Assignment Report

Database Assignment 1

Prepared By:

Devraj Neupane

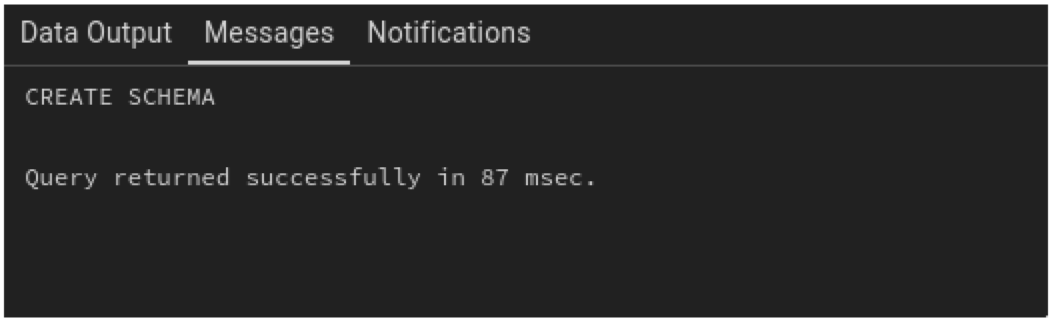
Roll No: 7

Group: D

Create assignment1 schema

```
CREATE SCHEMA IF NOT EXISTS assignment1 AUTHORIZATION postgres;
```

Screenshot:

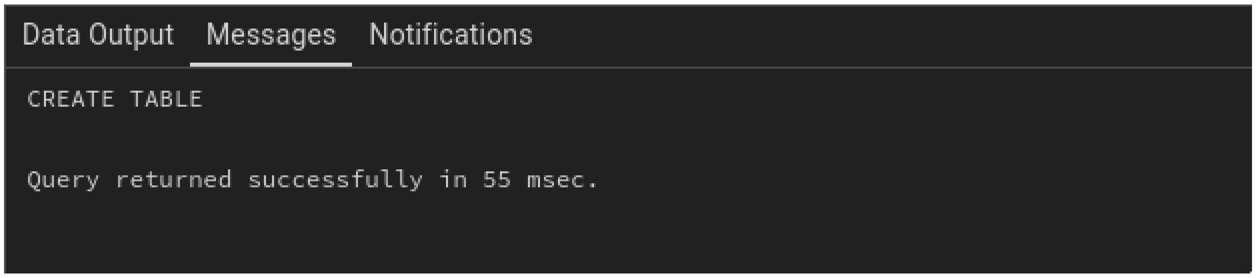


```
CREATE SCHEMA

Query returned successfully in 87 msec.
```

Create products table

```
CREATE TABLE IF NOT EXISTS assignment1.products (
  product_id SERIAL PRIMARY KEY,
  product_name VARCHAR(100) NOT NULL,
  category VARCHAR(50) NOT NULL,
  price INT NOT NULL
);
```

Screenshot:

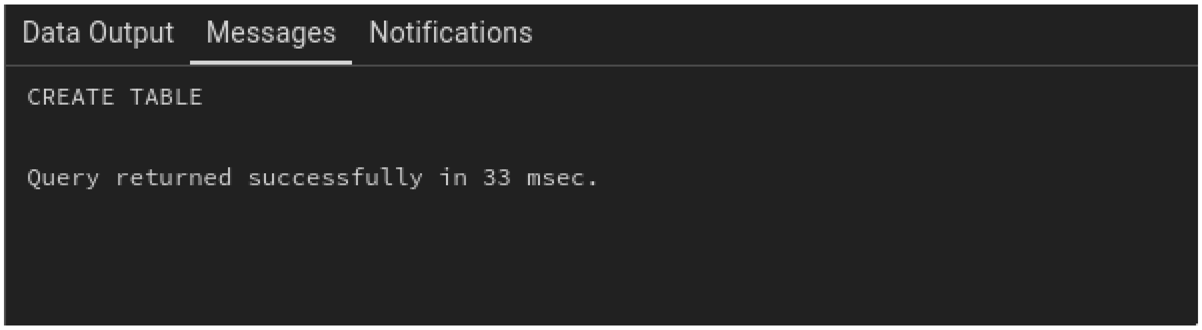


```
CREATE TABLE

Query returned successfully in 55 msec.
```

