USE CASE STUDY REPORT

Group No: Group 02

Students Names: Karan Paresh and Rashmi Nambiar Pappinisseri Puthenveettil

Executive Summary:

The primary objective of the case study is to implement a database model for a vehicle insurance company. Vehicle insurance company requires information on their customers, vehicles owned by them, insurance policy details like premium and validity, claims made in case of accidents and payment made by the company to the customers.

Three main methods are used for implementation of the model. First is a MySQL implementation wherein all the entity types aforementioned along with the required relationship types are modelled as per the relational model. Random data are generated using www.generatedata.com and obtained as excel files which are then used to create the tables for the schema by importing them to MySQL Workbench.

Secondly, NoSQL implementation of the partial model is done using Neo4j sandbox online environment wherein queries are performed using Cypher. Here, customer, vehicle, accident and policy are taken as nodes along with the relationships among them. Even with a partial implementation, useful insights could be obtained through meaningful queries.

Finally, the MySQL database is accessed using R. This proves to be valuable in terms of analysis of data using visualization and can provide the insurance company using this database insights on quarterly revenue or loss based on number of claims settled and payment made during the corresponding period.

The model can be used by claim analysts or underwriters to track the customers with more claims, the models of vehicles which are damaged the most and the locations prone to higher number of accidents, among others. This would not only enable the company to perform a risk assessment by tracking those customers who cause significant loss, but also help in adjusting premium amount to be charged in certain regions and improving sales in other regions. Further, more claim analysts can be assigned to those regions where accidents occur in higher numbers. The model can be improved to include vehicles which do not participate in claim but are insured. Nevertheless, the model proves to add value to the insurer by providing insights on their customers.

I. Introduction

Auto insurance is a contract between a customer who owns one or more vehicles and an insurance company which protects the customer against financial loss in the event of an accident or theft of the vehicle(s). The insurance company agrees to pay part or whole of losses as stated in the insurance policy in exchange for a premium amount (hereafter stated as ‘premium’) from the customer. The problem here is based on insurance covered in the event of an accident involving one or more vehicles.

When a vehicle accident occurs, the person involved (consider the driver) contacts the enquiry center of the insurance company with which the owner of the vehicle has a contract. The enquiry center seeks information such as policy number, date of accident, the details of accident and sequence of events, driving license (DL) number, name of the driver at the time of accident and the place of occurrence of the incident. The policy number, in turn, reveals the information about the policy holder or name of insured person, validity of policy and whether the policy covers the risk. Moreover, the policy also has details about the vehicle registered, which includes engine number, chassis number, make and model of the vehicle and the validity of the policy. With the information thus obtained from the driver, a claim number is initiated at the enquiry center which is intimated to either a claim analyst or a first-point-of-contact (FPOC). If the vehicle is at the site of the incident, the FPOC assigns the claim analyst, or if the vehicle is moved to a car garage, a claim analyst closest to that location will be assigned immediately. In either case, the claim analyst checks some basic documents pertaining to the accident such as claim form which needs to be filled out by the customer detailing the events occurred, DL of the driver, registration certificate (RC) of the vehicle (containing the registration number, chassis number and engine number of the vehicle) which is verified with details of the same as specified in the policy.

