Name: Akshata
ID: 230842030
Course: Big Data science

I'm using the skeleton structure from starterkit.py provided in coursework for all the tasks and I have provided snapshots of the results in concise way to make report more readable, I have attached full screenshots in the zip file.

Task1:

<u>(Task1)Q1</u>: For loading both the csv files rideshare_data.csv and taxi_zone_lookup.csv I'm simply using the readily available spark.read.csv function provided by spark and I have mentioned the S3 bucket path with file details, I have given header=true so that it depicts first line in the file as header.

<u>(Task1)Q2:</u> In this section, since we want to join two dataframes that we read earlier, I have taken advantage of join function which comes from the Spark DataFrame API in PySpark. It's a method that allows us to join two DataFrames based on a specified condition. As per the requirement I have first joined using pickup_location and LocationID field of taxi_zone_lookup table and later with dropoff_location field, also I have dropped LocationID field from the joined data frame as we don't need it anymore and it aligns with the schema that we need for our joined data frame. Along with each join I have renamed the columns accordingly, earlier I tried to rename the columns separately after both join operations and I was facing issue of columns being duplicated, so I got this idea of renaming them along with their join operations.

<u>(Task1)Q3:</u> 'withColumn' function is a powerful tool for adding new columns or transforming existing columns in Spark DataFrames, allowing for flexible data manipulation and transformation, so using this on joined_df 's 'date' column, I have used from_unixtime(col("date").cast("bigint"), "yyyy-MM-dd") expression which helps us to determine the values of the new "date" column. Where col("date") retrieves the values from the existing "date" column. cast("bigint") casts the values of the "date" column to bigint and from_unixtime() function converts Unix timestamps to a string representing the date in the specified format ("yyyy-MM-dd"). I have displayed sample data to show the changes as shown below, I have highlighted the changes

```
2024-04-01 13:48:43,151 INFO codegen.CodeGenerator: Code generated in 38.305792 ms
```

business	pickup_location	dropoff_location	trip_length	request_to_pickup	total_ride_time	on_scene_to_pickup	on_scene_to_dropoff	time_of_day	date	passenger_fare	driver_total_pay	rideshare_profit	hourly_rate	dollars_per_mile	Pickup_Borough	Pickup_Zone	Pickup_service_zone	Dropoff_Borough	Dropoff_Zone	Dropoff_service_zone
Uber	151	244	4.98	226.0	761.0	19.0	780.0	morning	2023-05-22	22.82	13.69	9.13	63.18	2.75	Manhattan	Manhattan Valley	Yellow Zone	Manhattan	Washington Height...	Boro Zone
Uber	244	78	4.35	197.0	1423.0	120.0	1543.0	morning	2023-05-22	24.27	19.1	5.17	44.56	4.39	Manhattan	Washington Height...	Boro Zone	Bronx	East Tremont	Boro Zone
Uber	151	138	8.82	171.0	1527.0	12.0	1539.0	morning	2023-05-22	47.67	25.94	21.73	60.68	2.94	Manhattan	Manhattan Valley	Yellow Zone	Queens	LaGuardia Airport	Airports
Uber	138	151	8.72	260.0	1761.0	44.0	1805.0	morning	2023-05-22	45.67	28.01	17.66	55.86	3.21	Queens	LaGuardia Airport	Airports	Manhattan	Manhattan Valley	Yellow Zone
Uber	36	129	5.05	208.0	1762.0	37.0	1799.0	morning	2023-05-22	33.49	26.47	7.02	52.97	5.24	Brooklyn	Bushwick North	Boro Zone	Queens	Jackson Heights	Boro Zone

```
only showing top 5 rows
```

<u>(Task1)Q4:</u> I acquired the correct number rows and schema with my code, below is the screenshot of my result:

Name: Akshata
ID: 230842030
Course: Big Data science

```
2024-04-01 13:48:42,196 INFO scheduler.TaskSchedulerImpl: Ki
2024-04-01 13:48:42,197 INFO scheduler.DAGScheduler: Job 3 f
Number of rows: 69725864
root
 |-- business: string (nullable = true)
 |-- pickup_location: string (nullable = true)
 |-- dropoff_location: string (nullable = true)
 |-- trip_length: string (nullable = true)
 |-- request_to_pickup: string (nullable = true)
 |-- total_ride_time: string (nullable = true)
 |-- on_scene_to_pickup: string (nullable = true)
 |-- on_scene_to_dropoff: string (nullable = true)
 |-- time_of_day: string (nullable = true)
 |-- date: string (nullable = true)
 |-- passenger_fare: string (nullable = true)
 |-- driver_total_pay: string (nullable = true)
 |-- rideshare_profit: string (nullable = true)
 |-- hourly_rate: string (nullable = true)
 |-- dollars_per_mile: string (nullable = true)
 |-- Pickup_Borough: string (nullable = true)
 |-- Pickup_Zone: string (nullable = true)
 |-- Pickup_service_zone: string (nullable = true)
 |-- Dropoff_Borough: string (nullable = true)
 |-- Dropoff_Zone: string (nullable = true)
 |-- Dropoff_service_zone: string (nullable = true)

2024-04-01 13:48:42,367 INFO datasources.FileSourceStrategy:
2024-04-01 13:48:42,367 INFO datasources.FileSourceStrategy:
```

Task3: Querying is one of my strong pursuits so I have adapted my code to use spark.sql function, which helps me query on the joined dataframe for required result. I have created temporary view of joined_df using createOrReplaceTempView() so that I can run spark SQL queries against the joined_df DataFrame.

(Task3)Q1: First I have created SQL query to calculate the number of trips (trip_count) for each pickup borough (Pickup_Borough) in each month (Month). Basically, it groups the data by pickup borough and month, and then sorts the results by month and trip count in descending order. Then I have stored this result in top_pickup_boroughs_each_month DataFrame. Then I have created another temporary view named "top_pickup_boroughs_each_month" which I'm using in the subsequent query to rank the trip counts by month.

So now the subsequent query ranks the pickup boroughs (Pickup_Borough) by the number of trips (trip_count) in each month (Month). Here I have used the ROW_NUMBER() window function to assign a rank (rn) to each pickup borough within each month based on the trip count. then selected only the top 5 pickup boroughs for each month and ordered the results by month and trip count in descending order. Which again I have stored in the top_5_pickup_boroughs_each_month DataFrame.

Finally, I have used show() function to display these top 5 pickup boroughs for each month, showing upto 100 rows(so that none of the data is missed because by default terminal limits and shows only 20 rows) and without truncating the data in the DataFrame for which I have passed 'truncate=False' in the function. See below snapshot

```
+--------------+-----+----------+
|Pickup_Borough|Month|trip_count|
+--------------+-----+----------+
|Manhattan     |1    |5854818   |
|Brooklyn      |1    |3360373   |
|Queens        |1    |2589034   |
|Bronx         |1    |1607789   |
|Staten Island |1    |173354    |
|Manhattan     |2    |5808244   |
|Brooklyn      |2    |3283003   |
|Queens        |2    |2447213   |
|Bronx         |2    |1581889   |
|Staten Island |2    |166328    |
|Manhattan     |3    |6194298   |
|Brooklyn      |3    |3632776   |
|Queens        |3    |2757895   |
|Bronx         |3    |1785166   |
|Staten Island |3    |191935    |
|Manhattan     |4    |6002714   |
|Brooklyn      |4    |3481220   |
|Queens        |4    |2666671   |
|Bronx         |4    |1677435   |
|Staten Island |4    |175356    |
|Manhattan     |5    |5965594   |
|Brooklyn      |5    |3586009   |
|Queens        |5    |2826599   |
|Bronx         |5    |1717137   |
|Staten Island |5    |189924    |
+--------------+-----+----------+

2024-04-05 05:01:00,943 INFO datasourc
2024-04-05 05:01:00,944 INFO datasourc
```

