W2025 CIS115-01 Intro to Databases and SQL

ASSIGNMENT 1

Topic: Queries from single Table

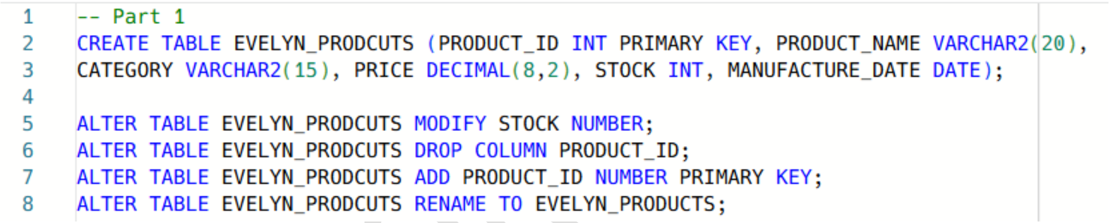
Name: EVELYN LUO
Date: MARCH 2, 2025

Instructions - Run the queries and paste the snapshots of queries and its output under each question. Choose your own data for filling out columns of the table to avoid plagiarism. The table name should be preceded by your firstname_

1. Create a table *yourname_products* with the following fields:
    a. *product_id*: number (primary key)
    b. *product_name*: varchar(20)
    c. *category*: varchar(15)
    d. *price*: decimal(8, 2)
    e. *stock*: number
    f. *manufacture_date*: date

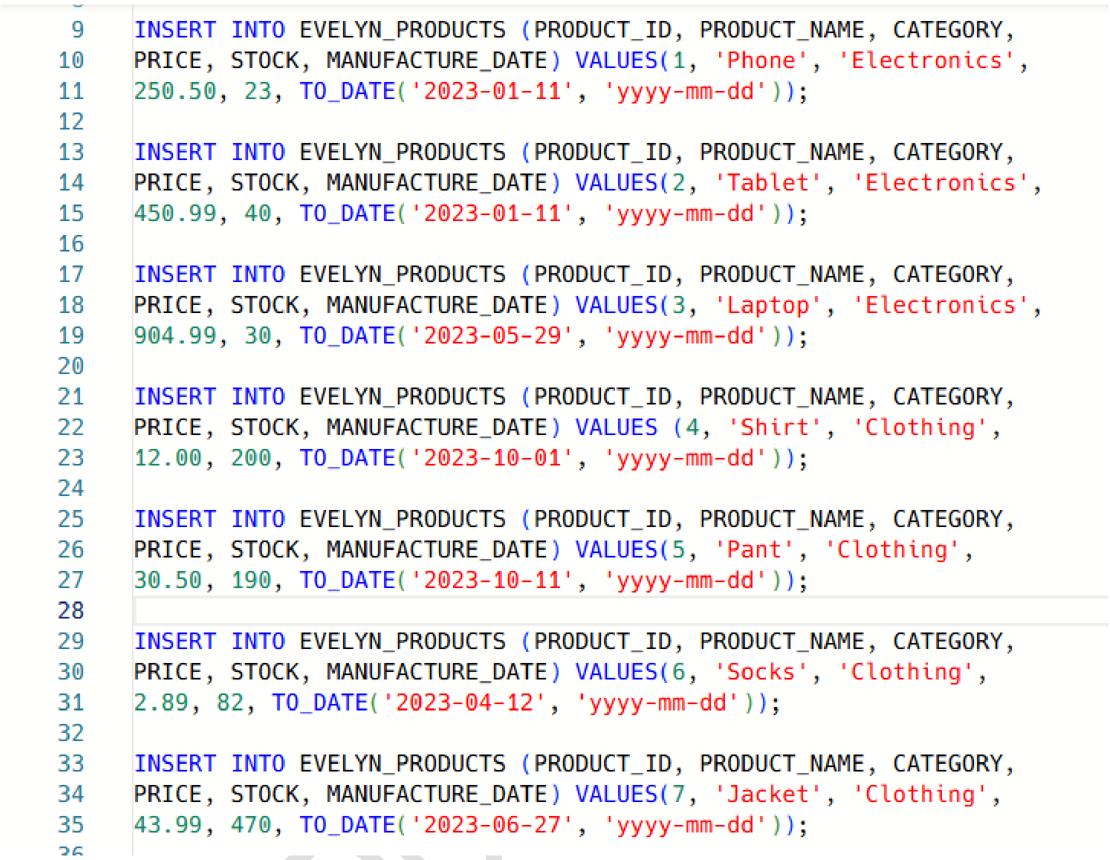
    using the following command:

    CREATE TABLE yourname_products (product_id NUMBER PRIMARY KEY, product_name VARCHAR2(20), category VARCHAR2(15), price DECIMAL (8, 2), stock NUMBER, manufacture_date DATE);

