DBMS ASSIGNMENT – 9

**Cursors and Triggers**

***Name: BHAGYA VINOD RANA Roll Number: U19CS012***

**(A) Cursors:**

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

Cursor:

DECLARE

-- Variables to Hold Data

s\_id SELLER.SELLER\_ID%TYPE;

cus\_id CUSTOMER.CUSTOMER\_ID%TYPE;

-- CURSOR to Count the Number of Sellers

CURSOR seller\_cnt IS

SELECT

    DISTINCT SELLER\_ID

FROM

    SELLER;

-- CURSOR to Count the Number of Customers

CURSOR customer\_cnt IS

SELECT

    DISTINCT CUSTOMER\_ID

FROM

    CUSTOMER;

BEGIN

OPEN seller\_cnt;

    LOOP FETCH seller\_cnt INTO s\_id;

    EXIT WHEN seller\_cnt%notfound;

    END LOOP;

    -- Print the Seller Count after Iterating Whole SELLER Table

    dbms\_output.put\_line('Sellers Count : ' || seller\_cnt%rowcount);

CLOSE seller\_cnt;

OPEN customer\_cnt;

    LOOP FETCH customer\_cnt INTO cus\_id;

    EXIT WHEN customer\_cnt%notfound;

    END LOOP;

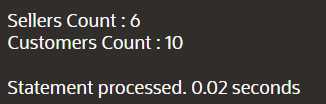
     -- Print the Customer Count after Iterating Whole SELLER Table

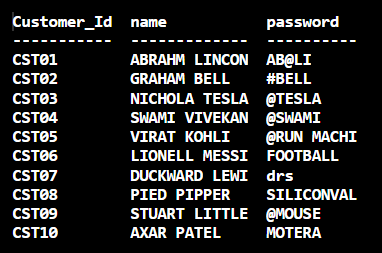
    dbms\_output.put\_line('Customers Count : ' || customer\_cnt%rowcount);

CLOSE customer\_cnt;

END;/

Output:



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

Cursor:

DECLARE

-- Variables to Hold Data

prod\_prodid PRODUCT.PRODUCT\_ID%TYPE;

prod\_name PRODUCT.PRODUCT%TYPE;

prod\_amt PRODUCT.AMOUNT%TYPE;

prod\_quant PRODUCT.QUANTITY\_REM%TYPE;

prod\_catid PRODUCT.CATEGORY\_ID%TYPE;

prod\_sellerid PRODUCT.SELLER\_ID%TYPE;

prod\_rating PRODUCT.RATING%TYPE;

-- CURSOR

CURSOR prod\_details IS

SELECT PRODUCT\_ID,PRODUCT,AMOUNT,QUANTITY\_REM,CATEGORY\_ID,SELLER\_ID,RATING

FROM

    PRODUCT

WHERE

    RATING IS NOT NULL AND RATING>3.5;

BEGIN

    dbms\_output.put\_line( 'P\_ID' || ' | ' || 'PRODUCT' || ' | ' || 'AMOUNT' || ' | ' || 'QUANTITY' || ' | ' || 'CAT\_ID' || ' | ' || 'SELLER\_ID' || ' | ' || 'RATING');

OPEN prod\_details;

    -- Loop to Print the Output

    LOOP FETCH prod\_details INTO prod\_prodid, prod\_name, prod\_amt, prod\_quant, prod\_catid, prod\_sellerid, prod\_rating;

    EXIT WHEN prod\_details%NOTFOUND;

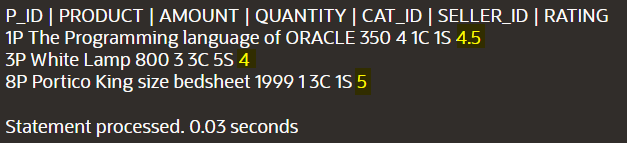
    dbms\_output.put\_line( prod\_prodid || ' ' || prod\_name || ' ' || prod\_amt || ' ' || prod\_quant || ' ' || prod\_catid || ' ' || prod\_sellerid || ' ' || prod\_rating);

    END LOOP;

CLOSE prod\_details;

END;/

Output:



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

Cursor:

DECLARE

-- Variables to Hold Data

prod\_prodid PRODUCT.PRODUCT\_ID%TYPE;

prod\_name PRODUCT.PRODUCT%TYPE;

prod\_amt PRODUCT.AMOUNT%TYPE;

prod\_quant PRODUCT.QUANTITY\_REM%TYPE;

prod\_catid PRODUCT.CATEGORY\_ID%TYPE;

prod\_sellerid PRODUCT.SELLER\_ID%TYPE;

prod\_rating PRODUCT.RATING%TYPE;

