CYCLE-1

SQL PRACTICE QUESTIONS

I. Create a table with following columns.

ID	character	5
DeptID	numeric	2
Name	character	15
Design	character	15
Basic	numeric	10,2
Gender	character	1

ID	DeptID	Name	Designation	Basic	Gender
101	1	Ram	Typist	2000	M
102	2	Arun	Analyst	6000	M
121	1	Ruby	Typist	2010	F
156	3	Mary	Manager	4500	F
123	2	Mridula	Analyst	6000	F
114	4	Menon	Clerk	1500	M
115	4	Tim	Clerk	1500	M
127	2	Kiran	Manager	4000	M
147	2	Maya	Clerk	4000	F

```
user@user-Desktop:~$ sudo mysql
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.34-0ubuntu0.22.04.1 (Ubuntu)
Copyright (c) 2000, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
Owners.

mysql> create database vidya_2024;
Query OK, 1 row affected (0.03 sec)
mysql> use vidya_2024;
Database changed
mysql> create table Employee_57(ID varchar(5) primary key,DeptID
varchar(5),Name varchar(15),Designation varchar(15),Basic
decimal(10,2),Gender varchar(1));
Query OK, 0 rows affected (0.07 sec)
```

1. Get the description of the table.

```
mysql> desc Employee_57;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| ID          | varchar(5)    | NO   | PRI | NULL    |       |
| DeptID      | varchar(5)    | YES  |     | NULL    |       |
| Name        | varchar(15)   | YES  |     | NULL    |       |
| Designation | varchar(15)   | YES  |     | NULL    |       |
| Basic       | decimal(10,2) | YES  |     | NULL    |       |
| Gender      | varchar(1)    | YES  |     | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
6 rows in set (0.03 sec)
```

2. Display all the records from the above table.

```
mysql>select * from Employee_57;
+-----+--------+---------+-------------+---------+--------+
| ID  | DeptID | Name    | Designation | Basic   | Gender |
+-----+--------+---------+-------------+---------+--------+
| 101 | 1      | Ram     | Typist      | 2000.00 | M      |
| 102 | 2      | Arun    | Analyst     | 6000.00 | M      |
| 114 | 4      | Menon   | Clerk       | 1500.00 | M      |
| 115 | 4      | Tim     | Clerk       | 1500.00 | M      |
| 121 | 1      | Ruby    | Typist      | 2010.00 | F      |
| 123 | 2      | Mridula | Analyst     | 6000.00 | F      |
| 127 | 2      | Kiran   | Manager     | 4000.00 | M      |
| 156 | 2      | Maya    | Clerk       | 4000.00 | F      |
+-----+--------+---------+-------------+---------+--------+
8 rows in set (0.01 sec)
```

3. Display the ID, name, designation and basic salary of all the employees.

```
mysql> select ID,Name,Designation,Basic from Employee_57;
+-----+---------+-------------+---------+
| ID  | Name    | Designation | Basic   |
+-----+---------+-------------+---------+
| 101 | Ram     | Typist      | 2000.00 |
| 102 | Arun    | Analyst     | 6000.00 |
| 114 | Menon   | Clerk       | 1500.00 |
| 115 | Tim     | Clerk       | 1500.00 |
| 121 | Ruby    | Typist      | 2010.00 |
| 123 | Mridula | Analyst     | 6000.00 |
| 127 | Kiran   | Manager     | 4000.00 |
| 156 | Maya    | Clerk       | 4000.00 |
+-----+---------+-------------+---------+
8 rows in set (0.01 sec)
```

4. Display ID and name of all the employees from department no.2

```
mysql> select ID,Name from Employee_57 where DeptID='2';
+-----+---------+
| ID  | Name    |
+-----+---------+
| 102 | Arun    |
| 123 | Mridula |
| 127 | Kiran   |
| 156 | Maya    |
+-----+---------+
3 rows in set (0.01 sec)
```

5. Display ID, name, designation,deptID and basic, DA, HRA and net salary of all employees with suitable headings as DA, HRA and NET_SAL respectively.(DA is 7.5% of basic, and NET_SAL is Basic + DA + HRA)

```
mysql>select ID, Name, Designation, DeptID, Basic,(Basic*0.075) AS
DA,(Basic * 0.10) AS HRA,(Basic + (Basic * 0.075) + (Basic * 0.10)) AS
NET_SAL from Employee_57;

+-----+---------+-------------+--------+---------+-----------+----------+------------+
| ID  | Name    | Designation | DeptID | Basic   | DA        | HRA      | NET_SAL    |
+-----+---------+-------------+--------+---------+-----------+----------+------------+
| 101 | Ram     | Typist      | 1      | 2000.00 | 150.00000 | 200.0000 | 2350.00000 |
| 102 | Arun    | Analyst     | 2      | 6000.00 | 450.00000 | 600.0000 | 7050.00000 |
| 114 | Menon   | Clerk       | 4      | 1500.00 | 112.50000 | 150.0000 | 1762.50000 |
| 115 | Tim     | Clerk       | 4      | 1500.00 | 112.50000 | 150.0000 | 1762.50000 |
| 121 | Ruby    | Typist      | 1      | 2010.00 | 150.75000 | 201.0000 | 2361.75000 |
| 123 | Mridula | Analyst     | 2      | 6000.00 | 450.00000 | 600.0000 | 7050.00000 |
| 127 | Kiran   | Manager     | 2      | 4000.00 | 300.00000 | 400.0000 | 4700.00000 |
| 156 | Mary    | Manager     | 3      | 4500.00 | 337.50000 | 450.0000 | 5287.50000 |
+-----+---------+-------------+--------+---------+-----------+----------+------------+
8 rows in set (0.01 sec)
```

6. Display ID, name, designation, deptID and basic salary in the descending order of basic pay.

```
mysql> select ID, Name, Designation, DeptID, Basic from employee_57 order
by Basic desc;
+-----+---------+-------------+--------+---------+
| ID  | Name    | Designation | DeptID | Basic   |
+-----+---------+-------------+--------+---------+
| 102 | Arun    | Analyst     | 2      | 6000.00 |
| 123 | Mridula | Analyst     | 2      | 6000.00 |
| 156 | Mary    | Manager     | 3      | 4500.00 |
| 127 | Kiran   | Manager     | 2      | 4000.00 |
| 156 | Maya    | Clerk       | 2      | 4000.00 |
| 121 | Ruby    | Typist      | 1      | 2010.00 |
| 101 | Ram     | Typist      | 1      | 2000.00 |
| 114 | Menon   | Clerk       | 4      | 1500.00 |
| 115 | Tim     | Clerk       | 4      | 1500.00 |
+-----+---------+-------------+--------+---------+
```

```
8 rows in set (0.00 sec)
```

7. Display the employees whose designation is TYPIST.

```
mysql> select * from Employee_57 where Designation = 'Typist';
+-----+--------+------+-------------+---------+--------+
| ID  | DeptID | Name | Designation | Basic   | Gender |
+-----+--------+------+-------------+---------+--------+
| 101 | 1      | Ram  | Typist      | 2000.00 | M      |
| 121 | 1      | Ruby | Typist      | 2010.00 | F      |
+-----+--------+------+-------------+---------+--------+
2 rows in set (0.00 sec)
```

8. Display all details of employees whose designation is either ANALYST or MANAGER.

```
mysql> select * from Employee_57 where Designation in
('Analyst','Manager');
+-----+--------+---------+-------------+---------+--------+
| ID  | DeptID | Name    | Designation | Basic   | Gender |
+-----+--------+---------+-------------+---------+--------+
| 102 | 2      | Arun    | Analyst     | 6000.00 | M      |
| 123 | 2      | Mridula | Analyst     | 6000.00 | F      |
| 127 | 2      | Kiran   | Manager     | 4000.00 | M      |
| 156 | 3      | Mary    | Manager     | 4500.00 | F      |
+-----+--------+---------+-------------+---------+--------+
4 rows in set (0.00 sec)
```

9. Display all designations without duplicate values.

```
mysql> select distinct Designation from Employee_57;

+-------------+
| Designation |
+-------------+
| Typist      |
| Analyst     |
| Clerk       |
| Manager     |
+-------------+
4 rows in set (0.00 sec)
```

10. Display the ID, name, department and basic of all the employees who are either MANAGER or CLERK and the basic salary is in the range of 1400 and 4500.

```
mysql> select ID, Name, DeptID, Basic from Employee_57 where Designation in
('Manager', 'Clerk') and Basic between 1400 and 4500;
```

```
+-----+-------+--------+---------+
| ID  | Name  | DeptID | Basic   |
+-----+-------+--------+---------+
| 114 | Menon | 4      | 1500.00 |
| 115 | Tim   | 4      | 1500.00 |
| 127 | Kiran | 2      | 4000.00 |
| 156 | Mary  | 3      | 4500.00 |
+-----+-------+--------+---------+
4 rows in set (0.00 sec)
```

11. Display the number of male staff members.

```
mysql> select count(*) as Male_count from Employee_57 where Gender = 'M';
+------------+
| Male_count |
+------------+
|          5 |
+------------+
1 row in set (0.01 sec)
```

12. Find the maximum salary of each designation.

```
mysql> select Designation, max(Basic) as Maximum_Salary from Employee_57
group by Designation;
+-------------+----------------+
| Designation | Maximum_Salary |
+-------------+----------------+
| Typist      |    2010.00     |
| Analyst     |    6000.00     |
| Clerk       |    1500.00     |
| Manager     |    4500.00     |
+-------------+----------------+
4 rows in set (0.01 sec)
```

13. Add a column manager-id into the above table.

```
alter table Employee_57 add Manager_ID varchar(5);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

14. Update values of manager id of employees as null for 101, 101 for 102, 121, 156. 102 for 123,114,115.121 for 127.

```
mysql>alter table Employee_57 add Manager_ID varchar(5);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> update Employee_57 set Manager_ID = null where ID = '101';
Query OK, 0 rows affected (0.00 sec)
```

```
Rows matched: 1  Changed: 0  Warnings: 0

mysql> update Employee_57 set Manager_ID = '101' where ID in ('102', '121',
'156');
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> update Employee_57 set Manager_ID = '102' where ID in ('123', '114',
'115');
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> update Employee_57 set Manager_ID = '121' where ID = '127';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

15. Display the manager id of the employee Ram.

```
mysql> select Manager_ID from Employee_57 where Name = 'Ram';

+------------+
| Manager_ID |
+------------+
| NULL       |
+------------+
1 row in set (0.00 sec)
```

16. Display the employee names and their manager name.

```
mysql> select E.Name as Employee_Name, M.Name as Manager_Name from
Employee_57 E,Employee_57 M where E.Manager_ID = M.ID;

+---------------+--------------+
| Employee_Name | Manager_Name |
+---------------+--------------+
| Ram           | NULL         |
| Arun          | Ram          |
| Menon         | Arun         |
| Tim           | Arun         |
| Ruby          | Ram          |
| Mridula       | Arun         |
| Kiran         | Ruby         |
| Mary          | Ram          |
+---------------+--------------+
8 rows in set (0.00 sec)
```

17. Find the average salary of each department.

```
mysql> select DeptID, avg(Basic) as Avg_Salary from Employee_57 group by
DeptID;
```

```
+--------+-------------+
| DeptID | Avg_Salary  |
+--------+-------------+
| 1      | 2005.000000 |
| 2      | 5333.333333 |
| 4      | 1500.000000 |
| 3      | 4500.000000 |
+--------+-------------+
4 rows in set (0.01 sec)
```

18. Find the maximum salary given to employees.

```
mysql> select max(Basic) as Max_Salary from Employee_57;
+------------+
| Max_Salary |
+------------+
|    6000.00 |
+------------+
1 row in set (0.01 sec)
```

19. Find the number of employees in each department.

```
mysql> select DeptID, count(*) as Emp_Count from Employee_57 group by
DeptID;
+--------+-------------+
| DeptID |  Emp_Count  |
+--------+-------------+
| 1      |          2  |
| 2      |          3  |
| 4      |          2  |
| 3      |          1  |
+--------+-------------+
4 rows in set (0.00 sec)
```

20. Find the number of departments existing in the organisation.

```
mysql> select count(distinct DeptID) as Dept_Count from Employee_57;
+------------+
| Dept_Count |
+------------+
|          4 |
+------------+
1 row in set (0.01 sec)
```

21. Display the different designations existing in the organisation.

```
mysql> select distinct Designation from Employee_57;
```

```
+-------------+
| Designation |
+-------------+
| Typist      |
| Analyst     |
| Clerk       |
| Manager     |
+-------------+
4 rows in set (0.00 sec)
```

22. Display the number of different designations existing in the organisation.

```
mysql> select count(distinct Designation) as Designation_Count from
Employee_57;
+-------------------+
| Designation_Count |
+-------------------+
|                 4 |
+-------------------+
1 row in set (0.00 sec)
```

23. Display the maximum salary given for female employees.

```
mysql> select max(Basic) as Max_Female_Sal from Employee_57 where Gender =
'F';
+-------------------+
|  Max_Female_Sal   |
+-------------------+
|           6000.00 |
+-------------------+
1 row in set (0.00 sec)
```

24. Display the female typist.

```
mysql> select * from Employee_57 where Designation = 'Typist' and Gender =
'F';
+-----+--------+------+-------------+---------+--------+------------+
| ID  | DeptID | Name | Designation | Basic   | Gender | Manager_ID |
+-----+--------+------+-------------+---------+--------+------------+
| 121 | 1      | Ruby | Typist      | 2010.00 | F      | 101        |
+-----+--------+------+-------------+---------+--------+------------+
1 row in set (0.00 sec)
```

25. Display the male clerks getting salaries more than 3000.

```
mysql> select * from Employee_57 where Designation = 'Clerk' and Gender =
'M' and Basic > 3000;
```

```
+-----+--------+------+-------------+---------+--------+------------+
| ID  | DeptID | Name | Designation | Basic   | Gender | Manager_ID |
+-----+--------+------+-------------+---------+--------+------------+
| 147 | 2      | Amal | Clerk       | 4000.00 | M      | 101        |
+-----+--------+------+-------------+---------+--------+------------+
1 row in set (0.00 sec)
```

26. Display the details of managers or analysts working for dept id 2.

```
mysql> select * from Employee_57 where DeptID = '2' and Designation in
('Manager', 'Analyst');

+-----+--------+---------+-------------+---------+--------+------------+
| ID  | DeptID | Name    | Designation | Basic   | Gender | Manager_ID |
+-----+--------+---------+-------------+---------+--------+------------+
| 102 | 2      | Arun    | Analyst     | 6000.00 | M      | 101        |
| 123 | 2      | Mridula | Analyst     | 6000.00 | F      | 102        |
| 127 | 2      | Kiran   | Manager     | 4000.00 | M      | 121        |
+-----+--------+---------+-------------+---------+--------+------------+
3 rows in set (0.00 sec)
```

27. Display the designation and salary of Ruby.

```
mysql> select Designation, Basic from Employee_57 where Name = 'Ruby';
+-------------+---------+
| Designation | Basic   |
+-------------+---------+
| Typist      | 2010.00 |
+-------------+---------+
1 row in set (0.00 sec)
```

28. Add a column joining date to the above table.

```
mysql> alter table Employee_57 add Join_Date date;
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

29. Update appropriate values for the joining date field.