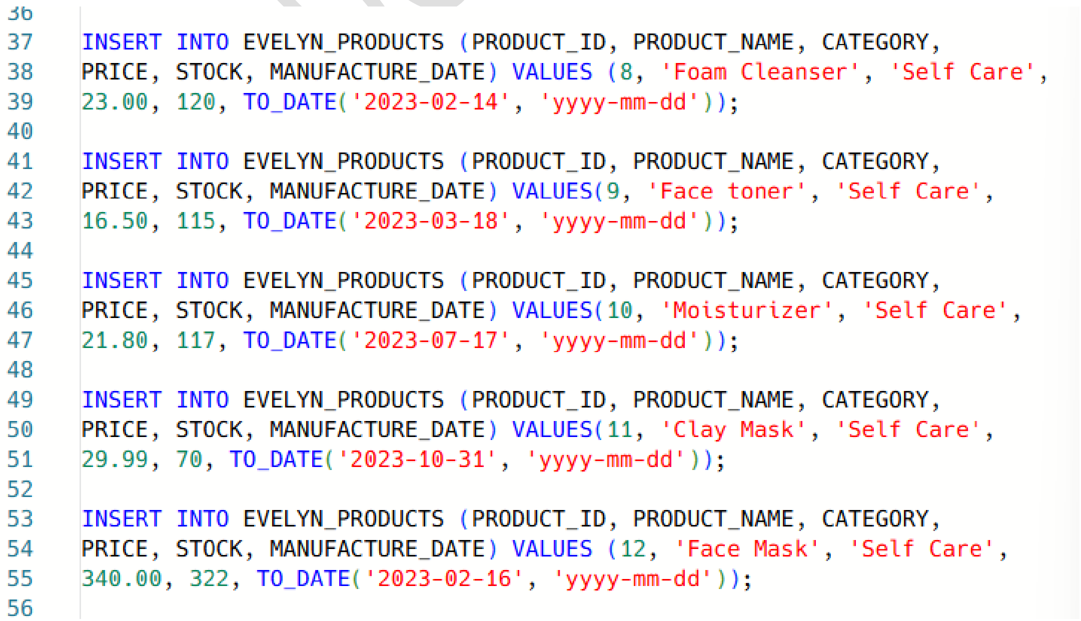


```
-- Part 1
CREATE TABLE EVELYN_PRODCUTS (PRODUCT_ID INT PRIMARY KEY, PRODUCT_NAME VARCHAR2(20),
CATEGORY VARCHAR2(15), PRICE DECIMAL(8,2), STOCK INT, MANUFACTURE_DATE DATE);

ALTER TABLE EVELYN_PRODCUTS MODIFY STOCK NUMBER;
ALTER TABLE EVELYN_PRODCUTS DROP COLUMN PRODUCT_ID;
ALTER TABLE EVELYN_PRODCUTS ADD PRODUCT_ID NUMBER PRIMARY KEY;
ALTER TABLE EVELYN_PRODCUTS RENAME TO EVELYN_PRODUCTS;
```

2. Insert at least 12 rows of data into the table, ensuring:
   a. Prices range between 100 and 1000.
   b. Stock values are unique.
   c. Use any categories and manufacture dates between January 2023 and December 2023.

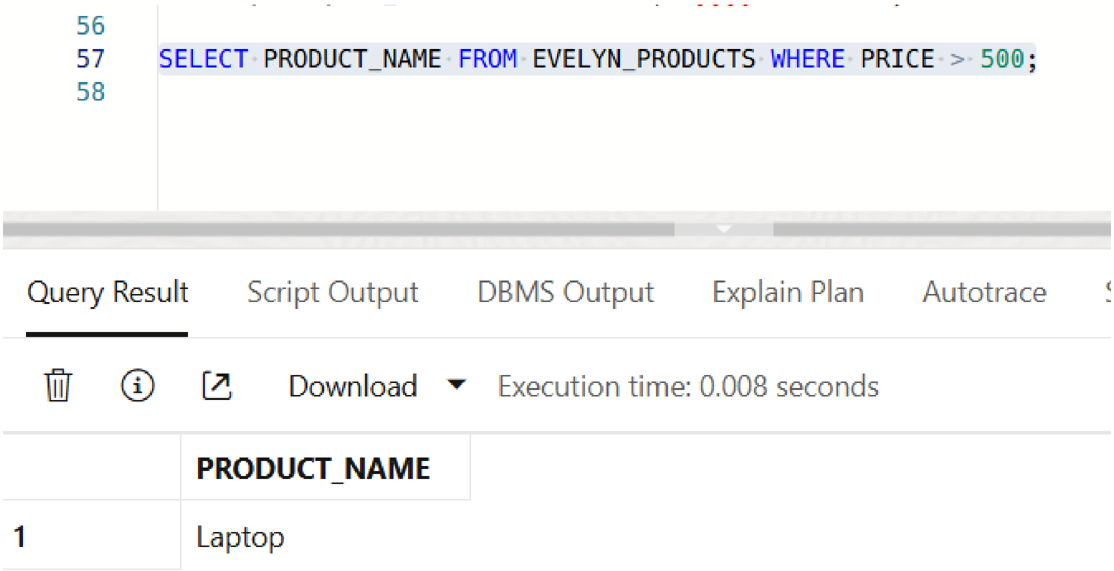


```
INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES(1, 'Phone', 'Electronics',
250.50, 23, TO_DATE('2023-01-11', 'yyyy-mm-dd'));

INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES(2, 'Tablet', 'Electronics',
450.99, 40, TO_DATE('2023-01-11', 'yyyy-mm-dd'));

INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES(3, 'Laptop', 'Electronics',
904.99, 30, TO_DATE('2023-05-29', 'yyyy-mm-dd'));

INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES (4, 'Shirt', 'Clothing',
12.00, 200, TO_DATE('2023-10-01', 'yyyy-mm-dd'));

INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES(5, 'Pant', 'Clothing',
30.50, 190, TO_DATE('2023-10-11', 'yyyy-mm-dd'));

INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES(6, 'Socks', 'Clothing',
2.89, 82, TO_DATE('2023-04-12', 'yyyy-mm-dd'));

INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES(7, 'Jacket', 'Clothing',
43.99, 470, TO_DATE('2023-06-27', 'yyyy-mm-dd'));
```



```
INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES (8, 'Foam Cleanser', 'Self Care',
23.00, 120, TO_DATE('2023-02-14', 'yyyy-mm-dd'));

INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES(9, 'Face toner', 'Self Care',
16.50, 115, TO_DATE('2023-03-18', 'yyyy-mm-dd'));

INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES(10, 'Moisturizer', 'Self Care',
21.80, 117, TO_DATE('2023-07-17', 'yyyy-mm-dd'));

INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES(11, 'Clay Mask', 'Self Care',
29.99, 70, TO_DATE('2023-10-31', 'yyyy-mm-dd'));

INSERT INTO EVELYN_PRODUCTS (PRODUCT_ID, PRODUCT_NAME, CATEGORY,
PRICE, STOCK, MANUFACTURE_DATE) VALUES (12, 'Face Mask', 'Self Care',
340.00, 322, TO_DATE('2023-02-16', 'yyyy-mm-dd'));
```

