**Sales Order Case Study**

# Total available Tables

**SELECT \* FROM products;**

**SELECT \* FROM customers;**

**SELECT \* FROM employees;**

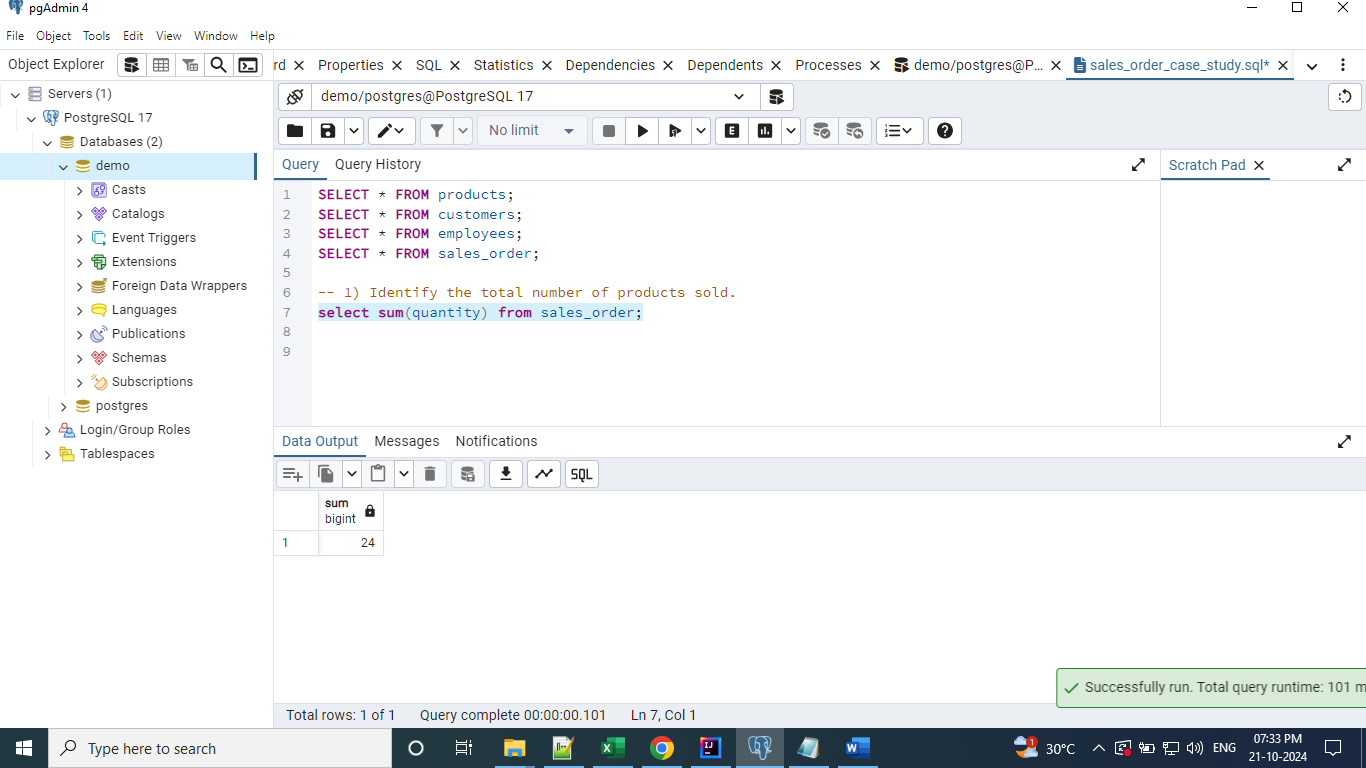
**SELECT \* FROM sales\_order;**

# Problems

## Identify the total number of products sold.

**We can get the total number of products details from the Products table. But the twist here is we need to get the total number of Products sold. The Sales\_Order is the only table which has the quantity of all products sold.**

