<u>USE CASE STUDY REPORT</u>

Group 17
Hrithik Sarda and Valli Meenaa Vellaiyan

I. INTRODUCTION

Majority of the logistic companies deal with their accounts and customers manually follow a paper trial. It becomes tedious to maintain records of past shipments, all the bills generated, vehicle tracking, and customer information. With all the data spread out in different sheets of paper it is difficult to segregate and analyze the information, which leads to no growth or minimal growth of the company over the years. Moreover, important pieces of information on paper can easily be lost when needed the most, for example - during a scenario of court cases or customer billing. Idea for creating and managing a database for logistic companies was inspired by these reasons. This database is designed for a logistic company called “Boston Convenience” - which is a B2B transport company that offers four routes which travel via 21 pick-up and drop-off locations. It stores the details of services offered, requests made, trips taken, vehicles owned, and customer and employee information. We use the data to provide insights for making profitable business decisions. In addition to that, it is used for creating a ranking system to manage the employees, and for business expansion. Most importantly, it can be used for inbound marketing, that is, attracting new customers and maintaining long-term customer relationships. Real-time data was obtained from DataWorld, where all the information about trucks, warehouses, employee salaries, and other transactional attributes is provided by several logistic companies. Employees and customer information, and all the necessary tables are populated using MySQL Workbench. MySQL and MongoDB are used for executing analytical queries. Python is used to create bar plots and pie charts for displaying analytics. It is also used to dynamically create the list of locations for a particular route, given the pick-up and drop-off locations.

II. CONCEPTUAL DATA MODELING

An EER diagram has been designed to capture all the information required to create the database for Boston Convenience Transport.

<u>Description:</u>

The customer company is uniquely identified by its name, address, and phone number. The goods owned by the company are handled by their goods manager, who will act as our point of contact throughout the transportation. We store each customer company manager’s name, his/her unique ID, and his/her phone number. The goods are characterized by their weight and dimension; they are identified by an invoice, which indicates the customer’s ownership of the goods.

The customer makes a request for a service by giving information about the “pickup location” and “drop location”, as well as a “deadline for the delivery”. The “date” on which the request is made also gets logged. Each request is identified by an “ID”.

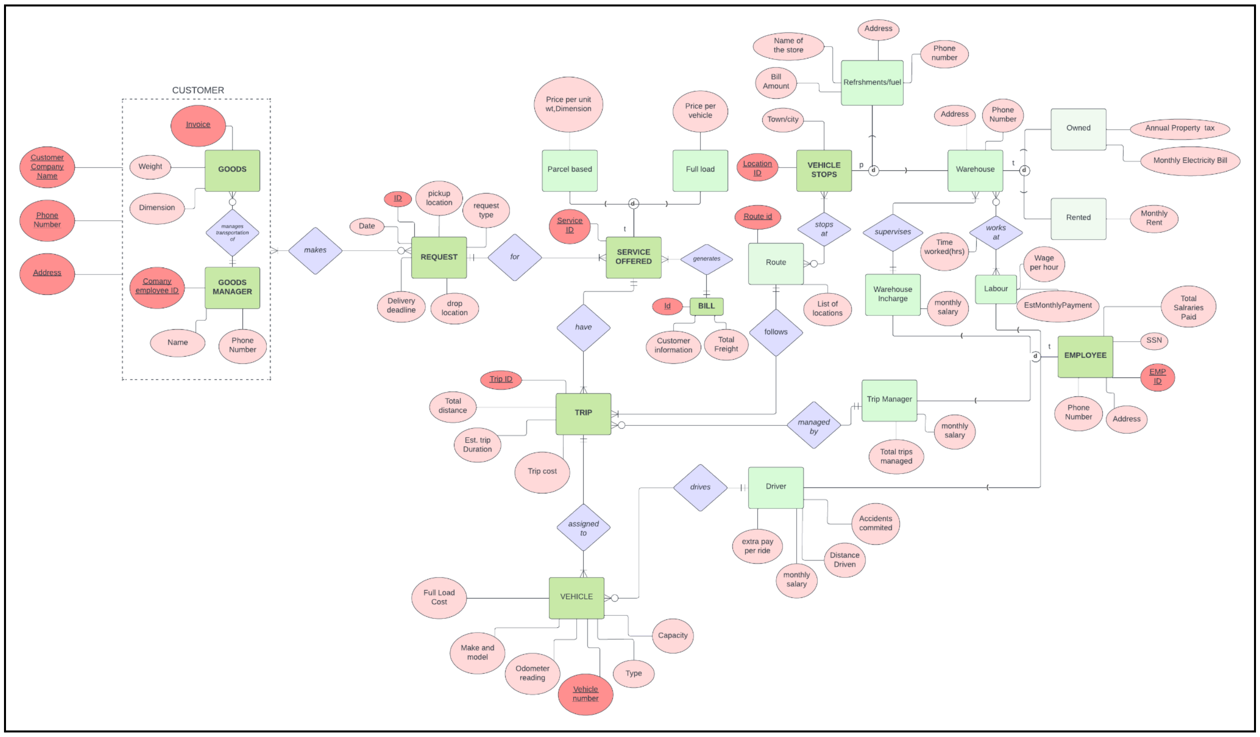


Figure 1: *EER*
***Click here to view the EER more clearly.*

While making a request, the customer can opt for two kinds of services: parcel based service or full load service, where each service has a unique “ID”. In parcel based service, the price is assigned based on the weight and dimensions of the parcel to be delivered, whereas in full load service, the price is assigned for each vehicle required by the customer to make the delivery. Bills are generated for all the services, and each bill is characterized by a unique “ID”, “total freight”, and “customers information”. For each service offered a Bill is generated with a unique Bill_ID having Bill_Date, Total_Frieght and customer information. The Total Freight is then sent to the customer as the Fare amount.

Services have one or many trips that are to be taken, but every trip is linked to a single service and is uniquely identified by an ID. “Total Trip distance”, “Estimated Trip duration” and “Trip cost” are additional characteristics of the trip that are stored in the database. Every trip is managed by a trip manager who decides the route and assigns vehicles required to complete the trip. A trip manager can manage many trips at a time. One or more vehicles are assigned to a trip. They are identified by their unique “vehicle number”. Each vehicle’s “capacity”, “make and mode”, “type” and “odometer reading” are stored to measure total distance driven per trip and the customer is allowed to choose the type of vehicle required from our inventory. These vehicles are driven by company employed drivers.

