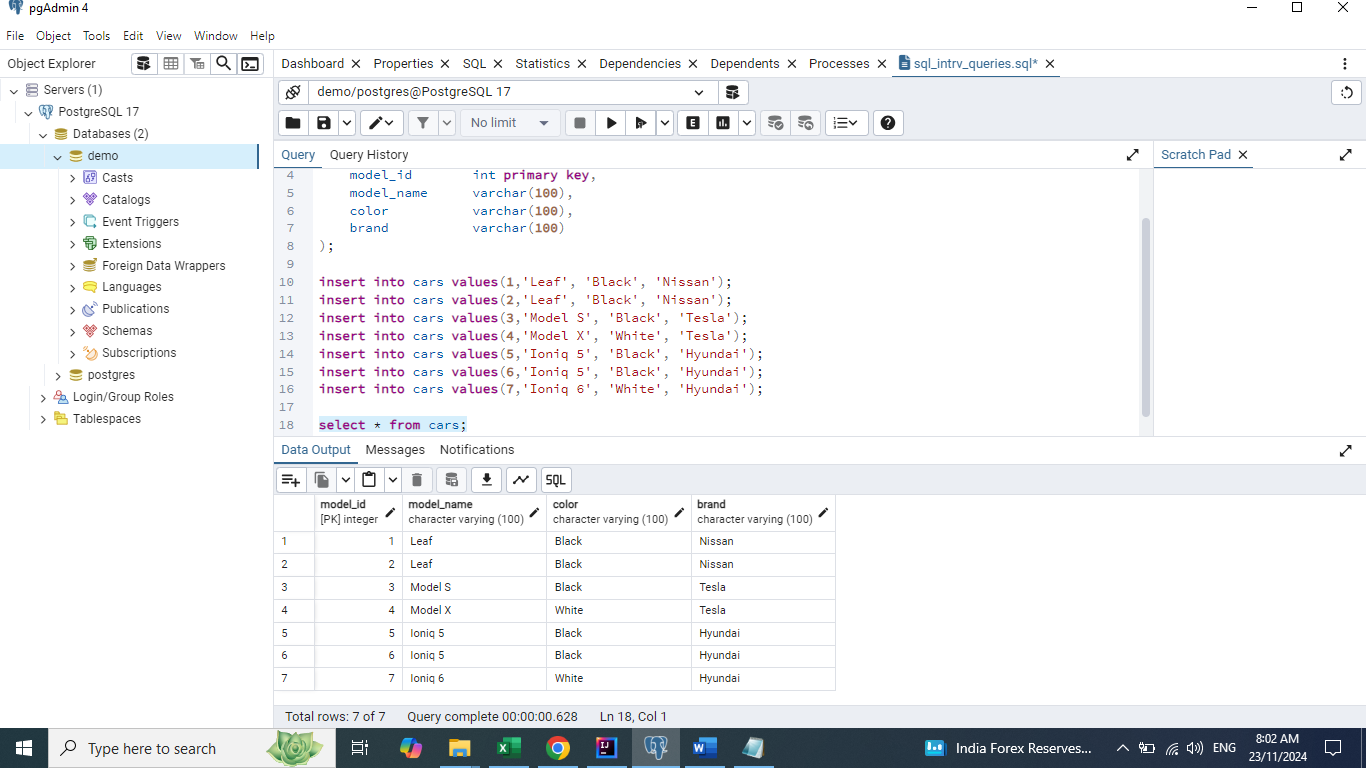
**SQL Interview Questions**

# Delete Duplicate Records



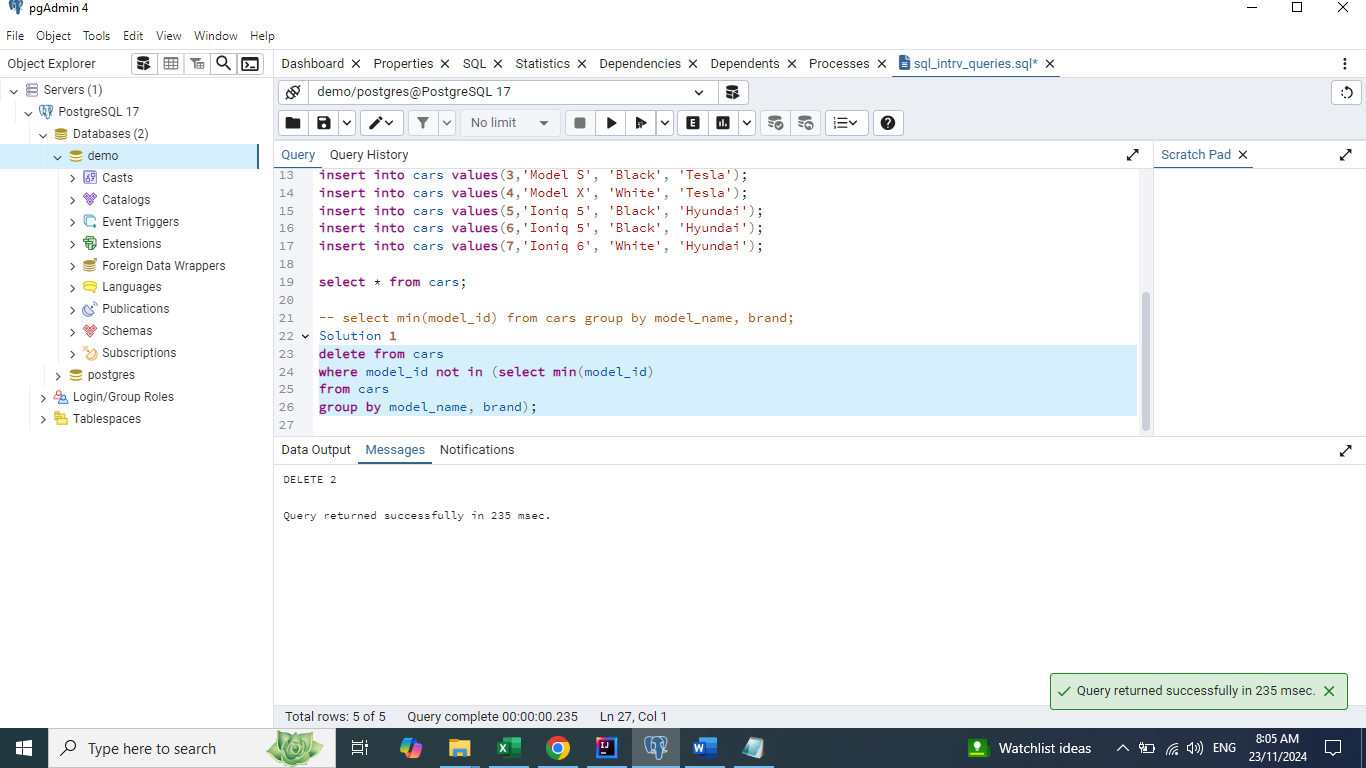
## Solution 1

**delete from cars**

**where model\_id not in (select min(model\_id)**

**from cars**

**group by model\_name, brand);**



## Solution 2

**Note:**

ctid - Kind of unique identifier which is present in PostgresSQL by default.

