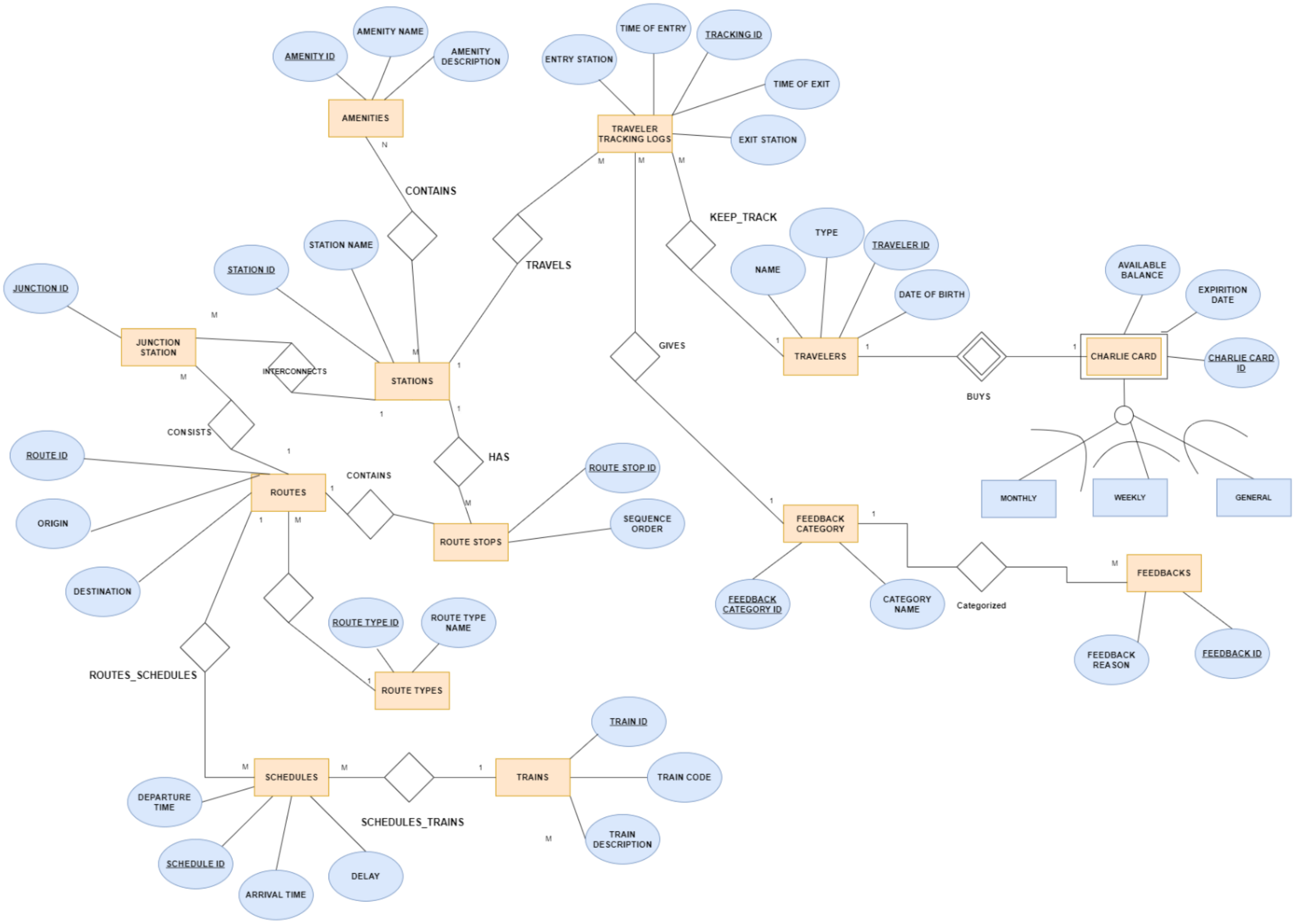
## Introduction:

The Massachusetts Bay Transportation Authority (MBTA), commonly known as the "T," is a comprehensive public transportation system serving the Greater Boston region. Established in 1964, MBTA stands as one of the largest transit agencies in the United States, connecting neighborhoods, suburbs, and communities. It plays a vital role in the daily lives of Boston residents and visitors exploring the city.

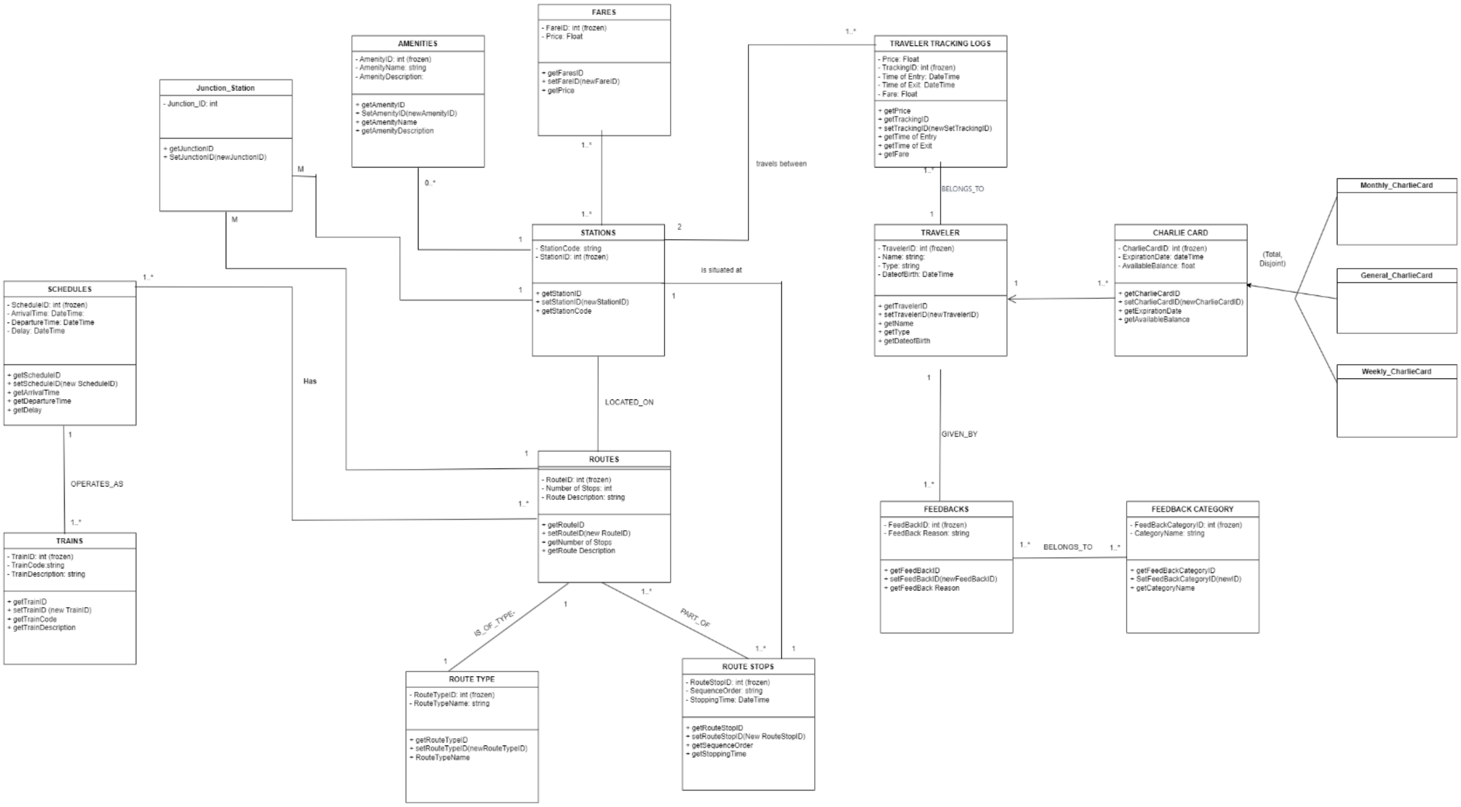
The main aim of our project is to provide a scalable and reliable solution that supports day to day operations and decision-making processes of MBTA along with assuring data accuracy, consistency and security. It would efficiently store and manage data of fares, vehicles, passenger information, routes, stations and schedules. Some of the use cases we would solve using this management system are route management which includes the planning and management of various transportation routes in a very efficient manner. It would help in assigning stations and platforms to route, designing the fare structure of each route, also creating route maps for easy reference to the end users. Second feature we can enhance is station and platform management which includes tracking platform-specific schedules. Other one is schedule management which includes monitoring of daily departure and arrival times and handling delays or updates in the schedule of train. Other feature is, passenger tracking, which includes tracking ticket sales, aligning tickets with passengers and trips, enabling passengers to tap the card. Lastly, amenity management which includes aligning amenities with specific station and updating amenity details. The main problem with the existing MBTA management system is that we have to pay the same amount regardless of you dropping off at any station. We are also going to add this feature for tracking the start station and end station and calculating the fare accordingly. Also, there will be analysis of overcrowded stations which would help give an insight and better idea for future infrastructure building. . This system will efficiently manage and store data related to fares, vehicles, passenger information, routes, stations, and schedules. It will address several key use cases, including route management, station and platform management, schedule management, passenger tracking, amenity management, fare calculation based on start and end stations, and analysis of overcrowded stations to inform future infrastructure decisions

MBTA could consider introducing a carpool service, allowing people to hire taxis at a reasonable cost and providing an alternative to the potentially lengthy subway commutes. This carpool service would share the same pickup locations as the subway stations, with restrictions in place to limit drop-off points to designated subway stations. In other words, passengers would be required to travel to and from subway stops, eliminating the option to teleport to locations beyond these stations. The introduction of a carpool service under the MBTA umbrella offers numerous benefits. It provides users with greater flexibility and convenience, making it an good option for those seeking efficient last-mile connectivity to their destinations. Furthermore, this initiative could serve as a revenue-generating source for the government, which is crucial given the substantial financial losses incurred in running MBTA as a public service.

# EER Diagram:


AMENITY ID
AMENITY NAME
AMENITY DESCRIPTION
AMENITIES
N
CONTAINS
ENTRY STATION
TIME OF ENTRY
TRACKING ID
TIME OF EXIT
TRAVELER TRACKING LOGS
EXIT STATION
M
M
M
TRAVELS
KEEP_TRACK
TYPE
TRAVELER ID
NAME
DATE OF BIRTH
AVAILABLE BALANCE
EXPIRITION DATE
JUNCTION ID
STATION ID
STATION NAME
M
JUNCTION STATION
M
INTERCONNECTS
M
STATIONS
1
1
1
GIVES
1
TRAVELERS
1
BUYS
1
CHARLIE CARD
CHARLIE CARD ID
CONSISTS
1
HAS
ROUTE ID
CONTAINS
ROUTES
1
ROUTE STOP ID
M
ROUTE STOPS
SEQUENCE ORDER
ORIGIN
1
M
MONTHLY
WEEKLY
GENERAL
1
FEEDBACK CATEGORY
1
DESTINATION
FEEDBACK CATEGORY ID
CATEGORY NAME
Categorized
M
FEEDBACKS
ROUTE TYPE ID
ROUTE TYPE NAME
ROUTES_SCHEDULES
1
ROUTE TYPES
FEEDBACK REASON
FEEDBACK ID
TRAIN ID
M
SCHEDULES
M
1
TRAINS
TRAIN CODE
DEPARTURE TIME
SCHEDULES_TRAINS
M
TRAIN DESCRIPTION
SCHEDULE ID
ARRIVAL TIME
DELAY

# UML Class Diagram:

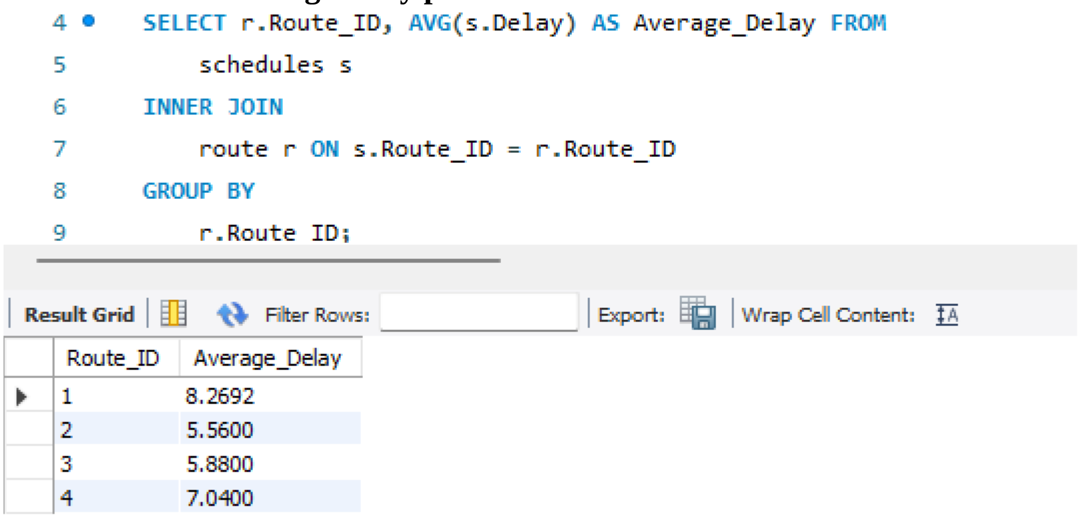


## Relational Model:

1. **Amenities** (**<u>Amenity_ID</u>**, Amenity_Name, Amenity_Description)
2. **Amenity_station** (*Amenity_ID*[FK], *Station_ID*[FK])
3. **CharlieCard** (**<u>CC_ID</u>**, *Traveler_ID*[FK], AvailableBalance, Expiry_Date)
4. **Fare** (**<u>Fare_ID</u>**, Price)
5. **Feedback** (**<u>Feedback_ID</u>**, Feedback_Reason, Feedback, *Category_ID*[FK])
6. **Feedbackcategory** (**<u>FeedbackCategory_ID</u>**, Category_Name)
7. **General_charliecard** (*CC_ID*[FK], *Traveler_ID*[FK])
8. **Junction_station** (**<u>Junction_ID</u>**, *Station_ID*[FK], *Route_ID*[FK])
9. **Monthly_Charliecard** (*CC_ID*[FK], *Traveler_ID*[FK])
10. **Route** (**<u>Route_ID</u>**, Origin, Destination, *RouteType_ID*[FK])
11. **Route_Type** (**<u>RouteType_ID</u>**, RouteType_Name)
12. **Route_Stops** (**<u>RouteStop_ID</u>**, Sequence, *Station_ID*[FK], *Route_ID*[FK])
13. **Schedules** (**<u>Schedule_ID</u>**, Departure_Time, Arrival_Time, Delay, *Route_ID*[FK], *Train_ID*[FK])
14. **Stations** (**<u>Station_ID</u>**, Station_Name)
15. **Trains** (**<u>Train_ID</u>**, Train_Code, Train_Description)
16. **Travelers_TrackingLogs** (**<u>Tracking_ID</u>**, TimeOfEntry , TimeOfExit, *Entry_Station_ID*[FK], *Exit_Station_ID*[FK], *FeedbackCategory_ID*[FK], *Traveler_ID*[FK] )
17. **Weekly_Charliecard** (*CC_ID*[FK], *Traveler_ID*[FK])

