Exp: 1	Problem Identification

Aim: To identify team's project title and its problem statement.

Insurance Management System

It is required to build an insurance management system to automate various insurance-related processes and operations. The current manual system needs a centralized database that leads to data redundancy and consistency, which results in delays and errors, lacks in managing customer information and interactions, and leads to slow processing and time-consuming. The system should allow for efficient storage, retrieval, and manipulation of data to streamline insurance-related processes and enhance better customer service.

Result: Project title and problem statement is identified.

Exp: 2	Requirement Gathering

Aim: To work on collecting project requirements.

Requirement gathering – Insurance Management System

i. Stakeholder Identification:

- Identify the key stakeholders involved in the insurance processes.

ii. Use Cases:

- Develop detailed use cases to describe various scenarios, interactions, and workflows within the insurance management system. These use cases help to understand the functional requirements of the system.

iii. Data Requirements:

- Determine the types of data that need to be stored, such as policy data, customer data, claims data, and financial data.

iv. Policy Management:

- Consider aspects like policy types, coverage options, premium calculation, and renewal processes.

v. Claims Processing:

- Document requirements related to claims submission, processing, and settlement. This includes the ability to report a claim, track its status, and calculate claim payouts.

vi. Customer Management:

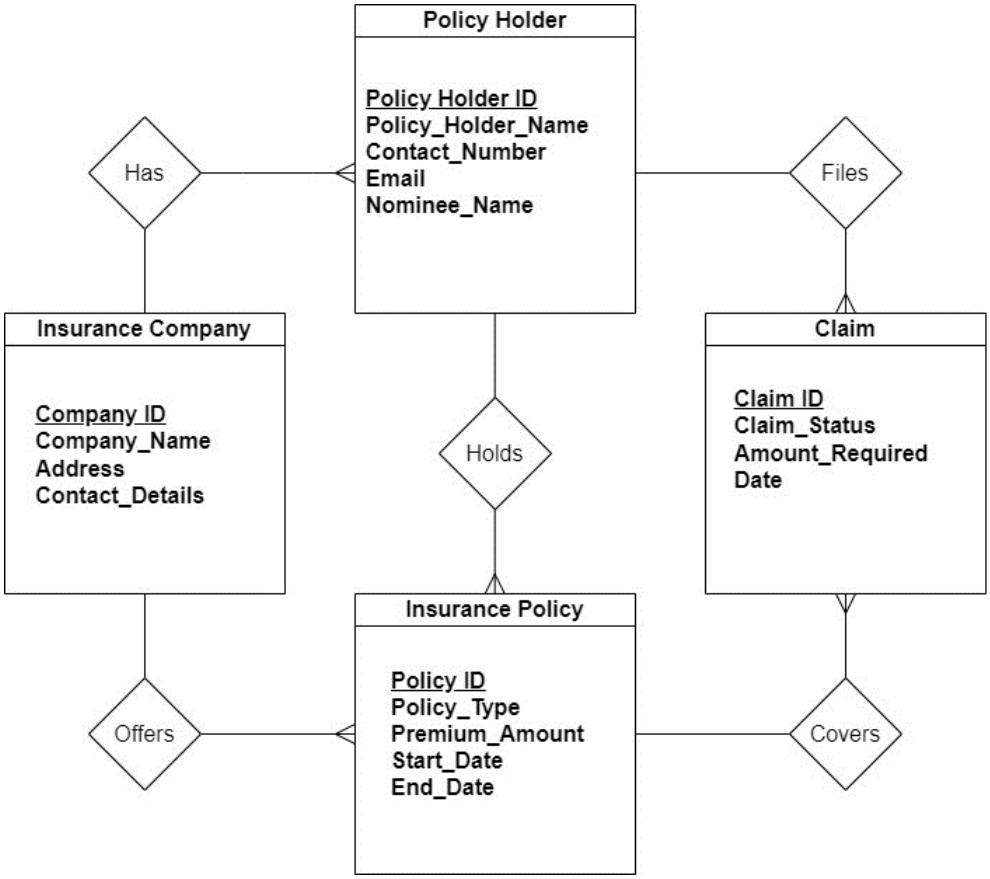
- Define the requirements for managing customer information, including customer communication, and support. Consider features like customer portals for policy management.

Result: Gathering project requirements is completed.

Exp: 3	Design Using ER Diagram

Aim: To work on project designing using ER diagram.

ER Diagram – Insurance Management System.



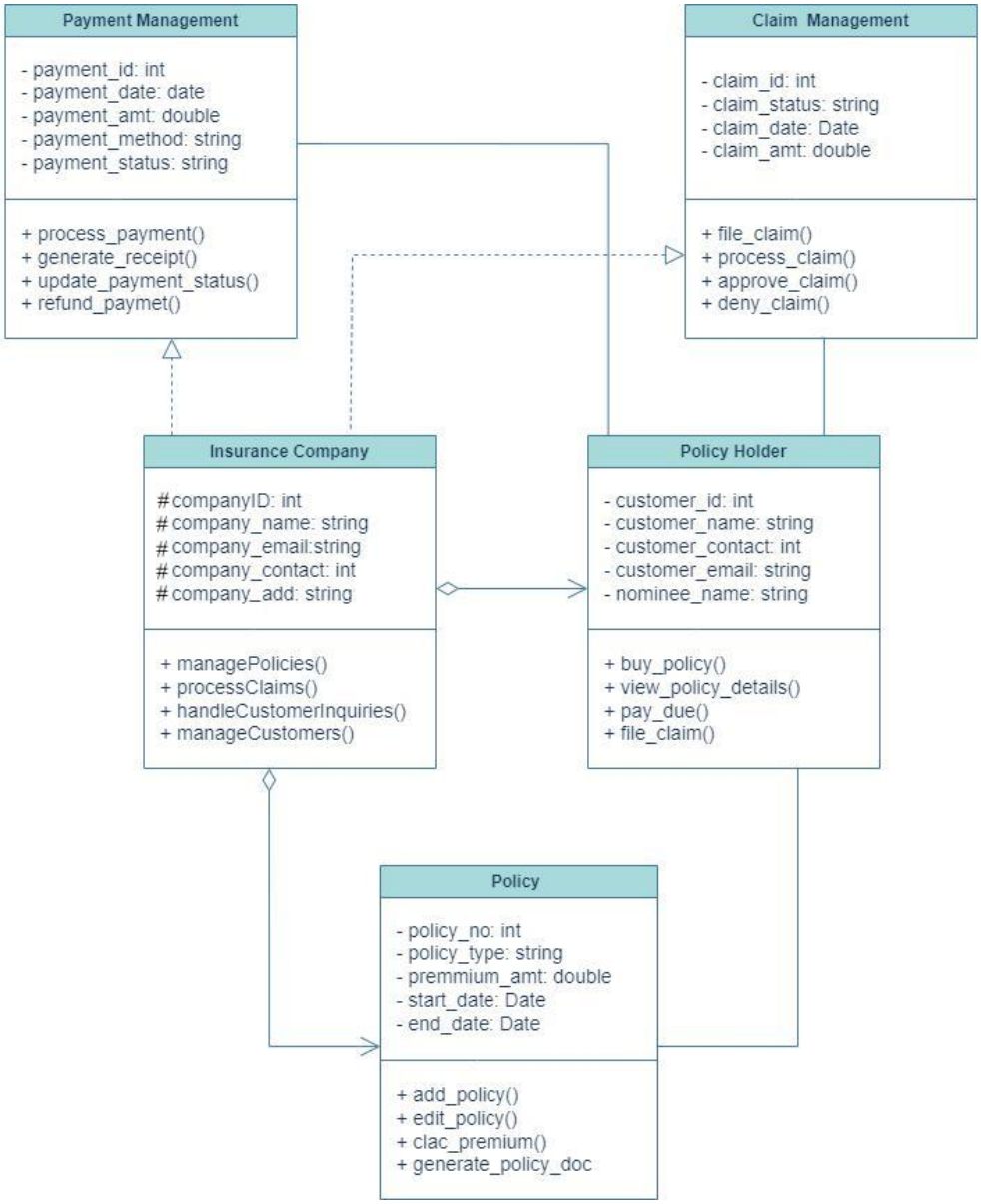
Result: To work on project designing using ER Diagram is successfully completed.

Exp: 4	Design Using UML Diagrams

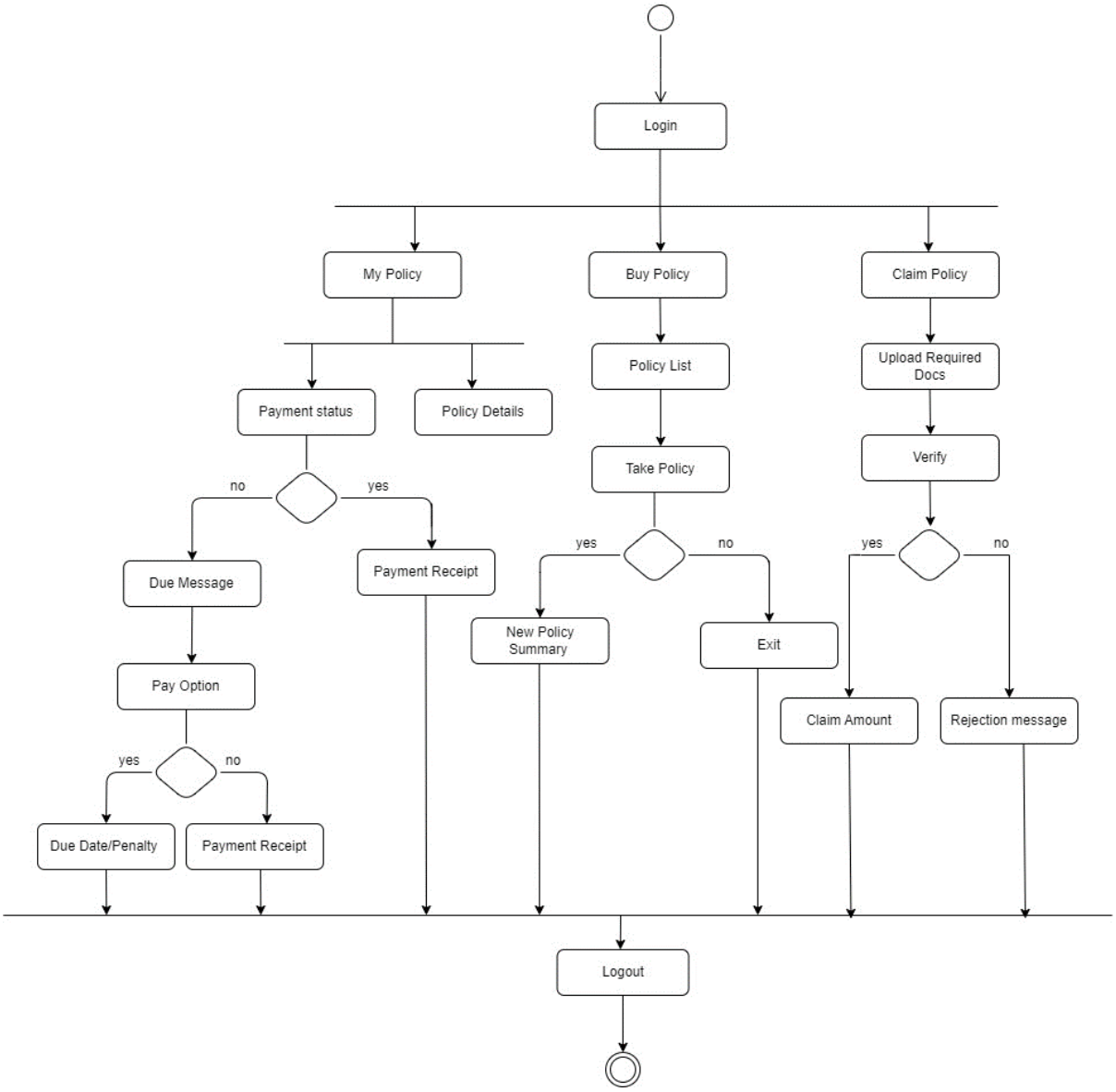
Aim: To work on project designing using ULM diagrams.

UML Diagram - Insurance Management System.