select max(ctid) from cars group by model\_name, brand having count(1) > 1;

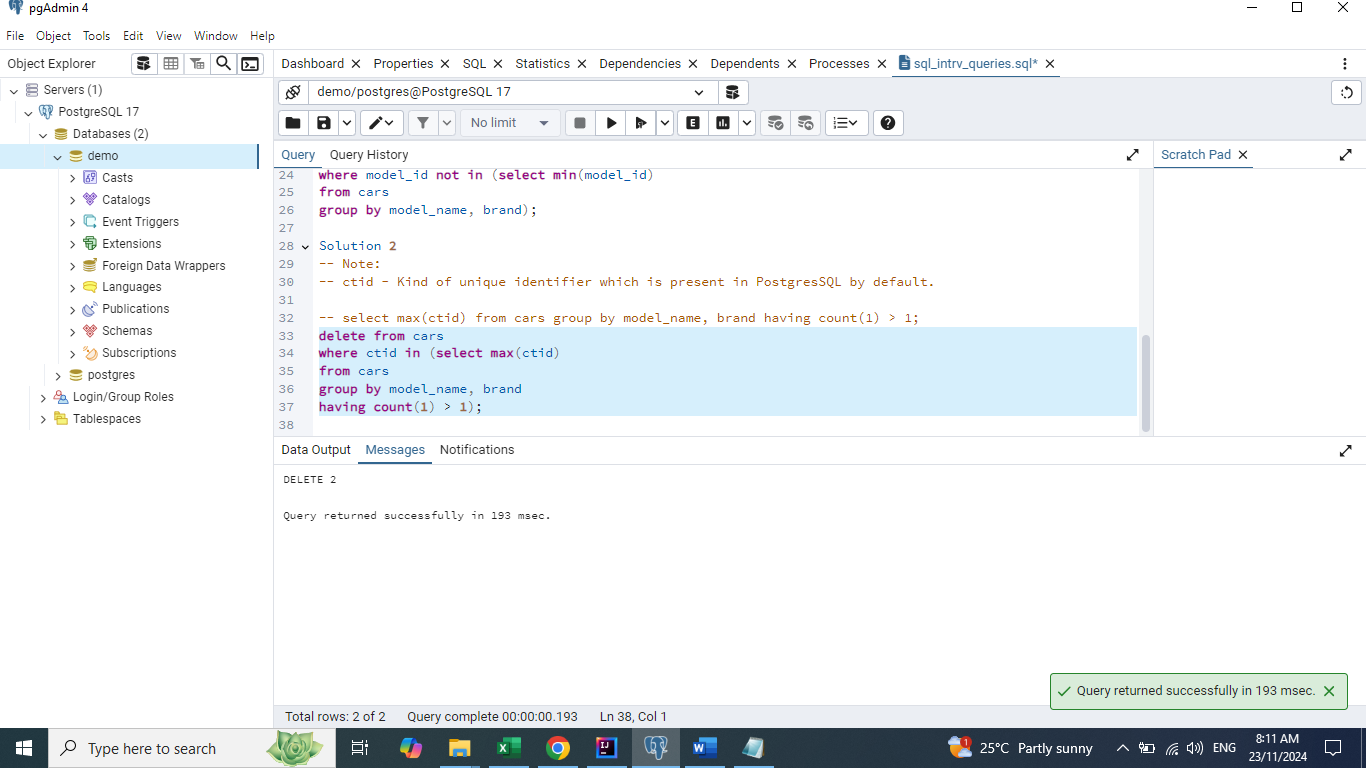
**delete from cars**

**where ctid in (select max(ctid)**

**from cars**

**group by model\_name, brand**

**having count(1) > 1);**



## Solution 3

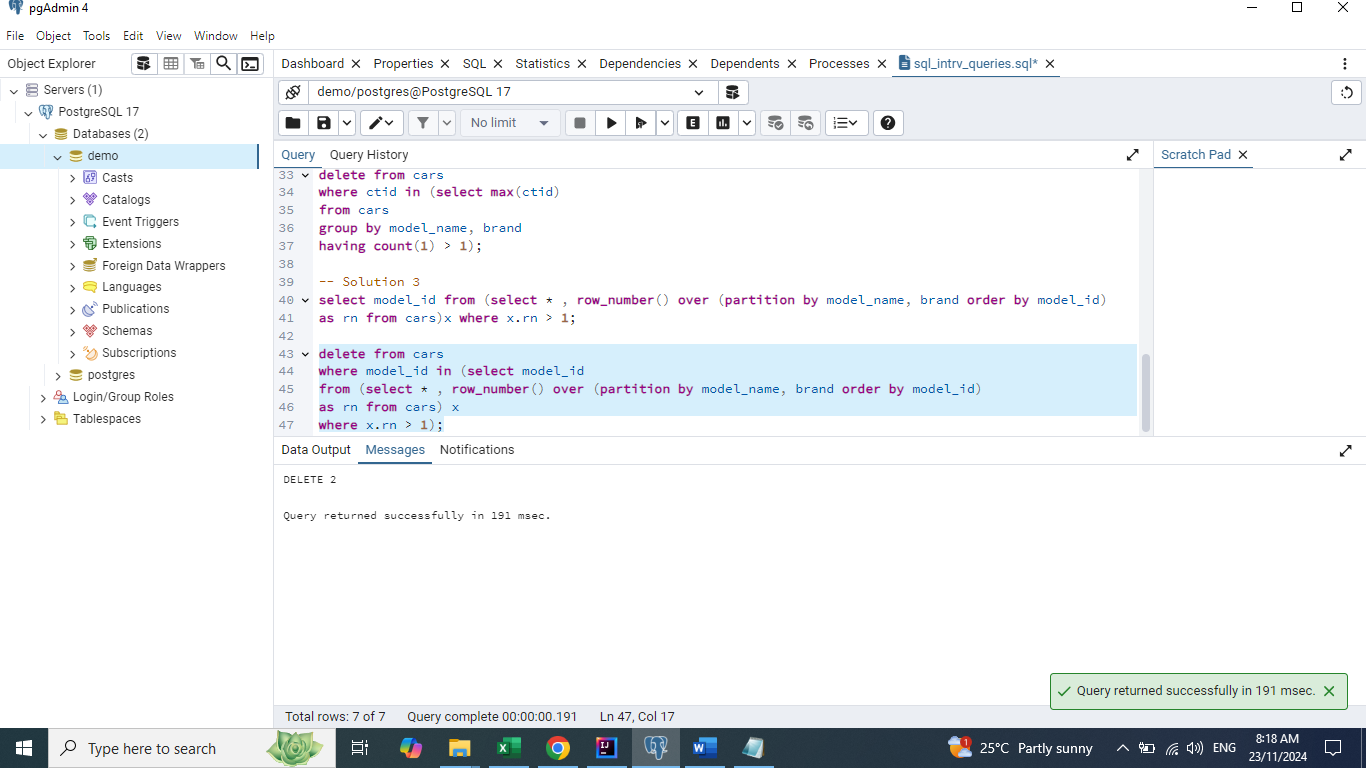
**delete from cars**

**where model\_id in (select model\_id**

**from (select \* , row\_number() over (partition by model\_name, brand order by model\_id)**

**as rn from cars) x**

**where x.rn > 1);**



# Display highest and lowest salary

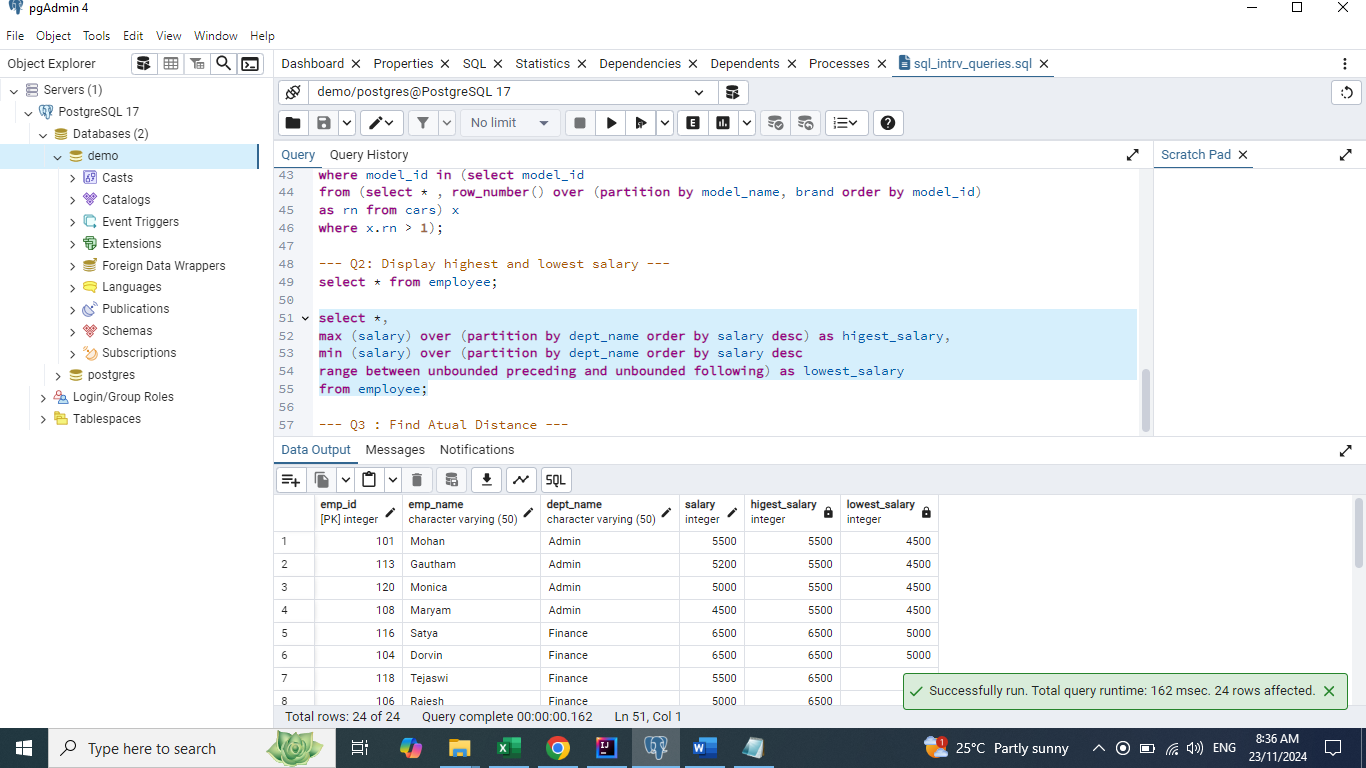
**select \*,**

**max (salary) over (partition by dept\_name order by salary desc) as higest\_salary,**

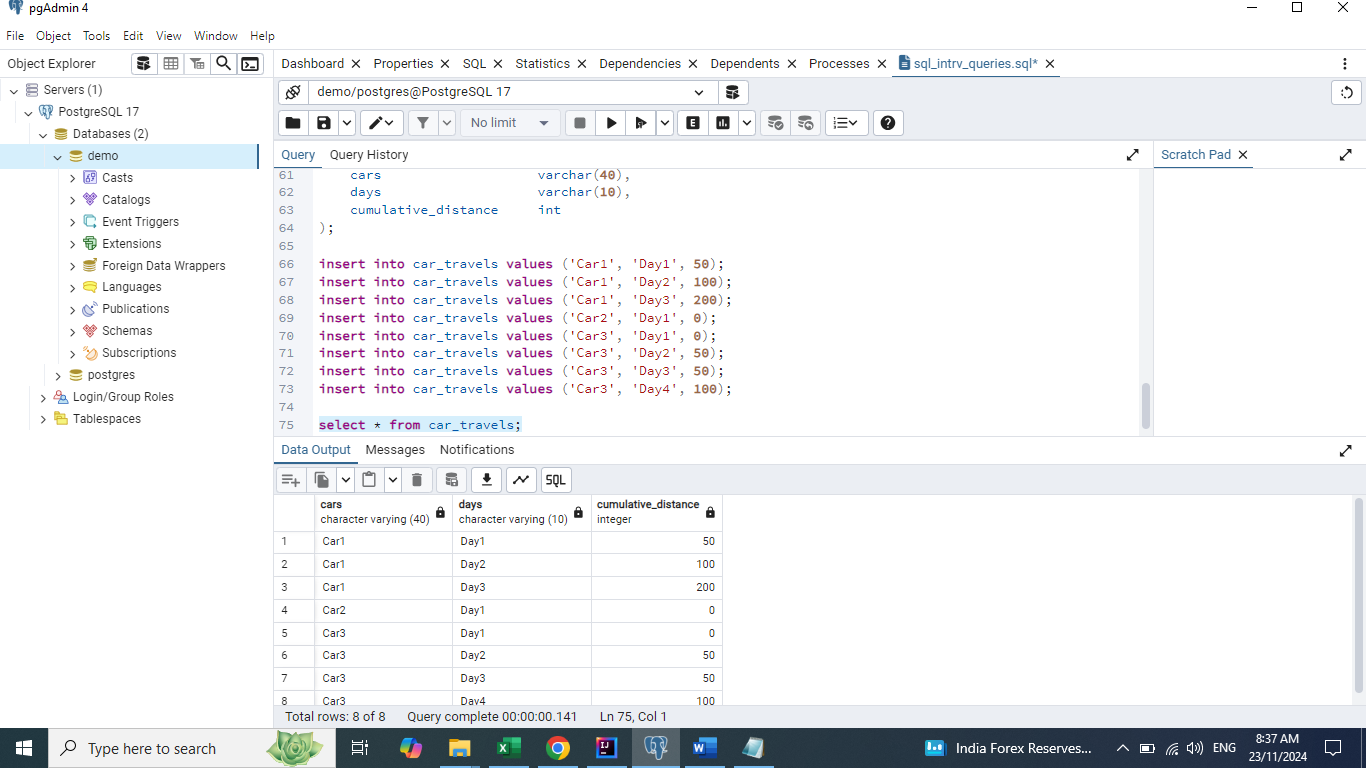
**min (salary) over (partition by dept\_name order by salary desc**

