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.

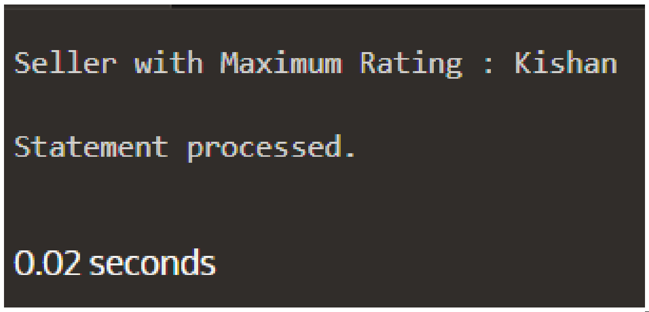
Function/Stored Procedure:

```
-- 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.SE
LLER_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;
```

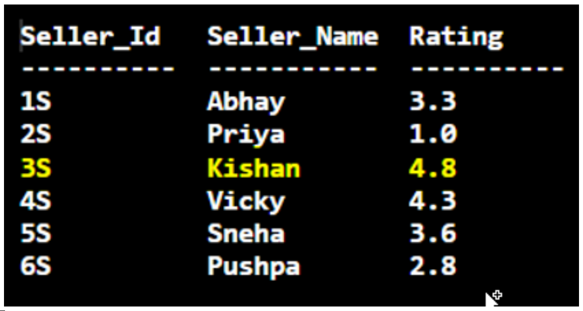
Test:

```
-- 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:



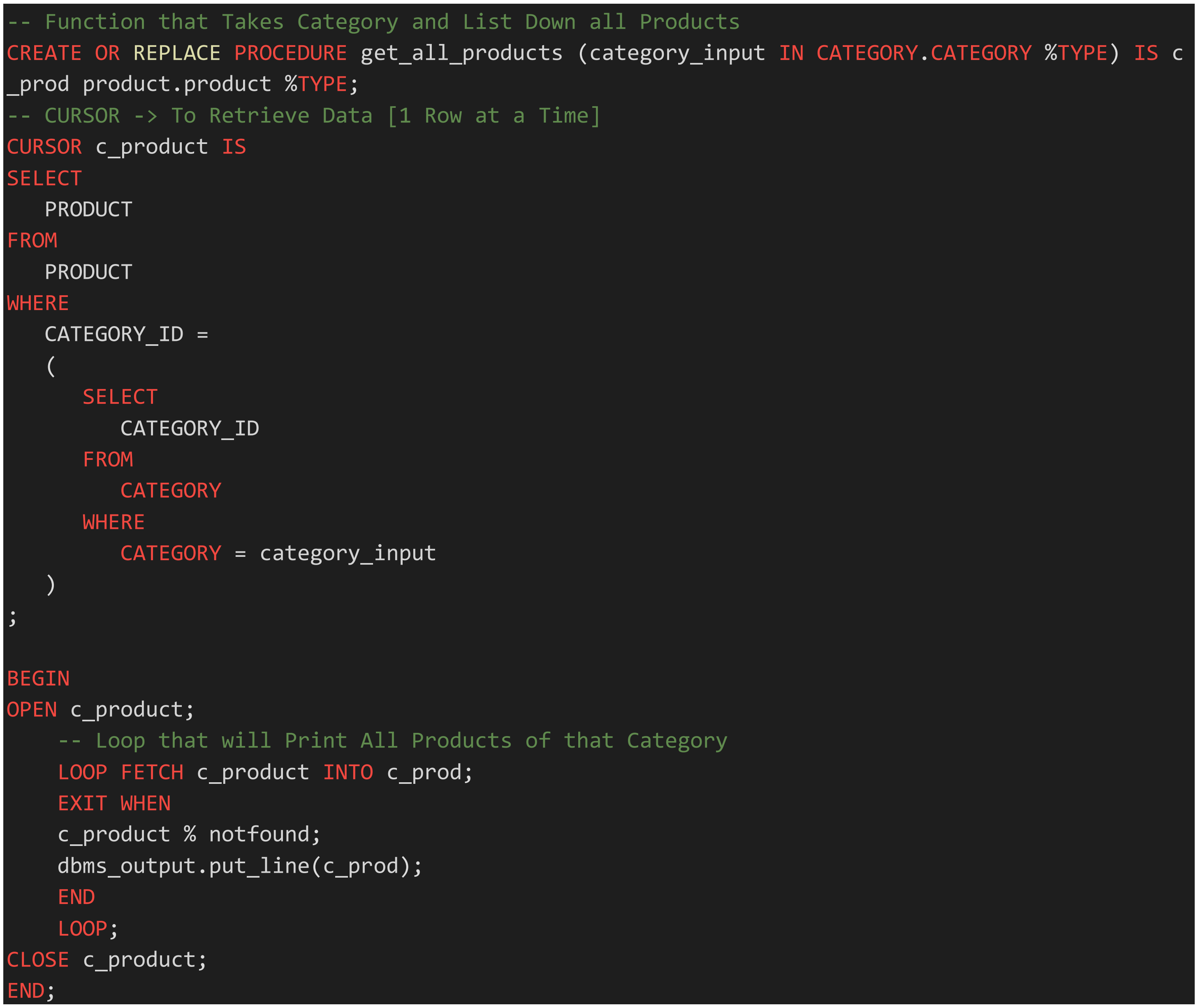
```
Seller with Maximum Rating : Kishan

Statement processed.

0.02 seconds
```



```
Seller_Id     Seller_Name  Rating
----------    -----------  ----------
1S            Abhay        3.3
2S            Priya        1.0
3S            Kishan       4.8
4S            Vicky        4.3
5S            Sneha        3.6
6S            Pushpa       2.8
```

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

Function/Stored Procedure:

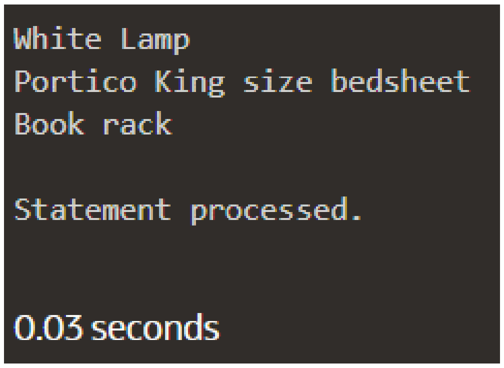


