Budget and Financials of MBTA

Milestone: Project Report

Group 21

Gouru Karthikeya Reddy
Giri Manohar Vemula

857-313-5329 (Tel of Student 1)
857-832-1320 (Tel of Student 2)

reddy.go@northeastern.edu
vemula.gi@northeastern.edu

Percentage of Effort Contributed by Student1: 50

Percentage of Effort Contributed by Student2: 50

Signature of Student 1: Gouru Karthikeya Reddy

Signature of Student 2: Giri Manohar Vemula

Submission Date: 04-23-2023

I. Introduction

Budget and financials relate to all the revenue-generating activities a corporation or organization engages in, including sales and marketing. The revenue generation of an organization helps determine the profitability of the services it offers by keeping track of the organization's revenues and expenses. A company wants to reach a given revenue goal and employs a variety of ways to achieve it. Most commuters in the Greater Boston area use the MBTA, which is a public transportation system. We want to concentrate on the three primary services offered, namely Commuter rail, Subway lines, and Bus routes.

The MBTA organization maintains a consolidated database containing all employee's backgrounds, electric expenses, maintenance bills, card production costs, and consumer data. By monitoring the information, they can manage the schedule and frequency of the trains and keep track of the quarterly repair expenditures. Based on the estimation of the revenue earned by the customer's usage of the common card to access services, we develop the database model to optimize the operation and functionality of the MBTA based on its quarterly consumption. This system will assist us in making better judgments on the effective use of cash.

The employee's ID, name, address, and the city must be stored by the organization for verification and security purposes. When the card is provided by a different organization, it is necessary for consumers to know their ID, the time, and the name of the institution that issued the card. In particular, the card's purchase price and validity duration should be noted. For each fare, record and broadcast the length of each live transit site. In addition, the customer journey and transaction procedure must be documented. Importantly, for customer adaptability, the business must record consumer purchasing behavior. Each time a follower purchases a product, a unique order number, product type, order time, quantity, and the price will be generated. There are two modes of transportation in our project, namely buses and trains, and three types of transportation, including buses, subways, and commuters. Furthermore, we are assessing the MBTA's quarterly maintenance fees which include refurbishment, sanitation, and advertising.

Requirements

1) An employee can work for only one department. A department may consist of N employees.

2) N number of salary statements can be generated for a single employee based on the quarter or hours worked.

3) One electricity bill generated must be specific to a particular station. A station could have N electricity bills. But no bill can be generated without pertaining to one particular station.

