quantiphi

Data Lake Immersion Day Lab Manual

Date: 28 Feb, 2021 | Author: Quantiphi, Inc.

Contents

Introduction **4**

About Heart Disease Data Set **5**

Prerequisites **7**

Section A: Launching your CloudFormation stack **8**

Section B: AWS S3 **12**
Step 1: Create an s3 bucket for source and target files. 12
Step 2: Upload the source file to the source s3 bucket 12

Section C: AWS Glue DataBrew **13**
Part A: Prepare the source dataset 14
Step 1: Add Source Dataset in AWS Glue DataBrew 14
Part B: Create a project 15
Part C: Create profile job 21
Part D: Data Validation 24
Step 1: Change the column names of schema 25
Step 2: Fill the missing values with median using the box plot graph 25
Step 3: Publish / save the recipe for future use 26
Step 4: View the recipe 27
Step 5: Save the output location of job run 27

Section D: AWS Glue Studio **28**
Step 1: Start the job creation process 29
Step 2: Edit the data source node in the job graph 30
Step 3: Edit the transform node of the job 31
Step 4: Edit the data target node of the job 35
Step 5: View the final job script 36
Step 6: Specify the job details and save the job 36
Step 7: Run the job 37

Section E: AWS S3 Select **39**
Step 1: View the output in s3 bucket and use s3 select for querying 39
Step 2: Enter the select query and select the file format and output format 40

Section F: AWS Athena **41**

Step 1: Query using AWS Athena by adding the data source 41
Step 2: Select the option select table and schema manually 42
Step 3: Provide database name, table name and S3 target location 42
Step 4: Select file format as CSV 43
Step 5: Add the column and data type using bulk column feature 43
Step 6: Create Partitions if necessary 44
Step 7: Save it and the table ddl will updated in the query editor and table will be created 45
Step 8: Query the table using AWS Athena 46
Step 9: Save results of query into s3 46

Introduction

Heart disease describes a range of conditions that affect your heart. Diseases under the heart disease umbrella include blood vessel diseases, such as coronary artery disease, heart rhythm problems and heart defects you're born with, among others.

With the advent of machine learning and artificial learning, it can help with classifying whether a person is suffering from heart disease or not, using one of the most used Cleveland Heart Disease dataset from the UCI Repository.

In this article, we will be walking through various stages of the process of staging and preparing data for applying Machine Learning. We will be utilizing various AWS services like AWS Cloud formation, AWS S3, AWS Glue, AWS Glue Data Brew, AWS Glue Studio ,AWS S3 Query and AWS Athena for our use case.

This article will explain the various aws services and how to use them at basic level. This is intended for the intermediate level cloud practitioners and beginners who have basic knowledge to handle data transformations.

About Heart Disease Data Set

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date.

The "num" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 3. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3) from absence (value 0).

https://archive.ics.uci.edu/ml/datasets/Heart+Disease

Attribute Information

Only 14 attributes used:

1. #3 (age): age in years

2. #4 (sex): sex (1 = male; 0 = female)

3. #9 (cp): chest pain type

    -- Value 1: typical angina

    -- Value 2: atypical angina

    -- Value 3: non-anginal pain

    -- Value 4: asymptomatic

4. #10 (trestbps): resting blood pressure (in mm Hg on admission to the hospital)

5. #12 (chol): serum cholesterol in mg/dl

6. #16 (fbs): (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

7. #19 (restecg): resting electrocardiographic results

-- Value 0: normal

-- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

8. #32 (thalach): maximum heart rate achieved
9. #38 (exang): exercise induced angina (1 = yes; 0 = no)
10. #40 (oldpeak): ST depression induced by exercise relative to rest
11. #41 (slope): the slope of the peak exercise ST segment

-- Value 1: upsloping

-- Value 2: flat

-- Value 3: downsloping

12. #44 (ca): number of major vessels (0-3) colored by fluoroscopy
13. #51 (thal): 3 = normal; 6 = fixed defect; 7 = reversible defect
14. #58 (num) (the predicted attribute 0, 1, 2, 3)

Prerequisites

This tutorial has the following prerequisites:

- You should have an AWS account.
- You should have an AWS Identity and Access Management (IAM) user.
- You should be able to run the AWS Cloud formation template needed to create the necessary IAM role for the tutorial.
- You have to create everything in us-east-1-virginia

Section A: Launching your CloudFormation stack

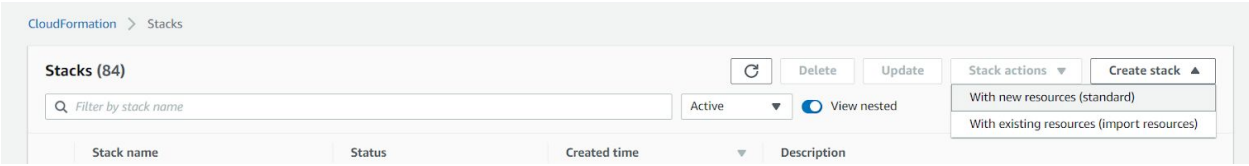
Your account has all the necessary permissions for creating and running a job for an Amazon S3 data source and data target. You have created an AWS Identity and Access Management role for the job to use. You can also choose an IAM role for the job that includes permissions for all your data sources, data targets, temporary directory, scripts, and any libraries used by the job.

Here the cloud formation template creates two roles:

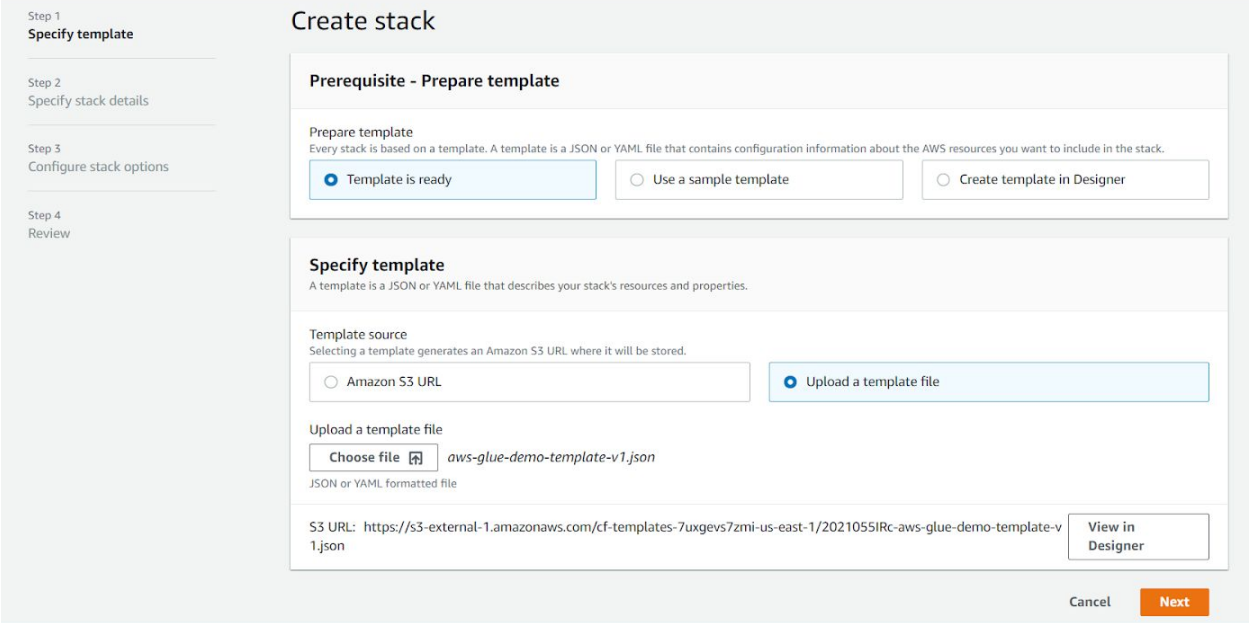
- aws-glue-lab-databrew-role
- aws-glue-lab-studio-role

To create your resources for this use case, complete the following steps:

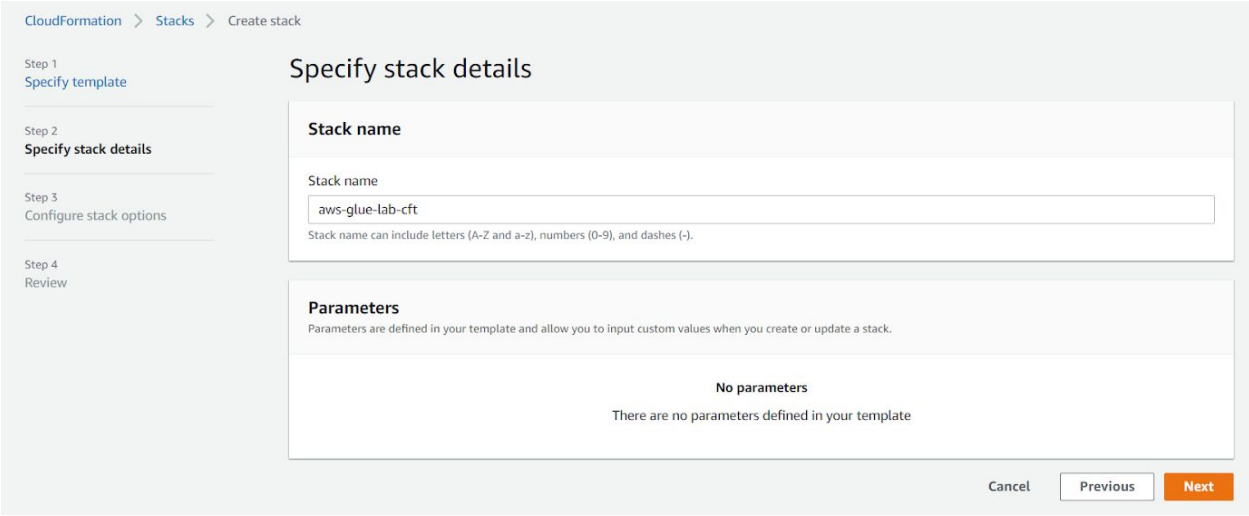
- Select Create stack with new resources option



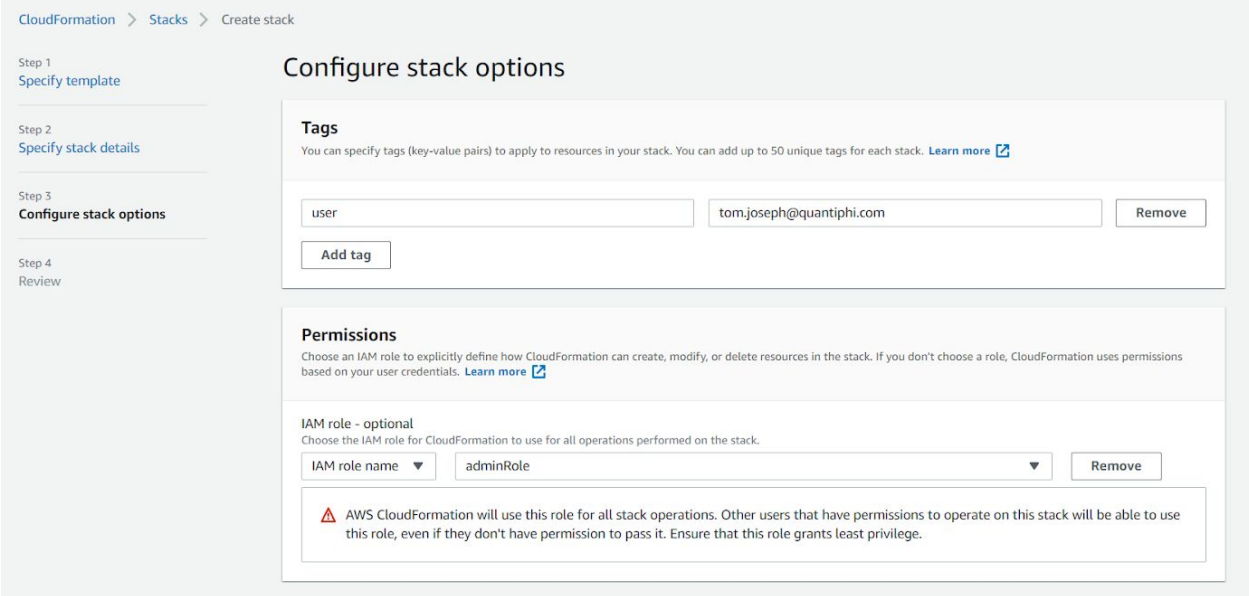
- Use the template provided in JSON format



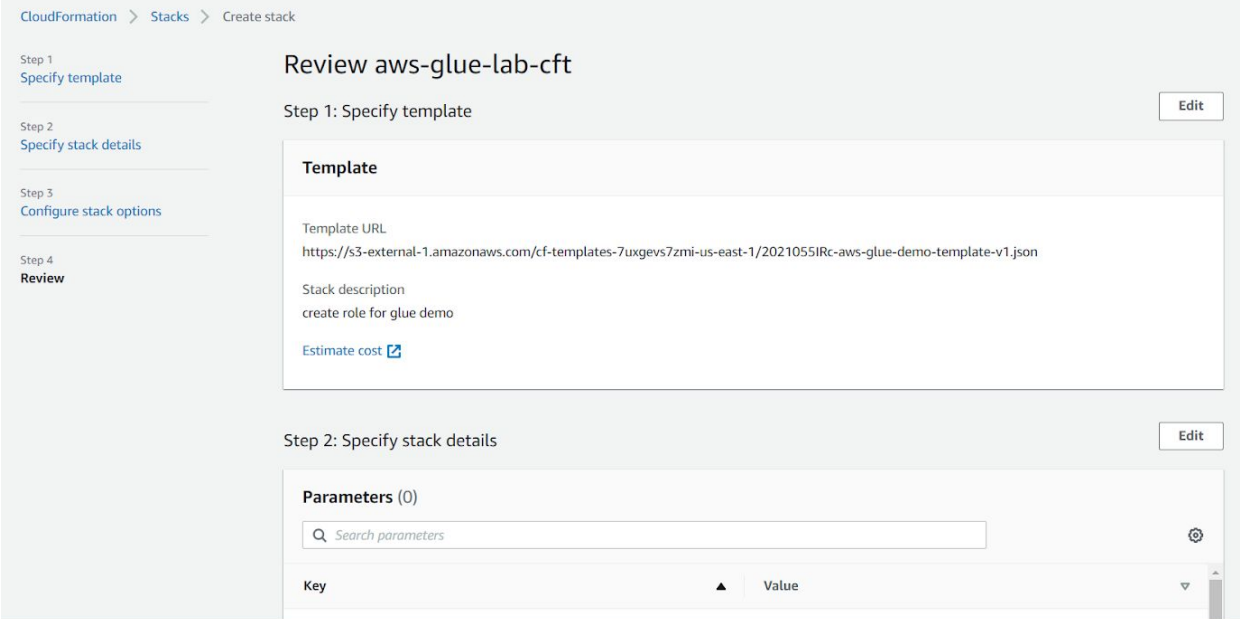
- Provide the stack name aws-glue-lab-cft

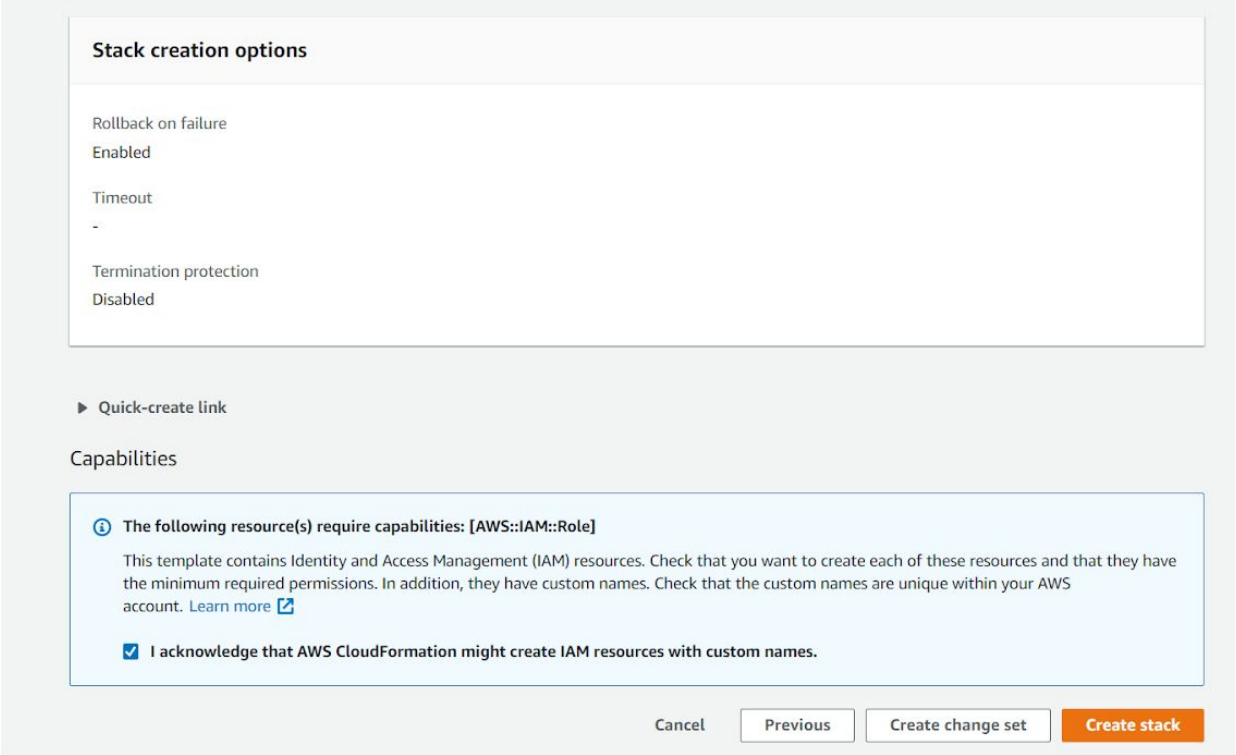


- Provide the tags if needed and also select the IAM role if required.



