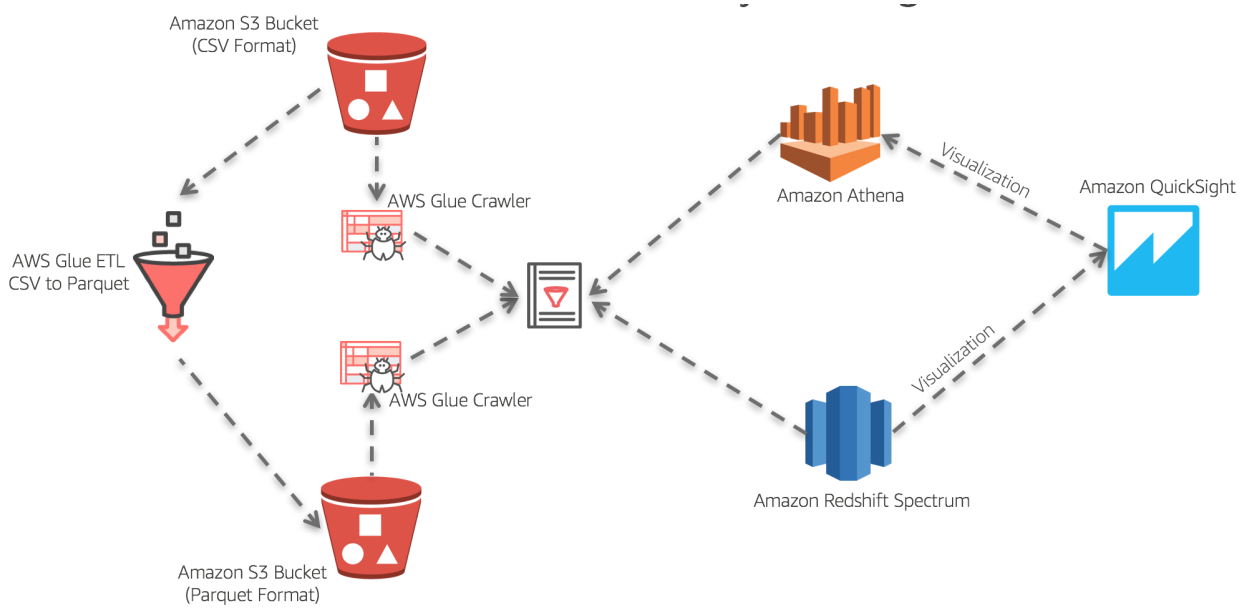
Building an End-to-End Serverless Data Analytics Solution on AWS

Overview

In this lab, you are going to build a serverless architecture to analyze the data directly from Amazon S3 using Amazon Athena and visualize the data in Amazon QuickSight. The data set that you are going to use is a public data set that includes trip records from all trips completed in Yellow and Green taxis in NYC from 2009 to 2016, and all trips in for-hire vehicles (FHV) from 2015 to 2016. Records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data set is already partitioned and converted from CSV to Apache Parquet. In the first part of the lab you will be building SQL like queries using Amazon Athena to query both the data formats directly form Amazon S3 and comparing the query performance. In the second part, using Amazon Athena table, create in first part, as the data source for Amazon QuickSight you will generate visualization and meaningful insights from the data set in Amazon S3. An optional lab is included to incorporate serverless ETL using AWS Glue to optimize query performance. We also give you access to a take-home lab for you to reapply the same design and directly query the same dataset in Amazon S3 from an Amazon Redshift data warehouse using Redshift Spectrum.



AWS Console

Verifying your region in the AWS Management Console

With Amazon Ec2, you can place instances in multiple locations. Amazon EC2 locations are composed of regions that contain more that one Availability Zones. Regions are dispersed and located in separate geographic areas (US, EU, etc.). Availability Zones are distinct locations within a region. They are are engineered to be isolated from failures in other Availability Zones and to provide inexpensive, low-latency network connectivity to other Availability Zones in the same region.

By launching instances in separate regions, you can design your application to be closer to specific customers or to meet legal or other requirements. By launching instances in separate Availability Zones, you can protect your application from localized regional failures.

Verify your Region

The AWS region name is always listed in the upper-right corner of the AWS Management Console, in the navigation bar.

- Make a note of the AWS region name, for example, for this lab you will need to choose the **US West (Oregon)** region.
- Use the chart below to determine the region code. Choose **us-west-2 for this lab.**

Region Name	Region Code
US East (Northern Virginia) Region	us-east-1
US West (Oregon) Region	us-west-2
Asia Pacific (Tokyo) Region	ap-northeast-1
Asia Pacific (Seoul) Region	ap-northeast-2
Asia Pacific (Singapore) Region	ap-southeast-1
Asia Pacific (Sydney) Region	ap-southeast-2
EU (Ireland) Region	eu-west-1
EU (Frankfurt) Region	eu-central-1

Labs

Pre-requisites

- Working, non-production AWS Account if you don't have one.

Lab	Name
Lab 1	Analysis of data in Amazon S3 using Amazon Athena
Lab 2	Visualization using Amazon QuickSight
Lab 3	ETL and Data Discovery using Amazon Glue

AMAZON ATHENA

What is 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. To get started, just log into the Athena Management Console, define your schema, and start querying. Amazon Athena

uses Presto with full standard SQL support and works with a variety of standard data formats, including CSV, JSON, ORC, Apache Parquet and Avro. While Amazon Athena is ideal for quick, ad-hoc querying and integrates with Amazon QuickSight for easy visualization, it can also handle complex analysis, including large joins, window functions, and arrays.

What can I do with Amazon Athena?

Amazon Athena helps you analyze data stored in Amazon S3. You can use Athena to run ad-hoc queries using ANSI SQL, without the need to aggregate or load the data into Athena. Amazon Athena can process unstructured, semi-structured, and structured data sets. Examples include CSV, JSON, Avro or columnar data formats such as Apache Parquet and Apache ORC. Amazon Athena integrates with Amazon QuickSight for easy visualization. You can also use Amazon Athena to generate reports or to explore data with business intelligence tools or SQL clients, connected via a JDBC driver.

How do you access Amazon Athena?

Amazon Athena can be accessed via the AWS management console and a JDBC driver. You can programmatically run queries, add tables or partitions using the JDBC driver.

What is the underlying technology behind Amazon Athena?

Amazon Athena uses Presto with full standard SQL support and works with a variety of standard data formats, including CSV, JSON, ORC, Avro, and Parquet. Athena can handle complex analysis, including large joins, window functions, and arrays. Because Amazon Athena uses Amazon S3 as the underlying data store, it is highly available and durable with data redundantly stored across multiple facilities and multiple devices in each facility.

How do I create tables and schemas for my data on Amazon S3?

Amazon Athena uses Apache Hive DDL to define tables. You can run DDL statements using the Athena console, via a JDBC driver, or using the Athena create table wizard. When you create a new table schema in Amazon Athena the schema is stored in the data catalog and used when executing queries, but it does not modify your data in S3. Athena uses an approach known as schema-on-read, which allows you to project your schema onto your data at the time you execute a query. This eliminates the need for any data loading or transformation. Learn more about creating tables.

What data formats does Amazon Athena support?

Amazon Athena supports a wide variety of data formats like CSV, TSV, JSON, or Textfiles and also supports open source columnar formats such as Apache ORC and Apache Parquet. Athena also supports compressed data in Snappy, Zlib, LZO, and GZIP formats. By compressing, partitioning, and using columnar formats you can improve performance and reduce your costs.

For more details refer [Amazon Athena FAQ] (https://aws.amazon.com/athena/faqs/)

AMAZON QUICKSIGHT

What is Amazon QuickSight?

Amazon QuickSight is a fast, cloud-powered business analytics service that makes it easy to build visualizations, perform ad-hoc analysis, and quickly get business insights from your data. Using our cloud-based service you

can easily connect to your data, perform advanced analysis, and create stunning visualizations and rich dashboards that can be accessed from any browser or mobile device.

How is Amazon QuickSight different from traditional Business Intelligence (BI) solutions?

Traditional BI solutions often require teams of data engineers to spend months building complex data models before generating a report. They typically lack interactive ad-hoc data exploration and visualization, limiting users to canned reports and pre-selected queries. Traditional BI solutions also require significant up-front investment in complex and costly hardware and software, and then customers to invest in even more infrastructure to maintain fast query performance as database sizes grow. This cost and complexity makes it difficult for companies to enable analytics solutions across their organizations. Amazon QuickSight has been designed to solve these problems by bringing the scale and flexibility of the AWS Cloud to business analytics. Unlike traditional BI or data discovery solutions, getting started with Amazon QuickSight is simple and fast. When you log in, Amazon QuickSight seamlessly discovers your data sources in AWS services such as Amazon Redshift, Amazon RDS, Amazon Athena, and Amazon Simple Storage Service (Amazon S3). You can connect to any of the data sources discovered by Amazon QuickSight and get insights from this data in minutes. You can choose for Amazon QuickSight to keep the data in SPICE up-to-date as the data in the underlying sources change. SPICE supports rich data discovery and business analytics capabilities to help customers derive valuable insights from their data without worrying about provisioning or managing infrastructure. Organizations pay a low monthly fee for each Amazon QuickSight user, eliminating the cost of long-term licenses. With Amazon QuickSight, organizations can deliver rich business analytics functionality to all employees without incurring a huge cost upfront.

Which data sources does Amazon QuickSight support?

You can connect to AWS data sources including Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon Athena and Amazon S3. You can also upload Excel spreadsheets or flat files (CSV, TSV, CLF, and ELF), connect to on-premises databases like SQL Server, MySQL and PostgreSQL and import data from SaaS applications like Salesforce.

For more details refer Amazon QuickSight FAQ

AWS Glue

What is AWS Glue?

AWS Glue is a fully-managed, pay-as-you-go, extract, transform, and load (ETL) service that automates the time-consuming steps of data preparation for analytics. AWS Glue automatically discovers and profiles your data via the Glue Data Catalog, recommends and generates ETL code to transform your source data into target schemas, and runs the ETL jobs on a fully managed, scale-out Apache Spark environment to load your data into its destination. It also allows you to setup, orchestrate, and monitor complex data flows.

What are the main components of AWS Glue?

AWS Glue consists of a Data Catalog which is a central metadata repository, an ETL engine that can automatically generate Python code, and a flexible scheduler that handles dependency resolution, job monitoring, and retries. 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.

When should I use AWS Glue?

You should use AWS Glue to discover properties of the data you own, transform it, and prepare it for analytics. Glue can automatically discover both structured and semi-structured data stored in your data lake on Amazon S3, data warehouse in Amazon Redshift, and various databases running on AWS. It provides a unified view of your data via the Glue Data Catalog that is available for ETL, querying and reporting using services like Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. Glue automatically generates Python code for your ETL jobs that you can further customize using tools you are already familiar with. AWS Glue is serverless, so there are no compute resources to configure and manage.

What data sources does AWS Glue support?

AWS Glue natively supports data stored in Amazon Aurora, Amazon RDS for MySQL, Amazon RDS for Oracle, Amazon RDS for PostgreSQL, Amazon RDS for SQL Server, Amazon Redshift, and Amazon S3, as well as MySQL, Oracle, Microsoft SQL Server, and PostgreSQL databases in your Virtual Private Cloud (Amazon VPC) running on Amazon EC2. The metadata stored in the AWS Glue Data Catalog can be readily accessed from Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. You can also write custom PySpark code and import custom libraries in your Glue ETL jobs to access data sources not natively supported by AWS Glue. For more details on importing custom libraries, refer to our documentation.

What is the AWS Glue Data Catalog?

The AWS Glue Data Catalog is a central repository to store structural and operational metadata for all your data assets. For a given data set, you can store its table definition, physical location, add business relevant attributes, as well as track how this data has changed over time.

The AWS Glue Data Catalog is Apache Hive Metastore compatible and is a drop-in replacement for the Apache Hive Metastore for Big Data applications running on Amazon EMR. For more information on setting up your EMR cluster to use AWS Glue Data Catalog as an Apache Hive Metastore, click here.

The AWS Glue Data Catalog also provides out-of-box integration with Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. Once you add your table definitions to the Glue Data Catalog, they are available for ETL and also readily available for querying in Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum so that you can have a common view of your data between these services.

How can I customize the ETL code generated by AWS Glue?

AWS Glue's ETL script recommendation system generates PySpark code. It leverages Glue's custom ETL library to simplify access to data sources as well as manage job execution. You can find more details about the library in our documentation. You can write ETL code using AWS Glue's custom library or write arbitrary Spark code in Python (PySpark code) by using inline editing via the AWS Glue Console script editor, downloading the auto-generated code, and editing it in your own IDE. You can also start with one of the many samples hosted in our Github repository and customize that code.

When should I use AWS Glue vs. AWS Data Pipeline?

AWS Glue provides a managed ETL service that runs on a serverless Apache Spark environment. This allows you to focus on your ETL job and not worry about configuring and managing the underlying compute resources. AWS Glue takes a data first approach and allows you to focus on the data properties and data manipulation to

transform the data to a form where you can derive business insights. It provides an integrated data catalog that makes metadata available for ETL as well as querying via Amazon Athena and Amazon Redshift Spectrum.

AWS Data Pipeline provides a managed orchestration service that gives you greater flexibility in terms of the execution environment, access and control over the compute resources that run your code, as well as the code itself that does data processing. AWS Data Pipeline launches compute resources in your account allowing you direct access to the Amazon EC2 instances or Amazon EMR clusters.

Furthermore, AWS Glue ETL jobs are PySpark based. If your use case requires you to use an engine other than Apache Spark or if you want to run a heterogeneous set of jobs that run on a variety of engines like Hive, Pig, etc., then AWS Data Pipeline would be a better choice.

For more details refer AWS Glue FAQ

ADDITIONAL RESOURCES

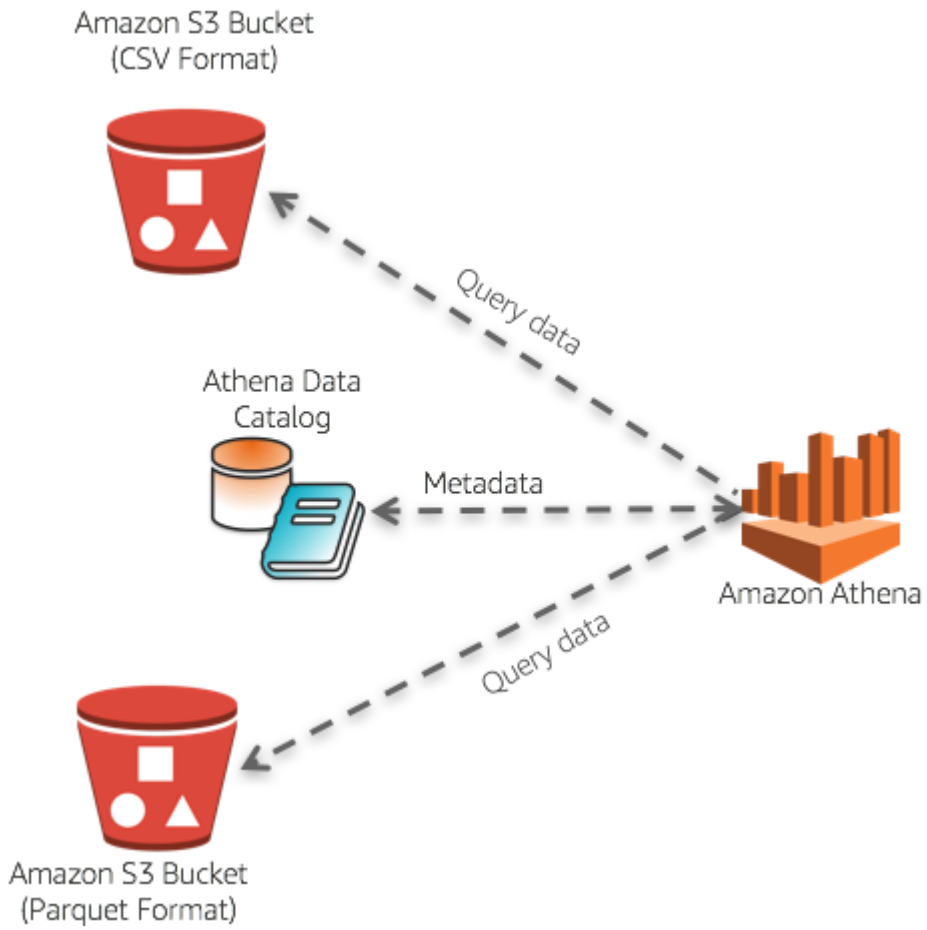
- https://aws.amazon.com/blogs/big-data/top-10-performance-tuning-tips-for-amazon-athena/
- http://docs.aws.amazon.com/athena/latest/ug/convert-to-columnar.html
- https://aws.amazon.com/blogs/big-data/streamline-aws-cloudtrail-log-visualization-using-aws-glue-and-amazon-quicksight/
- https://aws.amazon.com/blogs/big-data/build-a-data-lake-foundation-with-aws-glue-and-amazon-s3/
- https://aws.amazon.com/blogs/big-data/derive-insights-from-iot-in-minutes-using-aws-iot-amazon-kinesis-firehose-amazon-athena-and-amazon-quicksight/
- https://aws.amazon.com/blogs/big-data/build-a-serverless-architecture-to-analyze-amazon-cloudfront-access-logs-using-aws-lambda-amazon-athena-and-amazon-kinesis-analytics/

License

Lab 1: Analysis of data in Amazon S3 using Amazon Athena

- Creating Amazon Athena Database and Table
  - Create Athena Database
  - Create Athena Table
- Querying data from Amazon S3 using Amazon Athena
- Querying partitioned data using Amazon Athena
  - Create Athena Table with Partitions
  - Adding partition metadata to Amazon Athena
  - Querying partitioned data set
- Summary

Architectural Diagram



Creating Amazon Athena Database and Table

Amazon Athena uses Apache Hive to define tables and create databases. Databases are a logical grouping of tables. When you create a database and table in Athena, you are simply describing the schema and location of the table data in Amazon S3. In case of Hive, databases and tables don't store the data along with the schema definition unlike traditional relational database systems. The data is read from Amazon S3 only when you query the table. The other benefit of using Hive is that the metastore found in Hive can be used in many other big data

applications such as Spark, Hadoop, and Presto. With Athena catalog, you can now have Hive-compatible metastore in the cloud without the need for provisioning a Hadoop cluster or RDS instance. For guidance on databases and tables creation refer Apache Hive documentation. The following steps provides guidance specifically for Amazon Athena.

