

**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
   1. Project Title
   2. Aim
   3. Purpose
   4. Scope
   5. Project Benefits
   6. Information about MySQL
   7. Software Requirements
2. Team & Major Roles Played
3. Database Modelling
   1. Conceptual Data Modelling (CDM)
   2. Logical Data Modelling (LDM)
   3. 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.

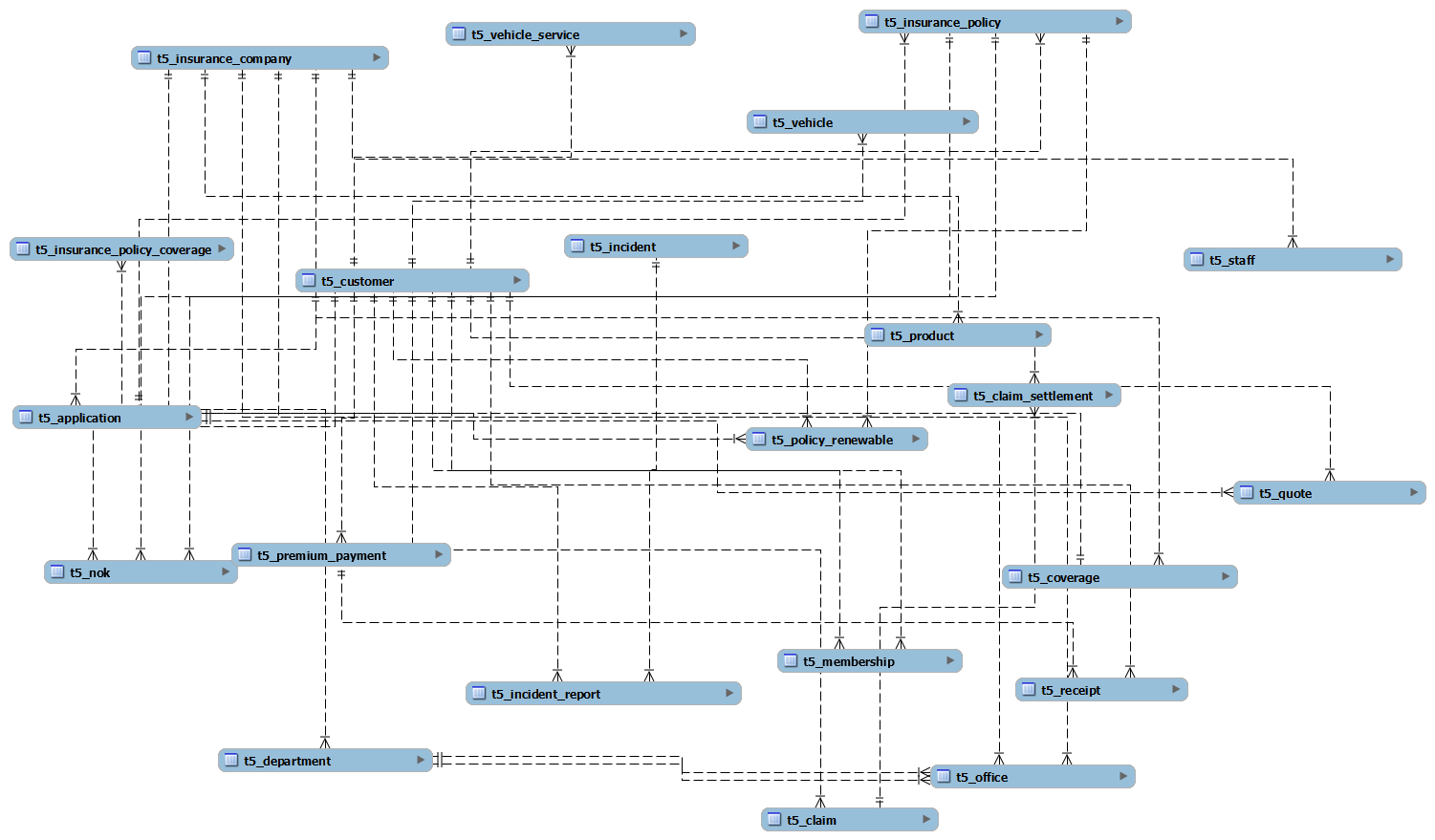
|  |  |  |
| --- | --- | --- |
| Name | Roll No. | Role/Description |
|  |  |  |
| GALI YASWANTH | 20BCS046 | Data Collection and SQL queries, Report |
| GUDISEVA DEEPAK SUJAY | 20BCS049 | Data Collection and SQL queries, ERD |
| KODURU ANIL KUMAR REDDY | 20BCS072 | Data Collection and SQL queries, Conceptual  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

1. **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.

