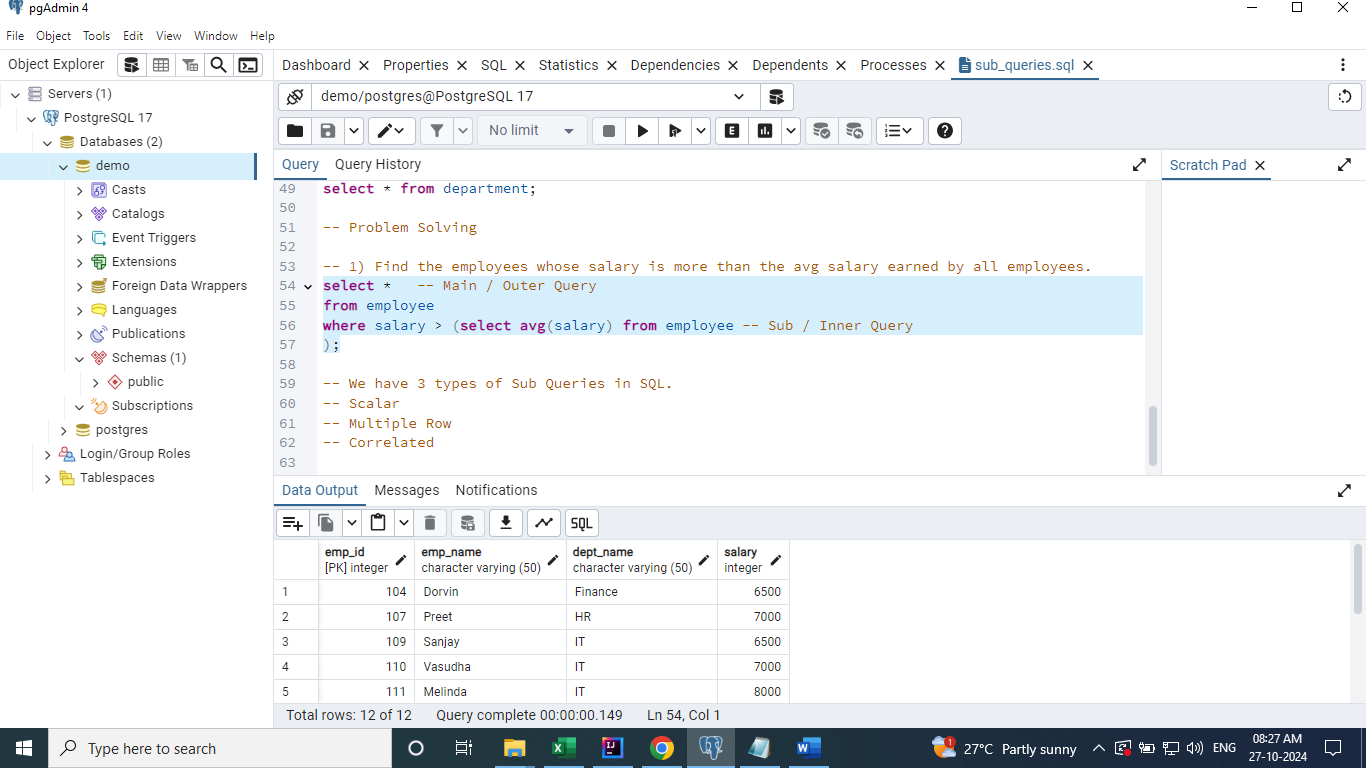
# Find the employees whose salary is more than the avg salary earned by all employees.

**select \* -- Main / Outer Query**

**from employee**

**where salary > (select avg(salary) from employee -- Sub / Inner Query**

**);**



-- We have 3 types of Sub Queries in SQL.

-- Scalar

-- Multiple Row

-- Correlated

-- Scalar SubQuery

-- It will return one row and one column. It can be used in from and where clauses.

