



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

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
 - a. Project Title
 - b. Aim
 - c. Purpose
 - d. Scope
 - e. Project Benefits
 - f. A Quick 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. 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.

SL. No	Name	Roll No.	Email Address	Github Id	Major Role / Job Description
1	S. Sampath	18BCS087	18bcs087@iiitdwd.ac.in	Ssampath34	Data Collected & Optimized to Excel Data Collection & Optimized to Excel Helped in Data Collection & in Project Queries
2	Umesh Anand Babu	18BCS105	18bcs105@iiitdwd.ac.in	anand2517	
3	Yaganti Mokshith	18BCS112	18bcs112@iiitdwd.ac.in	mokshithyaganti	
4	Shaik Fharook	18BCS091	18bcs091@iiitdwd.ac.in	fharookshaik	Idea & Implementation of Python Code & Helped in Project Queries Helped in Implementation of Python Code & Maintained Report Helped in Implementation of Python Code & Maintained Report
5	Perumalla Tushar	18BCS065	18bcs065@iiitdwd.ac.in	tusharboruto	
6	Rupesh Kumar	18bcs081	18bcs081@iiitdwd.ac.in	rupesh-k-iiitdwd	
7	Pokala Dattatreya	18BCS067	18bcs067@iiitdwd.ac.in	dattu24	Maintained SQL Code & Project Queries Mantained CDM, LDM & Project Queries Maintained SQL Code & Project Queries Helped in Project Queries
8	Rama Dundi Saketh	18BCS076	18bcs076@iiitdwd.ac.in	DundiSaketh	
9	Vytla Harsha Vardhan	18BCS118	18bcs118@iiitdwd.ac.in	harsha-varadhan-vytla	
10	K Sai Manoj	18BCS040	18bcs040@iiitdwd.ac.in	saimanoj007	

3. Data Modelling

a. 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.

