Kieran     | Harrison | Westminster    |       8
Heidi      | Davis    | Warwick        |       8
```

```
Sara       | Anthony  | Waco           |       8
Bree       | Buck     | Valdez         |       8
Evangeline | Sampson  | Trenton        |       8
Kendall    | Keith    | Stillwater     |       8
Bertha     | Bishop   | Stevens Point  |       8
Patricia   | Anderson | South Portland |       8
(10 rows)
```

DENGAN subquery klausa

Lihat DENGAN klausa.

Subquery yang berkorelasi

Contoh berikut berisi subquery berkorelasi dalam klausa WHERE; subquery semacam ini berisi satu atau lebih korelasi antara kolom dan kolom yang dihasilkan oleh kueri luar. Dalam hal ini, korelasinya adalah`where s.listid=l.listid`. Untuk setiap baris yang dihasilkan kueri luar, subquery dijalankan untuk memenuhi syarat atau mendiskualifikasi baris.

```
select salesid, listid, sum(pricepaid) from sales s
where qtysold=
(select max(numtickets) from listing l
where s.listid=l.listid)
group by 1,2
order by 1,2
limit 5;

salesid | listid |   sum
--------+--------+----------
 27     |     28 | 111.00
 81     |    103 | 181.00
 142    |    149 | 240.00
 146    |    152 | 231.00
 194    |    210 | 144.00
(5 rows)
```

Pola subquery berkorelasi yang tidak didukung

Perencana kueri menggunakan metode penulisan ulang kueri yang disebut decorrelation subquery untuk mengoptimalkan beberapa pola subquery berkorelasi untuk eksekusi di lingkungan MPP. Beberapa jenis subkueri berkorelasi mengikuti pola yang tidak AWS Clean Rooms dapat

mendekorasi dan tidak mendukung. Kueri yang berisi referensi korelasi berikut mengembalikan kesalahan:

- Referensi korelasi yang melewati blok kueri, juga dikenal sebagai “referensi korelasi tingkat lewati.” Misalnya, dalam kueri berikut, blok yang berisi referensi korelasi dan blok yang dilewati dihubungkan oleh predikat NOT EXISTS:

  ```
  select event.eventname from event
  where not exists
  (select * from listing
  where not exists
  (select * from sales where event.eventid=sales.eventid));
  ```

  Blok yang dilewati dalam kasus ini adalah subquery terhadap tabel LISTING. Referensi korelasi menghubungkan tabel EVENT dan SALES.

- Referensi korelasi dari subquery yang merupakan bagian dari klausa ON dalam kueri luar:

  ```
  select * from category
  left join event
  on category.catid=event.catid and eventid =
  (select max(eventid) from sales where sales.eventid=event.eventid);
  ```

  Klausa ON berisi referensi korelasi dari SALES di subquery ke EVENT di kueri luar.

- Referensi korelasi sensitif nol ke tabel sistem. AWS Clean Rooms Sebagai contoh:

  ```
  select attrelid
  from my_locks sl, my_attribute
  where sl.table_id=my_attribute.attrelid and 1 not in
  (select 1 from my_opclass where sl.lock_owner = opcowner);
  ```

- Referensi korelasi dari dalam subquery yang berisi fungsi jendela.

  ```
  select listid, qtysold
  from sales s
  where qtysold not in
  (select sum(numtickets) over() from listing l where s.listid=l.listid);
  ```

- Referensi dalam kolom GROUP BY ke hasil subquery yang berkorelasi. Sebagai contoh:

  ```
  select listing.listid,
  ```

```
(select count (sales.listid) from sales where sales.listid=listing.listid) as list
from listing
group by list, listing.listid;
```

- Referensi korelasi dari subquery dengan fungsi agregat dan klausa GROUP BY, terhubung ke kueri luar oleh predikat IN. (Pembatasan ini tidak berlaku untuk fungsi agregat MIN dan MAX.) Sebagai contoh:

```
select * from listing where listid in
(select sum(qtysold)
from sales
where numtickets>4
group by salesid);
```

Fungsi SQL di AWS Clean Rooms

AWS Clean Rooms mendukung fungsi SQL berikut:

Topik

Fungsi agregat

AWS Clean Rooms mendukung fungsi agregat berikut:

Topik

- Fungsi MAX
- Fungsi MEDIAN
- Fungsi MIN
- Fungsi PERCENTILE_CONT
- Fungsi STDDEV_SAMP dan STDDEV_POP
- SUMdan SUM DISTINCT fungsi
- Fungsi VAR_SAMP dan VAR_POP

Fungsi ANY_VALUE

Fungsi ANY_VALUE mengembalikan nilai apapun dari nilai ekspresi masukan nondeterministik. Fungsi ini dapat mengembalikan NULL jika ekspresi input tidak menghasilkan baris apa pun yang dikembalikan.

Sintaks

```
ANY_VALUE ( [ DISTINCT | ALL ] expression )
```

Argumen

BERBEDA | SEMUA

Tentukan DISTINCT atau ALL untuk mengembalikan nilai apa pun dari nilai ekspresi input. Argumen DISTINCT tidak berpengaruh dan diabaikan.

ekspresi

Kolom target atau ekspresi di mana fungsi beroperasi. Ekspresi adalah salah satu tipe data berikut:

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- REAL
- PRECISON GANDA
- BOOLEAN

- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- JADWAL
- VARBYTE
- SUPER

Pengembalian

Mengembalikan tipe data yang sama sebagai ekspresi.

Catatan penggunaan

Jika pernyataan yang menentukan fungsi ANY_VALUE untuk kolom juga menyertakan referensi kolom kedua, kolom kedua harus muncul dalam klausa GROUP BY atau disertakan dalam fungsi agregat.

Contoh-contoh

Contoh berikut mengembalikan sebuah instance dari `dateid` mana pun di mana `eventname` adalah`Eagles`.

```
select any_value(dateid) as dateid, eventname from event where eventname ='Eagles'
 group by eventname;
```

Berikut ini adalah hasilnya.

```
dateid | eventname
-------+---------------
 1878  | Eagles
```

Contoh berikut mengembalikan sebuah instance dari `dateid` mana pun di mana `eventname` adalah `Eagles` atau`Cold War Kids`.

```
select any_value(dateid) as dateid, eventname from event where eventname in('Eagles',
 'Cold War Kids') group by eventname;
```

Berikut ini adalah hasilnya.

```
dateid | eventname
-------+---------------
 1922  | Cold War Kids
 1878  | Eagles
```

PERKIRAAN fungsi PERCENTILE_DISC

PERKIRAAN PERCENTILE_DISC adalah fungsi distribusi terbalik yang mengasumsikan model distribusi diskrit. Dibutuhkan nilai persentil dan spesifikasi semacam dan mengembalikan elemen dari set yang diberikan. Pendekatan memungkinkan fungsi berjalan lebih cepat, dengan kesalahan relatif rendah sekitar 0,5 persen.

Untuk nilai persentil tertentu, PERKIRAAN PERCENTILE_DISC menggunakan algoritma ringkasan kuantil untuk memperkirakan persentil diskrit ekspresi dalam klausa ORDER BY. PERKIRAAN PERCENTILE_DISC mengembalikan nilai dengan nilai distribusi kumulatif terkecil (sehubungan dengan spesifikasi jenis yang sama) yang lebih besar dari atau sama dengan persentil.

PERKIRAAN PERCENTILE_DISC adalah fungsi compute-node saja. Fungsi mengembalikan kesalahan jika kueri tidak mereferensikan tabel atau tabel AWS Clean Rooms sistem yang ditentukan pengguna.

Sintaks

```
APPROXIMATE  PERCENTILE_DISC ( percentile )
WITHIN GROUP (ORDER BY expr)
```

Argumen

persentil

Konstanta numerik antara 0 dan 1. Null diabaikan dalam perhitungan.

DALAM GRUP (ORDER BY expr)

Klausul yang menentukan nilai numerik atau tanggal/waktu untuk mengurutkan dan menghitung persentil atas.

Pengembalian

Tipe data yang sama dengan ekspresi ORDER BY dalam klausa WITHIN GROUP.

Catatan penggunaan

Jika pernyataan PERKIRAAN PERCENTILE_DISC menyertakan klausa GROUP BY, kumpulan hasil terbatas. Batas bervariasi berdasarkan jenis node dan jumlah node. Jika batas terlampaui, fungsi gagal dan mengembalikan kesalahan berikut.

```
GROUP BY limit for approximate percentile_disc exceeded.
```

Jika Anda perlu mengevaluasi lebih banyak grup daripada batas izin, pertimbangkan untuk menggunakannyaFungsi PERCENTILE_CONT.

Contoh-contoh

Contoh berikut mengembalikan jumlah penjualan, total penjualan, dan nilai persentil kelima puluh untuk 10 tanggal teratas.

```
select top 10 date.caldate,
count(totalprice), sum(totalprice),
approximate percentile_disc(0.5)
within group (order by totalprice)
from listing
join date on listing.dateid = date.dateid
group by date.caldate
order by 3 desc;

caldate    | count | sum        | percentile_disc
-----------+-------+------------+----------------
2008-01-07 |   658 | 2081400.00 |         2020.00
2008-01-02 |   614 | 2064840.00 |         2178.00
2008-07-22 |   593 | 1994256.00 |         2214.00
2008-01-26 |   595 | 1993188.00 |         2272.00
2008-02-24 |   655 | 1975345.00 |         2070.00
2008-02-04 |   616 | 1972491.00 |         1995.00
2008-02-14 |   628 | 1971759.00 |         2184.00
2008-09-01 |   600 | 1944976.00 |         2100.00
2008-07-29 |   597 | 1944488.00 |         2106.00
2008-07-23 |   592 | 1943265.00 |         1974.00
```

Fungsi AVG

AVGFungsi mengembalikan rata-rata (rata-rata aritmatika) dari nilai ekspresi masukan. AVGFungsi ini bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaks

```
AVG (column)
```

Pendapat

kolom

Kolom target tempat fungsi beroperasi. Kolom adalah salah satu tipe data berikut:

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- DOUBLE

Tipe Data

Tipe argumen yang didukung oleh AVG fungsi adalahSMALLINT,INTEGER,BIGINT,DECIMAL, danDOUBLE.

Jenis pengembalian yang didukung oleh AVG fungsi adalah:

- BIGINTuntuk setiap argumen tipe integer
- DOUBLEuntuk argumen floating point
- Mengembalikan tipe data yang sama sebagai ekspresi untuk jenis argumen lainnya

Presisi default untuk hasil AVG fungsi dengan DECIMAL argumen adalah 38. Skala hasilnya sama dengan skala argumen. Misalnya, DEC(5,2) kolom mengembalikan tipe DEC(38,2) data. AVG

Contoh

Temukan jumlah rata-rata yang terjual per transaksi dari SALES tabel.

```
select avg(qtysold)from sales;
```

Fungsi BOOL_AND

Fungsi BOOL_AND beroperasi pada satu kolom atau ekspresi Boolean atau integer. Fungsi ini menerapkan logika yang mirip dengan fungsi BIT_AND dan BIT_OR. Untuk fungsi ini, tipe kembali adalah nilai Boolean (`true`atau`false`).

Jika semua nilai dalam satu set adalah true, fungsi BOOL_AND mengembalikan `true` (). `t` Jika ada nilai palsu, fungsi mengembalikan `false` (`f`).

Sintaks

```
BOOL_AND ( [DISTINCT | ALL] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ekspresi ini harus memiliki tipe data BOOLEAN atau integer. Jenis kembali dari fungsi ini adalah BOOLEAN.

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat untuk ekspresi yang ditentukan sebelum menghitung hasilnya. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat. ALL adalah default.

Contoh-contoh

Anda dapat menggunakan fungsi Boolean terhadap ekspresi Boolean atau ekspresi integer.

Misalnya, query berikut mengembalikan hasil dari tabel USERS standar dalam database TICKIT, yang memiliki beberapa kolom Boolean.

Fungsi BOOL_AND kembali `false` untuk semua lima baris. Tidak semua pengguna di masing-masing negara bagian menyukai olahraga.

```
select state, bool_and(likesports) from users
group by state order by state limit 5;
```

```
state | bool_and
------+---------
AB    | f
AK    | f
AL    | f
AZ    | f
BC    | f
(5 rows)
```

Fungsi BOOL_OR

Fungsi BOOL_OR beroperasi pada satu kolom atau ekspresi Boolean atau integer. Fungsi ini menerapkan logika yang mirip dengan fungsi BIT_AND dan BIT_OR. Untuk fungsi ini, tipe kembali adalah nilai Boolean (`true`,`false`, or`NULL`).

Jika nilai dalam satu set adalah`true`, fungsi BOOL_OR mengembalikan `true` (). `t` Jika nilai dalam himpunan adalah`false`, fungsi mengembalikan `false` (`f`). NULL dapat dikembalikan jika nilainya tidak diketahui.

Sintaks

```
BOOL_OR ( [DISTINCT | ALL] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ekspresi ini harus memiliki tipe data BOOLEAN atau integer. Jenis kembali dari fungsi ini adalah BOOLEAN.

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat untuk ekspresi yang ditentukan sebelum menghitung hasilnya. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat. ALL adalah default.

Contoh-contoh

Anda dapat menggunakan fungsi Boolean dengan ekspresi Boolean atau ekspresi integer. Misalnya, query berikut mengembalikan hasil dari tabel USERS standar dalam database TICKIT, yang memiliki beberapa kolom Boolean.

Fungsi BOOL_OR kembali `true` untuk semua lima baris. Setidaknya satu pengguna di masing-masing negara bagian menyukai olahraga.

```
select state, bool_or(likesports) from users
group by state order by state limit 5;

state | bool_or
------+--------
AB    | t
AK    | t
AL    | t
AZ    | t
BC    | t
(5 rows)
```

Contoh berikut mengembalikan NULL.

```
SELECT BOOL_OR(NULL = '123')
               bool_or
------
NULL
```

COUNTdan COUNT DISTINCT fungsi

COUNTFungsi menghitung baris yang ditentukan oleh ekspresi. COUNT DISTINCTFungsi menghitung jumlah nilai non-Null yang berbeda dalam kolom atau ekspresi. Ini menghilangkan semua nilai duplikat dari ekspresi yang ditentukan sebelum melakukan penghitungan.

Sintaks

```
COUNT (column)
```

```
COUNT (DISTINCT column)
```

Pendapat

kolom

Kolom target tempat fungsi beroperasi.

Tipe Data

COUNTFungsi dan COUNT DISTINCT fungsi mendukung semua tipe data argumen.

COUNT DISTINCTFungsi kembaliBIGINT.

Contoh-contoh

Hitung semua pengguna dari negara bagian Florida.

```
select count (identifier) from users where state='FL';
```

Hitung semua ID tempat unik dari EVENT tabel.

```
select count (distinct (venueid)) as venues from event;
```

Fungsi COUNT

Fungsi COUNT menghitung baris yang ditentukan oleh ekspresi.

Fungsi COUNT memiliki variasi berikut.

- COUNT (*) menghitung semua baris dalam tabel target apakah mereka termasuk nol atau tidak.
- COUNT (ekspresi) menghitung jumlah baris dengan nilai non-Null dalam kolom atau ekspresi tertentu.
- COUNT (ekspresi DISTINCT) menghitung jumlah nilai non-Null yang berbeda dalam kolom atau ekspresi.
- PERKIRAAN COUNT DISTINCT mendekati jumlah nilai non-NULL yang berbeda dalam kolom atau ekspresi.

Sintaks

```
COUNT( * | expression )
```

```
COUNT ( [ DISTINCT | ALL ] expression )
```

```
APPROXIMATE COUNT ( DISTINCT expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Fungsi COUNT mendukung semua tipe data argumen.

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat dari ekspresi yang ditentukan sebelum melakukan penghitungan. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung. ALL adalah default.

KIRA-KIRA

Ketika digunakan dengan PERKIRAAN, fungsi COUNT DISTINCT menggunakan HyperLogLog algoritma untuk memperkirakan jumlah nilai non-NULL yang berbeda dalam kolom atau ekspresi. Kueri yang menggunakan kata kunci PERKIRAAN berjalan lebih cepat, dengan kesalahan relatif rendah sekitar 2%. Perkiraan dijamin untuk kueri yang mengembalikan sejumlah besar nilai berbeda, dalam jutaan atau lebih per kueri, atau per grup, jika ada klausa grup demi kelompok. Untuk set nilai berbeda yang lebih kecil, dalam ribuan, perkiraan mungkin lebih lambat daripada hitungan yang tepat. PERKIRAAN hanya dapat digunakan dengan COUNT DISTINCT.

Jenis pengembalian

Fungsi COUNT mengembalikan BIGINT.

Contoh-contoh

Hitung semua pengguna dari negara bagian Florida:

```
select count(*) from users where state='FL';

count
-------
510
```

Hitung semua nama acara dari tabel EVENT:

```
select count(eventname) from event;

count
```

```
-------
8798
```

Hitung semua nama acara dari tabel EVENT:

```
select count(all eventname) from event;

count
-------
8798
```

Hitung semua ID venue unik dari tabel EVENT:

```
select count(distinct venueid) as venues from event;

venues
--------
204
```

Hitung berapa kali setiap penjual mencantumkan batch lebih dari empat tiket untuk dijual. Kelompokkan hasil berdasarkan ID penjual:

```
select count(*), sellerid from listing
where numtickets > 4
group by sellerid
order by 1 desc, 2;

count | sellerid
------+----------
12    |    6386
11    |    17304
11    |    20123
11    |    25428
...
```

Contoh berikut membandingkan nilai pengembalian dan waktu eksekusi untuk COUNT dan PERKIRAAN COUNT.

```
select  count(distinct pricepaid) from sales;

count
```

```
-------
  4528


Time: 48.048 ms


select approximate count(distinct pricepaid) from sales;

count
-------
  4553


Time: 21.728 ms
```

Fungsi LISTAGG

Untuk setiap grup dalam kueri, fungsi agregat LISTAGG mengurutkan baris untuk grup tersebut sesuai dengan ekspresi ORDER BY, lalu menggabungkan nilai menjadi satu string.

LISTAGG adalah fungsi compute-node saja. Fungsi mengembalikan kesalahan jika kueri tidak mereferensikan tabel atau tabel AWS Clean Rooms sistem yang ditentukan pengguna.

Sintaks

```
LISTAGG( [DISTINCT] aggregate_expression [, 'delimiter' ] )
[ WITHIN GROUP (ORDER BY order_list) ]
```

Argumen

DISTINCT

(Opsional) Klausa yang menghilangkan nilai duplikat dari ekspresi yang ditentukan sebelum digabungkan. Spasi trailing diabaikan, sehingga string `'a'` dan `'a '` diperlakukan sebagai duplikat. LISTAGG menggunakan nilai pertama yang ditemui. Untuk informasi selengkapnya, lihat Signifikansi trailing blanko.

aggregate_expression

Ekspresi yang valid (seperti nama kolom) yang memberikan nilai untuk digabungkan. Nilai NULL dan string kosong diabaikan.

pembatas

(Opsional) Konstanta string untuk memisahkan nilai gabungan. Default-nya adalah NULL.

AWS Clean Rooms mendukung sejumlah spasi utama atau belakang di sekitar koma opsional atau titik dua serta string kosong atau sejumlah spasi.

Contoh nilai yang valid adalah:

```
', '
': '
' '
```

DALAM GRUP (PESANAN BERDASARKAN order_list)

(Opsional) Sebuah klausa yang menentukan urutan dari nilai agregat.

Pengembalian

VARCHAR (MAKS). Jika set hasil lebih besar dari ukuran VARCHAR maksimum (64K-1, atau 65535), maka LISTAGG mengembalikan kesalahan berikut:

```
Invalid operation: Result size exceeds LISTAGG limit
```

Catatan penggunaan

Jika pernyataan menyertakan beberapa fungsi LISTAGG yang menggunakan klausa WITHERE GROUP, setiap klausa WITHIN GROUP harus menggunakan nilai ORDER BY yang sama.

Misalnya, pernyataan berikut akan mengembalikan kesalahan.

```
select listagg(sellerid)
within group (order by dateid) as sellers,
listagg(dateid)
within group (order by sellerid) as dates
from winsales;
```

Pernyataan berikut akan berjalan dengan sukses.

```
select listagg(sellerid)
```

```
within group (order by dateid) as sellers,
listagg(dateid)
within group (order by dateid) as dates
from winsales;

select listagg(sellerid)
within group (order by dateid) as sellers,
listagg(dateid) as dates
from winsales;
```

Contoh-contoh

Contoh berikut menggabungkan ID penjual, diurutkan berdasarkan ID penjual.

```
select listagg(sellerid, ', ') within group (order by sellerid) from sales
where eventid = 4337;
listagg

-------------------------------------------------------------------------------------------
380, 380, 1178, 1178, 1178, 2731, 8117, 12905, 32043, 32043, 32043, 32432, 32432,
 38669, 38750, 41498, 45676, 46324, 47188, 47188, 48294
```

Contoh berikut menggunakan DISTINCT untuk mengembalikan daftar ID penjual unik.

```
select listagg(distinct sellerid, ', ') within group (order by sellerid) from sales
where eventid = 4337;

listagg

-------------------------------------------------------------------------------------------
380, 1178, 2731, 8117, 12905, 32043, 32432, 38669, 38750, 41498, 45676, 46324, 47188,
 48294
```

Contoh berikut menggabungkan ID penjual dalam urutan tanggal.

```
select listagg(sellerid)
within group (order by dateid)
from winsales;

   listagg
-------------
 31141242333
```

Contoh berikut mengembalikan daftar tanggal penjualan yang dipisahkan pipa untuk pembeli B.

```
select listagg(dateid,'|')
within group (order by sellerid desc,salesid asc)
from winsales
where buyerid  = 'b';

             listagg
-------------------------------------
2003-08-02|2004-04-18|2004-04-18|2004-02-12
```

Contoh berikut mengembalikan daftar ID penjualan yang dipisahkan koma untuk setiap ID pembeli.

```
select buyerid,
listagg(salesid,',')
within group (order by salesid) as sales_id
from winsales
group by buyerid
order by buyerid;

   buyerid | sales_id
-----------+------------------------
        a  |10005,40001,40005
        b  |20001,30001,30004,30003
        c  |10001,20002,30007,10006
```

Fungsi MAX

Fungsi MAX mengembalikan nilai maksimum dalam satu set baris. DISTINCT atau ALL dapat digunakan tetapi tidak mempengaruhi hasilnya.

Sintaks

```
MAX ( [ DISTINCT | ALL ] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ekspresi adalah salah satu tipe data berikut:

- SMALLINT

- INTEGER
- BIGINT
- DECIMAL
- REAL
- PRECISON GANDA
- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- JADWAL
- VARBYTE
- SUPER

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat dari ekspresi yang ditentukan sebelum menghitung maksimum. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung maksimum. ALL adalah default.

Tipe Data

Mengembalikan tipe data yang sama sebagai ekspresi.

Contoh-contoh

Temukan harga tertinggi yang dibayarkan dari semua penjualan:

```
select max(pricepaid) from sales;

max
----------
12624.00
(1 row)
```

Temukan harga tertinggi yang dibayarkan per tiket dari semua penjualan:

```
select max(pricepaid/qtysold) as max_ticket_price
from sales;

max_ticket_price
-----------------
2500.00000000
(1 row)
```

Fungsi MEDIAN

Menghitung nilai median untuk rentang nilai. Nilai NULL dalam rentang diabaikan.

MEDIAN adalah fungsi distribusi terbalik yang mengasumsikan model distribusi kontinu.

MEDIAN adalah kasus khusus PERSENTILE_CONT (.5).

MEDIAN adalah fungsi compute-node saja. Fungsi mengembalikan kesalahan jika kueri tidak mereferensikan tabel atau tabel AWS Clean Rooms sistem yang ditentukan pengguna.

Sintaks

```
MEDIAN ( median_expression )
```

Argumen

median_expression

Kolom target atau ekspresi tempat fungsi beroperasi.

Tipe Data

Jenis pengembalian ditentukan oleh tipe data median_expression. Tabel berikut menunjukkan tipe kembali untuk setiap tipe data median_expression.

Jenis masukan	Jenis pengembalian
NUMERIK, DESIMAL	DECIMAL
MENGAPUNG, GANDA	DOUBLE

Jenis masukan	Jenis pengembalian
DATE	DATE
TIMESTAMP	TIMESTAMP
TIMESTAMPTZ	TIMESTAMPTZ

Catatan penggunaan

Jika argumen median_expression adalah tipe data DECIMAL yang didefinisikan dengan presisi maksimum 38 digit, ada kemungkinan MEDIAN akan mengembalikan hasil yang tidak akurat atau kesalahan. Jika nilai pengembalian fungsi MEDIAN melebihi 38 digit, hasilnya terpotong agar sesuai, yang menyebabkan hilangnya presisi. Jika, selama interpolasi, hasil antara melebihi presisi maksimum, luapan numerik terjadi dan fungsi mengembalikan kesalahan. Untuk menghindari kondisi ini, sebaiknya gunakan tipe data dengan presisi lebih rendah atau mentransmisikan argumen median_expression ke presisi yang lebih rendah.

Jika pernyataan menyertakan beberapa panggilan ke fungsi agregat berbasis sortir (LISTAGG, PERCENTILE_CONT, atau MEDIAN), semuanya harus menggunakan nilai ORDER BY yang sama. Perhatikan bahwa MEDIAN menerapkan urutan implisit oleh pada nilai ekspresi.

Misalnya, pernyataan berikut mengembalikan kesalahan.

```
select top 10 salesid, sum(pricepaid),
percentile_cont(0.6) within group (order by salesid),
median (pricepaid)
from sales group by salesid, pricepaid;

An error occurred when executing the SQL command:
select top 10 salesid, sum(pricepaid),
percentile_cont(0.6) within group (order by salesid),
median (pricepaid)
from sales group by salesid, pricepai...

ERROR: within group ORDER BY clauses for aggregate functions must be the same
```

Pernyataan berikut berjalan dengan sukses.

```
select top 10 salesid, sum(pricepaid),
```

```
percentile_cont(0.6) within group (order by salesid),
median (salesid)
from sales group by salesid, pricepaid;
```

Contoh-contoh

Contoh berikut menunjukkan bahwa MEDIAN menghasilkan hasil yang sama seperti PERCENTILE_CONT (0.5).

```
select top 10  distinct sellerid, qtysold,
percentile_cont(0.5) within group (order by qtysold),
median (qtysold)
from sales
group by sellerid, qtysold;

sellerid | qtysold | percentile_cont | median
---------+---------+-----------------+-------
       1 |       1 |             1.0 |    1.0
       2 |       3 |             3.0 |    3.0
       5 |       2 |             2.0 |    2.0
       9 |       4 |             4.0 |    4.0
      12 |       1 |             1.0 |    1.0
      16 |       1 |             1.0 |    1.0
      19 |       2 |             2.0 |    2.0
      19 |       3 |             3.0 |    3.0
      22 |       2 |             2.0 |    2.0
      25 |       2 |             2.0 |    2.0
```

Fungsi MIN

Fungsi MIN mengembalikan nilai minimum dalam satu set baris. DISTINCT atau ALL dapat digunakan tetapi tidak mempengaruhi hasilnya.

Sintaks

```
MIN ( [ DISTINCT | ALL ] expression )
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi. Ekspresi adalah salah satu tipe data berikut:

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- REAL
- PRECISON GANDA
- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- JADWAL
- VARBYTE
- SUPER

BERBEDA | SEMUA

Dengan argumen DISTINCT, fungsi menghilangkan semua nilai duplikat dari ekspresi yang ditentukan sebelum menghitung minimum. Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung minimum. ALL adalah default.

Tipe Data

Mengembalikan tipe data yang sama sebagai ekspresi.

Contoh-contoh

Temukan harga terendah yang dibayarkan dari semua penjualan:

```
select min(pricepaid) from sales;

min
-------
20.00
```

```
(1 row)
```

Temukan harga terendah yang dibayarkan per tiket dari semua penjualan:

```
select min(pricepaid/qtysold)as min_ticket_price
from sales;

min_ticket_price
------------------
20.00000000
(1 row)
```

Fungsi PERCENTILE_CONT

PERCENTILE_CONT adalah fungsi distribusi terbalik yang mengasumsikan model distribusi kontinu. Dibutuhkan nilai persentil dan spesifikasi sortir, dan mengembalikan nilai interpolasi yang akan jatuh ke dalam nilai persentil yang diberikan sehubungan dengan spesifikasi sortir.

PERCENTILE_CONT menghitung interpolasi linier antara nilai setelah mengurutkannya. Menggunakan nilai persentil `(P)` dan jumlah baris bukan nol `(N)` dalam grup agregasi, fungsi menghitung nomor baris setelah mengurutkan baris sesuai dengan spesifikasi pengurutan. Nomor baris ini `(RN)` dihitung sesuai dengan `RN = (1+ (P*(N-1))` rumus. Hasil akhir dari fungsi agregat dihitung dengan interpolasi linier antara nilai-nilai dari baris pada nomor baris dan. `CRN = CEILING(RN)` `FRN = FLOOR(RN)`

Hasil akhirnya adalah sebagai berikut.

Jika `(CRN = FRN = RN)` maka hasilnya adalah `(value of expression from row at RN)`

Jika tidak, hasilnya adalah sebagai berikut:

`(CRN - RN) * (value of expression for row at FRN) + (RN - FRN) * (value of expression for row at CRN)`.

PERCENTILE_CONT adalah fungsi compute-node saja. Fungsi mengembalikan kesalahan jika kueri tidak mereferensikan tabel atau tabel AWS Clean Rooms sistem yang ditentukan pengguna.

Sintaks

```
PERCENTILE_CONT ( percentile )
```

```
WITHIN GROUP (ORDER BY expr)
```

Argumen

persentil

Konstanta numerik antara 0 dan 1. Null diabaikan dalam perhitungan.

DALAM GRUP (ORDER BY expr)

Menentukan nilai numerik atau tanggal/waktu untuk mengurutkan dan menghitung persentil atas.

Pengembalian

Tipe pengembalian ditentukan oleh tipe data ekspresi ORDER BY dalam klausa WITHIN GROUP. Tabel berikut menunjukkan tipe pengembalian untuk setiap tipe data ekspresi ORDER BY.

Jenis masukan	Jenis pengembalian
SMALLINT, INTEGER, BIGINT, NUMERIK, DESIMAL	DECIMAL
MENGAPUNG, GANDA	DOUBLE
DATE	DATE
TIMESTAMP	TIMESTAMP
TIMESTAMPTZ	TIMESTAMPTZ

Catatan penggunaan

Jika ekspresi ORDER BY adalah tipe data DECIMAL yang ditentukan dengan presisi maksimum 38 digit, ada kemungkinan bahwa PERCENTILE_CONT akan mengembalikan hasil yang tidak akurat atau kesalahan. Jika nilai pengembalian fungsi PERCENTILE_CONT melebihi 38 digit, hasilnya terpotong agar sesuai, yang menyebabkan hilangnya presisi.. Jika, selama interpolasi, hasil antara melebihi presisi maksimum, luapan numerik terjadi dan fungsi mengembalikan kesalahan. Untuk menghindari kondisi ini, sebaiknya gunakan tipe data dengan presisi lebih rendah atau mentransmisikan ekspresi ORDER BY ke presisi yang lebih rendah.

Jika pernyataan menyertakan beberapa panggilan ke fungsi agregat berbasis sortir (LISTAGG, PERCENTILE_CONT, atau MEDIAN), semuanya harus menggunakan nilai ORDER BY yang sama. Perhatikan bahwa MEDIAN menerapkan urutan implisit oleh pada nilai ekspresi.

Misalnya, pernyataan berikut mengembalikan kesalahan.

```
select top 10 salesid, sum(pricepaid),
percentile_cont(0.6) within group (order by salesid),
median (pricepaid)
from sales group by salesid, pricepaid;

An error occurred when executing the SQL command:
select top 10 salesid, sum(pricepaid),
percentile_cont(0.6) within group (order by salesid),
median (pricepaid)
from sales group by salesid, pricepai...

ERROR: within group ORDER BY clauses for aggregate functions must be the same
```

Pernyataan berikut berjalan dengan sukses.

```
select top 10 salesid, sum(pricepaid),
percentile_cont(0.6) within group (order by salesid),
median (salesid)
from sales group by salesid, pricepaid;
```

Contoh-contoh

Contoh berikut menunjukkan bahwa MEDIAN menghasilkan hasil yang sama seperti PERCENTILE_CONT (0.5).

```
select top 10  distinct sellerid, qtysold,
percentile_cont(0.5) within group (order by qtysold),
median (qtysold)
from sales
group by sellerid, qtysold;

sellerid | qtysold | percentile_cont | median
---------+---------+-----------------+-------
       1 |       1 |             1.0 |    1.0
       2 |       3 |             3.0 |    3.0
       5 |       2 |             2.0 |    2.0
```

```
 9 |       4 |             4.0 |    4.0
12 |       1 |             1.0 |    1.0
16 |       1 |             1.0 |    1.0
19 |       2 |             2.0 |    2.0
19 |       3 |             3.0 |    3.0
22 |       2 |             2.0 |    2.0
25 |       2 |             2.0 |    2.0
```

Fungsi STDDEV_SAMP dan STDDEV_POP

Fungsi STDDEV_SAMP dan STDDEV_POP mengembalikan sampel dan standar deviasi populasi dari satu set nilai numerik (integer, desimal, atau floating-point). Hasil dari fungsi STDDEV_SAMP setara dengan akar kuadrat dari varians sampel dari kumpulan nilai yang sama.

STDDEV_SAMP dan STDDEV adalah sinonim untuk fungsi yang sama.

Sintaks

```
STDDEV_SAMP | STDDEV ( [ DISTINCT | ALL ] expression)
STDDEV_POP ( [ DISTINCT | ALL ] expression)
```

Ekspresi harus memiliki tipe data integer, desimal, atau floating point. Terlepas dari tipe data ekspresi, tipe pengembalian fungsi ini adalah angka presisi ganda.

> **Note**
>
> Standar deviasi dihitung menggunakan aritmatika floating point, yang dapat mengakibatkan sedikit ketidaktepatan.

Catatan penggunaan

Ketika standar deviasi sampel (STDDEV atau STDDEV_SAMP) dihitung untuk ekspresi yang terdiri dari satu nilai, hasil fungsinya adalah NULL bukan 0.

Contoh-contoh

Kueri berikut mengembalikan rata-rata nilai di kolom VENUESEATS dari tabel VENUE, diikuti oleh standar deviasi sampel dan standar deviasi populasi dari kumpulan nilai yang sama. VENUESEATS adalah kolom INTEGER. Skala hasil dikurangi menjadi 2 digit.

```
select avg(venueseats),
cast(stddev_samp(venueseats) as dec(14,2)) stddevsamp,
cast(stddev_pop(venueseats) as dec(14,2)) stddevpop
from venue;

avg  | stddevsamp | stddevpop
-------+------------+-----------
17503 |   27847.76 |  27773.20
(1 row)
```

Kueri berikut mengembalikan standar deviasi sampel untuk kolom KOMISI dalam tabel PENJUALAN. KOMISI adalah kolom DESIMAL. Skala hasilnya dikurangi menjadi 10 digit.

```
select cast(stddev(commission) as dec(18,10))
from sales;

stddev
----------------
130.3912659086
(1 row)
```

Kueri berikut menampilkan standar deviasi sampel untuk kolom COMMISSION sebagai integer.

```
select cast(stddev(commission) as integer)
from sales;

stddev
--------
130
(1 row)
```

Kueri berikut mengembalikan standar deviasi sampel dan akar kuadrat dari varians sampel untuk kolom KOMISI. Hasil perhitungan ini sama.

```
select
cast(stddev_samp(commission) as dec(18,10)) stddevsamp,
cast(sqrt(var_samp(commission)) as dec(18,10)) sqrtvarsamp
from sales;

stddevsamp   |  sqrtvarsamp
---------------+----------------
```

```
130.3912659086 | 130.3912659086
(1 row)
```

SUMdan SUM DISTINCT fungsi

SUMFungsi mengembalikan jumlah kolom input atau nilai ekspresi. SUMFungsi ini bekerja dengan nilai numerik dan mengabaikan NULL nilai.

SUM DISTINCTFungsi menghilangkan semua nilai duplikat dari ekspresi yang ditentukan sebelum menghitung jumlah.

Sintaks

```
SUM (column)
```

```
SUM (DISTINCT column )
```

Pendapat

kolom

Kolom target tempat fungsi beroperasi. Kolom adalah salah satu tipe data berikut:

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- DOUBLE

Tipe Data

Tipe argumen yang didukung oleh SUM fungsi adalahSMALLINT,INTEGER,BIGINT,DECIMAL, danDOUBLE.

SUMFungsi ini mendukung jenis pengembalian berikut:

- BIGINTuntukBIGINT,SMALLINT, dan INTEGER argumen

- DOUBLEuntuk argumen floating point
- Mengembalikan tipe data yang sama sebagai ekspresi untuk jenis argumen lainnya

Presisi default untuk hasil SUM fungsi dengan DECIMAL argumen adalah 38. Skala hasilnya sama dengan skala argumen. Misalnya, DEC(5,2) kolom mengembalikan tipe DEC(38,2) data. SUM

Contoh-contoh

Temukan jumlah semua komisi yang dibayarkan dari SALES tabel.

```
select sum(commission) from sales
```

Temukan jumlah semua komisi berbeda yang dibayarkan dari SALES tabel.

```
select sum (distinct (commission)) from sales
```

Fungsi VAR_SAMP dan VAR_POP

Fungsi VAR_SAMP dan VAR_POP mengembalikan sampel dan varians populasi dari sekumpulan nilai numerik (integer, desimal, atau floating-point). Hasil dari fungsi VAR_SAMP setara dengan standar deviasi sampel kuadrat dari kumpulan nilai yang sama.

VAR_SAMP dan VARIANCE adalah sinonim untuk fungsi yang sama.

Sintaks

```
VAR_SAMP | VARIANCE ( [ DISTINCT | ALL ] expression)
VAR_POP ( [ DISTINCT | ALL ] expression)
```

Ekspresi harus memiliki tipe data integer, desimal, atau floating-point. Terlepas dari tipe data ekspresi, tipe pengembalian fungsi ini adalah angka presisi ganda.

> **Note**
>
> Hasil dari fungsi-fungsi ini mungkin bervariasi di seluruh cluster data warehouse, tergantung pada konfigurasi cluster dalam setiap kasus.

Catatan penggunaan

Ketika varians sampel (VARIANCE atau VAR_SAMP) dihitung untuk ekspresi yang terdiri dari satu nilai, hasil fungsinya adalah NULL bukan 0.

Contoh-contoh

Kueri berikut mengembalikan sampel bulat dan varians populasi kolom NUMTICKETS dalam tabel LISTING.

```
select avg(numtickets),
round(var_samp(numtickets)) varsamp,
round(var_pop(numtickets)) varpop
from listing;

avg | varsamp | varpop
-----+---------+--------
10 |      54 |     54
(1 row)
```

Kueri berikut menjalankan perhitungan yang sama tetapi melemparkan hasilnya ke nilai desimal.

```
select avg(numtickets),
cast(var_samp(numtickets) as dec(10,4)) varsamp,
cast(var_pop(numtickets) as dec(10,4)) varpop
from listing;

avg | varsamp | varpop
-----+---------+---------
10 | 53.6291 | 53.6288
(1 row)
```

Fungsi array

Bagian ini menjelaskan fungsi array untuk SQL yang didukung diAWS Clean Rooms.

Topik

- fungsi array
- fungsi array_concat

- fungsi array_flatten
- fungsi get_array_length
- fungsi split_to_array
- fungsi subarray

fungsi array

Membuat array tipe data SUPER.

Sintaksis

```
ARRAY( [ expr1 ] [ , expr2 [ , ... ] ] )
```

Pendapat

expr1, expr2

Ekspresi tipe data apa pun kecuali tipe tanggal dan waktu. Argumen tidak harus dari tipe data yang sama.

Jenis pengembalian

Fungsi array mengembalikan tipe data SUPER.

Contoh

Contoh berikut menunjukkan array nilai numerik dan array tipe data yang berbeda.

```
--an array of numeric values
select array(1,50,null,100);
      array
------------------
 [1,50,null,100]
(1 row)

--an array of different data types
select array(1,'abc',true,3.14);
        array
```

```
-----------------------
 [1,"abc",true,3.14]
(1 row)
```

fungsi array_concat

Fungsi array_concat menggabungkan dua array untuk membuat array yang berisi semua elemen dalam array pertama diikuti oleh semua elemen dalam array kedua. Kedua argumen tersebut harus berupa array yang valid.

Sintaksis

```
array_concat( super_expr1,  super_expr2 )
```

Pendapat

super_expr1

Nilai yang menentukan yang pertama dari dua array untuk digabungkan.

super_expr2

Nilai yang menentukan kedua dari dua array untuk digabungkan.

Jenis pengembalian

Fungsi array_concat mengembalikan nilai data SUPER.

Contoh

Contoh berikut menunjukkan penggabungan dua array dari jenis yang sama dan penggabungan dua array dari jenis yang berbeda.

```
-- concatenating two arrays
SELECT ARRAY_CONCAT(ARRAY(10001,10002),ARRAY(10003,10004));
              array_concat
-----------------------------------
 [10001,10002,10003,10004]
(1 row)
```

```
-- concatenating two arrays of different types
SELECT ARRAY_CONCAT(ARRAY(10001,10002),ARRAY('ab','cd'));
          array_concat
-----------------------------
 [10001,10002,"ab","cd"]
(1 row)
```

fungsi array_flatten

Menggabungkan beberapa array ke dalam satu array tipe SUPER.

Sintaksis

```
array_flatten( super_expr1,super_expr2,.. )
```

Pendapat

super_expr1, super_expr2

Ekspresi SUPER yang valid dari bentuk array.

Jenis pengembalian

Fungsi array_flatten mengembalikan nilai data SUPER.

Contoh

Contoh berikut menunjukkan fungsi array_flatten.

```
SELECT ARRAY_FLATTEN(ARRAY(ARRAY(1,2,3,4),ARRAY(5,6,7,8),ARRAY(9,10)));
     array_flatten
-----------------------
 [1,2,3,4,5,6,7,8,9,10]
(1 row)
```

fungsi get_array_length

Mengembalikan panjang array yang ditentukan. Fungsi GET_ARRAY_LENGTH mengembalikan panjang array SUPER diberikan objek atau jalur array.

Sintaksis

```
get_array_length( super_expr )
```

Pendapat

super_expr

Ekspresi SUPER yang valid dari bentuk array.

Jenis pengembalian

Fungsi get_array_length mengembalikan BIGINT.

Contoh

Contoh berikut menunjukkan fungsi get_array_length.

```
SELECT GET_ARRAY_LENGTH(ARRAY(1,2,3,4,5,6,7,8,9,10));
 get_array_length
----------------------
            10
(1 row)
```

fungsi split_to_array

Menggunakan pembatas sebagai parameter opsional. Jika tidak ada pembatas, maka defaultnya adalah koma.

Sintaksis

```
split_to_array( string,delimiter )
```

Pendapat

tali

String input yang akan dibagi.

pembatas

Nilai opsional di mana string input akan dibagi. Default adalah koma.

Jenis pengembalian

Fungsi split_to_array mengembalikan nilai data SUPER.

Contoh

Contoh berikut menunjukkan fungsi split_to_array.

```
SELECT SPLIT_TO_ARRAY('12|345|6789', '|');
     split_to_array
------------------------
 ["12","345","6789"]
(1 row)
```

fungsi subarray

Memanipulasi array untuk mengembalikan subset dari array input.

Sintaksis

```
SUBARRAY( super_expr, start_position, length )
```

Pendapat

super_expr

Ekspresi SUPER valid dalam bentuk array.

start_position

Posisi dalam array untuk memulai ekstraksi, dimulai dari posisi indeks 0. Posisi negatif dihitung mundur dari akhir array.

panjang

Jumlah elemen untuk mengekstrak (panjang substring).

Jenis pengembalian

Fungsi subarray mengembalikan nilai data SUPER.

Contoh

Berikut ini adalah contoh dari fungsi subarray.

```
 SELECT SUBARRAY(ARRAY('a', 'b', 'c', 'd', 'e', 'f'), 2, 3);
   subarray
---------------
 ["c","d","e"]
(1 row)
```

Ekspresi bersyarat

AWS Clean Rooms mendukung ekspresi bersyarat berikut:

Topik

- Ekspresi bersyarat CASE
- COALESCEekspresi
- Fungsi TERBESAR dan PALING KECIL
- Fungsi NVL dan COALESCE
- Fungsi NVL2
- Fungsi NULLIF

Ekspresi bersyarat CASE

Ekspresi CASE adalah ekspresi bersyarat yang serupa dengan pernyataan if/then/else yang ditemukan dalam bahasa lain. CASE digunakan untuk menentukan hasil jika terdapat beberapa kondisi. Gunakan CASE di mana ekspresi SQL valid, seperti dalam perintah SELECT.

Ada dua jenis ekspresi CASE: sederhana dan dicari.

- Dalam ekspresi CASE sederhana, ekspresi dibandingkan dengan nilai. Ketika kecocokan ditemukan, tindakan yang ditentukan dalam klausul THEN diterapkan. Jika tidak ada kecocokan ditemukan, tindakan dalam klausul ELSE diterapkan.

- Dalam ekspresi CASE yang dicari, setiap CASE dievaluasi berdasarkan ekspresi Boolean, dan pernyataan CASE mengembalikan CASE yang cocok pertama. Jika tidak ada kecocokan yang ditemukan di antara klausa WHEN, tindakan dalam klausa ELSE dikembalikan.

Sintaks

Pernyataan CASE sederhana yang digunakan untuk menyesuaikan kondisi:

```
CASE expression
  WHEN value THEN result
  [WHEN...]
  [ELSE result]
END
```

Pernyataan CASE yang dicari digunakan untuk mengevaluasi setiap kondisi:

```
CASE
  WHEN condition THEN result
  [WHEN ...]
  [ELSE result]
END
```

Argumen

ekspresi

Nama kolom atau ekspresi yang valid.

value

Nilai yang dibandingkan dengan ekspresi, seperti konstanta numerik atau string karakter.

hasil

Nilai target atau ekspresi yang dikembalikan ketika ekspresi atau kondisi Boolean dievaluasi. Tipe data dari semua ekspresi hasil harus dikonversi ke tipe output tunggal.

ketentuan

Ekspresi Boolean yang mengevaluasi benar atau salah. Jika kondisi benar, nilai ekspresi CASE adalah hasil yang mengikuti kondisi, dan sisa ekspresi CASE tidak diproses. Jika kondisinya

salah, klausa WHEN berikutnya dievaluasi. Jika tidak ada hasil kondisi WHEN yang benar, nilai ekspresi CASE adalah hasil dari klausa ELSE. Jika klausa ELSE dihilangkan dan tidak ada kondisi yang benar, hasilnya adalah nol.

