**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 |

krysh@AnandhaKrishnan:~$ 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 ak;

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)

mysql> INSERT INTO Employee (ID, DeptID, Name, Design, Basic, Gender)

-> VALUES

-> (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');

Query OK, 8 rows affected (0.04 sec)

Records: 8 Duplicates: 0 Warnings: 0

**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> 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> 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> SELECT

-> ID,

-> Name,

-> Design AS Desig,

-> DeptID,

-> Basic,

-> (Basic \* 0.075) AS DA, -- DA is 7.5% of Basic

-> (Basic \* 0.05) AS HRA, -- HRA is 5% of Basic (adjust as needed)

-> (Basic + (Basic \* 0.075) + (Basic \* 0.05)) AS 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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>INSERT INTO `Dept` VALUES (1,'DESIGN'),(2,'CODING'),(3,'TESTING'),(4,'RESEARCH');

**35. Display the employee name and department name.**

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> 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> 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> 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> 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> 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> select min(Base) from Employee;

+-----------+

| min(Base) |

+-----------+

| 1500 |

+-----------+

1 row in set (0.00 sec)

**45. Display the second minimum salary.**

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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> 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> SELECT D.\*

-> 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> 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> 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> 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)

**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)

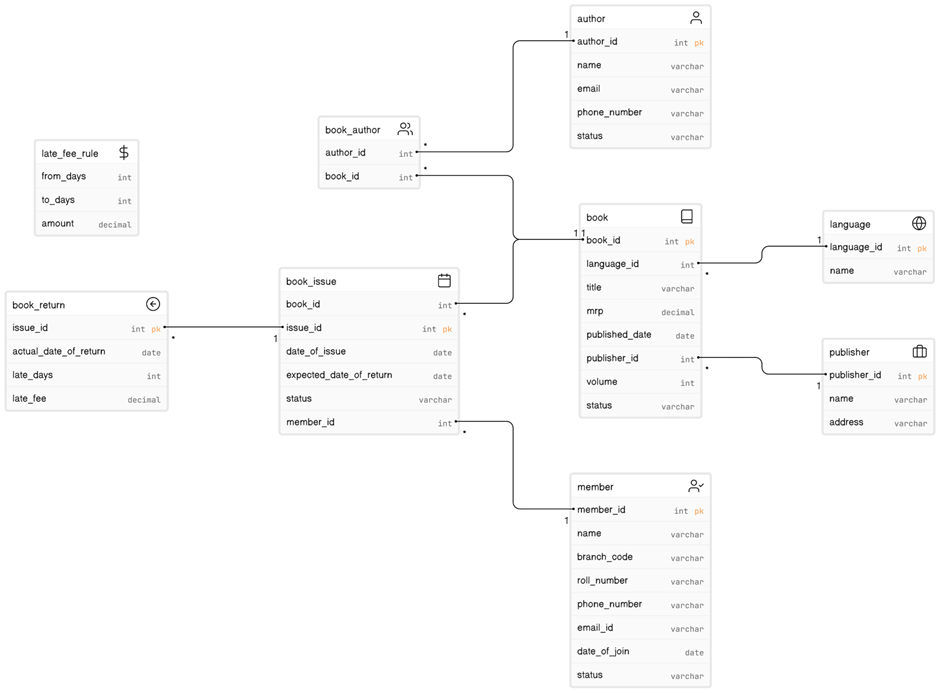
LATE\_FEE\_RULE Table:

FromDays (Part of Composite Primary Key, int)

ToDays (Part of Composite Primary Key, int)

Amount (decimal)

**2. Create 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)**

mysql> -- Create LANGUAGE Table

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 PUBLISHER Table

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 AUTHOR Table

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 BOOK Table

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 BOOK\_AUTHOR Table

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 MEMBER Table

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>

mysql> -- Create BOOK\_ISSUE Table

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>

mysql> -- Create BOOK\_RETURN Table

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>

mysql> -- Create LATE\_FEE\_RULE Table

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> -- Drop tables with and without FOREIGN KEY constraints

