Aim: SQL Functions and Procedures

Objectives:

The purpose of learning SQL Functions and Procedures is to understand how to simplify and organize database operations. Functions and procedures help developers write reusable code directly in the database, making tasks like calculations, data processing, or enforcing rules easier and faster. The goal is to learn how to use these tools to avoid repetition, improve efficiency, and ensure the database works smoothly.

Tools Used:

- MySQL Workbench

Concept:

SQL Functions and Procedures are features that allow you to group tasks into a single block of code.

- **Functions**: These are used to perform a specific task and return a single value, like calculating a total or formatting text.
- **Procedures**: These can perform multiple tasks, like inserting or updating records, and they don't always return a value.

Using functions and procedures makes it easier to handle repetitive tasks, follow business rules, and keep the database organized and efficient.

Example:

Function

```
CREATE DEFINER=`root`@`localhost` FUNCTION `cubeCalutaor`( num int) RETURNS int
    DETERMINISTIC
BEGIN

declare cube_of_num int;
 set cube_of_num = num * num * num;
RETURN cube_of_num;
END
```

Procedure

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Get_Orders_Status`(IN input_status
varchar(20))
BEGIN
select
      year(orderDate) as Year,
      count(*) as Total_Orders
   from
```

```
    orders
  where
    status = input_status
  group by
    year(orderDate);
END
```

Problem Statement:

Assignment on Function Question:

1) Create a function to find the cube of a number.

2) Use Classicmodels. Create a function which will return city of the given officeCode.

3) Use Classicmodels. Create a function to show the highest MSRP for each productline using window functions.

4) Use Classicmodels. Create a function to show the customername who has used thehighest CreditLimit.

Assignment on Procedure Question:

Use classicmodels. Create a procedure Get_Orders_Status which should accept the status value from user and show the number of orders for each year for that status.

Table- Orders The output should look as shown in below image.

***Shipped** Status*

Year	Total Orders
2003	108
2004	145
2005	50

***Cancelled** Status*

Year	Total Orders
2003	2
2004	4

***On Hold** Status*

Year	Total Orders
2004	1
2005	3

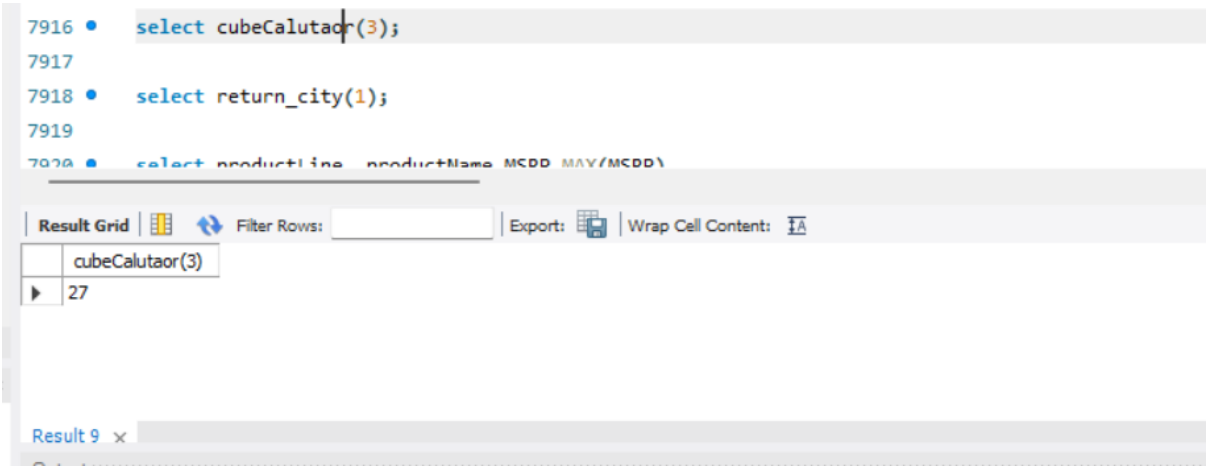
Solution:

Function

1)

```
CREATE DEFINER=`root`@`localhost` FUNCTION `cubeCalutaor`( num int) RETURNS int
    DETERMINISTIC
BEGIN

declare cube_of_num int;
 set cube_of_num = num * num * num;
RETURN cube_of_num;
END
--------
select cubeCalutaor(3);
```



2)

```
CREATE DEFINER=`root`@`localhost` FUNCTION `return_city`(officeCode_entered int) RETURNS
varchar(50) CHARSET latin1
    DETERMINISTIC
BEGIN
declare city_op varchar(50);
select city into city_op from offices where officeCode = officeCode_entered;
```

RETURN city_op;

END

---

select return_city(1);

