Indian Institute of Information Technology Surat

Lab Report on

Advanced Database Management (CS 604) Practical

Submitted by

[RAHUL KUMAR SINGH] (UI21CS44)

Course Faculty

Mr. Rishi Sharma

Department of Computer Science and Engineering

Indian Institute of Information Technology Surat

Gujarat-394190, India

Jan-2024

Lab No: 1

Aim: Perform basic SQL Query on three tables (Employee, Title, Bonus)

Description: Create a Database for an Organization and create the following tables in the Organization Database:

Employee(EMP_ID(PK), FIRST_NAME, LAST_NAME, SALARY, JOINING_DATE, DEPARTMENT)
Bonus (EMP_REF_ID(FK EMP_ID), BONUS_AMOUNT, BONUS_DATE)
Title (EMP_REF_ID(FKEMP_ID), EMP_TITLE, AFFECTED_FROM)

Insert a minimum of 50 records in each table. Retrieve the following information from the Organization database:

1. SQL query to print all Employee details from the Employee table order by FIRST_NAME Ascending and DEPARTMENT Descending.

2. SQL query to fetch the count of employees working in the department 'Admin'.

3. SQL query to fetch Employee names with salaries >= 50000 and <= 100000.

4. SQL query to print details of the Workers who are also Managers.

5. SQL query to fetch duplicate records having matching data in some fields of a table.

6. SQL query to show only even rows from a table.

7. SQL query to show records from one table that another table does not have. Find employees in employee table that do not exist in bonus table (i.e. who did not get bonus)

8. SQL query to show the to pn(say10) records of a table.

9. Find people who have the same salary

10. SQL query to fetch the first 50% records from a table.

11. Find the highest 2 salaries without LIMIT or TOP.

12. Create a trigger to ensure that no employee of age less than 18 can be inserted in the database.

13. Create a trigger which will work before deletion in the employee table and create a duplicate copy of the record in another table employee_backup.

14. Create a trigger to count the number of new tuples inserted using each insert statement.

Source Code:

Database Creation:

```
CREATE DATABASE IF NOT EXISTS Organization;
USE Organization;
```

Create the Employee table

```
CREATE TABLE IF NOT EXISTS Employee (
    EMP_ID INT PRIMARY KEY,
    FIRST_NAME VARCHAR(50),
    LAST_NAME VARCHAR(50),
    SALARY DECIMAL(10, 2),
    JOINING_DATE DATE,
    DEPARTMENT VARCHAR(50)
);
```

Create the Bonus table

```
CREATE TABLE IF NOT EXISTS Bonus (
    EMP_REF_ID INT,
    BONUS_AMOUNT DECIMAL(10, 2),
    BONUS_DATE DATE,
    FOREIGN KEY (EMP_REF_ID) REFERENCES Employee(EMP_ID)
);
```

Create the Title table

```
CREATE TABLE IF NOT EXISTS Title (
    EMP_REF_ID INT,
    EMP_TITLE VARCHAR(50),
    AFFECTED_FROM DATE,
    FOREIGN KEY (EMP_REF_ID) REFERENCES Employee(EMP_ID)
);
```

Task 1:

```
SELECT * FROM Employee ORDER BY FIRST_NAME ASC, DEPARTMENT DESC;
```

Task 2:

```
SELECT COUNT(*) FROM Employee WHERE DEPARTMENT = 'Admin';
```

Task 3:

```
SELECT FIRST_NAME, LAST_NAME FROM Employee WHERE SALARY BETWEEN 50000 AND 100000;
```

Task 4:

```
SELECT Employee.* FROM Employee INNER JOIN Title ON Employee.EMP_ID = Title.EMP_REF_ID WHERE Title.EMP_TITLE =
'Manager';
```

Task 5:

```
SELECT SALARY, DEPARTMENT, COUNT(*) FROM Employee GROUP BY SALARY, DEPARTMENT HAVING COUNT(*) > 1;
```

Task 6:

```
WITH RankedRows AS (SELECT  *,  ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS RowNum FROM Employee) SELECT
* FROM RankedRows WHERE RowNum % 2 = 0;
```

Task 7:

```
SELECT Employee.* FROM Employee LEFT JOIN Bonus ON Employee.EMP_ID = Bonus.EMP_REF_ID WHERE Bonus.EMP_REF_ID
IS NULL;
```

Task 8:

```
SELECT * FROM Employee LIMIT 10;
```

Task 9:

```
SELECT SALARY, COUNT(*) FROM Employee GROUP BY SALARY HAVING COUNT(*) > 1;
```

Task 10:

```
WITH RankedRows AS (SELECT  *,  ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS RowNum FROM Employee) SELECT
* FROM RankedRows WHERE RowNum <= (SELECT COUNT(*)/2 FROM Employee);
```

Task 11:

```
WITH RankedRows AS (SELECT  *,  ROW_NUMBER() OVER (ORDER BY Salary DESC) AS RowNum FROM Employee) SELECT *
FROM RankedRows WHERE RowNum <= 2;
```

Task 12:

```
DELIMITER //
CREATE TRIGGER age_insert_employee
BEFORE INSERT ON Employee
FOR EACH ROW
BEGIN
   DECLARE emp_age INT;
   SET emp_age = YEAR(CURDATE()) - YEAR(NEW.JOINING_DATE) - (DATE_FORMAT(CURDATE(), '%m%d') <
DATE_FORMAT(NEW.JOINING_DATE, '%m%d'));
   IF emp_age < 18 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Cannot insert employee with age less than 18.';
   END IF;
END;
//
DELIMITER ;
```

Task 13:

```
CREATE TABLE IF NOT EXISTS Employee_backup (
   EMP_ID INT PRIMARY KEY,
   FIRST_NAME VARCHAR(50),
   LAST_NAME VARCHAR(50),
   SALARY DECIMAL(10, 2),
   JOINING_DATE DATE,
   DEPARTMENT VARCHAR(50)
);

DELIMITER //
CREATE TRIGGER before_delete_employee BEFORE DELETE ON Employee FOR EACH ROW
BEGIN
   INSERT INTO employee_backup (EMP_ID, FIRST_NAME, LAST_NAME, SALARY, JOINING_DATE, DEPARTMENT) VALUES
(OLD.EMP_ID, OLD.FIRST_NAME, OLD.LAST_NAME, OLD.SALARY, OLD.JOINING_DATE, OLD.DEPARTMENT);
END;
//
DELIMITER ;
```

Task 14:

```
CREATE TABLE insert_count (
   table_name VARCHAR(255) PRIMARY KEY,
   insert_count INT DEFAULT 0
);

DELIMITER //
CREATE TRIGGER after_insert_count_employee
AFTER INSERT ON Employee
FOR EACH ROW
BEGIN
   INSERT INTO insert_count (table_name, insert_count) VALUES ('Employee', 1) ON DUPLICATE KEY UPDATE insert_count =
insert_count + 1;
END;
//
DELIMITER ;
```

Output:

Task 1:

```
mysql> SELECT * FROM Employee ORDER BY FIRST_NAME ASC, DEPARTMENT DESC;
+------------+------------+-------------+-------------+--------------+------------------+
| EMP_ID     | FIRST_NAME | LAST_NAME   | SALARY      | JOINING_DATE | DEPARTMENT       |
+------------+------------+-------------+-------------+--------------+------------------+
|       9925 | Alanis     | Murphy      |      625.25 | 1982-02-14   | IT               |
|   29876816 | Anthony    | Ritchie     |    17591.97 | 1986-05-30   | Admin            |
| 2147483647 | Ari        | Schamberger |      984.90 | 2000-10-07   | Site Reliability |
|   65392242 | Arnoldo    | Beatty      | 18320386.20 | 1977-11-08   | Site Reliability |
|          6 | Assunta    | Paucek      |     5732.16 | 2016-05-10   | Admin            |
|        272 | Astrid     | Reilly      |   411176.20 | 2012-09-03   | Site Reliability |
|          9 | Blanca     | O'Conner    |        0.00 | 2015-09-28   | Finance          |
|         51 | Christiana | Ernser      |   208561.10 | 1996-01-06   | Admin            |
|          0 | Dalton     | Kilback     |      996.08 | 1997-09-10   | IT               |
|        120 | Delmer     | Tremblay    |    51420.77 | 2020-03-13   | IT               |
|         63 | Demond     | Mayert      |    12693.70 | 2016-01-23   | Sales            |
|          7 | Einar      | Hyatt       |        0.00 | 1985-09-05   | Site Reliability |
|          8 | Elisa      | Effertz     |     1256.37 | 2010-10-10   | Sales            |
|          2 | Emilio     | Fay         |        0.00 | 2007-03-22   | Site Reliability |
|        830 | Esteban    | Kuhic       |       24.47 | 1992-02-26   | IT               |
|         73 | Fernando   | Fisher      |    11979.81 | 1993-10-26   | Sales            |
|         60 | Fiona      | Gutkowski   |  4100378.98 | 1973-12-19   | IT               |
|         49 | Florine    | Reynolds    |        3.00 | 2012-03-23   | Finance          |
|    5806210 | Hermina    | Satterfield |      683.51 | 1982-03-13   | IT               |
|        569 | Hildegard  | Goldner     |        0.00 | 1986-11-10   | Finance          |
|       6288 | Hillary    | O'Kon       | 24164568.77 | 1978-03-06   | Sales            |
|         12 | Ismael     | Schneider   |        0.00 | 1976-02-08   | Site Reliability |
|       3828 | Jacinto    | Mosciski    |    13324.64 | 2006-10-30   | Sales            |
|          5 | Jaiden     | Hermann     |       12.66 | 1989-08-20   | IT               |
|        891 | Janis      | Bednar      | 99999999.99 | 2012-11-11   | Admin            |
|   59625769 | Jaren      | Dooley      |        0.00 | 2010-01-02   | Finance          |
|         81 | Jorge      | Powlowski   | 99999999.99 | 1981-11-22   | Admin            |
|        302 | Kennith    | D'Amore     |     1384.29 | 2014-11-22   | Site Reliability |
|        787 | Kian       | Gorczany    |   682896.00 | 2022-05-28   | Site Reliability |
|         56 | Kim        | Hayes       |       32.08 | 1972-10-02   | Admin            |
|      57880 | Laura      | Schmidt     |        2.18 | 2008-08-26   | IT               |
|        754 | Leonor     | White       |        0.00 | 1979-01-30   | Finance          |
|         31 | Lou        | Price       |   112791.61 | 2007-08-10   | Admin            |
|         57 | Luella     | Bradtke     |       91.02 | 2013-01-22   | Site Reliability |
|         41 | Marisa     | Emard       |       91.94 | 1999-05-01   | Admin            |
|        823 | Minnie     | Hilll       |     3119.20 | 1980-06-10   | Sales            |
|        987 | Mireya     | Kreiger     |        0.00 | 2018-01-12   | Site Reliability |
|     752068 | Nolan      | Schaden     |   141053.00 | 1993-03-20   | Sales            |
|        152 | Osborne    | Cremin      | 60144415.04 | 2001-01-18   | Site Reliability |
|          1 | Peggie     | Raynor      |   500190.78 | 2009-05-14   | Admin            |
|          3 | Rene       | Hintz       |        0.00 | 1986-02-22   | Sales            |
|        889 | Reyes      | Smitham     |     2046.90 | 1972-08-21   | Finance          |
|         40 | Ryan       | Kub         |       30.34 | 1989-03-30   | IT               |
|      16427 | Samanta    | Tillman     |       52.67 | 1985-08-05   | Site Reliability |
|         28 | Samara     | Glover      |    80301.10 | 2001-10-04   | Sales            |
|         68 | Shakira    | Wuckert     |        3.08 | 1992-10-10   | Sales            |
|        425 | Trey       | Emmerich    |        0.00 | 2016-08-06   | IT               |
|          4 | Zelda      | Tromp       |   297279.78 | 2007-12-23   | Finance          |
|     287508 | Zetta      | Schumm      | 60557563.96 | 2001-11-25   | Sales            |
|    1741200 | Zoey       | Donnelly    | 99999999.99 | 2016-02-09   | IT               |
+------------+------------+-------------+-------------+--------------+------------------+
```

Task 2:

```
mysql> SELECT COUNT(*) FROM Employee WHERE DEPARTMENT = 'Admin';
+----------+
| COUNT(*) |
+----------+
|        9 |
+----------+
1 row in set (0.00 sec)
```

Task 3:

```
mysql> SELECT FIRST_NAME, LAST_NAME FROM Employee WHERE SALARY BETWEEN 50000 AND 100000;
+------------+-----------+
| FIRST_NAME | LAST_NAME |
+------------+-----------+
| Samara     | Glover    |
| Delmer     | Tremblay  |
+------------+-----------+
```

Task 4:

```
mysql> SELECT Employee.* FROM Employee INNER JOIN Title ON Employee.EMP_ID = Title.EMP_REF_ID WHERE Title.EMP_TITLE = 'Manager';
+--------+------------+-----------+-----------+--------------+------------------+
| EMP_ID | FIRST_NAME | LAST_NAME | SALARY    | JOINING_DATE | DEPARTMENT       |
+--------+------------+-----------+-----------+--------------+------------------+
|      0 | Dalton     | Kilback   |    996.08 | 1997-09-10   | IT               |
|      1 | Peggie     | Raynor    | 500190.78 | 2009-05-14   | Admin            |
|      2 | Emilio     | Fay       |      0.00 | 2007-03-22   | Site Reliability |
|      3 | Rene       | Hintz     |      0.00 | 1986-02-22   | Sales            |
|      4 | Zelda      | Tromp     | 297279.78 | 2007-12-23   | Finance          |
|      5 | Jaiden     | Hermann   |     12.66 | 1989-08-20   | IT               |
+--------+------------+-----------+-----------+--------------+------------------+
6 rows in set (0.00 sec)
```

Task 5:

```
mysql> SELECT SALARY, DEPARTMENT, COUNT(*) FROM Employee GROUP BY SALARY, DEPARTMENT HAVING COUNT(*) > 1;
+-------------+------------------+----------+
| SALARY      | DEPARTMENT       | COUNT(*) |
+-------------+------------------+----------+
|        0.00 | Site Reliability |        4 |
|        0.00 | Finance          |        4 |
| 99999999.99 | Admin            |        2 |
+-------------+------------------+----------+
```

Task 6:

```
mysql> WITH RankedRows AS (SELECT  *,  ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS RowNum FROM Employee) SELECT * FROM RankedRows WHERE RowNum % 2 = 0;
+------------+------------+------------+-------------+--------------+------------------+--------+
| EMP_ID     | FIRST_NAME | LAST_NAME  | SALARY      | JOINING_DATE | DEPARTMENT       | RowNum |
+------------+------------+------------+-------------+--------------+------------------+--------+
|          1 | Peggie     | Raynor     |   500190.78 | 2009-05-14   | Admin            |      2 |
|          3 | Rene       | Hintz      |        0.00 | 1986-02-22   | Sales            |      4 |
|          5 | Jaiden     | Hermann    |       12.66 | 1989-08-20   | IT               |      6 |
|          7 | Einar      | Hyatt      |        0.00 | 1985-09-05   | Site Reliability |      8 |
|          9 | Blanca     | O'Conner   |        0.00 | 2015-09-28   | Finance          |     10 |
|         28 | Samara     | Glover     |    80301.10 | 2001-10-04   | Sales            |     12 |
|         40 | Ryan       | Kub        |       30.34 | 1989-03-30   | IT               |     14 |
|         49 | Florine    | Reynolds   |        3.00 | 2012-03-23   | Finance          |     16 |
|         56 | Kim        | Hayes      |       32.08 | 1972-10-02   | Admin            |     18 |
|         60 | Fiona      | Gutkowski  |  4100378.98 | 1973-12-19   | IT               |     20 |
|         68 | Shakira    | Wuckert    |        3.08 | 1992-10-10   | Sales            |     22 |
|         81 | Jorge      | Powlowski  | 99999999.99 | 1981-11-22   | Admin            |     24 |
|        152 | Osborne    | Cremin     | 60144415.04 | 2001-01-18   | Site Reliability |     26 |
|        302 | Kennith    | D'Amore    |     1384.29 | 2014-11-22   | Site Reliability |     28 |
|        569 | Hildegard  | Goldner    |        0.00 | 1986-11-10   | Finance          |     30 |
|        787 | Kian       | Gorczany   |   682896.00 | 2022-05-28   | Site Reliability |     32 |
|        830 | Esteban    | Kuhic      |       24.47 | 1992-02-26   | IT               |     34 |
|        891 | Janis      | Bednar     | 99999999.99 | 2012-11-11   | Admin            |     36 |
|       3828 | Jacinto    | Mosciski   |    13324.64 | 2006-10-30   | Sales            |     38 |
|       9925 | Alanis     | Murphy     |      625.25 | 1982-02-14   | IT               |     40 |
|      57880 | Laura      | Schmidt    |        2.18 | 2008-08-26   | IT               |     42 |
|     752068 | Nolan      | Schaden    |   141053.00 | 1993-03-20   | Sales            |     44 |
|    5806210 | Hermina    | Satterfield |      683.51 | 1982-03-13   | IT               |     46 |
|   59625769 | Jaren      | Dooley     |        0.00 | 2010-01-02   | Finance          |     48 |
| 2147483647 | Ari        | Schamberger |      984.90 | 2000-10-07   | Site Reliability |     50 |
+------------+------------+------------+-------------+--------------+------------------+--------+
```

