GOVERNMENT ENGINEERING COLLEGE

IDUKKI, PAINAVU- 685 603

CSL 333 DATABASE MANAGENMENT SYSTEMS

LABORATORY RECORD

NAME : .

SEMESTER : S5 **ROLL No :** .

BRANCH : COMPUTER SCIENCE

GOVERNMENT ENGINEERING COLLEGE
IDUKKI, PAINAVU 685 603

CSL 333 DATABASE MANAGENMENT SYSTEMS
LABORATORY RECORD

NAME : .

SEMESTER : S5 **ROLL No :** .

BRANCH : COMPUTER SCIENCE

Certified that the Bonafide Record of work done by

Staff-in-charge

Uni.Ex:Reg:No : **of Month/Year………………………**

External Examiner……………………Internal Examiner………….............

INDEX

Sl. No	NAME OF EXPERIMENT	DATE	PAGE No.	Initials of Teacher

Experiment no:- 1

SQL PRACTISE 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

```
user@user-Desktop:~$ sudo mysql -u root -p
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.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> use userDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> CREATE TABLE Employee (ID INT,DeptID INT,Name VARCHAR(15),Design VARCHAR(15),Basic
DECIMAL(10,2),Gender CHAR(1));
Query OK, 0 rows affected (0.04 sec)
```

1. Get the description of the table.

```
mysql> DESC Employee;
+--------+---------------+------+-----+---------+-------+
| Field  | Type          | Null | Key | Default | Extra |
+--------+---------------+------+-----+---------+-------+
| ID     | int           | YES  |     | NULL    |       |
| DeptID | int           | YES  |     | NULL    |       |
| Name   | varchar(15)   | YES  |     | NULL    |       |
| Design | varchar(15)   | YES  |     | NULL    |       |
| Basic  | decimal(10,2) | YES  |     | NULL    |       |
| Gender | char(1)       | YES  |     | NULL    |       |
+--------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

2. Display all the records from the above table.

```
mysql> SELECT * FROM Employee;
+------+--------+---------+---------+---------+--------+
| ID   | DeptID | Name    | Design  | Basic   | Gender |
+------+--------+---------+---------+---------+--------+
|  101 |      1 | Ram     | Typist  | 2000.00 | M      |
|  102 |      2 | Arun    | Analyst | 6000.00 | M      |
|  121 |      1 | Ruby    | Typist  | 2010.00 | F      |
|  156 |      3 | Mary    | Manager | 4500.00 | F      |
|  123 |      2 | Mridula | Analyst | 6000.00 | F      |
|  114 |      4 | Menon   | Clerk   | 1500.00 | M      |
|  115 |      4 | Tim     | Clerk   | 1500.00 | M      |
|  127 |      2 | Kiran   | Manager | 4000.00 | M      |
+------+--------+---------+---------+---------+--------+
8 rows in set (0.00 sec)
```

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

```
mysql>
mysql> SELECT ID, Name, Design, Basic FROM Employee;
+------+---------+---------+---------+
| ID   | Name    | Design  | Basic   |
+------+---------+---------+---------+
|  101 | Ram     | Typist  | 2000.00 |
|  102 | Arun    | Analyst | 6000.00 |
|  121 | Ruby    | Typist  | 2010.00 |
|  156 | Mary    | Manager | 4500.00 |
|  123 | Mridula | Analyst | 6000.00 |
|  114 | Menon   | Clerk   | 1500.00 |
|  115 | Tim     | Clerk   | 1500.00 |
|  127 | Kiran   | Manager | 4000.00 |
+------+---------+---------+---------+
8 rows in set (0.00 sec)
```

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

```
mysql>
mysql> SELECT ID, Name FROM Employee WHERE DeptID = 2;
+------+---------+
| ID   | Name    |
+------+---------+
|  102 | Arun    |
|  123 | Mridula |
|  127 | Kiran   |
+------+---------+
3 rows in set (0.00 sec)
```

5. Display ID, name, desig,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>
mysql> select ID,name,design,basic,deptID,basic,basic*0.075 DA,basic*0.05
HRA,basic+basic*0.075+basic*0.05 NET_SAL from employee;
+------+---------+---------+--------+---------+-----------+----------+------------+
| ID   | Name    | Desig   | DeptID | Basic   | DA        | HRA      | NET_SAL    |
+------+---------+---------+--------+---------+-----------+----------+------------+
|  101 | Ram     | Typist  |      1 | 2000.00 | 150.00000 | 100.0000 | 2250.00000 |
|  102 | Arun    | Analyst |      2 | 6000.00 | 450.00000 | 300.0000 | 6750.00000 |
|  121 | Ruby    | Typist  |      1 | 2010.00 | 150.75000 | 100.5000 | 2261.25000 |
|  156 | Mary    | Manager |      3 | 4500.00 | 337.50000 | 225.0000 | 5062.50000 |
|  123 | Mridula | Analyst |      2 | 6000.00 | 450.00000 | 300.0000 | 6750.00000 |
|  114 | Menon   | Clerk   |      4 | 1500.00 | 112.50000 |  75.0000 | 1687.50000 |
|  115 | Tim     | Clerk   |      4 | 1500.00 | 112.50000 |  75.0000 | 1687.50000 |
|  127 | Kiran   | Manager |      2 | 4000.00 | 300.00000 | 200.0000 | 4500.00000 |
+------+---------+---------+--------+---------+-----------+----------+------------+
8 rows in set (0.00 sec)
```

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

```
mysql>
mysql> SELECT ID,Name,Design,DeptID,Basic FROM Employee ORDER BY Basic DESC;
+------+---------+---------+--------+---------+
| ID   | Name    | Design  | 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 |
|  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>
mysql> SELECT ID,Name,Design AS Desig,DeptID,Basic FROM Employee WHERE Design = 'Typist';
+------+------+--------+--------+---------+
| ID   | Name | Desig  | DeptID | Basic   |
+------+------+--------+--------+---------+
|  101 | Ram  | Typist |      1 | 2000.00 |
|  121 | Ruby | Typist |      1 | 2010.00 |
+------+------+--------+--------+---------+
2 rows in set (0.00 sec)
```

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

```
mysql>
mysql> SELECT ID,Name,Design AS Desig,DeptID,Basic FROM Employee WHERE Design IN
('Analyst', 'Manager');

+------+---------+---------+--------+---------+
| ID   | Name    | Desig   | DeptID | Basic   |
+------+---------+---------+--------+---------+
|  102 | Arun    | Analyst |      2 | 6000.00 |
|  156 | Mary    | Manager |      3 | 4500.00 |
|  123 | Mridula | Analyst |      2 | 6000.00 |
|  127 | Kiran   | Manager |      2 | 4000.00 |
+------+---------+---------+--------+---------+
4 rows in set (0.00 sec)
```

9. Display all designations without duplicate values.

```
mysql>
mysql> SELECT DISTINCT Design AS Unique_Designations FROM Employee;
+---------------------+
| Unique_Designations |
+---------------------+
| Typist              |
| Analyst             |
| Manager             |
| Clerk               |
+---------------------+
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>
mysql> SELECT ID,Name,DeptID,Basic FROM Employee WHERE (Design IN ('Manager', 'Clerk'))
AND (Basic BETWEEN 1400 AND 4500);
```

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

11. Display the number of male staff members

```
mysql>
mysql> SELECT COUNT(*) AS Male_Staff_Count FROM Employee WHERE Gender = 'M';
+------------------+
| Male_Staff_Count |
+------------------+
|                5 |
+------------------+
1 row in set (0.00 sec)
```

12. Find the maximum salary of each designation.

```
mysql>
mysql> SELECT Design AS Designation,MAX(Basic) AS Max_Salary FROM Employee WHERE Design IN
('Typist', 'Analyst', 'Manager', 'Clerk') GROUP BY Design;

+-------------+------------+
| Designation | Max_Salary |
+-------------+------------+
| Typist      |    2010.00 |
| Analyst     |    6000.00 |
| Manager     |    4500.00 |
| Clerk       |    1500.00 |
+-------------+------------+
4 rows in set (0.00 sec)
```

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

```
mysql>
mysql> ALTER TABLE Employee ADD COLUMN Manager_ID INT;
Query OK, 0 rows affected (0.11 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>
mysql> UPDATE Employee SET Manager_ID = NULL WHERE ID IN (101, 121, 156);
Query OK, 0 rows affected (0.00 sec)
Rows matched: 3  Changed: 0  Warnings: 0

mysql>
mysql> UPDATE Employee SET Manager_ID = 101 WHERE ID = 102;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql>
mysql> UPDATE Employee SET Manager_ID = 102 WHERE ID IN (123, 114, 115);
Query OK, 0 rows affected (0.00 sec)
Rows matched: 3  Changed: 0  Warnings: 0

mysql>
mysql> UPDATE Employee SET Manager_ID = 121 WHERE ID = 127;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```

15. Display the manager id of the employee Ram.

```
mysql>
mysql> SELECT Manager_ID FROM Employee WHERE Name = 'Ram';

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

16. Display the employee names and their manager name.

```
mysql>
mysql> SELECT E1.Name AS Employee_Name,E2.Name AS Manager_Name FROM Employee E1 LEFT JOIN
Employee E2 ON E1.Manager_ID = E2.ID;
+---------------+--------------+
| Employee_Name | Manager_Name |
+---------------+--------------+
| Ram           | NULL         |
| Arun          | Ram          |
| Ruby          | NULL         |
| Mary          | NULL         |
| Mridula       | Arun         |
| Menon         | Arun         |
| Tim           | Arun         |
| Kiran         | Ruby         |
+---------------+--------------+
8 rows in set (0.00 sec)
```

17. Find the average salary of each department.

```
mysql>
mysql> SELECT DeptID AS Department,Design AS Designation,AVG(Basic) AS Average_Salary FROM
Employee WHERE Design IN ('Typist', 'Analyst', 'Manager', 'Clerk') GROUP BY DeptID,
Design;
+------------+-------------+----------------+
| Department | Designation | Average_Salary |
+------------+-------------+----------------+
|          1 | Typist      |    2005.000000 |
|          2 | Analyst     |    6000.000000 |
|          3 | Manager     |    4500.000000 |
|          4 | Clerk       |    1500.000000 |
|          2 | Manager     |    4000.000000 |
+------------+-------------+----------------+
5 rows in set (0.00 sec)
```

18. Find the maximum salary given to employees.

```
mysql>
mysql> SELECT MAX(Basic) AS Max_Salary FROM Employee;

+------------+
| Max_Salary |
+------------+
|    6000.00 |
+------------+
1 row in set (0.00 sec)
```

19. Find the number of employees in each department.

```
mysql>
mysql> SELECT DeptID AS Department,COUNT(*) AS Employee_Count FROM Employee GROUP BY
DeptID;
```

```
+------------+----------------+
| Department | Employee_Count |
+------------+----------------+
|          1 |              2 |
|          2 |              3 |
|          3 |              1 |
|          4 |              2 |
+------------+----------------+
4 rows in set (0.00 sec)
```

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

```
mysql>
mysql> SELECT COUNT(DISTINCT DeptID) AS Department_Count FROM Employee;

+------------------+
| Department_Count |
+------------------+
|                4 |
+------------------+
1 row in set (0.00 sec)
```

21. Display the different designations existing in the organisation.

```
mysql>
mysql> SELECT DISTINCT Design AS Designation FROM Employee;

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

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

```
mysql>
mysql> SELECT COUNT(DISTINCT Design) AS Number_of_Designations FROM Employee;

+------------------------+
| Number_of_Designations |
+------------------------+
|                      4 |
+------------------------+
1 row in set (0.01 sec)
```

23. Display the maximum salary given for female employees.

```
mysql>
mysql> SELECT MAX(Basic) AS Max_Salary FROM Employee WHERE Gender = 'F';
+------------+
| Max_Salary |
+------------+
|    6000.00 |
+------------+
1 row in set (0.00 sec)
```

24. Display the female typist.

