



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

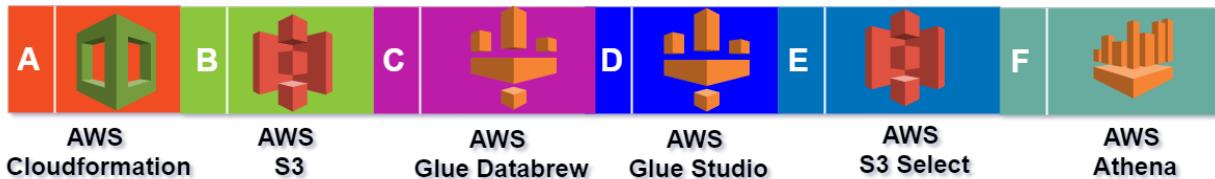
Introduction

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

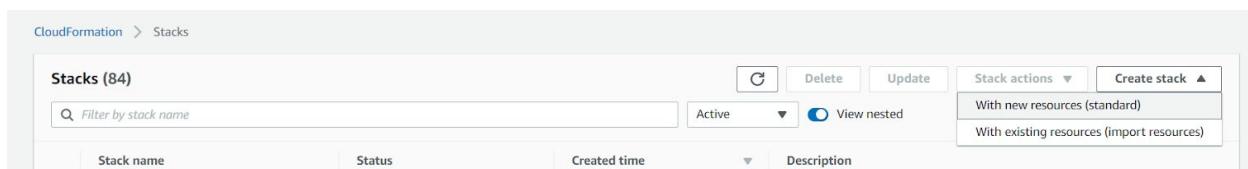
Your account has all the necessary permissions for creating and running a job for an Amazon S3 data source and data target. You have created an AWS Identity and Access Management role for the job to use. You can also choose an IAM role for the job that includes permissions for all your data sources,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

Step 1
Specify template

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Create stack

Prerequisite - Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Template is ready Use a sample template Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL Upload a template file

Upload a template file
Choose file aws-glue-demo-template-v1.json
JSON or YAML formatted file

S3 URL: <https://s3-external-1.amazonaws.com/cf-templates-7uxgevs7zmi-us-east-1/20210551rc-aws-glue-demo-template-v1.json>

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

CloudFormation > Stacks > Create stack

Step 1
Specify template

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Specify stack details

Stack name

Stack name

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

No parameters
There are no parameters defined in your template

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

CloudFormation > Stacks > Create stack

Step 1
Specify template

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Configure stack options

Tags
You can specify tags (key-value pairs) to apply to resources in your stack. You can add up to 50 unique tags for each stack. [Learn more](#)

Permissions
Choose an IAM role to explicitly define how CloudFormation can create, modify, or delete resources in the stack. If you don't choose a role, CloudFormation uses permissions based on your user credentials. [Learn more](#)

IAM role - optional
Choose the IAM role for CloudFormation to use for all operations performed on the stack.

⚠ AWS CloudFormation will use this role for all stack operations. Other users that have permissions to operate on this stack will be able to use this role, even if they don't have permission to pass it. Ensure that this role grants least privilege.

- Review and launch this stack creates AWS resources

CloudFormation > Stacks > Create stack

Step 1
Specify template

Step 2
Specify stack details

Step 3
Configure stack options

Step 4
Review

Review aws-glue-lab-cft

Step 1: Specify template Edit

Template

Template URL
<https://s3-external-1.amazonaws.com/cf-templates-7uxgevs7zmi-us-east-1/2021055IRc-aws-glue-demo-template-v1.json>

Stack description
create role for glue demo

[Estimate cost](#)

Step 2: Specify stack details Edit

Parameters (0)

Search parameters

Key	Value
-----	-------

Stack creation options

Rollback on failure
Enabled

Timeout
-

Termination protection
Disabled

► Quick-create link

Capabilities

ⓘ The following resource(s) require capabilities: [AWS::IAM::Role]

This template contains Identity and Access Management (IAM) resources. Check that you want to create each of these resources and that they have the minimum required permissions. In addition, they have custom names. Check that the custom names are unique within your AWS account. [Learn more](#)

I acknowledge that AWS CloudFormation might create IAM resources with custom names.

Cancel Previous Create change set Create stack

Section B: AWS S3

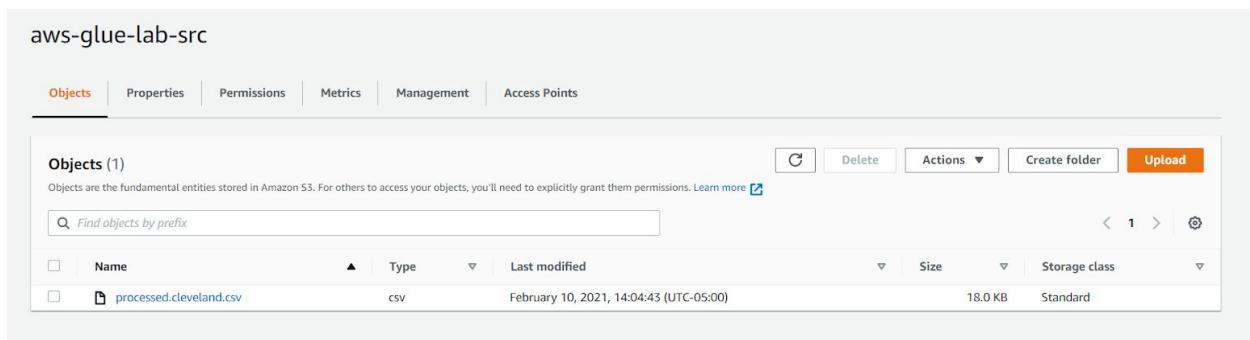
Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as data lakes, 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

<input type="radio"/>	aws-glue-lab-demo-landing	US East (N. Virginia) us-east-1	Bucket and objects not public
<input type="radio"/>	aws-glue-lab-demo-query-output	US East (N. Virginia) us-east-1	Bucket and objects not public
<input type="radio"/>	aws-glue-lab-demo-src	US East (N. Virginia) us-east-1	Bucket and objects not public
<input type="radio"/>	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



Name	Type	Last modified	Size	Storage class
processed.cleveland.csv	csv	February 10, 2021, 14:04:43 (UTC-05:00)	18.0 KB	Standard

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.

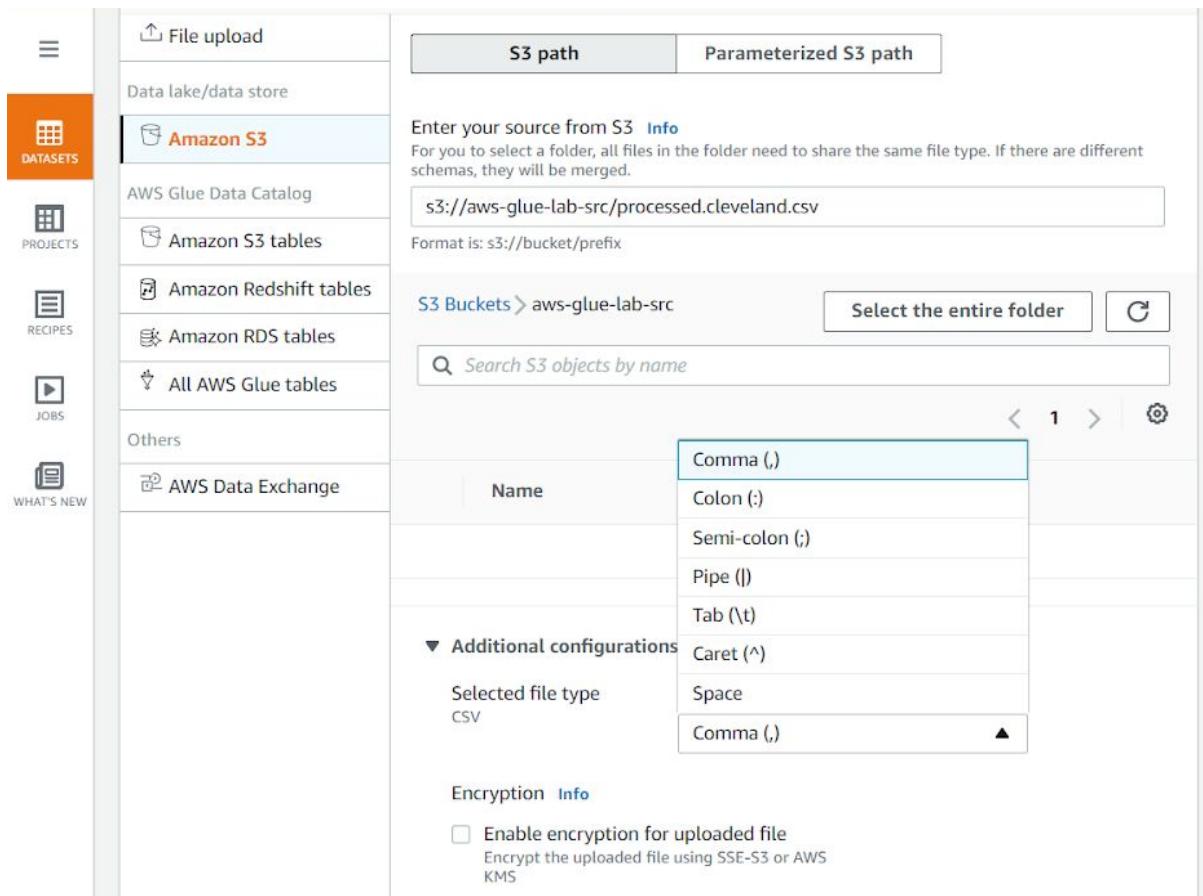
To prepare your data with DataBrew, you follow these steps:

- 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.

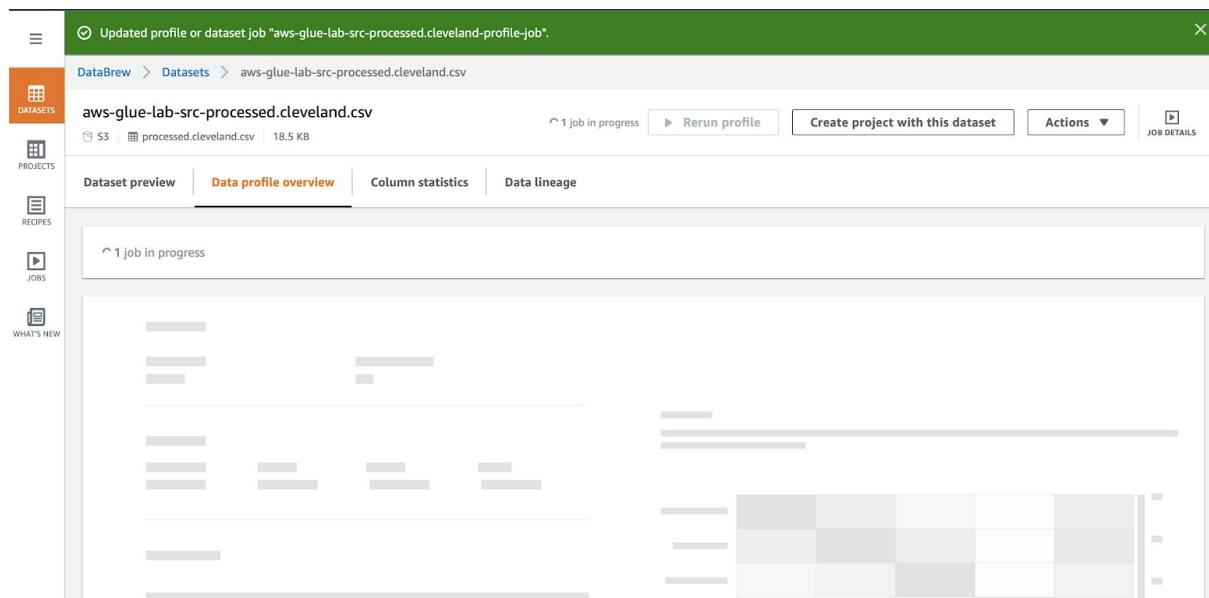
Part A: Prepare the source dataset

Step 1: Add Source Dataset in AWS Glue DataBrew



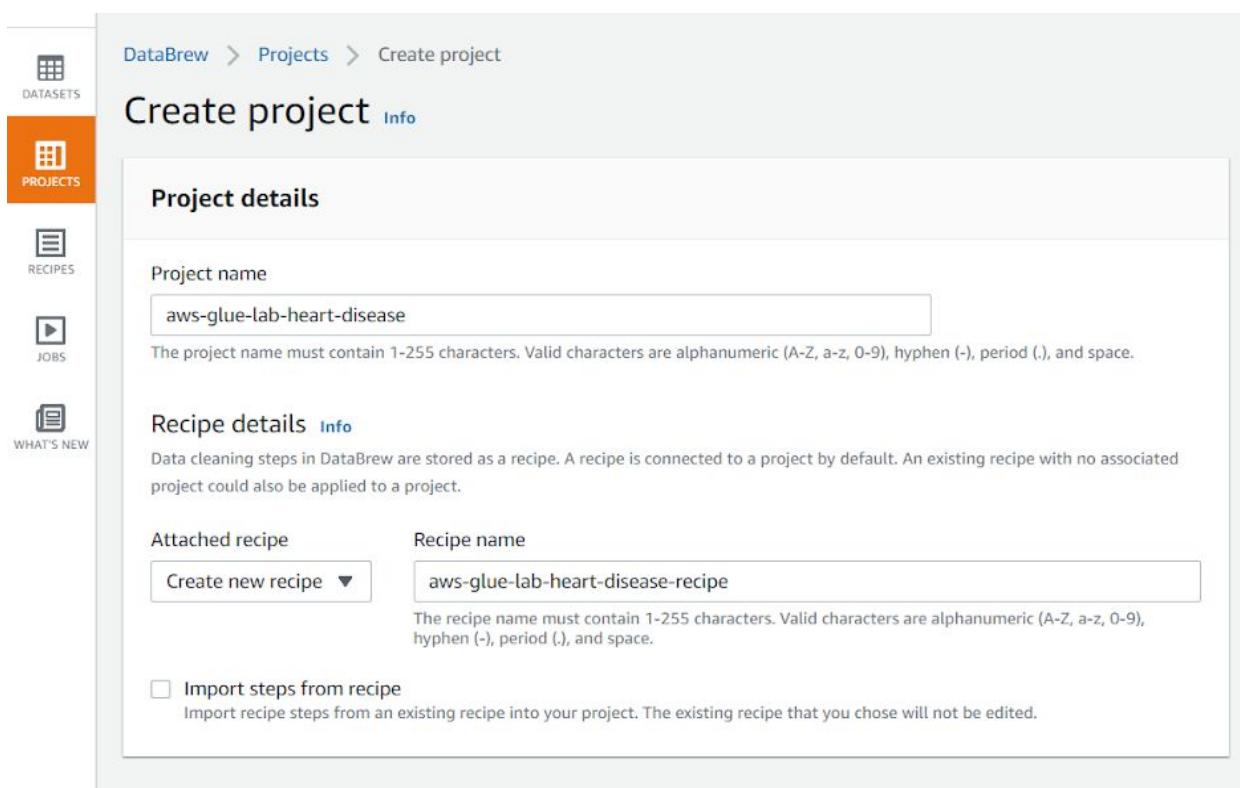
The screenshot shows the AWS Glue DataBrew interface. On the left, there's a sidebar with icons for DATASETS (highlighted in orange), PROJECTS, RECIPES, JOBS, and WHAT'S NEW. The main area is titled 'File upload' and shows a list of data sources. Under 'Data lake/data store', 'Amazon S3' is selected. The 'S3 path' field contains 's3://aws-glue-lab-src/processed.cleveland.csv'. Below it, a note says 'For you to select a folder, all files in the folder need to share the same file type. If there are different schemas, they will be merged.' A dropdown menu for 'Selected file type' is open, showing options like 'Comma (,), Colon (:), Semi-colon (;), Pipe (|), Tab (\t), Caret (^), Space', with 'Comma (,) selected. There's also an 'Additional configurations' section with an 'Encryption' option that is unchecked.