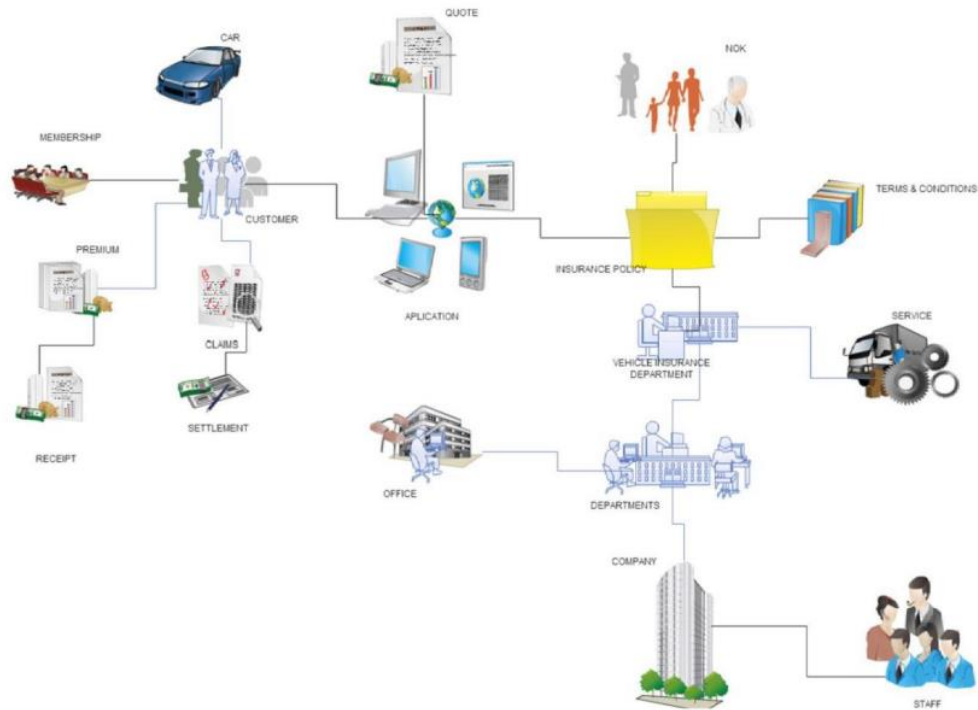
Table of Entity

Entity Type	Description
CUSTOMER	Records all the personal details about the customer
APPLICATION	Records details of the insurance cover requested by Customer
QUOTE	Records details of customer potential cost of the insurance product
INSURANCE POLICY	Records details of Insurance agreement
PREMIUM	Records details of customer payments
VEHICLE	Records details of Vehicle model, cost and registration
CLAIMS	Records details of customer claims in case of an incident
SETTLEMENTS	Records details of settlement made on claims
STAFF	Records details of employees
DEPARTMENT	Records details of the various departments
OFFICE	Records details of different office locations
MEMBERSHIP	Records details of customer membership
SERVICE	Records details of different car services offered
NOK	Records details of the next o kin
TERMS_CONDITIONS	Records all terms and conditions in regard to the policy
VEHICLE INSURANCE DEPARTMENT	Records details of vehicle insurance cover
RECEIPT	Records details of Receipt of Premiums
COMPANY	Details of the Insurance organization giving the insurance cover

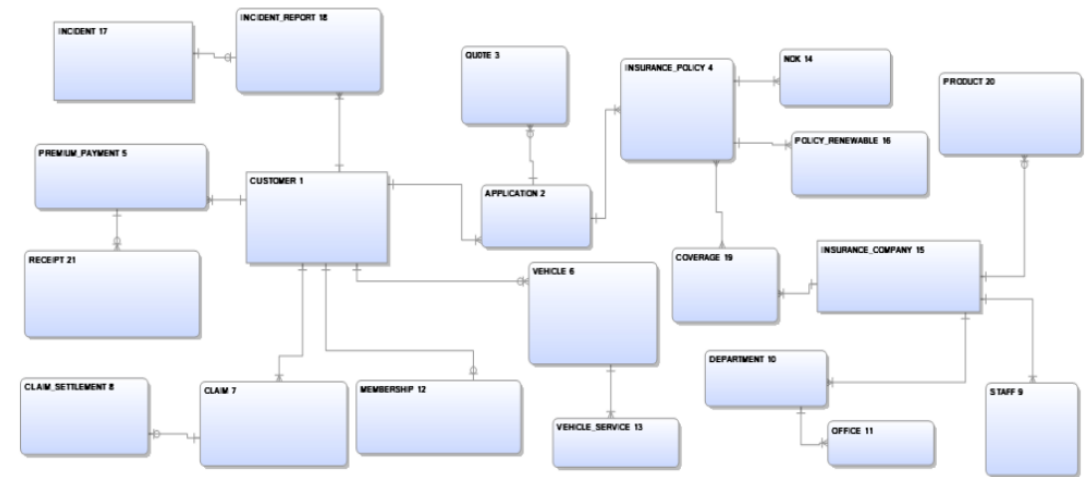
Relationships b/w Entities

Entity type	Related To Entities	Relationship
QUOTE	APPLICATION	one to one
APPLICATION	INSURANCE POLICY CUSTOMER	one to many one to many
CUSTOMER	MEMBERSHIP PREMIUM CLAIMS VEHICLE	many to many one to many one to many one to one, one to many
INSURANCE POLICY	VEHICLE INSURANCE DEPARTMENT TERM AND CONDITION NOK	one to many many to many one to many
PREMIUM	RECEIPT	one to many
CLAIMS	SETTLEMENT	one to one
VEHICLE INSURANCE DEPARTMENT	DEPARTMENT SERVICE	one to one, one to many one to many
DEPARTMENT	OFFICE COMPANY	many to many one to many
COMPANY	STAFF	many to many