```
mysql> update Employee_57 set Join_Date = '2022-02-15' where ID = '102';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0


mysql> update Employee_57 set Join_Date = '2022-03-05' where ID = '114';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> update Employee_57 set Join_Date = '2022-03-20' where ID = '115';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update Employee_57 set Join_Date = '2022-04-10' where ID = '121';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update Employee_57 set Join_Date = '2022-05-25' where ID = '123';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update Employee_57 set Join_Date = '2022-06-01' where ID = '127';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update Employee_57 set Join_Date = '2022-07-15' where ID = '147';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update Employee_57 set Join_Date = '2022-08-10' where ID = '156';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

30. Display the details of employees according to their seniority

```
mysql> select * from Employee_57 order by Join_Date;

+-----+--------+---------+-------------+---------+--------+------------+--------------+
| ID  | DeptID | Name    | Designation | Basic   | Gender | Manager_ID | Join_Date    |
+-----+--------+---------+-------------+---------+--------+------------+--------------+
| 101 | 1      | Ram     | Typist      | 2000.00 | M      | NULL       | 2022-01-10   |
| 102 | 2      | Arun    | Analyst     | 6000.00 | M      | 101        | 2022-02-15   |
| 114 | 4      | Menon   | Clerk       | 1500.00 | M      | 102        | 2022-03-05   |
| 115 | 4      | Tim     | Clerk       | 1500.00 | M      | 102        | 2022-03-20   |
| 121 | 1      | Ruby    | Typist      | 2010.00 | F      | 101        | 2022-04-10   |
| 123 | 2      | Mridula | Analyst     | 6000.00 | F      | 102        | 2022-05-25   |
| 127 | 2      | Kiran   | Manager     | 4000.00 | M      | 121        | 2022-06-01   |
| 147 | 2      | Amal    | Clerk       | 4000.00 | M      | 101        | 2022-07-15   |
| 156 | 3      | Mary    | Manager     | 4500.00 | F      | 101        | 2022-08-10   |
+-----+--------+---------+-------------+---------+--------+------------+------------+
9 rows in set (0.00 sec)
```

31. Display the details of employees according to the descending order of their salaries.

```
mysql> select * from Employee_57 order by Basic desc;
+-----+--------+---------+-------------+---------+--------+------------+--------------+
| ID  | DeptID | Name    | Designation | Basic   | Gender | Manager_ID | Join_Date    |
+-----+--------+---------+-------------+---------+--------+------------+--------------+
| 102 | 2      | Arun    | Analyst     | 6000.00 | M      | 101        | 2022-02-15   |
| 123 | 2      | Mridula | Analyst     | 6000.00 | F      | 102        | 2022-05-25   |
| 156 | 3      | Mary    | Manager     | 4500.00 | F      | 101        | 2022-08-10   |
| 127 | 2      | Kiran   | Manager     | 4000.00 | M      | 121        | 2022-06-01   |
| 147 | 2      | Amal    | Clerk       | 4000.00 | M      | 101        | 2022-07-15   |
| 121 | 1      | Ruby    | Typist      | 2010.00 | F      | 101        | 2022-04-10   |
| 101 | 1      | Ram     | Typist      | 2000.00 | M      | NULL       | 2022-01-10   |
| 114 | 4      | Menon   | Clerk       | 1500.00 | M      | 102        | 2022-03-05   |
| 115 | 4      | Tim     | Clerk       | 1500.00 | M      | 102        | 2022-03-20   |
+-----+--------+---------+-------------+---------+--------+------------+--------------+
9 rows in set (0.00 sec)
```

32. Create a new table DEPARTMENT with fields DEPTID and DNAME. Make DEPTID as the
primary key.

```
mysql> create table Department_57 (
      -> DeptID varchar(2) primary key,
      -> DName varchar(15)
      -> );
Query OK, 0 rows affected (0.01 sec)

mysql> desc Department_57;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| DeptID | varchar(2)  | NO   | PRI | NULL    |       |
| DName  | varchar(15) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

33. Make DEPTID in the employee table to refer to the DEPARTMENT table.

```
mysql> alter table Employee_57
      -> add constraint Dept_fk
      -> foreign key (DeptID) references Department_57(DeptID);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

34. Insert values into the DEPARTMENT table. Make sure that all the existing values for DEPTID in employees are inserted into this table. Sample values are DESIGN,CODING,TESTING,RESEARCH.

```
mysql> insert into Department_57 (DeptID, DName) values ('1', 'DESIGN'),
('2', 'CODING'), ('3', 'TESTING'), ('4', 'RESEARCH');
Query OK, 4 rows affected (0.01 sec)
```

```
Records: 4  Duplicates: 0  Warnings: 0
```

35. Display the employee name and department name.

```
mysql> select E.Name as Employee_Name, D.DName as Department_Name from
Employee_57 E,Department_57 D where E.DeptID = D.DeptID;
+---------------+-----------------+
| Employee_Name | Department_Name |
+---------------+-----------------+
| Ram           | DESIGN          |
| Arun          | CODING          |
| Menon         | RESEARCH        |
| Tim           | RESEARCH        |
| Ruby          | DESIGN          |
| Mridula       | CODING          |
| Kiran         | CODING          |
| Amal          | CODING          |
| Mary          | TESTING         |
+---------------+-----------------+
9 rows in set (0.00 sec)
```

36. Display the department name of employee Arun.

```
mysql> select DName from Department_57 where DeptID in (select DeptID from
Employee_57 where Name='Arun');
+-----------------+
| Department_Name |
+-----------------+
| CODING          |
+-----------------+
1 row in set (0.00 sec)
```

37. Display the salary given by the DESIGN department.

```
mysql> select Name,Basic as Salary from Employee_57 where DeptID in (select
DeptID from Department_57 where Dname='DESIGN');
+-------+---------+
| Name  | Salary  |
+-------+---------+
| Ram   | 2000.00 |
| Ruby  | 2010.00 |
+-------+---------+
2 rows in set (0.00 sec)
```

38. Display the details of typists working in the DESIGN department.

```
mysql> select ID,Name,Designation,Basic from Employee_57 where
Designation='Typist' and DeptID in(select DeptID from Department_57 where
Dname='DESIGN');
```

```
+-----+-------+-------------+---------+--------+
| ID  | Name  | Designation | Basic   | Gender |
+-----+-------+-------------+---------+--------+
| 101 | Ram   | Typist      | 2000.00 | M      |
| 121 | Ruby  | Typist      | 2010.00 | F      |
+-----+-------+-------------+---------+--------+
2 rows in set (0.00 sec)
```

39. Display the salary of employees working in the RESEARCH department.

```
mysql> select Name,Basic as Salary from Employee_57 where DeptID in(select
DeptID from Department_57 where Dname='RESEARCH');
+-------+---------+
| Name  | Salary  |
+-------+---------+
| Menon | 1500.00 |
| Tim   | 1500.00 |
+-------+---------+
2 rows in set (0.00 sec)
```

40. List the female employees working in the TESTING department.

```
mysql> select ID,Name,Designation,Basic,Gender from Employee_57 where
Gender='F' and DeptID in(select DeptID from Department_57 where
Dname='TESTING');
+-----+------+-------------+---------+--------+
| ID  | Name | Designation | Basic   | Gender |
+-----+------+-------------+---------+--------+
| 156 | Mary | Manager     | 4500.00 | F      |
+-----+------+-------------+---------+--------+
1 row in set (0.00 sec)
```

41. Display the details of employees not working in the CODING or TESTING department.

```
mysql> select E.ID, E.Name, E.Designation, E.Basic, E.Gender, D.DName as
Department_Name from Employee_57 E,Department_57 D where E.DeptID =
D.DeptID and D.DName not in ('CODING', 'TESTING');

+-----+-------+-------------+---------+--------+-----------------+
| ID  | Name  | Designation | Basic   | Gender | Department_Name |
+-----+-------+-------------+---------+--------+-----------------+
| 101 | Ram   | Typist      | 2000.00 | M      | DESIGN          |
| 121 | Ruby  | Typist      | 2010.00 | F      | DESIGN          |
| 156 | Mary  | Manager     | 4500.00 | F      | TESTING         |
+-----+-------+-------------+---------+--------+-----------------+
3 rows in set (0.00 sec)
```

42. Display the names of departments giving maximum salary.

```
mysql> select DName from Department_57 where DeptID in(select DeptID from
Employee_57 where Basic in(select max(Basic)from Employee_57));
+--------+
| DName  |
+--------+
| CODING |
| TESTING|
+--------+
2 rows in set (0.00 sec)
```

43. Display the names of departments with a minimum number of employees.

```
mysql> select D.DName
      -> from Employee_57 E,Department_57 D
      -> where E.DeptID = D.DeptID
      -> group by D.DName
      -> having count(E.ID) = (select min(emp_count) from
        (select count(ID) as emp_count from Employee_57 group by DeptID) as
dept_counts);
+--------+
| DName  |
+--------+
| DESIGN |
+--------+
1 row in set (0.00 sec)
```

44. Display the second maximum salary.

```
mysql> select max(Basic)from Employee_57 where Basic<(select max(Basic)from
Employee_57);
+---------+
| Basic   |
+---------+
| 2010.00 |
+---------+
1 row in set (0.00 sec)
```

45. Display the second minimum salary

```
mysql> select min(Basic)from Employee_57 where Basic>(select min(Basic)from
Employee_57);

+---------+
| Basic   |
+---------+
| 1500.00 |
+---------+
1 row in set (0.00 sec)
```

46. Display the names of employees getting salaries greater than the average salary of their department.

```
mysql> select E1.Name from Employee_57 E1 where E1.Basic >  (select
avg(Basic) from Employee_57 E2 where E1.DeptID = E2.DeptID);
+----------+
| Name     |
+----------+
| Arun     |
| Mridula  |
| Kiran    |
| Mary     |
+----------+
4 rows in set (0.00 sec)
```

47. Display the names of employees working under the manager Ram.

```
mysql> select Name from Employee_57 where Manager_ID = (select ID from
Employee_57 where Name = 'Ram');
+----------+
| Name     |
+----------+
| Arun     |
| Ruby     |
| Menon    |
| Tim      |
| Kiran    |
| Amal     |
+----------+
6 rows in set (0.00 sec)
```

48. Display the deptid and total number of employees as "Number of Dept_Employees" for only those departments with more than 3 employees.

```
mysql> select DeptID, count(*) as 'Number of Dept_Employees' from
Employee_57 group by DeptID having count(*) > 3;
+--------+-------------------------+
| DeptID | Number of Dept_Employees |
+--------+-------------------------+
| 2      | 4                       |
+--------+-------------------------+
1 row in set (0.00 sec)
```

49. Display the deptid and minimum salary as "Lowest Salary" for those departments with minimum salary above 2500.

```
mysql> select DeptID, min(Basic) as 'Lowest Salary'
      -> from Employee_57
      -> group by DeptID
      -> having min(Basic) > 2500;
```

```
+--------+---------------+
| DeptID | Lowest Salary |
+--------+---------------+
| 2      | 3000.00       |
+--------+---------------+
1 row in set (0.00 sec)
```

50. Display the names of employees whose salary is the maximum given by their department.

```
mysql> select E1.Name
      -> from Employee_57 E1
      -> where Basic = (
      -> select max(Basic)
      -> from Employee_57 as E2 where E1.DeptID = E2.DeptID);
+----------+
| Name     |
+----------+
| Arun     |
| Mridula  |
| Kiran    |
| Mary     |
+----------+
4 rows in set (0.00 sec)
```

51. Display the names of the employees, if their salary is greater than the salary of some other Employees.

```
mysql>select distinct E1.Name from Employee_57 E1,Employee_57 E2 where
E1.Basic>E2.Basic;

+-------+
| Name  |
+-------+
| Arun  |
| Ruby  |
|Mridula|
| Amal  |
| Kiran |
| Mary  |
| Ram   |
+-------+
4 rows in set (0.00 sec)
```

52. Display the names of the employees, if their salary is greater than the salary of some other employees or less than the salary of some other employees.

```
mysql> select Name from Employee_57 E where Basic != (select Basic from
Employee_57 where Basic = E.Basic);
+--------+
| Name   |
+--------+
| Arun   |
| Ram    |
| Ruby   |
| Mridula|
| Menon  |
| Tim    |
| Kiran  |
| Amal   |
| Mary   |
+--------+
9 rows in set (0.00 sec)
```

53. Add a column city for the employee table.

```
mysql> alter table Employee_57 add column City varchar(15);
Query OK, 0 rows affected (0.03 sec)
```

54. Add a column city for the department.

```
mysql> alter table Department_57 add column City varchar(15);
Query OK, 0 rows affected (0.02 sec)
```

55. Find the names of employees who are from the same city as their company.

```
mysql> select E.Name from Employee_57 E,Department_57 D where
E.DeptId=D.DeptID and E.City=D.City;
+-------+
| Name  |
+-------+
| Tim   |
| Kiran |
+-------+
2 rows in set (0.00 sec)
```

56. Display the names of the departments giving the smallest total salary.

```
mysql> select DName from Department_57 where DeptID in( select DeptID from
Employee_57 group by DeptID order by sum(Basic) limit 1);
+----------+
| DName    |
+----------+
| CODING   |
+----------+
1 row in set (0.00 sec)
```

57. Display the names of employees who joined during 1990's.

```
mysql> select Name from Employee_57 where Join_Date between '1990-01-01'
and '1999-12-31';
+-----------+
| Name      |
+-----------+
| Ram       |
| Arun      |
| Ruby      |
| Kiran     |
+-----------+
4 rows in set (0.00 sec)
```

58. Display the names of employees joined during the month of August.

```
mysql> select Name from Employee_57 where month(Join_Date) = 8;
+----------+
| Name     |
+----------+
| Menon    |
| Tim      |
+----------+
2 rows in set (0.00 sec)
```

59. Display the details of departments not having any employees (take the help of exists clause to do this)

```
mysql> select * from Department_57 D where not exists ( select * from
Employee_57 E where E.DeptID = D.DeptID);
+--------+------------+-------------+
| DeptID | DName      | City        |
+--------+------------+-------------+
| 4      | TESTING    | NULL        |
+--------+------------+-------------+
1 row in set (0.00 sec)
```

60. Display the details of departments having more than 2 employees.

```
mysql> select * from Department_57 where DeptID in(select DeptID from
Employee_57 group by DeptID having count(ID)>2);
+--------+------------+-------------+
| DeptID | DName      | City        |
+--------+------------+-------------+
| 2      | CODING     | NULL        |
+--------+------------+-------------+
1 row in set (0.00 sec)
```

61.Display the details of employees who are getting salaries more than 5000.

```
mysql> select * from Employee_57 where Basic > 5000;
+-----+--------+---------+-------------+---------+--------+------------+--------------+
| ID  | DeptID | Name    | Designation | Basic   | Gender | Manager_ID | Join_Date    |
+-----+--------+---------+-------------+---------+--------+------------+--------------+
| 102 | 2      | Arun    | Analyst     | 6000.00 | M      | 101        | 2022-02-15   |
| 123 | 2      | Mridula | Analyst     | 6000.00 | F      | 102        | 2022-05-25   |
+-----+--------+---------+-------------+---------+--------+------------+--------------+
2 rows in set (0.09 sec)
```

62. Insert the details of some employees who are not assigned with a department.(did is null);

```
mysql> insert into Employee_57 (ID, DeptID, Name, Designation, Basic,
Gender, Manager_ID, Joining_Date) values (201, NULL, 'John', 'Typist',
2500.00, 'M', NULL, '2024-09-10'),(202, NULL, 'Sarah', 'Clerk', 2700.00,
'F', NULL, '2024-09-15');

Query OK, 2 rows affected (0.01 sec)
```

63. Display the names of employees and their department ids. If an employee is not assigned with a department, display his name with department id as “null”.

```
mysql> select Name, ifnull(DeptID, 'null') as DeptID from Employee_57;
+-----------+--------+
| Name      | DeptID |
+-----------+--------+
| Ram       | 1      |
| Arun      | 2      |
| Ruby      | 1      |
| Mridula   | 2      |
| Menon     | 4      |
| Tim       | 4      |
| Kiran     | 2      |
| Amal      | 2      |
| Mary      | 3      |
| John      | null   |
| Sarah     | null   |
+----------+---------+
11 rows in set (0.00 sec)
```

64. Display the names of employees and their department ids. If an employee is not assigned with a department, display his name with department id as 0.

```
mysql> select Name, ifnull(DeptID, 0) as DeptID from Employee_57;
```

```
+-----------+--------+
| Name      | DeptID |
+-----------+--------+
| Ram       | 1      |
| Arun      | 2      |
| Ruby      | 1      |
| Mridula   | 2      |
| Menon     | 4      |
| Tim       | 4      |
| Kiran     | 2      |
| Amal      | 2      |
| Mary      | 3      |
| John      | 0      |
| Sarah     | 0      |
+----------+---------+
11 rows in set (0.00 sec)
```

Result:

Queries are executed successfully and output obtained.