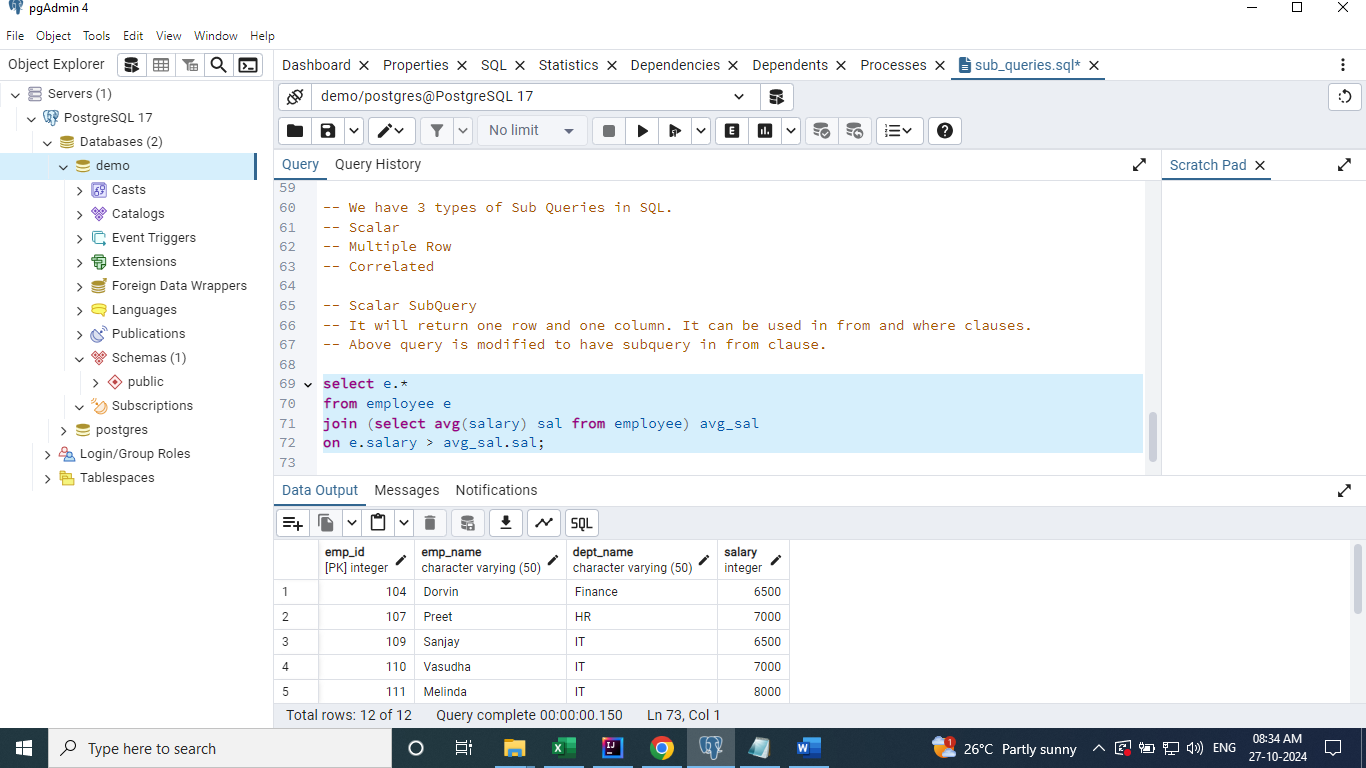
-- Above query is modified to have subquery in from clause.

**select e.\***

**from employee e**

**join (select avg(salary) sal from employee) avg\_sal**

**on e.salary > avg\_sal.sal;**



-- Multiple Row SubQuery

-- Subquery which returns multiple rows and columns.

-- Subquery which returns single column and multiple rows.