- Review and launch this stack creates AWS resources





Section B: AWS S3

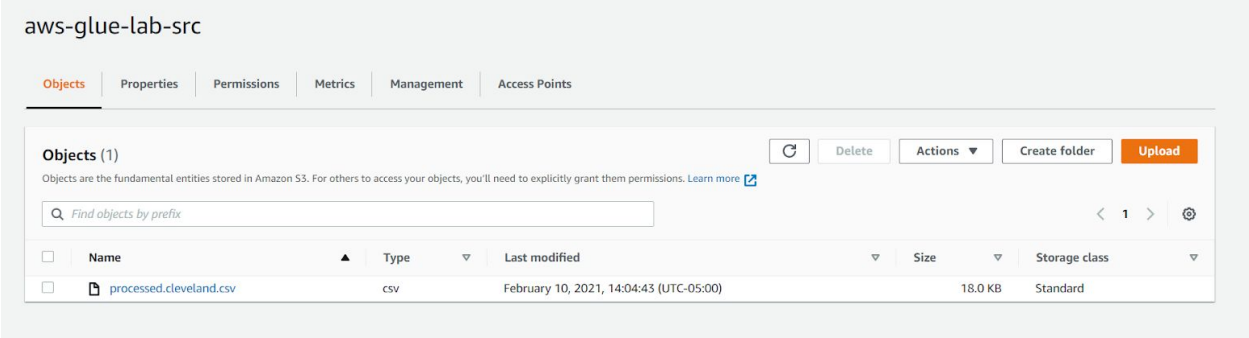
Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides easy-to-use management features so you can organize your data and configure finely-tuned access controls to meet your specific business, organizational, and compliance requirements.

Step 1: Create an s3 bucket for source and target files.

Note: Please provide globally unique name

○	aws-glue-lab-demo-landing	US East (N. Virginia) us-east-1	Bucket and objects not public
○	aws-glue-lab-demo-query-output	US East (N. Virginia) us-east-1	Bucket and objects not public
○	aws-glue-lab-demo-src	US East (N. Virginia) us-east-1	Bucket and objects not public
○	aws-glue-lab-demo-tgt	US East (N. Virginia) us-east-1	Bucket and objects not public

Step 2: Upload the source file to the source s3 bucket



Section C: AWS Glue DataBrew

AWS Glue DataBrew provides a visual interface that quickly connects to your data stored in Amazon Simple Storage Service (S3), Amazon Redshift, Amazon Relational Database Service (RDS), any JDBC accessible data store, or data indexed by the AWS Glue Data Catalog. You can then explore the data, look for patterns, and apply transformations. For example, you can apply joins and pivots, merge different data sets, or use functions to manipulate data.

Once your data is ready, you can immediately use it with AWS and third-party services to gain further insights, such as Amazon SageMaker for machine learning, Amazon Redshift and Amazon Athena for analytics, and Amazon QuickSight and Tableau for business intelligence.

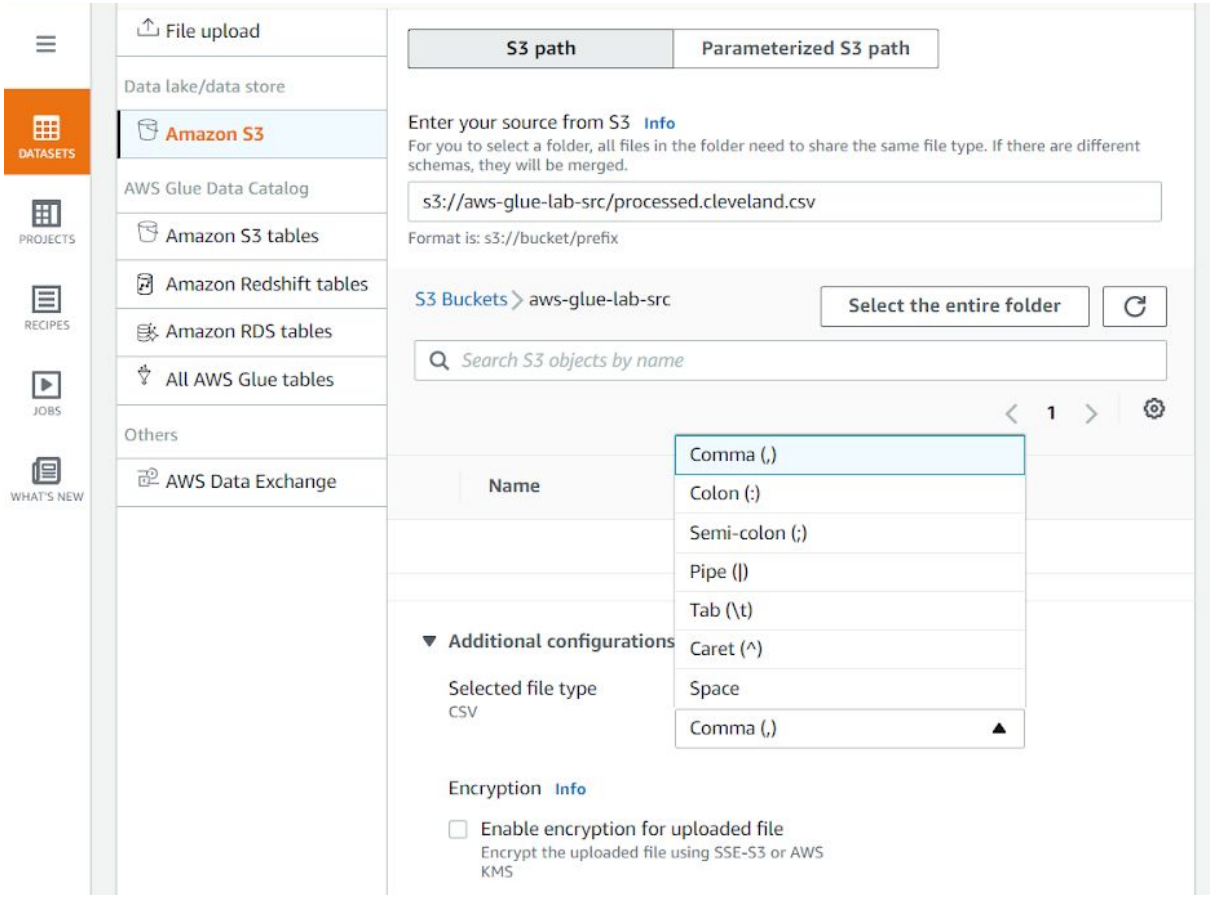
To prepare your data with DataBrew, you follow these steps:

- Connect one or more datasets from S3 or the Glue data catalog (S3, Redshift, RDS). You can also upload a local file to S3 from the DataBrew console. CSV, JSON, Parquet, and .XLSX formats are supported.
- Create a project to visually explore, understand, combine, clean, and normalize data in a dataset. You can merge or join multiple datasets. From the console, you can quickly spot anomalies in your data with value distributions, histograms, box plots, and other visualizations.
- Generate a rich data profile for your dataset with over 40 statistics by running a job in the profile view.
- When you select a column, you get recommendations on how to improve data quality.
- You can clean and normalize data using more than 250 built-in transformations. For example, you can remove or replace null values, or create encodings. Each transformation is automatically added as a step to build a recipe.
- You can then save, publish, and version recipes, and automate the data preparation tasks by applying recipes on all incoming data. To apply recipes to or generate profiles for large datasets, you can run jobs.
- At any point in time, you can visually track and explore how datasets are linked to projects, recipes, and job runs. In this way, you can

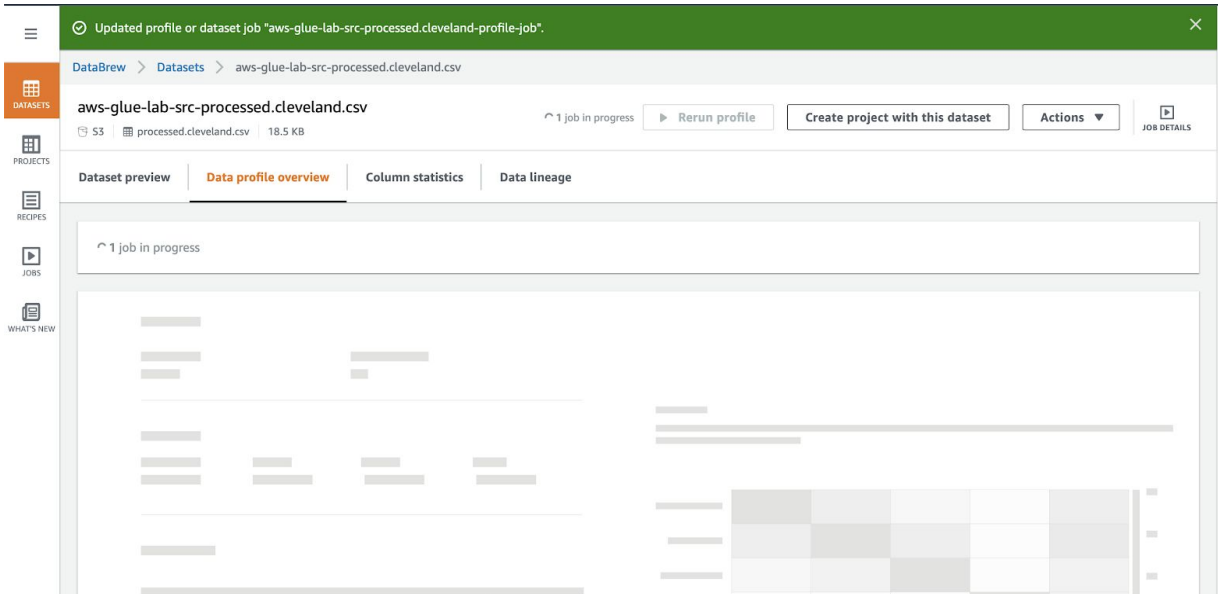
understand how data flows and what are the changes. This information is called data lineage and can help you find the root cause in case of errors in your output.

Part A: Prepare the source dataset

Step 1: Add Source Dataset in AWS Glue DataBrew

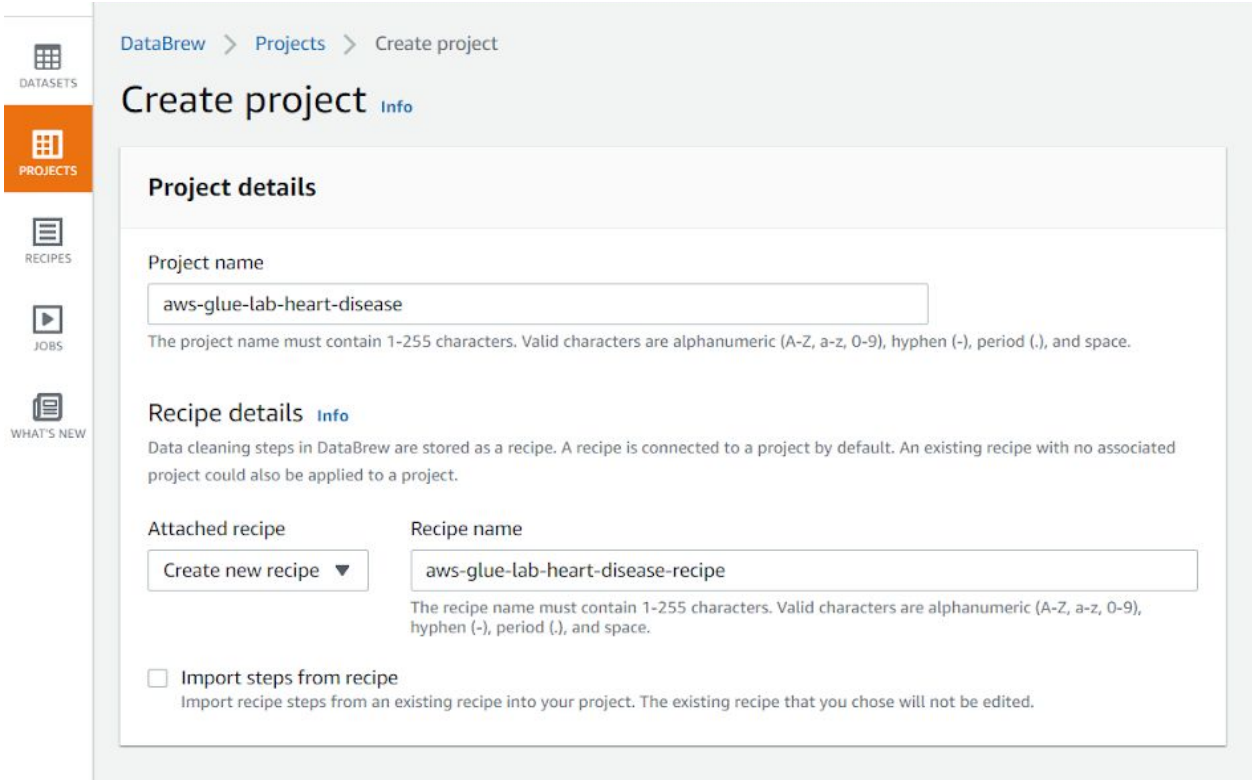


Dataset Screen getting updated



Part B: Create a project

1. Sign in to the AWS Management Console and open the DataBrew console .
2. On the navigation pane, choose **PROJECTS**. Then choose **Create project**.



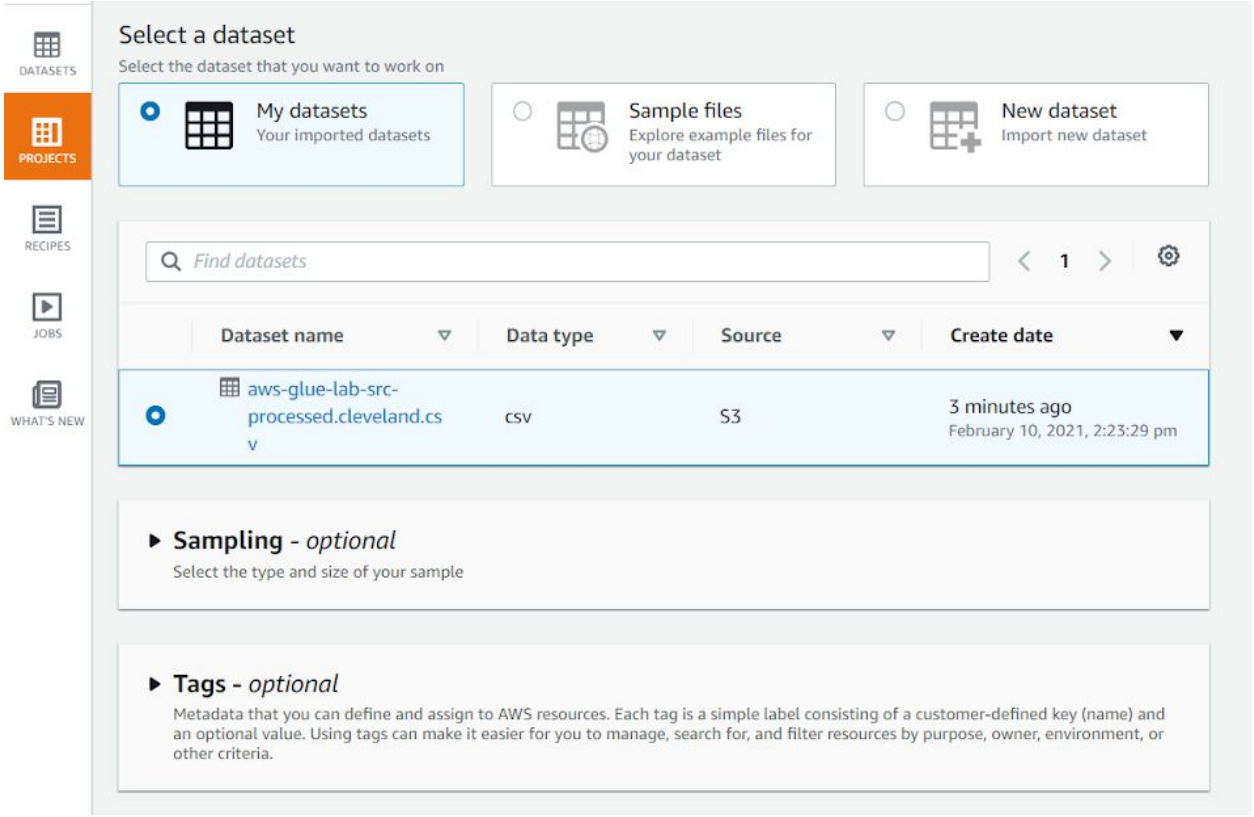
3. Enter a name for your project. Then choose a recipe to attach to your project:
    - Choose **Create a new recipe** if you are starting from the beginning. Doing this creates a new, empty recipe and attaches it to your project.
    - Choose **Edit existing recipe** if you have a previously published recipe that you want to use for this project. If the recipe is currently attached to another project, or has any jobs defined for it, then you can't use it in your new project. Choose **Browse recipes** to see what recipes are available.
    - Choose **Import steps from recipe** if you have an existing recipe that's been published previously and want to import its steps, and then do the following:
        a. Choose **Browse recipes** to see what recipes are available.

b. Choose the published version of the recipe that you want to use. A recipe can have multiple versions, depending on how often you published it while working in a project view.
c. Choose **View recipe steps** to examine the data transformations in the recipe.