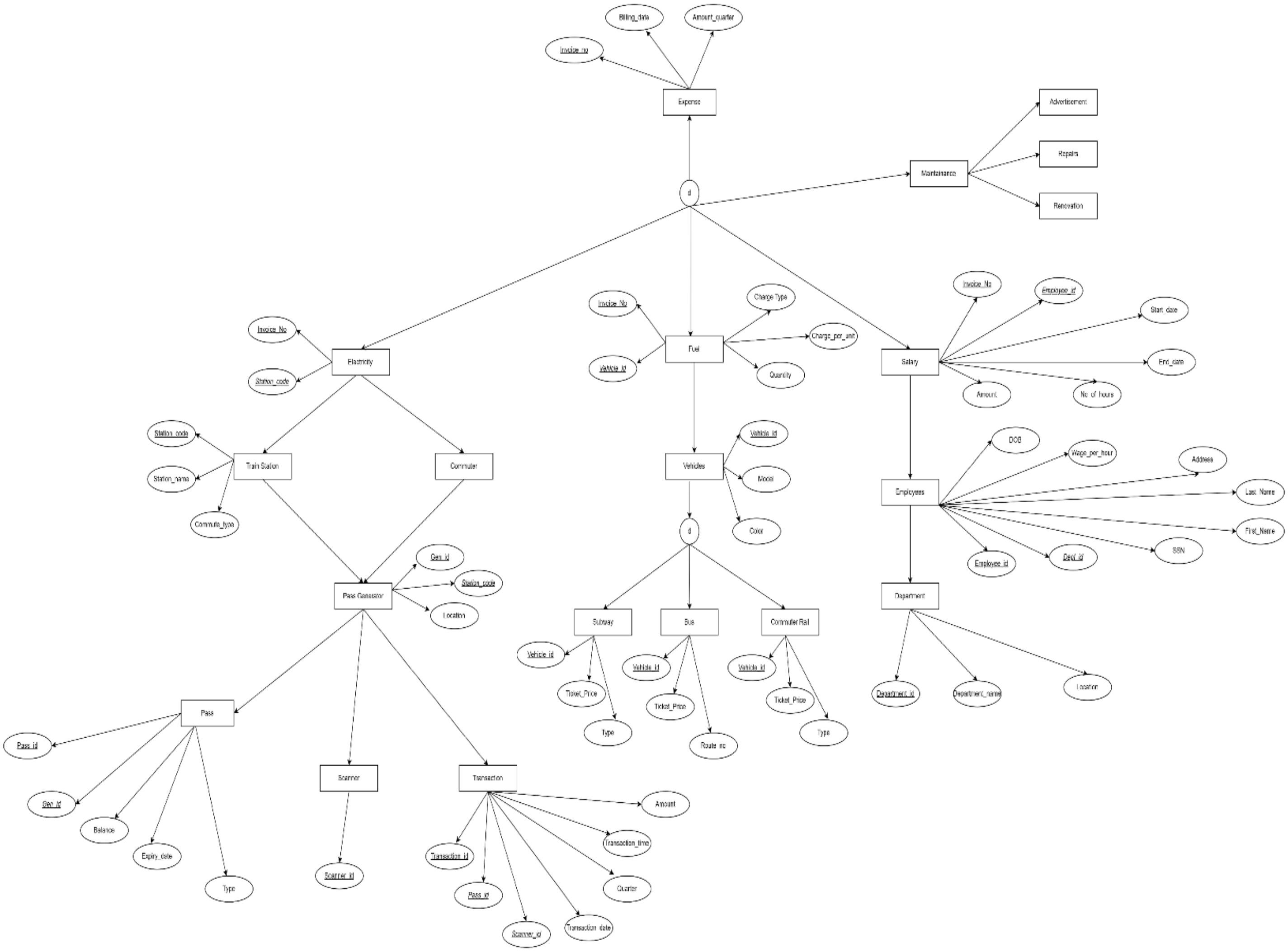
4) Pass generators are always present only on stations. There might be a few stations that do not have a pass generator. A pass generator may not generate any passes at all. Not all train stations have scanners.

5) When a transaction is made it must be made using one scanner. A transaction cannot be made without a scanner. A scanner on the other hand may not participate in any transaction.

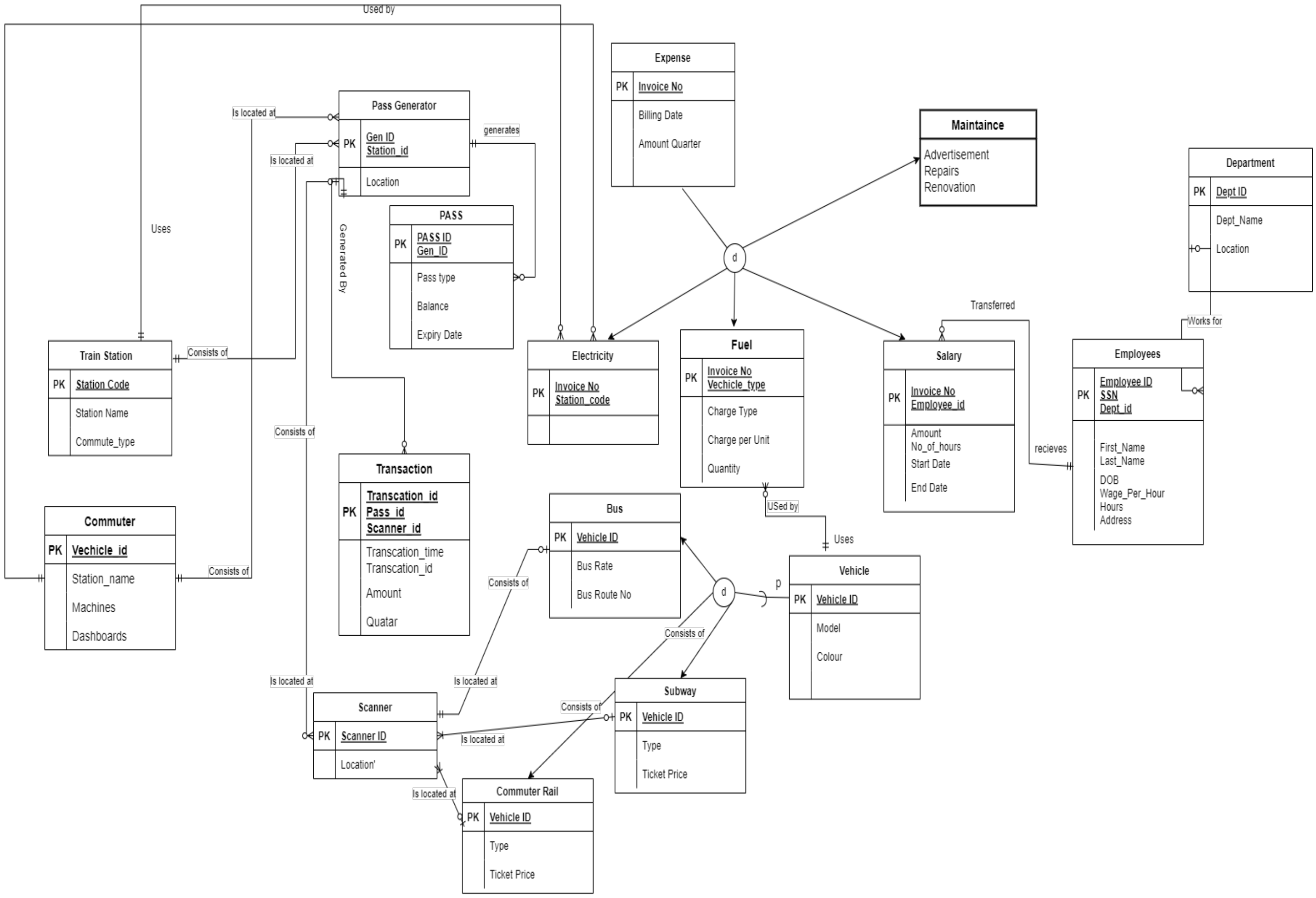
6) Fuel records cannot exist without linking them to a specific vehicle. A vehicle in turn may not have any fuel record present.

