aws

开发人员指南

AWS Glue DataBrew

AWS Glue DataBrew: 开发人员指南

Table of Contents

什么是 AWS Glue DataBrew？

AWS Glue DataBrew 是一种可视化数据准备工具，让用户无需编写任何代码即可清理数据并实现标准化。与定制开发的数据准备相比，使用 DataBrew 可将准备用于分析和机器学习 (ML) 的数据所需的时间缩短多达 80%。您可以从 250 多种现成的转换功能中进行选择，以自动执行数据准备任务，例如筛选异常、将数据转换为标准格式以及更正无效值。

通过使用 DataBrew，业务分析师、数据科学家和数据工程师可以更轻松地进行协作，从原始数据中获取见解。由于 DataBrew 是无服务器的，因此无论您的技术水平如何，您都可以浏览和转换数 TB 的原始数据，而无需创建集群或管理任何基础架构。

借助直观的 DataBrew 界面，您可以交互式地发现、可视化、清理和转换原始数据。 DataBrew 提出明智的建议，帮助您识别可能难以发现且修复耗时的数据质量问题。 DataBrew 准备数据后，您可以利用自己的时间对结果采取行动，并更快地进行迭代。您可以将转换保存为配方中的步骤，以后可以对其进行更新或在其他数据集中重复使用，并持续部署。

下图显示了高级 DataBrew 工作原理。

要使用 DataBrew，您需要创建一个项目并连接到您的数据。在项目工作区中，您可以看到您的数据显示在类似网格的可视界面中。在这里，您可以浏览数据并查看价值分布和图表，以了解其概况。

要准备数据，您可以从 250 多种 point-and-click 转换中进行选择。其中包括删除空值、替换缺失值、修复架构不一致、基于函数创建列等等。您还可以使用转换来应用自然语言处理 (NLP) 技术将句子拆

分为短语。即时预览会显示转换前后的部分数据，因此您可以在将配方应用于整个数据集之前对其进行修改。

DataBrew 在数据集上运行配方后，输出内容将存储在 Amazon Simple Storage Service (Amazon Simple S3) 中。将经过清理、准备好的数据集存入 Amazon S3 后，您的其他数据存储或数据管理系统可以将其摄取。

中的核心概念和术语 AWS Glue DataBrew

下面，您可以在中找到核心概念和术语的概述 AWS Glue DataBrew。在阅读本节后，请参阅入门 AWS Glue DataBrew，该节将指导您完成创建项目、连接数据集和运行作业的过程。

主题

- 项目
- 数据集
- 配方
- 作业
- 数据沿袭
- 数据分析文件

项目

中的交互式数据准备工作区 DataBrew 称为项目。使用数据项目，您可以管理一系列相关项目：数据、转换和计划流程。在创建项目时，您可以选择或创建要处理的数据集。接下来，你创建一个食谱，这是一组你 DataBrew 要执行的说明或步骤。这些操作会将您的原始数据转换为可供数据管道使用的表单。

数据集

数据集只是指一组数据，即分为列或字段的行或记录。创建 DataBrew 项目时，您可以连接或上传要转换或准备的数据。 DataBrew 可以处理来自任何来源、从格式化文件导入的数据，并且可以直接连接到越来越多的数据存储列表。

对于 DataBrew，数据集是指与您的数据的只读连接。 DataBrew 收集一组描述性元数据以引用数据。任何实际数据都不能被修改或存储 DataBrew。为简单起见，我们使用数据集来指代实际的数据集和 DataBrew 使用的元数据。

配方

在中 DataBrew，配方是您 DataBrew 要处理的数据的一组说明或步骤。一个配方可以包含多个步骤，每个步骤可以包含许多操作。您可以使用工具栏上的转换工具来设置要对数据进行的所有更改。稍后，当你准备好查看食谱的成品时，你可以将这项工作分配给 DataBrew 并安排时间。 DataBrew 存储有关数据转换的说明，但它不存储您的任何实际数据。您可以在其他项目中下载和重复使用配方。您还可以发布多个食谱版本。

作业

DataBrew 通过运行你在制作食谱时设置的指令，负责转换数据。运行这些指令的过程称为作业。作业可以根据预设的时间表将您的数据配方付诸实践。但是你并不局限于日程安排。您还可以按需运行作业。如果您想分析一些数据，则不需要配方。在这种情况下，您只需设置配置文件作业即可创建数据配置文件。

数据沿袭

DataBrew 在可视化界面中跟踪您的数据以确定其来源，称为数据谱系。此视图向您展示了数据如何通过不同实体与其最初来源的不同实体流动。你可以看到它的起源、它受到影响的其他实体、它随着时间的推移发生了什么以及它被存储在哪里。

数据分析文件

当您对数据进行分析时， DataBrew 会创建一个名为数据配置文件的报告。此摘要向您介绍数据的现有形状，包括内容的上下文、数据的结构及其关系。您可以通过运行数据配置文件作业为任何数据集创建数据配置文件。

产品和服务集成

使用本部分了解哪些产品和服务与哪些产品和服务集成 DataBrew。

DataBrew 可与以下联网、管理和治理 AWS 服务配合使用：

- Amazon CloudFront
- AWS CloudFormation
- AWS CloudTrail
- Amazon CloudWatch

- AWS Step Functions

DataBrew 适用于以下 AWS 数据湖和数据存储：

- AWS Lake Formation
- Amazon S3

DataBrew 支持以下文件格式和扩展名用于上传数据。

格式	文件扩展名（可选）	压缩文件的扩展名（必填）
逗号分隔的值	.csv	.gz .snappy .lz4 .bz2 .deflate
Micrososoft	.xlsx	不支持压缩
JSON（JSON 文档和 JSON 行）	.json, .jsonl	.gz .snappy .lz4 .bz2 .deflate
Apache ORC	.orc	.zlib .snappy
Apache Parquet	.parquet	.gz .snappy

格式	文件扩展名（可选）	压缩文件的扩展名（必填）
		.lz4

DataBrew 将输出文件写入 Amazon S3，并支持以下文件格式和扩展名。

格式	文件扩展名（未压缩）	文件扩展名（压缩）
逗号分隔的值	.csv	.csv.snappy ，.csv.gz, .csv.lz4, csv.bz2, .csv.deflate ，csv.br
制表符分隔的值	.csv	.tsv.snappy ，.tsv.gz, .tsv.lz4, tsv.bz2, .tsv.deflate ，tsv.br
Apache Parquet	.parquet	.parquet.snappy ， .parquet.gz ，.parquet. lz4 ，.parquet.lzo ， .parquet.br
AWS Glue Parq	不支持	.glue.parquet.snappy
Apache Avro	.avro	.avro.snappy ，.avro.gz, .avro.lz4 ，.avro.bz2 ，.avro.deflate ， .avro.br
Apache ORC	.orc	.orc.snappy ，.orc.lzo, .orc.zlib
XML	.xml	.xml.snappy ，.xml.gz, .xml.lz4, .xml.bz2, .xml.deflate ，.xml.br
JSON（仅限 JSON 行格式）	.json	.json.snappy ，.json.gz, .json.lz4 ，json.bz2,

格式	文件扩展名（未压缩）	文件扩展名（压缩）
		`.json.deflate`，`.json.br`
Tableaau	不支持	不适用

设置 AWS Glue DataBrew

在开始使用之前 AWS Glue DataBrew，你需要设置一些权限、一个用户和一个角色。首先执行以下步骤：

1. 根据需要注册 AWS 账户，并创建 AWS Identity and Access Management (IAM) 策略以使用户能够运行 DataBrew：
   - 注册新 AWS 账户并添加用户。有关更多信息，请参阅 设置一个新 AWS 账户。
   - 为控制台用户添加 IAM 策略。 具有这些权限的用户可以访问 DataBrew AWS Management Console。
   - 为 IAM 角色添加数据资源权限。 具有这些权限的 IAM 角色可以代表用户访问数据。

   您需要成为 IAM 管理员才能创建用户、角色和策略。
2. 为添加用户或群组 DataBrew。具有正确权限的用户或群组可以在控制台 DataBrew 上进行访问。
3. 添加有权访问其数据的角色 DataBrew。具有正确权限的角色可以代表用户访问数据。

设置一个新 AWS 账户

如果您没有 AWS 账户，请注册一个 AWS 账户并创建一个 IAM 管理员用户。

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

要创建管理员用户，请选择以下选项之一。

选择一种方法来管理您的管理员	目的	方式	您也可以
在 IAM Identity Center 中（建议）	使用短期凭证访问 AWS。 这符合安全最佳实操。有关最佳实践的信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。	有关说明，请参阅《AWS IAM Identity Center 用户指南》中的入门。	通过在《AWS Command Line Interface 用户指南》 AWS IAM Identity Center中配置 AWS CLI 要使用的来配置编程访问权限。
在 IAM 中（不推荐使用）	使用长期凭证访问 AWS。	按照《IAM 用户指南》中的创建您的首个 IAM 管理员用户和组的说明操作。	按照《IAM 用户指南》中的管理 IAM 用户的访问密钥，配置编程式访问。

有关更多信息，请参阅 IAM 用户指南中的以下主题：

- 什么是 IAM？
- 开始使用 IAM 进行设置
- 创建管理用户和群组（控制台）

设置 AWS CLI

如果您打算使用 JupyterLab 或 DataBrew API，请务必安装 AWS Command Line Interface (AWS CLI)。你不需要它来使用 DataBrew 控制台或执行入门练习中的步骤。

要设置 AWS CLI

1. 使用以下步骤下载并配置： AWS CLI

- 安装 AWS CLI
- 配置基础知识

2. 在命令提示符下输入以下 DataBrew 命令以验证设置。

```
aws databrew help
```

如果此语句返回错误“aws: error: argument command: Invalid choice”，后面是一长串服务，请卸载 AWS CLI，然后重新安装。此操作不会覆盖您的现有配置。

AWS CLI 命令使用配置中的默认 AWS 区域，除非您使用参数或配置文件进行设置。您可以将--region参数添加到每条命令中。

如果你愿意，可以在~/.aws/config或%UserProfile%/.aws/config（在 Microsoft Windows 上）中添加一个命名的配置文件。命名配置文件还可以保留其他设置，如以下示例所示。

```
[profile databrew]
aws_access_key_id = ACCESS-KEY-ID-OF-IAM-USER
aws_secret_access_key = SECRET-ACCESS-KEY-ID-OF-IAM-USER
region = us-east-1
output = text
```

设置 AWS Identity and Access Management (IAM) 权限

在开始之前，您需要在 IAM 中进行一些设置。你需要成为管理员或者得到管理员的帮助。但是，如果您拥有管理员访问权限的帐户，则可以自己完成这些任务。您可以在本节中找到每项任务的简单说明。

以下是您需要执行的操作的概述：

- 作为此过程的一部分，您需要添加一个用户。您不必添加新的用户，您也可以使用现有用户。您可以附加 DataBrew 权限，以便用户可以打开 DataBrew 控制台。
- 创建一个 IAM 角色。角色允许某些操作，并在使用时在限制范围内授予权限。例如，它仅适用于您 AWS 账户中的用户。您稍后可以添加更多限制。
- 创建您需要的一个或多个 IAM 策略。策略是允许用户执行的操作的列表。要创建策略，请打开另一个控制台页面，然后粘贴您下载的文件中的文本。

> **Note**
>
> 我们在这里提供的是基本的设置信息。我们建议您花点时间自定义权限，使其满足您的安全性和合规性需求。如果需要帮助，请联系您的管理员或 Supp AWS ort。

添加所需的权限

1. DataBrew 通过以下操作创建 IAM 策略以使用户能够运行：

   - 为控制台用户添加自定义 IAM 策略。如果您不需要自定义策略，则可以改为选择 AWS-managed 策略。只需在步骤 2 中将其添加到用户即可。具有这些权限的用户可以访问 DataBrew 服务控制台。
   - 为数据资源添加权限。具有这些权限的 IAM 角色可以代表用户访问数据。

   您需要是管理员才能创建用户、角色和策略。
2. 为添加用户或群组 DataBrew。具有正确权限的用户或群组可以访问 DataBrew控制台。
3. 添加一个有权访问其数据的角色 DataBrew。具有正确权限的角色可以代表用户访问数据。

设置 IAM 策略 DataBrew

您使用 IAM 策略来管理权限。使用策略可以更轻松地一次添加所有相关权限，而不是一次添加一个权限。

建议您使用我们提供的相同名称创建策略。在整篇文档中，我们在这些政策中使用以下所示的名称。如果您需要联系 Support，使用这些名称还可以更轻松地联系 AWS 。不过，您可以选择更改策略名称及其内容。有关 IAM 策略的更多信息，请参阅 I A M 用户指南中的创建客户托管策略。

创建需要使用的策略后 DataBrew，将其附加到用户和角色。本节稍后将介绍如何执行此操作。

主题

- 为控制台用户添加 IAM 策略
- 为 IAM 角色添加数据资源权限
- 配置 IAM 策略 DataBrew

为控制台用户添加 IAM 策略

为用户设置权限 AWS Management Console 是可选的，但如果您需要控制台访问权限，请先执行此步骤。

要设置控制台 DataBrew 上的访问权限，请选择下列选项之一：

- 使用由 AWS:管理的策略`AwsGlueDataBrewFullAccessPolicy`。如果选择此选项，请跳至下一个策略为 IAM 角色添加数据资源权限。
- 创建本节中描述的策略`AwsGlueDataBrewCustomUserPolicy`。使用此选项，您可以根据其他自定义安全要求自定义策略。

以下策略授予运行 DataBrew 控制台所需的权限。您通过使用 IAM 提供这些权限。

定义 DataBrew （控制台） AwsGlueDataBrewCustomUserPolicy 的 IAM 策略

1. 下载 AwsGlueDataBrewCustomUserPolicyIAM 策略的 JSON。
2. 登录到， AWS Management Console 然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
3. 在导航窗格中，选择策略。
4. 对于每个策略，选择创建策略。
5. 在 “创建策略” 屏幕上，导航到 JSON 选项卡。
6. 复制您下载的策略 JSON 语句。将其粘贴到编辑器的示例语句上。
7. 确认该政策是根据您的账户、安全要求和所需 AWS 资源定制的。如果需要进行更改，可以在编辑器中进行更改，可以在编辑器中进行更改。
8. 选择查看策略。

为 DataBrew (AWS CLI) 定义 AwsGlueDataBrewCustomUserPolicy IAM 策略

1. 下载 AwsGlueDataBrewCustomUserPolicyIAM 策略的 JSON。
2. 如前面的步骤的第一步所述，自定义策略。
3. 运行以下命令创建策略。

```
aws iam create-policy --policy-name AwsGlueDataBrewCustomUserPolicy --policy-
document file://iam-policy-AwsGlueDataBrewCustomUserPolicy.json
```

为 IAM 角色添加数据资源权限

要连接到数据，AWS Glue DataBrew 需要有一个可以代表用户传递的 IAM 角色。接下来，您会发现如何创建稍后附加到 IAM 角色的策略。

该AwsGlueDataBrewDataResourcePolicy策略授予使用连接数据所需的权限 DataBrew。对于访问其他 AWS 资源中的数据的任何操作（例如访问您在 Amazon S3 中的对象），DataBrew 需要代表您访问该资源的权限。

定义 DataBrew （控制台） AwsGlueDataBrewDataResourcePolicy 的 IAM 策略

1. 下载适用的 JSON AwsGlueDataBrewDataResourcePolicy。
2. 登录到，AWS Management Console 然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
3. 在导航窗格中，选择策略。
4. 对于每个策略，选择创建策略。
5. 在 “创建策略” 屏幕上，导航到 JSON 选项卡。
6. 复制您下载的策略 JSON 语句。将其粘贴到编辑器的示例语句上。
7. 确认该政策是根据您的账户、安全要求和所需 AWS 资源定制的。如果需要进行更改，可以在编辑器中进行更改，可以在编辑器中进行更改。
8. 选择查看策略。

为 DataBrew (AWS CLI) 定义 AwsGlueDataBrewDataResourcePolicy IAM 策略

1. 下载适用的 JSON AwsGlueDataBrewDataResourcePolicy。
2. 如前面的步骤的第一步所述，自定义策略。
3. 运行以下命令创建策略。

```
aws iam create-policy --policy-name AwsGlueDataBrewDataResourcePolicy --policy-
document file://iam-policy-AwsGlueDataBrewDataResourcePolicy.json
```

配置 IAM 策略 DataBrew

在下面，您可以找到有关可与之配合使用的 IAM 策略的示例 DataBrew。此处提供了有关基本政策的详细信息。另外，还有更多不需要使用的示例 DataBrew。它们是你可能在某些情况下使用的其他配置。

主题

AwsGlueDataBrewCustomUserPolicy

该`AwsGlueDataBrewCustomUserPolicy`策略授予使用 DataBrew 控制台所需的大部分权限。此策略中指定的某些资源引用了由使用的服务 DataBrew。这些名称包括 Amazon S3 存储桶、Amazon CloudWatch 日志和 AWS KMS 资源的名称。 AWS Glue Data Catalog它类似于名`AwsGlueDataBrewFullAccessPolicy`为 AWS的托管策略。

下表描述了此策略授予的权限。

操作	资源	描述
`"databrew:*"`	`"*"`	授予运行所有 DataBrew API 操作的权限。

`"glue:GetDatabases"`
`"glue:GetPartitions"`
`"glue:GetTable"`
`"glue:GetTables"`
`"glue:GetDataCatalogEncryptionSettings"`
	`"*"`	允许列出 AWS Glue 数据库和表。

`"dataexchange:ListDataSets"`
`"dataexchange:ListDataSetRevisions"`
`"dataexchange:ListRevisionAssets"`
	`"*"`	允许在 AWS 数据集中列出 Data Exchange 资源。

操作	资源	描述
"dataexchange:CreateJob" "dataexchange:StartJob" "dataexchange:GetJob"		
"kms:DescribeKey" "kms:ListKeys" "kms:ListAliases"	"*"	允许列出用于加密任务输出的 AWS KMS 密钥。
"kms:GenerateDataKey"	"arn:aws:kms:::key/key_ids"	允许对任务输出进行加密。
"s3:ListAllMyBuckets" "s3:GetBucketCORS" "s3:GetBucketLocation" "s3:GetEncryptionConfiguration"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许为项目、数据集和任务列示 Amazon S3 存储桶。允许将输出文件发送到 S3。
"sts:GetCallerIdentity"	"*"	获取有关当前呼叫者的信息。
"cloudtrail:LookupEvents",	"*"	允许列出数据集 AWS CloudTrail 的事件（数据沿袭）。
"iam:ListRoles" "iam:GetRole"	"*"	允许列出用于项目和任务的 IAM 角色。

AwsGlueDataBrewDataResourcePolicy

该AwsGlueDataBrewDataResourcePolicy策略授予连接数据和进行配置所需的权限 DataBrew。

下表描述了此策略授予的权限。

操作	资源	描述
"s3:GetObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许您预览文件。
"s3:PutObject" "s3:PutBucketCORS"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许将输出文件发送到 S3。
"s3:DeleteObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许删除由创建的对象 DataBrew。
"s3:ListBucket"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许通过项目、数据集和任务列示 Amazon S3 存储桶。
"kms:Decrypt"	"arn:aws:kms:::key/key_ids"	允许对加密的数据集进行解密。
"kms:GenerateDataKey"	"arn:aws:kms:::key/key_ids"	允许对任务输出进行加密。
"ec2:DescribeVpcEndpoints" "ec2:DescribeRouteTables" "ec2:DeleteNetworkInterface"	"*"	允许在运行任务和项目时设置 Amazon EC2 网络项，如 Virtual Private Cloud（VPC）。

操作	资源	描述
"ec2:DescribeNetworkInterfaces" "ec2:DescribeSecurityGroups" "ec2:DescribeSubnets" "ec2:DescribeVpcAttribute" "ec2:CreateNetworkInterface"		
"ec2:DeleteNetworkInterface"	"*"	允许删除 VPC 中的网络接口。
"ec2:CreateTags" "ec2:DeleteTags"	"arn:aws:ec2:::network-interface/*", "arn:aws:ec2:::security-group/*"	允许创建和删除标签。 如果您使用启用了 VPC AWS Glue 的数据目录，则需要这些权限。 DataBrew 将数据传递 AWS Glue 给以运行您的作业和项目。这些权限允许标记为开发终端节点创建的 Amazon EC2 资源。 AWS Glue 标记 Amazon EC2 网络接口、安全组和带有的实例aws-glue-service-resource 。

操作	资源	描述
"logs:CreateLogGroup" "logs:CreateLogStream" "logs:PutLogEvents"	"arn:aws:logs:::log-group:/aws-glue-databrew/*"	允许将日志写入 Amazon CloudWatch Logs DataBrew 将日志写入名称以开头的日志组aws-glue-databrew 。
"lakeformation:GetDataAccess"	"*"	允许访问 AWS Lake Formation，前提"Glue":"GetTable" 是也允许访问 使用 Lake Formation 需要在 Lake Formation 控制台中进行进一步配置。

使用 Amazon S3 对象的 IAM 策略 DataBrew

该AwsGlueDataBrewSpecificS3BucketPolicy策略授予代表非管理员用户访问 S3 所需的权限。

按如下方式自定义策略：

1. 替换策略中的 Amazon S3 路径，使其指向您要使用的路径。在示例文本中，*BUCKET-NAME-1/SPECIFIC-OBJECT-NAME*表示特定的对象或文件。 *BUCKET-NAME-2/*表示路径名以开头的所有对象 (*) BUCKET-NAME-2/。更新它们以命名您正在使用的存储桶。
2. （可选）在 Amazon S3 路径中使用通配符进一步限制权限。有关更多信息，请参阅 IAM 用户指南中的 IAM policy 元素：变量和标签。

   作为执行此操作的一部分，您可以限制操作s3:PutObject和的权限s3:PutBucketCORS。只有创建 DataBrew 项目的用户才需要这些操作，因为这些用户需要能够将输出文件发送到 S3。

要了解更多信息并查看您可以向 Amazon S3 的 IAM 策略添加内容的一些示例，请参阅 Amazon S 3 开发人员指南中的存储桶策略示例。

下表描述了此策略授予的权限。

操作	资源	描述
"s3:GetObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许您预览文件。
"s3:PutObject" "s3:PutBucketCORS"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许将输出文件发送到 S3。
"s3:DeleteObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许删除对象。

为 DataBrew（控制台）定义 AwsGlueDataBrewSpecific S3 BucketPolicy IAM 策略

1. 下载 AwsGlueDataBrewSpecificS3BucketPolicyIAM 策略的 JSON。
2. 登录到， AWS Management Console 然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
3. 在导航窗格中，选择策略。
4. 对于每个策略，选择创建策略。
5. 在 “创建策略” 屏幕上，导航到 JSON 选项卡。
6. 在编辑器中将策略 JSON 语句粘贴到示例语句上。
7. 确认该政策是根据您的账户、安全要求和所需 AWS 资源定制的。如果需要进行更改，可以在编辑器中进行更改，可以在编辑器中进行更改。
8. 选择查看策略。

为 DataBrew (AWS CLI) 定义 AwsGlueDataBrewSpecific S3 BucketPolicy IAM 策略

1. 下载适用的 JSON AwsGlueDataBrewSpecificS3BucketPolicy。

2. 如前面的步骤的第一步所述，自定义策略。
3. 运行以下命令创建策略。

```
aws iam create-policy --policy-name AwsGlueDataBrewSpecificS3BucketPolicy --policy-
document file://iam-policy-AwsGlueDataBrewSpecificS3BucketPolicy.json
```

使用加密的 IAM 政策 DataBrew

该AwsGlueDataBrewS3EncryptedPolicy策略授予代表非管理员用户访问用 AWS Key Management Service (AWS KMS) 加密的 S3 对象所需的权限。

按如下方式自定义策略：

1. 替换策略中的 Amazon S3 路径，使其指向您要使用的路径。在示例文本中，*BUCKET-NAME-1/SPECIFIC-OBJECT-NAME*表示特定的对象或文件。 *BUCKET-NAME-2/*表示路径名以开头的所有对象 (*) BUCKET-NAME-2/。更新它们以命名您正在使用的存储桶。
2. （可选）在 Amazon S3 路径中使用通配符进一步限制权限。有关更多信息，请参阅 IAM policy 元素：变量和标签。

   作为执行此操作的一部分，您可以限制操作s3:PutObject和的权限s3:PutBucketCORS。只有创建 DataBrew 项目的用户才需要这些操作，因为这些用户需要能够将输出文件发送到 S3。

   要了解更多信息并查看您可以向 Amazon S3 的 IAM 策略添加内容的一些示例，请参阅存储桶策略示例。
3. 在ToUseKms文件中找到以下资源 ARN。

```
  "arn:aws:kms:AWS-REGION-NAME:AWS-ACCOUNT-ID-WITHOUT-DASHES:key/KEY-IDS",
"arn:aws:kms:AWS-REGION-NAME:AWS-ACCOUNT-ID-WITHOUT-DASHES:key/KEY-IDS"
```

4. 将示例 AWS 账户更改为您的 AWS 账号（不含连字符）。
5. 更改示例列表，改为列出您要使用的 IAM 角色。我们建议您将 IAM 策略的范围限定为尽可能小的权限集。但是，您可以允许您的用户访问所有 IAM 角色，例如，如果您使用的是包含示例数据的个人学习账户。要允许列表访问所有 IAM 角色，请将示例列表更改为一个条目："arn:aws:iam::*111122223333*:role/*"。

下表描述了此策略授予的权限。

操作	资源	描述
"s3:GetObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许您预览文件。
"s3:ListBucket"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许通过项目、数据集和任务列示 Amazon S3 存储桶。
"s3:PutObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许将输出文件发送到 S3。
"s3:DeleteObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	允许删除由创建的对象 DataBrew。
"kms:Decrypt"	"arn:aws:kms:::key/key_ids"	允许对加密的数据集进行解密。
"kms:GenerateDataKey*"	"arn:aws:kms:::key/key_ids"	允许对任务输出进行加密。

为 DataBrew（控制台）定义 AwsGlueDataBrew S3 EncryptedPolicy IAM 策略

1. 下载 AwsGlueDataBrewS3EncryptedPolicyIAM 策略的 JSON。
2. 登录到， AWS Management Console 然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
3. 在导航窗格中，选择策略。
4. 对于每个策略，选择创建策略。
5. 在“创建策略”屏幕上，导航到 JSON 选项卡。

6. 在编辑器中将策略 JSON 语句粘贴到示例语句上。
7. 确认该政策是根据您的账户、安全要求和所需 AWS 资源定制的。如果需要进行更改，可以在编辑器中进行更改，可以在编辑器中进行更改。
8. 选择查看策略。

为 DataBrew (AWS CLI) 定义 AwsGlueDataBrew S3 EncryptedPolicy IAM 策略

1. 下载适用的 JSON AwsGlueDataBrewS3EncryptedPolicy。
2. 如前面的步骤的第一步所述，自定义策略。
3. 运行以下命令创建策略。

```
aws iam create-policy --policy-name AwsGlueDataBrewS3EncryptedPolicy --policy-
document file://iam-policy-AwsGlueDataBrewS3EncryptedPolicy.json
```

添加具有 DataBrew 权限的用户或群组

您可以为角色分配策略，为用户和组分配角色以管理权限。有关更多信息，请参阅 IAM 用户指南中的 IAM 身份（用户、群组和角色）。

在开始之前，您需要至少有一个用户向其分配权限。

使用以下步骤为需要在 DataBrew 控制台中工作或在 CLI 中运行 DataBrew 命令的用户设置 DataBrew 权限。

要设置 DataBrew 权限

1. 创建访问密钥供您的用户使用 f AWS CLI o DataBrew r 和其他开发工具。
2. 启用AWS Management Console 访问权限以允许用户使用 AWS 控制台。
3. 为 DataBrew 用户或群组创建角色。
4. 选择您正在使用的策略。请执行以下操作之一：
   - 如果您已创建AwsGlueDataBrewCustomUserPolicy，请从列表中选择。
   - 要使用 AWS 托管的策略，请AwsGlueDataBrewFullAccessPolicy从列表中选择。
5. 将该策略分配给该角色。
6. 为角色设置信任关系，以便用户或群组可以担任相关角色。

- 如果您不使用群组，请信任具有该角色的用户。
- 如果您使用的是组，请信任该组具有该角色并将该用户添加到该组中。

添加具有数据资源权限的 IAM 角色

您可以使用 IAM 角色来管理一起分配的策略。IAM 角色可以由扮演特定角色的人使用，例如 DataBrew 用户或 DataBrew 其自己。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

使用以下过程创建 DataBrew 项目访问数据所需的 IAM 角色。

将所需的 IAM 策略附加到新的 IAM 角色 DataBrew

1. 在导航窗格中，选择 角色 和 创建角色。
2. 在 “选择可信实体类型” 中，选择标有卡片的AWS 服务。
3. DataBrew从列表中选择，然后选择 “下一步：权限”。
4. **AwsGlueDataBrewDataResourcePolicy**在搜索框中输入（您在之前的步骤中创建的 IAM 策略）。选择策略并选择下一步：标签。
5. 选择 下一步: 审核。
6. 对于角色名称，输入 **AwsGlueDataBrewDataAccessRole**，然后选择创建角色。

设置 AWS IAM Identity Center （IAM 身份中心）

使用 AWS IAM Identity Center (IAM Identity Center)，您的用户可以使用简单的 URL 登录 AWS Management Console ，无需登录，也无需 AWS 账户。 DataBrew

设置 IAM 身份中心

1. 打开AWS Organizations 控制台，如果您还没有组织，请创建一个组织。默认情况下，该组织的所有功能均处于启用状态。

   有关更多信息，请参阅AWS IAM Identity Center 先决条件和创建和管理组织。
2. 打开 AWS IAM Identity Center 控制台
3. 选择您的身份来源。

默认情况下，您可以获得 IAM Identity Center 存储，以便快速轻松地管理用户。或者，您可以改为连接外部身份提供商，或者将 AWS Managed Microsoft AD 目录与本地 Active Directory 连接起来。在本指南中，我们使用默认的 IAM 身份中心存储。

有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的选择您的身份来源。

4. 创建 DataBrew 访问权限集：

   a. 在 IAM 身份中心导航窗格中，选择AWS 账户，然后选择权限集。

   b. 在 “创建权限集” 页面上，选择 “创建自定义权限集”。

   c. 对于 “中继状态”，输入`https://console.aws.amazon.com/databrew/home?region=us-east-1#landing`。

      输入此字段后，您的用户就可以直接转到 DataBrew。

   d. 选择附加 AWS 托管策略 DataBrew，搜索并选择AwsGlueDataBrewFullAccessPolicy。选择此选项可为您的用户提供他们所需的所有权限 DataBrew。您可以在中找到更多详细信息为控制台用户添加 IAM 策略。

   e. （可选）选择 “创建自定义权限策略”，然后为您的用户自定义权限。

5. 在 IAM 身份中心导航窗格中，选择群组，然后选择创建群组。输入群组名称并选择创建。

6. 将用户添加到 IAM 身份中心存储：

   a. 在 IAM 身份中心导航窗格中，选择用户。

   b. 在添加用户屏幕上，输入所需信息，然后选择向用户发送包含密码设置说明的电子邮件。用户应该会收到一封关于后续设置步骤的电子邮件。

   c. 选择 “下一步：群组”，选择所需的群组，然后选择 “添加用户”。

      用户应收到一封邀请他们使用 SSO 的电子邮件。在这封电子邮件中，他们需要选择接受邀请并设置密码。他们还可以在电子邮件中找到门户 URL。他们可以使用此 URL 进行访问 DataBrew。

7. 为每个用户分配一个帐户：

   a. 打开 I AM 身份中心控制台，然后在导航窗格中选择AWS 账户。

   b. 选择AWS 组织并选择一个 AWS 帐户。

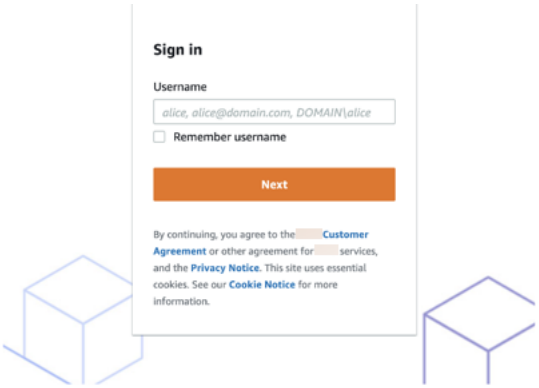
   c. 在分配用户屏幕上，选择群组选项卡，然后选择所需的群组。

   d. 选择 Next: Permissions sets (下一步：权限集)。

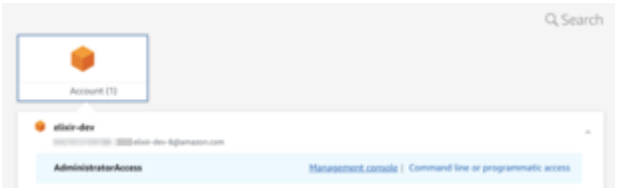
   e. 选择的权限集 DataBrew，然后选择 “完成”。

启用 IAM 身份中心的用户的登录步骤

1. AWS 使用支持 IAM 身份中心的账户登录。



2. 点击AWS 账户身份



3. 单击 “管理控制台”，一键重定向到控制台。 DataBrew

在中 DataBrew 用作扩展 JupyterLab

如果您更喜欢在 Jupyter Notebook 环境中准备数据，则可以使用中的所有功能。 AWS Glue DataBrew JupyterLab

JupyterLab 是 Jupyter Notebook 的基于 Web 的交互式开发环境。在本地 JupyterLab网页中，您可以为终端、SQL 会话、Python 等添加部分。安装 AWS Glue DataBrew 扩展程序后，您可以为 DataBrew 控制台添加一个部分。它可以直接从 JupyterLab 环境中运行任何现有的笔记本或其他扩展程序。

主题

- 先决条件
- 配置 JupyterLab 为使用该扩展
- 为启用 DataBrew 扩展 JupyterLab

先决条件

开始之前，请设置以下项目：

- 一个 AWS 账户 — 如果你还没有，请从这里开始设置一个新 AWS 账户。
- 有权访问所需权限的 AWS Identity and Access Management (IAM) 用户 DataBrew — 有关更多信息，请参阅添加具有 DataBrew 权限的用户或群组。
- 在 DataBrew 操作中使用的 IAM 角色 — 您可以使用默认角色（如果`AwsGlueDataBrewDataAccessRole`已配置）。要设置其他 IAM 角色，请参阅添加具有数据资源权限的 IAM 角色。
- JupyterLab 安装（版本 2.2.6 或更高版本）— 有关更多信息，请参阅JupyterLab文档中的以下主题：
  - JupyterLab 先决条件
  - JupyterLab 安装 — 我们建议使用`pip install jupyterlab`。
- Node.js 安装（版本 12.0 或更高版本）。
- AWS Command Line Interface (AWS CLI) 安装-有关更多信息，请参阅设置 AWS CLI。
- AWS Jupyter 代理安装 (`pip install aws-jupyter-proxy`) — 此扩展与 AWS 服务端点一起使用，可安全地传递您的 AWS 凭据。有关更多信息，请参aws-jupyter-proxy阅 GitHub。

要验证是否已安装必备组件，可以在命令行运行类似于以下内容的测试，如以下示例所示。

```
echo "
AWS CLI:"
which aws
aws --version
aws configure list
aws sts get-caller-identity

echo "
Python (current environment):"
which python
python --version

echo "
Node.JS:"
which node
node --version
```

```
echo "
Jupyter:"
where jupyter
jupyter --version
jupyter serverextension list
pip3 freeze | grep jupyter
```

输出应与以下内容类似。目录因操作系统和配置而异。

```
AWS CLI:
/usr/local/bin/aws
aws-cli/2.1.2 Python/3.7.4 Darwin/19.6.0 exe/x86_64
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************VXW4 shared-credentials-file
secret_key     ****************MRJN shared-credentials-file
    region                us-east-1      config-file    ~/.aws/config
{
    "UserId": "",
    "Account": "111122223333",
    "Arn": "arn:aws:iam::111122223333:user/user2"
}

Python (current environment):
/usr/local/opt/python /libexec/bin/python
Python 3.8.5

Node.JS:
/usr/local/bin/node
v15.0.1

Jupyter:
/usr/local/bin/jupyter
jupyter core     : 4.6.3
jupyter-notebook : 6.0.3
qtconsole        : 4.7.5
ipython          : 7.16.1
ipykernel        : 5.3.2
jupyter client   : 6.1.6
jupyter lab      : 2.2.9
nbconvert        : 5.6.1
ipywidgets       : 7.5.1
```

```
nbformat         : 5.0.7
traitlets        : 4.3.3

config dir: /usr/local/etc/jupyter
    aws_jupyter_proxy  enabled
    - Validating...
      aws_jupyter_proxy  OK
    jupyterlab  enabled
    - Validating...
      jupyterlab 2.2.9 OK

aws-jupyter-proxy==0.1.0
jupyter-client==6.1.7
jupyter-core==4.7.0
jupyterlab==2.2.9
jupyterlab-pygments==0.1.2
jupyterlab-server==1.2.0
```

配置 JupyterLab 为使用该扩展

安装之后 JupyterLab，您需要配置它才能保护数据访问并启用服务器扩展。

配置密码和加密

1. 设置密码以保护您计划在扩展程序中添加的数据。Jupyter 提供了一个密码实用工具。运行以下命令，在命令提示符处输入您的首选密码。

   ```
   jupyter notebook password
   ```

   输出如下所示。

   ```
   Enter password:
   Verify password:
   [NotebookPasswordApp] Wrote hashed password to /home/ubuntu/.jupyter/
   jupyter_notebook_config.json
   ```

2. 在 Jupyter 服务器上启用加密。如果您在本地计算机上安装 Jupyter，但没有人可以通过网络访问它，则可以跳过此步骤。

   要使用传输层安全性协议 (TLS) 设置加密，请创建针对您的环境自定义的证书。有关更多信息，请参阅 Jupyter 文档中的 “保护服务器” 中的 “使用 L et's Enc rypt”。

3. 要重新启动 JupyterLab，请在命令提示符中运行以下命令。

```
jupyter lab
```

有关更多信息，请参阅 JupyterLab 文档 JupyterLab中的 “开始”。

4. 运行 JupyterLab 时，您可以通过类似于以下网址访问它：http://localhost:8888/lab。如果您设置了加密，请使用`https`而不是`http`。如果您自定义了端口，请替换您的端口号8888。

使用以下步骤启用第三方扩展。

要在中启用第三方扩展 JupyterLab

1. 在 JupyterLab 网页上，选择左侧菜单中的扩展管理器图标。
2. 阅读有关运行第三方扩展程序风险的警告。只安装你信任的开发者提供的扩展。
3. 要在中启用第三方扩展 JupyterLab，请选择 “启用”。
4. 按照提示进行重建和重新加载 JupyterLab。

为启用 DataBrew 扩展 JupyterLab

在启用了扩展程序的情况下安全安装之后，请安装 DataBrew 扩展程序，这样您就可以在笔记本电脑 DataBrew 上运行。 JupyterLab

安装 DataBrew （控制台）的扩展

1. 要重新启动 JupyterLab，请在命令提示符中运行以下命令。

```
jupyter lab
```

2. 在 JupyterLab 网页上，选择左侧菜单中的扩展管理器图标。
3. 在左上角的 “搜索**`brew`**” 中输入 “” 来搜索分 DataBrew 机。
4. 在列表中找到 aws_glue_databrew_jupyter，但不要点击它。如果您单击突出显示的扩展程序名称，则会打开一个新的浏览器窗口，并打开 aws_g lue_databrew_j upyter 页面。 GitHub
5. 要安装 DataBrew 扩展，请选择下列选项之一：

   - 在命令行处，运行`jupyter labextension install aws_glue_databrew_jupyter`。
   - 选择扩展卡底部的灰色字母的 “aws_gl ue_databrew_j upyter” 下方的安装。

DataBrew 扩展与 1.2 和 2.x JupyterLab 版本兼容。

6. 要验证它是否已安装，请运行`jupyter labextension list`。输出应与以下内容类似。

```
JupyterLab v2.2.9
Known labextensions:
   app dir: /usr/local/share/jupyter/lab  # varies by OS
        aws_glue_databrew_jupyter v1.0.1  enabled  OK
```

7. 使用以下 JupyterLab 方法之一进行重建：

   - 在命令提示符下，运行`jupyter lab build`。
   - 在网页中，选择左上角的 “重建”。

8. 生成完成之后，请执行下列操作之一：

   - 在命令提示符下，运行`jupyter lab`。
   - 在网页中，选择 “构建完成” 消息中的 “重新加载”。

9. 在 JupyterLab 网页中，通过在左侧菜单中选择扩展管理器的图标将其关闭。

   要打开扩展程序，请 AWS Glue DataBrew从 “启动器” 选项卡的 “其他” 部分选择 “启动”。该扩展程序使用您当前的访问密钥和 AWS 区域设置 AWS CLI 配置。

完成设置后，您可以使用该AWS Glue DataBrew选项卡 DataBrew 从内部进行交互 JupyterLab。

入门 AWS Glue DataBrew

您可以使用以下教程来指导您创建第一个 DataBrew 项目。您可以加载示例数据集，在该数据集上运行转换，构建用于捕获这些转换的配方，然后运行任务将转换后的数据写入 Amazon S3。

主题

先决条件

在继续操作之前，请按照中的适用说明进行操作设置 AWS Glue DataBrew。然后继续步骤 1：创建项目。

步骤 1：创建项目

在此步骤中，您将使用 DataBrew 控制台快速开始示例项目。

创建项目

1. 登录 AWS Management Console 并打开 DataBrew 控制台，网址为 https://console.aws.amazon.com/databrew/。
2. 确保在控制台的右上角选择了您 AWS 所在的地区。 DataBrew有关支持的 AWS 区域列表 DataBrew，请参阅中的DataBrew 终端节点和配额AWS 一般参考。
3. 在导航窗格上，选择 “项目”，然后选择 “创建项目”。
4. 在项目详细信息窗格上，执行以下操作：
   - 在 “项目名称” 中，输入`chess-project`。

- 对于附加食谱，创建一个新配方。提供了食谱的建议名称（chess-project-recipe）。

5. 在选择数据集窗格上，选择示例文件。
6. 在示例文件窗格上，选择著名的国际象棋游戏动作。该数据集包含超过 20,000 场国际象棋游戏的详细信息。

   对于数据集名称，提供了数据集的建议名称 (chess-games)。
7. 在 “访问权限” 窗格上，选择AwsGlueDataBrewDataAccessRole。这是一个服务相关角色，允许代表您 DataBrew 访问 Amazon S3 存储桶。
8. 选择 “创建项目”，然后等到项目准备 DataBrew 完毕。窗口看上去类似下面这样。

   您看到的数据代表数据chess-games集中的样本。默认情况下，样本包含数据集中的前 500 行。您以后可以更改此项目设置。

   工具栏提供对数百种数据转换的访问权限，您可以将其应用于数据。

   DataBrew 控制台右侧的配方窗格会跟踪你到目前为止应用的变换。

步骤 2：汇总数据

在此步骤中，您将构建一个 DataBrew 配方，即一组可以应用于此数据集和其他类似数据集的转换。配方完成后，您可以将其发布以供使用。

在国际象棋游戏中，可以根据玩家与其他玩家的表现来对他们进行评级。（有关更多信息，请参阅https://en.wikipedia.org/wiki/Chess_rating_system）。在本教程中，你只关注两个玩家都是 A 级（即他们的评分均为 1800 或更高）的游戏。

总结数据

1. 在变换工具栏上，选择筛选、按条件、大于或等于。
2. 按如下方式设置这些选项：

   - 来源专栏-white_rating
   - 筛选条件-大于或等于 1800

   要查看变换的工作原理，请选择 “预览更改”。然后选择 Apply（应用）。
3. 重复上一个步骤，但这次将 “来源” 列设置为black_rating。应用更改后，样本数据仅包含双方玩家（黑白）均为 A 级或以上级别的游戏。

4. 汇总数据以确定双方赢了多少场比赛。为此，请在变换工具栏上选择 “群组”。
5. 对于群组属性，请执行以下操作：

   a. 在第一行中，选择 “winner列名”。将 “聚合” 设置为 “分组依据”。

   b. 在第二行中victory_status，选择列名。将 “聚合” 设置为 “分组依据”。

   c. 选择 “添加另一列”。

   d. 在第三行中，选择 “winner列名”。将聚合设置为计数。

   e. 对于 “组类型”，选择 “分组为新表”。预览窗格会显示结果的样子。

   f. 选择完成。
6. 选择 “发布” 以保存您所做的工作，位于食谱窗格的右侧。
7. 在版本描述中，输入我的食谱的第一个版本。然后选择 “发布”。

步骤 3：添加更多转换

在此步骤中，您将向食谱添加更多转换并发布其另一个版本。为了完善我们的示例，我们使用的信息是，并非所有国际象棋游戏都会产生明确的赢家；有些游戏是平局的。

添加更多食谱转换并重新发布

1. 从变换工具栏中，选择 “过滤”、“按条件”、“不是”，以移除平局玩过的游戏。
2. 按如下方式设置这些选项：

   - 来源专栏-victory_status
   - 筛选条件-不是 draw

   要将此变换添加到您的食谱中，请选择 “应用”。
3. 更改数据，victory_status使其更有意义。为此，请从转换工具栏中选择 “清理”、“替换”、“替换值” 或 “模式”。
4. 按如下方式设置这些选项：

   - 来源专栏-victory_status
   - 指定要替换的值-值或模式
   - 要替换的值-mate
   - 用@@ 值替换-checkmate

要将此变换添加到您的食谱中，请选择 “应用”。

5. 重复上一个步骤，但更改resign为other player resigned。
6. 重复上一个步骤，但更改outoftime为time ran out。
7. 选择 “发布” 以保存您所做的工作，位于食谱窗格的右侧。

步骤 4：审核 DataBrew 资源

现在，您已经使用了一个示例项目，请查看到目前为止创建的 DataBrew 资源。

查看您的 DataBrew 资源

1. 在导航窗格上，选择数据集。

   创建示例项目时，为您 DataBrew 创建了一个数据集 (chess-games)。源数据文件存储在亚马逊 S3 中，采用微软 Excel 格式 (chess-games.xlsx)。该文件包含来自 20,000 多场国际象棋游戏的元数据。chess-games数据集提供了读取该文件中数据 DataBrew 所需的信息。

2. 在导航窗格上，选择项目。

   您应该看到在前面的步骤中使用的项目（chess-project）。在这种情况下，每个项目都需要一个数据集chess-games。每个项目还需要一个配方，这样你就可以随心所欲地添加数据转换步骤。创建此示例项目时，为您 DataBrew 创建了一个新的（空）配方，并将其附加到项目中。

3. 在导航窗格上，选择 “食谱”，然后在 “食谱名称” 列中选择chess-project-recipe。这向你展示了为你的项目 DataBrew 创建的配方，以及你通过向项目添加转换步骤来完善的配方。
4. 在左边，查看已发布的食谱版本。选择其中一个即可查看其食谱步骤选项卡，其中显示了该版本的食谱详细信息和步骤。
5. 查看 “数据世系” 选项卡，该选项卡显示了数据的来源和使用方式。要了解更多详细信息，请选择图表中的任意图标。

步骤 5：创建数据配置文件

在处理项目时， DataBrew 会显示统计信息，例如样本中的行数和每列中唯一值的分布。这些统计数据以及更多统计数据代表了样本的概况。

要请求数据配置文件，请创建并运行分析作业。

分析数据集

1. 在导航窗格上，选择作业。
2. 在 “分析作业” 选项卡上，选择 “创建作业”。
3. 在 Job 名称中输入chess-data-profile。
4. 对于 Job 类型，选择 “创建个人资料作业”。
5. 在 Job 输入窗格上，执行以下操作：
   - 对于 “运行于”，选择 “数据集”。
   - 选择选择数据集以查看可用数据集列表，然后选择chess-games。
6. 在 Job 输出设置窗格上，执行以下操作：
   - 对于文件类型，选择 JSON（JavaScript 对象表示法）。
   - 选择 S3 位置以查看可用 Amazon S3 存储桶的列表，然后选择要使用的存储桶。然后选择浏览。在文件夹列表中，选择databrew-output，然后选择选择。
7. 在 “访问权限” 窗格上，选择AwsGlueDataBrewDataAccessRole。这是一个服务关联角色，可代表您 DataBrew 访问 Amazon S3 存储桶。
8. 选择创建并运行作业。 DataBrew 使用您的设置创建作业，然后运行该作业。
9. 在 Job 运行历史记录窗格上，等待任务状态从变Running为Succeeded。
10. 要查看个人资料，请选择查看个人资料：

    将显示 “数据集” 窗口。花点时间浏览以下选项卡：
    - 数据集预览
    - 个人资料概述
    - 列统计数据
    - 数据沿袭

步骤 6：转换数据集

到目前为止，您仅在数据集的样本上测试了您的食谱。现在是时候通过创建 DataBrew 配方作业来转换整个数据集了。

任务运行时， DataBrew 将您的配方应用于数据集中的所有数据，并将转换后的数据写入 Amazon S3 存储桶。转换后的数据与原始数据集是分开的。 DataBrew 不会更改源数据。

在继续操作之前，请确保您的账户中存在可写入的 Amazon S3 存储桶。在该存储桶中，创建一个文件夹来捕获作业输出 DataBrew。要执行这些步骤，请使用以下过程。

创建 S3 存储桶和文件夹来捕获作业输出

1. 登录到， AWS Management Console 然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/databrew/。

   如果您已有 Amazon S3 存储桶可用，并且您具有写入权限，请跳过下一个步骤。
2. 如果您没有 Amazon S3 存储桶，请选择创建存储桶。在存储桶名称中，输入新存储桶的唯一名称。选择创建存储桶。
3. 从存储桶列表中，选择要使用的存储桶。
4. 请选择 Create folder（创建文件夹）。
5. 在 “文件夹名称” 中`databrew-output`，输入并选择 “创建文件夹”。

在创建用于包含任务的 S3 存储桶和文件夹后，请使用以下过程运行您的作业。

创建和运行配方作业

1. 在导航窗格上，选择作业。
2. 在 “配方作业” 选项卡上，选择 “创建作业”。
3. 在 Job 名称中输入`chess-winner-summary`。
4. 对于 Job 类型，选择创建配方作业。
5. 在 Job 输入窗格上，执行以下操作：
   - 对于 “运行于”，选择 “数据集”。
   - 选择选择数据集以查看可用数据集列表，然后选择`chess-games`。
   - 选择 “选择食谱” 以查看可用食谱列表，然后选择`chess-project-recipe`。
6. 在 Job 输出设置窗格上，执行以下操作：
   - 文件类型-选择 CSV（逗号分隔值）。
   - S3 位置-选择此字段可查看可用 Amazon S3 存储桶的列表，然后选择要使用的存储桶。然后选择浏览。在文件夹列表中，选择`databrew-output`，然后选择选择。

7. 在 “访问权限” 窗格上，选择AwsGlueDataBrewDataAccessRole。此服务相关角色可代表您DataBrew访问 Amazon S3 存储桶。
8. 选择创建并运行作业。 DataBrew 使用您的设置创建作业，然后运行该作业。
9. 在 Job 运行历史记录窗格上，等待任务状态从变Running为Succeeded。
10. 选择 “输出” 以访问 Amazon S3 控制台。选择您的 S3 存储桶，然后选择用于访问任务输出的databrew-output文件夹。
11. （可选）选择 “下载” 以下载文件并查看其内容。

第 7 步：（可选）清除

演练已完成。您可以继续使用自己创建的 DataBrew 和 Amazon S3 资源，也可以将其删除。

清理资源

1. 通过 https://console.aws.amazon.com/databrew/ 打开 DataBrew 控制台，然后在导航窗格上选择 Pro j ects。
2. 选择您的项目（示例项目）。对于操作，选择删除。
3. 在删除示例项目窗格上，选择删除附加的配方。然后选择删除。您的项目及其配方和任务将被删除。
4. 在导航窗格上，选择数据集。
5. 选择您的数据集 (chess-games)，然后在 “操作” 中选择 “删除”。
6. 打开 Amazon S3 控制台，网址为：https://console.aws.amazon.com/s3/。删除databrew-output文件夹及其内容。

   （可选）如果您确定不再需要 Amazon S3 存储桶，则可以将其删除。

使用连接数据 AWS Glue DataBrew

在中 AWS Glue DataBrew，数据集表示从文件上传或存储在其他位置的数据。例如，数据可以存储在 Amazon S3、支持的 JDBC 数据源或 AWS Glue 数据目录中。如果您没有将文件直接上传到 DataBrew，则数据集还包含有关 DataBrew 如何连接到数据的详细信息。

创建数据集时（例如，`inventory-dataset`），只需输入一次连接详细信息。从那时起， DataBrew 可以为你访问基础数据。通过这种方法，您可以创建项目并对数据进行转换，而不必担心连接细节或文件格式。

主题

数据源支持的文件类型

以下文件要求适用于存储在 Amazon S3 中的文件以及您从本地驱动器上传的文件。 DataBrew 支持以下文件格式：逗号分隔值 (CSV)、微软 Excel、JSON、ORC 和 Parquet。如果文件属于支持的类型之一，则可以使用带有非标准扩展名或没有扩展名的文件。

DataBrew 如果无法推断出文件类型，请确保自己选择正确的文件类型（CSV、Excel、JSON、ORC 或 Parquet）。支持压缩的 CSV、JSON、ORC 和 Parquet 文件，但是 CSV 和 JSON 文件必须包含压缩编解码器作为文件扩展名。如果要导入文件夹，则该文件夹中的所有文件类型必须相同。

下表显示了文件格式和支持的压缩算法。

> **Note**
>
> CSV、Excel 和 JSON 文件必须使用 Unicode 编码 (UTF-8)。

格式	文件扩展名（可选）	压缩文件的扩展名（必填）

以逗号分隔的值	`.csv`	`.gz`
		`.snappy`
		`.lz4`
		`.bz2`
		`.deflate`

微软 Excel 工作簿	`.xlsx`	不支持压缩

JSON（JSON 文档和 JSON 行）	`.json, .jsonl`	`.gz`
		`.snappy`
		`.lz4`
		`.bz2`
		`.deflate`

Apache ORC	`.orc`	`.zlib` `.snappy`

Apache Parquet	`.parquet`	`.gz`
		`.snappy`
		`.lz4`

支持的数据源和输出连接

您可以连接到以下数据源以执行 DataBrew 配方作业。这些数据源包括任何不是您要直接上传到的文件的数据源 DataBrew。您正在使用的数据源可能称为数据库、数据仓库或其他东西。我们将所有数据提供者称为数据源或连接。

您可以使用以下任何一种作为数据源来创建数据集。

您还可以使用 Amazon S3 或 Amazon RDS 支持的 JDBC 数据库来输出 DataBrew 配方任务。 AWS Glue Data Catalog AWS Data Exchange 不支持 Amazon AppFlow 和 Amazon 存储 DataBrew 食谱任务输出的数据存储。

- Amazon S3

  您可以使用 S3 来存储和保护任意数量的数据。要创建数据集，请指定 DataBrew 可以访问数据文件的 S3 URL，例如：`s3://your-bucket-name/inventory-data.csv`

  DataBrew 还可以读取 S3 文件夹中的所有文件，这意味着您可以创建跨多个文件的数据集。为此，请使用以下形式指定 S3 URL：`s3://your-bucket-name/your-folder-name/`。

  DataBrew 仅支持以下 Amazon S3 存储类别：标准、低冗余、标准 IA 和 S3 单区-IA。 DataBrew 忽略具有其他存储类别的文件。 DataBrew 还会忽略空文件（包含 0 字节的文件）。有关 Amazon S3 存储类的更多信息，请参阅 Amazon S3 控制台用户指南中的使用 Amazon S3 存储类别。
- AWS Glue Data Catalog

  您可以使用数据目录来定义对存储在 AWS 云中的数据的引用。使用数据目录，您可以建立与以下服务中各个表的连接：
  - 数据目录 Amazon S3
  - 数据目录 Amazon Redshift
  - 数据目录 Amazon RDS
  - AWS Glue

  DataBrew 还可以读取 Amazon S3 文件夹中的所有文件，这意味着您可以创建跨多个文件的数据集。为此，请按以下格式指定 Amazon S3 网址：`s3://your-bucket-name/your-folder-name/`

  要与一起使用 DataBrew，中定义的 Amazon S3 表必须添加一个名为 a 的表属性`classification`，该属性将数据的格式标识为`csv`、`json`或`parquet`，以及 a `typeOfData` `s file`。 AWS Glue Data Catalog如果在创建表时未添加表属性，则可以使用 AWS Glue 控制台添加该属性。

  DataBrew 仅支持 Amazon S3 存储类别 “标准”、“低冗余”、“标准 — IA” 和 “S3 单区 — IA”。 DataBrew 忽略具有其他存储类别的文件。 DataBrew 还会忽略空文件（包含 0 字节的文件）。有关 Amazon S3 存储类的更多信息，请参阅 Amazon S3 控制台用户指南中的使用 Amazon S3 存储类别。

DataBrew 如果创建了适当的资源策略，也可以从其他账户访问 AWS Glue Data Catalog S3 表。您可以在 AWS Glue 控制台的数据目录下方的设置选项卡上创建策略。以下是专门针对单个的策略示例 AWS 区域。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect" : "Allow",
            "Principal" : {
                "AWS" : "*$ACCOUNT_TO*"
            },
            "Action" : "glue:*",
            "Resource" : "arn:aws:glue:*us-east-1*:*$ACCOUNT_FROM*:*"
        }
    ]
}
```

> **Warning**
>
> 这是一项高度宽松的资源策略，可授予对数据目录的*$ACCOUNT_TO*无限制访问权限。*$ACCOUNT_FROM*在大多数情况下，我们建议您将资源策略锁定为特定的目录或表。有关更多信息，请参阅《AWS Glue 开发人员指南》中的访问控制AWS Glue 资源策略。

在某些情况下，您可能需要创建项目或在中运行任务，其中*$ACCOUNT_TO*包含指向也 AWS Glue DataBrew 位于的 S3 位置*$ACCOUNT_FROM*的 AWS Glue Data Catalog S3 表*$ACCOUNT_FROM*。在这种情况下，在中创建项目和任务时使用的 IAM 角色*$ACCOUNT_TO*必须具有列出和获取该 S3 位置中对象的权限*$ACCOUNT_FROM*。有关更多信息，请参阅《AWS Glue 开发者指南》中的授予跨账户访问权限。

- 使用 JDBC 驱动程序连接数据

  您可以通过使用支持的 JDBC 驱动程序连接到数据来创建数据集。有关更多信息，请参阅 将驱动程序与 AWS Glue DataBrew。

  DataBrew 使用 Java 数据库连接 (JDBC) 正式支持以下数据源：

  - Microsoft SQL Server
  - MySQL

- Oracle
- PostgreSQL
- Amazon Redshift
- 适用于 Spark 的雪花连接器

数据源可以位于您可以从中连接到它们的任何地方 DataBrew。此列表仅包括我们测试过并因此可以支持的 JDBC 连接。

适用于 Spark 数据源的 Amazon Redshift 和 Snowflake Connector 可以通过以下任一方式进行连接：

- 带有表名。
- 使用跨越多个表和操作的 SQL 查询。

SQL 查询是在启动项目或作业运行时执行的。

要连接到需要未列出的 JDBC 驱动程序的数据，请确保该驱动程序与 JDK 8 兼容。要使用该驱动程序，请将其存储在 S3 中的存储桶中，您可以在其中使用您的 IAM 角色对其进行访问 DataBrew。然后将您的数据集指向驱动程序文件。有关更多信息，请参阅 将驱动程序与 AWS Glue DataBrew。

基于 SQL 的数据集的查询示例：