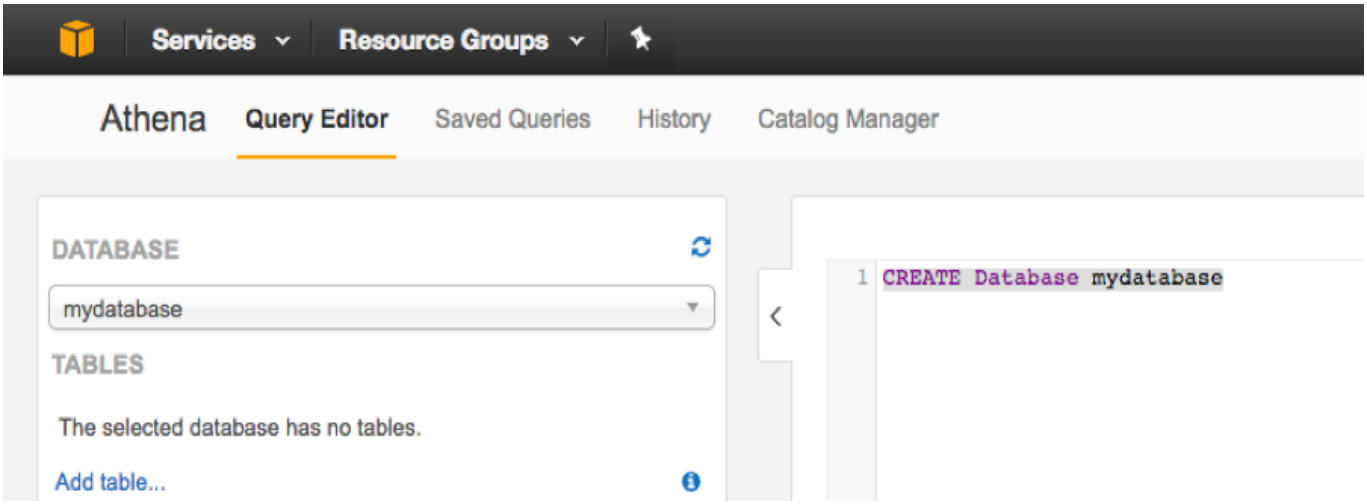
Create Database

1. Open the AWS Management Console for Athena.
2. If this is your first time visiting the AWS Management Console for Athena, you will get a Getting Started page. Choose **Get Started** to open the Query Editor. If this isn't your first time, the Athena **Query Editor** opens.
3. Make a note of the AWS region name, for example, for this lab you will need to choose the **US West (Oregon)** region.
4. In the Athena **Query Editor**, you will see a query pane with an example query. Now you can start entering your query in the query pane.
5. To create a database named *mydatabase*, copy the following statement, and then choose **Run Query**:

```
CREATE DATABASE <username>
```

> **Note:** Where `username` is the AIM user you are logged into the console as.

6. Ensure `<username>`appears in the DATABASE list on the **Catalog** dashboard.



Create a Table

Now that you have a database, you are ready to create a table that is based on the New York taxi sample data. You define columns that map to the data, specify how the data is delimited, and provide the location in Amazon S3 for the file.

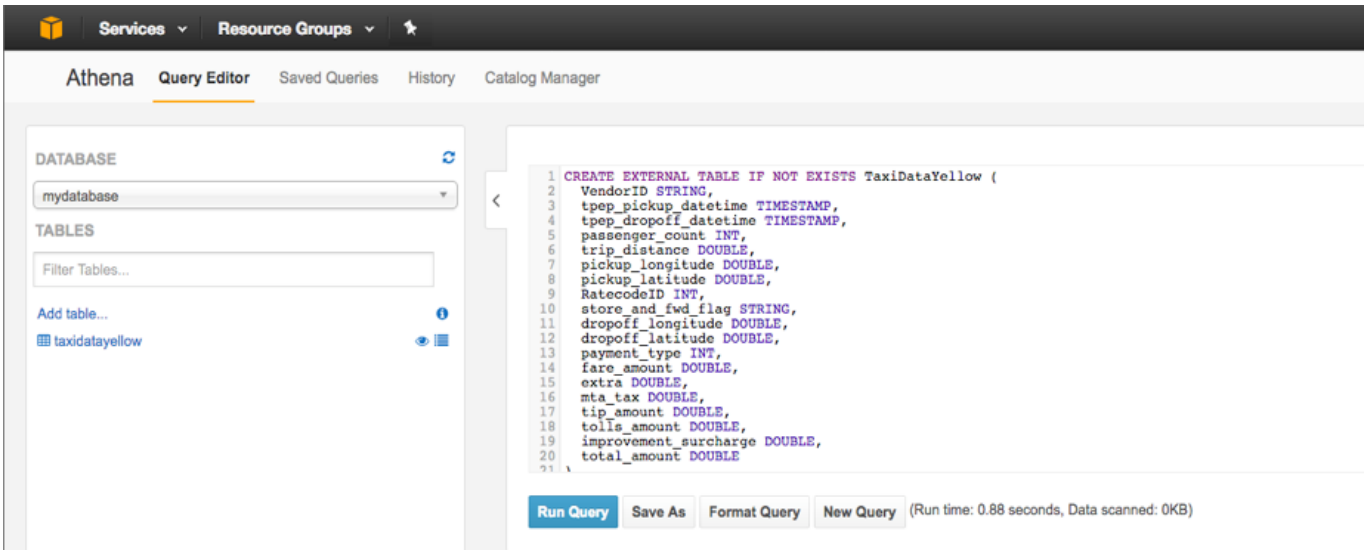
> **Note:** When creating the table, you need to consider the following:
>
> - You must have the appropriate permissions to work with data in the Amazon S3 location. For more information, refer Setting User and Amazon S3 Bucket Permissions.
> - The data can be in a different region from the primary region where you run Athena as long as the data is not encrypted in Amazon S3. Standard inter-region data transfer rates for Amazon S3 apply

> in addition to standard Athena charges.
> - If the data is encrypted in Amazon S3, it must be in the same region, and the user or principal who creates the table must have the appropriate permissions to decrypt the data. For more information, refer Configuring Encryption Options.
> - Athena does not support different storage classes within the bucket specified by the LOCATION clause, does not support the GLACIER storage class, and does not support Requester Pays buckets. For more information, see Storage Classes,Changing the Storage Class of an Object in Amazon S3, and Requester Pays Buckets in the Amazon Simple Storage Service Developer Guide.

1. Ensure that current AWS region is **US West (Oregon)** region
2. Ensure the <username> database is selected from the **DATABASE** list and then choose **New Query**.
3. In the query pane, copy the following statement to create TaxiDataYellow table, and then choose **Run Query**:

```
CREATE EXTERNAL TABLE IF NOT EXISTS taxi_csv(
    VendorID STRING,
    tpep_pickup_datetime TIMESTAMP,
    tpep_dropoff_datetime TIMESTAMP,
    passenger_count INT,
    trip_distance DOUBLE,
    pickup_longitude DOUBLE,
    pickup_latitude DOUBLE,
    RatecodeID INT,
    store_and_fwd_flag STRING,
    dropoff_longitude DOUBLE,
    dropoff_latitude DOUBLE,
    payment_type INT,
    fare_amount DOUBLE,
    extra DOUBLE,
    mta_tax DOUBLE,
    tip_amount DOUBLE,
    tolls_amount DOUBLE,
    improvement_surcharge DOUBLE,
    total_amount DOUBLE
    )
    ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
    STORED AS TEXTFILE
    LOCATION 's3://us-west-2.serverless-analytics/NYC-Pub/yellow/'
```

> **Note:**
>
> - If you use CREATE TABLE without the EXTERNAL keyword, you will get an error as only tables with the EXTERNAL keyword can be created in Amazon Athena. We recommend that you always use the EXTERNAL keyword. When you drop a table, only the table metadata is removed and the data remains in Amazon S3.
> - You can also query data in regions other than the region where you are running Amazon Athena. Standard inter-region data transfer rates for Amazon S3 apply in addition to standard Amazon Athena charges.
> - Ensure the table you just created appears on the Catalog dashboard for the selected database.



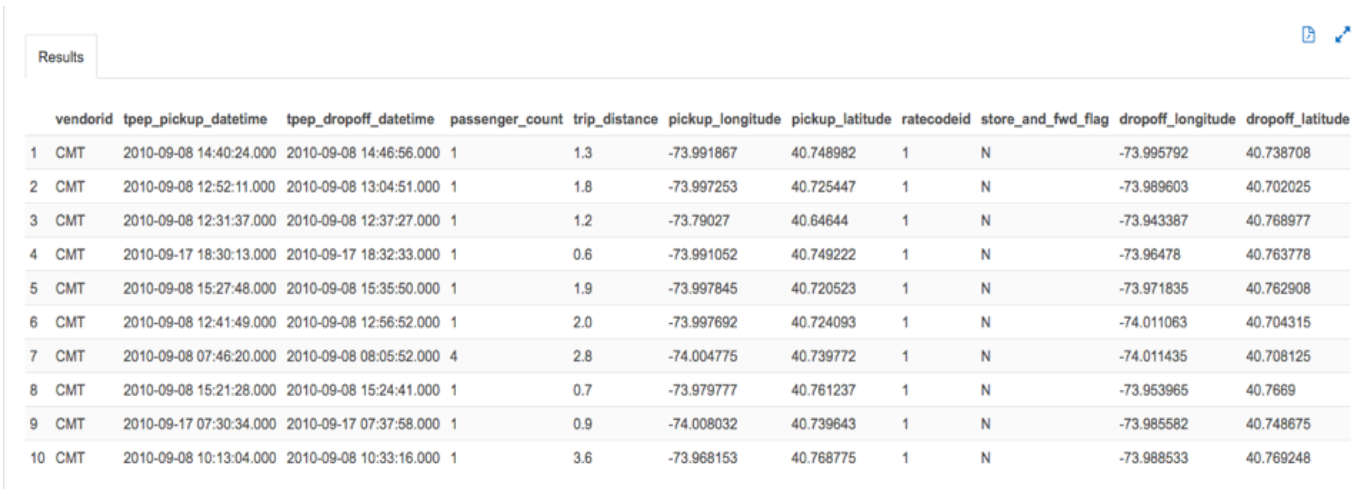
Querying data from Amazon S3 using Amazon Athena

Now that you have created the table, you can run queries on the data set and see the results in AWS Management Console for Amazon Athena.

1. Choose **New Query**, copy the following statement into the query pane, and then choose **Run Query**.

```
SELECT * FROM taxi_csv limit 10
```

Results for the above query look like the following:



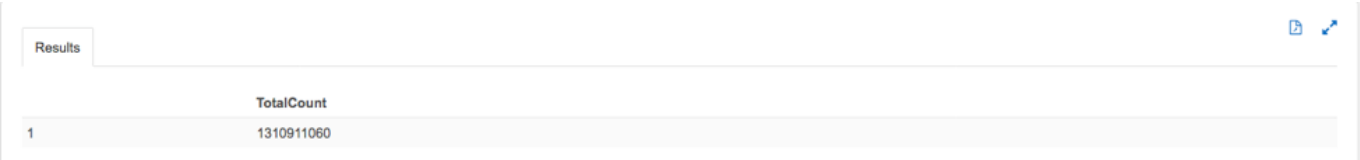
Results

	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	pickup_longitude	pickup_latitude	ratecodeid	store_and_fwd_flag	dropoff_longitude	dropoff_latitude
1	CMT	2010-09-08 14:40:24.000	2010-09-08 14:46:56.000	1	1.3	-73.991867	40.748982	1	N	-73.995792	40.738708
2	CMT	2010-09-08 12:52:11.000	2010-09-08 13:04:51.000	1	1.8	-73.997253	40.725447	1	N	-73.989603	40.702025
3	CMT	2010-09-08 12:31:37.000	2010-09-08 12:37:27.000	1	1.2	-73.79027	40.64644	1	N	-73.943387	40.768977
4	CMT	2010-09-17 18:30:13.000	2010-09-17 18:32:33.000	1	0.6	-73.991052	40.749222	1	N	-73.96478	40.763778
5	CMT	2010-09-08 15:27:48.000	2010-09-08 15:35:50.000	1	1.9	-73.997845	40.720523	1	N	-73.971835	40.762908
6	CMT	2010-09-08 12:41:49.000	2010-09-08 12:56:52.000	1	2.0	-73.997692	40.724093	1	N	-74.011063	40.704315
7	CMT	2010-09-08 07:46:20.000	2010-09-08 08:05:52.000	4	2.8	-74.004775	40.739772	1	N	-74.011435	40.708125
8	CMT	2010-09-08 15:21:28.000	2010-09-08 15:24:41.000	1	0.7	-73.979777	40.761237	1	N	-73.953965	40.7669
9	CMT	2010-09-17 07:30:34.000	2010-09-17 07:37:58.000	1	0.9	-74.008032	40.739643	1	N	-73.985582	40.748675
10	CMT	2010-09-08 10:13:04.000	2010-09-08 10:33:16.000	1	3.6	-73.968153	40.768775	1	N	-73.988533	40.769248

2. Choose **New Query**, copy the following statement into the query pane, and then choose **Run Query** to get the total number of taxi rides for yellow cabs.

```
SELECT COUNT(1) as TotalCount FROM taxi_csv
```

Results for the above query look like the following:



Results

	TotalCount
1	1310911060

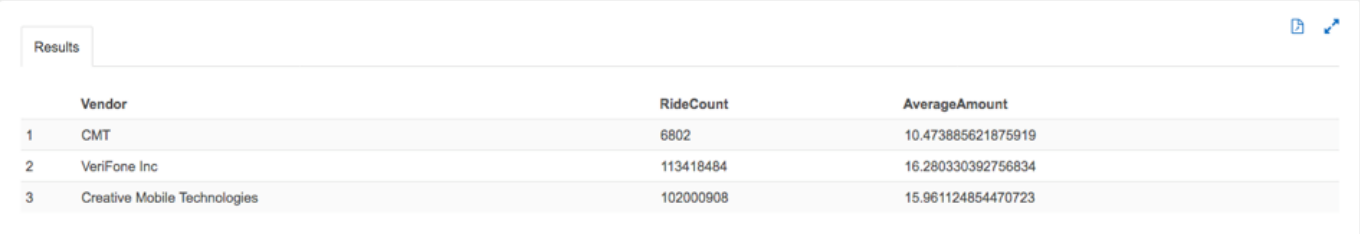
> **Note:** The current data format is CSV and this query is scanning **~207GB** of data and takes **~14.6** seconds to execute the query.

3. Make a note of query execution time for later comparison while querying the data set in Apache Parquet format in Lab 4.

4. Choose **New Query**, copy the following statement into the query pane, and then choose **Run Query** to query for the number of rides per vendor, along with the average fair amount for yellow taxi rides

```
SELECT
CASE vendorid
    WHEN '1' THEN 'Creative Mobile Technologies'
    WHEN '2' THEN 'VeriFone Inc'
    ELSE vendorid END AS Vendor,
COUNT(1) as RideCount,
avg(total_amount) as AverageAmount
FROM taxi_csv
WHERE total_amount > 0
GROUP BY (1)
```

Results for the above query should look similar the following:



Results

	Vendor	RideCount	AverageAmount
1	CMT	6802	10.473885621875919
2	VeriFone Inc	113418484	16.280330392756834
3	Creative Mobile Technologies	102000908	15.961124854470723

> **Note:** The current data format is CSV and this query is scanning **~207GB** of data and takes **~17.1** seconds to execute the query.

