WILFRID LAURIER UNIVERSITY

LAURIER

Inspiring Lives.

CP 422 – Programming for Big Data

Fall 2024

Group 23

Assignment #2

Member 1 Name: Nicholas Cai	ID: 210555500
Member 2 Name: Isha Pabla	ID: 190296510
Member 3 Name: Propa Roy	ID: 190453380
Member 4 Name: Saihaan Baig	ID: 211725230
Member 5 Name: Rajan Dosanjh	ID: 200829900
Member 6 Name: Akif Rahman	ID: 210290530

Submission Date: November 11, 2024

All listed members have contributed, read, and approved this submission. All listed members have acknowledged each team member's equal and reasonable contribution to this submission.

Member 1 Name: Nicholas Cai	Signature: NC
Member 2 Name: Isha Pabla	Signature: IP
Member 3 Name: Propa Roy	Signature: PR
Member 4 Name: Saihaan Baig	Signature: SB
Member 5 Name: Rajan Dosanjh	Signature: RD
Member 6 Name: Akif Rahman	Signature: AR

Group-ID-23-CP422-A2.pdf (6)

November 11, 2024

#CP422 Assignment 2

0.1 Loading Data

[ ]:
```
from pyspark.sql import SparkSession

# Initialize SparkSession
spark = SparkSession.builder \
    .appName("TaxiDataAnalysis") \
    .getOrCreate()
# Read the CSV file into a Spark DataFrame
df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("dbfs:/FileStore/tables/yellow_tripdata_2015_01.csv")

# Create a temporary view for SQL queries
df.createOrReplaceTempView("yellow_taxi_data")
```

0.2 Q1: Outlier Detection

0.2.1 Task 1: Write a SQL query to find trips with fare amounts over $1000.

[ ]:
```
# SQL query to find trips with fare amounts over $1000
query = """
SELECT *
FROM yellow_taxi_data
WHERE fare_amount > 1000
"""

# Run the query
result_df = spark.sql(query)

# Show the results
result_df.show()
```

```
[Stage 136:===================================================>      (9 + 1) / 10]
```

```
+--------+-------------------+--------------------+---------------+-----------
--+------------------+------------------+----------+------------------+---------
---------+------------------+------------+----------+-----+-------+----------+-
-----------+--------------------+------------+
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distan
ce|  pickup_longitude|   pickup_latitude|RateCodeID|store_and_fwd_flag|
dropoff_longitude|  dropoff_latitude|payment_type|fare_amount|extra|mta_tax|tip_
amount|tolls_amount|improvement_surcharge|total_amount|
+--------+-------------------+--------------------+---------------+-----------
--+------------------+------------------+----------+------------------+---------
---------+------------------+------------+----------+-----+-------+----------+-
-----------+--------------------+------------+
|       2| 2015-01-02 20:06:34|  2015-01-02 20:23:33|              1|
0.4|-74.01433563232422|40.711856842041016|         1|
N|-73.98519134521484| 40.76046371459961|           2|     3005.5| 0.05|    0.5|
0.0|         0.0|                  0.3|     3006.35|
|       1| 2015-01-22 21:12:26|  2015-01-22 21:20:36|              1|
1.7|-73.96153259277344| 40.77063751220703|         1|
N|-73.97850799560547|40.749515533447266|           2|     4008.0|  0.5|    0.5|
0.0|         0.0|                  0.3|      4009.3|
+--------+-------------------+--------------------+---------------+-----------
--+------------------+------------------+----------+------------------+---------
---------+------------------+------------+----------+-----+-------+----------+-
-----------+--------------------+------------+
```

0.2.2 Task 2: Write another query to find trips with zero or negative fare amounts.

[ ]:
```python
# SQL query to find trips with zero or negative fare amounts
query = """
SELECT *
FROM yellow_taxi_data
WHERE fare_amount <= 0
"""

# Run the query
result_df = spark.sql(query)

# Show the results
result_df.show()
```

```
+--------+-------------------+--------------------+---------------+-----------
--+------------------+------------------+----------+------------------+---------
---------+------------------+------------+----------+-----+-------+----------+-
-----------+--------------------+------------+
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distan
```

```
ce|  pickup_longitude|   pickup_latitude|RateCodeID|store_and_fwd_flag|
dropoff_longitude|  dropoff_latitude|payment_type|fare_amount|extra|mta_tax|tip_
amount|tolls_amount|improvement_surcharge|total_amount|
+--------+--------------------+---------------------+---------------+-----------
--+------------------+------------------+----------+------------------+---------
---------+------------------+------------+-----------+-----+-------+----------+-
-----------+---------------------+------------+
|       1| 2015-01-28 20:22:19|  2015-01-28 20:23:19|              2|
4.8|-74.03569030761719|40.743648529052734|         5|
N|-74.03571319580078| 40.74365997314453|           3|        0.0|  0.0|    0.0|
0.0|         0.0|                   0.3|         0.3|
|       2| 2015-01-17 22:40:27|  2015-01-17 22:43:04|              1|
0.11|-74.00235748291016| 40.73982620239258|         1|
N|-74.00111389160156| 40.74110794067383|           4|       -3.5| -0.5|   -0.5|
0.0|         0.0|                   0.3|        -4.8|
|       2| 2015-01-15 17:33:24|  2015-01-15 17:33:31|              2|
0.0| -73.9825668334961| 40.73979949951172|         1|                 N|
-73.9825668334961| 40.73979949951172|           3|       -2.5| -1.0|   -0.5|
-0.7|         0.0|                   0.3|        -5.0|
|       1| 2015-01-21 10:16:35|  2015-01-21 10:16:54|              1|
0.0| -73.9929428100586| 40.76789855957031|         5|                 N|
-73.9929428100586|40.767887115478516|           1|        0.0|  0.0|    0.0|
11.0|         0.0|                   0.3|        11.3|
|       2| 2015-01-06 12:43:31|  2015-01-06 12:46:07|              5|
0.23| -73.9603271484375| 40.76001739501953|         2|
N|-73.96344757080078| 40.76166534423828|           2|        0.0|  0.0|    0.5|
0.0|         0.0|                   0.3|         0.0|
|       1| 2015-01-23 23:57:43|  2015-01-24 00:35:26|              2|
13.4|-73.97903442382812|  40.7663688659668|         5|
N|-74.15727996826172| 40.73886489868164|           4|        0.0|  0.0|    0.0|
0.0|         0.0|                   0.3|         0.3|
|       2| 2015-01-16 16:00:45|  2015-01-16 16:00:53|              1|
0.0| -73.9377212524414| 40.75819396972656|         1|                 N|
-73.9377212524414| 40.75819396972656|           3|       -2.5| -1.0|   -0.5|
0.0|         0.0|                   0.3|        -4.3|
|       1| 2015-01-08 22:26:34|  2015-01-08 22:26:34|              1|
0.0|               0.0|               0.0|         5|                 Y|
0.0|               0.0|           2|        0.0|  0.0|    0.0|        0.0|
0.0|                   0.3|         0.3|
|       2| 2015-01-31 23:38:52|  2015-01-31 23:38:54|              2|
0.0|               0.0|               0.0|         2|                 N|
0.0|               0.0|           2|      -52.0|  0.0|   -0.5|        0.0|
0.0|                   0.3|       -52.8|
|       1| 2015-01-22 09:32:57|  2015-01-22 09:54:39|              3|
2.4|-73.95448303222656| 40.74155807495117|         5|
N|-73.99308013916016| 40.74636459350586|           1|        0.0|  0.0|    0.0|
10.0|         0.0|                   0.3|        10.3|
|       1| 2015-01-05 02:27:56|  2015-01-05 02:32:51|              2|
```

```
2.9|              0.0|                0.0|         5|
N|-73.97557067871094| 40.74790573120117|           3|         0.0|  0.0|    0.0|
0.0|       5.33|                  0.3|       5.63|
|       2| 2015-01-10 02:23:53|  2015-01-10 02:23:58|              2|
0.0|              0.0|                0.0|         5|                 N|
0.0|              0.0|           1|        -6.8|  0.0|    0.0|      -1.0|
0.0|                 0.3|       -8.1|
|       2| 2015-01-13 08:45:10|  2015-01-13 08:46:32|              5|
0.0|-73.90199279785156| 40.76407241821289|         1|
N|-73.90202331542969|40.764068603515625|           2|         0.0|  0.0|    0.0|
0.0|        0.0|                  0.3|        0.0|
|       2| 2015-01-14 11:52:09|  2015-01-14 11:52:20|              1|
0.0|-73.78995513916016| 40.64694595336914|         2|                 N|
0.0|              0.0|           3|       -52.0|  0.0|   -0.5|    -14.33|
-5.33|                 0.3|      -72.46|
|       2| 2015-01-03 02:01:25|  2015-01-03 02:01:54|              1|
0.03|-73.95340728759766| 40.81114959716797|         1|
N|-73.95375061035156|40.811302185058594|           2|        -2.5| -0.5|   -0.5|
0.0|        0.0|                  0.3|       -3.8|
|       2| 2015-01-20 20:44:47|  2015-01-20 20:44:48|              1|
0.0|              0.0|                0.0|         1|                 N|
-73.9373779296875|40.758209228515625|           1|         0.0|  0.0|    0.0|
0.0|        0.0|                  0.3|        0.0|
|       2| 2015-01-17 11:12:35|  2015-01-17 11:14:52|              1|
0.0|-73.93765258789062|40.758121490478516|         1|
N|-73.93766021728516| 40.75809097290039|           1|         0.0|  0.0|    0.0|
0.0|        0.0|                  0.3|        0.0|
|       2| 2015-01-12 15:07:29|  2015-01-12 15:07:35|              1|
0.0|              0.0|                0.0|         2|                 N|
0.0|              0.0|           2|       -52.0|  0.0|   -0.5|       0.0|
0.0|                 0.3|      -52.8|
|       2| 2015-01-06 14:07:25|  2015-01-06 14:08:27|              1|
0.03|-73.99456024169922|40.740318298339844|         1|
N|-73.99533081054688| 40.74095153808594|           4|        -2.5|  0.0|   -0.5|
0.0|        0.0|                  0.3|       -3.3|
|       2| 2015-01-10 21:10:20|  2015-01-10 21:12:39|              1|
0.03|     -73.986328125|40.755279541015625|         1|
N|-73.98542022705078|40.755088806152344|           4|        -3.5| -0.5|   -0.5|
0.0|        0.0|                  0.3|       -4.8|
+--------+--------------------+---------------------+---------------+----------
--+------------------+-------------------+----------+------------------+--------
---------+------------------+------------+-----------+-----+-------+----------+-
-----------+---------------------+------------+
only showing top 20 rows
```

0.3 Q2: Correlation Analysis

###Task 1.1: Write SQL queries to calculate the correlation between for fare_amount and trip_distance.

