

**Database of a vehicle insurance company**

Report

CS310

Database Management Systems

Under the guidance of :-

* Dr. Uma S
* Dr.Pramod Yelmewad
* Ms. Supriya Nadiger

Team members **:-**

* Gitesh Ambre(20bcs009)
* Michael Roy(20bcs010)
* Ashis J Kalathil(20bcs022)
* Chaitanya Giri (20bcs034)
* Dipesh Mishra (20bcs043)
* Harshal Ram Dube(20bcs052)
* Hemang Ranga(20bcs057)
* Prathamesh pai(20bcs103)
* Sunil Patidar(20bcs131)

Objective**:** To gain practical knowledge on the working of a DBMS, to gain practical experience in advanced entity modelling, normalisation & transactional relational database design and being able to apply all these principles in order to make a good DB, that could be used with analytical tools and faster in delivering the right data at the right time for better decision making.

Abstract**:** The Insurance management system organizes and tracks insurance vendors and the policies provided under different coverage. Details of payment, time period, vehicle details, customer personal details, insurance specifications are updated to the database.

There is a need for an automated system, which can efficiently manage the company, records, provides instant access and one that improves productivity. As a result of this automated system, the activities of the company are performed within the stipulated time and the reliable and efficient service is ensured to its users.

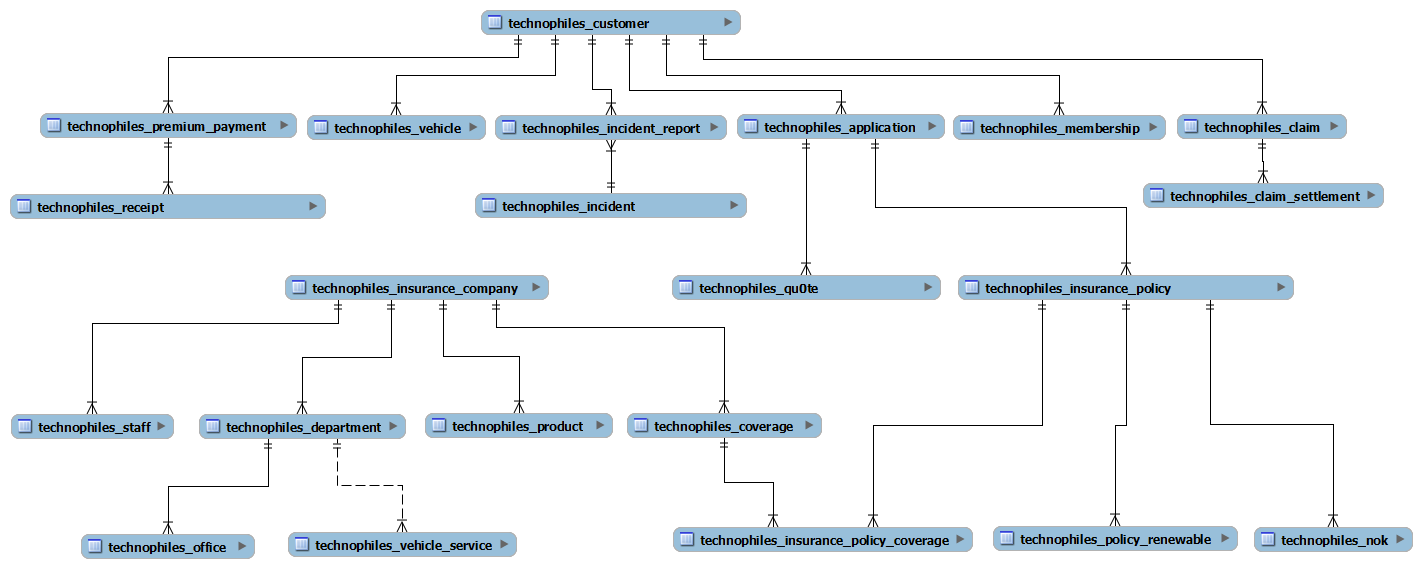
The insurance company needs to keep track of details of its target companies, agents, policyholders, their premium payments and the various products that are available with it. Hence it is under tremendous pressure maintaining their day-to-day activities, which is currently being done manually.