```
-- Function that Takes Category and List Down all Products
CREATE OR REPLACE PROCEDURE get_all_products (category_input IN CATEGORY.CATEGORY %TYPE) IS c
_prod product.product %TYPE;
-- CURSOR -> To Retrieve Data [1 Row at a Time]
CURSOR c_product IS
SELECT
   PRODUCT
FROM
   PRODUCT
WHERE
   CATEGORY_ID =
   (
      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;
```

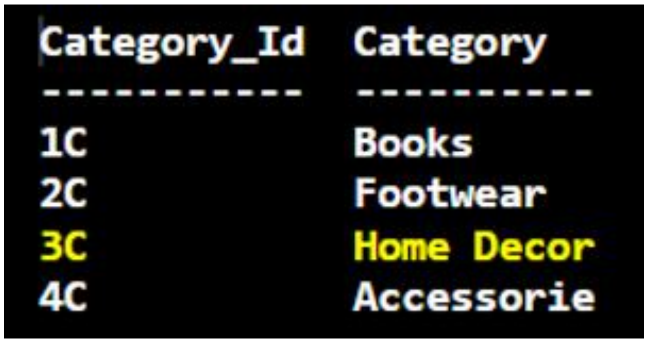
Test:

```
BEGIN
   get_all_products('Home Decor');
END;
```

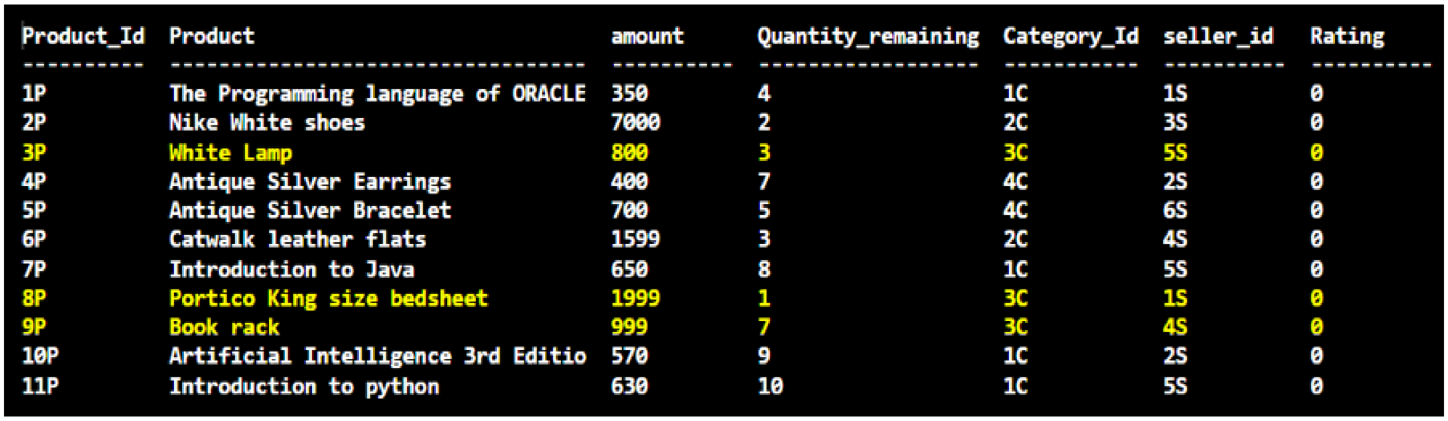
Output:



```
White Lamp
Portico King size bedsheet
Book rack

Statement processed.


0.03 seconds
```



```
Category_Id  Category
-----------  ----------
1C           Books
2C           Footwear
3C           Home Decor
4C           Accessorie
```



```
Product_Id Product                           amount     Quantity_remaining Category_Id seller_id  Rating
---------- --------------------------------- ---------- ------------------ ----------- ---------- ----------
1P         The Programming language of ORACLE 350       4                  1C          1S         0
2P         Nike White shoes                  7000       2                  2C          3S         0
3P         White Lamp                        800        3                  3C          5S         0
4P         Antique Silver Earrings           400        7                  4C          2S         0
5P         Antique Silver Bracelet           700        5                  4C          6S         0
6P         Catwalk leather flats             1599       3                  2C          4S         0
7P         Introduction to Java              650        8                  1C          5S         0
8P         Portico King size bedsheet        1999       1                  3C          1S         0
9P         Book rack                         999        7                  3C          4S         0
10P        Artificial Intelligence 3rd Editio 570       9                  1C          2S         0
11P        Introduction to python            630        10                 1C          5S         0
```

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

Function/Stored Procedure:

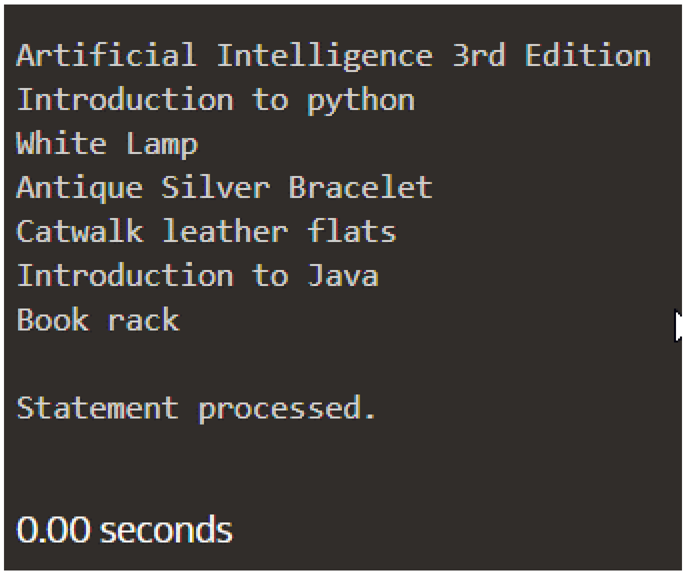
```
-- Take Input Two Parameters [low & high] and Display all Product within that Range
CREATE OR REPLACE PROCEDURE prod_in_range(low_lmt IN PRODUCT.AMOUNT %TYPE, up_lmt IN PRODUCT.
AMOUNT %TYPE) IS c_prod product.product %TYPE;

