DBMS ASSIGNMENT – 8

**Functions and Stored Procedures**

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

Q)Considering the Tables of Assignment 7, create Stored Procedure and Functions as required.

I created the Database in XAMPP & **Imported** the Tables in Apex Oracle Website.

1.) Create a **Function** which returns the seller's name with the highest rating.

Function/Stored Procedure:

-- 1.) Create a Function which returns the seller's name with the highest rating.

CREATE OR REPLACE FUNCTION seller\_max\_rating RETURN SELLER.SELLER\_NAME %TYPE IS ans SELLER.SELLER\_NAME %TYPE;

-- Main Execution Part

BEGIN

   SELECT

      SELLER\_NAME INTO ans -- SELLER\_NAME stored in ans Variable

   FROM

      SELLER

   WHERE

      RATING =

      (

         SELECT

            MAX(RATING)

         FROM

            SELLER

      )

;

-- Return SELLER NAME with Max Rating via ans

RETURN ans;

END;

Test:

-- Declare a Variable 'ans' to Store the Result of Function of Data Type "SELLER.SELLER\_NAME"

DECLARE ans SELLER.SELLER\_NAME %TYPE;

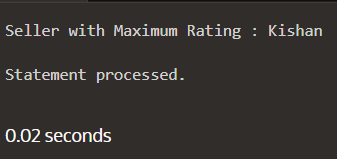
BEGIN

    ans := seller\_max\_rating;   -- Function Returns the Name of Seller with highest Rating

    dbms\_output.put\_line('Seller with Maximum Rating : ' || ans);

END;

Output:

2.) Create **Stored procedure** which takes as an input ‘category’ and outputs all the products of that category.

Function/Stored Procedure:

-- Function that Takes Category and List Down all Products

CREATE OR REPLACE PROCEDURE get\_all\_products (category\_input IN CATEGORY.CATEGORY %TYPE) IS c\_prod product.product %TYPE;

-- CURSOR -> To Retrieve Data [1 Row at a Time]

CURSOR c\_product IS

SELECT

   PRODUCT

FROM

   PRODUCT

WHERE

   CATEGORY\_ID =

   (

      SELECT

         CATEGORY\_ID

      FROM

         CATEGORY

      WHERE

         CATEGORY = category\_input

   )

;

BEGIN

OPEN c\_product;

    -- Loop that will Print All Products of that Category

    LOOP FETCH c\_product INTO c\_prod;

    EXIT WHEN

    c\_product % notfound;

    dbms\_output.put\_line(c\_prod);

    END

    LOOP;

CLOSE c\_product;

END;

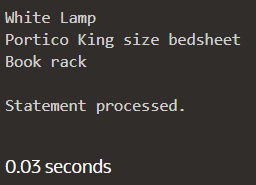
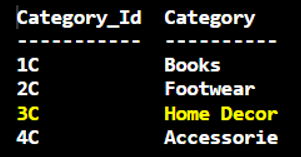
Test:

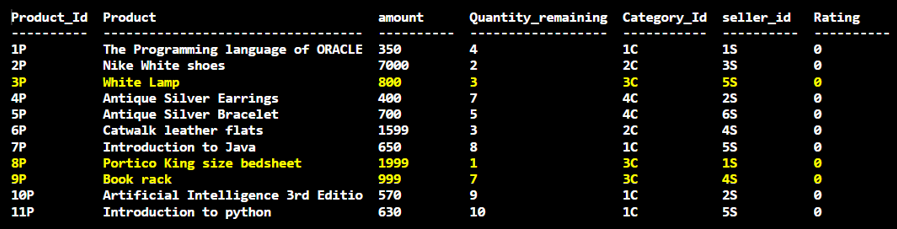
BEGIN

   get\_all\_products('Home Decor');

END;

Output:



3.) Create **Stored procedure** to take a range of prices as input and output all the products in the provided range.

