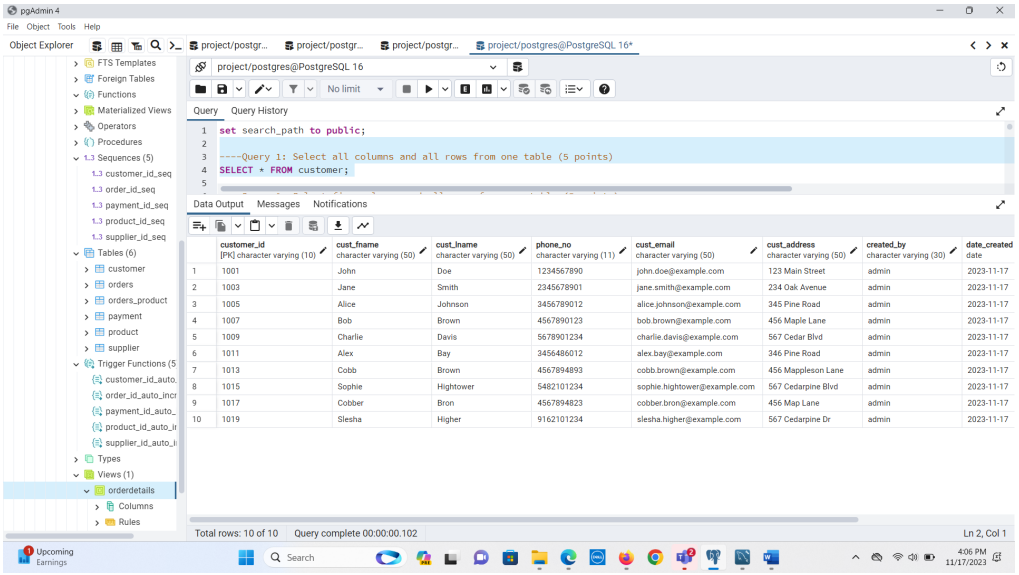
BUAN 6320.0W2 - Group Project
Group No.3
Task: Queries with their output

Members: **Roopali Reddy Kallem (RXK180054)**
Nguyen Thanh Nguyen (NXN210036)
Nihar Vinnakota (NXV230017)
Haritha Jampani (HXJ220031)

----**Query 1**: Select all columns and all rows from one table (5 points)

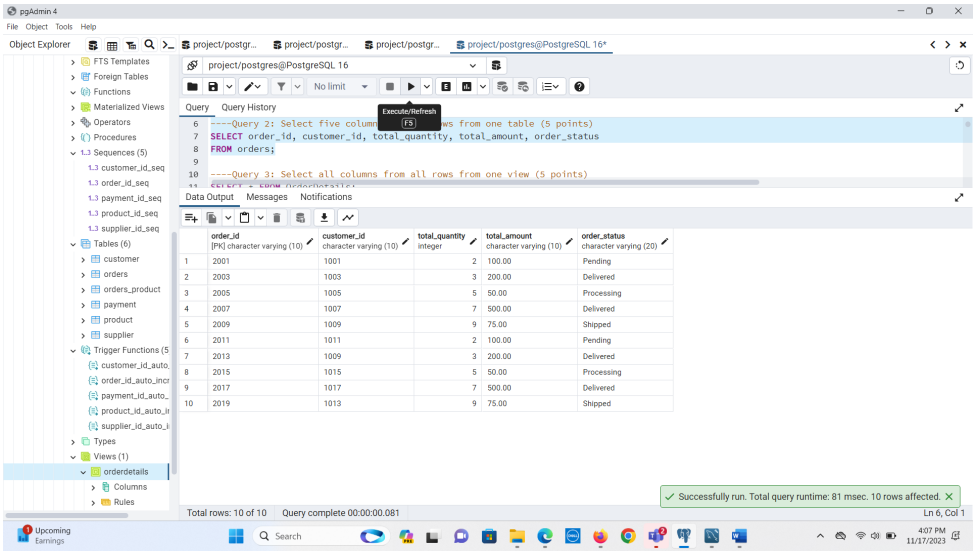
SELECT * FROM customer;



----**Query 2**: Select five columns and all rows from one table (5 points)

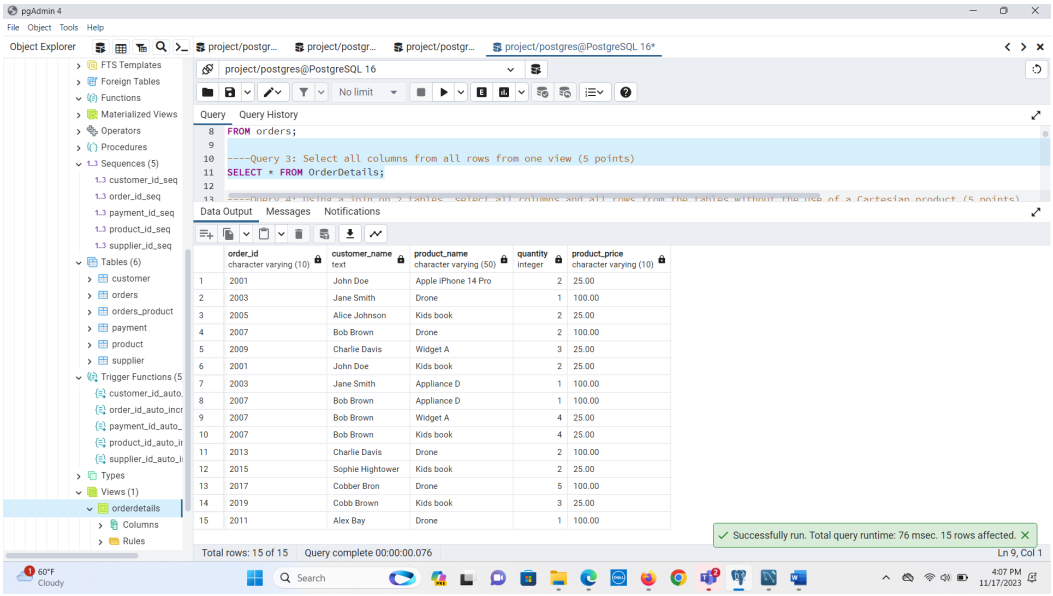
SELECT order_id, customer_id, total_quantity, total_amount, order_status

FROM orders;