Contoh-contoh

Gunakan ekspresi CASE sederhana untuk mengganti `New York City` `Big Apple` dengan query terhadap tabel VENUE. Ganti semua nama kota lainnya dengan `other`.

```
select venuecity,
  case venuecity
    when 'New York City'
    then 'Big Apple' else 'other'
  end
from venue
order by venueid desc;

venuecity        |   case
-----------------+-----------
Los Angeles      | other
New York City    | Big Apple
San Francisco    | other
Baltimore        | other
...
```

Gunakan ekspresi CASE yang dicari untuk menetapkan nomor grup berdasarkan nilai PRICEPAID untuk penjualan tiket individu:

```
select pricepaid,
  case when pricepaid <10000 then 'group 1'
    when pricepaid >10000 then 'group 2'
    else 'group 3'
  end
from sales
order by 1 desc;

pricepaid |  case
----------+---------
12624     | group 2
10000     | group 3
10000     | group 3
```

```
9996      | group 1
9988      | group 1
...
```

COALESCEekspresi

COALESCEEkspresi mengembalikan nilai ekspresi pertama dalam daftar yang tidak null. Jika semua ekspresi nol, hasilnya adalah null. Ketika nilai non-null ditemukan, ekspresi yang tersisa dalam daftar tidak dievaluasi.

Jenis ekspresi ini berguna ketika Anda ingin mengembalikan nilai cadangan untuk sesuatu ketika nilai yang diinginkan hilang atau nol. Misalnya, kueri mungkin mengembalikan salah satu dari tiga nomor telepon (sel, rumah, atau kantor, dalam urutan itu), mana yang ditemukan pertama kali dalam tabel (bukan nol).

Sintaks

```
COALESCE (expression, expression, ... )
```

Contoh-contoh

Terapkan COALESCE ekspresi ke dua kolom.

```
select coalesce(start_date, end_date)
from datetable
order by 1;
```

Nama kolom default untuk ekspresi NVL adalah. COALESCE Query berikut mengembalikan hasil yang sama.

```
select coalesce(start_date, end_date) from datetable order by 1;
```

Fungsi TERBESAR dan PALING KECIL

Mengembalikan nilai terbesar atau terkecil dari daftar sejumlah ekspresi.

Sintaks

```
GREATEST (value [, ...])
```

```
LEAST (value [, ...])
```

Parameter

expression_list

Daftar ekspresi yang dipisahkan koma, seperti nama kolom. Ekspresi semua harus dikonversi ke tipe data umum. Nilai NULL dalam daftar diabaikan. Jika semua ekspresi mengevaluasi ke NULL, hasilnya adalah NULL.

Pengembalian

Mengembalikan nilai terbesar (untuk GREATEST) atau least (untuk LEAST) dari daftar ekspresi yang disediakan.

Contoh

Contoh berikut mengembalikan nilai tertinggi menurut abjad untuk `firstname` atau. `lastname`

```
select firstname, lastname, greatest(firstname,lastname) from users
where userid < 10
order by 3;

 firstname | lastname  | greatest
-----------+-----------+-----------
 Alejandro | Rosalez   | Ratliff
 Carlos    | Salazar   | Carlos
 Jane      | Doe       | Doe
 John      | Doe       | Doe
 John      | Stiles    | John
 Shirley   | Rodriguez | Rodriguez
 Terry     | Whitlock  | Terry
 Richard   | Roe       | Richard
 Xiulan    | Wang      | Wang
(9 rows)
```

Fungsi NVL dan COALESCE

Mengembalikan nilai ekspresi pertama yang tidak null dalam serangkaian ekspresi. Ketika nilai non-null ditemukan, ekspresi yang tersisa dalam daftar tidak dievaluasi.

NVL identik dengan COALESCE. Mereka adalah sinonim. Topik ini menjelaskan sintaks dan berisi contoh untuk keduanya.

Sintaks

```
NVL( expression, expression, ... )
```

Sintaks untuk COALESCE adalah sama:

```
COALESCE( expression, expression, ... )
```

Jika semua ekspresi nol, hasilnya adalah null.

Fungsi-fungsi ini berguna ketika Anda ingin mengembalikan nilai sekunder ketika nilai primer hilang atau null. Misalnya, kueri mungkin mengembalikan yang pertama dari tiga nomor telepon yang tersedia: ponsel, rumah, atau kantor. Urutan ekspresi dalam fungsi menentukan urutan evaluasi.

Argumen

ekspresi

Ekspresi, seperti nama kolom, yang akan dievaluasi untuk status null.

Jenis pengembalian

AWS Clean Rooms menentukan tipe data dari nilai yang dikembalikan berdasarkan ekspresi input. Jika tipe data dari ekspresi input tidak memiliki tipe umum, maka kesalahan dikembalikan.

Contoh-contoh

Jika daftar berisi ekspresi integer, fungsi mengembalikan integer.

```
SELECT COALESCE(NULL, 12, NULL);

coalesce
--------------
12
```

Contoh ini, yang sama dengan contoh sebelumnya, kecuali bahwa ia menggunakan NVL, mengembalikan hasil yang sama.

```
SELECT NVL(NULL, 12, NULL);

coalesce
--------------
12
```

Contoh berikut mengembalikan tipe string.

```
SELECT COALESCE(NULL, 'AWS Clean Rooms', NULL);

coalesce
--------------
AWS Clean Rooms
```

Contoh berikut menghasilkan kesalahan karena tipe data bervariasi dalam daftar ekspresi. Dalam hal ini, ada tipe string dan tipe angka dalam daftar.

```
SELECT COALESCE(NULL, 'AWS Clean Rooms', 12);
ERROR: invalid input syntax for integer: "AWS Clean Rooms"
```

Fungsi NVL2

Mengembalikan salah satu dari dua nilai berdasarkan apakah ekspresi tertentu mengevaluasi ke NULL atau TIDAK NULL.

Sintaks

```
NVL2 ( expression, not_null_return_value, null_return_value )
```

Argumen

ekspresi

Ekspresi, seperti nama kolom, yang akan dievaluasi untuk status null.

not_null_return_value

Nilai yang dikembalikan jika ekspresi mengevaluasi ke NOT NULL. Nilai not_null_return_value harus memiliki tipe data yang sama dengan ekspresi atau secara implisit dapat dikonversi ke tipe data tersebut.

null_return_value

Nilai yang dikembalikan jika ekspresi mengevaluasi ke NULL. Nilai null_return_value harus memiliki tipe data yang sama dengan ekspresi atau secara implisit dapat dikonversi ke tipe data tersebut.

Jenis pengembalian

Jenis pengembalian NVL2 ditentukan sebagai berikut:

- Jika not_null_return_value atau null_return_value adalah null, tipe data dari ekspresi not-null dikembalikan.

Jika kedua not_null_return_value dan null_return_value tidak null:

- Jika not_null_return_value dan null_return_value memiliki tipe data yang sama, tipe data tersebut dikembalikan.
- Jika not_null_return_value dan null_return_value memiliki tipe data numerik yang berbeda, tipe data numerik terkecil yang kompatibel dikembalikan.
- Jika not_null_return_value dan null_return_value memiliki tipe data datetime yang berbeda, tipe data stempel waktu dikembalikan.
- Jika not_null_return_value dan null_return_value memiliki tipe data karakter yang berbeda, tipe data not_null_return_value dikembalikan.
- Jika not_null_return_value dan null_return_value memiliki tipe data numerik dan non-numerik campuran, tipe data not_null_return_value dikembalikan.

> **Important**
>
> Dalam dua kasus terakhir di mana tipe data not_null_return_value dikembalikan, null_return_value secara implisit dilemparkan ke tipe data tersebut. Jika tipe data tidak kompatibel, fungsi gagal.

Catatan penggunaan

Untuk NVL2, pengembalian akan memiliki nilai parameter not_null_return_value atau null_return_value, mana yang dipilih oleh fungsi, tetapi akan memiliki tipe data not_null_return_value.

Misalnya, dengan asumsi kolom1 adalah NULL, kueri berikut akan mengembalikan nilai yang sama. Namun, tipe data nilai pengembalian DECODE adalah INTEGER dan tipe data nilai pengembalian NVL2 akan menjadi VARCHAR.

```
select decode(column1, null, 1234, '2345');
select nvl2(column1, '2345', 1234);
```

Contoh

Contoh berikut memodifikasi beberapa data sampel, kemudian mengevaluasi dua bidang untuk memberikan informasi kontak yang sesuai bagi pengguna:

```
update users set email = null where firstname = 'Aphrodite' and lastname = 'Acevedo';

select (firstname + ' ' + lastname) as name,
nvl2(email, email, phone) AS contact_info
from users
where state = 'WA'
and lastname  like 'A%'
order by lastname, firstname;

name        contact_info
--------------------+--------------------------------------------
Aphrodite Acevedo (555) 555-0100
Caldwell Acevedo  Nunc.sollicitudin@example.ca
Quinn Adams     vel@example.com
Kamal Aguilar   quis@example.com
Samson Alexander  hendrerit.neque@example.com
Hall Alford     ac.mattis@example.com
Lane Allen      et.netus@example.com
Xander Allison    ac.facilisis.facilisis@example.com
Amaya Alvarado    dui.nec.tempus@example.com
Vera Alvarez    at.arcu.Vestibulum@example.com
Yetta Anthony   enim.sit@example.com
Violet Arnold   ad.litora@example.comm
August Ashley   consectetuer.euismod@example.com
Karyn Austin    ipsum.primis.in@example.com
Lucas Ayers     at@example.com
```

Fungsi NULLIF

Sintaks

Ekspresi NULLIF membandingkan dua argumen dan mengembalikan null jika argumennya sama. Jika mereka tidak sama, argumen pertama dikembalikan. Ekspresi ini adalah kebalikan dari ekspresi NVL atau COALESCE.

```
NULLIF ( expression1, expression2 )
```

Argumen

ekspresi1, ekspresi2

Kolom target atau ekspresi yang dibandingkan. Tipe pengembalian sama dengan tipe ekspresi pertama. Nama kolom default hasil NULLIF adalah nama kolom dari ekspresi pertama.

Contoh-contoh

Dalam contoh berikut, query mengembalikan string `first` karena argumen tidak sama.

```
SELECT NULLIF('first', 'second');

case
-------
first
```

Dalam contoh berikut, query kembali NULL karena argumen literal string sama.

```
SELECT NULLIF('first', 'first');

case
-------
NULL
```

Dalam contoh berikut, query kembali 1 karena argumen integer tidak sama.

```
SELECT NULLIF(1, 2);

case
```

```
-------
1
```

Dalam contoh berikut, query kembali NULL karena argumen integer sama.

```
SELECT NULLIF(1, 1);

case
-------
NULL
```

Dalam contoh berikut, query mengembalikan null ketika nilai LISTID dan SALESID cocok:

```
select nullif(listid,salesid), salesid
from sales where salesid<10 order by 1, 2 desc;

listid  | salesid
--------+---------
     4  |       2
     5  |       4
     5  |       3
     6  |       5
     10 |       9
     10 |       8
     10 |       7
     10 |       6
        |       1
(9 rows)
```

Fungsi pemformatan tipe data

Menggunakan fungsi pemformatan tipe data, Anda dapat mengonversi nilai dari satu tipe data ke tipe data lainnya. Untuk masing-masing fungsi ini, argumen pertama selalu nilai yang akan diformat dan argumen kedua berisi template untuk format baru. AWS Clean Rooms mendukung beberapa fungsi pemformatan tipe data.

Topik

- Fungsi CAST
- Fungsi CONVERT
- TO_CHAR

- Fungsi TO_DATE
- TO_NUMBER
- String format datetime
- String format numerik
- Teradata-gaya pemformatan karakter untuk data numerik

Fungsi CAST

Fungsi CAST mengubah satu tipe data ke tipe data lain yang kompatibel. Misalnya, Anda dapat mengonversi string ke tanggal, atau tipe numerik menjadi string. CAST melakukan konversi runtime, yang berarti konversi tidak mengubah tipe data nilai dalam tabel sumber. Itu hanya berubah dalam konteks kueri.

Fungsi CAST sangat mirip denganthe section called “MENGUBAH”, karena keduanya mengonversi satu tipe data ke tipe data lainnya, tetapi mereka disebut berbeda.

Tipe data tertentu memerlukan konversi eksplisit ke tipe data lain menggunakan fungsi CAST atau CONVERT. Tipe data lain dapat dikonversi secara implisit, sebagai bagian dari perintah lain, tanpa menggunakan CAST atau CONVERT. Lihat Ketik kompatibilitas dan konversi.

Sintaks

Gunakan salah satu dari dua bentuk sintaks yang setara ini untuk mentransmisikan ekspresi dari satu tipe data ke tipe data lainnya.

```
CAST ( expression AS type )
expression :: type
```

Argumen

ekspresi

Ekspresi yang mengevaluasi satu atau lebih nilai, seperti nama kolom atau literal. Mengkonversi nilai null mengembalikan nol. Ekspresi tidak dapat berisi string kosong atau kosong.

jenis

Salah satu yang didukungJenis Data, kecuali untuk tipe data VARBYTE, BINARY, dan BINARY VARY VARY.

Jenis pengembalian

CAST mengembalikan tipe data yang ditentukan oleh argumen tipe.

> **Note**
>
> AWS Clean Rooms mengembalikan kesalahan jika Anda mencoba untuk melakukan konversi bermasalah, seperti konversi DECIMAL yang kehilangan presisi, seperti berikut ini:
>
> ```
> select 123.456::decimal(2,1);
> ```
>
> atau konversi INTEGER yang menyebabkan overflow:
>
> ```
> select 12345678::smallint;
> ```

Contoh-contoh

Dua pertanyaan berikut adalah setara. Keduanya memberikan nilai desimal ke bilangan bulat:

```
select cast(pricepaid as integer)
from sales where salesid=100;

pricepaid
-----------
162
(1 row)
```

```
select pricepaid::integer
from sales where salesid=100;

pricepaid
-----------
162
(1 row)
```

Berikut ini menghasilkan hasil yang serupa. Tidak memerlukan data sampel untuk dijalankan:

```
select cast(162.00 as integer) as pricepaid;
```

```
pricepaid
-----------
162
(1 row)
```

Dalam contoh ini, nilai dalam kolom stempel waktu dilemparkan sebagai tanggal, yang mengakibatkan penghapusan waktu dari setiap hasil:

```
select cast(saletime as date), salesid
from sales order by salesid limit 10;

 saletime  | salesid
-----------+---------
2008-02-18 |       1
2008-06-06 |       2
2008-06-06 |       3
2008-06-09 |       4
2008-08-31 |       5
2008-07-16 |       6
2008-06-26 |       7
2008-07-10 |       8
2008-07-22 |       9
2008-08-06 |      10

(10 rows)
```

Jika Anda tidak menggunakan CAST seperti yang diilustrasikan dalam sampel sebelumnya, hasilnya akan mencakup waktu: 2008-02-18 02:36:48.

Kueri berikut melemparkan data karakter variabel ke tanggal. Tidak memerlukan data sampel untuk dijalankan.

```
select cast('2008-02-18 02:36:48' as date) as mysaletime;

mysaletime
--------------------
2008-02-18
(1 row)
```

Dalam contoh ini, nilai dalam kolom tanggal dilemparkan sebagai stempel waktu:

```
select cast(caldate as timestamp), dateid
```

```
from date order by dateid limit 10;

      caldate       | dateid
--------------------+--------
2008-01-01 00:00:00 |   1827
2008-01-02 00:00:00 |   1828
2008-01-03 00:00:00 |   1829
2008-01-04 00:00:00 |   1830
2008-01-05 00:00:00 |   1831
2008-01-06 00:00:00 |   1832
2008-01-07 00:00:00 |   1833
2008-01-08 00:00:00 |   1834
2008-01-09 00:00:00 |   1835
2008-01-10 00:00:00 |   1836

(10 rows)
```

Dalam kasus seperti sampel sebelumnya, Anda dapat memperoleh kontrol tambahan atas pemformatan output dengan menggunakanTO_CHAR.

Dalam contoh ini, bilangan bulat dilemparkan sebagai string karakter:

```
select cast(2008 as char(4));

bpchar
--------
2008
```

Dalam contoh ini, nilai DECIMAL (6,3) dilemparkan sebagai nilai DECIMAL (4,1):

```
select cast(109.652 as decimal(4,1));

numeric
---------
109.7
```

Contoh ini menunjukkan ekspresi yang lebih kompleks. Kolom PRICEPAID (kolom DECIMAL (8,2)) dalam tabel PENJUALAN dikonversi ke kolom DECIMAL (38,2) dan nilainya dikalikan dengan 100000000000000000000:

```
select salesid, pricepaid::decimal(38,2)*100000000000000000000
as value from sales where salesid<10 order by salesid;
```

```
 salesid |           value
---------+--------------------------
       1 | 728000000000000000000.00
       2 |  760000000000000000000.00
       3 | 350000000000000000000.00
       4 | 175000000000000000000.00
       5 | 154000000000000000000.00
       6 | 394000000000000000000.00
       7 | 788000000000000000000.00
       8 | 197000000000000000000.00
       9 | 591000000000000000000.00

(9 rows)
```

Fungsi CONVERT

SepertiFungsi CAST, fungsi CONVERT mengubah satu tipe data ke tipe data lain yang kompatibel. Misalnya, Anda dapat mengonversi string ke tanggal, atau tipe numerik menjadi string. CONVERT melakukan konversi runtime, yang berarti konversi tidak mengubah tipe data nilai dalam tabel sumber. Itu hanya berubah dalam konteks kueri.

Tipe data tertentu memerlukan konversi eksplisit ke tipe data lain menggunakan fungsi CONVERT. Tipe data lain dapat dikonversi secara implisit, sebagai bagian dari perintah lain, tanpa menggunakan CAST atau CONVERT. Lihat Ketik kompatibilitas dan konversi.

Sintaks

```
CONVERT ( type, expression )
```

Argumen

jenis

Salah satu yang didukungJenis Data, kecuali untuk tipe data VARBYTE, BINARY, dan BINARY VARY VARY.

ekspresi

Ekspresi yang mengevaluasi satu atau lebih nilai, seperti nama kolom atau literal. Mengkonversi nilai null mengembalikan nol. Ekspresi tidak dapat berisi string kosong atau kosong.

Jenis pengembalian

CONVERT mengembalikan tipe data yang ditentukan oleh argumen tipe.

> **Note**
>
> AWS Clean Rooms mengembalikan kesalahan jika Anda mencoba untuk melakukan konversi bermasalah, seperti konversi DECIMAL yang kehilangan presisi, seperti berikut ini:
>
> ```
> SELECT CONVERT(decimal(2,1), 123.456);
> ```
>
> atau konversi INTEGER yang menyebabkan overflow:
>
> ```
> SELECT CONVERT(smallint, 12345678);
> ```

Contoh-contoh

Query berikut menggunakan fungsi CONVERT untuk mengubah kolom desimal menjadi bilangan bulat

```
SELECT CONVERT(integer, pricepaid)
FROM sales WHERE salesid=100;
```

Contoh ini mengkonversi integer menjadi string karakter.

```
SELECT CONVERT(char(4), 2008);
```

Dalam contoh ini, tanggal dan waktu saat ini dikonversi ke tipe data karakter variabel:

```
SELECT CONVERT(VARCHAR(30), GETDATE());

getdate
---------
2023-02-02 04:31:16
```

Contoh ini mengubah kolom saletime menjadi hanya waktu, menghapus tanggal dari setiap baris.

```
SELECT CONVERT(time, saletime), salesid
```

```
FROM sales order by salesid limit 10;
```

Contoh berikut mengkonversi data karakter variabel menjadi objek datetime.

```
SELECT CONVERT(datetime, '2008-02-18 02:36:48') as mysaletime;
```

TO_CHAR

TO_CHAR mengkonversi timestamp atau ekspresi numerik ke format data karakter-string.

Sintaks

```
TO_CHAR (timestamp_expression | numeric_expression , 'format')
```

Argumen

timestamp_expression

Ekspresi yang menghasilkan nilai tipe TIMESTAMP atau TIMESTAMPTZ atau nilai yang secara implisit dapat dipaksa ke stempel waktu.

numeric_expression

Ekspresi yang menghasilkan nilai tipe data numerik atau nilai yang secara implisit dapat dipaksa ke tipe numerik. Untuk informasi selengkapnya, lihat Jenis numerik. TO_CHAR menyisipkan spasi di sebelah kiri string angka.

> **Note**
>
> TO_CHAR tidak mendukung nilai DESIMAL 128-bit.

format

Format untuk nilai baru. Untuk format yang valid, lihat String format datetime dan String format numerik.

Jenis pengembalian

VARCHAR

Contoh-contoh

Contoh berikut mengkonversi stempel waktu ke nilai dengan tanggal dan waktu dalam format dengan nama bulan empuk untuk sembilan karakter, nama hari dalam seminggu, dan nomor hari bulan.

```
select to_char(timestamp '2009-12-31 23:15:59', 'MONTH-DY-DD-YYYY HH12:MIPM');
to_char
-------------------------
DECEMBER -THU-31-2009 11:15PM
```

Contoh berikut mengonversi stempel waktu menjadi nilai dengan nomor hari dalam setahun.

```
select to_char(timestamp '2009-12-31 23:15:59', 'DDD');

to_char
-------------------------
365
```

Contoh berikut mengonversi stempel waktu ke nomor hari ISO dalam seminggu.

```
select to_char(timestamp '2022-05-16 23:15:59', 'ID');

to_char
-------------------------
1
```

Contoh berikut mengekstrak nama bulan dari tanggal.

```
select to_char(date '2009-12-31', 'MONTH');

to_char
-------------------------
DECEMBER
```

Contoh berikut mengkonversi setiap nilai STARTTIME dalam tabel EVENT ke string yang terdiri dari jam, menit, dan detik.

```
select to_char(starttime, 'HH12:MI:SS')
from event where eventid between 1 and 5
order by eventid;
```

```
to_char
----------
02:30:00
08:00:00
02:30:00
02:30:00
07:00:00
(5 rows)
```

Contoh berikut mengkonversi seluruh nilai timestamp ke dalam format yang berbeda.

```
select starttime, to_char(starttime, 'MON-DD-YYYY HH12:MIPM')
from event where eventid=1;

      starttime      |       to_char
---------------------+---------------------
 2008-01-25 14:30:00 | JAN-25-2008 02:30PM
(1 row)
```

Contoh berikut mengkonversi stempel waktu literal ke string karakter.

```
select to_char(timestamp '2009-12-31 23:15:59','HH24:MI:SS');
to_char
----------
23:15:59
(1 row)
```

Contoh berikut mengkonversi angka ke string karakter dengan tanda negatif di akhir.

```
select to_char(-125.8, '999D99S');
to_char
---------
125.80-
(1 row)
```

Contoh berikut mengkonversi angka ke string karakter dengan simbol mata uang.

```
select to_char(-125.88, '$S999D99');
to_char
---------
```

```
$-125.88
(1 row)
```

Contoh berikut mengkonversi angka ke string karakter menggunakan kurung sudut untuk angka negatif.

```
select to_char(-125.88, '$999D99PR');
to_char
---------
$<125.88>
(1 row)
```

Contoh berikut mengkonversi angka ke string angka Romawi.

```
select to_char(125, 'RN');
to_char
---------
CXXV
(1 row)
```

Contoh berikut menampilkan hari dalam seminggu.

```
SELECT to_char(current_timestamp, 'FMDay, FMDD HH12:MI:SS');
               to_char
-----------------------
Wednesday, 31 09:34:26
```

Contoh berikut menampilkan akhiran nomor urut untuk angka.

```
SELECT to_char(482, '999th');
               to_char
-----------------------
 482nd
```

Contoh berikut mengurangi komisi dari harga yang dibayarkan dalam tabel penjualan. Perbedaannya kemudian dibulatkan dan diubah menjadi angka romawi, ditunjukkan pada `to_char` kolom:

```
select salesid, pricepaid, commission, (pricepaid - commission)
as difference, to_char(pricepaid - commission, 'rn') from sales
group by sales.pricepaid, sales.commission, salesid
```

```
order by salesid limit 10;

 salesid | pricepaid | commission | difference |     to_char
---------+-----------+------------+------------+-----------------
       1 |    728.00 |     109.20 |     618.80 |           dcxix
       2 |     76.00 |      11.40 |      64.60 |             lxv
       3 |    350.00 |      52.50 |     297.50 |        ccxcviii
       4 |    175.00 |      26.25 |     148.75 |           cxlix
       5 |    154.00 |      23.10 |     130.90 |           cxxxi
       6 |    394.00 |      59.10 |     334.90 |         cccxxxv
       7 |    788.00 |     118.20 |     669.80 |           dclxx
       8 |    197.00 |      29.55 |     167.45 |          clxvii
       9 |    591.00 |      88.65 |     502.35 |              dii
      10 |     65.00 |       9.75 |      55.25 |               lv
(10 rows)
```

Contoh berikut menambahkan simbol mata uang ke nilai selisih yang ditunjukkan pada `to_char` kolom:

```
select salesid, pricepaid, commission, (pricepaid - commission)
as difference, to_char(pricepaid - commission, 'l99999D99') from sales
group by sales.pricepaid, sales.commission, salesid
order by salesid limit 10;

salesid | pricepaid | commission | difference |  to_char
--------+-----------+------------+------------+------------
      1 |    728.00 |     109.20 |     618.80 | $   618.80
      2 |     76.00 |      11.40 |      64.60 | $    64.60
      3 |    350.00 |      52.50 |     297.50 | $   297.50
      4 |    175.00 |      26.25 |     148.75 | $   148.75
      5 |    154.00 |      23.10 |     130.90 | $   130.90
      6 |    394.00 |      59.10 |     334.90 | $   334.90
      7 |    788.00 |     118.20 |     669.80 | $   669.80
      8 |    197.00 |      29.55 |     167.45 | $   167.45
      9 |    591.00 |      88.65 |     502.35 | $   502.35
     10 |     65.00 |       9.75 |      55.25 | $    55.25
(10 rows)
```

Contoh berikut mencantumkan abad di mana setiap penjualan dilakukan.

```
select salesid, saletime, to_char(saletime, 'cc') from sales
order by salesid limit 10;
```

```
 salesid |      saletime       | to_char
---------+---------------------+---------
       1 | 2008-02-18 02:36:48 | 21
       2 | 2008-06-06 05:00:16 | 21
       3 | 2008-06-06 08:26:17 | 21
       4 | 2008-06-09 08:38:52 | 21
       5 | 2008-08-31 09:17:02 | 21
       6 | 2008-07-16 11:59:24 | 21
       7 | 2008-06-26 12:56:06 | 21
       8 | 2008-07-10 02:12:36 | 21
       9 | 2008-07-22 02:23:17 | 21
      10 | 2008-08-06 02:51:55 | 21
(10 rows)
```

Contoh berikut mengkonversi setiap nilai STARTTIME dalam tabel EVENT ke string yang terdiri dari jam, menit, detik, dan zona waktu.

```
select to_char(starttime, 'HH12:MI:SS TZ')
from event where eventid between 1 and 5
order by eventid;

to_char
----------
02:30:00 UTC
08:00:00 UTC
02:30:00 UTC
02:30:00 UTC
07:00:00 UTC
(5 rows)

(10 rows)
```

Contoh berikut menunjukkan pemformatan untuk detik, milidetik, dan mikrodetik.

```
select sysdate,
to_char(sysdate, 'HH24:MI:SS') as seconds,
to_char(sysdate, 'HH24:MI:SS.MS') as milliseconds,
to_char(sysdate, 'HH24:MI:SS:US') as microseconds;

timestamp           | seconds  | milliseconds | microseconds
--------------------+----------+--------------+----------------
2015-04-10 18:45:09 | 18:45:09 | 18:45:09.325 | 18:45:09:325143
```

Fungsi TO_DATE

TO_DATE mengonversi tanggal yang diwakili oleh string karakter ke tipe data DATE.

Sintaks

```
TO_DATE(string, format)
```

```
TO_DATE(string, format, is_strict)
```

Argumen

tali

Sebuah string yang akan dikonversi.

format

Sebuah string literal yang mendefinisikan format string input, dalam hal bagian tanggalnya. Untuk daftar format hari, bulan, dan tahun yang valid, lihatString format datetime.

is_strict

Nilai Boolean opsional yang menentukan apakah kesalahan dikembalikan jika nilai tanggal masukan berada di luar jangkauan. Ketika is_strict disetel keTRUE, kesalahan dikembalikan jika ada nilai di luar jangkauan. Ketika is_strict disetel keFALSE, yang merupakan default, maka nilai overflow diterima.

Jenis pengembalian

TO_DATE mengembalikan DATE, tergantung pada nilai format.

Jika konversi ke format gagal, maka kesalahan dikembalikan.

Contoh-contoh

Pernyataan SQL berikut mengubah tanggal `02 Oct 2001` menjadi tipe data tanggal.

```
select to_date('02 Oct 2001', 'DD Mon YYYY');

to_date
```

```
------------
2001-10-02
(1 row)
```

Pernyataan SQL berikut mengkonversi string 20010631 ke tanggal.

```
select to_date('20010631', 'YYYYMMDD', FALSE);
```

Hasilnya adalah 1 Juli 2001, karena hanya ada 30 hari di bulan Juni.

```
to_date
------------
2001-07-01
```

Pernyataan SQL berikut mengkonversi string 20010631 ke tanggal:

```
to_date('20010631', 'YYYYMMDD', TRUE);
```

Hasilnya adalah kesalahan karena hanya ada 30 hari di bulan Juni.

```
ERROR:  date/time field date value out of range: 2001-6-31
```

TO_NUMBER

TO_NUMBER mengkonversi string ke nilai numerik (desimal).

Sintaks

```
to_number(string, format)
```

Argumen

tali

String yang akan dikonversi. Formatnya harus berupa nilai literal.

format

Argumen kedua adalah string format yang menunjukkan bagaimana string karakter harus diurai untuk membuat nilai numerik. Misalnya, format '99D999' menentukan bahwa string

yang akan dikonversi terdiri dari lima digit dengan titik desimal di posisi ketiga. Misalnya, `to_number('12.345','99D999')` kembali `12.345` sebagai nilai numerik. Untuk daftar format yang valid, lihat String format numerik.

Jenis pengembalian

TO_NUMBER mengembalikan nomor DECIMAL.

Jika konversi ke format gagal, maka kesalahan dikembalikan.

Contoh-contoh

Contoh berikut mengkonversi string `12,454.8-` ke nomor:

```
select to_number('12,454.8-', '99G999D9S');

to_number
-----------
-12454.8
```

Contoh berikut mengkonversi string `$ 12,454.88` ke nomor:

```
select to_number('$ 12,454.88', 'L 99G999D99');

to_number
-----------
12454.88
```

Contoh berikut mengkonversi string `$ 2,012,454.88` ke nomor:

```
select to_number('$ 2,012,454.88', 'L 9,999,999.99');

to_number
-----------
2012454.88
```

String format datetime

String format datetime berikut berlaku untuk fungsi seperti TO_CHAR. String ini dapat berisi pemisah datetime (seperti '', '`-`', atau '`/:`') dan “dateparts” dan “timeparts” berikut.

Untuk contoh pemformatan tanggal sebagai string, lihat. TO_CHAR

Datepart atau timepart	Arti
BC atau SM, AD atau A.D., b.c. atau bc, ad atau a.d.	Indikator era huruf besar dan kecil
CC	Nomor abad dua digit
YYY, YYY, Y	Nomor 4 digit, 3 digit, 2 digit, 1 digit tahun
Y, YYY	Nomor 4 digit tahun dengan koma
IYYY, IYY, IY, I	4 digit, 3 digit, 2 digit, 1 digit nomor tahun Organisasi Internasional untuk Standardisasi (ISO)
Q	Angka kuartal (1 hingga 4)
BULAN, Bulan, bulan	Nama bulan (huruf besar, huruf campuran, huruf kecil, empuk kosong hingga 9 karakter)
SENIN, Sen, mon	Nama bulan disingkat (huruf besar, huruf campuran, huruf kecil, empuk kosong hingga 3 karakter)
MM	Nomor bulan (01-12)
RM, rm	Nomor bulan dalam angka romawi (I — XII, dengan I menjadi Januari, huruf besar atau kecil)
W	Minggu bulan (1-5; minggu pertama dimulai pada hari pertama bulan itu.)
WW	Jumlah minggu tahun (1—53; minggu pertama dimulai pada hari pertama tahun itu.)
IW	Nomor minggu ISO tahun (Kamis pertama tahun baru adalah di minggu 1.)

Datepart atau timepart	Arti
HARI, Hari, hari	Nama hari (huruf besar, huruf campuran, huruf kecil, empuk kosong hingga 9 karakter)
DY, Dy, dy	Nama hari yang disingkat (huruf besar, huruf campuran, huruf kecil, empuk kosong hingga 3 karakter)
DDD	Hari dalam setahun (001—366)
IDDD	Hari ISO 8601 tahun penomoran minggu (001-371; hari 1 tahun ini adalah Senin minggu ISO pertama)
DD	Hari dalam sebulan sebagai angka (01-31)
D	Hari dalam seminggu (1—7; Minggu adalah 1) **Note** Datepart D berperilaku berbeda dari datepart day of week (DOW) yang digunakan untuk fungsi datetime DATE_PART dan EXTRACT. DOW didasarkan pada bilangan bulat 0-6, di mana hari Minggu adalah 0. Untuk informasi selengkapnya, lihat Bagian tanggal untuk fungsi tanggal atau stempel waktu.
ID	ISO 8601 hari dalam seminggu, Senin (1) sampai Minggu (7)
J	Hari Julian (hari sejak 1 Januari 4712 SM)
HH24	Jam (jam 24 jam, 00—23)

Datepart atau timepart	Arti
HH atau HH12	Jam (jam 12 jam, 01-12)
MI	Menit (00—59)
SS	Detik (00—59)
MS	Milidetik (.000)
US	Mikrodetik (.000000)
AM atau PM, A.M. atau PM, a.m. atau p.m., am atau pm	Indikator meridian huruf besar dan kecil (untuk jam 12 jam)
TZ, tz	Singkatan zona waktu huruf besar dan kecil; hanya berlaku untuk TIMESTAMPTZ
DARI	Offset dari UTC; hanya berlaku untuk TIMESTAMPTZ

> **Note**
>
> Anda harus mengelilingi pemisah datetime (seperti '-', '/' atau ':') dengan tanda kutip tunggal, tetapi Anda harus mengelilingi “bagian tanggal” dan “bagian waktu” yang tercantum dalam tabel sebelumnya dengan tanda kutip ganda.

String format numerik

String format numerik berikut berlaku untuk fungsi seperti TO_NUMBER dan TO_CHAR.

- Untuk contoh memformat string sebagai angka, lihat. TO_NUMBER
- Untuk contoh memformat angka sebagai string, lihat. TO_CHAR

format	Deskripsi
9	Nilai numerik dengan jumlah digit yang ditentukan.
0	Nilai numerik dengan angka nol di depan.
. (periode), D	Titik desimal.
, (koma)	Ribuan pemisah.
CC	Kode abad. Misalnya, abad ke-21 dimulai pada 2001-01-01 (hanya didukung untuk TO_CHAR).
FM	Mode isi. Menekan padding kosong dan nol.
PR	Nilai negatif dalam kurung sudut.
D	Tanda tangani berlabuh ke nomor.
L	Simbol mata uang di posisi yang ditentukan.
G	Pemisah kelompok.
MI	Tanda minus di posisi yang ditentukan untuk angka yang kurang dari 0.
PL	Ditambah tanda di posisi yang ditentukan untuk angka yang lebih besar dari 0.
SG	Tanda plus atau minus di posisi yang ditentuka n.
RN	Angka romawi antara 1 dan 3999 (hanya didukung untuk TO_CHAR).
TH atau th	Akhiran nomor urut. Tidak mengubah angka pecahan atau nilai yang kurang dari nol.

Teradata-gaya pemformatan karakter untuk data numerik

Topik ini menunjukkan kepada Anda bagaimana fungsi TEXT_TO_INT_ALT dan TEXT_TO_NUMERIC_ALT menafsirkan karakter dalam string ekspresi input. Dalam tabel berikut, Anda juga dapat menemukan daftar karakter yang dapat Anda tentukan dalam frasa format. Selain itu, Anda dapat menemukan deskripsi perbedaan antara pemformatan gaya Teradata dan AWS Clean Rooms untuk opsi format.

format	Deskripsi
G	Tidak didukung sebagai pemisah grup dalam string ekspresi input. Anda tidak dapat menentukan karakter ini dalam frasa format.
D	Simbol Radix. Anda dapat menentukan karakter ini dalam frasa format. Karakter ini setara dengan. (periode). Simbol radix tidak dapat muncul dalam frasa format yang berisi salah satu karakter berikut: • . (periode) • S (huruf besar 's') • V (huruf besar 'v')
/ , : %	Karakter penyisipan/(garis miring ke depan), koma (,),: (titik dua), dan% (tanda persen). Anda tidak dapat memasukkan karakter ini dalam frasa format. AWS Clean Rooms mengabaikan karakter ini dalam string ekspresi input.
.	Periode sebagai karakter radix, yaitu titik desimal. Karakter ini tidak dapat muncul dalam frasa format yang berisi salah satu karakter berikut:

format	Deskripsi
	• D (huruf besar 'd') • S (huruf besar 's') • V (huruf besar 'v')
B	Anda tidak dapat menyertakan karakter spasi kosong (B) dalam frasa format. Dalam string ekspresi input, spasi depan dan belakang diabaikan dan spasi antar digit tidak diperbole hkan.
+ -	Anda tidak dapat menyertakan tanda plus (+) atau tanda minus (-) dalam frasa format. Namun, tanda plus (+) dan tanda minus (-) diuraikan secara implisit sebagai bagian dari nilai numerik jika muncul dalam string ekspresi input.
V	Indikator posisi titik desimal. Karakter ini tidak dapat muncul dalam frasa format yang berisi salah satu karakter berikut: • D (huruf besar 'd') • . (periode)
Z	Digit desimal yang ditekan nol. AWS Clean Rooms memangkas angka nol terkemuka. Karakter Z tidak dapat mengikuti karakter 9. Karakter Z harus berada di sebelah kiri karakter radix jika bagian fraksi berisi karakter 9.
9	Digit desimal.

format	Deskripsi
CHAR (n)	Untuk format ini, Anda dapat menentukan yang berikut: • CHAR terdiri dari Z atau 9 karakter. AWS Clean Rooms tidak mendukung + (plus) atau - (minus) dalam nilai CHAR. • n adalah konstanta integer, I, atau F. Untuk I, ini adalah jumlah karakter yang diperluka n untuk menampilkan bagian integer dari data numerik atau integer. Untuk F, ini adalah jumlah karakter yang diperlukan untuk menampilkan bagian pecahan data numerik.
-	Karakter tanda hubung (-). Anda tidak dapat memasukkan karakter ini dalam frasa format. AWS Clean Rooms mengabaikan karakter ini dalam string ekspresi input.

format	Deskripsi
D	Ditandatangani desimal dikategorikan. Karakter S harus mengikuti digit desimal terakhir dalam frasa format. Karakter terakhir dari string ekspresi input dan konversi numerik yang sesuai tercantum dalamKarakter pemformatan data untuk desimal zonasi yang ditandatangani, pemformatan data numerik gaya Teradata . Karakter S tidak dapat muncul dalam frasa format yang berisi salah satu karakter berikut: • + (tanda plus) • . (periode) • D (huruf besar 'd') • Z (huruf besar 'z') • F (huruf besar 'f') • E (huruf besar 'e')
E	Notasi eksponensial. String ekspresi input dapat mencakup karakter eksponen. Anda tidak dapat menentukan E sebagai karakter eksponen dalam frasa format.
FN9	Tidak didukung di AWS Clean Rooms.
FNE	Tidak didukung di AWS Clean Rooms.

format	Deskripsi
$, USD, Dolar AS	Tanda dolar ($), simbol mata uang ISO (USD), dan nama mata uang Dolar AS. Simbol mata uang ISO USD dan nama mata uang Dolar AS peka huruf besar/kecil. AWS Clean Rooms hanya mendukung mata uang USD. String ekspresi input dapat mencakup spasi antara simbol mata uang USD dan nilai numerik, misalnya '$ 123E2' atau '123E2 $'.
L	Simbol mata uang. Karakter simbol mata uang ini hanya dapat muncul sekali dalam frasa format. Anda tidak dapat menentukan karakter simbol mata uang berulang.
C	Simbol mata uang ISO. Karakter simbol mata uang ini hanya dapat muncul sekali dalam frasa format. Anda tidak dapat menentukan karakter simbol mata uang berulang.
T	Nama mata uang lengkap. Karakter simbol mata uang ini hanya dapat muncul sekali dalam frasa format. Anda tidak dapat menentukan karakter simbol mata uang berulang.
O	Simbol mata uang ganda. Anda tidak dapat menentukan karakter ini dalam frasa format.
U	Simbol mata uang ISO ganda. Anda tidak dapat menentukan karakter ini dalam frasa format.
A	Nama mata uang ganda lengkap. Anda tidak dapat menentukan karakter ini dalam frasa format.

Karakter pemformatan data untuk desimal zonasi yang ditandatangani, pemformatan data numerik gaya Teradata

Anda dapat menggunakan karakter berikut dalam frase format fungsi TEXT_TO_INT_ALT dan TEXT_TO_NUMERIC_ALT untuk nilai desimal yang dikategorikan yang ditandatangani.

Karakter terakhir dari string input	Konversi numerik
{atau 0	n... 0
A atau 1	n... 1
B atau 2	n... 2
C atau 3	n... 3
D atau 4	n... 4
E atau 5	n... 5
F atau 6	n... 6
G atau 7	n... 7
H atau 8	n... 8
Saya atau 9	n... 9
}	- n... 0
J	- n... 1
K	- n... 2
L	- n... 3
M	- n... 4
T	- n... 5
O	- n... 6

Karakter terakhir dari string input	Konversi numerik
P	- n... 7
Q	- n... 8
R	- n... 9

Fungsi tanggal dan waktu

AWS Clean Rooms mendukung fungsi tanggal dan waktu berikut:

Topik

Ringkasan fungsi tanggal dan waktu

Tabel berikut memberikan ringkasan fungsi tanggal dan waktu yang digunakan dalam AWS Clean Rooms.

Fungsi	Sintaks	Pengembalian
+ Operator (Penggabungan) Menggabungkan tanggal ke waktu di kedua sisi simbol + dan mengembalikan TIMESTAMP atau TIMESTAMPTZ.	tanggal+waktu	`TIMESTAMP` atau `TIMESTAMPZ`
ADD_MONTHS Menambahkan jumlah bulan yang ditentukan ke tanggal atau stempel waktu.	ADD_MONTHS ({tanggal | cap waktu}, bilangan bulat)	`TIMESTAMP`
Fungsi CURRENT_DATE Mengembalikan tanggal di zona waktu sesi saat ini (UTC secara default) untuk memulai transaksi saat ini.	CURRENT_DATE	`DATE`
DATEADD Menambah tanggal atau waktu dengan interval tertentu.	DATEADD (datepart, interval, {tanggal | waktu | jadwal | cap waktu})	`TIMESTAMP` atau `TIME` atau `TIMETZ`
DATEDIFF Mengembalikan selisih antara dua tanggal atau waktu untuk bagian tanggal tertentu, seperti hari atau bulan.	DATEDIFF (datepart, {tanggal | waktu | jadwal waktu | timestamp} , {tanggal | waktu | jadwal waktu})	`BIGINT`
DATE_PART Mengekstrak nilai bagian tanggal dari tanggal atau waktu.	DATE_PART (bagian tanggal, {tanggal | cap waktu})	`DOUBLE`

Fungsi	Sintaks	Pengembalian
DATE_TRUNC Mempotong stempel waktu berdasarkan bagian tanggal.	DATE_TRUNC ('datepart', stempel waktu)	`TIMESTAMP`
EKSTRAK Mengekstrak bagian tanggal atau waktu dari stempel waktu, timestamptz, waktu, atau jadwal.	EKSTRAK (datepart DARI sumber)	`INTEGER or DOUBLE`
Fungsi GETDATE Mengembalikan tanggal dan waktu saat ini di zona waktu sesi saat ini (UTC secara default). Tanda kurung diperlukan.	GETDATE ()	`TIMESTAMP`
SYSDATE Mengembalikan tanggal dan waktu di UTC untuk memulai transaksi saat ini.	SYSDATE	`TIMESTAMP`
WAKTUHARI Mengembalikan hari kerja, tanggal, dan waktu saat ini di zona waktu sesi saat ini (UTC secara default) sebagai nilai string.	WAKTUHARI ()	`VARCHAR`
TO_TIMESTAMP Mengembalikan timestamp dengan zona waktu untuk format timestamp dan zona waktu yang ditentukan.	TO_TIMESTAMP ('cap waktu', 'format')	`TIMESTAMP TZ`

> **Note**
>
> Detik kabisat tidak dipertimbangkan dalam perhitungan waktu berlalu.

Fungsi tanggal dan waktu dalam transaksi

Ketika Anda menjalankan fungsi berikut dalam blok transaksi (BEGIN... END), fungsi mengembalikan tanggal mulai atau waktu transaksi saat ini, bukan awal dari pernyataan saat ini.

- SYSDATE
- TIMESTAMP
- CURRENT_DATE

Fungsi-fungsi berikut selalu mengembalikan tanggal mulai atau waktu pernyataan saat ini, bahkan ketika mereka berada dalam blok transaksi.

- GETDATE
- WAKTUHARI

+ Operator (Penggabungan)

Menggabungkan literal numerik, literal string, dan/atau literal datetime dan interval. Mereka berada di kedua sisi simbol + dan mengembalikan jenis yang berbeda berdasarkan input di kedua sisi simbol +.

Sintaks

```
numeric + string
```

```
date + time
```

```
date + timetz
```

Urutan argumen dapat dibalik.

Argumen

literal numerik

Literal atau konstanta yang mewakili angka dapat berupa integer atau floating-point.

string literal

String, string karakter, atau konstanta karakter

tanggal

DATEKolom atau ekspresi yang secara implisit mengkonversi ke. DATE

waktu

TIMEKolom atau ekspresi yang secara implisit mengkonversi ke. TIME

jadwal

TIMETZKolom atau ekspresi yang secara implisit mengkonversi ke. TIMETZ

Contoh

Contoh tabel berikut TIME_TEST memiliki kolom TIME_VAL (tipeTIME) dengan tiga nilai dimasukkan.

```
select date '2000-01-02' + time_val as ts from time_test;
```

Fungsi ADD_MONTHS

ADD_MONTHS menambahkan jumlah bulan yang ditentukan ke nilai atau ekspresi tanggal atau stempel waktu. DATEADDFungsi ini menyediakan fungsionalitas serupa.

Sintaks

```
ADD_MONTHS( {date | timestamp}, integer)
```

Argumen

tanggal | stempel waktu

Kolom tanggal atau stempel waktu atau ekspresi yang secara implisit mengkonversi ke tanggal atau stempel waktu. Jika tanggal adalah hari terakhir bulan itu, atau jika bulan yang dihasilkan

lebih pendek, fungsi mengembalikan hari terakhir bulan dalam hasilnya. Untuk tanggal lain, hasilnya berisi nomor hari yang sama dengan ekspresi tanggal.

bilangan bulat

Sebuah bilangan bulat positif atau negatif. Gunakan angka negatif untuk mengurangi bulan dari tanggal.

Jenis pengembalian

TIMESTAMP

Contoh

Query berikut menggunakan fungsi ADD_MONTHS di dalam fungsi TRUNC. Fungsi TRUNC menghapus waktu hari dari hasil ADD_MONTHS. Fungsi ADD_MONTHS menambahkan 12 bulan ke setiap nilai dari kolom CALDATE.

```
select distinct trunc(add_months(caldate, 12)) as calplus12,
trunc(caldate) as cal
from date
order by 1 asc;

 calplus12  |    cal
------------+------------
 2009-01-01 | 2008-01-01
 2009-01-02 | 2008-01-02
 2009-01-03 | 2008-01-03
...
(365 rows)
```

Contoh berikut menunjukkan perilaku ketika fungsi ADD_MONTHS beroperasi pada tanggal dengan bulan yang memiliki jumlah hari yang berbeda.

```
select add_months('2008-03-31',1);

add_months
---------------------
2008-04-30 00:00:00
(1 row)
```

```
select add_months('2008-04-30',1);

add_months
---------------------
2008-05-31 00:00:00
(1 row)
```

Fungsi CONVERT_TIMEZONE

CONVERT_TIMEZONE mengonversi stempel waktu dari satu zona waktu ke zona waktu lainnya. Fungsi ini secara otomatis menyesuaikan waktu musim panas.