II. Conceptual Data Modeling

ER Model:



UML



III. Mapping Conceptual Model to Data Model

Expense (Invoice_no, Billing_date, Amount_quater)

Electricity (Station_code, Invoice_no)

Fuel (Invoice_no, Vehicle_type, Charge_type, charge_per_unit, quantity)

Salary (Invoice_No, Employee_id, Start_date, End_date, No_of_hours, Amount)

Maintenance (Invoice_no, Advertisement, Repairs, Renovation)

Train Station (Station_code, station_name, Commuter_type)

Commuter (Vehicle_ID, Station_name, Machines, Dashboards)

Pass Generator (Gen_id, Station_code, Location)

Pass (Pass_id, Gen_id, Balance, Expiry_date, Type)

Scanner(Scanner_id, *Gen_id* , Location)

Transaction (Transaction_id, Pass_id, Scanner_id, Transaction_date, Quarter, Transaction_time, Amount)

Vehicles (Vehicle_id, Model, Color, *Filling*)

Subway (Vehicle_id, Ticket_Price, Type)

Bus (Vehicle_id, Ticket_Price, Route_no)

Commuter Rail (Vehicle_id, Ticket_Price, Type)

Employees (Employee_id, Dept_id, SSN, DOB, Hours, Wage_per_hour, Address, Last_Name, First_Name, *WorkFor*)

Department (Department_id, Department_name, Location)

IV. Implementation of Relation Model via MySQL and NoSQL

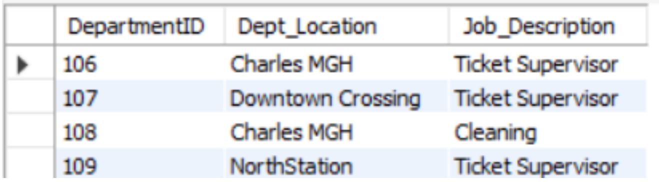
The database was created in MySQL and the following queries were performed.

MySQL Implementation:

Query 1: Find the details from department table where departmentid is greater than 105.

```
select * from department

where departmentid > 105;
```



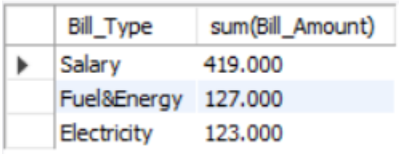
	DepartmentID	Dept_Location	Job_Description
▶	106	Charles MGH	Ticket Supervisor
	107	Downtown Crossing	Ticket Supervisor
	108	Charles MGH	Cleaning
	109	NorthStation	Ticket Supervisor

Query 2: Find bill type and sum of the bill amount from expense table.

```
SELECT Bill_Type, sum(Bill_Amount)

FROM expense

GROUP BY Bill_Type;
```



