DBMS Assignment 9

Cursor and Triggers

Considering the tables of Assignment 7, perform the following tasks:

Cursors:

1. Create a cursor to fetch the count of customers and sellers.

Command:

```
DECLARE
  cur sys_refcursor;
  sel_rec seller%rowtype;
  cust_rec customer%rowtype;
BEGIN
  OPEN cur FOR SELECT * FROM seller;
  LOOP
    FETCH cur INTO sel_rec;
    EXIT WHEN cur%notfound;
  END LOOP;

  dbms_output.put_line('Total Rows: ' || cur%rowcount);--here you will get total row count

  OPEN cur FOR SELECT * FROM customer;
  LOOP
    FETCH cur INTO cust_rec;
    EXIT WHEN cur%notfound;
  END LOOP;

  dbms_output.put_line('Total Rows: ' || cur%rowcount);--here you will get total row count
END;
```

Output:

```
Results   Explain   Describe   Saved SQL   History

Total Rows: 7
Total Rows: 10

Statement processed.

0.04 seconds
```

2. Create a cursor to display all the product details with rating more than 3.5.

Command:

```
DECLARE
cur sys_refcursor;
pro  product%rowtype;

BEGIN
dbms_output.put_line('productid' ||' '|| 'product' ||' '|| 'amount' ||' '|| 'quantityremainin
g' ||' '|| 'categoryid' ||' '|| 'sellerid'||' ' || 'rating');
OPEN cur FOR SELECT * FROM product where rating>3.5;
LOOP
FETCH cur INTO pro;
EXIT WHEN cur%notfound;
dbms_output.put_line(pro.productid||''||pro.product||''||pro.amount||''||pro.quant
ityremaining ||' ' ||pro.categoryid  ||' '||pro.sellerid  ||' '|| pro.rating );
end LOOP;
end;
```

Output:

```
Results   Explain   Describe   Saved SQL   History

productid product amount quantityremaining categoryid sellerid rating
1P The Programming lang 350 4 1C 1S 4.5
3P White Lamp 800 3 3C 5S 4
8P Portico King
size b 1999 1 3C 1S 5

Statement processed.

0.01 seconds
```

3. Create a cursor to display all the products category wise.

Command:

```
DECLARE
cur sys_refcursor;
pro  product%rowtype;

BEGIN
dbms_output.put_line('product' ||' '||'categoryid');
OPEN cur FOR SELECT * FROM product Order by categoryid;
LOOP
FETCH cur INTO pro;
EXIT WHEN cur%notfound;
dbms_output.put_line(pro.product || ' ' || pro.categoryid);

end LOOP;
end;
```

Output:

Results | Explain | Describe | Saved SQL | History

```
product categoryid
Introduction to
pyt 1C
The Programming lang 1C
Artificial
Intellig 1C
Introduction to
Jav 1C
Catwalk
leather fla 2C
Classmate Notebook 2C
Nike White
shoes 2C
Book rack 3C
White Lamp 3C
Portico King
size b 3C
Antique Silver
Earr 4C
Antique Silver
Brac 4C

Statement processed.

0.01 seconds
```

Triggers:

1. Create a trigger to update the remaining quantity of product in the product table, when a new entry in order_products table is inserted.

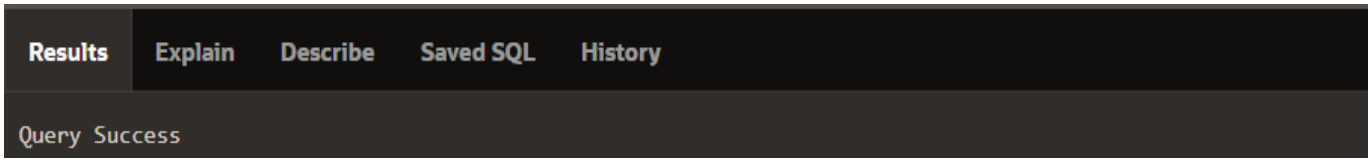
Script:

```
CREATE OR REPLACE TRIGGER MODIFY_QUANTITY
 AFTER INSERT ON order_products
 FOR EACH ROW
 ENABLE
 DECLARE qtybought number;
 BEGIN qtybought:= :NEW.Quantity;
UPDATE product SET QuantityRemaining = QuantityRemaining-
qtybought where productid=:NEW.productid;
dbms_output.put_line('Query changed!');
END;
```