Sintaks

```
CONVERT_TIMEZONE ( ['source_timezone',] 'target_timezone', 'timestamp')
```

Argumen

source_timezone

(Opsional) Zona waktu stempel waktu saat ini. Defaultnya adalah UTC.

target_zona waktu

Zona waktu untuk stempel waktu baru.

stempel waktu

Kolom timestamp atau ekspresi yang secara implisit mengkonversi ke stempel waktu.

Jenis pengembalian

TIMESTAMP

Contoh-contoh

Contoh berikut mengkonversi nilai timestamp dari zona waktu UTC default untuk PST.

```
select convert_timezone('PST', '2008-08-21 07:23:54');
```

```
 convert_timezone
-----------------------
2008-08-20 23:23:54
```

Contoh berikut mengkonversi nilai timestamp dalam kolom LISTTIME dari zona waktu UTC default ke PST. Meskipun stempel waktu berada dalam periode waktu siang hari, itu diubah menjadi waktu standar karena zona waktu target ditentukan sebagai singkatan (PST).

```
select listtime, convert_timezone('PST', listtime) from listing
where listid = 16;

     listtime       |   convert_timezone
--------------------+-------------------
2008-08-24 09:36:12     2008-08-24 01:36:12
```

Contoh berikut mengonversi timestamp kolom LISTTIME dari zona waktu UTC default ke zona waktu AS/Pasifik. Zona waktu target menggunakan nama zona waktu, dan stempel waktu berada dalam periode waktu siang hari, sehingga fungsi mengembalikan waktu siang hari.

```
select listtime, convert_timezone('US/Pacific', listtime) from listing
where listid = 16;

     listtime       |   convert_timezone
--------------------+--------------------
2008-08-24 09:36:12 | 2008-08-24 02:36:12
```

Contoh berikut mengkonversi string timestamp dari EST ke PST:

```
select convert_timezone('EST', 'PST', '20080305 12:25:29');

 convert_timezone
-------------------
2008-03-05 09:25:29
```

Contoh berikut mengubah stempel waktu ke Waktu Standar Timur AS karena zona waktu target menggunakan nama zona waktu (America/New_York) dan stempel waktu berada dalam periode waktu standar.

```
select convert_timezone('America/New_York', '2013-02-01 08:00:00');
```

```
 convert_timezone
---------------------
2013-02-01 03:00:00
(1 row)
```

Contoh berikut mengubah stempel waktu menjadi US Eastern Daylight Time karena zona waktu target menggunakan nama zona waktu (America/New_York) dan stempel waktu berada dalam periode waktu siang hari.

```
select convert_timezone('America/New_York', '2013-06-01 08:00:00');

 convert_timezone
---------------------
2013-06-01 04:00:00
(1 row)
```

Contoh berikut menunjukkan penggunaan offset.

```
SELECT CONVERT_TIMEZONE('GMT','NEWZONE +2','2014-05-17 12:00:00') as newzone_plus_2,
CONVERT_TIMEZONE('GMT','NEWZONE-2:15','2014-05-17 12:00:00') as newzone_minus_2_15,
CONVERT_TIMEZONE('GMT','America/Los_Angeles+2','2014-05-17 12:00:00') as la_plus_2,
CONVERT_TIMEZONE('GMT','GMT+2','2014-05-17 12:00:00') as gmt_plus_2;

   newzone_plus_2    | newzone_minus_2_15  |      la_plus_2      |     gmt_plus_2
---------------------+---------------------+---------------------+---------------------
2014-05-17 10:00:00 | 2014-05-17 14:15:00 | 2014-05-17 10:00:00 | 2014-05-17 10:00:00
(1 row)
```

Fungsi CURRENT_DATE

CURRENT_DATE mengembalikan tanggal di zona waktu sesi saat ini (UTC secara default) dalam format default: YYYY-MM-DD.

> **Note**
>
> CURRENT_DATE mengembalikan tanggal mulai untuk transaksi saat ini, bukan untuk awal pernyataan saat ini. Pertimbangkan skenario di mana Anda memulai transaksi yang berisi beberapa pernyataan pada 10/01/08 23:59, dan pernyataan yang berisi CURRENT_DATE berjalan pada 10/02/08 00:00. CURRENT_DATE kembali`10/01/08`, tidak. `10/02/08`

Sintaks

```
CURRENT_DATE
```

Jenis pengembalian

DATE

Contoh

Contoh berikut mengembalikan tanggal saat ini (di Wilayah AWS mana fungsi berjalan).

```
select current_date;

   date
------------
2008-10-01
```

Fungsi DATEADD

Menambah nilai DATE, TIME, TIMETZ, atau TIMESTAMP dengan interval tertentu.

Sintaks

```
DATEADD( datepart, interval, {date|time|timetz|timestamp} )
```

Argumen

datepart

Bagian tanggal (tahun, bulan, hari, atau jam, misalnya) tempat fungsi beroperasi. Untuk informasi selengkapnya, lihat Bagian tanggal untuk fungsi tanggal atau stempel waktu.

interval

Integer yang menentukan interval (jumlah hari, misalnya) untuk ditambahkan ke ekspresi target. Sebuah integer negatif mengurangi interval.

tanggal | waktu | timetz | stempel waktu

Kolom TANGGAL, WAKTU, TIMETZ, atau TIMESTAMP atau ekspresi yang secara implisit mengkonversi ke TANGGAL, WAKTU, TIMETZ, atau TIMESTAMP. Ekspresi DATE, TIME, TIMETZ, atau TIMESTAMP harus berisi bagian tanggal yang ditentukan.

Jenis pengembalian

TIMESTAMP atau TIME atau TIMETZ tergantung pada tipe data input.

Contoh dengan kolom DATE

Contoh berikut menambahkan 30 hari untuk setiap tanggal di bulan November yang ada di tabel DATE.

```
select dateadd(day,30,caldate) as novplus30
from date
where month='NOV'
order by dateid;

novplus30
---------------------
2008-12-01 00:00:00
2008-12-02 00:00:00
2008-12-03 00:00:00
...
(30 rows)
```

Contoh berikut menambahkan 18 bulan ke nilai tanggal literal.

```
select dateadd(month,18,'2008-02-28');

date_add
---------------------
2009-08-28 00:00:00
(1 row)
```

Nama kolom default untuk fungsi DATEADD adalah DATE_ADD. Stempel waktu default untuk nilai tanggal adalah. `00:00:00`

Contoh berikut menambahkan 30 menit ke nilai tanggal yang tidak menentukan stempel waktu.

```
select dateadd(m,30,'2008-02-28');

date_add
---------------------
2008-02-28 00:30:00
(1 row)
```

Anda dapat memberi nama bagian tanggal secara lengkap atau menyingkatnya. Dalam hal ini, m berarti menit, bukan bulan.

Contoh dengan kolom TIME

Berikut contoh tabel TIME_TEST memiliki kolom TIME_VAL (tipe TIME) dengan tiga nilai dimasukkan.

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

Contoh berikut menambahkan 5 menit untuk setiap TIME_VAL dalam tabel TIME_TEST.

```
select dateadd(minute,5,time_val) as minplus5 from time_test;

minplus5
---------------
20:05:00
00:05:00.5550
01:03:00
```

Contoh berikut menambahkan 8 jam ke nilai waktu literal.

```
select dateadd(hour, 8, time '13:24:55');

date_add
---------------
21:24:55
```

Contoh berikut menunjukkan kapan waktu berjalan lebih dari 24:00:00 atau di bawah 00:00:00.

```
select dateadd(hour, 12, time '13:24:55');

date_add
---------------
01:24:55
```

Contoh dengan kolom TIMETZ

Nilai output dalam contoh ini ada di UTC yang merupakan zona waktu default.

Contoh tabel berikut TIMETZ_TEST memiliki kolom TIMETZ_VAL (tipe TIMETZ) dengan tiga nilai dimasukkan.

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

Contoh berikut menambahkan 5 menit untuk setiap TIMETZ_VAL dalam tabel TIMETZ_TEST.

```
select dateadd(minute,5,timetz_val) as minplus5_tz from timetz_test;

minplus5_tz
---------------
04:05:00+00
00:05:00.5550+00
06:03:00+00
```

Contoh berikut menambahkan 2 jam ke nilai timetz literal.

```
select dateadd(hour, 2, timetz '13:24:55 PST');

date_add
---------------
23:24:55+00
```

Contoh dengan kolom TIMESTAMP

Nilai output dalam contoh ini ada di UTC yang merupakan zona waktu default.

Contoh tabel berikut TIMESTAMP_TEST memiliki kolom TIMESTAMP_VAL (tipe TIMESTAMP) dengan tiga nilai dimasukkan.

```
SELECT timestamp_val FROM timestamp_test;

timestamp_val
------------------
1988-05-15 10:23:31
2021-03-18 17:20:41
2023-06-02 18:11:12
```

Contoh berikut menambahkan 20 tahun hanya ke nilai TIMESTAMP_VAL di TIMESTAMP_TEST dari sebelum tahun 2000.

```
SELECT dateadd(year,20,timestamp_val)
FROM timestamp_test
WHERE timestamp_val < to_timestamp('2000-01-01 00:00:00', 'YYYY-MM-DD HH:MI:SS');

date_add
---------------
2008-05-15 10:23:31
```

Contoh berikut menambahkan 5 detik ke nilai stempel waktu literal yang ditulis tanpa indikator detik.

```
SELECT dateadd(second, 5, timestamp '2001-06-06');

date_add
---------------
2001-06-06 00:00:05
```

Catatan penggunaan

Fungsi DATEADD (bulan,...) dan ADD_MONTHS menangani tanggal yang jatuh pada akhir bulan secara berbeda:

- ADD_MONTHS: Jika tanggal yang Anda tambahkan adalah hari terakhir bulan itu, hasilnya selalu hari terakhir dari bulan hasil, terlepas dari panjang bulan. Misalnya, 30 April+1 bulan adalah 31 Mei.

```
select add_months('2008-04-30',1);

add_months
---------------------
2008-05-31 00:00:00
(1 row)
```

- DATEADD: Jika ada lebih sedikit hari pada tanggal yang Anda tambahkan daripada di bulan hasil, hasilnya adalah hari yang sesuai dari bulan hasil, bukan hari terakhir bulan itu. Misalnya, 30 April+1 bulan adalah 30 Mei.

```
select dateadd(month,1,'2008-04-30');

date_add
---------------------
2008-05-30 00:00:00
(1 row)
```

Fungsi DATEADD menangani tanggal tahun kabisat 02-29 secara berbeda saat menggunakan dateadd (month, 12,...) atau dateadd (year, 1,...).

```
select dateadd(month,12,'2016-02-29');

date_add
---------------------
2017-02-28 00:00:00

select dateadd(year, 1, '2016-02-29');

date_add
---------------------
2017-03-01 00:00:00
```

Fungsi DATEDIFF

DATEDIFF mengembalikan perbedaan antara bagian tanggal dari dua ekspresi tanggal atau waktu.

Sintaks

```
DATEDIFF ( datepart, {date|time|timetz|timestamp}, {date|time|timetz|timestamp} )
```

Argumen

datepart

Bagian spesifik dari nilai tanggal atau waktu (tahun, bulan, atau hari, jam, menit, detik, milidetik, atau mikrodetik) tempat fungsi beroperasi. Untuk informasi selengkapnya, lihat Bagian tanggal untuk fungsi tanggal atau stempel waktu.

Secara khusus, DATEDIFF menentukan jumlah batas bagian tanggal yang disilangkan antara dua ekspresi. Misalnya, Anda menghitung perbedaan tahun antara dua tanggal, `12-31-2008` dan`01-01-2009`. Dalam hal ini, fungsi mengembalikan 1 tahun meskipun fakta bahwa tanggal-tanggal ini hanya terpisah satu hari. Jika Anda menemukan perbedaan jam antara dua stempel waktu, `01-01-2009 8:30:00` dan`01-01-2009 10:00:00`, hasilnya adalah 2 jam. Jika Anda menemukan perbedaan jam antara dua stempel waktu, `8:30:00` dan`10:00:00`, hasilnya adalah 2 jam.

tanggal | waktu | timetz | stempel waktu

Kolom atau ekspresi TANGGAL, WAKTU, TIMETZ, atau TIMESTAMP yang secara implisit dikonversi ke TANGGAL, WAKTU, TIMETZ, atau TIMESTAMP. Ekspresi harus berisi tanggal atau bagian waktu yang ditentukan. Jika tanggal atau waktu kedua lebih lambat dari tanggal atau waktu pertama, hasilnya positif. Jika tanggal atau waktu kedua lebih awal dari tanggal atau waktu pertama, hasilnya negatif.

Jenis pengembalian

BIGINT

Contoh dengan kolom DATE

Contoh berikut menemukan perbedaan, dalam jumlah minggu, antara dua nilai tanggal literal.

```
select datediff(week,'2009-01-01','2009-12-31') as numweeks;

numweeks
----------
```

```
52
(1 row)
```

Contoh berikut menemukan perbedaan, dalam jam, antara dua nilai tanggal literal. Ketika Anda tidak memberikan nilai waktu untuk tanggal, defaultnya adalah 00:00:00.

```
select datediff(hour, '2023-01-01', '2023-01-03 05:04:03');

date_diff
----------
53
(1 row)
```

Contoh berikut menemukan perbedaan, dalam hari, antara dua nilai TIMESTAMETZ literal.

```
Select datediff(days, 'Jun 1,2008  09:59:59 EST', 'Jul 4,2008  09:59:59 EST')

date_diff
----------
33
```

Contoh berikut menemukan perbedaan, dalam hari, antara dua tanggal dalam baris tabel yang sama.

```
select * from date_table;

start_date |   end_date
-----------+-----------
2009-01-01 | 2009-03-23
2023-01-04 | 2024-05-04
(2 rows)

select datediff(day, start_date, end_date) as duration from date_table;

duration
---------
      81
     486
(2 rows)
```

Contoh berikut menemukan perbedaan, dalam jumlah kuartal, antara nilai literal di masa lalu dan tanggal hari ini. Contoh ini mengasumsikan bahwa tanggal saat ini adalah 5 Juni 2008. Anda dapat

memberi nama bagian tanggal secara lengkap atau menyingkatnya. Nama kolom default untuk fungsi DATEDIFF adalah DATE_DIFF.

```
select datediff(qtr, '1998-07-01', current_date);

date_diff
-----------
40
(1 row)
```

Contoh berikut bergabung dengan tabel PENJUALAN dan DAFTAR untuk menghitung berapa hari setelah mereka terdaftar, tiket apa pun dijual untuk daftar 1000 hingga 1005. Penantian terpanjang untuk penjualan daftar ini adalah 15 hari, dan yang terpendek kurang dari satu hari (0 hari).

```
select priceperticket,
datediff(day, listtime, saletime) as wait
from sales, listing where sales.listid = listing.listid
and sales.listid between 1000 and 1005
order by wait desc, priceperticket desc;

priceperticket | wait
---------------+------
 96.00         |   15
 123.00        |   11
 131.00        |    9
 123.00        |    6
 129.00        |    4
 96.00         |    4
 96.00         |    0
(7 rows)
```

Contoh ini menghitung jumlah rata-rata jam penjual menunggu semua penjualan tiket.

```
select avg(datediff(hours, listtime, saletime)) as avgwait
from sales, listing
where sales.listid = listing.listid;

avgwait
---------
465
(1 row)
```

Contoh dengan kolom TIME

Berikut contoh tabel TIME_TEST memiliki kolom TIME_VAL (tipe TIME) dengan tiga nilai dimasukkan.

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

Contoh berikut menemukan perbedaan jumlah jam antara kolom TIME_VAL dan literal waktu.

```
select datediff(hour, time_val, time '15:24:45') from time_test;

 date_diff
-----------
        -5
        15
        15
```

Contoh berikut menemukan perbedaan jumlah menit antara dua nilai waktu literal.

```
select datediff(minute, time '20:00:00', time '21:00:00') as nummins;

nummins
----------
60
```

Contoh dengan kolom TIMETZ

Contoh tabel berikut TIMETZ_TEST memiliki kolom TIMETZ_VAL (tipe TIMETZ) dengan tiga nilai dimasukkan.

```
select timetz_val from timetz_test;

timetz_val
------------------
```

```
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

Contoh berikut menemukan perbedaan jumlah jam, antara literal TIMETZ dan timetz_val.

```
select datediff(hours, timetz '20:00:00 PST', timetz_val) as numhours from timetz_test;

numhours
----------
0
-4
1
```

Contoh berikut menemukan perbedaan jumlah jam, antara dua nilai TIMETZ literal.

```
select datediff(hours, timetz '20:00:00 PST', timetz '00:58:00 EST') as numhours;

numhours
----------
1
```

Fungsi DATE_PART

DATE_PART mengekstrak nilai bagian tanggal dari ekspresi. DATE_PART adalah sinonim dari fungsi PGDATE_PART.

Sintaks

```
DATE_PART(datepart, {date|timestamp})
```

Argumen

datepart

Pengidentifikasi literal atau string dari bagian tertentu dari nilai tanggal (misalnya, tahun, bulan, atau hari) tempat fungsi beroperasi. Untuk informasi selengkapnya, lihat Bagian tanggal untuk fungsi tanggal atau stempel waktu.

{tanggal | stempel waktu}

Kolom tanggal, kolom stempel waktu, atau ekspresi yang secara implisit mengkonversi ke tanggal atau stempel waktu. Kolom atau ekspresi dalam tanggal atau stempel waktu harus berisi bagian tanggal yang ditentukan dalam datepart.

Jenis pengembalian

DOUBLE

Contoh-contoh

Nama kolom default untuk fungsi DATE_PART adalah. `pgdate_part`

Contoh berikut menemukan menit dari stempel waktu literal.

```
SELECT DATE_PART(minute, timestamp '20230104 04:05:06.789');

pgdate_part
-----------
          5
```

Contoh berikut menemukan nomor minggu dari literal stempel waktu. Perhitungan angka minggu mengikuti standar ISO 8601. Untuk informasi lebih lanjut, lihat ISO 8601 di Wikipedia.

```
SELECT DATE_PART(week, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
         18
```

Contoh berikut menemukan hari dalam sebulan dari stempel waktu literal.

```
SELECT DATE_PART(day, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
          2
```

Contoh berikut menemukan hari dalam seminggu dari stempel waktu literal. Perhitungan angka minggu mengikuti standar ISO 8601. Untuk informasi lebih lanjut, lihat ISO 8601 di Wikipedia.

```
SELECT DATE_PART(dayofweek, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
          1
```

Contoh berikut menemukan abad dari stempel waktu literal. Perhitungan abad mengikuti standar ISO 8601. Untuk informasi lebih lanjut, lihat ISO 8601 di Wikipedia.

```
SELECT DATE_PART(century, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
          21
```

Contoh berikut menemukan milenium dari literal stempel waktu. Perhitungan milenium mengikuti standar ISO 8601. Untuk informasi lebih lanjut, lihat ISO 8601 di Wikipedia.

```
SELECT DATE_PART(millennium, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
          3
```

Contoh berikut menemukan mikrodetik dari literal stempel waktu. Perhitungan mikrodetik mengikuti standar ISO 8601. Untuk informasi lebih lanjut, lihat ISO 8601 di Wikipedia.

```
SELECT DATE_PART(microsecond, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
     789000
```

Contoh berikut menemukan bulan dari tanggal literal.

```
SELECT DATE_PART(month, date '20220502');

pgdate_part
-----------
          5
```

Contoh berikut menerapkan fungsi DATE_PART ke kolom dalam tabel.

```
SELECT date_part(w, listtime) AS weeks, listtime
FROM listing
WHERE listid=10


weeks |      listtime
------+---------------------
 25   | 2008-06-17 09:44:54
(1 row)
```

Anda dapat memberi nama bagian tanggal secara lengkap atau menyingkatnya; dalam hal ini, w berarti minggu.

Bagian tanggal hari minggu mengembalikan bilangan bulat dari 0-6, dimulai dengan hari Minggu. Gunakan DATE_PART dengan dow (DAYOFWEEK) untuk melihat acara pada hari Sabtu.

```
SELECT date_part(dow, starttime) AS dow, starttime
FROM event
WHERE date_part(dow, starttime)=6
ORDER BY 2,1;

 dow |      starttime
-----+---------------------
   6 | 2008-01-05 14:00:00
   6 | 2008-01-05 14:00:00
   6 | 2008-01-05 14:00:00
   6 | 2008-01-05 14:00:00
...
(1147 rows)
```

Fungsi DATE_TRUNC

Fungsi DATE_TRUNC memotong ekspresi stempel waktu atau literal berdasarkan bagian tanggal yang Anda tentukan, seperti jam, hari, atau bulan.

Sintaks

```
DATE_TRUNC('datepart', timestamp)
```

Argumen

datepart

Bagian tanggal untuk memotong nilai stempel waktu. Stempel waktu input dipotong sesuai presisi datepart input. Misalnya, `month` memotong ke hari pertama bulan itu. Format yang valid adalah sebagai berikut:

- mikrodetik, mikrodetik
- milidetik, milidetik
- kedua, detik
- menit, menit
- jam, jam
- hari, hari
- minggu, minggu
- bulan, bulan
- seperempat, kuartal
- tahun, tahun
- dekade, dekade
- abad, berabad-abad
- milenium, milenium

Untuk informasi selengkapnya tentang singkatan dari beberapa format, lihat Bagian tanggal untuk fungsi tanggal atau stempel waktu

stempel waktu

Kolom timestamp atau ekspresi yang secara implisit mengkonversi ke stempel waktu.

Jenis pengembalian

TIMESTAMP

Contoh-contoh

Potong stempel waktu masukan ke yang kedua.

```
SELECT DATE_TRUNC('second', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:05:06
```

Potong stempel waktu input ke menit.

```
SELECT DATE_TRUNC('minute', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:05:00
```

Potong stempel waktu input ke jam.

```
SELECT DATE_TRUNC('hour', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:00:00
```

Potong stempel waktu input ke hari itu.

```
SELECT DATE_TRUNC('day', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 00:00:00
```

Potong stempel waktu input ke hari pertama dalam sebulan.

```
SELECT DATE_TRUNC('month', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-01 00:00:00
```

Potong stempel waktu input ke hari pertama seperempat.

```
SELECT DATE_TRUNC('quarter', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-01 00:00:00
```

Potong stempel waktu input ke hari pertama dalam setahun.

```
SELECT DATE_TRUNC('year', TIMESTAMP '20200430 04:05:06.789');
date_trunc
```

```
2020-01-01 00:00:00
```

Potong stempel waktu masukan ke hari pertama abad.

```
SELECT DATE_TRUNC('millennium', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2001-01-01 00:00:00
```

Potong stempel waktu input ke hari Senin dalam seminggu.

```
select date_trunc('week', TIMESTAMP '20220430 04:05:06.789');
date_trunc
2022-04-25 00:00:00
```

Dalam contoh berikut, fungsi DATE_TRUNC menggunakan bagian tanggal 'minggu' untuk mengembalikan tanggal untuk hari Senin setiap minggu.

```
select date_trunc('week', saletime), sum(pricepaid) from sales where
saletime like '2008-09%' group by date_trunc('week', saletime) order by 1;

date_trunc  |    sum
------------+-------------
2008-09-01  | 2474899
2008-09-08  | 2412354
2008-09-15  | 2364707
2008-09-22  | 2359351
2008-09-29  |  705249
```

Fungsi EKSTRAK

Fungsi EXTRACT mengembalikan bagian tanggal atau waktu dari nilai TIMESTAMP, TIMESTAMPTZ, TIME, atau TIMETZ. Contohnya termasuk hari, bulan, tahun, jam, menit, detik, milidetik, atau mikrodetik dari stempel waktu.

Sintaks

```
EXTRACT(datepart FROM source)
```

Argumen

datepart

Subbidang tanggal atau waktu untuk mengekstrak, seperti hari, bulan, tahun, jam, menit, detik, milidetik, atau mikrodetik. Untuk nilai yang mungkin, lihatBagian tanggal untuk fungsi tanggal atau stempel waktu.

sumber

Kolom atau ekspresi yang mengevaluasi tipe data TIMESTAMP, TIMESTAMPTZ, TIME, atau TIMETZ.

Jenis pengembalian

INTEGER jika nilai sumber mengevaluasi tipe data TIMESTAMP, TIME, atau TIMETZ.

PRESISI GANDA jika nilai sumber mengevaluasi tipe data TIMESTAMPTZ.

Contoh dengan TIMESTAMP

Contoh berikut menentukan angka minggu untuk penjualan di mana harga yang dibayarkan adalah $10.000 atau lebih.

```
select salesid, extract(week from saletime) as weeknum
from sales
where pricepaid > 9999
order by 2;

salesid | weeknum
--------+---------
 159073 |       6
 160318 |       8
 161723 |      26
```

Contoh berikut mengembalikan nilai menit dari nilai timestamp literal.

```
select extract(minute from timestamp '2009-09-09 12:08:43');

date_part
--
```

Contoh berikut mengembalikan nilai milidetik dari nilai timestamp literal.

```
select extract(ms from timestamp '2009-09-09 12:08:43.101');

date_part
-----------
101
```

Contoh dengan TIMESTAMPTZ

Contoh berikut mengembalikan nilai tahun dari nilai timestamptz literal.

```
select extract(year from timestamptz '1.12.1997 07:37:16.00 PST');

date_part
-----------
1997
```

Contoh dengan waktu

Berikut contoh tabel TIME_TEST memiliki kolom TIME_VAL (tipe TIME) dengan tiga nilai dimasukkan.

```
select time_val from time_test;

time_val
--------------------
20:00:00
00:00:00.5550
00:58:00
```

Contoh berikut mengekstrak menit dari setiap time_val.

```
select extract(minute from time_val) as minutes from time_test;

minutes
-----------
         0
         0
         58
```

Contoh berikut mengekstrak jam dari setiap time_val.

```
select extract(hour from time_val) as hours from time_test;

hours
-----------
         20
         0
         0
```

Contoh berikut mengekstrak milidetik dari nilai literal.

```
select extract(ms from time '18:25:33.123456');

 date_part
-----------
     123
```

Contoh dengan TIMETZ

Contoh tabel berikut TIMETZ_TEST memiliki kolom TIMETZ_VAL (tipe TIMETZ) dengan tiga nilai dimasukkan.

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

Contoh berikut mengekstrak jam dari setiap timetz_val.

```
select extract(hour from timetz_val) as hours from time_test;

hours
-----------
         4
         0
         5
```

Contoh berikut mengekstrak milidetik dari nilai literal. Literal tidak dikonversi ke UTC sebelum ekstraksi diproses.

```
select extract(ms from timetz '18:25:33.123456 EST');

 date_part
-----------
     123
```

Contoh berikut mengembalikan jam offset zona waktu dari UTC dari nilai timetz literal.

```
select extract(timezone_hour from timetz '1.12.1997 07:37:16.00 PDT');

date_part
-----------
-7
```

Fungsi GETDATE

GETDATEFungsi mengembalikan tanggal dan waktu saat ini di zona waktu sesi saat ini (UTC secara default).

Ini mengembalikan tanggal mulai atau waktu pernyataan saat ini, bahkan ketika itu berada dalam blok transaksi.

Sintaks

```
GETDATE()
```

Tanda kurung diperlukan.

Jenis pengembalian

TIMESTAMP

Contoh

Contoh berikut menggunakan GETDATE fungsi untuk mengembalikan stempel waktu penuh untuk tanggal saat ini.

```
select getdate();
```

fungsi SYSDATE

SYSDATE mengembalikan tanggal dan waktu saat ini di zona waktu sesi saat ini (UTC secara default).

> **Note**
>
> SYSDATE mengembalikan tanggal dan waktu mulai untuk transaksi saat ini, bukan untuk dimulainya pernyataan saat ini.

Sintaks

```
SYSDATE
```

Fungsi ini tidak memerlukan argumen.

Jenis pengembalian

TIMESTAMP

Contoh-contoh

Contoh berikut menggunakan fungsi SYSDATE untuk mengembalikan stempel waktu penuh untuk tanggal saat ini.

```
select sysdate;

timestamp
----------------------------
2008-12-04 16:10:43.976353
(1 row)
```

Contoh berikut menggunakan fungsi SYSDATE di dalam fungsi TRUNC untuk mengembalikan tanggal saat ini tanpa waktu.

```
select trunc(sysdate);

trunc
------------
```

```
2008-12-04
(1 row)
```

Kueri berikut mengembalikan informasi penjualan untuk tanggal yang berada di antara tanggal ketika kueri dikeluarkan dan tanggal berapa pun 120 hari sebelumnya.

```
select salesid, pricepaid, trunc(saletime) as saletime, trunc(sysdate) as now
from sales
where saletime between trunc(sysdate)-120 and trunc(sysdate)
order by saletime asc;

salesid | pricepaid |  saletime  |    now
---------+-----------+------------+------------
91535 |    670.00 | 2008-08-07 | 2008-12-05
91635 |    365.00 | 2008-08-07 | 2008-12-05
91901 |   1002.00 | 2008-08-07 | 2008-12-05
...
```

Fungsi TIMEOFDAY

TIMEOFDAY adalah alias khusus yang digunakan untuk mengembalikan hari kerja, tanggal, dan waktu sebagai nilai string. Ia mengembalikan string time of day untuk pernyataan saat ini, bahkan ketika itu berada dalam blok transaksi.

Sintaks

```
TIMEOFDAY()
```

Jenis pengembalian

VARCHAR

Contoh-contoh

Contoh berikut mengembalikan tanggal dan waktu saat ini dengan menggunakan fungsi TIMEOFDAY.

```
select timeofday();
timeofday
------------
Thu Sep 19 22:53:50.333525 2013 UTC
```

```
(1 row)
```

Fungsi TO_TIMESTAMP

TO_TIMESTAMP mengonversi string TIMESTAMP ke TIMESTAMPTZ.

Sintaks

```
to_timestamp (timestamp, format)
```

```
to_timestamp (timestamp, format, is_strict)
```

Argumen

stempel waktu

Sebuah string yang mewakili nilai timestamp dalam format yang ditentukan oleh format. Jika argumen ini dibiarkan kosong, nilai stempel waktu defaultnya. `0001-01-01 00:00:00`

format

Sebuah string literal yang mendefinisikan format nilai timestamp. Format yang menyertakan zona waktu (**`TZ`**,**`tz`**, atau**`OF`**) tidak didukung sebagai input. Untuk format stempel waktu yang valid, lihat. String format datetime

is_strict

Nilai Boolean opsional yang menentukan apakah kesalahan dikembalikan jika nilai timestamp masukan berada di luar jangkauan. Ketika is_strict disetel ke TRUE, kesalahan dikembalikan jika ada nilai di luar rentang. Ketika is_strict disetel ke FALSE, yang merupakan default, maka nilai overflow diterima.

Jenis pengembalian

TIMESTAMPTZ

Contoh-contoh

Contoh berikut menunjukkan penggunaan fungsi TO_TIMESTAMP untuk mengonversi string TIMESTAMP ke TIMESTAMPTZ.

```
select sysdate, to_timestamp(sysdate, 'YYYY-MM-DD HH24:MI:SS') as second;

timestamp                  | second
---------------------------   ---------------------
2021-04-05 19:27:53.281812 | 2021-04-05 19:27:53+00
```

Dimungkinkan untuk melewatkan TO_TIMESTAMP bagian dari tanggal. Bagian tanggal yang tersisa diatur ke nilai default. Waktu termasuk dalam output:

```
SELECT TO_TIMESTAMP('2017','YYYY');

to_timestamp
--------------------------
2017-01-01 00:00:00+00
```

Pernyataan SQL berikut mengonversi string '2011-12-18 24:38:15 'menjadi TIMESTAMPTZ. Hasilnya adalah TIMESTAMPTZ yang jatuh pada hari berikutnya karena jumlah jam lebih dari 24 jam:

```
SELECT TO_TIMESTAMP('2011-12-18 24:38:15', 'YYYY-MM-DD HH24:MI:SS');

to_timestamp
---------------------
2011-12-19 00:38:15+00
```

Pernyataan SQL berikut mengonversi string '2011-12-18 24:38:15 'menjadi TIMESTAMPTZ. Hasilnya adalah kesalahan karena nilai waktu dalam stempel waktu lebih dari 24 jam:

```
SELECT TO_TIMESTAMP('2011-12-18 24:38:15', 'YYYY-MM-DD HH24:MI:SS', TRUE);

ERROR:  date/time field time value out of range: 24:38:15.0
```

Bagian tanggal untuk fungsi tanggal atau stempel waktu

Tabel berikut mengidentifikasi nama bagian tanggal dan waktu bagian dan singkatan yang diterima sebagai argumen untuk fungsi berikut:

- DATEADD
- DATEDIFF
- DATE_PART

- EKSTRAK

Tanggal paruh waktu atau paruh waktu	Singkatan
milenium, milenium	mil, mil
abad, berabad-abad	c, sen, sen
dekade, dekade	Desember, decs
jangka waktu	epoch (didukung oleh) EKSTRAK
tahun, tahun	y, thn, thn
seperempat, kuartal	qtr, qtrs
bulan, bulan	mon, mons
minggu, minggu	w
hari dalam seminggu	dayofweek, dow, dw, hari kerja (didukung oleh dan) DATE_PART Fungsi EKSTRAK Mengembalikan integer dari 0-6, dimulai dengan hari Minggu. **Note** Bagian tanggal DOW berperilaku berbeda dari bagian tanggal hari minggu (D) yang digunakan untuk string format datetime. D didasarkan pada bilangan bulat 1-7, di mana hari Minggu adalah 1. Untuk informasi selengkap nya, lihat String format datetime.
hari dalam setahun	dayofyear, doy, dy, yearday (didukung oleh) EKSTRAK
hari, hari	d
jam, jam	h, jam, jam

Tanggal paruh waktu atau paruh waktu	Singkatan
menit, menit	m, min, menit
kedua, detik	s, detik, detik
milidetik, milidetik	ms, msec, msec, mdetik, mdetik, milidetik, milidetik, milidetik, milidetik
mikrodetik, mikrodetik	mikrosec, mikrodetik, mikrodetik, usecond, useconds, us, usec, usec
zona waktu, zona waktu_jam , zona waktu_menit	Didukung oleh EKSTRAK untuk timestamp dengan zona waktu (TIMESTAMPTZ) saja.

Variasi hasil dengan detik, milidetik, dan mikrodetik

Perbedaan kecil dalam hasil kueri terjadi ketika fungsi tanggal yang berbeda menentukan detik, milidetik, atau mikrodetik sebagai bagian tanggal:

- Fungsi EXTRACT mengembalikan bilangan bulat untuk bagian tanggal yang ditentukan saja, mengabaikan bagian tanggal tingkat yang lebih tinggi dan lebih rendah. Jika bagian tanggal yang ditentukan adalah detik, milidetik dan mikrodetik tidak termasuk dalam hasil. Jika bagian tanggal yang ditentukan adalah milidetik, detik dan mikrodetik tidak termasuk. Jika bagian tanggal yang ditentukan adalah mikrodetik, detik dan milidetik tidak termasuk.
- Fungsi DATE_PART mengembalikan bagian detik lengkap dari stempel waktu, terlepas dari bagian tanggal yang ditentukan, mengembalikan nilai desimal atau bilangan bulat sesuai kebutuhan.

Catatan CENTURY, EPOCH, DECADE, dan MIL

ABAD atau ABAD

AWS Clean Rooms menafsirkan CENTURY untuk memulai dengan tahun ## #1 dan diakhiri dengan tahun: ###0

```
select extract (century from timestamp '2000-12-16 12:21:13');
date_part
```

```
-----------
20
(1 row)

select extract (century from timestamp '2001-12-16 12:21:13');
date_part
-----------
21
(1 row)
```

EPOCH

AWS Clean Rooms Implementasi EPOCH relatif terhadap 1970-01-01 00:00:00.000 000 terlepas dari zona waktu di mana cluster berada. Anda mungkin perlu mengimbangi hasil dengan perbedaan jam tergantung pada zona waktu di mana cluster berada.

DEKADE atau DEKADE

AWS Clean Rooms menafsirkan DECADE atau DECADECADES DATEPART berdasarkan kalender umum. Misalnya, karena kalender umum dimulai dari tahun 1, dekade pertama (dekade 1) adalah 0001-01-01 hingga 0009-12-31, dan dekade kedua (dekade 2) adalah 0010-01-01 hingga 0019-12-31. Misalnya, dekade 201 membentang dari 2000-01-01 hingga 2009-12-31:

```
select extract(decade from timestamp '1999-02-16 20:38:40');
date_part
-----------
200
(1 row)

select extract(decade from timestamp '2000-02-16 20:38:40');
date_part
-----------
201
(1 row)

select extract(decade from timestamp '2010-02-16 20:38:40');
date_part
-----------
202
(1 row)
```

MIL atau MILS

AWS Clean Rooms menafsirkan MIL untuk memulai dengan hari pertama tahun #001 dan diakhiri dengan hari terakhir tahun#000:

```
select extract (mil from timestamp '2000-12-16 12:21:13');
date_part
-----------
2
(1 row)

select extract (mil from timestamp '2001-12-16 12:21:13');
date_part
-----------
3
(1 row)
```

Fungsi hash

Fungsi hash adalah fungsi matematika yang mengubah nilai input numerik menjadi nilai lain. AWS Clean Roomsmendukung fungsi hash berikut:

Topik

- Fungsi MD5
- Fungsi SHA
- Fungsi SHA1
- Fungsi SHA2
- MURMUR3_32_HASH

Fungsi MD5

Menggunakan fungsi hash kriptografi MD5 untuk mengubah string panjang variabel menjadi string 32 karakter yang merupakan representasi teks dari nilai heksadesimal dari checksum 128-bit.

Sintaksis

```
MD5(string)
```

Pendapat

tali

String panjang variabel.

Jenis pengembalian

Fungsi MD5 mengembalikan string 32-karakter yang merupakan representasi teks dari nilai heksadesimal dari checksum 128-bit.

Contoh

Contoh berikut menunjukkan nilai 128-bit untuk string "AWS Clean Rooms:

```
select md5('AWS Clean Rooms');
md5
----------------------------------
f7415e33f972c03abd4f3fed36748f7a
(1 row)
```

Fungsi SHA

Sinonim dari fungsi SHA1.

Lihat Fungsi SHA1.

Fungsi SHA1

Fungsi SHA1 menggunakan fungsi hash kriptografi SHA1 untuk mengubah string panjang variabel menjadi string 40 karakter yang merupakan representasi teks dari nilai heksadesimal dari checksum 160-bit.

Sintaksis

SHA1 adalah sinonim dari. Fungsi SHA

```
SHA1(string)
```

Pendapat

tali

String panjang variabel.

Jenis pengembalian

Fungsi SHA1 mengembalikan string 40 karakter yang merupakan representasi teks dari nilai heksadesimal dari checksum 160-bit.

Contoh

Contoh berikut mengembalikan nilai 160-bit untuk kata "AWS Clean Rooms:

```
select sha1('AWS Clean Rooms');
```

Fungsi SHA2

Fungsi SHA2 menggunakan fungsi hash kriptografi SHA2 untuk mengubah string panjang variabel menjadi string karakter. String karakter adalah representasi teks dari nilai heksadesimal checksum dengan jumlah bit yang ditentukan.

Sintaksis

```
SHA2(string, bits)
```

Pendapat

tali

String panjang variabel.

bilangan bulat

Jumlah bit dalam fungsi hash. Nilai yang valid adalah 0 (sama dengan 256), 224, 256, 384, dan 512.

Jenis pengembalian

Fungsi SHA2 mengembalikan string karakter yang merupakan representasi teks dari nilai heksadesimal checksum atau string kosong jika jumlah bit tidak valid.

Contoh

Contoh berikut mengembalikan nilai 256-bit untuk kata "AWS Clean Rooms:

```
select sha2('AWS Clean Rooms', 256);
```

MURMUR3_32_HASH

Fungsi MURMUR3_32_HASH menghitung hash non-kriptografi Murmur3A 32-bit untuk semua tipe data umum termasuk tipe numerik dan string.

Sintaksis

```
MURMUR3_32_HASH(value [, seed])
```

Pendapat

nilai

Nilai input untuk hash. AWS Clean Roomshash representasi biner dari nilai input. Perilaku ini mirip dengan FNV_HASH, tetapi nilainya dikonversi ke representasi biner yang ditentukan oleh spesifikasi hash Murmur3 32-bit Apache Iceberg.

benih

Benih INT dari fungsi hash. Argumen ini opsional. Jika tidak diberikan, AWS Clean Rooms gunakan seed default 0. Ini memungkinkan menggabungkan hash dari beberapa kolom tanpa konversi atau penggabungan apa pun.

Jenis pengembalian

Fungsi mengembalikan INT.

Contoh

Contoh berikut mengembalikan hash Murmur3 dari angka, string 'AWS Clean Rooms', dan rangkaian keduanya.

```
select MURMUR3_32_HASH(1);

    MURMUR3_32_HASH
----------------------
 -5968735742475085980
(1 row)
```

```
select MURMUR3_32_HASH('AWS Clean Rooms');

    MURMUR3_32_HASH
----------------------
 7783490368944507294
(1 row)
```

```
select MURMUR3_32_HASH('AWS Clean Rooms', MURMUR3_32_HASH(1));

    MURMUR3_32_HASH
----------------------
 -2202602717770968555
(1 row)
```

Catatan penggunaan

Untuk menghitung hash tabel dengan beberapa kolom, Anda dapat menghitung hash Murmur3 dari kolom pertama dan meneruskannya sebagai benih ke hash kolom kedua. Kemudian, ia melewati hash Murmur3 dari kolom kedua sebagai benih ke hash kolom ketiga.

Contoh berikut menciptakan benih untuk hash tabel dengan beberapa kolom.

```
select MURMUR3_32_HASH(column_3, MURMUR3_32_HASH(column_2, MURMUR3_32_HASH(column_1)))
 from sample_table;
```

Properti yang sama dapat digunakan untuk menghitung hash dari rangkaian string.

```
select MURMUR3_32_HASH('abcd');
```

```
   MURMUR3_32_HASH
---------------------
 -281581062704388899
(1 row)
```

```
select MURMUR3_32_HASH('cd', MURMUR3_32_HASH('ab'));

   MURMUR3_32_HASH
---------------------
 -281581062704388899
(1 row)
```

Fungsi hash menggunakan jenis input untuk menentukan jumlah byte untuk hash. Gunakan casting untuk menegakkan jenis tertentu, jika perlu.

Contoh berikut menggunakan jenis input yang berbeda untuk menghasilkan hasil yang berbeda.

```
select MURMUR3_32_HASH(1::smallint);

   MURMUR3_32_HASH
--------------------
 589727492704079044
(1 row)
```

```
select MURMUR3_32_HASH(1);

   MURMUR3_32_HASH
----------------------
 -5968735742475085980
(1 row)
```

```
select MURMUR3_32_HASH(1::bigint);

   MURMUR3_32_HASH
----------------------
 -8517097267634966620
(1 row)
```

Fungsi JSON

Ketika Anda perlu menyimpan kumpulan pasangan kunci-nilai yang relatif kecil, Anda dapat menghemat ruang dengan menyimpan data dalam format JSON. Karena string JSON dapat disimpan dalam satu kolom, menggunakan JSON mungkin lebih efisien daripada menyimpan data Anda dalam format tabel.

Example

Misalnya, Anda memiliki tabel jarang, di mana Anda harus memiliki banyak kolom untuk sepenuhnya mewakili semua atribut yang mungkin. Namun, sebagian besar nilai kolom adalah NULL untuk setiap baris tertentu atau kolom tertentu. Dengan menggunakan JSON untuk penyimpanan, Anda mungkin dapat menyimpan data untuk baris dalam pasangan kunci-nilai dalam string JSON tunggal dan menghilangkan kolom tabel yang jarang diisi.

Selain itu, Anda dapat dengan mudah memodifikasi string JSON untuk menyimpan pasangan kunci: nilai tambahan tanpa perlu menambahkan kolom ke tabel.

Kami merekomendasikan menggunakan JSON dengan hemat. JSON bukanlah pilihan yang baik untuk menyimpan kumpulan data yang lebih besar karena, dengan menyimpan data yang berbeda dalam satu kolom, JSON tidak menggunakan arsitektur penyimpanan kolom. AWS Clean Rooms

JSON menggunakan string teks yang dikodekan UTF-8, sehingga string JSON dapat disimpan sebagai tipe data CHAR atau VARCHAR. Gunakan VARCHAR jika string menyertakan karakter multi-byte.

String JSON harus benar diformat JSON, sesuai dengan aturan berikut:

- Tingkat root JSON dapat berupa objek JSON atau array JSON. Objek JSON adalah kumpulan pasangan kunci:nilai yang dipisahkan koma yang tidak berurutan yang diapit oleh kurawal kurawal.

  Misalnya, `{"one":1, "two":2}`
- Array JSON adalah sekumpulan nilai yang dipisahkan koma yang diurutkan yang diapit oleh tanda kurung.

  Contohnya adalah sebagai berikut: `["first", {"one":1}, "second", 3, null]`
- Array JSON menggunakan indeks berbasis nol; elemen pertama dalam array berada pada posisi 0. Dalam pasangan kunci JSON: nilai, kuncinya adalah string dalam tanda kutip ganda.

- Nilai JSON dapat berupa salah satu dari berikut ini:
  - Objek JSON
  - Array JSON
  - String dalam tanda kutip ganda
  - Angka (integer dan float)
  - Boolean
  - Nol
- Objek kosong dan array kosong adalah nilai JSON yang valid.
- Bidang JSON peka huruf besar/kecil.
- Ruang putih antara elemen struktural JSON (seperti`{ }, [ ]`) diabaikan.

Fungsi AWS Clean Rooms JSON dan perintah AWS Clean Rooms COPY menggunakan metode yang sama untuk bekerja dengan data berformat JSON.

Topik

- Fungsi CAN_JSON_PARSE
- Fungsi JSON_EXTRACT_ARRAY_ELEMENT_TEXT
- Fungsi JSON_EXTRACT_PATH_TEXT
- Fungsi JSON_PARSE
- Fungsi JSON_SERIALIZE
- Fungsi JSON_SERIALIZE_TO_VARBYTE

Fungsi CAN_JSON_PARSE

Fungsi CAN_JSON_PARSE mem-parsing data dalam format JSON dan mengembalikan `true` jika hasilnya dapat dikonversi ke nilai menggunakan fungsi JSON_PARSE. `SUPER`

Sintaksis

```
CAN_JSON_PARSE(json_string)
```

Pendapat

json_string

Ekspresi yang mengembalikan JSON serial dalam bentuk VARBYTE atauVARCHAR.

Jenis pengembalian

BOOLEAN

Contoh

Untuk melihat apakah array JSON [10001,10002,"abc"] dapat diubah menjadi tipe SUPER data, gunakan contoh berikut.