-- CURSOR

CURSOR prod\_details IS

SELECT PRODUCT\_ID,PRODUCT,AMOUNT,QUANTITY\_REM,CATEGORY\_ID,SELLER\_ID,RATING

FROM

    PRODUCT

ORDER BY

    CATEGORY\_ID;

BEGIN

    dbms\_output.put\_line( 'P\_ID' || ' | ' || 'PRODUCT' || ' | ' || 'AMOUNT' || ' | ' || 'QUANTITY' || ' | ' || 'CAT\_ID' || ' | ' || 'SELLER\_ID' || ' | ' || 'RATING');

OPEN prod\_details;

    -- Loop to Print the Output

    LOOP FETCH prod\_details INTO prod\_prodid, prod\_name, prod\_amt, prod\_quant, prod\_catid, prod\_sellerid, prod\_rating;

    EXIT WHEN prod\_details%NOTFOUND;

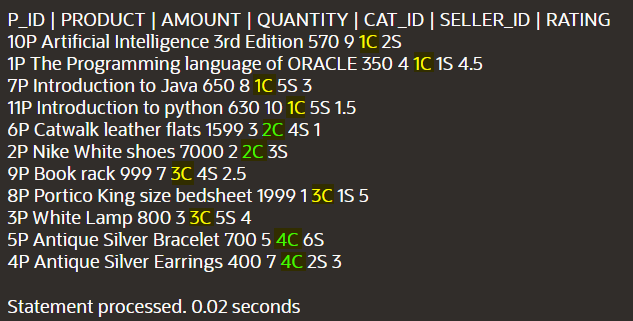
    dbms\_output.put\_line( prod\_prodid || ' ' || prod\_name || ' ' || prod\_amt || ' ' || prod\_quant || ' ' || prod\_catid || ' ' || prod\_sellerid || ' ' || prod\_rating);

    END LOOP;

CLOSE prod\_details;

END;/

Output:



**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.

Trigger:

CREATE OR REPLACE TRIGGER qty\_trigger

AFTER INSERT ON

    ORDER\_PRODUCT

FOR EACH ROW

BEGIN

    UPDATE

        PRODUCT

    SET

        QUANTITY\_REM = QUANTITY\_REM - :NEW.QUANTITY

    WHERE

        QUANTITY\_REM > 0 AND PRODUCT\_ID = :NEW.PRODUCT\_ID;

    IF SQL%ROWCOUNT = 0 THEN

        dbms\_output.put\_line('No Row Updated! [Update was Triggered]');

    ELSE

        dbms\_output.put\_line('Remaining Quantity of Product Updated! [Update Triggered]');

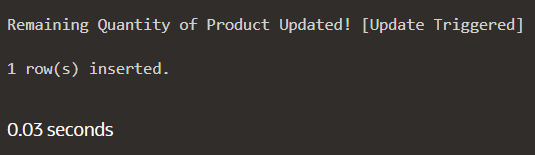
    END IF;

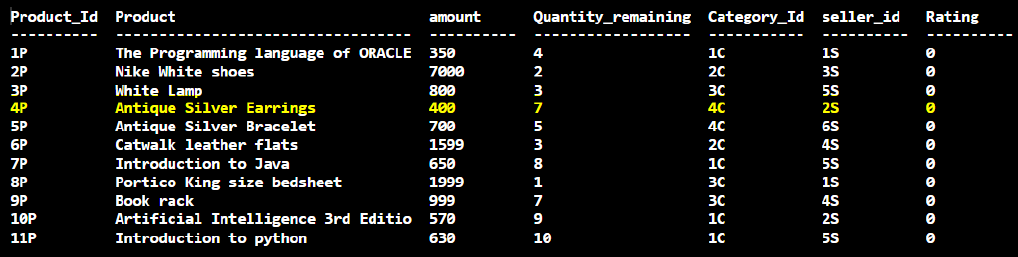
END qty\_trigger;

