

**DATABASE MANAGEMENT SYSTEMS**

**(COURSE CODE: CS310)**

Report On

**DATABASE DESIGN ON CAR VEHICLE INSURANCE COMPANY**

**Submitted by**

Group -2

**Under the Guidance of:**

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

Ms. Supriya Nadiger (Lab Assistant, IIIT DHARWAD)

**(Project GitHub Link:** <https://github.com/fharookshaik/DBMS-Project-Vehicle-Insurance>**)**

**Index…**

1. About the Project
   1. Project Title
   2. Aim
   3. Purpose
   4. Scope
   5. Project Benefits
   6. A Quick 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. Python Project (Generating the optimized SQL Queries using data extracted from excel sheet)
5. Project Queries (1 to 6)
6. Conclusion & Future Improvements

**1. About the Project**

**Project Title:** A database for a Vehicle Insurance company.

**Aim:**

The aim of this project is to create and maintain a car insurance database implemented in MySQL Database System and to retrieve information i.e., stored in database both efficiently and conveniently.

**Purpose:**

The purpose of this project is to acquire good amount of knowledge as well as practical experience in advanced entity modelling, normalisation, transactional relational database design, SQL and PL/SQL coding, and generation of data backed management reports.

**Scope:**

The scope of this project is confined to a database administrator or 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 and basic knowledge of python. In this project, we’ve implemented the entire database in MySQL Database. To proceed with the project, one need to be familiar with MySQL Workbench and MySQL Server. As of now, this project is entirely developed in a WINDOWS machine and hope it’ll work with the Linux and mac machines.

**Project Benefits:**

To develop a good DB, that could be used with analytical tools and faster in delivering the accurate right information at accurate times for a better decision making.

**A Quick information about MySQL:**

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS (Relational Database management system(s)) being used for developing various web-based software applications. MySQL is developed, marketed 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 task of managing information would simply use SQL to transmit instructions.

**Software Requirements:**

* A Windows/ Linux/ Mac system.
* Properly installed MySQL server & MySQL Workbench.
* Python (version: 3.8.5 or higher)
* An Excel sheet Viewer.

**2. Team & Major Roles Played**

As said, not everyone in a team, can work(s) on every part of the project. We as a team of 10, have divided the entire work right from the start of gathering the information to implementing the database. We’ve divided the entire work into **3** major parts.

1. Collection of data into an excel sheet, maintaining and optimizing it for better execution.
2. Creating a database and maintaining it.
3. Importing the data into the database.

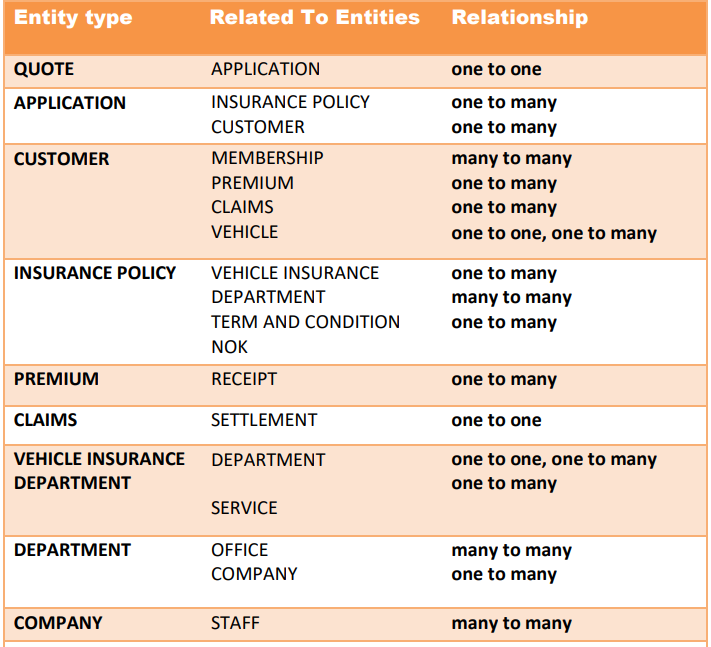
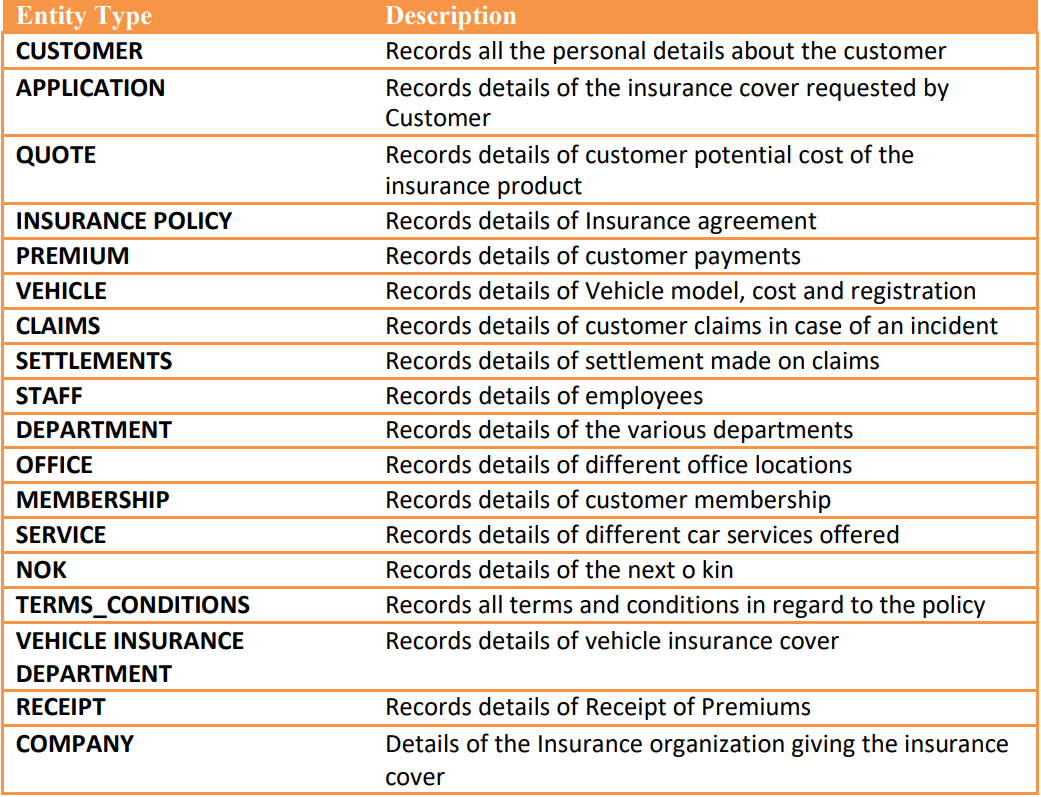
Although, everyone is actively participated while tacking and finding solutions the Project Queries. The following table just shows information of people who worked majorly in particular assigned parts.

