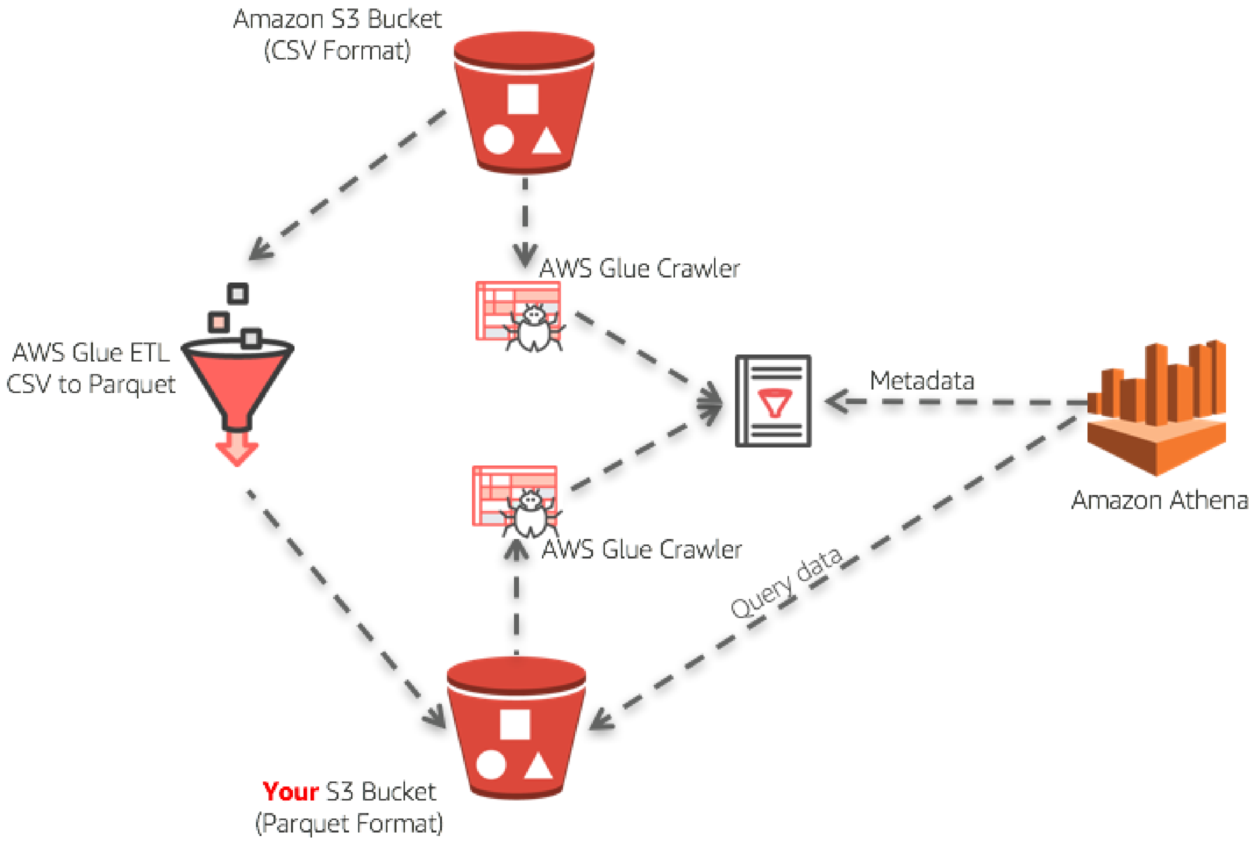
Lab 3: Serverless ETL and Data Discovery using Amazon Glue

- Create an IAM Role
- Create an Amazon S3 bucket
- Discover the Data
- Optimize the Queries and convert into Parquet
- Query the Partitioned Data using Amazon Athena
- Deleting the Glue database, crawlers and ETL Jobs created for this Lab
- Summary

Architectural Diagram



Create an IAM Role

Create an IAM role that has permission to your Amazon S3 sources, targets, temporary directory, scripts, **AWSGlueServiceRole** and any libraries used by the job. You can click here to create a new role. For additional documentation to create a role click here.

1. On the IAM page, click on **Create Role**.
2. Choose the service as **Glue** and click on **Next: Permissions** on the bottom.
3. On the Attach permissions policies, search policies for S3 and check the box for **AmazonS3FullAccess**.

   > Do not click on the policy, you just have to check the corresponding checkbox.

4. On the same page, now search policies for Glue and check the box for **AWSGlueServiceRole** and **AWSGlueConsoleFullAccess**.

   > Do not click on the policy, you just have to check the corresponding checkbox.

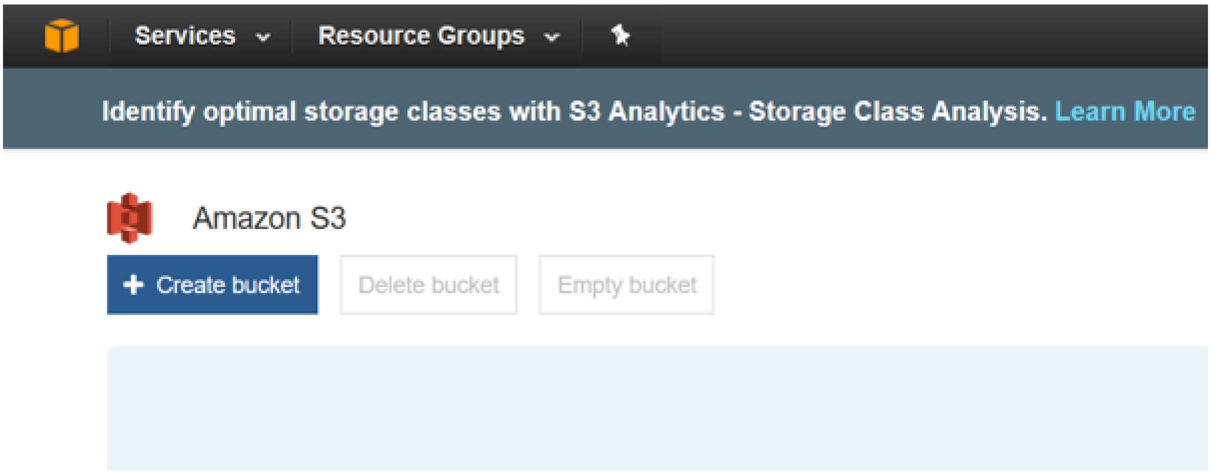
5. Click on **Next: Review**.
6. Enter Role name as

```
nycitytaxianalysis-reinv
```

and click **Create role**.