-- 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;
```

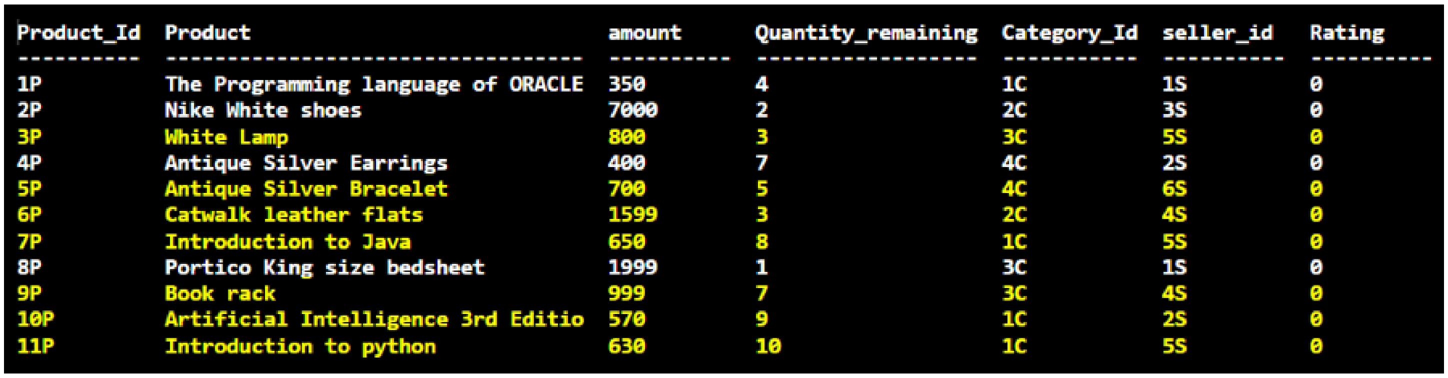
Test:

```
-- Function CALL to Stored Procedure
BEGIN
    prod_in_range(550,1700);
END;
```

Output:



```
Artificial Intelligence 3rd Edition
Introduction to python
White Lamp
Antique Silver Bracelet
Catwalk leather flats
Introduction to Java
Book rack

Statement processed.


0.00 seconds
```



```
Product_Id Product                            amount     Quantity_remaining Category_Id seller_id  Rating
---------- ---------------------------------- ---------- ------------------ ----------- ---------- ----------
1P         The Programming language of ORACLE 350        4                  1C          1S         0
2P         Nike White shoes                   7000       2                  2C          3S         0
3P         White Lamp                         800        3                  3C          5S         0
4P         Antique Silver Earrings            400        7                  4C          2S         0
5P         Antique Silver Bracelet            700        5                  4C          6S         0
6P         Catwalk leather flats              1599       3                  2C          4S         0
7P         Introduction to Java               650        8                  1C          5S         0
8P         Portico King size bedsheet         1999       1                  3C          1S         0
9P         Book rack                          999        7                  3C          4S         0
10P        Artificial Intelligence 3rd Editio 570        9                  1C          2S         0
11P        Introduction to python             630        10                 1C          5S         0
```

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

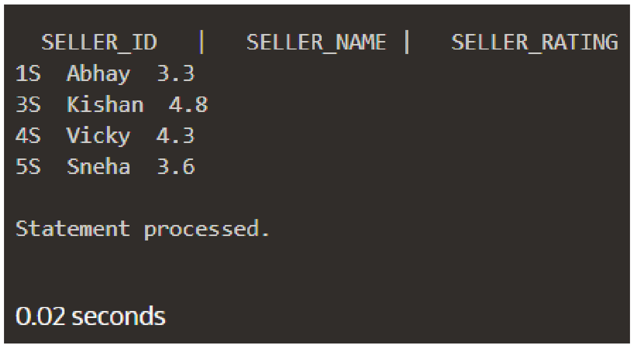
Function/Stored Procedure:

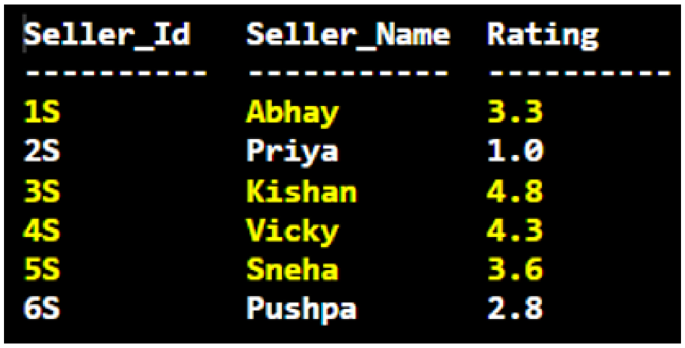
```
- returning a sys_refcursor you allow the client to fetch as many or few of the rows from the
 query as it requires [STACKOVERFLOW]

CREATE OR REPLACE FUNCTION get_good_rated_sellers RETURN SYS_REFCURSOR IS s_details SYS_REFCU
RSOR;

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:


```
  SELLER_ID   |   SELLER_NAME |   SELLER_RATING
1S  Abhay  3.3
3S  Kishan  4.8
4S  Vicky  4.3
5S  Sneha  3.6

Statement processed.

0.02 seconds
```


```
Seller_Id    Seller_Name  Rating
----------   -----------  ----------
1S           Abhay        3.3
2S           Priya        1.0
3S           Kishan       4.8
4S           Vicky        4.3
5S           Sneha        3.6
6S           Pushpa       2.8
```

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

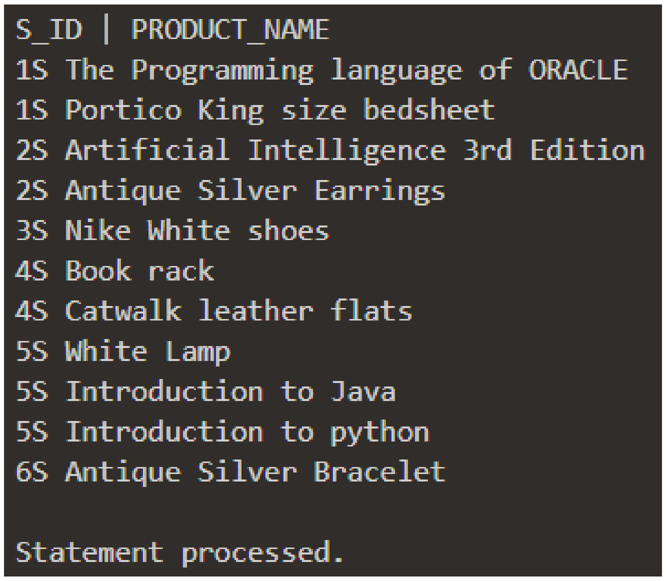
Function/Stored Procedure:

```
-- 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_REF
CURSOR;
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;
```

