DBMS LAB-10

NAME:- ABHISHIKTH BODA

ROLL NUMBER:- S20210010044

DATE:- 09-11-2022

STORED PROCEDURES AND FUNCTIONS

EXAMPLE 1:

Display details of all books.

Procedure:-

```
mysql> delimiter ..
mysql> create procedure display_book()
    -> begin
    -> select *from book;
    -> end ..
Query OK, 0 rows affected (0.05 sec)
```

Execution:-

```
mysql> call display_book();
    -> ..
+--------+------+-------------------------------+--------+--------+-------+-------+
| BookId | ISBN | book_name                     | author | ed_num | price | pages |
+--------+------+-------------------------------+--------+--------+-------+-------+
|      1 |    1 | Glimpses of the past          |      1 |      1 |   650 |   396 |
|      2 |    2 | Beyond The Horizons of Venus  |      1 |      1 |   650 |   396 |
|      3 |    3 | Ultrasonic Aquaculture        |      2 |      1 |   799 |   500 |
|      4 |    4 | Cyrogenic Engines             |      2 |      1 |   499 |   330 |
+--------+------+-------------------------------+--------+--------+-------+-------+
4 rows in set (0.01 sec)

Query OK, 0 rows affected (0.06 sec)
```

EXAMPLE 2:

Update price of a book taking ISBN of the book and its new price as Input.

Procedure:-

```
mysql> delimiter ..
mysql> create procedure update_price (IN temp_ISBN varchar(10), IN new_price
    -> integer) begin
    -> update book set price=new_price where ISBN=temp_ISBN;
    -> end ..
Query OK, 0 rows affected (0.01 sec)
```

Execution:-

```
mysql> call update_price(2,780);..
Query OK, 1 row affected (0.01 sec)

mysql> select * from book;
    -> ..
+--------+------+------------------------------+--------+--------+-------+-------+
| BookId | ISBN | book_name                    | author | ed_num | price | pages |
+--------+------+------------------------------+--------+--------+-------+-------+
|      1 |    1 | Glimpses of the past         |      1 |      1 |   650 |   396 |
|      2 |    2 | Beyond The Horizons of Venus |      1 |      1 |   780 |   396 |
|      3 |    3 | Ultrasonic Aquaculture       |      2 |      1 |   799 |   500 |
|      4 |    4 | Cyrogenic Engines            |      2 |      1 |   499 |   330 |
+--------+------+------------------------------+--------+--------+-------+-------+
4 rows in set (0.00 sec)

mysql>
```

EXAMPLE 3:

Display the highest price among all the books with an output Parameter.

Procedure:-

```
ysql> delimiter ..
ysql> create procedure disp_max(OUT highestprice integer)
    -> begin
    -> select max(price) into highestprice from book;
    -> end ..
uery OK, 0 rows affected (0.01 sec)
```

Execution:-

```
mysql> call disp_max(@max);
    -> ..
Query OK, 1 row affected (0.01 sec)

mysql> select @max;..
+------+
| @max |
+------+
|  799 |
+------+
1 row in set (0.01 sec)

mysql>
```

EXAMPLE 4:

Accept gender as input and display no.of authors having the given gender.

Procedure:-

```
mysql> delimiter ..
mysql> create procedure disp_gender(INOUT mfgender integer, IN emp_gender varchar(6))
    -> begin
    -> select count(gender) into mfgender from author where gender = emp_gender;
    -> end ..
Query OK, 0 rows affected (0.01 sec)
```

Execution:-

```
mysql> call disp_gender(@count,'Male');..
Query OK, 1 row affected (0.01 sec)

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

mysql>
```

STORED PROCEDURE VS TRIGGER:

EXAMPLE:

Let's create a trigger named updateItemPrice. This particular trigger is activated whenever the items table is updated. When this event occurs, the trigger checks each row to see if the product cost (cost) value is being changed. If it is, then the trigger automatically sets the item's new price (price) to 1.40 times the item's new cost (in other words, a 40% markup).