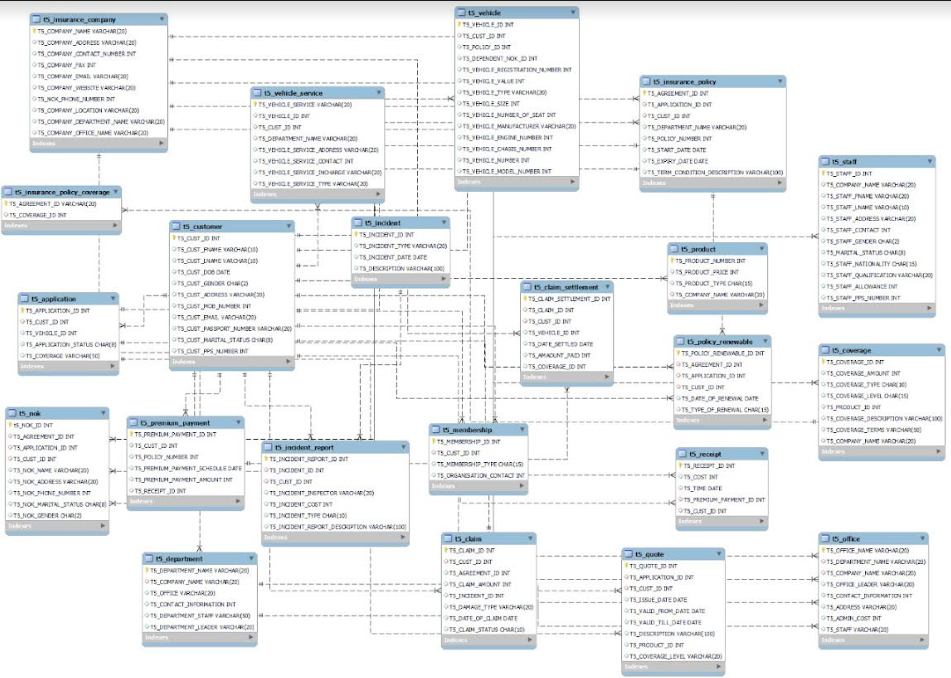


**b. Logical Data Modelling (LDM)**

* 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.

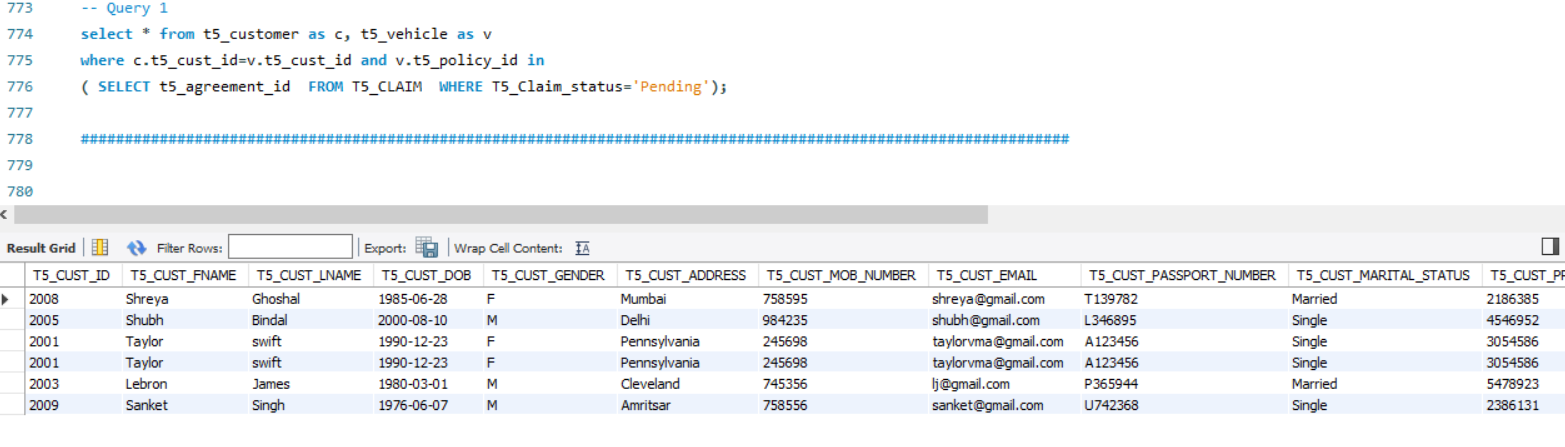
**Query-1:**

**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:**

1. select \* from t5\_customer as c, t5\_vehicle as v
2. where c.t5\_cust\_id=v.t5\_cust\_id and v.t5\_policy\_id in
3. ( SELECT t5\_agreement\_id FROM T5\_CLAIM WHERE T5\_Claim\_status='Pending');