**range between unbounded preceding and unbounded following) as lowest\_salary**

**from employee;**



# Find Actual Distance

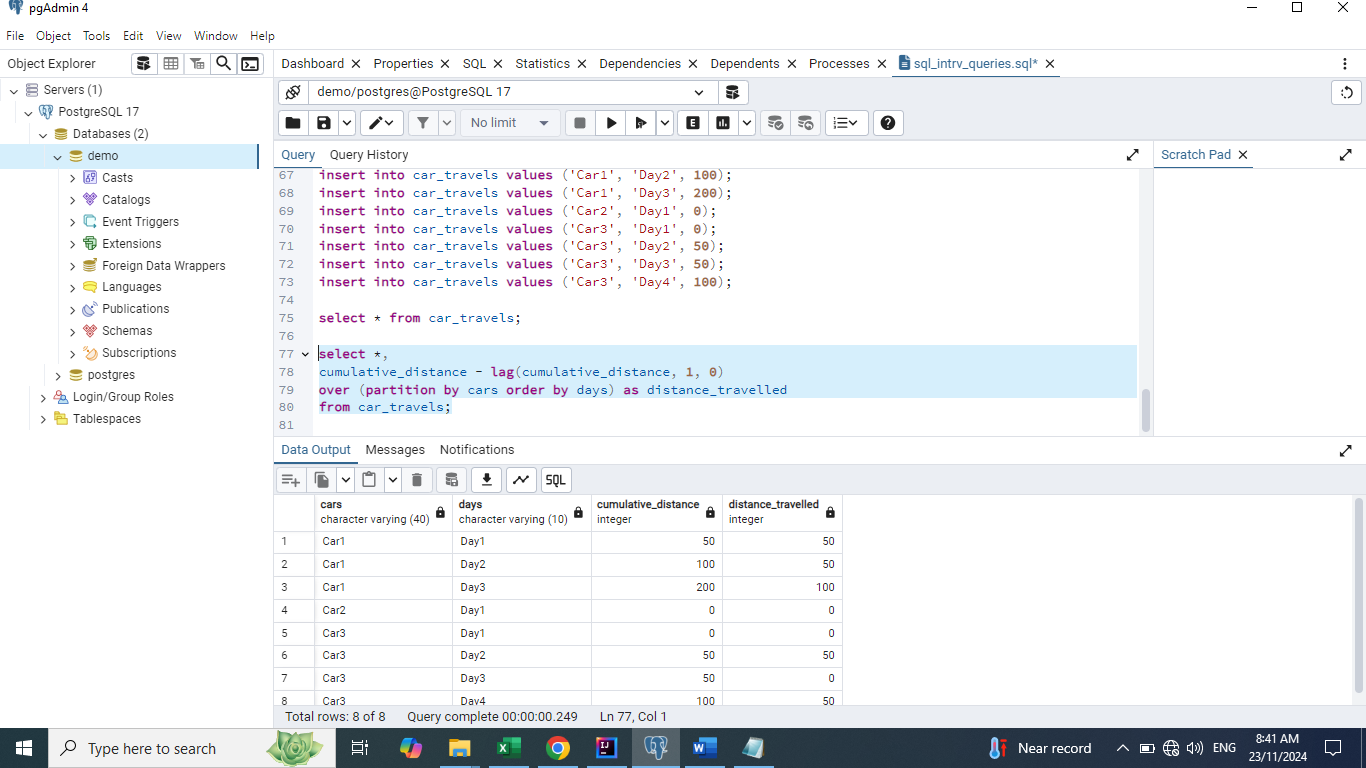


**select \*,**

**cumulative\_distance - lag(cumulative\_distance, 1, 0)**

**over (partition by cars order by days) as distance\_travelled**

**from car\_travels;**



# Convert the given input to expected output

**with cte as**

**(select \*,**

**row\_number() over() as rn**

**from src\_dest\_distance)**

**select t1.source, t1.destination, t1.distance**

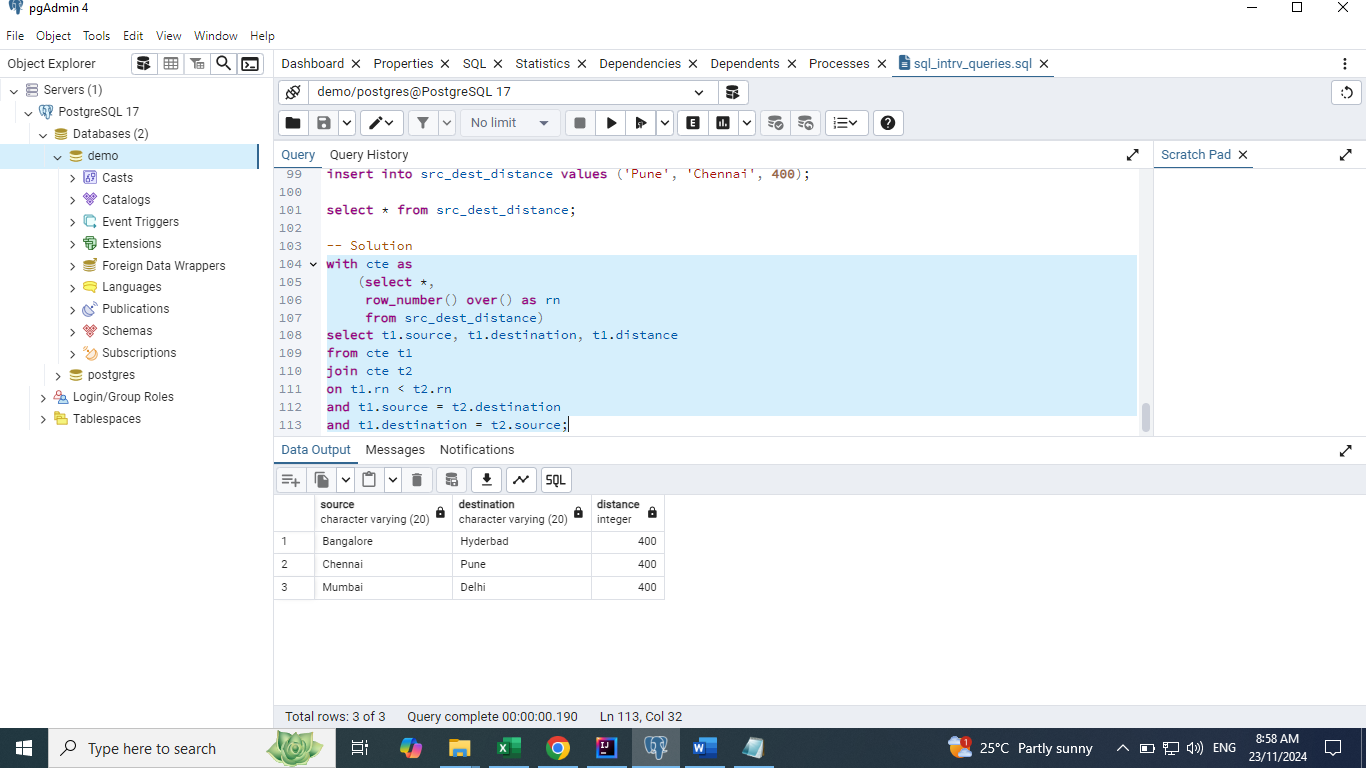
**from cte t1**

**join cte t2**

**on t1.rn < t2.rn**

**and t1.source = t2.destination**

**and t1.destination = t2.source;**



# Ungroup the given input data

**with recursive cte as**

**(select id, item\_name, total\_count, 1 as level**

**from travel\_items**

**union all**

**select cte.id, cte.item\_name, cte.total\_count - 1, level + 1 as level**

**from cte**

**join travel\_items t**

**on t.item\_name = cte.item\_name**

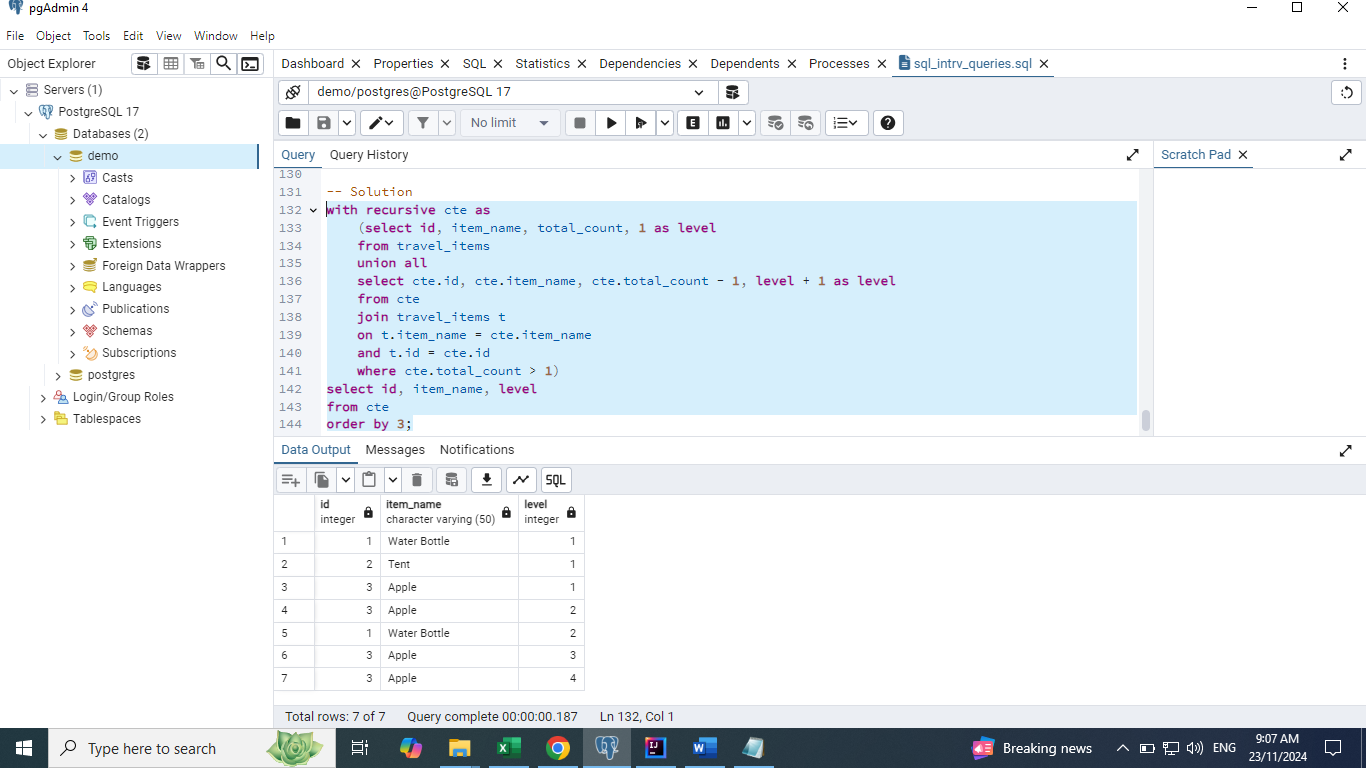