Command:

```
BEGIN
INSERT INTO order_products (Orderid,Productid,Quantity,Sellerid,OriginalAmount,Discount,Rating) values("140" , "12P" , 1 , '4S' , 400 , 100 , 3.9);
 END
```

Output:



2. Create a trigger to update product rating and seller rating when a new entry in the order_products table is inserted.

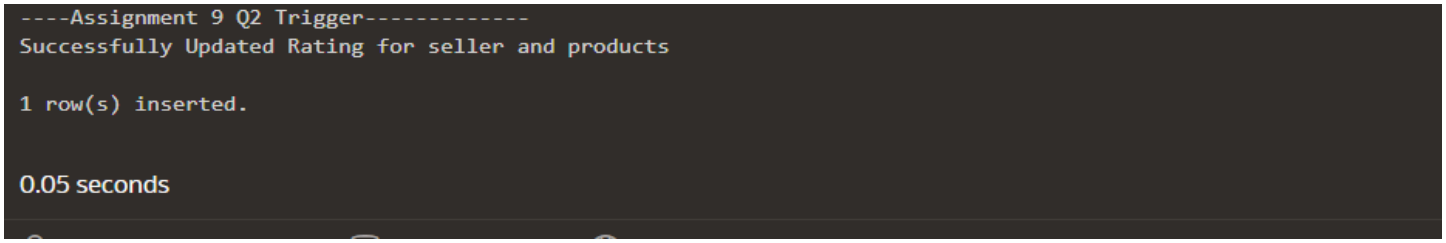
SCRIPT:

```
CREATE OR REPLACE TRIGGER set_rating AFTER INSERT ON order_products
BEGIN DBMS_OUTPUT.PUT_LINE("----Assignment 9 Q2 Trigger ------------------ ");
UPDATE product p SET p.rating = (SELECT AVG(rating) FROM order_products GROUP BY productid HAVING productid = p.productid);
UPDATE seller s SET s.rating = (SELECT AVG(rating) FROM order_products GROUP BY sellerid HAVING sellerid=s.sellerid);
dbms_output.put_line('Successfully Updated Rating for seller and products');
IF SQL%ROWCOUNT=0 THEN DBMS_OUTPUT.PUT_LINE('No rows affected");
 END IF;
 END set_rating;
```

COMMAND:

```
BEGIN INSERT INTO order_products (Orderid,Productid,Quantity,Sellerid,OriginalAmount,Discount,Rating) values('15O' , '4P' , 1 , '6S' , 400 , 100 , 2.3); END
```

OUTPUT:



3. Create a trigger to check when a new entry is to be inserted in the order_products table the quantity column satisfies the remaining quantity column from the product table.

SCRIPT:

```
CREATE OR REPLACE TRIGGER check_quantity BEFORE INSERT ON order_products
FOR EACH ROW
 ENABLE
 DECLARE tmp PRODUCT.quantityremaining%type;
 BEGIN DBMS_OUTPUT.PUT_LINE("----Assignment 9 Q3 Trigger ------------------ ");
 SELECT quantityremaining INTO tmp FROM PRODUCT WHERE productid=:new.productid;
 IF (:new.QUANTITY < tmp ) THEN
UPDATE product SET QuantityRemaining = QuantityRemaining-
:NEW.QUANTITY where productid=:NEW.productid;
dbms_output.put_line('New entry satisfies remaining quantity");
ELSE dbms_output.put_line('New entry does not satisfy remaining quantity');
END IF;
IF SQL%ROWCOUNT=0
THEN dbms_output.put_line('No row affected");
END IF;
END check_quantity;
```

COMMAND:

```
BEGIN INSERT INTO order_products (Orderid,Productid,Quantity,Sellerid,OriginalAmount,Discount
,Rating) values('16O' , '7P' , 6 , '7S' , 600 , 100 , 3.6); END
```

Output:

Results | Explain | Describe | Saved SQL | History

```
----Assignment 9 Q3 Trigger-------------
New entry does not satisfy remaining quantity
```