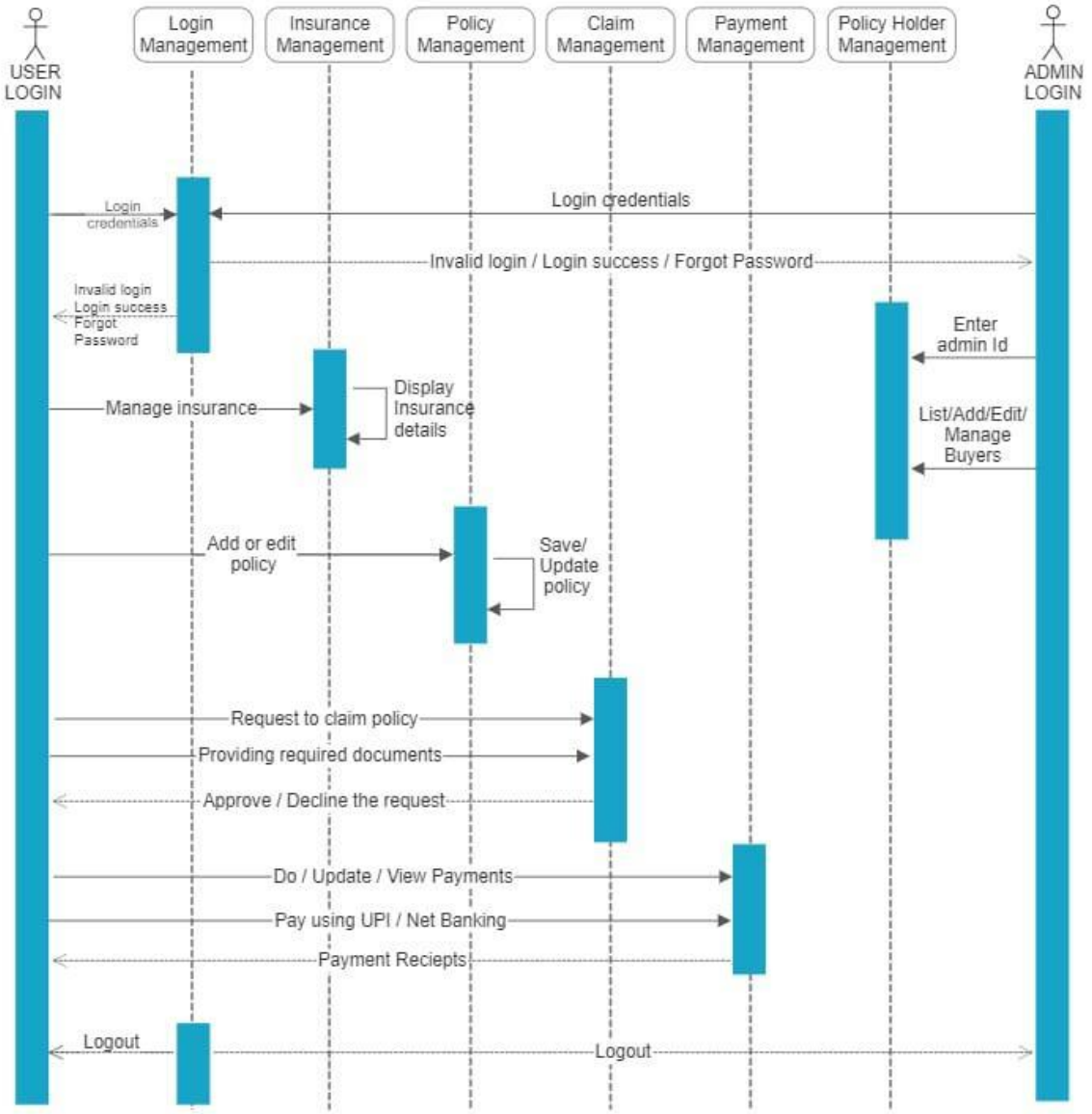
i. Class Diagram



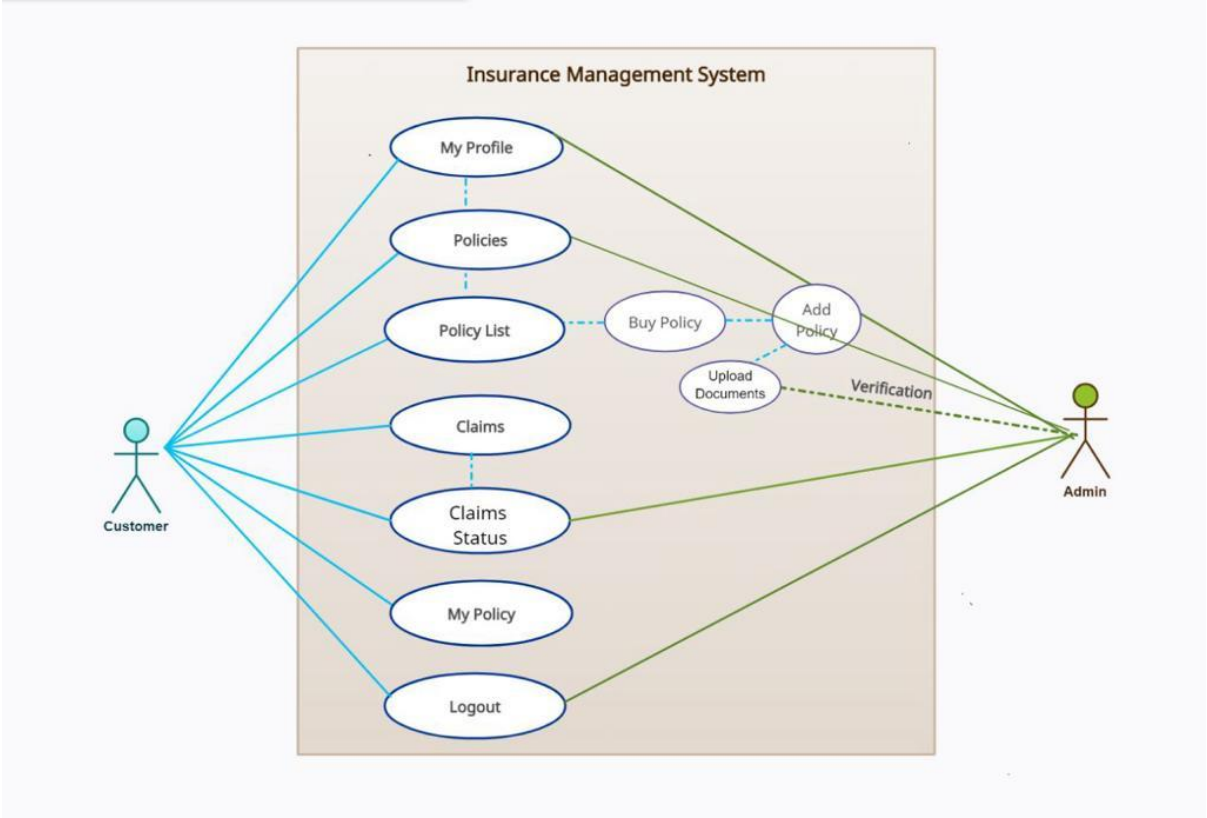
ii. Activity Diagram



iii. Sequence Diagram



iv. Use-Case Diagram



Result: To work on project designing using UML diagrams is successfully completed.

Exp: 5	Working with DDL and DML Commands

Aim: To work with DDL & DML commands.

1. **DDL COMMANDS:**

a) **CREATE**: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).

```
mysql> CREATE TABLE employee_details(EmpID int PRIMARY KEY, Name varchar(20), Ph_No int, City varchar(20));
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> desc employee_details;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| EmpID | int         | NO   | PRI | NULL    |       |
| Name  | varchar(20) | YES  |     | NULL    |       |
| Ph_No | int         | YES  |     | NULL    |       |
| City  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

b) **DROP:** This command is used to delete objects from the database.

```
mysql> desc employee_details;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| EmpID | int         | NO   | PRI | NULL    |       |
| Name  | varchar(20) | YES  |     | NULL    |       |
| Ph_No | int         | YES  |     | NULL    |       |
| City  | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> drop employee_details;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
 for the right syntax to use near 'employee_details' at line 1
mysql> drop table employee_details;
Query OK, 0 rows affected (0.01 sec)

mysql> desc employee_details;
ERROR 1146 (42S02): Table 'db1.employee_details' doesn't exist
mysql>
```

c) **ALTER:** This is used to alter the structure of the database.

```
mysql> ALTER TABLE employee_details ADD Email varchar(20);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

d) **TRUNCATE:** This is used to remove all records from a table, including all spaces allocated for the records are removed.

```
mysql> SELECT * from employee_details;
+-------+--------+-----------+---------+----------------------+
| EmpID | Name   | Ph_No     | City    | Email                |
+-------+--------+-----------+---------+----------------------+
|     1 | Saran  |  29381039 | Chennai | ba@snuchennai.edu.in |
|     2 | Suriya | 295481012 | Chennai | suriya24@gmail.com   |
+-------+--------+-----------+---------+----------------------+
2 rows in set (0.00 sec)

mysql> TRUNCATE TABLE employee_details;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * from employee_details;
Empty set (0.00 sec)
```

2. DML COMMANDS:

a) INSERT: It is used to insert data into a table.

```
mysql> insert into employee_details Values(2,"Suriya",295481012,"Chennai","suriya24@gmail.com");
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * from employee_details;
+-------+--------+-----------+---------+----------------------+
| EmpID | Name   | Ph_No     | City    | Email                |
+-------+--------+-----------+---------+----------------------+
|     1 | Saran  |  29381039 | Chennai | ba@snuchennai.edu.in |
|     2 | Suriya | 295481012 | Chennai | suriya24@gmail.com   |
+-------+--------+-----------+---------+----------------------+
2 rows in set (0.00 sec)
```

b) **UPDATE:** It is used to update existing data within a table.

```
mysql> UPDATE employee_details set Email="ba@snuchennai.edu.in" where EmpID=1;
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

c) **DELETE:** It is used to delete records from a database table.

```
mysql> DELETE FROM employee_details where EmpID=2;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * from employee_details;
+-------+-------+----------+---------+--------------------+
| EmpID | Name  | Ph_No    | City    | Email              |
+-------+-------+----------+---------+--------------------+
|     1 | Saran | 29381039 | Chennai | saran234@gmail.com |
+-------+-------+----------+---------+--------------------+
1 row in set (0.00 sec)
```

Result: Thus, working with DDL & DML commands on MySQL is successfully verified.

Exp: 6	Working With Join, Set Operations and Aggregate Functions

Aim: To perform join, set and aggregate functions on MySQL.

SET OPERATIONS

Tables

```
mysql> select
    -> * from l1;
+----+--------+------+------+
| ID | Name   | Age  | Dept |
+----+--------+------+------+
|  1 | Aditi  |   19 | CBS  |
|  2 | Bhavan |   18 | AIDS |
|  3 | Cherry |   19 | AIDS |
|  4 | Devi   |   19 | IoT  |
|  5 | Emma   |   17 | CBS  |
+----+--------+------+------+
5 rows in set (0.00 sec)

mysql> select * from l2;
+----+-------------+------+------+
| ID | Name        | Age  | Dept |
+----+-------------+------+------+
|  1 | Fasil       |   20 | IoT  |
|  3 | Cherry      |   19 | AIDS |
|  5 | Guru Brahma |   19 | AIDS |
+----+-------------+------+------+
3 rows in set (0.00 sec)
```

Union

```
mysql> select * from l1 union select * from l2;
+----+-------------+------+------+
| ID | Name        | Age  | Dept |
+----+-------------+------+------+
|  1 | Aditi       |   19 | CBS  |
|  2 | Bhavan      |   18 | AIDS |
|  3 | Cherry      |   19 | AIDS |
|  4 | Devi        |   19 | IoT  |
|  5 | Emma        |   17 | CBS  |
|  1 | Fasil       |   20 | IoT  |
|  5 | Guru Brahma |   19 | AIDS |
+----+-------------+------+------+
7 rows in set (0.00 sec)
```

Union All

```
mysql> select * from l1 union all select * from l2;
+----+-------------+------+------+
| ID | Name        | Age  | Dept |
+----+-------------+------+------+
|  1 | Aditi       |   19 | CBS  |
|  2 | Bhavan      |   18 | AIDS |
|  3 | Cherry      |   19 | AIDS |
|  4 | Devi        |   19 | IoT  |
|  5 | Emma        |   17 | CBS  |
|  1 | Fasil       |   20 | IoT  |
|  3 | Cherry      |   19 | AIDS |
|  5 | Guru Brahma |   19 | AIDS |
+----+-------------+------+------+
8 rows in set (0.00 sec)
```

Intersect

```
mysql> select * from l1 intersect select * from l2;
+----+--------+------+------+
| ID | Name   | Age  | Dept |
+----+--------+------+------+
|  3 | Cherry |   19 | AIDS |
+----+--------+------+------+
1 row in set (0.00 sec)

mysql>
```

Minus

```
mysql> select * from l1 except select * from l2;
+----+--------+------+------+
| ID | Name   | Age  | Dept |
+----+--------+------+------+
|  1 | Aditi  |   19 | CBS  |
|  2 | Bhavan |   18 | AIDS |
|  4 | Devi   |   19 | IoT  |
|  5 | Emma   |   17 | CBS  |
+----+--------+------+------+
4 rows in set (0.00 sec)
```

AGGREGATE FUNCTIONS

```
mysql> select min(Intern_salary) from l2;
+--------------------+
| min(Intern_salary) |
+--------------------+
|               4500 |
+--------------------+
1 row in set (0.00 sec)

mysql> select max(Intern_salary) from l2;
+--------------------+
| max(Intern_salary) |
+--------------------+
|              17000 |
+--------------------+
1 row in set (0.00 sec)

mysql> select avg(Intern_salary) from l2;
+--------------------+
| avg(Intern_salary) |
+--------------------+
|         10833.3333 |
+--------------------+
1 row in set (0.00 sec)

mysql> select sum(Intern_salary) from l2;
+--------------------+
| sum(Intern_salary) |
+--------------------+
|              32500 |
+--------------------+
1 row in set (0.00 sec)

mysql> select count(Intern_salary) from l2;
+----------------------+
| count(Intern_salary) |
+----------------------+
|                    3 |
+----------------------+
1 row in set (0.00 sec)
```

JOIN OPERATIONS

Tables

```
mysql> select * from h1;
+------+--------+
| ID   | Name   |
+------+--------+
|    1 | Aditi  |
|    2 | Rahul  |
|    3 | Meera  |
|    4 | Lokesh |
|    5 | Sneha  |
+------+--------+
5 rows in set (0.00 sec)

mysql> select * from h2;
+---------+------+--------+
| Name    | DEPT | Salary |
+---------+------+--------+
| Rahul   | Iot  |  27000 |
| Sneha   | CBS  |  12000 |
| Gautham | AIDS |  25000 |
| Meera   | CBS  |  14000 |
| Lokesh  | IoT  |  50000 |
| Dravid  | AIDS |  40000 |
+---------+------+--------+
6 rows in set (0.00 sec)
```

Natural join

```
mysql> select * from h1 natural join h2;
+--------+------+------+--------+
| Name   | ID   | DEPT | Salary |
+--------+------+------+--------+
| Rahul  |    2 | Iot  |  27000 |
| Sneha  |    5 | CBS  |  12000 |
| Meera  |    3 | CBS  |  14000 |
| Lokesh |    4 | IoT  |  50000 |
+--------+------+------+--------+
4 rows in set (0.00 sec)
```

Inner join

```
mysql> select h1.ID, h1.Name, h2.DEPT, h2.Salary from h1 inner join h2 on h1.Name= h2.Name;
+------+--------+------+--------+
| ID   | Name   | DEPT | Salary |
+------+--------+------+--------+
|    2 | Rahul  | Iot  |  27000 |
|    5 | Sneha  | CBS  |  12000 |
|    3 | Meera  | CBS  |  14000 |
|    4 | Lokesh | IoT  |  50000 |
+------+--------+------+--------+
4 rows in set (0.00 sec)
```

Left join

```
mysql> select * from h1 left join h2 on h1.Name=h2.Name;
+------+--------+--------+------+--------+
| ID   | Name   | Name   | DEPT | Salary |
+------+--------+--------+------+--------+
|    1 | Aditi  | NULL   | NULL |   NULL |
|    2 | Rahul  | Rahul  | Iot  |  27000 |
|    3 | Meera  | Meera  | CBS  |  14000 |
|    4 | Lokesh | Lokesh | IoT  |  50000 |
|    5 | Sneha  | Sneha  | CBS  |  12000 |
+------+--------+--------+------+--------+
5 rows in set (0.00 sec)
```

Right join