3. Display the *product_name* and *price* of all products where the price is greater than 500.

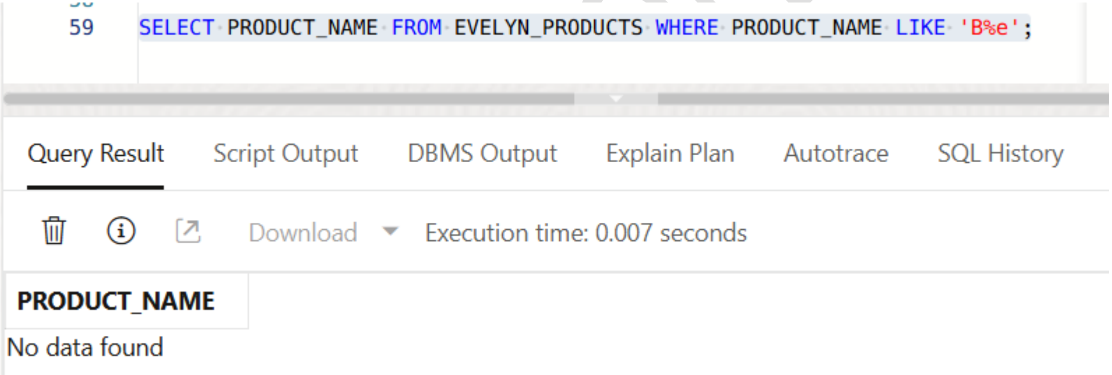


```
SELECT PRODUCT_NAME FROM EVELYN_PRODUCTS WHERE PRICE > 500;
```

Execution time: 0.008 seconds

	PRODUCT_NAME
1	Laptop

4. List the products whose *product_name* starts with "B" and ends with "e".

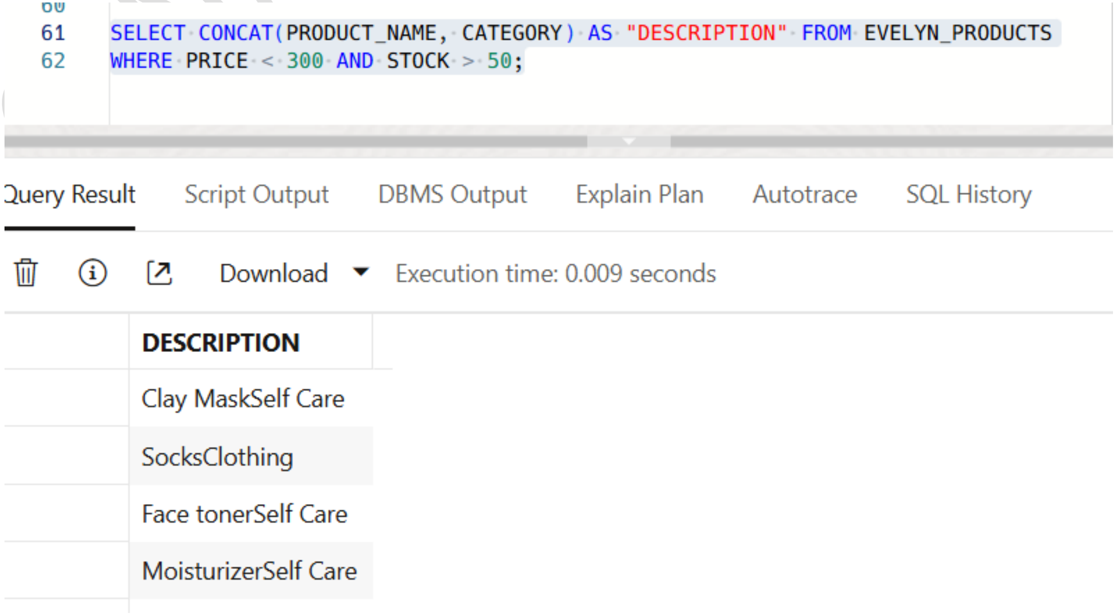


```
SELECT PRODUCT_NAME FROM EVELYN_PRODUCTS WHERE PRODUCT_NAME LIKE 'B%e';
```

Execution time: 0.007 seconds

PRODUCT_NAME
No data found

5. Concatenate *product_name* and *category* into a new column named "Description" for products with prices below 300 and stock greater than 50.