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> -- Alter table without data

mysql> ALTER TABLE MEMBER

-> ADD COLUMN Remarks VARCHAR(255);

Query OK, 0 rows affected (0.04 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> -- Alter table with data

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> 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> 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> 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.Member\_Id,

-> m.Name AS Member\_Name,

-> m.Branch\_Code,

-> m.Roll\_Number,

-> m.Phone\_Number,

-> m.Email\_Id,

-> m.Date\_of\_Join,

-> m.Status

-> FROM

-> MEMBER m

-> JOIN

-> BOOK\_ISSUE bi ON m.Member\_Id = bi.Member\_Id

-> JOIN

-> BOOK b ON bi.Book\_Id = b.Book\_Id

-> LEFT JOIN

-> BOOK\_RETURN br ON bi.Issue\_Id = br.Issue\_Id

-> WHERE

-> b.Language\_Id = 6

-> AND (

-> b.Book\_Id IS NULL -- No book issued (optional, depends on your logic)

-> OR br.Issue\_Id IS NULL -- Book issued but not returned

-> );

Empty set (0.01 sec)

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

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> -- Get the total fine collected for the current quarter

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> SELECT

-> m.Member\_Id,

-> m.Name AS Member\_Name,

-> m.Branch\_Code,

-> m.Roll\_Number,

-> m.Phone\_Number,

-> m.Email\_Id,

-> m.Date\_of\_Join,

-> m.Status,

-> bi.Issue\_Id,

-> bi.Date\_Of\_Issue,

-> b.Title AS Book\_Title,

-> bi.Expected\_Date\_Of\_Return

-> FROM

-> MEMBER m

-> JOIN

-> BOOK\_ISSUE bi ON m.Member\_Id = bi.Member\_Id

-> JOIN

-> BOOK b ON bi.Book\_Id = b.Book\_Id

-> LEFT JOIN

-> BOOK\_RETURN br ON bi.Issue\_Id = br.Issue\_Id

-> WHERE

-> br.Issue\_Id IS NULL -- Book issued but not returned

-> AND bi.Expected\_Date\_Of\_Return < CURRENT\_DATE(); -- Overdue books

+-----------+-------------+-------------+-------------+--------------+-------------------+--------------+----------+----------+---------------+------------+-------------------------+

| Member\_Id | Member\_Name | Branch\_Code | Roll\_Number | Phone\_Number | Email\_Id | Date\_of\_Join | Status | Issue\_Id | Date\_Of\_Issue | Book\_Title | Expected\_Date\_Of\_Return |

+-----------+-------------+-------------+-------------+--------------+-------------------+--------------+----------+----------+---------------+------------+-------------------------+

| 2 | Member B | B002 | R002 | 444-555-6666 | memberB@email.com | 2022-02-01 | Inactive | 7 | 2022-07-01 | 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> SELECT

-> bi.Issue\_Id,

-> m.Member\_Id,

-> m.Name AS Member\_Name,

-> b.Title AS Book\_Title,

-> bi.Expected\_Date\_Of\_Return,

-> br.Actual\_Date\_Of\_Return,

-> DATEDIFF(CURRENT\_DATE(), bi.Expected\_Date\_Of\_Return) AS Days\_Delayed,

-> CASE

-> WHEN DATEDIFF(CURRENT\_DATE(), bi.Expected\_Date\_Of\_Return) BETWEEN 0 AND 7 THEN

-> (DATEDIFF(CURRENT\_DATE(), bi.Expected\_Date\_Of\_Return) \* lfr.Amount) -- Fine for 0-7 days

-> WHEN DATEDIFF(CURRENT\_DATE(), bi.Expected\_Date\_Of\_Return) BETWEEN 8 AND 30 THEN