```
mysql>
mysql> SELECT ID,Name,Design AS Designation,DeptID AS Department,Basic FROM Employee WHERE
Gender = 'F' AND Design = 'Typist';
```

```
+------+------+-------------+------------+---------+
| ID   | Name | Designation | Department | Basic   |
+------+------+-------------+------------+---------+
|  121 | Ruby | Typist      |          1 | 2010.00 |
+------+------+-------------+------------+---------+
1 row in set (0.00 sec)
```

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

```
mysql>
mysql> SELECT ID,Name,Design AS Designation,DeptID AS Department,Basic FROM Employee WHERE
Gender = 'M' AND Design = 'Clerk' AND Basic > 3000;
Empty set (0.00 sec)
```

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

```
mysql>
mysql> SELECT ID,Name,Design AS Designation,DeptID AS Department,Basic FROM Employee WHERE
(Design = 'Manager' OR Design = 'Analyst') AND DeptID = 2;

+------+---------+-------------+------------+---------+
| ID   | Name    | Designation | Department | Basic   |
+------+---------+-------------+------------+---------+
|  102 | Arun    | Analyst     |          2 | 6000.00 |
|  123 | Mridula | Analyst     |          2 | 6000.00 |
|  127 | Kiran   | Manager     |          2 | 4000.00 |
+------+---------+-------------+------------+---------+
3 rows in set (0.00 sec)
```

27. Display the designation and salary of Ruby.

```
mysql>
mysql> SELECT Design AS Designation,Basic AS Salary FROM Employee WHERE Name = 'Ruby';

+-------------+---------+
| Designation | Salary  |
+-------------+---------+
| Typist      | 2010.00 |
+-------------+---------+
1 row in set (0.00 sec)
```

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

```
mysql>
mysql> ALTER TABLE Employee ADD COLUMN JoiningDate DATE;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

29. Update appropriate values for the joining date field.

```
mysql>
mysql> UPDATE Employee SET JoiningDate = DATE_ADD('2020-01-01', INTERVAL FLOOR(RAND() *
365) DAY);
Query OK, 8 rows affected (0.05 sec)
Rows matched: 8  Changed: 8  Warnings: 0

mysql>
mysql> SELECT *FROM Employee;
```

```
+------+--------+---------+---------+---------+--------+------------+-------------+
| ID   | DeptID | Name    | Design  | Basic   | Gender | Manager_ID | JoiningDate |
+------+--------+---------+---------+---------+--------+------------+-------------+
|  101 |      1 | Ram     | Typist  | 2000.00 | M      |       NULL | 2020-07-24  |
|  102 |      2 | Arun    | Analyst | 6000.00 | M      |        101 | 2020-07-25  |
|  121 |      1 | Ruby    | Typist  | 2010.00 | F      |       NULL | 2020-02-22  |
|  156 |      3 | Mary    | Manager | 4500.00 | F      |       NULL | 2020-01-07  |
|  123 |      2 | Mridula | Analyst | 6000.00 | F      |        102 | 2020-08-30  |
|  114 |      4 | Menon   | Clerk   | 1500.00 | M      |        102 | 2020-04-04  |
|  115 |      4 | Tim     | Clerk   | 1500.00 | M      |        102 | 2020-04-23  |
|  127 |      2 | Kiran   | Manager | 4000.00 | M      |        121 | 2020-10-07  |
+------+--------+---------+---------+---------+--------+------------+-------------+
8 rows in set (0.00 sec)
```

30. Display the details of employees according to their seniority.

```
mysql>
mysql> SELECT ID,Name,Design AS Designation,DeptID AS Department,Basic,JoiningDate FROM
Employee ORDER BY JoiningDate ASC;

+------+---------+-------------+------------+---------+-------------+
| ID   | Name    | Designation | Department | Basic   | JoiningDate |
+------+---------+-------------+------------+---------+-------------+
|  156 | Mary    | Manager     |          3 | 4500.00 | 2020-01-07  |
|  121 | Ruby    | Typist      |          1 | 2010.00 | 2020-02-22  |
|  114 | Menon   | Clerk       |          4 | 1500.00 | 2020-04-04  |
|  115 | Tim     | Clerk       |          4 | 1500.00 | 2020-04-23  |
|  101 | Ram     | Typist      |          1 | 2000.00 | 2020-07-24  |
|  102 | Arun    | Analyst     |          2 | 6000.00 | 2020-07-25  |
|  123 | Mridula | Analyst     |          2 | 6000.00 | 2020-08-30  |
|  127 | Kiran   | Manager     |          2 | 4000.00 | 2020-10-07  |
+------+---------+-------------+------------+---------+-------------+
8 rows in set (0.01 sec)
```

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

```
mysql>
mysql> SELECT ID,Name,Design AS Designation,DeptID AS Department,Basic,JoiningDate FROM
Employee ORDER BY Basic DESC;

+------+---------+-------------+------------+---------+-------------+
| ID   | Name    | Designation | Department | Basic   | JoiningDate |
+------+---------+-------------+------------+---------+-------------+
|  102 | Arun    | Analyst     |          2 | 6000.00 | 2020-07-25  |
|  123 | Mridula | Analyst     |          2 | 6000.00 | 2020-08-30  |
|  156 | Mary    | Manager     |          3 | 4500.00 | 2020-01-07  |
|  127 | Kiran   | Manager     |          2 | 4000.00 | 2020-10-07  |
|  121 | Ruby    | Typist      |          1 | 2010.00 | 2020-02-22  |
|  101 | Ram     | Typist      |          1 | 2000.00 | 2020-07-24  |
|  114 | Menon   | Clerk       |          4 | 1500.00 | 2020-04-04  |
|  115 | Tim     | Clerk       |          4 | 1500.00 | 2020-04-23  |
+------+---------+-------------+------------+---------+-------------+
8 rows in set (0.00 sec)
```

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

```
mysql>
mysql> CREATE TABLE DEPARTMENT (DEPTID INT PRIMARY KEY,DNAME VARCHAR(255));
Query OK, 0 rows affected (0.08 sec)
```

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

```
mysql>
mysql> ALTER TABLE Employee ADD CONSTRAINT FK_Employee_Department FOREIGN KEY (DeptId)
REFERENCES Dept(Dep_ID);;
Query OK, 8 rows affected (0.17 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

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

```
mysql>
mysql>INSERT INTO `Dept` VALUES (1,'DESIGN'),(2,'CODING'),(3,'TESTING'),(4,'RESEARCH');
```

35. Display the employee name and department name.

```
mysql>
mysql> Select  Employee,Dept where Employee.DeptId=Dept.Dep_ID;

+---------+----------+
| Name    | D_name   |
+---------+----------+
| Ram     | DESIGN   |
| Arun    | CODING   |
| Ruby    | DESIGN   |
| Mary    | TESTING  |
| Mridula | CODING   |
| Menon   | RESEARCH |
| Tim     | RESEARCH |
| Kiran   | CODING   |
+---------+----------+
8 rows in set (0.02 sec)
```

36. Display the department name of employee Arun.

```
mysql>
mysql> select D_name from Dept where Dep_ID=(select Deptid from Employee Where
Name="Arun");
+--------+
| D_name |
+--------+
| CODING |
+--------+
1 row in set (0.00 sec)
```

37. Display the salary given by DESIGN department.

```
mysql>
mysql> select Base from Employee where DeptId=(select Dep_ID from Dept where
D_name=("DESIGN"));
+------+
| Base |
+------+
| 2000 |
| 2010 |
+------+
2 rows in set (0.00 sec)
```

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

```
select* from Employee where Designation="Typist" and DeptID=(select DeptID from Dept where
D_name="DESIGN");

+------+--------+------+------------+------+--------+------+------+---------+------------
+------------+
| ID   | DeptId | Name | Designation | Base | Gender | HRA  | DA   | NET_SAL | Manager_Id
| Join_date  |
+------+--------+------+------------+------+--------+------+------+---------+------------
+------------+
| 101  |      1 | Ram  | typist      | 2000 | M      | 1000 | 1500 |    4500 |        101
| 2000-04-03 |
| 121  |      1 | Ruby | typist      | 2010 | F      | 1000 | 1508 |    4518 |        101
| 2000-12-20 |
+------+--------+------+------------+------+--------+------+------+---------+------------
+------------+
2 rows in set (0.00 sec)
```

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

```
select Base from Employee where DeptID=(select Dep_ID from Dept where D_name="RESEARCH");
+------+
| Base |
+------+
| 1500 |
| 1500 |
+------+
2 rows in set (0.00 sec)
```

40. List the female employees working in TESTING department.

```
mysql>
mysql> select name from Employee where Gender="F" and DeptID=(select Dep_ID from Dept
where D_name="TESTING");
+------+
| name |
+------+
| Mary |
+------+
1 row in set (0.00 sec)
```

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

```
select*from Employee where DeptID in(select DeptID from Dept where D_name not in
('TESTING','CODING'));

+------+--------+---------+-------------+------+--------+------+------+---------
+------------+------------+
| ID   | DeptId | Name    | Designation | Base | Gender | HRA  | DA   | NET_SAL |
Manager_Id | Join_date  |
+------+--------+---------+-------------+------+--------+------+------+---------
+------------+------------+
| 101  |      1 | Ram     | typist      | 2000 | M      | 1000 | 1500 |    4500 |
101 | 2000-04-03 |
| 102  |      2 | Arun    | analyst     | 6000 | F      | 1000 | 4500 |   11500 |
101 | 2003-08-21 |
| 121  |      1 | Ruby    | typist      | 2010 | F      | 1000 | 1508 |    4518 |
101 | 2000-12-20 |
| 156  |      3 | Mary    | Manager     | 4500 | F      | 1000 | 3375 |    8875 |
101 | 2000-10-24 |
| 123  |      2 | Mridula | analyst     | 6000 | F      | 1000 | 4500 |   11500 |
102 | 2007-02-14 |
| 114  |      4 | Menon   | clerk       | 1500 | M      | 1000 | 1125 |    3625 |
102 | 2005-05-11 |
| 115  |      4 | Tim     | clerk       | 1500 | M      | 1000 | 1125 |    3625 |
102 | 2003-09-11 |
| 127  |      2 | Kiran   | Manager     | 4000 | M      | 1000 | 3000 |    8000 |
121 | 2002-09-21 |
+------+--------+---------+-------------+------+--------+------+------+---------
+------------+------------+
8 rows in set (0.00 sec)
```

42. Display the names of department giving maximum salary.

```
mysql>
mysql> select D_name from Dept where Dep_ID in(select DeptId from  Employee where
Base=(select Max(Base) from Employee));
+--------+
| D_name |
+--------+
| CODING |
+--------+
1 row in set (0.00 sec)
```

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

```
mysql>
mysql> select D_name from Dept where Dep_ID in(select DeptId from  Employee where
DeptId=(select min(DeptId) from Employee));

+--------+
| D_name |
+--------+
| DESIGN |
+--------+
1 row in set (0.00 sec)
```

44. Display the second maximum salary.

```
mysql>
mysql> select min(Base) from Employee;
+-----------+
| min(Base) |
+-----------+
|      1500 |
+-----------+
1 row in set (0.00 sec)
```

45. Display the second minimum salary.

