भारतीय सूचना प्रौद्योगिकी संस्थान धारवाड़

Indian Institute of Information Technology Dharwad

DATABASE MANAGEMENT SYSTEMS

(COURSE CODE: CS310)

Report On

DATABASE DESIGN ON CAR VEHICLE INSURANCE COMPANY

Submitted by

Team – 5(DATAcated)

Under the Guidance of:

Dr. Uma Sheshadri(Professor, Dept. of CSE, IIIT DHARWAD)

Dr. Pramod Yelmewad(Assistant Professor, Dept. of CSE, IIIT DHARWAD)

Ms. Supriya Nadiger(Lab Assistant, IIIT DHARWAD)

Index…

1. About the Project
    a. Project Title
    b. Aim
    c. Purpose
    d. Scope
    e. Project Benefits
    f. Information about MySQL
    g. Software Requirements

2. Team & Major Roles Played

3. Database Modelling
    a. Conceptual Data Modelling (CDM)
    b. Logical Data Modelling (LDM)
    c. Physical Data Modelling (PDM)

4. Stored Procedures and Functions

5. Project Queries (1 to 6)

6. Conclusion

1.) About the Project

Project title:

Database design for a Vehicle Insurance Company.

Aim:

To create and Maintain a VehicleInsurance Database implemented in MySQL Database System and retrieve information from it and provide a comprehensive understanding and practical experience in data Modelling, use of normalization techniques, transactional relational database design, and implementation of SQL queries.

Purpose:

The purpose of this project is to acquire a good amount of knowledge as well as practical experience in Advanced Entity Modelling, Normalisation, Relational Database Design, and acquiring good knowledge in SQL.

Scope:

The scope of this project is compacted to a Database Administrator A Data Analyst or A Software Engineer, who is familiar with the concepts of Database Management Systems (DBMS), and who can write and understand SQL queries for retrieving information from the database. In this project, all of my teammates implemented the entire database in MySQL Database. To proceed with the project, one needs to be familiar with MySQL Workbench and MySQL Server.

Project Benefits:

When a developer builds an application or software, that Software needs Data to perform day-to-day operations and analytics over-processed data which is something driving the business nowadays to excel in their respective areas of operations. So as a developer, we need a DataBase Management System where we can create, update, delete, administer, and analyze the data. We developed a good database, that could be used with analytical tools and faster in delivering the accurate information at accurate times for better decision making and understanding.

Information about MySQL:

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the prominent RDBMS being used for developing various web-based software applications. MySQL is developed and supported by MySQL AB, which is a Swedish company.SQL is a computer language supported by several database software programs. It makes accessing database data for other programs easy. Programs that need database software for handling low-level tasks of managing information would simply use SQL to transmit instructions.

Software Requirements:

- A Windows/ Linux/ Mac system.
- Properly installed MySQL server & MySQL Workbench.

