Coursework for ECS765P - Big Data Processing

Task 3: Analysis of Chicago Taxi Trips Data (25 points)

References:

1. Ramamonjison, R. (2025). Apache Spark Graph Processing `https://search.library.qmul.ac.uk/iii/encore/record/C__Rb2519539__SApache%20Spark%20Graph%20Processing%20By%20Rindra%20Ramamonjison__Orightresult__U__X2?lang=eng&suite=def`
2. Week 6 Lecture, Lab and related resources.

Dataset Description

The dataset contains information about taxi trips in Chicago starting from January 2024. It includes details such as trip start and end times, trip duration, distance, fare, tips, and pickup and dropoff locations. The data is anonymized to protect privacy. The dataset used in this coursework is `chicago_taxi_trips.csv` from `//data-repository-bkt/ECS765/Chicago_Taxitrips/` directory.

Purpose

The purpose of this task is to analyze taxi trip data to gain insights into trip patterns, fare distribution, and the connectivity between different community areas in Chicago.

Column Descriptions

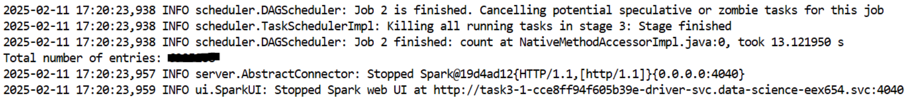
Column Name	**Description**
Trip ID	A unique identifier for the trip.
Taxi ID	A unique identifier for the taxi.
Trip Start Timestamp	When the trip started, rounded to the nearest 15 minutes.
Trip End Timestamp	When the trip ended, rounded to the nearest 15 minutes.

Column Name	Description
Trip Seconds	Time of the trip in seconds.
Trip Miles	Distance of the trip in miles.
Pickup Census Tract	The Census Tract where the trip began.
Dropoff Census Tract	The Census Tract where the trip ended.
Pickup Community Area	The Community Area where the trip began.
Dropoff Community Area	The Community Area where the trip ended.
Fare	The fare for the trip.
Tips	The tip for the trip.
Tolls	The tolls for the trip.
Extras	Extra charges for the trip.
Trip Total	Total cost of the trip.
Payment Type	Type of payment for the trip.
Company	The taxi company.
Pickup Centroid Latitude	Latitude of the pickup location.
Pickup Centroid Longitude	Longitude of the pickup location.
Pickup Centroid Location	Location of the pickup centroid.
Dropoff Centroid Latitude	Latitude of the dropoff location.
Dropoff Centroid Longitude	Longitude of the dropoff location.
Dropoff Centroid Location	Location of the dropoff centroid.

Questions

1. **Load Data (2 points)**

   - Load the taxi trips dataset into a DataFrame.
   - Print the total number of entries in the DataFrame.
   - Include a screenshot of your results in your report. For example:

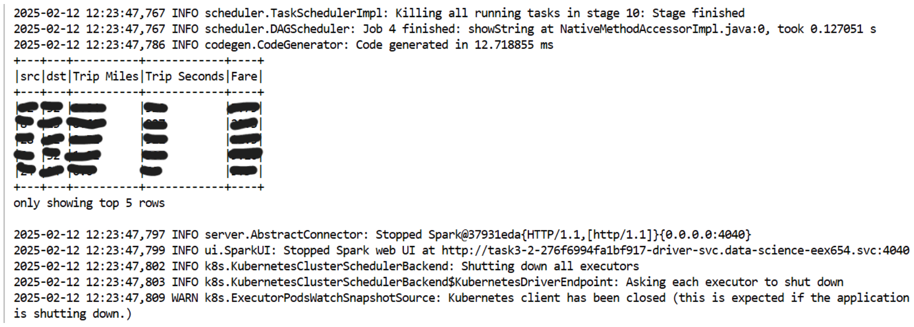


```
2025-02-11 17:20:23,938 INFO scheduler.DAGScheduler: Job 2 is finished. Cancelling potential speculative or zombie tasks for this job
2025-02-11 17:20:23,938 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 3: Stage finished
2025-02-11 17:20:23,938 INFO scheduler.DAGScheduler: Job 2 finished: count at NativeMethodAccessorImpl.java:0, took 13.121950 s
Total number of entries: [redacted]
2025-02-11 17:20:23,957 INFO server.AbstractConnector: Stopped Spark@19d4ad12{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2025-02-11 17:20:23,959 INFO ui.SparkUI: Stopped Spark web UI at http://task3-1-cce8ff94f605b39e-driver-svc.data-science-eex654.svc:4040
```