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**.



Project name
aws-glue-lab-heart-disease

The project name must contain 1-255 characters. Valid characters are alphanumeric (A-Z, a-z, 0-9), hyphen (-), period (.), and space.

Attached recipe
Create new recipe ▾

Recipe name
aws-glue-lab-heart-disease-recipe

The recipe name must contain 1-255 characters. Valid characters are alphanumeric (A-Z, a-z, 0-9), hyphen (-), period (.), and space.

Import steps from recipe
Import recipe steps from an existing recipe into your project. The existing recipe that you chose will not be edited.

3. Enter a name for your project. Then choose a recipe to attach to your project:

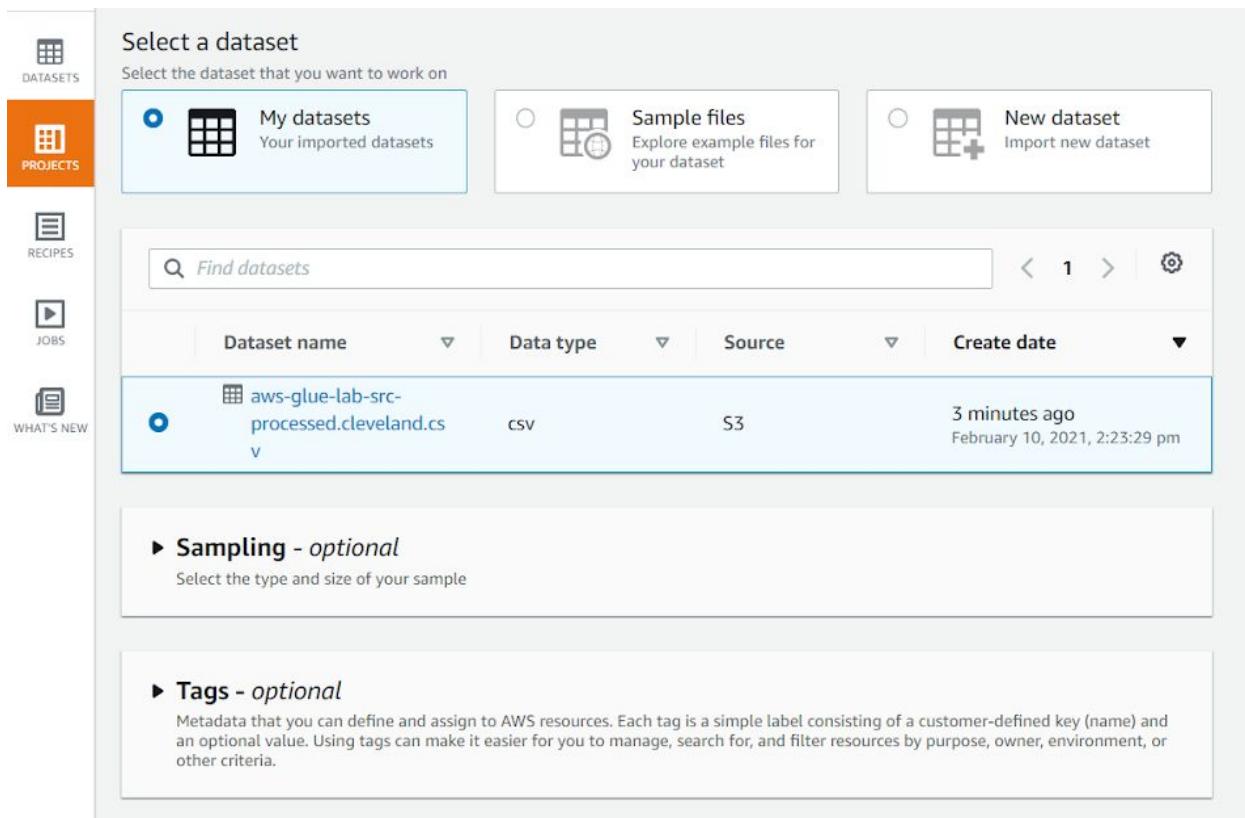
- Choose **Create a new recipe** if you are starting from the beginning. Doing this creates a new, empty recipe and attaches it to your project.
- Choose **Edit existing recipe** if you have a previously published recipe that you want to use for this project. If the recipe is currently attached to another 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.



Select a dataset

Select the dataset that you want to work on

- My datasets Your imported datasets
- Sample files Explore example files for your dataset
- New dataset Import new dataset

Find datasets

Dataset name	Data type	Source	Create date
aws-glue-lab-src-processed.cleveland.cs	csv	S3	3 minutes ago February 10, 2021, 2:23:29 pm

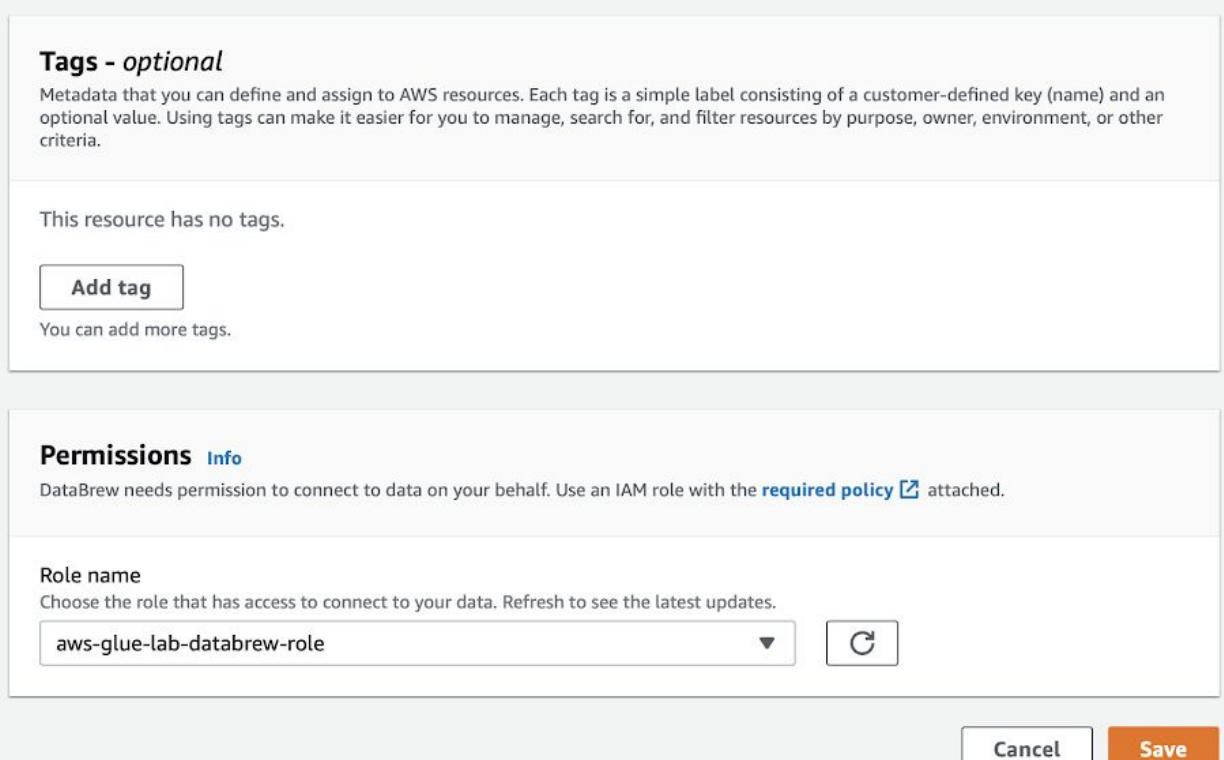
▶ Sampling - optional
Select the type and size of your sample

▶ Tags - optional
Metadata that you can define and assign to AWS resources. Each tag is a simple label consisting of a customer-defined key (name) and an optional value. Using tags can make it easier for you to manage, search for, and filter resources by purpose, owner, environment, or other criteria.

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:

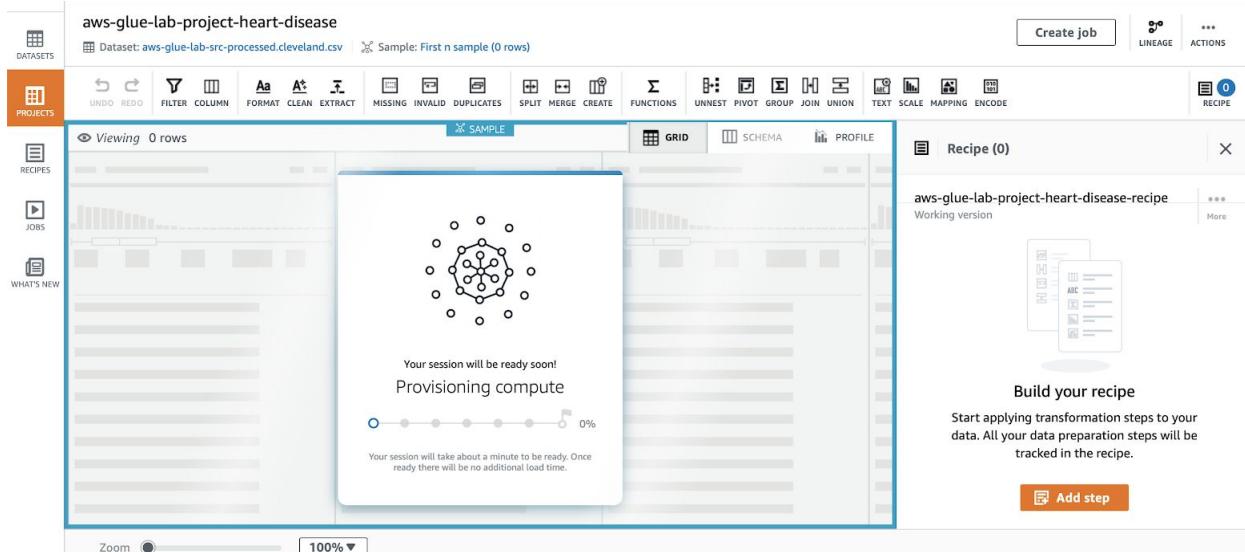
- Use **First n rows** to create a sample based on the first rows in the dataset.
- Use **Random rows** to create a sample based on a random selection of rows in the dataset.
- Choose the number of rows to appear in the sample: 500, 1,000, 2,500, or a custom sample size, up to a maximum of 5,000 rows. 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.

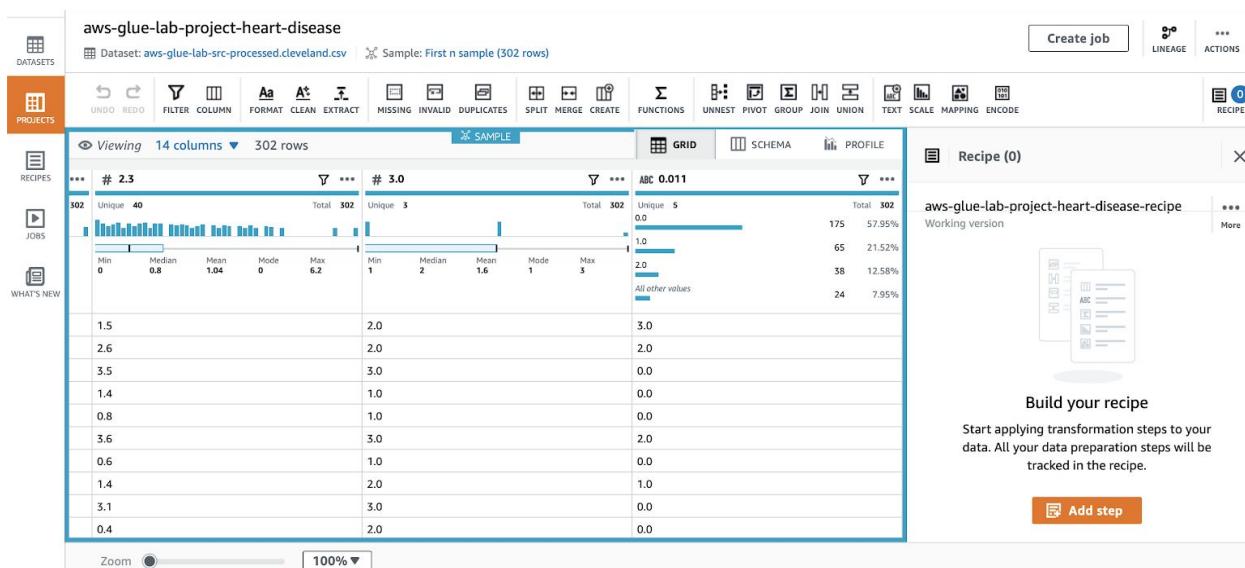
Tags are simple labels consisting of a user-defined key and an optional value that can make it easier to manage, search for, and filter DataBrew projects by purpose, owner, environment, or other criteria.

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



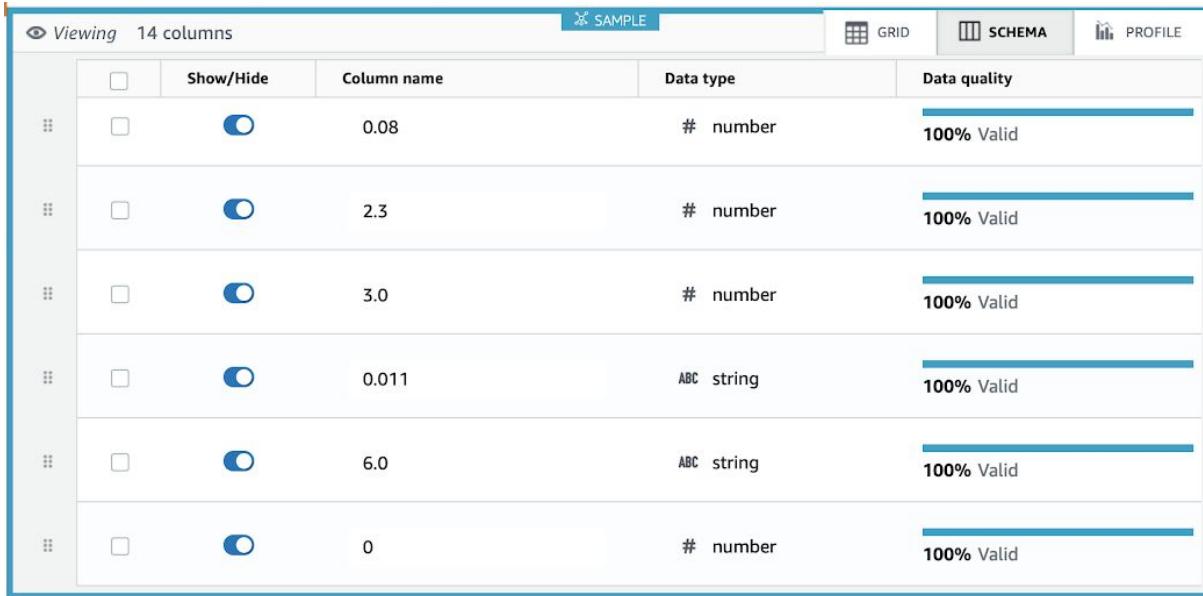
The screenshot shows the AWS Glue DataBrew interface. On the left, there's a sidebar with 'PROJECTS' selected, showing 'RECIPIES', 'JOBS', and 'WHAT'S NEW'. The main area has a title 'aws-glue-lab-project-heart-disease' and a sub-section 'Dataset: aws-glue-lab-src-processed.cleveland.csv | Sample: First n sample (0 rows)'. A large central window displays a scatter plot with the message 'Your session will be ready soon! Provisioning compute' and a progress bar at 0%. To the right, a 'Create job' button is at the top, followed by 'LINEAGE' and 'ACTIONS' buttons. Below these are sections for 'RECIPE' (empty) and 'Recipe (0)' which lists 'aws-glue-lab-project-heart-disease-recipe' as the 'Working version'. A 'Build your recipe' section with a 'Add step' button is also present.