-> lfr.Amount -- Fine for 8-30 days

-> ELSE

-> (DATEDIFF(CURRENT\_DATE(), bi.Expected\_Date\_Of\_Return) \* lfr.Amount) -- Fine for more than 30 days

-> END AS FineAmount

-> FROM

-> BOOK\_ISSUE bi

-> JOIN

-> MEMBER m ON bi.Member\_Id = m.Member\_Id

-> JOIN

-> BOOK b ON bi.Book\_Id = b.Book\_Id

-> LEFT JOIN

-> BOOK\_RETURN br ON bi.Issue\_Id = br.Issue\_Id

-> LEFT JOIN

-> LATE\_FEE\_RULE lfr ON DATEDIFF(CURRENT\_DATE(), bi.Expected\_Date\_Of\_Return) BETWEEN lfr.FromDays AND lfr.ToDays

-> WHERE

-> br.Issue\_Id IS NULL -- Book issued but not returned

-> AND bi.Expected\_Date\_Of\_Return < CURRENT\_DATE(); -- Overdue books

+----------+-----------+-------------+------------+-------------------------+-----------------------+--------------+------------+

| Issue\_Id | Member\_Id | Member\_Name | Book\_Title | Expected\_Date\_Of\_Return | Actual\_Date\_Of\_Return | Days\_Delayed | FineAmount |

+----------+-----------+-------------+------------+-------------------------+-----------------------+--------------+------------+

| 7 | 2 | Member B | Book 1 | 2022-07-10 | NULL | 499 | NULL |

+----------+-----------+-------------+------------+-------------------------+-----------------------+--------------+------------+

1 row in set (0.00 sec)

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

mysql> SELECT

-> m.Member\_Id,

-> m.Name AS Member\_Name,

-> m.Branch\_Code,

-> m.Roll\_Number,

-> m.Phone\_Number,

-> m.Email\_Id,

-> m.Date\_of\_Join,

-> m.Status

-> FROM

-> MEMBER m

-> WHERE

-> m.Date\_of\_Join > '2021-01-01'

-> AND m.Member\_Id NOT IN (SELECT DISTINCT Member\_Id FROM BOOK\_ISSUE);

+-----------+--------------+-------------+-------------+--------------+----------------------+--------------+--------+

| Member\_Id | Member\_Name | Branch\_Code | Roll\_Number | Phone\_Number | Email\_Id | Date\_of\_Join | Status |

+-----------+--------------+-------------+-------------+--------------+----------------------+--------------+--------+

| 6 | New Member A | B006 | R006 | 111-222-3333 | newmemberA@email.com | 2021-02-01 | Active |

| 7 | New Member B | B007 | R007 | 444-555-6666 | newmemberB@email.com | 2021-03-15 | Active |

| 8 | New Member C | B008 | R008 | 666-888-1111 | newmemberC@email.com | 2021-05-01 | Active |

+-----------+--------------+-------------+-------------+--------------+----------------------+--------------+--------+

3 rows in set (0.01 sec)

**PL/SQL PRACTISE QUESTIONS**

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

mysql> delimiter //

mysql> --ANANDHA KRISHNAN

-> create procedure greater(a int, b int)

-> BEGIN

-> if (a > b) then

-> select a;

-> else

-> select b;

-> end if;

-> END;

-> //

mysql> call greater(3,4); //

+------+

| b |

+------+

| 4 |

+------+

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> --ANANDHA KRISHNAN

-> CREATE procedure three(a int, b int, c int)

-> BEGIN

-> IF (a > b) && (a > c) then

-> select a;

-> ELSEIF (b > a) && (b > c) then

-> select b;

-> ELSE

-> select c;

-> END IF;

-> END;

-> //

mysql> call three(5,8,9); //

+------+

| c |

+------+

| 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 the numbers between them.**

mysql> --ANANDHA KRISHNAN

-> CREATE PROCEDURE print(IN a INT,IN b INT)

-> BEGIN

-> DECLARE c INT;

-> SELECT CONCAT('Numbers between ', a, ' and ', b, ':') AS message;