Bill_Type	sum(Bill_Amount)
Salary	419.000
Fuel&Energy	127.000
Electricity	123.000

Query 3:

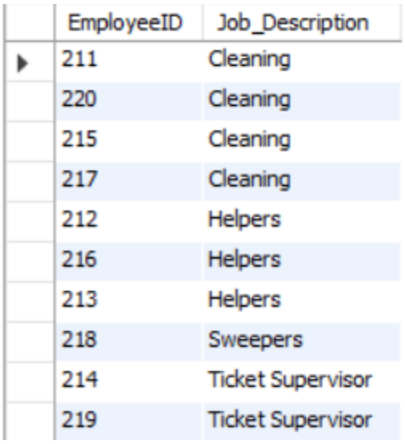
Find the employeeid and job description who started working 2 years ago from today.

SELECT e.employeeid, d.job_description

FROM employee e JOIN department d

ON e.departmentid = d.departmentid

WHERE startdate <= date_sub(curdate(), interval 2 year);



EmployeeID	Job_Description
211	Cleaning
220	Cleaning
215	Cleaning
217	Cleaning
212	Helpers
216	Helpers
213	Helpers
218	Sweepers
214	Ticket Supervisor
219	Ticket Supervisor

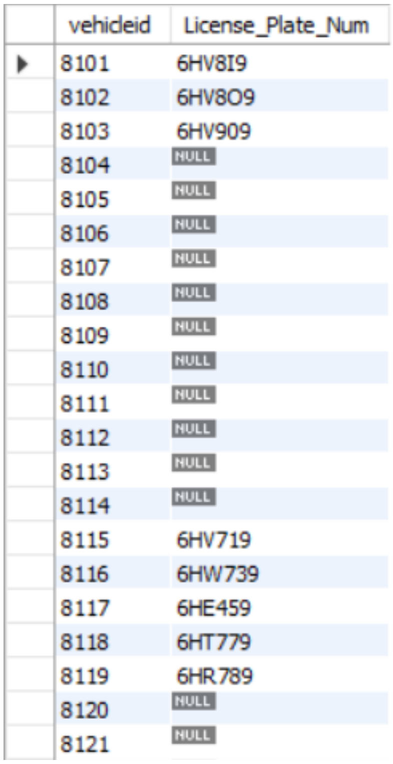
Query 4:

Find the details of vehicles and bus license by joining bus table with vehicles.

SELECT v.vehicleid, b.License_Plate_Num

FROM bus b RIGHT OUTER JOIN vehicle v

ON v.vehicleid=b.vehicleid;



vehicleid	License_Plate_Num
8101	6HV8I9
8102	6HV8O9
8103	6HV909
8104	NULL
8105	NULL
8106	NULL
8107	NULL
8108	NULL
8109	NULL
8110	NULL
8111	NULL
8112	NULL
8113	NULL
8114	NULL
8115	6HV719
8116	6HW739
8117	6HE459
8118	6HT779
8119	6HR789
8120	NULL
8121	NULL

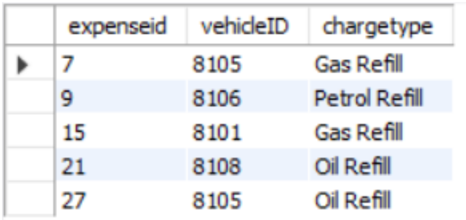
Query 5:

Find the expenseid, vehicleid, chargetype from fuelenergy table where unitsused is more than the avg unitsused.

```
SELECT expenseid, vehicleID, chargetype

FROM fuelenergy

WHERE unitsused > (SELECT AVG(unitsused) FROM fuelenergy);
```



expenseid	vehicleID	chargetype
7	8105	Gas Refill
9	8106	Petrol Refill
15	8101	Gas Refill
21	8108	Oil Refill
27	8105	Oil Refill

Query 6:

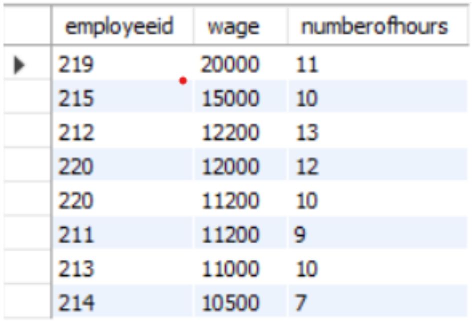
Write a corelated query to select details from salary where wage is greater than avg wage and order by wage in descending order.

