TASK 1 : BIG DATA ANALYSIS

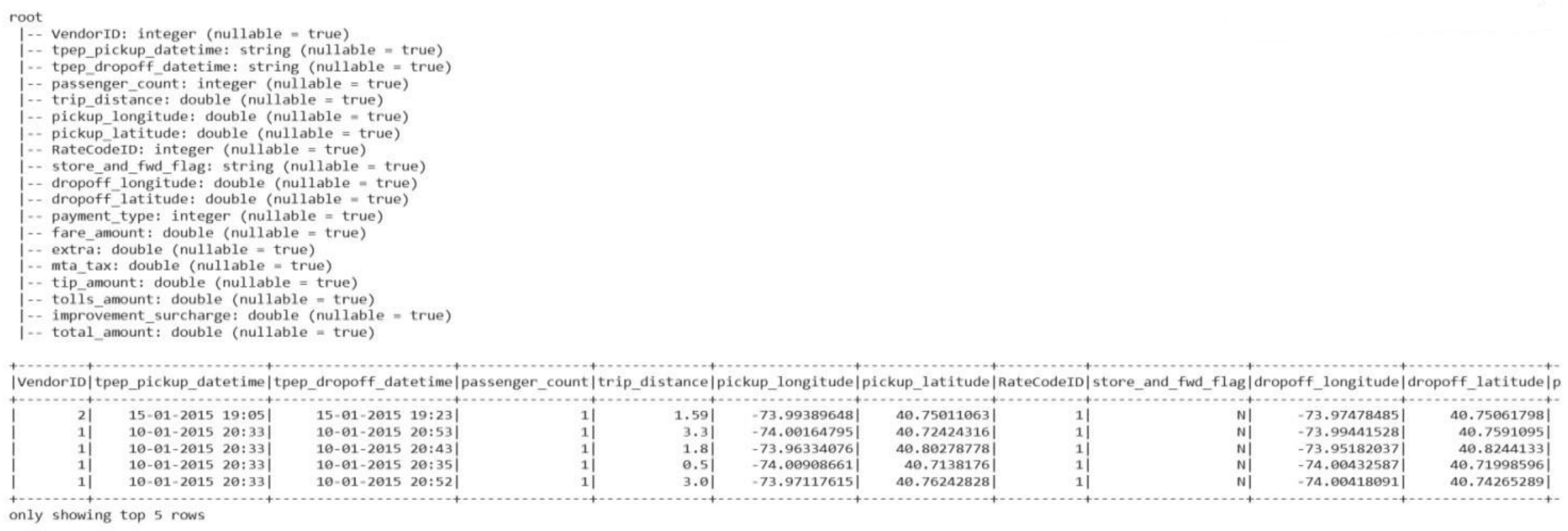
```
pip install pyspark
```

OUTPUT:

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.11/dist-packages (3.5.1)

Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.11/dist-packages (from pyspark) (0.10.9.7)
```

```
# NYC Yellow Taxi Trip Data - Big Data Analysis using PySpark

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, hour, dayofweek, avg, count, sum as spark_sum, max
as spark_max, min as spark_min




# Step 1: Start Spark session

spark = SparkSession.builder.appName("NYC Yellow Taxi Analysis").getOrCreate()



# Step 2: Load the dataset

file_path = "/content/NYC Yellow Taxi Trip Data.csv"

df = spark.read.csv(file_path, header=True, inferSchema=True)



# Step 3: Inspect schema and data

df.printSchema()

df.show(5)
```

OUTPUT:



```
root
 |-- VendorID: integer (nullable = true)
 |-- tpep_pickup_datetime: string (nullable = true)
 |-- tpep_dropoff_datetime: string (nullable = true)
 |-- passenger_count: integer (nullable = true)
 |-- trip_distance: double (nullable = true)
 |-- pickup_longitude: double (nullable = true)
 |-- pickup_latitude: double (nullable = true)
 |-- RateCodeID: integer (nullable = true)
 |-- store_and_fwd_flag: string (nullable = true)
 |-- dropoff_longitude: double (nullable = true)
 |-- dropoff_latitude: double (nullable = true)
 |-- payment_type: integer (nullable = true)
 |-- fare_amount: double (nullable = true)
 |-- extra: double (nullable = true)
 |-- mta_tax: double (nullable = true)
 |-- tip_amount: double (nullable = true)
 |-- tolls_amount: double (nullable = true)
 |-- improvement_surcharge: double (nullable = true)
 |-- total_amount: double (nullable = true)