Graphical Representation of Conceptual Data Model

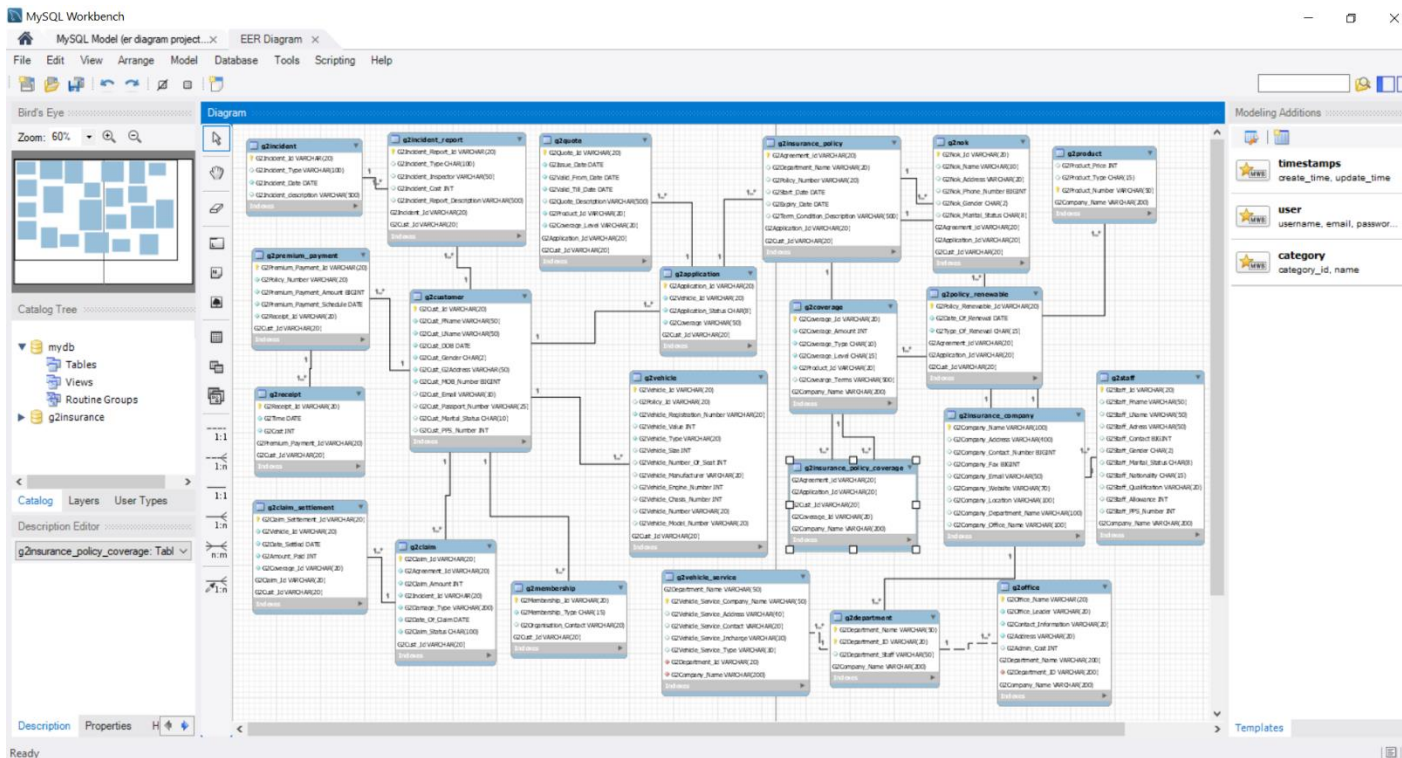


Conceptual data model of Car Insurance Database



b. 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)

c. 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.

2. **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.