# Find the employees who earn the highest salary in each department.

**select \***

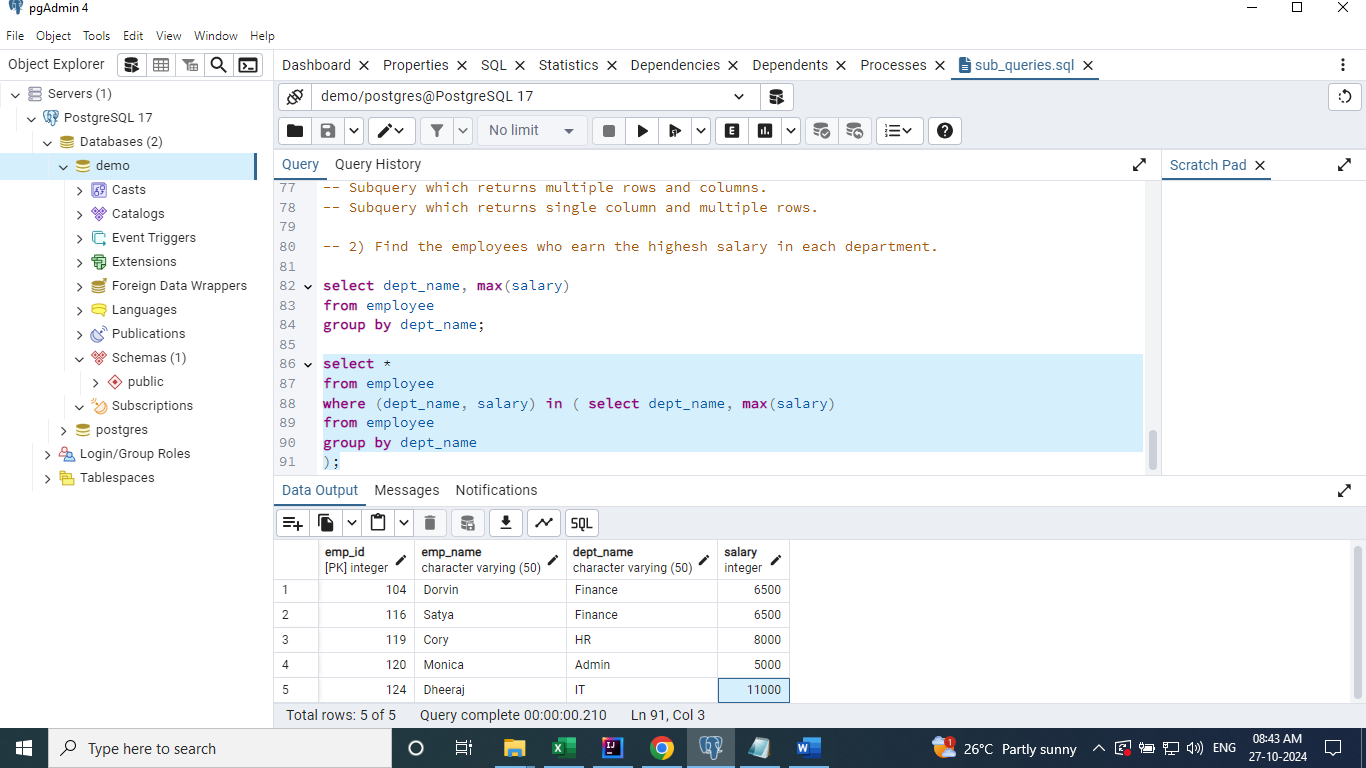
**from employee**

**where (dept\_name, salary) in ( select dept\_name, max(salary)**

**from employee**

**group by dept\_name**

**);**



# Find the department who do not have any employees.

-- Single column and multiple rows

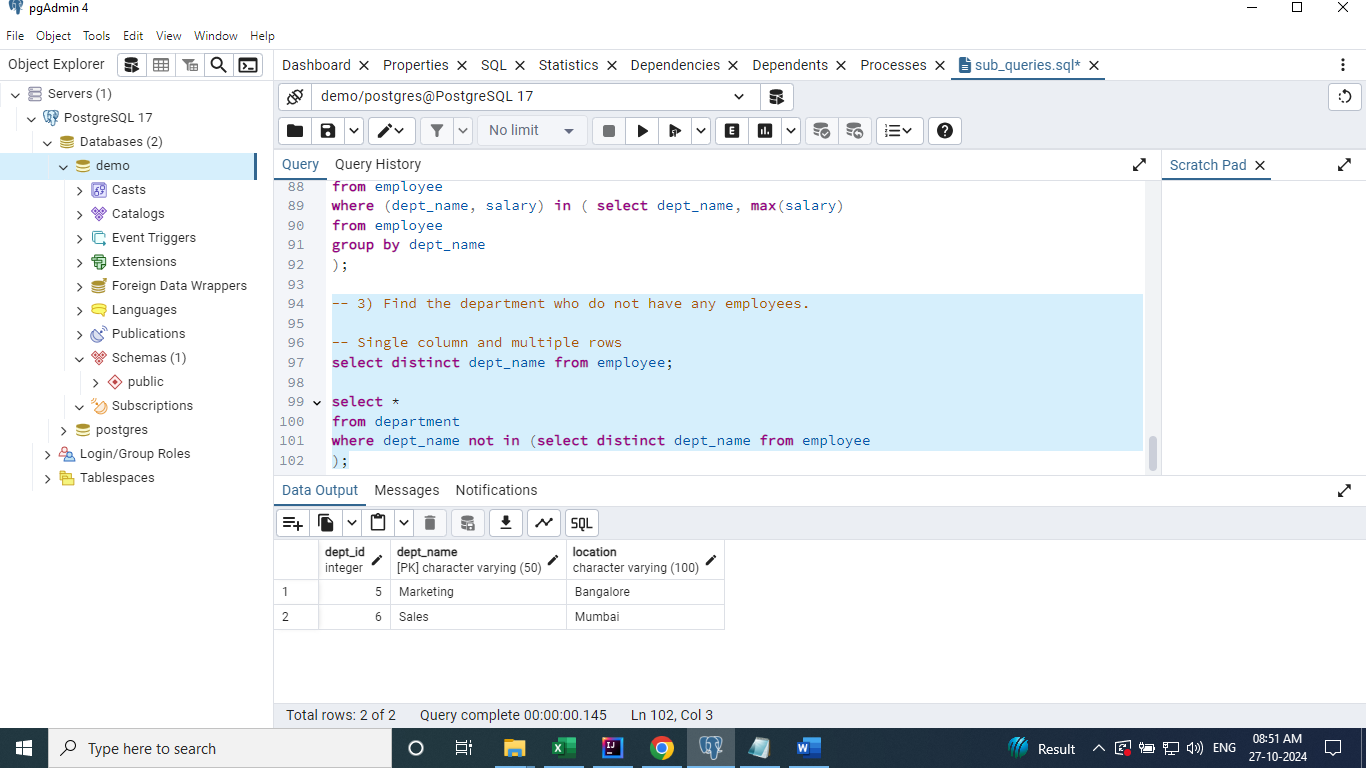
select distinct dept\_name from employee;