**Output:**



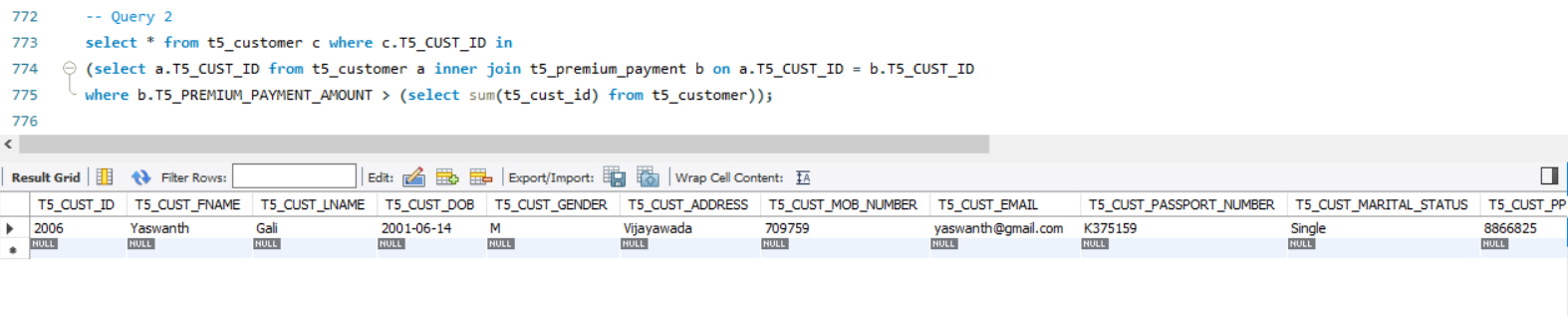
**Query-2:**

**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:**

1. select \* from t5\_customer c
2. where c.T5\_CUST\_ID in
3. (select a.T5\_CUST\_ID from t5\_customer a
4. inner join t5\_premium\_payment b on a.T5\_CUST\_ID = b.T5\_CUST\_ID
5. where b.T5\_PREMIUM\_PAYMENT\_AMOUNT > (select sum(t5\_cust\_id) from t5\_customer));

**Output:**



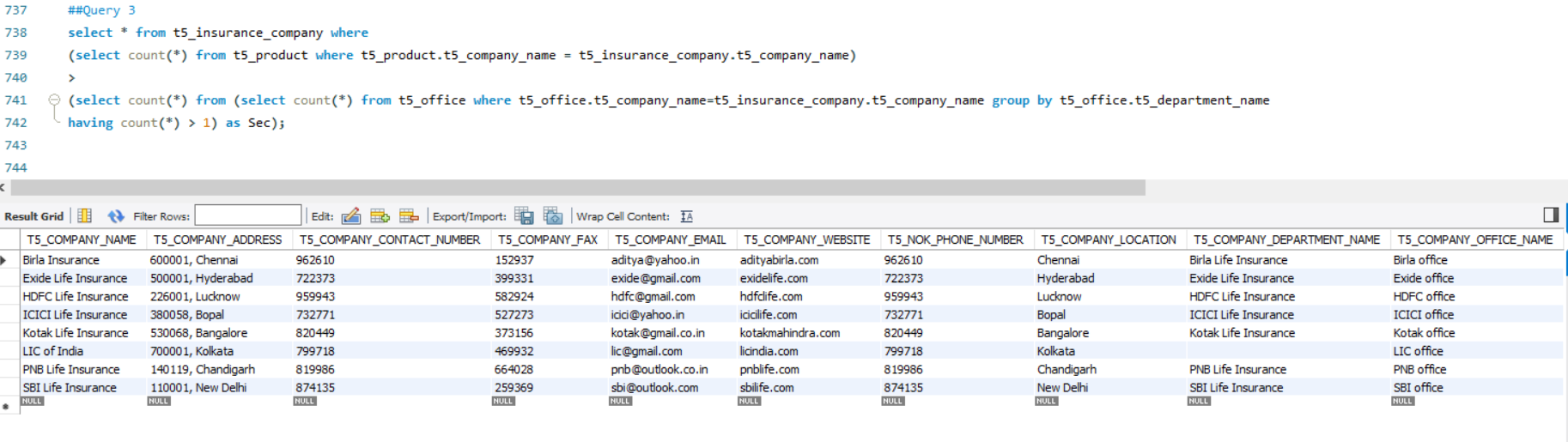
**Query-3:**

**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:**

1. select \* from t5\_insurance\_company where
2. (select count(\*) from t5\_product
3. where t5\_product.t5\_company\_name = t5\_insurance\_company.t5\_company\_name)
4. >
5. (select count(\*) from (select count(\*) from t5\_office
6. where t5\_office.t5\_company\_name=t5\_insurance\_company.t5\_company\_name
7. group by t5\_office.t5\_department\_name
8. having count(\*) > 1) as Sec);

