Serverless Data Prep with AWS Glue

Workshop Lab Guide

TABLE OF CONTENTS

I.1 **Learn** Workshop Goals 2

I.2 **Get** your AWS Account Ready 4

I.2.1 Setup pre-requisite resources *4*

II.1 **Explore** raw dataset 6

II.1.1 Create and run an AWS Glue Crawler *6*

II.1.2 Explore table schema and metadata *7*

II.1.3 Query raw data with Amazon Athena *8*

II.2 **Create** an optimized dataset 10

II.2.1 Create an Amazon SageMaker notebook instance *11*

II.2.2 Interactively author and run ETL code in Jupyter *11*

II.3 **Explore** optimized dataset 13

II.3.1 Catalog optimized data with an AWS Glue crawler *13*

II.3.2: Query optimized data with Amazon Athena *15*

II.4 **Setup** an AWS Glue ETL pipeline 16

II.4.1 Schedule AWS Glue crawlers *17*

II.4.2 Create an AWS Glue job *18*

II.4.3 Create an AWS Glue Trigger *20*

Advanced exercise: AWS Glue job bookmarks *21*

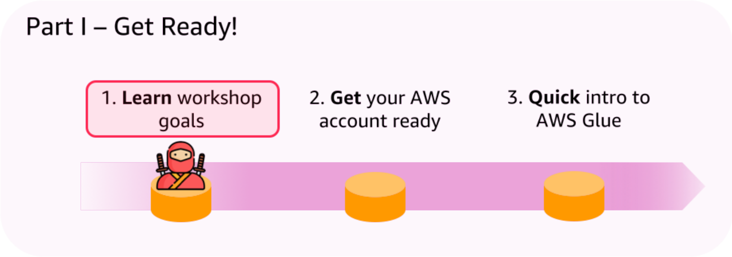
II.5 **Solve** a Machine Learning Problem 23

II.5.1: Interactively develop a Machine Learning model in Jupyter *24*

Account clean-up instructions 25

Additional Resources 26

I.1 **LEARN** WORKSHOP GOALS



Part I – Get Ready!

1. **Learn** workshop goals
2. **Get** your AWS account ready
3. **Quick** intro to AWS Glue

You are a data engineer at **AnyCompany**, a ride-hailing startup. You've been asked to help with the following tasks.

Create a dataset for **reporting and visualization**

Cleanse

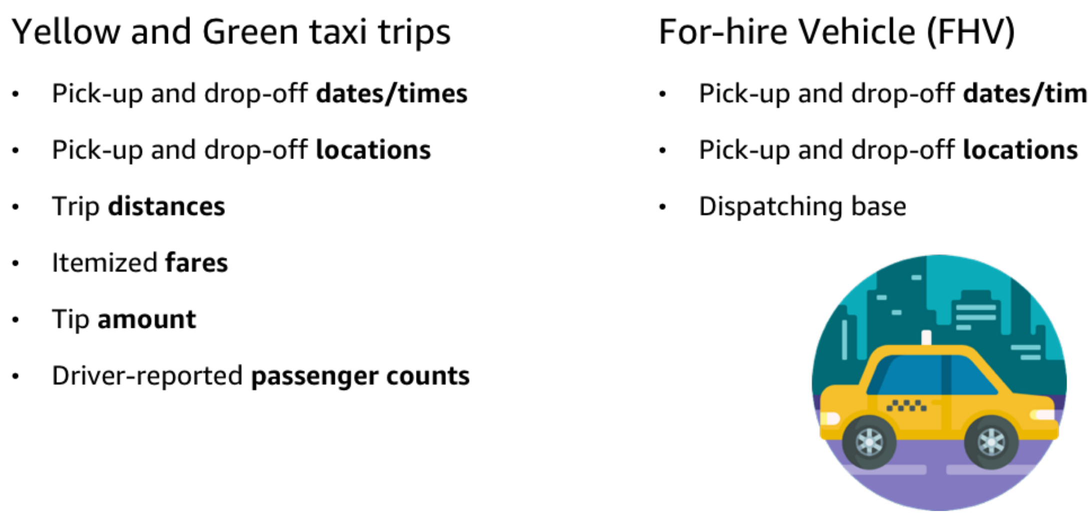
Transform

Optimize for reporting queries

Help solve a **Machine Learning** problem

AnyCompany's data scientists need to understand **passenger tipping behavior**

The dataset that you'll use is the New York City Taxi Trips dataset. The dataset consists of taxi trip records of three kinds of NYC taxis: Yellow, Green, and For-hire Vehicles (FHV). More on this dataset can be found online here. Here's a sample of the fields for Yellow, Green, and FHV trips.



