ECS765P - Big Data Processing

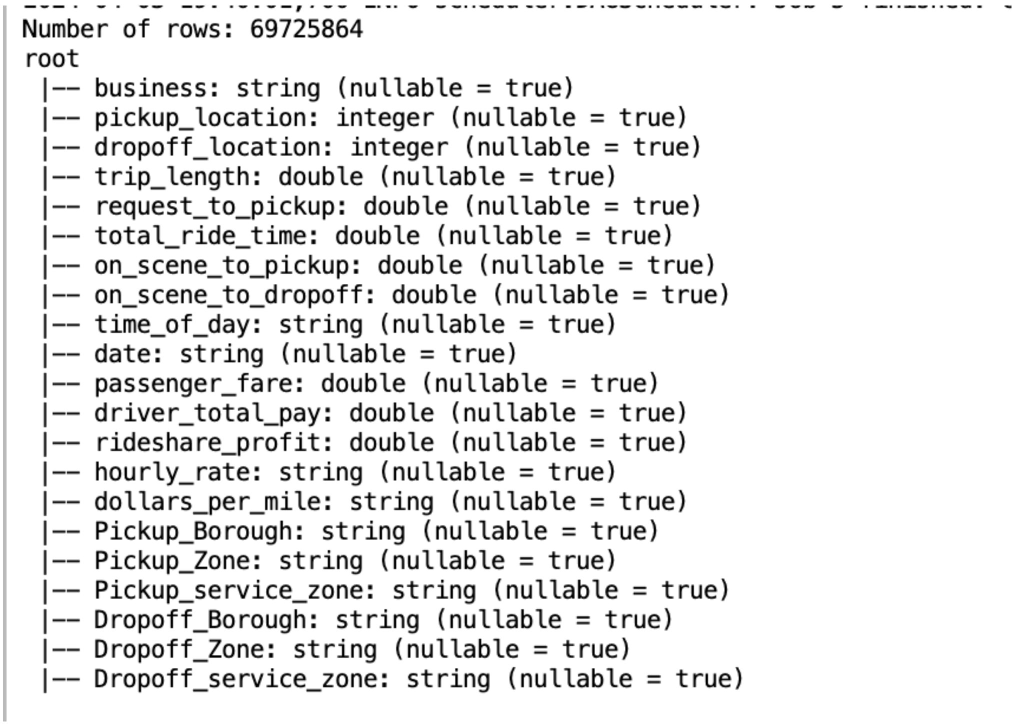
Student ID - 230669703

Task 1

Steps and API Implementation

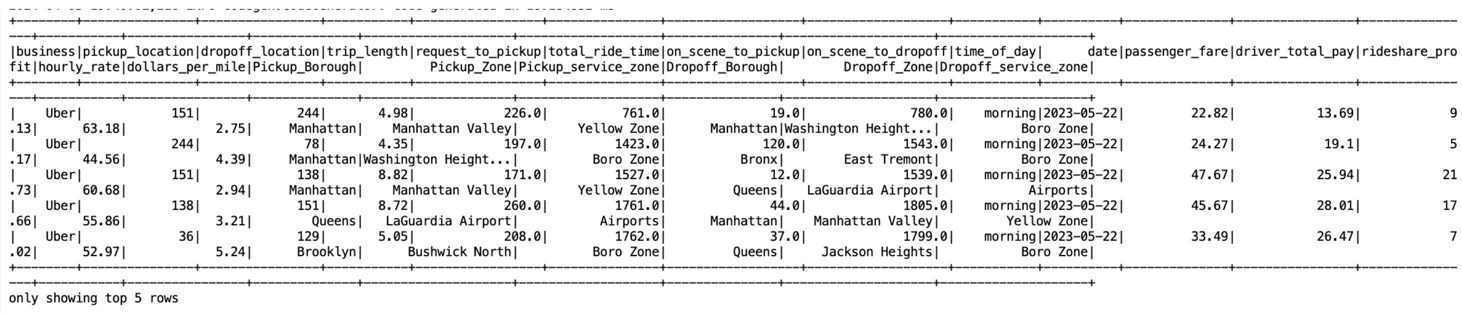
- The process begins with the retrieval of two separate CSV datasets from a cloud-based storage service. Utilizing Spark's data reading functionality, which is adept at processing CSV files, it includes an option to discern the first line as the header for columns.
- The method involves blending the rideshare records with the geographical information from the taxi zones. This merging is executed first for the pickup data and subsequently for the dropoff data, making use of Spark's DataFrame 'join' operation, correlating the data through the location IDs.
- After merging, the script adjusts the names of the columns to clearly indicate if they pertain to pickup or dropoff data. This renaming is efficiently performed using the 'withColumnRenamed' feature.
- The 'date' column undergoes a transformation from a UNIX timestamp to a conventional date format, enhancing both the legibility and the utility for further processing. The transformation employs the 'from_unixtime' function.
- To ensure the accuracy of the transformation, the total number of entries and the data structure are outputted using the 'count()' and 'printSchema()' functions, respectively.
- To offer a sneak peek at the modified dataset, the code exhibits the top five entries, providing an immediate opportunity to verify the alterations.