```
SELECT
    *
FROM
    public.customer as c
JOIN
    public.customer_address as ca on c.current_address=ca.current_address
WHERE
    ca.address_id>0 AND ca.address_id<10001 ORDER BY ca.address_id
```

自定义 SQL 的局限性

如果您使用 JDBC 连接访问数据 DataBrew 集的数据，请记住以下几点：

- AWS Glue DataBrew 不会验证您在创建数据集时提供的自定义 SQL。SQL 查询将在您启动项目或作业运行时执行。 DataBrew 获取您提供的查询，并使用默认或提供的 JDBC 驱动程序将其传递给数据库引擎。
- 使用无效查询创建的数据集在项目或作业中使用时将失败。在创建数据集之前验证您的查询。

- 验证 SQL 功能仅适用于基于 Amazon Redshift 的数据源。
- 如果要在项目中使用数据集，请将 SQL 查询运行时间限制在三分钟以内，以避免项目加载期间出现超时。在创建项目之前，请检查查询运行时间。

- Amazon AppFlow

  使用亚马逊 AppFlow，你可以将数据从第三方 S oftware-as-a 服务 (SaaS) 应用程序传输到亚马逊 S3，例如 Salesforce、Zendesk、Slack 和。 ServiceNow然后，您可以使用这些数据来创建 DataBrew 数据集。

  在 Amazon 中 AppFlow，您可以创建连接和流程，以便在您的第三方应用程序和目标应用程序之间传输数据。将亚马逊 AppFlow 与配合使用时 DataBrew，请确保亚马逊的 AppFlow 目标应用程序是 Amazon S3。除了 Amazon S3 之外，Amazon AppFlow 目标应用程序不会出现在 DataBrew 控制台中。有关从第三方应用程序传输数据以及创建 Amazon AppFlow 连接和流程的更多信息，请参阅 Amazon AppFlow 文档。

  当您在的 “数据集” 选项卡中选择 “连接新数据集” DataBrew 并单击 Amazon 时 AppFlow，您将看到亚马逊 AppFlow 中所有配置为 Amazon S3 作为目标应用程序的流程。要将流程的数据用于您的数据集，请选择该流程。

  AppFlow 在 DataBrew 控制台中选择 “创建流程”、“管理流程” 和 “查看亚马逊详情” 将打开 Amazon AppFlow 控制台，以便您可以执行这些任务。

  从 Amazon 创建数据集后 AppFlow，您可以在查看数据集详细信息或任务详细信息时运行流程并查看最新的流程运行详情。当您在中运行流程时 DataBrew，数据集将在 S3 中更新并准备好在中使用 DataBrew。

  当您在 DataBrew 控制台中选择 Amazon AppFlow 流程来创建数据集时，可能会出现以下情况：

  - 数据尚未汇总-如果流量触发器为按需运行或按计划运行且具有完整数据传输，请务必在使用流程创建数据 DataBrew 集之前汇总该流程的数据。聚合流程会将流程中的所有记录合并到一个文件中。触发器类型为 “通过增量数据传输按计划运行” 或 “在事件上运行” 的流程不需要聚合。要在 Amazon 中聚合数据 AppFlow，请选择编辑流程配置 > 目标详情 > 其他设置 > 数据传输首选项。
  - 流程尚未运行-如果流程的运行状态为空，则表示以下情况之一：
    - 如果运行流程的触发器是 “按需运行”，则该流程尚未运行。
    - 如果运行流程的触发器是 Run on ev ent，则触发事件尚未发生。
    - 如果运行流程的触发器是 “按计划运行”，则尚未进行计划运行。

    在创建包含流程的数据集之前，请为该流程选择运行流程。

有关更多信息，请参阅《亚马逊 AppFlow 用户指南》中的亚马逊 AppFlow 流程。

- AWS Data Exchange

  您可以从中提供的数百种第三方数据源中进行选择 AWS Data Exchange。通过订阅这些数据源，您可以获得最大 up-to-date 版本的数据。

  要创建数据集，您需要指定您已订阅并有权使用的 AWS Data Exchange 数据产品的名称。

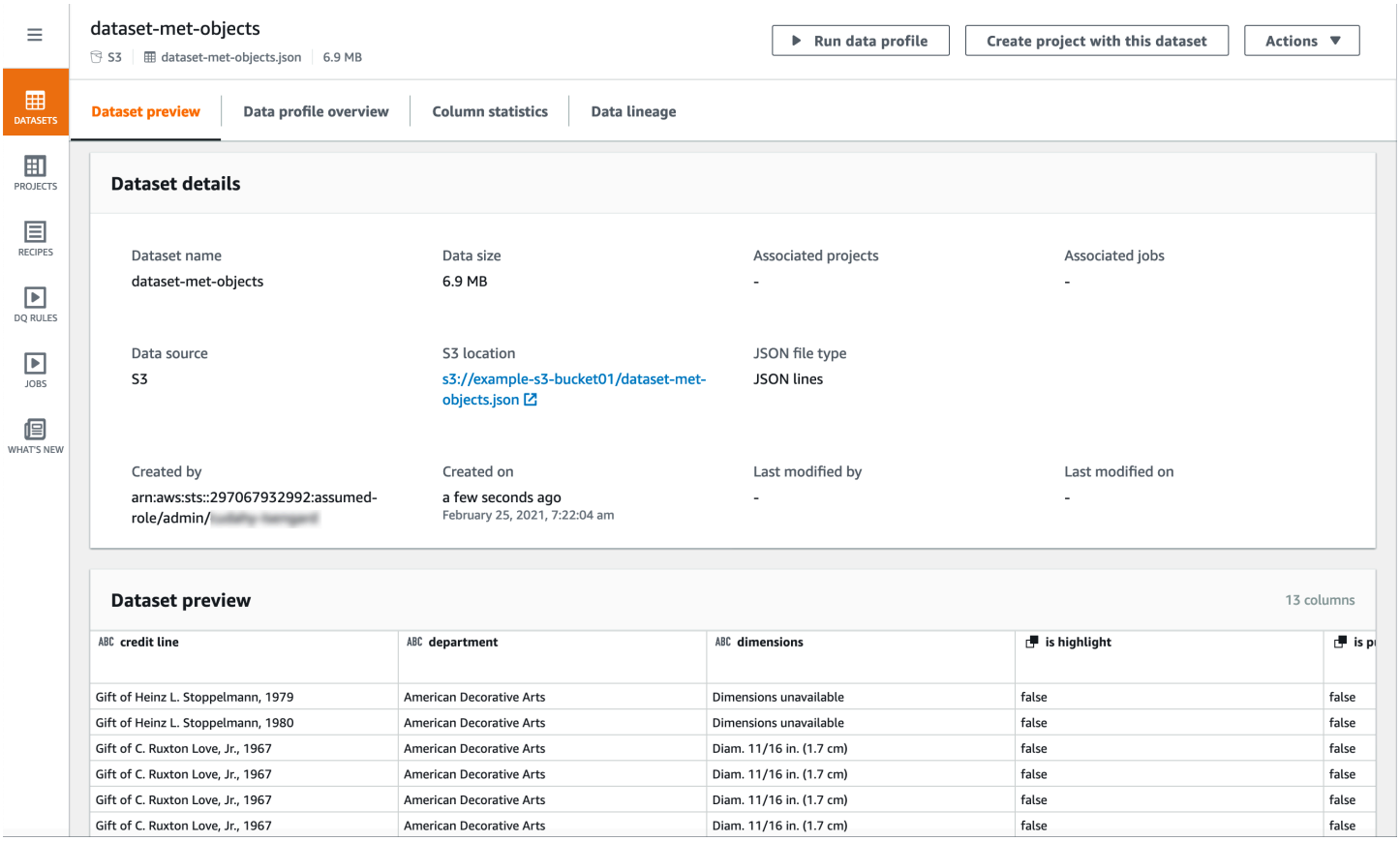
在中使用数据集 AWS Glue DataBrew

要在 DataBrew 控制台中查看您的数据集列表，请选择左侧的 D ATA SET。在数据集页面中，您可以通过单击每个数据集的名称或从其快捷菜单中选择 “操作”、“编辑” 来查看每个数据集的详细信息。

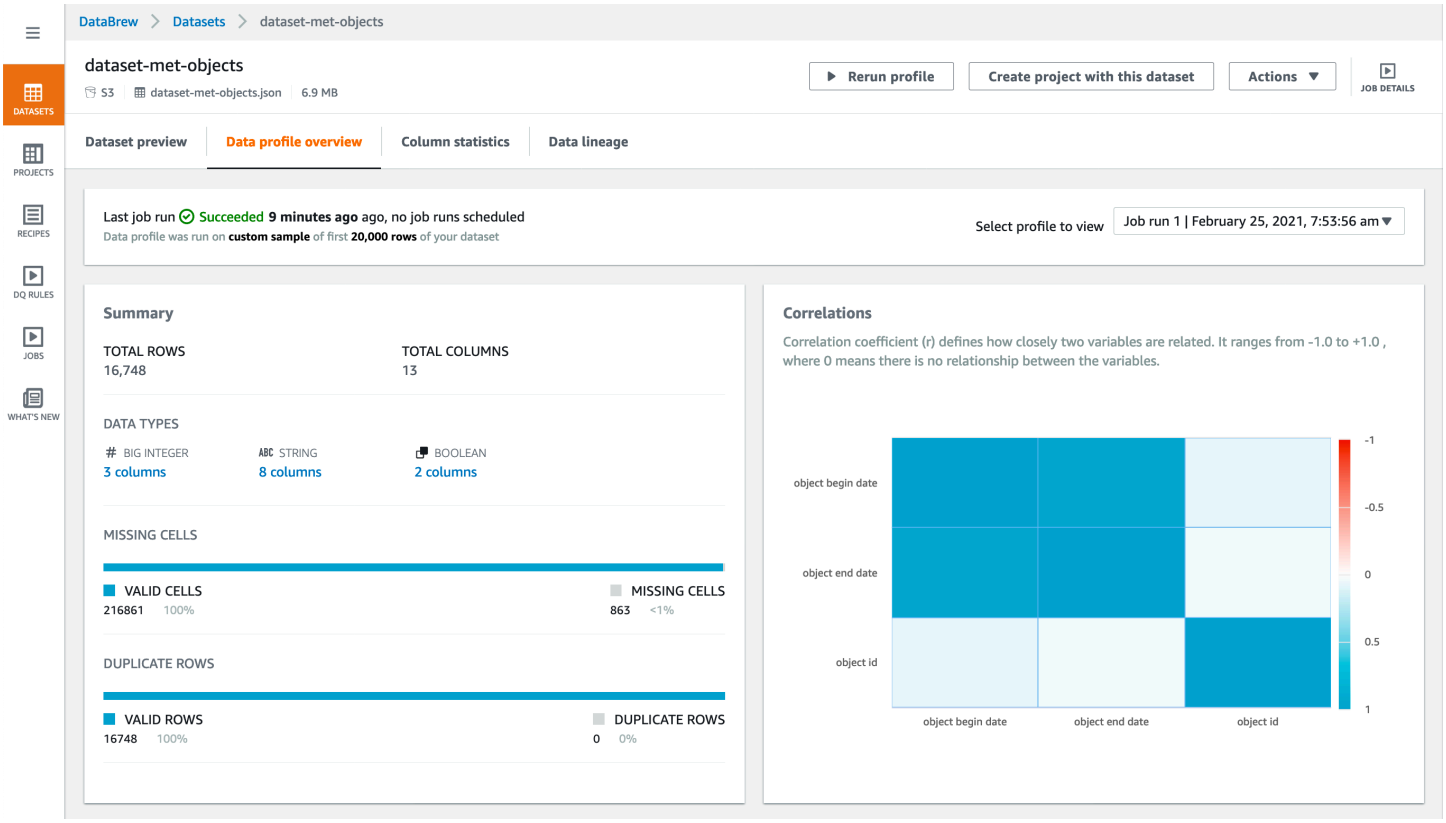
要创建新数据集，请选择 DATASE T、Connect 新数据集。不同的数据源具有不同的连接参数，您输入这些参数是为了 DataBrew 进行连接。保存连接并选择 “创建数据集” 后， DataBrew 将连接到您的数据并开始加载数据。有关更多信息，请参阅 连接到您的数据。

数据集页面包含以下元素，可帮助您浏览数据。

数据集预览-在此选项卡上，您可以找到数据集的连接信息以及数据集整体结构的概述，如下所示。

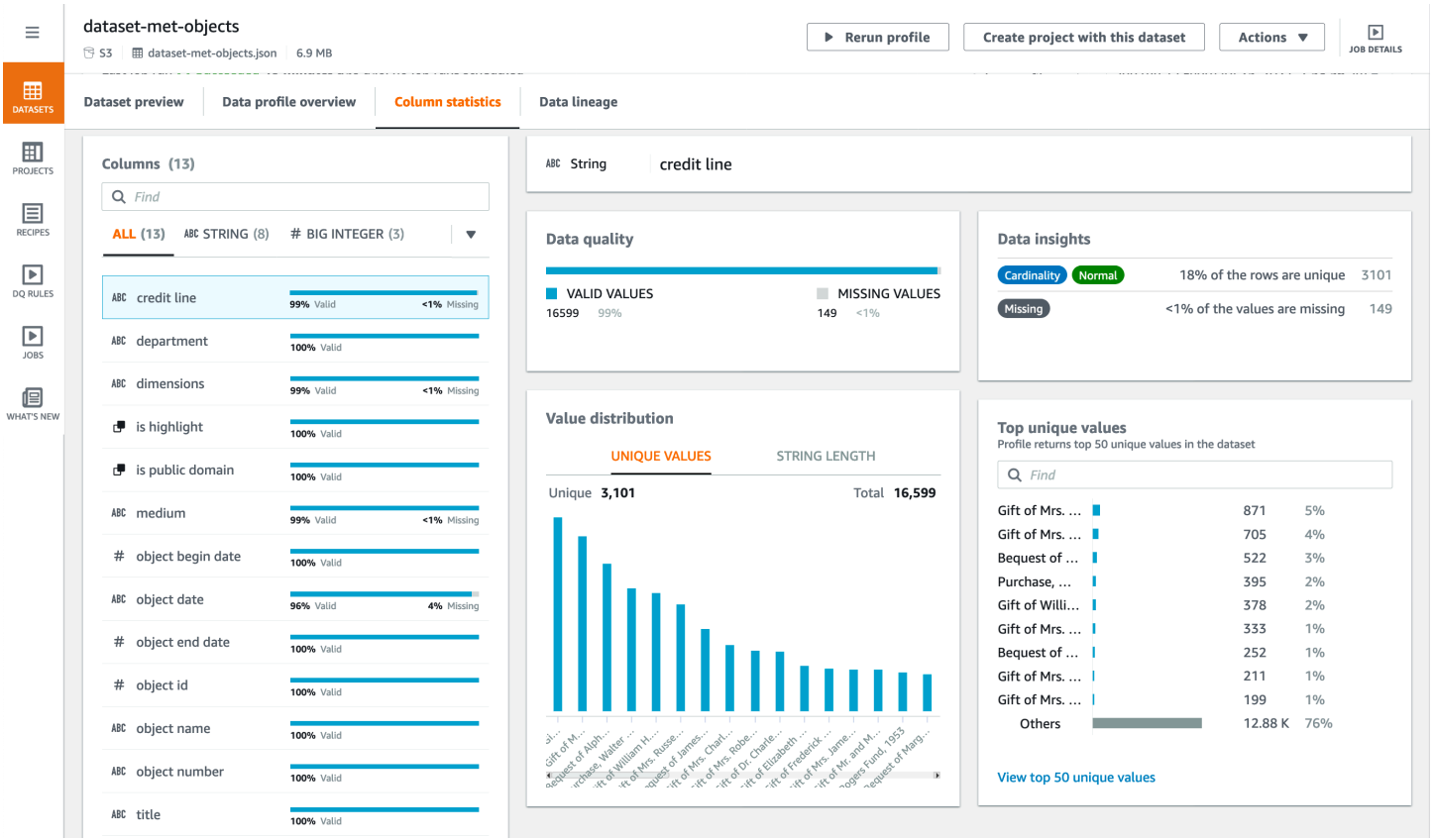


数据配置文件概述 — 在此选项卡上，您可以找到数据集的统计数据和体积的图形数据配置文件，如下所示。

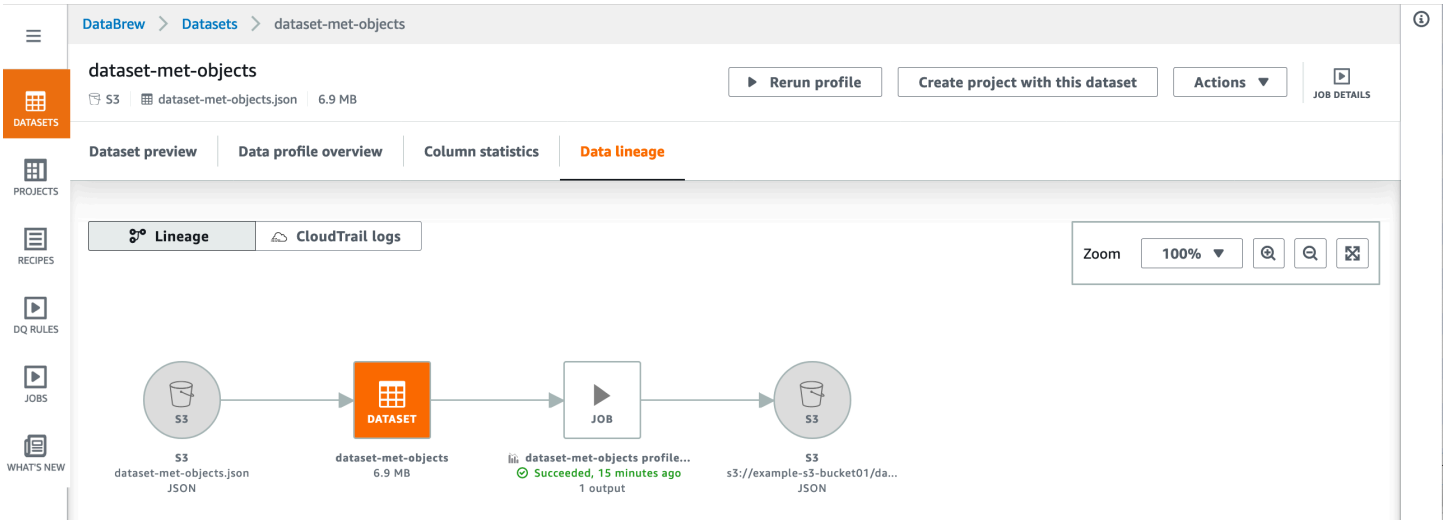


> **Note**
>
> 要创建数据配置文件，请对您的数据集运行 DataBrew 分析作业。有关如何执行此操作的信息，请参阅 步骤 5：创建数据配置文件。

列统计信息-在此选项卡上，您可以找到有关数据集中每列的详细统计信息，如下所示。



数据谱系-此选项卡以图形方式显示您的数据集是如何创建的，以及如何使用它 DataBrew，如下所示。



主题

- 删除数据集

删除数据集

如果您不再需要数据集，可以将其删除。删除数据集不会对底层数据源产生任何影响。它只是删除 DataBrew 用于访问数据源的信息。

如果有其他 DataBrew 资源依赖于某个数据集，则无法将其删除。例如，如果您当前有一个使用该数据集的 DataBrew 项目，请先删除该项目，然后再删除该数据集。

要删除数据集，请从导航窗格中选择数据集。选择要删除的数据集，然后在 “操作” 中选择 “删除”。

连接到您的数据

有关连接到以下数据源的更多信息，请选择适用于您的部分。

- AWS Glue Data Catalog— 您可以使用数据目录来定义对存储在 AWS 云中的数据对象的引用，包括以下服务：
  - Amazon Redshift
  - Aurora MySQL
  - Aurora PostgreSQL
  - Amazon RDS for MySQL
  - Amazon RDS for PostgreSQL

  DataBrew 识别应用于数据目录资源的所有 Lake Formation 权限，因此 DataBrew 用户只有在获得授权后才能访问这些资源。

  要创建数据集，需要指定数据目录数据库名称和表名。 DataBrew 处理其他连接细节。
- AWS Data Exchange — 您可以从 Data Exchange 中提供的数百种第三方 AWS 数据源中进行选择。通过订阅这些数据源，您始终可以获得最多的数据 up-to-date 版本。

  要创建数据集，您需要指定您已订阅或有权使用的 Data Exchange 数据产品的名称。
- JDBC 驱动程序连接-您可以通过连接到 JDBC 兼容的数据源 DataBrew 来创建数据集。 DataBrew 支持通过 JDBC 连接到以下来源：
  - Amazon Redshift
  - Microsoft SQL Server
  - MySQL
  - Oracle

- PostgreSQL
- Snowflake

主题

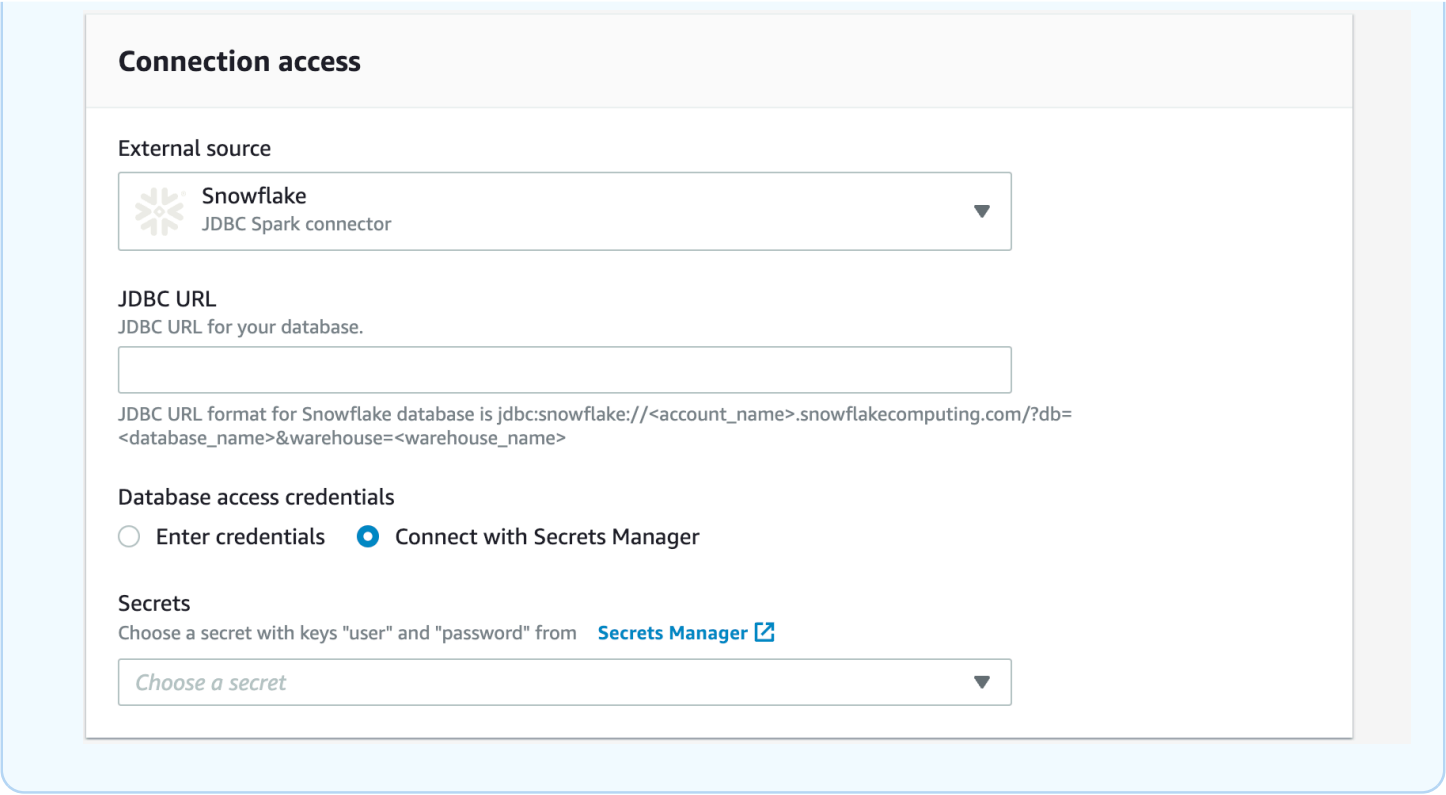
将驱动程序与 AWS Glue DataBrew

数据库驱动程序是实现数据库连接协议（例如 Java 数据库连接 (JDBC)）的文件或 URL。该驱动程序充当特定数据库管理系统 (DBMS) 和另一个系统之间的适配器或转换器。

在这种情况下，它允许 AWS Glue DataBrew 连接到您的数据。然后，您可以从支持的数据源访问数据库对象，例如表或视图。您正在使用的数据源可能称为数据库、数据仓库或其他东西。但是，在本文档中，我们将所有数据提供者称为数据源或连接。

要使用 JDBC 驱动程序或 jar 文件，请下载所需的一个或多个文件并将其放入 S3 存储桶中。用于访问数据的 IAM 角色需要对两个驱动程序文件具有读取权限。

> **Note**
>
> With AWS Glue4.0，本机支持作为数据源连接到 Snowflake。您无需提供自定义`jar`文件。
> 在中 AWS Glue DataBrew，选择 Snowflake 作为外部源连接，并提供您的 Snowflake 实例的 URL。URL 将使用表单 `https://account_identifier.snowflakecomputing.com` 中的主机名。
> 提供数据访问凭证、Snowflake 数据库名称和 Snowflake 架构名称。此外，如果您的 Snowflake 用户没有设置默认仓库，则需要提供仓库名称。
> Snowflake 连接使用 AWS Secrets Manager 密钥来提供凭据信息。您的项目和工作角色必须具有读取此密钥的权限。



要将驱动程序与 DataBrew

1. 使用产品提供的方法，找出您使用的是哪个版本的数据源。
2. 查找所需的最新版本的连接器和驱动程序。您可以在数据提供商网站上找到这些信息。
3. 下载所需版本的 JDBC 文件。这些文件通常存储为 Java 档案 (.JAR) 文件。
4. 要么将驱动程序从控制台上传到 S3 存储桶，要么提供.JAR 文件的 S3 路径。
5. 输入基本的连接详细信息，例如类别、实例等。
6. 输入您的数据源需要的任何其他配置信息，例如虚拟私有云 (VPC) 信息。

支持的 JDBC 驱动程序

产品	支持的版本	驱动程序说明和下载	支持 SQL 查询
	v6.x 或更高版本	适用于 SQL Server 的微软 JDBC 驱动程序	不支持

产品	支持的版本	驱动程序说明和下载	支持 SQL 查询
Microsoft SQL Server			
MySQL	v5.1 或更高版本	MySQL 连接器	不支持
Oracle	v11.2 或更高版本	Oracle JDBC 下载量	不支持
PostgreSQL	v4.2.x 或更高版本	PostgreSQL JDBC 驱动程序	不支持
Amazon Redshift	v4.1 或更高版本	使用 JDBC 连接亚马逊 Redshift	支持
Snowflake	要查看你的 Snowflake 版本，请按照 Snowflake 文档中的说明使用 CURRENTERSION。	要连接到 Snowflake，你需要以下两项： • Snowflake JDBC 驱动程序 • 适用于 Spark 的雪花连接器	支持

要连接到需要不同于 DataBrew 原生支持的驱动程序版本的数据库或数据仓库，您可以提供自己选择的 JDBC 驱动程序。该驱动程序必须与 JDK 8 或 Java 8 兼容。有关如何为您的数据库查找最新驱动程序版本的说明，请参阅将驱动程序与 AWS Glue DataBrew。

使用连接文本文件中的数据 DataBrew

您可以为 DataBrew 支持的输入文件配置以下格式选项：

- 逗号分隔值 (CSV) 文件
  - 分隔符

    .csv 文件的默认分隔符是逗号。如果您的文件使用不同的分隔符，请在创建数据集时在 “其他配置” 部分中选择 CSV 分隔符的分隔符。.csv 文件支持以下分隔符：
    - 逗号 (,)
    - 结肠 (:)
    - 分号 (;)
    - 竖线 (|)
    - 制表符 (\t)
    - Caret (^)
    - 反斜杠 (\)
    - 空格
  - 列标题值

    您的 CSV 文件可以包含标题行作为文件的第一行。如果不是，则为你 DataBrew 创建一个标题行。
    - 如果您的 CSV 文件包含标题行，请选择将第一行视为标题。如果这样做，则 CSV 文件的第一行将被视为包含列标题值。
    - 如果您的 CSV 文件不包含标题行，请选择添加默认标题。如果这样做，则会为文件 DataBrew 创建一个标题行，并且不会将您的第一行数据视为包含标题值。 DataBrew 创建的标题由下划线和文件中每列的数字组成`Column_1`，格式为`Column_2Column_3`、、等。
- JSON 文件

  DataBrew 支持 JSON 文件的两种格式，即 JSON 行和 JSON 文档。JSON 行文件每行包含一行。在 JSON 文档文件中，所有行都包含在单个 JSON 结构或数组中。创建 JSON 数据集时，可以在其他配置部分中指定您的 JSON 文件类型。默认格式为 JSON 行。

- Excel 文件

  以下内容适用于中的 Excel 工作表 DataBrew：

  - Excel 表格加载

    默认情况下， DataBrew 加载您的 Excel 文件中的第一张工作表。但是，创建 Excel 数据集时，您可以在 “其他配置” 部分中指定不同的表单编号或工作表名称。

  - 列标题值

    您的 Excel 工作表可以将标题行作为文件的第一行，但如果没有，则 DataBrew 会为您创建一个标题行。

    - 如果您的 Excel 工作表包含标题行，请选择 “将第一行视为标题”。如果这样做，Excel 工作表的第一行将被视为包含列标题值。
    - 如果您的 Excel 文件不包含标题行，请选择 “添加默认标题”。通过这样做，您可以指定 DataBrew应该为文件创建标题行，而不是将您的第一行数据视为包含标题值。 DataBrew创建的标题由下划线和文件中每列的数字组成`Column_1`，格式为`Column_2Column_3`、、等。

连接 Amazon S3 中多个文件中的数据

通过 DataBrew 控制台，您可以浏览 Amazon S3 存储桶和文件夹，并为数据集选择文件。但是，一个数据集不必局限于一个文件。

假设您有一个名为的 S3 存储桶`my-databrew-bucket`，其中包含一个名为的文件夹`databrew-input`。在该文件夹中，假设您有许多 JSON 文件，所有文件格式和`.json`文件扩展名都相同。在控制台上，您可以将源 URL 指定为`s3://my-databrew-bucket/databrew-input/`。然后，您可以在 DataBrew 控制台上选择此文件夹。您的数据集由该文件夹中的所有 JSON 文件组成。

DataBrew 可以处理 S3 文件夹中的所有文件，但前提是满足以下条件：

- 该文件夹中的所有文件都具有相同的格式。
- 该文件夹中的所有文件都具有相同的文件扩展名。

有关支持的文件格式和扩展名的更多信息，请参阅DataBrew input formats。

使用多个文件作为数据集时的架构

当使用多个文件作为 DataBrew 数据集时，所有文件的架构必须相同。否则，Project Workspace 会自动尝试从多个文件中选择一个架构，并尝试使其余的数据集文件符合该架构。这种行为会导致在 Project Workspace 期间显示的视图不规则，因此，作业输出也将是不规则的。

如果您的文件必须具有不同的架构，则需要创建多个数据集并分别对其进行分析。

使用适用于 Amazon S3 的参数化路径

在某些情况下，您可能需要创建一个包含遵循特定命名约定的文件的数据集，或者创建可以跨越多个 Amazon S3 文件夹的数据集。或者，您可能希望将相同的数据集重复用于在 S3 位置定期生成的结构相同的数据，其路径依赖于某些参数。例如，以数据生成日期命名的路径。

DataBrew 通过参数化的 S3 路径支持这种方法。参数化路径是包含正则表达式或自定义路径参数（或两者兼而有之）的 Amazon S3 网址。

使用正则表达式定义带有 S3 路径的数据集

路径中的正则表达式可用于匹配一个或多个文件夹中的多个文件，同时筛选出这些文件夹中不相关的文件。

以下是几个例子：

- 定义一个数据集，其中包含名称以开头的文件夹中的所有 JSON 文件`invoice`。
- 定义一个数据集，该数据集包含文件夹中的所有文件，其名称`2020`中包含这些文件。

您可以通过在数据集 S3 路径中使用正则表达式来实现这种方法。这些正则表达式可以替换 S3 URL 密钥中的任何子字符串（但不能替换存储桶名称）。

作为 S3 URL 中密钥的示例，请参阅以下内容。这里，`my-bucket`是存储桶名称，美国东部（俄亥俄州）是 AWS 区域，`puppy.png`是密钥名称。

```
https://my-bucket.s3.us-west-2.amazonaws.com/puppy.png
```

在参数化的 S3 路径中，两个尖括号（<和>）之间的任何字符都被视为正则表达式。以下是两个示例：

- `s3://my-databrew-bucket/databrew-input/invoice<.*>/data.json`匹配名称以`data.json`开头的所有子文件夹中`databrew-input`名为的所有文件。`invoice`

- s3://my-databrew-bucket/databrew-input/<.*>2020<.*>/将文件夹中的所有文件与2020其名称相匹配。

在这些示例中，.*匹配零个或多个字符。

> **Note**
>
> 您只能在 S3 路径的关键部分（存储桶名称之后的部分）中使用正则表达式。因此，s3://my-databrew-bucket/<.*>-input/是有效的，但s3://my-<.*>-bucket/<.*>-input/不是。

我们建议您测试正则表达式，确保它们仅匹配您想要的 S3 网址，而不匹配您不想要的 S3 网址。

以下是其他一些正则表达式的示例：

- <\d{2}>匹配恰好由两个连续数字组成的字符串，例如07或03，但不是1a2。
- <[a-z]+.*>匹配以一个或多个小写拉丁字母开头且后面有零个或多个其他字符的字符串。一个例子是a3abc/def、或a-z，但不是A2。
- <[^/]+>匹配包含除斜杠 (/) 之外的任何字符的字符串。在 S3 URL 中，使用斜杠来分隔路径中的文件夹。
- <.*=.*>匹配包含等号 (=) 的字符串，例如、或 month=02 abc/day=2=10，但不是。test
- <\d.*\d>匹配以数字开头和结尾且数字之间可以包含任何其他字符的字符串，例如1abc201-02-032020/Jul/21、或，但不是123a。

使用自定义参数定义带有 S3 路径的数据集

当您可能想为 S3 位置提供参数时，使用自定义参数定义参数化数据集比使用正则表达式更具优势：

- 您可以获得与使用正则表达式相同的结果，而无需知道正则表达式的语法。您可以使用诸如 “开头为” 和 “包含” 之类的熟悉术语来定义参数。
- 使用路径中的参数定义动态数据集时，可以在定义中包含一个时间范围，例如 “过去一个月” 或 “过去 24 小时”。这样，您的数据集定义将在以后用于处理新的传入数据。

以下是您何时可能想要使用动态数据集的一些示例：

- 将按上次更新日期或其他有意义的属性分区的多个文件连接到单个数据集中。然后，您可以将这些分区属性捕获为数据集中的其他列。
- 将数据集中的文件限制在满足特定条件的 S3 位置。例如，假设您的 S3 路径包含基于日期的文件夹，例如`folder/2021/04/01/`。在这种情况下，您可以参数化日期并将其限制在某个范围内，例如 “2021年3月1日至2021年4月1日之间” 或 “过去一周”。

要使用参数定义路径，请使用以下格式定义参数并将其添加到路径中：

```
s3://my-databrew-bucket/some-folder/{parameter1}/file-{parameter2}.json
```

> **Note**
>
> 与 S3 路径中的正则表达式一样，您只能在路径的关键部分（存储桶名称之后的部分）中使用参数。

参数定义中需要两个字段，即名称和类型。类型可以是字符串、数字或日期。日期类型的参数必须定义日期格式，这样 DataBrew 才能正确解释和比较日期值。或者，您可以为参数定义匹配条件。当 DataBrew 作业或交互式会话加载数据集时，您也可以选择将参数的匹配值作为列添加到数据集中。

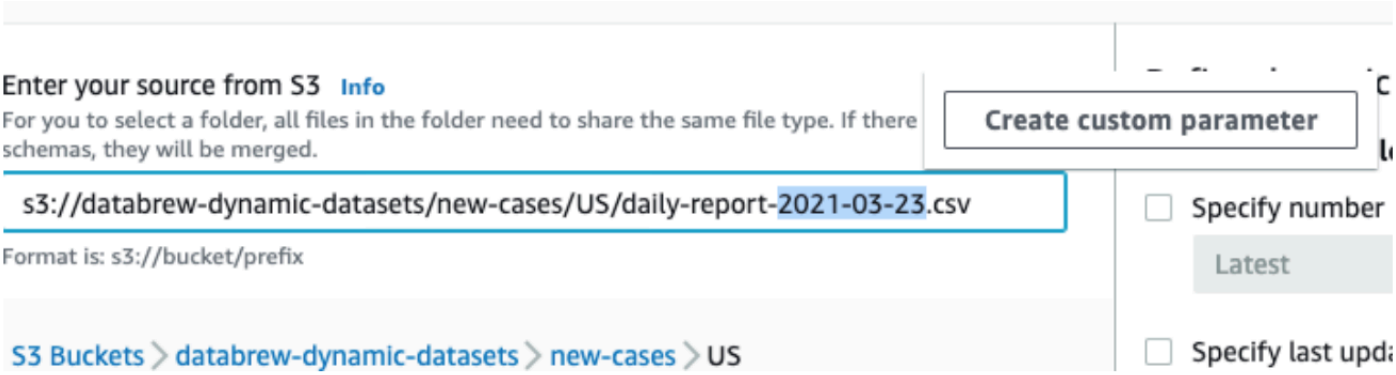
示例

让我们考虑一个使用 DataBrew 控制台中的参数定义动态数据集的示例。在此示例中，假设输入数据定期使用以下位置写入 S3 存储桶：

- `s3://databrew-dynamic-datasets/new-cases/UR/daily-report-2021-03-30.csv`
- `s3://databrew-dynamic-datasets/new-cases/UR/daily-report-2021-03-31.csv`
- `s3://databrew-dynamic-datasets/new-cases/US/daily-report-2021-03-30.csv`
- `s3://databrew-dynamic-datasets/new-cases/US/daily-report-2021-03-31.csv`

这里有两个动态部分：国家代码（如美国）和文件名中的日期（如 2021-03-30）。在这里，您可以对所有文件应用相同的清理方法。假设你想每天执行清理工作。以下是如何为此场景定义参数化路径：

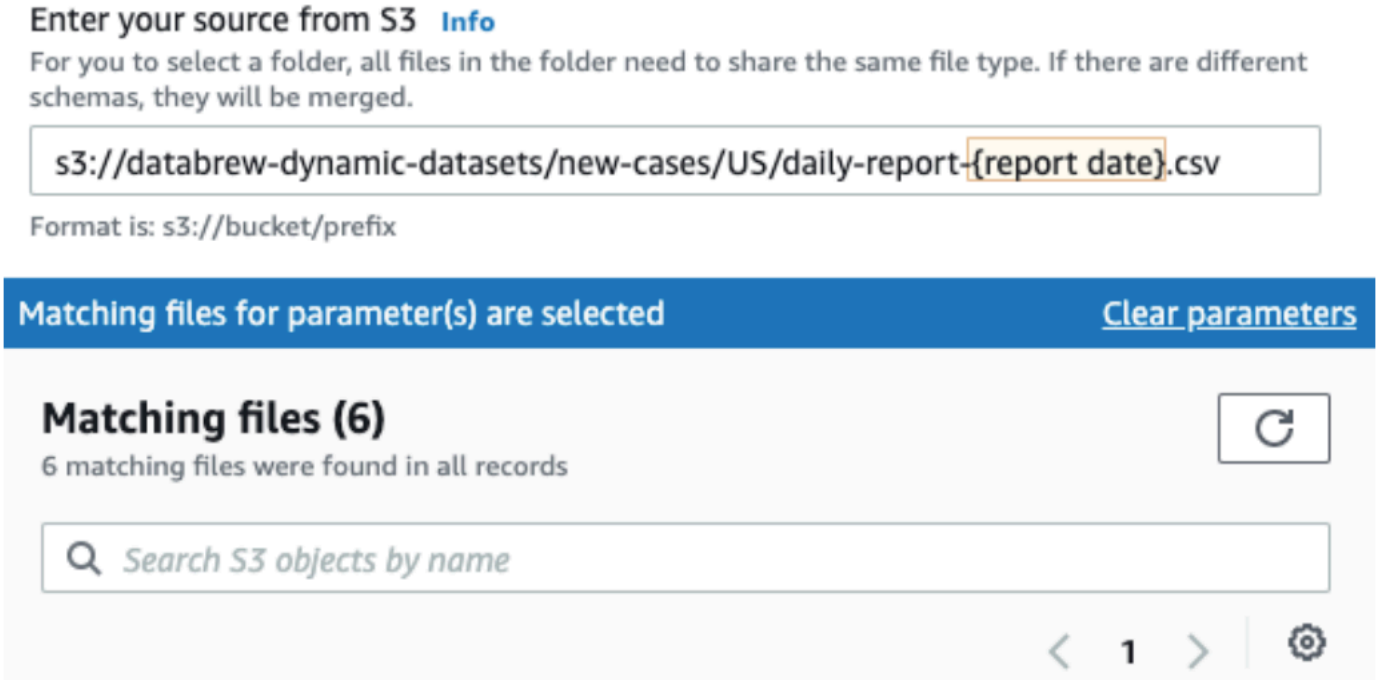
1. 导航到特定文件。
2. 然后选择一个不同的部分，例如日期，并将其替换为参数。在这种情况下，请替换日期。



3. 打开 “创建自定义参数” 的上下文（右键单击）菜单并为其设置属性：
   - 名称：报告日期
   - 类型：日期
   - 日期格式：yyyy-mm-dd（从预定义的格式中选择）
   - 条件（时间范围）：过去 24 小时
   - 添加为列：true（已选中）

   将其他字段保留为其默认值。
4. 选择创建。

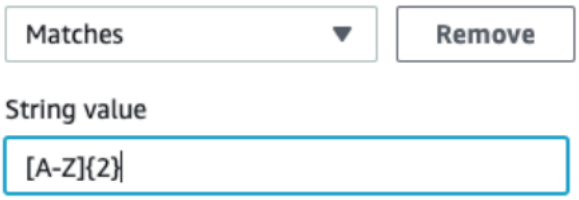
完成后，您将看到更新的路径，如以下屏幕截图所示。



现在，您可以对国家/地区代码执行同样的操作，并按如下方式对其进行参数化：

- 名称：国家/地区代码
- 类型：字符串
- 添加为列：true（已选中）

如果所有值都相关，则不必指定条件。例如，在该new-cases文件夹中，我们只有带有国家/地区代码的子文件夹，因此无需附加条件。如果您还有其他要排除的文件夹，则可以使用以下条件。



这种方法将新案例的子文件夹限制为包含两个大写拉丁字符。

完成此参数化后，您的数据集中只有匹配的文件，可以选择 “创建数据集”。

> **Note**
>
> 在条件中使用相对时间范围时，将在加载数据集时计算时间范围。无论它们是 “过去 24 小时” 之类的预定义时间范围，还是像 “5 天前” 这样的自定义时间范围，都是如此。无论数据集是在交互式会话初始化期间加载还是在作业启动期间加载，这种评估方法都适用。

选择 “创建数据集” 后，您的动态数据集就可以使用了。例如，您可以先使用它来创建项目，然后使用交互式 DataBrew 会话定义清理方案。然后，您可以创建一个计划每天运行的作业。此作业可能会将清理方法应用于作业开始时满足参数条件的数据集文件。

动态数据集支持的条件

您可以使用条件通过参数或上次修改日期属性筛选匹配的 S3 文件。

下面，您可以找到每种参数类型的支持条件列表。

与字符串参数一起使用的条件

DataBrew SDK 中的名称	SDK 同义	DataBrew 控制台中的名称	描述
为	eq，==	正是	参数的值与条件中提供的值相同。
不是	不是 eq，！ =	Is not	参数的值与条件中提供的值不同。
contains		包含	参数的字符串值包含条件中提供的值。
不包含		不包含	参数的字符串值不包含条件中提供的值。
开头_with		开始于	参数的字符串值以条件中提供的值开头。
不是以 _with 开头		开头不是	参数的字符串值不是以条件中提供的值开头。
结束_with		结束于	参数的字符串值以条件中提供的值结尾。
不结尾_with		结尾不是	参数的字符串值不以条件中提供的值结尾。
匹配		匹配项	参数的值与条件中提供的正则表达式相匹配。
不匹配		不匹配	参数的值与条件中提供的正则表达式不匹配。

> **Note**
>
> String 参数的所有条件都使用区分大小写的比较。如果您不确定 S3 路径中使用的大小写，则可以将 “matches” 条件与以开头的正则表达式值一起使用`(?i)`。这样做会导致不区分大小写的比较。
> 例如，假设您希望字符串参数以开头`abc`，但也可以`ABC`使用`Abc`或。在这种情况下，您可以使用 “匹配” 条件`(?i)^abc`作为条件值。

与数字参数一起使用的条件

DataBrew SDK 中的名称	SDK 同义	DataBrew 控制台中的名称	描述
为	eq，==	正是	参数的值与条件中提供的值相同。
不是	不是 eq，！ =	Is not	参数的值与条件中提供的值不同。
小于	lt，<	Less than	参数的数值小于条件中提供的值。
小于等于	lte，<=	小于或等于	参数的数值小于或等于条件中提供的值。
大于	gt，>	Greater than	参数的数值大于条件中提供的值。
大于等于	gete，>=	大于或等于	参数的数值大于或等于条件中提供的值。

与日期参数一起使用的条件

DataBrew SDK 中的名称	DataBrew 控制台中的名称	条件值格式 (SDK)	描述
之后	启动	ISO 8601 日期格式，比如或 `2021-03-3`	日期参数的值晚于条件中提供的日期。

DataBrew SDK 中的名称	DataBrew 控制台中的名称	条件值格式 (SDK)	描述
		0T01:00:0 0Z 2021-03-3 0T01:00-07:00	
以前	结束	ISO 8601 日期格式，比如或 2021-03-3 0T01:00:0 0Z 2021-03-3 0T01:00-07:00	日期参数的值早于条件中提供的日期。
relative_aft	开始（相对）	正数或负数的时间单位，如-48h或+7d。	日期参数的值晚于条件中提供的相对日期。 相对日期是在数据集加载时计算的，无论是在初始化交互式会话时，还是在启动关联作业时。这是示例中称为 “现在” 的时刻。
相对之前	结束（相对）	正数或负数的时间单位，如-48h或+7d。	日期参数的值早于条件中提供的相对日期。 相对日期是在数据集加载时计算的，无论是在初始化交互式会话时，还是在启动关联作业时。这是示例中称为 “现在” 的时刻。

如果您使用SDK，请按以下格式提供相对日期：`±{number_of_time_units}{time_unit}`。您可以使用以下时间单位：

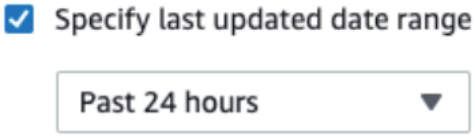
- -1h（1 小时前）
- +2d（2 天后）
- -120 米（120 分钟前）
- 5000（从现在起 5,000 秒）
- -3w（3 周前）
- +400 万（4 个月后）
- -1年（1 年前）

相对日期是在数据集加载时计算的，无论是在初始化交互式会话时，还是在启动关联作业时。在前面的例子中，这个时刻被称为 “现在”。

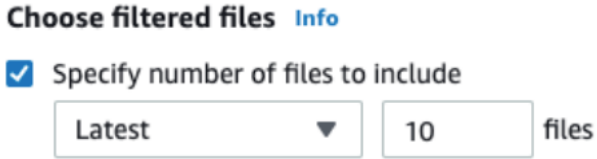
配置动态数据集的设置

除了提供参数化的 S3 路径外，您还可以为包含多个文件的数据集配置其他设置。这些设置按上次修改日期筛选 S3 文件并限制文件数量。

与在路径中设置日期参数类似，您可以定义更新匹配文件的时间范围，并仅将这些文件包含到数据集中。您可以使用绝对日期（例如 “2021 年 3 月 30 日”）或相对范围（例如 “过去 24 小时”）来定义这些范围。



要限制匹配文件的数量，请选择大于 0 的文件数量，以及您想要的是最新还是最旧的匹配文件。



数据类型

数据集中每列的数据将转换为以下数据类型之一：

- byte — 1 字节有符号整数。数字的范围从 -128 到 127 不等。

- s@@ hor t — 2 字节有符号整数。数字的范围从 -32768 到 32767 不等。
- 整数 — 4 字节有符号整数。数字范围从 -2147483648 到 2147483647。
- long — 8 字节的有符号整数。数字范围从-9223372036854775808到9223372036854775807。
- float — 4 字节的单精度浮点数。
- doub@@ le — 8 字节的双精度浮点数。
- 十进制 — 带符号的十进制数字，总数最多 38 位数，小数点后最多 18 位数。
- 字符串-字符串值。
- boolean — 布尔类型有两个可能的值之一：“真” 和 “假” 或 “是” 和 “否”。
- timestamp — 包含年、月、日、小时、分钟和秒等字段的值。
- 日期-包含年、月和日字段的值。

高级数据类型

高级数据类型是在项目的字符串列中 DataBrew 检测到的数据类型，因此不是数据集的一部分。有关高级数据类型的信息，请参阅高级数据类型。

高级数据类型

高级数据类型是通过模式匹配在项目的字符串列中进行 DataBrew 检测的数据类型。当您单击字符串列时，如果该列中有 50% 或更多的值符合该数据类型的标准，则该列将被标记为相应的高级数据类型。

DataBrew 可以检测的数据类型有：

- 日期/时间戳
- SSN
- 电话号码
- 电子邮件地址
- 信用卡
- 性别
- IP 地址
- URL
- 邮政编码
- Country

- 货币
- 省/自治区/直辖市
- City（城市）

您可以使用以下转换来处理高级数据类型：

- GET_ADVANCED_DATATYPE：给定一个字符串列，标识该列的高级数据类型（如果有）。
- 提取_高级_数据类型_详细信息：提取高级数据类型的详细信息。
- 高级数据类型过滤器：根据高级数据类型检测筛选当前源列。
- ADVANCED_DATATYPE_FLAG：根据当前源列的值创建新的标志列。

验证中的数据质量 AWS Glue DataBrew

为确保数据集的质量，可以在规则集中定义数据质量规则列表。规则集是一组规则，用于将不同的数据指标与预期值进行比较。如果未满足任何规则的标准，则整个规则集将无法通过验证。然后，您可以检查每条规则的单独结果。对于任何导致验证失败的规则，您可以进行必要的更正并重新验证。

以下为规则示例：

- 列中的值介"APY"于 0 和 100 之间
- 列中缺失值的数量group_name不超过 5%

您可以为单个列定义每条规则，也可以将其单独应用于多个选定列，例如：

- "rate"、"pay"列的最大值不超过 100 "increase"。

一个规则可以由多个简单的检查组成。你可以定义它们是全部为真还是任意，例如：

- 列中的值"ProductId"应以 AND 开头"asin-"，列中值的长度"ProductId"为 32。

您可以根据聚合值（例如maxmin、或）仅比较一个值的number of duplicate values聚合值或列中每行中的非聚合值来验证规则。在后一种情况下，您还可以定义 “通过” 阈值，例如value in columnA > value in columnB for at least 95% of rows。

与配置文件信息一样，您只能为简单类型的列（例如字符串和数字）定义列级数据质量规则。您无法为复杂类型的列（例如数组或结构）定义数据质量规则。有关使用个人资料信息的更多详细信息，请参阅创建和使用 AWS Glue DataBrew 个人资料职位。

验证数据质量规则

定义规则集后，您可以将其添加到配置文件作业中进行验证。您可以为一个数据集定义多个规则集。

例如，一个规则集可能包含具有最低可接受标准的规则。该规则集的验证失败可能意味着该数据不可进一步使用。例如，用于机器学习训练的数据集的关键列中缺少值。您可以使用具有更严格规则的第二个规则集来验证数据集的质量是否如此之好，以至于不需要清理。

您可以在配置文件作业配置中应用为给定数据集定义的一个或多个规则集。配置文件作业运行时，除了数据配置文件外，它还会生成验证报告。验证报告与您的个人资料数据位于同一位置。与个人资料信

息一样，您可以在 DataBrew控制台中浏览结果。在数据集详细信息视图中，选择数据质量选项卡以查看结果。有关使用个人资料信息的更多详细信息，请参阅创建和使用 AWS Glue DataBrew 个人资料职位。

根据验证结果执行操作

DataBrew 配置任务完成后， DataBrew 会发送一个 Amazon CloudWatch 事件，其中包含该任务运行的详细信息。如果您还将作业配置为验证数据质量规则，则会为每个经过验证的规则集 DataBrew发送一个事件。该事件包含其结果（SUCCEEDEDFAILED、或ERROR）以及指向详细数据质量验证报告的链接。然后，您可以根据验证状态通过调用下一个操作来自动执行进一步的操作。有关将事件与目标操作（例如 Amazon SNS 通知、 AWS Lambda 函数调用等）关联的更多信息，请参阅 Amazon 入门。EventBridge

下面是 DataBrew 验证结果事件的示例：

```
{
  "version": "0",
  "id": "fb27348b-112d-e7c2-560d-85e7c2c09964",
  "detail-type": "DataBrew Ruleset Validation Result",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2021-11-18T13:15:46Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "datasetName": "MyDataset",
    "jobName": "MyProfileJob",
    "jobRunId": "db_f07954d20d083de0c1fc1eee11498d8635ee5be4ca416af27d33933e91ff4e6e",
    "rulesetName": "MyRuleset",
    "validationState": "FAILED",
    "validationReportLocation": "s3://MyBucket/MyKey/
MyDataset_f07954d20d083de0c1fc1eee11498d8635ee5be4ca416af27d33933e91ff4e6e_dq-
validation-report.json"
  }
}
```

您可以在 Amazon Eventbridge 中使用事件的detail属性（例如）source和属性的嵌套属性来创建事件模式。detail-type例如，匹配任何 DataBrew 作业中所有失败验证的事件模式如下所示：

