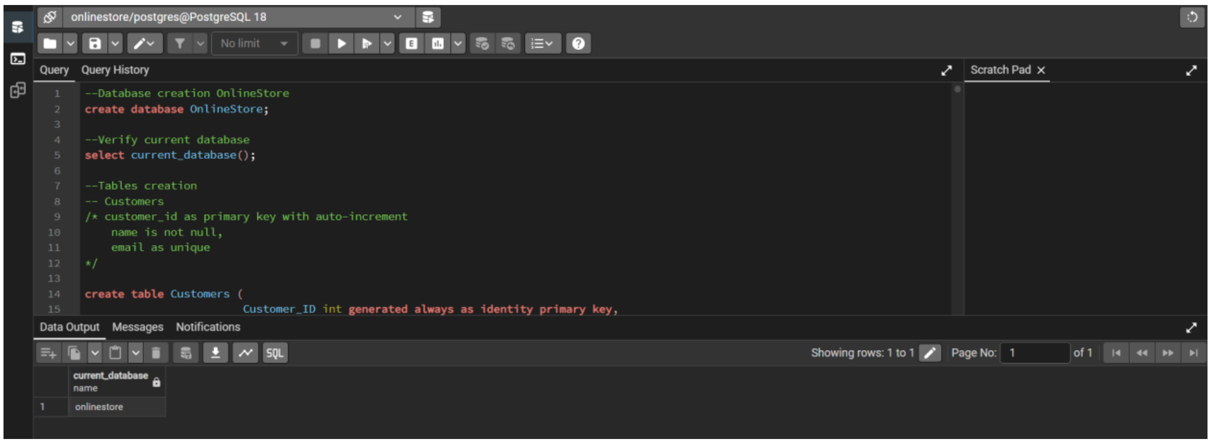
Task 3

Project: Online Store Order Management System (PostgreSQL)

Objective: Create a system to manage orders, customers, and products for an online store.

GITHUB: https://github.com/xrahulcrx/Online_Store_Order_Management_System

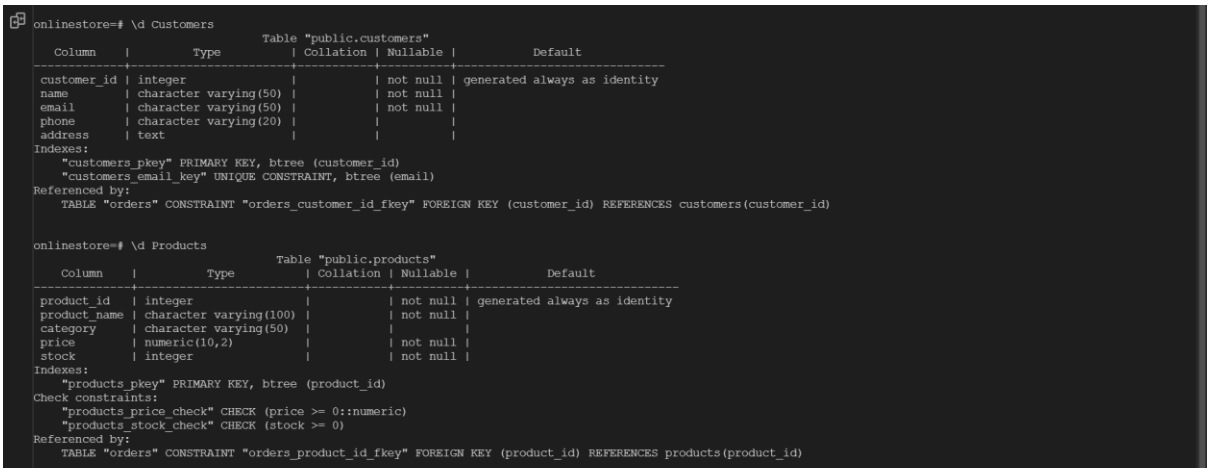
Database Creation:



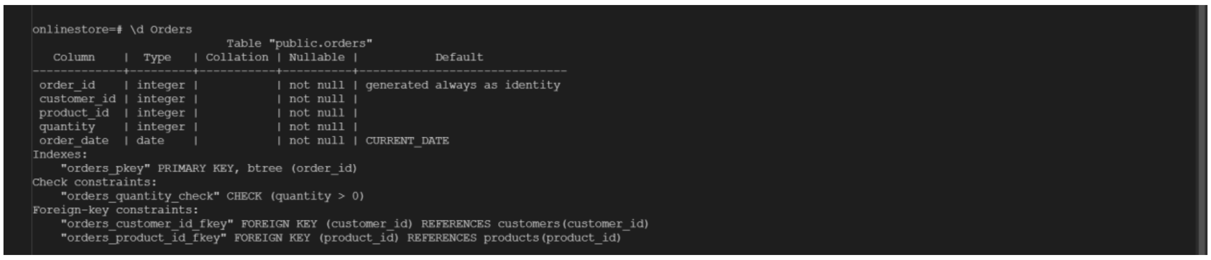
```sql
--Database creation OnlineStore
create database OnlineStore;

--Verify current database
select current_database();

--Tables creation
-- Customers
/* customer_id as primary key with auto-increment
    name is not null,
    email as unique
*/

create table Customers (
                        Customer_ID int generated always as identity primary key,
```

	current_database name
1	onlinestore

Creation of tables:

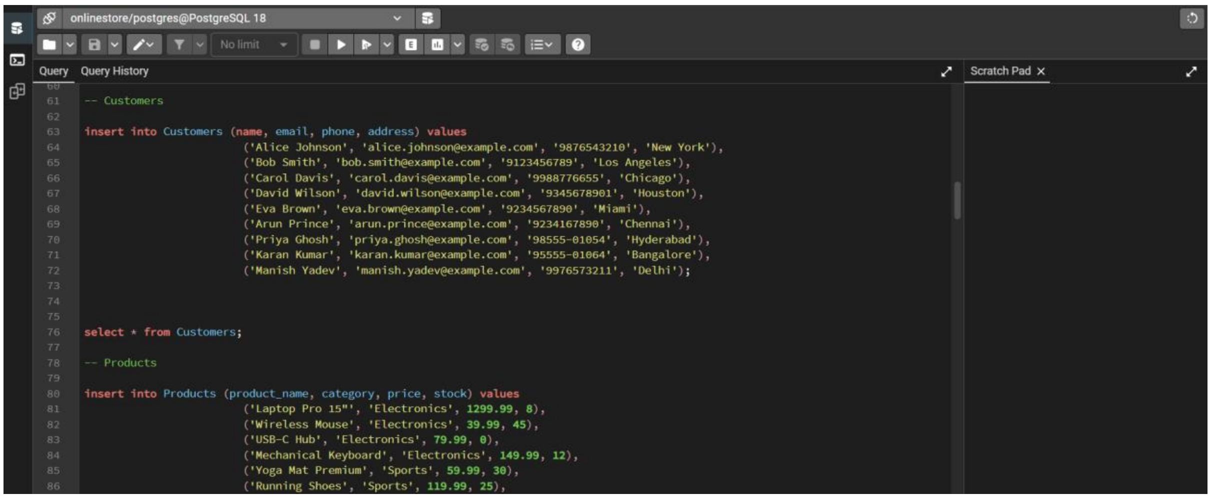


```
onlinestore=# \d Customers
                              Table "public.customers"
   Column    |         Type          | Collation | Nullable |           Default
-------------+-----------------------+-----------+----------+------------------------------
 customer_id | integer               |           | not null | generated always as identity
 name        | character varying(50) |           | not null |
 email       | character varying(50) |           | not null |
 phone       | character varying(20) |           |          |
 address     | text                  |           |          |
Indexes:
    "customers_pkey" PRIMARY KEY, btree (customer_id)
    "customers_email_key" UNIQUE CONSTRAINT, btree (email)
Referenced by:
    TABLE "orders" CONSTRAINT "orders_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)


onlinestore=# \d Products
                               Table "public.products"
    Column    |          Type          | Collation | Nullable |           Default
--------------+------------------------+-----------+----------+------------------------------
 product_id   | integer                |           | not null | generated always as identity
 product_name | character varying(100) |           | not null |
 category     | character varying(50)  |           |          |
 price        | numeric(10,2)          |           | not null |
 stock        | integer                |           | not null |
Indexes:
    "products_pkey" PRIMARY KEY, btree (product_id)
Check constraints:
    "products_price_check" CHECK (price >= 0::numeric)
    "products_stock_check" CHECK (stock >= 0)
Referenced by:
    TABLE "orders" CONSTRAINT "orders_product_id_fkey" FOREIGN KEY (product_id) REFERENCES products(product_id)
```

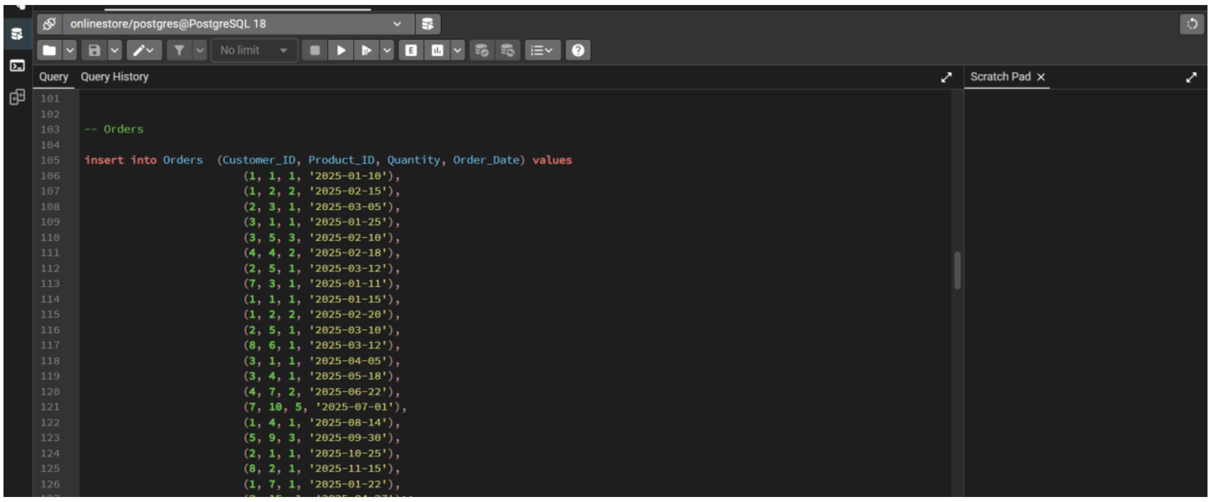


```
onlinestore=# \d Orders
                          Table "public.orders"
   Column    |  Type   | Collation | Nullable |           Default
-------------+---------+-----------+----------+------------------------------
 order_id    | integer |           | not null | generated always as identity
 customer_id | integer |           | not null |
 product_id  | integer |           | not null |
 quantity    | integer |           | not null |
 order_date  | date    |           | not null | CURRENT_DATE
Indexes:
    "orders_pkey" PRIMARY KEY, btree (order_id)
Check constraints:
    "orders_quantity_check" CHECK (quantity > 0)
Foreign-key constraints:
    "orders_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    "orders_product_id_fkey" FOREIGN KEY (product_id) REFERENCES products(product_id)
```