Querying partitioned data using Amazon Athena

By partitioning your data, you can restrict the amount of data scanned by each query, thus improving performance and reducing cost. Athena leverages Hive for partitioning data. You can partition your data by any key. A common practice is to partition the data based on time, often leading to a multi-level partitioning scheme. For example, a customer who has data coming in every hour might decide to partition by year, month, date, and hour. Another customer, who has data coming from many different sources but loaded one time per day, may partition by a data source identifier and date.

Create a Table with Partitions

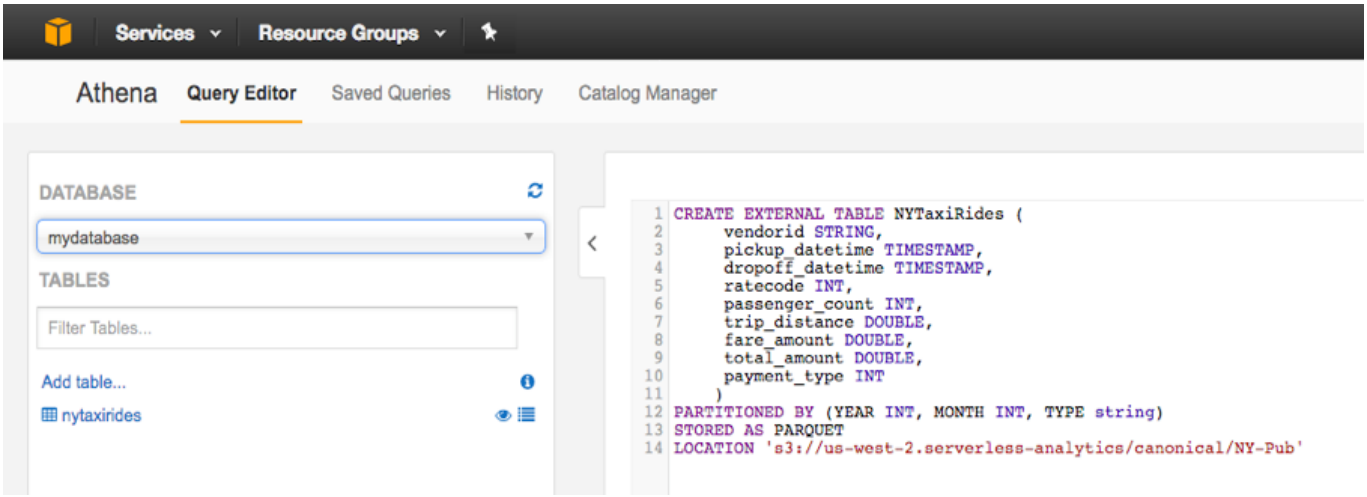
1. Ensure that current AWS region is **US West (Oregon)** region

2. Ensure <username> is selected from the DATABASE list and then choose **New Query**.

3. In the query pane, copy the following statement to create a the NYTaxiRides table, and then choose **Run Query**:

```
CREATE EXTERNAL TABLE taxi_parquet (
    vendorid STRING,
    pickup_datetime TIMESTAMP,
    dropoff_datetime TIMESTAMP,
    ratecode INT,
    passenger_count INT,
    trip_distance DOUBLE,
    fare_amount DOUBLE,
    total_amount DOUBLE,
    payment_type INT
    )
  PARTITIONED BY (YEAR INT, MONTH INT, TYPE string)
  STORED AS PARQUET
  LOCATION 's3://us-west-2.serverless-analytics/canonical/NY-Pub'
```

4.Ensure the table you just created appears on the Catalog dashboard for the selected database.



> **Note:** Running the following sample query on the TaxiData_parquet table you just created will not return any result as no metadata about the partition is added to the Amazon Athena table catalog.
>
> ```
> SELECT * FROM taxi_parquet limit 10
> ```

Adding partition metadata to Amazon Athena

Now that you have created the table you need to add the partition metadata to the Amazon Athena Catalog.

1. Choose **New Query**, copy the following statement into the query pane, and then choose **Run Query** to add partition metadata.

```
MSCK REPAIR TABLE taxi_parquet
```

The returned result will contain information for the partitions that are added to NYTaxiRides for each taxi type (yellow, green, fhv) for every month for the year from 2009 to 2016

> **Note:** The MSCK REPAIR TABLE automatically adds partition data based on the New York taxi ride data to in the Amazon S3 bucket is because the data is already converted to Apache Parquet format partitioned by year, month and type, where type is the taxi type (yellow, green or fhv). If the data layout does not confirm with the requirements of MSCK REPAIR TABLE the alternate approach is to add each partition manually using ALTER TABLE ADD PARTITION. You can also automate adding partitions by using the JDBC driver.

Querying partitioned data set

Now that you have added the partition metadata to the Athena data catalog you can now run your query.

1. Choose **New Query**, copy the following statement into the query pane, and then choose **Run Query** to get the total number of taxi rides

```
SELECT count(1) as TotalCount from taxi_parquet
```

Results for the above query look like the following:

Results

	TotatlCount
1	2870781820

2. Choose **New Query**, copy the following statement into the query pane, and then choose **Run Query** to get the total number of taxi rides by year

```
SELECT YEAR, count(1) as TotalCount from taxi_parquet GROUP BY YEAR
```

Results for the above query look like the following:

Results

	YEAR	TotalCount
1	2015	375327352
2	2010	338002306
3	2012	357088648
4	2011	353794416
5	2014	346065723
6	2016	411140768
7		168
8	2009	341792110
9	2013	347570329

3. Choose **New Query**, copy the following statement into the query pane, and then choose **Run Query** to get the top 12 months by total number of rides across all the years

```
SELECT YEAR, MONTH, COUNT(1) as TotalCount
FROM taxi_parquet
GROUP BY (1), (2)
ORDER BY (3) DESC LIMIT 12
```

Results for the above query look like the following:

Results

	YEAR	MONTH	TotalCount
1	2016	5	36063253
2	2016	3	35722884
3	2016	4	35654963
4	2016	10	35614908
5	2016	12	35562724
6	2015	10	34934049
7	2016	6	34545922
8	2016	2	33772790
9	2016	11	33700663
10	2015	12	33354426
11	2016	9	33088841
12	2016	7	33014293

4. Choose **New Query**, copy the following statement into the query pane, and then choose **Run Query** to get the monthly ride counts per taxi time for the year 2016.

```
SELECT MONTH, TYPE, COUNT(1) as TotalCount
FROM taxi_parquet
WHERE YEAR = 2016
GROUP BY (1), (2)
ORDER BY (1), (2)
```

Results for the above query look like the following:

Results

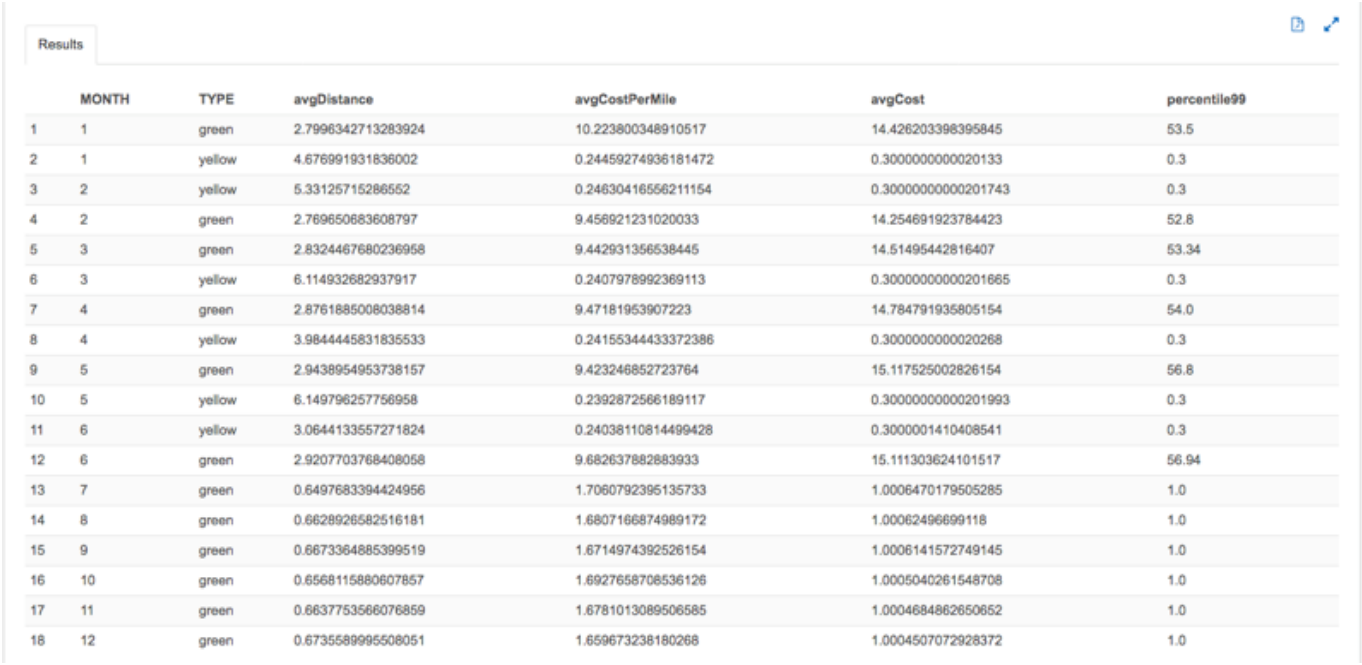
	MONTH	TYPE	TotalCount
1	1	fhv	8756726
2	1	green	1445285
3	1	yellow	21813716
4	2	fhv	9497970
5	2	green	1510722
6	2	yellow	22764098
7	3	fhv	9724587
8	3	green	1576393

> **Note:** Now the execution time is ~ 4 seconds, as the amount of data scanned by the query is restricted thus improving performance. This is because the data set is partitioned and it in optimal format – Apache Parquet, an open source columnar format.

5. Choose **New Query**, copy the following statement anywhere into the query pane, and then choose **Run Query**.

```
SELECT MONTH,
TYPE,
    avg(trip_distance) as  avgDistance,
    avg(total_amount/trip_distance) as avgCostPerMile,
    avg(total_amount) as avgCost,
    approx_percentile(total_amount, .99) percentile99
FROM taxi_parquet
WHERE YEAR = 2016 AND (TYPE = 'yellow' OR TYPE = 'green') AND
trip_distance > 0 AND total_amount > 0
GROUP BY MONTH, TYPE
ORDER BY MONTH
```

Results for the above query look like the following:



Results

	MONTH	TYPE	avgDistance	avgCostPerMile	avgCost	percentile99
1	1	green	2.7996342713283924	10.223800348910517	14.426203398395845	53.5
2	1	yellow	4.676991931836002	0.24459274936181472	0.3000000000020133	0.3
3	2	yellow	5.33125715286552	0.24630416556211154	0.30000000000201743	0.3
4	2	green	2.769650683608797	9.456921231020033	14.254691923784423	52.8
5	3	green	2.8324467680236958	9.442931356538445	14.51495442816407	53.34
6	3	yellow	6.114932682937917	0.2407978992369113	0.30000000000201665	0.3
7	4	green	2.8761885008038814	9.47181953907223	14.784791935805154	54.0
8	4	yellow	3.9844445831835533	0.24155344433372386	0.3000000000020268	0.3
9	5	green	2.9438954953738157	9.423246852723764	15.117525002826154	56.8
10	5	yellow	6.149796257756958	0.2392872566189117	0.30000000000201993	0.3
11	6	yellow	3.0644133557271824	0.24038110814499428	0.3000001410408541	0.3
12	6	green	2.9207703768408058	9.682637882883933	15.111303624101517	56.94
13	7	green	0.6497683394424956	1.7060792395135733	1.0006470179505285	1.0
14	8	green	0.6628926582516181	1.6807166874989172	1.00062496699118	1.0
15	9	green	0.6673364885399519	1.6714974392526154	1.0006141572749145	1.0
16	10	green	0.6568115880607857	1.6927658708536126	1.0005040261548708	1.0
17	11	green	0.6637753566076859	1.6781013089506585	1.0004684862650652	1.0
18	12	green	0.6735589995508051	1.659673238180268	1.0004507072928372	1.0

Summary

This lab has highlighted the following key points:

1. Creating Databases and Tables using Athena cn be done using either the console wizard or SQL commands.
2. Partitioning the data improved query significantly as the following table shows:

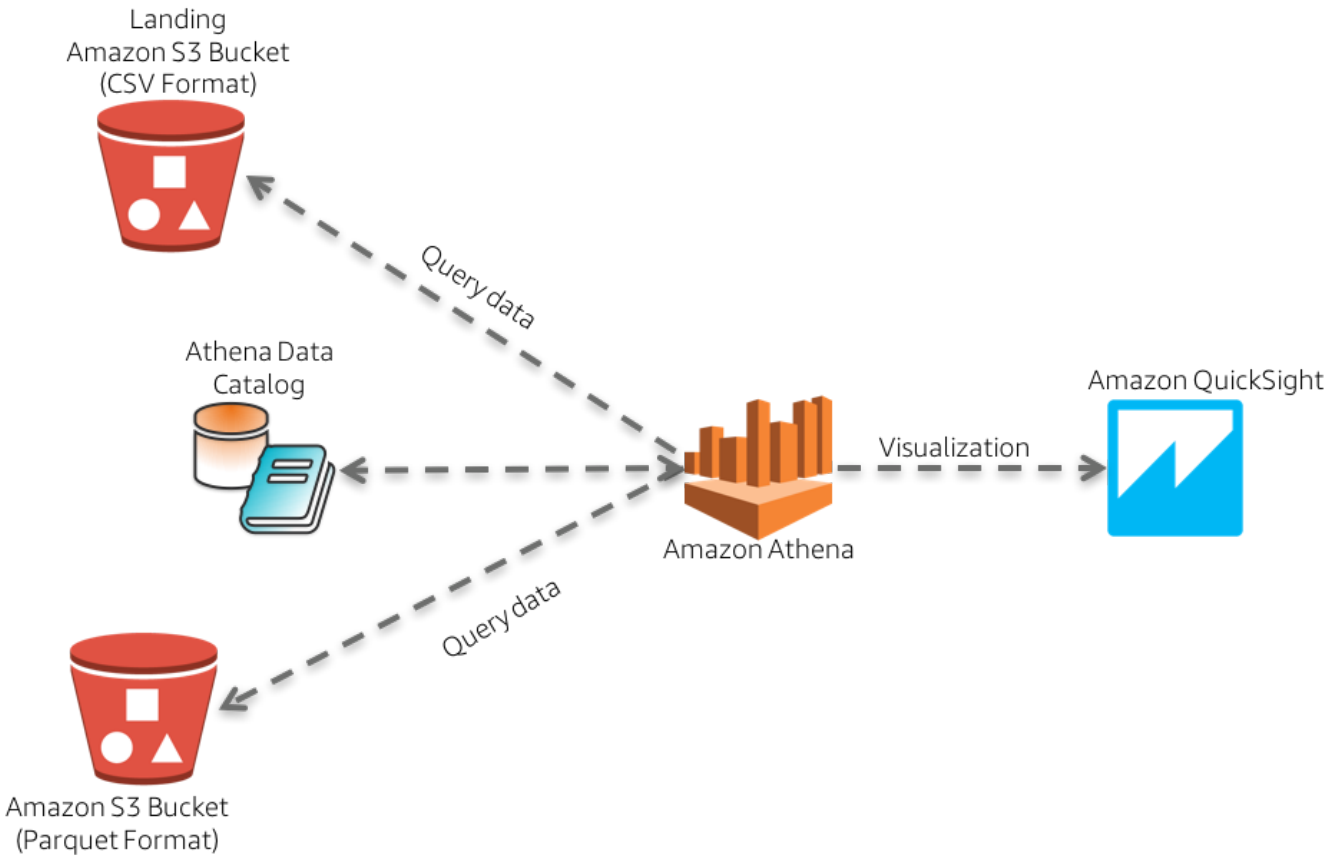
	Query	Run Time	Data Scanned	Results
CSV	SELECT count(*) as count FROM TaxiData_csv	~21.74 seconds	~207.54GB	1,310,911,060
Parquet	SELECT count(*) as count FROM TaxiData_parquet	~4.79 seconds	0KB	2,870,781,820
CSV	SELECT * FROM TaxiData_csv limit 1000	~1.013 seconds	~479.7MB	-
Parquet	SELECT * FROM TaxiData_parquet limit 1000	~1.1 seconds	~5.6MB	-

License

Lab 2: Visualization using Amazon QuickSight

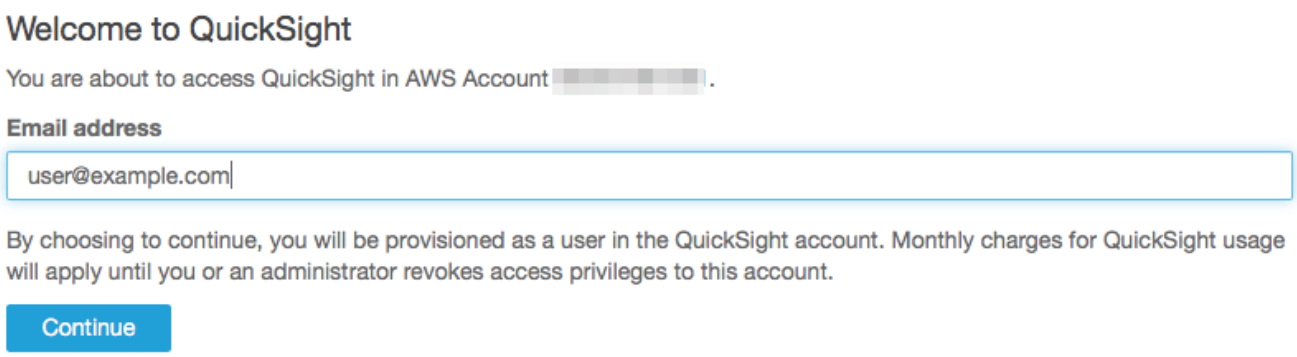
- Signing in to Amazon QuickSight Standard Edition for the first time-->
- Configuring Amazon QuickSight to use Amazon Athena as data source
- Visualizing the data using Amazon QuickSight
    - Add year based filter to visualize the dataset for the year 2016
    - Add the month based filter for the month of January
    - Visualize the data by hour of day for the month of January 2016
    - Visualize the data for the month of January 2016 for all taxi types(yellow, green, fhv)
- Summary

Architectural Diagram



Signing in to Amazon Quicksight Standard Edition for the first time

1. Open the AWS ManagementConsole for QuickSight.
2. If this is the first time you are accessing QuickSight, you will see a welcome landing page for QuickSight.



