USE CASE STUDY REPORT

Group No.: Group 14

“Vehicle Insurance Policy Data Management System”

Student Names: Yashasvi Sharma and Sunkari Vikas Goud

Executive Summary:

The goal of the study was to implement a robust vehicle insurance database management system which can be used by customers, companies, government, and analysts to draw useful insights and make insurance claiming and tracking process hassle free.

Customers' and the insurance company's required data fields were included in the database's architecture. After the EER and UML diagrams were modeled, the conceptual model was translated to a relational model with the required primary and foreign keys. It was investigated whether this database can be used in a NoSQL context. It was then built in MySQL and MongoDB databases.

The developed database is linked to Python, the customers and insurance firm benefit from the display of the studied data to raise the bar for the Insurance Management System. Additionally, it enables us to anticipate and efficiently manage the resources required by consumers.

I. Introduction

A car insurance claim is a request for financial compensation that a driver files with an insurance company after their vehicle is damaged or they are injured in a car accident. More than $170 billion in auto insurance claims payments are made by U.S. insurance companies each year.

With over 284 million vehicles operating on roads in 2022 throughout the United States, there is a need for a robust vehicle's insurance database management system to be developed to retrieve information regarding the accident, vehicles involved and help customers with their policy and claim their deserved compensation from their opted insurance companies.

The insurance sector depends heavily on databases, and in the current environment, an insurance company's performance is related to how well it has optimized its database and how soon are they able to process a claim and so, efficient data management is crucial.

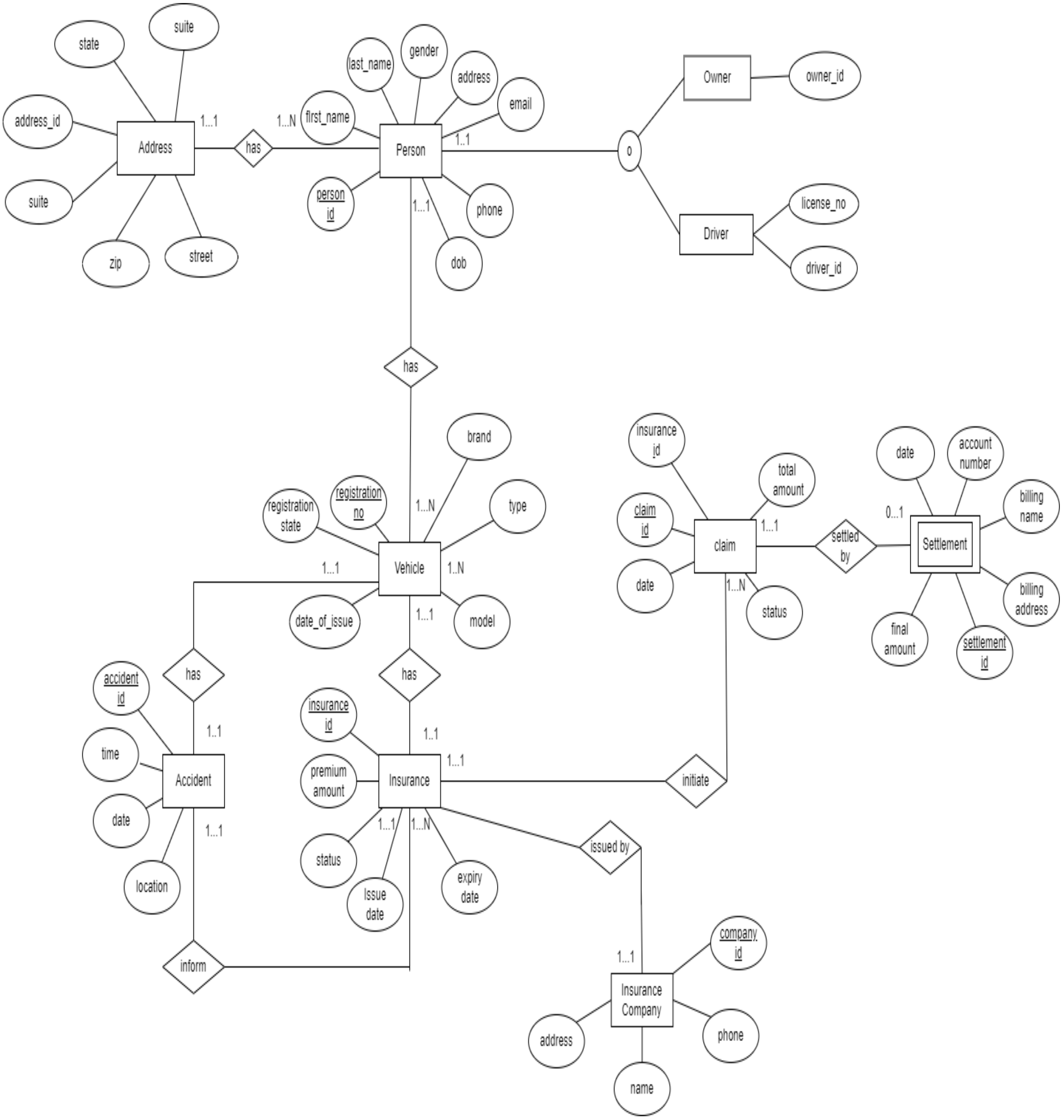
- **The problem:**
  We plan to develop a vehicle insurance policy management system. The platform will connect the Owner details, vehicle's details, Driver attributes, damage, Policy details, claim amount and settlement.