Insert sample data for customers, products, and orders.



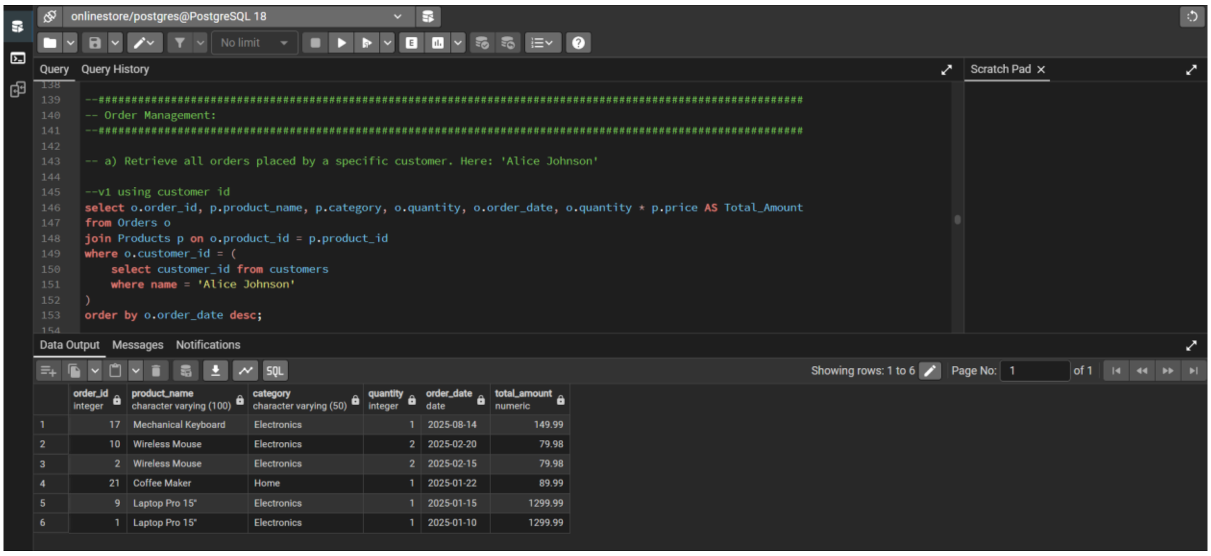
```sql
-- Customers

insert into Customers (name, email, phone, address) values
                        ('Alice Johnson', 'alice.johnson@example.com', '9876543210', 'New York'),
                        ('Bob Smith', 'bob.smith@example.com', '9123456789', 'Los Angeles'),
                        ('Carol Davis', 'carol.davis@example.com', '9988776655', 'Chicago'),
                        ('David Wilson', 'david.wilson@example.com', '9345678901', 'Houston'),
                        ('Eva Brown', 'eva.brown@example.com', '9234567890', 'Miami'),
                        ('Arun Prince', 'arun.prince@example.com', '9234167890', 'Chennai'),
                        ('Priya Ghosh', 'priya.ghosh@example.com', '98555-01054', 'Hyderabad'),
                        ('Karan Kumar', 'karan.kumar@example.com', '95555-01064', 'Bangalore'),
                        ('Manish Yadev', 'manish.yadev@example.com', '9976573211', 'Delhi');



select * from Customers;

-- Products

insert into Products (product_name, category, price, stock) values
                        ('Laptop Pro 15"', 'Electronics', 1299.99, 8),
                        ('Wireless Mouse', 'Electronics', 39.99, 45),
                        ('USB-C Hub', 'Electronics', 79.99, 0),
                        ('Mechanical Keyboard', 'Electronics', 149.99, 12),
                        ('Yoga Mat Premium', 'Sports', 59.99, 30),
                        ('Running Shoes', 'Sports', 119.99, 25),
```



```sql


-- Orders

insert into Orders  (Customer_ID, Product_ID, Quantity, Order_Date) values
                        (1, 1, 1, '2025-01-10'),
                        (1, 2, 2, '2025-02-15'),
                        (2, 3, 1, '2025-03-05'),
                        (3, 1, 1, '2025-01-25'),
                        (3, 5, 3, '2025-02-10'),
                        (4, 4, 2, '2025-02-18'),
                        (2, 5, 1, '2025-03-12'),
                        (7, 3, 1, '2025-01-11'),
                        (1, 1, 1, '2025-01-15'),
                        (1, 2, 2, '2025-02-20'),
                        (2, 5, 1, '2025-03-10'),
                        (8, 6, 1, '2025-03-12'),
                        (3, 1, 1, '2025-04-05'),
                        (3, 4, 1, '2025-05-18'),
                        (4, 7, 2, '2025-06-22'),
                        (7, 10, 5, '2025-07-01'),
                        (1, 4, 1, '2025-08-14'),
                        (5, 9, 3, '2025-09-30'),
                        (2, 1, 1, '2025-10-25'),
                        (8, 2, 1, '2025-11-15'),
                        (1, 7, 1, '2025-01-22'),
```