**select sum(quantity) from sales\_order;**

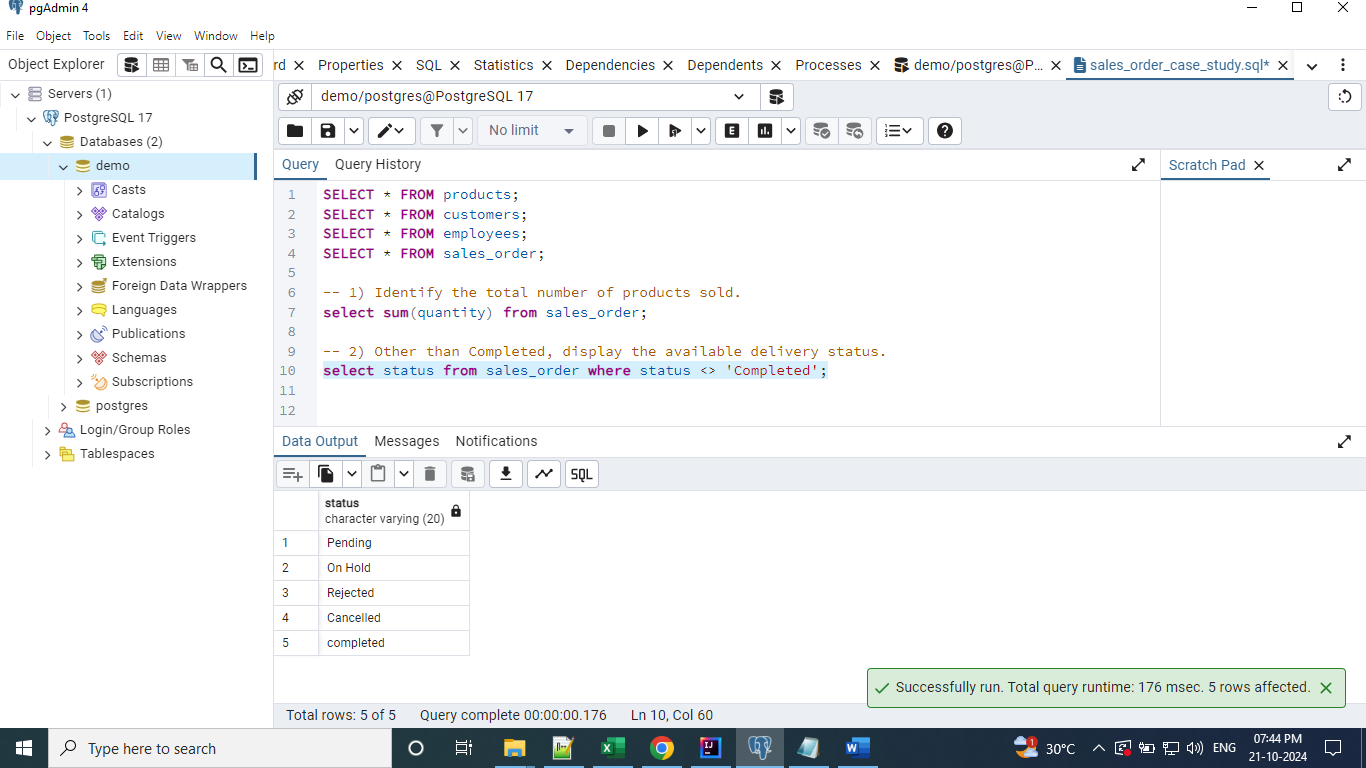


## Other than completed, display the available delivery status.

**We can get the details of order status from the Sales\_Order table.**

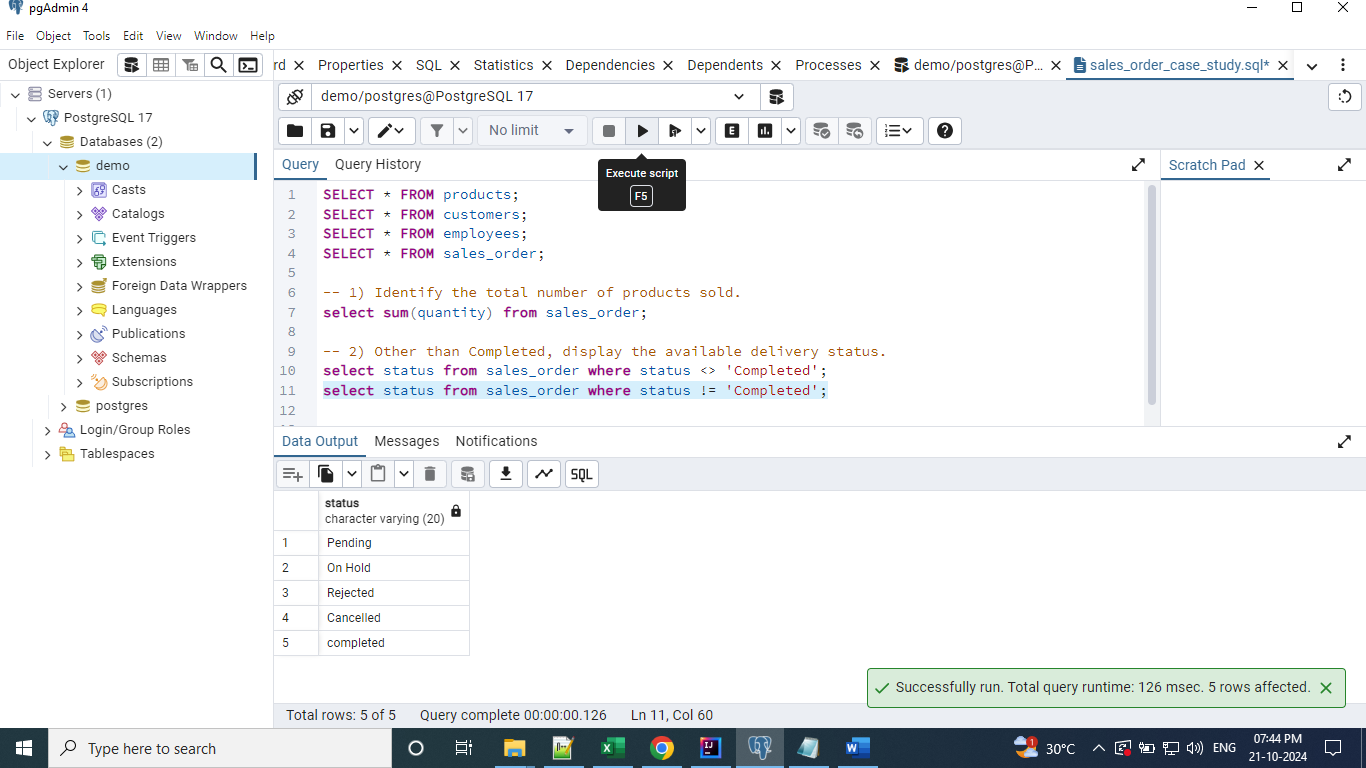
**# Approach 1**

**select status from sales\_order where status <> 'Completed';**



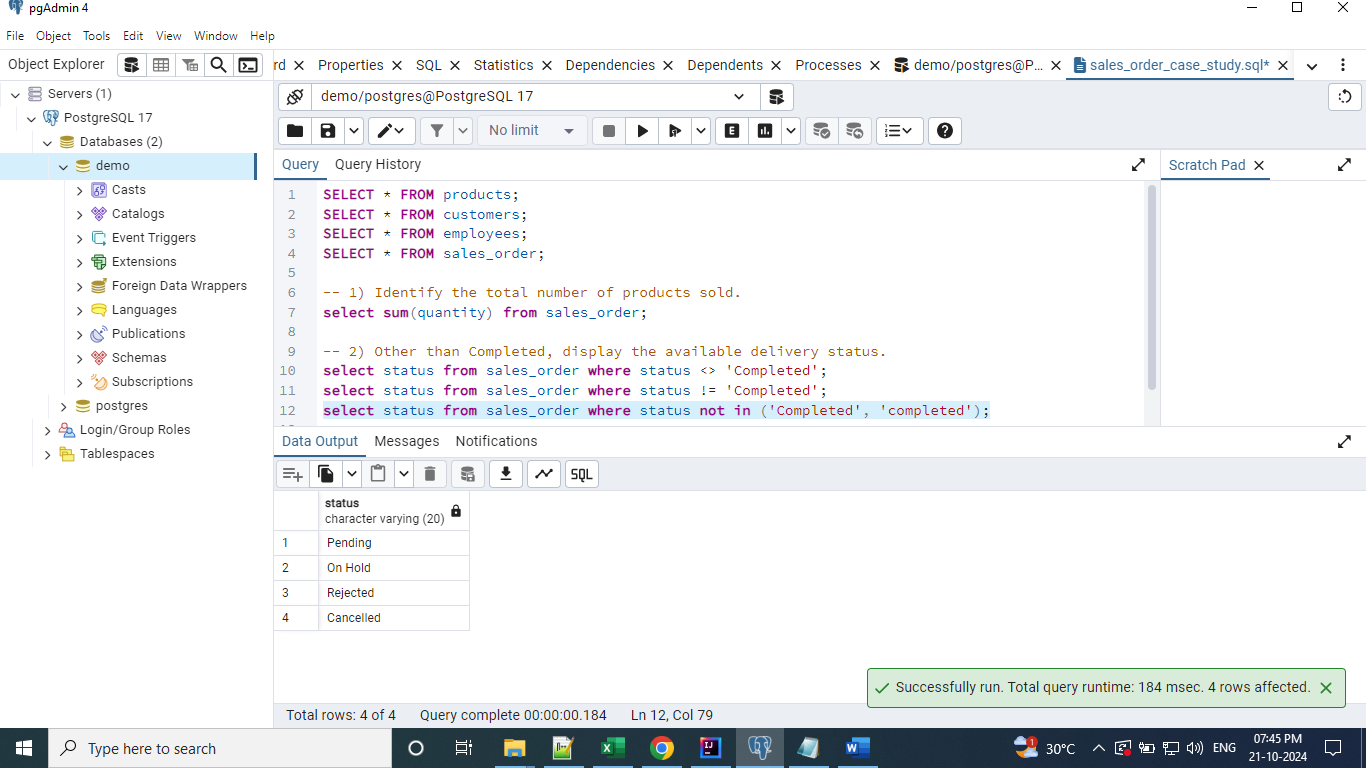
**# Approach 2**

**select status from sales\_order where status != 'Completed';**



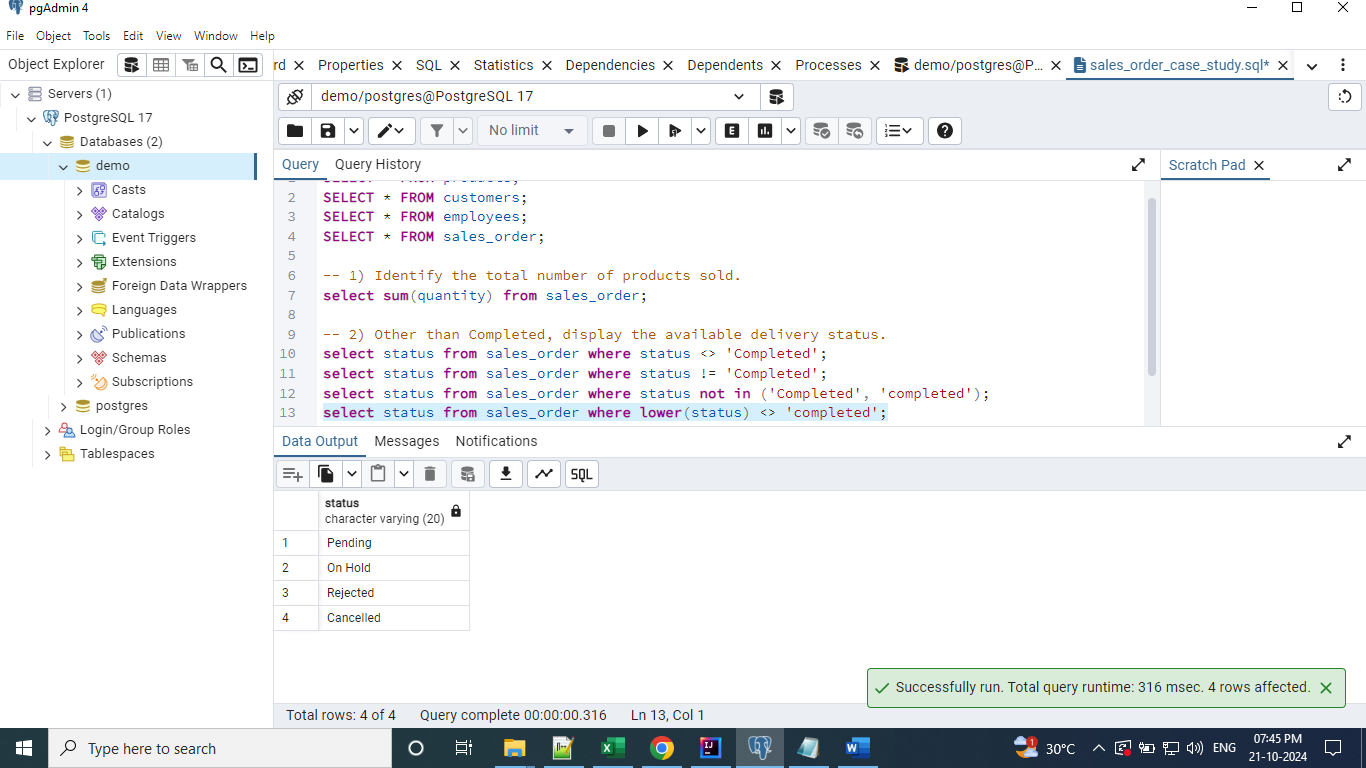
**# Approach 3**

**select status from sales\_order where status not in ('Completed', 'completed');**



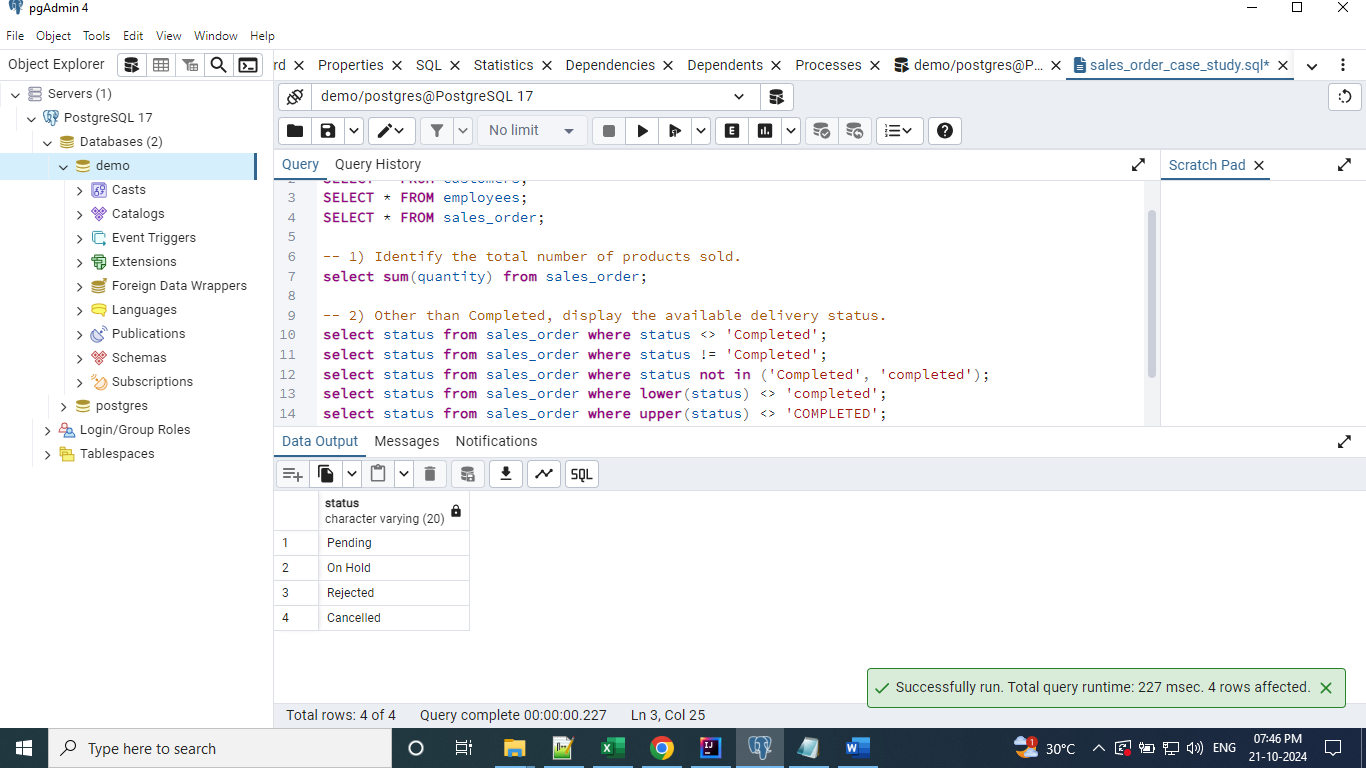
**# Approach 4**

**select status from sales\_order where lower(status) <> 'completed';**



**# Approach 5**

**select status from sales\_order where upper(status) <> 'COMPLETED';**



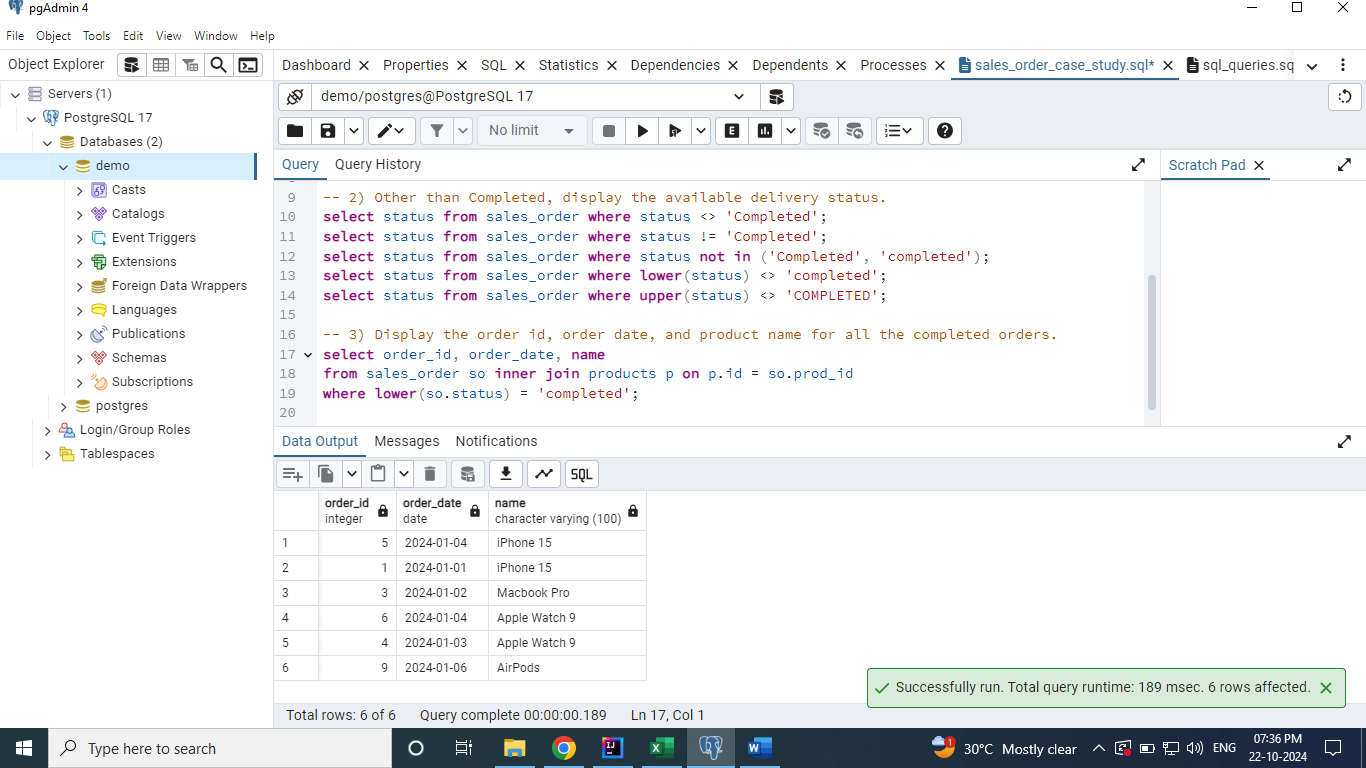