Visualization



```
Number of rows: 69725864
root
 |-- business: string (nullable = true)
 |-- pickup_location: integer (nullable = true)
 |-- dropoff_location: integer (nullable = true)
 |-- trip_length: double (nullable = true)
 |-- request_to_pickup: double (nullable = true)
 |-- total_ride_time: double (nullable = true)
 |-- on_scene_to_pickup: double (nullable = true)
 |-- on_scene_to_dropoff: double (nullable = true)
 |-- time_of_day: string (nullable = true)
 |-- date: string (nullable = true)
 |-- passenger_fare: double (nullable = true)
 |-- driver_total_pay: double (nullable = true)
 |-- rideshare_profit: double (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)
```

(Figure 1.2)



```
+--------+---------------+----------------+-----------+-----------------+---------------+------------------+-------------------+-----------+----------+--------------+----------------+-------------
---+-----------+----------------+--------------+--------------------+-------------------+---------------+--------------------+--------------------+
|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_pro
fit|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
```

(Figure 1.3 & 1.4)

Challenges Addressed

- **Managing Extensive Data**: The extensive dataset poses significant challenges in terms of processing. By deploying Spark's powerful data distribution and processing features, the task of manipulating such a sizable dataset is made manageable.
- **Clear Delineation of Data Points**: Post-join, it's imperative to maintain clarity amongst numerous alike fields. This was accomplished by systematically reassigning names to the columns, ensuring that each piece of data is distinctly identifiable.

Insights Gained

- The significance of preparing data with careful consideration has been accentuated by this task, laying the groundwork for any analytical or machine learning tasks that may follow.
- The assignment showcases the effectiveness of Spark's DataFrame operations in handling and transforming datasets of significant volume.
- By renaming the columns post-integration, the data becomes more navigable and straightforward, facilitating future tasks involving this dataset.

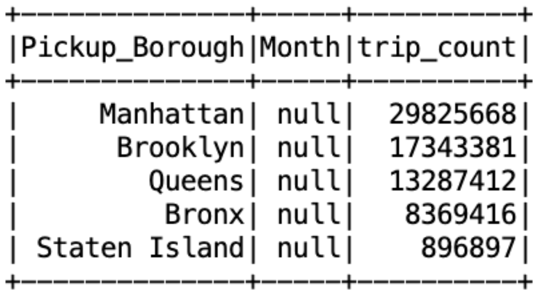
Task 3

Data Transformation and Aggregation

The task focused on analyzing the rideshare data to ascertain the most frequented pickup and dropoff boroughs and the most lucrative routes. This required additional data transformation and the use of aggregation functions to summarize the dataset.

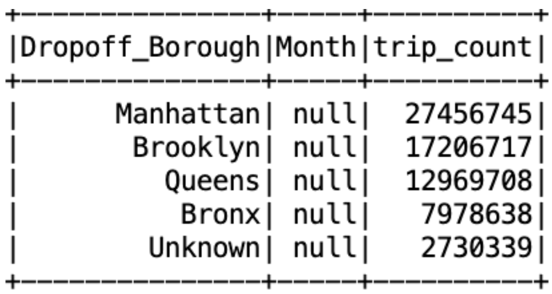
- **Temporal Segmentation**: A new column, 'Month', was introduced to the dataset by extracting the month from the 'date' field. This enabled segmentation of data for monthly analysis.
- **Window Specification**: A window specification was defined to assist in ranking the data within each month based on the trip count.
- **Pickup Borough Aggregation**: The data was grouped by 'Pickup_Borough' and 'Month', counted, and then ordered to find the most popular pickup locations on a monthly basis. The top five boroughs for each month were extracted using the window specification.
- **Dropoff Borough Aggregation**: A similar aggregation was performed for 'Dropoff_Borough' to identify the top dropoff locations, again extracting the top five for each month.
- **Route Profitability Calculation**: After casting 'driver_total_pay' to a float for accurate summation, a 'Route' column was created concatenating 'Pickup_Borough' and 'Dropoff_Borough'. The dataset was then grouped by 'Route' to calculate the total profits, from which the top thirty most profitable routes were determined.