Create an Amazon S3 bucket

1. Open the AWS Management console for Amazon S3
2. On the S3 Dashboard, Click on **Create Bucket**.

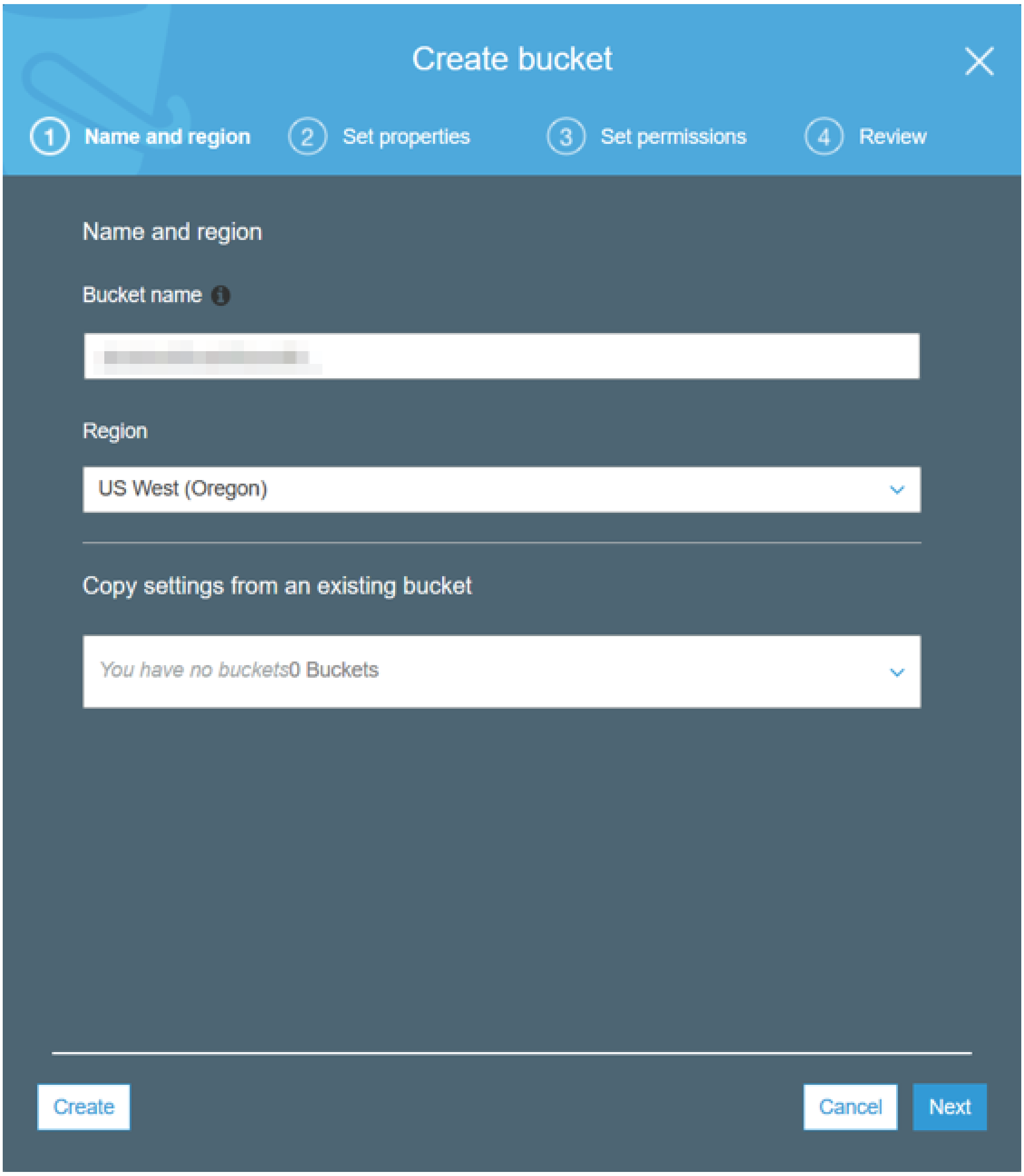

Services
Resource Groups
Identify optimal storage classes with S3 Analytics - Storage Class Analysis. Learn More
Amazon S3
+ Create bucket
Delete bucket
Empty bucket

1. In the **Create Bucket** pop-up page, input a unique **Bucket name.** So it's advised to choose a large bucket name, with many random characters and numbers (no spaces). It will be easier to name your bucket

```
aws-glue-scripts-<YOURAWSACCOUNTID>-us-west-2
```

and it would be easier to choose/select this bucket for the remainder of this Lab3.

1. Select the region as **Oregon.**
2. Click **Next** to navigate to the next tab.
3. In the **Set properties** tab, leave all options as default.
4. In the **Set permissions** tag, leave all options as default.
5. In the **Review** tab, click on **Create Bucket**



Name and region

Bucket name

[redacted]

Region

US West (Oregon)

Copy settings from an existing bucket

*You have no buckets*0 Buckets

2. Now, in this newly created bucket, create two sub-folders **tmp** and **target** using the same instructions as the above step. We will use these buckets as part of Lab3 later on.

Discover the Data

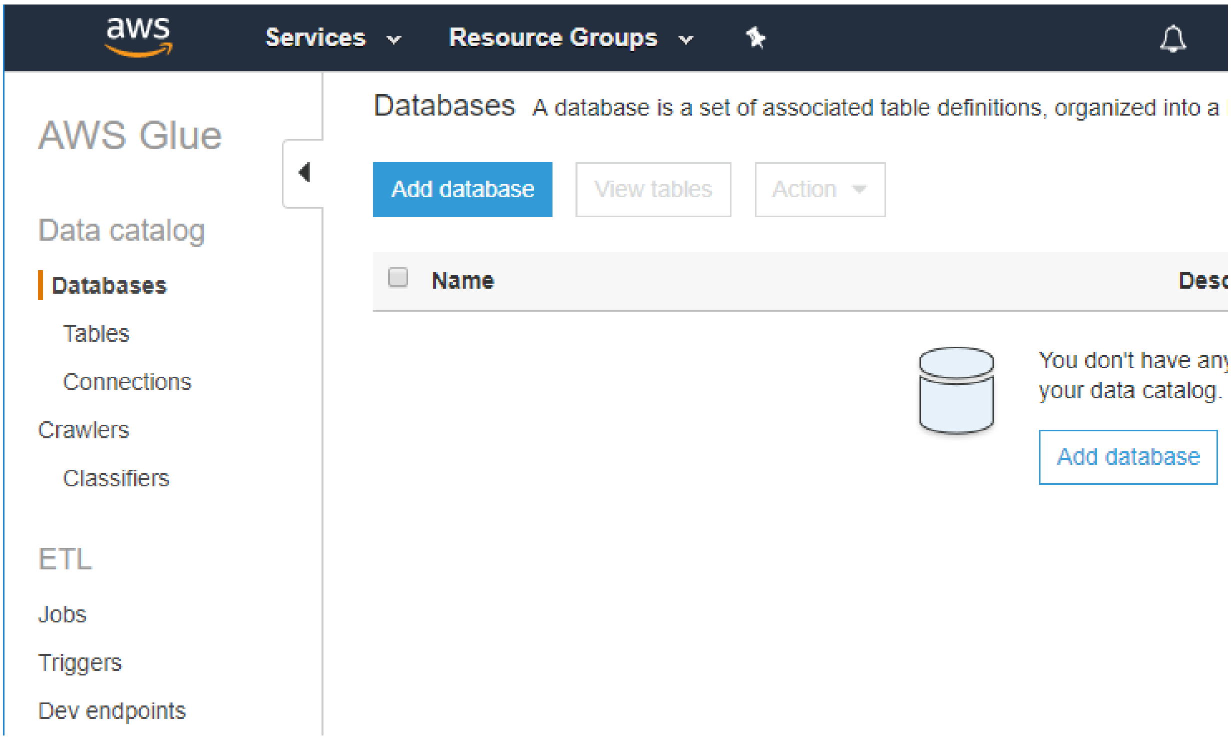
During this workshop, we will focus on one month of the New York City Taxi Records dataset, however you could easily do this for the entire eight years of data. As you crawl this unknown dataset, you discover that the data is in different formats, depending on the type of taxi. You then convert the data to a canonical form, start to analyze it, and build a set of visualizations. All without launching a single server.

> For this lab, you will need to choose the **US West (Oregon)** region.

1. Open the AWS Management console for Amazon Glue.

2. To analyze all the taxi rides for January 2016, you start with a set of data in S3. First, create a database for this workshop within AWS Glue. A database is a set of associated table definitions, organized into a logical group. In Athena, database names are all lowercase, no matter what you type.

   i. Click on **Databases** under Data Catalog column on the left.