SELECT employeeid, wage, numberofhours

FROM salary s

WHERE wage > (select avg(wage) FROM salary WHERE employeeid=s.EmployeeID)

ORDER BY wage desc;



employeeid	wage	numberofhours
219	20000	11
215	15000	10
212	12200	13
220	12000	12
220	11200	10
211	11200	9
213	11000	10
214	10500	7

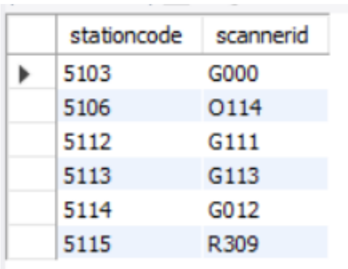
Query 7:

Find the stationcode and scannerid from stationscanner where it exists in transactions.

SELECT stationcode, scannerid

FROM stationscanner

WHERE EXISTS (SELECT scannerid FROM transactions);



stationcode	scannerid
5103	G000
5106	O114
5112	G111
5113	G113
5114	G012
5115	R309

}

NoSQL Implementation:

We have created 6 collections in MongoDB and performed the below queries.

Query 1:

Find the details of employees who are in department 101.

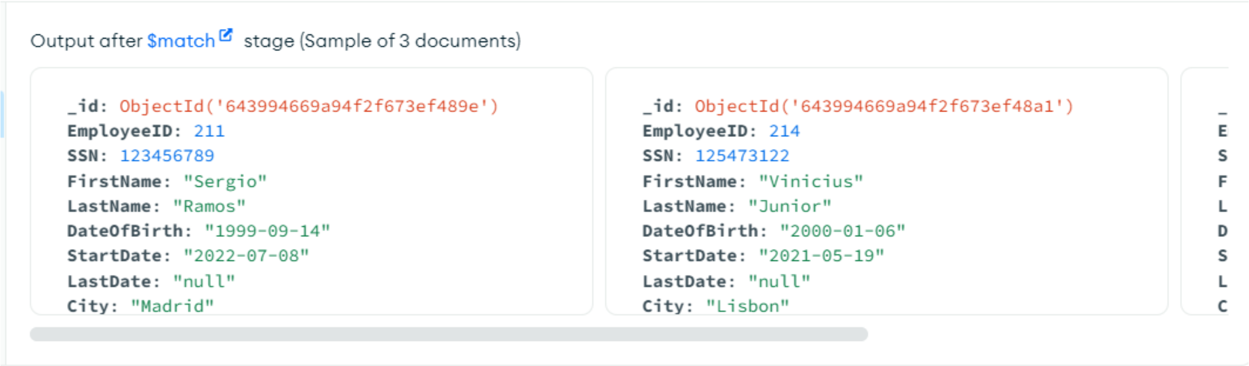
```
Mbta.employee.aggregate([

$match : {

 DepartmentID: 101,

}

])
```



Output after $match stage (Sample of 3 documents)

```
_id: ObjectId('643994669a94f2f673ef489e')
EmployeeID: 211
SSN: 123456789
FirstName: "Sergio"
LastName: "Ramos"
DateOfBirth: "1999-09-14"
StartDate: "2022-07-08"
LastDate: "null"
City: "Madrid"
```

```
_id: ObjectId('643994669a94f2f673ef48a1')
EmployeeID: 214
SSN: 125473122
FirstName: "Vinicius"
LastName: "Junior"
DateOfBirth: "2000-01-06"
StartDate: "2021-05-19"
LastDate: "null"
City: "Lisbon"
```

```
_
E
S
F
L
D
S
L
C
```

Query 2:

Find the count of employees with id greater than 215 and order them in descending order of date of birth.

```
Mbta.employees.aggregate([

$match : {

 EmployeeID: {

  $gte: 215,

 }

}
```

$Sort :