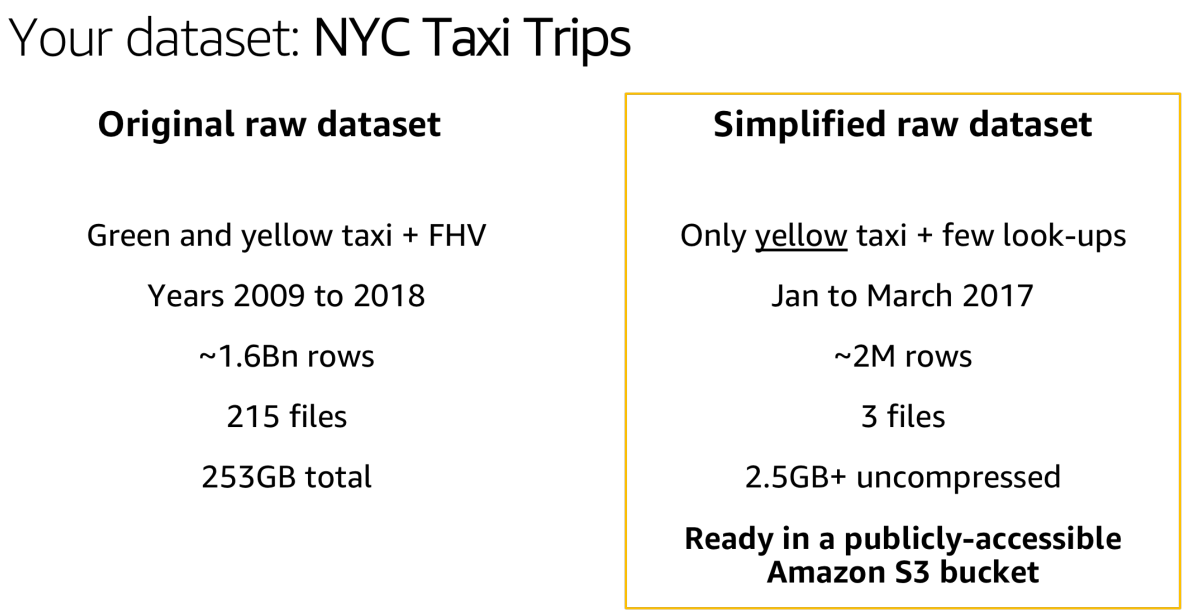
Yellow and Green taxi trips

- Pick-up and drop-off **dates/times**
- Pick-up and drop-off **locations**
- Trip **distances**
- Itemized **fares**
- Tip **amount**
- Driver-reported **passenger counts**

For-hire Vehicle (FHV)

- Pick-up and drop-off **dates/times**
- Pick-up and drop-off **locations**
- Dispatching base

The entire raw dataset covers 9 years and has 1.6 Billion rows. For practical purposes, we'll work in this workshop with a **simplified raw** dataset instead. Here are some key characteristics of the original raw and the simplified raw datasets.



Your dataset: **NYC Taxi Trips**

Original raw dataset	**Simplified raw dataset**
Green and yellow taxi + FHV	Only yellow taxi + few look-ups
Years 2009 to 2018	Jan to March 2017
~1.6Bn rows	~2M rows
215 files	3 files
253GB total	2.5GB+ uncompressed
	Ready in a publicly-accessible Amazon S3 bucket

I.2 **GET** YOUR AWS ACCOUNT READY

Part I – Get Ready!

1. **Learn** workshop goals

2. **Get** your AWS account ready

3. **Quick** intro to AWS Glue

<u>NOTE:</u> If you have received AWS credentials to use for the workshop, then we have already created and setup an AWS account for you. Please follow the instructions in this side note to retrieve your Amazon S3 bucket name.

1. Navigate to **console.aws.amazon.com**
2. Sign into your AWS account with your credentials
3. From the top-right menu, change region to **Ireland (eu-west-1)**
4. Navigate to the **AWS CloudFormation** console
5. Click on the stack named **ant313**
6. Expand the **Outputs** tab, then copy the **value** of **S3BucketName**.

Keep it in an open text file for reference in later exercises.

Once done, skip the remainder of the "Get your AWS account ready" section.

Follow these steps to get your own AWS account ready. Make sure you have sufficient privileges in your AWS account.

I.2.1 SETUP PRE-REQUISITE RESOURCES

We will use an AWS CloudFormation template to setup a number of required resources for the workshop.

1. Navigate to console.aws.amazon.com
2. Sign into your AWS account with your credentials

3. From the top-right menu, change region to **Ireland (eu-west-1)**
4. Navigate to the **AWS CloudFormation** console
5. Click **Create Stack**
6. Select **Specify an Amazon S3 template URL**.
7. Copy and paste the following Amazon S3 URL:

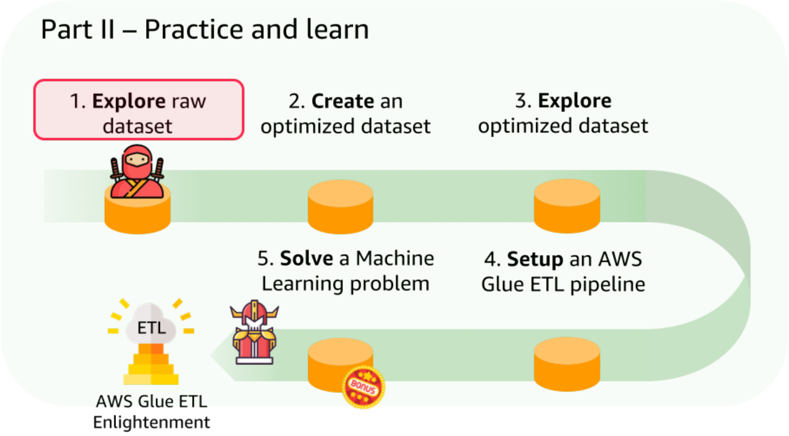
   http://bit.ly/2YI32vU
8. Click **Next**. This takes you to **Specify Details** step.
9. For **Stack name**, enter **ant313** (or you can choose any other name)
10. Click **Next**. In the **Options** step, click **Next** again.
11. Check next to **I acknowledge that AWS CloudFormation might create IAM resources with custom names**
12. Click **Create.** The AWS CloudFormation stack takes **about 7 minutes** to create.
13. After the stack creates successfully, go to the **Outputs** tab
14. Look for the key **S3BucketName**. **Copy and keep** its value. This is your Amazon S3 bucket's name that we will use throughout the workshop.

The AWS CloudFormation template will create the following key resources for you.

- **A new data lake Amazon S3 bucket** – copy and paste that name and keep it for your reference
- **Necessary IAM policies and roles** for AWS Glue, Amazon Athena, and Amazon SageMaker, and AWS Lambda
- An **AWS Glue** development endpoint
- A number of **named queries in Amazon Athena**
- An **AWS Lambda function** that can copy NYC Taxi trips raw dataset files into your Amazon S3 bucket

Finally, the **NYC Taxi trips raw dataset is copied** into your Amazon S3 bucket. Check out the AWS CloudFormation template for the complete list of resources created.