**Output:**



**Query-4:**

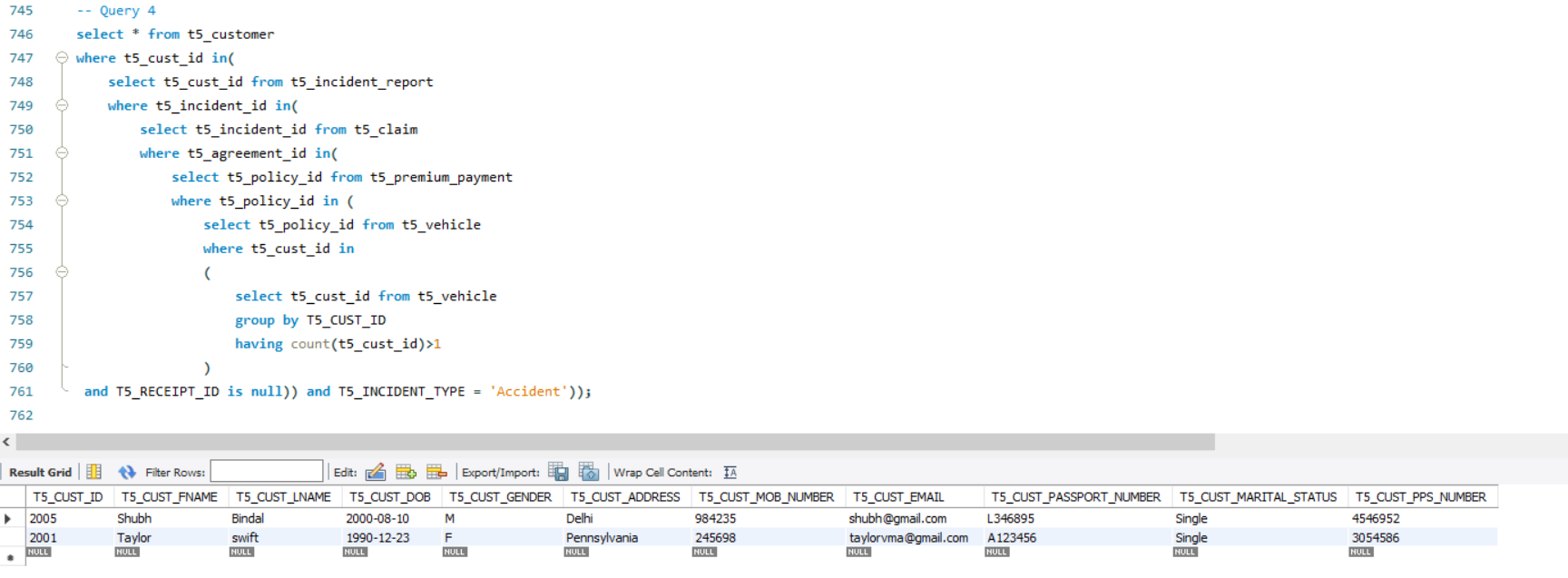
**Question:** Select Customers who have more than one vehicle, where the premium for one of the Vehicles is not paid and it is involved in an accident

**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:**

1. select \* from t5\_customer
2. where t5\_cust\_id in(
3. select t5\_cust\_id from t5\_incident\_report
4. where t5\_incident\_id in(
5. select t5\_incident\_id from t5\_claim
6. where t5\_agreement\_id in(
7. select t5\_policy\_id from t5\_premium\_payment
8. where t5\_policy\_id in (
9. select t5\_policy\_id from t5\_vehicle
10. where t5\_cust\_id in
11. (select t5\_cust\_id from t5\_vehicle
12. group by T5\_CUST\_ID
13. having count(t5\_cust\_id)>1)
14. and T5\_RECEIPT\_ID is null)) and T5\_INCIDENT\_TYPE = 'Accident'));