Task 7:

```
mysql> SELECT Employee.* FROM Employee LEFT JOIN Bonus ON Employee.EMP_ID = Bonus.EMP_REF_ID WHERE Bonus.EMP_REF_ID IS NULL;
+------------+------------+-------------+-------------+--------------+------------------+
| EMP_ID     | FIRST_NAME | LAST_NAME   | SALARY      | JOINING_DATE | DEPARTMENT       |
+------------+------------+-------------+-------------+--------------+------------------+
|      16427 | Samanta    | Tillman     |       52.67 | 1985-08-05   | Site Reliability |
|      57880 | Laura      | Schmidt     |        2.18 | 2008-08-26   | IT               |
|     287508 | Zetta      | Schumm      | 60557563.96 | 2001-11-25   | Sales            |
|     752068 | Nolan      | Schaden     |   141053.00 | 1993-03-20   | Sales            |
|    1741200 | Zoey       | Donnelly    | 99999999.99 | 2016-02-09   | IT               |
|    5806210 | Hermina    | Satterfield |      683.51 | 1982-03-13   | IT               |
|   29876816 | Anthony    | Ritchie     |    17591.97 | 1986-05-30   | Admin            |
|   59625769 | Jaren      | Dooley      |        0.00 | 2010-01-02   | Finance          |
|   65392242 | Arnoldo    | Beatty      | 18320386.20 | 1977-11-08   | Site Reliability |
| 2147483647 | Ari        | Schamberger |      984.90 | 2000-10-07   | Site Reliability |
+------------+------------+-------------+-------------+--------------+------------------+
```

Task 8:

```
mysql> SELECT * FROM Employee LIMIT 10;
+--------+------------+-----------+-----------+--------------+------------------+
| EMP_ID | FIRST_NAME | LAST_NAME | SALARY    | JOINING_DATE | DEPARTMENT       |
+--------+------------+-----------+-----------+--------------+------------------+
|      0 | Dalton     | Kilback   |    996.08 | 1997-09-10   | IT               |
|      1 | Peggie     | Raynor    | 500190.78 | 2009-05-14   | Admin            |
|      2 | Emilio     | Fay       |      0.00 | 2007-03-22   | Site Reliability |
|      3 | Rene       | Hintz     |      0.00 | 1986-02-22   | Sales            |
|      4 | Zelda      | Tromp     | 297279.78 | 2007-12-23   | Finance          |
|      5 | Jaiden     | Hermann   |     12.66 | 1989-08-20   | IT               |
|      6 | Assunta    | Paucek    |   5732.16 | 2016-05-10   | Admin            |
|      7 | Einar      | Hyatt     |      0.00 | 1985-09-05   | Site Reliability |
|      8 | Elisa      | Effertz   |   1256.37 | 2010-10-10   | Sales            |
|      9 | Blanca     | O'Conner  |      0.00 | 2015-09-28   | Finance          |
+--------+------------+-----------+-----------+--------------+------------------+
```

Task 9:

```
mysql> SELECT SALARY, COUNT(*) FROM Employee GROUP BY SALARY HAVING COUNT(*) > 1;
+-------------+----------+
| SALARY      | COUNT(*) |
+-------------+----------+
|        0.00 |       10 |
| 99999999.99 |        3 |
+-------------+----------+
```

Task 10:

```
mysql> WITH RankedRows AS (SELECT  *,  ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS RowNum FROM Employee) SELECT * FROM RankedRows WHERE RowNum <= (SELECT
COUNT(*)/2 FROM Employee);
+--------+------------+-----------+-------------+--------------+------------------+--------+
| EMP_ID | FIRST_NAME | LAST_NAME | SALARY      | JOINING_DATE | DEPARTMENT       | RowNum |
+--------+------------+-----------+-------------+--------------+------------------+--------+
|      0 | Dalton     | Kilback   |      996.08 | 1997-09-10   | IT               |      1 |
|      1 | Peggie     | Raynor    |   500190.78 | 2009-05-14   | Admin            |      2 |
|      2 | Emilio     | Fay       |        0.00 | 2007-03-22   | Site Reliability |      3 |
|      3 | Rene       | Hintz     |        0.00 | 1986-02-22   | Sales            |      4 |
|      4 | Zelda      | Tromp     |   297279.78 | 2007-12-23   | Finance          |      5 |
|      5 | Jaiden     | Hermann   |       12.66 | 1989-08-20   | IT               |      6 |
|      6 | Assunta    | Paucek    |     5732.16 | 2016-05-10   | Admin            |      7 |
|      7 | Einar      | Hyatt     |        0.00 | 1985-09-05   | Site Reliability |      8 |
|      8 | Elisa      | Effertz   |     1256.37 | 2010-10-10   | Sales            |      9 |
|      9 | Blanca     | O'Conner  |        0.00 | 2015-09-28   | Finance          |     10 |
|     12 | Ismael     | Schneider |        0.00 | 1976-02-08   | Site Reliability |     11 |
|     28 | Samara     | Glover    |    80301.10 | 2001-10-04   | Sales            |     12 |
|     31 | Lou        | Price     |   112791.61 | 2007-08-10   | Admin            |     13 |
|     40 | Ryan       | Kub       |       30.34 | 1989-03-30   | IT               |     14 |
|     41 | Marisa     | Emard     |       91.94 | 1999-05-01   | Admin            |     15 |
|     49 | Florine    | Reynolds  |        3.00 | 2012-03-23   | Finance          |     16 |
|     51 | Christiana | Ernser    |   208561.10 | 1996-01-06   | Admin            |     17 |
|     56 | Kim        | Hayes     |       32.08 | 1972-10-02   | Admin            |     18 |
|     57 | Luella     | Bradtke   |       91.02 | 2013-01-22   | Site Reliability |     19 |
|     60 | Fiona      | Gutkowski |  4100378.98 | 1973-12-19   | IT               |     20 |
|     63 | Demond     | Mayert    |    12693.70 | 2016-01-23   | Sales            |     21 |
|     68 | Shakira    | Wuckert   |        3.08 | 1992-10-10   | Sales            |     22 |
|     73 | Fernando   | Fisher    |    11979.81 | 1993-10-26   | Sales            |     23 |
|     81 | Jorge      | Powlowski | 99999999.99 | 1981-11-22   | Admin            |     24 |
|    120 | Delmer     | Tremblay  |    51420.77 | 2020-03-13   | IT               |     25 |
+--------+------------+-----------+-------------+--------------+------------------+--------+
25 rows in set (0.00 sec)
```

Task 11:

```
mysql> WITH RankedRows AS (SELECT  *,  ROW_NUMBER() OVER (ORDER BY Salary DESC) AS RowNum FROM Employee) SELECT * FROM RankedRows WHERE RowNum <= 2;
+--------+------------+-----------+-------------+--------------+------------+--------+
| EMP_ID | FIRST_NAME | LAST_NAME | SALARY      | JOINING_DATE | DEPARTMENT | RowNum |
+--------+------------+-----------+-------------+--------------+------------+--------+
|    891 | Janis      | Bednar    | 99999999.99 | 2012-11-11   | Admin      |      1 |
|     81 | Jorge      | Powlowski | 99999999.99 | 1981-11-22   | Admin      |      2 |
+--------+------------+-----------+-------------+--------------+------------+--------+
2 rows in set (0.00 sec)
```

Task 12:

```
mysql> INSERT INTO Employee (EMP_ID, FIRST_NAME, LAST_NAME, SALARY, JOINING_DATE, DEPARTMENT) VALUES (103,'hyu','guy',89,'2017-09-10','IT');
ERROR 1644 (45000): Cannot insert employee with age less than 18.
mysql>
```

Task 13:

```
mysql> INSERT INTO Employee (EMP_ID, FIRST_NAME, LAST_NAME, SALARY, JOINING_DATE, DEPARTMENT) VALUES (103,'hyu','guy',89,'1997-09-10','IT');
Query OK, 1 row affected (0.01 sec)

mysql> delete from employee where EMP_ID = 103;
Query OK, 1 row affected (0.01 sec)

mysql> select * from employee_backup;
+--------+------------+-----------+--------+--------------+------------+
| EMP_ID | FIRST_NAME | LAST_NAME | SALARY | JOINING_DATE | DEPARTMENT |
+--------+------------+-----------+--------+--------------+------------+
|    103 | hyu        | guy       |  89.00 | 1997-09-10   | IT         |
+--------+------------+-----------+--------+--------------+------------+
1 row in set (0.00 sec)
```

Task 14:

```
mysql> INSERT INTO Employee (EMP_ID, FIRST_NAME, LAST_NAME, SALARY, JOINING_DATE, DEPARTMENT) VALUES (103,'hyu','guy',89,'1997-09-10','IT');
Query OK, 1 row affected (0.01 sec)

mysql> select * from insert_count;
+------------+--------------+
| table_name | insert_count |
+------------+--------------+
| Employee   |            1 |
+------------+--------------+
1 row in set (0.00 sec)

mysql> INSERT INTO Employee (EMP_ID, FIRST_NAME, LAST_NAME, SALARY, JOINING_DATE, DEPARTMENT) VALUES (104,'hyu','guy',89,'1997-09-10','IT');
Query OK, 1 row affected (0.01 sec)

mysql> select * from insert_count;
+------------+--------------+
| table_name | insert_count |
+------------+--------------+
| Employee   |            2 |
+------------+--------------+
1 row in set (0.00 sec)
```

Conclusion:

- Triggers are powerful mechanisms in SQL that can be used to enforce data integrity, automate tasks, and maintain historical records.
- Triggers enhance the reliability and security of the database by enforcing rules and executing actions automatically in response to specific events.
- Using basic sql statements to solve complex queries.

Lab No: 2

Aim: Write a PL/SQL code block to find total and average of 6 subjects and display the grade.