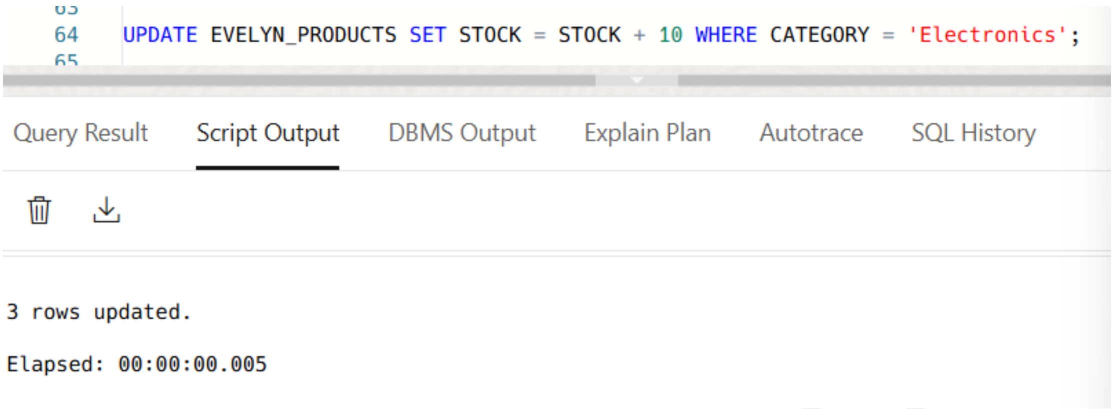


```
SELECT CONCAT(PRODUCT_NAME, CATEGORY) AS "DESCRIPTION" FROM EVELYN_PRODUCTS
WHERE PRICE < 300 AND STOCK > 50;
```

Execution time: 0.009 seconds

	DESCRIPTION
	Clay MaskSelf Care
	SocksClothing
	Face tonerSelf Care
	MoisturizerSelf Care

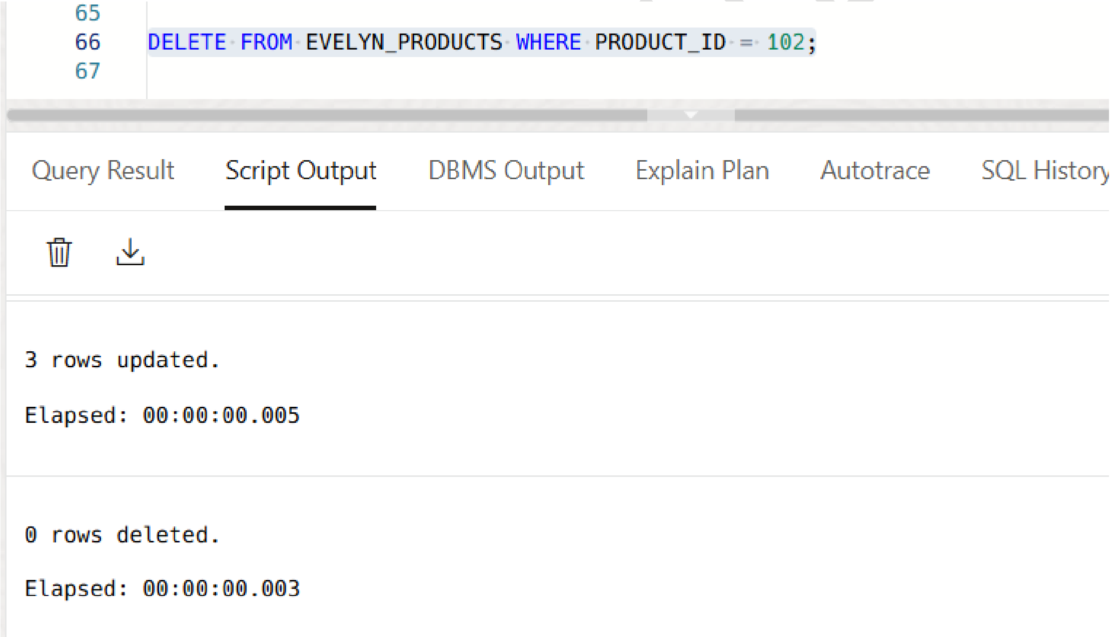
6. Increase the stock of all products by 10 if the product belongs to the "Electronics" category.



```
UPDATE EVELYN_PRODUCTS SET STOCK = STOCK + 10 WHERE CATEGORY = 'Electronics';
```

```
3 rows updated.

Elapsed: 00:00:00.005
```

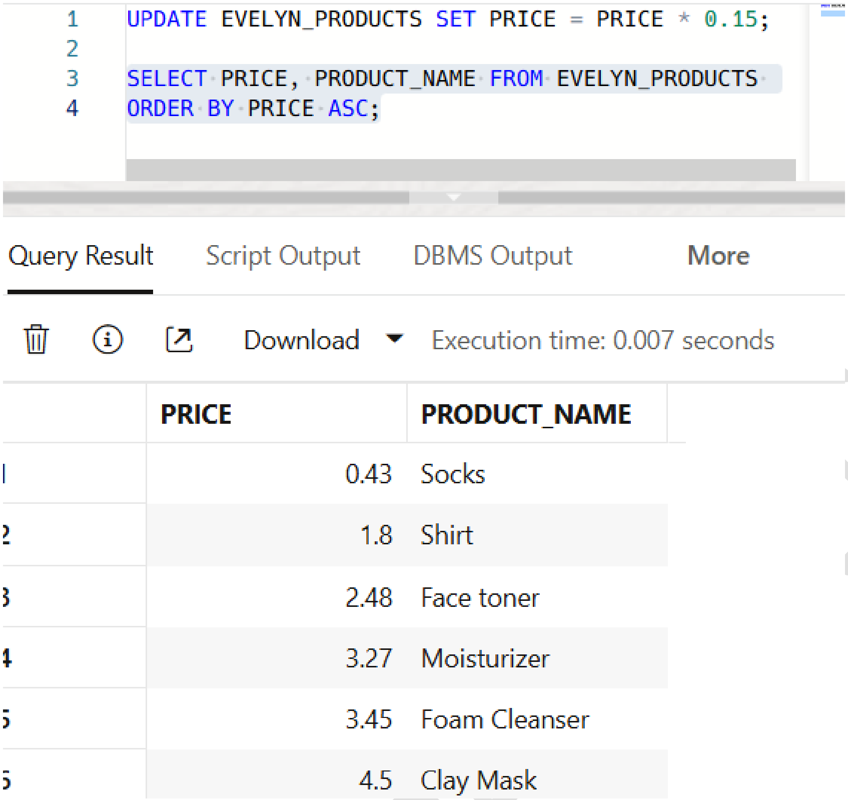
7. Delete the record of the product with *product_id* = 102.



```
DELETE FROM EVELYN_PRODUCTS WHERE PRODUCT_ID = 102;
```

```
3 rows updated.

Elapsed: 00:00:00.005

0 rows deleted.

Elapsed: 00:00:00.003
```