**3. Data Modelling**

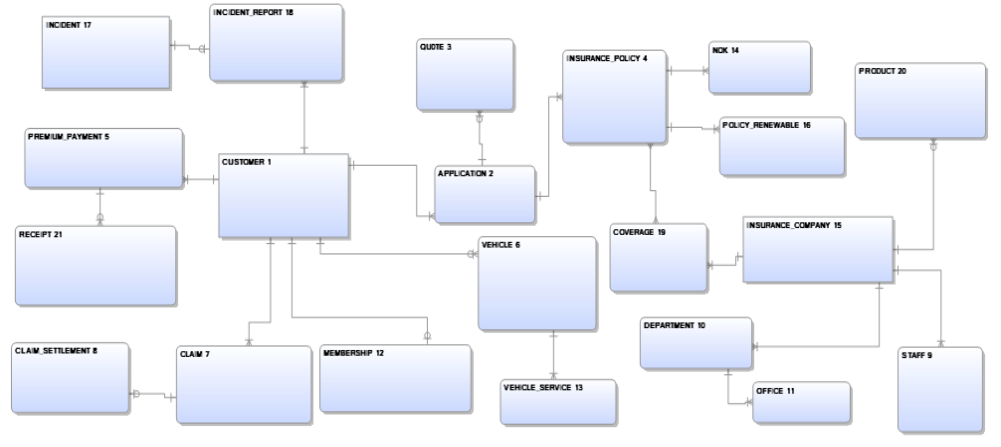
1. **Conceptual Data Modelling (CDM)**

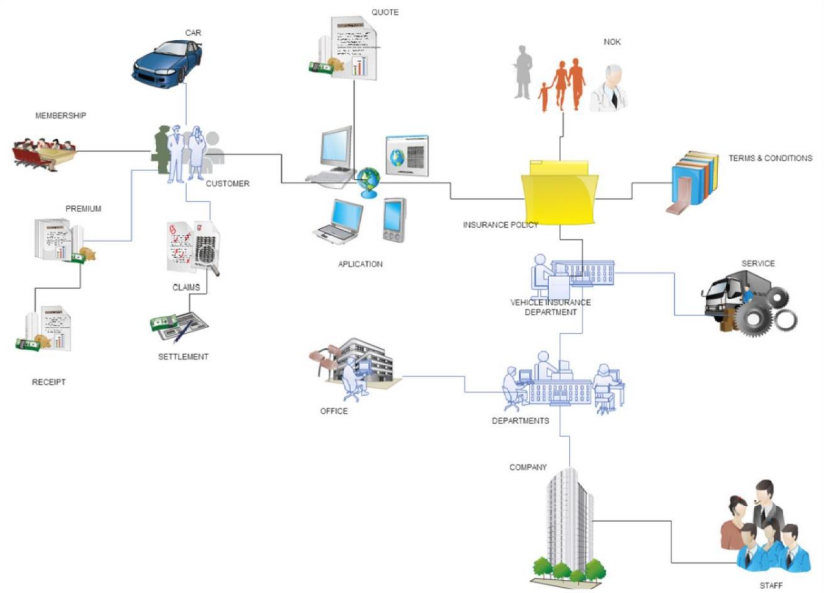
* The Conceptual data model mainly focusses on the entities and their relationships and properties that are embedded in the problem. It’s a best use for communication with stakeholders.
* It’s basically a Graphical representation of the actual database.
* In this modelling, all the entities are described along with their relationships.
* The following tables will show the actual entities used and their relationships with other entities.

