Task 3 (25 points) NYC Green Taxi dataset

In task 3, you will analyze the **NYC Green Taxi dataset** and may need to use the **Spark Graph Processing** techniques. The **dataset** is stored in files named with different months of year 2023 under the path **//data-repository-bkt/ECS765/nyc_taxi/green_tripdata/2023/.** You can view the files using the **ccc method bucket ls** command.

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):

lpep_pickup_datetime: The date and time when the meter was engaged.

lpep_dropoff_datetime: The date and time when the meter was disengaged.

PULocationID: Taxi Zone in which the taximeter was engaged. Refer to **'taxi_zone_lookup.csv'** for details.

DOLocationID: Taxi Zone in which the taximeter was engaged. Refer to **'taxi_zone_lookup.csv'** for details.

passenger_count: The number of passengers in the vehicle

trip_distance: The elapsed trip distance in miles reported by the taximeter

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: Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.

tolls_amount: Total amount of all tolls paid in trip

ehail_fee: Dont care column

total_amount: The total amount charged to passengers. It does not include cash tips

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

trip_type: A code indicating whether the trip was a street hail or a dispatch that is automatically assigned based on the metered rate in use but can be altered by the driver

congestion_surcharge: Total amount collected in trip for congestion surcharge

taxi_type: Green Taxi

We will use the two fields, **“PULocationID”** and **“DOLocationID”,** from the **dataset.** The table below shows a few sample records of **“PULocationID”** and **“DOLocationID”** from the **dataset**.

PULocationID	DOLocationID
166	143
24	43
223	179
41	238

There is another file named **“taxi_zone_lookup.csv”** stored under the path //**data-repository-bkt/ECS765/nyc_taxi/**. The **“taxi_zone_lookup.csv”** file contains the location info of each LocationID in the **dataset**. This table has the following schema:

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”.**

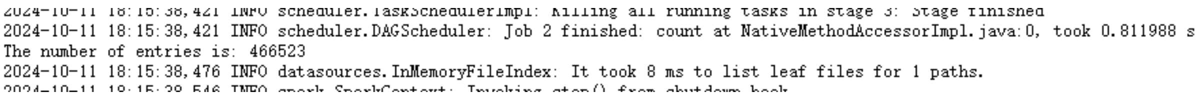
LocationID	Borough	Zone	service_zone
1	EWR	Newark Airport	EWR
2	Queens	Jamaica Bay	Boro Zone
3	Bronx	Allerton/Pelham Gardens	Boro Zone
4	Manhattan	Alphabet City	Yellow Zone

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.

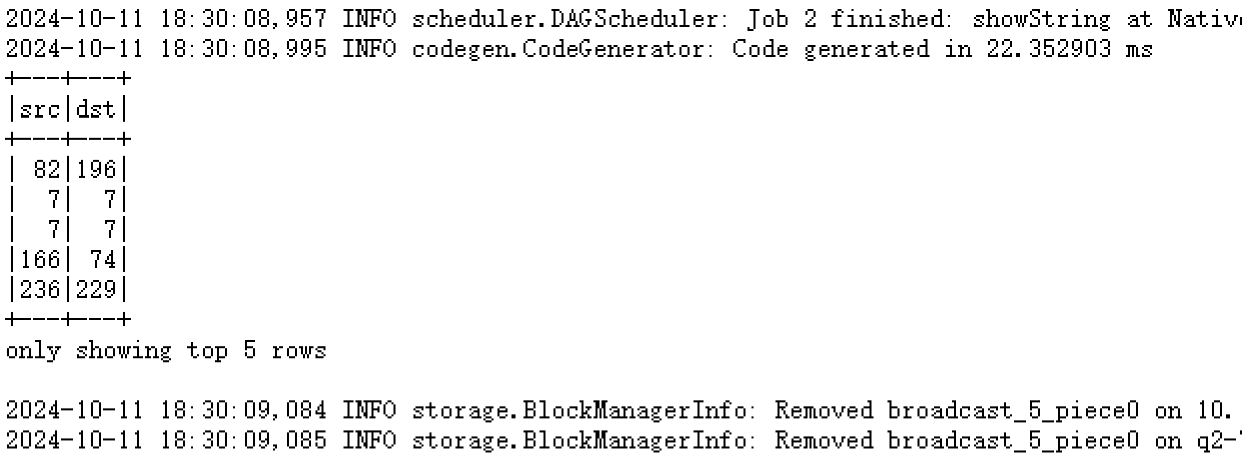
1) **(2 points)** Load all monthly CSV files from the NYC Green Taxi dataset into a single dataframe and print the total number of entries in the dataframe. You will need to include a screenshot of your results in your report. For example:



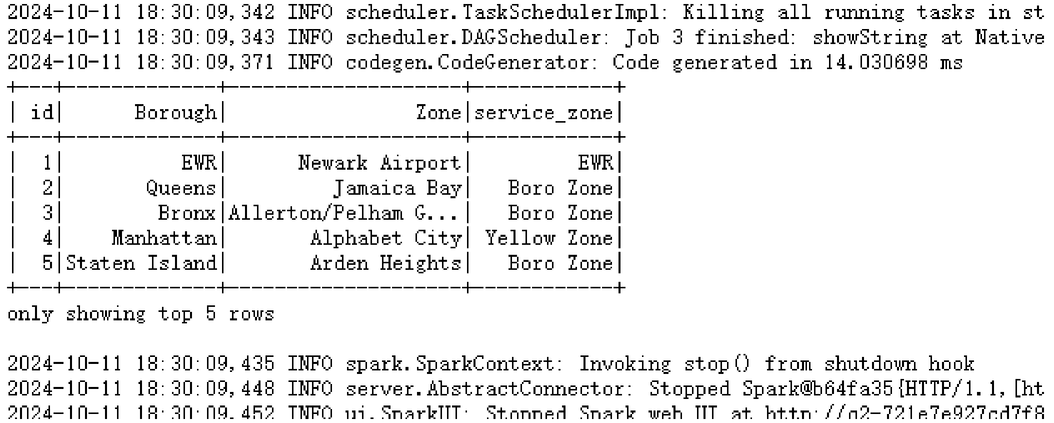
```
2024-10-11 18:15:38,421 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 3: Stage finished
2024-10-11 18:15:38,421 INFO scheduler.DAGScheduler: Job 2 finished: count at NativeMethodAccessorImpl.java:0, took 0.811988 s
The number of entries is: 466523
2024-10-11 18:15:38,476 INFO datasources.InMemoryFileIndex: It took 8 ms to list leaf files for 1 paths.
2024-10-11 18:15:38,546 INFO spark.SparkContext: Invoking stop() from shutdown hook
```

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

2) **(5 points)** Define the **StructType** for the vertexSchema and edgeSchema, and construct the edges dataFrame and vertices dataFrame. Use the **"taxi_zone_lookup.csv"** as the vertex information and the **"PULocationID"** and **"DOLocationID"** fields from the **NYC Green Taxi dataset** as the edge information.
Then, Display 5 samples of both the edges and vertices DataFrames, ensuring that **field names are not truncated**. The vertices DataFrame should include the fields: **"id", "Borough", "Zone", and "service_zone",** while the edges DataFrame should include **"src" and "dst".** Include a screenshot of the results in your report. For example,



