1

```
data2015_1c = spark.read.csv('dbfs:/FileStore/yellow_tripdata_2015_01.csv', header=True, inferSchema=True)
```

2

```
#Q1: Use SQL queries to find outliers
data2015_1c.createOrReplaceTempView("data")

#SQL query to find taxi trips that have a fare amount over $1000
over_1000 = spark.sql("""
    select tpep_pickup_datetime, tpep_dropoff_datetime, trip_distance, passenger_count, fare_amount
    from data
    where fare_amount > 1000
""")

print("Taxi trips with fare amount over $1000")
over_1000.show()

#SQL query to find taxi trips with fare amounts that are 0 or negative
zero_and_negative = spark.sql("""
    select tpep_pickup_datetime, tpep_dropoff_datetime, trip_distance, passenger_count, fare_amount
    from data
    where fare_amount <= 0
""")

print("Taxi trips with fare amounts that are 0 or negative")
zero_and_negative.show()
```

```
| 2015-01-15 17:33:24|  2015-01-15 17:33:31|          0.0|              2|       -2.5|
| 2015-01-21 10:16:35|  2015-01-21 10:16:54|          0.0|              1|        0.0|
| 2015-01-06 12:43:31|  2015-01-06 12:46:07|         0.23|              5|        0.0|
| 2015-01-23 23:57:43|  2015-01-24 00:35:26|         13.4|              2|        0.0|
| 2015-01-16 16:00:45|  2015-01-16 16:00:53|          0.0|              1|       -2.5|
| 2015-01-08 22:26:34|  2015-01-08 22:26:34|          0.0|              1|        0.0|
| 2015-01-31 23:38:52|  2015-01-31 23:38:54|          0.0|              2|      -52.0|
| 2015-01-22 09:32:57|  2015-01-22 09:54:39|          2.4|              3|        0.0|
| 2015-01-05 02:27:56|  2015-01-05 02:32:51|          2.9|              2|        0.0|
| 2015-01-10 02:23:53|  2015-01-10 02:23:58|          0.0|              2|       -6.8|
| 2015-01-13 08:45:10|  2015-01-13 08:46:32|          0.0|              5|        0.0|
| 2015-01-14 11:52:09|  2015-01-14 11:52:20|          0.0|              1|      -52.0|
| 2015-01-03 02:01:25|  2015-01-03 02:01:54|         0.03|              1|       -2.5|
| 2015-01-20 20:44:47|  2015-01-20 20:44:48|          0.0|              1|        0.0|
| 2015-01-17 11:12:35|  2015-01-17 11:14:52|          0.0|              1|        0.0|
| 2015-01-12 15:07:29|  2015-01-12 15:07:35|          0.0|              1|      -52.0|
| 2015-01-06 14:07:25|  2015-01-06 14:08:27|         0.03|              1|       -2.5|
| 2015-01-10 21:10:20|  2015-01-10 21:12:39|         0.03|              1|       -3.5|
+--------------------+---------------------+-------------+---------------+-----------+
only showing top 20 rows
```

3

```
#Q2: Correlation analysis using SQL

#SQL query to calculate the correlation coefficient for fare amount and trip distance
corr_fare_distance = spark.sql("""
    select FORMAT_STRING('%.10E',CORR(fare_amount, trip_distance))
    as corr_fare_distance
    from data
    where fare_amount >= 0 AND trip_distance >= 0
""")

corr_fare_distance.show()

#SQL query to calculate the correlation coefficient for total amount and trip distance
corr_totalAmount_distance = spark.sql("""
    select FORMAT_STRING('%.10E',CORR(total_amount, trip_distance))
    as corr_totalAmount_distance
    from data
    where total_amount >= 0 AND trip_distance >= 0
""")

corr_totalAmount_distance.show()

#the results are 4.4191699063E-04 and 3.3291432678E-06. Both of these numbers are very small indicating there are v
correlations between fare amount and distance, and total amount and distance. This could be because of other factor
traffic and the area of the pick up point and destination
```

```
+------------------+
|corr_fare_distance|
+------------------+
|  4.4191699063E-04|
+------------------+

+-------------------------+
|corr_totalAmount_distance|
+-------------------------+
|         3.3291432678E-06|
+-------------------------+
```



```
#Q3: Finding trip duration

#SQL query to calculate the average trip duration based on the amount of passengers
avg_trip_dur_passenger_count = spark.sql("""
    select passenger_count, AVG((unix_timestamp(tpep_dropoff_datetime)-unix_timestamp(tpep_pickup_datetime))/60) AS
    avg_trip_dur_passenger_count
    from data
    where tpep_dropoff_datetime > tpep_pickup_datetime
    group by passenger_count
    order by passenger_count
""")

avg_trip_dur_passenger_count.show()
```