## Implementation in SQL:

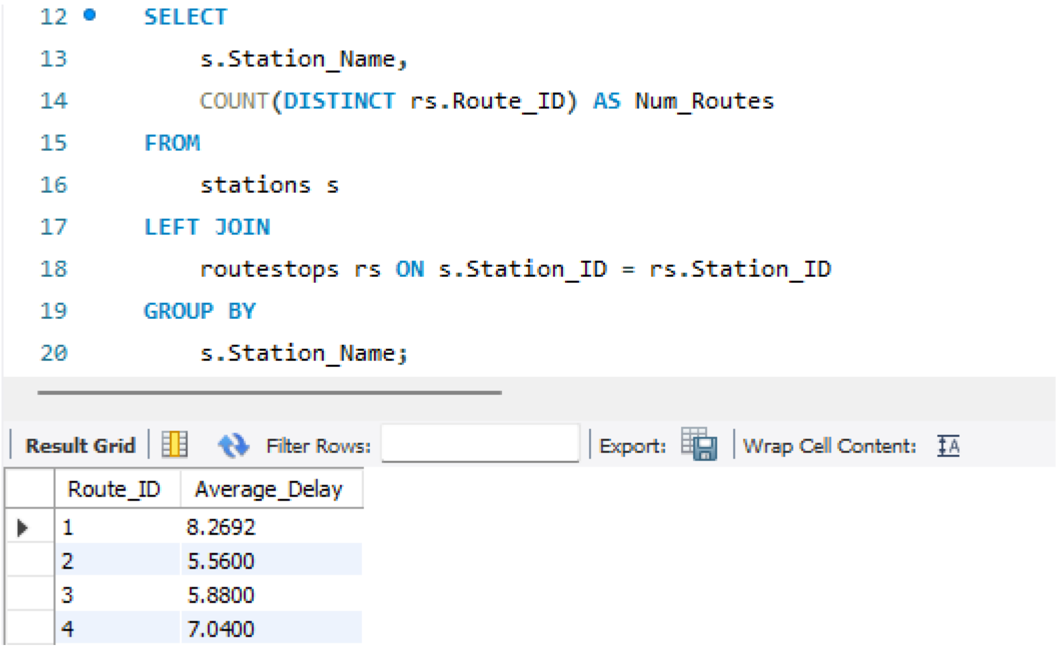
1. **To calculate the average delay per route**



```
SELECT r.Route_ID, AVG(s.Delay) AS Average_Delay FROM
    schedules s
INNER JOIN
    route r ON s.Route_ID = r.Route_ID
GROUP BY
    r.Route_ID;
```

| Route_ID | Average_Delay |
|---|---|
| 1 | 8.2692 |
| 2 | 5.5600 |
| 3 | 5.8800 |
| 4 | 7.0400 |

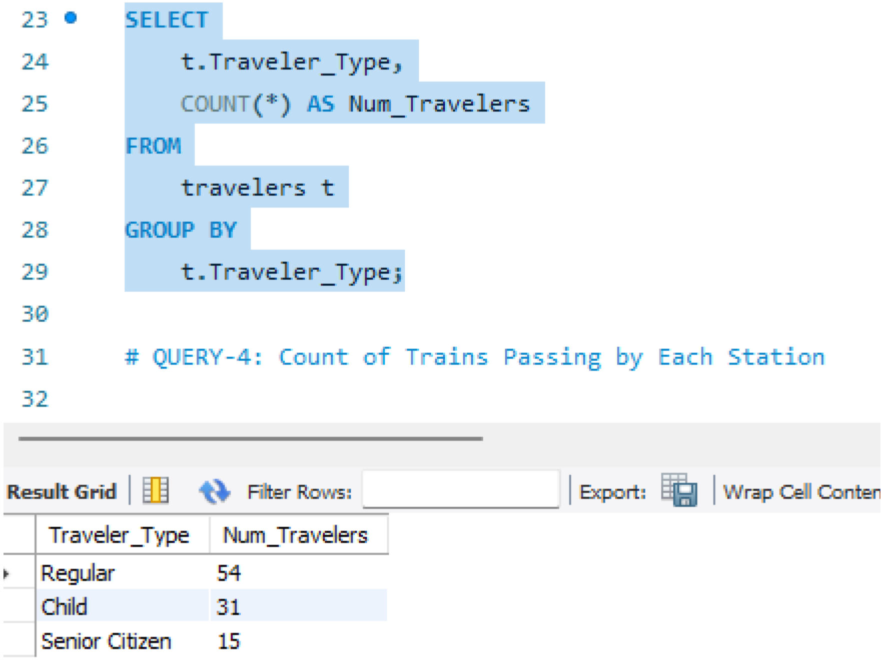
2. **To calculate the number of Routes per Station**



```
SELECT
    s.Station_Name,
    COUNT(DISTINCT rs.Route_ID) AS Num_Routes
FROM
    stations s
LEFT JOIN
    routestops rs ON s.Station_ID = rs.Station_ID
GROUP BY
    s.Station_Name;
```

| Route_ID | Average_Delay |
|---|---|
| 1 | 8.2692 |
| 2 | 5.5600 |
| 3 | 5.8800 |
| 4 | 7.0400 |

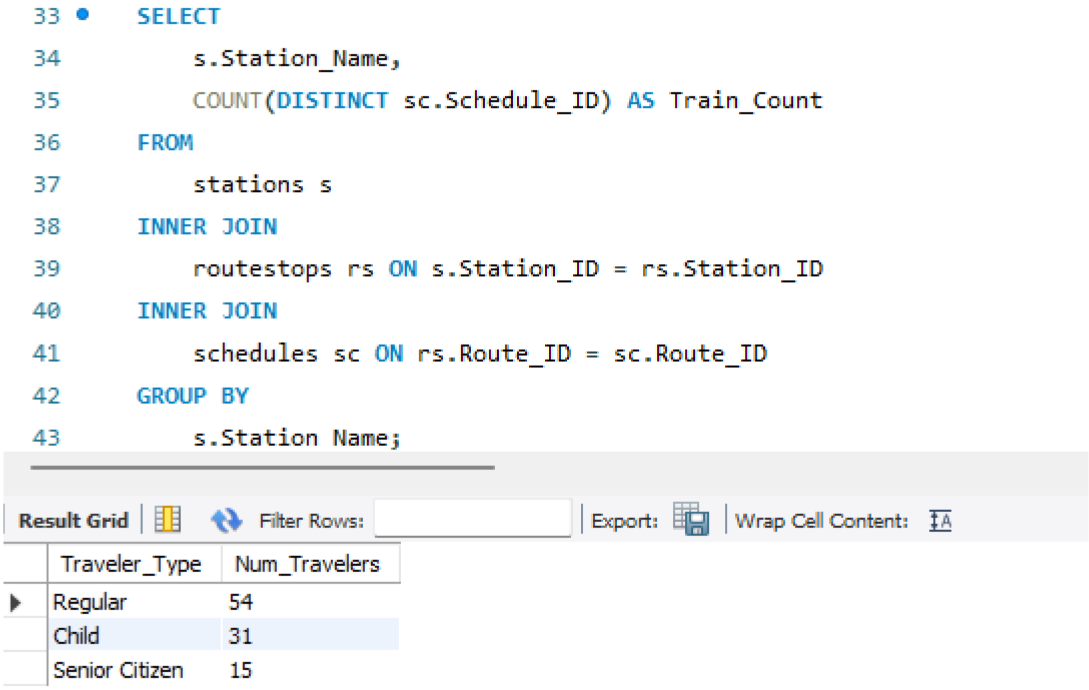
3. **Number of traveler's in each category**



```sql
SELECT
    t.Traveler_Type,
    COUNT(*) AS Num_Travelers
FROM
    travelers t
GROUP BY
    t.Traveler_Type;

# QUERY-4: Count of Trains Passing by Each Station

```

| Traveler_Type | Num_Travelers |
|---|---|
| Regular | 54 |
| Child | 31 |
| Senior Citizen | 15 |

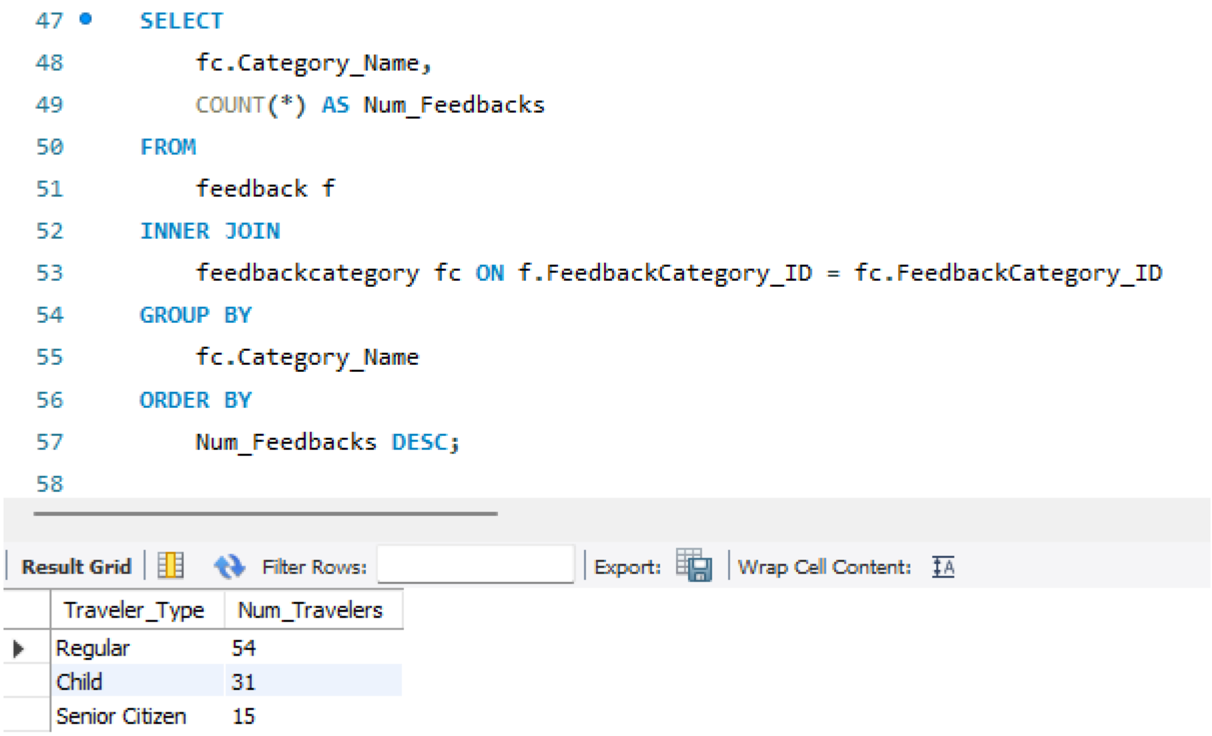
## 4. Count of number of trains at every station



```sql
SELECT
    s.Station_Name,
    COUNT(DISTINCT sc.Schedule_ID) AS Train_Count
FROM
    stations s
INNER JOIN
    routestops rs ON s.Station_ID = rs.Station_ID
INNER JOIN
    schedules sc ON rs.Route_ID = sc.Route_ID
GROUP BY
    s.Station_Name;
```

| Traveler_Type | Num_Travelers |
|---|---|
| Regular | 54 |
| Child | 31 |
| Senior Citizen | 15 |

## 5. Calculate the feedback category frequency



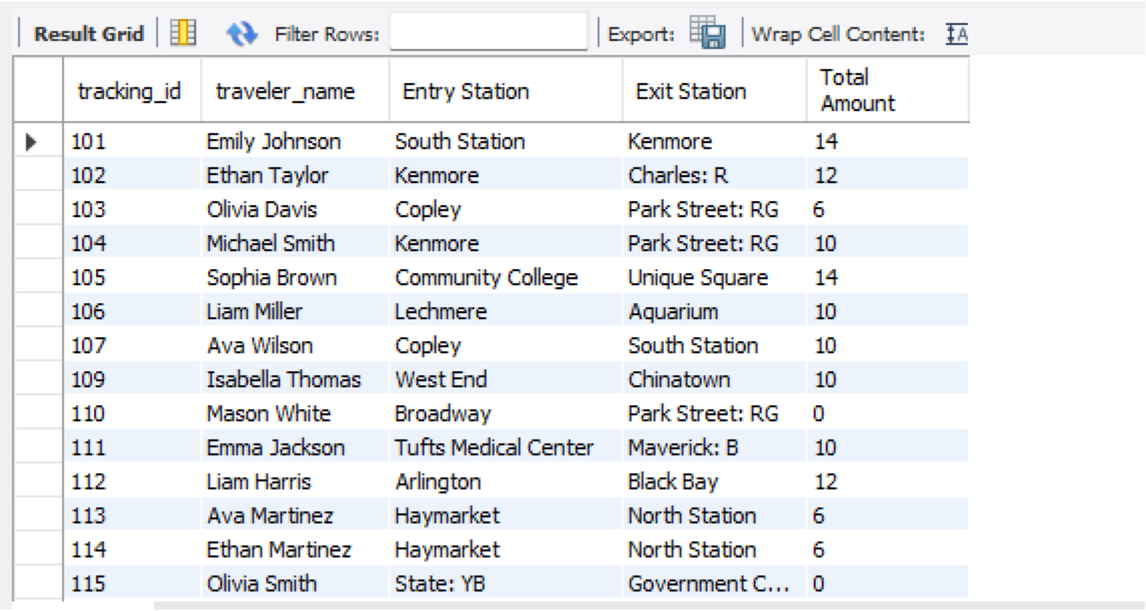
```sql
SELECT
    fc.Category_Name,
    COUNT(*) AS Num_Feedbacks
FROM
    feedback f
INNER JOIN
    feedbackcategory fc ON f.FeedbackCategory_ID = fc.FeedbackCategory_ID
GROUP BY
    fc.Category_Name
ORDER BY
    Num_Feedbacks DESC;
```

| Traveler_Type | Num_Travelers |
|---|---|
| Regular | 54 |
| Child | 31 |
| Senior Citizen | 15 |

