STORED PROCEDURE

Write a stored procedure to read three numbers and find the greatest among them.

```
DROP PROCEDURE IF EXISTS FindGreatestNumber;
DELIMITER $$
CREATE PROCEDURE FindGreatestNumber(
        IN a INT,
        IN b INT,
        IN c INT,
        OUT greatest INT
)
BEGIN
        IF a >= b AND a >= c THEN
                SET greatest = a;
        ELSEIF b >= a AND b >= c THEN
                SET greatest = b;
        ELSE
                SET greatest = c;
        END IF;

END $$

DELIMITER ;
```

```
mysql> \. greatest.sql
Query OK, 0 rows affected (0.08 sec)

Query OK, 0 rows affected (0.06 sec)

mysql> call FindGreatestNumber(10,20,30,@result);
Query OK, 0 rows affected (0.00 sec)

mysql> select @result;
+---------+
| @result |
+---------+
|      30 |
+---------+
1 row in set (0.00 sec)
```

Create a ‘Customer’ table with attributes customer id, name, city and credits. Write a stored procedure to display the details of a particular customer from the customer table, where name is passed as a parameter.

```
DELIMITER $$

DROP PROCEDURE IF EXISTS CUSTOMER_NAME;

CREATE PROCEDURE CUSTOMER_NAME(
        IN cname VARCHAR(15),
        OUT cid INT,
        OUT ccity VARCHAR(15),
        OUT ccredits VARCHAR(10)
)

BEGIN

        SELECT id, city, credits
        INTO cid, ccity, ccredits
        FROM customer
        WHERE name=cname;
END $$

DELIMITER ;
```

```
mysql> \. customer.sql
Query OK, 0 rows affected (0.07 sec)

Query OK, 0 rows affected (0.14 sec)

mysql> call CUSTOMER_NAME('anjaly',@cid,@ccity,@ccredits);
Query OK, 1 row affected (0.01 sec)

mysql> select @cid,@ccity,@ccredits;
+------+--------+-----------+
| @cid | @ccity | @ccredits |
+------+--------+-----------+
|  101 | uk     | 5.6       |
+------+--------+-----------+
1 row in set (0.00 sec)
```

Write a stored procedure to read two numbers and print all the numbers between them.

```
DROP PROCEDURE IF EXISTS PRINT_NUMBERS;

DELIMITER $$

CREATE PROCEDURE PRINT_NUMBERS(IN a INT,IN b INT)

BEGIN
        DECLARE counter INT;
        DECLARE RESULT VARCHAR(100);
        SET counter = LEAST(a,b);
        SET RESULT = " ";
        num: LOOP
                SET RESULT = CONCAT(RESULT,counter,' ');
                SET counter=counter+1;
                IF counter >= GREATEST (a,b) THEN
                        LEAVE num;
                END IF;
        END LOOP;
        SELECT RESULT AS numbers;
END $$

DELIMITER ;
```

```
mysql> \. between.sql
Query OK, 0 rows affected (0.09 sec)

Query OK, 0 rows affected (0.07 sec)

mysql> call PRINT_NUMBERS(10,20);
+--------------------------------+
| numbers                        |
+--------------------------------+
|  10 11 12 13 14 15 16 17 18 19  |
+--------------------------------+
1 row in set (0.00 sec)

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

Write a stored procedure to read N and find the sum of the series 1+2+3 +... N

```
DELIMITER $$

DROP PROCEDURE IF EXISTS SumSeries;

CREATE PROCEDURE SumSeries(IN a INT, OUT sum INT)

BEGIN
   DECLARE counter INT DEFAULT 1;
   DECLARE total INT DEFAULT 0;

   WHILE counter <= a DO
      SET total = total + counter;
      SET counter = counter + 1;
   END WHILE;
   SET sum = total;

END$$

DELIMITER ;
```

CURSOR

Write a stored procedure using cursor to calculate salary of each employee. Consider an Emp_salary table have the following attributes emp_id, emp_name, no_of_working_days, designation and salary

```
DROP PROCEDURE IF EXISTS SALARY;

DELIMITER $$

CREATE PROCEDURE SALARY()

BEGIN
	DECLARE finished INTEGER DEFAULT 0;
	DECLARE idy VARCHAR(10);
	DECLARE desig VARCHAR(20);
	DECLARE days INT;
	DECLARE sal INT;
	DECLARE curSal CURSOR FOR SELECT
EMP_NO,CADRE,DAYS_WORKED FROM EMP;
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

	OPEN curSal;

	L1 : LOOP
		FETCH curSal INTO idy,desig,days;
		IF finished = 1 THEN
			LEAVE L1;
		END IF;
	IF(desig = 'AP') THEN
		SET sal = days * 1750;
	ELSEIF(desig = 'PR') THEN
		SET sal = days * 1250;
	ELSEIF(desig = 'CL') THEN
		SET sal = days * 750;
	END IF;
	UPDATE EMP SET SALARY=sal WHERE EMP_NO=idy;
	END LOOP L1;

	CLOSE curSal;
END $$

DELIMITER ;
```

```
mysql> \. salary.sql
Query OK, 0 rows affected (0.07 sec)