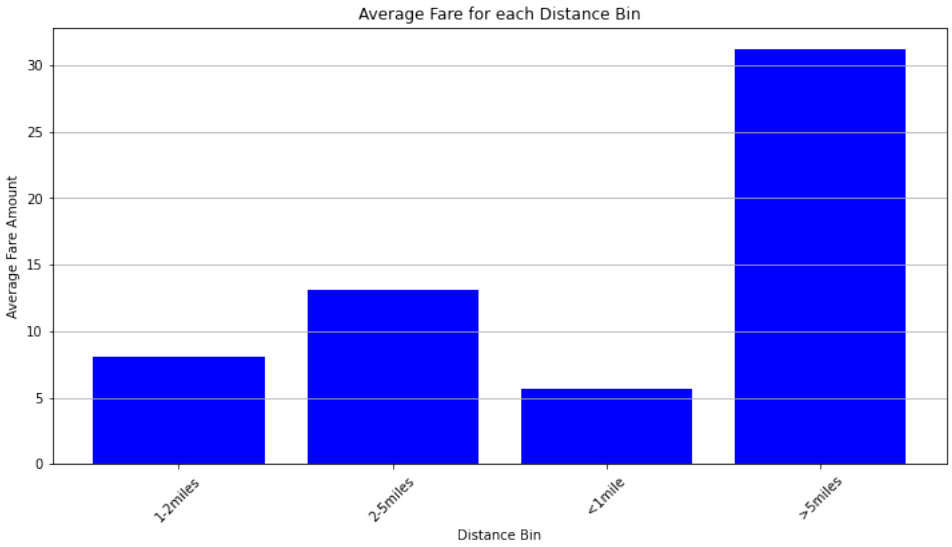
```
+---------------+---------------------------+
|passenger_count|avg_trip_dur_passenger_count|
+---------------+---------------------------+
|              0|          12.75524296025211|
|              1|          14.26690237690622|
|              2|         13.857116808789936|
|              3|          14.03887655721572|
|              4|         13.856105178678012|
|              5|         14.518081252308454|
|              6|         14.051818285817365|
|              7|          8.837037037037037|
|              8|          5.345000000000001|
|              9|         15.793939393939395|
```

```
+---------------+----------------------------+
```

5

```
#Q4 and Q5: Grouping data based on data and trip amount and displaying the results as a table and chart
import matplotlib.pyplot as plt

#SQL query that groups the data into bins based on the trip distance, then finds the average trip distance of each
distance_bins = spark.sql("""
    select
        case
            when trip_distance <1 then '<1mile'
            when trip_distance >=1 and trip_distance <2 then '1-2miles'
            when trip_distance >=2 and trip_distance <5 then '2-5miles'
            else '>5miles'
        end as distance_bin,
        round(AVG(fare_amount),2) as avg_fare
    from data
    where trip_distance >=0
    group by distance_bin
    order by distance_bin
""")

#table to display the results
distance_bins.show()

distance_bins = distance_bins.toPandas()

#bar graph to display the results
plt.figure(figsize = (12,6))
plt.bar(distance_bins['distance_bin'], distance_bins['avg_fare'], color = 'blue')
plt.title("Average Fare for each Distance Bin")
plt.xlabel("Distance Bin")
plt.ylabel("Average Fare Amount")
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```

```
+------------+--------+
|distance_bin|avg_fare|
+------------+--------+
|    1-2miles|    8.04|
|    2-5miles|   13.09|
|      <1mile|    5.67|
|     >5miles|   31.26|
+------------+--------+
```



6

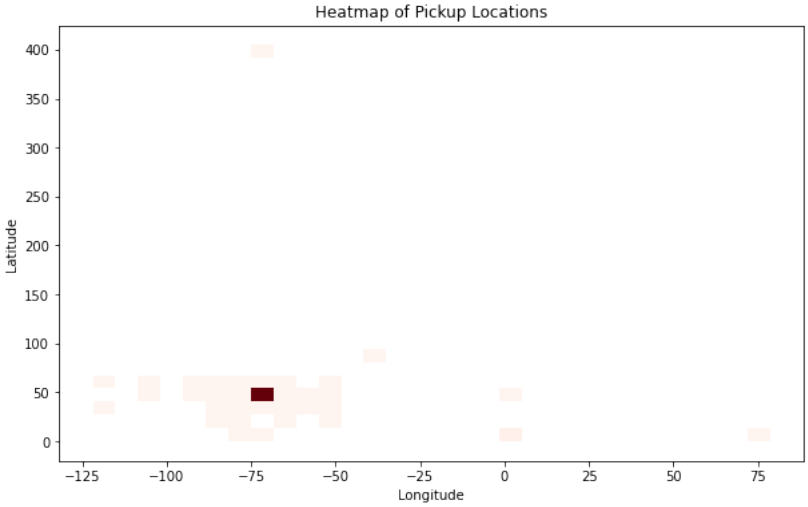
```
# Q6: Passenger Count Distribution
# SQL query to count the number of trips and calculate the average fare for each passenger count
passenger_count_distribution = spark.sql("""
    select
        passenger_count,
        count(*) as trip_count,
        round(avg(fare_amount), 2) as avg_fare
    from data
    where fare_amount <= 0
    group by passenger_count
    order by passenger_count
""")

# displaying the result
passenger_count_distribution.show()
```

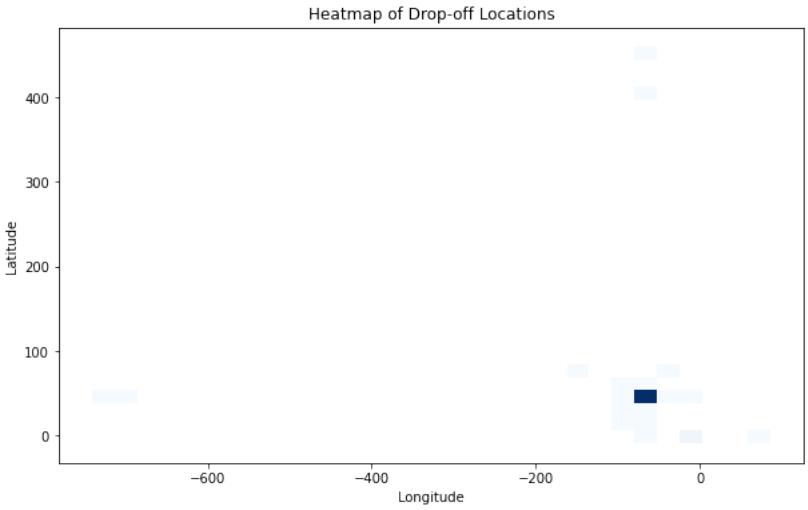
```
+---------------+----------+--------+
|passenger_count|trip_count|avg_fare|
+---------------+----------+--------+
|              0|        62|   -1.61|
|              1|      5183|   -4.71|
|              2|      1101|   -6.63|
|              3|       355|   -4.22|
|              4|       224|   -4.77|
|              5|       497|    -4.7|
|              6|       244|   -4.85|
+---------------+----------+--------+
```

7