Function/Stored Procedure:

-- Take Input Two Parameters [low & high] and Display all Product within that Range

CREATE OR REPLACE PROCEDURE prod\_in\_range(low\_lmt IN PRODUCT.AMOUNT %TYPE, up\_lmt IN PRODUCT.AMOUNT %TYPE) IS c\_prod product.product %TYPE;

-- Retrieve Data [1 Row at a Time]

CURSOR c\_product IS

SELECT

  PRODUCT

FROM

  PRODUCT

WHERE

  AMOUNT BETWEEN low\_lmt AND up\_lmt;

BEGIN

OPEN c\_product;

  LOOP FETCH c\_product INTO c\_prod;

    EXIT WHEN

    c\_product %notfound;

    dbms\_output.put\_line(c\_prod);

  END LOOP;

CLOSE c\_product;

END;

Test:

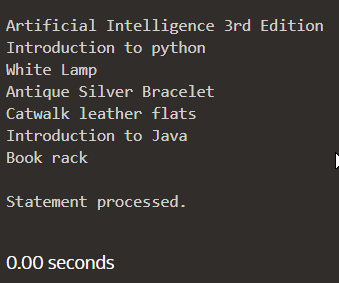
-- Function CALL to Stored Procedure

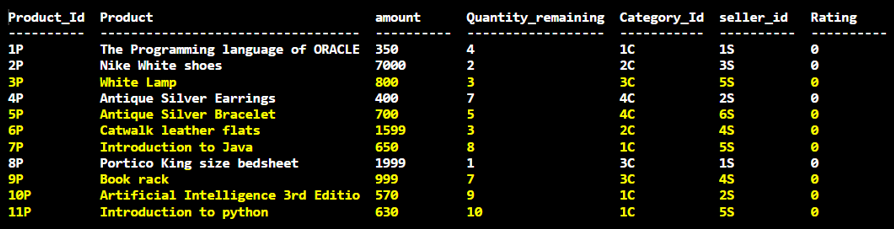
BEGIN

    prod\_in\_range(550,1700);

END;

Output:





4.) Create **Function** to display all the seller details with rating more than 3.

Function/Stored Procedure:

- returning a sys\_refcursor you allow the client to fetch as many or few of the rows from the query as it requires [STACKOVERFLOW]

CREATE OR REPLACE FUNCTION get\_good\_rated\_sellers RETURN SYS\_REFCURSOR IS s\_details SYS\_REFCURSOR;

BEGIN

OPEN s\_details FOR

  SELECT DISTINCT   --Avoid Duplicate Entries

    SELLER\_ID,

    SELLER\_NAME,

    RATING

  FROM

    SELLER

  WHERE

    RATING > 3;

RETURN s\_details;

END;

Test:

-- Declare all Necessary Variables

DECLARE s\_details SYS\_REFCURSOR;

s\_id SELLER.SELLER\_ID %type;

s\_name SELLER.SELLER\_name %type;

s\_rating SELLER.rating %type;

BEGIN

    s\_details := get\_good\_rated\_sellers;

    dbms\_output.put\_line('  SELLER\_ID   |   SELLER\_NAME |   SELLER\_RATING   ');

    -- Loop to Display the Output

    LOOP FETCH s\_details INTO s\_id, s\_name, s\_rating;

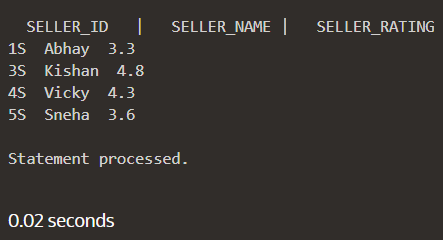
    EXIT WHEN s\_details % NOTFOUND;

    dbms\_output.put\_line(s\_id || '  ' || s\_name || '  ' || s\_rating);

    END LOOP;

END;

Output:

5.) Create a **Function** to display all the products, seller wise.

Function/Stored Procedure:

-- fetch as many or few of the rows from the query as it requires = SYS\_REFCURSOR

CREATE OR REPLACE FUNCTION display\_products\_seller\_wise RETURN SYS\_REFCURSOR IS prods SYS\_REFCURSOR;

BEGIN

    OPEN prods FOR

    SELECT PRODUCT, SELLER\_ID

    FROM PRODUCT SELLER

    ORDER BY SELLER\_ID;

    RETURN prods;

END;

Test:

-- Declare all Necessary Variables

DECLARE details SYS\_REFCURSOR;

p\_name SELLER.SELLER\_name %type;

s\_id SELLER.SELLER\_ID %type;

BEGIN

    details := display\_products\_seller\_wise;

    dbms\_output.put\_line('S\_ID | PRODUCT\_NAME ');

    -- Loop to Display the Output

    LOOP FETCH details INTO p\_name,s\_id;

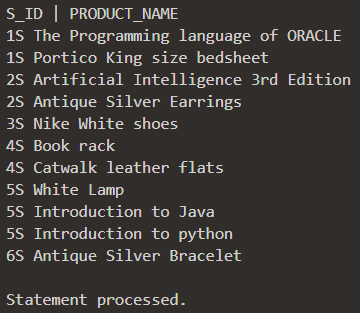
    EXIT WHEN details%NOTFOUND;

    dbms\_output.put\_line(s\_id || ' ' || p\_name);

    END LOOP;

END;

Output: [Seller-Wise Product List]



6.) Create a **Stored procedure** which checks all the entries in Order\_Products table and update seller and product table accordingly.