-> SET c = a;

-> WHILE c < b - 1 DO

-> SELECT c + 1 AS number;

-> SET c = c + 1;

-> END WHILE;

-> END;

-> //

mysql> call print(5,8); //

+--------------------------+

| message |

+--------------------------+

| Numbers between 5 and 8: |

+--------------------------+

1 row in set (0.00 sec)

+--------+

| number |

+--------+

| 6 |

+--------+

1 row in set (0.00 sec)

+--------+

| number |

+--------+

| 7 |

+--------+

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 the series 1+2+3 +... N.**

mysql> --ANANDHA KRISHNAN

-> create procedure four(n int)

-> begin

-> declare s int default(0);

-> declare i int default(1);

->

-> while(i<=n) do

-> set s := s+i;

-> set i := i+1;

-> end while;

-> select s;

-> end;

-> //

mysql> call four(5); //

+------+

| s |

+------+

| 15 |

+------+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

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

mysql> --ANANDHA KRISHNAN

-> CREATE PROCEDURE grade(n int)

-> begin

-> if (n>95) && (n<=100) then

-> select 'S' as grade;

-> elseif (n>90) && (n<=100) then

-> select 'A+' as grade;

-> elseif (n>85) && (n<=100) then

-> select 'A' as grade;

-> elseif (n>80) && (n<=100) then

-> select 'B+' as grade;

-> elseif (n>75) && (n<=100) then

-> select 'B' as grade;

-> elseif (n>70) && (n<=100) then

-> select 'C+' as grade;

-> elseif (n>65) && (n<=100) then

-> select 'C' as grade;

-> elseif (n>60) && (n<=100) then

-> select 'D' as grade;

-> elseif (n>45) && (n<=100) then

-> select 'P' as grade;

-> elseif (n>100) then

-> select 'Invalid' as grade;

-> else

-> select 'F' as grade;

-> end if;

-> end;

-> //

mysql> call grade(50); //

+-------+

| grade |

+-------+

| P |

+-------+

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> --ANANDHA KRISHNAN

-> CREATE procedure reverse(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;

-> //

mysql> call reverse(896); //

+------+

| b |

+------+

| 698 |

+------+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

**7. Write a PL/SQL block to read a number and invert the given number by taking it as a**

**string.**

mysql> CREATE PROCEDURE invert\_number\_proc(

-> IN p\_input\_number INT,

-> OUT p\_output\_number INT

-> )

-> BEGIN

-> DECLARE v\_input\_string VARCHAR(100);

-> DECLARE v\_output\_string VARCHAR(100) DEFAULT '';

-> DECLARE i INT DEFAULT 0;

->

-> -- Convert the input number to a string

-> SET v\_input\_string = CAST(p\_input\_number AS CHAR);

->

-> -- Reverse the characters of the string

-> SET i = LENGTH(v\_input\_string);

-> WHILE i > 0 DO

-> SET v\_output\_string = CONCAT(v\_output\_string, SUBSTRING(v\_input\_string, i, 1));

-> SET i = i - 1;

-> END WHILE;

->

-> -- Convert the reversed string back to a number

-> SET p\_output\_number = CAST(v\_output\_string AS SIGNED);

-> END;

-> //

Query OK, 0 rows affected (0.04 sec)

mysql> SET @input\_number = 12345;

-> SET @output\_number = 0;

->

-> -- Call the procedure

-> CALL invert\_number\_proc(@input\_number, @output\_number);

->

-> -- Display the result

-> SELECT @output\_number AS inverted\_number;

-> //

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

+-----------------+

| inverted\_number |

+-----------------+

| 54321 |

+-----------------+

1 row in set (0.00 sec)

**PL/SQL SYLABUS QUESTIONS**

**Create a Table:**

**EMPLOYEE: (ID, NAME, SALARY, DEPNO, BDATE)**

**Then do the following questions**

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

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

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

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

between them.

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

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

19. 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.