```
# Q7: Heatmap of Trip Frequencies
import seaborn as sns
import matplotlib.pyplot as plt

# SQL query to find the frequent pickup locations
frequent_pickup_locations = spark.sql("""
    select
        round(pickup_latitude, 3) as pickup_lat,
        round(pickup_longitude, 3) as pickup_lon,
        count(*) as pickup_frequency
    from data
    group by round(pickup_latitude, 3), round(pickup_longitude, 3)
    order by pickup_frequency desc
""")

frequent_pickup_locations.show()

# SQL query to find the frequent dropoff locations
frequent_dropoff_locations = spark.sql("""
    select
        round(dropoff_latitude, 3) as dropoff_lat,
        round(dropoff_longitude, 3) as dropoff_lon,
        count(*) as dropoff_frequency
    from data
    group by round(dropoff_latitude, 3), round(dropoff_longitude, 3)
    order by dropoff_frequency desc
""")

frequent_dropoff_locations.show()

# Converting to Pandas DataFrame
pickup_df = frequent_pickup_locations.toPandas()
dropoff_df = frequent_dropoff_locations.toPandas()

# Pickup Location Heatmap
plt.figure(figsize=(10, 6))
sns.histplot(x=pickup_df['pickup_lon'], y=pickup_df['pickup_lat'], weights=pickup_df['pickup_frequency'],
             bins=30, pmax=0.9, cmap="Reds")
plt.title("Heatmap of Pickup Locations")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()

# Drop-off Location Heatmap
plt.figure(figsize=(10, 6))
sns.histplot(x=dropoff_df['dropoff_lon'], y=dropoff_df['dropoff_lat'], weights=dropoff_df['dropoff_frequency'],
             bins=30, pmax=0.9, cmap="Blues")
plt.title("Heatmap of Drop-off Locations")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```

```
|     40.756|    -73.975|             15928|
|     40.768|    -73.982|             15284|
+-----------+-----------+------------------+
only showing top 20 rows
```



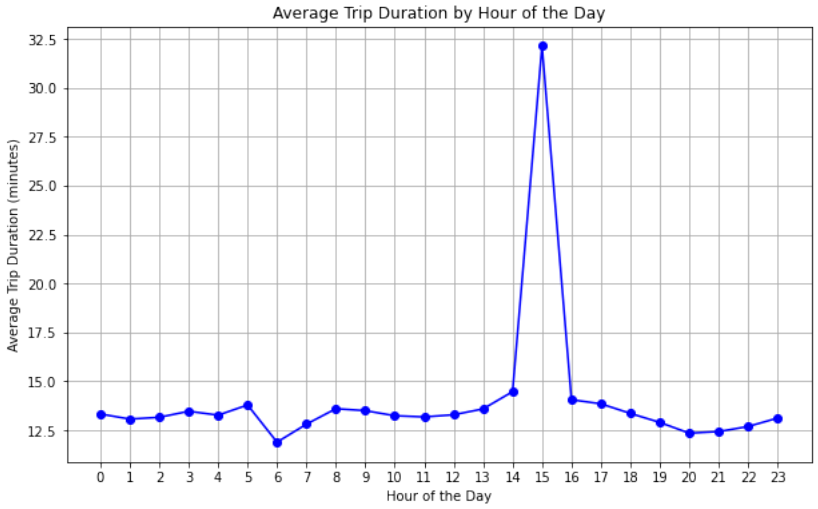


```
# Q8: Busiest Days and Times Analysis
# SQL query to find the revue of the busiest day
busiest_day_revenue = spark.sql("""
    select
        date_format(tpep_pickup_datetime, 'EEEE') as day_of_week,
        round(sum(total_amount), 2) as total_revenue
    from data
    group by date_format(tpep_pickup_datetime, 'EEEE')
    order by total_revenue desc
""")

busiest_day_revenue.show()

# SQL query to find the busiest hour of the day
busiest_hour = spark.sql("""
    select
        hour(tpep_pickup_datetime) as pickup_hour,
        count(*) as pickup_count
    from data
    group by hour(tpep_pickup_datetime)
    order by pickup_count desc
""")

# show data
busiest_hour.show()
```

```
|         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|
+-----------+------------+
only showing top 20 rows
```

```
# Q9: Trip Duration and Time of Day Analysis
import matplotlib.pyplot as plt

# SQL query to calculate the average trip duration for each hour of the day
avg_trip_duration_by_hour = spark.sql("""
    select
        hour(tpep_pickup_datetime) as pickup_hour,
        round(avg(unix_timestamp(tpep_dropoff_datetime) - unix_timestamp(tpep_pickup_datetime)) / 60, 2) as
avg_trip_duration
    from data
    where tpep_dropoff_datetime > tpep_pickup_datetime
    group by hour(tpep_pickup_datetime)
    order by pickup_hour
""")

# show the results
avg_trip_duration_by_hour.show()

# conv to pandas df to make it easier
trip_duration_df = avg_trip_duration_by_hour.toPandas()

# line chart of trip duration by hour
plt.figure(figsize=(10, 6))
plt.plot(trip_duration_df['pickup_hour'], trip_duration_df['avg_trip_duration'], marker='o', color='b')
plt.title("Average Trip Duration by Hour of the Day")
plt.xlabel("Hour of the Day")
plt.ylabel("Average Trip Duration (minutes)")
plt.xticks(range(0, 24))
plt.grid()
plt.show()
```