+--------+--------------------+---------------------+---------------+-------------+----------------+---------------+----------+------------------+-----------------+----------------+-
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distance|pickup_longitude|pickup_latitude|RateCodeID|store_and_fwd_flag|dropoff_longitude|dropoff_latitude|p
+--------+--------------------+---------------------+---------------+-------------+----------------+---------------+----------+------------------+-----------------+----------------+-
|       2|    15-01-2015 19:05|     15-01-2015 19:23|              1|         1.59|    -73.99389648|    40.75011063|         1|                 N|     -73.97478485|     40.75061798|
|       1|    10-01-2015 20:33|     10-01-2015 20:53|              1|          3.3|    -74.00164795|    40.72424316|         1|                 N|     -73.99441528|      40.7591095|
|       1|    10-01-2015 20:33|     10-01-2015 20:43|              1|          1.8|    -73.96334076|    40.80278778|         1|                 N|     -73.95182037|      40.8244133|
|       1|    10-01-2015 20:33|     10-01-2015 20:35|              1|          0.5|    -74.00908661|     40.7138176|         1|                 N|     -74.00432587|     40.71998596|
|       1|    10-01-2015 20:33|     10-01-2015 20:52|              1|          3.0|    -73.97117615|    40.76242828|         1|                 N|     -74.00418091|     40.74265289|
+--------+--------------------+---------------------+---------------+-------------+----------------+---------------+----------+------------------+-----------------+----------------+-
only showing top 5 rows
```

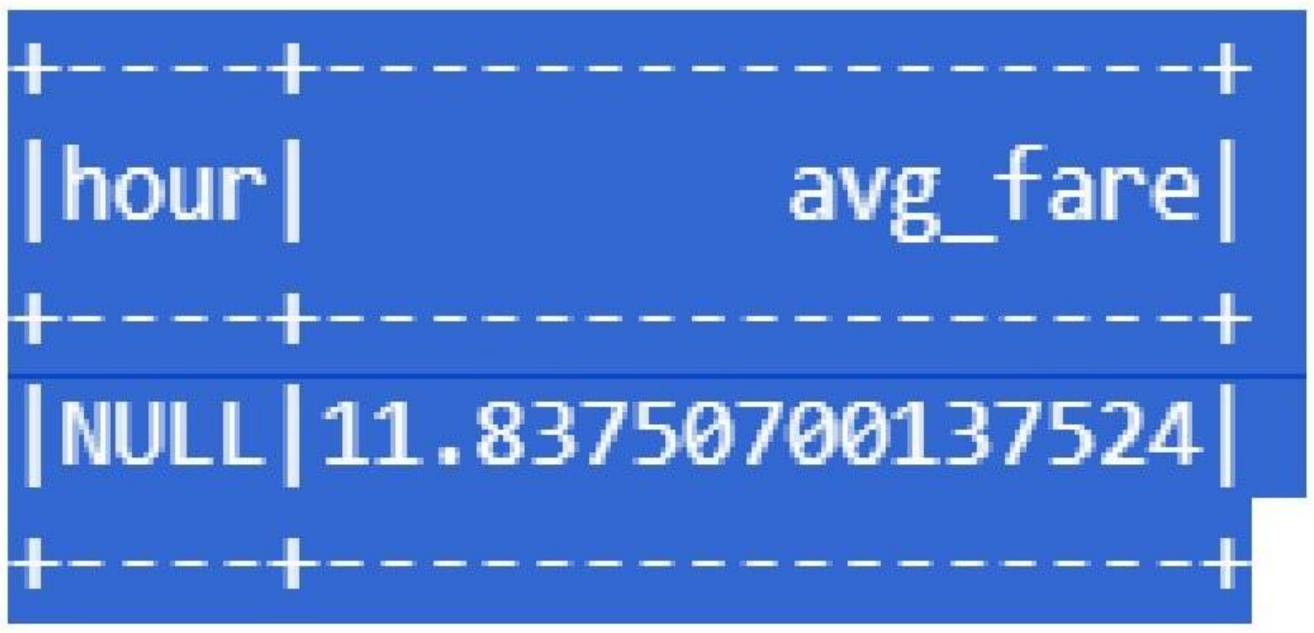
# Step 4: Data Cleaning - Remove nulls and invalid values