```
mysql> select * from h1 right join h2 on h1.Name=h2.Name;
+------+--------+---------+------+--------+
| ID   | Name   | Name    | DEPT | Salary |
+------+--------+---------+------+--------+
|    2 | Rahul  | Rahul   | Iot  |  27000 |
|    5 | Sneha  | Sneha   | CBS  |  12000 |
| NULL | NULL   | Gautham | AIDS |  25000 |
|    3 | Meera  | Meera   | CBS  |  14000 |
|    4 | Lokesh | Lokesh  | IoT  |  50000 |
| NULL | NULL   | Dravid  | AIDS |  40000 |
+------+--------+---------+------+--------+
6 rows in set (0.00 sec)
```

Full join

```
mysql> select h1.Name, h2.Name, h2.DEPT from h1 left join h2 on h1.Name=h2.Name union select h1.Name, h2.Name, h2.DEPT from h1 right join h2 on h1.Name=h2.Name;
+--------+---------+------+
| Name   | Name    | DEPT |
+--------+---------+------+
| Aditi  | NULL    | NULL |
| Rahul  | Rahul   | Iot  |
| Meera  | Meera   | CBS  |
| Lokesh | Lokesh  | IoT  |
| Sneha  | Sneha   | CBS  |
| NULL   | Gautham | AIDS |
| NULL   | Dravid  | AIDS |
+--------+---------+------+
7 rows in set (0.00 sec)
```

Result: Thus, working with Join, Set operations and Aggregate functions on MySQL is successfully verified.

Exp: 7	Working With Queries Related to Project

Aim: To work with queries related to project on MySQL.

Insurance Management System – Data Base.

- TABLES

```
mysql> show tables;
+-------------------+
| Tables_in_dbms    |
+-------------------+
| claim             |
| insurance_company |
| insurance_policy  |
| policy_holder     |
+-------------------+
```

- Claim

```
mysql> select * from claim;
+---------+--------------+-----------------+------------+
| ClaimID | Claim_Status | Amount_Required | Date       |
+---------+--------------+-----------------+------------+
|    1122 | Denied       |           600.9 | 2023-10-25 |
|    1234 | Approved     |           26000 | 2023-08-21 |
|    1235 | Pending      |           80000 | 2023-10-01 |
|    2345 | Denied       |          300.75 | 2023-10-05 |
|    3344 | Approved     |          1700.4 | 2023-10-28 |
|    3456 | Pending      |          1200.8 | 2023-10-12 |
|    4567 | Approved     |          1800.6 | 2023-10-18 |
|    5678 | Approved     |           15000 | 2023-10-01 |
|    6789 | Approved     |         2000.25 | 2023-10-08 |
|    7890 | Denied       |           450.2 | 2023-10-15 |
|    8901 | Pending      |           950.3 | 2023-10-22 |
+---------+--------------+-----------------+------------+
11 rows in set (0.00 sec)
```

- Insurance_Company

```
mysql> select * from insurance_company;
+------------+--------------------+-----------+-----------------+
| Company_ID | Company_Name       | Address   | Contact_Details |
+------------+--------------------+-----------+-----------------+
|       1234 | TATA AIG           | Chennai   |       912738901 |
|       1235 | HDFC Ergo          | Mumbai    |       987654321 |
|       1357 | USAA               | Kolkata   |      1098765432 |
|       2345 | New India          | Bangalore |       765432109 |
|       2468 | Bharti AXA         | Lucknow   |       987654321 |
|       3456 | United India       | Hyderabad |       543210987 |
|       4567 | Bajaj Allianz      | Ahmedabad |       321098765 |
|       5678 | ICICI Lombard      | Delhi     |       876543210 |
|       6789 | Oriental           | Chennai   |       654321098 |
|       7890 | Reliance           | Kolkata   |       432109876 |
|       8901 | Travelers Insurance | Bengaluru |      2109876543 |
|       8902 | SBI General        | Pune      |       210987654 |
+------------+--------------------+-----------+-----------------+
12 rows in set (0.00 sec)
```

- Insurance_policy

```
mysql> select * from insurance_policy;
+----------+-----------------------+----------------+------------+------------+
| PolicyID | Policy_Type           | Premium_Amount | Start_Date | End_Date   |
+----------+-----------------------+----------------+------------+------------+
|     1234 | General               |          10000 | 2022-10-21 | 2023-08-21 |
|     1498 | Life Insurance        |          20000 | 2023-10-01 | 2024-09-30 |
|     2156 | Health                |          12000 | 2023-11-01 | 2024-10-31 |
|     3098 | Pet Insurance         |          10000 | 2023-09-20 | 2024-09-19 |
|     3874 | Auto Insurance        |          80000 | 2023-10-15 | 2024-10-14 |
|     4203 | Renters Insurance     |          18000 | 2023-11-25 | 2024-11-24 |
|     5021 | Home Insurance        |          15000 | 2023-11-05 | 2024-11-04 |
|     5831 | Business Insurance    |          25000 | 2023-10-08 | 2024-09-07 |
|     6947 | Disability Insurance  |          12000 | 2023-12-01 | 2024-11-30 |
|     7652 | Travel Insurance      |          60000 | 2023-12-10 | 2024-11-30 |
|     8265 | Umbrella Insurance    |          30000 | 2023-09-15 | 2024-08-31 |
+----------+-----------------------+----------------+------------+------------+
11 rows in set (0.00 sec)
```

- Policy_holder

```
mysql> select * from policy_holder;
+-----------------+--------------------+----------------+--------------------------+-----------------+
| Policy_HolderID | Policy_Holder_Name | Contact_Number | Email                    | Nomine_Name     |
+-----------------+--------------------+----------------+--------------------------+-----------------+
|            1234 | Saran              |       98403977 | saran234@gmail.com       | Saratha         |
|            1235 | Haytham            |        1482285 | haytham234@gmail.com     | Akash           |
|            1236 | Dursan             |       15952822 | dursanak61@gmail.com     | Leo             |
|            1237 | Harini             |       42814285 | ns200@gmail.com          | Lokesh          |
|            1238 | Akash              |       48128950 | gmailid@gmail.com        | Kamal           |
|            1239 | Sebin              |       42198152 | gayler69@gmail.com       | Rajnican't      |
|            1240 | Janaaki            |       69696969 | jaimaatha@gmail.com      | jaihind         |
|            2345 | Sarah White        |      765432109 | sarah.white@email.com    | Mike Johnson    |
|            3456 | Emily Wilson       |      543210987 | emily.wilson@email.com   | James Taylor    |
|            4567 | Jessica Clark      |      321098765 | jessica.clark@email.com  | Ethan Harris    |
|            5678 | John Miller        |      876543210 | john.miller@email.com    | Emma Davis      |
|            6785 | Olivia Adams       |      789012345 | olivia.adams@email.com   | Noah Williams   |
|            6789 | David Brown        |      654321098 | david.brown@email.com    | Olivia Anderson |
|            7890 | Brian Davis        |      432109876 | brian.davis@email.com    | Lily Thompson   |
|            8901 | Michael Turner     |      210987654 | michael.turner@email.com | Sophia Martinez |
|            9012 | Ethan Moore        |      876501234 | ethan.moore@email.com    | Ava Rodriguez   |
+-----------------+--------------------+----------------+--------------------------+-----------------+
16 rows in set (0.00 sec)
```

Questions and Quires:

1. Find the total number of claims pending approval.

```
mysql> SELECT COUNT(*) FROM claim WHERE Claim_Status = 'Pending';
+----------+
| COUNT(*) |
+----------+
|        3 |
+----------+
1 row in set (0.00 sec)
```

2. Retrieve the contact details of all policy holders.

```
mysql> SELECT Contact_Number, Email FROM policy_holder;
+----------------+--------------------------+
| Contact_Number | Email                    |
+----------------+--------------------------+
|       98403977 | saran234@gmail.com       |
|        1482285 | haytham234@gmail.com     |
|       15952822 | dursanak61@gmail.com     |
|       42814285 | ns200@gmail.com          |
|       48128950 | gmailid@gmail.com        |
|       42198152 | gayler69@gmail.com       |
|       69696969 | jaimaatha@gmail.com      |
|      765432109 | sarah.white@email.com    |
|      543210987 | emily.wilson@email.com   |
|      321098765 | jessica.clark@email.com  |
|      876543210 | john.miller@email.com    |
|      789012345 | olivia.adams@email.com   |
|      654321098 | david.brown@email.com    |
|      432109876 | brian.davis@email.com    |
|      210987654 | michael.turner@email.com |
|      876501234 | ethan.moore@email.com    |
+----------------+--------------------------+
16 rows in set (0.00 sec)
```

3. Display the total sum of claim amounts for a given time period.

```
mysql> SELECT SUM(Amount_Required) AS Total_Claim_Amount
    -> FROM claim
    -> WHERE Date BETWEEN 'start_date' AND 'end_date';
+--------------------+
| Total_Claim_Amount |
+--------------------+
|               NULL |
+--------------------+
1 row in set, 22 warnings (0.00 sec)
```

4. List the claims that have been approved and their corresponding claim amounts.

```
mysql> SELECT ClaimID, Amount_Required
    -> FROM claim
    -> WHERE Claim_Status = 'Approved';
+---------+-----------------+
| ClaimID | Amount_Required |
+---------+-----------------+
|    1234 |           26000 |
|    3344 |          1700.4 |
|    4567 |          1800.6 |
|    5678 |           15000 |
|    6789 |         2000.25 |
+---------+-----------------+
5 rows in set (0.00 sec)
```

5. Calculate the average premium amount for each policy type.

```
mysql> SELECT Policy_Type, AVG(Premium_Amount) AS Average_Premium
    -> FROM insurance_policy
    -> GROUP BY Policy_Type;
+----------------------+-----------------+
| Policy_Type          | Average_Premium |
+----------------------+-----------------+
| General              |           10000 |
| Life Insurance       |           20000 |
| Health               |           12000 |
| Pet Insurance        |           10000 |
| Auto Insurance       |           80000 |
| Renters Insurance    |           18000 |
| Home Insurance       |           15000 |
| Business Insurance   |           25000 |
| Disability Insurance |           12000 |
| Travel Insurance     |           60000 |
| Umbrella Insurance   |           30000 |
+----------------------+-----------------+
11 rows in set (0.00 sec)
```

6. Calculate the total claims amount for a specific policy.

```
mysql> SELECT PolicyID, SUM(Amount_Required) AS Total_Claims_Amount
    -> FROM claim
    -> JOIN insurance_policy ON claim.ClaimID = insurance_policy.PolicyID
    -> WHERE insurance_policy.PolicyID = 'your_policy_id';
+----------+----------------------+
| PolicyID | Total_Claims_Amount |
+----------+----------------------+
|     NULL |                 NULL |
+----------+----------------------+
1 row in set, 1 warning (0.00 sec)
```

7. Retrieve clients with multiple policies.

```
mysql> SELECT Policy_HolderID, Policy_Holder_Name
    -> FROM policy_holder
    -> WHERE Policy_HolderID IN (
    ->     SELECT Policy_HolderID
    ->     FROM insurance_policy
    ->     GROUP BY Policy_HolderID
    ->     HAVING COUNT(*) > 1
    -> );
+-----------------+-------------------+
| Policy_HolderID | Policy_Holder_Name |
+-----------------+-------------------+
|            1234 | Saran             |
|            1235 | Haytham           |
|            1236 | Dursan            |
|            1237 | Harini            |
|            1238 | Akash             |
|            1239 | Sebin             |
|            1240 | Janaaki           |
|            2345 | Sarah White       |
|            3456 | Emily Wilson      |
|            4567 | Jessica Clark     |
|            5678 | John Miller       |
|            6785 | Olivia Adams      |
|            6789 | David Brown       |
|            7890 | Brian Davis       |
|            8901 | Michael Turner    |
|            9012 | Ethan Moore       |
+-----------------+-------------------+
16 rows in set (0.00 sec)
```

8. Count the number of policies for each policy type.

```
mysql> SELECT Policy_Type, COUNT(*) AS Policy_Count
    -> FROM insurance_policy
    -> GROUP BY Policy_Type;
+----------------------+--------------+
| Policy_Type          | Policy_Count |
+----------------------+--------------+
| General              |            1 |
| Life Insurance       |            1 |
| Health               |            1 |
| Pet Insurance        |            1 |
| Auto Insurance       |            1 |
| Renters Insurance    |            1 |
| Home Insurance       |            1 |
| Business Insurance   |            1 |
| Disability Insurance |            1 |
| Travel Insurance     |            1 |
| Umbrella Insurance   |            1 |
+----------------------+--------------+
11 rows in set (0.00 sec)
```

9. Retrieve policies with a premium amount above a certain value.

```
mysql> SELECT *
    -> FROM insurance_policy
    -> WHERE Premium_Amount > 20000;
+----------+--------------------+----------------+------------+------------+
| PolicyID | Policy_Type        | Premium_Amount | Start_Date | End_Date   |
+----------+--------------------+----------------+------------+------------+
|     3874 | Auto Insurance     |          80000 | 2023-10-15 | 2024-10-14 |
|     5831 | Business Insurance |          25000 | 2023-10-08 | 2024-09-07 |
|     7652 | Travel Insurance   |          60000 | 2023-12-10 | 2024-11-30 |
|     8265 | Umbrella Insurance |          30000 | 2023-09-15 | 2024-08-31 |
+----------+--------------------+----------------+------------+------------+
4 rows in set (0.00 sec)
```

10.Retrieve claims filed by a specific customer.

```
mysql> UPDATE policy_holder
    -> SET Email = 'darshan@leo.com'
    -> WHERE Policy_HolderID = '1236';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from policy_holder where Policy_HolderID ='1236';
+-----------------+--------------------+----------------+-----------------+-------------+
| Policy_HolderID | Policy_Holder_Name | Contact_Number | Email           | Nomine_Name |
+-----------------+--------------------+----------------+-----------------+-------------+
|            1236 | Dursan             |       15952822 | darshan@leo.com | Leo         |
+-----------------+--------------------+----------------+-----------------+-------------+
1 row in set (0.00 sec)
```

11.Delete a customer and associated data.