[ ]:
```
query = """
SELECT corr(fare_amount, trip_distance) AS fare_trip_distance_corr
FROM yellow_taxi_data
"""

# Run the query
result_df = spark.sql(query)

# Show the results
result_df.show()
```

```
[Stage 138:=============================>                          (8 + 7) / 15]

+-----------------------+
|fare_trip_distance_corr|
+-----------------------+
|   4.422117955836895E-4|
+-----------------------+
```

0.3.1 Task 1.2: Write SQL queries to calculate the correlation between total_amount and trip_distance.

[ ]:
```
query="""
SELECT corr(total_amount, trip_distance) AS total_trip_distance_corr
FROM yellow_taxi_data
"""
# Run the query
result_df = spark.sql(query)

# Show the results
result_df.show()
```

```
[Stage 141:=============================>                          (8 + 7) / 15]

+------------------------+
|total_trip_distance_corr|
+------------------------+
|    3.339064563073854…|
+------------------------+
```

0.4 Task 2: Analysis of Correlations

Correlation between fare_amount and trip_distance: - A positive correlation coefficient close to 1 indicates a strong positive linear relationship, suggesting that as the trip distance increases, the fare amount also increases proportionally.

Correlation between total_amount and trip_distance: - This correlation measures how the total amount (including fare and additional charges) relates to trip distance. A strong positive correlation implies that longer trips tend to have higher total charges.

0.5 Q3: Trip Duration Prediction

0.5.1 Task 1: Calculate the trip duration in minutes for each trip.

[ ]:
```
from pyspark.sql.functions import unix_timestamp

# Add a new column 'trip_duration_minutes' to the DataFrame
df_with_duration = df.withColumn(
    "trip_duration_minutes",
    (unix_timestamp("tpep_dropoff_datetime") -␣
 ↪unix_timestamp("tpep_pickup_datetime")) / 60
)

# Create a temporary view with the new column
df_with_duration.createOrReplaceTempView("yellow_taxi_with_duration")
```

0.5.2 Task 2: Write a SQL query to find the average trip duration for trips with different passenger counts.

[ ]:
```
query = """
SELECT passenger_count, AVG(trip_duration_minutes) AS average_trip_duration
FROM yellow_taxi_with_duration
GROUP BY passenger_count
ORDER BY passenger_count
"""

# Run the query
result_df = spark.sql(query)

# Show the results
result_df.show()
```

```
[Stage 144:=================================>                    (9 + 6) / 15]

+---------------+---------------------+
|passenger_count|average_trip_duration|
+---------------+---------------------+
|              0|    12.558855039350089|
|              1|    14.246676076779718|
|              2|    13.842955421433144|
```

```
|              3|   14.025823263687862|
|              4|   13.840397849632195|
|              5|   14.517415328235217|
|              6|   14.051230948944927|
|              7|    8.837037037037037|
|              8|    5.345000000000001|
|              9|   15.793939393939395|
+---------------+---------------------+
```

0.6 Q4: Trip Clustering

0.7 Task 1: Use SQL to categorize trips into distance bins (e.g., <1 mile, 1-2 miles, 2-5 miles, >5 miles).

[ ]:
```
query="""
-- SQL query to categorize trips into distance bins
SELECT *,
       CASE
           WHEN trip_distance < 1 THEN '<1 mile'
           WHEN trip_distance >= 1 AND trip_distance < 2 THEN '1-2 miles'
           WHEN trip_distance >= 2 AND trip_distance <= 5 THEN '2-5 miles'
           ELSE '>5 miles'
       END AS distance_bin
FROM yellow_taxi_data
"""

# Run the query
result_df = spark.sql(query)

# Show the results
result_df.show()
```

```
+--------+--------------------+---------------------+---------------+-----------
--+------------------+------------------+----------+------------------+---------
---------+------------------+------------+----------+-----+-------+----------+-
-----------+---------------------+------------+------------+
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distan
ce|  pickup_longitude|   pickup_latitude|RateCodeID|store_and_fwd_flag|
dropoff_longitude|  dropoff_latitude|payment_type|fare_amount|extra|mta_tax|tip_
amount|tolls_amount|improvement_surcharge|total_amount|distance_bin|
+--------+--------------------+---------------------+---------------+-----------
--+------------------+------------------+----------+------------------+---------
---------+------------------+------------+----------+-----+-------+----------+-
-----------+---------------------+------------+------------+
|       2| 2015-01-15 19:05:39|  2015-01-15 19:23:42|              1|
1.59|  -73.993896484375|  40.7501106262207|         1|
```

```
N|-73.97478485107422| 40.75061798095703|           1|        12.0|  1.0|    0.5|
3.25|         0.0|                   0.3|       17.05|   1-2 miles|
|       1| 2015-01-10 20:33:38|  2015-01-10 20:53:28|               1|
3.3|-74.00164794921875|  40.7242431640625|          1|
N|-73.99441528320312| 40.75910949707031|           1|        14.5|  0.5|    0.5|
2.0|         0.0|                   0.3|        17.8|   2-5 miles|
|       1| 2015-01-10 20:33:38|  2015-01-10 20:43:41|               1|
1.8|-73.96334075927734| 40.80278778076172|          1|
N|-73.95182037353516| 40.82441329956055|           2|         9.5|  0.5|    0.5|
0.0|         0.0|                   0.3|        10.8|   1-2 miles|
|       1| 2015-01-10 20:33:39|  2015-01-10 20:35:31|               1|
0.5|-74.00908660888672| 40.71381759643555|          1|
N|-74.00432586669922| 40.71998596191406|           2|         3.5|  0.5|    0.5|
0.0|         0.0|                   0.3|         4.8|     <1 mile|
|       1| 2015-01-10 20:33:39|  2015-01-10 20:52:58|               1|
3.0|-73.97117614746094|40.762428283691406|          1|
N|-74.00418090820312|40.742652893066406|           2|        15.0|  0.5|    0.5|
0.0|         0.0|                   0.3|        16.3|   2-5 miles|
|       1| 2015-01-10 20:33:39|  2015-01-10 20:53:52|               1|
9.0|-73.87437438964844|  40.7740478515625|          1|
N|-73.98697662353516| 40.75819396972656|           1|        27.0|  0.5|    0.5|
6.7|        5.33|                   0.3|       40.33|    >5 miles|
|       1| 2015-01-10 20:33:39|  2015-01-10 20:58:31|               1|
2.2| -73.9832763671875|40.726009368896484|          1|
N|-73.99246978759766|  40.7496337890625|           2|        14.0|  0.5|    0.5|
0.0|         0.0|                   0.3|        15.3|   2-5 miles|
|       1| 2015-01-10 20:33:39|  2015-01-10 20:42:20|               3|
0.8| -74.0026626586914|  40.7341423034668|          1|
N|-73.99501037597656| 40.72632598876953|           1|         7.0|  0.5|    0.5|
1.66|         0.0|                   0.3|        9.96|     <1 mile|
|       1| 2015-01-10 20:33:39|  2015-01-10 21:11:35|               3|
18.2|-73.78304290771484| 40.64435577392578|          2|
N|-73.98759460449219| 40.75935745239258|           2|        52.0|  0.0|    0.5|
0.0|        5.33|                   0.3|       58.13|    >5 miles|
|       1| 2015-01-10 20:33:40|  2015-01-10 20:40:44|               2|
0.9|-73.98558807373047|40.767948150634766|          1|
N|-73.98591613769531| 40.75936508178711|           1|         6.5|  0.5|    0.5|
1.55|         0.0|                   0.3|        9.35|     <1 mile|
|       1| 2015-01-10 20:33:40|  2015-01-10 20:41:39|               1|
0.9|-73.98861694335938| 40.72310256958008|          1|                  N|
-74.00439453125| 40.72858428955078|           1|        7.0|  0.5|    0.5|
1.66|         0.0|                   0.3|        9.96|     <1 mile|
|       1| 2015-01-10 20:33:41|  2015-01-10 20:43:26|               1|
1.1|-73.99378204345703| 40.75141906738281|          1|                  N|
-73.9674072265625| 40.75721740722656|           1|        7.5|  0.5|    0.5|
1.0|         0.0|                   0.3|         9.8|   1-2 miles|
|       1| 2015-01-10 20:33:41|  2015-01-10 20:35:23|               1|
0.3|-74.00836181640625|40.704376220703125|          1|
```

```
N|-74.00977325439453|40.707725524902344|           2|         3.0|  0.5|    0.5|
0.0|         0.0|                     0.3|         4.3|     <1 mile|
|       1| 2015-01-10 20:33:41|  2015-01-10 21:03:04|              1|
3.1|-73.97394561767578| 40.76044845581055|          1|
N|-73.99734497070312| 40.73521041870117|           1|        19.0|  0.5|    0.5|
3.0|         0.0|                     0.3|        23.3|   2-5 miles|
|       1| 2015-01-10 20:33:41|  2015-01-10 20:39:23|              1|
1.1|-74.00672149658203| 40.73177719116211|          1|                 N|
-73.9952163696289| 40.73989486694336|           2|         6.0|  0.5|    0.5|
0.0|         0.0|                     0.3|         7.3|   1-2 miles|
|       2| 2015-01-15 19:05:39|  2015-01-15 19:32:00|              1|
2.38|-73.97642517089844|40.739810943603516|          1|
N|-73.98397827148438| 40.75788879394531|           1|        16.5|  1.0|    0.5|
4.38|         0.0|                     0.3|       22.68|   2-5 miles|
|       2| 2015-01-15 19:05:40|  2015-01-15 19:21:00|              5|
2.83|-73.96870422363281| 40.75424575805664|          1|
N|-73.95512390136719| 40.78685760498047|           2|        12.5|  1.0|    0.5|
0.0|         0.0|                     0.3|        14.3|   2-5 miles|
|       2| 2015-01-15 19:05:40|  2015-01-15 19:28:18|              5|
8.33| -73.8630599975586| 40.76958084106445|          1|
N|-73.95271301269531| 40.78578186035156|           1|        26.0|  1.0|    0.5|
8.08|        5.33|                     0.3|       41.21|     >5 miles|
|       2| 2015-01-15 19:05:41|  2015-01-15 19:20:36|              1|
2.37|-73.94554138183594|40.779422760009766|          1|
N|-73.98085021972656| 40.78608322143555|           1|        11.5|  1.0|    0.5|
0.0|         0.0|                     0.3|        13.3|   2-5 miles|
|       2| 2015-01-15 19:05:41|  2015-01-15 19:20:22|              2|
7.13|-73.87445831298828|40.774009704589844|          1|
N|-73.95237731933594|40.718589782714844|           1|        21.5|  1.0|    0.5|
4.5|         0.0|                     0.3|        27.8|     >5 miles|
+--------+--------------------+--------------------+---------------+-----------
--+------------------+------------------+----------+------------------+---------
---------+------------------+------------+----------+-----+-------+----------+-
-----------+--------------------+------------+------------+
only showing top 20 rows
```