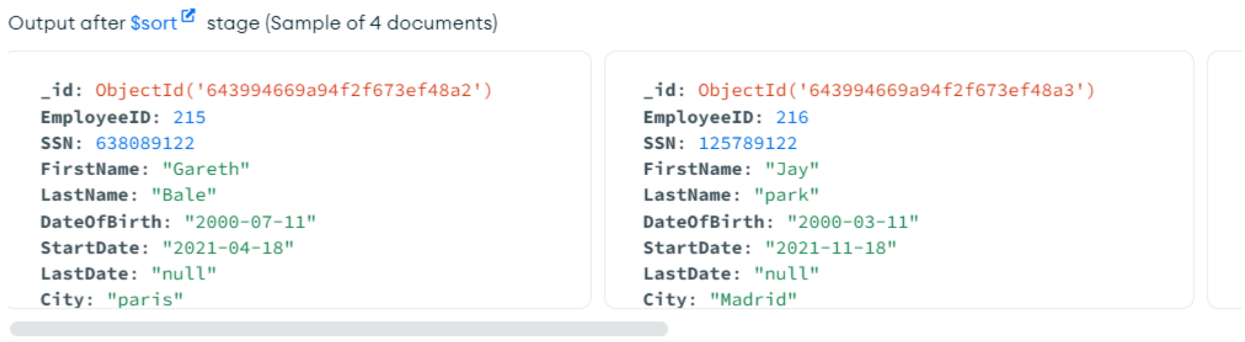
{

DateOfBirth: -1,

}

$Count : "Number of employees :"

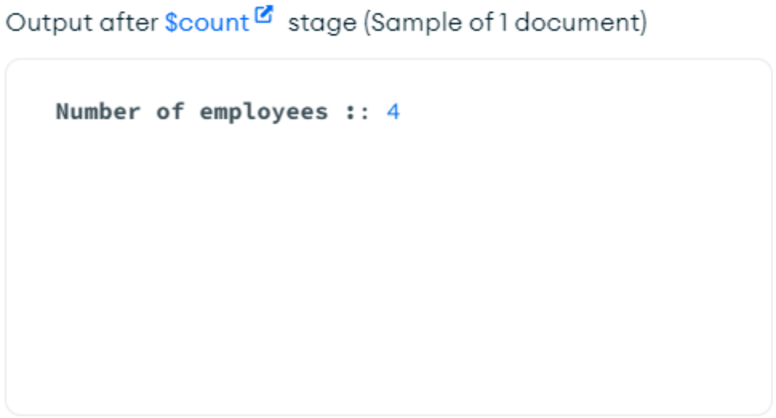
])



Output after $sort stage (Sample of 4 documents)

```
_id: ObjectId('643994669a94f2f673ef48a2')
EmployeeID: 215
SSN: 638089122
FirstName: "Gareth"
LastName: "Bale"
DateOfBirth: "2000-07-11"
StartDate: "2021-04-18"
LastDate: "null"
City: "paris"
```

```
_id: ObjectId('643994669a94f2f673ef48a3')
EmployeeID: 216
SSN: 125789122
FirstName: "Jay"
LastName: "park"
DateOfBirth: "2000-03-11"
StartDate: "2021-11-18"
LastDate: "null"
City: "Madrid"
```



Output after $count stage (Sample of 1 document)

```
Number of employees :: 4
```

Query 3:

Join Busses collection with vehicle collection and display only the vehicleid, license number, and the details of busses and limit the count to 3.

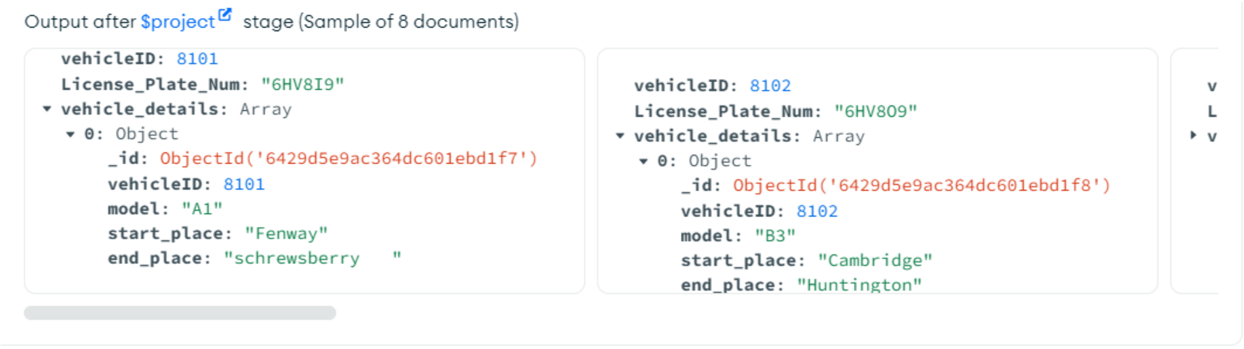
Mbta.bus.aggregate([

$lookup : {

from: "vehicle",

localField: "vehicleID",

```
 foreignField: "vehicleID",
 as: "vehicle_details"
}
$project : {
vehicleID: 1,
 License_Plate_Num: 1,
 _id: 0,
 vehicle_details: 1,
}
$limit : 3
])
```