II.1 **EXPLORE** RAW DATASET


Part II – Practice and learn
1. Explore raw dataset
2. Create an optimized dataset
3. Explore optimized dataset
5. Solve a Machine Learning problem
4. Setup an AWS Glue ETL pipeline
ETL
AWS Glue ETL Enlightenment
BONUS

II.1.1 CREATE AND RUN AN AWS GLUE CRAWLER

One of the first steps to exploring a dataset is understanding its schema and its other properties. AWS Glue crawlers can help us with that. A crawler scans your data, classifying its format (CSV, TSV, et.) and inferring its schema including fields and potential datatypes. Crawlers also collect useful metadata such as total record count, total dataset size, number of files scanned, type of compression used, and more. After a crawler completes, it writes a dataset's schema and properties to the AWS Glue Data Catalog.

Let's run an AWS Glue crawler on the raw NYC Taxi trips dataset.

1. Navigate to the **AWS Glue** console
2. In the left menu, click **Crawlers** → **Add crawler**
3. On **Crawler info** step, enter crawler name ***nyctaxi-raw-crawler*** and write a description.
4. Click **Next**
5. On **Data store** step…
    a. Choose **Specified path in my account**
    b. In **Include path**, copy-and-paste an NYC Taxi dataset S3 URL:

        s3://**<your_bucket_name>**/data/raw/nyctaxi/

Replace **<your_bucket_name>** with your actual bucket's name.

   c. Click **Next** then **Next** once again.

6. On **IAM role** step…

   a. Click **Choose an existing IAM role**

   b. In **IAM Role** menu, choose ***AWSGlueCrawlerRole-nyctaxi***

7. On **Schedule** step, click **Next**

8. On **Output** step…

   a. Click **Add database** → enter **nyctaxi** as the name → click **Create**

   b. Expand Configuration options → select **Add new columns only**

   c. Click **Next**

9. On **Review all steps**, click **Finish**.

10. Finally, click on the green **Run it now?** prompt at the top of the crawlers page.

Your crawler should immediately start. Upon completion, it adds four tables in the **nyctaxi** database in your AWS Glue data catalog: **yellow**, **paymenttype**, **ratecode**, and **taxizone**.

II.1.2 EXPLORE TABLE SCHEMA AND METADATA

Now that we have cataloged the raw NYC Taxi trips dataset using a crawler, let's explore the crawler's output in the AWS Glue data catalog.

1. Navigate to the AWS Glue console

2. In the left menu, under **Data Catalog**, click **Tables**

3. Note the list of NYC Taxi dataset tables cataloged by your crawler: **yellow**, **paymenttype**, **ratecode**, and **taxizone**.

4. Click on the **yellow** table

5. Notice table version and table properties

a. Notice **Classification**, showing the table was classified as **CSV**

b. Notice **objectCount**, showing how many files were found

c. Notice **sizeKey** property, showing total dataset size

d. Notice **recordCount** property, showing total number of records

e. Notice **compressionType**, showing the files are compressed in **bzip2**

6. Scroll down, **examine table schema**. Fields have been identified along with their potential datatypes. You can manually change any datatype.

II.1.3 QUERY RAW DATA WITH AMAZON ATHENA

Beginning with the end in mind, you know that AnyCompany's business end users need a solution for ad-hoc analysis, reporting, and visualization. Evaluating the options based on requirements, you decide on proposing Amazon Athena along with Amazon QuickSight.

Amazon Athena is a serverless interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. It integrates with the AWS Glue data catalog and takes advantage of the metadata registered by AWS Glue crawlers. **Amazon QuickSight** is a fast, cloud-powered BI service that makes it easy to build visualizations, perform ad-hoc analysis, and quickly get business insights from your data.

Because Amazon Athena will act as your data lake's querying service, you decide to use Athena to explore the raw NYC Taxi trips dataset. The goal here is to see how Amazon Athena performs on the raw dataset and establish a performance baseline.

Let's run two SQL queries in Amazon Athena: a sample reporting query (***sample_report_qry***) and a sample aggregation query (***sample_agg_qry***). Both queries represent typical reporting and analytics queries that AnyCompany's business end users may want to run on the dataset.

1. Navigate to **Amazon Athena** console, click **Get Started**. Click **X** to skip tutorial.

2. Ensure database **nyctaxi** is selected, and the four tables are listed: **yellow**, **paymenttype**, **ratecode**, and **taxizone**.

3. Click on the tab **Saved Queries**

4. In the search field, type ***nyctaxi:raw***

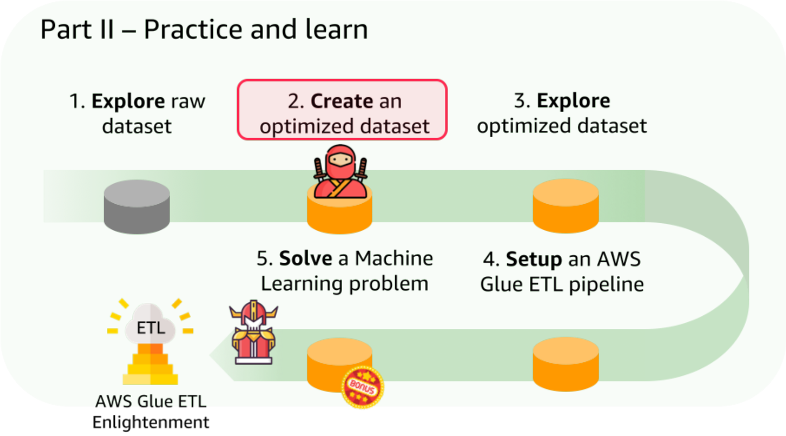
5. Click on the ***sample_report_qry*** query to open in Athena query editor

Note: After opening the query, review the comment before the SQL statement.