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

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

22. Create a trigger to maintain an audit trail for employee table for tracking salary modifications. When salary is updates, insert into emp\_sal\_trail table a row with values of employee id, name, salary before modification, salary after modification and date of modification.

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

24. Create a trigger to prevent salary modification of an employee on Sunday.

25.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.

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

mysql> --ANANDHA KRYSHNAN

-> CREATE PROCEDURE Q (X INT)

-> BEGIN

-> SELECT NAME, SALARY

-> FROM EMPLOYEE

-> WHERE ID =X;

-> END;

-> //

mysql> call Q(5); //

+---------------+----------+

| NAME | SALARY |

+---------------+----------+

| Charlie Brown | 55000.25 |

+---------------+----------+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

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

mysql> --ANANDHA KRYSHNAN

-> CREATE PROCEDURE Y (X INT)

-> BEGIN

-> SELECT NAME, BDATE

-> FROM EMPLOYEE

-> WHERE ID =X;

-> END;

-> //

mysql> call Y(3); //

+-------------+------------+

| NAME | BDATE |

+-------------+------------+

| Bob Johnson | 1988-11-30 |

+-------------+------------+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

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

mysql> --ANANDHA KRYSHNAN

-> CREATE PROCEDURE month(x int)

-> begin

-> select month(BDATE)

-> FROM EMPLOYEE WHERE ID =X;

-> END;

-> //

mysql> call month(2); //

+--------------+

| month(BDATE) |

+--------------+

| 8 |

+--------------+

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> --ANANDHA KRYSHNAN

-> CREATE FUNCTION diff(id1 INT, id2 INT) RETURNS INT DETERMINISTIC

-> BEGIN

-> DECLARE salary1 INT;

-> DECLARE salary2 INT;

-> DECLARE salary\_difference INT;

->

-> SELECT SALARY INTO salary1 FROM EMPLOYEE WHERE ID = id1;

-> SELECT SALARY INTO salary2 FROM EMPLOYEE WHERE ID = id2;

->

-> SET salary\_difference = salary1 - salary2;

->

-> RETURN salary\_difference;

-> END;

-> //

Query OK, 0 rows affected (0.01 sec)

mysql> SELECT diff(2,1) AS salary\_difference;//

+-------------------+

| salary\_difference |

+-------------------+

| 10000 |

+-------------------+

1 row in set (0.00 sec)

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

mysql> --ANANDHA KRYSHNAN

-> CREATE PROCEDURE display\_top\_salaries()

-> BEGIN

-> DECLARE done BOOLEAN DEFAULT FALSE;

-> DECLARE v\_salary DECIMAL(10, 2);

->

-> -- Declare a cursor to fetch the highest 10 salaries

-> DECLARE cursor\_top\_salaries CURSOR FOR

-> SELECT SALARY

-> FROM EMPLOYEE

-> ORDER BY SALARY DESC

-> LIMIT 10;

->

-> -- Declare a continue handler to exit the loop

-> DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

->

-> -- Open the cursor

-> OPEN cursor\_top\_salaries;

->

-> -- Fetch and display the salaries

-> FETCH cursor\_top\_salaries INTO v\_salary;

-> WHILE NOT done DO

-> -- Display the salary (you can modify this part based on your needs)

-> SELECT v\_salary AS TopSalary;

-> -- Fetch the next row

-> FETCH cursor\_top\_salaries INTO v\_salary;

-> END WHILE;

->

-> -- Close the cursor

-> CLOSE cursor\_top\_salaries;

-> END;

-> //

Query OK, 0 rows affected (0.04 sec)

mysql> CALL display\_top\_salaries(); //

+-----------+

| TopSalary |

+-----------+

| 80000.75 |

+-----------+

1 row in set (0.00 sec)

+-----------+

| TopSalary |

+-----------+

| 75000.50 |

+-----------+

1 row in set (0.00 sec)

