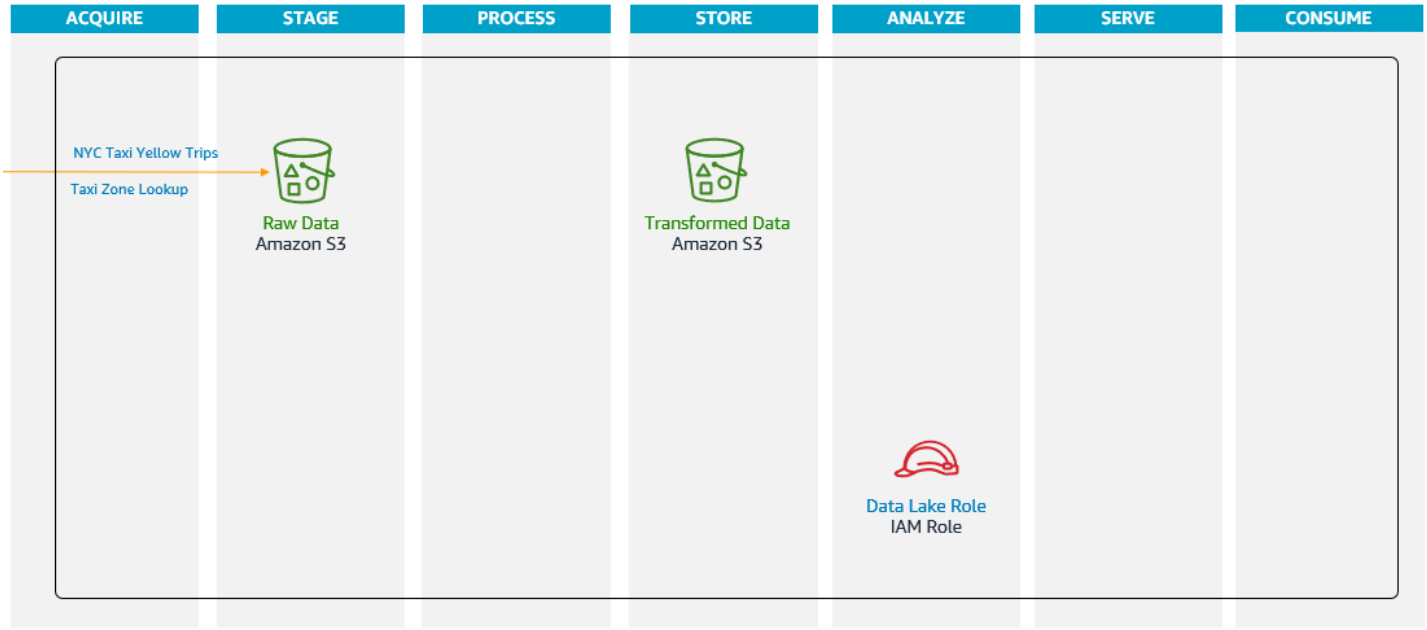
AWS Glue and Athena

Building a Serverless Data Lake

Overview

In preparation for the workshop, we will create Amazon S3 buckets, copy sample data files to Amazon S3, and create an IAM service role.



What is a data lake?

A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale. You can store your data as-is, without having to first structure the data, and run different types of analytics—from dashboards and visualizations to big data processing, real-time analytics, and machine learning to guide better decisions.

Introducing Amazon S3

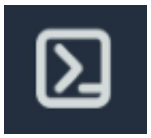
Amazon Simple Storage Service (S3) is the largest and most performant object storage service for structured and unstructured data and the storage service of choice to build a data lake. With Amazon S3, you can cost-effectively build and scale a data lake of any size in a secure environment where data is protected by 99.999999999% (11 9s) of durability.

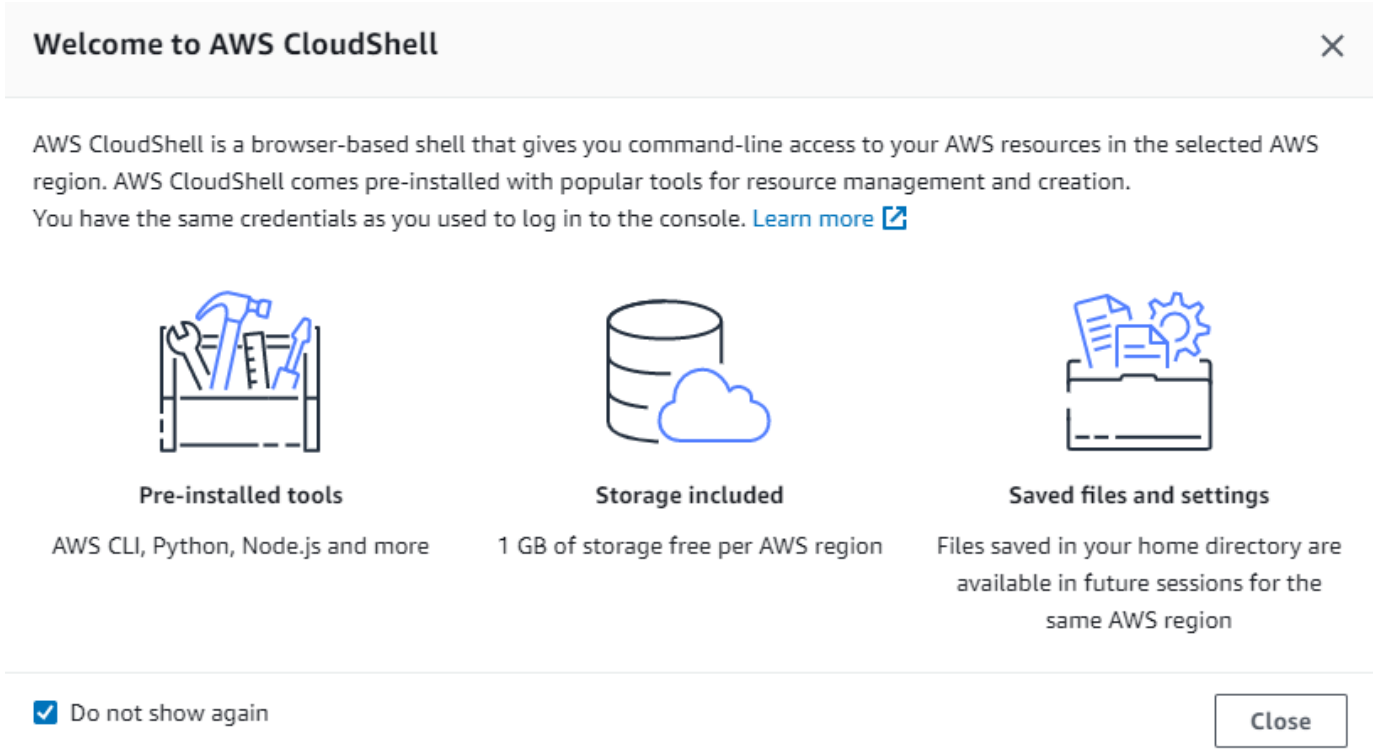
With data lakes built on Amazon S3, you can use native AWS services to run big data analytics, artificial intelligence (AI) and Machine Learning (ML) to gain insights from your unstructured data sets. Data can be collected from multiple sources and moved into the data lake in its original format. It can be accessed or processed with your choice of purpose-built AWS analytics tools and frameworks. Making it easy and quick to run analytics without the need to move your data to a separate analytics system.

Start an AWS CloudShell environment

> You will build the solution from ground up instead of using a CloudFormation template to provision the base resources. We will use the Command Line Interface (CLI) to accomplish this.

AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. You can run AWS CLI commands against AWS services using your preferred shell (Bash, PowerShell, or Z shell). And you can do this without needing to download or install command line tools.

1. Login to the AWS Management Console.
2. In the upper right menu section, select **N. Virginia (us-east-1)** region.
3. Start an AWS CloudShell session by clicking the CloudShell icon on the top menu bar. 
4. A welcome dialog box will pop up, check **do not show this again**, and then click **Close**.



5. CloudShell will start preparing a Linux environment and you should see a terminal in your browser in a few moment.

Create S3 buckets and upload data

Create an S3 bucket to store raw data and upload dataset

An **S3 bucket** is a storage resource that act as a container for objects stored in Amazon S3. You can review the naming rules before creating your buckets.

1. In the CloudShell terminal, run the following command to create a unique S3 bucket.

```
aws s3api create-bucket --bucket [your-bucket-name]-raw --region us-east-1
```

Upload the datasets to your Amazon S3 bucket

We will use the New York City Taxi and Limousine Commission (TLC) Trip Record Data. You can find more information about the dataset in the Registry of Open Data on AWS and by visiting the New York City Taxi and Limousine Commission (TLC) data record page

2. Run the following command to upload the yellow taxi trip data dataset to the S3 bucket you just created.

```
aws s3 cp s3://nyc-tlc/csv_backup/ s3://[your-bucket-name]-raw/nyc-taxi/yellow-tripdata/
--exclude "*" --include "yellow_tripdata_2020*" --recursive
```

3. Run the following command to upload the taxi zone lookup table to the same S3 bucket.

```
aws s3 cp "s3://nyc-tlc/misc/taxi _zone_lookup.csv" s3://[your-bucket-name]-raw/nyc-
taxi/taxi_zone_lookup/taxi_zone_lookup.csv
```

4. Run the following command to list the files you uploaded.

```
aws s3 ls s3://[your-bucket-name]-raw --recursive
```

Create another Amazon S3 bucket to store transformed data

1. Run the following command to create another unique S3 bucket. We will use this bucket to store transformed data.

```
aws s3api create-bucket --bucket [your-bucket-name]-transformed --region us-east-1
```

Create an IAM Service Role

Create an IAM Service role for AWS Glue

> **IAM Service Role** defines a set of permissions for making service requests that a trusted entity, an AWS service, can assume and perform.

1. In the CloudShell terminal, run the following command to download and install Nano Text Editor. When prompted with "Is this ok", type `y`.

   ```
   sudo yum install nano
   ```

2. Start nano to create a file.

   ```
   nano trust-policy.json
   ```

3. In the editor, paste the following content. Type **Ctrl-X**, type **Y** and then **ENTER** to save and exit.

   ```
   {
      "Version":"2012-10-17",
      "Statement":[
         {
            "Effect":"Allow",
            "Principal":{
               "Service":"glue.amazonaws.com"
            },
            "Action":"sts:AssumeRole"
         }
      ]
   }
   ```

4. Create another file using nano editor.

```
nano s3access-policy.json
```

5. In the editor, paste the following content and update the bucket name. Type **Ctrl-X**, type **Y** and then **ENTER** to save and exit.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::[your-bucket]-raw/*",
                "arn:aws:s3:::[your-bucket]-raw",
                "arn:aws:s3:::[your-bucket]-transformed/*",
                "arn:aws:s3:::[your-bucket]-transformed"
            ]
        }
    ]
}
```

6. In the CloudShell terminal, run the following commands to create an IAM role and attach permissions.

```
aws iam create-role --role-name AWSGlueServiceRole-SDL-Jumpstart --assume-role-policy-document file://trust-policy.json
```

```
aws iam put-role-policy --role-name AWSGlueServiceRole-SDL-Jumpstart --policy-name Glue-SDLS3Access --policy-document file://s3access-policy.json
```

```
aws iam attach-role-policy --role-name AWSGlueServiceRole-SDL-Jumpstart --policy-arn arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole
```

7. To check if the IAM role was successfully created, run the following commands.