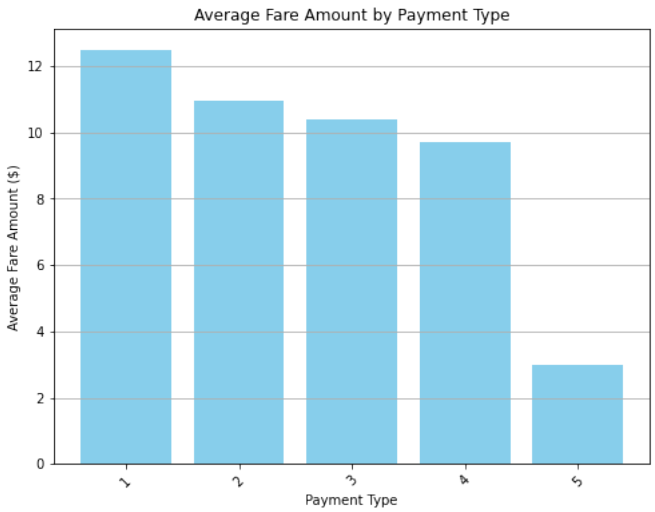
```
|          2|            13.16|
|          3|            13.47|
|          4|            13.27|
|          5|            13.79|
|          6|            11.89|
|          7|            12.81|
|          8|             13.6|
|          9|             13.5|
|         10|            13.24|
|         11|            13.18|
|         12|            13.29|
|         13|            13.59|
|         14|            14.47|
|         15|            32.13|
|         16|            14.06|
|         17|            13.85|
|         18|            13.36|
|         19|             12.9|
+-----------+------------------+
only showing top 20 rows
```



10

```
# Q10: Payment Type Fare Comparison
import matplotlib.pyplot as plt

# SQL query to calculate the average fare amount for each payment type
avg_fare_by_payment_type = spark.sql("""
    select
        payment_type,
        round(avg(fare_amount), 2) as avg_fare_amount
    from data
    group by payment_type
    order by payment_type
""")

# show the results
avg_fare_by_payment_type.show()

# conv the results to a pamndas df
fare_by_payment_df = avg_fare_by_payment_type.toPandas()

# bar chart of average fare amount by payment type
plt.figure(figsize=(8, 6))
plt.bar(fare_by_payment_df['payment_type'], fare_by_payment_df['avg_fare_amount'], color='skyblue')
plt.title("Average Fare Amount by Payment Type")
plt.xlabel("Payment Type")
plt.ylabel("Average Fare Amount ($)")
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```

```
+------------+---------------+
|payment_type|avg_fare_amount|
+------------+---------------+
|           1|           12.5|
|           2|          10.95|
|           3|           10.4|
|           4|           9.72|
|           5|            3.0|
+------------+---------------+
```

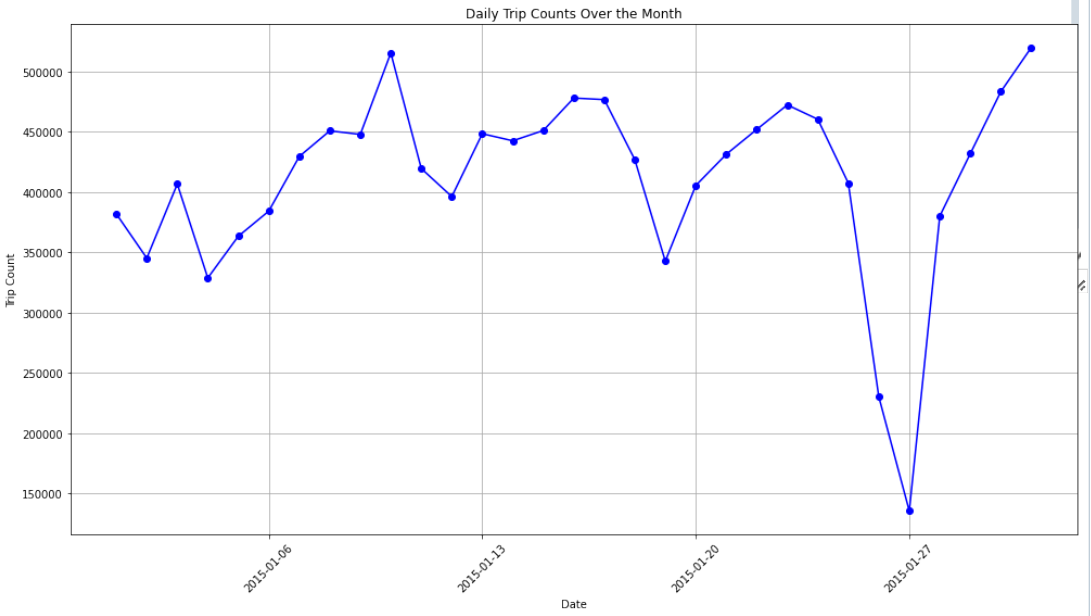


11

```
# Q11: Time Series Analysis of Trips
import matplotlib.dates as mdates
# SQL query to count the number of trips per day
daily_trip_counts = spark.sql("""
    select
        date(tpep_pickup_datetime) as trip_date,
        count(*) as trip_count
    from data
    group by date(tpep_pickup_datetime)
    order by trip_date
    """)