8. Simulate a 15% discount on all products and display the updated prices without making any changes to the table.

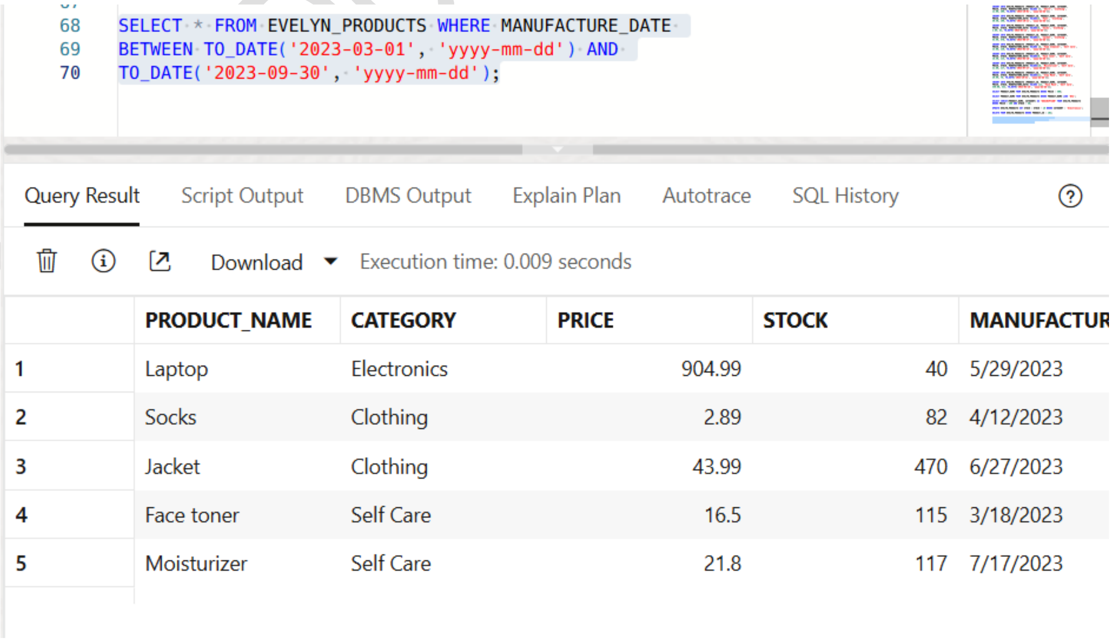


```sql
UPDATE EVELYN_PRODUCTS SET PRICE = PRICE * 0.15;

SELECT PRICE, PRODUCT_NAME FROM EVELYN_PRODUCTS
ORDER BY PRICE ASC;
```

Execution time: 0.007 seconds

	PRICE	PRODUCT_NAME
1	0.43	Socks
2	1.8	Shirt
3	2.48	Face toner
4	3.27	Moisturizer
5	3.45	Foam Cleanser
6	4.5	Clay Mask

9. List all products manufactured between March 2023 and September 2023.

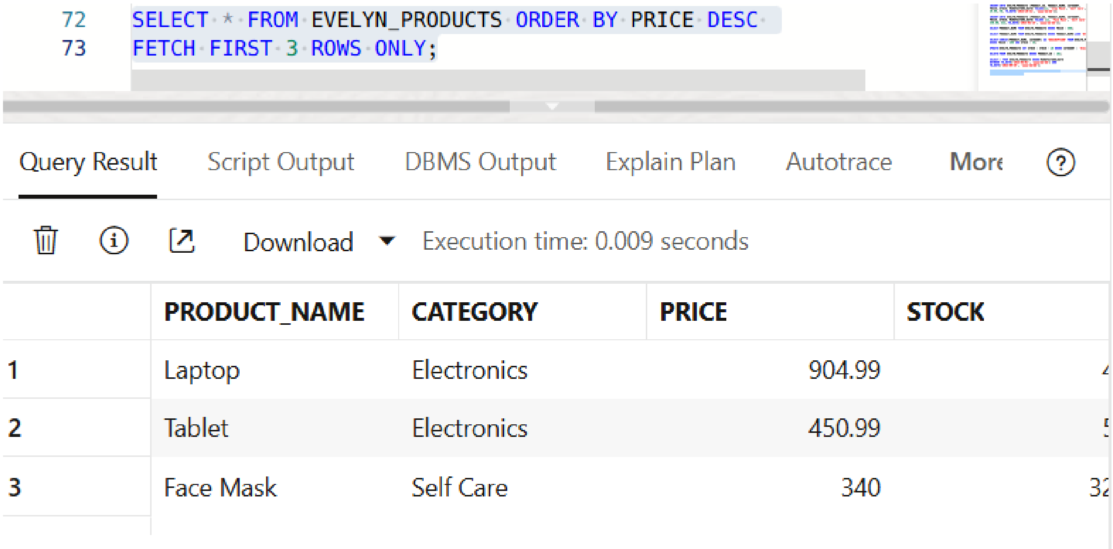


```sql
SELECT * FROM EVELYN_PRODUCTS WHERE MANUFACTURE_DATE
BETWEEN TO_DATE('2023-03-01', 'yyyy-mm-dd') AND
TO_DATE('2023-09-30', 'yyyy-mm-dd');
```