```
aws iam get-role --role-name AWSGlueServiceRole-SDL-Jumpstart
```

```
aws iam  list-role-policies --role-name AWSGlueServiceRole-SDL-Jumpstart
```

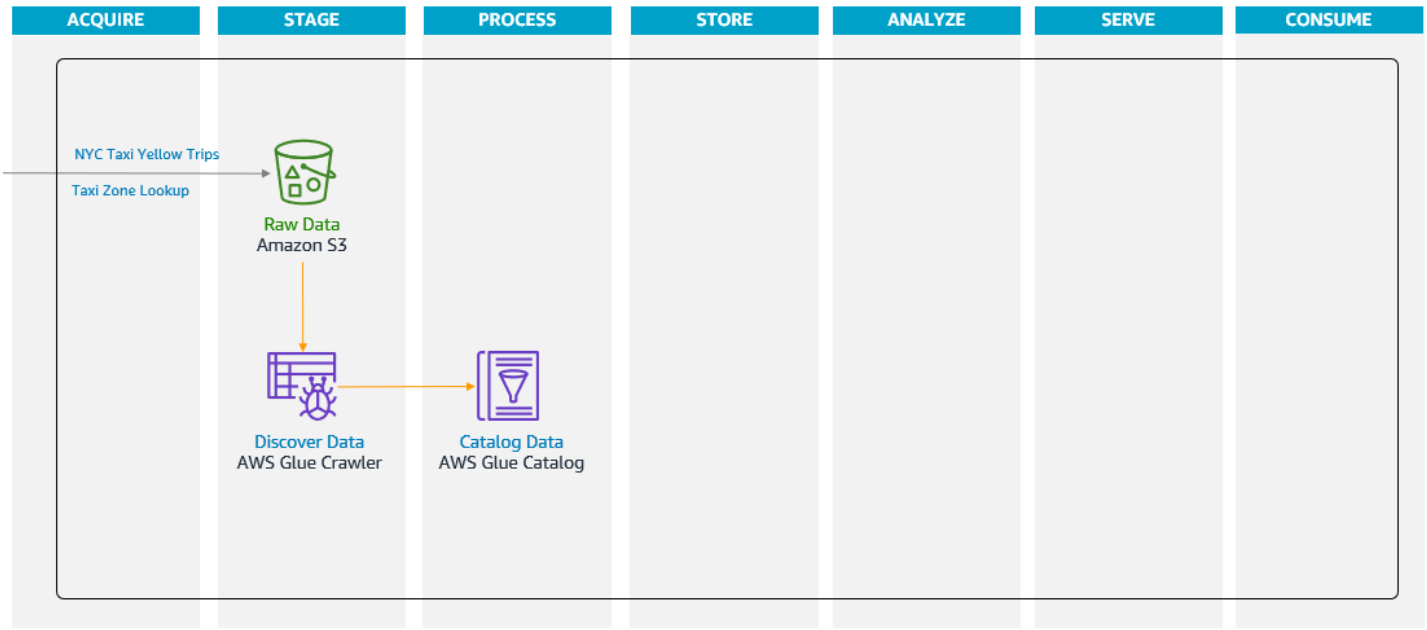
```
aws iam  list-attached-role-policies --role-name AWSGlueServiceRole-SDL-Jumpstart
```

It is a best practice to follow the **principle of least privilege**. Grant only permissions required to perform a task. Determine what users and/or roles need to do and then craft policies that allow them to perform only those tasks.

Lab 1 - Discovering and Cataloging your Data

With an array of data sources and formats in your data lake, it's important to have the ability to discover and catalog it order to better understand the data that you have and at the same time enable integration with other purpose-built AWS analytics.

In this lab, we will create an AWS Glue Crawlers to auto discover the schema of the data stored in Amazon S3. The discovered information about the data stored in S3 will be registered in the AWS Glue Catalog. This allows AWS Glue to use the information stored in the catalog for ETL processing. It also allows other AWS services like Amazon Athena to run queries on the data stored in Amazon S3.



Introducing AWS Glue

AWS Glue is a fully managed serverless extract, transform, and load (ETL) service that makes it simple and cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores. AWS Glue consists of the following core components:

- **Data Catalog** – a central repository to store structural and operational metadata of your data assets.

- **ETL Engine** – automatically generate Scala or Python code.
- **Jobs System** – provides managed infrastructure to orchestrate your ETL workflow.

AWS Glue also includes additional components like:

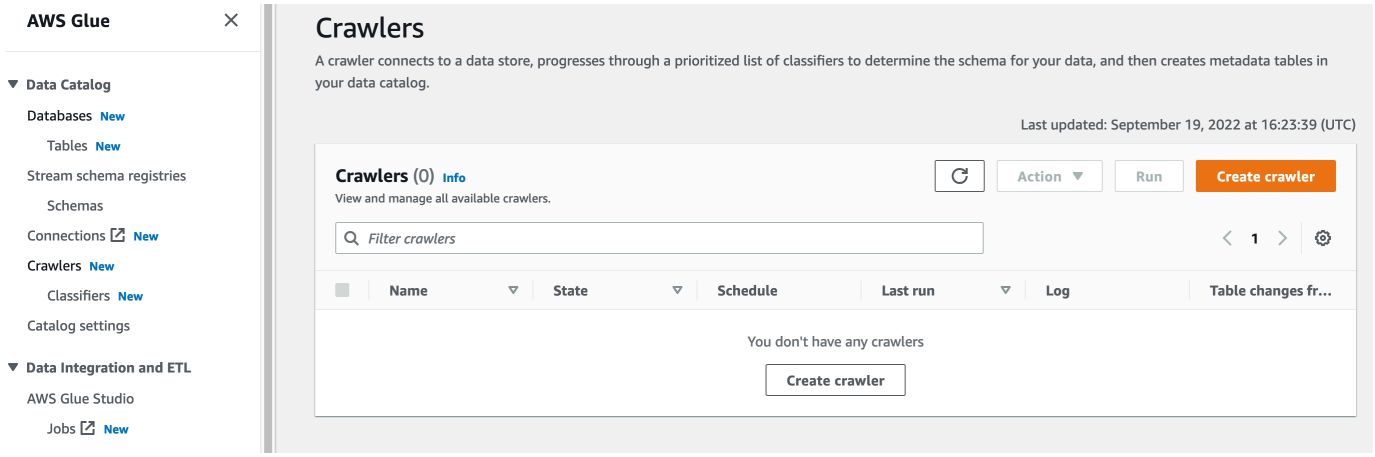
- AWS Glue DataBrew - A visual data preparation tool that makes it easy for data analysts and data scientists to prepare data with an interactive, point-and-click visual interface without writing code.
- AWS Glue Elastic Views - Build materialized views that combine and replicate data across multiple data stores without you having to write custom code.

Together, these automate much of the undifferentiated heavy lifting involved with discovering, categorizing, cleaning, enriching, and moving data, so you can spend more time analyzing your data.

Create a Glue Crawler

> **Glue Crawler** is a feature that automatically infer database and table schema from your source data then stores the associated metadata in the AWS Glue Data Catalog.

1. Go to the AWS Glue Console.
2. In the left navigation menu, click **Crawlers**.
3. On the Crawlers page, click **Create a crawler**.

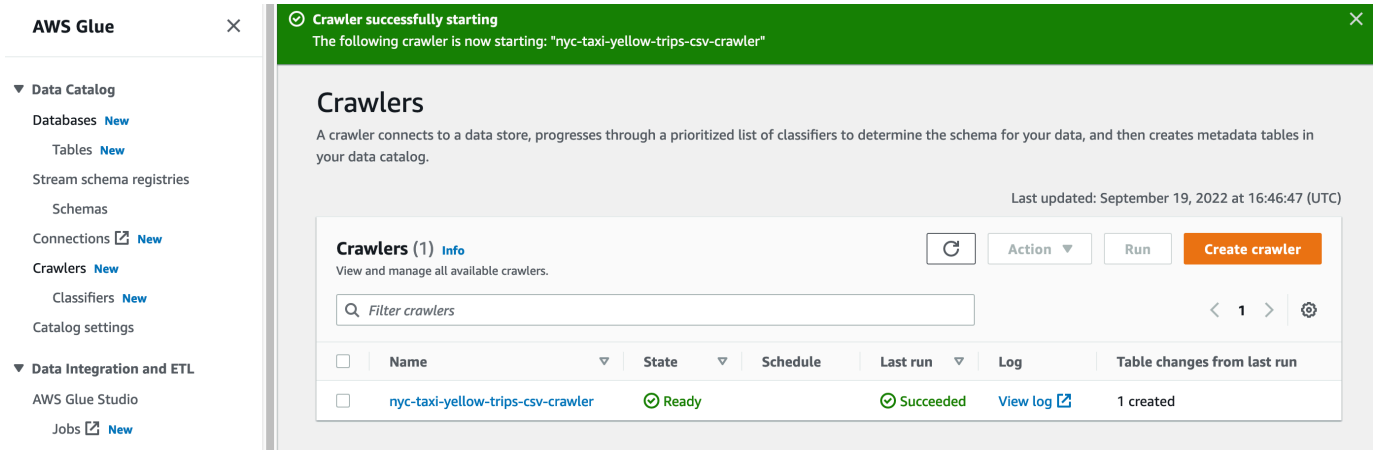


4. Specify `nyc-taxi-yellow-trips-csv-crawler` as the crawler name, click **Next**.
5. On the Choose data sources and classifiers screen, specify the following information, and then click **Next**.
    - Click Add a data source
    - Choose a Data source – **S3**
    - Select Location of S3 data - **In this account**
    - Include S3 path – `s3://[your-bucket-name]-raw/nyc-taxi/yellow-tripdata`
    - For Subsequent crawler runs, select to **Crawl all sub-folders**
    - Then click **Add an S3 data source**.

6. On Configure security settings, choose **AWSGlueServiceRole-SDL-Jumpstart** from the Existing IAM role, click **Next**.
7. On the Set output and scheduling screen, click **Add database**.
8. Specify `nyctaxi_db` as the unique database name, and then click **Create database**.
9. Go back to the previous tab (Set output and scheduling screen), refresh the selection for Target database and choose the newly created database `nyctaxi_db`.
10. Specify `raw_` in the Table name prefix - optional field.
11. On the Crawler schedule, leave the frequency **On demand**, click **Next**.
12. Review the crawler details, click **Create crawler**.

Run the Glue Crawler

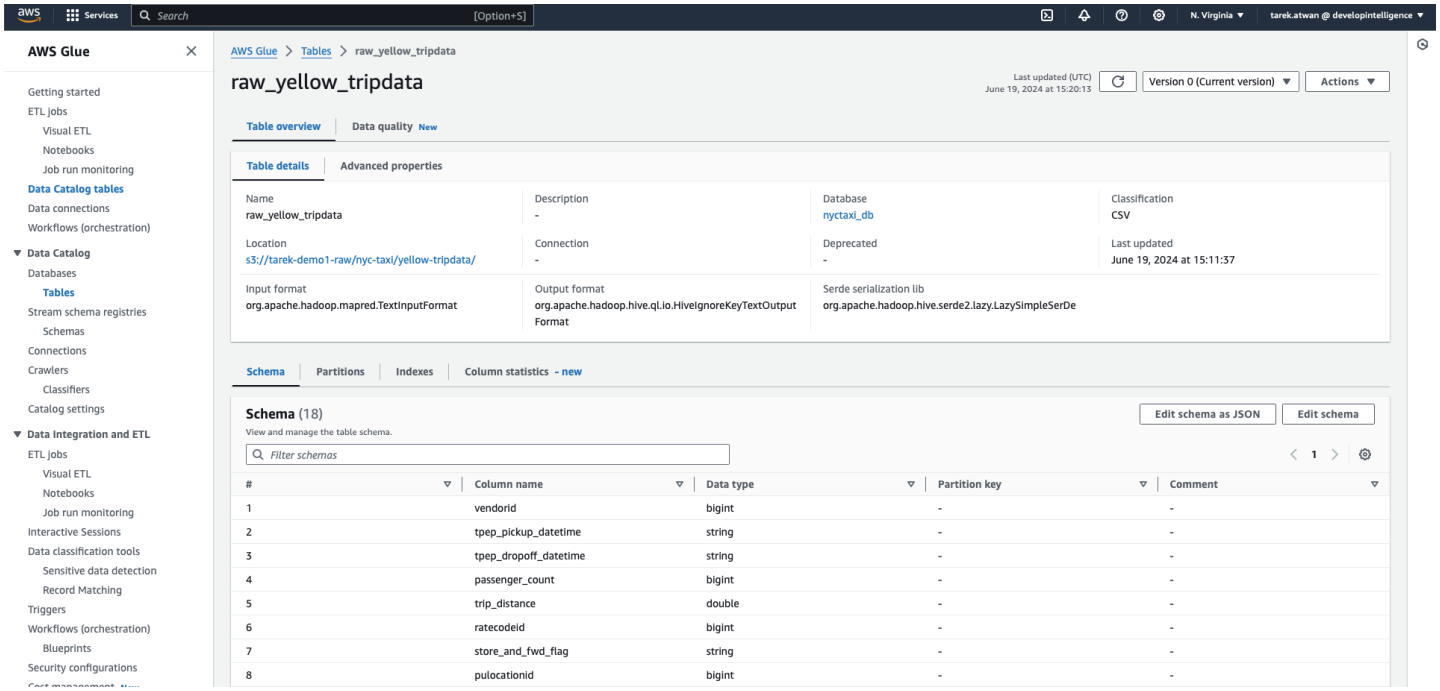
1. On the Crawlers page, select **nyc-taxi-yellow-trips-csv-crawler**, and then click **Run crawler**.
2. Upon successful completion of the crawler, you should see a value of **1** in the Tables added column.



Review the metadata in Glue Data Catalog

Glue Data Catalog contain references to data that is used as sources and targets of your extract, transform, and load (ETL) jobs in AWS Glue. It also maintain a unified view of the data and enables various AWS services such as Athena, EMR, and Redshift Spectrum to access it.

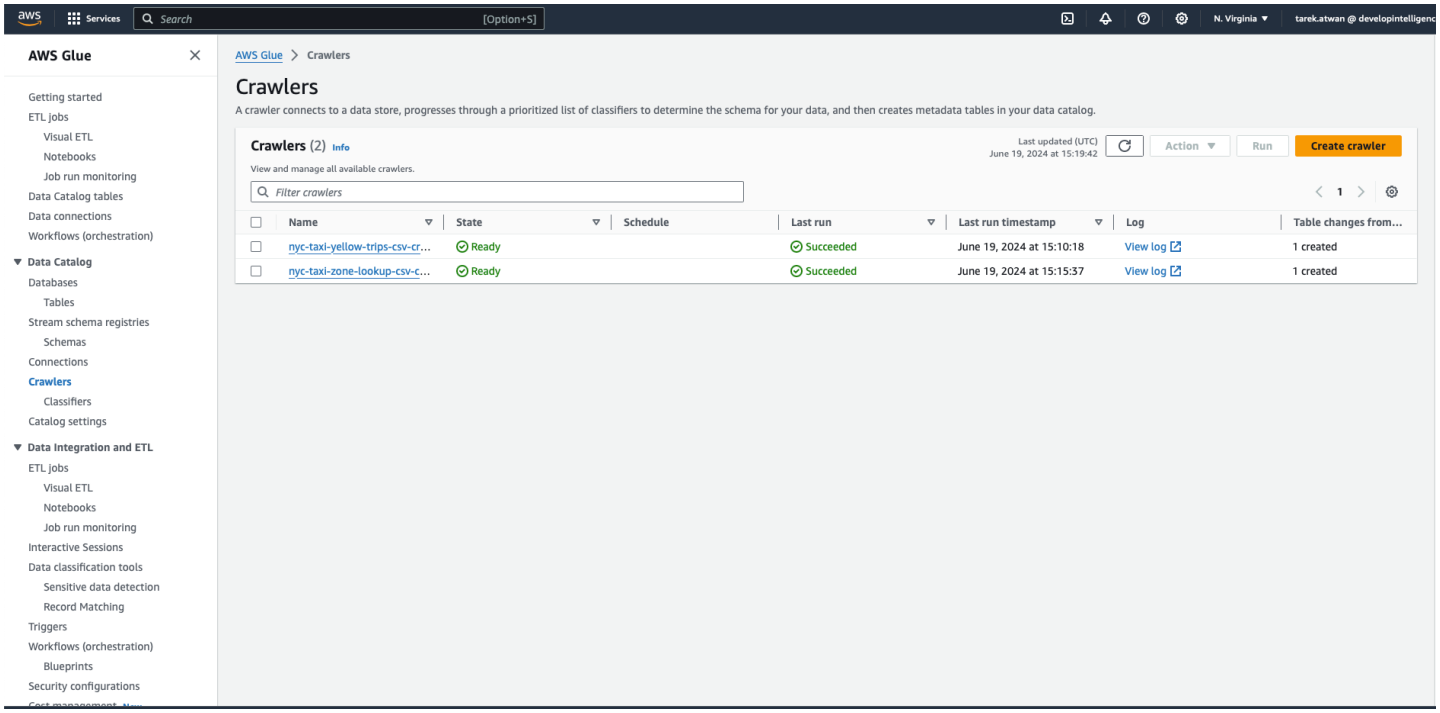
1. In the left navigation menu, click **Tables**.
2. On the Tables page, click on the name **raw_yellow_tripdata** to review the table metadata and schema information.

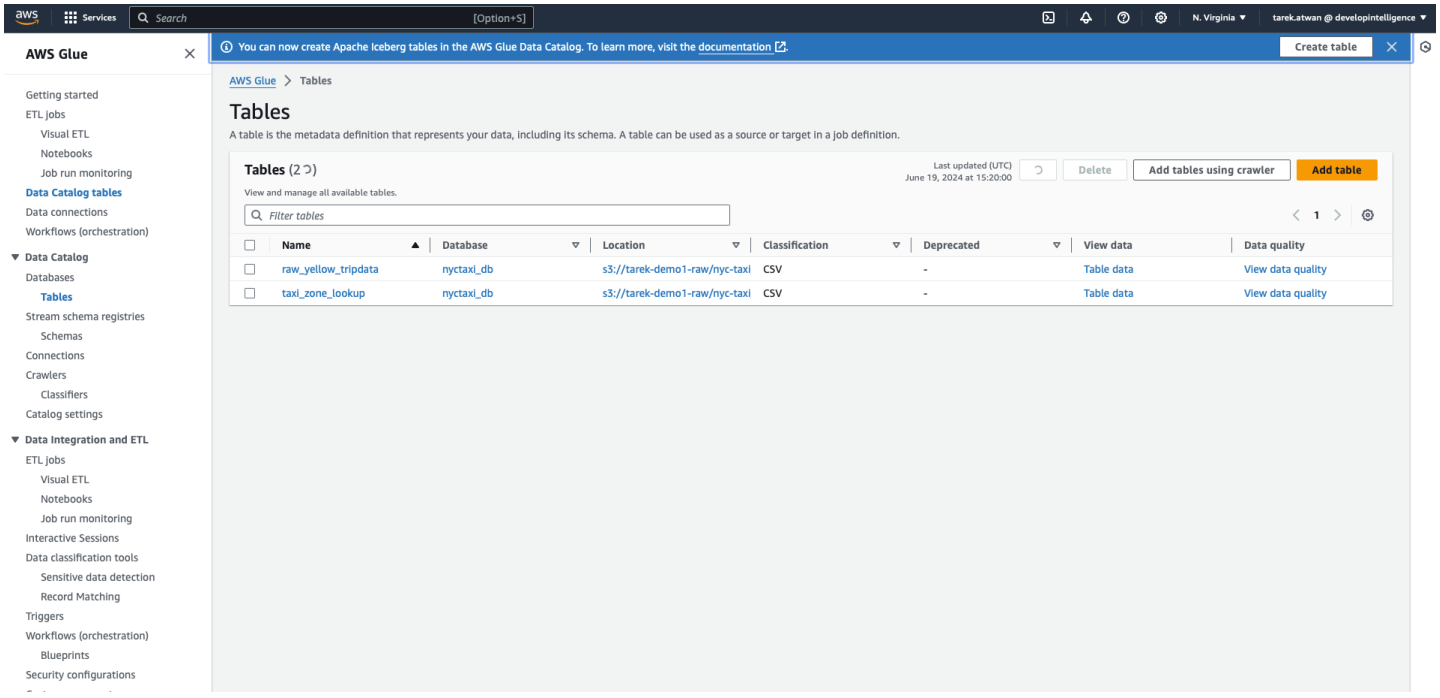


Create another Glue Crawler

Repeat the same steps to "crawl" the taxi zone lookup data stored in S3

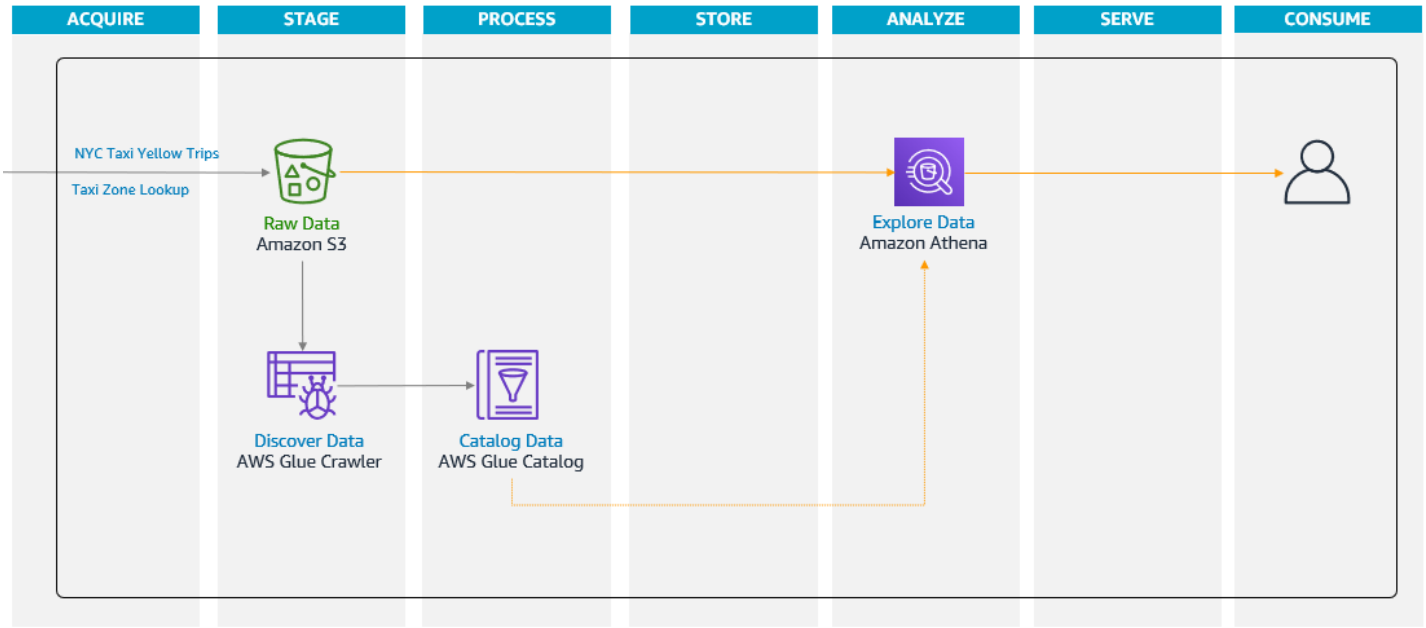
1. In the left navigation menu, click **Crawlers**.
2. On the Crawlers page, click **add crawler**.
3. Specify `nyc-taxi-zone-lookup-csv-crawler` as the crawler name, click **Next**.
4. On the Choose data sources and classifiers screen, specify the following information, and then click Next.
    - Click Add a data source
    - Choose Data source - **S3**
    - Select Location of S3 data - **In this account**
    - Include S3 path - `s3://[your-bucket-name]-raw/nyc-taxi/taxi_zone_lookup`
    - For Subsequent crawler runs, select to **Crawl all sub-folders**
    - Then click **Add an S3 data source**.
5. On the Configure security settings, choose **AWSGlueServiceRole-SDL-Jumpstart** from the Existing IAM role, click **Next**.
6. On the Set output and scheduling screen, choose **nyctaxi_db** as the database.
7. On the Crawler schedule, leave the frequency **On demand**, click **Next**.
8. Review the crawler details, click **Create crawler**.
9. On the Crawlers page, select **nyc-taxi-zone-lookup-csv-crawler**, and then click **Run**
10. A new table, `taxi_zone_lookup`, will be created in the Glue Catalog after the crawler successfully completes. Review the metadata and schema information of the table.





Lab 2 - Exploring your Data

In this lab, we will use Amazon Athena to explore our data and identify data quality issues. We will also learn how to update table properties in AWS Glue Catalog.



Introducing Amazon Athena

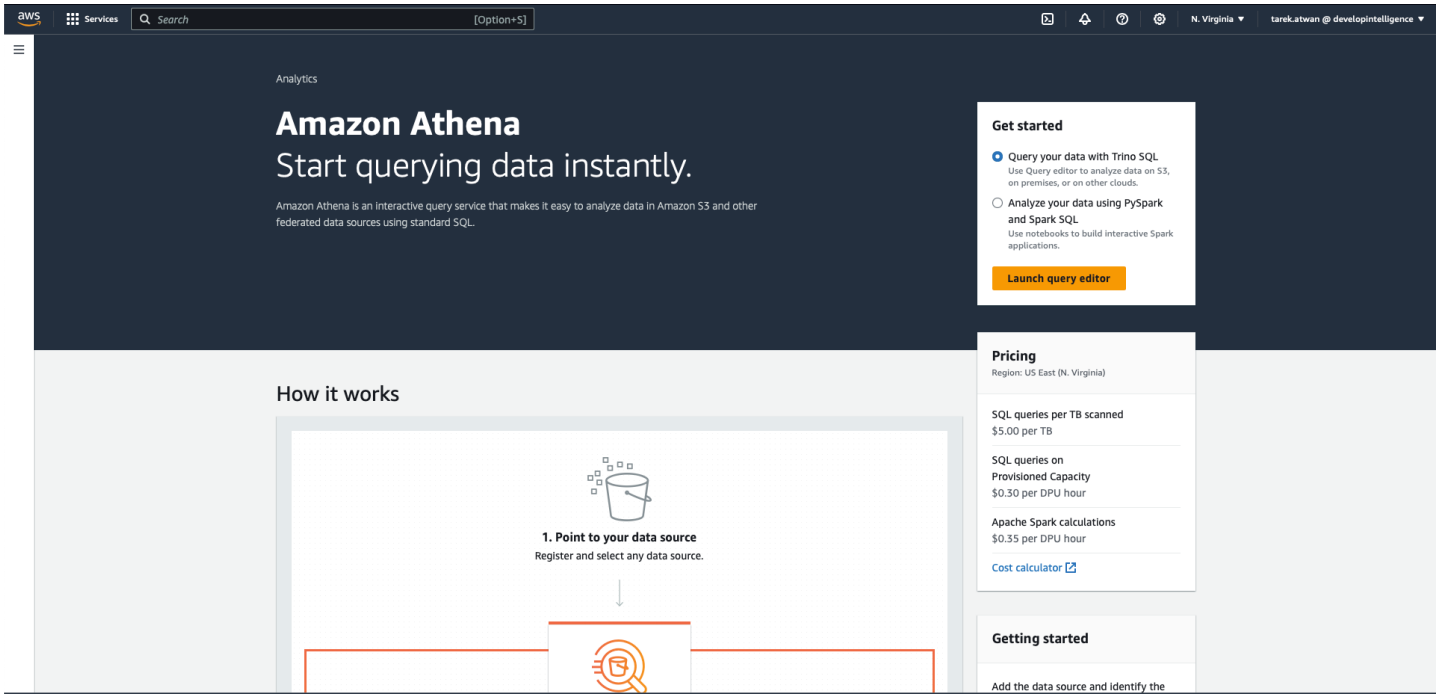
Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to setup or manage, and you can start analyzing data immediately. You don't even need to load your data into Athena, it works directly with data stored in S3.

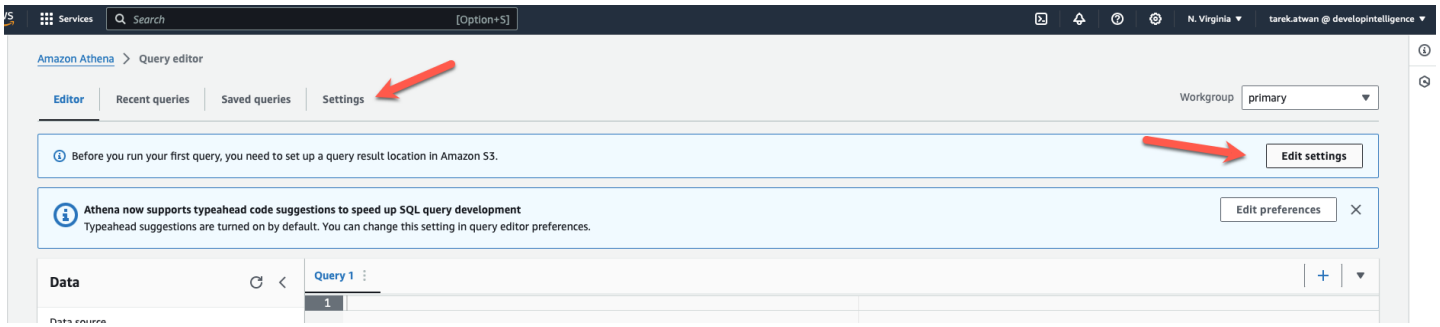
> Amazon Athena automatically stores query results and metadata information for each query that runs in a query result location that you can specify in Amazon S3. If necessary, you can access the files in this location to work with them. You can also download query result files directly from the Athena console.

1. Launch a CloudShell terminal, and then run the following command to create an S3 bucket to store Athena's query results.

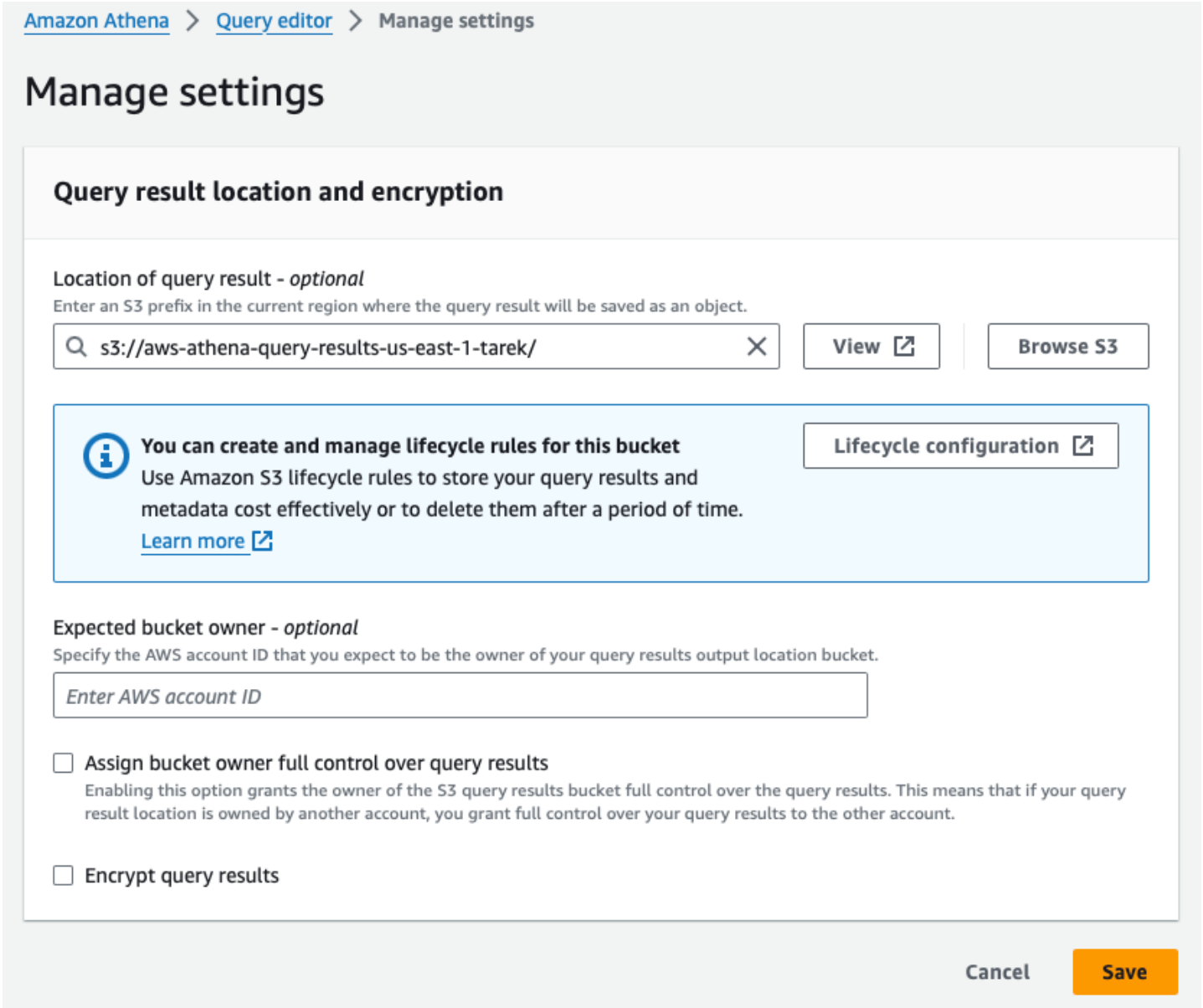
```
aws s3api create-bucket --bucket aws-athena-query-results-us-east-1-[Account-ID] --region us-east-1
```

2. Go to Athena console. On the **Get Started** click **Launch Query Editor** or just select Query Editor from the left side panel/menu.



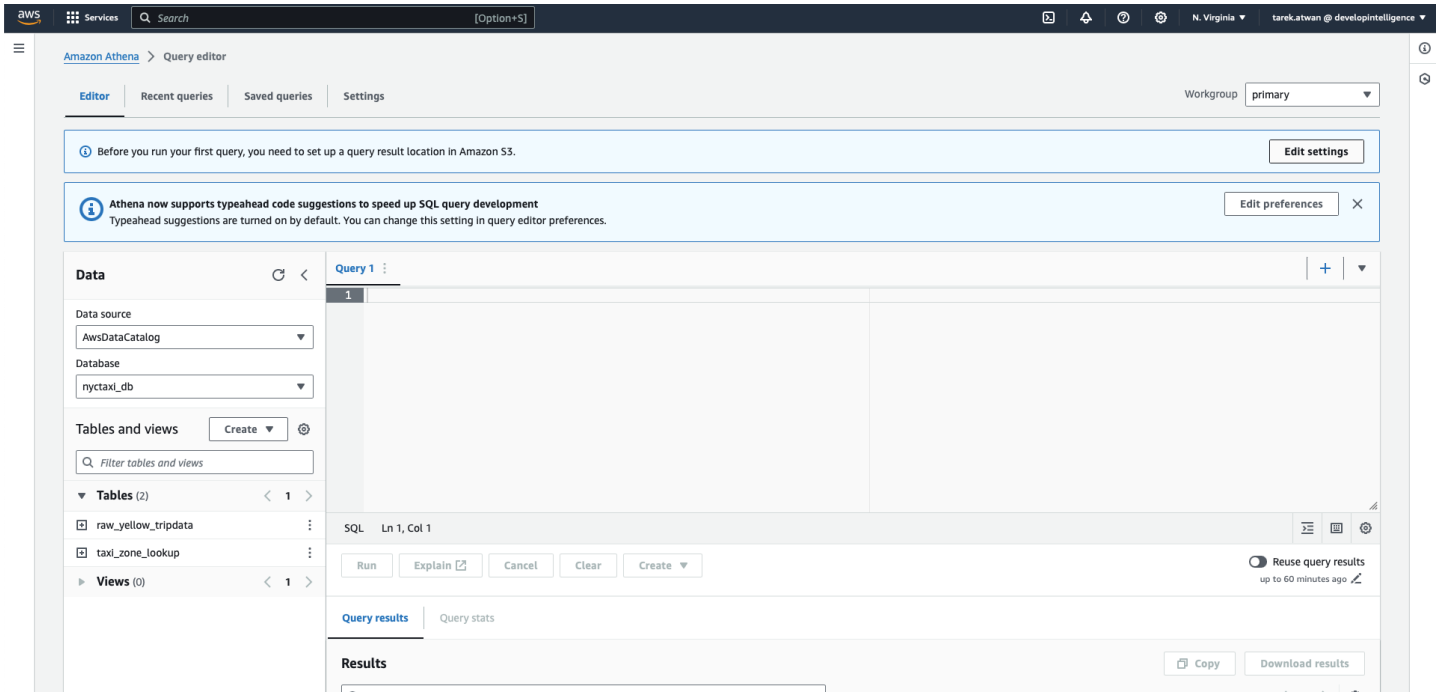


3. Choose **Settings**, click on **Browse S3** and select **aws-athena-query-results-us-east-1-[Account-ID]** as the value for the Location of query result - optional field.



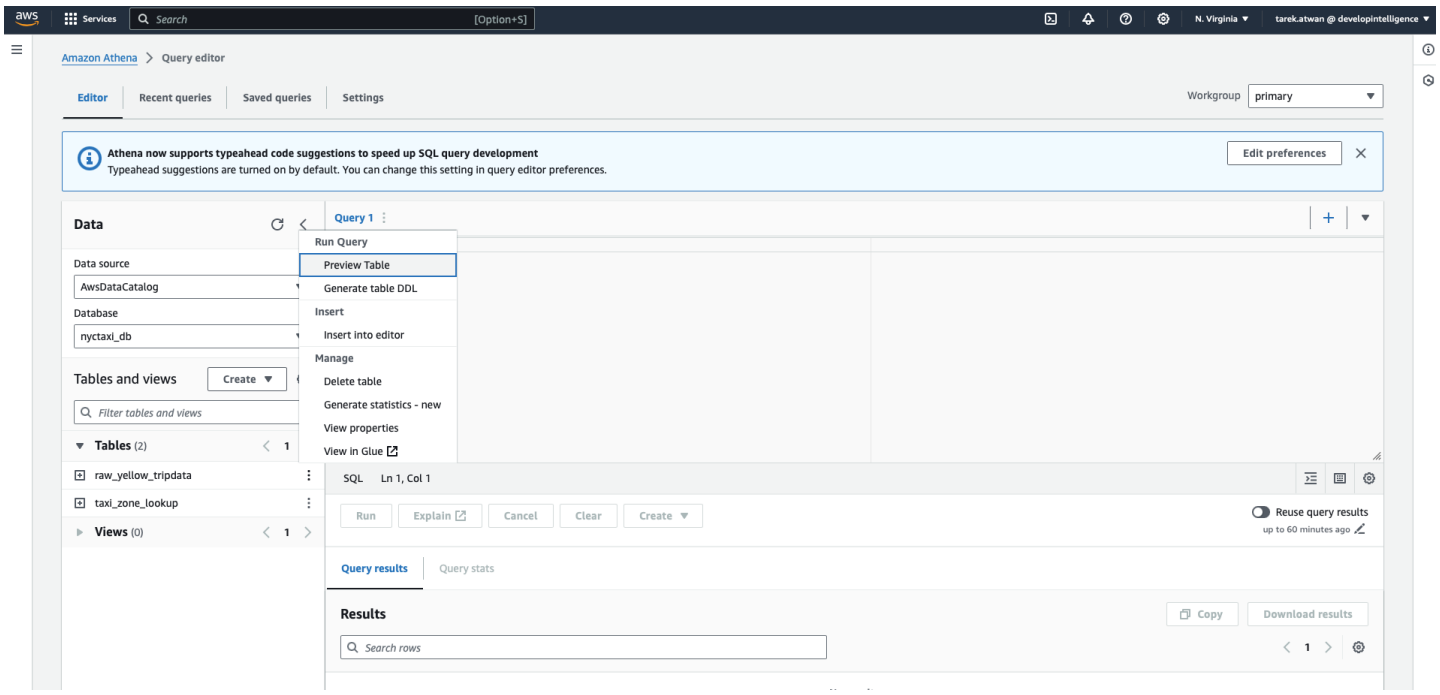
4. Go to the bottom of the page, click **Save**.

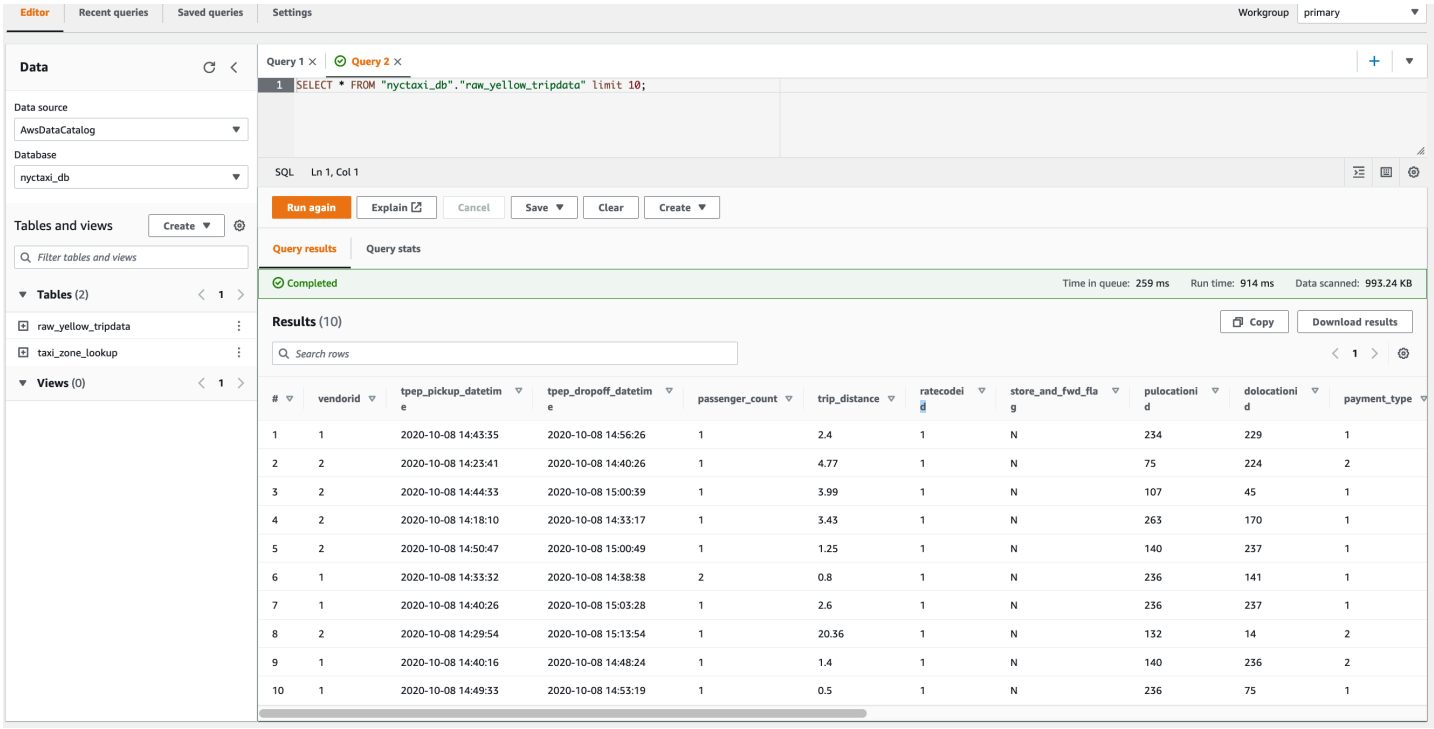
5. Go to the top menu, click on **Editor** to return back to the Query editor page.



Preview Table Data

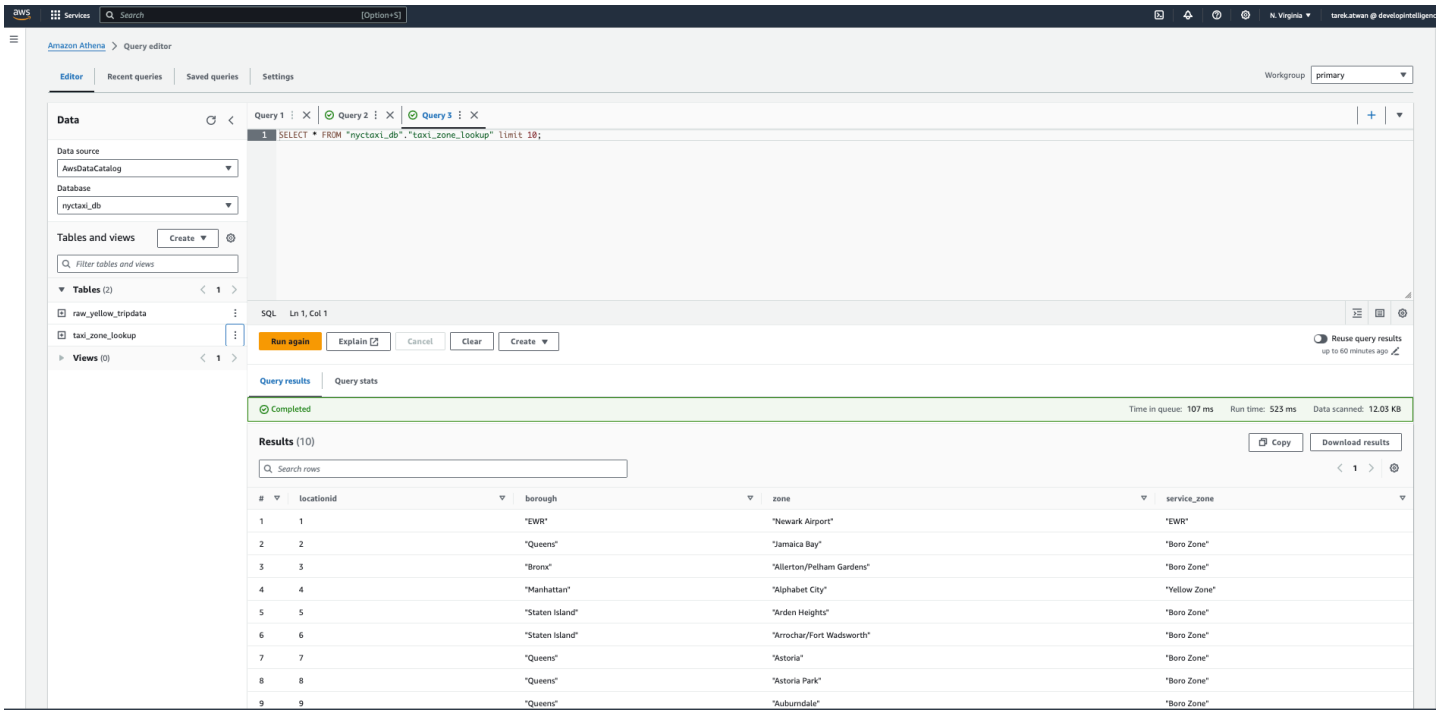
1. On the Query editor page, click on the actions menu icon ⋮ besides **raw_yellow_tripdata**, and then select **Preview table**.
2. Examine the data.





#	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	ratecodeid	store_and_fwd_flag	pulocationid	dolocationid	payment_type
1	1	2020-10-08 14:43:35	2020-10-08 14:56:26	1	2.4	1	N	234	229	1
2	2	2020-10-08 14:23:41	2020-10-08 14:40:26	1	4.77	1	N	75	224	2
3	2	2020-10-08 14:44:33	2020-10-08 15:00:39	1	3.99	1	N	107	45	1
4	2	2020-10-08 14:18:10	2020-10-08 14:33:17	1	3.43	1	N	263	170	1
5	2	2020-10-08 14:50:47	2020-10-08 15:00:49	1	1.25	1	N	140	237	1
6	1	2020-10-08 14:33:32	2020-10-08 14:38:38	2	0.8	1	N	236	141	1
7	1	2020-10-08 14:40:26	2020-10-08 15:03:28	1	2.6	1	N	236	237	1
8	2	2020-10-08 14:29:54	2020-10-08 15:13:54	1	20.36	1	N	132	14	2
9	1	2020-10-08 14:40:16	2020-10-08 14:48:24	1	1.4	1	N	140	236	2
10	1	2020-10-08 14:49:33	2020-10-08 14:53:19	1	0.5	1	N	236	75	1

3. Do the same for **taxi_zone_lookup** table. Observe that all string values are enclosed with **"**.



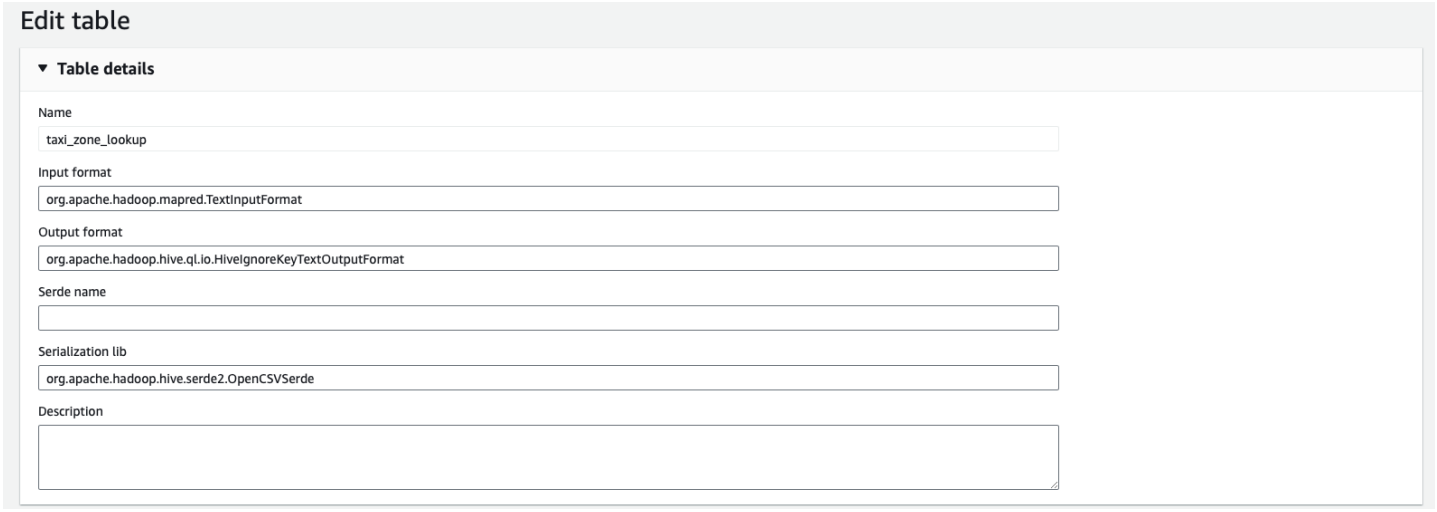
#	locationid	borough	zone	service_zone
1	1	"EWR"	"Newark Airport"	"EWR"
2	2	"Queens"	"Jamaica Bay"	"Boro Zone"
3	3	"Bronx"	"Allerton/Pelham Gardens"	"Boro Zone"
4	4	"Manhattan"	"Alphabet City"	"Yellow Zone"
5	5	"Staten Island"	"Arden Heights"	"Boro Zone"
6	6	"Staten Island"	"Arrochar/Fort Wadsworth"	"Boro Zone"
7	7	"Queens"	"Astoria"	"Boro Zone"
8	8	"Queens"	"Astoria Park"	"Boro Zone"
9	9	"Queens"	"Auburndale"	"Boro Zone"

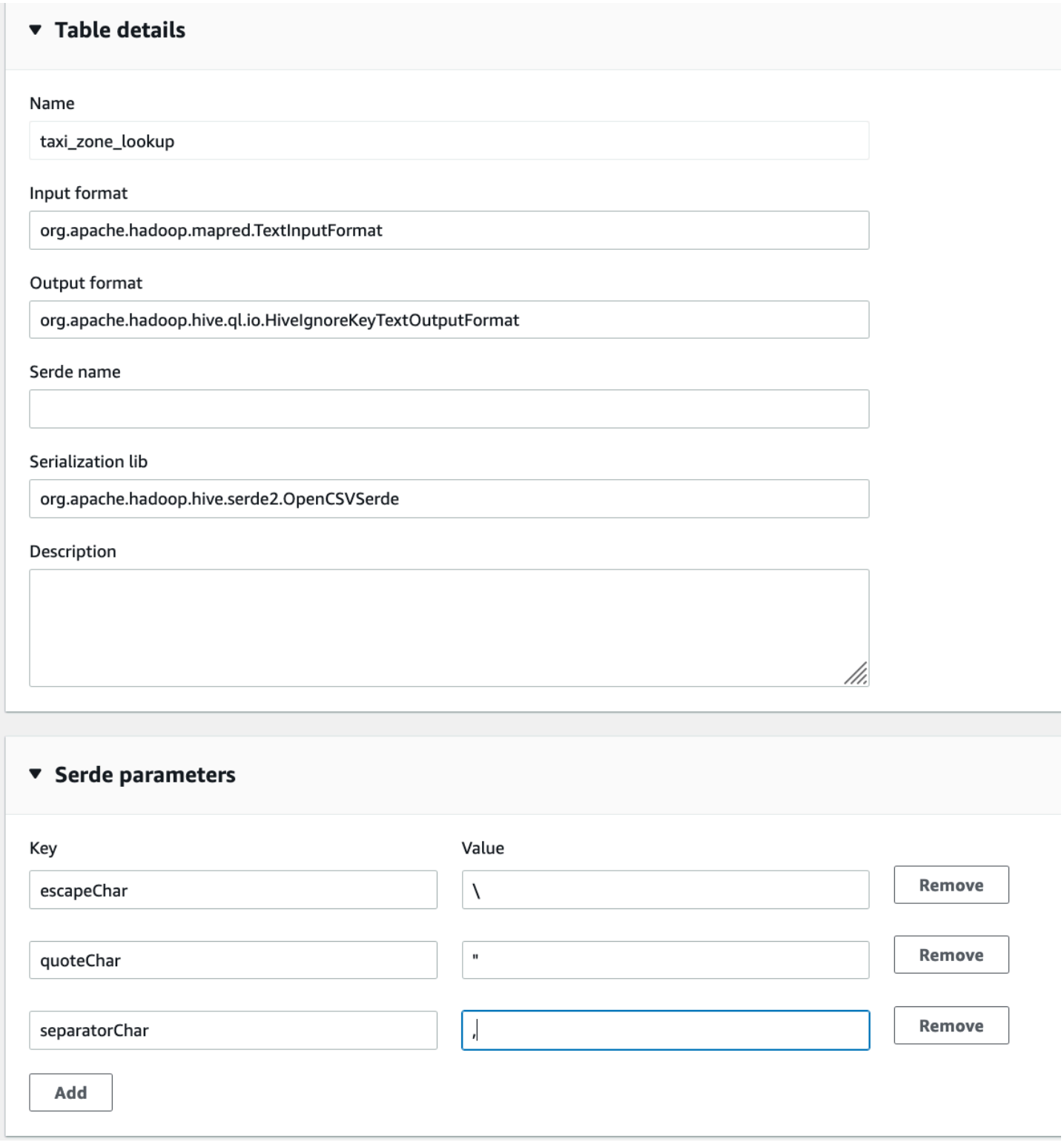
Working with CSV data enclosed in quotes

Working with CSV files enclosed in Quotes

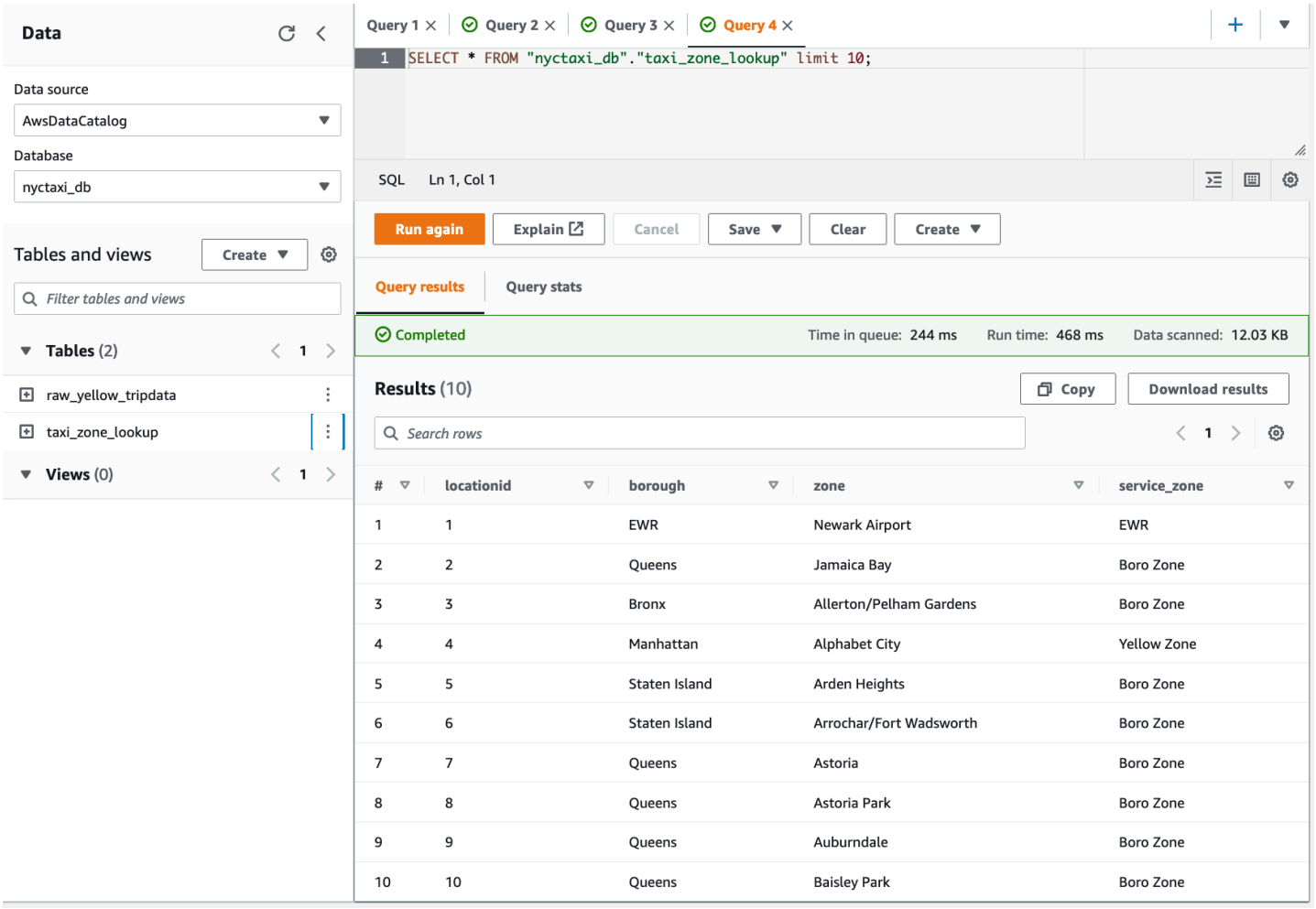
CSV files occasionally have quotes around the data values intended for each column which aren't part of the data to be analyzed. To read the CSV file properly, we can update the table properties in AWS Glue to use the OpenCSVSerDe.

1. Go to Glue console.
2. In the left navigation menu, click Tables under the Data Catalog section.
3. On the Table screen, click on the **taxi_zone_lookup table**.
4. Click **Actions**, then click on **Edit table**.
5. Update the **Serde serialization lib** with `org.apache.hadoop.hive.serde2.OpenCSVSerde`.
6. Remove existing Serde parameters and then add the following:
    - `escapeChar`, enter a backslash `\`
    - `quoteChar`, enter a double quote `"`
    - `separatorChar`, enter a comma `,`





7. Click **Save**
8. Go to back to the Athena console.
9. On the Query editor page, click on the actions menu icon ⋮ besides **taxi_zone_lookup**, and then click **Preview table**.



Run SQL queries to explore the data

1. Check number of yellow taxi trip records.