## Display the order id, order date, and product name for all the completed orders.

**Here, we have to fetch the column values from two tables, so need to use the join function. Inner Join is used here to fetch the records.**

**select order\_id, order\_date, name**

**from sales\_order so inner join products p on p.id = so.prod\_id**

**where lower(so.status) = 'completed';**



## Sort the above query to show the earliest orders at the top. Also display the customer who purchased these orders.

**Here, we need to print the customer details too. So, we have joined the customer table along with previous join and sorted using the order by clause.**

**select order\_id, order\_date, p.name, c.name**

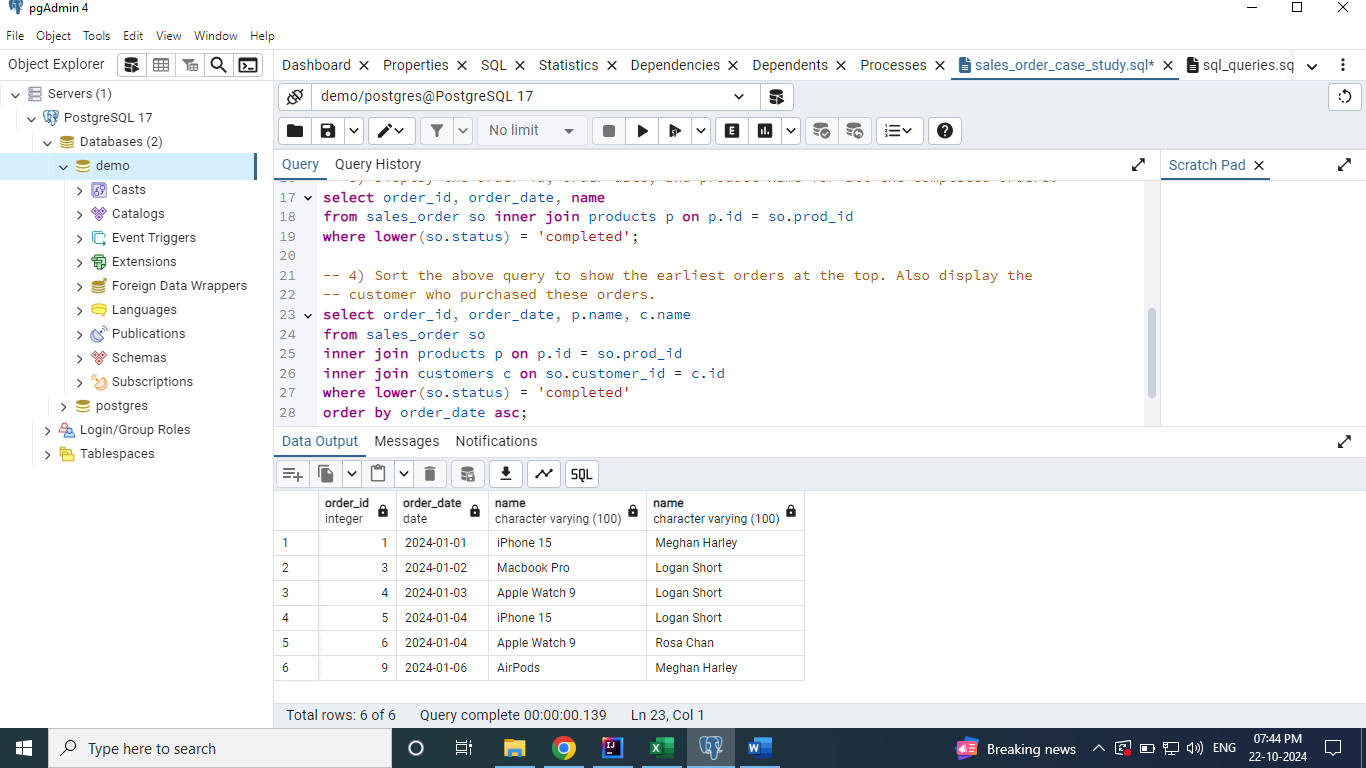
**from sales\_order so**

**inner join products p on p.id = so.prod\_id**

**inner join customers c on so.customer\_id = c.id**

**where lower(so.status) = 'completed'**

**order by order\_date asc;**



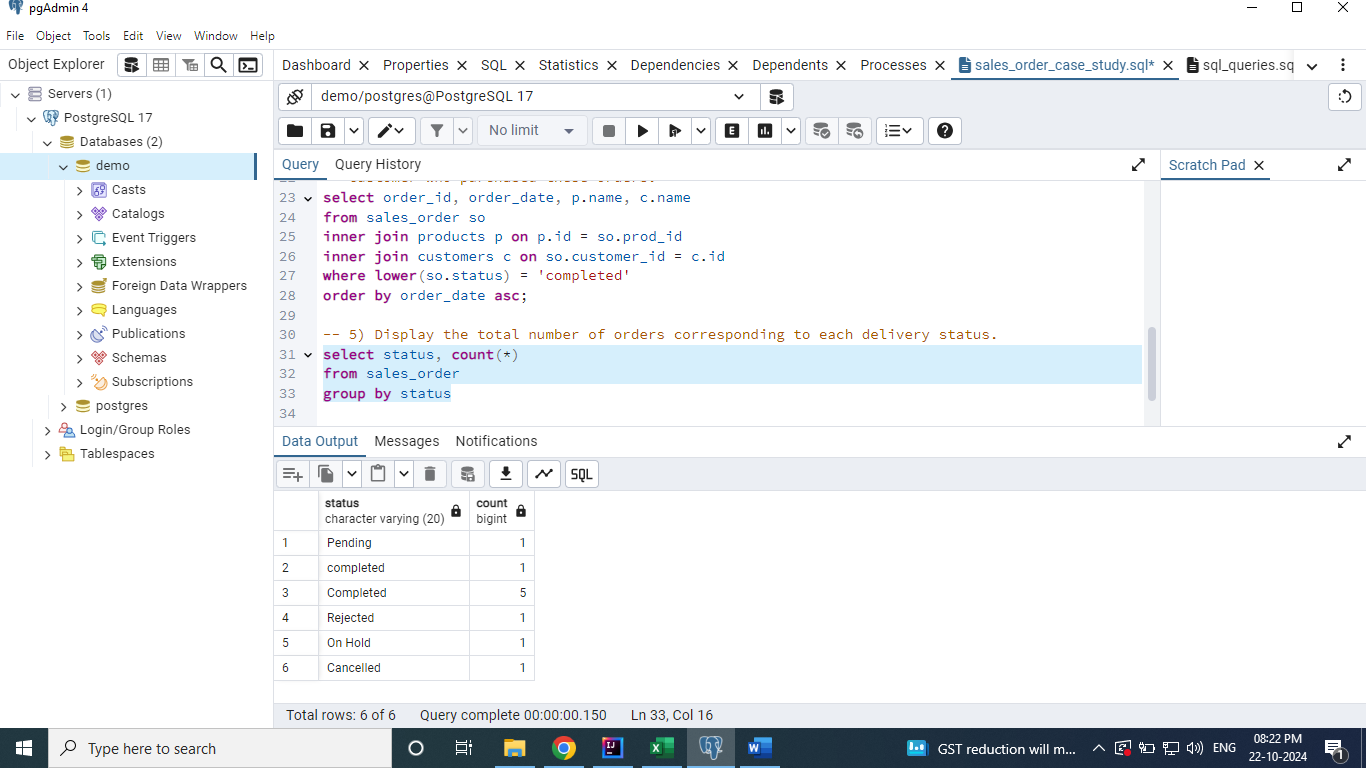
## Display the total number of orders corresponding to each delivery status.