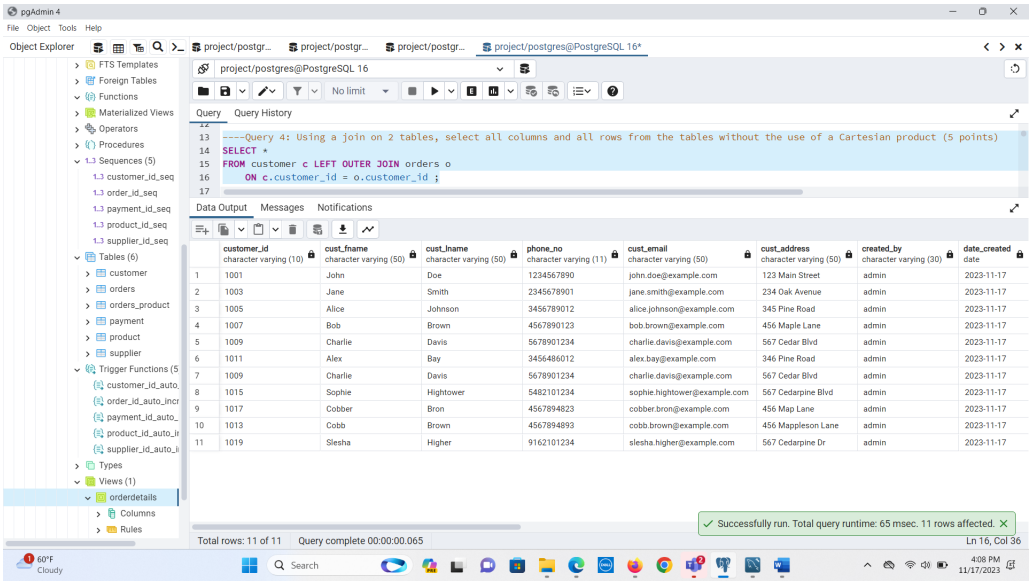
----**Query 3**: Select all columns from all rows from one view (5 points)

SELECT * FROM OrderDetails;



----**Query 4**: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product (5 points)

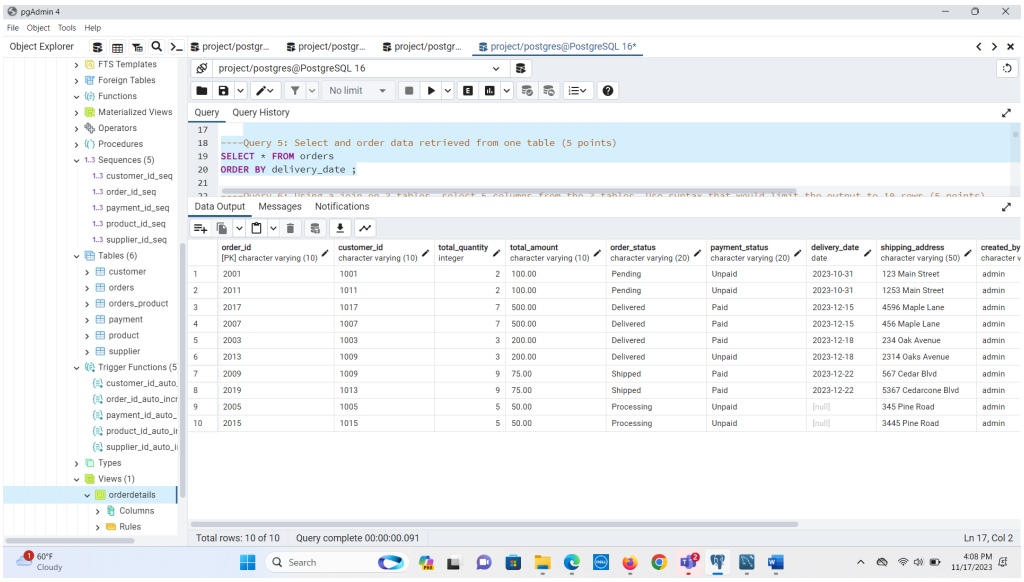
SELECT * FROM customer c LEFT OUTER JOIN orders o ON c.customer_id = o.customer_id ;



----**Query 5**: Select and order data retrieved from one table (5 points)

SELECT * FROM orders

ORDER BY delivery_date ;



	order_id [PK] character varying (10)	customer_id character varying (10)	total_quantity integer	total_amount character varying (10)	order_status character varying (20)	payment_status character varying (20)	delivery_date date	shipping_address character varying (50)	created_by character v
1	2001	1001	2	100.00	Pending	Unpaid	2023-10-31	123 Main Street	admin
2	2011	1011	2	100.00	Pending	Unpaid	2023-10-31	1253 Main Street	admin
3	2017	1017	7	500.00	Delivered	Paid	2023-12-15	4596 Maple Lane	admin
4	2007	1007	7	500.00	Delivered	Paid	2023-12-15	456 Maple Lane	admin
5	2003	1003	3	200.00	Delivered	Paid	2023-12-18	234 Oak Avenue	admin
6	2013	1009	3	200.00	Delivered	Unpaid	2023-12-18	2314 Oaks Avenue	admin
7	2009	1009	9	75.00	Shipped	Paid	2023-12-22	567 Cedar Blvd	admin
8	2019	1013	9	75.00	Shipped	Paid	2023-12-22	5367 Cedarcone Blvd	admin
9	2005	1005	5	50.00	Processing	Unpaid	[null]	345 Pine Road	admin
10	2015	1015	5	50.00	Processing	Unpaid	[null]	3445 Pine Road	admin

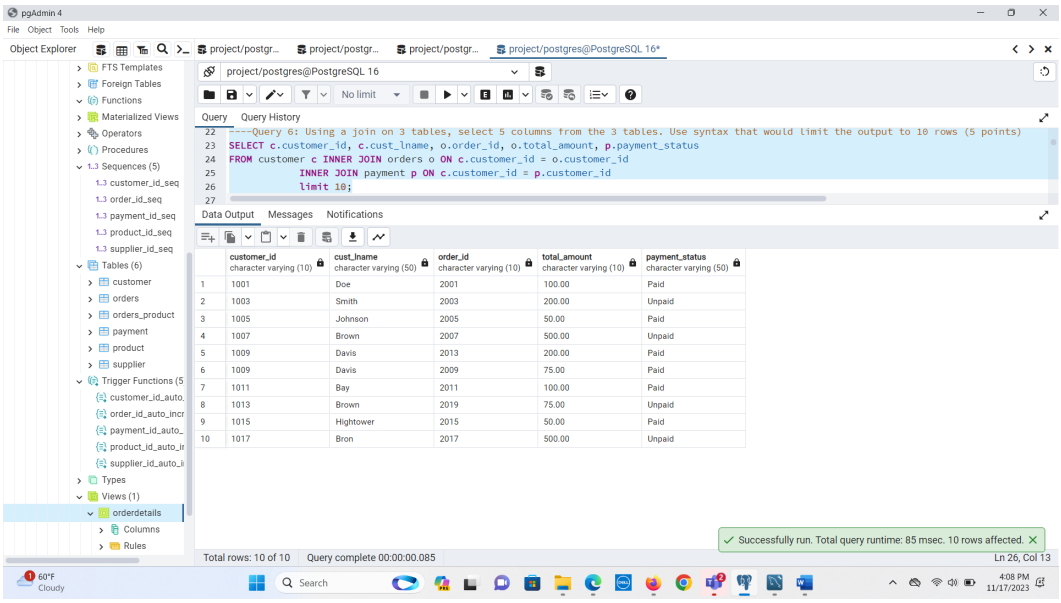
----**Query 6**: Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows (5 points)