**Table of Entity Relationships b/w Entities**



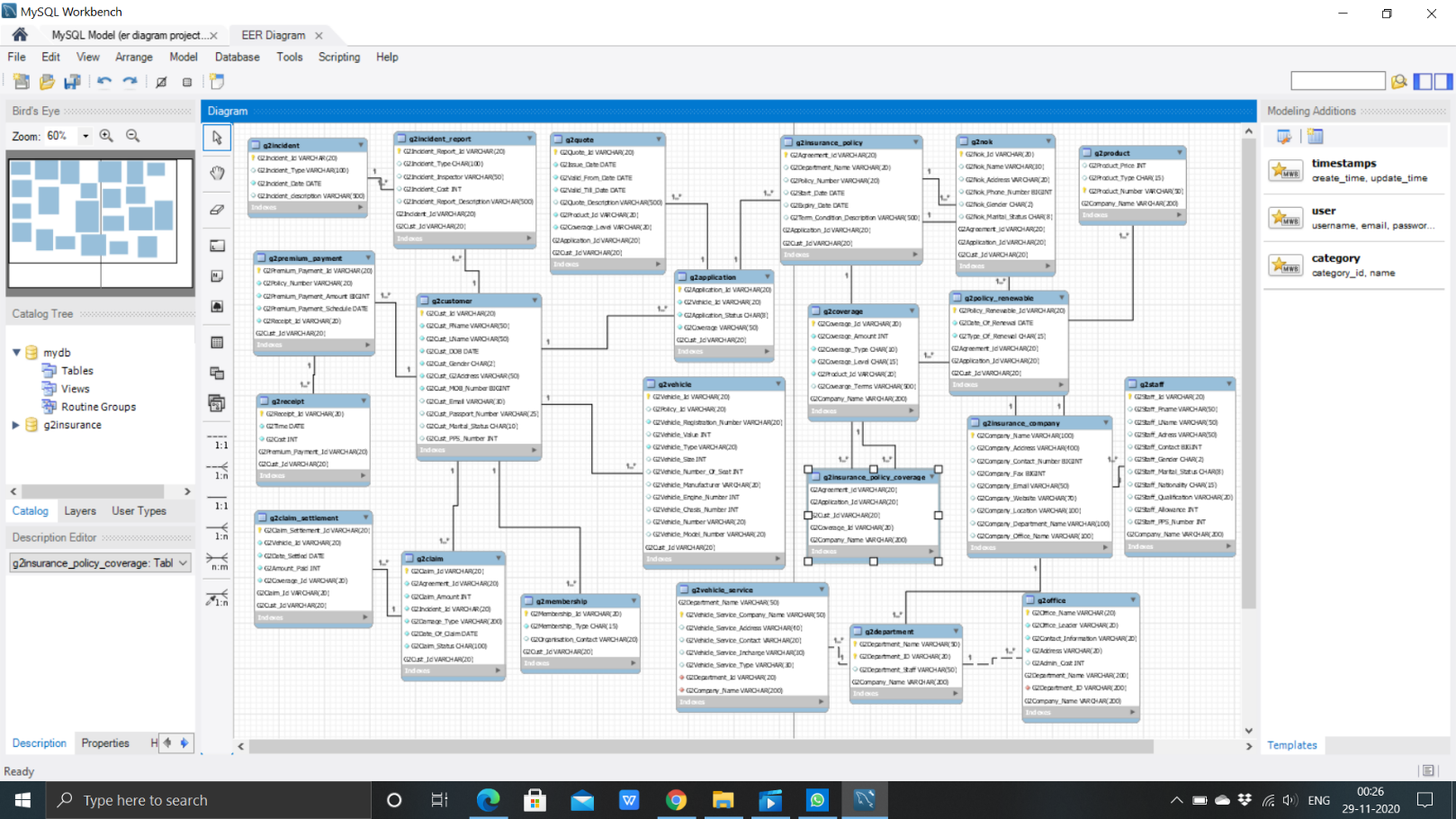
**Graphical Representation of Conceptual Data Model Conceptual data model of Car Insurance Database**

****

****

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

* The logical data modelling is a step from conceptual data model to a data management technology (relationship databases) and is subject of normalisation concept.
* The input to a LDM is the output of CDM i.e., conceptual data model of car insurance company (with entity relationships)
* The output of a LDM is the ER(Entity-Relationship) diagram with all the assigned entity types and foreign key, indexes etc.



* We’ve used the EER modelling 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. (for a better picture please find the ‘.mwb’ file in the project files)

1. **Physical Data Modelling (PDM)**

* This is the model where actual tuples are implemented with data entities with optimizations that have partitioning or merging entities, duplicating data, 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 Constrains that related to 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.
* The actual SQL code for implementing all the tuples etc. can be find under project files ‘G2INSURANCE.sql’

**Most Commonly-Used Oracle Data Types:**

1. **char (size) -** these are fixed-length character data of length-sized bytes. Maximum size is 2000 bytes.