Project is populating the data of the source dataset



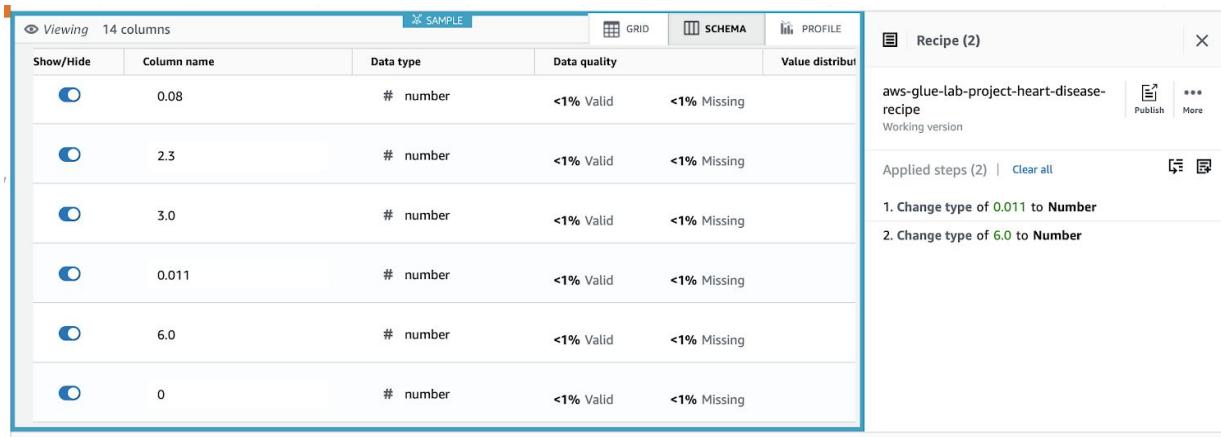
This screenshot shows the same AWS Glue DataBrew interface after the session has initialized. The central 'SAMPLE' view now displays a table with 14 columns and 302 rows. The first few rows of data are visible, showing numerical values for various attributes. The 'Create job' button is still at the top, along with 'LINEAGE' and 'ACTIONS'. The 'RECIPE' section and 'Recipe (0)' section remain the same, with the 'aws-glue-lab-project-heart-disease-recipe' listed as the 'Working version'. The 'Build your recipe' section with a 'Add step' button is also present.

Schema is being populated



	Show/Hide	Column name	Data type	Data quality
⋮	<input checked="" type="checkbox"/>	0.08	# number	100% Valid
⋮	<input checked="" type="checkbox"/>	2.3	# number	100% Valid
⋮	<input checked="" type="checkbox"/>	3.0	# number	100% Valid
⋮	<input checked="" type="checkbox"/>	0.011	ABC string	100% Valid
⋮	<input checked="" type="checkbox"/>	6.0	ABC string	100% Valid
⋮	<input checked="" type="checkbox"/>	0	# number	100% Valid

Changing the necessary column data types



Show/Hide	Column name	Data type	Data quality	Value distribution
<input checked="" type="checkbox"/>	0.08	# number	<1% Valid	<1% Missing
<input checked="" type="checkbox"/>	2.3	# number	<1% Valid	<1% Missing
<input checked="" type="checkbox"/>	3.0	# number	<1% Valid	<1% Missing
<input checked="" type="checkbox"/>	0.011	# number	<1% Valid	<1% Missing
<input checked="" type="checkbox"/>	6.0	# number	<1% Valid	<1% Missing
<input checked="" type="checkbox"/>	0	# number	<1% Valid	<1% Missing

Recipe (2)

aws-glue-lab-project-heart-disease-recipe
Working version

Applied steps (2) | Clear all

1. Change type of 0.011 to Number
2. Change type of 6.0 to Number

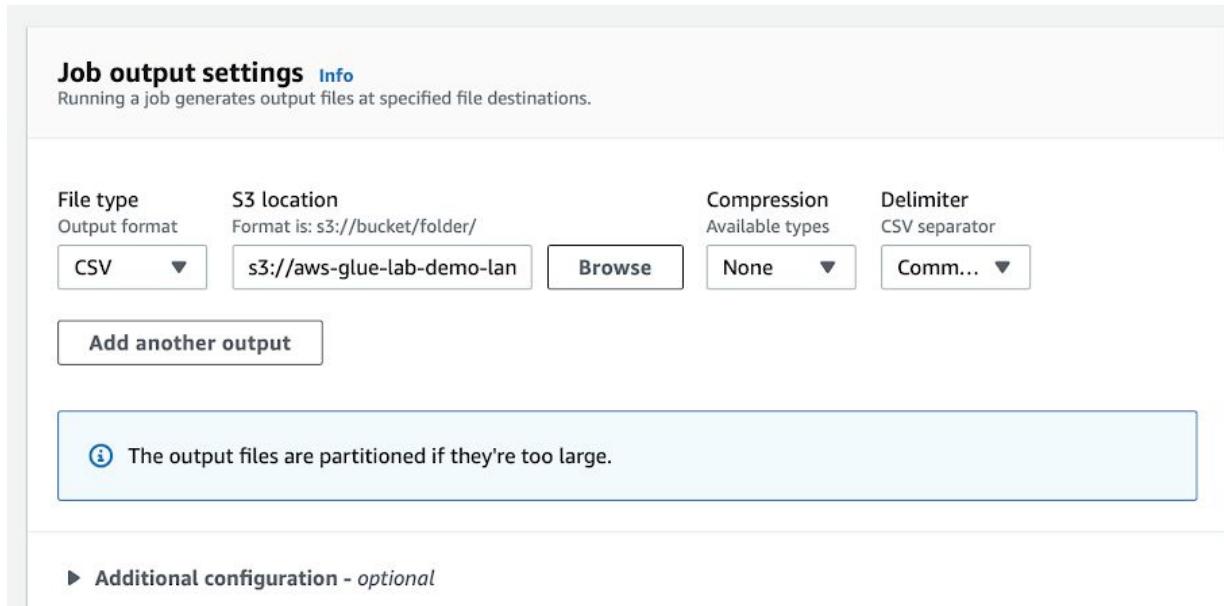
Part C: Create profile job

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.



Job output settings Info

Running a job generates output files at specified file destinations.

File type Output format	S3 location Format is: s3://bucket/folder/	Compression Available types	Delimiter CSV separator
CSV	s3://aws-glue-lab-demo-lan	Browse	None
			Comm...

Add another output

(i) The output files are partitioned if they're too large.

► Additional configuration - *optional*

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.

► Additional configuration - *optional*

► Advanced job settings - *optional* [Info](#)

Settings that control the processing and compute used for the jobs run on your project

► Associated schedules - *optional* [Info](#)

You can associate up to 2 schedules to automate your job.

► Tags - *optional*

Metadata that you can define and assign to AWS resources. Each tag is a simple label consisting of a customer-defined key (name) and an optional value. Using tags can make it easier for you to manage, search for, and filter resources by purpose, owner, environment, or other criteria.

Permissions [Info](#)

DataBrew needs permission to connect to data on your behalf. Use an IAM role with the **required policy**  attached.

Role name

Choose the role that has access to connect to your data. Refresh to see the latest updates.

AWSPractice-Developer



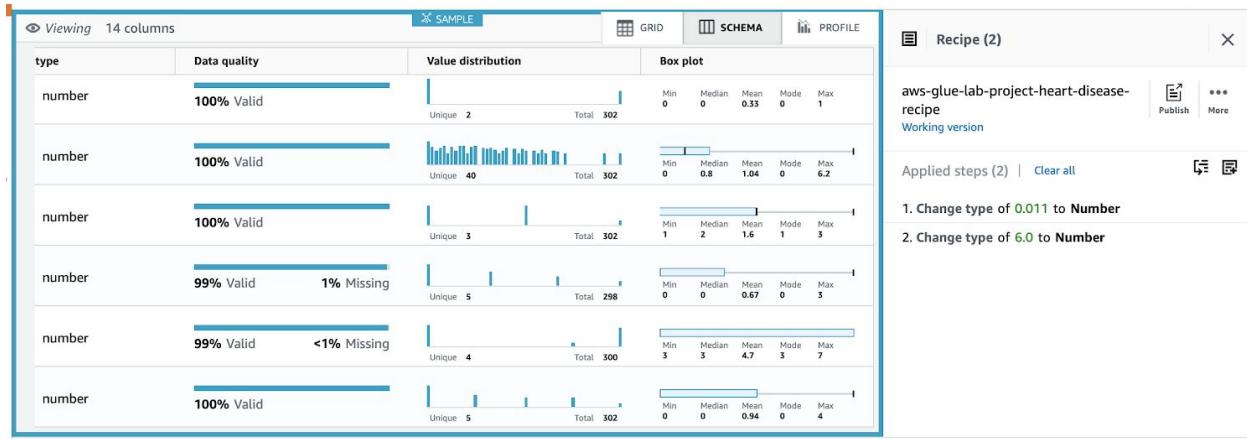
8. On Advanced job settings pane, you can choose more options for how your job is to run:
 - Maximum number of units – DataBrew processes jobs using multiple compute nodes, running in parallel. The default number of nodes is 5. The maximum number of nodes is 149.
 - Job timeout – If a job takes more 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

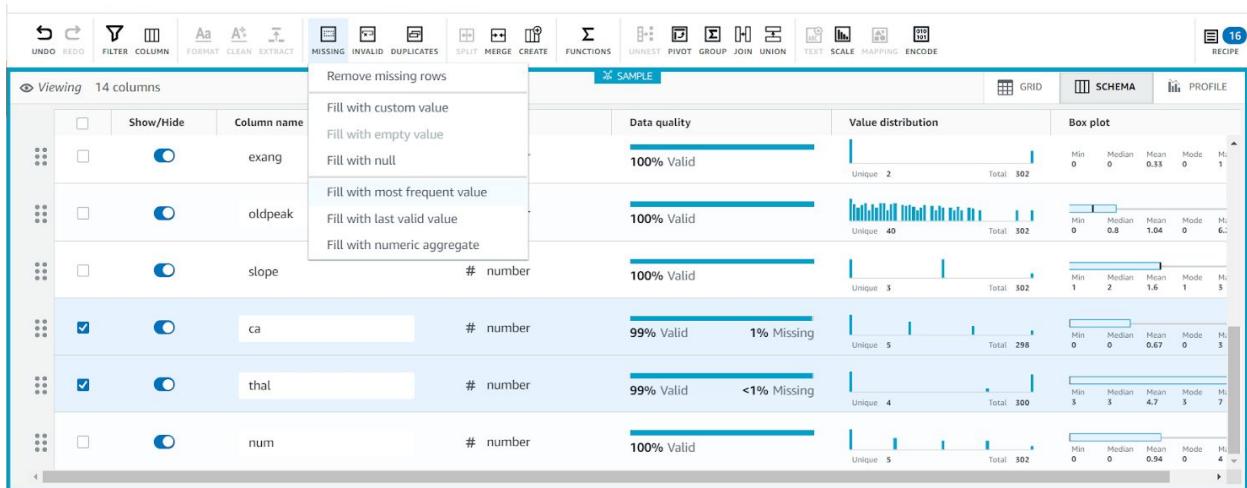
Part D: Data Validation

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

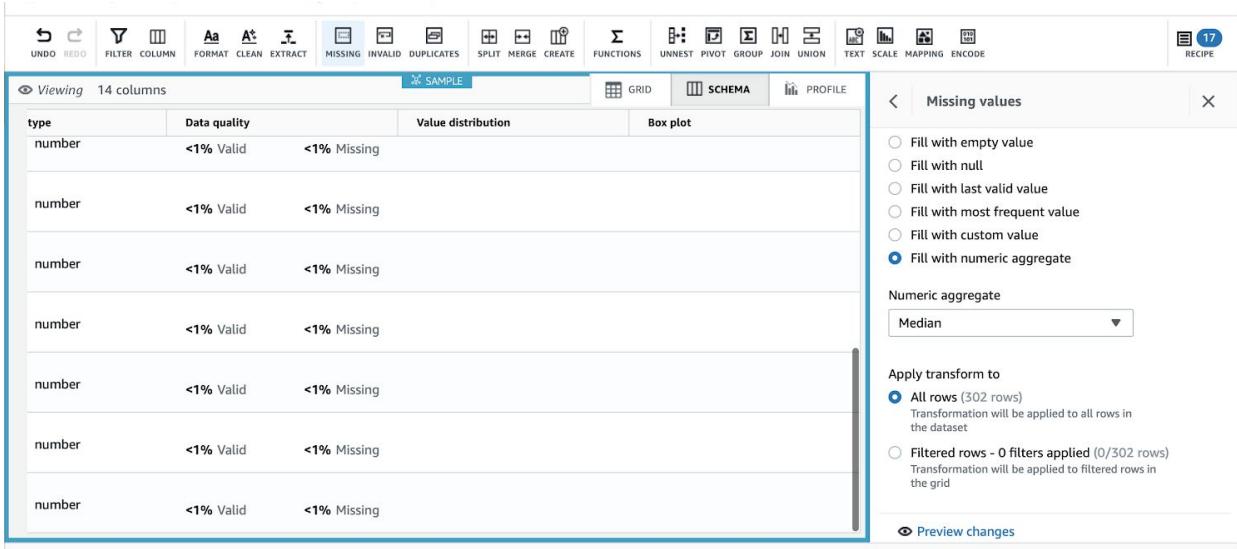


Step 1: Change the column names of schema

The changes which will be added to the recipe.