```
7917
7918 •  select return_city(1);
7919
7920 •  select productLine, productName,MSRP,MAX(MSRP)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

return_city(1)
San Francisco

Result 10 ×

Output

3)

select productLine, productName,MSRP,MAX(MSRP)

over (partition by productLine) as highestMSRP

from

products;

```
7920 •  select productLine, productName,MSRP,MAX(MSRP)
7921    over (partition by productLine) as highestMSRP
7922    from
7923    products;
7924
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

productLine	productName	MSRP	highestMSRP
Classic Cars	1952 Alpine Renault 1300	214.30	214.30
Classic Cars	1972 Alfa Romeo GTA	136.00	214.30
Classic Cars	1962 LanciaA Delta 16V	147.74	214.30
Classic Cars	1968 Ford Mustang	194.57	214.30
Classic Cars	2001 Ferrari Enzo	207.80	214.30
Classic Cars	1969 Corvair Monza	151.08	214.30
Classic Cars	1968 Dodge Charger	117.44	214.30

Result 11 ×

Output

4)

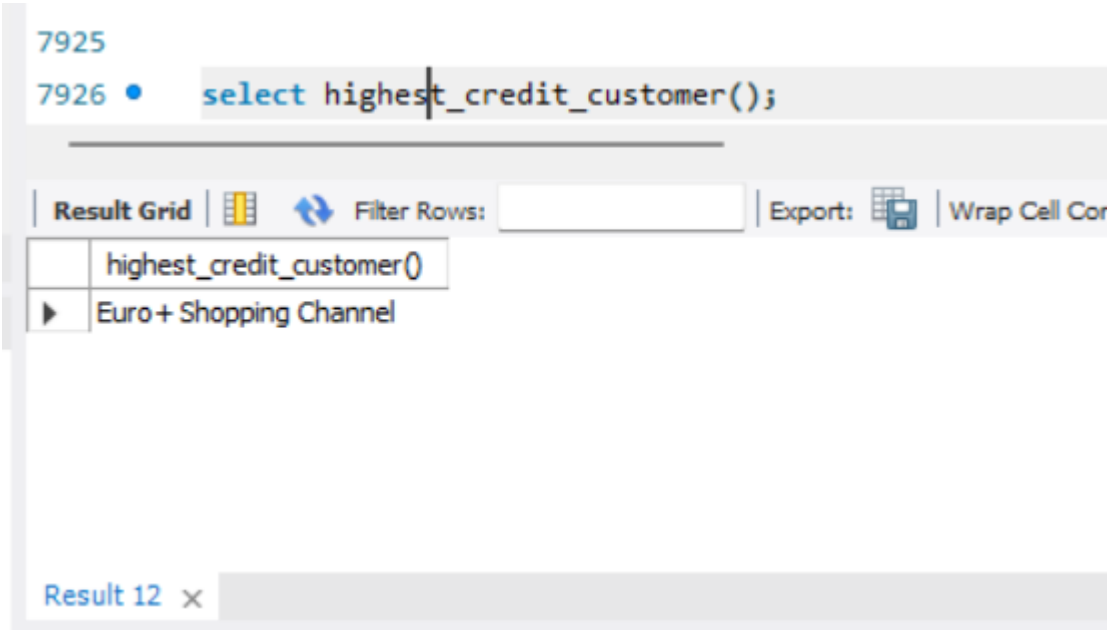
```
CREATE DEFINER=`root`@`localhost` FUNCTION `highest_credit_customer`() RETURNS varchar(50)
CHARSET latin1
    DETERMINISTIC
BEGIN

  DECLARE customerName_op varchar(50);

SELECT customerName into customerName_op
FROM customers
ORDER BY CreditLimit DESC
LIMIT 1;

    RETURN customerName_op
    ;
END
```

```
select highest_credit_customer();
```



Procedure

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Get_Orders_Status`(IN input_status varchar(20))
BEGIN
select
    year(orderDate) as Year,
    count(*) as Total_Orders
  from
    orders
  where
    status = input_status
  group by
    year(orderDate);
END
```

```
7927
7928 •  call Get_Orders_Status('Shipped');
7929 •  call Get_Orders_Status('Cancelled');
7930 •  call Get_Orders_Status('On Hold');
7931
7932
```

Result Grid | Filter Rows: | Export: | Wrap

Year	Total_Orders
2003	108
2004	145
2005	50

Result 14 × Result 15 Result 16

Output

```
7927
7928 •  call Get_Orders_Status('Shipped');
7929 •  call Get_Orders_Status('Cancelled');
7930 •  call Get_Orders_Status('On Hold');
7931
7932
```

Result Grid | Filter Rows: | Export: | Wrap Cell

Year	Total_Orders
2003	2
2004	4

Result 14 Result 15 × Result 16

Output

```
7927
7928 •  call Get_Orders_Status('Shipped');
7929 •  call Get_Orders_Status('Cancelled');
7930 •  call Get_Orders_Status('On Hold');
7931
7932
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Year	Total_Orders
2004	1
2005	3

Result 14 Result 15 Result 16 ×

Observation:

Functions and procedures in SQL are important tools that help make database operations easier and more efficient. They allow you to group tasks into reusable blocks of code. Functions are great for doing calculations or getting a specific result, while procedures are used for performing multiple steps or tasks. By using these, you can save time, avoid repeating code, and make sure the database runs smoothly. Functions and procedures also help keep data accurate and organized, reducing errors and improving the system's reliability.