Tech stack used**:**

* MySQL Workbench
* GitHub
* Node JS
* React JS
* Express JS
* HTML 5
* CSS 3

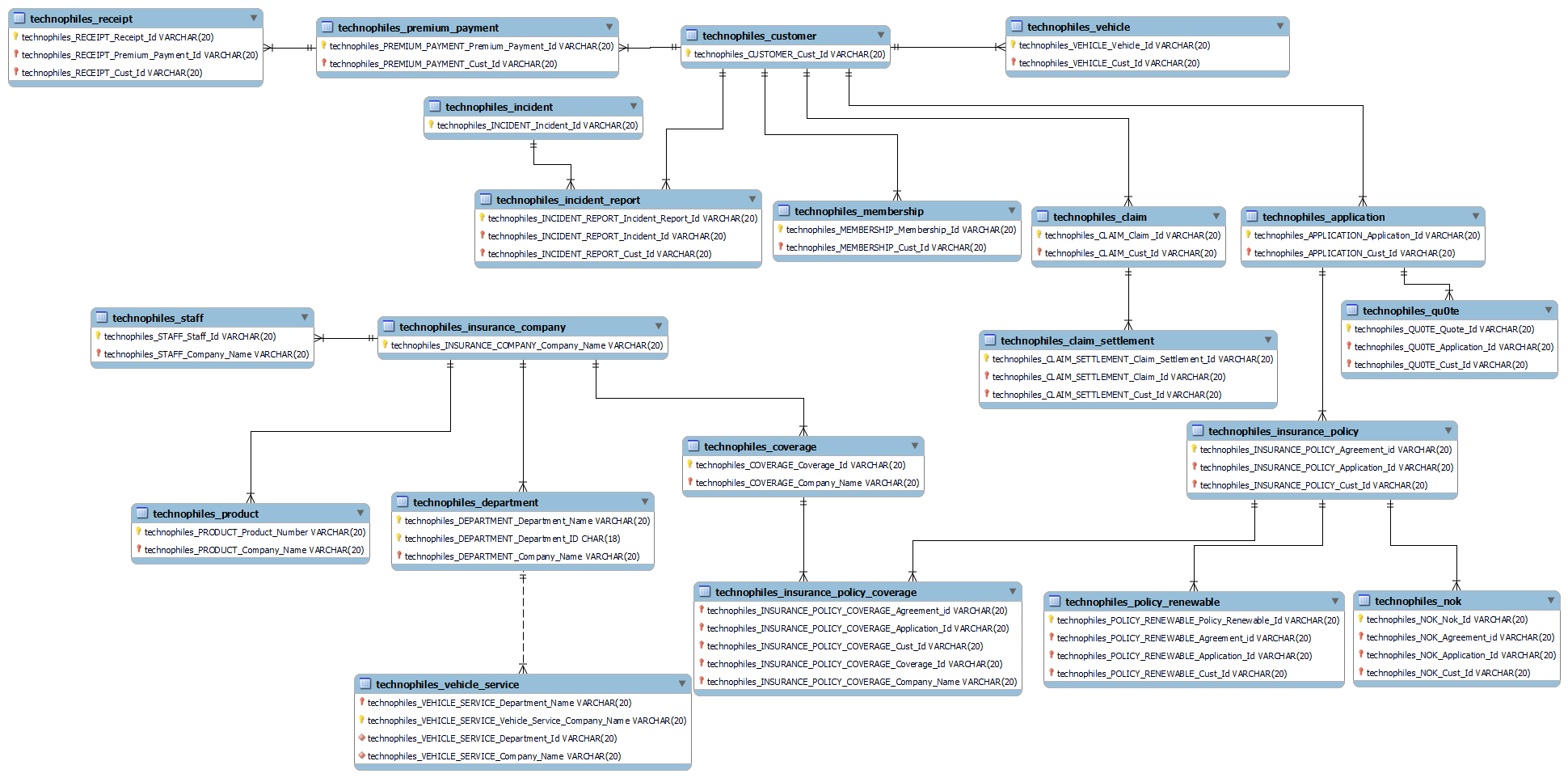
Conceptual database model**:**

The conceptual data model is a structured business view of the data required to support business processes, record business events, and track related performance measures. This model focuses on identifying the data used in the business but not its processing flow or physical characteristics. This model’s perspective is independent of any underlying business applications.



