National Institute of Technology Kurukshetra, Kurukshetra - 136119

Master Of Computer Application

MCA – 132 (2024-27)

Database Management System Lab

Lab coordinator: Dr. S Suresh

Submitted by:

Name: Jayesh Solanki

Semester: 2nd

Section: G-4

Roll no: 524110055

Submitted to:

Mr. Kuldeep Sir

Date: 02-05-2025

DECLARATION

I, **Jayesh Solanki**, hereby declare that the **Database Management System Lab Manual** submitted by me is my own original and independently completed work.

While I have referred to online resources and sample code snippets to enhance my understanding of the concepts, the structure, implementation, and content of the manual are based entirely on my own logic and effort.

This manual is distinct from others because:

1. I have included **screenshots of program outputs**, organized in a folder named after me, as proof of successful execution on my own system.
2. Each program is **modular, well-structured**, and developed using my personal understanding of the topics.
3. **No part** of this manual has been copied from any student or external source.
4. The manual is systematically organized with a **comprehensive table of contents** for easy navigation.

I fully understand and uphold the principles of academic integrity and take complete responsibility for the work I have submitted.

\

INDEX

S. No.	Assignment	Questions	Page No.
1	Assignment-1	1. Create table Student (Rno, Name, DOB, Gender, Class, College, City, Marks). Insert five records in student table. 2. Display the detail structure of student table and Display the information of all the students. 3. Display Rno, Name and Class information of 'Patiala' students. Display information in ascending order of marks. 4. Change the marks of Rno 5 to 89 and Change the name and city of Rno 9. 5. Delete the information of 'Amritsar' city records and also Delete the records of student where marks<30.	1
2	Assignment-2	1. Get employee no and employee name who work in dept no 10, 20 and 30? 2. Display the employee names of those clerks whose salary > 2000 and also get all details of employees whose salary between 2000 and 3000? 3. Display name of those employees whose commission is NULL? 4. Display name of employees having two 'a' or 'A' chars in the name and also display the name of the employees whose second char is 'b' or 'B'? 5. Display maximum, minimum, average salary of deptno 10 employees and also display total number of employees working in deptno 20.	3
3	Assignment-3	1. Write queries to: - Display the system date, current day, current month and spell out year, spell out current date, round the system date on month, display the date of next Friday, truncate the system date on month, find the day after three days, and also check whether it is AM or PM right now. 2. Queries Based on EMP table: - Display day of date of joining column, display those employees who join the company on Monday, display those employees who join the company this month, and display those employees who join the company in last 30 days. 3. Create a table train having four columns (Train Number, date of Departure, time of departure, time of arrival) - Display all the records, display those trains which arrived on PM, and display train number who are going to depart in next one hour.	6
4	Assignment-4	1. Create table emp (empno, ename, job, sal, deptno) with constraints. a) Insert records, check error messages, list constraint names. b) Check constraint name for applied constraints. c) Drop unique constraint on ename and change data type. 2. Create tables S (Salesperson), P (Part), SP (sno, pno, qty) with appropriate keys. a) Insert records and violate foreign key constraints. b) Drop Foreign Key constraint on sno and pno. c) Alter table SP to drop foreign key constraint.	9
5	Assignment-5	1. Identify primary and foreign keys and create tables. 2. Violate the foreign key constraint and identify errors.	13

		3. Delete record of supplier S1 from S and write a note. 4. Get Qty supplied for Red parts and Sname supplying Red part. 5. For each part supplied get part number and names of all cities supplying the part. 6. Get all pairs of supplier numbers in the same city. 7. Drop table SP and create SP1 with delete cascade. 8. Delete S1 from S and write note. 9. Drop SP1 and create SP2 with delete set null. 10. Delete S1 from S and write note. 11. Alter SP2 and drop foreign key constraint. 12. Add foreign key constraint with delete set default option.	
6	Assignment-6	1. Column constraints: City in ('Amritsar', 'Delhi', 'Batala', 'Qadian') 2. Column constraints: Qty 100-1000, Weight NOT NULL, Shame NOT NULL, Phame UNIQUE 3. Get supplier name who supplies at least one red part. 4. Get supplier number who supplies maximum quantity. 5. Get supplier number who supply quantity > average quantity. 6. Increase quantity of part P1 by 10%. 7. Total parts supplied by S1. 8. Count parts with red color. 9. Count red parts supplied by Ajay. 10. Total quantity supplied by S1. 11. Change red parts supplied by Ajay to Green. 12. Delete all entries of part P1.	17
7	Assignment-7	1) PL/SQL block to find sum of two numbers. 2) PL/SQL block to find larger of two numbers. 3) PL/SQL block to find greatest of three numbers. 4) PL/SQL block to generate multiplication table for 3 to n. 5) PL/SQL block to calculate fine for book return based on days late. 6) PL/SQL block to generate multiplication table.	22
8	Assignment-8	1. PL/SQL to update total sal for empno 100. 2. PL/SQL to update total sal for all employees. 3. Add exception handlers for sal update of empno 100. 4. Exception handlers for check, primary key, foreign key, value width. 5. User-defined exception if date of issue > date of return.	26
9	Assignment-9	1. WAP to print employees of deptno 10. 2. WAP to find top 5 highest paid employees. 3. Perform Q1 & Q2 using cursor FOR loop. 4. PL/SQL to calculate simple interest for principal 1000, time 2 years, rate 5-15. 5. PL/SQL to print employees of a department (user input deptno).	32
10	Assignment-10	1. Write a stored procedure to calculate addition, subtraction, multiplication and division of two numbers. Call this procedure from a block. Write a local procedure for the same. 2. Write functions and procedures to perform Insert, Update, Delete and Retrieve operations on emp. 3. Write a trigger to implement primary key constraint and foreign key constraint.	38

		4. Write a trigger such that no updation on emp can takes place on Sunday. 5. Write a trigger such that commission cannot be greater than sal. (eno, ename, job, sal, comm., dno)	

Assignment-1

1. **Create table Student (Roll no, name, dob, gender, class, college, city, marks).**

Query:

```
CREATE TABLE tbl_student (
        rno INT PRIMARY KEY,
        student_name VARCHAR(50),
        student_dob DATE,
        student_gender CHAR(1),
        student_class VARCHAR(50),
        student_college VARCHAR(50),
        student_city VARCHAR(50),
        student_marks INT
);

INSERT INTO tbl_student VALUES (1, 'John Doe', '2000-01-01', 'M', 'XII', 'ABC College', 'Patiala', 90);
INSERT INTO tbl_student VALUES (2, 'Jane Doe', '2000-01-02', 'F', 'XII', 'ABC College', 'Amritsar', 85);
INSERT INTO tbl_student VALUES (5, 'Alice', '2000-01-03', 'F', 'XII', 'ABC College', 'Patiala', 70);
INSERT INTO tbl_student VALUES (4, 'Bob', '2000-01-04', 'M', 'XII', 'ABC College', 'Kolkata', 45);
INSERT INTO tbl_student VALUES (9, 'Charlie', '2000-01-05', 'M', 'XII', 'ABC College', 'Delhi', 20);
```

2. **Display the detail structure of student table and Display the information of all the students.**

Query:

```
desc tbl_student;
Select * from tbl_student;
```

Output:

```
mysql> desc tbl_student;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| rno             | int         | NO   | PRI | NULL    |       |
| student_name    | varchar(50) | YES  |     | NULL    |       |
| student_dob     | date        | YES  |     | NULL    |       |
| student_gender  | char(1)     | YES  |     | NULL    |       |
| student_class   | varchar(50) | YES  |     | NULL    |       |
| student_college | varchar(50) | YES  |     | NULL    |       |
| student_city    | varchar(50) | YES  |     | NULL    |       |
| student_marks   | int         | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)

mysql> Select * from tbl_student;
+-----+--------------+-------------+----------------+---------------+-----------------+--------------+
| rno | student_name | student_dob | student_gender | student_class | student_college | student_city |
+-----+--------------+-------------+----------------+---------------+-----------------+--------------+
|   1 | John Doe     | 2000-01-01  | M              | XII           | ABC College     | Patiala      |
|   2 | Jane Doe     | 2000-01-02  | F              | XII           | ABC College     | Amritsar     |
|   4 | Bob          | 2000-01-04  | M              | XII           | ABC College     | Kolkata      |
|   5 | Alice        | 2000-01-03  | F              | XII           | ABC College     | Patiala      |
|   9 | Charlie      | 2000-01-05  | M              | XII           | ABC College     | Delhi        |
+-----+--------------+-------------+----------------+---------------+-----------------+--------------+
5 rows in set (0.00 sec)
```

3. **Display Roll no, name and class information of 'Patiala' students. Display information in ascending order of marks.**

Query:

SELECT rno, student_name, student_class FROM tbl_student WHERE student_city='Patiala';
SELECT * FROM tbl_student ORDER BY student_city ASC;

Output:

```
mysql> SELECT rno, student_name, student_class FROM tbl_student WHERE student_city='Patiala';
+-----+--------------+---------------+
| rno | student_name | student_class |
+-----+--------------+---------------+
|   1 | John Doe     | XII           |
|   5 | Alice        | XII           |
+-----+--------------+---------------+
2 rows in set (0.00 sec)
```

4. **Change the marks of Roll no 5 to 89 and change the name and city of Roll no 9.**

Query:

UPDATE tbl_student SET student_marks=89 WHERE rno=5;
UPDATE tbl_student SET student_name='Abhishek Paswan', student_city='Kolkata' WHERE rno=9;

Output:

```
mysql> SELECT * FROM tbl_student ORDER BY student_city ASC;
+-----+--------------+-------------+----------------+---------------+-----------------+--------------+
| rno | student_name | student_dob | student_gender | student_class | student_college | student_city |
+-----+--------------+-------------+----------------+---------------+-----------------+--------------+
|   2 | Jane Doe     | 2000-01-02  | F              | XII           | ABC College     | Amritsar     |
|   9 | Charlie      | 2000-01-05  | M              | XII           | ABC College     | Delhi        |
|   4 | Bob          | 2000-01-04  | M              | XII           | ABC College     | Kolkata      |
|   1 | John Doe     | 2000-01-01  | M              | XII           | ABC College     | Patiala      |
|   5 | Alice        | 2000-01-03  | F              | XII           | ABC College     | Patiala      |
+-----+--------------+-------------+----------------+---------------+-----------------+--------------+
5 rows in set (0.00 sec)
```

5. **Delete the information of 'Amritsar' city records and also Delete the records of students where marks<30.**

Query:

DELETE FROM tbl_student WHERE student_city='Amritsar';
DELETE FROM tbl_student WHERE student_marks<30;

Output:

```
mysql> DELETE FROM tbl_student WHERE student_city='Amritsar';
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM tbl_student WHERE student_marks<30;
Query OK, 1 row affected (0.00 sec)
```

Assignment-2

Create Emp table: Columns are EmpNo, Ename, Job, Salary, date of joining, Commission, DeptNO. Insert 5 records by storing Null value in some records for commission column.