4. Select the dataset

After you have a recipe, choose the dataset that you want to work with on the **Select a dataset** pane:

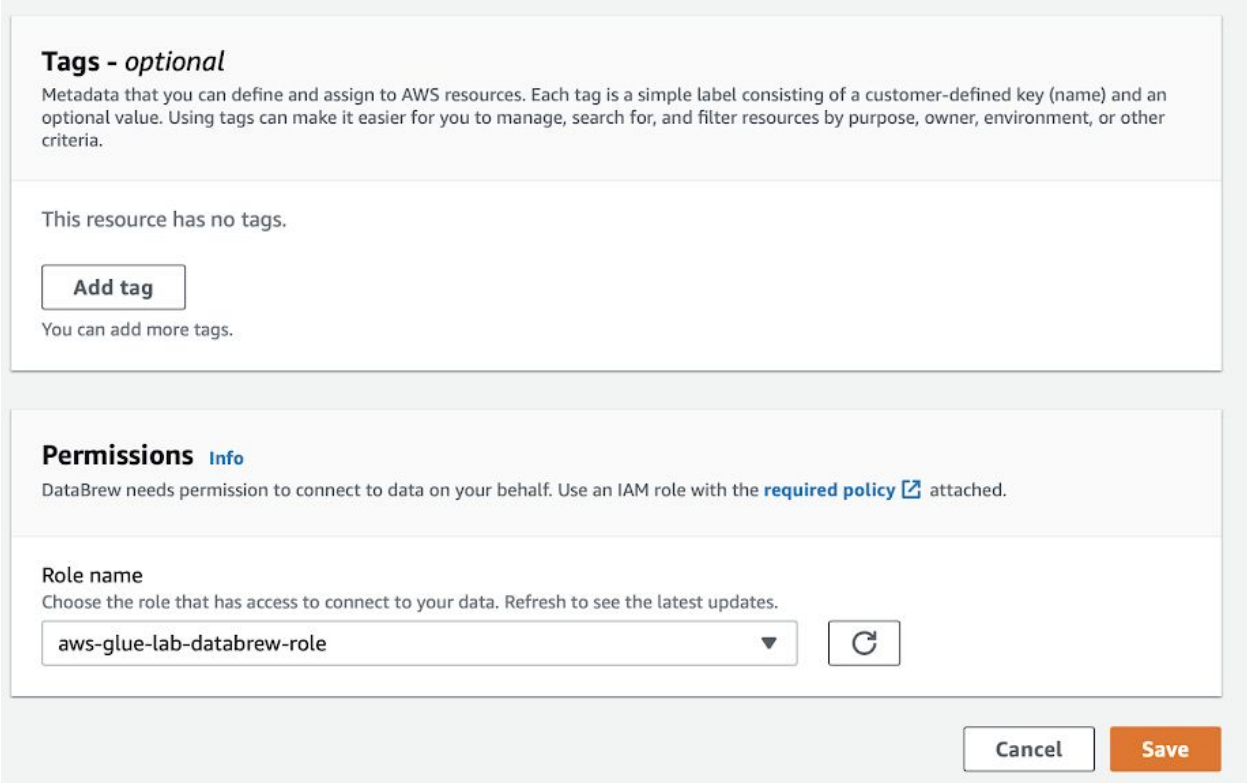
- **My datasets** – Choose a dataset that you created previously. For more information, see Creating a project.)
- **Sample files** – Create a new dataset based on sample data maintained by AWS. This sample data is a great way to explore what DataBrew can do, without having to provide your own data. Make sure to enter a name for your dataset.
- **New dataset** – Create a new dataset. For more information, see Creating a project.



5. Select The AWS IAM Role

For **Access permissions**, choose an AWS Identity and Access Management (IAM) role that allows DataBrew to read from your Amazon S3 input location. For an S3 location owned by your AWS account, you can choose the AwsGlueDataBrewDataAccessRole service-managed role. Doing this allows DataBrew to access S3 resources that you own.

Here I am using the AWS-Practice developer role which has similar policies created using the stack and incorporates the AwsGlueDataBrewDataAccessRole service-managed role.



6. Select sampling options

   On the **Sampling** pane, you can find options for DataBrew to build a sample of data from your dataset.

   For **Type**, choose how DataBrew should get rows from your dataset:

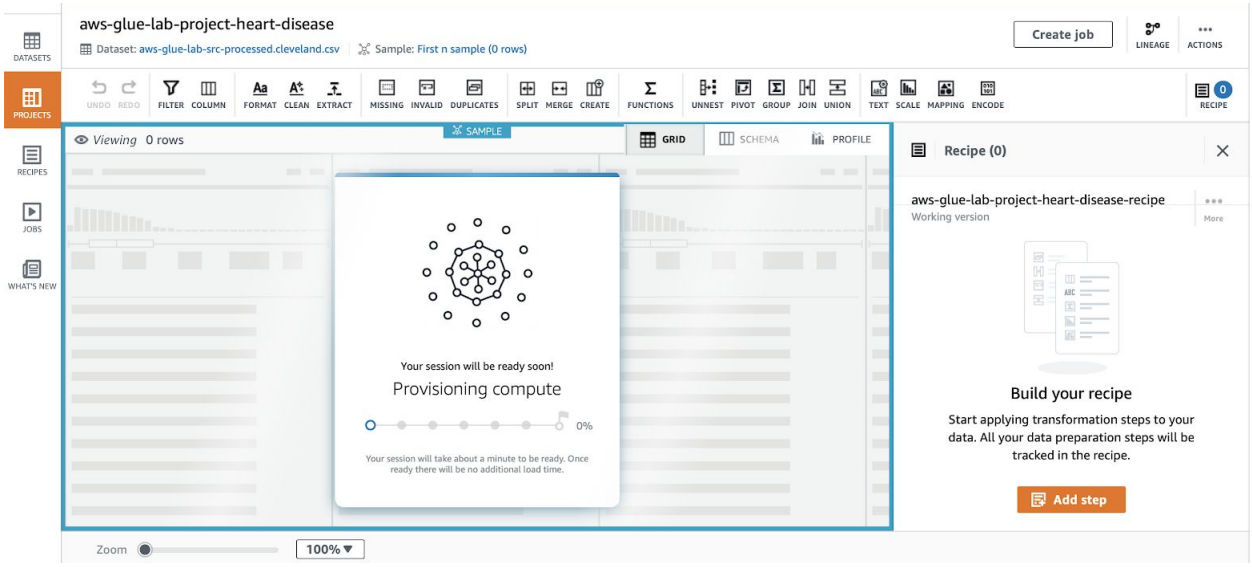
   - Use **First n rows** to create a sample based on the first rows in the dataset.
   - Use **Random rows** to create a sample based on a random selection of rows in the dataset.
   - Choose the number of rows to appear in the sample: 500, 1,000, 2,500, or a custom sample size, up to a maximum of 5,000 rows. A smaller sample size allows DataBrew to perform transformations faster, saving you time as you develop your recipe. A larger sample size more accurately reflects the makeup of the

underlying source data. However, project session initialization and interactive transformations are slower.

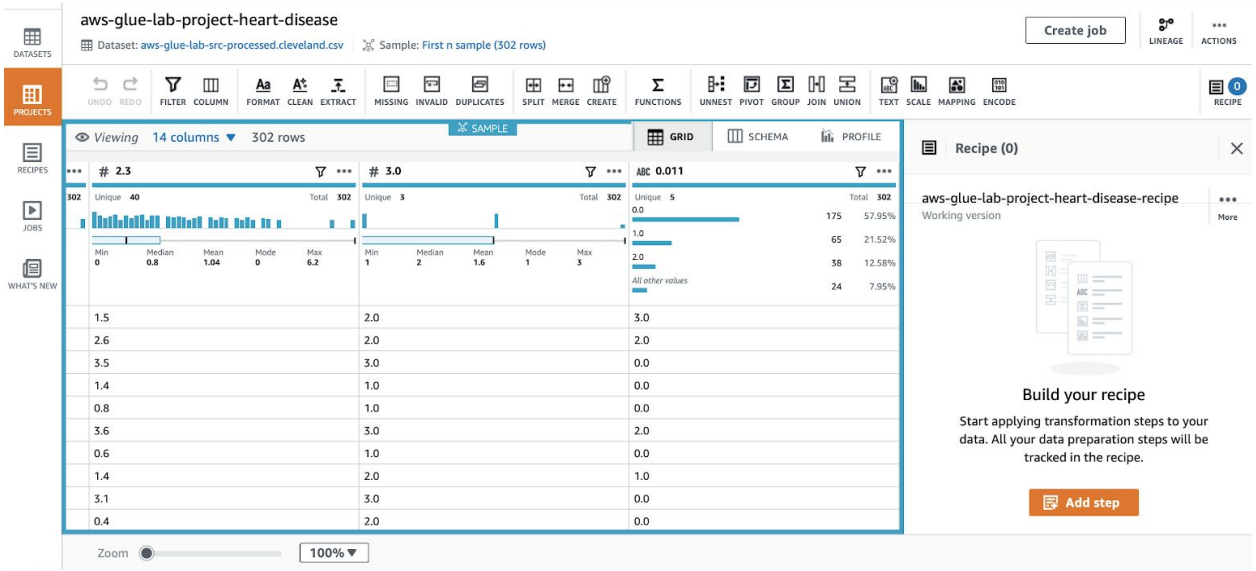
7. (Optional) Choose **Tags** to attach tags to your dataset.

   Tags are simple labels consisting of a user-defined key and an optional value that can make it easier to manage, search for, and filter DataBrew projects by purpose, owner, environment, or other criteria.

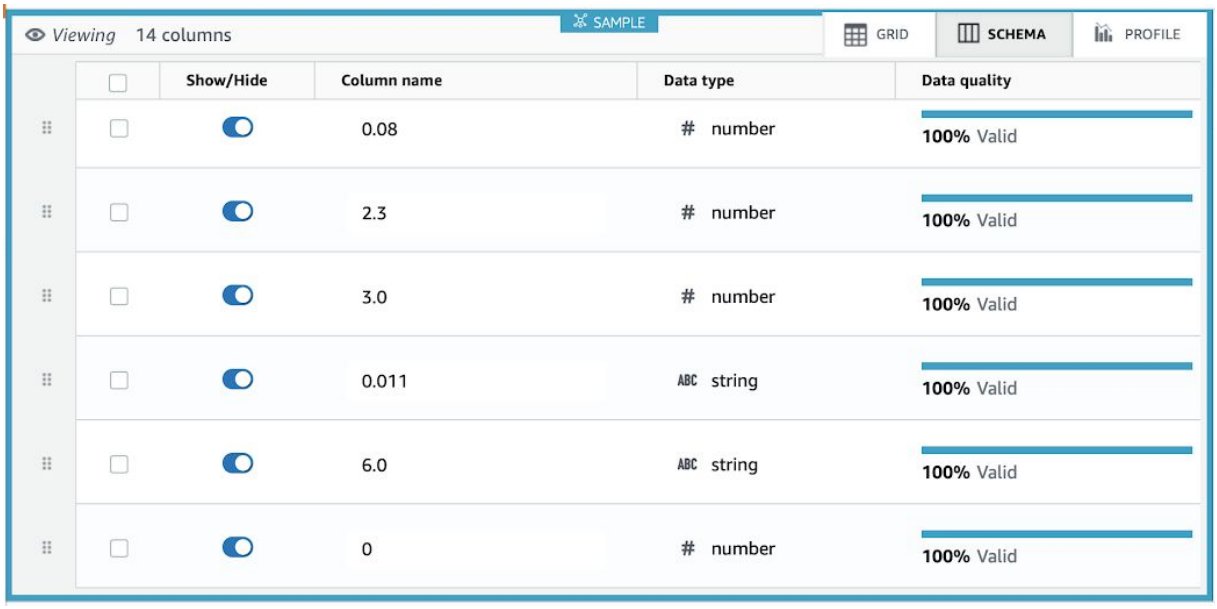
8. When the settings are as you want them, choose **Create job**.



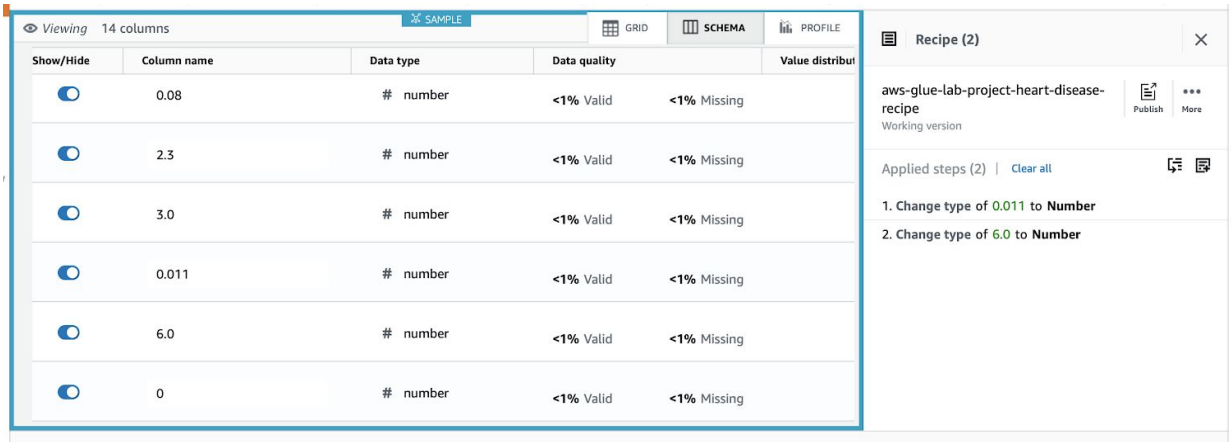
Project is populating the data of the source dataset



Schema is being populated



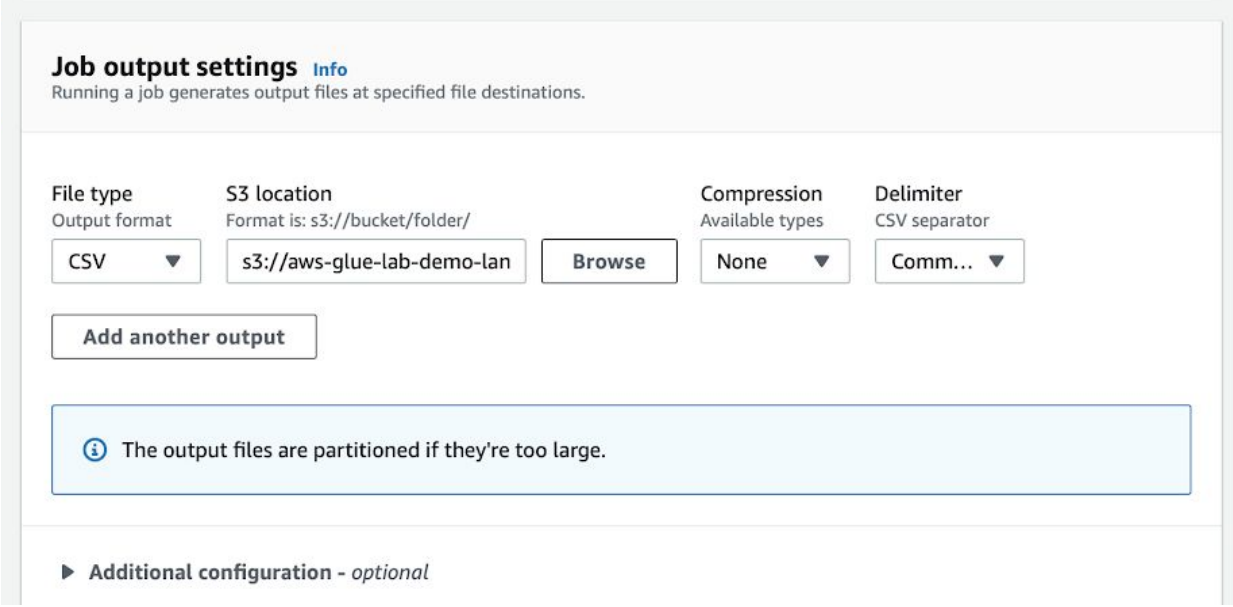
Changing the necessary column data types



Part C: Create profile job

1. Sign in to the AWS Management Console and open the DataBrew console .
2. Choose JOBS from the navigation pane, choose either the Recipe jobs tab or the Profile jobs tab, and then choose Create job.
3. Enter a name for your job, and then choose either Create a recipe job or Create a profile job.