Description: The provided PL/SQL code block calculates the total and average scores of a student in six subjects and determines the corresponding grade based on a simple grading system.

- The code assumes the existence of a table named student_scores with columns for student ID and scores in each of the six subjects.
- It fetches the scores for a specified student ID, calculates the total and average, and assigns a grade (A, B, C, D, or F) based on the average score.
- The results, including the student ID, total, average, and grade, are then displayed using the DBMS_OUTPUT.PUT_LINE function.

Source Code:

With Table:

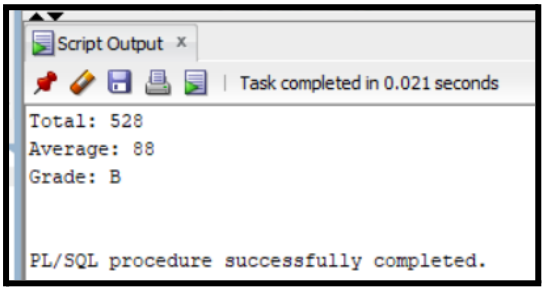
```
-- Table Creation
CREATE TABLE student_scores (
  student_id NUMBER,
  sub1 NUMBER,
  sub2 NUMBER,
  sub3 NUMBER,
  sub4 NUMBER,
  sub5 NUMBER,
  sub6 NUMBER
);

-- PL/SQL block to calculate total, average, and display grade
DECLARE
   v_student_id student_scores.student_id%TYPE;
   v_sub1 student_scores.sub1%TYPE;
   v_sub2 student_scores.sub2%TYPE;
   v_sub3 student_scores.sub3%TYPE;
   v_sub4 student_scores.sub4%TYPE;
   v_sub5 student_scores.sub5%TYPE;
   v_sub6 student_scores.sub6%TYPE;
   v_total NUMBER;
   v_average NUMBER;
   v_grade VARCHAR2(2);
BEGIN
   SELECT student_id, sub1, sub2, sub3, sub4, sub5, sub6
   INTO v_student_id, v_sub1, v_sub2, v_sub3, v_sub4, v_sub5, v_sub6
   FROM student_scores
   WHERE student_id = 1;

   -- Calculate total and average
   v_total := v_sub1 + v_sub2 + v_sub3 + v_sub4 + v_sub5 + v_sub6;
   v_average := v_total / 6;

   -- Grade Determination
   IF v_average >= 90 THEN
      v_grade := 'A';
   ELSIF v_average >= 80 THEN
      v_grade := 'B';
   ELSIF v_average >= 70 THEN
      v_grade := 'C';
   ELSIF v_average >= 60 THEN
      v_grade := 'D';
   ELSE
```

```
        v_grade := 'F';
    END IF;
    -- Results
    DBMS_OUTPUT.PUT_LINE('Student ID: ' || v_student_id);
    DBMS_OUTPUT.PUT_LINE('Total: ' || v_total);
    DBMS_OUTPUT.PUT_LINE('Average: ' || v_average);
    DBMS_OUTPUT.PUT_LINE('Grade: ' || v_grade);
END;
/
```

Without_Table:

```
SET SERVEROUTPUT ON;
DECLARE
   subject1 NUMBER := 85;
   subject2 NUMBER := 92;
   subject3 NUMBER := 78;
   subject4 NUMBER := 90;
   subject5 NUMBER := 88;
   subject6 NUMBER := 95;
   total NUMBER;
   average NUMBER;
   grade VARCHAR2(2);

BEGIN
   total := subject1 + subject2 + subject3 + subject4 + subject5 + subject6;
   average := total / 6;
   IF average >= 90 THEN
      grade := 'A';
   ELSIF average >= 80 THEN
      grade := 'B';
   ELSIF average >= 70 THEN
      grade := 'C';
   ELSIF average >= 60 THEN
      grade := 'D';
   ELSE
      grade := 'F';
   END IF;

   DBMS_OUTPUT.PUT_LINE('Total: ' || total);
   DBMS_OUTPUT.PUT_LINE('Average: ' || average);
   DBMS_OUTPUT.PUT_LINE('Grade: ' || grade);
END;
/
```

Output:



Conclusion:

- The code is structured in a modular manner using a PL/SQL block for better understanding.
- The code is designed for execution in interactive environments.
- Utilized DECLARE and BEGIN sections to define variables and execute procedural logic.
- Applied the DBMS_OUTPUT.PUT_LINE function for displaying total marks, average marks, and the corresponding grade.

Lab No: 3

Aim: Write a PL/SQL (MySQL Procedure) code block to perform specific tasks on tables Teacher, Class and Pay_scale.

Description: Consider the following table to write PL/SQL code as specified under

- Teacher (t_no, f_name, l_name, salary, supervisor, joining_date, birth_date, title)
- Class (class_no, t_no, room_no)
- Pay_scale (Min_limit, Max_limit, grade)

1. Accept a range of salary and print the details of teachers from the teacher table.
2. By using cursor - Calculate the bonus amount to be given to a teacher depending on the following conditions:
   a) if salary < 10000 then bonus is 10% of the salary.
   b) if the salary is between 10000 and 20000 then bonus is 20% of the salary.
   c) if the salary is between 20000 and 25000 then bonus is 25% of the salary.
   d) if the salary exceeds 25000 then the bonus is 30% of the salary.
3. Using a simple LOOP structure, list the first 10 records of the 'teachers' table.
4. Accept the room number and display the teacher details like t_no, f_name, l_name, birth_date, title from table Teacher.

Source Code:

Teacher Table:

```
CREATE TABLE Teacher (
    t_no INT PRIMARY KEY,
    f_name VARCHAR(50) NOT NULL,
    l_name VARCHAR(50) NOT NULL,
    salary DECIMAL(10, 2) NOT NULL,
    supervisor INT,
    joining_date DATE NOT NULL,
    birth_date DATE NOT NULL,
    title VARCHAR(50) NOT NULL
);
```

Class Table:

```
CREATE TABLE Class (
    class_no INT PRIMARY KEY,
    t_no INT,
    room_no INT,
    FOREIGN KEY (t_no) REFERENCES Teacher(t_no),
    UNIQUE KEY unique_teacher_class (t_no, room_no)
);
```

Pay_scale Table:

```
CREATE TABLE Pay_scale (
    Min_limit DECIMAL(10, 2) NOT NULL,
    Max_limit DECIMAL(10, 2) NOT NULL,
    grade VARCHAR(10) PRIMARY KEY
);
```

Task 1:

```
DELIMITER //
CREATE PROCEDURE GetTeachersBySalaryRange(
    IN minSalary DECIMAL(10, 2),
    IN maxSalary DECIMAL(10, 2)
)
BEGIN
    SELECT *
    FROM Teacher
    WHERE salary BETWEEN minSalary AND maxSalary;
```

```
END //
DELIMITER ;
```

Task 2:

```
DELIMITER //
CREATE PROCEDURE CalculateTeacherBonus()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE t_no_var INT;
   DECLARE salary_var DECIMAL(10, 2);
   DECLARE bonus_var DECIMAL(10, 2);
   DECLARE teacher_cursor CURSOR FOR
      SELECT t_no, salary
      FROM Teacher;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
   OPEN teacher_cursor;
   teacher_loop: LOOP
      FETCH teacher_cursor INTO t_no_var, salary_var;
      IF done THEN
         LEAVE teacher_loop;
      END IF;
      IF salary_var < 10000 THEN
         SET bonus_var = salary_var * 0.10;
      ELSEIF salary_var BETWEEN 10000 AND 20000 THEN
         SET bonus_var = salary_var * 0.20;
      ELSEIF salary_var BETWEEN 20000 AND 25000 THEN
         SET bonus_var = salary_var * 0.25;
      ELSE
         SET bonus_var = salary_var * 0.30;
      END IF;
      SELECT t_no_var AS Teacher_ID, salary_var AS Salary, bonus_var AS Bonus;
   END LOOP;
   CLOSE teacher_cursor;
END //
DELIMITER ;
```

Task 3:

```
DELIMITER //
CREATE PROCEDURE ListFirst10Teachers()
BEGIN
   DECLARE teacher_cursor CURSOR FOR
      SELECT t_no, f_name, l_name, salary, supervisor, joining_date, birth_date, title
      FROM Teacher;
   OPEN teacher_cursor;
   teacher_loop: LOOP
      FETCH teacher_cursor INTO t_no_var, f_name_var, l_name_var, salary_var, supervisor_var, joining_date_var, birth_date_var, title_var;
      IF counter >= 10 OR t_no_var IS NULL THEN
         LEAVE teacher_loop;
      END IF;
      SET counter = counter + 1;
      SELECT t_no_var AS Teacher_ID, f_name_var AS First_Name, l_name_var AS Last_Name, salary_var AS Salary,
           supervisor_var AS Supervisor, joining_date_var AS Joining_Date, birth_date_var AS Birth_Date, title_var AS Title;
   END LOOP;
   CLOSE teacher_cursor;
END //
DELIMITER ;
```

Task 4:

```
DELIMITER //
CREATE PROCEDURE GetTeachersByRoomNumber(IN roomNumber INT)
BEGIN
   SELECT t.t_no, t.f_name, t.l_name, t.birth_date, t.title
   FROM Teacher t
   JOIN Class c ON t.t_no = c.t_no
   WHERE c.room_no = roomNumber;
END //
DELIMITER ;
```