CYCLE- 2

BOOK TABLE

The requirement: A library wants to maintain the record of books, members, book issue, book return, and fines collected for late returns, in a database. The database can be loaded with book information. Students can register with the library to be a member. Books can be issued to students with a valid library membership. A student can keep an issued book with him/her for a maximum period of two weeks from the date of issue, beyond which a fine will be charged. Fine is calculated based on the delay in days of return. For 0-7 days: Rs 10, For 7 – 30 days: Rs 100, and for days above 30 days: Rs 10 will be charged per day.

Sample Database Design
BOOK (Book_Id, Title, Language_Id, MRP, Publisher_Id, Published_Date, Volume, Status)

Language_Id, Publisher_Id are FK (Foreign Key)

AUTHOR(Author_Id, Name, Email, Phone_Number, Status)

BOOK_AUTHOR(Book_Id, Author_Id) // many-to-many relationship, both columns are PKFK (Primary Key and Foreign Key)

PUBLISHER(Publisher_id, Name, Address)

MEMBER(Member_Id, Name, Branch_Code, Roll_Number, Phone_Number, Email_Id, Date_of_Join, Status)

BOOK_ISSUE(Issue_Id, Date_Of_Issue, Book_Id, Member_Id, Expected_Date_Of_Return, Status) Book_Id and Member_Id are FKs

BOOK_RETURN(Issue_Id, Actual_Date_Of_Return, LateDays, LateFee) // Issue_Id is PK and FK

LANGUAGE(Language_id, Name) //Static Table for storing permanent data

LATE_FEE_RULE(FromDays, ToDays, Amount) // Composite Key

EXERCISES

1. Create a normalised database design with proper tables, columns, column types and constraints.

BOOK Table:
Book_Id (Primary Key, int)
Title (varchar)
Language_Id (Foreign Key, int)
MRP (decimal)
Publisher_Id (Foreign Key, int)
Published_Date (date)
Volume (int)
Status (varchar)

AUTHOR Table:
Author_Id (Primary Key, int)
Name (varchar)
Email (varchar)
Phone_Number (varchar)
Status (varchar)
BOOK_AUTHOR Table:
Book_Id (Foreign Key, int)
Author_Id (Foreign Key, int)
(Composite Primary Key)

PUBLISHER Table:
Publisher_Id (Primary Key, int)
Name (varchar)
Address (varchar)

MEMBER Table:
Member_Id (Primary Key, int)
Name (varchar)
Branch_Code (varchar)
Roll_Number (varchar)
Phone_Number (varchar)
Email_Id (varchar)
Date_of_Join (date)
Status (varchar)

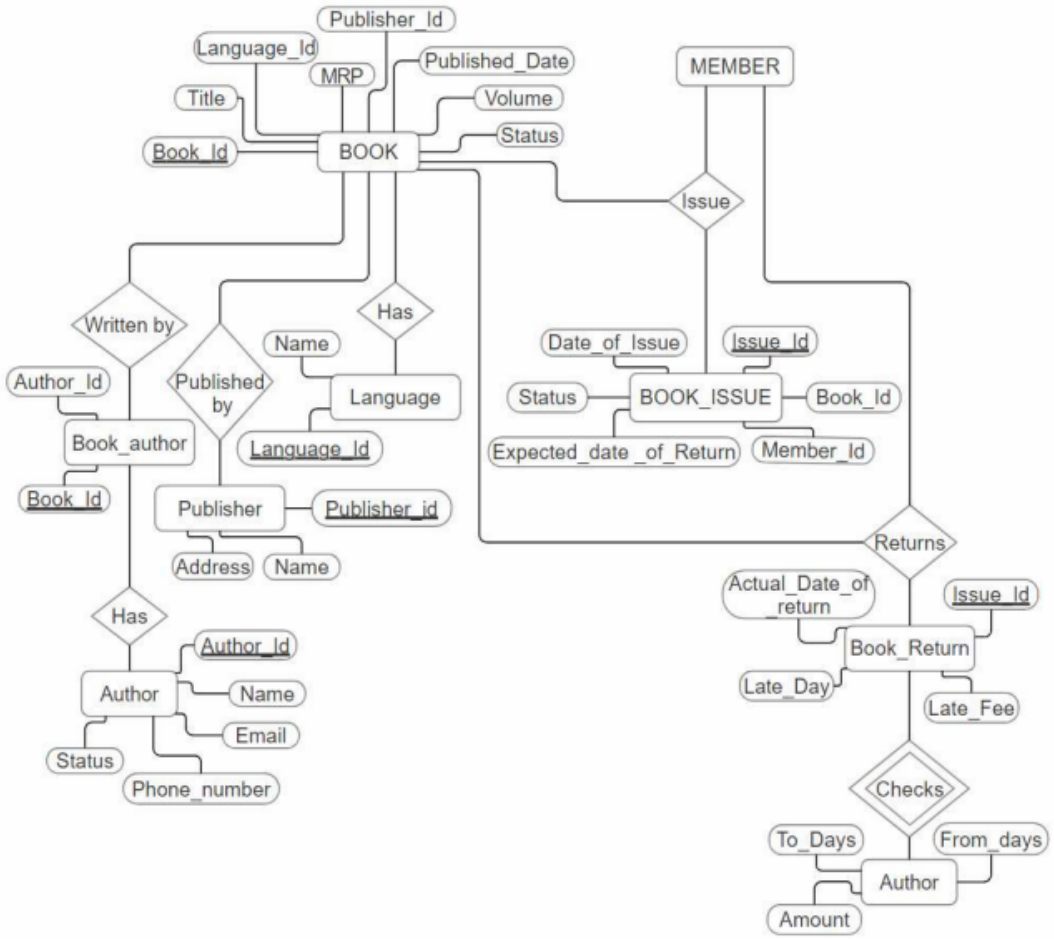
BOOK_ISSUE Table:
Issue_Id (Primary Key, int)
Date_Of_Issue (date)
Book_Id (Foreign Key, int)
Member_Id (Foreign Key, int)
Expected_Date_Of_Return (date)
Status (varchar)

BOOK_RETURN Table:
Issue_Id (Primary Key and Foreign Key, int)
Actual_Date_Of_Return (date)
LateDays (int)
LateFee (decimal)

LANGUAGE Table:
Language_Id (Primary Key, int)
Name (varchar)

LATE_FEE_RULE Table:
FromDays (Part of Composite Primary Key, int)
ToDays (Part of Composite Primary Key, int)
Amount (decimal)

2. Create an ER diagram for the above database design.



3. Create an ER diagram for this specification and then convert the ER diagram into relational model

BOOK

Book_ID	Title	Language_ID	MRP	Publisher_ID	Published_Date	Volume	Status

LANGUAGE

Language_ID	Name

BOOK_AUTHOR

Book_ID	Author_ID

BOOK_ISSUE

Issue_ID	Date_of_Issue	Book_ID	Member_ID	Expected_Date_of_Return	Status

PUBLISHER

Publisher_ID	Name	Address

AUTHOR

Author_ID	Name	Email	Phone_Number	Status

MEMBER

Member_ID	Name	Branch_Code	Roll_No	Phone_Number	Email	Date_of_Join	Status

BOOK_RETURN

Issue_ID	Actual_Date_of_Return	LateDays	LateFee

LATE_FEE_RULE

FromDays	ToDays	Amount

4. Write SQL commands to
a. Create DDL statements and create the tables and constraints (from the design)

```
mysql> create table LANGUAGE_57(Language_Id int primary key,Name
varchar(255) not null,Address varchar(255) not null);
Query OK, 0 rows affected (0.56 sec)

mysql> create table PUBLISHER_57(Publisher_Id int primary key, Name
varchar(255) not null,Address varchar(255) not null);
Query OK, 0 rows affected (0.51 sec)

mysql> create table AUTHOR_57(Author_Id int primary key, Name varchar(255)
not null,Email varchar(255),Phone_Number varchar(20),Status varchar(50) not
null);
Query OK, 0 rows affected (0.98 sec)

mysql> create table BOOK_57(Book_Id int primary key,Title varchar(255) not
null,Language_Id int,MRP decimal(10,2),Publisher_Id int,Published_Date
date,Volume int,Status varchar(50) not null, FOREIGN KEY(Language_Id)
REFERENCES LANGUAGE_57(Language_Id),FOREIGN KEY(Publisher_Id)REFERENCES
PUBLISHER_57(Publisher_Id));
Query OK, 0 rows affected (1.83 sec)

mysql> create table BOOK_AUTHOR_57(Book_Id int,Author_Id int,PRIMARY
KEY(Book_Id,Author_Id),FOREIGN KEY(Book_Id) REFERENCES
BOOK_57(Book_Id),FOREIGN KEY(Author_Id) REFERENCES AUTHOR_57(Author_Id));
Query OK, 0 rows affected (0.71 sec)

mysql> create table MEMBER_57(Member_Id int primary key,Name varchar(255)
not null,Branch_Code varchar(20) not null,Roll_Number varchar(20) not
null,Phone_Number varchar(20),Email_Id varchar(255),Date_of_Join
date,Status varchar(50) not null);
Query OK, 0 rows affected (0.60 sec)

mysql> create table BOOK_ISSUE_57(Issue_Id int primary key,Date_of_Issue
date,Book_Id int,Member_Id int,Expected_Date_of_Return date,Status
varchar(50) not null,FOREIGN KEY(Book_Id) REFERENCES
BOOK_57(Book_Id),FOREIGN KEY(Member_Id)REFERENCES MEMBER_57(Member_Id));
Query OK, 0 rows affected (0.71 sec)

mysql> create table BOOK_RETURN_57(Issue_Id int primary
key,Actual_Date_of_Return date,LateDays int,LateFee decimal(10,2),FOREIGN
KEY(Issue_Id) REFERENCES BOOK_ISSUE_57(Issue_Id));
Query OK, 0 rows affected (0.65 sec)

mysql> create table LATE_FEE_RULE_57(FromDays int,ToDays int,Amount
decimal(10,2),primary key(FromDays,ToDays));
Query OK, 0 rows affected (0.58 sec)
```

```
mysql> show tables;
+---------------------+
| Tables_in_Vidya_2024|
+---------------------+
| AUTHOR_57           |
| BOOK_57             |
| BOOK_AUTHOR_57      |
| BOOK_ISSUE_57       |
| BOOK_RETURN_57      |
| Department_57       |
| Employee_57         |
| LANGUAGE_57         |
| LATE_FEE_RULE_57    |
| MEMBER_57           |
| PUBLISHER_57        |
+---------------------+
11 rows in set (0.00 sec)

mysql> INSERT INTO LANGUAGE_57(Language_Id, Name) VALUES  (1, 'English'),
(2, 'French'), (3, 'Tamil'), (4, 'Hindi'),(5, 'Malayalam');
Query OK, 5 rows affected (0.08 sec)
Records: 5  Duplicates: 0  Warnings: 0


mysql> select * from LANGUAGE_57;
+-------------+-----------+
| Language_Id | Name      |
+-------------+-----------+
|           1 | English   |
|           2 | French    |
|           3 | Tamil     |
|           4 | Hindi     |
|           5 | Malayalam |
+-------------+-----------+
5 rows in set (0.00 sec)

mysql> INSERT INTO PUBLISHER_57(Publisher_Id, Name, Address) VALUES  (1,
'Shueisha', 'Chiyoda, Tokyo, Japan'), (2, 'Bantam Spectra (US), Voyager
Books (UK)', '195 Broadway, New York City, New York, U.S.'), (3, 'Current
Books', 'Thrissur, Kerala, India'), (4, 'Penguin Books (English), Green
Books (Malayalam)', 'Trivandrum, Kerala, India'), (5, 'DC Books',
'Kottayam, Kerala, India'), (6, 'Vanathi Pathippagam', 'Chennai, Tamil
Nadu, India'), (7, 'Rupa Publications Pvt. Ltd', 'Bangalore,Karnataka,
India');
Query OK, 7 rows affected (0.08 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> select * from PUBLISHER_57;
```

```
+-------------+-----------------------------------------------+-----------------------------------------+
| Publisher_Id | Name                                          | Address                                 |
+-------------+-----------------------------------------------+-----------------------------------------+
|           1 | Shueisha                                      | Chiyoda, Tokyo, Japan                   |
|           2 | Bantam Spectra (US), Voyager Books (UK)       | 195 Broadway, New York City, New York, U.S. |
|           3 | Current Books                                 | Thrissur, Kerala, India                 |
|           4 | Penguin Books (English), Green Books (Malayalam) | Trivandrum, Kerala, India               |
|           5 | DC Books                                      | Kottayam, Kerala, India                 |
|           6 | Vanathi Pathippagam                           | Chennai, Tamil Nadu, India              |
|           7 | Rupa Publications Pvt. Ltd                    | Bangalore, Karnataka, India             |
+-------------+-----------------------------------------------+-----------------------------------------+
7 rows in set (0.00 sec)

mysql> INSERT INTO AUTHOR_57(Author_Id, Name, Email, Phone_Number, Status)
    -> VALUES
    -> (1, 'Sir Arthur Conan Doyle', 'arthur.doyle@example.com',
'1234567890', 'Active'),
    -> (2, 'George R. R. Martin', 'george.martin@example.com',
'0987654321', 'Active'),
    -> (3, 'M. T. Vasudevan Nair', 'mt.vasudevan@example.com',
'1122334455', 'Active'),
    -> (4, 'Benyamin', 'benyamin@example.com', '2233445566', 'Active'),
    -> (5, 'Akhil P Dharmajan', 'akhil.pi@example.com', '3344556677',
'Active'),
    -> (6, 'Kalki Krishnamurthy', 'kalki.k@example.com', '4455667788',
'Inactive'),
    -> (7, 'Chetan Bhagat', 'chetan.bhagat@example.com', '5566778899',
'Active');
Query OK, 7 rows affected (0.11 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> select * from AUTHOR_57;
+-----------+-----------------------+---------------------------+--------------+----------+
| Author_Id | Name                  | Email                     | Phone_Number | Status   |
+-----------+-----------------------+---------------------------+--------------+----------+
|         1 | Sir Arthur Conan Doyle | arthur.doyle@example.com  | 1234567890   | Active   |
|         2 | George R. R. Martin    | george.martin@example.com | 0987654321   | Active   |
|         3 | M. T. Vasudevan Nair   | mt.vasudevan@example.com  | 1122334455   | Active   |
|         4 | Benyamin               | benyamin@example.com      | 2233445566   | Active   |
|         5 | Akhil Pi Dharmajan     | akhil.pi@example.com      | 3344556677   | Active   |
|         6 | Kalki Krishnamurthy    | kalki.k@example.com       | 4455667788   | Inactive |
|         7 | Chetan Bhagat          | chetan.bhagat@example.com | 5566778899   | Active   |
+-----------+-----------------------+---------------------------+--------------+----------+
7 rows in set (0.00 sec)

mysql> INSERT INTO BOOK_57 (Book_Id, Title, Language_Id, MRP, Publisher_Id,
Published_Date, Volume, Status) VALUES
    -> (1, 'Sherlock Holmes', 1, 500.00, 1, '1892-01-01', 1, 'Available'),
    -> (2, 'GOT (VOL 1&2)', 1, 1000.00, 2, '1996-08-06', 2, 'Available'),
    -> (3, 'Randamoozham', 1, 300.00, 3, '1984-01-01', 1, 'Available'),
    -> (4, 'Aadujeevitham', 1, 350.00, 4, '2008-01-01', 1, 'Available'),
    -> (5, 'RAM C/O ANANDHI', 1, 250.00, 5, '2019-01-01', 1, 'Available'),
    -> (6, 'Ponniyin Selvan (VOL 1&2)', 1, 600.00, 6, '1950-01-01', 2,
'Available'),
    -> (7, '2 States: The Story of My Marriage', 1, 200.00, 7, '2009-10-
01', 1, 'Available'),
    -> (8, 'Half Girlfriend', 1, 150.00, 7, '2014-10-01', 1, 'Available');
```

```
Query OK, 8 rows affected (0.12 sec)
Records: 8  Duplicates: 0  Warnings: 0
mysql> select * from BOOK_57;
```