**Typical use**: for official International Currency Codes which are a fixed three characters in length such as USD, FFR.

1. **VARCHAR2 (size) -** Variable-length character string having maximum length-sized bytes. Maximum size is 4000, and minimum is 1. This is the most commonly-used data type and you should use it if you are not sure which one to use. It replaces the old Oracle version 6 CHAR data type.

**Typical use**: for storing individual ASCII text lines of unlimited length ASCII texts on which you need to be able to search using a wildcard.

1. **NUMBER -** This data type is used for numerical values, with or without a decimal, of virtually unlimited size. This data type is used for data on which calculation or sorting should be possible. Avoid its use for numbers like a phone number, where the value does not have any meaning.

**Typical use**: amount of money, quantities, generated unique key values.

1. **DATE** - Valid date range from January 1, 4712 BC to December 31, 4712 AD. A date data type also contains time components. You should use it only when you know the full date including day, month, and year. The time component is often set to 00:00 (midnight) in normal use of dates.

**Typical use**: any date where the full date is known.

1. **LONG** - Character data of variable length up to 2 gigabytes. Obsolete since Oracle8. Was used for ASCII text files where you do not need to search using the wildcard or substring functionality. Use CLOB data type instead.

**Typical use**: for storing the source code of HTML pages.

**4. Python Project**

**Title:** Generating the Optimized SQL Queries using the data extracted from Excel Sheet.

**Aim & About the Project:**

Although you’ve implemented the database successfully, importing the data into the database is definitely an overhead for a programmer/ a data analyst/ a data administer. The main aim of the project is to generate the SQL queries that are optimized for execution data extracted from respective Excel Sheet.

**Python Packages needed:**

* MySQL Connector
* openpyxl
* OS
* shutil
* time

**Experience:**

Being familiarity with the python programming language is more than enough. One should know how to download python packages/ modules using **‘pip’** and a minimum amount of knowledge how to import python packages.

**Input:**

* The input to the program is the excel files placed under ‘attachments’ folder in the present working directory.
* The code is written in such a way that it’ll look into the excel files under the folder ‘attachments’ folder.

**Output:**

* The output of the program is the optimized SQL queries all place under a new folder automatically created by code named ‘sqlfiles’.
* For our convenience, we’ve prepared a single SQL file which contains all the projectdata.

**Python Code:**

# INPUT -> ALL EXCEL FILES PLACED UNDER attachments folder

# OUTPUT -> Import data from each excel file and outputs the sql query for each table

import mysql.connector as mysqlc

import openpyxl

import shutil

import os

import time

# Load the data from the excel sheet into a list named ‘data’.

def dataFromExcel(notebook):

    print(f"[{time.ctime()}] IMPORTING DATA FROM {notebook}")

    data = []

    excelFile = openpyxl.load\_workbook(notebook)

    sheet = excelFile.active

    for row in sheet.rows:

        rowData = []

        for cell in row:

            rowData.append(cell.value)

        data.append(tuple(rowData))

    print(f"[{time.ctime()}] DATA IMPORTED SUCCESSFULLY FROM {notebook}")

    return data

# Conversion of type table attributes according to the MySQL database for better importing

def typeconversion(attrType,values):

    values = list(values)

    for i in range(len(attrType)):

        if attrType[i] == b'date':

            temp = str(values[i])

            values[i] = temp[:-9]

        if values[i] == None:

            values[i] = ''

    return tuple(values)

if \_\_name\_\_ == "\_\_main\_\_":

    print(f"[{time.ctime()}] STARTING PROCESS")

    print(f"[{time.ctime()}] CONNECTING TO MYSQL DATABASE")

    # Connecting to database

    conn = mysqlc.connect(

        host = "localhost",

        username = "root",

        password = "1106",

        database = "g2insurance"

    )

    if conn != None:

        print(f"[{time.ctime()}] CONNECTED TO MYSQL DATABASE")

        print(conn)

    print(f"[{time.ctime()}] SETTING UP CURSOR")

    cursor = conn.cursor()

    # Excel files folder

    folderPath = os.getcwd() + r"**\\**attachments"

    print(f"[{time.ctime()}] LOOKING FOR EXCEL SHEETS IN {folderPath}")

    # New folder for SQL files

    sqlPath = os.getcwd() + r"**\\**sqlfiles"

    if os.path.isdir(sqlPath):

        print(f"[{time.ctime()}] {sqlPath} FOUND. DELETING IT AND CREATING A NEW ONE")

        # os.rmdir(sqlPath) -> Linux

        shutil.rmtree(sqlPath)

        os.mkdir(sqlPath)

    else:

        print(f"[{time.ctime()}] {sqlPath} NOT FOUND. CREATING A NEW ONE")

        os.mkdir(sqlPath)

    fileNames = os.listdir(folderPath)

    for i in fileNames:

        # loading the excel file

        dataNotebook = folderPath + f"\\{i}"

        # new sql table name

        tableName = "g2" + f"{i[:-5].lower()}"

        print("\n\n\n")

        print(f"[{time.ctime()}] LOOKING INTO '{i}', CREATED A NEW TABLE NAMED '{tableName}' ")

        data = dataFromExcel(dataNotebook)

        sqlfile = open(f'{tableName}.sql','a')

        print(f"[{time.ctime()}] CREATING A NEW SQL FILE {tableName}.sql")

# looking for attributes and it’s types from MYSQL Database for optimizing the data extracted from Excel sheet.

        attr = []

        attrType = []

        print(f"[{time.ctime()}] LOOKING FOR {tableName} IN MYSQL DATABASE")

        cursor.execute(f"DESC {tableName}")

        print(f"[{time.ctime()}] FOUND {tableName}. ACQUIRING ATTRIBUTE TYPES OF THE TABLE")

        for x in cursor:

            attr.append(x[0])

            attrType.append(x[1])

        # print(attr)

        # print(attrType)

        print(f"[{time.ctime()}] OPTIMIZING DATA FOR CREATING SQL QUERIES & GENERATING SQL QUERIES FOR {tableName} DATA IMPORTED FROM {dataNotebook}")

        for i in range(1,len(data)):

            values = data[i]

            finalvalues = typeconversion(attrType,values)

            # query = f"INSERT INTO {tableName} {tuple(attr)} VALUES {finalvalues} ;\n"

            query = f"INSERT INTO {tableName} VALUES {finalvalues} ;\n"

            sqlfile.write(query)

        sqlfile.close()

        print(f"[{time.ctime()}] SUCCESSFULLY CREATED {tableName}.sql in '{sqlPath}' ")

        originalPath = os.getcwd() + f"\\{tableName}.sql"

        finalPath = f"{sqlPath}" + f"\\{tableName}.sql"

        shutil.move(originalPath, finalPath)

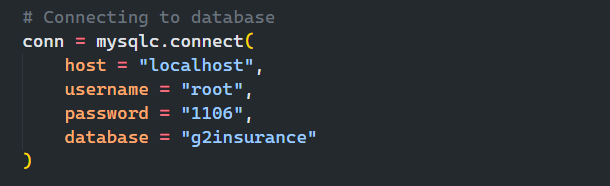
        time.sleep(1)

    print(f"\n\n[{time.ctime()}] ALL DONE")

* For more information regarding the python code please review the project video named ‘G2 Intro & Python Code’

**Important points to be noted in order to execute the python code in your machine:**

* All the excel files should be placed under ‘attachments’ folder and only excel files should be there in that particular folder.
* As of now, the project is still in development stage. So please keep in mind that all the type conversions may not be converted.
* The database connection details mentioned below should be modified according to your needs.

****

* Due to time constraint, the program is writes according to the requirements we need.
* The program, written one of our team mates who solely developed the entire project will be working for better execution the program in different machines. For more information regarding the project find the <https://github.com/fharookshaik/excel-data-to-sql-queries>

**5. Project Queries**

* As a part of the project we’re suppose to execute 6 queries. A brief information regarding each query are provided below.

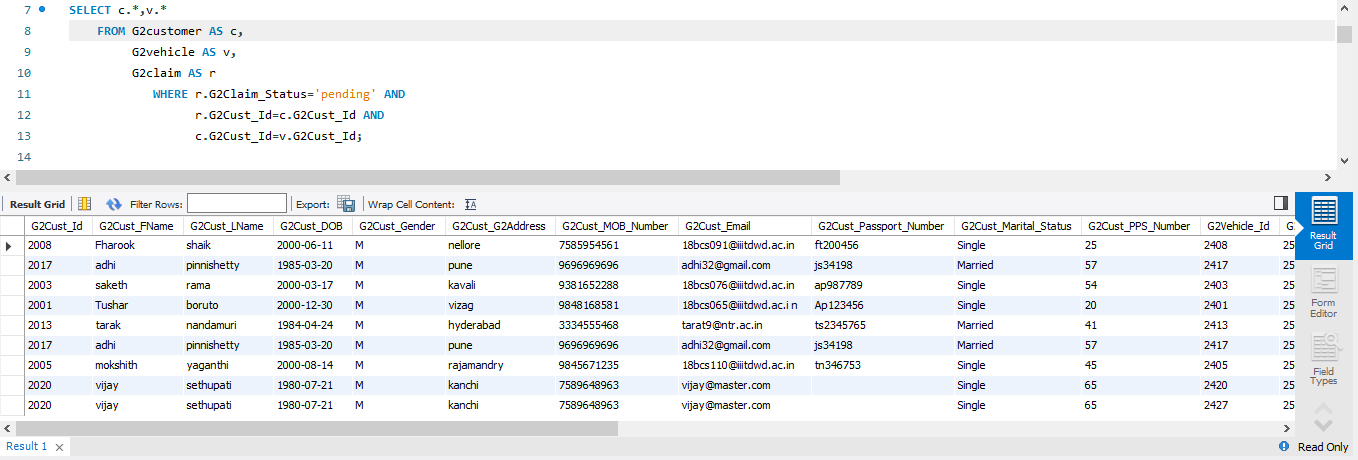
**Query -1:**

**Question:** Retrieve Customer and Vehicle details who has been involved in an incident and claim status is pending**.**

