Task 1 (25 points) NYC Yellow Taxi dataset

In task 1, you will analyze the **NYC Yellow Taxi dataset** using **Spark**. The **NYC Yellow Taxi dataset** is stored in files named with different months of year 2023 under the path **//data-repository-bkt/ECS765/nyc_taxi/yellow_tripdata/2023/.** You can view the files using **ccc method bucket ls**.

The description of each field in order for the **dataset** is shown below (a sample CSV file was given to explore/visualise on your machine):

tpep_pickup_datetime: The date and time when the meter was engaged.
tpep_dropoff_datetime: The date and time when the meter was disengaged
passenger_count: The number of passengers in the vehicle
trip_distance: The elapsed trip distance in miles reported by the taximeter
PULocationID: Taxi Zone in which the taximeter was engaged
DOLocationID: Taxi Zone in which the taximeter was disengaged
payment_type: A numeric code signifying how the passenger paid for the trip. 1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip
fare_amount: The time-and-distance fare calculated by the meter
extra: Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges.
mta_tax: $0.50 MTA tax that is automatically triggered based on the metered rate in use
tip_amount: $0.50 MTA tax that is automatically triggered based on the metered rate in use
tolls_amount: $0.50 MTA tax that is automatically triggered based on the metered rate in use
total_amount: $0.50 MTA tax that is automatically triggered based on the metered rate in use
congestion_surcharge: $0.50 MTA tax that is automatically triggered based on the metered rate in use
airport_fee: $0.50 MTA tax that is automatically triggered based on the metered rate in use
taxi_type: yellow taxi

The pictures below show a few records as samples of the **NYC Yellow Taxi dataset**.

FIELD1	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	PULocationID	DOLocationID
0	2023-01-01 00:32:10	2023-01-01 00:40:36	1.0	0.97	161	141
1	2023-01-01 00:55:08	2023-01-01 01:01:27	1.0	1.1	43	237
2	2023-01-01 00:25:04	2023-01-01 00:37:49	1.0	2.51	48	238
3	2023-01-01 00:03:48	2023-01-01 00:13:25	0.0	1.9	138	7
4	2023-01-01 00:10:29	2023-01-01 00:21:19	1.0	1.43	107	79
5	2023-01-01 00:50:34	2023-01-01 01:02:52	1.0	1.84	161	137

payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	total_amount	congestion_surcharge	airport_fee	taxi_type
2	9.3	1.0	0.5	0.0	0.0	14.3	2.5	0.0	yellow_taxi
1	7.9	1.0	0.5	4.0	0.0	16.9	2.5	0.0	yellow_taxi
1	14.9	1.0	0.5	15.0	0.0	34.9	2.5	0.0	yellow_taxi
1	12.1	7.25	0.5	0.0	0.0	20.85	0.0	1.25	yellow_taxi
1	11.4	1.0	0.5	3.28	0.0	19.68	2.5	0.0	yellow_taxi
1	12.8	1.0	0.5	10.0	0.0	27.8	2.5	0.0	yellow_taxi

There is another file named **"taxi_zone_lookup.csv"** stored under the path **//data-repository-bkt/ECS765/nyc_taxi/**. The **"taxi_zone_lookup.csv"** has location information for each **LocationID** used in the **dataset**. This schema of the table:

LocationID: integer (nullable = true)
Borough: string (nullable = true)
Zone: string (nullable = true)
service_zone: string (nullable = true)

In the table below, we show a few sample records of **"taxi_zone_lookup.csv".**

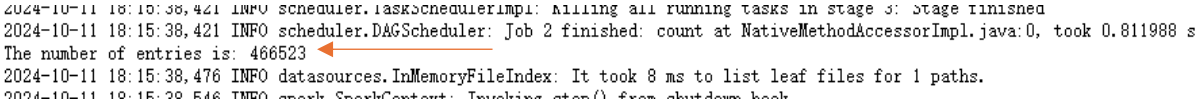
LocationID	Borough	Zone	service_zone
1	EWR	Newark Airport	EWR
2	Queens	Jamaica Bay	Boro Zone
3	Bronx	Allerton/Pelham Gardens	Boro Zone
4	Manhattan	Alphabet City	Yellow Zone

As you can see, the **"PULocationID"** and **"DOLocationID"** fields in the **dataset** are encoded with numbers that you can find their counterpart (**LocationID** field) with location information in **"taxi_zone_lookup.csv"**. Please, note the following:

1. The LocationID 264 and 265 are **Unknown** in the Borough field; we see the **Unknown** as one of the borough names;
2. In the Borough field, you can see the same **Borough** has a different **LocationID**, it does not matter because the **LocationID** is a unique key;
3. If you see any obscure description (like NV, NA, N/A, etc) in any field, purely see it as valid names.