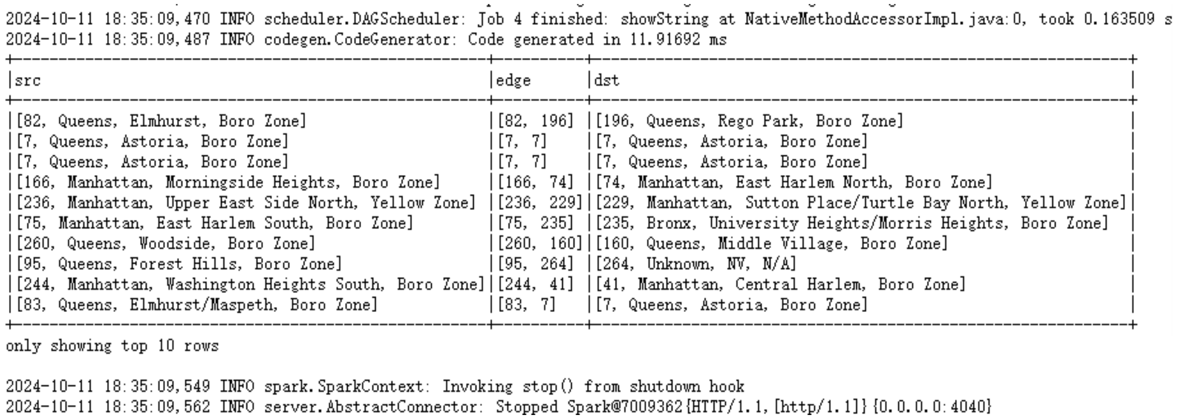
```
2024-10-11 18:30:08,957 INFO scheduler.DAGScheduler: Job 2 finished: showString at Nativ
2024-10-11 18:30:08,995 INFO codegen.CodeGenerator: Code generated in 22.352903 ms
+---+---+
|src|dst|
+---+---+
| 82|196|
|  7|  7|
|  7|  7|
|166| 74|
|236|229|
+---+---+
only showing top 5 rows

2024-10-11 18:30:09,084 INFO storage.BlockManagerInfo: Removed broadcast_5_piece0 on 10.
2024-10-11 18:30:09,085 INFO storage.BlockManagerInfo: Removed broadcast_5_piece0 on q2-
```



```
2024-10-11 18:30:09,342 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in st
2024-10-11 18:30:09,343 INFO scheduler.DAGScheduler: Job 3 finished: showString at Native
2024-10-11 18:30:09,371 INFO codegen.CodeGenerator: Code generated in 14.030698 ms
+---+-------------+--------------------+------------+
| id|      Borough|                Zone|service_zone|
+---+-------------+--------------------+------------+
|  1|          EWR|      Newark Airport|         EWR|
|  2|       Queens|         Jamaica Bay|   Boro Zone|
|  3|        Bronx|Allerton/Pelham G...|   Boro Zone|
|  4|    Manhattan|       Alphabet City| Yellow Zone|
|  5|Staten Island|       Arden Heights|   Boro Zone|
+---+-------------+--------------------+------------+
only showing top 5 rows

2024-10-11 18:30:09,435 INFO spark.SparkContext: Invoking stop() from shutdown hook
2024-10-11 18:30:09,448 INFO server.AbstractConnector: Stopped Spark@b64fa35{HTTP/1.1,[ht
2024-10-11 18:30:09,452 INFO ui.SparkUI: Stopped Spark web UI at http://q2-721e7e927cd7f8
```

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

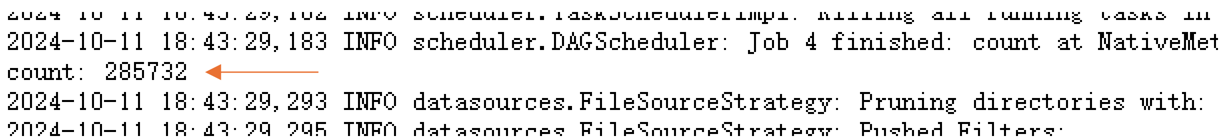
3) **(4 points)** Create a graph using the vertices and edges. Show 10 samples of the graph DataFrame with columns **"src"**, **"edge"**, and **"dst"**. **Do not truncate the names of fields**. You need to give a screenshot of your results in your report. For example,