Query:

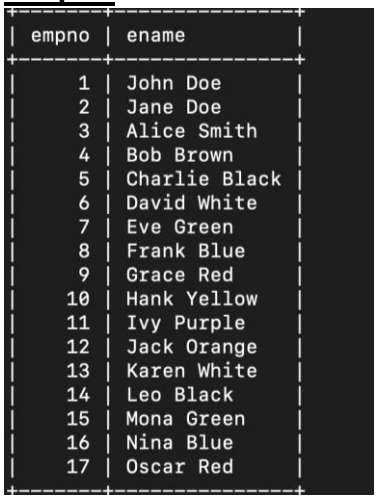
```
CREATE TABLE tbl_emp(
   empno int primary key,
    ename varchar(50),
    job varchar(50),
    salary float,
    doj date,
    commission int,
    deptno int
);
INSERT INTO tbl_emp VALUES (1, 'John Doe', 'Manager',5000.56,'2000-01-03', 10, 20);
INSERT INTO tbl_emp VALUES (2, 'Jane Doe', 'Receptionist',2000,'2002-01-15', 11,10);
INSERT INTO tbl_emp VALUES (3, 'Alice Smith', 'Developer', 4500.75, '2005-03-12', 12, 30);
INSERT INTO tbl_emp VALUES (4, 'Bob Brown', 'Analyst', 4000.00, '2008-07-23', 13, 20);
INSERT INTO tbl_emp VALUES (5, 'Charlie Black', 'Clerk', 2500.50, '2010-11-05', 14, 10);
INSERT INTO tbl_emp VALUES (6, 'David White', 'Salesman', 3000.00, '2012-09-17', 15, 20);
INSERT INTO tbl_emp VALUES (7, 'Eve Green', 'HR', 3500.25, '2014-05-30', 16, 30);
INSERT INTO tbl_emp VALUES (8, 'Frank Blue', 'Support', 2800.80, '2016-02-14', 17, 20);
INSERT INTO tbl_emp VALUES (9, 'Grace Red', 'Consultant', 5000.00, '2018-08-21', 18, 10);
INSERT INTO tbl_emp VALUES (10, 'Hank Yellow', 'Engineer', 4700.60, '2020-12-01', 19, 30);
INSERT INTO tbl_emp VALUES (11, 'Ivy Purple', 'Technician', 3200.45, '2021-04-18', 20, 10);
INSERT INTO tbl_emp VALUES (12, 'Jack Orange', 'Designer', 3800.90, '2022-06-25', 21, 20);
INSERT INTO tbl_emp VALUES (13, 'Karen White', 'Manager', 5100.00, '2023-01-10', NULL, 30);
INSERT INTO tbl_emp VALUES (14, 'Leo Black', 'Developer', 4600.75, '2023-02-15', NULL, 20);
INSERT INTO tbl_emp VALUES (15, 'Mona Green', 'Analyst', 4200.00, '2023-03-20', NULL, 10);
INSERT INTO tbl_emp VALUES (16, 'Nina Blue', 'Clerk', 2700.50, '2023-04-25', NULL, 20);
INSERT INTO tbl_emp VALUES (17, 'Oscar Red', 'Salesman', 3100.00, '2023-05-30', NULL, 30);
```

1. **Get employee no and employee name who work in dept no 10, 20 and 30?**

Query:

```
SELECT empno,ename FROM tbl_emp WHERE deptno in (10,20,30);
```

Output:



empno	ename
1	John Doe
2	Jane Doe
3	Alice Smith
4	Bob Brown
5	Charlie Black
6	David White
7	Eve Green
8	Frank Blue
9	Grace Red
10	Hank Yellow
11	Ivy Purple
12	Jack Orange
13	Karen White
14	Leo Black
15	Mona Green
16	Nina Blue
17	Oscar Red

2. **Display Display the employee names of those clerks whose salary > 2000 and also get all details of employees whose salary between 2000 and 3000 ?**

Query:

SELECT ename FROM tbl_emp WHERE job='Clerk' and salary>2000;
SELECT * FROM tbl_emp where salary between 2000 and 3000;

Output:

```
+---------------+
| ename         |
+---------------+
| Charlie Black |
| Nina Blue     |
+---------------+
2 rows in set (0.00 sec)
```

```
+-------+---------------+--------------+--------+------------+------------+--------+
| empno | ename         | job          | salary | doj        | commission | deptno |
+-------+---------------+--------------+--------+------------+------------+--------+
|     2 | Jane Doe      | Receptionist |   2000 | 2002-01-15 |         11 |     10 |
|     5 | Charlie Black | Clerk        | 2500.5 | 2010-11-05 |         14 |     10 |
|     6 | David White   | Salesman     |   3000 | 2012-09-17 |         15 |     20 |
|     8 | Frank Blue    | Support      | 2800.8 | 2016-02-14 |         17 |     20 |
|    16 | Nina Blue     | Clerk        | 2700.5 | 2023-04-25 |       NULL |     20 |
+-------+---------------+--------------+--------+------------+------------+--------+
5 rows in set (0.00 sec)
```

3. **Display name of those employees whose commission is NULL?**

Query:

SELECT ename FROM tbl_emp WHERE commission is NULL;

Output:

```
+-------------+
| ename       |
+-------------+
| Karen White |
| Leo Black   |
| Mona Green  |
| Nina Blue   |
| Oscar Red   |
+-------------+
```

4. **Display name of employees having two 'e' or 'E' chars in the name and also display the name of the employees whose second char is 'b' or ‘B'?**

Query:

```
SELECT ename from tbl_emp where length(ename)-length(replace(ename,'e',''))=2 or
length(ename)-length(replace(ename,'E',''))=2;
SELECT ename from tbl_emp where ename like '_e%' or '_E%';
```

Output:

```
+-------------+
| ename       |
+-------------+
| Jane Doe    |
| Grace Red   |
| Karen White |
| Mona Green  |
+-------------+
4 rows in set (0.00 sec)
```

```
+------------+
| ename      |
+------------+
| Leo Black  |
+------------+
1 row in set,
```

5. **Display maximum, minimum, average salary of deptno 10 employees and also display total number of employees working in deptno 20.**

Query:

```
SELECT MAX(salary),MIN(salary),AVG(salary) from tbl_emp WHERE deptno=10;
SELECT COUNT(empno) AS EmployeeCount, ename
FROM tbl_emp
WHERE deptno = 20
GROUP BY ename;
```

Output:

```
+-------------+-------------+-------------------+
| MAX(salary) | MIN(salary) | AVG(salary)       |
+-------------+-------------+-------------------+
|        5000 |        2000 | 3380.189990234375 |
+-------------+-------------+-------------------+
1 row in set (0.00 sec)
```

```
+---------------+-------------+
| EmployeeCount | ename       |
+---------------+-------------+
|             1 | John Doe    |
|             1 | Bob Brown   |
|             1 | David White |
|             1 | Frank Blue  |
|             1 | Jack Orange |
|             1 | Leo Black   |
|             1 | Nina Blue   |
+---------------+-------------+
7 rows in set (0.01 sec)
```

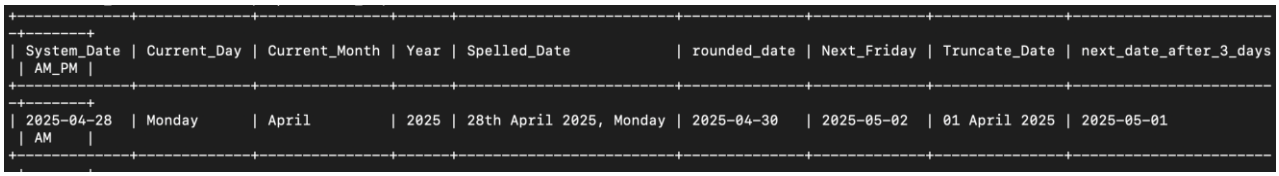
Assignment-3

1. **Display the system date, current day, current month and spell out year, spell out current date, round the system date on month, display the date of next Friday, truncate the system date on month, find the day after three days, and also check whether it is AM or PM right now.**

Query:

```
Select curdate() as System_Date ,
dayname(curdate()) as Current_Day,
monthname(curdate()) as Current_Month,
year(curdate()) as Year,
date_format(curdate(),"%D %M %Y, %W") as Spelled_Date,
case
when day(curdate())<=15 then date_format(curdate(),"01 %m %y")
else last_day(curdate())
end as rounded_date,
date_add(curdate(), INTERVAL (if(weekday(current_date())  < 4,4 - weekday(current_date()),11-
                        weekday(current_date())))  DAY) as Next_Friday,
date_format(curdate(),"01 %M %Y") as Truncate_Date,
date_add(curdate(), Interval 3 day) as next_date_after_3_days,
time_format(curtime(),"%p") as AM_PM;
```

Output:



System_Date	Current_Day	Current_Month	Year	Spelled_Date	rounded_date	Next_Friday	Truncate_Date	next_date_after_3_days	AM_PM
2025-04-28	Monday	April	2025	28th April 2025, Monday	2025-04-30	2025-05-02	01 April 2025	2025-05-01	AM

2. **Display day of date of joining column, display those employees who join the company on Monday, display those employees who join the company this month, and display those employees who join the company in last 30 days.**