```
{
  "source": ["aws.databrew"],
```

```
  "detail-type": ["DataBrew Ruleset Validation Result"],
  "detail": {
   "validationState": ["FAILED"]
  }
}
```

有关创建规则集并验证其规则的示例，请参阅。创建包含数据质量规则的规则集有关在中处理 CloudWatch 事件的更多信息 DataBrew，请参阅 DataBrew 使用 CloudWatch 事件自动化

创建包含数据质量规则的规则集

在以下过程中，您可以找到创建规则集并将其应用于数据集的示例。规则集是一组规则，用于将不同的数据指标与预期值进行比较。然后，您可以在配置文件作业中使用此规则集来验证其中包含的数据质量规则。

创建包含数据质量规则的示例规则集

1. 登录 AWS Management Console 并打开 DataBrew 控制台，网址为 https://console.aws.amazon.com/databrew/。
2. 从导航窗格中选择 DQ 规则，然后选择创建数据质量规则集。
3. 输入规则集的名称。（可选）输入规则集的描述。
4. 在“关联的数据集”下，选择要与规则集关联的数据集。

   选择数据集后，可以在右侧查看数据集预览窗格。
5. 在确定要创建的数据质量规则时，使用数据集预览窗格中的预览来浏览数据集的值和架构。预览可以让你深入了解数据可能存在的潜在问题。

   某些数据源（例如数据库）不支持数据预览。在这种情况下，您无需先验证数据质量规则即可运行分析作业。然后，您可以使用数据配置文件获取有关数据架构和值分布的信息。
6. 查看“建议”选项卡，其中列出了一些可以在创建规则集时使用的规则建议。您可以选择全部、部分或不选择任何推荐。

   选择相关建议后，选择“添加到规则集”。

   这会将规则添加到您的规则集中。如有必要，请检查和修改参数。请注意，数据质量规则中只能使用简单类型的列，例如字符串、数字和布尔值。
7. 选择“添加其他规则”以添加建议未涵盖的规则。您可以更改规则名称，以便日后更容易解释验证结果。

8. 使用数据质量检查范围来选择是在此规则中每次检查时选择单个列，还是将其应用于您选择的一组列。例如，如果您的数据集有多个数值列，其值应介于 0 和 100 之间，则可以定义一次规则，然后选择所有要按此规则检查的列。
9. 如果您的规则将有多项检查，则在规则成功标准下拉列表中，选择是否应满足所有检查或哪些检查符合标准。
10. 在数据质量检查下拉列表中选择要执行的检查以验证此规则。有关可用支票的更多信息，请参阅可用的支票。
11. 如果您为数据质量检查范围中的每列选择了单独检查，请选择一列。选择或键入此支票的列名。
12. 根据检查选择参数。有些条件仅接受提供的自定义值，有些条件还支持引用另一列。
13. 如果您选择检查列值，例如字符串值的包含条件，则可以指定 “通过” 阈值。例如，如果您希望至少 95% 的值满足该条件，则需要选择大于等于作为阈值的条件，输入 95 作为阈值，然后在 “阈值” 部分的下一个下拉列表中保留 “%（百分比）行”。或者，如果您希望不超过 10 行，其中缺少值的条件为真，则可以选择 “小于等于” 作为条件，在 “阈值” 中输入 10，然后在下一个下拉列表中选择行。请注意，如果您在验证期间使用不同大小的样本，可能会得到不同的结果。
14. 如果需要，可以添加更多规则。
15. 选择 “创建规则集”。

使用规则集创建个人资料作业

如前所述，创建规则集后，您将被定向到数据质量规则页面，其中显示了您账户中的所有规则集。

创建包含规则集的配置文件作业

1. 选择您之前创建的规则集的名称以查看其详细信息。
2. 选择使用规则集创建个人资料作业。

   Job 名称会自动填充，但您可以根据需要进行更改。
3. 对于 Job 运行示例，您可以选择运行整个数据集或有限数量的行。

   如果您选择运行有限的样本量，请注意，对于某些规则，结果可能会与完整数据集有所不同。
4. 在任务输出设置中，为任务输出选择一个 S3 位置。选择您有权访问的已命名的 Amazon S3 存储桶中的任何文件夹。如果您为此存储桶输入的文件夹名称不存在，则会创建此文件夹。

   成功完成配置文件作业后，此文件夹将包含 JSON 格式的数据和数据质量规则验证报告的配置文件。
5. 在数据质量规则下，请注意您的规则集列在数据质量规则集名称下。

6. 在“权限”下，选择或创建角色以授予 DataBrew 从输入 Amazon S3 位置读取和写入任务输出位置的权限。如果您还没有准备好角色，请选择创建新的 IAM 角色。
7. 如有必要，请按中所创建和使用 AWS Glue DataBrew 个人资料职位述修改任何其他可选设置。
8. 选择“创建并运行作业”。

检查验证结果并更新数据质量规则

配置文件任务完成后，您可以查看数据质量规则的验证结果，并根据需要更新规则。

要查看数据质量规则的验证数据

1. 在 DataBrew 控制台上，选择查看数据配置文件。这样做会显示数据集的数据配置文件概述选项卡。
2. 选择数据质量规则选项卡。在此选项卡上，您可以查看所有数据质量规则的结果。
3. 选择单个规则，了解有关该规则的更多详细信息。

对于任何未通过验证的规则，您可以进行必要的更正。

更新数据质量规则

1. 在导航窗格上，选择 DQ 规则。
2. 在数据质量规则集名称下，选择包含您计划编辑的规则的数据集。
3. 选择要更改的规则，然后选择编辑。
4. 进行必要的更正，然后选择“更新规则集”。
5. 重新运行作业。重复此过程，直到所有验证通过。

可用的支票

下表列出了可在您的规则中使用的所有可用条件的参考资料。请注意，聚合条件不能与同一规则中的非聚合条件组合。

> **Note**
>
> 对于 SDK 用户，要将相同的规则应用于多个列，请使用规则的ColumnSelectors属性，并使用名称或正则表达式指定经过验证的列。在这种情况下，你应该使用隐式CheckExpression。例如，“> :val”将每个选定列中的值与提供的值进行比较。 DataBrew 使用隐式语

法FilterExpression在动态数据集中进行定义。如果要为每项检查单独指定列，请不要设置该ColumnSelectors属性。相反，请提供一个明确的表达式。例如，“:col > :val”如规则CheckExpression中的 a。

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
汇总数据集条件	行数		与自定义值的数值比较	`"CheckExpression": "AGG(ROWS_COUNT) > :val", "SubstitutionMap": {":val", "10000"}`
	列数		与自定义值的数值比较	`"CheckExpression": "AGG(COLUMNS_COUNT) == :val", "SubstitutionMap": {":val", "20"}`
	重复行		与自定义值的数值比较	`"CheckExpression": "AGG(DUPLICATE_ROWS_COUNT) < :val", "SubstitutionMap": {":val", "100"}`

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
				或者 "CheckExpression": "AGG(DUPLICATE_ROWS_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
汇总列统计数据	缺失值		与自定义值的数值比较	"CheckExpression": "AGG(MISSING_VALUES_COUNT) < :val", "SubstitutionMap": {":val", "100"} 或者 "CheckExpression": "AGG(MISSING_VALUES_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
	重复值		与自定义值的数值比较	"CheckExpression": "AGG(DUPLICATE_VALUES_COUNT) < :val", "SubstitutionMap": {":val", "100"} 或者 "CheckExpression": "AGG(DUPLICATE_VALUES_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
	有效值		与自定义值的数值比较	"CheckExpression": "AGG(VALID_VALUES_COUNT) > :val", "SubstitutionMap": {":val", "10000"} 或者 "CheckExpression": "AGG(VALID_VALUES_PERCENTAGE) > :val", "SubstitutionMap": {":val", "95"}

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
	DISCT 值		与自定义值的数值比较	"CheckExpression": "AGG(DISTINCT_VALUES_COUNT) > :val", "SubstitutionMap": {":val", "1000"} 或者 "CheckExpression": "AGG(DISTINCT_VALUES_PERCENTAGE) >= :val", "SubstitutionMap": {":val", "50"}

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
	唯一值		与自定义值的数值比较	"CheckExpression": "AGG(UNIQUE_VALUES_COUNT) > :val", "SubstitutionMap": {":val", "100"} 或者 "CheckExpression": "AGG(UNIQUE_VALUES_PERCENTAGE) > :val", "SubstitutionMap": {":val", "20"}

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
	异常值	Z 分数阈值	与自定义值的数值比较	"CheckExpression": "AGG(Z_SCORE_OUTLIERS_COUNT, :zscore_dev) < :val", "SubstitutionMap": {":zscore_dev": "4", ":val", "100"} 或者 "CheckExpression": "AGG(Z_SCORE_OUTLIERS_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例

	价值分布统计	统计数据名称（参见下表）	与自定义值的数值比较	`"CheckExpression": "AGG(<STAT_NAME>) < :val", "SubstitutionMap": {":val", "100"}`
				
				或者
				
				`"CheckExpression": "AGG(<STAT_NAME>, :param) < :val", "SubstitutionMap": {":param": "0.25", :val", "5"}`
				
				Note
				有关可能的STAT_NAME值，请参见下表

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例

	数值统计	统计数据名称（参见下表）	与自定义值的数值比较	`"CheckExpression": "AGG(<STAT_NAME>) < :val", "SubstitutionMap": {":val", "100"}`
				
				或者
				
				`"CheckExpression": "AGG(<STAT_NAME>, :param) < :val", "SubstitutionMap": {":param": "0.25", :val", "5"}`
				
				Note
				有关可能的STAT_NAME值，请参见下表

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
非聚合（接受阈值）	值恰好值		与值列表进行精确比较	`"CheckExpression": ":col IN :list", "SubstitutionMap": {":col": "`size`", ":list": "[\"S\",\"M\",\"L\",\"XL\"]"}`
	值不完全是		值不应与列表中的任何值完全匹配	`"CheckExpression": ":col NOT IN :list", "SubstitutionMap": {":col": "`domain`", ":list": "[\"GOV\",\"ORG\"]"}`

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
	字符串值		与自定义值或其他字符串列进行字符串比较	"CheckExpression": ":col STARTS_WITH :val", "SubstitutionMap": {":col": "`url`", ":val": "http"} 或者 "CheckExpression": ":col1 contains :col2", "SubstitutionMap": {":col1": "`url`", ":col2": "`company_name`"}

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
	数字值		与自定义值或其他数值列进行数值比较	"CheckExpression": ":col IS_BETWEEN :val1 and :val2", "SubstitutionMap": {":col": "`APY`", ":val1": "0", ":val2": "10"} 或者 "CheckExpression": ":col1 <= :col2", "SubstitutionMap": {":col1": "`bank_rate`", ":col2": "`fed_rate`"}

条件类型	数据质量检查	其他参数	比较类型	SDK 语法示例
	值字符串长度		与自定义值或其他数值列进行数值比较	"CheckExpression": "length(:col) IS_BETWEEN :val1 and :val2", "SubstitutionMap": {":col": "`identifier`", ":val1": "8", ":val2": "12"} 或者 "CheckExpression": "length(:col1) <= :col2", "SubstitutionMap": {":col1": "`name`", ":col2": "`max_name_len`"}

数字比较

DataBrew 支持以下数值比较运算：等于 (= =)、不等于 (! =)、小于 (<)、小于等于 (< =) 、大于 (>)、大于等于 (> =) 和介于 (is_between: val1 和:val2) 之间。

字符串比较

支持以下字符串比较：开头为、不开头、结尾为、不结尾、包含、不包含、不包含、等于、不等于、不等于、匹配、不匹配。

下表显示了可用于值分布统计和数值统计的可用统计量：

数据质量检查	统计数据名称	其他参数	SDK 语法
价值分布统计	最小值		`"CheckExpression": "AGG(MAX) < :val", "SubstitutionMap": {":val", "100"}`
	最大值		`"CheckExpression": "AGG(MIN) > :val", "SubstitutionMap": {":val", "0"}`
	中间值		`"CheckExpression": "AGG(MEDIAN) >= :val", "SubstitutionMap": {":val", "50"}`
	平均值		`"CheckExpression": "AGG(MEAN`

数据质量检查	统计数据名称	其他参数	SDK 语法
			) <= :val", "Substitu tionMap": {":val", "10"}
	Mode		"CheckExp ression": "AGG(MODE) > :val", "Substitu tionMap": {":val", "0"}
	标准差		"CheckExp ression": "AGG(STAN DARD_DEVI ATION) > :val", "Substitu tionMap": {":val", "0"}
	熵		"CheckExp ression": "AGG(ENTR OPY) > :val", "Substitu tionMap": {":val", "0"}

数据质量检查	统计数据名称	其他参数	SDK 语法
数值统计	总和		"CheckExpression": "AGG(SUM) > :val", "SubstitutionMap": {":val", "0"}
	峰度		"CheckExpression": "AGG(KURTOSIS) > :val", "SubstitutionMap": {":val", "0"}
	偏斜		"CheckExpression": "AGG(SKEWNESS) > :val", "SubstitutionMap": {":val", "0"}
	方差		"CheckExpression": "AGG(VARIANCE) > :val", "SubstitutionMap": {":val", "0"}

数据质量检查	统计数据名称	其他参数	SDK 语法
	绝对偏差		"CheckExpression": "AGG(MEDIAN_ABSOLUTE_DEVIATION) > :val", "SubstitutionMap": {":val", "0"}
	分位数	分位数：'0.25'、'0.5'、'0.75' 中的一个	"CheckExpression": "AGG(QUANTILE, :pct) > :val", "SubstitutionMap": {":pct": "0.25", ":val", "0"}

创建和使用 AWS Glue DataBrew 项目

在中 AWS Glue DataBrew，项目是数据分析和转换工作的核心。

创建项目时，需要将两个基本组件组合在一起：

- 数据集，可提供对源数据的只读访问权限。有关更多信息，请参阅 使用连接数据 AWS Glue DataBrew。
- 一种方法，用于将 DataBrew 数据转换应用于数据集。有关更多信息，请参阅 创建和使用 AWS Glue DataBrew 食谱。

DataBrew 控制台以高度交互且直观的用户界面呈现您的项目。它鼓励你尝试数百种数据转换，这样你就可以了解它们是如何工作的，以及它们对你的数据有什么影响。

您在项目视图中看到的数据是您的数据集的示例。由于数据集可能非常大，有数千甚至数百万行，因此使用样本有助于确保 DataBrew 控制台在以各种方式转换样本数据时保持响应能力。默认情况下，样本包含数据集中的前 500 行数据。您可以为样本大小选择不同的设置，也可以选择哪些行。

在转换示例数据时， DataBrew 可以帮助您构建和完善项目配方，这是您迄今为止应用 step-by-step 的一系列转换。您的 work-in-progress食谱会自动保存，因此您可以随时离开项目视图，稍后返回，然后从上次停下来的地方继续前进。

当你的食谱准备好可供使用时，你可以发布它。发布配方使其可供 DataBrew 作业子系统使用，在那里你可以将配方应用于整个数据集，或者创建广泛的数据配置文件，让你了解数据的结构、内容和统计特征。

主题

- 创建项目
- DataBrew 项目会议概述
- 删除项目

创建项目

要创建项目，请按照以下过程操作。

创建项目

1. 登录到 AWS Management Console 并打开 DataBrew 控制台。

2. 在导航窗格上，选择项目。然后选择 “创建项目”。
3. 输入项目的名称。然后选择要附加到项目中的配方：

   - 如果您从头开始，请选择 “创建新食谱”。这样做会创建一个新的空配方并将其附加到您的项目中。
   - 如果您有之前发布的食谱要用于此项目，请选择 “编辑现有食谱”。如果该配方当前已附加到另一个项目，或者已为其定义了任何作业，则无法在新项目中使用它。选择浏览配方，查看可用的配方。
   - 如果您有以前发布过的现有食谱并想要导入其步骤，请选择从食谱中导入步骤，然后执行以下操作：

     1. 选择浏览配方，查看可用的配方。
     2. 选择要使用的配方的已发布版本。一个配方可以有多个版本，具体取决于你在项目视图中工作时发布它的频率。
     3. 选择 “查看配方步骤” 以检查配方中的数据转换。
4. 完成配方后，在选择数据集窗格中，选择要处理的数据集：

   - 我的数据集-选择您之前创建的数据集。有关更多信息，请参阅创建项目。
   - 示例文件-根据由维护的示例数据创建新的数据集 AWS。此示例数据是探索 DataBrew 可以做什么的好方法，而无需提供自己的数据。请确保输入数据集的名称。
   - 新数据集-创建新数据集。有关更多信息，请参阅 创建项目。
5. 要获得访问权限，请选择允许从您的 Amazon S3 输入位置 DataBrew 进行读取的 AWS Identity and Access Management (IAM) 角色。对于您的 AWS 账户拥有的 S3 地点，您可以选择`AwsGlueDataBrewDataAccessRole`服务托管角色。这样做可以 DataBrew 访问您拥有的 S3 资源。
6. 在采样窗格上，您可以找到 DataBrew 用于从您的数据集构建数据样本的选项。

   在 “类型” 中，选择 DataBrew 应如何从数据集中获取行：

   - 使用前 n 行根据数据集中的前几行创建样本。
   - 使用随机行根据数据集中随机选择的行创建样本。
   - 选择要在样本中显示的行数：500、1,000、2,500，或者自定义样本大小，最多 5,000 行。较小的样本量可以 DataBrew 更快地执行转换，从而节省开发配方的时间。样本量越大，就能更准确地反映基础源数据的构成。但是，项目会话初始化和交互式转换的速度较慢。
7. （可选）选择标签以将标签附加到您的数据集。

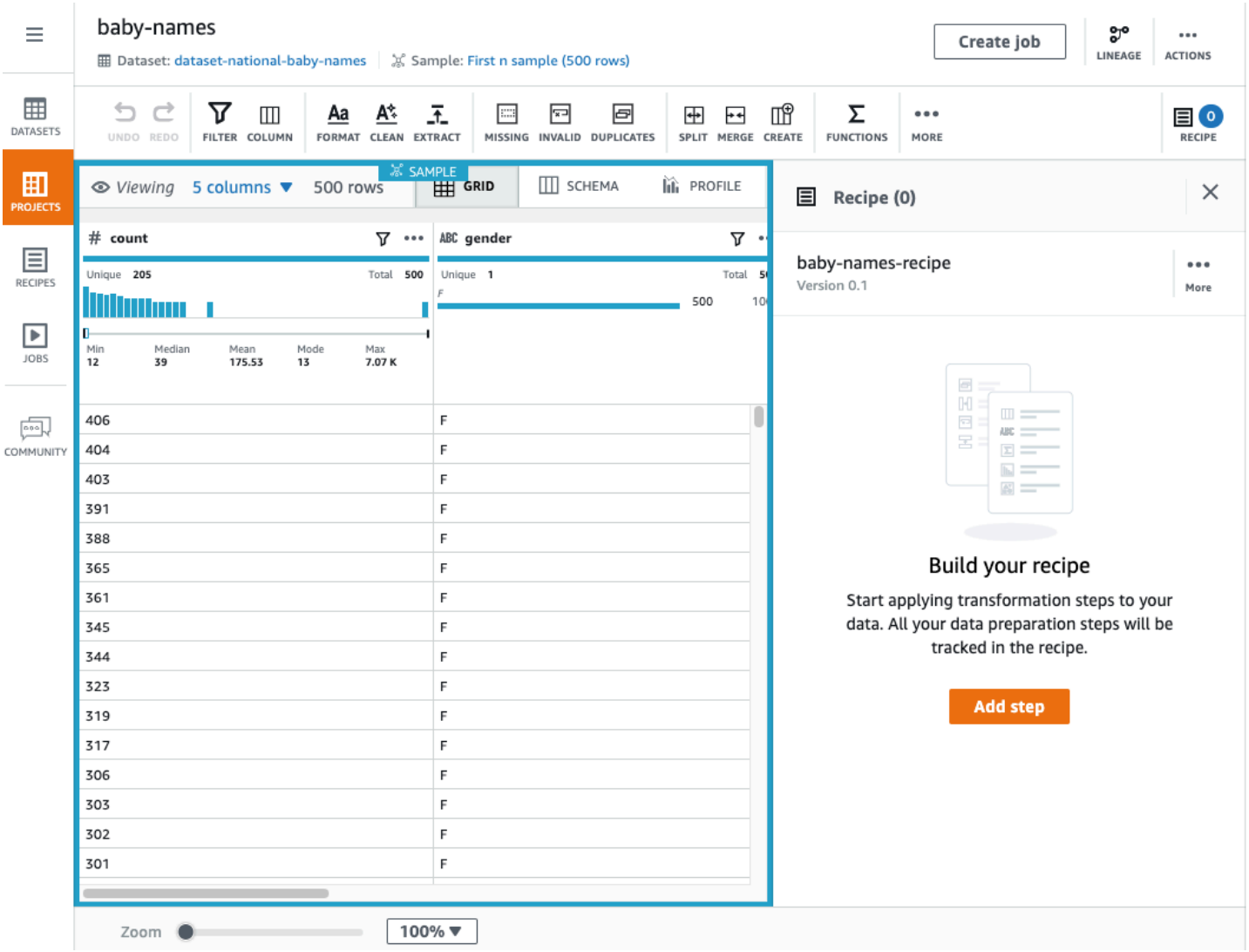
标签是简单标注，包含一个用户定义的键和一个可选值，方便管理、搜索和筛选 DataBrew 项目，方便按目的、所有者、环境或其他标准进行项目。

8. 根据需要进行设置后，选择 Create job（创建作业）。

DataBrew 根据需要创建新数据集，根据需要创建新配方，构建数据样本，并创建交互式项目会话。此过程可能需要几分钟完成。项目准备就绪后，您可以开始处理数据样本。

DataBrew 项目会议概述

在 DataBrew 项目会话中，您在交互式工作区中工作。



左侧窗格显示数据的当前视图。右侧窗格显示项目的转换配方，该配方当前为空。

在数据网格的右上角，有三个选项卡：GRIDSCHEMA、和。PROFILE选择其中一个选项卡将在工作区中显示相应的视图；接下来将介绍这些视图。

网格视图

网格视图是默认视图，其中样本以表格格式显示。使用以下步骤简要介绍网格视图。

浏览网格视图

1. 首先查看整个空间：
   a. 向左和向右滚动以查看所有列。
   b. 向上和向下滚动以查看所有数据值。
   c. 使用工作区底部的缩放控件来调整网格的放大率。
2. 在右上角，查看样本中显示了多少列以及样本中当前的行数。

   要更改显示的列，请选择 N 列链接（其中 N 是当前显示的列数）。选择所需的列，然后选择 “显示所选列”。
3. 现在，您可以开始尝试 DataBrew 变换。尝试以下操作：
   a. 在转换工具栏中，选择选择格式，更改为大写。
   b. 在 “源列” 中，选择包含字符数据的列。
   c. 保留其他设置的默认值。
   d. 要查看转换后的数据会是什么样子，请选择 “预览更改”。然后，要将此转换添加到您的食谱中，请选择 “应用”。

   无论何时应用数据转换，都要将其 DataBrew 添加到配方的工作副本中。它显示在工作区的右侧。
4. 尝试以下操作：
   a. 在转换工具栏中，选择创建，基于函数。
   b. 在 “选择函数” 中，选择SQUARE ROOT。
   c. 在源列中，选择包含数值数据的列。
   d. 将其他设置保留默认值。
   e. 选择 “预览更改” 以查看转换后的数据是什么样子。然后，要将此转换添加到您的食谱中，请选择 “应用”。
5. 选择 “食谱”，折叠右上角的食谱窗格。要展开食谱窗格，请再次选择 “食谱”。

发布配方的新版本

随着您继续应用变换，配方中的步骤数会增加。您可以随时发布新版本的配方。发布食谱可在其他地方使用 DataBrew。通过执行此操作，您可以运行配方作业来转换整个数据集，而不是仅转换项目数据样本。

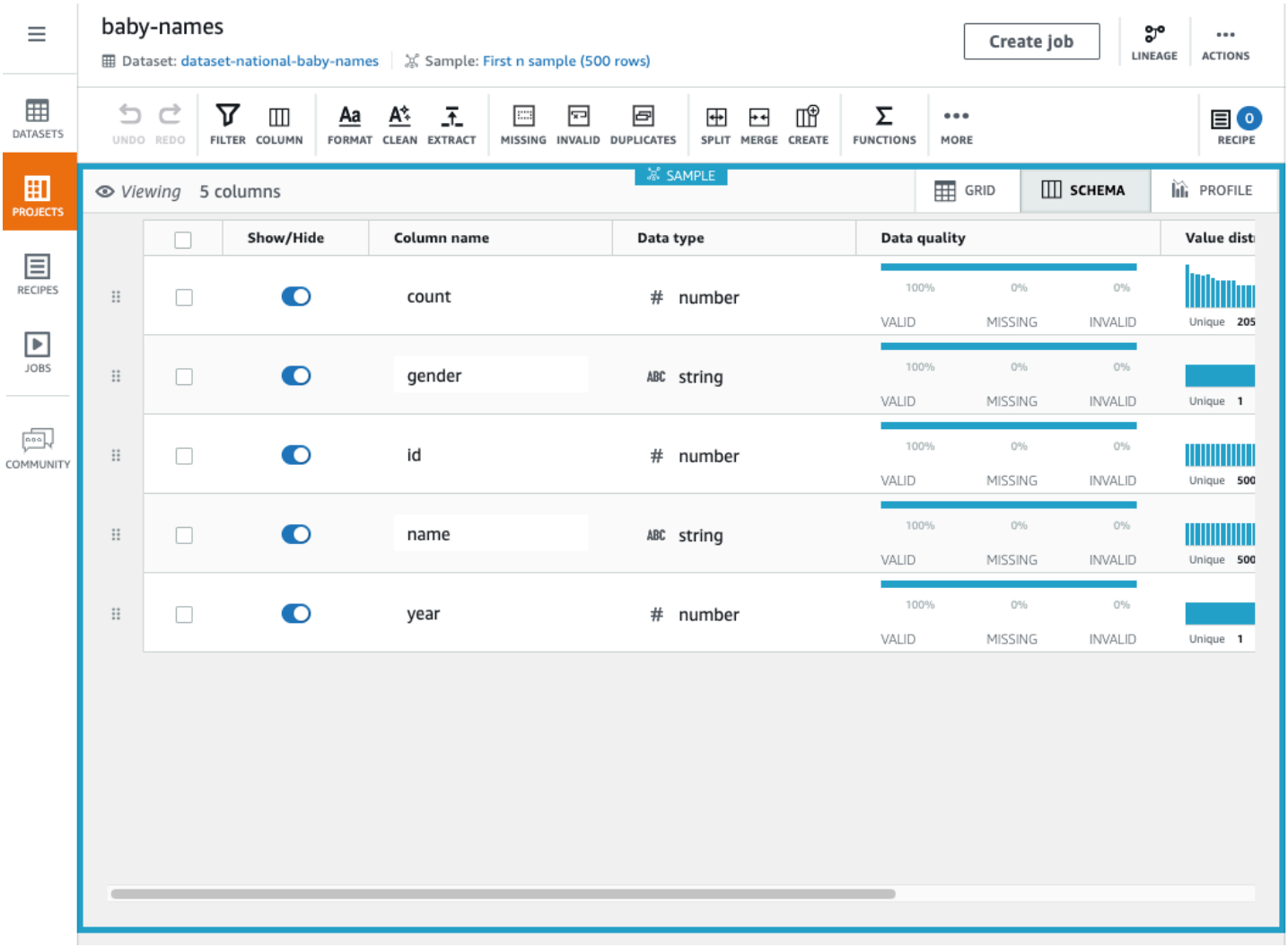
发布食谱还鼓励采用渐进、迭代的方法来开发食谱：你可以随时发布食谱的新版本，这样你就可以根据需要回退到 “最后一个已知的好食谱” 版本。

发布配方的新版本

- 在配方窗格中，选择 “发布”。输入此版本食谱的描述，然后选择 “发布”。

架构视图

如果选择 “架构” 选项卡，视图会发生变化，如以下屏幕截图所示。



在架构视图中，您可以查看有关每列中数据值的统计信息。

在最左侧列的 “显示/隐藏” 旁边，选择任意数据列。列详细信息窗格出现在右侧。此窗格显示列值的统计摘要。

您可以通过为 Column name (列名称) 输入新名称来重命名列。

您可以通过拖放列来重新排列列顺序。

个人资料视图

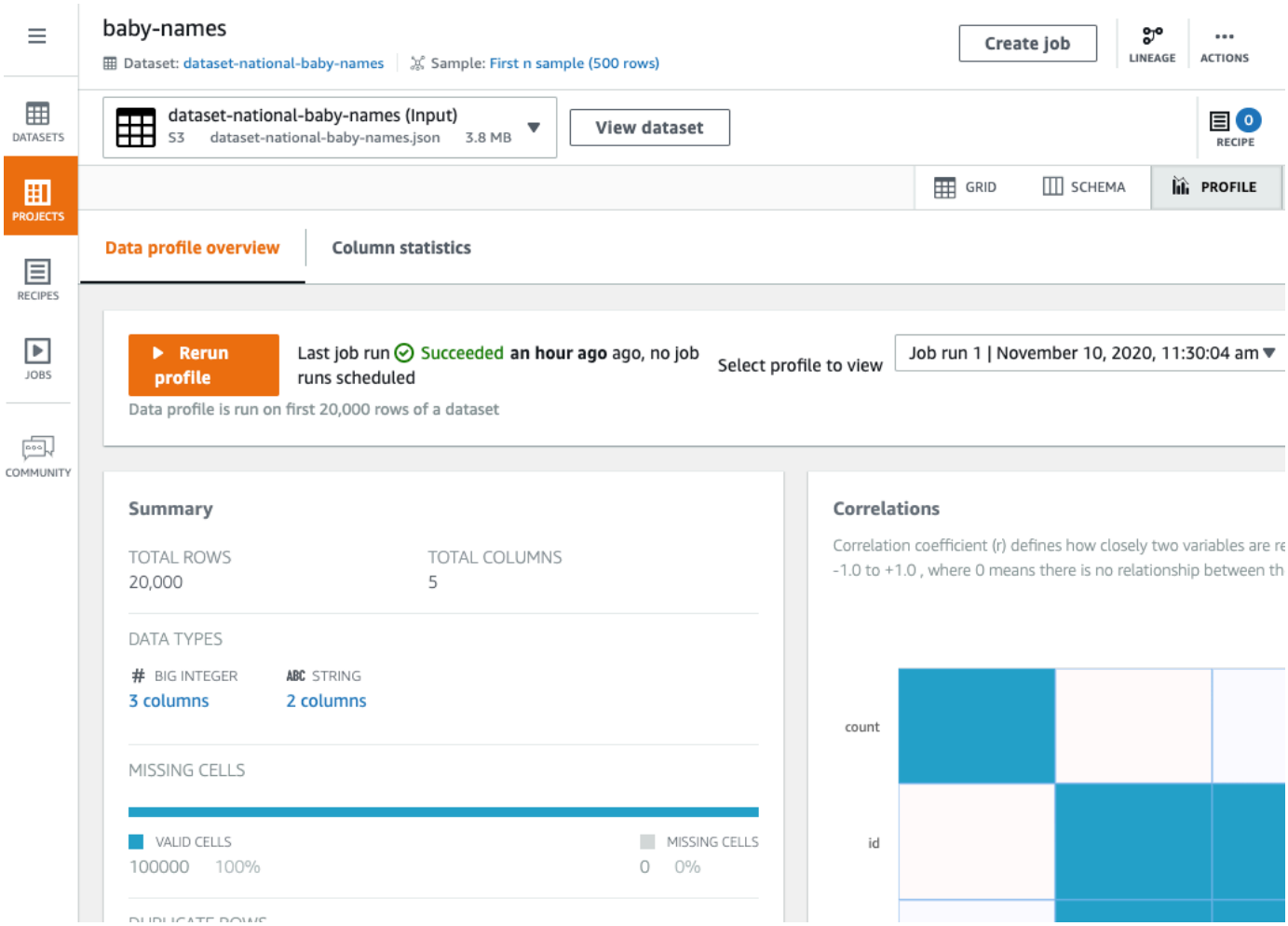
如果选择 “配置文件” 选项卡，则可以查看有关项目的详细体积信息。在执行此操作之前，您需要运行 DataBrew 任务来创建配置文件。

浏览个人资料视图

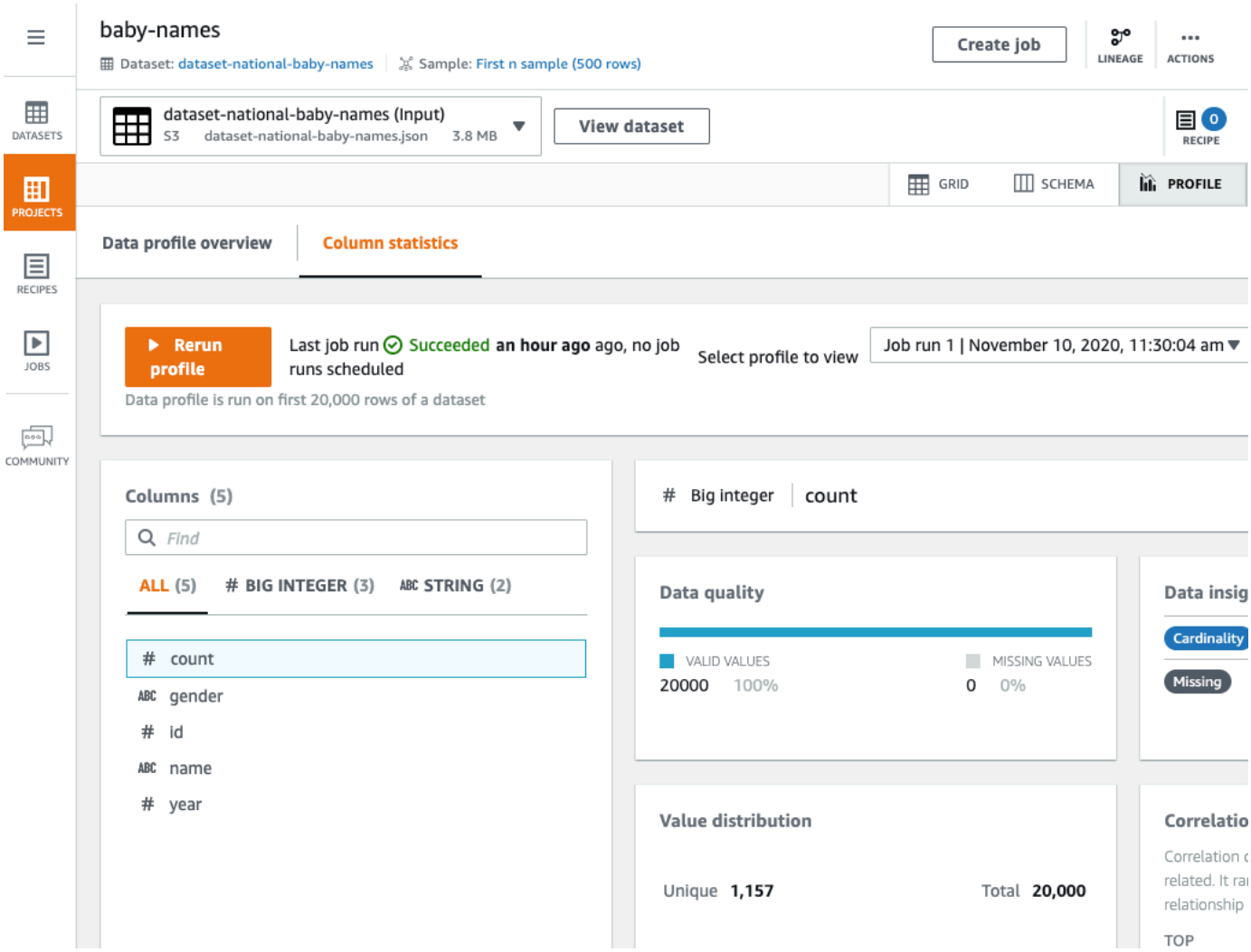
1. 选择创建作业，然后输入作业的名称。
2. 对于 Job 输出，为文件类型选择 CSV。
3. 在您的 AWS 账户中找到或创建您想要写入任务输出的 Amazon S3 存储桶和文件夹： DataBrew
   - 如果您已有此 Amazon S3 存储桶和文件夹，请选择 “浏览” 并找到它们。请确保您对两者都具有写入权限。
   - 如果您没有此 Amazon S3 存储桶和文件夹，请创建它们：
     1. 打开 Amazon S3 控制台，网址为：https://console.aws.amazon.com/s3/。
     2. 如果您没有 Amazon S3 存储桶，请选择创建存储桶。在存储桶名称中，输入新存储桶的唯一名称。选择创建存储桶。
     3. 从存储桶列表中，选择要使用的存储桶。
     4. 请选择 Create folder（创建文件夹）。在 “文件夹名称” 中`databrew-output`，输入并选择 “创建文件夹”。
4. DataBrew 要获得访问权限，请选择允许写入您的 Amazon S3 输出位置的 IAM 角色。

   对于您的 AWS 账户拥有的 S3 地点，您可以选择`AwsGlueDataBrewDataAccessRole`服务托管角色。这样做可以 DataBrew 访问您拥有的 S3 资源。
5. 将其他设置保留默认值，然后选择创建并运行作业。
6. 作业运行完成后，工作区将显示数据配置文件的图形摘要。

   数据配置文件概述选项卡显示了数据特征的高级摘要，如以下屏幕截图所示。



“列统计” 选项卡显示了数据值的 column-by-column 细分：



删除项目

如果您不再需要某个项目，可将其删除。

删除项目

1. 在导航窗格上，选择项目。
2. 选择要删除的项目，然后在 “操作” 中选择 “删除”。 。

创建和使用 AWS Glue DataBrew 食谱

在中 DataBrew，配方是一组数据转换步骤。您可以将这些步骤应用于数据样本，也可以将相同的方法应用于数据集。

开发配方的最简单方法是创建一个 DataBrew 项目，您可以在其中以交互方式处理数据样本——有关更多信息，请参阅。创建和使用 AWS Glue DataBrew 项目作为项目创建工作流程的一部分，将创建一个新的（空的）配方并将其附加到项目中。然后，您可以通过添加数据转换开始构建配方。

> **Note**
>
> 一个 DataBrew 配方最多可包含 100 个数据转换。

在继续开发食谱时，您可以通过发布食谱来保存您所做的工作。 DataBrew 为您的食谱维护已发布版本的列表。您可以在配方作业中使用任何已发布的版本来运行配方（在配方作业中）来转换您的数据集。您也可以下载配方步骤的副本，以便可以在其他项目或其他数据集转换中重复使用该配方。

您也可以使用 AWS Command Line Interface (AWS CLI) 或其中一个 AWS SDK 以编程方式开发 DataBrew 配方。在 DataBrew API 中，转换被称为配方操作。

> **Note**
>
> 在交互式 DataBrew 项目会话中，您应用的每次数据转换都会导致对 DataBrew API 的调用。这些 API 调用会自动发生，您无需知道 behind-the-scenes 细节。

即使你不是程序员，了解配方的结构以及如何 DataBrew 组织配方操作也会很有帮助。

主题

发布配方的新版本

您可以在交互式 DataBrew 项目会话中发布配方的新版本。

发布的新配方版本

1. 在配方窗格中，选择 “发布”。
2. 输入此版本食谱的描述，然后选择 “发布”。

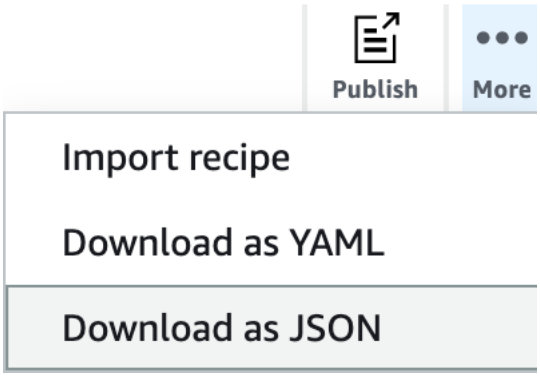
您可以通过从导航窗格中选择 PROJECTS 来查看所有已发布的食谱及其版本。

定义配方结构

首次使用 DataBrew 控制台创建项目时，需要定义与该项目关联的配方。如果您没有现有配方，控制台将为您创建配方。

在控制台中处理项目时，您可以使用转换工具栏对数据集中的示例数据应用操作。在您继续构建配方时，控制台会显示配方步骤以及这些步骤的顺序。您可以对配方进行迭代和完善，直到您对步骤感到满意为止。

在中入门 AWS Glue DataBrew，你构建了一个用于转换著名国际象棋游戏数据集的配方。您可以通过选择 “下载为 JSON” 或 “下载为 YAML” 来下载配方步骤的副本，如以下屏幕截图所示。



下载的 JSON 文件包含与您添加到配方中的转换相对应的配方操作。

新食谱没有任何步骤。您可以将新配方表示为空的 JSON 列表，如下所示。

```
[ ]
```

下面是此文件的一个示例，适用于chess-project-recipe。JSON 列表包含几个描述配方步骤的对象。JSON 列表中的每个对象都用大括号 () { } 括起来。JSON 行由逗号分隔。

```
[
    {
        "Action": {
```

```
            "Operation": "REMOVE_VALUES",
            "Parameters": {
                "sourceColumn": "black_rating"
            }
        },
        "ConditionExpressions": [
            {
                "Condition": "LESS_THAN",
                "Value": "1800",
                "TargetColumn": "black_rating"
            }
        ]
    },
    {
        "Action": {
            "Operation": "REMOVE_VALUES",
            "Parameters": {
                "sourceColumn": "white_rating"
            }
        },
        "ConditionExpressions": [
            {
                "Condition": "LESS_THAN",
                "Value": "1800",
                "TargetColumn": "white_rating"
            }
        ]
    },
    {
        "Action": {
            "Operation": "GROUP_BY",
            "Parameters": {
                "groupByAggFunctionOptions": "[{\"sourceColumnName\":\"winner\",
\"targetColumnName\":\"winner_count\",\"targetColumnDataType\":\"int\",\"functionName
\":\"COUNT\"}]",
                "sourceColumns": "[\"winner\",\"victory_status\"]",
                "useNewDataFrame": "true"
            }
        }
    },
    {
        "Action": {
            "Operation": "REMOVE_VALUES",
            "Parameters": {
```

```
                "sourceColumn": "winner"
            }
        },
        "ConditionExpressions": [
            {
                "Condition": "IS",
                "Value": "[\"draw\"]",
                "TargetColumn": "winner"
            }
        ]
    },
    {
        "Action": {
            "Operation": "REPLACE_TEXT",
            "Parameters": {
                "pattern": "mate",
                "sourceColumn": "victory_status",
                "value": "checkmate"
            }
        }
    },
    {
        "Action": {
            "Operation": "REPLACE_TEXT",
            "Parameters": {
                "pattern": "resign",
                "sourceColumn": "victory_status",
                "value": "other player resigned"
            }
        }
    },
    {
        "Action": {
            "Operation": "REPLACE_TEXT",
            "Parameters": {
                "pattern": "outoftime",
                "sourceColumn": "victory_status",
                "value": "ran out of time"
            }
        }
    }
]
```

如果我们只为新操作添加新行，则更容易看到每个操作都是单独的行，如下所示。

```
[
 { "Action": { "Operation": "REMOVE_VALUES", "Parameters": { "sourceColumn":
 "black_rating" } }, "ConditionExpressions": [ { "Condition": "LESS_THAN", "Value":
 "1800", "TargetColumn": "black_rating" } ] },
 { "Action": { "Operation": "REMOVE_VALUES", "Parameters": { "sourceColumn":
 "white_rating" } }, "ConditionExpressions": [ { "Condition": "LESS_THAN", "Value":
 "1800", "TargetColumn": "white_rating" } ] },
 { "Action": { "Operation": "GROUP_BY", "Parameters": { "groupByAggFunctionOptions":
 "[{\"sourceColumnName\":\"winner\",\"targetColumnName\":\"winner_count\",
\"targetColumnDataType\":\"int\",\"functionName\":\"COUNT\"}]", "sourceColumns":
 "[\"winner\",\"victory_status\"]", "useNewDataFrame": "true" } } },
 { "Action": { "Operation": "REMOVE_VALUES", "Parameters": { "sourceColumn":
 "winner" } }, "ConditionExpressions": [ { "Condition": "IS", "Value": "[\"draw\"]",
 "TargetColumn": "winner" } ] },
 { "Action": { "Operation": "REPLACE_TEXT", "Parameters": { "pattern": "mate",
 "sourceColumn": "victory_status", "value": "checkmate" } } },
 { "Action": { "Operation": "REPLACE_TEXT", "Parameters": { "pattern": "resign",
 "sourceColumn": "victory_status", "value": "other player resigned" } } },
 { "Action": { "Operation": "REPLACE_TEXT", "Parameters": { "pattern": "outoftime",
 "sourceColumn": "victory_status", "value": "ran out of time" } } }
]
```

这些操作按顺序执行，顺序与文件中的顺序相同：

- REMOVE_VALUES— 要筛选出所有玩家评分低于 1,800 的游戏，这是成为 A 级国际象棋选手所需的最低评分。这种动作有两次出现，一次是移除黑方不是 A 级玩家的玩家，另一次是移除白方没有达到这个等级的玩家。
- GROUP_BY— 汇总数据。在这种情况下，GROUP_BY 根据winner（black和white）的值将行分成几组。然后进一步细分每个组，根据 victory_status (、materesignoutoftime、和draw) 的值将行分成子组。最后，计算每个子组的出现次数。然后，生成的摘要将替换原始数据样本。
- REMOVE_VALUES— 删除结尾为的游戏的结果draw。
- REPLACE_TEXT— 修改的值victory_status。此操作会出现三次，分别用于mate、和。resign oufoftime

在交互式 DataBrew 项目会话中，每个会话RecipeAction对应于您应用于数据样本的数据转换。

DataBrew 提供 200 多个配方操作。有关更多信息，请参阅 配方步骤和功能参考。

使用条件

您可以使用条件来缩小配方操作的范围。条件用于筛选数据的转换，例如，根据特定的列值删除不需要的行。

让我们更仔细地看看中的配方操作chess-project-recipe。

```
{
  "Action": {
    "Operation": "REMOVE_VALUES",
    "Parameters": {
      "sourceColumn": "black_rating"
    }
  },
  "ConditionExpressions": [
    {
      "Condition": "LESS_THAN",
      "Value": "1800",
      "TargetColumn": "black_rating"
    }
  ]
}
```

此转换会读取black_rating列中的值。该ConditionExpressions列表决定了筛选标准：任何black_rating值小于 1,800 的行都将从数据集中移除。

对于，配方中的后续改造也会起到同样的作用white_rating。这样，数据仅限于每位玩家（黑人或白人）被评为A级或以上的游戏。

以下是应用于一列字符数据的条件的另一个示例。

```
{
  "Action": {
    "Operation": "REMOVE_VALUES",
    "Parameters": {
      "sourceColumn": "winner"
    }
  },
  "ConditionExpressions": [
    {
      "Condition": "IS",
      "Value": "[\"draw\"]",
```

```
        "TargetColumn": "winner"
      }
    ]
  }
```

此转换读取winner列中的值，查找值draw并删除这些行。这样，数据仅限于那些有明显赢家的游戏。

DataBrew 支持以下条件：

- IS-此列中的值与条件中提供的值相同。
- IS_NOT-此列中的值与条件中提供的值不同。
- IS_BETWEEN— 列中的值介于GREATER_THAN_EQUAL和LESS_THAN_EQUAL参数之间。
- CONTAINS-此列中的字符串值包含条件中提供的值。
- NOT_CONTAINS-此列中的值不包含条件中提供的字符串。
- STARTS_WITH-此列中的值以条件中提供的字符串开始。
- NOT_STARTS_WITH-此列中的值不是条件中提供的字符串。
- ENDS_WITH-此列中的值以条件中提供的字符串结尾。
- NOT_ENDS_WITH-此列中的值不是条件中提供的字符串。
- LESS_THAN-此列中的值小于条件中提供的值。
- LESS_THAN_EQUAL-此列中的值小于或等于条件中提供的值。
- GREATER_THAN-此列中的值大于条件中提供的值。
- GREATER_THAN_EQUAL-此列中的值大于或等于条件中提供的值。
- IS_INVALID— 该列中的值的数据类型不正确。
- IS_MISSING— 该列中没有值。

创建、运行和调度 AWS Glue DataBrew 作业

AWS Glue DataBrew 有一个作业子系统，它有两个用途：

1. 将数据转换方法应用于 DataBrew 数据集。你用一份 DataBrew 食谱工作来做这件事。
2. 分析数据集以创建数据的全面概况。你用 DataBrew 个人资料工作来做这件事。

主题

创建和处理 AWS Glue DataBrew 配方作业

使用 DataBrew 配方作业清理和标准化数据 DataBrew集中的数据，并将结果写入您选择的输出位置。运行配方作业不会影响数据集或底层源数据。当作业运行时，它会以只读方式连接到源数据。任务输出将写入您在 Amazon S3 AWS Glue Data Catalog、或支持的 JDBC 数据库中定义的输出位置。

使用以下步骤创建 DataBrew 配方作业。

创建配方任务

1. 登录 AWS Management Console 并打开 DataBrew 控制台，网址为 https://console.aws.amazon.com/databrew/。
2. 从导航窗格中选择 “作业”，选择 “食谱作业” 选项卡，然后选择 “创建作业”。
3. 输入任务的名称，然后选择 “创建配方作业”。
4. 在 J ob 输入中，输入要创建的作业的详细信息：要处理的数据集的名称和要使用的配方。

   配方作业使用 DataBrew 配方来转换数据集。要使用食谱，请务必先将其发布。
5. 配置您的任务输出设置。

   为您的任务输出提供目的地。如果您没有为输出目标配置 DataBrew 连接，请先在数据集选项卡上进行配置，如中所述支持的数据源和输出连接。选择以下输出目标之一：

   - Amazon S3，有或没有 AWS Glue Data Catalog 支持
   - 亚马逊 Redshift，有或没有支持 AWS Glue Data Catalog

- JDBC
- 雪花桌
- AWS Glue Data Catalog 支持的 Amazon RDS 数据库表。Amazon RDS 数据库表支持以下数据库引擎：
  - Amazon Aurora
  - MySQL
  - Oracle
  - PostgreSQL
  - Microsoft SQL Server
- 有 AWS Glue Data Catalog 支持的 Amazon S3。

对于基于的 AWS Glue Data Catalog 输出 AWS Lake Formation，仅 DataBrew 支持替换现有文件。在这种方法中，文件将被替换，以保持数据访问角色的现有 Lake Formation 权限不变。此外， DataBrew 优先考虑 AWS Glue Data Catalog 表中的 Amazon S3 位置。因此，在创建配方任务时，您无法覆盖 Amazon S3 的位置。

在某些情况下，任务输出中的 Amazon S3 位置与数据目录表中的 Amazon S3 位置不同。在这些情况下，使用目录表中的 Amazon S3 位置自动 DataBrew 更新任务定义。它会在您更新或启动现有任务时执行此操作。

6. 仅对于 Amazon S3 的输出目的地，您还有其他选择：

   a. 为 Amazon S3 选择一种可用的数据输出格式、可选的压缩格式和可选的自定义分隔符。输出文件支持的分隔符与输入文件支持的分隔符相同：逗号、冒号、分号、竖线、制表符、插入符号、反斜杠和空格。有关格式的详细信息，请参阅下表。

格式	文件扩展名（未压缩）	文件扩展名（压缩）
以逗号分隔的值	`.csv`	`.csv.snappy` ，`.csv.gz`, `.csv.lz4`, `csv.bz2`, `.csv.deflate` ，`csv.br`
制表符分隔的值	`.csv`	`.tsv.snappy` ，`.tsv.gz`, `.tsv.lz4`, `tsv.bz2`, `.tsv.deflate` ，`tsv.br`

格式	文件扩展名（未压缩）	文件扩展名（压缩）
Apache Parquet	`.parquet`	`.parquet.snappy` ，`.parquet.gz` ，`.parquet.lz4` ，`.parquet.lzo` ，`.parquet.br`
AWS Glue 实木地板	不支持	`.glue.parquet.snappy`
Apache Avro	`.avro`	`.avro.snappy` ，`.avro.gz`, `.avro.lz4` ，`.avro.bz2` ，`.avro.deflate` ，`.avro.br`
Apache ORC	`.orc`	`.orc.snappy` ，`.orc.lzo`, `.orc.zlib`
XML	`.xml`	`.xml.snappy` ，`.xml.gz`, `.xml.lz4`, `.xml.bz2`, `.xml.deflate` ，`.xml.br`
JSON（仅限 JSON 行格式）	`.json`	`.json.snappy` ，`.json.gz`, `.json.lz4` ，`json.bz2`, `.json.deflate` ，`.json.br`
Tableau 超级	不支持	不适用

b. 选择是输出单个文件还是多个文件。Amazon S3 有三个文件输出选项：

- 自动生成文件（推荐）-已 DataBrew 确定输出文件的最佳数量。
- 单文件输出-生成单个输出文件。此选项可能会导致额外的作业执行时间，因为需要进行后期处理。
- 多文件输出-是否为任务输出指定文件数。有效值为 2—999。如果使用列分区或输出中的行数少于您指定的文件数，则输出的文件数量可能会少于您指定的数量。

c. （可选）为配方任务输出选择列分区。

列分区提供了另一种将配方任务输出分成多个文件的方法。列分区可以与新的或现有的 Amazon S3 输出一起使用，也可以与新的数据目录 Amazon S3 输出一起使用。它不能用于现有的数据目录 Amazon S3 表。输出文件基于您指定的列名的值。如果您指定的列名是唯一的，则生成的 Amazon S3 文件夹路径将基于列名的顺序。

有关列分区的示例列分区示例，请参见以下内容。

7. （可选）选择 “为作业输出启用加密” 以加密 DataBrew 写入输出位置的作业输出，然后选择加密方法：

- 使用 SSE-S3 加密 — 输出使用服务器端加密和 Amazon S3 托管加密密钥进行加密。
- 使用 AWS Key Management Service (AWS KMS)-使用对输出进行加密 AWS KMS。要使用此选项，请选择要使用的 AWS KMS 密钥的 Amazon 资源名称 (ARN)。如果您没有 AWS KMS 密钥，则可以通过选择创建密钥来创建 AWS KMS 密钥。

8. 要获得访问权限，请选择允许 DataBrew 写入您的输出位置的 AWS Identity and Access Management (IAM) 角色。对于您的 AWS 账号拥有的营业地点，您可以选择`AwsGlueDataBrewDataAccessRole`服务管理角色。这样做可以 DataBrew 访问您拥有的 AWS 资源。

9. 在高级作业设置窗格上，您可以为作业的运行方式选择更多选项：

- 最大单位数-使用并行运行的多个计算节点 DataBrew 处理作业。默认节点数为 5。最大节点数为 149。
- Job timeout — 如果作业运行时间超过您在此处设置的分钟数，则该作业会失败并显示超时错误。默认值为 2,880 分钟或 48 小时。
- 重试次数-如果作业在运行时失败， DataBrew 可以尝试再次运行。默认情况下，不会重试该作业。
- 为作业启用 Amazon CloudWatch 日志- DataBrew 允许将诊断信息发布到 CloudWatch 日志。这些日志可用于故障排除或获取有关如何处理任务的更多详细信息。

10. 对于 Schedule 作业，您可以应用 DataBrew 作业计划，以便您的作业在特定时间运行，或者定期运行。有关更多信息，请参阅 按计划自动运行作业。

11. 当设置符合您的需要时，选择创建作业。或者，如果您想立即运行作业，请选择创建并运行作业。

您可以通过在作业运行时检查其状态来监控作业的进度。作业运行完成后，状态将更改为 “成功”。现在，作业输出可在您选择的输出位置获得。

DataBrew 保存您的作业定义，以便您以后可以运行相同的作业。要重新运行作业，请从导航窗格中选择 “作业”。选择要处理的作业，然后选择 “运行作业”。

列分区示例

作为列分区的示例，假设您指定了三列，其中每行包含两个可能值中的一个。该Dept列的值可以是Admin或Eng。该Staff-type列的值可以是Part-time或Full-time。该Location列的值可以是Office1或Office2。用于任务输出的 Amazon S3 存储桶如下所示。

```
s3://bucket/output-folder/Dept=Admin/Staff-type=Part-time/Area=Office1/
jobId_timestamp_part0001.csv
s3://bucket/output-folder/Dept=Admin/Staff-type=Part-time/Location=Office2/
jobId_timestamp_part0002.csv
s3://bucket/output-folder/Dept=Admin/Staff-type=Full-time/Location=Office1/
jobId_timestamp_part0003.csv
s3://bucket/output-folder/Dept=Admin/Staff-type=Full-time/Location=Office2/
jobId_timestamp_part0004.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Part-time/Location=Office1/
jobId_timestamp_part0005.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Part-time/Location=Office2/
jobId_timestamp_part0006.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Full-time/Location=Office1/
jobId_timestamp_part0007.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Full-time/Location=Office2/
jobId_timestamp_part0008.csv
```

按计划自动运行作业

您可以随时重新运行 DataBrew 作业，也可以按计划自动运行 DataBrew 作业。

重新运行作业 DataBrew

1. 登录 AWS Management Console 并打开 DataBrew 控制台，网址为 https://console.aws.amazon.com/databrew/。
2. 在导航窗格上，选择作业。选择要运行的作业，然后选择运行作业。

要在特定时间运行 DataBrew 作业，或定期运行作业，请创建 DataBrew 作业计划。然后，您可以将作业设置为按计划运行。

创建 DataBrew 作业时间表

1. 在 DataBrew 控制台的导航窗格上，选择作业。选择 “计划” 选项卡，然后选择 “添加计划”。
2. 输入计划名称，然后为运行频率选择一个值：
   - 重复-选择您希望作业运行的频率（例如，每 12 小时运行一次）。然后选择在哪一天或几天运行作业。或者，您可以输入一天中作业运行的时间。
   - 在特定时间-输入您希望作业运行的时间。然后选择在哪一天或几天运行作业。
   - 输入 CRON-通过输入有效的 cron 表达式来定义作业计划。有关更多信息，请参阅 使用配方作业的 cron 表达式。
3. 根据需要进行设置后，选择 Save (保存)。

将作业与计划关联

1. 在导航窗格上，选择作业。
2. 选择要处理的作业，然后在 “操作” 中选择 “编辑”。 。
3. 在 “安排作业” 窗格上，选择 “关联计划”。选择要使用的时间表的名称。
4. 根据需要进行设置后，选择 Save (保存)。

使用配方作业的 cron 表达式

Cron 表达式有六个必填字段，之间以空格分隔。语法如下所示。

```
Minutes Hours Day-of-month Month Day-of-week Year
```

在前面的语法中，以下值和通配符用于指定的字段。

字段	值	通配符
分钟	0-59	, - * /
小时	0–23	, - * /
D ay-of-month	1-31	, - * ? / L W
月	1-12 或 JAN-DEC	, - * /

字段	值	通配符
D ay-of-week	1-7 或 SUN-SAT	, - * ? / L
年	1970-2199	, - * /

按如下方式使用这些通配符：

- ,（逗号）通配符包含其他值。在现Month场，JAN,FEB,MAR包括一月、二月和三月。
- -（破折号）通配符指定范围。在该Day字段中，1—15 包括指定月份的第 1 天到第 15 天。
- *（星号）通配符包含该字段中的所有值。在该Hours字段中，* 包括每小时。
- /（斜杠）通配符用于指定增量。在该Minutes字段中，您可以输入**1/10**指定从一小时的第一分钟（例如，第 11 分钟、第 21 分钟和第 31 分钟）开始每隔 10 分钟。
- ?（问号）通配符用于指定一个或另一个。例如，假设您在Day-of-month字段中输入 7。如果你不在乎第七天是哪一天，那么你可以输入？ 在Day-of-week野外。
- Day-of-month或Day-of-week字段中的 L 通配符指定一个月或一周的最后一天。
- Day-of-month 字段中的 W 通配符用于指定工作日。在 Day-of-month 字段中，3W 用于指定最靠近当月的第三周的日。

这些字段和值具有以下限制：

- 您无法在同一 cron 表达式中为 Day-of-month 和 Day-of-week 字段同时指定值。如果您在其中一个字段中指定了值，则必须在另一个字段中使用 ?（问号）。
- 不支持导致速率快于 5 分钟的 Cron 表达式。

在创建计划时，您可以使用以下示例 cron 字符串。

分钟	小时	日期	月份	星期几	年	含义
0	10	*	*	?	*	每天上午 10:00（世界标准时间）跑步

分钟	小时	日期	月份	星期几	年	含义
15	12	*	*	?	*	每天下午 12:15 (UTC) 运行
0	18	?	*	MON-FRI	*	每星期一到星期五下午 6:00 (UTC) 运行
0	8	1	*	?	*	每月第一天上午 8:00（UTC）运行
0/15	*	*	*	?	*	每 15 分钟运行一次
0/10	*	?	*	MON-FRI	*	从星期一到星期五，每 10 分钟运行一次
0/5	8-17	?	*	MON-FRI	*	在每星期一到星期五的上午 8:00 到下午 5:55 (UTC) 之间，每 5 分钟运行一次

例如，您可以使用以下 cron 表达式在每天 12:15 UTC 运行作业。

```
15 12 * * ? *
```

删除作业和作业计划

如果您不再需要作业或作业计划，则可以将其删除。

删除任务

1. 在导航窗格上，选择作业。
2. 选择要删除的作业，然后在 “操作” 中选择 “删除”。 。

删除作业计划

1. 在导航窗格上，选择作业，然后选择计划选项卡。
2. 选择要删除的计划，然后在 “操作” 中选择 “删除”。 。

创建和使用 AWS Glue DataBrew 个人资料职位

分析任务对数据集进行一系列评估，并将结果输出到 Amazon S3。数据分析收集的信息可帮助您了解您的数据集，并决定在配方作业中可能要运行哪种数据准备步骤。

运行配置文件作业的最简单方法是使用默认 DataBrew 设置。您可以在运行配置文件作业之前对其进行配置，使其仅返回您想要的信息。

使用以下步骤创建 DataBrew 分析作业。

创建个人资料职位

1. 登录 AWS Management Console 并打开 DataBrew 控制台，网址为 https://console.aws.amazon.com/databrew/。
2. 从导航窗格中选择 J OBS，选择 “分析作业” 选项卡，然后选择 “创建作业”。
3. 输入职位名称，然后选择 “创建个人资料职位”。
4. 对于 Job 输入，请提供要分析的数据集的名称。
5. （可选）在数据配置文件配置窗格上配置以下内容：

   - 数据集级别配置-为数据集中的所有列配置个人资料作业的详细信息。

     或者，您可以开启检测和计算数据集中重复行的功能。您也可以选择 “启用相关性矩阵”，然后选择列，以查看多列中值的相关程度。有关可在数据集级别配置的统计数据的详细信息，请

参阅数据集级别的可配置统计数据。您可以在 DataBrew 控制台上配置统计信息，也可以使用 DataBrew API 或 AWS 软件开发工具包配置统计信息。

- 列级别配置-使用默认配置文件配置设置，您可以选择要包含在配置文件作业中的列。使用添加配置覆盖来选择要限制收集的统计信息数量的列，或者覆盖某些统计信息的默认配置。有关可以在列级别配置的统计信息的详细信息，请参阅列级别的可配置统计数据。您可以在 DataBrew 控制台上配置统计信息，也可以使用 DataBrew API 或 AWS 软件开发工具包配置统计信息。

  请确保您指定的任何配置覆盖都适用于您在配置文件作业中包含的列。如果您为某列配置的不同覆盖之间存在冲突，则最后一个冲突的覆盖具有优先级。

6. （可选）您可以创建数据质量规则并应用与此数据集关联的其他规则集，也可以删除已应用的规则集。有关数据质量验证的更多信息，请参阅验证中的数据质量 AWS Glue DataBrew。
7. 在高级作业设置窗格上，您可以为作业的运行方式选择更多选项：

   - 最大单位数-使用并行运行的多个计算节点 DataBrew 处理作业。默认节点数为 5。最大节点数为 149。
   - Job timeout — 如果作业运行时间超过您在此处设置的分钟数，则该作业会失败并显示超时错误。默认值为 2,880 分钟或 48 小时。
   - 重试次数-如果作业在运行时失败， DataBrew 可以尝试再次运行。默认情况下，不会重试该作业。
   - 为作业启用 Amazon CloudWatch 日志- DataBrew 允许将诊断信息发布到 CloudWatch 日志。这些日志可用于故障排除或获取有关如何处理任务的更多详细信息。
8. 对于 A ssociated S DataBrew chedule，您可以应用作业计划，以便您的作业在特定时间运行，或者定期运行。有关更多信息，请参阅 按计划自动运行作业。
9. 当设置符合您的需要时，选择创建作业。或者，如果您想立即运行作业，请选择创建并运行作业。

在中以编程方式构建配置文件作业配置 AWS Glue DataBrew

在本节中，您可以找到配置文件作业步骤和可以以编程方式使用的功能的描述。您可以从 AWS Command Line Interface (AWS CLI) 中使用它们，也可以使用其中一个 AWS SDK 来使用它们。

在分析作业中，您可以自定义配置以控制数据集的 DataBrew 评估方式。您可以将配置应用于数据集或将其应用于特定列。您可以在创建配置文件作业时构建配置，然后随时对其进行更新。

配置文件配置结构包括四个部分：

- ProfileColumns 部分

- DatasetStatisticsConfiguration 部分
- ColumnStatisticsConfigurations 部分
- EntityDetectorConfiguration 用于配置 PII 的部分

以下为示例。

```
{
    "ProfileColumns": [
        {
            "Name": "example"
        },
        {
            "Regex": "example.*"
        }
    ],
    "DatasetStatisticsConfiguration": {
        "IncludedStatistics": [
            "CORRELATION"
        ],
        "Overrides": [
            {
                "Statistic": "CORRELATION",
                "Parameters": {
                    "columnSelectors": "[{\"name\":\"example\"}, {\"regex\":\"example.*
\"}]"
                }
            }
        ]
    },
    "ColumnStatisticsConfigurations": [
        {
            "Selectors": [
                {
                    "Name": "example"
                }
            ],
            "Statistics": {
                "IncludedStatistics": [
                    "CORRELATION",
                    "DUPLICATE_ROWS_COUNT"
                ],
                "Overrides": [
```

```
                    {
                        "Statistic": "VALUE_DISTRIBUTION",
                        "Parameters": {
                            "binNumber": "10"
                        }
                    }
                ]
            }
        }
    ]
}
```

ProfileColumns 部分

在结构ProfileColumns部分中，设置要在个人资料作业中评估的数据集列。 ProfileColumns是列选择器列表 (Selectors)。可以在列选择器中指定列名或正则表达式。下面是一个示例。

```
"ProfileColumns": [{"Name": "example"}, {"Regex": "example.*"}]
```

如果指定，ProfileColumns则只有名称与中的名称或正则表达式匹配的列才ProfileColumns会包含在分析作业中。如果分析作业不支持选定列的数据类型，则在作业运行期间 DataBrew 跳过选定列。

如果 ProfileColumns 未定义，则分析作业将评估所有支持的列。支持的列是包含支持的数据类型的列：ByteType、、ShortType、IntegerType、LongType、FloatTypeDoubleType、String、或Boolean。

DatasetStatisticsConfiguration 部分

在结构的DatasetStatisticsConfiguration部分中，您可以为列间评估构建配置。该配置包括IncludedStatistics和Overrides。下面是一个示例。

```
"DatasetStatisticsConfiguration": {
    "IncludedStatistics": ["CORRELATION"],
    "Overrides": [
        {
            "Statistic": "CORRELATION",
            "Parameters": {
                "columnSelectors": "[{\"name\":\"example\"}, {\"regex\":\"example.*
\"}]"
            }
```

```
        }
    ]
}
```

您可以通过向中添加评估名称来选择想要的评估IncludedStatistics。下面是一个示例。

```
"IncludedStatistics": ["CORRELATION", "DUPLICATE_ROWS_COUNT"]
```

如果指定IncludedStatistics，则只有列表中的评估才会包含在分析作业中。如果IncludedStatistics未定义，则分析作业使用默认设置运行所有支持的评估。您可以通过向中添加 NONE 来排除所有评估IncludedStatistics。下面是一个示例。

```
"IncludedStatistics": ["NONE"]
```

数据集级别的可配置统计数据

在您的结构DatasetStatisticsConfiguration部分中，配置文件作业支持下表所示的评估。

统计名称	描述	支持的数据类型	默认状态	个人资料结果的属性	个人资料结果的类型
重复行数	数据集中重复行的计数	all	Enable	duplicate RowsCoun	Int
相关性	两列之间的 Pearson 相关系数	number	Enable	相关性（在每个选定列中）	对象

在中IncludedStatistics，您可以通过添加替代来覆盖每个评估的默认设置。每个覆盖都包括特定评估的名称和参数映射。

在中DatasetStatisticsConfiguration，配置文件作业支持CORRELATION覆盖。此覆盖计算选定列列表中两列之间的 Pearson 相关系数。默认设置是选择前 10 个数字列。您可以指定列数或列选择器列表来覆盖默认设置。

CORRELATION采用以下参数：

- columnNumber— 数字列的数量。分析作业从数据集中选择前 n 列。该值应大于 1。"ALL"用于选择所有数字列。
- columnSelectors:— 列选择器列表。每个选择器可以有列名或正则表达式。

下面是一个示例。

```
{
    "Statistic": "CORRELATION",
    "Parameters": {
        "columnSelectors": "[{\"name\":\"example\"}, {\"regex\":\"example.*\"}]"
    }
}
```

ColumnStatisticsConfigurations 部分

在结构ColumnStatisticsConfigurations部分中，您可以为特定列构建配置。ColumnStatisticsConfigurations是ColumnStatisticsConfiguration设置列表。里ColumnStatisticsConfiguration面有一个列Selectors选择器列表，Statistics用于配置统计信息。下面是一个示例。

```
{
    "Selectors": [{"Name": "example"}
    ],
    "Statistics": {
       "IncludedStatistics": ["CORRELATION", "DUPLICATE_ROWS_COUNT"]
        "Overrides": [
            {
                "Statistic": "VALUE_DISTRIBUTION",
                "Parameters": {
                    "binNumber": "10"
                }
            }
        ]
    }
}
```

Selectors是列选择器的列表。与一样ProfileColumns，您可以在每个列选择器中指定列名或正则表达式。当您指定时Selectors，列配置将应用于与中的任何列选择器匹配的列Selectors。否则，该配置将应用于所有支持的列。

在中Statistics，您可以覆盖选定列的设置。和，一样DatasetStatisticsConfiguration，Statistics有IncludedStatistics和Overrides。

要选择所需的评估，请向中添加评估名称IncludedStatistics。

```
"IncludedStatistics": ["CORRELATION", "DUPLICATE_ROWS_COUNT"]
```

如果指定IncludedStatistics，则只有列表中的评估才会包含在分析作业中。否则，配置文件作业将使用默认设置运行所有支持的评估。

您可以通过添加NONE来排除所有评估IncludedStatistics。

```
"IncludedStatistics": ["NONE"]
```

在某些情况下，可能存在多个不同的配置IncludedStatistics，您可以将其应用于同一列。ColumnStatisticsConfigurations在这些情况下，配置文件作业会选择中的最后一个配置ColumnStatisticsConfigurations并将其应用IncludedStatistics于所选列。新配置会覆盖较旧的配置。

列级别的可配置统计数据

在中ColumnStatisticsConfigurations，配置文件作业支持下表所示的评估。

此表number中支持的数据类型为意味着该属性的数据类型为以下类型之一：ByteType、ShortType、IntegerTypeLongType、FloatType、或DoubleType。

统计名称	描述	支持的数据类型	默认状态	个人资料结果的属性	个人资料结果的类型
–	列的名称。	all	–	name	字符串
–	列的数据类型。	all	–	type	字符串

统计名称	描述	支持的数据类型	默认状态	个人资料结果的属性	个人资料结果的类型
不同值计数	不同值的数量。唯一值是指至少出现一次的值。	数字/布尔值/字符串	已启用	distinctValuesCount	Int
熵	熵（信息论）。	数字/布尔值/字符串	已启用	熵	Double
INTER_QUARTIL_RANGE	范围介于数字的25%和75%之间。	number	已启用	四分位间距	Double
峰度	列的峰度。	number	已启用	峰度	Double
MAX	列中的最大值。	数字/字符串长度	已启用	max	Int/Double
最大值	列中最大值及其计数的列表。	number	已启用	最大值	列出
MEAN	列中值的平均值。	数字/字符串长度	已启用	mean	Double
MEDIAN	列中值的中位数。	数字/字符串长度	已启用	median	Double
中位数绝对偏差	每个数据点与数值列中位数之间绝对差异的中位数。	number	已启用	medianAbsoluteDeviation	Double
MIN	列中的最小值。	数字/字符串长度	已启用	min	Int/Double
最小值	列中最小值及其计数的列表。	number	已启用	最小值	列出

统计名称	描述	支持的数据类型	默认状态	个人资料结果的属性	个人资料结果的类型
缺失值计数	列中缺失值的数量。空字符串和空字符串被视为缺失。	all	已启用	missingValuesCount	Int
MODE	列中出现频率最高的值。如果经常出现几个值，则模式就是其中一个值。	数字/字符串长度	已启用	mode	Int/Double
最常用的值	列中最常见的值的列表。	数字/布尔值/字符串	已启用	mostCommnValues	列出
异常值检测	通过 z_score 算法检测列中的异常值。计算异常值的数量并从检测到的异常值中提取样本列表。	数字/字符串长度	已启用	zScoreOutliers计数，zScoreOutliers采样	Int/List
百分位数	数字列的百分位数值（5%、25%、75%、95%）。	number	已启用	百分位数 5、百分位数 25、百分位数 75、百分位数 95	Double
RANGE	列中的值范围。	number	已启用	range	Int/Double
偏度	列中值的偏度。	number	已启用	偏度	Double
标准偏差	列中值的无偏样本标准差。	数字/字符串长度	已启用	标准差	Double
SUM	列中值的总和。	number	已启用	sum	Int/Double

统计名称	描述	支持的数据类型	默认状态	个人资料结果的属性	个人资料结果的类型
唯一值计数	唯一值的数量。唯一值意味着该值仅出现一次。	数字/布尔值/字符串	已启用	uniqueValuesCount	Int
价值分布	按范围测量列中值的分布。	数字/字符串长度	已启用	价值分配	列出
VARIANCE	列中值的方差。	number	已启用	variance	Double
Z_SCORE_分布	按范围测量数据点 z 得分值的分布。	number	已启用	zScoreDistribution	列出
零计数	列中零 (0) 的数量。	number	已启用	ZerosCount	Int

在中IncludedStatistics，您可以通过添加替代来覆盖每个评估的默认参数。每个覆盖都包括特定评估的名称和参数映射。

ColumnStatisticsConfigurations 列的参数

在中ColumnStatisticsConfigurations，分析作业支持以下参数。

在某些情况下，可能存在多个不同的配置IncludedStatistics，您可以将其应用于同一列。ColumnStatisticsConfigurations在这些情况下，配置文件作业会选择中的最后一个配置ColumnStatisticsConfigurations并将其应用IncludedStatistics于所选列。新配置会覆盖较旧的配置。

最大值

列出数值列中的最大值及其计数。默认列表大小为 5。您可以通过为指定值来覆盖列表大小sampleSize。

设置

sampleSize— 列表的大小，包括数值列中的最大数目和最大值数。该值应大于 0。"ALL"用于列出所有值。

示例

```
{
    "Statistic": "MAXIMUM_VALUES",
    "Parameters": {
        "sampleSize": "5"
    }
}
```

最小值

列出数值列中的最小值及其计数。默认列表大小为 5。您可以通过为指定值来覆盖列表大小sampleSize。

设置

sampleSize— 列表的大小，包括数值列中的最大数目和最大值数。该值应大于 0。"ALL"用于列出所有值。

示例

```
{
    "Statistic": "MINIMUM_VALUES",
    "Parameters": {
        "sampleSize": "5"
    }
}
```

最常用的值

列出列中最常见的值及其计数。默认列表大小为 50。您可以通过为指定值来覆盖列表大小sampleSize。

设置

sampleSize— 列表的大小，包括数值列中的最大数目和最大值数。该值应大于 0。"ALL"用于列出所有值。

示例