```sql
-- total records
-- 24648499
SELECT COUNT(*) "Count" FROM raw_yellow_tripdata;
```

2. Explore data categories.

```sql
-- observe NULL values
-- 809568
SELECT vendorid, COUNT(*) "Count"
FROM  raw_yellow_tripdata
GROUP BY vendorid
ORDER BY 1;
```

```sql
-- observe other categories
SELECT pulocationid, COUNT(*) "Count"
FROM   raw_yellow_tripdata
GROUP BY pulocationid
ORDER BY 1;
```

```sql
-- observe NULL values
-- 809568
SELECT payment_type, COUNT(*) "Count"
FROM   raw_yellow_tripdata
GROUP BY payment_type
ORDER BY 1;
```

3. Explore records with NULL Vendor ID.

```sql
-- observe other columns with NULL values
-- passenger_count, ratecodeid, store_and_fwd_flag, payment_type
SELECT *
FROM   raw_yellow_tripdata
WHERE  vendorid IS NULL
LIMIT 100;
```

4. Explore records by time period.

```sql
-- tpep_pickup_datetime is defined as STRING
-- observe record counts that falls outside of the time period
SELECT SUBSTR(tpep_pickup_datetime, 1, 7) "Period", COUNT(*) "Total Records"
FROM   raw_yellow_tripdata
GROUP BY SUBSTR(tpep_pickup_datetime, 1, 7)
ORDER BY 1;
```

5. Count records that falls outside of year 2020.

```sql
-- records with incorrect pickup datetime values
-- 280
SELECT COUNT(*) "Count"
FROM   raw_yellow_tripdata
WHERE  SUBSTR(tpep_pickup_datetime, 1, 7) NOT LIKE '2020%';
```

6. Count records with NULL values (based on Vendor ID) that falls within 2020.

```sql
-- Records with NULL categories like Vendor ID
-- 809568
SELECT COUNT(*) "Count"
FROM   raw_yellow_tripdata
WHERE  vendorid IS NULL
AND    SUBSTR(tpep_pickup_datetime, 1, 7) LIKE '2020%';
```

7. Count records that falls in the last quarter of 2020, exclude records with missing Vendor ID.

```sql
-- Total records in BER months, excluding columns with missing Vendor ID
-- 4347658
SELECT COUNT(*) "Count"
FROM   raw_yellow_tripdata
WHERE  vendorid IS NOT NULL
AND    SUBSTR(tpep_pickup_datetime, 1, 7) LIKE '2020-1%';
```

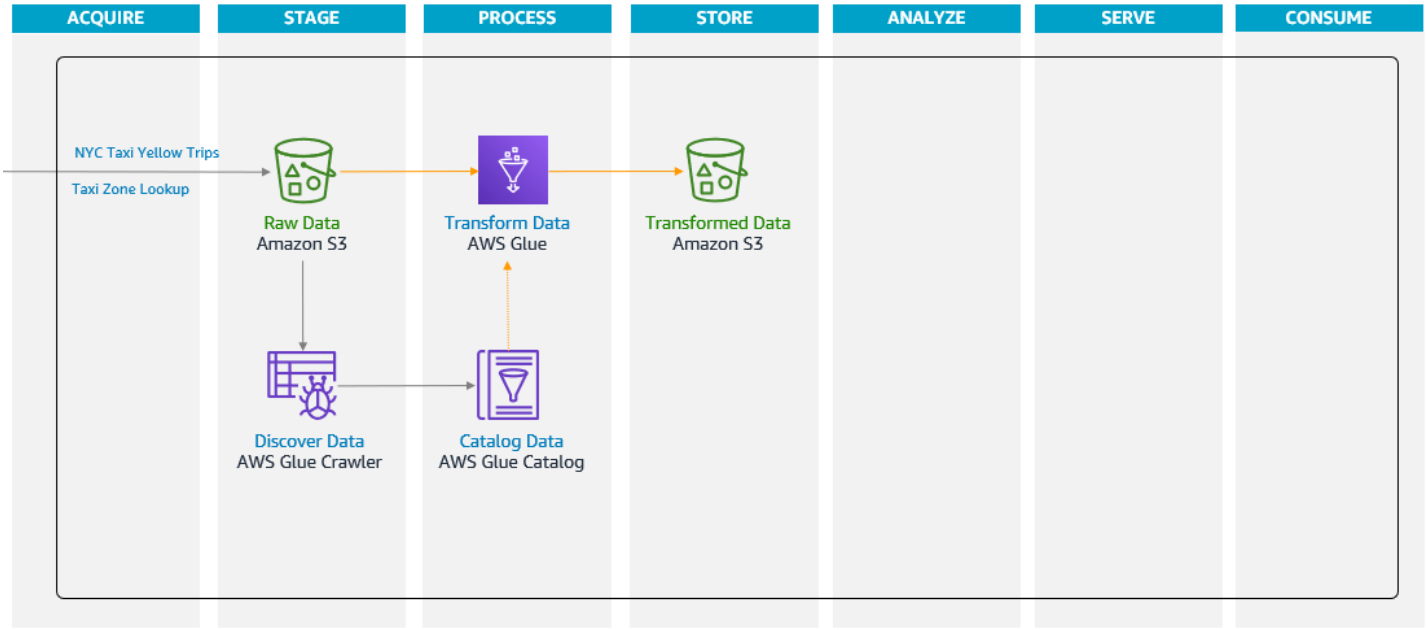
8. Join taxi trips data with taxi zone look up table.

```sql
-- explore data with lookup information
-- observe column names from lookup tables
SELECT td.*, pu.*, do.*
FROM   raw_yellow_tripdata td,
       taxi_zone_lookup pu,
       taxi_zone_lookup do
WHERE  td.pulocationid = pu.locationid AND
       td.pulocationid = do.locationid AND
       vendorid IS NOT NULL AND
       SUBSTR(tpep_pickup_datetime, 1, 7) LIKE '2020-1%'
LIMIT 100;
```