Query:

```
CREATE TABLE tbl_emp_new (
        EMP_ID INT PRIMARY KEY,
        EMP_NAME VARCHAR(100),
        DOJ DATE
);
INSERT INTO tbl_emp_new (EMP_ID, EMP_NAME, DOJ)
VALUES
(1, 'Abhishek', '2025-05-19'),
(2, 'John', '2025-01-13'),
(3, 'Alice', '2025-03-18'),
(4, 'Robby', '2024-08-19'),
(5, 'Carter', '2024-01-19'),
(6, 'Henry', '2025-01-1');
```

Output:

1. SELECT EMP_ID,EMP_NAME,DOJ,DAYNAME(DOJ) AS DAYOFJOINING FROM tbl_emp_new;

```
mysql> SELECT EMP_ID,EMP_NAME,DOJ,DAYNAME(DOJ) AS
+--------+----------+------------+--------------+
| EMP_ID | EMP_NAME | DOJ        | DAYOFJOINING |
+--------+----------+------------+--------------+
|      1 | Abhishek | 2025-05-19 | Monday       |
|      2 | John     | 2025-01-13 | Monday       |
|      3 | Alice    | 2025-03-18 | Tuesday      |
|      4 | Robby    | 2024-08-19 | Monday       |
|      5 | Carter   | 2024-01-19 | Friday       |
|      6 | Henry    | 2025-01-01 | Wednesday    |
+--------+----------+------------+--------------+
6 rows in set (0.01 sec)
```

2. SELECT DAYNAME(DOJ) AS DayOfWeek FROM tbl_emp_new;

```
+-----------+
| DayOfWeek |
+-----------+
| Monday    |
| Monday    |
| Tuesday   |
| Monday    |
| Friday    |
| Wednesday |
+-----------+
6 rows in set
```

3. SELECT EMP_NAME FROM tbl_emp_new WHERE DAYNAME(DOJ)='MONDAY';

```
+----------+
| EMP_NAME |
+----------+
| Abhishek |
| John     |
| Robby    |
+----------+
3 rows in set
```

4. SELECT EMP_NAME FROM tbl_emp_new WHERE MONTHNAME(DOJ) = 'January';

```
Empty set (0.01 sec)
```

5. SELECT *FROM tbl_emp_new WHERE DOJ < CURDATE() AND DOJ>=CURDATE() - INTERVAL 30 DAY;

```
+----------+
| EMP_NAME |
+----------+
| John     |
| Carter   |
| Henry    |
+----------+
3 rows in set
```

3. **Create a table train having four columns (Train Number, date of Departure, time of departure, time of arrival) Display all the records, display those trains which arrived on PM, and display train number who are going to depart in next one hour.**

Query:

```
create table train (
        train_number INT,
        departure_date DATE,
        departure_time TIME,
        arrival_time TIME
);
insert into train (train_number, departure_date, departure_time, arrival_time)
values
        (101, '2025-02-01', '9:15:00', '2:45:00'),
        (102, '2025-02-01', '06:30:00', '4:30:00'),
        (103, '2025-02-01', '13:30:00', '04:45:00'),
        (104, '2025-02-01', '21:15:00', '12:15:00'),
        (105, '2025-01-31', '18:00:00', '13:30:00'),
        (106, '2025-01-2', '11:30:00', '10:30:00'),
        (107, '2025-01-31', '18:30:00', '14:30:00');

select * from train;

select train_number, departure_date, departure_time, arrival_time from train where
time_format(departure_time,"%p") = "PM";

SELECT train_number FROM train WHERE departure_time BETWEEN CURRENT_TIME() AND
time(date_add(now(), interval 1 Hour));
```

Output:

```
+--------------+----------------+----------------+--------------+
| train_number | departure_date | departure_time | arrival_time |
+--------------+----------------+----------------+--------------+
|          101 | 2025-02-01     | 09:15:00       | 02:45:00     |
|          102 | 2025-02-01     | 06:30:00       | 04:30:00     |
|          103 | 2025-02-01     | 13:30:00       | 04:45:00     |
|          104 | 2025-02-01     | 21:15:00       | 12:15:00     |
|          105 | 2025-01-31     | 18:00:00       | 13:30:00     |
|          106 | 2025-01-02     | 11:30:00       | 10:30:00     |
|          107 | 2025-01-31     | 18:30:00       | 14:30:00     |
+--------------+----------------+----------------+--------------+
7 rows in set (0.00 sec)
```

```
+--------------+----------------+----------------+--------------+
| train_number | departure_date | departure_time | arrival_time |
+--------------+----------------+----------------+--------------+
|          103 | 2025-02-01     | 13:30:00       | 04:45:00     |
|          104 | 2025-02-01     | 21:15:00       | 12:15:00     |
|          105 | 2025-01-31     | 18:00:00       | 13:30:00     |
|          107 | 2025-01-31     | 18:30:00       | 14:30:00     |
+--------------+----------------+----------------+--------------+
4 rows in set (0.00 sec)
```

```
Empty set (0.00 sec)
```

Assignment-4

1. **Create table emp which has the following attributes (empno, ename, job, sal, deptno) Where empno is primary key, ename is unique, job in (Prof, AP, and Lect), sal is not NULL, and deptno default is 10.**

Query:

```
CREATE TABLE tbl_emp2 (
        empno INT PRIMARY KEY,
        ename VARCHAR(50) UNIQUE,
        job VARCHAR(10) CHECK (job IN ('Prof', 'AP', 'Lect')),
        sal DECIMAL(10, 2) NOT NULL,
        deptno INT DEFAULT 10
);
```

   a. **Insert appropriate records, check error messages in case of violation and list all the constraint names for given table.**

Query:

```
INSERT INTO tbl_emp2 (empno, ename, job, sal, deptno) VALUES (1, 'John', 'Prof', 50000, 20);
INSERT INTO tbl_emp2 (empno, ename, job, sal) VALUES (2, 'Jane', 'AP', 45000);
INSERT INTO tbl_emp2 (empno, ename, job, sal, deptno) VALUES (3, 'Doe', 'Lect', 40000, 30);
INSERT INTO tbl_emp2 (empno, ename, job, sal) VALUES (4, 'Alice', 'Prof', 55000);
INSERT INTO tbl_emp2 (empno, ename, job, sal, deptno) VALUES (5, 'Bob', 'AP', 47000);

-- This will violate the UNIQUE constraint on ename
INSERT INTO tbl_emp2 (empno, ename, job, sal, deptno) VALUES (6, 'John', 'AP', 48000, 20);

-- This will violate the CHECK constraint on job
INSERT INTO tbl_emp2 (empno, ename, job, sal, deptno) VALUES (7, 'Eve', 'Manager', 60000,
40);
```

Output:

```
mysql> INSERT INTO tbl_emp2 (empno, ename, job, sal, deptno) VALUES (5, 'Bob', 'AP', 47000);
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> -- This will violate the UNIQUE constraint on ename
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO tbl_emp2 (empno, ename, job, sal, deptno) VALUES (6, 'John', 'AP', 48000, 20);
ERROR 1062 (23000): Duplicate entry 'John' for key 'tbl_emp2.ename'
mysql>
mysql> -- This will violate the CHECK constraint on job
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO tbl_emp2 (empno, ename, job, sal, deptno) VALUES (7, 'Eve', 'Manager', 60000, 40
ERROR 3819 (HY000): Check constraint 'tbl_emp2_chk_1' is violated.
mysql>
mysql> -- This will violate the NOT NULL constraint on sal
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO tbl_emp2 (empno, ename, job, sal, deptno) VALUES (8, 'Charlie', 'Lect', NULL, 50
ERROR 1048 (23000): Column 'sal' cannot be null
```

b. Check the constraint name for applied constraints?

Query:

```
SELECT constraint_name
FROM information_schema.table_constraints
WHERE table_name = 'tbl_emp2';

SELECT constraint_name, constraint_type
FROM information_schema.table_constraints
WHERE table_name = ‘tbl_emp2';
```

Output:

```
+-----------------+
| CONSTRAINT_NAME |
+-----------------+
| PRIMARY         |
| tbl_emp2_chk_1  |
| ename           |
| PRIMARY         |
| tbl_emp2_chk_1  |
+-----------------+
```

c. Drop the unique constraint on ename and Change Data type of ename.

Query:

```
ALTER TABLE tbl_emp2 DROP CONSTRAINT ename;
ALTER TABLE tbl_emp2 MODIFY COLUMN ename TYPE varchar(100);
```

Output:

```
+-----------------+-----------------+
| CONSTRAINT_NAME | CONSTRAINT_TYPE |
+-----------------+-----------------+
| PRIMARY         | PRIMARY KEY     |
| tbl_emp2_chk_1  | CHECK           |
| ename           | UNIQUE          |
| PRIMARY         | PRIMARY KEY     |
| tbl_emp2_chk_1  | CHECK           |
+-----------------+-----------------+
5 rows in set (0.00 sec)
```

2. **Create table S (Salesperson table) which has the following attributes (sno, sname, city) Where sno is primary key. Create table P (Part table) which has the following attributes (pno, pname, color) Where pno is primary key. Create table SP which has the following attributes (sno, pno qty) Where combination of (sno, pno) is primary key, also sno and pno are foreign keys.**

Query:

```
CREATE TABLE S (
        sno INT PRIMARY KEY,
        sname VARCHAR(50),
        city VARCHAR(50)
);
CREATE TABLE P (
        pno INT PRIMARY KEY,
        pname VARCHAR(50),
        color VARCHAR(20)
);
CREATE TABLE SP
        sno INT,
        pno INT,
        qty INT,
        PRIMARY KEY (sno, pno),
        FOREIGN KEY (sno) REFERENCES S(sno),
        FOREIGN KEY (pno) REFERENCES P(pno)
);
```

a. **Insert appropriate records and violate the foreign key constraint and identify the errors messages.**

Query:

```
-- Insert appropriate records and violate the foreign key constraint
INSERT INTO S (sno, sname, city) VALUES (1, 'Alice', 'New York');
INSERT INTO P (pno, pname, color) VALUES (1, 'Widget', 'Red');
INSERT INTO SP (sno, pno, qty) VALUES (1, 1, 50);

-- This will violate the foreign key constraint on sno
INSERT INTO SP (sno, pno, qty) VALUES (2, 1, 100);

-- This will violate the foreign key constraint on pno
INSERT INTO SP (sno, pno, qty) VALUES (1, 2, 200);
```

Output:

```
mysql> -- This will violate the foreign key constraint on sno
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO SP (sno, pno, qty) VALUES (2, 1, 100);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`practical_db`.`sp`, CONSTRAINT `sp_ibfk_1` FOREIGN KEY (`sn
o`) REFERENCES `s` (`sno`))
mysql>
mysql> -- This will violate the foreign key constraint on pno
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO SP (sno, pno, qty) VALUES (1, 2, 200);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`practical_db`.`sp`, CONSTRAINT `sp_ibfk_2` FOREIGN KEY (`pn
o`) REFERENCES `p` (`pno`))
```

b. Drop the Foreign Key constraint on sno and pno.