A pre-decided route is followed by every trip. Each route has a “route ID” and a “list of locations” that the vehicle needs to stop at. Each vehicle stops at one or more “towns/cities” on the route and the vehicle stop is identified by a “location ID”. At a particular location the vehicle generally stops at an already decided fuel/refreshment store or at the company's warehouse. Fuel/refreshment store is identified by its “address”, “name” and “phone number”; this information is stored to confirm the vehicle’s arrival and departure, which helps in tracking the vehicle. The company has warehouses at every location ID, which is either rented or owned. “Monthly rent” is associated with the warehouses that are rented and is stored in the database. “Address” and “phone number” of every warehouse is stored along with its unique “ID”. Each warehouse is supervised by a warehouse incharge and has laborers working in it.

Elaborating on the various employees working at our company, each one is assigned an employee ID. In addition to this, their SSN, address, and phone number are also stored in the database. All the employees working in the company are divided into 4 separate categories -

1) A Trip manager plans and overlooks all the proceedings related to a particular trip. His monthly salary and the number of trips he manages are stored in the database.
2) A Warehouse incharge manages the inflow and outflow of goods at a specific warehouse. His monthly salary is stored.
3) Each warehouse has a set of laborers that are called upon for loading and unloading purposes. They are paid based on the number of hours they work at the warehouse.
4) Drivers are paid a fixed monthly salary. Their “total distance driven” and “number of accidents committed” are stored in the database to measure their performance relative to other drivers. All drivers are paid additionally per ride; this “extra pay” is fixed per ride and is calculated based on the trip duration. If the trip extends for multiple days (i.e., for more than 24 hours), the extra pay is recalculated by adding his daily food/refreshment and living expenses.

<u>Explanation of the constraints in the UML:</u>

1) self.ParcelBasedgoods.Weight > 11 lbs: This constraint takes the weight of goods from the goods table . A request type , if 'parcel-based', can be considered if and only if the weight of the goods or multiple goods togethers is more than 11lbs. Requests for transportation of goods with weight less than 11 lbs will be declined.
2) self.Worksat.HoursWorked >= 1 hour: This constraint states that for a laborer to be paid hourly, he needs to work for a minimum of 1 hour. If he works for less than 1 hour he will not be paid.

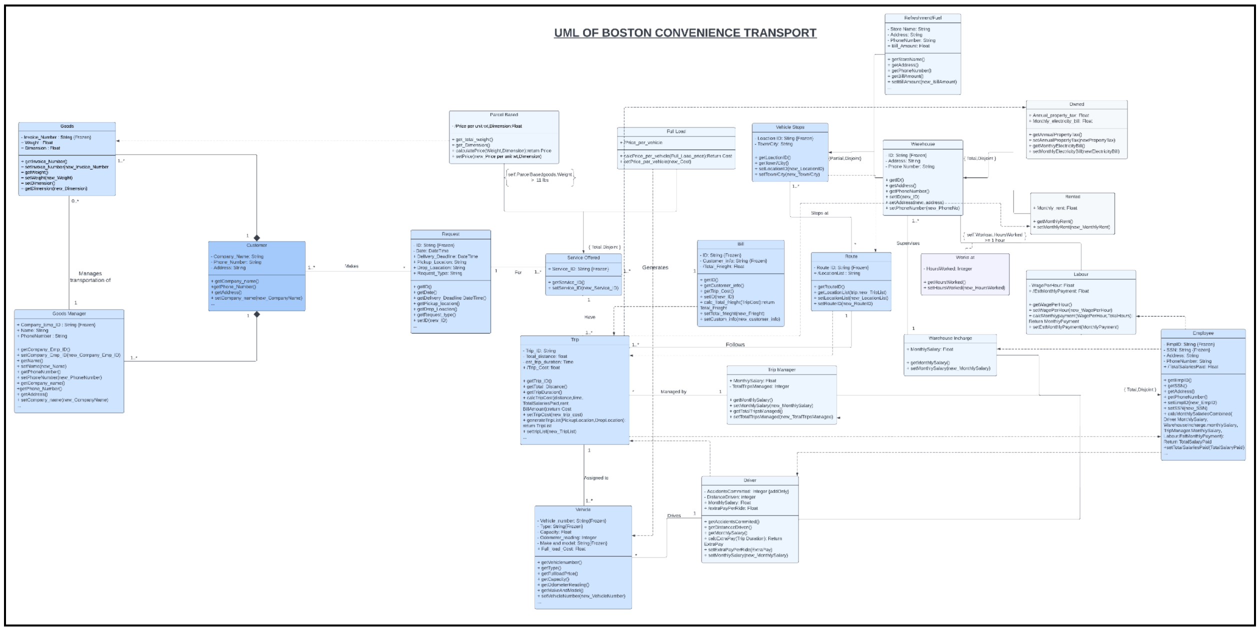


Figure 2: *UML*
***[Click here](#) to view the UML more clearly.*

III. MAPPING CONCEPTUAL MODEL TO RELATIONAL MODEL

GOODS(<u>Invoice</u>, Weight, Dimension, *Employee ID*)

- Invoice: Primary key
- Employee ID: Foreign key refers to Company Employee ID in GOODS MANAGER, NOT NULL

GOODS MANAGER(<u>Company employee ID</u>, Name, Phone Number)

- Company employee ID: Primary key

CUSTOMER(Customer Company Name, Phone Number, Address, <u>Invoice</u>, <u>Customer Employee ID,Bill Id</u>)

- Invoice: Primary key
- Customer Employee ID: Primary key
- This is the aggregate entity
- Bill_Id: Foreign key refers to ID in BILL, NOT NULL

MAKES(*<u>Request_ID</u>*, *<u>Invoice</u>*, *<u>Employee ID</u>*)

- Request ID: Foreign key refers to ID in REQUEST, NOT NULL
- Invoice: Foreign key refers to Invoice in CUSTOMER, NOT NULL
- Employee ID: Foreign key refers to Customer Employee ID in CUSTOMER, NOT NULL
- All the foreign keys together form the primary key of the relation

REQUEST(<u>ID</u>, Date, pickup location, drop location, Delivery deadline, Request type)

- ID: Primary key

PARCEL BASED SERVICE(<u>Service ID</u>, Price per unit, *Request ID, Bill ID*)
- Service ID: Primary key
- Request ID: Foreign key refers to ID in REQUEST, NOT NULL
- Bill ID: Foreign key refers to ID in BILL, NOT NULL

FULL LOAD SERVICE(<u>Service ID</u>, Price per vehicle, *Request ID, Bill ID*)
- Service ID: Primary key
- Request ID: Foreign key refers to ID in REQUEST, NOT NULL
- Bill ID: Foreign key refers to ID in BILL, NOT NULL

While mapping the EER to the relational model, disjointness constraint cannot be enforced in the modeling of the "service offered" entity type into "parcel based service" and "full load service" subclasses.

