For more information on cluster access control, read: <https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/cluster-acl>

For more detail, read more about [Enabling workspace object access control](https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/workspace-acl) here: <https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/workspace-acl>

For more information on table access control, read more about [enabling table access control for your workspace](https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/table-acl) here: <https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/table-acl>.

For more information, read more about [managing Personal Access Tokens](https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/tokens) here: <https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/tokens>

1. Work Space Visibility Control: For more information, read more about how to [prevent users from seeing workspace objects they do not have access to](https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/workspace-acl#workspace-object-visibility) here: <https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/workspace-acl#workspace-object-visibility>
2. Cluster Visibility Control: For more information, read more about how to [prevent users from seeing clusters they do not have access to](https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/cluster-acl#cluster-visibility) here: <https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/cluster-acl#cluster-visibility>
3. Job Visibility Control: For more information, read more about how to [prevent users from seeing jobs they do not have access to](https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/jobs-acl#jobs-visibility) here: https://docs.microsoft.com/en-us/azure/databricks/administration-guide/access-control/jobs-acl#jobs-visibility

The following code will be executed in a Python Databricks Notebook and will extract the NYC Taxi Yellow Trip Data for 2019 into a data frame.

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)

The following code will add a few new columns to the data frame to include a new column for the year and a new column for the Vendor Name which is based on a case statement referencing the Vendor ID.

from pyspark.sql.functions import \*

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

nyctaxiDF = nyctaxiDF.withColumn("VendorName",

expr("case when VendorID = '1' then 'Vendor1' " +

"when VendorID = '2' then 'Vendor2' " +

"when VendorID = '4' then 'Vendor4' "

"else 'Unknown' end"))

The dataframe can be written to ADLS2 in delta format and partitioned by year with the following code.

(

nyctaxiDF

.write

.partitionBy("Year")

.format("delta")

.mode("overwrite")

.save("abfss://data@adl001.dfs.core.windows.net/raw/delta/nyctaxi\_delta")

)

You can create external/hive tables using the nyctaxi Delta location with the following code.

spark.sql("CREATE TABLE nyctaxi USING DELTA LOCATION 'abfss://data@adl001.dfs.core.windows.net/raw/delta/nyctaxi\_delta/'")

Here is the SQL query that you will need to run to re-create the results shown in Figure 19-18.

%sql

SELECT DISTINCT(VendorName) FROM nyctaxi ORDER BY VendorName ASC

The following SQL Query, shown in the code below, embeds the IS\_MEMBER function in the query to verify whether the current user is in the specified group.

%sql

SELECT

\*, IS\_MEMBER('Vendor1')

FROM nyctaxi

Similarly, you can also run a variation of the above query by changing the group member to Vendor2, as shown in the code below.

%sql

SELECT

\*, IS\_MEMBER('Vendor2')

FROM nyctaxi

This next query shown in the code below, adds VendorName to the IS\_MEMBER function and the results indicate either a true or false.

%sql

SELECT

\*, IS\_MEMBER(VendorName)

FROM nyctaxi

Next, add the same IS\_MEMBER (VendorName) function to the where clause of the following SQL Query, shown in the code below.

%sql

SELECT

\*

FROM nyctaxi

WHERE IS\_MEMBER(VendorName)

Here is the query that you will need to run to re-create the results shown in Figure 19-24.

%sql

SELECT

\*

FROM nyctaxi

WHERE IS\_MEMBER(VendorName)

After you have defined and tested your SQL query which uses the IS\_MEMBER function dynamically to apply the desired row level security filters for users logged into the workspace, you will need to create a view, much like the code shown below, to ensure that the row level security is fine-grained on a per user basis and then set up and manage appropriate administrative privileges such as ‘GRANT SELECT’ on the view while limiting direct access to the underlying source tables. With this approach, users will be able to access the appropriate row-level secured view by running a query as simple as Select \* from vendor\_nyctaxi to retrieve the row-level data that they have access to. It will also prevent them from being able to change filters to access data that they should not be accessing.

%sql

CREATE VIEW vendor\_nyctaxi AS

SELECT

\*

FROM nyctaxi

WHERE IS\_MEMBER(VendorName)

Proceed with granting select on the views to the respective vendors and this will ensure that only those users that have access to their respective groups will be able to see the corresponding secured row level vendor data.

%sql

GRANT SELECT ON VIEW vendor\_nyctaxi TO `Vendor1`;

GRANT SELECT ON VIEW vendor\_nyctaxi TO `Vendor2`;

GRANT SELECT ON VIEW vendor\_nyctaxi TO `Vendor3`;

GRANT SELECT ON VIEW vendor\_nyctaxi TO `Vendor4`;