- **The goal of your study:**
  The platform will be a one-stop solution for customers, making the insurance claiming and tracking process a hassle-free one.

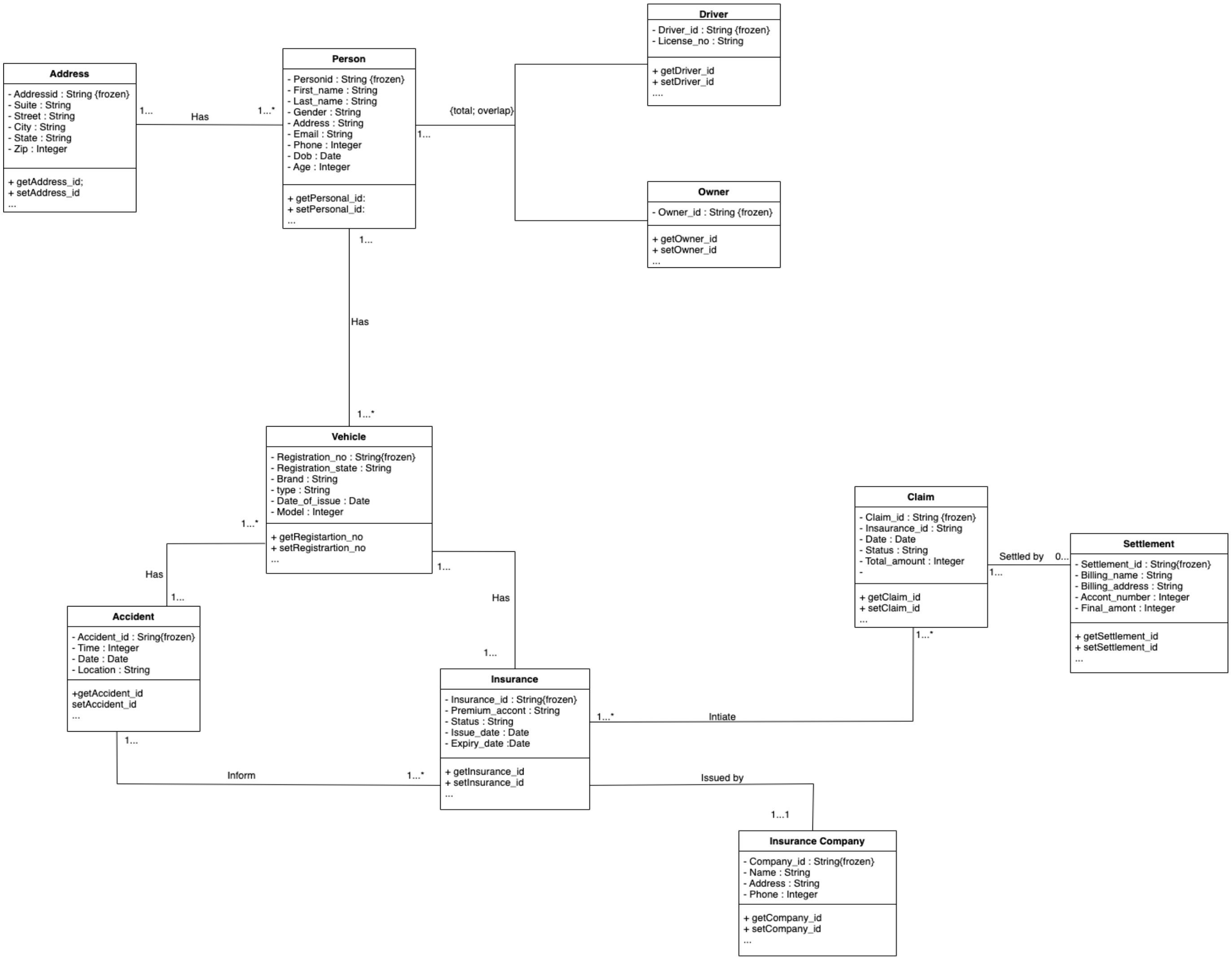
- **The requirements:**
  MySQL, MySQL Workbench, MongoDB, MongoDB Compass, Python and Jupyter Notebook.