Query:

ALTER TABLE SP DROP CONSTRAINT sp_ibfk_1;
ALTER TABLE SP DROP CONSTRAINT sp_ibfk_2;

Output:

```
mysql> ALTER TABLE SP DROP CONSTRAINT sp_ibfk_1;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE SP DROP CONSTRAINT sp_ibfk_2;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Assignment-5

Create table and insert appropriate records.
S (Salesperson table) (Sno, Sname, City)
P (Part table) (Pno, Pname, Color)
SP(Sno, Pno, Qty)

1. **Identify primary and foreign keys in each of these tables and create all the tables.**
2. **Violate the foreign key constraint and identify the oracle errors.**
3. **Delete record of supplier S1 from S table and write a note on the output of query.**
4. **Get Qty supplied for Red parts and get Sname supplying Red part.**
5. **For each part supplied get part number and names of all cities supplying the part.**
6. **Get all pairs of suppliers numbers such that the two suppliers are located in the same city.**
7. **Drop table SP and create another table SPl with delete cascade foreign key option.**
8. **Delete record of supplier S1 from S table and write a note on the output of query.**
9. **Drop table SP1 and create another table SP2 with delete set null option of foreign key.**
10. **Delete record of supplier S1 from S table and write a note on the output of query.**
11. **Alter table SP2 and drop existing foreign key constraint.**
12. **Add foreign key constraint with alter table option with on delete set default option.**

Query:

```
CREATE TABLE S (
   Sno INT PRIMARY KEY,
   Sname VARCHAR(50),
   City VARCHAR(100)
);
CREATE TABLE P (
   Pno INT PRIMARY KEY,
   Pname VARCHAR(50),
   Color VARCHAR(100)
);
CREATE TABLE SP (
   Sno INT,
   Pno INT,
   Qty INT,
   PRIMARY KEY (Sno, Pno),
   FOREIGN KEY (Sno) REFERENCES S(Sno),
   FOREIGN KEY (Pno) REFERENCES P(Pno)
);

INSERT INTO S (sno, sname, city) VALUES (1, 'Alice', 'New York');
INSERT INTO P (pno, pname, color) VALUES (1, 'Widget', 'Red');

SELECT constraint_name
FROM information_schema.table_constraints
WHERE table_name = 'S' or table_name='P' or table_name='SP';

INSERT INTO SP (sno, pno, qty) VALUES (1, 1, 50);
```

```
-- This will violate the foreign key constraint on sno
INSERT INTO SP (sno, pno, qty) VALUES (2, 1, 100);

-- This will violate the foreign key constraint on pno
INSERT INTO SP (sno, pno, qty) VALUES (1, 2, 200);

DELETE from S WHERE Sno=1;

Select Sname,Qty from S join SP where SP.Pno = (Select Pno from P where Color='red');
SELECT SP.Pno, S.City
FROM SP
JOIN S ON SP.Sno = S.Sno;

SELECT S1.Sno AS Supplier1, S2.Sno AS Supplier2
FROM S S1, S S2
WHERE S1.City = S2.City AND S1.Sno < S2.Sno;

DROP TABLE SP;
CREATE TABLE SPl (
   Sno INT,
   Pno INT,
   Qty INT,
   PRIMARY KEY (Sno, Pno),
   FOREIGN KEY (Sno) REFERENCES S(Sno) ON DELETE CASCADE,
   FOREIGN KEY (Pno) REFERENCES P(Pno) ON DELETE CASCADE
);

DELETE FROM S WHERE Sno = 1;

DROP TABLE SPl;

CREATE TABLE SP2 (
   Sno INT,
   Pno INT,
   Qty INT,
   PRIMARY KEY (Sno, Pno),
   FOREIGN KEY (Sno) REFERENCES S(Sno) ON DELETE SET NULL,
   FOREIGN KEY (Pno) REFERENCES P(Pno) ON DELETE SET NULL
);

DELETE FROM S WHERE Sno = 1;

SELECT CONSTRAINT_NAME
FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE TABLE_NAME = 'SP2';

ALTER TABLE SP2 DROP FOREIGN KEY sp2;

ALTER TABLE SP2 MODIFY Sno INT DEFAULT 0;

ALTER TABLE SP2
ADD CONSTRAINT fk_sno
FOREIGN KEY (Sno) REFERENCES S(Sno)
ON DELETE SET DEFAULT;
```

Output:

```
mysql> SELECT constraint_name
    -> FROM information_schema.table_constraints
    -> WHERE table_name = 'S' or table_name='P' or table_name='SP';
+-----------------+
| CONSTRAINT_NAME |
+-----------------+
| PRIMARY         |
| PRIMARY         |
| PRIMARY         |
| PRIMARY         |
| PRIMARY         |
| PRIMARY         |
| sp_ibfk_1       |
| sp_ibfk_2       |
| PRIMARY         |
| s_chk_1         |
| Pname           |
| PRIMARY         |
| PRIMARY         |
| sp_ibfk_1       |
| sp_ibfk_2       |
| sp_chk_1        |
| PRIMARY         |
| PRIMARY         |
| PRIMARY         |
| sp_ibfk_1       |
| sp_ibfk_2       |
+-----------------+
21 rows in set (0.02 sec)

mysql>
mysql> INSERT INTO SP (sno, pno, qty) VALUES (1, 1, 50);
Query OK, 1 row affected (0.00 sec)

mysql> -- This will violate the foreign key constraint on sno
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO SP (sno, pno, qty) VALUES (2, 1, 100);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`practical_db`.`sp`, CONSTRAINT `sp_ibfk_1` FOREIGN KEY (`Sno`) REFERENCES `s` (`Sno`))
mysql>
mysql> -- This will violate the foreign key constraint on pno
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO SP (sno, pno, qty) VALUES (1, 2, 200);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`practical_db`.`sp`, CONSTRAINT `sp_ibfk_2` FOREIGN KEY (`Pno`) REFERENCES `p` (`Pno`))
mysql>
mysql> DELETE from S WHERE Sno=1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`practical_db`.`sp`, CONSTRAINT `sp_ibfk_1` FOREIGN KEY (`Sno`) REFERENCES `s` (`Sno`))
mysql>
mysql> Select Sname,Qty from S join SP where SP.Pno = (Select Pno from P where Color='Red');
+-------+------+
| Sname | Qty  |
+-------+------+
| Alice |   50 |
+-------+------+
```

```
mysql> SELECT SP.Pno, S.City
    -> FROM SP
    -> JOIN S ON SP.Sno = S.Sno;
+------+----------+
| Pno | City     |
+------+----------+
|   1 | New York |
+------+----------+
1 row in set (0.00 sec)

mysql>
mysql> SELECT S1.Sno AS Supplier1, S2.Sno AS Supplier2
    -> FROM S S1, S S2
    -> WHERE S1.City = S2.City AND S1.Sno < S2.Sno;
Empty set (0.00 sec)

mysql>
mysql> DROP TABLE SP;
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> CREATE TABLE SP1 (
    ->     Sno INT,
    ->     Pno INT,
    ->     Qty INT,
    ->     PRIMARY KEY (Sno, Pno),
    ->     FOREIGN KEY (Sno) REFERENCES S(Sno) ON DELETE CASCADE,
    ->     FOREIGN KEY (Pno) REFERENCES P(Pno) ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELETE FROM S WHERE Sno = 1;
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> DROP TABLE SP1;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> CREATE TABLE SP2 (
    ->     Sno INT NULL,
    ->     Pno INT NULL,
    ->     Qty INT,
    ->     PRIMARY KEY (Sno, Pno),
    ->     FOREIGN KEY (Sno) REFERENCES S(Sno) ON DELETE SET NULL,
    ->     FOREIGN KEY (Pno) REFERENCES P(Pno) ON DELETE SET NULL
    -> );
ERROR 1171 (42000): All parts of a PRIMARY KEY must be NOT NULL; if you need NULL in a key, use UNIQUE instead
mysql> CREATE TABLE SP2 (
    ->     Sno INT NULL,
    ->     Pno INT NULL,
    ->     Qty INT,
    ->     UNIQUE (Sno, Pno),
    ->     FOREIGN KEY (Sno) REFERENCES S(Sno) ON DELETE SET NULL,
    ->     FOREIGN KEY (Pno) REFERENCES P(Pno) ON DELETE SET NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> DELETE FROM S WHERE Sno = 1;
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> SELECT CONSTRAINT_NAME
    -> FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
    -> WHERE TABLE_NAME = 'SP2';
+-----------------+
| CONSTRAINT_NAME |
+-----------------+
| Sno             |
| Sno             |
| sp2_ibfk_1      |
| sp2_ibfk_2      |
+-----------------+
4 rows in set (0.00 sec)

mysql> ALTER TABLE SP2 DROP FOREIGN KEY sp2;
ERROR 1091 (42000): Can't DROP 'sp2'; check that column/key exists
mysql>
mysql> ALTER TABLE SP2 MODIFY Sno INT DEFAULT 0;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE SP2
    -> ADD CONSTRAINT fk_sno
    -> FOREIGN KEY (Sno) REFERENCES S(Sno)
    -> ON DELETE SET DEFAULT;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Assignment-6

Create table and Insert appropriate records.