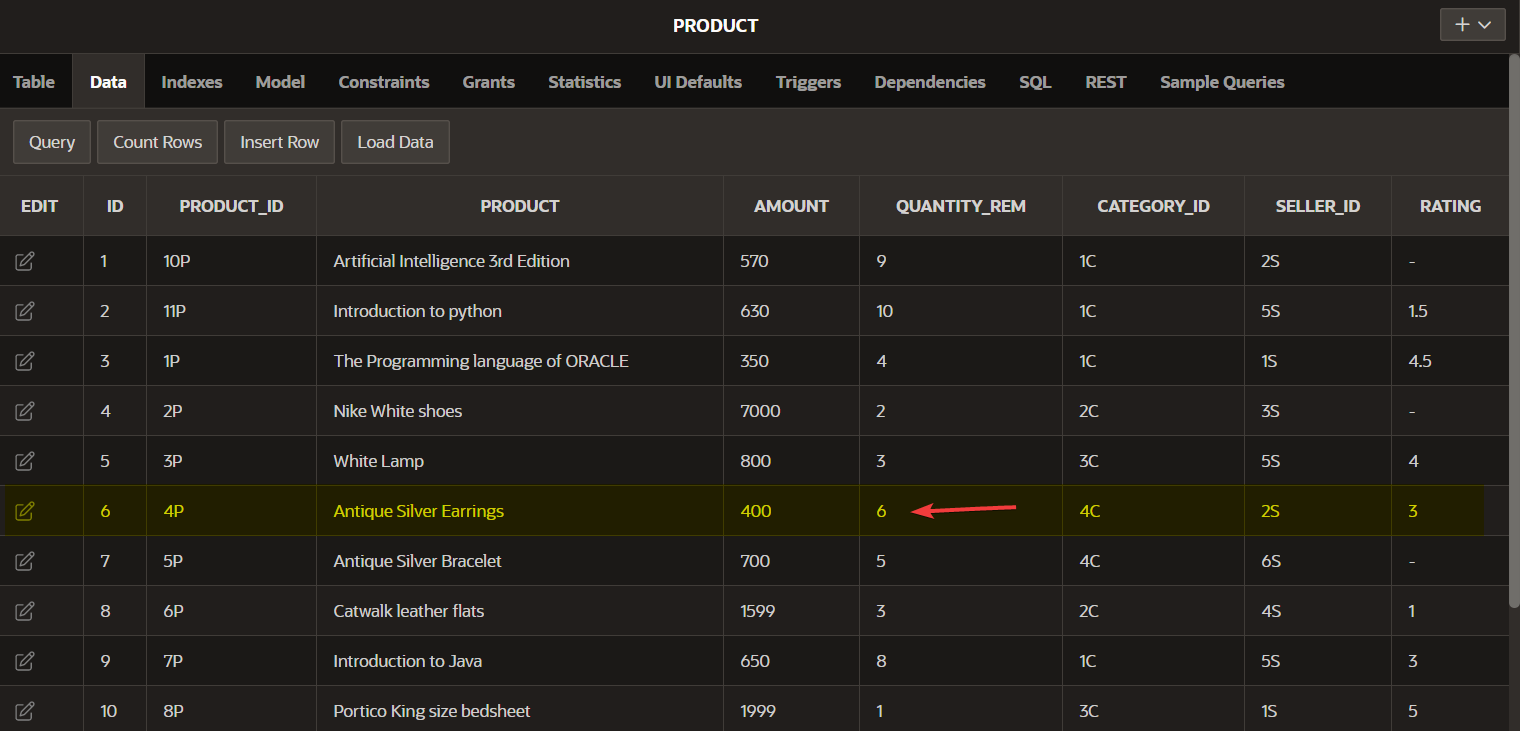
**Test:**

--Lets Order on Product with Product ID 4P and after this Order the Quantity in Product Table Should Decrease!

INSERT INTO ORDER\_PRODUCT(ORDER\_ID, PRODUCT\_ID, QUANTITY, SELLER\_ID, ORIGINAL\_AMT, DISCOUNT, PROD\_RATING) VALUES('6O' , '4P' , 1 , '2S' , 400 , 0 , 4);







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

order\_products table is inserted.

