CLASS 5

SQL

TASKS

NAME: - Karan Shah

Batch: - Full Stack Data Analytics Course

```
# Task 1 :Create a loop for a table to insert a record into table for two columns in
# first column insert data ranging from 1 to 100 in second column insert data square of the first column
# Creating a table with two columns with Int datatyoe to insert data as per given condition
CREATE TABLE IF NOT EXISTS fortask(
column1 INT,
   column2 INT
);

# Creating a procedure of loop as per the given task
DELIMITER &&
CREATE PROCEDURE taskone()
BEGIN
SET @var1 = 1;
   for_task_one : loop
   INSERT INTO fortask VALUES(@var1, (@var1 * @var1));
   SET @var1 = @var1 + 1;
   IF @var1 = 100 THEN
LEAVE for_task_one;
END IF;
   END loop for_task_one;
```

```
END &&
# Calling the procedure
CALL taskone();
# Printing all the records
SELECT * FROM fortask;
```

Task 2 : Create a UDF to find out a date differences in number of days (date1, date2) returns number of days between two dates.

Solution

```
DELIMITER &&
CREATE FUNCTION task2(date1 DATE, date2 DATE)
RETURNS INT
DETERMINISTIC
BEGIN
DECLARE difference_between INT;
SET difference_between = DATEDIFF(date1, date2);
   RETURN ABS(difference_between);
END &&

SELECT task2(new_ship_date,new_order_date) from sales;
SELECT task2(new_order_date, new_ship_date) from sales;
SELECT * FROM sales;
```

Task 3 : Create a UDF to find out log of base 10 of any even number

Solution

```
DELIMITER &&
CREATE FUNCTION task3(num1 INT)
RETURNS DECIMAL(20,6)
DETERMINISTIC
BEGIN
IF num1 % 2 != 0 THEN
RETURN 0;
ELSE
RETURN LOG10(num1);
END IF;
END &&
SELECT task3(2);
```

Task 4 : Create a UDF which will be able to check total number of records available in your table

Solution

```
DELIMITER &&
CREATE FUNCTION task4(var VARCHAR(60))
RETURNS INT
DETERMINISTIC
BEGIN
DECLARE var1 INT;
   SET var1 = (SELECT MAX(TABLE_ROWS) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = var);
   RETURN var1;
END &&
```

Testing the function task4

```
SELECT task4('sales');
```

```
# Task 5 : Create a procedure to find out 5th highest profit in your sales table without using
# rank or windowing function.
# SOLUTION
DELIMITER &&
CREATE PROCEDURE task5(IN n INT)
BEGIN
SELECT sales_records FROM sales s1
   WHERE n =
   (SELECT count(DISTINCT(sales_records)) FROM sales s2
   WHERE s2.sales_records > s1.sales_records);
END &&



SELECT DISTINCT(sales) from sales order by sales DESC LIMIT 5;


# Finding the highest sales
SELECT MAX(sales) FROM sales;


# Finding the 2nd highest sales from sales
SELECT max(sales) FROM sales WHERE sales NOT IN (select max(sales) FROM sales);


# Selecting the 3rd, 4th, 5th highest sales from sales
CALL task5(3);
```