II. Conceptual Data Modeling

EER:



UML:



While implementing this database, following constrains must be followed:

- A person is identified by a PersonID. They will be required to fill in their personal details such as first name, last name, gender, email, phone and dob.
- A person will be specialized into a Driver or an Owner having total and overlapping participation
- An owner can own multiple vehicles.
- Each vehicle is identified by a registration_no, state, brand, type and date_of_issue of registration number.
- One vehicle can have one driver only
- Every time an accident happens an accident_ID and date, time and location is also updated
- One vehicle can meet with one accident but there can be more than one vehicle involved in an accident.
- An insurance is informed of the accident as soon as it happens
- One Insurance company can have many policies.
- One vehicle can be covered with one Insurance policy

- One policy can generate one settlement for the claimed amount.
- One person can claim one policy.
- A settlement is completed as soon as the amount reaches the customer's account. A person can get 0 to many settlements but a settlement can be awarded to one person only.

To design database for this system, we have identified the following entities:

- Person
- Driver
- Owner
- Vehicle
- Accident
- Insurance
- Insurance Company
- Claim
- Settlement

III. Mapping Conceptual Model to Relational Model

Relational Model:
(Primary keys underlined; foreign keys in italics): [Normalized]

Person (person_ID, first_name, last_name, gender, dob, email, phone, address_ID)
address_ID: foreign key, refers to address_ID in Address, NULL value not allowed

Address (address_ID, suite, street, city, state, zip)

Driver (driver_ID, license_no)
driver_ID: foreign key, refers to person_ID in Person, NULL value not allowed

Owner (owner_ID)
owner_ID: foreign key, refers to person_ID in Person, NULL value not allowed

Vehicle (registration_no, state, brand, model, date_of_issue, person_id, accident_id)
person_ID: foreign key, refers to person_ID in Person, NULL value not allowed
accident_ID: foreign key, refers to accident_ID in Accident, NULL value not allowed

Accident (accident_ID, date, time, city, state, insurance _ID)
insurance_ ID: foreign key, refers to insurance_ID in Insurance, NULL value allowed

Insurance (insurance_ID, premium_amount, issue_date, expiration_date, vehicle_reg_no, insurance_comp_ID)
vehicle_reg_ID: foreign key, refers to registration_no in Vehicle, NULL value not allowed
insurance_comp_ID: foreign key, refers to company_ID in Insurance Company, NULL value not allowed

Insurance Company (company_ID, name, city, state, phone)

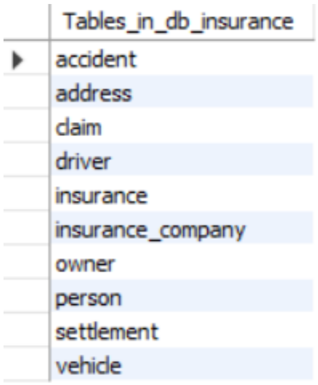
Claim (claim_ID, date, total_amount, settlement_ID, insurance_ID)
settlement_ID: foreign key, refers to settlement_ID in Settlement, NULL value allowed
insurance _ID: foreign key, refers to insurance_ID in Insurance, NULL value not allowed

Settlement (settlement_ID, date, final_amount, account_no)
Describe mapping of the EER and UML to a relation model, including tables, primary keys, foreign keys, relational constraints, and normalization

IV. Implementation of Relation Model via MySQL and NoSQL

Implementation of the relational model in MySQL:

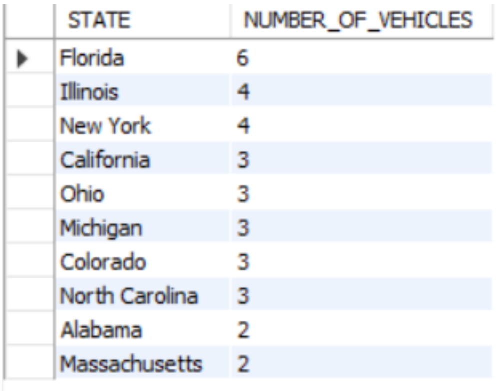
SHOW TABLES; (to see tables in our database)



Tables_in_db_insurance
accident
address
claim
driver
insurance
insurance_company
owner
person
settlement
vehicle

.Find top 10 states with maximum no. of registered car.

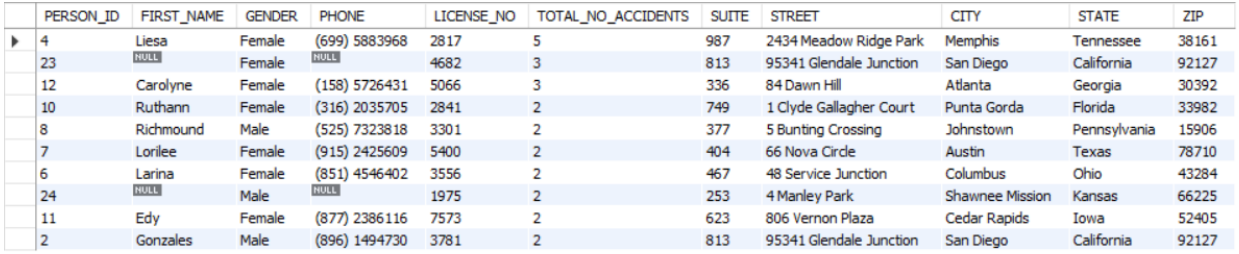
```
SELECT STATE, COUNT(REGISTRATION_NO) AS NUMBER_OF_VEHICLES
FROM VEHICLE
GROUP BY STATE
ORDER BY COUNT(REGISTRATION_NO) DESC
LIMIT 10;
```