```
df_clean = df.dropna().filter(
    (col("passenger_count") > 0) &
    (col("trip_distance") > 0) &
    (col("fare_amount") > 0)
)
```

# Step 6: Analysis 1 - Avg Fare by Hour

```
avg_fare_by_hour =
df_clean.groupBy("hour").agg(avg("fare_amount").alias("avg_fare")).orderBy("hour")
avg_fare_by_hour.show()
```

OUTPUT:

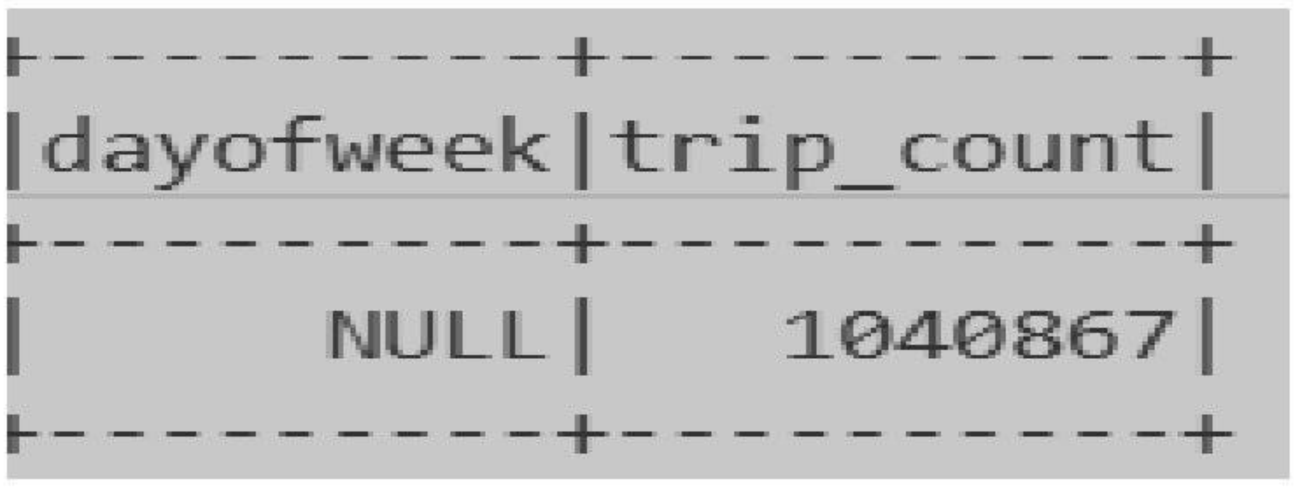


```
+----+-----------------+
|hour|         avg_fare|
+----+-----------------+
|NULL|11.83750700137524|
+----+-----------------+
```

# Step 7: Analysis 2 - Total Trips by Day of Week

```
trips_by_day =
df_clean.groupBy("dayofweek").agg(count("*").alias("trip_count")).orderBy("dayofweek")