Name: Akshata
ID: 230842030
Course: Big Data science

(Task3)Q2: as this question is similar to previous one I have used the similar approach, First I have calculated the number of trips (trip_count) for each drop-off borough (Dropoff_Borough) in each month (Month). It groups the data by drop-off borough and month, and then sorts the results by month and trip count in descending order. Then I have stored result in top_dropoff_boroughs_each_month DataFrame. I created a temporary view named "top_dropoff_boroughs_each_month" for this DataFrame. Then used this view in the subsequent query to rank the trip counts by month.

In the subsequent query I rank the drop-off boroughs (Dropoff_Borough) by the number of trips (trip_count) in each month (Month). Similar to earlier I used the ROW_NUMBER() window function to assign a rank (rn) to each drop-off borough within each month based on trip count. So that I can select only the top 5 drop-off boroughs for each month and order the results by month and trip count in descending order. Which I have stored in the top_5_dropoff_boroughs_each_month DataFrame. And the displayed the top 5 drop-off boroughs for each month, showing up to 100 rows(so that none of the data is missed as by default terminal limits and shows only 20 rows) without truncating the data in the DataFrame. See the screenshot of result below:

2024-04-01 23:45:37,950 INFO Scheduler

Dropoff_Borough	Month	trip_count
Manhattan	1	5444345
Brooklyn	1	3337415
Queens	1	2480080
Bronx	1	1525137
Unknown	1	535610
Manhattan	2	5381696
Brooklyn	2	3251795
Queens	2	2390783
Bronx	2	1511014
Unknown	2	497525
Manhattan	3	5671301
Brooklyn	3	3608960
Queens	3	2713748
Bronx	3	1706802
Unknown	3	566798
Manhattan	4	5530417
Brooklyn	4	3448225
Queens	4	2605086
Bronx	4	1596505
Unknown	4	551857
Manhattan	5	5428986
Brooklyn	5	3560322
Queens	5	2780011
Bronx	5	1639180
Unknown	5	578549

3.02 / 12288.00 MB

(Task3)Q3: In this case, I have calculated the total profit earned by rideshare drivers for each route, where a route is defined by the combination of pickup borough (Pickup_Borough) and drop-off borough (Dropoff_Borough). So I have grouped the data by pickup and drop-off boroughs, calculated the sum of driver total pay (driver_total_pay) for each route, ordered the results by total profit in descending order, and limited the output to the top 30 earning routes. Then stored the result is in top_30_earnest_routes DataFrame. Which I have displayed using show() function on dataframe, showing up to 100 rows without truncating the data in the DataFrame. Below is the screenshot of the result:

Name: Akshata
ID: 230842030
Course: Big Data science

```
2024-04-05 12:05:44,967 INFO scheduler.DAGScheduler: Jc
2024-04-05 12:05:44,995 INFO codegen.CodeGenerator: Coc
```

Route	total_profit
Manhattan to Queens	1.0173842820999995E8
Queens to Manhattan	8.603540026000004E7
Manhattan to Unknown	8.010710241999993E7
Manhattan to Brooklyn	6.799047559000002E7
Brooklyn to Manhattan	6.3176161049999975E7
Brooklyn to Queens	5.045416243000007E7
Queens to Brooklyn	4.729286536000015E7
Queens to Unknown	4.6292999900000036E7
Bronx to Manhattan	3.248632517000009E7
Manhattan to Bronx	3.1978763450000063E7
Manhattan to EWR	2.375088861999999E7
Brooklyn to Unknown	1.0848827569999998E7
Bronx to Unknown	1.0464800209999992E7
Bronx to Queens	1.0292266499999998E7
Queens to Bronx	1.018289873E7
Brooklyn to Bronx	5848822.5600000005
Bronx to Brooklyn	5629874.41
Brooklyn to EWR	3292761.71
Brooklyn to Staten Island	2417853.82
Staten Island to Brooklyn	2265856.460000001
Manhattan to Staten Island	2223727.3700000006
Staten Island to Manhattan	1612227.72
Queens to EWR	1192758.6600000001
Staten Island to Unknown	891285.8100000002
Queens to Staten Island	865603.38
Staten Island to Queens	807064.1199999999
Staten Island to EWR	547553.2200000001
Bronx to EWR	381787.07999999996
Bronx to Staten Island	207356.61
Staten Island to Bronx	203073.06000000003