```sql
-- Count total joined records for the last quarter of 2020.
-- 4347658
SELECT COUNT(*) "Count"
FROM   raw_yellow_tripdata td,
       taxi_zone_lookup pu,
       taxi_zone_lookup do
WHERE  td.pulocationid = pu.locationid AND
       td.pulocationid = do.locationid AND
       vendorid IS NOT NULL AND
       SUBSTR(tpep_pickup_datetime, 1, 7) LIKE '2020-1%';
```

Lab 3 - Transforming your Data

Data enrichment is the process of enhancing existing data with additional data from other sources to provide additional context or meaning, it makes data more useful and insightful during analysis.



Introducing AWS Glue Studio

AWS Glue Studio is a graphical interface that makes it easy to create, run, and monitor extract, transform, and load (ETL) jobs in AWS Glue. You can visually compose data transformation workflows, AWS Glue studio will generate Apache Spark code on your behalf, and let it seamlessly run them on AWS Glue's Apache Spark-based serverless ETL engine.

Create a job using Glue Studio

Planning the Data Transformation Steps

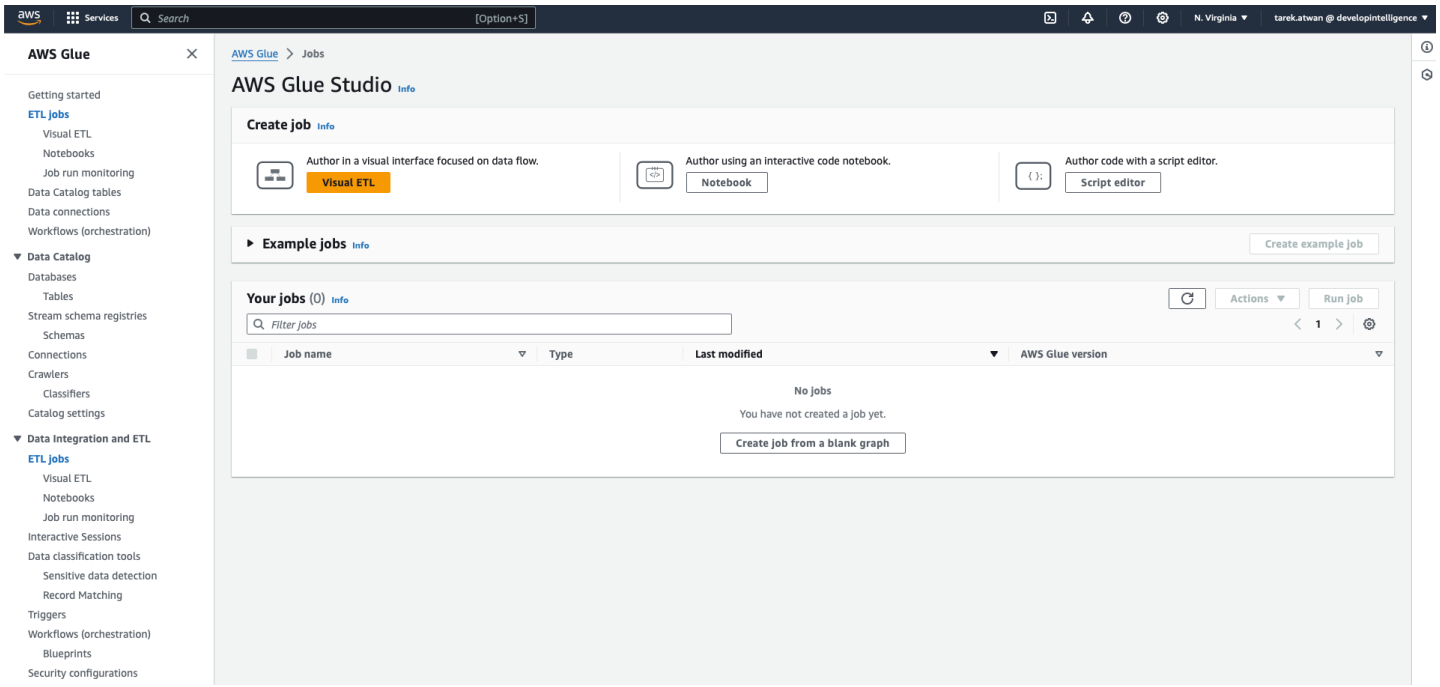
It's a good practice to plan the steps to transform your data. Based on the information we captured during data exploration stage, we can come up with the following transformation step:

1. Read yellow trip data from S3, `raw_yellow_tripdata` table.
2. Clean yellow trip data data.
    - Remove records with NULL values (vendorid, payment_type, passenger count, ratecodeid).
    - Filter records within a time period (remove records with invalid pickup datetime, narrow down data to be processed).
3. Join yellow trip data with taxi zone lookup to obtain pickup location information.
    - Read lookup data from S3, `taxi_zone_lookup` table.
    - Rename column names of lookup data to differentiate pickup locations from drop-off locations.
    - Perform the join.

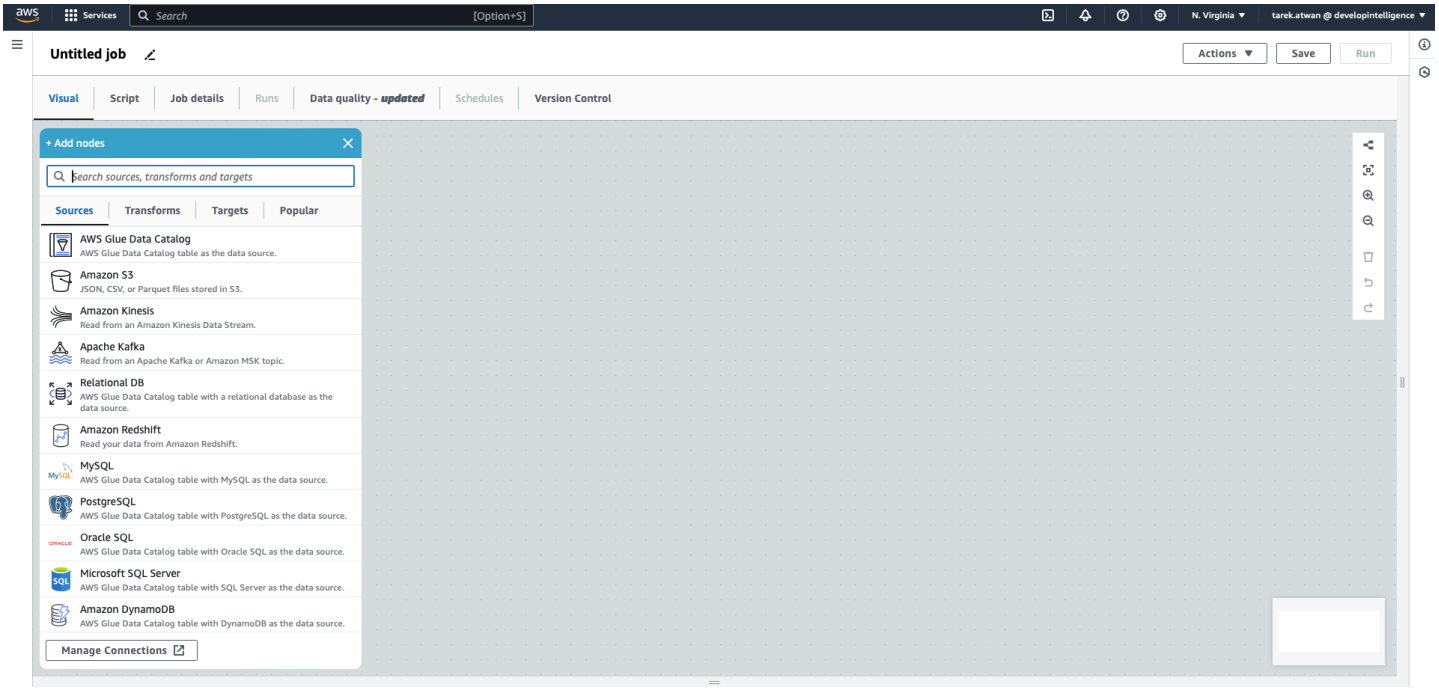
4. Join yellow trip data with taxi zone lookup to obtain drop-off location information.
    - Read lookup data from S3, `taxi_zone_lookup` table.
    - Rename column names of lookup data to differentiate drop-off locations from pickup locations.
    - Perform the join.
5. Perform data transformation on joined dataset.
    - Rename column names.
    - Set appropriate data types.
    - Remove redundant columns as a result of table joins.
6. Save processed dataset to S3 in a query optimized format.

Create a job using Glue Studio

1. Go to Glue console.
2. In the left navigation menu, under the ETL section, click **AWS Glue Studio**.



3. On the AWS Glue Studio page, under **Create Job** click **Visual ETL**.
4. In the Glue Studio editor page, go to the Job details tab, set the following configuration:
    1. Name – `Transform NYC Taxi Trip Data`
    2. IAM Role – **AWSGlueServiceRole-SDL-Jumpstart**
    3. Job bookmark – **Disable**
    4. Number of retries - `0`



Transform NYC Taxi Trip Data

Visual | Script | Job details | Runs | Data quality - *updated* | Schedules | Version Control

Basic properties Info

Name

Transform NYC Taxi Trip Data

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.

AWSGlueServiceRoleSDLJumpstart

Type

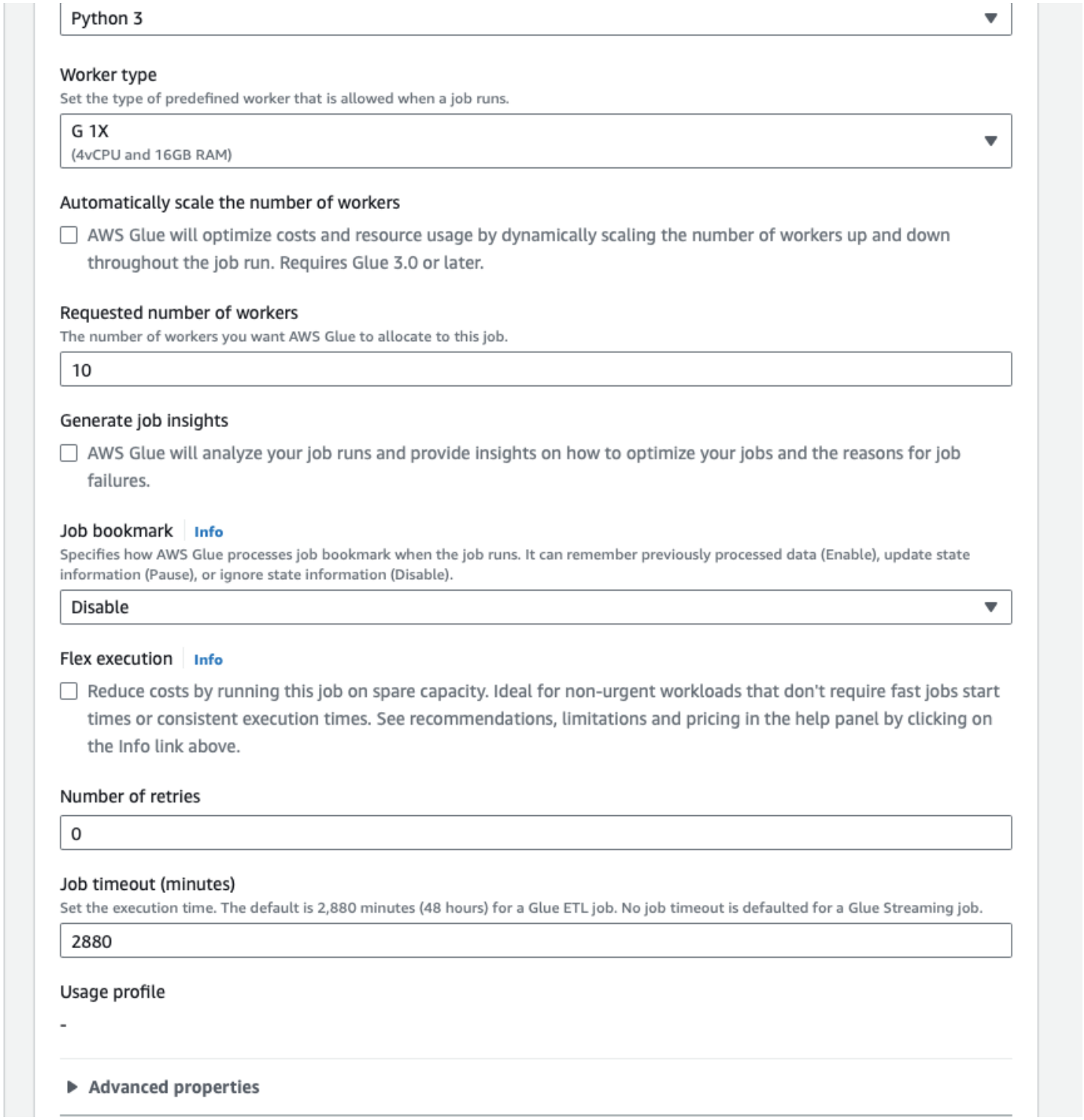
The type of ETL job. This is set automatically based on the types of data sources you have selected.

Spark

Glue version Info

Glue 4.0 - Supports spark 3.3, Scala 2, Python 3

Language



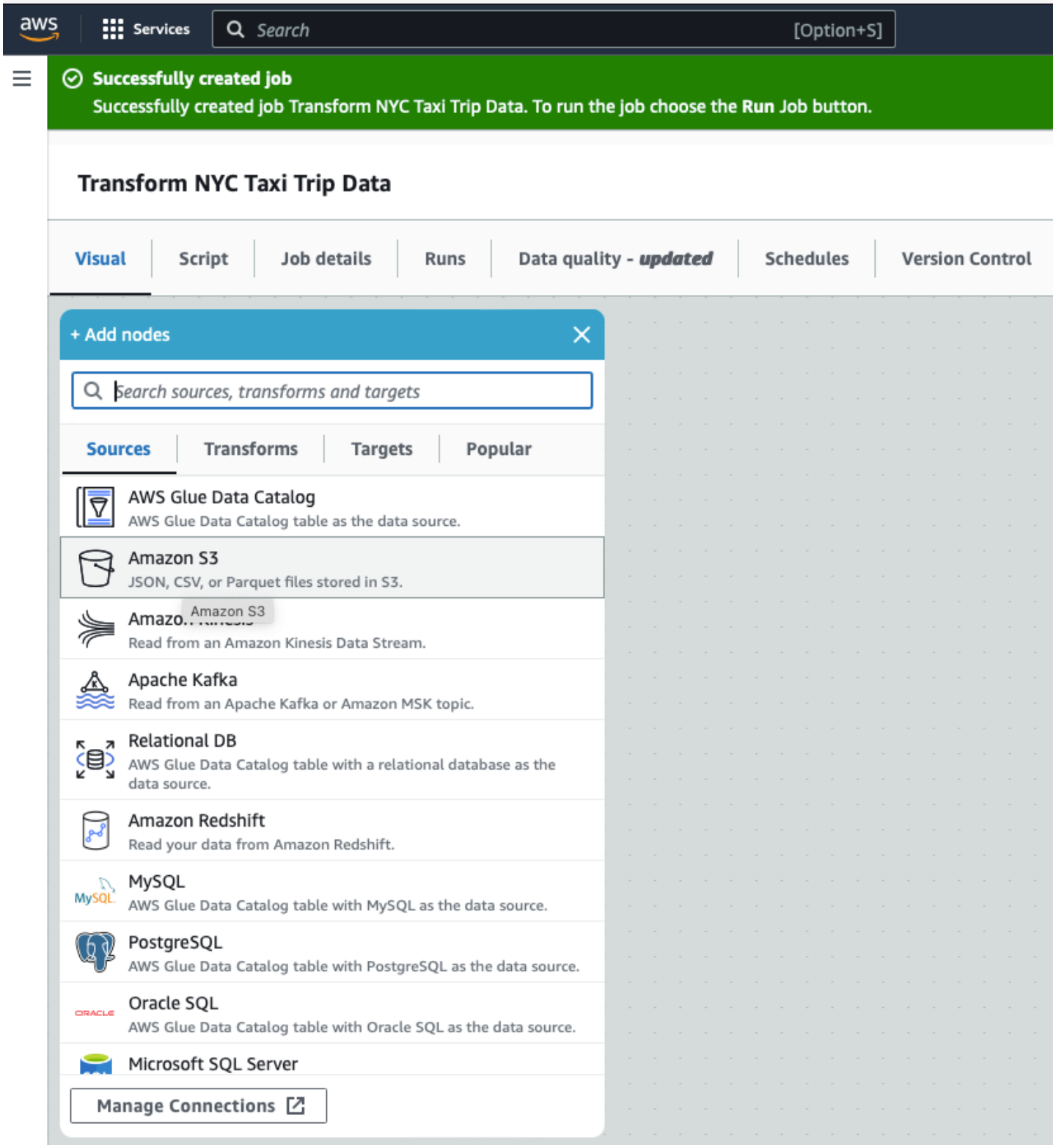
5. Click **Save**. Go back to the Visual tab.

AWS Glue **job bookmark** feature helps maintain state information and prevent the reprocessing of old data. It enables the job to process only new data when rerunning on a scheduled interval.

Add a data source

Adding Yellow Trip data from Amazon S3

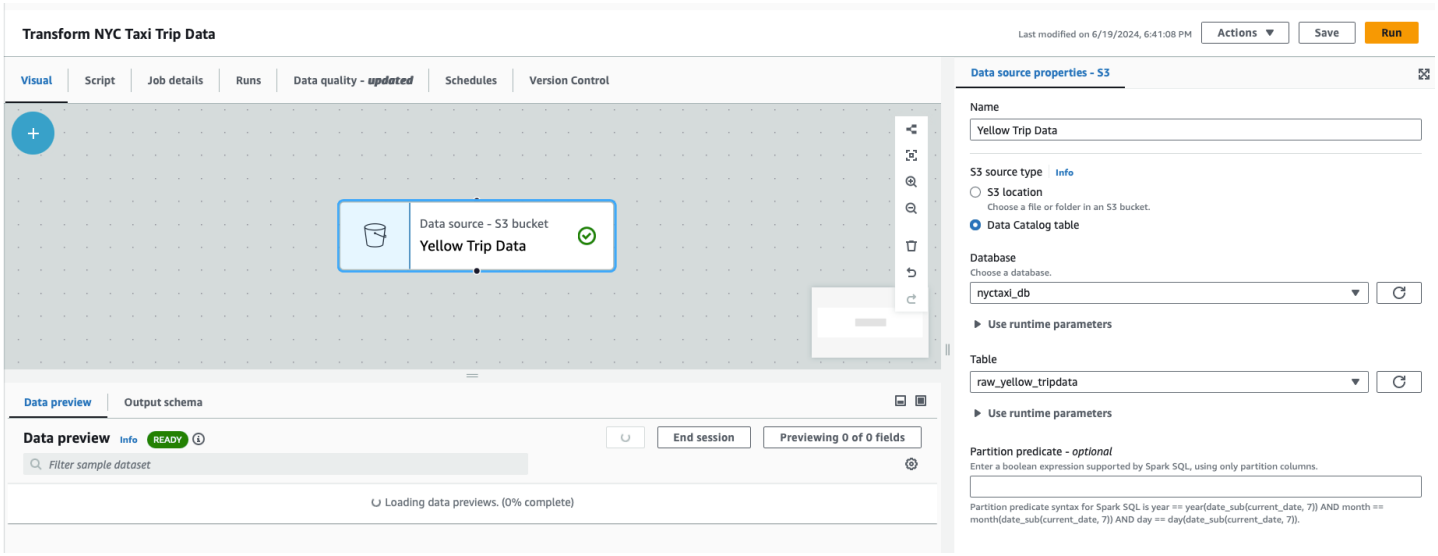
1. Click on the **Source** icon, choose **S3**.



2. In the Data source - S3 bucket node, the specify the following information:

- Node properties tab, **Name** - `Yellow Trip Data`

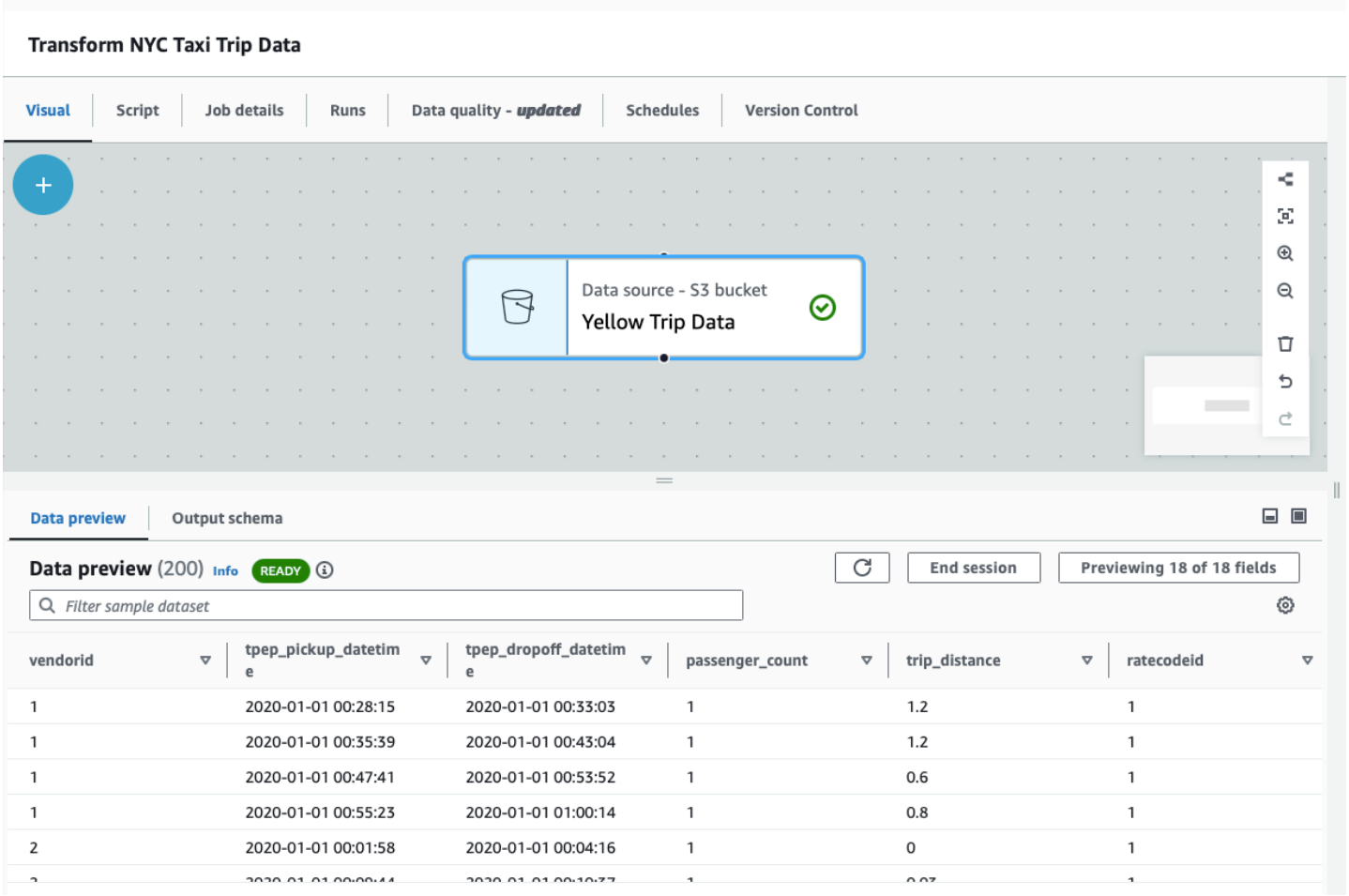
- Data source properties tab, **Database** - `nyctaxi_db`
- Data source properties tab, **Table** - `raw_yellow_tripdata`

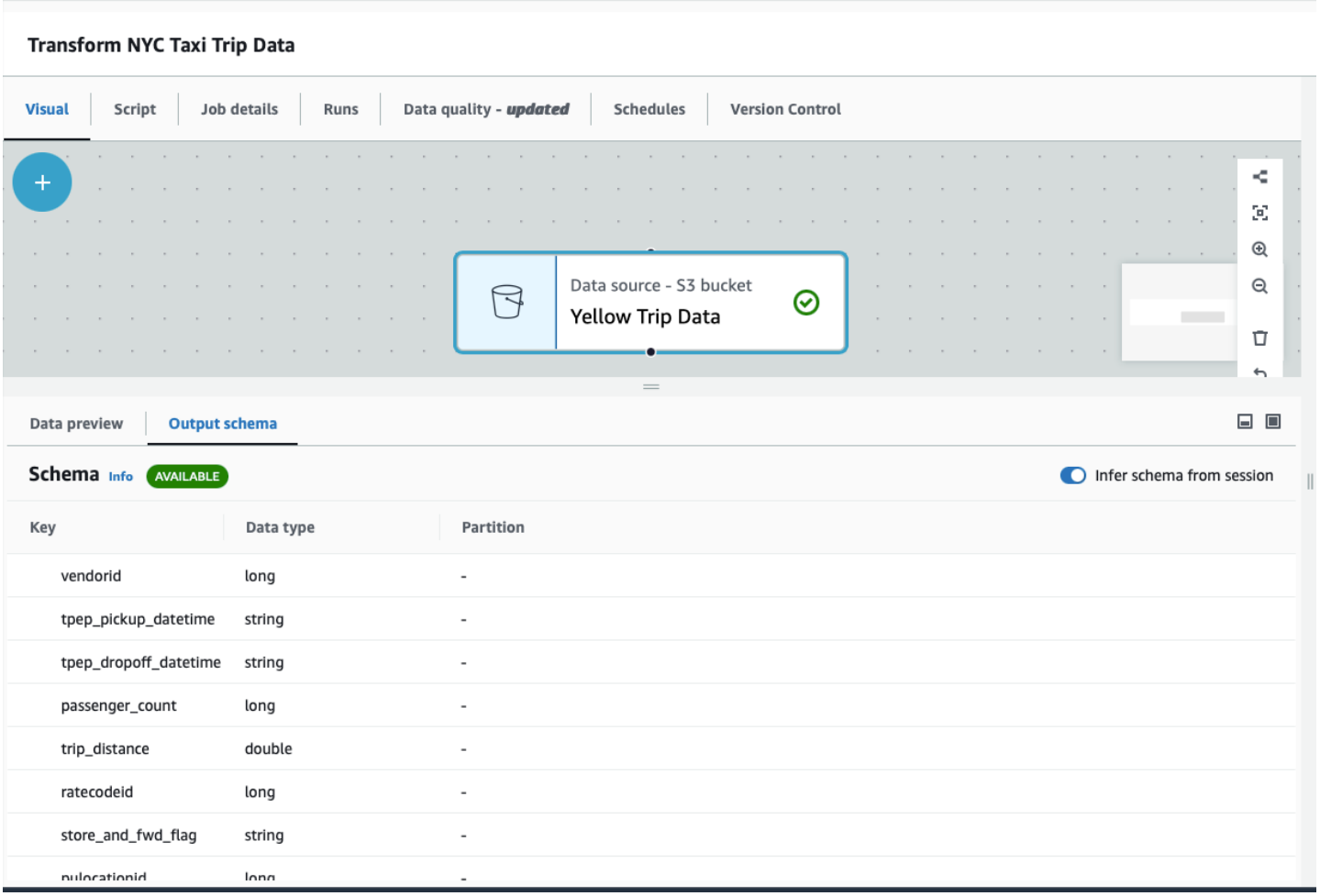


3. Click **Save**. Remember to save your work as you progress on building the transformation steps.

Review resulting data schema and previewing data

4. Go to the Output schema tab to review the resulting data schema.
5. Go to Data preview tab to check sample data set while editing your job.
    - If asked for an IAM role select **AWSGlueServiceRole-SDL-Jumpstart**


Transform NYC Taxi Trip Data
Visual
Script
Job details
Runs
Data quality - updated
Schedules
Version Control
Data source - S3 bucket
Yellow Trip Data
Data preview
Output schema
Data preview (200) Info
READY
End session
Previewing 18 of 18 fields
Filter sample dataset
vendorid
tpep_pickup_datetime
tpep_dropoff_datetime
passenger_count
trip_distance
ratecodeid
1 2020-01-01 00:28:15 2020-01-01 00:33:03 1 1.2 1
1 2020-01-01 00:35:39 2020-01-01 00:43:04 1 1.2 1
1 2020-01-01 00:47:41 2020-01-01 00:53:52 1 0.6 1
1 2020-01-01 00:55:23 2020-01-01 01:00:14 1 0.8 1
2 2020-01-01 00:01:58 2020-01-01 00:04:16 1 0 1

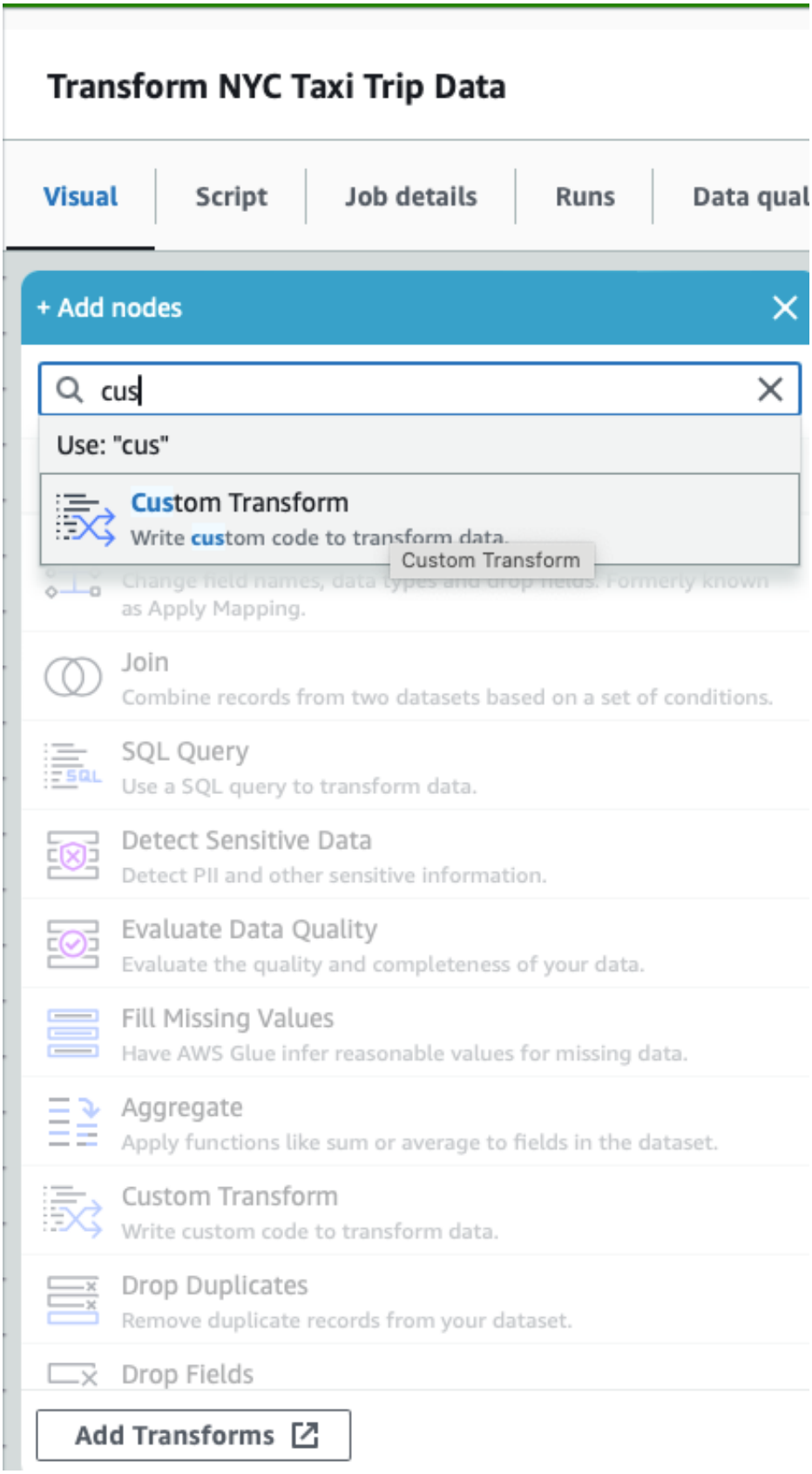


AWS Glue Studio now allows you to preview your data at each step of the visual job authoring process so you can test and debug your transformations without having to save or run the job.

The first time you choose the Data preview tab, you are prompted to choose an IAM role to use. The IAM role you choose must have the necessary permissions to create the data previews. This can be the same role that you plan to use for your job, or it can be a different role.

After you choose an IAM role, it takes about 20 to 30 seconds before the data appears. You are charged for data preview usage as soon as you choose the IAM role.

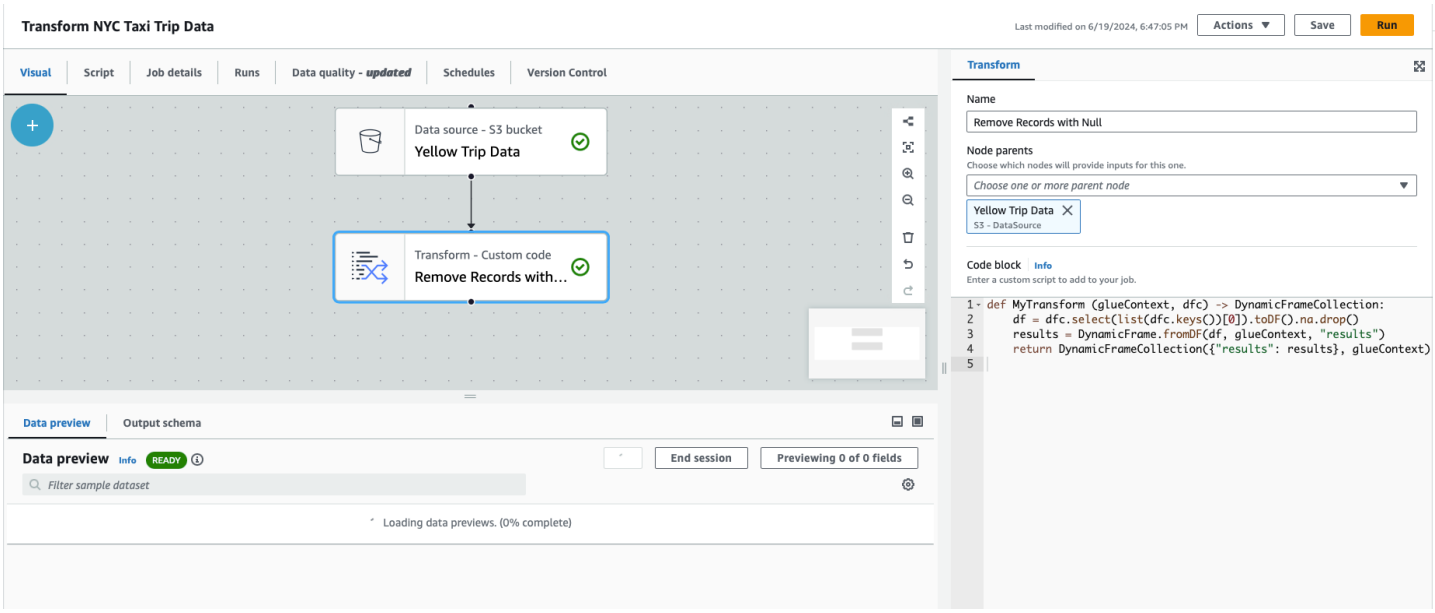
Remove records with NULL values



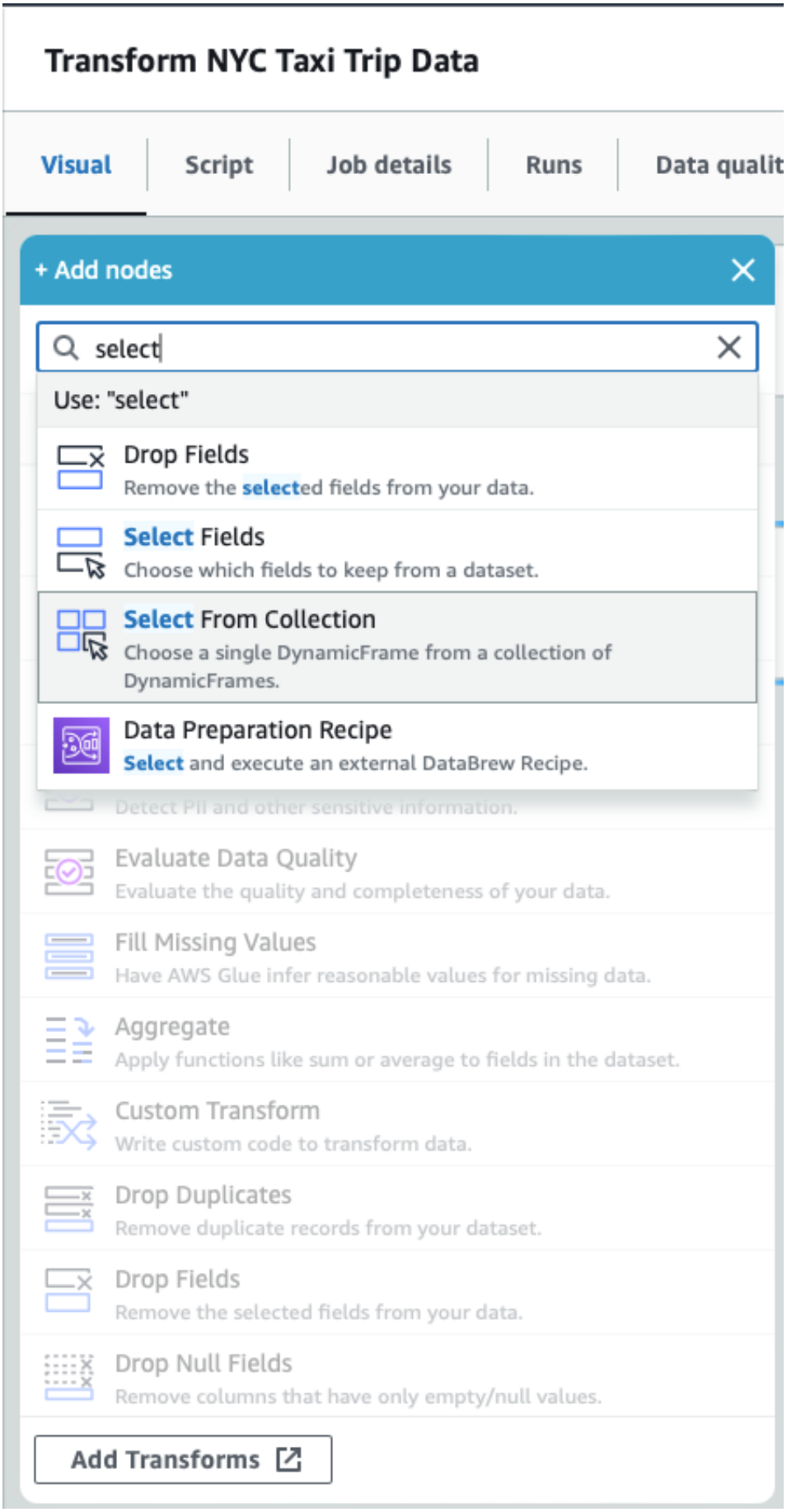
1. Click on the **Transforms** tab and choose **Custom Transform**. You can also search for the specific node.
2. In the Transform – Custom code node, specify the following information:
    - Node properties tab, **Name** - `Remove Records with NULL`

- Add the following in the Code block:

```
def MyTransform (glueContext, dfc) -> DynamicFrameCollection:
    df = dfc.select(list(dfc.keys())[0]).toDF().na.drop()
    results = DynamicFrame.fromDF(df, glueContext, "results")
    return DynamicFrameCollection({"results": results}, glueContext)
```

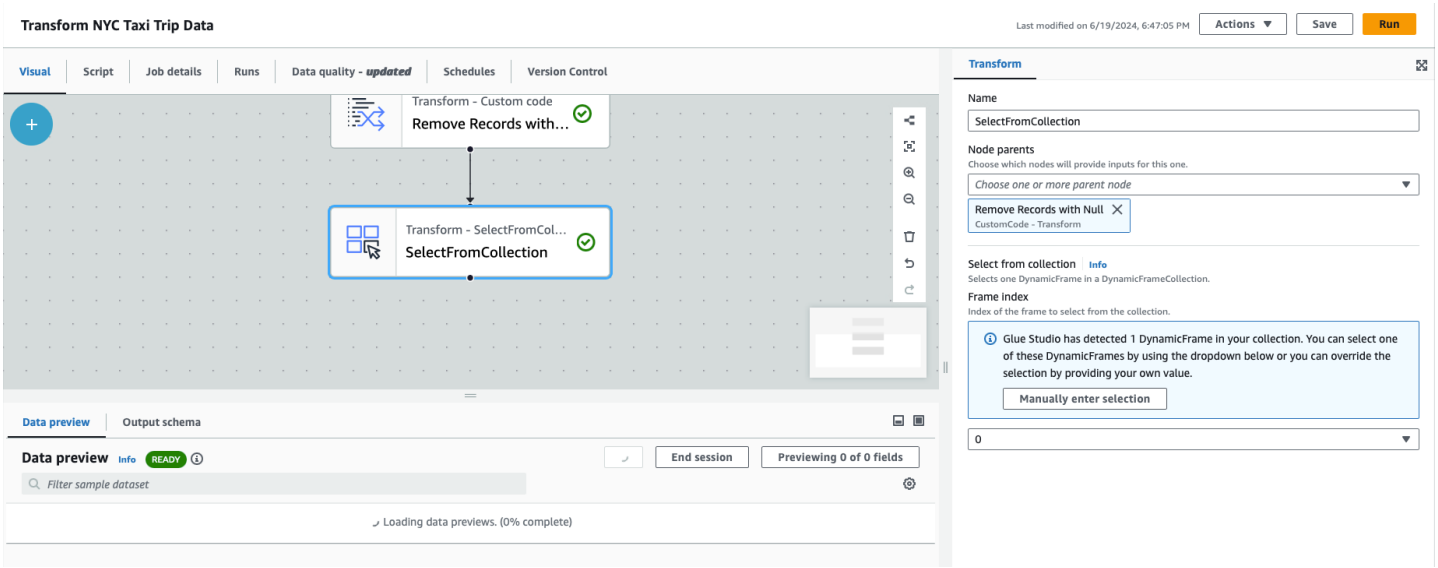


3. Click on the **Transform** icon, choose **SelectFromCollection**.


Transform NYC Taxi Trip Data
Visual
Script
Job details
Runs
Data qualit
+ Add nodes
select
Use: "select"
Drop Fields
Remove the selected fields from your data.
Select Fields
Choose which fields to keep from a dataset.
Select From Collection
Choose a single DynamicFrame from a collection of DynamicFrames.
Data Preparation Recipe
Select and execute an external DataBrew Recipe.
Detect PII and other sensitive information.
Evaluate Data Quality
Evaluate the quality and completeness of your data.
Fill Missing Values
Have AWS Glue infer reasonable values for missing data.
Aggregate
Apply functions like sum or average to fields in the dataset.
Custom Transform
Write custom code to transform data.
Drop Duplicates
Remove duplicate records from your dataset.
Drop Fields
Remove the selected fields from your data.
Drop Null Fields
Remove columns that have only empty/null values.
Add Transforms

4. Specify the following information:

- Node properties tab, **Name** - `SelectFromCollection`
- Transform tab, **Frame index** - `0`



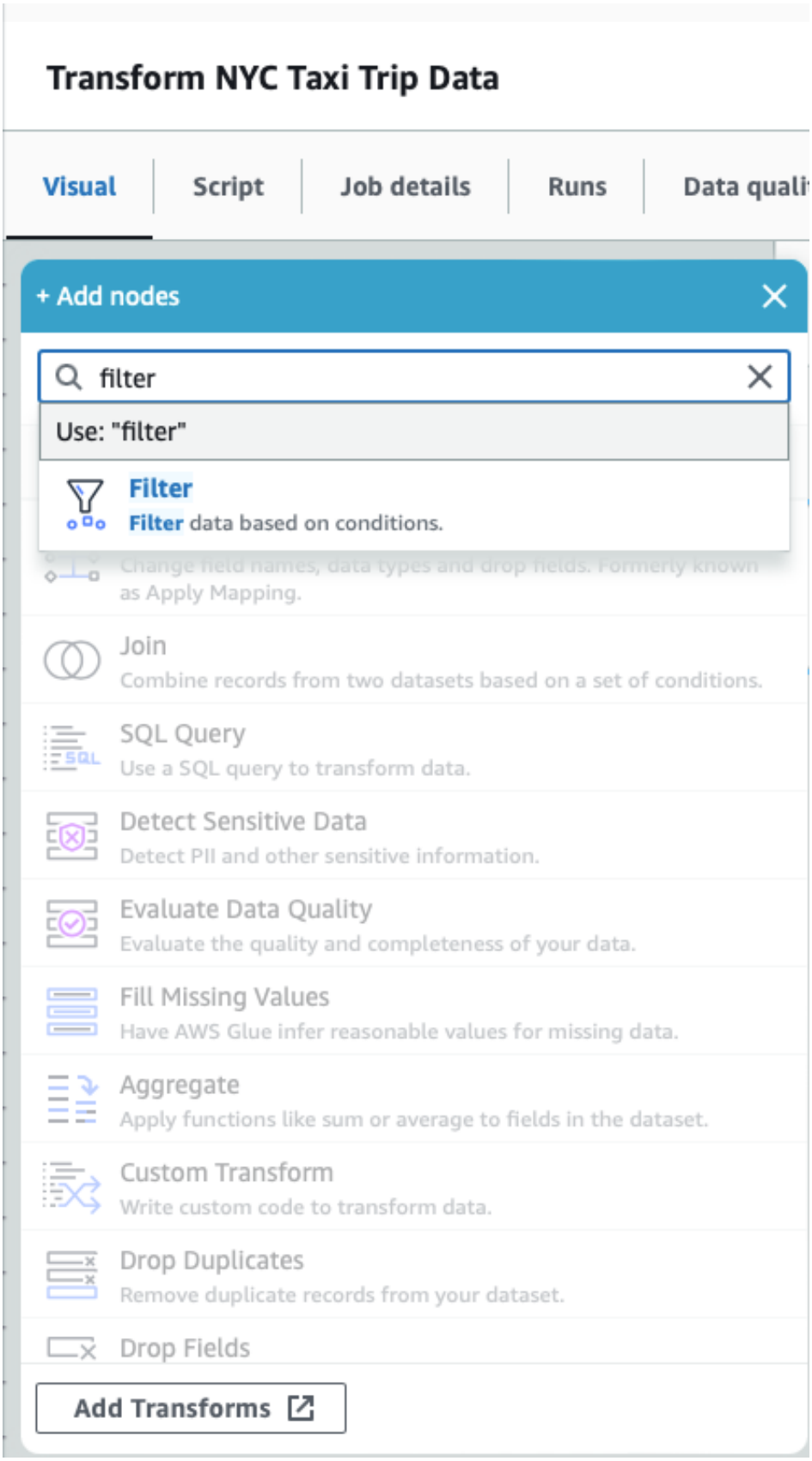
5. Remember to save your work.

Filter Records

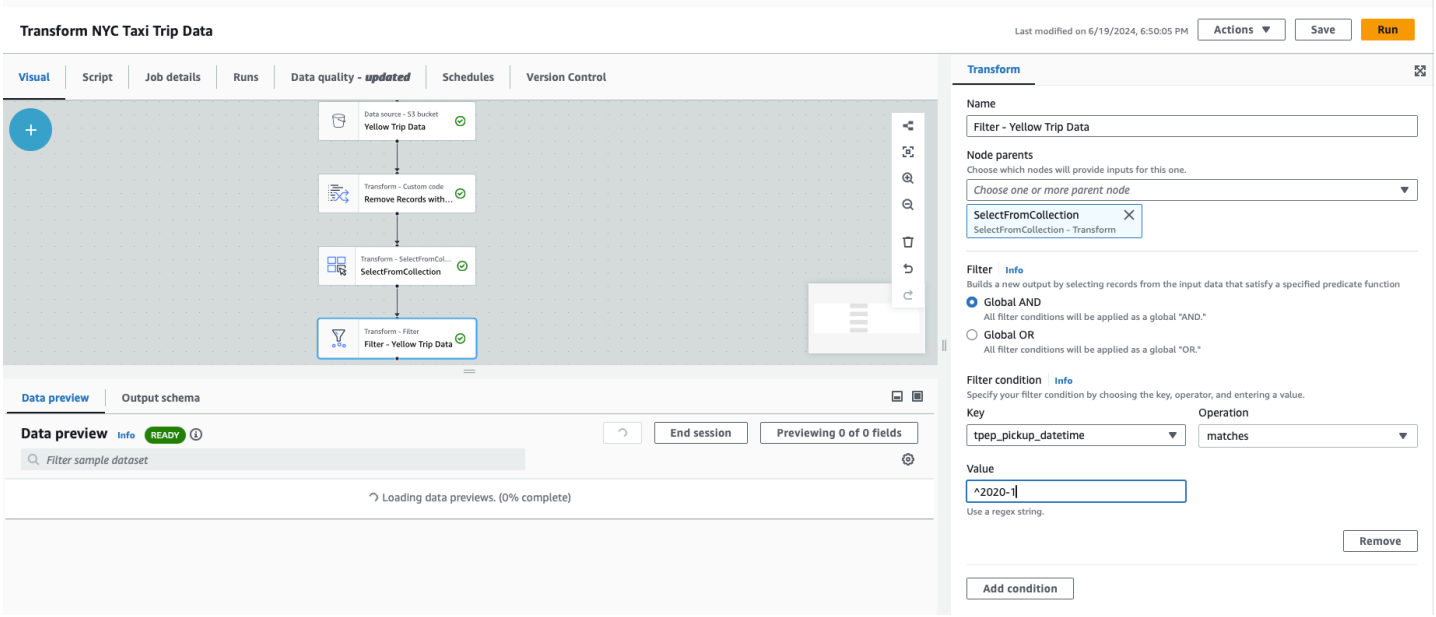
Filter records to remove records with invalid Pickup DateTime

In the interest of time, we will filter the records between *October 2020* and *December 2020* to reduce job processing time during the workshop.

1. Click on the **Transform** icon, choose **Filter**.



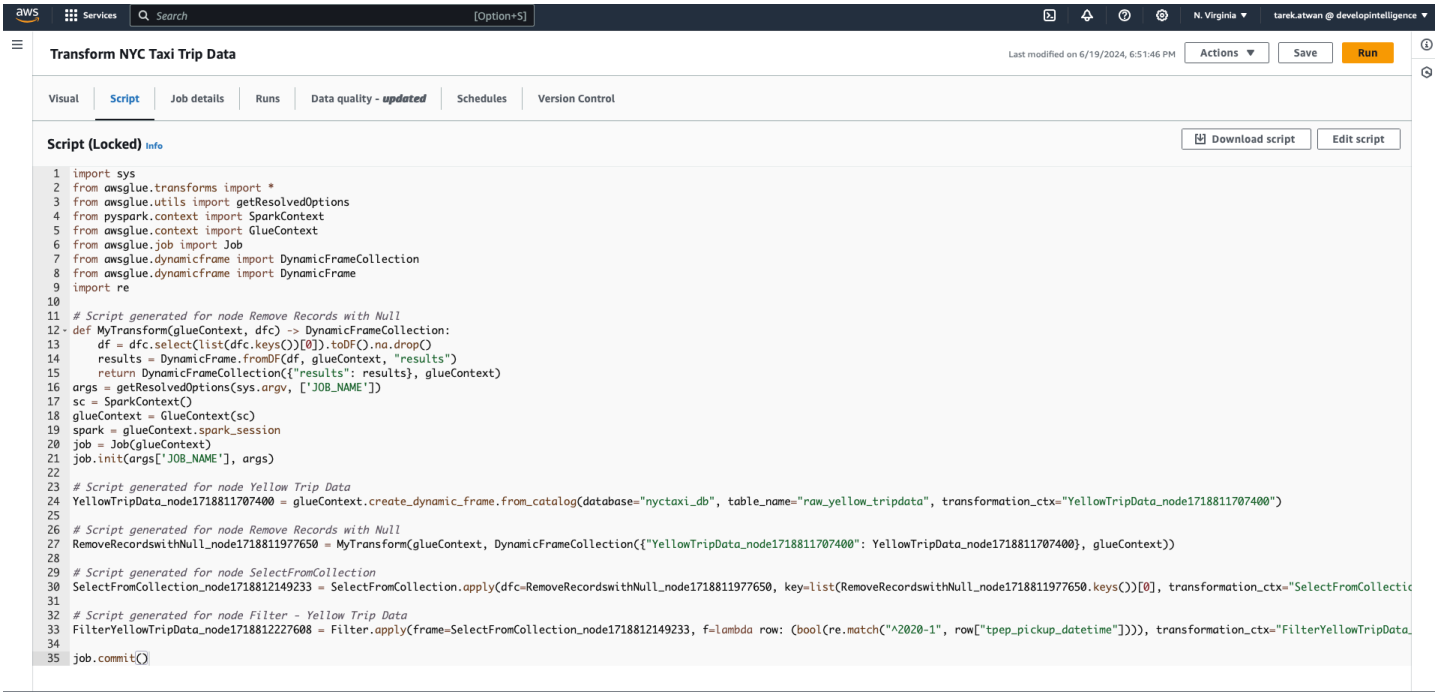
2. Specify the following information:
    - Node properties tab, **Name** – `Filter - Yellow Trip Data`
    - Transform tab, **Filter condition** – `tpep_pickup_datetime` `matches` `^2020-1`



3. Remember to save your work

Review the automatically generated script

1. Go to the Script details tab.
2. Observe that the code of the script are automatically generated by Glue Studio.

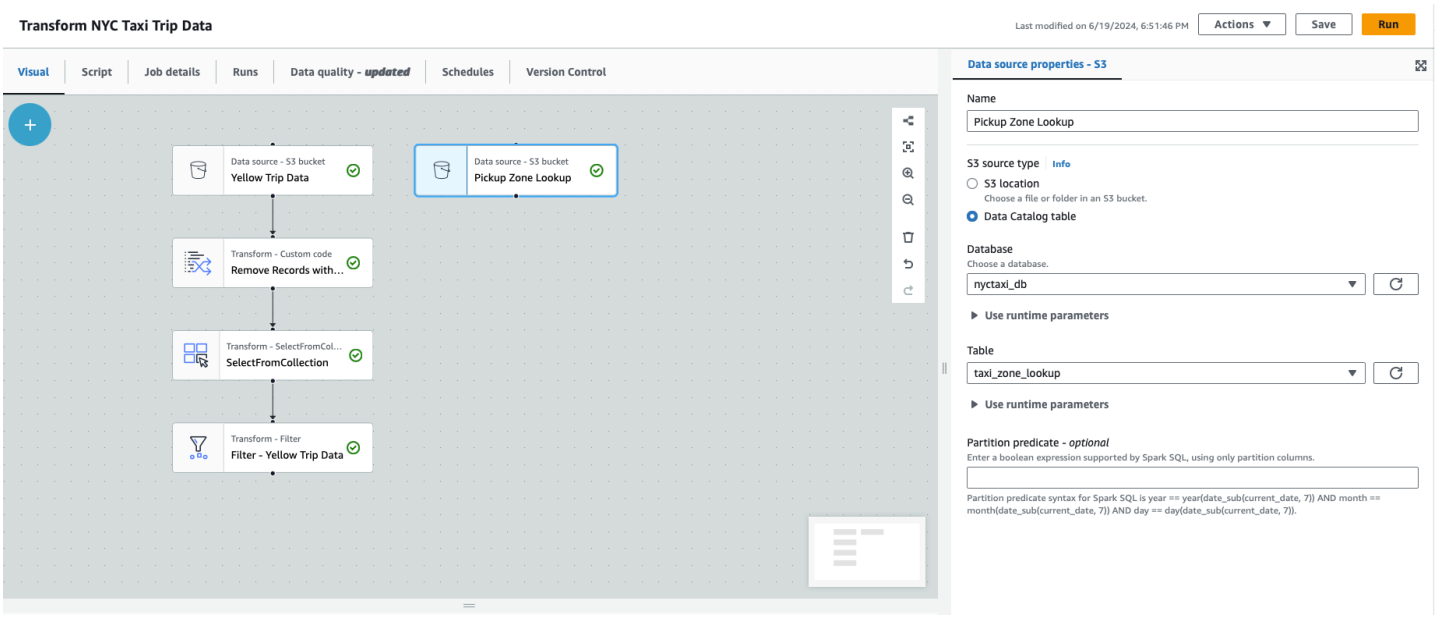


Add another data source

Add Lookup table for Taxi Pickup Zone

1. Return to the Visual tab

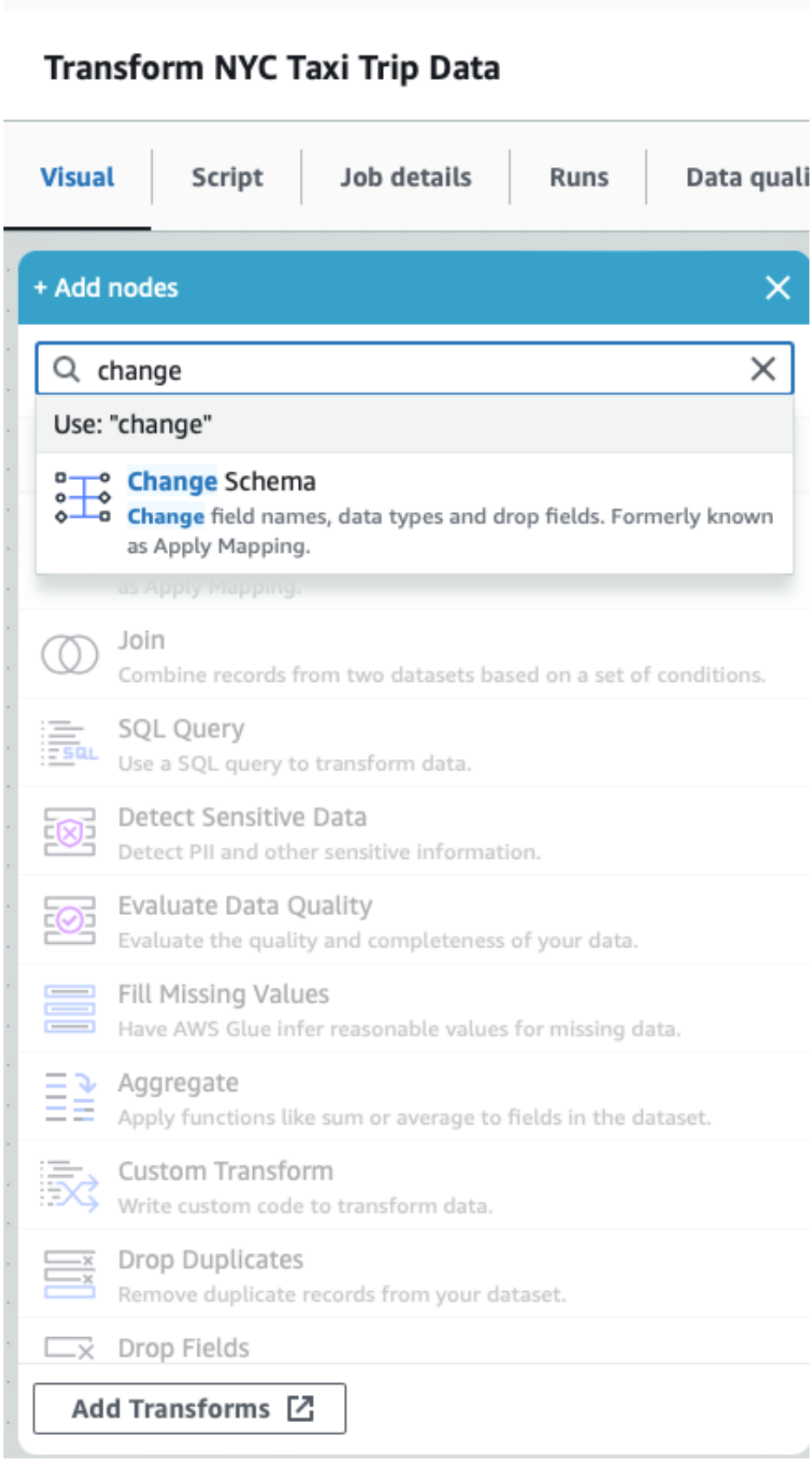
2. Click on the **Source icon**, choose **S3**
3. In the **Data source - S3 bucket** node, the specify the following information:
    - Node properties tab, **Name** - `Pickup Zone Lookup`
    - Data source properties tab, **Database** - `nyctaxi_db`
    - Data source properties tab, **Table** - `taxi_zone_lookup`

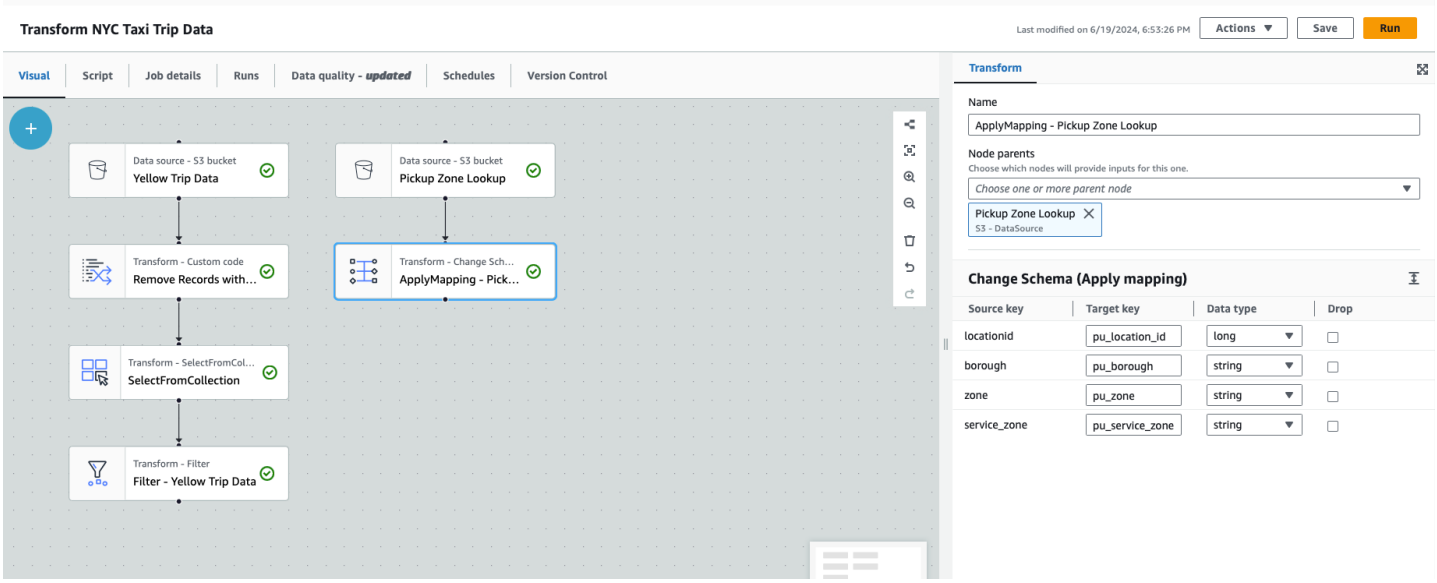


4. Remember to save your work.

Modify column names of Pickup Taxi Zone Lookup table

1. Make sure the Amazon S3 - Pickup Zone Lookup node is selected.
2. Click on the Transform icon, choose **Choose Schema**. Formerly known as **Apply Mapping**
3. Specify the following information:
    - Node properties tab, **Name** - `ApplyMapping - Pickup Zone Lookup`
    - Transform tab, modify the target key
        - **locationid** to `pu_location_id`
        - **borough** to `pu_borough`
        - **zone** to `pu_zone`
        - **service_zone** to `pu_service_zone`


Transform NYC Taxi Trip Data
Visual
Script
Job details
Runs
Data quali
+ Add nodes
change
Use: "change"
Change Schema
Change field names, data types and drop fields. Formerly known as Apply Mapping.
Join
Combine records from two datasets based on a set of conditions.
SQL Query
Use a SQL query to transform data.
Detect Sensitive Data
Detect PII and other sensitive information.
Evaluate Data Quality
Evaluate the quality and completeness of your data.
Fill Missing Values
Have AWS Glue infer reasonable values for missing data.
Aggregate
Apply functions like sum or average to fields in the dataset.
Custom Transform
Write custom code to transform data.
Drop Duplicates
Remove duplicate records from your dataset.
Drop Fields
Add Transforms

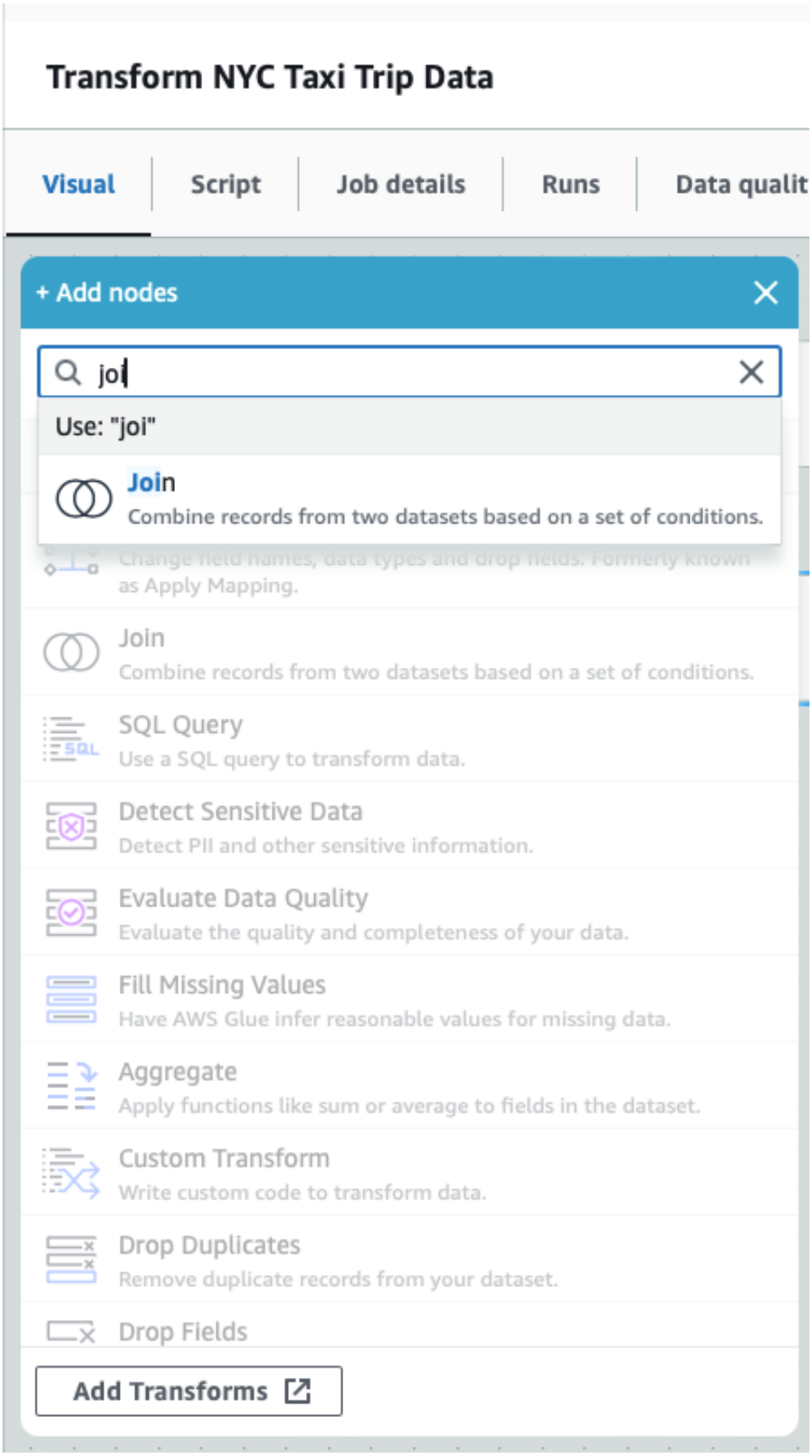


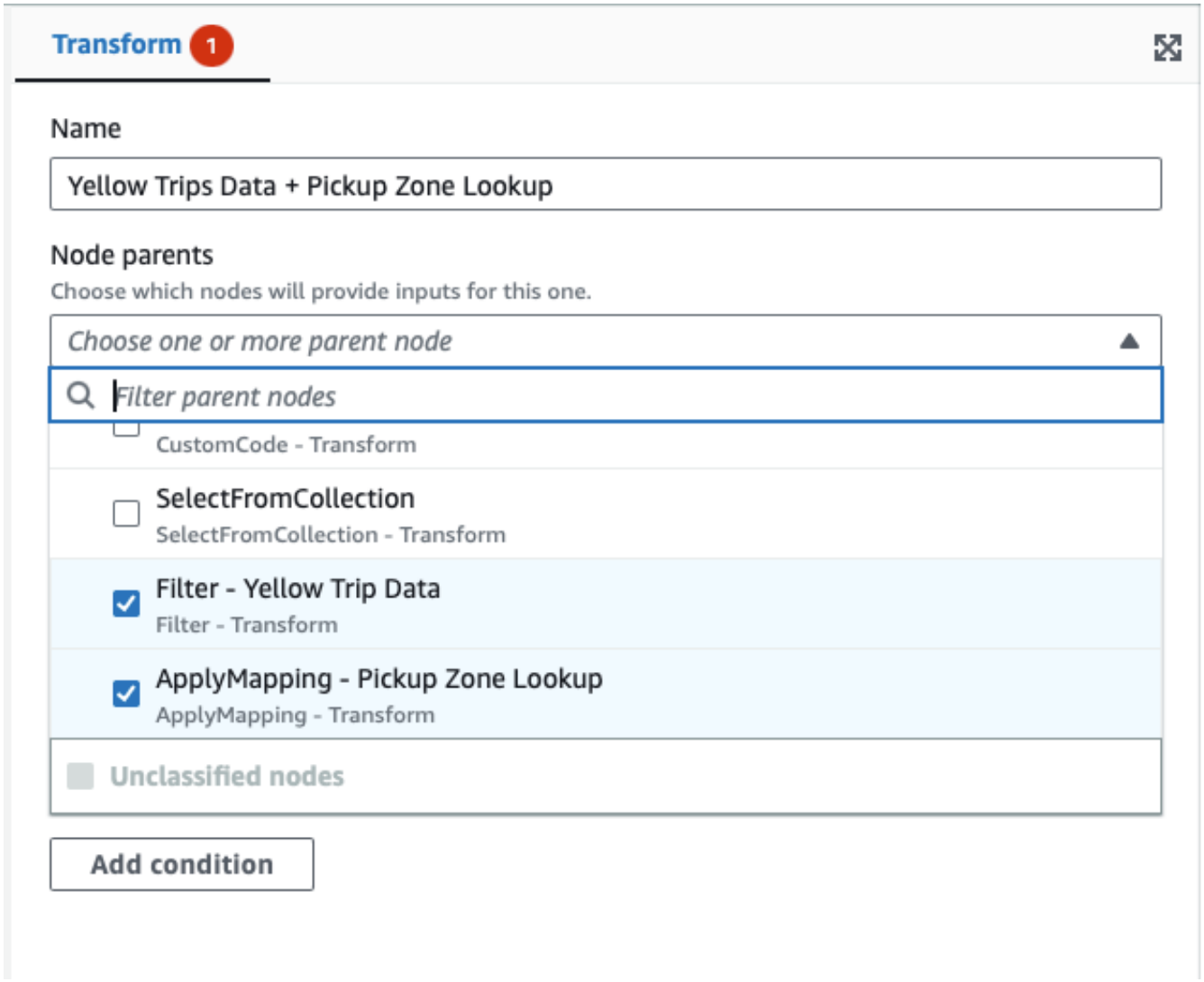
4. Remember to save your work.

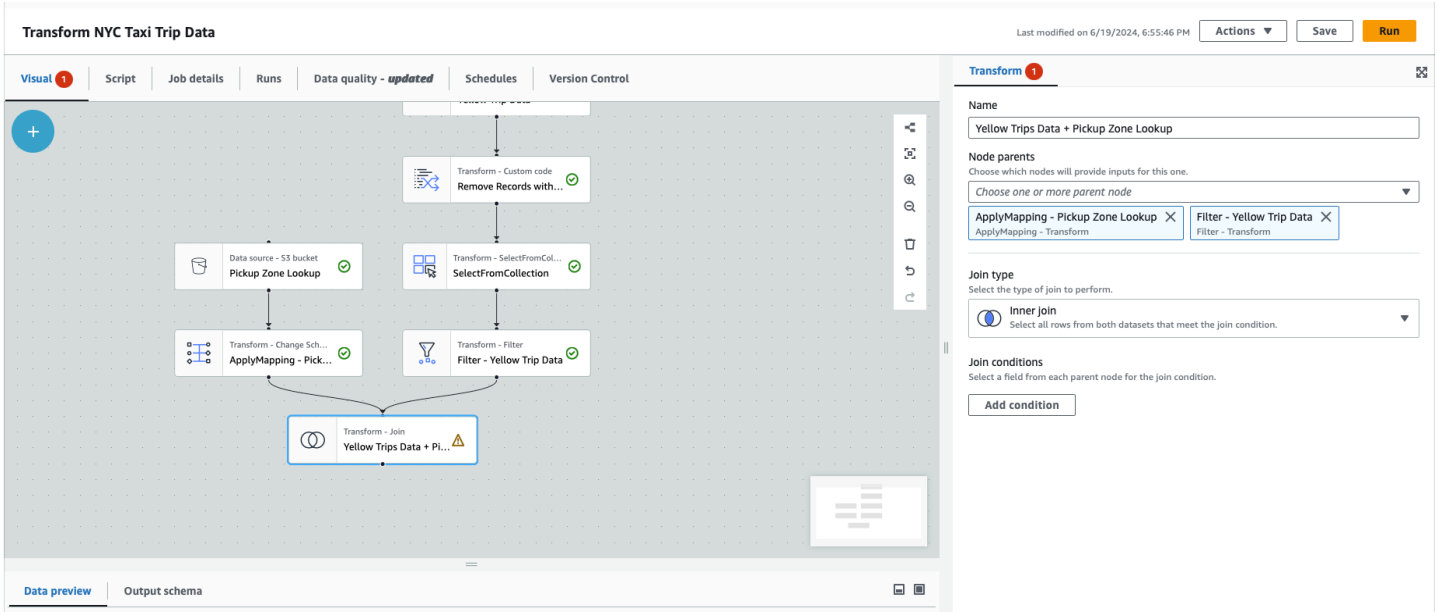
Join Data

Join the Yellow Trips data with Pickup Taxi Zone Lookup data

1. Click on the **Transform** icon, choose **Join.**
2. Specify the following information:
    - Node properties tab, **Name** - `Yellow Trips Data + Pickup Zone Lookup`
    - Node properties tab, **Node Parents**
        - `ApplyMapping - Pickup Zone Lookup`
        - `Filter - Yellow Trip Data`

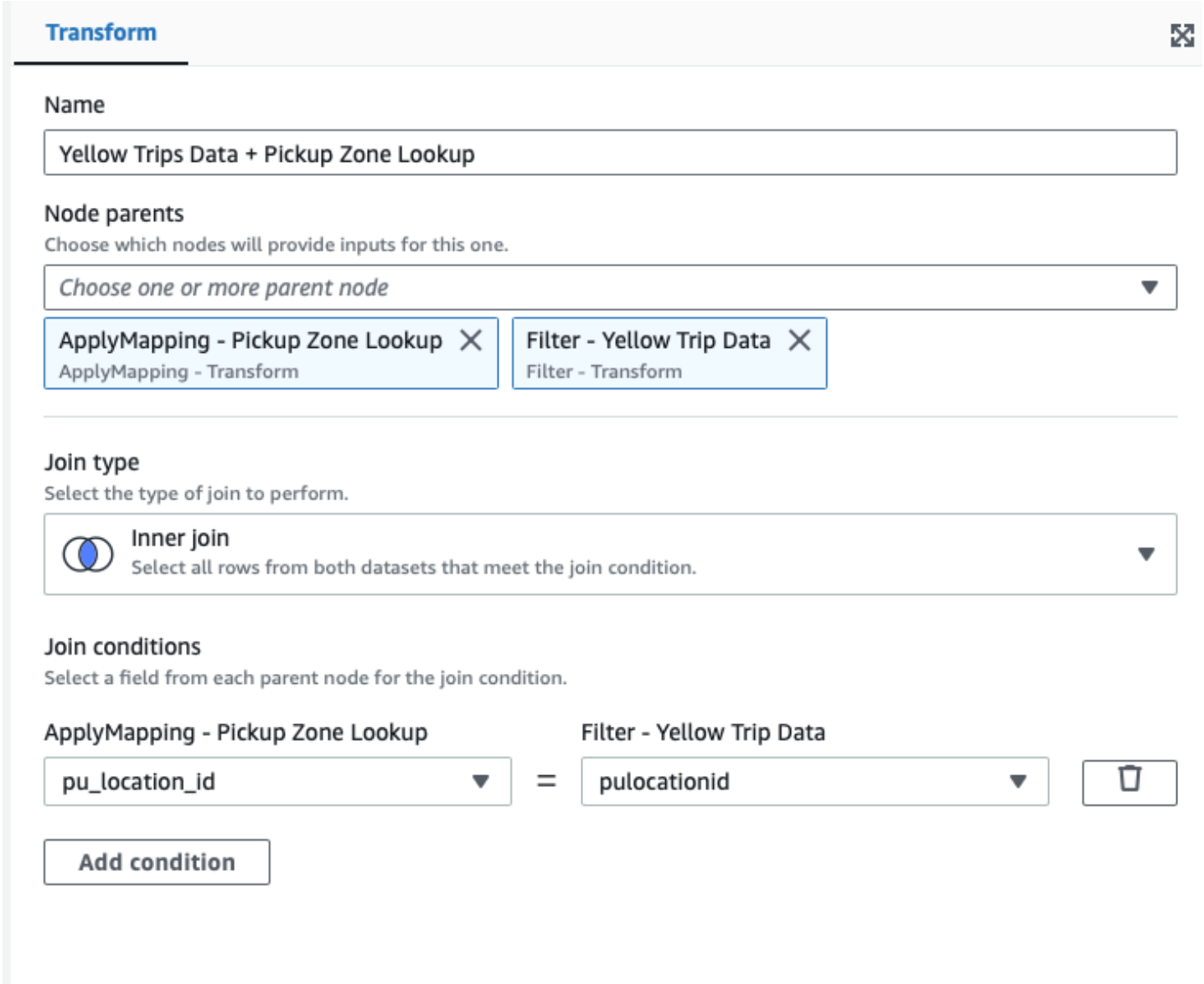

Transform NYC Taxi Trip Data
Visual
Script
Job details
Runs
Data qualit
+ Add nodes
joi
Use: "joi"
Join
Combine records from two datasets based on a set of conditions.
as Apply Mapping.
Join
Combine records from two datasets based on a set of conditions.
SQL Query
Use a SQL query to transform data.
Detect Sensitive Data
Detect PII and other sensitive information.
Evaluate Data Quality
Evaluate the quality and completeness of your data.
Fill Missing Values
Have AWS Glue infer reasonable values for missing data.
Aggregate
Apply functions like sum or average to fields in the dataset.
Custom Transform
Write custom code to transform data.
Drop Duplicates
Remove duplicate records from your dataset.
Drop Fields
Add Transforms


Transform 1
Name
Yellow Trips Data + Pickup Zone Lookup
Node parents
Choose which nodes will provide inputs for this one.
Choose one or more parent node
Filter parent nodes
CustomCode - Transform
SelectFromCollection
SelectFromCollection - Transform
Filter - Yellow Trip Data
Filter - Transform
ApplyMapping - Pickup Zone Lookup
ApplyMapping - Transform
Unclassified nodes
Add condition


Transform NYC Taxi Trip Data
Last modified on 6/19/2024, 6:55:46 PM
Actions
Save
Run
Visual 1
Script
Job details
Runs
Data quality - updated
Schedules
Version Control
Transform - Custom code
Remove Records with...
Data source - S3 bucket
Pickup Zone Lookup
Transform - SelectFromCol...