```
mysql> DELETE FROM policy_holder
    -> WHERE Policy_HolderID = '5678';
Query OK, 1 row affected (0.00 sec)

mysql> select * from policy_holder;
+----------------+--------------------+---------------+-------------------------+-----------------+
| Policy_HolderID | Policy_Holder_Name | Contact_Number | Email                   | Nomine_Name     |
+----------------+--------------------+---------------+-------------------------+-----------------+
|           1234 | Saran              |      98403977 | saran234@gmail.com      | Saratha         |
|           1235 | Haytham            |       1482285 | haytham234@gmail.com    | Akash           |
|           1236 | Dursan             |      15952822 | darshan@leo.com         | Leo             |
|           1237 | Harini             |      42814285 | ns200@gmail.com         | Lokesh          |
|           1238 | Akash              |      48128950 | gmailid@gmail.com       | Kamal           |
|           1239 | Sebin              |      42198152 | gayler69@gmail.com      | Rajnican't      |
|           1240 | Janaaki            |      69696969 | jaimaatha@gmail.com     | jaihind         |
|           2345 | Sarah White        |     765432109 | sarah.white@email.com   | Mike Johnson    |
|           3456 | Emily Wilson       |     543210987 | emily.wilson@email.com  | James Taylor    |
|           4567 | Jessica Clark      |     321098765 | jessica.clark@email.com | Ethan Harris    |
|           6785 | Olivia Adams       |     789012345 | olivia.adams@email.com  | Noah Williams   |
|           6789 | David Brown        |     654321098 | david.brown@email.com   | Olivia Anderson |
|           7890 | Brian Davis        |     432109876 | brian.davis@email.com   | Lily Thompson   |
|           8901 | Michael Turner     |     210987654 | michael.turner@email.com | Sophia Martinez |
|           9012 | Ethan Moore        |     876501234 | ethan.moore@email.com   | Ava Rodriguez   |
+----------------+--------------------+---------------+-------------------------+-----------------+
15 rows in set (0.00 sec)
```

12.What is the count of total number of policy holders?

```
mysql> SELECT COUNT(*) AS total_policy_holders
    -> FROM policy_holder;
+----------------------+
| total_policy_holders |
+----------------------+
|                    7 |
+----------------------+
1 row in set (0.00 sec)
```

13.Retrieve the policy IDs and their corresponding end dates for policies that have already ended.

```
mysql> SELECT PolicyID, End_date
    -> FROM insurance_policy
    -> WHERE End_date < CURRENT_DATE;
+----------+------------+
| PolicyID | End_date   |
+----------+------------+
|     1234 | 2023-08-21 |
+----------+------------+
```

14. List all the policy IDs with premium amount with a premium amount greater than 25,000.

```
mysql> SELECT PolicyID, Premium_amount
    -> FROM insurance_policy
    -> WHERE Premium_amount > 25000;
+----------+----------------+
| PolicyID | Premium_amount |
+----------+----------------+
|     3874 |          80000 |
|     7652 |          60000 |
|     8265 |          30000 |
+----------+----------------+
3 rows in set (0.00 sec)
```

15.How many claims require amount more than Rs 20,000?

```
mysql> SELECT COUNT(*) AS Num_Claims
    -> FROM claim
    -> WHERE Amount_Required > 20000;
+------------+
| Num_Claims |
+------------+
|          2 |
+------------+
1 row in set (0.00 sec)
```

16.How many claims have been successfully approved?

```
mysql> SELECT COUNT(*)
    -> FROM claim
    -> WHERE Claim_Status = 'Approved';
+----------+
| COUNT(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
```

17. Retrieve the policy id of disability insurance policy.

```
mysql> SELECT PolicyID
    -> FROM insurance_policy
    -> WHERE Policy_Type = 'Disability Insurance';
+----------+
| PolicyID |
+----------+
|     6947 |
+----------+
```

18. Count the total number of health insurance policy holders.

```
mysql> SELECT COUNT(*) AS total_health_policy_holders
    -> FROM insurance_policy
    -> WHERE Policy_type = 'Life Insurance';
+-----------------------------+
| total_health_policy_holders |
+-----------------------------+
|                           1 |
+-----------------------------+
1 row in set (0.00 sec)
```

19.Display policy holder with specific type of policy.

```
mysql> SELECT policy_holder.*
    -> FROM policy_holder
    -> JOIN insurance_policy ON policy_holder.Policy_HolderID = insurance_policy.PolicyID
    -> WHERE insurance_policy.Policy_Type = 'General';
+----------------+--------------------+---------------+--------------------+-------------+
| Policy_HolderID | Policy_Holder_Name | Contact_Number | Email              | Nomine_Name |
+----------------+--------------------+---------------+--------------------+-------------+
|           1234 | Saran              |      98403977 | saran234@gmail.com | Saratha     |
+----------------+--------------------+---------------+--------------------+-------------+
1 row in set (0.00 sec)
```

20.Find the principal amount of a person's policy.

```
mysql> SELECT Policy_Holder_Name, Premium_Amount
    -> FROM policy_holder
    -> JOIN insurance_policy ON policy_holder.Policy_HolderID = insurance_policy.PolicyID
    -> WHERE policy_holder.Policy_Holder_Name = 'Haytham';
Empty set (0.00 sec)
```

21.Find the total principal amount.

```
mysql> SELECT SUM(Premium_Amount) AS Total_Principal_Amount
    -> FROM insurance_policy;
+------------------------+
| Total_Principal_Amount |
+------------------------+
|                 292000 |
+------------------------+
1 row in set (0.00 sec)
```

22.List of policies that had expired.

```
mysql> SELECT *
    -> FROM insurance_policy
    -> WHERE End_Date < NOW();
+----------+-------------+----------------+------------+------------+
| PolicyID | Policy_Type | Premium_Amount | Start_Date | End_Date   |
+----------+-------------+----------------+------------+------------+
|     1234 | General     |          10000 | 2022-10-21 | 2023-08-21 |
+----------+-------------+----------------+------------+------------+
1 row in set (0.00 sec)
```

23.Find the remaining duration of policy.

```
mysql> SELECT PolicyID, DATEDIFF(End_Date, NOW()) AS Remaining_Duration_Days
    -> FROM insurance_policy;
+----------+------------------------+
| PolicyID | Remaining_Duration_Days |
+----------+------------------------+
|     1234 |                    -62 |
|     1498 |                    344 |
|     2156 |                    375 |
|     3098 |                    333 |
|     3874 |                    358 |
|     4203 |                    399 |
|     5021 |                    379 |
|     5831 |                    321 |
|     6947 |                    405 |
|     7652 |                    405 |
|     8265 |                    314 |
+----------+------------------------+
11 rows in set (0.00 sec)
```

24.Find average amount of policies.

```
mysql> SELECT AVG(Premium_Amount) AS Average_Premium_Amount
    -> FROM insurance_policy;
+------------------------+
| Average_Premium_Amount |
+------------------------+
|     26545.454545454544 |
+------------------------+
1 row in set (0.00 sec)
```

25.Listing policy holders in order.

```
mysql> SELECT *
    -> FROM policy_holder
    -> ORDER BY Policy_Holder_Name ASC;
+-----------------+--------------------+----------------+-------------------------+-----------------+
| Policy_HolderID | Policy_Holder_Name | Contact_Number | Email                   | Nomine_Name     |
+-----------------+--------------------+----------------+-------------------------+-----------------+
|            1238 | Akash              |       48128950 | gmailid@gmail.com       | Kamal           |
|            7890 | Brian Davis        |      432109876 | brian.davis@email.com   | Lily Thompson   |
|            6789 | David Brown        |      654321098 | david.brown@email.com   | Olivia Anderson |
|            1236 | Dursan             |       15952822 | darshan@leo.com         | Leo             |
|            3456 | Emily Wilson       |      543210987 | emily.wilson@email.com  | James Taylor    |
|            9012 | Ethan Moore        |      876501234 | ethan.moore@email.com   | Ava Rodriguez   |
|            1237 | Harini             |       42814285 | ns200@gmail.com         | Lokesh          |
|            1235 | Haytham            |        1482285 | haytham234@gmail.com    | Akash           |
|            1240 | Janaaki            |       69696969 | jaimaatha@gmail.com     | jaihind         |
|            4567 | Jessica Clark      |      321098765 | jessica.clark@email.com | Ethan Harris    |
|            8901 | Michael Turner     |      210987654 | michael.turner@email.com| Sophia Martinez |
|            6785 | Olivia Adams       |      789012345 | olivia.adams@email.com  | Noah Williams   |
|            2345 | Sarah White        |      765432109 | sarah.white@email.com   | Mike Johnson    |
|            1234 | Saran              |       98403977 | saran234@gmail.com      | Saratha         |
|            1239 | Sebin              |       42198152 | gayler69@gmail.com      | Rajnican't      |
+-----------------+--------------------+----------------+-------------------------+-----------------+
15 rows in set (0.00 sec)
```

26.List all policy holders along with their nominee names.

```
mysql> SELECT Policy_Holder_Name, Nomine_Name
    -> FROM policy_holder;
+--------------------+-----------------+
| Policy_Holder_Name | Nomine_Name     |
+--------------------+-----------------+
| Saran              | Saratha         |
| Haytham            | Akash           |
| Dursan             | Leo             |
| Harini             | Lokesh          |
| Akash              | Kamal           |
| Sebin              | Rajnican't      |
| Janaaki            | jaihind         |
| Sarah White        | Mike Johnson    |
| Emily Wilson       | James Taylor    |
| Jessica Clark      | Ethan Harris    |
| John Miller        | Emma Davis      |
| Olivia Adams       | Noah Williams   |
| David Brown        | Olivia Anderson |
| Brian Davis        | Lily Thompson   |
| Michael Turner     | Sophia Martinez |
| Ethan Moore        | Ava Rodriguez   |
+--------------------+-----------------+
16 rows in set (0.00 sec)
```

27.Retrieve policy holders who have policies that expire within the next month.

```
mysql> SELECT ph.Policy_Holder_Name
    -> FROM policy_holder ph
    -> JOIN insurance_policy ip ON ph.Policy_HolderID = ip.PolicyID
    -> WHERE ip.End_Date BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 1 MONTH);
Empty set (0.00 sec)
```

28. Get the policy holder's name and contact details for a given policy ID.

```
mysql> SELECT ph.Policy_Holder_Name, ph.Contact_Number, ph.Email
    -> FROM policy_holder ph
    -> JOIN insurance_policy ip ON ph.Policy_HolderID = ip.PolicyID
    -> WHERE ip.PolicyID = 'your_policy_id';
Empty set, 1 warning (0.00 sec)
```

29. List all policy holders in alphabetical order.

```
mysql> SELECT Policy_Holder_Name
    -> FROM policy_holder
    -> ORDER BY Policy_Holder_Name;
+--------------------+
| Policy_Holder_Name |
+--------------------+
| Akash              |
| Brian Davis        |
| David Brown        |
| Dursan             |
| Emily Wilson       |
| Ethan Moore        |
| Harini             |
| Haytham            |
| Janaaki            |
| Jessica Clark      |
| John Miller        |
| Michael Turner     |
| Olivia Adams       |
| Sarah White        |
| Saran              |
| Sebin              |
+--------------------+
16 rows in set (0.00 sec)
```

30. Get the count of all claims that have been approved.

```
mysql> SELECT COUNT(*)
    -> FROM claim
    -> WHERE Claim_Status = 'Approved';
+----------+
| COUNT(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
```

31. List claims with a pending status.

```
mysql> SELECT *
    -> FROM claim
    -> WHERE Claim_Status = 'Pending';
+---------+--------------+-----------------+------------+
| ClaimID | Claim_Status | Amount_Required | Date       |
+---------+--------------+-----------------+------------+
|    1235 | Pending      |           80000 | 2023-10-01 |
|    3456 | Pending      |          1200.8 | 2023-10-12 |
|    8901 | Pending      |           950.3 | 2023-10-22 |
+---------+--------------+-----------------+------------+
3 rows in set (0.00 sec)
```

32.Get the count of general insurance policy holders.

```
mysql> SELECT COUNT(*)
    -> FROM insurance_policy
    -> WHERE Policy_Type = 'General';
+----------+
| COUNT(*) |
+----------+
|        1 |
+----------+
1 row in set (0.00 sec)
```

33.Update the premium amount to 20000 whose policy ID is 1234

```
mysql> SELECT policy_holder.Policy_HolderID,
    ->        policy_holder.Policy_Holder_Name,
    ->        COUNT(claim.ClaimID) AS Total_Claims,
    ->        SUM(claim.Amount_Required) AS Total_Claim_Amount
    -> FROM policy_holder
    -> LEFT JOIN insurance_policy ON policy_holder.Policy_HolderID = insurance_policy.PolicyID
    -> LEFT JOIN claim ON insurance_policy.PolicyID = claim.ClaimID
    -> GROUP BY policy_holder.Policy_HolderID, policy_holder.Policy_Holder_Name;
+----------------+--------------------+--------------+--------------------+
| Policy_HolderID | Policy_Holder_Name | Total_Claims | Total_Claim_Amount |
+----------------+--------------------+--------------+--------------------+
|           1234 | Saran              |            1 |              26000 |
|           1235 | Haytham            |            0 |               NULL |
|           1236 | Dursan             |            0 |               NULL |
|           1237 | Harini             |            0 |               NULL |
|           1238 | Akash              |            0 |               NULL |
|           1239 | Sebin              |            0 |               NULL |
|           1240 | Janaaki            |            0 |               NULL |
|           2345 | Sarah White        |            0 |               NULL |
|           3456 | Emily Wilson       |            0 |               NULL |
|           4567 | Jessica Clark      |            0 |               NULL |
|           6785 | Olivia Adams       |            0 |               NULL |
|           6789 | David Brown        |            0 |               NULL |
|           7890 | Brian Davis        |            0 |               NULL |
|           8901 | Michael Turner     |            0 |               NULL |
|           9012 | Ethan Moore        |            0 |               NULL |
+----------------+--------------------+--------------+--------------------+
15 rows in set (0.00 sec)
```

34. From the table insurance company order the address in descending

```
mysql> SELECT *
    -> FROM insurance_company
    -> ORDER BY Address DESC;
+------------+---------------------+-----------+------------------+
| Company_ID | Company_Name        | Address   | Contact_Details |
+------------+---------------------+-----------+------------------+
|       8902 | SBI General         | Pune      |       210987654 |
|       1235 | HDFC Ergo           | Mumbai    |       987654321 |
|       2468 | Bharti AXA          | Lucknow   |       987654321 |
|       1357 | USAA                | Kolkata   |      1098765432 |
|       7890 | Reliance            | Kolkata   |       432109876 |
|       3456 | United India        | Hyderabad |       543210987 |
|       5678 | ICICI Lombard       | Delhi     |       876543210 |
|       1234 | TATA AIG            | Chennai   |       912738901 |
|       6789 | Oriental            | Chennai   |       654321098 |
|       8901 | Travelers Insurance | Bengaluru |      2109876543 |
|       2345 | New India           | Bangalore |       765432109 |
|       4567 | Bajaj Allianz       | Ahmedabad |       321098765 |
+------------+---------------------+-----------+------------------+
12 rows in set (0.00 sec)
```

35. Select all the approved claims.