STATE	NUMBER_OF_VEHICLES
Florida	6
Illinois	4
New York	4
California	3
Ohio	3
Michigan	3
Colorado	3
North Carolina	3
Alabama	2
Massachusetts	2

2.Find top 10 people with maximum no. of accidents along with their address

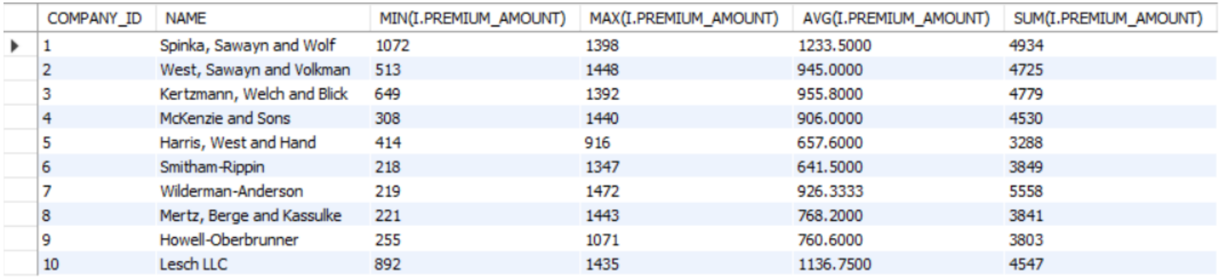
```
SELECT P.PERSON_ID, P.FIRST_NAME, P.GENDER, P.PHONE, D.LICENSE_NO,
COUNT(V.ACCIDENT_ID) AS TOTAL_NO_ACCIDENTS, A.SUITE, A.STREET,
A.CITY,A.STATE, A.ZIP
FROM VEHICLE V, DRIVER D, PERSON P, ADDRESS A
WHERE P.PERSON_ID = V.PERSON_ID
AND V.PERSON_ID = D.DRIVER_ID
AND P.ADDRESS_ID = A.ADDRESS_ID
GROUP BY P.PERSON_ID
ORDER BY COUNT(V.ACCIDENT_ID) DESC
LIMIT 10;
```



PERSON_ID	FIRST_NAME	GENDER	PHONE	LICENSE_NO	TOTAL_NO_ACCIDENTS	SUITE	STREET	CITY	STATE	ZIP
4	Liesa	Female	(699) 5883968	2817	5	987	2434 Meadow Ridge Park	Memphis	Tennessee	38161
23	NULL	Female	NULL	4682	3	813	95341 Glendale Junction	San Diego	California	92127
12	Carolyne	Female	(158) 5726431	5066	3	336	84 Dawn Hill	Atlanta	Georgia	30392
10	Ruthann	Female	(316) 2035705	2841	2	749	1 Clyde Gallagher Court	Punta Gorda	Florida	33982
8	Richmound	Male	(525) 7323818	3301	2	377	5 Bunting Crossing	Johnstown	Pennsylvania	15906
7	Lorilee	Female	(915) 2425609	5400	2	404	66 Nova Circle	Austin	Texas	78710
6	Larina	Female	(851) 4546402	3556	2	467	48 Service Junction	Columbus	Ohio	43284
24	NULL	Male	NULL	1975	2	253	4 Manley Park	Shawnee Mission	Kansas	66225
11	Edy	Female	(877) 2386116	7573	2	623	806 Vernon Plaza	Cedar Rapids	Iowa	52405
2	Gonzales	Male	(896) 1494730	3781	2	813	95341 Glendale Junction	San Diego	California	92127

3. Retrieve the min, max, average and total premium amount company wise.

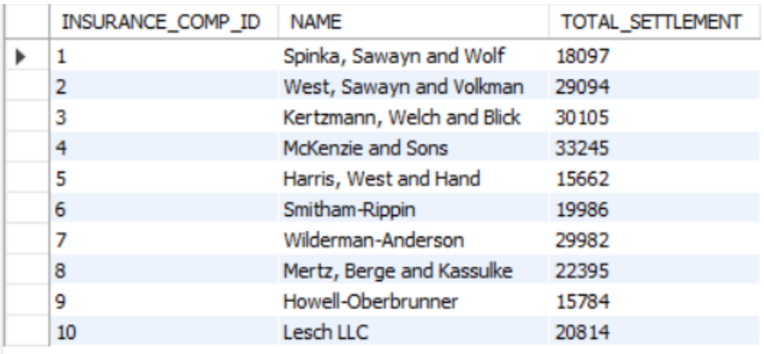
```
SELECT IC.COMPANY_ID, IC.NAME, MIN(I.PREMIUM_AMOUNT), MAX(I.PREMIUM_AMOUNT),
AVG(I.PREMIUM_AMOUNT), SUM(I.PREMIUM_AMOUNT)
FROM INSURANCE I, INSURANCE_COMPANY IC
WHERE I.INSURANCE_COMP_ID = IC.COMPANY_ID
GROUP BY IC.COMPANY_ID;
```