SelectFromCollection
Transform - Change Sch...
ApplyMapping - Pick...
Transform - Filter
Filter - Yellow Trip Data
Transform - Join
Yellow Trips Data + Pi...
Data preview
Output schema
Transform 1
Name
Yellow Trips Data + Pickup Zone Lookup
Node parents
Choose which nodes will provide inputs for this one.
Choose one or more parent node
ApplyMapping - Pickup Zone Lookup
ApplyMapping - Transform
Filter - Yellow Trip Data
Filter - Transform
Join type
Select the type of join to perform.
Inner join
Select all rows from both datasets that meet the join condition.
Join conditions
Select a field from each parent node for the join condition.
Add condition

Transform properties tab, under the **Join conditions**, select the following keys:

- **ApplyMapping - Pickup Zone Lookup** - `pu_location_id`
- **Filter - Yellow Trip Data** - `pulocationid`



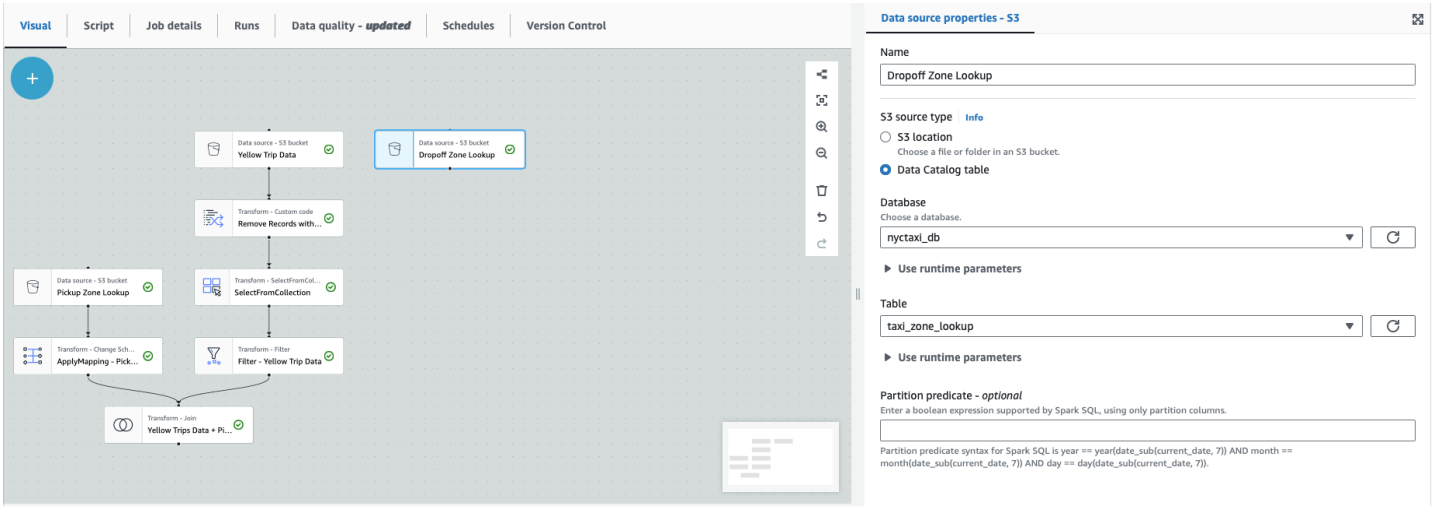
3. Remember to save your work.

Add and Join another dataset

Add Lookup table for Taxi Drop-off Zone

1. Click on the **Source** icon, choose **S3**
2. In the Data source – S3 bucket node, the specify the following information:
    - Node properties tab, **Name** - `Dropoff Zone Lookup`

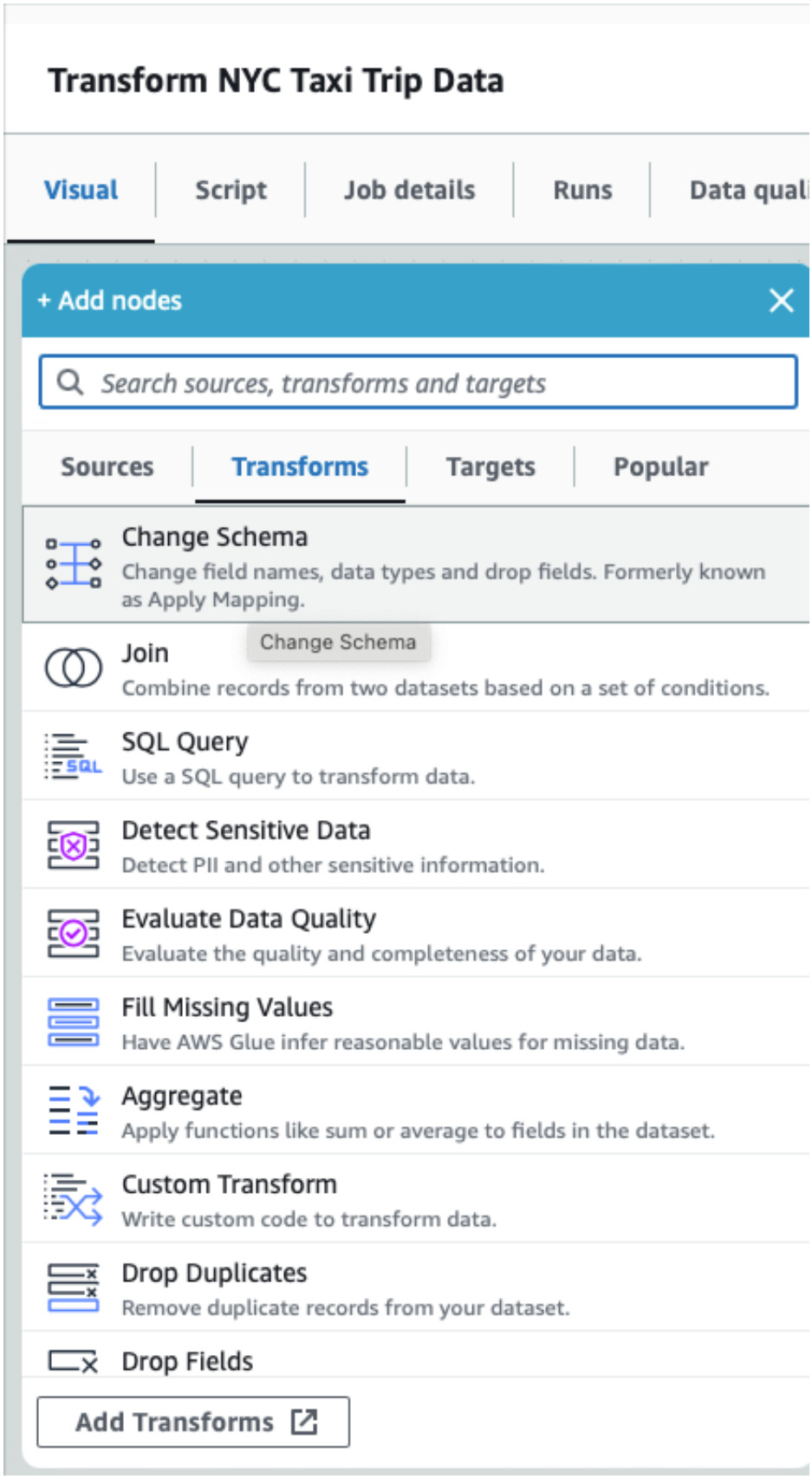
- Data source properties tab, **Database** - `nyctaxi_db`
- Data source properties tab, **Table** - `taxi_zone_lookup`

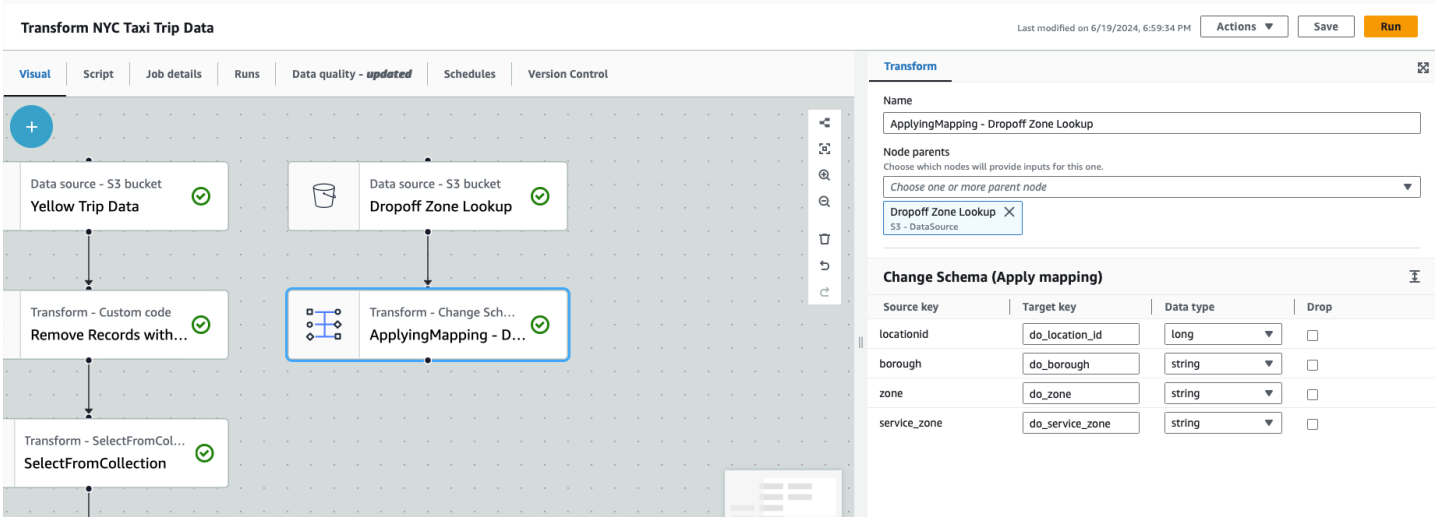


3. Remember to save your work.

Modify column names of Drop-off Taxi Zone Lookup Table

1. Make sure the Amazon S3 - Dropoff Zone Lookup node is selected.
2. Click on the **Transform** icon, choose **Change Schema**.
3. Specify the following information:
    - Node properties tab, **Name** - `ApplyMapping - Dropoff Zone Lookup`
    - Transform tab, modify the target key of the following:
        - **locationid** to `do_location_id`
        - **borough** to `do_borough`
        - **zone** to `do_zone`
        - **service_zone** to `do_service_zone`

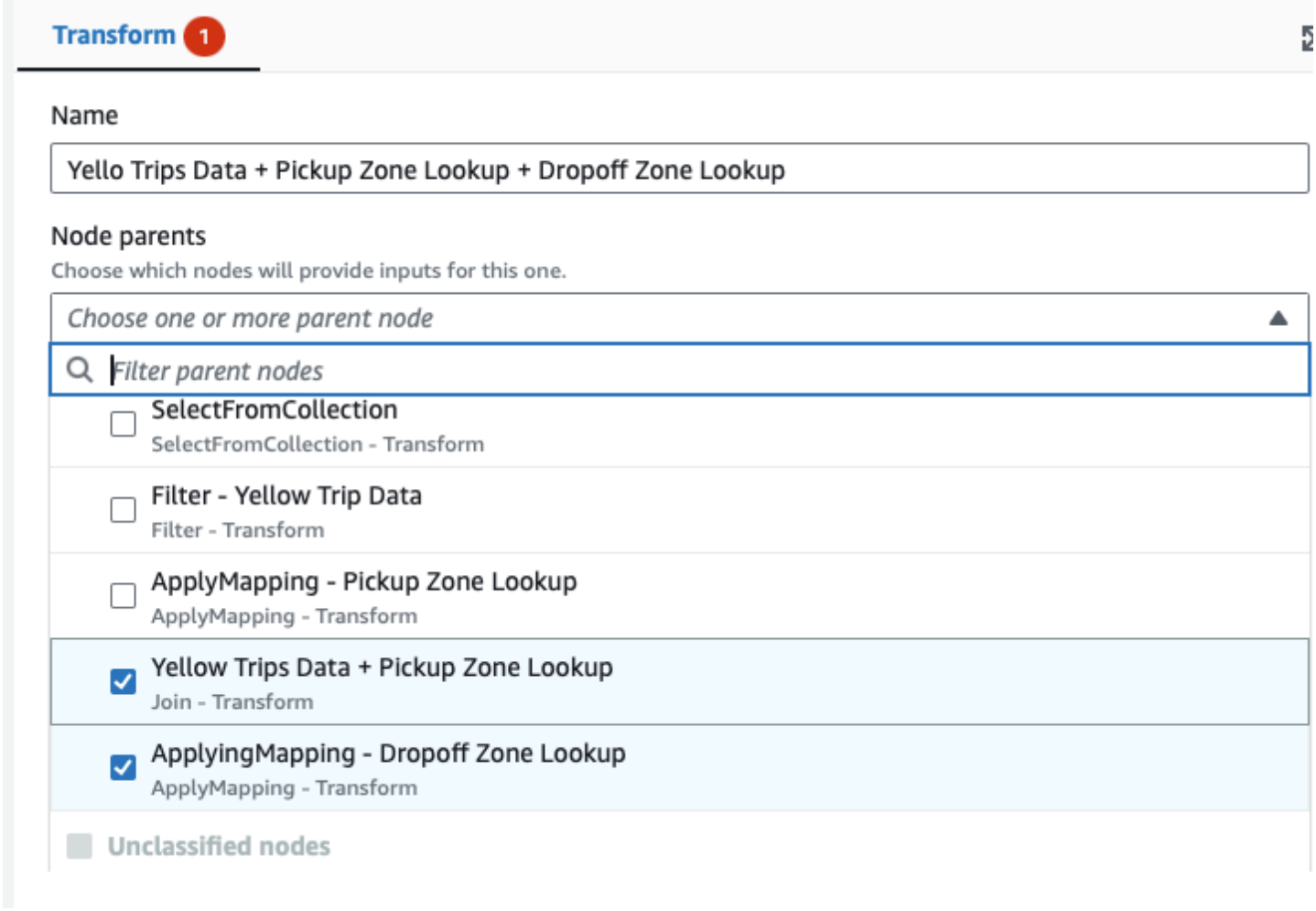

Transform NYC Taxi Trip Data
Visual
Script
Job details
Runs
Data quali
+ Add nodes
Search sources, transforms and targets
Sources
Transforms
Targets
Popular
Change Schema
Change field names, data types and drop fields. Formerly known as Apply Mapping.
Change Schema
Join
Combine records from two datasets based on a set of conditions.
SQL Query
Use a SQL query to transform data.
Detect Sensitive Data
Detect PII and other sensitive information.
Evaluate Data Quality
Evaluate the quality and completeness of your data.
Fill Missing Values
Have AWS Glue infer reasonable values for missing data.
Aggregate
Apply functions like sum or average to fields in the dataset.
Custom Transform
Write custom code to transform data.
Drop Duplicates
Remove duplicate records from your dataset.
Drop Fields
Add Transforms

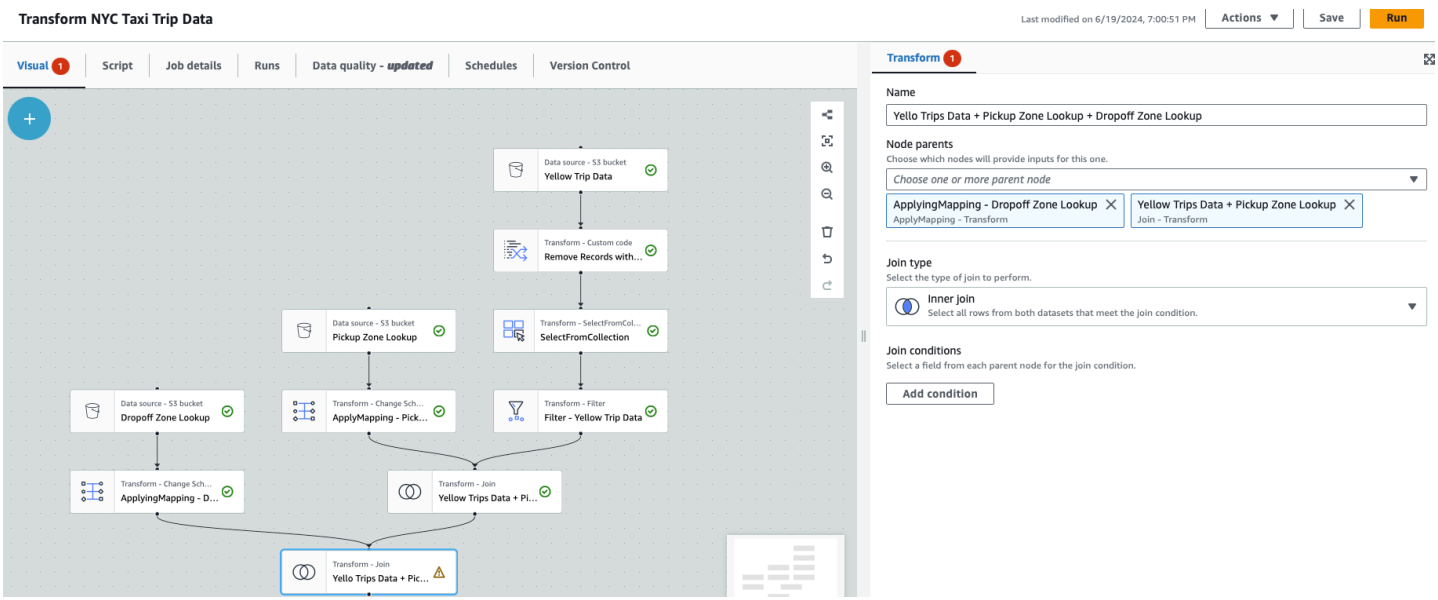


4. Remember to save your work.

Join Yellow Trips data and Dropoff Taxi Zone Lookup data

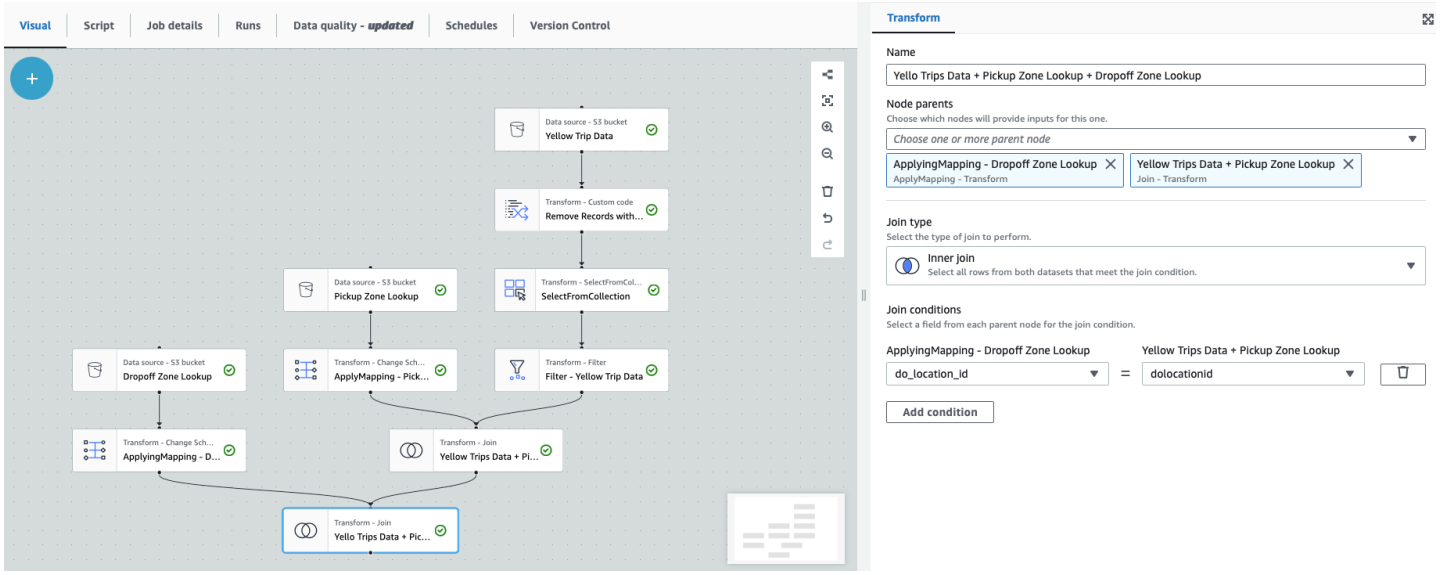
1. Click on the **Transform** icon, choose **Join**.
2. Specify the following information:
    - Node properties tab, **Name** - `Yellow Trips Data + Pickup Zone Lookup + Dropoff Zone Lookup`
    - Node properties tab, Node Parents
        - `ApplyMapping - Dropoff Zone Lookup`
        - `Yellow Trips Data + Pickup Zone Lookup`





Transform properties tab, under the **Join conditions**, select the following keys:

- **ApplyMapping - Dropoff Zone Lookup** – `do_location_id`
- **Yellow Trips Data + Pickup Zone Lookup** – `dolocationid`

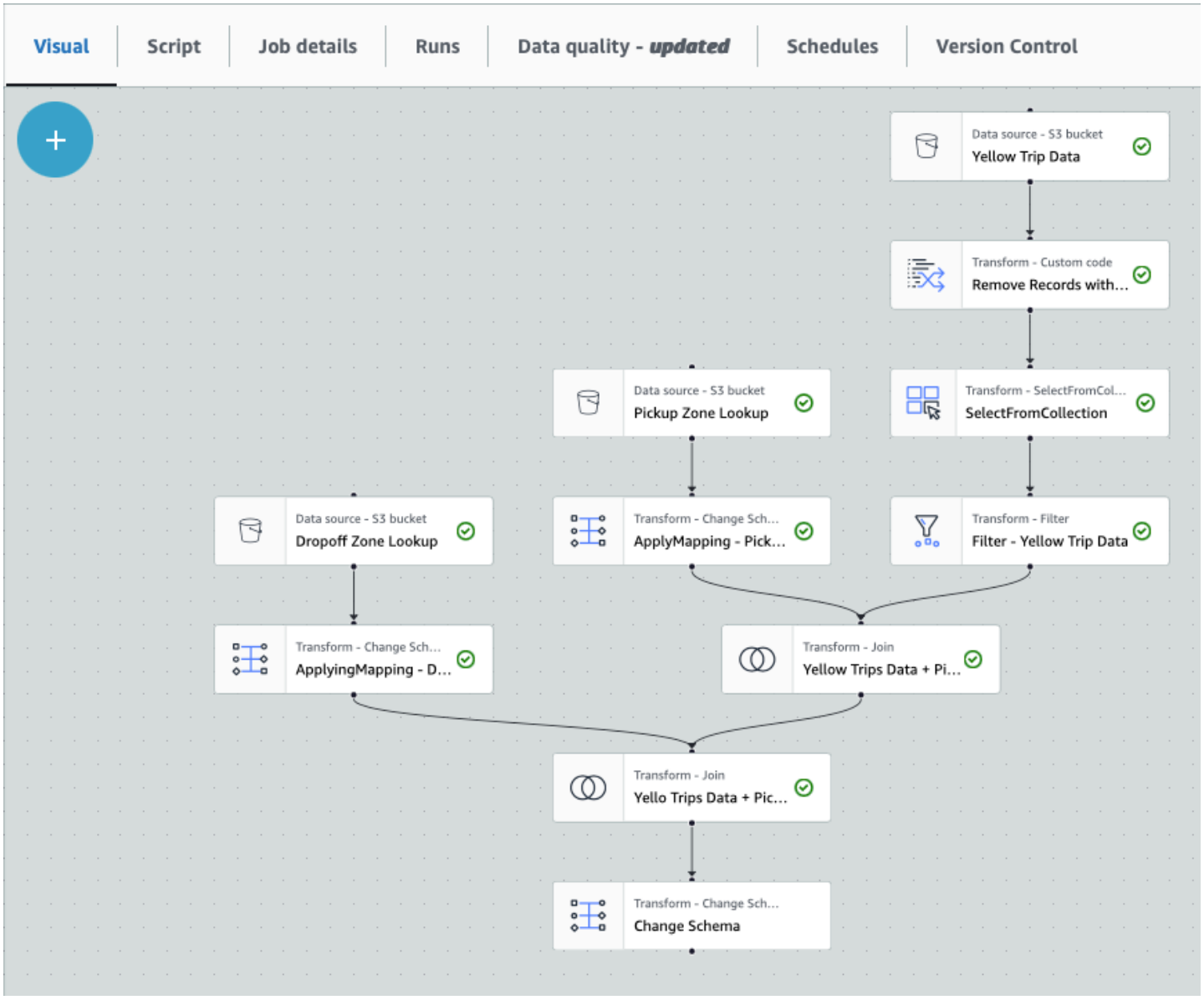


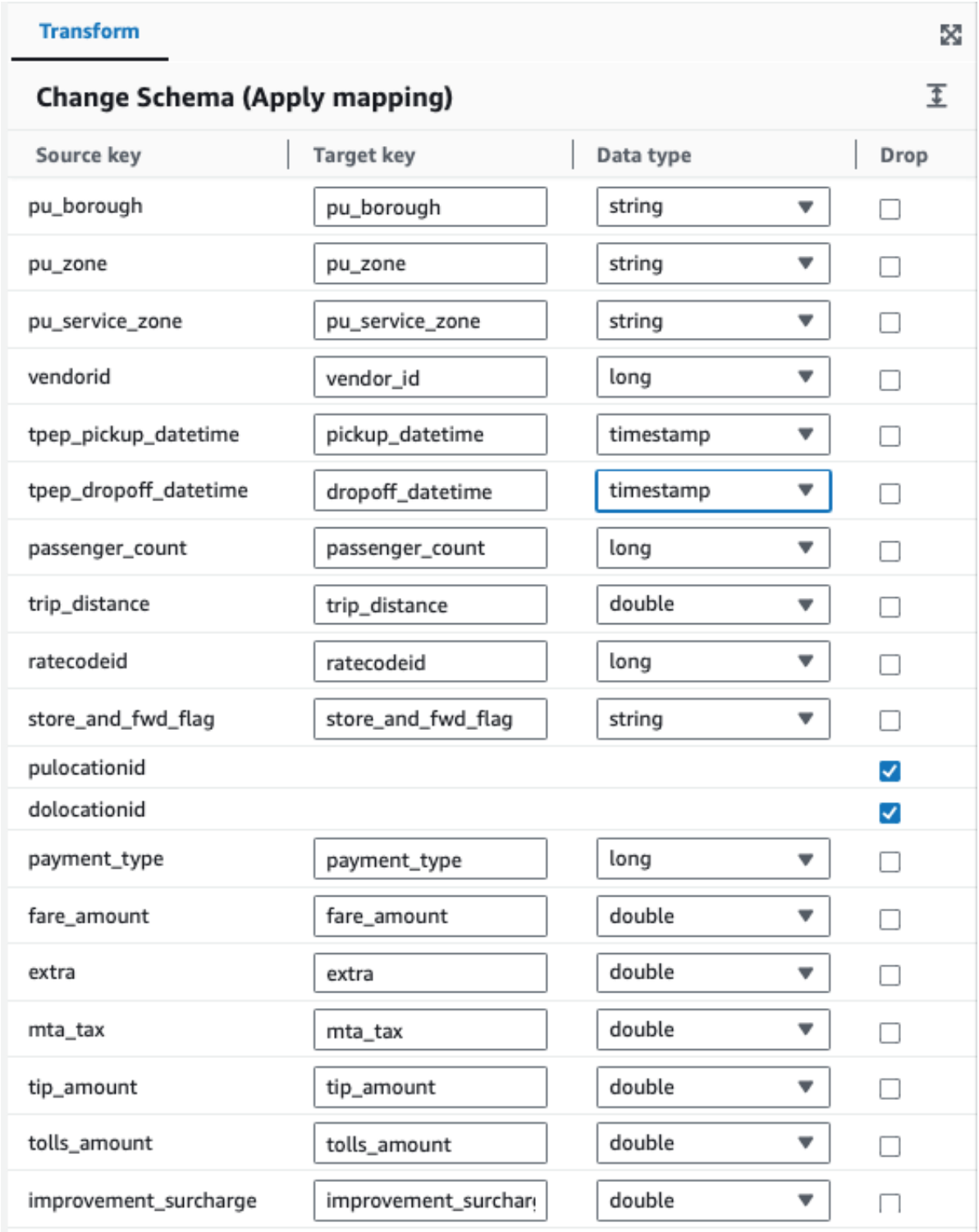
3. Remember to save your work.

Transform data and save to target

Modify column names and data types of the joined dataset

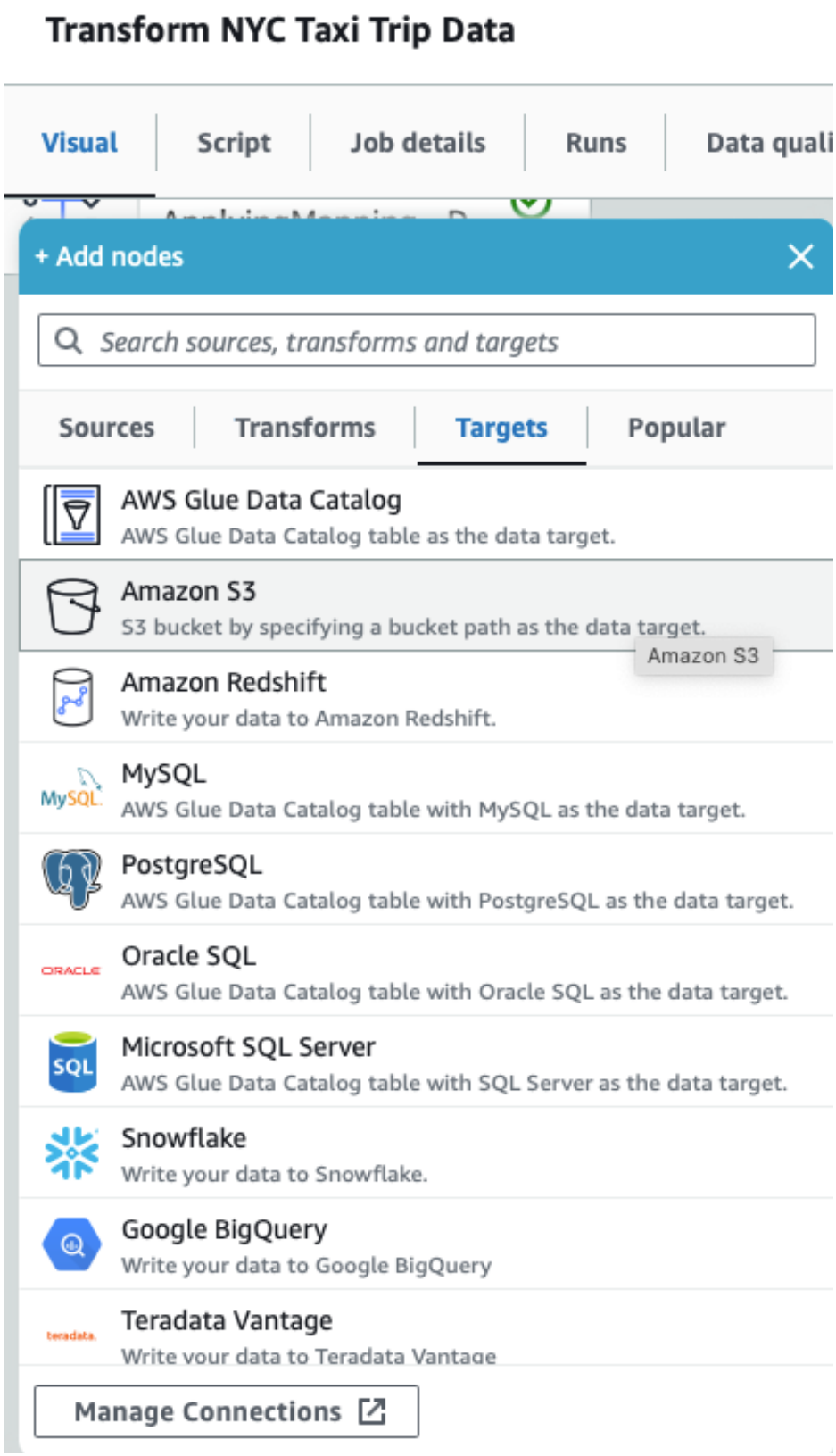
1. Click on the Transform icon, choose **Change Schema**.
2. Specify the following information:
    - Node properties tab, **Name** - `ApplyMapping - Joined Data`
    - Transform tab, modify the **Target key** and **Data type** of the following:
        - **vendorid** to `vendor_id`
        - **tpep_pickup_datetime** to `pickup_datetime`, **string** to `timestamp`
        - **tpep_dropoff_datetime** to `dropoff_datetime`, **string** to `timestamp`
    - Transform tab, drop the following **Source keys**:
        - `pulocationid`
        - `dolocationid`


Visual
Script
Job details
Runs
Data quality - updated
Schedules
Version Control
Data source - S3 bucket
Yellow Trip Data
Transform - Custom code
Remove Records with...
Data source - S3 bucket
Pickup Zone Lookup
Transform - SelectFromCol...
SelectFromCollection
Data source - S3 bucket
Dropoff Zone Lookup
Transform - Change Sch...
ApplyMapping - Pick...
Transform - Filter
Filter - Yellow Trip Data
Transform - Change Sch...
ApplyingMapping - D...
Transform - Join
Yellow Trips Data + Pi...
Transform - Join
Yello Trips Data + Pic...
Transform - Change Sch...
Change Schema



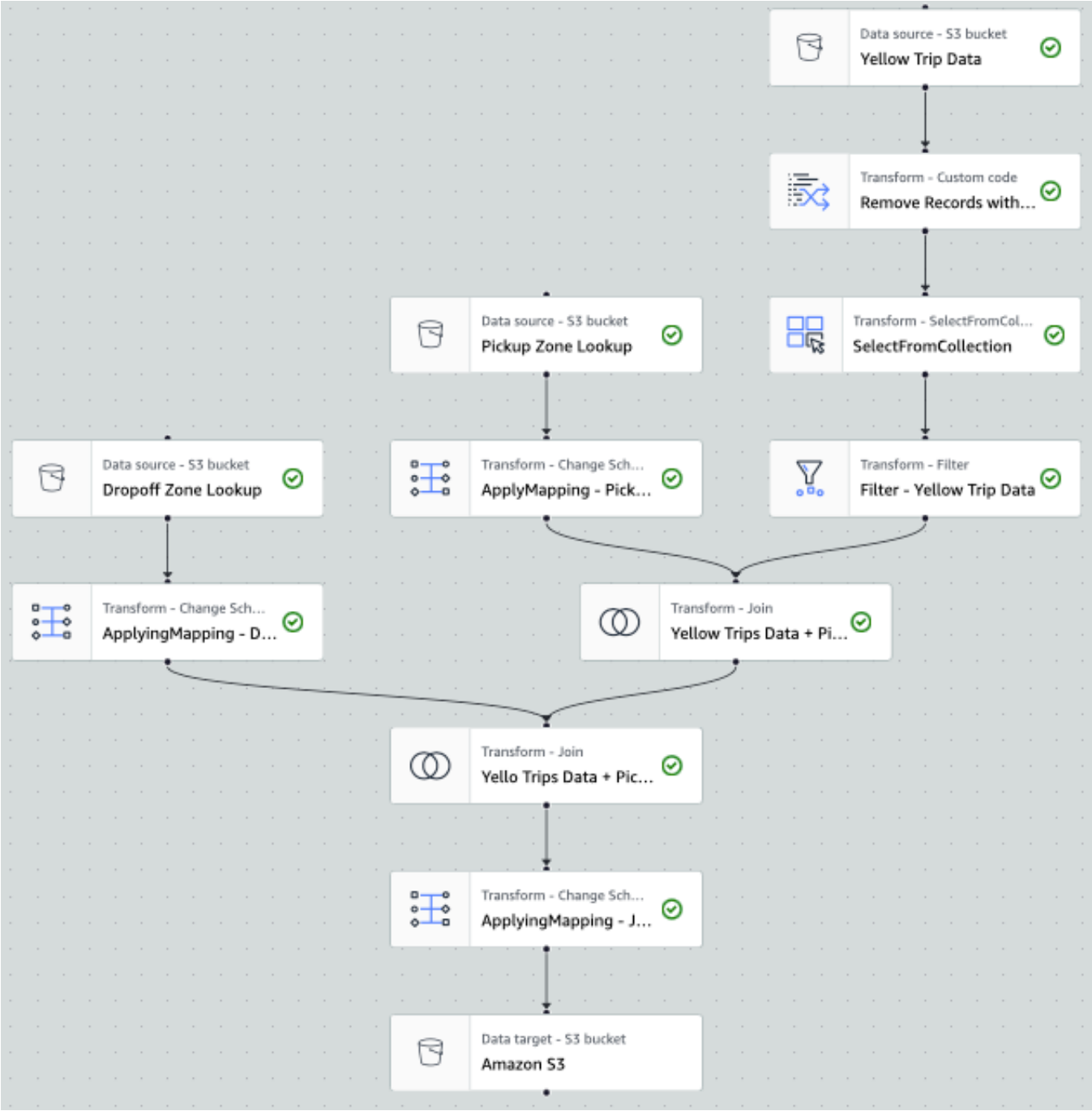
3. Remember to save your work.

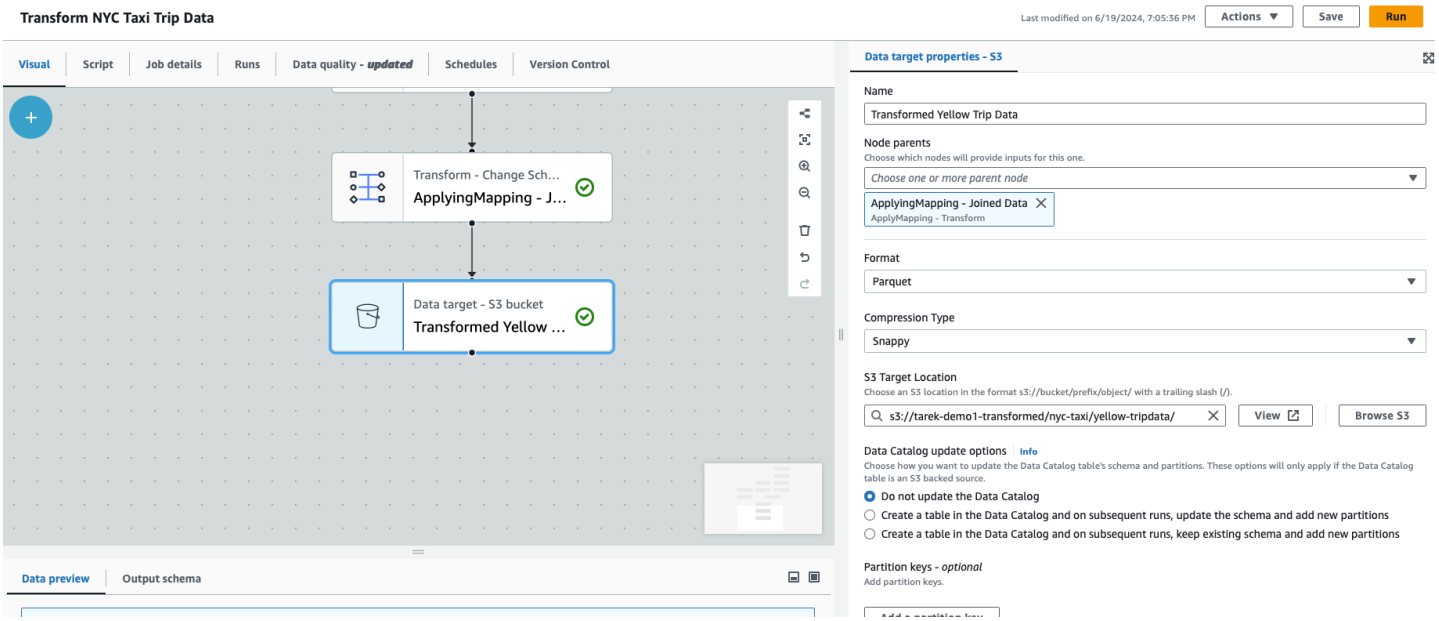
Save transformed data to Amazon S3



1. Click on **Targets** tab and choose **Amazon S3**
2. Specify the following information:
    - Node properties tab, **Name** - `Transformed Yellow Trip Data`
    - Data target properties - S3 tab, specify the following:

    - **Format** – `Glue Parquet`
    - **Compression Type** - `Snappy`
- **S3 Target Location** – `s3://[your-bucket-name]-transformed/nyc-taxi/yellow-tripdata/`



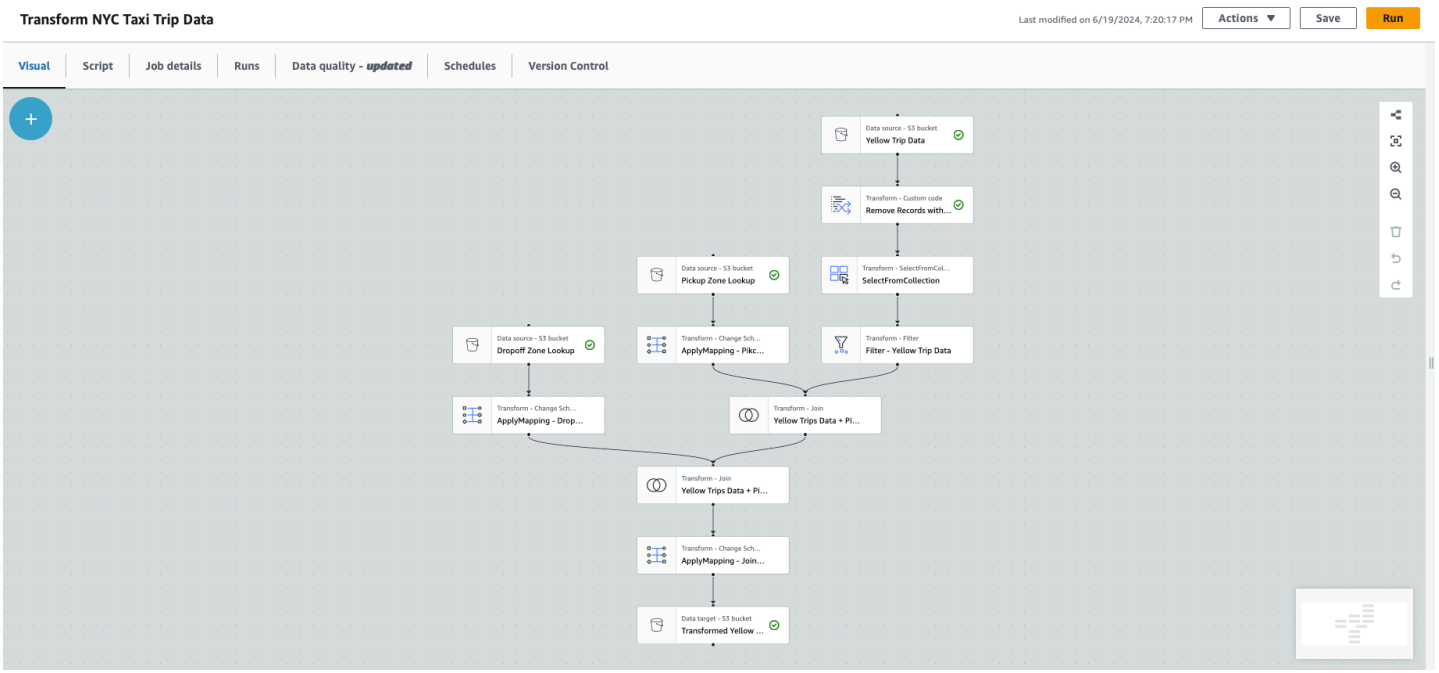


3. Remember to save your work.

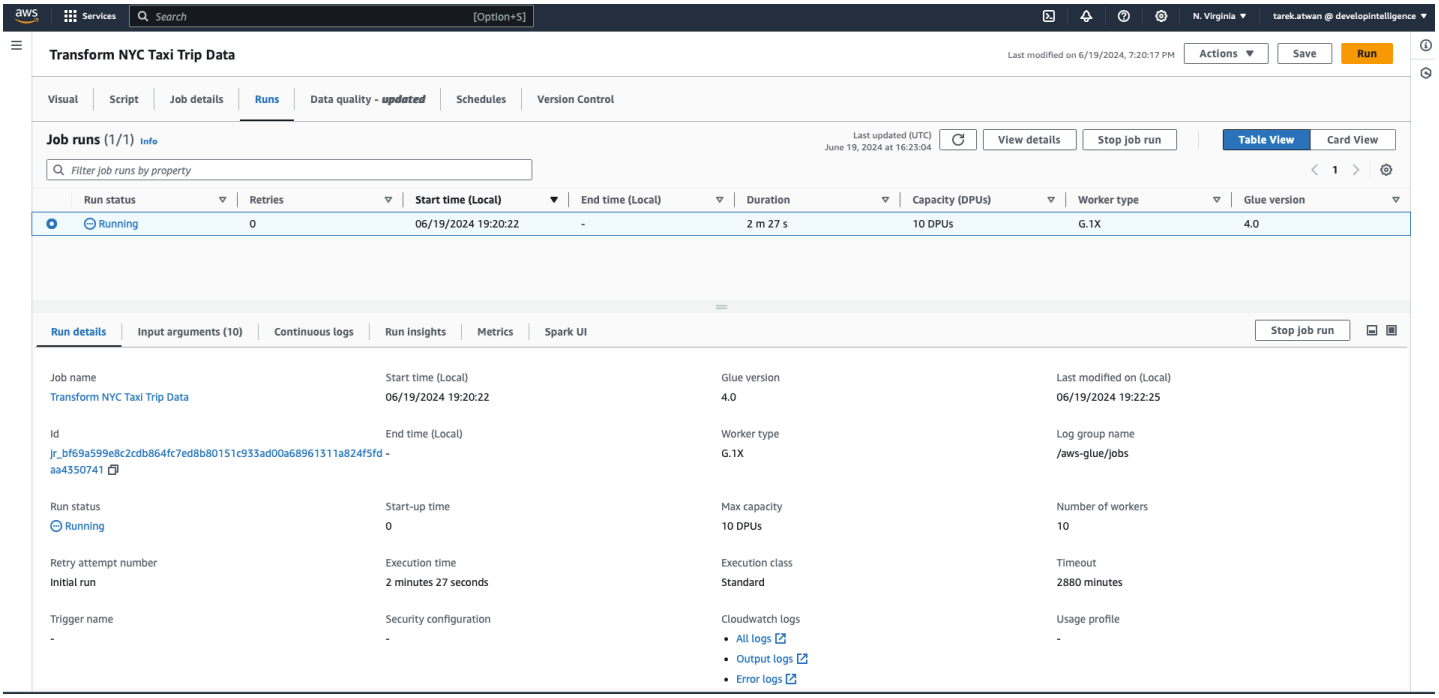
Run the Job

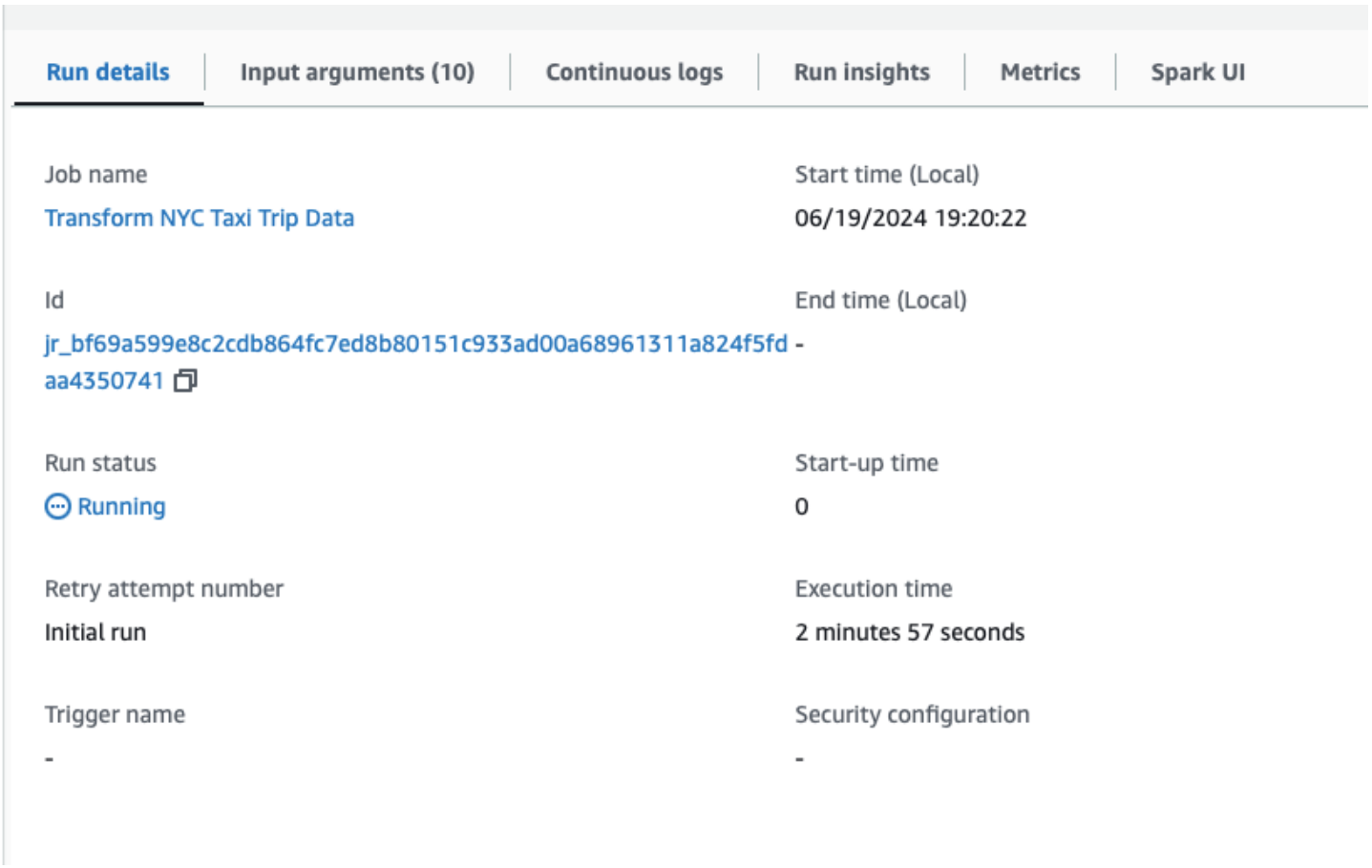
Run the data transformation job

1. On the upper right section of the AWS Glue Studio page, click **Run**.

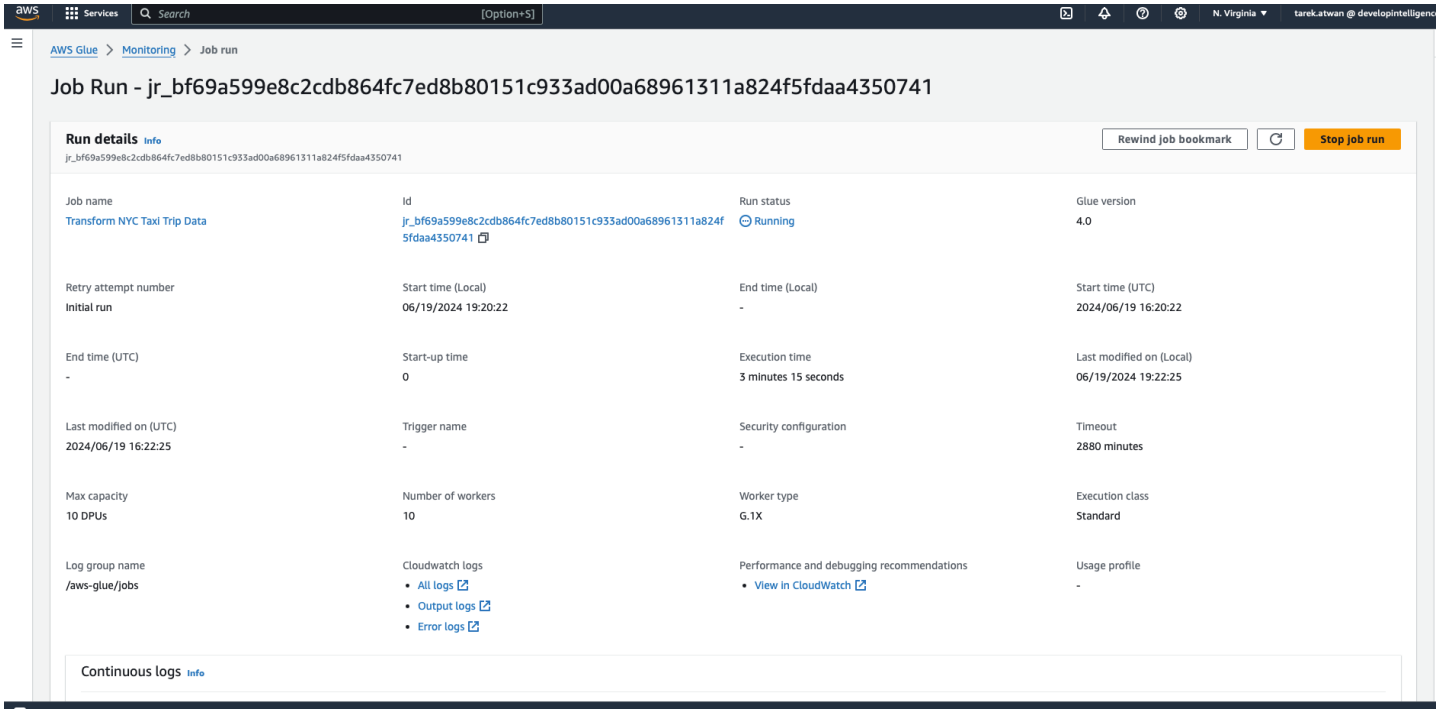


2. Go to the Runs tab to view the status on of the job.
3. Click on the **Job Id** to monitor the job run.





You can scroll down to see additional information about the job run and overall status



Continuous logs Info

Driver logs

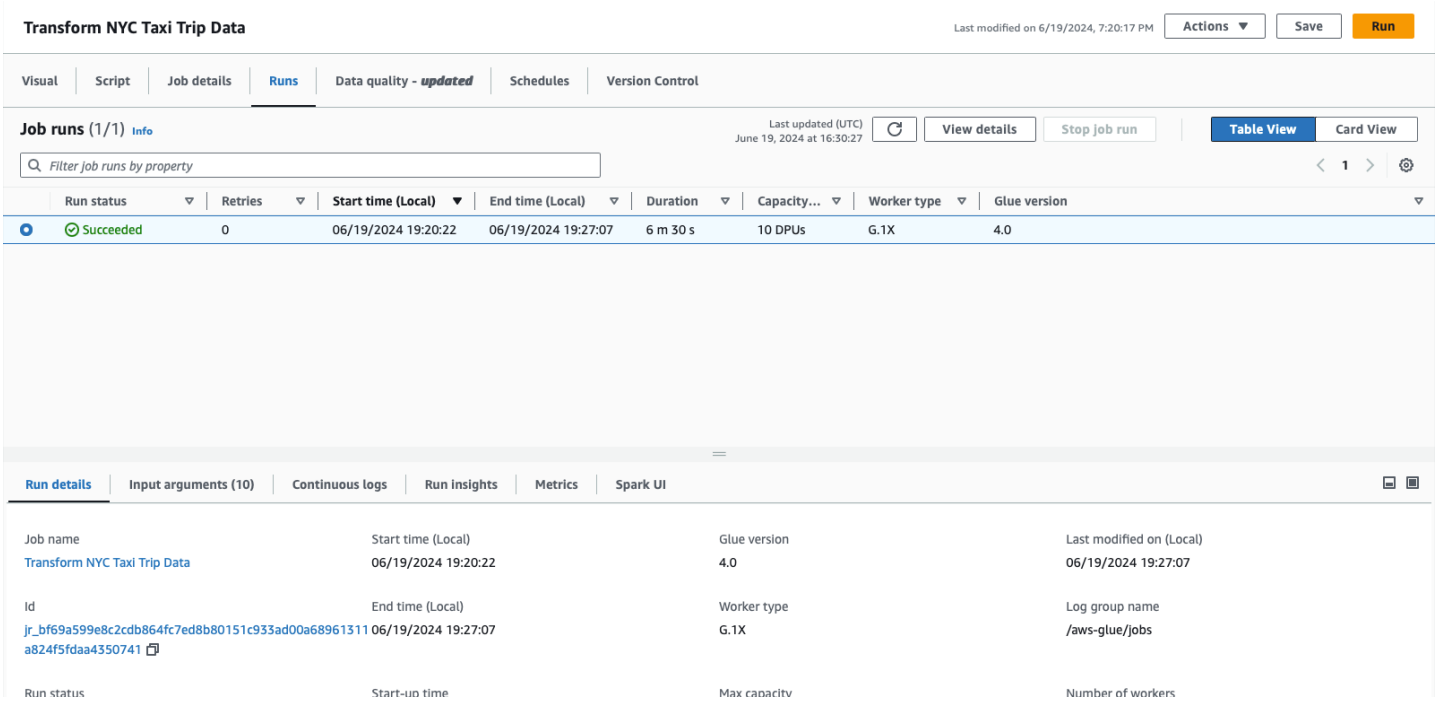
Driver and executor log streams