- **S(Sno, Sname, City, Status)**
- **P(Pno. Pname, Color, Weight)**
- **SP(Sno, Pno, Qty)**

a) Apply Column level constraints on City such that it may be only "Amritsar", "Delhi", "Batala" and "Qadian".
b) Apply Column level constraints such that Qty should be between 100 to 1000, Weight is NOT NULL, Sname is NOT NULL and Pname is UNIQUE.
c) Get supplier name who supplies at least one red part.
d) Get supplier number who supplies maximum quantity.
e) Get supplier number who supply quantity greater than average quantity.
f) Increase the Quantity of part P1 by 10%.
g) Get total number of parts supplied by supplier S1.
h) Count the parts having red color.
i) Count red parts supplied by Ajay. j Get the total quantity supplied by S1./
j) Change the color of Red parts supplied by Ajay to Green.
k) Delete all the entries of part Pl.

Query:

```
CREATE TABLE S (
   Sno INT PRIMARY KEY,
   Sname VARCHAR(50) NOT NULL,
   City VARCHAR(50) CHECK (City in ('Amritsar','Delhi','Batala','Qadian')),
   Status  VARCHAR(50)
);

CREATE TABLE P (
   Pno INT PRIMARY KEY,
   Pname VARCHAR(50) UNIQUE,
   Color VARCHAR(20),
   Weight FLOAT NOT NULL
);

CREATE TABLE SP (
   Sno INT,
   Pno INT,
   Qty INT CHECK (Qty>100 and Qty<1000),
   PRIMARY KEY (Sno, Pno),
   FOREIGN KEY (Sno) REFERENCES S(Sno),
   FOREIGN KEY (Pno) REFERENCES P(Pno)
);
```

```
-- Insert values into S table

INSERT INTO S (Sno, Sname, City, Status) VALUES
(1, 'Supplier1', 'Amritsar', 'Active'),
(2, 'Supplier2', 'Delhi', 'Inactive'),
(3, 'Supplier3', 'Batala', 'Active'),
(4, 'Supplier4', 'Qadian', 'Inactive'),
(5, 'Supplier5', 'Amritsar', 'Active'),
(6, 'Supplier6', 'Delhi', 'Inactive'),
(7, 'Supplier7', 'Batala', 'Active'),
(8, 'Supplier8', 'Qadian', 'Inactive'),
(9, 'Supplier9', 'Amritsar', 'Active'),
(10, 'Supplier10', 'Delhi', 'Inactive');

-- Insert values into P table

INSERT INTO P (Pno, Pname, Color, Weight) VALUES
(1, 'Part1', 'Red', 12.5),
(2, 'Part2', 'Blue', 15.0),
(3, 'Part3', 'Green', 10.0),
(4, 'Part4', 'Yellow', 20.0),
(5, 'Part5', 'Black', 25.0),
(6, 'Part6', 'White', 30.0),
(7, 'Part7', 'Purple', 35.0),
(8, 'Part8', 'Orange', 40.0),
(9, 'Part9', 'Pink', 45.0),
(10, 'Part10', 'Brown', 50.0);

-- Insert values into SP table

INSERT INTO SP (Sno, Pno, Qty) VALUES
(1, 1, 150),
(1, 2, 200),
(2, 3, 250),
(2, 4, 300),
(3, 5, 350),
(3, 6, 400),
(4, 7, 450),
(4, 8, 500),
(5, 9, 550),
(5, 10, 600);

-- Question c
SELECT Sname FROM SP Inner join P on SP.Pno=P.Pno Inner join S on SP.Sno=S.Sno where
P.Color='Red' and S.Sname='Ajay';

INSERT INTO P (Pno, Pname, Color, Weight) VALUES
(13, 'Part13', 'Red', 12.5);

INSERT INTO S (Sno, Sname, City, Status) VALUES
(13, 'Ajay', 'Amritsar', 'Active');

INSERT INTO SP (Sno, Pno, Qty) VALUES
(13, 13, 420);
```

```
-- Question d
SELECT S.Sname,sum(SP.Qty) as TotalQty FROM SP Inner join P on SP.Pno=P.Pno Inner join S
on SP.Sno=S.Sno group by S.Sno order by sum(SP.Qty) DESC Limit 1;


-- Question e
-- Get supplier numbers who supply quantity greater than average quantity
SELECT SP.Sno FROM SP GROUP BY SP.Sno HAVING SUM(SP.Qty) > (SELECT
AVG(SP.Qty) FROM SP);


-- Question f
-- Increase the quantity of part P1 by 10%
UPDATE SP SET Qty = Qty * 1.10 WHERE Pno = 1;


-- Question g
-- Get total number of parts supplied by supplier S1
SELECT Count(Pno) AS TotalParts FROM SP WHERE Sno = 1;


-- Question h
-- Count the parts having red color
SELECT Count(Pno) FROM P where Color='Red';


-- Question i
-- Count the red parts supplied by Ajay
SELECT Count(SP.Sno) FROM SP Inner join P on SP.Pno=P.Pno Inner join S on SP.Sno=S.Sno
where P.Color='Red' and S.Sname='Ajay';


-- Question j
-- Total Quantity Supplied by S1
SELECT sum(SP.Qty) as TotalQty FROM SP Inner join S on SP.Sno=S.Sno where SP.Sno=1;


-- Question k
UPDATE P SET Color = 'Green' WHERE Pno IN (SELECT Pno FROM SP INNER JOIN S ON
S.Sno=SP.Sno where S.Sname='Ajay' and P.Color='Red');


-- Question l
DELETE from SP where Pno=1;
```

Output:

```
mysql> -- Question c
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT Sname FROM SP Inner join P on SP.Pno=P.Pno Inner join S on SP.Sno=S.Sno where P.Col
Empty set (0.00 sec)

mysql>
mysql> INSERT INTO P (Pno, Pname, Color, Weight) VALUES
    -> (13, 'Part13', 'Red', 12.5);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO S (Sno, Sname, City, Status) VALUES
    -> (13, 'Ajay', 'Amritsar', 'Active');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO SP (Sno, Pno, Qty) VALUES
    -> (13, 13, 420);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> -- Question d
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT S.Sname,sum(SP.Qty) as TotalQty FROM SP Inner join P on SP.Pno=P.Pno Inner join S o
+-----------+----------+
| Sname     | TotalQty |
+-----------+----------+
| Supplier5 |     1150 |
+-----------+----------+
1 row in set (0.01 sec)

mysql>
mysql> -- Question e
Query OK, 0 rows affected (0.00 sec)

mysql> -- Get supplier numbers who supply quantity greater than average quantity
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT SP.Sno FROM SP GROUP BY SP.Sno HAVING SUM(SP.Qty) > (SELECT AVG(SP.Qty) FROM SP);
+-----+
| Sno |
+-----+
|   2 |
|   3 |
|   4 |
|   5 |
|  13 |
+-----+
5 rows in set (0.00 sec)
```

```
mysql> SELECT Count(Pno) AS TotalParts FROM SP WHERE Sno = 1;
+------------+
| TotalParts |
+------------+
|          2 |
+------------+
1 row in set (0.00 sec)

mysql>
mysql> -- Question h
Query OK, 0 rows affected (0.00 sec)

mysql> -- Count the parts having red color
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT Count(Pno) FROM P where Color='Red';
+------------+
| Count(Pno) |
+------------+
|          2 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT Count(SP.Sno) FROM SP
+---------------+
| Count(SP.Sno) |
+---------------+
|             1 |
+---------------+
1 row in set (0.00 sec)

mysql>
mysql> -- Question j
Query OK, 0 rows affected (0.00 sec

mysql> -- Total Quantity Supplied by
Query OK, 0 rows affected (0.00 sec

mysql> SELECT sum(SP.Qty) as TotalQt
+----------+
| TotalQty |
+----------+
|      365 |
+----------+
1 row in set (0.00 sec)
```

Assignment-7

1. **Write a PL/SQL block to find the sum of two numbers.**

Query:

```
DELIMITER //

CREATE PROCEDURE calculate_sum()
BEGIN
   DECLARE num1 INT DEFAULT 10;
   DECLARE num2 INT DEFAULT 20;
   DECLARE sum INT;

   SET sum = num1 + num2;
   SELECT CONCAT('The sum of ', num1, ' and ', num2, ' is ', sum) AS result;
END //

DELIMITER ;
CALL calculate_sum();
```

Output:

```
+---------------------------+
| result                    |
+---------------------------+
| The sum of 10 and 20 is 30 |
+---------------------------+
1 row in set (0.00 sec)
```

2. **Write a PL/SQL block to find the larger of two numbers.**

Query:

```
DELIMITER //
CREATE PROCEDURE find_larger()
BEGIN
   DECLARE num1 INT DEFAULT 10;
   DECLARE num2 INT DEFAULT 20;
   DECLARE larger INT;

   IF num1 > num2 THEN
      SET larger = num1;
   ELSE
      SET larger = num2;
   END IF;

   SELECT CONCAT('The larger number between ', num1, ' and ', num2, ' is ', larger) AS result;
END //
DELIMITER ;

CALL find_larger();
```

Output:

```
+-----------------------------------------+
| result                                  |
+-----------------------------------------+
| The larger number between 10 and 20 is 20 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

3. Write a PL/SQL block to find the greatest of three numbers.

Query:

```
DELIMITER //
CREATE PROCEDURE find_largest()
BEGIN
   DECLARE num1 INT DEFAULT 10;
   DECLARE num2 INT DEFAULT 20;
   DECLARE num3 INT DEFAULT 30;
   DECLARE largest INT;

   IF num1 > num2 AND num1 > num3 THEN
      SET largest = num1;
   ELSEIF num2 > num1 AND num2 > num3 THEN
      SET largest = num2;
   ELSE
      SET largest = num3;
   END IF;

   SELECT CONCAT('The largest number among ', num1, ', ', num2, ' and ', num3, ' is ', largest) AS
result;
END //
DELIMITER ;