COMPANY_ID	NAME	MIN(I.PREMIUM_AMOUNT)	MAX(I.PREMIUM_AMOUNT)	AVG(I.PREMIUM_AMOUNT)	SUM(I.PREMIUM_AMOUNT)
1	Spinka, Sawayn and Wolf	1072	1398	1233.5000	4934
2	West, Sawayn and Volkman	513	1448	945.0000	4725
3	Kertzmann, Welch and Blick	649	1392	955.8000	4779
4	McKenzie and Sons	308	1440	906.0000	4530
5	Harris, West and Hand	414	916	657.6000	3288
6	Smitham-Rippin	218	1347	641.5000	3849
7	Wilderman-Anderson	219	1472	926.3333	5558
8	Mertz, Berge and Kassulke	221	1443	768.2000	3841
9	Howell-Oberbrunner	255	1071	760.6000	3803
10	Lesch LLC	892	1435	1136.7500	4547

4. Retrieve total settled amount company wise.

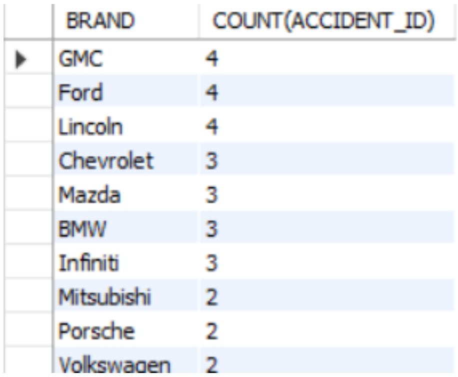
```
SELECT I.INSURANCE_COMP_ID, IC.NAME, SUM(S.FINAL_AMOUNT) AS TOTAL_SETTLEMENT
FROM INSURANCE I, SETTLEMENT S, INSURANCE_COMPANY IC, CLAIM C
WHERE I.INSURANCE_COMP_ID = IC.COMPANY_ID
AND S.SETTLEMENT_ID = C.SETTLEMENT_ID
AND C.INSURANCE_ID = I.INSURANCE_ID
GROUP BY I.INSURANCE_COMP_ID
LIMIT 10;
```



INSURANCE_COMP_ID	NAME	TOTAL_SETTLEMENT
1	Spinka, Sawayn and Wolf	18097
2	West, Sawayn and Volkman	29094
3	Kertzmann, Welch and Blick	30105
4	McKenzie and Sons	33245
5	Harris, West and Hand	15662
6	Smitham-Rippin	19986
7	Wilderman-Anderson	29982
8	Mertz, Berge and Kassulke	22395
9	Howell-Oberbrunner	15784
10	Lesch LLC	20814

5. Find car brand that are least reliable

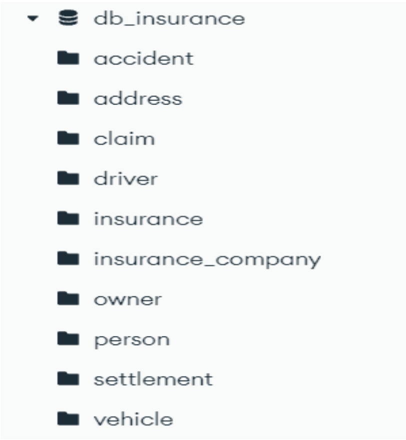
```
SELECT BRAND, COUNT(ACCIDENT_ID)
FROM VEHICLE
GROUP BY BRAND
ORDER BY COUNT(ACCIDENT_ID) DESC
LIMIT 10;
```



BRAND	COUNT(ACCIDENT_ID)
GMC	4
Ford	4
Lincoln	4
Chevrolet	3
Mazda	3
BMW	3
Infiniti	3
Mitsubishi	2
Porsche	2
Volkswagen	2

Implementation of the relational model in NoSQL (MongoDB):

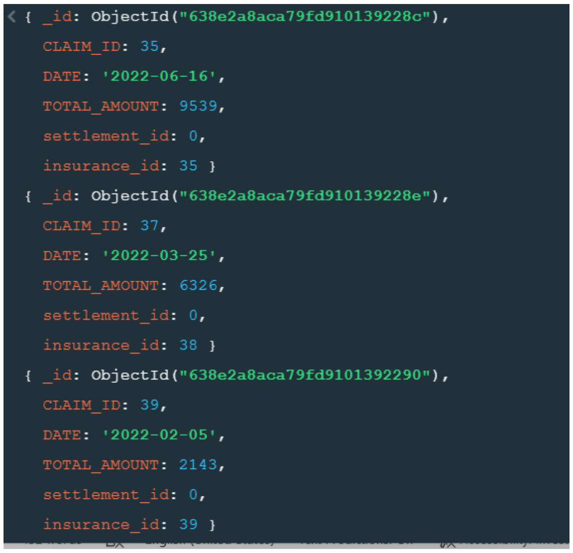
show collections



db_insurance
accident
address
claim
driver
insurance
insurance_company
owner
person
settlement
vehicle