```
SELECT CAN_JSON_PARSE('[10001,10002,"abc"]');

+----------------+
| can_json_parse |
+----------------+
| true           |
+----------------+
```

Fungsi JSON_EXTRACT_ARRAY_ELEMENT_TEXT

Fungsi JSON_EXTRACT_ARRAY_ELEMENT_TEXT mengembalikan elemen array JSON dalam array terluar dari string JSON, menggunakan indeks berbasis nol. Elemen pertama dalam array berada pada posisi 0. Jika indeks negatif atau keluar dari terikat, JSON_EXTRACT_ARRAY_ELEMENT_TEXT mengembalikan string kosong. Jika argumen null_if_invalid disetel ke true dan string JSON tidak valid, fungsi mengembalikan NULL alih-alih mengembalikan kesalahan.

Untuk informasi selengkapnya, lihat Fungsi JSON.

Sintaksis

```
json_extract_array_element_text('json string', pos [, null_if_invalid ] )
```

Pendapat

json_string

String JSON yang diformat dengan benar.

pos

Sebuah integer yang mewakili indeks elemen array yang akan dikembalikan, menggunakan indeks array berbasis nol.

null_if_invalid

Nilai Boolean yang menentukan apakah akan mengembalikan NULL jika string input JSON tidak valid alih-alih mengembalikan kesalahan. Untuk mengembalikan NULL jika JSON tidak valid, tentukan (). `true` `t` Untuk mengembalikan kesalahan jika JSON tidak valid, tentukan `false` (). `f` Defaultnya adalah `false`.

Jenis pengembalian

Sebuah string VARCHAR mewakili elemen array JSON direferensikan oleh pos.

Contoh

Contoh berikut mengembalikan elemen array pada posisi 2, yang merupakan elemen ketiga dari indeks array berbasis nol:

```
select json_extract_array_element_text('[111,112,113]', 2);

json_extract_array_element_text
-------------------------------
113
```

Contoh berikut mengembalikan kesalahan karena JSON tidak valid.

```
select json_extract_array_element_text('["a",["b",1,["c",2,3,null,]]]',1);

An error occurred when executing the SQL command:
select json_extract_array_element_text('["a",["b",1,["c",2,3,null,]]]',1)
```

Contoh berikut menetapkan null_if_invalid ke true, sehingga pernyataan mengembalikan NULL alih-alih mengembalikan kesalahan untuk JSON yang tidak valid.

```
select json_extract_array_element_text('["a",["b",1,["c",2,3,null,]]]',1,true);

json_extract_array_element_text
-------------------------------
```

Fungsi JSON_EXTRACT_PATH_TEXT

Fungsi JSON_EXTRACT_PATH_TEXT mengembalikan nilai untuk pasangan key:value direferensikan oleh serangkaian elemen path dalam string JSON. Jalur JSON dapat bersarang hingga kedalaman lima tingkat. Elemen jalur peka huruf besar/kecil. Jika elemen path tidak ada dalam string JSON, JSON_EXTRACT_PATH_TEXT mengembalikan string kosong. Jika `null_if_invalid` argumen diatur ke `true` dan string JSON tidak valid, fungsi mengembalikan NULL bukannya mengembalikan kesalahan.

Untuk informasi tentang fungsi JSON tambahan, lihatFungsi JSON.

Sintaksis

```
json_extract_path_text('json_string', 'path_elem' [,'path_elem'[, …] ]
 [, null_if_invalid ] )
```

Pendapat

json_string

String JSON yang diformat dengan benar.

path_elem

Sebuah elemen path dalam string JSON. Satu elemen jalur diperlukan. Elemen jalur tambahan dapat ditentukan, hingga lima tingkat dalam.

null_if_invalid

Nilai Boolean yang menentukan apakah akan mengembalikan NULL jika string input JSON tidak valid alih-alih mengembalikan kesalahan. Untuk mengembalikan NULL jika JSON tidak valid, tentukan (). `true` `t` Untuk mengembalikan kesalahan jika JSON tidak valid, tentukan `false` (). `f` Defaultnya adalah `false`.

Dalam string JSON, AWS Clean Rooms mengenali `\n` sebagai karakter baris baru dan `\t` sebagai karakter tab. Untuk memuat garis miring terbalik, lepaskan dengan garis miring terbalik (). `\\`

Jenis pengembalian

VARCHAR string mewakili nilai JSON direferensikan oleh elemen jalur.

Contoh

Contoh berikut mengembalikan nilai untuk jalan`'f4', 'f6'`.

```
select json_extract_path_text('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}','f4', 'f6');


json_extract_path_text
----------------------
star
```

Contoh berikut mengembalikan kesalahan karena JSON tidak valid.

```
select json_extract_path_text('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}','f4', 'f6');

An error occurred when executing the SQL command:
select json_extract_path_text('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}','f4', 'f6')
```

Contoh berikut menetapkan null_if_invalid ke true, sehingga pernyataan mengembalikan NULL untuk JSON yang tidak valid alih-alih mengembalikan kesalahan.

```
select json_extract_path_text('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}','f4',
 'f6',true);

json_extract_path_text
------------------------------
NULL
```

Contoh berikut mengembalikan nilai untuk jalur`'farm', 'barn', 'color'`, di mana nilai diambil adalah pada tingkat ketiga. Sampel ini diformat dengan alat lint JSON, agar lebih mudah dibaca.

```
select json_extract_path_text('{
    "farm": {
        "barn": {
```

```
            "color": "red",
            "feed stocked": true
        }
    }
}', 'farm', 'barn', 'color');

json_extract_path_text
----------------------
red
```

Contoh berikut mengembalikan NULL karena `'color'` elemen hilang. Sampel ini diformat dengan alat lint JSON.

```
select json_extract_path_text('{
    "farm": {
        "barn": {}
    }
}', 'farm', 'barn', 'color');

json_extract_path_text
----------------------
NULL
```

Jika JSON valid, mencoba mengekstrak elemen yang hilang mengembalikan NULL.

Contoh berikut mengembalikan nilai untuk jalan`'house', 'appliances', 'washing machine', 'brand'`.

```
select json_extract_path_text('{
  "house": {
    "address": {
      "street": "123 Any St.",
      "city": "Any Town",
      "state": "FL",
      "zip": "32830"
    },
    "bathroom": {
      "color": "green",
      "shower": true
    },
    "appliances": {
      "washing machine": {
```

```
        "brand": "Any Brand",
        "color": "beige"
      },
      "dryer": {
        "brand": "Any Brand",
        "color": "white"
      }
    }
  }
}', 'house', 'appliances', 'washing machine', 'brand');

json_extract_path_text
----------------------
Any Brand
```

Fungsi JSON_PARSE

Fungsi JSON_PARSE mem-parsing data dalam format JSON dan mengubahnya menjadi representasi SUPER.

Untuk menyerap ke dalam tipe data SUPER menggunakan perintah INSERT atau UPDATE, gunakan fungsi JSON_PARSE. Saat Anda menggunakan JSON_PARSE () untuk mengurai string JSON menjadi nilai SUPER, pembatasan tertentu berlaku.

Sintaksis

```
JSON_PARSE(json_string)
```

Pendapat

json_string

Ekspresi yang mengembalikan JSON serial sebagai tipe varbyte atau varchar.

Jenis pengembalian

SUPER

Contoh

Contoh berikut adalah contoh dari fungsi JSON_PARSE.

```
SELECT JSON_PARSE('[10001,10002,"abc"]');
     json_parse
-------------------------
 [10001,10002,"abc"]
(1 row)
```

```
SELECT JSON_TYPEOF(JSON_PARSE('[10001,10002,"abc"]'));
 json_typeof
---------------
 array
(1 row)
```

Fungsi JSON_SERIALIZE

Fungsi JSON_SERIALIZE membuat serial ekspresi SUPER menjadi representasi JSON tekstual untuk mengikuti RFC 8259. Untuk informasi lebih lanjut tentang RFC itu, lihat Format Pertukaran Data Notasi JavaScript Objek (JSON).

Batas ukuran SUPER kira-kira sama dengan batas blok, dan batas varchar lebih kecil dari batas ukuran SUPER. Oleh karena itu, fungsi JSON_SERIALIZE mengembalikan kesalahan ketika format JSON melebihi batas varchar sistem.

Sintaksis

```
JSON_SERIALIZE(super_expression)
```

Pendapat

super_ekspresi

Ekspresi atau kolom super.

Jenis pengembalian

varchar

Contoh

Contoh berikut membuat serial nilai SUPER ke string.

```
 SELECT JSON_SERIALIZE(JSON_PARSE('[10001,10002,"abc"]'));
   json_serialize
--------------------
 [10001,10002,"abc"]
(1 row)
```

Fungsi JSON_SERIALIZE_TO_VARBYTE

Fungsi JSON_SERIALIZE_TO_VARBYTE mengonversi nilai SUPER ke string JSON yang mirip dengan JSON_SERIALIZE (), tetapi disimpan dalam nilai VARBYTE sebagai gantinya.

Sintaksis

```
JSON_SERIALIZE_TO_VARBYTE(super_expression)
```

Pendapat

super_ekspresi

Ekspresi atau kolom super.

Jenis pengembalian

varbyte

Contoh

Contoh berikut membuat serial nilai SUPER dan mengembalikan hasilnya dalam format VARBYTE.

```
SELECT JSON_SERIALIZE_TO_VARBYTE(JSON_PARSE('[10001,10002,"abc"]'));
```

```
       json_serialize_to_varbyte
----------------------------------------
 5b31303030312c31303030322c22616263225d
```

Contoh berikut membuat serial nilai SUPER dan melemparkan hasilnya ke format VARCHAR.

```
SELECT JSON_SERIALIZE_TO_VARBYTE(JSON_PARSE('[10001,10002,"abc"]'))::VARCHAR;
```

```
  json_serialize_to_varbyte
----------------------------
 [10001,10002,"abc"]
```

Fungsi matematika

Bagian ini menjelaskan operator matematika dan fungsi yang didukung dalam AWS Clean Rooms.

Topik

- Fungsi ROUND
- Fungsi SIGN
- Fungsi SIN
- Fungsi SQRT
- Fungsi TRUNC

Simbol operator matematika

Tabel berikut mencantumkan operator matematika yang didukung.

Operator yang didukung

Operator	Deskripsi	Contoh	Hasil
+	tambahan	2 + 3	5
-	pengurangan	2 - 3	-1
*	perkalian	2 * 3	6
/	pembagian	4/2	2
%	modulo	5% 4	1
^	eksponensial	2.0 ^ 3.0	8
|/	akar kuadrat	|/25.0	5
||/	akar kubus	||/27,0	3
@	nilai absolut	@ -5.0	5

Contoh-contoh

Hitung komisi yang dibayarkan ditambah biaya penanganan $2,00 untuk transaksi tertentu:

```
select commission, (commission + 2.00) as comm
from sales where salesid=10000;

commission | comm
-----------+-------
28.05      | 30.05
(1 row)
```

Hitung 20 persen dari harga jual untuk transaksi tertentu:

```
select pricepaid, (pricepaid * .20) as twentypct
from sales where salesid=10000;

pricepaid | twentypct
----------+-----------
187.00    |    37.400
(1 row)
```

Forecast penjualan tiket berdasarkan pola pertumbuhan berkelanjutan. Dalam contoh ini, subquery mengembalikan jumlah tiket yang terjual pada tahun 2008. Hasil itu dikalikan secara eksponensial dengan tingkat pertumbuhan berkelanjutan sebesar 5 persen selama 10 tahun.

```
select (select sum(qtysold) from sales, date
where sales.dateid=date.dateid and year=2008)
^ ((5::float/100)*10) as qty10years;

qty10years
------------------
587.664019657491
(1 row)
```

Temukan total harga yang dibayarkan dan komisi untuk penjualan dengan ID tanggal yang lebih besar dari atau sama dengan 2.000. Kemudian kurangi total komisi dari total harga yang dibayarkan.

```
select sum (pricepaid) as sum_price, dateid,
sum (commission) as sum_comm, (sum (pricepaid) - sum (commission)) as value
from sales where dateid >= 2000
group by dateid order by dateid limit 10;

 sum_price | dateid | sum_comm |   value
-----------+--------+----------+-----------
```

```
 364445.00 |   2044 | 54666.75 | 309778.25
 349344.00 |   2112 | 52401.60 | 296942.40
 343756.00 |   2124 | 51563.40 | 292192.60
 378595.00 |   2116 | 56789.25 | 321805.75
 328725.00 |   2080 | 49308.75 | 279416.25
 349554.00 |   2028 | 52433.10 | 297120.90
 249207.00 |   2164 | 37381.05 | 211825.95
 285202.00 |   2064 | 42780.30 | 242421.70
 320945.00 |   2012 | 48141.75 | 272803.25
 321096.00 |   2016 | 48164.40 | 272931.60
(10 rows)
```

Fungsi ABS

ABS menghitung nilai absolut dari suatu angka, di mana angka itu dapat berupa literal atau ekspresi yang mengevaluasi angka.

Sintaks

```
ABS (number)
```

Argumen

jumlah

Angka atau ekspresi yang mengevaluasi angka. Ini bisa berupa tipe SMALLINT, INTEGER, BIGINT, DECIMAL, FLOAT4, atau FLOAT8.

Jenis pengembalian

ABS mengembalikan tipe data yang sama dengan argumennya.

Contoh-contoh

Hitung nilai absolut -38:

```
select abs (-38);
abs
-------
38
```

```
(1 row)
```

Hitung nilai absolut (14-76):

```
select abs (14-76);
abs
-------
62
(1 row)
```

Fungsi ACOS

ACOS adalah fungsi trigonometri yang mengembalikan kosinus busur suatu angka. Nilai kembali dalam radian dan berada di antara `0` dan`PI`.

Sintaks

```
ACOS(number)
```

Argumen

jumlah

Parameter input adalah `DOUBLE PRECISION` angka.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh-contoh

Untuk mengembalikan arc cosinus`-1`, gunakan contoh berikut.

```
SELECT ACOS(-1);

+-------------------+
|       acos        |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

Fungsi ASIN

ASIN adalah fungsi trigonometri yang mengembalikan sinus busur dari suatu angka. Nilai kembali dalam radian dan berada di antara `PI/2` dan`-PI/2`.

Sintaks

```
ASIN(number)
```

Argumen

jumlah

Parameter input adalah `DOUBLE PRECISION` angka.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh-contoh

Untuk mengembalikan sinus busur1, gunakan contoh berikut.

```
SELECT ASIN(1) AS halfpi;

+--------------------+
|       halfpi       |
+--------------------+
| 1.5707963267948966 |
+--------------------+
```

Fungsi ATAN

ATAN adalah fungsi trigonometri yang mengembalikan garis singgung busur dari suatu bilangan. Nilai kembali dalam radian dan berada di antara `-PI` dan`PI`.

Sintaks

```
ATAN(number)
```

Argumen

jumlah

Parameter input adalah DOUBLE PRECISION angka.

Jenis pengembalian

DOUBLE PRECISION

Contoh-contoh

Untuk mengembalikan garis singgung busur 1 dan kalikan dengan 4, gunakan contoh berikut.

```
SELECT ATAN(1) * 4 AS pi;

+-------------------+
|        pi         |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

Fungsi ATAN2

ATAN2 adalah fungsi trigonometri yang mengembalikan tangen busur dari satu angka dibagi dengan angka lain. Nilai kembali dalam radian dan berada di antara PI/2 dan-PI/2.

Sintaks

```
ATAN2(number1, number2)
```

Argumen

nomor1

Sebuah DOUBLE PRECISION angka.

nomor2

Sebuah DOUBLE PRECISION angka.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh-contoh

Untuk mengembalikan garis singgung busur 2/2 dan kalikan dengan 4, gunakan contoh berikut.

```
SELECT ATAN2(2,2) * 4 AS PI;

+-------------------+
|        pi         |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

Fungsi CBRT

Fungsi CBRT adalah fungsi matematika yang menghitung akar kubus dari suatu angka.

Sintaks

```
CBRT (number)
```

Pendapat

CBRT mengambil nomor PRESISI GANDA sebagai argumen.

Jenis pengembalian

CBRT mengembalikan nomor PRESISI GANDA.

Contoh-contoh

Hitung akar kubus dari komisi yang dibayarkan untuk transaksi tertentu:

```
select cbrt(commission) from sales where salesid=10000;

cbrt
------------------
```

```
3.03839539048843
(1 row)
```

Fungsi CEILING (atau CEIL)

Fungsi CEILING atau CEIL digunakan untuk membulatkan angka ke bilangan bulat berikutnya. (Fungsi FLOORMembulatkan angka ke bawah ke bilangan bulat berikutnya.)

Sintaks

```
CEIL | CEILING(number)
```

Argumen

jumlah

Angka atau ekspresi yang mengevaluasi angka. Ini bisa berupa tipe SMALLINT, INTEGER, BIGINT, DECIMAL, FLOAT4, atau FLOAT8.

Jenis pengembalian

CEILING dan CEIL mengembalikan tipe data yang sama dengan argumennya.

Contoh

Hitung plafon komisi yang dibayarkan untuk transaksi penjualan tertentu:

```
select ceiling(commission) from sales
where salesid=10000;

ceiling
---------
29
(1 row)
```

Fungsi COS

COS adalah fungsi trigonometri yang mengembalikan kosinus suatu bilangan. Nilai kembalinya dalam radian dan berada di antara -1 dan1, inklusif.

Sintaks

```
COS(double_precision)
```

Pendapat

jumlah

Parameter input adalah angka presisi ganda.

Jenis pengembalian

Fungsi COS mengembalikan angka presisi ganda.

Contoh-contoh

Contoh berikut mengembalikan cosinus dari 0:

```
select cos(0);
cos
-----
1
(1 row)
```

Contoh berikut mengembalikan kosinus PI:

```
select cos(pi());
cos
-----
-1
(1 row)
```

Fungsi COT

COT adalah fungsi trigonometri yang mengembalikan kotangen angka. Parameter input harus bukan nol.

Sintaks

```
COT(number)
```

Pendapat

jumlah

Parameter input adalah `DOUBLE PRECISION` angka.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh-contoh

Untuk mengembalikan kotangen 1, gunakan contoh berikut.

```
SELECT COT(1);

+--------------------+
|        cot         |
+--------------------+
| 0.6420926159343306 |
+--------------------+
```

Fungsi DERAJAT

Mengubah sudut dalam radian menjadi setara dalam derajat.

Sintaks

```
DEGREES(number)
```

Pendapat

jumlah

Parameter input adalah `DOUBLE PRECISION` angka.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

Untuk mengembalikan derajat setara dengan 0,5 radian, gunakan contoh berikut.

```
SELECT DEGREES(.5);

+-------------------+
|      degrees      |
+-------------------+
| 28.64788975654116 |
+-------------------+
```

Untuk mengkonversi PI radian ke derajat, gunakan contoh berikut.

```
SELECT DEGREES(pi());

+---------+
| degrees |
+---------+
|     180 |
+---------+
```

Fungsi DEXP

Fungsi DEXP mengembalikan nilai eksponensial dalam notasi ilmiah untuk bilangan presisi ganda. Satu-satunya perbedaan antara fungsi DEXP dan EXP adalah bahwa parameter untuk DEXP harus a. `DOUBLE PRECISION`

Sintaks

```
DEXP(number)
```

Pendapat

jumlah

Parameter input adalah `DOUBLE PRECISION` angka.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

```
SELECT (SELECT SUM(qtysold)
FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * DEXP((7::FLOAT/100)*10) qty2010;

+-------------------+
|      qty2010      |
+-------------------+
| 695447.4837722216 |
+-------------------+
```

Fungsi DLOG1

Fungsi DLOG1 mengembalikan logaritma natural dari parameter input.

Fungsi DLOG1 adalah sinonim dari. Fungsi LN

Fungsi DLOG10

DLOG10 mengembalikan logaritma basis 10 dari parameter input.

Fungsi DLOG10 adalah sinonim dari. Fungsi LOG

Sintaks

```
DLOG10(number)
```

Pendapat

jumlah

Parameter input adalah angka presisi ganda.

Jenis pengembalian

Fungsi DLOG10 mengembalikan angka presisi ganda.

Contoh

Contoh berikut mengembalikan basis 10 logaritma dari angka 100:

```
select dlog10(100);

dlog10
--------
2
(1 row)
```

Fungsi EXP

Fungsi EXP mengimplementasikan fungsi eksponensial untuk ekspresi numerik, atau dasar logaritma natural,, e dinaikkan ke kekuatan ekspresi. Fungsi EXP adalah kebalikan dari. Fungsi LN

Sintaks

```
EXP (expression)
```

Pendapat

ekspresi

Ekspresi harus berupa tipe data INTEGER, DECIMAL, atau DOUBLE PRECISION.

Jenis pengembalian

EXP mengembalikan nomor PRESISI GANDA.

Contoh

Gunakan fungsi EXP untuk memperkirakan penjualan tiket berdasarkan pola pertumbuhan berkelanjutan. Dalam contoh ini, subquery mengembalikan jumlah tiket yang terjual pada tahun 2008. Hasil itu dikalikan dengan hasil fungsi EXP, yang menentukan tingkat pertumbuhan berkelanjutan 7% selama 10 tahun.

```
select (select sum(qtysold) from sales, date
where sales.dateid=date.dateid
and year=2008) * exp((7::float/100)*10) qty2018;

qty2018
------------------
695447.483772222
```

```
(1 row)
```

Fungsi FLOOR

Fungsi FLOOR membulatkan angka ke bawah ke bilangan bulat berikutnya.

Sintaks

```
FLOOR (number)
```

Pendapat

jumlah

Angka atau ekspresi yang mengevaluasi angka. Ini bisa berupa tipe SMALLINT, INTEGER, BIGINT, DECIMAL, FLOAT4, atau FLOAT8.

Jenis pengembalian

FLOOR mengembalikan tipe data yang sama sebagai argumennya.

Contoh

Contoh menunjukkan nilai komisi yang dibayarkan untuk transaksi penjualan tertentu sebelum dan sesudah menggunakan fungsi FLOOR.

```
select commission from sales
where salesid=10000;

floor
-------
28.05
(1 row)

select floor(commission) from sales
where salesid=10000;

floor
-------
28
(1 row)
```

Fungsi LN

Fungsi LN mengembalikan logaritma natural dari parameter input.

Fungsi LN adalah sinonim dari. Fungsi DLOG1

Sintaks

```
LN(expression)
```

Pendapat

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

> **Note**
>
> Fungsi ini mengembalikan kesalahan untuk beberapa tipe data jika ekspresi referensi tabel yang AWS Clean Rooms dibuat pengguna atau tabel sistem AWS Clean Rooms STL atau STV.

Ekspresi dengan tipe data berikut menghasilkan kesalahan jika mereka mereferensikan tabel yang dibuat pengguna atau sistem.

- BOOLEAN
- CHAR
- DATE
- DESIMAL atau NUMERIK
- TIMESTAMP
- VARCHAR

Ekspresi dengan tipe data berikut berjalan dengan sukses pada tabel yang dibuat pengguna dan tabel sistem STL atau STV:

- BIGINT
- DOUBLE PRECISION
- INTEGER

- REAL
- SMALLINT

Jenis pengembalian

Fungsi LN mengembalikan tipe yang sama dengan ekspresi.

Contoh

Contoh berikut mengembalikan logaritma natural, atau logaritma basis e, dari angka 2.718281828:

```
select ln(2.718281828);
ln
--------------------
0.9999999998311267
(1 row)
```

Perhatikan bahwa jawabannya hampir sama dengan 1.

Contoh ini mengembalikan logaritma natural dari nilai-nilai dalam kolom USERID dalam tabel USERID:

```
select username, ln(userid) from users order by userid limit 10;

 username |        ln
----------+-------------------
 JSG99FHE |                 0
 PGL08LJI | 0.693147180559945
 IFT66TXU |  1.09861228866811
 XDZ38RDD |  1.38629436111989
 AEB55QTM |   1.6094379124341
 NDQ15VBM |  1.79175946922805
 OWY35QYB |  1.94591014905531
 AZG78YIP |  2.07944154167984
 MSD36KVR |  2.19722457733622
 WKW41AIW |  2.30258509299405
(10 rows)
```

Fungsi LOG

Mengembalikan basis 10 logaritma dari sebuah angka.

Sinonim dari. Fungsi DLOG10

Sintaks

```
LOG(number)
```

Pendapat

jumlah

Parameter input adalah angka presisi ganda.

Jenis pengembalian

Fungsi LOG mengembalikan nomor presisi ganda.

Contoh

Contoh berikut mengembalikan basis 10 logaritma dari angka 100:

```
select log(100);
dlog10
--------
2
(1 row)
```

Fungsi MOD

Mengembalikan sisa dari dua angka, atau dikenal sebagai operasi modulo. Untuk menghitung hasilnya, parameter pertama dibagi dengan yang kedua.

Sintaks

```
MOD(number1, number2)
```

Argumen

nomor1

Parameter input pertama adalah bilangan INTEGER, SMALLINT, BIGINT, atau DECIMAL. Jika salah satu parameter adalah tipe DECIMAL, parameter lainnya juga harus tipe DECIMAL. Jika

salah satu parameter adalah INTEGER, parameter lainnya dapat berupa INTEGER, SMALLINT, atau BIGINT. Kedua parameter juga dapat berupa SMALLINT atau BIGINT, tetapi satu parameter tidak dapat menjadi SMALLINT jika yang lain adalah BIGINT.

nomor2

Parameter kedua adalah bilangan INTEGER, SMALLINT, BIGINT, atau DECIMAL. Aturan tipe data yang sama berlaku untuk number2 untuk number1.

Jenis pengembalian

Jenis pengembalian yang valid adalah DECIMAL, INT, SMALLINT, dan BIGINT. Jenis pengembalian fungsi MOD adalah tipe numerik yang sama dengan parameter input, jika kedua parameter input adalah tipe yang sama. Jika salah satu parameter input adalah INTEGER, bagaimanapun, tipe kembali juga akan menjadi INTEGER.

Catatan penggunaan

Anda dapat menggunakan% sebagai operator modulo.

Contoh-contoh

Contoh berikut mengembalikan sisanya ketika angka dibagi dengan yang lain:

```
SELECT MOD(10, 4);

 mod
------
 2
```

Contoh berikut mengembalikan hasil desimal:

```
SELECT MOD(10.5, 4);

 mod
------
 2.5
```

Anda dapat mentransmisikan nilai parameter:

```
SELECT MOD(CAST(16.4 as integer), 5);
```

```
 mod
------
 1
```

Periksa apakah parameter pertama genap dengan membaginya dengan 2:

```
SELECT mod(5,2) = 0 as is_even;

 is_even
--------
 false
```

Anda dapat menggunakan% sebagai operator modulo:

```
SELECT 11 % 4 as remainder;

 remainder
-----------
 3
```

Contoh berikut mengembalikan informasi untuk kategori bernomor ganjil dalam tabel CATEGORY:

```
select catid, catname
from category
where mod(catid,2)=1
order by 1,2;

 catid |  catname
-------+-----------
     1 | MLB
     3 | NFL
     5 | MLS
     7 | Plays
     9 | Pop
    11 | Classical

(6 rows)
```

Fungsi PI

Fungsi PI mengembalikan nilai pi ke 14 tempat desimal.

Sintaks

```
PI()
```

Jenis pengembalian

`DOUBLE PRECISION`

Contoh-contoh

Untuk mengembalikan nilai pi, gunakan contoh berikut.

```
SELECT PI();

+-------------------+
|        pi         |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

Fungsi POWER

Fungsi POWER adalah fungsi eksponensial yang meningkatkan ekspresi numerik ke kekuatan ekspresi numerik kedua. Misalnya, 2 hingga daya ketiga dihitung sebagai`POWER(2,3)`, dengan hasil dari8.

Sintaks

```
{POW | POWER}(expression1, expression2)
```

Argumen

ekspresi1

Ekspresi numerik yang akan dinaikkan. Harus berupa`INTEGER`,`DECIMAL`, atau tipe `FLOAT` data.

ekspresi2

Kekuatan untuk meningkatkan ekspresi1. Harus berupa`INTEGER`,`DECIMAL`, atau tipe `FLOAT` data.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

```
SELECT (SELECT SUM(qtysold) FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * POW((1+7::FLOAT/100),10) qty2010;

+------------------+
|      qty2010     |
+------------------+
| 679353.7540885945 |
+------------------+
```

Fungsi RADIANS

Fungsi RADIANS mengubah sudut dalam derajat ke ekuivalennya dalam radian.

Sintaks

```
RADIANS(number)
```

Pendapat

jumlah

Parameter input adalah `DOUBLE PRECISION` angka.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

Untuk mengembalikan radian setara 180 derajat, gunakan contoh berikut.

```
SELECT RADIANS(180);

+------------------+
```

```
|      radians      |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

fungsi RANDOM

Fungsi RANDOM menghasilkan nilai acak antara 0,0 (inklusif) dan 1,0 (eksklusif).

Sintaks

```
RANDOM()
```

Jenis pengembalian

RANDOM mengembalikan nomor PRESISI GANDA.

Contoh-contoh

1. Hitung nilai acak antara 0 dan 99. Jika angka acak adalah 0 hingga 1, kueri ini menghasilkan angka acak dari 0 hingga 100:

   ```
   select cast (random() * 100 as int);

   INTEGER
   ------
   24
   (1 row)
   ```

2. Ambil sampel acak seragam dari 10 item:

   ```
   select *
   from sales
   order by random()
   limit 10;
   ```

   Sekarang ambil sampel acak 10 item, tetapi pilih item secara proporsional dengan harganya. Misalnya, item yang dua kali harga yang lain akan dua kali lebih mungkin muncul dalam hasil kueri:

   ```
   select *
   ```

```
from sales
order by log(1 - random()) / pricepaid
limit 10;
```

3. Contoh ini menggunakan perintah SET untuk menetapkan nilai SEED sehingga RANDOM menghasilkan urutan angka yang dapat diprediksi.

   Pertama, kembalikan tiga bilangan bulat RANDOM tanpa mengatur nilai SEED terlebih dahulu:

```
select cast (random() * 100 as int);
INTEGER
------
6
(1 row)

select cast (random() * 100 as int);
INTEGER
------
68
(1 row)

select cast (random() * 100 as int);
INTEGER
------
56
(1 row)
```

   Sekarang, atur nilai SEED ke `.25`, dan kembalikan tiga angka RANDOM lagi:

```
set seed to .25;
select cast (random() * 100 as int);
INTEGER
------
21
(1 row)

select cast (random() * 100 as int);
INTEGER
------
79
(1 row)

select cast (random() * 100 as int);
```

```
INTEGER
------
12
(1 row)
```

Terakhir, setel ulang nilai SEED ke.25, dan verifikasi bahwa RANDOM mengembalikan hasil yang sama dengan tiga panggilan sebelumnya:

```
set seed to .25;
select cast (random() * 100 as int);
INTEGER
------
21
(1 row)

select cast (random() * 100 as int);
INTEGER
------
79
(1 row)

select cast (random() * 100 as int);
INTEGER
------
12
(1 row)
```

Fungsi ROUND

Fungsi ROUND membulatkan angka ke bilangan bulat atau desimal terdekat.

Fungsi ROUND secara opsional dapat menyertakan argumen kedua sebagai bilangan bulat untuk menunjukkan jumlah tempat desimal untuk pembulatan, di kedua arah. Ketika Anda tidak memberikan argumen kedua, fungsi dibulatkan ke bilangan bulat terdekat. Ketika argumen kedua >n ditentukan, fungsi dibulatkan ke angka terdekat dengan n tempat desimal presisi.

Sintaks

```
ROUND (number [ , integer ] )
```

Pendapat

jumlah

Angka atau ekspresi yang mengevaluasi angka. Ini bisa berupa tipe DECIMAL atau FLOAT8. AWS Clean Rooms dapat mengonversi tipe data lain sesuai aturan konversi implisit.

bilangan bulat (opsional)

Bilangan bulat yang menunjukkan jumlah tempat desimal untuk pembulatan di kedua arah.

Jenis pengembalian

ROUND mengembalikan tipe data numerik yang sama dengan argumen masukan.

Contoh-contoh

Bulatkan komisi yang dibayarkan untuk transaksi tertentu ke seluruh nomor terdekat.

```
select commission, round(commission)
from sales where salesid=10000;

commission | round
-----------+-------
     28.05 |    28
(1 row)
```

Bulatkan komisi yang dibayarkan untuk transaksi tertentu ke tempat desimal pertama.

```
select commission, round(commission, 1)
from sales where salesid=10000;

commission | round
-----------+-------
     28.05 |  28.1
(1 row)
```

Untuk kueri yang sama, perluas presisi ke arah yang berlawanan.

```
select commission, round(commission, -1)
from sales where salesid=10000;
```

```
commission | round
-----------+-------
     28.05 |    30
(1 row)
```

Fungsi SIGN

Fungsi SIGN mengembalikan tanda (positif atau negatif) dari suatu angka. Hasil dari fungsi SIGN adalah `1`-`1`,, atau `0` menunjukkan tanda argumen.

Sintaks

```
SIGN (number)
```

Pendapat

jumlah

Angka atau ekspresi yang mengevaluasi angka. Ini bisa menjadi tipe Decimalor FLOAT8. AWS Clean Rooms dapat mengonversi tipe data lain sesuai aturan konversi implisit.

Jenis pengembalian

SIGN mengembalikan tipe data numerik yang sama dengan argumen masukan. Jika input adalah DESIMAL, outputnya adalah DESIMAL (1,0).

Contoh-contoh

Untuk menentukan tanda komisi yang dibayarkan untuk transaksi tertentu dari tabel PENJUALAN, gunakan contoh berikut.

```
SELECT commission, SIGN(commission)
FROM sales WHERE salesid=10000;

+------------+------+
| commission | sign |
+------------+------+
|      28.05 |    1 |
+------------+------+
```

Fungsi SIN

SIN adalah fungsi trigonometri yang mengembalikan sinus suatu angka. Nilai yang dikembalikan adalah antara `-1` dan`1`.

Sintaks

```
SIN(number)
```

Pendapat

jumlah

`DOUBLE PRECISION`Angka dalam radian.

Jenis pengembalian

`DOUBLE PRECISION`

Contoh

Untuk mengembalikan sinus`-PI`, gunakan contoh berikut.

```
SELECT SIN(-PI());

+-------------------------+
|           sin           |
+-------------------------+
| -0.00000000000000012246 |
+-------------------------+
```

Fungsi SQRT

Fungsi SQRT mengembalikan akar kuadrat dari nilai numerik. Akar kuadrat adalah angka yang dikalikan dengan sendirinya untuk mendapatkan nilai yang diberikan.

Sintaks

```
SQRT (expression)
```

Pendapat

ekspresi

Ekspresi harus memiliki tipe data integer, desimal, atau floating-point. Ekspresi dapat mencakup fungsi. Sistem mungkin melakukan konversi tipe implisit.

Jenis pengembalian

SQRT mengembalikan nomor PRESISI GANDA.

Contoh-contoh

Contoh berikut mengembalikan akar kuadrat dari angka.

```
select sqrt(16);

sqrt
---------------
4
```

Contoh berikut melakukan konversi tipe implisit.

```
select sqrt('16');

sqrt
---------------
4
```

Contoh sarang berikut berfungsi untuk melakukan tugas yang lebih kompleks.

```
select sqrt(round(16.4));

sqrt
---------------
4
```

Contoh berikut menghasilkan panjang jari-jari ketika diberi luas lingkaran. Ini menghitung radius dalam inci, misalnya, ketika diberi luas dalam inci persegi. Area dalam sampel adalah 20.

```
select sqrt(20/pi());
```

Ini mengembalikan nilai 5.046265044040321.

Contoh berikut mengembalikan akar kuadrat untuk nilai KOMISI dari tabel PENJUALAN. Kolom KOMISI adalah kolom DESIMAL. Contoh ini menunjukkan bagaimana Anda dapat menggunakan fungsi dalam kueri dengan logika kondisional yang lebih kompleks.

```
select sqrt(commission)
from sales where salesid < 10 order by salesid;

sqrt
------------------
10.4498803820905
3.37638860322683
7.24568837309472
5.1234753829798
...
```

Kueri berikut mengembalikan akar kuadrat bulat untuk set nilai KOMISI yang sama.

```
select salesid, commission, round(sqrt(commission))
from sales where salesid < 10 order by salesid;

salesid | commission | round
--------+------------+-------
      1 |     109.20 |    10
      2 |      11.40 |     3
      3 |      52.50 |     7
      4 |      26.25 |     5
...
```

Untuk informasi selengkapnya tentang data sampel di AWS Clean Rooms, lihat Database sampel.

Fungsi TRUNC

Fungsi TRUNC memotong angka ke bilangan bulat atau desimal sebelumnya.

Fungsi TRUNC secara opsional dapat menyertakan argumen kedua sebagai bilangan bulat untuk menunjukkan jumlah tempat desimal untuk pembulatan, di kedua arah. Ketika Anda tidak

memberikan argumen kedua, fungsi dibulatkan ke bilangan bulat terdekat. Ketika argumen kedua >n ditentukan, fungsi dibulatkan ke angka terdekat dengan>n tempat desimal presisi. Fungsi ini juga memotong stempel waktu dan mengembalikan tanggal.

Sintaks

```
TRUNC (number [ , integer ] |
timestamp )
```

Argumen

jumlah

Angka atau ekspresi yang mengevaluasi angka. Ini bisa berupa tipe DECIMAL atau FLOAT8. AWS Clean Rooms dapat mengonversi tipe data lain sesuai aturan konversi implisit.

bilangan bulat (opsional)

Bilangan bulat yang menunjukkan jumlah tempat desimal presisi, di kedua arah. Jika tidak ada bilangan bulat yang disediakan, angka tersebut terpotong sebagai bilangan bulat; jika bilangan bulat ditentukan, angka tersebut dipotong ke tempat desimal yang ditentukan.

stempel waktu

Fungsi ini juga dapat mengembalikan tanggal dari stempel waktu. (Untuk mengembalikan nilai stempel waktu dengan `00:00:00` waktu, lemparkan hasil fungsi ke stempel waktu.)

Jenis pengembalian

TRUNC mengembalikan tipe data yang sama dengan argumen masukan pertama. Untuk stempel waktu, TRUNC mengembalikan tanggal.

Contoh-contoh

Memangkas komisi yang dibayarkan untuk transaksi penjualan tertentu.

```
select commission, trunc(commission)
from sales where salesid=784;

commission | trunc
```

```
-----------+-------
    111.15 |   111

(1 row)
```

Memangkas nilai komisi yang sama ke tempat desimal pertama.

```
select commission, trunc(commission,1)
from sales where salesid=784;

commission | trunc
-----------+-------
    111.15 | 111.1

(1 row)
```

Potong komisi dengan nilai negatif untuk argumen kedua; `111.15` dibulatkan ke bawah. `110`

```
select commission, trunc(commission,-1)
from sales where salesid=784;

commission | trunc
-----------+-------
    111.15 |   110
(1 row)
```

Kembalikan bagian tanggal dari hasil fungsi SYSDATE (yang mengembalikan stempel waktu):

```
select sysdate;

timestamp
----------------------------
2011-07-21 10:32:38.248109
(1 row)

select trunc(sysdate);

trunc
------------
2011-07-21
(1 row)
```

Terapkan fungsi TRUNC ke kolom TIMESTAMP. Jenis pengembalian adalah tanggal.

```
select trunc(starttime) from event
order by eventid limit 1;

trunc
------------
2008-01-25
(1 row)
```

Fungsi string

Topik

- Fungsi REPLICATE
- Fungsi REVERSE
- Fungsi RTRIM
- Fungsi SOUNDEX
- Fungsi SPLIT_PART
- fungsi STRPOS
- Fungsi SUBSTR
- Fungsi SUBSTRING
- Fungsi TEXTLEN
- FUNGSI TRANSLATE
- Fungsi TRIM
- Fungsi UPPER

Fungsi string memproses dan memanipulasi string karakter atau ekspresi yang mengevaluasi string karakter. Ketika argumen string dalam fungsi ini adalah nilai literal, itu harus diapit dalam tanda kutip tunggal. Tipe data yang didukung termasuk CHAR dan VARCHAR.

Bagian berikut menyediakan nama fungsi, sintaks, dan deskripsi untuk fungsi yang didukung. Semua offset menjadi string berbasis satu.

|| Operator (Penggabungan)

Menggabungkan dua ekspresi di kedua sisi simbol || dan mengembalikan ekspresi gabungan.

Operator penggabungan mirip dengan. Fungsi CONCAT

> **Note**
>
> Untuk kedua fungsi CONCAT dan operator penggabungan, jika salah satu atau kedua ekspresi adalah nol, hasil penggabungan adalah nol.

Sintaks

```
expression1 || expression2
```

Argumen

ekspresi1, ekspresi2

Kedua argumen dapat berupa string atau ekspresi karakter fixed-length atau variable-length.

Jenis pengembalian

Operator || mengembalikan string. Jenis string sama dengan argumen input.

Contoh

Contoh berikut menggabungkan bidang FIRSTNAME dan LASTNAME dari tabel USERS:

```
select firstname || ' ' || lastname
from users
order by 1
limit 10;

concat
-----------------
Aaron Banks
Aaron Booth
Aaron Browning
Aaron Burnett
Aaron Casey
Aaron Cash
Aaron Castro
Aaron Dickerson
Aaron Dixon
Aaron Dotson
(10 rows)
```

Untuk menggabungkan kolom yang mungkin berisi nol, gunakan ekspresi. Fungsi NVL dan COALESCE Contoh berikut menggunakan NVL untuk mengembalikan 0 setiap kali NULL ditemui.

```
select venuename || ' seats ' || nvl(venueseats, 0)
from venue where venuestate = 'NV' or venuestate = 'NC'
order by 1
```

```
limit 10;

seating
----------------------------------
Ballys Hotel seats 0
Bank of America Stadium seats 73298
Bellagio Hotel seats 0
Caesars Palace seats 0
Harrahs Hotel seats 0
Hilton Hotel seats 0
Luxor Hotel seats 0
Mandalay Bay Hotel seats 0
Mirage Hotel seats 0
New York New York seats 0
```

Fungsi BTRIM

Fungsi BTRIM memangkas string dengan menghapus bagian depan dan belakang kosong atau dengan menghapus karakter utama dan belakang yang cocok dengan string tertentu opsional.

Sintaks

```
BTRIM(string [, trim_chars ] )
```

Argumen

tali

String input VARCHAR yang akan dipangkas.

trim_chars

String VARCHAR yang berisi karakter yang akan dicocokkan.

Jenis pengembalian

Fungsi BTRIM mengembalikan string VARCHAR.

Contoh-contoh

Contoh berikut memangkas bagian depan dan belakang kosong dari string: `' abc '`

```
select '     abc    ' as untrim, btrim('     abc    ') as trim;

untrim    | trim
----------+------
   abc    | abc
```

Contoh berikut menghapus string depan dan trailing dari `'xyz'` string. `'xyzaxyzbxyzcxyz'` Kejadian leading dan trailing `'xyz'` dihapus, tetapi kejadian yang internal di dalam string tidak dihapus.

```
select 'xyzaxyzbxyzcxyz' as untrim,
btrim('xyzaxyzbxyzcxyz', 'xyz') as trim;

     untrim      |   trim
-----------------+-----------
 xyzaxyzbxyzcxyz | axyzbxyzc
```

Contoh berikut menghapus bagian depan dan belakang dari string `'setuphistorycassettes'` yang cocok dengan salah satu karakter dalam daftar trim_chars. `'tes'` Apa pun `te`,, atau `s` yang terjadi sebelum karakter lain yang tidak ada dalam daftar trim_chars di awal atau akhir string input dihapus.

```
SELECT btrim('setuphistorycassettes', 'tes');

     btrim
-----------------
 uphistoryca
```

Fungsi CHAR_LENGTH

Sinonim dari fungsi LEN.

Lihat Fungsi LEN.

Fungsi CHARACTER_LENGTH

Sinonim dari fungsi LEN.

Lihat Fungsi LEN.

Fungsi CHARINDEX

Mengembalikan lokasi substring tertentu dalam string.

Lihat Fungsi POSISI dan fungsi STRPOS untuk fungsi serupa.

Sintaks

```
CHARINDEX( substring, string )
```

Argumen

substring

Substring untuk mencari dalam string.

tali

String atau kolom yang akan dicari.

Jenis pengembalian

Fungsi CHARINDEX mengembalikan bilangan bulat yang sesuai dengan posisi substring (berbasis satu, bukan berbasis nol). Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal.

Catatan penggunaan

CHARINDEX mengembalikan 0 jika substring tidak ditemukan dalam: `string`

```
select charindex('dog', 'fish');

charindex
----------
0
(1 row)
```

Contoh-contoh

Contoh berikut menunjukkan posisi string `fish` dalam kata`dogfish`:

```
select charindex('fish', 'dogfish');
```

```
 charindex
----------
         4
(1 row)
```

Contoh berikut mengembalikan jumlah transaksi penjualan dengan KOMISI lebih dari 999.00 dari tabel PENJUALAN:

```
select distinct charindex('.', commission), count (charindex('.', commission))
from sales where charindex('.', commission) > 4 group by charindex('.', commission)
order by 1,2;

charindex | count
----------+-------
 5        |    629
(1 row)
```

Fungsi CONCAT

Fungsi CONCAT menggabungkan dua ekspresi dan mengembalikan ekspresi yang dihasilkan. Untuk menggabungkan lebih dari dua ekspresi, gunakan fungsi CONCAT bersarang. Operator penggabungan (| |) antara dua ekspresi menghasilkan hasil yang sama dengan fungsi CONCAT.

> **Note**
>
> Untuk kedua fungsi CONCAT dan operator penggabungan, jika salah satu atau kedua ekspresi adalah nol, hasil penggabungan adalah nol.

Sintaks

```
CONCAT ( expression1, expression2 )
```

Argumen

ekspresi1, ekspresi2

Kedua argumen dapat berupa string karakter fixed-length, string karakter panjang variabel, ekspresi biner, atau ekspresi yang mengevaluasi salah satu input ini.

Jenis pengembalian

CONCAT mengembalikan ekspresi. Tipe data ekspresi adalah tipe yang sama dengan argumen masukan.

Jika ekspresi input dari jenis yang berbeda, AWS Clean Rooms mencoba untuk secara implisit mengetik cast salah satu ekspresi. Jika nilai tidak dapat dilemparkan, kesalahan dikembalikan.

Contoh-contoh

Contoh berikut menggabungkan dua literal karakter:

```
select concat('December 25, ', '2008');

concat
-------------------
December 25, 2008
(1 row)
```

Kueri berikut, menggunakan || operator bukan CONCAT, menghasilkan hasil yang sama:

```
select 'December 25, '||'2008';

concat
-------------------
December 25, 2008
(1 row)
```

Contoh berikut menggunakan dua fungsi CONCAT untuk menggabungkan tiga string karakter:

```
select concat('Thursday, ', concat('December 25, ', '2008'));

concat
----------------------------
Thursday, December 25, 2008
(1 row)
```

Untuk menggabungkan kolom yang mungkin berisi nol, gunakan. Fungsi NVL dan COALESCE
Contoh berikut menggunakan NVL untuk mengembalikan 0 setiap kali NULL ditemui.

```
select concat(venuename, concat(' seats ', nvl(venueseats, 0))) as seating
from venue where venuestate = 'NV' or venuestate = 'NC'
```

```
order by 1
limit 5;

seating
-----------------------------------
Ballys Hotel seats 0
Bank of America Stadium seats 73298
Bellagio Hotel seats 0
Caesars Palace seats 0
Harrahs Hotel seats 0
(5 rows)
```

Kueri berikut menggabungkan nilai CITY dan STATE dari tabel VENUE:

```
select concat(venuecity, venuestate)
from venue
where venueseats > 75000
order by venueseats;

concat
-------------------
DenverCO
Kansas CityMO
East RutherfordNJ
LandoverMD
(4 rows)
```

Kueri berikut menggunakan fungsi CONCAT bersarang. Kueri menggabungkan nilai CITY dan STATE dari tabel VENUE tetapi membatasi string yang dihasilkan dengan koma dan spasi:

```
select concat(concat(venuecity,', '),venuestate)
from venue
where venueseats > 75000
order by venueseats;

concat
---------------------
Denver, CO
Kansas City, MO
East Rutherford, NJ
Landover, MD
(4 rows)
```

Fungsi KIRI dan KANAN

Fungsi-fungsi ini mengembalikan jumlah karakter paling kiri atau paling kanan yang ditentukan dari string karakter.