**select \***

**from department**

**where dept\_name not in (select distinct dept\_name from employee**

**);**



# Find the employees in each department who earn more than the average salary in that department

**-- Correlated SubQuery**

**-- A Subquery which is related to the outer query.**

**select avg(salary) from employee where dept\_name = "specific\_department";**

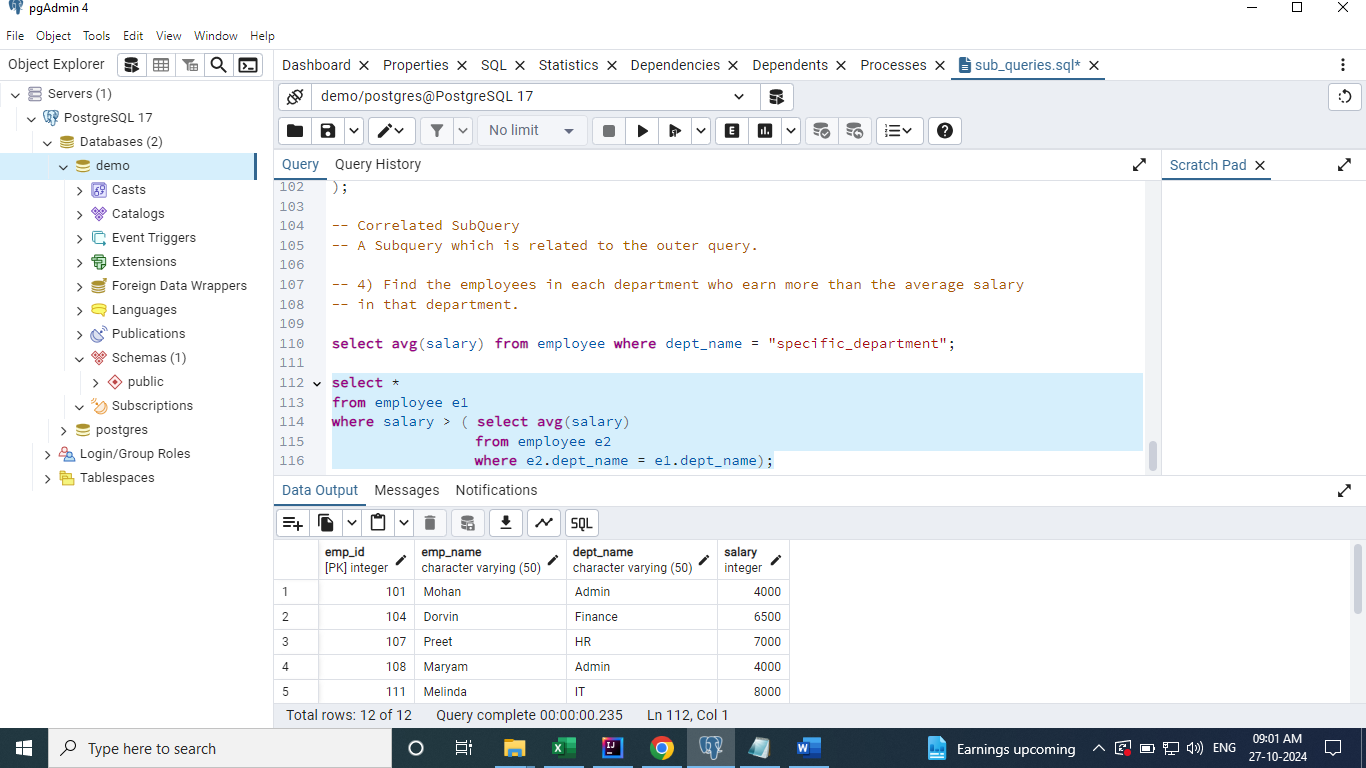
**select \***

**from employee e1**

**where salary > ( select avg(salary)**

**from employee e2**

**where e2.dept\_name = e1.dept\_name);**



# Find the department who do not have any employees using Correlated SubQueries.

**select \***

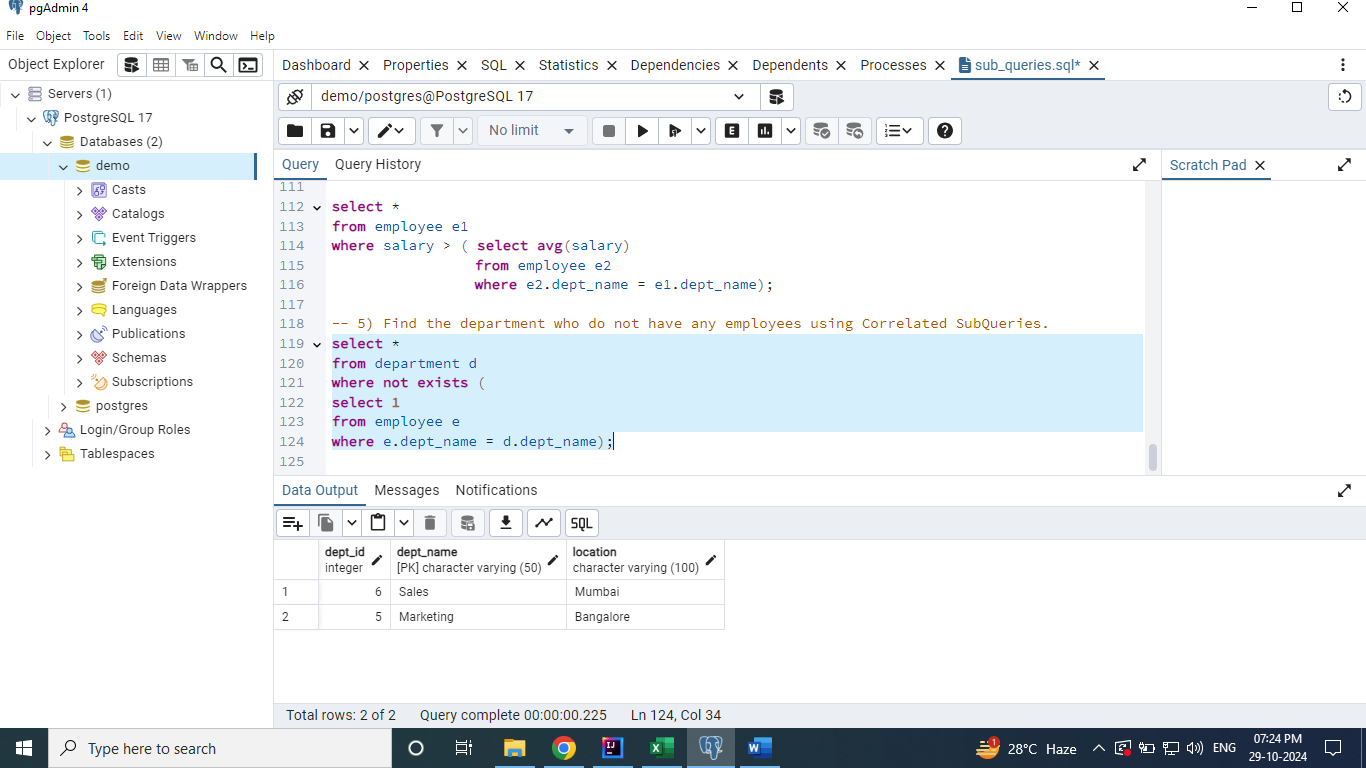