**and t.id = cte.id**

**where cte.total\_count > 1)**

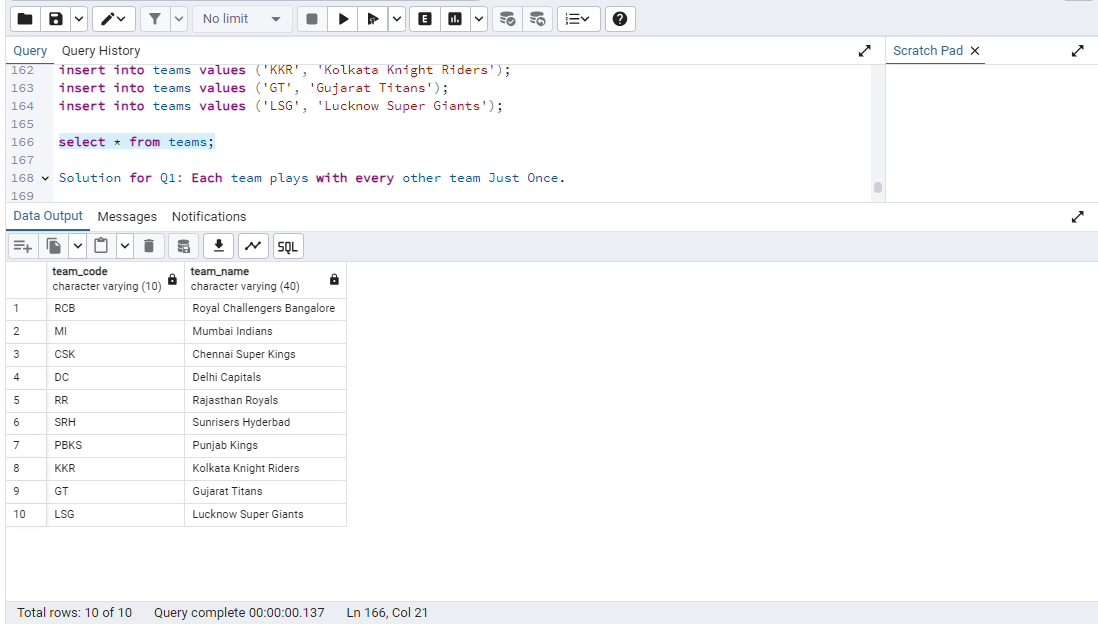
**select id, item\_name, level**

**from cte**

**order by 3;**



# IPL Matches



## Solution for Q1: Each team plays with every other team Just Once.

**with matches as**

**(select row\_number() over(order by team\_name) as id, t.\***

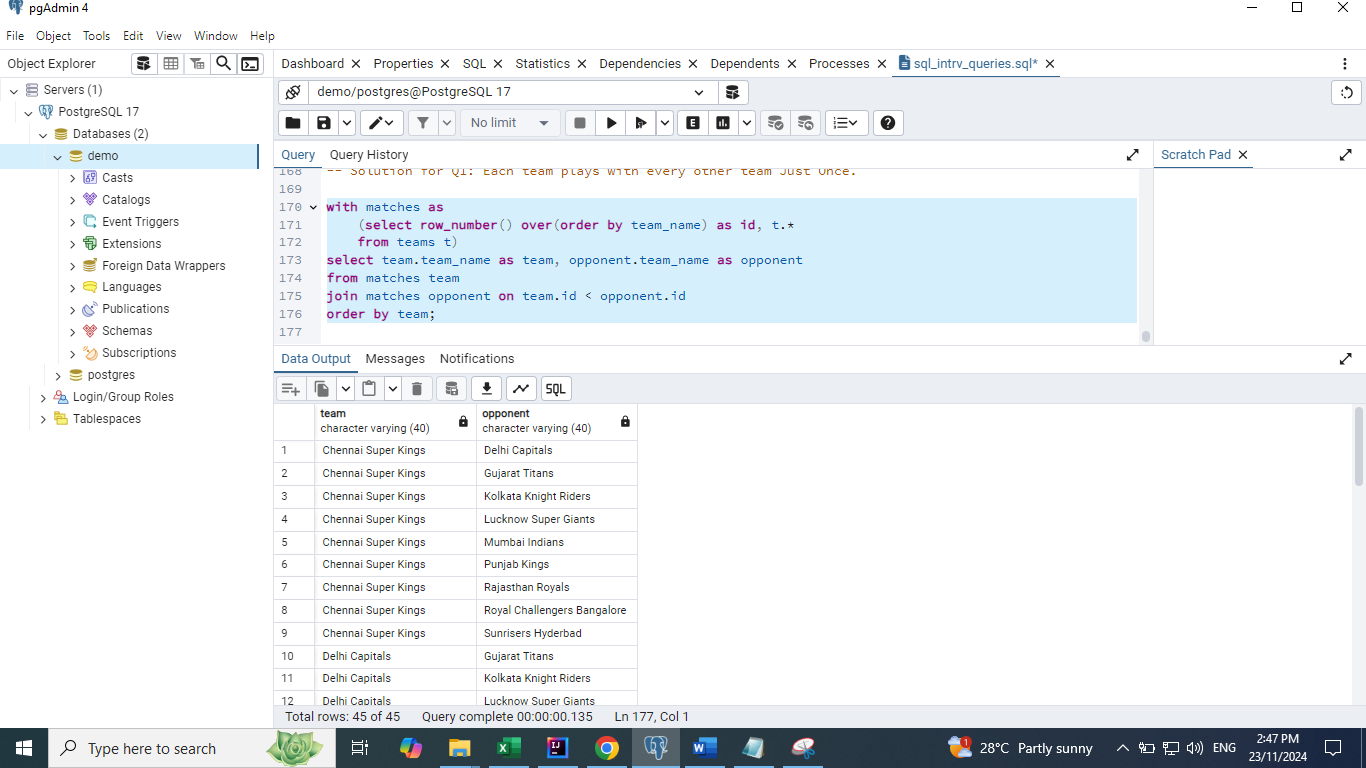
**from teams t)**

**select team.team\_name as team, opponent.team\_name as opponent**

**from matches team**

**join matches opponent on team.id < opponent.id**

**order by team;**



## Solution for Q2: Each team plays with every other team twice.

**with matches as**

**(select row\_number() over(order by team\_name) as id, t.\***

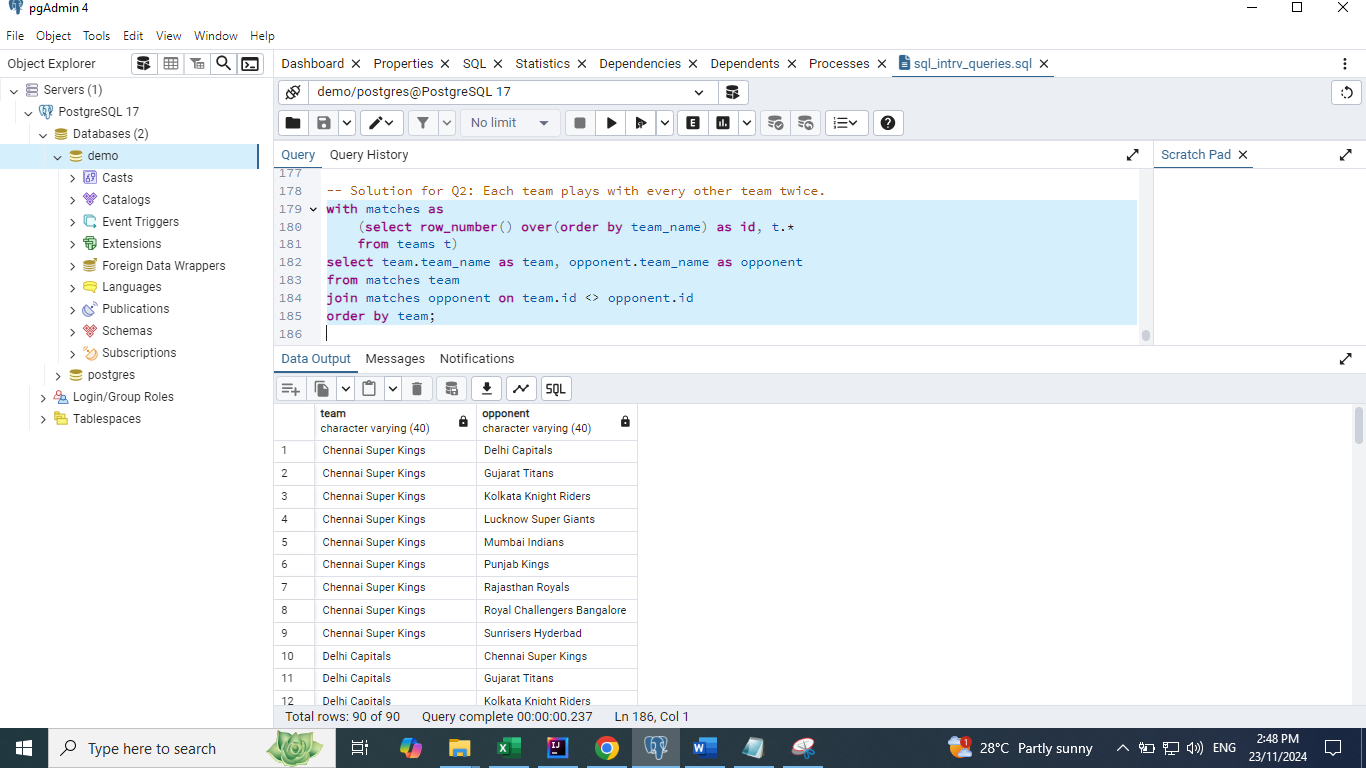