Query OK, 0 rows affected (0.06 sec)

mysql> call SALARY();
Query OK, 0 rows affected (0.23 sec)

mysql> select @result;
+------------------+
| @result          |
+------------------+
| NULL             |
+------------------+
1 row in set (0.00 sec)

mysql> select * from EMP;
+--------+-------+-------------+--------+
| EMP_NO | CADRE | DAYS_WORKED | SALARY |
+--------+-------+-------------+--------+
| 101    | AP    |          25 |  43750 |
| 102    | PR    |          24 |  30000 |
| 103    | CL    |          28 |  21000 |
+--------+-------+-------------+--------+
3 rows in set (0.00 sec)
```

Write a stored procedure using cursor to display email of each employee as a single list

```
DELIMITER $$

DROP PROCEDURE IF EXISTS EMAIL_LIST;

CREATE PROCEDURE EMAIL_LIST(INOUT List VARCHAR(4000))
BEGIN
	DECLARE finished INTEGER DEFAULT 0;
	DECLARE email_id VARCHAR(20);
	DECLARE  curEmail CURSOR FOR SELECT EMAIL FROM EMP;
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

	OPEN curEmail;
		FETCH curEmail INTO list;
	L1 : LOOP
		IF finished = 1 THEN
			LEAVE L1;
		END IF;
		FETCH curEmail INTO email_id;
		SET list = CONCAT(list,';',email_id);
	END LOOP L1;

	CLOSE curEmail;
END $$

DELIMITER ;
```

```
mysql> \. email.sql
Query OK, 0 rows affected (0.07 sec)

Query OK, 0 rows affected (0.13 sec)

mysql> call EMAIL_LIST(@result);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @RESULT;
+-----------------------------------------------------------+
| @RESULT                                                   |
+-----------------------------------------------------------+
| 101@rit.ac.in;102@rit.ac.in;103@rit.ac.in;103@rit.ac.in |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

TRIGGER

Create a trigger on employee table such that whenever a row is updated, it is moved to table named 'EMP_AUDIT' with the same structure as employee table. 'Emp_history' will contain an additional column "Date" to store the date on which the row is updated. [Before Update Trigger]

```
DELIMITER //

DROP TRIGGER IF EXISTS before_employee_update;

CREATE TRIGGER before_employee_update
BEFORE UPDATE ON EMP
FOR EACH ROW

BEGIN
    INSERT INTO EMP_AUDIT(ACTION,EMP_NUM,NAME,DATE)
    VALUES('update',OLD.EMP_NO,OLD.NAME,NOW());
END;
//

DELIMITER ;
```

```
mysql> \. audit.sql
Query OK, 0 rows affected (0.06 sec)

Query OK, 0 rows affected (0.13 sec)

mysql> UPDATE EMP SET NAME='ANAMIKA' WHERE EMP_NO='101';
Query OK, 0 rows affected (0.05 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> select * from EMP_AUDIT;
+--------+---------+---------+------------+
| ACTION | EMP_NUM | NAME    | DATE       |
+--------+---------+---------+------------+
| update | 101     | ANU     | 2025-04-08 |
| update | 101     | ANAMIKA | 2025-04-08 |
+--------+---------+---------+------------+
2 rows in set (0.00 sec)
```

Create a trigger on employee table such that whenever a row is deleted, it is moved to history table named ‘Emp_history’ with the same structure as employee table. ‘Emp_history’ will contain an additional column “Date” to store the date on which the row is removed. [After Delete Trigger]

```
DELIMITER //

DROP TRIGGER IF EXISTS after_employee_delete;

CREATE TRIGGER after_employee_delete

AFTER DELETE ON EMPLOYEE
FOR EACH ROW

BEGIN
        INSERT INTO EMP_HISTORY(ACTION,EMP_NO,EMP_NAME,DATE)
        VALUES('delete',OLD.EMP_NO,OLD.EMP_NAME,NOW());
END;
//

DELIMITER ;
```

```
mysql> \. history.sql
Query OK, 0 rows affected (0.07 sec)

Query OK, 0 rows affected (0.13 sec)

mysql> DELETE FROM EMPLOYEE WHERE EMP_NO=101;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM EMP_HISTORY;
+--------+----------+------------+--------+
| EMP_NO | EMP_NAME | DATE       | ACTION |
+--------+----------+------------+--------+
|    101 | Anamika  | 2025-04-10 | delete |
+--------+----------+------------+--------+
1 row in set (0.00 sec)
```

Before insert a new record in employee table, create a trigger that check the column value of EMP_NAME and the value will be converted to upper cases by UPPER () function. [Before Insert Trigger]

```
DELIMITER $$

DROP TRIGGER IF EXISTS before_insert_details;

CREATE TRIGGER before_insert_details
BEFORE INSERT ON EMPLOYEE
FOR EACH ROW

BEGIN
        SET NEW.EMP_NAME = UPPER(NEW.EMP_NAME);
END $$

DELIMITER ;
```

```
mysql> \. details.sql
Query OK, 0 rows affected (0.06 sec)

Query OK, 0 rows affected (0.13 sec)

mysql> INSERT INTO EMPLOYEE VALUES(103,'minna');
Query OK, 1 row affected (0.05 sec)

mysql> SELECT * FROM EMPLOYEE;
+--------+----------+
| EMP_NO | EMP_NAME |
+--------+----------+
|    102 | Anjaly   |
|    103 | MINNA    |
+--------+----------+
2 rows in set (0.00 sec)
```