**Assumption:** The claim status = 'pending' indirectly mean that the particular vehicle is met with any incident.

**SQL Query:**

1. SELECT c.\*,v.\*
2. FROM G2customer AS c,
3. G2vehicle AS v,
4. G2claim AS r
5. WHERE r.G2Claim\_Status='pending' AND
6. r.G2Cust\_Id=c.G2Cust\_Id AND
7. c.G2Cust\_Id=v.G2Cust\_Id;

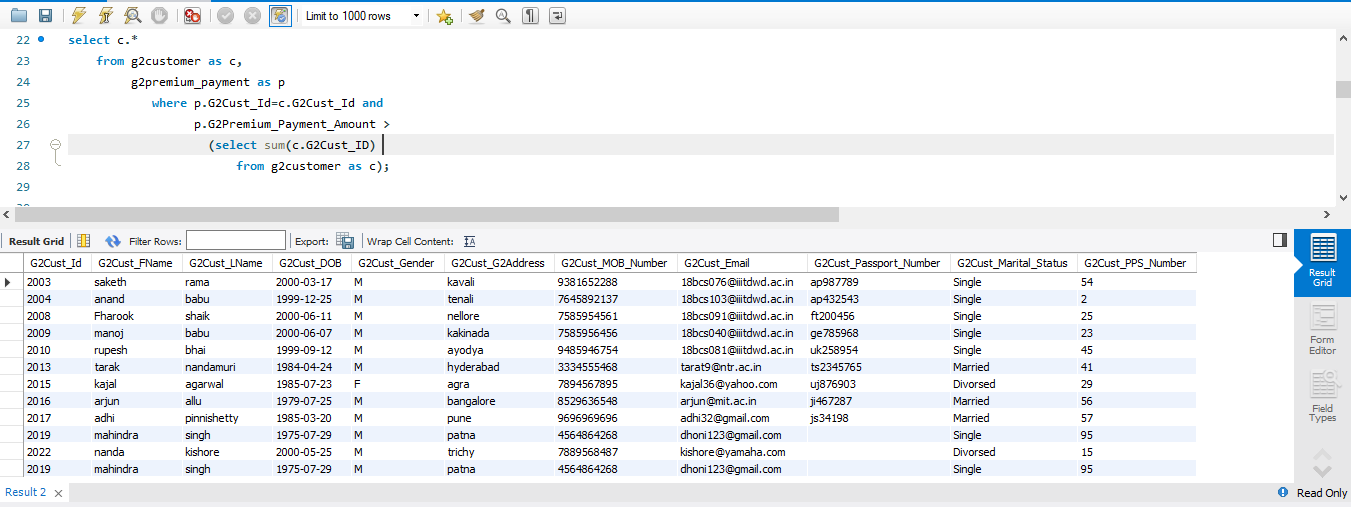
**** **Output:**

**Query -2:**

**Question:** Retrieve customer details who has premium payment amount greater than the sum of all the customer Id’s in the database.

**SQL Query:**

1. select c.\*
2. from g2customer as c,
3. g2premium\_payment as p
4. where p.G2Cust\_Id=c.G2Cust\_Id and
5. p.G2Premium\_Payment\_Amount >
6. (select sum(c.G2Cust\_ID)
7. from g2customer as c);

**** **Output:**

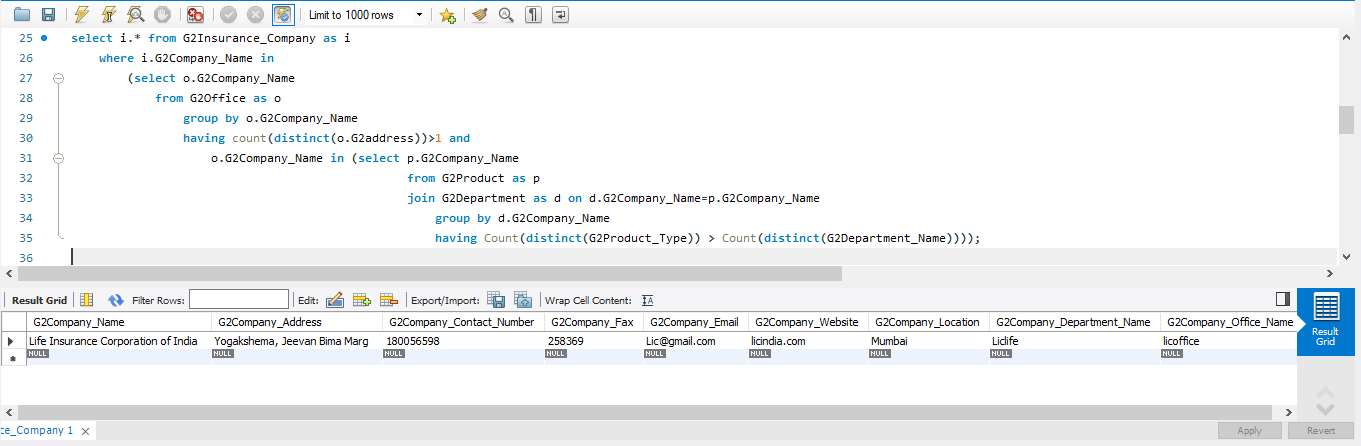
**Query -3:**

**Question:** Retrieve customer details who has premium payment amount greater than the sum of all the customer Id’s in the database.

