https://docs.databricks.com/data/databricks-datasets.html

Here is the code to read the flights data and load to a data frame, as shown in Figure 13-1.

flights = spark.read.format("csv") \

.option("header", "true") \

.option("inferSchema", "true") \

.load("/databricks-datasets/asa/airlines/2008.csv")

Run the following script which will create a mount point to an Azure Data Lake Storage Gen2 account where the data will be persisted. Ensure that the access key is replaced in the script below.

spark.conf.set(

"fs.azure.account.key.rl001adls2.dfs.core.windows.net",

"ENTER-ACCESS\_KEY\_HERE"

)

Run the following code to write the flights data frame to the data lake folder in Delta format and partitioned by the Origin column.

(

flights

.write

.partitionBy("Origin")

.format("delta")

.mode("overwrite")

.save("abfss://data@rl001adls2.dfs.core.windows.net/raw/delta/flights\_delta")

)

the following code will infer the schema and load a data frame with the 2019 yellow trip data.

Data = "/databricks-datasets/nyctaxi/tripdata/yellow/yellow\_tripdata\_2019-\*"

SchemaDF = spark.read.format("csv") \

.option("header", "true") \

.option("inferSchema", "true") \

.load("/databricks-datasets/nyctaxi/tripdata/yellow/yellow\_tripdata\_2019-02.csv.gz")

nyctaxiDF = spark.read.format("csv") \

.option("header", "true") \

.schema(SchemaDF.schema) \

.load(Data)

This next code block will add few partition fields to the existing data frame for Year, Year\_Month, and Year\_Month\_Day. These additional columns will be based on the Datetime stamp and will help with both partitioning, data skipping, Z-ordering and ultimately more performant querying speeds.

from pyspark.sql.functions import \*

nyctaxiDF = nyctaxiDF.withColumn('Year', year(col("tpep\_pickup\_datetime")))

nyctaxiDF = nyctaxiDF.withColumn('Year\_Month', date\_format(col("tpep\_pickup\_datetime"),"yyyyMM"))

nyctaxiDF = nyctaxiDF.withColumn('Year\_Month\_Day', date\_format(col("tpep\_pickup\_datetime"),"yyyyMMdd"))

Create a Hive table using the Delta location with the following script.

spark.sql("CREATE TABLE flights USING DELTA LOCATION 'abfss://data@rl001adls2.dfs.core.windows.net/raw/delta/flights\_delta/'")

Here is the script shown in Figure 13-6 that you will need to run.

%sql

SELECT Count(\*) from flights

Here is the script that you will need to run to recreate the results shown in Figure 13-7.

%sql

SELECT count(\*) as Flights, Dest, Month from flights WHERE DayofMonth = 5 GROUP BY Month, Dest

Here is the code that you will need to run to recreate the results shown in Figure 13-8.

%sql

OPTIMIZE flights ZORDER BY (DayofMonth)

Here is the query you will need to run to recreate the results shown in Figure 13-15.

%sql

SELECT count(\*) as Flights, Origin, Dest, Month from flights WHERE Origin = 'IAD' GROUP BY Origin, Month, Dest