Please note that the example tables/pictures provided in this task are based on my results. Your results may differ depending on how you process and display the data. When a screenshot is required, in addition to the results, you must also include your running timestamp in your screenshot. A running timestamp is the recorded time (printed at the beginning of each line in the logs) during the execution of a Spark job; you can capture the running timestamp in the logs. The example tables/pictures provided in each question illustrate the expected format (including the running timestamp and your results) for your screenshot.

1) **(2 points)** Load the **taxi_zone_lookup.csv** file into one dataframe. Then, load all monthly CSV files for the **NYC Yellow Taxi dataset** into another dataframe and print the total number of entries in the **NYC Yellow Taxi dataframe**. You will need to include a screenshot of your results in your report. For example:



Note that the above figures/values may not represent the actual result.

2) **(5 points)** Find the number of trips that have a **fare_amount** greater than $50 but a **trip_distance** of less than 1 mile for each day from 2023-02-01 to 2023-02-07 (**tpep_pickup_datetime field**). Sort your results by date. You need to convert the **tpep_pickup_datetime** field **to "YYYY-MM_DD"** format. You will need to include a screenshot of your results in your report. **Do not truncate the fields.** The format for the results of this question is provided. Please ensure that you include the timestamp in your report, as required. For example,

trip_date	trip_count
2023-02-01	200
2023-02-02	300
2023-02-03	100
2023-02-04	50
2023-02-05	134
2023-02-06	346
2023-02-07	567

Note that the above figures/values do not represent the actual result.

3) **(3 points)** Apply the **join** function on the **PULocationID** and **DOLocationID** fields from the **NYC Yellow Taxi dataframe** with the **LocationID** field from the **taxi_zone_lookup.csv dataframe**.

The joining process should be conducted in 4 steps:

1. Join the datasets using **PULocationID**, and rename the corresponding columns as **Pickup_Borough**, **Pickup_Zone**, and **Pickup_service_zone**.
2. Drop the **LocationID** and **PULocationID** columns.
3. Use the result from this step to perform a second join using **DOLocationID**, renaming the corresponding fields as **Dropoff_Borough**, **Dropoff_Zone**, and **Dropoff_service_zone**.
4. Drop the **LocationID** and **DOLocationID** columns.

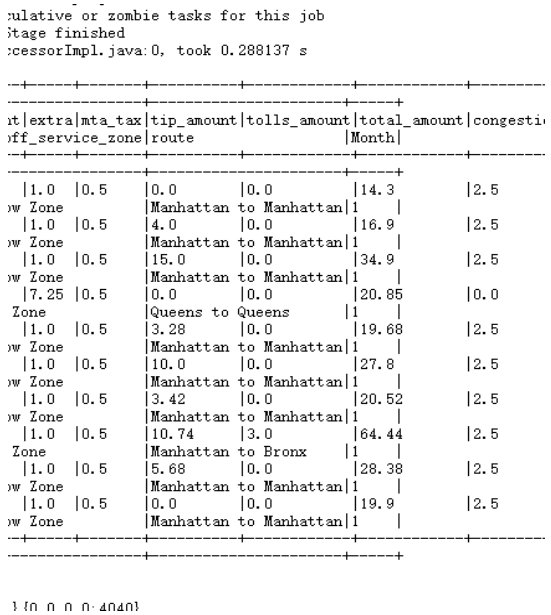
You need to include a screenshot of the resulting schema in your report. Your resulting schema should match the one shown in the image below.

```
2024-10-13 09:52:36,429 INFO spark.SparkContext: Created broadcast 6 from csv
2024-10-13 09:52:36,429 INFO execution.FileSourceScanExec: Planning scan with
root
 |-- tpep_pickup_datetime: string (nullable = true)
 |-- tpep_dropoff_datetime: string (nullable = true)
 |-- passenger_count: string (nullable = true)
 |-- trip_distance: string (nullable = true)
 |-- payment_type: string (nullable = true)
 |-- fare_amount: string (nullable = true)
 |-- extra: string (nullable = true)
 |-- mta_tax: string (nullable = true)
 |-- tip_amount: string (nullable = true)
 |-- tolls_amount: string (nullable = true)
 |-- total_amount: string (nullable = true)
 |-- congestion_surcharge: string (nullable = true)
 |-- airport_fee: string (nullable = true)
 |-- taxi_type: 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)

2024-10-13 09:52:36,583 INFO server.AbstractConnector: Stopped Spark@17746844{
2024-10-13 09:52:36,585 INFO ui.SparkUI: Stopped Spark web UI at http://q3-d82
2024-10-13 09:52:36,588 INFO k8s.KubernetesClusterSchedulerBackend: Shutting d
```