```
mysql>
mysql> select min(Base) from Employee where Base<(1500);

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

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

```
mysql>
mysql> select Name from Employee where(select avg(Base)from Employee);
+---------+
| Name    |
+---------+
| Ram     |
| Arun    |
| Ruby    |
| Mary    |
| Mridula |
| Menon   |
| Tim     |
| Kiran   |
+---------+
8 rows in set (0.00 sec)
```

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

```
mysql>
mysql> select Name from Employee where Manager_Id=(select ID from Employee where
Name='Ram');
+------+
| Name |
+------+
| Ram  |
| Arun |
| Ruby |
| Mary |
+------+
4 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>
mysql> SELECT DeptId, COUNT(*) AS 'Number_of_Dept_Employees'
-> FROM Employee
-> GROUP BY DeptId HAVING COUNT(*) > 3;
Empty 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(Base) AS 'Lowest Salary'
-> FROM Employee GROUP BY DeptId HAVING MIN(Base) > 2500;
+--------+---------------+
| DeptId | Lowest Salary |
+--------+---------------+
|      2 |          4000 |
|      3 |          4500 |
+--------+---------------+
2 rows in set (0.00 sec)
```

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

```
mysql> SELECT e.Name, e.DeptId, e.NET_SAL AS 'Maximum_Salary' FROM Employee e JOIN (SELECT
DeptId, MAX(NET_SAL) AS MaxSalary FROM Employee GROUP BY DeptId) emax ON e.DeptId =
emax.DeptId AND e.NET_SAL = emax.MaxSalary;
+---------+--------+----------------+
| Name    | DeptId | Maximum_Salary |
+---------+--------+----------------+
| Arun    |      2 |          11500 |
| Ruby    |      1 |           4518 |
| Mary    |      3 |           8875 |
| Mridula |      2 |          11500 |
| Menon   |      4 |           3625 |
| Tim     |      4 |           3625 |
+---------+--------+----------------+
6 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>
mysql> SELECT e1.Name, e1.NET_SAL AS 'Salary', e1.DeptId FROM Employee e1 JOIN Employee e2
ON e1.DeptId = e2.DeptId AND e1.NET_SAL > e2.NET_SAL ORDER BY e1.DeptId, e1.NET_SAL DESC;

+---------+--------+--------+
| Name    | Salary | DeptId |
+---------+--------+--------+
| Ruby    |   4518 |      1 |
| Arun    |  11500 |      2 |
| Mridula |  11500 |      2 |
+---------+--------+--------+
3 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>
mysql> SELECT e1.Name, e1.NET_SAL AS 'Salary', e1.DeptId FROM Employee e1 WHERE EXISTS
(SELECT 1 FROM Employee e2 WHERE e1.DeptId = e2.DeptId AND (e1.NET_SAL > e2.NET_SAL OR
e1.NET_SAL < e2.NET_SAL) ) ORDER BY e1.DeptId, e1.NET_SAL DESC;
+---------+--------+--------+
| Name    | Salary | DeptId |
+---------+--------+--------+
| Ruby    |   4518 |      1 |
| Ram     |   4500 |      1 |
| Arun    |  11500 |      2 |
| Mridula |  11500 |      2 |
| Kiran   |   8000 |      2 |
+---------+--------+--------+
5 rows in set (0.00 sec)
```

53. Add a column city for employee table.

```
mysql>
mysql> ALTER TABLE Employee ADD COLUMN City VARCHAR(255) DEFAULT NULL;
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

54. Add a column city for department.

```
mysql>
mysql> ALTER TABLE Dept ADD COLUMN City VARCHAR(255) DEFAULT NULL;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

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

```
mysql>
mysql> SELECT e.Name FROM Employee e JOIN Dept d ON e.DeptId = d.Dep_ID AND e.City =
d.City;
+------+
| Name |
+------+
| Arun |
| Mary |
| Tim  |
+------+
3 rows in set (0.01 sec)
```

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

```
mysql>
mysql> SELECT d.D_name, SUM(e.NET_SAL) AS Total_Salary FROM Dept d JOIN Employee e ON
d.Dep_ID = e.DeptId GROUP BY d.Dep_ID, d.D_name ORDER BY Total_Salary ASC LIMIT 1;
+----------+--------------+
| D_name   | Total_Salary |
+----------+--------------+
| RESEARCH |         7250 |
+----------+--------------+
1 row in set (0.00 sec)
```

57. Display the names of employees joined during 1990s

```
mysql>
mysql> SELECT Name, Join_date FROM Employee WHERE YEAR(Join_date) BETWEEN 1990 AND 1999;
Empty set (0.00 sec)
```

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

```
mysql>
mysql> SELECT Name, Join_date FROM Employee WHERE MONTH(Join_date) = 8;
+------+------------+
| Name | Join_date  |
+------+------------+
| Arun | 2003-08-21 |
+------+------------+
1 row 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>
mysql> SELECT * FROM Dept D WHERE NOT EXISTS ( SELECT 1 FROM Employee E WHERE E.DeptId =
D.Dep_ID);
Empty set (0.00 sec)
```

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

```
mysql>
mysql> SELECT D.* FROM Dept D JOIN Employee E ON D.Dep_ID = E.DeptId GROUP BY D.Dep_ID
HAVING COUNT(E.ID) > 2;
+--------+--------+-------+
| Dep_ID | D_name | City  |
+--------+--------+-------+
|      2 | CODING | Delhi |
+--------+--------+-------+
1 row in set (0.00 sec)
```

61. For each department that has more than 4 employees, retrieve the department id and number of employees who are getting salary more than 5000.

```
mysql>
mysql> SELECT E.DeptId, COUNT(*) AS Num_Employees FROM Employee E WHERE E.NET_SAL > 5000
AND E.DeptId IN (SELECT DeptId FROM Employee GROUP BY DeptId HAVING COUNT(*) > 4) GROUP BY
E.DeptId;
Empty set (0.00 sec)
```

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

```
mysql> INSERT INTO Employee (ID, DeptId, Name, Designation, Base, Gender, HRA, DA,
NET_SAL, Manager_Id, Join_date) VALUES ('189', NULL, 'John', 'Engineer', 6000, 'M', 1000,
4500, 10500, 101, '2023-01-15'),('199', NULL, 'Jane', 'Analyst', 7000, 'F', 1200, 5000,
13200, 102, '2023-02-20');
Query OK, 2 rows affected (0.05 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

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 E.Name, COALESCE(E.DeptId, 'null') AS DeptId FROM Employee E;
+---------+--------+
| Name    | DeptId |
+---------+--------+
| Ram     | 1      |
| Arun    | 2      |
| Ruby    | 1      |
| Mary    | 3      |
| Mridula | 2      |
| Menon   | 4      |
| Tim     | 4      |
| Kiran   | 2      |
| John    | null   |
| Jane    | null   |
+---------+--------+
10 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>
mysql> SELECT E.Name, COALESCE(E.DeptId, 0) AS DeptId FROM Employee E;

+---------+--------+
| Name    | DeptId |
+---------+--------+
| Ram     |      1 |
| Arun    |      2 |
| Ruby    |      1 |
| Mary    |      3 |
| Mridula |      2 |
| Menon   |      4 |
| Tim     |      4 |
| Kiran   |      2 |
| John    |      0 |
| Jane    |      0 |
+---------+--------+
10 rows in set (0.01 sec)
```

Experiment no:-2

SQL SYLABUS EXERCISE

Design a normalised database schema for the following requirement.
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

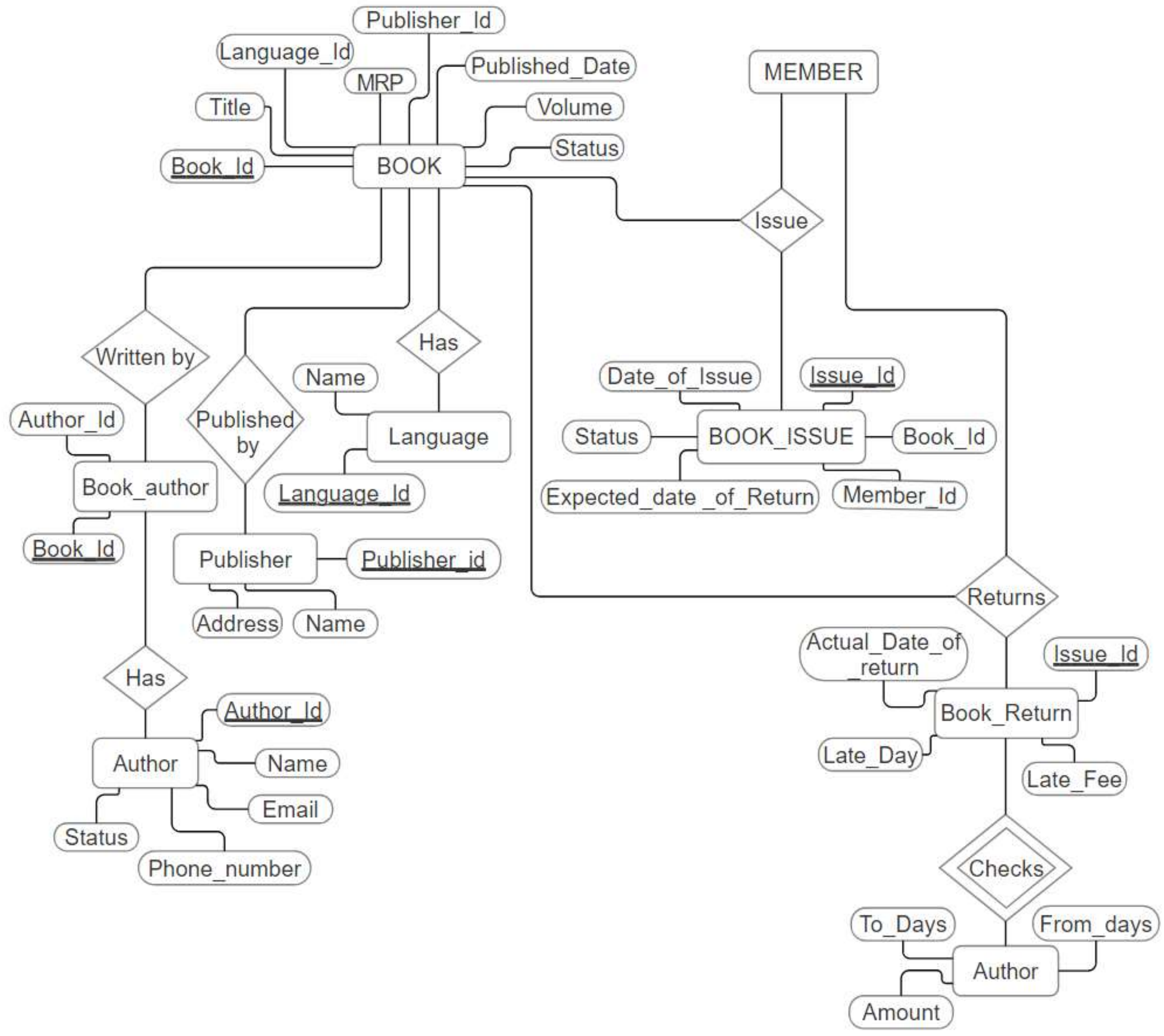
1. Create a normalized database design with proper tables, columns, column types, and constraints
2. Create an ER diagram for the above database design.
3. Write SQL commands to:
a. Create DDL statements and create the tables and constraints (from the design)
b. Create and execute DROP TABLE command in tables with and without FOREIGN KEY constraints.
c. Create and execute ALTER TABLE command in tables with data and without data.
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
b. Get the list of publishers and the number of books published by each publisher
c. Get the list of books that are issued but not returned
d. Get the list of students who reads only ‘Malayalam’ books
e. Get the total fine collected for the current month and current quarter
f. Get the list of students who have overdue (not returned the books even on due date)
g. Calculate the fine (as of today) to be collected from each overdue book.
h. Members who joined after Jan 1 2021 but has not taken any books

1. Create a normalized 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 a ER Diagram for the above database design



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

```
mysql>
mysql> CREATE TABLE LANGUAGE (Language_Id INT PRIMARY KEY,Name VARCHAR(255) NOT NULL);
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> CREATE TABLE PUBLISHER (Publisher_Id INT PRIMARY KEY,Name VARCHAR(255) NOT
NULL,Address VARCHAR(255) NOT NULL);
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql> CREATE TABLE AUTHOR (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 (0.03 sec)

mysql>
mysql> CREATE TABLE BOOK (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(Language_Id),FOREIGN KEY
(Publisher_Id) REFERENCES PUBLISHER(Publisher_Id));
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> CREATE TABLE BOOK_AUTHOR (Book_Id INT, Author_Id INT,PRIMARY KEY (Book_Id,
Author_Id),FOREIGN KEY (Book_Id) REFERENCES BOOK(Book_Id),FOREIGN KEY (Author_Id)
REFERENCES AUTHOR(Author_Id));
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> CREATE TABLE MEMBER (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 (0.03 sec)

mysql> CREATE TABLE BOOK_ISSUE (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(Book_Id),FOREIGN KEY (Member_Id) REFERENCES MEMBER(Member_Id));
Query OK, 0 rows affected (0.08 sec)

mysql> CREATE TABLE BOOK_RETURN (Issue_Id INT PRIMARY KEY,Actual_Date_Of_Return DATE,
LateDays INT,LateFee DECIMAL(10, 2),FOREIGN KEY (Issue_Id) REFERENCES
BOOK_ISSUE(Issue_Id));
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE LATE_FEE_RULE (FromDays INT,ToDays INT,Amount DECIMAL(10, 2),PRIMARY
KEY (FromDays, ToDays));
Query OK, 0 rows affected (0.04 sec)

mysql> show Tables;

+---------------+
| Tables_in_ak  |
+---------------+
| AUTHOR        |
| BOOK          |
| BOOK_AUTHOR   |
| BOOK_ISSUE    |
| BOOK_RETURN   |
| Dept          |
| Employee      |
| LANGUAGE      |
| LATE_FEE_RULE |
| MEMBER        |
| PUBLISHER     |
+---------------+
11 rows in set (0.00 sec)
```