Visualization



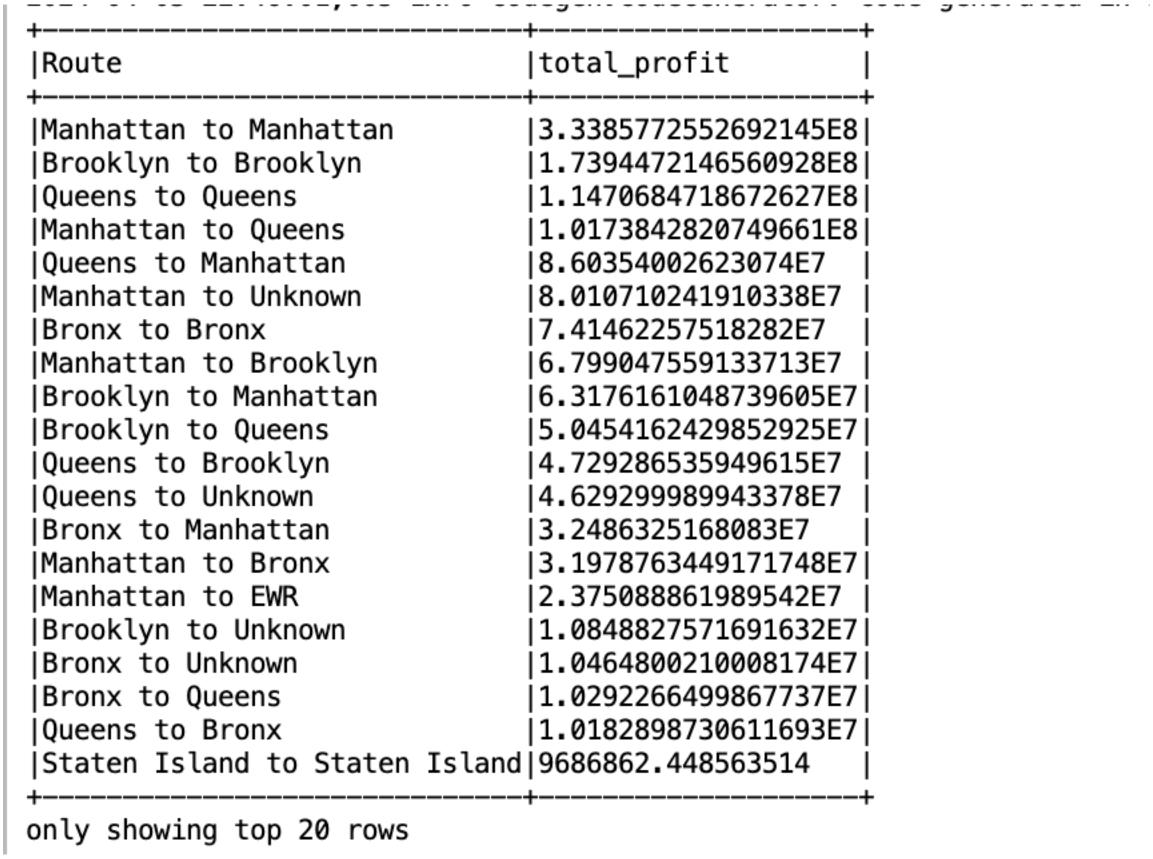
```
+--------------+-----+----------+
|Pickup_Borough|Month|trip_count|
+--------------+-----+----------+
|     Manhattan| null|  29825668|
|      Brooklyn| null|  17343381|
|        Queens| null|  13287412|
|         Bronx| null|   8369416|
| Staten Island| null|    896897|
+--------------+-----+----------+
```

(Figure 3.1)



```
+---------------+-----+----------+
|Dropoff_Borough|Month|trip_count|
+---------------+-----+----------+
|      Manhattan| null|  27456745|
|       Brooklyn| null|  17206717|
|         Queens| null|  12969708|
|          Bronx| null|   7978638|
|        Unknown| null|   2730339|
+---------------+-----+----------+
```

(Figure 3.2)



```
+------------------------------+--------------------+
|Route                         |total_profit        |
+------------------------------+--------------------+
|Manhattan to Manhattan        |3.3385772552692145E8|
|Brooklyn to Brooklyn          |1.7394472146560928E8|
|Queens to Queens              |1.1470684718672627E8|
|Manhattan to Queens           |1.0173842820749661E8|
|Queens to Manhattan           |8.60354002623074E7  |
|Manhattan to Unknown          |8.010710241910338E7 |
|Bronx to Bronx                |7.41462257518282E7  |
|Manhattan to Brooklyn         |6.799047559133713E7 |
|Brooklyn to Manhattan         |6.3176161048739605E7|
|Brooklyn to Queens            |5.0454162429852925E7|
|Queens to Brooklyn            |4.729286535949615E7 |
|Queens to Unknown             |4.629299989943378E7 |
|Bronx to Manhattan            |3.2486325168083E7   |
|Manhattan to Bronx            |3.1978763449171748E7|
|Manhattan to EWR              |2.375088861989542E7 |
|Brooklyn to Unknown           |1.0848827571691632E7|
|Bronx to Unknown              |1.0464800210008174E7|
|Bronx to Queens               |1.0292266499867737E7|
|Queens to Bronx               |1.0182898730611693E7|
|Staten Island to Staten Island|9686862.448563514   |
+------------------------------+--------------------+
only showing top 20 rows
```

(Figure 3.3)

Challenges Addressed

- **Data Partitioning for Monthly Analysis**: Breaking down the data month-by-month posed a challenge due to the need for precise temporal grouping. This was resolved by creating a dedicated 'Month' column and employing window functions.
- **Ranking and Filtering**: Determining the top five boroughs required careful application of ranking functions within each partition. The row_number function, in conjunction with a window specification, facilitated this process.
- **Handling Numerical Data**: The 'driver_total_pay' field needed to be converted to a numeric type to enable accurate profit calculations. Casting the column to a float type ensured that the subsequent aggregation would reflect precise earnings.

Insights Gained

- The aggregation of rideshare data revealed preferential travel patterns within specific boroughs, indicating areas of high demand.
- Analysis of the profitability of routes provided an understanding of which journeys yielded the highest returns, information that can be leveraged to optimize driver routes for increased earnings.
- The synthesis of the data emphasized the importance of versatile data manipulation techniques in extracting meaningful information from large datasets, demonstrating the capabilities of PySpark in handling complex analytical tasks.