0.8 Task 2: Write a SQL query to calculate the average fare amount for each distance bin.

[ ]:
```python
from pyspark.sql.functions import when

# Create the distance_bin column based on the conditions
df_with_bins = df.withColumn(
    "distance_bin",
    when(df.distance <= 1, "<1 mile")
    .when((df.distance > 1) & (df.distance <= 2), "1-2 miles")
```

```
    .when((df.distance > 2) & (df.distance <= 5), "2-5 miles")
    .when(df.distance > 5, ">5 miles")
    .otherwise("Unknown")
)

# Create a temporary view with the new column
df_with_bins.createOrReplaceTempView("yellow_taxi_with_bins")

# Now run the SQL query without the CASE expression in the ORDER BY clause
query = """
SELECT distance_bin, AVG(fare_amount) AS average_fare_amount
FROM yellow_taxi_with_bins
GROUP BY distance_bin
ORDER BY distance_bin
"""

# Run the query
result_df = spark.sql(query)

# Show the results
result_df.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
File <command-1915345996946306>:6
      1 from pyspark.sql.functions import when
      3 # Create the distance_bin column based on the conditions
      4 df_with_bins = df.withColumn(
      5     "distance_bin",
----> 6     when(df.distance <= 1, "<1 mile")
      7     .when((df.distance > 1) & (df.distance <= 2), "1-2 miles")
      8     .when((df.distance > 2) & (df.distance <= 5), "2-5 miles")
      9     .when(df.distance > 5, ">5 miles")
     10     .otherwise("Unknown")
     11 )
     13 # Create a temporary view with the new column
     14 df_with_bins.createOrReplaceTempView("yellow_taxi_with_bins")

File /databricks/spark/python/pyspark/instrumentation_utils.py:48, in␣
↪_wrap_function.<locals>.wrapper(*args, **kwargs)
     46 start = time.perf_counter()
     47 try:
---> 48     res = func(*args, **kwargs)
     49     logger.log_success(
     50         module_name, class_name, function_name, time.perf_counter() -␣
↪start, signature
     51     )
```

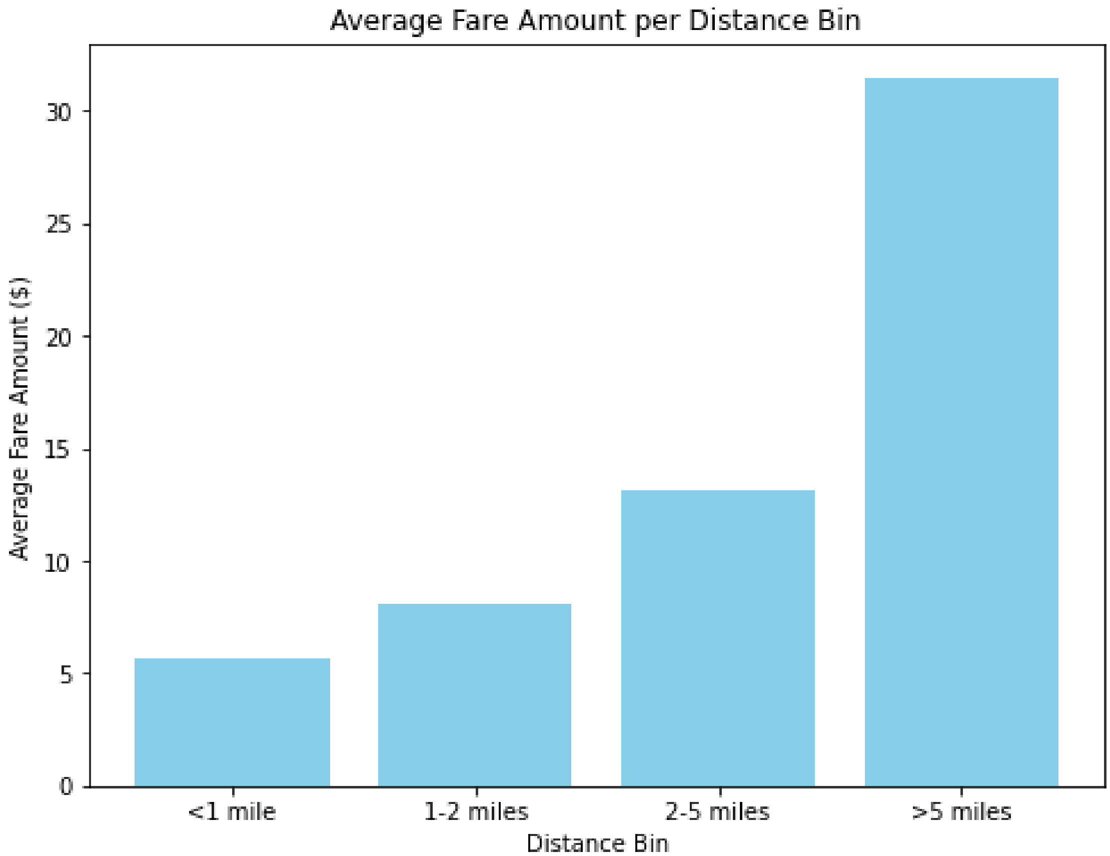
```
    52     return res

File /databricks/spark/python/pyspark/sql/dataframe.py:2964, in DataFrame.
↪__getattr__(self, name)
  2934 """Returns the :class:`Column` denoted by ``name``.
  2935
  2936 .. versionadded:: 1.3.0
  (…)
  2961 +----+
  2962 """
  2963 if name not in self.columns:
-> 2964     raise AttributeError(
  2965         "'%s' object has no attribute '%s'" % (self.__class__.__name__,␣
↪name)
  2966     )
  2967 jc = self._jdf.apply(name)
  2968 return Column(jc)

AttributeError: 'DataFrame' object has no attribute 'distance'
```

0.9 Visualization:

We can visualize the average fare amount per distance bin using a bar chart.

```
[ ]: # Convert the query result to a Pandas DataFrame for plotting
average_fare_by_bin = spark.sql("""
    SELECT distance_bin, AVG(fare_amount) AS average_fare_amount
    FROM yellow_taxi_with_bins
    GROUP BY distance_bin
""").toPandas()

import pandas as pd
import matplotlib.pyplot as plt

# Ensure the bins are in the correct order
bin_order = ['<1 mile', '1-2 miles', '2-5 miles', '>5 miles']
average_fare_by_bin['distance_bin'] = pd.
 ↪Categorical(average_fare_by_bin['distance_bin'], categories=bin_order,␣
 ↪ordered=True)
average_fare_by_bin = average_fare_by_bin.sort_values('distance_bin')

# Plot the average fare amount per distance bin
plt.figure(figsize=(8,6))
plt.bar(average_fare_by_bin['distance_bin'],␣
 ↪average_fare_by_bin['average_fare_amount'], color='skyblue')
plt.xlabel('Distance Bin')
plt.ylabel('Average Fare Amount ($)')
```

```
plt.title('Average Fare Amount per Distance Bin')
plt.show()
```



0.10 Q5:

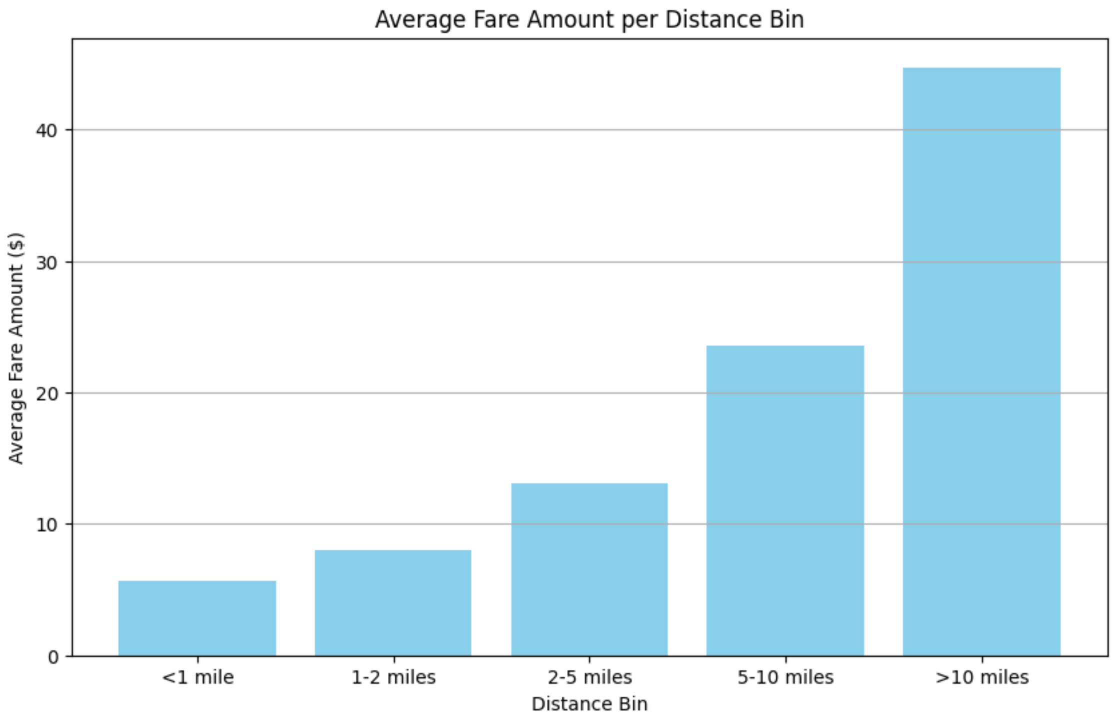
```
[ ]: # SQL query to create distance bins and calculate average fare for each bin
query = """
SELECT
    CASE
        WHEN trip_distance < 1 THEN '<1 mile'
        WHEN trip_distance >= 1 AND trip_distance < 2 THEN '1-2 miles'
        WHEN trip_distance >= 2 AND trip_distance < 5 THEN '2-5 miles'
        WHEN trip_distance >= 5 AND trip_distance < 10 THEN '5-10 miles'
        ELSE '>10 miles'
    END AS distance_bin,
    AVG(fare_amount) AS average_fare
FROM yellow_taxi_data
GROUP BY distance_bin
ORDER BY
```

```
    CASE
        WHEN distance_bin = '<1 mile' THEN 1
        WHEN distance_bin = '1-2 miles' THEN 2
        WHEN distance_bin = '2-5 miles' THEN 3
        WHEN distance_bin = '5-10 miles' THEN 4
        WHEN distance_bin = '>10 miles' THEN 5
    END
"""

# Execute the query
distance_fare_bins = spark.sql(query)

# Show the result
distance_fare_bins.show()
```

```
[Stage 148:=================================>                    (9 + 6) / 15]

+------------+------------------+
|distance_bin|      average_fare|
+------------+------------------+
|     <1 mile| 5.672120578999713|
|   1-2 miles|  8.04041196316821|
|   2-5 miles|13.087495501926742|
|  5-10 miles|23.568011860162635|
|   >10 miles| 44.72180584584484|
+------------+------------------+
```

