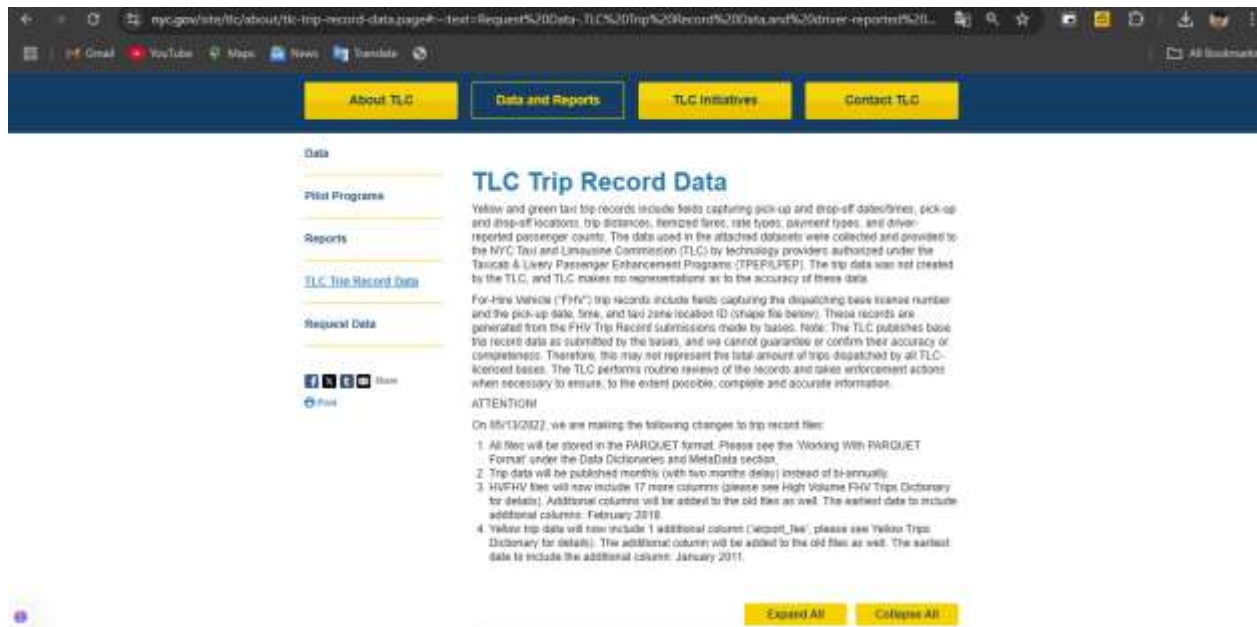


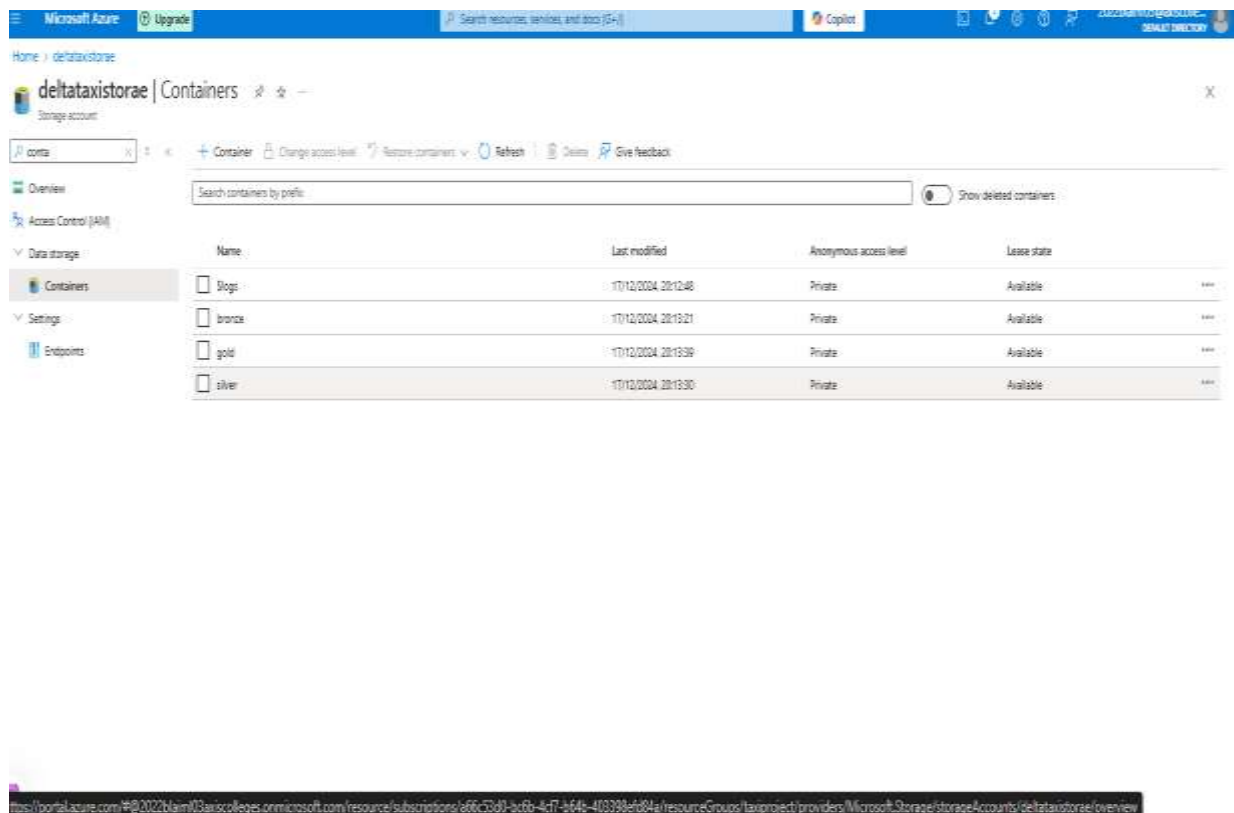
API SOURCE



The screenshot shows the NYC.gov website's "TLC Trip Record Data" page. The page has a blue header with navigation links: "About TLC", "Data and Reports", "TLC Initiatives", and "Contact TLC". On the left, a sidebar contains links for "Data", "Pilot Programs", "Reports", "TLC Trip Record Data" (highlighted), and "Request Data". The main content area is titled "TLC Trip Record Data" and includes a detailed description of the data, a list of updates, and a table of trip records. The updates section lists four changes: 1. All files will be stored in the PARQUET format. 2. Trip data will be published monthly (with two month delay) instead of bi-annually. 3. HVFHV files will now include 17 more columns. 4. Yellow trip data will now include 1 additional column. The table below shows the first four rows of trip records.