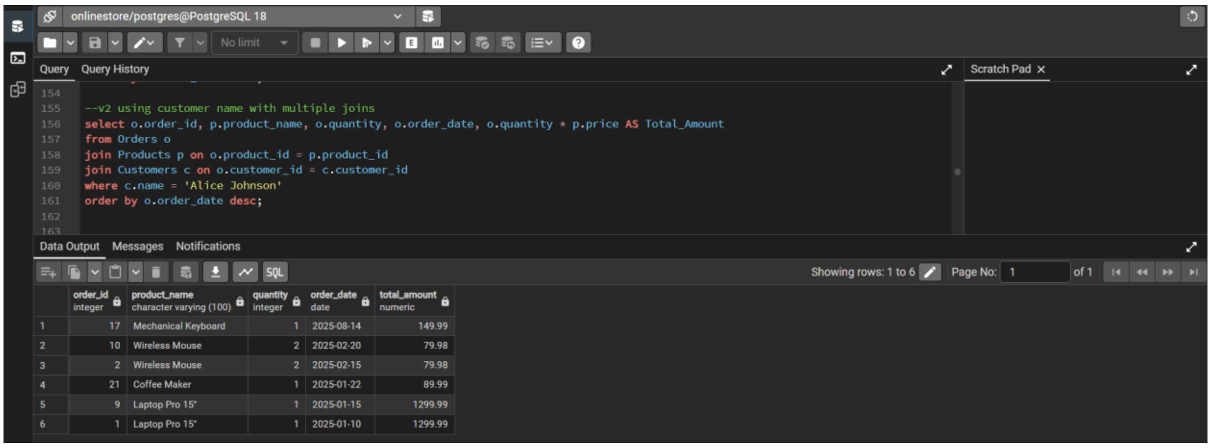
Order Management:

a) **Retrieve all orders placed by a specific customer.**



```sql
--##########################################################################################################
-- Order Management:
--##########################################################################################################

-- a) Retrieve all orders placed by a specific customer. Here: 'Alice Johnson'

--v1 using customer id
select o.order_id, p.product_name, p.category, o.quantity, o.order_date, o.quantity * p.price AS Total_Amount
from Orders o
join Products p on o.product_id = p.product_id
where o.customer_id = (
    select customer_id from customers
    where name = 'Alice Johnson'
)
order by o.order_date desc;
```

```
Showing rows: 1 to 6    Page No: 1 of 1

   order_id  product_name             category                quantity  order_date  total_amount
   integer   character varying (100)  character varying (50)  integer   date        numeric
1        17  Mechanical Keyboard      Electronics                    1  2025-08-14        149.99
2        10  Wireless Mouse           Electronics                    2  2025-02-20         79.98
3         2  Wireless Mouse           Electronics                    2  2025-02-15         79.98
4        21  Coffee Maker             Home                           1  2025-01-22         89.99
5         9  Laptop Pro 15"           Electronics                    1  2025-01-15       1299.99
6         1  Laptop Pro 15"           Electronics                    1  2025-01-10       1299.99
```

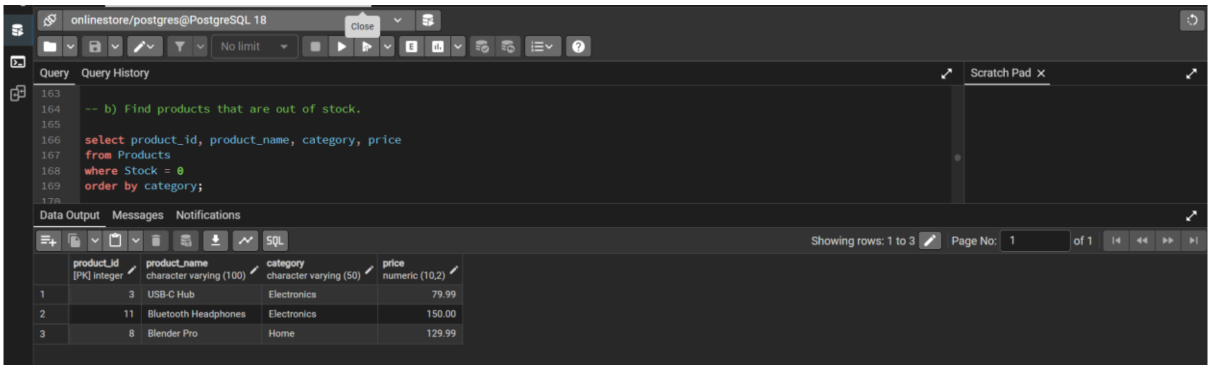


```sql
--v2 using customer name with multiple joins
select o.order_id, p.product_name, o.quantity, o.order_date, o.quantity * p.price AS Total_Amount
from Orders o
join Products p on o.product_id = p.product_id
join Customers c on o.customer_id = c.customer_id
where c.name = 'Alice Johnson'
order by o.order_date desc;
```

Showing rows: 1 to 6 Page No: 1 of 1

	order_id integer	product_name character varying (100)	quantity integer	order_date date	total_amount numeric
1	17	Mechanical Keyboard	1	2025-08-14	149.99
2	10	Wireless Mouse	2	2025-02-20	79.98
3	2	Wireless Mouse	2	2025-02-15	79.98
4	21	Coffee Maker	1	2025-01-22	89.99
5	9	Laptop Pro 15"	1	2025-01-15	1299.99
6	1	Laptop Pro 15"	1	2025-01-10	1299.99

b) Find products that are out of stock.