Output:

Task 1:

```
mysql> CALL GetTeachersBySalaryRange(100000,200000);
+-------+-----------+------------+-----------+------------+--------------+------------+-----------------------------------------------------+
| t_no  | f_name    | l_name     | salary    | supervisor | joining_date | birth_date | title                                               |
+-------+-----------+------------+-----------+------------+--------------+------------+-----------------------------------------------------+
|     0 | accusamus | voluptatem | 184162.00 |          0 | 2008-07-13   | 1993-03-21 | Neque sed in officia nisi velit placeat nulla.      |
|     2 | ipsam     | magnam     | 194450.00 |          0 | 1998-11-20   | 1992-06-13 | Debitis enim consequatur error.                     |
|     6 | eligendi  | omnis      | 108521.00 |          1 | 2020-07-20   | 1994-03-21 | Ex porro corrupti sint.                             |
|  5527 | qui       | qui        | 104527.00 |          0 | 2007-12-26   | 1979-07-17 | Et nemo itaque minima asperiores vel quia quis.     |
| 45571 | cum       | nihil      | 149521.00 |          1 | 1985-11-02   | 2021-12-25 | Itaque minima repellat odit tempore corrupti nisi   |
+-------+-----------+------------+-----------+------------+--------------+------------+-----------------------------------------------------+
```

Task 2:

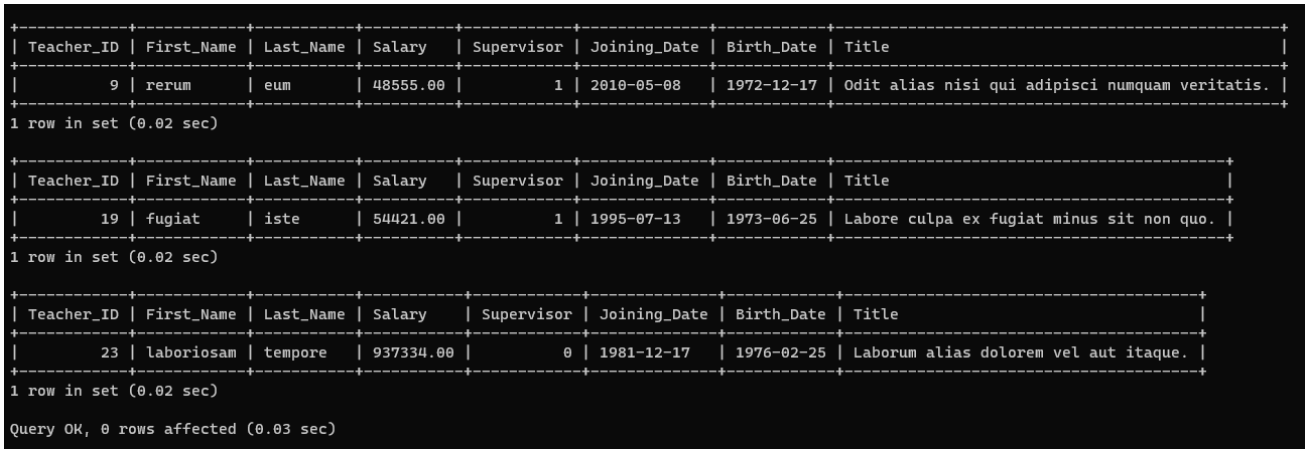
```
mysql> CALL CalculateTeacherBonus();
+------------+-----------+----------+
| Teacher_ID | Salary    | Bonus    |
+------------+-----------+----------+
|          0 | 184162.00 | 55248.60 |
+------------+-----------+----------+
1 row in set (0.00 sec)

+------------+-----------+----------+
| Teacher_ID | Salary    | Bonus    |
+------------+-----------+----------+
|          1 | 239637.00 | 71891.10 |
+------------+-----------+----------+
1 row in set (0.00 sec)
```

```
+------------+-----------+----------+
| Teacher_ID | Salary    | Bonus    |
+------------+-----------+----------+
|          2 | 194450.00 | 58335.00 |
+------------+-----------+----------+
1 row in set (0.00 sec)

+------------+-----------+-----------+
| Teacher_ID | Salary    | Bonus     |
+------------+-----------+-----------+
|          3 | 731563.00 | 219468.90 |
+------------+-----------+-----------+
1 row in set (0.00 sec)
```

```
+------------+-----------+-----------+
| Teacher_ID | Salary    | Bonus     |
+------------+-----------+-----------+
|          5 | 661912.00 | 198573.60 |
+------------+-----------+-----------+
1 row in set (0.00 sec)
```

Task 3:

```
mysql> CALL ListFirst10Teachers();
+------------+------------+------------+-----------+------------+--------------+------------+-----------------------------------------------------+
| Teacher_ID | First_Name | Last_Name  | Salary    | Supervisor | Joining_Date | Birth_Date | Title                                               |
+------------+------------+------------+-----------+------------+--------------+------------+-----------------------------------------------------+
|          0 | accusamus  | voluptatem | 184162.00 |          0 | 2008-07-13   | 1993-03-21 | Neque sed in officia nisi velit placeat nulla.      |
+------------+------------+------------+-----------+------------+--------------+------------+-----------------------------------------------------+
1 row in set (0.01 sec)

+------------+-------------+-----------+-----------+------------+--------------+------------+--------------------------------------+
| Teacher_ID | First_Name  | Last_Name | Salary    | Supervisor | Joining_Date | Birth_Date | Title                                |
+------------+-------------+-----------+-----------+------------+--------------+------------+--------------------------------------+
|          1 | consequatur | beatae    | 239637.00 |          1 | 1985-10-10   | 1972-06-21 | Beatae tempore deleniti doloribus qui. |
+------------+-------------+-----------+-----------+------------+--------------+------------+--------------------------------------+
1 row in set (0.01 sec)

+------------+------------+-----------+-----------+------------+--------------+------------+--------------------------------+
| Teacher_ID | First_Name | Last_Name | Salary    | Supervisor | Joining_Date | Birth_Date | Title                          |
+------------+------------+-----------+-----------+------------+--------------+------------+--------------------------------+
|          2 | ipsam      | magnam    | 194450.00 |          0 | 1998-11-20   | 1992-06-13 | Debitis enim consequatur error. |
+------------+------------+-----------+-----------+------------+--------------+------------+--------------------------------+
1 row in set (0.01 sec)

+------------+------------+-----------+-----------+------------+--------------+------------+----------------------------+
| Teacher_ID | First_Name | Last_Name | Salary    | Supervisor | Joining_Date | Birth_Date | Title                      |
+------------+------------+-----------+-----------+------------+--------------+------------+----------------------------+
|          3 | ut         | saepe     | 731563.00 |          0 | 1991-04-19   | 1991-07-17 | Autem et dolores accusantium. |
+------------+------------+-----------+-----------+------------+--------------+------------+----------------------------+
1 row in set (0.01 sec)

+------------+------------+-----------+-----------+------------+--------------+------------+-----------------------------------------------------+
| Teacher_ID | First_Name | Last_Name | Salary    | Supervisor | Joining_Date | Birth_Date | Title                                               |
+------------+------------+-----------+-----------+------------+--------------+------------+-----------------------------------------------------+
|          5 | totam      | ab        | 661912.00 |          1 | 2013-11-16   | 1986-03-30 | Ut aperiam qui iusto velit unde similique illo imp |
+------------+------------+-----------+-----------+------------+--------------+------------+-----------------------------------------------------+
1 row in set (0.02 sec)

+------------+------------+-----------+-----------+------------+--------------+------------+-------------------------+
| Teacher_ID | First_Name | Last_Name | Salary    | Supervisor | Joining_Date | Birth_Date | Title                   |
+------------+------------+-----------+-----------+------------+--------------+------------+-------------------------+
|          6 | eligendi   | omnis     | 108521.00 |          1 | 2020-07-20   | 1994-03-21 | Ex porro corrupti sint. |
+------------+------------+-----------+-----------+------------+--------------+------------+-------------------------+
1 row in set (0.02 sec)

+------------+-------------+-----------+-----------+------------+--------------+------------+-----------------------------------------+
| Teacher_ID | First_Name  | Last_Name | Salary    | Supervisor | Joining_Date | Birth_Date | Title                                   |
+------------+-------------+-----------+-----------+------------+--------------+------------+-----------------------------------------+
|          7 | cupiditate  | eligendi  | 505168.00 |          1 | 1972-11-23   | 1988-10-22 | Enim vero et dolorem consectetur voluptatem. |
+------------+-------------+-----------+-----------+------------+--------------+------------+-----------------------------------------+
1 row in set (0.02 sec)
```



Teacher_ID	First_Name	Last_Name	Salary	Supervisor	Joining_Date	Birth_Date	Title
9	rerum	eum	48555.00	1	2010-05-08	1972-12-17	Odit alias nisi qui adipisci numquam veritatis.