BILL(<u>ID</u>, Customer Information, Total Freight,Bill_Date)
- ID: Primary key

TRIP(<u>Trip ID</u>, Total distance, Est. trip Duration, Trip cost, *Service ID*, *Route ID*, *Manager Emp ID*)
- Trip ID: Primary key
- Service ID: Foreign key refers to Service ID in PARCEL BASED SERVICE or FULL LOAD SERVICE
- Route ID: Foreign key refers to Route ID in ROUTE
- Manager Emp ID: Foreign key refers to Trip Manager ID in TRIP MANAGER

VEHICLE(<u>Vehicle number</u>, Full Load cost, Make, Model, Odometer reading, Type, Capacity, *Trip ID*, *Driver Emp ID*)
- Vehicle number: Primary key
- Trip ID: Foreign key refers to Trip ID in TRIP, NOT NULL
- Driver Emp ID: Foreign key refers to Driver ID in DRIVER, NOT NULL

ROUTE(<u>Route ID</u>, List of locations)
- Route ID: Primary key

STOPS AT(<u>*Route ID*</u>, <u>*Location ID*</u>)
- Route ID: Foreign key refers to Route ID in ROUTE, NOT NULL
- Location ID: Foreign key refers to Store ID/Warehouse ID in REFRESHMENT AND FUEL and WAREHOUSE respectively, NOT NULL
- Both the foreign keys together make the primary key of the relation

REFRESHMENTS AND FUEL(<u>*Store ID*</u>, Store name, Address, Phone number, Bill Amount)
- Store ID: Foreign key refers to Location ID in VEHICLE STOPS, NOT NULL; it is also the primary key of the relation

WAREHOUSE(<u>*Warehouse ID*</u>, Address, Phone number, Annual Property tax, Monthly Electricity Bill, Monthly rent, *Warehouse Incharge ID*)
- Warehouse ID: Foreign key refers to Location ID in VEHICLE STOPS, NOT NULL; it is also the primary key of the relation

- Warehouse Incharge ID: Foreign key refers to Warehouse Incharge ID in WAREHOUSE INCHARGE

While mapping the EER to the relational model, disjointness constraint cannot be enforced in the modeling of the "vehicle stops" entity type into "refreshment and fuel" and "warehouse" subclasses.

WORKS AT(*Warehouse ID*, *Labour ID*, Time worked)

- Warehouse ID: Foreign key refers to Warehouse ID in WAREHOUSE, NOT NULL
- Labour ID: Foreign key refers to Labour ID in LABOR, NOT NULL
- Both the foreign keys together make up the primary key of the relation

LABOUR(Labour ID, Total Salaries paid, SSN, Address, Phone number, Wage per hour, Est. Monthly Payment)

- Labour ID: Primary key

WAREHOUSE INCHARGE(Warehouse Incharge ID, Total Salaries paid, SSN, Address, Phone number, monthly salary)

- Warehouse Incharge ID: Primary key

TRIP MANAGER(Trip Manager ID, Total Salaries paid, SSN, Address, Phone number, Total trips manager, monthly salary)

- Trip Manager ID: Primary key

DRIVER(Driver ID, Total Salaries paid, SSN, Address, Phone number, extra pay per ride, monthly salary, Distance Driven, Accidents committed)

- Driver ID: Primary key

While mapping the EER to the relational model, disjointness constraint cannot be enforced in the modeling of the "Employee" entity type into "labor", "warehouse incharge", "trip manager", and "driver" subclasses.

IV. IMPLEMENTATION OF RELATIONAL MODEL VIA MYSQL AND NOSQL

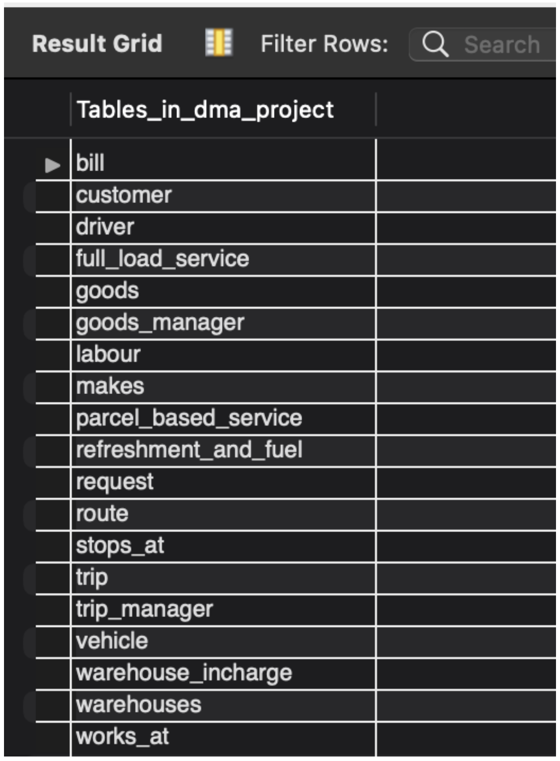
Implementation in MySQL:

Creating our database "DMA_PROJECT":

```
create schema DMA_PROJECT;
show schemas;
use DMA_PROJECT;
```

Displaying all the tables in our database:

```
show tables;
```

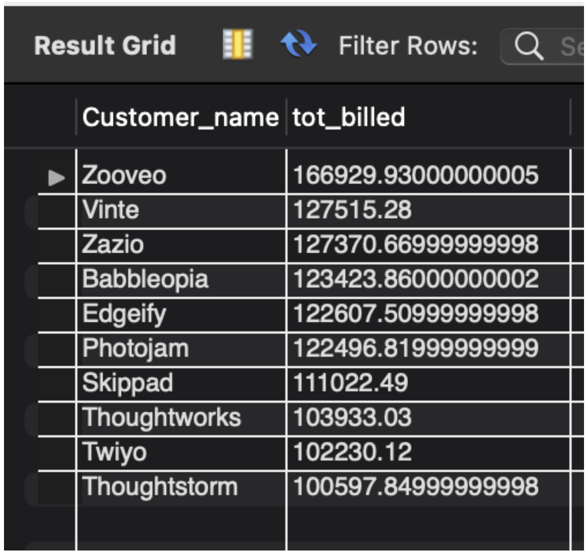


Tables_in_dma_project
bill
customer
driver
full_load_service
goods
goods_manager
labour
makes
parcel_based_service
refreshment_and_fuel
request
route
stops_at
trip
trip_manager
vehicle
warehouse_incharge
warehouses
works_at

<u>Query 1</u>

select c.Customer_name, sum(b.total_freight) as tot_billed from customer c, bill b where c.bill_id = b.bill_id group by c.Customer_name order by tot_billed desc limit 10;

Output:



Customer_name	tot_billed
Zooveo	166929.93000000005
Vinte	127515.28
Zazio	127370.66999999998
Babbleopia	123423.86000000002
Edgeify	122607.50999999998
Photojam	122496.81999999999
Skippad	111022.49
Thoughtworks	103933.03
Twiyo	102230.12
Thoughtstorm	100597.84999999998

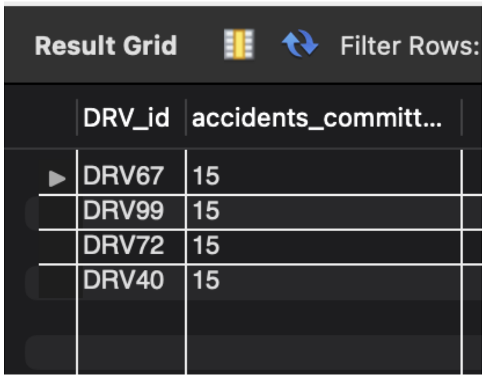
Analytic Purpose:

This query gives a list of top 10 customers who carry out the most business with Boston Convenience. To appreciate their loyalty with us, we can give them a better quote so that they continue doing business with us. We can also send them goodies for being the most loyal clients.