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

```
mysql>
mysql> DROP TABLE IF EXISTS BOOK_RETURN;
Query OK, 0 rows affected (0.05 sec)

mysql> DROP TABLE IF EXISTS BOOK_ISSUE;
Query OK, 0 rows affected (0.03 sec)

mysql> DROP TABLE IF EXISTS BOOK_AUTHOR;
Query OK, 0 rows affected (0.03 sec)

mysql> DROP TABLE IF EXISTS AUTHOR;
Query OK, 0 rows affected (0.03 sec)

mysql> DROP TABLE IF EXISTS BOOK;
Query OK, 0 rows affected (0.05 sec)

mysql> DROP TABLE IF EXISTS LANGUAGE;
Query OK, 0 rows affected (0.02 sec)

mysql> DROP TABLE IF EXISTS PUBLISHER;
Query OK, 0 rows affected (0.02 sec)

mysql> DROP TABLE IF EXISTS MEMBER;
Query OK, 0 rows affected (0.03 sec)

mysql> DROP TABLE IF EXISTS LATE_FEE_RULE;
Query OK, 0 rows affected (0.02 sec)
```

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

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

mysql>
mysql> ALTER TABLE MEMBER ADD COLUMN Remark 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>
mysql> SELECT COUNT(*) AS NumberOfBooks FROM BOOK_AUTHOR ba JOIN BOOK b ON ba.Book_Id =
b.Book_Id WHERE ba.Author_Id = 1;
+---------------+
| NumberOfBooks |
+---------------+
|             1 |
+---------------+
1 row in set (0.01 sec)
```

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

```
mysql>
mysql> SELECT p.Publisher_Id, p.Name AS Publisher_Name, COUNT(b.Book_Id) AS
Number_of_Books_Published FROM PUBLISHER p LEFT JOIN BOOK b ON p.Publisher_Id =
b.Publisher_Id GROUP BY p.Publisher_Id, p.Name ORDER BY Number_of_Books_Published DESC;
```

```
+--------------+----------------+--------------------------+
| Publisher_Id | Publisher_Name | Number_of_Books_Published |
+--------------+----------------+--------------------------+
|            1 | Publisher A    |                        1 |
|            2 | Publisher B    |                        1 |
|            3 | Publisher C    |                        1 |
|            4 | Publisher D    |                        1 |
|            5 | Publisher E    |                        1 |
+--------------+----------------+--------------------------+
5 rows in set (0.00 sec)
```

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

```
mysql>
mysql> SELECT bi.Issue_Id,bi.Date_Of_Issue,bi.Book_Id, b.Title AS Book_Title,
bi.Member_Id,m.Name AS Member_Name,bi.Expected_Date_Of_Return FROM BOOK_ISSUE bi JOIN BOOK
b ON bi.Book_Id = b.Book_Id JOIN MEMBER m ON bi.Member_Id = m.Member_Id LEFT JOIN
BOOK_RETURN br ON bi.Issue_Id = br.Issue_Id WHERE br.Issue_Id IS NULL;
Empty set (0.00 sec)
```

d. Get the list of students who reads only 'Malayalam' books

```
mysql> select m.name from member66 m join book_issue bi on m.mem_id=bi.mem_id join book66
b on bi.book_id=b.book_id where b.lang_id=(select lang_id from language where
name='malayalam');

Empty set (0.01 sec)
```

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

```
mysql>
mysql> SELECT SUM(LateFee) AS TotalFineCurrentMonth FROM BOOK_RETURN WHERE
MONTH(Actual_Date_Of_Return) = MONTH(CURRENT_DATE()) AND YEAR(Actual_Date_Of_Return) =
YEAR(CURRENT_DATE());
+-----------------------+
| TotalFineCurrentMonth |
+-----------------------+
|                  NULL |
+-----------------------+
1 row in set (0.00 sec)


mysql> SELECT SUM(LateFee) AS TotalFineCurrentQuarter FROM BOOK_RETURN
WHERE(MONTH(Actual_Date_Of_Return) BETWEEN (QUARTER(CURRENT_DATE()) - 1) * 3 + 1 AND
QUARTER(CURRENT_DATE()) * 3) AND YEAR(Actual_Date_Of_Return) = YEAR(CURRENT_DATE());
+-------------------------+
| TotalFineCurrentQuarter |
+-------------------------+
|                    NULL |
+-------------------------+
1 row in set (0.00 sec)
```

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

```
mysql>
mysql> select m.name as studentname,b.title as booktitle,bi.expe_date_of_return from
member66 m join book_issue bi on m.mem_id=bi.mem_id join book66 b on bi.book_id=b.book_id
where bi.expe_date_of_return<current_date() and bi.issue_id not in(select issue_id from
book_return);
+-------------++------------------------+----------+
| studentname | Booktitle  | Expected_Date_Of_Return |
+-------------+------------+------------------------+
| Member B    | Book 1     | 2022-07-10             |
+-------------+-------------+-------------+----------+
1 row in set (0.00 sec)
```

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

```
mysql>
mysql> select b.title as booktitle,m.name as
membername,bi.expe_date_of_return,datediff(current_date(),bi.expe_date_of_return) as
daysoverdue,case when datediff(current_date(),bi.expe_date_of_return)<=7 then 10 when
datediff(current_date(),bi.expe_date_of_return)<=30 then 100 else 100+10*
(datediff(current_date(),bi.expe_date_of_return)-30) end as fineamound from book_issue bi
join book66 b on bi.book_id=b.book_id join member66 m on bi.mem_id=m.mem_id where
bi.expe_date_of_return < current_date() and bi.issue_id not in(select issue_id from
book_return);
+-------------+------------+------------------------+--------------+------------+
| Member_Name | Book_Title | Expected_Date_Of_Return | Days_Delayed | FineAmount |
+-------------+------------+------------------------+--------------+------------+
| Member B    | Book 1     | 2022-07-10             |          499 |       NULL |
+-------------+------------+------------------------+--------------+------------+
1 row in set (0.00 sec)
```

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

```
mysql>
mysql>select m.name from member66 m left join book_issue bi on m.mem_id=bi.mem_id where
bi.issue_id is null and m.date_of_join>'2021-01-01';
+--------------+
| Member_Name  |
+--------------+
| New Member A |
| New Member B |
| New Member C |
+--------------+
3 rows in set (0.01 sec)
```

EXPERIMENT NO:- 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 MAX (A INT,B INT)
    -> BEGIN
    -> IF(A>B) THEN
    -> SELECT A;
    -> ELSE
    -> SELECT B;
    -> END IF;
    -> END; //
Query OK, 0 rows affected (0.16 sec)

mysql> CALL MAX (9,3) //
+------+
|  A   |
+------+
|  9   |
+------+
1 row in set (0.00 sec)

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

2.WRITE A PL/SQL BLOCK TO READ THREE NUMBERS AND FIND THE GREATEST AMONG THEM.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GREAT (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 (0.13 sec)

mysql> CALL GREAT (5,9,1) //
+------+
| B       |
+------+
|    9    |
+------+
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 NUMBERS BETWEEN THEM.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE NUMBERS (NUM1 INT,NUM2 INT)
    -> BEGIN
    -> DECLARE C INT;
    -> SET C = NUM1;
    -> WHILE C<NUM2-1 DO
    -> SELECT C+1 AS NUMBER;
    -> SET C=C+1;
    -> END WHILE;
    -> END; //
Query OK, 0 rows affected (0.16 sec)

mysql> CALL NUMBERS (2,7) //
```

```
+--------+
| NUMBER |
+--------+
|      3 |
+--------+
1 row in set (0.00 sec)

+--------+
| NUMBER |
+--------+
|      4 |
+--------+
1 row in set (0.00 sec)

+--------+
| NUMBER |
+--------+
|      5 |
+--------+
1 row in set (0.00 sec)

+--------+
| NUMBER |
+--------+
|      6 |
+--------+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
```

4.WRITE A PL/SQL BLOCK TO READ N AND FIND THE SUM OF SERIES 1+2+3+...+N.

```
mysql> CREATE PROCEDURE SUM (NUM INT)
    -> BEGIN
    -> DECLARE C INT;
    -> DECLARE SUM INT;
    -> SET SUM = 0;
    -> SET C = 1;
    -> WHILE C<=NUM DO
    -> SET SUM = SUM +C;
    -> SET C=C+1;
    -> END WHILE;
    -> SELECT SUM AS SUM;
    -> END; //
Query OK, 0 rows affected (0.14 sec)

mysql> CALL SUM (5) //
+------+
| SUM  |
+------+
|   15    |
+------+
1 row in set (0.00 sec)

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

5.WRITE PL/SQL BLOCK TO READ A MARKS AND DISPLAY THE GRADE.

```
mysql> CREATE PROCEDURE MARK3 (MARK INT)
    -> BEGIN
    -> IF (MARK >=50) THEN
    -> SELECT ('A') AS GRADE ;
    -> ELSEIF (MARK >=40) AND (MARK<50) THEN
    -> SELECT ('B')AS GRADE;
    -> ELSEIF (MARK >=30) AND (MARK<40) THEN
    -> SELECT ('C') AS GRADE;
    -> ELSEIF (MARK >=20) AND (MARK<30) THEN
    -> SELECT ('D') AS GRADE;
    -> ELSE
    -> SELECT ('F') AS GRADE;
    -> END IF;
    -> END; //
Query OK, 0 rows affected (0.15 sec)

mysql> CALL MARK3 (45) //
+-------+
| GRADE |
+-------+
| B     |
+-------+
1 row in set (0.01 sec)
Query OK, 0 rows affected (0.01 sec)
```

6.WRITE A PL/SQL TO READ A NUMBER AND INVERT THE GIVEN NUMBER.

```
mysql> delimiter //
mysql> create procedure invert (a int)
    -> begin
    -> declare b int;
    -> set b=0;
    -> while a>0 do
    -> set b=(b*10)+mod(a,10);
    -> set a=floor(a/10);
    -> end while;
    -> select b;
    -> end;
    -> //
Query OK, 0 rows affected (0.17 sec)

mysql> call invert(102) //
+------+
| b      |
+------+
|  201  |
+------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

CREATE TABLE:
EMPLOYEE: (ID,NAME,SALARY,DEPNO,BDATE)
THEN DO FOLOOWING QUESTIONS.

7.WRITE A PL/SQL BLOCK TO READ ID OF AN EMPLOYEE AND DISPLAY HIS SALARY.

```
mysql> delimiter //
mysql> create procedure getsa(iid varchar(5))
    -> begin
    -> select basic from emp66 where iid=id;
    -> end;
    -> //
Query OK, 0 rows affected (0.15 sec)

mysql> call getsa('115') //
+---------+
| basic     |
+---------+
| 1500.00 |
+---------+
1 row in set (0.00 sec)

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

8.WRITE A PL/SQL BLOCK TO READ ID OF AN EMPLOYEE AND DISPLAY HIS NAME AND BIRTHDATE.