(Task3)Q4: we can draw following insights from above three results:

-number of trips between different borough pairs vary. For example, there are more trips between Manhattan and Brooklyn compared to other borough pairs.

-The profitability of each route is also varying. Some routes generate higher profits than others, likes Manhattan to Queens and Queens to Manhattan.

- and profitability varies for different boroughs, with some boroughs being more profitable to operate in than others.

Looking at these findings, we can draw following strategies:

1. Allocating resources to boroughs and routes that are more profitable and have higher demand and focusing marketing efforts on routes with high trip counts. Like Manhattan, Queens, Brooklyn seem to have most trip counts as well as have high profits

2. Prioritizing the routes with higher profit margins, such as those identified from the total profit by route data. Ex: Manhattan to Queens

Task4:

(Task4)Q1: Here we want to calculate the average driver total pay for different time of day periods. So to crack this, I have written spark sql query where I select the time_of_day column and calculate the average of driver_total_pay for each time of day group which I have grouped by time_of_day and ORDER BY clause sorts the results by average driver total pay in descending order. Below is the screenshot of result

Name: Akshata
ID: 230842030
Course: Big Data science

```
2024-04-02 13:48:05,024 INFO codegen.Code
+-----------+----------------------+
|time_of_day|average_drive_total_pay|
+-----------+----------------------+
|afternoon  |21.21242875659354     |
|night      |20.08743800359271     |
|evening    |19.77742770239839     |
|morning    |19.633332793944835    |
+-----------+----------------------+

2024-04-02 13:48:05,168 INFO datasources.
2024-04-02 13:48:05,169 INFO datasources
```

(Task4)Q2: To calculate average trip length for different time of day periods, I have selected the time_of_day column and calculated the average of trip_length for each time of day group. Then I have grouped it by time_of_day. Then by using ORDER BY clause I have sorted the results by average trip length in descending order. Below is the screenshot of result

```
2024-04-02 14:07:19,726 INFO scheduler
2024-04-02 14:07:19,727 INFO scheduler
+-----------+------------------+
|time_of_day|average_trip_length|
+-----------+------------------+
|night      |5.32398480196174  |
|morning    |4.927371866442786 |
|afternoon  |4.861410525661208 |
|evening    |4.484750367447518 |
+-----------+------------------+

2024-04-02 14:07:20,038 INFO datasourc
```

(Task4)Q3: In this query I have joined the results of the previous two questions to calculate the average earning per mile for each time of day period. I selected time_of_day column and calculated the average of driver_total_pay and trip_length for each time of day group. Then using AVG(driver_total_pay) / AVG(trip_length) expression I calculated the average earning per mile. I have grouped the results by time_of_day. Then used ORDER BY clause to sort the results by average earning per mile in descending order. Below is the screenshot of the result.

```
2024-04-02 14:31:35,656 INFO scheduler.DA
+-----------+-----------------------+
|time_of_day|average_earning_per_mile|
+-----------+-----------------------+
|evening    |4.409928330894958      |
|afternoon  |4.363430869420022      |
|morning    |3.984544565766398      |
|night      |3.7730081416068377     |
+-----------+-----------------------+

2024-04-02 14:31:35,680 INFO server.Abstr
```

(Task4)Q4: From looking at above three results, we can observe certain patterns:

-From first result we can see that afternoon has the highest average driver_total_pay, followed by night, evening, and morning.