Trigger:

CREATE OR REPLACE TRIGGER rating\_trigger

AFTER INSERT ON ORDER\_PRODUCT

BEGIN

-- Update Product Rating

UPDATE

   product p

SET

   p.rating =

   (

      SELECT

         AVG(prod\_rating)

      FROM

         order\_product

      GROUP BY

         product\_id

      HAVING

         product\_id = p.product\_id

   );

-- Update Seller Rating

UPDATE

   seller s

SET

    s.rating = (

    SELECT

        AVG(prod\_rating)

    FROM

        order\_product

    GROUP BY

        seller\_id

    HAVING

        seller\_id = s.seller\_id

   );

-- Output Update Info

    IF SQL%ROWCOUNT = 0 THEN

        dbms\_output.put\_line('No Updates Made in Database!');

    ELSE

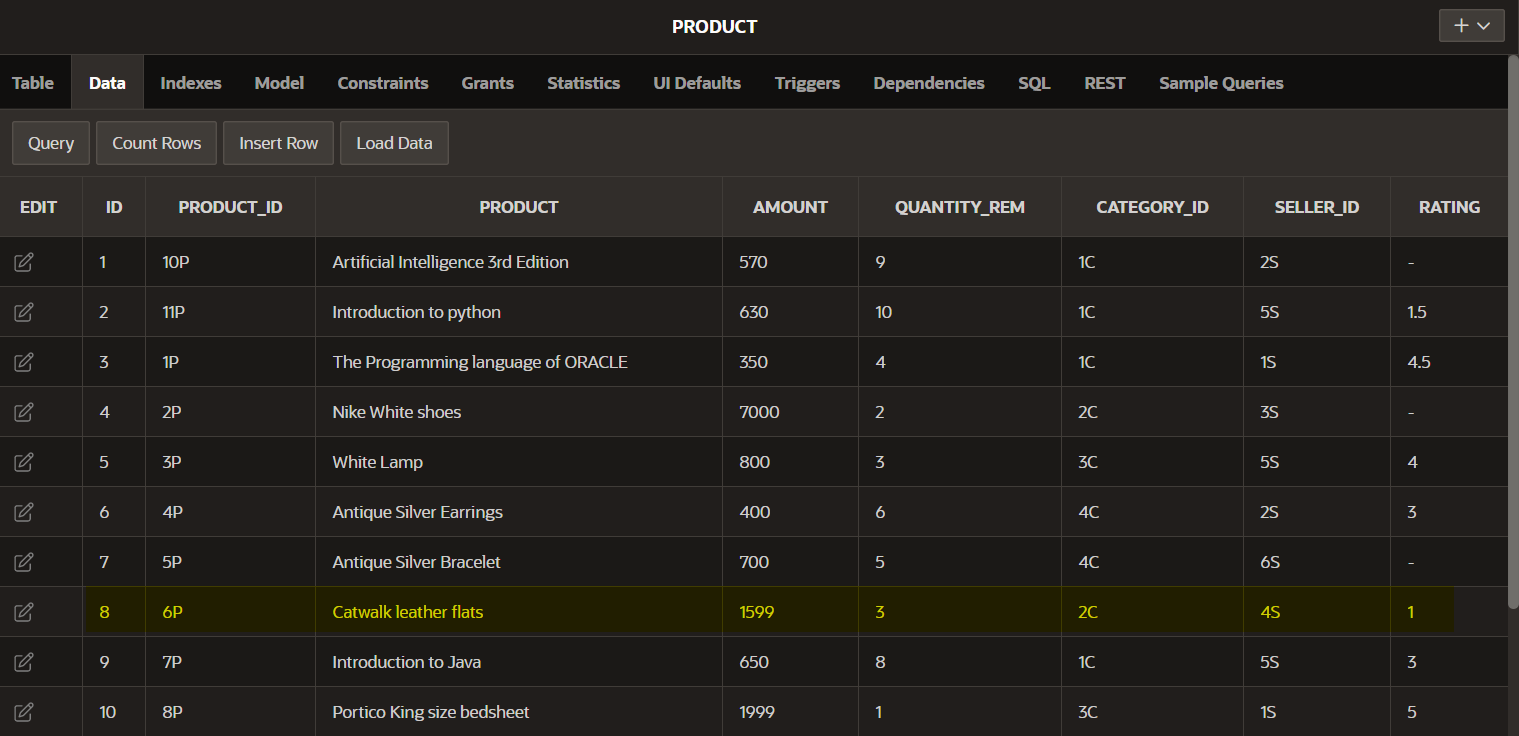
        dbms\_output.put\_line('Product Rating and Seller Rating Updated Successfully! [Update Triggered]');

    END IF;

END rating\_trigger;