SELECT c.customer_id, c.cust_lname, o.order_id, o.total_amount, p.payment_status

FROM customer c INNER JOIN orders o ON c.customer_id = o.customer_id

INNER JOIN payment p ON c.customer_id = p.customer_id

limit 10;

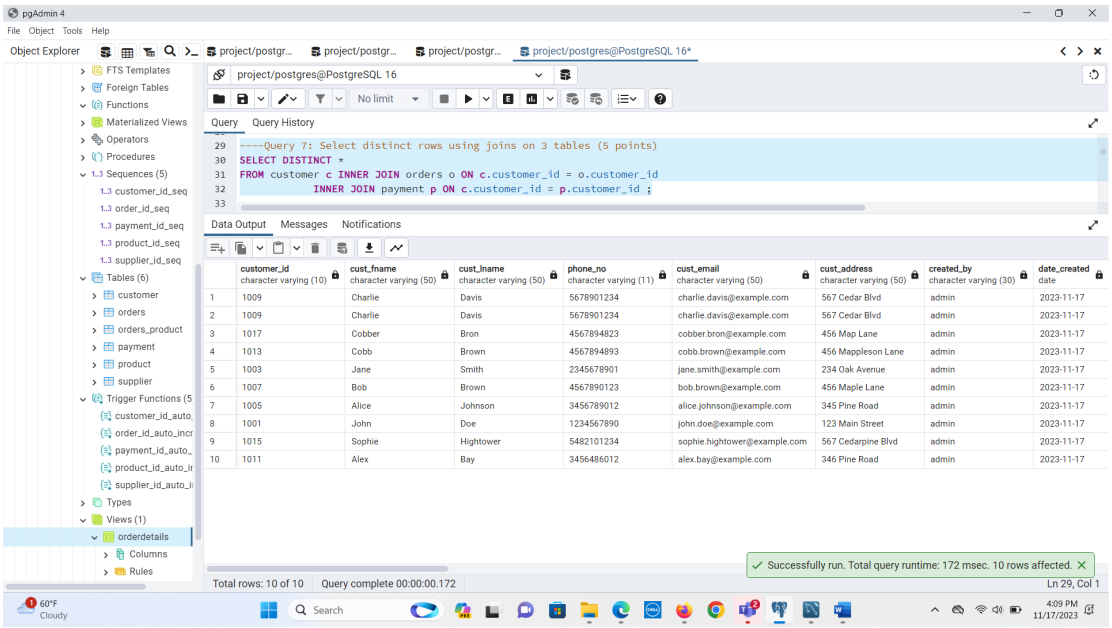


----**Query 7**: Select distinct rows using joins on 3 tables (5 points)

SELECT DISTINCT *

FROM customer c INNER JOIN orders o ON c.customer_id = o.customer_id

INNER JOIN payment p ON c.customer_id = p.customer_id ;



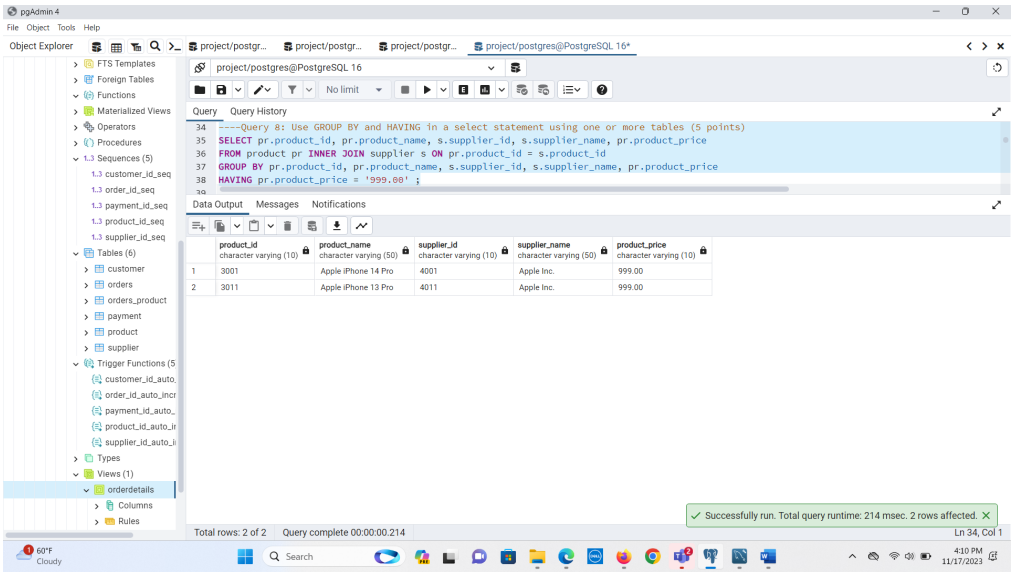
----**Query 8:** Use GROUP BY and HAVING in a select statement using one or more tables (5 points)

SELECT pr.product_id, pr.product_name, s.supplier_id, s.supplier_name, pr.product_price

FROM product pr INNER JOIN supplier s ON pr.product_id = s.product_id

GROUP BY pr.product_id, pr.product_name, s.supplier_id, s.supplier_name, pr.product_price