```
24/06/19 16:23:40 INFO DAGScheduler: Failed: Set()
24/06/19 16:23:40 INFO TaskSchedulerImpl: Removed TaskSet 4.0, whose tasks have all completed, from pool
24/06/19 16:23:40 INFO DAGScheduler: ShuffleMapStage 4 (map at DynamicFrame.scala:973) finished in 0.672 s
24/06/19 16:23:40 INFO TaskSetManager: Finished task 0.0 in stage 4.0 (TID 86) in 655 ms on 172.36.35.178 (executor 7) (1/1)
24/06/19 16:23:40 INFO TaskSetManager: Starting task 35.0 in stage 5.0 (TID 122) (172.36.35.178, executor 7, partition 35, ANY, 4694 bytes) taskResourceAssignments Map()
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_10_piece0 in memory on 172.34.163.232:43915 (size: 25.4 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_10_piece0 in memory on 172.34.105.73:34087 (size: 25.4 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_2_piece0 in memory on 172.36.35.178:38243 (size: 34.7 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_10_piece0 in memory on 172.34.4.7:39015 (size: 25.4 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_10_piece0 in memory on 172.38.32.85:33975 (size: 25.4 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_10_piece0 in memory on 172.38.227.19:45577 (size: 25.4 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_10_piece0 in memory on 172.36.35.178:38243 (size: 25.4 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_10_piece0 in memory on 172.35.189.113:36755 (size: 25.4 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_9_piece0 in memory on 172.36.35.178:38243 (size: 9.4 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_10_piece0 in memory on 172.35.83.109:34995 (size: 25.4 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO BlockManagerInfo: Added broadcast_10_piece0 in memory on 172.39.164.8:37993 (size: 25.4 KiB, free: 5.8 GiB)
24/06/19 16:23:39 INFO TaskSetManager: Starting task 34.0 in stage 5.0 (TID 121) (172.34.105.73, executor 3, partition 34, ANY, 4694 bytes) taskResourceAssignments Map()
24/06/19 16:23:39 INFO TaskSetManager: Starting task 32.0 in stage 5.0 (TID 119) (172.38.227.19, executor 6, partition 32, ANY, 4694 bytes) taskResourceAssignments Map()
```

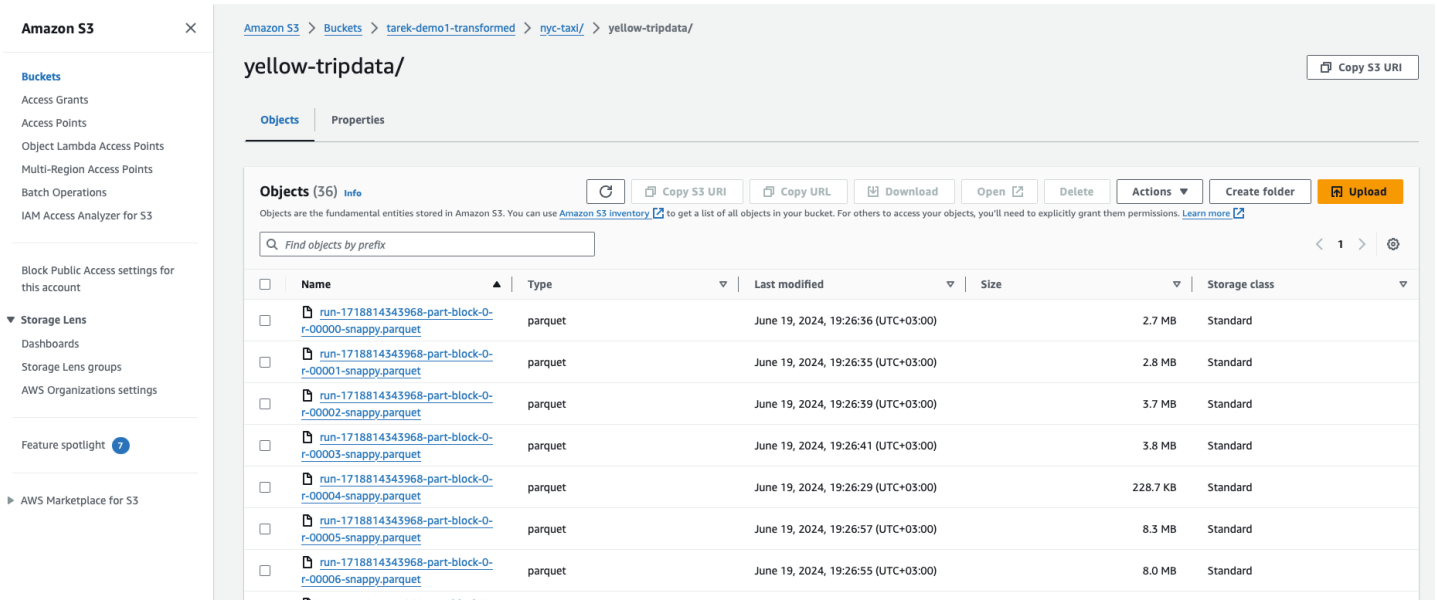
Input arguments (10)

Metrics Info

Once the job is completed you can further inspect the S3 bucket to see the newly populated data “transformed data”

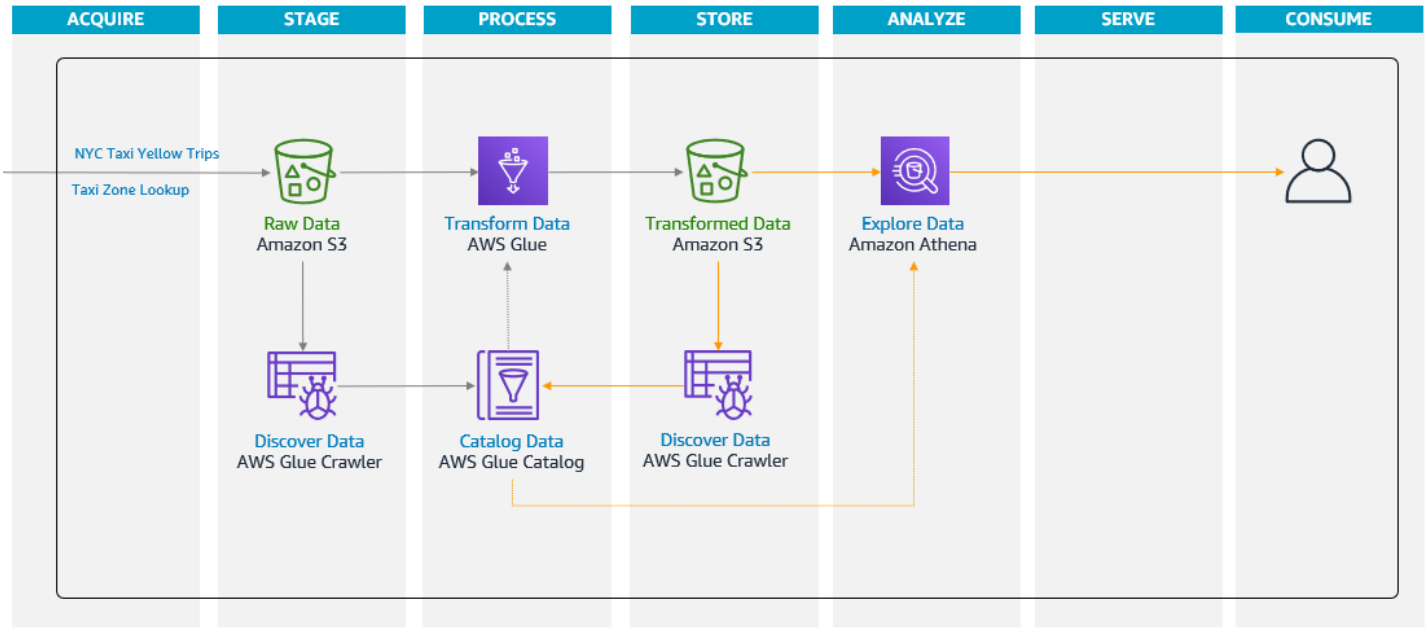


Here is an example of the parquet files in the destination/target bucket for the transformed data



Lab 4 - Enriching your Data

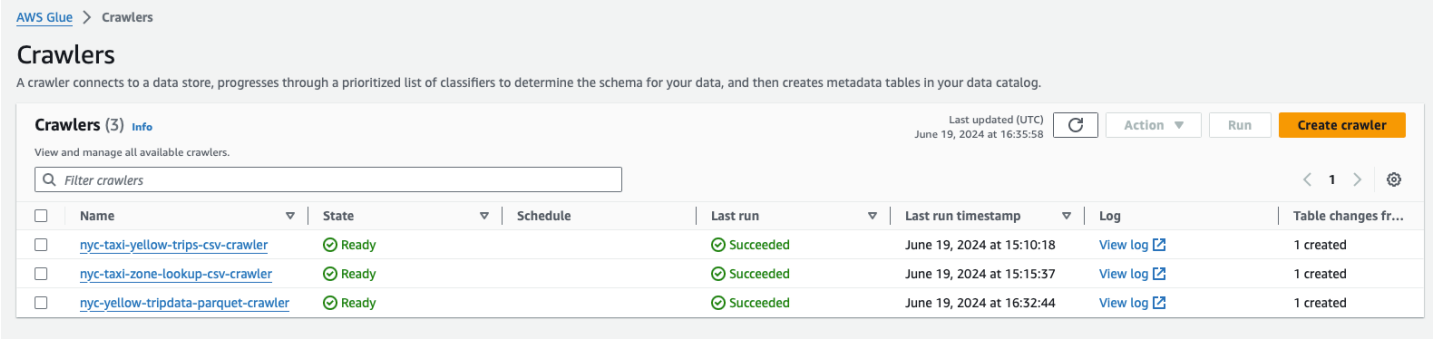
A picture is worth 1000 words. Data visualization presents data in graphical format which makes data easier to digest and understand. It allows users to gain better insights from data to make informed decisions.



Catalog Transformed Data

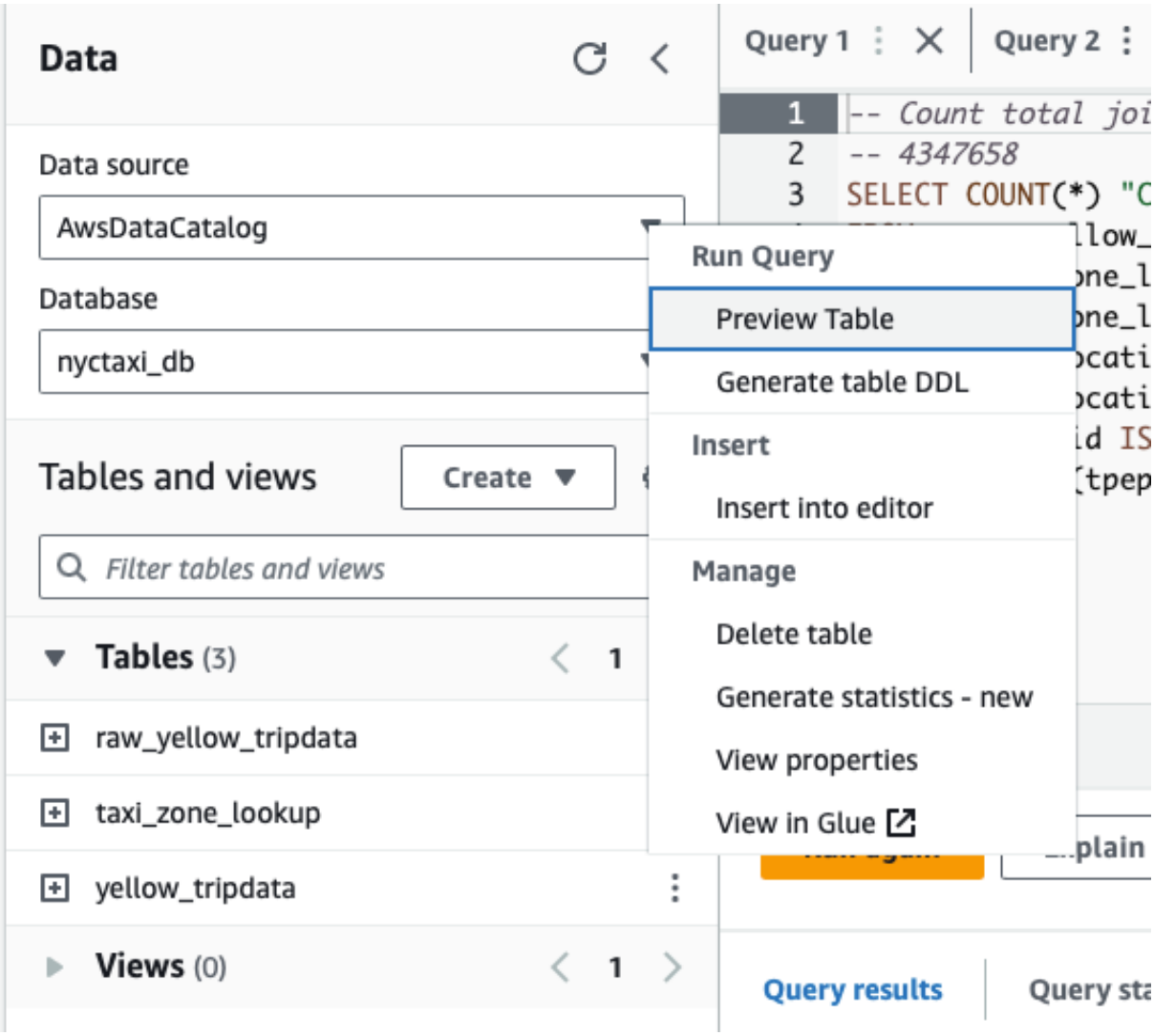
Create a Glue Crawler

1. Go to the AWS Glue Console.
2. In the left navigation menu, click **Crawlers**.
3. On the Crawlers page, click **add crawler**.
4. Specify `nyc-yellow-tripdata-parquet-crawler` as the crawler name, click **Next**.
5. On the Choose data sources and classifiers screen, specify the following information, and then click **Next**.
    - Click **Add a data source**
    - Choose Data source – **S3**
    - Select Location of S3 data - **In this account**
    - For Subsequent crawler runs, select to **Crawl all sub-folders**
    - Include S3 path – `s3://your-bucket-name-transformed/nyc-taxi/yellow-tripdata`
    - For Subsequent crawler runs, select to **Crawl all sub-folders**
    - Then click **Add an S3 data source**.
6. On the Configure security settings, choose **AWSGlueServiceRole-SDL-Jumpstart** from the Existing IAM role, click **Next**.
7. On the Set output and scheduling screen, choose **nyctaxi_db** as the database.
8. On the Crawler schedule, leave the frequency **On demand**, click **Next**.
9. Review the crawler details, click **Create crawler**.
10. On the Crawlers page, select **nyc-yellow-tripdata-parquet-crawler**, and then click **Run crawler**.



Validate transformed data

1. Go to Athena console.
2. Select **Preview table** to examine the data in **yellow_tripdata** table.



3. On the Query editor page, run the following queries to examine the data.