6. Click **Run query**
7. Note **query run time** and **data scanned**. Copy and paste it into an open text file for later comparison.
8. Repeat **steps 3-7** for query ***sample_agg_qry***

The query run time and data scanned values you obtained are a result of running the queries directly on the raw dataset. Later on, we'll create an optimized dataset and query it again. And then, we'll be able to compare query run times and data scanned on raw dataset versus optimized dataset.

II.2 **CREATE** AN OPTIMIZED DATASET


Part II – Practice and learn
1. **Explore** raw dataset
2. **Create** an optimized dataset
3. **Explore** optimized dataset
5. **Solve** a Machine Learning problem
4. **Setup** an AWS Glue ETL pipeline
ETL
AWS Glue ETL Enlightenment
BONUS

After running test SQL queries in Amazon Athena, you verify that you can make the following improvements to the NYC Taxi trips dataset. Ultimately, these improvements together reduce per-query Amazon Athena cost and improve AnyCompany's business users' experience.

- **Convert the dataset from CSV** to a compressed and splittable columnar file format, such as Parquet. This enables Amazon Athena to optimize query execution by avoiding scanning columns that are not needed by your query and rows that are filtered out by your query.
- **Partition the dataset** in Amazon S3 by year and month into Hive-style partitions. The partitioning scheme enables queries that filter on a specific year or month to avoid scanning unnecessary data for other years and months, thereby acting as a crude index.
- **De-normalize the dataset** by joining the primary fact table ***yellow*** with the dimension tables ***taxizone***, ***ratecode***, and ***paymenttype***. This creates a new table that is simpler to query and run reports on.
- **Create new columns and drop unnecessary columns**.

To develop and test code to apply such transformations on the large NYC Taxi trips dataset, you consider using an Apache Spark environment that can access and manipulate your data directly in Amazon S3. You decide to take advantage of AWS Glue development endpoints for that purpose. Also, instead of setting up and managing your own Zeppelin or Jupyter notebook server, you leverage AWS Glue's ability to launch a fully-managed Jupyter notebook instance for interactive ETL and Machine Learning development. Your Jupyter notebook will use your AWS Glue development endpoint to run your ETL code written in PySpark or Scala.

Let's go through the steps to setup your interactive ETL notebook environment and apply the above transformations.

II.2.1 CREATE AN AMAZON SAGEMAKER NOTEBOOK INSTANCE

First, we'll ensure your AWS Glue dev endpoint is **READY** by following these steps. The AWS Glue dev endpoint was automatically created for you by the AWS CloudFormation template your ran in section "I.2 Get your AWS account ready".

1. Navigate to the AWS Glue console
2. In the left menu, under **ETL**, click **Dev endpoints**
3. Verify there is a dev endpoint named **nyctaxi-dev-endpoint**-*****
4. Verify that the status is **READY**

Proceed to create a SageMaker notebook instance. Follow these steps.

1. In the left menu, under **ETL** → **Dev endpoints**, click **Notebooks**
2. Click **Create notebook**
3. For **Notebook name**, enter ***nyctaxi-notebook***
4. For **Attach to development endpoint**, select **nyctaxi-dev-endpoint-*****
5. Select **Choose an existing IAM role** then choose ***AWSGlueServiceSageMakerNotebookRole-nyctaxi***
6. Click **Create notebook**. This will take you back to the **Notebooks** page.
7. Click the **Refresh** 🔄 button, if needed. You should notice a notebook with the name **aws-glue-nyctaxi-notebook** and the status **Starting**.

The Amazon SageMaker notebook should take **about 6 minutes** to transfer into a **Ready** status.

II.2.2 INTERACTIVELY AUTHOR AND RUN ETL CODE IN JUPYTER

First, you'll download the ETL notebook we prepared for you. Then, you will upload the notebook to your own Amazon SageMaker notebook, from the AWS Glue console.

1. Download notebook file nyctaxi_raw_dataset_etl.ipynb

2. Navigate to the **AWS Glue** console

3. In the left menu, under **ETL** → **Dev endpoints**, click **Notebooks**

4. Select **aws-glue-nyctaxi-notebook**, then click **Open notebook** and then **OK**. A new browser tab will open showing **Jupyter** interface

5. In **Jupyter**, in the upper right corner, click the **Upload** button

6. Browse to and select **nyctaxi_raw_dataset_etl.ipynb**, then in Jupyter, click **Upload again.** After upload, the notebook should appear in Jupyter.

7. Click on **nyctaxi_raw_dataset_etl.ipynb** to open notebook. Spend a few moments to have an overview of the documented ETL code and what it does.

Next, you'll edit the ETL code in the notebook and then run it on the raw NYC Taxi trips dataset.

1. In **Jupyter**, ensure the **nyctaxi_raw_dataset_etl.ipynb** notebook is open

2. In the **first PySpark code cell**, look for the variable **your_bucket_name**

3. Set the value of that string variable to **your own Amazon S3 bucket's name**

4. Click the **save button** 💾 to save the change you made to your notebook.

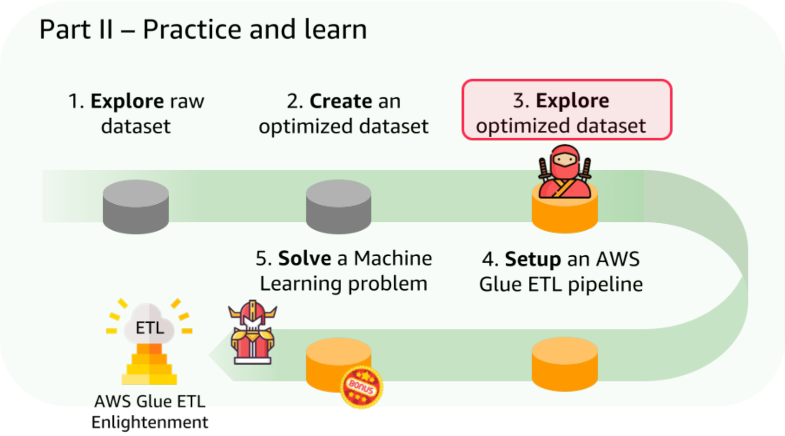
5. Click on the **Cells** menu → **Run All**

6. This will run ETL code in the notebook. The code finally creates a new dataset under the Amazon S3 prefix in your own account:

   s3://<your_bucket_name>/data/**staging**/nyctaxi/**yellow_opt**

The notebook should take **about 8 minutes** to complete.