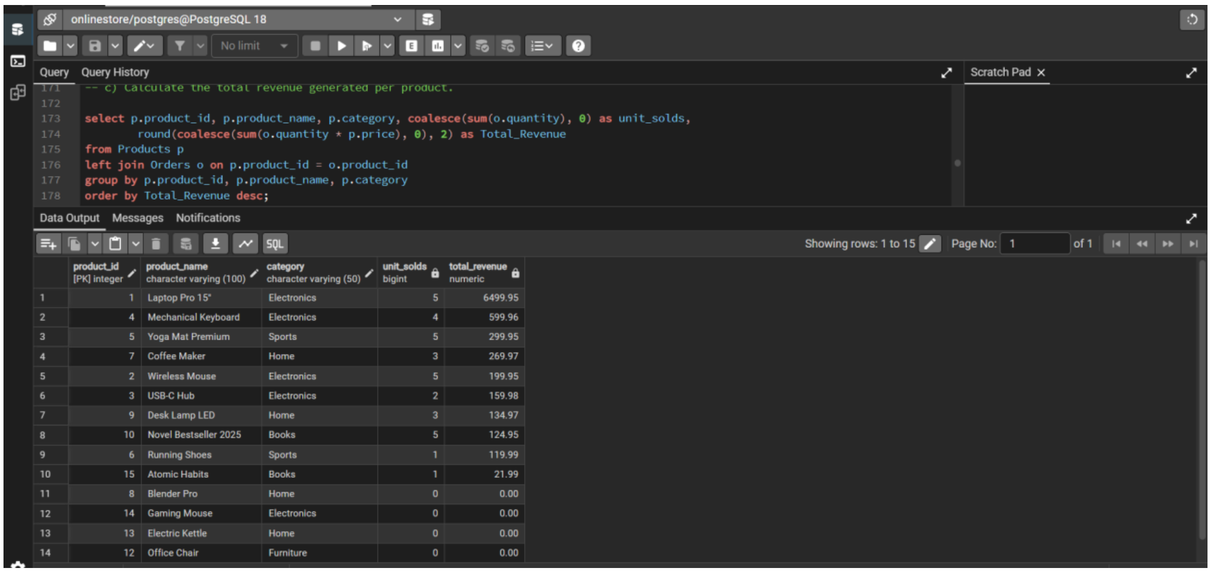


```sql
-- b) Find products that are out of stock.

select product_id, product_name, category, price
from Products
where Stock = 0
order by category;
```

Showing rows: 1 to 3 Page No: 1 of 1

	product_id [PK] integer	product_name character varying (100)	category character varying (50)	price numeric (10,2)
1	3	USB-C Hub	Electronics	79.99
2	11	Bluetooth Headphones	Electronics	150.00
3	8	Blender Pro	Home	129.99

c) Calculate the total revenue generated per product.

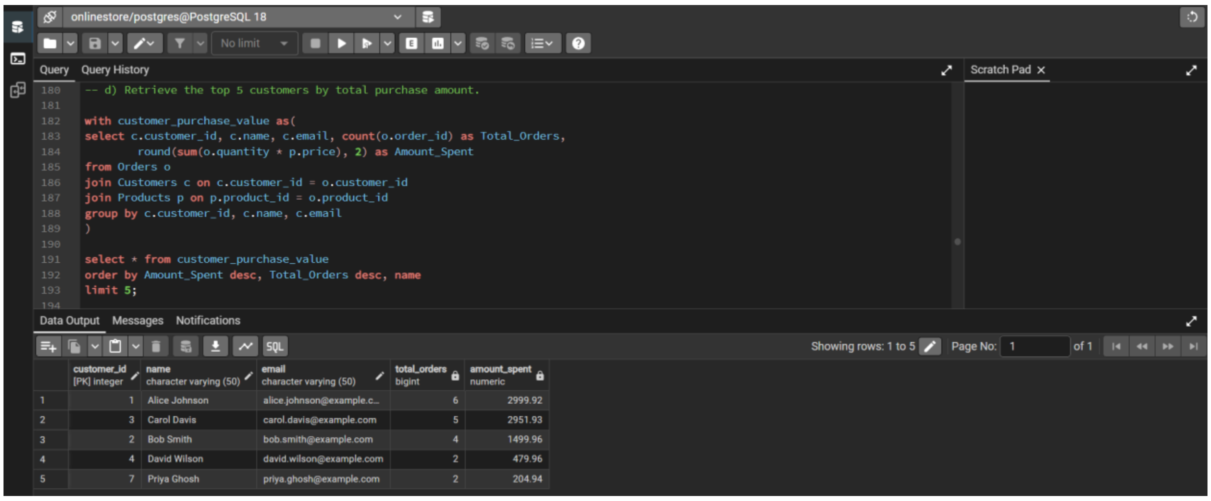


```sql
-- c) Calculate the total revenue generated per product.

select p.product_id, p.product_name, p.category, coalesce(sum(o.quantity), 0) as unit_solds,
       round(coalesce(sum(o.quantity * p.price), 0), 2) as Total_Revenue
from Products p
left join Orders o on p.product_id = o.product_id
group by p.product_id, p.product_name, p.category
order by Total_Revenue desc;
```