2. **Define Schemas and Construct DataFrames (2 points)**

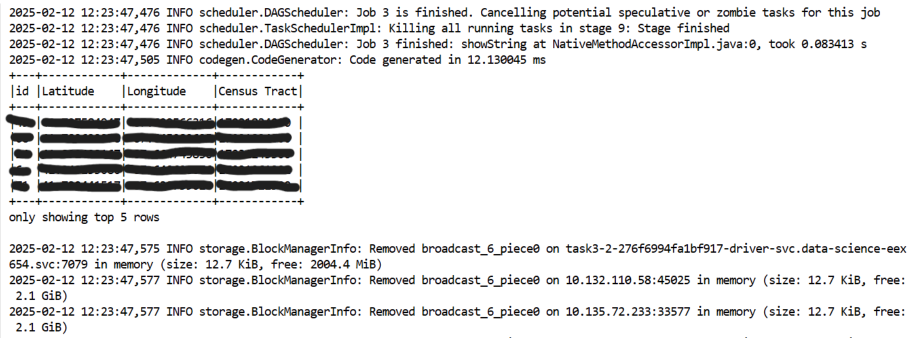
   - Define the `StructType` for the vertexSchema and edgeSchema.

- Construct the vertices DataFrame using the `Pickup Community Area` and `Dropoff Community Area` fields as the vertex information. Include the following fields in the vertices DataFrame:
  - `id`: Community area ID (either Pickup or Dropoff Community Area)
  - `Latitude`: Latitude of the centroid location (Pickup Centroid Latitude or Dropoff Centroid Latitude)
  - `Longitude`: Longitude of the centroid location (Pickup Centroid Longitude or Dropoff Centroid Longitude)
  - `Census Tract`: Census tract information (Pickup Census Tract or Dropoff Census Tract)
- Construct the edges DataFrame using the `Pickup Community Area` and `Dropoff Community Area` fields as the edge information. Include the following fields in the edges DataFrame:
  - `src`: Pickup Community Area
  - `dst`: Dropoff Community Area
  - `Trip Miles`: Distance of the trip
  - `Trip Seconds`: Duration of the trip
  - `Fare`: Fare amount
- Display 5 samples of both the edges and vertices DataFrames, ensuring that field names are not truncated.
- Include a screenshot of the results in your report. For example:



```
2025-02-12 12:23:47,767 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 10: Stage finished
2025-02-12 12:23:47,767 INFO scheduler.DAGScheduler: Job 4 finished: showString at NativeMethodAccessorImpl.java:0, took 0.127051 s
2025-02-12 12:23:47,786 INFO codegen.CodeGenerator: Code generated in 12.718855 ms
+---+---+----------+------------+----+
|src|dst|Trip Miles|Trip Seconds|Fare|
+---+---+----------+------------+----+
|[redacted]|[redacted]|[redacted]      |[redacted]         |[redacted]|
|[redacted]|[redacted]|[redacted]      |[redacted]         |[redacted]|
|[redacted]|[redacted]|[redacted]      |[redacted]         |[redacted]|
|[redacted]|[redacted]|[redacted]      |[redacted]         |[redacted]|
|[redacted]|[redacted]|[redacted]      |[redacted]         |[redacted]|
+---+---+----------+------------+----+
only showing top 5 rows

2025-02-12 12:23:47,797 INFO server.AbstractConnector: Stopped Spark@37931eda{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2025-02-12 12:23:47,799 INFO ui.SparkUI: Stopped Spark web UI at http://task3-2-276f6994fa1bf917-driver-svc.data-science-eex654.svc:4040
2025-02-12 12:23:47,802 INFO k8s.KubernetesClusterSchedulerBackend: Shutting down all executors
2025-02-12 12:23:47,803 INFO k8s.KubernetesClusterSchedulerBackend$KubernetesDriverEndpoint: Asking each executor to shut down
2025-02-12 12:23:47,809 WARN k8s.ExecutorPodsWatchSnapshotSource: Kubernetes client has been closed (this is expected if the application
is shutting down.)
```