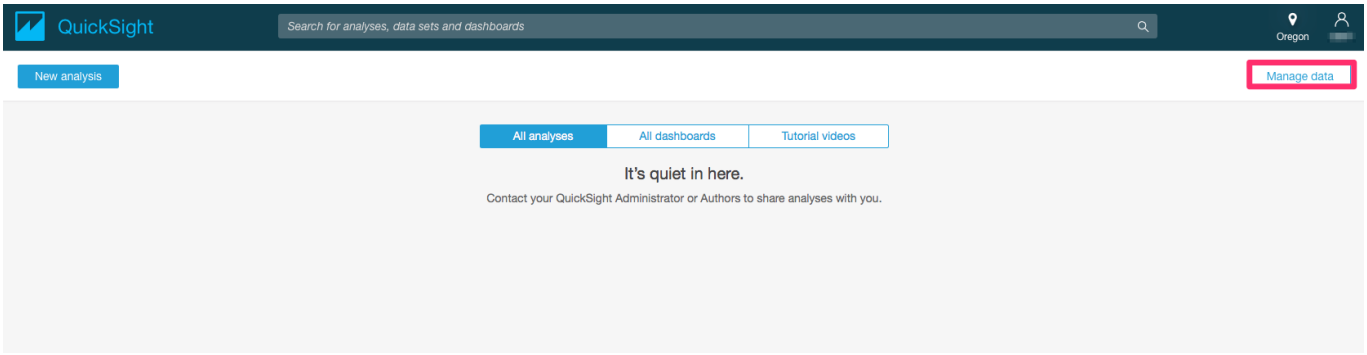
3. Enter your **Email address** and click **Continue**.

> **Note:** Chrome browser might timeout at this step. If that's the case, try this step in Firefox/Microsoft Edge/Safari.

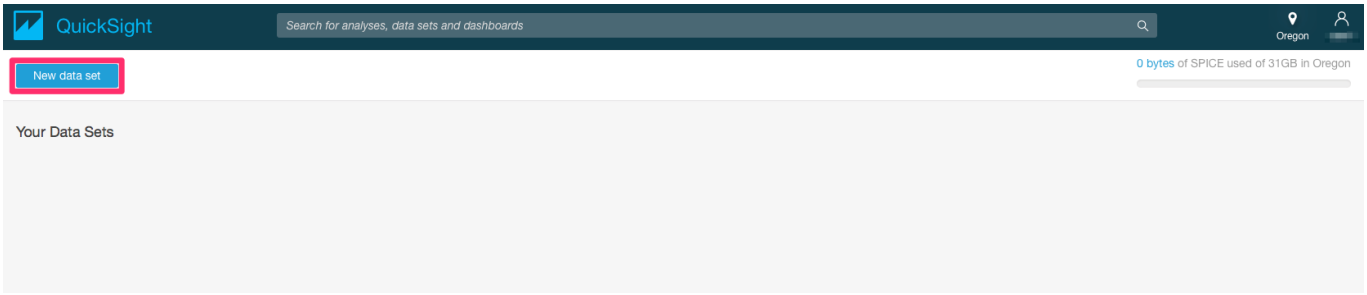
Configuring Amazon QuickSight to use Amazon Athena as data source

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

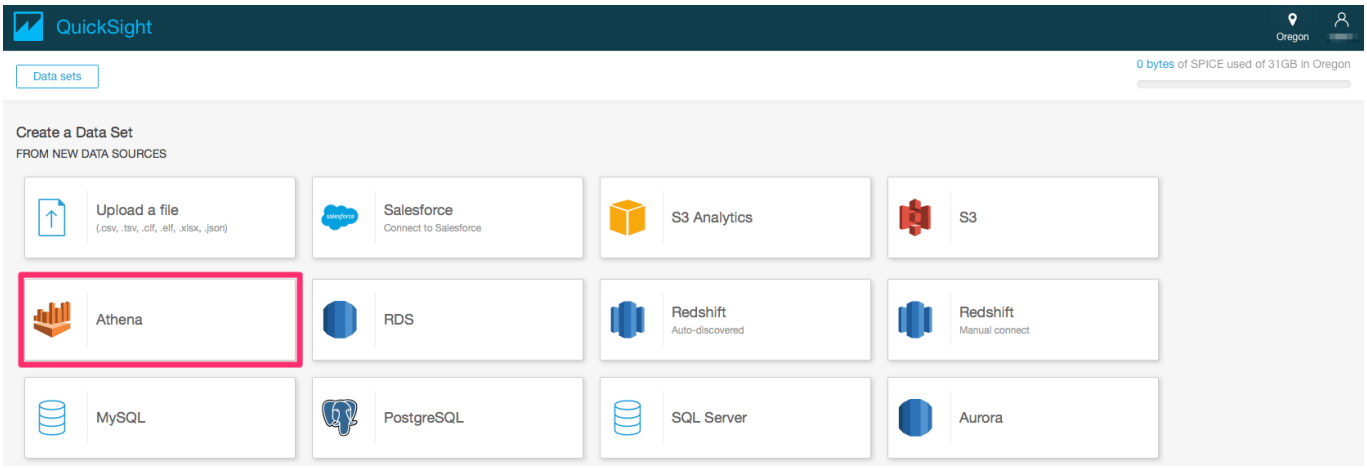
1. Click on the region icon on the top-right corner of the page, and select **US West (Oregon)**.
2. Click on **Manage data** on the top-right corner of the webpage to review existing data sets.



3. Click on **New data set** on the top-left corner of the webpage and review the options.

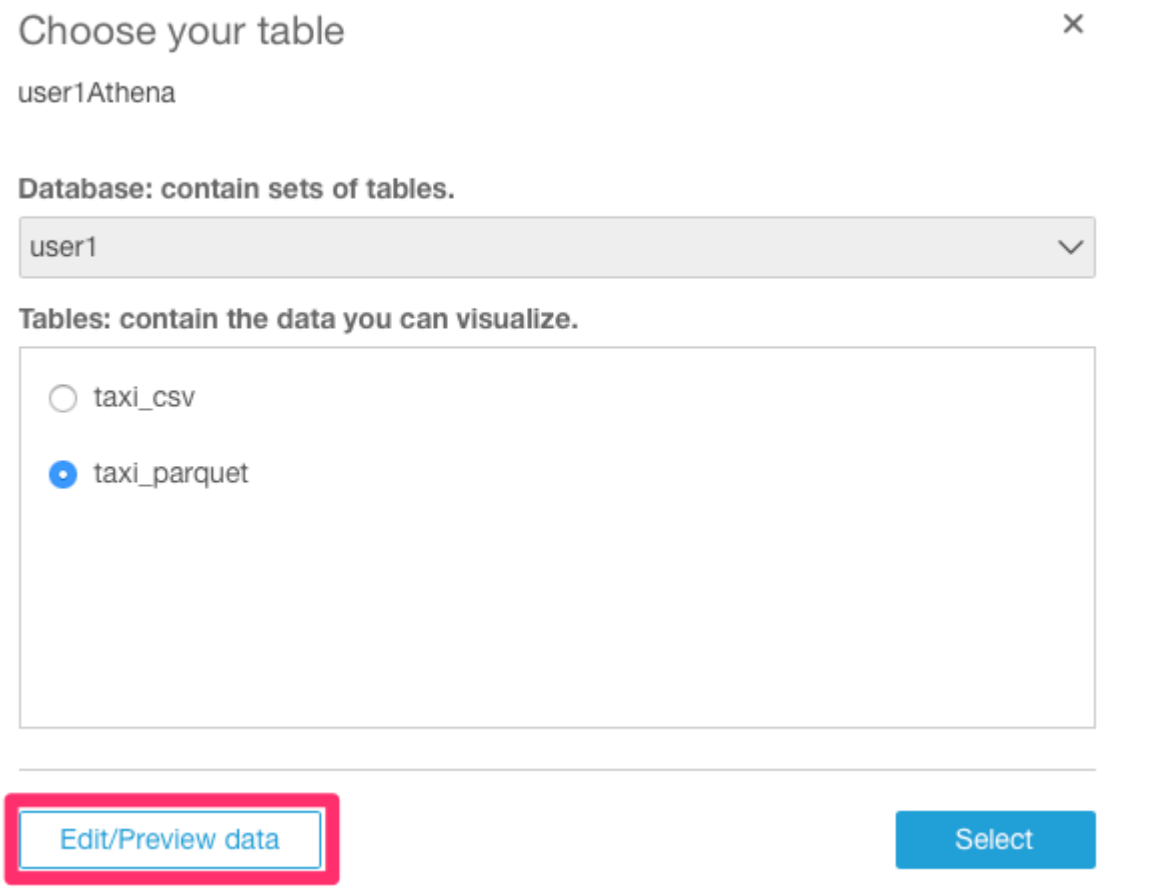


4. Select **Athena** as a Data source.



5. Enter the **Data source name** (e.g. *user1Athena*).
6. Click **Create data source**.
7. Select the `<username>` database.

8. Choose the `taxi_parquet` table.
9. Choose **Edit/Preview** data.



> **Note:** This is a crucial step. Please ensure you choose **Edit/Preview** data.

10. As an example of augmenting the existing data, we can add a new field, derived from existing data. Under **Fields** on the left column, choose **New field**

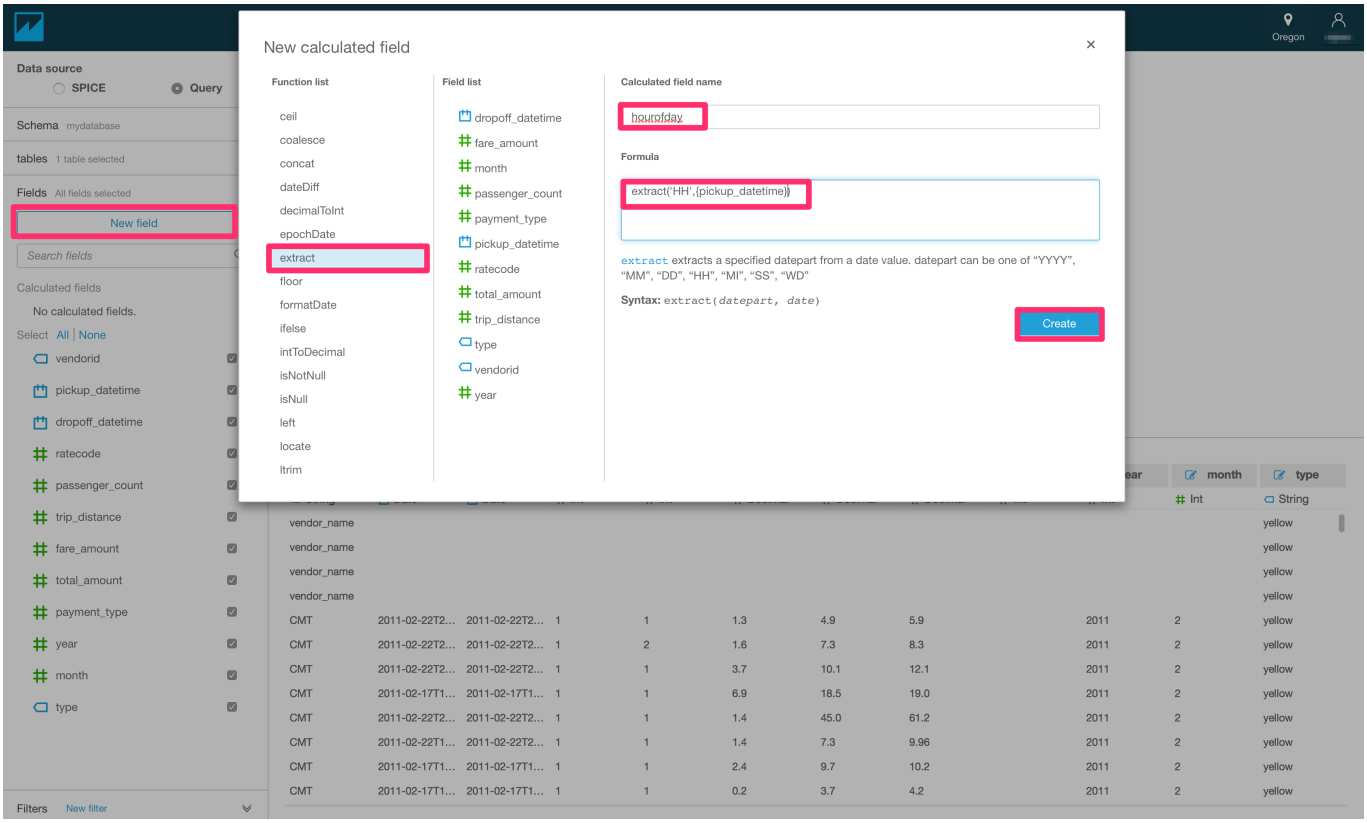
    i. Select the **extract** operation from Function list.

ii. Select `pickup_datetime` from the **Field list**.

iii. For **Calculated field name**, type `hourofday`.

iv. Type 'HH' so the Formula is `extract('HH',{pickup_datetime})`

v. Choose **Create** to add a field which is calculated from an existing field. In this case, the **hourofday** field is calculated from the **pickup_datetime filed** based on the specified formula.



11. Choose **Save and Visualize** on top of the page.

Visualizing the data using Amazon QuickSight

Now that you have configured the data source and created a new filed to represent the hour of the day, in this section you will filter the data by year followed by month to visualize the taxi data for the entire month of January 2016 based on the **pickup_datetime** field.