**Test:**

**Before Query:**

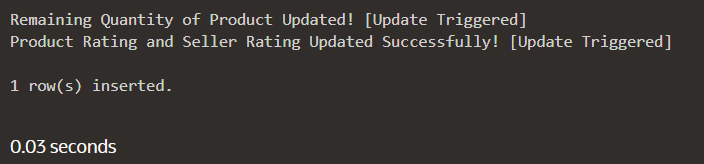


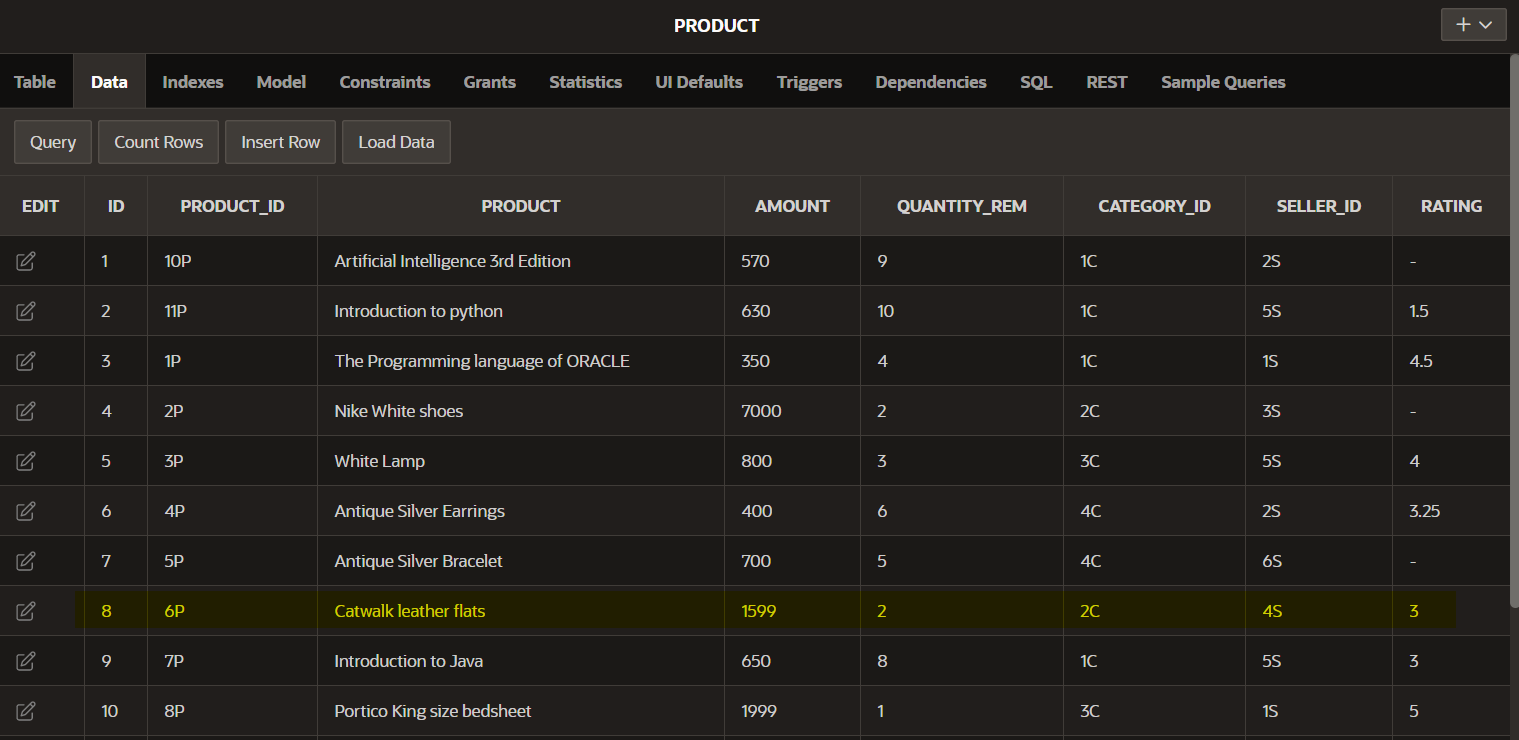


**Test:**

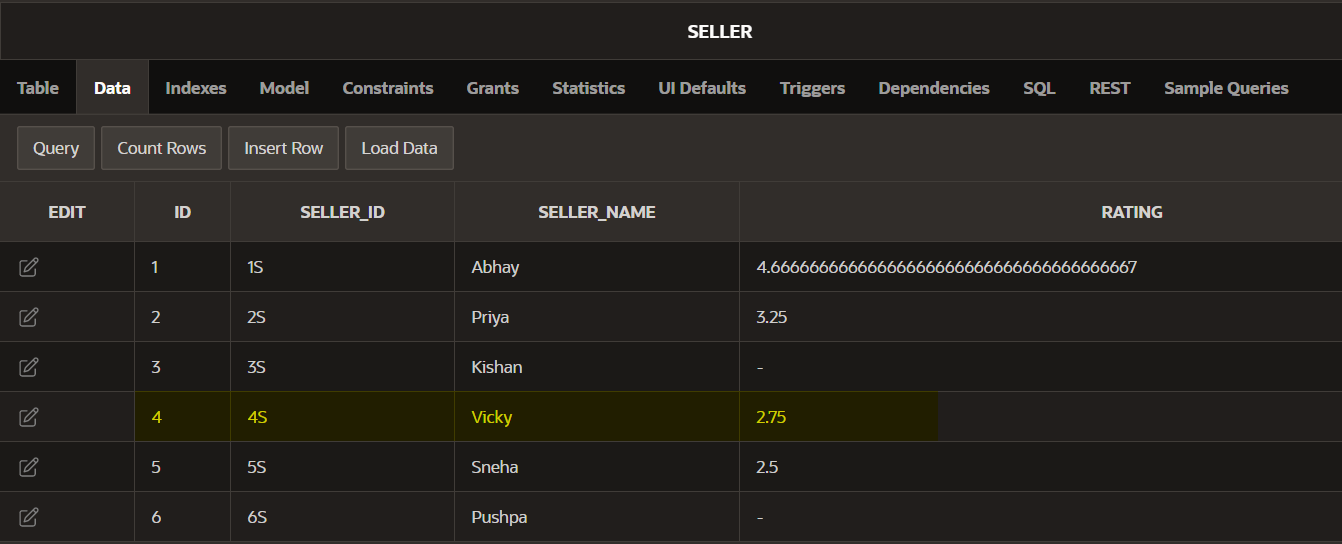
-- Lets Update the Rating of Product 6P which is Sold by 4S Seller To 5 Star

INSERT INTO ORDER\_PRODUCT(ORDER\_ID, PRODUCT\_ID, QUANTITY, SELLER\_ID, ORIGINAL\_AMT, DISCOUNT, PROD\_RATING) VALUES('12O' , '6P' , 1 , '4S' , 1599 , 0 , 5);





(1+5)/2 = 3 <= New Rating of Product



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.

Trigger:

CREATE OR REPLACE TRIGGER qty\_check

AFTER INSERT

ON ORDER\_PRODUCT

FOR EACH ROW

-- To Store Current Quantity & Compare with Given Input One

DECLARE

    tmp PRODUCT.QUANTITY\_REM%TYPE;

BEGIN

    SELECT

        QUANTITY\_REM INTO tmp

    FROM

        PRODUCT

    WHERE

        product\_id = :NEW.product\_id;