Query 2

select DRV_id, accidents_committed from driver where accidents_committed = (select max(accidents_committed) from driver);

Output:



Result Grid Filter Rows:

DRV_id	accidents_committ...
DRV67	15
DRV99	15
DRV72	15
DRV40	15

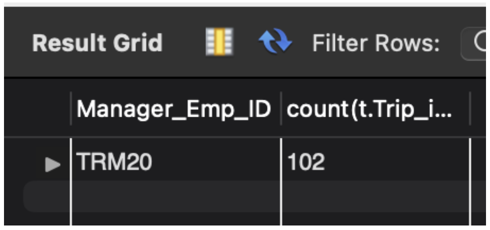
Analytic Purpose:

This query returns those drivers who have committed the most number of accidents till date. It can be used to provide a warning to these drivers, and they can be put under notice for a period of time, until they clear their record (by assigning them shorter trips or reduction in their salary).

Query 3

select t.Manager_Emp_ID, count(t.Trip_id) from trip_manager tm, trip t where tm.TRM_id = t.Manager_Emp_ID group by t.Manager_Emp_ID order by count(t.Trip_id) desc limit 1;

Output:



Result Grid Filter Rows:

Manager_Emp_ID	count(t.Trip_i...
TRM20	102

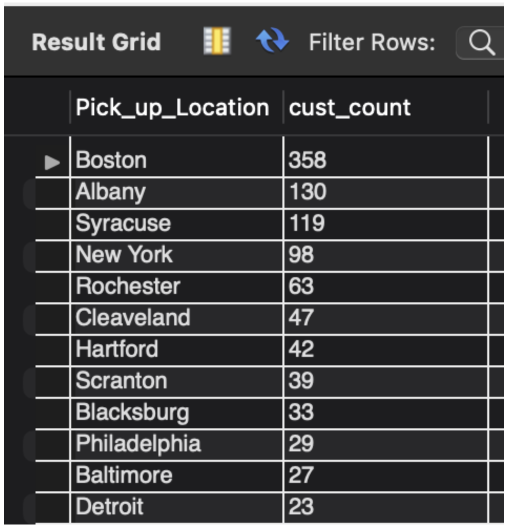
Analytic Purpose:

This query gives the trip manager who has overlooked the most number of trips till date. This way, we can assign this manager to high profile clients and a performance bonus would be provided to him/her.

Query 4

select r.Pick_up_Location,count(c.Customer_name) as cust_count from customer c, makes m, request r where c.Invoice_Number = m.Invoice_Number and c.Employee_ID=m.Employee_ID and m.req_id = r.req_id group by r.Pick_up_Location order by cust_count desc;

Output:



Pick_up_Location	cust_count
Boston	358
Albany	130
Syracuse	119
New York	98
Rochester	63
Cleaveland	47
Hartford	42
Scranton	39
Blacksburg	33
Philadelphia	29
Baltimore	27
Detroit	23

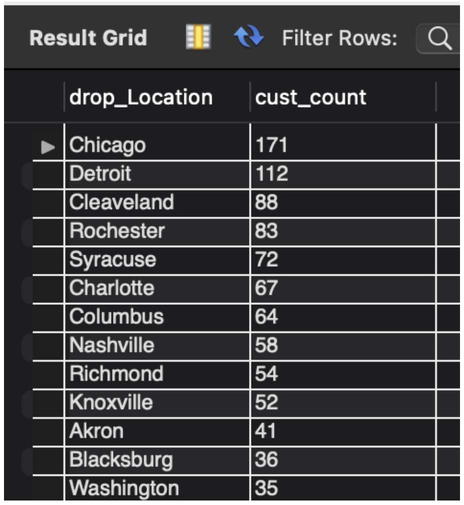
Analytic purpose:

From this query, it is observed that Boston is the most frequently chosen location for the pick up of delivery by the customers. Therefore, we can arrange extra services (like transporting their goods from their warehouse to our warehouse) and work towards arranging more rented/owned warehouses in and around Boston.

<u>Query 5</u>

select r.drop_Location,count(c.Customer_name) as cust_count from customer c,makes m,request r where c.Invoice_Number=m.Invoice_Number and c.Employee_ID=m.Employee_ID and m.req_id=r.req_id group by r.drop_Location order by cust_count desc;

Output:



drop_Location	cust_count
Chicago	171
Detroit	112
Cleaveland	88
Rochester	83
Syracuse	72
Charlotte	67
Columbus	64
Nashville	58
Richmond	54
Knoxville	52
Akron	41
Blacksburg	36
Washington	35

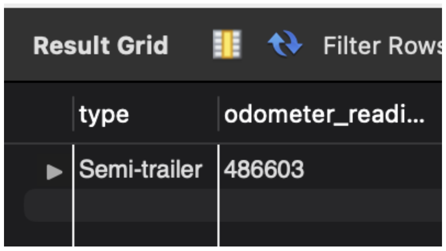
Analytic purpose:

From this query, it is observed that Chicago is the most frequently chosen location for the drop off of goods by the customers. Therefore, we can arrange extra services (like transporting their goods from our warehouse to their warehouse) and work towards arranging more rented/owned warehouses in and around Chicago. In addition to this, we can start our business expansion from these cities

Query 6

select v.type,(v.odometer_reading) from vehicle v order by v.odometer_reading desc limit 1;

Output:



type	odometer_readi...
Semi-trailer	486603

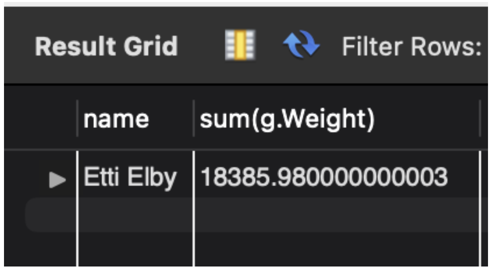
Analytic Purpose:

From this query, it is observed that Semi-trailer trucks are being preferred the most by clients. This implies that the demand is high for these trucks relative to other types of trucks. Therefore, this can be considered while purchasing new trucks in the future.