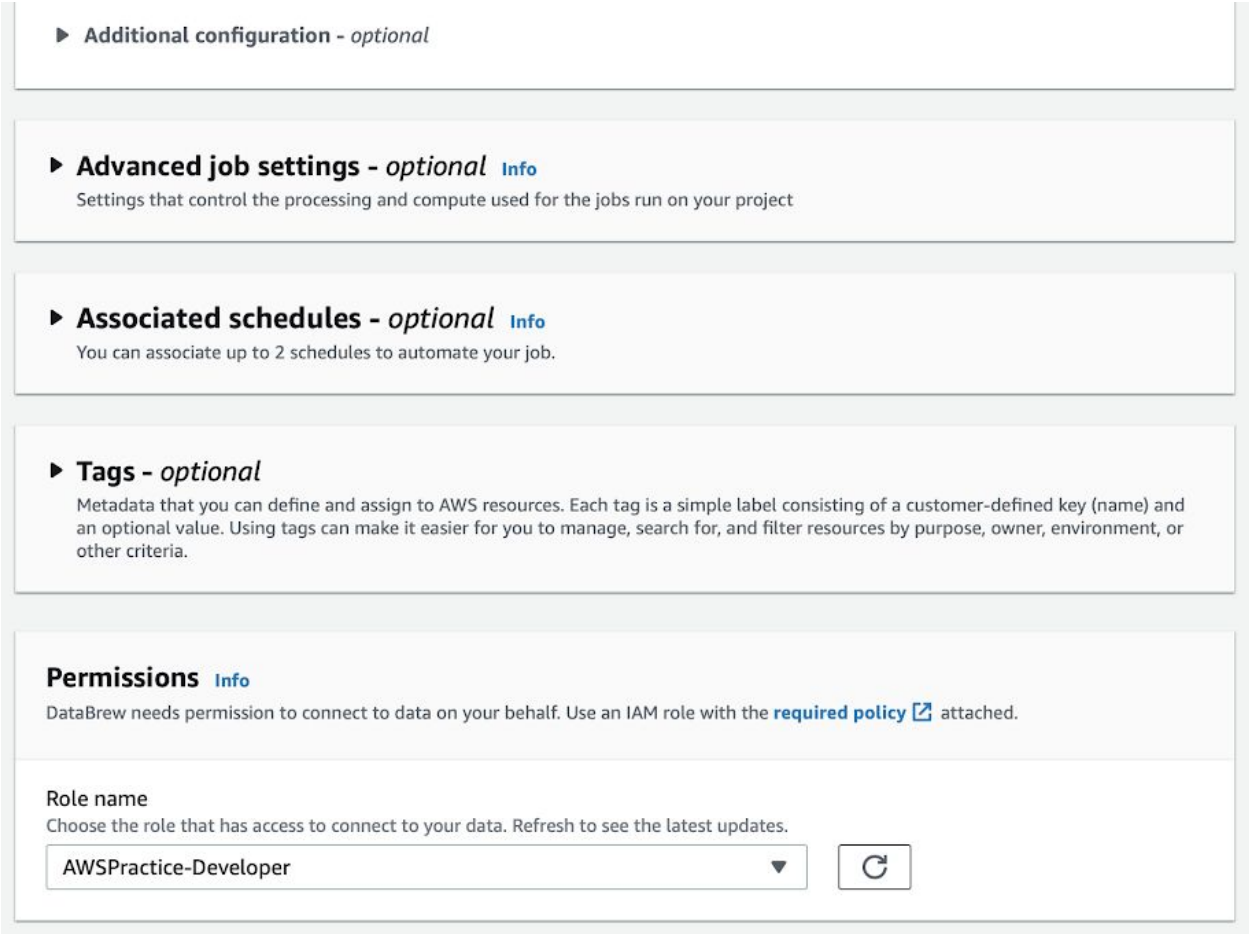
4. For Job input, provide details on the job that you want to create. What to provide depends on what kind of job you want to create:
    - A recipe job uses a DataBrew recipe to transform a dataset. For Job input, provide the name of the dataset to be processed, and the recipe to use. To use a recipe, make sure to publish it first.
      Tip
      If you've been working on a DataBrew project, you can simply choose the project name. DataBrew infers the attached dataset and recipe from the project.
    - A profile job requires only the name of a dataset for Job input.
5. For Job output settings, choose one of the available data output formats, optional compression, and an optional custom delimiter. Supported delimiters for output files are the same as those for input: comma, colon, semicolon, pipe, tab, caret, and space.
   Provide a destination for the job output. To do this, enter the location of an Amazon S3 bucket and folder where you want the output to be written.



6. (Optional) Choose Enable encryption for job output to encrypt the job output that DataBrew writes to Amazon S3, and then choose the encryption method:

- Use SSE-S3 encryption – The output is encrypted using server-side encryption with Amazon S3–managed encryption keys.
- Use AWS Key Management Service (AWS KMS) – The output is encrypted using AWS Key Management Service (AWS KMS). To use this option, choose the Amazon Resource Name (ARN) of the AWS KMS key that you want to use. If you don't have an AWS KMS key, you can create one by choosing Create an AWS KMS key.

7. For Access permissions, choose an IAM role that allows DataBrew to write to your Amazon S3 output location. For an S3 location owned by your AWS account, you can choose the AwsGlueDataBrewDataAccessRole service-managed role. Doing this allows DataBrew to access S3 resources that you own.



8. On Advanced job settings pane, you can choose more options for how your job is to run:
    - Maximum number of units – DataBrew processes jobs using multiple compute nodes, running in parallel. The default number of nodes is 5. The maximum number of nodes is 149.
    - Job timeout – If a job takes more than the number of minutes that you set here to run, it fails with a timeout error. The default value is 2,880 minutes, or 48 hours.
    - Number of retries – If a job fails while running, DataBrew can try to run it again. By default, the job isn't retried.
    - Enable Amazon CloudWatch Logs for job – Allows DataBrew to publish diagnostic information to CloudWatch Logs. These logs can be useful for troubleshooting purposes, or for more details on how the job is processed.
9. For Schedule jobs, you can apply a DataBrew job schedule so that your job runs at a particular time, or on a recurring basis. For more information, see Automating job runs with a schedule.
10. When the settings are as you want them, choose Create job. Or, if you want to run the job immediately, choose Create and run the job.

You can monitor your job's progress by checking its status while the job is running. When the job run is complete, the status changes to Succeeded.

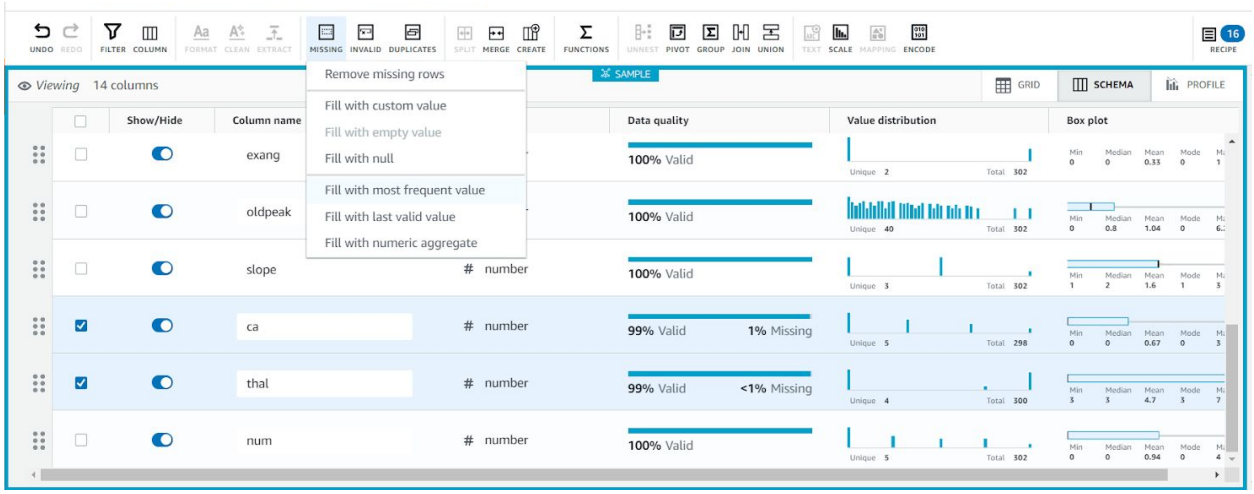
The job output is now available at your chosen Amazon S3 location

Part D: Data Validation

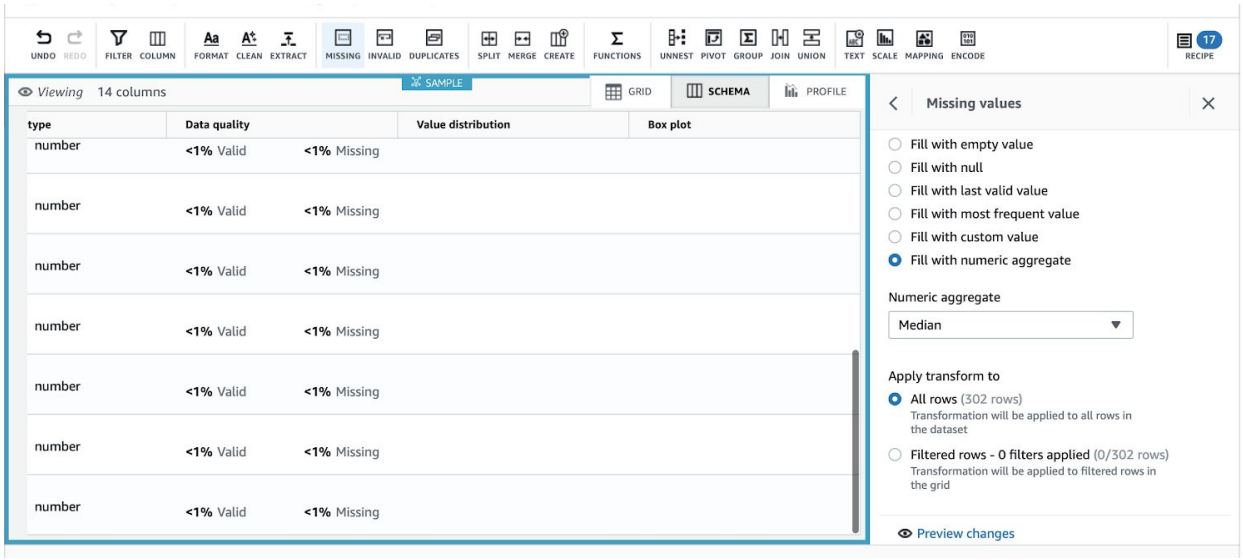
After profiling which provides us with value distribution and box plot for mean,median,mode

Step 1: Change the column names of schema

The changes which will be added to the recipe.

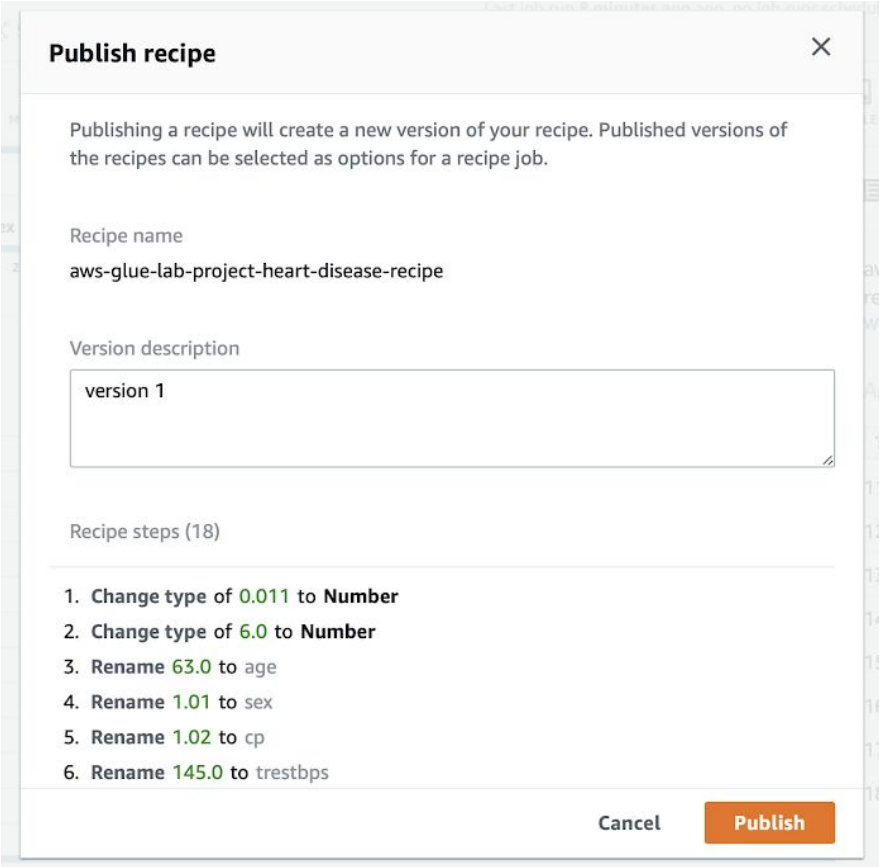


Step 2: Fill the missing values with median using the box plot graph

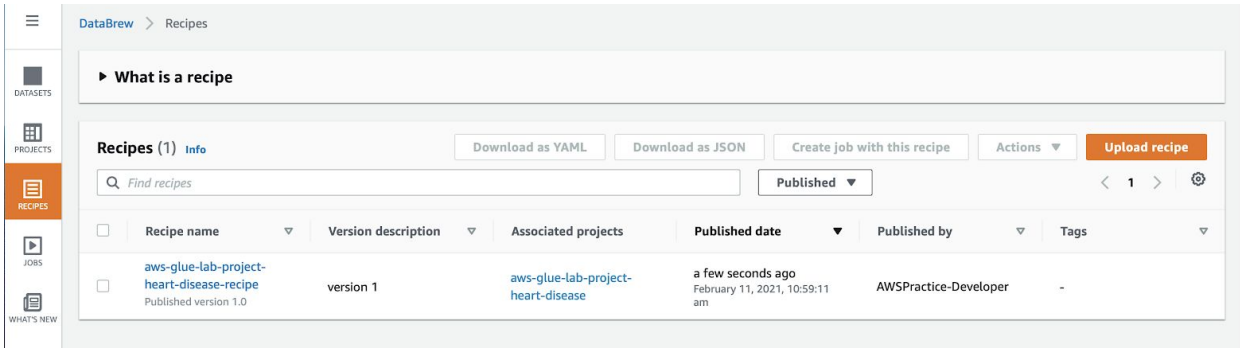


All the steps will be added to recipe

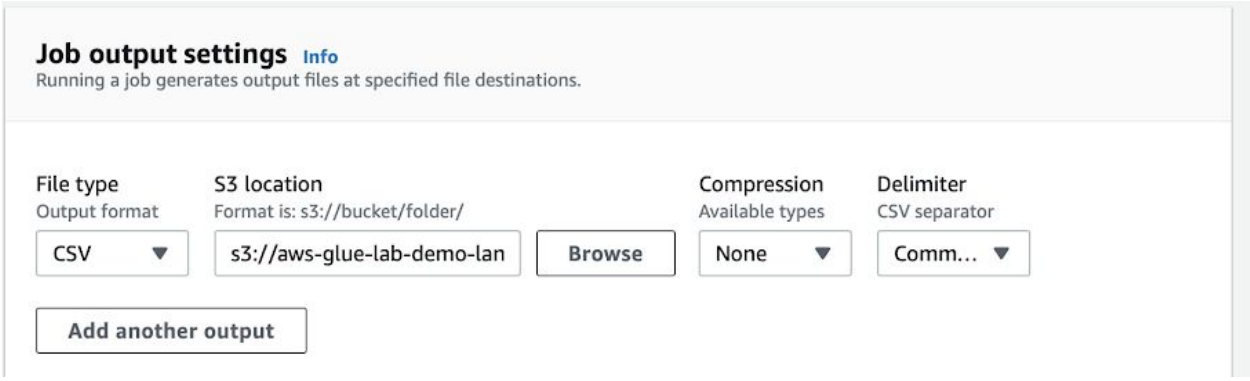
Step 3: Publish / save the recipe for future use



Step 4: View the recipe



Step 5: Save the output location of job run



Section D: AWS Glue Studio

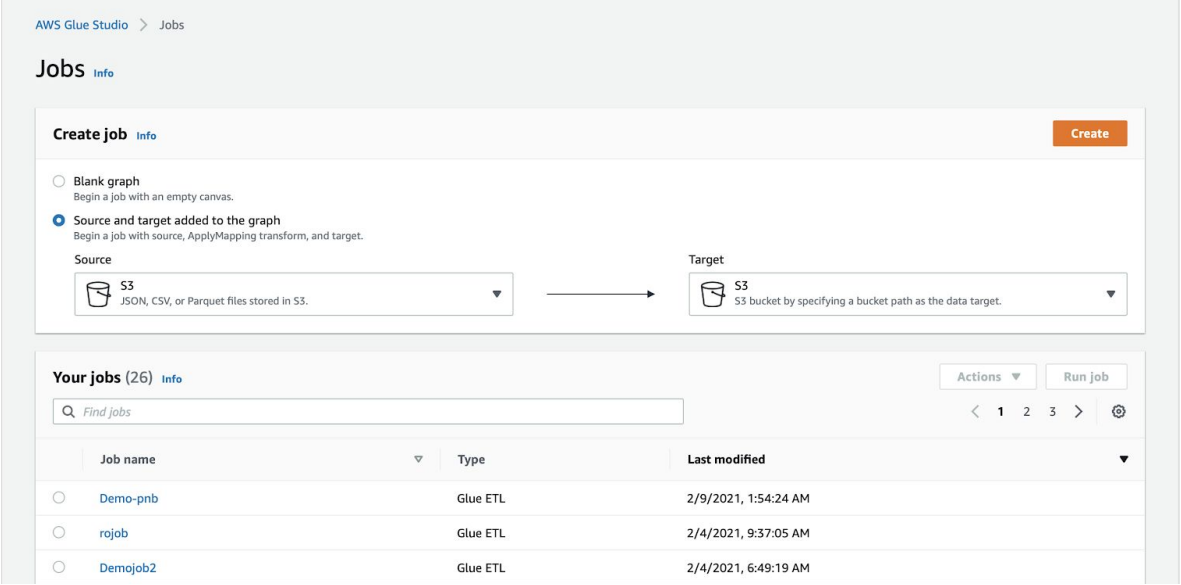
AWS Glue Studio is an easy-to-use graphical interface that speeds up the process of authoring, running, and monitoring extract, transform, and load (ETL) jobs in AWS Glue. The visual interface allows those who don't know Apache Spark to design jobs without coding experience and accelerates the process for those who do.