To create this trigger, run the following MySQL statements:

Trigger:-

```
mysql> DELIMITER ..
mysql> CREATE TRIGGER `updateItemPrice`
    -> BEFORE UPDATE ON `items`
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.cost <> OLD.cost
    ->
    -> THEN
    -> SET NEW.price = NEW.cost * 1.40;
    -> END IF ;
    -> END..
Query OK, 0 rows affected (0.03 sec)
```

Execution:-

```
mysql> DELIMITER ;
mysql> UPDATE items SET cost = 7.00 WHERE id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM items;
+----+--------------+-------+-------+
| id | name         | cost  | price |
+----+--------------+-------+-------+
|  1 | Basic Widget |     7 |   9.8 |
|  2 | Micro Widget |  0.95 |  1.35 |
|  3 | Mega Widget  | 99.95 |   140 |
+----+--------------+-------+-------+
3 rows in set (0.00 sec)
```

ERROR HANDLING:

EXAMPLE:-

Procedure:-

```
mysql> DELIMITER ..
mysql> CREATE PROCEDURE emp_details
    -> (
    -> InputID INTEGER
    -> ,InputName VARCHAR(50)
    -> ,InputDept VARCHAR(50)
    -> )
    -> BEGIN
    -> DECLARE EXIT HANDLER FOR SQLEXCEPTION
    -> SELECT 'Error occured';
    -> INSERT INTO employees VALUES(InputID, InputName, InputDept);
    -> SELECT *FROM employees;
    -> END
    -> ..
Query OK, 0 rows affected (0.01 sec)
```

Execution:-

```
mysql> DELIMITER ;
mysql> call emp_details(200,'Johny','HR');
+-----+-------+------------+
| ID  | name  | department |
+-----+-------+------------+
| 200 | Johny | HR         |
+-----+-------+------------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> call emp_details(200,'Johny','HR');
+---------------+
| Error occured |
+---------------+
| Error occured |
+---------------+
1 row in set (0.01 sec)
```

MYSQL CURSOR:

EXAMPLE:-

Cursor to iterate emp name and their place:

Procedure:-

```
mysql> DELIMITER ..
mysql> CREATE PROCEDURE emp_curs()
    -> BEGIN
    -> DECLARE finished INTEGER DEFAULT 0;
    -> DECLARE ename varchar(100);
    -> DECLARE eplace varchar(100);
    ->
    -> DEClARE curname CURSOR FOR SELECT emp_name, place FROM
    -> employee;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
    -> OPEN curname;
    -> getname: LOOP
    -> FETCH curname INTO ename, eplace;
    -> IF finished = 1 THEN
    -> LEAVE getname;
    -> END IF;
    -> SELECT ename,eplace;
    -> END LOOP getname;
    -> CLOSE curname;
    -> END..
Query OK, 0 rows affected (0.00 sec)
```

Execution:-

```
mysql> call emp_curs();
+---------+------------+
| ename   | eplace     |
+---------+------------+
| peter   | Newyork    |
+---------+------------+
1 row in set (0.03 sec)

+---------+------------+
| ename   | eplace     |
+---------+------------+
| Mark    | California |
+---------+------------+
1 row in set (0.03 sec)

+---------+------------+
| ename   | eplace     |
+---------+------------+
| Donald  | Arizona    |
+---------+------------+
1 row in set (0.03 sec)

+---------+------------+
| ename   | eplace     |
+---------+------------+
| Obama   | Florida    |
+---------+------------+
```

Example 2:

Let's say we sell products of some types. We want to count how many products of each type exist.