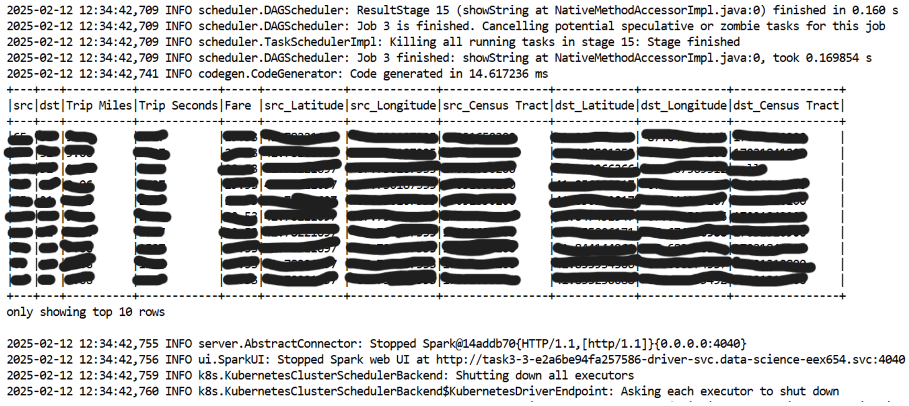


```
2025-02-12 12:23:47,476 INFO scheduler.DAGScheduler: Job 3 is finished. Cancelling potential speculative or zombie tasks for this job
2025-02-12 12:23:47,476 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 9: Stage finished
2025-02-12 12:23:47,476 INFO scheduler.DAGScheduler: Job 3 finished: showString at NativeMethodAccessorImpl.java:0, took 0.083413 s
2025-02-12 12:23:47,505 INFO codegen.CodeGenerator: Code generated in 12.130045 ms
+---+------------+------------+-----------+
|id |Latitude    |Longitude   |Census Tract|
+---+------------+------------+-----------+
|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|
+---+------------+------------+-----------+
only showing top 5 rows

2025-02-12 12:23:47,575 INFO storage.BlockManagerInfo: Removed broadcast_6_piece0 on task3-2-276f6994fa1bf917-driver-svc.data-science-eex
654.svc:7079 in memory (size: 12.7 KiB, free: 2004.4 MiB)
2025-02-12 12:23:47,577 INFO storage.BlockManagerInfo: Removed broadcast_6_piece0 on 10.132.110.58:45025 in memory (size: 12.7 KiB, free:
 2.1 GiB)
2025-02-12 12:23:47,577 INFO storage.BlockManagerInfo: Removed broadcast_6_piece0 on 10.135.72.233:33577 in memory (size: 12.7 KiB, free:
 2.1 GiB)
```

3. **Create Graph and Display Samples (2 points)**

    - Create a graph using the vertices and edges DataFrames. In this context:
        - **Vertices** represent the nodes in the graph, which correspond to the community areas where taxi trips start or end.
        - **Edges** represent the connections between these nodes, which correspond to the taxi trips between different community areas.
    - Show 10 samples of the graph DataFrame with columns `src`, `dst`, `Trip Miles`, `Trip Seconds`, `Fare`, `src_Latitude`, `src_Longitude`, `src_Census Tract`, `dst_Latitude`, `dst_Longitude`, and `dst_Census Tract`.
    - Include a screenshot of your results. For Example:



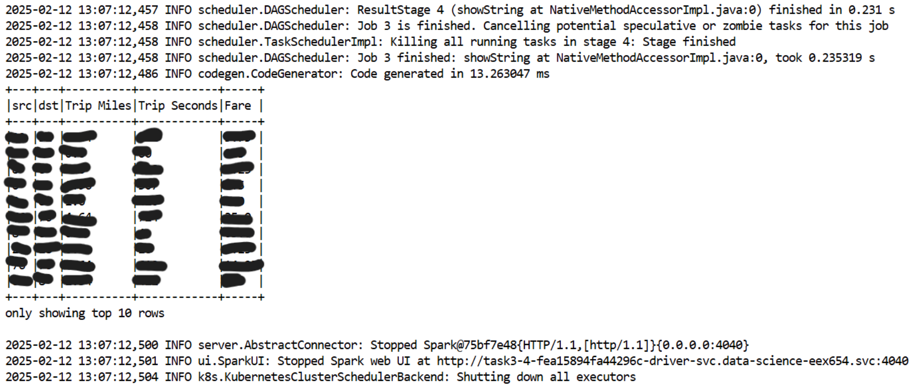
```
2025-02-12 12:34:42,709 INFO scheduler.DAGScheduler: ResultStage 15 (showString at NativeMethodAccessorImpl.java:0) finished in 0.160 s
2025-02-12 12:34:42,709 INFO scheduler.DAGScheduler: Job 3 is finished. Cancelling potential speculative or zombie tasks for this job
2025-02-12 12:34:42,709 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 15: Stage finished
2025-02-12 12:34:42,709 INFO scheduler.DAGScheduler: Job 3 finished: showString at NativeMethodAccessorImpl.java:0, took 0.169854 s
2025-02-12 12:34:42,741 INFO codegen.CodeGenerator: Code generated in 14.617236 ms
+---+---+----------+------------+-----+------------+-------------+----------------+------------+-------------+----------------+
|src|dst|Trip Miles|Trip Seconds|Fare |src_Latitude|src_Longitude|src_Census Tract|dst_Latitude|dst_Longitude|dst_Census Tract|
+---+---+----------+------------+-----+------------+-------------+----------------+------------+-------------+----------------+
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
+---+---+----------+------------+-----+------------+-------------+----------------+------------+-------------+----------------+
only showing top 10 rows

2025-02-12 12:34:42,755 INFO server.AbstractConnector: Stopped Spark@14addb70{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2025-02-12 12:34:42,756 INFO ui.SparkUI: Stopped Spark web UI at http://task3-3-e2a6be94fa257586-driver-svc.data-science-eex654.svc:4040
2025-02-12 12:34:42,759 INFO k8s.KubernetesClusterSchedulerBackend: Shutting down all executors
2025-02-12 12:34:42,760 INFO k8s.KubernetesClusterSchedulerBackend$KubernetesDriverEndpoint: Asking each executor to shut down
```

4. **Connected Community Areas Analysis (3 points)**

    - Count the number of edges where both the source (src) and destination (dst) vertices belong to the same community area.

- Display the total number of such connected vertices and 10 samples of the outcome.
- Include a screenshot of your results. For Example:

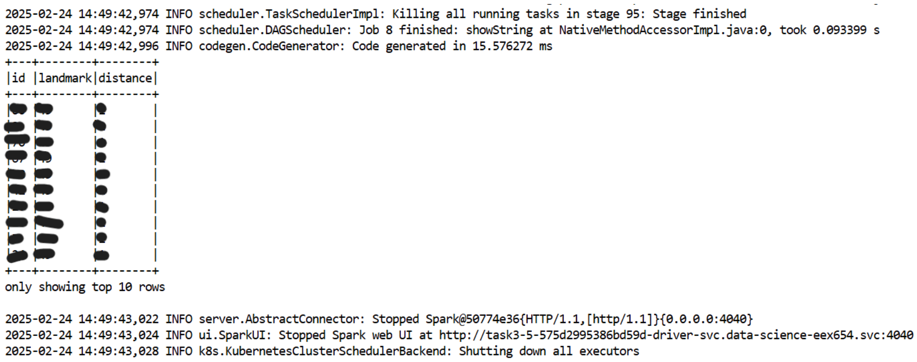


```
2025-02-12 13:07:12,457 INFO scheduler.DAGScheduler: ResultStage 4 (showString at NativeMethodAccessorImpl.java:0) finished in 0.231 s
2025-02-12 13:07:12,458 INFO scheduler.DAGScheduler: Job 3 is finished. Cancelling potential speculative or zombie tasks for this job
2025-02-12 13:07:12,458 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 4: Stage finished
2025-02-12 13:07:12,458 INFO scheduler.DAGScheduler: Job 3 finished: showString at NativeMethodAccessorImpl.java:0, took 0.235319 s
2025-02-12 13:07:12,486 INFO codegen.CodeGenerator: Code generated in 13.263047 ms
+---+---+----------+------------+-----+
|src|dst|Trip Miles|Trip Seconds|Fare |
+---+---+----------+------------+-----+
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|[redacted]|[redacted]|
+---+---+----------+------------+-----+
only showing top 10 rows

2025-02-12 13:07:12,500 INFO server.AbstractConnector: Stopped Spark@75bf7e48{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2025-02-12 13:07:12,501 INFO ui.SparkUI: Stopped Spark web UI at http://task3-4-fea15894fa44296c-driver-svc.data-science-eex654.svc:4040
2025-02-12 13:07:12,504 INFO k8s.KubernetesClusterSchedulerBackend: Shutting down all executors
```