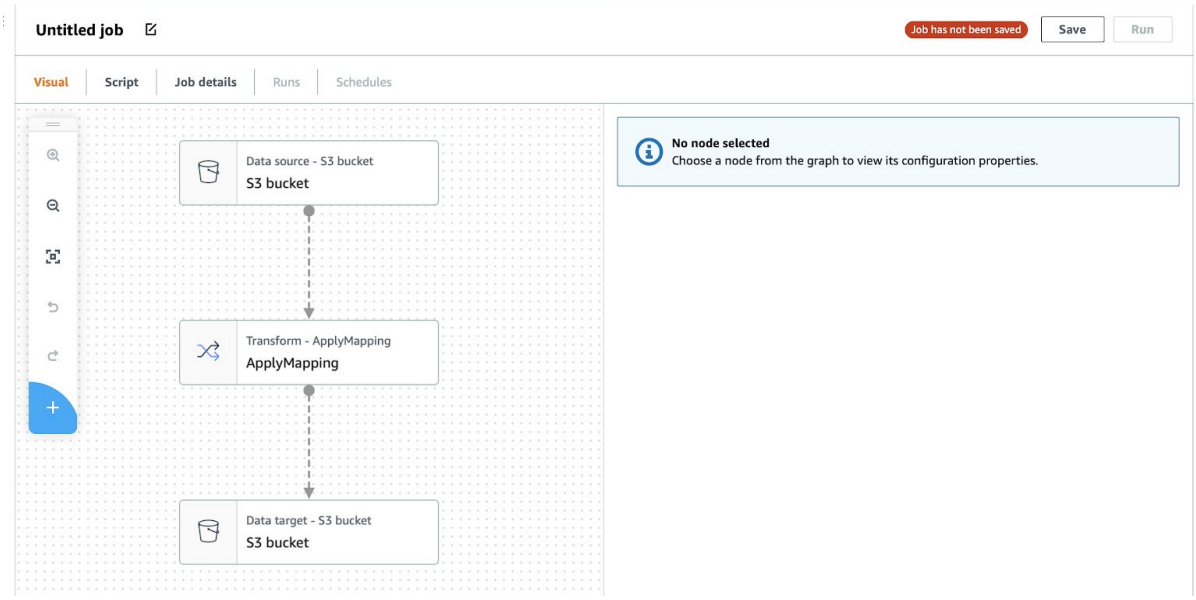
AWS Glue Studio was designed to help you create ETL jobs easily. After you design a job in the graphical interface, it generates Apache Spark code for you, abstracting users from the challenges of coding. When the job is ready, you can run it and monitor the job status using the integrated UI. AWS Glue Studio supports different types of data sources, both structured and

semi-structured, and offers data processing in real time and batch. You can extract data from sources like Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon Kinesis, and Apache Kafka. It also offers Amazon S3 and tables defined in the AWS Glue Data Catalog as destinations.

You can use AWS Glue Studio to create jobs that extract structured or semi-structured data from a data source, perform a transformation of that data, and save the result set in a data target.Here we are going to do same process as AWS Glue databrew or we can extract from one location and load to another location.

Step 1: Start the job creation process

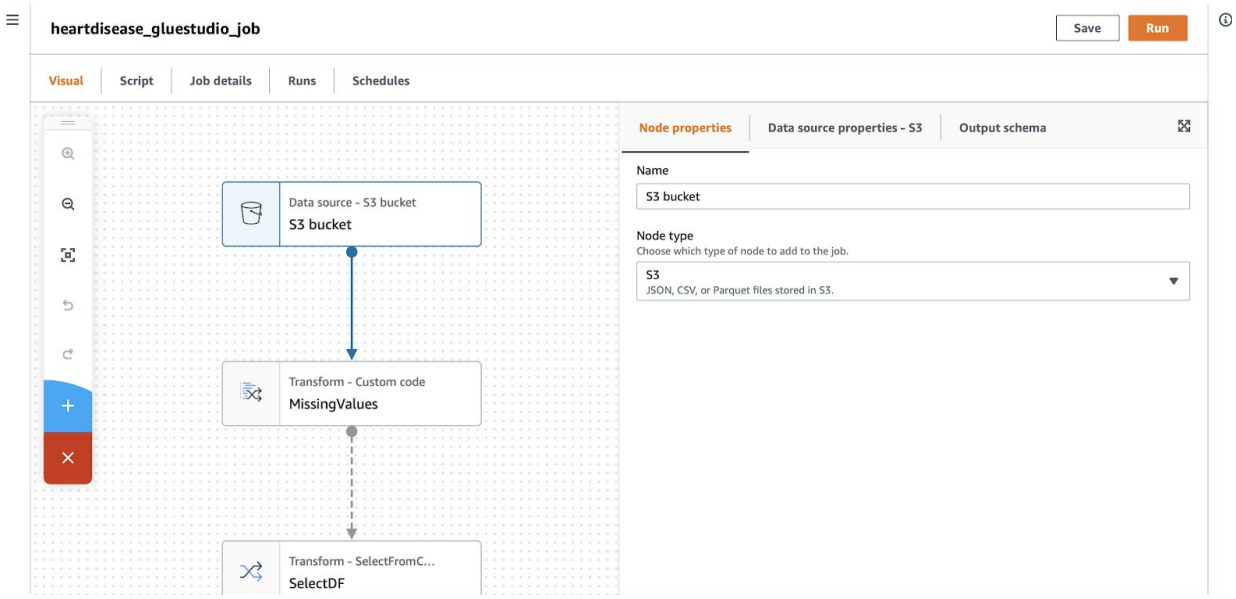


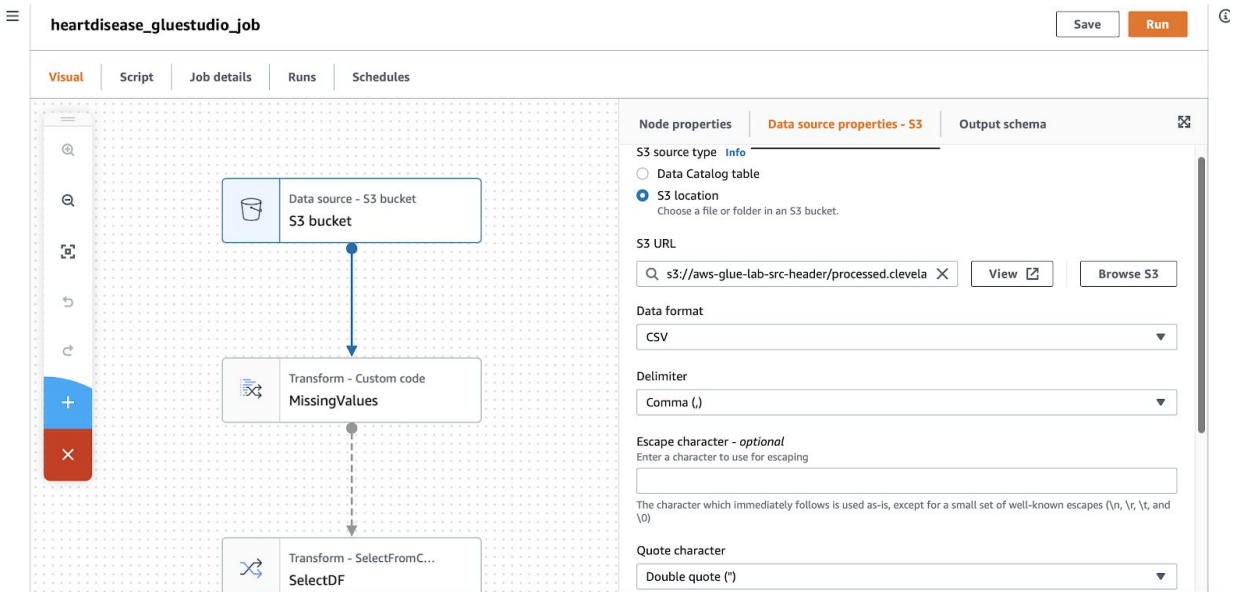


Remove the apply mapping transformation by deleting it

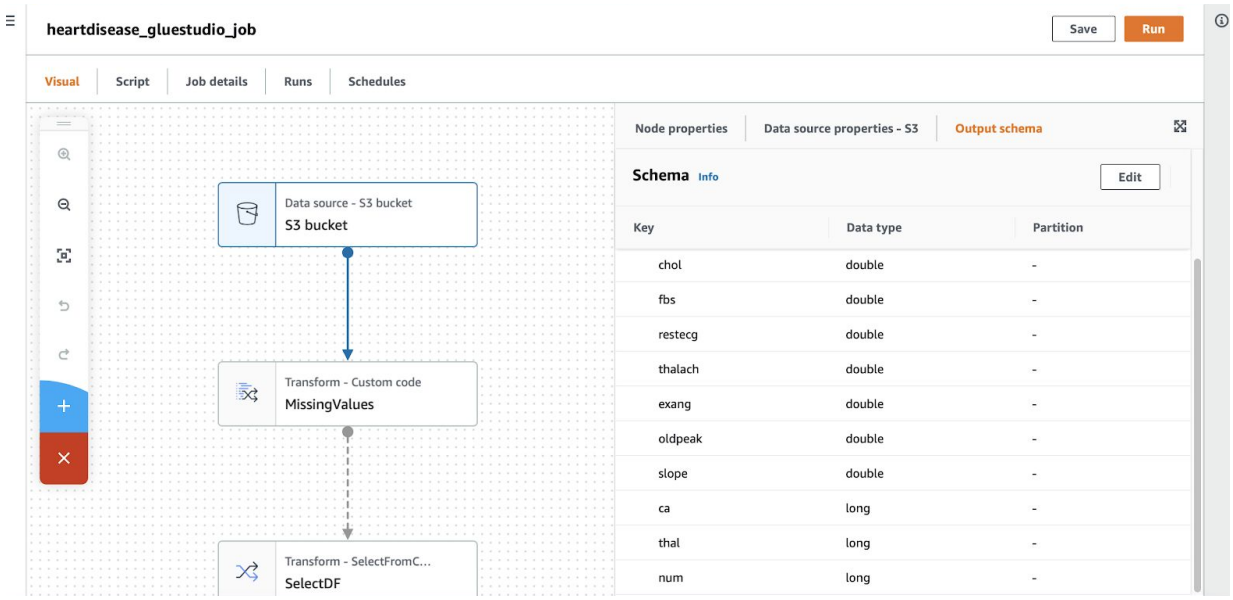
Step 2: Edit the data source node in the job graph

Rename the job name



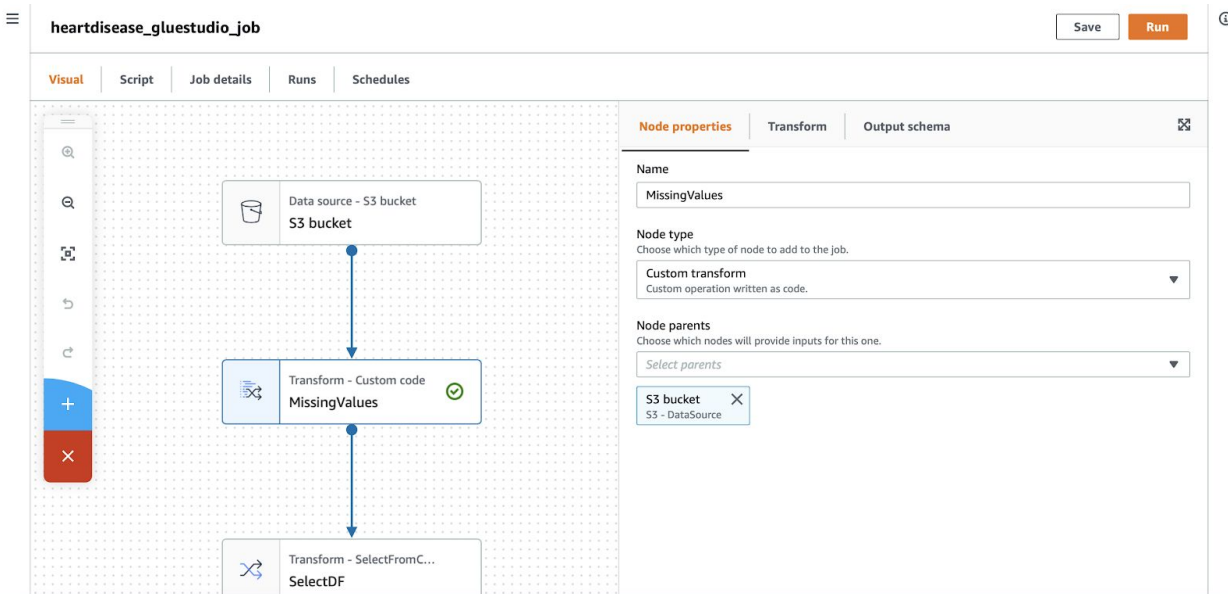


Edit the schema if data types are different



Step 3: Edit the transform node of the job

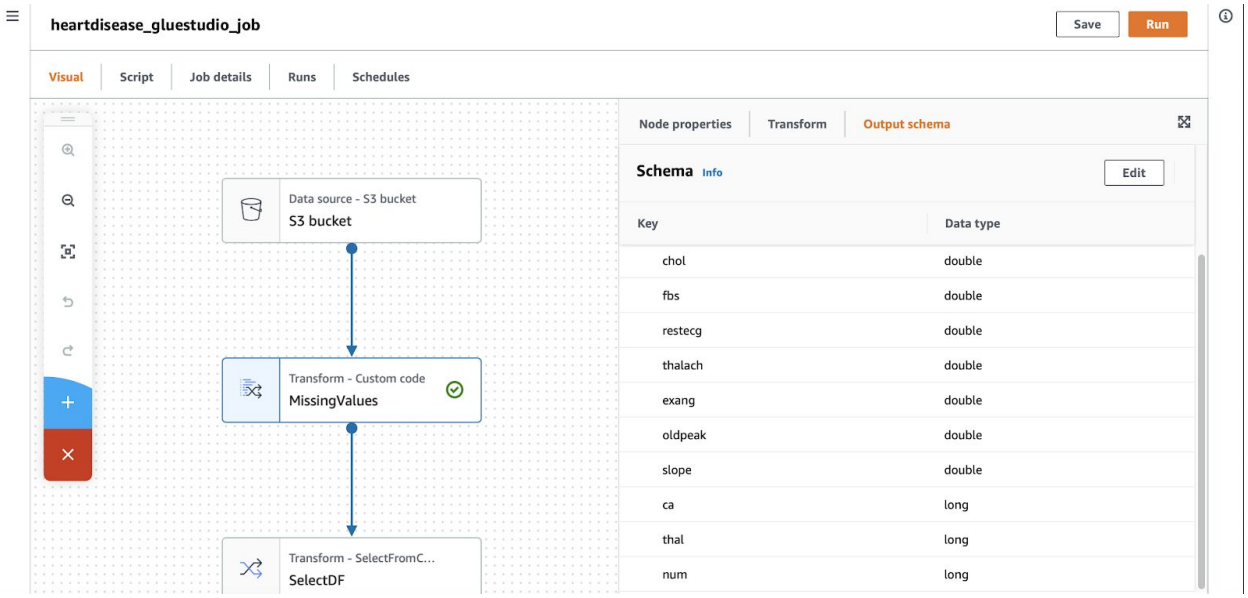
Please select a custom transform and copy paste the below function