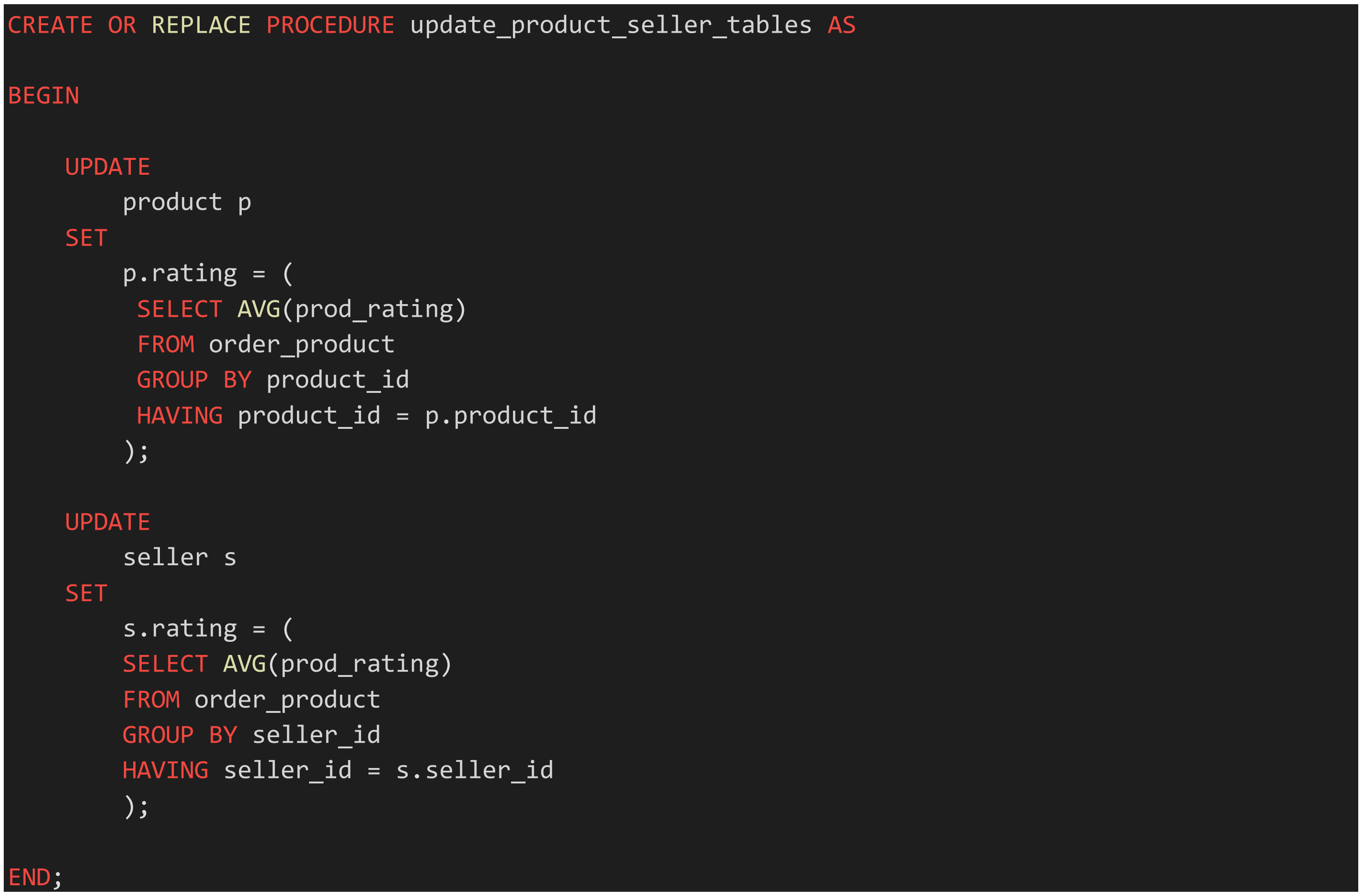
Output: [Seller-Wise Product List]



```
S_ID | PRODUCT_NAME
1S The Programming language of ORACLE
1S Portico King size bedsheet
2S Artificial Intelligence 3rd Edition
2S Antique Silver Earrings
3S Nike White shoes
4S Book rack
4S Catwalk leather flats
5S White Lamp
5S Introduction to Java
5S Introduction to python
6S Antique Silver Bracelet

Statement processed.
```

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

Function/Stored Procedure:

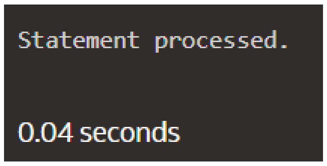


```
CREATE OR REPLACE PROCEDURE update_product_seller_tables AS

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:



```
Statement processed.

0.04 seconds
```

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 pro
d_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;
```

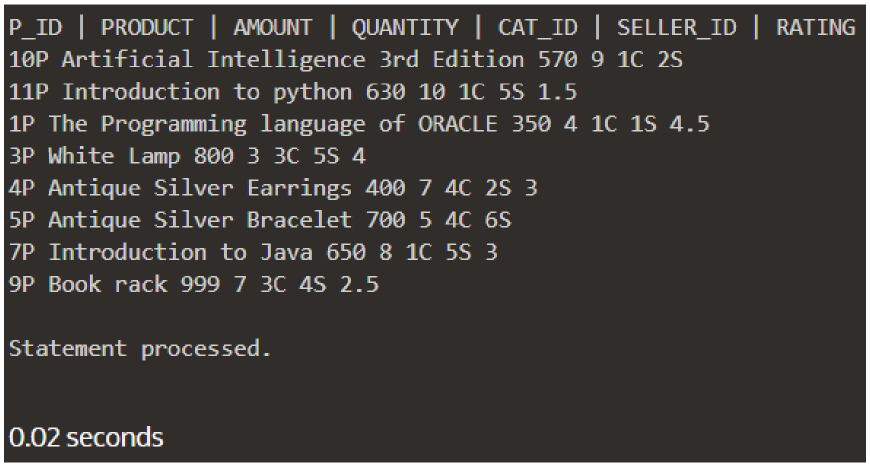
```
-- Loop to Print Output

    LOOP FETCH prod_details INTO prod_prodid, prod_name, prod_amt, prod_quant, prod_catid, pr
od_sellerid, prod_rating;
    EXIT WHEN prod_details%NOTFOUND;
    dbms_output.put_line( prod_prodid || ' ' || prod_name || ' ' || prod_amt || ' ' || prod_q
uant || ' ' || prod_catid || ' ' || prod_sellerid || ' ' || prod_rating);
    END LOOP;

END;
```