II.3 **EXPLORE** OPTIMIZED DATASET


Part II – Practice and learn
1. Explore raw dataset
2. Create an optimized dataset
3. Explore optimized dataset
5. Solve a Machine Learning problem
4. Setup an AWS Glue ETL pipeline
ETL
AWS Glue ETL Enlightenment
Bonus

In the previous step, you created a new dataset that is optimized for querying and reporting. Now, it's time to test the optimizations you've done using SQL queries in Amazon Athena. What tangible **improvement in query performance** and **reduction of data scanned** have your optimizations achieved?

Let's find out by performing the following steps.

II.3.1 CATALOG OPTIMIZED DATA WITH AN AWS GLUE CRAWLER

To be able to query the new optimized NYC Taxi trips dataset with Amazon Athena, we need to catalog the dataset using an AWS Glue crawler.

1. Navigate to the AWS Glue console
2. In the left menu, click **Crawlers** → **Add crawler**
3. On **Crawler info** step, enter crawler name **nyctaxi-optimized-crawler** and write a description. Click **Next**.
4. On **Data store** step…
    a. Choose **Specified path in my account**
    b. In **Include path,** enter:

       s3://**<you_bucket_name>**/data/**staging**/nyctaxi/**yellow_opt**/

       Replace **<your_bucket_name>** with the actual name of your Amazon S3 bucket.

c. Click **Next**, then for **Add another data store**, select **Yes**. Click **Next**, again.

d. Repeat steps **(a.)** to **(c.)** for the following NYC Taxi dataset S3 path:

s3://**<you_bucket_name>**/data/**prod**/nyctaxi/**yellow_rpt/**

Note that this path does NOT exist yet. Why do you need to add this S3 path? Later on, in the exercise 'Advanced Exercise: AWS Glue Job Bookmarks', we'll generate a new optimized dataset for **production use** under that path. Then, we'll need to crawl it and catalog it. In a real scenario, you may choose to create a separate crawler to crawl that production dataset path specifically. However, to save time and avoid repetition in this lab, we'll use a single crawler – the one you're creating here – to crawl both datasets.

5. On **IAM role** step…

   a. Click **Choose an existing IAM role**

   b. In **IAM Role** menu, choose ***AWSGlueCrawlerRole-nyctaxi***

6. On **Schedule** step, click **Next**

7. On **Output** step…

   a. For **Database**, select **nyctaxi**.

   b. Expand **Configuration options** → select **Add new columns only**

   We choose this option when we plan on making changes to the table schema manually later on. In such a case, we do not want the AWS Glue crawler to override our changes in subsequent runs.

   c. Click **Next**

8. On **Review all steps**, click **Finish**

9. Finally, click on the green **Run it now?** prompt at the top of the crawlers page.

The crawler will run over the new optimized NYC Taxi trips dataset. It will then create a new table, called **yellow_opt** in the AWS Glue data catalog.

Let's take a look at what the crawler created.

1. In the left menu, under **Data Catalog**, click **Tables**

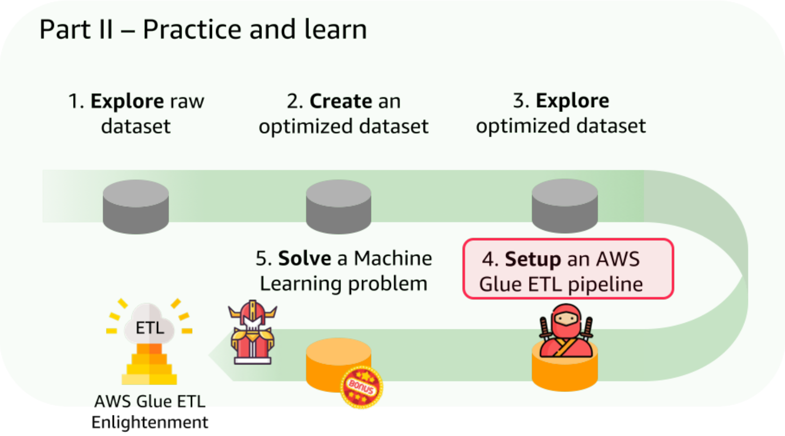
2. Click on the **yellow_opt** table
3. Notice **table properties**
4. Scroll down, **examine table schema**. Notice the many new fields added. Also notice two new fields **pu_year** and **pu_month** and how they were identified as partitioning columns.

II.3.2: QUERY OPTIMIZED DATA WITH AMAZON ATHENA

Now that the data has been cataloged, let's test the sample reporting and aggregation queries **on the optimized NYC Taxi trips dataset**.