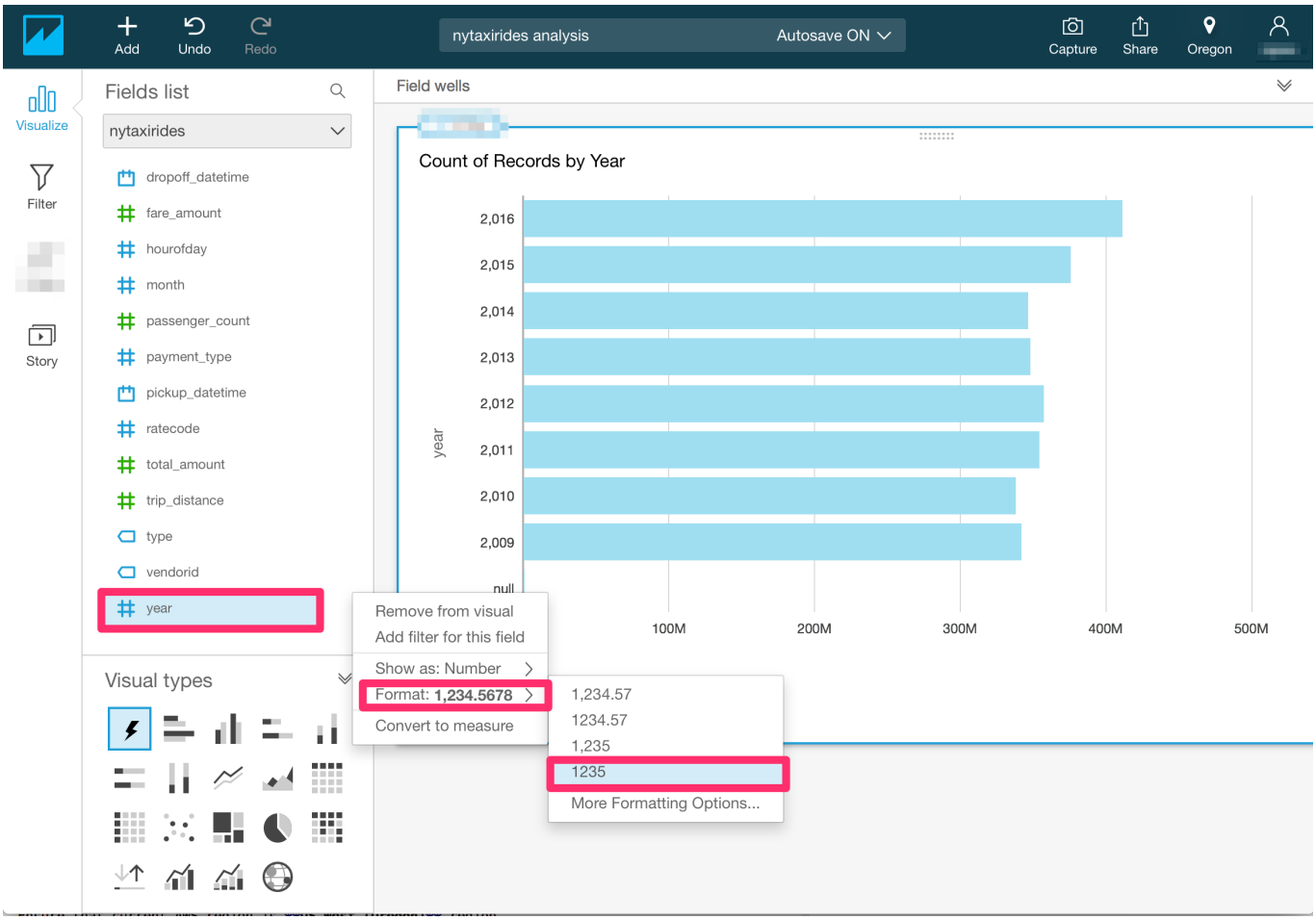
Add year based filter to visualize the dataset for the year 2016

1. Ensure that current AWS region is **US West (Oregon)** region.
2. Under the **Fields List**, select the **year** field to show the distribution of fares per year.
3. To reformat the **year** without comma

   i. Select the dropdown arrow for the **year** field.

   ii. Select **Format 1,234.5678** from the dropdown menu.

   iii. Select **1235**.



4. To add a filter on the **year** filed:

   i. Select the dropdown for **year** field from the **Fields list**.

   ii. Select **Add filter to the field** from the dropdown menu.

5. To filter the data only for the year 2016

   i. Choose the new filter that you just created by clicking on **#** next to filter name **year** under the **Edit filter** menu.

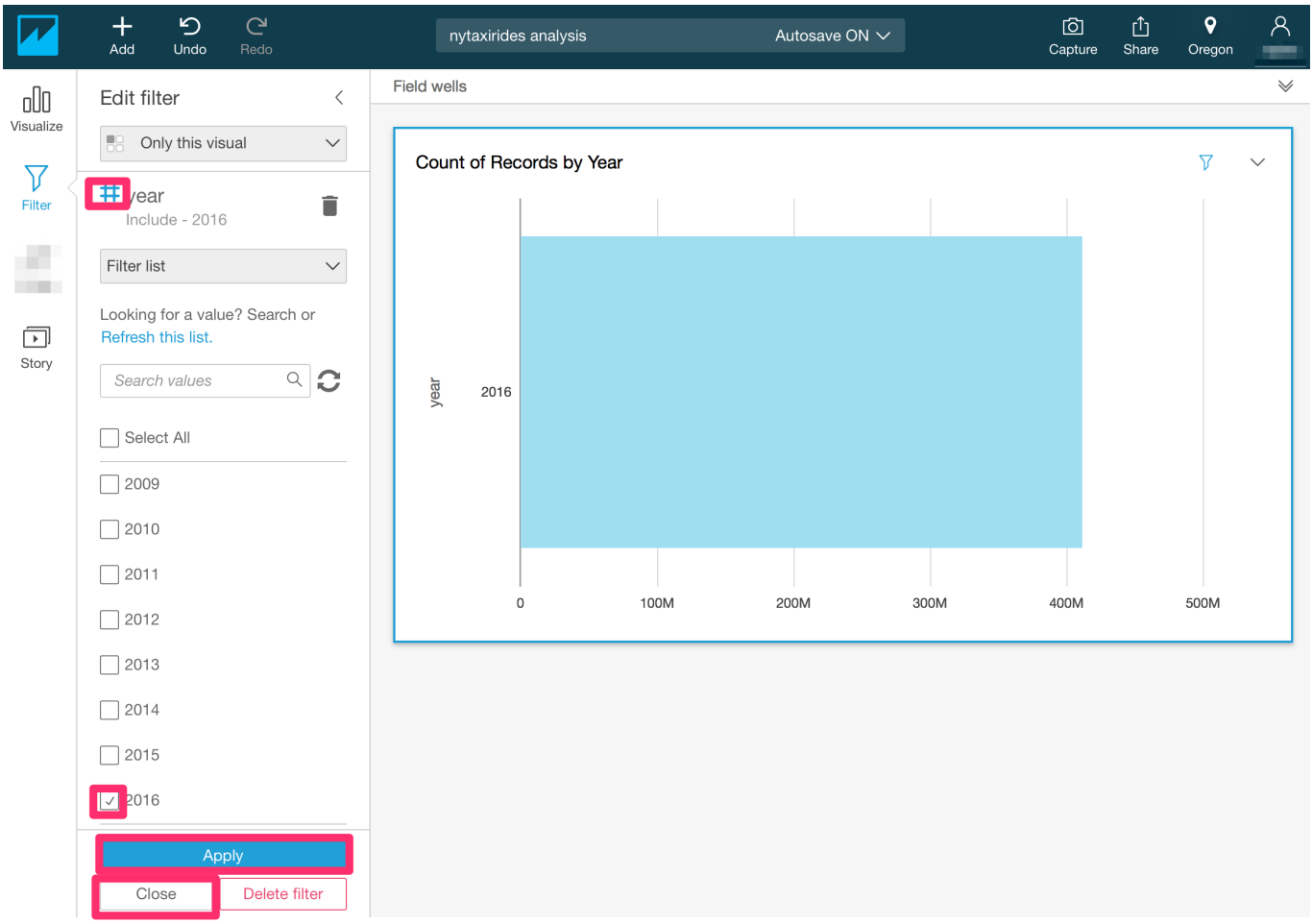
   ii. Select **Filter list** for the two dropdowns under the filter name.

   iii. Deselect **Select All**.

   iv. Select only **2016**.

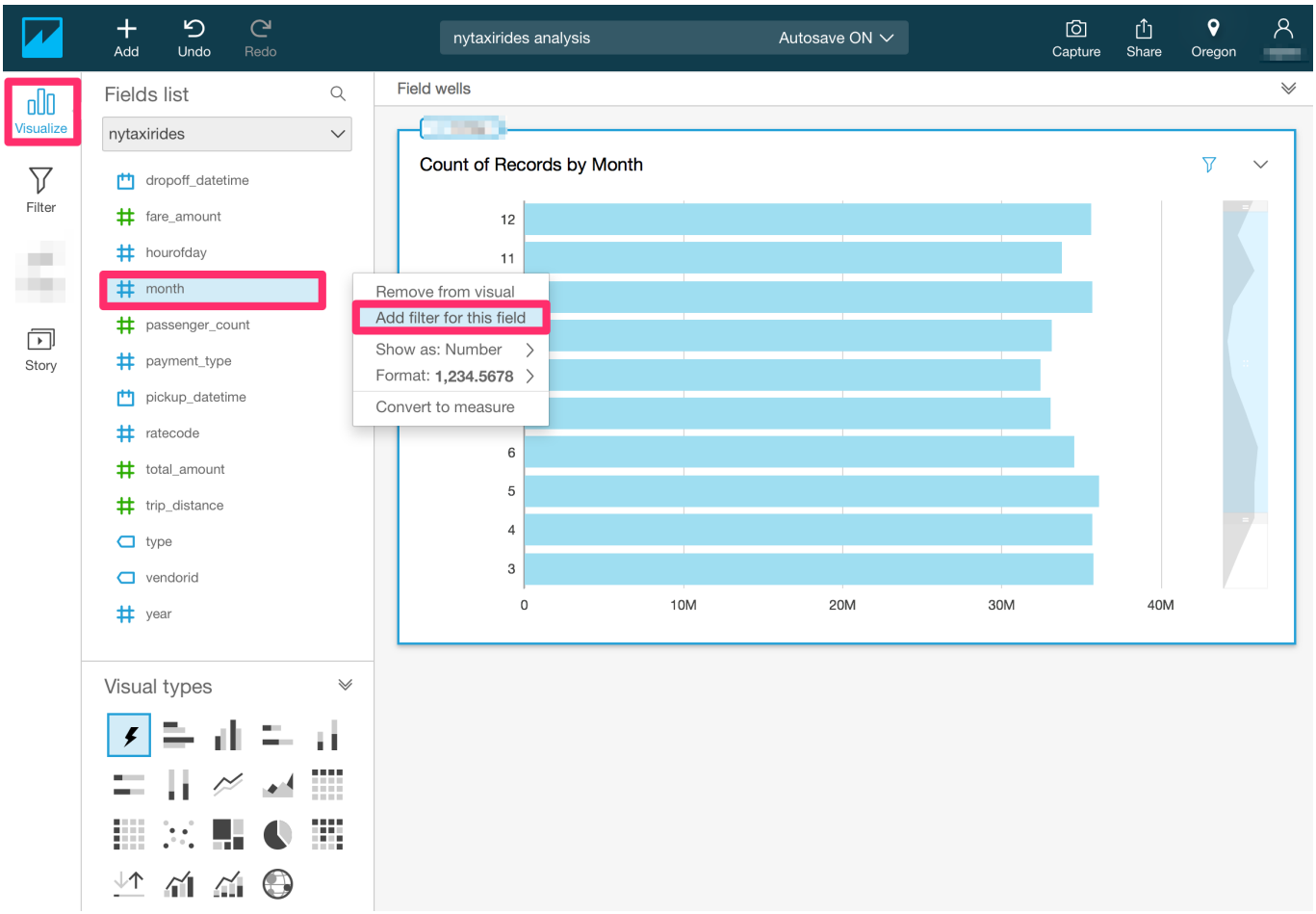
   v. Click **Apply**.

   vi. Click **Close**.



Add the month based filter for the month of January

1. Ensure that current AWS region is **US West(Oregon)** region.
2. Select **Visualize** from the navigation menu in the left-hand corner.
3. Under the **Fields list**, deselect **year** by clicking on **year** field name.
4. Select **month** by clicking on the **month** field name from the **Fields list**.



5. To filter the data set for the month of January (Month 1)

   i. Select the dropdown arrow for **month** field under the **Fields List**.

   ii. Select **Add filter to the field**.

6. To filter the data for month of January 2016 (Month 1),

   i. Choose the new filter that you just created by clicking on **#** next to filter name **month** under the **Edit Filter** menu.

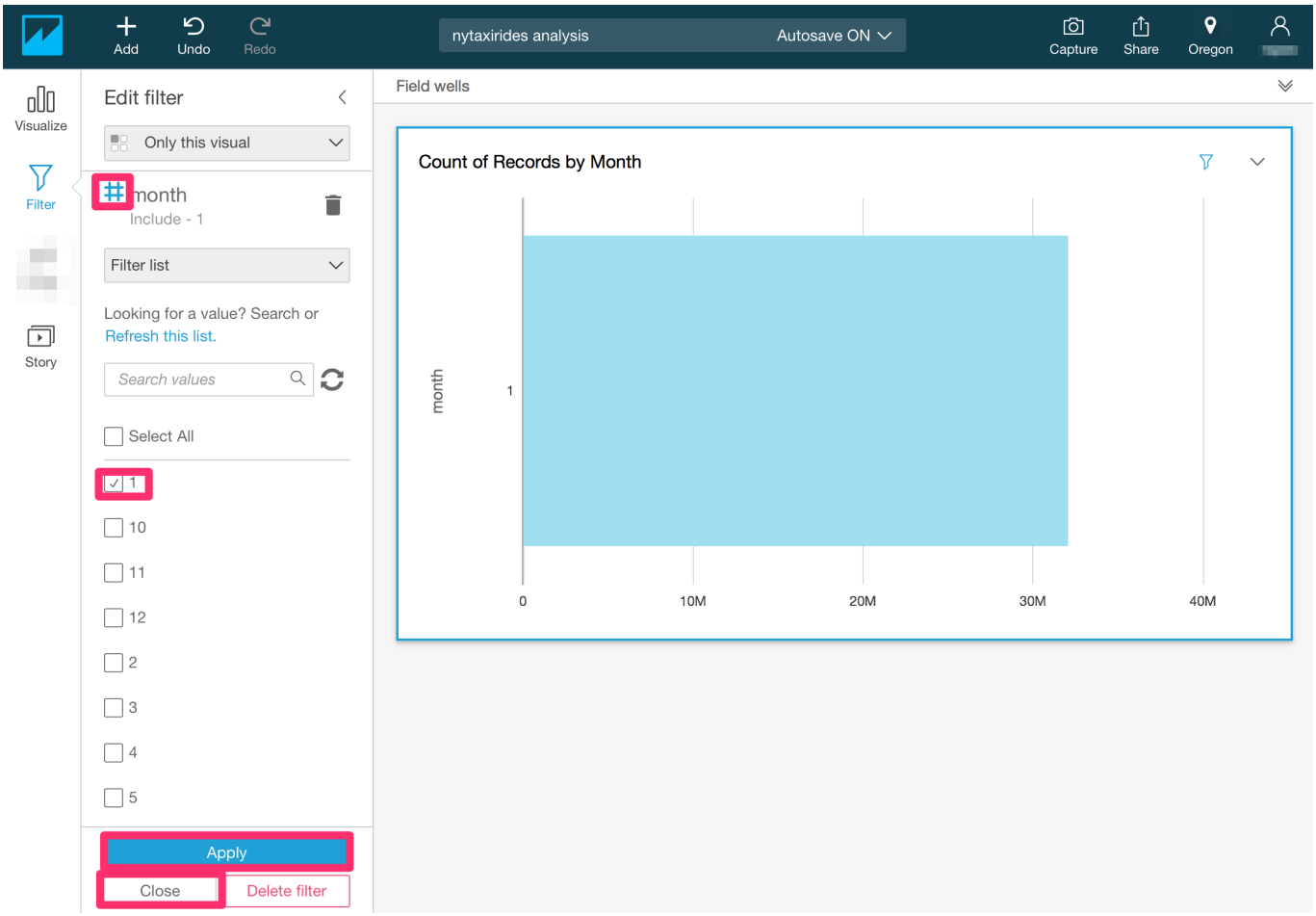
   ii. Select **Filter list** for the two dropdown lists, under the filter name.

   iii. Deselect **ALL**.

   iv. Select only **1**.