Showing rows: 1 to 15 Page No: 1 of 1

	product_id [PK] integer	product_name character varying (100)	category character varying (50)	unit_solds bigint	total_revenue numeric
1	1	Laptop Pro 15"	Electronics	5	6499.95
2	4	Mechanical Keyboard	Electronics	4	599.96
3	5	Yoga Mat Premium	Sports	5	299.95
4	7	Coffee Maker	Home	3	269.97
5	2	Wireless Mouse	Electronics	5	199.95
6	3	USB-C Hub	Electronics	2	159.98
7	9	Desk Lamp LED	Home	3	134.97
8	10	Novel Bestseller 2025	Books	5	124.95
9	6	Running Shoes	Sports	1	119.99
10	15	Atomic Habits	Books	1	21.99
11	8	Blender Pro	Home	0	0.00
12	14	Gaming Mouse	Electronics	0	0.00
13	13	Electric Kettle	Home	0	0.00
14	12	Office Chair	Furniture	0	0.00

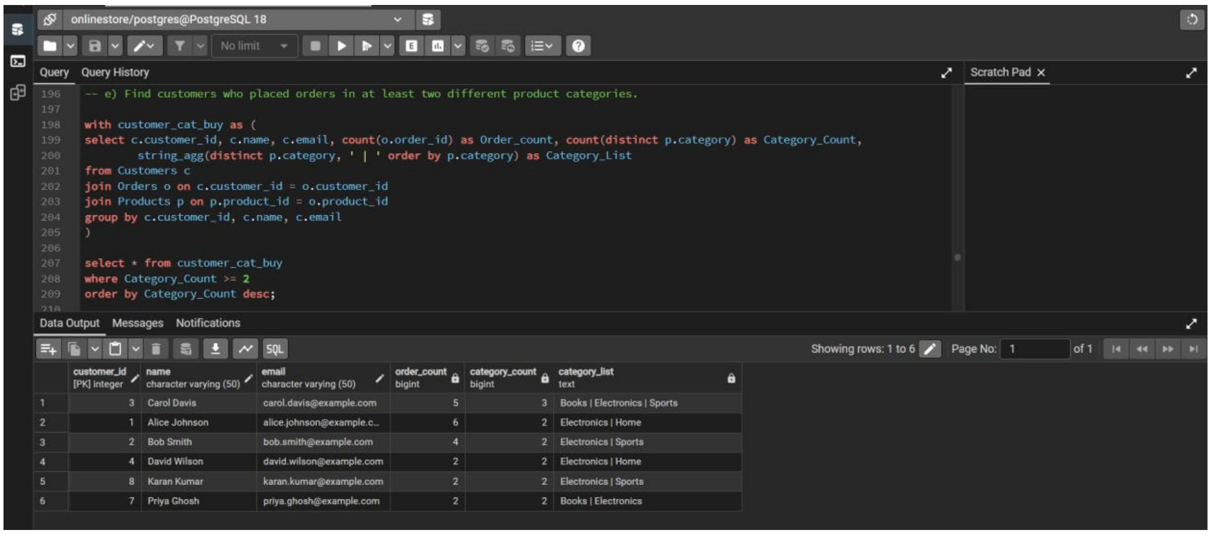
d) Retrieve the top 5 customers by total purchase amount.



```sql
-- d) Retrieve the top 5 customers by total purchase amount.

with customer_purchase_value as(
select c.customer_id, c.name, c.email, count(o.order_id) as Total_Orders,
        round(sum(o.quantity * p.price), 2) as Amount_Spent
from Orders o
join Customers c on c.customer_id = o.customer_id
join Products p on p.product_id = o.product_id
group by c.customer_id, c.name, c.email
)

select * from customer_purchase_value
order by Amount_Spent desc, Total_Orders desc, name
limit 5;
```

	customer_id [PK] integer	name character varying (50)	email character varying (50)	total_orders bigint	amount_spent numeric
1	1	Alice Johnson	alice.johnson@example.c...	6	2999.92
2	3	Carol Davis	carol.davis@example.com	5	2951.93
3	2	Bob Smith	bob.smith@example.com	4	1499.96
4	4	David Wilson	david.wilson@example.com	2	479.96
5	7	Priya Ghosh	priya.ghosh@example.com	2	204.94

e) Find customers who placed orders in at least two different product categories.