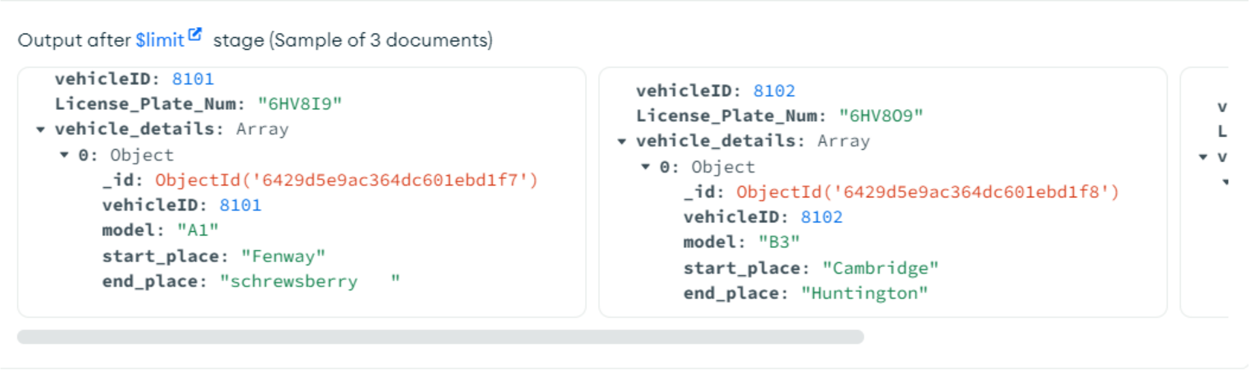


Output after $project stage (Sample of 8 documents)

```
  vehicleID: 8101
  License_Plate_Num: "6HV8I9"
▾ vehicle_details: Array
  ▾ 0: Object
      _id: ObjectId('6429d5e9ac364dc601ebd1f7')
      vehicleID: 8101
      model: "A1"
      start_place: "Fenway"
      end_place: "schrewsberry     "
```

```
  vehicleID: 8102
  License_Plate_Num: "6HV8O9"
▾ vehicle_details: Array
  ▾ 0: Object
      _id: ObjectId('6429d5e9ac364dc601ebd1f8')
      vehicleID: 8102
      model: "B3"
      start_place: "Cambridge"
      end_place: "Huntington"
```

```
  v
  L
▸ v
```



Output after $limit stage (Sample of 3 documents)

```
  vehicleID: 8101
  License_Plate_Num: "6HV8I9"
▾ vehicle_details: Array
  ▾ 0: Object
      _id: ObjectId('6429d5e9ac364dc601ebd1f7')
      vehicleID: 8101
      model: "A1"
      start_place: "Fenway"
      end_place: "schrewsberry     "
```

```
  vehicleID: 8102
  License_Plate_Num: "6HV8O9"
▾ vehicle_details: Array
  ▾ 0: Object
      _id: ObjectId('6429d5e9ac364dc601ebd1f8')
      vehicleID: 8102
      model: "B3"
      start_place: "Cambridge"
      end_place: "Huntington"
```