+-----------+

| TopSalary |

+-----------+

| 70000.00 |

+-----------+

1 row in set (0.00 sec)

+-----------+

| TopSalary |

+-----------+

| 60000.00 |

+-----------+

1 row in set (0.00 sec)

+-----------+

| TopSalary |

+-----------+

| 55000.25 |

+-----------+

1 row in set (0.00 sec)

+-----------+

| TopSalary |

+-----------+

| 50000.00 |

+-----------+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

**18. Create a procedure to display Welcome to PL/SQL.**

mysql> --ANANDHA KRYSHNAN

-> CREATE PROCEDURE display\_welcome\_message()

-> BEGIN

-> SELECT 'Welcome to PL/SQL' AS message;

-> END;

-> //

Query OK, 0 rows affected (0.03 sec)

mysql> CALL display\_welcome\_message(); //

+-------------------+

| message |

+-------------------+

| Welcome to PL/SQL |

+-------------------+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

**19. 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> --ANANDHA KRYSHNAN

-> CREATE PROCEDURE display\_employees\_in\_department(IN p\_dno INT)

-> BEGIN

-> SELECT ID, NAME, SALARY

-> FROM EMPLOYEE

-> WHERE DEPNO = p\_dno;

-> END;

-> //

Query OK, 0 rows affected (0.04 sec)

mysql> CALL display\_employees\_in\_department(101); //

+----+-------------+----------+

| ID | NAME | SALARY |

+----+-------------+----------+

| 1 | John Doe | 50000.00 |

| 3 | Bob Johnson | 75000.50 |

+----+-------------+----------+

2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

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

mysql> --ANANDHA KRYSHNAN

-> CREATE FUNCTION get\_employee\_salary(p\_employee\_id INT) RETURNS DECIMAL(10, 2) DETERMINISTIC

-> BEGIN

-> DECLARE v\_salary DECIMAL(10, 2);

-> SELECT SALARY INTO v\_salary

-> FROM EMPLOYEE

-> WHERE ID = p\_employee\_id;

-> RETURN v\_salary;

-> END;

-> //

Query OK, 0 rows affected (0.01 sec)

mysql> SELECT get\_employee\_salary(1) AS employee\_salary; //

+-----------------+

| employee\_salary |

+-----------------+

| 50000.00 |

+-----------------+

1 row in set (0.00 sec)

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

-> -- Create the EMP\_TRAIL table for audit trail

-> CREATE TABLE EMP\_TRAIL (

-> AUDIT\_ID INT AUTO\_INCREMENT PRIMARY KEY,

-> EMPLOYEE\_ID INT,

-> OPERATION VARCHAR(10),

-> OPERATION\_DATE TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

-> FOREIGN KEY (EMPLOYEE\_ID) REFERENCES EMPLOYEE(ID)

-> );

mysql> CREATE TRIGGER employee\_audit\_trigger

-> AFTER INSERT ON EMPLOYEE

-> FOR EACH ROW

-> BEGIN

-> -- Insert a row into EMP\_TRAIL for audit trail

-> INSERT INTO EMP\_TRAIL (EMPLOYEE\_ID, OPERATION) VALUES (NEW.ID, 'INSERT');

-> END //

Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TRIGGER employee\_audit\_trigger\_update

-> AFTER UPDATE ON EMPLOYEE

-> FOR EACH ROW

-> BEGIN

-> INSERT INTO EMP\_TRAIL (EMPLOYEE\_ID, OPERATION) VALUES (OLD.ID, 'UPDATE');

-> END //

Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER employee\_audit\_trigger\_delete

-> AFTER DELETE ON EMPLOYEE

-> FOR EACH ROW

-> BEGIN

-> INSERT INTO EMP\_TRAIL (EMPLOYEE\_ID, OPERATION) VALUES (OLD.ID, 'DELETE');

-> END //

Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO EMPLOYEE (ID, NAME, SALARY, DEPNO, BDATE) VALUES (9, 'Joh', 50000.00, 100, '1990-01-15');