Book_Id	Title	Language_Id	MRP	Publisher_Id	Published_Date	Volume	Status
1	Sherlock Holmes	1	500.00	1	1892-01-01	1	Available
2	GOT (VOL 1&2)	1	1000.00	2	1996-08-06	2	Available
3	Randamoozham	1	300.00	3	1984-01-01	1	Available
4	Aadujeevitham	1	350.00	4	2008-01-01	1	Available
5	RAM C/O ANANDHI	1	250.00	5	2019-01-01	1	Available
6	Ponniyin Selvan (VOL 1&2)	1	600.00	6	1950-01-01	2	Available
7	2 States: The Story of My Marriage	1	200.00	7	2009-10-01	1	Available
8	Half Girlfriend	1	150.00	7	2014-10-01	1	Available

8 rows in set (0.01 sec)

```
mysql> insert into BOOK_AUTHOR_57(Book_Id,Author_Id)
values(1,1),(2,2),(3,3),(4,4),(5,5),(6,6),(7,7),(8,7);
Query OK, 8 rows affected (0.08 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> select * from BOOK_AUTHOR_57;
```

Book_Id	Author_Id
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	7

8 rows in set (0.00 sec)

```
mysql> INSERT INTO MEMBER_57(Member_Id, Name, Branch_Code, Roll_Number,
Phone_Number, Email_Id, Date_of_Join, Status) VALUES
(3,'Arun','ME','13426','9847234563','arun@example.com','2023-09-
05','Active'),(4,'Avani','EEE','13429','8606543432','avani@example.com','20
24-01-06','Active');
Query OK, 2 rows affected (0.73 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from MEMBER_57;
```

Member_Id	Name	Branch_Code	Roll_Number	Phone_Number	Email_Id	Date_of_Join	Status
1	Raju	CS	12345	9876543210	raju@example.com	2023-01-01	Active
2	Kavya	IT	67890	8765432109	kavya@example.com	2023-02-01	Active
3	Arun	ME	13426	9847234563	arun@example.com	2023-09-05	Active
4	Avani	EEE	13429	8606543432	avani@example.com	2024-01-06	Active

4 rows in set (0.00 sec)

```
mysql> INSERT INTO BOOK_ISSUE_57(Issue_Id, Date_Of_Issue, Book_Id,
Member_Id, Expected_Date_Of_Return, Status) VALUES  (1, '2023-09-01', 1, 1,
'2023-09-15', 'Issued'),(2, '2023-09-05', 2, 2, '2023-09-19', 'Issued'),
(3, '2023-03-21', 5, 3, '2023-04-06', 'Issued');
Query OK, 3 rows affected (0.20 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from BOOK_ISSUE_57;
+----------+--------------+---------+-----------+------------------------+--------+
| Issue_Id | Date_of_Issue | Book_Id | Member_Id | Expected_Date_of_Return | Status |
+----------+--------------+---------+-----------+------------------------+--------+
|        1 | 2023-09-01    |       1 |         1 | 2023-09-15             | Issued |
|        2 | 2023-09-05    |       2 |         2 | 2023-09-19             | Issued |
|        3 | 2023-03-21    |       5 |         3 | 2023-04-06             | Issued |
+----------+--------------+---------+-----------+------------------------+--------+
3 rows in set (0.00 sec)

mysql> INSERT INTO BOOK_RETURN_57 (Issue_Id, Actual_Date_Of_Return,
LateDays, LateFee) VALUES  (1, '2023-09-17', 2, 10.00), (2, '2023-10-01',
12, 100.00);
Query OK, 2 rows affected (0.20 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from BOOK_RETURN_57;
+----------+-----------------------+----------+---------+
| Issue_Id | Actual_Date_of_Return | LateDays | LateFee |
+----------+-----------------------+----------+---------+
|        1 | 2023-09-17            |        2 |   10.00 |
|        2 | 2023-10-01            |       12 |  100.00 |
+----------+-----------------------+----------+---------+
2 rows in set (0.00 sec)

mysql> INSERT INTO LATE_FEE_RULE_57(FromDays, ToDays, Amount) VALUES  (0,
7, 10.00), (8, 30, 100.00), (31, 999, 10.00);
Query OK, 3 rows affected (0.10 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from LATE_FEE_RULE_57;
+----------+--------+--------+
| FromDays | ToDays | Amount |
+----------+--------+--------+
|        0 |      7 |  10.00 |
|        8 |     30 | 100.00 |
|       31 |    999 |  10.00 |
+----------+--------+--------+
3 rows in set (0.00 sec)
```

b. Create and execute DROP TABLE commands in tables with and without FOREIGN KEY constraints.

```
mysql> DROP TABLE IF EXISTS BOOK_RETURN_57;
Query OK, 0 rows affected (0.05 sec)
mysql> DROP TABLE IF EXISTS BOOK_ISSUE_57;
Query OK, 0 rows affected (0.03 sec)
mysql> DROP TABLE IF EXISTS BOOK_AUTHOR_57;
Query OK, 0 rows affected (0.03 sec)
mysql> DROP TABLE IF EXISTS AUTHOR_57;
Query OK, 0 rows affected (0.03 sec)
mysql> DROP TABLE IF EXISTS BOOK_57;
Query OK, 0 rows affected (0.05 sec)
mysql> DROP TABLE IF EXISTS LANGUAGE_57;
Query OK, 0 rows affected (0.02 sec)
mysql> DROP TABLE IF EXISTS PUBLISHER_57;
Query OK, 0 rows affected (0.02 sec)
mysql> DROP TABLE IF EXISTS MEMBER_57;
Query OK, 0 rows affected (0.03 sec)
mysql> DROP TABLE IF EXISTS LATE_FEE_RULE_57;
Query OK, 0 rows affected (0.02 sec)
```

c. Create and execute ALTER TABLE commands in tables with data and without data.

```
mysql> ALTER TABLE MEMBER_57 ADD COLUMN Remarks VARCHAR(255);
Query OK, 0 rows affected (0.04 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql>
mysql> ALTER TABLE MEMBER_57 ADD COLUMN Remarks VARCHAR(255);
Query OK, 0 rows affected (0.07 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

4. Based on the above relational database design, Write SQL Query to retrieve the following information.
a. Get the number of books written by a given author

```
mysql> SELECT count(*) from BOOK_AUTHOR_57 where Author_Id in(select
Author_Id from AUTHOR_57 where Name='Benyamin');
+-----------------+
| Number_of_Books |
+-----------------+
|               1 |
+-----------------+
1 row in set (0.00 sec)
```

b. Get the list of publishers and the number of books published by each publisher

```
mysql> SELECT P.Name AS Publisher,COUNT(B.Book_Id)AS Number_of_books FROM
BOOK_57 B,PUBLISHER_57 P WHERE B.Publisher_Id = P.Publisher_Id;

+-------------------------------------------------+-----------------+
| Publisher                                       | Number_of_Books |
+-------------------------------------------------+-----------------+
| Shueisha                                        |               1 |
| Bantam Spectra (US), Voyager Books (UK)         |               1 |
| Current Books                                   |               1 |
| Penguin Books (English), Green Books (Malayalam) |               1 |
| DC Books                                        |               1 |
| Vanathi Pathippagam                             |               1 |
| Rupa Publications Pvt. Ltd                      |               2 |
+-------------------------------------------------+-----------------+
7 rows in set (0.00 sec)
```

c. Get the list of books that are issued but not returned

```
mysql> SELECT Title FROM BOOK_57 WHERE Book_Id IN (SELECT Book_Id FROM
BOOK_ISSUE_57 WHERE Status = 'Issued');

+-----------------+
| Title           |
+-----------------+
| Sherlock Holmes |
| GOT (VOL 1&2)   |
| RAM C/O ANANDHI |
+-----------------+
3 rows in set (0.02 sec)
```

d. Get the list of students who read only ‘Malayalam’ books.

```
mysql> SELECT Name FROM MEMBER_57 WHERE Member_Id NOT IN ( SELECT Member_Id
FROM BOOK_ISSUE_57 WHERE Book_Id NOT IN ( SELECT Book_Id FROM BOOK_57 WHERE
Language_Id = (SELECT Language_Id FROM LANGUAGE_57 WHERE Name =
'Malayalam')));

+-------+
| Name  |
+-------+
| Avani |
+-------+
1 row in set (0.00 sec)
```

e. Get the total fine collected for the current month and current quarter.

```
mysql>  SELECT SUM(LateFee) AS TotalFine FROM BOOK_RETURN_57 WHERE
```

```
MONTH(Actual_Date_of_Return) = MONTH(CURDATE()) AND
YEAR(Actual_Date_of_Return) = YEAR(CURDATE());
+-----------+
| TotalFine |
+-----------+
|     100.00|
+-----------+
1 row in set (0.01 sec)

mysql>  SELECT SUM(LateFee) AS TotalFine FROM BOOK_RETURN_57 WHERE
QUARTER(Actual_Date_of_Return) = QUARTER(CURDATE()) AND
YEAR(Actual_Date_of_Return) = YEAR(CURDATE());
+-----------+
| TotalFine |
+-----------+
|    310.00 |
+-----------+
1 row in set (0.01 sec)
```

f. Get the list of students who have overdue (not returned the books even on due date)

```
mysql> SELECT Name FROM MEMBER_57 WHERE Member_Id IN(SELECT Member_Id FROM
BOOK_ISSUE_57 WHERE Expected_Date_of_Return < CURDATE() AND Status =
'Issued');
+-------+
| Name  |
+-------+
| Raju  |
| Kavya |
| Arun  |
+-------+
3 rows in set (0.00 sec)
```

g. Calculate the fine (as of today) to be collected from each overdue book.

```
mysql> SELECT Issue_Id, DATEDIFF(CURDATE(), Expected_Date_of_Return) *
(SELECT Amount FROM LATE_FEE_RULE_57 WHERE FromDays <= DATEDIFF(CURDATE(),
Expected_Date_of_Return) AND ToDays >= DATEDIFF(CURDATE(),
Expected_Date_of_Return)) AS Fine FROM BOOK_ISSUE_57 WHERE
Expected_Date_of_Return < CURDATE() AND Status = 'Issued';
+----------+---------+
| Issue_Id | Fine    |
+----------+---------+
|        1 | 3770.00 |
|        2 | 3730.00 |
|        3 | 5390.00 |
+----------+---------+
3 rows in set (0.00 sec)
```

h. Members who joined after Jan 1 2021 but has not taken any books

```
mysql> SELECT Name FROM MEMBER_57 WHERE Date_of_Join > '2021-01-01' AND
Member_Id NOT IN (SELECT Member_Id FROM BOOK_ISSUE_57);
+-------+
| Name  |
+-------+
| Avani |
+-------+
1 row in set (0.00 sec)
```

Result:

Queries are executed successfully and output obtained.

CYCLE-3

PL/SQL PRACTICE QUESTIONS

1. Write a PL/SQL block to read two numbers and find the greatest among them.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE MAXIMUM (A INT,B INT)
    -> BEGIN
    -> IF (A>B) THEN
    -> SELECT A;
    -> ELSE
    -> SELECT B;
    -> END IF;
    -> END;//
Query OK, 0 rows affected (0.13 sec)

mysql> CALL MAXIMUM (9,4);//
+------+
| A    |
+------+
|    9 |
+------+
1 row in set (0.02 sec)
```

2. Write a PL/SQL block to read three numbers and find the greatest among them.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE MAX_OF_THREE (A INT,B INT, C INT)
    -> BEGIN
    -> IF (A>B) AND (A>C) THEN
    -> SELECT A;
    -> ELSEIF (B>A) AND (B>C) THEN
    -> SELECT B;
    -> ELSE
    -> SELECT C;
    -> END IF;
    -> END;//
Query OK, 0 rows affected (0.14 sec)

mysql> CALL MAX_OF_THREE (12,24,18);//
+------+
|   B  |
+------+
|   24 |
+------+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
```

3. Write a PL/SQL block to read two numbers and print all the numbers between them.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE SEQUENCE_PRINT(A INT,B INT)
    -> BEGIN
    -> DECLARE I INT;
    -> SET I=A;
    -> WHILE I<B DO
    -> SELECT I+1 AS NUMBER;
    -> SET I=I+1;
    -> END WHILE;
    -> END;//
Query OK, 0 rows affected (0.16 sec)

mysql> CALL SEQUENCE_PRINT(4,11);//
+--------+
| NUMBER |
+--------+
|      5 |
+--------+
1 row in set (0.01 sec)
+--------+
| NUMBER |
+--------+
|      6 |
+--------+
1 row in set (0.01 sec)
+--------+
| NUMBER |
+--------+
|      7 |
+--------+
1 row in set (0.01 sec)
+--------+
| NUMBER |
+--------+
|      8 |
+--------+
1 row in set (0.01 sec)
+--------+
| NUMBER |
+--------+
|      9 |
+--------+
1 row in set (0.01 sec)
+--------+
| NUMBER |
+--------+
|      10|
+--------+
1 row in set (0.01 sec)
Query OK, 0 rows affected (0.01 sec)
```

4. Write a PL/SQL block to read N and find the sum of the series 1+2+3 +... N.

```
mysql> CREATE PROCEDURE SUM_OF_SERIES(N INT)
mysql> BEGIN
mysql> DECLARE I INT;
mysql> DECLARE SUM INT;
mysql> SET SUM=0;
mysql> SET I=1;
mysql> WHILE I<=N DO SET SUM=SUM+I;
mysql> SET I=I+1;
mysql> END WHILE;
mysql> SELECT SUM AS SUM;
mysql> END;//
Query OK, 0 rows affected (0.12 sec)
mysql> CALL SUM_OF_SERIES (10);//
+------+
| SUM  |
+------+
|   55 |
+------+
1 row in set (0.01 sec)
Query OK, 0 rows affected (0.01 sec)
```

5.Write a PL/SQL block to read the marks and display the grade.

```
mysql> CREATE PROCEDURE GRADE (MARK INT)
    -> BEGIN
    -> DECLARE GRADE VARCHAR(2);
    -> IF MARK>=90 THEN
    -> SET GRADE='A';
    -> ELSEIF MARK>80 THEN
    -> SET GRADE='B';
    -> ELSEIF MARK>70 THEN
    -> SET GRADE='C';
    -> ELSEIF MARK>60 THEN
    -> SET GRADE='D';
    -> ELSE
    -> SET GRADE='F';
    -> END IF;
    -> SELECT GRADE;
    -> END;//
Query OK, 0 rows affected (0.20 sec)
mysql> CALL GRADE (88);//
+-------+
| GRADE |
+-------+
|  B    |
+-------+
```

```
1 row in set (0.00 sec)

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

6. Write a PL/SQL block to read a number and invert the given number.

```
mysql> CREATE PROCEDURE INVERT_NUMBER (NUM INT)
    -> BEGIN
    ->DECLARE REVNUM INT DEFAULT 0;
    ->DECLARE REM INT;
    ->WHILE NUM!= 0 DO
    ->SET REM=NUM%10;
    ->SET REVNUM=REVNUM*10+REM;
    ->SET NUM=NUM/10;
    ->END WHILE;
    ->SELECT REVNUM AS 'Reversed Number';
    -> END;//
Query OK, 0 rows affected (0.18 sec)

mysql> CALL INVERT_NUMBER (1124);//
+-----------------+
| Reversed Number |
+-----------------+
|            4211 |
+-----------------+
1 row in set (0.00 sec)

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

Create a Table: EMPLOYEE: (ID, NAME, SALARY, DEPNO, BDATE)
Then do the following questions

7. Create a procedure to display Welcome to PL/SQL

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE WELCOME_MSG ()
    -> BEGIN
    -> SELECT 'Welcome to PL/SQL' AS Message;
    -> END;//
Query OK, 0 rows affected (0.19 sec)

mysql> CALL WELCOME_MSG ();//
+-------------------+
| Message           |
+-------------------+
| Welcome to PL/SQL |
+-------------------+
1 row in set (0.00 sec)

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

8. Write a PL/SQL block to read the ID of an employee and display his salary.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE SALARY (EmpID INT)
    -> BEGIN
    -> SELECT Basic AS Salary FROM Employee_57 WHERE ID=EmpID;
    -> END;//
Query OK, 0 rows affected (0.16 sec)

mysql> CALL SALARY (127);//
+---------+
| Salary  |
+---------+
| 4000.00 |
+---------+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
```

9. Write a PL/SQL block to read the ID of an employee and display his name and birthdate.