```
mysql> delimiter //
mysql> create procedure getdet(iid varchar(5))
    -> begin
    -> select name,dob from emp66 where iid=id;
    -> end;
    -> //
Query OK, 0 rows affected (0.18 sec)

mysql> call getdet(124) //
+------+------------+
| name | dob          |
+------+------------+
| iva  | 2001-09-26 |
+------+------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

9.WRITE A PL/SQL BLOCK TO READ ID OF AN EMPLOYEE AND DISPLAY HIS MONTH OF BIRTH.

```
mysql> delimiter //
mysql> create procedure getmon(iid varchar(5))
    -> begin
    -> select name,month(dob) as month from emp66 where iid=id;
    -> end;
    -> //
Query OK, 0 rows affected (0.16 sec)

mysql> call getmon(114) //
+-------+-------+
| name  | month|
+-------+-------+
| Menon |     6  |
+-------+-------+
1 row in set (0.02 sec)

Query OK, 0 rows affected (0.02 sec)
```

10.WRITE A PL/SQL BLOCK TO READ IDS OF TWO EMPLOYEES AND DISPLAY THE DIFFERENCE IN SALARY BETWEEN THEM.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GETDIFERENCE(IN ID1 VARCHAR(5), IN ID2 VARCHAR(5),OUT DEF DOUBLE)
    -> BEGIN
    -> DECLARE SAL,SAL1 DOUBLE;
    -> SELECT BASIC FROM emp66 WHERE ID1=ID INTO SAL;
    -> SELECT BASIC FROM emp66 WHERE ID2=ID INTO SAL1;
    -> SET DEF=SAL-SAL1;
    -> END;
    -> //
Query OK, 0 rows affected (0.13 sec)

mysql> CALL GETDIFERENCE('124','111',@P) //
Query OK, 1 row affected (0.01 sec)

mysql> SELECT @P;//
+------+
| @P   |
+------+
| 1000 |
+------+
1 row in set (0.00 sec)
```

11. CREATE A CURSOR TO DISLAY THE HIGHEST 10 SALARIES OF THE EMPLOYEE TABLE.

```
mysql> create procedure cr()
    -> begin
    -> declare i INT DEFAULT FALSE;
    -> declare e_name varchar(10);
    -> declare e_sal decimal(10,2);
    -> declare c1 cursor for select name,basic from emp66 order by basic desc limit 10;
    -> declare CONTINUE HANDLER FOR NOT FOUND SET i=1;
    -> open c1;
    -> read_loop:LOOP
    -> fetch c1 into e_name,e_sal;
    -> IF i THEN
    -> LEAVE read_loop;
    -> END IF;
    -> select e_name,e_sal;
    -> END LOOP;
    -> close c1;
    -> end;//
Query OK, 0 rows affected (0.20 sec)

mysql> call cr() //
+--------+---------+
| e_name | e_sal   |
+--------+---------+
| Arun   | 6000.00 |
+--------+---------+
1 row in set (0.07 sec)

+---------+---------+
| e_name  | e_sal   |
+---------+---------+
| Mridula | 6000.00 |
+---------+---------+
1 row in set (0.07 sec)

+--------+---------+
| e_name | e_sal   |
+--------+---------+
| Mary   | 4500.00 |
+--------+---------+
1 row in set (0.07 sec)

+--------+---------+
| e_name | e_sal   |
+--------+---------+
| Kiran  | 4000.00 |
+--------+---------+
1 row in set (0.07 sec)

+--------+---------+
| e_name | e_sal   |
+--------+---------+
| iva    | 4000.00 |
+--------+---------+
1 row in set (0.07 sec)

+--------+---------+
| e_name | e_sal   |
+--------+---------+
| jithin | 3000.00 |
+--------+---------+
1 row in set (0.07 sec)

+--------+---------+
| e_name | e_sal   |
+--------+---------+
| Ruby   | 2010.00 |
+--------+---------+
1 row in set (0.07 sec)

+--------+---------+
| e_name | e_sal   |
+--------+---------+
| Ram    | 2000.00 |
+--------+---------+
1 row in set (0.07 sec)

+--------+---------+
| e_name | e_sal   |
+--------+---------+
| Menon  | 1500.00 |
+--------+---------+
1 row in set (0.07 sec)

+--------+---------+
| e_name | e_sal   |
+--------+---------+
| Tim    | 1500.00 |
+--------+---------+
1 row in set (0.07 sec)

Query OK, 0 rows affected (0.07 sec)
```

12.CREATE A PROCEDURE TO DISPLAY WELCOME TO PL/SQL.

```
mysql> delimiter //
mysql> create procedure display (name varchar(20))
    -> begin
    -> select concat(name,"welcome to pl/sql");
    -> end;
    -> //
Query OK, 0 rows affected (0.28 sec)

mysql> call display("HEY");//
+---------------------------------+
| concat(name,"welcome to pl/sql") |
+---------------------------------+
| HEY welcome to pl/sql           |
+---------------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 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 getem(dno int)
    -> begin
    -> select id,name,basic from emp66 where deptid=dno;
    -> end;
    -> //
Query OK, 0 rows affected (0.17 sec)

mysql> call getem(4) //
+------+-------+---------+
| id   | name  | basic   |
+------+-------+---------+
| 114  | Menon | 1500.00 |
| 115  | Tim   | 1500.00 |
+------+-------+---------+
2 rows in set (0.03 sec)

Query OK, 0 rows affected (0.03 sec)
```

14.CREATE A FUNCTION TO ACCEPT THE ID OF AN EMPLOYEE AND RETURN HIS SALARY.

```
mysql> delimiter //
mysql> create function sal(iid varchar(5))
    -> returns double deterministic
    -> begin
    -> declare sal1 double;
    -> select basic from emp66 where iid=ID into sal1;
    -> return sal1;
    -> end ;
    -> //
Query OK, 0 rows affected (0.17 sec)
ss
mysql> select sal('111') //
+------------+
| sal('111') |
+------------+
|       3000 |
+------------+
1 row in set (0.00 sec)
```

15.CRETAE A TRIGGER TO MAINTAIN AN ADULT TRAIL FOR EMPLOYEE TABLE.WHEN INSERT,UPDATE,DELETE IS PERFORMED AN EMPLOYEE TABLE INSERT A ROW INTO EMP_TRAIL TABLE WITH VALUE SPECIFYING THE OPERATION AND DATE OF OPERATION.

```
mysql> delimiter //
mysql> create trigger trail_audit_insert after insert on employee for each row
    -> begin
    -> insert into emp_trail(operation,date) values ('insert',current_date());
    -> end; //
Query OK, 0 rows affected (0.17 sec)

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

mysql> delimiter //
mysql> create trigger trail_audit_delete after delete on employee for each row
    -> begin
    -> insert into emp_trail(operation,date) values ('delete',current_date());
    -> end; //
Query OK, 0 rows affected (0.21 sec)

mysql> create table emp_trail(operation varchar(20) not null,date date);
Query OK, 0 rows affected (0.54 sec)

mysql> insert into employee values(123,3,"iva","Analyst",4500,'f'); //
Query OK, 1 row affected (0.11 sec)

mysql> update employee set basic=1600 where ID=114; //
Query OK, 1 row affected (0.10 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> delete from employee where id=102; //
Query OK, 1 row affected (0.09 sec)

mysql> select * from emp_trail; //
+-----------+------------+
| operation | date       |
+-----------+------------+
| insert    | 2023-11-29 |
| update    | 2023-11-29 |
| delete    | 2023-11-29 |
+-----------+------------+
3 rows in set (0.00 sec)
```

16.CREATE A TRIGGER TO MAINTAIN AN ADULT 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(empid varchar(15),name varchar(15),sal_before
decimal(10,2),sal_after decimal(10,2),date_mod date);//
Query OK, 0 rows affected (0.48 sec)

mysql> delimiter //
mysql> create trigger emp_sal_audit
    -> after update on employee
    -> for each row
    -> begin
    -> if ! (new.basic<=>old.basic) then
    -> insert into emp_sal_trail values(old.id,old.name,old.basic,new.basic,CURRENT_DATE);
    -> end if;
    -> end;
    -> //
Query OK, 0 rows affected, 1 warning (0.16 sec)

mysql> update employee set basic=4000 where ID=123; //
Query OK, 2 rows affected (0.14 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from emp_sal_trail; //

+-------+---------+------------+-----------+------------+
| empid | name    | sal_before | sal_after | date_mod   |
+-------+---------+------------+-----------+------------+
| 123   | Mridula |    6000.00 |   4000.00 | 2023-11-29 |
| 123   | iva     |    4500.00 |   4000.00 | 2023-11-29 |
+-------+---------+------------+-----------+------------+
2 rows in set (0.00 sec)
```

17.CREATE A TRIGGER TO PREVENT SALARY MODIFICATION OF EMPLOYEE IF SALARY AFTER MODIFICATION IS LESS THAN THE SALARY BEFORE MODIFICATION.

```
mysql> delimiter //
mysql> create trigger p_sal before update on employee 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 (0.16 sec)

mysql> update employee set basic=1400 where ID =115; //
ERROR 1644 (45000): cannot decrease salary

mysql> update employee set basic=2200 where ID =121; //
Query OK, 1 row affected (0.18 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

18. CREATE A TRIGGER TO PREVENT SALARY MODIFICATION OF AN EMPLOYEE ON SUNDAY.

```
mysql> delimiter //
mysql> create trigger modify before update on employee for each row
    -> begin
    -> if(dayname(current_date())='wednesday') then
    -> signal sqlstate '45000'
    -> set message_text="not allowed";
    -> end if;
    -> end;
    -> //
Query OK, 0 rows affected (0.16 sec)

mysql> update employee set basic=1800 where id=115; //
```

```
ERROR 1644 (45000): not allowed
```

19.ASSUME A TABLE DEPARTMENT WITH COLUMNS DEPTNO AND TOTAL_SAL.TOTAL_SAL MAINTAINS THE TOTAL SALARY GIVEN BY THAT DEPARTMENT.CREATE TRIGGER ON EMPLOYEE TABLE FOR MAINTAINING TOTAL_SAL IN DEPARTMENT TABLE.

```
mysql> delimiter //
mysql> create trigger update_on_insert_total_sal
    -> after insert on employee
    -> for each row
    -> begin
    -> update dept
    -> set total_sal=total_sal+
    -> (select sum(basic) from employee where deptid=new.deptid)
    -> where deptid=new.deptid;
    -> end;
    -> //
Query OK, 0 rows affected (0.17 sec)

mysql> delimiter //
mysql> create trigger update_on_total_sal
    -> after update on employee
    -> for each row
    -> begin
    -> update dept
    -> set total_sal=total_sal+
    -> (select sum(basic) from employee where deptid=new.deptid)
    -> where deptid=new.deptid;
    -> end;
    -> //
Query OK, 0 rows affected (0.18 sec)

mysql> insert into employee values(126,1,'mallu','Typist',2000,'f'); //
Query OK, 1 row affected (0.10 sec)

mysql> update employee set basic=2200 where ID=114; //
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from dept;
    -> //
+--------+----------+-----------+
| deptid | dname    | total_sal |
+--------+----------+-----------+
|      1 | design   |  10400.00 |
|      2 | coding   |   8000.00 |
|      3 | testing  |   8500.00 |
|      4 | research |   6800.00 |
+--------+----------+-----------+
4 rows in set (0.00 sec)
```

EXPERIMENT NO:- 4

PL/SQL SYLABUS EXERCISE

1.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> delimiter //
mysql> create procedure book_returnn(in issue_id int,in actual_date date,in late_days
int,late_fee int)
    -> begin
    -> start transaction;
    -> insert into book_return(issue_id,actual_date_of_return,latedays,latefee)
    -> values(issue_id,actual_date,late_days,late_fee);
    -> update book_issue
    -> set status="returned"
    -> where issue_id=issue_id;
    -> commit;
    -> end;
    -> //
Query OK, 0 rows affected (0.25 sec)


mysql> call book_returnn(19023,'2023-11-28',41,570); //
Query OK, 0 rows affected (0.17 sec)

mysql> select * from book_return; //
+----------+-----------------------+----------+---------+
| issue_id | actual_date_of_return | latedays | latefee |
+----------+-----------------------+----------+---------+
|    19023 | 2023-09-27            |       41 |     570 |
|    19025 | 2023-09-13            |        1 |      10 |
|    19026 | 2023-09-08            |        5 |      10 |
|    19027 | 2023-11-03            |       25 |     100 |
|    19021 | 2023-10-14            |        0 |       0 |
|    19022 | 2023-10-21            |        0 |       0 |
|    19024 | 2023-04-24            |        0 |       0 |
|    19023 | 2023-11-28            |       41 |     570 |
|    19023 | 2023-11-28            |       41 |     570 |
|    19023 | 2023-11-28            |       41 |     570 |
|    19023 | 2023-11-28            |       41 |     570 |
+----------+-----------------------+----------+---------+
11 rows in set (0.00 sec)

mysql> select * from book_issue; //
+----------+---------------+---------+--------+---------------------+----------+
| issue_id | date_of_issue | book_id | mem_id | expe_date_of_return | status   |
+----------+---------------+---------+--------+---------------------+----------+
|    19021 | 2023-10-01    |     102 |     12 | 2023-10-15          | returned |
|    19022 | 2023-10-15    |     104 |     11 | 2023-10-29          | returned |
|    19023 | 2023-08-02    |     101 |     13 | 2023-08-17          | returned |
|    19024 | 2023-04-10    |     103 |     14 | 2023-04-24          | returned |
|    19025 | 2023-09-28    |     105 |     15 | 2023-09-11          | returned |
|    19026 | 2023-08-18    |     106 |     12 | 2023-09-02          | returned |
|    19027 | 2023-09-24    |     107 |     14 | 2023-10-09          | returned |
|    19028 | 2023-07-10    |     108 |     17 | 2023-07-24          | returned |
+----------+---------------+---------+--------+---------------------+----------+
8 rows in set (0.00 sec)
```