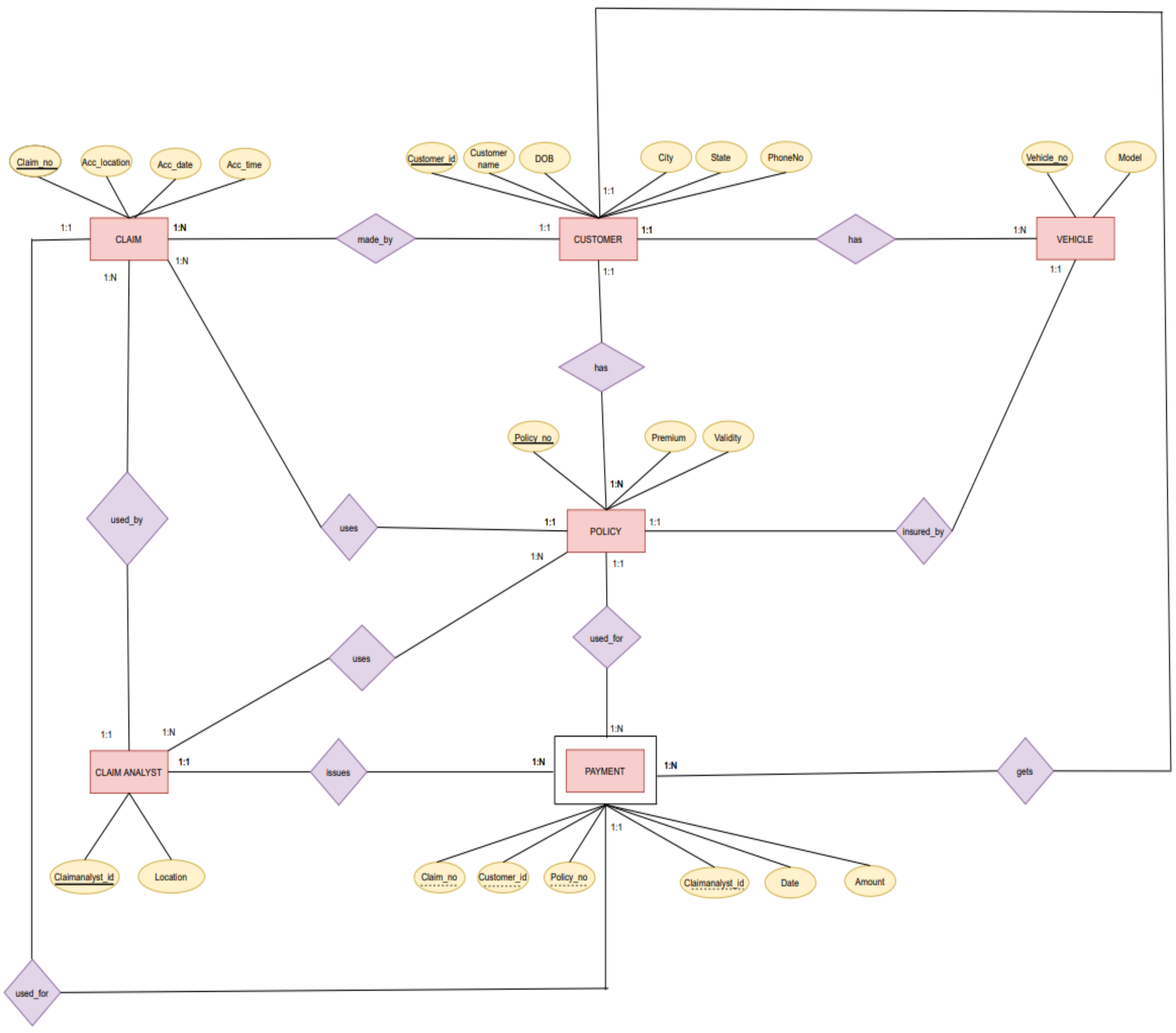
Further, this information is transferred from claim analyst to an underwriter to decides whether the risk needs to be considered or not. If considered, the customer receives the amount as calculated by the underwriter. Further, the underwriter maintains a record of number of claims made by the customer and number of policies issued to the customer, in case of ownership of multiple vehicles, and the premium paid. A cost assessment reveals how frequently the customer makes claims, the vehicle type for which damages occur frequently and location where accidents with most vehicle damage occurs (based on claimed amount). This is further used to analyze if the customer creates loss and conveyed to sales department of the insurance company.

In implementing the database, the requirements are as follows:

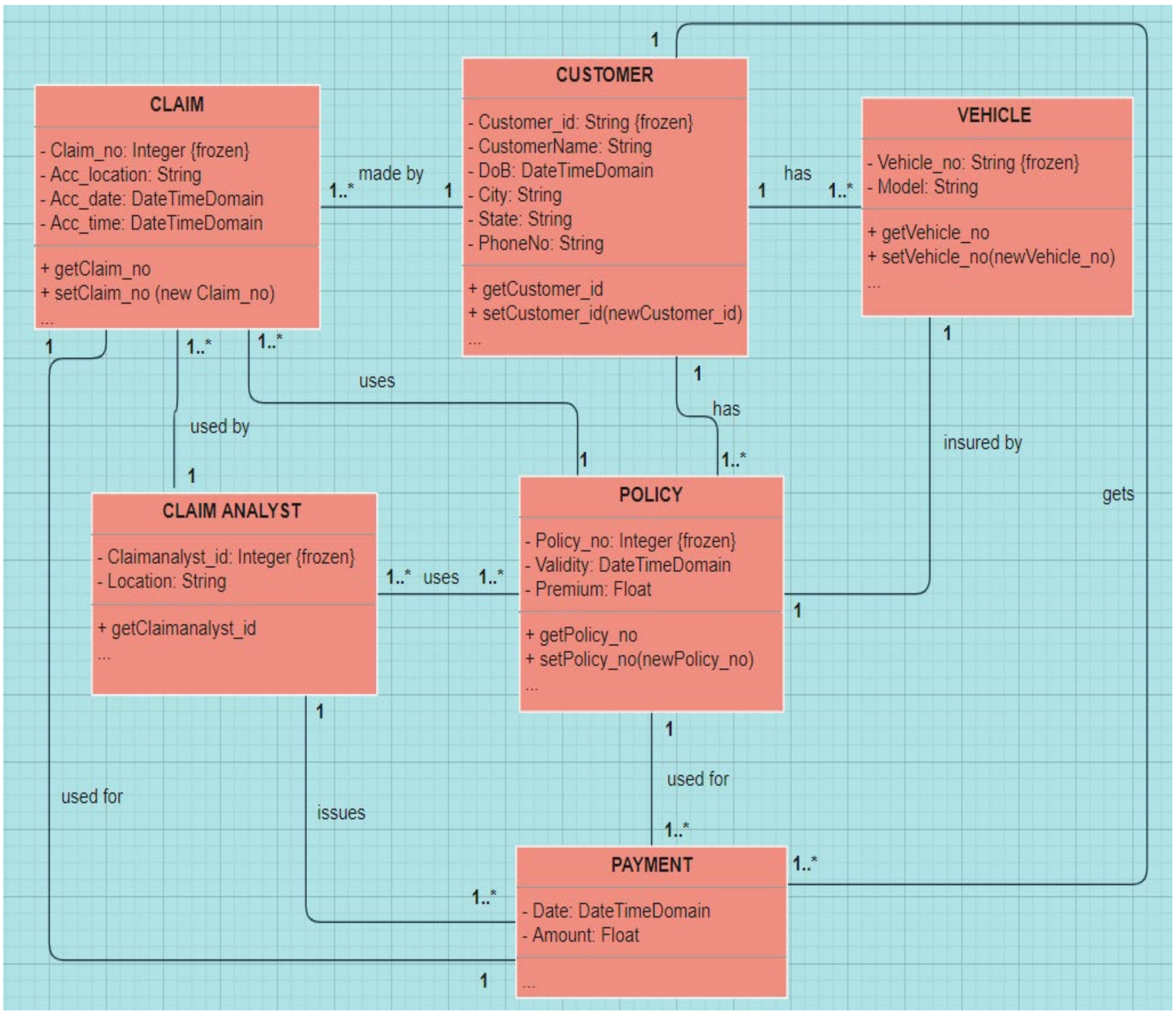
1. A customer owns one or more vehicles.
2. A customer can insure multiple vehicles with the same insurance company.
3. One insurance policy has details of a single vehicle.
4. Claim analyst and/or underwriter uses the database as only vehicles involved in accident are considered.

II. Conceptual Data Modeling:

a. EER Model



a. UML Model



III. Mapping Conceptual Model to Relational Model:

Primary key – <u>Underlined</u>; Foreign key - *Italicized*

- Customer (<u>CustomerID</u>, CustomerName, DoB, City, State,PhoneNo)

- Vehicle (<u>VehicleNo</u>, Model, *CustomerID*)
  CustomerID foreign key refers to CustomerID in Customer; NULL NOT ALLOWED

- Policy (<u>PolicyNo</u>, Premium, Validity, *CustomerID*, *VehicleNo*)
  CustomerID foreign key refers to CustomerID in Customer; NULL NOT ALLOWED
  VehicleNo foreign key refers to VehicleNo in Vehicle; NULL NOT ALLOWED

- Claim (ClaimNo, Acc_location, Acc_date, Acc_time, *CustomerID, PolicyNo, CID*)
  CustomerID foreign key refers to CustomerID in Customer; NULL NOT ALLOWED
  PolicyNo foreign key refers to PolicyNo in Customer; NULL NOT ALLOWED
  CID foreign key refers to CID in ClaimAnalyst; NULL NOT ALLOWED