```
{
    "Statistic": "MOST_COMMON_VALUES",
    "Parameters": {
        "sampleSize": "50"
    }
}
```

异常值检测

通过 z_score 算法检测数字列或字符串列中的异常值（基于字符串长度）。

您的个人资料作业会计算异常值的数量，并生成异常值及其的 z 分数的样本列表。样本列表按 z 分数的绝对值排序。默认列表大小为 50。

当一个值与均值的偏差超过标准差阈值时，Z_Score 算法会将其识别为异常值。默认异常值阈值为 3。

您可以再提供一个阈值，即温和的阈值，以获取更多信息。您的轻度阈值应小于阈值。默认情况下，此功能处于关闭状态。如果指定了温和的阈值，则您的个人资料工作会再返回一个计数zScoreMildOutliersCount。此外，在本例中zScoreOutliersSample可以包括温和阈值异常值的样本。

设置

- threshold— 检测异常值时使用的阈值。该值应大于或等于 0。
- mildThreshold— 检测异常值时使用的温和阈值。该值应大于或等于 0 且小于threshold。
- sampleSize— 列中包含异常值的列表的大小。"ALL"用于列出所有值。

示例

```
{
    "Statistic": "OUTLIER_DETECTION",
    "Parameters": {
        "threshold": "5",
        "mildThreshold": "3.5",
        "sampleSize": "20"
    }
```

```
}
```

价值分布

按值范围测量列中值的分布。配置任务按数字范围将数字列或字符串列（基于字符串长度）中的值分组到数据桶中，并生成数据桶列表。分区是连续的，存储桶的上限是下一个存储桶的下限。

设置

binNumber— 垃圾箱数量。该值应大于 0。

示例

```
{
    "Statistic": "VALUE_DISTRIBUTION",
    "Parameters": {
        "binNumber": "5"
    }
}
```

Z_SCORE_分布

测量数值的 z 分数在数值列中的分布。分析作业按数值范围将值的 z 分数分组为数据桶，并生成数据桶列表。分区是连续的，存储桶的上限是下一个存储桶的下限。

设置

binNumber— 垃圾箱数量。该值应大于 0。

示例

```
{
    "Statistic": "Z_SCORE_DISTRIBUTION",
    "Parameters": {
        "binNumber": "5"
    }
}
```

EntityDetectorConfiguration 用于配置 PII 的部分

在结构EntityDetectorConfiguration部分中，您可以将数据集中要 DataBrew 检测为个人识别信息 (PII) 的实体类型配置为个人资料作业的个人身份信息 (PII)。

EntityTypes

您可以将要 DataBrew 检测的实体类型配置为个人资料作业的 PII。如果EntityDetectorConfiguration未定义，则禁用实体检测。可以在您的数据集中检测到以下实体类型：

- USA_SSN
- EMAIL
- USA_ITIN
- USA_PASSPORT_NUMBER
- PHONE_NUMBER
- USA_DRIVING_LICENSE
- BANK_ACCOUNT
- CREDIT_CARD
- IP_ADDRESS
- MAC_ADDRESS
- USA_DEA_NUMBER
- USA_HCPCS_CODE
- USA_NATIONAL_PROVIDER_IDENTIFIER
- USA_NATIONAL_DRUG_CODE
- USA_HEALTH_INSURANCE_CLAIM_NUMBER
- USA_MEDICARE_BENEFICIARY_IDENTIFIER
- USA_CPT_CODE
- PERSON_NAME
- DATE

还支持实体类型组，USA_ALL该组包括除PERSON_NAME和之外的所有上述实体类型DATE。

的类型EntityTypes是一个字符串数组。

AllowedStatistics

配置允许在包含检测到的实体的列上运行的统计信息。如果未定义，AllowedStatistics则不会对包含检测到的实体的列计算任何统计数据。有关列级别的可配置统计数据该AllowedStatistics参数的有效值列表，请参见。

的类型AllowedStatistics是一个AllowedStatistics对象数组。

安全性 AWS Glue DataBrew

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方 AWS 的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用的合规计划 AWS Glue DataBrew，请参阅 “按合规计划划分的范围AWS 服务” 中的 “按合规计划”。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用时如何应用分担责任模型 AWS Glue DataBrew。以下主题向您介绍如何进行配置 DataBrew 以满足您的安全和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 DataBrew 资源。

主题

- 中的数据保护 AWS Glue DataBrew
- 的身份和访问管理 AWS Glue DataBrew
- 登录和监控 DataBrew
- 合规性验证 AWS Glue DataBrew
- 韧性在 AWS Glue DataBrew
- 基础设施安全 AWS Glue DataBrew
- 中的配置和漏洞分析 AWS Glue DataBrew

中的数据保护 AWS Glue DataBrew

DataBrew 提供了多项旨在帮助保护您的数据的功能。

主题

- 静态加密
- 传输中加密

- 密钥管理
- 识别和处理个人身份信息 (PII)
- DataBrew 对其他 AWS 服务的依赖

分 AWS 担责任模型适用于中的数据保护 AWS Glue DataBrew。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 AWS 责任共担模式和 GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用 SSL/TLS 与资源通信。 AWS 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API DataBrew 或 SDK 或以其他 AWS 服务 方式使用控制台 AWS CLI、API 或 AWS SDK 的情况。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

静态加密

DataBrew 支持 DataBrew 项目和作业的静态数据加密。项目和作业可以读取加密数据，作业可以通过调用 AWS Key Management Service (AWS KMS) 来写入加密数据来生成密钥和解密数据。您还可以使用 KMS 密钥对作业生成的任务日志进行加密。 DataBrew 您可以使用 DataBrew 控制台或 DataBrew API 指定加密密钥。

> **Important**
>
> AWS Glue DataBrew 仅支持对称 AWS KMS 密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS 密钥。

在 DataBrew 启用加密的情况下创建任务时，您可以使用 DataBrew 控制台指定 S3 托管的服务器端加密密钥 (SSE-S3) 或存储在 (SSE-KMS) 中的 AWS KMS KMS 密钥来加密静态数据。

> **Important**
>
> 当您使用 Amazon Redshift 数据集时，卸载到所提供的临时目录的对象将使用 SSE-S3 进行加密。

加密作业写入的数据 DataBrew

DataBrew 任务可以写入加密的 Amazon S3 目标和加密的 Amazon CloudWatch 日志。

主题

- 设置 DataBrew 为使用加密
- 为 VPC 任务创建通往 AWS KMS 的路由
- 使用 AWS KMS 密钥设置加密

设置 DataBrew 为使用加密

按照以下步骤将您的 DataBrew 环境设置为使用加密。

将您的 DataBrew 环境设置为使用加密

1. 创建或更新您的 AWS KMS 密钥以向传递给 DataBrew 任务的 AWS Identity and Access Management (IAM) 角色 AWS KMS 授予权限。这些 IAM 角色用于加密 CloudWatch 日志和 Amazon S3 目标。有关更多信息，请参阅 Amazon Lo CloudWatch gs 用户指南 AWS KMS中的使用加密 CloudWatch 日志数据。

   在以下示例中，*"role1""role2"*、和*"role3"*是传递给 DataBrew 任务的 IAM 角色。本策略声明描述了 KMS 密钥策略，该策略允许列出的 IAM 角色使用此 KMS 密钥进行加密和解密。

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "logs.region.amazonaws.com",
        "AWS": [
            "role1",
            "role2",
            "role3"
        ]
    },
    "Action": [
        "kms:Encrypt*",
        "kms:Decrypt*",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:Describe*"
    ],
    "Resource": "*"
}
```

如果您使用密钥对 CloudWatch 日志进行加密`"Service": "logs.region.amazonaws.com"`，则必须使用该`Service`语句（如所示）。

2. 确保在使用 AWS KMS 密钥`ENABLED`之前将其设置为。

有关使用 AWS KMS 密钥策略指定权限的更多信息，请参阅中的使用密钥策略 AWS KMS。

为 VPC 任务创建通往 AWS KMS 的路由

您可以通过 Virtual Private Cloud (VPC) 中的私有终端节点直接连接到 AWS KMS ，而不是通过互联网连接。当您使用 VPC 终端节点时，您 AWS KMS 的 VPC 和 VPC 之间的通信完全在 AWS 网络内进行。

您可以在 AWS KMS VPC 内创建 VPC 终端节点。如果不执行此步骤，您的 DataBrew 作业可能会失败`kms timeout`。有关详细说明，请参阅《AWS Key Management Service 开发人员指南》中的“AWS KMS 通过 VPC 终端节点连接”。

按照这些说明进行操作时，请务必在 VPC 控制台上执行以下操作：

- 选择启用私有 DNS 名称。

- 对于安全组，选择用于访问 Java 数据库连接 (JDBC) 的 DataBrew 作业的安全组（包括自引用规则）。

运行访问 JDBC 数据存储的 DataBrew 作业时， DataBrew 必须有通往终端节点的路由。 AWS KMS 您可以为路由提供网络地址转换 (NAT) 网关或 AWS KMS VPC 终端节点。要创建 NAT 网关，请参阅 Amazon VPC 用户指南中的 NAT 网关。

使用 AWS KMS 密钥设置加密

当您对任务启用加密时，它会同时适用于 Amazon S3 和 CloudWatch。传递的 IAM 角色必须具有以下 AWS KMS 权限。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey*"
        ],
        "Resource": "arn:aws:kms:region:account-id:key/key-id"
    }
}
```

有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的以下主题：

- 有关 SSE-S3 的更多信息，请参阅使用具有 Amazon S3 托管加密密钥 (SSE-S3) 的服务器端加密 (SSE-S3) 保护数据。
- 有关信息SSE-KMS，请参阅使用 KMS AWS 托管密钥的服务器端加密 (SSE-KMS) 保护数据。

传输中加密

AWS 为传输中的数据提供安全套接字层 (SSL) 加密。

DataBrew 对 JDBC 数据源的支持得以实现。 AWS Glue连接到 JDBC 数据源时， DataBrew 使用 AWS Glue 连接上的设置，包括 “需要 SSL 连接” 选项。有关更多信息，请参阅《AWS Glue 开发人员指南》 AWS Glue中的AWS Glue 连接属性。

AWS KMS 提供 “自带密钥” 加密和服务器端加密，用于 DataBrew 提取、转换、加载 (ETL) 处理和。AWS Glue Data Catalog

密钥管理

您可以将 IAM 与结合使用 DataBrew 来定义用户、 AWS 资源、群组、角色以及有关访问、拒绝等的精细策略。

您可以根据组织的需求，使用基于资源和基于身份的策略来定义对元数据的访问权限。基于资源的策略列出了允许或拒绝访问您的资源的委托人，允许您设置跨账户访问等策略。身份策略专门附加到 IAM 中的用户、组和角色。

DataBrew 支持创建自己 AWS KMS key 的 “自带密钥” 加密。 DataBrew 还使用来自的 KMS 密钥 AWS KMS 为 DataBrew 任务提供服务器端加密。

识别和处理个人身份信息 (PII)

在构建分析函数或机器学习模型时，需要采取保护措施来防止个人身份信息 (PII) 数据泄露。PII 是可用于识别个人身份的个人数据，例如地址、银行账号或电话号码。例如，当数据分析师和数据科学家使用数据集来发现一般的人口统计信息时，他们不应有权访问特定个人的 PII。

DataBrew 提供数据屏蔽机制，用于在数据准备过程中模糊处理 PII 数据。根据贵组织的需求，有不同的 PII 数据编辑机制可供选择。您可以对 PII 数据进行模糊处理，这样用户就无法将其还原，也可以使混淆变得可逆的。

识别和屏蔽中的 PII 数据 DataBrew 涉及构建一组转换，客户可以使用这些转换来编辑 PII 数据。此过程的一部分是在 DataBrew 控制台上的数据配置文件概述仪表板中提供 PII 数据检测和统计信息。

您可以使用以下数据屏蔽技术：

- 替换-将 PII 数据替换为其他外观真实的值。
- 洗牌 — 将同一列的值洗到不同的行中。
- 确定性加密-对列值应用确定性加密算法。确定性加密总是为值生成相同的密文。
- 概率加密-对列值应用概率加密算法。概率加密每次应用时都会生成不同的密文。
- 解密-根据加密密钥解密列。
- 清空或删除-将特定字段替换为空值或删除该列。
- 屏蔽-使用字符乱写或屏蔽列中的某些部分。

- 哈希-对列值应用哈希函数。

有关使用转换的更多信息，请参阅个人身份信息 (PII) 配方步骤。有关使用配置文件作业检测 PII 的更多信息，包括可以检测到的实体类型列表，请参阅以编程方式构建配置文件作业配置中有关配置 PII 的 EntityDetectorConfiguration 部分。

DataBrew 对其他 AWS 服务的依赖

要使用 DataBrew 控制台，您需要一组最低权限才能使用 AWS 账户的 DataBrew 资源。除这些 DataBrew 权限外，控制台还需要以下服务的权限：

- CloudWatch 记录显示日志的权限。
- 用于列出和传递角色的 IAM 权限。
- 列出 VPC、子网、安全组、实例和其他对象的 Amazon EC2 权限。 DataBrew 在运行 DataBrew 任务时使用这些权限来设置 Amazon EC2 项目，例如 VPC。
- Amazon S3 列出存储桶和对象的权限。
- AWS Glue 读取 AWS Glue 架构对象（例如数据库、分区、表和连接）的权限。
- AWS Lake Formation 使用 Lake Formation 数据湖的权限。

的身份和访问管理 AWS Glue DataBrew

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（拥有权限）使用 DataBrew 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

- 使用身份进行身份验证
- 使用策略管理访问
- AWS Glue DataBrew 和 AWS Lake Formation
- 如何 AWS Glue DataBrew 与 IAM 配合使用
- 适用于 AWS Glue DataBrew的基于身份的策略示例
- AWS 的托管策略 AWS Glue DataBrew
- 对中的身份和访问进行故障排除 AWS Glue DataBrew

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的多重身份验证和《IAM 用户指南》中的在 AWS中使用多重身份验证（MFA）。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建帐户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

用户和组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
  - 服务相关角色-服务相关角色是一种与服务相关联的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

- 在 A@@ mazon EC2 上运行的应用程序 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

DataBrew 不支持基于资源的策略。

访问控制列表 (ACL)

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》 中的访问控制列表（ACL）概览。

DataBrew 不支持 ACL。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 - 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的

策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

AWS Glue DataBrew 和 AWS Lake Formation

AWS Glue DataBrew 支持对 AWS Glue Data Catalog 表的 AWS Lake Formation 权限。当数据集使用在 Lake Formation 中注册的 AWS Glue Data Catalog 表时，提供给项目或作业的 IAM 角色必须对该表具有描述和 SE LECT Lake Formation 权限。

AWS Glue DataBrew 支持基于写入 AWS Glue Data Catalog 表 AWS Lake Formation。当 DataBrew 作业使用在 Lake Formation 中注册的数据目录时，提供给这些任务的 IAM 角色必须拥有 Lake Formation 对相关表的插入、更改和删除权限。IAM 角色必须拥有`glue:UpdateTable`权限以及对与数据目录表关联的数据位置的权限。

如何 AWS Glue DataBrew 与 IAM 配合使用

在使用 IAM 管理访问权限之前 DataBrew，您应该了解哪些可用的 IAM 功能 DataBrew。要全面了解如何 DataBrew 和其他 AWS 服务与 IAM 配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS服务。

主题

- DataBrew 基于身份的策略
- 中基于资源的政策 DataBrew
- DataBrew IAM 角色

DataBrew 基于身份的策略

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源，以及允许或拒绝操作的条件。DataBrew 支持特定的操作、资源和条件键。要了解在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》 中的 IAM JSON 策略元素参考。

操作

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说， AWS JSON 策略可以指定哪些委托人可以在什么条件下对哪些资源执行操作。

JSON 策略的操作元素描述了您可以在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

正在执行的策略操作在操作前 DataBrew 使用以下前缀:`databrew:`. 例如，要授予某人使用 Amazon EC2 `RunInstances` API 操作运行 Amazon EC2 实例的权限，您应将 `ec2:RunInstances` 操作纳入其策略。策略语句必须包含 `Action` 或 `NotAction` 元素。 DataBrew 定义了它自己的一组操作，这些操作描述了你可以用它执行的任务。

要在单个语句中指定多项 操作，请使用逗号将它们隔开，如下所示。

```
"Action": [
      "databrew:CreateRecipeJob",
      "databrew:UpdateSchedule"
```

您也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，请包括以下操作。

```
"Action": "databrew:Describe*"
```

要查看 DataBrew 操作列表，请参阅 IAM 用户指南 AWS Glue DataBrew中的定义操作。

资源

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

以下是不支持资源级权限 DataBrew 的 API：

- ListDatasets
- ListJobs
- ListProjects
- ListRecipes
- ListRulesets
- ListSchedules

DataBrew 数据集资源具有以下亚马逊资源名称 (ARN)。

```
arn:${Partition}:databrew:${Region}:${Account}:dataset/${Name}
```

有关 ARN 格式的更多信息，请参阅 A mazon 资源名称 (ARN) 和 AWS 服务命名空间。

例如，要在语句中指定 `i-1234567890abcdef0` 实例，请使用以下 ARN。

```
"Resource": "arn:aws:databrew:us-east-1:123456789012:dataset/my-chess-dataset"
```

要指定属于特定账户的所有实例，请使用通配符 (*)。

```
"Resource": "arn:aws:databrew:us-east-1:123456789012:dataset/*"
```

您无法对特定资源执行某些 DataBrew 操作，例如创建资源的操作。在这些情况下，您必须使用通配符 (*)。

```
"Resource": "*"
```

要查看 DataBrew 资源类型及其 ARN 的列表，请参阅 IAM 用户指南 AWS Glue DataBrew中的由定义的资源。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 AWS Glue DataBrew定义的操作。

条件键

DataBrew 不提供任何特定于服务的条件密钥，但它确实支持使用一些全局条件密钥。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

示例

要查看 DataBrew 基于身份的策略的示例，请参阅。适用于 AWS Glue DataBrew的基于身份的策略示例

中基于资源的政策 DataBrew

DataBrew 不支持基于资源的策略。

DataBrew IAM 角色

I AM 角色是您的 AWS 账户中具有特定权限的实体。

将临时证书与 DataBrew

可以使用临时凭证进行联合身份验证登录，分派 IAM 角色或分派跨账户角色。您可以通过调用AssumeRole或之类的 AWS STS API 操作来获得临时安全证书GetFederationToken。

DataBrew 支持使用临时证书。

服务相关角色

服务相关角色允许 AWS 服务访问其他服务中的资源以代表您完成操作。服务相关角色显示在 IAM 账户中，并归该服务所有。管理员可以查看但不能编辑服务相关角色的权限。

在中选择 IAM 角色 DataBrew

在中创建数据集资源时 DataBrew，您可以选择一个 IAM 角色来允许您进行 DataBrew 访问。如果您之前创建过服务角色或服务相关角色，则会 DataBrew 为您提供可供选择的角色列表。请务必根据需要选择允许对 Amazon S3 存储桶或 AWS Glue Data Catalog 资源进行读取权限的角色。

适用于 AWS Glue DataBrew的基于身份的策略示例

默认情况下，用户和角色无权创建或修改 DataBrew 资源。他们也无法使用 AWS Management Console AWS CLI、或 AWS API 执行任务。管理员必须创建 IAM policy，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅《IAM 用户指南》中的在 JSON 选项卡上创建策略。

主题

- 策略最佳实践
- 使用控制 DataBrew 台
- 允许用户查看他们自己的权限
- 基于标签管理 DataBrew 资源

策略最佳实践

基于身份的策略决定了某人是否可以在您的账户中创建、访问或删除 DataBrew 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限 – 在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的 配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实操。

使用控制 DataBrew 台

要访问 AWS Glue DataBrew 控制台，您必须拥有一组最低权限。这些权限必须使您能够列出和查看有关您 AWS 账户中 DataBrew 资源的详细信息。如果您创建的基于身份的策略比所需的最低权限更严格，则控制台将无法按预期运行，适用于使用该策略的用户或角色。

为确保用户和角色可以使用 DataBrew 控制台，还要将以下 AWS 托管策略附加到实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

```
AWSDataBrewConsoleAccess
```

对于仅调用 AWS CLI 或 DataBrew API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与您尝试执行的 API 操作相匹配的操作。

允许用户查看他们自己的权限

该示例说明了如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

基于标签管理 DataBrew 资源

您可以使用基于身份的策略中的条件根据标签管理 DataBrew 资源，例如删除、更新或描述资源。以下示例显示了拒绝删除项目的策略。但是，只有当项目标签 Owner 的值为 admin 时，才会拒绝删除。此策略还授予在控制台上拒绝此操作所需的权限。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "DeleteResourceInConsole",
         "Effect": "Allow",
         "Action": "databrew:DeleteProject",
         "Resource": "*"
       },
       {
         "Sid": "DenyDeleteProjectIfAdminTag",
         "Effect": "Deny",
         "Action": "databrew:DeleteProject",
         "Resource": "arn:aws:databrew:*:*:project/*",
         "Condition": {
            "StringEquals": {"aws:ResourceTag/Owner": "admin"}
         }
      }
   ]
}
```

您可以将此策略附加到您账户中的用户。如果名为 richard-roe 的用户试图删除 DataBrew 项目，则不得将该资源标记为 owner=admin 或 owner =admin。否则，用户将被拒绝删除项目的权限。条件标签密钥所有者同时匹配所有者和所有者，因为条件键名称不区分大小写。有关更多信息，请参阅 IAM 用户指南 中的 IAM JSON 策略元素：条件。

> **Note**
>
> ListDatasets、 ListJobs、 ListProjects、 ListRecipes、 ListRulesets、和 ListSchedules 不支持基于标签的访问控制。

AWS 的托管策略 AWS Glue DataBrew

要向用户、群组和角色添加权限，使用 AWS 托管策略比自己编写策略要容易得多。创建仅为团队提供所需权限的 IAM 客户管理型策略需要时间和专业知识。要快速入门，您可以使用我们的 AWS 托管策略。这些政策涵盖常见用例，可在您的 AWS 账户中使用。有关 AWS 托管策略的更多信息，请参阅 IAM 用户指南中的AWS 托管策略。

AWS 服务维护和更新 AWS 托管策略。您无法更改 AWS 托管策略中的权限。服务偶尔会向 AWS 托管策略添加其他权限以支持新功能。此类更新会影响附加策略的所有身份（用户、组和角色）。当推出新功能或有新操作可用时，服务最有可能更新 AWS 托管策略。服务不会从 AWS 托管策略中移除权限，因此策略更新不会破坏您的现有权限。

此外，还 AWS 支持跨多个服务的工作职能的托管策略。例如，ReadOnlyAccess AWS 托管策略提供对所有 AWS 服务和资源的只读访问权限。当服务启动一项新功能时， AWS 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅《IAM 用户指南》中的适用于工作职能的AWS 托管策略。

DataBrew AWS 托管策略的更新

查看 DataBrew 自该服务开始跟踪这些更改以来 AWS 托管策略更新的详细信息。要获得有关此页面变更的自动提醒，请订阅 “ DataBrew 文档历史记录” 页面上的 RSS feed。可在 AWS IAM 控制台上找到托管策略，网址为AwsGlueDataBrewFullAccessPolicy。

更改	描述	日期
AWSGlueDataBrewServiceRole— 添加 AWS Glue 了的读取权限。	此更新添加`glue:GetCustomEntityType` 了。在启用 PII 识别的情况下执行 AWS Glue DataBrew 配置文件作业需要此权限。	2024 年 3 月 20 日
AWSGlueDataBrewServiceRole-已添加的 AWS Glue 读取权限。	此更新添加`glue:BatchGetCustomEntityTypes` 了。在启用 PII 识别的情况下执行 AWS Glue DataBrew 配置文件作业需要此权限。	2022 年 5 月 9 日
AwsGlueDataBrewFullAccessPolicy-添加了 Amazon	此更新增加了`redshift-data:DescribeState`	2022 年 2 月 4 日

更改	描述	日期
Redshift-Data DescribeS tatements 和 Ama GetLifecy cleConfiguration zon S3 的读取权限。	ment 在创建基于 Amazon Redshift 的数据集时验证您的 SQL 的支持。它还可以评估s3:GetLifecycleCon figuration 您作为临时目录提供的 Amazon S3 存储桶前缀是否配置了生命周期。此外，此更改将 “databrew: *” 权限替换为包含所有 DataBrew API 的明确权限列表。	
AwsGlueDataBrewFul lAccessPolicy-添加了 S AWS ecrets Manager 的读/写权限。	此更新为名secretsma nager:GetSecretVal ue databrew! default 为secretsma nager:CreateSecret 的密钥添加了一个用于 DataBrew 变换的默认密钥。此外，它还为前缀 CreateSecret 为的密钥添加了权限，AwsGlueDa taBrew- 以便从 DataBrew 控制台创建密钥。GenerateR andom AWS Key Managemen t ServiceAPI 参考中所述，用于生成加密安全的随机字节字符串。	2021 年 11 月 18 日
AWSGlueDataBrewSer viceRole-添加了 S AWS ecrets Manager 的读/写权限。	此更新为名secretsma nager:GetSecretVal ue databrew!default 为的密钥添加了一个用于 DataBrew 变换的默认密钥。	2021 年 11 月 18 日

更改	描述	日期
AwsGlueDataBrewFullAccessPolicy-添加了 S AWS ecrets Manager 的读/写权限。	此更新为名secretsmanager:GetSecretValue databrew!default 为secretsmanager:CreateSecret 的密钥添加了一个用于 DataBrew 变换的默认密钥。此外，它还为前缀 CreateSecret 为的密钥添加了权限，AwsGlueDataBrew- 以便从 DataBrew 控制台创建密钥。 kms:GenerateRandom (https://docs.aws.amazon.com/kms/latest/APIReference/API_GenerateRandom.html) 用于生成加密安全的随机字节字符串。	2021 年 11 月 18 日
AWSGlueDataBrewServiceRole-添加了 S AWS ecrets Manager 的读/写权限。	此更新为名secretsmanager:GetSecretValue databrew!default 为的密钥添加了一个用于 DataBrew 变换的默认密钥。	2021 年 11 月 18 日
AwsGlueDataBrewFullAccessPolicy-添加了 AWS Glue 目录数据库的读取权限和 AWS Glue 目录表的创建权限。	此更新增加了列出 AWS Glue 目录数据库和在现有数据库下创建新目录表的权限，这是配置 DataBrew 作业输出的一部分。	2021 年 6 月 30 日
AwsGlueDataBrewFullAccessPolicy-添加了 Amazon AppFlow 数据集功能的读/写权限。	此更新增加了读取现有 Amazon AppFlow 流程和流程执行以及创建流程执行的权限。	2021 年 4 月 28 日

更改	描述	日期
AwsGlueDataBrewFullAccessPolicy-添加了数据库数据集的读取权限。	此更新增加了读取现有 AWS Glue 连接和创建用于的新 AWS Glue 连接的权限 DataBrew。 此外，为了简化创建新连接的控制台体验，它允许列出 Amazon VPC 资源和 Amazon Redshift 集群。它还允许列出但不能读取 AWS Secrets Manager 机密。	2021 年 3 月 30 日
DataBrew 开始跟踪更改	DataBrew 开始跟踪其 AWS 托管策略的更改。	2021 年 3 月 30 日

对中的身份和访问进行故障排除 AWS Glue DataBrew

使用以下信息来帮助您诊断和修复在使用 DataBrew 和 IAM 时可能遇到的常见问题。

主题

- 我无权在以下位置执行操作 DataBrew
- 我无权执行 iam：PassRole
- 我想允许 AWS 账户之外的人访问我的 DataBrew 资源

我无权在以下位置执行操作 DataBrew

如果 AWS Management Console 告诉您您无权执行某项操作，请联系您的管理员寻求帮助。管理员是向您提供登录凭证的人。

当 `mateojackson` 用户尝试使用控制台查看有关项目的详细信息但不具有 `databrew:DescribeProject` 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 databrew:DescribeProject on resource: my-example-project
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `databrew:GetProject` 操作访问 *`my-example-project`* 资源。

我无权执行 iam：PassRole

如果您收到错误消息，提示您无权执行`iam:PassRole`操作，则必须更新您的策略以允许您将角色传递给 DataBrew。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的 IAM 用户`marymajor`尝试使用控制台在中执行操作时，会出现以下示例错误 DataBrew。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许 AWS 账户之外的人访问我的 DataBrew 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表（ACL）的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解是否 DataBrew 支持这些功能，请参阅如何 AWS Glue DataBrew 与 IAM 配合使用。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户存取之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

登录和监控 DataBrew

监控是维护 AWS 解决方案的可靠性、可用性和性能的重要组成部分。 DataBrew 您应该从 AWS 解决方案的所有部分收集监控数据，以便在出现多点故障时可以更轻松地进行调试。 AWS 提供了多种用于监控您的 DataBrew 资源和应对潜在事件的工具：

亚马逊 CloudWatch 警报

使用 Amazon CloudWatch 警报，您可以监控您指定的时间段内的单个指标。如果指标超过给定阈值，则会向 Amazon SNS 主题或 AWS Auto Scaling 政策发送通知。 CloudWatch 警报不会调用操作，因为它们处于特定状态。而是必须在状态已改变并在指定的若干个时间段内保持不变后才调用。

AWS CloudTrail 日志

CloudTrail 提供了用户、角色或 AWS 服务在中执行的操作的记录 DataBrew。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 DataBrew、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

合规性验证 AWS Glue DataBrew

AWS Glue DataBrew 作为多个合规计划的一部分，第三方审计师对安全性和 AWS 合规性进行评估。其中包括 SOC、PCI、FedRAMP、HIPAA 及其他。

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了在这些基础上 AWS 部署以安全性和合规性为重点的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规架构 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业指导方针和法规。
- AWS Security Hub— 这 AWS 服务 提供了您内部安全状态的全面视图 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

韧性在 AWS Glue DataBrew

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。 AWS 区域提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络相连。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

对于 AWS Glue DataBrew，我们建议您将任务配置为使用一次或多次重试。作业的重试次数是在 DataBrew 控制台的 “高级作业设置” 下配置的。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。

基础设施安全 AWS Glue DataBrew

作为托管服务的一部分， AWS Glue DataBrew 受《Amazon Web Services：安全流程概述》白皮书中描述的 AWS 全球网络安全程序的保护。

您可以使用 AWS 已发布的 API 调用 DataBrew 通过网络进行访问。客户端必须支持传输层安全性协议 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密（PFS）的密码套件，例如 Ephemeral Diffie-Hellman（DHE）或 Elliptic Curve Ephemeral Diffie-Hellman（ECDHE）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

主题

在您的 AWS Glue DataBrew VPC 中使用

如果您使用 Amazon VPC 托管 AWS 资源，则可以配置 AWS Glue DataBrew 为基于亚马逊 VPC 服务的虚拟私有云 (VPC) 路由流量。 DataBrew 为此，首先在您指定的子网中配置一个 elastic network 接口。 DataBrew 然后将您指定的安全组附加到该网络接口以控制访问权限。指定的安全组必须具有适用于所有流量的自引用入站和出站规则。此外，您的 VPC 必须启用 DNS 主机名和解析。有关更多信息，请参阅AWS Glue 开发人员指南中的设置 VPC 以连接到 JDBC 数据存储。

对于 AWS Glue Data Catalog 数据集，VPC 信息是在您在数据目录中创建 AWS Glue 连接时配置的。要为此连接创建数据目录表，请从 AWS Glue 控制台运行 Crawler。有关更多信息，请参阅《AWS Glue 开发人员指南》 AWS Glue Data Catalog中的填充。

对于数据库数据集，请在从 DataBrew 控制台创建连接时指定您的 VPC 信息。

要 AWS Glue DataBrew 与没有 NAT 的 VPC 子网配合使用，您必须有一个连接到 Amazon S3 的网关 VPC 终端节点和用于 AWS Glue 接口的 VPC 终端节点。有关更多信息，请参阅 Amazon VPC 文档中的创建网关终端节点和接口 VPC 终端节点 (AWS PrivateLink)。由配置的弹性接口 DataBrew 没有公有 IPv4 地址，因此它不支持使用 VPC Internet Gateway。

目前不支持 Amazon S3 接口终端节点。如果您使用 AWS Secrets Manager 存储密钥，则需要一条通往 Secrets Manager 的路径。如果您使用的是加密，则需要一个到 AWS Key Management Service (AWS KMS) 的路由。

AWS Glue DataBrew 与 VPC 终端节点一起使用

如果您使用 Amazon VPC 托管 AWS 资源，则可以通过预置 VPC 终端节点在您 DataBrew 的 VPC 之间建立私有连接。使用此 VPC 终端节点，您可以进行 DataBrew API 调用。

您的 DataBrew VPC 无需使用 DataBrew VPC 终端节点。有关更多信息，请参阅 在您的 AWS Glue DataBrew VPC 中使用。

您可以在所有 AWS 区域中同时支持两者的 VPC 终端节点 AWS Glue 和 VPC 终端节点 AWS Glue 一起使用。

有关更多信息，请参阅《Amazon VPC 用户指南》中的以下主题：

- 什么是 Amazon VPC？
- 创建接口终端节点

中的配置和漏洞分析 AWS Glue DataBrew

配置和 IT 控制由您（我们的客户）共同 AWS 负责。有关更多信息，请参阅责任 AWS 共担模型。

监控 AWS Glue DataBrew

监控是维护和其他 AWS 解决方案的可靠性、可用性和性能的重要组成部分。 AWS Glue DataBrew AWS 提供以下监控工具 DataBrew，供您监视、报告问题并在适当时自动采取措施：

- Amazon 会实时 CloudWatch监控您的 AWS 资源和您运行 AWS 的应用程序。您可以收集和跟踪指标，创建自定义的控制平面，以及 设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以 CloudWatch 跟踪您的 Amazon EC2 实例的 CPU 使用率或其他指标，并在需要时自动启动新实例。有关更多信息，请参阅 Amazon CloudWatch 用户指南。
- Amazon CloudWatch Events 允许您为中的特定事件设置自动通知 DataBrew。来自 DataBrew 的事件以近乎实时的方式传递到 CloudWatch 事件。您可以将 CloudWatch 事件配置为监控事件，并调用目标以响应表明您的资源共享发生变化的事件。对资源共享进行更改会针对资源共享的所有者以及获授权访问资源共享的主体触发事件。有关更多信息，请参阅 Amazon CloudWatch 活动用户指南。
- Amazon CloudWatch Logs 允许您监控、存储和访问来自 Amazon EC2 实例和其他来源的日志文件。 CloudTrail CloudWatch 日志可以监视日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南。
- AWS CloudTrail捕获由您的账户或代表您的 AWS 账户进行的 API 调用和相关事件。然后它将日志文件传送到您指定的 Amazon S3 存储桶。您可以识别哪些用户和账户拨打了电话 AWS、发出呼叫的源 IP 地址以及呼叫发生的时间。有关更多信息，请参阅《AWS CloudTrail 用户指南》。

主题

DataBrew 使用 Amazon 进行监控 CloudWatch

您可以使用 DataBrew 进行监控 CloudWatch，它收集原始数据并将其处理为可读的近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

AWS Glue DataBrew 报告命AWS/DataBrew名空间中的以下指标。

指标	描述
SessionCount	客户账户中的 DataBrew 会话总数 有效尺寸： LogGroupName 有效统计数据：Sum 单位：计数

DataBrew 使用 CloudWatch 事件自动化

Amazon CloudWatch Events 使您能够实现 AWS 服务自动化，并自动响应系统事件，例如应用程序可用性问题或资源更改。来自 AWS 服务的事件几乎实时地传递到 CloudWatch 活动。您可以编写简单规则来指示您关注的事件，并指示要在事件匹配规则时执行的自动化操作。可自动触发的操作包括：

- 调用 Amazon EC2 运行命令
- 将事件中继到 Amazon Kinesis Data Streams
- 激活 AWS Step Functions 状态机
- 通知 Amazon SNS 主题或 Amazon SQS 队列

DataBrew 每当您的 AWS 账户中的资源状态发生变化时，都会向 Events 报告 CloudWatch 事件。尽最大努力发出事件。

以下是几个事件的示例，显示了 DataBrew 作业的不同状态：SUCCEEDEDFAILED、TIMEOUT、和STOPPED。

```
{
  "version": "0",
  "id": "abcdef00-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-09-07T18:57:21Z",
  "region": "us-west-2",
  "resources": [],
```

```
  "detail": {
    "jobName": "MyJob",
    "severity": "INFO",
    "state": "SUCCEEDED",
    "jobRunId": "db_abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789",
    "message": "Job run succeeded"
  }
}

{
  "version": "0",
  "id": "abcdef01-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-09-07T06:02:03Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "ERROR",
    "state": "FAILED",
    "jobRunId": "db_0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef",
    "message": "AnalysisException: 'Path does not exist: s3://MyBucket/MyFile;'"
  }
}

{
  "version": "0",
  "id": "abcdef00-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-11-20T20:22:06Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "WARN",
    "state": "TIMEOUT",
    "jobRunId": "db_abc0123456789abcdef0123456789abcdef0123456789abcdef0123456789def",
    "message": "Job run timed out"
  }
}
```

```
{
  "version": "0",
  "id": "abcdef00-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-11-20T20:22:06Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "INFO",
    "state": "STOPPED",
    "jobRunId": "db_abc0123456789abcdef0123456789abcdef0123456789abcdef0123456789def",
    "message": "Job run stopped"
  }
}
```

有关更多信息，请参阅 Amazon CloudWatch 活动用户指南。

DataBrew 使用 CloudWatch 日志进行监控

您可以使用日志监控 DataBrew 作业， CloudWatch 日志从 DataBrew作业子系统收集详细信息并使其可供查看。如果您想深入了解您的个人资料和食谱任务正在使用的资源，或者出于故障排除的目的，这些 CloudWatch 日志可能会很有用。有关更多信息，请参阅 Amazon Logs 用户指南。

使用记录 DataBrew API 调用 AWS CloudTrail

DataBrew 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在中执行的操作的记录 DataBrew。 CloudTrail 将所有 API 调用捕获 DataBrew 为事件。捕获的调用包括来自 DataBrew 控制台的调用和对 DataBrew API 操作的代码调用。如果您创建了跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括的事件 DataBrew。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 DataBrew。还可以确定发出请求的源 IP 地址、请求方、请求时间以及其他详细信息。

要了解更多信息 CloudTrail，请参阅AWS CloudTrail 用户指南。

DataBrew 中的信息 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当活动发生在中时 DataBrew，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在自己的 AWS 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 CloudTrail 事件历史记录查看事件。

要持续记录您 AWS 账户中的事件，包括的事件 DataBrew，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，当您在控制台中创建跟踪时，该跟踪将应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的以下内容：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

所有 DataBrew 操作都由记录 CloudTrail 并记录在API 参考... 例如，调用`UpdateRecipe`和`StartJobRun`操作会在 CloudTrail 日志文件中生成条目。`CreateDataset`

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根凭证还是用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅CloudTrail 用户身份元素。

了解 DataBrew 日志文件条目

同样， CloudTrail 跟踪是一种配置，它允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例显示了演示该`CreateProfileJob`操作的 CloudTrail 日志条目。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::1234567890:user/joe",
        "accountId": "1234567890",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "joe"
    },
    "eventTime": "2020-11-09T18:54:44Z",
    "eventSource": "databrew.amazonaws.com",
    "eventName": "CreateProfileJob",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "requestParameters": {
        "OutputLocation": {
            "Bucket": "bucketName",
            "Key": "keyName"
        },
        "DatasetName": "my-chess-dataset",
        "RoleArn": "arn:aws:iam::1234567890:role/custom-role",
        "Name": "my-profile-job"
    },
    "responseElements": {
        "Name": "my-profile-job"
    },
    "requestID": "993bc3b8-3980-48dd-961e-c1c8529eb248",
    "eventID": "f8128dfa-df29-458b-a2d5-34805b46eefd",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "1234567890"
}
```

在 D AWS Glue atabrew 中使用 AWS 用户通知

您可以使用AWS 用户通知来设置传递渠道，以获得有关 AWS Glue Databrew 事件的通知。当事件与指定的规则匹配时，会收到通知。可以通过多个渠道接收事件通知，包括电子邮件、AWS Chatbot聊天通知或AWS Console Mobile Application推送通知。还可以在控制台通知中心中查看通知。 AWS 用户通知支持聚合，这可以减少您在特定事件期间收到的通知数量。

配方步骤和功能参考

在本参考文献中，您可以找到配方步骤和函数的描述，您可以通过编程方式使用这些步骤和函数，AWS CLI 或者使用其中一个 AWS SDK。在中 DataBrew，配方步骤是一种将原始数据转换为可供数据管道使用的表单的操作。 DataBrew 函数是一种特殊的配方步骤，它根据参数执行计算。

用户界面中转换的类别包括以下几种：

- 基本专栏配方步骤
  - 筛选条件
  - 列
- 数据清理配方步骤
  - 格式
  - 清除
  - Extract
- 数据质量配方步骤
  - 缺失
  - 无效
  - 重复
  - 异常值
- 个人身份信息（PII）配方步骤
  - 屏蔽个人信息
  - 替换个人信息
  - 加密个人信息
  - 随机排列行
- 列结构配方步骤
  - Split
  - 合并
  - 创建
- 列格式化配方步骤
  - DECIMAL
  - 千位分隔符

    - 缩写
- 数据结构配方步骤
    - Nest-Unnest
    - Pivot
    - 组
    - 联接
    - Union
- 数据科学配方步骤
    - 文本
    - 比例
    - Mapping
    - 编码
- 函数
    - 数学函数
    - 聚合函数
    - 文本函数
    - 日期和时间函数
    - 窗口函数
    - 网络函数
    - 其他函数

有关如何在配方中使用这些配方步骤和函数（包括条件表达式的使用）的更多信息，请参阅定义配方结构。

以下各节描述了配方步骤和功能，按其作用进行组织。

主题

- 基本专栏配方步骤
- 数据清理配方步骤
- 数据质量配方步骤
- 个人身份信息（PII）配方步骤
- 异常值检测和处理配方步骤

- 列结构配方步骤
- 列格式化配方步骤
- 数据结构配方步骤
- 数据科学配方步骤
- 数学函数
- 聚合函数
- 文本函数
- 日期和时间函数
- 窗口函数
- 网络函数
- 其他函数

基本专栏配方步骤

使用这些基本的列配方操作对数据执行简单的转换。

主题

- 更改数据类型
- 删除
- 重复
- JSON_TO_STRUCTS
- 移动
- 之前移动
- 移到尽头
- 移至索引
- 移至起点
- RENAME
- SORT
- TO_BOOLEAN_COLUMN
- TO_DOUBLE_COLU

- TO_NUMBER_列
- 到字符串列

更改数据类型

更改现有列的数据类型。

如果无法将列值转换为新类型，则该列值将被替换为 NULL。当字符串列转换为整数列时，可能会发生这种情况。例如，字符串 “123” 将变为整数 123，但字符串 “ABC” 不能变成数字，因此它将被替换为空值。

参数

- `sourceColumn` – 现有列的名称。
- `columnDataType`— 列的新类型。支持以下数据类型：
  - byte：1 字节有符号整数。数字的范围从 -128 到 127 不等。
  - 短：2 字节有符号整数。数字的范围从 -32768 到 32767 不等。
  - int：4 字节有符号整数。数字的范围从 -2147483648 到 2147483647 不等。
  - long：8 字节的有符号整数。数字范围从-9223372036854775808到9223372036854775807。
  - float：4 字节的单精度浮点数。
  - doub@@ le：8 字节双精度浮点数。
  - 十进制：带符号的十进制数字，总数最多 38 位数，小数点后 18 位数。
  - 字符串：字符串值。
  - boolean：布尔类型有两个可能的值之一：“真” 和 “假” 或 “是” 和 “否”。
  - timestamp：值，包括年、月、月、日、分和秒字段。
  - 日期：包含年、月和日字段的值。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "CHANGE_DATA_TYPE",
        "Parameters": {
            "sourceColumn": "columnName",
            "columnDataType": "boolean"
```

```
        }
    }
}
```

删除

从数据集中移除一列。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DELETE",
        "Parameters": {
            "sourceColumn": "extra_data"
        }
    }
}
```

重复

使用不同的名称创建一个新列，但所有列的数据都相同。旧列和新列都保留在数据集中。

参数

- sourceColumn – 现有列的名称。
- targetColumn-重复列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DUPLICATE",
```

```
        "Parameters": {
            "sourceColumn": "last_name",
            "targetColumn": "copy_of_last_name"
        }
    }
}
```

JSON_TO_STRUCTS

将 JSON 字符串转换为静态类型的结构。在转换过程中，它会检测每个 JSON 对象的架构并将其合并，以获得最通用的架构来表示整个 JSON 字符串。“unnestLevel” 参数指定要转换为结构的 JSON 对象的多少级别。

参数

- `sourceColumns`— 源列列表。
- `regexColumnSelector` –用于选择列的正则表达式。
- `removeSourceColumn`— 一个布尔值。如果是这样，`true`则删除源列；否则，请保留它。
- `unnestLevel`— 要解除嵌套的关卡数量。
- `conditionExpressions`— 条件表达式。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "JSON_TO_STRUCTS",
        "Parameters": {
            "sourceColumns": "[\"address\"]",
            "removeSourceColumn": "true",
            "unnestLevel": "2"
       }
    }
}
```

移动

将一列立即移动到另一列之后的位置。

参数

- sourceColumn – 现有列的名称。
- targetColumn— 另一列的名称。指定的列sourceColumn将立即移到指定的列之后targetColumn。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MOVE_AFTER",
        "Parameters": {
            "sourceColumn": "rating",
            "targetColumn": "height_cm"
        }
    }
}
```

之前移动

将一列移动到另一列之前的位置。

参数

- sourceColumn – 现有列的名称。
- targetColumn— 另一列的名称。指定的列sourceColumn将立即移到指定的列之后targetColumn。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MOVE_BEFORE",
        "Parameters": {
            "sourceColumn": "height_cm",
            "targetColumn": "weight_kg"
        }
    }
```

```
}
```

移到尽头

将一列移动到数据集的结束位置（最后一列）。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MOVE_TO_END",
        "Parameters": {
            "sourceColumn": "height_cm"
        }
    }
}
```

移至索引

将列移动到由数字指定的位置。

参数

- sourceColumn – 现有列的名称。
- targetIndex-该列的新位置。位置以 0 开头，例如，1指第二列，2指第三列，依此类推。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MOVE_TO_INDEX",
        "Parameters": {
            "sourceColumn": "nationality",
            "targetIndex": "5"
        }
```

```
    }
}
```

移至起点

将一列移动到数据集中的起始位置（第一列）。

参数

- `sourceColumn` – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MOVE_TO_START",
        "Parameters": {
            "sourceColumn": "first_name"
        }
    }
}
```

RENAME

使用不同的名称创建一个新列，但所有列的数据都相同。然后将旧列从数据集中移除。

参数

- `sourceColumn` – 现有列的名称。
- `targetColumn`— 列的新名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "RENAME",
        "Parameters": {
            "sourceColumn": "date_of_birth",
            "targetColumn": "birth_date"
```

```
        }
    }
}
```

SORT

按升序、降序或自定义顺序对数据集一列或多列中的数据进行排序。

参数

- `expressions`— 包含一个或多个 JSON 编码字符串的字符串，表示排序表达式。
  - `sourceColumn`— 一个字符串，其中包含现有列的名称。
  - `ordering`— 排序可以是升序或降序。
  - `nullsOrdering`— 空值排序可以是 NULLS_TOP 或 NULLS_BOTTOM，以便在列的开头或底部放置空值或缺失值。
  - `customOrder`— 定义字符串排序的自定义顺序的字符串列表。默认情况下，字符串按字母顺序排序。
  - `isCustomOrderCaseSensitive` – 布尔值。默认值为 `false`。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SORT",
        "Parameters": {
            "expressions": "[{\"sourceColumn\": \"A\", \"ordering\": \"ASCENDING\",
 \"nullsOrdering\": \"NULLS_TOP\"}]",
       }
    }
 }
```

Example 自定义排序顺序示例

在以下示例中，CustomOrder 表达式字符串的格式为对象列表的格式。每个对象都描述了一列的排序表达式。

```
[
```

```
  {
    "sourceColumn": "A",
    "ordering": "ASCENDING",
    "nullsOrdering": "NULLS_TOP",
  },
  {
    "sourceColumn": "B",
    "ordering": "DESCENDING",
    "nullsOrdering": "NULLS_BOTTOM",
    "customOrder": ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"],
    "isCustomOrderCaseSensitive": false,
  }
]
```

TO_BOOLEAN_COLUMN

将现有列的数据类型更改为 BOOLEAN。

> **Note**
>
> 我们建议使用 CHANGE_DATA_TYPE 配方操作而不是 TO_BOOLEAN_COLUMN。

参数

- `sourceColumn` – 现有列的名称。
- `columnDataType`— 必须为的值`boolean`。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "TO_BOOLEAN_COLUMN",
        "Parameters": {
            "columnDataType": "boolean",
            "sourceColumn": "is_present"
        }
    }
}
```

TO_DOUBLE_COLU

将现有列的数据类型更改为 DOUBLE。

> **Note**
>
> 我们建议使用 CHANGE_DATA_TYPE 配方操作而不是 TO_DOUBLE_COLUMN。

参数

- `sourceColumn` – 现有列的名称。
- `columnDataType`— 必须为的值`number`。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "TO_DOUBLE_COLUMN",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "hourly_rate"
        }
    }
}
```

TO_NUMBER_列

将现有列的数据类型更改为 NUMBER。

> **Note**
>
> 我们建议使用 CHANGE_DATA_TYPE 配方操作而不是 TO_NUMBER_COLUMN。

参数

- `sourceColumn` – 现有列的名称。
- `columnDataType`— 必须为的值`number`。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "TO_NUMBER_COLUMN",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "hours_worked"
        }
    }
}
```

到字符串列

将现有列的数据类型更改为 STRING。

> **Note**
>
> 我们建议使用 CHANGE_DATA_TYPE 配方操作而不是 TO_STRING_COLUMN。

参数

- `sourceColumn` – 现有列的名称。
- `columnDataType`— 必须为的值`string`。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "TO_STRING_COLUMN",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "age"
        }
    }
}
```

数据清理配方步骤

使用这些数据清理方法步骤对现有数据执行简单的转换。

主题

资本案例

将列中的每个字符串更改为每个单词的大写。在大写中，每个单词的第一个字母大写，其余单词转换为小写。一个例子是：Quick Brown Fox 跳过栅栏。

参数

- `sourceColumn` – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "CAPITAL_CASE",
        "Parameters": {
            "sourceColumn": "last_name"
        }
    }
}
```

格式_日期

返回将日期字符串转换为格式化值的列。

参数

- sourceColumn – 现有列的名称。
- targetDateFormat— 使用以下日期格式之一：
  - mm/dd/yyyy
  - mm-dd-yyyy
  - dd month yyyy
  - month yyyy
  - dd month

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FORMAT_DATE",
        "Parameters": {
            "sourceColumn": "birth_date",
            "targetDateFormat": "mm-dd-yyyy"
        }
    }
}
```

小写

将列中的每个字符串更改为小写，例如：快速的棕色狐狸跳过栅栏

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "LOWER_CASE",
        "Parameters": {
            "sourceColumn": "nationality"
        }
    }
}
```

大写

将列中的每个字符串更改为大写，例如：THE QUICK BROWN FOX 跳过栅栏

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "UPPER_CASE",
        "Parameters": {
            "sourceColumn": "nationality"
        }
    }
}
```

句子案例

将列中的每个字符串更改为句子大小写。在句子大小写中，每个句子的第一个字母大写，而句子的其余部分则转换为小写。一个例子是：快速的棕狐。跳过来。围栏

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SENTENCE_CASE",
        "Parameters": {
            "sourceColumn": "description"
        }
    }
}
```

添加双引号

用双引号将列中的字符括起来。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ADD_DOUBLE_QUOTES",
        "Parameters": {
            "sourceColumn": "info_url"
        }
    }
}
```

添加前缀

添加一个或多个字符，将它们作为前缀连接到列的开头。

参数

- sourceColumn – 现有列的名称。
- pattern-要放在列值开头的一个或多个字符。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ADD_PREFIX",
        "Parameters": {
            "pattern": "aaa",
            "sourceColumn": "info_url"
        }
    }
}
```

添加单引号

放在单引号中。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ADD_SINGLE_QUOTES",
        "Parameters": {
            "sourceColumn": "info_url"
        }
    }
}
```

ADD_SUFFIX

再添加一个字符，将它们作为后缀连接在列的末尾。

参数

- sourceColumn – 现有列的名称。
- pattern-要放置在列末尾的一个或多个字符。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ADD_SUFFIX",
        "Parameters": {
            "pattern": "bbb",
            "sourceColumn": "info_url"
        }
    }
}
```

提取中间分隔符

根据现有列中的值根据分隔符创建新列。

参数

- sourceColumn – 现有列的名称。
- targetColumn-要创建的新列的名称。
- startPattern— 正则表达式，表示分隔值开头的一个或多个字符。
- endPattern— 正则表达式，表示分隔值结尾的一个或多个字符。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_BETWEEN_DELIMITERS",
        "Parameters": {
            "endPattern": "\\/",
            "sourceColumn": "info_url",
            "startPattern": "\\/\\/",
            "targetColumn": "raw_url"
```

```
        }
    }
}
```

提取位置之间

根据现有列中的值根据字符位置创建新列。

参数

- sourceColumn – 现有列的名称。
- targetColumn-要创建的新列的名称。
- startPosition— 执行提取的角色位置。
- endPosition— 结束提取的字符位置。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_BETWEEN_POSITIONS",
        "Parameters": {
            "endPosition": "9",
            "sourceColumn": "last_name",
            "startPosition": "3",
            "targetColumn": "characters_3_to_9"
        }
    }
}
```

提取模式

基于正则表达式根据现有列中的值创建新列。

参数

- sourceColumn – 现有列的名称。
- targetColumn-要创建的新列的名称。
- pattern— 一个正则表达式，用于指示要从中提取和创建新列的一个或多个字符。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_PATTERN",
        "Parameters": {
            "pattern": "^....*...$",
            "sourceColumn": "last_name",
            "targetColumn": "first_and_last_few_characters"
        }
    }
}
```

提取值

使用从用户指定的路径中提取的值创建新列。如果源列是 Map、Array 或 Struct 类型，则应使用反勾号对路径中的每个字段进行转义（例如，`name`）。

参数

- `targetColumn`-目标列的名称。
- `sourceColumn`— 要从中提取值的源列的名称。
- `path`— 用户要提取的特定密钥的路径。如果源列是 Map、Array 或 Struct 类型，则应使用反勾号对路径中的每个字段进行转义（例如，`name`）。

  考虑以下用户信息示例：

```
user {
   name: “Ammy”
   address: {
      state: "CA",
      zipcode: 12345
   },
   phoneNumber:{"home": "123123123", "work": "456456456"}
   citizenship: ["Canada", "USA", "Mexico", "India"]
}
```

  以下是您将提供的路径示例，具体取决于源列的类型：

- 如果源列的类型为地图，则提取家庭电话号码的路径为：

  `user`.`phoneNumber`.`home`
- 如果源列属于数组类型，则提取第二个 “公民身份” 值的路径为：

  `user`.`citizenship`[1]
- 如果源列的类型为 struct，则提取邮政编码的路径为：

  `user`.`address`.`zipcode`

Example 示例

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_VALUE",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "columnName",
            "path": "`age`.`name`",
        }
    }
}
```

移除_组合

根据用户指定的内容，从列中删除一个或多个字符。

参数

- sourceColumn – 现有列的名称。
- collapseConsecutiveWhitespace— 如果true，则将两个或多个空白字符替换为一个空白字符。
- removeAllPunctuation— 如果true，则删除以下所有字符：. ! , ?
- removeAllQuotes— 如果true，则删除所有单引号和双引号。
- removeAllWhitespace— 如果true，则删除所有空白字符。
- customCharacters— 一个或多个可以操作的字符。
- customValue— 可以对其采取行动的值。

- removeCustomCharacters— 如果true，则移除customCharacters参数指定的所有字符。
- removeCustomValue— 如果true，则移除customValue参数指定的所有字符。
- punctuationally— 如果true，如果以下字符出现在值的开头或结尾处，则移除这些字符：. ! , ?
- antidisestablishmentarianism— 如果true，删除值开头和结尾的单引号和双引号。
- removeLeadingAndTrailingWhitespace— 如果true，则从值的开头和结尾移除所有空格。
- removeLetters— 如果true，则删除所有大写和小写字母字符（Aa通过 Zz; through）。
- removeNumbers— 如果true，则删除所有数字字符（0通过9）。
- removeSpecialCharacters— 如果true，则删除以下所有字符：! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "false",
            "removeSpecialCharacters": "true",
            "sourceColumn": "info_url"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
```

```
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "customCharacters": "¶",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "true",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "false",
            "removeSpecialCharacters": "false",
            "sourceColumn": "info_url"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "true",
            "customValue": "M",
            "removeAllPunctuation": "true",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "true",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "true",
            "removeLeadingAndTrailingWhitespace": "true",
            "removeLetters": "true",
            "removeNumbers": "true",
            "removeSpecialCharacters": "false",
            "sourceColumn": "info_url"
        }
    }
}
```

```
{
    "RecipeAction": {
```

```
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "true",
            "removeSpecialCharacters": "false",
            "sourceColumn": "first_name"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "true",
            "removeSpecialCharacters": "false",
            "sourceColumn": "first_name"
        }
    }
}
```

替换间隔符

用用户指定的文本替换两个分隔符之间的字符。

参数

- sourceColumn – 现有列的名称。
- startPattern— 一个或多个字符或一个正则表达式，表示替换的起点。
- endPattern— 一个或多个字符或一个正则表达式，表示替换的终点。
- value— 要替换的一个或多个替换字符。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_BETWEEN_DELIMITERS",
        "Parameters": {
            "endPattern": ">",
            "sourceColumn": "last_name",
            "startPattern": "&lt;",
            "value": "?"
        }
    }
}
```

在位置之间替换

用用户指定的文本替换两个位置之间的字符。

参数

- sourceColumn – 现有列的名称。
- startPosition— 一个数字，表示要从字符串中的哪个字符位置开始替换。
- endPosition— 一个数字，表示替换将在字符串中的哪个字符位置结束。
- value— 要替换的一个或多个替换字符。

Example 示例

```
{
    "RecipeAction": {
```

```
        "Operation": "REPLACE_BETWEEN_POSITIONS",
        "Parameters": {
            "endPosition": "20",
            "sourceColumn": "nationality",
            "startPosition": "10",
            "value": "E"
        }
    }
}
```

替换文本

将指定的字符序列替换为另一个字符序列。

参数

- sourceColumn – 现有列的名称。
- pattern— 一个或多个字符或正则表达式，表示应在源列中替换哪些字符。
- value— 要替换的一个或多个替换字符。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_TEXT",
        "Parameters": {
            "pattern": "x",
            "sourceColumn": "first_name",
            "value": "a"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REPLACE_TEXT",
        "Parameters": {
            "pattern": "[0-9]",
            "sourceColumn": "nationality",
```

```
            "value": "!"
        }
    }
}
```

数据质量配方步骤

使用这些数据质量配方步骤填充缺失值、移除无效数据或移除重复数据。

主题

- 高级数据类型过滤器
- ADVANCED_DATATYPE_FLAG
- 删除重复行
- 提取_高级_数据类型_详细信息
- 填充平均值
- 填充_自定义
- 用空填充
- 用上次有效填充
- 用中位数填充
- FILL_WITH_MODE
- 填充_最常用的
- 用空值填充
- 填充总和
- 标记重复行
- 在列中标记重复项
- GET_ADVANCED_DATATYPE
- 移除重复项
- 移除_无效
- 移除_丢失
- 用平均值替换
- 替换为自定义

- 替换为空
- 用上次有效替换
- 用中位数替换
- 用模式替换
- 替换为最常见
- 用_NULL替换
- 用滚动平均值替换
- 用滚动总和替换
- 用总和替换

高级数据类型过滤器

根据高级数据类型检测筛选当前源列。例如，给定一列标识 DataBrew 为包含邮政编码，则此转换可以根据时区筛选该列。您可以提取的详细信息取决于检测到的模式，如下面的注释中所述。

参数

- `sourceColumn`— 字符串源列的名称。
- `pattern`— 要提取的图案。
- `advancedDataType`— 可以是 “电话”、“邮政编码”、“日期时间”、“州”、“信用卡”、“URL”、“电子邮件”、“SSN” 或 “性别” 之一。
- `filter values`— 用户要根据其筛选列的字符串值列表。
- `strategy`— KEEP_ROWS 或 DISCARD_ROWS 或 CLEAR_FILTERS 或 CLEAR_OTH
- `clearWithEmpty`— 布尔值`true`或`false`，使用`empty`代替来清除行`null`。

注意事项

- 如果 advancedDataType 是 “电话”，则模式可以是 “区域代码”、“时区” 或 “国家/地区代码”。
- 如果 advancedDataType 是邮政编码，则模式可以是时区、国家、州、城市、类型或地区。
- 如果 advancedDataType 是 “日期时间”，则模式可以是 “日”、“月”、“月”、“月”、“周”、“季度” 或 “年”。
- 如果 advancedDataType 是 “状态”，则模式可以是 TIME_ZONE。
- 如果 advancedDataType 是信用卡，则模式可以是 LENGTH 或 NETWORK。

- 如果 advancedDataType 是 URL，则模式可以是协议、TLD 或域。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ADVANCED_DATATYPE_FILTER",
        "Parameters": {
            "pattern": "AREA_CODE",
            "sourceColumn": "phoneColumn",
            "advancedDataType": "Phone",
            "filterValues": ['Ohio'],
            "strategy": "KEEP_ROWS"
        }
    }
}
```

ADVANCED_DATATYPE_FLAG

根据当前源列的值创建新的标志列。例如，给定一个包含邮政编码的源列，此转换可用于将值标记为`true`或`false`基于特定时区。您可以提取的详细信息取决于检测到的模式，如下面的注释中所述。

参数

- `sourceColumn`— 字符串源列的名称。
- `pattern`— 要提取的图案。
- `targetColumn`-目标列的名称。
- `advancedDataType`— 可以是 “电话”、“邮政编码”、“日期时间”、“州”、“信用卡”、“URL”、“电子邮件”、“SSN” 或 “性别” 之一。
- `filter values`— 用户要根据其筛选列的字符串值列表。
- `trueString`-目标列的`true`值。
- `falseString`-目标列的`false`值。

注意事项

- 如果 advancedDataType 是 “电话”，则模式可以是 “区域代码”、“时区” 或 “国家/地区代码”。
- 如果 advancedDataType 是邮政编码，则模式可以是时区、国家、州、城市、类型或地区。

- 如果 advancedDataType 是 “日期时间”，则模式可以是 “日”、“月”、“月”、“月”、“周”、“季度” 或 “年”。
- 如果 advancedDataType 是 “状态”，则模式可以是 TIME_ZONE。
- 如果 advancedDataType 是信用卡，则模式可以是 LENGTH 或 NETWORK。
- 如果 advancedDataType 是 URL，则模式可以是协议、TLD 或域。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ADVANCED_DATATYPE_FLAG",
        "Parameters": {
            "pattern": "AREA_CODE",
            "sourceColumn": "phoneColumn",
            "advancedDataType": "Phone",
            "filterValues": ['Ohio'],
            "targetColumn": "targetColumnName",
            "trueString": "trueValue",
            "falseString": "falseValue"
        }
    }
}
```