**from department d**

**where not exists (**

**select 1**

**from employee e**

**where e.dept\_name = d.dept\_name);**



# Find the stores whose sales is better than the average sales across all stores.

**select \***

**from ( select store\_name, sum(price) as total\_sales**

**from sales**

**group by store\_name) sales**

**join ( select avg(total\_sales) as avg\_sales**

**from ( select store\_name, sum(price) as total\_sales**

**from sales**

**group by store\_name)) as avg\_sales**

**on sales.total\_sales > avg\_sales.avg\_sales;**

**With clause is used to replace the sub queries which is used in multiple places.**

**with sales as ( select store\_name, sum(price) as total\_sales**

**from sales**

**group by store\_name )**

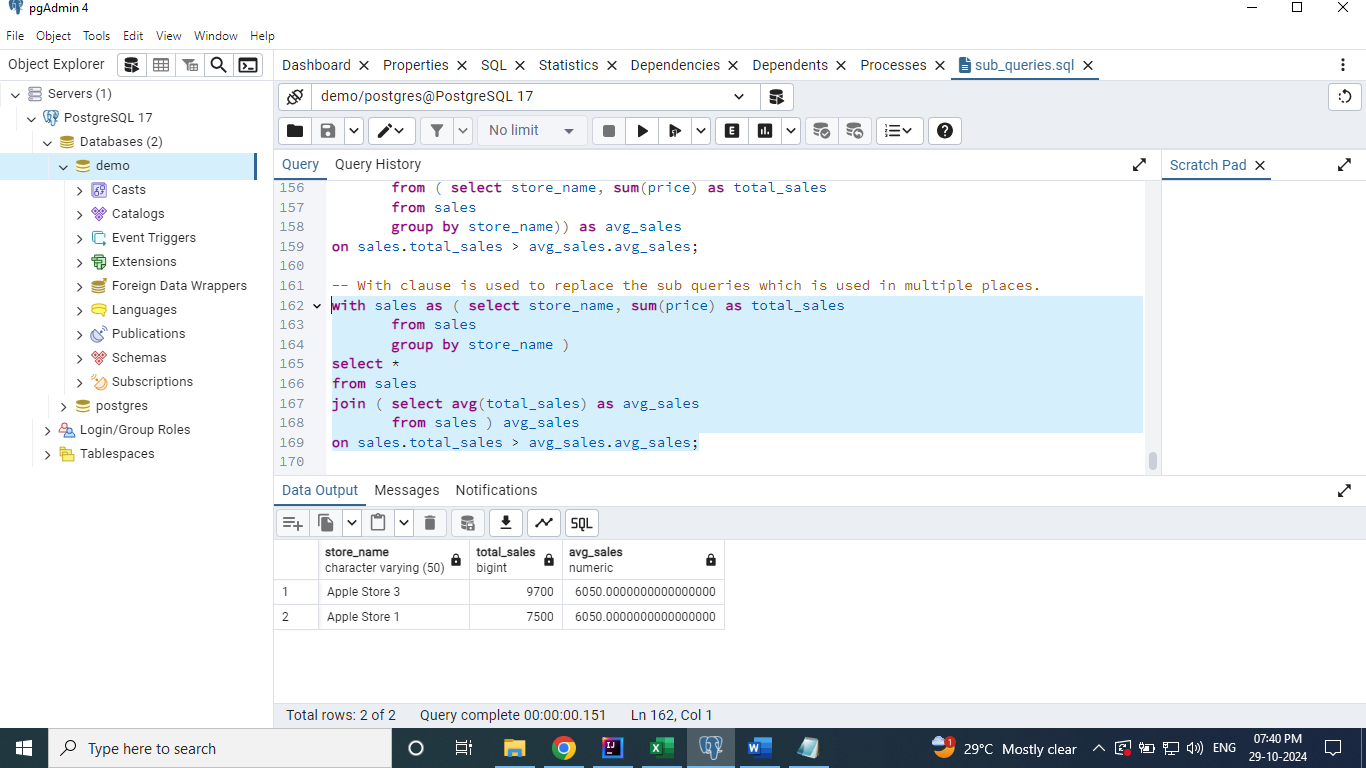