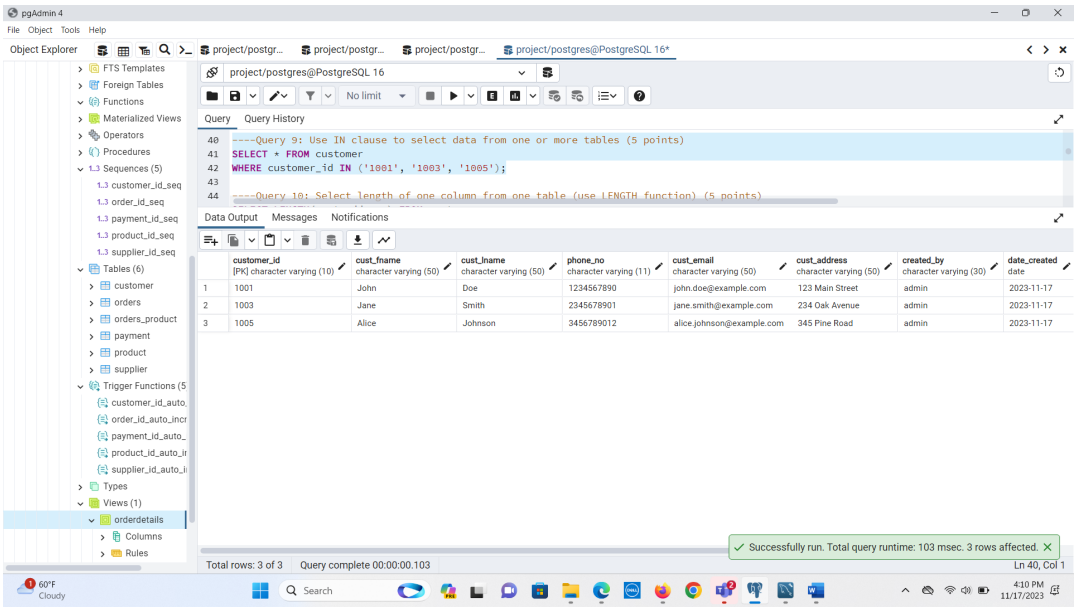
HAVING pr.product_price = '999.00' ;



----**Query 9:** Use IN clause to select data from one or more tables (5 points)

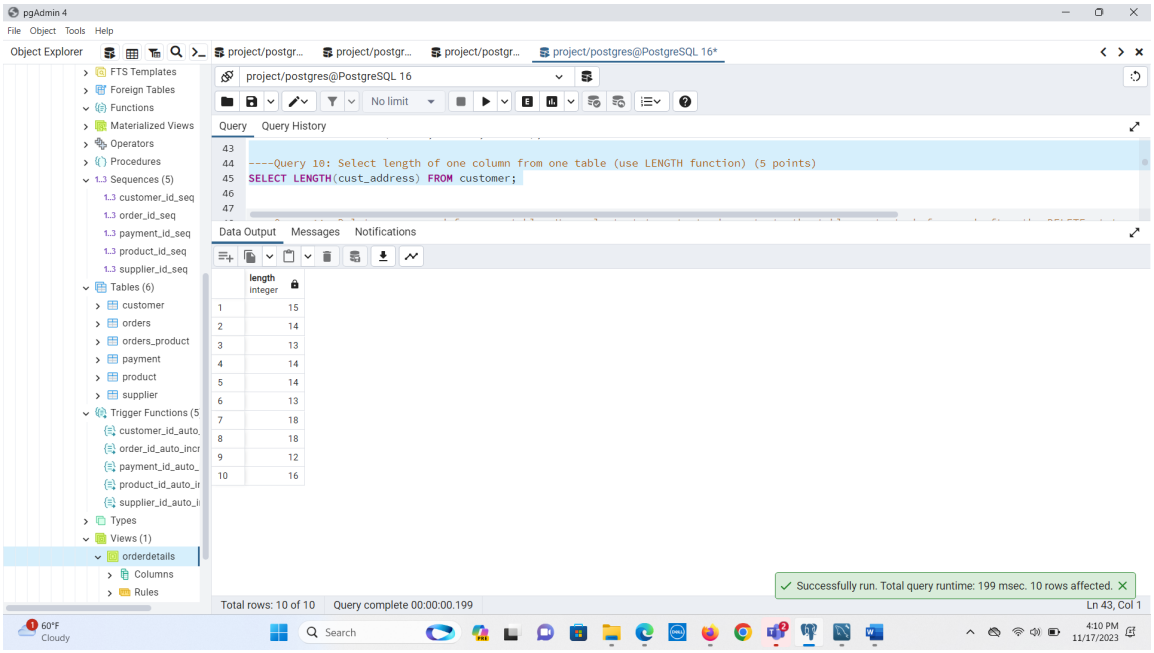
SELECT * FROM customer

WHERE customer_id IN ('1001', '1003', '1005');



----**Query 10:** Select length of one column from one table (use LENGTH function) (5 points)

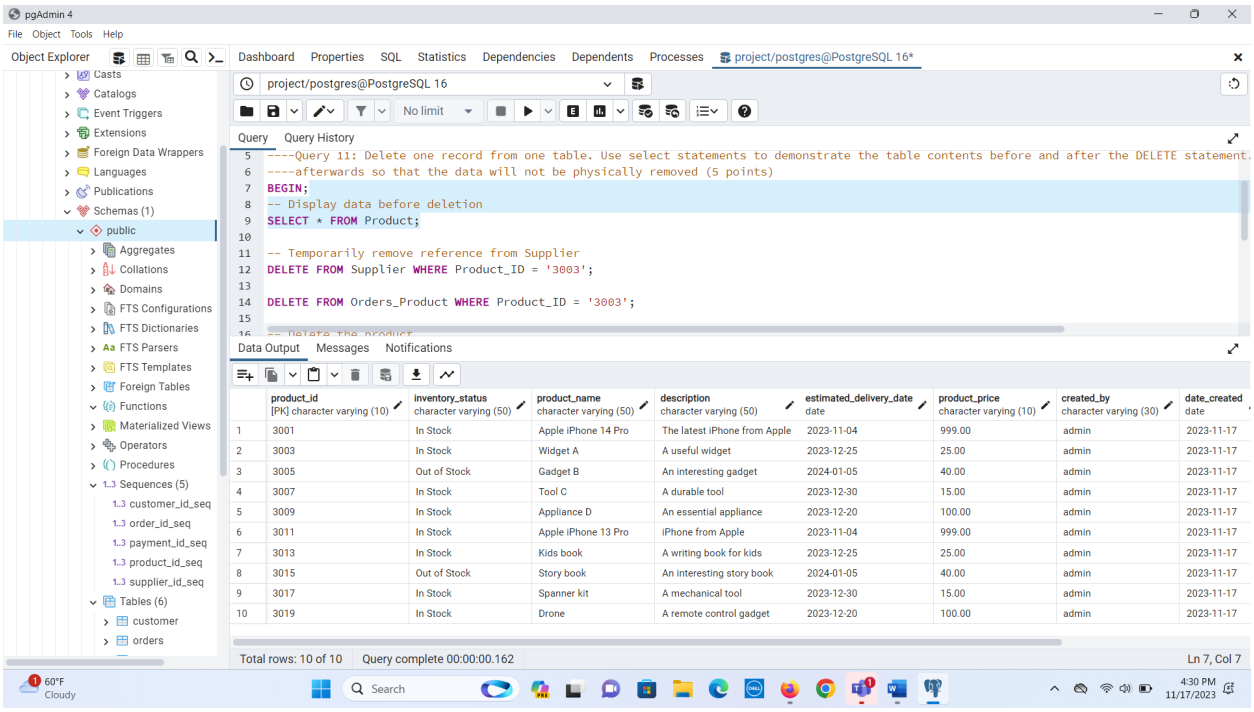
SELECT LENGTH(cust_address) FROM customer;