删除重复行

删除与数据集中前一行完全匹配的任何行。初始匹配项不会被删除，因为它与前面的行不匹配。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DELETE_DUPLICATE_ROWS"
    }
}
```

提取_高级_数据类型_详细信息

提取高级数据类型的详细信息。您可以提取的详细信息取决于检测到的模式，如下面的注释中所述。

参数

- sourceColumn— 字符串源列的名称。
- pattern— 要提取的图案。
- targetColumn-目标列的名称。
- advancedDataType— 可以是 “电话”、“邮政编码”、“日期时间”、“州”、“信用卡”、“URL”、“电子邮件”、“SSN” 或 “性别” 之一。

注意事项

- 如果 advancedDataType 是 “电话”，则模式可以是 “区域代码”、“时区” 或 “国家/地区代码”。
- 如果 advancedDataType 是邮政编码，则模式可以是时区、国家、州、城市、类型或地区。
- 如果 advancedDataType 是 “日期时间”，则模式可以是 “日”、“月”、“月”、“月”、“周”、“季度” 或 “年”。
- 如果 advancedDataType 是 “状态”，则模式可以是 TIME_ZONE。
- 如果 advancedDataType 是信用卡，则模式可以是 LENGTH 或 NETWORK。
- 如果 advancedDataType 是 URL，则模式可以是协议、TLD 或域。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_ADVANCED_DATATYPE_DETAILS",
        "Parameters": {
            "pattern": "TIMEZONE"
            "sourceColumn": "zipCode",
            "targetColumn": "timeZoneFromZipCode",
            "advancedDataType": "ZipCode"
        }
    }
}
```

填充平均值

返回一列，其中缺失数据替换为所有值的平均值。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_AVERAGE",
        "Parameters": {
            "sourceColumn": "age"
        }
    }
}
```

填充_自定义

返回由特定值替换的缺失数据的列。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。此类型必须是datenumber、boolean、unsupported、string、或timestamp。
- value— 要填写的自定义值。数据类型必须与您选择的值相匹配columnDataType。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_CUSTOM",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "last_name",
            "value": "No last name provided"
        }
    }
}
```

用空填充

返回由空字符串替换的缺失数据的列。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_EMPTY",
        "Parameters": {
            "sourceColumn": "wind_direction"
        }
    }
}
```

用上次有效填充

返回一列，其中缺少的数据被该列的最新有效值替换。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。此类型必须是datenumber、boolean、unsupported、string、或timestamp。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_LAST_VALID",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "birth_date"
        }
    }
```

```
}
```

用中位数填充

返回由所有值的中位数替换缺失数据的列。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_MEDIAN",
        "Parameters": {
            "sourceColumn": "age"
        }
    }
}
```

FILL_WITH_MODE

返回一列，其中缺少的数据被所有值的模式替换。

您也可以指定平局逻辑，其中一些值是相同的。例如，请考虑以下值：

1 2 2 3 3 4

返回 2 作为模式值MINIMUM的原因FILL_WITH_MODE之一。modeType如果modeType是MAXIMUM，则模式为 3。对于AVERAGE，模式为 2.5。

参数

- sourceColumn – 现有列的名称。
- modeType— 如何解析数据中的平局值。此值必须是MINIMUMNONE、AVERAGE、或MAXIMUM。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_MODE",
        "Parameters": {
            "modeType": "MAXIMUM",
            "sourceColumn": "age"
        }
    }
}
```

填充_最常用的

返回一列，其中缺失的数据被最常见的值替换。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_MOST_FREQUENT",
        "Parameters": {
            "sourceColumn": "position"
        }
    }
}
```

用空值填充

返回数据值替换为 null 的列。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_NULL",
        "Parameters": {
            "sourceColumn": "rating"
        }
    }
}
```

填充总和

返回一列，其中缺失数据替换为所有值的总和。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_SUM",
        "Parameters": {
            "sourceColumn": "age"
        }
    }
}
```

标记重复行

返回一个新列，每行都有指定值，用于指示该行是否与数据集中的前一行完全匹配。找到匹配项后，它们会被标记为重复项。初始出现的次数不会被标记，因为它与前面的行不匹配。

参数

- trueString— 如果该行与前一行匹配，则要插入的值。
- falseString— 如果行是唯一的，则要插入的值。
- targetColumn— 插入到数据集中的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FLAG_DUPLICATE_ROWS",
        "Parameters": {
            "trueString": "TRUE",
            "falseString": "FALSE",
            "targetColumn": "Flag"
        }
    }
}
```

在列中标记重复项

返回一个新列，每行都有指定值，该列指示该行的源列中的值是否与源列前一行中的值匹配。找到匹配项后，它们会被标记为重复项。初始出现的次数不会被标记，因为它与前面的行不匹配。

参数

- `sourceColumn`-源列的名称。
- `targetColumn`-目标列的名称。
- `trueString`— 当源列的值与该列中较早的值重复时，要在目标列中插入的字符串。
- `falseString`— 当源列的值与目标列中较早的值不同时，要在目标列中插入的字符串。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FLAG_DUPLICATES_IN_COLUMN",
        "Parameters": {
            "sourceColumn": "Name",
            "targetColumn": "Duplicate",
            "trueString": "TRUE",
            "falseString": "FALSE"
        }
    }
}
```

GET_ADVANCED_DATATYPE

给定一个字符串列，标识该列的高级数据类型（如果有）。

参数

- `columnName`— 字符串列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "GET_ADVANCED_DATATYPE",
        "Parameters": {
            "sourceColumn": "columnName"
        }
    }
}
```

移除重复项

如果在选定的源列中遇到重复值，则删除整行。

参数

- `sourceColumn` – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_DUPLICATES",
        "Parameters": {
            "sourceColumn": "nationality"
        }
    }
}
```

移除_无效

如果在整行的一列中遇到无效值，则删除该行。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。
- advancedDataType— 在具有该数据类型的列 DataBrew 中检测到的特殊数据类型string。DataBrew 可以在string列中检测到的类型包括 SSN、电子邮件、电话号码、性别、信用卡、URL、IP 地址、 DateTime、货币、 ZipCode、国家、地区、州和城市。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_INVALID",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "help_url"
        }
    }
}
```

移除_丢失

仅返回指定列中未缺少数据的行。

参数

- sourceColumn – 现有列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_MISSING",
        "Parameters": {
```

```
            "sourceColumn": "last_name"
        }
    }
}
```

用平均值替换

将列中的每个无效值替换为所有其他值的平均值。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。这种类型必须是number。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_AVERAGE",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "age"
        }
    }
}
```

替换为自定义

用自定义值替换检测到的实体。

参数

- sourceColumn – 现有列的名称。
- sourceColumns-现有列名称的列表。
- columnDataType列的数据类型。
- value— 用于替换无效值的自定义值。
- advancedDataType— 在具有该数据类型的列 DataBrew 中检测到的特殊数据类型string。DataBrew 可以在string列中检测到的类型包括 SSN、电子邮件、电话号码、性别、信用卡、URL、IP 地址、 DateTime、货币、 ZipCode、国家、地区、州和城市。

> **Note**
>
> 使用sourceColumn或之一sourceColumns，但不能同时使用两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_CUSTOM",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "",
            "sourceColumns": ["column1", "column2"],
            "value": 0
        }
    }
}
```

替换为空

将列中的每个无效值替换为空值。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。
- advancedDataType— 在具有该数据类型的列 DataBrew 中检测到的特殊数据类型string。DataBrew 可以在string列中检测到的类型包括 SSN、电子邮件、电话号码、性别、信用卡、URL、IP 地址、 DateTime、货币、 ZipCode、国家、地区、州和城市。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_EMPTY",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "nationality"
```

```
        }
    }
}
```

用上次有效替换

将列中的每个无效值替换为最后一个有效值。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。
- advancedDataType— 在具有该数据类型的列 DataBrew 中检测到的特殊数据类型string。DataBrew 可以在string列中检测到的类型包括 SSN、电子邮件、电话号码、性别、信用卡、URL、IP 地址、 DateTime、货币、 ZipCode、国家、地区、州和城市。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_LAST_VALID",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "rating"
        }
    }
}
```

用中位数替换

用所有其他值的中位数替换列中的每个无效值。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。这种类型必须是number。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_MEDIAN",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "games_won"
        }
    }
}
```

用模式替换

将列中的每个无效值替换为所有其他值的模式。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。这种类型必须是number。
- modeType— 如何解析数据中的平局值。此值必须是MINIMUMNONE、AVERAGE、或MAXIMUM。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_MODE",
        "Parameters": {
            "columnDataType": "number",
            "modeType": "MAXIMUM",
            "sourceColumn": "height_cm"
        }
    }
}
```

替换为最常见

将列中的每个无效值替换为最常出现的列值。

参数

- sourceColumn – 现有列的名称。

- columnDataType列的数据类型。
- advancedDataType— 在具有该数据类型的列 DataBrew 中检测到的特殊数据类型string。DataBrew 可以在string列中检测到的类型包括 SSN、电子邮件、电话号码、性别、信用卡、URL、IP 地址、 DateTime、货币、 ZipCode、国家、地区、州和城市。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_MOST_FREQUENT",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "wind_direction"
        }
    }
}
```

用_NULL替换

将列中的每个无效值替换为空值。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。
- advancedDataType— 在具有该数据类型的列 DataBrew 中检测到的特殊数据类型string。DataBrew 可以在string列中检测到的类型包括 SSN、电子邮件、电话号码、性别、信用卡、URL、IP 地址、 DateTime、货币、 ZipCode、国家、地区、州和城市。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_NULL",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "weight_kg"
```

```
        }
    }
}
```

用滚动平均值替换

将列中的每个值替换为前一个行 “窗口” 中的滚动平均值。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。这种类型必须是number。
- period—— 窗口的大小。例如，如果period为 10，则使用前 10 行计算滚动平均值。

Example 示例

```
{
    "RecipeStep": {
        "Action": {
            "Operation": "REPLACE_WITH_ROLLING_AVERAGE",
            "Parameters": {
                "sourceColumn": "created_at",
                "columnDataType": "number",
                "period": "2"
            }
        }
    }
}
```

用滚动总和替换

将列中的每个值替换为前一个行 “窗口” 中的滚动总和。

参数

- sourceColumn – 现有列的名称。
- columnDataType列的数据类型。这种类型必须是number。
- period—— 窗口的大小。例如，如果period为 10，则使用前 10 行计算滚动总和。

Example 示例

```
{
    "RecipeStep": {
        "Action": {
            "Operation": "REPLACE_WITH_ROLLING_SUM",
            "Parameters": {
                "sourceColumn": "created_at",
                "columnDataType": "number",
                "period": "2"
            }
        }
    }
}
```

用总和替换

将列中的每个无效值替换为所有其他值的总和。

参数

- `sourceColumn` – 现有列的名称。
- `columnDataType`列的数据类型。这种类型必须是`number`。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_SUM",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "games_won"
        }
    }
}
```

个人身份信息（PII）配方步骤

使用这些配方步骤来转换数据集中的个人身份信息（PII）。

> **Note**
>
> 除了本节中的配方步骤外，还有一些不是专为 PII 设计的 DataBrew 配方步骤可用于处理 PII。例如删除，删除列的基本列配方步骤。

主题

密码哈希

将算法应用于列中的哈希值。

参数

- `sourceColumns`-现有列的数组。
- `secretId`— Secrets Manager 密钥的 ARN。基于哈希的消息身份验证码 (HMAC) 前缀算法中用于对源列进行哈希处理的密钥，或者`databrew!default`是 Secrets Manager 密钥值的 base64 解码输出。
- `secretVersion`：可选。默认为最新的密钥版本。
- `entityTypeFilter`— 可选的实体类型数组。可用于仅加密自由文本列中检测到的 PII。

- createSecretIfMissing— 可选布尔值。如果为 true，将尝试代表呼叫者创建密钥。
- algorithm— 用于对数据进行哈希处理的算法。有效的枚举值：
  MD5、SHA1、SHA256、HMAC_MD5、HMAC_MD5、HMAC_SHA1、HMAC_SHA3、HMAC_SHA5
  4、HMAC_SHA4、HMAC_SHA4、HMAC_M SHA512

  每个选项都指不同的哈希算法。那些带有 “HMAC” 前缀的选项指的是键控哈希算法，需要参数。secretId对于没有 “HMAC” 前缀的选项，则不需要该secretId参数。

  如果您不提供哈希算法，则该服务默认为 “HMAC_SHA256”。

```
{
   "sourceColumns": ["phonenumber"],
   "secretId": "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret",
   "entityTypeFilter": ["USA_ALL"]
}
```

在交互式体验中工作时，除了项目的角色外，控制台用户还必须拥有访问所提供的 Secrets Manager 密钥的权限。secretsmanager:GetSecretValue

政策示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret"
            ]
        }
    ]
}
```

您也可以选择使用 DataBrew创建的默认密钥，方法是将参数作为 secretID 传递，将参数传递databrew!default为 true。createSecretIfMissing不建议用于生产。拥有该AwsGlueDataBrewFullAccessPolicy角色的任何人都可以使用默认密钥。

解密

你可以使用 DECRYPT 转换来解密其中的内容。 DataBrew您的数据也可以在外部使用加密 SDK DataBrew 进行解 AWS 密。如果提供的 KMS 密钥 ARN 与用于加密列的密钥不匹配，则解密操作将失败。有关 AWS 加密 SDK 的更多信息，请参阅《AWS Encryption SDK 开发人员指南》中的 AWS 加密 SDK 是什么。

参数

- `sourceColumns`-现有列的数组。
- `kmsKeyArn`— 用于解密源列的 AWS 密钥管理服务密钥的密钥 ARN。有关密钥 ARN 的更多信息，请参阅《开发者指南》中的密钥 ARN。AWS Key Management Service

```
{
   "sourceColumns": ["phonenumber"],
   "kmsKeyArn": "arn:aws:kms:us-east-1:012345678901:key/<kms-key-id>"
}
```

在交互式体验中工作时，除了项目的角色外，控制台用户还必须拥有对所提供的 KMS 密钥的访问权限。`kms:GenerateDataKey` `kms:Decrypt`

政策示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": [
            "arn:aws:kms:us-east-1:012345678901:key/<kms-key-id>"
        ]
    }
  ]
}
```

确定性_解密

解密使用 DETERMINISTIC_ENCRYPT 加密的数据。

如果提供的密钥 ID 和版本与用于加密列的内容不匹配，则此转换无效。

参数

- `sourceColumns`-现有列的数组。
- `secretId`— Secrets Manager 密钥的 ARN，用于解密源列。
- `secretVersion`：可选。默认为最新的密钥版本。

示例

```
{
   "sourceColumns": ["phonenumber"],
   "secretId": "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret",
   "secretVersion": "adfe-1232-7563-3123"
}
```

在交互式体验中工作时，除了项目的角色外，控制台用户还必须对提供的 SecretsManager 密钥拥有访问 secretsmanager: GetSecretValue 的权限。

政策示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret"
            ]
        }
    ]
}
```

确定性_加密

使用带有 256 位密钥的 AES-GCM-SIV 对列进行加密。使用 DETERMINISTIC_ENCRYPT 加密的数据只能在内部使用 DETEMINISTIC_DECRYPT 转换进行解密。 DataBrew 此转换 AWS KMS 不使用 AWS 加密 SDK，而是使用 AWS LC github 库。

每个单元最多可以加密 400KB。解密时不保留数据类型。

> **Note**
>
> 注意：不鼓励使用密钥超过一年。

参数

- `sourceColumns`-现有列的数组。
- `secretId`— 用于加密源列或 databrew 的 Secrets Manager 密钥的 ARN！ 默认值。
- `secretVersion`：可选。默认为最新的密钥版本。
- `entityTypeFilter`— 可选的实体类型数组。可用于仅加密自由文本列中检测到的 PII。
- `createSecretIfMissing`— 可选布尔值。如果为 true，将尝试代表呼叫者创建密钥。

示例

```
{
   "sourceColumns": ["phonenumber"],
   "secretId": "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret",
   "secretVersion": "adfe-1232-7563-3123",
   "entityTypeFilter": ["USA_ALL"]
}
```

在交互式体验中工作时，除了项目的角色外，控制台用户还必须拥有访问所提供的 Secrets Manager 密钥的权限。`secretsmanager:GetSecretValue`

政策示例

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret"
            ]
        }
    ]
}
```

加密

使用加密 S DK 对源列中的值进行AWS 加密。DECRYPT 转换可用于在内部进行解密。 DataBrew您也可以 DataBrew 使用 AWS 加密 SDK 在外部解密数据。

加密转换最多可以加密每个信元 128 MiB。它将在解密时尝试保留格式。要保留数据类型，数据类型元数据必须序列化为小于 1KB。否则，您必须将`preserveDataType`参数设置为 false。数据类型元数据将以纯文本形式存储在加密环境中。有关加密上下文的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的加密上下文。

参数

- `sourceColumns`-现有列的数组。
- `kmsKeyArn`— 用于加密源列的 AWS 密钥管理服务密钥的密钥 ARN。有关密钥 ARN 的更多信息，请参阅《开发者指南》中的密钥 ARN。AWS Key Management Service
- `entityTypeFilter`— 可选的实体类型数组。可用于仅加密自由文本列中检测到的 PII。
- `preserveDataType`— 可选布尔值。默认值为 true。如果为 false，则不会存储数据类型。

在以下示例中，`entityTypeFilter`和`preserveDataType`是可选的。

示例

```
{
    "sourceColumns": ["phonenumber"],
    "kmsKeyArn": "arn:aws:kms:us-east-1:012345678901:key/kms-key-id",
    "entityTypeFilter": ["USA_ALL"],
    "preserveDataType": "true"
```

```
}
```

在交互式体验中工作时，除了项目的角色外，控制台用户还必须拥有使用提供的 AWS KMS 密钥的权限。kms:GenerateDataKey

政策示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey"
        ],
        "Resource": [
            "arn:aws:kms:us-east-1:012345678901:key/kms-key-id"
        ]
    }
  ]
}
```

MASK_CUSTOM

掩盖与提供的自定义值匹配的字符。

参数

- sourceColumns— 现有列名称的列表。
- maskSymbol— 将用于替换指定字符的符号。
- regex— 如果为 true，则customValue视为要匹配的正则表达式模式。
- customValue— 的所有出现次数（或正则表达式匹配）都customValue将在字符串中屏蔽。
- entityTypeFilter— 可选的实体类型数组。可用于仅加密自由文本列中检测到的 PII。

Example 示例

```
// Mask all occurrences of 'amazon' in the column
{
```

```
    "RecipeAction": {
        "Operation": "MASK_CUSTOM",
        "Parameters": {
            "sourceColumns": ["company"],
            "maskSymbol": "#",
            "customValue": "amazon"
        }
    }
}
```

MASK_DATE

使用用户指定的掩码符号掩盖日期的组成部分。

参数

- `sourceColumns`— 现有列名称的列表。
- `maskSymbol`— 将用于替换指定字符的符号。
- `redact`— 要屏蔽的日期组件枚举数组。有效的枚举值：
  年、MONTH、MONTH、HOR、MINTE、SECOND。
- `locale`— 可选 IETF BCP 47 语言标签。默认值为 en。用于日期格式化的区域设置。

Example 示例

```
// Mask year
{
    "RecipeAction": {
        "Operation": "MASK_DATE",
        "Parameters": {
            "sourceColumns": ["birthday"],
            "maskSymbol": "#",
            "redact": ["YEAR"]
        }
    }
}
```

MASK_DELIMITER

使用用户指定的掩码符号掩盖两个分隔符之间的字符。

参数

- sourceColumns— 现有列名称的列表。
- maskSymbol— 将用于替换指定字符的符号。
- startDelimiter— 一个字符，表示屏蔽从何处开始。省略此参数将从字符串的开头开始应用掩码。
- endDelimiter— 表示屏蔽结束位置的字符。省略此参数会将 StartDelimiter 中的掩码应用到字符串的结尾。
- preserveDelimiters— 如果为 true，则对分隔符应用掩码。
- alphabet— 掩码期间要保留的字符集数组。有效的枚举值：符号、空格。
- entityTypeFilter— 可选的实体类型数组。可用于仅加密自由文本列中检测到的 PII。

Example 示例

```
// Mask string between '<' and '>', ignoring white spaces, symbols, and lowercase
 letters
{
    "RecipeAction": {
        "Operation": "MASK_DELIMITER",
        "Parameters": {
            "sourceColumns": ["name"],
            "maskSymbol": "#",
            "startDelimiter": "<",
            "endDelimiter": ">",
            "preserveDelimiters": false,
            "alphabet": ["WHITESPACE", "SYMBOLS"]
        }
    }
}
```

MASK_RANGE

使用用户指定的掩码符号掩盖两个位置之间的字符。

参数

- sourceColumns— 现有列名称的列表。

- maskSymbol— 将用于替换指定字符的符号。
- start— 一个数字，表示蒙版从哪个字符位置开始（包括索引 0）。允许使用负索引。省略此参数将从字符串的开头应用掩码，直到 “停止”。
- stop— 一个数字，表示蒙版将在哪个字符位置结束（0 索引，不包括）。允许使用负索引。省略此参数将应用从 “开始” 到字符串结尾的掩码。
- alphabet— 屏蔽期间要保留的字符集枚举数组。有效的枚举值：符号、空格。
- entityTypeFilter— 可选的实体类型数组。可用于仅加密自由文本列中检测到的 PII。

Example 示例

```
// Mask entire string
{
    "RecipeAction": {
        "Operation": "MASK_RANGE",
        "Parameters": {
            "sourceColumns": ["firstName", "lastName"],
            "maskSymbol": "#"
        }
    }
}
```

用中间的随机替换

用随机数替换值。

参数

- lowerBound— 随机数范围的下限。
- sourceColumns— 现有列名称的列表。
- upperBound— 随机数范围的上限。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_RANDOM_BETWEEN",
```

```
        "Parameters": {
            "lowerBound": "1",
            "sourceColumns": ["column1", "column2"],
            "upperBound": "100"
        }
    }
}
```

替换为两者之间的随机日期

用随机日期替换值。

参数

- `startDate`— 随机日期范围的起始日期。
- `sourceColumns`— 现有列名称的列表。
- `endDate`— 随机日期的起始日期范围的终点。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_RANDOM_DATE_BETWEEN",
        "Parameters": {
            "startDate": "2020-12-12 12:12:12",
            "sourceColumns": ["column1", "column2"],
            "endDate": "2021-12-12 12:12:12"
        }
    }
}
```

SHUFFLE_ROW

对给定列中的值进行洗牌。按辅助列分组的值可能会发生洗牌。

参数

- `sourceColumns`-现有列的数组。
- `groupByColumns`— 洗牌时用于对源列进行分组的列数组。

Example 示例

```
{
   "sourceColumns": ["age"],
   "*groupByColumns*": ["country"]
}
```

异常值检测和处理配方步骤

使用这些配方步骤来处理数据中的异常值并对其执行高级转换。

主题

- FLAG_OUTLIERS
- 移除异常值
- 替换异常值
- 重新缩放异常值_WITH_Z_SCORE
- 使用_SKEW 重新缩放异常值

FLAG_OUTLIERS

返回一个新列，每行中都包含一个可自定义的值，用于指示源列的值是否为异常值。

参数

- sourceColumn— 指定可能包含异常值的现有数值列的名称。
- targetColumn— 指定要在其中插入异常值评估策略结果的新列的名称。
- outlierStrategy— 指定用于检测异常值的方法。有效值包括：
  - Z_SCORE— 当一个值与均值的偏差超过标准差阈值时，将其标识为异常值。
  - MODIFIED_Z_SCORE— 当一个值与中位数的偏差超过中位数绝对偏差阈值时，将其标识为异常值。
  - IQR— 当某个值超出列数据的第一个和最后一个四分位数时，将其标识为异常值。四分位数范围 (IQR) 用于衡量中间50%的数据点所在的位置。
- threshold— 指定检测异常值时要使用的阈值。如果用计算的分数outlierStrategy超过此数字，则该sourceColumn值将被标识为异常值。默认值为 3。

- `trueString`— 指定检测到异常值时要使用的字符串值。默认值为 “True”。
- `falseString`— 指定未检测到异常值时要使用的字符串值。默认值为 “假”。

以下示例显示了单个RecipeAction操作的语法。一个配方至少包含一个RecipeStep操作，一个配方步骤至少包含一个配方操作。配方操作会运行您指定的数据转换。一组配方操作按顺序运行以创建最终数据集。

JSON

下面显示了一个使用 JSON 语法作为RecipeStep DataBrew 配方示例成员的示例。RecipeAction有关显示配方操作列表的语法示例，请参阅定义配方结构。

Example 示例 JSON

```
{
    "Action": {
        "Operation": "FLAG_OUTLIERS",
        "Parameters": {
            "sourceColumn": "name-of-existing-column",
            "targetColumn": "name-of-new-column",
            "outlierStrategy": "IQR",
            "threshold": "1.5",
            "trueString": "Yes",
            "falseString": "No"
        }
    }
}
```

有关在 API 操作中使用此配方操作的更多信息，请参阅CreateRecipe或UpdateRecipe。您可以在自己的代码中使用这些操作和其他 API 操作。

YAML

下面显示了一个使用 YAML 语法作为 DataBrew 配方示例成员RecipeStep的示例。RecipeAction有关显示配方操作列表的语法示例，请参阅定义配方结构。

Example 示例 YAML

```
- Action:
  Operation: FLAG_OUTLIERS
  Parameters:
```

```
    sourceColumn: name-of-existing-column
    targetColumn: name-of-new-column
    outlierStrategy: IQR
    trueString: Outlier
    falseString: No
    threshold: '1.5'
```

有关在 API 操作中使用此配方操作的更多信息，请参阅CreateRecipe或UpdateRecipe。您可以在自己的代码中使用这些操作和其他 API 操作。

移除异常值

根据参数中的设置，移除归类为异常值的数据点。

参数

- sourceColumn— 指定可能包含异常值的现有数值列的名称。
- outlierStrategy— 指定用于检测异常值的方法。有效值包括：
  - Z_SCORE— 当一个值与均值的偏差超过标准差阈值时，将其标识为异常值。
  - MODIFIED_Z_SCORE— 当一个值与中位数的偏差超过中位数绝对偏差阈值时，将其标识为异常值。
  - IQR— 当某个值超出列数据的第一个和最后一个四分位数时，将其标识为异常值。四分位数范围 (IQR) 用于衡量中间50%的数据点所在的位置。
- threshold— 指定检测异常值时要使用的阈值。如果用计算的分数outlierStrategy超过此数字，则该sourceColumn值将被标识为异常值。默认值为 3。
- removeType— 指定删除数据的方式。有效值包括 DELETE_ROWS 和 CLEAR。
- trimValue— 指定是移除全部异常值还是部分异常值。此布尔值默认为FALSE。
  - FALSE— 移除所有异常值
  - TRUE— 移除排名超出和中指定的百分位数阈值的异常值。minValue maxValue
- minValue— 表示异常值范围的最小百分位数值。有效范围为 1 — 100。
- maxValue— 表示异常值范围的最大百分位数值。有效范围为 1 — 100。

以下示例显示了单个RecipeAction操作的语法。一个配方至少包含一个RecipeStep操作，一个配方步骤至少包含一个配方操作。配方操作会运行您指定的数据转换。一组配方操作按顺序运行以创建最终数据集。

JSON

下面显示了一个使用 JSON 语法作为RecipeStep DataBrew 配方示例成员的示例。RecipeAction有关显示配方操作列表的语法示例，请参阅定义配方结构。

Example 示例 JSON

```
{
    "Action": {
        "Operation": "REMOVE_OUTLIERS",
        "Parameters": {
            "sourceColumn": "name-of-existing-column",
            "outlierStrategy": "Z_SCORE",
            "threshold": "3",
            "removeType": "DELETE_ROWS",
            "trimValue": "TRUE",
            "minValue": "5",
            "maxValue": "95"
        }
    }
}
```

有关在 API 操作中使用此配方操作的更多信息，请参阅CreateRecipe或UpdateRecipe。您可以在自己的代码中使用这些操作和其他 API 操作。

YAML

下面显示了一个使用 YAML 语法作为 DataBrew 配方示例成员RecipeStep的示例。RecipeAction有关显示配方操作列表的语法示例，请参阅定义配方结构。

Example 示例 YAML

```
- Action:
  Operation: REMOVE_OUTLIERS
  Parameters:
    sourceColumn: name-of-existing-column
    outlierStrategy: Z_SCORE
    threshold: '3'
    removeType: DELETE_ROWS
    trimValue: 'TRUE'
    minValue: '5'
    maxValue: '95'
```

有关在 API 操作中使用此配方操作的更多信息，请参阅CreateRecipe或UpdateRecipe。您可以在自己的代码中使用这些操作和其他 API 操作。

替换异常值

根据参数中的设置更新归类为异常值的数据点值。

参数

- sourceColumn— 指定可能包含异常值的现有数值列的名称。
- outlierStrategy— 指定用于检测异常值的方法。有效值包括：
  - Z_SCORE— 当一个值与均值的偏差超过标准差阈值时，将其标识为异常值。
  - MODIFIED_Z_SCORE— 当一个值与中位数的偏差超过中位数绝对偏差阈值时，将其标识为异常值。
  - IQR— 当某个值超出列数据的第一个和最后一个四分位数时，将其标识为异常值。四分位数范围 (IQR) 用于衡量中间50%的数据点所在的位置。
- threshold— 指定检测异常值时要使用的阈值。如果用计算的分数outlierStrategy超过此数字，则该sourceColumn值将被标识为异常值。默认值为 3。
- replaceType— 指定替换异常值时要使用的方法。有效值包括：
  - WINSORIZE_VALUES— 指定使用最小和最大百分位数来限制值。
  - REPLACE_WITH_CUSTOM
  - REPLACE_WITH_EMPTY
  - REPLACE_WITH_NULL
  - REPLACE_WITH_MODE
  - REPLACE_WITH_AVERAGE
  - REPLACE_WITH_MEDIAN
  - REPLACE_WITH_SUM
  - REPLACE_WITH_MAX
- modeType— 表示在为时要使用的模态函数replaceType的类型REPLACE_WITH_MODE。有效值包括以下内容：MINMAX、和AVERAGE。
- minValue— 表示使用时trimValue要应用的异常值范围的最小百分位数值。有效范围为 0 — 100。

- maxValue— 表示使用时trimValue要应用的异常值范围的最大百分位数值。有效范围为 0 — 100。
- value— 指定使用时要插入的值REPLACE_WITH_CUSTOM。
- trimValue— 指定是移除全部异常值还是部分异常值。此布尔值设置为 wh TRUE e replaceTypen 是REPLACE_WITH_NULL、REPLACE_WITH_MODE、或WINSORIZE_VALUES。FALSE对于所有其他人，则默认为。
  - FALSE— 移除所有异常值
  - TRUE— 移除排名超出和中指定的百分位数上限阈值的异常值。minValue maxValue

以下示例显示了单个RecipeAction操作的语法。一个配方至少包含一个RecipeStep操作，一个配方步骤至少包含一个配方操作。配方操作会运行您指定的数据转换。一组配方操作按顺序运行以创建最终数据集。

JSON

下面显示了一个使用 JSON 语法作为RecipeStep DataBrew 配方示例成员的示例。RecipeAction有关显示配方操作列表的语法示例，请参阅定义配方结构。

Example 示例 JSON 文件

```
{
    "Action": {
        "Operation": "REPLACE_OUTLIERS",
        "Parameters": {
            "maxValue": "95",
            "minValue": "5",
            "modeType": "AVERAGE",
            "outlierStrategy": "Z_SCORE",
            "replaceType": "REPLACE_WITH_MODE",
            "sourceColumn": "name-of-existing-column",
            "threshold": "3",
            "trimValue": "TRUE"
        }
    }
}
```

有关在 API 操作中使用此配方操作的更多信息，请参阅CreateRecipe或UpdateRecipe。您可以在自己的代码中使用这些操作和其他 API 操作。

YAML

下面显示了一个使用 YAML 语法作为 DataBrew 配方示例成员RecipeStep的示例。RecipeAction有关显示配方操作列表的语法示例，请参阅定义配方结构。

Example 示例 YAML 文件

```
- Action:
  Operation: REMOVE_OUTLIERS
  Parameters:
    sourceColumn: name-of-existing-column
    outlierStrategy: Z_SCORE
    threshold: '3'
    replaceType: REPLACE_WITH_MODE
    modeType: AVERAGE
    minValue: '5'
    maxValue: '95'
    trimValue: 'TRUE'
```

有关在 API 操作中使用此配方操作的更多信息，请参阅CreateRecipe或UpdateRecipe。您可以在自己的代码中使用这些操作和其他 API 操作。

重新缩放异常值_WITH_Z_SCORE

根据参数中的设置，返回一个新列，每行中都有一个重新缩放的异常值值。此操作还将 Z 分数归一化应用于线性缩放的数据值，使平均值 (μ) 为 0，标准差 (σ) 为 1。我们建议使用此操作来处理异常值。

参数

- sourceColumn— 指定可能包含异常值的现有数值列的名称。
- targetColumn— 指定可能包含异常值的现有数值列的名称。
- outlierStrategy— 指定用于检测异常值的方法。有效值包括：
  - Z_SCORE— 当一个值与均值的偏差超过标准差阈值时，将其标识为异常值。
  - MODIFIED_Z_SCORE— 当一个值与中位数的偏差超过中位数绝对偏差阈值时，将其标识为异常值。
  - IQR— 当某个值超出列数据的第一个和最后一个四分位数时，将其标识为异常值。四分位数范围 (IQR) 用于衡量中间50%的数据点所在的位置。
- threshold— 检测异常值时使用的阈值。如果用计算的分数outlierStrategy超过此数字，则该sourceColumn值将被标识为异常值。默认值为 3。

以下示例显示了单个RecipeAction操作的语法。一个配方至少包含一个RecipeStep操作，一个配方步骤至少包含一个配方操作。配方操作会运行您指定的数据转换。一组配方操作按顺序运行以创建最终数据集。

JSON

下面显示了一个使用 JSON 语法作为RecipeStep DataBrew 配方操作示例成员的示例。RecipeAction有关显示配方操作列表的语法示例，请参阅定义配方结构。

Example 示例 JSON

```
{
    "Action": {
        "Operation": "RESCALE_OUTLIERS_WITH_Z_SCORE",
        "Parameters": {
            "sourceColumn": "name-of-existing-column",
            "targetColumn": "name-of-new-column",
            "outlierStrategy": "Z_SCORE",
            "threshold": "3"
        }
    }
}
```

有关在 API 操作中使用此配方操作的更多信息，请参阅CreateRecipe或UpdateRecipe。您可以在自己的代码中使用这些操作和其他 API 操作。

YAML

下面显示了一个使用 YAML 语法作为 DataBrew 配方操作示例RecipeStep成员的示例。RecipeAction有关显示配方操作列表的语法示例，请参阅定义配方结构。

Example 示例 YAML

```
- Action:
  Operation: REMOVE_OUTLIERS
  Parameters:
    sourceColumn: name-of-existing-column
    targetColumn: name-of-new-column
    outlierStrategy: Z_SCORE
    threshold: '3'
```

有关在 API 操作中使用此配方操作的更多信息，请参阅CreateRecipe或UpdateRecipe。您可以在自己的代码中使用这些操作和其他 API 操作。

使用_SKEW 重新缩放异常值

根据参数中的设置，返回一个新列，每行中都有一个重新缩放的异常值值。此操作通过应用指定的对数或根变换来减少分布偏度。我们建议使用此操作来处理偏斜的数据。

参数

- sourceColumn— 指定可能包含异常值的现有数值列的名称。
- targetColumn— 指定可能包含异常值的现有数值列的名称。
- outlierStrategy— 指定用于检测异常值的方法。有效值包括：
  - Z_SCORE— 当一个值与均值的偏差超过标准差阈值时，将其标识为异常值。
  - MODIFIED_Z_SCORE— 当一个值与中位数的偏差超过中位数绝对偏差阈值时，将其标识为异常值。
  - IQR— 当某个值超出列数据的第一个和最后一个四分位数时，将其标识为异常值。四分位数范围 (IQR) 用于衡量中间50%的数据点所在的位置。
- threshold— 指定检测异常值时要使用的阈值。如果用计算的分数outlierStrategy超过此数字，则该sourceColumn值将被标识为异常值。默认值为 3。
- skewFunction— 指定替换异常值时要使用的方法。有效值包括：
  - LOG — 应用强变换以减少正向和负偏差。这是一个自然对数 (2.718281828)。
  - ROOT (w value = 3 ith)-应用相当强的变换以减少正向和负偏差。（立方根）
  - ROOT (w value = 2 ith)-应用适度变换以仅减少正偏差。（平方根）
  - SQUARE — 应用适度的变换以减少负偏差。（方形对象）
  - 自定义转换-使用value参数中提供的自定义数字应用指定的ROOT变换LOG或转换。
- value— 指定用于自定义变换的值。如果skewFunction是 LOG，则此值表示日志的基础。如果skewFunction是 ROOT，则此值表示根的次方。

以下示例显示了单个RecipeAction操作的语法。一个配方至少包含一个RecipeStep操作，一个配方步骤至少包含一个配方操作。配方操作会运行您指定的数据转换。一组配方操作按顺序运行以创建最终数据集。

JSON

下面显示了一个使用 JSON 语法作为RecipeStep DataBrew 配方示例成员的示例。RecipeAction有关显示配方操作列表的语法示例，请参阅定义配方结构。

Example JSON 格式示例

```
{
    "Action": {
        "Operation": "RESCALE_OUTLIERS_WITH_SKEW",
        "Parameters": {
            "outlierStrategy": "Z_SCORE",
            "threshold": "3",
            "skewFunction": "ROOT",
            "sourceColumn": "name-of-existing-column",
            "targetColumn": "name-of-new-column",
            "value": "4"
        }
    }
}
```

有关在 API 操作中使用此配方操作的更多信息，请参阅CreateRecipe或UpdateRecipe。您可以在自己的代码中使用这些操作和其他 API 操作。

YAML

下面显示了一个使用 YAML 语法作为 DataBrew 配方示例成员RecipeStep的示例。RecipeAction有关显示配方操作列表的语法示例，请参阅定义配方结构。

Example 示例 YAML

```
- Action:
  Operation: RESCALE_OUTLIERS_WITH_SKEW
  Parameters:
    outlierStrategy: Z_SCORE
    threshold: '3'
    skewFunction: ROOT
    sourceColumn: name-of-existing-column
    targetColumn: name-of-new-column
    value: '4'
```

有关在 API 操作中使用此配方操作的更多信息，请参阅CreateRecipe或UpdateRecipe。您可以在自己的代码中使用这些操作和其他 API 操作。

列结构配方步骤

使用这些列结构配方步骤来修改数据的列结构。

主题

布尔运算

根据逻辑条件 IF 的结果创建新列。如果布尔表达式为真，则返回真值；如果布尔表达式为假，则返回假值，或者返回自定义值。

参数

- `trueValueExpression`— 满足条件时的结果。
- `falseValueExpression`— 未满足条件时的结果。
- `valueExpression`— 布尔条件。
- `withExpressions`— 聚合结果的配置。
- `targetColumn`-新创建的列的名称。

您可以在、和 ValueExpression 中使用常量值 trueValueExpression、列引用 falseValueExpression 和聚合结果。

Example 示例：常量值

保持不变的值，例如数字或句子。

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "It is true.",
        "falseValueExpression": "It is false.",
        "valueExpression": "`column.1` < 2000",
        "targetColumn": "result.column"
      }
    }
  }
}
```

Example 示例：列引用

作为数据集中的列的值。

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "`column.2`",
        "falseValueExpression": "`column.3`",
        "valueExpression": "`column.1` < `column.4`",
        "targetColumn": "result.column"
      }
    }
  }
}
```

Example 示例：汇总结果

由聚合函数计算的值。聚合函数对列执行计算，并返回单个值。

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
```

```
      "Parameters": {
        "trueValueExpression": "`:mincolumn.2`",
        "falseValueExpression": "`:maxcolumn.3`",
        "valueExpression": "`column.1` < `:avgcolumn.4`",
        "withExpressions": "[{\"name\":\"mincolumn.2\",\"value\":\"min(`column.2`)\",
\"type\":\"aggregate\"},{\"name\":\"maxcolumn.3\",\"value\":\"max(`column.3`)\",\"type
\":\"aggregate\"},{\"name\":\"avgcolumn.4\",\"value\":\"avg(`column.4`)\",\"type\":
\"aggregate\"}]",
        "targetColumn": "result.column"
      }
    }
  }
}
```

用户需要通过转义将 JSON 转换为字符串。

请注意，、和 ValueExpressi trueValueExpression on falseValueExpression 中的参数名称必须与 withExpressions 中的名称匹配。要使用某些列的聚合结果，您需要为它们创建参数并提供聚合函数。

Example 例如：

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "It is true.",
        "falseValueExpression": "It is false.",
        "valueExpression": "`column.1` < 2000",
        "targetColumn": "result.column"
      }
    }
  }
}
```

Example 示例：和/或

您可以使用 and 和 or 组合多个条件。

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
```

```
      "Parameters": {
        "trueValueExpression": "It is true.",
        "falseValueExpression": "It is false.",
        "valueExpression": "`column.1` < 2000 and `column.2` >= `column.3",
        "targetColumn": "result.column"
      }
    }
  }
}
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "`column.4`",
        "falseValueExpression": "`column.5`",
        "valueExpression": "startsWith(`column1`, 'value1') or endsWith(`column2`,
 'value2')",
        "targetColumn": "result.column"
      }
    }
  }
}
```

有效的聚合函数

下表显示了可以在布尔运算中使用的所有有效聚合函数。

列类型	状况	值表达式	使用表达式	返回值
数值	总和	`: sum.column.1`	`[ { "name": "sum.column.1", "value": "sum(`column.1`)", "type":`	返回以下之和 `column.1`

列类型	状况	值表达式	使用表达式	返回值
			"aggregate" }]	

	平均值	`: mean.column.1`	[
			{
			"name": "mean.column.1",
			"value": "avg(`column.1`)",
			"type": "aggregate"
			}
			]
				返回的均值 column.1

列类型	状况	值表达式	使用表达式	返回值
	平均绝对偏差	`:mean 绝对偏差。column.1`	`[ { "name": "meanabsolutedeviation.column.1", "value": "mean_absolute_deviation(`column.1`)", "type": "aggregate" } ]`	返回的平均绝对偏差 `column.1`
	中位数	`: median.column.1`	`[ { "name": "median.column.1", "value": "median(`column.1`)", "type": "aggregate" } ]`	返回的中位数 `column.1`

列类型	状况	值表达式	使用表达式	返回值
	产品	`: product.column.1`	`[ { "name": "product.column.1", "value": "product(`column.1`)", "type": "aggregate" } ]`	返回以下值的产品 column.1
	标准差	`: 标准差。column.1`	`[ { "name": "standard deviation.column.1", "value": "stddev(`column.1`)", "type": "aggregate" } ]`	返回标准差 column.1

列类型	状况	值表达式	使用表达式	返回值
	方差	`: variance.column.1`	`[ { "name": "variance.column.1", "value": "variance(`column.1`)", "type": "aggregate" } ]`	返回的方差 `column.1`
	平均值的标准误差	`: standarderrorofmean.column.1`	`[ { "name": "standarderrorofmean.column.1", "value": "standard_error_of_mean(`column.1`)", "type": "aggregate" } ]`	返回均值的标准误差 `column.1`

列类型	状况	值表达式	使用表达式	返回值
	偏斜	`: skewness.column.1`	`[ { "name": "skewness.column.1", "value": "skewness(`column.1`)", "type": "aggregate" } ]`	返回的偏度 `column.1`
	峰度	`:kurtosis.column.1`	`[ { "name": "kurtosis.column.1", "value": "kurtosis(`column.1`)", "type": "aggregate" } ]`	返回的峰度 `column.1`

列类型	状况	值表达式	使用表达式	返回值
日期时间/数字/文本	计数	`: count.column.1`	`[ { "name": "count.column.1", "value": "count(`column.1`)", "type": "aggregate" } ]`	返回中的总行数 column.1
	去重计数	`: countdistict.column.1`	`[ { "name": "count.column.1", "value": "count(distinct `column.1`)", "type": "aggregate" } ]`	返回中不同行的总数 column.1

列类型	状况	值表达式	使用表达式	返回值
	最小值	`: min.column.1`	`[ { "name": "min.column.1", "value": "min(`column.1`)", "type": "aggregate" } ]`	返回最小值 column.1
	最大值	`: max.column.1`	`[ { "name": "max.column.1", "value": "max(`column.1`)", "type": "aggregate" } ]`	返回的最大值 column.1

ValueExpression 中的有效条件

下表显示了支持的条件和您可以使用的值表达式。

列类型	状况	值表达式	描述
字符串	包含	包含（“列”，“文本”）	用于测试列中的值是否包含文本的条件
	不包含	！包含（“列”，“文本”）	用于测试列中的值是否不包含文本的条件
	匹配项	匹配（“列”、“模式”）	测试列中的值是否与模式匹配的条件
	不匹配	！匹配（“列”、“模式”）	用于测试列中的值是否与模式不匹配的条件
	开始于	开头为（“列”，“文本”）	测试列中的值是否以文本开头的条件
	开头不是	！开头为（“列”，“文本”）	用于测试列中的值是否不是以文本开头的条件
	结束于	endSwith（“列”，“文本”）	测试列中的值是否以文本结尾的条件
	结尾不是	！endSwith（“列”，“文本”）	用于测试列中的值是否不以文本结尾的条件
数值	Less than	`列` < 数字	用于测试列中的值是否小于数字的条件
	小于或等于	`列` <= 数字	测试列中的值是否小于或等于数字的条件
	Greater than	“列” > 数字	用于测试列中的值是否大于数字的条件

列类型	状况	值表达式	描述
	大于或等于	`列` >= 数字	测试列中的值是否大于或等于数字的条件
	介于	isBetween（“列”、minNumber、maxNumber）	用于测试列中的值是否介于 minNumber 和 maxNumber 之间的条件
	不介于	！isBetween（“列”、minNumber、maxNumber）	用于测试列中的值是否不在 minNumber 和 maxNumber 之间的条件
布尔值	true	`column` = TRUE	测试列中的值是否为布尔值 TRUE 的条件
	FALSE	`column` = FALSE	测试列中的值是否为布尔值 FALSE 的条件
日期/时间戳	早于	`列` < '日期'	用于测试列中的值是否早于日期的条件
	早于或等于	`列` <= '日期'	用于测试列中的值是否早于或等于日期的条件
	晚于	`列` > '日期'	用于测试列中的值是否晚于日期的条件
	晚于或等于	`列` >= '日期'	用于测试列中的值是否晚于或等于日期的条件
字符串/数字/日期/时间戳	true	`列` = '值'	测试列中的值是否正好为值的条件

列类型	状况	值表达式	描述
	Is not	`专栏` ！ = 'value'	用于测试列中的值是否不是值的条件
	缺失值	缺失（“列”）	测试列中是否缺少值的条件
	没有缺失值	！ 缺失（“列”）	用于测试列中值是否未丢失的条件
	有效	是有效的（“列”，数据类型）	用于测试列中的值是否有效的条件（该值属于数据类型或可以转换为数据类型）
	无效	！ 是有效的（“列”，数据类型）	用于测试列中的值是否无效的条件（该值属于数据类型或可以转换为数据类型）
嵌套	缺失值	缺失（“列”）	测试列中是否缺少值的条件
	没有缺失值	！ 缺失（“列”）	用于测试列中值是否未丢失的条件
	有效	是有效的（“列”，数据类型）	用于测试列中的值是否有效的条件（该值属于数据类型或可以转换为数据类型）
	无效	！ 是有效的（“列”，数据类型）	用于测试列中的值是否无效的条件（该值属于数据类型或可以转换为数据类型）

案例操作

根据逻辑条件 CASE 的结果创建新列。case 操作会遍历案例条件，并在满足第一个条件时返回一个值。条件为真后，该操作将停止读取并返回结果。如果没有条件为真，则返回默认值。

参数

- `valueExpression`— 条件。
- `withExpressions`— 聚合结果的配置。
- `targetColumn`— 新创建列的名称。

Example 示例

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "CASE_OPERATION",
      "Parameters": {
        "valueExpression": "case when `column11` < `column.2` then 'result1' when
 `column2` < 'value2' then 'result2' else 'high' end",
        "targetColumn": "result.column"
      }
    }
  }
}
```

有效的聚合函数

下表显示了可以在 case 操作中使用的所有有效聚合函数。

列类型	状况	值表达式	使用表达式	返回值
数值	总和	`: sum.column.1`	`[ { "name": "sum.colu mn.1",`	返回以下之和 `column.1`

列类型	状况	值表达式	使用表达式	返回值
			"value": "sum(`column.1`)", "type": "aggregate" }]	
	平均值	`: mean.column.1`	[{ "name": "mean.column.1", "value": "avg(`column.1`)", "type": "aggregate" }]	返回的均值 column.1

列类型	状况	值表达式	使用表达式	返回值
	平均绝对偏差	`:mean 绝对偏差。column.1`	`[ { "name": "meanabsolutedeviation.column.1", "value": "mean_absolute_deviation(`column.1`)", "type": "aggregate" } ]`	返回的平均绝对偏差 `column.1`
	中位数	`: median.column.1`	`[ { "name": "median.column.1", "value": "median(`column.1`)", "type": "aggregate" } ]`	返回的中位数 `column.1`

列类型	状况	值表达式	使用表达式	返回值
	产品	`: product.column.1`	`[ { "name": "product.column.1", "value": "product(`column.1`)", "type": "aggregate" } ]`	返回以下值的产品 column.1
	标准差	`: 标准差。column.1`	`[ { "name": "standard deviation.column.1", "value": "stddev(`column.1`)", "type": "aggregate" } ]`	返回标准差 column.1

列类型	状况	值表达式	使用表达式	返回值
	方差	`: variance.column.1`	`[ { "name": "variance.column.1", "value": "variance(`column.1`)", "type": "aggregate" } ]`	返回的方差 column.1
	平均值的标准误差	`: standarderrorofmean.column.1`	`[ { "name": "standarderrorofmean.column.1", "value": "standard_error_of_mean(`column.1`)", "type": "aggregate" } ]`	返回均值的标准误差 column.1

列类型	状况	值表达式	使用表达式	返回值
	偏斜	`: skewness.column.1`	`[ { "name": "skewness.column.1", "value": "skewness(`column.1`)", "type": "aggregate" } ]`	返回的偏度 column.1
	峰度	`:kurtosis.column.1`	`[ { "name": "kurtosis.column.1", "value": "kurtosis(`column.1`)", "type": "aggregate" } ]`	返回的峰度 column.1

列类型	状况	值表达式	使用表达式	返回值
日期时间/数字/文本	计数	`: count.column.1`	`[ { "name": "count.column.1", "value": "count(`column.1`)", "type": "aggregate" } ]`	返回中的总行数 column.1
	去重计数	`: countdistict.column.1`	`[ { "name": "count.column.1", "value": "count(distinct `column.1`)", "type": "aggregate" } ]`	返回中不同行的总数 column.1

列类型	状况	值表达式	使用表达式	返回值
	最小值	`: min.column.1`	`[ { "name": "min.column.1", "value": "min(`column.1`)", "type": "aggregate" } ]`	返回最小值 column.1
	最大值	`: max.column.1`	`[ { "name": "max.column.1", "value": "max(`column.1`)", "type": "aggregate" } ]`	返回的最大值 column.1

ValueExpression 中的有效条件

下表显示了支持的条件和您可以使用的值表达式。

列类型	状况	值表达式	描述
字符串	包含	包含（“列”，“文本”）	用于测试列中的值是否包含文本的条件
	不包含	！包含（“列”，“文本”）	用于测试列中的值是否不包含文本的条件
	匹配项	匹配（“列”、“模式”）	测试列中的值是否与模式匹配的条件
	不匹配	！匹配（“列”、“模式”）	用于测试列中的值是否与模式不匹配的条件
	开始于	开头为（“列”，“文本”）	测试列中的值是否以文本开头的条件
	开头不是	！开头为（“列”，“文本”）	用于测试列中的值是否不是以文本开头的条件
	结束于	endSwith（“列”，“文本”）	测试列中的值是否以文本结尾的条件
	结尾不是	！endSwith（“列”，“文本”）	用于测试列中的值是否不以文本结尾的条件
数值	Less than	`列` < 数字	用于测试列中的值是否小于数字的条件
	小于或等于	`列` <= 数字	测试列中的值是否小于或等于数字的条件
	Greater than	“列” > 数字	用于测试列中的值是否大于数字的条件

列类型	状况	值表达式	描述
	大于或等于	`列` >= 数字	测试列中的值是否大于或等于数字的条件
	介于	isBetween（“列”、minNumber、maxNumber）	用于测试列中的值是否介于 minNumber 和 maxNumber 之间的条件
	不介于	！isBetween（“列”、minNumber、maxNumber）	用于测试列中的值是否不在 minNumber 和 maxNumber 之间的条件
布尔值	true	`column` = TRUE	测试列中的值是否为布尔值 TRUE 的条件
	FALSE	`column` = FALSE	测试列中的值是否为布尔值 FALSE 的条件
日期/时间戳	早于	`列` < '日期'	用于测试列中的值是否早于日期的条件
	早于或等于	`列` <= '日期'	用于测试列中的值是否早于或等于日期的条件
	晚于	`列` > '日期'	用于测试列中的值是否晚于日期的条件
	晚于或等于	`列` >= '日期'	用于测试列中的值是否晚于或等于日期的条件
字符串/数字/日期/时间戳	true	`列` = '值'	测试列中的值是否正好为值的条件

列类型	状况	值表达式	描述
	Is not	`专栏` ！ = 'value'	用于测试列中的值是否不是值的条件
	缺失值	缺失（“列”）	测试列中是否缺少值的条件
	没有缺失值	！ 缺失（“列”）	用于测试列中值是否未丢失的条件
	有效	是有效的（“列”，数据类型）	用于测试列中的值是否有效的条件（该值属于数据类型或可以转换为数据类型）
	无效	！ 是有效的（“列”，数据类型）	用于测试列中的值是否无效的条件（该值属于数据类型或可以转换为数据类型）
嵌套	缺失值	缺失（“列”）	测试列中是否缺少值的条件
	没有缺失值	！ 缺失（“列”）	用于测试列中值是否未丢失的条件
	有效	是有效的（“列”，数据类型）	用于测试列中的值是否有效的条件（该值属于数据类型或可以转换为数据类型）
	无效	！ 是有效的（“列”，数据类型）	用于测试列中的值是否无效的条件（该值属于数据类型或可以转换为数据类型）

FLAG_COLUMN_FROM_NULL

根据现有列中是否存在空值来创建新列。

参数

- sourceColumn – 现有列的名称。
- targetColumn— 要创建的新列的名称。
- flagType— 必须设置为的值Null values。
- trueString— 如果在源中找到空值，则为新列的值。如果未指定值，则默认值为 True。
- falseString— 如果在源中找到非 null 值，则为新列的值。如果未指定值，则默认值为 False。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FLAG_COLUMN_FROM_NULL",
        "Parameters": {
            "flagType": "Null values",
            "sourceColumn": "weight_kg",
            "targetColumn": "is_weight_kg_missing"
        }
    }
}
```

FLAG_COLUMN_FROM PATTERN

根据现有列中是否存在用户指定的模式来创建新列。

参数

- sourceColumn – 现有列的名称。
- targetColumn— 要创建的新列的名称。
- flagType— 必须设置为的值Pattern。
- pattern— 一个正则表达式，表示要评估的模式。
- trueString— 如果在源中找到空值，则为新列的值。如果未指定值，则默认值为 True。
- falseString— 如果在源中找到非 null 值，则为新列的值。如果未指定值，则默认值为 False。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FLAG_COLUMN_FROM_PATTERN",
        "Parameters": {
            "falseString": "No",
            "flagType": "Pattern",
            "pattern": "N.*",
            "sourceColumn": "wind_direction",
            "targetColumn": "northerly",
            "trueString": "yes"
        }
    }
}
```

MERGE

将两列或多列合并为一个新列。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示要合并的一列或多列的列表。
- `delimiter`— 值之间的可选分隔符，显示在目标列中。
- `targetColumn`-要创建的合并列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MERGE",
        "Parameters": {
            "delimiter": " ",
            "sourceColumns": "[\"first_name\",\"last_name\"]",
            "targetColumn": "Merged Column 1"
        }
    }
}
```

在分隔符之间拆分列

根据开头和结尾分隔符将一列拆分为三个新列。

参数

- sourceColumn – 现有列的名称。
- patternOption1— 一个 JSON 编码的字符串，表示一个或多个表示第一个分隔符的字符。
- patternOption2— 一个 JSON 编码的字符串，表示一个或多个表示第二个分隔符的字符。
- pattern— 拆分数据时用作分隔符的一个或多个字符。
- includeInSplit— 如果为 true，则在新列中包含该模式；否则，该模式将被丢弃。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_BETWEEN_DELIMITER",
        "Parameters": {
            "patternOption1": "{\"pattern\":\"H\",\"includeInSplit\":true}",
            "patternOption2": "{\"pattern\":\"M\",\"includeInSplit\":true}",
            "sourceColumn": "last_name"
        }
    }
}
```

在位置之间拆分列

根据您指定的偏移量将一列拆分为三个新列。

参数

- sourceColumn – 现有列的名称。
- startPosition— 分割开始的角色位置。
- endPosition— 分割要结束的角色位置。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_BETWEEN_POSITIONS",
        "Parameters": {
            "endPosition": "12",
            "sourceColumn": "last_name",
            "startPosition": "2"
        }
    }
}
```

从末尾拆分列

将一列拆分为两列新列，距离字符串末尾有一定的偏移量。

参数

- sourceColumn – 现有列的名称。
- position— 字符位置，从字符串的右端开始，即要进行拆分的位置。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_FROM_END",
        "Parameters": {
            "position": "1",
            "sourceColumn": "nationality"
        }
    }
}
```

从头开始拆分列

将一列拆分为两列新列，距离字符串开头有一定的偏移量。

参数

- sourceColumn – 现有列的名称。

- position— 字符位置，从字符串的左端开始，即要进行拆分的位置。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_FROM_START",
        "Parameters": {
            "position": "1",
            "sourceColumn": "first_name"
        }
    }
}
```

SPLIT_COLUMN_MULTIPLE_DELIMITER

根据多个分隔符拆分一列。

参数

- sourceColumn – 现有列的名称。
- patternOptions— 一个 JSON 编码的字符串，表示一个或多个确定拆分标准的模式。
- pattern— 拆分数据时用作分隔符的一个或多个字符。
- limit— 要执行多少次拆分。最小值为 1；最大值为 20。
- includeInSplit— 如果为 true，则在新列中包含该模式；否则，该模式将被丢弃。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_MULTIPLE_DELIMITER",
        "Parameters": {
            "limit": "1",
            "patternOptions": "[{\"pattern\":\",\",\"includeInSplit\":true},{\"pattern
\":\" \",\"includeInSplit\":true}]",
            "sourceColumn": "description"
        }
```

```
    }
}
```

SPLIT_COLUMN_SINGLE_SINGLEMITER

根据特定的分隔符将一列拆分为一列或多列新列。

参数

- sourceColumn – 现有列的名称。
- pattern— 拆分数据时用作分隔符的一个或多个字符。
- limit— 要执行多少次拆分。最小值为 1；最大值为 20。
- includeInSplit— 如果为 true，则在新列中包含该模式；否则，该模式将被丢弃。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_SINGLE_DELIMITER",
        "Parameters": {
            "includeInSplit": "true",
            "limit": "1",
            "pattern": "/",
            "sourceColumn": "info_url"
        }
    }
}
```

使用间隔拆分列

以 n 个字符的间隔拆分一列，其中指定 n。

参数

- sourceColumn – 现有列的名称。
- startPosition— 分割开始的角色位置。
- interval— 下次拆分之前要跳过的字符数。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_WITH_INTERVALS",
        "Parameters": {
            "interval": "4",
            "sourceColumn": "nationality",
            "startPosition": "1"
        }
    }
}
```

列格式化配方步骤

使用列格式设置方法步骤来更改列中数据的格式。

主题

- 数字格式
- 格式_电话_号码

数字格式

返回将数值转换为格式化字符串的列。

参数

- sourceColumn – 字符串。现有列的名称。
- decimalPlaces— 整数。小数分隔符之后的位数值。
- numericDecimalSeparator – 字符串。以下值之一，表示小数分隔符：
  - "."
  - ","
- numericThousandSeparator – 字符串。以下值之一，表示千位分隔符：
  - Null。表示未启用千位分隔符。
  - ","

  - " "
  - "."
  - "\\"
- numericAbbreviatedUnit – 字符串。以下值之一，表示缩写：
  - Null。表示缩写单位未启用。
  - “千”
  - “亿”
  - “亿”
  - “万亿”
- numericUnitAbbreviation – 字符串。以下值之一或任何自定义值，表示单位缩写：
  - Null。表示未启用单位缩写。
  -

缩写	Options
千	K、k、M、千、自定义
亿	M、m、MM、million、custom
亿	B、bn、十亿、自定义
万亿	T、tn、万亿、自定义

Example 示例

```
{
    "RecipeAction": {
        "Operation": "NUMBER_FORMAT",
        "Parameters": {
            "sourceColumn": "income",
            "decimalPlaces": "2",
            "numericDecimalSeparator": ".",
            "numericThousandSeparator": ",",
            "numericAbbreviatedUnit": "THOUSAND",
            "numericUnitAbbreviation": "K"
        }
    }
```

```
}
```

格式_电话_号码

返回将电话号码字符串转换为格式化值的列。

参数

- sourceColumn – 现有列的名称。
- phoneNumberFormat— 将电话号码转换为的格式。如果未指定格式E.164，则默认为国际公认的标准电话号码格式。有效值包括：
  - E164（省略之后的E句号）
- defaultRegion— 由两个或三个大写字母组成的有效区域代码，当电话号码本身没有国家代码时，它指定电话号码所在的地区。最多defaultRegionColumn只能提供其中一个defaultRegion或一个。
- defaultRegionColumn— 高级数据类型的列的名称Country。当电话号码本身不存在国家/地区代码时，指定列中的区域代码用于确定电话号码的国家/地区代码。最多defaultRegionColumn只能提供其中一个defaultRegion或一个。

注意事项

- 无法格式化为有效电话号码的输入将保持不变。
- 如果未提供默认区域，并且电话号码不是以加号 (+) 和国家/地区电话代码开头，则电话号码不会格式化。

Example

示例：固定默认区域

```
{
    "Action": {
        "Operation": "FORMAT_PHONE_NUMBER",
        "Parameters": {
            "sourceColumn": "Phone Number",
            "defaultRegion": "US"
        }
    }
```

```
}
```

示例：默认区域列选项

```
{
    "Action": {
        "Operation": "FORMAT_PHONE_NUMBER",
        "Parameters": {
            "sourceColumn": "Phone Number",
            "defaultRegionColumn": "Country Code"
        }
    }
}
```

数据结构配方步骤

使用这些配方步骤从不同的角度对数据进行制表和汇总，或者执行高级功能。

主题

- NEST_TO_ARRAY
- NEST_TO_MAP
- NEST_TO_STRUCT
- UNNEST_ARRAY
- UNNEST_MAP
- UNNEST_STRUCT
- UNNEST_STRUCT_N
- GROUP_BY
- JOIN
- PIVOT
- SCALE
- 调换顺序
- 联合
- UNPIVOT

NEST_TO_ARRAY

将用户选择的列转换为数组值。创建结果数组时，所选列的顺序保持不变。将不同的列数据类型转换为支持所有列数据类型的通用类型。

参数

- `sourceColumns`— 源列列表。
- `targetColumn`-目标列的名称。
- `removeSourceColumns`— 包含值`true`或`false`，表示用户是否要删除选定的源列。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "NEST_TO_ARRAY",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "columnName",
            "removeSourceColumns": "true"
        }
    }
}
```