Jumlahnya didasarkan pada jumlah karakter, bukan byte, sehingga karakter multibyte dihitung sebagai karakter tunggal.

Sintaks

```
LEFT ( string,  integer )

RIGHT ( string,  integer )
```

Argumen

tali

String karakter apa pun atau ekspresi apa pun yang mengevaluasi string karakter.

bilangan bulat

Integer positif.

Jenis pengembalian

KIRI dan KANAN mengembalikan string VARCHAR.

Contoh

Contoh berikut mengembalikan 5 karakter paling kiri dan paling kanan 5 dari nama acara yang memiliki ID antara 1000 dan 1005:

```
select eventid, eventname,
left(eventname,5) as left_5,
right(eventname,5) as right_5
from event
where eventid between 1000 and 1005
order by 1;
```

```
eventid |   eventname    | left_5 | right_5
--------+----------------+--------+---------
   1000 | Gypsy          | Gypsy  | Gypsy
   1001 | Chicago        | Chica  | icago
   1002 | The King and I | The K  | and I
   1003 | Pal Joey       | Pal J  |  Joey
   1004 | Grease         | Greas  | rease
   1005 | Chicago        | Chica  | icago
(6 rows)
```

Fungsi LEN

Mengembalikan panjang string yang ditentukan sebagai jumlah karakter.

Sintaks

LEN adalah sinonim dariFungsi LENGTH,, Fungsi CHAR_LENGTHFungsi CHARACTER_LENGTH, dan. Fungsi TEXTLEN

```
LEN(expression)
```

Pendapat

ekspresi

Parameter input adalah CHAR atau VARCHAR atau alias dari salah satu jenis input yang valid.

Jenis pengembalian

Fungsi LEN mengembalikan integer yang menunjukkan jumlah karakter dalam string input.

Jika string input adalah string karakter, fungsi LEN mengembalikan jumlah aktual karakter dalam string multi-byte, bukan jumlah byte. Misalnya, kolom VARCHAR (12) diperlukan untuk menyimpan tiga karakter Mandarin empat byte. Fungsi LEN akan mengembalikan 3 untuk string yang sama.

Catatan penggunaan

Perhitungan panjang tidak menghitung spasi tambahan untuk string karakter dengan panjang tetap tetapi menghitungnya untuk string panjang variabel.

Contoh

Contoh berikut mengembalikan jumlah byte dan jumlah karakter dalam string`français`.

```
select octet_length('français'),
len('français');

octet_length  | len
--------------+-----
           9  |   8
```

Contoh berikut mengembalikan jumlah karakter dalam string tanpa spasi tambahan dan `cat cat` dengan tiga spasi tambahan:

```
select len('cat'), len('cat   ');
 len | len
-----+-----
   3 |   6
```

Contoh berikut mengembalikan sepuluh entri VENUENAME terpanjang dalam tabel VENUE:

```
select venuename, len(venuename)
from venue
order by 2 desc, 1
limit 10;

venuename                             | len
--------------------------------------+-----
Saratoga Springs Performing Arts Center |  39
Lincoln Center for the Performing Arts  |  38
Nassau Veterans Memorial Coliseum       |  33
Jacksonville Municipal Stadium          |  30
Rangers BallPark in Arlington           |  29
University of Phoenix Stadium           |  29
Circle in the Square Theatre            |  28
Hubert H. Humphrey Metrodome            |  28
Oriole Park at Camden Yards             |  27
Dick's Sporting Goods Park              |  26
```

Fungsi LENGTH

Sinonim dari fungsi LEN.

Lihat Fungsi LEN.

Fungsi LOWER

Mengkonversi string ke huruf kecil. LOWER mendukung karakter multibyte UTF-8, hingga maksimal empat byte per karakter.

Sintaks

```
LOWER(string)
```

Pendapat

tali

Parameter input adalah string VARCHAR (atau tipe data lainnya, seperti CHAR, yang dapat secara implisit dikonversi ke VARCHAR).

Jenis pengembalian

Fungsi LOWER mengembalikan string karakter yang merupakan tipe data yang sama dengan string input.

Contoh-contoh

Contoh berikut mengonversi bidang CATNAME menjadi huruf kecil:

```
select catname, lower(catname) from category order by 1,2;

 catname  |   lower
----------+-----------
Classical | classical
Jazz      | jazz
MLB       | mlb
MLS       | mls
Musicals  | musicals
NBA       | nba
NFL       | nfl
NHL       | nhl
Opera     | opera
Plays     | plays
```

```
Pop       | pop
(11 rows)
```

Fungsi LPAD dan RPAD

Fungsi-fungsi ini menambahkan atau menambahkan karakter ke string, berdasarkan panjang tertentu.

Sintaks

```
LPAD (string1, length, [ string2 ])
```

```
RPAD (string1, length, [ string2 ])
```

Argumen

senar1

String karakter atau ekspresi yang mengevaluasi string karakter, seperti nama kolom karakter.

panjang

Sebuah integer yang mendefinisikan panjang hasil dari fungsi. Panjang string didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. Jika string1 lebih panjang dari panjang yang ditentukan, itu terpotong (di sebelah kanan). Jika panjang adalah angka negatif, hasil dari fungsi adalah string kosong.

senar2

Satu atau lebih karakter yang ditambahkan atau ditambahkan ke string1. Argumen ini opsional; jika tidak ditentukan, spasi digunakan.

Jenis pengembalian

Fungsi-fungsi ini mengembalikan tipe data VARCHAR.

Contoh-contoh

Potong satu set nama acara tertentu menjadi 20 karakter dan tambahkan nama yang lebih pendek dengan spasi:

```
select lpad(eventname,20) from event
where eventid between 1 and 5 order by 1;

 lpad
--------------------
              Salome
        Il Trovatore
       Boris Godunov
     Gotterdammerung
La Cenerentola (Cind
(5 rows)
```

Potong set nama acara yang sama menjadi 20 karakter tetapi tambahkan nama yang lebih pendek dengan. `0123456789`

```
select rpad(eventname,20,'0123456789') from event
where eventid between 1 and 5 order by 1;

 rpad
--------------------
Boris Godunov0123456
Gotterdammerung01234
Il Trovatore01234567
La Cenerentola (Cind
Salome01234567890123
(5 rows)
```

Fungsi LTRIM

Memangkas karakter dari awal string. Menghapus string terpanjang yang hanya berisi karakter dalam daftar karakter trim. Pemangkasan selesai ketika karakter trim tidak muncul di string input.

Sintaks

```
LTRIM( string [, trim_chars] )
```

Argumen

tali

Sebuah kolom string, ekspresi, atau string literal yang akan dipangkas.

trim_chars

Sebuah kolom string, ekspresi, atau string literal yang mewakili karakter yang akan dipangkas dari awal string. Jika tidak ditentukan, spasi digunakan sebagai karakter trim.

Jenis pengembalian

Fungsi LTRIM mengembalikan string karakter yang merupakan tipe data yang sama dengan string input (CHAR atau VARCHAR).

Contoh-contoh

Contoh berikut memangkas tahun dari `listime` kolom. Karakter trim dalam string literal `'2008-'` menunjukkan karakter yang akan dipangkas dari kiri. Jika Anda menggunakan karakter trim`'028-'`, Anda mencapai hasil yang sama.

```
select listid, listtime, ltrim(listtime, '2008-')
from listing
order by 1, 2, 3
limit 10;

listid |      listtime       |     ltrim
-------+---------------------+---------------
     1 | 2008-01-24 06:43:29 | 1-24 06:43:29
     2 | 2008-03-05 12:25:29 | 3-05 12:25:29
     3 | 2008-11-01 07:35:33 | 11-01 07:35:33
     4 | 2008-05-24 01:18:37 | 5-24 01:18:37
     5 | 2008-05-17 02:29:11 | 5-17 02:29:11
     6 | 2008-08-15 02:08:13 | 15 02:08:13
     7 | 2008-11-15 09:38:15 | 11-15 09:38:15
     8 | 2008-11-09 05:07:30 | 11-09 05:07:30
     9 | 2008-09-09 08:03:36 | 9-09 08:03:36
    10 | 2008-06-17 09:44:54 | 6-17 09:44:54
```

LTRIM menghapus salah satu karakter di trim_chars ketika mereka muncul di awal string. Contoh berikut memangkas karakter 'C', 'D', dan 'G' ketika mereka muncul di awal VENUENAME, yang merupakan kolom VARCHAR.

```
select venueid, venuename, ltrim(venuename, 'CDG')
from venue
where venuename like '%Park'
```

```
order by 2
limit 7;

venueid | venuename                  | btrim
--------+----------------------------+-------------------------
    121 | ATT Park                   | ATT Park
    109 | Citizens Bank Park         | itizens Bank Park
    102 | Comerica Park              | omerica Park
      9 | Dick's Sporting Goods Park | ick's Sporting Goods Park
     97 | Fenway Park                | Fenway Park
    112 | Great American Ball Park   | reat American Ball Park
    114 | Miller Park                | Miller Park
```

Contoh berikut menggunakan karakter trim 2 yang diambil dari `venueid` kolom.

```
select ltrim('2008-01-24 06:43:29', venueid)
from venue where venueid=2;

ltrim
------------------
008-01-24 06:43:29
```

Contoh berikut tidak memangkas karakter apa pun 2 karena a ditemukan sebelum karakter `'0'` trim.

```
select ltrim('2008-01-24 06:43:29', '0');

ltrim
-------------------
2008-01-24 06:43:29
```

Contoh berikut menggunakan karakter trim spasi default dan memangkas dua spasi dari awal string.

```
select ltrim('  2008-01-24 06:43:29');

ltrim
-------------------
2008-01-24 06:43:29
```

Fungsi POSISI

Mengembalikan lokasi substring tertentu dalam string.

Lihat Fungsi CHARINDEX dan fungsi STRPOS untuk fungsi serupa.

Sintaks

```
POSITION(substring IN string )
```

Argumen

substring

Substring untuk mencari dalam string.

tali

String atau kolom yang akan dicari.

Jenis pengembalian

Fungsi POSITION mengembalikan bilangan bulat yang sesuai dengan posisi substring (berbasis satu, bukan berbasis nol). Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal.

Catatan penggunaan

POSITION mengembalikan 0 jika substring tidak ditemukan dalam string:

```
select position('dog' in 'fish');

position
----------
 0
(1 row)
```

Contoh-contoh

Contoh berikut menunjukkan posisi string `fish` dalam kata`dogfish`:

```
select position('fish' in 'dogfish');

position
----------
```

```
 4
(1 row)
```

Contoh berikut mengembalikan jumlah transaksi penjualan dengan KOMISI lebih dari 999.00 dari tabel PENJUALAN:

```
select distinct position('.' in commission), count (position('.' in commission))
from sales where position('.' in commission) > 4 group by position('.' in commission)
order by 1,2;

position | count
---------+-------
       5 |    629
(1 row)
```

Fungsi REGEXP_COUNT

Mencari string untuk pola ekspresi reguler dan mengembalikan integer yang menunjukkan berapa kali pola terjadi dalam string. Jika tidak ada kecocokan yang ditemukan, maka fungsi mengembalikan 0.

Sintaks

```
REGEXP_COUNT ( source_string, pattern [, position [, parameters ] ] )
```

Argumen

source_string

Ekspresi string, seperti nama kolom, yang akan dicari.

pola

Sebuah string literal yang mewakili pola ekspresi reguler.

posisi

Sebuah integer positif yang menunjukkan posisi dalam source_string untuk mulai mencari. Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multibyte dihitung sebagai karakter tunggal. Default-nya adalah 1. Jika posisi kurang dari 1, pencarian dimulai pada karakter pertama source_string. Jika posisi lebih besar dari jumlah karakter di source_string, hasilnya adalah 0.

parameter

Satu atau lebih string literal yang menunjukkan bagaimana fungsi cocok dengan pola. Nilai yang mungkin adalah sebagai berikut:

- c — Lakukan pencocokan peka huruf besar/kecil. Defaultnya adalah menggunakan pencocokan peka huruf besar/kecil.
- i — Lakukan pencocokan case-insensitive.
- p — Menafsirkan pola dengan dialek Perl Compatible Regular Expression (PCRE).

Jenis pengembalian

Bilangan Bulat

Contoh

Contoh berikut menghitung berapa kali urutan tiga huruf terjadi.

```
SELECT regexp_count('abcdefghijklmnopqrstuvwxyz', '[a-z]{3}');

 regexp_count
 --------------
            8
```

Contoh berikut menghitung berapa kali nama domain tingkat atas adalah salah satu atau`org`. `edu`

```
SELECT email, regexp_count(email,'@[^.]*\\.(org|edu)')FROM users
ORDER BY userid LIMIT 4;

                     email                      | regexp_count
------------------------------------------------+--------------
 Etiam.laoreet.libero@sodalesMaurisblandit.edu |            1
 Suspendisse.tristique@nonnisiAenean.edu       |            1
 amet.faucibus.ut@condimentumegetvolutpat.ca   |            0
 sed@lacusUtnec.ca                             |            0
```

Contoh berikut menghitung kemunculan string, menggunakan pencocokan `FOX` case-insensitive.

```
SELECT regexp_count('the fox', 'FOX', 1, 'i');
```

```
 regexp_count
 --------------
            1
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang mengandung setidaknya satu angka dan satu huruf kecil. Ini menggunakan ?= operator, yang memiliki konotasi pandangan ke depan tertentu di PCRE. Contoh ini menghitung jumlah kemunculan kata-kata tersebut, dengan pencocokan peka huruf besar/kecil.

```
SELECT regexp_count('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 'p');

 regexp_count
 --------------
            2
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang mengandung setidaknya satu angka dan satu huruf kecil. Ini menggunakan ?= operator, yang memiliki konotasi khusus di PCRE. Contoh ini menghitung jumlah kemunculan kata-kata tersebut, tetapi berbeda dari contoh sebelumnya karena menggunakan pencocokan case-insensitive.

```
SELECT regexp_count('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 'ip');

 regexp_count
 --------------
            3
```

Fungsi REGEXP_INSTR

Mencari string untuk pola ekspresi reguler dan mengembalikan integer yang menunjukkan posisi awal atau posisi akhir dari substring yang cocok. Jika tidak ada kecocokan yang ditemukan, maka fungsi mengembalikan 0. REGEXP_INSTR mirip dengan fungsi POSITION, tetapi memungkinkan Anda mencari string untuk pola ekspresi reguler.

Sintaks

```
REGEXP_INSTR ( source_string, pattern [, position [, occurrence] [, option
 [, parameters ] ] ] ] )
```

Argumen

source_string

Ekspresi string, seperti nama kolom, yang akan dicari.

pola

Sebuah string literal yang mewakili pola ekspresi reguler.

posisi

Sebuah integer positif yang menunjukkan posisi dalam source_string untuk mulai mencari. Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multibyte dihitung sebagai karakter tunggal. Default-nya adalah 1. Jika posisi kurang dari 1, pencarian dimulai pada karakter pertama source_string. Jika posisi lebih besar dari jumlah karakter di source_string, hasilnya adalah 0.

kejadian

Sebuah bilangan bulat positif yang menunjukkan kemunculan pola yang akan digunakan. REGEXP_INSTR melewatkan kejadian pertama -1 pertandingan. Default-nya adalah 1. Jika kejadian kurang dari 1 atau lebih besar dari jumlah karakter di source_string, pencarian diabaikan dan hasilnya adalah 0.

pilihan

Nilai yang menunjukkan apakah akan mengembalikan posisi karakter pertama pertandingan (`0`) atau posisi karakter pertama setelah akhir pertandingan (1). Nilai bukan nol sama dengan 1. Nilai default-nya adalah 0.

parameter

Satu atau lebih string literal yang menunjukkan bagaimana fungsi cocok dengan pola. Nilai yang mungkin adalah sebagai berikut:

- c — Lakukan pencocokan peka huruf besar/kecil. Defaultnya adalah menggunakan pencocokan peka huruf besar/kecil.
- i — Lakukan pencocokan case-insensitive.
- e — Ekstrak substring menggunakan subexpression.

  Jika pola menyertakan subexpression, REGEXP_INSTR cocok dengan substring menggunakan subexpression pertama dalam pola. REGEXP_INSTR hanya mempertimbangkan

subexpression pertama; subexpressions tambahan diabaikan. Jika pola tidak memiliki subexpression, REGEXP_INSTR mengabaikan parameter 'e'.

- p — Menafsirkan pola dengan dialek Perl Compatible Regular Expression (PCRE).

Jenis pengembalian

Bilangan Bulat

Contoh

Contoh berikut mencari @ karakter yang memulai nama domain dan mengembalikan posisi awal kecocokan pertama.

```
SELECT email, regexp_instr(email, '@[^.]*')
FROM users
ORDER BY userid LIMIT 4;

                     email                      | regexp_instr
------------------------------------------------+--------------
 Etiam.laoreet.libero@example.com |           21
 Suspendisse.tristique@nonnisiAenean.edu       |           22
 amet.faucibus.ut@condimentumegetvolutpat.ca   |           17
 sed@lacusUtnec.ca                             |            4
```

Contoh berikut mencari varian kata `Center` dan mengembalikan posisi awal kecocokan pertama.

```
SELECT venuename, regexp_instr(venuename,'[cC]ent(er|re)$')
FROM venue
WHERE regexp_instr(venuename,'[cC]ent(er|re)$') > 0
ORDER BY venueid LIMIT 4;

       venuename       | regexp_instr
-----------------------+--------------
 The Home Depot Center |           16
 Izod Center           |            6
 Wachovia Center       |           10
 Air Canada Centre     |           12
```

Contoh berikut menemukan posisi awal dari kemunculan pertama stringF0X, menggunakan logika pencocokan case-insensitive.

```
SELECT regexp_instr('the fox', 'FOX', 1, 1, 0, 'i');

 regexp_instr
 --------------
            5
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang mengandung setidaknya satu angka dan satu huruf kecil. Ini menggunakan ?= operator, yang memiliki konotasi pandangan ke depan tertentu di PCRE. Contoh ini menemukan posisi awal dari kata kedua tersebut.

```
SELECT regexp_instr('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 0, 'p');

 regexp_instr
 --------------
           21
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang mengandung setidaknya satu angka dan satu huruf kecil. Ini menggunakan ?= operator, yang memiliki konotasi pandangan ke depan tertentu di PCRE. Contoh ini menemukan posisi awal dari kata kedua tersebut, tetapi berbeda dari contoh sebelumnya karena menggunakan pencocokan case-insensitive.

```
SELECT regexp_instr('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 0, 'ip');

 regexp_instr
 --------------
           15
```

Fungsi REGEXP_REPLACE

Mencari string untuk pola ekspresi reguler dan menggantikan setiap kemunculan pola dengan string yang ditentukan. REGEXP_REPLACE mirip dengan GANTI fungsi, tetapi memungkinkan Anda mencari string untuk pola ekspresi reguler.

REGEXP_REPLACE mirip dengan FUNGSI TRANSLATE dan GANTI fungsi, kecuali bahwa TRANSLATE membuat beberapa substitusi karakter tunggal dan REPLACE menggantikan satu

seluruh string dengan string lain, sementara REGEXP_REPLACE memungkinkan Anda mencari string untuk pola ekspresi reguler.

Sintaks

```
REGEXP_REPLACE ( source_string, pattern [, replace_string [ , position [, parameters
 ] ] ] )
```

Argumen

source_string

Ekspresi string, seperti nama kolom, yang akan dicari.

pola

Sebuah string literal yang mewakili pola ekspresi reguler.

replace_string

Ekspresi string, seperti nama kolom, yang akan menggantikan setiap kemunculan pola. Defaultnya adalah string kosong (“”).

posisi

Sebuah integer positif yang menunjukkan posisi dalam source_string untuk mulai mencari. Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multibyte dihitung sebagai karakter tunggal. Default-nya adalah 1. Jika posisi kurang dari 1, pencarian dimulai pada karakter pertama source_string. Jika posisi lebih besar dari jumlah karakter di source_string, hasilnya adalah source_string.

parameter

Satu atau lebih string literal yang menunjukkan bagaimana fungsi cocok dengan pola. Nilai yang mungkin adalah sebagai berikut:

- c — Lakukan pencocokan peka huruf besar/kecil. Defaultnya adalah menggunakan pencocokan peka huruf besar/kecil.
- i — Lakukan pencocokan case-insensitive.
- p — Menafsirkan pola dengan dialek Perl Compatible Regular Expression (PCRE).

Jenis pengembalian

VARCHAR

Jika salah satu pola atau replace_string adalah NULL, pengembaliannya adalah NULL.

Contoh

Contoh berikut menghapus @ dan nama domain dari alamat email.

```
SELECT email, regexp_replace(email, '@.*\\.(org|gov|com|edu|ca)$')
FROM users
ORDER BY userid LIMIT 4;

              email                                | regexp_replace
---------------------------------------------------+----------------
 Etiam.laoreet.libero@sodalesMaurisblandit.edu | Etiam.laoreet.libero
 Suspendisse.tristique@nonnisiAenean.edu       | Suspendisse.tristique
 amet.faucibus.ut@condimentumegetvolutpat.ca   | amet.faucibus.ut
 sed@lacusUtnec.ca                             | sed
```

Contoh berikut menggantikan nama domain alamat email dengan nilai ini:`internal.company.com`.

```
SELECT email, regexp_replace(email, '@.*\\.[[:alpha:]]{2,3}',
'@internal.company.com') FROM users
ORDER BY userid LIMIT 4;

                     email                     |               regexp_replace
-----------------------------------------------
+-----------------------------------------
 Etiam.laoreet.libero@sodalesMaurisblandit.edu |
 Etiam.laoreet.libero@internal.company.com
 Suspendisse.tristique@nonnisiAenean.edu       |
 Suspendisse.tristique@internal.company.com
 amet.faucibus.ut@condimentumegetvolutpat.ca   | amet.faucibus.ut@internal.company.com
 sed@lacusUtnec.ca                             | sed@internal.company.com
```

Contoh berikut menggantikan semua kemunculan string FOX dalam nilai, menggunakan pencocokan `quick brown fox` case-insensitive.

```
SELECT regexp_replace('the fox', 'FOX', 'quick brown fox', 1, 'i');
```

```
   regexp_replace
--------------------
 the quick brown fox
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang mengandung setidaknya satu angka dan satu huruf kecil. Ini menggunakan ?= operator, yang memiliki konotasi pandangan ke depan tertentu di PCRE. Contoh ini menggantikan setiap kemunculan kata seperti itu dengan nilainya`[hidden]`.

```
SELECT regexp_replace('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 '[hidden]', 1, 'p');

        regexp_replace
------------------------------
 [hidden] plain A1234 [hidden]
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang mengandung setidaknya satu angka dan satu huruf kecil. Ini menggunakan ?= operator, yang memiliki konotasi pandangan ke depan tertentu di PCRE. Contoh ini menggantikan setiap kemunculan kata seperti itu dengan nilai`[hidden]`, tetapi berbeda dari contoh sebelumnya karena menggunakan pencocokan case-insensitive.

```
SELECT regexp_replace('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 '[hidden]', 1, 'ip');

          regexp_replace
---------------------------------
 [hidden] plain [hidden] [hidden]
```

Fungsi REGEXP_SUBSTR

Mengembalikan karakter dari string dengan mencarinya untuk pola ekspresi reguler. REGEXP_SUBSTR mirip dengan Fungsi SUBSTRING fungsinya, tetapi memungkinkan Anda mencari string untuk pola ekspresi reguler. Jika fungsi tidak dapat mencocokkan ekspresi reguler dengan karakter apa pun dalam string, ia mengembalikan string kosong.

Sintaks

```
REGEXP_SUBSTR ( source_string, pattern [, position [, occurrence [, parameters ] ] ] )
```

Argumen

source_string

Ekspresi string yang akan dicari.

pola

Sebuah string literal yang mewakili pola ekspresi reguler.

posisi

Sebuah integer positif yang menunjukkan posisi dalam source_string untuk mulai mencari. Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. Default-nya adalah 1. Jika posisi kurang dari 1, pencarian dimulai pada karakter pertama source_string. Jika posisi lebih besar dari jumlah karakter di source_string, hasilnya adalah string kosong (“”).

kejadian

Sebuah bilangan bulat positif yang menunjukkan kemunculan pola yang akan digunakan. REGEXP_SUBSTR melewatkan kejadian pertama -1 pertandingan. Default-nya adalah 1. Jika kejadian kurang dari 1 atau lebih besar dari jumlah karakter di source_string, pencarian diabaikan dan hasilnya adalah NULL.

parameter

Satu atau lebih string literal yang menunjukkan bagaimana fungsi cocok dengan pola. Nilai yang mungkin adalah sebagai berikut:

- c — Lakukan pencocokan peka huruf besar/kecil. Defaultnya adalah menggunakan pencocokan peka huruf besar/kecil.
- i — Lakukan pencocokan case-insensitive.
- e — Ekstrak substring menggunakan subexpression.

  Jika pola menyertakan subexpression, REGEXP_SUBSTR cocok dengan substring menggunakan subexpression pertama dalam pola. Sebuah subexpression adalah ekspresi dalam pola yang dikurung dengan tanda kurung. Misalnya, untuk pola `'This is a (\\w+)'` cocok ekspresi pertama dengan string `'This is a '` diikuti oleh sebuah kata. Alih-alih mengembalikan pola, REGEXP_SUBSTR dengan e parameter hanya mengembalikan string di dalam subexpression.

REGEXP_SUBSTR hanya mempertimbangkan subexpression pertama; subexpressions tambahan diabaikan. Jika pola tidak memiliki subexpression, REGEXP_SUBSTR mengabaikan parameter 'e'.

- p — Menafsirkan pola dengan dialek Perl Compatible Regular Expression (PCRE).

Jenis pengembalian

VARCHAR

Contoh

Contoh berikut mengembalikan bagian dari alamat email antara karakter @ dan ekstensi domain.

```
SELECT email, regexp_substr(email,'@[^.]*')
FROM users
ORDER BY userid LIMIT 4;

                     email                      |      regexp_substr
------------------------------------------------+--------------------------
 Etiam.laoreet.libero@sodalesMaurisblandit.edu | @sodalesMaurisblandit
 Suspendisse.tristique@nonnisiAenean.edu       | @nonnisiAenean
 amet.faucibus.ut@condimentumegetvolutpat.ca   | @condimentumegetvolutpat
 sed@lacusUtnec.ca                             | @lacusUtnec
```

Contoh berikut mengembalikan bagian dari input yang sesuai dengan kejadian pertama stringF0X, menggunakan pencocokan case-insensitive.

```
SELECT regexp_substr('the fox', 'FOX', 1, 1, 'i');

 regexp_substr
---------------
 fox
```

Contoh berikut mengembalikan bagian pertama dari input yang dimulai dengan huruf kecil. Ini secara fungsional identik dengan pernyataan SELECT yang sama tanpa c parameter.

```
SELECT regexp_substr('THE SECRET CODE IS THE LOWERCASE PART OF 1931abc0EZ.', '[a-z]+',
 1, 1, 'c');
```

```
 regexp_substr
---------------
 abc
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang mengandung setidaknya satu angka dan satu huruf kecil. Ini menggunakan ?= operator, yang memiliki konotasi pandangan ke depan tertentu di PCRE. Contoh ini mengembalikan bagian dari input yang sesuai dengan kata kedua tersebut.

```
SELECT regexp_substr('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 'p');

 regexp_substr
---------------
 a1234
```

Contoh berikut menggunakan pola yang ditulis dalam dialek PCRE untuk menemukan kata-kata yang mengandung setidaknya satu angka dan satu huruf kecil. Ini menggunakan ?= operator, yang memiliki konotasi pandangan ke depan tertentu di PCRE. Contoh ini mengembalikan bagian input yang sesuai dengan kata kedua tersebut, tetapi berbeda dari contoh sebelumnya karena menggunakan pencocokan case-insensitive.

```
SELECT regexp_substr('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 'ip');

 regexp_substr
---------------
 A1234
```

Contoh berikut menggunakan subexpression untuk menemukan string kedua yang cocok dengan pola `'this is a (\\w+)'` menggunakan pencocokan case-insensitive. Ia mengembalikan subexpression di dalam tanda kurung.

```
select regexp_substr(
               'This is a cat, this is a dog. This is a mouse.',
               'this is a (\\w+)', 1, 2, 'ie');

 regexp_substr
---------------
 dog
```

Fungsi REPEAT

Mengulangi string jumlah yang ditentukan kali. Jika parameter input numerik, REPEAT memperlakukannya sebagai string.

Sinonim untuk. Fungsi REPLICATE

Sintaks

```
REPEAT(string, integer)
```

Argumen

tali

Parameter input pertama adalah string yang akan diulang.

bilangan bulat

Parameter kedua adalah bilangan bulat yang menunjukkan berapa kali untuk mengulangi string.

Jenis pengembalian

Fungsi REPEAT mengembalikan string.

Contoh-contoh

Contoh berikut mengulangi nilai kolom CATID dalam tabel CATEGORY tiga kali:

```
select catid, repeat(catid,3)
from category
order by 1,2;

 catid | repeat
-------+--------
     1 | 111
     2 | 222
     3 | 333
     4 | 444
     5 | 555
     6 | 666
```

```
     7 | 777
     8 | 888
     9 | 999
    10 | 101010
    11 | 111111
(11 rows)
```

GANTI fungsi

Menggantikan semua kemunculan satu set karakter dalam string yang ada dengan karakter tertentu lainnya.

REPLACE mirip dengan FUNGSI TRANSLATE danFungsi REGEXP_REPLACE, kecuali bahwa TRANSLATE membuat beberapa substitusi karakter tunggal dan REGEXP_REPLACE memungkinkan Anda mencari string untuk pola ekspresi reguler, sementara REPLACE mengganti satu seluruh string dengan string lain.

Sintaks

```
REPLACE(string1, old_chars, new_chars)
```

Argumen

tali

CHAR atau VARCHAR string yang akan dicari pencarian

old_chars

String CHAR atau VARCHAR untuk diganti.

new_chars

String CHAR atau VARCHAR baru menggantikan old_string.

Jenis pengembalian

VARCHAR

Jika old_chars atau new_chars adalah NULL, pengembaliannya adalah NULL.

Contoh-contoh

Contoh berikut mengkonversi string `Shows` ke `Theatre` dalam bidang CATGROUP:

```
select catid, catgroup,
replace(catgroup, 'Shows', 'Theatre')
from category
order by 1,2,3;

 catid | catgroup | replace
-------+----------+----------
     1 | Sports   | Sports
     2 | Sports   | Sports
     3 | Sports   | Sports
     4 | Sports   | Sports
     5 | Sports   | Sports
     6 | Shows    | Theatre
     7 | Shows    | Theatre
     8 | Shows    | Theatre
     9 | Concerts | Concerts
    10 | Concerts | Concerts
    11 | Concerts | Concerts
(11 rows)
```

Fungsi REPLICATE

Sinonim untuk fungsi REPEAT.

Lihat Fungsi REPEAT.

Fungsi REVERSE

Fungsi REVERSE beroperasi pada string dan mengembalikan karakter dalam urutan terbalik. Misalnya, `reverse('abcde')` mengembalikan `edcba`. Fungsi ini bekerja pada tipe data numerik dan tanggal serta tipe data karakter; Namun, dalam banyak kasus memiliki nilai praktis untuk string karakter.

Sintaks

```
REVERSE ( expression )
```

Pendapat

ekspresi

Ekspresi dengan karakter, tanggal, stempel waktu, atau tipe data numerik yang mewakili target pembalikan karakter. Semua ekspresi secara implisit dikonversi ke string karakter panjang variabel. Bagian belakang kosong dalam string karakter dengan lebar tetap diabaikan.

Jenis pengembalian

REVERSE mengembalikan VARCHAR.

Contoh-contoh

Pilih lima nama kota yang berbeda dan nama terbalik yang sesuai dari tabel USERS:

```
select distinct city as cityname, reverse(cityname)
from users order by city limit 5;

cityname | reverse
---------+----------
Aberdeen | needrebA
Abilene  | enelibA
Ada      | adA
Agat     | tagA
Agawam   | mawagA
(5 rows)
```

Pilih lima ID penjualan dan ID terbalik yang sesuai ditampilkan sebagai string karakter:

```
select salesid, reverse(salesid)::varchar
from sales order by salesid desc limit 5;

salesid | reverse
--------+---------
 172456 | 654271
 172455 | 554271
 172454 | 454271
 172453 | 354271
 172452 | 254271
(5 rows)
```

Fungsi RTRIM

Fungsi RTRIM memangkas satu set karakter tertentu dari akhir string. Menghapus string terpanjang yang hanya berisi karakter dalam daftar karakter trim. Pemangkasan selesai ketika karakter trim tidak muncul di string input.

Sintaks

```
RTRIM( string, trim_chars )
```

Argumen

tali

Sebuah kolom string, ekspresi, atau string literal yang akan dipangkas.

trim_chars

Sebuah kolom string, ekspresi, atau string literal yang mewakili karakter yang akan dipangkas dari akhir string. Jika tidak ditentukan, spasi digunakan sebagai karakter trim.

Jenis pengembalian

String yang merupakan tipe data yang sama dengan argumen string.

Contoh

Contoh berikut memangkas bagian depan dan belakang kosong dari string: `' abc '`

```
select '     abc    ' as untrim, rtrim('     abc    ') as trim;

untrim    | trim
----------+------
   abc    |    abc
```

Contoh berikut menghapus string trailing dari `'xyz'` string. `'xyzaxyzbxyzcxyz'` Kejadian tertinggal `'xyz'` dihapus, tetapi kejadian yang internal di dalam string tidak dihapus.

```
select 'xyzaxyzbxyzcxyz' as untrim,
rtrim('xyzaxyzbxyzcxyz', 'xyz') as trim;
```

```
     untrim      |   trim
-----------------+-----------
 xyzaxyzbxyzcxyz | xyzaxyzbxyzc
```

Contoh berikut menghapus bagian trailing dari string `'setuphistorycassettes'` yang cocok dengan salah satu karakter dalam daftar trim_chars. `'tes'` Apa pun `te`,, atau `s` yang terjadi sebelum karakter lain yang tidak ada dalam daftar trim_chars di akhir string input dihapus.

```
SELECT rtrim('setuphistorycassettes', 'tes');

     rtrim
-----------------
 setuphistoryca
```

Contoh berikut memangkas karakter 'Park' dari akhir VENUENAME di mana ada:

```
select venueid, venuename, rtrim(venuename, 'Park')
from venue
order by 1, 2, 3
limit 10;

venueid |         venuename          |          rtrim
--------+----------------------------+-------------------------
      1 | Toyota Park                | Toyota
      2 | Columbus Crew Stadium      | Columbus Crew Stadium
      3 | RFK Stadium                | RFK Stadium
      4 | CommunityAmerica Ballpark  | CommunityAmerica Ballp
      5 | Gillette Stadium           | Gillette Stadium
      6 | New York Giants Stadium    | New York Giants Stadium
      7 | BMO Field                  | BMO Field
      8 | The Home Depot Center      | The Home Depot Cente
      9 | Dick's Sporting Goods Park | Dick's Sporting Goods
     10 | Pizza Hut Park             | Pizza Hut
```

Perhatikan bahwa RTRIM menghapus salah satu karakter`P`,`a`,`r`, atau `k` ketika mereka muncul di akhir VENUENAME.

Fungsi SOUNDEX

Fungsi SOUNDEX mengembalikan nilai American Soundex yang terdiri dari huruf pertama diikuti oleh pengkodean 3 digit suara yang mewakili pengucapan bahasa Inggris dari string yang Anda tentukan.

Sintaks

```
SOUNDEX(string)
```

Argumen

tali

Anda menentukan string CHAR atau VARCHAR yang ingin Anda konversi ke nilai kode Soundex Amerika.

Jenis pengembalian

Fungsi SOUNDEX mengembalikan string VARCHAR (4) yang terdiri dari huruf kapital diikuti oleh pengkodean tiga digit dari suara yang mewakili pengucapan bahasa Inggris.

Catatan penggunaan

Fungsi SOUNDEX hanya mengkonversi huruf kecil alfabet bahasa Inggris dan huruf besar ASCII karakter, termasuk a—z dan A—Z. SOUNDEX mengabaikan karakter lain. SOUNDEX mengembalikan nilai Soundex tunggal untuk string beberapa kata yang dipisahkan oleh spasi.

```
select soundex('AWS Amazon');
```

```
 soundex
---------
 A252
```

SOUNDEX mengembalikan string kosong jika string input tidak mengandung huruf bahasa Inggris.

```
select soundex('+-*/%');
```

```
  soundex
---------

```

Contoh

Contoh berikut mengembalikan Soundex A525 untuk kata Amazon.

```
select soundex('Amazon');
```

```
 soundex
---------
 A525
```

Fungsi SPLIT_PART

Membagi string pada pembatas yang ditentukan dan mengembalikan bagian pada posisi yang ditentukan.

Sintaks

```
SPLIT_PART(string, delimiter, position)
```

Argumen

tali

Sebuah kolom string, ekspresi, atau string literal yang akan dibagi. String dapat berupa CHAR atau VARCHAR.

pembatas

String pembatas menunjukkan bagian dari string input.

Jika pembatas adalah literal, lampirkan dalam tanda kutip tunggal.

posisi

Posisi bagian string untuk kembali (menghitung dari 1). Harus bilangan bulat lebih besar dari 0. Jika posisi lebih besar dari jumlah bagian string, SPLIT_PART mengembalikan string kosong. Jika pembatas tidak ditemukan dalam string, maka nilai yang dikembalikan berisi isi dari bagian yang ditentukan, yang mungkin seluruh string atau nilai kosong.

Jenis pengembalian

Sebuah string CHAR atau VARCHAR, sama dengan parameter string.

Contoh-contoh

Contoh berikut membagi string literal menjadi beberapa bagian menggunakan $ pembatas dan mengembalikan bagian kedua.

```
select split_part('abc$def$ghi','$',2)

split_part
----------
def
```

Contoh berikut membagi string literal menjadi beberapa bagian menggunakan $ pembatas. Ia mengembalikan string kosong karena bagian 4 tidak ditemukan.

```
select split_part('abc$def$ghi','$',4)

split_part
----------

```

Contoh berikut membagi string literal menjadi beberapa bagian menggunakan # pembatas. Ia mengembalikan seluruh string, yang merupakan bagian pertama, karena pembatas tidak ditemukan.

```
select split_part('abc$def$ghi','#',1)

split_part
------------
abc$def$ghi
```

Contoh berikut membagi bidang timestamp LISTTIME menjadi komponen tahun, bulan, dan hari.

```
select listtime, split_part(listtime,'-',1) as year,
split_part(listtime,'-',2) as month,
split_part(split_part(listtime,'-',3),' ',1) as day
from listing limit 5;

      listtime       | year | month | day
---------------------+------+-------+------
 2008-03-05 12:25:29 | 2008 | 03    | 05
 2008-09-09 08:03:36 | 2008 | 09    | 09
 2008-09-26 05:43:12 | 2008 | 09    | 26
```

```
 2008-10-04 02:00:30 | 2008 | 10    | 04
 2008-01-06 08:33:11 | 2008 | 01    | 06
```

Contoh berikut memilih bidang timestamp LISTTIME dan membaginya pada `'-'` karakter untuk mendapatkan bulan (bagian kedua dari string LISTTIME), lalu menghitung jumlah entri untuk setiap bulan:

```
select split_part(listtime,'-',2) as month, count(*)
from listing
group by split_part(listtime,'-',2)
order by 1, 2;

 month | count
-------+-------
    01 | 18543
    02 | 16620
    03 | 17594
    04 | 16822
    05 | 17618
    06 | 17158
    07 | 17626
    08 | 17881
    09 | 17378
    10 | 17756
    11 | 12912
    12 | 4589
```

fungsi STRPOS

Mengembalikan posisi substring dalam string tertentu.

Lihat Fungsi CHARINDEX dan Fungsi POSISI untuk fungsi serupa.

Sintaks

```
STRPOS(string, substring )
```

Argumen

tali

Parameter input pertama adalah string yang akan dicari.

substring

Parameter kedua adalah substring untuk mencari di dalam string.

Jenis pengembalian

Fungsi STRPOS mengembalikan bilangan bulat yang sesuai dengan posisi substring (berbasis satu, bukan berbasis nol). Posisi didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal.

Catatan penggunaan

STRPOS mengembalikan 0 jika substring tidak ditemukan dalam string:

```
select strpos('dogfish', 'fist');
strpos
--------
0
(1 row)
```

Contoh-contoh

Contoh berikut menunjukkan posisi string `fish` dalam kata`dogfish`:

```
select strpos('dogfish', 'fish');
strpos
--------
4
(1 row)
```

Contoh berikut mengembalikan jumlah transaksi penjualan dengan KOMISI lebih dari 999.00 dari tabel PENJUALAN:

```
select distinct strpos(commission, '.'),
count (strpos(commission, '.'))
from sales
where strpos(commission, '.') > 4
group by strpos(commission, '.')
order by 1, 2;

strpos | count
```

```
-------+-------
 5     |    629
(1 row)
```

Fungsi SUBSTR

Sinonim dari fungsi SUBSTRING.

Lihat Fungsi SUBSTRING.

Fungsi SUBSTRING

Mengembalikan subset dari string berdasarkan posisi awal yang ditentukan.

Jika input adalah string karakter, posisi awal dan jumlah karakter yang diekstraksi didasarkan pada karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. Jika input adalah ekspresi biner, posisi awal dan substring yang diekstraksi didasarkan pada byte. Anda tidak dapat menentukan panjang negatif, tetapi Anda dapat menentukan posisi awal negatif.

Sintaks

```
SUBSTRING(character_string FROM start_position [ FOR number_characters ] )
```

```
SUBSTRING(character_string, start_position, number_characters )
```

```
SUBSTRING(binary_expression, start_byte, number_bytes )
```

```
SUBSTRING(binary_expression, start_byte )
```

Argumen

character_string

String yang akan dicari. Tipe data non-karakter diperlakukan seperti string.

start_position

Posisi dalam string untuk memulai ekstraksi, mulai dari 1. Start_position didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. Angka ini bisa negatif.

number_characters

Jumlah karakter untuk mengekstrak (panjang substring). Number_characters didasarkan pada jumlah karakter, bukan byte, sehingga karakter multi-byte dihitung sebagai karakter tunggal. Angka ini tidak bisa negatif.

start_byte

Posisi dalam ekspresi biner untuk memulai ekstraksi, mulai dari 1. Angka ini bisa negatif.

number_bytes

Jumlah byte untuk mengekstrak, yaitu, panjang substring. Angka ini tidak bisa negatif.

Jenis pengembalian

VARCHAR

Catatan penggunaan untuk string karakter

Contoh berikut mengembalikan string empat karakter yang dimulai dengan karakter keenam.

```
select substring('caterpillar',6,4);
substring
-----------
pill
(1 row)
```

Jika start_position + number_characters melebihi panjang string, SUBSTRING mengembalikan substring mulai dari start_position hingga akhir string. Sebagai contoh:

```
select substring('caterpillar',6,8);
substring
-----------
pillar
(1 row)
```

Jika negatif atau 0, fungsi SUBSTRING mengembalikan substring yang dimulai pada karakter pertama string dengan panjang `start_position` number_characters +-1. `start_position` Sebagai contoh:

```
select substring('caterpillar',-2,6);
```

```
substring
-----------
cat
(1 row)
```

Jika `start_position` + `number_characters` -1 kurang dari atau sama dengan nol, SUBSTRING mengembalikan string kosong. Sebagai contoh:

```
select substring('caterpillar',-5,4);
substring
-----------

(1 row)
```

Contoh-contoh

Contoh berikut mengembalikan bulan dari string LISTTIME dalam tabel LISTING:

```
select listid, listtime,
substring(listtime, 6, 2) as month
from listing
order by 1, 2, 3
limit 10;

 listid |      listtime       | month
--------+---------------------+-------
      1 | 2008-01-24 06:43:29 | 01
      2 | 2008-03-05 12:25:29 | 03
      3 | 2008-11-01 07:35:33 | 11
      4 | 2008-05-24 01:18:37 | 05
      5 | 2008-05-17 02:29:11 | 05
      6 | 2008-08-15 02:08:13 | 08
      7 | 2008-11-15 09:38:15 | 11
      8 | 2008-11-09 05:07:30 | 11
      9 | 2008-09-09 08:03:36 | 09
     10 | 2008-06-17 09:44:54 | 06
(10 rows)
```

Contoh berikut sama seperti di atas, tetapi menggunakan opsi FROM... FOR:

```
select listid, listtime,
```

```
substring(listtime from 6 for 2) as month
from listing
order by 1, 2, 3
limit 10;

 listid |      listtime       | month
--------+---------------------+-------
      1 | 2008-01-24 06:43:29 | 01
      2 | 2008-03-05 12:25:29 | 03
      3 | 2008-11-01 07:35:33 | 11
      4 | 2008-05-24 01:18:37 | 05
      5 | 2008-05-17 02:29:11 | 05
      6 | 2008-08-15 02:08:13 | 08
      7 | 2008-11-15 09:38:15 | 11
      8 | 2008-11-09 05:07:30 | 11
      9 | 2008-09-09 08:03:36 | 09
     10 | 2008-06-17 09:44:54 | 06
(10 rows)
```

Anda tidak dapat menggunakan SUBSTRING untuk mengekstrak awalan string yang mungkin berisi karakter multi-byte karena Anda perlu menentukan panjang string multi-byte berdasarkan jumlah byte, bukan jumlah karakter. Untuk mengekstrak segmen awal string berdasarkan panjang dalam byte, Anda dapat CAST string sebagai VARCHAR (byte_length) untuk memotong string, di mana byte_length adalah panjang yang diperlukan. Contoh berikut mengekstrak 5 byte pertama dari string`'Fourscore and seven'`.

```
select cast('Fourscore and seven' as varchar(5));

varchar
-------
Fours
```

Contoh berikut mengembalikan nama depan `Ana` yang muncul setelah spasi terakhir dalam string input`Silva, Ana`.