1. Find Claims id which have no settlement.

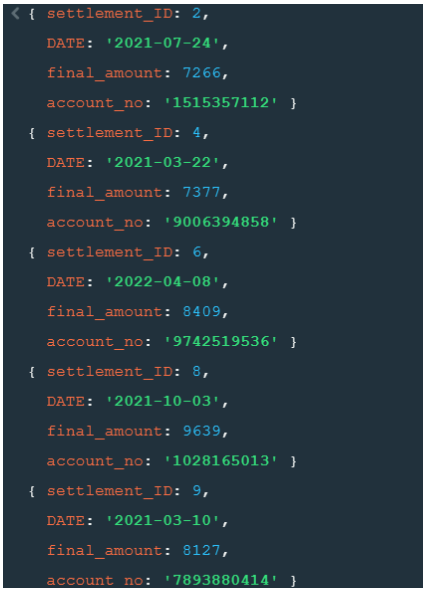
```
db.claim.find({settlement_id : 0})
```



```
< { _id: ObjectId("638e2a8aca79fd910139228c"),
    CLAIM_ID: 35,
    DATE: '2022-06-16',
    TOTAL_AMOUNT: 9539,
    settlement_id: 0,
    insurance_id: 35 }
  { _id: ObjectId("638e2a8aca79fd910139228e"),
    CLAIM_ID: 37,
    DATE: '2022-03-25',
    TOTAL_AMOUNT: 6326,
    settlement_id: 0,
    insurance_id: 38 }
  { _id: ObjectId("638e2a8aca79fd9101392290"),
    CLAIM_ID: 39,
    DATE: '2022-02-05',
    TOTAL_AMOUNT: 2143,
    settlement_id: 0,
    insurance_id: 39 }
```

2. Details of settlement greater than 5000 USD.

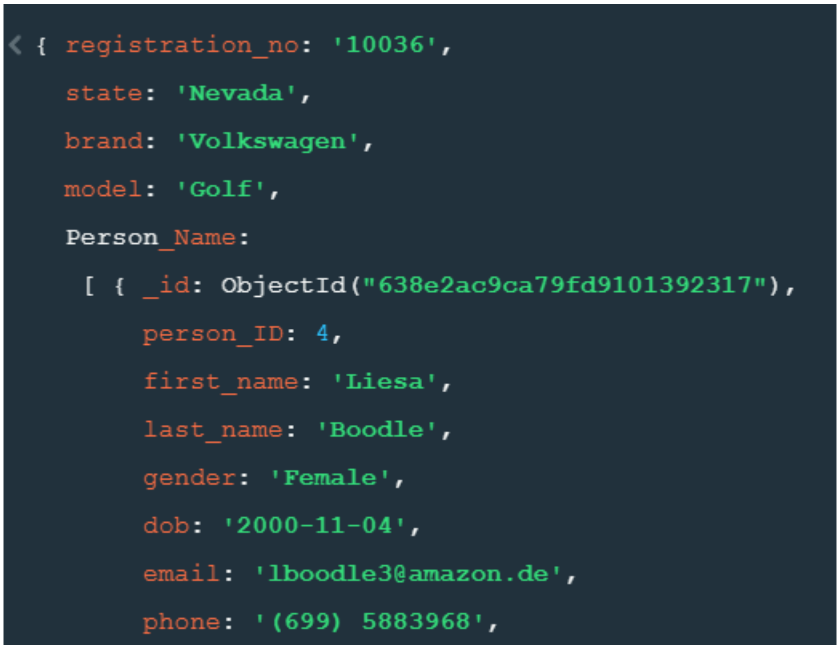
```
db.settlement.aggregate([
  {
   $match: {
     final_amount: {
       $gte: 5000,
     },
   },
 },
 {
   $project: {
     _id: 0,
   },
 }, {$limit : 10}
]
)
```



```
< { settlement_ID: 2,
    DATE: '2021-07-24',
    final_amount: 7266,
    account_no: '1515357112' }
  { settlement_ID: 4,
    DATE: '2021-03-22',
    final_amount: 7377,
    account_no: '9006394858' }
  { settlement_ID: 6,
    DATE: '2022-04-08',
    final_amount: 8409,
    account_no: '9742519536' }
  { settlement_ID: 8,
    DATE: '2021-10-03',
    final_amount: 9639,
    account_no: '1028165013' }
  { settlement_ID: 9,
    DATE: '2021-03-10',
    final_amount: 8127,
    account_no: '7893880414' }
```