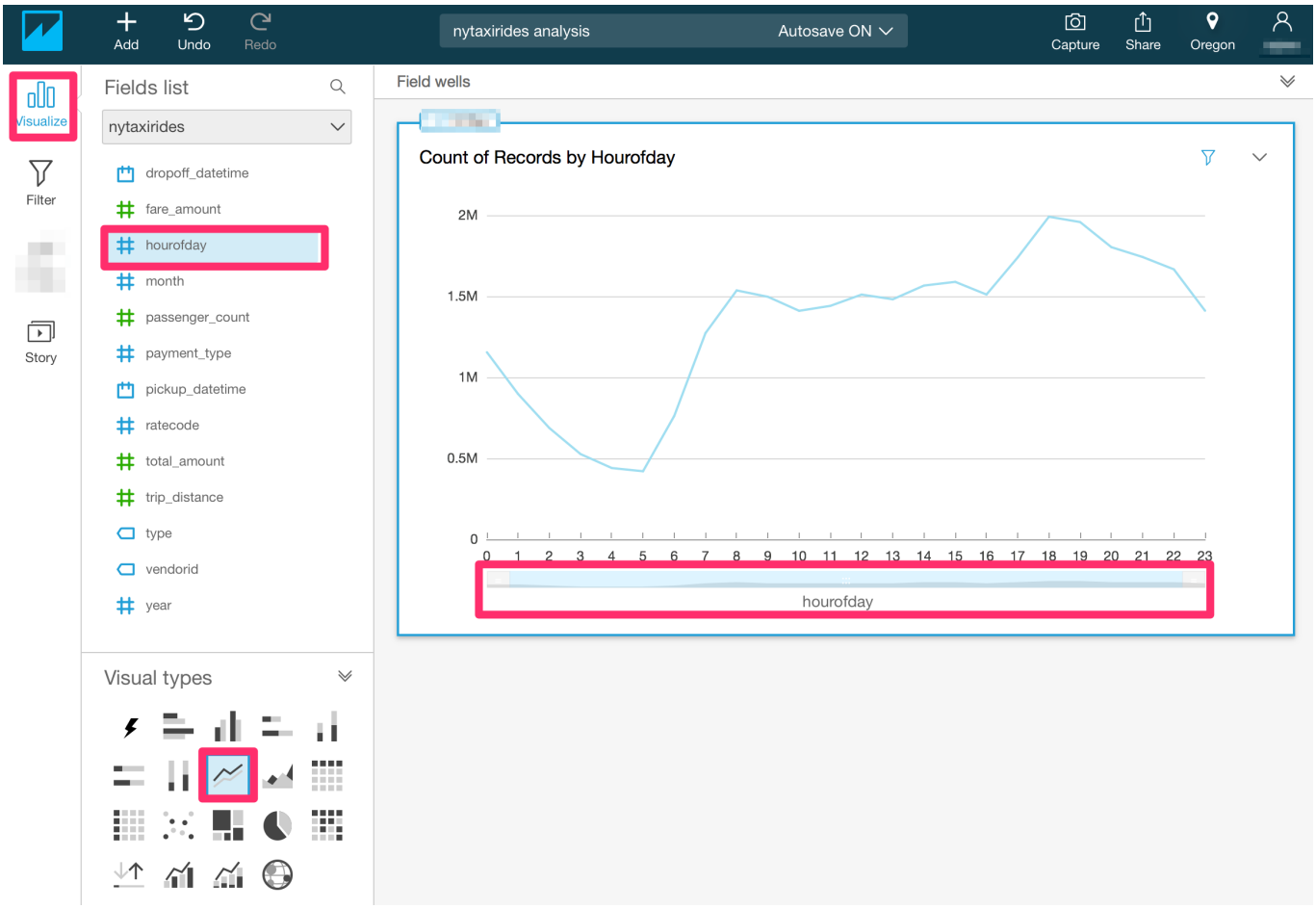
   v. Click **Apply**

   vi. Click **Close**.



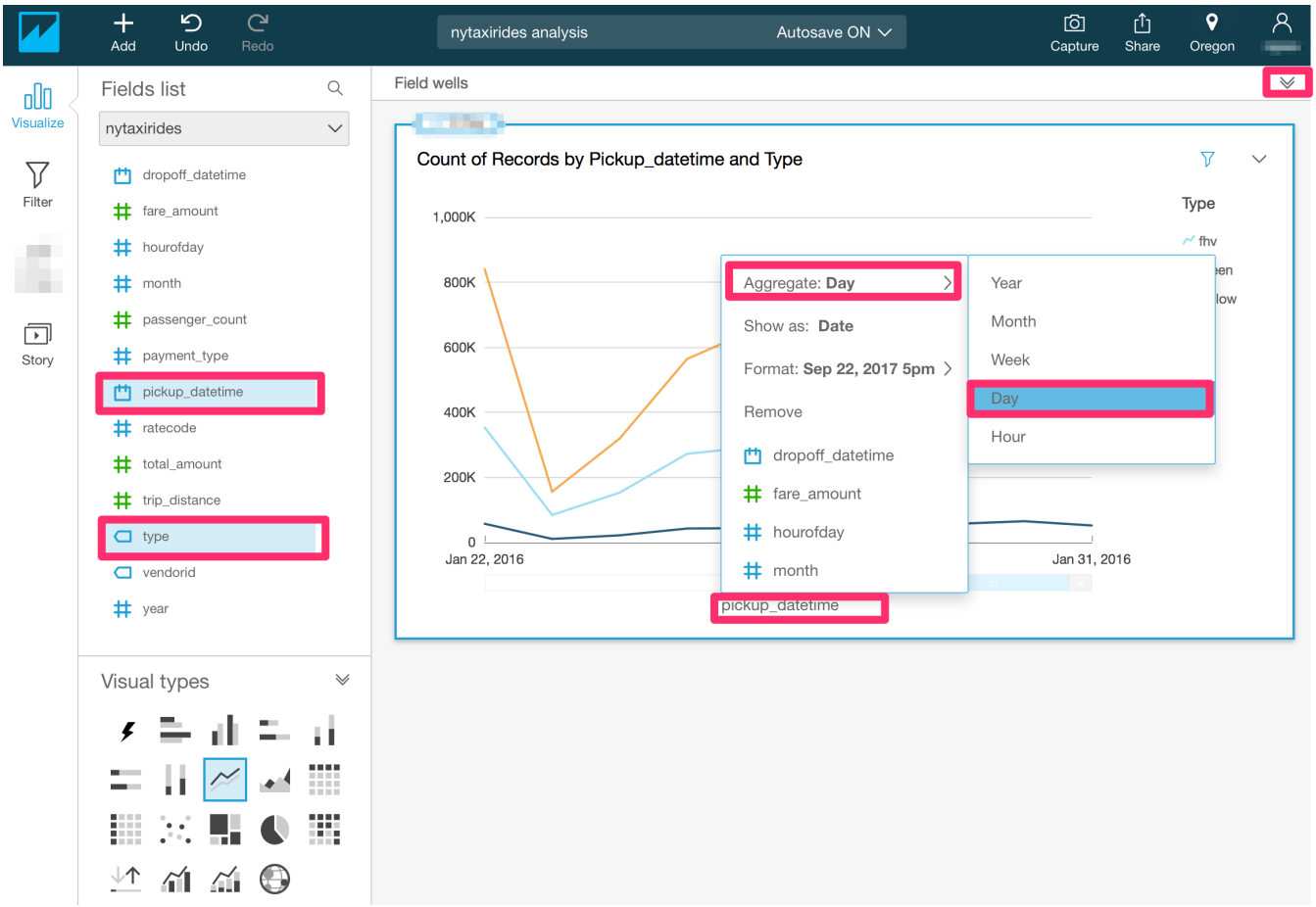
Visualize the data by hour of day for the month of January 2016

1. Select **Visualize** from the navigation menu in the left-hand corner.
2. Under the **Fields list**, deselect **month** by clicking on **month** field name.
3. Select **hourofday** by clicking on the **hourofday** field name from the **Fields list**.
4. Change the visual type to a line chart by selecting the line chart icon highlighted in the screenshot below under **Visual types**.
5. Using the slider on x-axis, select the entire range [0,23] for **hourofday** field.

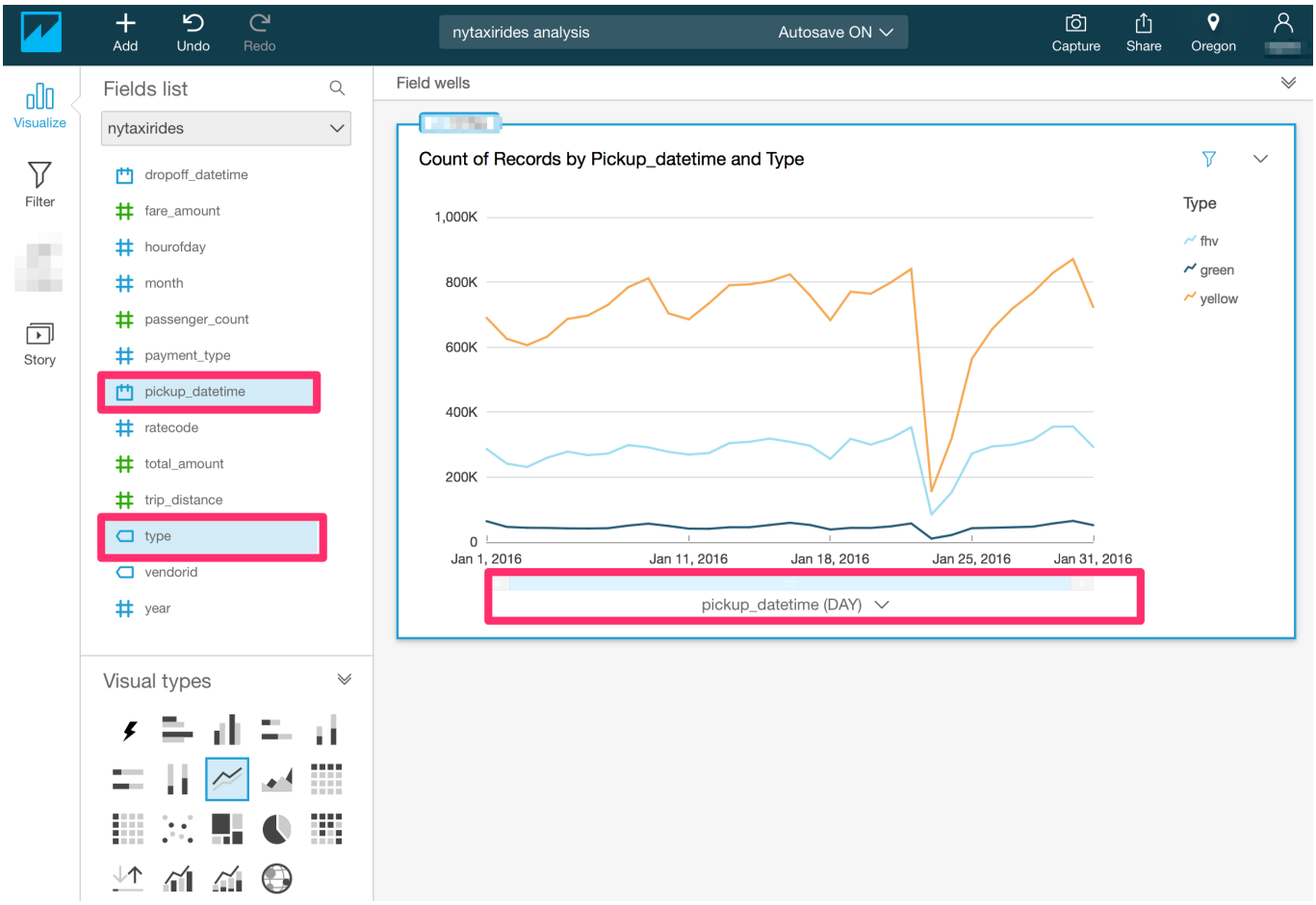


Visualize the data for the month of January 2016 for all taxi types(yellow, green, fhv)

1. Click on the double drop-down arrow underneath your username at the top-right corner of the page to reveal **X-axis**, **Value** and **Color** under **Field wells**.
2. Under the **Fields list**, deselect **hourofday** by clicking on **hourofday** field name.
3. Select **pickup_datetime** for x-axis by clicking on the **pickup_datetime** field name from **Fields list**.
4. Select **type** for Color by clicking on the **type** field name from **Fields list.**



5. Click on the field name **pickup_datetime** in x-axis to reveal a sub-menu.
6. Select **Aggregate:Day** to aggregate by day.
7. Using the slider on x-axis, select the entire month of January 2016 for **pickup_datetime** field.



> Note: The interesting outlier in the above graph is that on Jan23rd, 2016, you see the dip in the number of taxis across all types. Doing a quick google search for that date, gets us this weather article from NBC New York

Summary

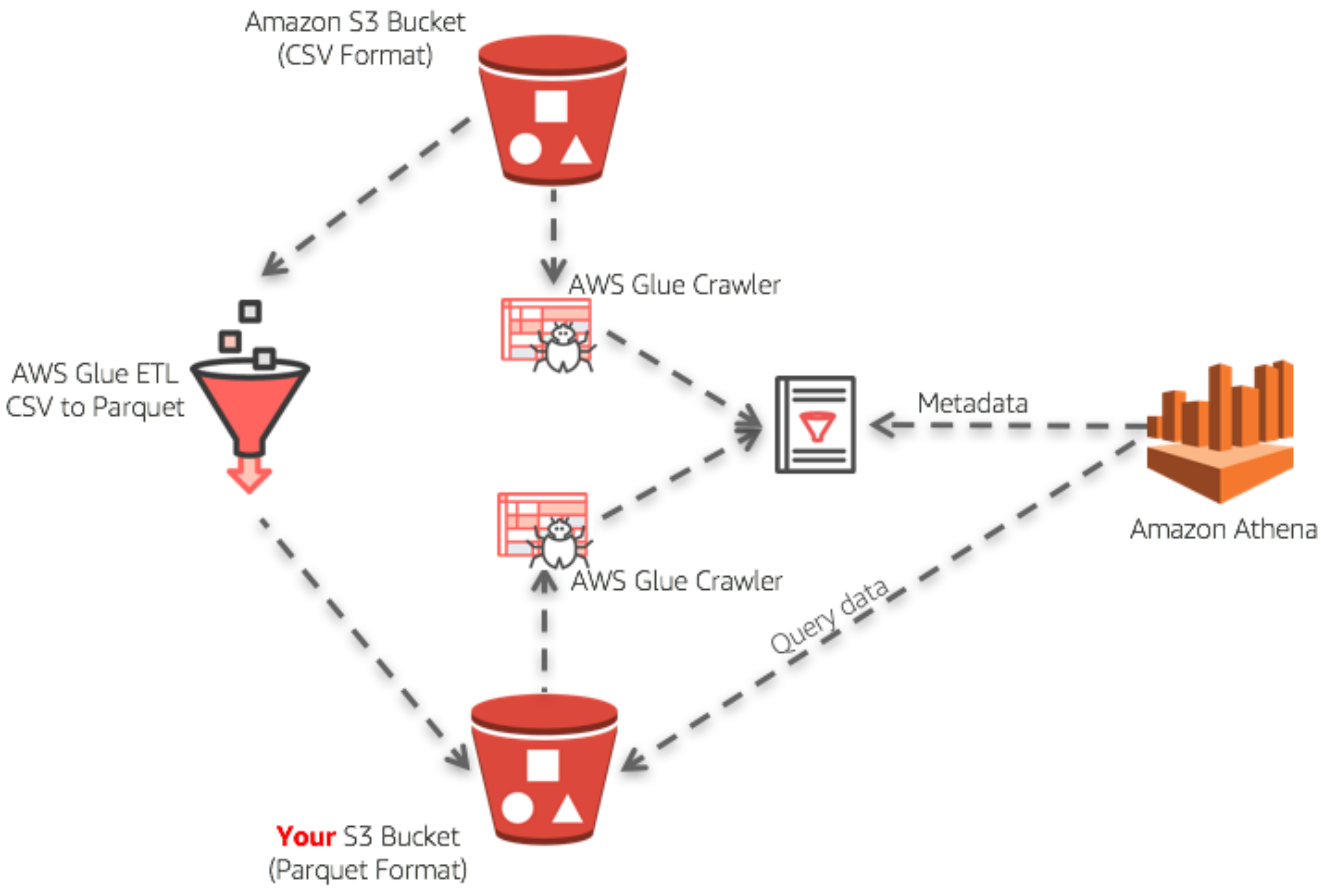
Using Amazon QuickSight, you were able to see patterns across a time-series data by building visualizations, performing ad-hoc analysis, and quickly generating insights.

License

Lab 3: 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 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.

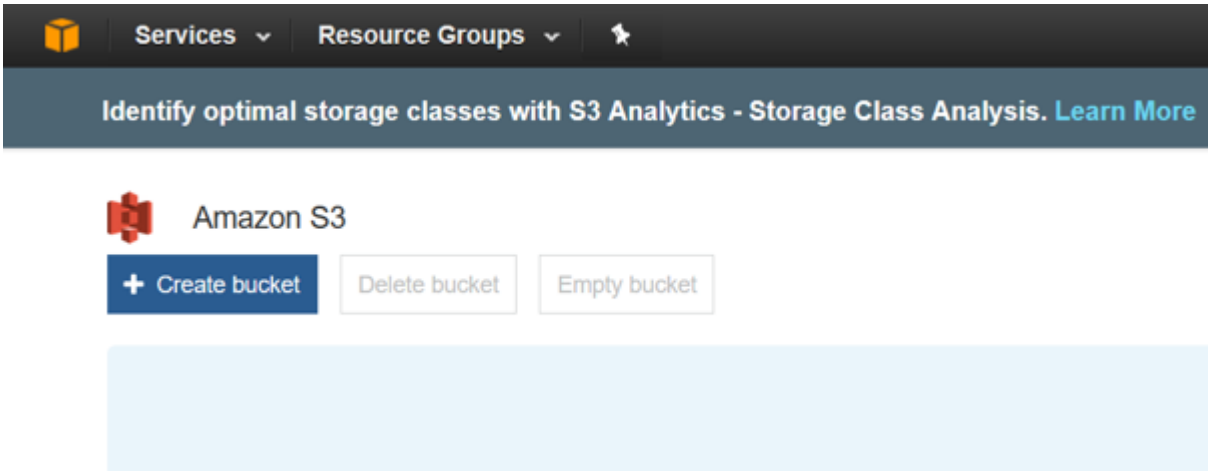
5. Click on **Next: Review**.
6. Enter Role name as:

```
<username>-glue-etl-role
```

7. 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**.