```
mysql> SELECT *
    -> FROM claim
    -> WHERE Claim_Status = 'Approved';
+---------+--------------+-----------------+------------+
| ClaimID | Claim_Status | Amount_Required | Date       |
+---------+--------------+-----------------+------------+
|    1234 | Approved     |           26000 | 2023-08-21 |
|    3344 | Approved     |          1700.4 | 2023-10-28 |
|    4567 | Approved     |          1800.6 | 2023-10-18 |
|    5678 | Approved     |           15000 | 2023-10-01 |
|    6789 | Approved     |         2000.25 | 2023-10-08 |
+---------+--------------+-----------------+------------+
5 rows in set (0.00 sec)
```

36. Select the policy ID whose premium amount is within the range 30000 to 40000.

```
mysql> SELECT PolicyID
    -> FROM insurance_policy
    -> WHERE Premium_Amount BETWEEN 30000 AND 40000;
+----------+
| PolicyID |
+----------+
|     8265 |
+----------+
1 row in set (0.00 sec)
```

37. Get the policy holders name and address for the policy holder ID 1234

```
mysql> UPDATE insurance_policy
    -> SET Premium_Amount = 20000
    -> WHERE PolicyID = 1234;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from insurance_policy;
+----------+--------------------+----------------+------------+------------+
| PolicyID | Policy_Type        | Premium_Amount | Start_Date | End_Date   |
+----------+--------------------+----------------+------------+------------+
|     1234 | General            |          20000 | 2022-10-21 | 2023-08-21 |
|     1498 | Life Insurance     |          20000 | 2023-10-01 | 2024-09-30 |
|     2156 | Health             |          12000 | 2023-11-01 | 2024-10-31 |
|     3098 | Pet Insurance      |          10000 | 2023-09-20 | 2024-09-19 |
|     3874 | Auto Insurance     |          80000 | 2023-10-15 | 2024-10-14 |
|     4203 | Renters Insurance  |          18000 | 2023-11-25 | 2024-11-24 |
|     5021 | Home Insurance     |          15000 | 2023-11-05 | 2024-11-04 |
|     5831 | Business Insurance |          25000 | 2023-10-08 | 2024-09-07 |
|     6947 | Disability Insurance |        12000 | 2023-12-01 | 2024-11-30 |
|     7652 | Travel Insurance   |          60000 | 2023-12-10 | 2024-11-30 |
|     8265 | Umbrella Insurance |          30000 | 2023-09-15 | 2024-08-31 |
+----------+--------------------+----------------+------------+------------+
11 rows in set (0.00 sec)
```

38. Update claim status to approved for claim ID 5678

```
mysql> UPDATE claim
    -> SET Claim_Status = 'Approved'
    -> WHERE ClaimID = 5678;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> select * from claim;
+---------+--------------+-----------------+------------+
| ClaimID | Claim_Status | Amount_Required | Date       |
+---------+--------------+-----------------+------------+
|    1122 | Denied       |           600.9 | 2023-10-25 |
|    1234 | Approved     |           26000 | 2023-08-21 |
|    1235 | Pending      |           80000 | 2023-10-01 |
|    2345 | Denied       |          300.75 | 2023-10-05 |
|    3344 | Approved     |          1700.4 | 2023-10-28 |
|    3456 | Pending      |          1200.8 | 2023-10-12 |
|    4567 | Approved     |          1800.6 | 2023-10-18 |
|    5678 | Approved     |           15000 | 2023-10-01 |
|    6789 | Approved     |         2000.25 | 2023-10-08 |
|    7890 | Denied       |           450.2 | 2023-10-15 |
|    8901 | Pending      |           950.3 | 2023-10-22 |
+---------+--------------+-----------------+------------+
11 rows in set (0.00 sec)
```

39.Find the policy holder who has made the highest claim.

```
mysql> SELECT ph.Policy_HolderID,
    ->        ph.Policy_Holder_Name,
    ->        ph.Contact_Number,
    ->        ph.Email,
    ->        ip.Policy_Type,
    ->        ip.Premium_Amount,
    ->        ip.Start_Date,
    ->        ip.End_Date
    -> FROM policy_holder AS ph, insurance_policy AS ip
    -> WHERE ph.Policy_HolderID = ip.PolicyID;
+----------------+------------------+---------------+------------------+------------+----------------+------------+------------+
| Policy_HolderID | Policy_Holder_Name | Contact_Number | Email            | Policy_Type | Premium_Amount | Start_Date | End_Date   |
+----------------+------------------+---------------+------------------+------------+----------------+------------+------------+
|           1234 | Saran            |       98403977 | saran234@gmail.com | General     |          20000 | 2022-10-21 | 2023-08-21 |
+----------------+------------------+---------------+------------------+------------+----------------+------------+------------+
1 row in set (0.00 sec)
```

40.Retrieve the policy holder with the highest premium amount paid.

```
mysql> SELECT Policy_HolderID, Policy_Holder_Name, Contact_Number, Email
    -> FROM policy_holder
    -> ORDER BY (SELECT MAX(Premium_Amount) FROM insurance_policy WHERE insurance_policy.PolicyID = policy_holder.Policy
_HolderID) DESC
    -> LIMIT 1;
+----------------+------------------+---------------+------------------+
| Policy_HolderID | Policy_Holder_Name | Contact_Number | Email            |
+----------------+------------------+---------------+------------------+
|           1234 | Saran            |       98403977 | saran234@gmail.com |
+----------------+------------------+---------------+------------------+
1 row in set (0.00 sec)
```

Result: Thus, working with queries related to project on MySQL is successfully verified.

Exp: 8	Working With Basic PL/SQL Programming

Aim: To work with basic PL/SQL programming on MySQL.

1. Write a PL/SL program to check given number is prime or not.

```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE is_prime(IN num INT, OUT is_prime BOOLEAN)
    -> BEGIN
    ->   DECLARE i INT;
    ->   SET is_prime = TRUE;
    ->
    ->   IF num <= 1 THEN
    ->     SET is_prime = FALSE;
    ->   ELSE
    ->     SET i = 2;
    ->     WHILE i * i <= num DO
    ->       IF num % i = 0 THEN
    ->         SET is_prime = FALSE;
    ->         LEAVE;
    ->       END IF;
    ->       SET i = i + 1;
    ->     END WHILE;
    ->   END IF;
    -> END;
    -> //
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ';      END IF;
      SET i = i + 1;
    END WHILE;
  END IF;
END' at line 13
mysql>
mysql> DELIMITER ;
mysql>
mysql> -- Example usage
mysql> SET @num_to_check = 17; -- Change this number to the one you want to check
Query OK, 0 rows affected (0.00 sec)

mysql> CALL is_prime(@num_to_check, @is_prime);
ERROR 1305 (42000): PROCEDURE sakila.is_prime does not exist
mysql>
mysql> SELECT IF(@is_prime, CONCAT(@num_to_check, ' is a prime number.'), CONCAT(@num_to_check, ' is not a prime number.')) AS result;
+---------------------------+
| result                    |
+---------------------------+
| 17 is not a prime number. |
+---------------------------+
1 row in set (0.00 sec)
```

2. Write a PL/SL program to display factorial of a given number.

```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE calculate_factorial(IN num INT)
    -> BEGIN
    ->   DECLARE result INT DEFAULT 1;
    ->   DECLARE i INT DEFAULT 1;
    ->
    ->   IF num < 0 THEN
    ->     SELECT 'Factorial is not defined for negative numbers.';
    ->   ELSE
    ->     WHILE i <= num DO
    ->       SET result = result * i;
    ->       SET i = i + 1;
    ->     END WHILE;
    ->
    ->     IF num = 0 THEN
    ->       SELECT 'Factorial of 0 is 1.';
    ->     ELSE
    ->       SELECT CONCAT('Factorial of ', num, ' is ', result);
    ->     END IF;
    ->   END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> -- Example usage
mysql> CALL calculate_factorial(5); -- Change the argument to calculate the factorial for a different number
+-------------------------------------------+
| CONCAT('Factorial of ', num, ' is ', result) |
+-------------------------------------------+
| Factorial of 5 is 120                     |
+-------------------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

3. Write a PL/SL program to reverser a given number.

```
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION reverse_number(num INT) RETURNS INT
    -> BEGIN
    ->   DECLARE reversed_num INT DEFAULT 0;
    ->   DECLARE digit INT;
    ->
    ->   WHILE num > 0 DO
    ->     SET digit = num % 10;
    ->     SET reversed_num = reversed_num * 10 + digit;
    ->     SET num = FLOOR(num / 10);
    ->   END WHILE;
    ->
    ->   RETURN reversed_num;
    -> END;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> -- Example usage
mysql> SELECT reverse_number(12345); -- Change the argument to reverse a different number
+----------------------+
| reverse_number(12345) |
+----------------------+
|                54321 |
+----------------------+
1 row in set (0.00 sec)
```

4. Write a PL/SL program to generate a Fibonacci series.

```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE generate_fibonacci_series(terms INT)
    -> BEGIN
    ->   DECLARE a INT DEFAULT 0;
    ->   DECLARE b INT DEFAULT 1;
    ->   DECLARE c INT;
    ->   DECLARE i INT DEFAULT 1;
    ->
    ->   IF terms = 0 THEN
    ->     SELECT 'No terms to generate.';
    ->   ELSEIF terms >= 1 THEN
    ->     SET i = 1;
    ->     SELECT 'Fibonacci Series: ';
    ->     SELECT a;
    ->
    ->     IF terms >= 2 THEN
    ->       SELECT ', ' + b;
    ->     END IF;
    ->
    ->     WHILE i < terms DO
    ->       SET c = a + b;
    ->       SELECT ', ' + c;
    ->       SET a = b;
    ->       SET b = c;
    ->       SET i = i + 1;
    ->     END WHILE;
    ->
    ->     SELECT ''; -- Output a newline
    ->   ELSE
    ->     SELECT 'Invalid number of terms.';
    ->   END IF;
    -> END;
    -> //
```

```
mysql> CALL generate_fibonacci_series(10); -- Change the argument to specify the number of terms
+--------------------+
| Fibonacci Series:  |
+--------------------+
| Fibonacci Series:  |
+--------------------+
1 row in set (0.00 sec)

+------+
| a    |
+------+
|    0 |
+------+
1 row in set (0.00 sec)

+----------+
| ', ' + b |
+----------+
|        1 |
+----------+
1 row in set (0.00 sec)

+----------+
| ', ' + c |
+----------+
|        1 |
+----------+
1 row in set (0.00 sec)
```

```
+----------+
| ', ' + c |
+----------+
|        3 |
+----------+
1 row in set (0.00 sec)

+----------+
| ', ' + c |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

+----------+
| ', ' + c |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)

+----------+
| ', ' + c |
+----------+
|       13 |
+----------+
1 row in set (0.01 sec)

+----------+
| ', ' + c |
+----------+
|       21 |
+----------+
1 row in set (0.01 sec)
```

5. Write a PL/SL program to check given number is palindrome or not

```
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION is_palindrome(num INT) RETURNS BOOLEAN
    -> BEGIN
    ->   DECLARE original_num INT;
    ->   DECLARE reversed_num INT DEFAULT 0;
    ->   DECLARE digit INT;
    ->
    ->   SET original_num = num;
    ->
    ->   WHILE num > 0 DO
    ->     SET digit = num % 10;
    ->     SET reversed_num = reversed_num * 10 + digit;
    ->     SET num = FLOOR(num / 10);
    ->   END WHILE;
    ->
    ->   RETURN original_num = reversed_num;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> -- Example usage
mysql> SELECT is_palindrome(12321); -- Change the argument to check a different number
+----------------------+
| is_palindrome(12321) |
+----------------------+
|                    1 |
+----------------------+
1 row in set (0.00 sec)
```

Result: Thus, working with basic PL/SQL programming on MySQL is successfully verified.

Exp: 9	Working With PL/SQL Procedures

Aim: To work with PL/SQL Procedures on MySQL.

1)Insurance Policy

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE InsertInsurancePolicy(
    ->     IN p_PolicyID INT,
    ->     IN p_Policy_Type VARCHAR(255),
    ->     IN p_Premium_Amount DECIMAL(10, 2),
    ->     IN p_Start_Date DATE,
    ->     IN p_End_Date DATE
    -> )
    -> BEGIN
    ->     INSERT INTO insurance_policy(PolicyID, Policy_Type, Premium_Amount, Start_Date, End_Date)
    ->     VALUES (p_PolicyID, p_Policy_Type, p_Premium_Amount, p_Start_Date, p_End_Date);
    -> END;
    -> //
```

```
mysql> CALL InsertInsurancePolicy(1896,'Marine Insurance','20010','2022-01-20','2023-01-21');
    -> //
Query OK, 1 row affected (0.01 sec)
```

```
+----------+----------------------+----------------+------------+------------+
| PolicyID | Policy_Type          | Premium_Amount | Start_Date | End_Date   |
+----------+----------------------+----------------+------------+------------+
|     1234 | General              |          20000 | 2022-10-21 | 2023-08-21 |
|     1498 | Life Insurance       |          20000 | 2023-10-01 | 2024-09-30 |
|     1896 | Marine Insurance     |          20010 | 2022-01-20 | 2023-01-21 |
|     2156 | Health               |          12000 | 2023-11-01 | 2024-10-31 |
|     3098 | Pet Insurance        |          10000 | 2023-09-20 | 2024-09-19 |
|     3874 | Auto Insurance       |          80000 | 2023-10-15 | 2024-10-14 |
|     4203 | Renters Insurance    |          18000 | 2023-11-25 | 2024-11-24 |
|     5021 | Home Insurance       |          15000 | 2023-11-05 | 2024-11-04 |
|     5486 | Health               |          10200 | 2022-08-03 | 2023-09-04 |
|     5831 | Business Insurance   |          25000 | 2023-10-08 | 2024-09-07 |
|     6947 | Disability Insurance |          12000 | 2023-12-01 | 2024-11-30 |
|     7652 | Travel Insurance     |          60000 | 2023-12-10 | 2024-11-30 |
|     8265 | Umbrella Insurance   |          30000 | 2023-09-15 | 2024-08-31 |
+----------+----------------------+----------------+------------+------------+
13 rows in set (0.00 sec)
```

2) Insurance Company

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE InsertInsuranceCompany(
    ->     IN p_Company_ID INT,
    ->     IN p_Company_Name VARCHAR(255),
    ->     IN p_Address VARCHAR(255),
    ->     IN p_Contact_Details VARCHAR(255)
    -> )
    -> BEGIN
    ->     INSERT INTO insurance_company(Company_ID, Company_Name, Address, Contact_Details)
    ->     VALUES (p_Company_ID, p_Company_Name, p_Address, p_Contact_Details);
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> DELIMITER //
mysql> CALL InsertInsuranceCompany(2382,'SATA','Trichy','96551981');
    -> //
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from insurance_company;
    -> //
+------------+---------------------+-----------+-----------------+
| Company_ID | Company_Name        | Address   | Contact_Details |
+------------+---------------------+-----------+-----------------+
|       1234 | TATA AIG            | Chennai   |       912738901 |
|       1235 | HDFC Ergo           | Mumbai    |       987654321 |
|       1357 | USAA                | Kolkata   |      1098765432 |
|       2345 | New India           | Bangalore |       765432109 |
|       2382 | SATA                | Trichy    |        96551981 |
|       2468 | Bharti AXA          | Lucknow   |       987654321 |
|       3456 | United India        | Hyderabad |       543210987 |
|       4567 | Bajaj Allianz       | Ahmedabad |       321098765 |
|       5678 | ICICI Lombard       | Delhi     |       876543210 |
|       6789 | Oriental            | Chennai   |       654321098 |
|       7890 | Reliance            | Kolkata   |       432109876 |
|       8901 | Travelers Insurance | Bengaluru |      2109876543 |
|       8902 | SBI General         | Pune      |       210987654 |
+------------+---------------------+-----------+-----------------+
13 rows in set (0.00 sec)
```