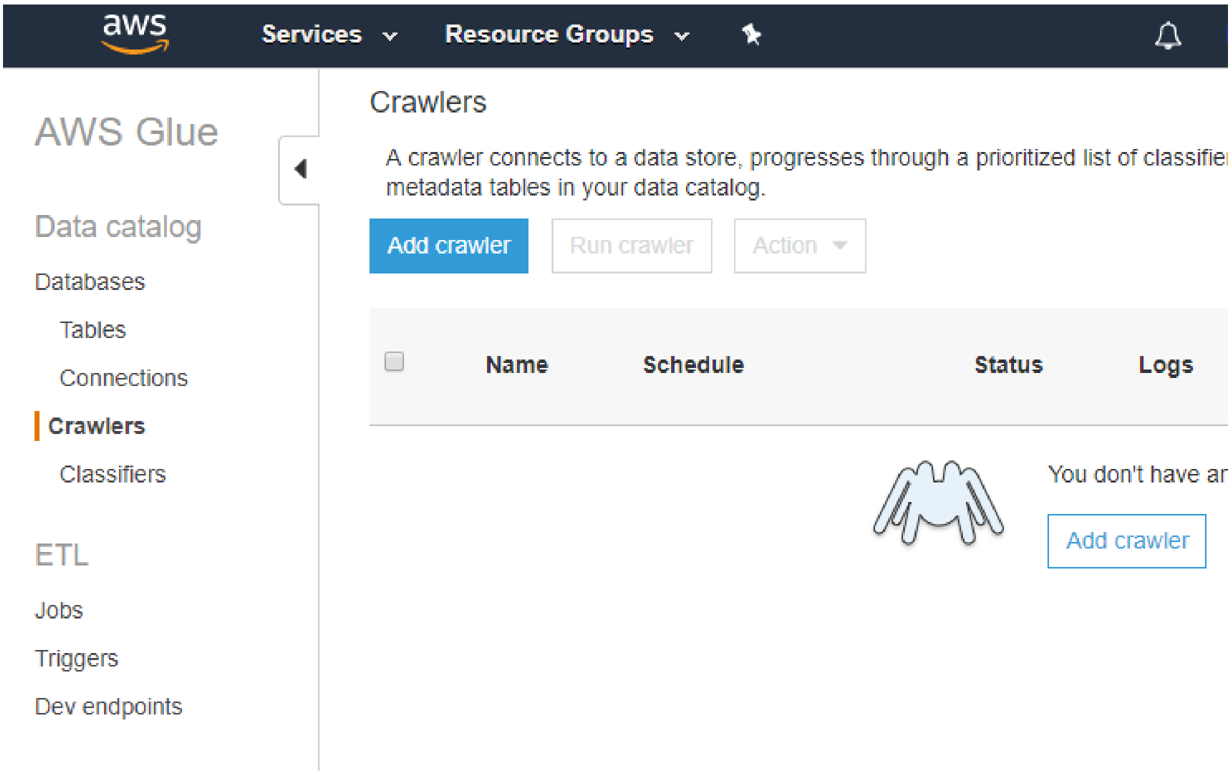
Databases A database is a set of associated table definitions, organized into a

You don't have any
your data catalog.

   ii. Click on the **Add Database** button.

   iii. Enter the Database name as **nycitytaxianalysis-reinv17**. You can skip the description and location fields and click on **Create**.

3. Click on **Crawlers** under Data Catalog column on the left.



A crawler connects to a data store, progresses through a prioritized list of classifie
metadata tables in your data catalog.

You don't have ar

i. Click on **Add Crawler** button.

ii. Under Add information about your crawler, for Crawler name type **nycitytaxianalysis-crawler-reinv17**. You can skip the Description and Classifiers field and click on **Next.**

iii. Under Specify crawler source type, make sure **Data stores** is selected. Click **Next**.

iv. When choosing a data store, make sure S3 is selected. Ensure the radio button for **Crawl Data in Specified path** is checked. For Include path, enter the following S3 path and click on **Next**.

```
s3://serverless-analytics/glue-blog
```

v. For Add Another data store, choose **No** and click on **Next**.

vi. For Choose an IAM Role, select **Create an IAM role** and enter the role name as following and click on **Next.**

```
nycitytaxianalysis-reinv17-crawler
```

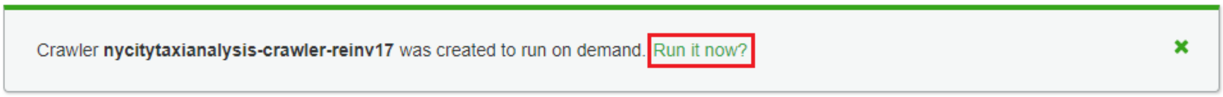
vii. For Create a schedule for this crawler, choose Frequency as **Run on Demand** and click on **Next.**

viii. Configure the crawler output database and prefix:

a. For **Database**, select the database created earlier, **nycitytaxianalysis-reinv17.**

b. For **Prefix added to tables (optional)**, type **reinv17_** and click on **Next**.

c. Review configuration and click on **Finish** and on the next page, click on **Run it now** in the green box on the top.

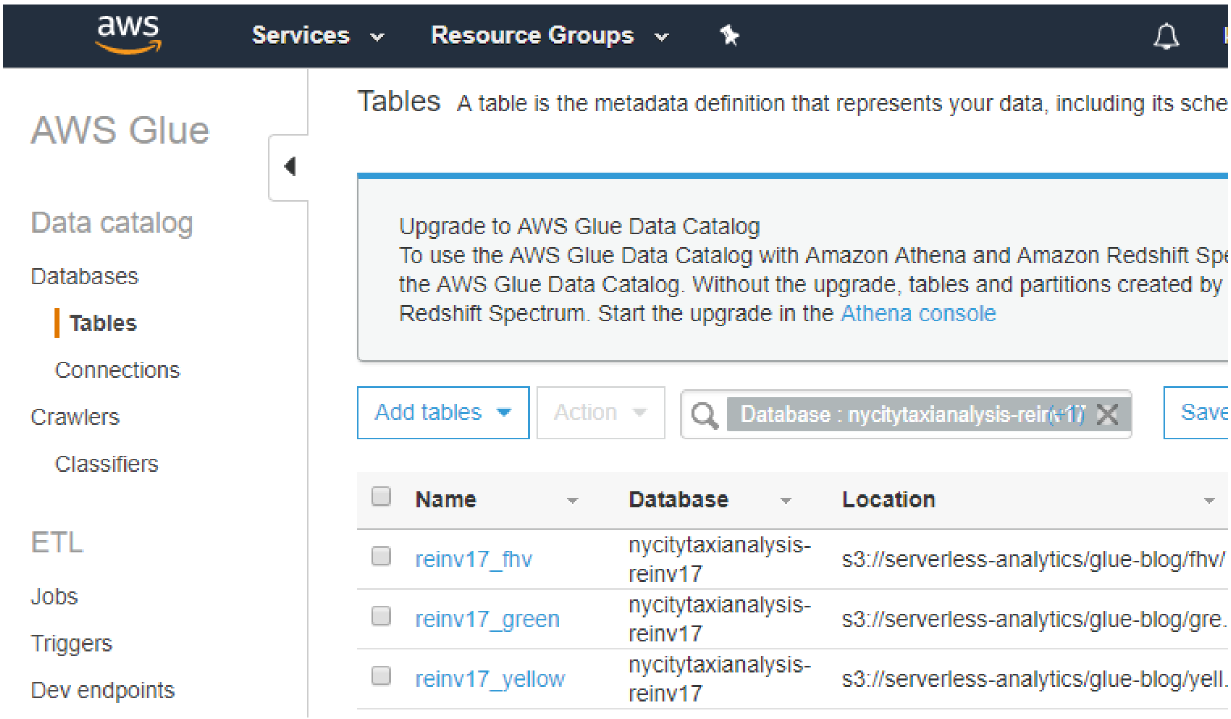


Crawler **nycitytaxianalysis-crawler-reinv17** was created to run on demand. Run it now?

d. The crawler runs and indicates that it found three tables.

4. Click on **Tables** under Data Catalog on the left column.

5. If you look under **Tables**, you can see the three new tables that were created under the database nycitytaxianalysis-reinv17.



Tables A table is the metadata definition that represents your data, including its sche

Upgrade to AWS Glue Data Catalog
To use the AWS Glue Data Catalog with Amazon Athena and Amazon Redshift Spe
the AWS Glue Data Catalog. Without the upgrade, tables and partitions created by
Redshift Spectrum. Start the upgrade in the Athena console

Database : nycitytaxianalysis-reinv17

□	Name	Database	Location
□	reinv17_fhv	nycitytaxianalysis-reinv17	s3://serverless-analytics/glue-blog/fhv/
□	reinv17_green	nycitytaxianalysis-reinv17	s3://serverless-analytics/glue-blog/gre.
□	reinv17_yellow	nycitytaxianalysis-reinv17	s3://serverless-analytics/glue-blog/yell.

6. The crawler used the built-in classifiers and identified the tables as CSV, inferred the columns/data types, and collected a set of properties for each table. If you look in each of those table definitions, you see the number of rows for each dataset found and that the columns don't match between tables. As an example, clicking on the reinv17_yellow table, you can see the yellow dataset for January 2017 with 8.7 million rows, the location on S3, and the various columns found.