2.) Team and Major Roles Played

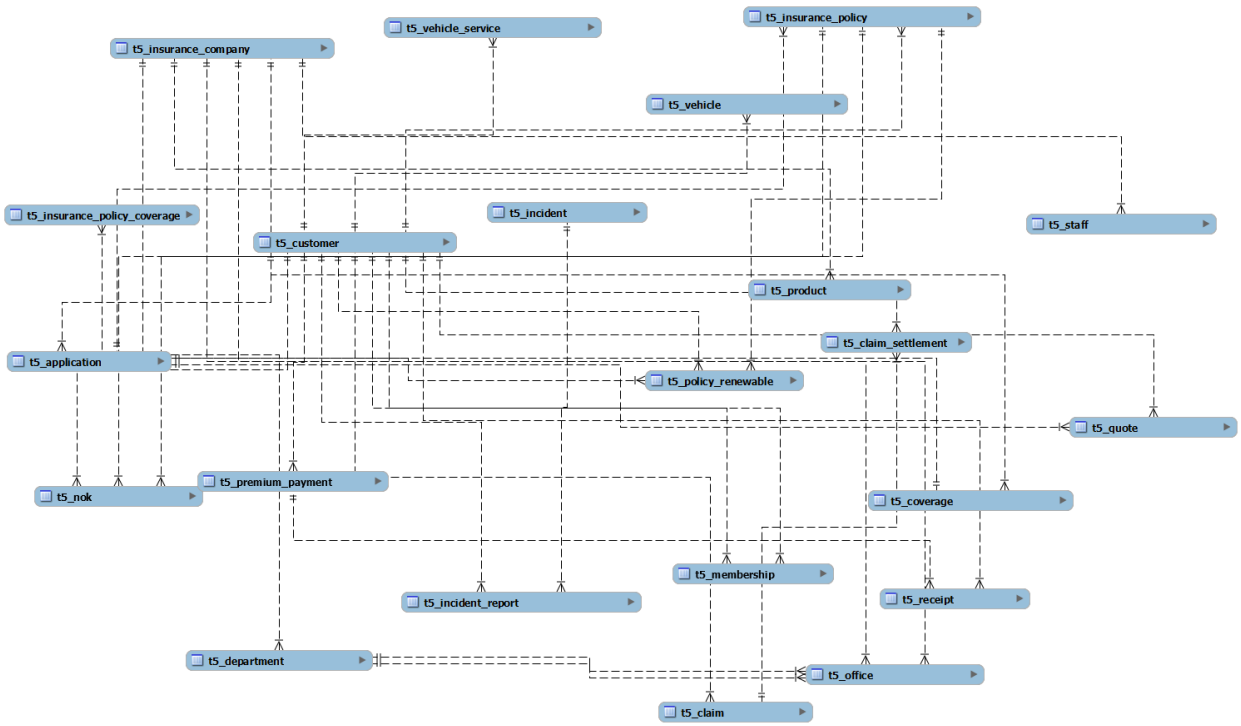
As everyone in a team cannot work(s) on every part of the project. We as a team of 9, have divided the entire work right from the start of gathering the information to implementing the database and queries. Although every member had an equal contribution to the project this table just shows the major role played by each team member in different parts of the project.

Name	Roll No.	Role/Description
GALI YASWANTH	20BCS046	Data Collection and SQL queries, Report
GUDISEVA DEEPAK SUJAY	20BCS049	Data Collection and SQL queries, ERD
KODURU ANIL KUMAR REDDY	20BCS072	Data Collection and SQL queries, Conceptual diagram
NIKAM YASH DINESH	20BCS093	Data Collection and SQL queries, Functions
NIMISH MANGEE	20BCS094	Data Collection and SQL query optimization
PARASA SAI TARUN	20BCS096	Data Collection and SQL queries, Report
PRAKASH AGARWAL	20BCS099	Data Collection and SQL queries, Procedures
VEMPRALA VENKATA SAI NIKHIL	20BCS135	Data Collection and SQL query optimization
SHUBH BINDAL	20BCS121	Data Collection and SQL queries, Presentation

3.) Data Modelling

a. Conceptual Data Modelling (CDM)

- The Conceptual data model is mainly used to define the relationship and the data entities which are used by the model.
- The conceptual data model is mostly used by data architects and business stakeholders.
- A Graphical representation of the actual database.
- It is used to define the relationship among the data entities but does not provide information about cardinality properties.