1 row in set (0.02 sec)

Teacher_ID	First_Name	Last_Name	Salary	Supervisor	Joining_Date	Birth_Date	Title
19	fugiat	iste	54421.00	1	1995-07-13	1973-06-25	Labore culpa ex fugiat minus sit non quo.

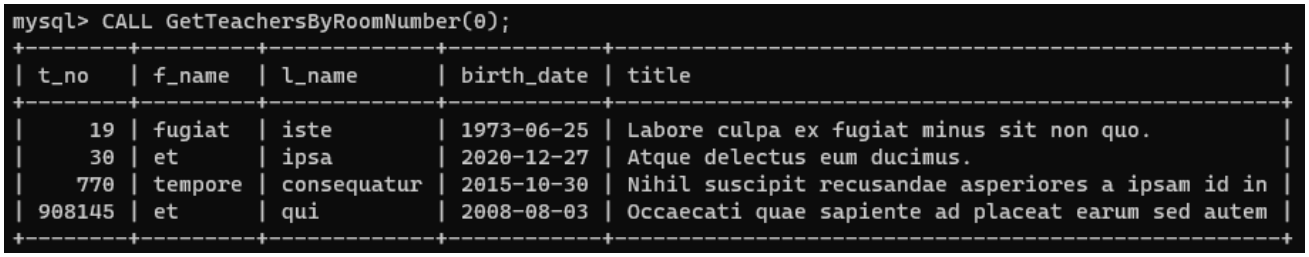
1 row in set (0.02 sec)

Teacher_ID	First_Name	Last_Name	Salary	Supervisor	Joining_Date	Birth_Date	Title
23	laboriosam	tempore	937334.00	0	1981-12-17	1976-02-25	Laborum alias dolorem vel aut itaque.

1 row in set (0.02 sec)

Query OK, 0 rows affected (0.03 sec)

Task 4:



```
mysql> CALL GetTeachersByRoomNumber(0);
```

t_no	f_name	l_name	birth_date	title
19	fugiat	iste	1973-06-25	Labore culpa ex fugiat minus sit non quo.
30	et	ipsa	2020-12-27	Atque delectus eum ducimus.
770	tempore	consequatur	2015-10-30	Nihil suscipit recusandae asperiores a ipsam id in
908145	et	qui	2008-08-03	Occaecati quae sapiente ad placeat earum sed autem

Conclusion:

- GetTeachersBySalaryRange: Accepts a salary range and retrieves details of teachers from the Teacher table within that range.
- CalculateTeacherBonus: Uses a cursor to calculate the bonus amount for teachers based on salary conditions, considering different bonus percentages for specific salary ranges.
- ListFirst10Teachers: Utilizes a simple LOOP structure to list the first 10 records from the Teacher table.
- GetTeachersByRoomNumber: Accepts a room number and displays specific details (t_no, f_name, l_name, birth_date, title) of teachers associated with that room from the Teacher table.
- The code is structured in a modular manner using a MySQL Procedure block for better understanding.
- The code is designed for execution in interactive environments.
- Utilized DECLARE and BEGIN sections to define variables and execute procedural logic.
- Applied the DBMS_OUTPUT.PUT_LINE function for displaying output for all the procedures.

.

Lab No: 4

Aim: Design and develop a suitable Student Database application. One of the attributes to me maintained is the attendance of a student in each subject for which he/she has enrolled.

Description: Using TRIGGERS, we write active rules to do the following:

a) Whenever attendance is updated, check if the attendance is less than 85%; if so notify the Head of Department concerned.

b) Whenever the marks in the Internal Assessment Test are entered, check if the marks are less than 40%; if so, notify the Head of the Department concerned.

Source Code:

Student_Attendance Table:

```
CREATE TABLE Student_Attendance (
        t_no INT PRIMARY KEY,
        Day1 TINYINT, Day2 TINYINT, Day3 TINYINT, Day4 TINYINT, Day5 TINYINT,
        Day6 TINYINT, Day7 TINYINT, Day8 TINYINT, Day9 TINYINT, Day10 TINYINT
);
```

Student_Marks Table:

```
CREATE TABLE Student_Marks (
        t_no INT PRIMARY KEY,
        Sub1 DECIMAL(5,2), Sub2 DECIMAL(5,2), Sub3 DECIMAL(5,2),
        Sub4 DECIMAL(5,2), Sub5 DECIMAL(5,2), Sub6 DECIMAL(5,2)
);
```

Task 1:

```
DELIMITER //
CREATE TRIGGER after_insert_attendance_student
AFTER INSERT ON Student_Attendance
FOR EACH ROW
BEGIN
  IF
((NEW.Day1+NEW.Day2+NEW.Day3+NEW.Day4+NEW.Day5+NEW.Day6+NEW.Day7+NEW.Day8+NEW.Day9+NEW.Day10)*(100/10) <
85) THEN
        -- CALL NotifyHeadOfDepartment(NEW.t_no, 'Low internal assessment marks');
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = "Notice: Attendance for the entered student is less than 85%";
  END IF;
END;
//
DELIMITER ;
```

Task 2:

```
DELIMITER //
CREATE TRIGGER after_insert_marks_student
AFTER INSERT ON Student_Marks
FOR EACH ROW
BEGIN
  IF ((NEW.Sub1+NEW.Sub2+NEW.Sub3+NEW.Sub4+NEW.Sub5+NEW.Sub6)/6 < 40) THEN
        -- CALL NotifyHeadOfDepartment(NEW.t_no, 'Low internal assessment marks');
        SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = "Notice: Marks for Internal Assessment are less than 40%";
  END IF;
END;
//
DELIMITER ;
```

Test:

```
CREATE TRIGGER AttendanceTrigger
AFTER UPDATE ON Enrollments
FOR EACH ROW
BEGIN
```

```
    IF NEW.Attendance < 85 THEN
        CALL NotifyHeadOfDepartment(NEW.StudentID, 'Low attendance');
    END IF;
END;

CREATE TRIGGER AssessmentTrigger
AFTER UPDATE ON Enrollments
FOR EACH ROW
BEGIN
    IF NEW.InternalAssessmentMarks < 40 THEN
        CALL NotifyHeadOfDepartment(NEW.StudentID, 'Low internal assessment marks');
    END IF;
END;

DELIMITER //
CREATE PROCEDURE NotifyHeadOfDepartment(IN studentID INT, IN message VARCHAR(255))
BEGIN
    SELECT studentID, ": ", message as Output;
    -- DECLARE departmentHeadEmail VARCHAR(255);
    -- SELECT Email INTO departmentHeadEmail
    -- FROM DepartmentHeads
    -- WHERE DepartmentID = (SELECT DepartmentID FROM Students WHERE StudentID = studentID);
    -- CALL SendEmail(departmentHeadEmail, message);
END //
DELIMITER ;
```

Output:

Task 1:

```
mysql> INSERT INTO `Student_Attendance` VALUES (5,0,0,1,0,1,0,1,0,1,1);
ERROR 1643 (02000): Notice: Attendance for the entered student is less than 85%
mysql> INSERT INTO `Student_Attendance` VALUES (6,0,1,1,1,1,1,1,1,1,1);
Query OK, 1 row affected (0.00 sec)
```

Task 2:

```
mysql> INSERT INTO `Student_Marks` VALUES (5,20.0,20.0,30.0,50.0,60.0,20.0);
ERROR 1643 (02000): Notice: Marks for Internal Assessment are less than 40%
mysql> INSERT INTO `Student_Marks` VALUES (6,50.0,40.0,30.0,50.0,60.0,30.0);
Query OK, 1 row affected (0.01 sec)
```

Conclusion:

- The code is structured in a modular manner using a MySQL Procedure block for better understanding.
- The code is designed for execution in interactive environments.
- Utilized BEGIN sections to define variables and execute trigger logic.
- Created “NotifyHeadOfDepartment” and “SendEmail” Procedure to notify and send mail to the respected authority.
- Applied the DBMS_OUTPUT.PUT_LINE function for displaying output for all the procedures.

Lab No: 5

Aim: To implement Deadlock Detection Algorithm for Distributed Database using Wait-for Graph to check for Deadlock.

Description:

- DetectDeadlock Procedure:
  - Creates a temporary table for wait-for graph.
  - Populates wait-for graph with data from deadlock_info.
- DepthFirstSearch Procedure:
  - Simulates stack for DFS using a temporary table.
  - Detects cycles by recursively traversing wait-for graph.
- DetectDeadlock Procedure Modification:
  - Inserts multiple rows into dfs_stack from wait_for_graph where requesting_node=start_node.