NEST_TO_MAP

将用户选择的列转换为键值对，每个键值对都有一个代表列名的键和一个代表行值的值。创建生成的地图时，所选列的顺序不会保持不变。将不同的列数据类型转换为支持所有列数据类型的通用类型。

参数

- `sourceColumns`— 源列列表。
- `targetColumn`-目标列的名称。
- `removeSourceColumns`— 包含值`true`或`false`，表示用户是否要删除选定的源列。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "NEST_TO_MAP",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "columnName",
            "removeSourceColumns": "true"
        }
    }
}
```

NEST_TO_STRUCT

将用户选择的列转换为键值对，每个键值对都有一个代表列名的键和一个代表行值的值。所选列的顺序和每列的数据类型都保留在生成的结构体中。

参数

- `sourceColumns`— 源列列表。
- `targetColumn`-目标列的名称。
- `removeSourceColumns`— 包含值`true`或`false`，表示用户是否要删除选定的源列。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "NEST_TO_STRUCT",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "columnName",
            "removeSourceColumns": "true"
        }
    }
}
```

UNNEST_ARRAY

`array`将类型为的列取消嵌套到新列中。如果数组包含多个值，则会生成与每个元素对应的一行。此函数仅解除数组列的一级嵌套。

参数

- sourceColumn— 现有列的名称。此列必须是struct类型。
- targetColumn— 生成的目标列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "UNNEST_ARRAY",
        "Parameters": {
            "sourceColumn": "address",
            "targetColumn": "address"
        }
    }
}
```

UNNEST_MAP

取消嵌套类型为的列map，并为键和值生成一列。如果有多个键值对，则会生成与每个键值对应的一行。此函数仅解除地图列的一个层级的嵌套。

参数

- sourceColumn— 现有列的名称。此列必须是struct类型。
- removeSourceColumn— 如果true，则在函数完成后删除源列。
- targetColumn— 如果提供，则生成的每列都将以此作为前缀。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "UNNEST_MAP",
        "Parameters": {
            "sourceColumn": "address",
            "removeSourceColumn": "false",
            "targetColumn": "address"
        }
```

```
    }
}
```

UNNEST_STRUCT

取消嵌套类型为的列，struct然后为结构中存在的每个键生成一列。此函数仅解除结构级别 1 的嵌套。

参数

- sourceColumn— 现有列的名称。此列必须是结构类型。
- removeSourceColumn— 如果true，则在函数完成后删除源列。
- targetColumn— 如果提供，则生成的每列都将以此作为前缀。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "UNNEST_STRUCT",
        "Parameters": {
            "sourceColumn": "address",
            "removeSourceColumn": "false"
            "targetColumn": "add"
        }
    }
}
```

UNNEST_STRUCT_N

为类型为所选列的每个字段创建一个新列struct。

例如，给定以下结构：

```
            user {
               name: “Ammy”
               address: {
                  state: "CA",
                  zipcode: 12345
               }
```

```
            }
```

此函数创建 3 列：

user.name	用户地址状态	用户地址.zipcode
Ammy	CA	12345

参数

- `sourceColumns`— 源列列表。
- `regexColumnSelector`— 用于选择要取消嵌套的列的正则表达式。
- `removeSourceColumn`— 布尔值。如果为 true，则删除源列；否则将其保留。
- `unnestLevel`— 要解除嵌套的关卡数量。
- `delimiter`— 在新创建的列名中使用分隔符来分隔结构的不同级别。例如：如果分隔符为 “/”，则列名将采用以下形式：“用户/地址/状态”。
- `conditionExpressions`— 条件表达式。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "UNNEST_STRUCT_N",
        "Parameters": {
            "sourceColumns": "[\"address\"]",
            "removeSourceColumn": "true",
            "unnestLevel": "2",
            "delimiter": "/"
        }
    }
}
```

GROUP_BY

通过按一列或多列对行进行分组，然后对每个组应用聚合函数来汇总数据。

参数

- sourceColumns— 一个 JSON 编码的字符串，表示构成每个组基础的列的列表。
- groupByAggFunctions— 一个 JSON 编码的字符串，表示要应用的聚合函数列表。（如果您不想要聚合，请指定UNAGGREGATED。）
- useNewDataFrame— 如果为 true，GROUP_BY 的结果将在项目会话中可用，替换其当前内容。

Example 示例

```
[
  {
    "Action": {
      "Operation": "GROUP_BY",
      "Parameters": {
        "groupByAggFunctionOptions": "[{\"sourceColumnName\":\"all_votes\",
\"targetColumnName\":\"all_votes_count\",\"targetColumnDataType\":\"number\",
\"functionName\":\"COUNT\"}]",
        "sourceColumns": "[\"year\",\"state_name\"]",
        "useNewDataFrame": "true"
      }
    }
  }
]
```

JOIN

对两个数据集执行连接操作。

参数

- joinKeys— 一个 JSON 编码的字符串，表示每个数据集中用作联接键的列的列表。
- joinType— 要执行的联接类型。必须是以下之一:INNER_JOINLEFT_JOIN | RIGHT_JOIN | OUTER_JOIN | LEFT_EXCLUDING_JOIN | RIGHT_EXCLUDING_JOIN | OUTER_EXCLUDING_JOIN
- leftColumns— 一个 JSON 编码的字符串，表示当前活动数据集中的列列表。
- rightColumns— 一个 JSON 编码的字符串，表示来自另一个（辅助）数据集的列列表，以连接到当前数据集。

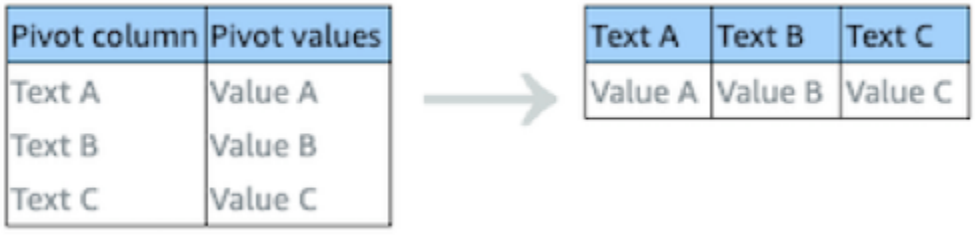
- secondInputLocation— 解析为辅助数据集数据文件的 Amazon S3 网址。
- secondaryDatasetName— 辅助数据集的名称。

Example 示例

```
{
    "Action": {
        "Operation": "JOIN",
        "Parameters": {
            "joinKeys": "[{\"key\":\"assembly_session\",\"value\":\"assembly_session
\"},{\"key\":\"state_code\",\"value\":\"state_code\"}]",
            "joinType": "INNER_JOIN",
            "leftColumns": "[\"year\",\"assembly_session\",\"state_code\",\"state_name
\",\"all_votes\",\"yes_votes\",\"no_votes\",\"abstain\",\"idealpoint_estimate
\",\"affinityscore_usa\",\"affinityscore_russia\",\"affinityscore_china\",
\"affinityscore_india\",\"affinityscore_brazil\",\"affinityscore_israel\"]",
            "rightColumns": "[\"assembly_session\",\"vote_id\",\"resolution\",
\"state_code\",\"state_name\",\"member\",\"vote\"]",
            "secondInputLocation": "s3://databrew-public-datasets-us-east-1/votes.csv",
            "secondaryDatasetName": "votes"
        }
    }
}
```

PIVOT

将选定列中的所有行值转换为带有值的单个列。



参数

- sourceColumn— 现有列的名称。该列最多可以有 10 个不同的值。
- valueColumn— 现有列的名称。该列最多可以有 10 个不同的值。
- aggregateFunction— 聚合函数的名称。如果您不想聚合，请使用关键字COLLECT_LIST。

Example 示例

```
{
    "Action": {
        "Operation": "PIVOT",
        "Parameters": {
            "aggregateFunction": "SUM",
            "sourceColumn": "state_name",
            "valueColumn": "all_votes"
        }
    }
}
```

SCALE

缩放或标准化数值列中的数据范围。

参数

- `sourceColumn`— 现有列的名称。
- `strategy`— 要应用于列值的操作：
  - `MIN_MAX`— 将值重新缩放到 [0,1] 范围内。
  - `SCALE_BETWEEN`— 将值重新缩放到两个指定值的范围内。
  - `MEAN_NORMALIZATION`— 在 [-1, 1] 的范围内重新调整数据的平均值 (μ) 为 0，标准差 (σ) 为 1。
  - `Z_SCORE`— 对数据值进行线性缩放，使平均值 (μ) 为 0，标准差 (σ) 为 1。最适合处理异常值。
- `targetColumn`— 要包含结果的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "NORMALIZATION",
        "Parameters": {
            "sourceColumn": "all_votes",
            "strategy": "MIN_MAX",
            "targetColumn": "all_votes_normalized"
        }
```

```
    }
}
```

调换顺序

将所有选定的行转换为列，将列转换为行。

Column 1	Column A	Column B	Column C
Row A	Value A	Value B	Value C
Row B	Value A1	Value B1	Value C1

New column	Row A	Row B
Column A	Value A	Value A1
Column B	Value B	Value B1
Column C	Value C	Value C1

参数

- `pivotColumns`— 一个 JSON 编码的字符串，表示行将转换为列名的列列表。
- `valueColumns`— 一个 JSON 编码的字符串，表示要转换为行的一列或多列的列表。
- `aggregateFunction`— 聚合函数的名称。如果您不想聚合，请使用关键字`COLLECT_LIST`。
- `newColumn`— 将转置后的列作为值保存的列。

Example 示例

```
{
    "Action": {
        "Operation": "TRANSPOSE",
        "Parameters": {
            "pivotColumns": "[\"Teacher\"]",
            "valueColumns": "[\"Tom\",\"John\",\"Harry\"]",
            "aggregateFunction": "COLLECT_LIST",
            "newColumn": "Student"
        }
    }
```

```
}
```

联合

将两个或更多数据集中的行合并到单个结果中。

参数

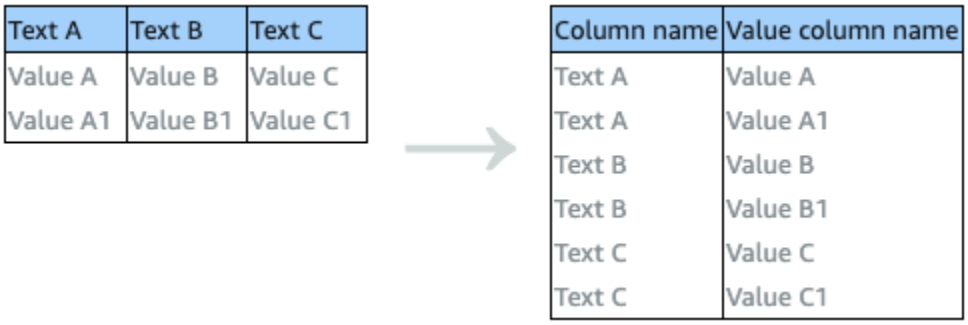
- `datasetsColumns`— 一个 JSON 编码的字符串，表示数据集中所有列的列表。
- `secondaryDatasetNames`— 一个 JSON 编码的字符串，表示一个或多个辅助数据集的列表。
- `secondaryInputs`— 一个 JSON 编码的字符串，表示 Amazon S3 存储桶和对象密钥名称的列表，这些名称告诉 DataBrew 在哪里可以找到辅助数据集。
- `targetColumnNames`— 一个 JSON 编码的字符串，表示结果的列名列表。

Example 示例

```
{
    "Action": {
        "Operation": "UNION",
        "Parameters": {
            "datasetsColumns": "[[\"assembly_session\",\"state_code\",
\"state_name\",\"year\",\"all_votes\",\"yes_votes\",\"no_votes\",\"abstain
\",\"idealpoint_estimate\",\"affinityscore_usa\",\"affinityscore_russia\",
\"affinityscore_china\",\"affinityscore_india\",\"affinityscore_brazil\",
\"affinityscore_israel\"],[\"assembly_session\",\"state_code\",\"state_name
\",null,null,null,null,null,null,null,null,null,null,null,null]]",
            "secondaryDatasetNames": "[\"votes\"]",
            "secondaryInputs": "[{\"S3InputDefinition\":{\"Bucket\":\"databrew-public-
datasets-us-east-1\",\"Key\":\"votes.csv\"}}]",
            "targetColumnNames": "[\"assembly_session\",\"state_code\",\"state_name\",
\"year\",\"all_votes\",\"yes_votes\",\"no_votes\",\"abstain\",\"idealpoint_estimate
\",\"affinityscore_usa\",\"affinityscore_russia\",\"affinityscore_china\",
\"affinityscore_india\",\"affinityscore_brazil\",\"affinityscore_israel\"]"
        }
    }
}
```

UNPIVOT

将选定行中的所有列值转换为带有值的单独行。



Text A	Text B	Text C
Value A	Value B	Value C
Value A1	Value B1	Value C1

Column name	Value column name
Text A	Value A
Text A	Value A1
Text B	Value B
Text B	Value B1
Text C	Value C
Text C	Value C1

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示要取消透视的一列或多列的列表。
- `unpivotColumn`— 取消透视操作的值列。
- `valueColumn`— 用于保存未旋转值的列。

Example 示例

```
{
    "Action": {
        "Operation": "UNPIVOT",
        "Parameters": {
            "sourceColumns": "[\"idealpoint_estimate\"]",
            "unpivotColumn": "unpivoted_idealpoint_estimate",
            "valueColumn": "unpivoted_column_values"
        }
    }
}
```

数据科学配方步骤

使用这些配方步骤从不同的角度对数据进行制表和汇总，或者执行高级转换。

主题

- 二值化
- 分桶化
- 分类映射
- ONE_HOT_ENCODING

- SCALE
- 偏斜
- 令牌

二值化

获取选定数值源列中的所有值，将其与阈值进行比较，然后输出每行为 1 或 0 的新列。

参数

- `sourceColumn` – 现有列的名称。

  `targetColumn`-要创建的新列的名称。

  `threshold`— 表示赋值 0 或 1 的阈值的数字。

  `flip`— 可以选择翻转二进制赋值，以便将较低的值赋为 1，将较高的值赋为 0。当 flip 参数为 true 时，小于或等于阈值的值会生成 1，大于阈值的值会生成 0。

Example 示例

```
{
    "Action": {
        "Operation": "BINARIZATION",
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "threshold": "100.0",
            "flip": "false"
        }
    }
}
```

分桶化

Bucketization（在控制台中称为 Binning）获取一列数值中的项目，将它们分组为由数值范围定义的分箱，然后输出一个显示每行分箱的新列。可以使用拆分（拆分）或百分比完成拆分。下面的第一个示例使用拆分，第二个示例使用百分比。

参数

- sourceColumn – 现有列的名称。

  targetColumn-要创建的新列的名称。

  bucketNames— 存储桶名称列表。

  splits— 存储桶等级列表。存储桶是连续的，存储桶的上限将是下一个存储桶的下限。

  percentage— 每个存储桶将以百分比来描述。

Example 使用拆分

```
{
    "Action": {
        "Operation": "BUCKETIZATION",
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "bucketNames": "[\"Bin1\",\"Bin2\",\"Bin3\"]",
            "splits": "[\"-Infinity\",\"2\",\"20\",\"Infinity\"]"
        }
    }
}
```

Example 使用百分比的示例

```
{
    "Action": {
        "Operation": "BUCKETIZATION",
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "bucketNames": "[\"Bin1\",\"Bin2\"]",
            "percentage": "50"
        }
    }
}
```

分类映射

将一个或多个类别值映射到数值或其他值

参数

- `sourceColumn` – 现有列的名称。

  `categoryMap`— 一个 JSON 编码的字符串，表示值到类别的映射。

  `deleteOtherRows`— 如果`true`，则所有未映射的行都将从数据集中移除。

  `other`— 如果提供，则所有未映射的值都将替换为该值。

  `keepOthers`— 如果为 true，则所有未映射的值将保持不变。

  `mapType`— 映射列的数据类型。

  `targetColumn`— 要包含结果的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "CATEGORICAL_MAPPING",
        "Parameters": {
            "categoryMap": "{\"United States of America\":\"1\",\"Canada\":\"2\",\"Cuba
\":\"3\",\"Haiti\":\"4\",\"Dominican Republic\":\"5\"}",
            "deleteOtherRows": "false",
            "keepOthers": "true",
            "mapType": "NUMERIC",
            "sourceColumn": "state_name",
            "targetColumn": "state_name_mapped"
        }
    }
}
```

ONE_HOT_ENCODING

创建 n 个数值列，其中 n 是所选类别变量中唯一值的数量。

例如，假设一个名为的列shirt_size。衬衫有小号、中号、大号或超大号可供选择。列数据可能类似如下所示。

```
shirt_size
-----------
L
XL
M
S
M
M
S
XL
M
L
XL
M
```

在这种情况下，有四个不同的值shirt_size。因此，ONE_HOT_ENCODING生成四个新列。每个新列都被命名shirt_size_*x*，其中*x*代表一个不同的shirt_size 值。

的结果shirt_size和生成的四列如下所示。

```
shirt_size    shirt_size_S    shirt_size_M    shirt_size_L    shirt_size_XL
------------    ------------    ------------    ------------    -------------
L               0               0               1               0
XL              0               0               0               1
M               0               1               0               0
S               1               0               0               0
M               0               1               0               0
M               0               1               0               0
S               1               0               0               0
XL              0               0               0               1
M               0               1               0               0
L               0               0               1               0
XL              0               0               0               1
M               0               1               0               0
```

您指定的列最多ONE_HOT_ENCODING可以有十 (10) 个不同的值。

参数

- sourceColumn – 现有列的名称。列最多可以有 10 个不同的值。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ONE_HOT_ENCODING",
        "Parameters": {
            "sourceColumn": "shirt_size"
        }
    }
}
```

SCALE

缩放或标准化数值列中的数据范围。

参数

- sourceColumn – 现有列的名称。
- strategy— 要应用于列值的操作：
  - MIN_MAX— 将值重新缩放到 [0,1] 的范围内
  - SCALE_BETWEEN— 将值重新缩放到 2 个指定值的范围内。
  - MEAN_NORMALIZATION— 在 [-1, 1] 的范围内重新调整数据的平均值 (μ) 为 0，标准差 (σ) 为 1
  - Z_SCORE— 线性缩放数据值，使平均值 (μ) 为 0，标准差 (σ) 为 1。最适合处理异常值。
- targetColumn— 要包含结果的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "NORMALIZATION",
        "Parameters": {
            "sourceColumn": "all_votes",
            "strategy": "MIN_MAX",
            "targetColumn": "all_votes_normalized"
        }
    }
}
```

偏斜

对数据值应用变换以更改分布形状及其偏差。

参数

- sourceColumn – 现有列的名称。

  targetColumn-要创建的新列的名称。

  skewFunction
  - ROOT— 提取值根。可以在value参数中提供根。

    LOG— 记录基值。可以在value参数中提供日志库。

    SQUARE— 方形函数

  value— SkewFunction 的参数。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SKEWNESS",
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "skewFunction": "LOG",
            "value": "2.718281828"
        }
    }
}
```

令牌

将文本拆分为较小的单位或标记，例如单个单词或术语。

参数

- sourceColumn – 现有列的名称。
- delimiter— 出现在标记化单词之间的自定义分隔符。（默认行为是用空格分隔每个令牌。）

- expandContractions— 如果ENABLED，展开缩写词。例如：“不要” 变成 “不要”。
- stemmingMode— 将文本拆分为较小的单位或标记，例如单个小写单词或术语。有两种词干模式可供选择：PORTER| LANCASTER。
- stopWordRemovalMode— 移除 a、an、the 等常用词。
- customStopWords— 对于StopWordRemovalMode，允许您指定自定义停用词列表。
- targetColumn— 要包含结果的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "TOKENIZATION",
        "Parameters": {
            "customStopWords": "[]",
            "delimiter": "- ",
            "expandContractions": "ENABLED",
            "sourceColumn": "dimensions",
            "stemmingMode": "PORTER",
            "stopWordRemovalMode": "DEFAULT",
            "targetColumn": "dimensions_tokenized"
        }
    }

}
```

数学函数

接下来，查找与配方操作配合使用的数学函数的参考主题。

主题

- ABSOLUTE
- ADD
- CEILING
- DEGREES
- 划分
- 指数

- FLOOR
- IS_EVEN
- IS_ODD
- LN
- LOG
- MOD
- 乘
- 否定
- PI
- POWER
- RADIANS
- 随机
- 随机介于
- ROUND
- SIGN
- SQUARE_ROOT
- 减去

ABSOLUTE

返回新列中输入数字的绝对值。绝对值是指数字与零的距离，无论它是正数还是负数

参数

- `sourceColumn` – 现有列的名称。
- `targetColumn`-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ABSOLUTE",
        "Parameters": {
```

```
            "sourceColumn": "freezingTemps",
            "targetColumn": "absValueOfFreezingTemps"
        }
    }
}
```

ADD

使用 (sourceColumn1+sourceColumn2) 或 (sourceColumn1+value1) 对新列中的输入列值求和。

参数

- sourceColumn1 – 现有列的名称。
- value1-数值。
- sourceColumn2 – 现有列的名称。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ADD",
        "Parameters": {
            "sourceColumn1": "weight_kg",
            "sourceColumn2": "height_cm",
            "targetColumn": "weight_plus_height"
        }
    }
}
```

CEILING

返回新列中大于或等于输入十进制数的最小整数。

参数

- sourceColumn – 现有列的名称。
- value1-数值。

- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "CEILING",
        "Parameters": {
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_CEILING"
        }
    }
}
```

DEGREES

将角度的弧度转换为度数，并在新列中返回结果。

参数

- sourceColumn – 现有列的名称。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DEGREES",
        "Parameters": {
            "sourceColumn": "height_cm",
            "targetColumn": "height_cm_DEGREES"
        }
    }
}
```

划分

将一个输入数字除以另一个输入数字，然后在新列中返回结果。

参数

- sourceColumn1 – 现有列的名称。
- value1-数值。
- sourceColumn2 – 现有列的名称。
- value2-数值。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DIVIDE",
        "Parameters": {
            "sourceColumn1": "height_cm",
            "targetColumn": "divide_by_2",
            "value2": "2"
        }
    }
}
```

指数

在新列中返回将欧拉数提高到第 n 个度。

参数

- sourceColumn – 现有列的名称。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "EXPONENT",
        "Parameters": {
            "sourceColumn": "age",
```

```
            "targetColumn": "age_EXPONENT"
        }
    }
}
```

FLOOR

返回新列中大于或等于输入数的最大整数。

参数

- sourceColumn1 – 现有列的名称。
- value-数值。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FLOOR",
        "Parameters": {
            "targetColumn": "FLOOR Column 1",
            "value": "42"
        }
    }
}
```

IS_EVEN

在新列中返回一个布尔值，该值指示源列或值是否为偶数。如果源列或值为十进制，则结果为 false。

参数

- sourceColumn – 现有列的名称。
- targetColumn-要创建的新列的名称。
- trueString— 表示该值是否为偶数的字符串。
- falseString— 表示该值是否为偶数的字符串。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "IS_EVEN",
        "Parameters": {
            "falseString": "Value is odd",
            "sourceColumn": "height_cm",
            "targetColumn": "height_cm_IS_EVEN",
            "trueString": "Value is even"
        }
    }
}
```

IS_ODD

在新列中返回一个布尔值，该值指示源列或值是否为奇数。如果源列或值为十进制，则结果为 false。

参数

- `sourceColumn` – 现有列的名称。
- `targetColumn`-要创建的新列的名称。
- `trueString`— 表示该值是否为奇数的字符串。
- `falseString`— 表示该值是否为奇数的字符串。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "IS_ODD",
        "Parameters": {
            "falseString": "Value is even",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_IS_ODD",
            "trueString": "Value is odd"
        }
    }
}
```

LN

返回新列中值的自然对数（欧拉数）。

参数

- sourceColumn – 现有列的名称。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "LN",
        "Parameters": {
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_LN"
        }
    }
}
```

LOG

返回新列中值的对数。

参数

- sourceColumn – 现有列的名称。
- targetColumn-要创建的新列的名称。
- base— 对数的底。默认值为 10。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "LOG",
        "Parameters": {
            "base": "10",
```

```
            "sourceColumn": "age",
            "targetColumn": "age_LOG"
        }
    }
}
```

MOD

返回新列中一个数字与另一个数字的百分比。

参数

- sourceColumn1 – 现有列的名称。
- sourceColumn2 – 现有列的名称。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MOD",
        "Parameters": {
            "sourceColumn1": "start_date",
            "sourceColumn2": "end_date",
            "targetColumn": "MOD Column 1"
        }
    }
}
```

乘

将两个数字相乘并在新列中返回结果。

参数

- sourceColumn1 – 现有列的名称。
- value1-数值。
- sourceColumn2 – 现有列的名称。
- value2-数值。

- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MULTIPLY",
        "Parameters": {
            "sourceColumn1": "hourly_rate",
            "sourceColumn2": "hours",
            "targetColumn": "total_pay"
        }
    }
}
```

否定

对一个值取反并在新列中返回结果。

参数

- sourceColumn – 现有列的名称。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "NEGATE",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_NEGATE"
        }
    }
}
```

PI

在新列中返回 pi (3.141592653589793) 的值。

参数

- `targetColumn`-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "PI",
        "Parameters": {
            "targetColumn": "PI Column 1"
        }
    }
}
```

POWER

返回数字的乘以新列中指数的乘。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要提高其值的数字。
- `targetColumn`-要创建的新列的名称。
- `exponent`— 价值将提高到的力量。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "POWER",
        "Parameters": {
```

```
            "exponent": "3",
            "sourceColumn": "age",
            "targetColumn": "age_cubed"
        }
    }
}
```

RADIANS

将度数转换为弧度（除以 180/pi），并在新列中返回值。

参数

- `sourceColumn` – 现有列的名称。
- `targetColumn`-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "RADIANS",
        "Parameters": {
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_RADIANS"
        }
    }
}
```

随机

在新列中返回 0 到 1 之间的随机数。

参数

- `targetColumn`-要创建的新列的名称。

Example 示例

```
{
```

```
    "RecipeAction": {
        "Operation": "RANDOM",
        "Parameters": {
            "targetColumn": "RANDOM Column 1"
        }
    }
}
```

随机介于

在新列中，返回一个介于指定下限（含）和指定上限（含）之间的随机数。

参数

- lowerBound— 随机数范围的下限。
- upperBound— 随机数范围的上限。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "RANDOM_BETWEEN",
        "Parameters": {
            "lowerBound": "1",
            "targetColumn": "RANDOM_BETWEEN Column 1",
            "upperBound": "100"
        }
    }
}
```

ROUND

将数值四舍五入到新列中最接近的整数。当分数等于或大于 0.5 时，它会向上舍入。

参数

- sourceColumn – 现有列的名称。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ROUND",
        "Parameters": {
            "sourceColumn": "rating",
            "targetColumn": "rating_ROUND"
        }
    }
}
```

SIGN

返回一个新列，如果值小于 0，则返回 -1；如果值为 0，则返回 0；如果值大于 0，则返回 +1。

参数

- `sourceColumn` – 现有列的名称。
- `targetColumn`-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SIGN",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_SIGN"
        }
    }
}
```

SQUARE_ROOT

返回新列中值的平方根。

参数

- `sourceColumn` – 现有列的名称。

- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SQUARE_ROOT",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_SQUARE_ROOT"
        }
    }
}
```

减去

从另一个数字中减去一个数字，然后在新列中返回结果。

参数

- sourceColumn1 – 现有列的名称。
- value1-数值。
- sourceColumn2 – 现有列的名称。
- value2-数值。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SUBTRACT",
        "Parameters": {
            "sourceColumn1": "weight_kg",
            "targetColumn": "weight_minus_10_kg",
            "value2": "10"
        }
    }
```

```
}
```

聚合函数

接下来，查找与配方操作配合使用的聚合函数的参考主题。

主题

- ANY
- AVERAGE
- COUNT
- 计数_不同
- KTH_MARGEST
- KTH_LARGEST_UNIQUE
- MAX
- MEDIAN
- MIN
- MODE
- 标准偏差
- SUM
- VARIANCE

ANY

在新列中返回选定源列中的任何值。将忽略 NULL 值和 NULL 值。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
```

```
        "Operation": "ANY",
        "Parameters": {
            "sourceColumns": "[\"age\",\"last_name\"]",
            "targetColumn": "ANY Column 1"
        }
    }
}
```

AVERAGE

计算源列中值的平均值，并在新列中返回结果。任何非数字都将被忽略。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "AVERAGE",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "AVERAGE Column 1"
        }
    }
}
```

COUNT

返回新列中选定源列的值的数量。将忽略 NULL 值和 NULL 值。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "COUNT",
        "Parameters": {
            "sourceColumns": "[\"ANY Column 1\",\"birth_date\",\"last_name\"]",
            "targetColumn": "COUNT Column 1"
        }
    }
}
```

计数_不同

返回新列中选定源列中不同值的总数。将忽略 NULL 值和 NULL 值。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "COUNT_DISTINCT",
        "Parameters": {
            "sourceColumns": "[\"long_name\",\"weight_kg\"]",
            "targetColumn": "COUNT_DISTINCT Column 1"
        }
    }
}
```

KTH_MARGEST

返回新列中选定源列的第 k 个最大数字。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

- value— 代表 k 的数字。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "KTH_LARGEST",
        "Parameters": {
            "sourceColumns": "[\"height_cm\",\"weight_kg\",\"age\"]",
            "targetColumn": "KTH_LARGEST Column 1",
            "value": "2"
        }
    }
}
```

KTH_LARGEST_UNIQUE

返回新列中选定源列中第 k 个最大的唯一数字。

参数

- sourceColumns— 一个 JSON 编码的字符串，表示现有列的列表。
- targetColumn-新创建的列的名称。

  value— 代表 k 的数字。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "KTH_LARGEST_UNIQUE",
        "Parameters": {
            "sourceColumns": "[\"age\",\"height_cm\",\"weight_kg\"]",
            "targetColumn": "KTH_LARGEST_UNIQUE Column 1",
            "value": "3"
        }
    }
}
```

MAX

返回新列中选定源列的最大数值。任何非数字都将被忽略。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MAX",
        "Parameters": {
            "sourceColumns": "[\"age\",\"height_cm\",\"weight_kg\"]",
            "targetColumn": "MAX Column 1"
        }
    }
}
```

MEDIAN

返回新列中选定源列的中位数，即一组已排序的数字的中间数。任何非数字都将被忽略。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MEDIAN",
        "Parameters": {
            "sourceColumns": "[\"age\",\"years_in_service\"]",
            "targetColumn": "MEDIAN Column 1"
```

```
        }
    }
}
```

MIN

返回新列中选定源列的最小值。任何非数字都将被忽略。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MIN",
        "Parameters": {
            "sourceColumns": "[\"age\",\"height_cm\",\"weight_kg\"]",
            "targetColumn": "MIN Column 1"
        }
    }
}
```

MODE

从新列中的选定源列中返回模式，即最常出现的数字。任何非数字都将被忽略。对于多种模式，使用模态函数计算模式。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
```

```
    "RecipeAction": {
        "Operation": "MODE",
        "Parameters": {
            "modeType": "MINIMUM",
            "sourceColumns": "[\"years_in_service\",\"age\"]",
            "targetColumn": "MODE Column 1"
        }
    }
}
```

标准偏差

返回新列中选定源列的标准差。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "STANDARD_DEVIATION",
        "Parameters": {
            "sourceColumns": "[\"years_in_sservice\",\"age\"]",
            "targetColumn": "STANDARD_DEVIATION Column 1"
        }
    }
}
```

SUM

返回新列中选定源列的值的总和。任何非数字都被视为 0。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SUM",
        "Parameters": {
            "sourceColumns": "[\"age\",\"years_in_service\"]",
            "targetColumn": "SUM Column 1"
        }
    }
}
```

VARIANCE

返回新列中选定源列的方差。方差定义为`Var(X) = [Sum ((X – mean(X))^2)]/Count(X)`。

参数

- `sourceColumns`— 一个 JSON 编码的字符串，表示现有列的列表。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "VARIANCE",
        "Parameters": {
            "sourceColumns": "[\"age\",\"years_in_service\"]",
            "targetColumn": "VARIANCE Column 1"
        }
    }
}
```

文本函数

接下来，查找与配方操作配合使用的文本函数的参考主题。

主题

CHAR

在新列中返回源列中每个整数的 Unicode 字符，或者返回自定义整数值的 Unicode 字符。

参数

- `sourceColumn` – 现有列的名称。

- value— 一个表示 Unicode 值的整数。
- targetColumn-要创建的新列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "CHAR",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_char"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "CHAR",
        "Parameters": {
            "value": 42,
            "targetColumn": "asterisk"
        }
    }
}
```

ENDS_WITH

如果指定数量的最右边的字符或自定义字符串与模式匹配，则在新列中返回true。

参数

- sourceColumn – 现有列的名称。

- value— 要计算的字符串。
- pattern— 必须匹配字符串结尾的正则表达式。
- targetColumn-要创建的新列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "ENDS_WITH",
        "Parameters": {
            "sourceColumn": "nationality",
            "pattern": "[Ss]",
            "targetColumn": "nationality_ends_with"
        }
    }
}
```

精确的

创建填充以下内容之一的新列：

- True如果一列（或值）中的一个字符串与另一列（或值）中的另一个字符串完全匹配。
- False如果没有匹配项。

参数

- sourceColumn1 – 现有列的名称。
- sourceColumn2 – 现有列的名称。
- value1— 要计算的字符串。
- value2— 要计算的字符串。
- targetColumn-要创建的新列的名称。

> **Note**
>
> 您只能指定下列组合之一：
>
> - 两者兼而有之sourceColumn*N*。
> - 其中之一sourceColumn*N*和一个value*N*.
> - 两者兼而有之value*N*。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "EXACT",
        "Parameters": {
            "sourceColumn1": "nationality",
            "value2": "Argentina",
            "targetColumn": "nationality_exact"
        }
    }
}
```

调查发现

从左向右搜索，从源列或自定义值中查找与指定字符串相匹配的字符串，然后在新列中返回结果。

参数

- sourceColumn – 现有列的名称。
- pattern— 要搜索的正则表达式。
- position— 从字符串左端开始的字符位置。
- ignoreCase— 如果true，则忽略字母之间的大小写差异（大写和小写之间）。要强制严格匹配，请false改用。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "FIND",
        "Parameters": {
            "sourceColumn": "city",
            "pattern": "[AEIOU]",
            "position": "1",
            "ignoreCase": "false",
            "targetColumn": "begins_with_a_vowel"
        }
    }
}
```

LEFT

给定一定数量的字符，从源列或自定义字符串中取出字符串中最左边的字符数，然后返回新列中指定数量的最左边的字符。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `position`— 从字符串左端开始的字符位置。
- `targetColumn`-要创建的新列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "LEFT",
        "Parameters": {
            "position": "3",
            "sourceColumn": "city",
```

```
            "targetColumn": "city_left"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "LEFT",
        "Parameters": {
            "position": "5",
            "value": "How now brown cow",
            "targetColumn": "how_now_5_left_chars"
        }
    }
}
```

LEN

在新列中返回源列中的字符串长度或自定义字符串的长度。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-要创建的新列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "LEN",
        "Parameters": {
```

```
            "sourceColumn": "last_name",
            "targetColumn": "last_name_len"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "LEN",
        "Parameters": {
            "value": "Hello",
            "targetColumn": "hello_len"
        }
    }
}
```

LOWER

将源列中的字符串或自定义字符串中的所有字母字符转换为小写，并在新列中返回结果。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-要创建的新列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "LOWER",
        "Parameters": {
```

```
            "sourceColumn": "last_name",
            "targetColumn": "last_name_lower"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "LOWER",
        "Parameters": {
            "value": "GOODBYE",
            "targetColumn": "goodbye_lower"
        }
    }
}
```

合并列和值

连接源列中的字符串，并在新列中返回结果。可以在合并后的值之间插入分隔符。

参数

- `sourceColumns`— 两个或多个现有列的名称，采用 JSON 编码格式。
- `delimiter`：可选。要在每两个源列值之间放置一个或多个字符。
- `targetColumn`-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MERGE_COLUMNS_AND_VALUES",
        "Parameters": {
            "sourceColumns": "[\"last_name\",\"birth_date\"]",
            "delimiter": " was born on: ",
            "targetColumn": "merged_column"
        }
    }
}
```

正确的

将源列中的字符串或自定义值中的所有字母字符转换为正确的大小写，并在新列中返回结果。

在适当情况下，也称为大写字母，每个单词的第一个字母大写，其余单词转换为小写。一个例子是：Quick Brown Fox 跳过栅栏

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-要创建的新列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "PROPER",
        "Parameters": {
            "sourceColumn": "first_name",
            "targetColumn": "first_name_proper"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "PROPER",
        "Parameters": {
            "value": "MR. H. SMITH, ESQ.",
            "targetColumn": "formal_name_proper"
        }
    }
```

```
}
```

移除符号

从源列或自定义字符串中的字符串中删除非字母、数字、带重音拉丁字符或空格的字符，并在新列中返回结果。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-要创建的新列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_SYMBOLS",
        "Parameters": {
            "sourceColumn": "info_url",
            "targetColumn": "info_url_remove_symbols"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_SYMBOLS",
        "Parameters": {
            "value": "$&#$&HEY!#@@",
            "targetColumn": "without_symbols"
        }
```

```
    }
}
```

移除空白

从源列或自定义字符串中的字符串中删除空白，并在新列中返回结果。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-要创建的新列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REMOVE_WHITESPACE",
        "Parameters": {
            "sourceColumn": "job_desc",
            "targetColumn": "job_desc_remove_whitespace"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_WHITESPACE",
        "Parameters": {
            "value": "This string has spaces in it",
            "targetColumn": "string_without_spaces"
        }
    }
```

```
}
```

重复字符串

将源列或自定义输入值中的字符串重复指定次数，然后在新列中返回结果。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `count`— 字符串重复次数。
- `targetColumn`-要创建的新列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "REPEAT_STRING",
        "Parameters": {
            "count": 3,
            "sourceColumn": "last_name",
            "targetColumn": "last_name_repeat_string"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REPEAT_STRING",
        "Parameters": {
            "count": 80,
            "value": "*",
            "targetColumn": "80_stars"
```

```
        }
    }
}
```

RIGHT

给定一定数量的字符，从源列或自定义字符串中提取字符串中最右边的字符数，然后返回新列中指定数量的最右边的字符。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `position`— 从字符串右侧开始的字符位置。
- `targetColumn`-要创建的新列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "RIGHT",
        "Parameters": {
            "sourceColumn": "nationality",
            "position": "3",
            "targetColumn": "nationality_right"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "RIGHT",
```

```
        "Parameters": {
            "value": "United States of America",
            "position": "7",
            "targetColumn": "usa_right"
        }
    }
}
```

正确查找

从右向左搜索，从源列或自定义值中查找与指定字符串相匹配的字符串，然后在新列中返回结果。

参数

- sourceColumn – 现有列的名称。
- pattern— 要搜索的正则表达式。
- position— 从字符串右端开始的字符位置。
- ignoreCase— 如果true，则忽略字母之间的大小写差异（大写和小写之间）。要强制严格匹配，请false改用。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "RIGHT_FIND",
        "Parameters": {
            "sourceColumn": "nationality",
            "pattern": "s",
            "position": "1",
            "ignoreCase": "true",
            "targetColumn": "ends_with_an_s"
        }
    }
}
```

STARTS_WITH

如果指定数量的最左边字符或自定义字符串与模式匹配，则在新列中返回true。

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。
- pattern— 必须与字符串开头匹配的正则表达式。
- targetColumn-要创建的新列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "STARTS_WITH",
        "Parameters": {
            "sourceColumn": "nationality",
            "pattern": "[AEIOU]",
            "targetColumn": "nationality_starts_with"
        }
    }
}
```

STRING_GREATER_THAN

创建填充以下内容之一的新列：

- True如果一列（或值）中的一个字符串大于另一列（或值）中的另一个字符串。
- False如果没有匹配项。

参数

- sourceColumn1 – 现有列的名称。
- sourceColumn2 – 现有列的名称。

- value1— 要计算的字符串。
- value2— 要计算的字符串。
- targetColumn-要创建的新列的名称。

> **Note**
>
> 您只能指定下列组合之一：
>
> - 两者兼而有之sourceColumn*N*。
> - 其中之一sourceColumn*N*和一个value*N*.
> - 两者兼而有之value*N*。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "STRING_GREATER_THAN",
        "Parameters": {
            "sourceColumn1": "first_name",
            "sourceColumn2": "last_name",
            "targetColumn": "string_greater_than"
        }
    }
}
```

字符串_大于_等于

创建填充以下内容之一的新列：

- True如果一列（或值）中的一个字符串大于或等于另一列（或值）中的另一个字符串。
- False如果没有匹配项。

参数

- sourceColumn1 – 现有列的名称。
- sourceColumn2 – 现有列的名称。

- value1— 要计算的字符串。
- value2— 要计算的字符串。
- targetColumn-要创建的新列的名称。

> **Note**
>
> 您只能指定下列组合之一：
>
> - 两者兼而有之sourceColumn*N*。
> - 其中之一sourceColumn*N*和一个value*N*.
> - 两者兼而有之value*N*。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "STRING_GREATER_THAN_EQUAL",
        "Parameters": {
            "sourceColumn1": "nationality",
            "targetColumn": "string_greater_than_equal",
            "value2": "s"
        }
    }
}
```

字符串_小于

创建填充以下内容之一的新列：

- True如果一列（或值）中的一个字符串小于另一列（或值）中的另一个字符串。
- False如果没有匹配项。

参数

- sourceColumn1 – 现有列的名称。
- sourceColumn2 – 现有列的名称。

- value1— 要计算的字符串。
- value2— 要计算的字符串。
- targetColumn-要创建的新列的名称。

> **Note**
>
> 您只能指定下列组合之一：
>
> - 两者兼而有之sourceColumn*N*。
> - 其中之一sourceColumn*N*和一个value*N*.
> - 两者兼而有之value*N*。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "STRING_LESS_THAN",
        "Parameters": {
            "sourceColumn1": "first_name",
            "sourceColumn2": "last_name",
            "targetColumn": "string_less_than"
        }
    }
}
```

字符串_小于_等于

创建填充以下内容之一的新列：

- True如果一列（或值）中的一个字符串小于或等于另一列（或值）中的另一个字符串。
- False如果没有匹配项。

参数

- sourceColumn1 – 现有列的名称。
- sourceColumn2 – 现有列的名称。

- value1— 要计算的字符串。
- value2— 要计算的字符串。
- targetColumn-要创建的新列的名称。

> **Note**
>
> 您只能指定下列组合之一：
>
> - 两者兼而有之sourceColumn*N*。
> - 其中之一sourceColumn*N*和一个value*N*.
> - 两者兼而有之value*N*。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "STRING_LESS_THAN_EQUAL",
        "Parameters": {
            "sourceColumn1": "first_name",
            "targetColumn": "string_less_than_equal",
            "value2": "s"
        }
    }
}
```

SUBSTRING

根据用户定义的起始和结束索引值，在新列中返回源列中的部分或全部指定字符串。

参数

- sourceColumn – 现有列的名称。
- startPosition— 从字符串左端开始的字符位置。
- endPosition— 从字符串左端开始的字符结尾位置。
- targetColumn-要创建的新列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SUBSTRING",
        "Parameters": {
            "sourceColumn": "last_name",
            "startPosition": "5",
            "endPosition": "8",
            "targetColumn": "chars_5_through_8"
        }
    }
}
```

TRIM

删除源列或自定义字符串中字符串的前导和尾随空格，并在新列中返回结果。单词之间的空格不会被删除。

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。
- targetColumn-要创建的新列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
```

```
    "RecipeAction": {
        "Operation": "TRIM",
        "Parameters": {
            "sourceColumn": "nationality",
            "targetColumn": "nationality_trim"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "TRIM",
        "Parameters": {
            "value": "   This string should be trimmed       ",
            "targetColumn": "string_trimmed"
        }
    }
}
```

UNICODE

在新列中返回源列中字符串的第一个字符或自定义字符串的 Unicode 索引值。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-要创建的新列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
```

```
        "Operation": "UNICODE",
        "Parameters": {
            "sourceColumn": "first_name",
            "targetColumn": "first_name_unicode"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "UNICODE",
        "Parameters": {
            "value": "?",
            "targetColumn": "sixty_three"
        }
    }
}
```

UPPER

将源列中的字符串或自定义字符串中的所有字母字符转换为大写，并在新列中返回结果。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-要创建的新列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
```

```
        "Operation": "UPPER",
        "Parameters": {
            "sourceColumn": "last_name",
            "targetColumn": "last_name_upper"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "UPPER",
        "Parameters": {
            "value": "a string of lowercase letters",
            "targetColumn": "string_upper"
        }
    }
}
```

日期和时间函数

接下来，查找与配方操作配合使用的日期和时间函数的参考主题。

主题

- CONVERT_TIMEZONE
- DATE
- DATE_ADD
- DATE_DIFF
- DATE_FORMAT
- DATE_TIME
- DAY
- HOUR
- MILLISECOND
- MINUTE
- MONTH
- 月份_名称

- NOW
- 25美分硬币
- SECOND
- TIME
- 今天
- UNIX_TIME
- UNIX_TIME_FORMAT
- 周_日
- 星期_NUMBER
- YEAR

CONVERT_TIMEZONE

根据指定的时区将源列中的时间值转换为新列。

参数

- `sourceColumn` – 现有列的名称。源列的类型可以是`stringdate`、或`timestamp`。
- `fromTimeZone`— 源值时区。如果未指定，默认时区为 UTC。
- `toTimeZone`— 要转换到的时区。如果未指定，默认时区为 UTC。
- `targetColumn`-新创建的列的名称。
- `dateTimeFormat`：可选。日期的格式字符串。如果未指定，则默认格式为：`yyyy-mm-dd HH:MM:SS`。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "CONVERT_TIMEZONE",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "fromTimeZone": "UTC+08:00",
            "toTimeZone": "UTC+08:00",
            "targetColumn": "DATETIME Column CONVERT_TIMEZONE",
```

```
            "dateTimeFormat": "yyyy-mm-dd HH:MM:SS"
        }
    }
}
```

DATE

根据源列或提供的值创建一个包含日期值的新列。

参数

- dateTimeFormat：可选。日期的格式字符串，因为它将出现在新列中。如果未指定，则默认格式为yyyy-mm-dd HH:MM:SS。
- dateTimeParameters— 一个 JSON 编码的字符串，表示日期和时间的组成部分：
  - year
  - value
  - month
  - day
  - hour
  - second

  每个组件都必须指定以下各项之一：
  - sourceColumn – 现有列的名称。
  - value— 要计算的字符串。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DATE",
        "Parameters": {
            "dateTimeFormat": "mm/dd/yy",
            "dateTimeParameters": "{\"year\":{\"value\":\"2019\"},\"month\":{\"value\":
\"12\"},\"day\":{\"value\":\"31\"},\"hour\":{},\"minute\":{},\"second\":{}}",
            "targetColumn": "DATE Column 1"
        }
```

```
    }
}
```

DATE_ADD

在源列或值中的日期中添加年、月或日，然后创建包含结果的新列。

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。
- units— 用于调整日期的计量单位。有效值为MONTHS、YEARS、MILLISECONDS、QUARTERS、HOURS、MICROSECONDS、WEEKS、SECONDS、DAYS和MINUTES。
- dateAddValue— units 要添加到日期的数字。
- dateTimeFormat：可选。日期的格式字符串，因为它将出现在新列中。如未指定，则默认格式为 yyyy-mm-dd HH:MM:SS。
- targetColumn-新创建的列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DATE_ADD",
        "Parameters": {
            "sourceColumn": "DATE Column 1",
            "units": "DAYS",
            "dateAddValue": "14",
            "dateTimeFormat": "mm/dd/yyyy",
            "targetColumn": "DATE Column 1_DATEADD"
        }
    }
```

```
}
```

DATE_DIFF

创建包含两个日期字段相差的新列。

参数

- sourceColumn1 – 现有列的名称。
- sourceColumn2 – 现有列的名称。
- value1— 要计算的字符串。
- value2— 要计算的字符串。
- units— 描述日期之间差异的计量单位。有效值为MONTHS、YEARS、MILLISECONDS、QUARTERS、HOURS、MICROSECONDS、WEEKS、SECONDS、DAYS和MINUTES。
- targetColumn-新创建的列的名称。

> **Note**
>
> 您只能指定以下组合之一：
>
> - 两者兼sourceColumn1而有之sourceColumn2。
> - sourceColumn1或之一sourceColumn2和value1或之一value2。
> - 两者兼value1而有之value2。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DATE_DIFF",
        "Parameters": {
            "value1": "2020-01-01",
            "value2": "2020-10-06",
            "units": "DAYS",
            "targetColumn": "DATEDIFF Column 1"
        }
```

```
    }
}
```

DATE_FORMAT

使用代表日期的字符串创建包含特定格式的日期的新列。

参数

- sourceColumn – 现有列的名称。
- value— 要评估的字符串。
- dateTimeFormat：可选。日期的格式字符串，因为它将出现在新列中。如未指定，则默认格式为 yyyy-mm-dd HH:MM:SS。
- targetColumn-新创建的列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DATE_FORMAT",
        "Parameters": {
            "sourceColumn": "DATE Column 1",
            "dateTimeFormat": "month*dd*yyyy",
            "targetColumn": "DATE Column 1_DATEFORMAT"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "DATE_FORMAT",
        "Parameters": {
            "value": "22:10:47",
```

```
            "dateTimeFormat": "HH:MM:SS",
            "targetColumn": "formatted_date_value"
        }
    }
}
```

DATE_TIME

根据源列或提供的值创建一个包含日期和时间值的新列。

参数

- dateTimeFormat：可选。日期的格式字符串，因为它将出现在新列中。如果未指定，则默认格式为yyyy-mm-dd HH:MM:SS。
- dateTimeParameters— 一个 JSON 编码的字符串，表示日期和时间的组成部分：
  - year
  - value
  - month
  - day
  - hour
  - second

  每个组件都必须指定以下各项之一：
  - sourceColumn – 现有列的名称。
  - value— 要计算的字符串。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DATE_TIME",
        "Parameters": {
            "dateTimeFormat": "yyyy-mm-dd HH:MM:SS",
            "dateTimeParameters": "{\"year\":{\"value\":\"2010\"},\"month\":{\"value\":
\"5\"},\"day\":{\"value\":\"21\"},\"hour\":{\"value\":\"13\"},\"minute\":{\"value\":
\"34\"},\"second\":{\"value\":\"25\"}}",
            "targetColumn": "DATETIME Column 1"
```

```
        }
    }
}
```

DAY

使用表示日期的字符串创建包含月份中的某一天的新列。

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。
- targetColumn-新创建的列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "DAY",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_DAY"
        }
    }
}
```

HOUR

使用表示日期的字符串创建包含小时值的新列。

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。

- `targetColumn`-新创建的列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "HOUR",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_HOUR"
        }
    }
}
```

MILLISECOND

创建一个包含源列或输入值中的毫秒值的新列。

参数

- `sourceColumn` – 现有列的名称。源列的类型可以是`stringdate`、或`timestamp`。
- `value`— 要计算的字符串。
- `targetColumn`-新创建的列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
```

```
    "RecipeAction": {
        "Operation": "MILLISECOND",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_MILLISECOND"
        }
    }
}
```

MINUTE

使用表示日期的字符串创建包含分钟值的新列。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-新创建的列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MINUTE",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_MINUTE"
        }
    }
}
```

MONTH

使用表示日期的字符串创建包含月份数的新列。

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。
- targetColumn-新创建的列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MONTH",
        "Parameters": {
            "value": "2018-05-27",
            "targetColumn": "MONTH Column 1"
        }
    }
}
```

月份_名称

使用表示日期的字符串创建包含月份名称的新列。

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。
- targetColumn-新创建的列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "MONTH_NAME",
        "Parameters": {
            "value": "2018-05-27",
            "targetColumn": "MONTHNAME Column 1"
        }
    }
}
```

NOW

创建包含当前日期和时间格式的新列yyyy-mm-dd HH:MM:SS。

参数

- timeZone— 时区的名称。如果未指定时区，则默认值为 UTC。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "NOW",
        "Parameters": {
            "timeZone": "US/Pacific",
            "targetColumn": "NOW Column 1"
        }
}
```

25美分硬币

使用表示日期的字符串创建包含基于日期的季度的新列。

> **Note**
>
> 在新列中将季度指定为 1、2、3 或者 4。

- 1 表示一月、二月和三月。
- 2 表示四月、五月和六月。
- 3 表示七月、八月和九月。
- 4 表示十月、十一月和十二月。

参数

- `sourceColumn` – 现有列的名称。源列的类型可以是`stringdate`、或`timestamp`。
- `value`— 要计算的字符串。
- `targetColumn`-新创建的列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "QUARTER",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_QUARTER"
        }
    }
}
```

SECOND

使用表示日期的字符串创建包含第二个值的新列。

参数

- `sourceColumn` – 现有列的名称。

- value— 要计算的字符串。
- targetColumn-新创建的列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "SECOND",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_SECOND"
        }
    }
}
```

TIME

根据提供的源列或值创建一个包含时间值的新列。

参数

- dateTimeFormat：可选。日期的格式字符串，因为它将出现在新列中。如果未指定，则默认格式为yyyy-mm-dd HH:MM:SS。
- dateTimeParameters— 一个 JSON 编码的字符串，表示日期和时间的组成部分：
  - year
  - value
  - month
  - day
  - hour
  - second

  每个组件都必须指定以下各项之一：

  - sourceColumn – 现有列的名称。
  - value— 要计算的字符串。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "TIME",
        "Parameters": {
            "dateTimeFormat": "HH:MM:SS",
            "dateTimeParameters": "{\"year\":{},\"month\":{},\"day\":{},\"hour\":
{\"sourceColumn\":\"rand_hour\"},\"minute\":{\"sourceColumn\":\"rand_minute\"},\"second
\":{\"sourceColumn\":\"rand_second\"}}",
            "targetColumn": "TIME Column 1"
        }
    }
}
```

今天

以该格式创建包含当前日期的新列yyyy-mm-dd。

参数

- timeZone— 时区的名称。如果未指定时区，则默认值为 UTC。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "TODAY",
        "Parameters": {
            "timeZone": "US/Pacific",
            "targetColumn": "TODAY Column 1"
        }
```

```
    }
}
```

UNIX_TIME

根据源列或输入值创建一个新列，其中包含一个表示纪元时间（Unix 时间）（自 1970 年 1 月 1 日以来的秒数）的数字。如果可以推断出时区，则输出为该时区。否则，输出采用通用协调时间 (UTC)。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-新创建的列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "UNIX_TIME",
        "Parameters": {
            "sourceColumn": "TIME Column 1",
            "targetColumn": "TIME Column 1_UNIXTIME"
        }
    }
}
```

UNIX_TIME_FORMAT

将源列或输入值的 Unix 时间转换为指定的数字日期格式，并在新列中返回结果。

参数

- `sourceColumn` – 现有列的名称。

- value— 一个表示 Unix 纪元时间戳的整数。
- dateTimeFormat：可选。日期的格式字符串，因为它将出现在新列中。如未指定，则默认格式为 yyyy-mm-dd HH:MM:SS。
- targetColumn-新创建的列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "UNIX_TIME_FORMAT",
        "Parameters": {
            "value": "1601936554",
            "dateTimeFormat": "yyyy-mm-dd HH:MM:SS",
            "targetColumn": "UNIXTIMEFORMAT Column 1"
        }
    }
}
```

周_日

使用代表日期的字符串创建包含星期的新列。

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。
- targetColumn-新创建的列的名称。

> **Note**
>
> 您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "WEEK_DAY",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_WEEKDAY"
        }
    }
}
```

星期_NUMBER

使用表示日期的字符串创建包含星期数（从 1 到 52）的新列。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-新创建的列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "WEEK_NUMBER",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_WEEK_NUMBER"
        }
    }
}
```

YEAR

使用代表日期的字符串创建包含年份的新列。

参数

- `sourceColumn` – 现有列的名称。
- `value`— 要计算的字符串。
- `targetColumn`-新创建的列的名称。

> **Note**
>
> 您可以指定 `sourceColumn` 或 `value`，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "YEAR",
        "Parameters": {
            "value": "2019-06-12",
            "targetColumn": "YEAR Column 1"
        }
    }
}
```

窗口函数

接下来，查找与配方操作配合使用的窗口函数的参考主题。

主题

- FILL
- NEXT
- 上一页
- 滚动平均值

- ROLLING_COUNT_A
- ROLLING_KTH_LARGEST
- ROLLING_KTH_LARGEST_UNIQUE
- ROLLING_MAX
- ROLLING_MIN
- 滚动模式
- 滚动标准偏差
- ROLLING_SUM
- 滚动方差
- ROW_NUMBER
- SESSION

FILL

根据指定的源列返回一个新列。对于源列中的任何缺失值或空值，请从相关源值前后的行窗口FILL中选择最新的非空值。然后将所选值放到新列中。

参数

- sourceColumn – 现有列的名称。
- numRowsBefore— 当前源行之前的行数，表示窗口的开始。
- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "FILL",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "last_name",
```

```
            "targetColumn": "last_name_FILL"
        }
    }
}
```

NEXT

返回一个新列，其中每个值代表一个位于源列中 n 行之后的值。

参数

- sourceColumn – 现有列的名称。
- numRows— 表示源列前面的 n 行的值。例如，如果numRows为 3，则NEXT使用下一个第三个sourceColumn值作为新targetColumn值。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "NEXT",
        "Parameters": {
            "numRows": "1",
            "sourceColumn": "age",
            "targetColumn": "age_NEXT"
        }
    }
}
```

上一页

返回一个新列，其中每个值代表一个位于源列前 n 行的值。

参数

- sourceColumn – 现有列的名称。
- numRows— 表示源列前面的 n 行的值。例如，如果numRows为 3，则PREV使用前面的第三个sourceColumn值作为新targetColumn值。

- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "PREV",
        "Parameters": {
            "numRows": "1",
            "sourceColumn": "age",
            "targetColumn": "age_PREV"
        }
    }
}
```

滚动平均值

在新列中返回指定列中指定行数之前到当前行之后的指定行数的值的累积平均值。

参数

- sourceColumn – 现有列的名称。
- numRowsBefore— 当前源行之前的行数，表示窗口的开始。
- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "ROLLING_AVERAGE",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_AVERAGE"
        }
    }
```

```
}
```

ROLLING_COUNT_A

在新列中返回从指定列中指定行数之前到当前行之后的指定行数的非空值的滚动计数。

参数

- sourceColumn – 现有列的名称。
- numRowsBefore— 当前源行之前的行数，表示窗口的开始。
- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "ROLLING_COUNT_A",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_COUNT_A"
        }
    }
}
```

ROLLING_KTH_LARGEST

在新列中返回从指定列中指定行数到当前行之后的指定行数的滚动第 k 个最大值。

参数

- sourceColumn – 现有列的名称。
- numRowsBefore— 当前源行之前的行数，表示窗口的开始。
- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- value-k 的值。
- targetColumn-新创建的列的名称。

Example 示例

```
{
  "Action": {
    "Operation": "ROLLING_KTH_LARGEST",
    "Parameters": {
      "sourceColumn": "weight_kg",
      "numRowsBefore": "5",
      "numRowsAfter": "5",
      "value": "3"
      "targetColumn": "weight_kg_ROLLING_KTH_LARGEST"
    }
  }
}
```

ROLLING_KTH_LARGEST_UNIQUE

在新列中返回从指定列中指定行数到指定列中当前行之后的指定行数的滚动唯一第 k 个最大值。

参数

- sourceColumn – 现有列的名称。
- numRowsBefore— 当前源行之前的行数，表示窗口的开始。
- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- value-k 的值。
- targetColumn-新创建的列的名称。

Example 示例

```
{
  "Action": {
    "Operation": "ROLLING_KTH_LARGEST_UNIQUE",
    "Parameters": {
      "sourceColumn": "games_played",
      "numRowsBefore": "3",
      "numRowsAfter": "3",
      "value": "5",
      "targetColumn": "weight_kg_ROLLING_KTH_LARGEST_UNIQUE"
```

```
      }
    }
  }
```

ROLLING_MAX

在新列中返回指定列中从指定行数之前的指定行数到当前行之后的指定行数的累积最大值。

参数

- sourceColumn – 现有列的名称。

  numRowsBefore— 当前源行之前的行数，表示窗口的开始。
- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "ROLLING_MAX",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_MAX"
        }
    }
}
```

ROLLING_MIN

在新列中返回指定列中从指定行数之前到当前行之后的指定行数的滚动最小值。

参数

- sourceColumn – 现有列的名称。

  numRowsBefore— 当前源行之前的行数，表示窗口的开始。

- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "ROLLING_MIN",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_MIN"
        }
    }
}
```

滚动模式

在新列中将滚动模式（最常见的值）从指定列中的指定行数返回到指定列中当前行之后的指定行数。

参数

- sourceColumn – 现有列的名称。
- numRowsBefore— 当前源行之前的行数，表示窗口的开始。
- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- ModeType — 要应用于窗口的模态函数。有效值为 NONE、MINIMUM、MAXIMUM 和 AVERAGE。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "ROLLING_MODE",
        "Parameters": {
            "modeType": "MINIMUM",
            "numRowsAfter": "10",
```

```
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_MODE"
        }
    }
}
```

滚动标准偏差

在新列中返回指定列中从指定行数之前的指定行数到当前行之后的指定行数的值的滚动标准差。

参数

- sourceColumn – 现有列的名称。
- numRowsBefore— 当前源行之前的行数，表示窗口的开始。
- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "ROLLING_STDEV",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_STDEV"
        }
    }
}
```

ROLLING_SUM

在新列中返回指定列中指定行数之前到当前行之后的指定行数的值的累积总和。

参数

- sourceColumn – 现有列的名称。

numRowsBefore— 当前源行之前的行数，表示窗口的开始。

- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "ROLLING_SUM",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_SUM"
        }
    }
}
```

滚动方差

在新列中返回指定列中从指定行数之前的指定行数到当前行之后的指定行数的值的滚动方差。

参数

- sourceColumn – 现有列的名称。
- numRowsBefore— 当前源行之前的行数，表示窗口的开始。
- numRowsAfter— 当前源行之后的行数，表示窗口的结尾。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "ROLLING_VAR",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
```

```
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_VAR"
        }
    }
}
```

ROW_NUMBER

根据由 “分组依据” 和 “排序依据” 语句中的列名创建的窗口，在新列中返回会话标识符。

参数

- groupByColumns— 描述 “分组依据” 列的 JSON 编码字符串。
- orderByColumns— 描述 “排序依据” 列的 JSON 编码字符串。
- targetColumn-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "ROW_NUMBER",
        "Parameters": {
            "groupByColumns": "[\"is public domain\"]",
            "orderByColumns": "[\"dimensions\"]",
            "targetColumn": "Row number"
        }
    }
}
```

SESSION

根据由 “分组依据” 和 “排序依据” 语句中的列名创建的窗口，在新列中返回会话标识符。

参数

- sourceColumn – 现有列的名称。
- units— 描述会话时长的计量单位。有效值为MONTHS、YEARS、MILLISECONDS、QUARTERS、HOURS、MICROSECONDS、WEEKS、SECONDS、DAYS和MINUTES。