Function/Stored Procedure:

CREATE OR REPLACE PROCEDURE update\_product\_seller\_tables AS

BEGIN

    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 s

    SET

        s.rating = (

        SELECT AVG(prod\_rating)

        FROM order\_product

        GROUP BY seller\_id

        HAVING seller\_id = s.seller\_id

        );

END;

Test:

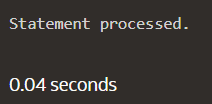
-- Function CALL to Stored Procedure

BEGIN

    update\_product\_seller\_tables;

END;

Output:



7.) Create **Stored procedure** which takes as input different filters such as price range, category, product rating, seller rating, out of stock and displays the list of products with all the details after applying filters

Function/Stored Procedure:

CREATE OR REPLACE PROCEDURE filter\_criteria(OPT IN NUMBER, FILTERING\_LIMIT IN varchar) IS prod\_details SYS\_REFCURSOR;

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;

BEGIN

-- Switch Case : [1 -> amount | 2 -> category | 3 -> product-rating | 4 -> seller-rating | 5 -> checking in stock]

CASE opt

    -- AMOUNT FILTER

    WHEN 1 THEN

         OPEN prod\_details FOR

         SELECT

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

         FROM

            PRODUCT

         WHERE

            AMOUNT < TO\_NUMBER(FILTERING\_LIMIT);

    -- CATEGORY FILTER

    WHEN 2 THEN

        OPEN prod\_details FOR

        SELECT

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

        FROM

            PRODUCT

        WHERE

            CATEGORY\_ID = FILTERING\_LIMIT;

    -- PRODUCT RATING FILTER

    WHEN 3 THEN

        OPEN prod\_details FOR

        SELECT

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

        FROM

            PRODUCT

        WHERE

            RATING >= TO\_NUMBER(FILTERING\_LIMIT);

    -- SELLER RATING FILTER

    WHEN 4 THEN

        OPEN prod\_details FOR

        SELECT

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

        FROM

            PRODUCT

        WHERE

        SELLER\_ID IN (

         SELECT

            SELLER\_ID

         FROM

            SELLER

         WHERE

            RATING >= TO\_NUMBER(FILTERING\_LIMIT)

        );

    -- CHECKING IN STOCK FILTER

    WHEN 5 THEN

        OPEN prod\_details FOR

        SELECT

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

        FROM

            PRODUCT

        WHERE

            QUANTITY\_REM <> 0;

END CASE;

-- Loop to Print 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;

END;

Test & Output:

-- Sorting Criteria :

[1 -> amount | 2 -> category | 3 -> product-rating | 4 -> seller-rating | 5 -> checking in stock]|[filtering limit]

BEGIN

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

    -- AMOUNT < 1000

    filter\_criteria(1, 1000);

    -- CATEGORY "1C"

    filter\_criteria(2, '1C');

    -- PRODUCT RATING >3

    filter\_criteria(3, 3);

    -- SELLER RATING >4

    filter\_criteria(4, 4);

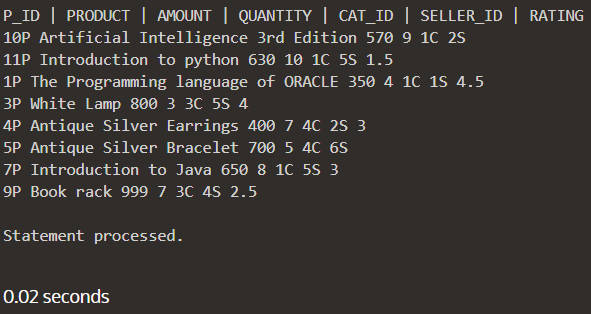
    -- STOCK AVAILABLE (2 nd Paramater Does Not Matter)

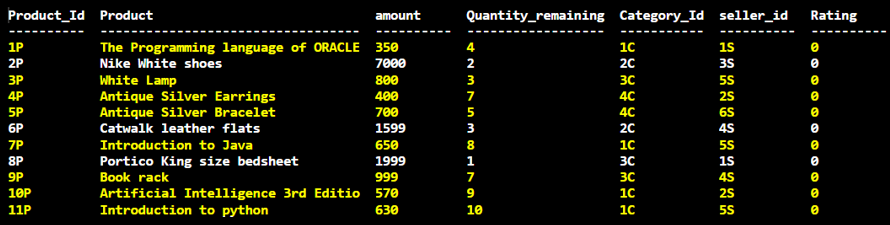
    filter\_criteria(5, 3);