### 6. Total fare calculated

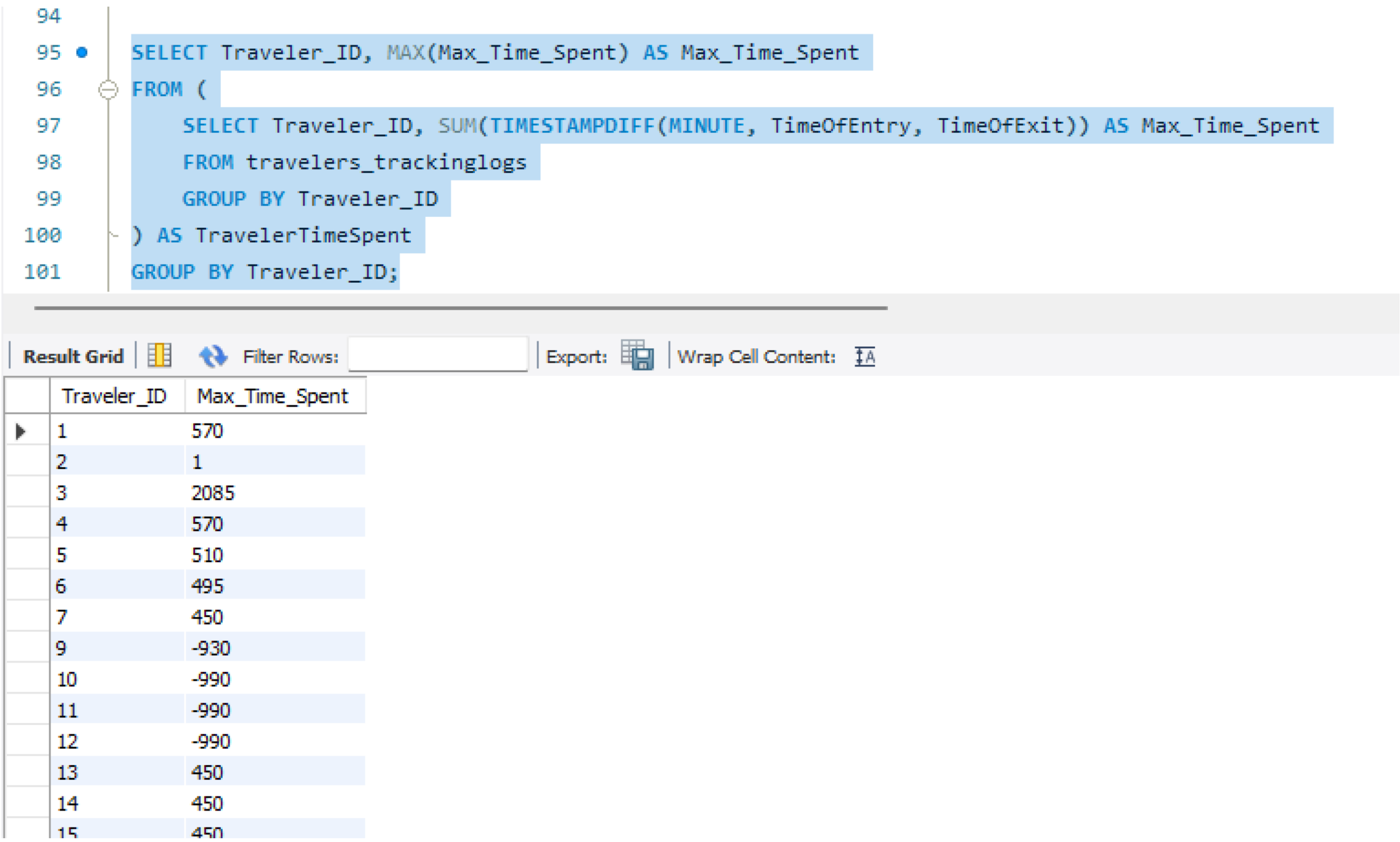
```
CASE
-- both are not junction
when ((select count(route_id) from routestops where station_id =ttl.entry_station_id)  = 1 and (select count(route_id) from routestops where station_id =ttl.exit_station_id ) = 1 )
   then
   case
     -- same routes
       when ( (select route_id from routestops where station_id =ttl.entry_station_id) = (select route_id from routestops where station_id =ttl.exit_station_id))
       then abs((select sequence from routestops where station_id =ttl.entry_station_id) - (select sequence from routestops where station_id =ttl.exit_station_id) )*2
       else
       -- diff routes
       abs((select sequence from routestops where station_id =ttl.entry_station_id ) - (select sequence from routestops where station_id =((select station_id from junction_station where route_id = (select route_id from routestops where station_id =ttl.entry_station_id))) and route_id = (select route_id from routestops where station_id =ttl.entry_station_id))) * 2
           +
       abs((select sequence from routestops where station_id =((select station_id from junction_station where route_id = (select route_id from routestops where station_id =ttl.exit_station_id))) and route_id = (select route_id from routestops where station_id =ttl.exit_station_id))-(select sequence from routestops where station_id =ttl.exit_station_id ))*2
  end
-- one of them is junction
when ((select count(route_id) from routestops where station_id =ttl.entry_station_id)  = 1 or (select count(route_id) from routestops where station_id =ttl.exit_station_id ) = 1 )
then
CASE
```

```
when ((select count(route_id) from routestops where station_id =ttl.entry_station_id)  = 1)
then abs((select sequence from routestops where station_id =ttl.entry_station_id) - (select sequence from routestops where station_id =ttl.exit_station_id and route_id = (select route_id from routestops where station_id = ttl.entry_station_id)) )*2
else abs((select sequence from routestops where station_id =ttl.exit_station_id) - (select sequence from routestops where station_id =ttl.entry_station_id and route_id = (select route_id from routestops where station_id = ttl.exit_station_id)) )*2

end
else 0
end
as 'Total Amount'
from travelers_trackinglogs ttl
inner join travelers t on t.traveler_id = ttl.traveler_id
inner join stations s on s.station_id = ttl.entry_station_id
inner join stations s1 on s1.station_id = ttl.exit_station_id;
```



| tracking_id | traveler_name | Entry Station | Exit Station | Total Amount |
|---|---|---|---|---|
| 101 | Emily Johnson | South Station | Kenmore | 14 |
| 102 | Ethan Taylor | Kenmore | Charles: R | 12 |
| 103 | Olivia Davis | Copley | Park Street: RG | 6 |
| 104 | Michael Smith | Kenmore | Park Street: RG | 10 |
| 105 | Sophia Brown | Community College | Unique Square | 14 |
| 106 | Liam Miller | Lechmere | Aquarium | 10 |
| 107 | Ava Wilson | Copley | South Station | 10 |
| 109 | Isabella Thomas | West End | Chinatown | 10 |
| 110 | Mason White | Broadway | Park Street: RG | 0 |
| 111 | Emma Jackson | Tufts Medical Center | Maverick: B | 10 |
| 112 | Liam Harris | Arlington | Black Bay | 12 |
| 113 | Ava Martinez | Haymarket | North Station | 6 |
| 114 | Ethan Martinez | Haymarket | North Station | 6 |
| 115 | Olivia Smith | State: YB | Government C... | 0 |

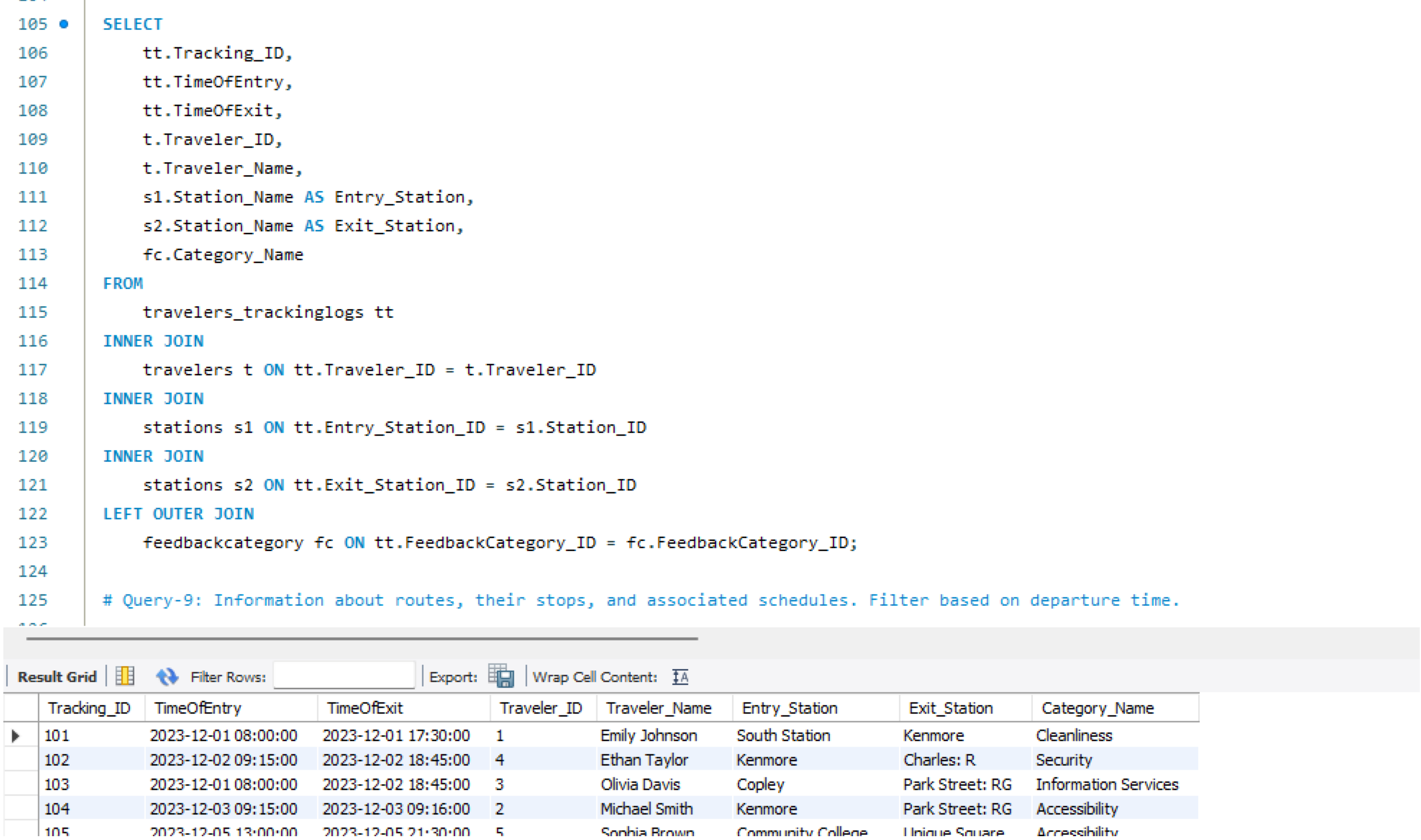
### **7. Traveler with maximum  time spent[sub query]**



```sql
SELECT Traveler_ID, MAX(Max_Time_Spent) AS Max_Time_Spent
FROM (
    SELECT Traveler_ID, SUM(TIMESTAMPDIFF(MINUTE, TimeOfEntry, TimeOfExit)) AS Max_Time_Spent
    FROM travelers_trackinglogs
    GROUP BY Traveler_ID
) AS TravelerTimeSpent
GROUP BY Traveler_ID;
```

| Traveler_ID | Max_Time_Spent |
|---|---|
| 1 | 570 |
| 2 | 1 |
| 3 | 2085 |
| 4 | 570 |
| 5 | 510 |
| 6 | 495 |
| 7 | 450 |
| 9 | -930 |
| 10 | -990 |
| 11 | -990 |
| 12 | -990 |
| 13 | 450 |
| 14 | 450 |
| 15 | 450 |