**select \***

**from sales**

**join ( select avg(total\_sales) as avg\_sales**

**from sales ) avg\_sales**

**on sales.total\_sales > avg\_sales.avg\_sales;**



# 7. Fetch all the employee details and add remarks to those employees who earn mor than the average pay.

**select \*,**

**(case when salary > (select avg(salary) from employee)**

**then 'higher than average'**

**else null**

**end) as remarks**

**from employee;**

**Modifying above query**

**select \*,**

**(case when salary > avg\_sal.sal**

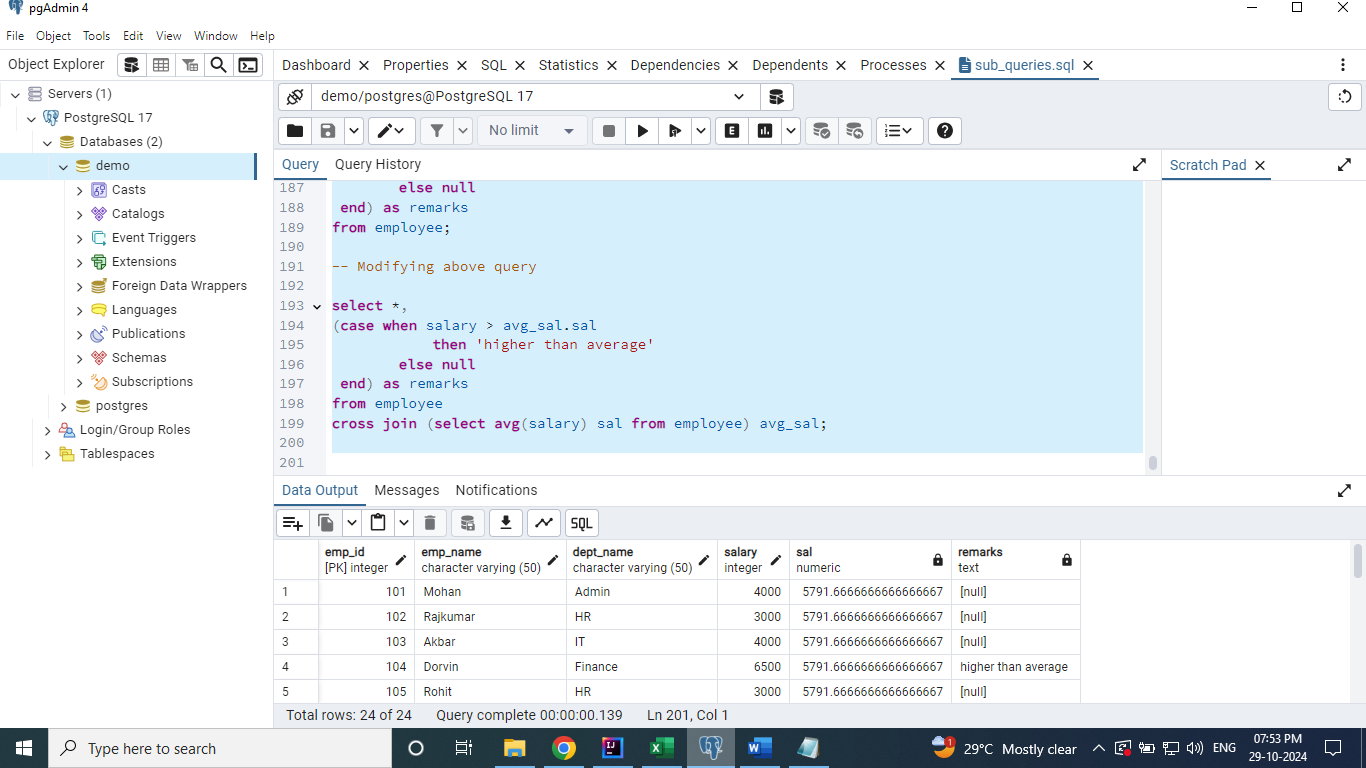
**then 'higher than average'**

**else null**

**end) as remarks**

**from employee**

**cross join (select avg(salary) sal from employee) avg\_sal;**



# 8. Find the stores which has sold more units than the average units sold by all the stores.

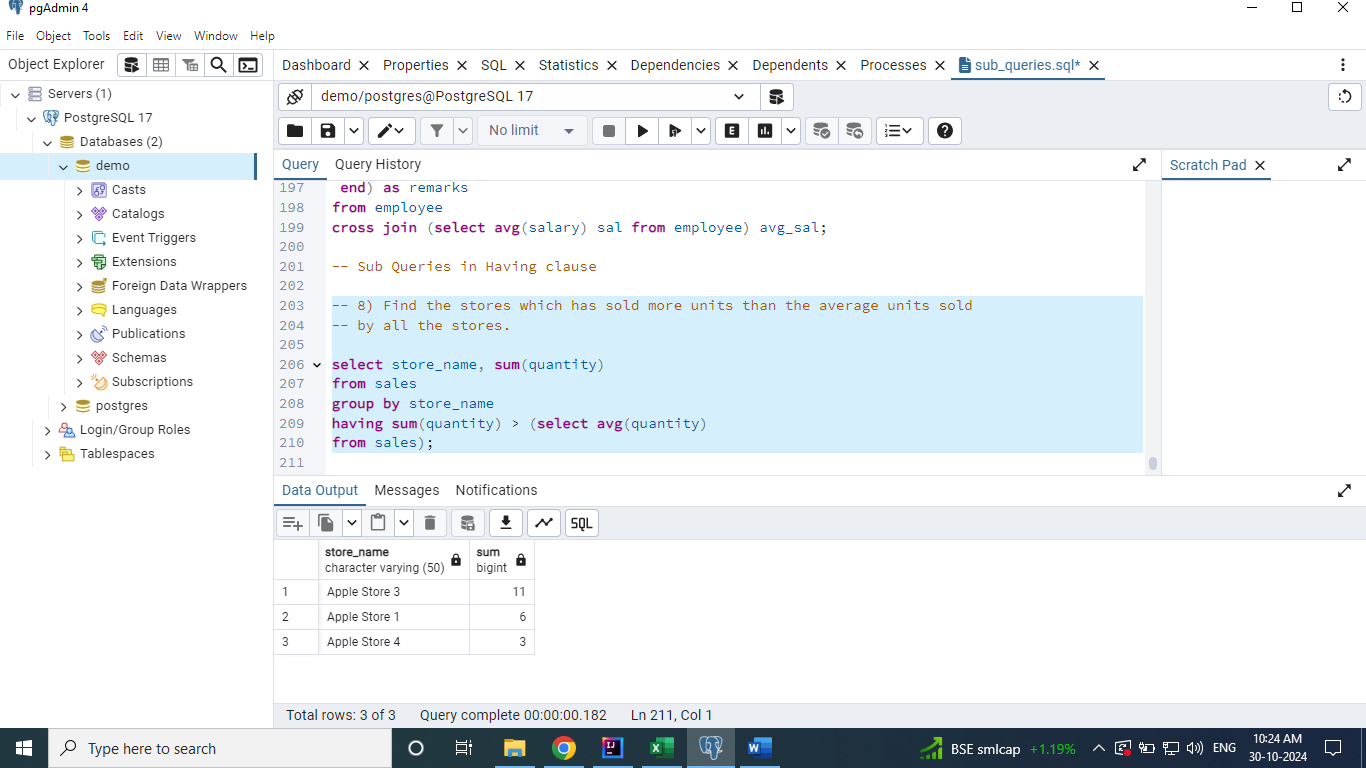