- ClaimAnalyst(ClaimAnalystID, Location)

- Uses (*CID, PolicyNo*)
  CID foreign key refers to ClaimAnalystID in ClaimAnalyst; NULL NOT ALLOWED
  PolicyNo foreign key refers to PolicyNo in Policy; NULL NOT ALLOWED

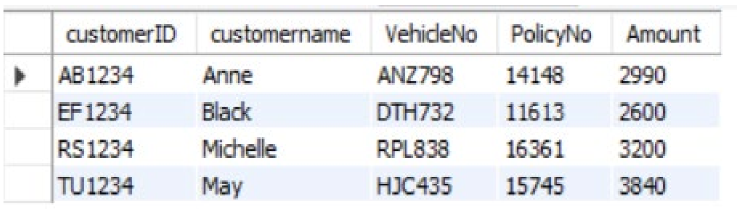
- Payment (*ClaimNo, PolicyNo, CID,CustomerID,* Payment_date, Amount)
  CID foreign key refers to ClaimAnalystID in ClaimAnalyst; NULL NOT ALLOWED
  PolicyNo foreign key refers to PolicyNo in Policy; NULL NOT ALLOWED
  CustomerID foreign key refers to CustomerID in Customer; NULL NOT ALLOWED
  ClaimNo foreign key refers to ClaimNo in Claim; NULL NOT ALLOWED

IV. Implementation of Relational Model via MySQL and NoSQL:

a. MySQL implementation

Query 1: Find customerID, name, vehicle no., policy no. and Amount paid greater than $2500.

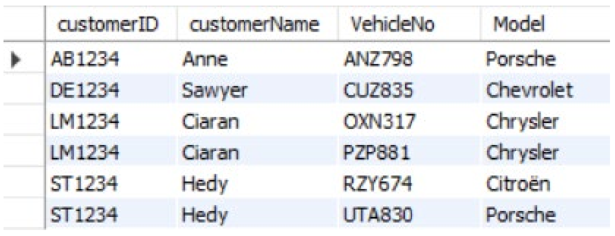
```
select p.customerID, c.customername,
pol.VehicleNo, p.PolicyNo, p.Amount
from customer c, policy pol, payment p
where c.customerID = p.customerID
and pol.PolicyNo = p.PolicyNo
and p.Amount > 2500;
```



customerID	customername	VehicleNo	PolicyNo	Amount
AB1234	Anne	ANZ798	14148	2990
EF1234	Black	DTH732	11613	2600
RS1234	Michelle	RPL838	16361	3200
TU1234	May	HJC435	15745	3840

Query 2: Find CustomerID, name, VehicleNo and model of those customers involved in more than 1 accident.

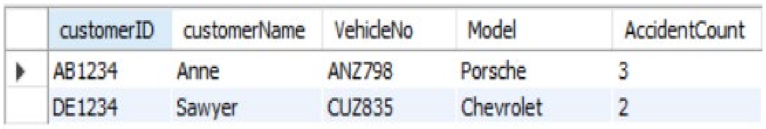
```
select c.customerID , c.customerName, v.VehicleNo, v.Model
from customer c, vehicle v
where c.customerID = v.customerID
and 1 < (select count(*)
        from claim cl
        where c.customerID = cl.customerID
        );
```



customerID	customerName	VehicleNo	Model
AB1234	Anne	ANZ798	Porsche
DE1234	Sawyer	CUZ835	Chevrolet
LM1234	Ciaran	OXN317	Chrysler
LM1234	Ciaran	PZP881	Chrysler
ST1234	Hedy	RZY674	Citroën
ST1234	Hedy	UTA830	Porsche

Query 3: Find CustomerID, name, VehicleNo and model of those vehicles involved in more than 1 accident.