- 3. 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.
- 4. 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.
- 5. 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")
    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 {t
ableName} 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.

```
# Connecting to database
conn = mysqlc.connect(
    host = "localhost",
    username = "root",
    password = "1106",
    database = "g2insurance"
)
```

- ❖ 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:

```
7 • SELECT c.*,v.*
8 FROM G2customer AS c,
9      G2vehicle AS v,
10     G2claim AS r
11     WHERE r.G2Claim_Status='pending' AND
12           r.G2Cust_Id=c.G2Cust_Id AND
13           c.G2Cust_Id=v.G2Cust_Id;
14
```

G2Cust_Id	G2Cust_FName	G2Cust_LName	G2Cust_DOB	G2Cust_Gender	G2Cust_G2Address	G2Cust_MOB_Number	G2Cust_Email	G2Cust_Passport_Number	G2Cust_Marital_Status	G2Cust_PPS_Number	G2Vehicle_Id	G2Claim_Id
2008	Fharook	shaik	2000-06-11	M	nellore	7585954561	18bcs091@iiitdwd.ac.in	ft200456	Single	25	2408	25
2017	adhi	pinnishetty	1985-03-20	M	pune	9696969696	adhi32@gmail.com	js34198	Married	57	2417	25
2003	saketh	rama	2000-03-17	M	kavali	9381652288	18bcs076@iiitdwd.ac.in	ap987789	Single	54	2403	25
2001	Tushar	boruto	2000-12-30	M	vizag	9848168581	18bcs065@iiitdwd.ac.in	Ap123456	Single	20	2401	25
2013	tarak	nandamuri	1984-04-24	M	hyderabad	3334555468	tarat9@ntr.ac.in	ts2345765	Married	41	2413	25
2017	adhi	pinnishetty	1985-03-20	M	pune	9696969696	adhi32@gmail.com	js34198	Married	57	2417	25
2005	mokshith	yaganthi	2000-08-14	M	rajamandry	9845671235	18bcs110@iiitdwd.ac.in	tn346753	Single	45	2405	25
2020	vijay	sethupati	1980-07-21	M	kanchi	7589648963	vijay@master.com		Single	65	2420	25
2020	vijay	sethupati	1980-07-21	M	kanchi	7589648963	vijay@master.com		Single	65	2427	25

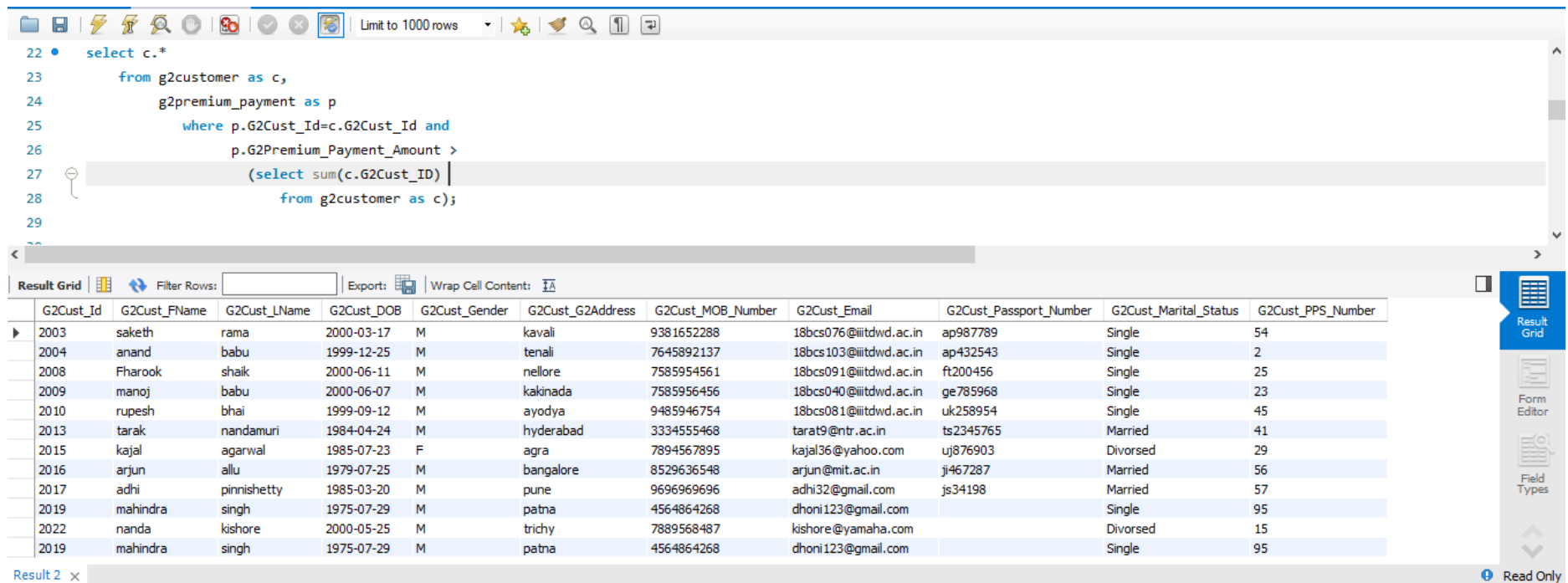
Query -2:

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:



The screenshot shows a database query tool interface. At the top, there's a toolbar with various icons. Below it, the SQL query is displayed in a text area. The query is:
select c.*
from g2customer as c,
g2premium_payment as p
where p.G2Cust_Id=c.G2Cust_Id and
p.G2Premium_Payment_Amount >
(select sum(c.G2Cust_ID)
from g2customer as c);
Below the query, there's a 'Result Grid' section. It shows a table with 12 columns: G2Cust_Id, G2Cust_FName, G2Cust_LName, G2Cust_DOB, G2Cust_Gender, G2Cust_G2Address, G2Cust_MOB_Number, G2Cust_Email, G2Cust_Passport_Number, G2Cust_Marital_Status, and G2Cust_PPS_Number. The table contains 15 rows of data. On the right side of the interface, there are buttons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom right, there's a 'Read Only' indicator.

	G2Cust_Id	G2Cust_FName	G2Cust_LName	G2Cust_DOB	G2Cust_Gender	G2Cust_G2Address	G2Cust_MOB_Number	G2Cust_Email	G2Cust_Passport_Number	G2Cust_Marital_Status	G2Cust_PPS_Number
▶	2003	saketh	rama	2000-03-17	M	kavali	9381652288	18bcs076@iitdwd.ac.in	ap987789	Single	54
	2004	anand	babu	1999-12-25	M	tenali	7645892137	18bcs103@iitdwd.ac.in	ap432543	Single	2
	2008	Fharook	shaik	2000-06-11	M	nellore	7585954561	18bcs091@iitdwd.ac.in	ft200456	Single	25
	2009	manoj	babu	2000-06-07	M	kakinada	7585956456	18bcs040@iitdwd.ac.in	ge785968	Single	23
	2010	rupesh	bhai	1999-09-12	M	ayodya	9485946754	18bcs081@iitdwd.ac.in	uk258954	Single	45
	2013	tarak	nandamuri	1984-04-24	M	hyderabad	3334555468	tarat9@ntr.ac.in	ts2345765	Married	41
	2015	kajal	agarwal	1985-07-23	F	agra	7894567895	kajal36@yahoo.com	uj876903	Divorsed	29
	2016	arjun	allu	1979-07-25	M	bangalore	8529636548	arjun@mit.ac.in	ji467287	Married	56
	2017	adhi	pinnishetty	1985-03-20	M	pune	9696969696	adhi32@gmail.com	js34198	Married	57
	2019	mahindra	singh	1975-07-29	M	patna	4564864268	dhoni123@gmail.com		Single	95
	2022	nanda	kishore	2000-05-25	M	trichy	7889568487	kishore@yamaha.com		Divorsed	15
	2019	mahindra	singh	1975-07-29	M	patna	4564864268	dhoni123@gmail.com		Single	95