0.11 Visualization

[ ]:
```
# Convert the Spark DataFrame to Pandas for plotting
distance_fare_bins_df = distance_fare_bins.toPandas()

# Import necessary libraries for plotting
import matplotlib.pyplot as plt

# Plot the average fare per distance bin
plt.figure(figsize=(10, 6))
plt.bar(distance_fare_bins_df['distance_bin'],␣
 ↪distance_fare_bins_df['average_fare'], color='skyblue')
plt.xlabel('Distance Bin')
plt.ylabel('Average Fare Amount ($)')
plt.title('Average Fare Amount per Distance Bin')
plt.grid(axis='y')
plt.show()
```



[ ]:
```
# Convert the Spark DataFrame to Pandas for plotting
distance_fare_bins_df = distance_fare_bins.toPandas()

# Import necessary libraries for plotting
import matplotlib.pyplot as plt

# Plot the average fare per distance bin
plt.figure(figsize=(10, 6))
plt.bar(distance_fare_bins_df['distance_bin'],␣
↪distance_fare_bins_df['average_fare'], color='skyblue')
plt.xlabel('Distance Bin')
plt.ylabel('Average Fare Amount ($)')
plt.title('Average Fare Amount per Distance Bin')
plt.grid(axis='y')
plt.show()
```

1 Q6: Passenger Count Distribution

1.1 Task 1 & 2:

1.1.1 Write a SQL query to count the number of trips by passenger count and calculate the average fare amount for each passenger count group

[ ]:
```
# SQL query to count number of trips by passanger count
trip_per_passanger = spark.sql("""
SELECT passenger_count, count(*) AS trips, avg(fare_amount) AS avg_fare --␣
↪trips counts total number of trips per passanger count, avg_fare take avg␣
↪fare amount for each group
FROM yellow_taxi_data
GROUP BY passenger_count  -- groups by disting passanger count
ORDER BY passenger_count ASC;
""")

trip_per_passanger.show()
```

```
+---------------+-------+------------------+
|passenger_count|  trips|          avg_fare|
+---------------+-------+------------------+
|              0|   6565|11.205294744859103|
|              1|8993870| 11.78195472471847|
|              2|1814594|12.420621907710354|
|              3| 528486|12.124618192345713|
|              4| 253228|12.202182618035936|
|              5| 697645|11.963545757512774|
|              6| 454568|11.797696494253884|
|              7|      9|11.255555555555555|
|              8|     10|29.580000000000002|
|              9|     11|52.900000000000006|
+---------------+-------+------------------+
```

2 Q7: Heatmap of Trip Frequencies

2.1 Task 1: Write SQL queries to find the most frequent pickup and drop-off locations rounded to three decimal places of latitude and longitude

[ ]:
```
#sql to find frequent pickup long and lat rounded by 3 grouped by long and lat
Frequent_pickup_locations = spark.sql("""
SELECT round(pickup_longitude, 3) AS longitude, round(pickup_latitude, 3) AS␣
↪latitude, count(*) AS frequency
FROM yellow_taxi_data
GROUP BY  longitude, latitude
""")
```

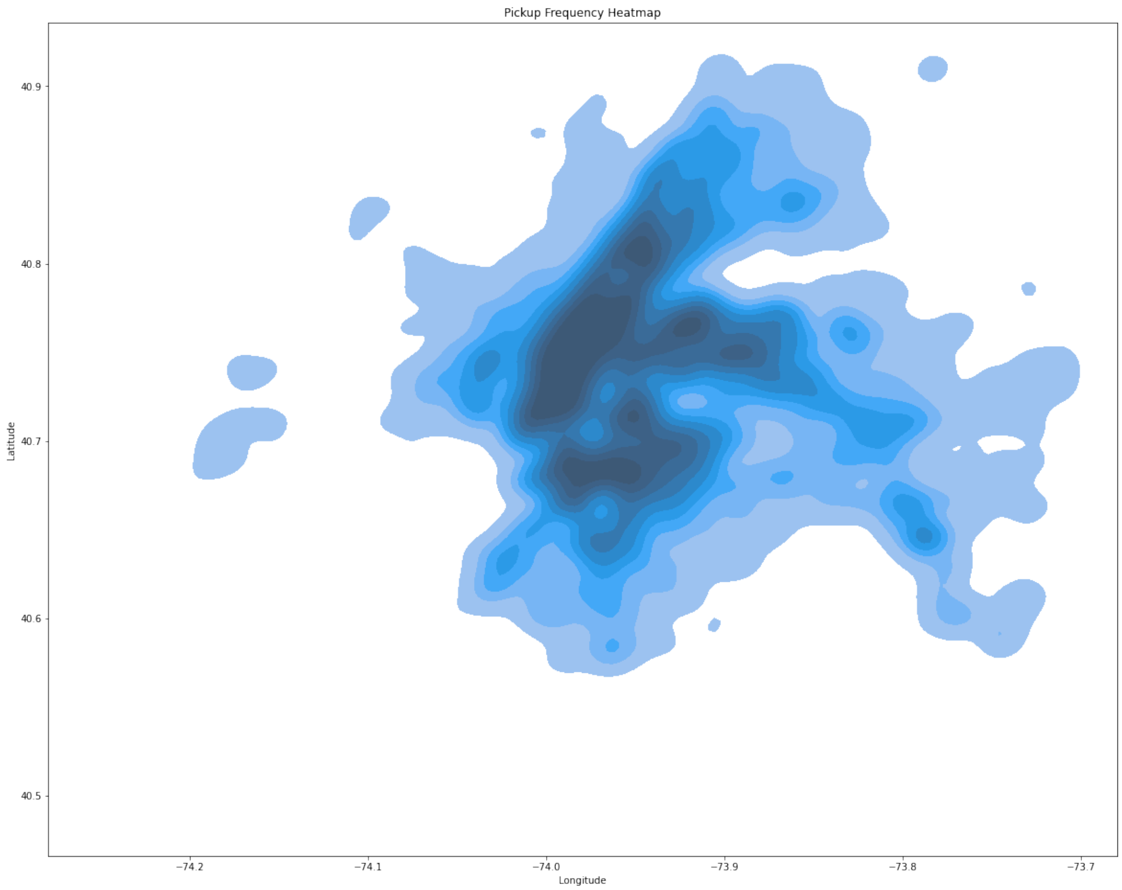
```
Frequent_pickup_locations.show()

#sql to find frequent dropofflong and lat rounded by 3 grouped by long and lat
Frequent_dropoff_locations = spark.sql("""
SELECT round(dropoff_longitude, 3) AS longitude, round(dropoff_latitude, 3) AS␣
↪latitude, count(*) AS frequency
FROM yellow_taxi_data
GROUP BY longitude, latitude
""")

Frequent_dropoff_locations.show()
```

```
+---------+--------+---------+
|longitude|latitude|frequency|
+---------+--------+---------+
|  -74.004|  40.748|    14040|
|  -73.984|  40.762|     8200|
|   -73.98|  40.786|     5622|
|  -73.979|  40.762|    22089|
|  -73.991|  40.724|     6126|
|  -73.997|  40.719|     2627|
|  -73.963|  40.777|     2433|
|  -73.961|  40.779|      864|
|  -73.995|  40.742|     1019|
|  -73.985|  40.756|     2336|
|  -73.944|  40.835|      539|
|   -73.97|  40.749|     1817|
|   -73.98|   40.78|     2244|
|    -74.0|  40.711|      240|
|  -74.008|  40.724|     2186|
|  -73.991|  40.714|      265|
|  -73.911|    40.7|      100|
|  -73.968|  40.761|     3584|
|    -73.9|  40.746|      175|
|   -73.98|  40.757|      722|
+---------+--------+---------+
only showing top 20 rows

+---------+--------+---------+
|longitude|latitude|frequency|
+---------+--------+---------+
|  -73.979|  40.762|    23961|
|  -73.959|  40.776|     1920|
|  -73.984|  40.762|     7583|
|  -74.004|  40.748|     9561|
|  -74.008|  40.724|     1273|
|   -73.98|  40.786|     4445|
|  -73.985|  40.756|     4172|
```

```
|  -73.995| 40.742|     2258|
|  -73.991| 40.714|      694|
|  -73.951| 40.669|      136|
|  -73.944| 40.835|      505|
|  -73.999| 40.762|      471|
|   -73.98|  40.78|     3119|
|  -73.991| 40.724|     5543|
|  -73.943| 40.802|      685|
|  -73.985| 40.692|      720|
|  -73.961| 40.779|     3675|
|   -73.97| 40.749|     3103|
|  -73.943| 40.707|      555|
|   -73.98| 40.757|     1825|
+---------+--------+---------+
only showing top 20 rows
```