```
mysql> CREATE PROCEDURE BirthDate (Emp_ID INT)
    -> BEGIN
    -> SELECT Name, BDate FROM Employee_57 WHERE ID=Emp_ID;
    -> END;//
Query OK, 0 rows affected (0.18 sec)

mysql> CALL BirthDate (108);//
+-------+------------+
| Name  | BDate      |
+-------+------------+
| Vivek | 2004-02-13 |
+-------+------------+
1 row in set (0.01 sec)
Query OK, 0 rows affected (0.01 sec)
```

10. Write a PL/SQL block to read the ID of an employee and display his month of birth.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE BIRTH_MONTH(Emp_ID INT)
    -> BEGIN
    -> SELECT MONTH(BDate) AS Birth_Month FROM Employee_57 WHERE ID=Emp_ID;
    -> END;//
Query OK, 0 rows affected (0.14 sec)

mysql> CALL BIRTH_MONTH(132);//
```

```
+-------------+
| Birth_Month |
+-------------+
| March       |
+-------------+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
```

11. Write a PL/SQL block to read IDs of two employees and display the difference in salary between them.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE SALARY_DIFF (E_ID1 INT, E_ID2 INT)
    -> BEGIN
    -> SELECT (E1.Basic - E2.Basic) AS Salary_Difference FROM Employee_57
E1, Employee_57 E2 WHERE E1.ID=E_ID1 AND E2.ID=E_ID2;
    -> END;//
Query OK, 0 rows affected (0.13 sec)

mysql> CALL SALARY_DIFF (123,121);//
+-------------------+
| Salary_Difference |
+-------------------+
|           3990.00 |
+-------------------+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
```

12. Create a cursor to display the highest 10 salaries of the employee table.

```
mysql> DELIMITER
mysql> CREATE PROCEDURE TOP_TEN_SAL()
    -> BEGIN
    -> DECLARE done INT DEFAULT 0;
    -> DECLARE emp_salary DECIMAL(10,2);
    -> DECLARE cur CURSOR FOR
    -> SELECT Basic FROM Employee_57 ORDER BY Basic DESC LIMIT 10;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET done=1;
    -> OPEN cur;
    -> read_loop
    -> : LOOP
    -> FETCH cur INTO emp_salary;
    -> IF done THEN
    -> LEAVE read_loop;
    -> END IF;
    -> SELECT emp_salary;
    -> END LOOP;
    -> CLOSE cur;
    -> END;//
```

```
Query OK, 0 rows affected (0.13 sec)

mysql> CALL TOP_TEN_SAL ();//
+------------+
| emp_salary |
+------------+
|    6000.00 |
+------------+
1 row in set (0.01 sec)

+------------+
| emp_salary |
+------------+
|    6000.00 |
+------------+
1 row in set (0.01 sec)

+------------+
| emp_salary |
+------------+
|    6000.00 |
+------------+
1 row in set (0.01 sec)

+------------+
| emp_salary |
+------------+
|    4500.00 |
+------------+
1 row in set (0.01 sec)

+------------+
| emp_salary |
+------------+
|    4000.00 |
+------------+
1 row in set (0.01 sec)

+------------+
| emp_salary |
+------------+
|    2010.00 |
+------------+
1 row in set (0.01 sec)

+------------+
| emp_salary |
+------------+
|    2000.00 |
+------------+
```

```
1 row in set (0.01 sec)
+------------+
| emp_salary |
+------------+
|    2000.00 |
+------------+
1 row in set (0.01 sec)
+------------+
| emp_salary |
+------------+
|    2000.00 |
+------------+
1 row in set (0.01 sec)
+------------+
| emp_salary |
+------------+
|    1500.00 |
+------------+
1 row in set (0.01 sec)
Query OK, 0 rows affected (0.01 sec)
```

13. Create a procedure to accept the dno and display the id, name and salary of all the employees working in that department. Execute this procedure and show the result.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GET_DEPT (dept_id INT)
    -> BEGIN
    -> SELECT ID,Name,Basic
    -> FROM Employee_57
    -> WHERE DeptID=dept_id;
    -> END;//
Query OK, 0 rows affected (0.13 sec)

mysql> CALL GET_DEPT (3);//
+-----+--------+---------+
| ID  | Name   | Basic   |
+-----+--------+---------+
| 131 | Raju   | 2000.00 |
| 132 | Aleena | 1500.00 |
| 156 | Mary   | 4500.00 |
| 201 | Nithin | 6000.00 |
+-----+--------+---------+
4 rows in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
```

14. Create a function to accept the id of an employee and return his salary.

```
mysql> CREATE FUNCTION SAL_FUN(emp_id INT)
    -> RETURNS DECIMAL(10,2) DETERMINISTIC
    -> BEGIN
    -> DECLARE emp_salary DECIMAL(10,2);
    -> SELECT Basic INTO emp_salary FROM Employee_57 WHERE ID=emp_id;
    -> RETURN emp_salary;
    -> END;//
Query OK, 0 rows affected (0.14 sec)

mysql> SELECT SAL_FUN(156);//
+--------------+
| SAL_FUN(156) |
+--------------+
|      4500.00 |
+--------------+
1 row in set (0.00 sec)
```

15. Create a trigger to maintain an audit trail for the employee table. When insert, update or delete is performed on the employee table insert a row into emp_trail table with value specifying the operation and date of operation.

```
mysql> delimiter //
mysql> create trigger audit_trail_insert after insert on Employee_57 for
each row
    -> begin
    -> insert into EMP_TRAIL_57(operation,date) values
('insert',current_date());
    -> end; //
Query OK, 0 rows affected (0.16 sec)


mysql> delimiter //
mysql> create trigger audit_trail_update after update on Employee_57 for
each row
    -> begin
    -> insert into EMP_TRAIL_57 (operation,date) values
('update',current_date());
    -> end;//
Query OK, 0 rows affected (0.18 sec)


mysql> delimiter //
mysql> create trigger audit_trail_delete after delete on Employee_57 for
each row
    -> begin
    -> insert into EMP_TRAIL_57(operation,date) values
('delete',current_date());
    -> end; //
    Query OK, 0 rows affected (0.24 sec)
```

```
mysql> create table EMP_TRAIL_57(operation varchar(20) not null,date date);
    -> //
Query OK, 0 rows affected (0.57 sec)

mysql> INSERT INTO Employee_57 VALUES ('165', '1', 'Ansif', 'Clerk', 2000,
'M', NULL, NULL, NULL, NULL);//
Query OK, 1 row affected (0.08 sec)

mysql> UPDATE Employee_57 SET city='Kannur' WHERE ID='156';//
Query OK, 1 row affected (0.14 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> DELETE FROM Employee_57 WHERE ID='102';//
Query OK, 1 row affected (0.14 sec)

mysql> SELECT * FROM EMP_TRAIL_57;//
+-----------+---------------------+
| operation | date                |
+-----------+---------------------+
| INSERT    | 2024-10-24          |
| UPDATE    | 2024-10-24          |
| DELETE    | 2024-10-24          |
+-----------+---------------------+
3 rows in set (0.00 sec)
```

16. Create a trigger to maintain an audit trail for employee table for tracking salary modifications. When salary is updated, insert into emp_sal_trail table a row with values of employee id, name, salary before modification, salary after modification and date of modification

```
mysql> CREATE TABLE Emp_Sal_Trail_57 (
    -> TrailId INT AUTO_INCREMENT PRIMARY KEY,
    -> EmpId VARCHAR(5),
    -> Name VARCHAR(15),
    -> SalBefore DECIMAL(10, 2),
    -> SalAfter DECIMAL(10, 2),
    -> ModifyDate DATETIME DEFAULT CURRENT_TIMESTAMP);//
Query OK, 0 rows affected (0.65 sec)

mysql> CREATE TRIGGER EMP_SAL_UPDATE
    -> AFTER UPDATE ON Employee_57
    -> FOR EACH ROW
    -> BEGIN
    -> IF OLD.Basic <> NEW.Basic THEN
    -> INSERT INTO Emp_Sal_Trail_57 (EmpId, Name, SalBefore, SalAfter)
    -> VALUES (NEW.ID, NEW.Name, OLD.Basic, NEW.Basic);
    -> END IF;
    -> END;//
Query OK, 0 rows affected (0.18 sec)
```

```
mysql> UPDATE Employee_57 SET Basic=3000 WHERE ID='121';//
Query OK, 1 row affected (0.10 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Emp_Sal_Trail_57;//
+---------+-------+------+-----------+----------+---------------------+
| TrailId | EmpId | Name | SalBefore | SalAfter | ModifyDate          |
+---------+-------+------+-----------+----------+---------------------+
|       1 | 121   | Ruby |   2010.00 |  3000.00 | 2024-10-24 11:17:32 |
+---------+-------+------+-----------+----------+---------------------+
1 row in set (0.00 sec)
```

17. Create a trigger to prevent salary modification of an employee, if salary after modification is less than the salary before modification.

```
mysql> delimiter //
mysql> create trigger sal_dec before update on Employee_57 for each row
      -> begin
      -> if(new.Basic<old.Basic) then signal sqlstate '45000'
      -> set message_text='cannot decrease salary';
      -> end if;
      -> end;//
Query OK, 0 rows affected (0.20 sec)

mysql> update Employee_57 set Basic=3300 where ID='123';//
ERROR 1644 (45000): cannot decrease salary
```

18. Create a trigger to prevent salary modification of an employee on Monday.

```
mysql> CREATE TRIGGER sal_update_monday
      -> BEFORE UPDATE ON Employee_57
      -> FOR EACH ROW
      -> BEGIN
      -> IF DAYNAME(CURDATE()) ='MONDAY'  THEN
      -> SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Salary modification is
not allowed on Monday!';
      -> END IF;
      -> END;//
Query OK, 0 rows affected (0.17 sec)

mysql> update Employee_57 set Basic=6500 where ID='121';//
ERROR 1644 (45000): Salary modification is not allowed on Monday!
```

19.Assume a table Department with columns DeptNo and Total_Sal. Total_Sal maintains the total salary given by that department. Create triggers on employee table for maintaining Total_Sal in Department table

```
mysql> DELIMITER //
mysql> CREATE TRIGGER total_sal_on_update
      -> AFTER UPDATE ON Employee_57
      -> FOR EACH ROW
      -> BEGIN
      -> UPDATE Dept_57
      -> SET total_sal = total_sal + (NEW.basic - OLD.basic)
      -> WHERE DeptID = NEW.DeptID;
      -> END;//
Query OK, 0 rows affected (0.22 sec)

mysql> DELIMITER //
mysql> CREATE TRIGGER total_sal_on_insert
      -> AFTER INSERT ON Employee_57
      -> FOR EACH ROW
      -> BEGIN
      -> UPDATE Dept_57
      -> SET total_sal = total_sal + NEW.Basic
      -> WHERE DeptID = NEW.DeptID;
      -> END;//
Query OK, 0 rows affected (0.18 sec)

mysql> INSERT INTO Employee_57 (ID, DeptID, Name, Designation, Basic,
Gender, ManagerID, Join_date, City, BDate,total_sal) VALUES ('133', '4',
'Amal', 'Clerk', 4500, 'M', '156', '2022-01-23', 'Kannur', '2000-01-
15',0);//

Query OK, 1 row affected (0.15 sec)

mysql> UPDATE Employee_57 set Basic=6500 WHERE ID='121';//
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> select * from Dept_57;//

+--------+----------+-----------+
| DeptID | Name     | total_sal |
+--------+----------+-----------+
|      1 | Design   |  15000.00 |
|      2 | Coding   |  23000.00 |
|      3 | Testing  |  12000.00 |
|      4 | Research |  18000.00 |
|      5 | HR       |  14000.00 |
+--------+----------+-----------+
5 rows in set (0.01 sec)
```

20. Book return should insert an entry into the Book_Return table and also update the status in Book_Issue table as ‘Returned’. (stored procedure).

```
mysql> CREATE PROCEDURE book_return (
      ->    p_issue_id INT,
      ->    p_actual_return_date DATE,
      ->    p_late_days INT,
      ->    p_late_fee DECIMAL(10,2)
      -> )
      -> BEGIN
      ->    INSERT INTO BOOK_RETURN_57 (Issue_Id, Actual_Date_of_Return,
LateDays, LateFee)
      -> VALUES (p_issue_id, p_actual_return_date, p_late_days,
p_late_fee);
      -> UPDATE BOOK_ISSUE_57
      -> SET Status = 'Returned'
      -> WHERE Issue_Id = p_issue_id;
      -> COMMIT;
      -> END;//
Query OK, 0 rows affected (0.15 sec)

mysql> call book_return (3,'2023-09-17',40,570.00);//
Query OK, 0 rows affected (0.20 sec)

mysql> select * from BOOK_RETURN_57;//
+----------+-----------------------+----------+---------+
| Issue_Id | Actual_Date_of_Return | LateDays | LateFee |
+----------+-----------------------+----------+---------+
|     1    | 2023-09-17            |      2   |   10.00 |
|     2    | 2023-10-01            |      12  |  100.00 |
|     3    | 2023-09-17            |      40  |  570.00 |
+----------+-----------------------+----------+---------+
3 rows in set (0.00 sec)

mysql> select * from BOOK_ISSUE_57;//
+----------+--------------+---------+-----------+------------------------+----------+
| Issue_Id | Date_of_Issue | Book_Id | Member_Id | Expected_Date_of_Return | Status   |
+----------+--------------+---------+-----------+------------------------+----------+
|      1   | 2023-09-01    |       1 |         1 | 2023-09-15             | Issued   |
|      2   | 2023-09-05    |       2 |         2 | 2023-09-19             | Issued   |
|      3   | 2023-03-21    |       5 |         3 | 2023-04-06             | Returned |
+----------+--------------+---------+-----------+------------------------+----------+
3 rows in set (0.00 sec)
```

21. Create a database view ‘Available_Books’, which will list out books that are currently available in the Library.

```
mysql> DELIMITER //
mysql> CREATE VIEW books_available AS
      -> SELECT B.Book_Id,B.Title,A.Name AS Author,P.Name AS Publisher from
BOOK_57 B
      -> INNER JOIN BOOK_AUTHOR_57 BA on B.Book_Id=BA.Book_Id
      -> INNER JOIN AUTHOR_57 A on BA.Author_Id=A.Author_Id
      -> INNER JOIN PUBLISHER_57 P on B.Publisher_Id=P.Publisher_Id
      -> WHERE B.Status="Available";//
```

```
Query OK, 0 rows affected (0.15 sec)

mysql> select * from books_available;//
+---------+---------------------------------+----------------------+---------------------------------------------+
| Book_Id | Title                           | Author               | Publisher                                   |
+---------+---------------------------------+----------------------+---------------------------------------------+
|       1 | Sherlock Holmes                 | Sir Arthur Conan Doyle| Shueisha                                    |
|       2 | GOT (VOL 1&2)                   | George R. R. Martin  | Bantam Spectra (US), Voyager Books (UK)     |
|       3 | Randamoozham                    | M. T. Vasudevan Nair | Current Books                               |
|       4 | Aadujeevitham                   | Benyamin             | Penguin Books (English), Green Books (Malayalam) |
|       5 | RAM C/O ANANDHI                 | Akhil P Dharmajan    | DC Books                                    |
|       6 | Ponniyin Selvan (VOL 1&2)       | Kalki Krishnamurthy  | Vanathi Pathippagam                         |
|       7 | 2 States:TheStory of My MArriage| Chetan Bhagat        | Rupa Publications Pvt. Ltd                  |
|       8 | Half Girlfriend                 | Chetan Bhagat        | Rupa Publications Pvt. Ltd                  |
+---------+---------------------------------+----------------------+---------------------------------------------+
8 rows in set (0.02 sec)
```