(Part4)

If I were a stakeholder in the rideshare company, here are three insights I would draw from the data:

- **Geographical Demand Concentration**: The concentration of trip counts in specific boroughs, particularly Manhattan and Brooklyn, indicates a high demand in these areas. As a CEO, this information would be pivotal in strategizing where to focus marketing efforts and allocate resources. For drivers, understanding these demand hotspots could inform where they start their shifts to maximize fare opportunities. For a stockbroker or investor, this pattern of demand could indicate the company's stronghold in the market and its potential for growth in less saturated areas.
- **Temporal Fluctuations in Service Use**: The null values in the 'Month' column suggest missing temporal data which could impact the accuracy of monthly demand and profitability assessments. However, assuming 'Month' data was available, trends in ridership over different months could help predict demand peaks and troughs. This would assist CEOs in workforce planning and promotional campaigns. For drivers, knowing the busier periods can help in planning their schedules to work more during high-demand times. Investors could use this to anticipate fluctuations in the company's performance.
- **Route Profitability**: The profitability of specific routes offers insights into where drivers can earn the most. As a driver, focusing on these profitable routes could increase earnings. For a CEO, this data is useful for pricing strategies or to introduce incentives for drivers on less popular but potentially high-earning routes. A stockbroker might interpret profitable routes as a sign of efficient operations and use this data to advise on the company's stock potential.

Task 4

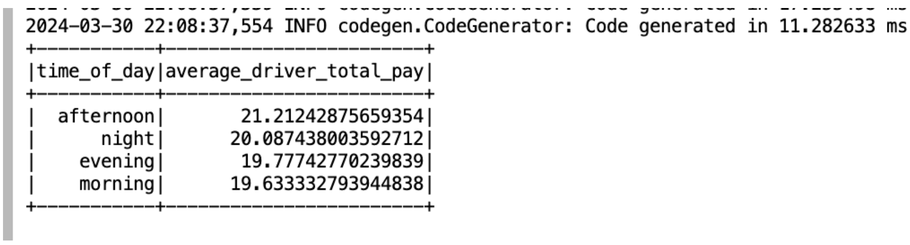
Execution and Data Processing Techniques

The task undertook a thorough examination of the data to uncover variances in average earnings and trip distances during specific times of the day, using PySpark's robust data handling features.

- **Income Evaluation**: The analysis started with segmenting the data by 'time_of_day' and determining the average income for drivers during these intervals, which illuminates the most financially beneficial times to drive.
- **Travel Distance Evaluation**: In a parallel approach, the data was again categorized by 'time_of_day' to calculate the mean travel distances, shedding light on the variations in trip lengths during different parts of the day.
- **Earnings Efficiency Measurement**: Integrating the insights from the earnings and distance evaluations led to the creation of an earnings efficiency indicator, termed 'average_earning_per_mile', juxtaposing profitability against the distance.

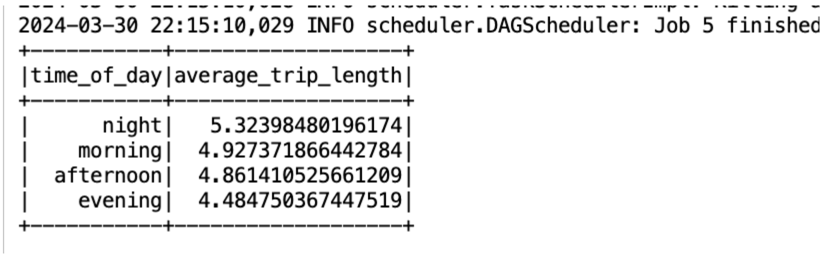
These analytical stages were ordered to highlight the segments with the most substantial figures, arranging the data in descending order for clarity.

Visualisation



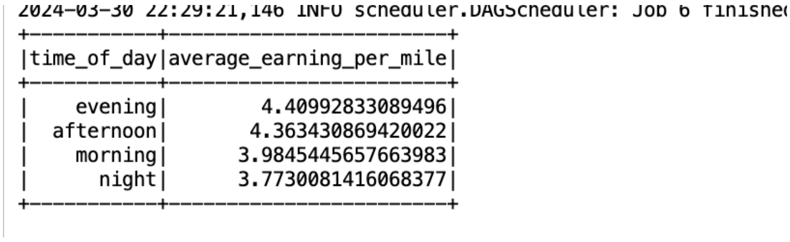
```
2024-03-30 22:08:37,554 INFO codegen.CodeGenerator: Code generated in 11.282633 ms
+-----------+------------------------+
|time_of_day|average_driver_total_pay|
+-----------+------------------------+
|  afternoon|       21.21242875659354|
|      night|      20.087438003592712|
|    evening|       19.77742770239839|
|    morning|      19.633332793944838|
+-----------+------------------------+
```

(Figure 4.1)



```
2024-03-30 22:15:10,029 INFO scheduler.DAGScheduler: Job 5 finished
+-----------+-------------------+
|time_of_day|average_trip_length|
+-----------+-------------------+
|      night|   5.32398480196174|
|    morning|  4.927371866442784|
|  afternoon|  4.861410525661209|
|    evening|  4.484750367447519|
+-----------+-------------------+
```

(Figure 4.2)



```
2024-03-30 22:29:21,146 INFO scheduler.DAGScheduler: Job 6 finishe
+-----------+-------------------------+
|time_of_day|average_earning_per_mile|
+-----------+-------------------------+
|    evening|        4.40992833089496|
|  afternoon|       4.363430869420022|
|    morning|      3.9845445657663983|
|      night|      3.7730081416068377|
+-----------+-------------------------+
```

(Figure 4.3)

Challenges Overcome

- **Numerical Accuracy**: The prerequisite for accurate numerical data types for 'driver_total_pay' and 'trip_length' was paramount. This was addressed by converting these fields to the correct numerical types where required.
- **Data Merging Technicalities**: The union of two separate sets of aggregated data necessitated a careful and precise join on the 'time_of_day' field to ensure a harmonious merge.