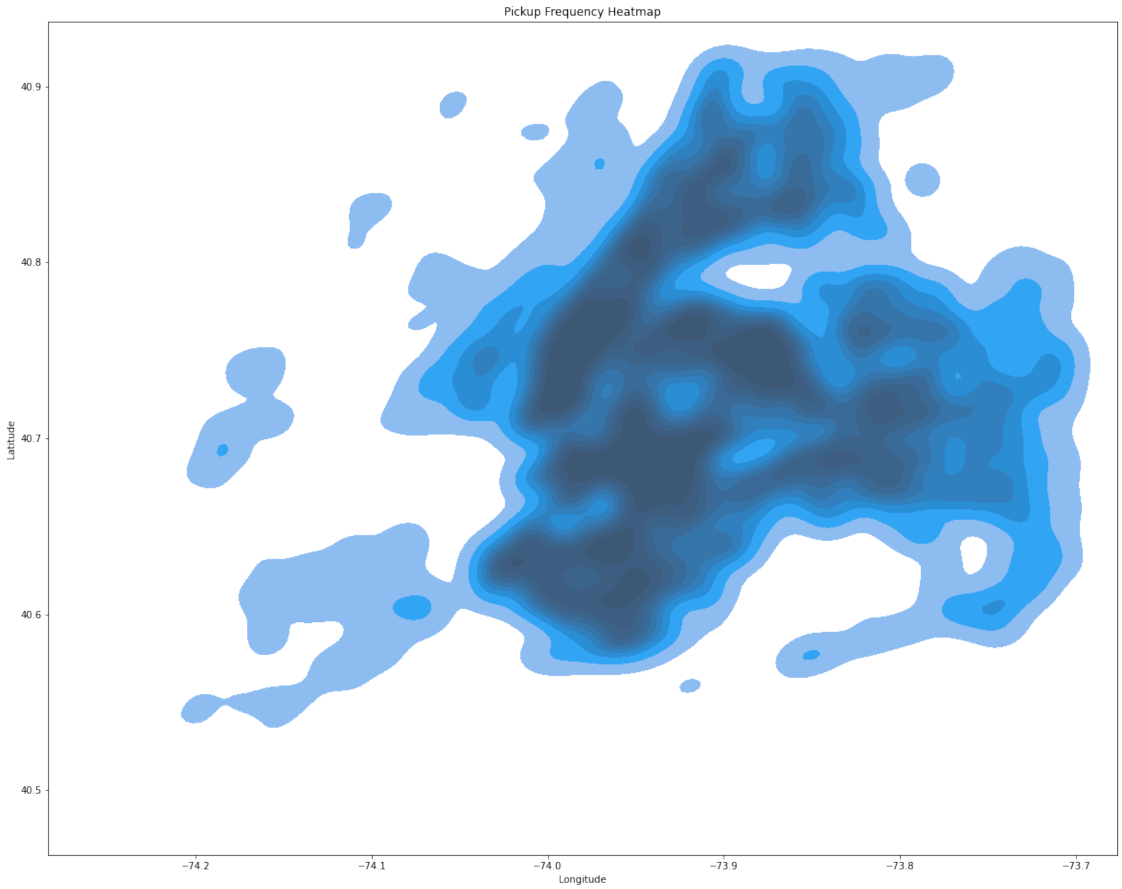
2.2 Task 2: Visualize the frequencies using a heatmap.

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = Frequent_pickup_locations.toPandas()

#nyc bound
lat_min, lat_max = 40.4774, 40.9176
lon_min, lon_max = -74.2591, -73.7004

# Filter points within nyc bound only
cleaned = data[(data['latitude'] >= lat_min) & (data['latitude'] <= lat_max) &
               (data['longitude'] >= lon_min) & (data['longitude'] <= lon_max)]


plt.figure(figsize=(20, 16))

#plot
sns.kdeplot(x=cleaned['longitude'], y=cleaned['latitude'], fill=True,␣
 ↪shade=True, bw_adjust=0.5)
plt.title('Pickup Frequency Heatmap')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



[ ]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = Frequent_dropoff_locations.toPandas()

#nyc bound
lat_min, lat_max = 40.4774, 40.9176
lon_min, lon_max = -74.2591, -73.7004

# Filter points within nyc bound only
cleaned = data[(data['latitude'] >= lat_min) & (data['latitude'] <= lat_max) &
            (data['longitude'] >= lon_min) & (data['longitude'] <= lon_max)]


plt.figure(figsize=(20, 16))

#plot
```

```
sns.kdeplot(x=cleaned['longitude'], y=cleaned['latitude'], fill=True,␣
↪shade=True, bw_adjust=0.5)
plt.title('Pickup Frequency Heatmap')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



3 Q8: Busiest Days and Times Analysis

3.1 Task 1: Write a SQL query to determine which day of the week generated the most revenue.

[ ]:
```
revenue_per_day = spark.sql("""
--> since yellow taxi payment is determined at the end of the trip the day of␣
↪the revenue generated will be based upon day at drop off
--> in cases where the pickup date is thursday and drop off is friday the fare␣
↪amount will be according to the drop off date of friday.
SELECT
CASE dayofweek(tpep_dropoff_datetime)
```

```
  WHEN 1 THEN 'Sunday'
  WHEN 2 THEN 'Monday'
  WHEN 3 THEN 'Tuesday'
  WHEN 4 THEN 'Wednesday'
  WHEN 5 THEN 'Thursday'
  WHEN 6 THEN 'Friday'
  WHEN 7 THEN 'Saturday'
END AS daysWeek, sum(fare_amount) AS Revenue
FROM yellow_taxi_data
GROUP BY daysWeek
ORDER BY Revenue DESC
""")

revenue_per_day.show()
```

```
+---------+--------------------+
| daysWeek|             Revenue|
+---------+--------------------+
| Saturday|2.7051229790000014E7|
|   Friday|2.6806555599999305E7|
| Thursday|2.6465400489999287E7|
|Wednesday|1.9920160110000014E7|
|   Sunday|1.9383329830000013E7|
|   Monday|1.6159452960000008E7|
|  Tuesday|1.5998956560000008E7|
+---------+--------------------+
```

3.2 Task 2: Write another query to find the busiest hour of the day based on the count of pickups

[ ]:
```
busiest_hour = spark.sql("""
SELECT hour(tpep_pickup_datetime) AS hour_of_the_day, Count(*) AS pickups
FROM yellow_taxi_data
GROUP BY hour_of_the_day
ORDER BY pickups DESC
""")

busiest_hour.show(24)
```

```
+---------------+-------+
|hour_of_the_day|pickups|
+---------------+-------+
|             19| 805230|
|             18| 799587|
|             20| 733952|
|             21| 711579|
|             22| 686959|
```

```
|              17| 668790|
|              14| 658887|
|              15| 648688|
|              12| 637479|
|              13| 635587|
|              11| 596504|
|              23| 592429|
|               9| 580034|
|              16| 576598|
|              10| 567818|
|               8| 561802|
|               0| 469971|
|               7| 456127|
|               1| 355145|
|               6| 268455|
|               2| 268133|
|               3| 198524|
|               4| 143271|
|               5| 127437|
+----------------+-------+
```

#Q9: Trip Duration and Time of Day Analysis

3.3 Task 1: Calculate the average trip duration for each hour of the day

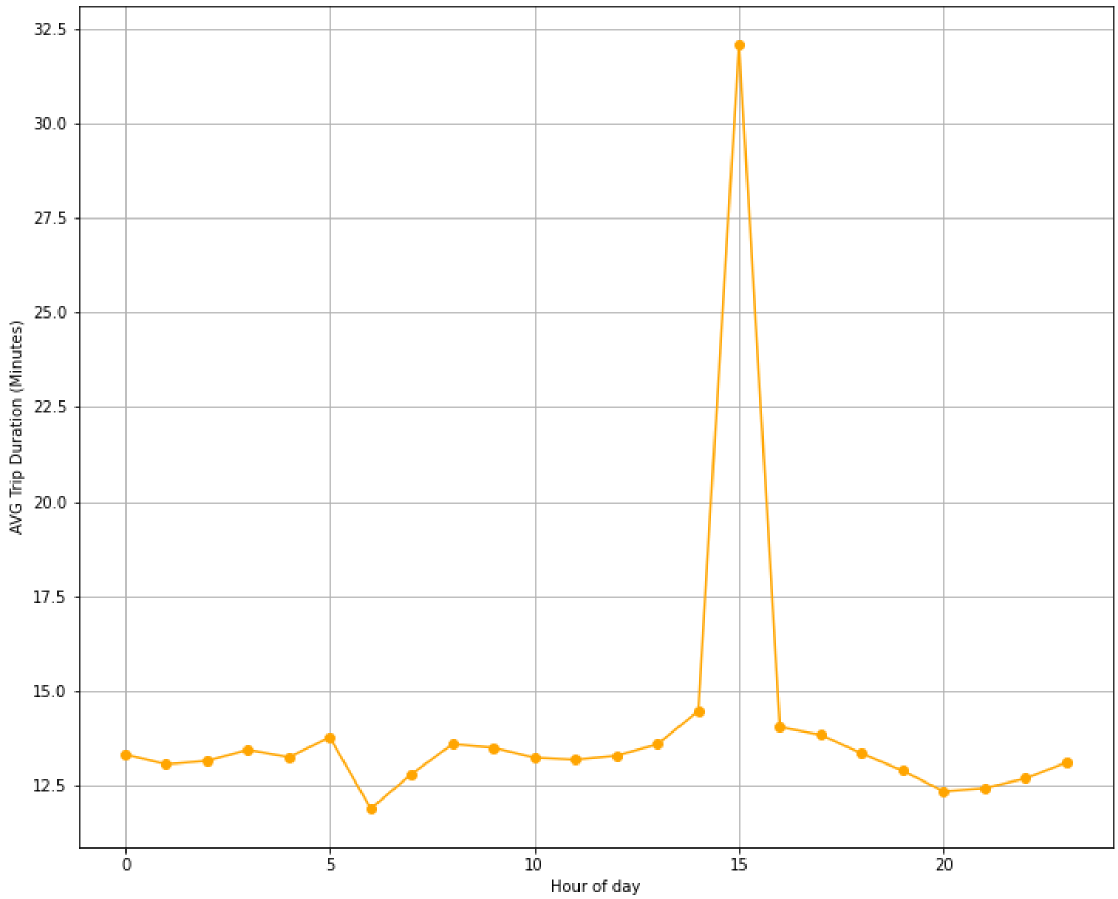
[ ]:
```
#the yellow_taxi_with_duration was created earlier in question3 task 1 with a
↪column called trip_duration_minutes calcualating the time for each trip.
#we can use this dataframe and group data by hour of the day at pickup and
↪average each trip duration for each hour of a day
avg_trip_dration_per_hour = spark.sql("""
SELECT HOUR(tpep_pickup_datetime) AS pickup_hour, avg(trip_duration_minutes) AS
↪avg_trip_minutes
FROM yellow_taxi_with_duration
GROUP BY pickup_hour
ORDER BY pickup_hour;
""")

avg_trip_dration_per_hour.show(24)
```

```
+-----------+------------------+
|pickup_hour|  avg_trip_minutes|
+-----------+------------------+
|          0|13.313322275913483|
|          1|13.054973602331446|
|          2|13.145008571616827|
|          3|13.435592589980732|
|          4|13.242055614883688|
```

```
|          5| 13.76538367977902|
|          6|11.877530064008267|
|          7|12.796801292914775|
|          8|13.587299440016215|
|          9|13.489896511813681|
|         10|13.228424923713806|
|         11| 13.17238671771968|
|         12|13.280298749710424|
|         13|13.579905845567446|
|         14|14.454319127053141|
|         15| 32.09129095959835|
|         16|14.041422302308805|
|         17|13.826888584857201|
|         18| 13.33810871529094|
|         19|12.883962532444134|
|         20|12.333422598934755|
|         21|12.416918945987238|
|         22|12.678930717359632|
|         23|13.098988655180626|
+-----------+------------------+
```