22. Create a database procedure to add, update and delete a book to the Library database (use parameters).

```
mysql> CREATE PROCEDURE ManageBook(A VARCHAR(10),B INT,C VARCHAR(255),D
INT,E DECIMAL(10,2),F INT,G DATE,H INT,I VARCHAR(50))
      -> BEGIN
      -> IF A = 'add' THEN
      -> INSERT INTO BOOK_57 (BookId, Title, LanguageId, MRP, PublisherId,
PublishedDate, Volume, Status)
      -> VALUES (B, C, D, E, F, G, H, I);
      -> ELSEIF A = 'update' THEN
      -> UPDATE BOOK_57
      -> SET
      -> Title = C, Language_Id = D, MRP = E, Publisher_Id = F,
      -> Published_Date = G, Volume = H, Status = I
      -> WHERE Book_Id = B;
      -> ELSEIF A = 'delete' THEN
      -> DELETE FROM BOOK_57
      -> WHERE Book_Id = B;
      -> ELSE
      -> SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid action';
      -> END IF;
      -> END;//
Query OK, 0 rows affected (0.25 sec)

mysql> CALL ManageBook('add', 6, 'Avengers', 1, 20, 2, '2023-08-10', 1,
'Available');//
Query OK, 1 row affected (0.15 sec)
mysql> SELECT * FROM BOOK_57;//

+--------+---------------------------+------------+-------+-------------+---------------+--------+-------------+
| BookId | Title                     | LanguageId | MRP   | PublisherId | PublishedDate | Volume | Status      |
+--------+---------------------------+------------+-------+-------------+---------------+--------+-------------+
|      1 | The Great Adventure       |          1 | 19.99 |           1 | 2023-01-15    |      1 | Available   |
|      2 | Learning SQL              |          2 | 29.99 |           2 | 2022-05-20    |      1 | Available   |
|      3 | Mystery of the Lost Island |         1 | 15.50 |           1 | 2023-07-10    |      2 | Checked Out |
|      4 | Kerala                    |          4 | 20.00 |           2 | 2023-08-10    |      1 | Available   |
|      5 | Malayalam Story           |          4 | 20.00 |           1 | 2023-08-10    |      1 | Available   |
|      6 | Avengers                  |          1 | 20.00 |           2 | 2023-08-10    |      1 | Available   |
+--------+---------------------------+------------+-------+-------------+---------------+--------+-------------+
6 rows in set (0.00 sec)
```

23. Use cursors and create a procedure to print Books Issue Register .

```
mysql> CREATE PROCEDURE BookIssueRegister()
      -> BEGIN
      -> DECLARE done INT DEFAULT FALSE;
      -> DECLARE A INT;
      -> DECLARE B DATE;
      -> DECLARE C INT;
      -> DECLARE D INT;
      -> DECLARE E DATE;
      -> DECLARE F VARCHAR(50);
      -> DECLARE issue_cursor CURSOR FOR
      -> SELECT IssueId, IssueDate, BookId, MemberId, ExpectReturnDate,
Status
      -> FROM BOOK_ISSUE_57;
      -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
      -> OPEN issue_cursor;
      -> WHILE done=0 DO
      -> FETCH issue_cursor INTO A, B, C, D, E, F;
      -> SELECT
      -> A AS IssueID,
      -> B AS IssueDate,
      -> C AS BookID,
      -> D AS MemberID,
      -> E AS ExpectReturnDate,
      -> F AS Status;
      -> END WHILE;
      -> CLOSE issue_cursor;
      -> END;//
Query OK, 0 rows affected (0.18 sec)

mysql> CALL BookIssueRegister();//

+---------+------------+--------+----------+------------------+----------+
| IssueID | IssueDate  | BookID | MemberID | ExpectReturnDate | Status   |
+---------+------------+--------+----------+------------------+----------+
|     1   | 2023-09-01 |      1 |         1| 2023-09-15       | Returned |
+---------+------------+--------+----------+------------------+----------+
1 row in set (0.06 sec)


+---------+------------+--------+----------+------------------+---------+
| IssueID | IssueDate  | BookID | MemberID | ExpectReturnDate | Status  |
+---------+------------+--------+----------+------------------+---------+
|     2   | 2023-09-05 |      2 |         2| 2023-09-12       | Pending |
+---------+------------+--------+----------+------------------+---------+
1 row in set (0.06 sec)
```

```
+---------+------------+--------+----------+------------------+---------+
| IssueID | IssueDate  | BookID | MemberID | ExpectReturnDate | Status  |
+---------+------------+--------+----------+------------------+---------+
|     3   | 2023-09-10 |      3 |         1| 2023-09-20       | Overdue |
+---------+------------+--------+----------+------------------+---------+
1 row in set (0.06 sec)



+---------+------------+--------+----------+------------------+----------+
| IssueID | IssueDate  | BookID | MemberID | ExpectReturnDate | Status   |
+---------+------------+--------+----------+------------------+----------+
|     4 | 2023-09-15 |        5 |         3 | 2023-09-30    | Returned   |
+---------+------------+--------+----------+------------------+----------+
1 row in set (0.06 sec)

+---------+------------+--------+----------+------------------+----------+
| IssueID | IssueDate  | BookID | MemberID | ExpectReturnDate | Status   |
+---------+------------+--------+----------+------------------+----------+
|     4 | 2023-09-15 |        5 |         3 | 2023-09-30    | Returned   |
+---------+------------+--------+----------+------------------+----------+
1 row in set (0.06 sec)
```

24. Create a history table (you may use the same structure without any keys) for the MEMBER table and copy the original values of the row being updated to the history table using a TRIGGER.

```
mysql> CREATE TABLE Member_History_57 (MemberId INT, Name VARCHAR(255),
BranchCode VARCHAR(20), RollNo VARCHAR(20), PhoneNumber VARCHAR(20), EMail
VARCHAR(255), JoinDate DATE, Status VARCHAR(50));//
Query OK, 0 rows affected (0.78 sec)

mysql> CREATE TRIGGER Member_Update
      -> BEFORE UPDATE ON MEMBER_57
      -> FOR EACH ROW
      -> BEGIN
      -> INSERT INTO Member_History_57 (
      -> MemberId,
      -> Name,
      -> BranchCode,
      -> RollNo,
      -> PhoneNumber,
      -> EMail,
      -> JoinDate,
      -> Status)
      -> VALUES (
      -> OLD.MemberId,
      -> OLD.Name,
      -> OLD.BranchCode,
      -> OLD.RollNo,
```

```
       -> OLD.PhoneNumber,
       -> OLD.EMail,
       -> OLD.JoinDate,
       -> OLD.Status);
       -> END;//
Query OK, 0 rows affected (0.18 sec)


mysql> SELECT * FROM MEMBER_57;//
+----------+-------------+------------+--------+--------------+-------------------------+------------+--------+
| MemberId | Name        | BranchCode | RollNo | PhoneNumber  | EMail                   | JoinDate   | Status |
+----------+-------------+------------+--------+--------------+-------------------------+------------+--------+
|        1 | Alice Brown | BC001      | R123   | 456-789-0123 | alice.brown@example.com | 2023-01-01 | Active |
|        2 | Bob White   | BC002      | R456   | 567-890-1234 | bob.white@example.com   | 2022-02-01 | Active |
|        3 | Shreya Nair | BC003      | R789   | 678-901-2345 | shreya.nair@example.com | 2023-03-15 | Active |
|        4 | Anjali Kumar | BC004      | R456   | 234-567-8901 | anjali.kumar@example.com | 2022-03-15 | Active |
+----------+-------------+------------+--------+--------------+-------------------------+------------+--------+
4 rows in set (0.00 sec)

mysql> UPDATE MEMBER_57 SET BranchCode='BC015' WHERE MemberId=1;//
Query OK, 1 row affected (0.16 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Member_History_57;//
+----------+-------------+------------+--------+--------------+-------------------------+------------+--------+
| MemberId | Name        | BranchCode | RollNo | PhoneNumber  | EMail                   | JoinDate   | Status |
+----------+-------------+------------+--------+--------------+-------------------------+------------+--------+
|        1 | Alice Brown | BC001      | R123   | 456-789-0123 | alice.brown@example.com | 2023-01-01 | Active |
+----------+-------------+------------+--------+--------------+-------------------------+------------+--------+
1 row in set (0.00 sec)
```

Result:

Queries are executed successfully and output obtained.

CYCLE-4

GROUP PROJECT

Patient Management System

Team Members

34, Gowripriya Biju, IDK22CS038
57, Vidya Roy , IDK22CS063
59, Vivek D V, IDK22CS066

Abstract

The Patient Management System is a comprehensive Java-based application designed to efficiently manage patient records within a healthcare environment. Featuring a user-friendly graphical user interface (GUI) built with Swing, the system allows users to perform a variety of essential operations including adding, updating, deleting, searching, and displaying patient information. Upon launching the application, users are presented with a dropdown menu to select their desired action, which triggers the appropriate forms for data entry and management.

The system includes functionalities for entering patient details such as name, age, gender, department, and contact information, which are securely stored in a MySQL database. It facilitates updating existing patient records through a unique patient ID and allows for the deletion of records after confirming the specified ID's existence. Users can also search for specific patients using their ID and view all patient records in a structured format for enhanced visibility.

With a connection to a MySQL database via JDBC, the application ensures efficient and secure data handling. Robust error handling mechanisms are in place to provide users with clear notifications in case of any issues during database operations. Overall, the Enhanced Patient Management System aims to streamline patient management tasks, significantly improving operational efficiency in healthcare settings.