# Show daily trip counts
daily_trip_counts.show()

# Convert to Pandas DataFrame
daily_trip_df = daily_trip_counts.toPandas()

# Plotting time series trend of trips over the month
plt.figure(figsize=(14, 8))
plt.plot(daily_trip_df['trip_date'], daily_trip_df['trip_count'], marker='o', color='b')
plt.title("Daily Trip Counts Over the Month")
plt.xlabel("Date")
plt.ylabel("Trip Count")
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.gca().xaxis.set_major_locator(mdates.WeekdayLocator())
plt.xticks(rotation=45)
plt.grid()
plt.tight_layout()
plt.show()
```

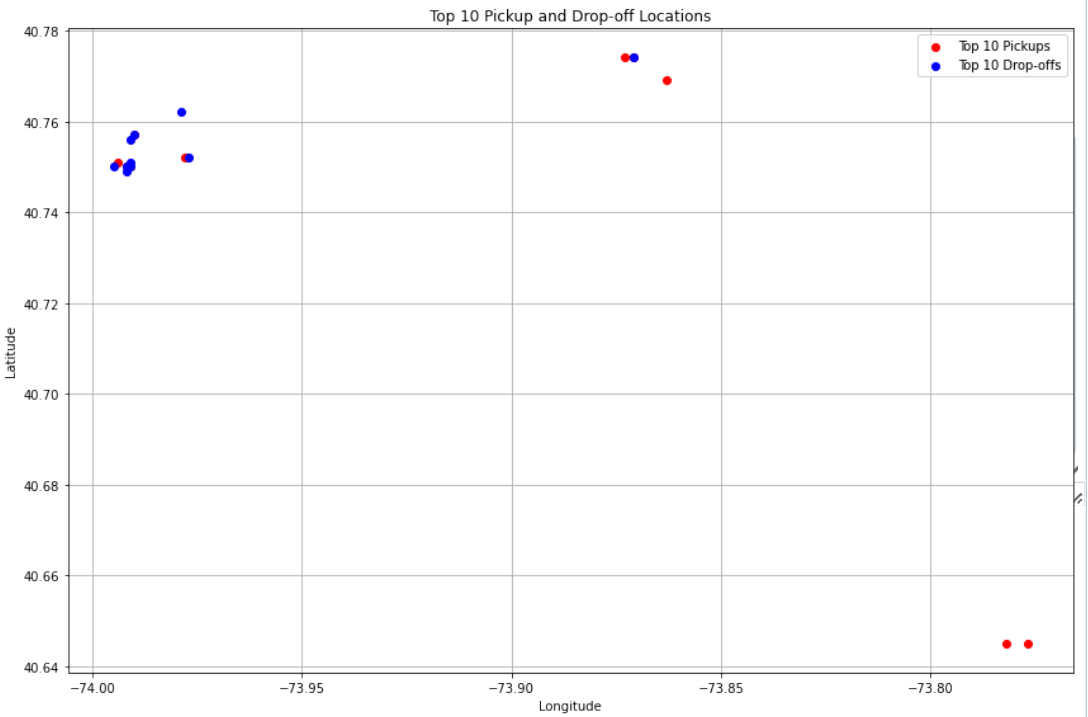
```
|2015-01-03|    406769|
|2015-01-04|    328848|
|2015-01-05|    363454|
|2015-01-06|    384324|
|2015-01-07|    429653|
|2015-01-08|    450920|
```

```
|2015-01-14|    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
```



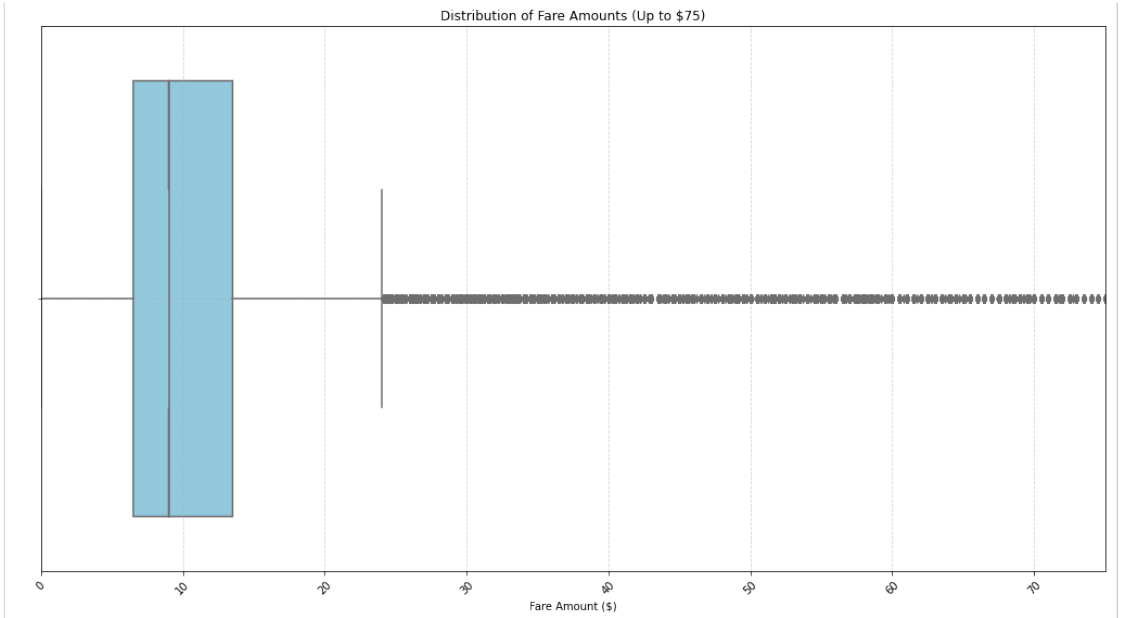
```
# Q12: Location Analysis
# SQL queries for top pickup
top_pickup_locations = spark.sql("""
    select
        round(pickup_latitude, 3) as pickup_lat,
        round(pickup_longitude, 3) as pickup_lon,
        count(*) as pickup_count
    from data
    where pickup_latitude != 0 and pickup_longitude != 0
    group by round(pickup_latitude, 3), round(pickup_longitude, 3)
    order by pickup_count desc
    limit 10
""")
top_pickup_locations.show()