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

```
mysql> delimiter //
mysql> create view avail_book as
    -> select b.book_id,b.title,a.name as author ,p.name as publisher
    -> from book66 b
    -> inner join book_author66 ba on b.book_id=ba.book_id
    -> inner join author66 a on ba.auth_id=a.auth_id
    -> inner join publisher66 p on b.publi_id=p.publi_id
    -> where b.status="avail";
    -> //
Query OK, 0 rows affected (0.19 sec)

mysql> select * from avail_book; //
+---------+---------------+---------------+----------------+
| book_id | title         | author        | publisher      |
+---------+---------------+---------------+----------------+
|     103 | thirukural    | muthuraj      | lat books      |
|     105 | elevan        | francis pop   | lat books      |
|     101 | neermathalam  | madhavi kuuty | dc books       |
|     102 | wings of fire | dr apj        | abc publishers |
|     108 | lola          | padmarajan    | dc books       |
+---------+---------------+---------------+----------------+
5 rows in set (0.04 sec)
```

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

```
mysql> delimiter //
mysql> create procedure add_update_del_book(in action varchar(20),
    -> in book_id int,
    -> in title varchar(20),
    -> in lang_id int,
    -> in mrp decimal(10,2),
    -> in publi_id int,
    -> in publi_date date,
    -> in volume int,
    -> in status varchar(15))
    -> begin
    -> if action='add' then
    -> insert into book66(book_id,title,lang_id,mrp,publi_id,publi_date,volume,status)
    -> values(book_id,title,lang_id,mrp,publi_id,publi_date,volume,status);
    -> elseif action='update' then
    -> update book66
    -> set title=title,
    -> lang_id=lang_id,
    -> mrp=mrp,
    -> publi_id=publi_id,
    -> publi_date=publi_date,
    -> volume=volume,
    -> status=status
    -> where book_id=book_id;
    -> elseif action='delete' then
    -> delete from book66
    -> where book_id=book_id;
    -> end if;
    -> end;
    -> //
Query OK, 0 rows affected (0.18 sec)

mysql> call add_update_del_book('add',110,'the alchemist',502,450,71,'2022-02=11',
26,'avail'); //
Query OK, 1 row affected, 1 warning (0.09 sec)

mysql> select * from book66; //
+---------+----------------+---------+--------+----------+------------+--------+--------+
| book_id | title          | lang_id | mrp    | publi_id | publi_date | volume | status |
+---------+----------------+---------+--------+----------+------------+--------+--------+
|     103 | thirukural     |     505 | 589.00 |       75 | 2010-06-14 |     14 | avail  |
|     104 | geethanjali    |     503 | 125.00 |       74 | 2009-11-20 |     23 | notav  |
|     105 | elevan         |     504 | 479.00 |       75 | 2022-11-11 |      2 | avail  |
|     106 | agnichirakukal |     505 | 320.00 |       72 | 2018-10-04 |     18 | notav  |
|     107 | rom-history    |     502 | 690.00 |       72 | 2023-10-20 |      1 | notav  |
|     101 | neermathalam   |     501 | 169.00 |       71 | 2020-02-23 |      7 | avail  |
|     102 | wings of fire  |     502 | 228.00 |       73 | 2019-04-25 |      9 | avail  |
|     108 | lola           |     501 | 110.00 |       71 | 2019-10-09 |     10 | avail  |
|     110 | the alchemist  |     502 | 450.00 |       71 | 2022-02-11 |     26 | avail  |
+---------+----------------+---------+--------+----------+------------+--------+--------+
9 rows in set (0.00 sec)
```

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

```
mysql> delimiter //
mysql> create procedure prt_issue_register(in num_rows int)
    -> begin
    -> declare done int default false;
    -> declare book_id int;
    -> declare issue_date date;
    -> declare mem_id int;
    -> declare return_date date;
    -> declare cur cursor for
    -> select bi.book_id,bi.date_of_issue,bi.mem_id,br.actual_date_of_return
    -> from book_issue bi
    -> left join book_return br on bi.issue_id=br.issue_id
    -> where bi.status='issued'
    -> order by bi.date_of_issue asc;
    -> declare continue handler for not found set done=true;
    -> open cur;
    -> set @count=0;
    -> read_loop:loop
    -> fetch cur into book_id,issue_date,mem_id,return_date;
    -> if done then
    -> leave read_loop;
    -> end if;
    -> select book_id,issue_date,mem_id,return_date;
    -> set @count=@count+1;
    -> if @count=num_rows then
    -> leave read_loop;
    -> end if;
    -> end loop;
    -> close cur;
    -> end;
    -> //
Query OK, 0 rows affected (0.12 sec)

mysql> update book_issue set status='issues' where issue_id=19026;
Query OK, 1 row affected (0.10 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> call prt_issue_register(2); //
+---------+------------+--------+-------------+
| book_id | issue_date | mem_id | return_date |
+---------+------------+--------+-------------+
|     103 | 2023-04-10 |     14 | 2023-04-24  |
+---------+------------+--------+-------------+
1 row in set (0.00 sec)

+---------+------------+--------+-------------+
| book_id | issue_date | mem_id | return_date |
+---------+------------+--------+-------------+
|     106 | 2023-08-18 |     12 | 2023-09-08  |
+---------+------------+--------+-------------+
1 row in set (0.00 sec)

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

5.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 mem_history(mem_id int,name varchar(20),branch_code int,roll_no
int,ph_no int,email_id varchar(20),date_of_join date,status varchar(10)); //
Query OK, 0 rows affected (1.00 sec)

mysql> create trigger update_mem_history after update on member66
    -> for each row
    -> begin
    -> insert into
mem_history(mem_id,name,branch_code,roll_no,ph_no,email_id,date_of_join,status)
    -> values(old.mem_id,old.name,old.branch_code,old.roll_no,old.ph_no,old.email_id,
        old.date_of_join,old.status);
    -> end;
    -> //
Query OK, 0 rows affected (0.18 sec)

mysql> update member66 set name='john smith' where mem_id=12; //
Query OK, 1 row affected (0.15 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> select * from mem_history; //
+--------+------+-------------+---------+-------+------------------+--------------
+----------+
| mem_id | name | branch_code | roll_no | ph_no | email_id         | date_of_join |
status   |
+--------+------+-------------+---------+-------+------------------+--------------
+----------+
|     12 | sam  |           2 |      56 | 34567 | semiya@gmail.com | 2021-05-23   |
inactive |
+--------+------+-------------+---------+-------+------------------+--------------
+----------+
1 row in set (0.00 sec)
```

Student Record Management

Team Members

13, Anex Antony, IDK21CS020
14, Arya Ashok, IDK21CS022
15, B Vishnu, IDK21CS25
16, Basil Varghesekutty, IDK21CS024
17, Chaithanya Menon, IDK21CS026
18, Devadarsh Babu, IDK21CS027

Department of Computer Science and Engineering
Government Engineering College
Idukki

Abstract

The "Student Record Management System" is a comprehensive database management project designed to efficiently handle and organize student information within an educational institution. The system ensures secure access to the database through a unique identifier, the KTU ID, allowing authorized users to perform a range of operations including insertion, updating, deletion, and search. This project is particularly focused on enhancing user experience and data management for administrative tasks related to student records.

The system caters to essential student details, such as blood group, admission number, email address, name, and phone number, all retrievable by inputting the KTU ID. This streamlined approach facilitates quick and precise access to individual student records, simplifying administrative processes. The Student Record Management System aims to improve overall efficiency, accuracy, and accessibility in handling student information, providing a user-friendly interface for seamless interaction with the database. With its emphasis on security and functionality, this project offers a valuable solution for educational institutions seeking an effective means of managing and updating student records in a centralized database.

1 ER Diagram

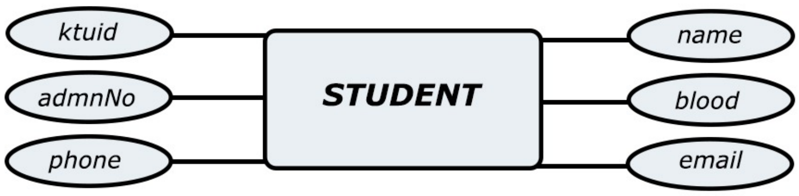


Figure 1: ER Diagram

2 Relational Schema

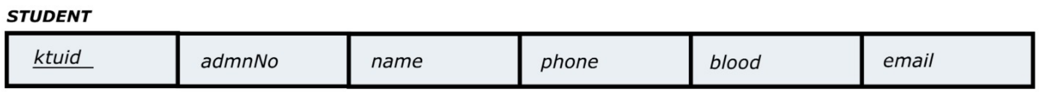
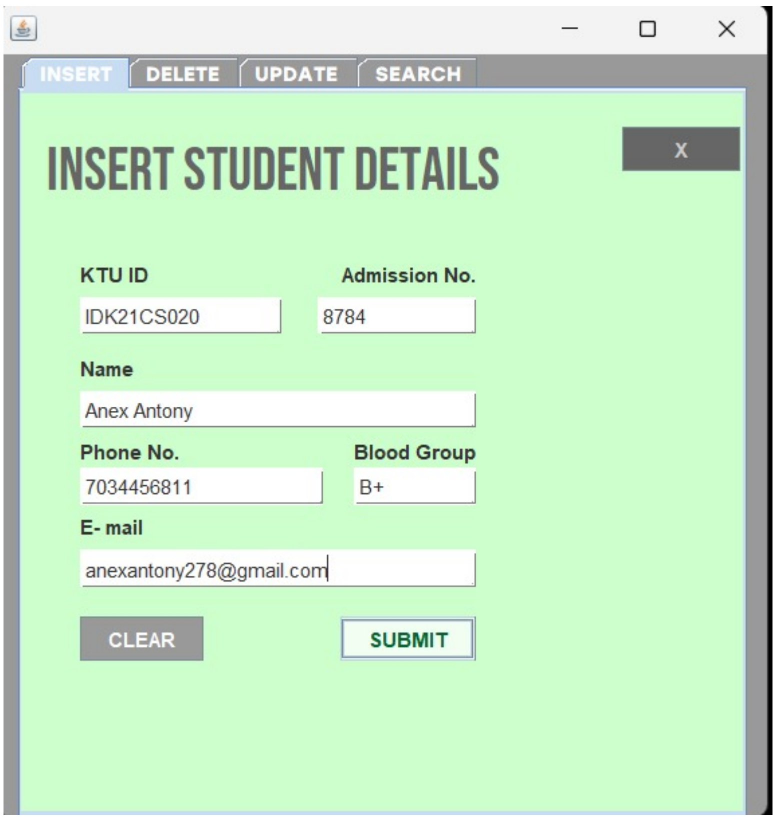