Insights Acquired

- The analysis pointed to the evening period as the peak time for earning efficiency, suggesting the most profitable hours for driving operations.
- Longer trip lengths were observed during the night, potentially due to various nocturnal factors such as reduced traffic or passenger preferences for longer trips at night.
- The afternoon hours appeared to be the most lucrative in terms of overall driver income, potentially reflecting a mix of trip frequency and length that optimizes drivers' revenue.

(Part4)

From the three results presented,

- **Time-Based Earnings Variability**: The data indicates that earnings per mile vary significantly throughout the day, with evenings being the most profitable. This suggests that the demand for rideshare services may increase during evening hours, possibly due to the close of business hours and social activities.
- **Trip Length Differences**: There is a noticeable fluctuation in the average length of trips at different times, with nights showing a tendency for longer trips. This could imply that during the night, routes are less congested, or that there's a different customer base using the services, possibly for longer distances.

How these insights guide strategy and decision-making:

- **For Drivers**: Understanding the dynamics of earnings and trip lengths can inform personal strategies to maximize income. Drivers can plan their schedules to work during times when earnings per mile are highest and when longer trips are more frequent, thus optimizing their working hours and earnings.
- **For Rideshare Companies**: These insights are invaluable for adjusting fare pricing, optimizing dispatch, and setting surge pricing during times of high demand to maximize revenue. Additionally, the company might consider special incentives for drivers to ensure availability during the most profitable hours.
- **For Investors or Analysts**: The patterns in profitability and trip lengths can inform market analyses and investment decisions. If a rideshare company shows a strategic understanding of these patterns and an ability to capitalize on them, it may suggest a stronger position in the market and potential for growth.

Task 6

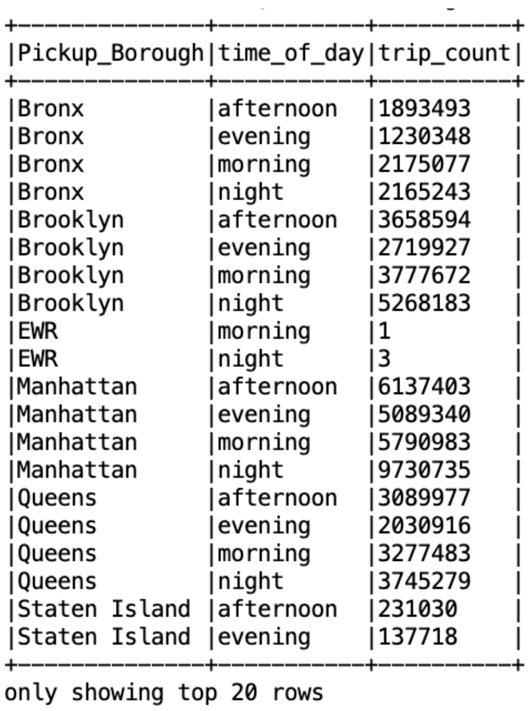
Detailed Analysis Process

In Task 6, the objective was to dissect and understand the distribution of trip counts based on pickup boroughs and times of the day. The analysis required filtering the data for realistic trip lengths and examining patterns within specific temporal contexts.

- **Trip Count Across Times of Day**: The initial focus was to determine the number of trips across different times of the day, grouped by the pickup boroughs. This was achieved by filtering out trips of unlikely lengths and then aggregating counts per borough and time segment.
- **Evening Trip Focus**: A deeper dive into evening trips specifically helped to capture the patterns of ridership during one of the busiest times of the day for rideshare services. The trips were grouped by pickup borough to understand which areas had the highest demand in the evening.
- **Brooklyn to Staten Island Route**: A specific route from Brooklyn to Staten Island was singled out to identify the volume of trips between these two boroughs, offering insights into the frequency of this inter-borough connectivity.

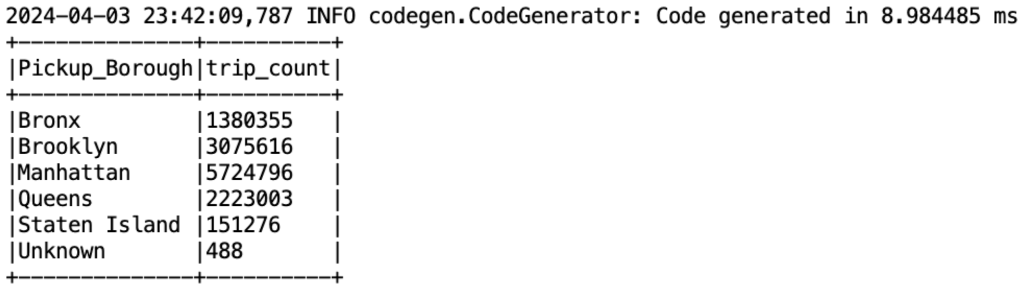
Each of these steps utilized PySpark's data filtering and grouping capabilities, with final data sorted and displayed for easy interpretation.