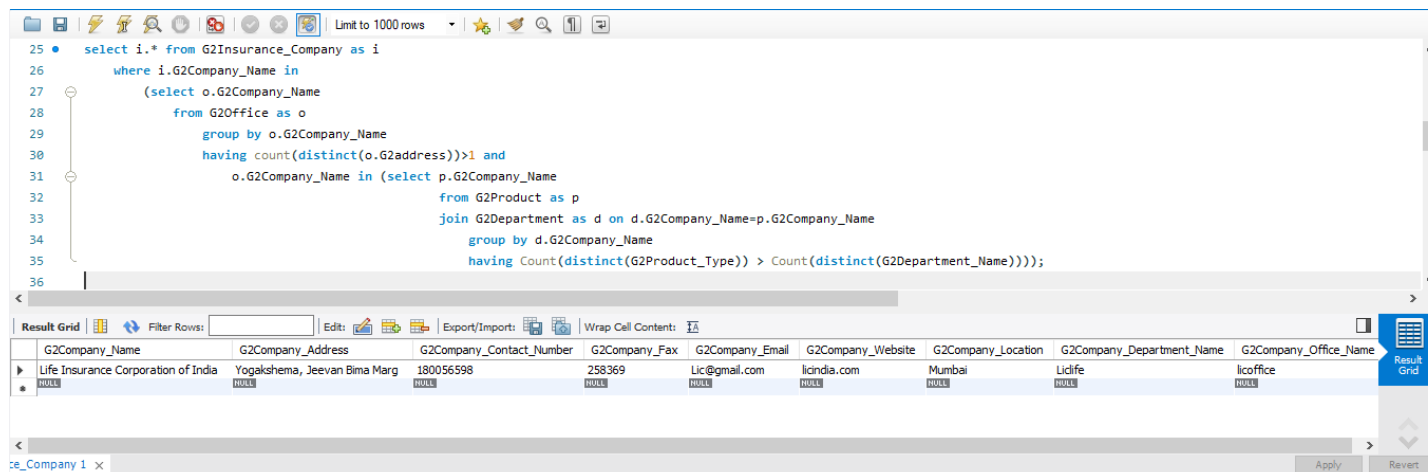
Query -3:

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:



The screenshot shows a database query editor with the SQL query pasted into the editor area. Below the editor, the 'Result Grid' is displayed, showing the output of the query. The result grid has columns for G2Company_Name, G2Company_Address, G2Company_Contact_Number, G2Company_Fax, G2Company_Email, G2Company_Website, G2Company_Location, G2Company_Department_Name, and G2Company_Office_Name. The first row of data is for 'Life Insurance Corporation of India'.

G2Company_Name	G2Company_Address	G2Company_Contact_Number	G2Company_Fax	G2Company_Email	G2Company_Website	G2Company_Location	G2Company_Department_Name	G2Company_Office_Name
Life Insurance Corporation of India	Yogakshema, Jeevan Bima Marg	180056598	258369	Lic@gmail.com	licindia.com	Mumbai	Lidife	licoffice

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:

The screenshot shows a database query editor with a SQL query and its result grid. The query is as follows:

```
47 SELECT C.*
48 FROM G2CUSTOMER AS C
49 WHERE C.G2Cust_Id IN(
50     SELECT IR.G2Cust_Id from g2incident_report AS IR
51     WHERE IR.G2Incident_Type = 'accident' and IR.G2Cust_Id in (
52         select P.G2Cust_Id from g2premium_payment as P
53         where P.G2Receipt_Id = 'null' and P.G2Cust_Id in (
54             select V.G2Cust_Id from G2VEHICLE AS V
55             group by V.G2Cust_Id
56             having count(V.G2Cust_Id) > 1
57         )
58     )
59 );
60
```

The result grid shows the following data:

G2Cust_Id	G2Cust_FName	G2Cust_LName	G2Cust_DOB	G2Cust_Gender	G2Cust_G2Address	G2Cust_MOB_Number	G2Cust_Email	G2Cust_Passport_Number	G2Cust_Marital_Status	G2Cust_PPS_Number
2019	mahindra	singh	1975-07-29	M	patna	4564864268	dhoni123@gmail.com		Single	95
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Result Grid' button. The bottom status bar shows 'G2CUSTOMER 2' and 'Apply Revert' buttons.