Source Code:

Table:

```
CREATE TABLE Deadlock_Info (
   transaction_id INT PRIMARY KEY AUTO_INCREMENT,
   requesting_node INT,
   holding_node INT
);
```

Detect Deadlock Procedure:

```
DELIMITER //
CREATE PROCEDURE DetectDeadlock()
BEGIN
   DECLARE result INT DEFAULT 0;
   DECLARE temp INT DEFAULT 0;
   DECLARE start_node INT;
   DECLARE current_node INT;
   DECLARE done INT DEFAULT 0;
   CREATE TEMPORARY TABLE IF NOT EXISTS wait_for_graph (
      requesting INT,
      holding INT
   );
   INSERT INTO wait_for_graph SELECT requesting_node, holding_node FROM deadlock_info;
   SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
   CREATE TEMPORARY TABLE IF NOT EXISTS distinct_nodes (
      node INT
   );
   INSERT INTO distinct_nodes
   SELECT DISTINCT requesting
   FROM wait_for_graph;
   WHILE (SELECT COUNT(*) FROM distinct_nodes) > 0 DO
      SELECT node INTO start_node FROM distinct_nodes ORDER BY node LIMIT 1;
      DELETE FROM distinct_nodes WHERE node = start_node;
      SELECT CONCAT("Start Node: ",start_node) as message;
      SELECT result;
      CALL DepthFirstSearch(start_node, start_node, temp);
      IF temp = 1 THEN
         SET result = 1;
      END IF;
      DELETE FROM wait_for_graph;
      INSERT INTO wait_for_graph SELECT requesting_node, holding_node FROM deadlock_info;
   END WHILE;
   SELECT result;
   IF result = 0  THEN
      SELECT "No Deadlock Detected!" as message;
   ELSE
      SELECT "Deadlock Detected!" as message;
```

```
    END IF;
    DROP TEMPORARY TABLE IF EXISTS wait_for_graph;
    DROP TEMPORARY TABLE IF EXISTS distinct_nodes;
END //
DELIMITER ;
```

Output:

In case of No Deadlock:

```
mysql> call removedeadlockinfo(3,2);
Query OK, 1 row affected (0.02 sec)

mysql> CALL DetectDeadlock();
+-----------------------+
| message               |
+-----------------------+
| No Deadlock Detected! |
+-----------------------+
1 row in set (0.01 sec)

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

In case of Deadlock:

```
mysql> CALL DetectDeadlock();
+-------------------------------+------------+------+--------------+
| Deadlock detected: Cycle from | start_node | to   | current_node |
+-------------------------------+------------+------+--------------+
| Deadlock detected: Cycle from |          2 |  to  |            2 |
+-------------------------------+------------+------+--------------+
1 row in set (0.01 sec)

+-------------------------------+------------+------+--------------+
| Deadlock detected: Cycle from | start_node | to   | current_node |
+-------------------------------+------------+------+--------------+
| Deadlock detected: Cycle from |          3 |  to  |            3 |
+-------------------------------+------------+------+--------------+
1 row in set (0.01 sec)

+--------------------+
| message            |
+--------------------+
| Deadlock Detected! |
+--------------------+
1 row in set (0.01 sec)

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

Conclusion:

- Detects deadlocks in a distributed database using a wait-for graph.
- Implemented depth-first search within MySQL stored procedures.
- Procedures manage deadlock information, simulate DFS, and execute deadlock detection.
- MySQL limitations for complex algorithms; use external languages for efficiency.
- Deadlock detection results in cycles which means careful consideration using it.

Lab No: 6

Aim: To Implement Object Oriented Approach for writing PL/SQL codes (MySQL)

Description:

A) Write a PL/SQL code to create a class for a "Person" with attributes such as name, age, and address.
B) Write a PL/SQL code to Implement methods in the "Person" class to display the details and update the age.
C) Write a PL/SQL code to implement a method to calculate the annual bonus based on the salary in the "Employee" class.
D) Write a PL/SQL code to create a "Manager" subclass inheriting from the "Employee" class, and add an attribute to store the number of employees managed.

Source Code:

A) Implementing a "Person" class:

```
Drop TABLE Person;
CREATE TABLE Person (
  objectId VARCHAR(100) PRIMARY KEY,
  name VARCHAR(100),
  age INT,
  address VARCHAR(200)
);
-- Dropping Procedures
DROP PROCEDURE DisplayDetails;
DROP PROCEDURE UpdateAge;
DROP PROCEDURE AppendPerson;
```

B) Implementing methods to create object, display details and update age:

```
DELIMITER //
CREATE PROCEDURE DisplayDetails(IN object_id VARCHAR(100))
BEGIN
  SELECT name, age, address FROM Person WHERE objectId = object_id;
END //
CREATE PROCEDURE UpdateAge(IN object_id VARCHAR(100), IN new_age INT)
BEGIN
  UPDATE Person SET age = new_age WHERE objectId = object_id;
END //
CREATE PROCEDURE AppendPerson(IN object_id VARCHAR(100), IN person_name VARCHAR(100), IN new_age INT, IN
person_address VARCHAR(100))
BEGIN
  INSERT INTO Person values (object_id, person_name, new_age, person_address);
END //
DELIMITER ;
-- Calling Procedures:
CALL AppendPerson("person1", "Rahul Kumar Singh", 20, "Raigarh");
CALL DisplayDetails("person1");
CALL UpdateAge("person1", 21);
CALL DisplayDetails("person1");
```

C) Implementing methods to create object, display details and calculate the annual bonus based on salary:

```
-- Dropping Procedures
DROP PROCEDURE DisplayEmpDetails;
DROP PROCEDURE AppendEmployee;
DROP PROCEDURE CalculateAnnualBonus;
DELIMITER //
CREATE PROCEDURE DisplayEmpDetails(IN object_id VARCHAR(100))
BEGIN
  SELECT * FROM Employee WHERE objectId = object_id;
END //
CREATE PROCEDURE AppendEmployee(IN object_id VARCHAR(100), IN person_name VARCHAR(100), IN new_age INT, IN
```

```
person_address VARCHAR(100), IN salary DECIMAL(10,2))
BEGIN
  INSERT INTO Employee values (object_id, person_name, new_age, person_address, salary);
END //
CREATE PROCEDURE CalculateAnnualBonus(IN object_id VARCHAR(100))
BEGIN
  SELECT salary * 0.1 FROM Employee WHERE objectId = object_id; -- 10% bonus rate.
END //
DELIMITER ;
-- Calling Procedures
CALL CalculateAnnualBonus(4000);
```

D) Implementing an "Employee" and "Manager" subclass:

```
Drop TABLE Employee;
Drop TABLE Manager;
CREATE TABLE Employee (
  objectId VARCHAR(100) PRIMARY KEY,
  name VARCHAR(100),
  age INT,
  address VARCHAR(200),
  salary DECIMAL(10,2)
);
CREATE TABLE Manager AS
SELECT * FROM Employee;
ALTER TABLE Manager
ADD num_employees_managed INT;
-- Dropping Procedures
DROP PROCEDURE DisplayManDetails;
DROP PROCEDURE CalculateManagerBonus;
DROP PROCEDURE AppendManager;
DELIMITER //
CREATE PROCEDURE DisplayManDetails(IN object_id VARCHAR(100))
BEGIN
  SELECT * FROM Manager WHERE objectId = object_id;
END //
CREATE PROCEDURE AppendManager(IN object_id VARCHAR(100), IN person_name VARCHAR(100), IN new_age INT, IN
person_address VARCHAR(100), IN salary DECIMAL(10,2), IN num_emp INT)
BEGIN
  INSERT INTO Manager values (object_id, person_name, new_age, person_address, salary, num_emp);
END //
CREATE PROCEDURE CalculateManagerBonus(IN object_id VARCHAR(100))
BEGIN
  SELECT salary * 0.15 + num_employees_managed * 1000 FROM Manager WHERE objectId = object_id; -- Bonus
END //
DELIMITER ;
-- Calling Procedures
CALL AppendManager("manager1", "Rahul Kumar Singh", 20, "Raigarh", 10000.00, 10);
CALL DisplayManDetails("manager1");
CALL CalculateManagerBonus("manager1");
```

Output:

A) Implementing a "Person" class:

```
mysql> CALL AppendPerson("person1", "Rahul Kumar Singh", 20, "Raigarh");
Query OK, 1 row affected (0.00 sec)

mysql> CALL DisplayDetails("person1");
+------------------+------+---------+
| name             | age  | address |
+------------------+------+---------+
| Rahul Kumar Singh |   20 | Raigarh |
+------------------+------+---------+
1 row in set (0.00 sec)

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

B) Implementing methods to create object, display details and update age:

```
mysql> CALL UpdateAge("person1", 21);
Query OK, 1 row affected (0.00 sec)

mysql> CALL DisplayDetails("person1");
+-------------------+------+---------+
| name              | age  | address |
+-------------------+------+---------+
| Rahul Kumar Singh |   21 | Raigarh |
+-------------------+------+---------+
1 row in set (0.00 sec)

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

C) Implementing methods to create object, display details and calculate the annual bonus based on salary:

```
mysql> CALL DisplayEmpDetails("employee1");
+-----------+-------------------+------+---------+----------+
| objectId  | name              | age  | address | salary   |
+-----------+-------------------+------+---------+----------+
| employee1 | Rahul Kumar Singh |   20 | Raigarh | 10000.00 |
+-----------+-------------------+------+---------+----------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> CALL CalculateAnnualBonus("employee1");
+--------------+
| salary * 0.1 |
+--------------+
|     1000.000 |
+--------------+
1 row in set (0.00 sec)

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

D) Implementing an "Employee" and "Manager" subclass:

```
mysql> CALL AppendManager("manager1", "Rahul Kumar Singh", 20, "Raigarh", 10000.00, 10);
Query OK, 1 row affected (0.00 sec)

mysql> CALL DisplayManDetails("manager1");
+----------+-------------------+------+---------+----------+-----------------------+
| objectId | name              | age  | address | salary   | num_employees_managed |
+----------+-------------------+------+---------+----------+-----------------------+
| manager1 | Rahul Kumar Singh |   20 | Raigarh | 10000.00 |                    10 |
+----------+-------------------+------+---------+----------+-----------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> CALL CalculateManagerBonus("manager1");
+-------------------------------------------+
| salary * 0.15 + num_employees_managed * 1000 |
+-------------------------------------------+
|                                11500.0000 |
+-------------------------------------------+
1 row in set (0.00 sec)

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

Conclusion:

- Help us to understand the importance of object-oriented approach.
- Provide various features of object-oriented approach like Polymorphism, Inheritance and Encapsulation.
- To be able to implement the MySQL code into an Object-oriented programming model.