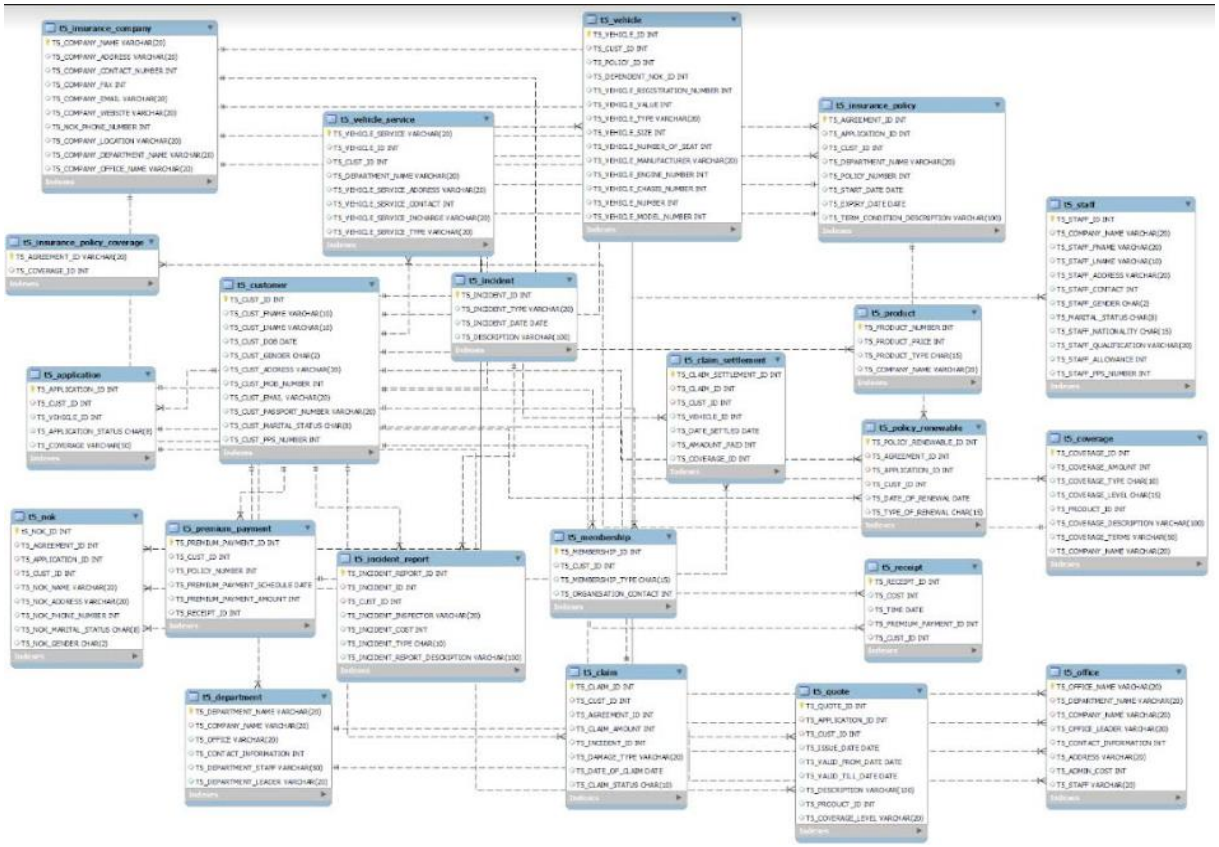


b. Logical Data Modelling (LDM)

- The Logical Data Modelling 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 input to the LDM is the output of CDM i.e., the conceptual data model of a car insurance company (with entity relationships). Normalization processes to the model are applied typically till 3NF.
- The output of the LDM is the ER(Entity-Relationship) diagram with all the assigned entity types and foreign keys, indexes, etc.
- We've used the EER modeling that exists in MySQL Workbench for creating the graphical representation of LDM i.e., an EER Diagram.
- The following diagram is the actual EER diagram of the car insurance company.

c. Physical Data Modelling (PDM)

- A Physical Data Model describes a database-specific implementation of the data model. It offers database abstraction and helps generate the schema.
- This is the model where actual tuples are implemented with data entities with optimizations that have partitioning or merging entities, duplicating data, and creating identification keys and indexes.
- Since it's a bit difficult to write/ code the entire SQL query for implementing tuples and managing different constraints related to primary key & foreign key, the Complete SQL is extracted by reverse engineering, a tool prebuilt in MySQL Workbench from the EER Diagram.
- Different Constraints related to a foreign key, primary key, attribute type(s), unique indexes, and normalization were resolved carefully by multiple times of reverse engineering of the EER Diagram by modifying it for better performance.



d. Most Commonly-Used My SQL Data Types:

1.) CHAR (size) - These are fixed-length character data of length-sized bytes. The maximum size of char is 2000 bytes.