Visualization



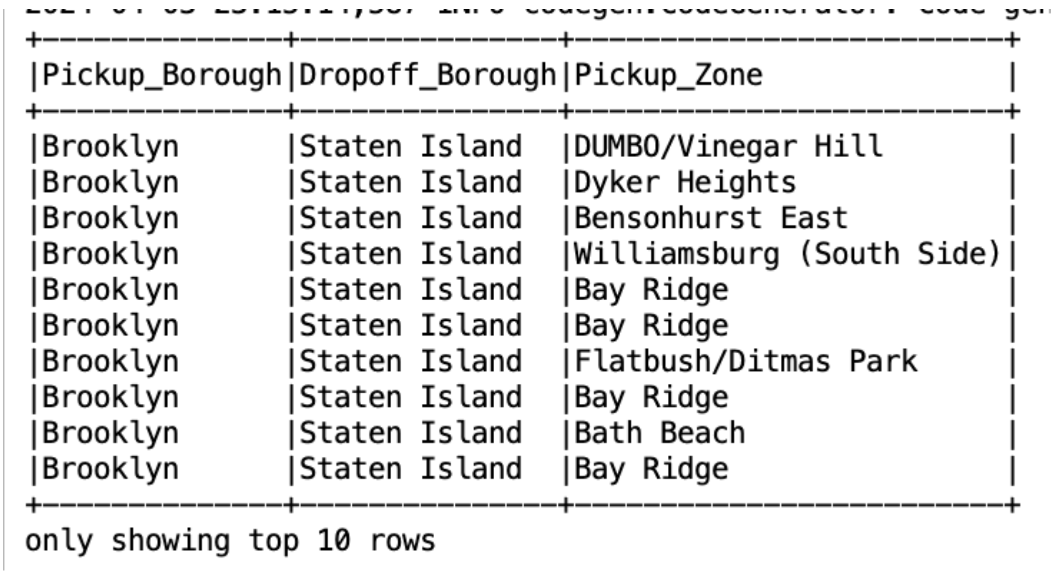
```
+--------------+-----------+----------+
|Pickup_Borough|time_of_day|trip_count|
+--------------+-----------+----------+
|Bronx         |afternoon  |1893493   |
|Bronx         |evening    |1230348   |
|Bronx         |morning    |2175077   |
|Bronx         |night      |2165243   |
|Brooklyn      |afternoon  |3658594   |
|Brooklyn      |evening    |2719927   |
|Brooklyn      |morning    |3777672   |
|Brooklyn      |night      |5268183   |
|EWR           |morning    |1         |
|EWR           |night      |3         |
|Manhattan     |afternoon  |6137403   |
|Manhattan     |evening    |5089340   |
|Manhattan     |morning    |5790983   |
|Manhattan     |night      |9730735   |
|Queens        |afternoon  |3089977   |
|Queens        |evening    |2030916   |
|Queens        |morning    |3277483   |
|Queens        |night      |3745279   |
|Staten Island |afternoon  |231030    |
|Staten Island |evening    |137718    |
+--------------+-----------+----------+
only showing top 20 rows
```

(Figure 6.1)



```
2024-04-03 23:42:09,787 INFO codegen.CodeGenerator: Code generated in 8.984485 ms
+--------------+----------+
|Pickup_Borough|trip_count|
+--------------+----------+
|Bronx         |1380355   |
|Brooklyn      |3075616   |
|Manhattan     |5724796   |
|Queens        |2223003   |
|Staten Island |151276    |
|Unknown       |488       |
+--------------+----------+
```

(Figure 6.2)



```
2024-04-03 23:13:14,387 INFO codegen.CodeGenerator: Code gen
+--------------+---------------+------------------------+
|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               |
+--------------+---------------+------------------------+
only showing top 10 rows
```

(Figure 6.3)

Challenges Addressed

- **Data Filtering:** Careful filtering was crucial to ensure only meaningful trip lengths were considered. The use of logical conditions in PySpark's filter method ensured data integrity.
- **Data Aggregation and Sorting:** Aggregating trip counts by different dimensions and sorting them required meticulous construction of PySpark queries to ensure accurate and insightful results.

Insights Gained

- **Temporal and Geographical Patterns:** The analysis revealed which boroughs experience the most traffic at various times, with a detailed view of the evening rush.
- **Specific Route Analysis:** The frequency of trips between Brooklyn and Staten Island highlighted a particular route's demand, which might be underserved or overutilized.

The findings from this task inform strategic decisions regarding resource allocation, service optimization, and targeted marketing. They could also influence operational strategies such as dynamic pricing, driver distribution, and route planning to better meet demand and improve service efficiency.