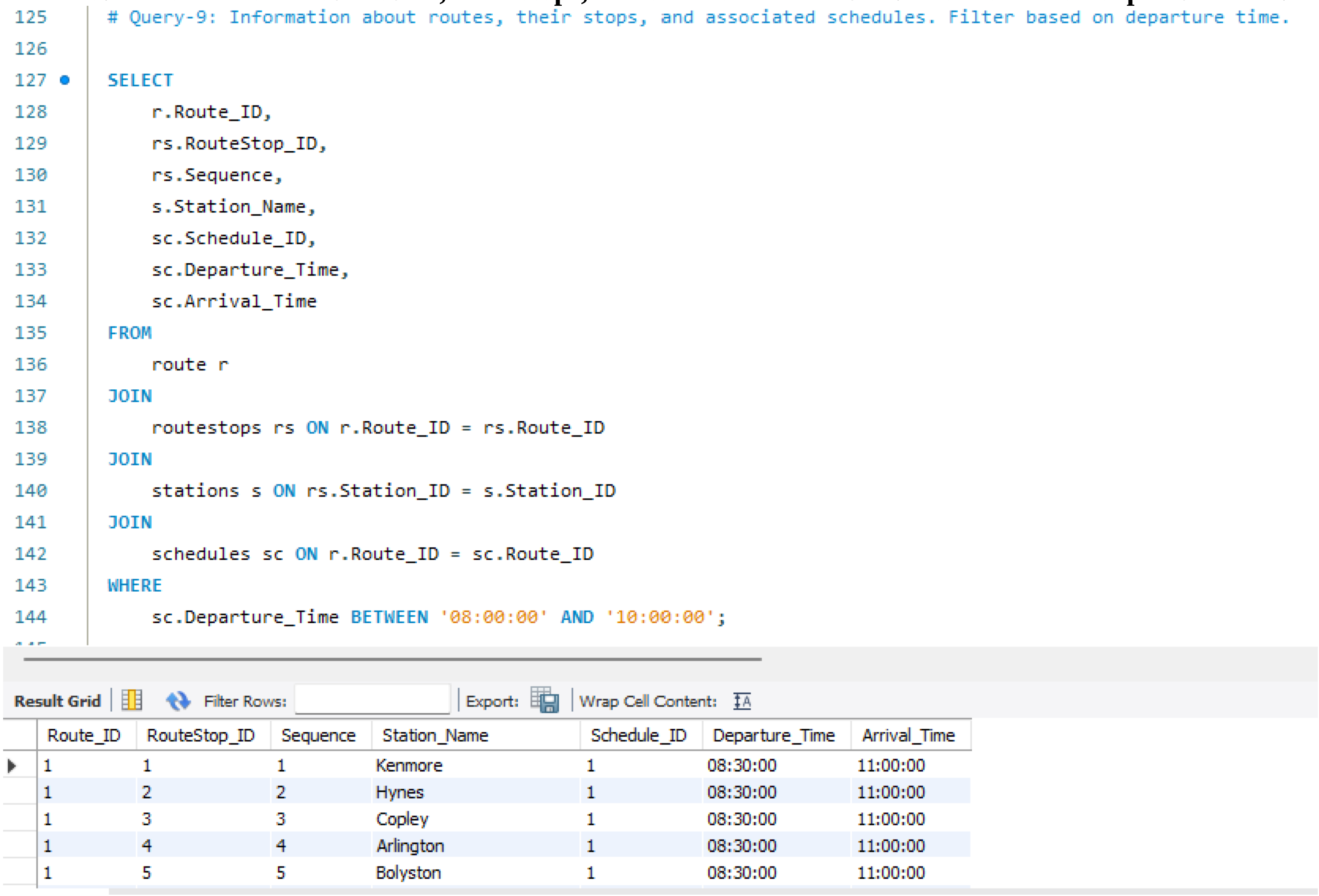
## 8. Information about travelers, their tracking logs, associated stations, and feedback categories



```sql
SELECT
    tt.Tracking_ID,
    tt.TimeOfEntry,
    tt.TimeOfExit,
    t.Traveler_ID,
    t.Traveler_Name,
    s1.Station_Name AS Entry_Station,
    s2.Station_Name AS Exit_Station,
    fc.Category_Name
FROM
    travelers_trackinglogs tt
INNER JOIN
    travelers t ON tt.Traveler_ID = t.Traveler_ID
INNER JOIN
    stations s1 ON tt.Entry_Station_ID = s1.Station_ID
INNER JOIN
    stations s2 ON tt.Exit_Station_ID = s2.Station_ID
LEFT OUTER JOIN
    feedbackcategory fc ON tt.FeedbackCategory_ID = fc.FeedbackCategory_ID;

# Query-9: Information about routes, their stops, and associated schedules. Filter based on departure time.
```

| Tracking_ID | TimeOfEntry | TimeOfExit | Traveler_ID | Traveler_Name | Entry_Station | Exit_Station | Category_Name |
|---|---|---|---|---|---|---|---|
| 101 | 2023-12-01 08:00:00 | 2023-12-01 17:30:00 | 1 | Emily Johnson | South Station | Kenmore | Cleanliness |
| 102 | 2023-12-02 09:15:00 | 2023-12-02 18:45:00 | 4 | Ethan Taylor | Kenmore | Charles: R | Security |
| 103 | 2023-12-01 08:00:00 | 2023-12-02 18:45:00 | 3 | Olivia Davis | Copley | Park Street: RG | Information Services |
| 104 | 2023-12-03 09:15:00 | 2023-12-03 09:16:00 | 2 | Michael Smith | Kenmore | Park Street: RG | Accessibility |
| 105 | 2023-12-05 13:00:00 | 2023-12-05 21:30:00 | 5 | Sophia Brown | Community College | Unique Square | Accessibility |

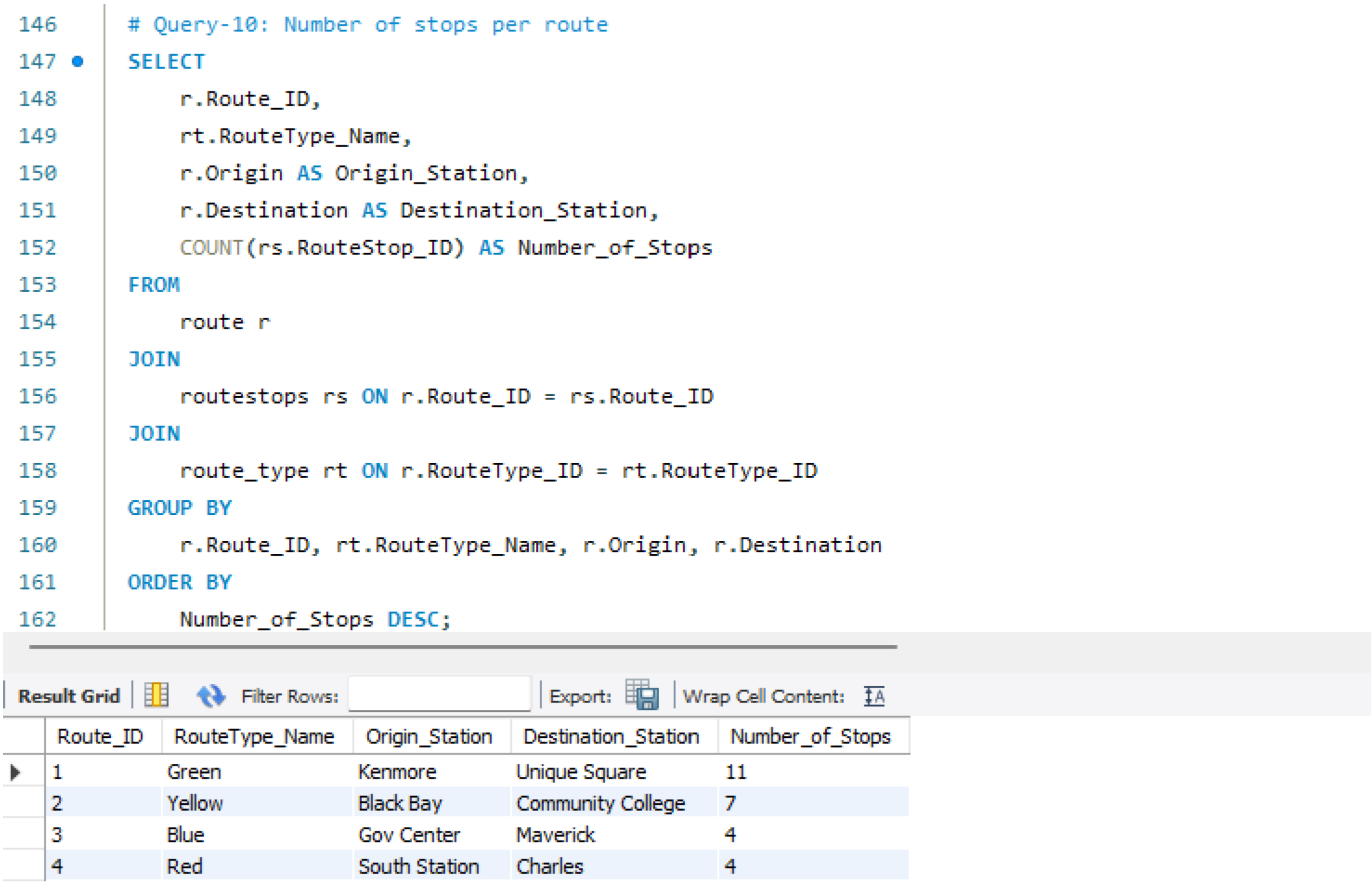
### 9. Information about routes, their stops, and associated schedules. Filter based on departure time.



```
# Query-9: Information about routes, their stops, and associated schedules. Filter based on departure time.

SELECT
    r.Route_ID,
    rs.RouteStop_ID,
    rs.Sequence,
    s.Station_Name,
    sc.Schedule_ID,
    sc.Departure_Time,
    sc.Arrival_Time
FROM
    route r
JOIN
    routestops rs ON r.Route_ID = rs.Route_ID
JOIN
    stations s ON rs.Station_ID = s.Station_ID
JOIN
    schedules sc ON r.Route_ID = sc.Route_ID
WHERE
    sc.Departure_Time BETWEEN '08:00:00' AND '10:00:00';
```

| Route_ID | RouteStop_ID | Sequence | Station_Name | Schedule_ID | Departure_Time | Arrival_Time |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | Kenmore | 1 | 08:30:00 | 11:00:00 |
| 1 | 2 | 2 | Hynes | 1 | 08:30:00 | 11:00:00 |
| 1 | 3 | 3 | Copley | 1 | 08:30:00 | 11:00:00 |
| 1 | 4 | 4 | Arlington | 1 | 08:30:00 | 11:00:00 |
| 1 | 5 | 5 | Bolyston | 1 | 08:30:00 | 11:00:00 |

### 10. Calculate the number of stops per route

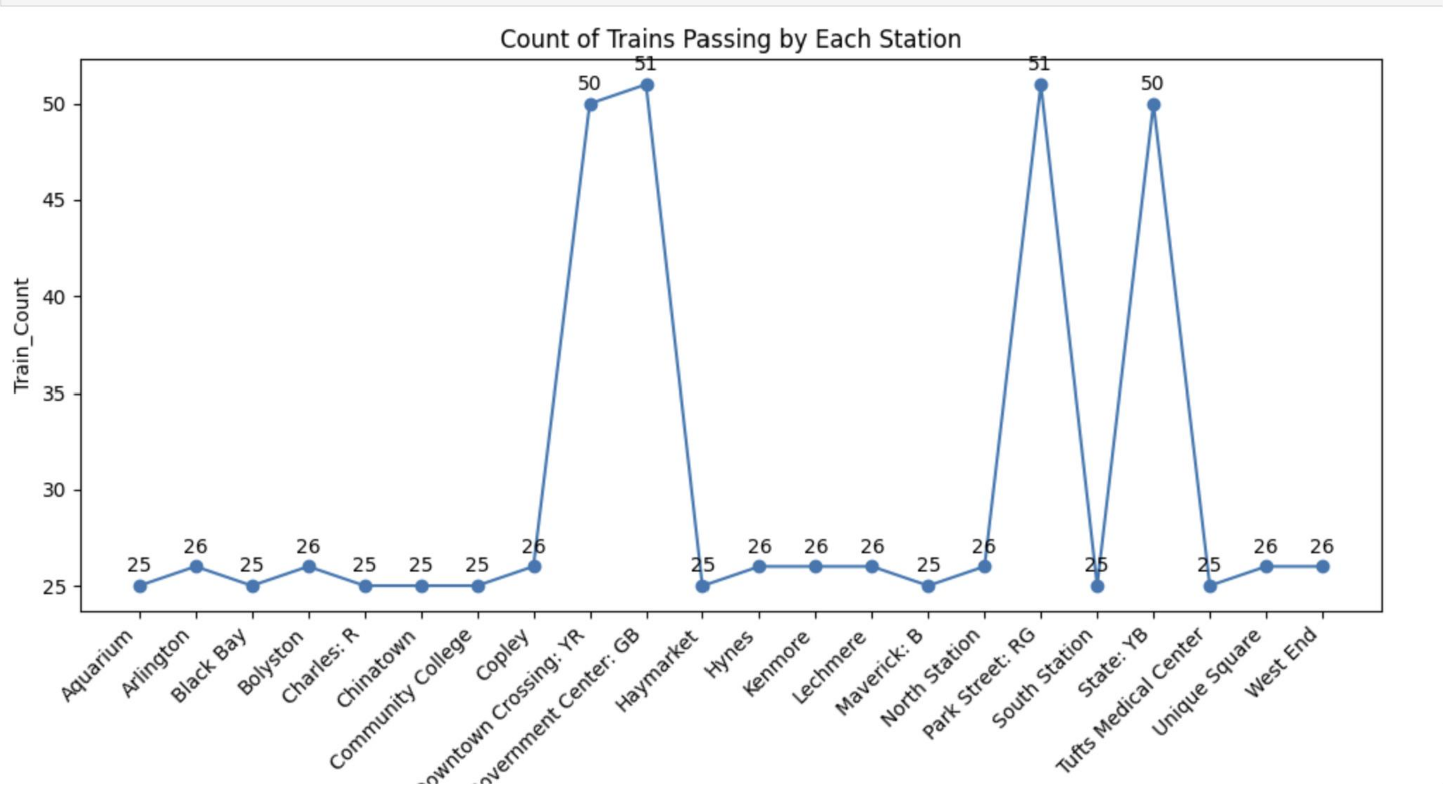


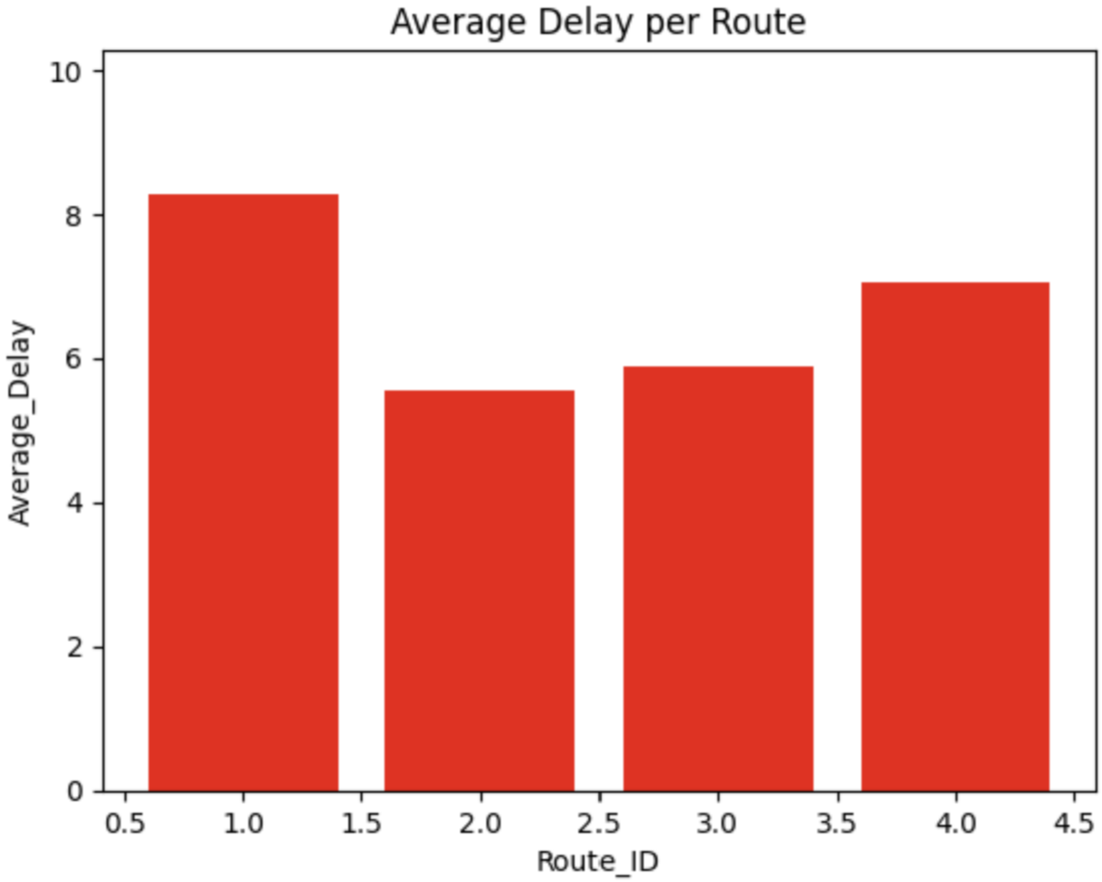
```sql
# Query-10: Number of stops per route
SELECT
    r.Route_ID,
    rt.RouteType_Name,
    r.Origin AS Origin_Station,
    r.Destination AS Destination_Station,
    COUNT(rs.RouteStop_ID) AS Number_of_Stops
FROM
    route r
JOIN
    routestops rs ON r.Route_ID = rs.Route_ID
JOIN
    route_type rt ON r.RouteType_ID = rt.RouteType_ID
GROUP BY
    r.Route_ID, rt.RouteType_Name, r.Origin, r.Destination
ORDER BY
    Number_of_Stops DESC;
```