Execution time: 0.009 seconds

	PRODUCT_NAME	CATEGORY	PRICE	STOCK	MANUFACTUR
1	Laptop	Electronics	904.99	40	5/29/2023
2	Socks	Clothing	2.89	82	4/12/2023
3	Jacket	Clothing	43.99	470	6/27/2023
4	Face toner	Self Care	16.5	115	3/18/2023
5	Moisturizer	Self Care	21.8	117	7/17/2023

10. Display the top 3 most expensive products.

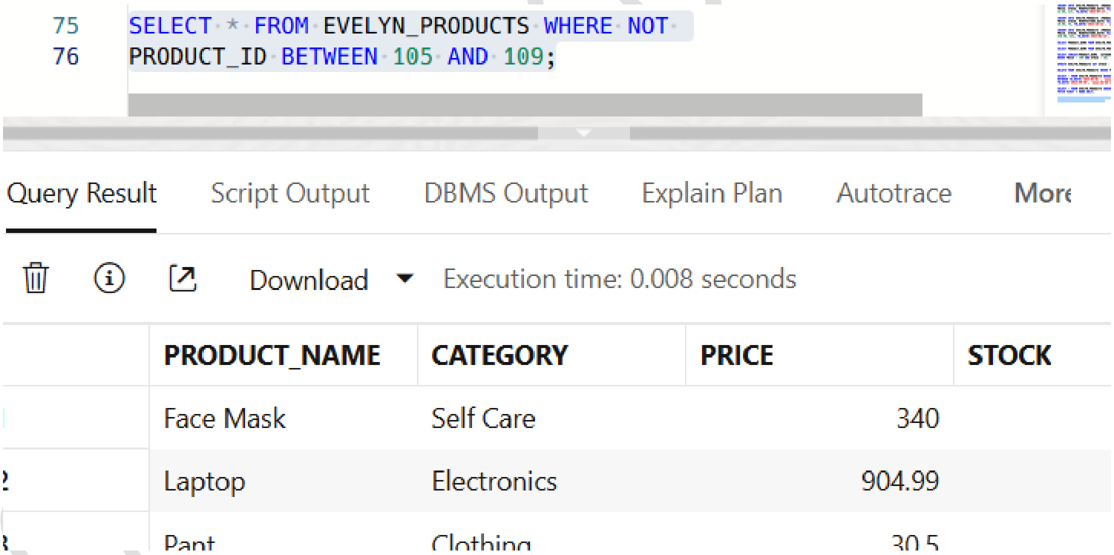


```
SELECT * FROM EVELYN_PRODUCTS ORDER BY PRICE DESC
FETCH FIRST 3 ROWS ONLY;
```

Execution time: 0.009 seconds

	PRODUCT_NAME	CATEGORY	PRICE	STOCK
1	Laptop	Electronics	904.99	4
2	Tablet	Electronics	450.99	5
3	Face Mask	Self Care	340	32

11. List products whose *product_id* does not fall between 105 and 109.

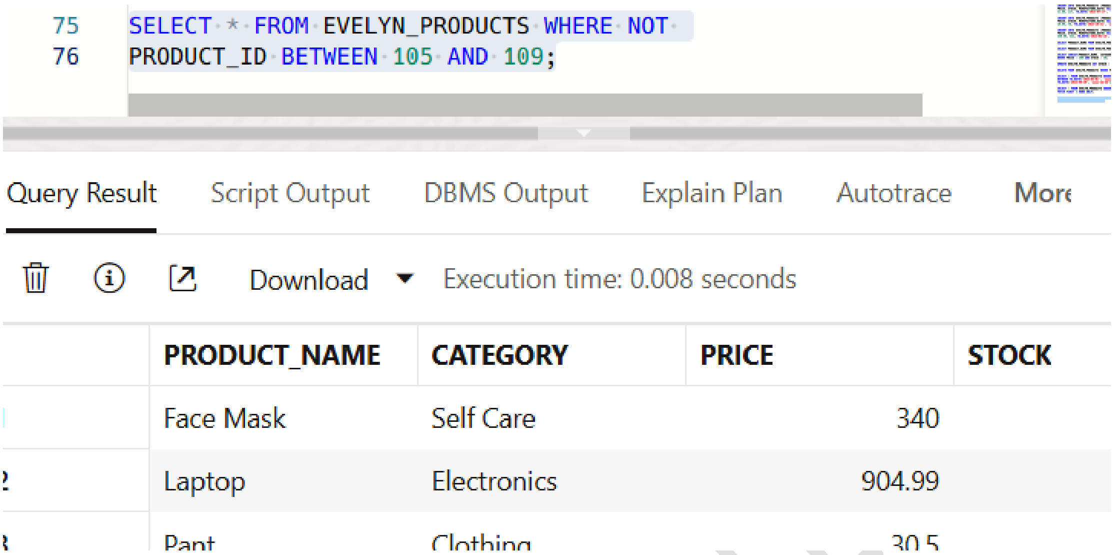


```
SELECT * FROM EVELYN_PRODUCTS WHERE NOT
PRODUCT_ID BETWEEN 105 AND 109;
```