1.ER Diagram

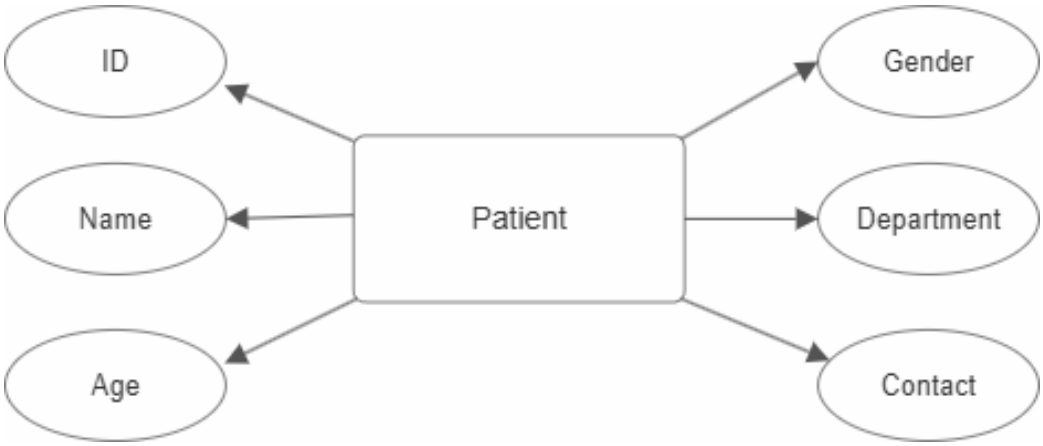


Figure 1:ER Diagram

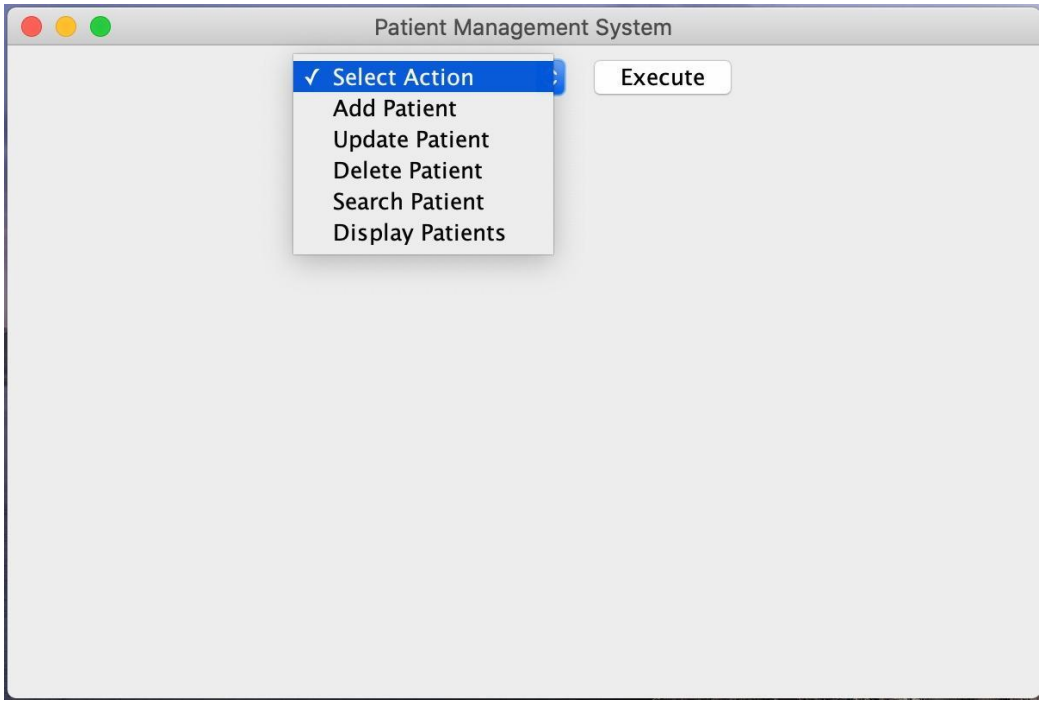
2.Relational Schema

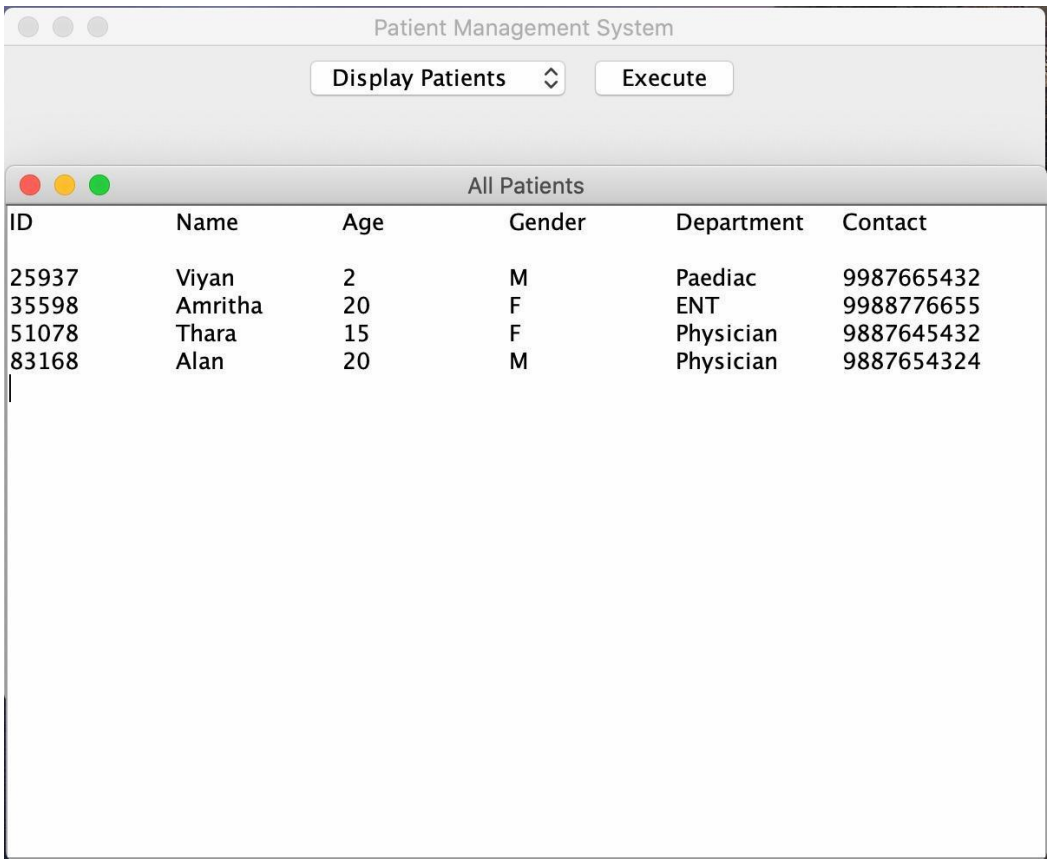
PATIENT

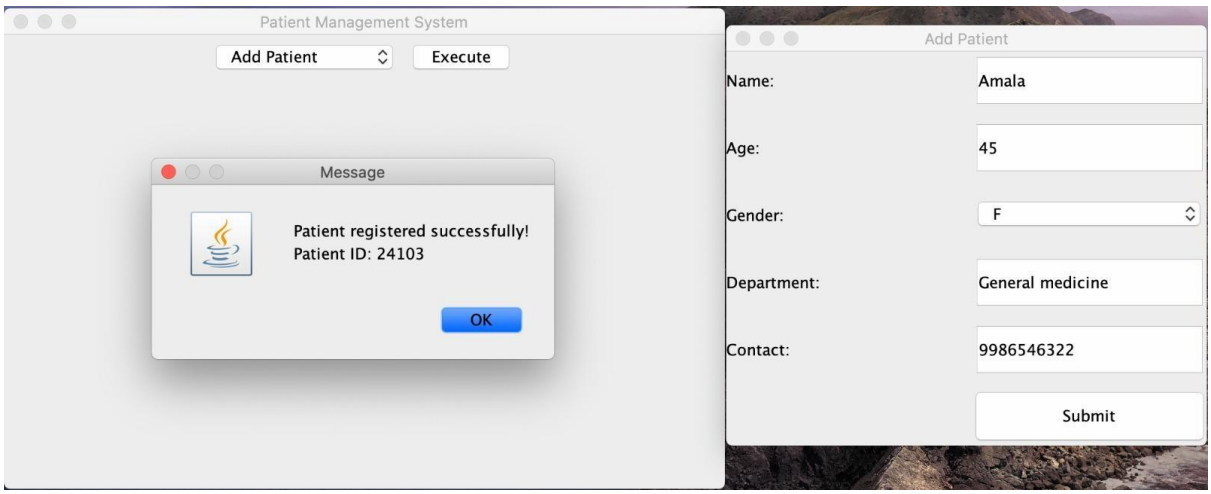
ID	Name	Age	Gender	Department	Contact

Figure 2: Relational Schema

3.Screenshot







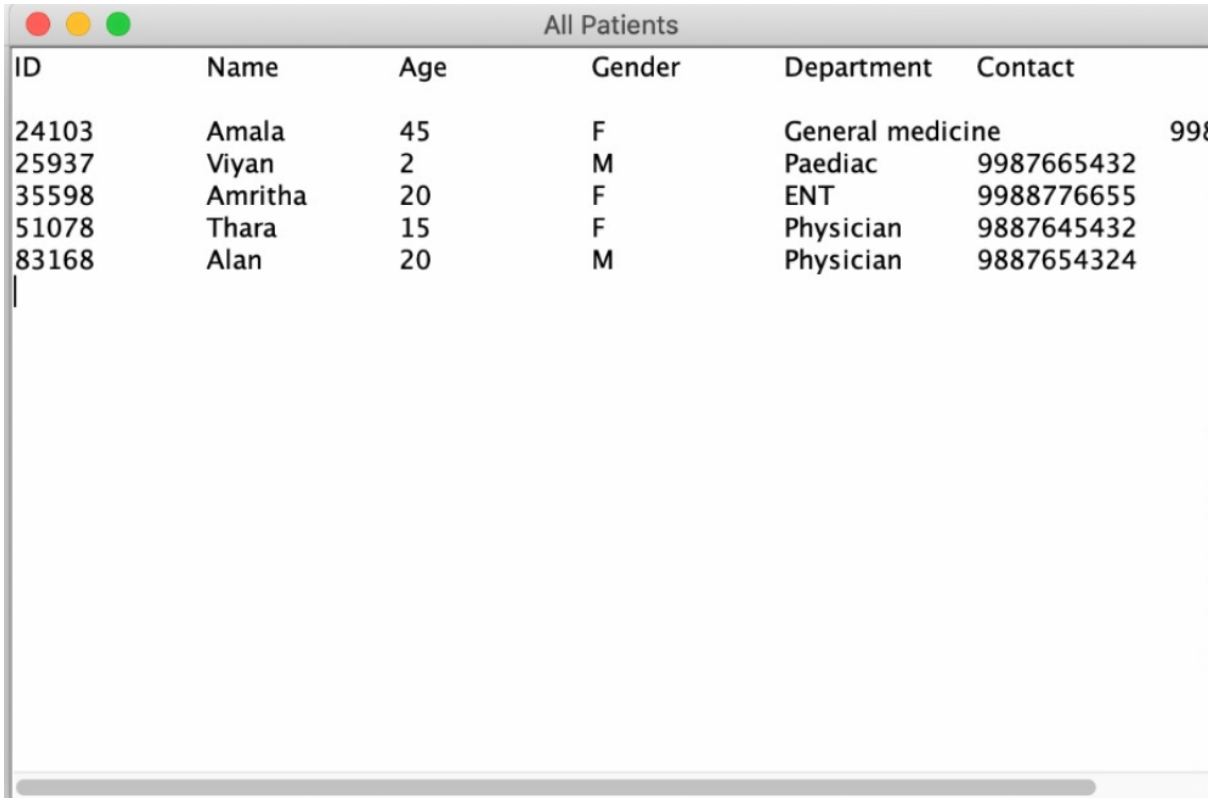


Figure 3: Insert

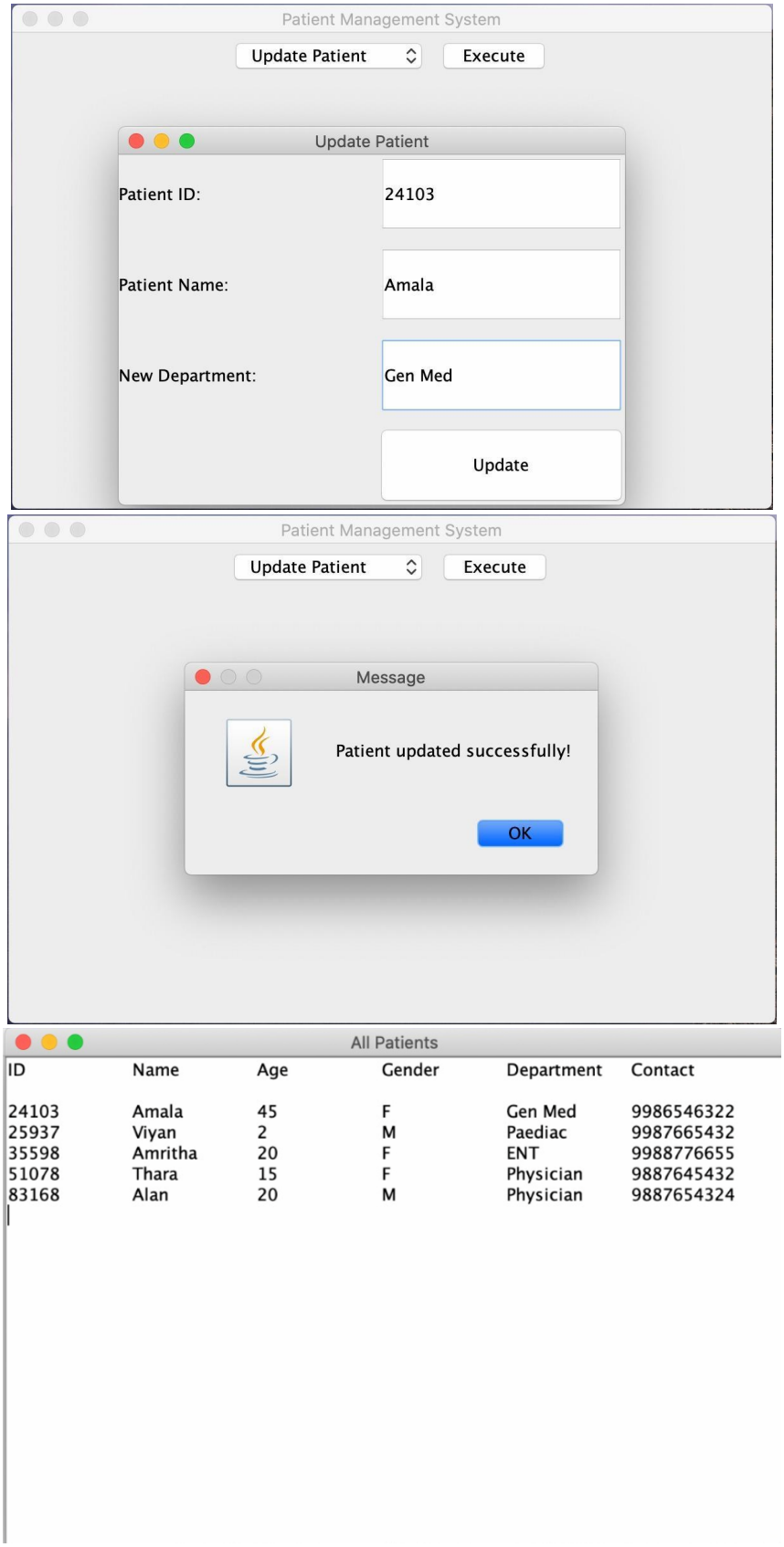
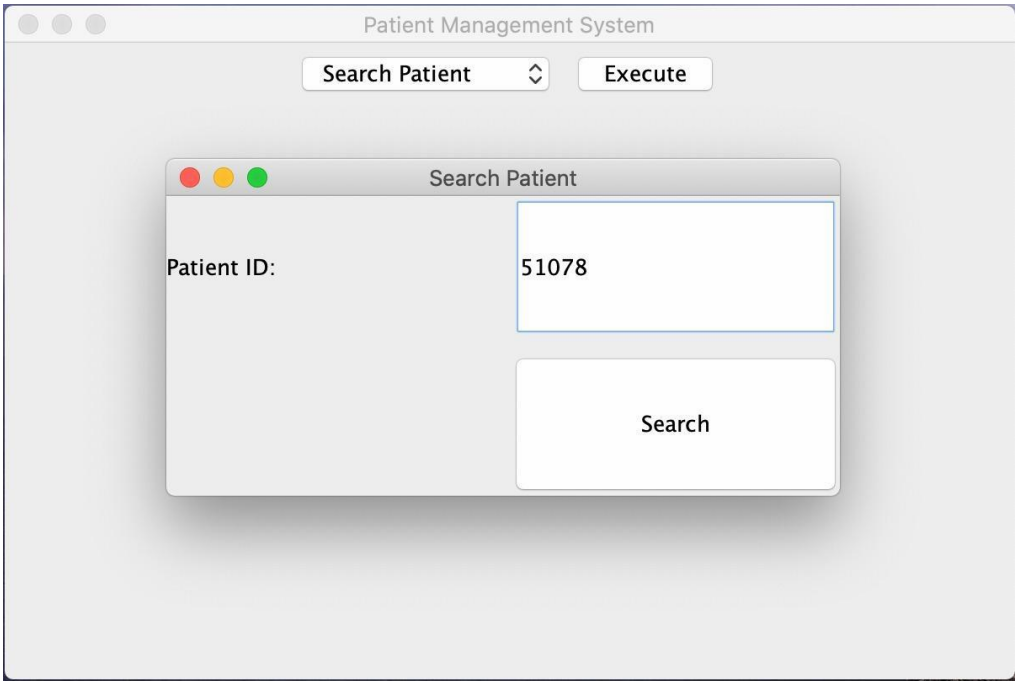