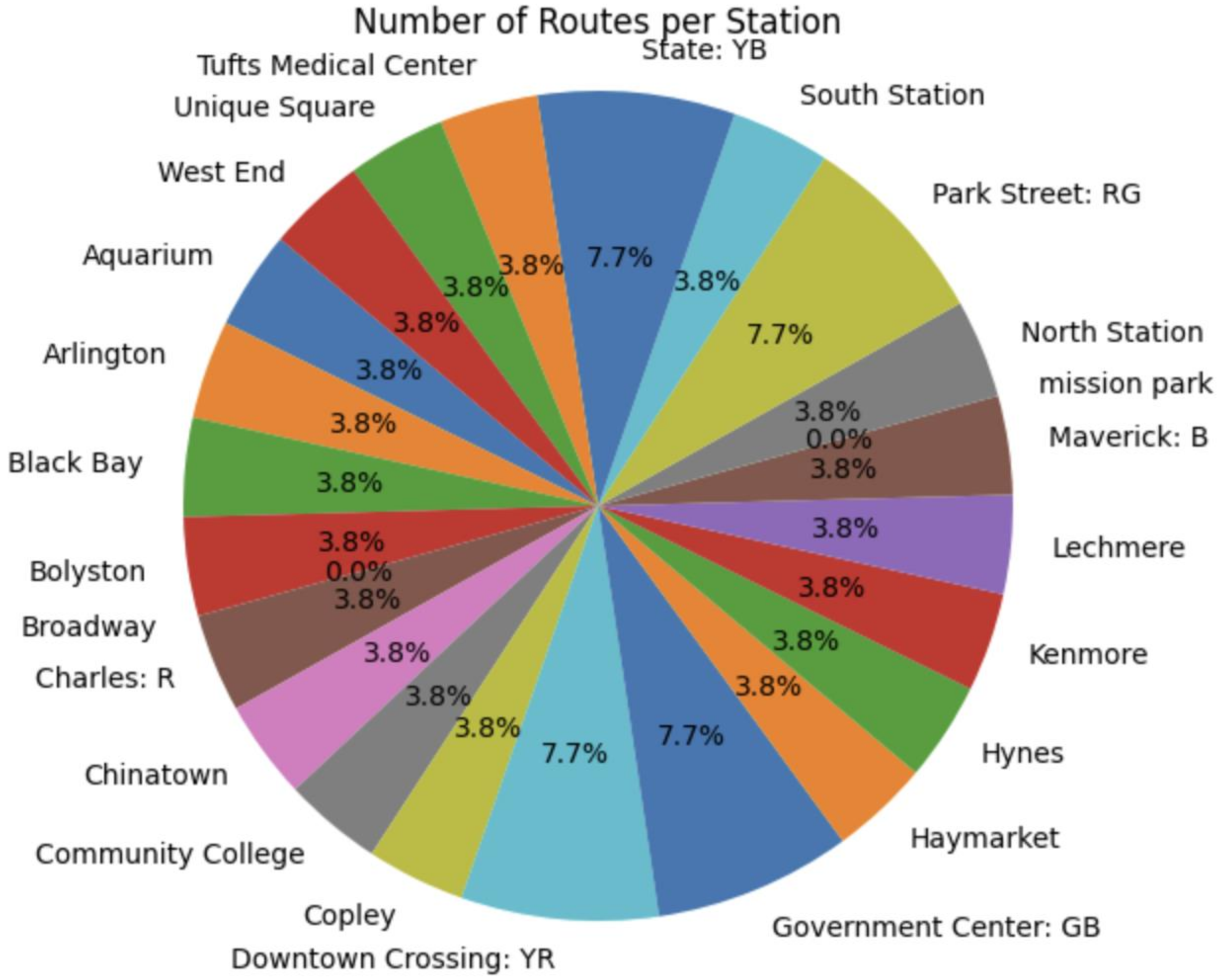
| Route_ID | RouteType_Name | Origin_Station | Destination_Station | Number_of_Stops |
|---|---|---|---|---|
| 1 | Green | Kenmore | Unique Square | 11 |
| 2 | Yellow | Black Bay | Community College | 7 |
| 3 | Blue | Gov Center | Maverick | 4 |
| 4 | Red | South Station | Charles | 4 |

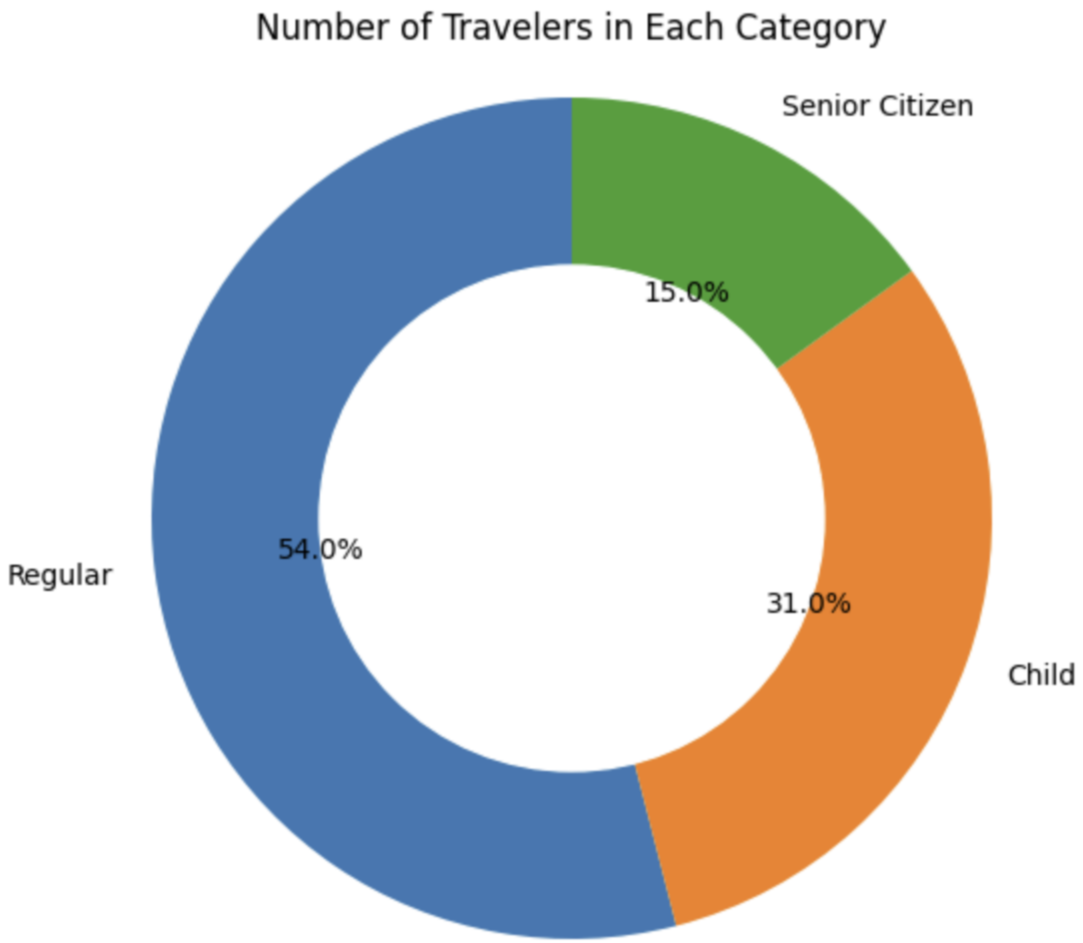
## Implementation Python:

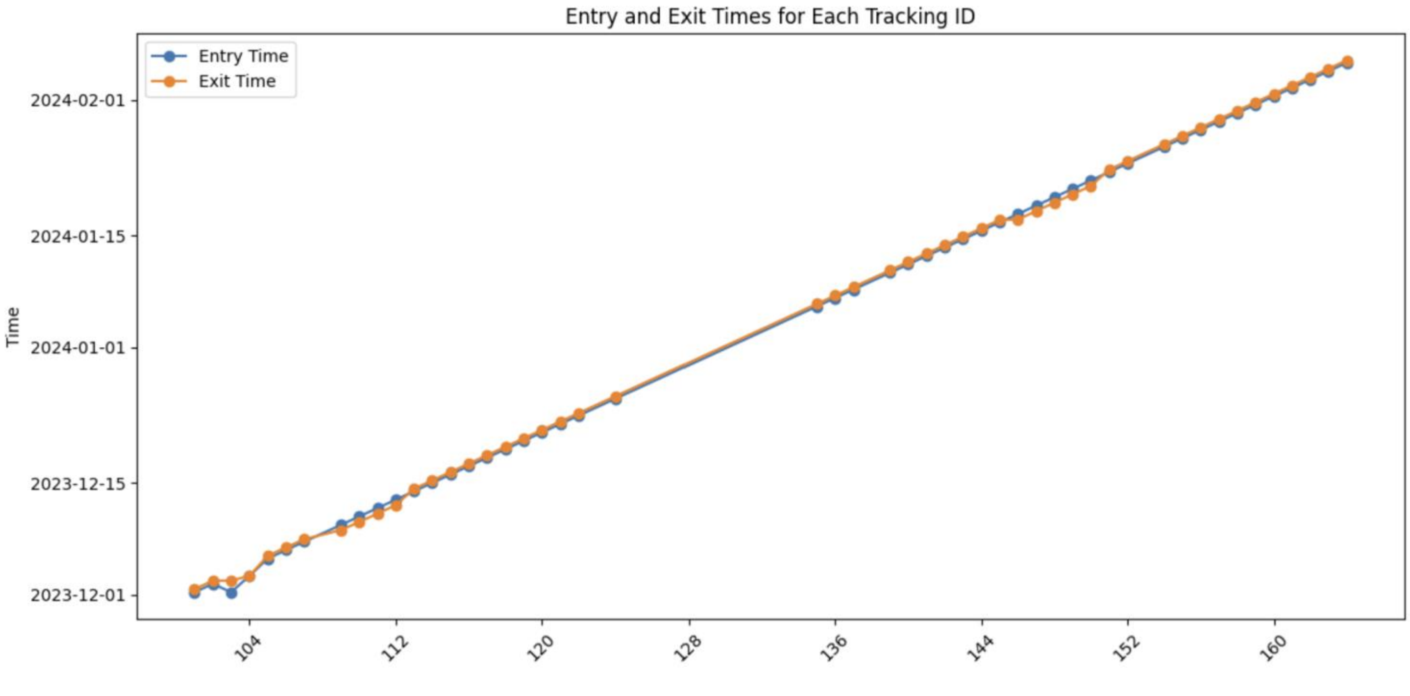
A comprehensive analysis of the MBTA database using MySQL queries and Python visualizations is presented. Leveraging mysql-connector for connectivity and matplotlib and seaborn for visualizations, the analysis covered insights such as product price distribution, order status distribution, and product category distribution by rating. The visualizations aid in making complex data accessible for informed decision-making in the MBTA platform.