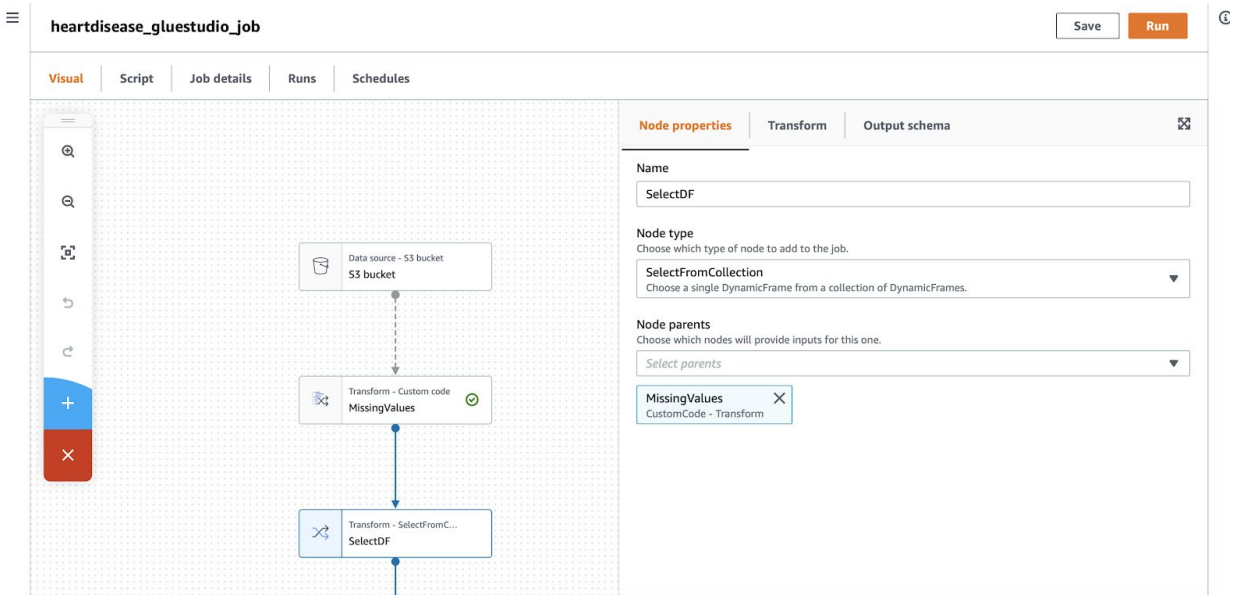

```
def MyTransform (glueContext, dfc) -> DynamicFrameCollection:

    #Select the dataframe from dynamicdataframecollection object
    selected = dfc.select(list(dfc.keys())[0]).toDF()

    #import regular expression function from pyspark library
    from pyspark.sql.functions import regexp_replace as regxx

    #find the mode for missing values for "ca" column using pyspark
sql
    modeCa = selected.groupby("ca").count().orderBy("count",
ascending=False).first()[0]
```

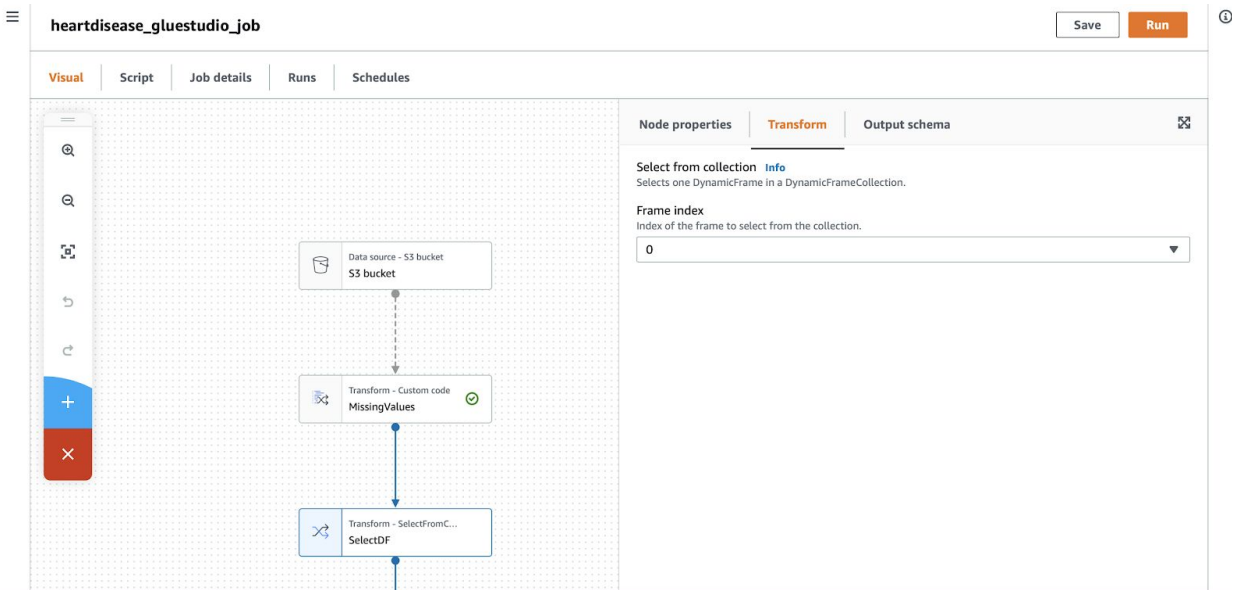
```
    #fill the missing values with mode values
    newDF = selected.withColumn('ca', regxx('ca', '\?', modeCa))

    #find the mode for missing values for "thal" column using pyspark
sql
    modeThal = newDF.groupby("thal").count().orderBy("count",
ascending=False).first()[0]

    #fill the missing values with mode values
    newDF = newDF.withColumn('thal', regxx('thal', '\?', modeThal))

    #Store the dataframe back to dynamicframe
    results = DynamicFrame.fromDF(newDF, glueContext, "results")

    #return the dynamicframe as dynamicframecollection object
    return DynamicFrameCollection({"results": results}, glueContext)
```

The objective of the above function is to fill the missing values for the “Ca” column and “Thal” column. It is achieved by finding the mode of that column using pyspark sql functions.

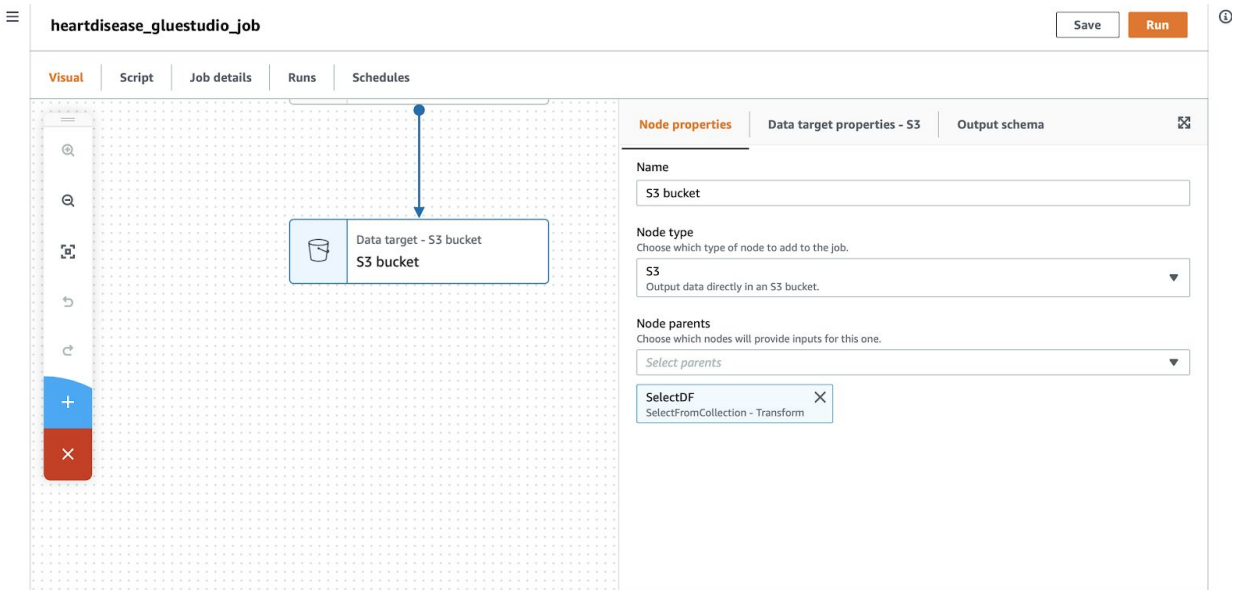


This creates automatically another transformation known as SelectFromCollection which can be used to select a dynamic frame from a dynamic frame collection.

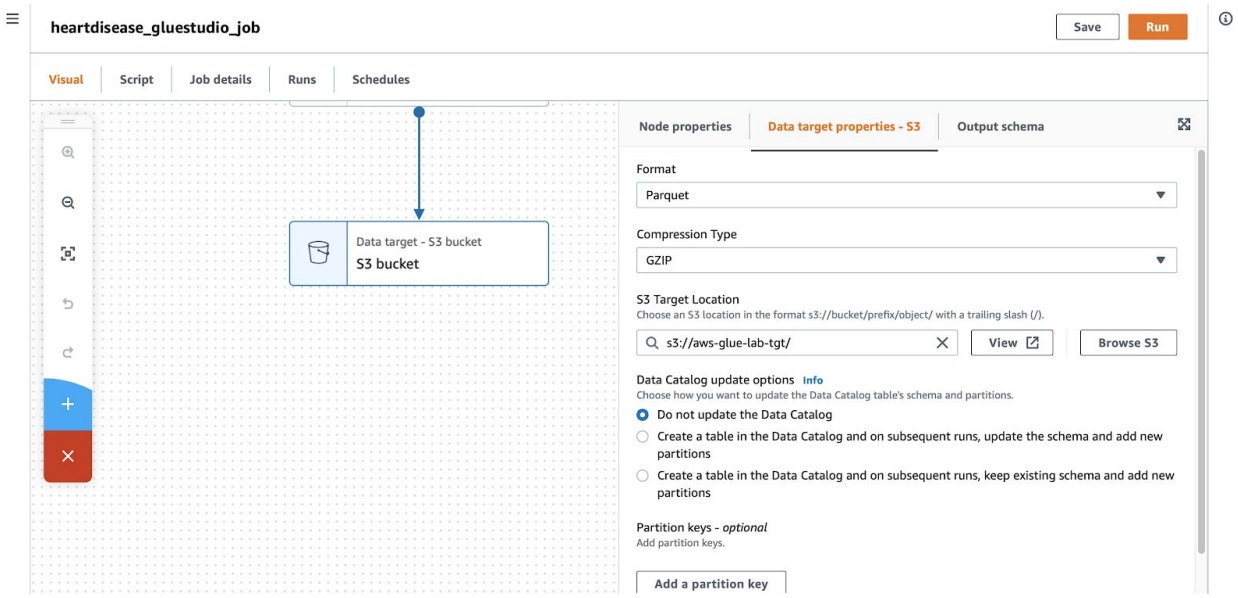




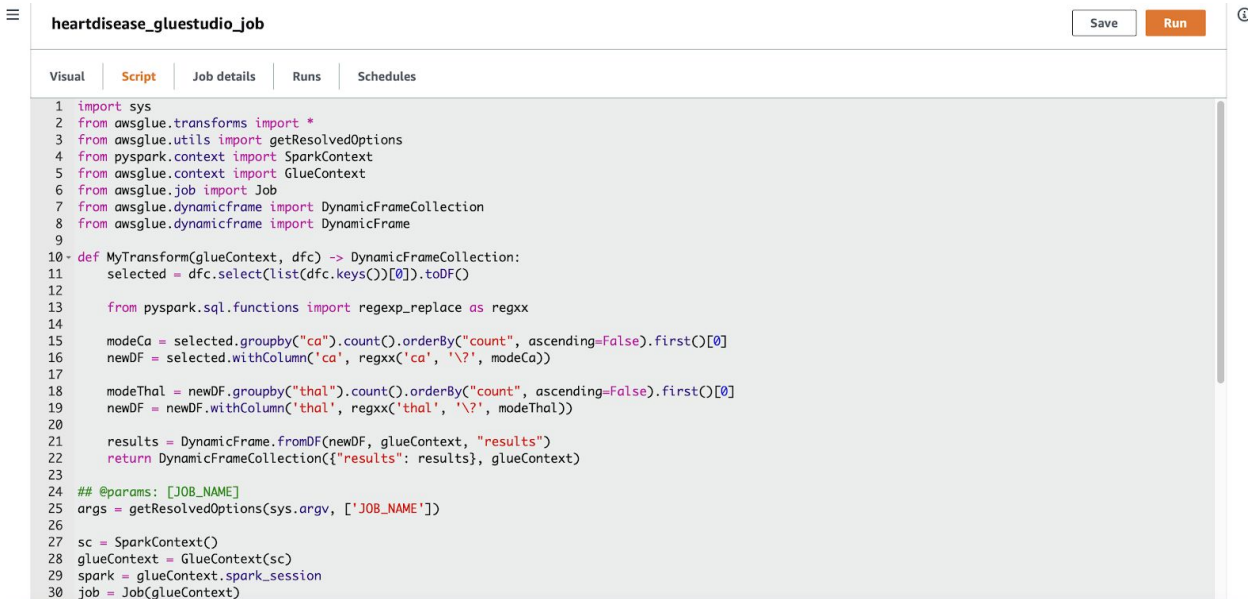
Step 4: Edit the data target node of the job



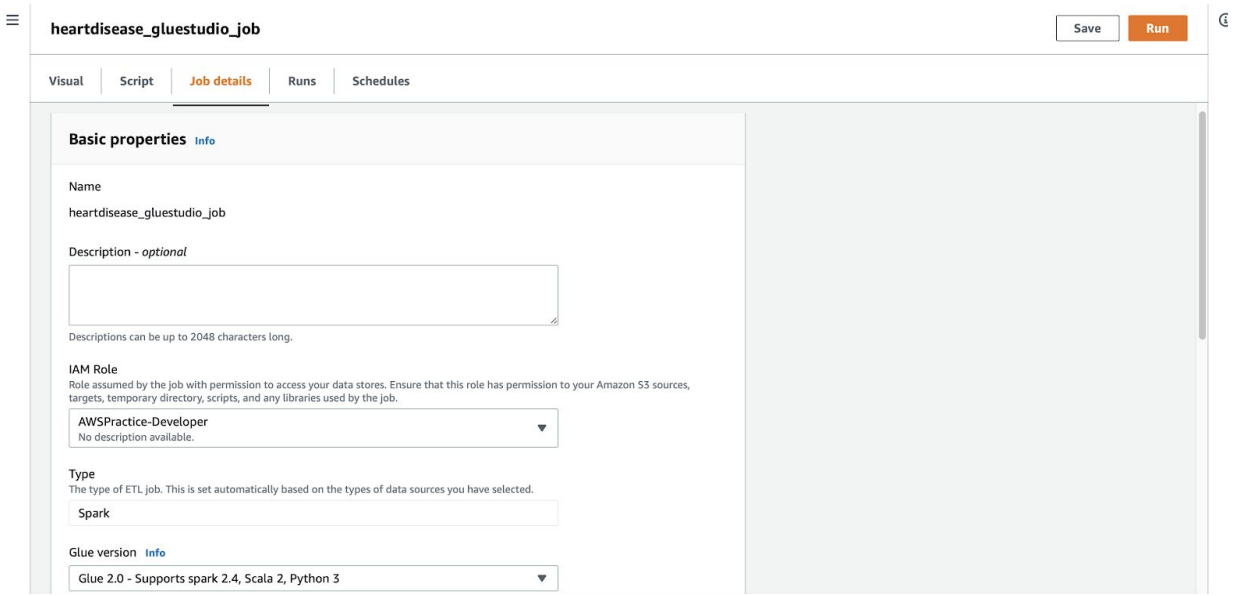
Specify the output format and compression type

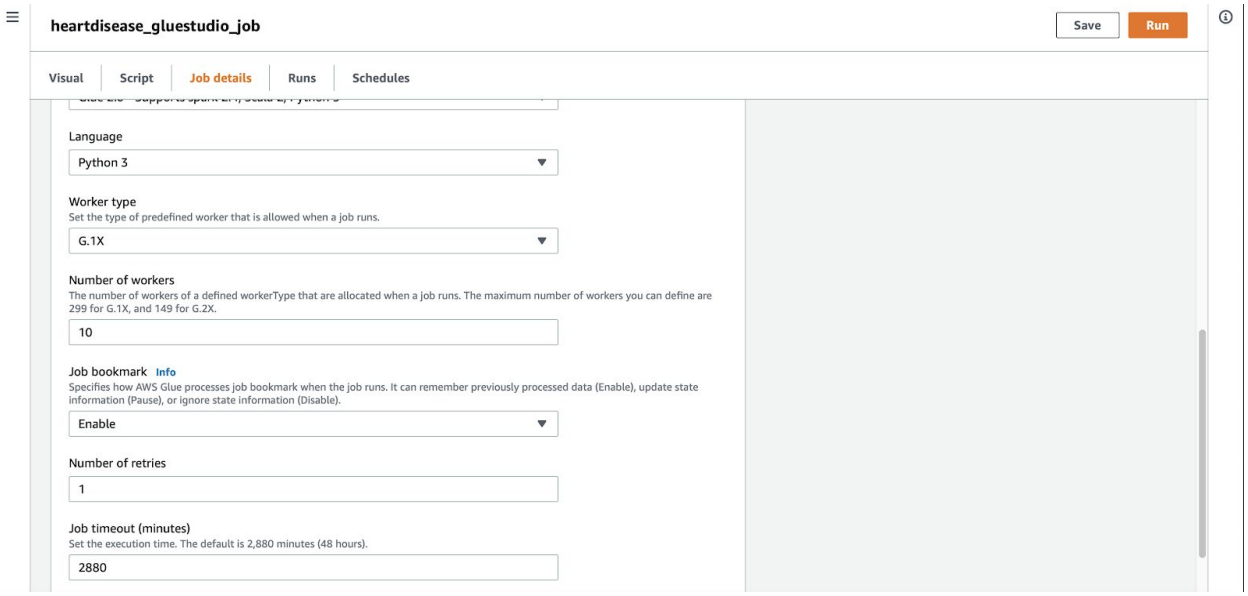


Step 5: View the final job script



Step 6: Specify the job details and save the job





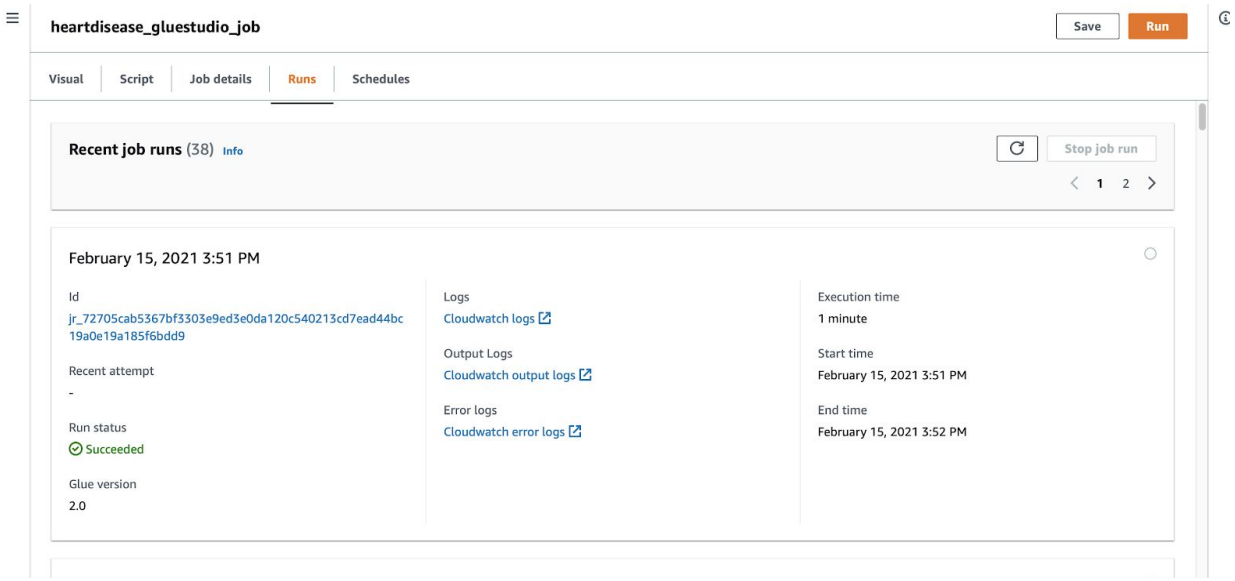
Step 7: Run the job

Now that the job has been saved, you can run the job. Choose the **Run** button at the top of the page. You should then see a notification that the job was successfully started.