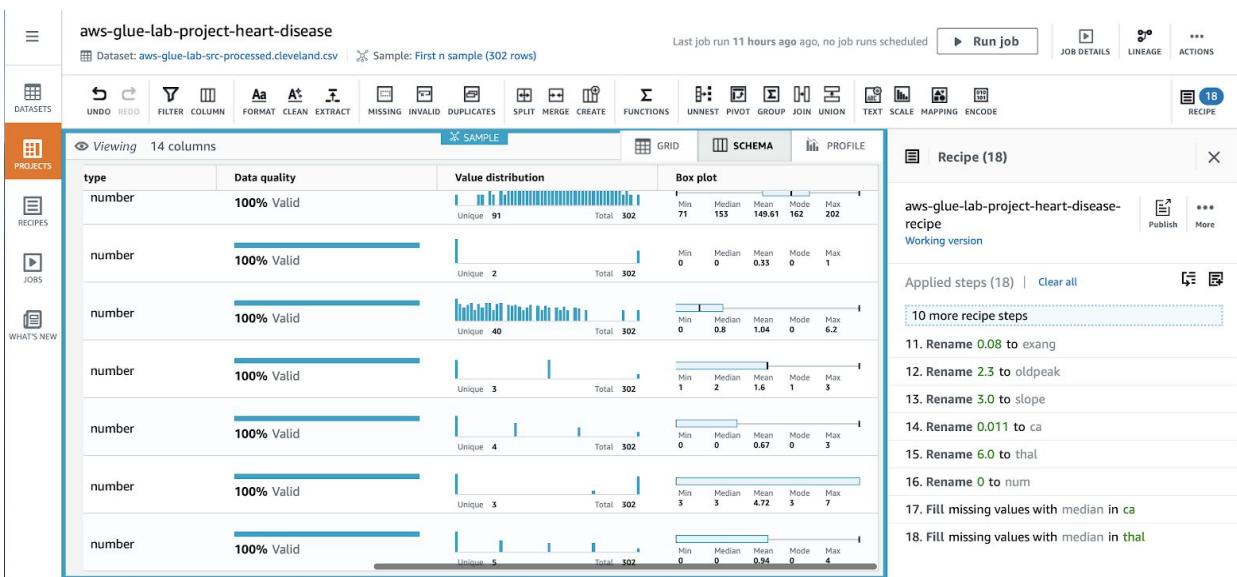


Step 2: Fill the missing values with median using the box plot graph



The screenshot shows the Quantiphi Data Profiler interface. The main area displays a table with 14 columns, mostly of type 'number'. The 'Data quality' column indicates '<1% Valid' and '<1% Missing' for most rows. The 'Value distribution' and 'Box plot' sections provide statistical details for each column. On the right, a sidebar titled 'Missing values' offers various fill options like 'Fill with empty value' or 'Fill with numeric aggregate' (set to 'Median'). Below this, 'Apply transform to' options are shown for 'All rows' or 'Filtered rows - 0 filters applied'.

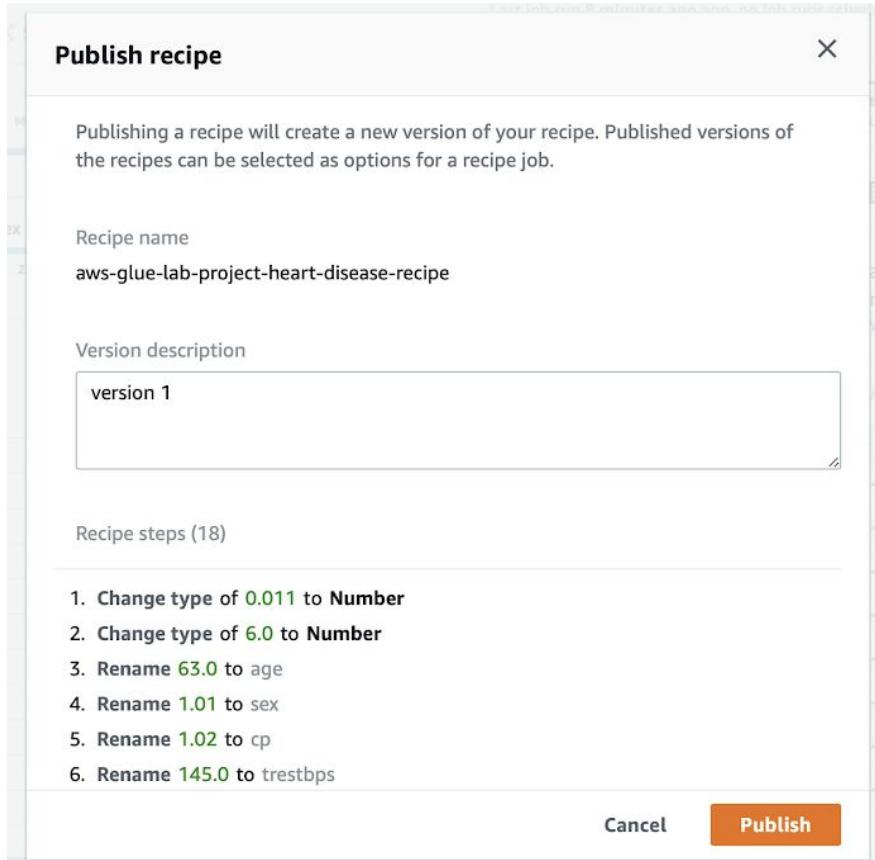
All the steps will be added to recipe



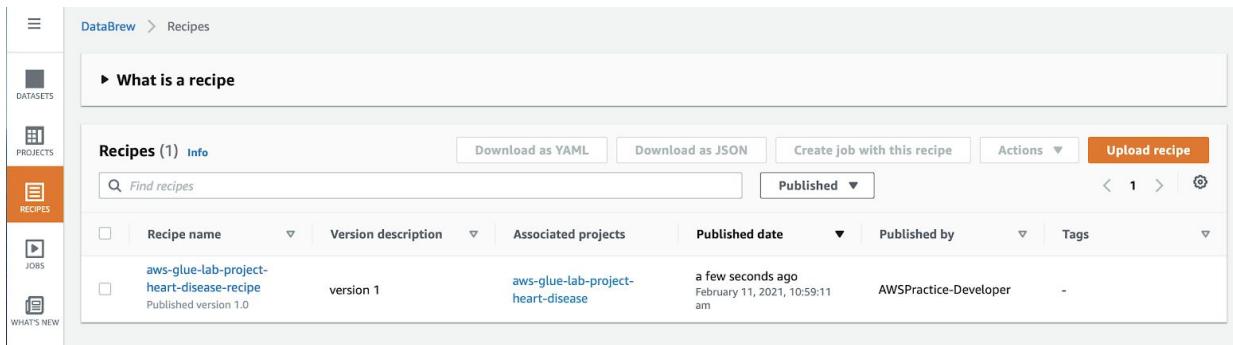
The screenshot shows the Quantiphi Data Profiler interface with the 'PROJECTS' sidebar selected. The main area displays the 'aws-glue-lab-project-heart-disease' dataset, where all columns are now marked as '100% Valid'. The 'Value distribution' and 'Box plot' sections are updated to reflect this high validity. On the right, the 'Recipe (18)' section lists the following steps:

- 10 more recipe steps
11. Rename 0.08 to exang
12. Rename 2.3 to oldpeak
13. Rename 3.0 to slope
14. Rename 0.011 to ca
15. Rename 6.0 to thal
16. Rename 0 to num
17. Fill missing values with median in ca
18. Fill missing values with median in thal

Step 3: Publish / save the recipe for future use



Step 4: View the recipe



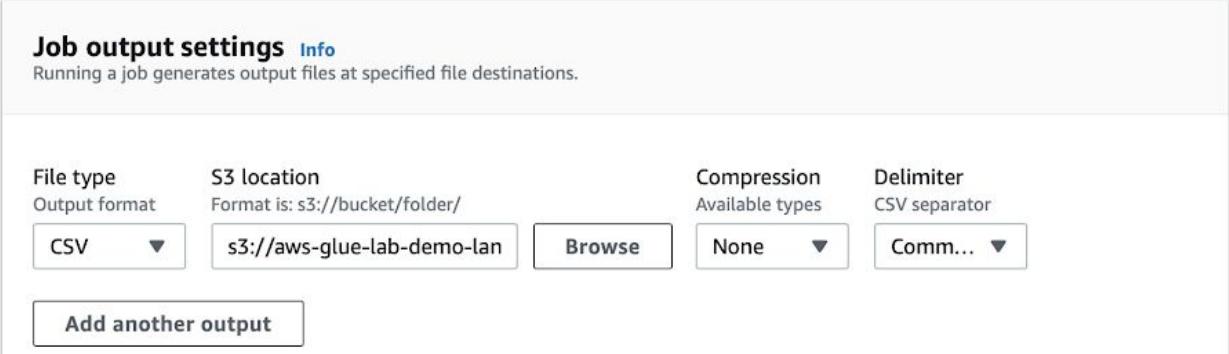
DataBrew > Recipes

► What is a recipe

Recipes (1) Info

Actions	Upload recipe				
Find recipes	Published				
Recipe name	Version description	Associated projects	Published date	Published by	Tags
aws-glue-lab-project-heart-disease-recipe Published version 1.0	version 1	aws-glue-lab-project-heart-disease	a few seconds ago February 11, 2021, 10:59:11 am	AWSPractice-Developer	-

Step 5: Save the output location of job run



The screenshot shows the 'Job output settings' section of the AWS Glue Studio interface. It includes fields for 'File type' (set to 'CSV'), 'S3 location' (set to 's3://aws-glue-lab-demo-lan'), 'Compression' (set to 'None'), and 'Delimiter' (set to 'Comma'). There is also a 'Browse' button for S3 location and an 'Add another output' button.

Section D: AWS Glue Studio

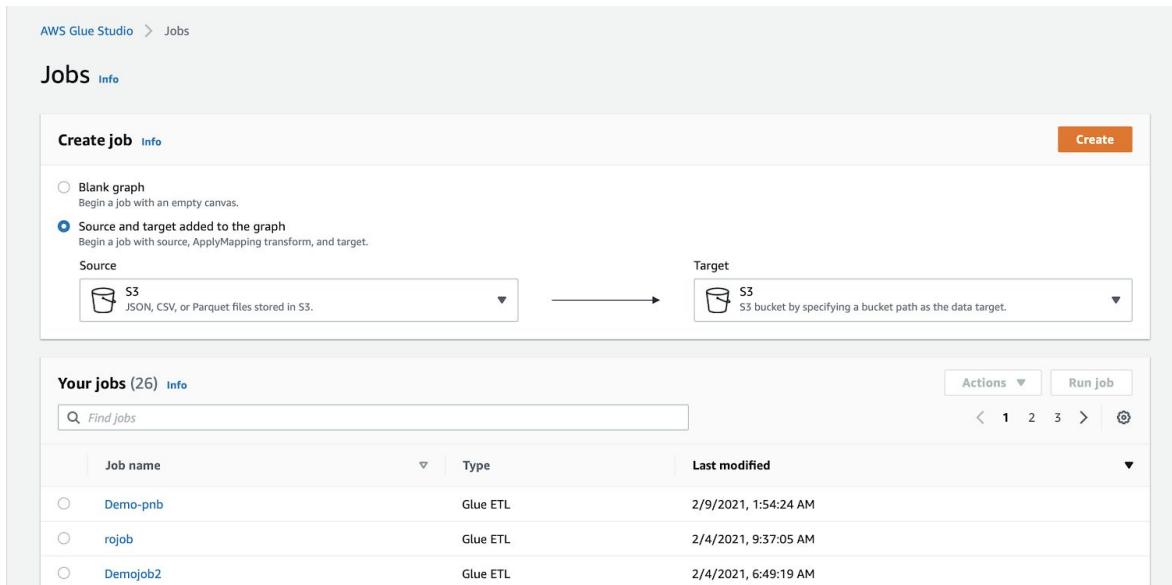
AWS Glue Studio is an easy-to-use graphical interface that speeds up the process of authoring, running, and monitoring extract, transform, and load (ETL) jobs in AWS Glue. The visual interface allows those who don't know Apache Spark to design jobs without coding experience and accelerates the process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

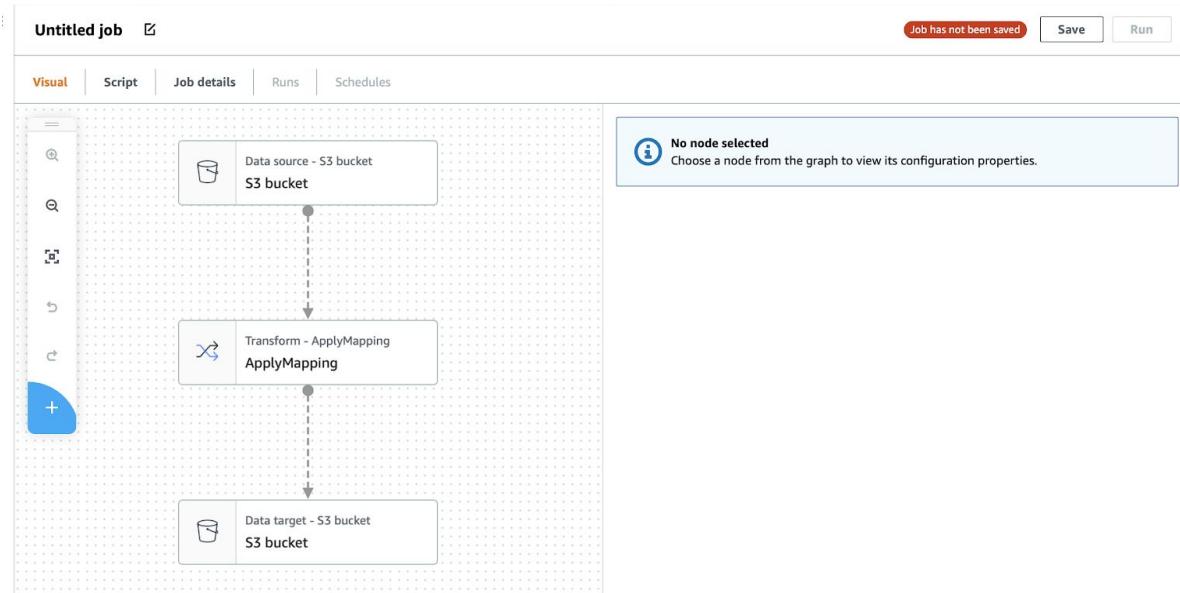
You can use AWS Glue Studio to create jobs that extract structured or semi-structured data from a data source, perform a transformation of that data, and save the result set in a data target. Here we are going to do same process as AWS Glue databrew or we can extract from one location and load to another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

Step 1: Start the job creation process



The screenshot shows the AWS Glue Studio interface. At the top, it says "AWS Glue Studio > Jobs". Below that, there's a "Jobs" section with a "Create job" button. Under "Create job", there are two options: "Blank graph" (radio button) and "Source and target added to the graph" (radio button, which is selected). The "Source" field is set to "S3" (JSON, CSV, or Parquet files stored in S3). An arrow points to the "Target" field, which is also set to "S3" (S3 bucket by specifying a bucket path as the data target). Below this, there's a "Your jobs (26)" section with a table showing three rows of existing jobs:

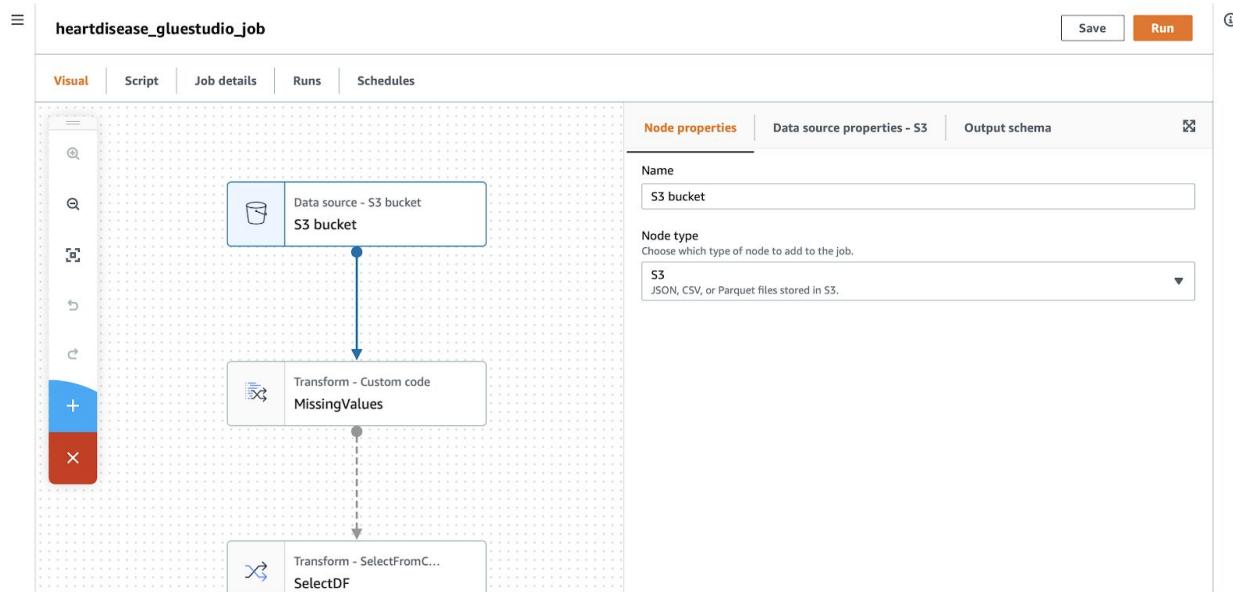
Job name	Type	Last modified
Demo-pnb	Glue ETL	2/9/2021, 1:54:24 AM
rojob	Glue ETL	2/4/2021, 9:37:05 AM
Demojob2	Glue ETL	2/4/2021, 6:49:19 AM

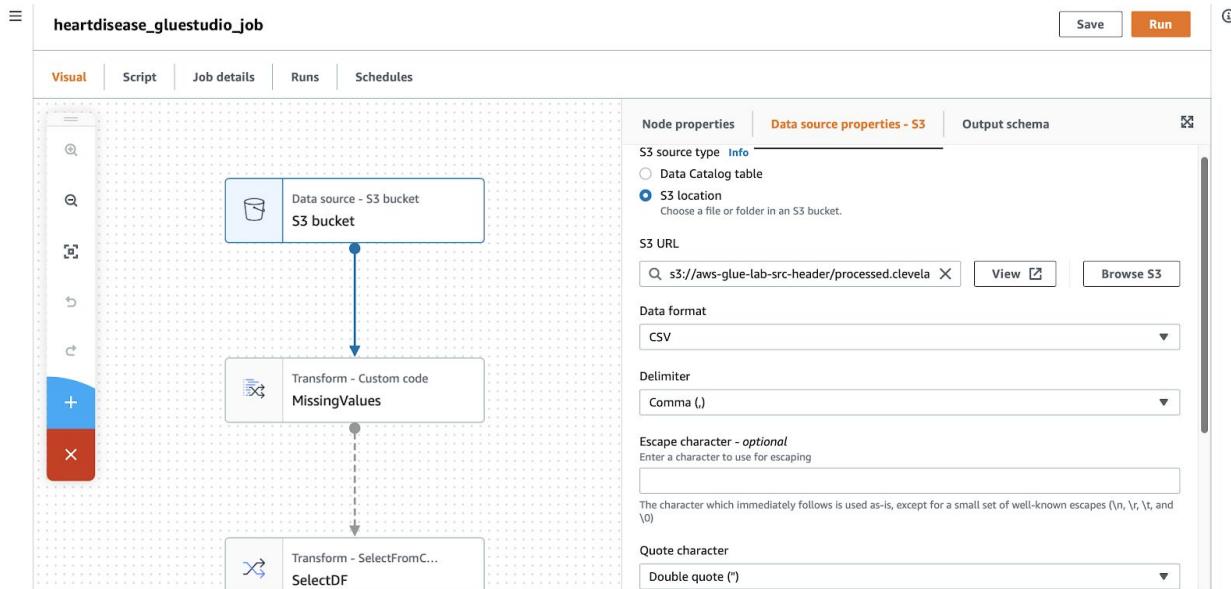


Remove the apply mapping transformation by deleting it

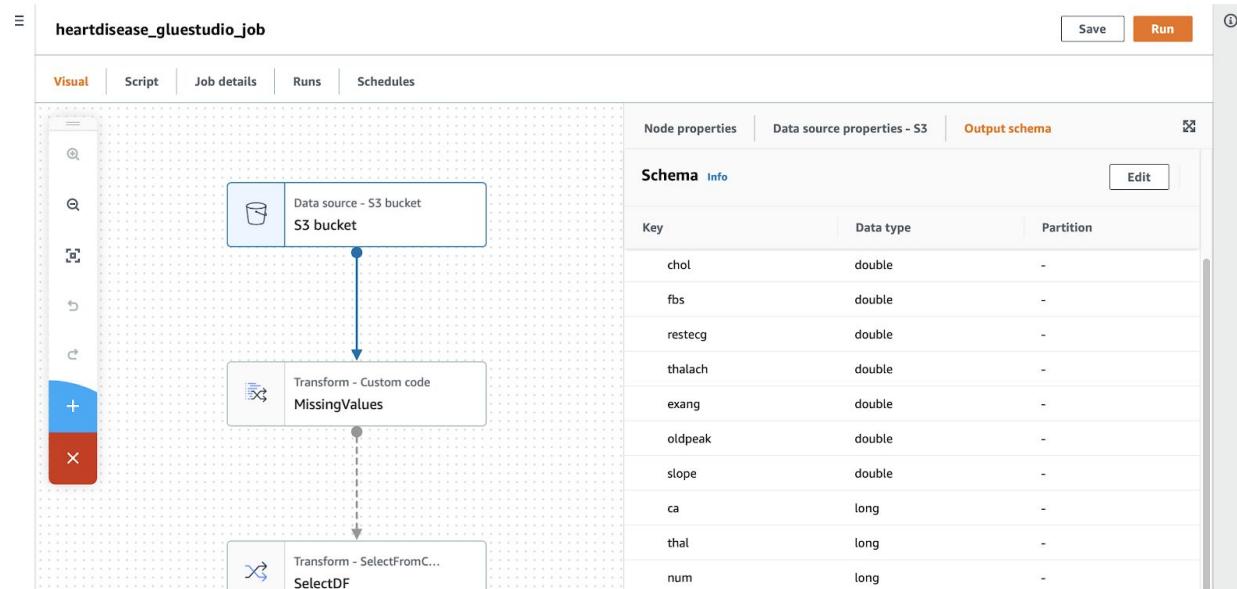
Step 2: Edit the data source node in the job graph

Rename the job name



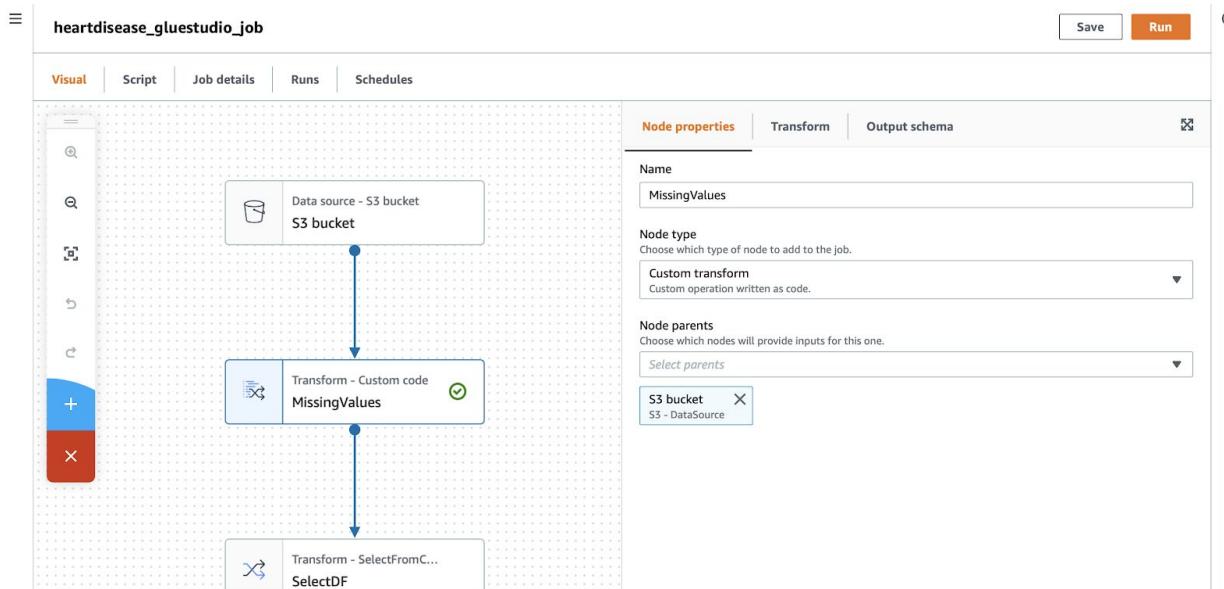


Edit the schema if data types are different



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 regex

  #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', regex('ca', '\?', modeCa))

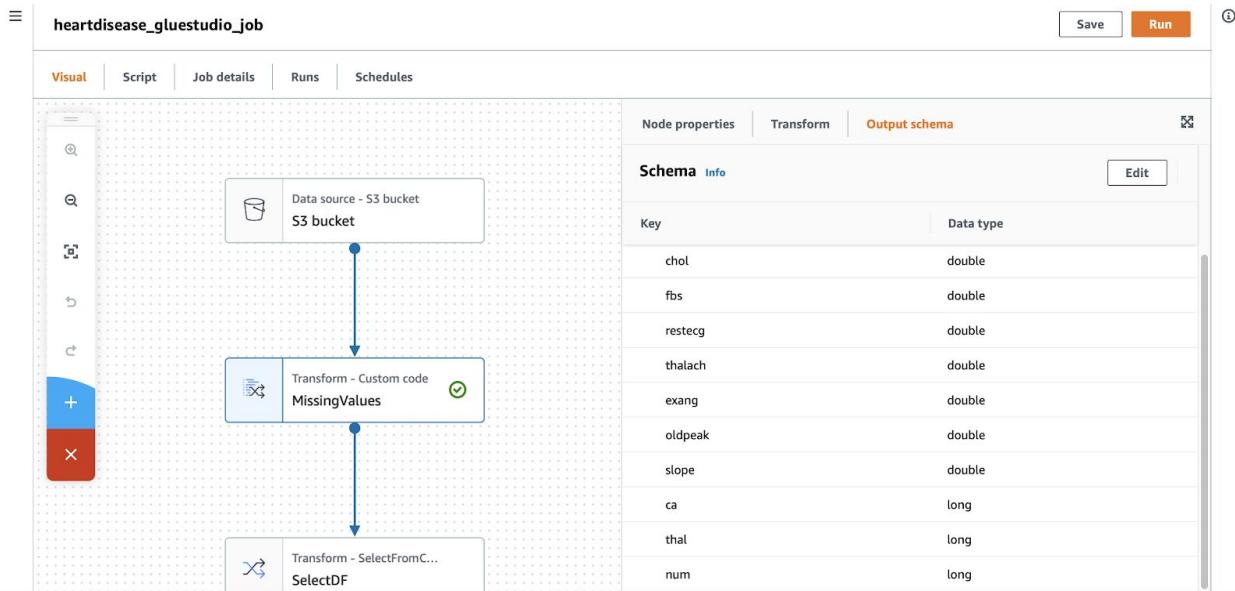
#find the mode for missing values for "thal" column using pyspark
sql
modeThal = newDF.groupby("thal").count().orderBy("count",
ascending=False).first()[0]

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

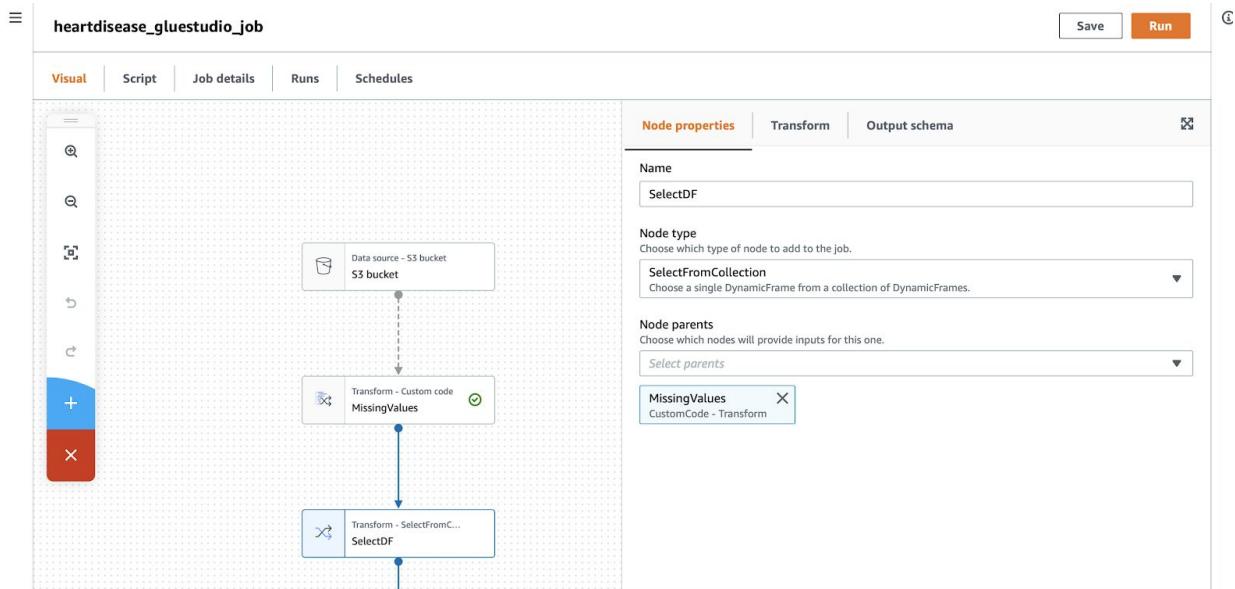
#Store the dataframe back to dynamicframe
results = DynamicFrame.fromDF(newDF, glueContext, "results")