```sql
-- total records
-- 4347658
SELECT COUNT(*) "Count"
FROM   yellow_tripdata;
```

```sql
-- 2020-10-01 1575353
-- 2020-11-01 1409851
-- 2020-12-01 1362454
SELECT DATE_TRUNC('month', pickup_datetime) "Period",
       COUNT(*) "Total Records"
FROM   yellow_tripdata
GROUP BY DATE_TRUNC('month', pickup_datetime)
ORDER BY 1;
```

Enrich Transformed Data

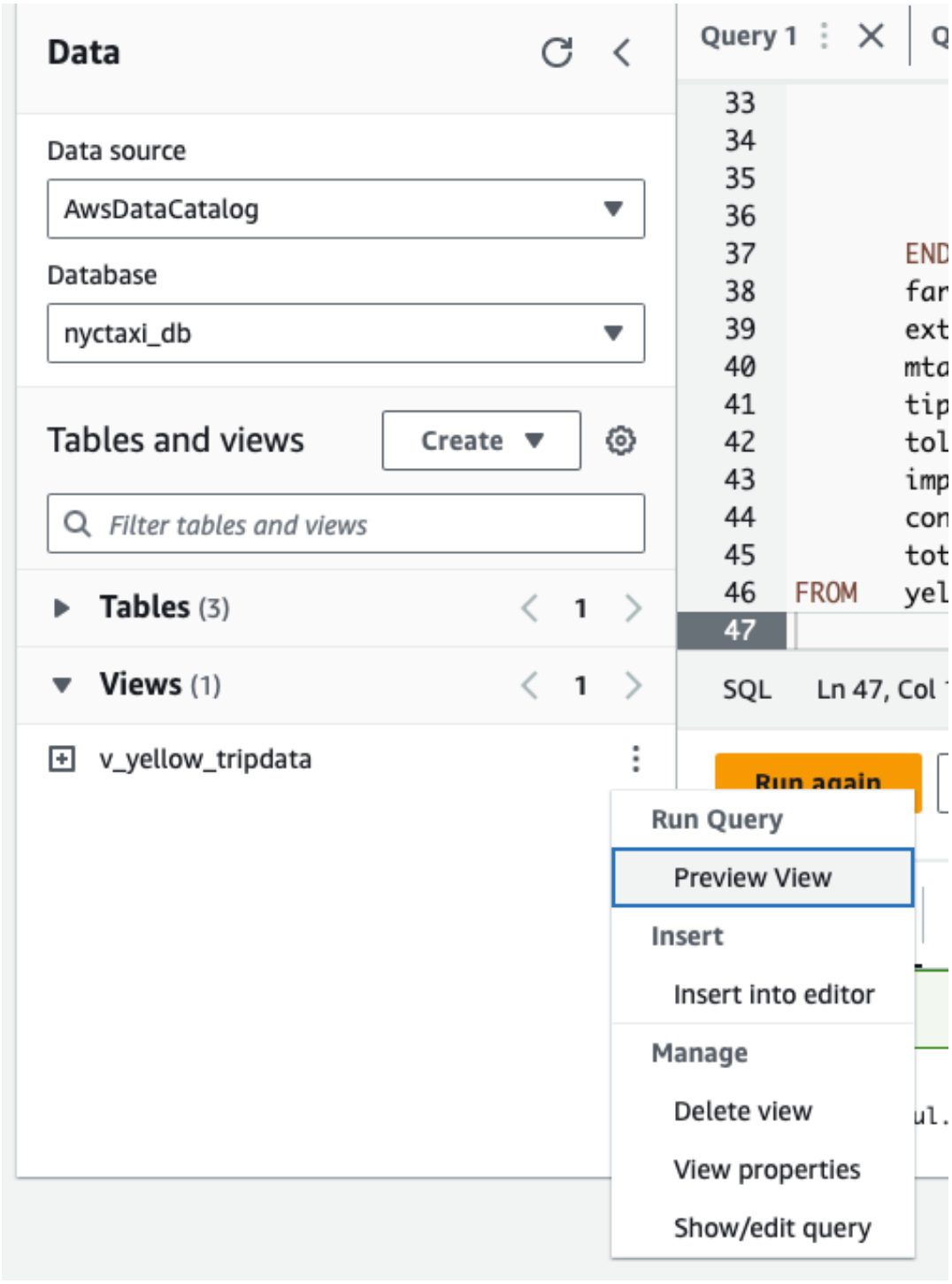
1. Run the following query to create a view to enrich the table with additional data.

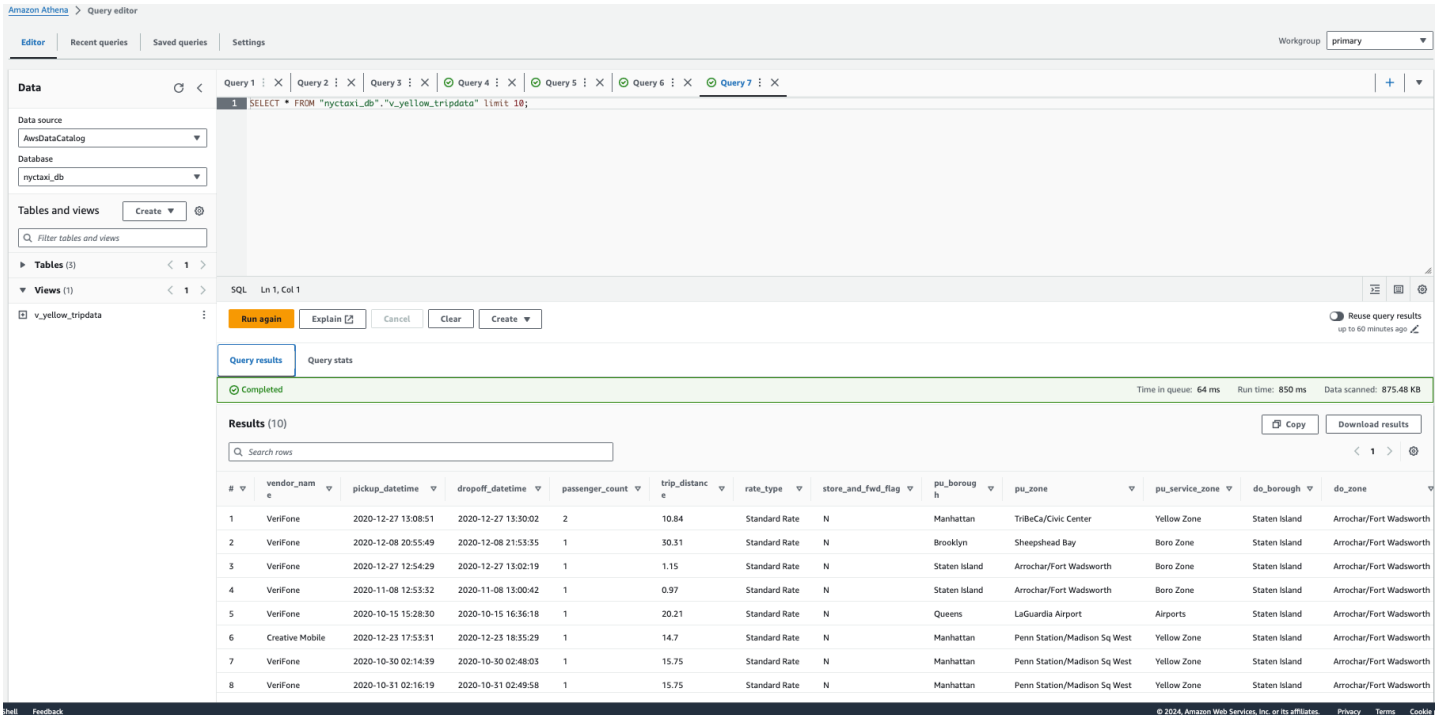
```sql
CREATE OR REPLACE VIEW v_yellow_tripdata
AS
SELECT CASE vendor_id
            WHEN 1 THEN 'Creative Mobile'
            WHEN 2 THEN 'VeriFone'
            ELSE 'No Data'
       END "vendor_name",
       pickup_datetime,
       dropoff_datetime,
       passenger_count,
       trip_distance,
       CASE ratecodeid
            WHEN 1 THEN 'Standard Rate'
            WHEN 2 THEN 'JFK'
            WHEN 3 THEN 'Newark'
            WHEN 4 THEN 'Nassau/Westchester'
            WHEN 5 THEN 'Negotiated Fare'
            WHEN 6 THEN 'Group Ride'
            WHEN 99 THEN 'Special Rate'
            ELSE 'No Data'
       END "rate_type",
       store_and_fwd_flag,
       pu_borough,
       pu_zone,
       pu_service_zone,
       do_borough,
```

```
        do_zone,
        do_service_zone,
        CASE payment_type
             WHEN 1 THEN 'Credit Card'
             WHEN 2 THEN 'Cash'
             WHEN 3 THEN 'No Charge'
             WHEN 4 THEN 'Dispute'
             WHEN 5 THEN 'Unknown'
             WHEN 6 THEN 'Voided Trip'
             ELSE 'No Data'
        END "payment_type",
        fare_amount,
        extra,
        mta_tax,
        tip_amount,
        tolls_amount,
        improvement_surcharge,
        congestion_surcharge,
        total_amount
FROM    yellow_tripdata;

```

2. Select **Preview table** to examine the enriched data in **v_yellow_tripdata** view.


Data
Data source
AwsDataCatalog
Database
nyctaxi_db
Tables and views
Create
Filter tables and views
Tables (3)
1
Views (1)
1
v_yellow_tripdata
Query 1
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
FROM
SQL
Ln 47, Col
Run again
Run Query
Preview View
Insert
Insert into editor
Manage
Delete view
View properties
Show/edit query



3. Run the following query to get insights.

```
SELECT vendor_name "Vendor",
       rate_type "Rate Type",
       payment_type "Payment Type",
       ROUND(AVG(fare_amount), 2) "Fare",
       ROUND(AVG(extra), 2) "Extra",
       ROUND(AVG(mta_tax), 2) "MTA",
       ROUND(AVG(tip_amount), 2) "Tip",
       ROUND(AVG(tolls_amount), 2) "Toll",
       ROUND(AVG(improvement_surcharge), 2) "Improvement",
       ROUND(AVG(congestion_surcharge), 2) "Congestion",
       ROUND(AVG(total_amount), 2) "Total"
FROM   v_yellow_tripdata
GROUP BY vendor_name,
         rate_type,
         payment_type
ORDER BY 1, 2, 3;

```