**We can use the group by clause to group the orders based on the status.**

**select status, count(\*)**

**from sales\_order**

**group by status**



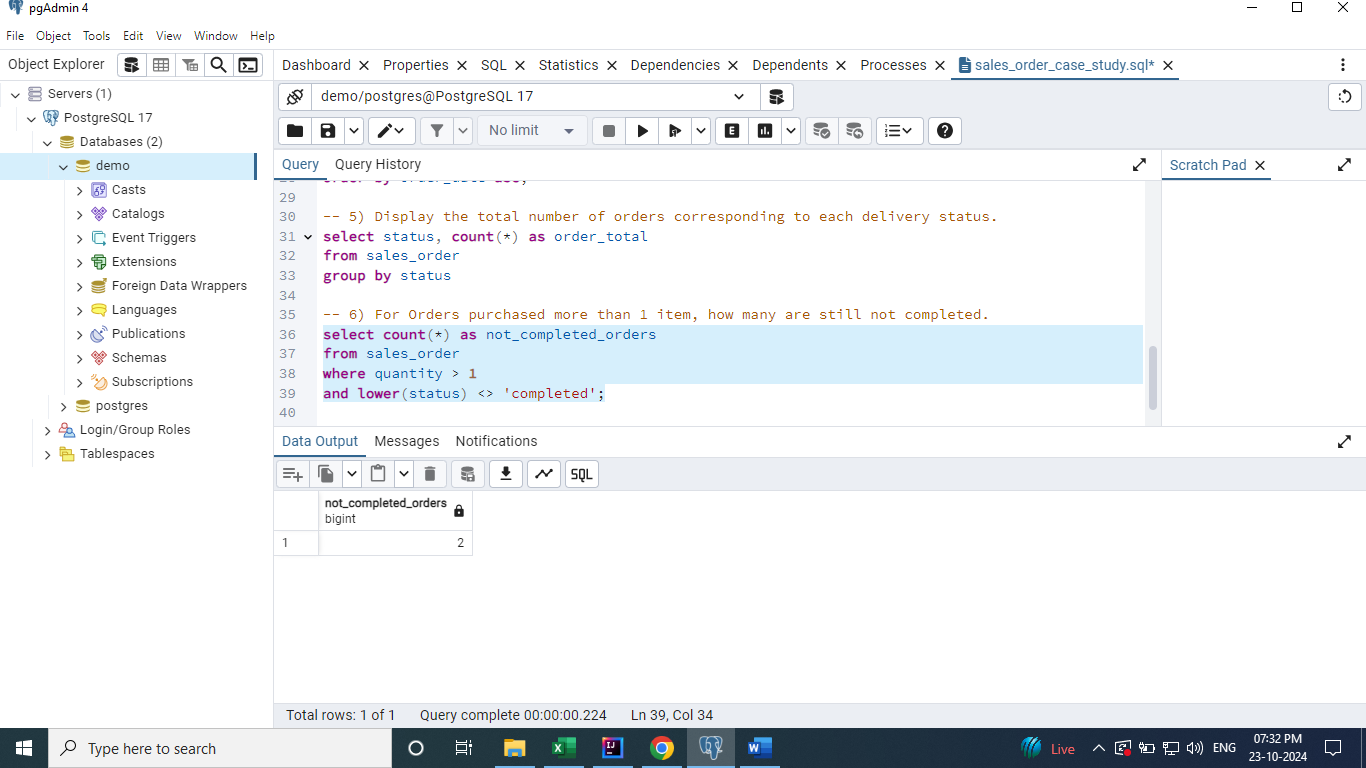
## For Orders purchased more than 1 item, how many are still not completed.

**select count(\*) as not\_completed\_orders**

**from sales\_order**

**where quantity > 1**

**and lower(status) <> 'completed';**

7. 

## 7. Find the total number of orders corresponding to each delivery status by ignoring the case in delivery status with highest number of orders should be at the top.