Create orders table

```
CREATE TABLE IF NOT EXISTS assignment1.orders (
  order_id SERIAL PRIMARY KEY,
  customer_name VARCHAR(100) NOT NULL,
  product_id INT REFERENCES assignment1.products (product_id),
  quantity INT DEFAULT 1,
  order_date DATE DEFAULT CURRENT_DATE
);
```

Screenshot:



Data Output   Messages   Notifications

```
CREATE TABLE

Query returned successfully in 33 msec.
```

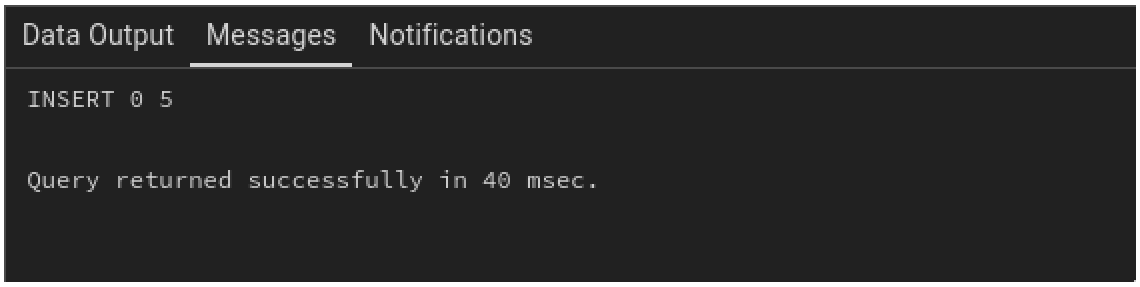
Q1. Perform CRUD

Create(Insert) Operation

Fill products table

```
INSERT INTO
  assignment1.products (product_name, category, price)
VALUES
  (
    'Product A',
    'Category A',
    floor(random () * 1000) + 1
  ),
  (
    'Product B',
    'Category B',
    floor(random () * 1000) + 1
  ),
  (
    'Product C',
    'Category A',
    floor(random () * 1000) + 1
  ),
  (
    'Product D',
    'Category C',
    floor(random () * 1000) + 1
  ),
  (
    'Product E',
    'Category B',
    floor(random () * 1000) + 1
  );
```

Screenshot:



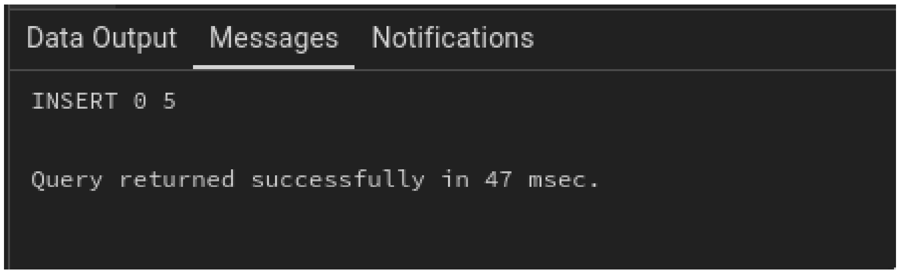
Data Output Messages Notifications

```
INSERT 0 5

Query returned successfully in 40 msec.
```