```
select reverse(substring(reverse('Silva, Ana'), 1, position(' ' IN reverse('Silva,
 Ana'))))

 reverse
-----------
 Ana
```

Fungsi TEXTLEN

Sinonim dari fungsi LEN.

Lihat Fungsi LEN.

FUNGSI TRANSLATE

Untuk ekspresi tertentu, menggantikan semua kemunculan karakter tertentu dengan pengganti tertentu. Karakter yang ada dipetakan ke karakter pengganti berdasarkan posisinya dalam argumen characters_to_replace dan characters_to_substitusi. Jika lebih banyak karakter ditentukan dalam argumen characters_to_replace daripada dalam argumen characters_to_substitusi, karakter tambahan dari argumen characters_to_replace dihilangkan dalam nilai pengembalian.

TRANSLATE mirip dengan GANTI fungsi danFungsi REGEXP_REPLACE, kecuali bahwa REPLACE mengganti satu seluruh string dengan string lain dan REGEXP_REPLACE memungkinkan Anda mencari string untuk pola ekspresi reguler, sementara TRANSLATE membuat beberapa substitusi karakter tunggal.

Jika ada argumen nol, pengembaliannya adalah NULL.

Sintaks

```
TRANSLATE ( expression, characters_to_replace, characters_to_substitute )
```

Argumen

ekspresi

Ekspresi yang akan diterjemahkan.

characters_to_replace

Sebuah string yang berisi karakter yang akan diganti.

characters_to_substitusi

Sebuah string yang berisi karakter untuk menggantikan.

Jenis pengembalian

VARCHAR

Contoh-contoh

Contoh berikut menggantikan beberapa karakter dalam string:

```
select translate('mint tea', 'inea', 'osin');

translate
-----------
most tin
```

Contoh berikut menggantikan tanda at (@) dengan periode untuk semua nilai dalam kolom:

```
select email, translate(email, '@', '.') as obfuscated_email
from users limit 10;

email                                          obfuscated_email
-------------------------------------------------------------------------------------------
Etiam.laoreet.libero@sodalesMaurisblandit.edu
 Etiam.laoreet.libero.sodalesMaurisblandit.edu
amet.faucibus.ut@condimentumegetvolutpat.ca
 amet.faucibus.ut.condimentumegetvolutpat.ca
turpis@accumsanlaoreet.org                 turpis.accumsanlaoreet.org
ullamcorper.nisl@Cras.edu                 ullamcorper.nisl.Cras.edu
arcu.Curabitur@senectusetnetus.com              arcu.Curabitur.senectusetnetus.com
ac@velit.ca                                  ac.velit.ca
Aliquam.vulputate.ullamcorper@amalesuada.org
 Aliquam.vulputate.ullamcorper.amalesuada.org
vel.est@velitegestas.edu                        vel.est.velitegestas.edu
dolor.nonummy@ipsumdolorsit.ca                  dolor.nonummy.ipsumdolorsit.ca
et@Nunclaoreet.ca                               et.Nunclaoreet.ca
```

Contoh berikut menggantikan spasi dengan garis bawah dan menghapus periode untuk semua nilai dalam kolom:

```
select city, translate(city, ' .', '_') from users
where city like 'Sain%' or city like 'St%'
group by city
order by city;

city            translate
--------------+------------------
Saint Albans     Saint_Albans
```

```
Saint Cloud      Saint_Cloud
Saint Joseph     Saint_Joseph
Saint Louis      Saint_Louis
Saint Paul       Saint_Paul
St. George       St_George
St. Marys        St_Marys
St. Petersburg   St_Petersburg
Stafford         Stafford
Stamford         Stamford
Stanton          Stanton
Starkville       Starkville
Statesboro       Statesboro
Staunton         Staunton
Steubenville     Steubenville
Stevens Point    Stevens_Point
Stillwater       Stillwater
Stockton         Stockton
Sturgis          Sturgis
```

Fungsi TRIM

Memangkas string dengan menghapus bagian depan dan belakang kosong atau dengan menghapus karakter utama dan belakang yang cocok dengan string tertentu opsional.

Sintaks

```
TRIM( [ BOTH ] [ trim_chars FROM ] string
```

Argumen

trim_chars

(Opsional) Karakter yang akan dipangkas dari string. Jika parameter ini dihilangkan, blanko dipangkas.

tali

Tali yang akan dipangkas.

Jenis pengembalian

Fungsi TRIM mengembalikan string VARCHAR atau CHAR. Jika Anda menggunakan fungsi TRIM dengan perintah SQL, secara AWS Clean Rooms implisit mengubah hasilnya ke VARCHAR. Jika Anda menggunakan fungsi TRIM dalam daftar SELECT untuk fungsi SQL, AWS Clean Rooms tidak secara implisit mengonversi hasilnya, dan Anda mungkin perlu melakukan konversi eksplisit untuk menghindari kesalahan ketidakcocokan tipe data. Lihat Fungsi CAST dan Fungsi CONVERT fungsi untuk informasi tentang konversi eksplisit.

Contoh

Contoh berikut memangkas bagian depan dan belakang kosong dari string: `'  abc  '`

```
select '     abc    ' as untrim, trim('     abc    ') as trim;

untrim    | trim
----------+------
   abc    | abc
```

Contoh berikut menghapus tanda kutip ganda yang mengelilingi string: `"dog"`

```
select trim('"' FROM '"dog"');

btrim
-------
dog
```

TRIM menghapus salah satu karakter di trim_chars ketika mereka muncul di awal string. Contoh berikut memangkas karakter 'C', 'D', dan 'G' ketika mereka muncul di awal VENUENAME, yang merupakan kolom VARCHAR.

```
select venueid, venuename, trim(venuename, 'CDG')
from venue
where venuename like '%Park'
order by 2
limit 7;

venueid | venuename                  | btrim
--------+----------------------------+-------------------------
    121 | ATT Park                   | ATT Park
```

```
    109 | Citizens Bank Park         | itizens Bank Park
    102 | Comerica Park              | omerica Park
      9 | Dick's Sporting Goods Park | ick's Sporting Goods Park
     97 | Fenway Park                | Fenway Park
    112 | Great American Ball Park   | reat American Ball Park
    114 | Miller Park                | Miller Park
```

Fungsi UPPER

Mengkonversi string ke huruf besar. UPPER mendukung karakter multibyte UTF-8, hingga maksimal empat byte per karakter.

Sintaks

```
UPPER(string)
```

Argumen

tali

Parameter input adalah string VARCHAR (atau tipe data lainnya, seperti CHAR, yang dapat secara implisit dikonversi ke VARCHAR).

Jenis pengembalian

Fungsi UPPER mengembalikan string karakter yang merupakan tipe data yang sama dengan string input.

Contoh-contoh

Contoh berikut mengkonversi bidang CATNAME ke huruf besar:

```
select catname, upper(catname) from category order by 1,2;

 catname  |   upper
----------+-----------
Classical | CLASSICAL
Jazz      | JAZZ
MLB       | MLB
```

```
MLS       | MLS
Musicals  | MUSICALS
NBA       | NBA
NFL       | NFL
NHL       | NHL
Opera     | OPERA
Plays     | PLAYS
Pop       | POP
(11 rows)
```

Fungsi informasi tipe SUPER

Bagian ini menjelaskan fungsi informasi untuk SQL untuk memperoleh informasi dinamis dari input tipe SUPER data yang didukung di. AWS Clean Rooms

Topik

Fungsi DECIMAL_PRECISION

Memeriksa ketepatan jumlah total digit desimal maksimum yang akan disimpan. Angka ini mencakup digit kiri dan kanan dari titik desimal. Kisaran presisi adalah dari 1 hingga 38, dengan default 38.

Sintaksis

```
DECIMAL_PRECISION(super_expression)
```

Pendapat

super_ekspresi

SUPEREkspresi atau kolom.

Jenis pengembalian

INTEGER

Contoh

Untuk menerapkan fungsi DECIMAL_PRECISION ke tabel t, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (3.14159);

SELECT DECIMAL_PRECISION(s) FROM t;

+-------------------+
| decimal_precision |
+-------------------+
|                 6 |
+-------------------+
```

Fungsi DECIMAL_SCALE

Memeriksa jumlah digit desimal yang akan disimpan di sebelah kanan titik desimal. Kisaran skala adalah dari 0 ke titik presisi, dengan default 0.

Sintaksis

```
DECIMAL_SCALE(super_expression)
```

Pendapat

super_ekspresi

SUPEREkspresi atau kolom.

Jenis pengembalian

INTEGER

Contoh

Untuk menerapkan fungsi DECIMAL_SCALE ke tabel t, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (3.14159);

SELECT DECIMAL_SCALE(s) FROM t;

+---------------+
| decimal_scale |
+---------------+
|             5 |
+---------------+
```

Fungsi IS_ARRAY

Memeriksa apakah variabel adalah array. Fungsi kembali `true` jika variabel adalah array. Fungsi ini juga mencakup array kosong. Jika tidak, fungsi kembali `false` untuk semua nilai lainnya, termasuk null.

Sintaksis

```
IS_ARRAY(super_expression)
```

Pendapat

super_ekspresi

SUPEREkspresi atau kolom.

Jenis pengembalian

`BOOLEAN`

Contoh

Untuk memeriksa `[1,2]` apakah array menggunakan fungsi IS_ARRAY, gunakan contoh berikut.

```
SELECT IS_ARRAY(JSON_PARSE('[1,2]'));

+----------+
| is_array |
+----------+
| true     |
+----------+
```

Fungsi IS_BIGINT

Memeriksa apakah suatu nilai adalah a`BIGINT`. Fungsi IS_BIGINT mengembalikan `true` jumlah skala 0 dalam rentang 64-bit. Jika tidak, fungsi kembali `false` untuk semua nilai lainnya, termasuk angka nol dan floating point.

Fungsi IS_BIGINT adalah superset dari IS_INTEGER.

Sintaksis

```
IS_BIGINT(super_expression)
```

Pendapat

super_ekspresi

`SUPER`Ekspresi atau kolom.

Jenis pengembalian

`BOOLEAN`

Contoh

Untuk memeriksa `5` apakah `BIGINT` menggunakan fungsi IS_BIGINT, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_BIGINT(s) FROM t;

+---+-----------+
| s | is_bigint |
+---+-----------+
| 5 | true      |
+---+-----------+
```

Fungsi IS_CHAR

Memeriksa apakah suatu nilai adalah a`CHAR`. Fungsi IS_CHAR mengembalikan `true` untuk string yang hanya memiliki karakter ASCII, karena tipe CHAR hanya dapat menyimpan karakter yang ada dalam format ASCII. Fungsi kembali `false` untuk nilai-nilai lainnya.

Sintaksis

```
IS_CHAR(super_expression)
```

Pendapat

super_ekspresi

`SUPER`Ekspresi atau kolom.

Jenis pengembalian

`BOOLEAN`

Contoh

Untuk memeriksa `t` apakah `CHAR` menggunakan fungsi IS_CHAR, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES ('t');
```

```
SELECT s, IS_CHAR(s) FROM t;

+-----+---------+
|  s  | is_char |
+-----+---------+
| "t" | true    |
+-----+---------+
```

Fungsi IS_DECIMAL

Memeriksa apakah suatu nilai adalah a`DECIMAL`. Fungsi IS_DECIMAL mengembalikan angka `true` yang bukan floating point. Fungsi kembali `false` untuk nilai-nilai lain, termasuk null.

Fungsi IS_DECIMAL adalah superset dari IS_BIGINT.

Sintaksis

```
IS_DECIMAL(super_expression)
```

Pendapat

super_ekspresi

`SUPER`Ekspresi atau kolom.

Jenis pengembalian

`BOOLEAN`

Contoh

Untuk memeriksa `1.22` apakah `DECIMAL` menggunakan fungsi IS_DECIMAL, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (1.22);
```

```
SELECT s, IS_DECIMAL(s) FROM t;

+------+------------+
|  s   | is_decimal |
+------+------------+
| 1.22 | true       |
+------+------------+
```

Fungsi IS_FLOAT

Memeriksa apakah suatu nilai adalah nomor floating point. Fungsi IS_FLOAT mengembalikan `true` untuk nomor floating point (FL0AT4dan). FL0AT8 Fungsi kembali `false` untuk nilai-nilai lainnya.

Himpunan IS_DECIMAL himpunan IS_FLOAT terputus.

Sintaksis

```
IS_FL0AT(super_expression)
```

Pendapat

super_ekspresi

SUPEREkspresi atau kolom.

Jenis pengembalian

B00LEAN

Contoh

Untuk memeriksa `2.22::`FL0AT apakah FL0AT menggunakan fungsi IS_FLOAT, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES(2.22::FLOAT);

SELECT s, IS_FLOAT(s) FROM t;
```

```
+---------+----------+
|    s    | is_float |
+---------+----------+
| 2.22e+0 | true     |
+---------+----------+
```

Fungsi IS_INTEGER

Pengembalian `true` untuk jumlah skala 0 dalam rentang 32-bit, dan `false` untuk hal lain (termasuk angka nol dan floating point).

Fungsi IS_INTEGER adalah superset dari fungsi IS_SMALLINT.

Sintaksis

```
IS_INTEGER(super_expression)
```

Pendapat

super_ekspresi

SUPEREkspresi atau kolom.

Jenis pengembalian

BOOLEAN

Contoh

Untuk memeriksa 5 apakah `INTEGER` menggunakan fungsi IS_INTEGER, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_INTEGER(s) FROM t;

+---+------------+
```

```
| s | is_integer |
+---+------------+
| 5 | true       |
+---+------------+
```

fungsi IS_OBJECT

Memeriksa apakah variabel adalah objek. Fungsi IS_OBJECT mengembalikan `true` untuk objek, termasuk objek kosong. Fungsi kembali `false` untuk nilai-nilai lain, termasuk null.

Sintaksis

```
IS_OBJECT(super_expression)
```

Pendapat

super_ekspresi

SUPEREkspresi atau kolom.

Jenis pengembalian

BOOLEAN

Contoh

Untuk memeriksa `{"name": "Joe"}` apakah objek menggunakan fungsi IS_OBJECT, gunakan contoh berikut.

```
CREATE TABLE t(s super);

INSERT INTO t VALUES (JSON_PARSE('{"name": "Joe"}'));

SELECT s, IS_OBJECT(s) FROM t;

+----------------+-----------+
|       s        | is_object |
+----------------+-----------+
| {"name":"Joe"} | true      |
```

```
+----------------+-----------+
```

Fungsi IS_SCALAR

Memeriksa apakah variabel adalah skalar. Fungsi IS_SCALAR mengembalikan `true` nilai apa pun yang bukan array atau objek. Fungsi kembali `false` untuk nilai-nilai lain, termasuk null.

Kumpulan IS_ARRAY, IS_OBJECT, dan IS_SCALAR mencakup semua nilai kecuali nol.

Sintaksis

```
IS_SCALAR(super_expression)
```

Pendapat

super_ekspresi

SUPEREkspresi atau kolom.

Jenis pengembalian

BOOLEAN

Contoh

Untuk memeriksa `{"name": "Joe"}` apakah skalar menggunakan fungsi IS_SCALAR, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (JSON_PARSE('{"name": "Joe"}'));

SELECT s, IS_SCALAR(s.name) FROM t;

+----------------+-----------+
|       s        | is_scalar |
+----------------+-----------+
| {"name":"Joe"} | true      |
+----------------+-----------+
```

Fungsi IS_SMALLINT

Memeriksa apakah variabel adalah a`SMALLINT`. Fungsi IS_SMALLINT mengembalikan `true` jumlah skala 0 dalam rentang 16-bit. Fungsi mengembalikan `false` nilai lainnya, termasuk angka nol dan floating point.

Sintaksis

```
IS_SMALLINT(super_expression)
```

Pendapat

super_ekspresi

SUPEREkspresi atau kolom.

Kembali

`BOOLEAN`

Contoh

Untuk memeriksa 5 apakah `SMALLINT` menggunakan fungsi IS_SMALLINT, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_SMALLINT(s) FROM t;

+---+-------------+
| s | is_smallint |
+---+-------------+
| 5 | true        |
+---+-------------+
```

Fungsi IS_VARCHAR

Memeriksa apakah variabel adalah a`VARCHAR`. Fungsi IS_VARCHAR mengembalikan `true` untuk semua string. Fungsi kembali `false` untuk nilai-nilai lainnya.

Fungsi IS_VARCHAR adalah superset dari fungsi IS_CHAR.

Sintaksis

```
IS_VARCHAR(super_expression)
```

Pendapat

super_ekspresi

SUPEREkspresi atau kolom.

Jenis pengembalian

BOOLEAN

Contoh

Untuk memeriksa abc apakah VARCHAR menggunakan fungsi IS_VARCHAR, gunakan contoh berikut.

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES ('abc');

SELECT s, IS_VARCHAR(s) FROM t;

+-------+------------+
|   s   | is_varchar |
+-------+------------+
| "abc" | true       |
+-------+------------+
```

Fungsi JSON_TYPEOF

Fungsi skalar JSON_TYPEOF mengembalikan a VARCHAR dengan nilai boolean, nomor, string, objek, array, atau null, tergantung pada jenis dinamis dari nilai. SUPER

Sintaksis

```
JSON_TYPEOF(super_expression)
```

Pendapat

super_ekspresi

SUPEREkspresi atau kolom.

Jenis pengembalian

VARCHAR

Contoh

Untuk memeriksa jenis JSON untuk array `[1,2]` menggunakan fungsi JSON_TYPEOF, gunakan contoh berikut.

```
SELECT JSON_TYPEOF(ARRAY(1,2));

+-------------+
| json_typeof |
+-------------+
| array       |
+-------------+
```

Fungsi VARBYTE

AWS Clean Roomsmendukung fungsi VARBYTE berikut.

Topik

- Fungsi FROM_HEX
- Fungsi FROM_VARBYTE
- Fungsi TO_HEX
- Fungsi TO_VARBYTE

Fungsi FROM_HEX

FROM_HEX mengkonversi heksadesimal ke nilai biner.

Sintaksis

```
FROM_HEX(hex_string)
```

Pendapat

hex_string

String heksadesimal tipe data VARCHAR atau TEXT yang akan dikonversi. Formatnya harus berupa nilai literal.

Jenis pengembalian

VARBYTE

Contoh

Untuk mengkonversi representasi heksadesimal '6162' ke nilai biner, gunakan contoh berikut. Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari nilai biner.

```
SELECT FROM_HEX('6162');

+----------+
| from_hex |
+----------+
|     6162 |
+----------+
```

Fungsi FROM_VARBYTE

FROM_VARBYTE mengkonversi nilai biner ke string karakter dalam format yang ditentukan.

Sintaksis

```
FROM_VARBYTE(binary_value, format)
```

Pendapat

binary_value

Nilai biner dari tipe dataVARBYTE.

format

Format string karakter yang dikembalikan. Nilai valid yang tidak peka huruf besar/kecil adalah `hexbinary`,`utf-8`, dan`utf8`.

Jenis pengembalian

`VARCHAR`

Contoh

Untuk mengonversi nilai biner `'ab'` menjadi heksadesimal, gunakan contoh berikut.

```
SELECT FROM_VARBYTE('ab', 'hex');

+--------------+
| from_varbyte |
+--------------+
|         6162 |
+--------------+
```

Fungsi TO_HEX

TO_HEX mengkonversi angka atau nilai biner ke representasi heksadesimal.

Sintaksis

```
TO_HEX(value)
```

Pendapat

nilai

Entah angka atau nilai biner (`VARBYTE`) yang akan dikonversi.

Jenis pengembalian

`VARCHAR`

Contoh

Untuk mengonversi angka ke representasi heksadesimal, gunakan contoh berikut.

```
SELECT TO_HEX(2147676847);

+----------+
|  to_hex  |
+----------+
| 8002f2af |
+----------+To create a table, insert the VARBYTE representation of 'abc' to a
 hexadecimal number, and select the column with the value, use the following example.
```

Fungsi TO_VARBYTE

TO_VARBYTE mengkonversi string dalam format tertentu ke nilai biner.

Sintaksis

```
TO_VARBYTE(string, format)
```

Pendapat

tali

A CHAR atau VARCHAR string.

format

Format string input. Nilai valid yang tidak peka huruf besar/kecil adalah `hexbinary`,`utf-8`, dan`utf8`.

Jenis pengembalian

`VARBYTE`

Contoh

Untuk mengkonversi hex 6162 ke nilai biner, gunakan contoh berikut. Hasilnya secara otomatis ditampilkan sebagai representasi heksadesimal dari nilai biner.

```
SELECT TO_VARBYTE('6162', 'hex');

+------------+
| to_varbyte |
+------------+
|       6162 |
+------------+
```

Fungsi jendela

Dengan menggunakan fungsi jendela, Anda dapat membuat kueri bisnis analitik dengan lebih efisien. Fungsi jendela beroperasi pada partisi atau "jendela" dari kumpulan hasil, dan mengembalikan nilai untuk setiap baris di jendela itu. Sebaliknya, fungsi non-windowed melakukan perhitungan mereka sehubungan dengan setiap baris dalam set hasil. Tidak seperti fungsi grup yang menggabungkan baris hasil, fungsi jendela mempertahankan semua baris dalam ekspresi tabel.

Nilai yang dikembalikan dihitung dengan menggunakan nilai dari kumpulan baris di jendela itu. Untuk setiap baris dalam tabel, jendela mendefinisikan satu set baris yang digunakan untuk menghitung atribut tambahan. Sebuah jendela didefinisikan menggunakan spesifikasi jendela (klausa OVER), dan didasarkan pada tiga konsep utama:

- Partisi jendela, yang membentuk kelompok baris (klausa PARTISI)
- Pengurutan jendela, yang mendefinisikan urutan atau urutan baris dalam setiap partisi (klausa ORDER BY)
- Bingkai jendela, yang didefinisikan relatif terhadap setiap baris untuk lebih membatasi set baris (spesifikasi ROWS)

Fungsi jendela adalah rangkaian operasi terakhir yang dilakukan dalam kueri kecuali klausa ORDER BY akhir. Semua klausa gabungan dan semua klausa WHERE, GROUP BY, dan HAVING selesai sebelum fungsi jendela diproses. Oleh karena itu, fungsi jendela hanya dapat muncul di daftar pilih atau klausa ORDER BY. Anda dapat menggunakan beberapa fungsi jendela dalam satu kueri dengan klausa bingkai yang berbeda. Anda juga dapat menggunakan fungsi jendela dalam ekspresi skalar lainnya, seperti CASE.

Ringkasan sintaks fungsi jendela

Fungsi jendela mengikuti sintaks standar, yaitu sebagai berikut.

```
function (expression) OVER (
[ PARTITION BY expr_list ]
[ ORDER BY order_list [ frame_clause ] ] )
```

Di sini, fungsi adalah salah satu fungsi yang dijelaskan dalam bagian ini.

Expr_list adalah sebagai berikut.

```
expression | column_name [, expr_list ]
```

Order_list adalah sebagai berikut.

```
expression | column_name [ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
[, order_list ]
```

Frame_clause adalah sebagai berikut.

```
ROWS
{ UNBOUNDED PRECEDING | unsigned_value PRECEDING | CURRENT ROW } |

{ BETWEEN
{ UNBOUNDED PRECEDING | unsigned_value { PRECEDING | FOLLOWING } | CURRENT ROW}
AND
{ UNBOUNDED FOLLOWING | unsigned_value { PRECEDING | FOLLOWING } | CURRENT ROW }}
```

Argumen

fungsi

Fungsi jendela. Untuk detailnya, lihat deskripsi fungsi individual.

DI ATAS

Klausul yang mendefinisikan spesifikasi jendela. Klausa OVER wajib untuk fungsi jendela, dan membedakan fungsi jendela dari fungsi SQL lainnya.

PARTISI OLEH expr_list

(Opsional) Klausa PARTITION BY membagi hasil yang ditetapkan menjadi partisi, seperti klausa GROUP BY. Jika klausa partisi hadir, fungsi dihitung untuk baris di setiap partisi. Jika tidak ada

klausa partisi yang ditentukan, partisi tunggal berisi seluruh tabel, dan fungsi dihitung untuk tabel lengkap itu.

Fungsi peringkat DENSE_RANK, NTILE, RANK, dan ROW_NUMBER memerlukan perbandingan global dari semua baris dalam kumpulan hasil. Ketika klausa PARTITION BY digunakan, pengoptimal kueri dapat menjalankan setiap agregasi secara paralel dengan menyebarkan beban kerja di beberapa irisan sesuai dengan partisi. Jika klausa PARTITION BY tidak ada, langkah agregasi harus dijalankan secara serial pada satu irisan, yang dapat memiliki dampak negatif yang signifikan pada kinerja, terutama untuk cluster besar.

AWS Clean Rooms tidak mendukung literal string dalam klausa PARTITION BY.

PESANAN BERDASARKAN order_list

(Opsional) Fungsi jendela diterapkan ke baris dalam setiap partisi yang diurutkan sesuai dengan spesifikasi pesanan di ORDER BY. Klausa ORDER BY ini berbeda dari dan sama sekali tidak terkait dengan klausa ORDER BY di frame_clause. Klausa ORDER BY dapat digunakan tanpa klausa PARTITION BY.

Untuk fungsi peringkat, klausa ORDER BY mengidentifikasi ukuran untuk nilai peringkat. Untuk fungsi agregasi, baris yang dipartisi harus diurutkan sebelum fungsi agregat dihitung untuk setiap frame. Untuk selengkapnya tentang jenis fungsi jendela, lihatFungsi jendela.

Pengidentifikasi kolom atau ekspresi yang mengevaluasi ke pengidentifikasi kolom diperlukan dalam daftar urutan. Baik konstanta maupun ekspresi konstan tidak dapat digunakan sebagai pengganti nama kolom.

Nilai NULLS diperlakukan sebagai grup mereka sendiri, diurutkan dan diberi peringkat sesuai dengan opsi NULLS FIRST atau NULLS LAST. Secara default, nilai NULL diurutkan dan diberi peringkat terakhir dalam urutan ASC, dan diurutkan dan diberi peringkat pertama dalam urutan DESC.

AWS Clean Rooms tidak mendukung literal string dalam klausa ORDER BY.

Jika klausa ORDER BY dihilangkan, urutan baris adalah nondeterministik.

> **Note**
>
> Dalam sistem paralel apa pun seperti AWS Clean Rooms, ketika klausa ORDER BY tidak menghasilkan urutan data yang unik dan total, urutan baris adalah nondeterministik. Artinya, jika ekspresi ORDER BY menghasilkan nilai duplikat (urutan sebagian), urutan

pengembalian baris tersebut dapat bervariasi dari satu proses AWS Clean Rooms ke yang berikutnya. Pada gilirannya, fungsi jendela mungkin mengembalikan hasil yang tidak terduga atau tidak konsisten. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

column_name

Nama kolom yang akan dipartisi oleh atau diurutkan oleh.

ASC | DESC

Opsi yang mendefinisikan urutan pengurutan untuk ekspresi, sebagai berikut:

- ASC: naik (misalnya, rendah ke tinggi untuk nilai numerik dan 'A' ke 'Z' untuk string karakter). Jika tidak ada opsi yang ditentukan, data diurutkan dalam urutan menaik secara default.
- DESC: turun (tinggi ke rendah untuk nilai numerik; 'Z' ke 'A' untuk string).

NULLS PERTAMA | NULLS TERAKHIR

Opsi yang menentukan apakah NULLS harus diurutkan terlebih dahulu, sebelum nilai non-null, atau terakhir, setelah nilai non-null. Secara default, NULLS diurutkan dan diberi peringkat terakhir dalam urutan ASC, dan diurutkan dan diberi peringkat pertama dalam urutan DESC.

frame_clause

Untuk fungsi agregat, klausa bingkai lebih lanjut menyempurnakan kumpulan baris di jendela fungsi saat menggunakan ORDER BY. Ini memungkinkan Anda untuk memasukkan atau mengecualikan set baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait.

Klausa bingkai tidak berlaku untuk fungsi peringkat. Selain itu, klausa bingkai tidak diperlukan ketika tidak ada klausa ORDER BY yang digunakan dalam klausa OVER untuk fungsi agregat. Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan.

Ketika tidak ada klausa ORDER BY yang ditentukan, bingkai tersirat tidak dibatasi, setara dengan BARIS ANTARA TIDAK TERBATAS SEBELUMNYA DAN TIDAK TERBATAS BERIKUT.

BARIS

Klausa ini mendefinisikan bingkai jendela dengan menentukan offset fisik dari baris saat ini.

Klausa ini menentukan baris di jendela atau partisi saat ini yang akan digabungkan dengan nilai dalam baris saat ini. Ini menggunakan argumen yang menentukan posisi baris, yang bisa sebelum

atau sesudah baris saat ini. Titik referensi untuk semua bingkai jendela adalah baris saat ini. Setiap baris menjadi baris saat ini secara bergantian saat bingkai jendela meluncur ke depan di partisi.

Bingkai dapat berupa serangkaian baris sederhana hingga dan termasuk baris saat ini.

```
{UNBOUNDED PRECEDING | offset PRECEDING | CURRENT ROW}
```

Atau bisa juga satu set baris antara dua batas.

```
BETWEEN
{ UNBOUNDED PRECEDING | offset { PRECEDING | FOLLOWING } | CURRENT ROW }
AND
{ UNBOUNDED FOLLOWING | offset { PRECEDING | FOLLOWING } | CURRENT ROW }
```

UNBOUNDED PRECEDING menunjukkan bahwa jendela dimulai pada baris pertama partisi; offset PRECEDING menunjukkan bahwa jendela memulai sejumlah baris yang setara dengan nilai offset sebelum baris saat ini. UNBOUNDED PRECEDING adalah default.

ROW SAAT INI menunjukkan jendela dimulai atau berakhir pada baris saat ini.

BERIKUT TIDAK TERBATAS menunjukkan bahwa jendela berakhir pada baris terakhir partisi; offset BERIKUT menunjukkan bahwa jendela mengakhiri sejumlah baris yang setara dengan nilai offset setelah baris saat ini.

offset mengidentifikasi jumlah fisik baris sebelum atau sesudah baris saat ini. Dalam hal ini, offset harus berupa konstanta yang mengevaluasi nilai numerik positif. Misalnya, 5 BERIKUT mengakhiri bingkai lima baris setelah baris saat ini.

Dimana ANTARA tidak ditentukan, frame secara implisit dibatasi oleh baris saat ini. Misalnya, `ROWS 5 PRECEDING` sama dengan`ROWS BETWEEN 5 PRECEDING AND CURRENT ROW`. Juga, `ROWS UNBOUNDED FOLLOWING` sama dengan`ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING`.

> **Note**
>
> Anda tidak dapat menentukan bingkai di mana batas awal lebih besar dari batas akhir. Misalnya, Anda tidak dapat menentukan salah satu frame berikut.
>
> ```
> between 5 following and 5 preceding
> ```

```
between current row and 2 preceding
between 3 following and current row
```

Urutan data yang unik untuk fungsi jendela

Jika klausa ORDER BY untuk fungsi jendela tidak menghasilkan urutan data yang unik dan total, urutan baris adalah nondeterministik. Jika ekspresi ORDER BY menghasilkan nilai duplikat (urutan sebagian), urutan pengembalian baris tersebut dapat bervariasi dalam beberapa kali proses. Dalam hal ini, fungsi jendela juga dapat mengembalikan hasil yang tidak terduga atau tidak konsisten.

Misalnya, kueri berikut mengembalikan hasil yang berbeda selama beberapa proses. Hasil yang berbeda ini terjadi karena `order by dateid` tidak menghasilkan urutan data yang unik untuk fungsi jendela SUM.

```
select dateid, pricepaid,
sum(pricepaid) over(order by dateid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;

dateid | pricepaid |   sumpaid
--------+-----------+-------------
1827 |   1730.00 |     1730.00
1827 |    708.00 |     2438.00
1827 |    234.00 |     2672.00
...

select dateid, pricepaid,
sum(pricepaid) over(order by dateid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;

dateid | pricepaid |   sumpaid
--------+-----------+-------------
1827 |    234.00 |      234.00
1827 |    472.00 |      706.00
1827 |    347.00 |     1053.00
...
```

Dalam hal ini, menambahkan kolom ORDER BY kedua ke fungsi jendela dapat menyelesaikan masalah.

```
select dateid, pricepaid,
sum(pricepaid) over(order by dateid, pricepaid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;

dateid | pricepaid | sumpaid
--------+-----------+---------
1827 |    234.00 |  234.00
1827 |    337.00 |  571.00
1827 |    347.00 |  918.00
...
```

Fungsi yang didukung

AWS Clean Rooms mendukung dua jenis fungsi jendela: agregat dan peringkat.

Berikut ini adalah fungsi agregat yang didukung:

- Fungsi jendela AVG
- Fungsi jendela COUNT
- Fungsi jendela CUME_DIST
- Fungsi jendela DENSE_RANK
- Fungsi jendela FIRST_VALUE
- Fungsi jendela LAG
- Fungsi jendela LAST_VALUE
- Fungsi jendela LEAD
- Fungsi jendela LISTAGG
- Fungsi jendela MAX
- Fungsi jendela MEDIAN
- Fungsi jendela MIN
- Fungsi jendela NTH_VALUE
- Fungsi jendela PERCENTILE_CONT
- Fungsi jendela PERCENTILE_DISC
- Fungsi jendela RATIO_TO_REPORT

- Fungsi jendela STDDEV_SAMP dan STDDEV_POP(STDDEV_SAMP dan STDDEV adalah sinonim)
- Fungsi jendela SUM
- Fungsi jendela VAR_SAMP dan VAR_POP(VAR_SAMP dan VARIANCE adalah sinonim)

Berikut ini adalah fungsi peringkat yang didukung:

- Fungsi jendela DENSE_RANK
- Fungsi jendela NTILE
- Fungsi jendela PERCENT_RANK
- Fungsi jendela RANK
- Fungsi jendela ROW_NUMBER

Contoh tabel untuk contoh fungsi jendela

Anda dapat menemukan contoh fungsi jendela tertentu dengan setiap deskripsi fungsi. Beberapa contoh menggunakan tabel bernama WINSALES, yang berisi 11 baris, seperti yang ditunjukkan pada tabel berikut.

SALESID	DATEID	SELLERID	PEMBELI	QTY	QTY_DIKIRIM
30001	8/2/2003	3	B	10	10
10001	12/24/2003	1	C	10	10
10005	12/24/2003	1	A	30	
40001	1/9/2004	4	A	40	
10006	1/18/2004	1	C	10	
20001	2/12/2004	2	B	20	20
40005	2/12/2004	4	A	10	10
20002	2/16/2004	2	C	20	20

SALESID	DATEID	SELLERID	PEMBELI	QTY	QTY_DIKIRIM
30003	4/18/2004	3	B	15	
30004	4/18/2004	3	B	20	
30007	9/7/2004	3	C	30	

Fungsi jendela AVG

Fungsi jendela AVG mengembalikan rata-rata (rata-rata aritmatika) dari nilai ekspresi masukan. Fungsi AVG bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaks

```
AVG ( [ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi AVG dalam hal satu atau beberapa ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_clause

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan. Klausa bingkai menyempurnakan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Tipe Data

Tipe argumen yang didukung oleh fungsi AVG adalah SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, REAL, dan DOUBLE PRECISION.

Jenis pengembalian yang didukung oleh fungsi AVG adalah:

- BIGINT untuk argumen SMALLINT atau INTEGER
- NUMERIK untuk argumen BIGINT
- PRESISI GANDA untuk argumen floating point

Contoh-contoh

Contoh berikut menghitung rata-rata bergulir dari jumlah yang dijual berdasarkan tanggal; pesan hasilnya berdasarkan ID tanggal dan ID penjualan:

```
select salesid, dateid, sellerid, qty,
avg(qty) over
(order by dateid, salesid rows unbounded preceding) as avg
from winsales
order by 2,1;

salesid |   dateid   | sellerid | qty | avg
---------+------------+----------+-----+-----
30001 | 2003-08-02 |        3 |  10 |  10
10001 | 2003-12-24 |        1 |  10 |  10
10005 | 2003-12-24 |        1 |  30 |  16
40001 | 2004-01-09 |        4 |  40 |  22
```

```
10006 | 2004-01-18 |        1 |  10 |  20
20001 | 2004-02-12 |        2 |  20 |  20
40005 | 2004-02-12 |        4 |  10 |  18
20002 | 2004-02-16 |        2 |  20 |  18
30003 | 2004-04-18 |        3 |  15 |  18
30004 | 2004-04-18 |        3 |  20 |  18
30007 | 2004-09-07 |        3 |  30 |  19
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Fungsi jendela COUNT

Fungsi jendela COUNT menghitung baris yang ditentukan oleh ekspresi.

Fungsi COUNT memiliki dua variasi. COUNT (*) menghitung semua baris dalam tabel target apakah mereka termasuk nol atau tidak. COUNT (ekspresi) menghitung jumlah baris dengan nilai non-Null dalam kolom atau ekspresi tertentu.

Sintaks

```
COUNT ( * | [ ALL ] expression) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi untuk menghitung. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi COUNT dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_clause

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan. Klausa bingkai menyempurnakan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Tipe Data

Fungsi COUNT mendukung semua tipe data argumen.

Jenis pengembalian yang didukung oleh fungsi COUNT adalah BIGINT.

Contoh-contoh

Contoh berikut menunjukkan ID penjualan, kuantitas, dan jumlah semua baris dari awal jendela data:

```
select salesid, qty,
count(*) over (order by salesid rows unbounded preceding) as count
from winsales
order by salesid;

salesid | qty | count
---------+-----+-----
10001 |  10 |   1
10005 |  30 |   2
10006 |  10 |   3
20001 |  20 |   4
20002 |  20 |   5
30001 |  10 |   6
30003 |  15 |   7
30004 |  20 |   8
30007 |  30 |   9
40001 |  40 |   10
40005 |  10 |   11
```

```
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Contoh berikut menunjukkan bagaimana ID penjualan, kuantitas, dan jumlah baris non-null dari awal jendela data. (Dalam tabel WINSALES, kolom QTY_SHIPPED berisi beberapa NULL.)

```
select salesid, qty, qty_shipped,
count(qty_shipped)
over (order by salesid rows unbounded preceding) as count
from winsales
order by salesid;

salesid | qty | qty_shipped | count
---------+-----+-------------+-------
10001 |  10 |          10 |   1
10005 |  30 |             |   1
10006 |  10 |             |   1
20001 |  20 |          20 |   2
20002 |  20 |          20 |   3
30001 |  10 |          10 |   4
30003 |  15 |             |   4
30004 |  20 |             |   4
30007 |  30 |             |   4
40001 |  40 |             |   4
40005 |  10 |          10 |   5
(11 rows)
```

Fungsi jendela CUME_DIST

Menghitung distribusi kumulatif nilai dalam jendela atau partisi. Dengan asumsi urutan naik, distribusi kumulatif ditentukan dengan menggunakan rumus ini:

```
count of rows with values <= x / count of rows in the window or partition
```

di mana x sama dengan nilai di baris kolom saat ini yang ditentukan dalam klausa ORDER BY. Dataset berikut menggambarkan penggunaan rumus ini:

```
Row# Value   Calculation    CUME_DIST
1        2500    (1)/(5)    0.2
2        2600    (2)/(5)    0.4
3        2800    (3)/(5)    0.6
4        2900    (4)/(5)    0.8
```

```
5        3100    (5)/(5)    1.0
```

Rentang nilai pengembalian adalah> 0 hingga 1, inklusif.

Sintaks

```
CUME_DIST ()
OVER (
[ PARTITION BY partition_expression ]
[ ORDER BY order_list ]
)
```

Argumen

DI ATAS

Sebuah klausa yang menentukan partisi jendela. Klausa OVER tidak dapat berisi spesifikasi bingkai jendela.

PARTISI OLEH partition_expression

Opsional. Ekspresi yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

PESANAN BERDASARKAN order_list

Ekspresi untuk menghitung distribusi kumulatif. Ekspresi harus memiliki tipe data numerik atau secara implisit dapat dikonversi menjadi satu. Jika ORDER BY dihilangkan, nilai kembalinya adalah 1 untuk semua baris.

Jika ORDER BY tidak menghasilkan urutan yang unik, urutan baris adalah nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

FLOAT8

Contoh-contoh

Contoh berikut menghitung distribusi kumulatif kuantitas untuk setiap penjual:

```
select sellerid, qty, cume_dist()
over (partition by sellerid order by qty)
from winsales;
```

```
sellerid   qty    cume_dist
-------------------------------------------------
1         10.00    0.33
1         10.64    0.67
1         30.37    1
3         10.04    0.25
3         15.15    0.5
3         20.75    0.75
3         30.55    1
2         20.09    0.5
2         20.12    1
4         10.12    0.5
4         40.23    1
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Fungsi jendela DENSE_RANK

Fungsi jendela DENSE_RANK menentukan peringkat nilai dalam sekelompok nilai, berdasarkan ekspresi ORDER BY dalam klausa OVER. Jika klausa PARTITION BY opsional ada, peringkat diatur ulang untuk setiap kelompok baris. Baris dengan nilai yang sama untuk kriteria peringkat menerima peringkat yang sama. Fungsi DENSE_RANK berbeda dari RANK dalam satu hal: Jika dua atau lebih baris terikat, tidak ada celah dalam urutan nilai peringkat. Misalnya, jika dua baris diberi peringkat 1, peringkat berikutnya adalah 2.

Anda dapat memiliki fungsi peringkat dengan klausa PARTITION BY dan ORDER BY yang berbeda dalam kueri yang sama.

Sintaks

```
DENSE_RANK () OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list ]
)
```

Argumen

( )

Fungsi ini tidak mengambil argumen, tetapi tanda kurung kosong diperlukan.

DI ATAS

Klausa jendela untuk fungsi DENSE_RANK.

PARTISI OLEH expr_list

Opsional. Satu atau lebih ekspresi yang menentukan jendela.

PESANAN BERDASARKAN order_list

Opsional. Ekspresi yang menjadi dasar nilai peringkat. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel. Jika ORDER BY dihilangkan, nilai kembalinya adalah 1 untuk semua baris.

Jika ORDER BY tidak menghasilkan urutan yang unik, urutan baris adalah nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

INTEGER

Contoh-contoh

Contoh berikut memesan tabel berdasarkan kuantitas yang terjual (dalam urutan menurun), dan menetapkan peringkat padat dan peringkat reguler untuk setiap baris. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan.

```
select salesid, qty,
dense_rank() over(order by qty desc) as d_rnk,
rank() over(order by qty desc) as rnk
from winsales
order by 2,1;

salesid | qty | d_rnk | rnk
---------+-----+-------+-----
10001 |  10 |     5 |   8
10006 |  10 |     5 |   8
30001 |  10 |     5 |   8
40005 |  10 |     5 |   8
30003 |  15 |     4 |   7
20001 |  20 |     3 |   4
20002 |  20 |     3 |   4
```

```
30004 |  20 |     3 |   4
10005 |  30 |     2 |   2
30007 |  30 |     2 |   2
40001 |  40 |     1 |   1
(11 rows)
```

Perhatikan perbedaan peringkat yang ditetapkan ke kumpulan baris yang sama saat fungsi DENSE_RANK dan RANK digunakan berdampingan dalam kueri yang sama. Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Contoh berikut mempartisi tabel oleh SELLERID dan memerintahkan setiap partisi dengan kuantitas (dalam urutan menurun) dan menetapkan peringkat padat untuk setiap baris. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan.

```
select salesid, sellerid, qty,
dense_rank() over(partition by sellerid order by qty desc) as d_rnk
from winsales
order by 2,3,1;

salesid | sellerid | qty | d_rnk
---------+----------+-----+-------
10001 |        1 |  10 |     2
10006 |        1 |  10 |     2
10005 |        1 |  30 |     1
20001 |        2 |  20 |     1
20002 |        2 |  20 |     1
30001 |        3 |  10 |     4
30003 |        3 |  15 |     3
30004 |        3 |  20 |     2
30007 |        3 |  30 |     1
40005 |        4 |  10 |     2
40001 |        4 |  40 |     1
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Fungsi jendela FIRST_VALUE

Diberikan kumpulan baris yang diurutkan, FIRST_VALUE mengembalikan nilai ekspresi yang ditentukan sehubungan dengan baris pertama di bingkai jendela.

Untuk informasi tentang memilih baris terakhir dalam bingkai, lihatFungsi jendela LAST_VALUE.

Sintaks

```
FIRST_VALUE( expression )[ IGNORE NULLS | RESPECT NULLS ]
OVER (
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

ABAIKAN NULLS

Ketika opsi ini digunakan dengan FIRST_VALUE, fungsi mengembalikan nilai pertama dalam frame yang tidak NULL (atau NULL jika semua nilai NULL).

RESPECT NULLS

Menunjukkan bahwa AWS Clean Rooms harus menyertakan nilai nol dalam penentuan baris mana yang akan digunakan. RESPECT NULLS didukung secara default jika Anda tidak menentukan IGNORE NULLS.

DI ATAS

Memperkenalkan klausa jendela untuk fungsi tersebut.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada klausa PARTITION BY yang ditentukan, ORDER BY mengurutkan seluruh tabel. Jika Anda menentukan klausa ORDER BY, Anda juga harus menentukan frame_clause.

Hasil fungsi FIRST_VALUE tergantung pada urutan data. Hasilnya nondeterministik dalam kasus-kasus berikut:

- Ketika tidak ada klausa ORDER BY ditentukan dan partisi berisi dua nilai yang berbeda untuk ekspresi

- Ketika ekspresi mengevaluasi nilai yang berbeda yang sesuai dengan nilai yang sama dalam daftar ORDER BY.

frame_clause

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan. Klausa bingkai menyempurnakan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis pengembalian

Fungsi-fungsi ini mendukung ekspresi yang menggunakan tipe AWS Clean Rooms data primitif. Tipe pengembalian sama dengan tipe data ekspresi.

Contoh-contoh

Contoh berikut mengembalikan kapasitas tempat duduk untuk setiap tempat di meja VENUE, dengan hasil yang diurutkan berdasarkan kapasitas (tinggi ke rendah). Fungsi FIRST_VALUE digunakan untuk memilih nama tempat yang sesuai dengan baris pertama dalam bingkai: dalam hal ini, baris dengan jumlah kursi tertinggi. Hasilnya dipartisi berdasarkan status, jadi ketika nilai VENUESTATE berubah, nilai pertama yang baru dipilih. Bingkai jendela tidak terbatas sehingga nilai pertama yang sama dipilih untuk setiap baris di setiap partisi.

Untuk California, `Qualcomm Stadium` memiliki jumlah kursi (`70561`) tertinggi, jadi nama ini adalah nilai pertama untuk semua baris di `CA` partisi.

```
select venuestate, venueseats, venuename,
first_value(venuename)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
order by venuestate;

venuestate | venueseats |           venuename            |         first_value
-----------+------------+--------------------------------
+-----------------------------
CA         |      70561 | Qualcomm Stadium               | Qualcomm Stadium
CA         |      69843 | Monster Park                   | Qualcomm Stadium
CA         |      63026 | McAfee Coliseum                | Qualcomm Stadium
```

```
CA         |      56000 | Dodger Stadium                | Qualcomm Stadium
CA         |      45050 | Angel Stadium of Anaheim      | Qualcomm Stadium
CA         |      42445 | PETCO Park                    | Qualcomm Stadium
CA         |      41503 | AT&T Park                     | Qualcomm Stadium
CA         |      22000 | Shoreline Amphitheatre        | Qualcomm Stadium
CO         |      76125 | INVESCO Field                 | INVESCO Field
CO         |      50445 | Coors Field                   | INVESCO Field
DC         |      41888 | Nationals Park                | Nationals Park
FL         |      74916 | Dolphin Stadium               | Dolphin Stadium
FL         |      73800 | Jacksonville Municipal Stadium | Dolphin Stadium
FL         |      65647 | Raymond James Stadium         | Dolphin Stadium
FL         |      36048 | Tropicana Field               | Dolphin Stadium
...
```

Fungsi jendela LAG

Fungsi jendela LAG mengembalikan nilai untuk baris pada offset tertentu di atas (sebelum) baris saat ini di partisi.