**Approach 1 [Sub Query]**

**select status,**

**case when status = 'completed'**

**then 'Completed'**

**else status**

**end as updated\_status from sales\_order;**

**select updated\_status, count(\*) as order\_total**

**from (select status,**

**case when status = 'completed'**

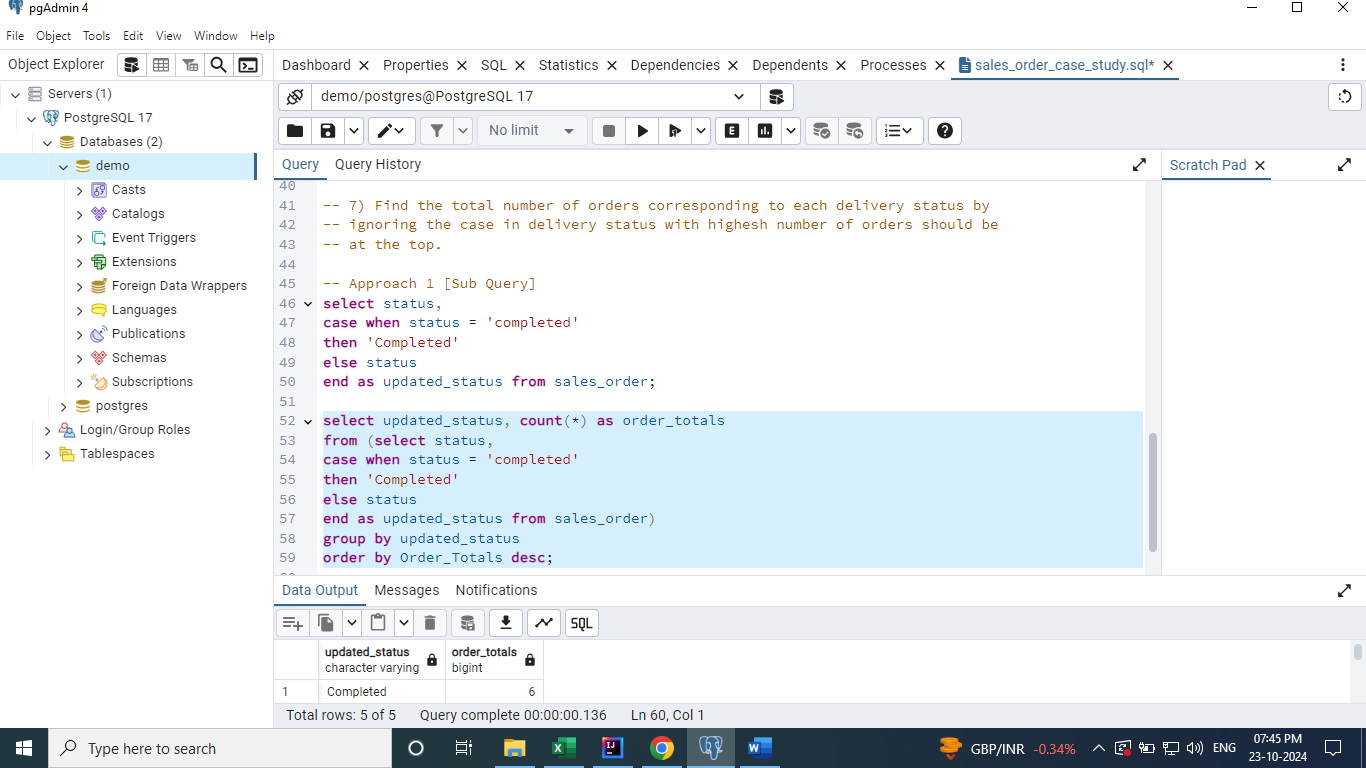
**then 'Completed'**

**else status**

**end as updated\_status from sales\_order)**

**group by updated\_status**

**order by order\_total desc;**



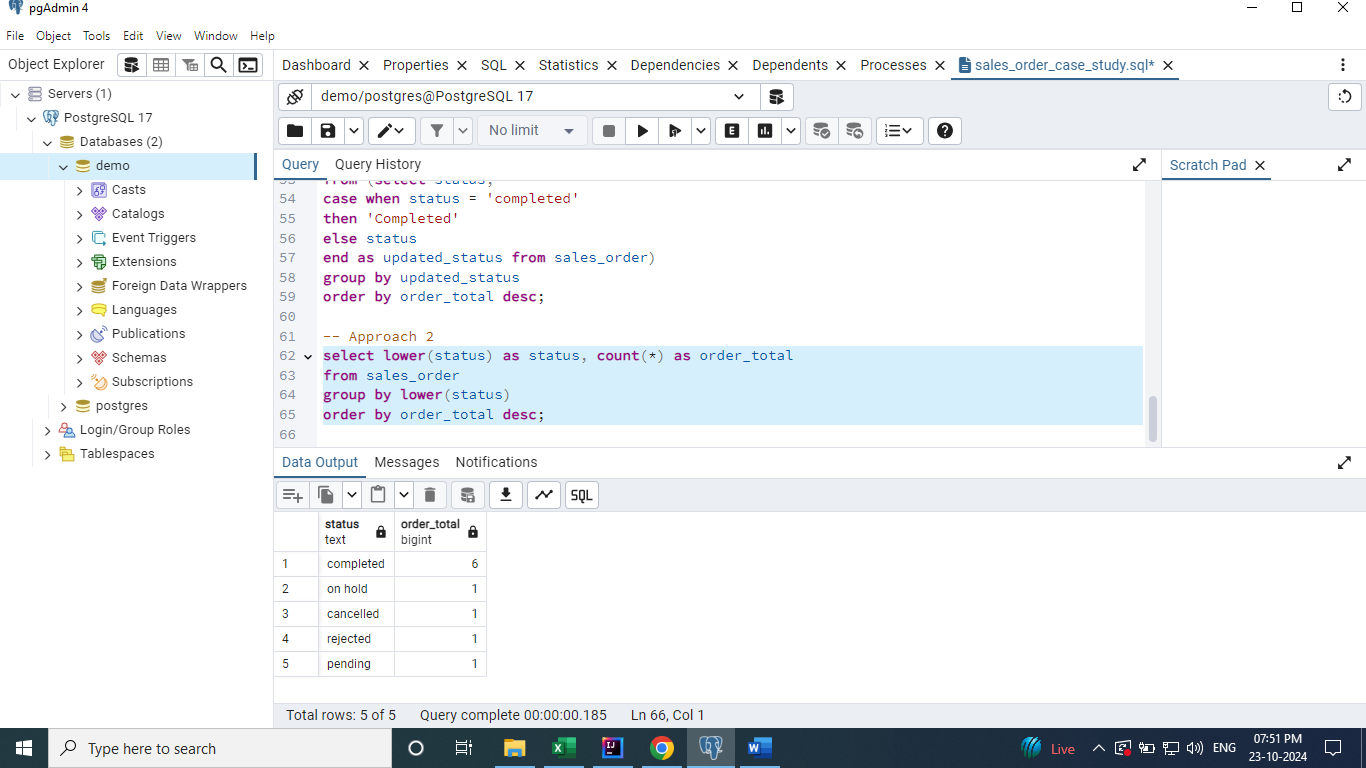
**Approach 2**

**select lower(status) as status, count(\*) as order\_total**

**from sales\_order**

**group by lower(status)**

**order by order\_total desc;**



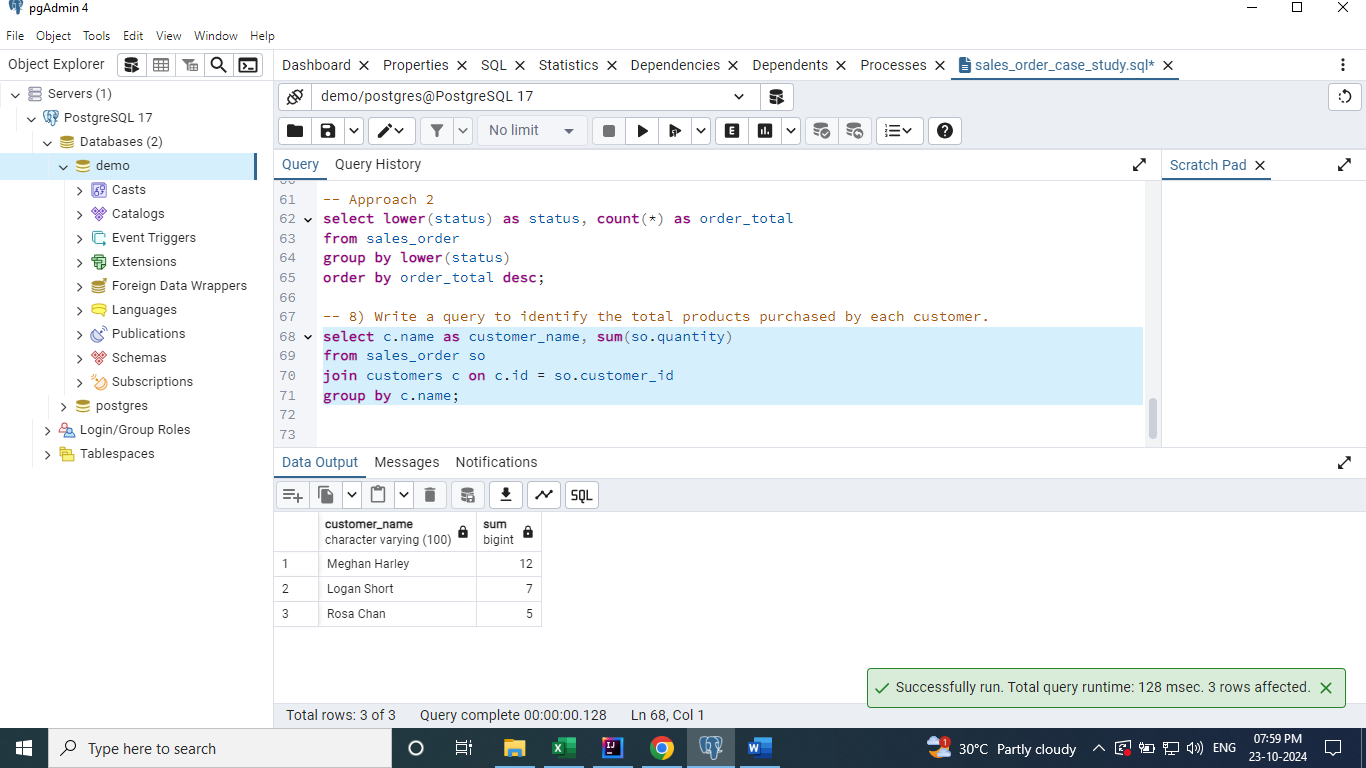
## Write a query to identify the total products purchased by each customer.

**select c.name as customer\_name, sum(so.quantity)**

**from sales\_order so**

**join customers c on c.id = so.customer\_id**

**group by c.name;**



## Display the total sales and average sales done for each day.

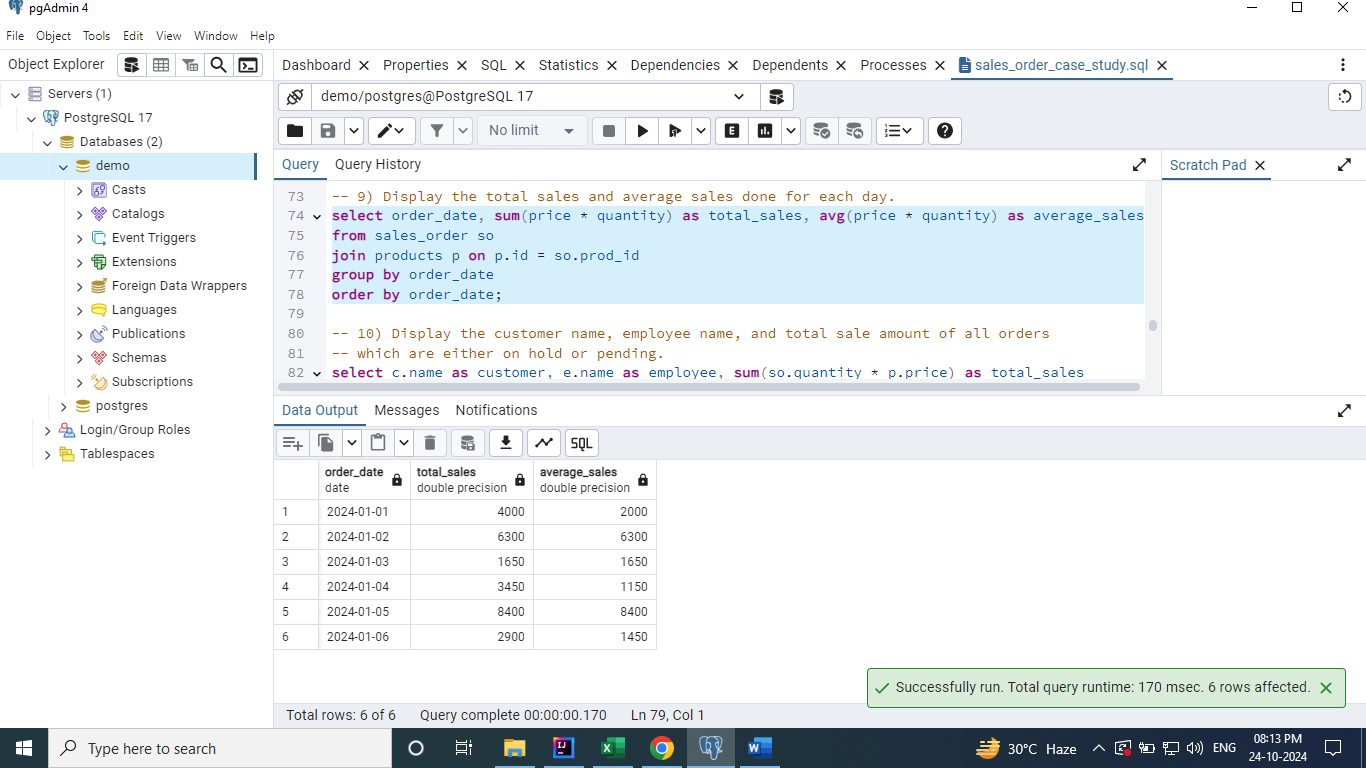
**select order\_date, sum(price \* quantity) as total\_sales, avg(price \* quantity) as average\_sales**

**from sales\_order so**

**join products p on p.id = so.prod\_id**

**group by order\_date**

**order by order\_date;**



## Display the customer name, employee name, and total sale amount of all orders which are either on hold or pending.

**select c.name as customer, e.name as employee, sum(so.quantity \* p.price) as total\_sales**

**from sales\_order so**

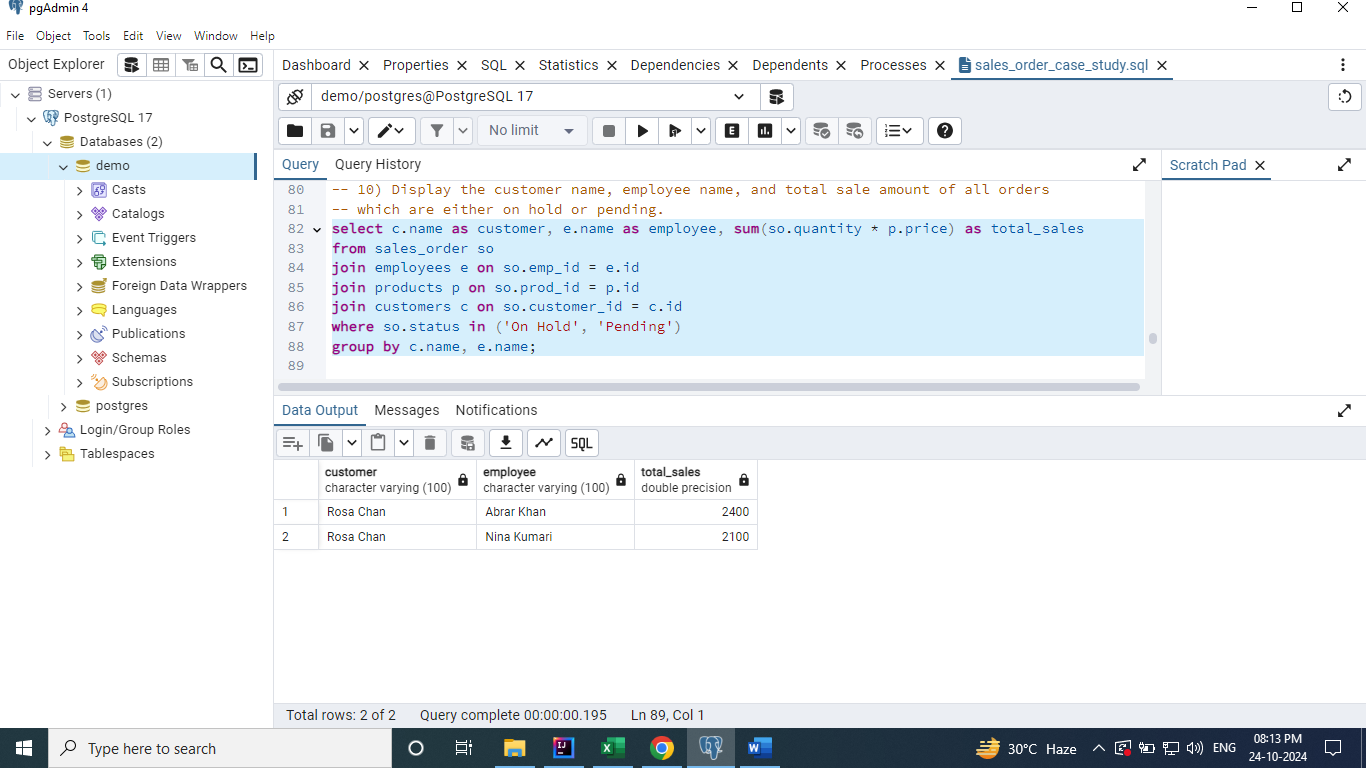
**join employees e on so.emp\_id = e.id**

**join products p on so.prod\_id = p.id**

**join customers c on so.customer\_id = c.id**

**where so.status in ('On Hold', 'Pending')**

**group by c.name, e.name;**



## Fetch all the orders which were neither completed / pending or were handled by the employee Abrar. Display employee name and all details of order.