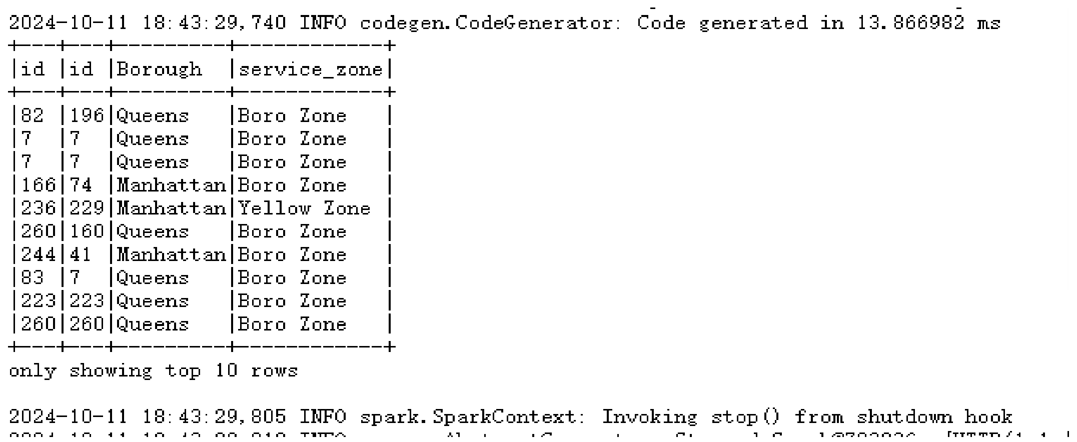
```
2024-10-11 18:35:09,470 INFO scheduler.DAGScheduler: Job 4 finished: showString at NativeMethodAccessorImpl.java:0, took 0.163509 s
2024-10-11 18:35:09,487 INFO codegen.CodeGenerator: Code generated in 11.91692 ms
+-----------------------------------------------------+----------+------------------------------------------------------------+
|src                                                  |edge      |dst                                                         |
+-----------------------------------------------------+----------+------------------------------------------------------------+
|[82, Queens, Elmhurst, Boro Zone]                    |[82, 196] |[196, Queens, Rego Park, Boro Zone]                         |
|[7, Queens, Astoria, Boro Zone]                      |[7, 7]    |[7, Queens, Astoria, Boro Zone]                             |
|[7, Queens, Astoria, Boro Zone]                      |[7, 7]    |[7, Queens, Astoria, Boro Zone]                             |
|[166, Manhattan, Morningside Heights, Boro Zone]     |[166, 74] |[74, Manhattan, East Harlem North, Boro Zone]               |
|[236, Manhattan, Upper East Side North, Yellow Zone] |[236, 229]|[229, Manhattan, Sutton Place/Turtle Bay North, Yellow Zone]|
|[75, Manhattan, East Harlem South, Boro Zone]        |[75, 235] |[235, Bronx, University Heights/Morris Heights, Boro Zone]  |
|[260, Queens, Woodside, Boro Zone]                   |[260, 160]|[160, Queens, Middle Village, Boro Zone]                    |
|[95, Queens, Forest Hills, Boro Zone]                |[95, 264] |[264, Unknown, NV, N/A]                                     |
|[244, Manhattan, Washington Heights South, Boro Zone]|[244, 41] |[41, Manhattan, Central Harlem, Boro Zone]                  |
|[83, Queens, Elmhurst/Maspeth, Boro Zone]            |[83, 7]   |[7, Queens, Astoria, Boro Zone]                             |
+-----------------------------------------------------+----------+------------------------------------------------------------+
only showing top 10 rows

2024-10-11 18:35:09,549 INFO spark.SparkContext: Invoking stop() from shutdown hook
2024-10-11 18:35:09,562 INFO server.AbstractConnector: Stopped Spark@7009362{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
```

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

4) **(5 points)** Count connected vertices (the **src** to **dst**) with the same Borough and service_zone. **Do not truncate the names of fields**. The table columns should include **'id'(src), 'id'(dst), 'Borough'**, and **'service_zone'**. You need to show the total number of connected vertices meeting the above condition and give a screenshot of 10 samples of that outcome. For example,