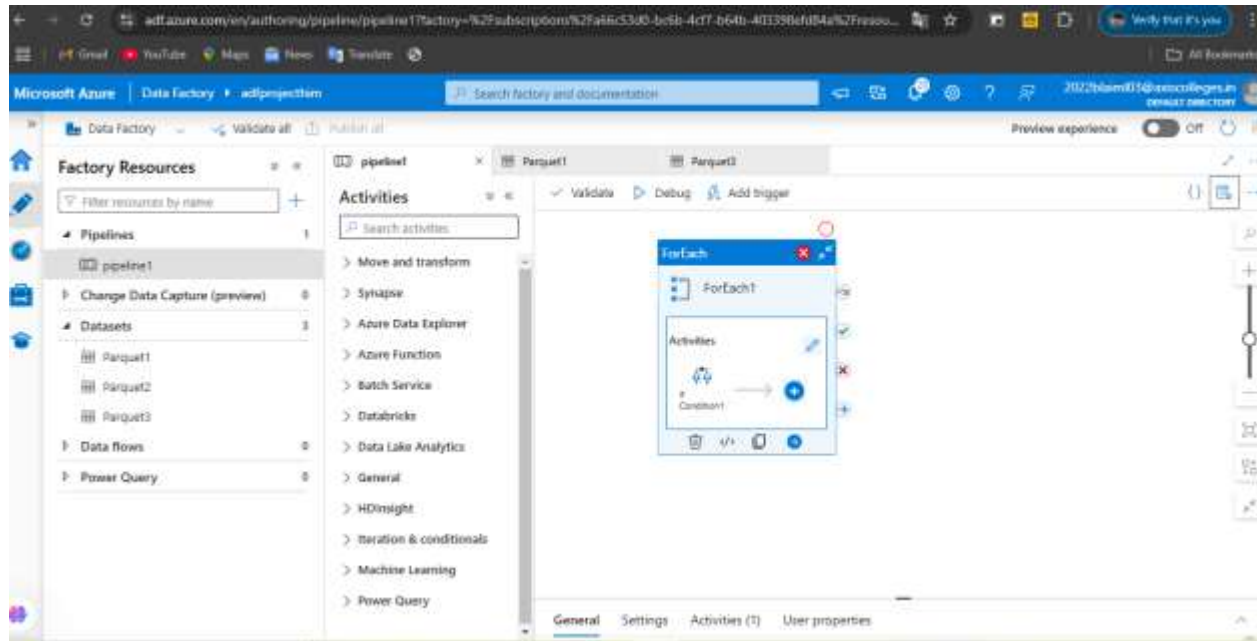
Name	Last modified	Anonymous access level	Lease state
<input type="checkbox"/> dogs	17/12/2024 20:12:48	Private	Available
<input type="checkbox"/> bronze	17/12/2024 20:19:21	Private	Available
<input type="checkbox"/> gold	17/12/2024 20:19:39	Private	Available
<input type="checkbox"/> silver	17/12/2024 20:19:30	Private	Available