# SQL quries for top drop-off
top_dropoff_locations = spark.sql("""
    select
        round(dropoff_latitude, 3) as dropoff_lat,
        round(dropoff_longitude, 3) as dropoff_lon,
        count(*) as dropoff_count
    from data
    where dropoff_latitude != 0 and dropoff_longitude != 0
    group by round(dropoff_latitude, 3), round(dropoff_longitude, 3)
    order by dropoff_count desc
    limit 10
""")
top_dropoff_locations.show()

# Convert to Pandas DataFrame
pickup_df = top_pickup_locations.toPandas()
dropoff_df = top_dropoff_locations.toPandas()

# Scatter plot for top 10 locations
plt.figure(figsize=(12, 8))
plt.scatter(pickup_df['pickup_lon'] + 0.0001, pickup_df['pickup_lat'] + 0.0001, color='red', label='Top 10
Pickups')
plt.scatter(dropoff_df['dropoff_lon'] + 0.0001, dropoff_df['dropoff_lat'] + 0.0001, color='blue', label='Top
10 Drop-offs')
plt.title("Top 10 Pickup and Drop-off Locations")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.legend()
plt.grid()
plt.tight_layout()
plt.show()
```

```
|    40.769|   -73.863|       39928|
|    40.645|   -73.777|       36112|
|    40.752|   -73.978|       35927|
|    40.757|    -73.99|       35294|
+----------+----------+------------+

+-----------+-----------+-------------+
|dropoff_lat|dropoff_lon|dropoff_count|
+-----------+-----------+-------------+
|      40.75|    -73.991|        58666|
|     40.749|    -73.992|        45459|
```


Top 10 Pickup and Drop-off Locations
Top 10 Pickups
Top 10 Drop-offs
40.78
40.76
40.74
40.72
40.70
40.68
40.66
40.64
Latitude
−74.00
−73.95
−73.90
−73.85
−73.80
Longitude

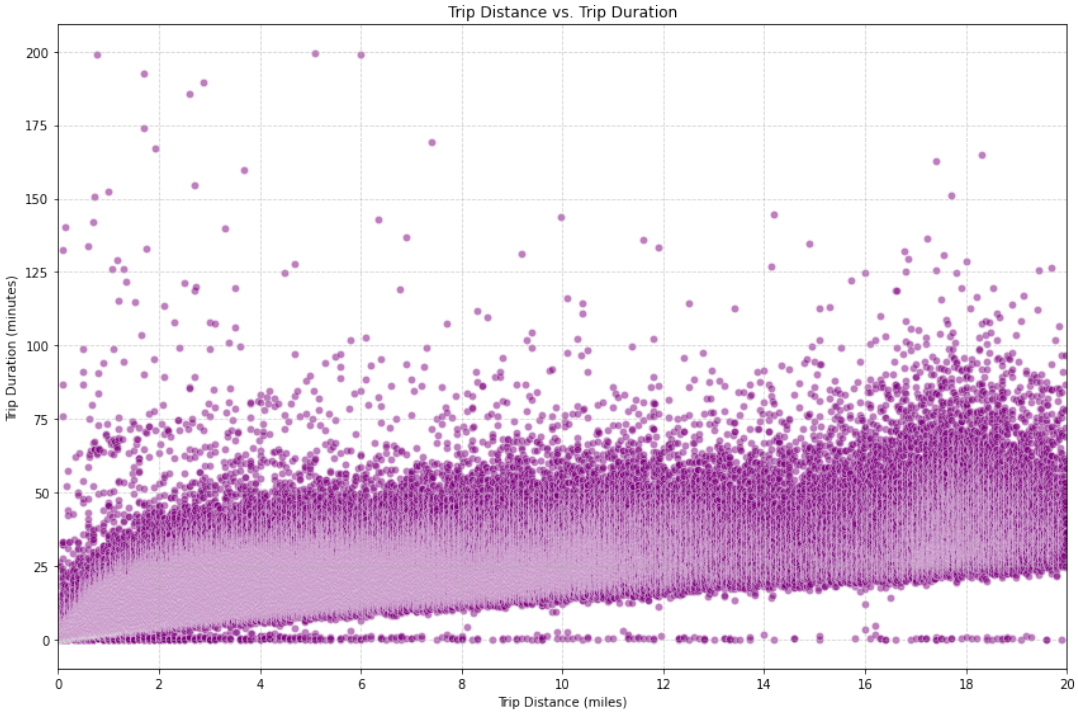
```
# Q13: Fare Amount Distribution Analysis
import pandas as pd
# summary statistics for fare amounts
fare_summary_stats = spark.sql("""
    select
        round(avg(fare_amount), 2) as avg_fare,
        round(stddev(fare_amount), 2) as stddev_fare,
        min(fare_amount) as min_fare,
        max(fare_amount) as max_fare
    from data
    where fare_amount > 0
""")