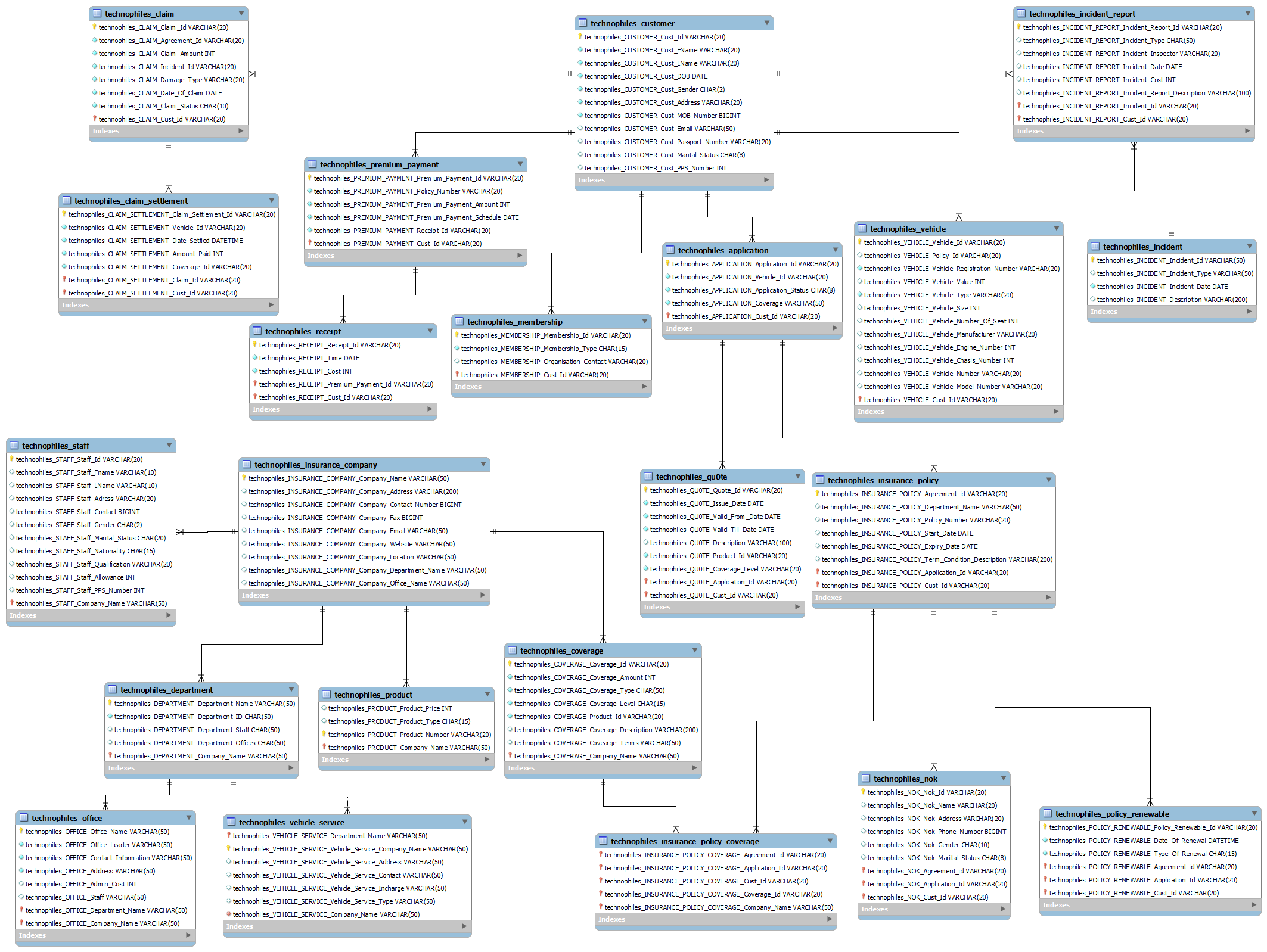
Logical data model**:**

The Logical Data Model is used to define the structure of data elements and to set relationships between them. The logical data model adds further information to the conceptual data model elements. The advantage of using a Logical data model is to provide a foundation to form the base for the Physical model. However, the modelling structure remains generic.



Physical Data Model**:**

A Physical Data Model describes a database-specific implementation of the data model. It offers database abstraction and helps generate the schema. This is because of the richness of meta-data offered by a Physical Data Model. The physical data model also helps in visualizing database structure by replicating database column keys, constraints, indexes, triggers, and other RDBMS features.



Queries**:**

**-- 1. Retrieve Customer and Vehicle details who has been involved in an incident and claim status is pending – Customer, vehicle, claim status, incident**

select

c.technophiles\_CUSTOMER\_Cust\_Id as cust\_id,

concat(

c.technophiles\_CUSTOMER\_Cust\_FName,

" ",

c.technophiles\_CUSTOMER\_Cust\_LName

) as cust\_name,

v.technophiles\_VEHICLE\_Vehicle\_Id as vehicle\_id,

v.technophiles\_VEHICLE\_Vehicle\_Type as vehicle\_type,

v.technophiles\_VEHICLE\_Vehicle\_Value as vehicle\_value,

i.technophiles\_INCIDENT\_REPORT\_Incident\_Report\_Id as incident\_report\_id,

i.technophiles\_INCIDENT\_REPORT\_Incident\_Type as incident\_type,

cl.technophiles\_CLAIM\_Claim\_Status as claim\_status

from

technophiles\_CUSTOMER c

join technophiles\_VEHICLE v on c.technophiles\_CUSTOMER\_Cust\_Id = v.technophiles\_VEHICLE\_Cust\_Id

join technophiles\_INCIDENT\_REPORT i on c.technophiles\_CUSTOMER\_Cust\_Id = i.technophiles\_INCIDENT\_REPORT\_Cust\_Id

join (

select

distinct c.technophiles\_CLAIM\_Cust\_Id,

c.technophiles\_CLAIM\_Claim\_Status

from

technophiles\_CLAIM c

where

c.technophiles\_CLAIM\_Claim\_Status = "Pending"

) as cl on cl.technophiles\_CLAIM\_Cust\_Id = c.technophiles\_CUSTOMER\_Cust\_Id;