Task 7

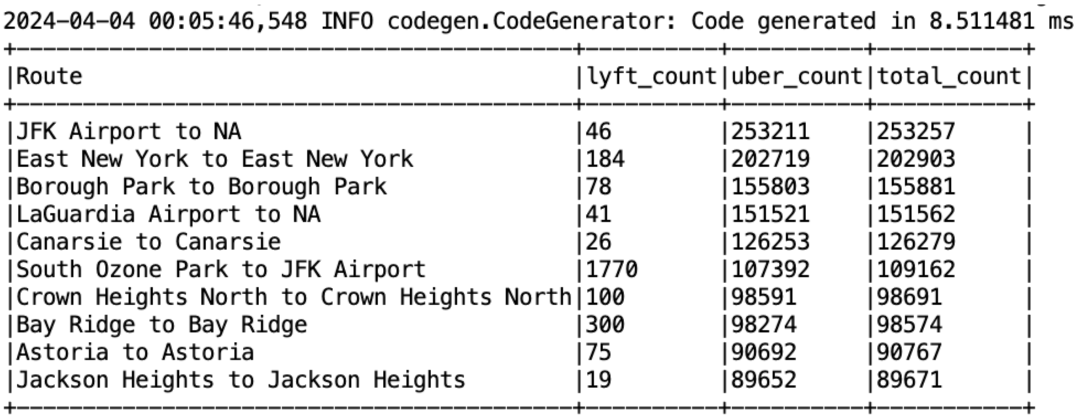
Analytical Approach and Data Handling

Task 7 aimed to dissect and identify the rideshare routes with the highest volume of traffic, differentiating between two key service providers.

- **Route Identification**: The preliminary step constructed a distinct 'Route' identifier by merging 'Pickup_Zone' and 'Dropoff_Zone', thereby creating a clear label for each journey path.
- **Data Compilation**: Trips were then aggregated by 'Route', and a pivot table was used to differentiate trip counts per rideshare company. In instances of absent data, zeros were introduced to maintain consistency across the dataset.
- **Summation of Trips**: A new column representing the combined trip counts from both services was instituted, yielding a comprehensive view of each route's total usage.
- **Isolation of Premier Routes**: The data was then organized to highlight the 10 routes with the highest cumulative trip counts, showcasing the most frequented travel paths.

The process capitalized on the advanced data processing power of PySpark, utilizing its capacity to pivot, aggregate, and summarize large datasets efficiently.

Visualization



```
2024-04-04 00:05:46,548 INFO codegen.CodeGenerator: Code generated in 8.511481 ms
+------------------------------------------+----------+----------+-----------+
|Route                                     |lyft_count|uber_count|total_count|
+------------------------------------------+----------+----------+-----------+
|JFK Airport to NA                         |46        |253211    |253257     |
|East New York to East New York            |184       |202719    |202903     |
|Borough Park to Borough Park              |78        |155803    |155881     |
|LaGuardia Airport to NA                   |41        |151521    |151562     |
|Canarsie to Canarsie                      |26        |126253    |126279     |
|South Ozone Park to JFK Airport           |1770      |107392    |109162     |
|Crown Heights North to Crown Heights North|100       |98591     |98691      |
|Bay Ridge to Bay Ridge                    |300       |98274     |98574      |
|Astoria to Astoria                        |75        |90692     |90767      |
|Jackson Heights to Jackson Heights        |19        |89652     |89671      |
+------------------------------------------+----------+----------+-----------+
```

(Figure 7)

Challenges Overcome

- **Uniform Data Representation**: The possibility of incomplete data for certain routes was preemptively addressed with a methodical approach to populate such instances with zero values, ensuring a balanced dataset.
- **Clear Data Conveyance**: Renaming the columns generated by the pivot was imperative to enhance the readability of the data, allowing for immediate comprehension of the summarized information.

Key Findings

- **High-Traffic Corridors**: The analysis highlighted major corridors, particularly those in and out of Manhattan, emphasizing their significance in urban transit.
- **Service Utilization Insights**: By comparing service providers on high-traffic routes, the analysis provided a lens into market dominance and areas for potential growth.
- **Insights Into Transit Patterns**: The aggregation of total trips per route illuminated the concentration of demand, indicating where rideshare services are most integral to the city's transportation network.

These insights are particularly valuable for strategic decision-making related to resource allocation, service enhancement, and market competition, offering guidance to bolster efficiency and customer satisfaction.

Task 5

Analytical Framework and Execution

Task 5 delved into the average waiting time for rideshare services throughout January. The analysis required the processing of date and time data to yield a daily average waiting time for the month.

- **Date Refinement**: Initially, the 'date' column was reformatted to a standardized 'yyyy-MM-dd' format to accurately process the dates within the dataset.
- **January Filtering**: The dataset was narrowed to only include data from January, isolating the first month of the year for focused analysis.
- **Daily Average Calculation**: For each day in January, the average waiting time was computed, with the results ordered by the day of the month to track any fluctuations over the period.

The sequence of these steps was facilitated by PySpark's robust data processing capabilities, particularly its date and aggregation functions.

Challenges Mitigated

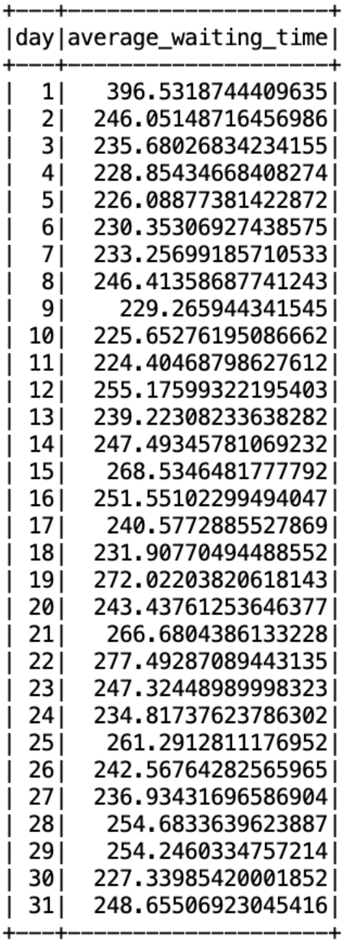
- **Date Handling**: The handling of dates in various formats can often present challenges. The use of the to_date function in PySpark provided a standardized approach, ensuring accuracy in filtering and grouping.
- **Daily Averages Representation**: The accurate representation of daily averages required precise aggregation functions. The avg function in PySpark allowed for a straightforward calculation of mean values.