Figure 4:Update



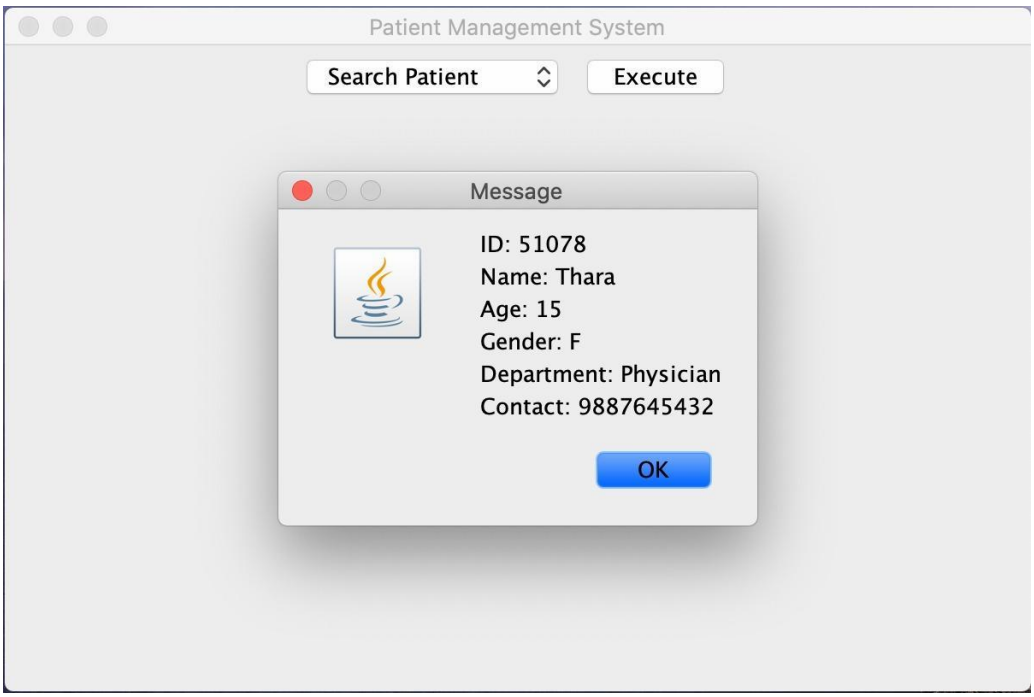


Figure 5: Search

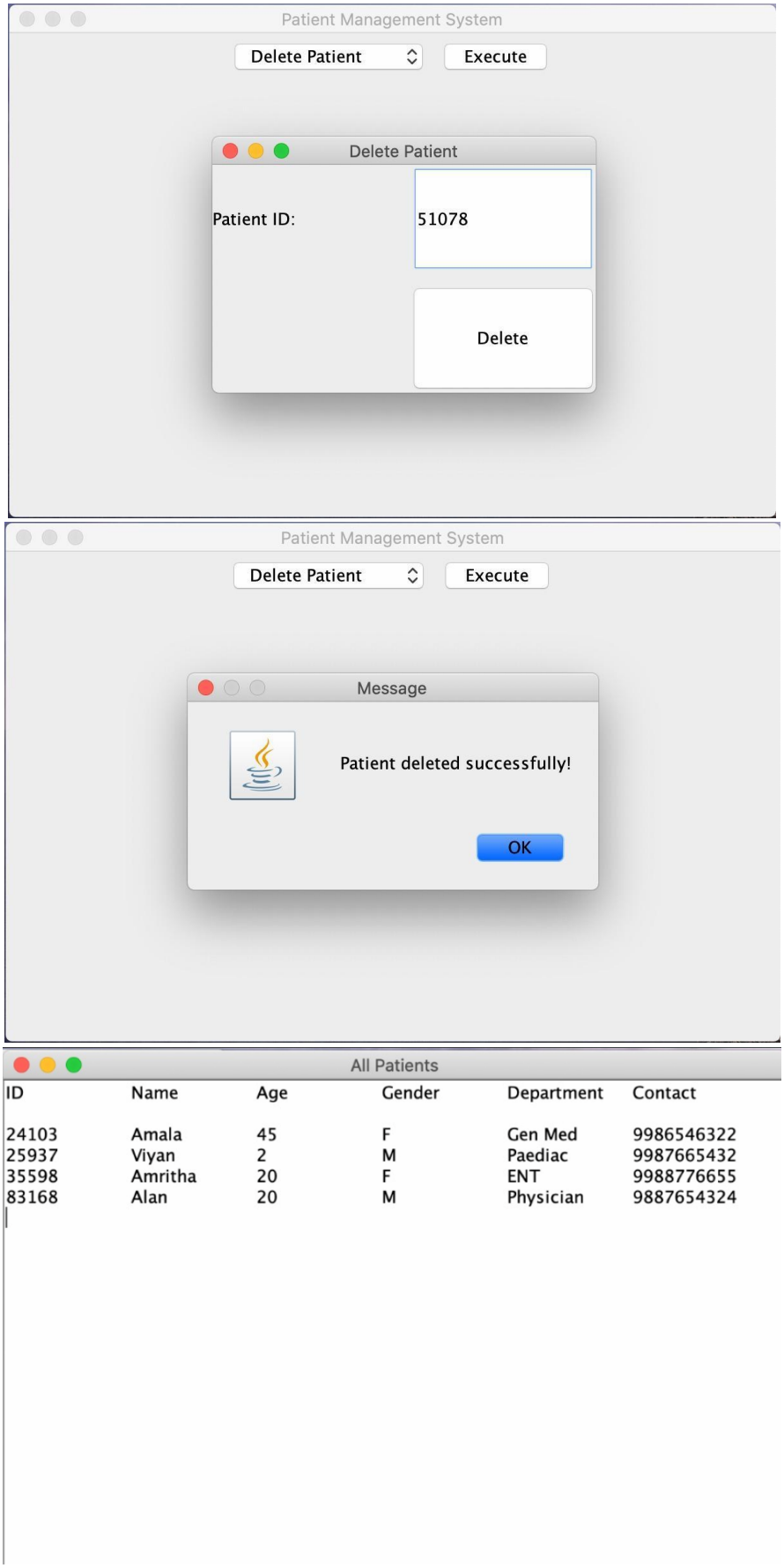


Figure 6:Delete

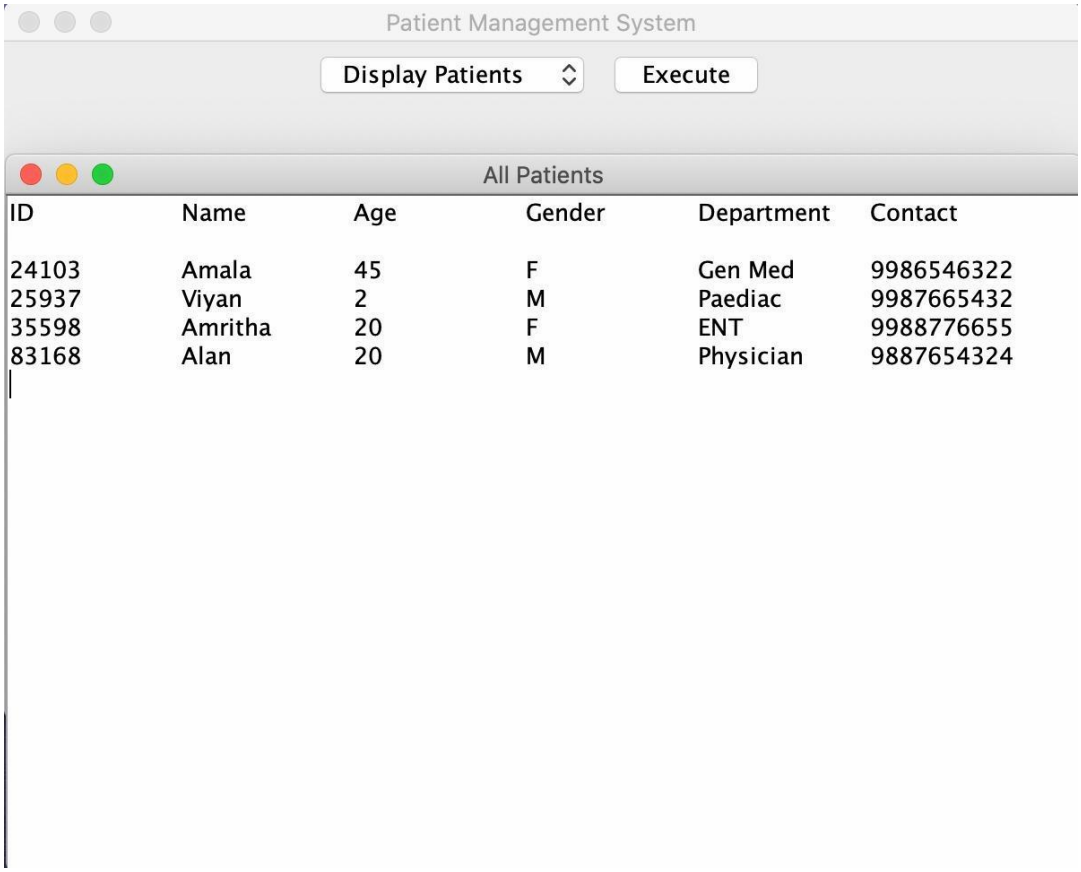


Figure 7: Display

4.Program Code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.Random;

public class EnhancedPatientManagementGUI extends JFrame {
    private static final String URL = "jdbc:mysql://localhost:3306/PMS";
    private static final String USER = "root";
    private static final String PASSWORD = "root@123";

    private JComboBox<String> actionDropdown;

    public EnhancedPatientManagementGUI() {
        setTitle("Patient Management System");
        setSize(600, 400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        actionDropdown = new JComboBox<>(new String[]{
            "Select Action",
            "Add Patient",
            "Update Patient",
            "Delete Patient",
            "Search Patient",
            "Display Patients"
        });

        JButton executeButton = new JButton("Execute");

        add(actionDropdown);
        add(executeButton);

        setVisible(true);

        executeButton.addActionListener(e -> executeAction());
    }

    private void executeAction() {
        String selectedAction = (String) actionDropdown.getSelectedItem();
```

```
        if ("Add Patient".equals(selectedAction)) {
            showAddPatientForm();
        } else if ("Update Patient".equals(selectedAction)) {
            showUpdatePatientForm();
        } else if ("Delete Patient".equals(selectedAction)) {
            showDeletePatientForm();
        } else if ("Search Patient".equals(selectedAction)) {
            showSearchPatientForm();
        } else if ("Display Patients".equals(selectedAction)) {
            showAllPatientsForm();
        } else {
            JOptionPane.showMessageDialog(this, "Please select a valid action.");
        }
    }

    private void showAddPatientForm() {
        JFrame addPatientFrame = new JFrame("Add Patient");
        addPatientFrame.setSize(400, 350);
        addPatientFrame.setLayout(new GridLayout(6, 2, 10, 10));

        JLabel nameLabel = new JLabel("Name:");
        JTextField nameField = new JTextField();
        JLabel ageLabel = new JLabel("Age:");
        JTextField ageField = new JTextField();
        JLabel genderLabel = new JLabel("Gender:");
        JComboBox<String> genderField = new JComboBox<>(new String[]{"M", "F",
"O"});
        JLabel specialtyLabel = new JLabel("Department:");
        JTextField specialtyField = new JTextField();
        JLabel contactLabel = new JLabel("Contact:");
        JTextField contactField = new JTextField();
        JButton submitButton = new JButton("Submit");

        addPatientFrame.add(nameLabel);
        addPatientFrame.add(nameField);
        addPatientFrame.add(ageLabel);
        addPatientFrame.add(ageField);
        addPatientFrame.add(genderLabel);
        addPatientFrame.add(genderField);
        addPatientFrame.add(specialtyLabel);
        addPatientFrame.add(specialtyField);
        addPatientFrame.add(contactLabel);
        addPatientFrame.add(contactField);
```

```
    addPatientFrame.add(new JLabel());
    addPatientFrame.add(submitButton);

    addPatientFrame.setVisible(true);

    submitButton.addActionListener(e -> {
        String name = nameField.getText();
        int age = Integer.parseInt(ageField.getText());
        String gender = (String) genderField.getSelectedItem();
        String specialty = specialtyField.getText();
        String contact = contactField.getText();
        addPatientToDatabase(name, age, gender, specialty, contact);
        addPatientFrame.dispose();
    });
}

private void showUpdatePatientForm() {
    JFrame updatePatientFrame = new JFrame("Update Patient");
    updatePatientFrame.setSize(400, 300);
    updatePatientFrame.setLayout(new GridLayout(4, 2, 10, 10));

    JLabel idLabel = new JLabel("Patient ID:");
    JTextField idField = new JTextField();
    JLabel nameLabel = new JLabel("Patient Name:");
    JTextField nameField = new JTextField();
    JLabel specializationLabel = new JLabel("New Department:");
    JTextField specializationField = new JTextField();
    JButton submitButton = new JButton("Update");

    updatePatientFrame.add(idLabel);
    updatePatientFrame.add(idField);
    updatePatientFrame.add(nameLabel);
    updatePatientFrame.add(nameField);
    updatePatientFrame.add(specializationLabel);
    updatePatientFrame.add(specializationField);
    updatePatientFrame.add(new JLabel());
    updatePatientFrame.add(submitButton);

    updatePatientFrame.setVisible(true);

    submitButton.addActionListener(e -> {
        int id = Integer.parseInt(idField.getText());
        String name = nameField.getText();
        String newSpecialization = specializationField.getText();
```

```
        updatePatientInDatabase(id, name, newSpecialization);
        updatePatientFrame.dispose();
    });
}

private void showDeletePatientForm() {
    JFrame deletePatientFrame = new JFrame("Delete Patient");
    deletePatientFrame.setSize(300, 200);
    deletePatientFrame.setLayout(new GridLayout(2, 2, 10, 10));

    JLabel idLabel = new JLabel("Patient ID:");
    JTextField idField = new JTextField();
    JButton deleteButton = new JButton("Delete");

    deletePatientFrame.add(idLabel);
    deletePatientFrame.add(idField);
    deletePatientFrame.add(new JLabel());
    deletePatientFrame.add(deleteButton);

    deletePatientFrame.setVisible(true);

    deleteButton.addActionListener(e -> {
        int id = Integer.parseInt(idField.getText());
        deletePatientFromDatabase(id);
        deletePatientFrame.dispose();
    });
}

private void showSearchPatientForm() {
    JFrame searchPatientFrame = new JFrame("Search Patient");
    searchPatientFrame.setSize(400, 200);
    searchPatientFrame.setLayout(new GridLayout(2, 2, 10, 10));

    JLabel idLabel = new JLabel("Patient ID:");
    JTextField idField = new JTextField();
    JButton searchButton = new JButton("Search");

    searchPatientFrame.add(idLabel);
    searchPatientFrame.add(idField);
    searchPatientFrame.add(new JLabel());
    searchPatientFrame.add(searchButton);

    searchPatientFrame.setVisible(true);
```

```
        searchButton.addActionListener(e -> {
            int id = Integer.parseInt(idField.getText());
            searchPatientInDatabase(id);
        });
    }

    private void showAllPatientsForm() {
        JFrame allPatientsFrame = new JFrame("All Patients");
        allPatientsFrame.setSize(600, 400);
        allPatientsFrame.setLayout(new BorderLayout());

        JTextArea textArea = new JTextArea();
        textArea.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(textArea);
        allPatientsFrame.add(scrollPane, BorderLayout.CENTER);

        Connection connection = getConnection();
        if (connection != null) {
            try {
                String query = "SELECT * FROM patients";
                Statement statement = connection.createStatement();
                ResultSet resultSet = statement.executeQuery(query);

                StringBuilder patientDetails = new
StringBuilder("ID\tName\tAge\tGender\tDepartment\tContact\n");
                patientDetails.append("\n");

                while (resultSet.next()) {
                    int id = resultSet.getInt("id");
                    String name = resultSet.getString("name");
                    int age = resultSet.getInt("age");
                    String gender = resultSet.getString("gender");
                    String specialty = resultSet.getString("specialty");
                    String contact = resultSet.getString("contact");
                    patientDetails.append(id).append("\t")
                            .append(name).append("\t")
                            .append(age).append("\t")
                            .append(gender).append("\t")
                            .append(specialty).append("\t")
                            .append(contact).append("\n");
                }

                textArea.setText(patientDetails.toString());
            } catch (SQLException e) {
```

```
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error while retrieving patient data:
" + e.getMessage());
        }
    }

    allPatientsFrame.setVisible(true);
}

private Connection getConnection() {
    try {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

private void addPatientToDatabase(String name, int age, String gender, String
specialty, String contact) {
    Connection connection = getConnection();
    if (connection != null) {
        try {
            Random random = new Random();
            int patientId = 10000 + random.nextInt(90000);

            String query = "INSERT INTO patients (id, name, age, gender, specialty,
contact) VALUES (?, ?, ?, ?, ?, ?)";
            PreparedStatement preparedStatement =
connection.prepareStatement(query);
            preparedStatement.setInt(1, patientId);
            preparedStatement.setString(2, name);
            preparedStatement.setInt(3, age);
            preparedStatement.setString(4, gender);
            preparedStatement.setString(5, specialty);
            preparedStatement.setString(6, contact);

            int rowsAffected = preparedStatement.executeUpdate();
            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(null, "Patient registered
successfully!\nPatient ID: " + patientId);
            } else {
                JOptionPane.showMessageDialog(null, "Failed to register patient.");
            }
```

```
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error while registering patient: " +
e.getMessage());
        }
    }
}


private void deletePatientFromDatabase(int id) {
Connection connection = getConnection();
if (connection != null) {
    try {
        String checkQuery = "SELECT * FROM patients WHERE id = ?";
        PreparedStatement checkStatement =
connection.prepareStatement(checkQuery);
        checkStatement.setInt(1, id);
        ResultSet checkResult = checkStatement.executeQuery();

        if (checkResult.next()) {
            String deleteQuery = "DELETE FROM patients WHERE id = ?";
            PreparedStatement deleteStatement =
connection.prepareStatement(deleteQuery);
            deleteStatement.setInt(1, id);
            deleteStatement.executeUpdate();
            JOptionPane.showMessageDialog(this, "Patient deleted successfully!");
        } else {
            JOptionPane.showMessageDialog(this, "Error: Patient ID not found.");
        }
    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error while deleting patient: " +
e.getMessage());
    }
}
}

private void updatePatientInDatabase(int id, String name, String newSpecialization) {
Connection connection = getConnection();
if (connection != null) {
    try {
        String checkQuery = "SELECT * FROM patients WHERE id = ?";
```

```
            PreparedStatement checkStatement =
connection.prepareStatement(checkQuery);
            checkStatement.setInt(1, id);
            ResultSet checkResult = checkStatement.executeQuery();

            if (checkResult.next()) {
                String updateQuery = "UPDATE patients SET name = ?, specialty = ?
WHERE id = ?";
                PreparedStatement updateStatement =
connection.prepareStatement(updateQuery);
                updateStatement.setString(1, name);
                updateStatement.setString(2, newSpecialization);
                updateStatement.setInt(3, id);
                updateStatement.executeUpdate();
                JOptionPane.showMessageDialog(this, "Patient updated successfully!");
            } else {
                JOptionPane.showMessageDialog(this, "Error: Patient ID not found.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error while updating patient: " +
e.getMessage());
        }
    }
}


    private void searchPatientInDatabase(int id) {
        Connection connection = getConnection();
        if (connection != null) {
            try {
                String query = "SELECT * FROM patients WHERE id = ?";
                PreparedStatement preparedStatement =
connection.prepareStatement(query);
                preparedStatement.setInt(1, id);

                ResultSet resultSet = preparedStatement.executeQuery();
                if (resultSet.next()) {
                    String patientInfo = "ID: " + resultSet.getInt("id") +
                            "\nName: " + resultSet.getString("name") +
                            "\nAge: " + resultSet.getInt("age") +
                            "\nGender: " + resultSet.getString("gender") +
                            "\nDepartment: " + resultSet.getString("specialty") +
                            "\nContact: " + resultSet.getString("contact");
```

```
                JOptionPane.showMessageDialog(this, patientInfo);
            } else {
                JOptionPane.showMessageDialog(this, "Patient not found.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

public static void main(String[] args) {
    new EnhancedPatientManagementGUI();
}
}
```