Query OK, 1 row affected (0.02 sec)

mysql> select\*from EMP\_TRAIL;

+----------+-------------+-----------+---------------------+

| AUDIT\_ID | EMPLOYEE\_ID | OPERATION | OPERATION\_DATE |

+----------+-------------+-----------+---------------------+

| 1 | 9 | INSERT | 2023-11-21 22:04:37 |

+----------+-------------+-----------+---------------------+

1 row in set (0.00 sec)

**22. Create a trigger to maintain an audit trail for employee table for tracking salary modifications. When salary is updates, 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 (

-> TRAIL\_ID INT AUTO\_INCREMENT PRIMARY KEY,

-> EMPLOYEE\_ID INT,

-> EMPLOYEE\_NAME VARCHAR(255),

-> SALARY\_BEFORE DECIMAL(10, 2),

-> SALARY\_AFTER DECIMAL(10, 2),

-> MODIFICATION\_DATE TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

-> FOREIGN KEY (EMPLOYEE\_ID) REFERENCES EMPLOYEE(ID)

-> );

Query OK, 0 rows affected (0.09 sec)

mysql>

mysql> -- Create the trigger for salary modifications

mysql> DELIMITER //

mysql>

mysql> CREATE TRIGGER salary\_audit\_trigger

-> AFTER UPDATE ON EMPLOYEE

-> FOR EACH ROW

-> BEGIN

-> IF NEW.SALARY != OLD.SALARY THEN

-> INSERT INTO EMP\_SAL\_TRAIL (EMPLOYEE\_ID, EMPLOYEE\_NAME, SALARY\_BEFORE, SALARY\_AFTER)

-> VALUES (NEW.ID, NEW.NAME, OLD.SALARY, NEW.SALARY);

-> END IF;

-> END //

Query OK, 0 rows affected (0.01 sec)

mysql> UPDATE EMPLOYEE

-> SET SALARY = 60000.00

-> WHERE ID = 1;

Query OK, 1 row affected (0.03 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> select \*from EMP\_SAL\_TRAIL;

+----------+-------------+---------------+---------------+--------------+---------------------+

| TRAIL\_ID | EMPLOYEE\_ID | EMPLOYEE\_NAME | SALARY\_BEFORE | SALARY\_AFTER | MODIFICATION\_DATE |

+----------+-------------+---------------+---------------+--------------+---------------------+

| 1 | 1 | John Doe | 50000.00 | 60000.00 | 2023-11-21 22:08:40 |

+----------+-------------+---------------+---------------+--------------+---------------------+

1 row in set (0.00 sec)

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

mysql> CREATE TRIGGER prevent\_salary\_modification

-> BEFORE UPDATE ON EMPLOYEE

-> FOR EACH ROW

-> BEGIN

-> IF NEW.SALARY < OLD.SALARY THEN

-> SIGNAL SQLSTATE '45000'

-> SET MESSAGE\_TEXT = 'Cannot update salary. New salary is less than current salary.';

-> END IF;

-> END //

Query OK, 0 rows affected (0.05 sec)

mysql> UPDATE EMPLOYEE

-> SET SALARY = 1000.00

-> WHERE ID = 1;

ERROR 1644 (45000): Cannot update salary. New salary is less than current salary.

**24. Create a trigger to prevent salary modification of an employee on Sunday.**

mysql> CREATE TRIGGER prevent\_salary\_modification\_sunday

-> BEFORE UPDATE ON EMPLOYEE

-> FOR EACH ROW

-> BEGIN

-> IF DAYOFWEEK(NOW()) = 1 THEN

-> SIGNAL SQLSTATE '45000'

-> SET MESSAGE\_TEXT = 'Cannot update salary on Sunday.';

-> END IF;

-> END //

Query OK, 0 rows affected (0.04 sec)

mysql> UPDATE EMPLOYEE SET SALARY = 60000.00 WHERE ID = 1;

ERROR 1644 (45000): Cannot update salary on Sunday.