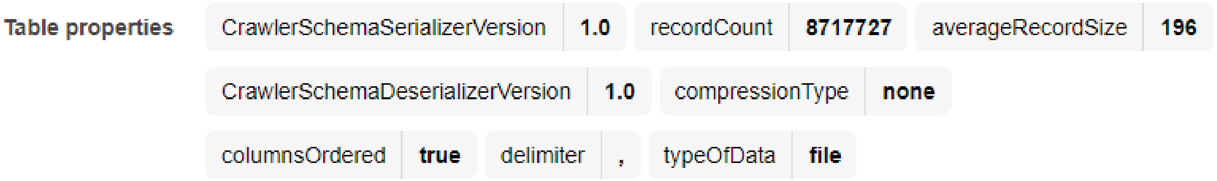


Table properties

Property	Value
CrawlerSchemaSerializerVersion	1.0
recordCount	8717727
averageRecordSize	196
CrawlerSchemaDeserializerVersion	1.0
compressionType	none
columnsOrdered	true
delimiter	,
typeOfData	file

Optimize the Queries and convert into Parquet

Create an ETL job to move this data into a query-optimized form. You convert the data into a column format, changing the storage type to Parquet, and writing the data to a bucket that you own.

1. Open the AWS Management console for Amazon Glue.

2. Click on **Jobs** under ETL on the left column and then click on the **Add Job** button.

3. Under Job properties, input name as **nycitytaxianalysis-reinv17-yellow**. Since we will be working with only the yellow dataset for this workshop.

   i. Under IAM Role, Choose the IAM role created at the beginning of this lab which should be named **nycitytaxianalysis-reinv**.

   x. Under This job runs, choose the radio button for **A proposed script generated by AWS Glue**.

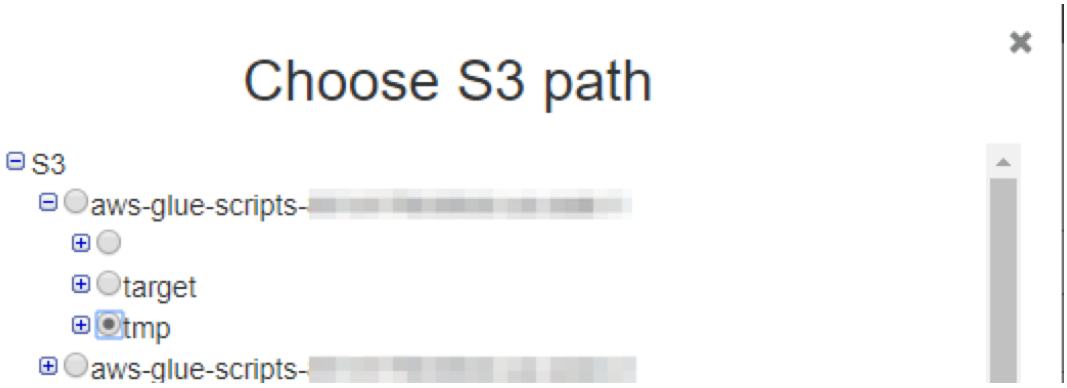
   xi. For Script file name, enter **nycitytaxianalysis-reinv17-yellow**.

   > For this workshop, we are only working on the yellow dataset. Feel free to run through these steps to also convert the green and FHV dataset.

   xii. For S3 path where script is stored, click on the Folder icon and choose the S3 bucket created at the beginning of this workshop. **Choose the newly created S3 bucket via the Folder icon and click Select.**

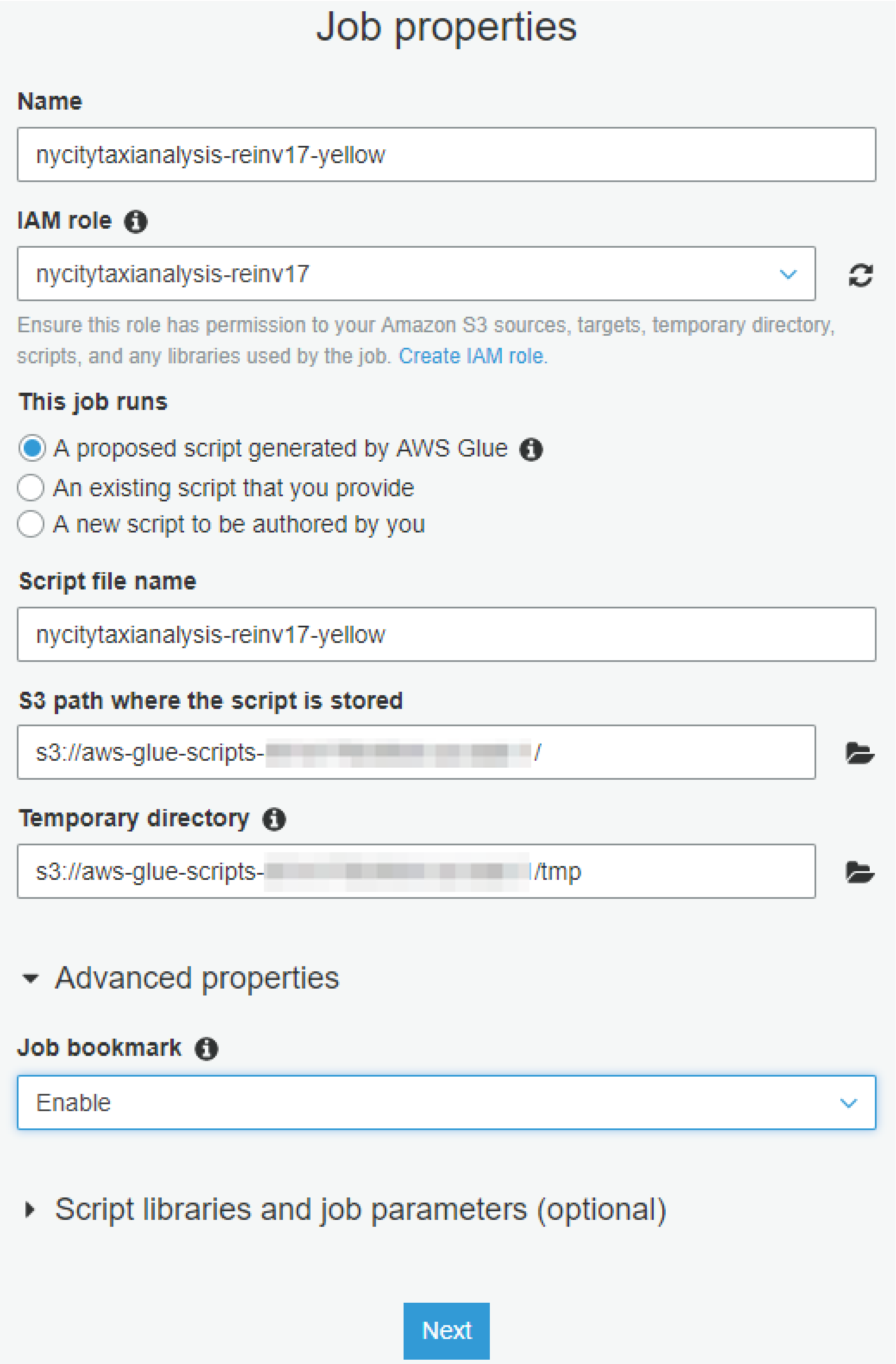
   xiii. For Temporary directory, choose the tmp folder created at the beginning of this workshop. **Choose the S3 bucket via the Folder icon** and click **Select**.

   > Ensure the temporary bucket is already created/available in your S3 bucket.


Choose S3 path
⊟ S3
⊟ ○ aws-glue-scripts-[redacted]
⊞ ○
⊞ ○ target
⊞ ● tmp
⊞ ○ aws-glue-scripts-[redacted]

   xiv. Click on Advanced properties, and select **Enable** for Job bookmark.

   xv. Here's a screenshot of a finished job properties window:



Job properties

Name

nycitytaxianalysis-reinv17-yellow

IAM role

nycitytaxianalysis-reinv17

Ensure this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. Create IAM role.