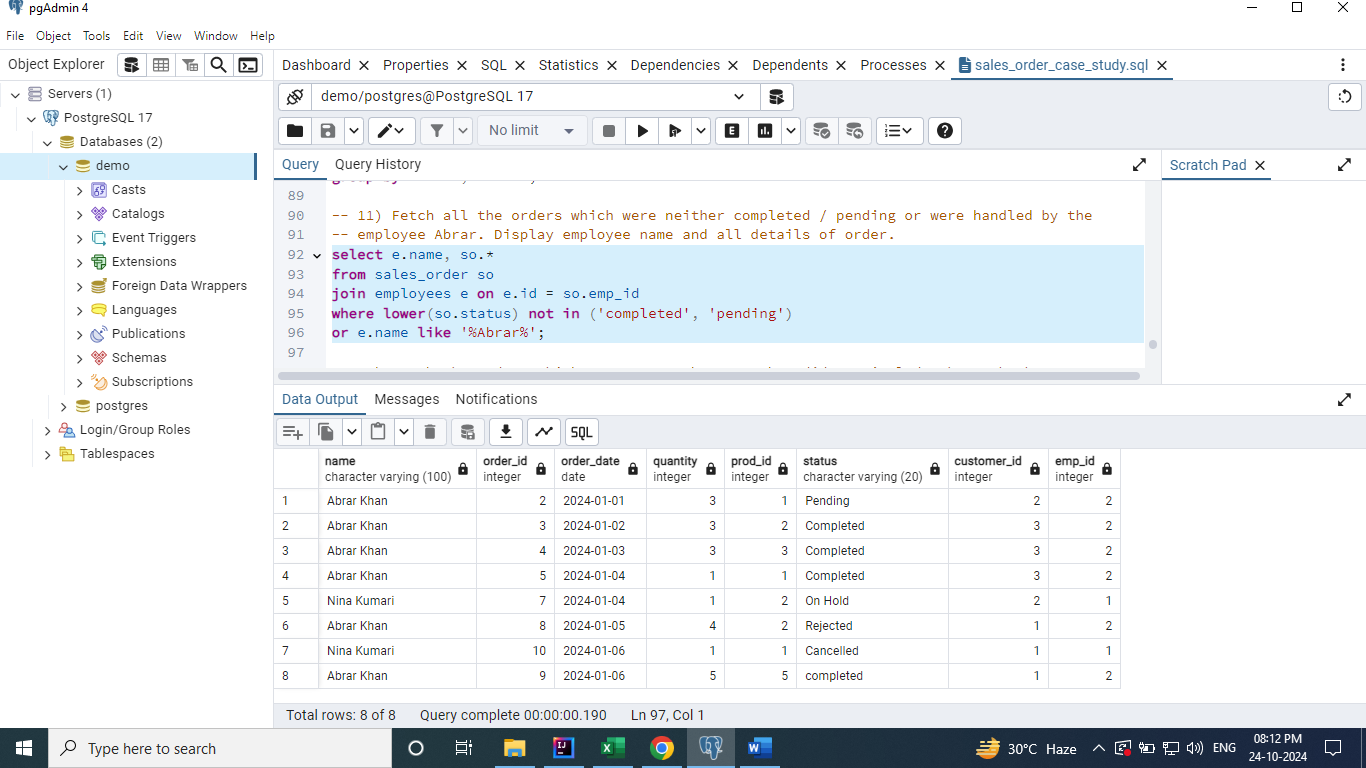
**select e.name, so.\***

**from sales\_order so**

**join employees e on e.id = so.emp\_id**

**where lower(so.status) not in ('completed', 'pending')**

**or e.name like '%Abrar%';**



## Fetch the orders which costs more than 2000 but did not include the Mac book pro. Print the total sale amount as well.

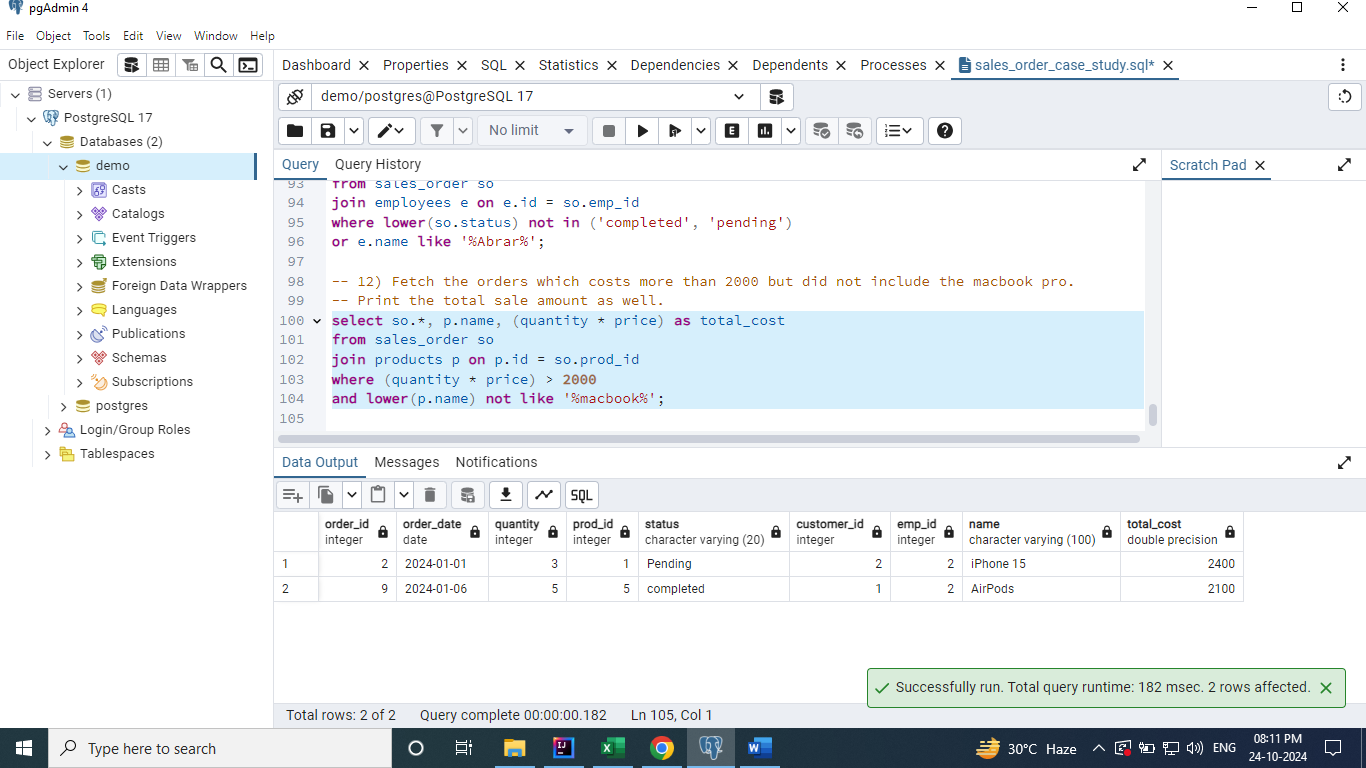
**select so.\*, p.name, (quantity \* price) as total\_cost**

**from sales\_order so**

**join products p on p.id = so.prod\_id**

**where (quantity \* price) > 2000**

**and lower(p.name) not like '%macbook%';**



## **Identify the customers who have not purchased any product yet.**

**Approach 1 [Sub Query]**

**select \***

**from customers**

**where id not in (select distinct customer\_id from sales\_order);**

**Approach 2 [Left Join]**

**select c.\***

**from customers c**

**left join sales\_order so**

**on c.id = so.customer\_id**

**where so.order\_id is null;**

**Approach 3 [Right Join]**

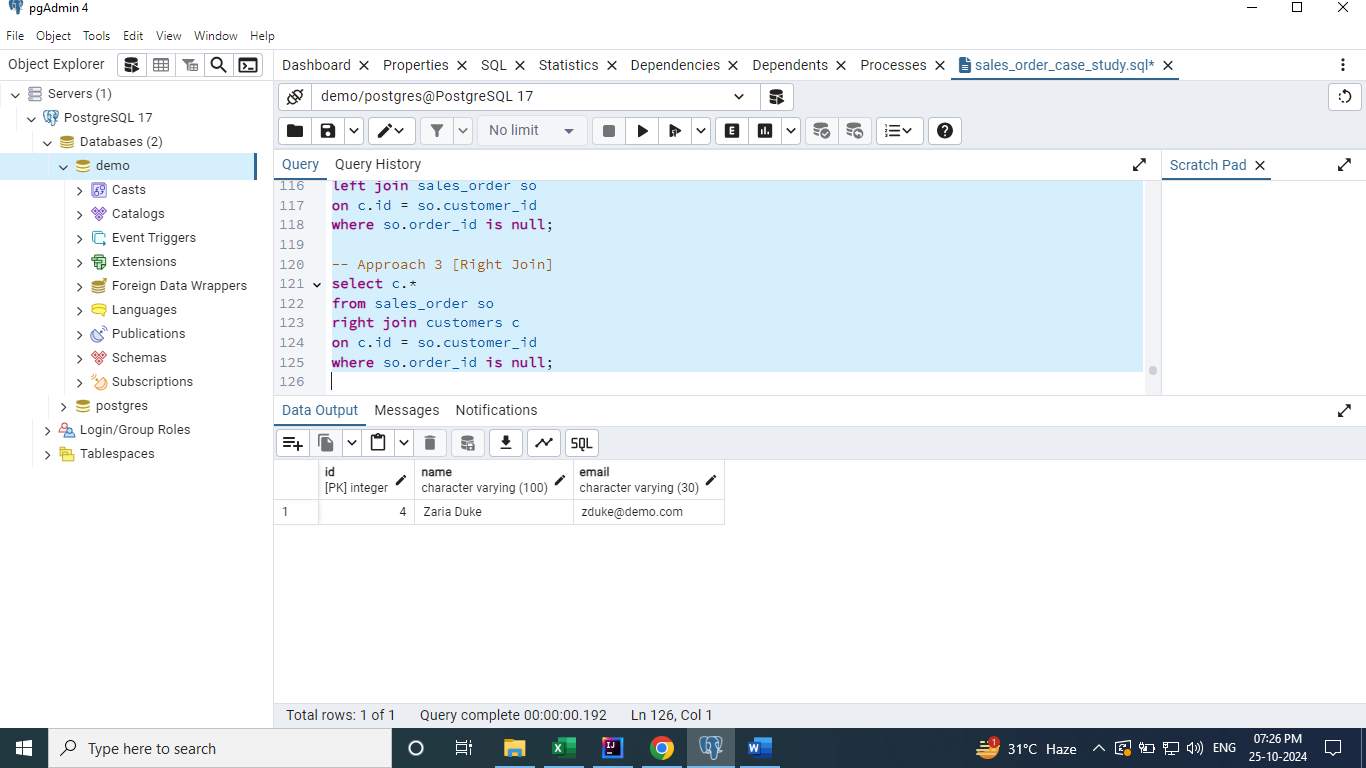
**select c.\***

**from sales\_order so**

**right join customers c**

**on c.id = so.customer\_id**

**where so.order\_id is null;**



## **Write a query to identify the total products purchased by each customer. Return all customers irrespective of whether they have made a purchase or not. Sort the result with highest number of orders at the top.**