**25.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> CREATE TABLE Department (

-> DeptNo INT PRIMARY KEY,

-> Total\_Sal DECIMAL(10, 2) DEFAULT 0.00);

Query OK, 0 rows affected (0.17 sec)

mysql> CREATE TRIGGER update\_total\_sal\_after\_insert

-> AFTER INSERT ON Employe

-> FOR EACH ROW

-> BEGIN

-> UPDATE Department

-> SET Total\_Sal = Total\_Sal + NEW.SALARY

-> WHERE DeptNo = NEW.DEPTNO;

-> END //

Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TRIGGER update\_total\_sal\_after\_update

-> AFTER UPDATE ON Employe

-> FOR EACH ROW

-> BEGIN

-> UPDATE Department

-> SET Total\_Sal = Total\_Sal - OLD.SALARY + NEW.SALARY

-> WHERE DeptNo = OLD.DEPTNO;

-> END //

Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TRIGGER update\_total\_sal\_after\_delete

-> AFTER DELETE ON Employe

-> FOR EACH ROW

-> BEGIN

-> UPDATE Department

-> SET Total\_Sal = Total\_Sal - OLD.SALARY

-> WHERE DeptNo = OLD.DEPTNO;

-> END //

Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO Department (DeptNo) VALUES (1);

ERROR 1062 (23000): Duplicate entry '1' for key 'Department.PRIMARY'

mysql> INSERT INTO Employe (ID, NAME, SALARY, DEPTNO, BDATE) VALUES (1, 'John Doe', 50000.00, 1, '1990-01-15');

ERROR 1062 (23000): Duplicate entry '1' for key 'Employe.PRIMARY'

mysql>

mysql> -- Verify the initial state of the tables

mysql> SELECT \* FROM Department;

+--------+-----------+

| DeptNo | Total\_Sal |

+--------+-----------+

| 1 | 0.00 |

+--------+-----------+

1 row in set (0.00 sec)

mysql> SELECT \* FROM Employe;

+----+----------+----------+--------+------------+

| ID | NAME | SALARY | DEPTNO | BDATE |

+----+----------+----------+--------+------------+

| 1 | John Doe | 50000.00 | 1 | 1990-01-15 |

+----+----------+----------+--------+------------+

1 row in set (0.00 sec)

mysql>

mysql> -- Update the salary of an employee

mysql> UPDATE Employee SET SALARY = 55000.00 WHERE ID = 1;

ERROR 1054 (42S22): Unknown column 'SALARY' in 'field list'

mysql>

mysql> -- Verify the updated state of the tables

mysql> SELECT \* FROM Department;

+--------+-----------+

| DeptNo | Total\_Sal |

+--------+-----------+

| 1 | 0.00 |

+--------+-----------+

1 row in set (0.00 sec)

mysql> SELECT \* FROM Employe;

+----+----------+----------+--------+------------+

| ID | NAME | SALARY | DEPTNO | BDATE |

+----+----------+----------+--------+------------+

| 1 | John Doe | 50000.00 | 1 | 1990-01-15 |

+----+----------+----------+--------+------------+

1 row in set (0.00 sec)

mysql>

mysql> -- Delete an employee

mysql> DELETE FROM Employee WHERE ID = 1;

Query OK, 0 rows affected (0.00 sec)

mysql>

mysql> -- Verify the final state of the tables

mysql> SELECT \* FROM Department;

+--------+-----------+

| DeptNo | Total\_Sal |

+--------+-----------+

| 1 | 0.00 |

+--------+-----------+

1 row in set (0.00 sec)

mysql> SELECT \* FROM Employe;

+----+----------+----------+--------+------------+

| ID | NAME | SALARY | DEPTNO | BDATE |

+----+----------+----------+--------+------------+

| 1 | John Doe | 50000.00 | 1 | 1990-01-15 |

+----+----------+----------+--------+------------+

1 row in set (0.00 sec)