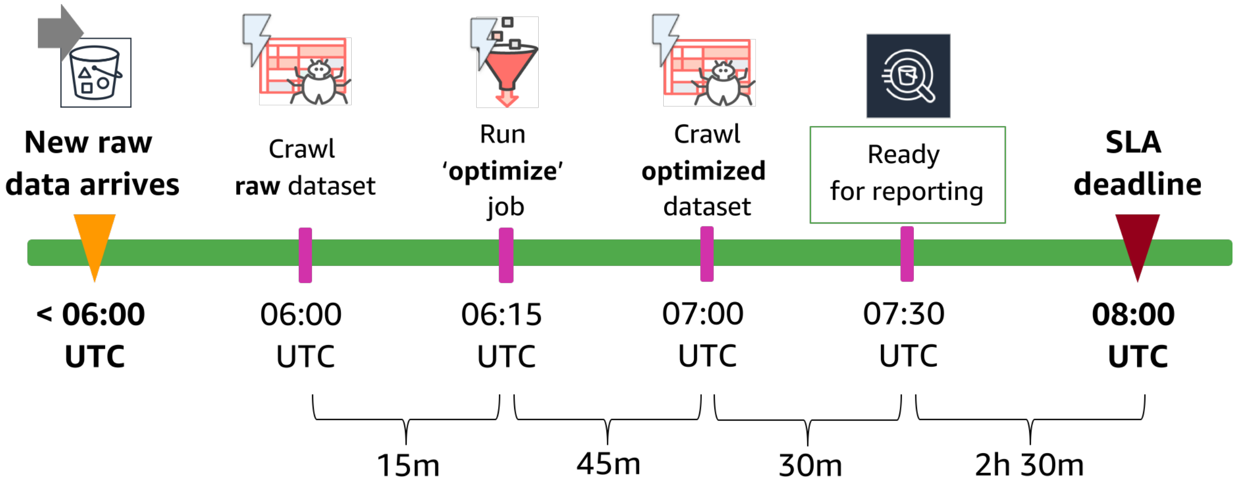
1. Navigate to **Amazon Athena** console
2. Ensure database **nyctaxi** is selected.
3. Click the **Refresh** 🔄 button, notice how Athena detected the new **yellow_opt** table.
4. Click on the tab **Saved Queries**
5. In the search field, type **nyctaxi:opt**
6. Click on the query ***sample_report_qry*** to open in Athena query editor
7. Click **Run query**
8. Observe **query run time** and **data scanned.** Compare with the results you saved from section "**II.1.3 Query raw data with Amazon Athena"** and notice the improvements.
9. Repeat **steps 3-7** for query ***sample_agg_qry***

II.4 **SETUP** AN AWS GLUE ETL PIPELINE


Part II – Practice and learn
1. **Explore** raw dataset
2. **Create** an optimized dataset
3. **Explore** optimized dataset
4. **Setup** an AWS Glue ETL pipeline
5. **Solve** a Machine Learning problem
ETL
AWS Glue ETL Enlightenment
Bonus

In the previous sections, you explored raw NYC Taxi trips dataset. You then developed and tested ETL code to transform the raw NYC Taxi trips dataset into a new dataset that is optimized for querying and reporting by AnyCompany's business end users. Now, it's time to deploy your code into a production ETL pipeline using AWS Glue.

In this section, we'll build a **schedule-driven** AWS Glue ETL pipeline that looks as follows. We work our way backwards from a daily data availability Service Level Agreement (SLA) goal. AnyCompany's business end users need their datasets to be refreshed daily and available by that time.



II.4.1 SCHEDULE AWS GLUE CRAWLERS

We'll start by scheduling the **raw dataset AWS Glue crawler**.

1. Navigate to the AWS Glue console.
2. In the left menu, click **Crawlers**
3. Click on **nyctaxi-raw-crawler**, then click **Edit**
4. In the left list of steps, click on **Schedule**
5. For **Frequency**, select **Daily**
6. Select **06:00 UTC**
7. In the left list, click on **Review all steps**
8. Scroll down and click **Finish**

Next, we'll schedule the **optimized** dataset AWS Glue crawler.

1. Navigate to the AWS Glue console.
2. In the left menu, click **Crawlers**
3. Click on **nyctaxi-optimized-crawler**, then click **Edit**
4. In the left list of steps, click on **Schedule**
5. For **Frequency**, select **Daily**
6. Select **07:00 UTC**
7. In the left list, click on **Review all steps**
8. Scroll down and click **Finish**

II.4.2 CREATE AN AWS GLUE JOB

The next step in setting up our AWS Glue ETL pipeline is to create an AWS Glue job.

1. Navigate to the AWS Glue console
2. In the left menu, under **ETL**, click **Jobs.** Click **Add job.**
3. In step **Job properties**…
   a. For **Name**, enter ***nyctaxi-create-optimized-dataset***
   b. For **IAM Role**, select ***AWSGlueServiceRole-nyctaxi-optimize-job***
   c. For **This job runs**, select ***An existing script that you provide***
   d. For **S3 path where the script is stored**, copy-and-paste this S3 URL:

      s3://**<you_bucket_name>**/scripts/nyctaxi_create_optimized_dataset_job.py

      Replace **<your_bucket_name>** with the actual name of your Amazon S3 bucket.
   e. For **Temporary directory**, specify the following S3 URL:

      s3://**<your_bucket_name>**/data/tmp
   f. Expand section **Advanced properties**
      i. For **Job bookmark**, select **Enable**
      ii. For **Job metrics**, select **Enable**
   g. Expand section **Security configuration, script libraries, and job parameters**
      i. For **Concurrent DPUs per job run**, enter **4**
      ii. Find the section named **Job parameters**
      iii. Under **Key**, enter --**your_bucket_name** (two dashes, then the word **your_bucket_name**)
      iv. Under **Value**, enter **your actual Amazon S3 bucket name**
   h. Click **Next**
4. In step **Connections**, click **Next**

5. In step **Review**, click **Save job and edit script**
6. Review the script. Notice AWS Glue **job setup and teardown code**.
7. Click **X** on the top right to return to AWS Glue console. You have successfully created an AWS Glue job.

You can run this AWS Glue job any time. With 4 DPUs allocated, it takes **about 7-9 minutes** from a cold start to complete. The job creates and writes an optimized NYC Taxi (Yellow) dataset to the following Amazon S3 path in your own account:

s3://<your_bucket_name>/data/**prod**/nyctaxi/**yellow_rpt**

You can download the job's script here.

TRY IT OUT LATER

You can have AWS Glue auto-generate an ETL script for you! You can then edit and customize the script to your needs.

To accomplish that, in **step 3.c** of the previous exercise, for **This job runs,** try to select **A proposed script generated by AWS Glue** instead. Follow the steps to create a fully-functional ETL script.