Azure Data Lake Gen 2

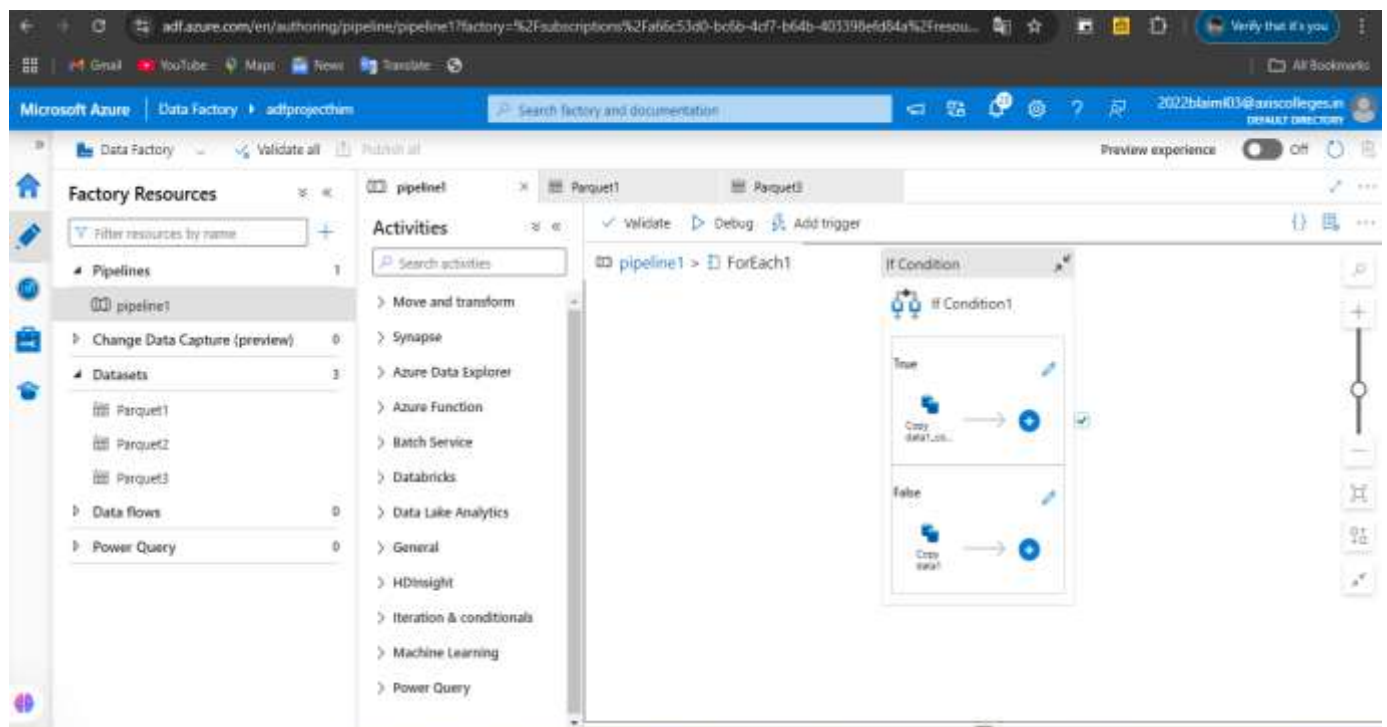


The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the "Microsoft Azure" logo, an "Upgrade" button, a search bar, and a "Copilot" button. The main content area is titled "deltatataxistoreae | Containers" and shows a list of containers. The containers are listed in a table with columns for Name, Last modified, Anonymous access level, and Lease state. The containers are "dogs", "bronze", "gold", and "silver". The "dogs" container is selected, and its details are shown in the right-hand pane. The URL at the bottom of the page is: <https://portal.azure.com/#@2022Name03axiscollages.commicrosoft.com/resource/subscriptions/66653d0-2c6b-4d7-464e-40339d684e/resourceGroups/taxiproject/providers/Microsoft.Storage/storageAccounts/deltatataxistoreae/overview>

AZURE DATA FACTORY PIPELINE(FOR EACH ACTIVITY)