- Then from second result we can see that average_trip_length by time of day, afternoon and evening have the longest average trip lengths but morning and night have shorter trip lengths.

- from third result average_earning_per_mile by time of day, Evening and afternoon yield the highest average earnings per trip, followed by morning and night.

With these findings, we can draw following strategies:

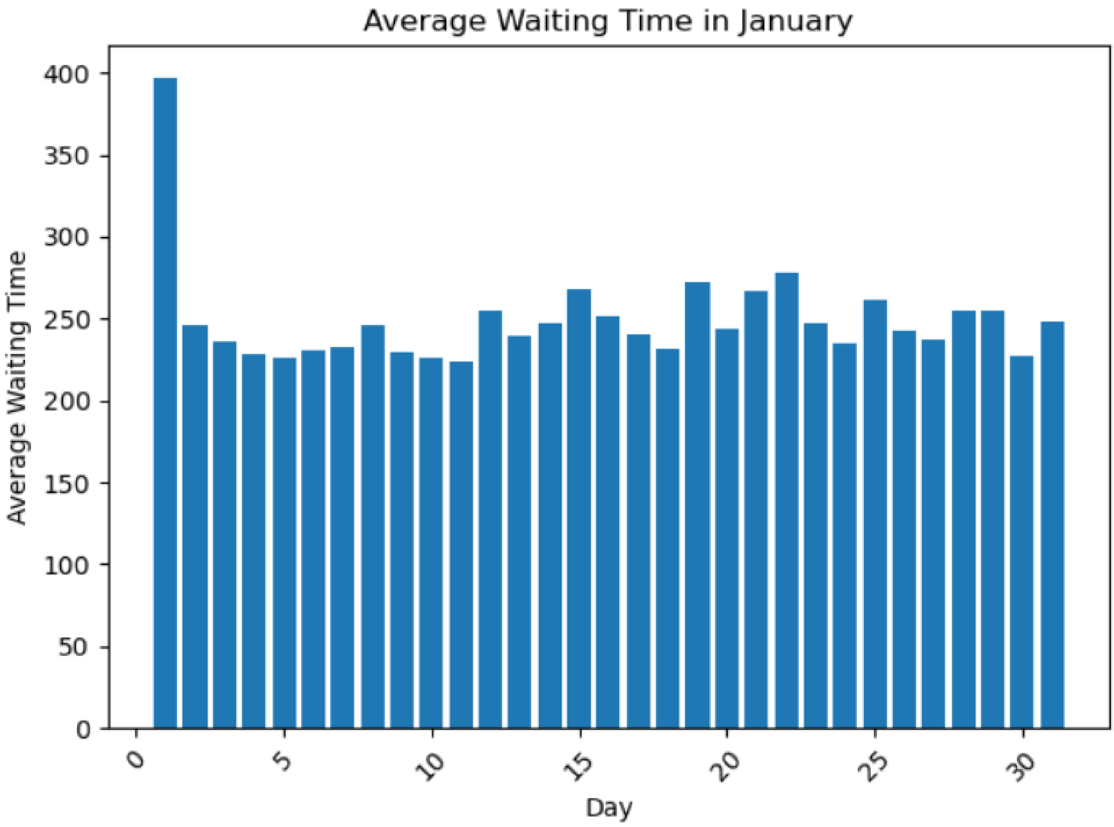
1. We can allocate more resources (such as drivers) during peak times, particularly in the afternoon and evening when both trip length and earnings per trip are high.
2. We can Implement dynamic pricing strategies to maximize earnings during peak times, specially in afternoon and evening.
3. We can target the market and promote encouraging more rides during off-peak times, like morning and night, to balance supply and demand.