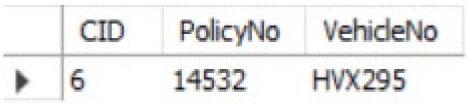
select c.customerID, c.customerName, v.VehicleNo, v.Model, count(*) as AccidentCount
from customer c, vehicle v, claim cl, policy pol
where cl.policyNo=pol.policyNo
and pol.vehicleno=v.vehicleno
and v.customerid = c.customerid
group by cl.policyNo
having AccidentCount > 1;



customerID	customerName	VehicleNo	Model	AccidentCount
AB1234	Anne	ANZ798	Porsche	3
DE1234	Sawyer	CUZ835	Chevrolet	2

Query 4: Find ClaimanalystID, VehicleNo, PolicyNo who worked on Claim number '129'

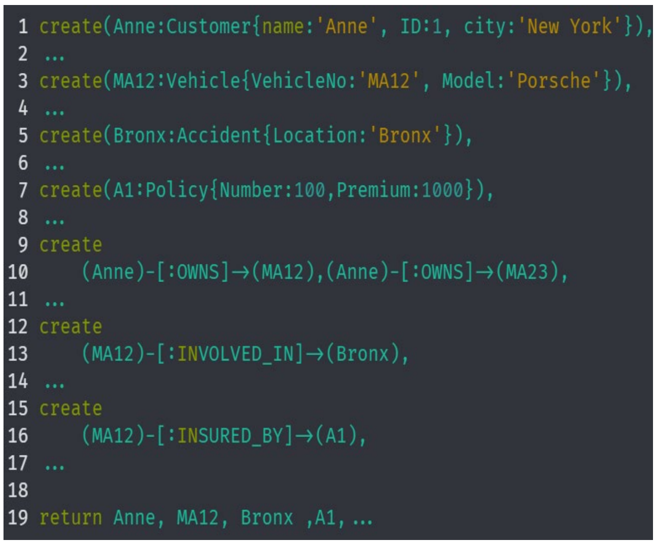
select p.CID, p.PolicyNo, pol.VehicleNo
from payment p, policy pol
where pol.PolicyNo = p.policyNo
and p.ClaimNo = 129;



CID	PolicyNo	VehicleNo
6	14532	HVX295

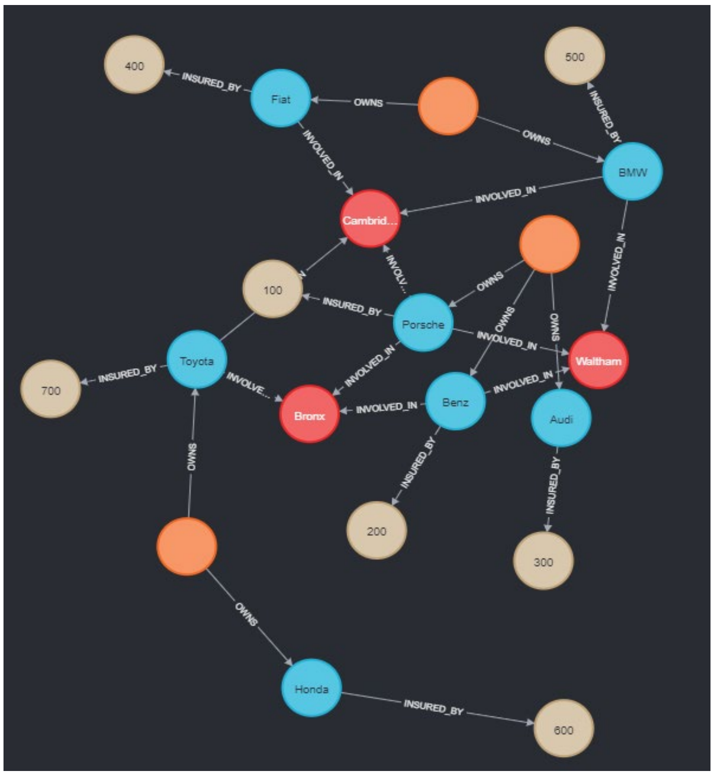
b. NoSQL implementation:

A partial NoSQL implementation of the model is done in Neo4j sandbox environment. The entity types Customer, Vehicle, Accident (instead of Claim entity in MySQL) and Policy are considered as nodes. Following is an example of nodes and relationships created. Query language used here is Cypher.