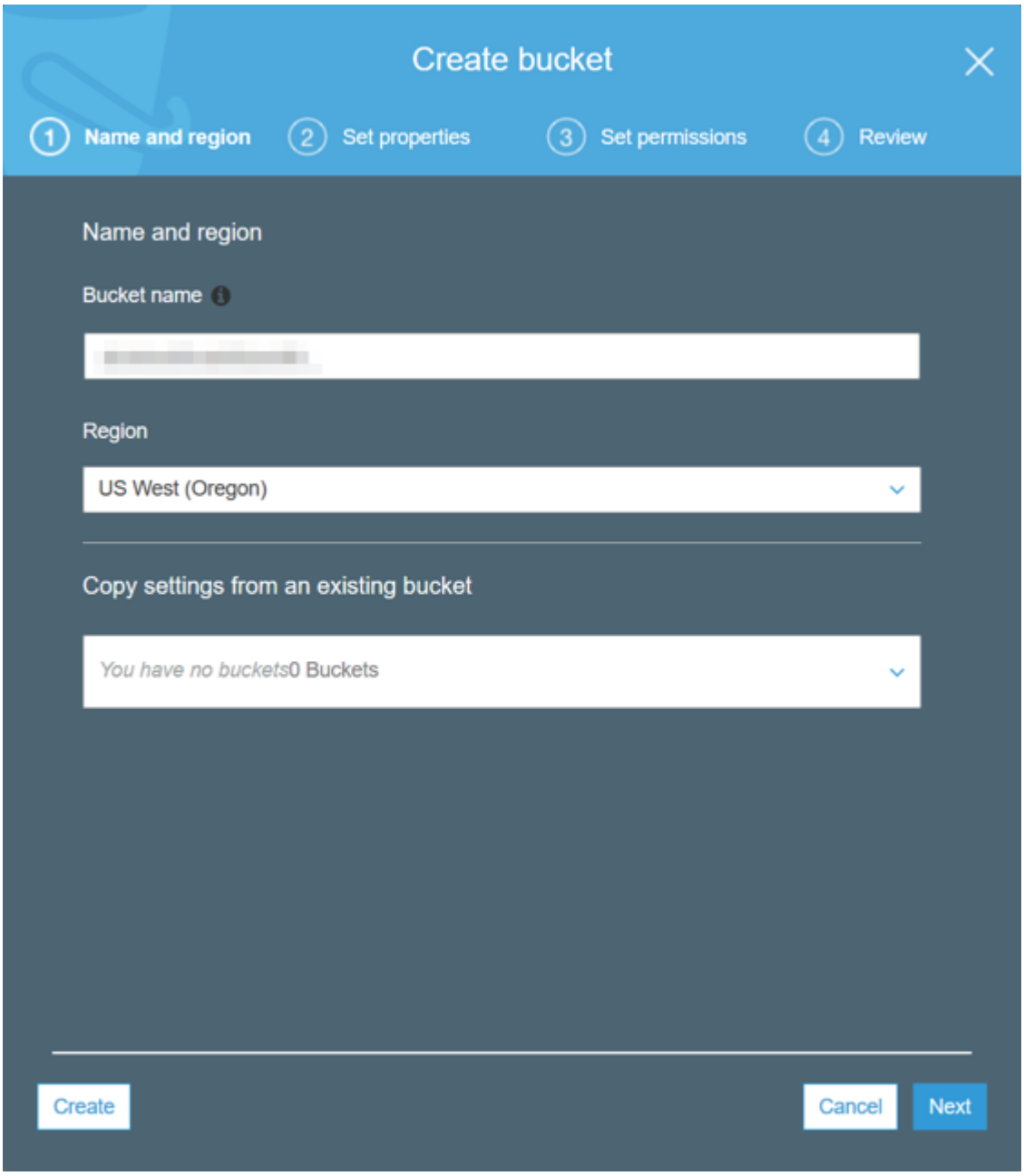


1. In the **Create Bucket** pop-up page, input a unique **Bucket name**. So it's advised to choose a large bucket name, with many random characters and numbers (no spaces). It will be easier to name your bucket

```
<username>-glue-scripts-us-west-2
```

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

i.Select the region as **Oregon**.
ii. Click **Next** to navigate to next tab.
iii. In the **Set properties** tab, leave all options as default.
iv. In the **Set permissions** tag, leave all options as default.
v. In the **Review** tab, click on **Create Bucket**.



2. Now, in this newly created bucket, create two folders **tmp** and **yellow**. We will use these buckets as part of the Lab later on.

Discover the Data

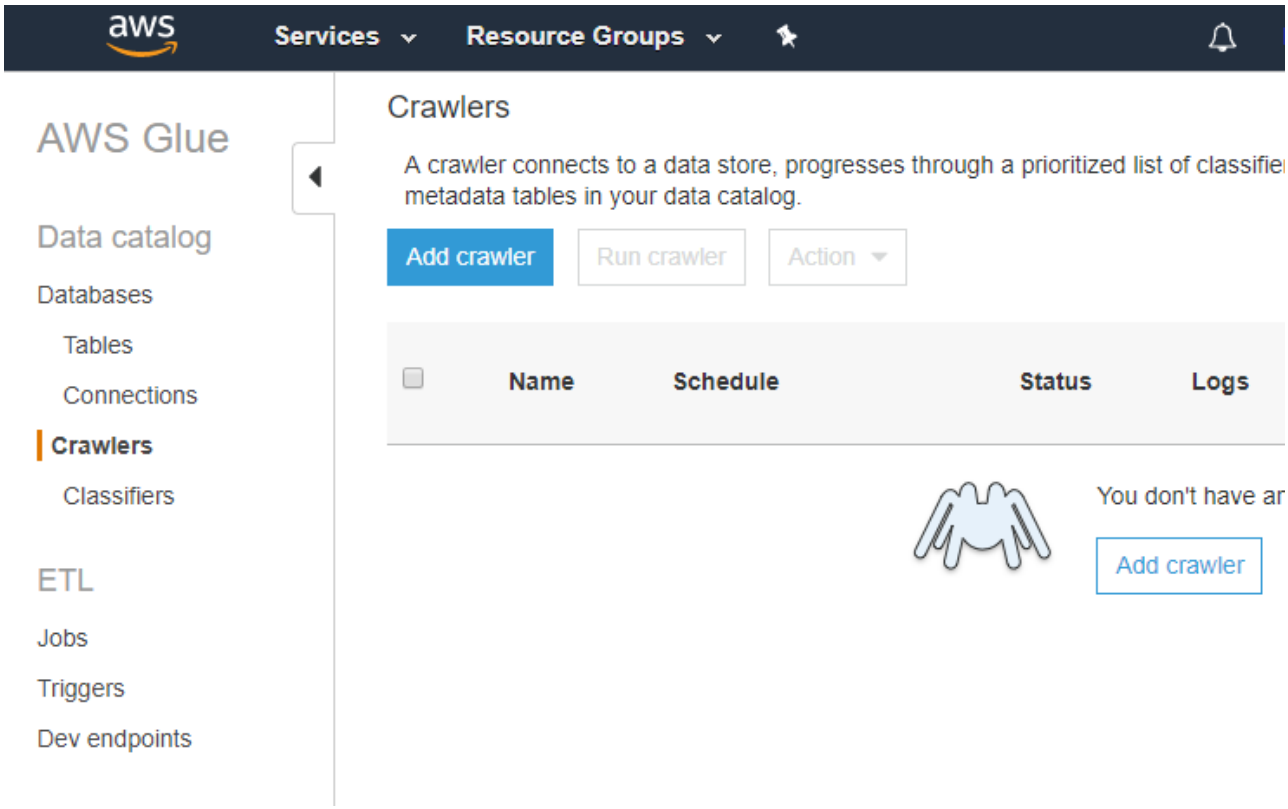
During this lab, we will focus on one month of the New York City Taxi Records dataset, however you could easily do this for the entire eight years of data. As you crawl this unknown dataset, you discover that the data is in different formats, depending on the type of taxi. You then convert the data to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

> **Note:** 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. We will use the database that was created during Lab 1. Remember, 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. For more information on Database nameing conventions, see the Athena Documentation.

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



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

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

iii. Under Data Store, choose S3 and ensure the radio button for **Specified path in another account** is checked.

iv. 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**.

```
<username>-crawler-role
```

> **Note:** The IAM Role Name should resemble `AWSGlueServiceRole-<username>-crawler-role`.

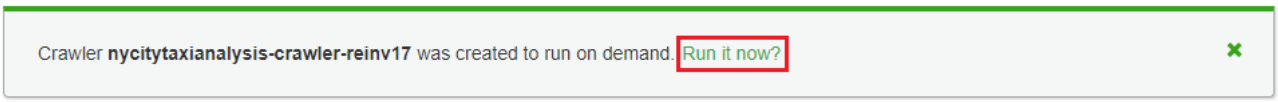
vii. For Create a schedule for this crawler, choose Frequency as **Run on Demand** and click on **Next**.

viii. Configure the crawler output database and prefix:

a. For **Database**, select the database created during Lab 1, `<username>`.

b. For **Prefix added to tables (optional)**, type `<username>_` 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.

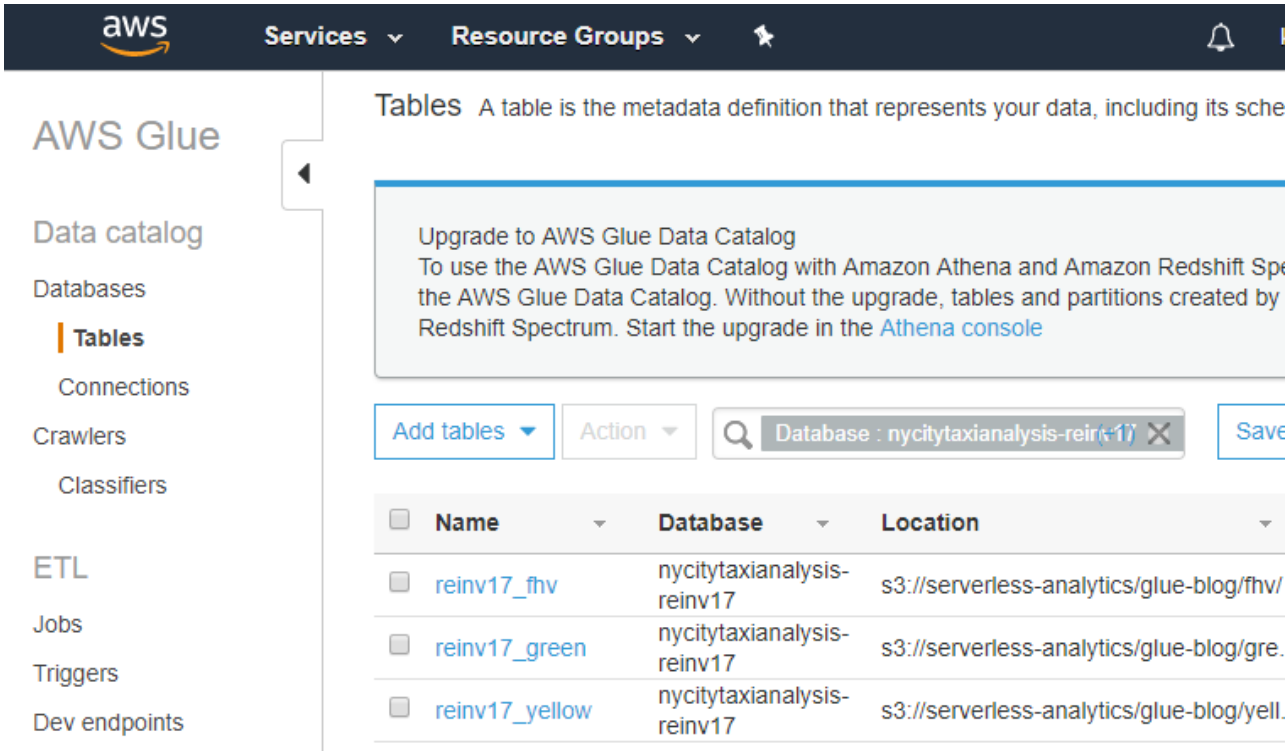


> **Note:** The crawler should take approximately 30 seconds to run, wait until the *"**Crawler `<username>-crawler` completed and made the following changes: 3 tables created, 0 tables updated. See the tables created in database `<username>`.**" message has appeared before proceeding to the next step.

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 `<username>`.



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 `<username>_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	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 `<username>-yellow-etl`. 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, e.g. `<username>-glue-etl-role`

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

   xi. For **ETL language**, choose the radio button for **Python**.

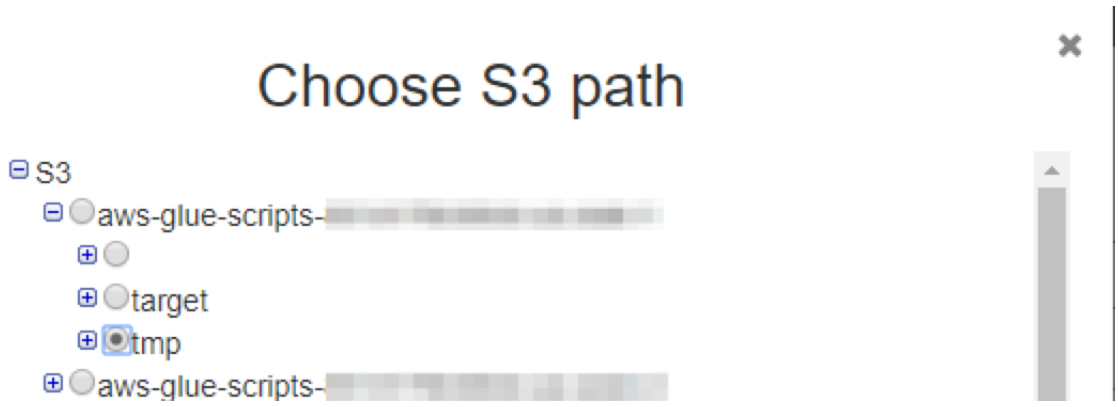
   xii. For **Script file name**, enter `<username>-yellow-etl`.

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

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

   xiiiv. For Temporary directory, choose the `tmp` folder created at the beginning of this Lab. **Choose the S3 bucket via the Folder icon** and click **Next**.

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



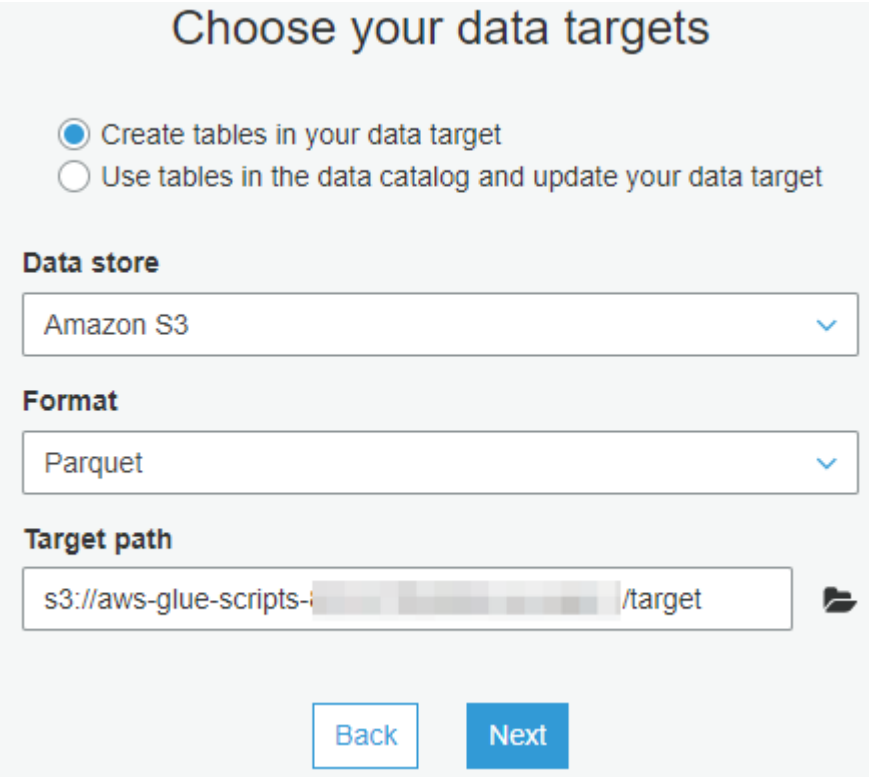
4. Click **Next**.
5. Under Choose your data sources, select `<username>_yellow` table as the data source and click on **Next**.