Lab No: 7

Aim: To Implement basic relational methods and statements using PostgreSQL.

Description:

1. Write a PSQL statement to create a simple table countries including columns country_id,country_name and region_id.
2. Write a PSQL statement to create a simple table countries including columns country_id,country_name and region_id which already exist.
3. Write a PSQL statement to create the structure of a table dup_countries similar to countries.
4. Write a PSQL statement to create a duplicate copy of countries table including structure and data by name dup_countries.
5. Write a PSQL statement to create a table countries set a constraint NULL.
6. Write a PSQL statement to create a table named jobs including columns job_id, job_title, min_salary, max_salary and check whether the max_salary amount exceeding the upper limit 25000.
7. Write a PSQL statement to create a table named countries including columns country_id, country_name and region_id and make sure that no countries except Italy, India and China will be entered in the table.
8. Write a PSQL statement to create a table named countries including columns country_id,country_name and region_id and make sure that no duplicate data against column country_id will be allowed at the time of insertion.
9. Write a PSQL statement to create a table named jobs including columns job_id, job_title, min_salary and max_salary, and make sure that, the default value for job_title is blank and min_salary is 8000 and max_salary is NULL will be entered automatically at the time of insertion if no value assigned for the specified columns.
10. Write a PSQL statement to create a table named countries including columns country_id, country_name and region_id and make sure that the country_id column will be a key field which will not contain any duplicate data at the time of insertion.
11. Write a PSQL statement to create a table countries including columns country_id, country_name and region_id and make sure that the column country_id will be unique and store an auto-incremented value.

Click me to see the solution

12. Write a PSQL statement to create a table countries including columns country_id, country_name and region_id and make sure that the combination of columns country_id and region_id will be unique.

Source Code & Output:

Task 1:

```
postgres=# CREATE TABLE countries (
postgres(#     country_id SERIAL PRIMARY KEY,
postgres(#     country_name VARCHAR(100) NOT NULL,
postgres(#     region_id INT
postgres(# );
CREATE TABLE
```

Task 2:

```
postgres=# CREATE TABLE new_countries AS
postgres-# SELECT country_id, country_name, region_id
postgres-# FROM countries;
SELECT 0
postgres=# select * from new_countries;
 country_id | country_name | region_id
------------+--------------+-----------
```

Task 3:

```
postgres=# CREATE TABLE dup_countries (LIKE countries INCLUDING ALL);
CREATE TABLE
postgres=# select * from dup_countries;
 country_id | country_name | region_id
------------+--------------+-----------
```

Task 4:

```
postgres=# CREATE TABLE dup_countries2 AS
postgres-# SELECT *
postgres-# FROM countries
postgres-# WITH NO DATA;
CREATE TABLE AS
postgres=# INSERT INTO dup_countries
postgres-# SELECT *
postgres-# FROM countries;
INSERT 0 0
postgres=# select * from dup_countries2;
 country_id | country_name | region_id
------------+--------------+-----------
```

Task 5:

```
postgres=# CREATE TABLE countries (
postgres(#     country_id SERIAL PRIMARY KEY,
postgres(#     country_name VARCHAR(100) NULL,
postgres(#     region_id INT
postgres(# );
CREATE TABLE
postgres=# select * from countries;
 country_id | country_name | region_id
------------+--------------+-----------
```

Task 6:

```
postgres=# CREATE TABLE jobs (
postgres(#     job_id SERIAL PRIMARY KEY,
postgres(#     job_title VARCHAR(100) NOT NULL,
postgres(#     min_salary NUMERIC(10, 2) DEFAULT 0.00,
postgres(#     max_salary NUMERIC(10, 2) CHECK (max_salary <= 25000)
postgres(# );
CREATE TABLE
```

```
postgres=# INSERT INTO jobs VALUES (3, 'A', 0, 122000);
ERROR:  new row for relation "jobs" violates check constraint "jobs_max_salary_check"
DETAIL:  Failing row contains (3, A, 0.00, 122000.00).
```

Task 7:

```
postgres=# CREATE TYPE allowed_countries AS ENUM ('Italy', 'India', 'China');
CREATE TYPE
postgres=# CREATE TABLE countries (
postgres(#     country_id SERIAL PRIMARY KEY,
postgres(#     country_name allowed_countries NOT NULL,
postgres(#     region_id INT
postgres(# );
CREATE TABLE
postgres=# INSERT INTO countries VALUES (1, 'USA', 1);
ERROR:  invalid input value for enum allowed_countries: "USA"
LINE 1: INSERT INTO countries VALUES (1, 'USA', 1);
                                         ^
postgres=# INSERT INTO countries VALUES (2, 'India', 1);
INSERT 0 1
```

Task 8:

```
postgres=# CREATE TABLE countries (
postgres(#     country_id SERIAL PRIMARY KEY,
postgres(#     country_name VARCHAR(100) NOT NULL,
postgres(#     region_id INT,
postgres(#     CONSTRAINT unique_country_id UNIQUE (country_id)
postgres(# );
CREATE TABLE
postgres=# INSERT INTO countries VALUES (1, 'India', 1);
INSERT 0 1
postgres=# INSERT INTO countries VALUES (1, 'China', 2);
ERROR:  duplicate key value violates unique constraint "unique_country_id"
DETAIL:  Key (country_id)=(1) already exists.
```

Task 9:

```
postgres=# CREATE TABLE jobs (
postgres(#     job_id SERIAL PRIMARY KEY,
postgres(#     job_title VARCHAR(100) DEFAULT '',
postgres(#     min_salary NUMERIC(10, 2) DEFAULT 8000.00,
postgres(#     max_salary NUMERIC(10, 2) DEFAULT NULL
postgres(# );
CREATE TABLE
postgres=# INSERT INTO jobs DEFAULT VALUES;
INSERT 0 1
postgres=# SELECT * FROM jobs;
 job_id | job_title | min_salary | max_salary
--------+-----------+------------+------------
      1 |           |    8000.00 |
(1 row)
```

Task 10:

```
postgres=# CREATE TABLE countries (
postgres(#     country_id SERIAL PRIMARY KEY,
postgres(#     country_name VARCHAR(100) NOT NULL,
postgres(#     region_id INT,
postgres(#     CONSTRAINT unique_country_id UNIQUE (country_id)
postgres(# );
CREATE TABLE
postgres=# INSERT INTO countries VALUES (1, 'India', 1);
INSERT 0 1
postgres=# INSERT INTO countries VALUES (1, 'China', 2);
ERROR:  duplicate key value violates unique constraint "unique_country_id"
DETAIL:  Key (country_id)=(1) already exists.
```

Task 11:

```
postgres=# CREATE TABLE jobs (
postgres(#     job_id SERIAL PRIMARY KEY,
postgres(#     job_title VARCHAR(100),
postgres(#     min_salary NUMERIC(10, 2),
postgres(#     max_salary NUMERIC(10, 2)
postgres(# );
CREATE TABLE
postgres=# INSERT INTO jobs
postgres-# (job_title, min_salary, max_salary)
postgres-# VALUES ('AI', 0, 10000);
INSERT 0 1
postgres=# SELECT * FROM jobs;
 job_id | job_title | min_salary | max_salary
--------+-----------+------------+------------
      1 | AI        |       0.00 |   10000.00
(1 row)
```

Task 12:

```
postgres=# CREATE TABLE countries (
postgres(#     country_id INT,
postgres(#     country_name VARCHAR(100) NOT NULL,
postgres(#     region_id INT NOT NULL,
postgres(#     CONSTRAINT unique_country_region UNIQUE (country_id, region_id)
postgres(# );
CREATE TABLE
postgres=# INSERT INTO countries VALUES (1, 'India', 1);
INSERT 0 1
postgres=# INSERT INTO countries VALUES (1, 'China', 2);
INSERT 0 1
postgres=# INSERT INTO countries VALUES (1, 'Italy', 2);
ERROR:  duplicate key value violates unique constraint "unique_country_region"
DETAIL:  Key (country_id, region_id)=(1, 2) already exists.
```

Conclusion:

- PSQL is PostgreSQL command-line interface for database management.
- Similarity to MySQL language in terms of both statements and structures.
- SERIAL in place of AUTO_INCREMENT
- Doesn't have an explicit statement for displaying a list of table (but could be done by using command **\dt**)