----**Query 11**: Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACK

----afterwards so that the data will not be physically removed (5 points)

SELECT * FROM Product;



	product_id [PK] character varying (10)	inventory_status character varying (50)	product_name character varying (50)	description character varying (50)	estimated_delivery_date date	product_price character varying (10)	created_by character varying (30)	date_created date
1	3001	In Stock	Apple iPhone 14 Pro	The latest iPhone from Apple	2023-11-04	999.00	admin	2023-11-17
2	3003	In Stock	Widget A	A useful widget	2023-12-25	25.00	admin	2023-11-17
3	3005	Out of Stock	Gadget B	An interesting gadget	2024-01-05	40.00	admin	2023-11-17
4	3007	In Stock	Tool C	A durable tool	2023-12-30	15.00	admin	2023-11-17
5	3009	In Stock	Appliance D	An essential appliance	2023-12-20	100.00	admin	2023-11-17
6	3011	In Stock	Apple iPhone 13 Pro	iPhone from Apple	2023-11-04	999.00	admin	2023-11-17
7	3013	In Stock	Kids book	A writing book for kids	2023-12-25	25.00	admin	2023-11-17
8	3015	Out of Stock	Story book	An interesting story book	2024-01-05	40.00	admin	2023-11-17
9	3017	In Stock	Spanner kit	A mechanical tool	2023-12-30	15.00	admin	2023-11-17
10	3019	In Stock	Drone	A remote control gadget	2023-12-20	100.00	admin	2023-11-17

-- Temporarily remove reference from Supplier

DELETE FROM Supplier WHERE Product_ID = '3003';

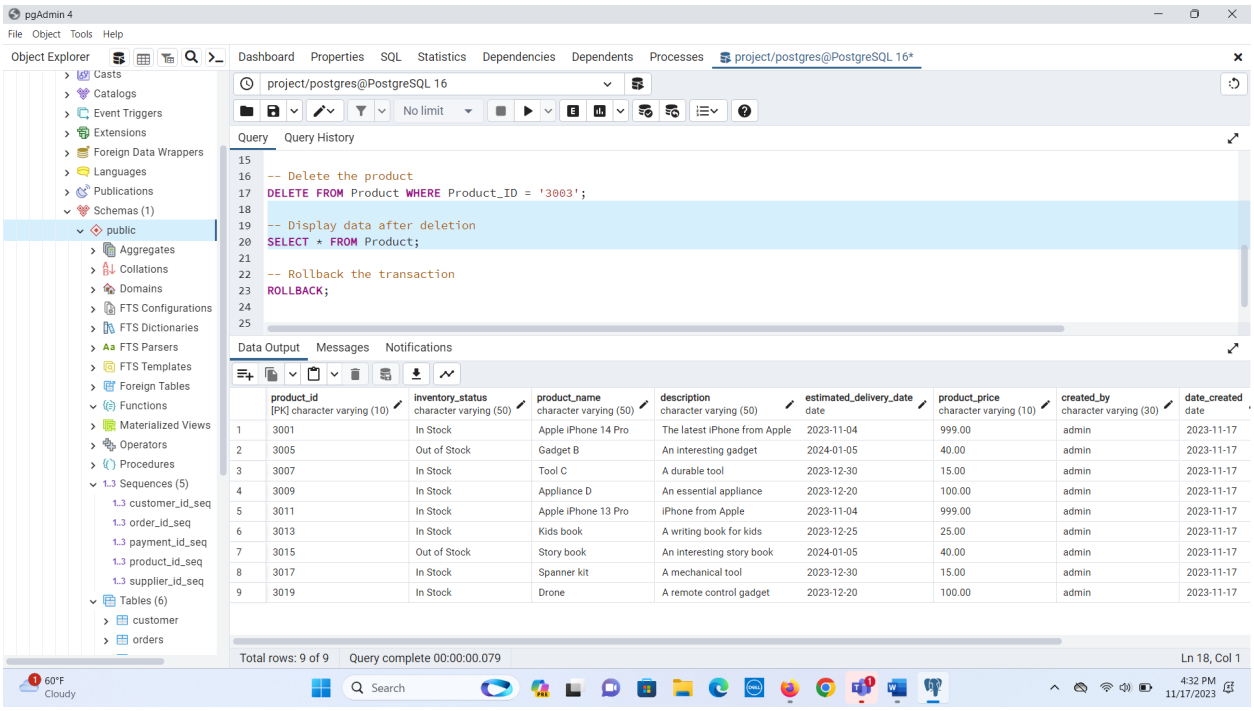
DELETE FROM Orders_Product WHERE Product_ID = '3003';

-- Delete the product

DELETE FROM Product WHERE Product_ID = '3003';

-- Display data after deletion

SELECT * FROM Product;