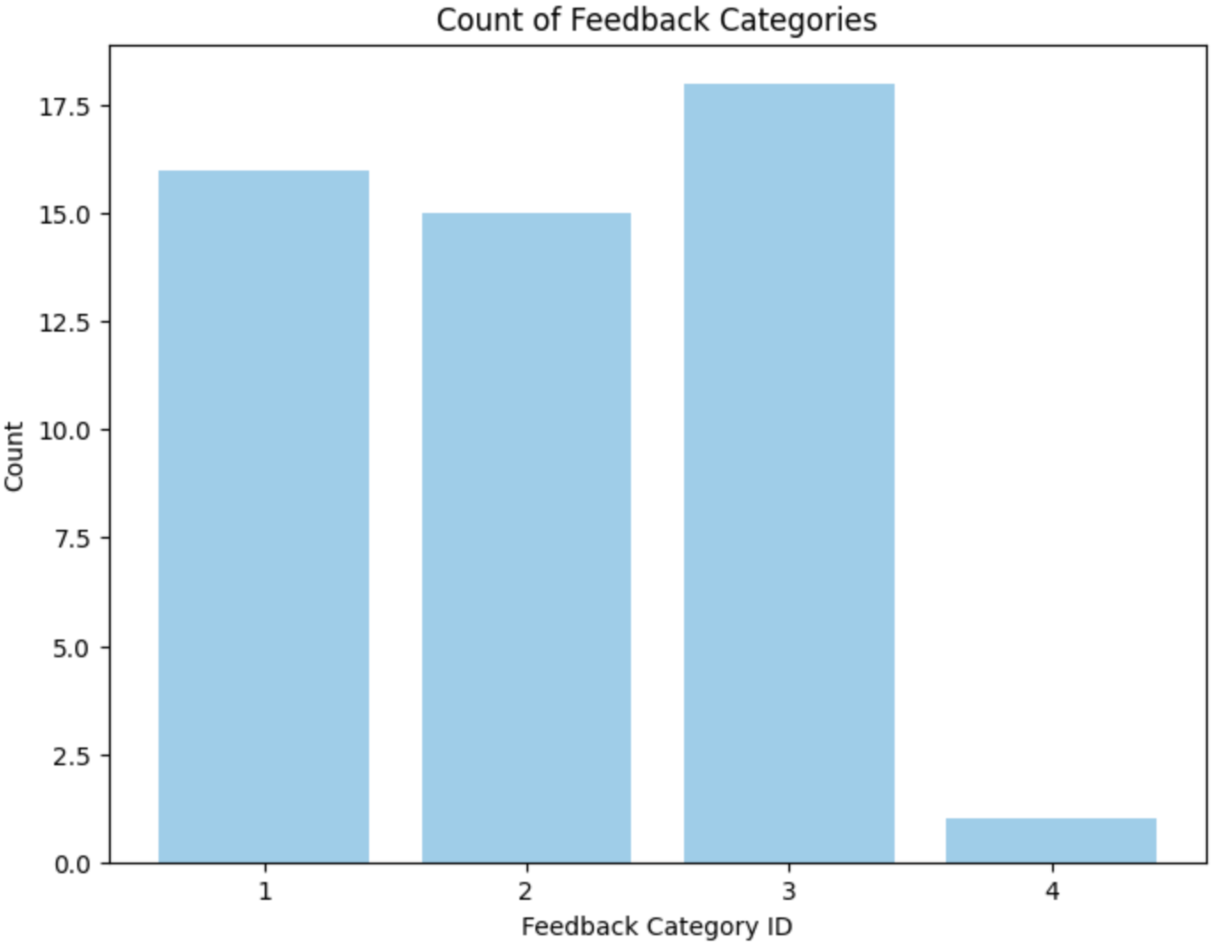



Average Delay per Route
Average_Delay
Route_ID
0
2
4
6
8
10
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5


Number of Routes per Station
State: YB
South Station
Park Street: RG
North Station
mission park
Maverick: B
Lechmere
Kenmore
Hynes
Haymarket
Government Center: GB
Downtown Crossing: YR
Copley
Community College
Chinatown
Charles: R
Broadway
Bolyston
Black Bay
Arlington
Aquarium
West End
Unique Square
Tufts Medical Center
7.7%
3.8%
7.7%
3.8%
0.0%
3.8%
3.8%
3.8%
3.8%
3.8%
7.7%
7.7%
3.8%
3.8%
3.8%
3.8%
0.0%
3.8%
3.8%
3.8%
3.8%
3.8%
3.8%
3.8%

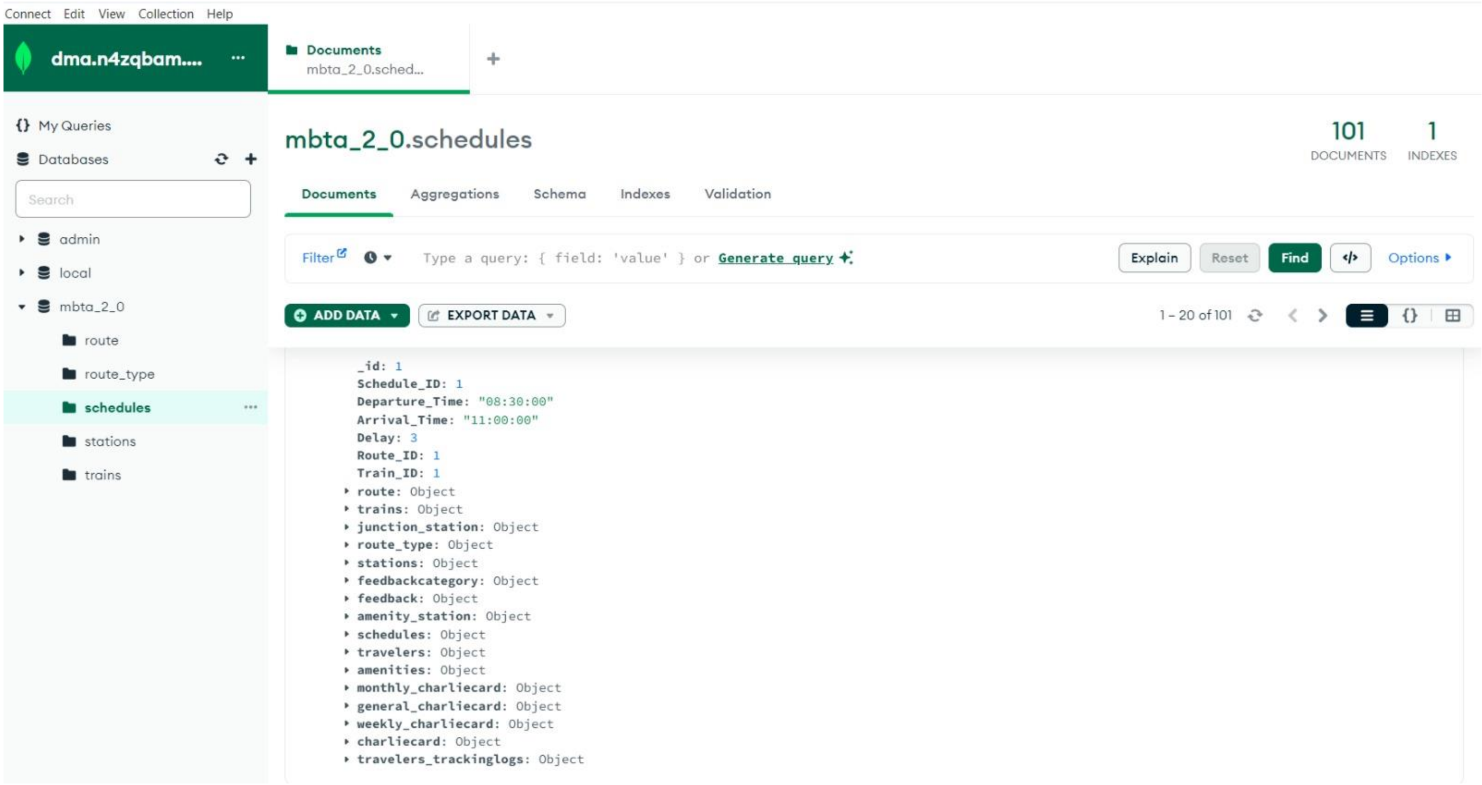

Number of Travelers in Each Category
Senior Citizen
15.0%
Regular
54.0%
31.0%
Child


Entry and Exit Times for Each Tracking ID
Entry Time
Exit Time
Time
2024-02-01
2024-01-15
2024-01-01
2023-12-15
2023-12-01
104
112
120
128
136
144
152
160


Count of Feedback Categories
Count
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1
2
3
4
Feedback Category ID

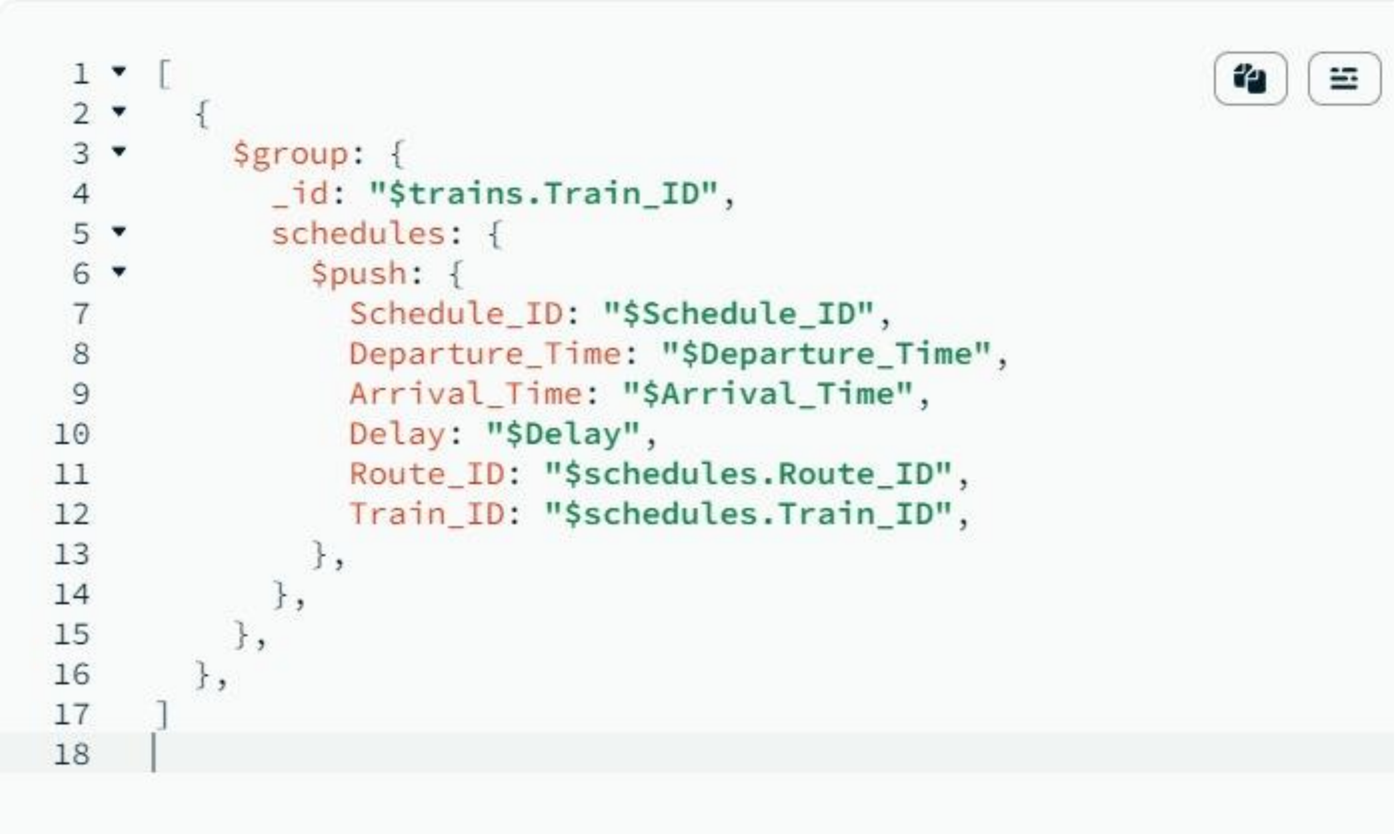
# Implementation in NoSQL

For the implementation in NoSQL we have created 4 collections as shown below

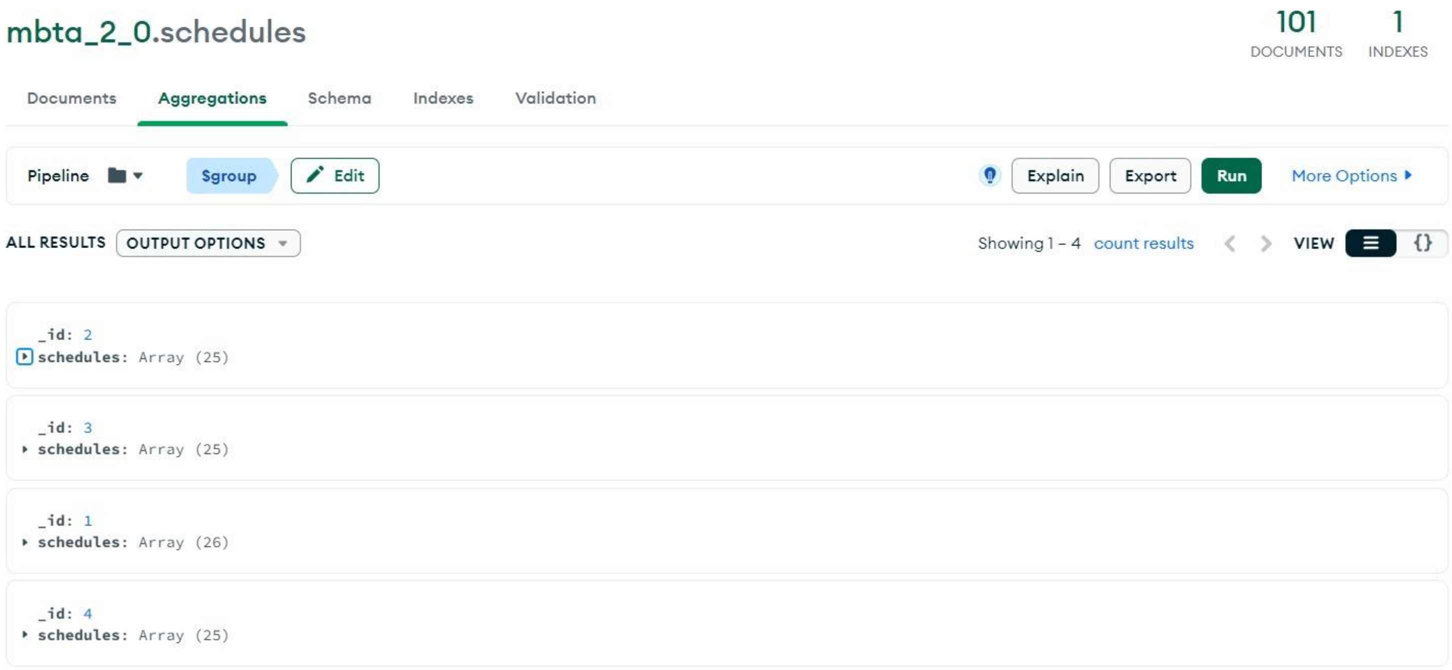


```
_id: 1
Schedule_ID: 1
Departure_Time: "08:30:00"
Arrival_Time: "11:00:00"
Delay: 3
Route_ID: 1
Train_ID: 1
▸ route: Object
▸ trains: Object
▸ junction_station: Object
▸ route_type: Object
▸ stations: Object
▸ feedbackcategory: Object
▸ feedback: Object
▸ amenity_station: Object
▸ schedules: Object
▸ travelers: Object
▸ amenities: Object
▸ monthly_charliecard: Object
▸ general_charliecard: Object
▸ weekly_charliecard: Object
▸ charliecard: Object
▸ travelers_trackinglogs: Object
```