6. 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 `yellow` folder previously created. **This S3 Bucket/Folder will contain the transformed Parquet data**.



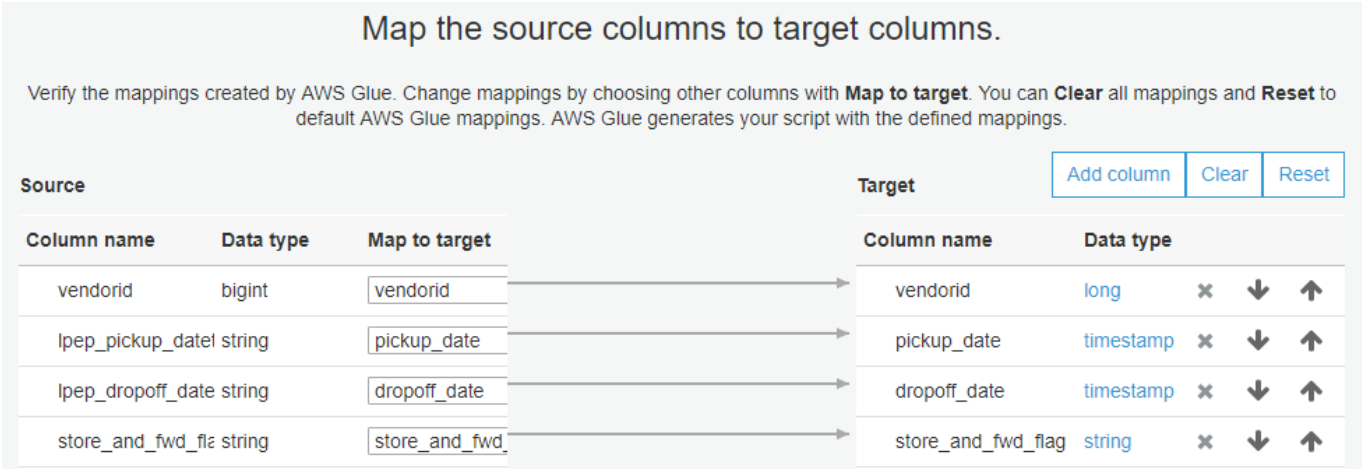
7. Under Map the source columns to target columns page,

   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. Click **Next**.

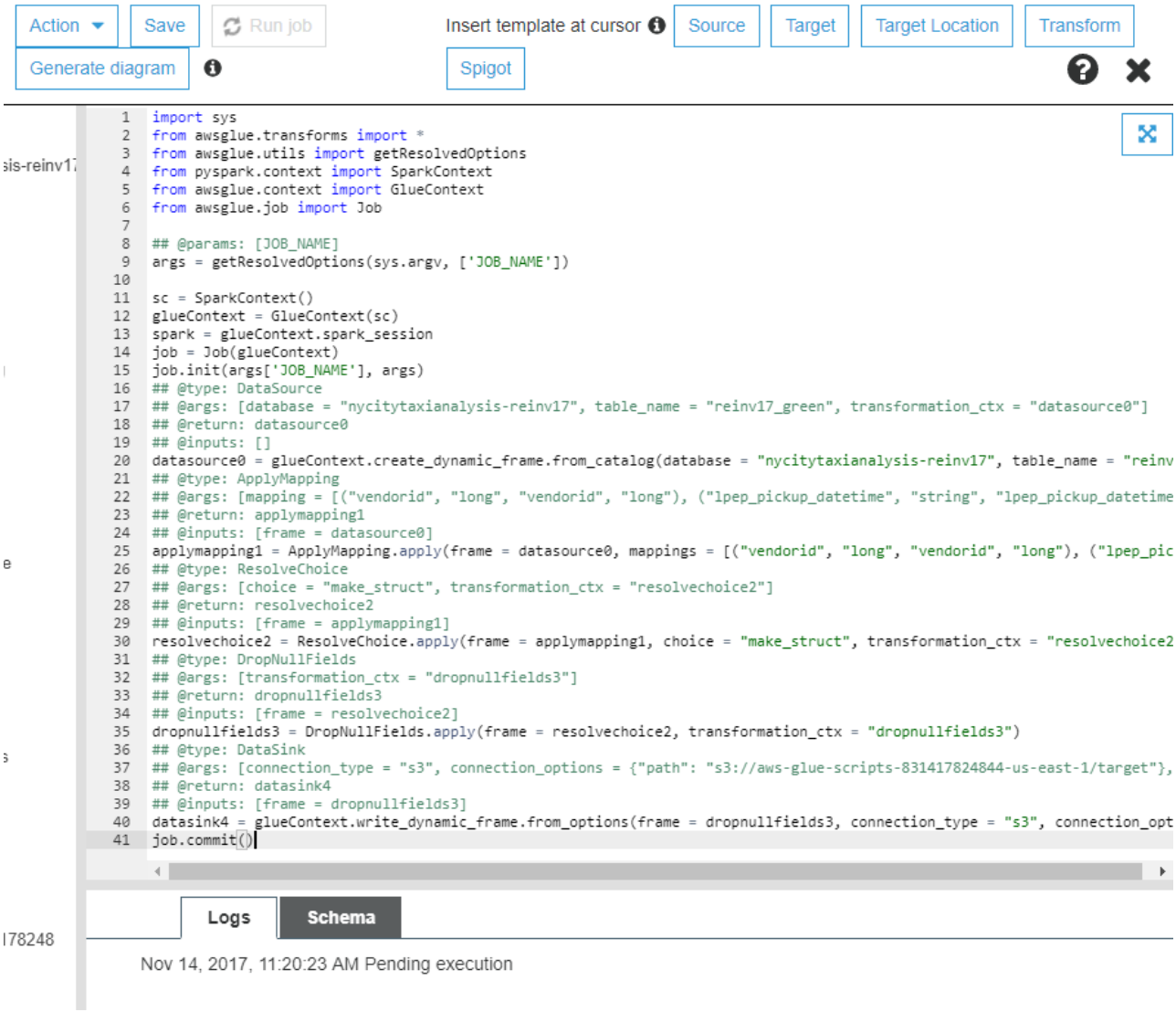
   iv. Verify the information and click **Save job and edit script**.



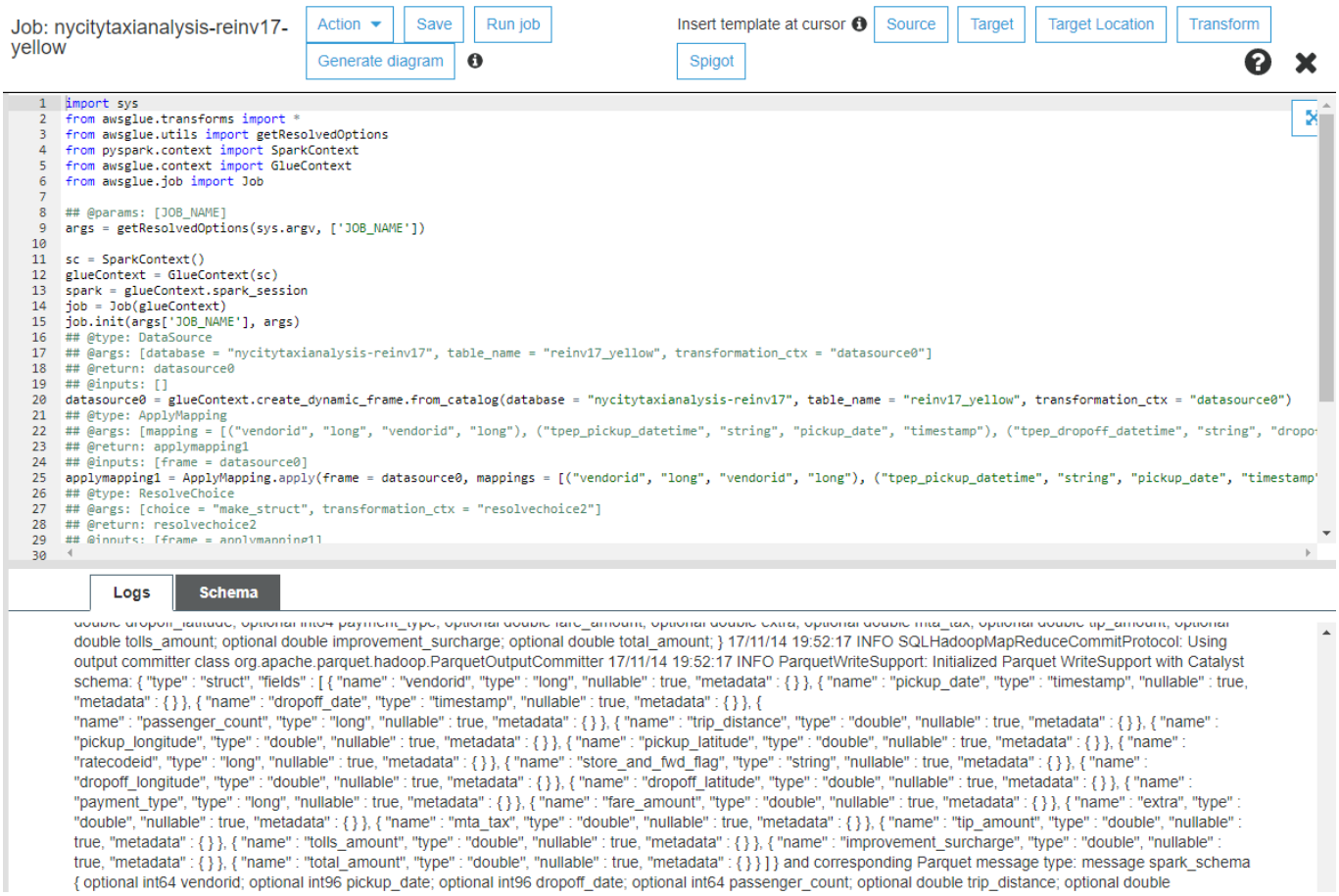
8. On the auto-generated script page, click on **Save** and **Run Job**.

8. In the parameters pop-up, for Job bookmark, ensure its **Disable** and click on **Run Job**.

9. This job will run for roughly 3 minutes.



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



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 `<username>` which was created during Lab 1.

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

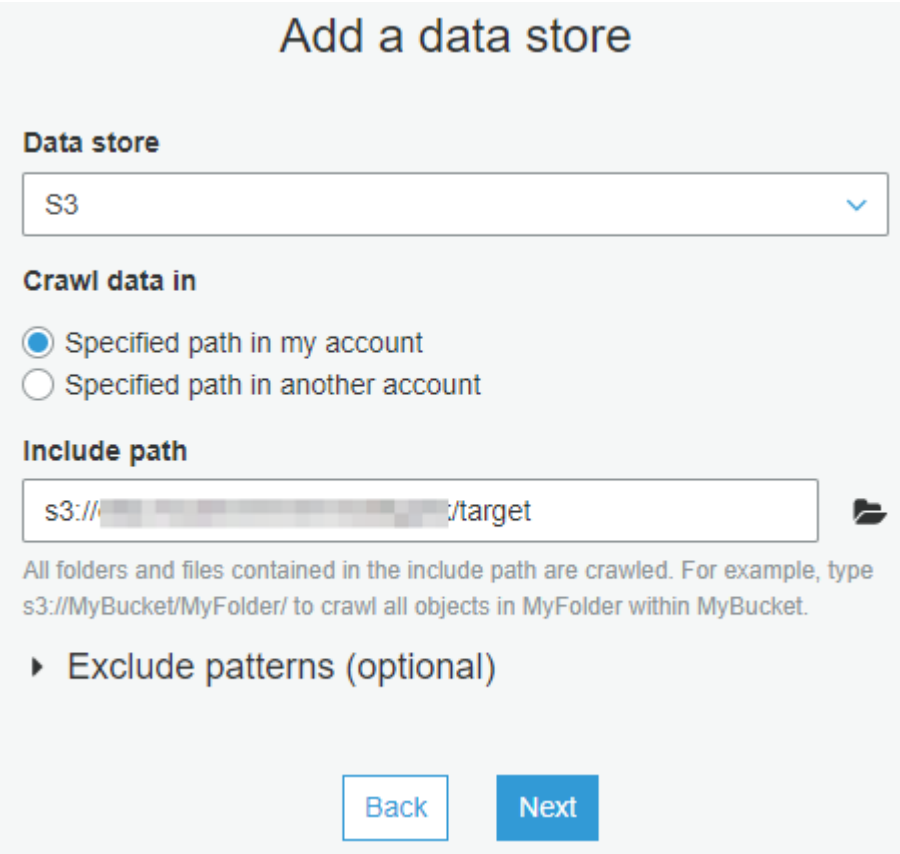
4. You will now be re-directed to the AWS Glue console to set up a crawler. The crawler connects to your data store and automatically determines its structure to create the metadata for your table. Click on **Continue**.

5. Enter Crawler name as `<username>-crawler-yellow` and Click **Next**.

6. Select 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, created from the previous section, and click on **Next**.



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, e.g. `<username>-glue-etl-role`, and 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, under **Database**, click the drop-down and selerct the database used for Lab 1 and Lab 2, i.e. `<username>`. For **Prefix added to tables**, enter `<username>_new_` and click **Next**.

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

14. After the Crawler has finished, 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, select the `<username>` database and you should see under Tables, `<username>_new_yellow`. If not **Manually refresh the table list** by clicking the **Refresh** icon.

17. In the query editor on the right, type

```
select count(*) from <username>_new_yellow;
```

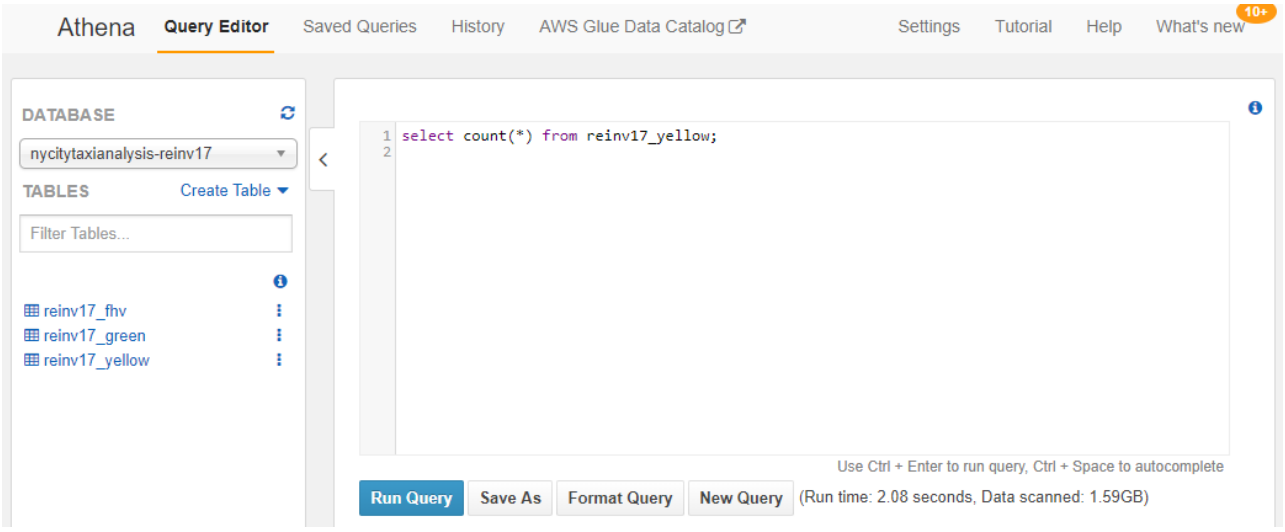
and take note the Run Time and Data scanned numbers here.

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

18. Now we compare the performance to the original table created in Lab 1. In the query editor on the right, type

```
select count(*) from <username>_yellow;
```

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



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

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. Additionally, you can see that since 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. The following table highlights this with the approximate results you should have seen from steps **17** and **18** above.

	Query	Run Time	Data Scanned	Results
`<username>_new_yellow`	select count(*) from `<username>_new_yellow`	~1.1 seconds	0KB	10906858
`<username>_new`	select count(*) from `<username>_yellow`	~21.98 seconds	~1.59GB	10906858

License