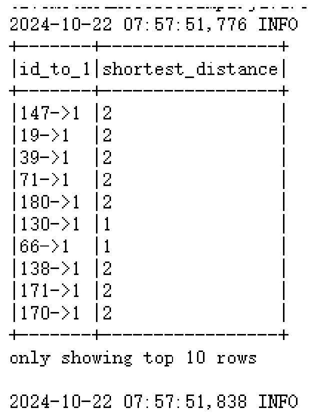
```
2024-10-11 18:43:29,182 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in
2024-10-11 18:43:29,183 INFO scheduler.DAGScheduler: Job 4 finished: count at NativeMet
count: 285732 ←
2024-10-11 18:43:29,293 INFO datasources.FileSourceStrategy: Pruning directories with:
2024-10-11 18:43:29,295 INFO datasources.FileSourceStrategy: Pushed Filters:
```



```
2024-10-11 18:43:29,740 INFO codegen.CodeGenerator: Code generated in 13.866982 ms
+---+---+---------+------------+
|id |id |Borough  |service_zone|
+---+---+---------+------------+
|82 |196|Queens   |Boro Zone   |
|7  |7  |Queens   |Boro Zone   |
|7  |7  |Queens   |Boro Zone   |
|166|74 |Manhattan|Boro Zone   |
|236|229|Manhattan|Yellow Zone |
|260|160|Queens   |Boro Zone   |
|244|41 |Manhattan|Boro Zone   |
|83 |7  |Queens   |Boro Zone   |
|223|223|Queens   |Boro Zone   |
|260|260|Queens   |Boro Zone   |
+---+---+---------+------------+
only showing top 10 rows

2024-10-11 18:43:29,805 INFO spark.SparkContext: Invoking stop() from shutdown hook
```

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

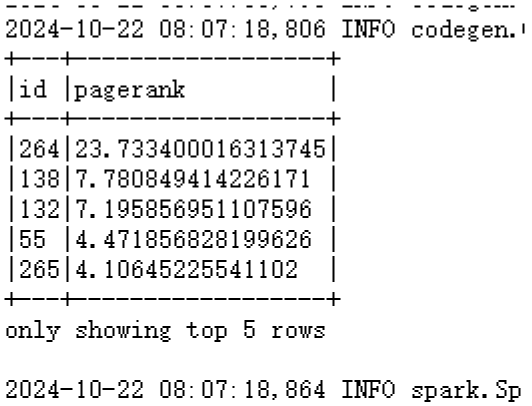
5) **(4 points)** Find the shortest paths from all other vertices (**Other LocationID**) to vertex 1 (**LocationID =1**). Show 10 samples from your result. The table columns should include **'id_to_1'** and **'shortest_distance'**. **Do not truncate the names of fields**. You need to give a screenshot of your 10 samples in your report. For example,



```
2024-10-22 07:57:51,776 INFO
+-------+-----------------+
|id_to_1|shortest_distance|
+-------+-----------------+
|147->1 |2                |
|19->1  |2                |
|39->1  |2                |
|71->1  |2                |
|180->1 |2                |
|130->1 |1                |
|66->1  |1                |
|138->1 |2                |
|171->1 |2                |
|170->1 |2                |
+-------+-----------------+
only showing top 10 rows

2024-10-22 07:57:51,838 INFO
```

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

6) **(5 points)** Perform page ranking on the graph dataframe. You will set the **'resetProbability'** to 0.17 and **'tol'** to 0.01. And sort **vertices (id)** by descending according to the value of PageRank. Show the top 5 results. You need to give a screenshot of your results in your report. The table columns should include **'id', 'pagerank'**. For example:



```
2024-10-22 08:07:18,806 INFO codegen.
+---+------------------+
|id |pagerank          |
+---+------------------+
|264|23.733400016313745|
|138|7.780849414226171 |
|132|7.195856951107596 |
|55 |4.471856828199626 |
|265|4.10645225541102  |
+---+------------------+
only showing top 5 rows

2024-10-22 08:07:18,864 INFO spark.Sp
```

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