### 1. For every train id give the schedule information

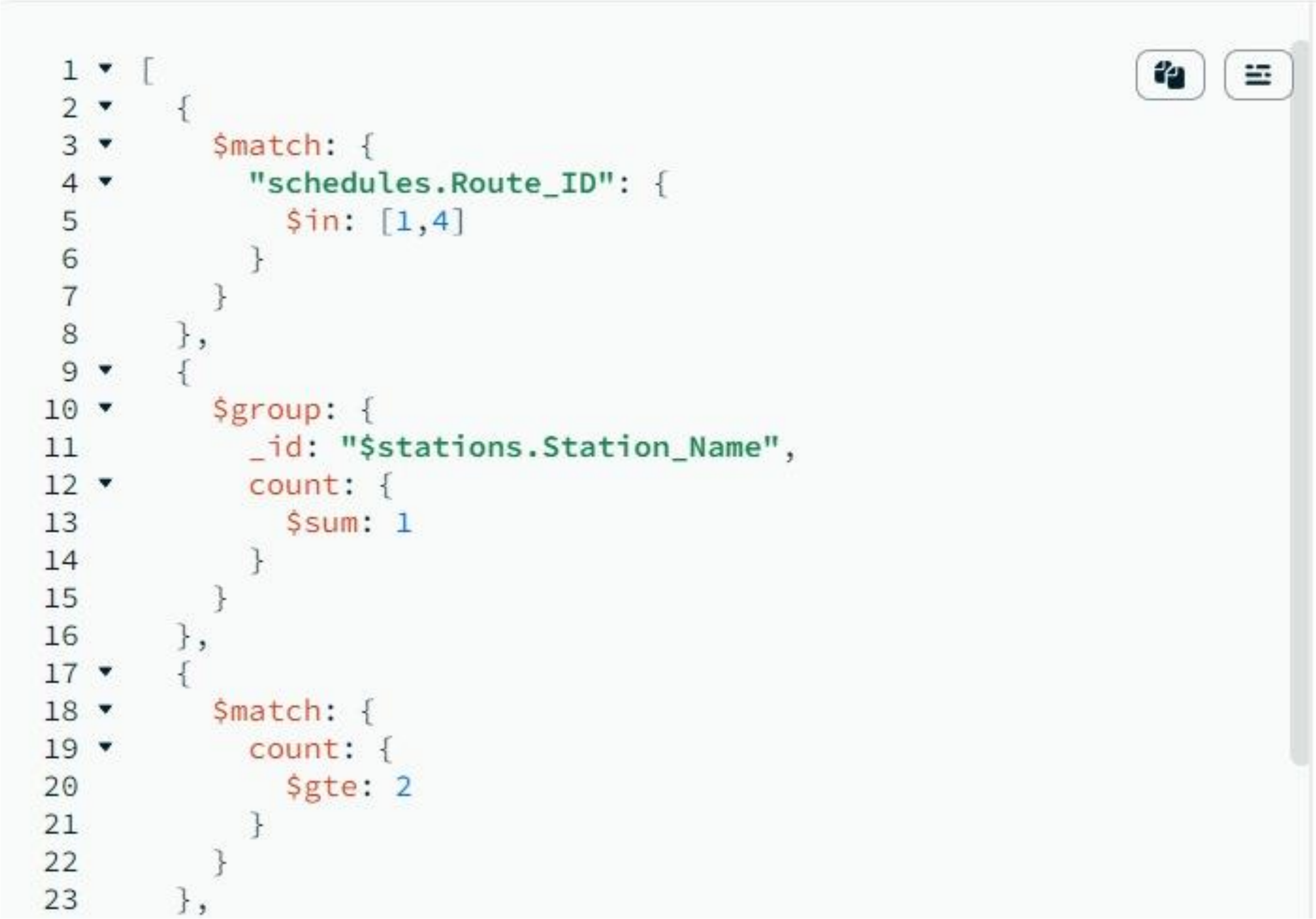


```
[
  {
    $group: {
      _id: "$trains.Train_ID",
      schedules: {
        $push: {
          Schedule_ID: "$Schedule_ID",
          Departure_Time: "$Departure_Time",
          Arrival_Time: "$Arrival_Time",
          Delay: "$Delay",
          Route_ID: "$schedules.Route_ID",
          Train_ID: "$schedules.Train_ID",
        },
      },
    },
  },
]
```

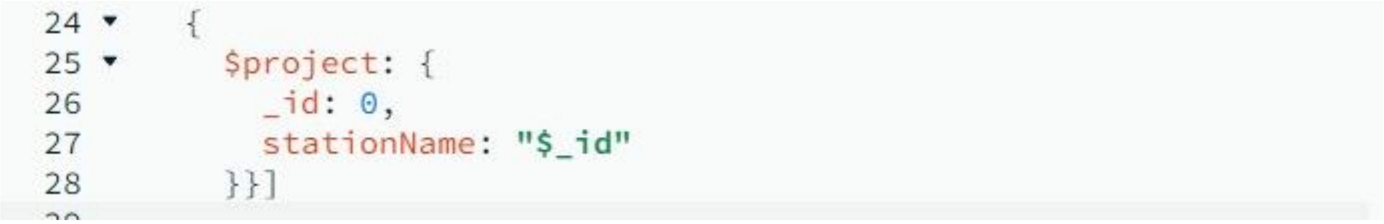


```
_id: 2
schedules: Array (25)

_id: 3
schedules: Array (25)

_id: 1
schedules: Array (26)

_id: 4
schedules: Array (25)
```

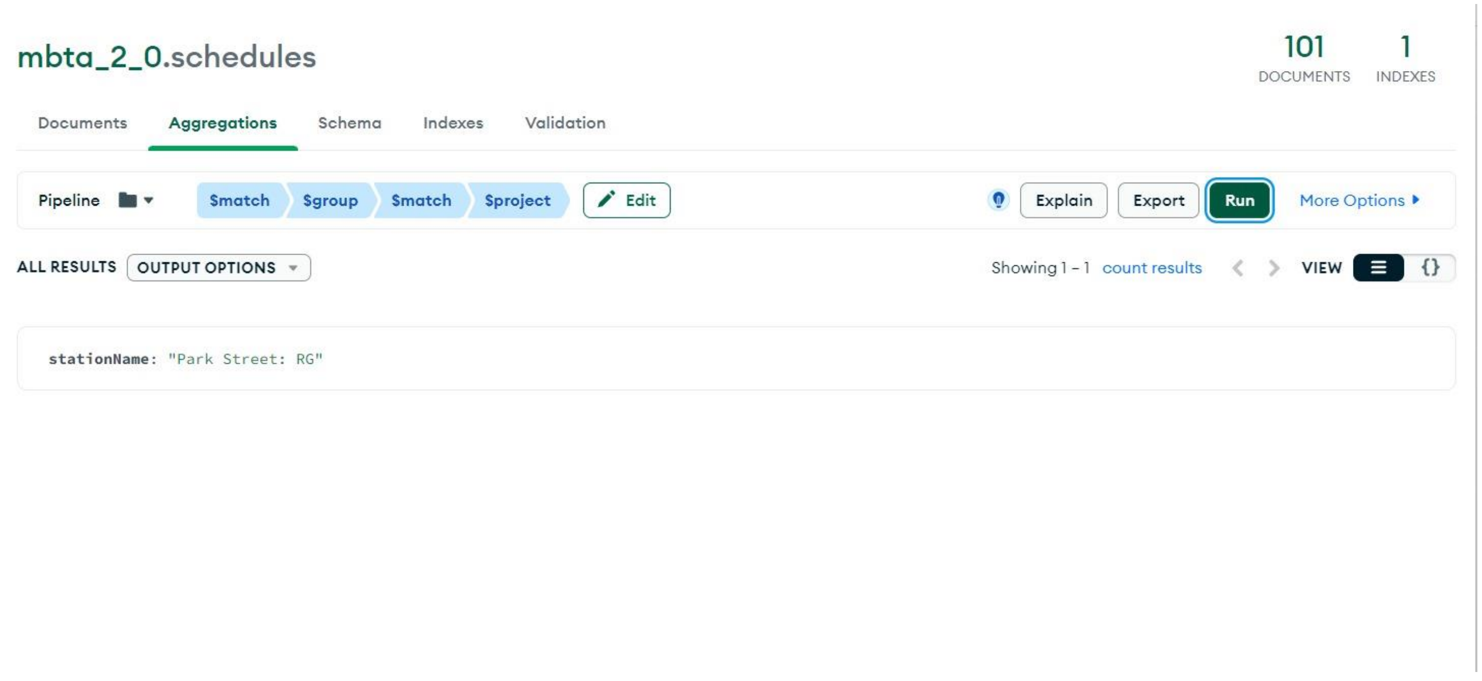
**2. Retrieve station names that are common in all Route**



```
[
  {
    $match: {
      "schedules.Route_ID": {
        $in: [1,4]
      }
    }
  },
  {
    $group: {
      _id: "$stations.Station_Name",
      count: {
        $sum: 1
      }
    }
  },
  {
    $match: {
      count: {
        $gte: 2
      }
    }
  },
```

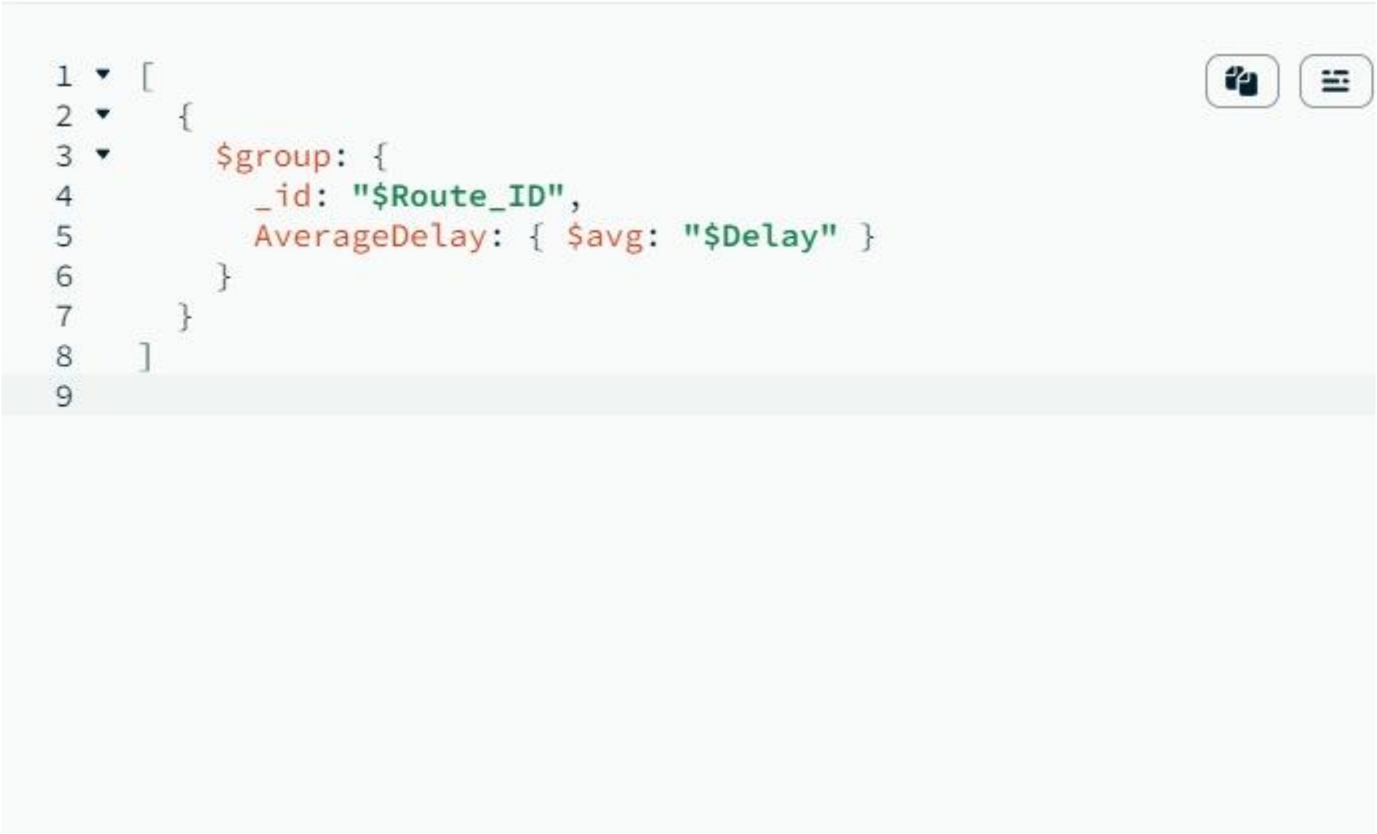


```
  {
    $project: {
      _id: 0,
      stationName: "$_id"
    }}]
```

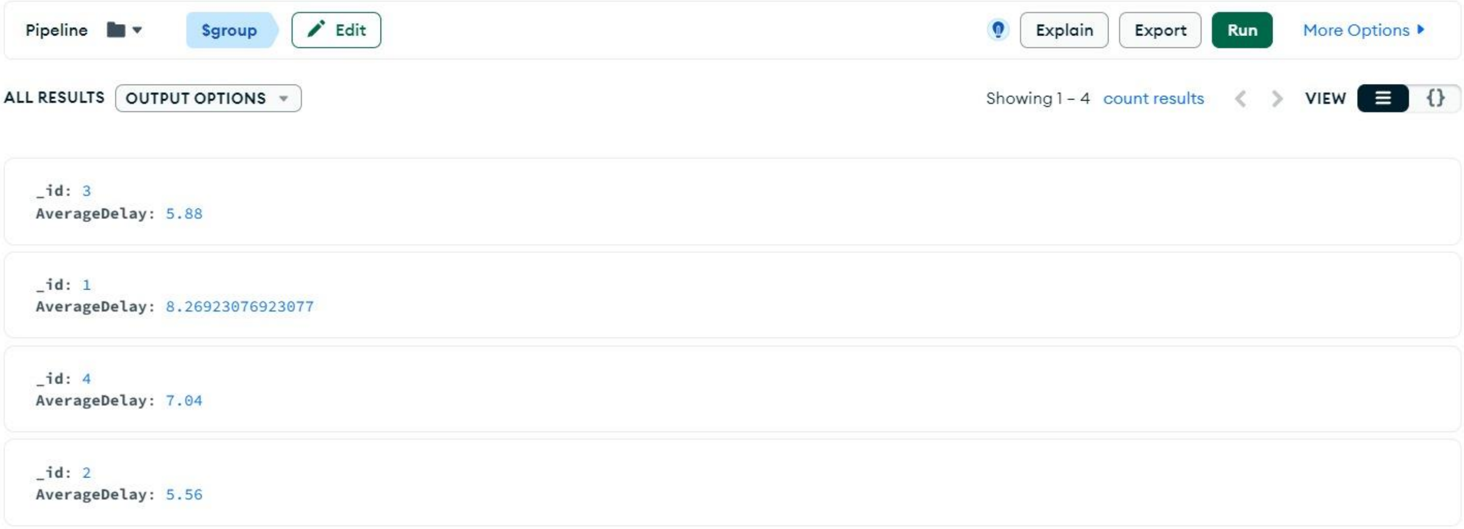


```
stationName: "Park Street: RG"
```