    IF (tmp > :NEW.QUANTITY) THEN

        UPDATE PRODUCT SET QUANTITY\_REM = QUANTITY\_REM - :NEW.QUANTITY;

        dbms\_output.put\_line('Remaining Quantity is Satisfied by New Order!');

    ELSE

        dbms\_output.put\_line('Quantity Ordered is More than Stock Available!');

    END IF;

    IF SQL%ROWCOUNT = 0 THEN

        dbms\_output.put\_line('No row affected');

    ELSE

        dbms\_output.put\_line('Remaining Quantity Check Trigger is Successful!');

    END IF;

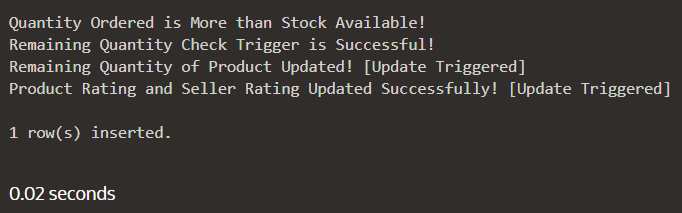
END qty\_check;

**Test:**

-- Lets Order More than Quantity Available for Product 7P which is Sold by Seller 5S

INSERT INTO ORDER\_PRODUCT(ORDER\_ID, PRODUCT\_ID, QUANTITY, SELLER\_ID, ORIGINAL\_AMT, DISCOUNT, PROD\_RATING) VALUES('15O' , '7P' , 15 , '5S' , 650 , 0 , 3);

**Output:**



**Submitted By:**

**BHAGYA VINOD RANA**

**U19CS012**