Test & Output:

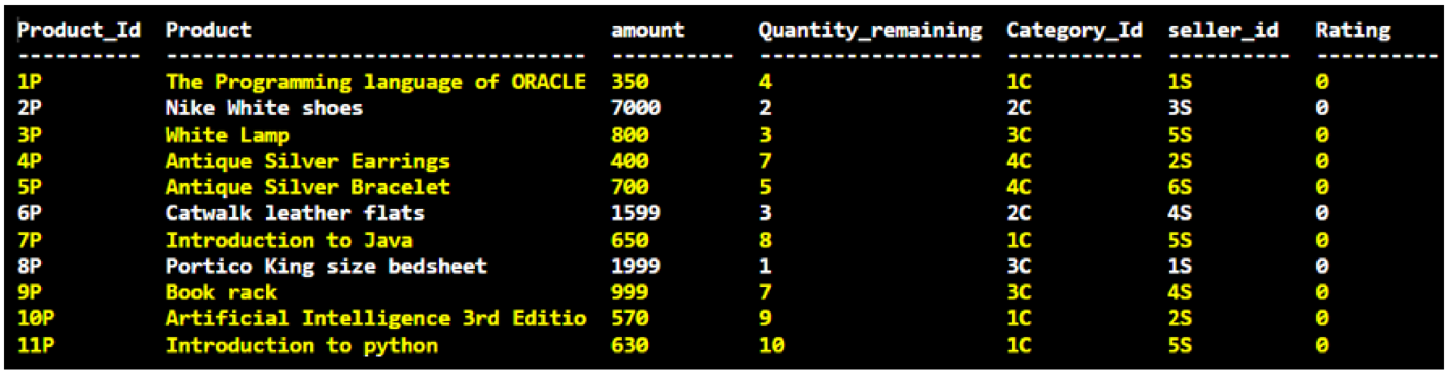
```
-- Sorting Criteria :
[1 -> amount | 2 -> category | 3 -> product-rating | 4 -> seller-rating | 5 -
> checking in stock]|[filtering limit]

BEGIN
    dbms_output.put_line( 'P_ID' || ' | ' || 'PRODUCT' || ' | ' || 'AMOUNT' || ' | ' || 'QUAN
TITY' || ' | ' || '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);
```

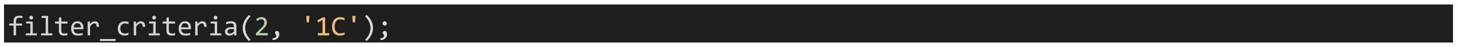


```
P_ID | PRODUCT | AMOUNT | QUANTITY | CAT_ID | SELLER_ID | RATING
10P Artificial Intelligence 3rd Edition 570 9 1C 2S
11P Introduction to python 630 10 1C 5S 1.5
1P The Programming language of ORACLE 350 4 1C 1S 4.5
3P White Lamp 800 3 3C 5S 4
4P Antique Silver Earrings 400 7 4C 2S 3
5P Antique Silver Bracelet 700 5 4C 6S
7P Introduction to Java 650 8 1C 5S 3
9P Book rack 999 7 3C 4S 2.5

Statement processed.


0.02 seconds
```

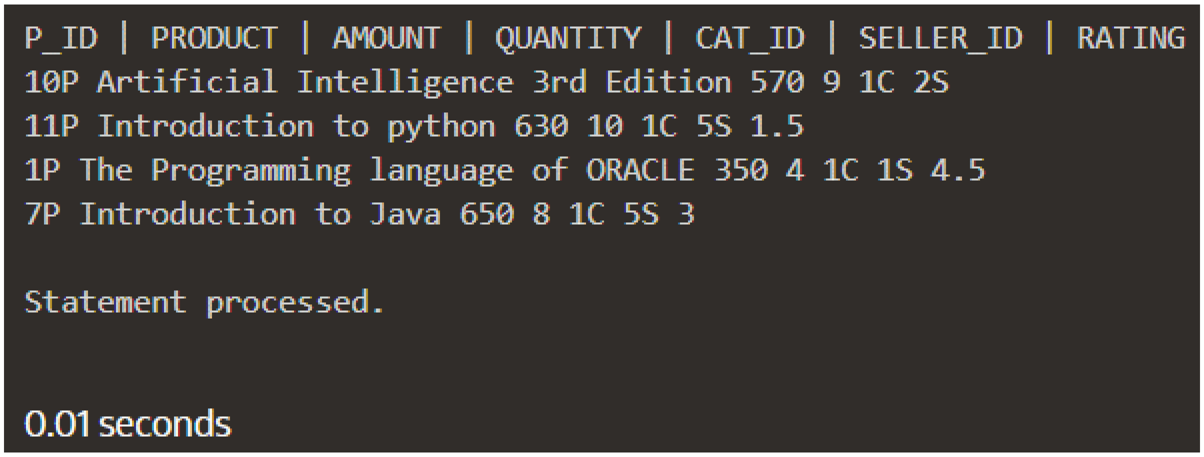


```
Product_Id Product                            amount     Quantity_remaining Category_Id seller_id  Rating
---------- ---------------------------------- ---------- ------------------ ----------- ---------- ----------
1P         The Programming language of ORACLE 350        4                  1C          1S         0
2P         Nike White shoes                   7000       2                  2C          3S         0
3P         White Lamp                         800        3                  3C          5S         0
4P         Antique Silver Earrings            400        7                  4C          2S         0
5P         Antique Silver Bracelet            700        5                  4C          6S         0
6P         Catwalk leather flats              1599       3                  2C          4S         0
7P         Introduction to Java               650        8                  1C          5S         0
8P         Portico King size bedsheet         1999       1                  3C          1S         0
9P         Book rack                          999        7                  3C          4S         0
10P        Artificial Intelligence 3rd Editio 570        9                  1C          2S         0
11P        Introduction to python             630        10                 1C          5S         0
```