### 3. Find the average delay of route

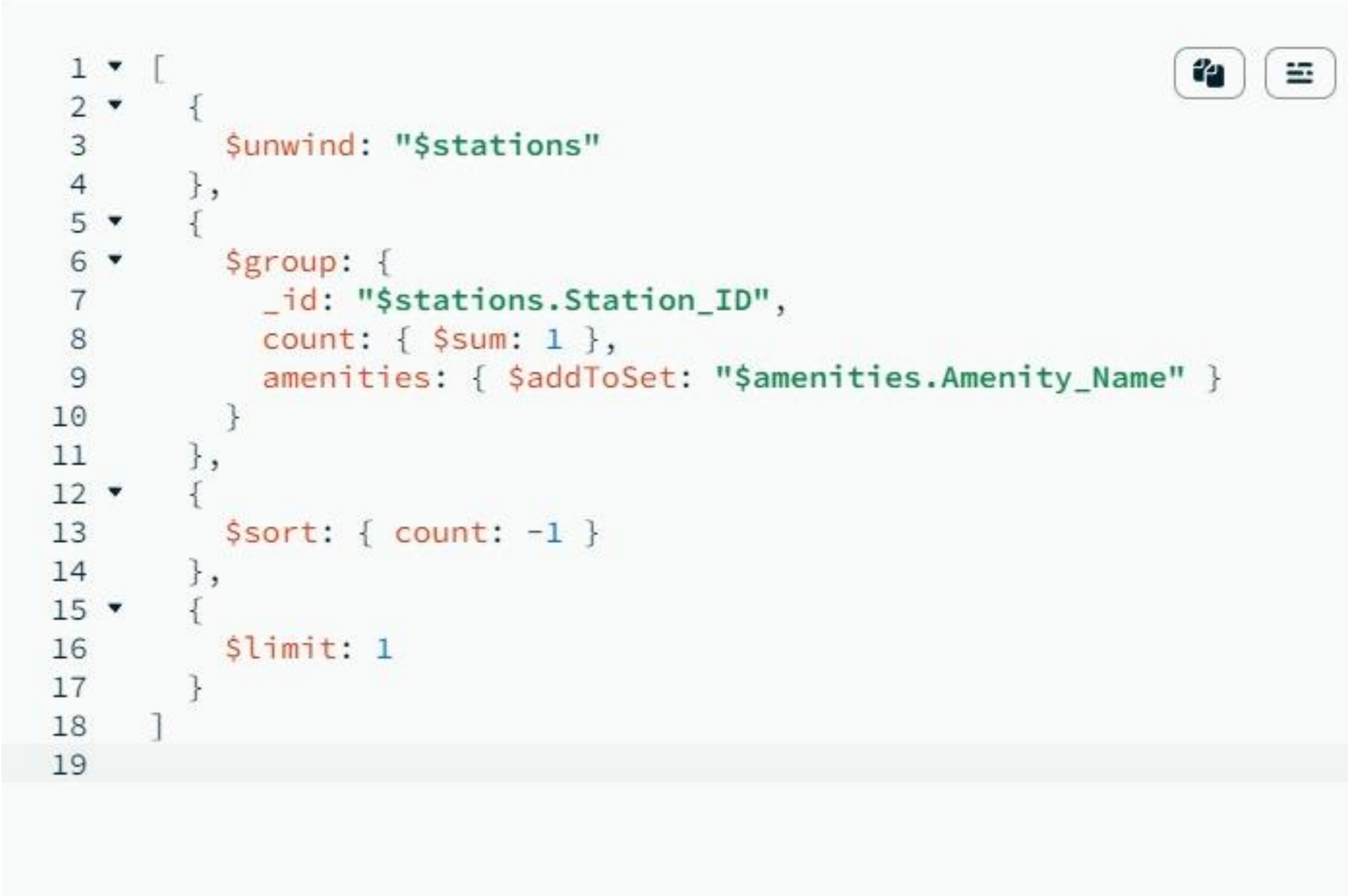


```
[
  {
    $group: {
      _id: "$Route_ID",
      AverageDelay: { $avg: "$Delay" }
    }
  }
]
```

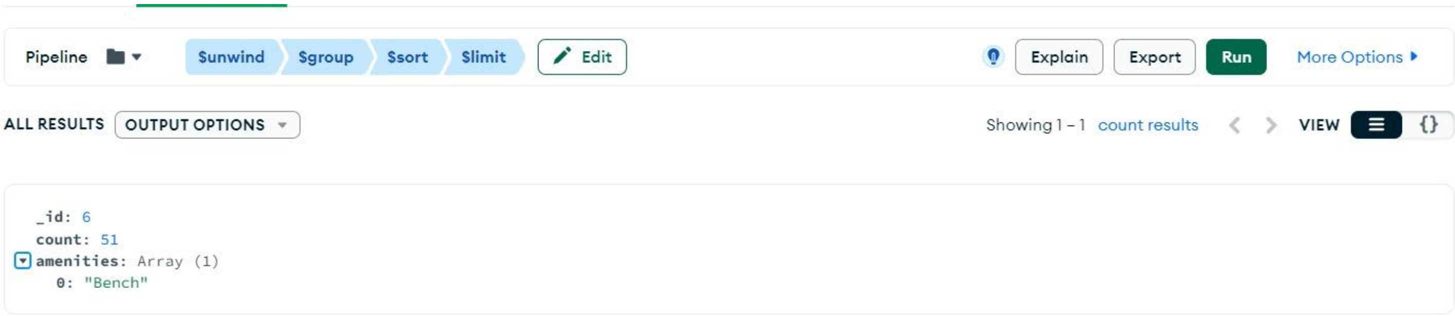


```
_id: 3
AverageDelay: 5.88

_id: 1
AverageDelay: 8.26923076923077

_id: 4
AverageDelay: 7.04

_id: 2
AverageDelay: 5.56
```

**4. find the count and names of amenities that are present in the maximum number of stations**

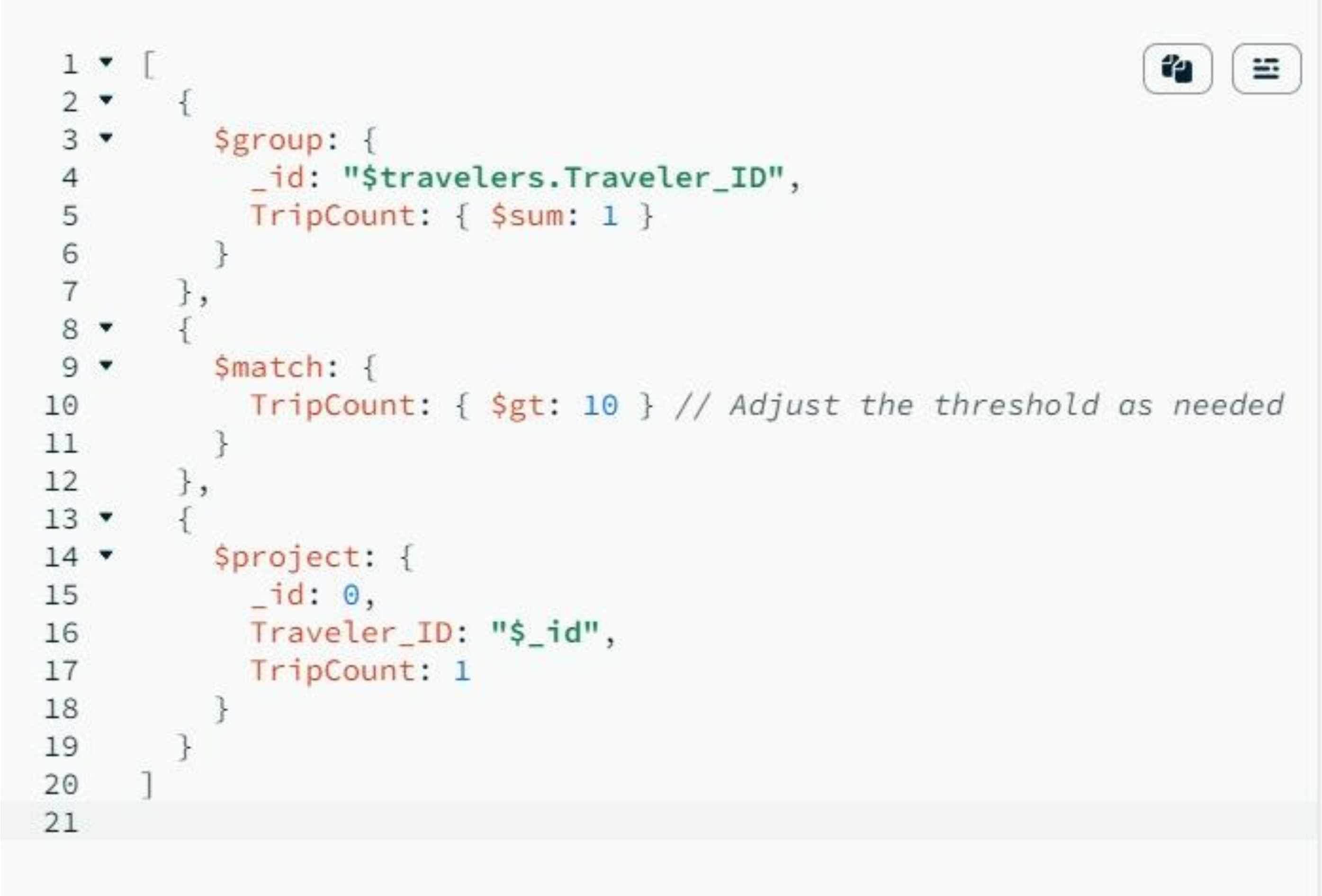


```
[
  {
    $unwind: "$stations"
  },
  {
    $group: {
      _id: "$stations.Station_ID",
      count: { $sum: 1 },
      amenities: { $addToSet: "$amenities.Amenity_Name" }
    }
  },
  {
    $sort: { count: -1 }
  },
  {
    $limit: 1
  }
]
```

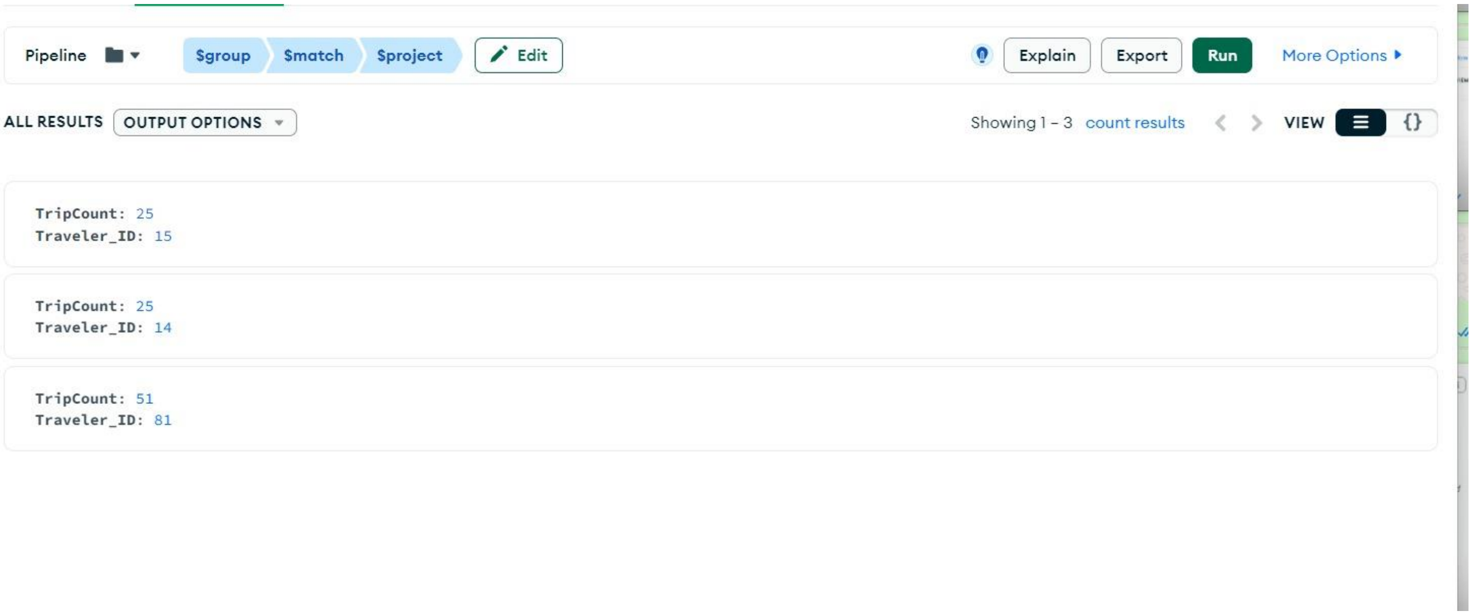


```
_id: 6
count: 51
amenities: Array (1)
  0: "Bench"
```

## 5. Find Travelers with High Trip Frequency



```
[
  {
    $group: {
      _id: "$travelers.Traveler_ID",
      TripCount: { $sum: 1 }
    }
  },
  {
    $match: {
      TripCount: { $gt: 10 } // Adjust the threshold as needed
    }
  },
  {
    $project: {
      _id: 0,
      Traveler_ID: "$_id",
      TripCount: 1
    }
  }
]
```



```
TripCount: 25
Traveler_ID: 15

TripCount: 25
Traveler_ID: 14

TripCount: 51
Traveler_ID: 81
```