**-- 2. Retrieve customer details who has premium payment amount greater than the sum of all the customerIds in the database – premium payment, customer**

select

c.technophiles\_CUSTOMER\_Cust\_Id as cust\_id,

concat(

c.technophiles\_CUSTOMER\_Cust\_FName,

" ",

c.technophiles\_CUSTOMER\_Cust\_LName

) as cust\_name,

c.technophiles\_CUSTOMER\_Cust\_Gender as cust\_gender,

p.technophiles\_PREMIUM\_PAYMENT\_Premium\_Payment\_Id as payment\_id,

p.technophiles\_PREMIUM\_PAYMENT\_Premium\_Payment\_Amount as payment\_amount,

(select

sum(c.technophiles\_CUSTOMER\_Cust\_Id)

from

technophiles\_CUSTOMER c) as sum\_of\_cust\_ids

from

technophiles\_CUSTOMER c

join technophiles\_PREMIUM\_PAYMENT p on p.technophiles\_PREMIUM\_PAYMENT\_Cust\_Id = c.technophiles\_CUSTOMER\_Cust\_Id

where

p.technophiles\_PREMIUM\_PAYMENT\_Premium\_Payment\_Amount > (

select

sum(c.technophiles\_CUSTOMER\_Cust\_Id)

from

technophiles\_CUSTOMER c

);

**-- 3. Retrieve Company details whose number of products is greater than departments, where the departments are located in more than one location—company, product, departments, office**

select

c.technophiles\_INSURANCE\_COMPANY\_Company\_id as company\_id,

c.technophiles\_INSURANCE\_COMPANY\_Company\_Name as company\_Name,

c.technophiles\_INSURANCE\_COMPANY\_Company\_Contact\_Number as "contact\_no.",

c.technophiles\_INSURANCE\_COMPANY\_Company\_Location as company\_location,

count(distinct d.technophiles\_DEPARTMENT\_Department\_id) as num\_of\_departments,

count(p.technophiles\_PRODUCT\_Product\_Number) as num\_of\_products

from

technophiles\_INSURANCE\_COMPANY c

join technophiles\_PRODUCT p on c.technophiles\_INSURANCE\_COMPANY\_Company\_id = p.technophiles\_PRODUCT\_Company\_id

join (

select

d.\*

from

technophiles\_DEPARTMENT d

join technophiles\_OFFICE o on o.technophiles\_OFFICE\_Department\_id = d.technophiles\_DEPARTMENT\_Department\_id

group by

d.technophiles\_DEPARTMENT\_Department\_id

having

count(o.technophiles\_OFFICE\_Office\_Name) > 1

) as d on d.technophiles\_DEPARTMENT\_Company\_id = c.technophiles\_INSURANCE\_COMPANY\_Company\_id

group by

c.technophiles\_INSURANCE\_COMPANY\_Company\_id

having

count(distinct d.technophiles\_DEPARTMENT\_Department\_id) < count(p.technophiles\_PRODUCT\_Product\_Number);

**-- 4. Select Customers who have more than one Vehicle, where the premium for one of the Vehicles is not paid and it is involved in accident**

select

c.technophiles\_CUSTOMER\_Cust\_Id as cust\_id,

concat(c.technophiles\_CUSTOMER\_Cust\_FName, " ", c.technophiles\_CUSTOMER\_Cust\_LName) as cust\_name,

c.technophiles\_CUSTOMER\_Cust\_Gender as cust\_gender,

count(v.technophiles\_VEHICLE\_Vehicle\_Id) as num\_of\_vehicles,

r.technophiles\_INCIDENT\_REPORT\_Incident\_Report\_Id as incident\_report\_id,

r.technophiles\_INCIDENT\_REPORT\_Incident\_Type as incident\_type,

Premium\_Payment\_Amount,

total\_paid

from

technophiles\_CUSTOMER c

join technophiles\_VEHICLE v on c.technophiles\_CUSTOMER\_Cust\_Id = v.technophiles\_VEHICLE\_Cust\_Id

join (

select

technophiles\_PREMIUM\_PAYMENT\_Cust\_Id,

technophiles\_PREMIUM\_PAYMENT\_Premium\_Payment\_Amount,

sum(r.technophiles\_RECEIPT\_Cost) as total\_paid,

technophiles\_PREMIUM\_PAYMENT\_Premium\_Payment\_Amount as Premium\_Payment\_Amount

from

technophiles\_PREMIUM\_PAYMENT p

join technophiles\_RECEIPT r on p.technophiles\_PREMIUM\_PAYMENT\_Premium\_Payment\_Id = r.technophiles\_RECEIPT\_Premium\_Payment\_Id

and p.technophiles\_PREMIUM\_PAYMENT\_Cust\_Id = r.technophiles\_RECEIPT\_Cust\_Id

group by

p.technophiles\_PREMIUM\_PAYMENT\_Cust\_Id,

p.technophiles\_PREMIUM\_PAYMENT\_Premium\_Payment\_Id

having

sum(r.technophiles\_RECEIPT\_Cost) < technophiles\_PREMIUM\_PAYMENT\_Premium\_Payment\_Amount

) as pr on c.technophiles\_CUSTOMER\_Cust\_Id = pr.technophiles\_PREMIUM\_PAYMENT\_Cust\_Id

join technophiles\_INCIDENT\_REPORT r on c.technophiles\_CUSTOMER\_Cust\_Id = r.technophiles\_INCIDENT\_REPORT\_Cust\_Id

group by

v.technophiles\_VEHICLE\_Cust\_Id

having

count(v.technophiles\_VEHICLE\_Vehicle\_Id) > 1;