3. Details of people from vehicle registration no 10036.

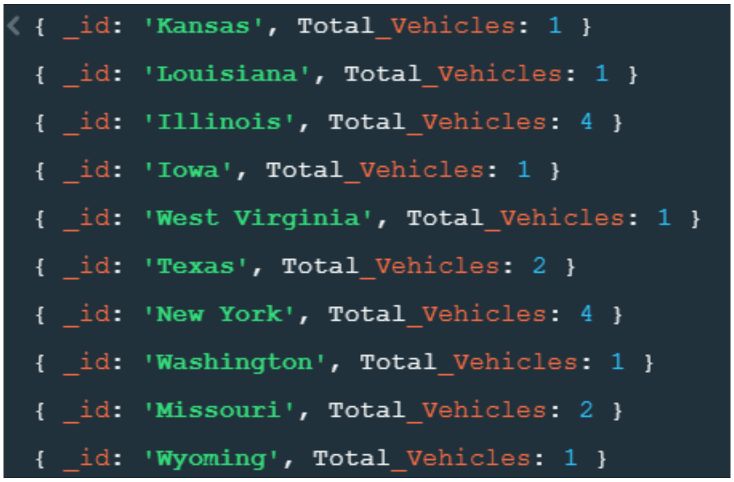
```
db.vehicle.aggregate([{
    $match: {
      registration_no: "10036",
    },
  },
  {
    $lookup: {
      from: "person",
      localField: "person_id",
            foreignField: "person_ID",
            as: "Person_Name",
        },
    },
    {
        $project: {
            _id: 0,
            registration_no: 1,
            Person_Name: 1,
            state: 1,
            brand: 1,
            model: 1,
        },
    },
]
)
```



```
< { registration_no: '10036',
    state: 'Nevada',
    brand: 'Volkswagen',
    model: 'Golf',
    Person_Name:
     [ { _id: ObjectId("638e2ac9ca79fd9101392317"),
         person_ID: 4,
         first_name: 'Liesa',
         last_name: 'Boodle',
         gender: 'Female',
         dob: '2000-11-04',
         email: 'lboodle3@amazon.de',
         phone: '(699) 5883968',
```

4. Find no of vehicles registered in each state.

```
db.vehicle.aggregate([
  {
    $group: {
      _id: "$state",
      Total_Vehicles: {
        $count: {},
      },
    },
  },
]
)
```



```
< { _id: 'Kansas', Total_Vehicles: 1 }
  { _id: 'Louisiana', Total_Vehicles: 1 }
  { _id: 'Illinois', Total_Vehicles: 4 }
  { _id: 'Iowa', Total_Vehicles: 1 }
  { _id: 'West Virginia', Total_Vehicles: 1 }
  { _id: 'Texas', Total_Vehicles: 2 }
  { _id: 'New York', Total_Vehicles: 4 }
  { _id: 'Washington', Total_Vehicles: 1 }
  { _id: 'Missouri', Total_Vehicles: 2 }
  { _id: 'Wyoming', Total_Vehicles: 1 }
```

V. Database Access via R or Python

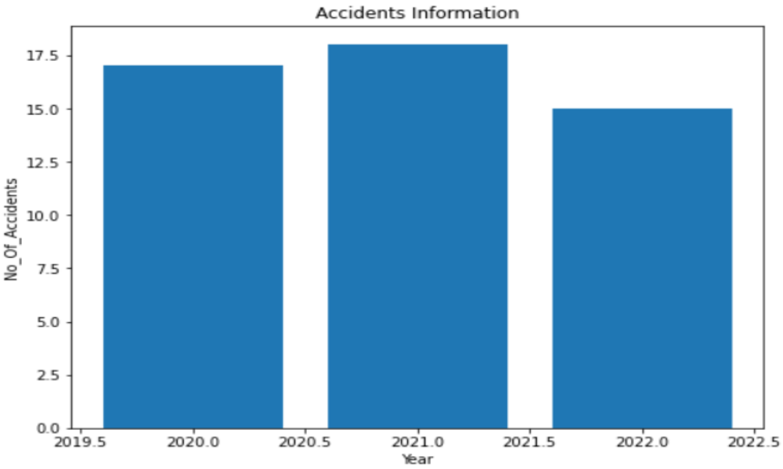
Code for connecting MySQL to Python

```
pip install mysql-connector-python
import mysql.connector as connection
import pandas as pd
import mysql.connector
import matplotlib.pyplot as plt
mydb = connection.connect(host="localhost", database = 'db_insurance',user="root",
passwd="root",use_pure=True)
```

Plot 1: Plotting bar graph to show the accidents that occurred based on year.