**Output:**



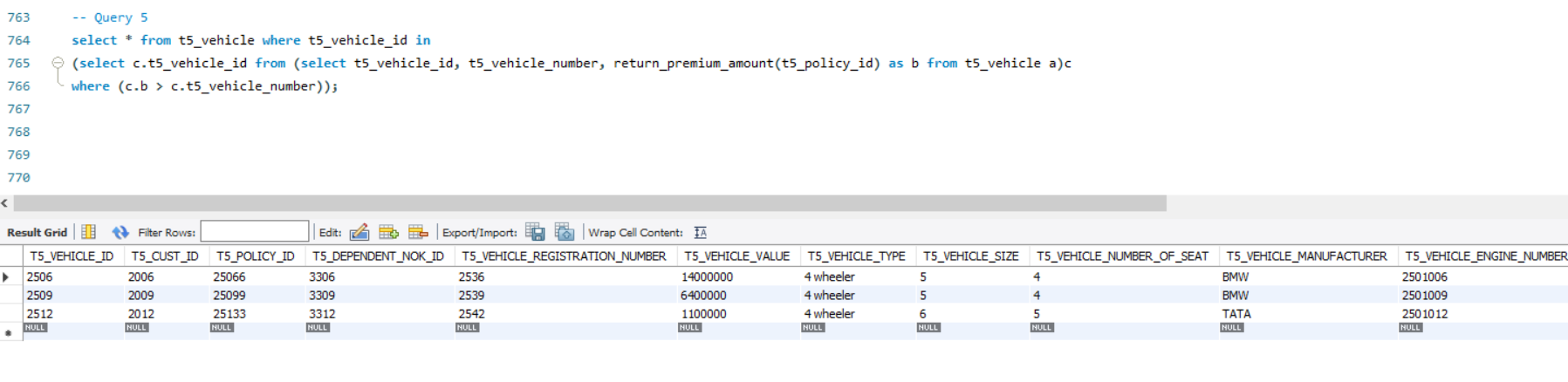
**Query-5:**

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

**SQL Query:**

1. select \* from t5\_vehicle
2. where t5\_vehicle\_id in
3. (select c.t5\_vehicle\_id from
4. (select t5\_vehicle\_id, t5\_vehicle\_number, return\_premium\_amount(t5\_policy\_id) as b
5. from t5\_vehicle a)c
6. where (c.b> c.t5\_vehicle\_number));

**Output:**



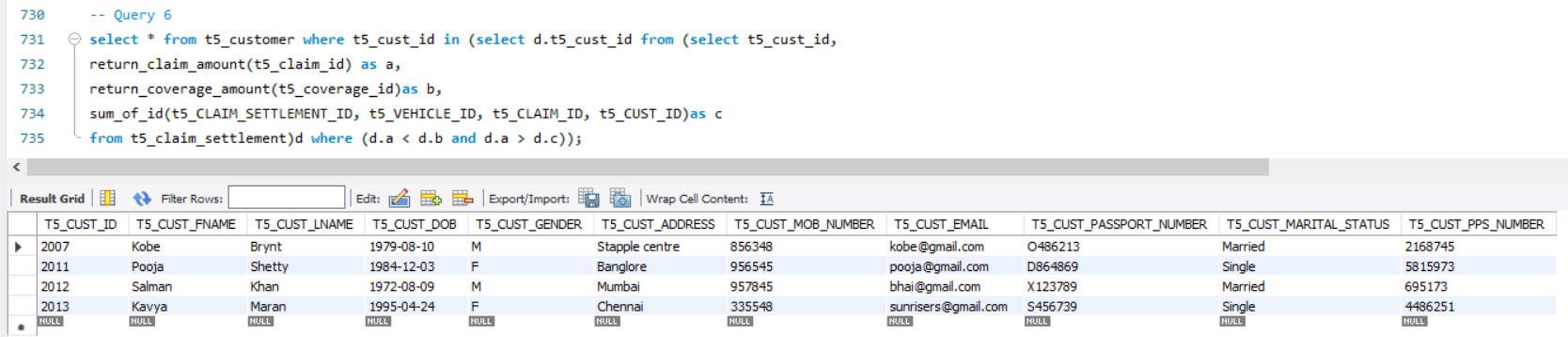
**Query-6:**

**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:**

1. select \* from t5\_customer
2. where t5\_cust\_id in (select d.t5\_cust\_id from (select t5\_cust\_id,
3. return\_claim\_amount(t5\_claim\_id) as a,
4. return\_coverage\_amount(t5\_coverage\_id)as b,
5. sum\_of\_id(t5\_CLAIM\_SETTLEMENT\_ID, t5\_VEHICLE\_ID, t5\_CLAIM\_ID, t5\_CUST\_ID)as c
6. from t5\_claim\_settlement)d where (d.a<d.b and d.a>d.c));

**Output:**



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.