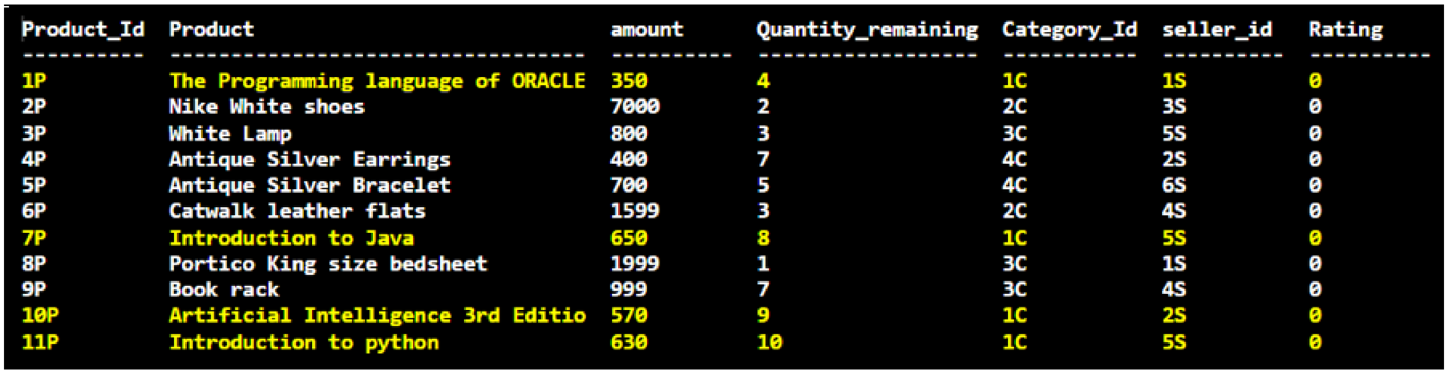
(B) Filter by **Category**



```
filter_criteria(2, '1C');
```



```
P_ID | PRODUCT | AMOUNT | QUANTITY | CAT_ID | SELLER_ID | RATING
10P Artificial Intelligence 3rd Edition 570 9 1C 2S
11P Introduction to python 630 10 1C 5S 1.5
1P The Programming language of ORACLE 350 4 1C 1S 4.5
7P Introduction to Java 650 8 1C 5S 3

Statement processed.


0.01 seconds
```



```
Product_Id Product                            amount     Quantity_remaining Category_Id seller_id  Rating
---------- ---------------------------------- ---------- ------------------ ----------- ---------- ----------
1P         The Programming language of ORACLE 350        4                  1C          1S         0
2P         Nike White shoes                   7000       2                  2C          3S         0
3P         White Lamp                         800        3                  3C          5S         0
4P         Antique Silver Earrings            400        7                  4C          2S         0
5P         Antique Silver Bracelet            700        5                  4C          6S         0
6P         Catwalk leather flats              1599       3                  2C          4S         0
7P         Introduction to Java               650        8                  1C          5S         0
8P         Portico King size bedsheet         1999       1                  3C          1S         0
9P         Book rack                          999        7                  3C          4S         0
10P        Artificial Intelligence 3rd Editio 570        9                  1C          2S         0
11P        Introduction to python             630        10                 1C          5S         0
```

(C) Filter by **Product Rating**

```
filter_criteria(3, 3);
```

```
P_ID | PRODUCT | AMOUNT | QUANTITY | CAT_ID | SELLER_ID | RATING
1P The Programming language of ORACLE 350 4 1C 1S 4.5
3P White Lamp 800 3 3C 5S 4
4P Antique Silver Earrings 400 7 4C 2S 3
7P Introduction to Java 650 8 1C 5S 3
8P Portico King size bedsheet 1999 1 3C 1S 5
```

(D) Filter by **Seller Rating**

```
    filter_criteria(4, 4);
```

```
P_ID | PRODUCT | AMOUNT | QUANTITY | CAT_ID | SELLER_ID | RATING
1P The Programming language of ORACLE 350 4 1C 1S 4.5
8P Portico King size bedsheet 1999 1 3C 1S 5

Statement processed.
```

(E) Filter by Stock Available

```
-- STOCK AVAILABLE (2 nd Paramater Does Not Matter)
    filter_criteria(5, 3);
```

```
P_ID | PRODUCT | AMOUNT | QUANTITY | CAT_ID | SELLER_ID | RATING
10P Artificial Intelligence 3rd Edition 570 9 1C 2S
11P Introduction to python 630 10 1C 5S 1.5
1P The Programming language of ORACLE 350 4 1C 1S 4.5
2P Nike White shoes 7000 2 2C 3S
3P White Lamp 800 3 3C 5S 4
4P Antique Silver Earrings 400 7 4C 2S 3
5P Antique Silver Bracelet 700 5 4C 6S
6P Catwalk leather flats 1599 3 2C 4S 1
7P Introduction to Java 650 8 1C 5S 3
8P Portico King size bedsheet 1999 1 3C 1S 5
9P Book rack 999 7 3C 4S 2.5

Statement processed.
```

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:

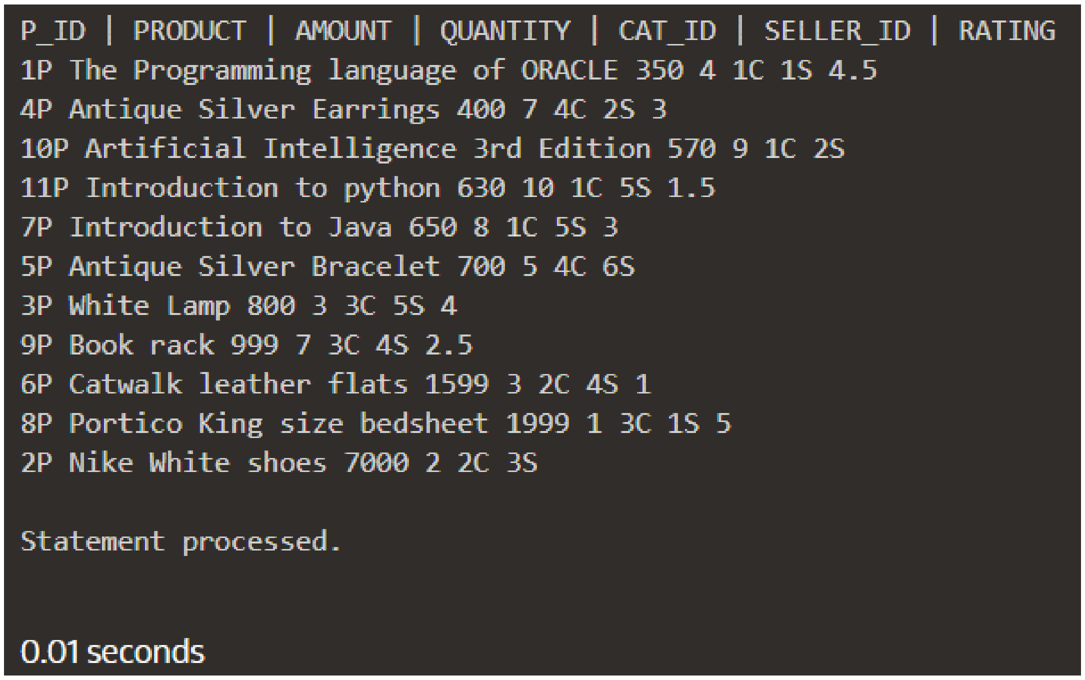
```
CREATE OR REPLACE FUNCTION sort_criteria(opt IN number) RETURN SYS_REFCURSOR IS prod_details
SYS_REFCURSOR;

    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' || ' | ' || 'QUAN
TITY' || ' | ' || '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, pr
od_sellerid, prod_rating;
    EXIT WHEN prod_details%NOTFOUND;
    dbms_output.put_line( prod_prodid || ' ' || prod_name || ' ' || prod_amt || ' ' || prod_q
uant || ' ' || prod_catid || ' ' || prod_sellerid || ' ' || prod_rating);
    END LOOP;

END;
```