**SQL Query:**

1. select i.\* from G2Insurance\_Company as i
2. where i.G2Company\_Name in
3. (select o.G2Company\_Name
4. from G2Office as o
5. group by o.G2Company\_Name
6. having count(distinct(o.G2address))>1 and
7. o.G2Company\_Name in (select p.G2Company\_Name
8. from G2Product as p
9. join G2Department as d on d.G2Company\_Name=p.G2Company\_Name
10. group by d.G2Company\_Name
11. having Count(distinct(G2Product\_Type)) >
12. Count(distinct(G2Department\_Name))));

**Output:**

****

**Query -4:**

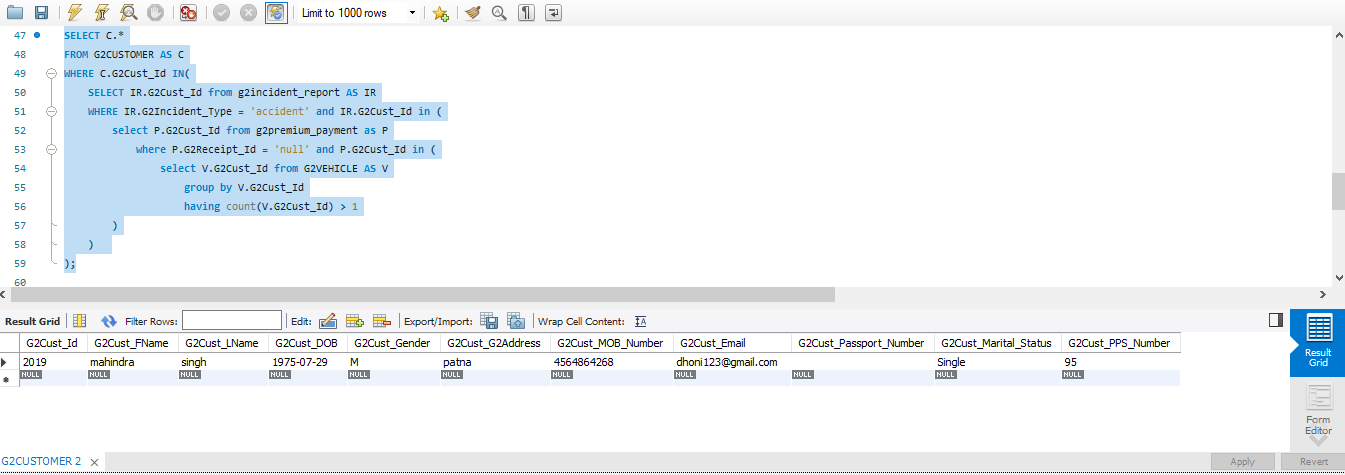
**Question:** Select Customers who have more than one Vehicle, where the premium for one of the

Vehicles is not paid and it is involved in accident

**Assumption:** In G2Premium\_Payment table, if the receipt id is null, then it means that the receipt isn't yet generated. So, the premium for one of the vehicles is not paid.

**SQL Query:**

1. SELECT C.\*
2. FROM G2CUSTOMER AS C
3. WHERE C.G2Cust\_Id IN(
4. SELECT IR.G2Cust\_Id from g2incident\_report AS IR
5. WHERE IR.G2Incident\_Type = 'accident' and IR.G2Cust\_Id in (
6. select P.G2Cust\_Id from g2premium\_payment as P
7. where P.G2Receipt\_Id = 'null' and P.G2Cust\_Id in (
8. select V.G2Cust\_Id from G2VEHICLE AS V
9. group by V.G2Cust\_Id
10. having count(V.G2Cust\_Id) > 1
11. )
12. )
13. );