- `value`— 用于定义时间段`units`的数字。
- `groupByColumns`— 描述 “分组依据” 列的 JSON 编码字符串。
- `orderByColumns`— 描述 “排序依据” 列的 JSON 编码字符串。
- `targetColumn`-新创建的列的名称。

Example 示例

```
{
    "Action": {
        "Operation": "SESSION",
        "Parameters": {
            "sourceColumn": "object number",
            "units": "MINUTES",
            "value": "10",
            "groupByColumns": "[\"is public domain\"]",
            "orderByColumns": "[\"dimensions\"]",
            "targetColumn": "object number_SESSION",
        }
    }
}
```

网络函数

接下来，查找与配方操作配合使用的 Web 函数的参考主题。

主题

- IP_TO_INT
- INT_TO_IP
- URL_PARAMS

IP_TO_INT

将源列或其他值的 Internet 协议版本 4 (IPv4) 值转换为目标列中相应的整数值，并在新列中返回结果。此功能仅适用于 IPv4 地址。

例如，请考虑以下 IP 地址。

```
192.168.1.1
```

如果您使用此值作为的输入IP_TO_INT，则输出值如下所示。

```
3232235777
```

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。
- targetColumn-要创建的新列的名称。

您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "IP_TO_INT",
        "Parameters": {
            "sourceColumn": "my_ip_address",
            "targetColumn": "IP_TO_INT Column 1"
        }
    }
}
```

INT_TO_IP

将源列或其他值的整数值转换为目标列中相应的 IPv4 值，并在新列中返回结果。此功能仅适用于 IPv4 地址。

例如，请考虑以下整数。

```
167772410
```

如果您使用此值作为的输入INT_TO_IP，则输出值如下所示。

```
10.0.0.250
```

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。
- targetColumn-要创建的新列的名称。

您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
[  {
    "RecipeAction": {
      "Operation": "INT_TO_IP",
      "Parameters": {
        "sourceColumn": "my_integer",
        "targetColumn": "INT_TO_IP Column 1"
      }
    }
  }
]
```

URL_PARAMS

从 URL 字符串中提取查询参数，将其格式化为 JSON 对象，然后在新列中返回结果。

例如，请考虑以下 URL 地址。

```
https://example.com/?firstParam=answer&secondParam=42
```

如果您使用此值作为的输入URL_PARAMS，则输出值如下所示。

```
{"firstParam": ["answer"], "secondParam": ["42"]}
```

参数

- sourceColumn – 现有列的名称。
- value— 要计算的字符串。
- targetColumn-要创建的新列的名称。

您可以指定 sourceColumn 或 value，但不能同时指定两者。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "URL_PARAMS",
        "Parameters": {
            "sourceColumn": "my_url",
            "targetColumn": "URL_PARAMS Column 1"
        }
    }
}
```

其他函数

接下来，查找与配方操作配合使用的其他函数的参考主题。

主题

- COALESCE
- 获取操作结果
- GET_STEP_DATAFRAME

COALESCE

在新列中返回在列数组中找到的第一个非空值。函数中列出的顺序决定了它们的搜索结果的顺序。

参数

- sourceColumns— 一个 JSON 编码的字符串，表示现有列的列表。
- targetColumn-要创建的新列的名称。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "COALESCE",
```

```
        "Parameters": {
            "sourceColumns": "[\"nation_position\",\"joined\"]",
            "targetColumn": "COALESCE Column 1"
        }
    }
}
```

获取操作结果

获取先前提交的操作的结果。仅用于交互式体验。

参数

- `actionId`— 在原始 SendProjectSessionAction 响应中 ActionId 返回的。

Example 示例

```
{
    "RecipeAction": {
        "Operation": "GET_ACTION_RESULT",
        "Parameters": {
            "actionId": "7",
        }
    }
}
```

GET_STEP_DATAFRAME

从项目配方中的一个步骤中获取数据框。仅用于交互式体验。与 ViewFrame 参数一起使用，在大型数据框中进行分页。

参数

- `stepIndex`— 项目配方中用于获取数据框的步骤的索引。

Example 示例

```
{
    "RecipeAction": {
```

```
        "Operation": "GET_STEP_DATAFRAME",
        "Parameters": {
            "stepIndex": "0"
        }
    }
}
```

API 参考

如果您是开发人员，则可以编写访问 DataBrew API（应用程序编程接口）的应用程序。建议您使用语言特定的 AWS SDK。有关更多信息，请参阅用于在 AWS上进行构建的工具。

S AWS DK 代表您构建低级 DataBrew API 请求并处理来自的响应。 DataBrew这可让您专注于应用程序逻辑而不是低级详细信息。

以下是的 API 操作、数据类型和例外情况 DataBrew。

主题

操作

支持以下操作：

- DeleteSchedule
- DescribeDataset
- DescribeJob
- DescribeJobRun
- DescribeProject
- DescribeRecipe
- DescribeRuleset
- DescribeSchedule
- ListDatasets
- ListJobRuns
- ListJobs
- ListProjects
- ListRecipes
- ListRecipeVersions
- ListRulesets
- ListSchedules
- ListTagsForResource
- PublishRecipe
- SendProjectSessionAction
- StartJobRun
- StartProjectSession
- StopJobRun
- TagResource
- UntagResource
- UpdateDataset
- UpdateProfileJob
- UpdateProject
- UpdateRecipe
- UpdateRecipeJob
- UpdateRuleset

- UpdateSchedule

BatchDeleteRecipeVersion

一次删除食谱的一个或多个版本。

在以下情况下，整个请求将被拒绝：

- 该配方不存在。
- 版本列表中存在无效的版本标识符。
- 版本列表为空。
- 版本列表大小超过 50。
- 版本列表包含重复的条目。

如果出现以下情况，请求将成功完成，但部分失败：

- 版本不存在。
- 作业正在使用一个版本。
- 你指定LATEST_WORKING，但它正在被一个项目使用。
- 版本删除失败。

仅当配方没有其他LATEST_WORKING版本时，才会删除该版本。如果您LATEST_WORKING在其他版本存在时尝试删除（或者无法将其删除），则LATEST_WORKING将在响应中列为部分失败。

请求语法

```
POST /recipes/name/batchDeleteRecipeVersion HTTP/1.1
Content-type: application/json

{
   "RecipeVersions": [ "string" ]
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要删除其版本的配方的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

RecipeVersions

要删除的配方版本的版本标识符数组。您可以指定数字版本 (X.Y) 或LATEST_WORKING。LATEST_PUBLISHED不支持。

类型：字符串数组

数组成员：最少 1 个物品。最多 50 项。

长度限制：长度下限为 1。最大长度为 16。

必需：是

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Errors": [
      {
         "ErrorCode": "string",
         "ErrorMessage": "string",
         "RecipeVersion": "string"
      }
   ],
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

修改过的配方的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

Errors

尝试删除配方版本时出现错误（如果有）。

类型：RecipeVersionErrorDetail 对象数组

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK

- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

CreateDataset

创建新的 DataBrew 数据集。

请求语法

```
POST /datasets HTTP/1.1
Content-type: application/json

{
   "Format": "string",
   "FormatOptions": {
      "Csv": {
         "Delimiter": "string",
         "HeaderRow": boolean
      },
      "Excel": {
         "HeaderRow": boolean,
         "SheetIndexes": [ number ],
         "SheetNames": [ "string" ]
      },
      "Json": {
         "MultiLine": boolean
      }
   },
   "Input": {
      "DatabaseInputDefinition": {
         "DatabaseTableName": "string",
         "GlueConnectionName": "string",
         "QueryString": "string",
         "TempDirectory": {
            "Bucket": "string",
            "BucketOwner": "string",
            "Key": "string"
         }
      },
      "DataCatalogInputDefinition": {
         "CatalogId": "string",
         "DatabaseName": "string",
         "TableName": "string",
         "TempDirectory": {
            "Bucket": "string",
            "BucketOwner": "string",
```

```
            "Key": "string"
         }
      },
      "Metadata": {
         "SourceArn": "string"
      },
      "S3InputDefinition": {
         "Bucket": "string",
         "BucketOwner": "string",
         "Key": "string"
      }
   },
   "Name": "string",
   "PathOptions": {
      "FilesLimit": {
         "MaxFiles": number,
         "Order": "string",
         "OrderedBy": "string"
      },
      "LastModifiedDateCondition": {
         "Expression": "string",
         "ValuesMap": {
            "string" : "string"
         }
      },
      "Parameters": {
         "string" : {
            "CreateColumn": boolean,
            "DatetimeOptions": {
               "Format": "string",
               "LocaleCode": "string",
               "TimezoneOffset": "string"
            },
            "Filter": {
               "Expression": "string",
               "ValuesMap": {
                  "string" : "string"
               }
            },
            "Name": "string",
            "Type": "string"
         }
      }
   },
```

```
   "Tags": {
      "string" : "string"
   }
}
```

URI 请求参数

该请求不使用任何 URI 参数。

请求体

请求接受采用 JSON 格式的以下数据。

Input

表示有关 DataBrew 如何在 Amazon S3 AWS Glue Data Catalog 或 Amazon S3 中查找数据的信息。

类型：Input 对象

必需：是

Name

要创建的数据集的名称。有效字符包括字母数字（A-Z、a-z、0-9）、连字符 (-)、句点 (.) 和空格。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

Format

从 Amazon S3 文件或文件夹创建的数据集的文件格式。

类型：字符串

有效值：CSV | JSON | PARQUET | EXCEL | ORC

必需：否

FormatOptions

表示一组选项，这些选项用于定义逗号分隔值 (CSV)、Excel 或 JSON 输入的结构。

类型：FormatOptions 对象

必需：否

PathOptions

一组选项，用于定义如何 DataBrew 解释数据集的 Amazon S3 路径。

类型：PathOptions 对象

必需：否

Tags

要应用于此数据集的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您创建的数据集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

AccessDeniedException

对指定资源的访问被拒绝。

HTTP 状态代码：403

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包

- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

CreateProfileJob

创建新作业以分析数据集并创建其数据配置文件。

请求语法

```
POST /profileJobs HTTP/1.1
Content-type: application/json

{
   "Configuration": {
      "ColumnStatisticsConfigurations": [
         {
            "Selectors": [
               {
                  "Name": "string",
                  "Regex": "string"
               }
            ],
            "Statistics": {
               "IncludedStatistics": [ "string" ],
               "Overrides": [
                  {
                     "Parameters": {
                        "string" : "string"
                     },
                     "Statistic": "string"
                  }
               ]
            }
         }
      ],
      "DatasetStatisticsConfiguration": {
         "IncludedStatistics": [ "string" ],
         "Overrides": [
            {
               "Parameters": {
                  "string" : "string"
               },
               "Statistic": "string"
            }
         ]
      },
```

```
      "EntityDetectorConfiguration": {
         "AllowedStatistics": [
            {
               "Statistics": [ "string" ]
            }
         ],
         "EntityTypes": [ "string" ]
      },
      "ProfileColumns": [
         {
            "Name": "string",
            "Regex": "string"
         }
      ]
   },
   "DatasetName": "string",
   "EncryptionKeyArn": "string",
   "EncryptionMode": "string",
   "JobSample": {
      "Mode": "string",
      "Size": number
   },
   "LogSubscription": "string",
   "MaxCapacity": number,
   "MaxRetries": number,
   "Name": "string",
   "OutputLocation": {
      "Bucket": "string",
      "BucketOwner": "string",
      "Key": "string"
   },
   "RoleArn": "string",
   "Tags": {
      "string" : "string"
   },
   "Timeout": number,
   "ValidationConfigurations": [
      {
         "RulesetArn": "string",
         "ValidationMode": "string"
      }
   ]
}
```

URI 请求参数

该请求不使用任何 URI 参数。

请求体

请求接受采用 JSON 格式的以下数据。

DatasetName

此作业要处理的数据集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

Name

要创建的任务的名称。有效字符包括字母数字（A-Z、a-z、0-9）、连字符 (-)、句点 (.) 和空格。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

必需：是

OutputLocation

表示 Amazon S3 位置（存储桶名称、存储桶拥有者和对象密钥），DataBrew 可以在其中读取输入数据或写入任务的输出。

类型：S3Location 对象

必需：是

RoleArn

DataBrew 运行任务时要假设的 AWS Identity and Access Management (IAM) 角色的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：是

Configuration

配置文件作业的配置。用于选择列、进行评估和覆盖评估的默认参数。当配置为空时，分析作业将使用默认设置运行。

类型：ProfileConfiguration 对象

必需：否

EncryptionKeyArn

用于保护任务的加密密钥的 Amazon 资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

EncryptionMode

作业的加密模式包括以下几种：

- SSE-KMS-SSE-KMS-使用 AWS KMS托管密钥进行服务器端加密。
- SSE-S3 - 使用 Amazon S3 托管密钥进行服务器端加密。

类型：字符串

有效值：SSE-KMS | SSE-S3

必需：否

JobSample

仅适用于配置文件作业的示例配置。确定要执行分析作业的行数。如果未提供 JobSample 值，则将使用默认值。模式参数的默认值为 CUSTOM_ROWS，大小参数的默认值为 20000。

类型：JobSample 对象

必需：否

LogSubscription

为任务启用或禁用 Amazon CloudWatch 日志记录。如果启用了日志记录，则为每个作业运行 CloudWatch 写入一个日志流。

类型：字符串

有效值：ENABLE | DISABLE

必需：否

MaxCapacity

作业处理数据时 DataBrew 可使用的最大节点数。

类型：整数

必需：否

MaxRetries

作业运行失败后重试此作业的最大次数。

类型：整数

有效范围：最小值为 0。

必需：否

Tags

适用于此任务的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

Timeout

作业的超时（以分钟为单位）。如果作业的运行时间超出此超时时间，作业将以 TIMEOUT 状态结束。

类型：整数

有效范围：最小值为 0。

必需：否

ValidationConfigurations

应用于配置文件作业的验证配置列表。

类型：ValidationConfiguration 对象数组

数组成员：最少 1 个物品。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

已创建的作业的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

错误

有关所有操作的常见错误信息，请参阅常见错误。

AccessDeniedException

对指定资源的访问被拒绝。

HTTP 状态代码：403

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

CreateProject

创建新 DataBrew 项目。

请求语法

```
POST /projects HTTP/1.1
Content-type: application/json

{
   "DatasetName": "string",
   "Name": "string",
   "RecipeName": "string",
   "RoleArn": "string",
   "Sample": {
      "Size": number,
      "Type": "string"
   },
   "Tags": {
      "string" : "string"
   }
}
```

URI 请求参数

该请求不使用任何 URI 参数。

请求体

请求接受采用 JSON 格式的以下数据。

DatasetName

要与该项目关联的现有数据集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

Name

新项目的唯一名称。有效字符包括字母数字（A-Z、a-z、0-9）、连字符 (-)、句点 (.) 和空格。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

RecipeName

要与项目关联的现有配方的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

RoleArn

为该请求承担的 (IAM) 角色的亚马逊资源名称 AWS Identity and Access Management (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：是

Sample

表示用于交互式数据分析 DataBrew 的样本数量和采样类型。

类型：Sample 对象

必需：否

Tags

要应用于此项目的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您创建的项目的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

InternalServerException

出现内部服务故障。

HTTP 状态代码：500

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

CreateRecipe

创建新 DataBrew 配方。

请求语法

```
POST /recipes HTTP/1.1
Content-type: application/json

{
   "Description": "string",
   "Name": "string",
   "Steps": [
      {
         "Action": {
            "Operation": "string",
            "Parameters": {
               "string" : "string"
            }
         },
         "ConditionExpressions": [
            {
               "Condition": "string",
               "TargetColumn": "string",
               "Value": "string"
            }
         ]
      }
   ],
   "Tags": {
      "string" : "string"
   }
}
```

URI 请求参数

该请求不使用任何 URI 参数。

请求体

请求接受采用 JSON 格式的以下数据。

Name

食谱的唯一名称。有效字符包括字母数字（A-Z、a-z、0-9）、连字符 (-)、句点 (.) 和空格。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

Steps

包含配方要执行的步骤的数组。每个配方步骤都由一个配方操作和（可选）一组条件表达式组成。

类型：RecipeStep 对象数组

必需：是

Description

食谱的描述。

类型：字符串

长度限制：最大长度为 1024。

必需：否

Tags

适用于此食谱的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您创建的食谱的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

CreateRecipeJob

使用现有 AWS Glue DataBrew 配方中定义的步骤创建用于转换输入数据的新作业

请求语法

```
POST /recipeJobs HTTP/1.1
Content-type: application/json

{
   "DatabaseOutputs": [
      {
         "DatabaseOptions": {
            "TableName": "string",
            "TempDirectory": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
         },
         "DatabaseOutputMode": "string",
         "GlueConnectionName": "string"
      }
   ],
   "DataCatalogOutputs": [
      {
         "CatalogId": "string",
         "DatabaseName": "string",
         "DatabaseOptions": {
            "TableName": "string",
            "TempDirectory": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
         },
         "Overwrite": boolean,
         "S3Options": {
            "Location": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
```

```
         },
         "TableName": "string"
      }
   ],
   "DatasetName": "string",
   "EncryptionKeyArn": "string",
   "EncryptionMode": "string",
   "LogSubscription": "string",
   "MaxCapacity": number,
   "MaxRetries": number,
   "Name": "string",
   "Outputs": [
      {
         "CompressionFormat": "string",
         "Format": "string",
         "FormatOptions": {
            "Csv": {
               "Delimiter": "string"
            }
         },
         "Location": {
            "Bucket": "string",
            "BucketOwner": "string",
            "Key": "string"
         },
         "MaxOutputFiles": number,
         "Overwrite": boolean,
         "PartitionColumns": [ "string" ]
      }
   ],
   "ProjectName": "string",
   "RecipeReference": {
      "Name": "string",
      "RecipeVersion": "string"
   },
   "RoleArn": "string",
   "Tags": {
      "string" : "string"
   },
   "Timeout": number
}
```

URI 请求参数

该请求不使用任何 URI 参数。

请求体

请求接受采用 JSON 格式的以下数据。

Name

作业的唯一名称。有效字符包括字母数字（A-Z、a-z、0-9）、连字符 (-)、句点 (.) 和空格。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

必需：是

RoleArn

DataBrew 运行任务时要假设的 AWS Identity and Access Management (IAM) 角色的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：是

DatabaseOutputs

表示 JDBC 数据库输出对象的列表，该对象定义了要写入的 DataBrew 配方作业的输出目标。

类型：DatabaseOutput 对象数组

数组成员：最少 1 个物品。

必需：否

DataCatalogOutputs

一个或多个工件，表示运行作业的 AWS Glue Data Catalog 输出。

类型：DataCatalogOutput 对象数组

数组成员：最少 1 个物品。

必需：否

DatasetName

此作业处理的数据集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

EncryptionKeyArn

用于保护任务的加密密钥的 Amazon 资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

EncryptionMode

作业的加密模式包括以下几种：

- SSE-KMS-使用由 AWS KMS管理的密钥进行服务器端加密。
- SSE-S3 - 使用 Amazon S3 托管密钥进行服务器端加密。

类型：字符串

有效值：`SSE-KMS | SSE-S3`

必需：否

LogSubscription

为任务启用或禁用 Amazon CloudWatch 日志记录。如果启用了日志记录，则为每个作业运行 CloudWatch 写入一个日志流。

类型：字符串

有效值：`ENABLE | DISABLE`

必需：否

MaxCapacity

任务处理数据时 DataBrew 可以消耗的最大节点数。

类型：整数

必需：否

MaxRetries

作业运行失败后重试此作业的最大次数。

类型：整数

有效范围：最小值为 0。

必需：否

Outputs

代表作业运行时 输出的一个或多个构件。

类型：Output 对象数组

数组成员：最少 1 个物品。

必需：否

ProjectName

要么是现有项目的名称，要么是要与配方关联的配方和数据集的组合。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

RecipeReference

表示 DataBrew 配方的名称和版本。

类型：RecipeReference 对象

必需：否

Tags

适用于此任务的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

Timeout

作业的超时（以分钟为单位）。如果作业的运行时间超出此超时时间，作业将以 TIMEOUT 状态结束。

类型：整数

有效范围：最小值为 0。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您创建的任务的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

错误

有关所有操作的常见错误信息，请参阅常见错误。

AccessDeniedException

对指定资源的访问被拒绝。

HTTP 状态代码：403

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

CreateRuleset

创建可用于分析作业的新规则集，以验证数据集的数据质量。

请求语法

```
POST /rulesets HTTP/1.1
Content-type: application/json

{
   "Description": "string",
   "Name": "string",
   "Rules": [
      {
         "CheckExpression": "string",
         "ColumnSelectors": [
            {
               "Name": "string",
               "Regex": "string"
            }
         ],
         "Disabled": boolean,
         "Name": "string",
         "SubstitutionMap": {
            "string" : "string"
         },
         "Threshold": {
            "Type": "string",
            "Unit": "string",
            "Value": number
         }
      }
   ],
   "Tags": {
      "string" : "string"
   },
   "TargetArn": "string"
}
```

URI 请求参数

该请求不使用任何 URI 参数。

请求体

请求接受采用 JSON 格式的以下数据。

Name

要创建的规则集的名称。有效字符包括字母数字（A-Z、a-z、0-9）、连字符 (-)、句点 (.) 和空格。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

Rules

使用规则集定义的规则列表。规则包括一项或多项要对 DataBrew 数据集进行验证的校验。

类型：Rule 对象数组

数组成员：最少 1 个物品。

必需：是

TargetArn

与规则集关联的资源（数据集）的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：是

Description

规则集的描述。

类型：字符串

长度限制：最大长度为 1024。

必需：否

Tags

要应用于规则集的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

创建的规则集的唯一名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

CreateSchedule

为一个或多个 DataBrew 作业创建新的时间表。作业可以在特定的日期和时间运行，也可以定期运行。

请求语法

```
POST /schedules HTTP/1.1
Content-type: application/json

{
   "CronExpression": "string",
   "JobNames": [ "string" ],
   "Name": "string",
   "Tags": {
      "string" : "string"
   }
}
```

URI 请求参数

该请求不使用任何 URI 参数。

请求体

请求接受采用 JSON 格式的以下数据。

CronExpression

运行作业的日期、日期和时间。有关更多信息，请参阅《 AWS Glue DataBrew 开发人员指南》中的 Cron 表达式。

类型：字符串

长度限制：长度下限为 1。最大长度为 512。

必需：是

Name

时间表的唯一名称。有效字符包括字母数字（ A-Z、a-z、0-9 ）、连字符 (-)、句点 (.) 和空格。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

JobNames

要运行的一个或多个作业的名称。

类型：字符串数组

数组成员：最多 50 项。

长度限制：长度下限为 1。最大长度为 240。

必需：否

Tags

适用于此计划的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

已创建的时间表的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3

- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DeleteDataset

从中删除数据集 DataBrew。

请求语法

```
DELETE /datasets/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要删除的数据集的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您删除的数据集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DeleteJob

删除指定的 DataBrew 作业。

请求语法

```
DELETE /jobs/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要删除的任务的名称。

长度限制：长度下限为 1。最大长度为 240。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您删除的任务的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DeleteProject

删除现有 DataBrew 项目。

请求语法

```
DELETE /projects/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要删除的项目的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您删除的项目的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DeleteRecipeVersion

删除食 DataBrew 谱的单个版本。

请求语法

```
DELETE /recipes/name/recipeVersion/recipeVersion HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

食谱的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

recipeVersion

要删除的配方的版本。您可以指定数字版本 (X.Y) 或LATEST_WORKING。 LATEST_PUBLISHED不支持。

长度限制：长度下限为 1。最大长度为 16。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string",
   "RecipeVersion": "string"
```

```
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

已删除的食谱的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

RecipeVersion

已删除的食谱版本。

类型：字符串

长度限制：长度下限为 1。最大长度为 16。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DeleteRuleset

删除规则集。

请求语法

```
DELETE /rulesets/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要删除的规则集的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

已删除的规则集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DeleteSchedule

删除指定的 DataBrew 时间表。

请求语法

```
DELETE /schedules/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要删除的时间表的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

已删除的时间表的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DescribeDataset

返回特定 DataBrew 数据集的定义。

请求语法

```
GET /datasets/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要描述的数据集的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "CreateDate": number,
   "CreatedBy": "string",
   "Format": "string",
   "FormatOptions": {
      "Csv": {
         "Delimiter": "string",
         "HeaderRow": boolean
      },
      "Excel": {
         "HeaderRow": boolean,
         "SheetIndexes": [ number ],
         "SheetNames": [ "string" ]
```

```
      },
      "Json": {
         "MultiLine": boolean
      }
   },
   "Input": {
      "DatabaseInputDefinition": {
         "DatabaseTableName": "string",
         "GlueConnectionName": "string",
         "QueryString": "string",
         "TempDirectory": {
            "Bucket": "string",
            "BucketOwner": "string",
            "Key": "string"
         }
      },
      "DataCatalogInputDefinition": {
         "CatalogId": "string",
         "DatabaseName": "string",
         "TableName": "string",
         "TempDirectory": {
            "Bucket": "string",
            "BucketOwner": "string",
            "Key": "string"
         }
      },
      "Metadata": {
         "SourceArn": "string"
      },
      "S3InputDefinition": {
         "Bucket": "string",
         "BucketOwner": "string",
         "Key": "string"
      }
   },
   "LastModifiedBy": "string",
   "LastModifiedDate": number,
   "Name": "string",
   "PathOptions": {
      "FilesLimit": {
         "MaxFiles": number,
         "Order": "string",
         "OrderedBy": "string"
      },
```

```
      "LastModifiedDateCondition": {
         "Expression": "string",
         "ValuesMap": {
            "string" : "string"
         }
      },
      "Parameters": {
         "string" : {
            "CreateColumn": boolean,
            "DatetimeOptions": {
               "Format": "string",
               "LocaleCode": "string",
               "TimezoneOffset": "string"
            },
            "Filter": {
               "Expression": "string",
               "ValuesMap": {
                  "string" : "string"
               }
            },
            "Name": "string",
            "Type": "string"
         }
      }
   },
   "ResourceArn": "string",
   "Source": "string",
   "Tags": {
      "string" : "string"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Input

表示有关 DataBrew 如何在 Amazon S3 AWS Glue Data Catalog 或 Amazon S3 中查找数据的信息。

类型：Input 对象

Name

数据集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

CreateDate

数据集的创建日期和时间。

类型：时间戳

CreatedBy

创建数据集的用户的标识符（用户名）。

类型：字符串

Format

从 Amazon S3 文件或文件夹创建的数据集的文件格式。

类型：字符串

有效值：`CSV | JSON | PARQUET | EXCEL | ORC`

FormatOptions

表示一组选项，这些选项用于定义逗号分隔值 (CSV)、Excel 或 JSON 输入的结构。

类型：FormatOptions 对象

LastModifiedBy

上次修改数据集的用户的标识符（用户名）。

类型：字符串

LastModifiedDate

上次修改数据集的日期和时间。

类型：时间戳

PathOptions

一组选项，用于定义如何 DataBrew 解释数据集的 Amazon S3 路径。

类型：PathOptions 对象

ResourceArn

数据集的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

Source

此数据集的数据位置，Amazon S3 或 AWS Glue Data Catalog。

类型：字符串

有效值：`S3 | DATA-CATALOG | DATABASE`

Tags

与此数据集关联的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DescribeJob

返回特定 DataBrew 作业的定义。

请求语法

```
GET /jobs/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要描述的任务的名称。

长度限制：长度下限为 1。最大长度为 240。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "CreateDate": number,
   "CreatedBy": "string",
   "DatabaseOutputs": [
      {
         "DatabaseOptions": {
            "TableName": "string",
            "TempDirectory": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
         },
```

```
        "DatabaseOutputMode": "string",
        "GlueConnectionName": "string"
      }
   ],
   "DataCatalogOutputs": [
      {
        "CatalogId": "string",
        "DatabaseName": "string",
        "DatabaseOptions": {
           "TableName": "string",
           "TempDirectory": {
              "Bucket": "string",
              "BucketOwner": "string",
              "Key": "string"
           }
        },
        "Overwrite": boolean,
        "S3Options": {
           "Location": {
              "Bucket": "string",
              "BucketOwner": "string",
              "Key": "string"
           }
        },
        "TableName": "string"
      }
   ],
   "DatasetName": "string",
   "EncryptionKeyArn": "string",
   "EncryptionMode": "string",
   "JobSample": {
      "Mode": "string",
      "Size": number
   },
   "LastModifiedBy": "string",
   "LastModifiedDate": number,
   "LogSubscription": "string",
   "MaxCapacity": number,
   "MaxRetries": number,
   "Name": "string",
   "Outputs": [
      {
        "CompressionFormat": "string",
        "Format": "string",
```

```
        "FormatOptions": {
           "Csv": {
              "Delimiter": "string"
           }
        },
        "Location": {
           "Bucket": "string",
           "BucketOwner": "string",
           "Key": "string"
        },
        "MaxOutputFiles": number,
        "Overwrite": boolean,
        "PartitionColumns": [ "string" ]
     }
  ],
  "ProfileConfiguration": {
     "ColumnStatisticsConfigurations": [
        {
           "Selectors": [
              {
                 "Name": "string",
                 "Regex": "string"
              }
           ],
           "Statistics": {
              "IncludedStatistics": [ "string" ],
              "Overrides": [
                 {
                    "Parameters": {
                       "string" : "string"
                    },
                    "Statistic": "string"
                 }
              ]
           }
        }
     ],
     "DatasetStatisticsConfiguration": {
        "IncludedStatistics": [ "string" ],
        "Overrides": [
           {
              "Parameters": {
                 "string" : "string"
              },
```

```
                "Statistic": "string"
            }
         ]
      },
      "EntityDetectorConfiguration": {
         "AllowedStatistics": [
            {
               "Statistics": [ "string" ]
            }
         ],
         "EntityTypes": [ "string" ]
      },
      "ProfileColumns": [
         {
            "Name": "string",
            "Regex": "string"
         }
      ]
   },
   "ProjectName": "string",
   "RecipeReference": {
      "Name": "string",
      "RecipeVersion": "string"
   },
   "ResourceArn": "string",
   "RoleArn": "string",
   "Tags": {
      "string" : "string"
   },
   "Timeout": number,
   "Type": "string",
   "ValidationConfigurations": [
      {
         "RulesetArn": "string",
         "ValidationMode": "string"
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

作业的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

CreateDate

创建作业的日期和时间。

类型：时间戳

CreatedBy

与创建任务相关的用户的标识符（用户名）。

类型：字符串

DatabaseOutputs

表示 JDBC 数据库输出对象的列表，该对象定义了要写入的 DataBrew 配方作业的输出目标。

类型：DatabaseOutput 对象数组

数组成员：最少 1 个物品。

DataCatalogOutputs

一个或多个工件，用于表示运行作业的 AWS Glue Data Catalog 输出。

类型：DataCatalogOutput 对象数组

数组成员：最少 1 个物品。

DatasetName

作业所依据的数据集。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

EncryptionKeyArn

用于保护任务的加密密钥的 Amazon 资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

EncryptionMode

作业的加密模式包括以下几种：

- SSE-KMS-使用由 AWS KMS管理的密钥进行服务器端加密。
- SSE-S3 - 使用 Amazon S3 托管密钥进行服务器端加密。

类型：字符串

有效值：SSE-KMS | SSE-S3

JobSample

仅适用于配置文件作业的示例配置。确定要执行分析作业的行数。

类型：JobSample 对象

LastModifiedBy

上次修改作业的用户的标识符（用户名）。

类型：字符串

LastModifiedDate

上次修改作业的日期和时间。

类型：时间戳

LogSubscription

表示此任务是否启用了 Amazon CloudWatch 日志记录。

类型：字符串

有效值：ENABLE | DISABLE

MaxCapacity

作业处理数据时 DataBrew 可以消耗的最大计算节点数。

类型：整数

MaxRetries

作业运行失败后重试此作业的最大次数。

类型：整数

有效范围：最小值为 0。

Outputs

代表作业运行时 输出的一个或多个构件。

类型：Output 对象数组

数组成员：最少 1 个物品。

ProfileConfiguration

配置文件作业的配置。用于选择列、进行评估和覆盖评估的默认参数。当配置为空时，分析作业将使用默认设置运行。

类型：ProfileConfiguration 对象

ProjectName

与此工作相关的 DataBrew 项目。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

RecipeReference

表示 DataBrew 配方的名称和版本。

类型：RecipeReference 对象

ResourceArn

任务的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

RoleArn

DataBrew 运行任务时要扮演的 AWS Identity and Access Management (IAM) 角色的 ARN。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

Tags

与此任务关联的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

Timeout

作业的超时（以分钟为单位）。如果作业的运行时间超出此超时时间，作业将以 TIMEOUT 状态结束。

类型：整数

有效范围：最小值为 0。

Type

作业类型，必须是以下类型之一：

- PROFILE-该作业分析数据集以确定其大小、数据类型、数据分布等。
- RECIPE-该作业对数据集应用一个或多个转换。

类型：字符串

有效值：PROFILE | RECIPE

ValidationConfigurations

应用于配置文件作业的验证配置列表。

类型：ValidationConfiguration 对象数组

数组成员：最少 1 个物品。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DescribeJobRun

表示 DataBrew 作业的一次运行。

请求语法

```
GET /jobs/name/jobRun/runId HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

此次运行期间正在处理的作业的名称。

长度限制：长度下限为 1。最大长度为 240。

必需：是

runId

作业运行的唯一标识符。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Attempt": number,
   "CompletedOn": number,
   "DatabaseOutputs": [
      {
         "DatabaseOptions": {
```

```
            "TableName": "string",
            "TempDirectory": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
         },
         "DatabaseOutputMode": "string",
         "GlueConnectionName": "string"
      }
   ],
   "DataCatalogOutputs": [
      {
         "CatalogId": "string",
         "DatabaseName": "string",
         "DatabaseOptions": {
            "TableName": "string",
            "TempDirectory": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
         },
         "Overwrite": boolean,
         "S3Options": {
            "Location": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
         },
         "TableName": "string"
      }
   ],
   "DatasetName": "string",
   "ErrorMessage": "string",
   "ExecutionTime": number,
   "JobName": "string",
   "JobSample": {
      "Mode": "string",
      "Size": number
   },
   "LogGroupName": "string",
   "LogSubscription": "string",
```

```
   "Outputs": [
      {
         "CompressionFormat": "string",
         "Format": "string",
         "FormatOptions": {
            "Csv": {
               "Delimiter": "string"
            }
         },
         "Location": {
            "Bucket": "string",
            "BucketOwner": "string",
            "Key": "string"
         },
         "MaxOutputFiles": number,
         "Overwrite": boolean,
         "PartitionColumns": [ "string" ]
      }
   ],
   "ProfileConfiguration": {
      "ColumnStatisticsConfigurations": [
         {
            "Selectors": [
               {
                  "Name": "string",
                  "Regex": "string"
               }
            ],
            "Statistics": {
               "IncludedStatistics": [ "string" ],
               "Overrides": [
                  {
                     "Parameters": {
                        "string" : "string"
                     },
                     "Statistic": "string"
                  }
               ]
            }
         }
      ],
      "DatasetStatisticsConfiguration": {
         "IncludedStatistics": [ "string" ],
         "Overrides": [
```

```
            {
               "Parameters": {
                  "string" : "string"
               },
               "Statistic": "string"
            }
         ]
      },
      "EntityDetectorConfiguration": {
         "AllowedStatistics": [
            {
               "Statistics": [ "string" ]
            }
         ],
         "EntityTypes": [ "string" ]
      },
      "ProfileColumns": [
         {
            "Name": "string",
            "Regex": "string"
         }
      ]
   },
   "RecipeReference": {
      "Name": "string",
      "RecipeVersion": "string"
   },
   "RunId": "string",
   "StartedBy": "string",
   "StartedOn": number,
   "State": "string",
   "ValidationConfigurations": [
      {
         "RulesetArn": "string",
         "ValidationMode": "string"
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

JobName

此次运行期间正在处理的作业的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

Attempt

尝试运行作业 DataBrew 的次数。

类型：整数

CompletedOn

任务完成处理的日期和时间。

类型：时间戳

DatabaseOutputs

表示 JDBC 数据库输出对象的列表，该对象定义了要写入的 DataBrew 配方作业的输出目标。

类型：DatabaseOutput 对象数组

数组成员：最少 1 个物品。

DataCatalogOutputs

一个或多个工件，表示运行作业的 AWS Glue Data Catalog 输出。

类型：DataCatalogOutput 对象数组

数组成员：最少 1 个物品。

DatasetName

要处理的作业的数据集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

ErrorMessage

一条消息，指示作业运行时遇到错误（如果有）。

类型：字符串

ExecutionTime

作业运行消耗资源的时间（以秒为单位）。

类型：整数

JobSample

仅适用于配置文件作业的示例配置。确定要执行分析作业的行数。如果未提供 JobSample 值，则将使用默认值。模式参数的默认值为 CUSTOM_ROWS，大小参数的默认值为 20000。

类型：JobSample 对象

LogGroupName

Amazon CloudWatch 日志组的名称，作业在运行时写入诊断消息。

类型：字符串

长度限制：长度下限为 1。最大长度为 512。

LogSubscription

Amazon CloudWatch 记录任务运行的当前状态。

类型：字符串

有效值：`ENABLE | DISABLE`

Outputs

作业运行中的一个或多个输出工件。

类型：Output 对象数组

数组成员：最少 1 个物品。

ProfileConfiguration

配置文件作业的配置。用于选择列、进行评估和覆盖评估的默认参数。当配置为空时，分析作业将使用默认设置运行。

类型：ProfileConfiguration 对象

RecipeReference

表示 DataBrew 配方的名称和版本。

类型：RecipeReference 对象

RunId

作业运行的唯一标识符。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

StartedBy

开始运行任务的用户的亚马逊资源名称 (ARN)。

类型：字符串

StartedOn

作业运行开始的日期和时间。

类型：时间戳

State

作业运行实体本身的当前状态。

类型：字符串

有效值：STARTING | RUNNING | STOPPING | STOPPED | SUCCEEDED | FAILED | TIMEOUT

ValidationConfigurations

应用于配置文件作业的验证配置列表。

类型：ValidationConfiguration 对象数组

数组成员：最少 1 个物品。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DescribeProject

返回特定 DataBrew 项目的定义。

请求语法

```
GET /projects/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要描述的项目的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "CreateDate": number,
   "CreatedBy": "string",
   "DatasetName": "string",
   "LastModifiedBy": "string",
   "LastModifiedDate": number,
   "Name": "string",
   "OpenDate": number,
   "OpenedBy": "string",
   "RecipeName": "string",
   "ResourceArn": "string",
   "RoleArn": "string",
   "Sample": {
```

```
      "Size": number,
      "Type": "string"
   },
   "SessionStatus": "string",
   "Tags": {
      "string" : "string"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

项目的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

CreateDate

项目的创建日期和时间。

类型：时间戳

CreatedBy

创建项目的用户的标识符（用户名）。

类型：字符串

DatasetName

与项目关联的数据集。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

LastModifiedBy

上次修改项目的用户的标识符（用户名）。

类型：字符串

LastModifiedDate

项目上次修改的日期和时间。

类型：时间戳

OpenDate

项目打开的日期和时间。

类型：时间戳

OpenedBy

打开项目以供使用的用户的标识符（用户名）。

类型：字符串

RecipeName

与此工作相关的配方。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

ResourceArn

项目的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

RoleArn

DataBrew 运行任务时要扮演的 AWS Identity and Access Management (IAM) 角色的 ARN。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

Sample

表示用于交互式数据分析 DataBrew 的样本数量和采样类型。

类型：Sample 对象

SessionStatus

描述会话的当前状态：

- PROVISIONING-为会议分配资源。
- INITIALIZING-为首次使用做好会话准备。
- ASSIGNED-会话已准备就绪，可供使用。

类型：字符串

有效值：ASSIGNED | FAILED | INITIALIZING | PROVISIONING | READY | RECYCLING | ROTATING | TERMINATED | TERMINATING | UPDATING

Tags

与此项目关联的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DescribeRecipe

返回与特定版本对应的特定 DataBrew 配方的定义。

请求语法

```
GET /recipes/name?recipeVersion=RecipeVersion HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要描述的食谱的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

RecipeVersion

配方版本标识符。如果未指定此参数，则返回最新发布的版本。

长度限制：长度下限为 1。最大长度为 16。

请求正文

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "CreateDate": number,
   "CreatedBy": "string",
   "Description": "string",
   "LastModifiedBy": "string",
   "LastModifiedDate": number,
   "Name": "string",
```

```
   "ProjectName": "string",
   "PublishedBy": "string",
   "PublishedDate": number,
   "RecipeVersion": "string",
   "ResourceArn": "string",
   "Steps": [
      {
         "Action": {
            "Operation": "string",
            "Parameters": {
               "string" : "string"
            }
         },
         "ConditionExpressions": [
            {
               "Condition": "string",
               "TargetColumn": "string",
               "Value": "string"
            }
         ]
      }
   ],
   "Tags": {
      "string" : "string"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

食谱的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

CreateDate

创建食谱的日期和时间。

类型：时间戳

CreatedBy

创建配方的用户的标识符（用户名）。

类型：字符串

Description

对食谱的描述。

类型：字符串

长度限制：最大长度为 1024。

LastModifiedBy

上次修改配方的用户的标识符（用户名）。

类型：字符串

LastModifiedDate

上次修改食谱的日期和时间。

类型：时间戳

ProjectName

与此配方关联的项目名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

PublishedBy

上次发布食谱的用户的标识符（用户名）。

类型：字符串

PublishedDate

食谱上次发布的日期和时间。

类型：时间戳

RecipeVersion

配方版本标识符。

类型：字符串

长度限制：长度下限为 1。最大长度为 16。

ResourceArn

脚本的 ARN。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

Steps

配方要执行的一个或多个步骤。每个步骤都由一个动作以及该操作应在何种条件下成功组成。

类型：RecipeStep 对象数组

Tags

与此项目关联的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DescribeRuleset

检索有关规则集的详细信息。

请求语法

```
GET /rulesets/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要描述的规则集的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "CreateDate": number,
   "CreatedBy": "string",
   "Description": "string",
   "LastModifiedBy": "string",
   "LastModifiedDate": number,
   "Name": "string",
   "ResourceArn": "string",
   "Rules": [
      {
         "CheckExpression": "string",
         "ColumnSelectors": [
            {
```

```
                "Name": "string",
                "Regex": "string"
            }
         ],
         "Disabled": boolean,
         "Name": "string",
         "SubstitutionMap": {
            "string" : "string"
         },
         "Threshold": {
            "Type": "string",
            "Unit": "string",
            "Value": number
         }
      }
   ],
   "Tags": {
      "string" : "string"
   },
   "TargetArn": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

规则集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

CreateDate

规则集的创建日期和时间。

类型：时间戳

CreatedBy

创建规则集的用户的亚马逊资源名称 (ARN)。

类型：字符串

Description

规则集的描述。

类型：字符串

长度限制：最大长度为 1024。

LastModifiedBy

上次修改规则集的用户的亚马逊资源名称 (ARN)。

类型：字符串

LastModifiedDate

规则集的修改日期和时间。

类型：时间戳

ResourceArn

规则集的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

Rules

使用规则集定义的规则列表。规则包括一项或多项要对 DataBrew 数据集进行验证的校验。

类型：Rule 对象数组

数组成员：最少 1 个物品。

Tags

已应用于规则集的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

TargetArn

与规则集关联的资源（数据集）的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版 SDK
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

DescribeSchedule

返回特定 DataBrew 时间表的定义。

请求语法

```
GET /schedules/name HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要描述的时间表的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "CreateDate": number,
   "CreatedBy": "string",
   "CronExpression": "string",
   "JobNames": [ "string" ],
   "LastModifiedBy": "string",
   "LastModifiedDate": number,
   "Name": "string",
   "ResourceArn": "string",
   "Tags": {
      "string" : "string"
   }
```

```
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

计划的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

CreateDate

计划创建的日期和时间。

类型：时间戳

CreatedBy

创建计划的用户的标识符（用户名）。

类型：字符串

CronExpression

按计划运行作业的日期、日期和时间。有关更多信息，请参阅《 AWS Glue DataBrew 开发人员指南》中的 Cron 表达式。

类型：字符串

长度限制：长度下限为 1。最大长度为 512。

JobNames

要使用计划运行的一个或多个作业的名称。

类型：字符串数组

数组成员：最多 50 项。

长度限制：长度下限为 1。最大长度为 240。

LastModifiedBy

上次修改计划的用户的标识符（用户名）。

类型：字符串

LastModifiedDate

上次修改时间表的日期和时间。

类型：时间戳

ResourceArn

计划的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

Tags

与此时间表关联的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListDatasets

列出所有 DataBrew 数据集。

请求语法

```
GET /datasets?maxResults=MaxResults&nextToken=NextToken HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

MaxResults

此请求中要返回的最大结果数。

有效范围：最小值为 1。最大值为 100。

NextToken

上一次调用返回的令牌，用于检索下一组结果。

长度限制：长度下限为 1。最大长度为 2000。

请求正文

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Datasets": [
      {
         "AccountId": "string",
         "CreateDate": number,
         "CreatedBy": "string",
         "Format": "string",
         "FormatOptions": {
            "Csv": {
               "Delimiter": "string",
```

```
            "HeaderRow": boolean
         },
         "Excel": {
            "HeaderRow": boolean,
            "SheetIndexes": [ number ],
            "SheetNames": [ "string" ]
         },
         "Json": {
            "MultiLine": boolean
         }
      },
      "Input": {
         "DatabaseInputDefinition": {
            "DatabaseTableName": "string",
            "GlueConnectionName": "string",
            "QueryString": "string",
            "TempDirectory": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
         },
         "DataCatalogInputDefinition": {
            "CatalogId": "string",
            "DatabaseName": "string",
            "TableName": "string",
            "TempDirectory": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
         },
         "Metadata": {
            "SourceArn": "string"
         },
         "S3InputDefinition": {
            "Bucket": "string",
            "BucketOwner": "string",
            "Key": "string"
         }
      },
      "LastModifiedBy": "string",
      "LastModifiedDate": number,
      "Name": "string",
```

```
         "PathOptions": {
            "FilesLimit": {
               "MaxFiles": number,
               "Order": "string",
               "OrderedBy": "string"
            },
            "LastModifiedDateCondition": {
               "Expression": "string",
               "ValuesMap": {
                  "string" : "string"
               }
            },
            "Parameters": {
               "string" : {
                  "CreateColumn": boolean,
                  "DatetimeOptions": {
                     "Format": "string",
                     "LocaleCode": "string",
                     "TimezoneOffset": "string"
                  },
                  "Filter": {
                     "Expression": "string",
                     "ValuesMap": {
                        "string" : "string"
                     }
                  },
                  "Name": "string",
                  "Type": "string"
               }
            }
         },
         "ResourceArn": "string",
         "Source": "string",
         "Tags": {
            "string" : "string"
         }
      }
   ],
   "NextToken": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Datasets

已定义的数据集列表。

类型：Dataset 对象数组

NextToken

可以在后续调用中用来检索下一组结果的令牌。

类型：字符串

长度限制：长度下限为 1。最大长度为 2000。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包

- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListJobRuns

列出特定 DataBrew 作业之前的所有运行情况。

请求语法

```
GET /jobs/name/jobRuns?maxResults=MaxResults&nextToken=NextToken HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

MaxResults

此请求中要返回的最大结果数。

有效范围：最小值为 1。最大值为 100。

name

作业的名称。

长度限制：长度下限为 1。最大长度为 240。

必需：是

NextToken

上一次调用返回的令牌，用于检索下一组结果。

长度限制：长度下限为 1。最大长度为 2000。

请求正文

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "JobRuns": [
      {
```

```
      "Attempt": number,
      "CompletedOn": number,
      "DatabaseOutputs": [
         {
            "DatabaseOptions": {
               "TableName": "string",
               "TempDirectory": {
                  "Bucket": "string",
                  "BucketOwner": "string",
                  "Key": "string"
               }
            },
            "DatabaseOutputMode": "string",
            "GlueConnectionName": "string"
         }
      ],
      "DataCatalogOutputs": [
         {
            "CatalogId": "string",
            "DatabaseName": "string",
            "DatabaseOptions": {
               "TableName": "string",
               "TempDirectory": {
                  "Bucket": "string",
                  "BucketOwner": "string",
                  "Key": "string"
               }
            },
            "Overwrite": boolean,
            "S3Options": {
               "Location": {
                  "Bucket": "string",
                  "BucketOwner": "string",
                  "Key": "string"
               }
            },
            "TableName": "string"
         }
      ],
      "DatasetName": "string",
      "ErrorMessage": "string",
      "ExecutionTime": number,
      "JobName": "string",
      "JobSample": {
```

```
            "Mode": "string",
            "Size": number
         },
         "LogGroupName": "string",
         "LogSubscription": "string",
         "Outputs": [
            {
               "CompressionFormat": "string",
               "Format": "string",
               "FormatOptions": {
                  "Csv": {
                     "Delimiter": "string"
                  }
               },
               "Location": {
                  "Bucket": "string",
                  "BucketOwner": "string",
                  "Key": "string"
               },
               "MaxOutputFiles": number,
               "Overwrite": boolean,
               "PartitionColumns": [ "string" ]
            }
         ],
         "RecipeReference": {
            "Name": "string",
            "RecipeVersion": "string"
         },
         "RunId": "string",
         "StartedBy": "string",
         "StartedOn": number,
         "State": "string",
         "ValidationConfigurations": [
            {
               "RulesetArn": "string",
               "ValidationMode": "string"
            }
         ]
      }
   ],
   "NextToken": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

JobRuns

指定作业已运行的作业列表。

类型：JobRun 对象数组

NextToken

可以在后续调用中用来检索下一组结果的令牌。

类型：字符串

长度限制：长度下限为 1。最大长度为 2000。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListJobs

列出所有已定义的 DataBrew 作业。

请求语法

```
GET /jobs?
datasetName=DatasetName&maxResults=MaxResults&nextToken=NextToken&projectName=ProjectName
 HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

DatasetName

数据集的名称。使用此参数表示仅返回那些作用于指定数据集的作业。

长度约束：最小长度为 1。最大长度为 255。

MaxResults

此请求中要返回的最大结果数。

有效范围：最小值为 1。最大值为 100。

NextToken

由生成的令牌 DataBrew ，用于指定在之前的请求被截断时在哪里继续分页。要获取下一组页面，请传入前一页调用的响应对象中的 NextToken值。

长度限制：长度下限为 1。最大长度为 2000。

ProjectName

项目的名称。使用此参数表示仅返回与指定项目关联的任务。

长度约束：最小长度为 1。最大长度为 255。

请求正文

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Jobs": [
      {
         "AccountId": "string",
         "CreateDate": number,
         "CreatedBy": "string",
         "DatabaseOutputs": [
            {
               "DatabaseOptions": {
                  "TableName": "string",
                  "TempDirectory": {
                     "Bucket": "string",
                     "BucketOwner": "string",
                     "Key": "string"
                  }
               },
               "DatabaseOutputMode": "string",
               "GlueConnectionName": "string"
            }
         ],
         "DataCatalogOutputs": [
            {
               "CatalogId": "string",
               "DatabaseName": "string",
               "DatabaseOptions": {
                  "TableName": "string",
                  "TempDirectory": {
                     "Bucket": "string",
                     "BucketOwner": "string",
                     "Key": "string"
                  }
               },
               "Overwrite": boolean,
               "S3Options": {
                  "Location": {
                     "Bucket": "string",
                     "BucketOwner": "string",
                     "Key": "string"
```

```
            }
         },
         "TableName": "string"
      }
   ],
   "DatasetName": "string",
   "EncryptionKeyArn": "string",
   "EncryptionMode": "string",
   "JobSample": {
      "Mode": "string",
      "Size": number
   },
   "LastModifiedBy": "string",
   "LastModifiedDate": number,
   "LogSubscription": "string",
   "MaxCapacity": number,
   "MaxRetries": number,
   "Name": "string",
   "Outputs": [
      {
         "CompressionFormat": "string",
         "Format": "string",
         "FormatOptions": {
            "Csv": {
               "Delimiter": "string"
            }
         },
         "Location": {
            "Bucket": "string",
            "BucketOwner": "string",
            "Key": "string"
         },
         "MaxOutputFiles": number,
         "Overwrite": boolean,
         "PartitionColumns": [ "string" ]
      }
   ],
   "ProjectName": "string",
   "RecipeReference": {
      "Name": "string",
      "RecipeVersion": "string"
   },
   "ResourceArn": "string",
   "RoleArn": "string",
```

```
         "Tags": {
            "string" : "string"
         },
         "Timeout": number,
         "Type": "string",
         "ValidationConfigurations": [
            {
               "RulesetArn": "string",
               "ValidationMode": "string"
            }
         ]
      }
   ],
   "NextToken": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Jobs

已定义的作业列表。

类型：Job 对象数组

NextToken

可以在后续调用中用来检索下一组结果的令牌。

类型：字符串

长度限制：长度下限为 1。最大长度为 2000。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListProjects

列出所有已定义的 DataBrew 项目。

请求语法

```
GET /projects?maxResults=MaxResults&nextToken=NextToken HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

MaxResults

此请求中要返回的最大结果数。

有效范围：最小值为 1。最大值为 100。

NextToken

上一次调用返回的令牌，用于检索下一组结果。

长度限制：长度下限为 1。最大长度为 2000。

请求正文

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Projects": [
      {
         "AccountId": "string",
         "CreateDate": number,
         "CreatedBy": "string",
         "DatasetName": "string",
         "LastModifiedBy": "string",
         "LastModifiedDate": number,
```

```
            "Name": "string",
            "OpenDate": number,
            "OpenedBy": "string",
            "RecipeName": "string",
            "ResourceArn": "string",
            "RoleArn": "string",
            "Sample": {
               "Size": number,
               "Type": "string"
            },
            "Tags": {
               "string" : "string"
            }
         }
      ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Projects

已定义的项目列表。

类型：Project 对象数组

NextToken

可以在后续调用中用来检索下一组结果的令牌。

类型：字符串

长度限制：长度下限为 1。最大长度为 2000。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListRecipes

列出所有已定义的 DataBrew 配方。

请求语法

```
GET /recipes?maxResults=MaxResults&nextToken=NextToken&recipeVersion=RecipeVersion
 HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

MaxResults

此请求中要返回的最大结果数。

有效范围：最小值为 1。最大值为 100。

NextToken

上一次调用返回的令牌，用于检索下一组结果。

长度限制：长度下限为 1。最大长度为 2000。

RecipeVersion

仅返回版本标识符为LATEST_WORKING或的配方LATEST_PUBLISHED。如果省略，RecipeVersion则ListRecipes返回所有LATEST_PUBLISHED配方版本。

有效值：LATEST_WORKING | LATEST_PUBLISHED

长度限制：长度下限为 1。最大长度为 16。

请求正文

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "NextToken": "string",
   "Recipes": [
      {
         "CreateDate": number,
         "CreatedBy": "string",
         "Description": "string",
         "LastModifiedBy": "string",
         "LastModifiedDate": number,
         "Name": "string",
         "ProjectName": "string",
         "PublishedBy": "string",
         "PublishedDate": number,
         "RecipeVersion": "string",
         "ResourceArn": "string",
         "Steps": [
            {
               "Action": {
                  "Operation": "string",
                  "Parameters": {
                     "string" : "string"
                  }
               },
               "ConditionExpressions": [
                  {
                     "Condition": "string",
                     "TargetColumn": "string",
                     "Value": "string"
                  }
               ]
            }
         ],
         "Tags": {
            "string" : "string"
         }
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Recipes

已定义的食谱列表。

类型：Recipe 对象数组

NextToken

可以在后续调用中用来检索下一组结果的令牌。

类型：字符串

长度限制：长度下限为 1。最大长度为 2000。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListRecipeVersions

列出特定 DataBrew 配方的版本，但以下版本除外LATEST_WORKING。

请求语法

```
GET /recipeVersions?maxResults=MaxResults&name=Name&nextToken=NextToken HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

MaxResults

此请求中要返回的最大结果数。

有效范围：最小值为 1。最大值为 100。

Name

要返回其版本信息的配方的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

NextToken

上一次调用返回的令牌，用于检索下一组结果。

长度限制：长度下限为 1。最大长度为 2000。

请求正文

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "NextToken": "string",
   "Recipes": [
      {
         "CreateDate": number,
         "CreatedBy": "string",
         "Description": "string",
         "LastModifiedBy": "string",
         "LastModifiedDate": number,
         "Name": "string",
         "ProjectName": "string",
         "PublishedBy": "string",
         "PublishedDate": number,
         "RecipeVersion": "string",
         "ResourceArn": "string",
         "Steps": [
            {
               "Action": {
                  "Operation": "string",
                  "Parameters": {
                     "string" : "string"
                  }
               },
               "ConditionExpressions": [
                  {
                     "Condition": "string",
                     "TargetColumn": "string",
                     "Value": "string"
                  }
               ]
            }
         ],
         "Tags": {
            "string" : "string"
         }
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Recipes

指定配方的版本列表。

类型：Recipe 对象数组

NextToken

可以在后续调用中用来检索下一组结果的令牌。

类型：字符串

长度限制：长度下限为 1。最大长度为 2000。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListRulesets

列出当前账户中可用的所有规则集或与特定资源（数据集）关联的规则集。

请求语法

```
GET /rulesets?maxResults=MaxResults&nextToken=NextToken&targetArn=TargetArn HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

MaxResults

此请求中要返回的最大结果数。

有效范围：最小值为 1。最大值为 100。

NextToken

由生成的令牌 DataBrew ，用于指定在之前的请求被截断时在哪里继续分页。要获取下一组页面，请传入前一页调用的响应对象中的 NextToken 值。

长度限制：长度下限为 1。最大长度为 2000。

TargetArn

资源（数据集）的亚马逊资源名称 (ARN)。使用此参数表示仅返回与指定资源关联的规则集。

长度约束：最小长度为 20。最大长度为 2048。

请求正文

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
```

```
   "NextToken": "string",
   "Rulesets": [
      {
         "AccountId": "string",
         "CreateDate": number,
         "CreatedBy": "string",
         "Description": "string",
         "LastModifiedBy": "string",
         "LastModifiedDate": number,
         "Name": "string",
         "ResourceArn": "string",
         "RuleCount": number,
         "Tags": {
            "string" : "string"
         },
         "TargetArn": "string"
      }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Rulesets

清单 RulesetItem. RulesetItem 包含规则集的元数据。

类型：RulesetItem 对象数组

NextToken

可以在后续调用中用来检索下一组结果的令牌。

类型：字符串

长度限制：长度下限为 1。最大长度为 2000。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListSchedules

列出已定义的 DataBrew 计划。

请求语法

```
GET /schedules?jobName=JobName&maxResults=MaxResults&nextToken=NextToken HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

JobName

这些时间表所适用的作业名称。

长度限制：长度下限为 1。最大长度为 240。

MaxResults

此请求中要返回的最大结果数。

有效范围：最小值为 1。最大值为 100。

NextToken

上一次调用返回的令牌，用于检索下一组结果。

长度限制：长度下限为 1。最大长度为 2000。

请求正文

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "NextToken": "string",
   "Schedules": [
      {
```

```
            "AccountId": "string",
            "CreateDate": number,
            "CreatedBy": "string",
            "CronExpression": "string",
            "JobNames": [ "string" ],
            "LastModifiedBy": "string",
            "LastModifiedDate": number,
            "Name": "string",
            "ResourceArn": "string",
            "Tags": {
               "string" : "string"
            }
         }
   ]
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Schedules

已定义的计划列表。

类型：Schedule 对象数组

NextToken

可以在后续调用中用来检索下一组结果的令牌。

类型：字符串

长度限制：长度下限为 1。最大长度为 2000。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

ListTagsForResource

列出 DataBrew 资源的所有标签。

请求语法

```
GET /tags/ResourceArn HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

ResourceArn

唯一标识资源的亚马逊资源名称 (ARN) 字符串。 DataBrew

长度约束：最小长度为 20。最大长度为 2048。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Tags

与 DataBrew 资源关联的标签列表。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

错误

有关所有操作的常见错误信息，请参阅常见错误。

InternalServerException

出现内部服务故障。

HTTP 状态代码：500

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK

- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

PublishRecipe

发布新版本的 DataBrew 食谱。

请求语法

```
POST /recipes/name/publishRecipe HTTP/1.1
Content-type: application/json

{
   "Description": "string"
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要发布的食谱的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

Description

此版本食谱中要发布的食谱的描述。

类型：字符串

长度限制：最大长度为 1024。

必需：否

响应语法

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您发布的食谱的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

SendProjectSessionAction

在当前打开的交互式 DataBrew 会话中执行配方步骤。

请求语法

```
PUT /projects/name/sendProjectSessionAction HTTP/1.1
Content-type: application/json

{
   "ClientSessionId": "string",
   "Preview": boolean,
   "RecipeStep": {
      "Action": {
         "Operation": "string",
         "Parameters": {
            "string" : "string"
         }
      },
      "ConditionExpressions": [
         {
            "Condition": "string",
            "TargetColumn": "string",
            "Value": "string"
         }
      ]
   },
   "StepIndex": number,
   "ViewFrame": {
      "Analytics": "string",
      "ColumnRange": number,
      "HiddenColumns": [ "string" ],
      "RowRange": number,
      "StartColumnIndex": number,
      "StartRowIndex": number
   }
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要应用操作的项目名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

ClientSessionId

交互式会话的唯一标识符，该会话当前已打开并已准备就绪。该操作将在此会话中执行。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

模式：`^[a-zA-Z0-9][a-zA-Z0-9-]*$`

必需：否

Preview

如果为 true，则将返回配方步骤的结果，但不会应用。

类型：布尔值

必需：否

RecipeStep

表示要执行的 DataBrew 配方中的一个步骤。

类型：RecipeStep 对象

必需：否

StepIndex

用于预览步骤的索引。此索引用于预览已经应用的步骤的结果，因此生成的视图框架来自视图框架堆栈中较早的部分。

类型：整数

有效范围：最小值为 0。

必需：否

ViewFrame

表示操作期间正在转换的数据。

类型：ViewFrame 对象

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "ActionId": number,
   "Name": "string",
   "Result": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

受该操作影响的项目的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

ActionId

所执行操作的唯一标识符。

类型：整数

Result

一条消息，指示执行操作的结果。

类型：字符串

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包

- AWS 适用于 Ruby V3 的 SDK

StartJobRun

运行作 DataBrew 业。

请求语法

```
POST /jobs/name/startJobRun HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要运行的作业的名称。

长度限制：长度下限为 1。最大长度为 240。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "RunId": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

RunId

系统为此特定作业运行生成的标识符。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包

- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

StartProjectSession

创建交互式会话，使您能够操作 DataBrew项目中的数据。

请求语法

```
PUT /projects/name/startProjectSession HTTP/1.1
Content-type: application/json

{
   "AssumeControl": boolean
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要执行的项目的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

AssumeControl

此值如果为 true，则允许您控制会话，即使其他客户端当前正在访问该项目也是如此。

类型：布尔值

必需：否

响应语法

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "ClientSessionId": "string",
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

要执行的项目的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

ClientSessionId

系统为会话生成的标识符。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

模式：^[a-zA-Z0-9][a-zA-Z0-9-]*$

错误

有关所有操作的常见错误信息，请参阅常见错误。

ConflictException

更新或删除资源可能会导致状态不一致。

HTTP 状态代码：409

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

StopJobRun

停止作业的特定运行。

请求语法