END;

(A) Filter by **Amount**

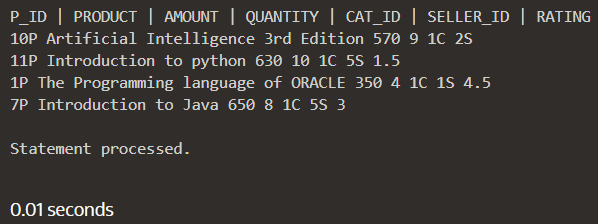
filter\_criteria(1, 1000);

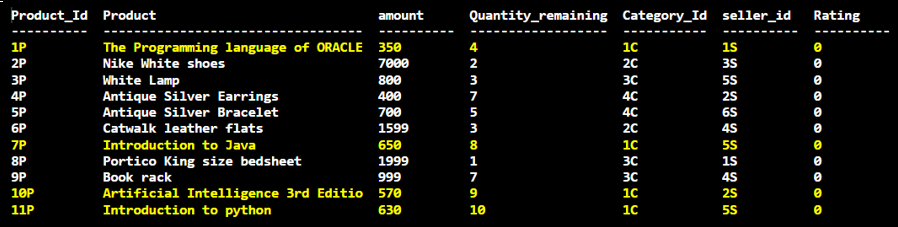




(B) Filter by **Category**

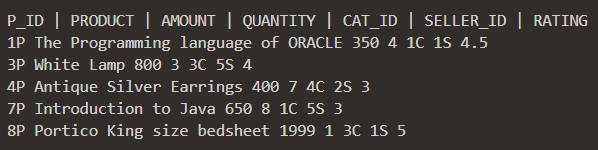
filter\_criteria(2, '1C');





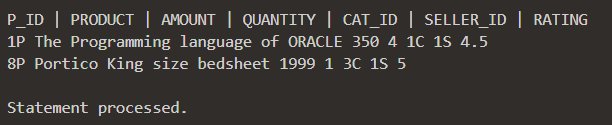
(C) Filter by **Product Rating**

filter\_criteria(3, 3);



(D) Filter by **Seller Rating**

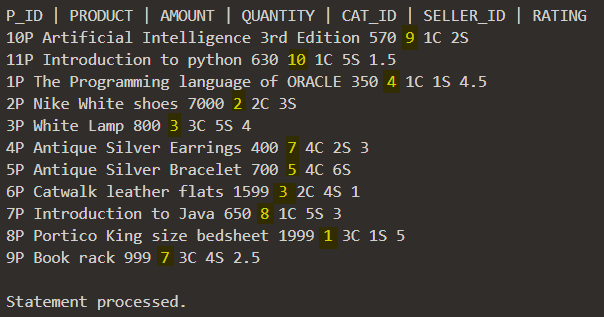
    filter\_criteria(4, 4);



(E) Filter by Stock Available

-- STOCK AVAILABLE (2 nd Paramater Does Not Matter)

    filter\_criteria(5, 3);



8.) Create a **Function** which takes as input sorting criteria like popularity or lowest price or highest price and display the product list accordingly.

Function/Stored Procedure:

CREATE OR REPLACE FUNCTION sort\_criteria(opt IN number) RETURN SYS\_REFCURSOR IS prod\_details SYS\_REFCURSOR;

    BEGIN CASE opt

        -- 1 : Price Lowest to Highest

        WHEN 1 THEN

            OPEN prod\_details FOR

            SELECT

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

            FROM

                PRODUCT

            ORDER BY

                AMOUNT;     -- By Default Ascending

        -- 2 : Price Highest to Lowest

        WHEN 2 THEN

            OPEN prod\_details FOR

            SELECT

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

            FROM

                PRODUCT

            ORDER BY

                AMOUNT DESC;

        -- 3 : Rating Lowest to Highest

        WHEN 3 THEN

            OPEN prod\_details FOR

            SELECT

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

            FROM

                PRODUCT

            ORDER BY

                RATING;     -- By Default Ascending

        -- 4 : Rating Highest to Lowest

        WHEN 4 THEN

            OPEN prod\_details FOR

            SELECT

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

            FROM

                PRODUCT

            ORDER BY

                RATING DESC;

