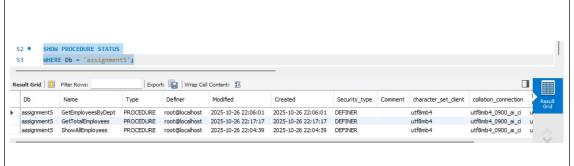
```
1. Write a SQL query to create a stored procedure without any parameters that
displays all employees from the Emp table.
CREATE PROCEDURE ShowAllEmployees()
BEGIN
  SELECT * FROM Employee;
END //
DELIMITER;
CALL ShowAllEmployees();
2. Write a SQL query to create a stored procedure with an IN parameter that
accepts a department ID and displays all employees belonging to that department.
CREATE PROCEDURE GetEmployeesByDept(IN dept id INT)
BEGIN
  SELECT EmpID, EmpName, DeptID, Salary, Job, City, JoiningDate
  FROM Employee
  WHERE DeptID = dept id;
END //
DELIMITER;
CALL GetEmployeesByDept(10);
3. Write a SQL query to create a stored procedure with an OUT parameter that
returns the total number of employees in the Emp table.
CREATE PROCEDURE GetTotalEmployees(OUT total count INT)
BEGIN
  SELECT COUNT(*) INTO total count
  FROM Employee;
END //
DELIMITER;
CALL GetTotalEmployees(@total);
SELECT @total AS TotalEmployees;
```



4. Write a SQL function that accepts an employee's salary as input and returns a grade based on the following conditions:

If salary $\geq 80,000 \rightarrow \text{Grade} = 'A'$

If salary \geq 50,000 and < 80,000 \rightarrow Grade = 'B'

If salary \geq 30,000 and < 50,000 \rightarrow Grade = 'C'

Otherwise \rightarrow Grade = 'D'

CREATE FUNCTION GetSalaryGrade(emp salary DECIMAL(10,2))

RETURNS CHAR(1)

DETERMINISTIC

BEGIN

```
DECLARE grade CHAR(1);
```

IF emp salary >= 80000 THEN

SET grade = 'A';

ELSEIF emp salary >= 50000 THEN

SET grade = 'B';

ELSEIF emp_salary >= 30000 THEN

SET grade = 'C';

ELSE

SET grade = 'D';

END IF;

RETURN grade;

```
END //
DELIMITER;
SELECT EmpName, Salary, GetSalaryGrade(Salary) AS Grade
FROM Employee;
DELIMITER //
5. Write a stored procedure that uses an explicit cursor to fetch and display the details
of all employees whose salary is greater than 60,000 from the Emp table. Make sure
to DECLARE, OPEN, FETCH, and CLOSE the cursor properly.
CREATE PROCEDURE ShowHighSalaryEmployees()
BEGIN
  --Declare variables to hold employee data
  DECLARE v EmpID INT;
  DECLARE v_EmpName VARCHAR(50);
  DECLARE v DeptID INT;
  DECLARE v_Salary DECIMAL(10,2);
  DECLARE v_Job VARCHAR(50);
  DECLARE v_City VARCHAR(50);
  DECLARE v JoiningDate DATE;
  -- Declare a flag to detect end of cursor
  DECLARE done INT DEFAULT 0;
  -- Declare the cursor
  DECLARE cur emp CURSOR FOR
    SELECT EmpID, EmpName, DeptID, Salary, Job, City, JoiningDate
    FROM Employee
    WHERE Salary > 60000;
```

```
-- Declare a handler for the end of cursor
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
  -- Open the cursor
  OPEN cur emp;
  -- Loop to fetch each row
  read_loop: LOOP
    FETCH cur emp INTO v EmpID, v EmpName, v DeptID, v Salary, v Job,
v City, v JoiningDate;
    IF done = 1 THEN
      LEAVE read loop;
    END IF;
    -- Display the fetched row (using SELECT)
    SELECT v EmpID AS EmpID, v EmpName AS EmpName, v DeptID AS
DeptID,
        v Salary AS Salary, v Job AS Job, v City AS City, v JoiningDate AS
JoiningDate;
  END LOOP;
  -- Close the cursor
  CLOSE cur emp;
END //
DELIMITER;
drop procedure ShowHighSalaryEmployees;
```

```
CALL ShowHighSalaryEmployees();
7. Write a trigger on the Emp table that checks before inserting a new employee
record: If the Salary is less than 10,000, prevent the insertion and raise an error
                   too low".
message "Salary
DELIMITER //
CREATE TRIGGER trg_CheckSalaryBeforeInsert
BEFORE INSERT ON Employee
FOR EACH ROW
BEGIN
  IF NEW.Salary < 10000 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Salary too low';
  END IF;
END //
DELIMITER;
INSERT INTO Employee (EmpID, EmpName, DeptID, Salary, Job, City, JoiningDate)
VALUES (14,'Rakesh', 20, 9000, 'HR Executive', 'Delhi', '2023-05-01');
Error Message: Salary too low.
8. Write a stored procedure in SQL to print numbers from 1 to 10 using a WHILE
loop.
DELIMITER //
CREATE PROCEDURE PrintNumbers()
BEGIN
  DECLARE i INT DEFAULT 1;
  WHILE i \le 10 DO
    SELECT i AS Number;
    SET i = i + 1;
  END WHILE;
```

```
END //
DELIMITER;
CALL PrintNumbers();
9. Write a stored procedure to print the multiplication table of 2 using a loop
      DELIMITER //
       CREATE PROCEDURE MultiplicationTable2()
      BEGIN
        DECLARE i INT DEFAULT 1;
        DECLARE result INT;
        WHILE i <= 10 DO
          SET result = 2 * i;
          SELECT CONCAT('2 x ', i, ' = ', result) AS Multiplication;
          SET i = i + 1;
        END WHILE;
      END //
      DELIMITER;
CALL MultiplicationTable2();
10. Write a function to check whether a number is even or odd.
   DELIMITER //
CREATE FUNCTION IsEvenOrOdd(num INT)
RETURNS VARCHAR(10)
DETERMINISTIC
BEGIN
  DECLARE result VARCHAR(10);
  IF MOD(num, 2) = 0 THEN
    SET result = 'Even';
  ELSE
```

```
SET result = 'Odd';
  END IF;
RETURN result;
END //
DELIMITER;
SELECT IsEvenOrOdd(7) AS Result;
SELECT IsEvenOrOdd(12) AS Result;
9. Write a user-defined function to calculate the factorial of a given number.
DELIMITER //
     CREATE FUNCTION Factorial(n INT)
     RETURNS BIGINT
     DETERMINISTIC
        BEGIN
     DECLARE result BIGINT DEFAULT 1;
        DECLARE i INT DEFAULT 1;
        IF n < 0 THEN
       RETURN NULL; -- Factorial not defined for negative numbers
  END IF;
   WHILE i <= n DO
  SET result = result * i;
  SET i = i + 1;
  END WHILE;
  RETURN result;
   END //
   DELIMITER;
   SELECT Factorial(5) AS Fact; -- Output: 120
```

SELECT Factorial(0) AS Fact; Output: 1
SELECT Factorial(10) AS Fact; Output: 3628800