You can choose either the link in the notification for **Run Details**, or choose the **Run Details** tab to view the run status of the job.

Note: avoid concurrent job runs by clicking once for run
Failed to start job - [gluestudio-service.us-east-1.amazonaws.com] startJobRun: ConcurrentRunsExceededException: Concurrent runs exceeded for heartdisease_gluestudio_job (Service: AWSGlue; Status Code: 400; Error Code: ConcurrentRunsExceededException; Request ID: 21fcec38-07c2-4264-8a89-6d928b489627; Proxy: null)

On the **Run Details** tab, there is a card for each recent run of the job with information about that job run. For more information about the job run information, see View information for recent job runs.

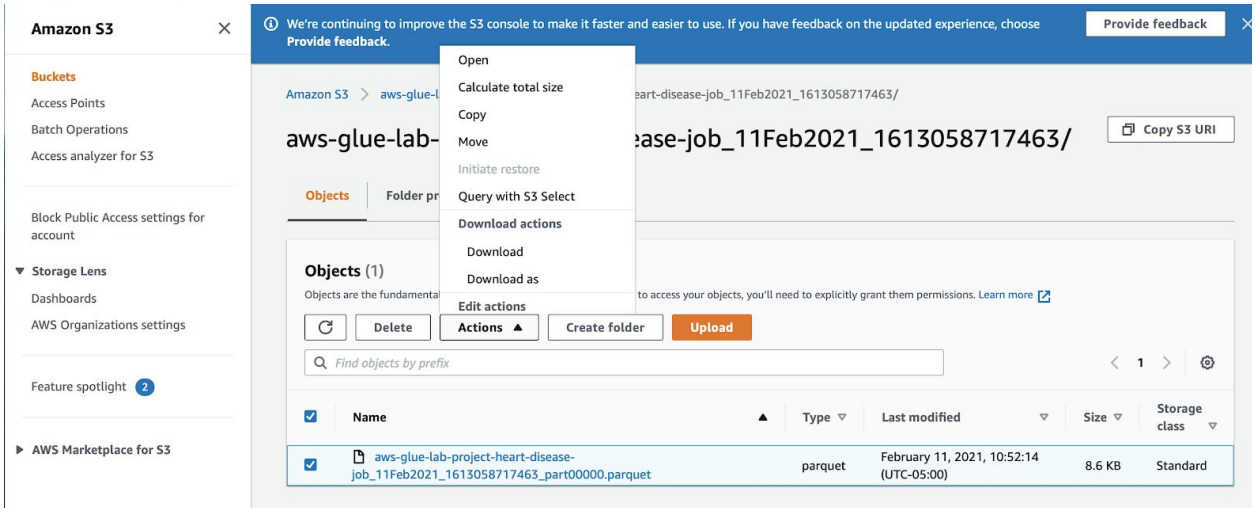


Section E: AWS S3 Select

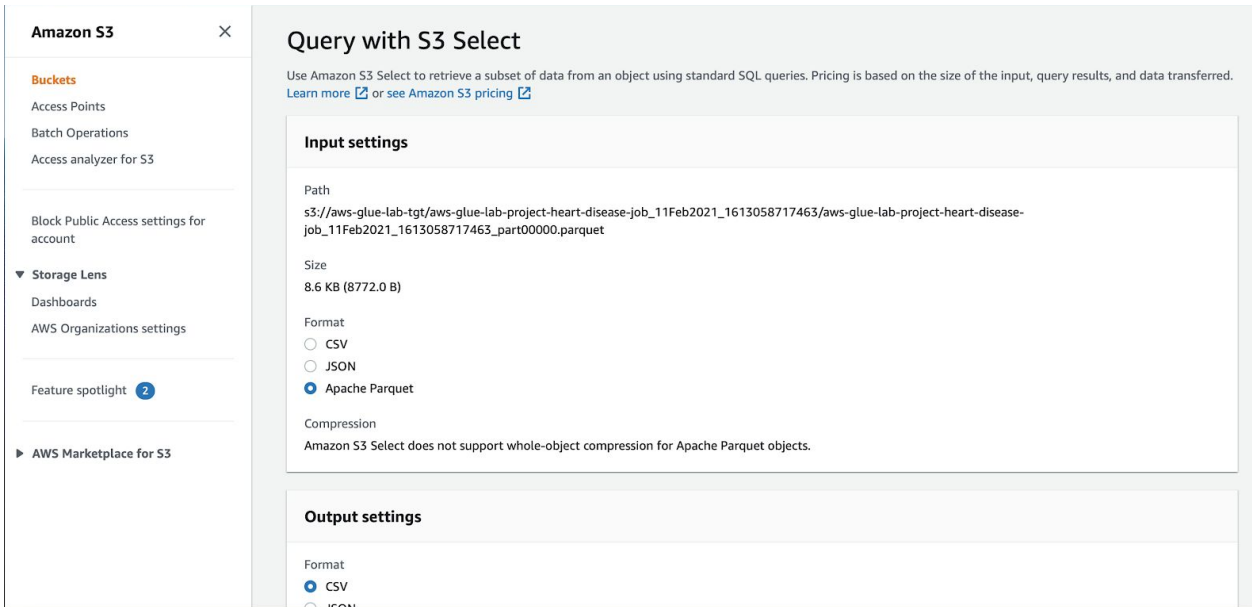
S3 Select, launching in preview now generally available, enables applications to retrieve only a subset of data from an object by using simple SQL

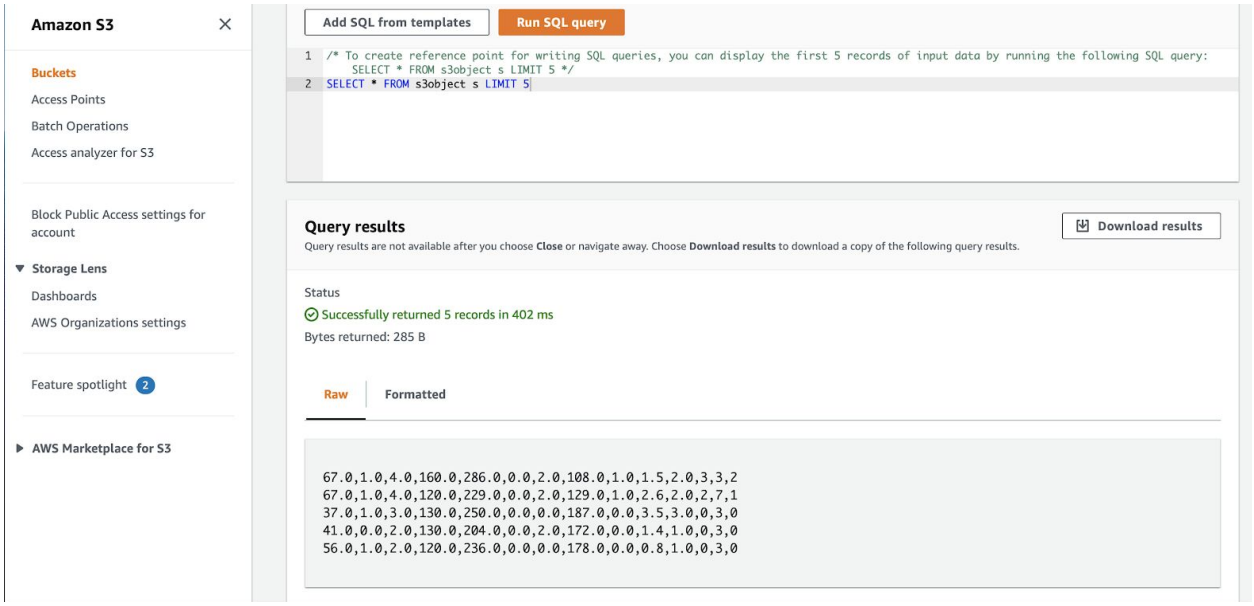
expressions. By using S3 Select to retrieve only the data needed by your application, you can achieve drastic performance increases.

Step 1: View the output in s3 bucket and use s3 select for querying



Step 2: Enter the select query and select the file format and output format





Section F: AWS Athena

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run.

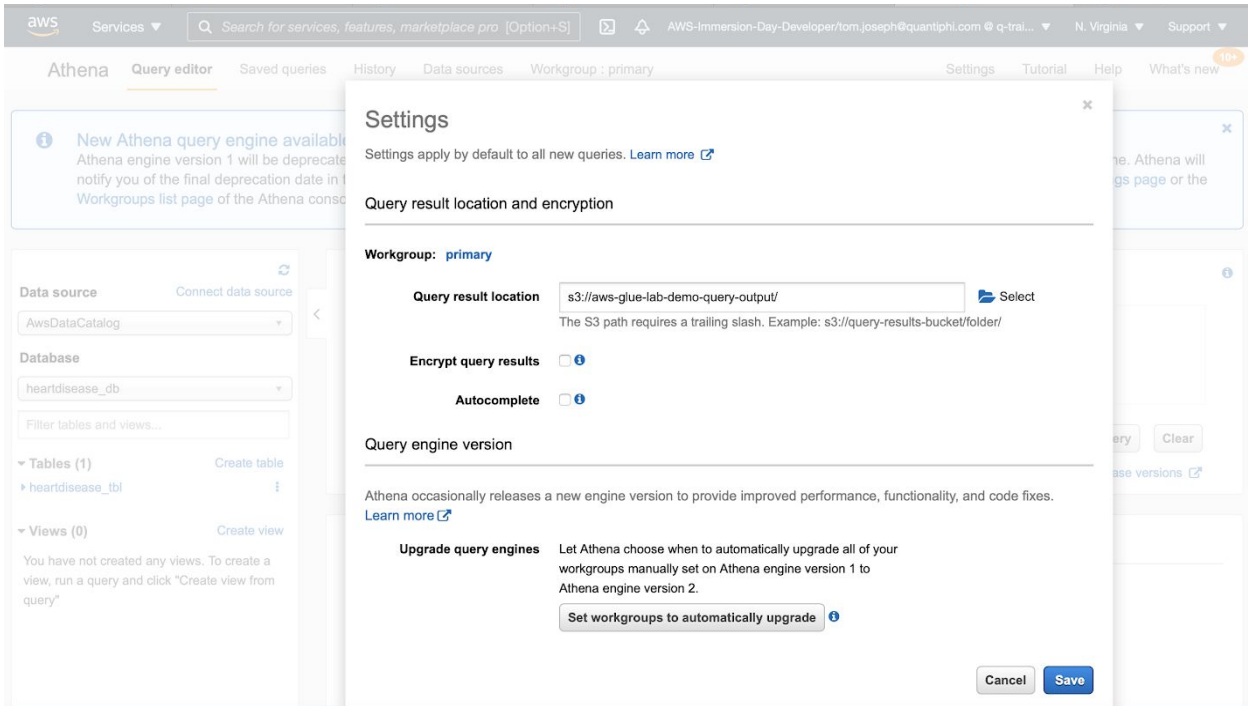
Athena is easy to use. Simply point to your data in Amazon S3, define the schema, and start querying using standard SQL. Most results are delivered within seconds. With Athena, there's no need for complex ETL jobs to prepare your data for analysis. This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets.

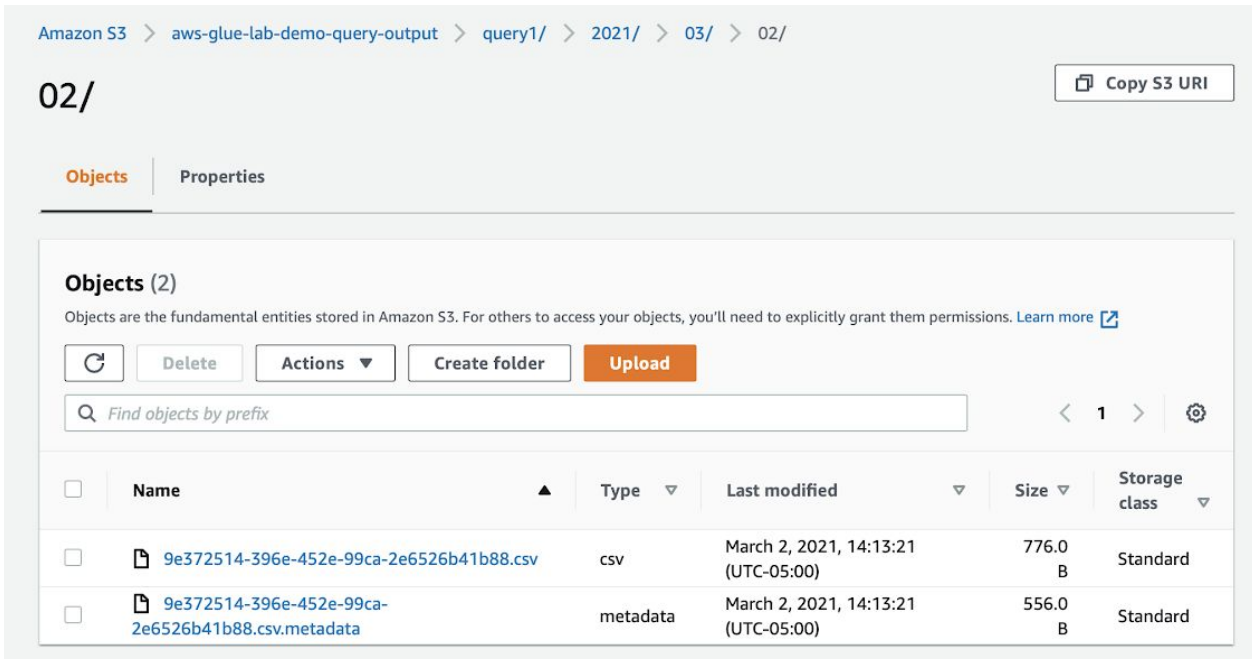
Athena is out-of-the-box integrated with AWS Glue Data Catalog, allowing you to create a unified metadata repository across various services, crawl data sources to discover schemas and populate your Catalog with new and modified table and partition definitions, and maintain schema versioning.

Step 1: Save results of query into s3

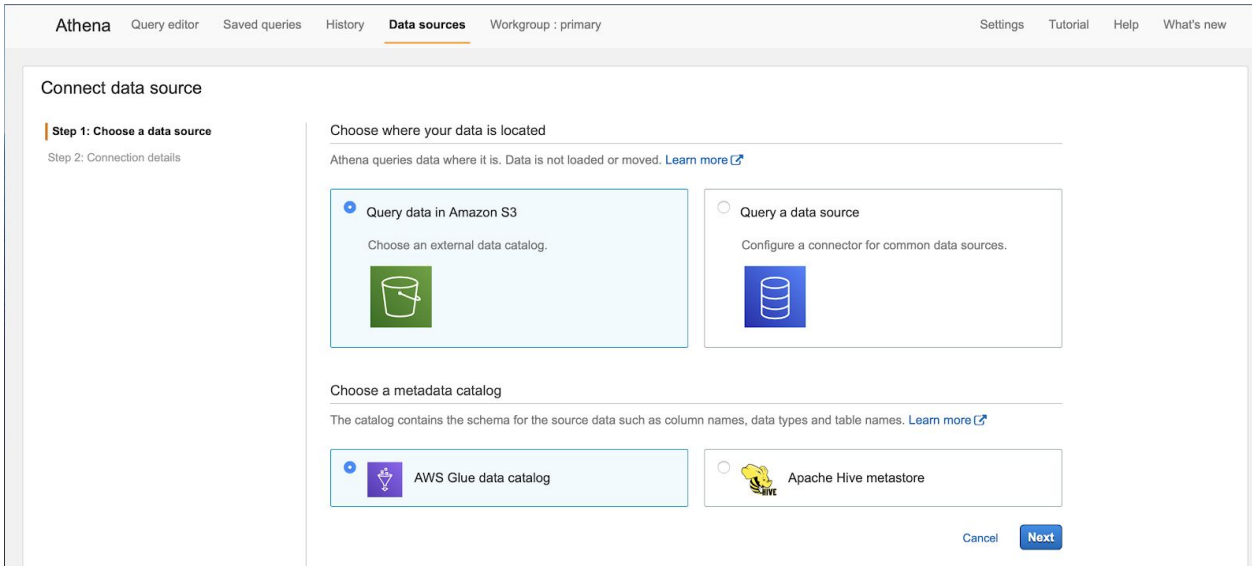
To specify a client-side setting query result location using the Athena console

1. From the navigation bar, choose Settings.
2. For Query result location, enter the path to an existing Amazon S3 folder, including the trailing slash.