   END CASE;

RETURN prod\_details;

END;

Test & Output:

-- Declare all Necessary Variables

DECLARE prod\_details SYS\_REFCURSOR;

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;

BEGIN

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

    -- 1 : Price Lowest to Highest

    prod\_details := sort\_criteria(1);

    -- 2 : Price Highest to Lowest

    prod\_details := sort\_criteria(2);

    -- 3 : Rating Lowest to Highest

    prod\_details := sort\_criteria(3);

    -- 4 : Rating Highest to Lowest

    prod\_details := sort\_criteria(4);

    -- 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;

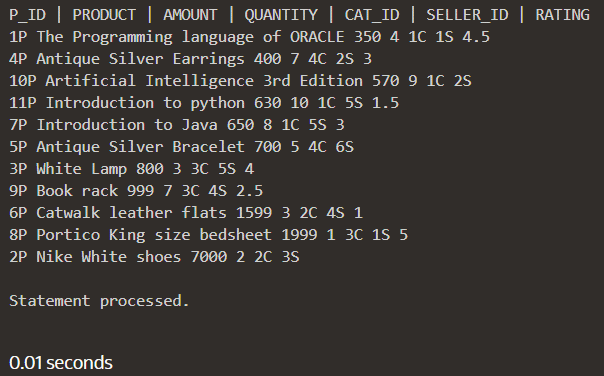
    dbms\_output.put\_line( prod\_prodid || ' ' || prod\_name || ' ' || prod\_amt || ' ' || prod\_quant || ' ' || prod\_catid || ' ' || prod\_sellerid || ' ' || prod\_rating);

    END LOOP;

END;

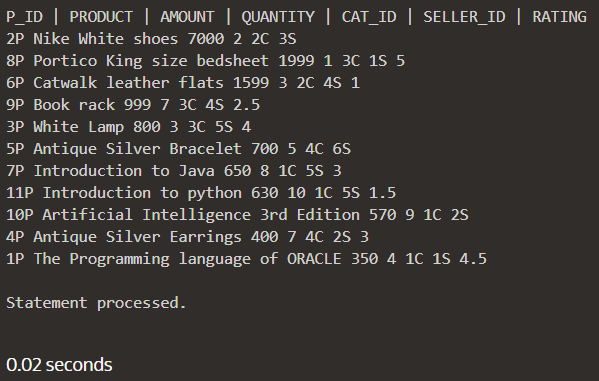
(A) Sort by Price [Lowest to Highest]

    prod\_details := sort\_criteria(1);



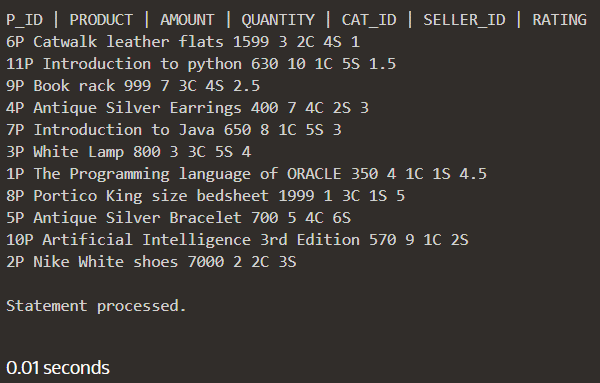
(B) Sort by Price [Highest to Lowest]

    prod\_details := sort\_criteria(2);



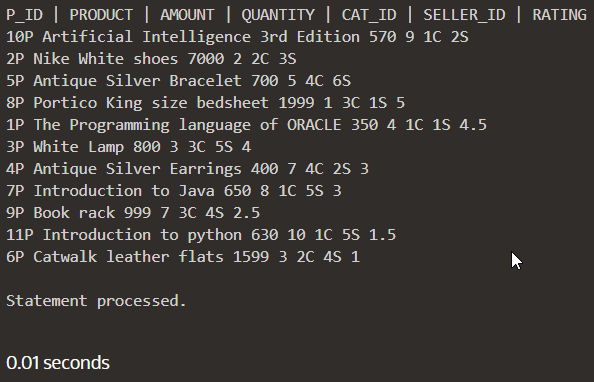
(C) Sort by Rating [Lowest to Highest]

    prod\_details := sort\_criteria(3);



(D) Sort by Rating [Highest to Lowest]

    prod\_details := sort\_criteria(4);



**Submitted By:**

**BHAGYA VINOD RANA**

**U19CS012**