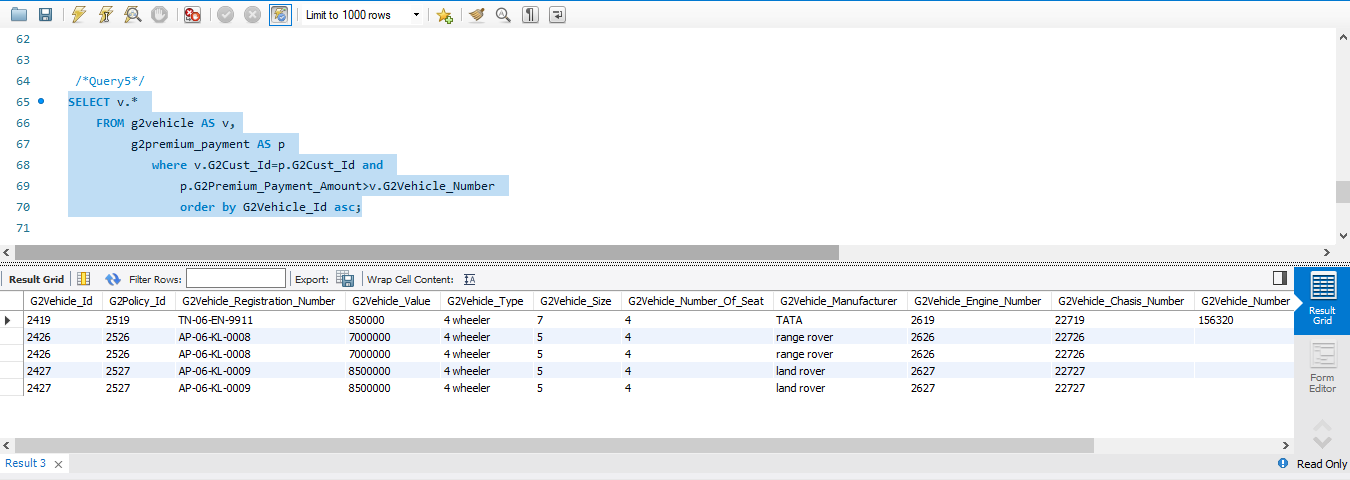
**** **Output:**

**Query -5:**

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

**SQL Query:**

1. SELECT v.\*
2. FROM g2vehicle AS v,
3. g2premium\_payment AS p
4. where v.G2Cust\_Id=p.G2Cust\_Id and
5. p.G2Premium\_Payment\_Amount>v.G2Vehicle\_Number
6. order by G2Vehicle\_Id asc;

**** **Output:**

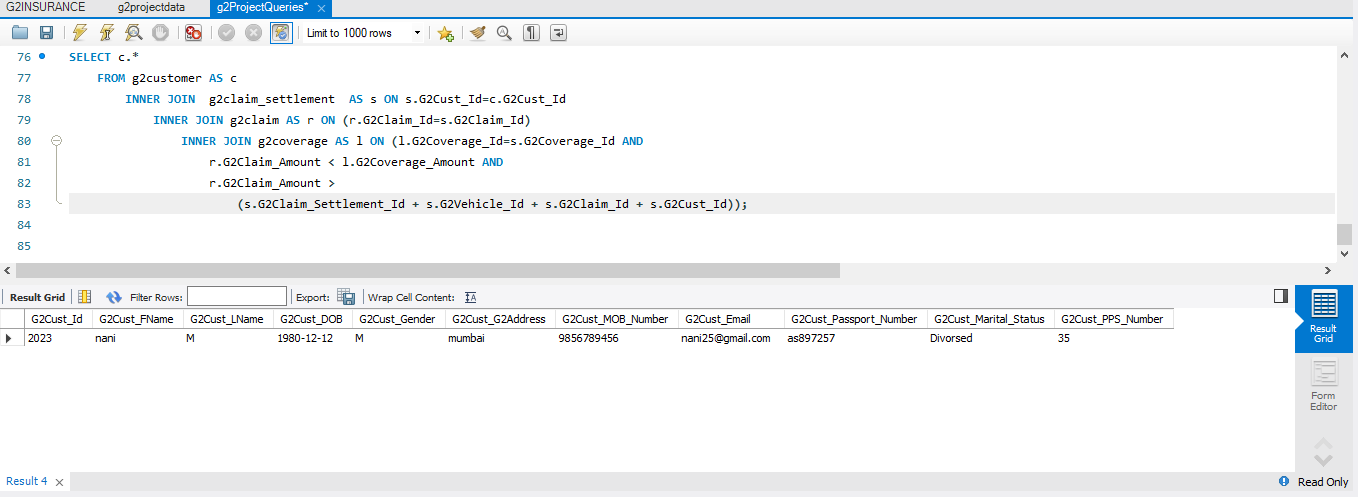
**Query -6:**

**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 c.\*
2. FROM g2customer AS c
3. INNER JOIN g2claim\_settlement AS s ON s.G2Cust\_Id=c.G2Cust\_Id
4. INNER JOIN g2claim AS r ON (r.G2Claim\_Id=s.G2Claim\_Id)
5. INNER JOIN g2coverage AS l ON (l.G2Coverage\_Id=s.G2Coverage\_Id AND
6. r.G2Claim\_Amount < l.G2Coverage\_Amount AND
7. r.G2Claim\_Amount >
8. (s.G2Claim\_Settlement\_Id + s.G2Vehicle\_Id + s.G2Claim\_Id + s.G2Cust\_Id));

****  **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. Please note that the python project implemented is still in development and it needs to be developed.

**Future Improvements:**

* Still more data will be supplied to the excel sheet.
* The Python code will be optimized for better execution.
* The project queries will be optimized & will be tested with loads of data.

Thank You

Group -2