Query 7

select concat(gm.first_name, ' ', gm.last_name) as name, sum(g.Weight) from goods g, goods_manager gm where gm.emp_id = g.Employee_ID group by name order by sum(g.Weight) desc limit 1;

Output:



name	sum(g.Weight)
Etti Elby	18385.980000000003

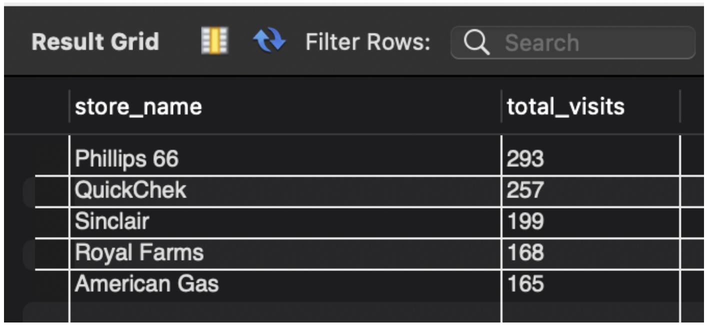
Analytic purpose:

This query returns the name of the goods manager who has prioritized our company for transportation over the rest of the competition. He can be offered extra commission to recommend our services to his client/customer list.

Query 8

select rf.store_name,count(rf.store_name) as total_visits from refreshment_and_fuel rf,stops_at sa where rf.store_id=sa.Location_ID group by rf.store_name order by total_visits desc limit 5;

Output:

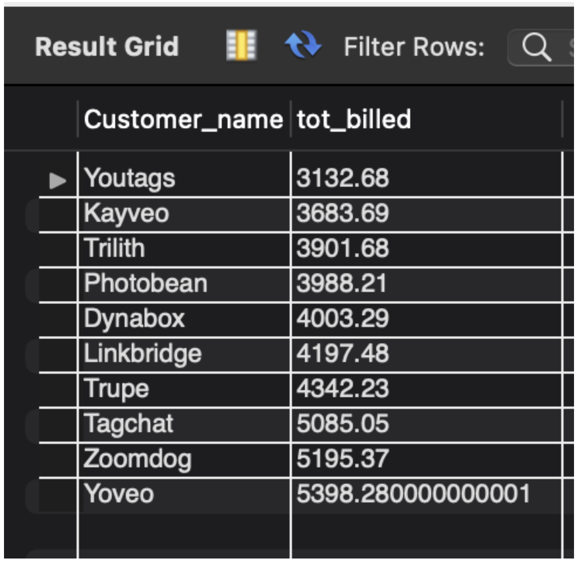


Result Grid Filter Rows: Search

store_name	total_visits
Phillips 66	293
QuickChek	257
Sinclair	199
Royal Farms	168
American Gas	165

Analytic purpose:
The above five refreshment/fuel stores are the ones that are most stopped at by our trucks, during any trip. In the future, we can partner with these stores to avail loyal customer discounts, that would decrease our trip cost, and therefore increase our profit margin.

Query 9
select c.Customer_name, sum(b.total_freight) as tot_billed from customer c, bill b where c.bill_id = b.bill_id group by c.Customer_name order by tot_billed limit 10;
Output:

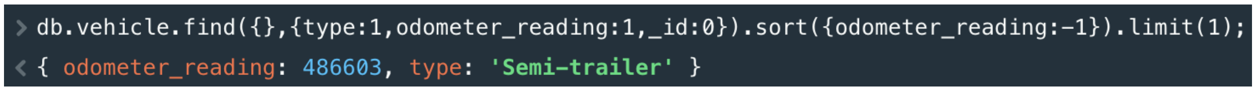


Result Grid Filter Rows:

Customer_name	tot_billed
Youtags	3132.68
Kayveo	3683.69
Trilith	3901.68
Photobean	3988.21
Dynabox	4003.29
Linkbridge	4197.48
Trupe	4342.23
Tagchat	5085.05
Zoomdog	5195.37
Yoveo	5398.280000000001

Analytic Purpose:
This query gives a list of 10 companies that have done the least business with Boston Convenience. We can concentrate our marketing towards these companies and try to adjust our quote that would attract them towards our service.

Implementation in NoSQL:
Query 1:
db.vehicle.find({},{type:1,odometer_reading:1,_id:0}).sort({odometer_reading:-1}).limit(1);
Output:



```
> db.vehicle.find({},{type:1,odometer_reading:1,_id:0}).sort({odometer_reading:-1}).limit(1);
< { odometer_reading: 486603, type: 'Semi-trailer' }
```

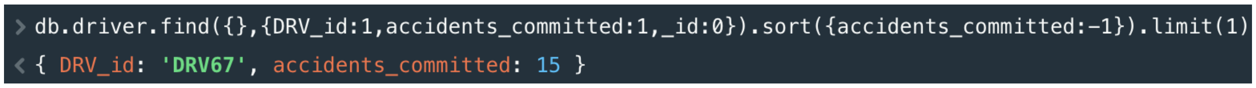
Analytic Purpose:

From this query, it is observed that Semi-trailer trucks are being preferred the most by clients. This implies that the demand is high for these trucks relative to other types of trucks. Therefore, this can be considered while purchasing new trucks in the future.

<u>Query 2:</u>

db.driver.find({},{DRV_id:1,accidents_committed:1,_id:0}).sort({accidents_committed:-1}).limit(1);

Output:



```
> db.driver.find({},{DRV_id:1,accidents_committed:1,_id:0}).sort({accidents_committed:-1}).limit(1)
< { DRV_id: 'DRV67', accidents_committed: 15 }
```

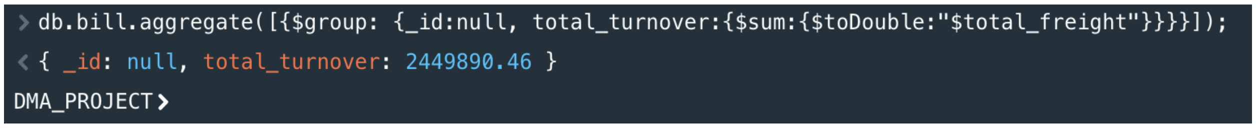
Analytic Purpose:

This query returns those drivers who have committed the most number of accidents till date. It can be used to provide a warning to these drivers, and they can be put under notice for a period of time, until they clear their record (by assigning them shorter trips or reduction in their salary).