**-- 5. Select all vehicles which have premium more than its vehicle number.**

select

c.technophiles\_CUSTOMER\_Cust\_Id as cust\_id,

concat(

c.technophiles\_CUSTOMER\_Cust\_FName,

" ",

c.technophiles\_CUSTOMER\_Cust\_LName

) as cust\_name,

-- c.technophiles\_CUSTOMER\_Cust\_Gender as cust\_gender,

v.technophiles\_VEHICLE\_Vehicle\_Id as vehicle\_id,

-- v.technophiles\_VEHICLE\_Vehicle\_Manufacturer as vehicle\_manufacturer,

v.technophiles\_VEHICLE\_Vehicle\_Type as vehicle\_type,

v.technophiles\_VEHICLE\_Vehicle\_Number as vehicle\_number,

p.technophiles\_PREMIUM\_PAYMENT\_Premium\_Payment\_Amount as premium\_amount

from

technophiles\_CUSTOMER c

join technophiles\_VEHICLE v on c.technophiles\_CUSTOMER\_Cust\_Id = v.technophiles\_VEHICLE\_Cust\_Id

join technophiles\_PREMIUM\_PAYMENT p on p.technophiles\_PREMIUM\_PAYMENT\_Cust\_Id = c.technophiles\_CUSTOMER\_Cust\_Id

where

v.technophiles\_VEHICLE\_Vehicle\_Number <= p.technophiles\_PREMIUM\_PAYMENT\_Premium\_Payment\_Amount;

**-- 6. Retrieve Customer details whose Claim Amount is less than Coverage Amount and Claim Amount is greater than Sum of (CLAIM\_SETTLEMENT\_ID, VEHICLE\_ID, CLAIM\_ID, CUST\_ID)**

select

c.technophiles\_CUSTOMER\_Cust\_Id as cust\_id,

concat(c.technophiles\_CUSTOMER\_Cust\_FName, " ", c.technophiles\_CUSTOMER\_Cust\_LName) as cust\_name,

c.technophiles\_CUSTOMER\_Cust\_Gender as cust\_gender,

cv.technophiles\_COVERAGE\_Coverage\_Amount as coverage\_amount,

cl.technophiles\_CLAIM\_Claim\_Amount as claim\_amount,

(

cs.technophiles\_CLAIM\_SETTLEMENT\_Claim\_Settlement\_Id + cs.technophiles\_CLAIM\_SETTLEMENT\_Vehicle\_Id + cs.technophiles\_CLAIM\_SETTLEMENT\_Claim\_Id + c.technophiles\_CUSTOMER\_Cust\_Id

) as sum\_of\_ids

from

technophiles\_CUSTOMER c

join technophiles\_CLAIM cl on c.technophiles\_CUSTOMER\_Cust\_Id = cl.technophiles\_CLAIM\_Cust\_Id

join technophiles\_CLAIM\_SETTLEMENT cs on cl.technophiles\_CLAIM\_Claim\_Id = cs.technophiles\_CLAIM\_SETTLEMENT\_Claim\_Id

join technophiles\_COVERAGE cv on cs.technophiles\_CLAIM\_SETTLEMENT\_Coverage\_Id = cv.technophiles\_COVERAGE\_Coverage\_Id

where

cl.technophiles\_CLAIM\_Claim\_Amount < cv.technophiles\_COVERAGE\_Coverage\_Amount

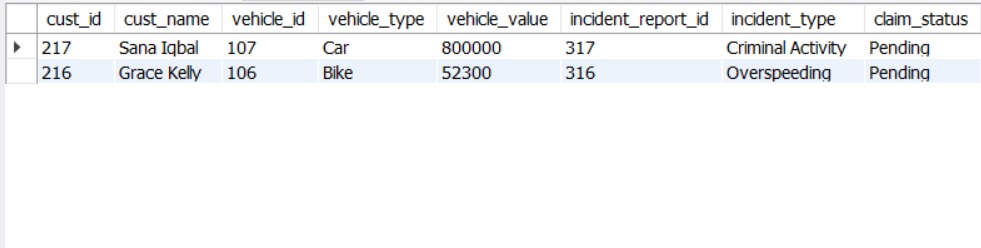
and cl.technophiles\_CLAIM\_Claim\_Amount > (

cs.technophiles\_CLAIM\_SETTLEMENT\_Claim\_Settlement\_Id + cs.technophiles\_CLAIM\_SETTLEMENT\_Vehicle\_Id + cs.technophiles\_CLAIM\_SETTLEMENT\_Claim\_Id + c.technophiles\_CUSTOMER\_Cust\_Id

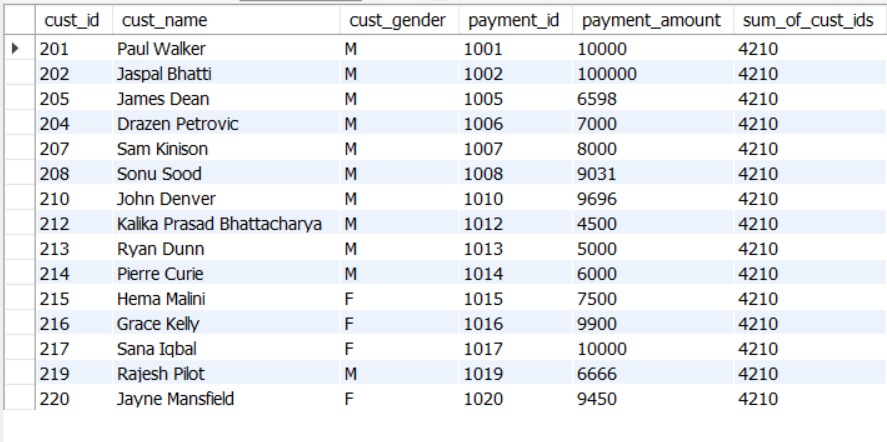
);