CALL find_largest();
```

Output:

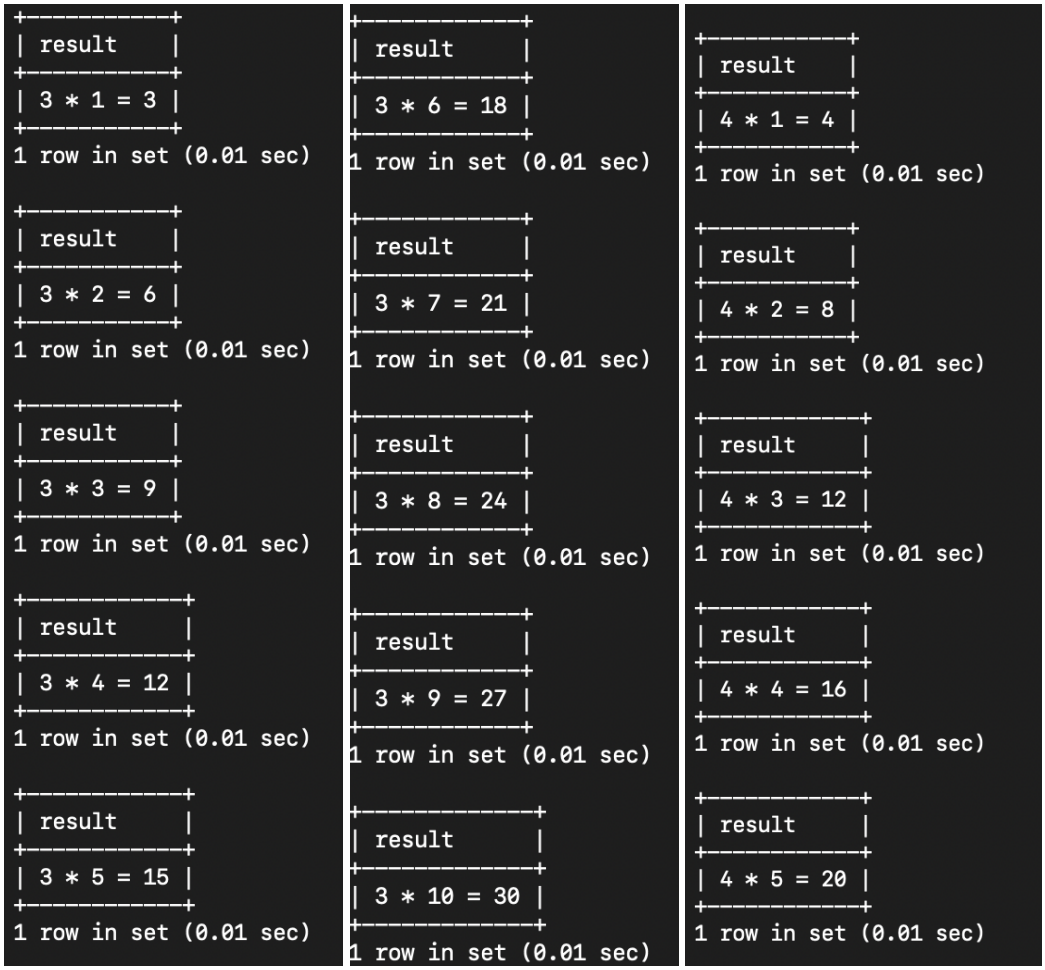
```
+-----------------------------------------+
| result                                  |
+-----------------------------------------+
| The largest number among 10, 20 and 30 is 30 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

4. Write a PL/SQL block to generate multiplication table for 3 to n.

Query:

```
DELIMITER //
CREATE PROCEDURE generate_multiplication_table(IN n INT)
BEGIN
   DECLARE i INT DEFAULT 3;
   DECLARE j INT;

   WHILE i <= n DO
      SET j = 1;
      WHILE j <= 10 DO
         SELECT CONCAT(i, ' * ', j, ' = ', i * j) AS result;
         SET j = j + 1;
      END WHILE;
      SET i = i + 1;
   END WHILE;
END //
DELIMITER ;

CALL generate_multiplication_table(5);
```

Output:



```
+-----------+
| result    |
+-----------+
| 3 * 1 = 3 |
+-----------+
1 row in set (0.01 sec)

+-----------+
| result    |
+-----------+
| 3 * 2 = 6 |
+-----------+
1 row in set (0.01 sec)

+-----------+
| result    |
+-----------+
| 3 * 3 = 9 |
+-----------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 3 * 4 = 12 |
+------------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 3 * 5 = 15 |
+------------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 3 * 6 = 18 |
+------------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 3 * 7 = 21 |
+------------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 3 * 8 = 24 |
+------------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 3 * 9 = 27 |
+------------+
1 row in set (0.01 sec)

+--------------+
| result       |
+--------------+
| 3 * 10 = 30  |
+--------------+
1 row in set (0.01 sec)

+-----------+
| result    |
+-----------+
| 4 * 1 = 4 |
+-----------+
1 row in set (0.01 sec)

+-----------+
| result    |
+-----------+
| 4 * 2 = 8 |
+-----------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 4 * 3 = 12 |
+------------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 4 * 4 = 16 |
+------------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 4 * 5 = 20 |
+------------+
1 row in set (0.01 sec)
```

```
+-------------+
| result      |
+-------------+
| 4 * 6 = 24 |
+-------------+
1 row in set (0.01 sec)

+-------------+
| result      |
+-------------+
| 4 * 7 = 28 |
+-------------+
1 row in set (0.01 sec)

+-------------+
| result      |
+-------------+
| 4 * 8 = 32 |
+-------------+
1 row in set (0.01 sec)

+-------------+
| result      |
+-------------+
| 4 * 9 = 36 |
+-------------+
1 row in set (0.01 sec)

+--------------+
| result       |
+--------------+
| 4 * 10 = 40 |
+--------------+
1 row in set (0.01 sec)
```

```
+-------------+
| result      |
+-------------+
| 5 * 1 = 5  |
+-------------+
1 row in set (0.01 sec)

+-------------+
| result      |
+-------------+
| 5 * 2 = 10 |
+-------------+
1 row in set (0.01 sec)

+-------------+
| result      |
+-------------+
| 5 * 3 = 15 |
+-------------+
1 row in set (0.01 sec)

+-------------+
| result      |
+-------------+
| 5 * 4 = 20 |
+-------------+
1 row in set (0.01 sec)

+-------------+
| result      |
+-------------+
| 5 * 5 = 25 |
+-------------+
1 row in set (0.01 sec)
```

```
+------------+
| result     |
+------------+
| 5 * 6 = 30 |
+------------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 5 * 7 = 35 |
+------------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 5 * 8 = 40 |
+------------+
1 row in set (0.01 sec)

+------------+
| result     |
+------------+
| 5 * 9 = 45 |
+------------+
1 row in set (0.01 sec)

+-------------+
| result      |
+-------------+
| 5 * 10 = 50 |
+-------------+
1 row in set (0.01 sec)

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

Assignment-8

1. **Write a PL/SQL block to update total sal for empno 100.Eno, ename, bp, da, hra, total.**
2. **Write a PL/SQL block to update total sal for all employees. Eno, ename, bp. da, hra, total.**
3. **Add exception handlers in PL/SQL block created earlier to update total sal for empno 100. (Eno,ename, bp,da hra, total.)**
4. **Add exception handlers in a block which contain an insert statement for the following errors:**
   - **violation of check constraint,**
   - **violation of primary key,**
   - **violation of foreign key,**
   - **violation of supplied value larger than the specified width of the column.**
5. **Write a PL/SQL in which user defined exception is trapped when date of issue is greater than date of return. (mo, bookno, doi, dor).**

Queries:

```
CREATE TABLE employees (
   empno INT PRIMARY KEY,
   ename VARCHAR(50),
   bp DECIMAL(10,2),
   da DECIMAL(10,2),
   hra DECIMAL(10,2),
   total DECIMAL(10,2)
);

INSERT INTO employees (empno, ename, bp, da, hra, total)
VALUES
(100, 'Alice', 5000, 2000, 1000, NULL),
(101, 'Bob', 6000, 2500, 1200, NULL),
(102, 'Charlie', 5500, 2200, 1100, NULL),
(103, 'David', 7000, 3000, 1500, NULL),
(104, 'Eve', 4800, 1900, 900, NULL),
(105, 'Frank', 6200, 2600, 1300, NULL),
(106, 'Grace', 5300, 2100, 1050, NULL),
(107, 'Hank', 7500, 3200, 1600, NULL),
(108, 'Ivy', 4700, 1800, 850, NULL),
(109, 'Jack', 6800, 2700, 1400, NULL);

CREATE TABLE books (
   mo INT,
   bookno INT PRIMARY KEY,
   doi DATE,
   dor DATE
);
```

```
INSERT INTO books (mo, bookno, doi, dor)
VALUES
(1, 101, '2023-10-01', '2023-10-10'),
(2, 102, '2023-10-05', '2023-10-15'),
(3, 103, '2023-10-08', '2023-10-18'),
(4, 104, '2023-10-12', '2023-10-20'),
(5, 105, '2023-10-15', '2023-10-25'),
(6, 106, '2023-10-18', '2023-10-28'),
(7, 107, '2023-10-20', '2023-10-30'),
(8, 108, '2023-10-22', '2023-11-01'),
(9, 109, '2023-10-25', '2023-11-05'),
(10, 110, '2023-10-28', '2023-11-08');
```

-- 1. Update total salary for empno 100

```
DELIMITER //
CREATE PROCEDURE UpdateTotalSalaryForEmp100()
BEGIN
   DECLARE v_bp DECIMAL(10,2);
   DECLARE v_da DECIMAL(10,2);
   DECLARE v_hra DECIMAL(10,2);
   DECLARE v_total DECIMAL(10,2);

   SELECT bp, da, hra INTO v_bp, v_da, v_hra
   FROM employees
   WHERE empno = 100;

   SET v_total = v_bp + v_da + v_hra;

   UPDATE employees
   SET total = v_total
   WHERE empno = 100;

   SELECT 'Total salary updated for empno 100.' AS Message;
END;
// DELIMITER ;
```

Output:

```
mysql> CALL UpdateTotalSalaryForEmp100();
+-----------------------------------+
| Message                           |
+-----------------------------------+
| Total salary updated for empno 100. |
+-----------------------------------+
1 row in set (0.00 sec)
```

-- 2. Update total salary for all employees

```
DELIMITER //
CREATE PROCEDURE UpdateTotalSalaryForAllEmployees()
BEGIN
   DECLARE done INT DEFAULT 0;
   DECLARE v_empno INT;
   DECLARE v_bp DECIMAL(10,2);
   DECLARE v_da DECIMAL(10,2);
   DECLARE v_hra DECIMAL(10,2);
   DECLARE v_total DECIMAL(10,2);
   DECLARE emp_cursor CURSOR FOR SELECT empno, bp, da, hra FROM
employees;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