**from teams t)**

**select team.team\_name as team, opponent.team\_name as opponent**

**from matches team**

**join matches opponent on team.id <> opponent.id**

**order by team;**



# Find the hierarchy

**with recursive cte as**

**(select \* from emp\_details**

**where name = 'Asha'**

**union**

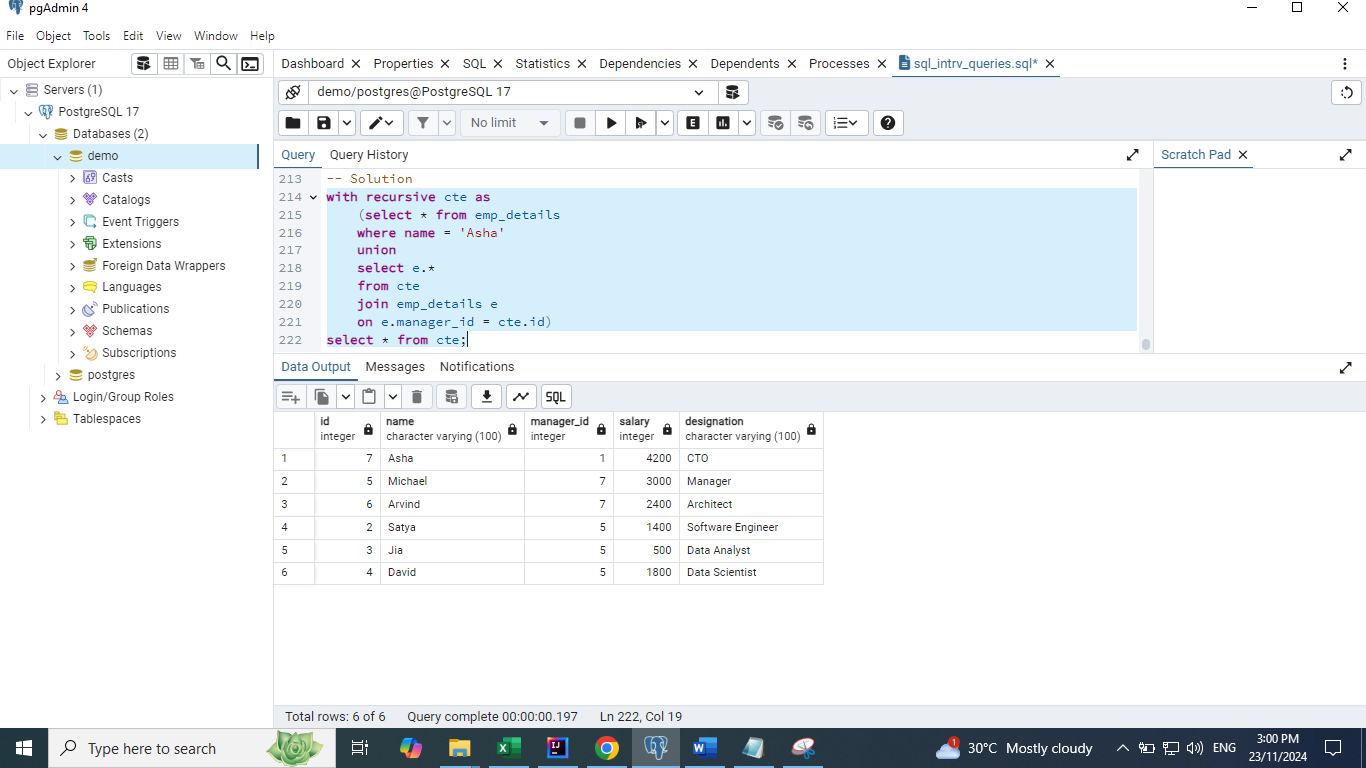
**select e.\***

**from cte**

**join emp\_details e**

**on e.manager\_id = cte.id)**

**select \* from cte;**



# Find difference in average sales

**with cte as**

**(select year\_id, month\_id, to\_char(order\_date, 'MON') as mon, avg(sales) as avg\_sales\_per\_mon**

**from sales\_order\_detail sod**

**where year\_id in (2003, 2004)**

**group by year\_id, month\_id, to\_char(order\_date, 'MON'))**

**select year2k3.mon, round(abs(year2k4.avg\_sales\_per\_mon - year2k3.avg\_sales\_per\_mon)::decimal, 2)**

**as sales\_difference**

**from cte year2k3**

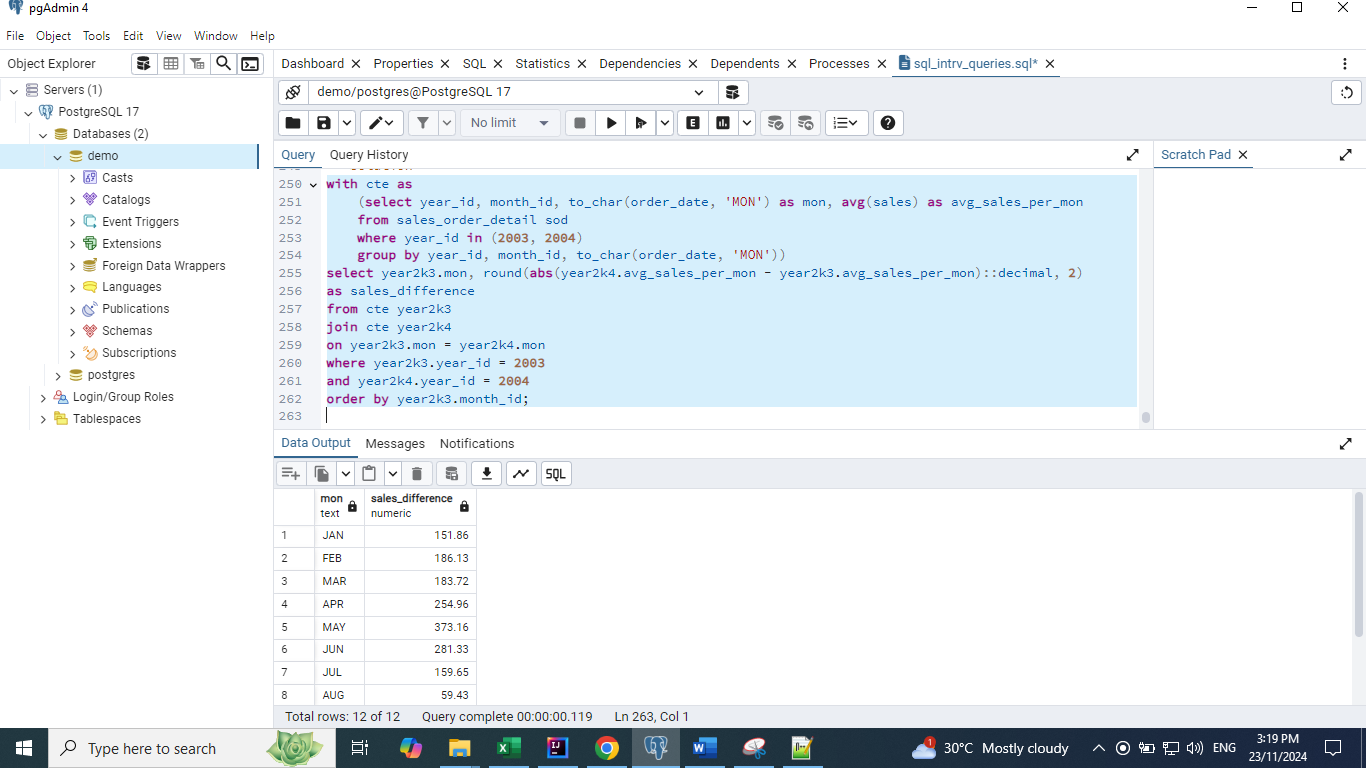
**join cte year2k4**

**on year2k3.mon = year2k4.mon**

**where year2k3.year\_id = 2003**

**and year2k4.year\_id = 2004**

**order by year2k3.month\_id;**



# Pizza Delivery Status

**A pizza company is taking orders from customers, and each pizza ordered is added to their database as a separate order.