This job runs

● A proposed script generated by AWS Glue

○ An existing script that you provide

○ A new script to be authored by you

Script file name

nycitytaxianalysis-reinv17-yellow

S3 path where the script is stored

s3://aws-glue-scripts-[redacted]/

Temporary directory

s3://aws-glue-scripts-[redacted]/tmp

▾ Advanced properties

Job bookmark

Enable

▸ Script libraries and job parameters (optional)

4. Click **Next.**

5. Under Choose your data sources, select **reinv17_yellow** table as the data source and click on **Next.**

6. Under Choose a transform type, make sure **Change schema** is selected and click on **Next.**

> For this workshop, we are only working on the yellow dataset. Feel free to run through these steps to also convert the green and FHV

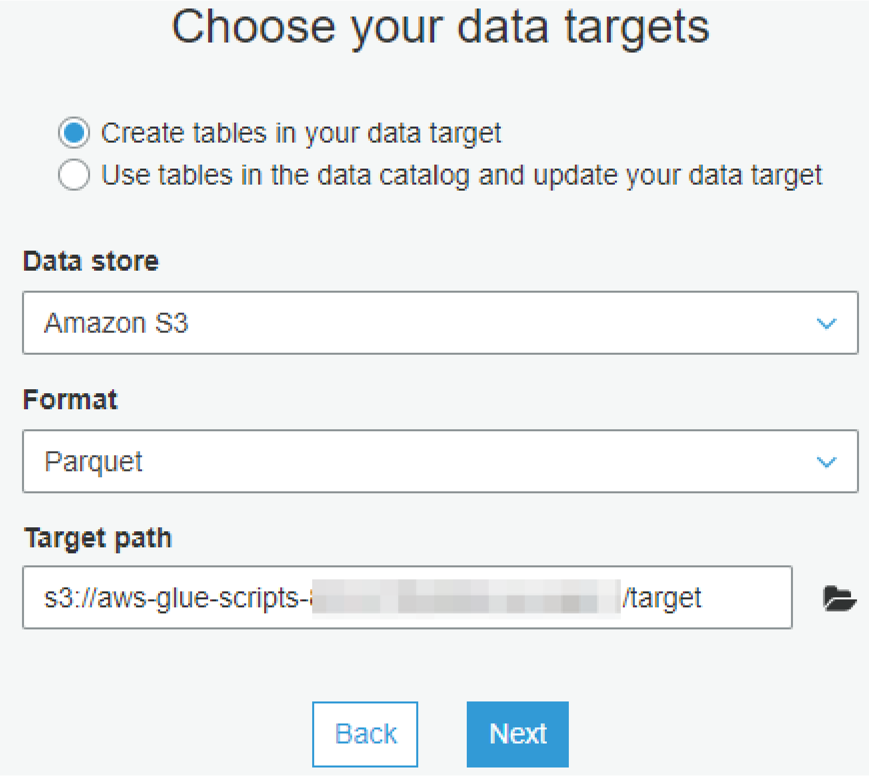
> dataset.

7. Under Choose your data targets, select the radio button for **Create tables in your data target**.

   i. For Data store, Choose **Amazon S3**.

   ii. For Format, choose **Parquet**.

   iii. For Target path, **click on the folder icon** and choose the target folder previously created. **This S3 Bucket/Folder will contain the transformed Parquet data**.



Choose your data targets

● Create tables in your data target
○ Use tables in the data catalog and update your data target

Data store

Amazon S3

Format

Parquet

Target path

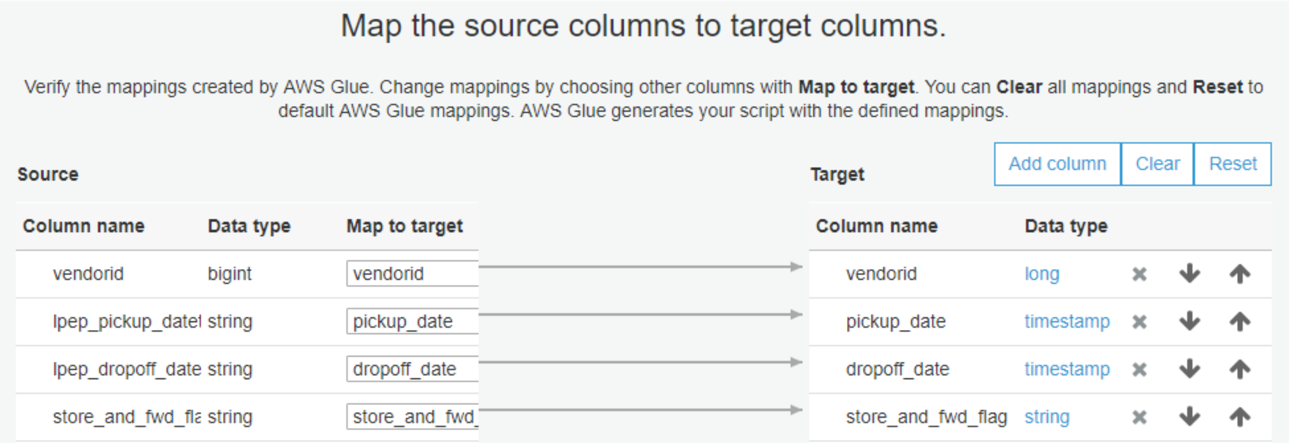
s3://aws-glue-scripts-[redacted]/target

7. Under Map the source columns to target columns page,

   i. Under Target, change the Column name **tpep_pickup_datetime** to **pickup_date**. Click on its respective **data type** field string and change the Column type to **TIMESTAMP** and click on **Update**.

   ii. Under Target, change the Column name **tpep_dropoff_datetime** to **dropoff_date**. Click on its respective **data type** field string and change the Column type to **TIMESTAMP** and click on **Update**.

   iii. Choose **Save job and edit script**.

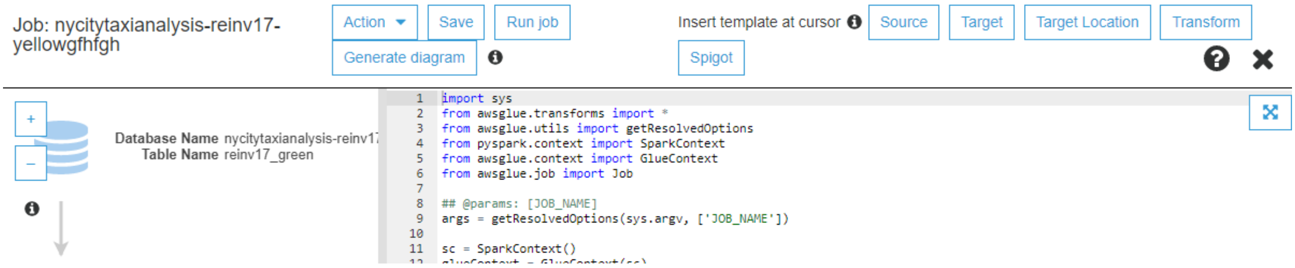


Map the source columns to target columns.

Verify the mappings created by AWS Glue. Change mappings by choosing other columns with **Map to target**. You can **Clear** all mappings and **Reset** to default AWS Glue mappings. AWS Glue generates your script with the defined mappings.