Task 5:

(Task5)Q1: I have read the rideshare_data.csv from the S3 bucket location (s3_data_repository_bucket) and stored it in a DataFrame named rideshare_data. Later in the code I Filtered the rideshare_data DataFrame to include only data for the month of January (january_data). Then, I have grouped the data by day of the month using groupBy() function, calculated the average waiting time for each day using agg(), and ordered the results by day of the month. Then stored this result in a DataFrame named average_waiting_time_by_day. Then I wrote this output average_waiting_time_by_day DataFrame to a CSV file in my S3 bucket location (s3_bucket).

```
jovyan@jupyter-ec23864:~/teaching_material/ECS765P$ ccc method bucket ls
                           PRE average_waiting_time_by_day05-04-2024_11:47:22/
                           PRE nasdaq_03-04-2024_16:45:08/
                           PRE olympic03-04-2024_17:54:38/
                           PRE sorted_counts_03-04-2024_13:36:47/
                           PRE sorted_counts_28-03-2024_21:53:31/
2024-03-28 21:42:56    3022566 sherlock.txt
jovyan@jupyter-ec23864:~/teaching_material/ECS765P$ ccc method bucket ls average_waiting_time_by_day05-04-2024_11:47:22/
2024-04-05 13:54:29          0 _SUCCESS
2024-04-05 13:54:28        707 part-00000-31a3dba6-9d15-47ae-ba3e-c6f8da5b22e3-c000.csv
```

Then I renamed above highlighted csv file created as ‘average_waiting_time_january.csv’(I have provided this csv file in zip file) and wrote code in jupyter notebook to import it using pandas read.csv and then using pyplot from matplotlib I have created the histogram with days on x axis and average waiting time on y axis, which looks like below.



(Task5)Q2: Average waiting time exceeds 300 seconds on Januaray 1st

(Task5)Q3: The average waiting time on 1st January is more as the occasion is New year. So there's high demand.

Task6:

(Task6)Q1: In this part I'm calculating the trip counts for different pickup boroughs at different times of the day. In the spark sql I have grouped the data by pickup borough and time of day, calculating the count of trips, and filtered out the trip counts greater than 0 and less than 1000. Finally, I have ordered the results by pickup borough and time of day

Name: Akshata
ID: 230842030
Course: Big Data science

```
2024-04-04 01:13:55,860 INFO codegen.CodeGener
2024-04-04 01:13:55,873 INFO codegen.CodeGener
+-------------+-----------+----------+
|Pickup_Borough|time_of_day|trip_count|
+-------------+-----------+----------+
|EWR          |afternoon  |2         |
|EWR          |morning    |5         |
|EWR          |night      |3         |
|Unknown      |afternoon  |908       |
|Unknown      |evening    |488       |
|Unknown      |morning    |892       |
|Unknown      |night      |792       |
+-------------+-----------+----------+

2024-04-04 01:13:55,886 INFO server.AbstractCo
2024-04-04 01:13:55,887 INFO ui.SparkUI: Stopp
```

(Task6)Q2: for this task I have calculated the number of trips for each pickup borough specifically during the evening time. I have filtered the data for the 'evening' time of day, grouped the data using groupby fuction on pickup borough, and calculated the count of trips, and ordered the results by pickup borough.

```
2024-04-03 23:37:40,225 INFO codegen.CodeGene
+--------------+-----------+----------+
|Pickup_Borough|time_of_day|trip_count|
+--------------+-----------+----------+
|Bronx         |evening    |1380355   |
|Brooklyn      |evening    |3075616   |
|Manhattan     |evening    |5724796   |
|Queens        |evening    |2223003   |
|Staten Island |evening    |151276    |
|Unknown       |evening    |488       |
+--------------+-----------+----------+
```