2.) VARCHAR2 (size) - Variable-length character string having maximum length-sized bytes. The maximum size is 4000, and the minimum is 1. This is the most commonly-used data type. It replaces the old Oracle version 6 CHAR data type.

3.) INT- This data type is used for numerical values, of limited size. This data type is used for data on which calculation or sorting should be possible.

4.) DATE - Valid date range from January 1, 4712 BC to December 31, 4712 AD. The date data type also contains time components. We should use it only when you know the full date including day, month, and year.

4.) Stored Procedures and Functions

We have created 3 Stored procedures for clean code and bring partial automation in creating, deleting the database, and inserting the values. The Procedures are Create_Online_XYZ_Insurances_Database_Entities, Insert_Dataset, and Delete_All_Tables. Let's see how the following procedures are written.

Create_Online_XYZ_Insurances_Database_Entities: Stored Procedure to create all the tables. (Full code can be found in -------)

```
DELIMITER $$
CREATE PROCEDURE CREATE_ONLINE_XYZ_INSURANCES_DATABASE_ENTITES
()
DETERMINISTIC
BEGIN

CREATE TABLE T5_Customer(…….. );
CREATE TABLE T5_INSURANCE_COMPANY (……);
.
.
.
.
CREATE TABLE T5_INSURANCE_POLICY_COVERAGE(…..);
SET SQL_SAFE_UPDATES=0;
SHOW TABLES;
END $$
```

Insert_Dataset: Stored Procedure to insert all the data into the tables.

```
DELIMITER //
CREATE PROCEDURE INSERT_DATASET ()
```

```
DETERMINISTIC

BEGIN


INSERT INTO t5_customer VALUES (2001, 'Taylor', 'swift',
'1990-12-23', 'F', 'Pennsylvania',245698,
'taylorvma@gmail.com', 'A123456', 'Single', 3054586) ;

INSERT INTO t5_customer VALUES (2002, 'James', 'Harden',
'1987-10-6', 'M', 'Brooklyn', 685684, 'james@gmail.com',
'T848758', 'Married', 5648624) ;

…

…

END //
```

Delete_All_Tables: Stored Procedure to delete all the tables.

```
DELIMITER //

CREATE PROCEDURE DELETE_ALL_TABLES ()

DETERMINISTIC

BEGIN

DROP TABLES t5_application , t5_claim ,t5_claim_settlement
,t5_coverage ,t5_customer ,t5_department , t5_incident ,
t5_incident_report , t5_insurance_company ,
t5_insurance_policy , t5_insurance_policy_coverage ,
t5_membership ,t5_nok , t5_office , t5_policy_renewable ,
t5_premium_payment , t5_product , t5_quote , t5_receipt ,
t5_staff , t5_vehicle , t5_vehicle_service;


SHOW TABLES;

END //
```

The following functions are used in Queries for some specific purposes concerning those queries. These functions make the query code look much cleaner and increase readability.

1.) Return_claim_amount: returns claim amount using claim id

```
delimiter $$

create function return_claim_amount (a int)

returns int

reads sql data

deterministic

begin

     declare claim_amount int;

    set claim_amount = (select t5_claim_amount from t5_claim
where T5_CLAIM_ID = a);

    return (claim_amount);


end$$
```

2.) Return_premium_amount: returns premium amount using policy id

```
delimiter $$

create function return_premium_amount (a int)

returns int

reads sql data

deterministic

begin

     declare premium_amount int;

    set premium_amount = (select t5_premium_payment_amount
from t5_premium_payment where T5_POLICY_ID = a);

    return (premium_amount);


end$$
```

3.) Return_coverage_amount: returns coverage amount using coverage id

```
delimiter $$
create function return_coverage_amount (a int)
returns int
reads sql data
deterministic
begin
     declare coverage_amount int;
    set coverage_amount = (select t5_coverage_amount from
t5_coverage where T5_COVERAGE_ID = a);
    return (coverage_amount);

end$$
```

4.) Return_sum_of_ids: returns the sum of the ids given.

```
delimiter $$
create function sum_of_id (a int , b int, c int, d int)
returns int
reads sql data
deterministic
begin
     declare sumofids int;
    set sumofids = a+b+c+d;
    return (sumofids);

end$$
```

5.) Queries

➔ As a part of the project, we're supposed to execute 6 queries. A piece of brief information regarding each query is provided below.