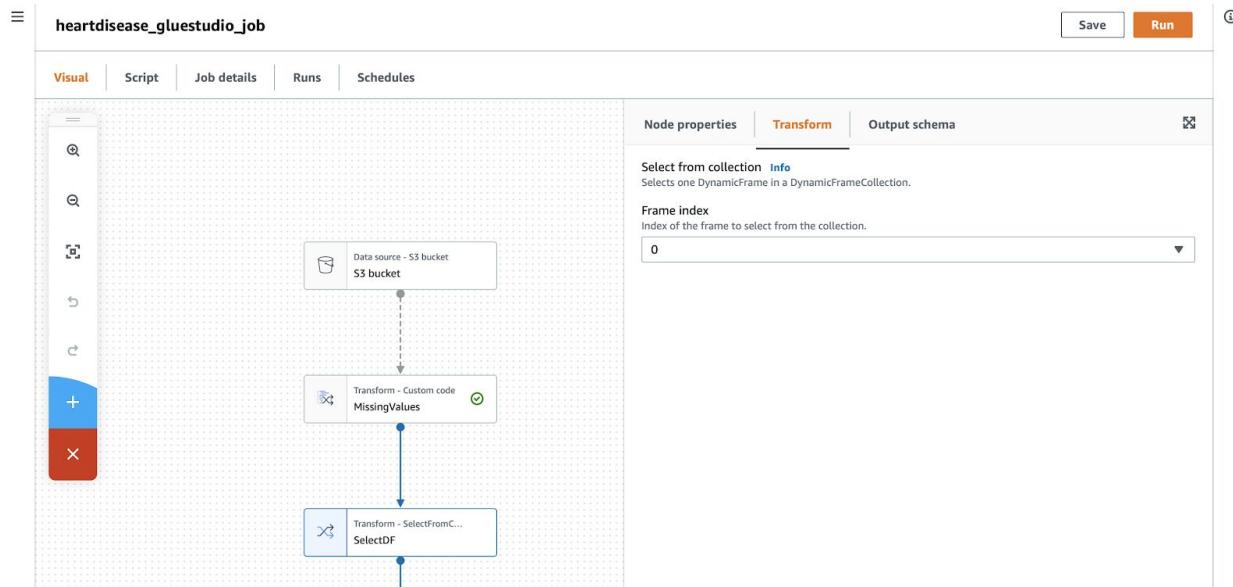
#return the dynamicframe as dynamicframecollection object
return DynamicFrameCollection({"results": results}, glueContext)
```

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.

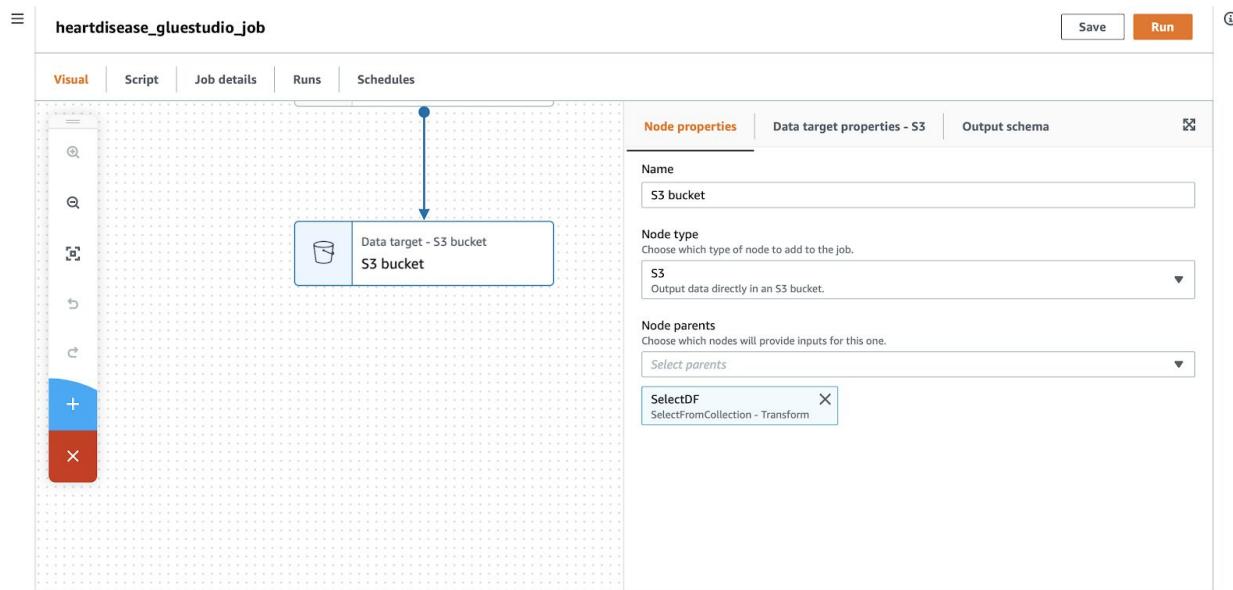


This creates automatically another transformation known as `SelectFromCollection` which can be used to select a dynamic frame from a dynamic frame collection.

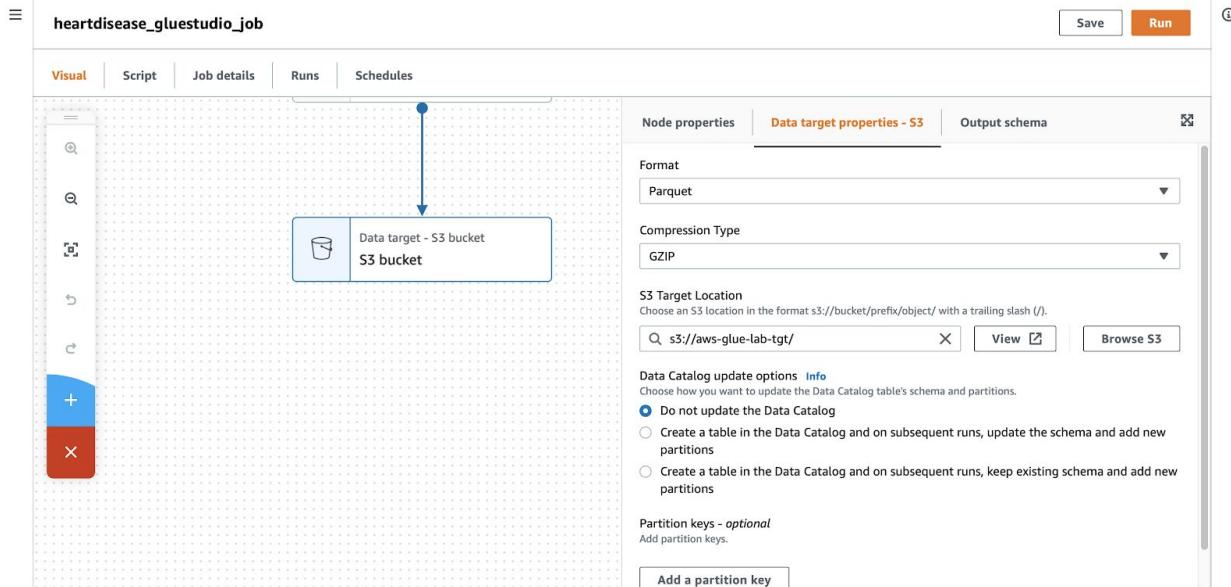




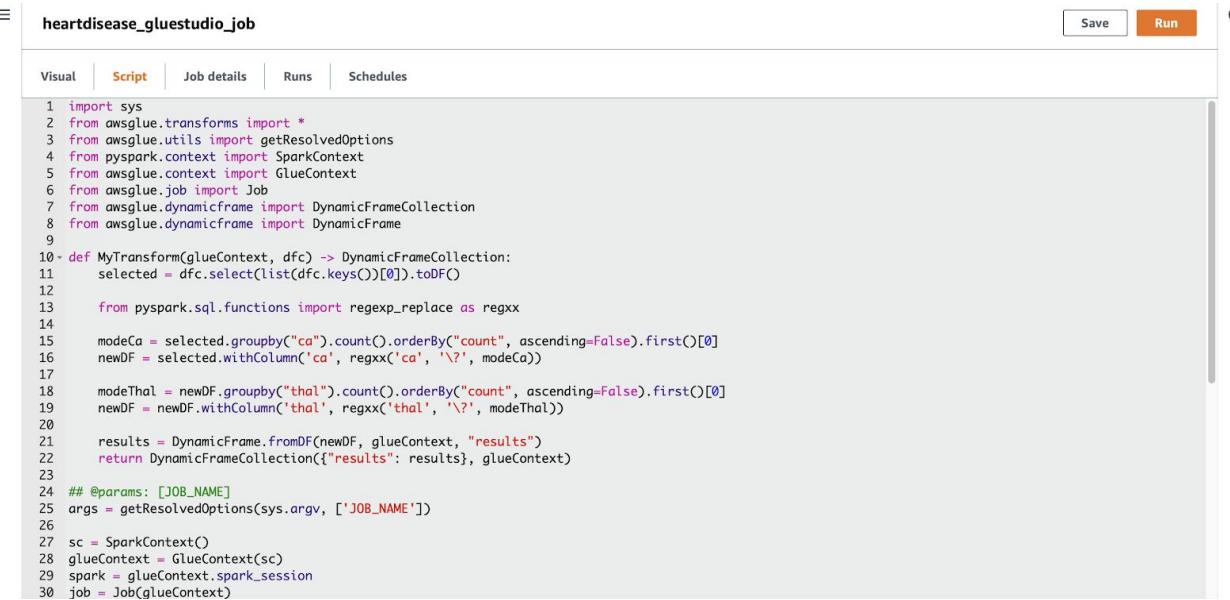
Step 4: Edit the data target node of the job



Specify the output format and compression type



Step 5: View the final job script



```

1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7 from awsglue.dynamicframe import DynamicFrameCollection
8 from awsglue.dynamicframe import DynamicFrame
9
10 def MyTransform(glueContext, dfc) -> DynamicFrameCollection:
11     selected = dfc.select(list(dfc.keys())[0]).toDF()
12
13     from pyspark.sql.functions import regexp_replace as regex
14
15     modeCa = selected.groupby("ca").count().orderBy("count", ascending=False).first()[0]
16     newDF = selected.withColumn("ca", regex("ca", '\?'), modeCa)
17
18     modeThal = newDF.groupby("thal").count().orderBy("count", ascending=False).first()[0]
19     newDF = newDF.withColumn("thal", regex("thal", '\?'), modeThal)
20
21     results = DynamicFrame.fromDF(newDF, glueContext, "results")
22     return DynamicFrameCollection({"results": results}, glueContext)
23
24 ## @params: [JOB_NAME]
25 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
26
27 sc = SparkContext()
28 glueContext = GlueContext(sc)
29 spark = glueContext.spark_session
30 job = Job(glueContext)
  
```

Step 6: Specify the job details and save the job

heartdisease_gluestudio_job

Save Run ⌂

Visual | Script | **Job details** | Runs | Schedules

Basic properties Info

Name
heartdisease_gluestudio_job

Description - optional

Descriptions can be up to 2048 characters long.

IAM Role
Role assumed by the job with permission to access your data stores. Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job.
AWSPractice-Developer
No description available.

Type
The type of ETL job. This is set automatically based on the types of data sources you have selected.
Spark

Glue version Info
Glue 2.0 - Supports spark 2.4, Scala 2, Python 3

heartdisease_gluestudio_job

Save Run ⌂

Visual | Script | **Job details** | Runs | Schedules

Language
Python 3

Worker type
Set the type of predefined worker that is allowed when a job runs.
G.1X

Number of workers
The number of workers of a defined workerType that are allocated when a job runs. The maximum number of workers you can define are 299 for G.1X, and 149 for G.2X.
10

Job bookmark Info
Specifies how AWS Glue processes job bookmark when the job runs. It can remember previously processed data (Enable), update state information (Pause), or ignore state information (Disable).
Enable

Number of retries
1

Job timeout (minutes)
Set the execution time. The default is 2,880 minutes (48 hours).
2880

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.

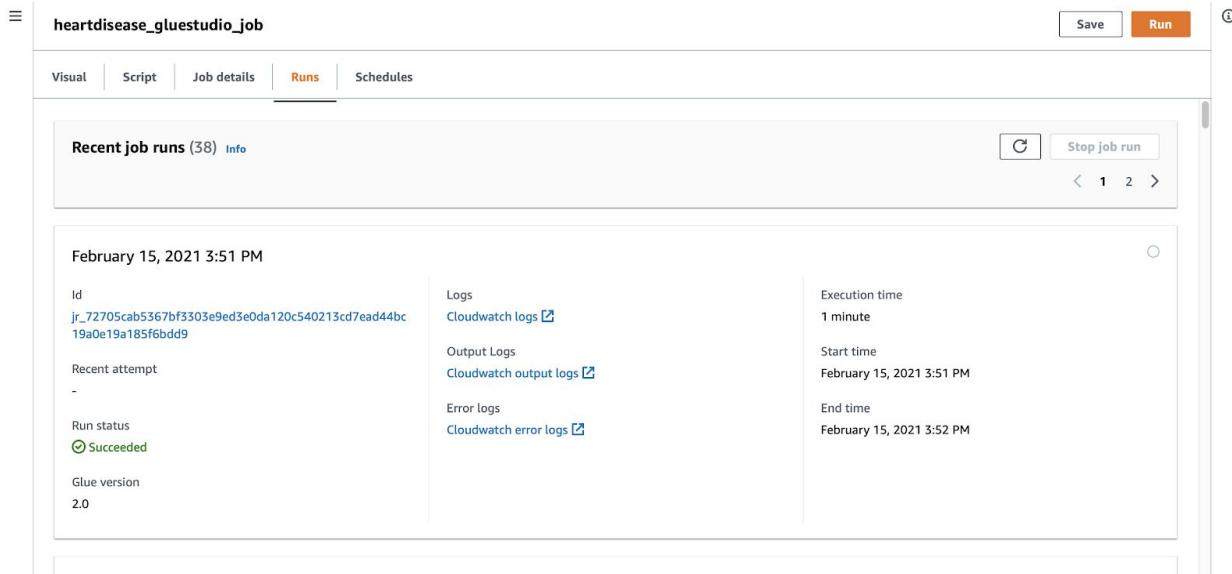
You can choose either the link in the notification for **Run Details**, or choose the **Run Details** tab to view the run status of the job.

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.



Recent job runs (38) [Info](#)

February 15, 2021 3:51 PM

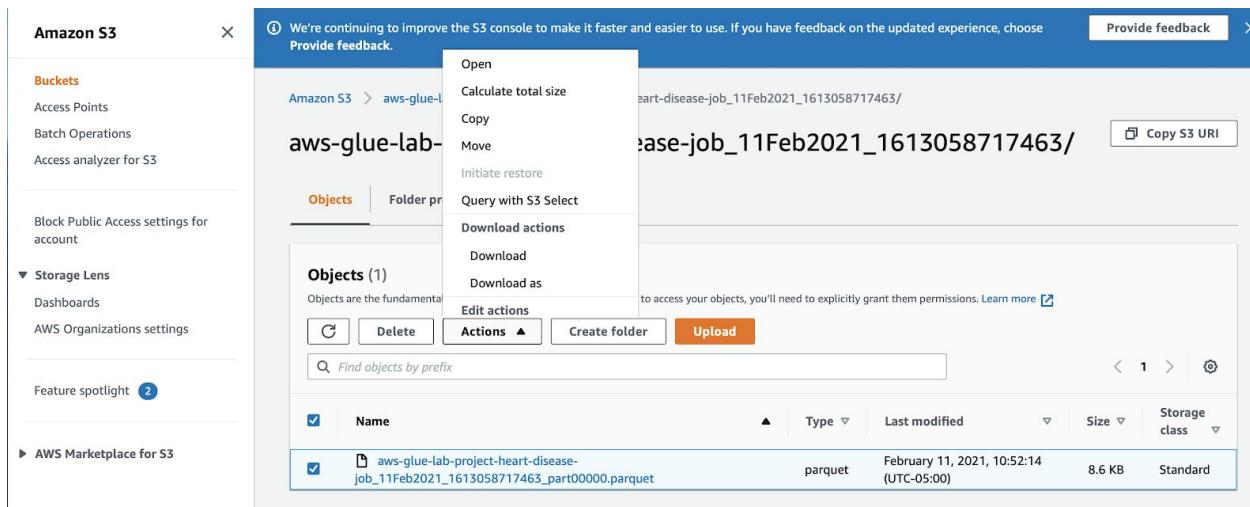
Id jr_72705cab5367bf3303e9ed3e0da120c540213cd7ead44bc 19a0e19a185f6bdd9	Logs Cloudwatch logs	Execution time 1 minute
Recent attempt -	Output Logs Cloudwatch output logs	Start time February 15, 2021 3:51 PM
Run status Succeeded	Error logs Cloudwatch error logs	End time February 15, 2021 3:52 PM
Glue version 2.0		

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

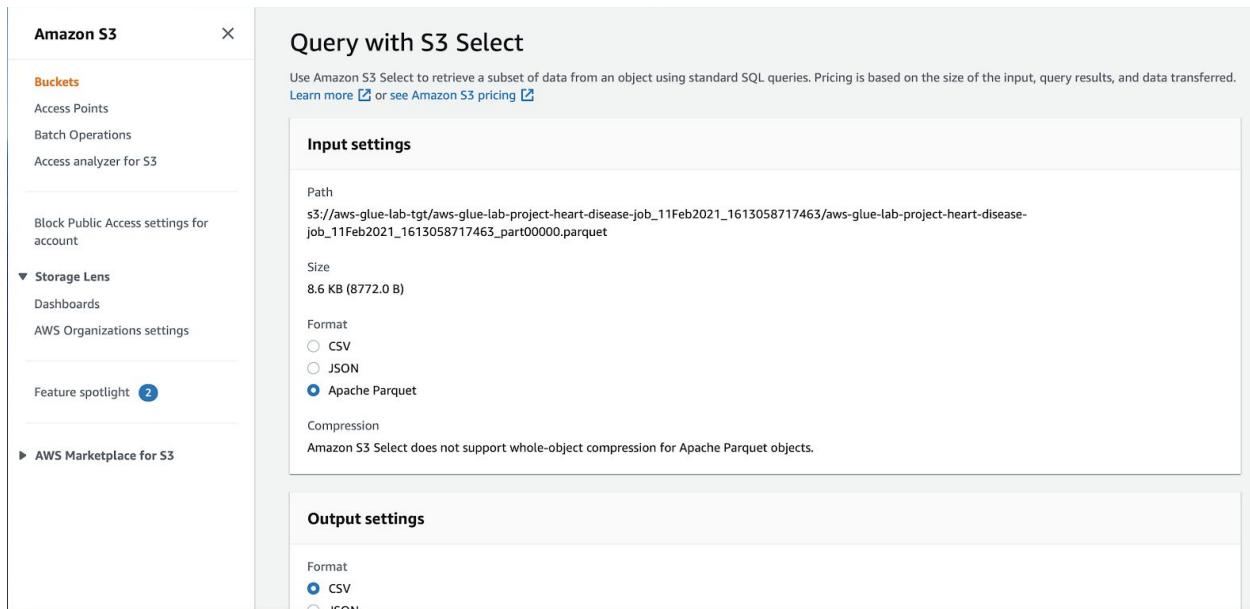


The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with navigation links such as Buckets, Storage Lens, and AWS Marketplace for S3. The main area displays the contents of the 'aws-glue-lab' bucket. A context menu is open over a specific file entry:

- Open
- Calculate total size
- Copy
- Move
- Initiate restore
- Query with S3 Select**
- Download actions
- Download
- Download as

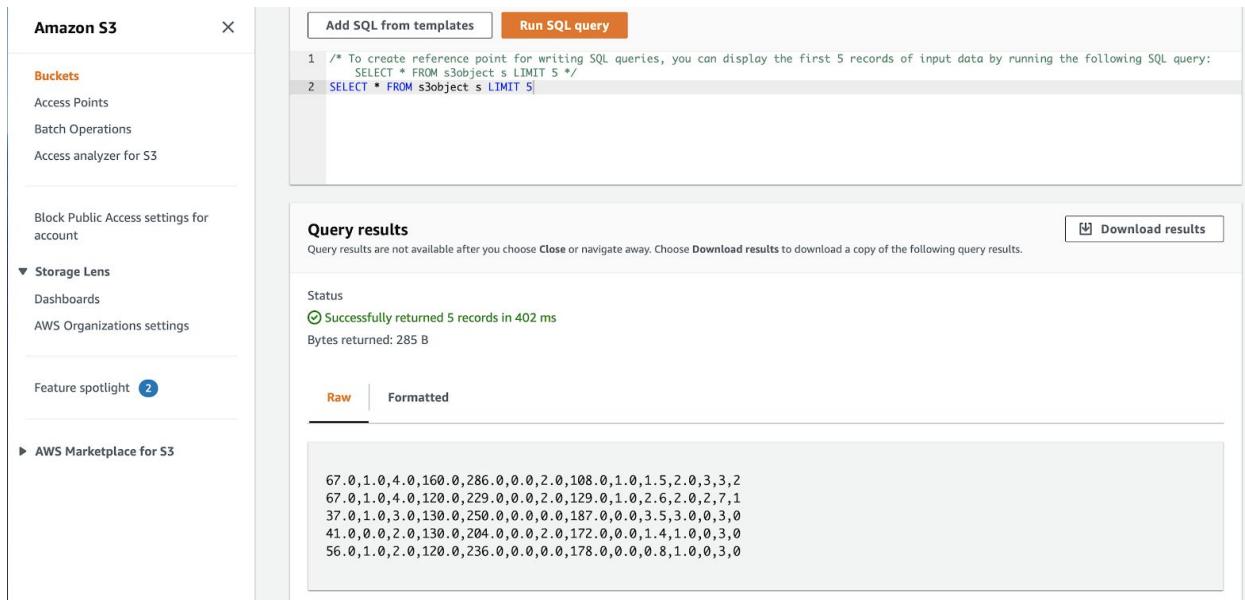
The file listed in the main table is 'aws-glue-lab-project-heart-disease-job_11Feb2021_1613058717463_part00000.parquet', which is a parquet file from February 11, 2021, at 8.6 KB in standard storage.

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



The screenshot shows the 'Query with S3 Select' dialog box. The left sidebar is identical to the one in the previous screenshot. The main dialog has two sections:

- Input settings**: Path is set to 's3://aws-glue-lab-tgt/aws-glue-lab-project-heart-disease-job_11Feb2021_1613058717463/aws-glue-lab-project-heart-disease-job_11Feb2021_1613058717463_part00000.parquet'. The file format is set to 'Apache Parquet'.
- Output settings**: The format is set to 'CSV'.



The screenshot shows the AWS Athena console interface. On the left, there's a sidebar with navigation links like 'Amazon S3', 'Buckets', 'Access Points', 'Batch Operations', 'Access analyzer for S3', 'Block Public Access settings for account', 'Storage Lens', 'Dashboards', 'AWS Organizations settings', 'Feature spotlight', and 'AWS Marketplace for S3'. The main area has tabs 'Add SQL from templates' and 'Run SQL query'. A SQL query is pasted into the editor:

```

1 /* To create reference point for writing SQL queries, you can display the first 5 records of input data by running the following SQL query:
   SELECT * FROM s3object s LIMIT 5 */
2 SELECT * FROM s3object s LIMIT 5
  
```

Below the editor, under 'Query results', it says 'Successfully returned 5 records in 402 ms' and 'Bytes returned: 285 B'. There are 'Raw' and 'Formatted' tabs, and a 'Download results' button.

```

67.0,1.0,4.0,160.0,286.0,0.0,2.0,108.0,1.0,1.5,2.0,3,3,2
67.0,1.0,4.0,120.0,229.0,0.0,2.0,129.0,1.0,2.6,2.0,2,7,1
37.0,1.0,3.0,130.0,250.0,0.0,0.0,187.0,0.0,3.5,3.0,0,3,0
41.0,0.0,2.0,130.0,204.0,0.0,2.0,172.0,0.0,1.4,1.0,0,3,0
56.0,1.0,2.0,120.0,236.0,0.0,0.0,178.0,0.0,0.8,1.0,0,3,0
  
```

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.

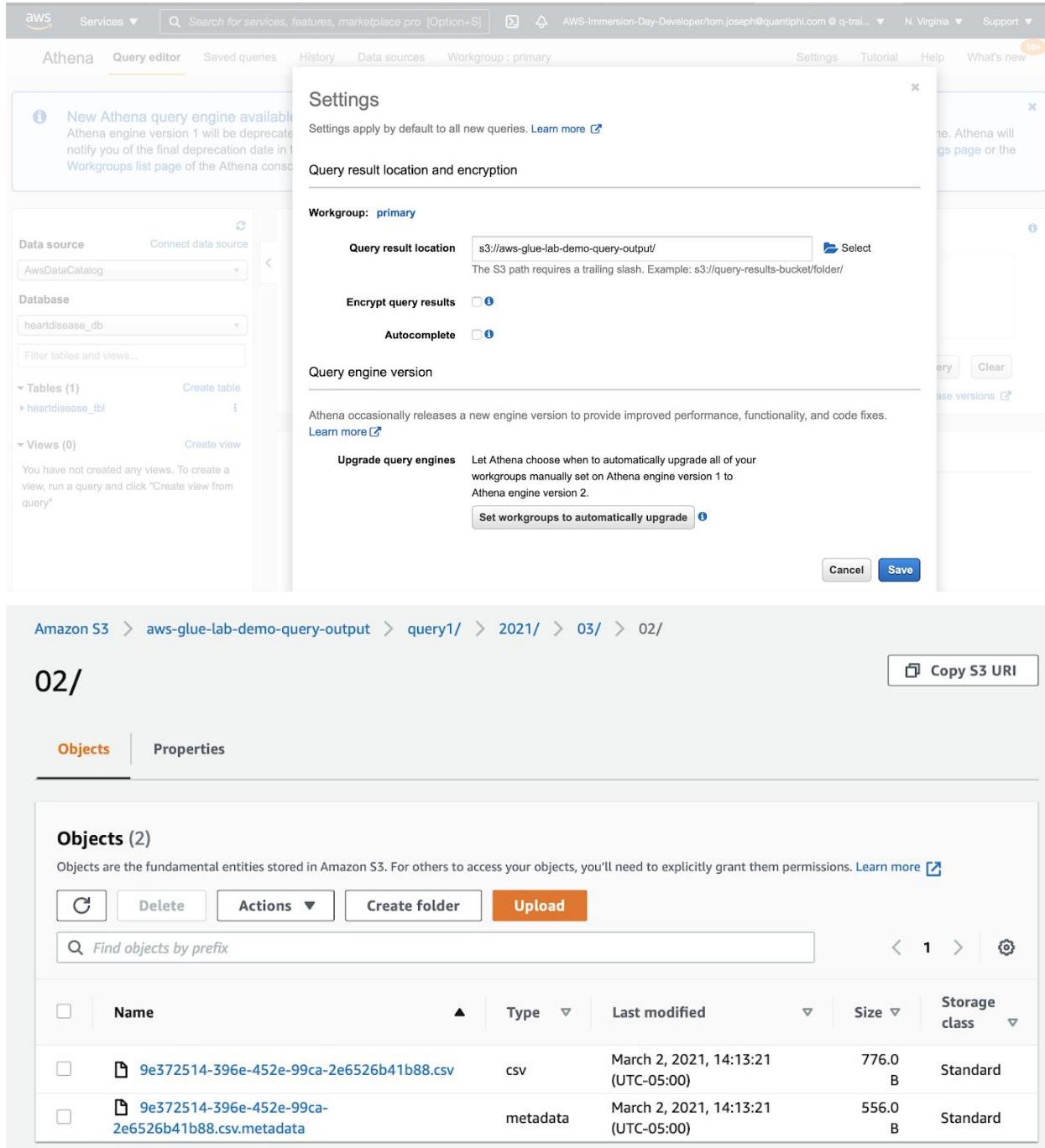
Athena is easy to use. Simply point to your data in Amazon S3, define the schema, and start querying using standard SQL. Most results are delivered within seconds. With Athena, there's no need for complex ETL jobs to prepare your data for analysis. This makes it easy for anyone with SQL skills to quickly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



The screenshot shows the AWS Athena Settings page and the corresponding AWS S3 Objects list.

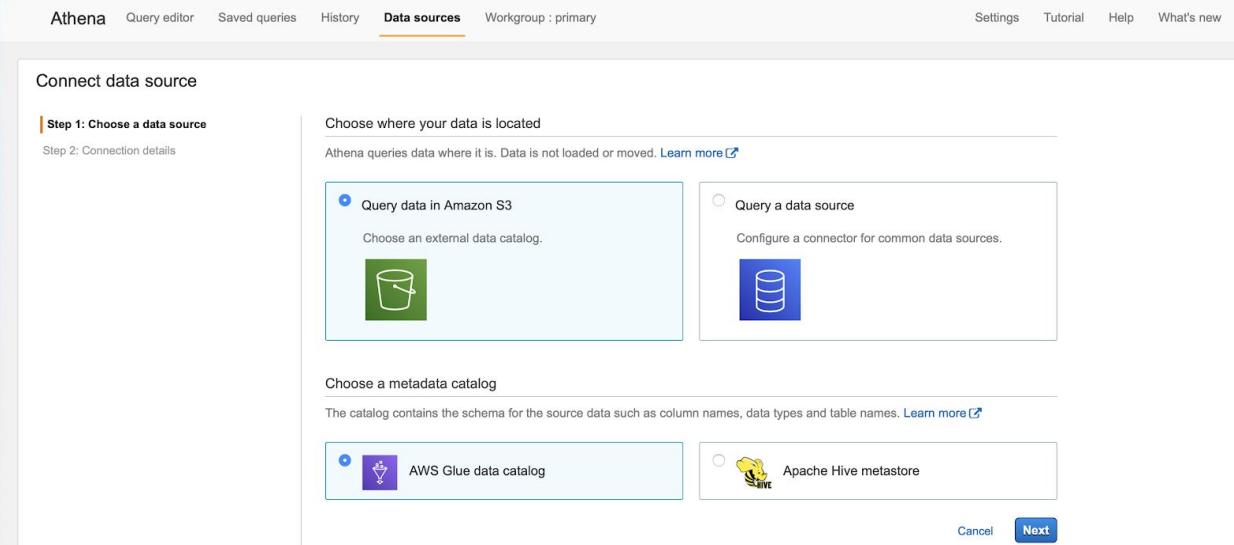
Athena Settings Page:

- Query result location:** s3://aws-glue-lab-demo-query-output/
- Workgroup:** primary
- Upgrade query engines:** Set workgroups to automatically upgrade

AWS S3 Objects List:

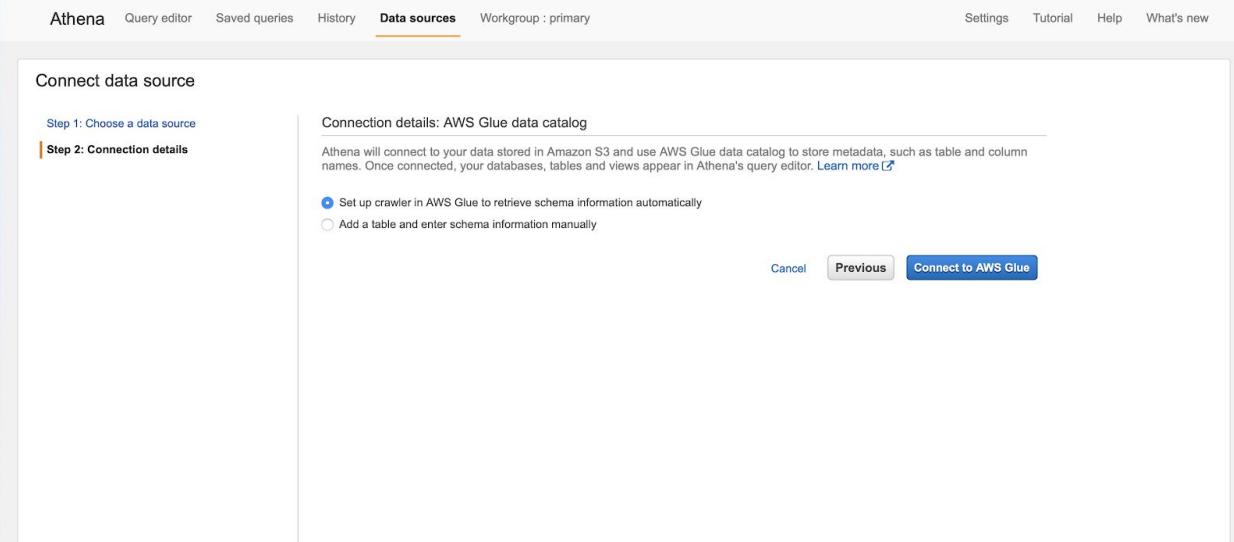
Name	Type	Last modified	Size	Storage class
9e372514-396e-452e-99ca-2e6526b41b88.csv	csv	March 2, 2021, 14:13:21 (UTC-05:00)	776.0 B	Standard
9e372514-396e-452e-99ca-2e6526b41b88.csv.metadata	metadata	March 2, 2021, 14:13:21 (UTC-05:00)	556.0 B	Standard

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



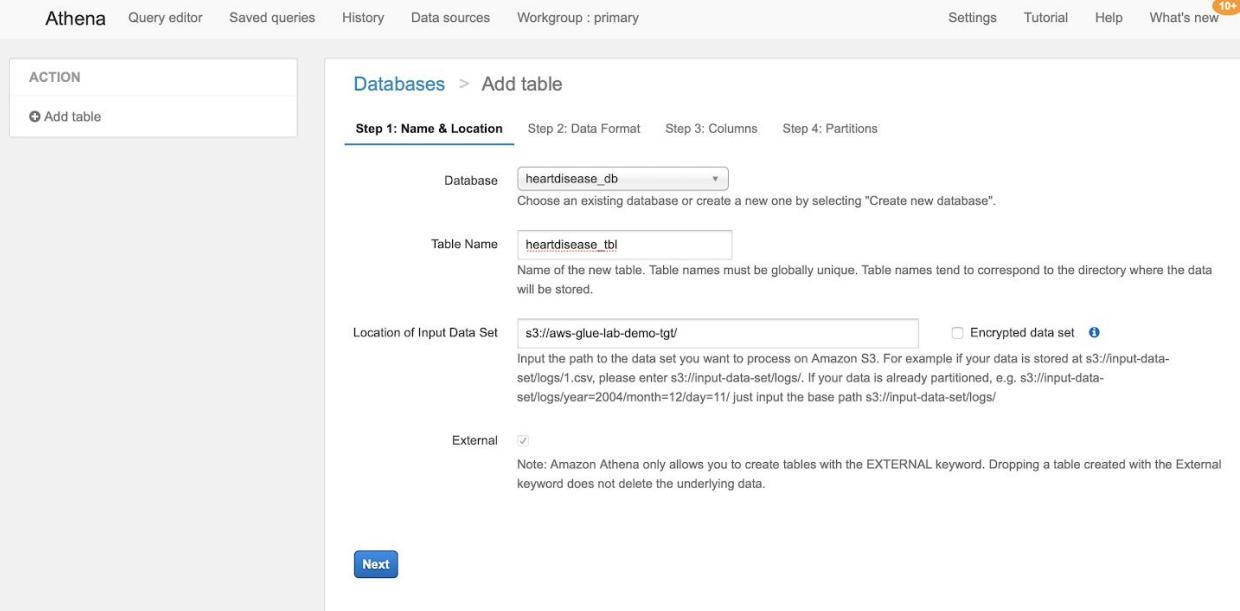
The screenshot shows the 'Data sources' step of the AWS Athena setup process. It asks to choose a data source and a metadata catalog. The 'Query data in Amazon S3' option is selected for the data source, and the 'AWS Glue data catalog' option is selected for the metadata catalog. Buttons for 'Cancel' and 'Next' are at the bottom.

Step 3: Select the option select table and schema manually



The screenshot shows the 'Data sources' step 2 of the AWS Athena setup process. It displays connection details for the AWS Glue data catalog, including the option to set up a crawler to retrieve schema information automatically. Buttons for 'Cancel', 'Previous', and 'Connect to AWS Glue' are at the bottom.

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



Athena Query editor Saved queries History Data sources Workgroup : primary Settings Tutorial Help What's new

Databases > Add table

Step 1: Name & Location Step 2: Data Format Step 3: Columns Step 4: Partitions

Database: **hearthdisease_db** Choose an existing database or create a new one by selecting "Create new database".

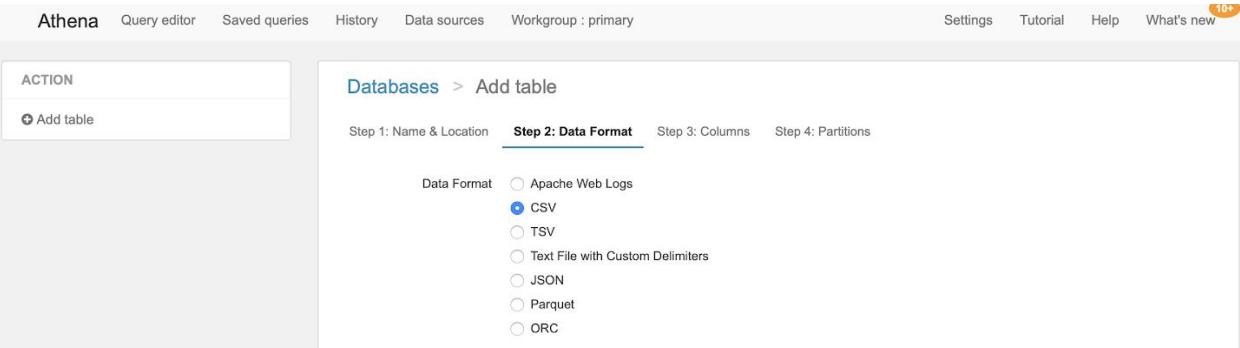
Table Name: **hearthdisease_tbl** Name of the new table. Table names must be globally unique. Table names tend to correspond to the directory where the data will be stored.

Location of Input Data Set: **s3://aws-glue-lab-demo-tgt/** Input the path to the data set you want to process on Amazon S3. For example if your data is stored at s3://input-data-set/logs/1.csv, please enter s3://input-data-set/logs/. If your data is already partitioned, e.g. s3://input-data-set/logs/year=2004/month=12/day=11/ just input the base path s3://input-data-set/logs/

External Note: Amazon Athena only allows you to create tables with the EXTERNAL keyword. Dropping a table created with the External keyword does not delete the underlying data.

Next

Step 5: Select file format as CSV



Athena Query editor Saved queries History Data sources Workgroup : primary Settings Tutorial Help What's new

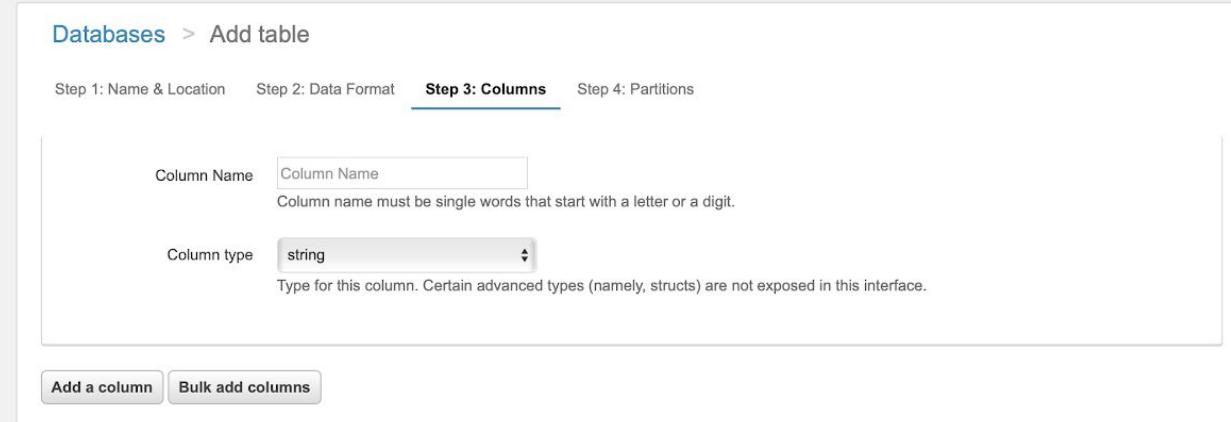
Databases > Add table

Step 1: Name & Location **Step 2: Data Format** Step 3: Columns Step 4: Partitions

Data Format:

- Apache Web Logs
- CSV
- TSV
- Text File with Custom Delimiters
- JSON
- Parquet
- ORC

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



Databases > Add table

Step 1: Name & Location Step 2: Data Format **Step 3: Columns** Step 4: Partitions

Column Name
Column name must be single words that start with a letter or a digit.

Column type ▼
Type for this column. Certain advanced types (namely, structs) are not exposed in this interface.