**select c.name, coalesce(sum(so.quantity), 0) as total\_products\_purchased**

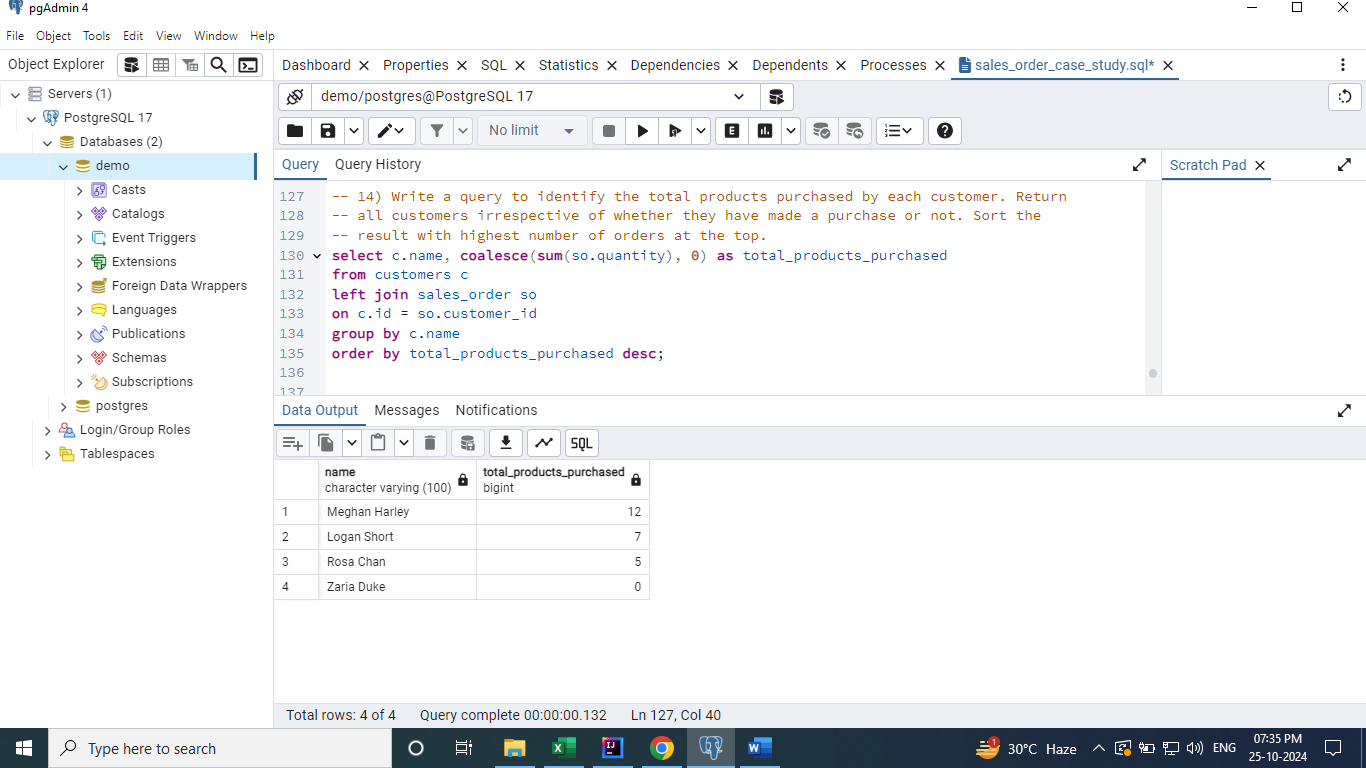
**from customers c**

**left join sales\_order so**

**on c.id = so.customer\_id**

**group by c.name**

**order by total\_products\_purchased desc;**



## **Corresponding to each employee, display the total sales they made of all the completed orders. Display the total sales as 0 if an employee made no sales yet.**

**select e.name as employee, coalesce(sum(quantity \* price),0) as total\_sales**

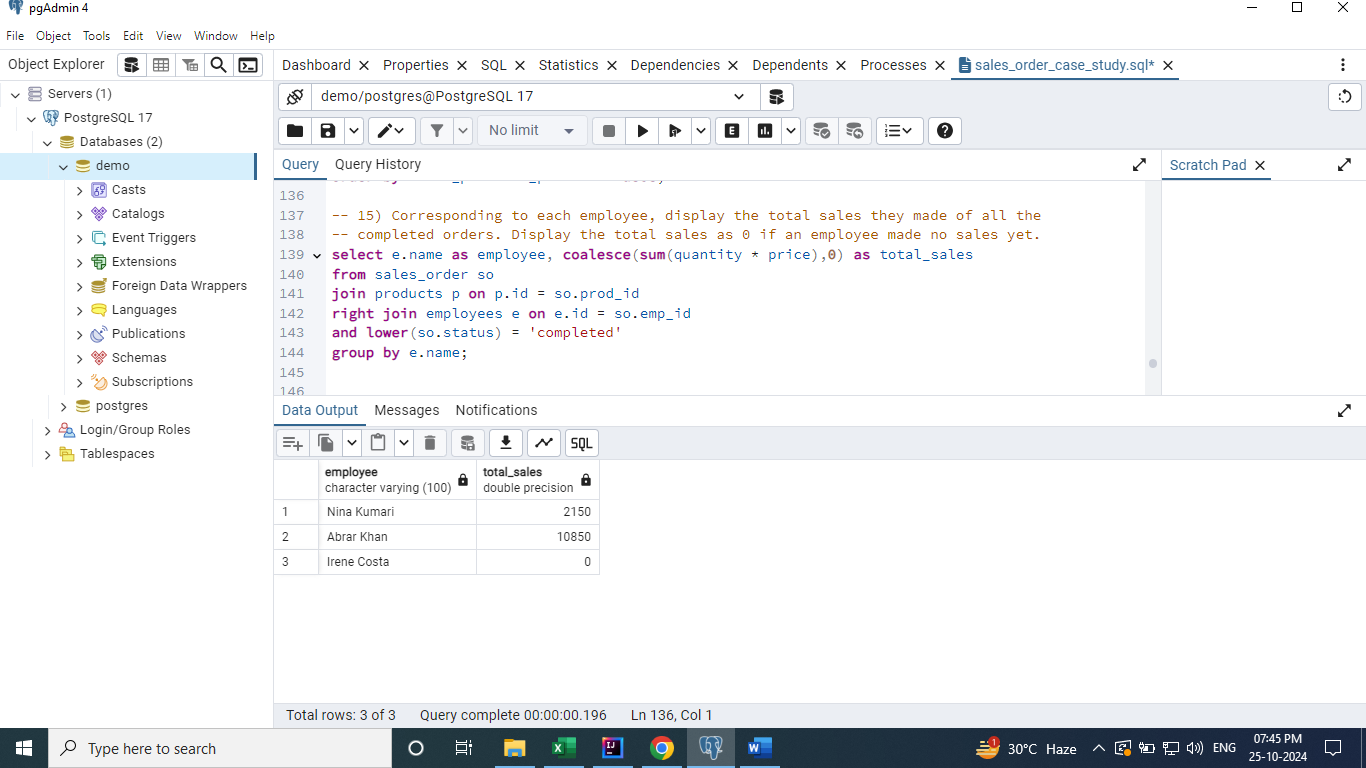
**from sales\_order so**

**join products p on p.id = so.prod\_id**

**right join employees e on e.id = so.emp\_id**

**and lower(so.status) = 'completed'**

**group by e.name;**



## **Rewrite the above query so as to display the total sales made by each employee corresponding to each customer. If an employee has not served a customer yet then display "-" under the customer.**

**select e.name as employee, coalesce(c.name,'-') as customer, coalesce(sum(quantity \* price),0) as total\_sales**

**from sales\_order so**

**join products p on p.id = so.prod\_id**

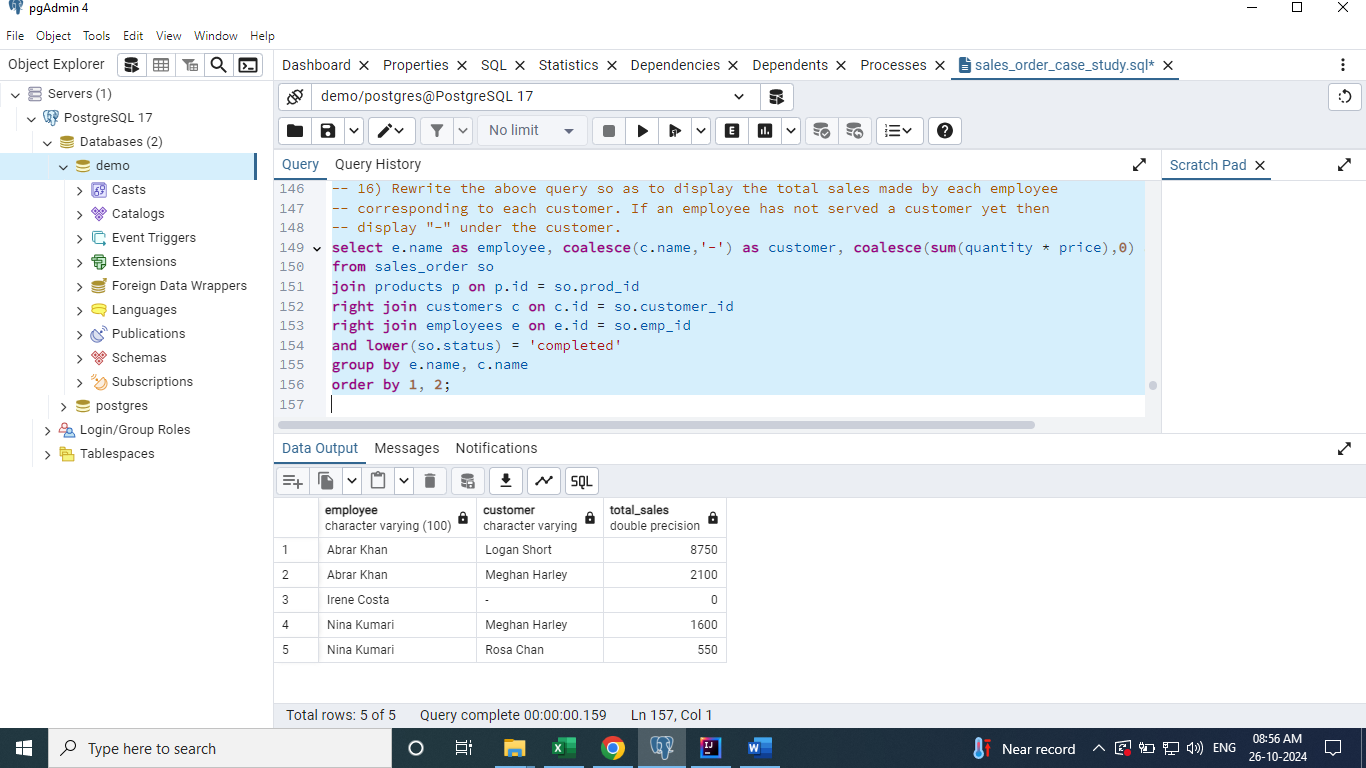
**right join customers c on c.id = so.customer\_id**