Procedure:-

```
mysql> DELIMITER ..
mysql> DROP PROCEDURE IF EXISTS product_count;
    -> CREATE PROCEDURE product_count()
    -> BEGIN
    -> DECLARE p_type VARCHAR(255);
    -> DECLARE p_count INT;
    -> DECLARE done INT DEFAULT 0;
    -> DECLARE product_curs CURSOR FOR
    -> SELECT type,COUNT(*)FROM product GROUP BY type;
    -> DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;
    -> TRUNCATE product_type;
    -> OPEN product_curs;
    -> REPEAT
    -> FETCH product_curs
    -> INTO p_type, p_count;
    -> IF NOT done
    -> THEN
    -> INSERT INTO product_type_count
    -> SET
    -> type = p_type,
    -> count = p_count;
    -> END IF;
    -> UNTIL done
    ->
    -> END REPEAT;
    -> CLOSE product_curs;
    -> END ..
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

Execution:-

```
mysql> DELIMITER ;
mysql> CALL product_count();
Query OK, 0 rows affected (0.04 sec)

mysql> select * from product_type_count;
+-------+-------+
| type  | count |
+-------+-------+
| dress |     2 |
| food  |     3 |
+-------+-------+
2 rows in set (0.00 sec)

mysql>
```

Exercise:-

Question 1:-

Consider the bank database. Let us define a view branch_cust as follows:

create view branch_cust as

select branch name, customer name

from depositor, account

where depositor.account number = account.account number

Suppose that the view is materialized; that is, the view is computed and stored. Write triggers to maintain the view, that is, to keep it up-to-date on insertions to and deletions from depositors or accounts. Do not bother about updates.

Trigger:-

```
mysql> create view branch_cust as
    -> select branch_name, customer_name
    -> from depositor_relation, account_relation
    -> where depositor_relation.account_number = account_relation.account_number
    -> DELIMITER ..
```

```
mysql> CREATE TRIGGER branch_cust_on_depositor_insert
    -> AFTER INSERT ON account_relation for each row
    -> BEGIN
    -> declare cust_name varchar(20);
    -> SELECT customer_name INTO cust_name
    -> FROM depositor_relation
    -> WHERE account_number= @new.account_number;
    -> IF cust_name IS NOT NULL THEN
    -> INSERT INTO branch_cust ( branch_name, customer_name ) VALUES (@new.branch_name,cust_name) ;
    -> END IF;
    -> END ..
Query OK, 0 rows affected (0.01 sec)
```

Execution:-

```
mysql> insert into account_relation values ('A-160','Redwood',800);
    -> insert into depositor_relation values('Smith','A-160');
    -> select * from branch_cust;..
Query OK, 1 row affected (0.00 sec)
```

Question 2:-

Consider the bank database. Write an SQL trigger to carry out the following action:

On delete of an account, for each owner of the account, check if the owner has any remaining accounts, and if she does not, delete her from the depositor relation.

Trigger:-

```
mysql> delimiter //
mysql> drop trigger if exists checkDelete;
    -> create trigger checkDelete after delete on account for each row
    -> begin
    -> declare hold varchar(30);
    -> declare decider int;
    -> set hold = old.account_number;
    -> select count(*) into decider from account where account_number = hold;
    -> if decider = 0 then
    -> delete from depositor where account_number = hold;
    -> end if;
    -> end //
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.02 sec)
```

Execution:-

```
mysql> delete from account where account_number = 'A-101';
    -> //
Query OK, 1 row affected (0.02 sec)

mysql> select * from account;//
+----------------+-------------+---------+
| account_number | branch_name | balance |
+----------------+-------------+---------+
| A-102          | Perryridge  |     400 |
| A-201          | Brighton    |     900 |
| A-215          | Mianus      |     700 |
| A-217          | Brighton    |     750 |
| A-222          | Redwood     |     700 |
| A-305          | Round Hill  |     350 |
| A-160          | new town    |     600 |
+----------------+-------------+---------+
7 rows in set (0.00 sec)
```

Question 3:-

Write a procedure to print the number of books written by a given author and their average price.

Procedure:-

```
mysql> delimiter ..
mysql> CREATE PROCEDURE Info_Author(IN id int)
    -> BEGIN
    -> select count(*) as no_of_books,avg(price) as Avg_price from book where author=id;
    -> END..
Query OK, 0 rows affected (0.00 sec)
```

Execution:-

```
mysql> call Info_Author(2);..
+-------------+-----------+
| no_of_books | Avg_price |
+-------------+-----------+
|           2 |  649.0000 |
+-------------+-----------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql>
```

←THE END→