**

**Each order has an associated status, "CREATED or SUBMITTED or DELIVERED'.**

**An order's Final\_ Status is calculated based on status as follows:**

**1. When all orders for a customer have a status of DELIVERED, that customer's order has a Final\_Status of COMPLETED.**

**2. If a customer has some orders that are not DELIVERED and some orders that are DELIVERED, the Final\_ Status is IN PROGRESS.**

**3. If all of a customer's orders are SUBMITTED, the Final\_Status is AWAITING PROGRESS.**

**4. Otherwise, the Final Status is AWAITING SUBMISSION.**

**Write a query to report the customer\_name and Final\_Status of each customer's arder. Order the results by customer name.**

**select distinct cust\_name as customer, 'COMPLETED' as status**

**from cust\_orders co**

**where co.status = 'DELIVERED'**

**and not exists (select 1 from cust\_orders co2**

**where co2.cust\_name = co.cust\_name**

**and co2.status in ('SUBMITTED', 'CREATED'))**

**union**

**select distinct cust\_name as customer, 'IN PROGRESS' as status**

**from cust\_orders co**

**where co.status = 'DELIVERED'**

**and exists (select 1 from cust\_orders co2**

**where co2.cust\_name = co.cust\_name**

**and co2.status in ('SUBMITTED', 'CREATED'))**

**union**

**select distinct cust\_name as customer, 'AWAITING PROGRESS' as status**

**from cust\_orders co**

**where co.status = 'SUBMITTED'**

**and not exists (select 1 from cust\_orders co2**

**where co2.cust\_name = co.cust\_name**

**and co2.status in ('DELIVERED'))**

**union**

**select distinct cust\_name as customer, 'AWAITING SUBMISSION' as status**

**from cust\_orders co**

**where co.status = 'CREATED'**

**and not exists (select 1 from cust\_orders co2**

**where co2.cust\_name = co.cust\_name**

**and co2.status in ('SUBMITTED', 'DELIVERED'));**