Execution time: 0.008 seconds

	PRODUCT_NAME	CATEGORY	PRICE	STOCK
1	Face Mask	Self Care	340	
2	Laptop	Electronics	904.99	
3	Pant	Clothing	30.5	

12. Find and list all products in the "Home Appliance" category with prices below 300.

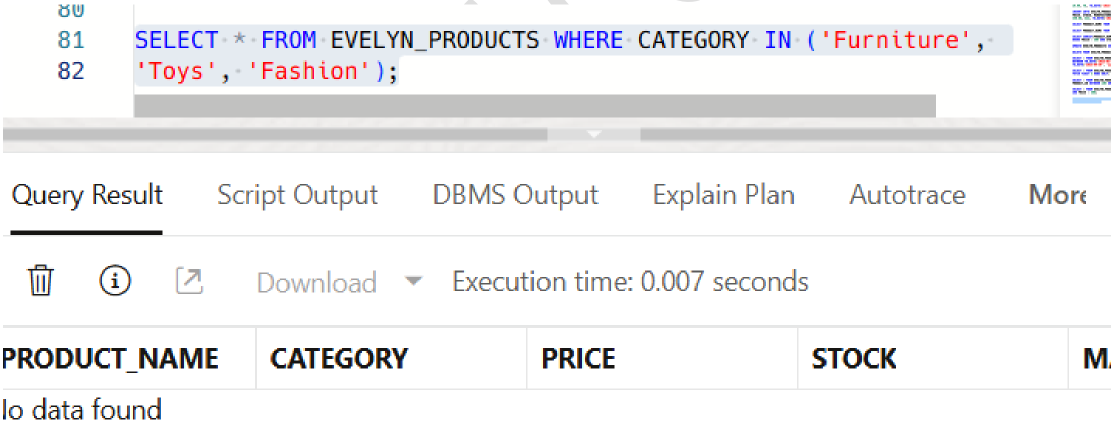


```
SELECT * FROM EVELYN_PRODUCTS WHERE NOT
PRODUCT_ID BETWEEN 105 AND 109;
```

Execution time: 0.008 seconds

	PRODUCT_NAME	CATEGORY	PRICE	STOCK
	Face Mask	Self Care	340	
2	Laptop	Electronics	904.99	
3	Pant	Clothing	30.5	

13. List the products available in the following categories: "Furniture", "Toys", "Fashion".



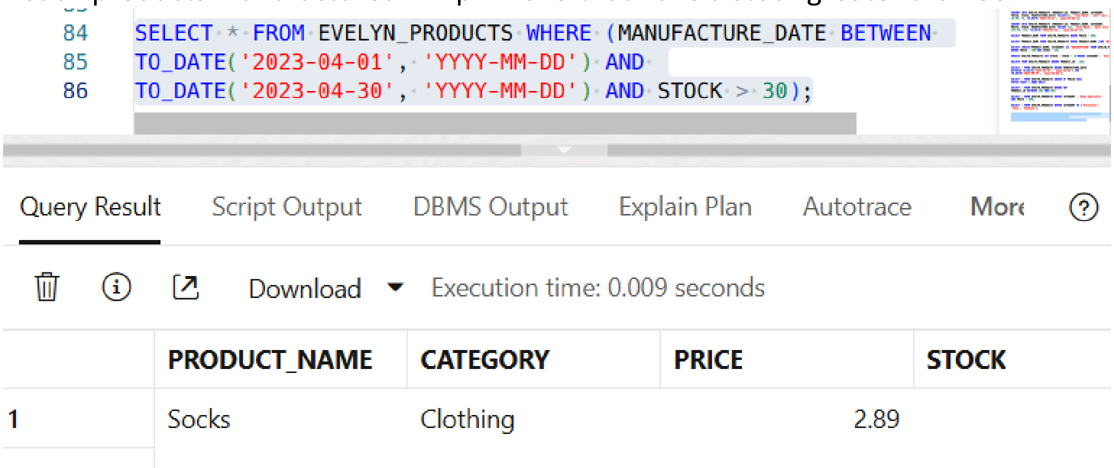
```
SELECT * FROM EVELYN_PRODUCTS WHERE CATEGORY IN ('Furniture',
'Toys', 'Fashion');
```