Conclusion **:-**

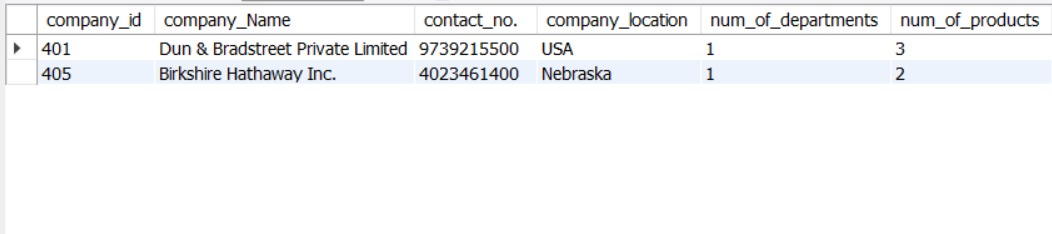
**Query 1:-**



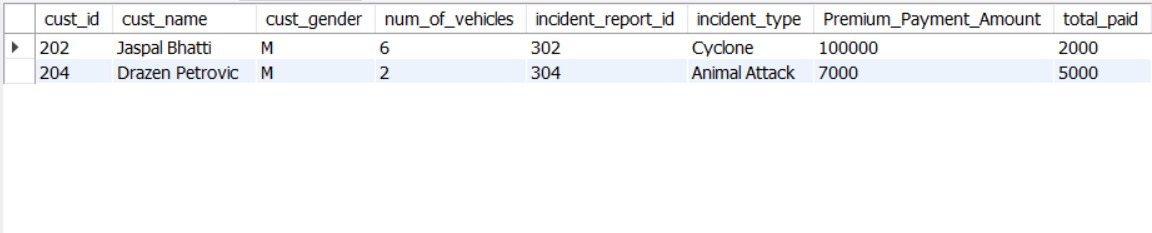
**Query 2:-**

****

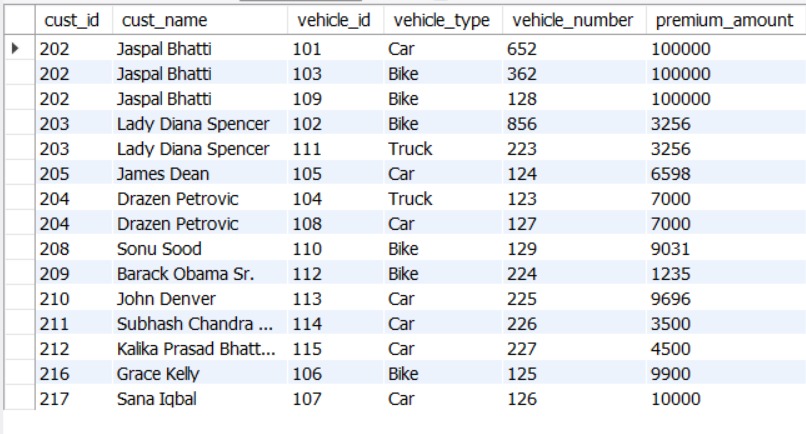