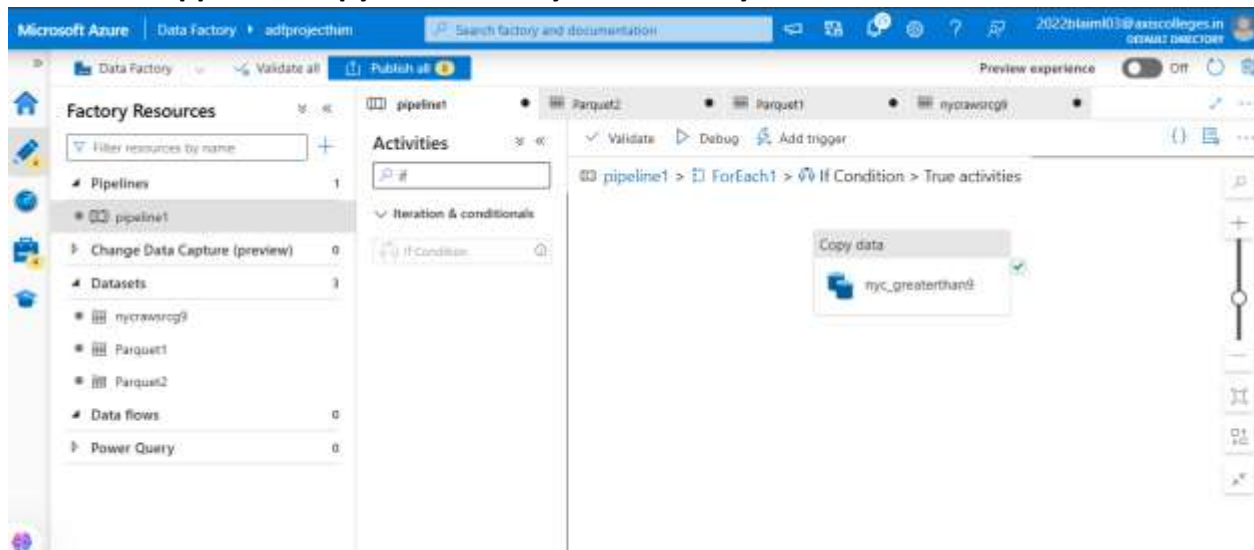


IF ELSE ACTIVITY IN FOR EACH

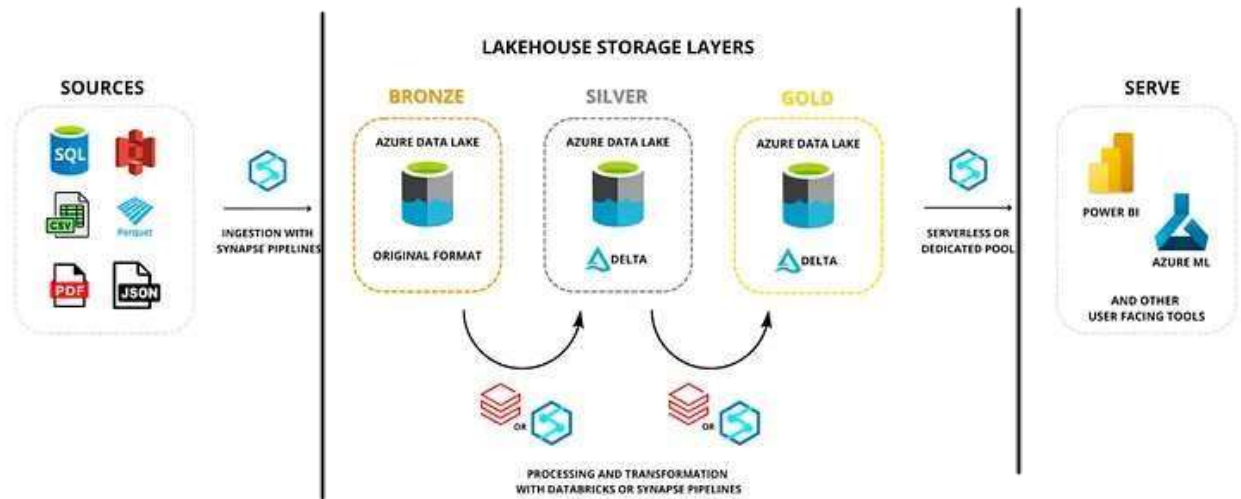


CODE - `trip-data/green_tripdata_2023-@{dataset()}.p_monthgreater}.parquet`

Implemented For Each Activity to iterate through monthly data (January to December), using If-Else Logic to direct data through conditional workflows, and in the "Else" condition, applied a Copy Data Activity to efficiently transfer and store the data.



CODE- trip-data/green_tripdata_2023-@{dataset().p_monthgreater}.parquet



#DATA ACCESS

```
SECRET_ID = "9.d8Q~VJku6GedTf74Rwr2hqQgg3RwsylxZ9aau."
APP_ID = "306b3f5c-f6ab-4ff7-9e1a-3d080ebb84f9" TENENT_ID
= "8122ffec-4fed-40ad-acb5-de53a17ae9d8"
```

```
spark.conf.set("fs.azure.account.auth.type.deltataxistora.dfs.core.wi
ndows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.deltataxistora.d
fs.core.windows.net",
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.deltataxistora.dfs.
core.windows.net", "306b3f5c-f6ab-4ff7-9e1a-3d080ebb84f9")
spark.conf.set("fs.azure.account.oauth2.client.secret.deltataxistora.
dfs.core.windows.net", "9.d8Q~VJku6GedTf74Rwr2hqQgg3RwsylxZ9aau.")
spark.conf.set("fs.azure.account.oauth2.client.endpoint.deltataxistora
e.dfs.core.windows.net",
"https://login.microsoftonline.com/8122ffec4fed-40ad-
acb5de53a17ae9d8/oauth2/token")
```

```
dbutils.fs.ls("abfss://bronze@deltataxistora.dfs.core.windows.net/")
[FileInfo(path='abfss://bronze@deltataxistora.dfs.core.windows.net/
2023/', name='2023/', size=0, modificationTime=1734713530000),
FileInfo(path='abfss://bronze@deltataxistora.dfs.core.windows.net/
trip_type/', name='trip_type/', size=0
modificationTime=1734633483000),
```

```
FileInfo(path='abfss://bronze@deltataxistorage.dfs.core.windows.net/trip_zone/', name='trip_zone/', size=0  
modificationTime=1734634179000)]
```