3) Claim

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE SubmitInsuranceClaim(
    ->     IN p_ClaimID INT,
    ->     IN p_Claim_Status VARCHAR(50),
    ->     IN p_Amount_Required DECIMAL(10, 2),
    ->     IN p_Date DATE
    -> )
    -> BEGIN
    ->     INSERT INTO claim(ClaimID, Claim_Status, Amount_Required, Date)
    ->     VALUES (p_ClaimID, p_Claim_Status, p_Amount_Required, p_Date);
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> CALL SubmitInsuranceClaim(1896,'Approved','10000','2023-10-25');
    -> //
Query OK, 1 row affected (0.01 sec)

mysql> select * from claim;
    -> //
+---------+--------------+-----------------+------------+
| ClaimID | Claim_Status | Amount_Required | Date       |
+---------+--------------+-----------------+------------+
|    1122 | Denied       |           600.9 | 2023-10-25 |
|    1234 | Approved     |           26000 | 2023-08-21 |
|    1235 | Pending      |           80000 | 2023-10-01 |
|    1896 | Approved     |           10000 | 2023-10-25 |
|    2345 | Denied       |          300.75 | 2023-10-05 |
|    3344 | Approved     |          1700.4 | 2023-10-28 |
|    3456 | Pending      |          1200.8 | 2023-10-12 |
|    4567 | Approved     |          1800.6 | 2023-10-18 |
|    5678 | Approved     |           15000 | 2023-10-01 |
|    6789 | Approved     |         2000.25 | 2023-10-08 |
|    7890 | Denied       |           450.2 | 2023-10-15 |
|    8901 | Pending      |           950.3 | 2023-10-22 |
+---------+--------------+-----------------+------------+
12 rows in set (0.00 sec)
```

4) Update Premium

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE UpdatePolicyPremium(
    ->     IN p_PolicyID INT,
    ->     IN p_New_Premium_Amount DECIMAL(10, 2)
    -> )
    -> BEGIN
    ->     UPDATE insurance_policy
    ->     SET Premium_Amount = p_New_Premium_Amount
    ->     WHERE PolicyID = p_PolicyID;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> CALL UpdatePolicyPremium(1234,10000);
    -> //
Query OK, 1 row affected (0.01 sec)

mysql> select * from insurance_policy;
    -> //
+----------+----------------------+----------------+------------+------------+
| PolicyID | Policy_Type          | Premium_Amount | Start_Date | End_Date   |
+----------+----------------------+----------------+------------+------------+
|     1234 | General              |          10000 | 2022-10-21 | 2023-08-21 |
|     1498 | Life Insurance       |          20000 | 2023-10-01 | 2024-09-30 |
|     1896 | Marine Insurance     |          20010 | 2022-01-20 | 2023-01-21 |
|     2156 | Health               |          12000 | 2023-11-01 | 2024-10-31 |
|     3098 | Pet Insurance        |          10000 | 2023-09-20 | 2024-09-19 |
|     3874 | Auto Insurance       |          80000 | 2023-10-15 | 2024-10-14 |
|     4203 | Renters Insurance    |          18000 | 2023-11-25 | 2024-11-24 |
|     5021 | Home Insurance       |          15000 | 2023-11-05 | 2024-11-04 |
|     5486 | Health               |          10200 | 2022-08-03 | 2023-09-04 |
|     5831 | Business Insurance   |          25000 | 2023-10-08 | 2024-09-07 |
|     6947 | Disability Insurance |          12000 | 2023-12-01 | 2024-11-30 |
|     7652 | Travel Insurance     |          60000 | 2023-12-10 | 2024-11-30 |
|     8265 | Umbrella Insurance   |          30000 | 2023-09-15 | 2024-08-31 |
+----------+----------------------+----------------+------------+------------+
13 rows in set (0.00 sec)
```

5)Insurance_policy

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE AddPolicyHolder(
    ->     IN p_Policy_HolderID INT,
    ->     IN p_Policy_Holder_Name VARCHAR(255),
    ->     IN p_Contact_Number VARCHAR(20),
    ->     IN p_Email VARCHAR(255),
    ->     IN p_Nominee_Name VARCHAR(255)
    -> )
    -> BEGIN
    ->     INSERT INTO policy_holder(Policy_HolderID, Policy_Holder_Name, Contact_Number, Email, Nominee_Name)
    ->     VALUES (p_Policy_HolderID, p_Policy_Holder_Name, p_Contact_Number, p_Email, p_Nominee_Name);
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> CALL AddPolicyHolder(1896,'Madhavv','984279872','madhavv139@gmail.com','Aneesha');
    -> //
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from policy_holder;
    -> ///
+-----------------+--------------------+----------------+--------------------------+-----------------+
| Policy_HolderID | Policy_Holder_Name | Contact_Number | Email                    | Nominee_Name    |
+-----------------+--------------------+----------------+--------------------------+-----------------+
|            1234 | Saran              |       98403977 | saran234@gmail.com       | Saratha         |
|            1235 | Haytham            |        1482285 | haytham234@gmail.com     | Akash           |
|            1236 | Dursan             |       15952822 | darshan@leo.com          | Leo             |
|            1237 | Harini             |       42814285 | ns200@gmail.com          | Lokesh          |
|            1238 | Akash              |       48128950 | gmailid@gmail.com        | Kamal           |
|            1239 | Sebin              |       42198152 | gayler69@gmail.com       | Rajnican't      |
|            1240 | Janaaki            |       69696969 | jaimaatha@gmail.com      | jaihind         |
|            1896 | Madhavv            |      984279872 | madhavv139@gmail.com     | Aneesha         |
|            2345 | Sarah White        |      765432109 | sarah.white@email.com    | Mike Johnson    |
|            3456 | Emily Wilson       |      543210987 | emily.wilson@email.com   | James Taylor    |
|            4567 | Jessica Clark      |      321098765 | jessica.clark@email.com  | Ethan Harris    |
|            6785 | Olivia Adams       |      789012345 | olivia.adams@email.com   | Noah Williams   |
|            6789 | David Brown        |      654321098 | david.brown@email.com    | Olivia Anderson |
|            7890 | Brian Davis        |      432109876 | brian.davis@email.com    | Lily Thompson   |
|            8901 | Michael Turner     |      210987654 | michael.turner@email.com | Sophia Martinez |
|            9012 | Ethan Moore        |      876501234 | ethan.moore@email.com    | Ava Rodriguez   |
+-----------------+--------------------+----------------+--------------------------+-----------------+
16 rows in set (0.00 sec)
```

Result: Thus, working with PL/SQL Procedures on MySQL is successfully verified.

Exp: 10	Working With PL/SQL Functions

Aim: To work with PL/SQL Functions on MySQL.

1) Function to Retrieve Policy Holder Information:

```
mysql> CREATE FUNCTION getPolicyHolderInfo(policy_holder_id INT) RETURNS VARCHAR(255)
    -> BEGIN
    ->     DECLARE holder_name VARCHAR(255);
    ->     SELECT Policy_Holder_Name INTO holder_name FROM policy_holder WHERE Policy_HolderID = policy_holder_id;
    ->     RETURN holder_name;
    -> END //
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SELECT getPolicyHolderInfo(1234);
+---------------------------+
| getPolicyHolderInfo(1234) |
+---------------------------+
| Saran                     |
+---------------------------+
1 row in set (0.00 sec)
```

2) Function to Update Claim Status:

```
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION updateClaimStatus(claim_id INT, new_status VARCHAR(50)) RETURNS BOOLEAN
    -> BEGIN
    ->     DECLARE success BOOLEAN;
    ->     UPDATE claim SET Claim_Status = new_status WHERE ClaimID = claim_id;
    ->     SET success = (ROW_COUNT() > 0);
    ->     RETURN success;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> SELECT updateClaimStatus(1122,'Approved');
+-----------------------------------+
| updateClaimStatus(1122,'Approved') |
+-----------------------------------+
|                                 1 |
+-----------------------------------+
1 row in set (0.01 sec)

mysql> select * from claim;
+---------+--------------+-----------------+------------+
| ClaimID | Claim_Status | Amount_Required | Date       |
+---------+--------------+-----------------+------------+
|    1122 | Approved     |           600.9 | 2023-10-25 |
|    1234 | Approved     |           26000 | 2023-08-21 |
|    1235 | Pending      |           80000 | 2023-10-01 |
|    1896 | Approved     |           10000 | 2023-10-25 |
|    2345 | Denied       |          300.75 | 2023-10-05 |
|    3344 | Approved     |          1700.4 | 2023-10-28 |
|    3456 | Pending      |          1200.8 | 2023-10-12 |
|    4567 | Approved     |          1800.6 | 2023-10-18 |
|    5678 | Approved     |           15000 | 2023-10-01 |
|    6789 | Approved     |         2000.25 | 2023-10-08 |
|    7890 | Denied       |           450.2 | 2023-10-15 |
|    8901 | Pending      |           950.3 | 2023-10-22 |
+---------+--------------+-----------------+------------+
12 rows in set (0.00 sec)
```

3) Function to Calculate Total Premium Collected by a Company:

```
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION calculateTotalPremiumByCompany(company_id INT) RETURNS DECIMAL(10,2)
    -> BEGIN
    ->     DECLARE total_premium DECIMAL(10,2);
    ->     SELECT SUM(Premium_Amount) INTO total_premium FROM insurance_policy WHERE Company_ID = company_id;
    ->     RETURN total_premium;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> Select calculateTotalPremiumByCompany(1234);
+-------------------------------------+
| calculateTotalPremiumByCompany(1234) |
+-------------------------------------+
|                           322210.00 |
+-------------------------------------+
1 row in set (0.00 sec)
```

4) Function to check Claim Eligibility:

```
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION isClaimEligible(claim_amount DECIMAL(10,2)) RETURNS BOOLEAN
    -> BEGIN
    ->     DECLARE eligibility BOOLEAN;
    ->     SET eligibility = (claim_amount <= Premium_Amount);
    ->     RETURN eligibility;
    -> END //
Query OK, 0 rows affected (0.01 sec)
```

5) Function to Calculate Premium Duration:

```
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION calculatePremiumDuration(start_date DATE, end_date DATE) RETURNS INT
    -> BEGIN
    ->     DECLARE duration INT;
    ->     SET duration = DATEDIFF(end_date, start_date);
    ->     RETURN duration;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> SELECT calculatePremiumDuration('2022-10-01','2023-09-02');
+-----------------------------------------------------+
| calculatePremiumDuration('2022-10-01','2023-09-02') |
+-----------------------------------------------------+
|                                                 336 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

Result: Thus, working with PL/SQL Functions on MySQL is successfully verified.

Exp: 11	Working With PL/SQL Cursors

Aim: To work with PL/SQL Cursors on MySQL.

1) List all insurance policies of a specific type:

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetPoliciesByType(IN policyType VARCHAR(255))
    -> BEGIN
    ->     DECLARE done INT DEFAULT FALSE;
    ->     DECLARE policy_id INT;
    ->     DECLARE policy_type VARCHAR(255);
    ->     DECLARE premium_amount DECIMAL(10, 2);
    ->     DECLARE start_date DATE;
    ->     DECLARE end_date DATE;
    ->
    ->     DECLARE cur CURSOR FOR
    ->         SELECT PolicyID, Policy_Type, Premium_Amount, Start_Date, End_Date
    ->         FROM insurance_policy
    ->         WHERE Policy_Type = policyType;
    ->
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    ->     OPEN cur;
    ->
    ->     read_loop: LOOP
    ->         FETCH cur INTO policy_id, policy_type, premium_amount, start_date, end_date;
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->         -- Process the data as needed
    ->         -- For example, you can print or return the data
    ->         SELECT policy_id, policy_type, premium_amount, start_date, end_date;
    ->     END LOOP;
    ->
    ->     CLOSE cur;
    -> END //
```

2) List all insurance companies.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetAllInsuranceCompanies()
    -> BEGIN
    ->     DECLARE done INT DEFAULT FALSE;
    ->     DECLARE company_id INT;
    ->     DECLARE company_name VARCHAR(255);
    ->     DECLARE address VARCHAR(255);
    ->     DECLARE contact_details VARCHAR(255);
    ->
    ->     DECLARE cur CURSOR FOR
    ->         SELECT Company_ID, Company_Name, Address, Contact_Details
    ->         FROM insurance_company;
    ->
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    ->     OPEN cur;
    ->
    ->     read_loop: LOOP
    ->         FETCH cur INTO company_id, company_name, address, contact_details;
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->         -- Process the data as needed
    ->         -- For example, you can print or return the data
    ->         SELECT company_id, company_name, address, contact_details;
    ->     END LOOP;
    ->
    ->     CLOSE cur;
    -> END //
```

3) List all claims with a specific status:

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetClaimsByStatus(IN claimStatus VARCHAR(255))
    -> BEGIN
    ->     DECLARE done INT DEFAULT FALSE;
    ->     DECLARE claim_id INT;
    ->     DECLARE claim_status VARCHAR(255);
    ->     DECLARE amount_required DECIMAL(10, 2);
    ->     DECLARE claim_date DATE;
    ->
    ->     DECLARE cur CURSOR FOR
    ->         SELECT ClaimID, Claim_Status, Amount_Required, Date
    ->         FROM claim
    ->         WHERE Claim_Status = claimStatus;
    ->
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    ->     OPEN cur;
    ->
    ->     read_loop: LOOP
    ->         FETCH cur INTO claim_id, claim_status, amount_required, claim_date;
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->         -- Process the data as needed
    ->         -- For example, you can print or return the data
    ->         SELECT claim_id, claim_status, amount_required, claim_date;
    ->     END LOOP;
    ->
    ->     CLOSE cur;
    -> END //
```

4) List all policy holders with their contact numbers:

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetPolicyHoldersWithContact()
    -> BEGIN
    ->     DECLARE done INT DEFAULT FALSE;
    ->     DECLARE holder_id INT;
    ->     DECLARE holder_name VARCHAR(255);
    ->     DECLARE contact_number VARCHAR(20);
    ->
    ->     DECLARE cur CURSOR FOR
    ->         SELECT Policy_HolderID, Policy_Holder_Name, Contact_Number
    ->         FROM policy_holder;
    ->
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    ->     OPEN cur;
    ->
    ->     read_loop: LOOP
    ->         FETCH cur INTO holder_id, holder_name, contact_number;
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->         -- Process the data as needed
    ->         -- For example, you can print or return the data
    ->         SELECT holder_id, holder_name, contact_number;
    ->     END LOOP;
    ->
    ->     CLOSE cur;
    -> END //
```

5) List all insurance policies that are active at a given date:

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetActivePoliciesAtDate(IN checkDate DATE)
    -> BEGIN
    ->     DECLARE done INT DEFAULT FALSE;
    ->     DECLARE policy_id INT;
    ->     DECLARE policy_type VARCHAR(255);
    ->     DECLARE premium_amount DECIMAL(10, 2);
    ->     DECLARE start_date DATE;
    ->     DECLARE end_date DATE;
    ->
    ->     DECLARE cur CURSOR FOR
    ->         SELECT PolicyID, Policy_Type, Premium_Amount, Start_Date, End_Date
    ->         FROM insurance_policy
    ->         WHERE Start_Date <= checkDate AND End_Date >= checkDate;
    ->
    ->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    ->
    ->     OPEN cur;
    ->
    ->     read_loop: LOOP
    ->         FETCH cur INTO policy_id, policy_type, premium_amount, start_date, end_date;
    ->         IF done THEN
    ->             LEAVE read_loop;
    ->         END IF;
    ->         -- Process the data as needed
    ->         -- For example, you can print or return the data
    ->         SELECT policy_id, policy_type, premium_amount, start_date, end_date;
    ->     END LOOP;
    ->
    ->     CLOSE cur;
    -> END //
```

Result: Thus, working with PL/SQL Cursors on MySQL is successfully verified.

Exp: 12	Working With PL/SQL Triggers

Aim: To work with PL/SQL Triggers on MySQL.

1) Trigger to update premium amount:

```
mysql> DELIMITER //
mysql> CREATE TRIGGER update_premium
    -> BEFORE INSERT ON insurance_policy
    -> FOR EACH ROW
    -> BEGIN
    ->     SET NEW.Premium_Amount = calculate_premium(NEW.Policy_Type);
    -> END;
    -> //
ERROR 1359 (HY000): Trigger already exists
mysql> DELIMITER ;
```

2) Trigger to check claim amount limits:

```
mysql> DELIMITER //
mysql> CREATE TRIGGER check_claim_limit
    -> BEFORE INSERT ON claim
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE policy_limit DECIMAL(10, 2);
    ->     SELECT Premium_Amount * 0.8 INTO policy_limit
    ->     FROM insurance_policy
    ->     WHERE PolicyID = NEW.PolicyID;
    ->
    ->     IF NEW.Amount_Required > policy_limit THEN
    ->         SIGNAL SQLSTATE '45000'
    ->         SET MESSAGE_TEXT = 'Claim amount exceeds policy limit';
    ->     END IF;
    -> END;
    -> //
```

3) Trigger to update claim status:

```
mysql> DELIMITER //
mysql> CREATE TRIGGER check_claim_limit
    -> BEFORE INSERT ON claim
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE policy_limit DECIMAL(10, 2);
    ->     SELECT Premium_Amount * 0.8 INTO policy_limit
    ->     FROM insurance_policy
    ->     WHERE PolicyID = NEW.PolicyID;
    ->
    ->     IF NEW.Amount_Required > policy_limit THEN
    ->         SIGNAL SQLSTATE '45000'
    ->         SET MESSAGE_TEXT = 'Claim amount exceeds policy limit';
    ->     END IF;
    -> END;
    -> //
```

4) Trigger to validate policy holder Email:

```
mysql> DELIMITER //
mysql> CREATE TRIGGER validate_email
    -> BEFORE INSERT ON policy_holder
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NOT REGEXP_LIKE(NEW.Email, '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}$') THEN
    ->         SIGNAL SQLSTATE '45000'
    ->         SET MESSAGE_TEXT = 'Invalid email format';
    ->     END IF;
    -> END;
    -> //
```

5) Trigger to archive expired policies:

```
mysql> DELIMITER //
mysql> CREATE TRIGGER archive_expired_policies
    -> BEFORE UPDATE ON insurance_policy
    -> FOR EACH ROW
    -> BEGIN
    ->     IF NEW.End_Date < CURDATE() THEN
    ->         INSERT INTO archived_policies (PolicyID, Policy_Type, Premium_Amount, Start_Date, End_Date)
    ->         VALUES (OLD.PolicyID, OLD.Policy_Type, OLD.Premium_Amount, OLD.Start_Date, OLD.End_Date);
    ->
    ->         DELETE FROM insurance_policy WHERE PolicyID = OLD.PolicyID;
    ->     END IF;
    -> END;
    -> //
```

Result: Thus, working with PL/SQL Triggers on MySQL is successfully verified.

Exp: 13	Project Report

INSURANCE MANAGEMENT SYSTEM

Introduction

Insurance management systems (IMS) are software applications that help insurance companies manage their business processes. IMS can be used to automate a variety of tasks, including policy administration, claims processing, and underwriting.

Problem statement

Insurance companies are still reliant on manual processes for many of their core business functions, such as policy administration, claims processing, and underwriting. It is required to build an insurance management system to automate various insurance-related processes and operations. The current manual system needs a centralized database that leads to data redundancy and consistency, which results in delays and errors, lacks in managing customer information and interactions, and leads to slow processing and time-consuming. The system should allow for efficient storage, retrieval, and manipulation of data to streamline insurance-related processes and enhance better customer service.

Overview

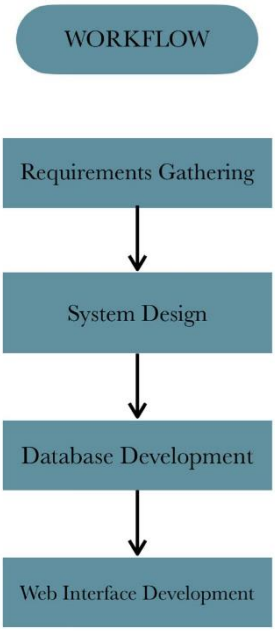
The Insurance Management System is a sophisticated software solution designed to revolutionize the way insurance companies operate and interact with their clients. This comprehensive platform serves as a central hub for managing various aspects of the insurance process, from policy creation to claims processing. It is built to address the unique challenges faced by the insurance industry and to capitalize on the opportunities presented by emerging technologies.

Objectives

The specific objectives of the project are to:

- Create a database to store insurance policy information, claims data, and other relevant data.
- Develop a web interface that allows users to view and manage their insurance policies, submit claims, and communicate with their insurance company.
- Implement security measures to protect user data.

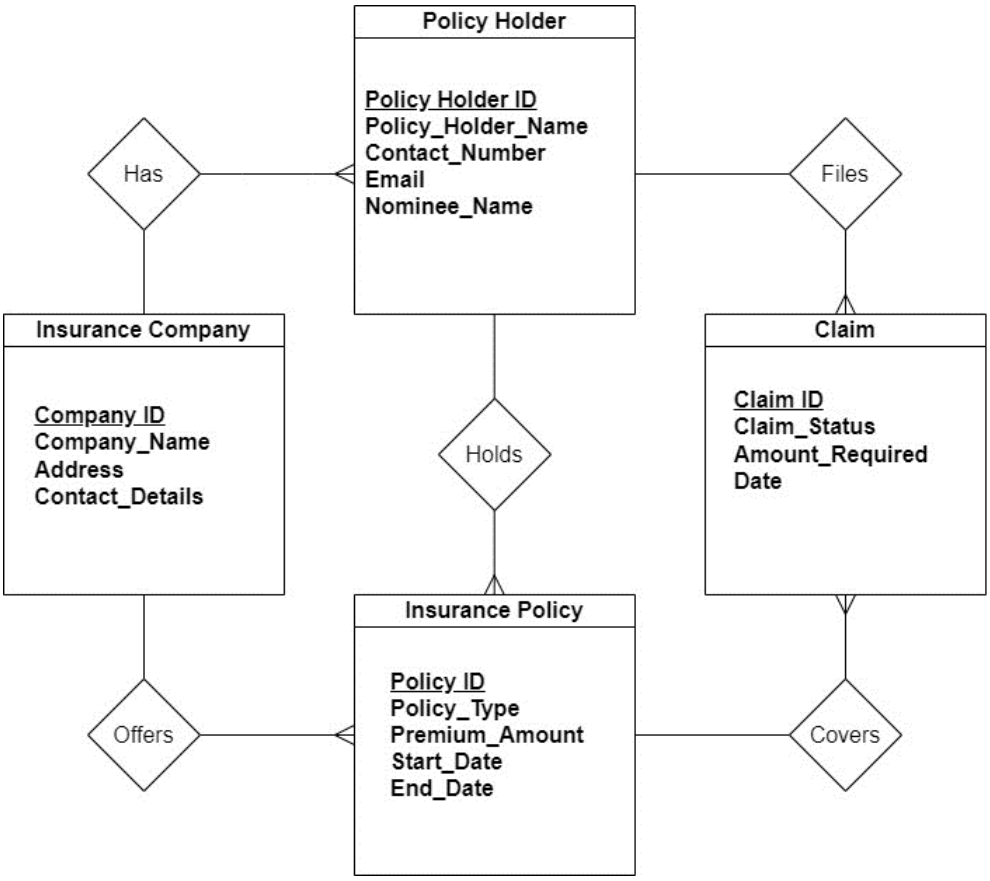
Workflow of Methodology of the project



1. Requirements gathering: The first step was to gather requirements from the users of the website. This was done by conducting interviews and surveys with insurance agents, policyholders, and claims adjusters.

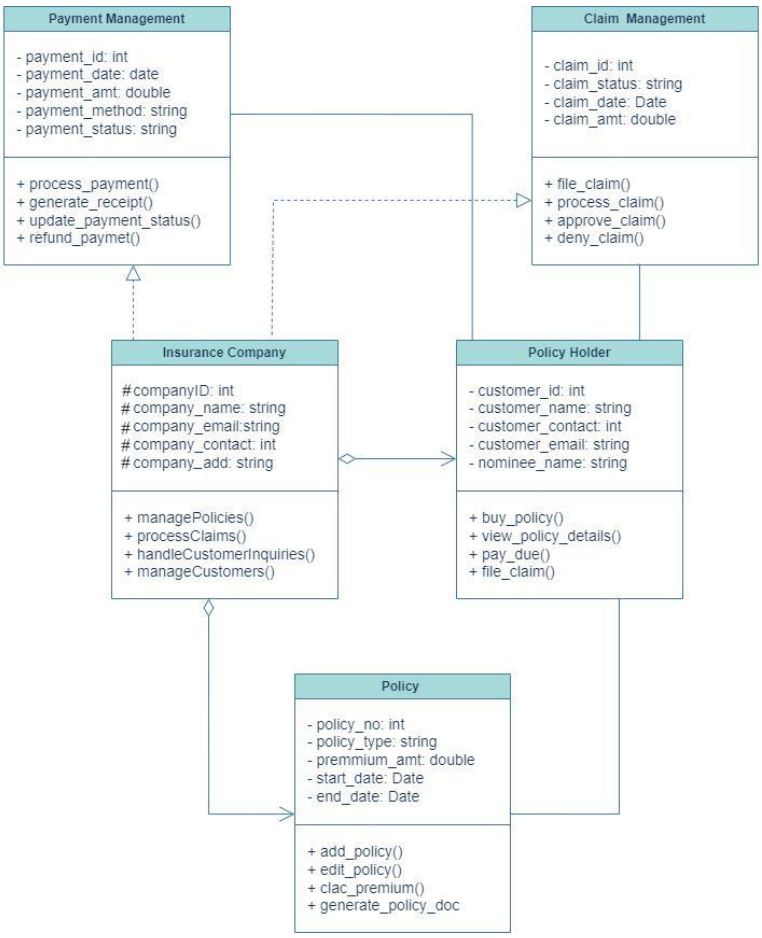
2. System design: Once the requirements were gathered, a system design was created. This design included the database schema, the web interface architecture, and the security measures that would be implemented.

I. ER DIAGRAM:

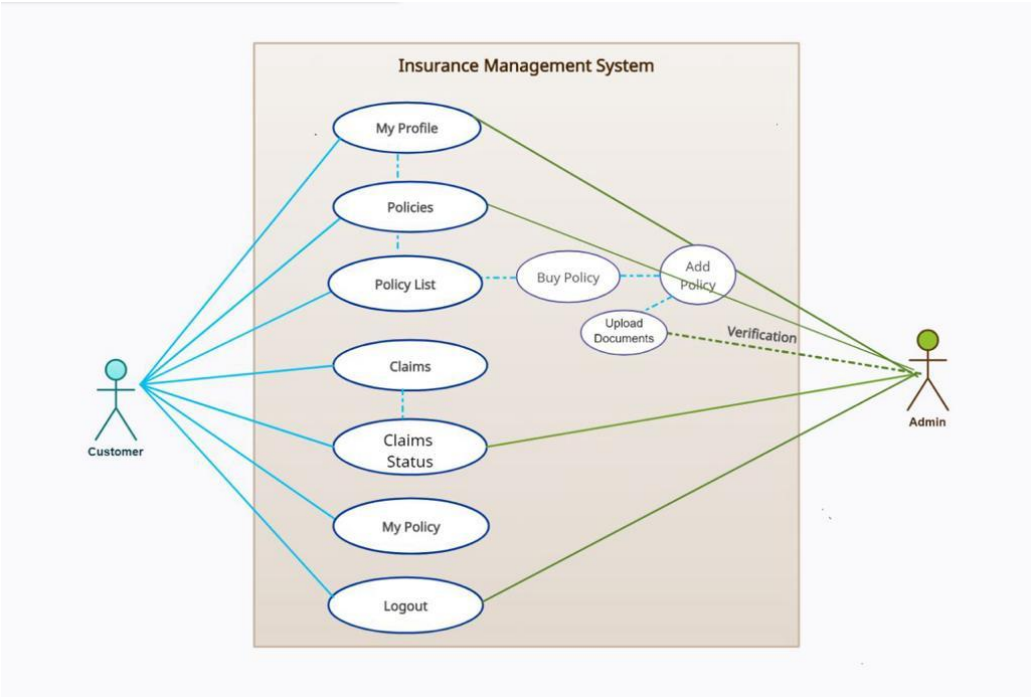


II. UML DIAGRAMS:

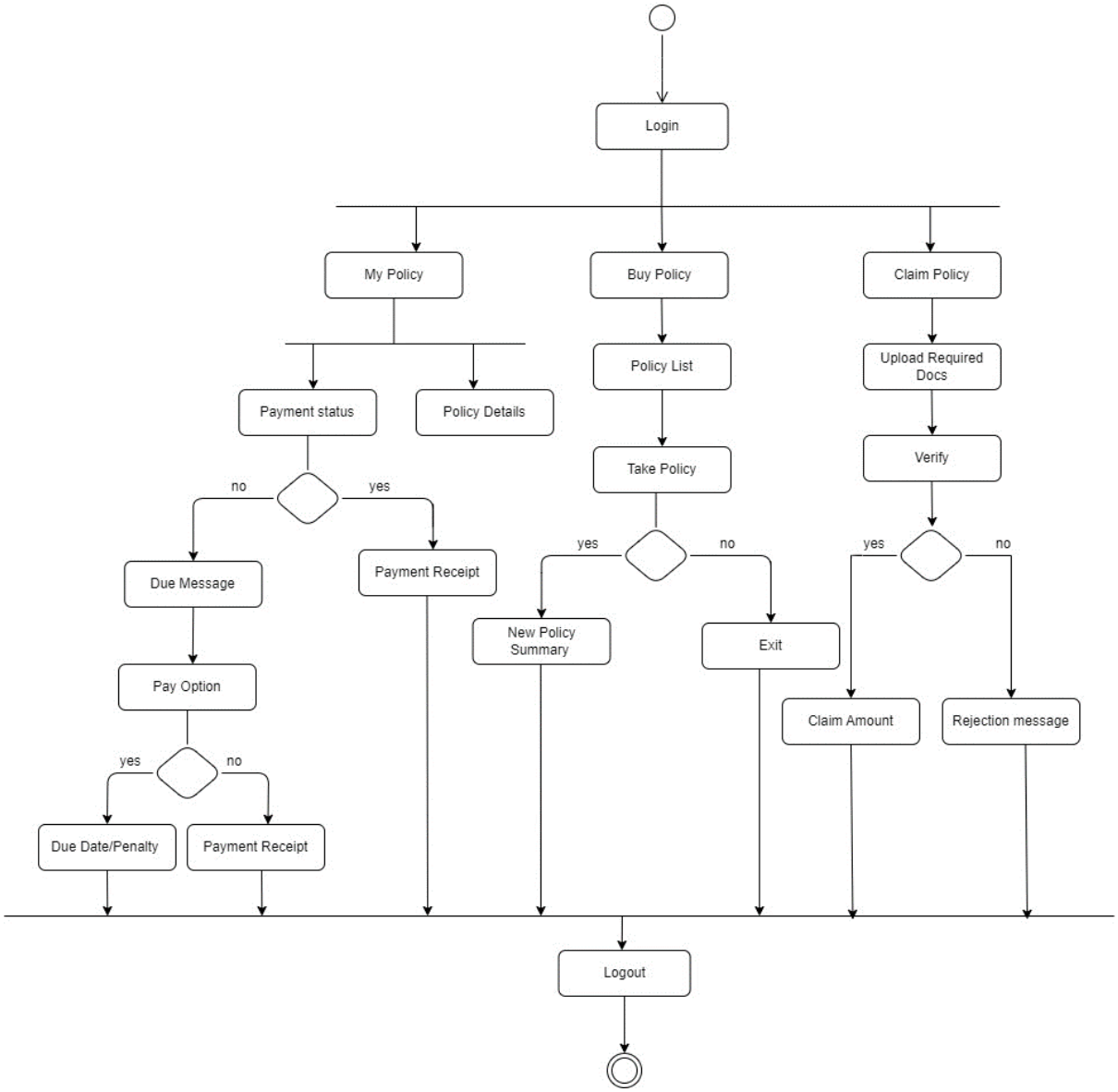
- **Class Diagram**



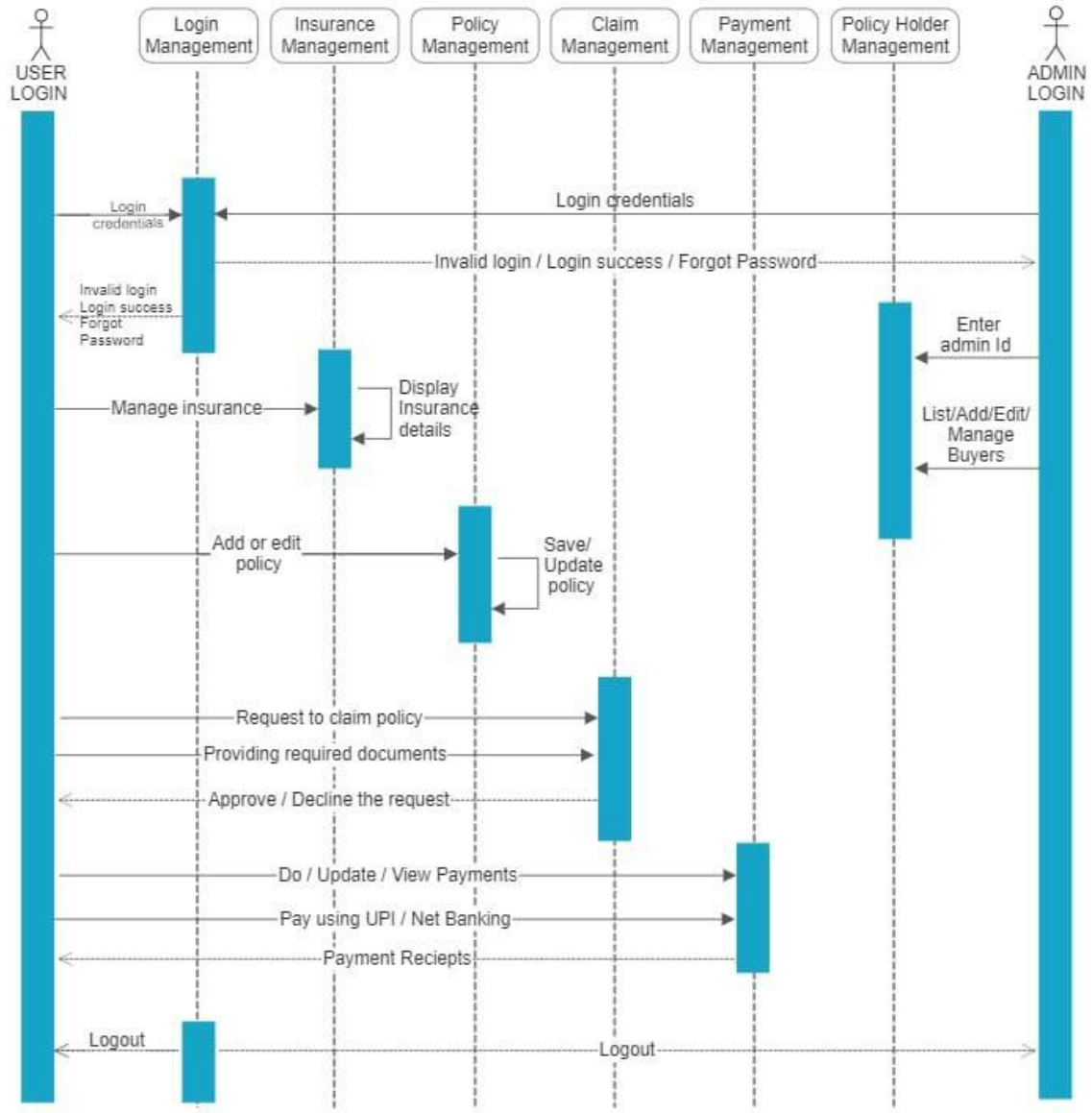
- **Use Case Diagram**



- **Activity Diagram**



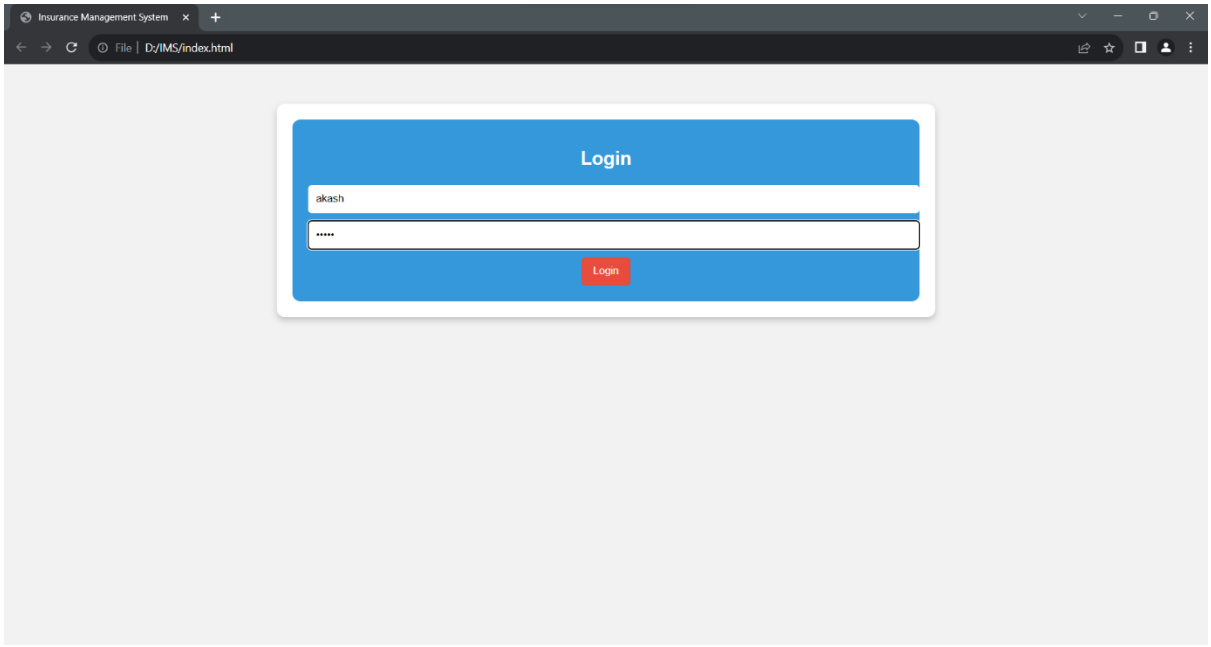
- **Sequence Diagram**

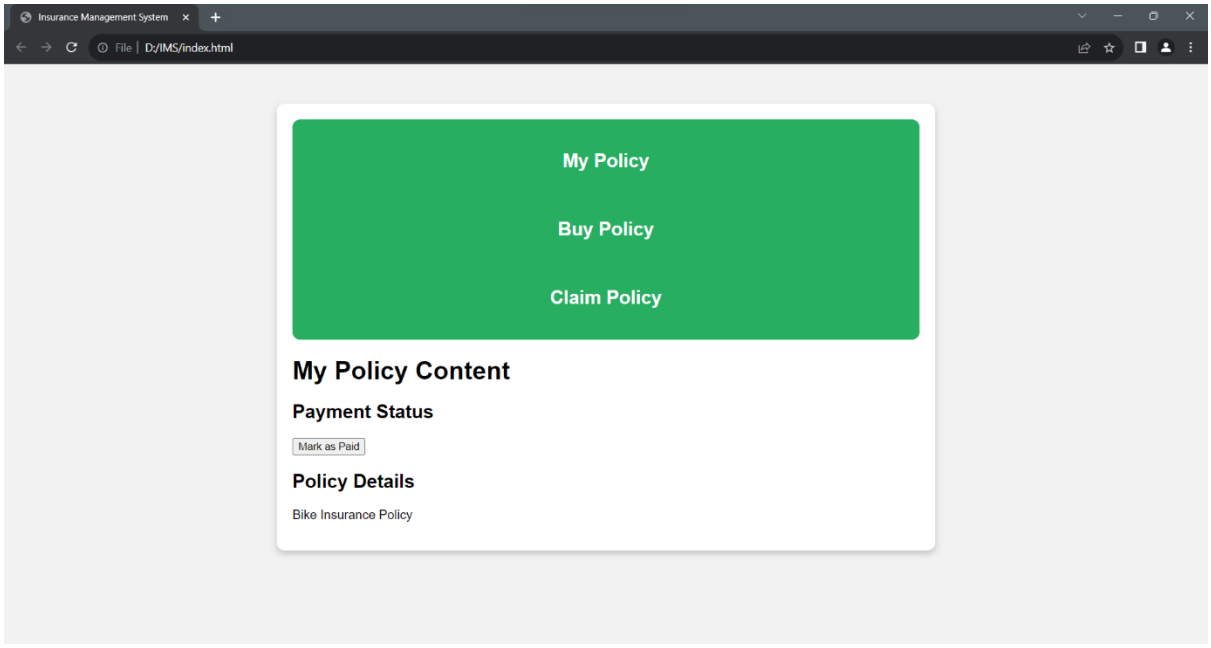


3. Database development: The next step was to develop the database. This was done using SQL.

4. Web interface development: The web interface was developed using HTML, CSS, and JavaScript.

Primitive Front-end design of IMS:





Results

A basic IMS website was successfully developed using SQL, HTML, CSS, and JavaScript. The website is user-friendly and efficient, and it allows users to view and manage their insurance policies, submit claims, and communicate with their insurance company.

Inferences

The following inferences can be drawn from the project:

- SQL is a powerful language for developing and managing databases.
- HTML, CSS, and JavaScript is essential for developing web interfaces.
- It is important to implement security measures to protect user data.
- Thorough testing is essential before deploying a website.

Conclusion

A basic IMS website was successfully developed using SQL, HTML, CSS, and JavaScript. The website is a valuable tool for insurance companies and their customers.

Future works

The following are some possible areas for future work:

- Implement additional features, such as the ability to generate reports and make payments.
- Develop a mobile app for the IMS website.
- Integrate the IMS website with other insurance systems, such as underwriting systems and claims processing systems.