Insights Derived

- **Fluctuation of Waiting Times**: The histogram and data table revealed that waiting times fluctuate throughout the month, with certain days experiencing significantly higher averages than others.
- **Peak Period Identification**: Identifying days with unusually long average waiting times could indicate peak demand periods or inefficiencies in the rideshare service operations.

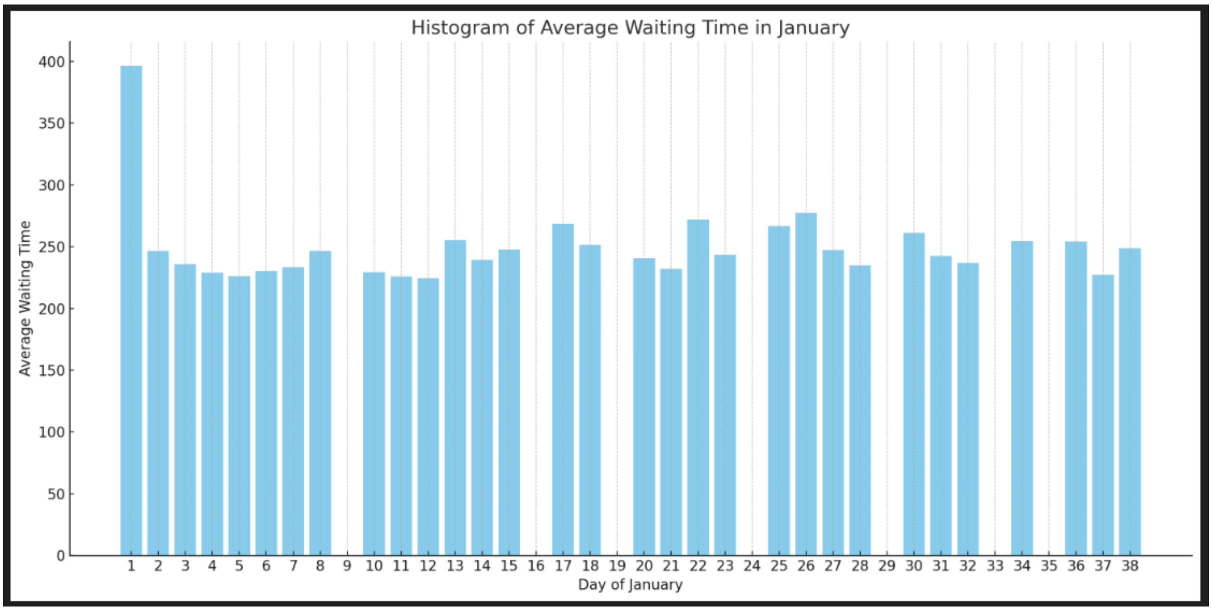
These insights, derived from the data, could inform operational adjustments to improve service efficiency. For instance, strategies such as dynamic scheduling of drivers, promotion of ride-sharing during peak times, or adjustments in pricing could be considered to manage demand and reduce waiting times. These measures would ultimately aim to enhance customer satisfaction and service reliability.

Visualization



```
+---+--------------------+
|day|average_waiting_time|
+---+--------------------+
|  1|   396.5318744409635|
|  2|  246.05148716456986|
|  3|  235.68026834234155|
|  4|  228.85434668408274|
|  5|  226.08877381422872|
|  6|  230.35306927438575|
|  7|  233.25699185710533|
|  8|  246.41358687741243|
|  9|    229.265944341545|
| 10|  225.65276195086662|
| 11|  224.40468798627612|
| 12|  255.17599322195403|
| 13|  239.22308233638282|
| 14|  247.49345781069232|
| 15|   268.5346481777792|
| 16|  251.55102299494047|
| 17|   240.5772885527869|
| 18|  231.90770494488552|
| 19|  272.02203820618143|
| 20|  243.43761253646377|
| 21|   266.6804386133228|
| 22|  277.49287089443135|
| 23|  247.32448989998323|
| 24|  234.81737623786302|
| 25|   261.2912811176952|
| 26|  242.56764282565965|
| 27|  236.93431696586904|
| 28|   254.6833639623887|
| 29|   254.2460334757214|
| 30|  227.33985420001852|
| 31|  248.65506923045416|
+---+--------------------+
```

(Figure 5.1)



(Figure for 5.1)

(Part2)

Upon examining the provided data output, it's clear that the average waiting time surpasses 300 seconds on the first day of January, where it notably peaks at approximately 396.5 seconds. This observation suggests a significant increase in waiting time on that particular day compared to the rest of the month.

(Part 3)

The extended average waiting time observed on the first day of January could be attributed to several factors. It's a day commonly associated with New Year celebrations, likely leading to an increased demand for rideshare services. Additionally, drivers may be less available due to the holiday, contributing to longer wait times for customers.