Execution time: 0.007 seconds

PRODUCT_NAME	CATEGORY	PRICE	STOCK	M

lo data found

14. List all products manufactured in April 2023 that have a stock greater than 30.

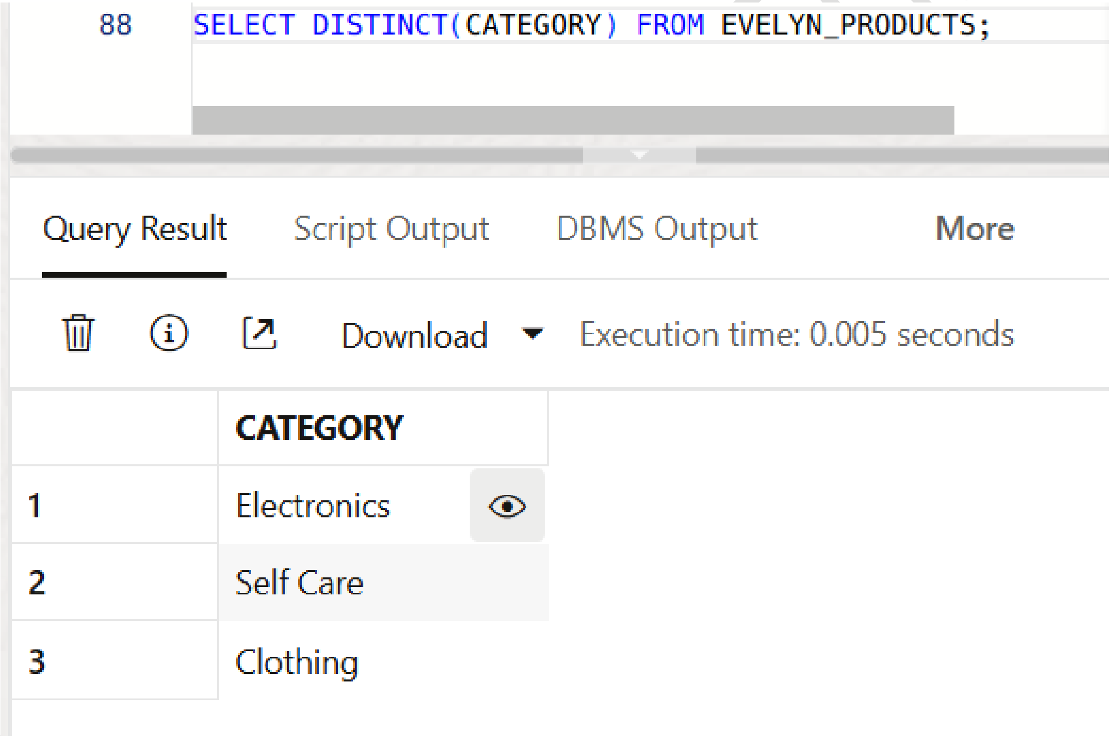


```
SELECT * FROM EVELYN_PRODUCTS WHERE (MANUFACTURE_DATE BETWEEN
TO_DATE('2023-04-01', 'YYYY-MM-DD') AND
TO_DATE('2023-04-30', 'YYYY-MM-DD') AND STOCK > 30);
```

Execution time: 0.009 seconds

	PRODUCT_NAME	CATEGORY	PRICE	STOCK
1	Socks	Clothing	2.89	

15. Display all unique categories available in the table

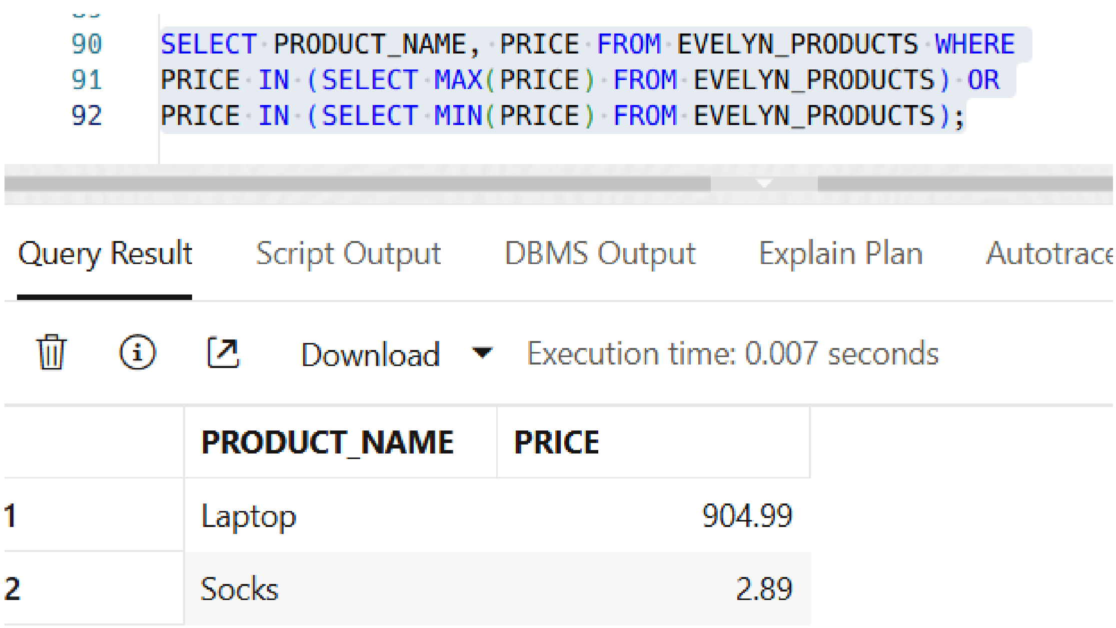


```
SELECT DISTINCT(CATEGORY) FROM EVELYN_PRODUCTS;
```

Execution time: 0.005 seconds

	CATEGORY
1	Electronics
2	Self Care
3	Clothing

16. Identify and display the *product_name* and *price* of the most expensive and least expensive products.



```
SELECT PRODUCT_NAME, PRICE FROM EVELYN_PRODUCTS WHERE
PRICE IN (SELECT MAX(PRICE) FROM EVELYN_PRODUCTS) OR
PRICE IN (SELECT MIN(PRICE) FROM EVELYN_PRODUCTS);
```

Execution time: 0.007 seconds

	PRODUCT_NAME	PRICE
1	Laptop	904.99
2	Socks	2.89