NYC Taxi Dataset Analysis using PySpark & Databricks

Project Overview

This project analyzes the NYC Yellow Taxi Trip Data (January 2020) using PySpark in Databricks. The dataset is loaded into Databricks File System (DBFS), transformed using DataFrames, and queried to extract key insights such as revenue, vendor performance, passenger counts, and popular routes. The final output is optionally saved as Parquet or Delta table.

Step 1: Download Dataset

```
wget https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2020-01.csv
```

Step 2: Upload Dataset to DBFS

```
dbutils.fs.cp("file:/local/path/yellow_tripdata_2020-01.csv", "dbfs:/FileStore/taxi/")
```

Step 3: Read CSV into DataFrame

```
from pyspark.sql import SparkSession
df = spark.read.csv("/FileStore/taxi/yellow_tripdata_2020-01.csv", header=True, inferSchema=True)
df.cache()
df.show(5)
```

Query 1: Add Revenue Column

```
from pyspark.sql.functions import col

df = df.withColumn("Revenue",
    col("fare_amount") + col("extra") + col("mta_tax") +
    col("improvement_surcharge") + col("tip_amount") +
    col("tolls_amount") + col("total_amount"))
df.select("Revenue").show(5)
```

Query 2: Passenger Count by Area

```
df.groupBy("PULocationID").sum("passenger_count").withColumnRenamed("sum(passenger_count)",
"total_passengers").show()
```

Query 3: Avg Fare by Vendor

```
df.groupBy("VendorID").agg({"fare_amount":"avg", "total_amount":"avg"}).show()
```

Query 4: Moving Count by Payment Mode

```
from pyspark.sql.functions import window

df_time = df.withColumn("tpep_pickup_datetime", col("tpep_pickup_datetime").cast("timestamp"))
```

```
df_time.groupBy(window("tpep_pickup_datetime",                    "5                    minutes"),
"payment_type").count().orderBy("window").show()
```

Query 5: Top Gaining Vendors

```
from pyspark.sql.functions import to_date, sum

df_date = df.withColumn("trip_date", to_date("tpep_pickup_datetime"))
df_date.groupBy("trip_date", "VendorID")    .agg(sum("total_amount").alias("total_revenue"),
         sum("passenger_count").alias("total_passengers"),
                       sum("trip_distance").alias("total_distance"))          .orderBy("trip_date",
col("total_revenue").desc()).show(2)
```

Query 6: Route with Most Passengers

```
df.groupBy("PULocationID",    "DOLocationID")                 .sum("passenger_count")
.orderBy(col("sum(passenger_count)").desc()).show(1)
```

Query 7: Top Pickup in Last 10 Sec

```
from pyspark.sql.functions import max as max_

latest_time = df.agg(max_("tpep_pickup_datetime")).collect()[0][0]
df.filter(col("tpep_pickup_datetime")  >=  latest_time  -  expr("INTERVAL  10  SECONDS"))
.groupBy("PULocationID")                             .sum("passenger_count")
.orderBy(col("sum(passenger_count)").desc()).show()
```

Optional: Save as Parquet Table

```
df.write.mode("overwrite").parquet("/mnt/datalake/nyc_taxi_parquet")
spark.sql("""
    CREATE TABLE IF NOT EXISTS nyc_taxi_external
    USING parquet
    LOCATION '/mnt/datalake/nyc_taxi_parquet'
""")
```

Optional: Save as Delta Table

```
df.write.format("delta").mode("overwrite").save("/delta/nyc_taxi")
```