(A) Sort by Price [Lowest to Highest]

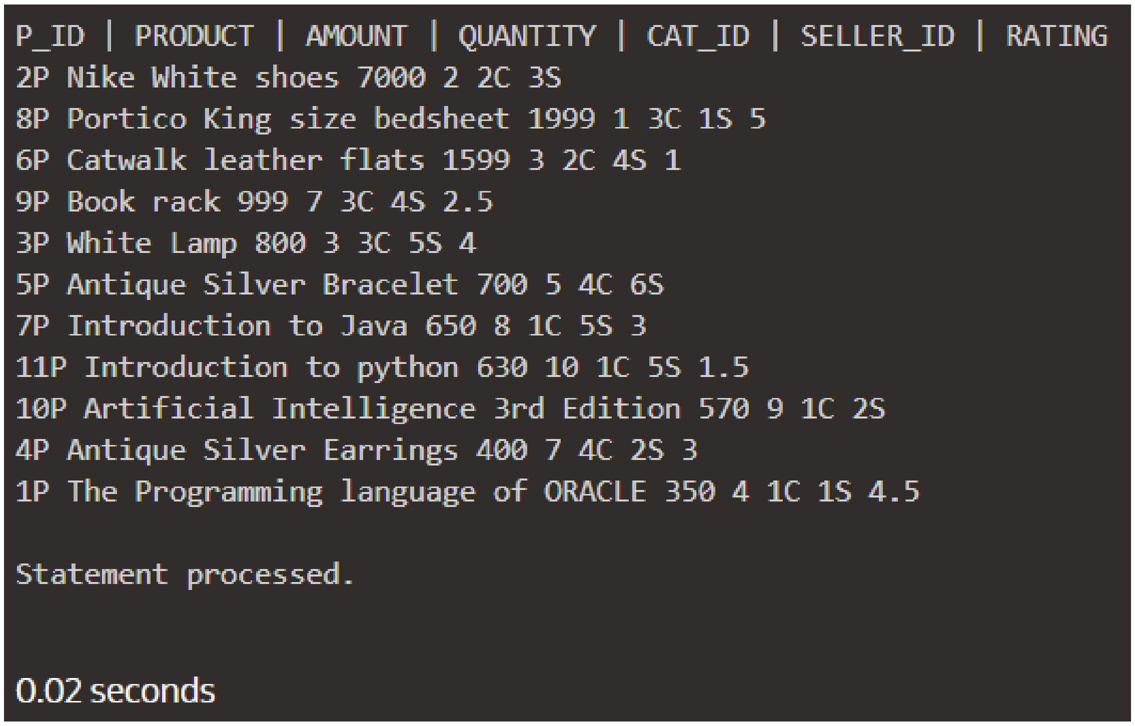
```
    prod_details := sort_criteria(1);
```



```
P_ID | PRODUCT | AMOUNT | QUANTITY | CAT_ID | SELLER_ID | RATING
1P The Programming language of ORACLE 350 4 1C 1S 4.5
4P Antique Silver Earrings 400 7 4C 2S 3
10P Artificial Intelligence 3rd Edition 570 9 1C 2S
11P Introduction to python 630 10 1C 5S 1.5
7P Introduction to Java 650 8 1C 5S 3
5P Antique Silver Bracelet 700 5 4C 6S
3P White Lamp 800 3 3C 5S 4
9P Book rack 999 7 3C 4S 2.5
6P Catwalk leather flats 1599 3 2C 4S 1
8P Portico King size bedsheet 1999 1 3C 1S 5
2P Nike White shoes 7000 2 2C 3S

Statement processed.


0.01 seconds
```

(B) Sort by Price [Highest to Lowest]