Sintaks

```
LAG (value_expr [, offset ])
[ IGNORE NULLS | RESPECT NULLS ]
OVER ( [ PARTITION BY window_partition ] ORDER BY window_ordering )
```

Argumen

value_expr

Kolom target atau ekspresi tempat fungsi beroperasi.

mengimbangi

Parameter opsional yang menentukan jumlah baris sebelum baris saat ini untuk mengembalikan nilai untuk. Offset dapat berupa bilangan bulat konstan atau ekspresi yang mengevaluasi ke bilangan bulat. Jika Anda tidak menentukan offset, AWS Clean Rooms gunakan 1 sebagai nilai default. Offset `0` menunjukkan baris saat ini.

ABAIKAN NULLS

Spesifikasi opsional yang menunjukkan bahwa AWS Clean Rooms harus melewati nilai nol dalam penentuan baris mana yang akan digunakan. Nilai nol disertakan jika IGNORE NULLS tidak terdaftar.

> **Note**
>
> Anda dapat menggunakan ekspresi NVL atau COALESCE untuk mengganti nilai null dengan nilai lain.

RESPECT NULLS

Menunjukkan bahwa AWS Clean Rooms harus menyertakan nilai nol dalam penentuan baris mana yang akan digunakan. RESPECT NULLS didukung secara default jika Anda tidak menentukan IGNORE NULLS.

DI ATAS

Menentukan jendela partisi dan pemesanan. Klausa OVER tidak dapat berisi spesifikasi bingkai jendela.

PARTISI OLEH window_partition

Argumen opsional yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

PESANAN DENGAN window_ordering

Mengurutkan baris dalam setiap partisi.

Fungsi jendela LAG mendukung ekspresi yang menggunakan salah satu tipe AWS Clean Rooms data. Jenis pengembalian sama dengan tipe value_expr.

Contoh-contoh

Contoh berikut menunjukkan jumlah tiket yang dijual kepada pembeli dengan ID pembeli 3 dan waktu pembeli 3 membeli tiket. Untuk membandingkan setiap penjualan dengan penjualan sebelumnya untuk pembeli 3, kueri mengembalikan jumlah sebelumnya yang dijual untuk setiap penjualan. Karena tidak ada pembelian sebelum 1/16/2008, nilai jual kuantitas pertama sebelumnya adalah nol:

```
select buyerid, saletime, qtysold,
lag(qtysold,1) over (order by buyerid, saletime) as prev_qtysold
from sales where buyerid = 3 order by buyerid, saletime;

buyerid |      saletime       | qtysold | prev_qtysold
---------+---------------------+---------+--------------
3 | 2008-01-16 01:06:09 |       1 |
3 | 2008-01-28 02:10:01 |       1 |            1
```

```
3 | 2008-03-12 10:39:53 |       1 |            1
3 | 2008-03-13 02:56:07 |       1 |            1
3 | 2008-03-29 08:21:39 |       2 |            1
3 | 2008-04-27 02:39:01 |       1 |            2
3 | 2008-08-16 07:04:37 |       2 |            1
3 | 2008-08-22 11:45:26 |       2 |            2
3 | 2008-09-12 09:11:25 |       1 |            2
3 | 2008-10-01 06:22:37 |       1 |            1
3 | 2008-10-20 01:55:51 |       2 |            1
3 | 2008-10-28 01:30:40 |       1 |            2
(12 rows)
```

Fungsi jendela LAST_VALUE

Diberikan kumpulan baris yang diurutkan, fungsi LAST_VALUE mengembalikan nilai ekspresi sehubungan dengan baris terakhir dalam bingkai.

Untuk informasi tentang memilih baris pertama dalam bingkai, lihatFungsi jendela FIRST_VALUE.

Sintaks

```
LAST_VALUE( expression )[ IGNORE NULLS | RESPECT NULLS ]
OVER (
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

ABAIKAN NULLS

Fungsi mengembalikan nilai terakhir dalam frame yang tidak NULL (atau NULL jika semua nilai NULL).

RESPECT NULLS

Menunjukkan bahwa AWS Clean Rooms harus menyertakan nilai nol dalam penentuan baris mana yang akan digunakan. RESPECT NULLS didukung secara default jika Anda tidak menentukan IGNORE NULLS.

DI ATAS

Memperkenalkan klausa jendela untuk fungsi tersebut.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada klausa PARTITION BY yang ditentukan, ORDER BY mengurutkan seluruh tabel. Jika Anda menentukan klausa ORDER BY, Anda juga harus menentukan frame_clause.

Hasilnya tergantung pada urutan data. Hasilnya nondeterministik dalam kasus-kasus berikut:

- Ketika tidak ada klausa ORDER BY ditentukan dan partisi berisi dua nilai yang berbeda untuk ekspresi
- Ketika ekspresi mengevaluasi nilai yang berbeda yang sesuai dengan nilai yang sama dalam daftar ORDER BY.

frame_clause

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan. Klausa bingkai menyempurnakan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Jenis pengembalian

Fungsi-fungsi ini mendukung ekspresi yang menggunakan tipe AWS Clean Rooms data primitif. Tipe pengembalian sama dengan tipe data ekspresi.

Contoh-contoh

Contoh berikut mengembalikan kapasitas tempat duduk untuk setiap tempat di meja VENUE, dengan hasil yang diurutkan berdasarkan kapasitas (tinggi ke rendah). Fungsi LAST_VALUE digunakan untuk memilih nama tempat yang sesuai dengan baris terakhir dalam bingkai: dalam hal ini, baris dengan jumlah kursi paling sedikit. Hasilnya dipartisi berdasarkan status, jadi ketika nilai VENUESTATE berubah, nilai terakhir yang baru dipilih. Bingkai jendela tidak terbatas sehingga nilai terakhir yang sama dipilih untuk setiap baris di setiap partisi.

Untuk California, `Shoreline Amphitheatre` dikembalikan untuk setiap baris di partisi karena memiliki jumlah kursi terendah (`22000`).

```
select venuestate, venueseats, venuename,
last_value(venuename)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
order by venuestate;

venuestate | venueseats |           venuename            |          last_value
-----------+------------+--------------------------------
+------------------------------
CA         |      70561 | Qualcomm Stadium               | Shoreline Amphitheatre
CA         |      69843 | Monster Park                   | Shoreline Amphitheatre
CA         |      63026 | McAfee Coliseum                | Shoreline Amphitheatre
CA         |      56000 | Dodger Stadium                 | Shoreline Amphitheatre
CA         |      45050 | Angel Stadium of Anaheim       | Shoreline Amphitheatre
CA         |      42445 | PETCO Park                     | Shoreline Amphitheatre
CA         |      41503 | AT&T Park                      | Shoreline Amphitheatre
CA         |      22000 | Shoreline Amphitheatre         | Shoreline Amphitheatre
CO         |      76125 | INVESCO Field                  | Coors Field
CO         |      50445 | Coors Field                    | Coors Field
DC         |      41888 | Nationals Park                 | Nationals Park
FL         |      74916 | Dolphin Stadium                | Tropicana Field
FL         |      73800 | Jacksonville Municipal Stadium | Tropicana Field
FL         |      65647 | Raymond James Stadium          | Tropicana Field
FL         |      36048 | Tropicana Field                | Tropicana Field
...
```

Fungsi jendela LEAD

Fungsi jendela LEAD mengembalikan nilai untuk baris pada offset tertentu di bawah (setelah) baris saat ini di partisi.

Sintaks

```
LEAD (value_expr [, offset ])
[ IGNORE NULLS | RESPECT NULLS ]
OVER ( [ PARTITION BY window_partition ] ORDER BY window_ordering )
```

Argumen

value_expr

Kolom target atau ekspresi tempat fungsi beroperasi.

mengimbangi

Parameter opsional yang menentukan jumlah baris di bawah baris saat ini untuk mengembalikan nilai untuk. Offset dapat berupa bilangan bulat konstan atau ekspresi yang mengevaluasi ke bilangan bulat. Jika Anda tidak menentukan offset, AWS Clean Rooms gunakan 1 sebagai nilai default. Offset `0` menunjukkan baris saat ini.

ABAIKAN NULLS

Spesifikasi opsional yang menunjukkan bahwa AWS Clean Rooms harus melewati nilai nol dalam penentuan baris mana yang akan digunakan. Nilai nol disertakan jika IGNORE NULLS tidak terdaftar.

> **Note**
>
> Anda dapat menggunakan ekspresi NVL atau COALESCE untuk mengganti nilai null dengan nilai lain.

RESPECT NULLS

Menunjukkan bahwa AWS Clean Rooms harus menyertakan nilai nol dalam penentuan baris mana yang akan digunakan. RESPECT NULLS didukung secara default jika Anda tidak menentukan IGNORE NULLS.

DI ATAS

Menentukan jendela partisi dan pemesanan. Klausa OVER tidak dapat berisi spesifikasi bingkai jendela.

PARTISI OLEH window_partition

Argumen opsional yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

PESANAN DENGAN window_ordering

Mengurutkan baris dalam setiap partisi.

Fungsi jendela LEAD mendukung ekspresi yang menggunakan salah satu tipe AWS Clean Rooms data. Jenis pengembalian sama dengan tipe value_expr.

Contoh-contoh

Contoh berikut memberikan komisi untuk acara-acara di tabel PENJUALAN di mana tiket dijual pada 1 Januari 2008 dan 2 Januari 2008 dan komisi yang dibayarkan untuk penjualan tiket untuk penjualan berikutnya.

```
select eventid, commission, saletime,
lead(commission, 1) over (order by saletime) as next_comm
from sales where saletime between '2008-01-01 00:00:00' and '2008-01-02 12:59:59'
order by saletime;

eventid | commission |      saletime       | next_comm
---------+------------+---------------------+-----------
6213 |      52.05 | 2008-01-01 01:00:19 |    106.20
7003 |     106.20 | 2008-01-01 02:30:52 |    103.20
8762 |     103.20 | 2008-01-01 03:50:02 |     70.80
1150 |      70.80 | 2008-01-01 06:06:57 |     50.55
1749 |      50.55 | 2008-01-01 07:05:02 |    125.40
8649 |     125.40 | 2008-01-01 07:26:20 |     35.10
2903 |      35.10 | 2008-01-01 09:41:06 |    259.50
6605 |     259.50 | 2008-01-01 12:50:55 |    628.80
6870 |     628.80 | 2008-01-01 12:59:34 |     74.10
6977 |      74.10 | 2008-01-02 01:11:16 |     13.50
4650 |      13.50 | 2008-01-02 01:40:59 |     26.55
4515 |      26.55 | 2008-01-02 01:52:35 |     22.80
5465 |      22.80 | 2008-01-02 02:28:01 |     45.60
5465 |      45.60 | 2008-01-02 02:28:02 |     53.10
7003 |      53.10 | 2008-01-02 02:31:12 |     70.35
4124 |      70.35 | 2008-01-02 03:12:50 |     36.15
1673 |      36.15 | 2008-01-02 03:15:00 |   1300.80
...
(39 rows)
```

Fungsi jendela LISTAGG

Untuk setiap grup dalam kueri, fungsi jendela LISTAGG memerintahkan baris untuk grup tersebut sesuai dengan ekspresi ORDER BY, lalu menggabungkan nilai menjadi satu string.

LISTAGG adalah fungsi compute-node saja. Fungsi mengembalikan kesalahan jika kueri tidak mereferensikan tabel atau tabel AWS Clean Rooms sistem yang ditentukan pengguna.

Sintaks

```
LISTAGG( [DISTINCT] expression [, 'delimiter' ] )
[ WITHIN GROUP (ORDER BY order_list) ]
OVER ( [PARTITION BY partition_expression] )
```

Argumen

DISTINCT

(Opsional) Klausa yang menghilangkan nilai duplikat dari ekspresi yang ditentukan sebelum digabungkan. Spasi trailing diabaikan, sehingga string `'a'` dan `'a '` diperlakukan sebagai duplikat. LISTAGG menggunakan nilai pertama yang ditemui. Untuk informasi selengkapnya, lihat Signifikansi trailing blanko.

aggregate_expression

Ekspresi yang valid (seperti nama kolom) yang memberikan nilai untuk digabungkan. Nilai NULL dan string kosong diabaikan.

pembatas

(Opsional) Konstanta string untuk memisahkan nilai gabungan. Default-nya adalah NULL.

AWS Clean Rooms mendukung sejumlah spasi utama atau belakang di sekitar koma opsional atau titik dua serta string kosong atau sejumlah spasi.

Contoh nilai yang valid adalah:

`', '`

`': '`

`' '`

DALAM GRUP (PESANAN BERDASARKAN order_list)

(Opsional) Sebuah klausa yang menentukan urutan dari nilai agregat. Deterministik hanya jika ORDER BY menyediakan urutan unik. Defaultnya adalah menggabungkan semua baris dan mengembalikan satu nilai.

DI ATAS

Sebuah klausa yang menentukan partisi jendela. Klausa OVER tidak dapat berisi urutan jendela atau spesifikasi bingkai jendela.

PARTISI OLEH partition_expression

(Opsional) Menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

Pengembalian

VARCHAR (MAKS). Jika set hasil lebih besar dari ukuran VARCHAR maksimum (64K-1, atau 65535), maka LISTAGG mengembalikan kesalahan berikut:

```
Invalid operation: Result size exceeds LISTAGG limit
```

Contoh-contoh

Contoh berikut menggunakan tabel WINSALES. Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Contoh berikut mengembalikan daftar ID penjual, diurutkan oleh ID penjual.

```
select listagg(sellerid)
within group (order by sellerid)
over() from winsales;

  listagg
------------
 11122333344
...
...
 11122333344
 11122333344
   (11 rows)
```

Contoh berikut mengembalikan daftar ID penjual untuk pembeli B, dipesan berdasarkan tanggal.

```
select listagg(sellerid)
within group (order by dateid)
```

```
over () as seller
from winsales
where buyerid = 'b' ;

  seller
---------
    3233
    3233
    3233
    3233

(4 rows)
```

Contoh berikut mengembalikan daftar tanggal penjualan yang dipisahkan koma untuk pembeli B.

```
select listagg(dateid,',')
within group (order by sellerid desc,salesid asc)
over () as dates
from winsales
where buyerid  = 'b';

             dates
----------------------------------------
2003-08-02,2004-04-18,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-04-18,2004-02-12

(4 rows)
```

Contoh berikut menggunakan DISTINCT untuk mengembalikan daftar tanggal penjualan unik untuk pembeli B.

```
select listagg(distinct dateid,',')
within group (order by sellerid desc,salesid asc)
over () as dates
from winsales
where buyerid  = 'b';

           dates
--------------------------------
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12
```

```
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12

(4 rows)
```

Contoh berikut mengembalikan daftar ID penjualan yang dipisahkan koma untuk setiap ID pembeli.

```
select buyerid,
listagg(salesid,',')
within group (order by salesid)
over (partition by buyerid) as sales_id
from winsales
order by buyerid;

   buyerid | sales_id
-----------+------------------------
         a  |10005,40001,40005
         a  |10005,40001,40005
         a  |10005,40001,40005
         b  |20001,30001,30004,30003
         b  |20001,30001,30004,30003
         b  |20001,30001,30004,30003
         b  |20001,30001,30004,30003
         c  |10001,20002,30007,10006
         c  |10001,20002,30007,10006
         c  |10001,20002,30007,10006
         c  |10001,20002,30007,10006
(11 rows)
```

Fungsi jendela MAX

Fungsi jendela MAX mengembalikan maksimum nilai ekspresi masukan. Fungsi MAX bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaks

```
MAX ( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Sebuah klausa yang menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi MAX dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_clause

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan. Klausa bingkai menyempurnakan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Tipe Data

Menerima tipe data apa pun sebagai input. Mengembalikan tipe data yang sama sebagai ekspresi.

Contoh-contoh

Contoh berikut menunjukkan ID penjualan, kuantitas, dan kuantitas maksimum dari awal jendela data:

```
select salesid, qty,
```

```
max(qty) over (order by salesid rows unbounded preceding) as max
from winsales
order by salesid;

salesid | qty | max
---------+-----+-----
10001 |  10 |  10
10005 |  30 |  30
10006 |  10 |  30
20001 |  20 |  30
20002 |  20 |  30
30001 |  10 |  30
30003 |  15 |  30
30004 |  20 |  30
30007 |  30 |  30
40001 |  40 |  40
40005 |  10 |  40
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Contoh berikut menunjukkan salesid, kuantitas, dan kuantitas maksimum dalam bingkai terbatas:

```
select salesid, qty,
max(qty) over (order by salesid rows between 2 preceding and 1 preceding) as max
from winsales
order by salesid;

salesid | qty | max
---------+-----+-----
10001 |  10 |
10005 |  30 |  10
10006 |  10 |  30
20001 |  20 |  30
20002 |  20 |  20
30001 |  10 |  20
30003 |  15 |  20
30004 |  20 |  15
30007 |  30 |  20
40001 |  40 |  30
40005 |  10 |  40
(11 rows)
```

Fungsi jendela MEDIAN

Menghitung nilai median untuk rentang nilai di jendela atau partisi. Nilai NULL dalam rentang diabaikan.

MEDIAN adalah fungsi distribusi terbalik yang mengasumsikan model distribusi kontinu.

MEDIAN adalah fungsi compute-node saja. Fungsi mengembalikan kesalahan jika kueri tidak mereferensikan tabel atau tabel AWS Clean Rooms sistem yang ditentukan pengguna.

Sintaks

```
MEDIAN ( median_expression )
OVER ( [ PARTITION BY partition_expression ] )
```

Argumen

median_expression

Ekspresi, seperti nama kolom, yang memberikan nilai untuk menentukan median. Ekspresi harus memiliki tipe data numerik atau datetime atau secara implisit dapat dikonversi menjadi satu.

DI ATAS

Sebuah klausa yang menentukan partisi jendela. Klausa OVER tidak dapat berisi urutan jendela atau spesifikasi bingkai jendela.

PARTISI OLEH partition_expression

Opsional. Ekspresi yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

Tipe Data

Jenis pengembalian ditentukan oleh tipe data median_expression. Tabel berikut menunjukkan tipe kembali untuk setiap tipe data median_expression.

Tipe input	Tipe Pengembalian
NUMERIK, DESIMAL	DECIMAL
MENGAPUNG, GANDA	DOUBLE

Tipe input	Tipe Pengembalian
DATE	DATE

Catatan penggunaan

Jika argumen median_expression adalah tipe data DECIMAL yang didefinisikan dengan presisi maksimum 38 digit, ada kemungkinan MEDIAN akan mengembalikan hasil yang tidak akurat atau kesalahan. Jika nilai pengembalian fungsi MEDIAN melebihi 38 digit, hasilnya terpotong agar sesuai, yang menyebabkan hilangnya presisi. Jika, selama interpolasi, hasil antara melebihi presisi maksimum, luapan numerik terjadi dan fungsi mengembalikan kesalahan. Untuk menghindari kondisi ini, sebaiknya gunakan tipe data dengan presisi lebih rendah atau mentransmisikan argumen median_expression ke presisi yang lebih rendah.

Misalnya, fungsi SUM dengan argumen DECIMAL mengembalikan presisi default 38 digit. Skala hasilnya sama dengan skala argumen. Jadi, misalnya, SUM kolom DECIMAL (5,2) mengembalikan tipe data DECIMAL (38,2).

Contoh berikut menggunakan fungsi SUM dalam argumen median_expression dari fungsi MEDIAN. Tipe data dari kolom PRICEPAID adalah DECIMAL (8,2), sehingga fungsi SUM mengembalikan DECIMAL (38,2).

```
select salesid, sum(pricepaid), median(sum(pricepaid))
over() from sales where salesid < 10 group by salesid;
```

Untuk menghindari potensi kehilangan presisi atau kesalahan luapan, lemparkan hasilnya ke tipe data DECIMAL dengan presisi lebih rendah, seperti yang ditunjukkan contoh berikut.

```
select salesid, sum(pricepaid), median(sum(pricepaid)::decimal(30,2))
over() from sales where salesid < 10 group by salesid;
```

Contoh-contoh

Contoh berikut menghitung jumlah penjualan rata-rata untuk setiap penjual:

```
select sellerid, qty, median(qty)
over (partition by sellerid)
from winsales
```

```
order by sellerid;


sellerid qty median
---------------------------
1  10 10.0
1  10 10.0
1  30 10.0
2  20 20.0
2  20 20.0
3  10 17.5
3  15 17.5
3  20 17.5
3  30 17.5
4  10 25.0
4  40 25.0
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Fungsi jendela MIN

Fungsi jendela MIN mengembalikan minimum nilai ekspresi masukan. Fungsi MIN bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaks

```
MIN ( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi MIN dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_clause

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan. Klausa bingkai menyempurnakan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Tipe Data

Menerima tipe data apa pun sebagai input. Mengembalikan tipe data yang sama sebagai ekspresi.

Contoh-contoh

Contoh berikut menunjukkan ID penjualan, kuantitas, dan kuantitas minimum dari awal jendela data:

```
select salesid, qty,
min(qty) over
(order by salesid rows unbounded preceding)
from winsales
order by salesid;

salesid | qty | min
---------+-----+-----
10001 |  10 |  10
10005 |  30 |  10
10006 |  10 |  10
20001 |  20 |  10
20002 |  20 |  10
30001 |  10 |  10
30003 |  15 |  10
```

```
30004 |  20 |  10
30007 |  30 |  10
40001 |  40 |  10
40005 |  10 |  10
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Contoh berikut menunjukkan ID penjualan, kuantitas, dan kuantitas minimum dalam bingkai terbatas:

```
select salesid, qty,
min(qty) over
(order by salesid rows between 2 preceding and 1 preceding) as min
from winsales
order by salesid;

salesid | qty | min
---------+-----+-----
10001 |  10 |
10005 |  30 |  10
10006 |  10 |  10
20001 |  20 |  10
20002 |  20 |  10
30001 |  10 |  20
30003 |  15 |  10
30004 |  20 |  10
30007 |  30 |  15
40001 |  40 |  20
40005 |  10 |  30
(11 rows)
```

Fungsi jendela NTH_VALUE

Fungsi jendela NTH_VALUE mengembalikan nilai ekspresi dari baris tertentu dari bingkai jendela relatif terhadap baris pertama jendela.

Sintaks

```
NTH_VALUE (expr, offset)
[ IGNORE NULLS | RESPECT NULLS ]
OVER
( [ PARTITION BY window_partition ]
```

```
[ ORDER BY window_ordering
                          frame_clause ] )
```

Argumen

expr

Kolom target atau ekspresi tempat fungsi beroperasi.

mengimbangi

Menentukan nomor baris relatif terhadap baris pertama di jendela untuk mengembalikan ekspresi. Offset dapat berupa konstanta atau ekspresi dan harus berupa bilangan bulat positif yang lebih besar dari 0.

ABAIKAN NULLS

Spesifikasi opsional yang menunjukkan bahwa AWS Clean Rooms harus melewati nilai nol dalam penentuan baris mana yang akan digunakan. Nilai nol disertakan jika IGNORE NULLS tidak terdaftar.

RESPECT NULLS

Menunjukkan bahwa AWS Clean Rooms harus menyertakan nilai nol dalam penentuan baris mana yang akan digunakan. RESPECT NULLS didukung secara default jika Anda tidak menentukan IGNORE NULLS.

DI ATAS

Menentukan partisi jendela, pemesanan, dan bingkai jendela.

PARTISI OLEH window_partition

Menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

PESANAN DENGAN window_ordering

Mengurutkan baris dalam setiap partisi. Jika ORDER BY dihilangkan, bingkai default terdiri dari semua baris di partisi.

frame_clause

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan. Klausa bingkai menyempurnakan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Fungsi jendela NTH_VALUE mendukung ekspresi yang menggunakan salah satu tipe data. AWS Clean Rooms Jenis pengembalian sama dengan tipe expr.

Contoh-contoh

Contoh berikut menunjukkan jumlah kursi di tempat terbesar ketiga di California, Florida, dan New York dibandingkan dengan jumlah kursi di tempat lain di negara bagian tersebut:

```
select venuestate, venuename, venueseats,
nth_value(venueseats, 3)
ignore nulls
over(partition by venuestate order by venueseats desc
rows between unbounded preceding and unbounded following)
as third_most_seats
from (select * from venue where venueseats > 0 and
venuestate in('CA', 'FL', 'NY'))
order by venuestate;

venuestate |           venuename            | venueseats | third_most_seats
-----------+--------------------------------+------------+------------------
CA         | Qualcomm Stadium               |      70561 |            63026
CA         | Monster Park                   |      69843 |            63026
CA         | McAfee Coliseum                |      63026 |            63026
CA         | Dodger Stadium                 |      56000 |            63026
CA         | Angel Stadium of Anaheim       |      45050 |            63026
CA         | PETCO Park                     |      42445 |            63026
CA         | AT&T Park                      |      41503 |            63026
CA         | Shoreline Amphitheatre         |      22000 |            63026
FL         | Dolphin Stadium                |      74916 |            65647
FL         | Jacksonville Municipal Stadium |      73800 |            65647
FL         | Raymond James Stadium          |      65647 |            65647
FL         | Tropicana Field                |      36048 |            65647
NY         | Ralph Wilson Stadium           |      73967 |            20000
NY         | Yankee Stadium                 |      52325 |            20000
NY         | Madison Square Garden          |      20000 |            20000
(15 rows)
```

Fungsi jendela NTILE

Fungsi jendela NTILE membagi baris yang diurutkan dalam partisi ke dalam jumlah kelompok peringkat yang ditentukan dengan ukuran yang sama mungkin dan mengembalikan grup tempat baris tertentu jatuh ke dalamnya.

Sintaks

```
NTILE (expr)
OVER (
[ PARTITION BY expression_list ]
[ ORDER BY order_list ]
)
```

Argumen

expr

Jumlah kelompok peringkat dan harus menghasilkan nilai integer positif (lebih besar dari 0) untuk setiap partisi. Argumen expr tidak boleh dibatalkan.

DI ATAS

Sebuah klausa yang menentukan jendela partisi dan pemesanan. Klausa OVER tidak dapat berisi spesifikasi bingkai jendela.

PARTISI OLEH window_partition

Opsional. Rentang catatan untuk setiap grup dalam klausa OVER.

PESANAN DENGAN window_ordering

Opsional. Ekspresi yang mengurutkan baris dalam setiap partisi. Jika klausa ORDER BY dihilangkan, perilaku peringkatnya sama.

Jika ORDER BY tidak menghasilkan urutan unik, urutan baris adalah nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

BIGINT

Contoh-contoh

Contoh berikut peringkat ke dalam empat kelompok peringkat harga yang dibayarkan untuk tiket Hamlet pada 26 Agustus 2008. Hasil set adalah 17 baris, dibagi hampir merata di antara peringkat 1 sampai 4:

```
select eventname, caldate, pricepaid, ntile(4)
```

```
over(order by pricepaid desc) from sales, event, date
where sales.eventid=event.eventid and event.dateid=date.dateid and eventname='Hamlet'
and caldate='2008-08-26'
order by 4;

eventname |  caldate   | pricepaid | ntile
----------+------------+-----------+-------
Hamlet    | 2008-08-26 |   1883.00 |     1
Hamlet    | 2008-08-26 |   1065.00 |     1
Hamlet    | 2008-08-26 |    589.00 |     1
Hamlet    | 2008-08-26 |    530.00 |     1
Hamlet    | 2008-08-26 |    472.00 |     1
Hamlet    | 2008-08-26 |    460.00 |     2
Hamlet    | 2008-08-26 |    355.00 |     2
Hamlet    | 2008-08-26 |    334.00 |     2
Hamlet    | 2008-08-26 |    296.00 |     2
Hamlet    | 2008-08-26 |    230.00 |     3
Hamlet    | 2008-08-26 |    216.00 |     3
Hamlet    | 2008-08-26 |    212.00 |     3
Hamlet    | 2008-08-26 |    106.00 |     3
Hamlet    | 2008-08-26 |    100.00 |     4
Hamlet    | 2008-08-26 |     94.00 |     4
Hamlet    | 2008-08-26 |     53.00 |     4
Hamlet    | 2008-08-26 |     25.00 |     4
(17 rows)
```

Fungsi jendela PERCENT_RANK

Menghitung peringkat persen dari baris yang diberikan. Peringkat persen ditentukan dengan menggunakan rumus ini:

```
(x - 1) / (the number of rows in the window or partition - 1)
```

dimana x adalah pangkat dari baris saat ini. Dataset berikut menggambarkan penggunaan rumus ini:

```
Row# Value Rank Calculation PERCENT_RANK
1 15 1 (1-1)/(7-1) 0.0000
2 20 2 (2-1)/(7-1) 0.1666
3 20 2 (2-1)/(7-1) 0.1666
4 20 2 (2-1)/(7-1) 0.1666
5 30 5 (5-1)/(7-1) 0.6666
6 30 5 (5-1)/(7-1) 0.6666
7 40 7 (7-1)/(7-1) 1.0000
```

Rentang nilai pengembalian adalah 0 hingga 1, inklusif. Baris pertama dalam set apa pun memiliki PERCENT_RANK 0.

Sintaks

```
PERCENT_RANK ()
OVER (
[ PARTITION BY partition_expression ]
[ ORDER BY order_list ]
)
```

Argumen

( )

Fungsi ini tidak mengambil argumen, tetapi tanda kurung kosong diperlukan.

DI ATAS

Sebuah klausa yang menentukan partisi jendela. Klausa OVER tidak dapat berisi spesifikasi bingkai jendela.

PARTISI OLEH partition_expression

Opsional. Ekspresi yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

PESANAN BERDASARKAN order_list

Opsional. Ekspresi untuk menghitung peringkat persen. Ekspresi harus memiliki tipe data numerik atau secara implisit dapat dikonversi menjadi satu. Jika ORDER BY dihilangkan, nilai kembalinya adalah 0 untuk semua baris.

Jika ORDER BY tidak menghasilkan urutan unik, urutan baris adalah nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

FLOAT8

Contoh-contoh

Contoh berikut menghitung peringkat persen dari jumlah penjualan untuk setiap penjual:

```
select sellerid, qty, percent_rank()
over (partition by sellerid order by qty)
from winsales;

sellerid qty  percent_rank
----------------------------------------
1  10.00  0.0
1  10.64  0.5
1  30.37  1.0
3  10.04  0.0
3  15.15  0.33
3  20.75  0.67
3  30.55  1.0
2  20.09  0.0
2  20.12  1.0
4  10.12  0.0
4  40.23  1.0
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Fungsi jendela PERCENTILE_CONT

PERCENTILE_CONT adalah fungsi distribusi terbalik yang mengasumsikan model distribusi kontinu. Dibutuhkan nilai persentil dan spesifikasi sortir, dan mengembalikan nilai interpolasi yang akan jatuh ke dalam nilai persentil yang diberikan sehubungan dengan spesifikasi sortir.

PERCENTILE_CONT menghitung interpolasi linier antara nilai setelah mengurutkannya. Menggunakan nilai persentil `(P)` dan jumlah baris bukan nol `(N)` dalam grup agregasi, fungsi menghitung nomor baris setelah mengurutkan baris sesuai dengan spesifikasi pengurutan. Nomor baris ini `(RN)` dihitung sesuai dengan `RN = (1+ (P*(N-1))` rumus. Hasil akhir dari fungsi agregat dihitung dengan interpolasi linier antara nilai-nilai dari baris pada nomor baris dan. `CRN = CEILING(RN)` `FRN = FLOOR(RN)`

Hasil akhirnya adalah sebagai berikut.

Jika `(CRN = FRN = RN)` maka hasilnya adalah `(value of expression from row at RN)`

Jika tidak, hasilnya adalah sebagai berikut:

`(CRN - RN) * (value of expression for row at FRN) + (RN - FRN) * (value of expression for row at CRN)`.

Anda hanya dapat menentukan klausa PARTITION dalam klausa OVER. Jika PARTITION ditentukan, untuk setiap baris, PERCENTILE_CONT mengembalikan nilai yang akan jatuh ke dalam persentil yang ditentukan di antara satu set nilai dalam partisi yang diberikan.

PERCENTILE_CONT adalah fungsi compute-node saja. Fungsi mengembalikan kesalahan jika kueri tidak mereferensikan tabel atau tabel AWS Clean Rooms sistem yang ditentukan pengguna.

Sintaks

```
PERCENTILE_CONT ( percentile )
WITHIN GROUP (ORDER BY expr)
OVER (  [ PARTITION BY expr_list ]  )
```

Argumen

persentil

Konstanta numerik antara 0 dan 1. Null diabaikan dalam perhitungan.

DALAM GRUP (ORDER BY expr)

Menentukan nilai numerik atau tanggal/waktu untuk mengurutkan dan menghitung persentil atas.

DI ATAS

Menentukan partisi jendela. Klausa OVER tidak dapat berisi urutan jendela atau spesifikasi bingkai jendela.

PARTISI OLEH expr

Argumen opsional yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

Pengembalian

Tipe pengembalian ditentukan oleh tipe data ekspresi ORDER BY dalam klausa WITHIN GROUP. Tabel berikut menunjukkan tipe pengembalian untuk setiap tipe data ekspresi ORDER BY.

Tipe input	Tipe Pengembalian
SMALLINTINTEGERBIGINTNUMERIK, DESIMAL	DECIMAL

Tipe input	Tipe Pengembalian
MENGAPUNG, GANDA	DOUBLE
DATE	DATE
TIMESTAMP	TIMESTAMP

Catatan penggunaan

Jika ekspresi ORDER BY adalah tipe data DECIMAL yang ditentukan dengan presisi maksimum 38 digit, ada kemungkinan bahwa PERCENTILE_CONT akan mengembalikan hasil yang tidak akurat atau kesalahan. Jika nilai pengembalian fungsi PERCENTILE_CONT melebihi 38 digit, hasilnya dipotong agar sesuai, yang menyebabkan hilangnya presisi. Jika, selama interpolasi, hasil antara melebihi presisi maksimum, luapan numerik terjadi dan fungsi mengembalikan kesalahan. Untuk menghindari kondisi ini, sebaiknya gunakan tipe data dengan presisi lebih rendah atau mentransmisikan ekspresi ORDER BY ke presisi yang lebih rendah.

Misalnya, fungsi SUM dengan argumen DECIMAL mengembalikan presisi default 38 digit. Skala hasilnya sama dengan skala argumen. Jadi, misalnya, SUM kolom DECIMAL (5,2) mengembalikan tipe data DECIMAL (38,2).

Contoh berikut menggunakan fungsi SUM dalam klausa ORDER BY dari fungsi PERCENTILE_CONT. Tipe data dari kolom PRICEPAID adalah DECIMAL (8,2), sehingga fungsi SUM mengembalikan DECIMAL (38,2).

```
select salesid, sum(pricepaid), percentile_cont(0.6)
within group (order by sum(pricepaid) desc) over()
from sales where salesid < 10 group by salesid;
```

Untuk menghindari potensi kehilangan presisi atau kesalahan luapan, lemparkan hasilnya ke tipe data DECIMAL dengan presisi lebih rendah, seperti yang ditunjukkan contoh berikut.

```
select salesid, sum(pricepaid), percentile_cont(0.6)
within group (order by sum(pricepaid)::decimal(30,2) desc) over()
from sales where salesid < 10 group by salesid;
```

Contoh-contoh

Contoh berikut menggunakan tabel WINSALES. Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

```
select sellerid, qty, percentile_cont(0.5)
within group (order by qty)
over() as median from winsales;

 sellerid | qty | median
----------+-----+--------
        1 |  10 |   20.0
        1 |  10 |   20.0
        3 |  10 |   20.0
        4 |  10 |   20.0
        3 |  15 |   20.0
        2 |  20 |   20.0
        3 |  20 |   20.0
        2 |  20 |   20.0
        3 |  30 |   20.0
        1 |  30 |   20.0
        4 |  40 |   20.0
(11 rows)
```

```
select sellerid, qty, percentile_cont(0.5)
within group (order by qty)
over(partition by sellerid) as median from winsales;

 sellerid | qty | median
----------+-----+--------
        2 |  20 |   20.0
        2 |  20 |   20.0
        4 |  10 |   25.0
        4 |  40 |   25.0
        1 |  10 |   10.0
        1 |  10 |   10.0
        1 |  30 |   10.0
        3 |  10 |   17.5
        3 |  15 |   17.5
        3 |  20 |   17.5
        3 |  30 |   17.5
(11 rows)
```

Contoh berikut menghitung PERCENTILE_CONT dan PERCENTILE_DISC dari penjualan tiket untuk penjual di negara bagian Washington.

```
SELECT sellerid, state, sum(qtysold*pricepaid) sales,
percentile_cont(0.6) within group (order by sum(qtysold*pricepaid::decimal(14,2) )
 desc) over(),
percentile_disc(0.6) within group (order by sum(qtysold*pricepaid::decimal(14,2) )
 desc) over()
from sales s, users u
where s.sellerid = u.userid and state = 'WA' and sellerid < 1000
group by sellerid, state;

 sellerid | state |  sales  | percentile_cont | percentile_disc
----------+-------+---------+-----------------+-----------------
      127 | WA    | 6076.00 |         2044.20 |         1531.00
      787 | WA    | 6035.00 |         2044.20 |         1531.00
      381 | WA    | 5881.00 |         2044.20 |         1531.00
      777 | WA    | 2814.00 |         2044.20 |         1531.00
       33 | WA    | 1531.00 |         2044.20 |         1531.00
      800 | WA    | 1476.00 |         2044.20 |         1531.00
        1 | WA    | 1177.00 |         2044.20 |         1531.00
(7 rows)
```

Fungsi jendela PERCENTILE_DISC

PERCENTILE_DISC adalah fungsi distribusi terbalik yang mengasumsikan model distribusi diskrit. Dibutuhkan nilai persentil dan spesifikasi semacam dan mengembalikan elemen dari set yang diberikan.

Untuk nilai persentil yang diberikan P, PERCENTILE_DISC mengurutkan nilai ekspresi dalam klausa ORDER BY dan mengembalikan nilai dengan nilai distribusi kumulatif terkecil (sehubungan dengan spesifikasi pengurutan yang sama) yang lebih besar dari atau sama dengan P.

Anda hanya dapat menentukan klausa PARTITION dalam klausa OVER.

PERCENTILE_DISC adalah fungsi compute-node saja. Fungsi mengembalikan kesalahan jika kueri tidak mereferensikan tabel atau tabel AWS Clean Rooms sistem yang ditentukan pengguna.

Sintaks

```
PERCENTILE_DISC ( percentile )
```

```
WITHIN GROUP (ORDER BY expr)
OVER (  [ PARTITION BY expr_list ]  )
```

Argumen

persentil

Konstanta numerik antara 0 dan 1. Null diabaikan dalam perhitungan.

DALAM GRUP (ORDER BY expr)

Menentukan nilai numerik atau tanggal/waktu untuk mengurutkan dan menghitung persentil atas.

DI ATAS

Menentukan partisi jendela. Klausa OVER tidak dapat berisi urutan jendela atau spesifikasi bingkai jendela.

PARTISI OLEH expr

Argumen opsional yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

Pengembalian

Tipe data yang sama dengan ekspresi ORDER BY dalam klausa WITHIN GROUP.

Contoh-contoh

Contoh berikut menggunakan tabel WINSALES. Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

```
select sellerid, qty, percentile_disc(0.5)
within group (order by qty)
over() as median from winsales;

sellerid | qty | median
----------+-----+--------
        1 |  10 |     20
        3 |  10 |     20
        1 |  10 |     20
        4 |  10 |     20
        3 |  15 |     20
        2 |  20 |     20
```

```
        2 |  20 |     20
        3 |  20 |     20
        1 |  30 |     20
        3 |  30 |     20
        4 |  40 |     20
(11 rows)
```

```
select sellerid, qty, percentile_disc(0.5)
within group (order by qty)
over(partition by sellerid) as median from winsales;

sellerid | qty | median
----------+-----+--------
        2 |  20 |     20
        2 |  20 |     20
        4 |  10 |     10
        4 |  40 |     10
        1 |  10 |     10
        1 |  10 |     10
        1 |  30 |     10
        3 |  10 |     15
        3 |  15 |     15
        3 |  20 |     15
        3 |  30 |     15
(11 rows)
```

Fungsi jendela RANK

Fungsi jendela RANK menentukan peringkat nilai dalam sekelompok nilai, berdasarkan ekspresi ORDER BY dalam klausa OVER. Jika klausa PARTITION BY opsional ada, peringkat diatur ulang untuk setiap kelompok baris. Baris dengan nilai yang sama untuk kriteria peringkat menerima peringkat yang sama. AWS Clean Rooms menambahkan jumlah baris terikat ke peringkat terikat untuk menghitung peringkat berikutnya dan dengan demikian peringkat mungkin bukan angka berurutan. Misalnya, jika dua baris diberi peringkat 1, peringkat berikutnya adalah 3.

RANK berbeda dari Fungsi jendela DENSE_RANK dalam satu hal: Untuk DENSE_RANK, jika dua atau lebih baris mengikat, tidak ada celah dalam urutan nilai peringkat. Misalnya, jika dua baris diberi peringkat 1, peringkat berikutnya adalah 2.

Anda dapat memiliki fungsi peringkat dengan klausa PARTITION BY dan ORDER BY yang berbeda dalam kueri yang sama.

Sintaks

```
RANK () OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list ]
)
```

Argumen

( )

Fungsi ini tidak mengambil argumen, tetapi tanda kurung kosong diperlukan.

DI ATAS

Jendela klausa untuk fungsi RANK.

PARTISI OLEH expr_list

Opsional. Satu atau lebih ekspresi yang menentukan jendela.

PESANAN BERDASARKAN order_list

Opsional. Mendefinisikan kolom yang menjadi dasar nilai peringkat. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel. Jika ORDER BY dihilangkan, nilai kembalinya adalah 1 untuk semua baris.

Jika ORDER BY tidak menghasilkan urutan unik, urutan baris adalah nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

INTEGER

Contoh-contoh

Contoh berikut memesan tabel berdasarkan kuantitas yang dijual (naik default), dan menetapkan peringkat untuk setiap baris. Nilai peringkat 1 adalah nilai peringkat tertinggi. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan:

```
select salesid, qty,
rank() over (order by qty) as rnk
```

```
from winsales
order by 2,1;

salesid | qty | rnk
--------+-----+-----
10001 |  10 |  1
10006 |  10 |  1
30001 |  10 |  1
40005 |  10 |  1
30003 |  15 |  5
20001 |  20 |  6
20002 |  20 |  6
30004 |  20 |  6
10005 |  30 |  9
30007 |  30 |  9
40001 |  40 |  11
(11 rows)
```

Perhatikan bahwa klausa ORDER BY luar dalam contoh ini menyertakan kolom 2 dan 1 untuk memastikan bahwa AWS Clean Rooms mengembalikan hasil yang diurutkan secara konsisten setiap kali kueri ini dijalankan. Misalnya, baris dengan ID penjualan 10001 dan 10006 memiliki nilai QTY dan RNK yang identik. Memesan hasil akhir yang ditetapkan oleh kolom 1 memastikan bahwa baris 10001 selalu jatuh sebelum 10006. Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Dalam contoh berikut, urutan dibalik untuk fungsi window (`order by qty desc`). Sekarang nilai peringkat tertinggi berlaku untuk nilai QTY terbesar.

```
select salesid, qty,
rank() over (order by qty desc) as rank
from winsales
order by 2,1;

 salesid | qty | rank
---------+-----+-----
   10001 |  10 |   8
   10006 |  10 |   8
   30001 |  10 |   8
   40005 |  10 |   8
   30003 |  15 |   7
   20001 |  20 |   4
   20002 |  20 |   4
```

```
   30004 |  20 |   4
   10005 |  30 |   2
   30007 |  30 |   2
   40001 |  40 |   1
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Contoh berikut mempartisi tabel oleh SELLERID dan mengurutkan setiap partisi dengan kuantitas (dalam urutan menurun) dan menetapkan peringkat untuk setiap baris. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan.

```
select salesid, sellerid, qty, rank() over
(partition by sellerid
order by qty desc) as rank
from winsales
order by 2,3,1;

salesid | sellerid | qty | rank
--------+----------+-----+-----
  10001 |        1 |  10 |  2
  10006 |        1 |  10 |  2
  10005 |        1 |  30 |  1
  20001 |        2 |  20 |  1
  20002 |        2 |  20 |  1
  30001 |        3 |  10 |  4
  30003 |        3 |  15 |  3
  30004 |        3 |  20 |  2
  30007 |        3 |  30 |  1
  40005 |        4 |  10 |  2
  40001 |        4 |  40 |  1
(11 rows)
```

Fungsi jendela RATIO_TO_REPORT

Menghitung rasio nilai dengan jumlah nilai di jendela atau partisi. Rasio terhadap nilai laporan ditentukan dengan menggunakan rumus:

`value of` ratio_expression ratio_expression `argument for the current row / sum of argument for the window or partition`

Dataset berikut menggambarkan penggunaan rumus ini:

```
Row# Value Calculation RATIO_TO_REPORT
1 2500 (2500)/(13900) 0.1798
2 2600 (2600)/(13900) 0.1870
3 2800 (2800)/(13900) 0.2014
4 2900 (2900)/(13900) 0.2086
5 3100 (3100)/(13900) 0.2230
```

Rentang nilai pengembalian adalah 0 hingga 1, inklusif. Jika ratio_expression adalah NULL, maka nilai kembaliannya adalah NULL.

Sintaks

```
RATIO_TO_REPORT ( ratio_expression )
OVER ( [ PARTITION BY partition_expression ] )
```

Argumen

ratio_expression

Ekspresi, seperti nama kolom, yang memberikan nilai untuk menentukan rasio. Ekspresi harus memiliki tipe data numerik atau secara implisit dapat dikonversi menjadi satu.

Anda tidak dapat menggunakan fungsi analitik lainnya di ratio_expression.

DI ATAS

Sebuah klausa yang menentukan partisi jendela. Klausa OVER tidak dapat berisi urutan jendela atau spesifikasi bingkai jendela.

PARTISI OLEH partition_expression

Opsional. Ekspresi yang menetapkan rentang catatan untuk setiap grup dalam klausa OVER.

Jenis pengembalian

FLOAT8

Contoh-contoh

Contoh berikut menghitung rasio jumlah penjualan untuk setiap penjual:

```
select sellerid, qty, ratio_to_report(qty)
```

```
over (partition by sellerid)
from winsales;

sellerid qty  ratio_to_report
----------------------------------------
2  20.12312341      0.5
2  20.08630000      0.5
4  10.12414400      0.2
4  40.23000000      0.8
1  30.37262000      0.6
1  10.64000000      0.21
1  10.00000000      0.2
3  10.03500000      0.13
3  15.14660000      0.2
3  30.54790000      0.4
3  20.74630000      0.27
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Fungsi jendela ROW_NUMBER

Menentukan nomor urut dari baris saat ini dalam sekelompok baris, dihitung dari 1, berdasarkan ekspresi ORDER BY dalam klausa OVER. Jika klausa PARTITION BY opsional ada, nomor urut diatur ulang untuk setiap kelompok baris. Baris dengan nilai yang sama untuk ekspresi ORDER BY menerima nomor baris yang berbeda secara nondeterministik.

Sintaks

```
ROW_NUMBER () OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list ]
)
```

Argumen

( )

Fungsi ini tidak mengambil argumen, tetapi tanda kurung kosong diperlukan.

DI ATAS

Klausa jendela untuk fungsi ROW_NUMBER.

PARTISI OLEH expr_list

Opsional. Satu atau lebih ekspresi yang mendefinisikan fungsi ROW_NUMBER.

PESANAN BERDASARKAN order_list

Opsional. Ekspresi yang mendefinisikan kolom yang menjadi dasar nomor baris. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

Jika ORDER BY tidak menghasilkan urutan unik atau dihilangkan, urutan baris adalah nondeterministik. Untuk informasi selengkapnya, lihat Urutan data yang unik untuk fungsi jendela.

Jenis pengembalian

BIGINT

Contoh-contoh

Contoh berikut mempartisi tabel oleh SELLERID dan memerintahkan setiap partisi dengan QTY (dalam urutan menaik), kemudian menetapkan nomor baris untuk setiap baris. Hasilnya diurutkan setelah hasil fungsi jendela diterapkan.