3.4 Visualization

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

data = avg_trip_dration_per_hour.toPandas()

plt.figure(figsize=(12, 10))
plt.plot(data['pickup_hour'], data['avg_trip_minutes'], marker='o',␣
 ↪color='orange')
plt.grid(True)
plt.xlabel('Hour of day')
plt.ylabel('AVG Trip Duration (Minutes)')

plt.show()
```



3.5 Q10: Payment Type Fare Comparison

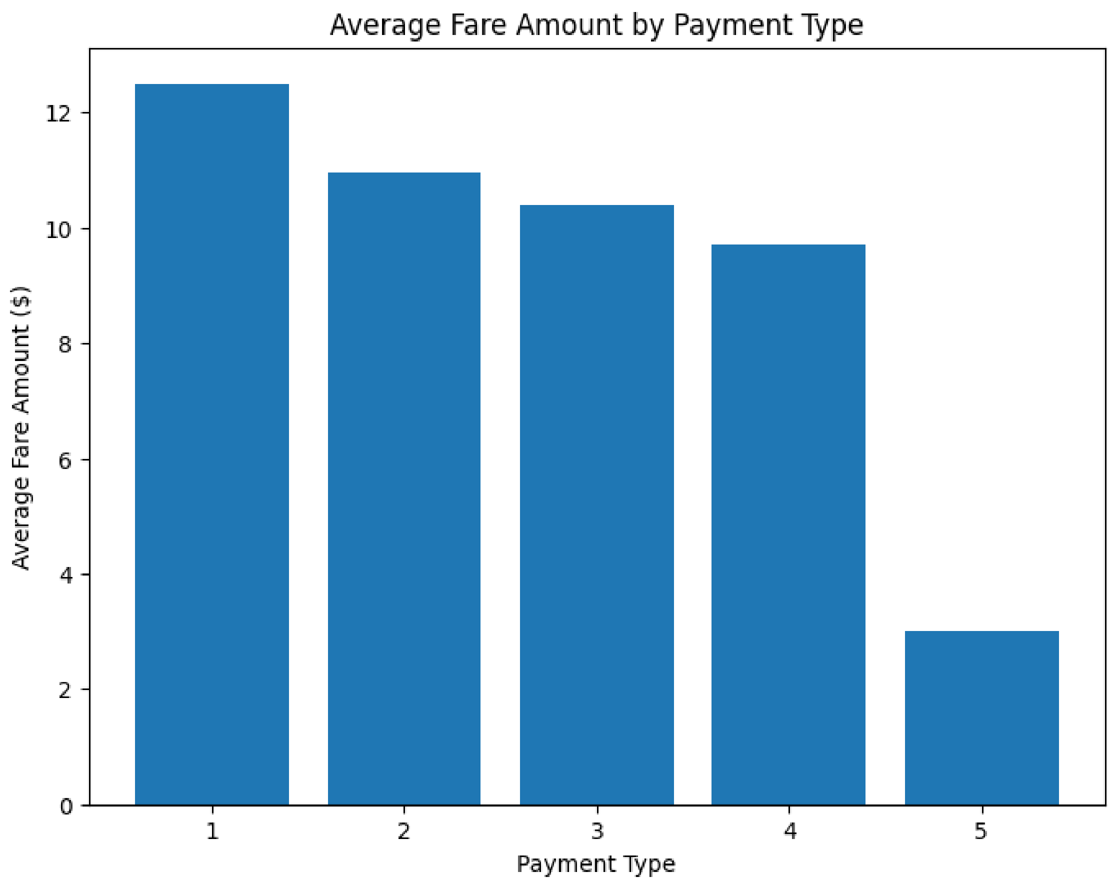
[ ]:
```
# SQL query to calculate average fare amount for each payment type
query = """
SELECT payment_type, AVG(fare_amount) AS average_fare_amount
FROM yellow_taxi_data
GROUP BY payment_type
"""

# Execute the query
result_df = spark.sql(query)

# Show the result
result_df.show()
```

```
[Stage 49:==================================>                    (9 + 6) / 15]

+------------+-------------------+
|payment_type|average_fare_amount|
+------------+-------------------+
```

```
|           1|  12.50128976774208|
|           3| 10.396839148892102|
|           4|   9.71586034079519|
|           2| 10.948657421477954|
|           5|                3.0|
+------------+-------------------+
```

3.6 Visualization:

```
[ ]: # Convert the Spark DataFrame to a Pandas DataFrame for plotting
average_fare_by_payment_type = result_df.toPandas()

# Import necessary libraries for plotting
import matplotlib.pyplot as plt

# Plot the average fare by payment type
plt.figure(figsize=(8, 6))
plt.bar(average_fare_by_payment_type['payment_type'],␣
 ↪average_fare_by_payment_type['average_fare_amount'])
plt.xlabel('Payment Type')
plt.ylabel('Average Fare Amount ($)')
plt.title('Average Fare Amount by Payment Type')
plt.show()
```



3.7 Q11: : Time Series Analysis of Trips

[ ]:
```
# SQL query to count trips per day
query = """
SELECT DATE(tpep_pickup_datetime) AS trip_date, COUNT(*) AS trip_count
FROM yellow_taxi_data
GROUP BY trip_date
ORDER BY trip_date
"""

# Execute the query
daily_trip_counts = spark.sql(query)

# Show the result
daily_trip_counts.show()
```

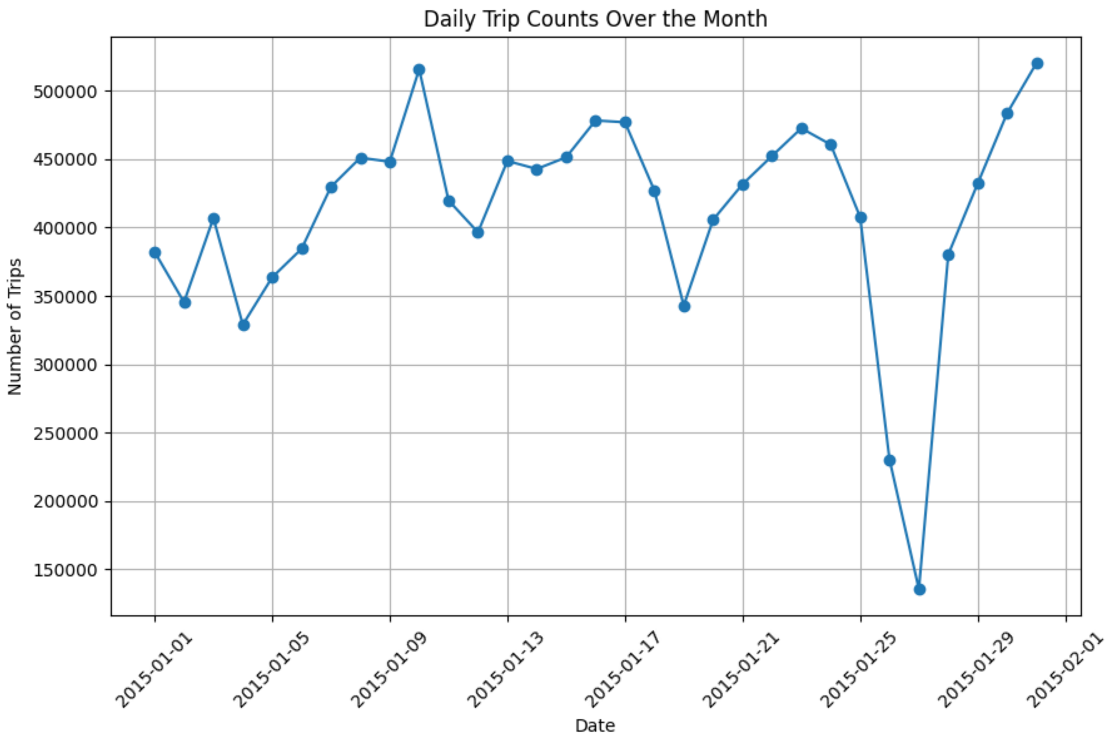
```
[Stage 55:==================================>                          (9 + 6) / 15]

+----------+----------+
| trip_date|trip_count|
```

```
+----------+----------+
|2015-01-01|    382014|
|2015-01-02|    345296|
|2015-01-03|    406769|
|2015-01-04|    328848|
|2015-01-05|    363454|
|2015-01-06|    384324|
|2015-01-07|    429653|
|2015-01-08|    450920|
|2015-01-09|    447947|
|2015-01-10|    515540|
|2015-01-11|    419629|
|2015-01-12|    396367|
|2015-01-13|    448517|
|2015-01-14|    442656|
|2015-01-15|    451186|
|2015-01-16|    478124|
|2015-01-17|    476827|
|2015-01-18|    427042|
|2015-01-19|    342795|
|2015-01-20|    405581|
+----------+----------+
only showing top 20 rows
```

3.8 Visualization:

[ ]:
```
# Convert the result to a Pandas DataFrame
import pandas as pd
daily_trip_counts_df = daily_trip_counts.toPandas()

# Import necessary libraries for plotting
import matplotlib.pyplot as plt

# Convert 'trip_date' to datetime format for accurate plotting
daily_trip_counts_df['trip_date'] = pd.
 ↪to_datetime(daily_trip_counts_df['trip_date'])

# Plot the daily trip counts
plt.figure(figsize=(10, 6))
plt.plot(daily_trip_counts_df['trip_date'], daily_trip_counts_df['trip_count'],␣
 ↪marker='o')
plt.xlabel('Date')
plt.ylabel('Number of Trips')
plt.title('Daily Trip Counts Over the Month')
plt.xticks(rotation=45)
```

```
plt.grid(True)
plt.show()
```



[ ]:

```

```

3.9 Q12: Location Analysis

[ ]:

```
# SQL query to find the top 10 pickup locations
pickup_query = """
SELECT pickup_longitude, pickup_latitude, COUNT(*) AS trip_count
FROM yellow_taxi_data
GROUP BY pickup_longitude, pickup_latitude
ORDER BY trip_count DESC
LIMIT 10
"""

# Execute the query
top_pickup_locations = spark.sql(pickup_query)

# Show the result
top_pickup_locations.show()
```

```
24/11/07 13:53:01 WARN RowBasedKeyValueBatch: Calling spill() on
```

```
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:01 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:01 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:01 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:01 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:01 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:01 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:01 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:05 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:05 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:05 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:05 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:05 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:05 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:05 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.