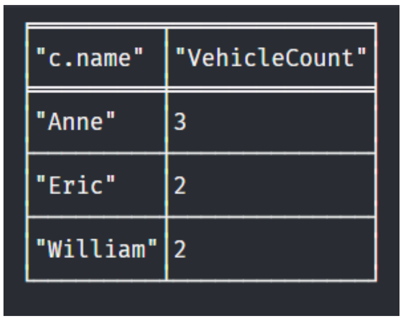
```
create(Anne:Customer{name:'Anne', ID:1, city:'New York'}),
...
create(MA12:Vehicle{VehicleNo:'MA12', Model:'Porsche'}),
...
create(Bronx:Accident{Location:'Bronx'}),
...
create(A1:Policy{Number:100,Premium:1000}),
...
create
    (Anne)-[:OWNS]→(MA12),(Anne)-[:OWNS]→(MA23),
...
create
    (MA12)-[:INVOLVED_IN]→(Bronx),
...
create
    (MA12)-[:INSURED_BY]→(A1),
...

return Anne, MA12, Bronx ,A1, ...
```



Query 1: Find customer names with more than 1 vehicle.

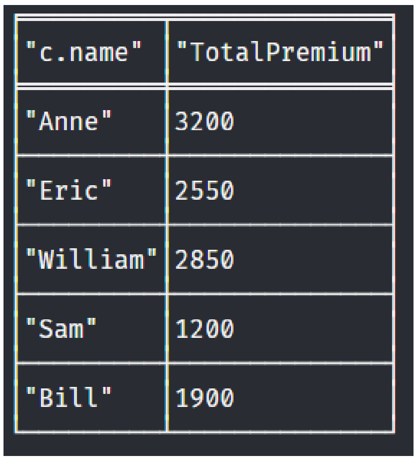
```
match (c:Customer)--(v:Vehicle)
with  c,count(*) as VehicleCount
where VehicleCount > 1
return distinct c.name, VehicleCount
```



"c.name"	"VehicleCount"
"Anne"	3
"Eric"	2
"William"	2

Query 2: Find total premium paid by each customer.

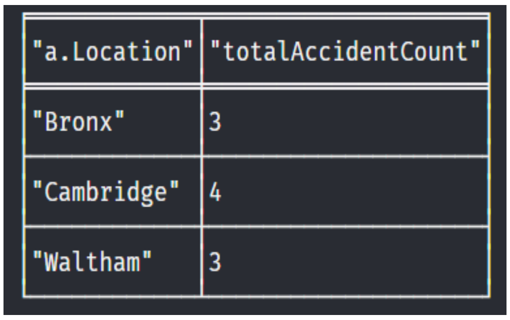
```
match (c:Customer)--(v:Vehicle)--(p:Policy)
with  c, sum(p.Premium) as TotalPremium
return distinct c.name, TotalPremium
```



"c.name"	"TotalPremium"
"Anne"	3200
"Eric"	2550
"William"	2850
"Sam"	1200
"Bill"	1900

Query 3: Find the total number of accidents by location.

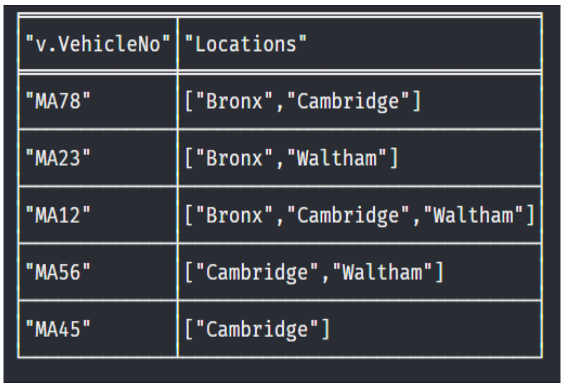
```
match (v:Vehicle)--(a:Accident)
with  a, count(*) as totalAccidentCount
return distinct a.Location, totalAccidentCount
```



"a.Location"	"totalAccidentCount"
"Bronx"	3
"Cambridge"	4
"Waltham"	3

Query 4: Find all locations where a vehicle was involved in an accident.

```
match (v:Vehicle)--(a:Accident)
with  v, collect(a.Location) as Locations
return distinct  v.VehicleNo, Locations
```