```
select salesid, sellerid, qty,
row_number() over
(partition by sellerid
 order by qty asc) as row
from winsales
order by 2,4;

 salesid | sellerid | qty | row
---------+----------+-----+-----
   10006 |        1 |  10 |   1
   10001 |        1 |  10 |   2
   10005 |        1 |  30 |   3
   20001 |        2 |  20 |   1
   20002 |        2 |  20 |   2
   30001 |        3 |  10 |   1
   30003 |        3 |  15 |   2
   30004 |        3 |  20 |   3
   30007 |        3 |  30 |   4
   40005 |        4 |  10 |   1
   40001 |        4 |  40 |   2
```

```
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Fungsi jendela STDDEV_SAMP dan STDDEV_POP

Fungsi jendela STDDEV_SAMP dan STDDEV_POP mengembalikan sampel dan standar deviasi populasi dari satu set nilai numerik (integer, desimal, atau floating-point). Lihat juga Fungsi STDDEV_SAMP dan STDDEV_POP.

STDDEV_SAMP dan STDDEV adalah sinonim untuk fungsi yang sama.

Sintaks

```
STDDEV_SAMP | STDDEV | STDDEV_POP
( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_clause

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan. Klausa bingkai menyempurnakan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Tipe Data

Jenis argumen yang didukung oleh fungsi STDDEV adalah SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, REAL, dan DOUBLE PRECISION.

Terlepas dari tipe data ekspresi, tipe pengembalian fungsi STDDEV adalah nomor presisi ganda.

Contoh-contoh

Contoh berikut menunjukkan bagaimana menggunakan fungsi STDDEV_POP dan VAR_POP sebagai fungsi jendela. Kueri menghitung varians populasi dan deviasi standar populasi untuk nilai PRICEPAID dalam tabel PENJUALAN.

```
select salesid, dateid, pricepaid,
round(stddev_pop(pricepaid) over
(order by dateid, salesid rows unbounded preceding)) as stddevpop,
round(var_pop(pricepaid) over
(order by dateid, salesid rows unbounded preceding)) as varpop
from sales
order by 2,1;

salesid | dateid | pricepaid | stddevpop | varpop
--------+--------+-----------+-----------+---------
  33095 |   1827 |    234.00 |         0 |       0
  65082 |   1827 |    472.00 |       119 |   14161
  88268 |   1827 |    836.00 |       248 |   61283
  97197 |   1827 |    708.00 |       230 |   53019
 110328 |   1827 |    347.00 |       223 |   49845
 110917 |   1827 |    337.00 |       215 |   46159
 150314 |   1827 |    688.00 |       211 |   44414
```

```
 157751 |   1827 |   1730.00 |       447 |  199679
 165890 |   1827 |   4192.00 |      1185 | 1403323
...
```

Fungsi standar deviasi dan varians sampel dapat digunakan dengan cara yang sama.

Fungsi jendela SUM

Fungsi jendela SUM mengembalikan jumlah kolom input atau nilai ekspresi. Fungsi SUM bekerja dengan nilai numerik dan mengabaikan nilai NULL.

Sintaks

```
SUM ( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi SUM dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_clause

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan. Klausa bingkai menyempurnakan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Tipe Data

Tipe argumen yang didukung oleh fungsi SUM adalah SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, REAL, dan DOUBLE PRECISION.

Jenis pengembalian yang didukung oleh fungsi SUM adalah:

- BIGINT untuk argumen SMALLINT atau INTEGER
- NUMERIK untuk argumen BIGINT
- PRESISI GANDA untuk argumen floating-point

Contoh-contoh

Contoh berikut membuat jumlah kumulatif (bergulir) dari jumlah penjualan yang diurutkan berdasarkan tanggal dan ID penjualan:

```
select salesid, dateid, sellerid, qty,
sum(qty) over (order by dateid, salesid rows unbounded preceding) as sum
from winsales
order by 2,1;

salesid |   dateid   | sellerid | qty | sum
---------+------------+----------+-----+-----
30001 | 2003-08-02 |        3 |  10 |  10
10001 | 2003-12-24 |        1 |  10 |  20
10005 | 2003-12-24 |        1 |  30 |  50
40001 | 2004-01-09 |        4 |  40 |  90
10006 | 2004-01-18 |        1 |  10 | 100
20001 | 2004-02-12 |        2 |  20 | 120
40005 | 2004-02-12 |        4 |  10 | 130
20002 | 2004-02-16 |        2 |  20 | 150
30003 | 2004-04-18 |        3 |  15 | 165
30004 | 2004-04-18 |        3 |  20 | 185
```

```
30007 | 2004-09-07 |        3 |  30 | 215
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Contoh berikut membuat jumlah kumulatif (bergulir) jumlah penjualan berdasarkan tanggal, mempartisi hasil dengan ID penjual, dan memesan hasil berdasarkan tanggal dan ID penjualan dalam partisi:

```
select salesid, dateid, sellerid, qty,
sum(qty) over (partition by sellerid
order by dateid, salesid rows unbounded preceding) as sum
from winsales
order by 2,1;

salesid |   dateid   | sellerid | qty | sum
---------+------------+----------+-----+-----
30001 | 2003-08-02 |        3 |  10 |  10
10001 | 2003-12-24 |        1 |  10 |  10
10005 | 2003-12-24 |        1 |  30 |  40
40001 | 2004-01-09 |        4 |  40 |  40
10006 | 2004-01-18 |        1 |  10 |  50
20001 | 2004-02-12 |        2 |  20 |  20
40005 | 2004-02-12 |        4 |  10 |  50
20002 | 2004-02-16 |        2 |  20 |  40
30003 | 2004-04-18 |        3 |  15 |  25
30004 | 2004-04-18 |        3 |  20 |  45
30007 | 2004-09-07 |        3 |  30 |  75
(11 rows)
```

Contoh berikut memberi nomor semua baris secara berurutan dalam kumpulan hasil, diurutkan oleh kolom SELLERID dan SALESID:

```
select salesid, sellerid, qty,
sum(1) over (order by sellerid, salesid rows unbounded preceding) as rownum
from winsales
order by 2,1;

salesid | sellerid |  qty | rownum
--------+----------+------+--------
10001 |        1 |   10 |      1
10005 |        1 |   30 |      2
```

```
10006 |        1 |   10 |     3
20001 |        2 |   20 |     4
20002 |        2 |   20 |     5
30001 |        3 |   10 |     6
30003 |        3 |   15 |     7
30004 |        3 |   20 |     8
30007 |        3 |   30 |     9
40001 |        4 |   40 |    10
40005 |        4 |   10 |    11
(11 rows)
```

Untuk deskripsi tabel WINSALES, lihatContoh tabel untuk contoh fungsi jendela.

Contoh berikut memberi nomor semua baris secara berurutan dalam kumpulan hasil, partisi hasil dengan SELLERID, dan urutkan hasilnya berdasarkan SELLERID dan SALESID dalam partisi:

```
select salesid, sellerid, qty,
sum(1) over (partition by sellerid
order by sellerid, salesid rows unbounded preceding) as rownum
from winsales
order by 2,1;

salesid | sellerid | qty | rownum
---------+----------+-----+--------
10001 |        1 |  10 |      1
10005 |        1 |  30 |      2
10006 |        1 |  10 |      3
20001 |        2 |  20 |      1
20002 |        2 |  20 |      2
30001 |        3 |  10 |      1
30003 |        3 |  15 |      2
30004 |        3 |  20 |      3
30007 |        3 |  30 |      4
40001 |        4 |  40 |      1
40005 |        4 |  10 |      2
(11 rows)
```

Fungsi jendela VAR_SAMP dan VAR_POP

Fungsi jendela VAR_SAMP dan VAR_POP mengembalikan sampel dan varians populasi dari sekumpulan nilai numerik (integer, desimal, atau floating-point). Lihat juga Fungsi VAR_SAMP dan VAR_POP.

VAR_SAMP dan VARIANCE adalah sinonim untuk fungsi yang sama.

Sintaks

```
VAR_SAMP | VARIANCE | VAR_POP
( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

Argumen

ekspresi

Kolom target atau ekspresi tempat fungsi beroperasi.

SEMUA

Dengan argumen ALL, fungsi mempertahankan semua nilai duplikat dari ekspresi. ALL adalah default. DISTINCT tidak didukung.

DI ATAS

Menentukan klausa jendela untuk fungsi agregasi. Klausa OVER membedakan fungsi agregasi jendela dari fungsi agregasi set normal.

PARTISI OLEH expr_list

Mendefinisikan jendela untuk fungsi dalam hal satu atau lebih ekspresi.

PESANAN BERDASARKAN order_list

Mengurutkan baris dalam setiap partisi. Jika tidak ada PARTITION BY yang ditentukan, ORDER BY menggunakan seluruh tabel.

frame_clause

Jika klausa ORDER BY digunakan untuk fungsi agregat, klausa bingkai eksplisit diperlukan. Klausa bingkai menyempurnakan kumpulan baris di jendela fungsi, termasuk atau mengecualikan kumpulan baris dalam hasil yang diurutkan. Klausa bingkai terdiri dari kata kunci ROWS dan penentu terkait. Lihat Ringkasan sintaks fungsi jendela.

Tipe Data

Tipe argumen yang didukung oleh fungsi VARIANCE adalah SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, REAL, dan DOUBLE PRECISION.

Terlepas dari tipe data ekspresi, tipe pengembalian fungsi VARIANCE adalah angka presisi ganda.

Kondisi SQL di AWS Clean Rooms

Kondisi adalah pernyataan dari satu atau lebih ekspresi dan operator logis yang mengevaluasi menjadi benar, salah, atau tidak diketahui. Kondisi juga kadang-kadang disebut sebagai predikat.

> **Note**
>
> Semua perbandingan string dan kecocokan pola LIKE peka huruf besar/kecil. Misalnya, 'A' dan 'a' tidak cocok. Namun, Anda dapat melakukan kecocokan pola case-insensitive dengan menggunakan predikat ILIKE.

Kondisi SQL berikut didukung di AWS Clean Rooms.

Topik

Kondisi perbandingan

Kondisi perbandingan menyatakan hubungan logis antara dua nilai. Semua kondisi perbandingan adalah operator biner dengan tipe pengembalian Boolean.AWS Clean Roomsmendukung operator perbandingan yang dijelaskan dalam tabel berikut.

Operator	Sintaksis	Deskripsi
<	a < b	Nilaiakurang dari nilaib.
>	a > b	Nilaialebih besar dari nilaib.

Operator	Sintaksis	Deskripsi
<=	`a <= b`	Nilaiakurang dari atau sama dengan nilaib.
>=	`a >= b`	Nilaialebih besar dari atau sama dengan nilaib.
=	`a = b`	Nilaiasama dengan nilaib.
<> atau! =	`a <> b or a != b`	Nilaiatidak sama dengan nilaib.
a = TRUE	`a IS TRUE`	Nilaiaadalah BooleanTRUE.

Catatan penggunaan

= APAPUN | BEBERAPA

APAPUN dan BEBERAPA kata kunci identik denganDIkondisi. ANY dan BEBERAPA kata kunci mengembalikan true jika perbandingan benar untuk setidaknya satu nilai yang dikembalikan oleh subquery yang mengembalikan satu atau lebih nilai.AWS Clean Roomsmendukung hanya = (sama) kondisi untuk APAPUN dan BEBERAPA. Kondisi ketidaksetaraan tidak didukung.

> **Note**
>
> Predikat ALL tidak didukung.

<> SEMUA

Kata kunci ALL identik dengan NOT IN (lihatDalam kondisikondisi) dan mengembalikan true jika ekspresi tidak termasuk dalam hasil subquery.AWS Clean Roomsmendukung hanya <> atau! = (tidak sama) kondisi untuk SEMUA. Kondisi perbandingan lainnya tidak didukung.

BENAR/SALAH/TIDAK DIKETAHUI

Nilai non-nol sama dengan TRUE, 0 setara dengan FALSE, dan null setara dengan UNKNOWN. LihatJenis Booleantipe data.

Contoh

Berikut adalah beberapa contoh sederhana dari kondisi perbandingan:

```
a = 5
a < b
min(x) >= 5
qtysold = any (select qtysold from sales where dateid = 1882
```

Kueri berikut mengembalikan tempat dengan lebih dari 10.000 kursi dari tabel VENUE:

```
select venueid, venuename, venueseats from venue
where venueseats > 10000
order by venueseats desc;

venueid |           venuename            | venueseats
---------+--------------------------------+------------
83 | FedExField                     |      91704
 6 | New York Giants Stadium        |      80242
79 | Arrowhead Stadium              |      79451
78 | INVESCO Field                  |      76125
69 | Dolphin Stadium                |      74916
67 | Ralph Wilson Stadium           |      73967
76 | Jacksonville Municipal Stadium |      73800
89 | Bank of America Stadium        |      73298
72 | Cleveland Browns Stadium       |      73200
86 | Lambeau Field                  |      72922
...
(57 rows)
```

Contoh ini memilih pengguna (USERID) dari tabel USERS yang suka musik rock:

```
select userid from users where likerock = 't' order by 1 limit 5;

userid
--------
3
5
6
13
16
(5 rows)
```

Contoh ini memilih pengguna (USERID) dari tabel USERS di mana tidak diketahui apakah mereka suka musik rock:

```
select firstname, lastname, likerock
from users
where likerock is unknown
order by userid limit 10;

firstname | lastname | likerock
----------+----------+----------
Rafael    | Taylor   |
Vladimir  | Humphrey |
Barry     | Roy      |
Tamekah   | Juarez   |
Mufutau   | Watkins  |
Naida     | Calderon |
Anika     | Huff     |
Bruce     | Beck     |
Mallory   | Farrell  |
Scarlett  | Mayer    |
(10 rows
```

Contoh dengan kolom TIME

Tabel contoh berikut TIME_TEST memiliki kolom TIME_VAL (tipe TIME) dengan tiga nilai dimasukkan.

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

Contoh berikut ekstrak jam dari setiap timetz_val.

```
select time_val from time_test where time_val < '3:00';
   time_val
---------------
 00:00:00.5550
 00:58:00
```

Contoh berikut membandingkan dua literal waktu.

```
select time '18:25:33.123456' = time '18:25:33.123456';
 ?column?
----------
 t
```

Contoh dengan kolom TIMETZ

Tabel contoh berikut TIMETZ_TEST memiliki kolom TIMETZ_VAL (tipe TIMETZ) dengan tiga nilai dimasukkan.

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

Contoh berikut memilih hanya nilai-nilai TIMETZ kurang dari`3:00:00` UTC. Perbandingan dilakukan setelah mengubah nilai ke UTC.

```
select timetz_val from timetz_test where timetz_val < '3:00:00 UTC';

   timetz_val
---------------
 00:00:00.5550+00
```

Contoh berikut membandingkan dua literal TIMETZ. Zona waktu diabaikan untuk perbandingan.

```
select time '18:25:33.123456 PST' < time '19:25:33.123456 EST';

 ?column?
----------
 t
```

Kondisi logis

Kondisi logis menggabungkan hasil dari dua kondisi untuk menghasilkan satu hasil. Semua kondisi logis adalah operator biner dengan tipe pengembalian Boolean.

Sintaksis

```
expression
{ AND | OR }
expression
NOT expression
```

kondisi logis menggunakan logika Boolean tiga-dihargai di mana nilai null mewakili hubungan yang tidak diketahui. Tabel berikut menjelaskan hasil untuk kondisi logis, di manaE1danE2mewakili ekspresi:

E1	E2	E1 DAN E2	E1 ATAU E2	BUKAN E2
BETUL	BETUL	BETUL	BETUL	SALAH
BETUL	SALAH	SALAH	BETUL	BETUL
BETUL	TIDAK DIKETAHUI	TIDAK DIKETAHUI	BETUL	TIDAK DIKETAHUI
SALAH	BETUL	SALAH	BETUL	
SALAH	SALAH	SALAH	SALAH	
SALAH	TIDAK DIKETAHUI	SALAH	TIDAK DIKETAHUI	
TIDAK DIKETAHUI	BETUL	TIDAK DIKETAHUI	BETUL	
TIDAK DIKETAHUI	SALAH	SALAH	TIDAK DIKETAHUI	
TIDAK DIKETAHUI	TIDAK DIKETAHUI	TIDAK DIKETAHUI	TIDAK DIKETAHUI	

Operator NOT dievaluasi sebelum AND, dan operator AND dievaluasi sebelum operator OR. Setiap tanda kurung yang digunakan dapat menimpa urutan default evaluasi ini.

Contoh

Contoh berikut mengembalikan USERID dan USERNAME dari tabel USERS di mana pengguna menyukai Las Vegas dan olahraga:

```
select userid, username from users
where likevegas = 1 and likesports = 1
order by userid;

userid | username
--------+----------
1 | JSG99FHE
67 | TWU10MZT
87 | DUF19VXU
92 | HYP36WEQ
109 | FPL38HZK
120 | DMJ24GUZ
123 | QZR22XGQ
130 | ZQC82ALK
133 | LBN45WCH
144 | UCX04JKN
165 | TEY68OEB
169 | AYQ83HGO
184 | TVX65AZX
...
(2128 rows)
```

Contoh berikutnya mengembalikan USERID dan USERNAME dari tabel USERS di mana pengguna suka Las Vegas, atau olahraga, atau keduanya. Query ini mengembalikan semua output dari contoh sebelumnya ditambah pengguna yang suka hanya Las Vegas atau olahraga.

```
select userid, username from users
where likevegas = 1 or likesports = 1
order by userid;

userid | username
--------+----------
1 | JSG99FHE
2 | PGL08LJI
3 | IFT66TXU
5 | AEB55QTM
6 | NDQ15VBM
```

```
9 | MSD36KVR
10 | WKW41AIW
13 | QTF33MCG
15 | OWU78MTR
16 | ZMG93CDD
22 | RHT62AGI
27 | KOY02CVE
29 | HUH27PKK
...
(18968 rows)
```

Query berikut menggunakan tanda kurung di sekitar0Rkondisi untuk menemukan tempat di New York atau California di mana Macbeth dilakukan:

```
select distinct venuename, venuecity
from venue join event on venue.venueid=event.venueid
where (venuestate = 'NY' or venuestate = 'CA') and eventname='Macbeth'
order by 2,1;

venuename                |   venuecity
-----------------------------------------+---------------
Geffen Playhouse                         | Los Angeles
Greek Theatre                            | Los Angeles
Royce Hall                               | Los Angeles
American Airlines Theatre                | New York City
August Wilson Theatre                    | New York City
Belasco Theatre                          | New York City
Bernard B. Jacobs Theatre                | New York City
...
```

Menghapus tanda kurung dalam contoh ini mengubah logika dan hasil query.

Contoh berikut menggunakanN0Toperator:

```
select * from category
where not catid=1
order by 1;

catid | catgroup |  catname  |                 catdesc
-------+----------+-----------+--------------------------------------------
2 | Sports   | NHL       | National Hockey League
3 | Sports   | NFL       | National Football League
4 | Sports   | NBA       | National Basketball Association
```

```
5 | Sports   | MLS       | Major League Soccer
...
```

Contoh berikut menggunakanN0Tkondisi diikuti olehANDkondisi:

```
select * from category
where (not catid=1) and catgroup='Sports'
order by catid;

catid | catgroup | catname |             catdesc
-------+----------+---------+---------------------------------
2 | Sports   | NHL     | National Hockey League
3 | Sports   | NFL     | National Football League
4 | Sports   | NBA     | National Basketball Association
5 | Sports   | MLS     | Major League Soccer
(4 rows)
```

Kondisi pencocokan pola

Sebuah operator pola-pencocokan mencari string untuk pola yang ditentukan dalam ekspresi kondisional dan mengembalikan true atau false tergantung pada apakah ia menemukan kecocokan.AWS Clean Roomsmenggunakan metode berikut untuk pencocokan pola:

- EKSPRESI SEPERTI

  Operator LIKE membandingkan ekspresi string, seperti nama kolom, dengan pola yang menggunakan karakter wildcard%(persen) dan_(garis bawah). Seperti pencocokan pola selalu mencakup seluruh string. LIKE melakukan pertandingan case-sensitive dan ILIKE melakukan pertandingan case-sensitive.

- MIRIP DENGAN ekspresi reguler

  Operator MIRIP DENGAN cocok dengan ekspresi string dengan pola ekspresi reguler standar SQL, yang dapat mencakup satu set metakarakter pencocokan pola yang mencakup dua didukung oleh operator LIKE. MIRIP DENGAN cocok dengan seluruh string dan melakukan pertandingan case-sensitive.

Topik

- SUKA

- SIMILAR TO

SUKA

Operator LIKE membandingkan ekspresi string, seperti nama kolom, dengan pola yang menggunakan karakter wildcard% (persen) dan _ (garis bawah). Seperti pencocokan pola selalu mencakup seluruh string. Untuk mencocokkan urutan di mana saja dalam string, pola harus dimulai dan diakhiri dengan tanda persen.

LIKE adalah case-sensitive; ILIKE adalah case-sensitive.

Sintaksis

```
expression [ NOT ] LIKE | ILIKE pattern [ ESCAPE 'escape_char' ]
```

Pendapat

ekspresi

Ekspresi karakter UTF-8 yang valid, seperti nama kolom.

SUKA | ILIKE

LIKE melakukan pencocokan pola case-sensitive. ILIKE melakukan pencocokan pola case-sensitive untuk karakter single-byte UTF-8 (ASCII). Untuk melakukan pencocokan pola case-sensitive untuk karakter multibyte, gunakanMENURUNKANfungsi padaekspresidanpoladengan kondisi LIKE.

Berbeda dengan predikat perbandingan, seperti = dan <>, PREDIKAT LIKE dan ILIKE tidak secara implisit mengabaikan spasi belakang. Untuk mengabaikan spasi tambahan, gunakan RTRIM atau secara eksplisit mentransmisikan kolom CHAR ke VARCHAR.

Yang~~operator setara dengan LIKE, dan~~*setara dengan ILIKE. Juga!~~dan!~~*operator setara dengan NOT LIKE dan NOT ILIKE.

pola

Ekspresi karakter UTF-8 yang valid dengan pola yang akan dicocokkan.

escape_char

Sebuah ekspresi karakter yang akan melarikan diri metakarakter karakter dalam pola. Defaultnya adalah dua garis miring terbalik ('\\').

Jikapolatidak mengandung metakarakter, maka pola hanya mewakili string itu sendiri; dalam hal seperti bertindak sama dengan operator sama.

Salah satu ekspresi karakter dapat CHAR atau VARCHAR tipe data. Jika mereka berbeda,AWS Clean Roomsberubahpolake tipe dataekspresi.

LIKE mendukung metakarakter pencocokan pola berikut:

Operasi	Deskripsi
%	Cocok dengan urutan nol atau lebih karakter.
_	Cocok setiap karakter tunggal.

Contoh

Tabel berikut menunjukkan contoh pencocokan pola menggunakan LIKE:

Ekspresi	Pengembalian
'abc' LIKE 'abc'	Benar
'abc' LIKE 'a%'	Benar
'abc' LIKE '_B_'	Salah
'abc' ILIKE '_B_'	Benar
'abc' LIKE 'c%'	Salah

Contoh berikut menemukan semua kota yang namanya dimulai dengan “E”:

```
select distinct city from users
where city like 'E%' order by city;
city
---------------
East Hartford
East Lansing
East Rutherford
```

```
East St. Louis
Easthampton
Easton
Eatontown
Eau Claire
...
```

Contoh berikut menemukan pengguna yang nama belakangnya berisi “sepuluh”:

```
select distinct lastname from users
where lastname like '%ten%' order by lastname;
lastname
-------------
Christensen
Wooten
...
```

Contoh berikut menemukan kota yang karakter ketiga dan keempat adalah “ea”. Perintah menggunakan ILIKE untuk menunjukkan ketidakpekaan kasus:

```
select distinct city from users where city ilike '__EA%' order by city;
city
-------------
Brea
Clearwater
Great Falls
Ocean City
Olean
Wheaton
(6 rows)
```

Contoh berikut menggunakan string escape default (\\) untuk mencari string yang mencakup “start_” (teks`start`diikuti oleh garis bawah_):

```
select tablename, "column" from my_table_def


where "column" like '%start\\_%'
limit 5;

     tablename     |    column
-------------------+---------------
```

```
 my_s3client      | start_time
 my_tr_conflict   | xact_start_ts
 my_undone        | undo_start_ts
 my_unload_log    | start_time
 my_vacuum_detail | start_row
(5 rows)
```

Contoh berikut menentukan '^' sebagai karakter escape, kemudian menggunakan karakter escape untuk mencari string yang mencakup “start_” (teksstartdiikuti oleh garis bawah_):

```
select tablename, "column" from my_table_def

where "column" like '%start^_%' escape '^'
limit 5;

     tablename     |    column
-------------------+---------------
 my_s3client      | start_time
 my_tr_conflict   | xact_start_ts
 my_undone        | undo_start_ts
 my_unload_log    | start_time
 my_vacuum_detail | start_row
(5 rows)
```

Contoh berikut menggunakan~~*operator untuk melakukan case-sensitive (ILIKE) mencari kota-kota yang dimulai dengan “Ag”.

```
select distinct city from users where city ~~* 'Ag%' order by city;

city
------------
Agat
Agawam
Agoura Hills
Aguadilla
```

SIMILAR TO

Operator MIRIP TO cocok dengan ekspresi string, seperti nama kolom, dengan pola ekspresi reguler standar SQL. Pola ekspresi reguler SQL dapat mencakup satu set metakarakter pencocokan pola, termasuk dua yang didukung olehSUKAoperator.

The MIRIP dengan operator mengembalikan true hanya jika polanya cocok dengan seluruh string, tidak seperti POSIX perilaku ekspresi reguler, di mana pola dapat mencocokkan setiap bagian dari string.

MIRIP DENGAN melakukan pertandingan case-sensitive.

> **Note**
>
> Pencocokan ekspresi reguler menggunakan MIRIP TO adalah komputasi mahal. Sebaiknya gunakan LIKE bila memungkinkan, terutama saat memproses sejumlah besar baris. Misalnya, kueri berikut identik secara fungsional, tetapi kueri yang menggunakan LIKE berjalan beberapa kali lebih cepat daripada kueri yang menggunakan ekspresi reguler:
>
> ```
> select count(*) from event where eventname SIMILAR TO '%(Ring|Die)%';
> select count(*) from event where eventname LIKE '%Ring%' OR eventname LIKE '%Die
> %';
> ```

Sintaksis

```
expression [ NOT ] SIMILAR TO pattern [ ESCAPE 'escape_char' ]
```

Pendapat

ekspresi

Ekspresi karakter UTF-8 yang valid, seperti nama kolom.

SIMILAR TO

MIRIP DENGAN melakukan kecocokan pola case-sensitive untuk seluruh string diekspresi.

pola

Ekspresi karakter UTF-8 yang valid mewakili pola ekspresi reguler standar SQL.

escape_char

Sebuah ekspresi karakter yang akan melarikan diri metakarakter dalam pola. Defaultnya adalah dua garis miring terbalik ('\\').

Jikapolatidak mengandung metakarakter, maka pola hanya mewakili string itu sendiri.

Salah satu ekspresi karakter dapat CHAR atau VARCHAR tipe data. Jika mereka berbeda,AWS Clean Roomsberubahpolake tipe dataekspresi.

MIRIP DENGAN mendukung metakarakter pencocokan pola berikut:

Operasi	Deskripsi
`%`	Cocok dengan urutan nol atau lebih karakter.
`_`	Cocok setiap karakter tunggal.
`|`	Menunjukkan pergantian (salah satu dari dua alternatif).
`*`	Ulangi item sebelumnya nol atau lebih kali.
`+`	Ulangi item sebelumnya satu kali atau lebih.
`?`	Ulangi item sebelumnya nol atau satu kali.
`{m}`	Ulangi item sebelumnya dengan tepatmkali.
`{m,}`	Ulangi item sebelumnyamatau lebih banyak kali.
`{m,n}`	Ulangi item sebelumnya setidaknyamdan tidak lebih darinkali.
`()`	Kurung item kelompok ke dalam item logis tunggal.
`[...]`	Sebuah ekspresi braket menentukan kelas karakter, seperti dalam ekspresi reguler POSIX.

Contoh

Tabel berikut menunjukkan contoh pencocokan pola menggunakan MIRIP DENGAN:

Ekspresi	Pengembalian
`'abc' SIMILAR TO 'abc'`	Benar
`'abc' SIMILAR TO '_b_'`	Benar

Ekspresi	Pengembalian
`'abc' SIMILAR TO '_A_'`	Salah
`'abc' SIMILAR TO '%(b|d)%'`	Benar
`'abc' SIMILAR TO '(b|c)%'`	Salah
`'AbcAbcdefgefg12efgefg12' SIMILAR TO '((Ab)?c)+d((efg)+(12))+'`	Benar
`'aaaaaab11111xy' SIMILAR TO 'a{6}_[0-9]{5}(x|y){2}'`	Benar
`'$0.87' SIMILAR TO '$[0-9]+(.[0-9][0-9])?'`	Benar

Contoh berikut menemukan kota yang namanya mengandung “E” atau “H”:

```
SELECT DISTINCT city FROM users
WHERE city SIMILAR TO '%E%|%H%' ORDER BY city LIMIT 5;

      city
-----------------
 Agoura Hills
 Auburn Hills
 Benton Harbor
 Beverly Hills
 Chicago Heights
```

Contoh berikut menggunakan string escape default ('\\') untuk mencari string yang mencakup”_“:

```
SELECT tablename, "column" FROM my_table_def
WHERE "column" SIMILAR TO '%start\\_%'

ORDER BY tablename, "column" LIMIT 5;

        tablename         |       column
--------------------------+---------------------
 my_abort_idle          | idle_start_time
 my_abort_idle          | txn_start_time
```

```
 my_analyze_compression | start_time
 my_auto_worker_levels  | start_level
 my_auto_worker_levels  | start_wlm_occupancy
```

Contoh berikut menentukan '^'sebagai escape string, kemudian menggunakan escape string untuk mencari string yang menyertakan”_“:

```
SELECT tablename, "column" FROM my_table_def

WHERE "column" SIMILAR TO '%start^_%' ESCAPE '^'
ORDER BY tablename, "column" LIMIT 5;

        tablename         |       column
--------------------------+---------------------
 stcs_abort_idle          | idle_start_time
 stcs_abort_idle          | txn_start_time
 stcs_analyze_compression | start_time
 stcs_auto_worker_levels  | start_level
 stcs_auto_worker_levels  | start_wlm_occupancy
```

ANTARA kondisi rentang

SEBUAHBETWEENkondisi tes ekspresi untuk dimasukkan dalam berbagai nilai, menggunakan kata kunciBETWEENdanAND.

Sintaksis

```
expression [ NOT ] BETWEEN expression AND expression
```

Ekspresi dapat numerik, karakter, atau datetime tipe data, tetapi mereka harus kompatibel. Rentang ini inklusif.

Contoh

Contoh pertama menghitung berapa banyak transaksi terdaftar penjualan baik 2, 3, atau 4 tiket:

```
select count(*) from sales
where qtysold between 2 and 4;
```

```
count
--------
104021
(1 row)
```

Kondisi rentang mencakup nilai awal dan akhir.

```
select min(dateid), max(dateid) from sales
where dateid between 1900 and 1910;

min  | max
-----+-----
1900 | 1910
```

Ekspresi pertama dalam kondisi rentang harus nilai yang lebih rendah dan ekspresi kedua nilai yang lebih besar. Contoh berikut akan selalu kembali nol baris karena nilai-nilai ekspresi:

```
select count(*) from sales
where qtysold between 4 and 2;

count
-------
0
(1 row)
```

Namun, menerapkan pengubah NOT akan membalikkan logika dan menghasilkan hitungan semua baris:

```
select count(*) from sales
where qtysold not between 4 and 2;

count
--------
172456
(1 row)
```

Query berikut mengembalikan daftar tempat dengan 20000-50000 kursi:

```
select venueid, venuename, venueseats from venue
where venueseats between 20000 and 50000
order by venueseats desc;
```

```
venueid |       venuename               | venueseats
---------+-------------------------------+------------
116 | Busch Stadium                 |      49660
106 | Rangers BallPark in Arlington |      49115
96 | Oriole Park at Camden Yards   |      48876
...
(22 rows)
```

Contoh berikut menunjukkan menggunakan ANTARA untuk nilai tanggal:

```
select salesid, qtysold, pricepaid, commission, saletime
from sales
where eventid between 1000 and 2000
   and saletime between '2008-01-01' and '2008-01-03'
order by saletime asc;

salesid | qtysold | pricepaid | commission |   saletime
--------+---------+-----------+------------+---------------
  65082 |       4 |       472 |       70.8 | 1/1/2008 06:06
 110917 |       1 |       337 |      50.55 | 1/1/2008 07:05
 112103 |       1 |       241 |      36.15 | 1/2/2008 03:15
 137882 |       3 |      1473 |     220.95 | 1/2/2008 05:18
  40331 |       2 |        58 |        8.7 | 1/2/2008 05:57
 110918 |       3 |      1011 |     151.65 | 1/2/2008 07:17
  96274 |       1 |       104 |       15.6 | 1/2/2008 07:18
 150499 |       3 |       135 |      20.25 | 1/2/2008 07:20
  68413 |       2 |       158 |       23.7 | 1/2/2008 08:12
```

Perhatikan bahwa meskipun rentang BETWEEN inklusif, tanggal default memiliki nilai waktu 00:00:00. Hanya berlaku 3 Januari baris untuk query sampel akan baris dengan saletime`1/3/2008 00:00:00`.

Kondisi nol

YangNULLtes kondisi untuk nulls, ketika nilai hilang atau tidak diketahui.

Sintaksis

```
expression IS [ NOT ] NULL
```

Pendapat

ekspresi

Ekspresi apa pun seperti kolom.

IS NULL

Benar ketika nilai ekspresi adalah null dan false ketika memiliki nilai.

IS NOT NULL

Adalah palsu ketika nilai ekspresi adalah null dan benar ketika memiliki nilai.

Contoh

Contoh ini menunjukkan berapa kali tabel SALES berisi null di bidang QTYSOLD:

```
select count(*) from sales
where qtysold is null;
count
-------
0
(1 row)
```

Kondisi EXISTS

EXISTS kondisi tes untuk keberadaan baris dalam subquery, dan kembali benar jika subquery mengembalikan setidaknya satu baris. Jika NOT ditentukan, kondisi mengembalikan true jika subquery mengembalikan tidak ada baris.

Sintaksis

```
[ NOT ] EXISTS (table_subquery)
```

Pendapat

ADA

Apakah benar ketikatable_subquerymengembalikan setidaknya satu baris.

TIDAK ADA

Apakah benar ketikatable_subquerymengembalikan tidak ada baris.

table_subquery

Sebuah subquery yang mengevaluasi ke tabel dengan satu atau lebih kolom dan satu atau lebih baris.

Contoh

Contoh ini mengembalikan semua pengidentifikasi tanggal, satu kali masing-masing, untuk setiap tanggal yang memiliki penjualan apapun:

```
select dateid from date
where exists (
select 1 from sales
where date.dateid = sales.dateid
)
order by dateid;

dateid
--------
1827
1828
1829
...
```

Dalam kondisi

SebuahINkondisi tes nilai untuk keanggotaan dalam satu set nilai atau dalam subquery.

Sintaksis

```
expression [ NOT ] IN (expr_list | table_subquery)
```

Pendapat

ekspresi

Sebuah numerik, karakter, atau ekspresi datetime yang dievaluasi terhadapexpr_listatautable_subquerydan harus kompatibel dengan tipe data dari daftar itu atau subquery.

expr_list

Satu atau lebih ekspresi dipisahkan koma, atau satu atau lebih set ekspresi dipisahkan koma dibatasi oleh tanda kurung.

table_subquery

Subquery yang mengevaluasi ke tabel dengan satu atau lebih baris, tetapi terbatas hanya satu kolom dalam daftar pilihannya.

IN | TIDAK DI

IN mengembalikan nilai true jika ekspresi adalah anggota dari daftar ekspresi atau query. NOT IN mengembalikan nilai true jika ekspresi bukan anggota. IN dan TIDAK DALAM kembali NULL dan tidak ada baris dikembalikan dalam kasus berikut: Jikaekspresimenghasilkan nol; atau jika tidak ada yang cocokexpr_listatautable_subquerynilai dan setidaknya satu dari baris perbandingan ini menghasilkan nol.

Contoh

Kondisi berikut hanya berlaku untuk nilai-nilai yang tercantum:

```
qtysold in (2, 4, 5)
date.day in ('Mon', 'Tues')
date.month not in ('Oct', 'Nov', 'Dec')
```

Optimalisasi untuk Daftar IN Besar

Untuk mengoptimalkan kinerja kueri, daftar IN yang mencakup lebih dari 10 nilai dievaluasi secara internal sebagai larik skalar. IN daftar dengan kurang dari 10 nilai dievaluasi sebagai serangkaian predikat OR. Optimasi ini didukung untuk tipe data SMALLINT, INTEGER, BIGINT, REAL, DOUBLE PRECISION, BOOLEAN, CHAR, VARCHAR, DATE, TIMESTAMP, dan TIMESTAMPTZ.

Lihatlah output MENJELASKAN untuk query untuk melihat efek optimasi ini. Misalnya:

```
explain select * from sales
QUERY PLAN
------------------------------------------------------------------------
XN Seq Scan on sales  (cost=0.00..6035.96 rows=86228 width=53)
Filter: (salesid = ANY ('{1,2,3,4,5,6,7,8,9,10,11}'::integer[]))
(2 rows)
```

Sintaks

```
comparison_condition
| logical_condition
| range_condition
| pattern_matching_condition
| null_condition
| EXISTS_condition
| IN_condition
```

Meneri data bersarang

AWS Clean Roomsmenawarkan akses yang kompatibel dengan SQL ke data relasional dan bersarang.

AWS Clean Roomsmenggunakan notasi putus-putus dan subskrip array untuk navigasi jalur saat mengakses data bersarang. Hal ini juga memungkinkanFROMitem klausa untuk mengulangi array dan digunakan untuk operasi unnest. Topik berikut memberikan deskripsi pola kueri berbeda yang menggabungkan penggunaan tipe data array/struct/map dengan navigasi jalur dan array, unnesting, dan gabungan.

Topik

Navigasi

AWS Clean Roomsmemungkinkan navigasi ke dalam array dan struktur menggunakan`[...]`braket dan notasi titik masing-masing. Selanjutnya, Anda dapat mencampur navigasi ke dalam struktur menggunakan notasi titik dan array menggunakan notasi braket.

Example

Sebagai contoh, kueri berikut dalam kueri berikut dalam`c_orders`kolom data array adalah array dengan struktur dan atribut diberi nama`o_orderkey`.

```
SELECT cust.c_orders[0].o_orderkey FROM customer_orders_lineitem AS cust;
```

Anda dapat menggunakan notasi titik dan tanda kurung di semua jenis kueri, seperti pemfilteran, gabungan, dan agregasi. Anda dapat menggunakan notasi ini dalam kueri di mana biasanya ada referensi kolom.

Example

Contoh berikut menggunakan pernyataan SELECT yang memfilter hasil.

```
SELECT count(*) FROM customer_orders_lineitem WHERE c_orders[0].o_orderkey IS NOT NULL;
```

Example

Contoh berikut menggunakan braket dan navigasi titik di kedua klausa GROUP BY dan ORDER BY.

```
SELECT c_orders[0].o_orderdate,
       c_orders[0].o_orderstatus,
       count(*)
FROM customer_orders_lineitem
WHERE c_orders[0].o_orderkey IS NOT NULL
GROUP BY c_orders[0].o_orderstatus,
         c_orders[0].o_orderdate
ORDER BY c_orders[0].o_orderdate;
```

Kueri yang tidak bersarang

Untuk kueri unnest,AWS Clean Roomsmemungkinkan iterasi melalui array. Hal ini dilakukan dengan menavigasi array menggunakan klausa FROM dari query.

Example

Menggunakan contoh sebelumnya, contoh berikut iterasi atas nilai atribut untuk`c_orders`.

```
SELECT o FROM customer_orders_lineitem c, c.c_orders o;
```

Sintaks unnesting adalah perpanjangan dari klausa FROM. Dalam SQL standar, klausa FROM`x (AS) y`berarti bahwayiterasi atas setiap tupel dalam kaitannya`x`. Dalam hal ini,`x`mengacu pada hubungan danymengacu pada alias untuk relasi`x`. Demikian pula, sintaks unnesting menggunakan item klausa FROM`x (AS) y`berarti bahwayiterasi atas setiap nilai dalam ekspresi array`x`. Dalam hal ini,`x`adalah ekspresi array danyadalah alias untuk`x`.

Operan kiri juga dapat menggunakan notasi titik dan braket untuk navigasi reguler.

Example

Dalam contoh sebelumnya:

- `customer_orders_lineitem c`adalah iterasi atas`customer_order_lineitem`meja dasar

- `c.c_orders` oadalah iterasi atas`c.c_orders array`

Untuk mengulangio`_lineitems`atribut, yang merupakan array dalam array, Anda menambahkan beberapa klausa.

```
SELECT o, l FROM customer_orders_lineitem c, c.c_orders o, o.o_lineitems l;
```

AWS Clean Roomsjuga mendukung indeks array saat mengulangi array menggunakanATkata kunci. Klausul`x AS y AT z`iterasi di atas array`x`dan menghasilkan bidang`z`, yang merupakan indeks array.

Example

Sebagai contoh berikut: bagaimana indeks array bekerja.

```
SELECT c_name,
       orders.o_orderkey AS orderkey,
       index AS orderkey_index
FROM customer_orders_lineitem c, c.c_orders AS orders AT index
ORDER BY orderkey_index;
c_name             | orderkey | orderkey_index
-------------------+----------+----------------
Customer#000008251 | 3020007  |        0
Customer#000009452 | 4043971  |        0  (2 rows)
```

Example

Contoh berikut iterasi atas array skalar.

```
CREATE TABLE bar AS SELECT json_parse('{"scalar_array": [1, 2.3, 45000000]}') AS data;

SELECT index, element FROM bar AS b, b.data.scalar_array AS element AT index;

 index | element
-------+----------
     0 | 1
1 | 2.3
2 | 45000000
(3 rows)
```

Example

Contoh berikut iterasi atas array dari beberapa level. Contoh menggunakan beberapa klausa unnest untuk beralih ke array terdalam. The`f.multi_level_array` ASarray iterasi`multi_level_array`. ArrayASelemen adalah iterasi atas array di dalam`multi_level_array`.

```
CREATE TABLE foo AS SELECT json_parse('[[1.1, 1.2], [2.1, 2.2], [3.1, 3.2]]') AS
 multi_level_array;

SELECT array, element FROM foo AS f, f.multi_level_array AS array, array AS element;

   array   | element
-----------+---------
 [1.1,1.2] | 1.1
 [1.1,1.2] | 1.2
 [2.1,2.2] | 2.1
 [2.1,2.2] | 2.2
 [3.1,3.2] | 3.1
 [3.1,3.2] | 3.2
(6 rows)
```

Semantik longgar

Secara default, operasi navigasi pada nilai data bersarang mengembalikan null alih-alih mengembalikan kesalahan saat navigasi tidak valid. Navigasi objek tidak valid jika nilai data bersarang bukan objek atau jika nilai data bersarang adalah objek tetapi tidak berisi nama atribut yang digunakan dalam kueri.

Example

Misalnya, kueri berikut mengakses nama atribut yang tidak valid di kolom data bersarang`c_orders`:

```
SELECT c.c_orders.something FROM customer_orders_lineitem c;
```

Navigasi array mengembalikan null jika nilai data bersarang bukan array atau indeks array di luar batas.

Example

Hasil kueri berikut karena`c_orders[1][1]`berada di luar batas.

```
SELECT c.c_orders[1][1] FROM customer_orders_lineitem c;
```

Jenis introspeksi

Kolom tipe data bersarang mendukung fungsi inspeksi yang mengembalikan tipe dan informasi tipe lainnya tentang nilai. AWS Clean Rooms mendukung fungsi boolean berikut untuk kolom data bersarang:

- DESIMAL_PRESISI
- SKALA DESIMAL
- IS_ARRAY
- IS_BIGINT
- IS_CHAR
- ADALAH_DESIMAL
- IS_MENGAPUNG
- IS_INTEGER
- IS_OBJEK
- IS_SKALAR
- IS_SMALLINT
- IS_VARCHAR
- JSON_TYPEOF

Semua fungsi ini mengembalikan false jika nilai input adalah null. IS_SCALAR, IS_OBJECT, dan IS_ARRAY saling eksklusif dan mencakup semua nilai yang mungkin kecuali untuk null. Untuk menyimpulkan jenis yang sesuai dengan data,AWS Clean Roomsmenggunakan fungsi JSON_TYPEOF yang mengembalikan tipe (tingkat atas) nilai data bersarang seperti yang ditunjukkan pada contoh berikut:

```
SELECT JSON_TYPEOF(r_nations) FROM region_nations;
 json_typeof
 -------------
array
(1 row)
```

```
SELECT JSON_TYPEOF(r_nations[0].n_nationkey) FROM region_nations;
 json_typeof
 -------------
 number
```

Riwayat dokumen untuk AWS Clean Rooms Referensi SQL

Tabel berikut menjelaskan rilis dokumentasi untuk Referensi AWS Clean Rooms SQL.

Untuk notifikasi tentang pembaruan-pembaruan dokumentasi ini, Anda dapat berlangganan ke sebuah umpan RSS. Untuk berlangganan pembaruan RSS, Anda harus mengaktifkan plug-in RSS untuk browser yang Anda gunakan.

Perubahan	Deskripsi	Tanggal
Perintah SQL dan fungsi SQL - perbarui	Contoh telah ditambahkan untuk klausa JOIN, KECUALI set operator, ekspresi bersyarat CASE, dan fungsi berikut: ANY_VALUE, NVL dan COALESCE, NULLIF, CAST, CONVERT, CONVERT_TIMEZONE, EXTRACT, MOD, SIGN, CONCAT, FIRST_VALUE, dan LAST_VALUE.	Februari 28, 2024
Fungsi SQL - pembaruan	AWS Clean Rooms sekarang mendukung fungsi SQL berikut: Array, SUPER, dan VARBYTE. Fungsi matematika berikut sekarang didukung: ACOS, ASIN, ATAN, ATAN2, COT, DEXP, PI, POW, RADIANS, dan SIN. Fungsi JSON berikut sekarang didukung: CAN_JSON_PARSE, JSON_PARSE, dan JSON_SERIALIZE.	Oktober 6, 2023

Dukungan tipe data bersarang	AWS Clean Rooms sekarang mendukung tipe data bersarang.	Agustus 30, 2023
Aturan penamaan SQL - perbarui	Perubahan hanya dokumentasi untuk memperjelas nama kolom yang dicadangkan.	16 Agustus 2023
Ketersediaan umum	Referensi AWS Clean Rooms SQL sekarang tersedia secara umum.	31 Juli 2023

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.