trips_by_day.show()
```

OUTPUT:

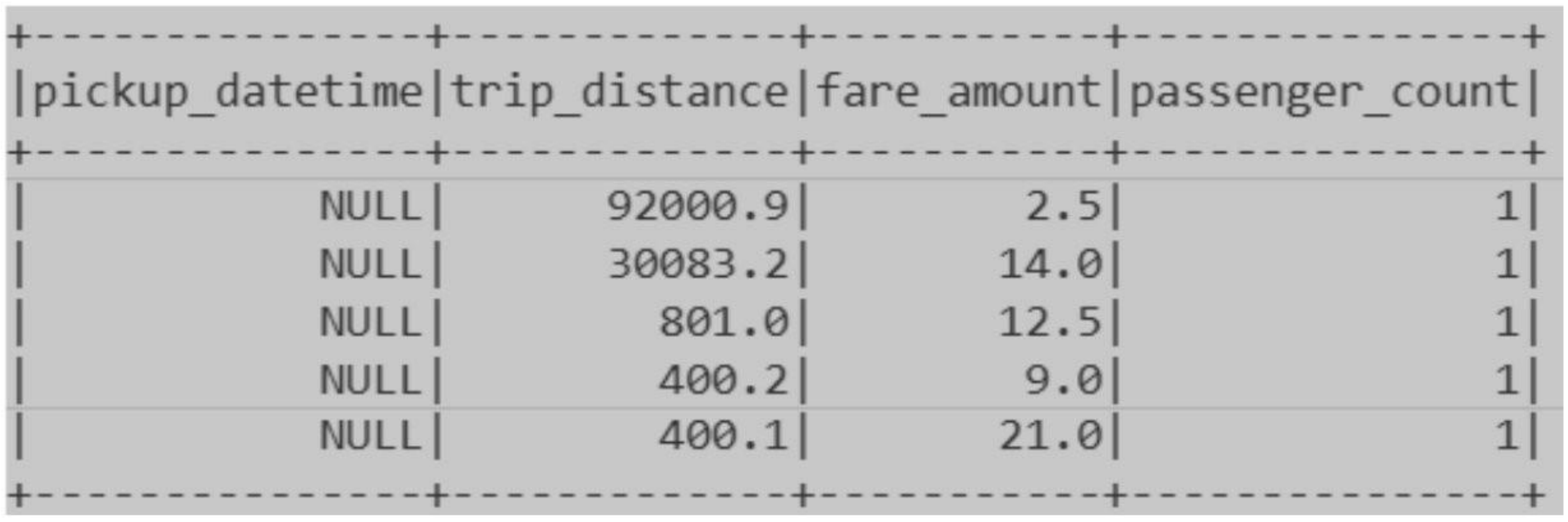


```
+---------+----------+
|dayofweek|trip_count|
+---------+----------+
|     NULL|   1040867|
+---------+----------+
```

# Step 8: Analysis 3 - Top 5 longest trips

```
longest_trips = df_clean.orderBy(col("trip_distance").desc()).select(
   "pickup_datetime", "trip_distance", "fare_amount", "passenger_count"
).limit(5)
longest_trips.show()
```

OUTPUT:

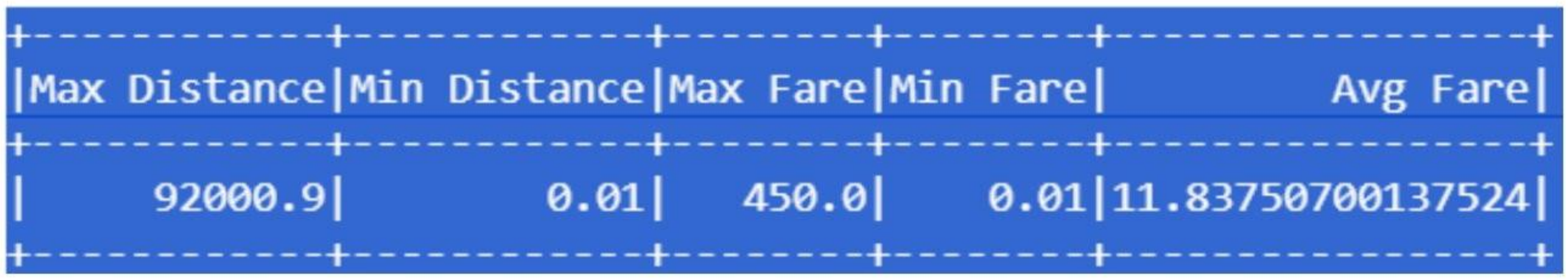


```
+---------------+-------------+-----------+---------------+
|pickup_datetime|trip_distance|fare_amount|passenger_count|
+---------------+-------------+-----------+---------------+
|           NULL|      92000.9|        2.5|              1|
|           NULL|      30083.2|       14.0|              1|
|           NULL|        801.0|       12.5|              1|
|           NULL|        400.2|        9.0|              1|
|           NULL|        400.1|       21.0|              1|
+---------------+-------------+-----------+---------------+
```

# Step 9: Summary Stats

```
summary = df_clean.select(
   spark_max("trip_distance").alias("Max Distance"),
   spark_min("trip_distance").alias("Min Distance"),
    spark_max("fare_amount").alias("Max Fare"),

    spark_min("fare_amount").alias("Min Fare"),

    avg("fare_amount").alias("Avg Fare")

)

summary.show()
```

OUTPUT:



```
+------------+------------+--------+--------+-----------------+
|Max Distance|Min Distance|Max Fare|Min Fare|         Avg Fare|
+------------+------------+--------+--------+-----------------+
|     92000.9|        0.01|   450.0|    0.01|11.83750700137524|
+------------+------------+--------+--------+-----------------+
```

# Stop Spark session

```
spark.stop()
```