II.4.3 CREATE AN AWS GLUE TRIGGER

Finally, we'll create a **time-based** AWS Glue trigger to trigger our job through the following steps.

1. Navigate to the AWS Glue console
2. In the left menu, under **ETL**, click **Triggers**, then click **Add trigger** button
3. In step **Trigger properties**'…
    a. For **Name**, enter ***nyctaxi-process-raw-dataset***
    b. For **Trigger type**, select **Schedule**
    c. For **Frequency**, select **Custom**
    d. For **Cron expression**, enter **15 07 ? * * * (mind the spaces!)**
    e. Click **Next**
4. In step **Jobs to start**…
    a. Under **Job**, look for **nyctaxi-optimize-raw-dataset**. Click **Add** next to it.
    b. Leave **Job bookmark** set to **Enabled**
    c. Click **Next**
5. In step **Review all steps**, check **Enable trigger on creation**
6. Click **Finish**

Congratulations! You have successfully set up your AWS Glue ETL pipeline. The pipeline will run on schedule. You can check your AWS Glue console at the times you scheduled to ensure that your crawlers and your AWS Glue job have run.

ADVANCED EXERCISE: AWS GLUE JOB BOOKMARKS

Let's try out the AWS Glue's job bookmarks feature, which enables production ETL jobs to process **only new data** when rerunning on a scheduled interval.

We'll start by simulating an initial run of the production ETL pipeline to process data files pertaining to January and February 2017.

1. Let the ETL pipeline run once on the scheduled time.

   Alternatively, you can manually start the AWS Glue job you've just created and wait for it to complete. Navigate to **Jobs console.** Check **nyctaxi-create-optimized-dataset**. Click **Action** → **Run job**. Expand **Advanced properties**, and for **Job bookmark**, ensure **Enable** is selected. Click **Run**.

   When the job completes, it will have created a new production dataset covering January and February 2017.

2. Let's crawl the new production dataset. Navigate to **Crawlers console**. Re-run the optimized dataset crawler: **nyctaxi-optimized-crawler**. Crawler will detect the new dataset and add it to a table named **yellow_rpt**.

3. Navigate to the **Amazon Athena console**. Ensure database **nyctaxi** is selected. **Refresh** tables list. **Copy, paste, and run** the following query against the newly-created production reporting dataset table, **yellow_rpt**.

   SELECT count(*) AS march_count FROM yellow_rpt WHERE cast(pu_month AS BigInt) = 3 GROUP BY yellow_rpt.pu_month

   Notice that there is a **zero count** for March.

Now, let's simulate as if a new file for March 2017 was added to our raw dataset.

4. Navigate to the **AWS Lambda** console. On the left menu, click **Functions**

5. In the **Functions list**, select function starting with **copy_raw_nyctaxi_data**

6. Click on **Actions** → **Test**

7. On the **Configure test event** dialog…

a. Select **Create new test event**

b. For **Event name**, enter **IngestMarch2017**

c. For **Event body**, copy and ***replace*** existing body with the JSON text below:

```
{
  "s3_prefix": "data/raw/nyctaxi/yellow/yellow_tripdata_2017-03.csv.bz2"
}
```

d. Click **Create**

8. Next, on the top right, click the **Test** button. Wait until you receive **Execution result: succeeded**. The March 2017 file has been ingested into your Amazon S3 bucket!

Finally, we'll manually re-run the production ETL job and verify that it processed the newly-ingested file for March 2017.

9. Navigate to the **AWS Glue** console. Click on **Jobs**

10. Manually run the **nyctaxi-create-optimized-dataset** job again. Wait for it to complete.

11. Navigate to **Crawlers**, and re-run the optimized dataset crawler **nyctaxi-optimized-crawler**.

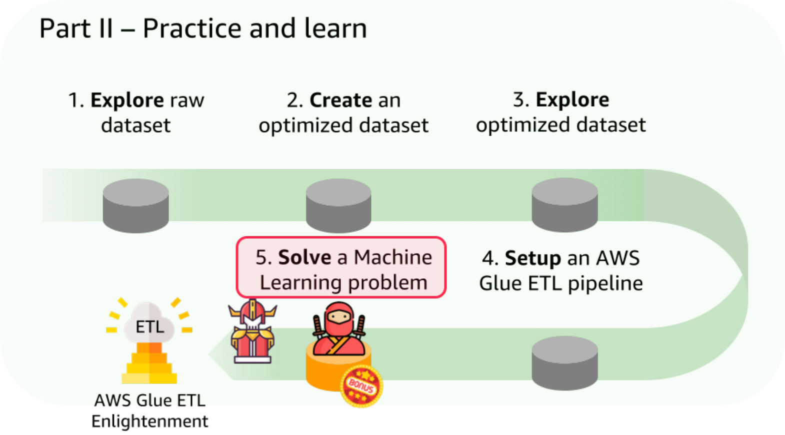
12. Repeat **step #3**. Does it show a count for March 2017 records?

Optionally, you can also navigate to Amazon S3 console and browse the following locations and verify that the Glue job did not re-process January and February data and append duplicate files. Check the number of files in the following S3 locations and ensure there are 31 objects for January and 28 objects for February:

s3://**<you_bucket_name>**/data/**prod**/nyctaxi/**yellow_rpt**/pu_year=2017/pu_month=1/

s3://**<you_bucket_name>**/data/**prod**/nyctaxi/**yellow_rpt**/pu_year=2017/pu_month=2/