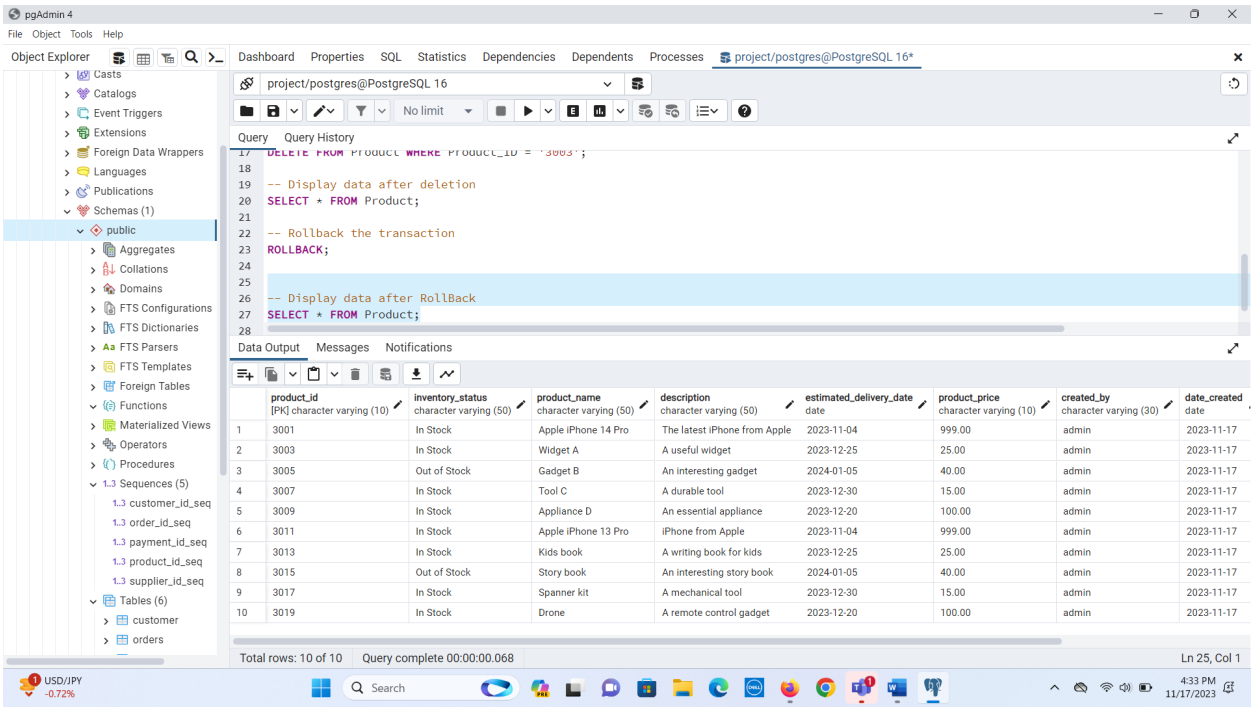


-- Rollback the transaction

ROLLBACK;

-- Display data after RollBack

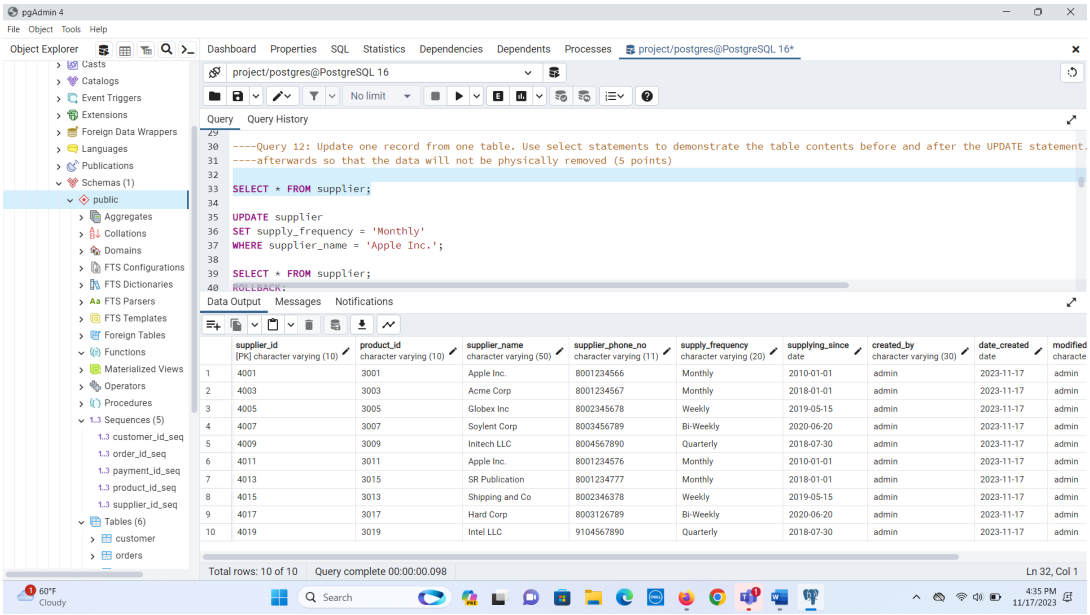
SELECT * FROM Product;



----**Query 12:** Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK

----afterwards so that the data will not be physically removed (5 points)

SELECT * FROM supplier;

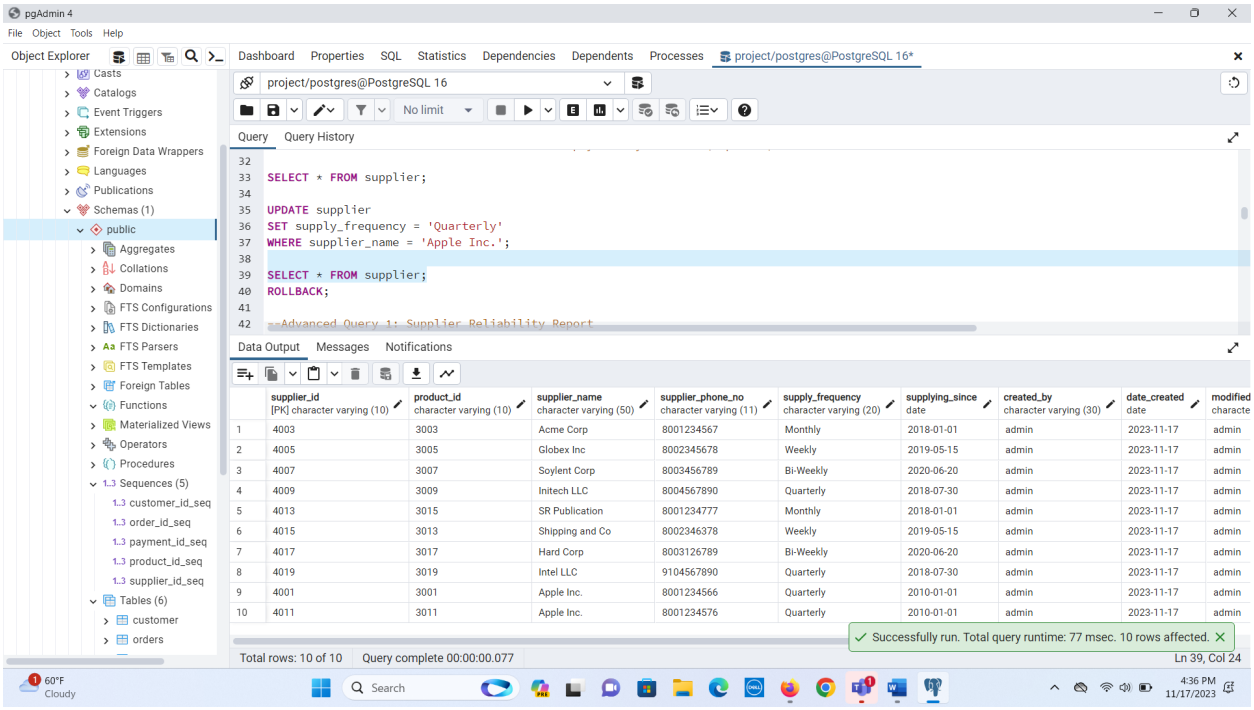


```
UPDATE supplier

SET supply_frequency = 'Quarterly'

WHERE supplier_name = 'Apple Inc.';

SELECT * FROM supplier;
```