Add a column **Bulk add columns**

```
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

Databases > Add table

Step 1: Name & Location Step 2: Data Format **Step 3: Columns** Step 4: Partitions

Column Name

age

Column name must be single words that start with a letter or a digit.

Column type

decimal

Type for this column. Certain advanced types (namely, structs) are not exposed in this interface.

Column Name

sex

Column name must be single words that start with a letter or a digit.

Column type

decimal

Type for this column. Certain advanced types (namely, structs) are not exposed in this interface.

Step 7: Create Partitions if necessary

Databases > Add table

Step 1: Name & Location Step 2: Data Format Step 3: Columns **Step 4: Partitions**

Configure Partitions (Optional)

Partitions are a way to group specific information together. Partition are virtual columns. In case of partitioned tables, subdirectories are created under the table's data directory for each unique value of a partition column. In case the table is partitioned on multiple columns, then nested subdirectories are created based on the order of partition columns in the table definition. [Learn more.](#)

[Add a partition](#)

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

Athena **Query editor** Saved queries History Data sources Workgroup : primary Settings Tutorial Help What's new

New Athena query engine available
 Athena engine version 1 will be deprecated in the near future. Workgroups still on Athena engine version 1 will be upgraded to Athena engine version 2 at that time. Athena will notify you of the final deprecation date in the Athena console 30 days ahead of time. To set all workgroups to upgrade query engines automatically, use the [Settings page](#) or the [Workgroups list page](#) of the Athena console. To upgrade a specific workgroup, use the [Edit workgroup page](#). [Learn more](#)

Data source Connect data source
 AwsDataCatalog

Database heartdisease_db

Tables (1) Create table
 heartdisease_tbl

Views (0) Create view
 You have not created any views. To create a view, run a query and click "Create view from query"

New query 1 New query 2 New query 4 +

```

1 CREATE EXTERNAL TABLE IF NOT EXISTS heartdisease_db.heartdisease_tbl (
2   `age` int,
3   `sex` int,
4   `cp` int,
5   `trestbps` double,
6   `chol` double,
7   `fbs` double,
8   `restecg` int,
9   `thalach` double,
10  `exang` int,
11  `oldpeak` double,
12  `slope` int,
13  `ca` int,
14  `thal` int,
15  `num` int
16 )
17 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
18 WITH SERDEPROPERTIES (
19   'serialization.format' = ',',
20   'field.delim' = ','
21 ) LOCATION 's3://.../heart_disease/heart.csv';
  
```

Run query Save as Create (Run time: 0.44 seconds, Data scanned: 0 KB) Format query Clear

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete Athena engine version 1 Release versions

```

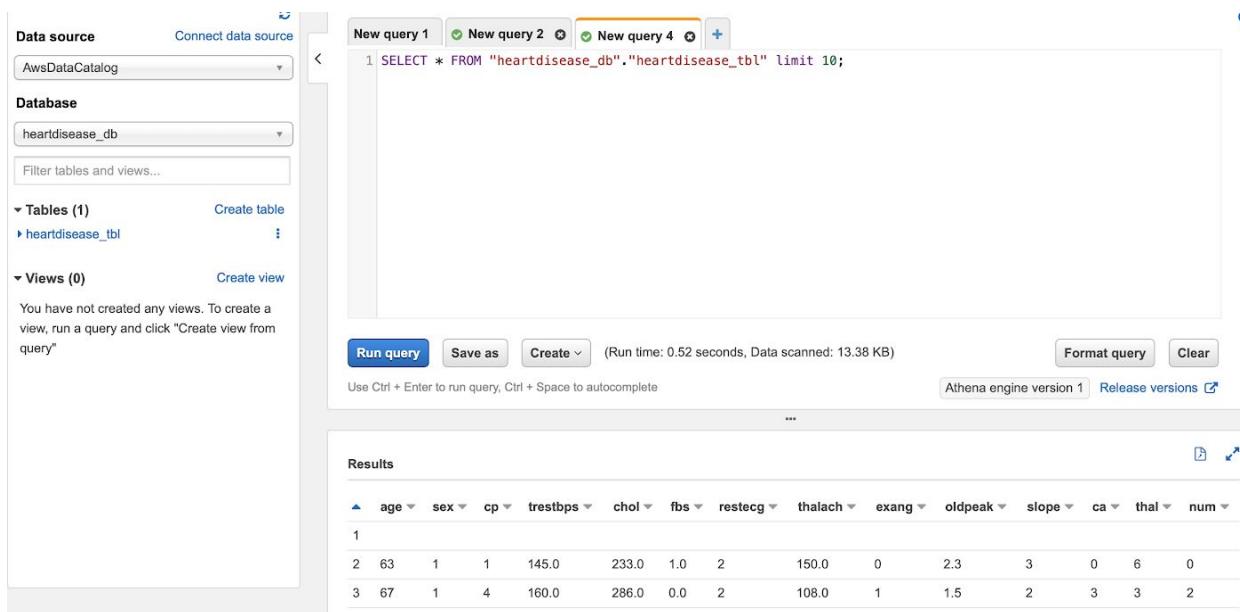
CREATE EXTERNAL TABLE IF NOT EXISTS heartdisease_db.heartdisease_tbl
(
  `age` int,
  `sex` int,
  `cp` int,
  `trestbps` double,
  `chol` double,
  `fbs` double,
  `restecg` int,
  `thalach` double,
  `exang` int,
  `oldpeak` double,
  `slope` int,
  `ca` int,
  `thal` int,
  `num` int
)
  
```

```

)
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



The screenshot shows the AWS Athena console interface. On the left, there's a sidebar with 'Data source' set to 'AwsDataCatalog', 'Database' set to 'heartdisease_db', and a list of 'Tables (1)' containing 'heartdisease_tbl'. Below that, under 'Views (0)', it says 'You have not created any views. To create a view, run a query and click "Create view from query"'. In the main area, there are four tabs: 'New query 1' (selected), 'New query 2', 'New query 4' (highlighted in orange), and '+'. A query is being run: '1 SELECT * FROM "heartdisease_db"."heartdisease_tbl" limit 10;'. Below the query, it says '(Run time: 0.52 seconds, Data scanned: 13.38 KB)'. At the bottom, there are buttons for 'Run query', 'Save as', 'Create', 'Format query', and 'Clear'. A note at the bottom says 'Use Ctrl + Enter to run query, Ctrl + Space to autocomplete'. The results section shows a table with 13 columns: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal, num. The first three rows of data are:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
1														
2	63	1	1	145.0	233.0	1.0	2	150.0	0	2.3	3	0	6	0
3	67	1	4	160.0	286.0	0.0	2	108.0	1	1.5	2	3	3	2