II.5 **SOLVE** A MACHINE LEARNING PROBLEM

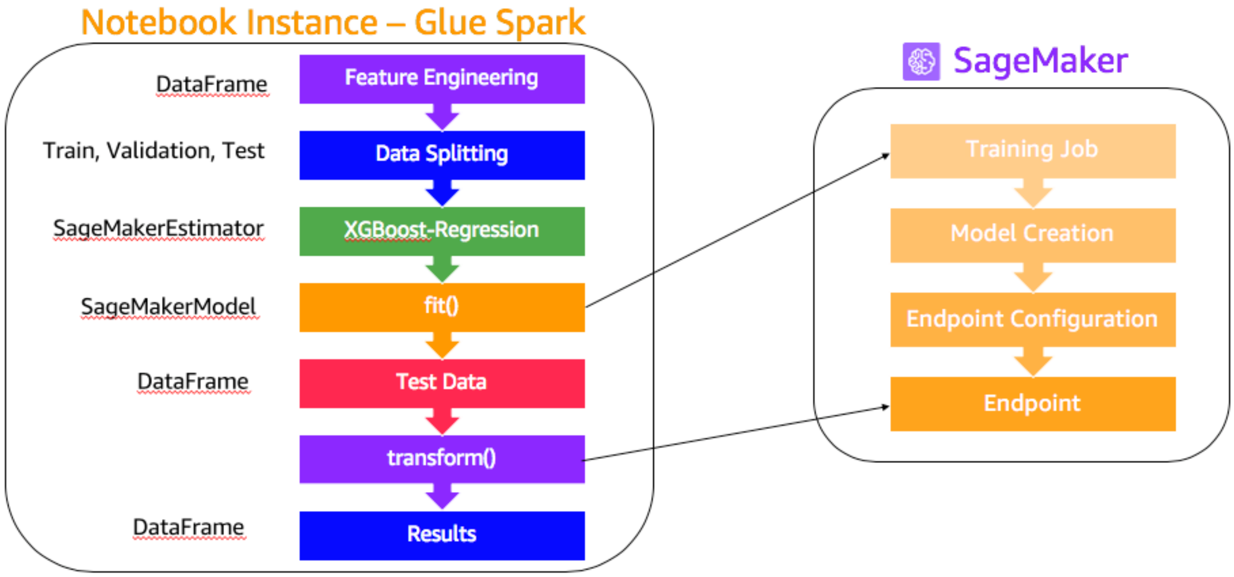

Part II – Practice and learn
1. **Explore** raw dataset
2. **Create** an optimized dataset
3. **Explore** optimized dataset
4. **Setup** an AWS Glue ETL pipeline
5. **Solve** a Machine Learning problem
ETL
AWS Glue ETL Enlightenment
Bonus

So far, we have curated and optimized the NYC Taxi dataset that can be utilized for analytics using Business Intelligence tools or by Data science team to solve key business problems as predictions. **Amazon SageMaker** is a fully managed machine learning service. With Amazon SageMaker, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment.

It is interesting to combine Glue's spark ML capabilities of feature engineering, orchestrating of SageMaker training, hosting model, end point creation and predictions via Glue SageMaker notebook.

In this section, we will utilize optimized NYC Taxi dataset to perform Tip Amount prediction for a particular ride. Below are high level steps.

- **Data Cleansing** – Remove biased features (e.g. remove samples with payment_type as 'CASH' as most drivers don't report tips for cash payments)
- **Feature Trend Analysis** – Understand trends in data relative to label (Tip Amount) to remove biased samples
- **Feature Engineering** – Extract features from raw data and transform features to make them suitable for training (e.g. convert day of week string to numeric and then binary vector)
- **Data Split** – Split data into Training dataset and Test dataset
- **Training and Hosting Model –** Using SageMaker spark SDK, launch XGBoost training, model creation and endpoint creation
- **Run Predictions –** Derive predictions using Spark transform () method



II.5.1: INTERACTIVELY DEVELOP A MACHINE LEARNING MODEL IN JUPYTER

1. Download nyctaxi_tips_prediction_xgboost.ipynb notebook file
2. Navigate to the AWS Glue console
3. In the left menu, under **ETL** → **Dev endpoints**, click **Notebooks**
4. Select **aws-glue-nyctaxi-notebook**, then click **Open** and **OK**. A new browser tab will open showing **Jupyter** interface
5. In Jupyter, in the upper right corner, click the **Upload** button and upload the notebook file.
6. Ensure the kernel is set to "Sparkmagic (PySpark)" in the upper right corner.
7. **Run one cell at a time** by clicking each cell from top and clicking ▶ Run
8. Observe output of each cell and also refer to documentation links provided in notebook along with code as appropriate

You've made it!

ACCOUNT CLEAN-UP INSTRUCTIONS

Once you are done with the lab, follow the instructions below to clean-up your account.

First, delete manually-created AWS Glue resources.

1. Navigate to the **AWS Glue** console
2. Go to **Databases** then select **nyctaxi**. Click **Action**, then **Delete database**. Confirm deletion.
3. Go to **Crawlers** then select **nyctaxi-optimized-crawler**. Click **Action**, then **Delete crawler**. Confirm deletion. Repeat for **nyctaxi-raw-crawler**.
4. Go to **Jobs**, then select **nyctaxi-create-optimized-dataset**. Click **Action**, then **Delete**. Confirm deletion.
5. Got to **Triggers**, then select **nyctaxi-process-raw-dataset**. Click **Action**, then **Delete**. Confirm deletion.
6. Go to **Notebooks**, then select **aws-glue-nyctaxi-notebook**. Click **Action**, then **Stop**. Wait until notebook status changes to **Stopped**. Click **Action** again, then **Delete**.

Finally, delete the workshop's AWS CloudFormation stack to clean-up remaining resources.

7. Navigate to the **AWS CloudFormation** console
8. Select the workshop's stack (e.g. ant313). Click **Actions**, then **Delete stack**. Confirm deletion.
9. Stack status will change to **DELETE_IN_PROGRESS**. The stack should take less than a minute to be deleted.

You're done cleaning-up your account!

ADDITIONAL RESOURCES

This workshop is accompanied by video walkthroughs for the following activities.

I.2 Get your AWS account ready

II.1 Explore raw dataset

II.2 Create an optimized dataset

II.3 Explore optimized dataset

For any questions or comments, please reach out to ravirala@amazon.com