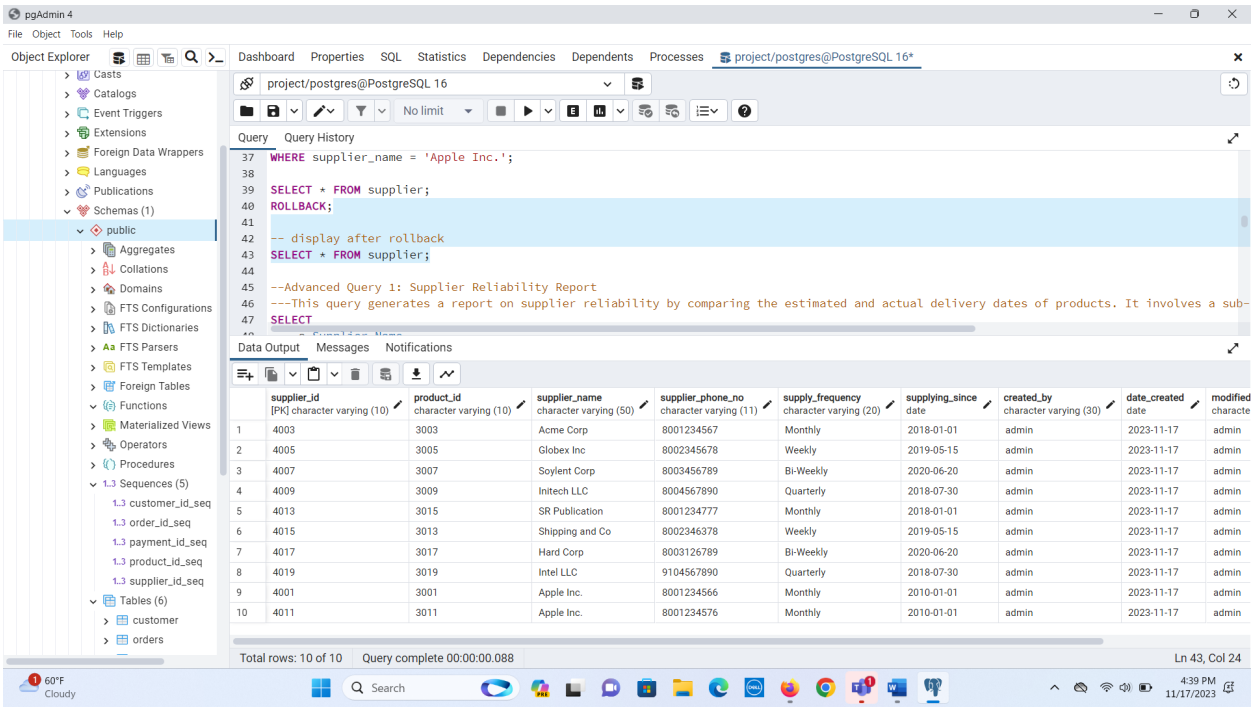


	supplier_id [PK] character varying (10)	product_id character varying (10)	supplier_name character varying (50)	supplier_phone_no character varying (11)	supply_frequency character varying (20)	supplying_since date	created_by character varying (30)	date_created date	modified
1	4003	3003	Acme Corp	8001234567	Monthly	2018-01-01	admin	2023-11-17	admin
2	4005	3005	Globex Inc	8002345678	Weekly	2019-05-15	admin	2023-11-17	admin
3	4007	3007	Soylent Corp	8003456789	Bi-Weekly	2020-06-20	admin	2023-11-17	admin
4	4009	3009	Initech LLC	8004567890	Quarterly	2018-07-30	admin	2023-11-17	admin
5	4013	3015	SR Publication	8001234777	Monthly	2018-01-01	admin	2023-11-17	admin
6	4015	3013	Shipping and Co	8002346378	Weekly	2019-05-15	admin	2023-11-17	admin
7	4017	3017	Hard Corp	8003126789	Bi-Weekly	2020-06-20	admin	2023-11-17	admin
8	4019	3019	Intel LLC	9104567890	Quarterly	2018-07-30	admin	2023-11-17	admin
9	4001	3001	Apple Inc.	8001234566	Quarterly	2010-01-01	admin	2023-11-17	admin
10	4011	3011	Apple Inc.	8001234576	Quarterly	2010-01-01	admin	2023-11-17	admin

```
ROLLBACK;

-- display after rollback

SELECT * FROM supplier;
```



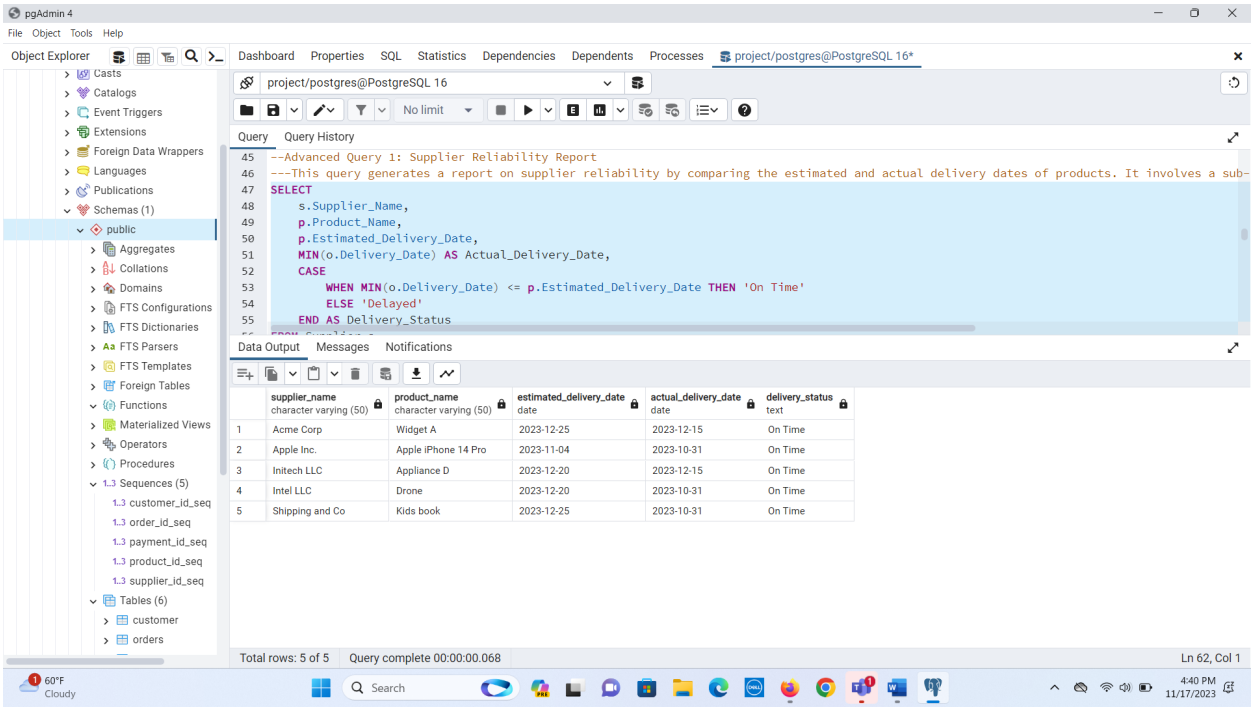
--**Advanced Query 1**: Supplier Reliability Report