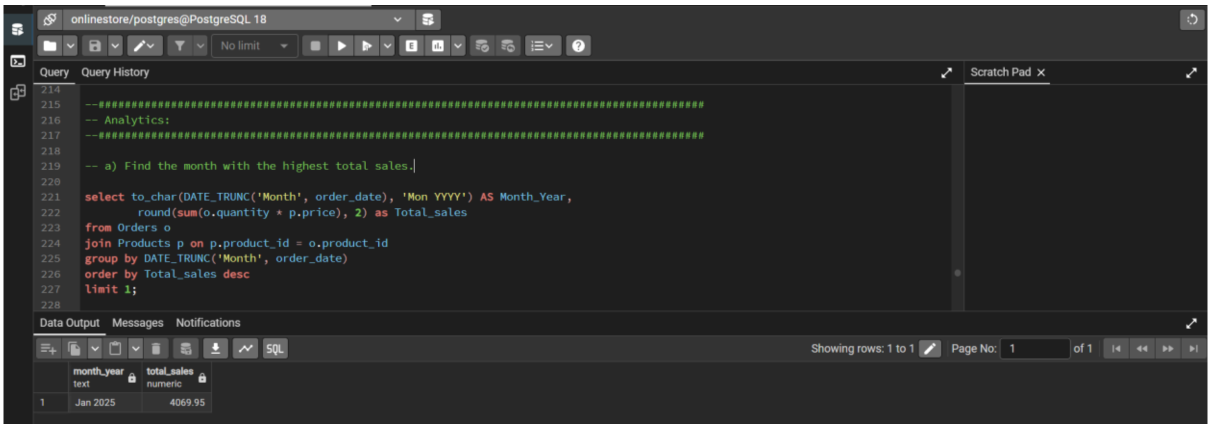


```sql
-- e) Find customers who placed orders in at least two different product categories.

with customer_cat_buy as (
select c.customer_id, c.name, c.email, count(o.order_id) as Order_count, count(distinct p.category) as Category_Count,
        string_agg(distinct p.category, ' | ' order by p.category) as Category_List
from Customers c
join Orders o on c.customer_id = o.customer_id
join Products p on p.product_id = o.product_id
group by c.customer_id, c.name, c.email
)

select * from customer_cat_buy
where Category_Count >= 2
order by Category_Count desc;
```

	customer_id [PK] integer	name character varying (50)	email character varying (50)	order_count bigint	category_count bigint	category_list text
1	3	Carol Davis	carol.davis@example.com	5	3	Books | Electronics | Sports
2	1	Alice Johnson	alice.johnson@example.c...	6	2	Electronics | Home
3	2	Bob Smith	bob.smith@example.com	4	2	Electronics | Sports
4	4	David Wilson	david.wilson@example.com	2	2	Electronics | Home
5	8	Karan Kumar	karan.kumar@example.com	2	2	Electronics | Sports
6	7	Priya Ghosh	priya.ghosh@example.com	2	2	Books | Electronics

Analytics:

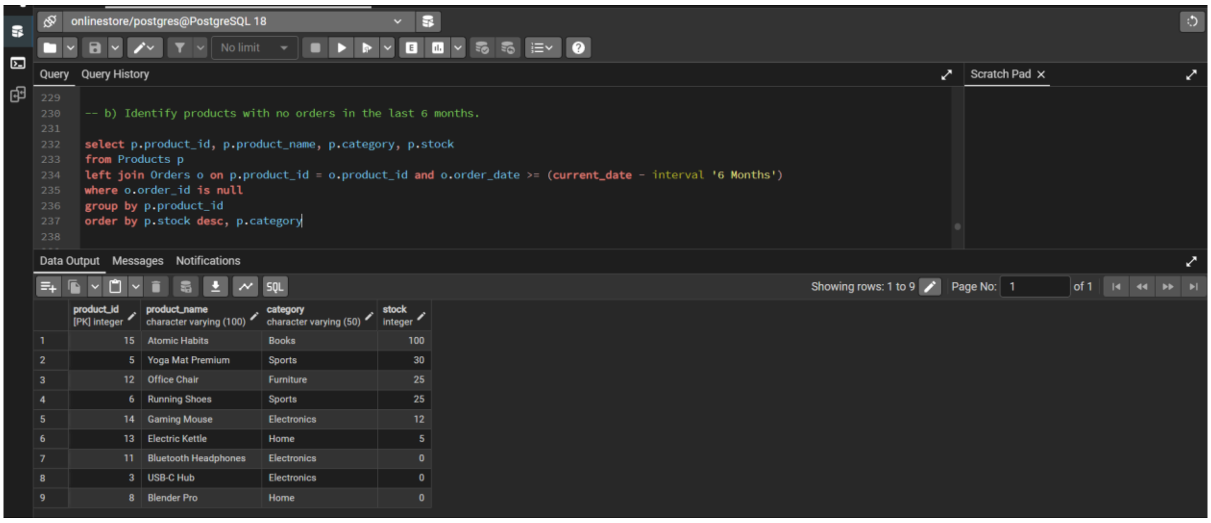
a) Find the month with the highest total sales.



```sql

--###########################################################################################
-- Analytics:
--###########################################################################################

-- a) Find the month with the highest total sales.

select to_char(DATE_TRUNC('Month', order_date), 'Mon YYYY') AS Month_Year,
        round(sum(o.quantity * p.price), 2) as Total_sales
from Orders o
join Products p on p.product_id = o.product_id
group by DATE_TRUNC('Month', order_date)
order by Total_sales desc
limit 1;
```

	month_year text	total_sales numeric
1	Jan 2025	4069.95

b) Identify products with no orders in the last 6 months.