**select store\_name, sum(quantity)**

**from sales**

**group by store\_name**

**having sum(quantity) > (select avg(quantity)**

**from sales);**



**-- The subqueries can be used in another sql commands as well.**

**-- INSERT**

**-- UPDATE**

**-- DELETE**

# 9. Insert data to employee history table. Make sure not insert duplicate records.

**Scenario: Whenever we need to insert data into a table based on the availability**

**of data in another tables. Then, we need to use Sub Queries.**

**insert into employee\_history**

**select e.emp\_id, e.emp\_name, d.dept\_name, e.salary, d.location**

**from employee e**

**join department d**

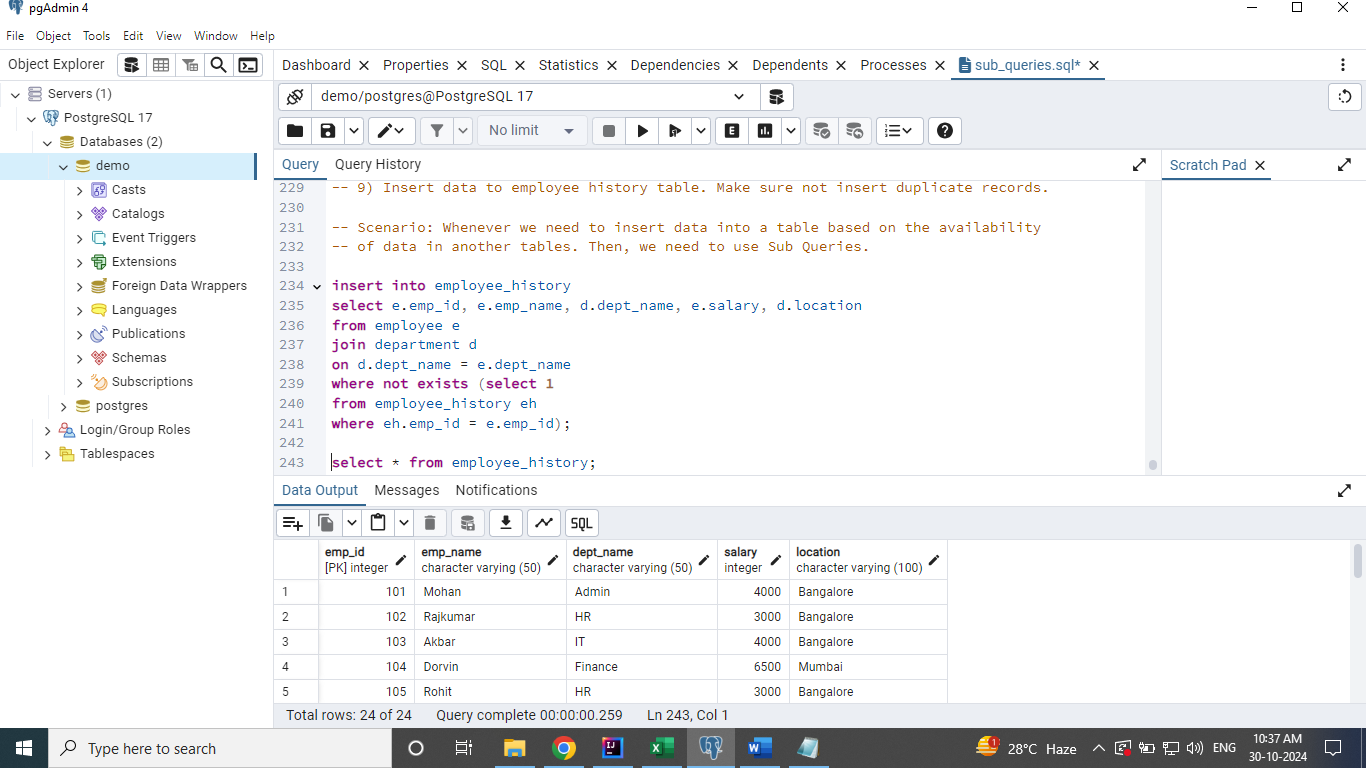
**on d.dept\_name = e.dept\_name**

**where not exists (select 1**

**from employee\_history eh**

**where eh.emp\_id = e.emp\_id);**

**select \* from employee\_history;**



# 10. Given 