DATA READING

*Import Libraries

```
from pyspark.sql.functions import *  
from pyspark.sql.types import *
```

##Reading CSV DATA ###Trip

Type Data

```
df_trip_type = spark.read.format('csv')\  
    .option('header', True)\  
    .option('inferSchema', True)\  
    .load  
    ("abfss://bronze@deltataxistorage.dfs.core.windows.net/trip_type")  
df_trip_type.display()
```

Read Data From Azure Blob Storage Trip Zone

```
df_trip_zone = spark.read.format('csv')\  
    .option('header', True)\  
    .option('inferSchema', True)\  
    .load  
    ("abfss://bronze@deltataxistorage.dfs.core.windows.net/trip_zone")  
df_trip_zone.display()
```

#Trip Data Define Schema for the trip data

```
from pyspark.sql.types import StructType, StructField, LongType,  
StringType, DoubleType, TimestampType
```

```
my_schema = StructType([  
    StructField("VendorID", LongType(), True),  
    StructField("lpep_pickup_datetime", TimestampType(), True),  
    StructField("lpep_dropoff_datetime", TimestampType(), True),  
    StructField("store_and_fwd_flag", StringType(), True),  
    StructField("RatecodeID", LongType(), True),  
    StructField("PULocationID", LongType(), True),  
    StructField("DOLocationID", LongType(), True),  
    StructField("passenger_count", LongType(), True),  
    StructField("trip_distance", DoubleType(), True),  
    StructField("fare_amount", DoubleType(), True),  
    StructField("extra", DoubleType(), True),  
    StructField("mta_tax", DoubleType(), True),
```

```

    StructField("tip_amount", DoubleType(), True),
    StructField("tolls_amount", DoubleType(), True),
    StructField("ehail_fee", DoubleType(), True),
    StructField("improvement_surcharge", DoubleType(), True),
StructField("total_amount", DoubleType(), True),
    StructField("payment_type", LongType(), True),
    StructField("trip_type", LongType(), True),
    StructField("congestion_surcharge", DoubleType(), True)
])

df_trip = spark.read.format('parquet')\
    .option('header', 'true')\
    .schema(my_schema)\
    .option('recursiveFileLookup', 'true')\
    .load("abfss://bronze@deltataxistorage.dfs.core.windows.net/2023")
df_trip.printSchema()

```

#Data Transformation

Taxi Trip Type

```

df_trip_type.display()

df_trip_type = df_trip_type.withColumnRenamed("trip_type",
"trip_type_id") df_trip_type =
df_trip_type.withColumnRenamed( "description",
"trip_description") df_trip_type.display()

df_trip_type.write.format("parquet")\
    .mode("append")\
    .option("path"
,"abfss://silver@deltataxistorage.dfs.core.windows.net/trip_type")\
    .save()

```

Trip Zone

```
df_trip_zone.display()

df_trip_zone = df_trip_zone.withColumn('zone1',split(col('Zone'),"/")
[0])\
                                .withColumn('zone2',split(col('Zone'),"/")
[1])
df_trip_zone.display()

%python df_trip_zone.write\
    .format('parquet')\
    .mode('append')\
    .option("path",
"abfss://silver@deltataxistorage.dfs.core.windows.net/trip_zone")\
    .save()
```