+-----------------+-----------------+----------+
| pickup_longitude|  pickup_latitude|trip_count|
+-----------------+-----------------+----------+
|              0.0|              0.0|    243478|
|-73.94863891601562| 40.74489974975586|      1043|
| -74.1863021850586| 40.69314193725586|       729|
| -73.9867172241211|  40.7222900390625|       429|
|-73.91512298583984| 40.74357604980469|       306|
|-74.00314331054688| 40.72767639160156|       233|
|-73.92151641845703|40.691463470458984|       153|
|-73.98845672607422|40.731502532958984|       147|
|-73.97827911376953|  40.6429443359375|       121|
|-73.94208526611328|40.754417419433594|       108|
+-----------------+-----------------+----------+


24/11/07 13:53:10 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:10 WARN RowBasedKeyValueBatch: Calling spill() on
```

```
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:10 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:10 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:10 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:10 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:10 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:10 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:15 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:15 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:15 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:15 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:15 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:15 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
24/11/07 13:53:15 WARN RowBasedKeyValueBatch: Calling spill() on
RowBasedKeyValueBatch. Will not spill but return 0.
[Stage 83:>                                                         (0 + 8) / 9]

+-----------------+-----------------+----------+
| dropoff_longitude|  dropoff_latitude|trip_count|
+-----------------+-----------------+----------+
|              0.0|              0.0|    235318|
|-73.94863891601562| 40.74489974975586|      1043|
| -74.1863021850586| 40.69314193725586|       729|
| -73.9867172241211|  40.7222900390625|       428|
|-73.91512298583984| 40.74357604980469|       322|
|-74.00314331054688| 40.72767639160156|       233|
|-73.98845672607422|40.731502532958984|       155|
|-73.92151641845703|40.691463470458984|       153|
|-73.97827911376953|  40.6429443359375|       121|
|-73.94208526611328|40.754417419433594|       108|
+-----------------+-----------------+----------+
```

[ ]:
```python
dropoff_query = """
SELECT dropoff_longitude, dropoff_latitude, COUNT(*) AS trip_count
FROM yellow_taxi_data
GROUP BY dropoff_longitude, dropoff_latitude
ORDER BY trip_count DESC
LIMIT 10
"""

# Execute the query
top_dropoff_locations = spark.sql(dropoff_query)

# Show the result
top_dropoff_locations.show()
```

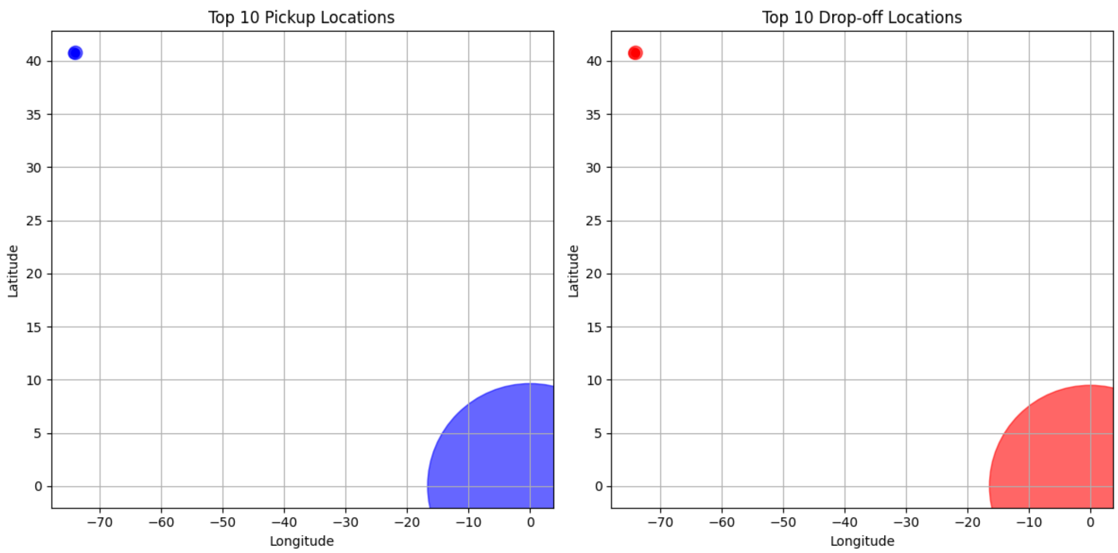
[ ]:
```python

```

3.10 Visualization

[ ]:
```python

```

[ ]:
```python
import matplotlib.pyplot as plt

# Convert the Spark DataFrames to Pandas DataFrames
pickup_df = top_pickup_locations.toPandas()
dropoff_df = top_dropoff_locations.toPandas()

# Plot settings
plt.figure(figsize=(12, 6))

# Scatter plot for top pickup locations
plt.subplot(1, 2, 1)
plt.scatter(pickup_df['pickup_longitude'], pickup_df['pickup_latitude'],
            s=pickup_df['trip_count'] * 0.1, c='blue', alpha=0.6)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Top 10 Pickup Locations')
plt.grid(True)

# Scatter plot for top drop-off locations
plt.subplot(1, 2, 2)
plt.scatter(dropoff_df['dropoff_longitude'], dropoff_df['dropoff_latitude'],
            s=dropoff_df['trip_count'] * 0.1, c='red', alpha=0.6)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Top 10 Drop-off Locations')
plt.grid(True)
```

```python
# Show the plots
plt.tight_layout()
plt.show()
```



3.11 Q13: Fare Amount Distribution Analysis

[ ]:
```python
# SQL query to get summary statistics for fare amounts
summary_query = """
SELECT
    MIN(fare_amount) AS min_fare,
    MAX(fare_amount) AS max_fare,
    AVG(fare_amount) AS avg_fare,
    PERCENTILE_APPROX(fare_amount, 0.5) AS median_fare,
    STDDEV(fare_amount) AS stddev_fare
FROM yellow_taxi_data
"""

# Execute the query
summary_stats = spark.sql(summary_query)

# Show the result
summary_stats.show()
```

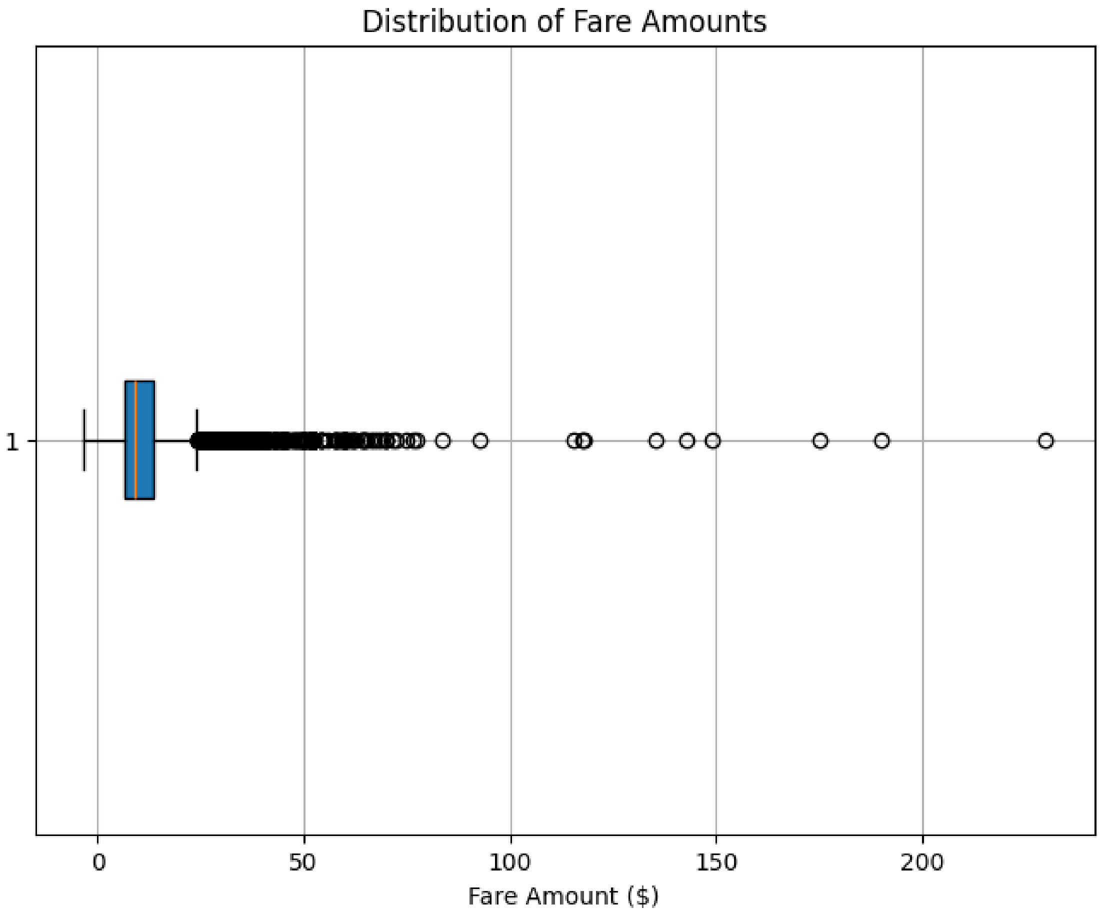
```
[Stage 110:=========================================>              (11 + 4) / 15]

+--------+--------+-----------------+-----------+------------------+
|min_fare|max_fare|         avg_fare|median_fare|       stddev_fare|
+--------+--------+-----------------+-----------+------------------+
```

```
|  -450.0|  4008.0|11.905659425776989|       9.0|10.302537135952232|
+--------+--------+------------------+----------+------------------+
```

3.12 Visualization:

[ ]:
```python
from pyspark.sql import SparkSession

# Initialize Spark session (if not already initialized)
spark = SparkSession.builder \
    .appName("FareAmountAnalysis") \
    .getOrCreate()

# Execute the query with a LIMIT to reduce data size for testing
limited_fare_amount_df = spark.sql("SELECT fare_amount FROM yellow_taxi_data␣
 ↪LIMIT 10000").toPandas()

# Import necessary libraries for plotting
import matplotlib.pyplot as plt

# Plot a box plot for fare amounts
plt.figure(figsize=(8, 6))
plt.boxplot(limited_fare_amount_df['fare_amount'], vert=False,␣
 ↪patch_artist=True)
plt.xlabel('Fare Amount ($)')
plt.title('Distribution of Fare Amounts')
plt.grid(True)
plt.show()
```