Query-1:

Question: Retrieve Customer and Vehicle details who has been involved in an incident and claim status is pending – Customer, vehicle, claim status, incident

SQL Query:

```
select * from t5_customer as c, t5_vehicle as v
where c.t5_cust_id=v.t5_cust_id and v.t5_policy_id in
( SELECT t5_agreement_id  FROM T5_CLAIM  WHERE
T5_Claim_status='Pending');
```

Output:

```
-- Query 1
select * from t5_customer as c, t5_vehicle as v
where c.t5_cust_id=v.t5_cust_id and v.t5_policy_id in
( SELECT t5_agreement_id  FROM T5_CLAIM  WHERE T5_Claim_status='Pending');

##########################################################################################
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

T5_CUST_ID	T5_CUST_FNAME	T5_CUST_LNAME	T5_CUST_DOB	T5_CUST_GENDER	T5_CUST_ADDRESS	T5_CUST_MOB_NUMBER	T5_CUST_EMAIL	T5_CUST_PASSPORT_NUMBER	T5_CUST_MARITAL_STATUS	T5_CUST_PF
2008	Shreya	Ghoshal	1985-06-28	F	Mumbai	758595	shreya@gmail.com	T139782	Married	2186385
2005	Shubh	Bindal	2000-08-10	M	Delhi	984235	shubh@gmail.com	L346895	Single	4546952
2001	Taylor	swift	1990-12-23	F	Pennsylvania	245698	taylorvma@gmail.com	A123456	Single	3054586
2001	Taylor	swift	1990-12-23	F	Pennsylvania	245698	taylorvma@gmail.com	A123456	Single	3054586
2003	Lebron	James	1980-03-01	M	Cleveland	745356	lj@gmail.com	P365944	Married	5478923
2009	Sanket	Singh	1976-06-07	M	Amritsar	758556	sanket@gmail.com	U742368	Single	2386131

Query-2:

Question: Retrieve Customer and Vehicle details who has been involved in an incident and claim status is pending – Customer, vehicle, claim status, incident

SQL Query:

```
select * from t5_customer c
where c.T5_CUST_ID in
(select a.T5_CUST_ID from t5_customer a
inner join t5_premium_payment b on a.T5_CUST_ID = b.T5_CUST_ID
where b.T5_PREMIUM_PAYMENT_AMOUNT > (select sum(t5_cust_id)
from t5_customer));
```

Output:

```
-- Query 2
select * from t5_customer c where c.T5_CUST_ID in
(select a.T5_CUST_ID from t5_customer a inner join t5_premium_payment b on a.T5_CUST_ID = b.T5_CUST_ID
where b.T5_PREMIUM_PAYMENT_AMOUNT > (select sum(t5_cust_id) from t5_customer));
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

T5_CUST_ID	T5_CUST_FNAME	T5_CUST_LNAME	T5_CUST_DOB	T5_CUST_GENDER	T5_CUST_ADDRESS	T5_CUST_MOB_NUMBER	T5_CUST_EMAIL	T5_CUST_PASSPORT_NUMBER	T5_CUST_MARITAL_STATUS	T5_CUST_PP
2006	Yaswanth	Gali	2001-06-14	M	Vijayawada	709759	yaswanth@gmail.com	K375159	Single	8866825
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query-3:

Question: 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

SQL Query:

```
select * from t5_insurance_company where
(select count(*) from t5_product
where t5_product.t5_company_name =
t5_insurance_company.t5_company_name)
>
(select count(*) from (select count(*) from t5_office
where
t5_office.t5_company_name=t5_insurance_company.t5_company_name
group by t5_office.t5_department_name
having count(*) > 1) as Sec);
```