Source				Target	
Column name	**Data type**	**Map to target**		**Column name**	**Data type**
vendorid	bigint	vendorid	→	vendorid	long
lpep_pickup_datet	string	pickup_date	→	pickup_date	timestamp
lpep_dropoff_date	string	dropoff_date	→	dropoff_date	timestamp
store_and_fwd_fla	string	store_and_fwd_	→	store_and_fwd_flag	string

8. On the auto-generated script page, click on **Save** and **Run Job**.



Job: nycitytaxianalysis-reinv17-yellowgfhfgh

Database Name nycitytaxianalysis-reinv17
Table Name reinv17_green

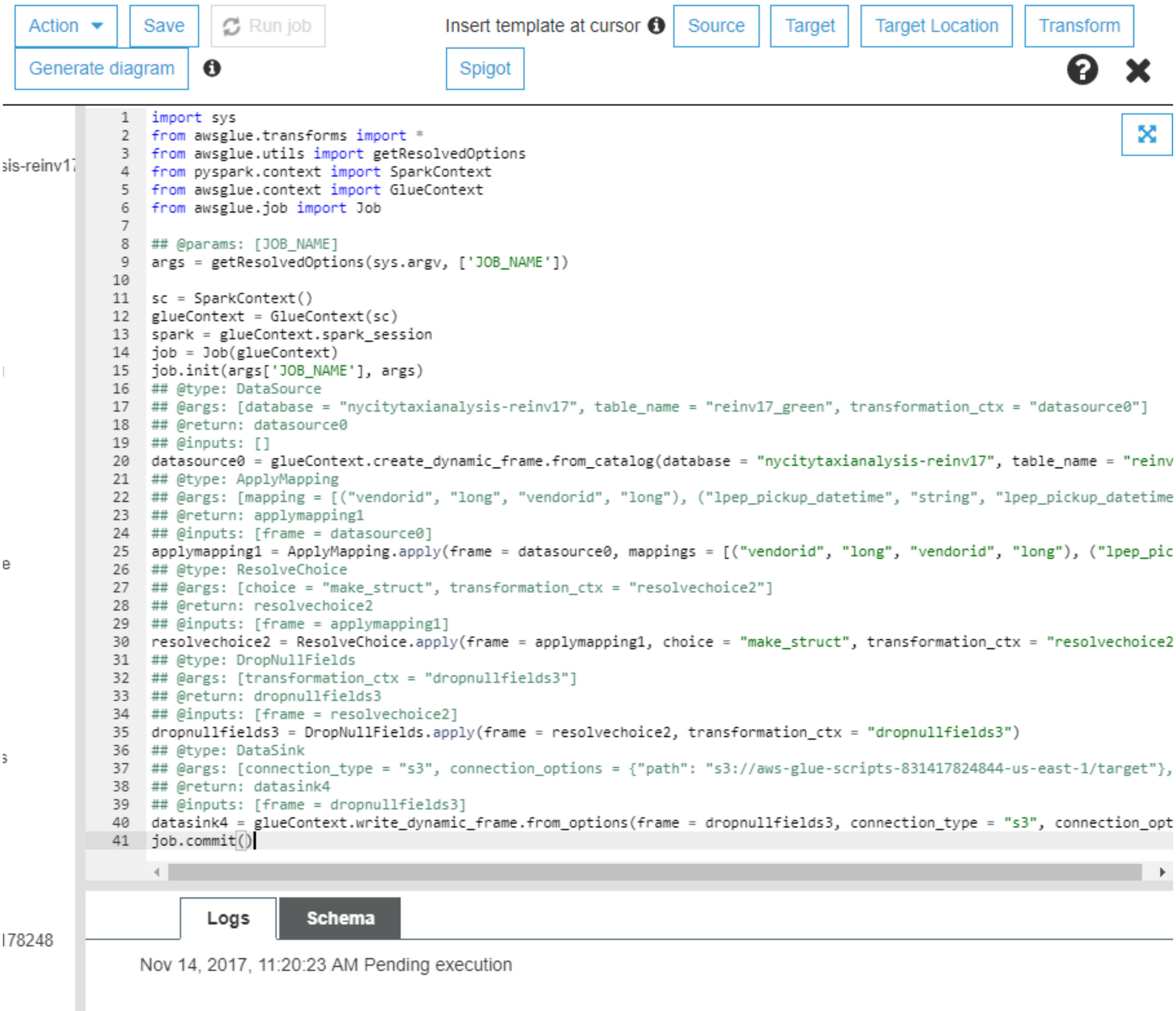
```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
```

8. In the parameters pop-up under Advanced properties, ensure Job bookmark is **Enabled** and click on **Run Job**.

9. This job will run for roughly around 30 minutes.



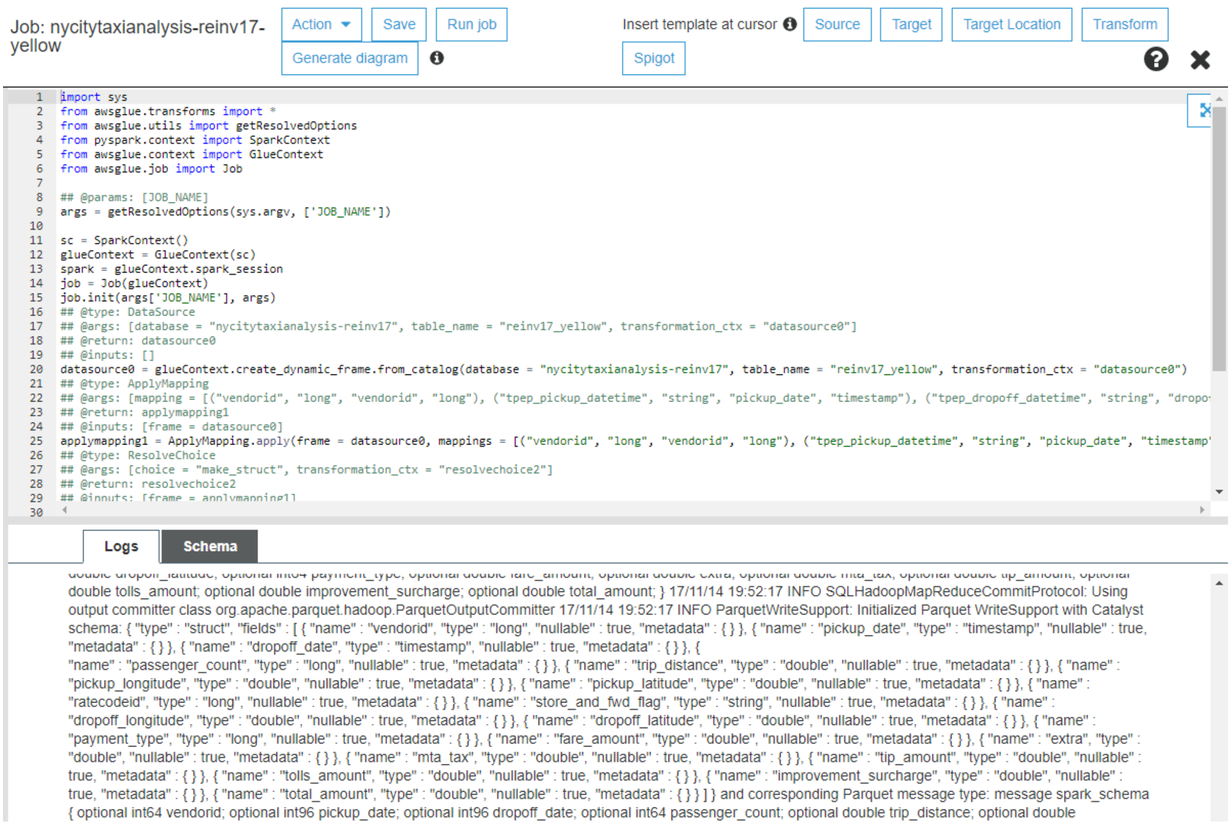
```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
## @type: DataSource
## @args: [database = "nycitytaxianalysis-reinv17", table_name = "reinv17_green", transformation_ctx = "datasource0"]
## @return: datasource0
## @inputs: []
datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "nycitytaxianalysis-reinv17", table_name = "reinv
## @type: ApplyMapping
## @args: [mapping = [("vendorid", "long", "vendorid", "long"), ("lpep_pickup_datetime", "string", "lpep_pickup_datetime
## @return: applymapping1
## @inputs: [frame = datasource0]
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [("vendorid", "long", "vendorid", "long"), ("lpep_pic
## @type: ResolveChoice
## @args: [choice = "make_struct", transformation_ctx = "resolvechoice2"]
## @return: resolvechoice2
## @inputs: [frame = applymapping1]
resolvechoice2 = ResolveChoice.apply(frame = applymapping1, choice = "make_struct", transformation_ctx = "resolvechoice2
## @type: DropNullFields
## @args: [transformation_ctx = "dropnullfields3"]
## @return: dropnullfields3
## @inputs: [frame = resolvechoice2]
dropnullfields3 = DropNullFields.apply(frame = resolvechoice2, transformation_ctx = "dropnullfields3")
## @type: DataSink
## @args: [connection_type = "s3", connection_options = {"path": "s3://aws-glue-scripts-831417824844-us-east-1/target"},
## @return: datasink4
## @inputs: [frame = dropnullfields3]
datasink4 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields3, connection_type = "s3", connection_opt
job.commit()
```