---This query generates a report on supplier reliability by comparing the estimated and actual delivery dates of products. It involves a sub-query to get the earliest order date per product.

```
SELECT
    s.Supplier_Name,
    p.Product_Name,
    p.Estimated_Delivery_Date,
    MIN(o.Delivery_Date) AS Actual_Delivery_Date,
    CASE
        WHEN MIN(o.Delivery_Date) <= p.Estimated_Delivery_Date THEN 'On Time'
        ELSE 'Delayed'
    END AS Delivery_Status
FROM Supplier s
JOIN Product p ON s.Product_ID = p.Product_ID
JOIN Orders_Product op ON p.Product_ID = op.Product_ID
JOIN Orders o ON op.Order_ID = o.Order_ID
```

GROUP BY s.Supplier_Name, p.Product_Name, p.Estimated_Delivery_Date

ORDER BY s.Supplier_Name, Delivery_Status;



--**Advanced Query 2**: Customer Spending Analysis by Payment Method

SELECT

c.cust_Fname || ' ' || c.cust_LName AS Customer_Name,

p.Payment_Method,

COUNT(*) AS Number_of_Orders,

SUM(p.Payment_Amount) AS Total_Spent,

CASE

WHEN SUM(p.Payment_Amount) > 500 THEN 'High Spender'

WHEN SUM(p.Payment_Amount) BETWEEN 200 AND 500 THEN 'Medium Spender'

ELSE 'Low Spender'

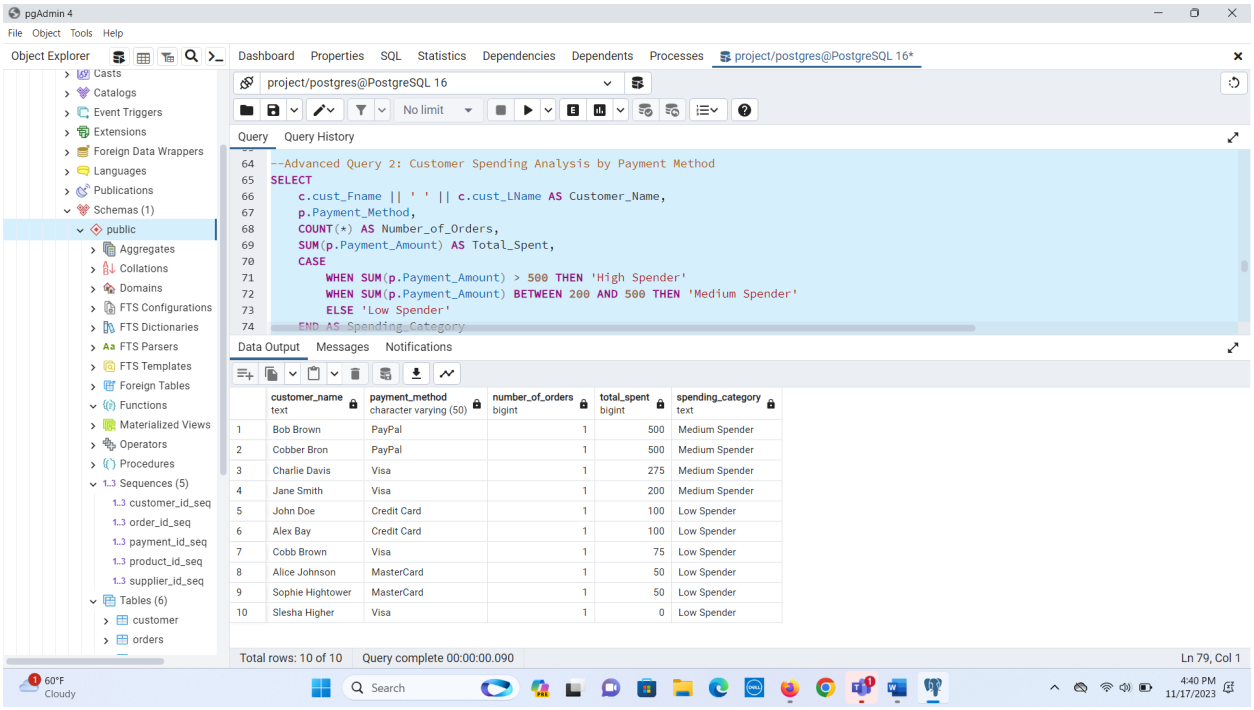
END AS Spending_Category

FROM Payment p

JOIN Customer c ON p.Customer_ID = c.Customer_ID

GROUP BY c.cust_Fname, c.cust_LName, p.Payment_Method

ORDER BY Total_Spent DESC;



--**Advanced Query3** with Subquery: Identifying High Spending Customers

SELECT

c.cust_Fname || ' ' || c.cust_LName AS Customer_Name,

COUNT(p.Payment_ID) AS Number_of_Payments,

SUM(p.Payment_Amount::numeric) AS Total_Spending

FROM Customer c

JOIN Payment p ON c.Customer_ID = p.Customer_ID

GROUP BY c.Customer_ID, c.cust_Fname, c.cust_LName

HAVING SUM(p.Payment_Amount::numeric) > (

SELECT AVG(Total_Amount::numeric)

FROM (

SELECT

```
        Customer_ID,
        SUM(Payment_Amount::numeric) AS Total_Amount
    FROM Payment
    GROUP BY Customer_ID
  ) AS SubQuery
)
ORDER BY Total_Spending DESC;
```