Trip Data

```
%python df_trip =
df_trip.withColumn("trip_date",
to_date("lpep_pickup_datetime")) \
.withColumn("trip_year",
year("lpep_pickup_datetime")) \
    .withColumn("trip_month",
month("lpep_pickup_datetime")) display(df_trip)

from pyspark.sql.types import StructType, StructField, LongType,
StringType, DoubleType, TimestampType

my_schema = StructType([
    StructField("VendorID", LongType(), True),
    StructField("lpep_pickup_datetime", TimestampType(), True),
    StructField("lpep_dropoff_datetime", TimestampType(), True),
    StructField("store_and_fwd_flag", StringType(), True),
    StructField("RatecodeID", LongType(), True),
    StructField("PULocationID", LongType(), True),
    StructField("DOLocationID", LongType(), True),
    StructField("passenger_count", LongType(), True),
    StructField("trip_distance", DoubleType(), True),
    StructField("fare_amount", DoubleType(), True),
    StructField("extra", DoubleType(), True),
    StructField("mta_tax", DoubleType(), True),
    StructField("tip_amount", DoubleType(), True),
    StructField("tolls_amount", DoubleType(), True),
    StructField("ehail_fee", DoubleType(), True),
    StructField("improvement_surcharge", DoubleType(), True),
    StructField("total_amount", DoubleType(), True),
    StructField("payment_type", LongType(), True),
```



```

        StructField("trip_type", LongType(), True),
        StructField("congestion_surcharge", DoubleType(), True)
    ]) df_tripa = df_trip.select('VendorID' , 'PULocationID' ,
                                'total_amount',
                                'DOLocationID', 'trip_date', 'trip_distance')

```

#Save the Parquet File into Silver Layer

```

df_tripa.write\
    .format("parquet") \
    .mode('overwrite') \
    .option("path",
"abfss://silver@deltataxistorage.dfs.core.windows.net/trip2023data") \
    .save()

```

storage variable

```

silver = "abfss://silver@deltataxistorage.dfs.core.windows.net" gold
= "abfss://gold@deltataxistorage.dfs.core.windows.net"

```

##Data Reading and Writing and Creating Delta Tables

```
%sql
SHOW EXTERNAL LOCATIONS;

spark.conf.set("fs.azure.account.auth.type.deltataxistorae.dfs.core.wi
ndows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.deltataxistorae.d
fs.core.windows.net",
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.deltataxistorae.dfs.
core.windows.net", "306b3f5c-f6ab-4ff7-9e1a-3d080ebb84f9")
spark.conf.set("fs.azure.account.oauth2.client.secret.deltataxistorae.
dfs.core.windows.net", "9.d8Q~VJku6GedTf74Rwr2hqQgg3RwsylxZ9aau.")
spark.conf.set("fs.azure.account.oauth2.client.endpoint.deltataxistora
e.dfs.core.windows.net",
"https://login.microsoftonline.com/8122ffec4fed-40ad-acb5-
de53a17ae9d8/oauth2/token")

from pyspark.sql.functions import *
from pyspark.sql.types import *

silver = "abfss://silver@deltataxistorae.dfs.core.windows.net"
gold = "abfss://gold@deltataxistorae.dfs.core.windows.net"
```

#Data Zone

```
df_zone = spark.read.format("parquet")\
.option('inferSchema', True)\
.option('header' , True)\
.load(f'{silver}/trip_zone') df_zone.display()

%sql create database gold

df_zone = df_zone.write.format('delta')\
.mode('append')\
.option('path',f'{gold}/trip_zone')\
.saveAsTable('gold.trip_zone')

%sql select*from gold.trip_zone;
```

trip_type

```
df_type = spark.read.format("parquet")\
.option('inferSchema', True)\
.option('header' , True)\
.load(f'{silver}/trip_type')
df_type.display()
```

```
df_type = df_type.dropDuplicates()
df_type.display()
```

write the data into gold layer

```
df_type = df_type.write.format('delta')\
                        .mode('append')\
                        .option('path',f'{gold}/trip_type')\
                        .saveAsTable('gold.trip_type')
```

trip2023data

```
df_type2023 = spark.read.format("parquet")\
                    .option('inferSchema', True)\
                    .option('header' , True)\
                    .load(f'{silver}/trip2023data') df_type2023.display()
```

save the data into Gold layer

```
df_type2023 = df_type2023.write.format('delta')\
                        .mode('append')\
                        .option('path',f'{gold}/trip_type2023')\
                        .saveAsTable('gold.trip2023')
```

```
%sql
select*from gold.trip2023 ;
```

CONNECT TO POWER BI FOR VISUALIZATION