(Task6)Q3: In this case I have filtered the joined DataFrame to get trips that started in Brooklyn and ended in Staten Island. I used the filter method to apply conditions on pickup and dropoff boroughs, then selected relevant columns ('Pickup_Borough', 'Dropoff_Borough', 'Pickup_Zone') using the select method, and limited the result to 10 rows using the limit method as the requirement was to get 10 samples. Finally, I displayed the result using the show() method.

```
2024-04-04 00:12:34,236 INFO codegen.CodeGenerator: Code gener
+--------------+---------------+-------------------------+
|Pickup_Borough|Dropoff_Borough|Pickup_Zone              |
+--------------+---------------+-------------------------+
|Brooklyn      |Staten Island  |DUMBO/Vinegar Hill       |
|Brooklyn      |Staten Island  |Dyker Heights            |
|Brooklyn      |Staten Island  |Bensonhurst East         |
|Brooklyn      |Staten Island  |Williamsburg (South Side)|
|Brooklyn      |Staten Island  |Bay Ridge                |
|Brooklyn      |Staten Island  |Bay Ridge                |
|Brooklyn      |Staten Island  |Flatbush/Ditmas Park     |
|Brooklyn      |Staten Island  |Bay Ridge                |
|Brooklyn      |Staten Island  |Bath Beach               |
|Brooklyn      |Staten Island  |Bay Ridge                |
+--------------+---------------+-------------------------+

2024-04-04 00:12:34,374 INFO storage.BlockManagerInfo: Removed
iB, free: 2004.0 MiB)
```

Then to calculate the number of trips, I filtered the joined DataFrame to count the total number of trips from Brooklyn to Staten Island. I used the filter method to apply conditions on pickup and dropoff boroughs and then used count() method to count the number of rows. Finally, I have printed the total count of such trips. I got count value 69437 as shown in below snapshot.

Name: Akshata
ID: 230842030
Course: Big Data science

```
2024-04-04 00:19:13,603 INFO scheduler.DAGScheduler: Job 13 finished
Total number of trips from Brooklyn to Staten Island: 69437
2024-04-04 00:19:13,624 INFO server.AbstractConnector: Stopped Spark
```

Task7:

(Task7)Q1: I have written this spark sql query to calculate the route counts where a route is defined by the combination of pickup and dropoff zones. It sums the occurrences of 'Uber' and 'Lyft' for each route and calculates the total count in the last column as per the requirement. The query filters out routes where the pickup and dropoff zones are the same so that we don't pick up same zones for both pickup and drop off(earlier I hadn't done this which gave me duplicated pickup and drop off zones) and I have ordered the results by the total count in descending order. Then I have stored it in DataFrame top_routes and then displayed the top 10 routes with the highest total counts without truncating the data in the DataFrame(as per the requirement). I have provided screenshot of the result below:

```
2024-04-04 00:45:17,402 INFO codegen.CodeGenerator: Code generated in 21.802067 ms
2024-04-04 00:45:17,422 INFO codegen.CodeGenerator: Code generated in 12.270643 ms
+-----------------------------------------------+----------+----------+-----------+
|Route                                          |uber_count|lyft_count|total_count|
+-----------------------------------------------+----------+----------+-----------+
|JFK Airport to NA                              |253211    |46        |253257     |
|LaGuardia Airport to NA                        |151521    |41        |151562     |
|South Ozone Park to JFK Airport                |107392    |1770      |109162     |
|Times Sq/Theatre District to NA                |83639     |4         |83643      |
|Midtown Center to NA                           |82044     |9         |82053      |
|Bushwick North to Bushwick South               |63979     |31        |64010      |
|Bushwick South to Bushwick North               |60272     |55        |60327      |
|Williamsburg (North Side) to Greenpoint        |58997     |2         |58999      |
|LaGuardia Airport to Times Sq/Theatre District|57112     |8         |57120      |
|Greenpoint to Williamsburg (North Side)        |56777     |1         |56778      |
+-----------------------------------------------+----------+----------+-----------+
only showing top 10 rows

2024-04-04 00:45:17,441 INFO server.AbstractConnector: Stopped Spark@16aeb414{HTTP/1.
```