   OPEN emp_cursor;

   emp_loop: LOOP
      FETCH emp_cursor INTO v_empno, v_bp, v_da, v_hra;
      IF done THEN
         LEAVE emp_loop;
      END IF;

      SET v_total = v_bp + v_da + v_hra;

      UPDATE employees
      SET total = v_total
      WHERE empno = v_empno;
   END LOOP;

   CLOSE emp_cursor;

   SELECT 'Total salary updated for all employees.' AS Message;
END;
//
DELIMITER ;
```

Output:

```
mysql> CALL UpdateTotalSalaryForAllEmployees();
+--------------------------------------+
| Message                              |
+--------------------------------------+
| Total salary updated for all employees. |
+--------------------------------------+
1 row in set (0.00 sec)
```

-- 3. Add exception handlers to update total salary for empno 100

```
DELIMITER //
CREATE PROCEDURE UpdateTotalSalaryForEmp100WithHandlers(IN id INT)
BEGIN
   DECLARE v_bp DECIMAL(10,2);
   DECLARE v_da DECIMAL(10,2);
   DECLARE v_hra DECIMAL(10,2);
   DECLARE v_total DECIMAL(10,2);
   DECLARE v_count INT;
   DECLARE error_message VARCHAR(255);

   -- Check if the employee exists
   SELECT COUNT(*) INTO v_count FROM employees WHERE empno = id;
   IF v_count = 0 THEN
      -- Construct the error message
      SET error_message = CONCAT('Employee with empno ', id, ' does not exist.');
      -- Raise an error with the constructed message
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = error_message;
   ELSE
      -- Fetch salary components and update total salary
      SELECT bp, da, hra INTO v_bp, v_da, v_hra
      FROM employees
      WHERE empno = id;
      SET v_total = v_bp + v_da + v_hra;
      UPDATE employees
      SET total = v_total
      WHERE empno = id;

      SELECT CONCAT('Total salary updated for empno ', id, '.') AS Message;
   END IF;
END;
//
DELIMITER ;

-- Call the procedure
CALL UpdateTotalSalaryForEmp100WithHandlers(100);
```

Output:

```
mysql> CALL UpdateTotalSalaryForAllEmployees();
+--------------------------------------+
| Message                              |
+--------------------------------------+
| Total salary updated for all employees. |
+--------------------------------------+
1 row in set (0.00 sec)
```

-- 4. Exception handlers for insert statement

```
DELIMITER //
CREATE PROCEDURE InsertEmployeeWithHandlers(IN id INT)
BEGIN
   DECLARE v_count INT;
   DECLARE error_message VARCHAR(255);

   -- Check if the employee already exists
   SELECT COUNT(*) INTO v_count FROM employees WHERE empno = id;

   IF v_count > 0 THEN
      -- Construct the error message
      SET error_message = CONCAT('Employee with empno ', id, ' already exists.');

      -- Raise an error with the constructed message
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = error_message;
   ELSE
      -- Insert the new employee if they do not exist
      INSERT INTO employees (empno, ename, bp, da, hra, total)
      VALUES (id, 'John Doe', 5000, 2000, 1000, NULL);

      SELECT 'Record inserted successfully.' AS Message;
   END IF;
END;
//
DELIMITER ;
-- Call the procedure
CALL InsertEmployeeWithHandlers(111);
CALL InsertEmployeeWithHandlers(100); -- This will raise an error
```

Output:

```
mysql> CALL InsertEmployeeWithHandlers(111);
+-----------------------------+
| Message                     |
+-----------------------------+
| Record inserted successfully. |
+-----------------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> CALL InsertEmployeeWithHandlers(100);
ERROR 1644 (45000): Employee with empno 100 already exists.
mysql>
```

-- 5. User-defined exception for date of issue greater than date of return

```
DELIMITER //
CREATE PROCEDURE InsertBookWithDateValidation()
BEGIN
   DECLARE v_doi DATE DEFAULT '2023-10-10';
   DECLARE v_dor DATE DEFAULT '2023-10-05';

   IF v_doi > v_dor THEN
      -- Raise an error if the date of issue is greater than the date of return
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Date of issue cannot be greater than date of return.';
   ELSE
      -- Insert the book record if the dates are valid
      IF EXISTS (SELECT 1 FROM books WHERE bookno = 101) THEN
         SIGNAL SQLSTATE '45000'
         SET MESSAGE_TEXT = 'Book number already exists.';
      END IF;

      INSERT INTO books (mo, bookno, doi, dor)
      VALUES (1, 101, v_doi, v_dor);

      SELECT 'Record inserted successfully.' AS Message;
   END IF;
END;
//
DELIMITER ;

-- Call the procedure
CALL InsertBookWithDateValidation();
```

Output:

```
mysql> CALL InsertBookWithDateValidation();
ERROR 1644 (45000): Date of issue cannot be greater than date of return.
mysql>
```

Assignment-9

1. **WAP to print name of employees belonging to deptno 10.**
2. **WAP to find top 5 highest paid employees**
3. **Perform Q1 & 2 with cursor for loop.**
4. **Write a PL/SQL block that calculate simple interest for principal 1000, time 2 years and rate of interest varies from 5 to 15. Store the computed information in following table: Principal | time | rate | interest**
5. **Write a PL/SQL block to print name of employees belonging of a particular department, the user will supply the value of deptno during run time.**

Queries:

```
CREATE TABLE employees (
   empno INT PRIMARY KEY,
   ename VARCHAR(50),
   deptno INT,
   sal DECIMAL(10,2),
   job VARCHAR(50)
);
INSERT INTO employees (empno, ename, deptno, sal, job)
VALUES
(1, 'Alice', 10, 50000.00, 'Manager'),
(2, 'Bob', 20, 40000.00, 'Developer'),
(3, 'Charlie', 10, 45000.00, 'Analyst'),
(4, 'David', 30, 60000.00, 'Team Lead'),
(5, 'Eve', 20, 55000.00, 'Developer'),
(6, 'Frank', 10, 70000.00, 'Manager'),
(7, 'Grace', 30, 35000.00, 'Tester'),
(8, 'Hank', 10, 80000.00, 'Director'),
(9, 'Ivy', 20, 30000.00, 'Intern'),
(10, 'Jack', 30, 65000.00, ‘Architect');
```

--1. Print the name of employees belonging to deptno = 10:

```
SELECT ename AS 'Employee Name'
FROM employees
WHERE deptno = 10;
```

Output:

```
+---------------+
| Employee Name |
+---------------+
| Alice         |
| Charlie       |
| Frank         |
| Hank          |
+---------------+
```

--2. Find the top 5 highest-paid employees:

```
SELECT ename AS 'Employee Name', sal AS 'Salary'
FROM employees
ORDER BY sal DESC
LIMIT 5;
```

Output:

```
+---------------+----------+
| Employee Name | Salary   |
+---------------+----------+
| Hank          | 80000.00 |
| Frank         | 70000.00 |
| Jack          | 65000.00 |
| David         | 60000.00 |
| Eve           | 55000.00 |
+---------------+----------+
```

--3. Perform Q1 & Q2 with a cursor FOR loop:
--Q1 with Cursor:

```
DELIMITER //

CREATE PROCEDURE PrintEmployeesByDept10()
BEGIN
   DECLARE emp_name VARCHAR(50);
   DECLARE done INT DEFAULT 0;

   DECLARE emp_cursor CURSOR FOR
      SELECT ename FROM employees WHERE deptno = 10;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

   OPEN emp_cursor;

   emp_loop: LOOP
      FETCH emp_cursor INTO emp_name;
      IF done THEN
         LEAVE emp_loop;
      END IF;

      SELECT CONCAT('Employee Name: ', emp_name) AS Output;
   END LOOP;

   CLOSE emp_cursor;
END;
```

```
//

DELIMITER ;

Call PrintEmployeesByDept10();
```

Output:

```
+----------------------+
| Output               |
+----------------------+
| Employee Name: Alice |
+----------------------+
1 row in set (0.00 sec)

+------------------------+
| Output                 |
+------------------------+
| Employee Name: Charlie |
+------------------------+
1 row in set (0.00 sec)

+----------------------+
| Output               |
+----------------------+
| Employee Name: Frank |
+----------------------+
1 row in set (0.00 sec)

+--------------------+
| Output             |
+--------------------+
| Employee Name: Hank |
+--------------------+
1 row in set (0.00 sec)

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

--Q2 with Cursor:

```
DELIMITER //

CREATE PROCEDURE PrintTop5HighestPaidEmployees()
BEGIN
   DECLARE emp_name VARCHAR(50);
   DECLARE emp_salary DECIMAL(10,2);
   DECLARE done INT DEFAULT 0;

   DECLARE emp_cursor CURSOR FOR
      SELECT ename, sal FROM employees ORDER BY sal DESC LIMIT 5;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

   OPEN emp_cursor;

   emp_loop: LOOP
      FETCH emp_cursor INTO emp_name, emp_salary;
```

```
        IF done THEN
           LEAVE emp_loop;
        END IF;

        SELECT CONCAT('Employee Name: ', emp_name, ', Salary: ', emp_salary) AS
Output;
    END LOOP;

    CLOSE emp_cursor;
END;
//

DELIMITER ;
Call PrintTop5HighestPaidEmployees();
```

Output:

```
+----------------------+
| Output               |
+----------------------+
| Employee Name: Alice |
+----------------------+
1 row in set (0.00 sec)

+------------------------+
| Output                 |
+------------------------+
| Employee Name: Charlie |
+------------------------+
1 row in set (0.00 sec)

+----------------------+
| Output               |
+----------------------+
| Employee Name: Frank |
+----------------------+
1 row in set (0.00 sec)

+--------------------+
| Output             |
+--------------------+
| Employee Name: Hank |
+--------------------+
1 row in set (0.00 sec)

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

--4. Calculate simple interest and store it in a table:

```
CREATE TABLE interest_info (
   principal DECIMAL(10,2),
   time INT,
   rate DECIMAL(5,2),
   interest DECIMAL(10,2)
);
DELIMITER //