Fill orders table

```
INSERT INTO
  assignment1.orders (customer_name, product_id, quantity,
order_date)
VALUES
  (
    'Customer A',
    (
      SELECT
        product_id
      FROM
        assignment1.products
      ORDER BY
        random ()
      LIMIT
        1
    ),
    2,
    CURRENT_DATE
  ),
  (
    'Customer B',
    (
      SELECT
        product_id
      FROM
        assignment1.products
      ORDER BY
        random ()
      LIMIT
        1
    ),
    1,
    CURRENT_DATE
  ),
  (
    'Customer C',
    (
      SELECT
        product_id
      FROM
        assignment1.products
      ORDER BY
        random ()
      LIMIT
        1
    ),
    3,
    CURRENT_DATE
  ),
  (
    'Customer D',
    (
      SELECT
        product_id
      FROM
        assignment1.products
      ORDER BY
        random ()
      LIMIT
        1
    ),
    2,
    CURRENT_DATE
  ),
  (
    'Customer E',
    (
      SELECT
        product_id
      FROM
        assignment1.products
      ORDER BY
        random ()
      LIMIT
        1
    ),
    5,
    CURRENT_DATE
  );
```

Screenshot:



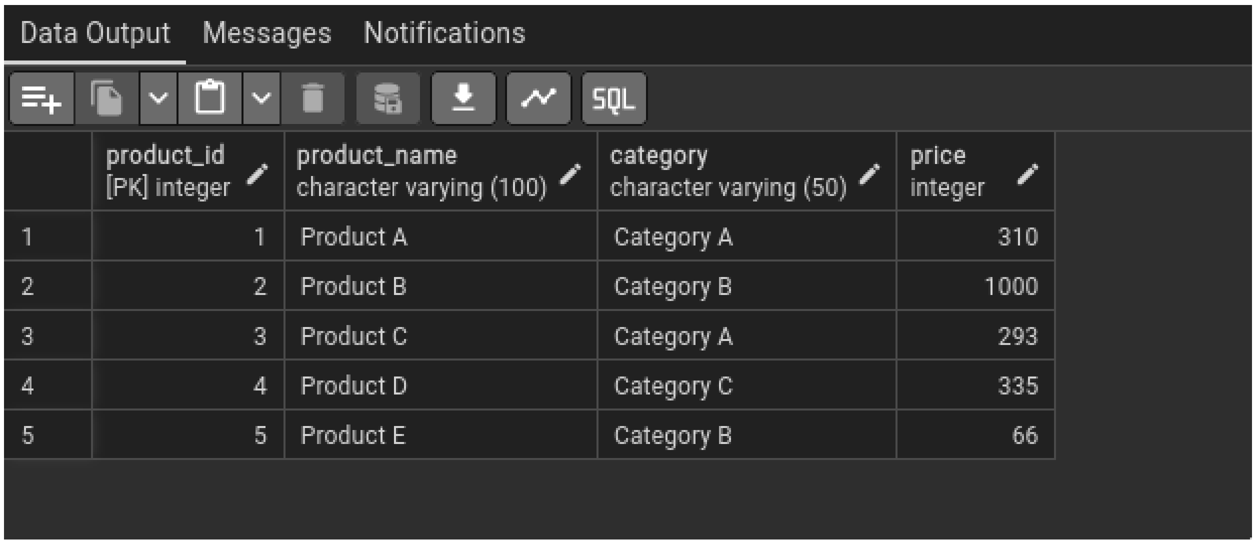
```
INSERT 0 5

Query returned successfully in 47 msec.
```

Read (Select) Operation

Select all products

```
SELECT
  *
FROM
  assignment1.products;
```

Screenshot:

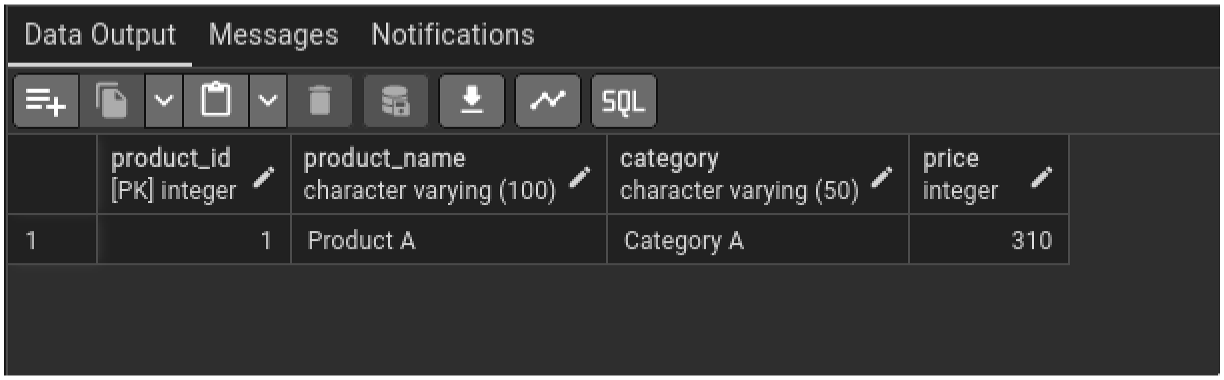


	product_id [PK] integer	product_name character varying (100)	category character varying (50)	price integer
1	1	Product A	Category A	310
2	2	Product B	Category B	1000
3	3	Product C	Category A	293
4	4	Product D	Category C	335
5	5	Product E	Category B	66

Select a specific product by product_id

```
SELECT
  *
FROM
  assignment1.products
WHERE
  product_id = 1;
```

Scrennshot:

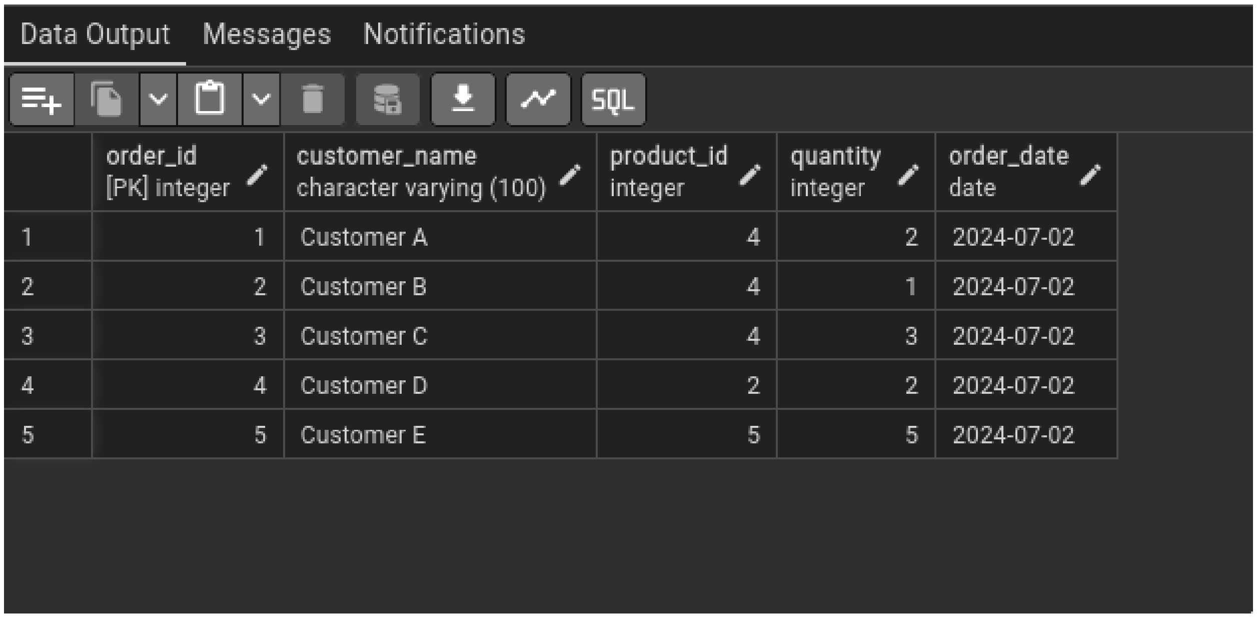


	product_id [PK] integer	product_name character varying (100)	category character varying (50)	price integer
1	1	Product A	Category A	310

Select all orders

```
SELECT
  *
FROM
  assignment1.orders;
```

Screenshot:

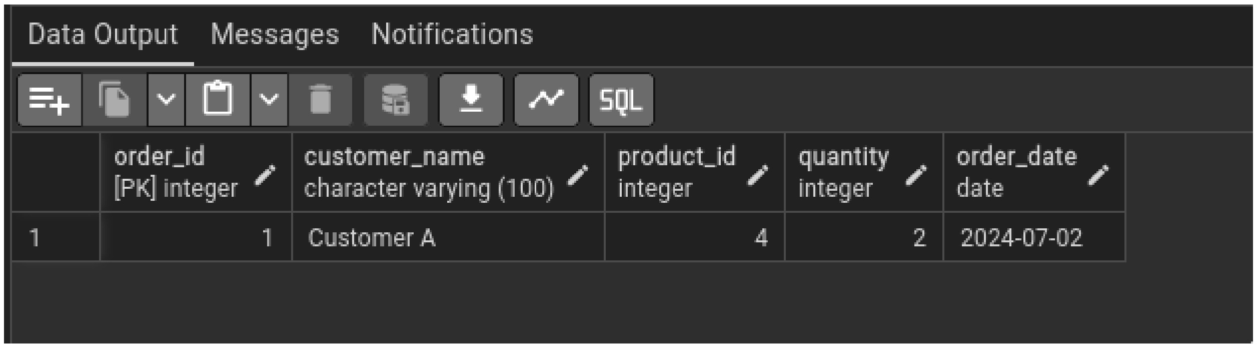


	order_id [PK] integer	customer_name character varying (100)	product_id integer	quantity integer	order_date date
1	1	Customer A	4	2	2024-07-02
2	2	Customer B	4	1	2024-07-02
3	3	Customer C	4	3	2024-07-02
4	4	Customer D	2	2	2024-07-02
5	5	Customer E	5	5	2024-07-02

Select orders for a specific customer

```
SELECT
  *
FROM
  assignment1.orders
WHERE
  customer_name = 'Customer A';
```

Screenshot:

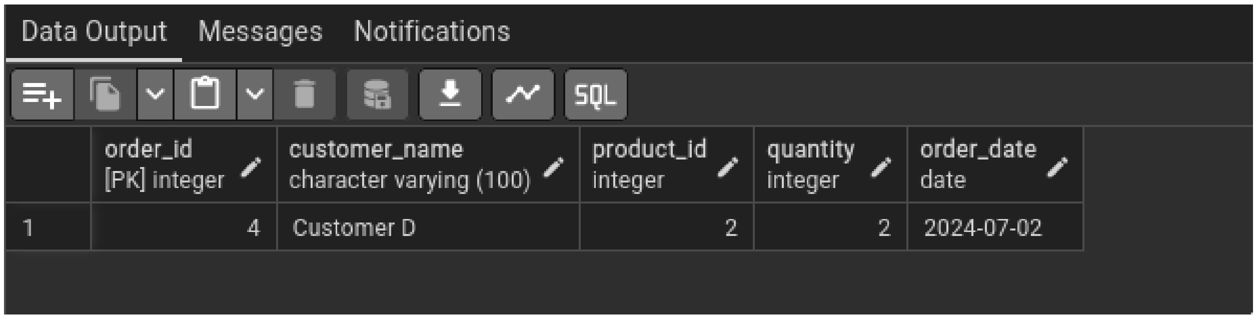


	order_id [PK] integer	customer_name character varying (100)	product_id integer	quantity integer	order_date date
1	1	Customer A	4	2	2024-07-02

Select orders for a specific product

```
SELECT
  *
FROM
  assignment1.orders
WHERE
  product_id = 2;
```

Screenshot:



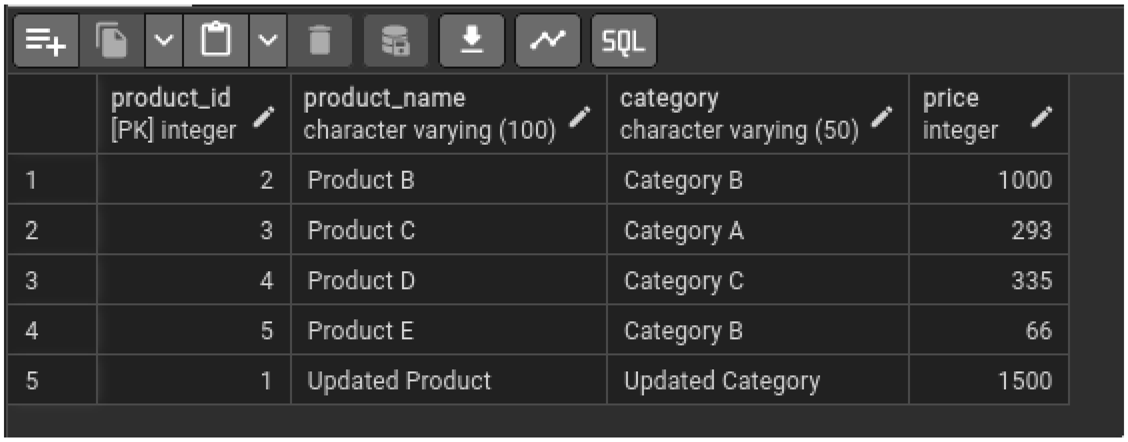
	order_id [PK] integer	customer_name character varying (100)	product_id integer	quantity integer	order_date date
1	4	Customer D	2	2	2024-07-02

Update Operation

Update a product

```
UPDATE assignment1.products
SET
  product_name = 'Updated Product',
  category = 'Updated Category',
  price = 1500
WHERE
  product_id = 1;
```

Screenshot:



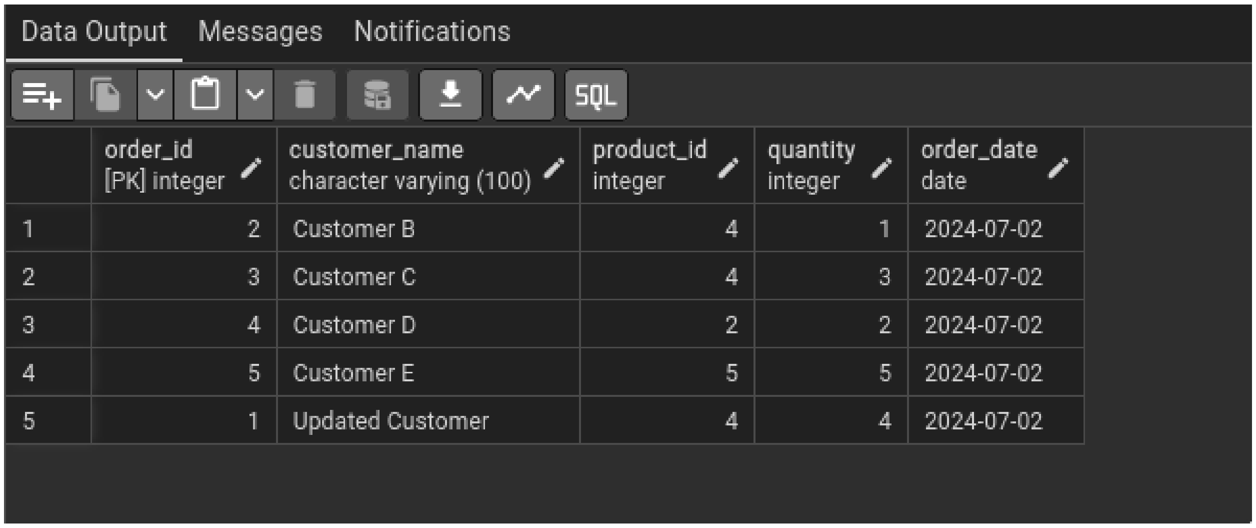
	product_id [PK] integer	product_name character varying (100)	category character varying (50)	price integer
1	2	Product B	Category B	1000
2	3	Product C	Category A	293
3	4	Product D	Category C	335
4	5	Product E	Category B	66
5	1	Updated Product	Updated Category	1500

Update an order

```
UPDATE assignment1.orders
SET
```

```sql
  customer_name = 'Updated Customer',
  quantity = 4
WHERE
  order_id = 1;
```

Screenshot:



Data Output Messages Notifications

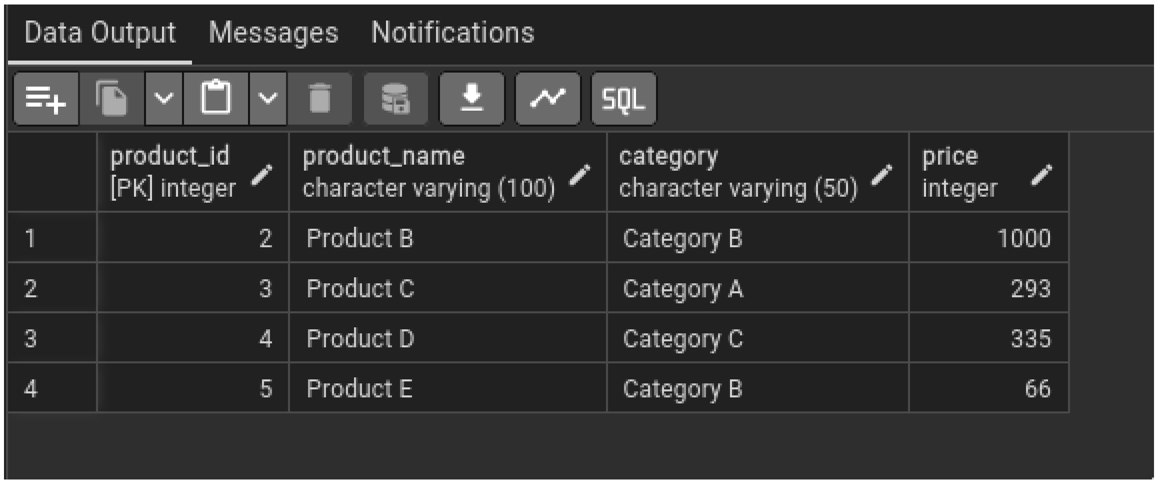
	order_id [PK] integer	customer_name character varying (100)	product_id integer	quantity integer	order_date date
1	2	Customer B	4	1	2024-07-02
2	3	Customer C	4	3	2024-07-02
3	4	Customer D	2	2	2024-07-02
4	5	Customer E	5	5	2024-07-02
5	1	Updated Customer	4	4	2024-07-02

Delete Operation

Delete a product

```sql
DELETE FROM assignment1.products
WHERE
  product_id = 1;
```

Screenshot:

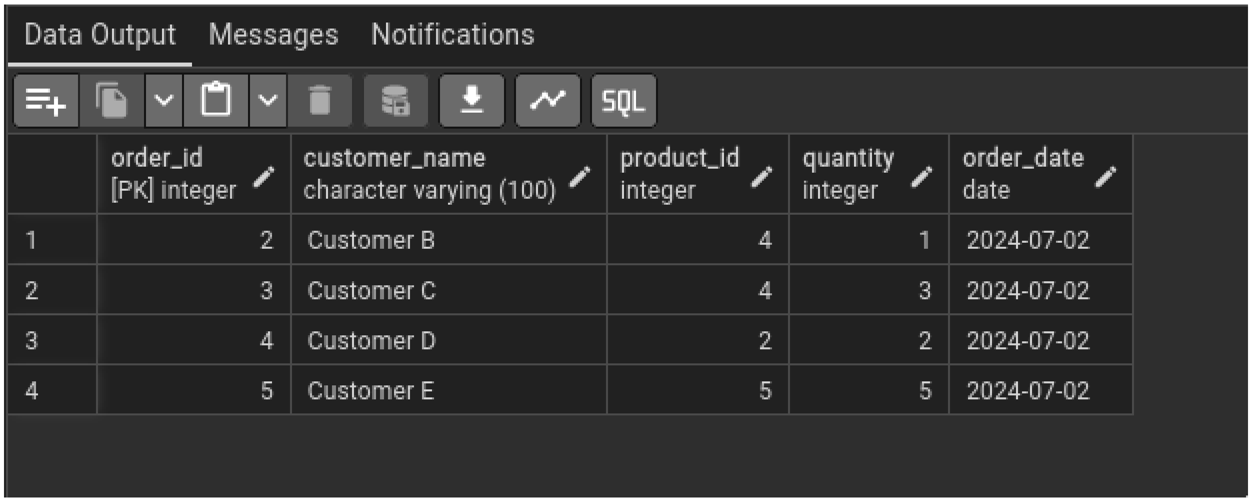


	product_id [PK] integer	product_name character varying (100)	category character varying (50)	price integer
1	2	Product B	Category B	1000
2	3	Product C	Category A	293
3	4	Product D	Category C	335
4	5	Product E	Category B	66

Delete an order

```
DELETE FROM assignment1.orders
WHERE
  order_id = 1;
```

Screenshot:

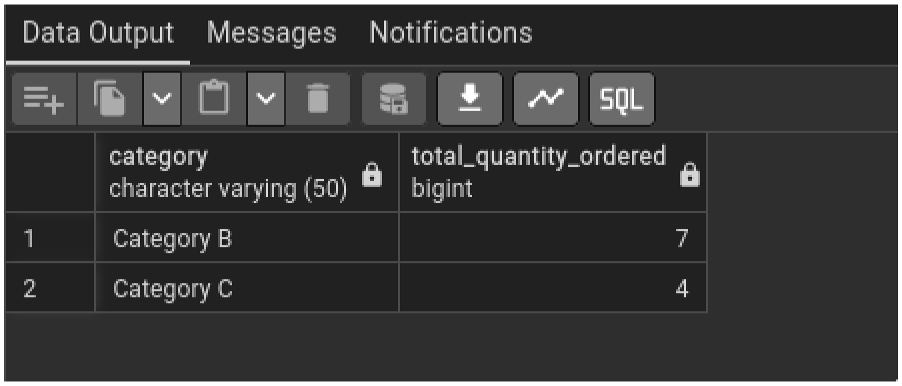


	order_id [PK] integer	customer_name character varying (100)	product_id integer	quantity integer	order_date date
1	2	Customer B	4	1	2024-07-02
2	3	Customer C	4	3	2024-07-02
3	4	Customer D	2	2	2024-07-02
4	5	Customer E	5	5	2024-07-02

Q2. Calculate the total quantity ordered for each product category in the orders table.

```
SELECT
  p.category,
  SUM(o.quantity) AS total_quantity_ordered
FROM
  assignment1.orders o
  JOIN assignment1.products p ON o.product_id = p.product_id
GROUP BY
  p.category
ORDER BY
  COUNT(o.order_id) DESC;
```

Screenshot:



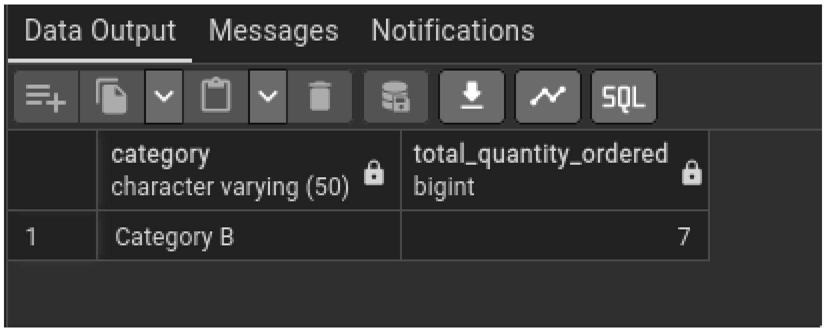
	category character varying (50)	total_quantity_ordered bigint
1	Category B	7
2	Category C	4

Q3. Find categories where the total number of products ordered is greater than 5.

```
SELECT
  p.category,
  SUM(o.quantity) AS total_quantity_ordered
FROM
  assignment1.orders o
  JOIN assignment1.products p ON o.product_id = p.product_id
GROUP BY
  p.category
HAVING
```

```
  SUM(o.quantity) > 5;
```

Screenshot:



	category character varying (50)	total_quantity_ordered bigint
1	Category B	7