10% increment to all the employees in bangalore location based on the maximum salary earned by employee in each dept. Only consider employees in employee\_history table.

**Update employee e**

**set salary = (select max(salary) + (max(salary) \* 0.1)**

**from employee\_history eh**

**where eh.dept\_name = e.dept\_name)**

**where e.dept\_name in (select dept\_name**

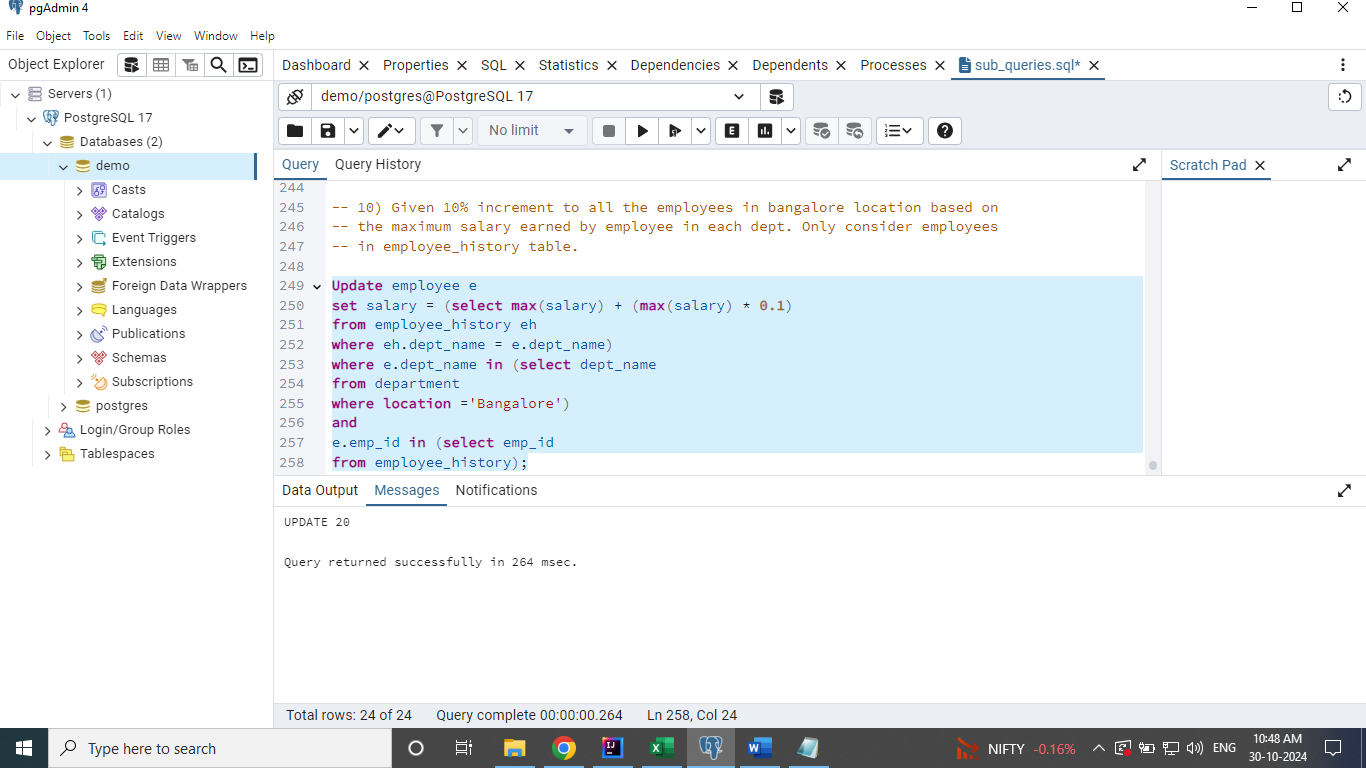
**from department**

**where location ='Bangalore')**

**and**

**e.emp\_id in (select emp\_id**

**from employee\_history);**



# 11. Delete all departments which do not have any employees.

**delete**

**from department**

**where dept\_name in (select dept\_name**

**from department d**

**where not exists (select 1**

**from employee e**

**where e.dept\_name = d.dept\_name));**