<u>Query 3:</u>

db.bill.aggregate([{$group: {_id:null, total_turnover: {$sum:{$toDouble:"$total_freight"}}}}]);

Output:



```
> db.bill.aggregate([{$group: {_id:null, total_turnover:{$sum:{$toDouble:"$total_freight"}}}}]);
< { _id: null, total_turnover: 2449890.46 }
DMA_PROJECT>
```

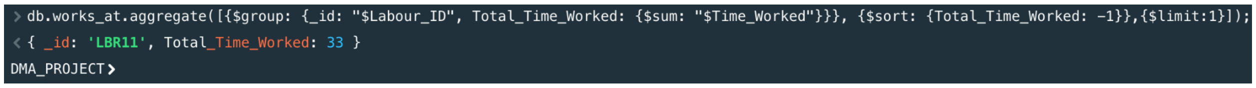
Analytic Purpose:

This query gives the company's total turnover till date. It can be used to compute average yearly income for the company, and can be compared with the actual yearly income, to compute during which years the company performed above average or below average. It also represents the company's status in the market.

<u>Query 4:</u>

db.works_at.aggregate([{$group: {_id: "$Labour_ID", Total_Time_Worked: {$sum:"$Time_Worked"}}}, {$sort: {Total_Time_Worked: -1}},{$limit:1}]);

Output:



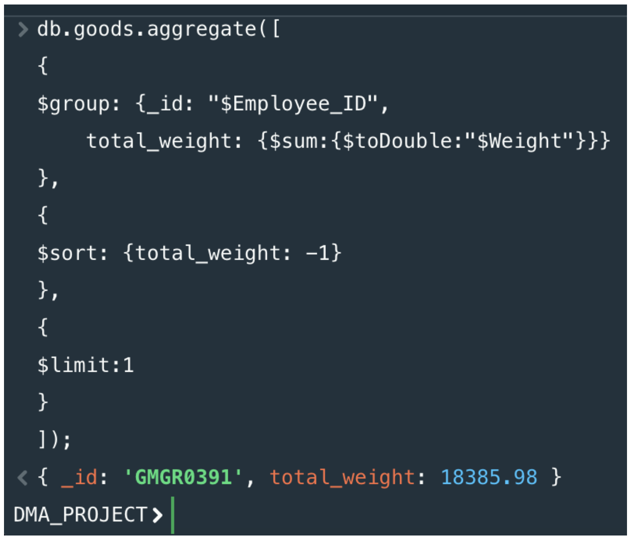
```
> db.works_at.aggregate([{$group: {_id: "$Labour_ID", Total_Time_Worked: {$sum: "$Time_Worked"}}}, {$sort: {Total_Time_Worked: -1}},{$limit:1}]);
< { _id: 'LBR11', Total_Time_Worked: 33 }
DMA_PROJECT>
```

Analytic Purpose:

This query is used to find the laborer who has worked for the most number of hours till date. He can be given the liberty to choose his own shifts, or he could be given an extra leave. If this particular laborer continues to work diligently even in the future, his hourly wage can be increased; he can be rewarded with a loyalty bonus.

Query 5:
db.goods.aggregate([{$group: {_id: "$Employee_ID", total_weight: {$sum:{$toDouble:"$Weight"}}}},{$sort: {total_weight: -1}},{$limit:1}]);

Output:



```
> db.goods.aggregate([
  {
  $group: {_id: "$Employee_ID",
      total_weight: {$sum:{$toDouble:"$Weight"}}}
  },
  {
  $sort: {total_weight: -1}
  },
  {
  $limit:1
  }
  ]);
< { _id: 'GMGR0391', total_weight: 18385.98 }
DMA_PROJECT>
```

Analytic Purpose:

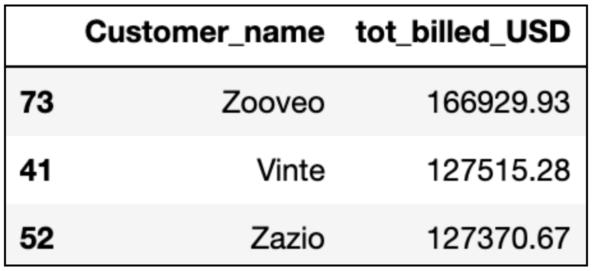
This query returns the name of the goods manager who has prioritized our company for transportation over the rest of the competition. He can be offered extra commission to recommend our services to his client/customer list.

V. DATABASE ACCESS VIA PYTHON

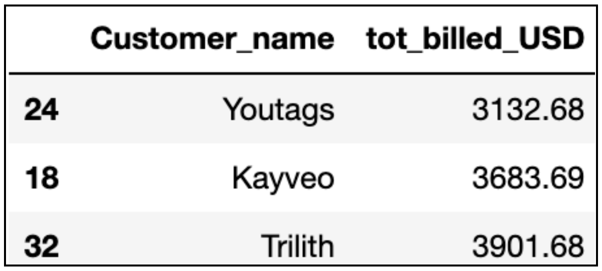
Accessing the MySQL database through Jupyter Notebook:

pip install mysql-connector-python
import mysql.connector
from mysql.connector import Error import pandas as pd
connection = mysql.connector.connect(user='root', password='Aishu.22112000',
host='localhost',
database='DMA_PROJECT')

Customers who carry out the most business and least business with Boston Convenience:



	Customer_name	tot_billed_USD
73	Zooveo	166929.93
41	Vinte	127515.28
52	Zazio	127370.67



	Customer_name	tot_billed_USD
24	Youtags	3132.68
18	Kayveo	3683.69
32	Trilith	3901.68