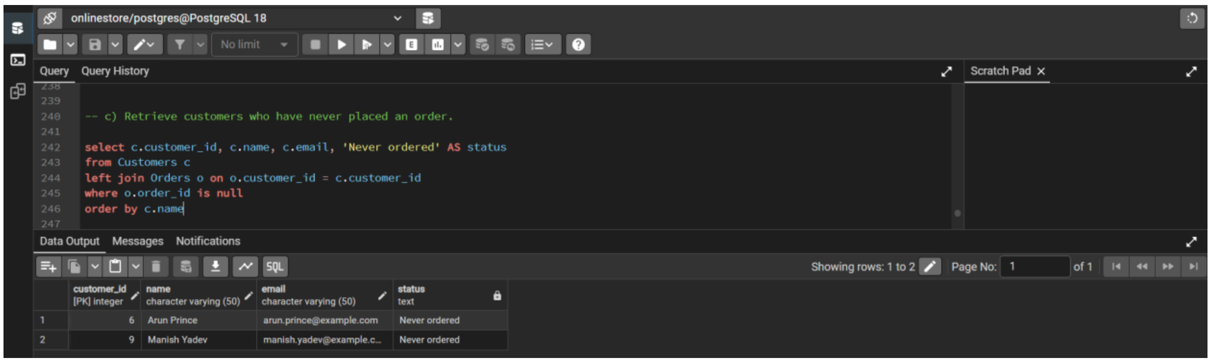
onlinestore/postgres@PostgreSQL 18

```sql
-- b) Identify products with no orders in the last 6 months.

select p.product_id, p.product_name, p.category, p.stock
from Products p
left join Orders o on p.product_id = o.product_id and o.order_date >= (current_date - interval '6 Months')
where o.order_id is null
group by p.product_id
order by p.stock desc, p.category
```

Showing rows: 1 to 9 Page No: 1 of 1

	product_id [PK] integer	product_name character varying (100)	category character varying (50)	stock integer
1	15	Atomic Habits	Books	100
2	5	Yoga Mat Premium	Sports	30
3	12	Office Chair	Furniture	25
4	6	Running Shoes	Sports	25
5	14	Gaming Mouse	Electronics	12
6	13	Electric Kettle	Home	5
7	11	Bluetooth Headphones	Electronics	0
8	3	USB-C Hub	Electronics	0
9	8	Blender Pro	Home	0

c) Retrieve customers who have never placed an order.



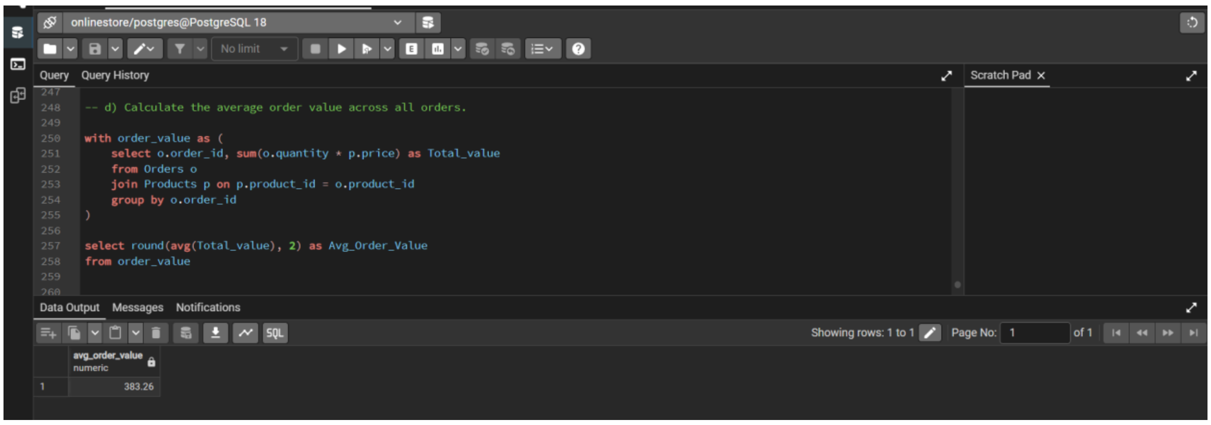
onlinestore/postgres@PostgreSQL 18

```sql
-- c) Retrieve customers who have never placed an order.

select c.customer_id, c.name, c.email, 'Never ordered' AS status
from Customers c
left join Orders o on o.customer_id = c.customer_id
where o.order_id is null
order by c.name
```

Showing rows: 1 to 2 Page No: 1 of 1

	customer_id [PK] integer	name character varying (50)	email character varying (50)	status text
1	6	Arun Prince	arun.prince@example.com	Never ordered
2	9	Manish Yadev	manish.yadev@example.c...	Never ordered

d) Calculate the average order value across all orders.



onlinestore/postgres@PostgreSQL 18

```sql
-- d) Calculate the average order value across all orders.

with order_value as (
    select o.order_id, sum(o.quantity * p.price) as Total_value
    from Orders o
    join Products p on p.product_id = o.product_id
    group by o.order_id
)

select round(avg(Total_value), 2) as Avg_Order_Value
from order_value
```

Showing rows: 1 to 1 Page No: 1 of 1

	avg_order_value numeric
1	383.26