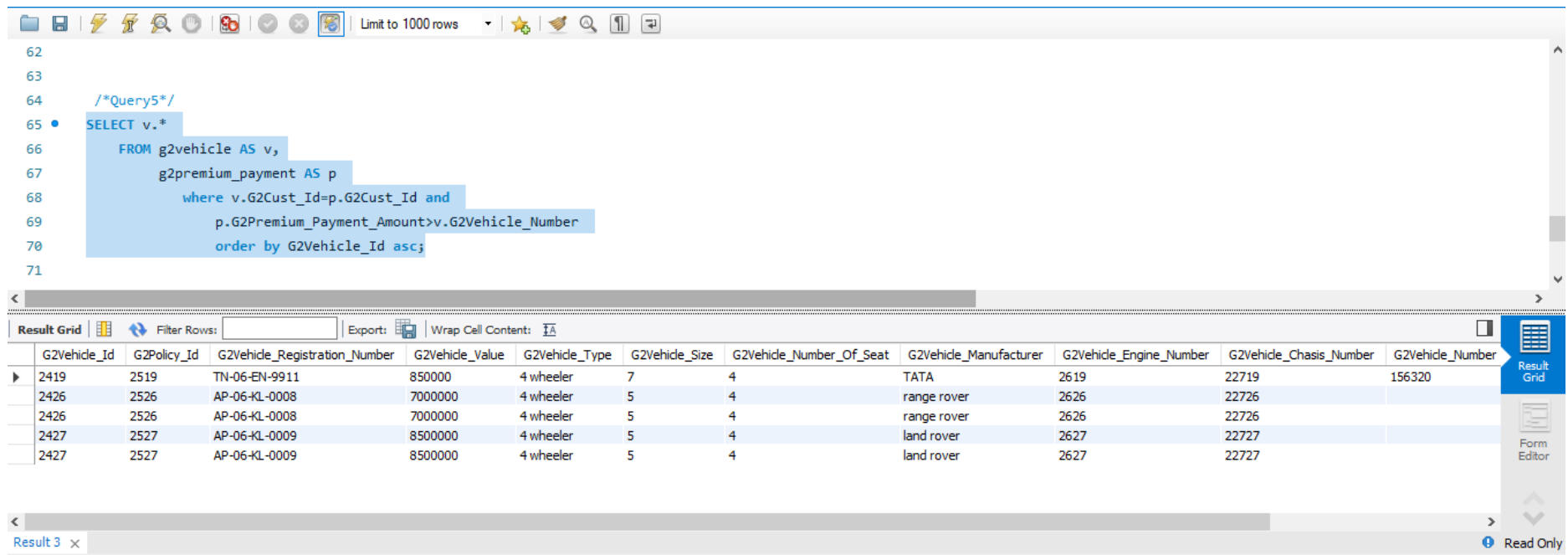
Query -5:

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:



The screenshot shows a database query editor interface. The top part displays the SQL query being executed. Below the query, the results are shown in a table format. The table has 11 columns: G2Vehicle_Id, G2Policy_Id, G2Vehicle_Registration_Number, G2Vehicle_Value, G2Vehicle_Type, G2Vehicle_Size, G2Vehicle_Number_Of_Seat, G2Vehicle_Manufacturer, G2Vehicle_Engine_Number, G2Vehicle_Chassis_Number, and G2Vehicle_Number. The results are sorted by G2Vehicle_Id in ascending order.