Nov 14, 2017, 11:20:23 AM Pending execution

10. You can view logs on the bottom page of the same page.



```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
## @type: DataSource
## @args: [database = "nycitytaxianalysis-reinv17", table_name = "reinv17_yellow", transformation_ctx = "datasource0"]
## @return: datasource0
## @inputs: []
datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "nycitytaxianalysis-reinv17", table_name = "reinv17_yellow", transformation_ctx = "datasource0")
## @type: ApplyMapping
## @args: [mapping = [("vendorid", "long", "vendorid", "long"), ("tpep_pickup_datetime", "string", "pickup_date", "timestamp"), ("tpep_dropoff_datetime", "string", "dropo
## @return: applymapping1
## @inputs: [frame = datasource0]
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [("vendorid", "long", "vendorid", "long"), ("tpep_pickup_datetime", "string", "pickup_date", "timestamp'
## @type: ResolveChoice
## @args: [choice = "make_struct", transformation_ctx = "resolvechoice2"]
## @return: resolvechoice2
## @inputs: [frame = applymapping1]
```

```
double dropoff_latitude; optional int64 payment_type; optional double fare_amount; optional double extra; optional double mta_tax; optional double tip_amount; optional
double tolls_amount; optional double improvement_surcharge; optional double total_amount; } 17/11/14 19:52:17 INFO SQLHadoopMapReduceCommitProtocol: Using
output committer class org.apache.parquet.hadoop.ParquetOutputCommitter 17/11/14 19:52:17 INFO ParquetWriteSupport: Initialized Parquet WriteSupport with Catalyst
schema: { "type" : "struct", "fields" : [ { "name" : "vendorid", "type" : "long", "nullable" : true, "metadata" : { } }, { "name" : "pickup_date", "type" : "timestamp", "nullable" : true,
"metadata" : { } }, { "name" : "dropoff_date", "type" : "timestamp", "nullable" : true, "metadata" : { } }, {
"name" : "passenger_count", "type" : "long", "nullable" : true, "metadata" : { } }, { "name" : "trip_distance", "type" : "double", "nullable" : true, "metadata" : { } }, { "name" :
"pickup_longitude", "type" : "double", "nullable" : true, "metadata" : { } }, { "name" : "pickup_latitude", "type" : "double", "nullable" : true, "metadata" : { } }, { "name" :
"ratecodeid", "type" : "long", "nullable" : true, "metadata" : { } }, { "name" : "store_and_fwd_flag", "type" : "string", "nullable" : true, "metadata" : { } }, { "name" :
"dropoff_longitude", "type" : "double", "nullable" : true, "metadata" : { } }, { "name" : "dropoff_latitude", "type" : "double", "nullable" : true, "metadata" : { } }, { "name" :
"payment_type", "type" : "long", "nullable" : true, "metadata" : { } }, { "name" : "fare_amount", "type" : "double", "nullable" : true, "metadata" : { } }, { "name" : "extra", "type" :
"double", "nullable" : true, "metadata" : { } }, { "name" : "mta_tax", "type" : "double", "nullable" : true, "metadata" : { } }, { "name" : "tip_amount", "type" : "double", "nullable" :
true, "metadata" : { } }, { "name" : "tolls_amount", "type" : "double", "nullable" : true, "metadata" : { } }, { "name" : "improvement_surcharge", "type" : "double", "nullable" :
true, "metadata" : { } }, { "name" : "total_amount", "type" : "double", "nullable" : true, "metadata" : { } } ] } and corresponding Parquet message type: message spark_schema
{ optional int64 vendorid; optional int96 pickup_date; optional int96 dropoff_date; optional int64 passenger_count; optional double trip_distance; optional double
```

11. The target folder (S3 Bucket) specified above (step 6 iii) will now have the converted parquet data.

Query the Partitioned Data using Amazon Athena

In regions where AWS Glue is supported, Athena uses the AWS Glue Data Catalog as a central location to store and retrieve table metadata throughout an AWS account. The Athena execution engine requires table metadata that instructs it where to read data, how to read it, and other information necessary to process the data. The AWS Glue Data Catalog provides a unified metadata repository across a variety of data sources and data formats, integrating not only with Athena, but with Amazon S3, Amazon RDS, Amazon Redshift, Amazon Redshift Spectrum, Amazon EMR, and any application compatible with the Apache Hive metastore.

1. Open the AWS Management console for Amazon Athena.

   > Ensure you are in the **US West (Oregon)** region.

2. Under Database, you should see the database **nycitytaxianalysis-reinv17** which was created during the previous section.

3. Click on **Create Table** right below the drop-down for Database and click on **From AWS Glue Crawler**. Click **Continue**.

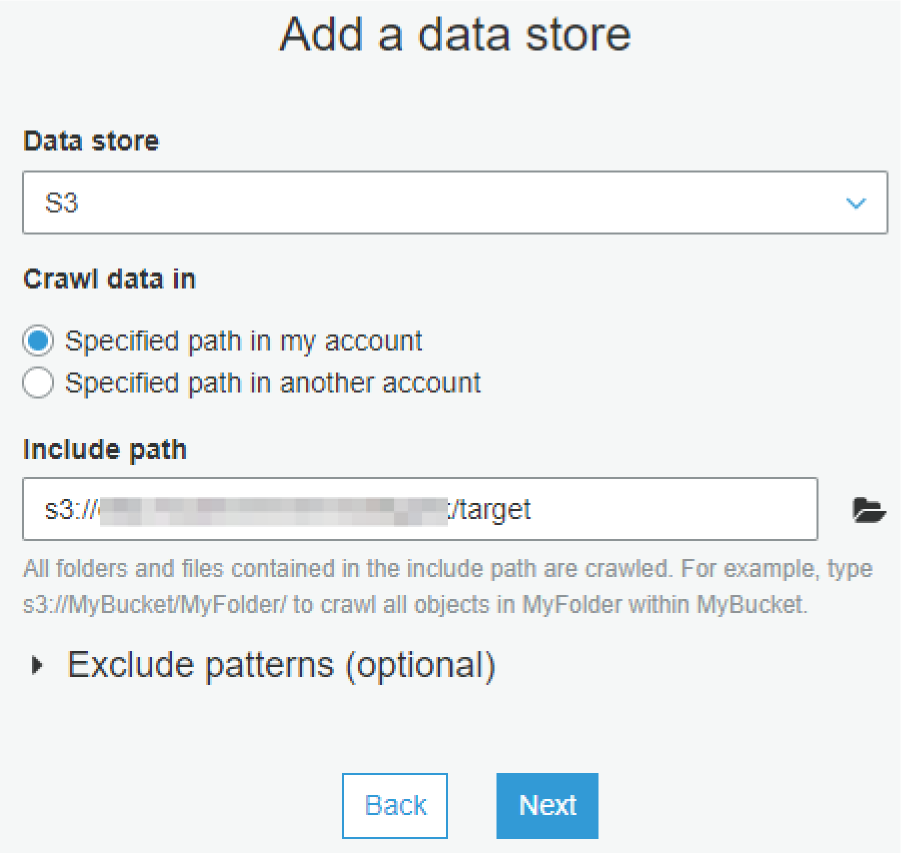
4. You will now be re-directed to the AWS Glue console to set up a crawler. The crawler connects to your data store and automatically determines its structure to create the metadata for your table.

5. Enter Crawler name as **nycitytaxianalysis-crawlerparquet-reinv17** and Click **Next.**

6. Under Specify crawler source type, make sure **Data stores** is selected. Hit **Next** and on the next page, select the Data store as **S3**.

7. Choose Crawl data in **Specified path in my account**.

8. For Include path, click on the folder Icon and choose the **target** folder previously made which contains the parquet data and click on **Next**.