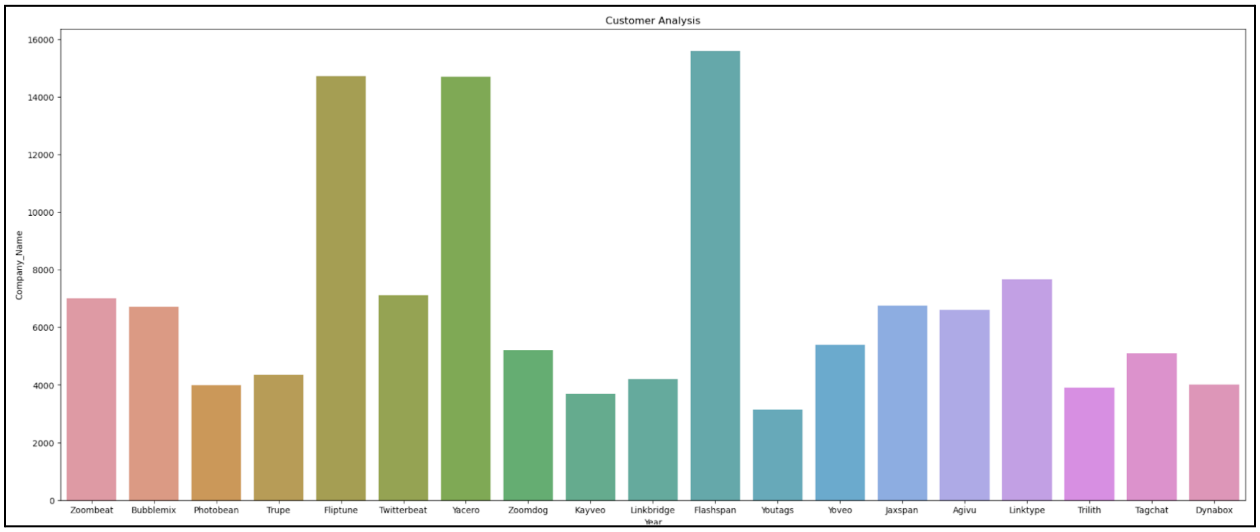
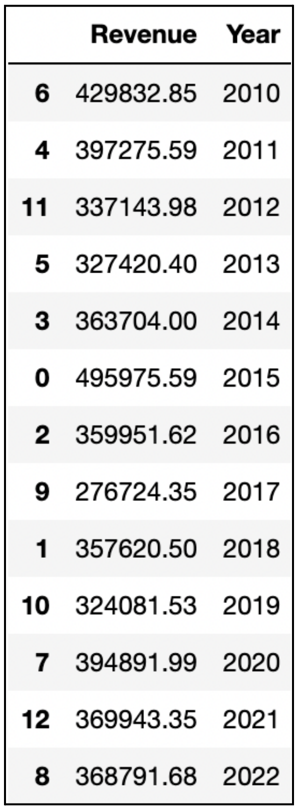


Figure 3: *Bar plot visualization for clients who barely used our service:*
Their total_bill < (sum(total_bills)).mean() - (sum(total_bills)).std()

**The python script as well as the enlarged image of the above analytics visualization can be viewed in the python program file (in the artifacts of Milestone 6).

<u>Yearly Turnover:</u>



	Revenue	Year
6	429832.85	2010
4	397275.59	2011
11	337143.98	2012
5	327420.40	2013
3	363704.00	2014
0	495975.59	2015
2	359951.62	2016
9	276724.35	2017
1	357620.50	2018
10	324081.53	2019
7	394891.99	2020
12	369943.35	2021
8	368791.68	2022

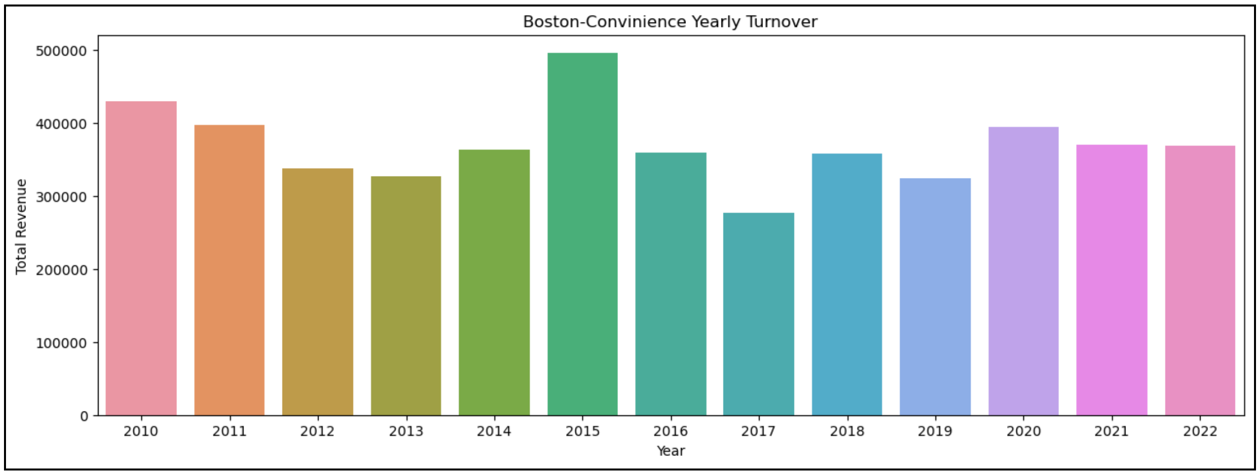
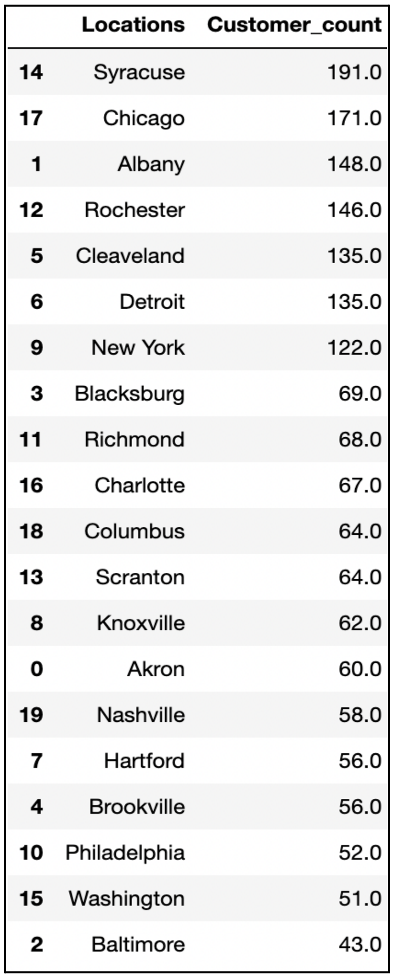


Figure 4: *Bar plot visualization for the yearly turnover at our company*

**The python script as well as the enlarged image of the above analytics visualization can be viewed in the python program file (in the artifacts of Milestone 6).

<u>Business Expansion: most frequently chosen pick-up and drop-off locations:</u>



	Locations	Customer_count
14	Syracuse	191.0
17	Chicago	171.0
1	Albany	148.0
12	Rochester	146.0
5	Cleaveland	135.0
6	Detroit	135.0
9	New York	122.0
3	Blacksburg	69.0
11	Richmond	68.0
16	Charlotte	67.0
18	Columbus	64.0
13	Scranton	64.0
8	Knoxville	62.0
0	Akron	60.0
19	Nashville	58.0
7	Hartford	56.0
4	Brookville	56.0
10	Philadelphia	52.0
15	Washington	51.0
2	Baltimore	43.0