CREATE PROCEDURE CalculateSimpleInterest()
BEGIN
   DECLARE rate INT;
   SET @principal = 1000;
   SET @time = 2;
   DELETE FROM interest_info; -- Clear the table before inserting new data
   SET rate = 5;
   WHILE rate <= 15 DO
      SET @interest = (@principal * @time * rate) / 100;c
      INSERT INTO interest_info (principal, time, rate, interest)
      VALUES (@principal, @time, rate, @interest);

      SET rate = rate + 1;
   END WHILE;
END;
//

DELIMITER ;
Call CalculateSimpleInterest();

SELECT * FROM interest_info;
```

Output:

```
+-----------+------+-------+----------+
| principal | time | rate  | interest |
+-----------+------+-------+----------+
|   1000.00 |    2 |  5.00 |   100.00 |
|   1000.00 |    2 |  6.00 |   120.00 |
|   1000.00 |    2 |  7.00 |   140.00 |
|   1000.00 |    2 |  8.00 |   160.00 |
|   1000.00 |    2 |  9.00 |   180.00 |
|   1000.00 |    2 | 10.00 |   200.00 |
|   1000.00 |    2 | 11.00 |   220.00 |
|   1000.00 |    2 | 12.00 |   240.00 |
|   1000.00 |    2 | 13.00 |   260.00 |
|   1000.00 |    2 | 14.00 |   280.00 |
|   1000.00 |    2 | 15.00 |   300.00 |
+-----------+------+-------+----------+
11 rows in set (0.00 sec)
```

--5. Print the name of employees belonging to a department supplied at runtime:

```
DELIMITER //

CREATE PROCEDURE PrintEmployeesByDept(IN deptno_input INT)
BEGIN
   DECLARE emp_name VARCHAR(50);
   DECLARE done INT DEFAULT 0;
   DECLARE emp_cursor CURSOR FOR
      SELECT ename FROM employees WHERE deptno = deptno_input;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
   OPEN emp_cursor;
   emp_loop: LOOP
      FETCH emp_cursor INTO emp_name;
      IF done THEN
         LEAVE emp_loop;
      END IF;

      SELECT CONCAT('Employee Name: ', emp_name) AS Output;
   END LOOP;
   CLOSE emp_cursor;
END;
//
DELIMITER ;
CALL PrintEmployeesByDept(10);
```

Output:

```
+----------------------+
| Output               |
+----------------------+
| Employee Name: Alice |
+----------------------+
1 row in set (0.00 sec)

+------------------------+
| Output                 |
+------------------------+
| Employee Name: Charlie |
+------------------------+
1 row in set (0.00 sec)

+----------------------+
| Output               |
+----------------------+
| Employee Name: Frank |
+----------------------+
1 row in set (0.00 sec)

+--------------------+
| Output             |
+--------------------+
| Employee Name: Hank |
+--------------------+
1 row in set (0.00 sec)

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

Assignment-10

1. **Write a stored procedure to calculate addition, subtraction, multiplication and division of two numbers. Call this procedure from a block. Write a local procedure for the same.**
2. **Write functions and procedures to perform Insert, Update, Delete and Retrieve operations on emp.**
3. **Write a trigger to implement primary key constraint and foreign key constraint.**
4. **Write a trigger such that no updation on emp can takes place on Sunday.**
5. **Write a trigger such that commission cannot be greater than sal. (eno, ename, job, sal, comm., dno)**

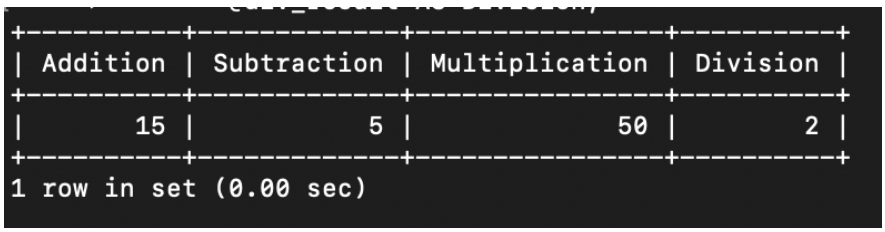
Queries:

--1. Stored Procedure for Arithmetic Operations

```
DELIMITER //

CREATE PROCEDURE ArithmeticOperations(
    IN num1 DOUBLE,
    IN num2 DOUBLE,
    OUT addition DOUBLE,
    OUT subtraction DOUBLE,
    OUT multiplication DOUBLE,
    OUT division DOUBLE
)
BEGIN
    SET addition = num1 + num2;
    SET subtraction = num1 - num2;
    SET multiplication = num1 * num2;
    IF num2 != 0 THEN
        SET division = num1 / num2;
    ELSE
        SET division = NULL; -- Handle division by zero
    END IF;
END;
//

DELIMITER ;

SHOW PROCEDURE STATUS WHERE Name = 'ArithmeticOperations';
CALL ArithmeticOperations(10, 5, @add_result, @sub_result, @mul_result,
@div_result);

SELECT @add_result AS Addition,
       @sub_result AS Subtraction,
```

```
    @mul_result AS Multiplication,
    @div_result AS Division;
```

Output:



```
+----------+-------------+----------------+----------+
| Addition | Subtraction | Multiplication | Division |
+----------+-------------+----------------+----------+
|       15 |           5 |             50 |        2 |
+----------+-------------+----------------+----------+
1 row in set (0.00 sec)
```

--Local Procedure:

```
DELIMITER //

CREATE PROCEDURE LocalArithmeticOperations()
BEGIN
   DECLARE num1 DOUBLE DEFAULT 10;
   DECLARE num2 DOUBLE DEFAULT 5;
   DECLARE addition DOUBLE;
   DECLARE subtraction DOUBLE;
   DECLARE multiplication DOUBLE;
   DECLARE division DOUBLE;

   SET addition = num1 + num2;
   SET subtraction = num1 - num2;
   SET multiplication = num1 * num2;
   IF num2 != 0 THEN
      SET division = num1 / num2;
   ELSE
      SET division = NULL;
   END IF;

   SELECT addition AS Addition,
        subtraction AS Subtraction,
        multiplication AS Multiplication,
        division AS Division;
END;
//

DELIMITER ;

CALL LocalArithmeticOperations();
```

Output:

```
+----------+-------------+----------------+----------+
| Addition | Subtraction | Multiplication | Division |
+----------+-------------+----------------+----------+
|       15 |           5 |             50 |        2 |
+----------+-------------+----------------+----------+
1 row in set (0.00 sec)
```

--2. Functions and Procedures for CRUD Operations on emp

```
CREATE TABLE emp (
   eno INT PRIMARY KEY,
   ename VARCHAR(50),
   job VARCHAR(50),
   sal DOUBLE,
   comm DOUBLE,
   dno INT
);
```

--Insert Procedure

```
DELIMITER //

CREATE PROCEDURE InsertEmp(
   IN eno INT,
   IN ename VARCHAR(50),
   IN job VARCHAR(50),
   IN sal DOUBLE,
   IN comm DOUBLE,
   IN dno INT
)
BEGIN
   INSERT INTO emp (eno, ename, job, sal, comm, dno)
   VALUES (eno, ename, job, sal, comm, dno);
END;
//

DELIMITER ;
```

--Update Procedure:

```
DELIMITER //

CREATE PROCEDURE UpdateEmp(
   IN eno INT,
   IN new_sal DOUBLE
)
BEGIN
```

```
    UPDATE emp
    SET sal = new_sal
    WHERE eno = eno;
END;
//

DELIMITER ;
```

--Delete Procedure:

```
DELIMITER //

CREATE PROCEDURE DeleteEmp(
    IN eno INT
)
BEGIN
    DELETE FROM emp
    WHERE eno = eno;
END;
//

DELIMITER ;
```

--Retrieve Function:

```
DELIMITER //

CREATE FUNCTION GetEmpDetails(eno INT)
RETURNS VARCHAR(255)
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE emp_details VARCHAR(255);
    SELECT CONCAT('Name: ', ename, ', Job: ', job, ', Salary: ', sal)
    INTO emp_details
    FROM emp
    WHERE emp.eno = eno;
    RETURN emp_details;
END;
//

DELIMITER ;
```

--3. Trigger to Implement Primary Key and Foreign Key Constraints
--Primary Key Constraint:

```
DELIMITER //

CREATE TRIGGER BeforeInsertEmp
BEFORE INSERT ON emp
FOR EACH ROW
BEGIN
   DECLARE emp_exists INT;
   SELECT COUNT(*) INTO emp_exists FROM emp WHERE eno = NEW.eno;
   IF emp_exists > 0 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Primary key constraint violated: Employee number
already exists.';
   END IF;
END;
//

DELIMITER ;
```

--Foreign Key Constraint:

```
CREATE TABLE dept (
   dno INT PRIMARY KEY,
   dname VARCHAR(50)
);

DELIMITER //

CREATE TRIGGER BeforeInsertEmpDept
BEFORE INSERT ON emp
FOR EACH ROW
BEGIN
   DECLARE dept_exists INT;
   SELECT COUNT(*) INTO dept_exists FROM dept WHERE dno = NEW.dno;
   IF dept_exists = 0 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Foreign key constraint violated: Department does not
exist.';
   END IF;
END;
//

DELIMITER ;
```

--4. Trigger to Prevent Updates on Sunday

```
DELIMITER //

CREATE TRIGGER PreventUpdateOnSunday
BEFORE UPDATE ON emp
FOR EACH ROW
BEGIN
   IF DAYOFWEEK(CURDATE()) = 1 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Updates are not allowed on Sunday.';
   END IF;
END;
//

DELIMITER ;
```

--5. Trigger to Ensure Commission Cannot Be Greater Than Salary

```
DELIMITER //

CREATE TRIGGER CheckCommission
BEFORE INSERT OR UPDATE ON emp
FOR EACH ROW
BEGIN
   IF NEW.comm > NEW.sal THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Commission cannot be greater than salary.';
   END IF;
END;
//

DELIMITER ;

INSERT INTO emp (eno, ename, job, sal, comm, dno)
VALUES (1, 'John Doe', 'Manager', 5000, 6000, 10);
```

Output:

```
ERROR 1644 (45000): Commission cannot be greater than salary.
```