**Query 3:-**



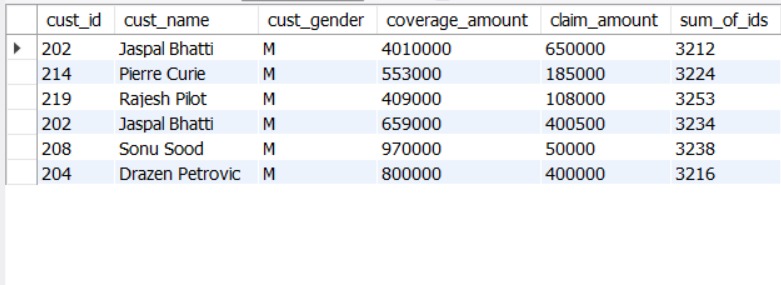
**Query 4:-**

****

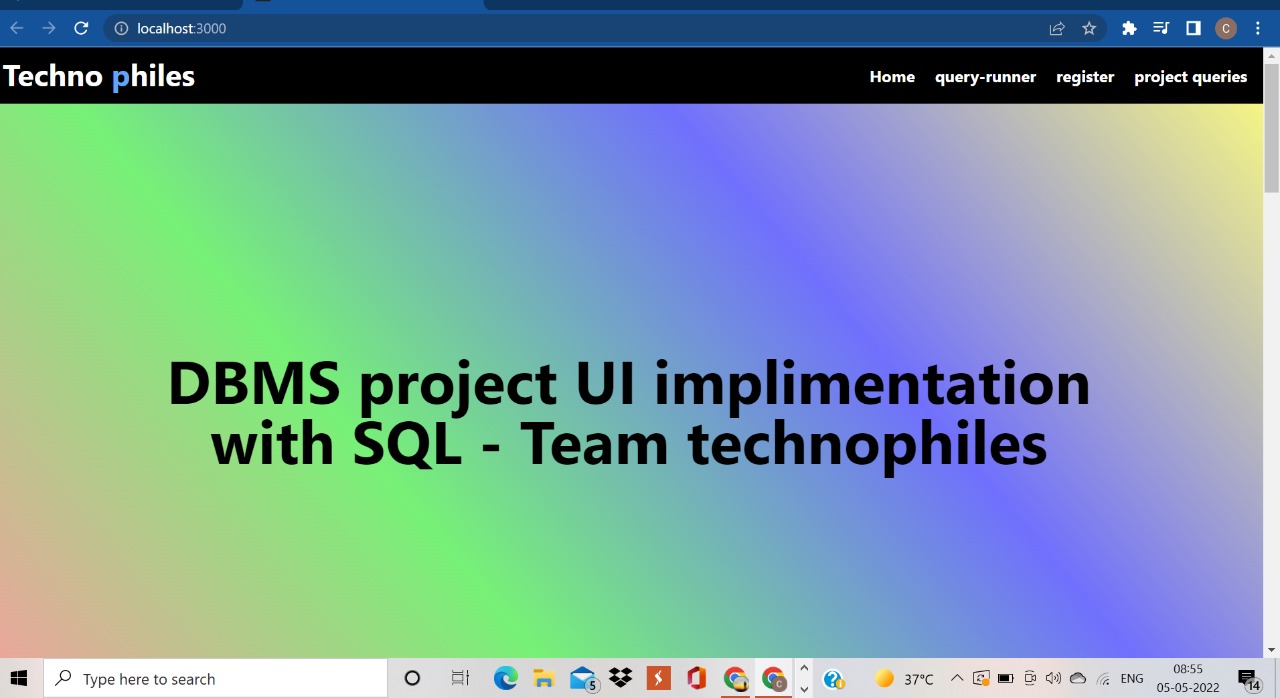
**Query 5:-**