**right join employees e on e.id = so.emp\_id**

**and lower(so.status) = 'completed'**

**group by e.name, c.name**

**order by 1, 2;**



## **Rewrite the above query so as to display only those records where the total sales is above 1000.**

**select e.name as employee, coalesce(c.name,'-') as customer, coalesce(sum(quantity \* price),0) as total\_sales**

**from sales\_order so**

**join products p on p.id = so.prod\_id**

**right join customers c on c.id = so.customer\_id**

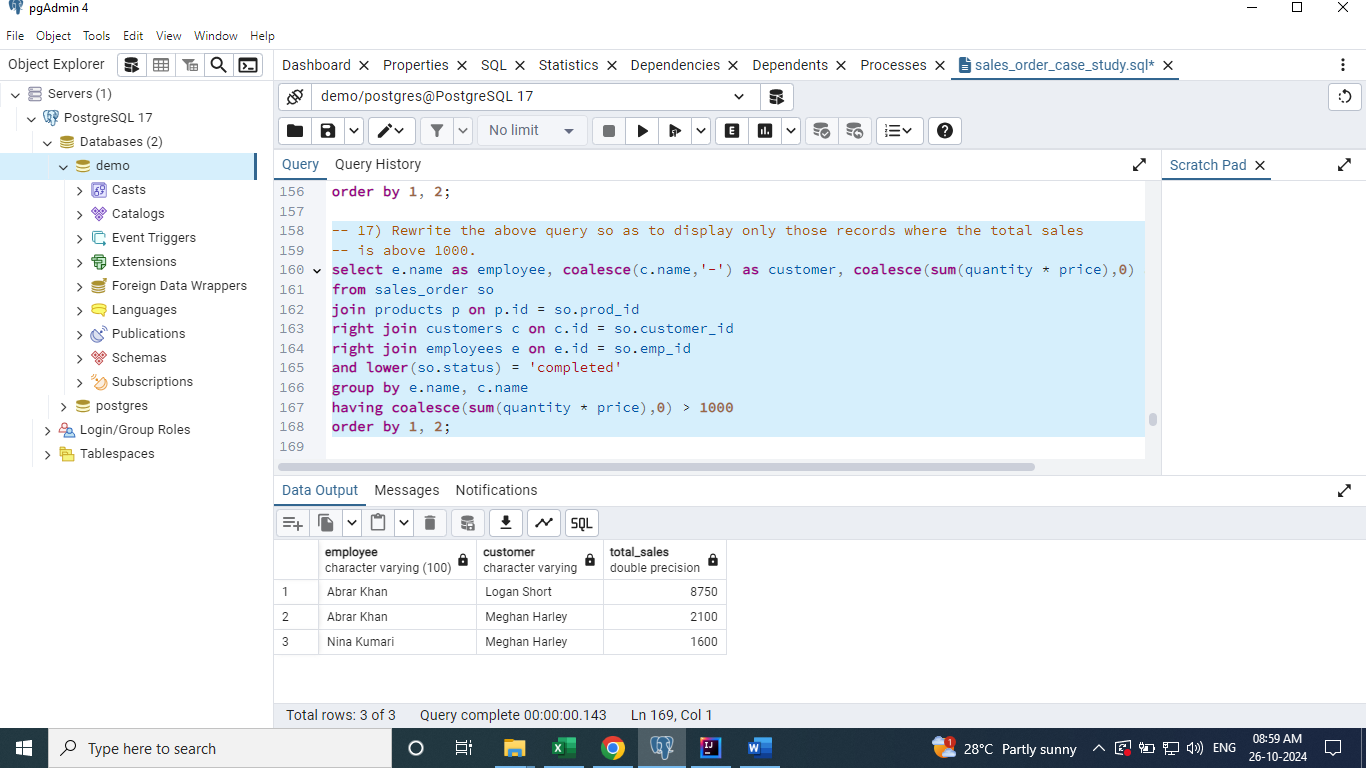
**right join employees e on e.id = so.emp\_id**

**and lower(so.status) = 'completed'**

**group by e.name, c.name**

**having coalesce(sum(quantity \* price),0) > 1000**

**order by 1, 2;**



## **Identify the employees who has served more than 2 customers.**

**select e.name as employee, count(distinct c.name) as customer**

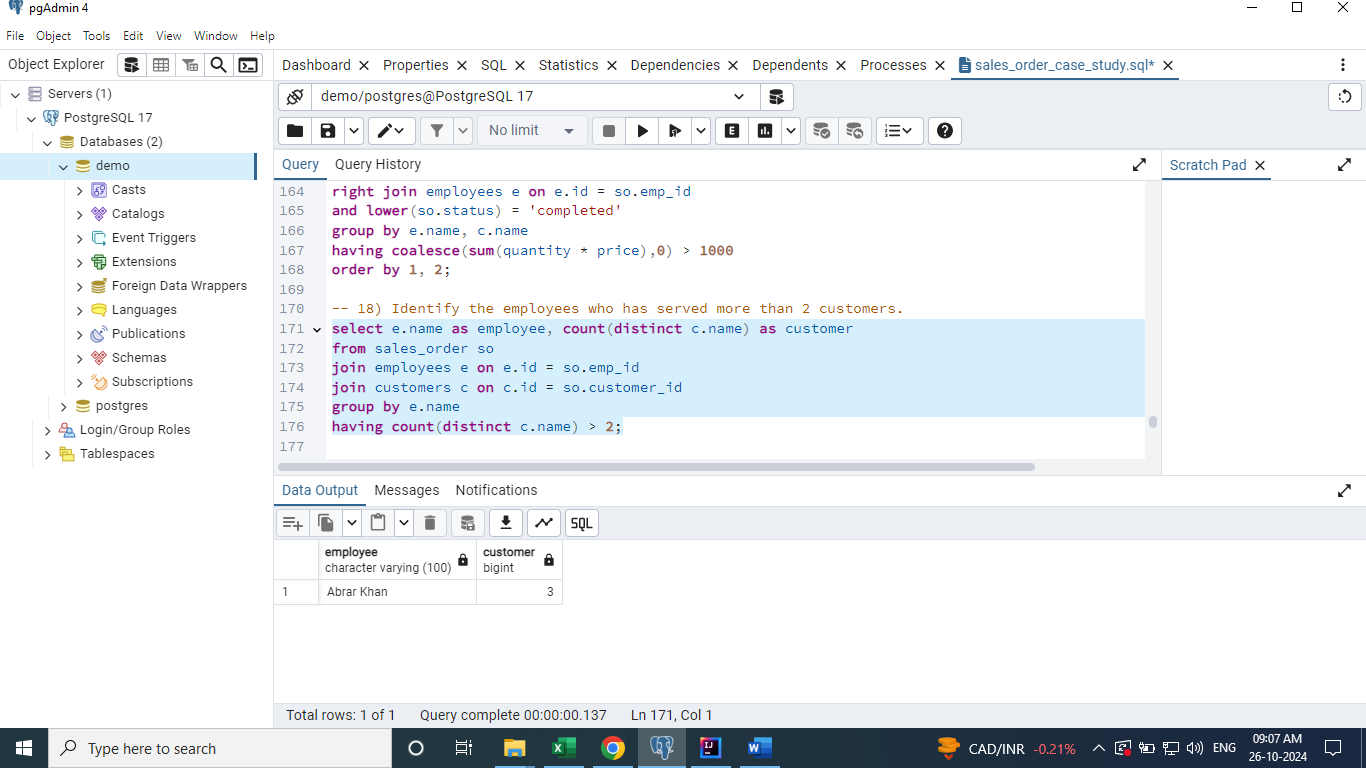
**from sales\_order so**

**join employees e on e.id = so.emp\_id**

**join customers c on c.id = so.customer\_id**

**group by e.name**

**having count(distinct c.name) > 2;**



## **Identify the customers who has purchased more than 5 products.**

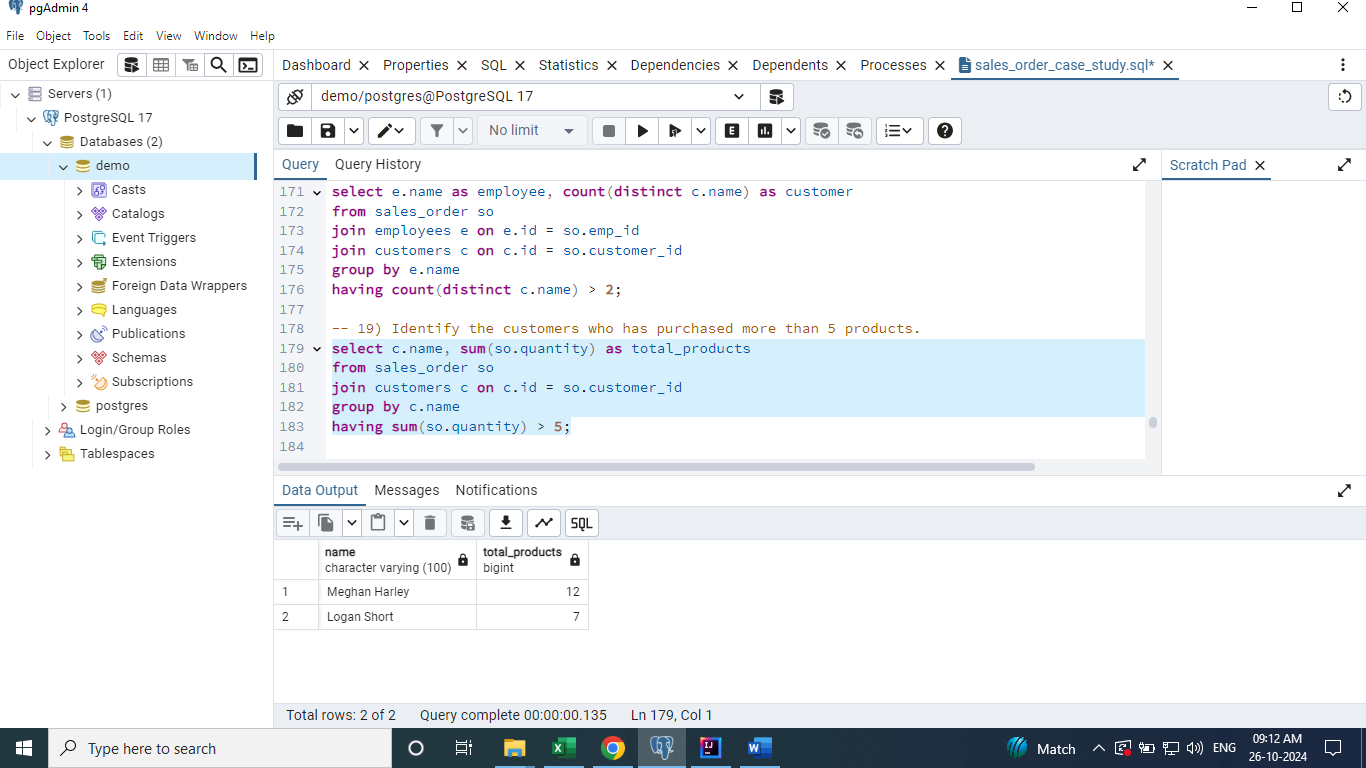
**select c.name, sum(so.quantity) as total\_products**

**from sales\_order so**

**join customers c on c.id = so.customer\_id**

**group by c.name**

**having sum(so.quantity) > 5;**



## **Identify customers whose average purchase cost exceeds the average sale of all the orders.**

**select c.name as customer, avg(quantity \* price) as average\_purchase**

**from sales\_order so**

**join customers c on c.id = so.customer\_id**

**join products p on p.id = so.prod\_id**

**group by c.name**

**having avg(quantity \* price) >**

**(select avg(quantity \* price)**

**from sales\_order so**

**join products p on p.id = so.prod\_id);**

