Code :

```
mysql> drop database if exists A7;
mysql> create database A7;
mysql> use A7;
Database changed
mysql>
mysql> CREATE TABLE customer(
   ->     id INT,
   ->     cust_name VARCHAR(20),
   ->     total_purchase INT,
   ->     PRIMARY KEY(id)
   ->     );
Query OK, 0 rows affected (0.05 sec)
mysql> CREATE TABLE category(
   ->     cust_id INT,
   ->     name VARCHAR(20),
   ->     class VARCHAR(20),
   ->     FOREIGN KEY(cust_id) REFERENCES customer(id) ON DELETE CASCADE
   ->     );
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> INSERT INTO customer VALUES(1,"Prathamesh",9000);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO customer VALUES(2,"Aditya",15000);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO customer VALUES(3,"Varun",3000);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO customer VALUES(4,"Prachi",2000);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO customer VALUES(5,"Utkarsh",11000);
Query OK, 1 row affected (0.02 sec)

mysql> delimiter //
mysql> CREATE PROCEDURE decideCategory(IN total_purchase INT, OUT class VARCHAR(20))
   -> BEGIN
   -> IF total_purchase<= 20000 AND total_purchase>=10000 THEN SET class="PLATINUM";
   -> END IF;
   -> IF total_purchase<10000 AND total_purchase >= 5000 THEN SET class="GOLD";
   -> END IF;
```

```
    -> IF total_purchase<5000 AND total_purchase >=2000 THEN SET class="SILVER";
    -> END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE PROCEDURE proc_category()
    -> BEGIN
    ->     DECLARE cust_name VARCHAR(20);
    ->     DECLARE b INT default 0;
    ->     DECLARE cust_id  INT;
    ->     DECLARE total_purchase INT;
    ->     DECLARE class INT;
    ->     DECLARE c1 cursor for SELECT * FROM customer;
    ->     DECLARE CONTINUE handler for NOT found SET b=1;
    ->     OPEN c1;
    ->     repeat
    ->     FETCH c1 INTO cust_id,cust_name,total_purchase;
    ->     IF NOT b THEN
    ->         CALL decideCategory(total_purchase,@class);
    ->         INSERT INTO category VALUES(cust_id,cust_name,@class);
    ->     END IF;
    ->     until b END repeat;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter ;
mysql> SELECT * FROM category;
Empty set (0.00 sec)

mysql>
mysql> CALL proc_category();
mysql> SELECT * FROM category;
+---------+------------+----------+
| cust_id | name       | class    |
+---------+------------+----------+
|       1 | Prathamesh | GOLD     |
|       2 | Aditya     | PLATINUM |
|       3 | Varun      | SILVER   |
|       4 | Prachi     | SILVER   |
|       5 | Utkarsh    | PLATINUM |
+---------+------------+----------+
5 rows in set (0.00 sec)

mysql>
mysql> delimiter //
mysql> CREATE FUNCTION insertCategory(cust_id INT) RETURNS VARCHAR(20) deterministic
```

```
    -> BEGIN
    -> DECLARE total_purchase INT;
    -> DECLARE class VARCHAR(20);
    -> DECLARE cust_name VARCHAR(20);
    -> SELECT customer.cust_name,customer.total_purchase INTO cust_name,total_purchase FROM
customer WHERE customer.id=cust_id;
    -> CALL decideCategory(total_purchase,@class);
    -> INSERT INTO category VALUES(cust_id,cust_name,@class);
    -> RETURN @class;
    -> END;
    -> //
Query OK, 0 rows affected (0.02 sec)

mysql> delimiter ;
mysql> DELETE FROM category;
Query OK, 5 rows affected (0.02 sec)

mysql>
mysql> SELECT insertCategory(1);
+-------------------+
| insertCategory(1) |
+-------------------+
| GOLD              |
+-------------------+
1 row in set (0.01 sec)

mysql> SELECT * FROM category;
+---------+------------+-------+
| cust_id | name       | class |
+---------+------------+-------+
|       1 | Prathamesh | GOLD  |
+---------+------------+-------+
1 row in set (0.00 sec)

mysql>
```

Assignment A7

Date of Completion : 28/09/2020

Date of Submission : 12/10/2020

Title: PL/SQL Stored Procedure & function.

Problem Statement:
Writing a stored procedure decide Category for categorization of customer. If purchase <= 20,000 & >= 10,000 then platinum category, if purchase between 5000 & 10,000 then gold category, if purchase between 2000 & 5000 then silver category.

Objective:
- To understand PL/SQL stored procedure
- To understand PL/SQL stored procedure.

Outcome:
Students will be able to
- Write PL/SQL stored procedures.
- Write PL/SQL stored functions

S/W & H/W Requirements:
MYSQL, Windows/linux, i5 processor, PL/SQL.

Theory:

MYSQL:

It supports 2 kinds of routines.

(1) Stored Procedures:

A procedure is a sub routine in a regular scripting language, stored in a database.

- Create a stored procedure.

```
Create procedure decideCategory(params)
  begin
  ...
  end;
```

- calling a procedure

Syntax:

```
call decideCategory(params)
```

- dropping a procedure:

```
drop procedure decideCategory;
```

- Parameter modes

1) IN: Value must be passed, original value is retained after procedure execution.

OUT: Value can change within the procedure & value is returned to calling application

INOUT: We can pass initial value, the procedure may change it & then return it.

- Stored function:
  A stored function is a set of SQL statements that perform some operation & return a single value.

Creating Stored function

Syntax:

```
create function func_name (params)
    return datatype [characteristics]
            [NOT] DETERMINISTIC
    begin
    ...
    end;
```

Calling a Stored function

A stored function can be called inside a procedure or select statement.

Deterministic function:

They always return the same result any time they are called with a specific set of input values.

Non deterministic function:
returns different result each time.

PL/SQL

Stored procedure:
A stored procedure is a named PL/SQL block which performs a specific task
Syntax: exec proc_name (params)

Stored function:
A function is a stored PL/SQL block which is ~~AAis~~ similar to a procedure the major difference is a function must always return a value.

- Stored function:

```
CREATE FUNCTION insertCategory (cust_id INT)
                RETURNS VARCHAR(20) deterministic
BEGIN
DECLARE tot_p INT;
DECLARE class VARCHAR(20);
DECLARE cust_name VARCHAR(20);
SELECT customer.cust_name, customer.tot_p
INTO cust_name, tot_p FROM customer
WHERE customer.id = cust_id
CALL decideCategory (tot_p, @class)
INSERT INTO category VALUES( cust_id,
                cust_name, @class)
```

```
RETURN @class
END;
```

- Stored procedure with params:

```
CREATE PROCEDURE decideCategory (IN tot_p INT,
                                 OUT class VARCHAR(20))
BEGIN
If tot_p <= 20000 AND tot_p >= 10000
        THEN SET class = "PLATINUM";
END IF;
IF tot_p < 10000 AND tot_p >= 5000
        THEN SET class = "GOLD";
END IF;
IF tot_p < 5000 AND tot_p >= 2000
        THEN SET class = "Silver";
END IF;
END;
```

- CONCLUSION:

Thus we implemented
1) Stored procedure without parameters
2) Stored procedure with parameters
3) Stored functions.