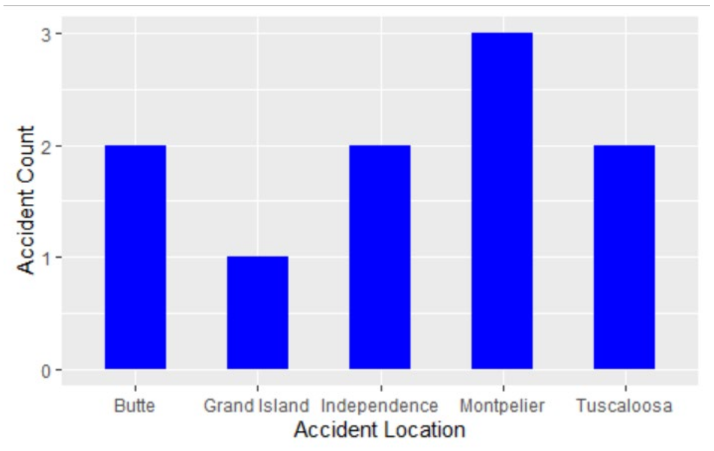


"v.VehicleNo"	"Locations"
"MA78"	["Bronx","Cambridge"]
"MA23"	["Bronx","Waltham"]
"MA12"	["Bronx","Cambridge","Waltham"]
"MA56"	["Cambridge","Waltham"]
"MA45"	["Cambridge"]

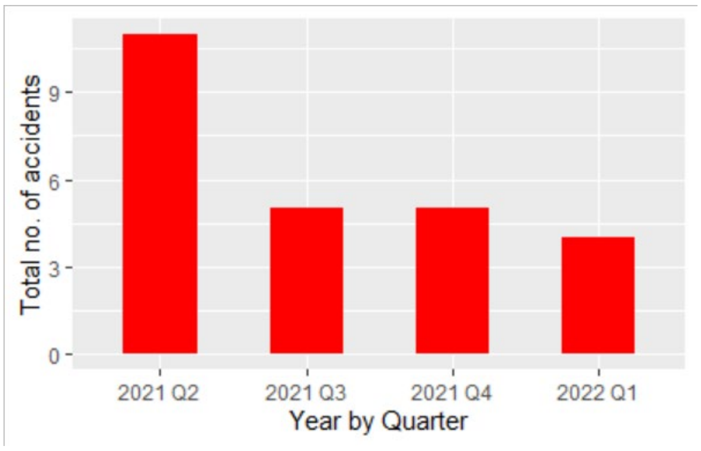
V. Database access via R:

The database in MySQL is accessed via R using RMySQL package. Further analysis and visualization are performed using ggplot.

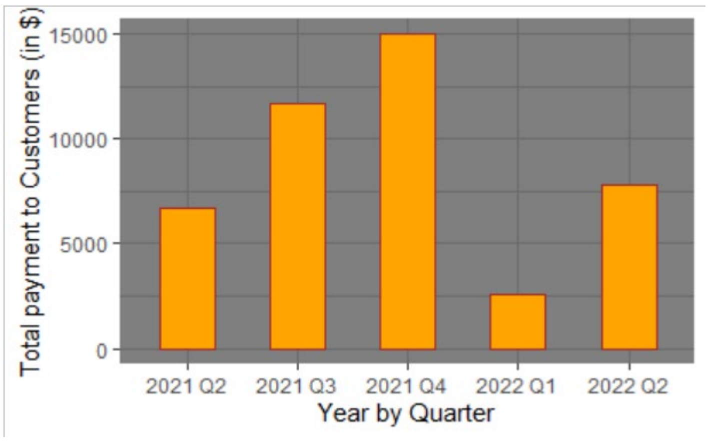
Plot 1: Top 5 accident locations with the accident count.



Plot 2: Total number of accidents by quarter.



Plot 3: Total amount paid by the company to its customers every quarter.



V. Summary and Recommendation

The database for a vehicle insurance company designed on MySQL is one that can be implemented in the industry. It could result in easier access of customer information and useful insights to the same, thereby proving to be valuable to the underwriter of the company who can decide the risk of insuring a customer (his/her vehicles) and loss incurred over a certain period.

Implementation of the model using Neo4j Cypher shows the simplicity of using NoSQL database. As this study includes only a partial implementation of the same, adding all the entity types and relationships would greatly help in realizing the potential of using such an environment. Further, connecting the database to R proves to be insightful in understanding the risk of insuring certain customers.

Several hurdles were faced during the EER modelling of the entire business process to include relevant details that would be required for an insurer while maintaining the simplicity and ease of use. For instance, connection between claim and vehicle entity types seems inevitable but had to be excluded, instead using policy entity type as a link between the two.

The model can be improved to include vehicles which do not participate in claim but are insured. In this study, the focus is primarily on those vehicles involved in accidents. More attribute types can be added to the model, for instance, customer can have his/her driving license number, registration details, vehicle-specific details like engine number and chassis number.