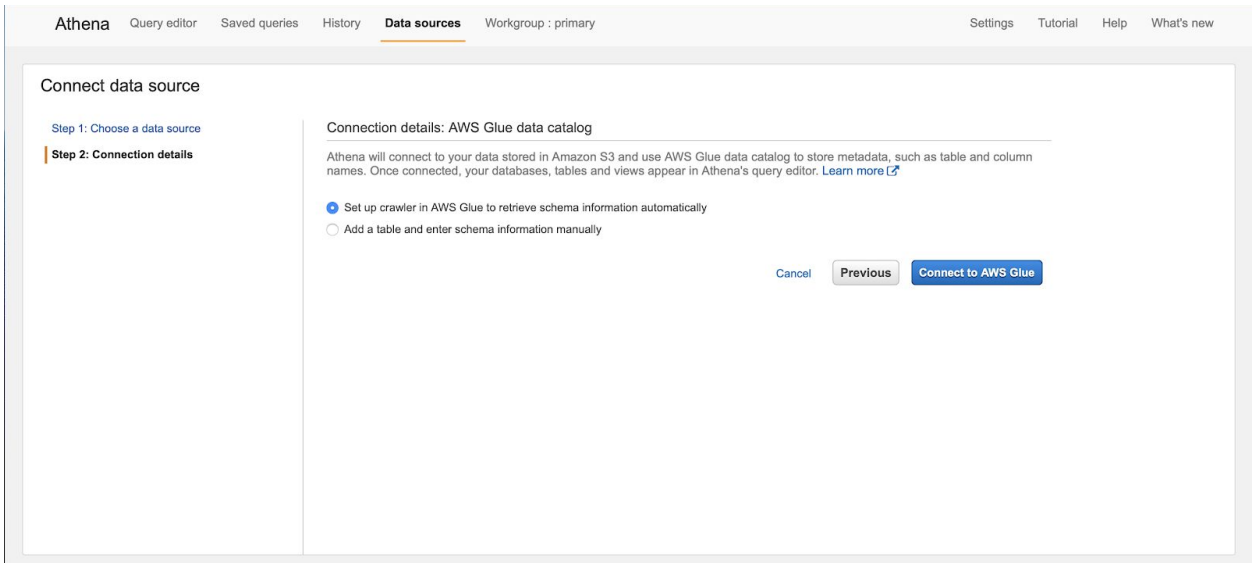




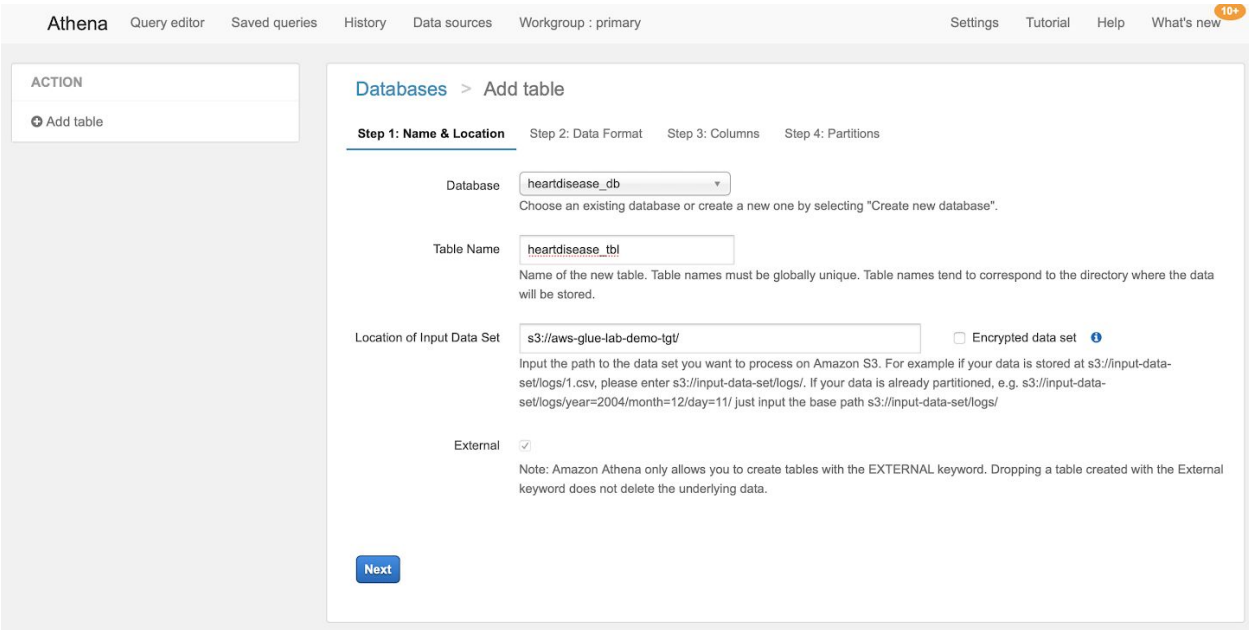
Step 2: Query using AWS Athena by adding the data source



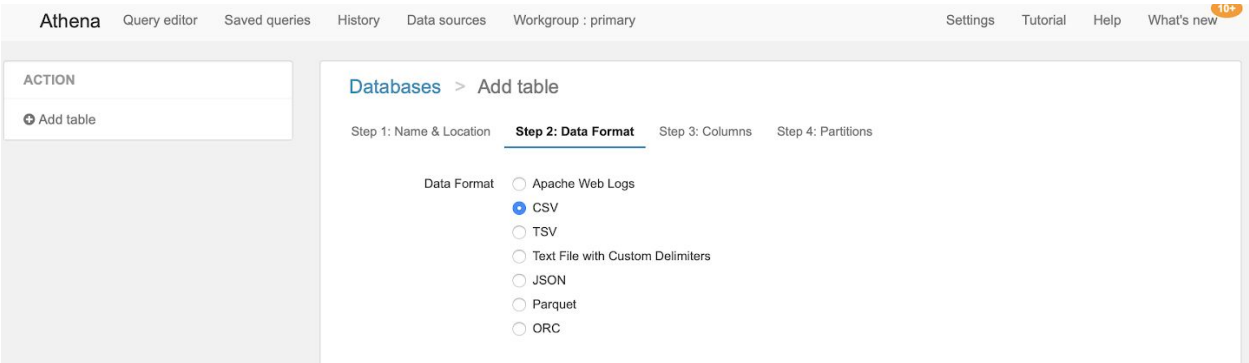
Step 3: Select the option select table and schema manually



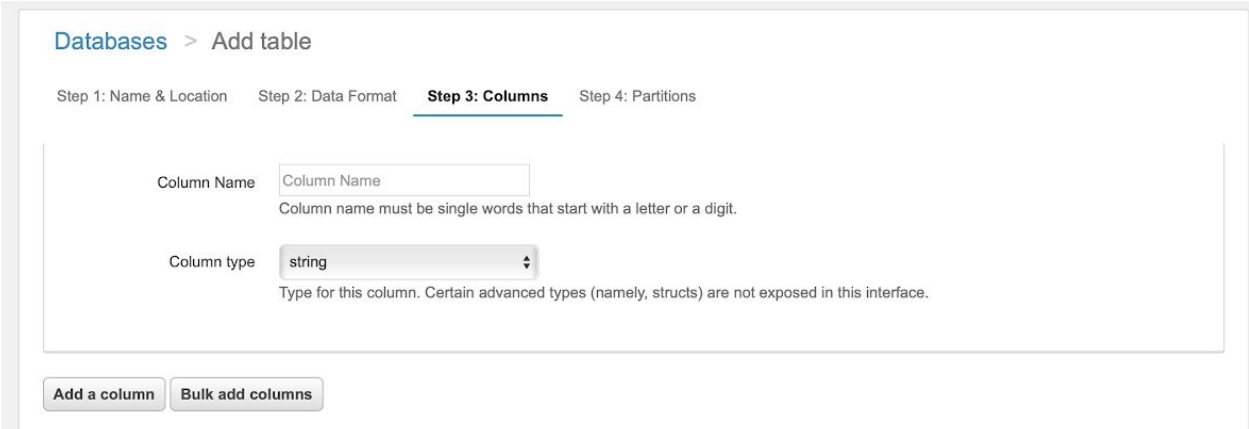
Step 4: Provide database name, table name and S3 target location



Step 5: Select file format as CSV

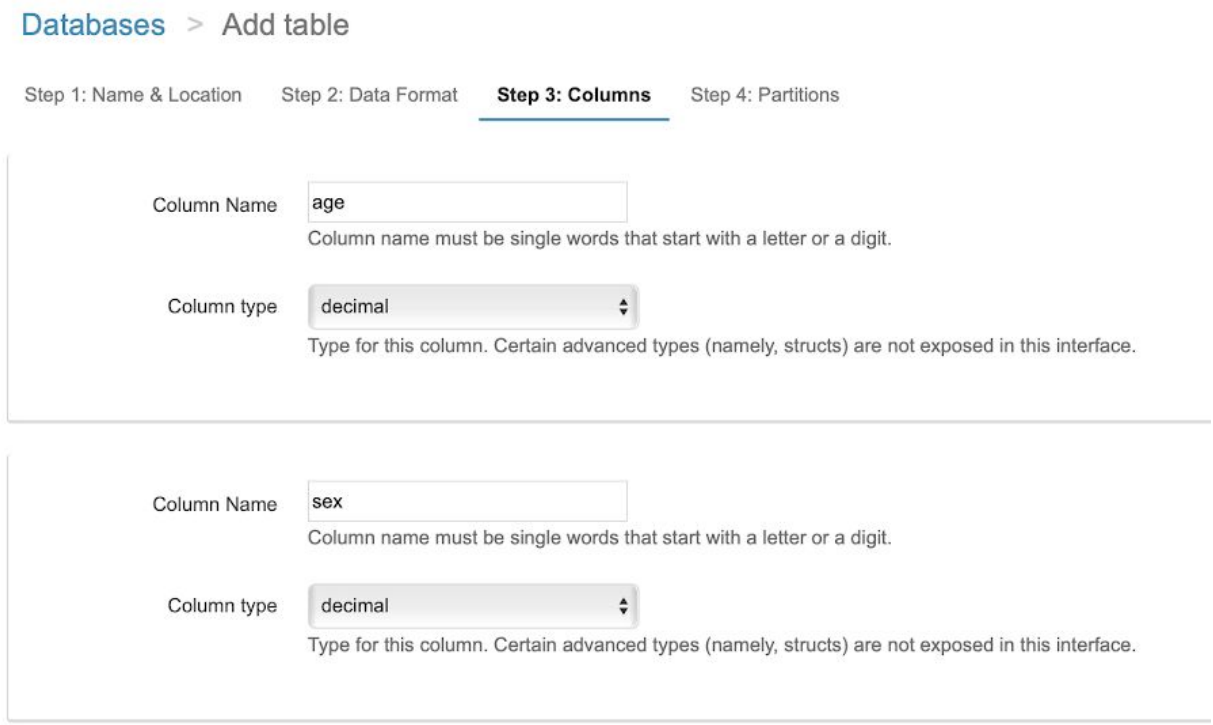


Step 6: Add the column and data type using bulk column feature

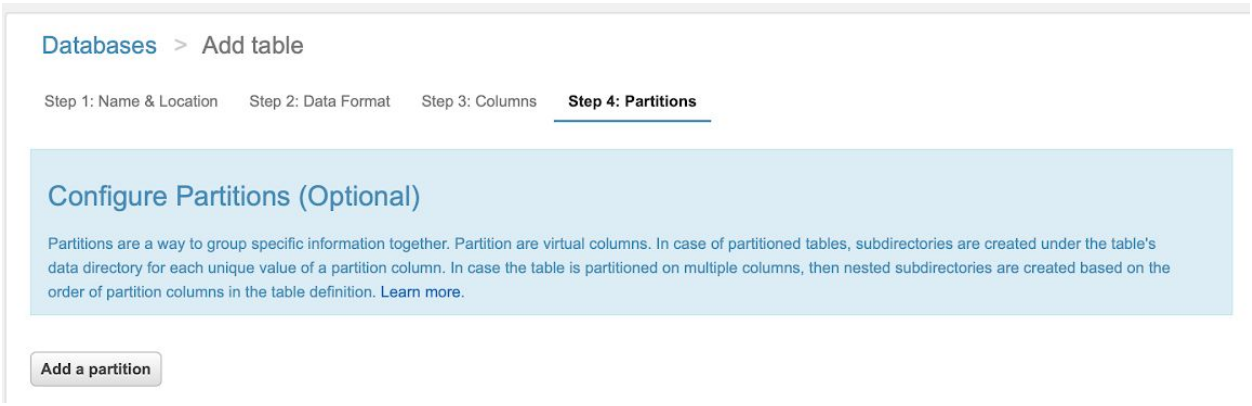



```
Age int,sex int,cp int,trestbps double,chol double,fbs double,restecg
int,thalach double,exang int,oldpeak double,slope int,ca int,thal
int,num int
```

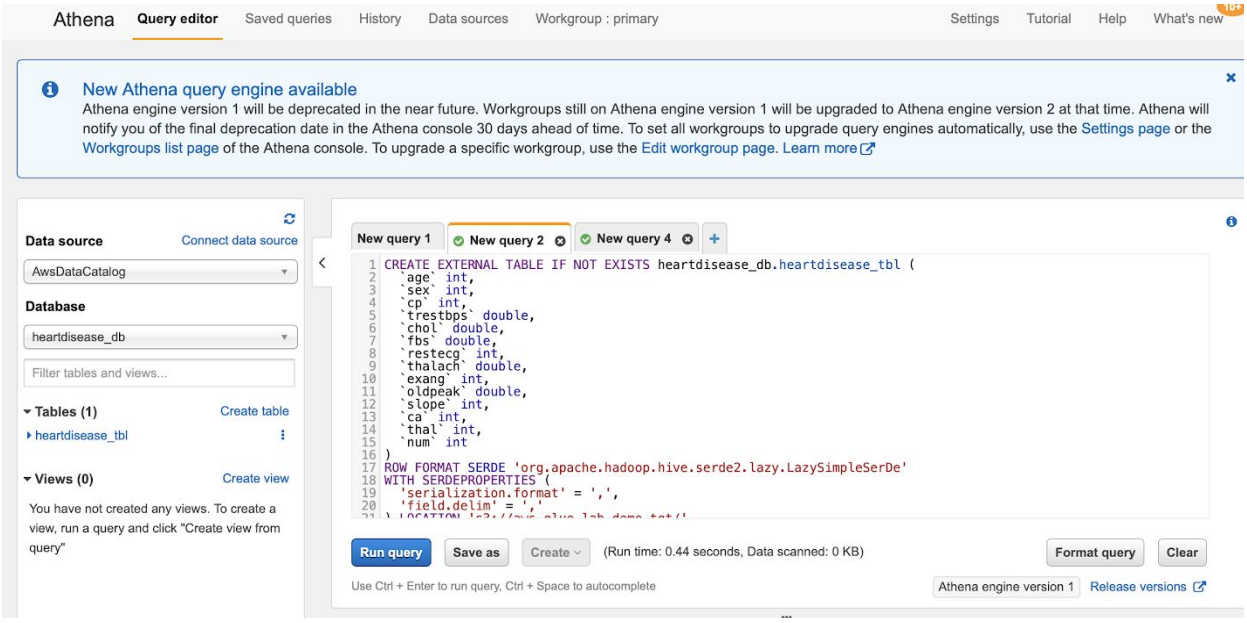
After adding it will update in screen



Step 7: Create Partitions if necessary



Step 8: Save it and the table ddl will updated in the query editor and table will be created




```
CREATE EXTERNAL TABLE IF NOT EXISTS heartdisease_db.heartdisease_tbl
(
  `age` int,
  `sex` int,
  `cp` int,
  `trestbps` double,
  `chol` double,
  `fbs` double,
  `restecg` int,
  `thalach` double,
  `exang` int,
  `oldpeak` double,
  `slope` int,
  `ca` int,
  `thal` int,
  `num` int
```

```
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = ',',
  'field.delim' = ','
) LOCATION 's3://aws-glue-lab-demo-tgt/'
TBLPROPERTIES ('has_encrypted_data'='false');
```

Step 9: Query the table using AWS Athena