# display summary statistics
fare_summary_stats.show()

fare_data_full = spark.sql("""
    select fare_amount
    from data
    where fare_amount > 0
""").limit(1000000).toPandas()

# Ensure fare_amount is numeric
fare_data_full['fare_amount'] = pd.to_numeric(fare_data_full['fare_amount'], errors='coerce')
fare_data_full.dropna(subset=['fare_amount'], inplace=True)

# boxplot for fare distribution
plt.figure(figsize=(14, 8))
sns.boxplot(x=fare_data_full['fare_amount'], notch=True, color='skyblue')
plt.xlim(0, 75)
plt.title("Distribution of Fare Amounts (Up to $75)")
plt.xlabel("Fare Amount ($)")
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

```
+--------+-----------+--------+--------+
|avg_fare|stddev_fare|min_fare|max_fare|
+--------+-----------+--------+--------+
|   11.92|      10.29|    0.01|  4008.0|
+--------+-----------+--------+--------+
```


Distribution of Fare Amounts (Up to $75)
0
10
20
30
40
50
60
70
Fare Amount ($)

```
# Q14: Distance vs. Duration Analysis
# Find average duration for different distance ranges
distance_bins = spark.sql("""
    select
        case
            when trip_distance < 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 and trip_distance < 10 then '5-10 miles'
            else '10+ miles'
        end as distance_range,
        round(avg((unix_timestamp(tpep_dropoff_datetime) - unix_timestamp(tpep_pickup_datetime)) / 60), 2) as
avg_trip_duration
    from data
    where trip_distance > 0
    and tpep_dropoff_datetime > tpep_pickup_datetime
    group by distance_range
    order by distance_range
""")

# Show the table summarizing average durations
distance_bins.show()

# collect data
distance_duration_data_full = spark.sql("""
    select
        trip_distance,
        (unix_timestamp(tpep_dropoff_datetime) - unix_timestamp(tpep_pickup_datetime)) / 60 as trip_duration
    from data
    where trip_distance > 0
    and tpep_dropoff_datetime > tpep_pickup_datetime
    and trip_distance < 20
    and (unix_timestamp(tpep_dropoff_datetime) - unix_timestamp(tpep_pickup_datetime)) / 60 < 200
""").limit(1000000).toPandas()

# sort by trip distance
distance_duration_data_full = distance_duration_data_full.sort_values(by='trip_distance')

# Scatter plot
plt.figure(figsize=(12, 8))
sns.scatterplot(x='trip_distance', y='trip_duration', data=distance_duration_data_full, alpha=0.5,
color='purple', edgecolor='w')
plt.xlim(0, 20)
plt.xticks(range(0, 21, 2))
plt.title("Trip Distance vs. Trip Duration")
plt.xlabel("Trip Distance (miles)")
plt.ylabel("Trip Duration (minutes)")
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

```
+--------------+-----------------+
|distance_range|avg_trip_duration|
+--------------+-----------------+
|      0-1 mile|             6.56|
|     1-3 miles|            12.71|
|     10+ miles|             37.2|
|     3-5 miles|            19.69|
|    5-10 miles|            24.39|
+--------------+-----------------+
```



15

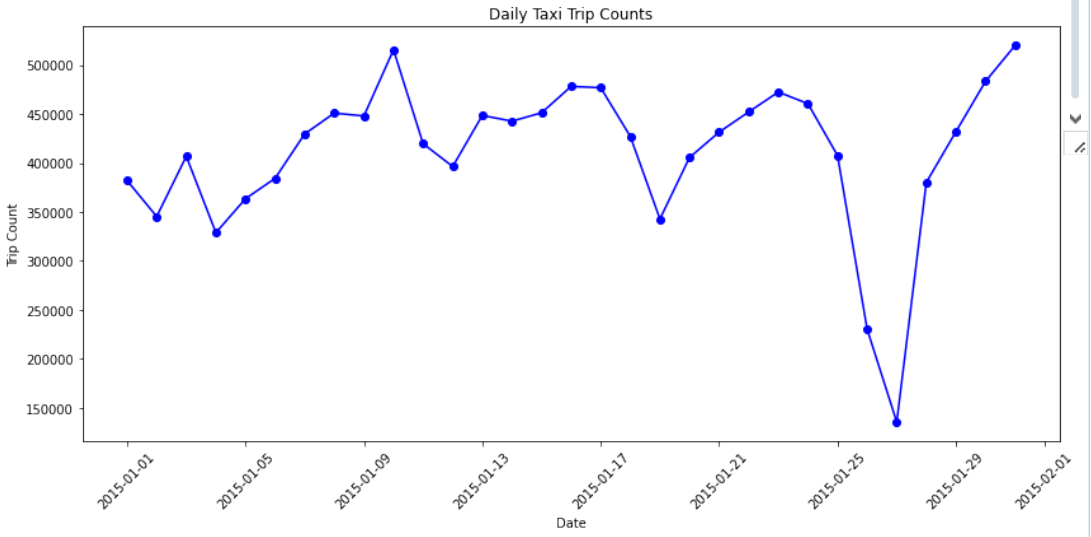
```
data2015_1c.createOrReplaceTempView("yellow_tripdata")
```

16

```
# Q15: Count daily trips
import matplotlib.pyplot as plt
daily_trip_counts = spark.sql("""
    SELECT DATE(tpep_pickup_datetime) AS trip_date, COUNT(*) AS trip_count
    FROM yellow_tripdata
    GROUP BY DATE(tpep_pickup_datetime)
    ORDER BY trip_date
""")
daily_trip_counts.show()
daily_trip_counts_pd = daily_trip_counts.toPandas()

plt.figure(figsize=(12, 6))
plt.plot(daily_trip_counts_pd['trip_date'], daily_trip_counts_pd['trip_count'], marker='o', color='b')
plt.title('Daily Taxi Trip Counts')
plt.xlabel('Date')
plt.ylabel('Trip Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