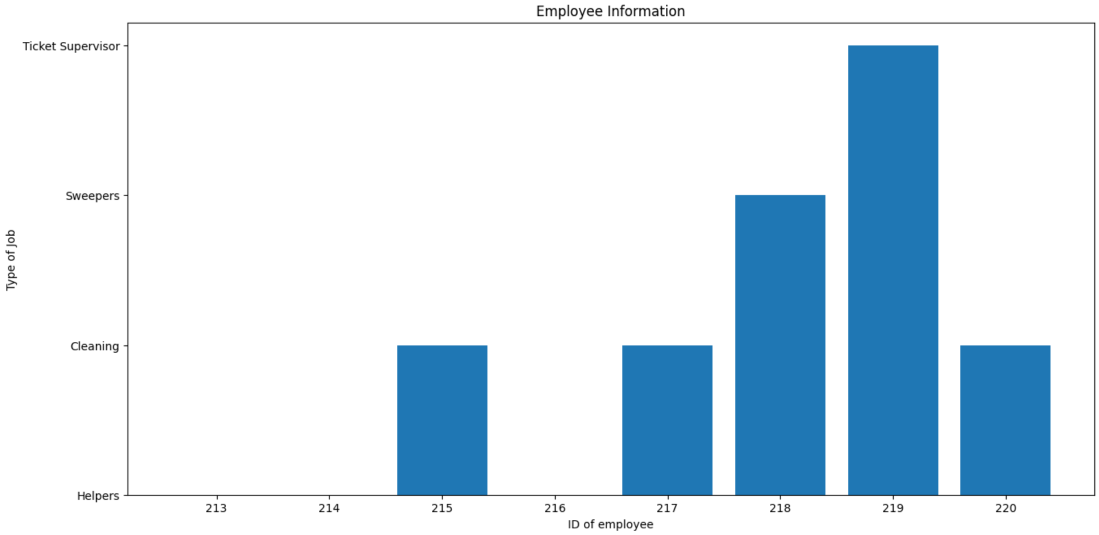
```
  v
  L
▾ v
```

V. Database access via Python

The database is accessed using Python and visualization of analyzed data is shown below. The connection of MySQL to Python is done using mysql.connector, followed by cursor.excecute to run and fetchall from query.

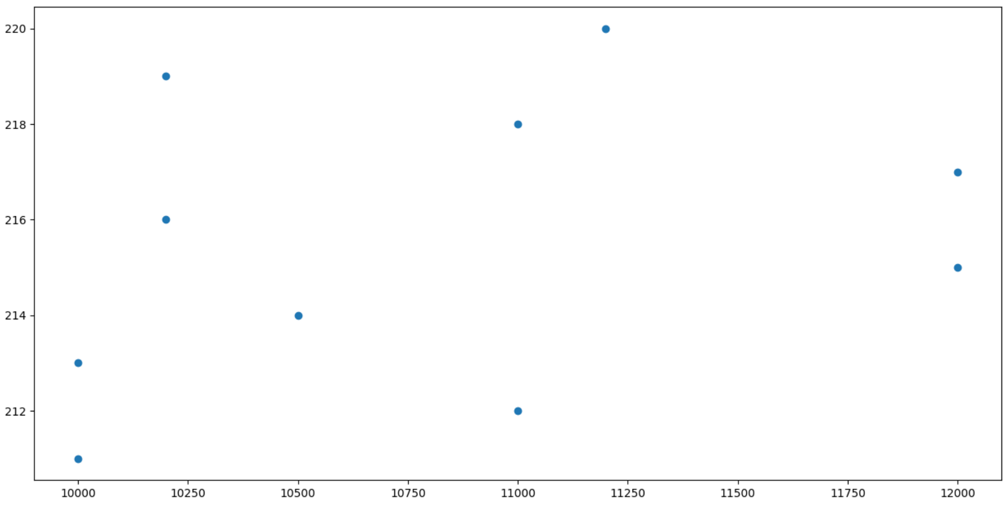
Graph -1:

To plot the employeeid and their job who have started working 2 years back from today.



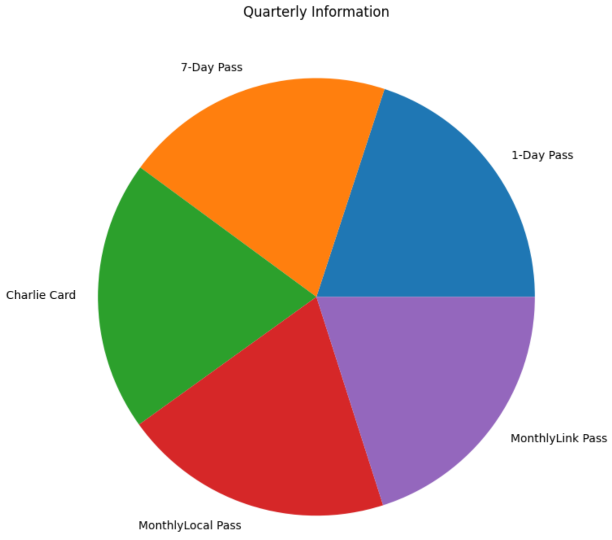
Graph 2:

Salaries of Employees



Graph 3:

Types of passes based on quarter.



VI. Summary and Conclusion

In conclusion, the MBTA receives revenue in different ways like passes, tickets for the subway, bus and commuter rail mode of transportation. They need to pay salaries to the employees and have maintenance costs like fuel and electricity, which happens to be most of their financial giving. Overall MBTA receives more revenue from subways due to the accessibility of subways which is more compared to commuter rails and busses.