Output:

```
##Query 3
select * from t5_insurance_company where
(select count(*) from t5_product where t5_product.t5_company_name = t5_insurance_company.t5_company_name)
>
(select count(*) from (select count(*) from t5_office where t5_office.t5_company_name=t5_insurance_company.t5_company_name group by t5_office.t5_department_name
having count(*) > 1) as Sec);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

T5_COMPANY_NAME	T5_COMPANY_ADDRESS	T5_COMPANY_CONTACT_NUMBER	T5_COMPANY_FAX	T5_COMPANY_EMAIL	T5_COMPANY_WEBSITE	T5_NOK_PHONE_NUMBER	T5_COMPANY_LOCATION	T5_COMPANY_DEPARTMENT_NAME	T5_COMPANY_OFFICE_NAME
Birla Insurance	600001, Chennai	962610	152937	aditya@yahoo.in	adityabirla.com	962610	Chennai	Birla Life Insurance	Birla office
Exide Life Insurance	500001, Hyderabad	722373	399331	exide@gmail.com	exidelife.com	722373	Hyderabad	Exide Life Insurance	Exide office
HDFC Life Insurance	226001, Lucknow	959943	582924	hdfc@gmail.com	hdfclife.com	959943	Lucknow	HDFC Life Insurance	HDFC office
ICICI Life Insurance	380058, Bopal	732771	527273	icici@yahoo.in	icicilife.com	732771	Bopal	ICICI Life Insurance	ICICI office
Kotak Life Insurance	530068, Bangalore	820449	373156	kotak@gmail.co.in	kotakmahindra.com	820449	Bangalore	Kotak Life Insurance	Kotak office
LIC of India	700001, Kolkata	799718	469932	lic@gmail.com	licindia.com	799718	Kolkata		LIC office
PNB Life Insurance	140119, Chandigarh	819986	664028	pnb@outlook.co.in	pnblife.com	819986	Chandigarh	PNB Life Insurance	PNB office
SBI Life Insurance	110001, New Delhi	874135	259369	sbi@outlook.com	sbilife.com	874135	New Delhi	SBI Life Insurance	SBI office
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query-4:

Question: Select Customers who have more than one vehicle, where the premium for one of the Vehicles is not paid and it is involved in an accident

Assumption: In the T5_Premium_Payment table, if the receipt id is null, then it means that the premium for one of the vehicles is not paid.

SQL Query:

```
select * from t5_customer
where t5_cust_id in(
  select t5_cust_id from t5_incident_report
  where t5_incident_id in(
         select t5_incident_id from t5_claim
         where t5_agreement_id in(
                select t5_policy_id from t5_premium_payment
                where t5_policy_id in (
                       select t5_policy_id from t5_vehicle
                              where t5_cust_id in
                              (select t5_cust_id from t5_vehicle
                                     group by T5_CUST_ID
                                     having count(t5_cust_id)>1)
  and T5_RECEIPT_ID is null)) and T5_INCIDENT_TYPE = 'Accident'));
```

Output:

```
-- Query 4
select * from t5_customer
where t5_cust_id in(
    select t5_cust_id from t5_incident_report
    where t5_incident_id in(
        select t5_incident_id from t5_claim
        where t5_agreement_id in(
            select t5_policy_id from t5_premium_payment
            where t5_policy_id in (
                select t5_policy_id from t5_vehicle
                where t5_cust_id in
                (
                    select t5_cust_id from t5_vehicle
                    group by T5_CUST_ID
                    having count(t5_cust_id)>1
                )
 and T5_RECEIPT_ID is null)) and T5_INCIDENT_TYPE = 'Accident'));
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

T5_CUST_ID	T5_CUST_FNAME	T5_CUST_LNAME	T5_CUST_DOB	T5_CUST_GENDER	T5_CUST_ADDRESS	T5_CUST_MOB_NUMBER	T5_CUST_EMAIL	T5_CUST_PASSPORT_NUMBER	T5_CUST_MARITAL_STATUS	T5_CUST_PPS_NUMBER
2005	Shubh	Bindal	2000-08-10	M	Delhi	984235	shubh@gmail.com	L346895	Single	4546952
2001	Taylor	swift	1990-12-23	F	Pennsylvania	245698	taylorvma@gmail.com	A123456	Single	3054586
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query-5:

Question: Select all vehicles which have a premium more than its vehicle number.