Code for creating bar plot

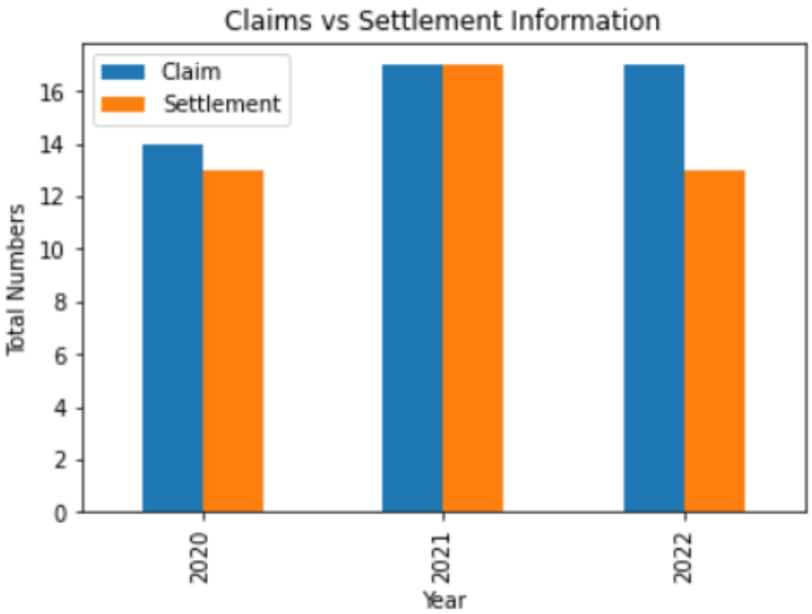
```
query = "select year(date),count(accident_id) from accident group by year(date);"
result_df = pd.read_sql(query,mydb)
result_df.set_index("year(date)", inplace = True)
plt.figure(figsize=(8, 6))
result_df.plot(kind="bar")
plt.xlabel("Year")
plt.ylabel("No_Of_Accidents")
plt.title("Accidents Information")
plt.show()
```



Plot 2: Plotting grouped bar graph to show the number of claims and settlements that occurred during the year 2019 – 2022.

Code for creating grouped bar plot

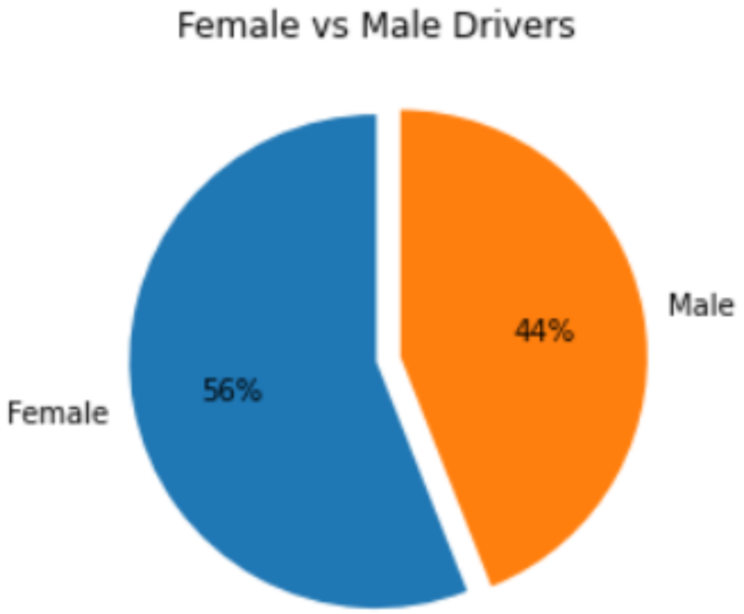
```
query = "select year(c.date) as Year, count(c.claim_id) as Claim, count(s.settlement_id) as
Settlement from claim c left join settlement s on  c.settlement_id = s.settlement_id group by
year(c.date) order by year(c.date);"
result_df = pd.read_sql(query,mydb)
result_df.set_index("Year", inplace = True)
result_df.plot(kind="bar")
plt.xlabel("Year")
plt.ylabel("Total Numbers")
plt.title("Claims vs Settlement Information")
plt.show()
```



Plot 3: Plotting pie - chart to show male vs female drivers' ratio who acquired insurance

Code for creating pie-chart

```
query = "select p.gender, count(person_id) as count from person p, driver d where p.person_id =
d.driver_id group by p.gender;"
result_df = pd.read_sql(query,mydb)
myexplode = (0, 0.1)
sizes = plt.pie(result_df['count'], labels = result_df['gender'], startangle = 90, explode =
myexplode)
plt.title("Female vs Male Drivers")
plt.show()
```



VII. Summary and recommendation

The Vehicle Insurance database management system was designed on MYSQL is an industry ready relational database that can be used by customers to ease their process of insurance claim and settlement. This will result in saving time and help companies target potential customers and provide a better profit margin at ease and provides great analytics capabilities, a sample of which is shown in this report utilizing Python.

The database designed in MYSQL can also be implemented as a website by designing the front end UX wireframes for the database as this will benefit customers to do self-based search of their needs and will reduce the workload from insurance firms.

Further the NoSQL implementations of this database on MongoDB, can certainly be used to build the database and take full advantage of the cloud to deliver zero downtime.

Future Scope:

- Accident data can be used to implement safety alerting systems on roads and support the advancement in road transportation infrastructure.
- At high level this model can be improved to send alerts about when the Insurance policy is about to expire