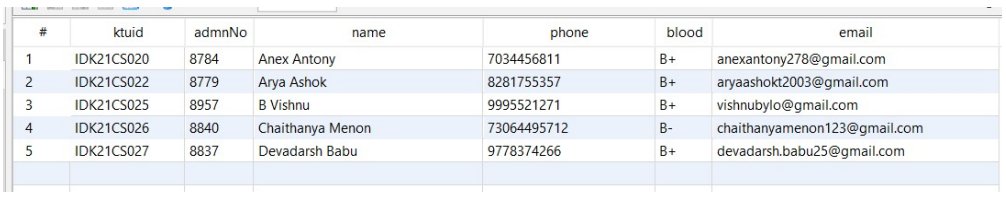
Figure 2: Relational schema

3 Screenshot

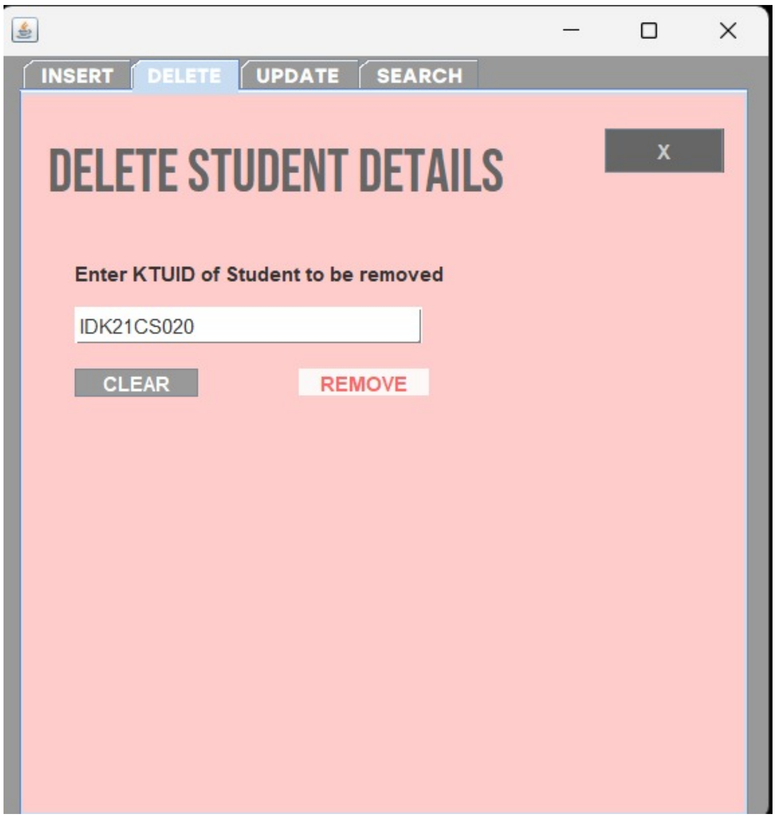


INSERT STUDENT DETAILS

KTU ID: IDK21CS020
Admission No.: 8784
Name: Anex Antony
Phone No.: 7034456811
Blood Group: B+
E- mail: anexantony278@gmail.com



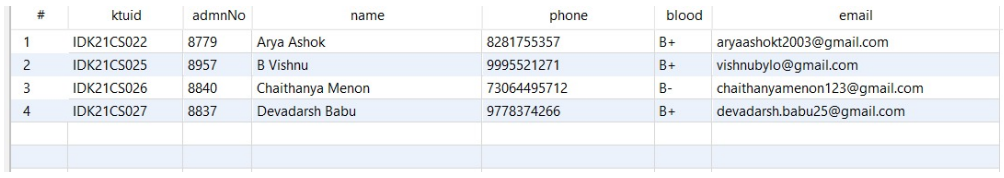
#	ktuid	admnNo	name	phone	blood	email
1	IDK21CS020	8784	Anex Antony	7034456811	B+	anexantony278@gmail.com
2	IDK21CS022	8779	Arya Ashok	8281755357	B+	aryaashokt2003@gmail.com
3	IDK21CS025	8957	B Vishnu	9995521271	B+	vishnubylo@gmail.com
4	IDK21CS026	8840	Chaithanya Menon	73064495712	B-	chaithanyamenon123@gmail.com
5	IDK21CS027	8837	Devadarsh Babu	9778374266	B+	devadarsh.babu25@gmail.com



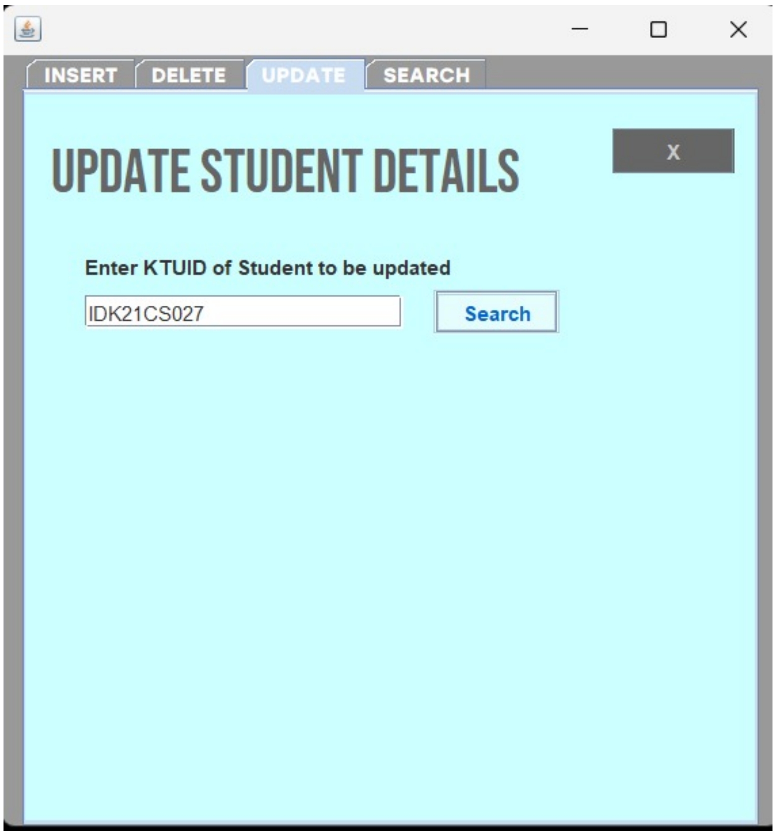
DELETE STUDENT DETAILS

Enter KTUID of Student to be removed

IDK21CS020



#	ktuid	admnNo	name	phone	blood	email
1	IDK21CS022	8779	Arya Ashok	8281755357	B+	aryaashokt2003@gmail.com
2	IDK21CS025	8957	B Vishnu	9995521271	B+	vishnubylo@gmail.com
3	IDK21CS026	8840	Chaithanya Menon	73064495712	B-	chaithanyamenon123@gmail.com
4	IDK21CS027	8837	Devadarsh Babu	9778374266	B+	devadarsh.babu25@gmail.com

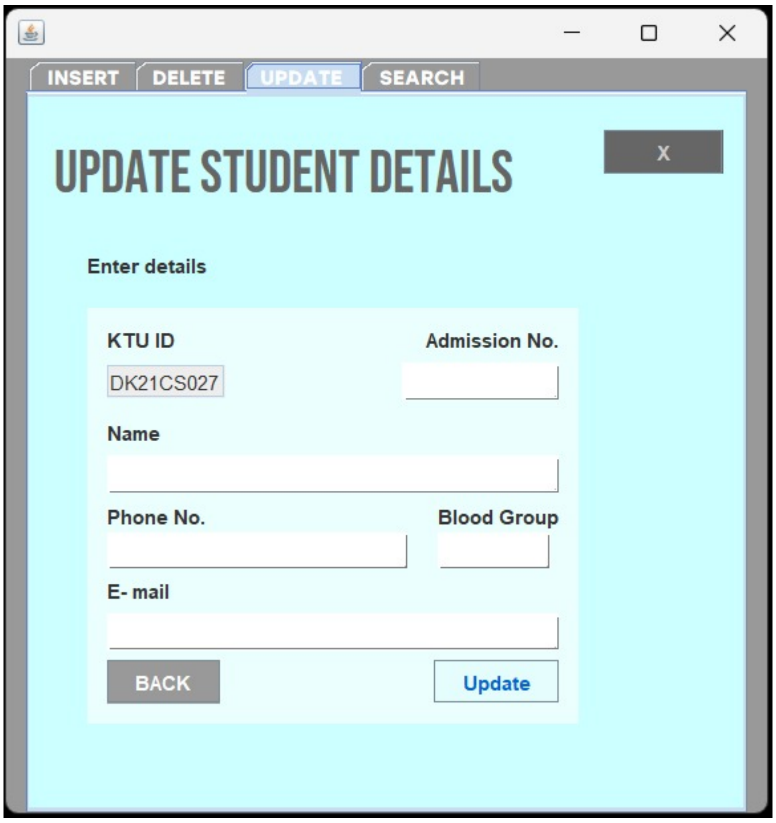


UPDATE STUDENT DETAILS

Enter KTUID of Student to be updated

IDK21CS027

Figure 3: Update



UPDATE STUDENT DETAILS

Enter details

KTU ID: DK21CS027

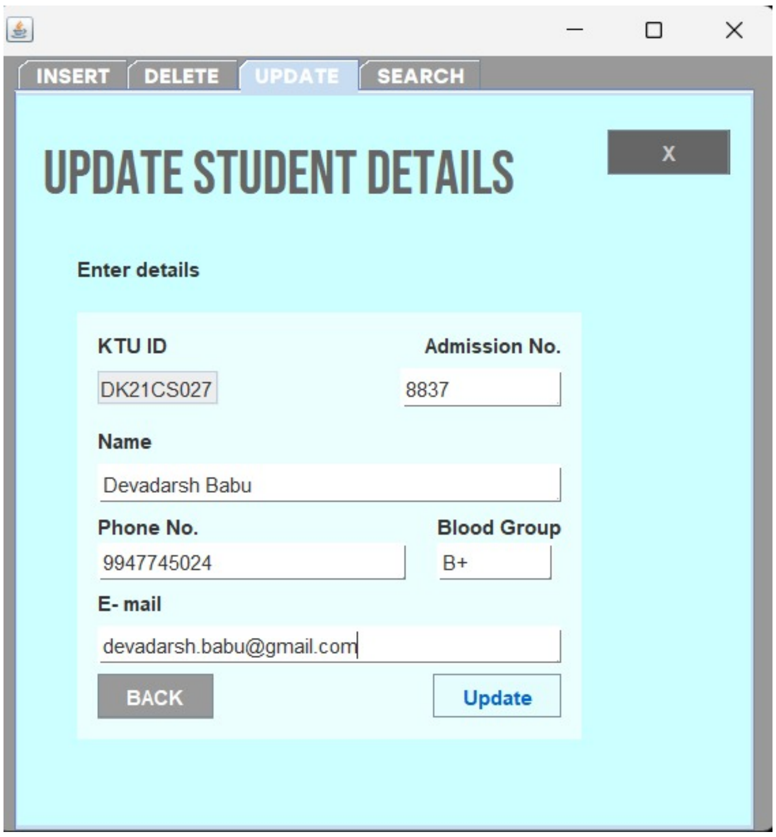
Admission No.:

Name:

Phone No.:

Blood Group:

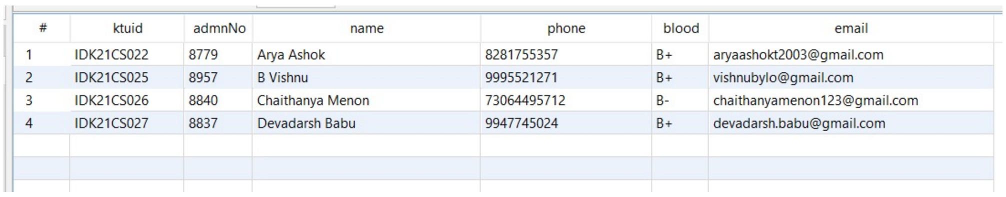
E- mail:



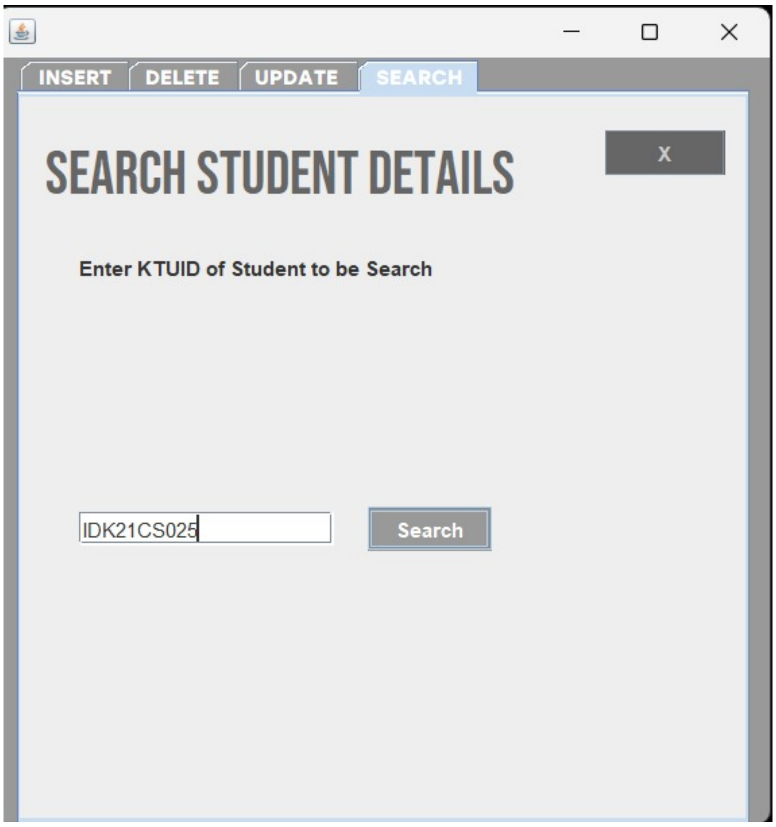
UPDATE STUDENT DETAILS

Enter details

KTU ID	Admission No.
DK21CS027	8837
Name	
Devadarsh Babu	
Phone No.	Blood Group
9947745024	B+
E- mail	
devadarsh.babu@gmail.com	



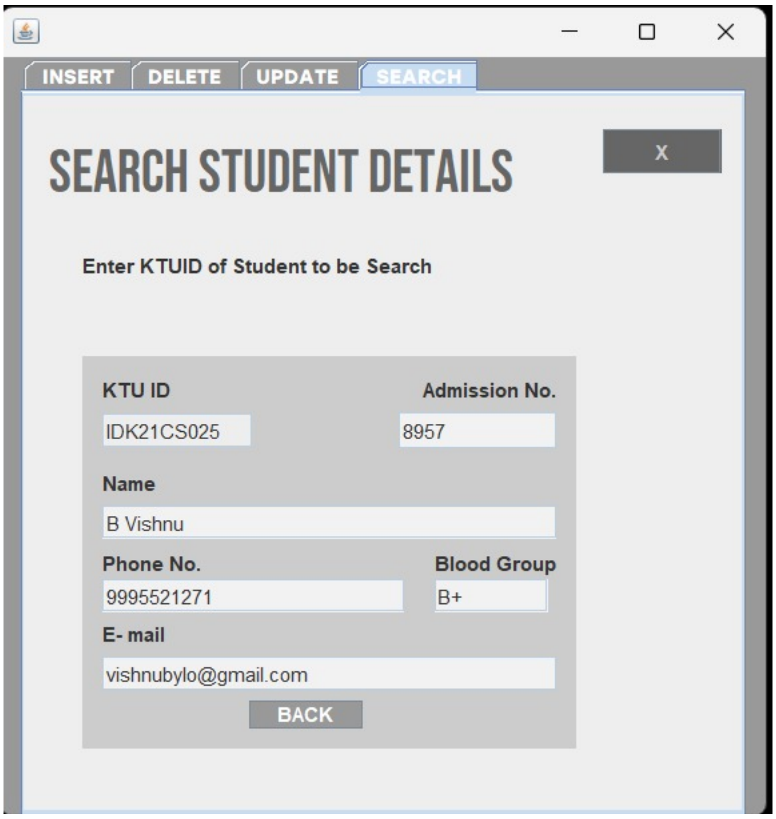
#	ktuid	admnNo	name	phone	blood	email
1	IDK21CS022	8779	Arya Ashok	8281755357	B+	aryaashokt2003@gmail.com
2	IDK21CS025	8957	B Vishnu	9995521271	B+	vishnubylo@gmail.com
3	IDK21CS026	8840	Chaithanya Menon	73064495712	B-	chaithanyamenon123@gmail.com
4	IDK21CS027	8837	Devadarsh Babu	9947745024	B+	devadarsh.babu@gmail.com



SEARCH STUDENT DETAILS

Enter KTUID of Student to be Search

IDK21CS025



SEARCH STUDENT DETAILS

Enter KTUID of Student to be Search

KTU ID	IDK21CS025
Admission No.	8957
Name	B Vishnu
Phone No.	9995521271
Blood Group	B+
E- mail	vishnubylo@gmail.com

Figure 4: Detail of Student

4 Program Code

DBConnection.java

package com.mycompany.studentrecordapp; import java.sql.*; import java.util.Properties; public class DBConnection

```
package com.mycompany.studentrecordapp;
import java.sql.*;
import java.util.Properties;

public class DBConnection {
    public static Connection connectDB() {
      Connection con;
       try {
          Properties props = new Properties();
          props.setProperty("zeroDateTimeBehavior",
          "CONVERT_TO_NULL");
           Class.forName("com.mysql.cj.jdbc.Driver");
           con = DriverManager.getConnection("jdbc:mysql:
           //localhost:3306/studentdb", "root", "root123");
           return con;
        }catch (Exception e) {
          e.printStackTrace();
          return null;
        }
    }
}
```

MainFrame.java

```
package com.mycompany.studentrecordapp;
import java.sql.*;
import com.mycompany.studentrecordapp.DBConnection.*;

public class MainFrame extends javax.swing.JFrame {


public MainFrame() {
    initComponents();
}

/**
 * This method is called from within the constructor
 to initialize the form.
 * WARNING: Do NOT modify this code. The content of
 this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed"
desc="Generated Code">//GEN-BEGIN:initComponents
private void initComponents() {

jPanel1 = new javax.swing.JPanel();
jTabbedPane1 = new javax.swing.JTabbedPane();
InsertPanel = new javax.swing.JPanel();
QuitButton = new javax.swing.JButton();
InsertLabel = new javax.swing.JLabel();
KTUIDLabel = new javax.swing.JLabel();
KTUIDTextFeild = new javax.swing.JTextField();
AdmnNoLabel = new javax.swing.JLabel();
AdmnNoTextField = new javax.swing.JTextField();
NameLabel = new javax.swing.JLabel();
NameTextField = new javax.swing.JTextField();
jLabel1 = new javax.swing.JLabel();
PhNoTextField = new javax.swing.JTextField();
BloodLabel = new javax.swing.JLabel();
BloodTextField = new javax.swing.JTextField();
MailLabel = new javax.swing.JLabel();
```

```
MailTextField = new javax.swing.JTextField();
InsertSubmitButton = new javax.swing.JButton();
ClearButton = new javax.swing.JButton();
DeletePanel = new javax.swing.JPanel();
Deletelabel = new javax.swing.JLabel();
RemoveLabel = new javax.swing.JLabel();
RemoveKTUIDTextField = new javax.swing.JTextField();
RemoveButton = new javax.swing.JButton();
Clear = new javax.swing.JButton();
QuitButton2 = new javax.swing.JButton();
UpdatePanel = new javax.swing.JPanel();
UpdateLabel = new javax.swing.JLabel();
UpdateDetailsLabel = new javax.swing.JLabel();
InputKTUIDTextField = new javax.swing.JTextField();
UpdateEnterButton = new javax.swing.JButton();
UpdateDetailsPanel = new javax.swing.JPanel();
KTUIDLabel1 = new javax.swing.JLabel();
KTUIDTextField1 = new javax.swing.JTextField();
AdmnNoLabel1 = new javax.swing.JLabel();
AdmnNoTextField1 = new javax.swing.JTextField();
NameLabel1 = new javax.swing.JLabel();
NameTextField1 = new javax.swing.JTextField();
PhNoLabel1 = new javax.swing.JLabel();
PhNoTextField1 = new javax.swing.JTextField();
BloodLabel1 = new javax.swing.JLabel();
BloodTextField1 = new javax.swing.JTextField();
MailLabel1 = new javax.swing.JLabel();
MailTextField1 = new javax.swing.JTextField();
UpdateButton = new javax.swing.JButton();
BackButton = new javax.swing.JButton();
QuitButton3 = new javax.swing.JButton();
SearchPanel = new javax.swing.JPanel();
SearchLabel = new javax.swing.JLabel();
SearchDeatailsLabel = new javax.swing.JLabel();
InputKTUIDTextField1 = new javax.swing.JTextField();
SearchEnterButton = new javax.swing.JButton();
SearchResultsPanel = new javax.swing.JPanel();
KTUIDLabel2 = new javax.swing.JLabel();
KTUIDTextField2 = new javax.swing.JTextField();
AdmnNoLabel2 = new javax.swing.JLabel();
AdmnNoTextField2 = new javax.swing.JTextField();
```

```
NameLabel2 = new javax.swing.JLabel();
NameTextField2 = new javax.swing.JTextField();
PhNoLabel2 = new javax.swing.JLabel();
PhNoTextField2 = new javax.swing.JTextField();
BloodLabel2 = new javax.swing.JLabel();
BloodTextField2 = new javax.swing.JTextField();
MailLabel2 = new javax.swing.JLabel();
MailTextField2 = new javax.swing.JTextField();
jButton2 = new javax.swing.JButton();
QuitButton1 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.
WindowConstants.EXIT_ON_CLOSE);
```

StartFrame.java

```
private void initComponents() {

jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.
WindowConstants.EXIT_ON_CLOSE);
setBackground(new java.awt.Color(0, 0, 0));
setPreferredSize(new java.awt.Dimension(624, 474));

jButton1.setBackground(new java.awt.Color(204, 255, 204));
jButton1.setText("START");
jButton1.addActionListener(new java.awt.
event.ActionListener() {
    public void actionPerformed(java.awt.
    event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setBackground(new java.awt.Color(254, 228, 228));
jButton2.setText("QUIT");
jButton2.addActionListener(new java.awt.
event.ActionListener() {
    public void actionPerformed(java.awt.
    event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentP
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addGroup(layout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(139, 139, 139)
        .addComponent(jButton1, javax.swing.
        GroupLayout.PREFERRED_SIZE, 96, javax.swing.GroupLayout.PREFERRE
    .addGroup(layout.createSequentialGroup()
        .addGap(152, 152, 152)
        .addComponent(jButton2)))
.addContainerGap(165, Short.MAX_VALUE))
);
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
.addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, layout.createSequentialGroup()
.addContainerGap(255, Short.MAX_VALUE)
.addComponent(jButton1, javax.swing.
GroupLayout.PREFERRED_SIZE, 43, javax.swing.
GroupLayout.PREFERRED_SIZE)
.addGap(18, 18, 18)
.addComponent(jButton2)
.addGap(111, 111, 111))
);
```

StudentRecord.java

```
package com.mycompany.studentrecordapp;

public class StudentRecordApp {

    public static void main(String[] args) {
        System.out.println("App Running...");
        StartFrame s=new StartFrame();
        s.setVisible(true);
        s.setBounds(0, 0, 400, 450);
    }
}
```