#The data showcases that there is a huge drop in number of taxi's on January 29th indicating that there could
have been an external event interfering with the daily taxi trips
```

```
|2015-01-03|    406769|
```

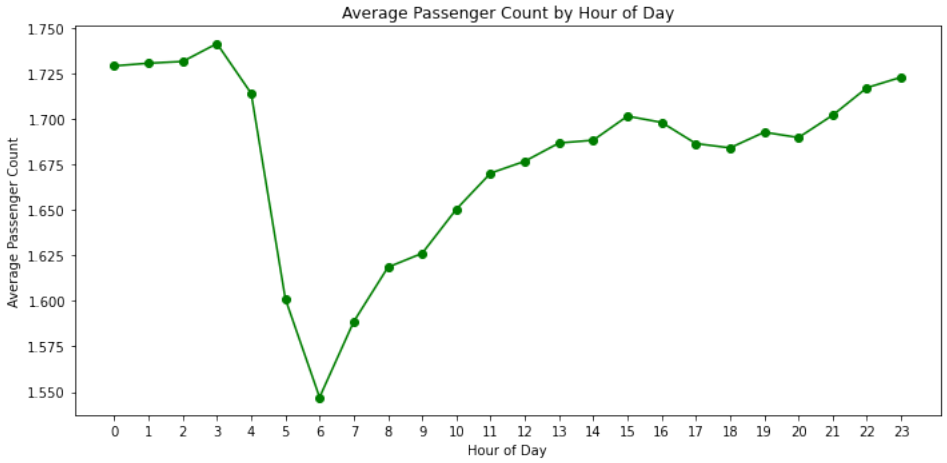
```
|2015-01-09|    447947|
|2015-01-10|    515540|
|2015-01-11|    419629|
|2015-01-12|    396367|
|2015-01-13|    448517|
|2015-01-14|    442656|
|2015-01-15|    451186|
|2015-01-16|    478124|
|2015-01-17|    476827|
|2015-01-18|    427042|
|2015-01-19|    342795|
|2015-01-20|    405581|
+----------+----------+
only showing top 20 rows
```



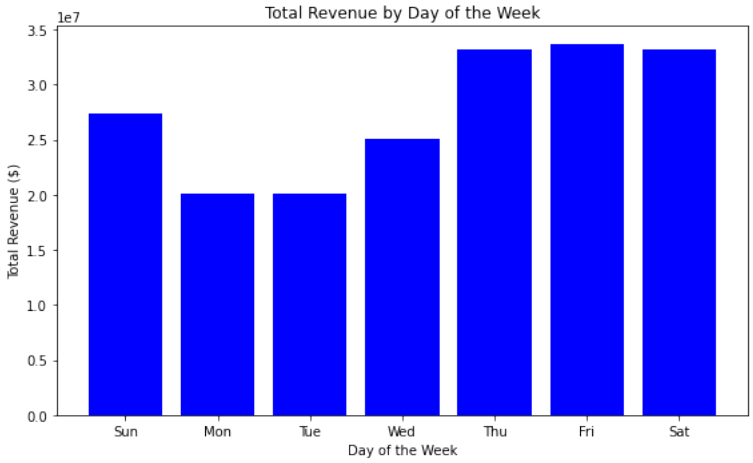
17

```
# Q17: Calculate average passenger count by hour
day_pass_count = spark.sql("""
    SELECT HOUR(tpep_pickup_datetime) AS hour_of_day, AVG(passenger_count) AS avg_passenger_count
    FROM yellow_tripdata
    GROUP BY HOUR(tpep_pickup_datetime)
    ORDER BY hour_of_day
""")

day_pass_count.show()
pass_count = day_pass_count.toPandas()


plt.figure(figsize=(10, 5))
plt.plot(pass_count['hour_of_day'], pass_count['avg_passenger_count'], marker='o', color='g')
plt.title('Average Passenger Count by Hour of Day')
plt.xlabel('Hour of Day')
plt.ylabel('Average Passenger Count')
plt.xticks(range(0, 24))
plt.tight_layout()
plt.show()

#There is a sharp dip at about hour 6 in the morning suggesting that the people who use the taxi to commute to
work often travel by themselves.As well as from Hour 1 to 4 there is a high passenger count indicating that
people often travel in groups in the early hours of the day.
```

```
|          4| 1.7139616530909954|
|          5| 1.6011127066707471|
|          6| 1.5471084539308264|
|          7|  1.588691307464807|
|          8| 1.6185239639588325|
|          9| 1.6261546736915422|
|         10| 1.6503897375567524|
|         11| 1.6701547684508402|
|         12| 1.6766826828805341|
|         13|  1.686695920464075|
|         14| 1.6882333389488637|
|         15| 1.7014373627999901|
|         16| 1.6979940964068554|
|         17| 1.6863888515079473|
|         18| 1.6840256282305741|
|         19|  1.692634402593048|
+-----------+-------------------+
only showing top 20 rows
```




```
+-----------+-------------------+
|day_of_week|      total_revenue|
+-----------+-------------------+
|          1| 2.740322290001997E7|
|          2|2.00411897700188 46E7|
|          3| 2.014959691001926E7|
|          4| 2.510422000002709E7|
|          5| 3.314028551001946E7|
|          6| 3.364367529002028E7|
|          7| 3.313324516002973E7|
+-----------+-------------------+
```


Total Revenue by Day of the Week
1e7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Total Revenue ($)
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Day of the Week