Here is the Scala code that you will need to run to create the required data set. You can either increase or decrease the number from 14 to size your dataset as desired.

%scala

import org.apache.spark.sql.functions.\_

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

val SchemaDF = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("/databricks-datasets/nyctaxi/tripdata/yellow/yellow\_tripdata\_2019-02.csv.gz")

val df = spark.read.format("csv").option("header", "true").schema(SchemaDF.schema).load(Data)

val nyctaxiDF = df

.withColumn("VendorID", explode(array((1 until 14).map(lit): \_\*)))

.selectExpr(df.columns: \_\*)

Here is the Scala code that you will need to run to get the count of the dataset.

%scala

nyctaxiDF.count()

Run the following Scala code to save the dataset to your ADLS gen2 storage account. Notice that the format is set to delta and the mode is overwrite.

%scala

val nyctaxiDF\_bloom = nyctaxiDF.write

.format("delta")

.mode("overwrite")

.save("dbfs:/mnt/rcpdlhcore/datalakehouse/dlhcore/raw/delta/nyctaxi\_bloom")

Here is the Scala code that you will need to run to create non bloom delta dataset in your ADLS gen2 account.

%scala

val nyctaxiDF\_nonbloom = nyctaxiDF.write

.format("delta")

.mode("overwrite")

.save("dbfs:/mnt/rcpdlhcore/datalakehouse/dlhcore/raw/delta/nyctaxi\_nonbloom")

Here is the SQL code that you will need to run to enable bloomFilter for both Spark and Delta.

SET spark.databricks.io.skipping.bloomFilter.enabled = true;

SET delta.bloomFilter.enabled = true;

Here is the SQL code that you will need to run to CREATE OR REPLACE the nyctaxi\_bloom table.

CREATE OR REPLACE TABLE nyctaxi\_bloom (

VendorID int,

tpep\_pickup\_datetime string,

tpep\_dropoff\_datetime string,

passenger\_count int,

trip\_distance double,

RatecodeID int,

store\_and\_fwd\_flag string,

PULocationID int,

DOLocationID int,

payment\_type int,

fare\_amount double,

extra double,

mta\_tax double,

tip\_amount double,

tolls\_amount double,

improvement\_surcharge double,

total\_amount double,

congestion\_surcharge double)

USING DELTA

You can also use the following syntax within your Python notebook to create a Delta table directly from a data frame: nyctaxiDF.write.format("delta").saveAsTable("nyctaxi\_bloom").

Run the following SQL code to truncate the table to ensure there is no data in the newly created table.

%sql

TRUNCATE TABLE nyctaxi\_bloom;

SELECT \* FROM nyctaxi\_bloom;

Here is the SQL code that will give you the count of the nyctaxi\_bloom table.

SELECT count(\*) FROM nyctaxi\_bloom

Here is the SQL code that you will need to run to create the nyctaxi\_nonbloom table.

CREATE TABLE nyctaxi\_nonbloom

USING DELTA

LOCATION 'dbfs:/mnt/rcpdlhcore/datalakehouse/dlhcore/raw/delta/nyctaxi\_nonbloom'

Here is the SQL code that you will need to run count the nyctaxi\_nonbloom rows.

SELECT count(\*) FROM nyctaxi\_nonbloom

Here is the SQL code which you will need to run on the nyctaxi\_bloom table to apply the Bloom Filter Index on the tpep\_dropoff\_datetime column. Leave the fpp to a default of 10% and apply 5% of the number of rows in the nyctaxi\_bloom table as the numItems. In this case 5% of 1 billion is 50million.

CREATE BLOOMFILTER INDEX

ON TABLE nyctaxi\_bloom

FOR COLUMNS(tpep\_dropoff\_datetime OPTIONS (fpp=0.1, numItems=50000000))

Here is the Scala code that you will need to run to get the meta-data of the table, which would include details of a Bloom Filter Index if it has been applied to any of the columns.

%scala

spark.table("nyctaxi\_bloom").schema.foreach(field => println(s"${field.name}: metadata=${field.metadata}"))

Here is the Insert SQL code that you will need to run.

INSERT INTO nyctaxi\_bloom TABLE nyctaxi\_nonbloom;

Here is the SQL code that you will need to run to optimize the nyctaxi\_bloom table and zorder by VendorID.

SET spark.databricks.delta.optimize.maxFileSize = 1600000000;

OPTIMIZE nyctaxi\_bloom

ZORDER BY VendorID

Here is the SQL count query which you will run as part of the performance test.

SELECT count(\*) FROM nyctaxi\_nonbloom WHERE tpep\_dropoff\_datetime = '2019-01-18 20:43:08'

The following Scala code spark.table("nyctaxi").schema.foreach(field => println(s"${field.name}: metadata=${field.metadata}")) which is also shown in Figure 15-14 will yield these results. This Scala code along with more details about the Bloom Filter Index and notebook samples can be found here within this Microsoft article: <https://docs.microsoft.com/en-us/azure/databricks/delta/optimizations/bloom-filters>