G2Vehicle_Id	G2Policy_Id	G2Vehicle_Registration_Number	G2Vehicle_Value	G2Vehicle_Type	G2Vehicle_Size	G2Vehicle_Number_Of_Seat	G2Vehicle_Manufacturer	G2Vehicle_Engine_Number	G2Vehicle_Chassis_Number	G2Vehicle_Number
2419	2519	TN-06-EN-9911	850000	4 wheeler	7	4	TATA	2619	22719	156320
2426	2526	AP-06-KL-0008	7000000	4 wheeler	5	4	range rover	2626	22726	
2426	2526	AP-06-KL-0008	7000000	4 wheeler	5	4	range rover	2626	22726	
2427	2527	AP-06-KL-0009	8500000	4 wheeler	5	4	land rover	2627	22727	
2427	2527	AP-06-KL-0009	8500000	4 wheeler	5	4	land rover	2627	22727	

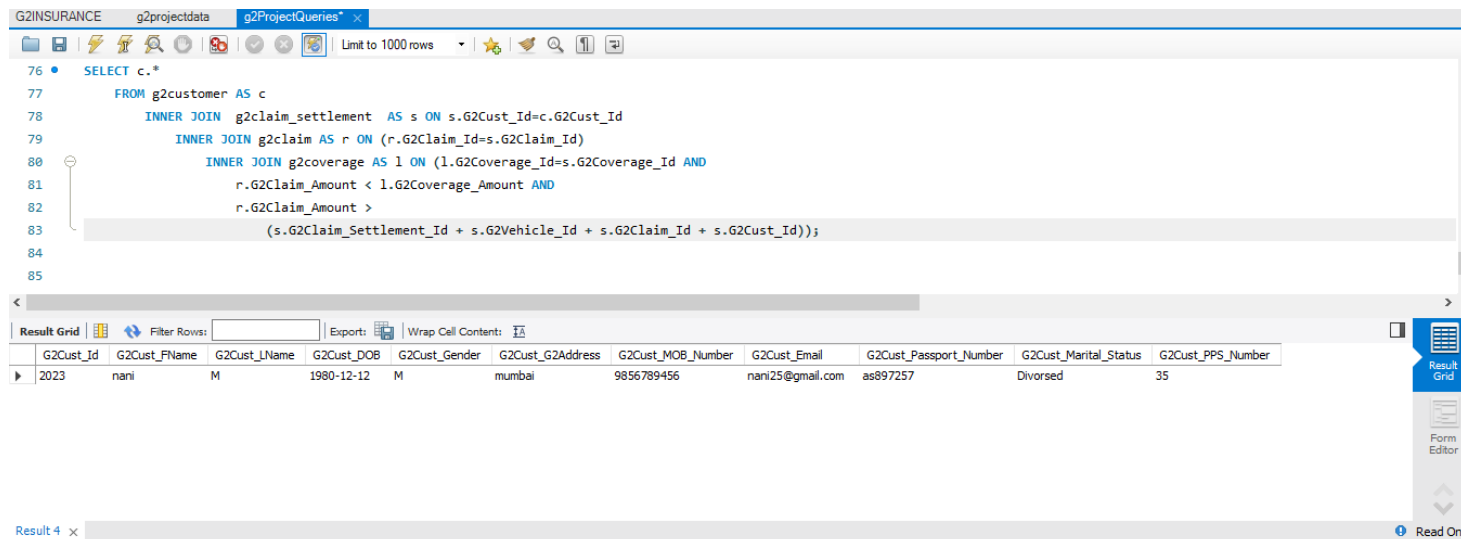
Query -6:

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:



The screenshot shows a database query tool interface. The top toolbar includes icons for file operations, query execution, and result viewing. The query editor displays the SQL query from the previous block. Below the editor, the 'Result Grid' tab is active, showing a single row of data for a customer named 'nani'.

G2Cust_Id	G2Cust_FName	G2Cust_LName	G2Cust_DOB	G2Cust_Gender	G2Cust_G2Address	G2Cust_MOB_Number	G2Cust_Email	G2Cust_Passport_Number	G2Cust_Marital_Status	G2Cust_PPS_Number
2023	nani	M	1980-12-12	M	mumbai	9856789456	nani25@gmail.com	as897257	Divorced	35

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