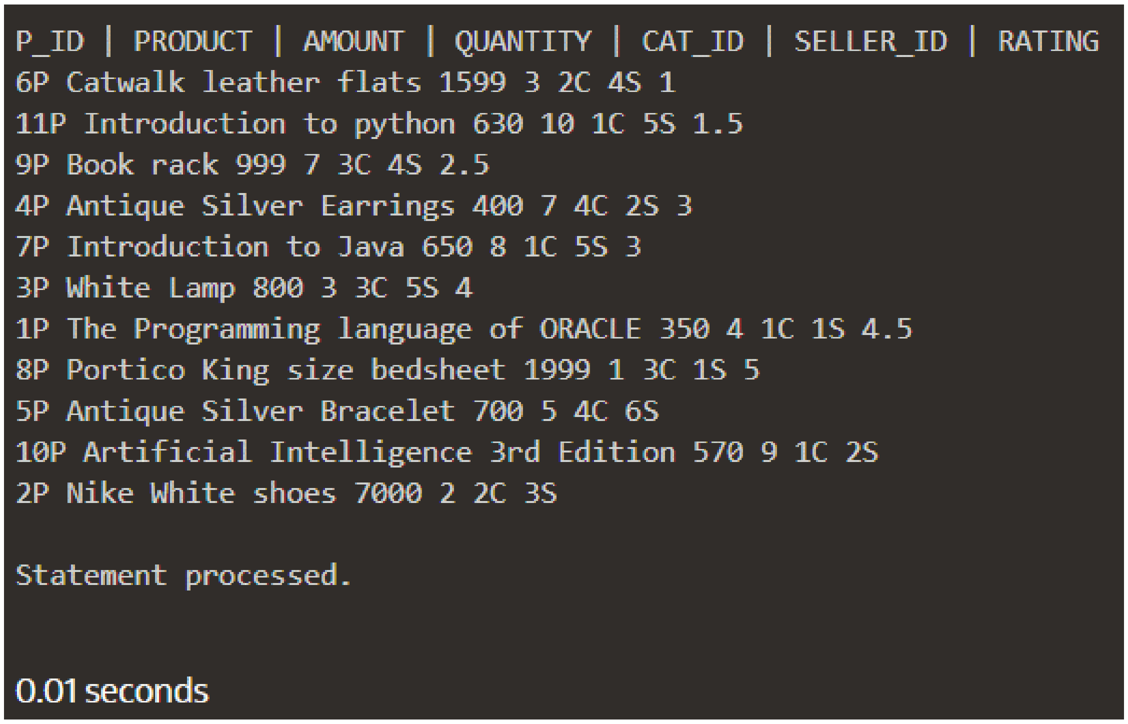
```
    prod_details := sort_criteria(2);
```



```
P_ID | PRODUCT | AMOUNT | QUANTITY | CAT_ID | SELLER_ID | RATING
2P Nike White shoes 7000 2 2C 3S
8P Portico King size bedsheet 1999 1 3C 1S 5
6P Catwalk leather flats 1599 3 2C 4S 1
9P Book rack 999 7 3C 4S 2.5
3P White Lamp 800 3 3C 5S 4
5P Antique Silver Bracelet 700 5 4C 6S
7P Introduction to Java 650 8 1C 5S 3
11P Introduction to python 630 10 1C 5S 1.5
10P Artificial Intelligence 3rd Edition 570 9 1C 2S
4P Antique Silver Earrings 400 7 4C 2S 3
1P The Programming language of ORACLE 350 4 1C 1S 4.5

Statement processed.


0.02 seconds
```

(C) Sort by Rating [Lowest to Highest]

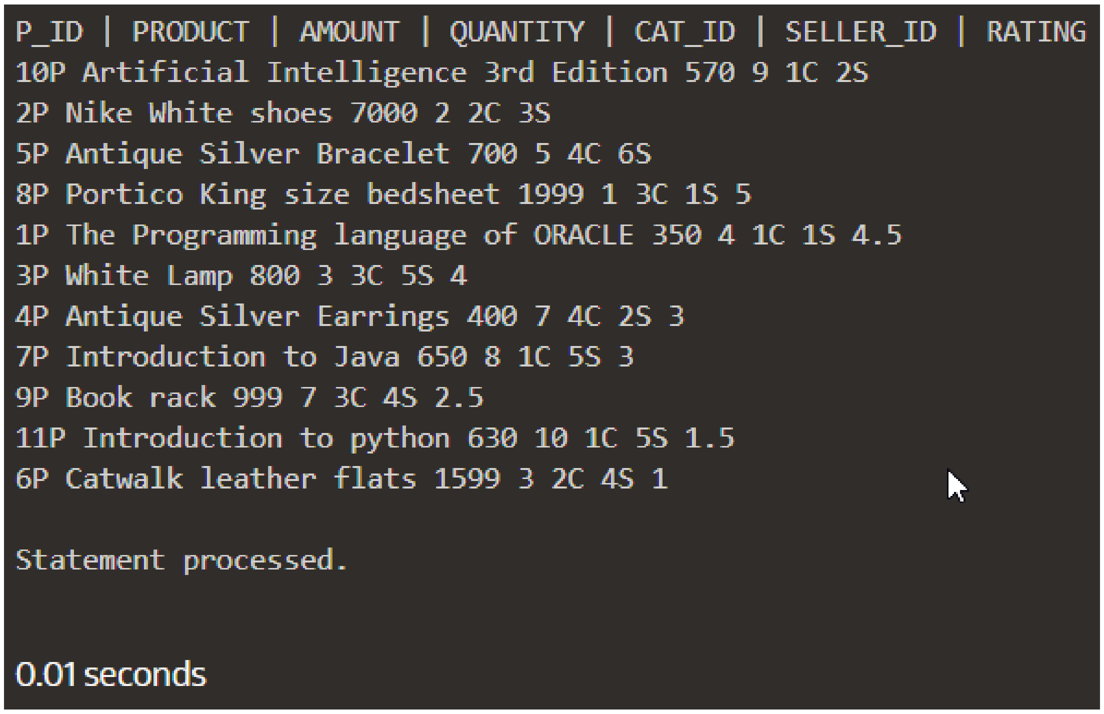
```
    prod_details := sort_criteria(3);
```



```
P_ID | PRODUCT | AMOUNT | QUANTITY | CAT_ID | SELLER_ID | RATING
6P Catwalk leather flats 1599 3 2C 4S 1
11P Introduction to python 630 10 1C 5S 1.5
9P Book rack 999 7 3C 4S 2.5
4P Antique Silver Earrings 400 7 4C 2S 3
7P Introduction to Java 650 8 1C 5S 3
3P White Lamp 800 3 3C 5S 4
1P The Programming language of ORACLE 350 4 1C 1S 4.5
8P Portico King size bedsheet 1999 1 3C 1S 5
5P Antique Silver Bracelet 700 5 4C 6S
10P Artificial Intelligence 3rd Edition 570 9 1C 2S
2P Nike White shoes 7000 2 2C 3S

Statement processed.


0.01 seconds
```

(D) Sort by Rating [Highest to Lowest]

```
    prod_details := sort_criteria(4);
```



```
P_ID | PRODUCT | AMOUNT | QUANTITY | CAT_ID | SELLER_ID | RATING
10P Artificial Intelligence 3rd Edition 570 9 1C 2S
2P Nike White shoes 7000 2 2C 3S
5P Antique Silver Bracelet 700 5 4C 6S
8P Portico King size bedsheet 1999 1 3C 1S 5
1P The Programming language of ORACLE 350 4 1C 1S 4.5
3P White Lamp 800 3 3C 5S 4
4P Antique Silver Earrings 400 7 4C 2S 3
7P Introduction to Java 650 8 1C 5S 3
9P Book rack 999 7 3C 4S 2.5
11P Introduction to python 630 10 1C 5S 1.5
6P Catwalk leather flats 1599 3 2C 4S 1

Statement processed.


0.01 seconds
```

Submitted By:

BHAGYA VINOD RANA

U19CS012