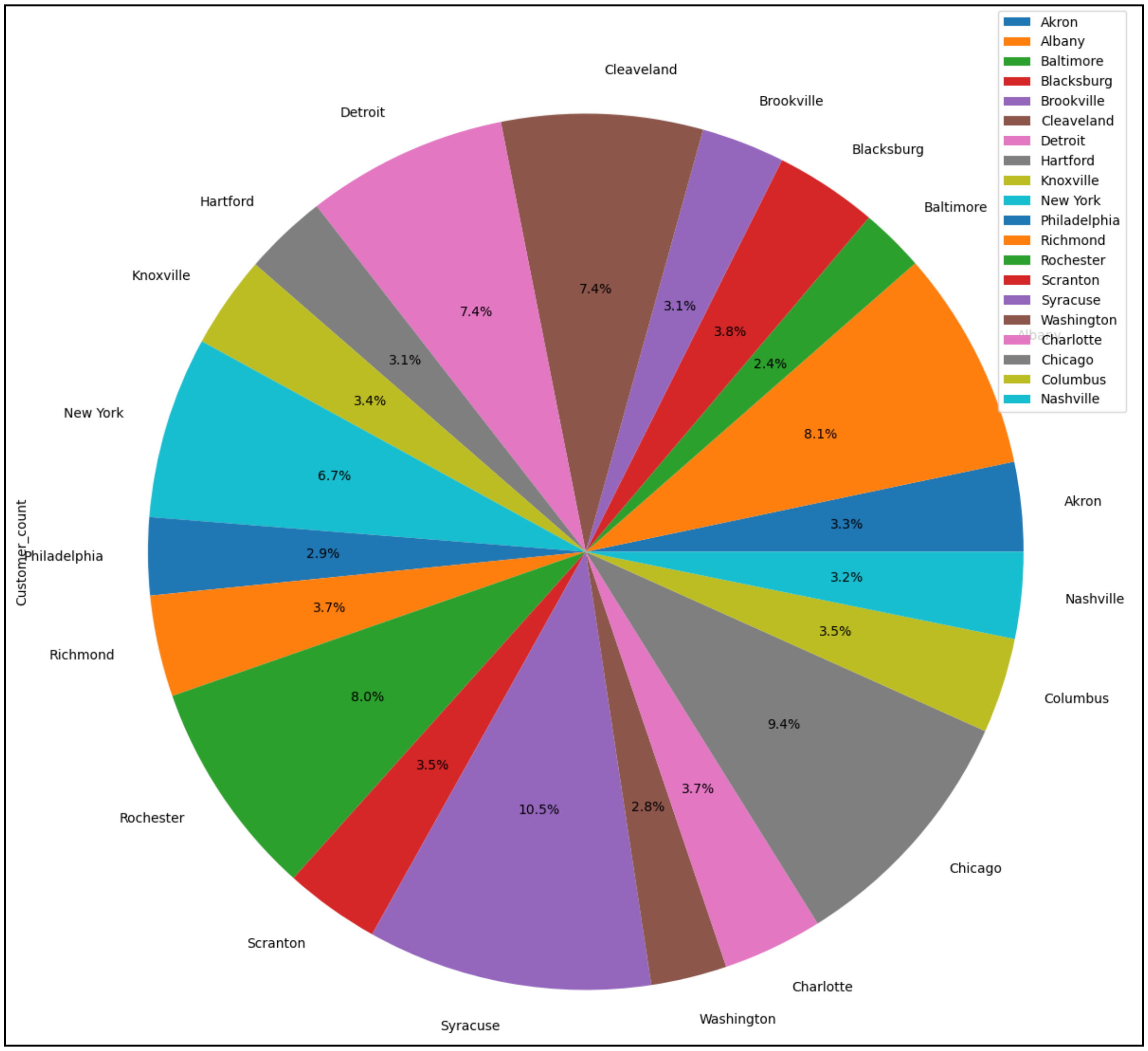


***Figure 5:** Pie chart visualization for the number of times each location is chosen for either pick-up or drop-off*

**The python script as well as the enlarged image of the above analytics visualization can be viewed in the python program file (in the artifacts of Milestone 6).

VI. SUMMARY AND RECOMMENDATION

Majority of the logistic companies deal with their accounts and customers manually follow a paper trial. It becomes tedious to maintain records of past shipments, all the bills generated, vehicle tracking, and customer information. With all the data spread out in different sheets of paper it is difficult to segregate and analyze the information, which leads to no growth or minimal growth of the company over the years. Moreover, important pieces of information on paper can easily be lost when needed the most, for example - during a scenario of court cases or customer billing. Idea for creating and managing a database was inspired by these reasons.

This database is designed for a logistic company called "**Boston Convenience**" - which is a B2B transport company that offers four routes which travel via 21 pick-up and drop-off locations. The routes are as follows:

Route number	Location 1	Location 2	Location 3	Location 4	Location 5	Location 6	Location 7
1	Boston	New York	Philadelphia	Washington	Richmond	Charlotte	-
2	Boston	Albany	Syracuse	Rochester	Cleveland	Detroit	Chicago
3	Boston	Hartford	Scranton	Brookville	Akron	Columbus	-
4	Boston	New York	Baltimore	Blacksburg	Knoxville	Nashville	-

Like any other logistic company, our database will be able to store the details of services offered, requests made, trips taken, vehicles owned, and customer and employee information. **Workflow** of our database is as follows: Customer makes a request for transportation. Services are provided with respect to the type of request initiated by the customer (either full load based or parcel based). For a service offered, a bill is generated and trips are initiated. One or more vehicles with drivers are assigned to a trip. A trip is assigned to one of the available trip managers. A trip follows a particular route based on the pick up and drop off locations. Vehicle either stops for fuel/refreshment or at a particular warehouse (rented or owned) at the locations decided in the route. Each warehouse has one warehouse incharge and at least one laborer for loading and unloading purposes.

Our Database can provide key insights to make critical business decisions. Performing analytics on our database can give us information that can help us classify customers based on their Business capacity. Respective to their class, corresponding **Inbound Marketing methods** can be implemented. After a successful run, entrepreneurs are often startled and clueless when it comes to **Business Expansion**. Data can be used to find potential hubs/cities for business expansion. This information can be used to partner with emergent transportation companies for increasing our clientbase. As portrayed above we have discovered that Syracuse and Chicago can be chosen as a second base of operation for "Boston Convenience". Meeting customer requirements while maintaining a progressive workplace for employees can be made easy by using data and creating an **Employee Ranking System**. Increment/Decrement in salaries and Promotions/Demotions can be decided based on simple statistical analysis based on employee performance. In addition to these, our DBMS can be used to justify small decisions like which type of vehicle to purchase.

Our DBMS model performs all functions of a logistic company systematically and efficiently while saving time and providing all the required information for taking critical business decisions. We have successfully implemented Analytics for Business Expansion, Methods for Inbound Marketing towards Customers, and Employee Management System. Same Database Model can be replicated and used for multiple Logistics Companies.

Adding multiple transport companies to our database can increase the network of Pickup and Drop locations. Live tracking of vehicles, automated trip cost calculation, customer specific commission prediction using ML and Employee ranking system are some of the upgrades that can be recommended for the future.