4) (**4 points**) Add a new column called **"route"** by concatenating the values of the **"Pickup_Borough"** and **"Dropoff_Borough"** columns, separated by **" to ".** Add a **"Month"** column by extracting the **month** from the **"tpep_pickup_datetime"** column. Show 10 rows of the results in your report (Include the **"route"** and **"Month"** fields in your screenshot). **Do not truncate the fields.** An example of the format of the results is only provided in this

question. Please ensure that you include the timestamp in your report, as required. For example,



```
:ulative or zombie tasks for this job
:tage finished
:cessorImpl.java:0, took 0.288137 s

+---+-----+-------+---------+-----------+------------+--------
---------------+--------------------+----+
rt|extra|mta_tax|tip_amount|tolls_amount|total_amount|congesti
:ff_service_zone|route               |Month|
+---+-----+-------+---------+-----------+------------+--------
---------------+--------------------+----+
  |1.0  |0.5    |0.0       |0.0         |14.3        |2.5
:w Zone         |Manhattan to Manhattan|1    |
  |1.0  |0.5    |4.0       |0.0         |16.9        |2.5
:w Zone         |Manhattan to Manhattan|1    |
  |1.0  |0.5    |15.0      |0.0         |34.9        |2.5
:w Zone         |Manhattan to Manhattan|1    |
  |7.25 |0.5    |0.0       |0.0         |20.85       |0.0
 Zone           |Queens to Queens      |1    |
  |1.0  |0.5    |3.28      |0.0         |19.68       |2.5
:w Zone         |Manhattan to Manhattan|1    |
  |1.0  |0.5    |10.0      |0.0         |27.8        |2.5
:w Zone         |Manhattan to Manhattan|1    |
  |1.0  |0.5    |3.42      |0.0         |20.52       |2.5
:w Zone         |Manhattan to Manhattan|1    |
  |1.0  |0.5    |10.74     |3.0         |64.44       |2.5
 Zone           |Manhattan to Bronx    |1    |
  |1.0  |0.5    |5.68      |0.0         |28.38       |2.5
:w Zone         |Manhattan to Manhattan|1    |
  |1.0  |0.5    |0.0       |0.0         |19.9        |2.5
:w Zone         |Manhattan to Manhattan|1    |
+---+-----+-------+---------+-----------+------------+--------
---------------+--------------------+----+


1 [0.0.0.0:4040]
```

5) **(5 points)** Group the dataset by **"Month"** and **"route"**, then aggregate the total tip amount and total passenger count for each group, renaming them as **"total_tip_amount"** and **"total_passenger_count"** respectively. After that, a new column named **"average_tip_per_passenger"** is created, which is calculated by dividing the **"total_tip_amount"** by the **"total_passenger_count"** for each group. You need to show 10 samples of the results in your report. **Do not truncate the fields.** The format for the results of this question is provided. Please ensure that you include the timestamp in your report, as required. For example,

Month	route	total_tip_amount	total_passenger_count	average_tip_per_passenger
4	EWR to Queens	55.5	5	11.1
2	EWR to Queens	444	4	111
2	EWR to Queens	30	3	10
10	EWR to Queens	2	1	1.0
1	EWR to Queens	2.22	2	1.11
……	……	……	……	……

Note that the above figures/values do not represent the actual result.

6) **(3 points)** Find the entries where **"average_tip_per_passenger"** is equal to 0. You need to include a screenshot of your results. **Do not truncate the fields**. The

format for the results of this question is provided. Please ensure that you include the timestamp in your report, as required. For example:

Month	route	total_tip_amount	total_passenger_count	average_tip_per_passenger
4	EWR to Queens	0	5	0
2	EWR to Queens	0	4	0
2	EWR to Queens	0	3	0
10	EWR to Queens	0	1	0
1	EWR to Queens	0	2	0
……	……	……	……	……

Note that the above figures/values may not represent the actual result.

7) **(3 points)** Find the top 10 routes in terms of the **average_tip_per_passenger**. You need to include a screenshot of your results. **Do not truncate the fields.** The format for the results of this question is provided. Please ensure that you include the timestamp in your report, as required. For example:

Month	route	total_tip_amount	total_passenger_count	average_tip_per_passenger
4	EWR to Queens	55	1	55
2	EWR to Queens	44	1	44
2	EWR to Queens	111	1	111
10	EWR to Queens	41	1	41
1	EWR to Queens	35	1	35
……	……	……	……	……

Note that the above figures/values may not represent the actual result