5. **Shortest Paths (5 points)**

   - Find the shortest paths from all other vertices (community areas) to a specific vertex (e.g., community area 49). Use BFS (breadth-first search) algorithm to find the shortest paths.
   - Show 10 samples from your result.
   - Include a screenshot of your results. For Example:



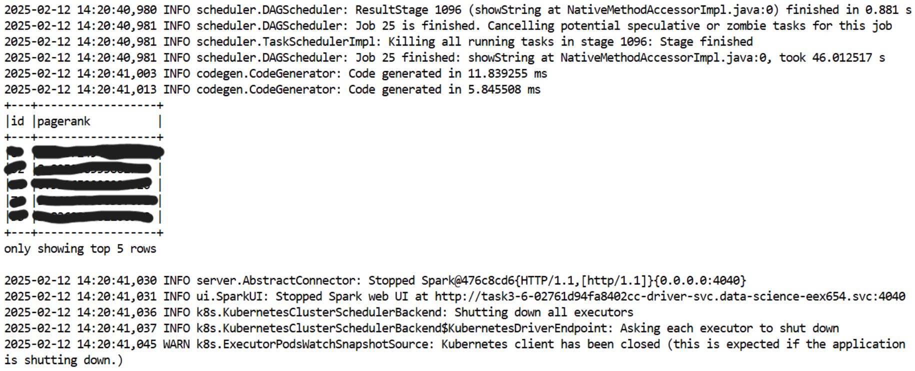
```
2025-02-24 14:49:42,974 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 95: Stage finished
2025-02-24 14:49:42,974 INFO scheduler.DAGScheduler: Job 8 finished: showString at NativeMethodAccessorImpl.java:0, took 0.093399 s
2025-02-24 14:49:42,996 INFO codegen.CodeGenerator: Code generated in 15.576272 ms
+---+--------+--------+
|id |landmark|distance|
+---+--------+--------+
|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|
|[redacted]|[redacted]|[redacted]|
+---+--------+--------+
only showing top 10 rows

2025-02-24 14:49:43,022 INFO server.AbstractConnector: Stopped Spark@50774e36{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2025-02-24 14:49:43,024 INFO ui.SparkUI: Stopped Spark web UI at http://task3-5-575d2995386bd59d-driver-svc.data-science-eex654.svc:4040
2025-02-24 14:49:43,028 INFO k8s.KubernetesClusterSchedulerBackend: Shutting down all executors
```

6. **PageRank Analysis (6 points)**

   - Perform PageRank on the graph DataFrame with `resetProbability` set to 0.15 and `tol` set to 0.01.
   - Deduplicate the vertices DataFrame to ensure unique vertices.
   - Create the GraphFrame using the deduplicated vertices and the edges.