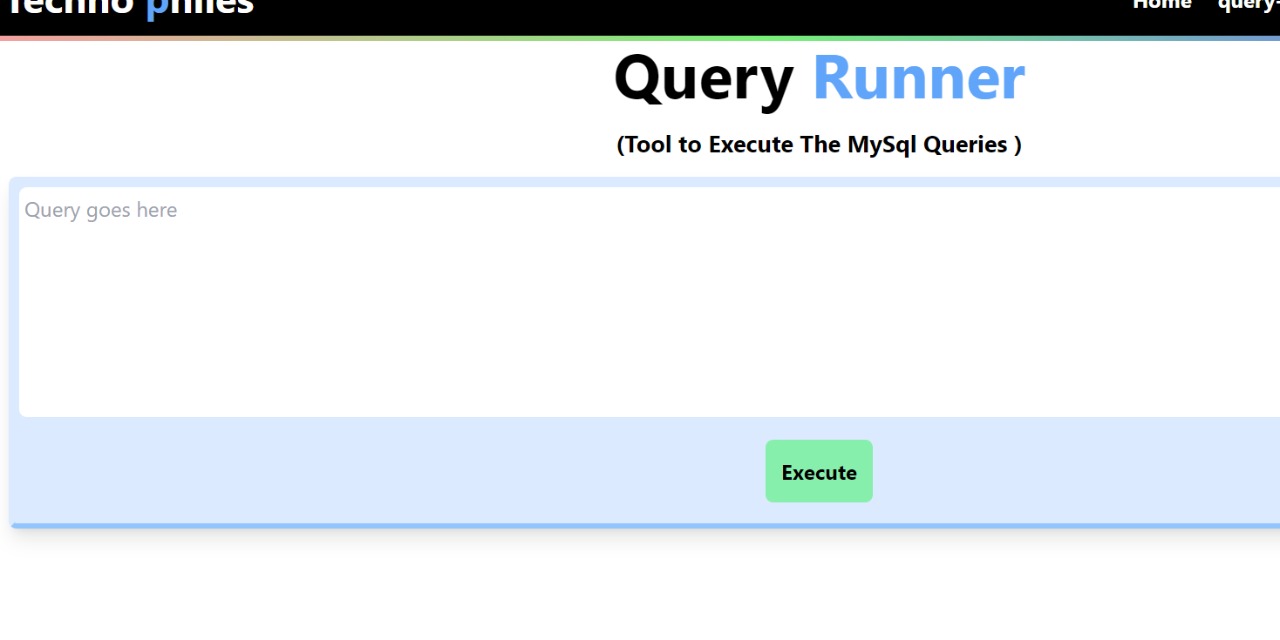


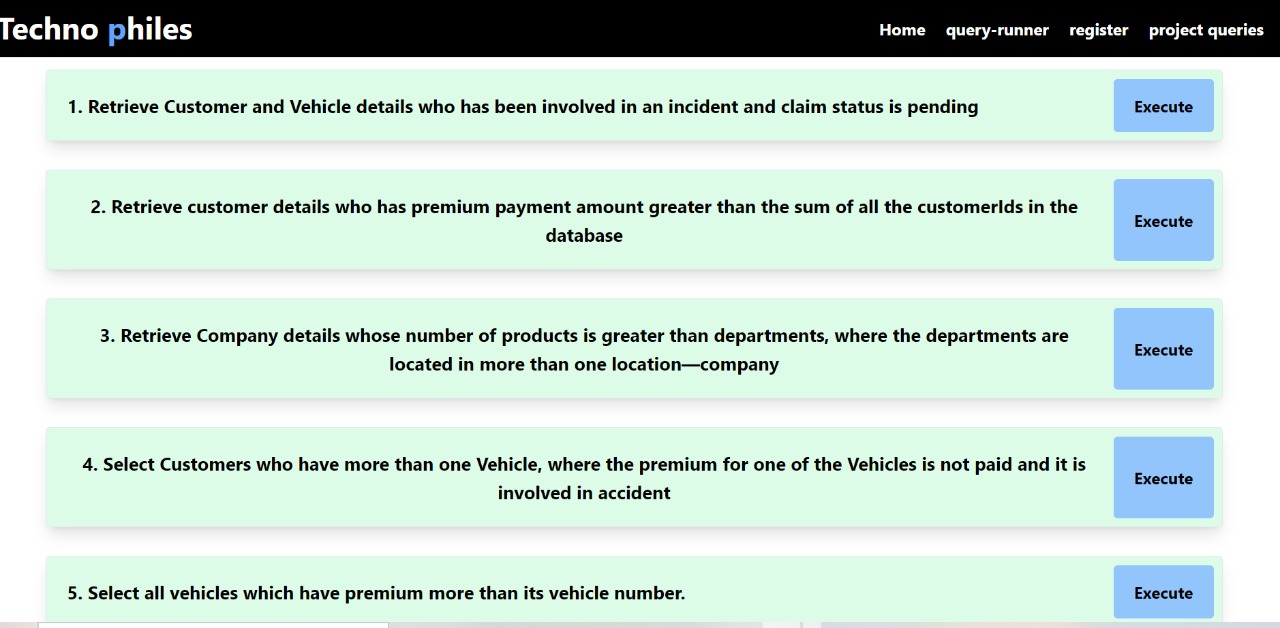
**Query 6:-**

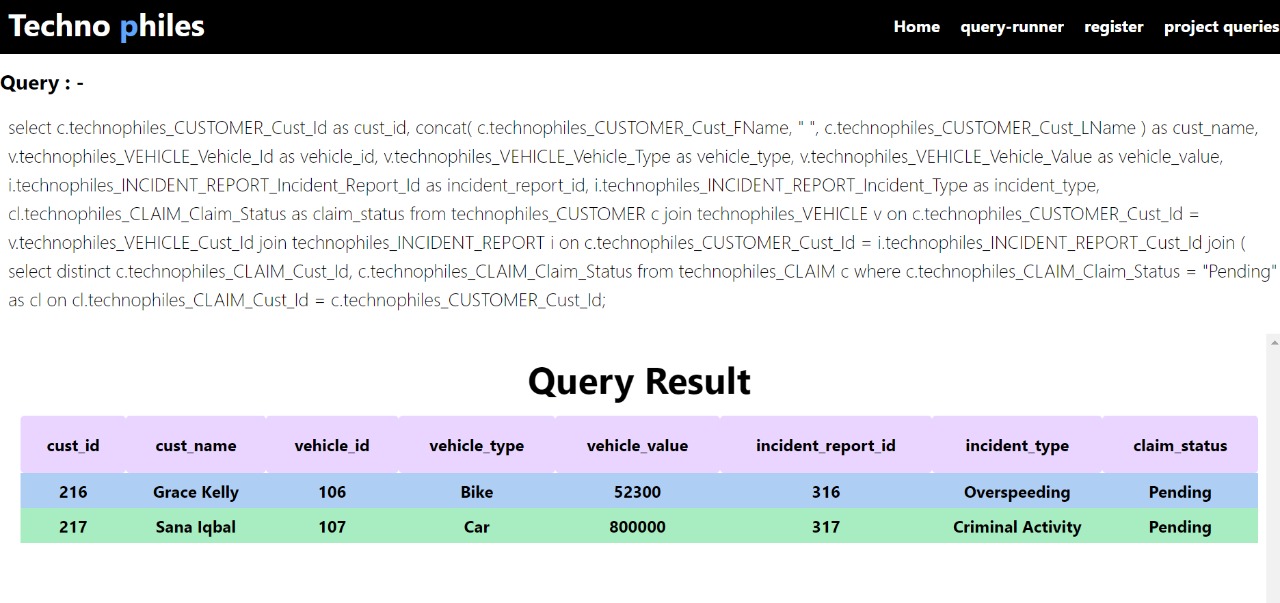
****

**Website :-**







****