SQL Query:

```
select * from t5_vehicle
where t5_vehicle_id in
(select c.t5_vehicle_id from
(select t5_vehicle_id, t5_vehicle_number,
return_premium_amount(t5_policy_id) as b
from t5_vehicle a)c
where (c.b> c.t5_vehicle_number));
```

Output:

```
-- Query 5
select * from t5_vehicle where t5_vehicle_id in
(select c.t5_vehicle_id from (select t5_vehicle_id, t5_vehicle_number, return_premium_amount(t5_policy_id) as b from t5_vehicle a)c
where (c.b > c.t5_vehicle_number));
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

T5_VEHICLE_ID	T5_CUST_ID	T5_POLICY_ID	T5_DEPENDENT_NOK_ID	T5_VEHICLE_REGISTRATION_NUMBER	T5_VEHICLE_VALUE	T5_VEHICLE_TYPE	T5_VEHICLE_SIZE	T5_VEHICLE_NUMBER_OF_SEAT	T5_VEHICLE_MANUFACTURER	T5_VEHICLE_ENGINE_NUMBER
2506	2006	25066	3306	2536	14000000	4 wheeler	5	4	BMW	2501006
2509	2009	25099	3309	2539	6400000	4 wheeler	5	4	BMW	2501009
2512	2012	25133	3312	2542	1100000	4 wheeler	6	5	TATA	2501012
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query-6:

Question: 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)

SQL Query:

```
select * from t5_customer
where t5_cust_id in (select d.t5_cust_id from (select
t5_cust_id,
return_claim_amount(t5_claim_id) as a,
return_coverage_amount(t5_coverage_id)as b,
sum_of_id(t5_CLAIM_SETTLEMENT_ID, t5_VEHICLE_ID, t5_CLAIM_ID,
t5_CUST_ID)as c
from t5_claim_settlement)d where (d.a<d.b and d.a>d.c));
```

Output:

```
-- Query 6
select * from t5_customer where t5_cust_id in (select d.t5_cust_id from (select t5_cust_id,
return_claim_amount(t5_claim_id) as a,
return_coverage_amount(t5_coverage_id)as b,
sum_of_id(t5_CLAIM_SETTLEMENT_ID, t5_VEHICLE_ID, t5_CLAIM_ID, t5_CUST_ID)as c
from t5_claim_settlement)d where (d.a < d.b and d.a > d.c));
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

T5_CUST_ID	T5_CUST_FNAME	T5_CUST_LNAME	T5_CUST_DOB	T5_CUST_GENDER	T5_CUST_ADDRESS	T5_CUST_MOB_NUMBER	T5_CUST_EMAIL	T5_CUST_PASSPORT_NUMBER	T5_CUST_MARITAL_STATUS	T5_CUST_PPS_NUMBER
2007	Kobe	Brynt	1979-08-10	M	Stapple centre	856348	kobe@gmail.com	O486213	Married	2168745
2011	Pooja	Shetty	1984-12-03	F	Banglore	956545	pooja@gmail.com	D864869	Single	5815973
2012	Salman	Khan	1972-08-09	M	Mumbai	957845	bhai@gmail.com	X123789	Married	695173
2013	Kavya	Maran	1995-04-24	F	Chennai	335548	sunrisers@gmail.com	S456739	Single	4486251
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

6.) Conclusion

A complete Car Vehicle Insurance company database is completely implemented and all the given project queries are executed and are completely working fine giving at least one line of output. Each table consists of at least 10 tuples of data. Further developments for this project can be making a user interface for this database to perform INSERT, UPDATE, DELETE operations, and Normalisation of a few tables till 4th NF.