```
24/11/07 14:03:04 WARN SparkSession: Using an existing Spark session; only
runtime SQL configurations will take effect.
```



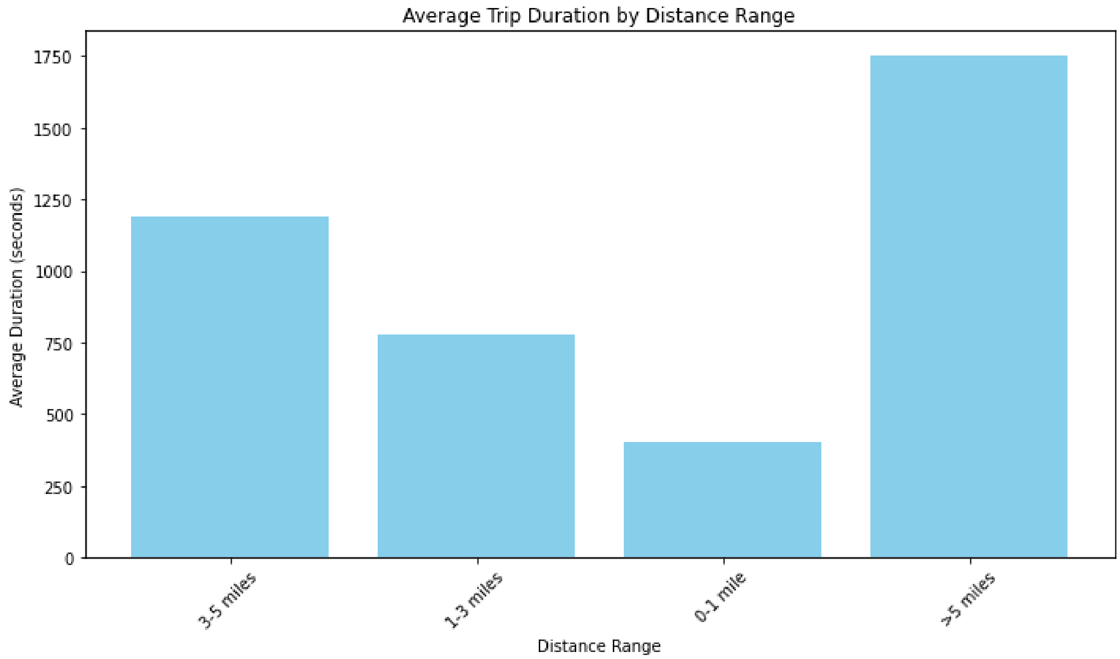
3.13 Question 14

```
[ ]: %sql
SELECT
    CASE
        WHEN trip_distance BETWEEN 0 AND 1 THEN '0-1 mile'
        WHEN trip_distance > 1 AND trip_distance <= 3 THEN '1-3 miles'
        WHEN trip_distance > 3 AND trip_distance <= 5 THEN '3-5 miles'
        WHEN trip_distance > 5 THEN '>5 miles'
    END AS distance_range,
    AVG(trip_duration) AS avg_duration
FROM (
    SELECT
        trip_distance,
        (unix_timestamp(tpep_dropoff_datetime) -
 ↪unix_timestamp(tpep_pickup_datetime)) / 60 AS trip_duration
    FROM yellow_taxi_data
) trips
```

```
GROUP BY distance_range;
```

3.14 Visualization

```
[ ]: query = """
SELECT
    CASE
        WHEN trip_distance BETWEEN 0 AND 1 THEN '0-1 mile'
        WHEN trip_distance > 1 AND trip_distance <= 3 THEN '1-3 miles'
        WHEN trip_distance > 3 AND trip_distance <= 5 THEN '3-5 miles'
        WHEN trip_distance > 5 THEN '>5 miles'
    END AS distance_range,
    AVG(trip_duration) AS avg_duration
FROM (
    SELECT
        trip_distance,
        (unix_timestamp(tpep_dropoff_datetime) -␣
 ↪unix_timestamp(tpep_pickup_datetime)) AS trip_duration
    FROM yellow_taxi_data
) trips
GROUP BY distance_range
"""
df = spark.sql(query).toPandas()  # Convert the Spark DataFrame to Pandas

# Plot the data
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.bar(df['distance_range'], df['avg_duration'], color='skyblue')
plt.xlabel('Distance Range')
plt.ylabel('Average Duration (seconds)')
plt.title('Average Trip Duration by Distance Range')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



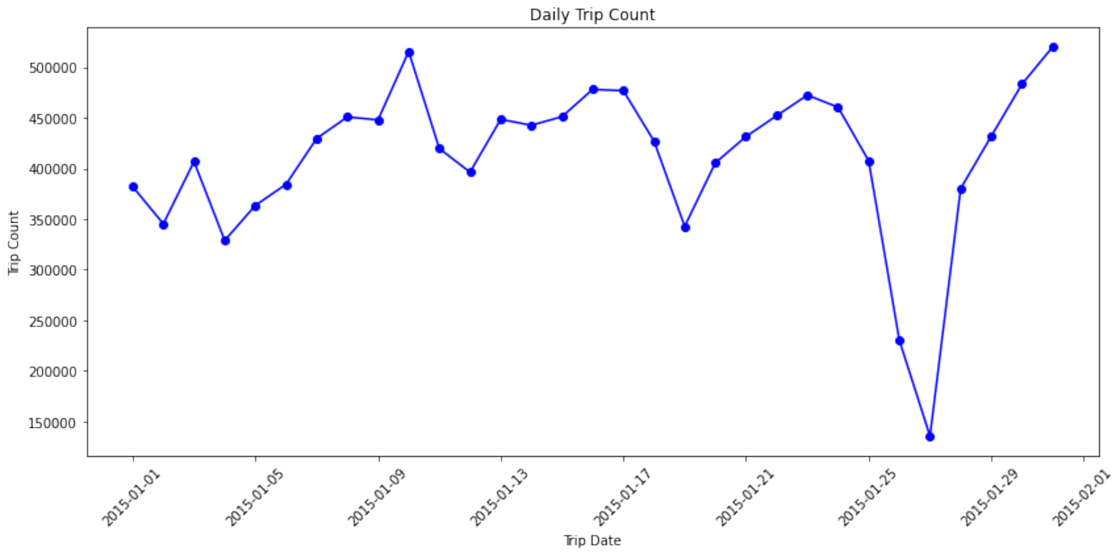
3.15 Question 15

[ ]:
```
%sql
SELECT
    DATE(tpep_pickup_datetime) AS trip_date,
    COUNT(*) AS trip_count
FROM yellow_taxi_data
GROUP BY trip_date
ORDER BY trip_date;
```

3.16 Question 15 Visualization

[ ]:
```
query = """
SELECT
    DATE(tpep_pickup_datetime) AS trip_date,
    COUNT(*) AS trip_count
FROM yellow_taxi_data
GROUP BY trip_date
ORDER BY trip_date
"""
df = spark.sql(query).toPandas()

# Plotting the data
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(12, 6))
plt.plot(df['trip_date'], df['trip_count'], marker='o', linestyle='-',␣
 ↪color='b')
plt.xlabel('Trip Date')
plt.ylabel('Trip Count')
plt.title('Daily Trip Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



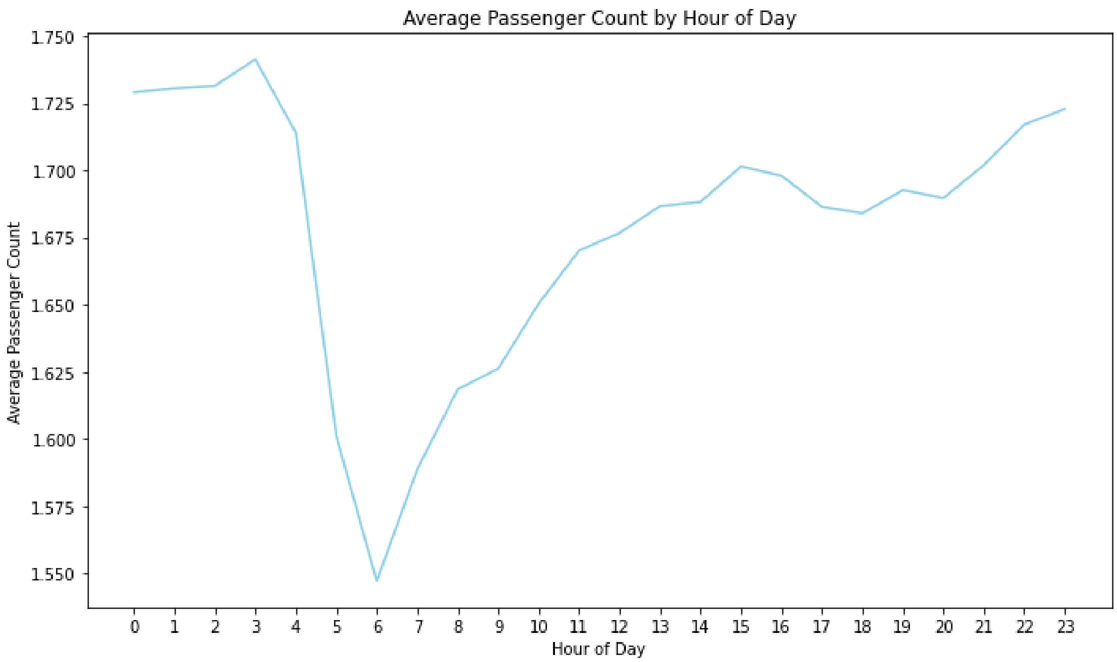
3.17 Question 17

[ ]:
```
%sql
SELECT
    HOUR(tpep_pickup_datetime) AS hour_of_day,
    AVG(passenger_count) AS avg_passenger_count
FROM yellow_taxi_data
GROUP BY hour_of_day
ORDER BY hour_of_day;
```

3.18 Question 17 Visualization

[ ]:
```
query = """
SELECT
    HOUR(tpep_pickup_datetime) AS hour_of_day,
    AVG(passenger_count) AS avg_passenger_count
FROM yellow_taxi_data
GROUP BY hour_of_day
```

```
ORDER BY hour_of_day
"""
df = spark.sql(query).toPandas()

# Plotting the data
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.plot(df['hour_of_day'], df['avg_passenger_count'], color='skyblue')
plt.xlabel('Hour of Day')
plt.ylabel('Average Passenger Count')
plt.title('Average Passenger Count by Hour of Day')
plt.xticks(range(0, 24))  # Show each hour from 0 to 23
plt.tight_layout()
plt.show()
```



3.19 Question 18

[ ]:
```
%sql
SELECT
    DAYOFWEEK(tpep_pickup_datetime) AS day_of_week,
    SUM(total_amount) AS total_revenue
FROM yellow_taxi_data
GROUP BY day_of_week
ORDER BY day_of_week;
```

3.20 Question 18 Visualization

[ ]:
```
query = """
SELECT
    DAYOFWEEK(tpep_pickup_datetime) AS day_of_week,
    SUM(total_amount) AS total_revenue
FROM yellow_taxi_data
GROUP BY day_of_week
ORDER BY day_of_week
"""
df = spark.sql(query).toPandas()

# Map day numbers to day names for better readability
day_names = {1: 'Sunday', 2: 'Monday', 3: 'Tuesday', 4: 'Wednesday',
             5: 'Thursday', 6: 'Friday', 7: 'Saturday'}
df['day_of_week'] = df['day_of_week'].map(day_names)

# Plotting the data
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.bar(df['day_of_week'], df['total_revenue'], color='skyblue')
plt.xlabel('Day of the Week')
plt.ylabel('Total Revenue')
plt.title('Total Revenue by Day of the Week')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```