```
POST /jobs/name/jobRun/runId/stopJobRun HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

name

要停止的任务的名称。

长度限制：长度下限为 1。最大长度为 240。

必需：是

runId

要停止的作业的 ID。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "RunId": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

RunId

您停止的作业运行的 ID。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包

- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

TagResource

向 DataBrew 资源添加元数据标签，例如数据集、项目、配方、作业或计划。

请求语法

```
POST /tags/ResourceArn HTTP/1.1
Content-type: application/json

{
   "Tags": {
      "string" : "string"
   }
}
```

URI 请求参数

请求使用以下 URI 参数。

ResourceArn

应向其添加标签的 DataBrew 资源。此参数的值是亚马逊资源名称 (ARN)。对于 DataBrew，您可以为数据集、作业、项目或配方添加标签。

长度约束：最小长度为 20。最大长度为 2048。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

Tags

要为资源分配一个或多个标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：是

响应语法

```
HTTP/1.1 200
```

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

有关所有操作的常见错误信息，请参阅常见错误。

InternalServerException

出现内部服务故障。

HTTP 状态代码：500

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK

- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UntagResource

从 DataBrew 资源中移除元数据标签。

请求语法

```
DELETE /tags/ResourceArn?tagKeys=TagKeys HTTP/1.1
```

URI 请求参数

请求使用以下 URI 参数。

ResourceArn

要从中移除一个或多个标签的 DataBrew 资源。此参数的值是亚马逊资源名称 (ARN)。

长度约束：最小长度为 20。最大长度为 2048。

必需：是

TagKeys

要删除的一个或多个标签的标签密钥（名称）。

数组成员：最少 1 个物品。最多 200 项。

长度限制：长度下限为 1。长度上限为 128。

必需：是

请求体

该请求没有请求正文。

响应语法

```
HTTP/1.1 200
```

响应元素

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP 200 响应。

错误

有关所有操作的常见错误信息，请参阅常见错误。

InternalServerException

出现内部服务故障。

HTTP 状态代码：500

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版 SDK
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UpdateDataset

修改现有 DataBrew 数据集的定义。

请求语法

```
PUT /datasets/name HTTP/1.1
Content-type: application/json

{
   "Format": "string",
   "FormatOptions": {
      "Csv": {
         "Delimiter": "string",
         "HeaderRow": boolean
      },
      "Excel": {
         "HeaderRow": boolean,
         "SheetIndexes": [ number ],
         "SheetNames": [ "string" ]
      },
      "Json": {
         "MultiLine": boolean
      }
   },
   "Input": {
      "DatabaseInputDefinition": {
         "DatabaseTableName": "string",
         "GlueConnectionName": "string",
         "QueryString": "string",
         "TempDirectory": {
            "Bucket": "string",
            "BucketOwner": "string",
            "Key": "string"
         }
      },
      "DataCatalogInputDefinition": {
         "CatalogId": "string",
         "DatabaseName": "string",
         "TableName": "string",
         "TempDirectory": {
            "Bucket": "string",
            "BucketOwner": "string",
```

```
            "Key": "string"
         }
      },
      "Metadata": {
         "SourceArn": "string"
      },
      "S3InputDefinition": {
         "Bucket": "string",
         "BucketOwner": "string",
         "Key": "string"
      }
   },
   "PathOptions": {
      "FilesLimit": {
         "MaxFiles": number,
         "Order": "string",
         "OrderedBy": "string"
      },
      "LastModifiedDateCondition": {
         "Expression": "string",
         "ValuesMap": {
            "string" : "string"
         }
      },
      "Parameters": {
         "string" : {
            "CreateColumn": boolean,
            "DatetimeOptions": {
               "Format": "string",
               "LocaleCode": "string",
               "TimezoneOffset": "string"
            },
            "Filter": {
               "Expression": "string",
               "ValuesMap": {
                  "string" : "string"
               }
            },
            "Name": "string",
            "Type": "string"
         }
      }
   }
```

```
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要更新的数据集的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

Input

表示有关 DataBrew 如何在 Amazon S3 AWS Glue Data Catalog 或 Amazon S3 中查找数据的信息。

类型：Input 对象

必需：是

Format

从 Amazon S3 文件或文件夹创建的数据集的文件格式。

类型：字符串

有效值：`CSV | JSON | PARQUET | EXCEL | ORC`

必需：否

FormatOptions

表示一组选项，这些选项用于定义逗号分隔值 (CSV)、Excel 或 JSON 输入的结构。

类型：FormatOptions 对象

必需：否

PathOptions

一组选项，用于定义如何 DataBrew 解释数据集的 Amazon S3 路径。

类型：PathOptions 对象

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您更新的数据集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

AccessDeniedException

对指定资源的访问被拒绝。

HTTP 状态代码：403

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UpdateProfileJob

修改现有档案作业的定义。

请求语法

```
PUT /profileJobs/name HTTP/1.1
Content-type: application/json

{
   "Configuration": {
      "ColumnStatisticsConfigurations": [
         {
            "Selectors": [
               {
                  "Name": "string",
                  "Regex": "string"
               }
            ],
            "Statistics": {
               "IncludedStatistics": [ "string" ],
               "Overrides": [
                  {
                     "Parameters": {
                        "string" : "string"
                     },
                     "Statistic": "string"
                  }
               ]
            }
         }
      ],
      "DatasetStatisticsConfiguration": {
         "IncludedStatistics": [ "string" ],
         "Overrides": [
            {
               "Parameters": {
                  "string" : "string"
               },
               "Statistic": "string"
            }
         ]
      },
```

```
      "EntityDetectorConfiguration": {
         "AllowedStatistics": [
            {
               "Statistics": [ "string" ]
            }
         ],
         "EntityTypes": [ "string" ]
      },
      "ProfileColumns": [
         {
            "Name": "string",
            "Regex": "string"
         }
      ]
   },
   "EncryptionKeyArn": "string",
   "EncryptionMode": "string",
   "JobSample": {
      "Mode": "string",
      "Size": number
   },
   "LogSubscription": "string",
   "MaxCapacity": number,
   "MaxRetries": number,
   "OutputLocation": {
      "Bucket": "string",
      "BucketOwner": "string",
      "Key": "string"
   },
   "RoleArn": "string",
   "Timeout": number,
   "ValidationConfigurations": [
      {
         "RulesetArn": "string",
         "ValidationMode": "string"
      }
   ]
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要更新的任务的名称。

长度限制：长度下限为 1。最大长度为 240。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

OutputLocation

表示 Amazon S3 位置（存储桶名称、存储桶拥有者和对象密钥），DataBrew 可以在其中读取输入数据或写入任务的输出。

类型：S3Location 对象

必需：是

RoleArn

DataBrew 运行任务时要假设的 AWS Identity and Access Management (IAM) 角色的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：是

Configuration

配置文件作业的配置。用于选择列、进行评估和覆盖评估的默认参数。当配置为空时，分析作业将使用默认设置运行。

类型：ProfileConfiguration 对象

必需：否

EncryptionKeyArn

用于保护任务的加密密钥的 Amazon 资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

EncryptionMode

作业的加密模式包括以下几种：

- SSE-KMS-使用由 AWS KMS管理的密钥进行服务器端加密。
- SSE-S3 - 使用 Amazon S3 托管密钥进行服务器端加密。

类型：字符串

有效值：SSE-KMS | SSE-S3

必需：否

JobSample

仅适用于分析任务的配置示例。确定要执行分析作业的行数。如果没有为分析作业提供 JobSample 值，则将使用默认值。模式参数的默认值为 CUSTOM_ROWS，大小参数的默认值为 20000。

类型：JobSample 对象

必需：否

LogSubscription

为任务启用或禁用 Amazon CloudWatch 日志记录。如果启用了日志记录，则为每个作业运行 CloudWatch 写入一个日志流。

类型：字符串

有效值：ENABLE | DISABLE

必需：否

MaxCapacity

作业处理数据时 DataBrew 可使用的最大计算节点数。

类型：整数

必需：否

MaxRetries

作业运行失败后重试此作业的最大次数。

类型：整数

有效范围：最小值为 0。

必需：否

Timeout

作业的超时（以分钟为单位）。如果作业的运行时间超出此超时时间，作业将以 TIMEOUT 状态结束。

类型：整数

有效范围：最小值为 0。

必需：否

ValidationConfigurations

应用于配置文件作业的验证配置列表。

类型：ValidationConfiguration 对象数组

数组成员：最少 1 个物品。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

已更新的任务的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

错误

有关所有操作的常见错误信息，请参阅常见错误。

AccessDeniedException

对指定资源的访问被拒绝。

HTTP 状态代码：403

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2

- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UpdateProject

修改现有 DataBrew 项目的定义。

请求语法

```
PUT /projects/name HTTP/1.1
Content-type: application/json

{
   "RoleArn": "string",
   "Sample": {
      "Size": number,
      "Type": "string"
   }
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要更新的项目名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

RoleArn

要为该请求担任的 IAM 角色的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：是

Sample

表示用于交互式数据分析 DataBrew 的样本数量和采样类型。

类型：Sample 对象

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "LastModifiedDate": number,
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您更新的项目的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

LastModifiedDate

项目上次修改的日期和时间。

类型：时间戳

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UpdateRecipe

修改 DataBrew配方LATEST_WORKING版本的定义。

请求语法

```
PUT /recipes/name HTTP/1.1
Content-type: application/json

{
   "Description": "string",
   "Steps": [
      {
         "Action": {
            "Operation": "string",
            "Parameters": {
               "string" : "string"
            }
         },
         "ConditionExpressions": [
            {
               "Condition": "string",
               "TargetColumn": "string",
               "Value": "string"
            }
         ]
      }
   ]
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要更新的配方的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

Description

对食谱的描述。

类型：字符串

长度限制：最大长度为 1024。

必需：否

Steps

配方要执行的一个或多个步骤。每个步骤都由一个动作以及该操作应在何种条件下成功组成。

类型：RecipeStep 对象数组

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

已更新的食谱的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UpdateRecipeJob

修改现有 DataBrew 配方任务的定义。

请求语法

```
PUT /recipeJobs/name HTTP/1.1
Content-type: application/json

{
   "DatabaseOutputs": [
      {
         "DatabaseOptions": {
            "TableName": "string",
            "TempDirectory": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
         },
         "DatabaseOutputMode": "string",
         "GlueConnectionName": "string"
      }
   ],
   "DataCatalogOutputs": [
      {
         "CatalogId": "string",
         "DatabaseName": "string",
         "DatabaseOptions": {
            "TableName": "string",
            "TempDirectory": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
         },
         "Overwrite": boolean,
         "S3Options": {
            "Location": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            }
```

```
            },
            "TableName": "string"
         }
      ],
      "EncryptionKeyArn": "string",
      "EncryptionMode": "string",
      "LogSubscription": "string",
      "MaxCapacity": number,
      "MaxRetries": number,
      "Outputs": [
         {
            "CompressionFormat": "string",
            "Format": "string",
            "FormatOptions": {
               "Csv": {
                  "Delimiter": "string"
               }
            },
            "Location": {
               "Bucket": "string",
               "BucketOwner": "string",
               "Key": "string"
            },
            "MaxOutputFiles": number,
            "Overwrite": boolean,
            "PartitionColumns": [ "string" ]
         }
      ],
      "RoleArn": "string",
      "Timeout": number
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要更新的任务的名称。

长度限制：长度下限为 1。最大长度为 240。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

RoleArn

DataBrew 运行任务时要假设的 AWS Identity and Access Management (IAM) 角色的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：是

DatabaseOutputs

表示 JDBC 数据库输出对象的列表，该对象定义了要写入的 DataBrew 配方作业的输出目标。

类型：DatabaseOutput 对象数组

数组成员：最少 1 个物品。

必需：否

DataCatalogOutputs

一个或多个工件，表示运行作业的 AWS Glue Data Catalog 输出。

类型：DataCatalogOutput 对象数组

数组成员：最少 1 个物品。

必需：否

EncryptionKeyArn

用于保护任务的加密密钥的 Amazon 资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

EncryptionMode

作业的加密模式包括以下几种：

- SSE-KMS-使用由 AWS KMS管理的密钥进行服务器端加密。
- SSE-S3 - 使用 Amazon S3 托管密钥进行服务器端加密。

类型：字符串

有效值：SSE-KMS | SSE-S3

必需：否

LogSubscription

为任务启用或禁用 Amazon CloudWatch 日志记录。如果启用了日志记录，则为每个作业运行 CloudWatch 写入一个日志流。

类型：字符串

有效值：ENABLE | DISABLE

必需：否

MaxCapacity

任务处理数据时 DataBrew 可以消耗的最大节点数。

类型：整数

必需：否

MaxRetries

作业运行失败后重试此作业的最大次数。

类型：整数

有效范围：最小值为 0。

必需：否

Outputs

代表作业运行时 输出的一个或多个构件。

类型：Output 对象数组

数组成员：最少 1 个物品。

必需：否

Timeout

作业的超时（以分钟为单位）。如果作业的运行时间超出此超时时间，作业将以 TIMEOUT 状态结束。

类型：整数

有效范围：最小值为 0。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

您更新的任务的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

错误

有关所有操作的常见错误信息，请参阅常见错误。

AccessDeniedException

对指定资源的访问被拒绝。

HTTP 状态代码：403

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UpdateRuleset

更新指定的规则集。

请求语法

```
PUT /rulesets/name HTTP/1.1
Content-type: application/json

{
   "Description": "string",
   "Rules": [
      {
         "CheckExpression": "string",
         "ColumnSelectors": [
            {
               "Name": "string",
               "Regex": "string"
            }
         ],
         "Disabled": boolean,
         "Name": "string",
         "SubstitutionMap": {
            "string" : "string"
         },
         "Threshold": {
            "Type": "string",
            "Unit": "string",
            "Value": number
         }
      }
   ]
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要更新的规则集的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

Rules

使用规则集定义的规则列表。规则包括一项或多项要对 DataBrew 数据集进行验证的校验。

类型：Rule 对象数组

数组成员：最少 1 个物品。

必需：是

Description

规则集的描述。

类型：字符串

长度限制：最大长度为 1024。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

更新后的规则集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

UpdateSchedule

修改现有 DataBrew 计划的定义。

请求语法

```
PUT /schedules/name HTTP/1.1
Content-type: application/json

{
   "CronExpression": "string",
   "JobNames": [ "string" ]
}
```

URI 请求参数

请求使用以下 URI 参数。

name

要更新的时间表的名称。

长度约束：最小长度为 1。最大长度为 255。

必需：是

请求体

请求接受采用 JSON 格式的以下数据。

CronExpression

运行作业的日期、日期和时间。有关更多信息，请参阅《 AWS Glue DataBrew 开发人员指南》中的 Cron 表达式。

类型：字符串

长度限制：长度下限为 1。最大长度为 512。

必需：是

JobNames

要为此计划运行的一个或多个作业的名称。

类型：字符串数组

数组成员：最多 50 项。

长度限制：长度下限为 1。最大长度为 240。

必需：否

响应语法

```
HTTP/1.1 200
Content-type: application/json

{
   "Name": "string"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

服务以 JSON 格式返回以下数据。

Name

已更新的时间表的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

错误

有关所有操作的常见错误信息，请参阅常见错误。

ResourceNotFoundException

找不到一个或多个资源。

HTTP 状态代码：404

ServiceQuotaExceededException

超过了服务配额。

HTTP 状态代码：402

ValidationException

此请求的输入参数未通过验证。

HTTP 状态代码：400

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 命令行界面
- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go v2 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS JavaScript V3 版软件开发工具包
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 SDK

数据类型

支持以下数据类型：

- AllowedStatistics
- ColumnSelector
- ColumnStatisticsConfiguration
- ConditionExpression
- CsvOptions

- CsvOutputOptions
- DatabaseInputDefinition
- DatabaseOutput
- DatabaseTableOutputOptions
- DataCatalogInputDefinition
- DataCatalogOutput
- Dataset
- DatasetParameter
- DatetimeOptions
- EntityDetectorConfiguration
- ExcelOptions
- FilesLimit
- FilterExpression
- FormatOptions
- Input
- Job
- JobRun
- JobSample
- JsonOptions
- Metadata
- Output
- OutputFormatOptions
- PathOptions
- ProfileConfiguration
- Project
- Recipe
- RecipeAction
- RecipeReference
- RecipeStep
- RecipeVersionErrorDetail

- Rule
- RulesetItem
- S3Location
- S3TableOutputOptions
- Sample
- Schedule
- StatisticOverride
- StatisticsConfiguration
- Threshold
- ValidationConfiguration
- ViewFrame

AllowedStatistics

允许在包含检测到的实体的列上运行的统计信息的配置。如果未定义，则不会对包含检测到的实体的列计算统计信息。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Statistics

允许包含检测到的实体的列的一个或多个列统计信息。

类型：字符串数组

数组成员：最少 1 个物品。

长度限制：长度下限为 1。长度上限为 128。

模式：`^[A-Z\_]+$`

必需：是

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

ColumnSelector

数据集中用于配置文件作业配置的列的选择器。一个选择器包含列名或正则表达式。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Name

数据集中的列的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

Regex

用于从数据集中选择列的正则表达式。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

ColumnStatisticsConfiguration

配置剖析作业的列评估。 ColumnStatisticsConfiguration 可用于为特定列选择计算值和覆盖评估参数。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Statistics

评估配置。统计信息可用于选择评估并覆盖评估的参数。

类型：StatisticsConfiguration 对象

必需：是

Selectors

列选择器列表。选择器可用于从数据集中选择列。当选择器未定义时，配置将应用于所有支持的列。

类型：ColumnSelector 对象数组

数组成员：最少 1 个物品。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

ConditionExpression

表示评估为 true 或 false 的单个条件。

条件与食谱操作一起使用。该操作仅对条件求值为 true 的列值执行。

如果食谱需要多个条件，则食谱必须指定多个 `ConditionExpression` 元素。在执行食谱操作之前，每个条件都首先应用于数据集中的行。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Condition

适用于食谱操作的特定条件。有关更多信息，请参阅《 AWS Glue DataBrew 开发人员指南》中的配方结构。

类型：字符串

长度限制：长度下限为 1。长度上限为 128。

模式：`^[A-Z\_]+$`

必需：是

TargetColumn

要应用此条件的列。

类型：字符串

长度限制：长度下限为 1。最大长度为 1024。

必需：是

Value

条件必须对其进行评估才能成功的值。

类型：字符串

长度限制：最大长度为 1024。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

CsvOptions

表示一组选项，用于定义从逗号分隔值 (CSV) 文件创建数据集时如何 DataBrew 读取该文件。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Delimiter

指定 CSV 文件中使用的分隔符的单个字符。

类型：字符串

长度限制：固定长度为 1。

必需：否

HeaderRow

指定是否将文件中的第一行解析为标题的变量。如果此值为 false，则列名称将自动生成。

类型：布尔值

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

CsvOutputOptions

表示一组选项，用于定义如何 DataBrew 写入逗号分隔值 (CSV) 文件。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Delimiter

指定用于创建 CSV 作业输出的分隔符的单个字符。

类型：字符串

长度限制：固定长度为 1。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

DatabaseInputDefinition

存储在数据库中的数据集输入文件的连接信息。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

GlueConnectionName

存储目标数据库连接信息的连接。 AWS Glue

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

DatabaseTableName

目标数据库中的表。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

QueryString

针对提供的 AWS Glue 连接运行的自定义 SQL。此 SQL 将用作 DataBrew 项目和作业的输入。

类型：字符串

长度限制：长度下限为 1。最大长度为 10000。

必需：否

TempDirectory

表示 Amazon S3 位置（存储桶名称、存储桶拥有者和对象密钥），DataBrew 可以在其中读取输入数据或写入任务的输出。

类型：S3Location 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

DatabaseOutput

表示一个 JDBC 数据库输出对象，该对象定义了要写入的 DataBrew 配方作业的输出目标。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

DatabaseOptions

表示用于指定如何以及在何处 DataBrew 写入由配方作业生成的数据库输出的选项。

类型：DatabaseTableOutputOptions 对象

必需：是

GlueConnectionName

存储目标数据库连接信息的连接。 AWS Glue

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

DatabaseOutputMode

要写入数据库的输出模式。当前支持的选项：NEW_TABLE。

类型：字符串

有效值：NEW_TABLE

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

DatabaseTableOutputOptions

表示用于指定如何以及在何处 DataBrew 写入由配方作业生成的数据库输出的选项。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

TableName

DataBrew 将在数据库中创建表名的前缀。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

TempDirectory

表示 DataBrew 可以存储中间结果的 Amazon S3 位置（存储桶名称和对象密钥）。

类型：S3Location 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

DataCatalogInputDefinition

表示存储在中的元数据在 DataBrew数据集中 AWS Glue Data Catalog 是如何定义的。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

DatabaseName

数据目录中数据库的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

TableName

数据目录中数据库表的名称。此表对应于一个 DataBrew数据集。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

CatalogId

的唯一标识符 AWS 账户 ，用于保存存储数据的数据目录。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

TempDirectory

表示 DataBrew 可以存储中间结果的 Amazon 地点。

类型：S3Location 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

DataCatalogOutput

表示用于指定配方作业生成的输出的 AWS Glue Data Catalog DataBrew 写入方式和写入位置的选项。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

DatabaseName

数据目录中数据库的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

TableName

数据目录中表的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

CatalogId

的唯一标识符 AWS 账户 ，用于保存存储数据的数据目录。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

DatabaseOptions

表示用于指定如何以及在何处 DataBrew 写入由配方作业生成的数据库输出的选项。

类型：DatabaseTableOutputOptions 对象

必需：否

Overwrite

一个值，如果为 true，则表示为输出指定位置的任何数据将被新输出覆盖。不支持 DatabaseOptions。

类型：布尔值

必需：否

S3Options

表示选项，用于指定如何以及在何处 DataBrew 写入由配方任务生成的 Amazon S3 输出。

类型：S3TableOutputOptions 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Dataset

表示可由处理的数据集 DataBrew。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Input

有关 DataBrew 如何在 AWS Glue Data Catalog 或 Amazon S3 中找到数据集的信息。

类型：Input 对象

必需：是

Name

数据集的唯一名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

AccountId

拥有数据集的 AWS 账户的 ID。

类型：字符串

长度限制：长度上限为 255。

必需：否

CreateDate

数据集的创建日期和时间。

类型：时间戳

必需：否

CreatedBy

创建数据集的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

Format

从 Amazon S3 文件或文件夹创建的数据集的文件格式。

类型：字符串

有效值：`CSV | JSON | PARQUET | EXCEL | ORC`

必需：否

FormatOptions

一组选项，用于定义如何 DataBrew 解释数据集中的数据。

类型：FormatOptions 对象

必需：否

LastModifiedBy

上次修改数据集的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

LastModifiedDate

数据集的最后修改日期和时间。

类型：时间戳

必需：否

PathOptions

一组选项，用于定义如何 DataBrew 解释数据集的 Amazon S3 路径。

类型：PathOptions 对象

必需：否

ResourceArn

数据集的唯一亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

Source

数据集的数据位置，可以是 Amazon S3 或 AWS Glue Data Catalog。

类型：字符串

有效值：`S3 | DATA-CATALOG | DATABASE`

必需：否

Tags

已应用于数据集的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2

- AWS 适用于 Ruby V3 的 SDK

DatasetParameter

表示一个数据集参数，用于定义数据集 Amazon S3 路径中参数的类型和条件。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Name

在数据集的 Amazon S3 路径中使用的参数的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

Type

数据集参数的类型可以是“String”、“Number”或“Datetime”之一。

类型：字符串

有效值：`Datetime | Number | String`

必需：是

CreateColumn

可选的布尔值，用于定义是否应使用此参数的捕获值在数据集中创建新列。

类型：布尔值

必需：否

DatetimeOptions

其他参数选项，例如格式和时区。对日期时间参数，为必需项。

类型：DatetimeOptions 对象

必需：否

Filter

用于将其他匹配条件应用于参数的可选筛选条件表达式结构。

类型：FilterExpression 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

DatetimeOptions

表示其他选项，这些选项用于正确解释数据集 Amazon S3 路径中使用的日期时间参数。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Format

必需选项，定义 Amazon S3 路径中日期参数使用的日期时间格式。应仅使用支持的日期时间说明符和分隔字符，所有字面的 a-z 或 A-Z 字符都应使用单引号进行转义。例如，"MM.dd.yyyy-'at'-HH:mm"。

类型：字符串

长度限制：最小长度为 2。最大长度为 100。

必需：是

LocaleCode

非美国区域设置代码的可选值，正确解释某些日期格式时需要此选项。

类型：字符串

长度限制：最小长度为 2。最大长度为 100。

模式：`^[A-Za-z0-9_\.#@\-]+$`

必需：否

TimezoneOffset

Amazon S3 路径中日期时间参数值的时区偏移的可选值。如果此参数的格式包括时区字段，则不应使用。如果未指定偏移，则认为是 UTC。

类型：字符串

长度限制：长度下限为 1。最大长度为 6。

模式：`^(Z|[-+](\d|\d{2}|\d{2}:?\d{2}))$`

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

EntityDetectorConfiguration

配置文件作业的实体检测配置。未定义时，禁用实体检测。

内容

> **Note**
> 下表中，首先描述的是必需参数。

EntityTypes

要检测的实体类型。可以是以下任一种：

- USA_SSN
- EMAIL
- USA_ITIN
- USA_PASSPORT_NUMBER
- PHONE_NUMBER
- USA_DRIVING_LICENSE
- BANK_ACCOUNT
- CREDIT_CARD
- IP_ADDRESS
- MAC_ADDRESS
- USA_DEA_NUMBER
- USA_HCPCS_CODE
- USA_NATIONAL_PROVIDER_IDENTIFIER
- USA_NATIONAL_DRUG_CODE
- USA_HEALTH_INSURANCE_CLAIM_NUMBER
- USA_MEDICARE_BENEFICIARY_IDENTIFIER
- USA_CPT_CODE
- PERSON_NAME
- DATE

还支持实体类型组 USA_ALL，它包括除了 PERSON_NAME 和 DATE 之外的所有上述实体类型。

类型：字符串数组

数组成员：最少 1 个物品。

长度限制：长度下限为 1。长度上限为 128。

模式：`^[A-Z_][A-Z\\d_]*$`

必需：是

AllowedStatistics

允许在包含检测到的实体的列上运行的统计信息的配置。如果未定义，则不会对包含检测到的实体的列计算统计信息。

类型：AllowedStatistics 对象数组

数组成员：最少 1 个物品。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

ExcelOptions

表示一组选项，用于定义在从 Microsoft Excel 文件创建数据集时如何 DataBrew 解释 Microsoft Excel 文件。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

HeaderRow

指定是否将文件中的第一行解析为标题的变量。如果此值为 false，则列名称将自动生成。

类型：布尔值

必需：否

SheetIndexes

将包含在数据集中的 Excel 文件中的一个或多个工作表编号。

类型：整数数组

数组成员：固定数量为 1 项。

有效范围：最小值为 0。最大值为 200。

必需：否

SheetNames

将包含在数据集中的 Excel 文件中有一个或多个命名工作表。

类型：字符串数组

数组成员：固定数量为 1 项。

长度限制：长度下限为 1。最大长度为 31。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

FilesLimit

表示对应从所连 Amazon S3 路径为数据集选择的 Amazon S3 文件数量施加的限制。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

MaxFiles

要选择的 Amazon S3 文件的数量。

类型：整数

有效范围：最小值为 1。

必需：是

Order

在选择 Amazon S3 文件前，对其进行排序时使用的条件。默认情况下使用 DESCENDING 顺序，即首先选择最近的文件。另一个可能的值是升序。

类型：字符串

有效值：`DESCENDING | ASCENDING`

必需：否

OrderedBy

在选择 Amazon S3 文件前，对其进行排序时使用的条件。默认情况下，使用 LAST_MODIFIED_DATE 作为排序条件。目前，这是唯一允许的值。

类型：字符串

有效值：`LAST_MODIFIED_DATE`

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

FilterExpression

表示用于定义参数条件的结构。支持的条件如下所述：《 AWS Glue DataBrew 开发人员指南》中动态数据集的支持条件。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Expression

包括条件名称并后跟替换变量的表达式，可能与其他条件一同分组并组合使用。例如，“(starts_with :prefix1 or starts_with :prefix2) and (ends_with :suffix1 or ends_with :suffix2)”。替换变量应以“:”符号开头。

类型：字符串

长度限制：最小长度为 4。长度上限为 1024。

模式：`^[<>0-9A-Za-z_.,:)(!= ]+$`

必需：是

ValuesMap

替换变量名称与此筛选条件表达式中所用值的映射。

类型：字符串到字符串映射

密钥长度限制：最小长度为 2。长度上限为 128。

键模式：`^:[A-Za-z0-9_]+$`

值长度约束：最大长度为 1024。

必需：是

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

FormatOptions

表示一组选项，这些选项用于定义逗号分隔值 (CSV)、Excel 或 JSON 输入的结构。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Csv

定义如何解释 CSV 输入的选项 DataBrew。

类型：CsvOptions 对象

必需：否

Excel

定义如何解释 Excel 输入的选项 DataBrew。

类型：ExcelOptions 对象

必需：否

Json

定义如何解释 JSON 输入的选项 DataBrew。

类型：JsonOptions 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Input

表示有关 DataBrew 如何在 Amazon S3 AWS Glue Data Catalog 或 Amazon S3 中查找数据的信息。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

DatabaseInputDefinition

存储在数据库中的数据集输入文件的连接信息。

类型：DatabaseInputDefinition 对象

必需：否

DataCatalogInputDefinition

数据的 AWS Glue Data Catalog 参数。

类型：DataCatalogInputDefinition 对象

必需：否

Metadata

包含特定数据集所需的其他资源信息。

类型：Metadata 对象

必需：否

S3InputDefinition

存储数据的 Amazon S3 位置。

类型：S3Location 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Job

表示 DataBrew 作业的所有属性。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Name

作业的唯一名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

必需：是

AccountId

拥有该任务的 AWS 账号的 ID。

类型：字符串

长度限制：长度上限为 255。

必需：否

CreateDate

创建作业的日期和时间。

类型：时间戳

必需：否

CreatedBy

创建任务的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

DatabaseOutputs

表示 JDBC 数据库输出对象的列表，该对象定义了要写入的 DataBrew 配方作业的输出目标。

类型：DatabaseOutput 对象数组

数组成员：最少 1 个物品。

必需：否

DataCatalogOutputs

一个或多个工件，表示运行作业的 AWS Glue Data Catalog 输出。

类型：DataCatalogOutput 对象数组

数组成员：最少 1 个物品。

必需：否

DatasetName

作业要处理的数据集。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

EncryptionKeyArn

用于保护作业输出的加密密钥的 Amazon 资源名称（ARN）。有关更多信息，请参阅加密作业写入 DataBrew 的数据

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

EncryptionMode

作业的加密模式包括以下几种：

- SSE-KMS-使用由 AWS KMS管理的密钥进行服务器端加密。

- SSE-S3 - 使用 Amazon S3 托管密钥进行服务器端加密。

类型：字符串

有效值：SSE-KMS | SSE-S3

必需：否

JobSample

仅用于配置文件作业的样本配置，其确定运行配置文件作业的行数。如果未提供 JobSample 值，则使用默认值。模式参数的默认值为 CUSTOM_ROWS，大小参数的默认值为 20,000。

类型：JobSample 对象

必需：否

LastModifiedBy

上次修改任务的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

LastModifiedDate

作业的修改日期和时间。

类型：时间戳

必需：否

LogSubscription

Amazon CloudWatch 登录任务的当前状态。

类型：字符串

有效值：ENABLE | DISABLE

必需：否

MaxCapacity

作业处理数据时可以使用的最大节点数。

类型：整数

必需：否

MaxRetries

作业运行失败后重试此作业的最大次数。

类型：整数

有效范围：最小值为 0。

必需：否

Outputs

代表作业运行时输出的一个或多个构件。

类型：Output 对象数组

数组成员：最少 1 个物品。

必需：否

ProjectName

与作业关联的项目的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

RecipeReference

作业运行的一组步骤。

类型：RecipeReference 对象

必需：否

ResourceArn

任务的唯一亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

RoleArn

此作业将担任的角色的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

Tags

已应用于作业的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

Timeout

作业的超时（以分钟为单位）。如果作业的运行时间超出此超时时间，作业将以 TIMEOUT 状态结束。

类型：整数

有效范围：最小值为 0。

必需：否

Type

作业类型必须为以下类型之一：

- PROFILE - 用于分析数据集、确定其大小、数据类型、数据分布等的作业。
- RECIPE - 将一个或多个转换应用于数据集的作业。

类型：字符串

有效值：PROFILE | RECIPE

必需：否

ValidationConfigurations

应用于配置文件作业的验证配置列表。

类型：ValidationConfiguration 对象数组

数组成员：最少 1 个物品。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

JobRun

表示 DataBrew 作业的一次运行。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Attempt

尝试运行作业 DataBrew 的次数。

类型：整数

必需：否

CompletedOn

任务完成处理的日期和时间。

类型：时间戳

必需：否

DatabaseOutputs

表示 JDBC 数据库输出对象的列表，该对象定义了要写入的 DataBrew 配方作业的输出目标。

类型：DatabaseOutput 对象数组

数组成员：最少 1 个物品。

必需：否

DataCatalogOutputs

一个或多个工件，用于表示运行作业的 AWS Glue Data Catalog 输出。

类型：DataCatalogOutput 对象数组

数组成员：最少 1 个物品。

必需：否

DatasetName

要处理的作业的数据集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

ErrorMessage

一条消息，指示作业运行时遇到错误（如果有）。

类型：字符串

必需：否

ExecutionTime

作业运行消耗资源的时间（以秒为单位）。

类型：整数

必需：否

JobName

此次运行期间正在处理的作业的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 240。

必需：否

JobSample

仅用于配置文件作业的样本配置，其确定运行配置文件作业的行数。如果未提供 `JobSample` 值，则使用默认值。模式参数的默认值为 CUSTOM_ROWS，大小参数的默认值为 20,000。

类型：JobSample 对象

必需：否

LogGroupName

Amazon CloudWatch 日志组的名称，作业在运行时写入诊断消息。

类型：字符串

长度限制：长度下限为 1。最大长度为 512。

必需：否

LogSubscription

Amazon CloudWatch 记录任务运行的当前状态。

类型：字符串

有效值：`ENABLE | DISABLE`

必需：否

Outputs

作业运行中的一个或多个输出工件。

类型：Output 对象数组

数组成员：最少 1 个物品。

必需：否

RecipeReference

作业处理的一组步骤。

类型：RecipeReference 对象

必需：否

RunId

作业运行的唯一标识符。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

StartedBy

启动任务运行的用户的 Amazon 资源名称 (ARN)。

类型：字符串

必需：否

StartedOn

作业运行开始的日期和时间。

类型：时间戳

必需：否

State

作业运行实体本身的当前状态。

类型：字符串

有效值：`STARTING | RUNNING | STOPPING | STOPPED | SUCCEEDED | FAILED | TIMEOUT`

必需：否

ValidationConfigurations

应用于配置文件作业运行的验证配置列表。

类型：ValidationConfiguration 对象数组

数组成员：最少 1 个物品。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

JobSample

仅用于配置文件作业的样本配置，其确定运行配置文件作业的行数。如果未提供 JobSample 值，则使用默认值。模式参数的默认值为 CUSTOM_ROWS，大小参数的默认值为 20,000。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Mode

一个值，用于确定配置文件作业是在整个数据集上运行还是在指定数量的行上运行。该值必须是以下内容之一：

- FULL_DATASET - 配置文件作业在整个数据集上运行。
- CUSTOM_ROWS - 配置文件作业在 Size 参数中指定的很多行上运行。

类型：字符串

有效值：FULL_DATASET | CUSTOM_ROWS

必需：否

Size

Size 参数仅在模式为 CUSTOM_ROWS 时需要。配置文件作业在指定的行数上运行。最大的大小值为 Long.MAX_VALUE。

Long.MAX_VALUE = 9223372036854775807

类型：长整型

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK

- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

JsonOptions

表示特定于 JSON 的选项，这些选项定义了如何解释输入。 AWS Glue DataBrew

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

MultiLine

指定 JSON 输入是否包含嵌入的新行字符的值。

类型：布尔值

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Metadata

包含特定数据集所需的其他资源信息。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

SourceArn

与数据集关联的 Amazon 资源名称（ARN）。目前， DataBrew 仅支持来自亚马逊 AppFlow的 ARN。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Output

表示选项，用于指定在 Amazon S3 中如何以及在何处 DataBrew 写入由配方任务或配置文件任务生成的输出。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Location

作业在 Amazon S3 中写入输出的位置。

类型：S3Location 对象

必需：是

CompressionFormat

用于压缩作业输出文本的压缩算法。

类型：字符串

有效值：`GZIP | LZ4 | SNAPPY | BZIP2 | DEFLATE | LZO | BROTLI | ZSTD | ZLIB`

必需：否

Format

作业输出的数据格式。

类型：字符串

有效值：`CSV | JSON | PARQUET | GLUEPARQUET | AVRO | ORC | XML | TABLEAUHYPER`

必需：否

FormatOptions

表示定义如何 DataBrew 格式化作业输出文件的选项。

类型：OutputFormatOptions 对象

必需：否

MaxOutputFiles

作业生成并写入输出文件夹的最大文件数。对于按列分区的输出，该 MaxOutputFiles 值为每个分区的最大文件数。

类型：整数

有效范围：最小值为 1。最大值为 999。

必需：否

Overwrite

一个值，如果为 true，则表示为输出指定位置的任何数据将被新输出覆盖。

类型：布尔值

必需：否

PartitionColumns

作业输出的一个或多个分区列的名称。

类型：字符串数组

数组成员：最多 200 项。

长度约束：最小长度为 1。最大长度为 255。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

OutputFormatOptions

表示定义逗号分隔值 (CSV) 作业输出结构的一组选项。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Csv

表示定义逗号分隔值 (CSV) 作业输出结构的一组选项。

类型：CsvOutputOptions 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

PathOptions

表示一组选项，用于定义如何为数据集中给定 Amazon S3 路径 DataBrew 选择文件。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

FilesLimit

如果提供，此结构将限制应该选择的文件数量。

类型：FilesLimit 对象

必需：否

LastModifiedDateCondition

如果提供，则此结构将根据 Amazon S3 对象在 Amazon S3 中的 LastModifiedDate 属性定义匹配这些对象的日期范围。

类型：FilterExpression 对象

必需：否

Parameters

将数据集的 Amazon S3 路径中使用的参数名称映射到其定义的结构。

类型：字符串到 DatasetParameter 对象的映射

映射条目：最多 10 项。

密钥长度限制：最小长度为 1。最大长度为 255。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

ProfileConfiguration

配置文件作业的配置。配置可用于选择列、进行评估并覆盖评估的默认参数。未定义配置时，配置文件作业将对所有支持的列应用默认设置。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

ColumnStatisticsConfigurations

色谱柱评估的配置列表。 ColumnStatisticsConfigurations 用于为特定列选择计算值和覆盖评估参数。未定义 ColumnStatisticsConfigurations 时，分析作业将分析所有支持的列并运行所有支持的评估。

类型：ColumnStatisticsConfiguration 对象数组

数组成员：最少 1 个物品。

必需：否

DatasetStatisticsConfiguration

列间评估的配置。配置可用于选择评估并覆盖评估的参数。未定义配置时，配置文件作业将运行所有受支持的列间评估。

类型：StatisticsConfiguration 对象

必需：否

EntityDetectorConfiguration

配置文件作业的实体检测配置。未定义时，禁用实体检测。

类型：EntityDetectorConfiguration 对象

必需：否

ProfileColumns

列选择器列表。 ProfileColumns 可用于从数据集中选择列。未定义 ProfileColumns 时，分析作业将对所有支持的列进行性能分析。

类型：ColumnSelector 对象数组

数组成员：最少 1 个物品。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Project

表示 DataBrew 项目的所有属性。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Name

项目的唯一名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

RecipeName

将在项目会话期间开发的食谱的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

AccountId

拥有该项目的 AWS 账户的 ID。

类型：字符串

长度限制：长度上限为 255。

必需：否

CreateDate

项目的创建日期和时间。

类型：时间戳

必需：否

CreatedBy

创建该项目的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

DatasetName

项目要执行操作的数据集。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

LastModifiedBy

上次修改项目的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

LastModifiedDate

项目的上次修改日期和时间。

类型：时间戳

必需：否

OpenDate

项目打开的日期和时间。

类型：时间戳

必需：否

OpenedBy

打开项目以供使用的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

ResourceArn

该项目的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

RoleArn

此项目应担任的角色的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

Sample

应用于数据的样本大小和采样类型。如果未指定此参数，则样本将包含数据集中的前 500 行。

类型：Sample 对象

必需：否

Tags

已应用于项目的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Recipe

表示要对 DataBrew 数据集执行的一项或多项操作。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Name

食谱的唯一名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

CreateDate

创建食谱的日期和时间。

类型：时间戳

必需：否

CreatedBy

创建配方的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

Description

对食谱的描述。

类型：字符串

长度限制：最大长度为 1024。

必需：否

LastModifiedBy

上次修改配方的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

LastModifiedDate

食谱的最后修改日期和时间。

类型：时间戳

必需：否

ProjectName

与配方关联的项目的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：否

PublishedBy

发布食谱的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

PublishedDate

食谱发布的日期和时间。

类型：时间戳

必需：否

RecipeVersion

食谱版本的标识符。必须是以下类型之一：

- 数字版本 (X.Y)-X Y 代表主版本号和次要版本号。每个数字的最大长度为 6 位数，且两者都不能为负值。X和Y都是必需的，而 “0.0” 不是有效的版本。

- LATEST_WORKING- DataBrew 项目中正在开发的最新有效版本。
- LATEST_PUBLISHED-最新发布的版本。

类型：字符串

长度限制：长度下限为 1。最大长度为 16。

必需：否

ResourceArn

配方的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

Steps

由食谱定义的步骤列表。

类型：RecipeStep 对象数组

必需：否

Tags

已应用于食谱的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

RecipeAction

表示用于对 DataBrew 数据集应用更改的转换和关联参数。有关更多信息，请参阅配方操作参考。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Operation

要对数据执行的有效 DataBrew 转换的名称。

类型：字符串

长度限制：长度下限为 1。长度上限为 128。

模式：`^[A-Z\_]+$`

必需：是

Parameters

转换的上下文参数。

类型：字符串到字符串映射

密钥长度限制：最小长度为 1。长度上限为 128。

键模式：`^[A-Za-z0-9]+$`

值长度限制：最小长度为 1。最大长度为 32768。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK

- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

RecipeReference

表示 DataBrew 食谱的名称和版本。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Name

食谱的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

RecipeVersion

食谱版本的标识符。

类型：字符串

长度限制：长度下限为 1。最大长度为 16。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

RecipeStep

表示要执行的 DataBrew 配方中的一个步骤。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Action

在食谱步骤中要执行的特定操作。

类型：RecipeAction 对象

必需：是

ConditionExpressions

为了使食谱步骤取得成功，必须满足一个或多个条件。

> **Note**
>
> 必须满足数组中的所有条件。换句话说，必须使用逻辑 AND 操作来组合所有条件。

类型：ConditionExpression 对象数组

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

RecipeVersionErrorDetail

表示尝试删除多个配方版本时遇到的任何错误。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

ErrorCode

错误的 HTTP 状态码。

类型：字符串

模式：`^[1-5][0-9][0-9]$`

必需：否

ErrorMessage

错误消息的文本。

类型：字符串

必需：否

RecipeVersion

与此错误相关的配方版本的标识符。

类型：字符串

长度限制：长度下限为 1。最大长度为 16。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Rule

表示应在此数据集范围内验证的单个数据质量要求。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

CheckExpression

包括列引用、条件名称并后跟变量引用的表达式，可能与其他条件一同分组并组合使用。例如，`(:col1 starts_with :prefix1 or :col1 starts_with :prefix2) and (:col1 ends_with :suffix1 or :col1 ends_with :suffix2)`。列和值引用是应以“:”符号开头的替代变量。根据上下文，替代变量的值可以是实际值或列名。这些值在中定义 SubstitutionMap。如果 a 以列引用 CheckExpression 开头，则 ColumnSelectors 在规则中应为空。如果 ColumnSelectors已定义，则条件的左侧不应有列引用，例如`is_between :val1 and :val2`。

有关更多信息，请参阅可用支票

类型：字符串

长度限制：最小长度为 4。长度上限为 1024。

模式：`^[<>0-9A-Za-z_.,:)(!= ]+$`

必需：是

Name

规则的名称。

类型：字符串

长度限制：长度下限为 1。长度上限为 128。

必需：是

ColumnSelectors

列选择器列表。选择器可用于使用数据集中的名称或正则表达式选择列。规则将应用于选定的列。

类型：ColumnSelector 对象数组

数组成员：最少 1 个物品。

必需：否

Disabled

指定是否禁用规则的值。禁用规则后，配置文件作业将不会在作业运行期间对其进行验证。默认值为 false。

类型：布尔值

必需：否

SubstitutionMap

替换变量名称与此检验表达式中所用值的映射。变量名称应以“:”（冒号）开头。变量值可以是实际值或列名称。为了区分两者，列名称应该用反引号括起来，例如，`":col1": "`Column A`"`.

类型：字符串到字符串映射

密钥长度限制：最小长度为 2。长度上限为 128。

键模式：`^:[A-Za-z0-9_]+$`

值长度约束：最大长度为 1024。

必需：否

Threshold

与非聚合校验表达式一起使用的阈值。非聚合校验表达式将应用于特定列中的每一行，阈值将用于确定验证是否成功。

类型：Threshold 对象

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK

- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

RulesetItem

包含关于规则集的元数据。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Name

规则集的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

TargetArn

与规则集关联的资源（数据集）的 Amazon 资源名称（ARN）。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：是

AccountId

拥有规则集的 AWS 账户的 ID。

类型：字符串

长度限制：长度上限为 255。

必需：否

CreateDate

规则集的创建日期和时间。

类型：时间戳

必需：否

CreatedBy

创建规则集的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

Description

规则集的描述。

类型：字符串

长度限制：最大长度为 1024。

必需：否

LastModifiedBy

上次修改规则集的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

LastModifiedDate

规则集的修改日期和时间。

类型：时间戳

必需：否

ResourceArn

规则集的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

RuleCount

在规则集中定义的规则数量。

类型：整数

有效范围：最小值为 0。

必需：否

Tags

已应用于规则集的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

S3Location

表示 Amazon S3 位置（存储桶名称、存储桶拥有者和对象密钥），DataBrew 可以在其中读取输入数据或写入任务的输出。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Bucket

Amazon S3 桶名称。

类型：字符串

长度约束：最小长度为 3。最大长度为 63。

必需：是

BucketOwner

存储桶拥有者的 AWS 账户 ID。

类型：字符串

长度限制：固定长度为 12。

模式：`^[0-9]{12}$`

必需：否

Key

存储桶中对象的唯一名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 1280。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

S3TableOutputOptions

表示选项，用于指定如何以及在何处 DataBrew 写入由配方任务生成的 Amazon S3 输出。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Location

表示 Amazon S3 位置（存储桶名称和对象密钥），DataBrew 可以在其中写入任务的输出。

类型：S3Location 对象

必需：是

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Sample

表示用于交互式数据分析 DataBrew 的样本数量和采样类型。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Type

从数据集中 DataBrew 获取行的方式。

类型：字符串

有效值：`FIRST_N | LAST_N | RANDOM`

必需：是

Size

示例中的行数。

类型：整数

有效范围：最小值为 1。最大值为 5000。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

Schedule

表示作业运行的一个或多个日期和时间。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Name

计划的名称。

类型：字符串

长度限制：长度下限为 1。最大长度为 255。

必需：是

AccountId

拥有该计划的 AWS 账户的 ID。

类型：字符串

长度限制：长度上限为 255。

必需：否

CreateDate

计划创建的日期和时间。

类型：时间戳

必需：否

CreatedBy

创建日程表的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

CronExpression

作业运行的日期和时间。有关更多信息，请参阅《 AWS Glue DataBrew 开发人员指南》中的使用配方作业的 cron 表达式。

类型：字符串

长度限制：长度下限为 1。最大长度为 512。

必需：否

JobNames

按时间表运行的作业列表。

类型：字符串数组

数组成员：最多 50 项。

长度限制：长度下限为 1。最大长度为 240。

必需：否

LastModifiedBy

上次修改计划的用户的亚马逊资源名称 (ARN)。

类型：字符串

必需：否

LastModifiedDate

上次修改时间表的日期和时间。

类型：时间戳

必需：否

ResourceArn

计划的亚马逊资源名称 (ARN)。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：否

Tags

已应用于时间表的元数据标签。

类型：字符串到字符串映射

地图条目：最大数量为 200 个项目。

密钥长度限制：最小长度为 1。长度上限为 128。

值长度限制：最大长度为 256。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

StatisticOverride

覆盖配置文件作业的特定评估。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Parameters

包含评估参数覆盖的映射。

类型：字符串到字符串映射

密钥长度限制：最小长度为 1。长度上限为 128。

键模式：`^[A-Za-z0-9]+$`

值长度限制：最小长度为 1。最大长度为 32768。

必需：是

Statistic

评估的名称

类型：字符串

长度限制：长度下限为 1。长度上限为 128。

模式：`^[A-Z\_]+$`

必需：是

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK

- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

StatisticsConfiguration

配置文件作业的评估配置。此配置可用于选择评估并覆盖选定评估的参数。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

IncludedStatistics

包含的评估列表。未定义列表时，将包括所有受支持的评估。

类型：字符串数组

数组成员：最少 1 个物品。

长度限制：长度下限为 1。长度上限为 128。

模式：`^[A-Z\_]+$`

必需：否

Overrides

评估的覆盖列表。

类型：StatisticOverride 对象数组

数组成员：最少 1 个物品。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2

- AWS 适用于 Ruby V3 的 SDK

Threshold

与非聚合校验表达式一起使用的阈值。非聚合校验表达式将应用于特定列中的每一行。然后，该阈值将用于确定验证是否成功。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

Value

阈值的值。

类型：双精度

有效范围：最小值为 0。

必需：是

Type

阈值的类型。用于将满足规则的实际行数与阈值进行比较。

类型：字符串

有效值：`GREATER_THAN_OR_EQUAL | LESS_THAN_OR_EQUAL | GREATER_THAN | LESS_THAN`

必需：否

Unit

阈值单位。可以是用于验证的完整样本量的 COUNT（计数）或 PERCENTAGE（百分比）。

类型：字符串

有效值：`COUNT | PERCENTAGE`

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

ValidationConfiguration

数据质量验证的配置。用于选择要在配置文件作业中使用的规则集和验证模式。如果 ValidationConfiguration 为 null，则配置文件作业将在不进行数据质量验证的情况下运行。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

RulesetArn

要在配置文件作业中验证的规则集的 Amazon 资源名称（ARN）。所选规则集 TargetArn 的应与与分析任务关联的数据集的 Amazon 资源名称 (ARN) 相同。

类型：字符串

长度约束：最小长度为 20。最大长度为 2048。

必需：是

ValidationMode

数据质量验证模式。默认模式为“CHECK_ALL”，用于验证选定规则集中定义的所有规则。

类型：字符串

有效值：CHECK_ALL

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

ViewFrame

表示操作期间正在转换的数据。

内容

> **Note**
>
> 下表中，首先描述的是必需参数。

StartColumnIndex

视图框架中要返回的列范围的起始索引。

类型：整数

有效范围：最小值为 0。

必需：是

Analytics

控制是启用还是禁用分析计算。默认情况下启用。

类型：字符串

有效值：`ENABLE | DISABLE`

必需：否

ColumnRange

要包含在视图框架中的列数，从`StartColumnIndex`值开始，忽略`HiddenColumns`列表中的任何列。

类型：整数

有效范围：最小值为 0。最大值为 20。

必需：否

HiddenColumns

要在视图框架中隐藏的列列表。

类型：字符串数组

长度约束：最小长度为 1。最大长度为 255。

必需：否

RowRange

视图框架中要包含的行数，从StartRowIndex值开始。

类型：整数

必需：否

StartRowIndex

视图框架中要返回的行范围的起始索引。

类型：整数

有效范围：最小值为 0。

必需：否

另请参阅

有关在特定语言的 AWS SDK 中使用此 API 的更多信息，请参阅以下内容：

- AWS 适用于 C++ 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 Ruby V3 的 SDK

常见错误

本部分列出了所有 AWS 服务的常见 API 操作错误。对于特定于此服务的 API 操作的错误，请参阅该 API 操作的主题。

AccessDeniedException

您没有足够的访问权限，无法执行该操作。

HTTP 状态代码：400

IncompleteSignature

请求签名不符合 AWS 标准

HTTP 状态代码：400

InternalFailure

由于未知错误、异常或故障，请求处理失败。

HTTP 状态代码：500

InvalidAction

所请求的操作无效。确认正确键入了操作。

HTTP 状态代码：400

InvalidClientTokenId

在我们的记录中没有所提供的 X.509 证书或 AWS 访问密钥 ID。

HTTP 状态代码：403

NotAuthorized

您无权执行此操作。

HTTP 状态代码：400

OptInRequired

AWS 访问密钥 ID 需要订阅服务。

HTTP 状态代码：403

RequestExpired

请求到达服务的时间超过请求上的日期戳 15 分钟或超过请求到期日期 15 分钟（例如，对于预签名 URL），或者请求上的日期戳比当前时间晚了 15 分钟以上。

HTTP 状态代码：400

ServiceUnavailable

由于服务器发生临时故障而导致请求失败。

HTTP 状态代码：503

ThrottlingException

由于请求限制而导致请求被拒绝。

HTTP 状态代码：400

ValidationError

输入未能满足 AWS 服务指定的约束。

HTTP 状态代码：400

常见参数

以下列表包含所有操作用于使用查询字符串对 Signature Version 4 请求进行签名的参数。任何特定于操作的参数都列在该操作的主题中。有关签名版本 4 的更多信息，请参阅 I AM 用户指南中的签署 AWS API 请求。

Action

要执行的操作。

类型：字符串。

必需：是

Version

编写请求所针对的 API 版本，格式为 YYYY-MM-DD。

类型：字符串。

必需：是

X-Amz-Algorithm

您用于创建请求签名的哈希算法。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

有效值：AWS4-HMAC-SHA256

必需：条件

X-Amz-Credential

凭证范围值，该值是一个字符串，其中包含您的访问密钥、日期、您要定位的区域、您请求的服务以及终止字符串（“aws4_request”）。值采用以下格式表示：access_key/YYYYMMDD/region/service/aws4_request。

有关更多信息，请参阅 I AM 用户指南中的创建已签名的 AWS API 请求。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

必需：条件

X-Amz-Date

用于创建签名的日期。格式必须为 ISO 8601 基本格式 (YYYYMMDD'T'HHMMSS'Z')。例如，以下日期时间是有效的 X-Amz-Date 值：`20120325T120000Z`。

条件：X-Amz-Date 对于所有请求都是可选的；它可以用于覆盖对请求签名所使用的日期。如果以 ISO 8601 基本格式指定 Date 标头，则不需要 X-Amz-Date。使用 X-Amz-Date 时，它始终会覆盖 Date 标头的值。有关更多信息，请参阅 I AM 用户指南中的 AWS API 请求签名元素。

类型：字符串

必需：条件

X-Amz-Security-Token

通过调用 AWS Security Token Service （ AWS STS ）获得的临时安全令牌。有关支持来自 AWS STS的临时安全凭证的服务列表，请参阅《IAM 用户指南》中的使用 IAM 的AWS 服务。

条件：如果您使用来自的临时安全凭证 AWS STS，则必须包含安全令牌。

类型：字符串

必需：条件

X-Amz-Signature

指定从要签名的字符串和派生的签名密钥计算的十六进制编码签名。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

必需：条件

X-Amz-SignedHeaders

指定作为规范请求的一部分包含的所有 HTTP 标头。有关指定签名标头的更多信息，请参阅 I AM 用户指南中的创建已签名的 AWS API 请求。

条件：当您在查询字符串中而不是 HTTP 授权标头中包括身份验证信息时，请指定此参数。

类型：字符串

必需：条件

的配额 AWS Glue DataBrew

您可以在 Service Quotas 控制台中查看您的 DataBrew AWS 服务配额。对于任何可调整的限额，您还可以请求提高限额。

《 AWS Glue DataBrew 开发人员指南》的文档历史记录

当前 API 版本：d atabrew-2017-07-25

下表介绍了此版本的文档 AWS Glue DataBrew。如果您希望在《AWS Glue DataBrew 开发人员指南》更新时收到通知，可以订阅 RSS 源。

变更	说明	日期
glue:GetCustomEntityType 已添加到 AWS 托管策略	在启用 PII 识别的情况下执行 AWS Glue DataBrew 配置文件作业需要此权限。有关更多信息，请参阅AWS 托管策略的 AWS Glue DataBrew 更新。	20年 20 月 20 日
在 CRYPTOGRAPHIC_HASH 转换中支持多种哈希算法	现在，您可以在对列中的值进行哈希处理时指定哈希算法。有关更多信息，请参阅 CRYPTOGRAPHIC_HA SH。	2023 年 8 月 11 日
glue:BatchGetCustomEntityTypes 已添加到 AWS 托管策略	在启用 PII 识别的情况下执行 AWS Glue DataBrew 配置文件作业需要此权限。有关更多信息，请参阅AWS 托管策略的 AWS Glue DataBrew 更新。	2022 年 5 月 9 日
支持 Apache ORC 文件格式	DataBrew 现在支持 Apache ORC 作为 DataBrew 数据源和输出的文件格式。有关更多信息，请参阅数据源支持的文件类型。	2022 年 3 月 31 日
支持跨账户访问 AWS Glue Data Catalog Amazon S3	现在， AWS 账户 如果在 AWS Glue 控制台中创建了适当的资源策略，则可以从其他人访问 AWS Glue Data Catalog S3 表。创建策略后，可以在创建数据DataBrew 集时选择相关的	2022 年 3 月 11 日

	数据目录 S3 表作为输入源。有关更多信息，请参阅支持的数据源和输出的连接。	
支持与 Amazon 的本地控制台集成 AppFlow	DataBrew 现在已经与 Amazon 集成了原生控制台 AppFlow。这种集成意味着你可以连接来自 Salesforce、Zendesk、Slack 和其他 (software-as-a-service SaaS) 应用程序的数据。 ServiceNow您还可以连接来自亚马逊 S3 和 Amazon Redshift AWS 服务 等的数据。有关更多信息，请参阅支持的数据源和输出的连接。	2021 年 11 月 18 日
Support 对数据质量规则的支持	DataBrew 现在支持创建数据质量规则，这些规则是可自定义的验证检查，用于定义特定数据的业务需求。有关更多信息，请参阅中的验证数据质量。 AWS Glue DataBrew	2021 年 11 月 18 日
Support 支持自定义 SQL 语句	DataBrew 现在支持用于从 Amazon Redshift 和 Snowflake 检索数据的自定义 SQL 语句。这种支持意味着您可以使用专门构建的查询来选择和限制从大型表返回的数据。有关更多信息，请参阅支持的数据源和输出的连接。	2021 年 11 月 18 日
Support 支持 PII 检测	DataBrew 现在支持检测个人身份信息 (PII)。这使您可以选择在数据准备期间屏蔽 PII。有关更多信息，请参阅识别和处理个人身份信息 (PII)。	2021 年 11 月 18 日

对其他 AWS 区域的 Support	DataBrew 现在支持其他 AWS 区域。有关支持的区域列表，请参阅AWS Glue DataBrew 终端节点和配额。	2021 年 10 月 5 日
支持将数据写入基于 Lake Formation 的 Amazon S3 表	DataBrew 现在支持基于将数据写入 AWS Glue Data Catalog S3 表 AWS Lake Formation。DataBrew 现在还支持将数据写入 Tableau Hyper 格式。有关更多信息，请参阅创建和使用 AWS Glue DataBrew 配方作业。	2021 年 8 月 13 日
Support 支持将数据写入 JDBC 目的地	DataBrew 现在支持将数据直接写入 JDBC 支持的数据库和数据仓库。其中包括Amazon Redshift、Snowflake、MySQL、Oracle Database 和 PostgreSQL。有关更多信息，请参阅创建和使用 AWS Glue DataBrew 配方作业。	2021 年 7 月 23 日
Support 支持指定为分析作业生成哪些数据质量统计数据	DataBrew 现在支持指定在分析作业中为数据集自动生成哪些数据质量统计数据。有关更多信息，请参阅创建和使用 AWS Glue DataBrew 配方作业。	2021 年 7 月 23 日

Support 支持将数据集写入 AWS Glue Data Catalog	DataBrew 现在支持将数据集直接写入 AWS Glue Data Catalog。您可以选择将根据运行数据准备配方的任务创建的数据集存储在数据目录中的 Amazon S3、Amazon Redshift 和 Amazon RDS 表中。支持的 RDS 表包括 Amazon Aurora、RDS for Oracle、RDS for Microsoft SQL Server、RDS for MySQL 和 RDS for PostgreSQL 的表。	2021 年 6 月 30 日
Support 支持识别高级数据类型	DataBrew 现在支持自动识别和标记列的高级数据类型，这样可以更轻松地对包含某些类型数据的列进行标准化。这些类型的数据包括社会安全号码、电子邮件地址、电话号码、性别、信用卡、URL、IP 地址、日期和时间、货币、邮政编码、国家、地区、州和城市。	2021 年 6 月 30 日
支持使用亚马逊 AppFlow 从 SAAS 应用程序传输数据	DataBrew 现在支持使用亚马逊 AppFlow 将数据从第三方 software-as-a-service (SaaS) 应用程序（例如 Salesforce、Zendesk、Slack 和 ServiceNow）传输到亚马逊 S3。有关更多信息，请参阅支持的数据源和输出的连接。	2021 年 4 月 29 日

Support 支持使用来自 JDBC 数据库的输入创建 DataBrew 数据集	DataBrew 现在支持从 JDBC 支持的数据库和数据仓库中创建数据集，包括 Amazon Redsoft、Snowflake、MySQL、Oracle Database 和 PostgreSQL。有关更多信息，请参阅支持的数据源和输出的连接。	2021 年 4 月 2 日
Support 支持其他 AWS 区域	DataBrew 现在支持其他 AWS 区域。有关支持的区域列表，请参阅AWS Glue DataBrew 终端节点和配额。	2021 年 1 月 28 日
用于处理重复的新变换	DataBrew 控制台和 API 中添加了四种用于处理重复的新转换。有关更多信息，请参阅数据质量配方步骤中的 DELETE_DUPLICATE_ROWS、FLAG_DUPLICATE_ROWS、FLAG_DUPLICATES_IN_COLUMN 和 REMOVE_DUPLICATES。	2021 年 1 月 28 日
其他 CSV 分隔符	DataBrew 现在，除了逗号分隔值 (CSV) 文件中用于创建数据集外，还支持其他分隔符。DataBrew 有关更多信息，请参阅创建和使用 AWS Glue DataBrew 数据集。	2021 年 1 月 28 日
DataBrew 的扩展 JupyterLab	现在，您可以在中 AWS Glue DataBrew 用作扩展 JupyterLab。有关更多信息，请参阅中的 DataBrew 用作扩展 JupyterLab。	2020 年 11 月 20 日

新的数据准备工具： AWS Glue DataBrew	这是 AWS Glue DataBrew 开发人员指南的首次发布。	2020 年 11 月 11 日

AWS 术语表

有关最新 AWS 术语，请参阅《AWS 词汇表 参考资料》中的AWS 词汇表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。