Add a data store

Data store

S3

Crawl data in

● Specified path in my account

○ Specified path in another account

Include path

s3://[redacted]/target

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

▸ Exclude patterns (optional)

9. In Add another data store, choose **No** and click on **Next.**

10. For Choose an IAM role, select Choose an existing IAM role, and in the drop-down pick the role made in the previous section which should look similar to **AWSGlueServiceRole-nycitytaxianalysis-reinv17-crawler**. Click on **Next.**

11. In Create a schedule for this crawler, pick frequency as **Run on demand** and click on **Next.**

12. For Configure the crawler's output, Click **Add Database** to create a new database. Enter **nycitytaxianalysis-reinv17-parquet** as the database name and click **Create.** For Prefix added to tables, you can enter a prefix **parq_** and click **Next**.

13. Review the Crawler Info and click **Finish.** Click on **Run it Now?.**

14. Click on **Tables** on the left, and for database nycitytaxianalysis-reinv17-parquet you should see the table parq_target. Click on the table name and you will see the MetaData for this converted table.

15. Open the [AWS Management console for Amazon Athena](#).

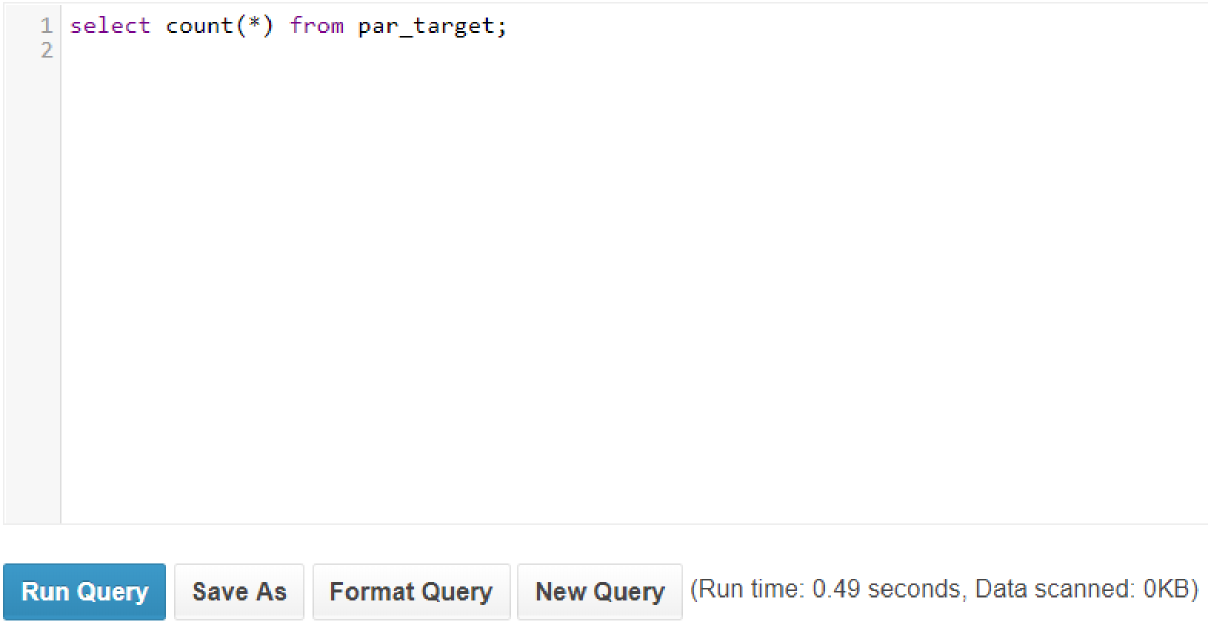
    > Ensure you are in the **US West (Oregon)** region.

16. Under Database, you should see the database **nycitytaxianalysis-reinv17-parquet** which was just created. Select this database and you should see under Tables **parq_target**.

17. In the query editor on the right, type

```
select count(*) from parq_target;
```

and take note the Run Time and Data scanned numbers here.



```
select count(*) from par_target;
```

(Run time: 0.49 seconds, Data scanned: 0KB)

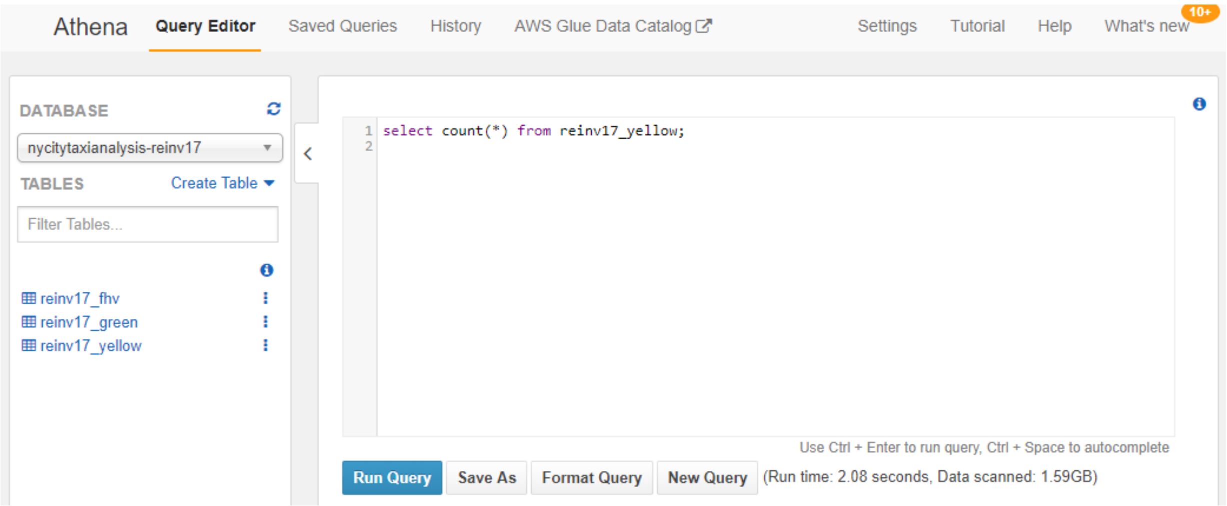
What we see is the Run time and Data scanned numbers for Amazon Athena to **query and scan the parquet data.**

18. Under Database, you should see the earlier made database **nycitytaxianalysis-reinv17** which was created in a previous section. Select this database and you should see under Tables **reinv17_yellow**.

19. In the query editor on the right, type

```
select count(*) from reinv17_yellow;
```

and take note the Run Time and Data scanned numbers here.



```
select count(*) from reinv17_yellow;
```

(Run time: 2.08 seconds, Data scanned: 1.59GB)

20. What we see is the Run time and Data scanned numbers for Amazon Athena to query and scan the uncompressed data from the previous section.

> Note: Athena charges you by the amount of data scanned per query. You can save on costs and get better performance if you partition the data, compress data, or convert it to columnar formats such as Apache Parquet.

Deleting the Glue database, crawlers and ETL Jobs created for this Lab

Now that you have successfully discovered and analyzed the dataset using Amazon Glue and Amazon Athena, you need to delete the resources created as part of this lab.

1. Open the AWS Management console for Amazon Glue. Ensure you are in the Oregon region (as part of this lab).
2. Click on **Databases** under Data Catalog column on the left.
3. Check the box for the Database that were created as part of this lab. Click on **Action** and select **Delete Database**. And click on **Delete**. This will also delete the tables under this database.
4. Click on **Crawlers** under Data Catalog column on the left.
5. Check the box for the crawler that were created as part of this lab. Click on **Action** and select **Delete Crawler**. And click on **Delete**.
6. Click on **Jobs** under ETL column on the left.
7. Check the box for the jobs that were created as part of this lab. Click on **Action** and select **Delete**. And click on **Delete**.
8. Open the AWS Management console for Amazon S3.
9. Click on the S3 bucket that was created as part of this lab. You need to click on its corresponding **Bucket icon** to select the bucket instead of opening the bucket. Click

on **Delete bucket** button on the top, to delete the S3 bucket. In the pop-up window, Type the name of the bucket (that was created as part of this lab), and click **Confirm.**

Summary

In the lab, you went from data discovery to analyzing a canonical dataset, without starting and setting up a single server. You started by crawling a dataset you didn’t know anything about and the crawler told you the structure, columns, and counts of records.

From there, you saw the datasets were in different formats, but represented the same thing: NY City Taxi rides. You then converted them into a canonical (or normalized) form that is easily queried through Athena and possible in QuickSight, in addition to a wide number of different tools not covered in this post.

License

This library is licensed under the Apache 2.0 License.