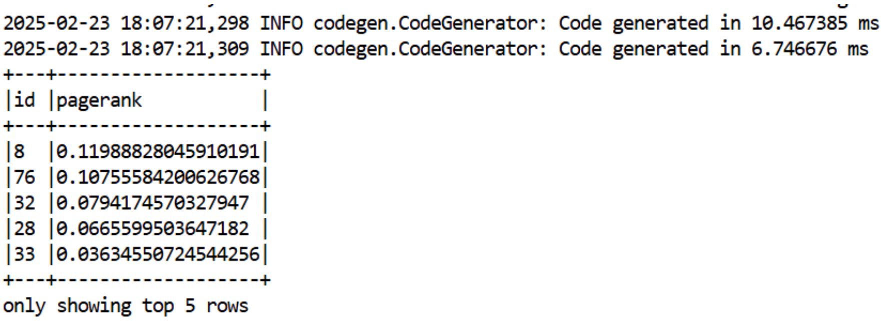
- Sort vertices by descending PageRank value and show the top 5 results.
- Include a screenshot of your results. For Example:



```
2025-02-12 14:20:40,980 INFO scheduler.DAGScheduler: ResultStage 1096 (showString at NativeMethodAccessorImpl.java:0) finished in 0.881 s
2025-02-12 14:20:40,981 INFO scheduler.DAGScheduler: Job 25 is finished. Cancelling potential speculative or zombie tasks for this job
2025-02-12 14:20:40,981 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 1096: Stage finished
2025-02-12 14:20:40,981 INFO scheduler.DAGScheduler: Job 25 finished: showString at NativeMethodAccessorImpl.java:0, took 46.012517 s
2025-02-12 14:20:41,003 INFO codegen.CodeGenerator: Code generated in 11.839255 ms
2025-02-12 14:20:41,013 INFO codegen.CodeGenerator: Code generated in 5.845508 ms
+---+-----------------+
|id |pagerank         |
+---+-----------------+
[redacted]
+---+-----------------+
only showing top 5 rows

2025-02-12 14:20:41,030 INFO server.AbstractConnector: Stopped Spark@476c8cd6{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2025-02-12 14:20:41,031 INFO ui.SparkUI: Stopped Spark web UI at http://task3-6-02761d94fa8402cc-driver-svc.data-science-eex654.svc:4040
2025-02-12 14:20:41,036 INFO k8s.KubernetesClusterSchedulerBackend: Shutting down all executors
2025-02-12 14:20:41,037 INFO k8s.KubernetesClusterSchedulerBackend$KubernetesDriverEndpoint: Asking each executor to shut down
2025-02-12 14:20:41,045 WARN k8s.ExecutorPodsWatchSnapshotSource: Kubernetes client has been closed (this is expected if the application
is shutting down.)
```

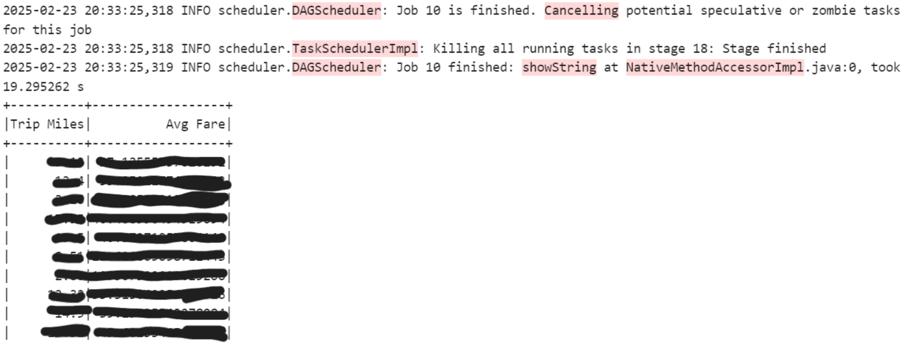
- Modify the PageRank algorithm to consider the weights of the edges (e.g., trip distance) instead of treating all edges equally. This will give more importance to vertices connected by higher-weight edges.
- Define the vertices by selecting the unique community areas from both the pickup and dropoff locations, ensuring no duplicates.
- Construct the edges by selecting the pickup and dropoff community areas along with the trip distance, and rename the trip distance to represent the weight of the edge.
- Compute the total outgoing weight for each node by aggregating the weights of all outgoing edges from each source node.
- Normalize the weights of the edges by dividing each edge's weight by the total outgoing weight of its source node.
- Create the GraphFrame using the deduplicated vertices and the normalized edges.
- Sort vertices by descending PageRank value and show the top 5 results.
- Include a screenshot of your results. For Example:



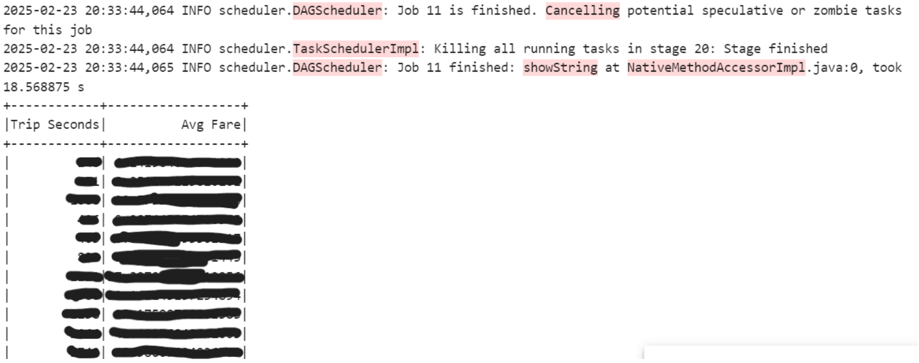
```
2025-02-23 18:07:21,298 INFO codegen.CodeGenerator: Code generated in 10.467385 ms
2025-02-23 18:07:21,309 INFO codegen.CodeGenerator: Code generated in 6.746676 ms
+---+-------------------+
|id |pagerank           |
+---+-------------------+
|8  |0.11988828045910191|
|76 |0.10755584200626768|
|32 |0.0794174570327947 |
|28 |0.0665599503647182 |
|33 |0.03634550724544256|
+---+-------------------+
only showing top 5 rows
```

7. **Fare Analysis with GraphFrames (5 points)**

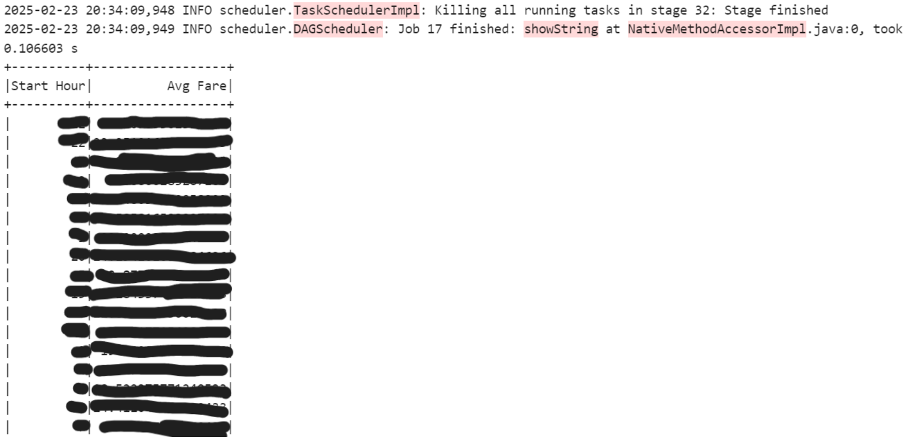
   - Analyze the distribution of fares using a `GraphFrame` representation. Create a graph where the community areas (pickup and dropoff) are the vertices, and taxi trips (with attributes such as trip distance, duration, and fare) are the edges.
   - Plot a histogram showing the fare distribution and analyze the factors influencing the fare amount, such as trip distance, duration, and time of day.
   - Using the `GraphFrame`, identify and discuss how node and edge attributes (e.g., community areas, trip distance, duration) might correlate with the fare.
   - Include a screenshot of your result displaying the fare distribution with respect to trip distance, trip duration and time of the day. For example:



```
2025-02-23 20:33:25,318 INFO scheduler.DAGScheduler: Job 10 is finished. Cancelling potential speculative or zombie tasks
for this job
2025-02-23 20:33:25,318 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 18: Stage finished
2025-02-23 20:33:25,319 INFO scheduler.DAGScheduler: Job 10 finished: showString at NativeMethodAccessorImpl.java:0, took
19.295262 s
+----------+------------------+
|Trip Miles|          Avg Fare|
+----------+------------------+
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
```



```
2025-02-23 20:33:44,064 INFO scheduler.DAGScheduler: Job 11 is finished. Cancelling potential speculative or zombie tasks
for this job
2025-02-23 20:33:44,064 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 20: Stage finished
2025-02-23 20:33:44,065 INFO scheduler.DAGScheduler: Job 11 finished: showString at NativeMethodAccessorImpl.java:0, took
18.568875 s
+------------+------------------+
|Trip Seconds|          Avg Fare|
+------------+------------------+
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
```



```
2025-02-23 20:34:09,948 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 32: Stage finished
2025-02-23 20:34:09,949 INFO scheduler.DAGScheduler: Job 17 finished: showString at NativeMethodAccessorImpl.java:0, took
0.106603 s
+----------+------------------+
|Start Hour|          Avg Fare|
+----------+------------------+
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
|[redacted]|[redacted]|
```