**SQL Window Function**

# Window Function

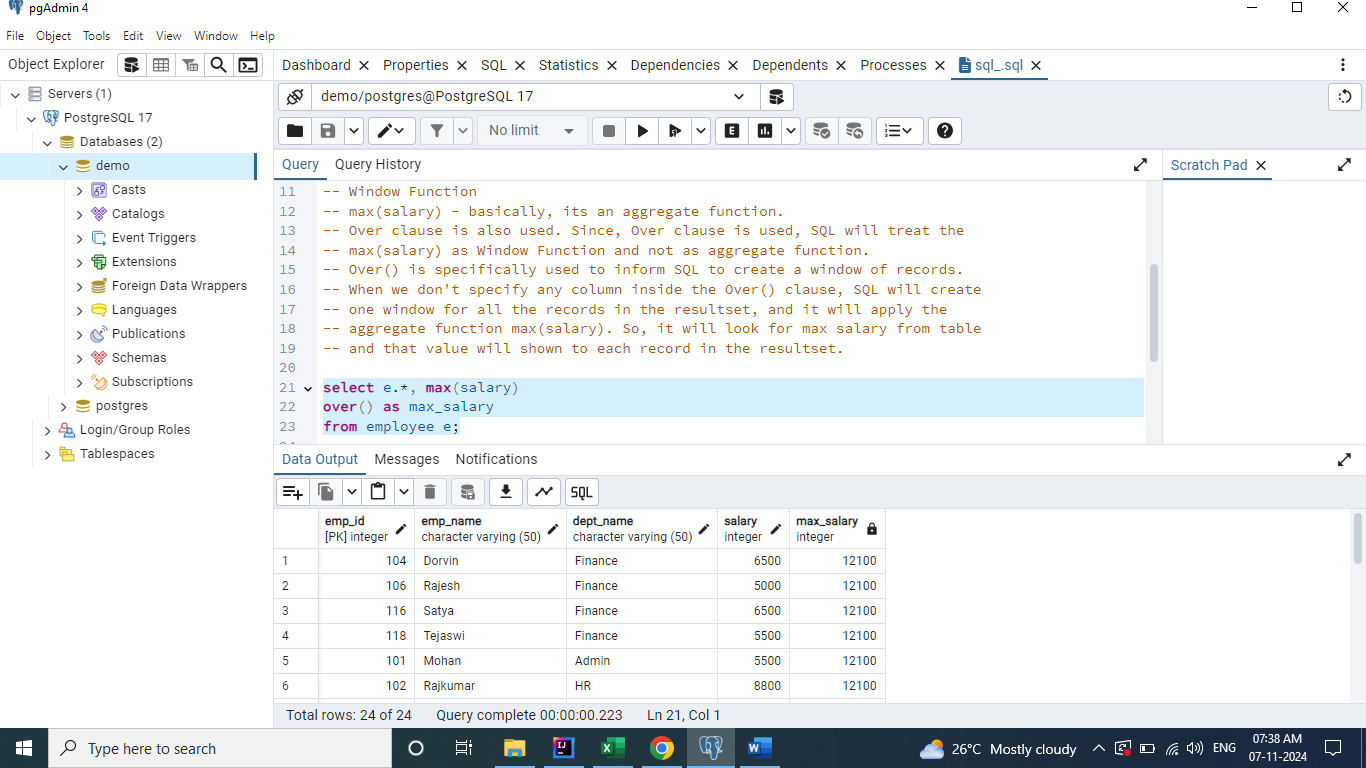
max(salary) - basically, it’s an aggregate function. Over clause is also used. Since, over clause is used, SQL will treat the max(salary) as Window Function and not as aggregate function.

Over() is specifically used to inform SQL to create a window of records. When we don't specify any column inside the Over() clause, SQL will create one window for all the records in the resultset, and it will apply the aggregate function max(salary). So, it will look for max salary from table and that value will show to each record in the resultset.

**select e.\*, max(salary)**

**over() as max\_salary**

**from employee e;**

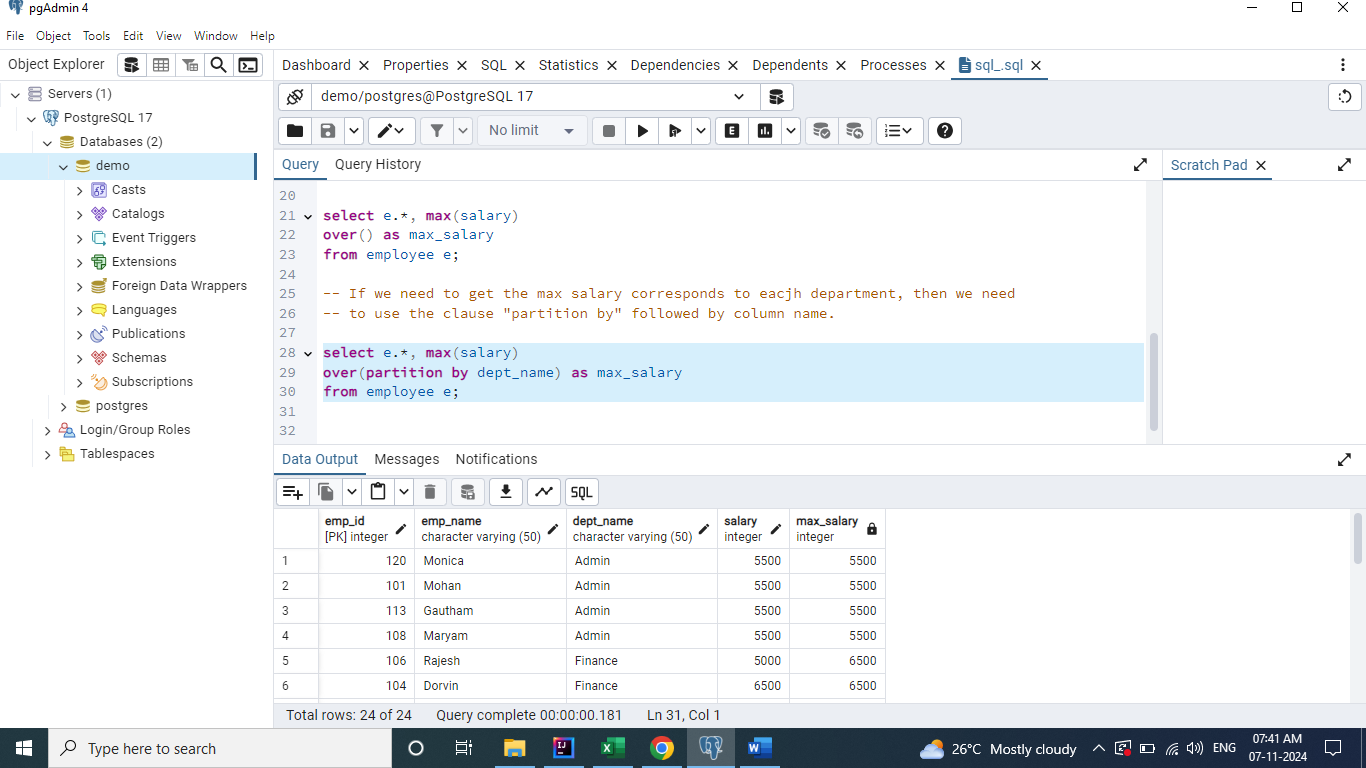


If we need to get the max salary corresponds to each department, then we need to use the clause "partition by" followed by column name.

**select e.\*, max(salary)**

**over(partition by dept\_name) as max\_salary**

**from employee e;**



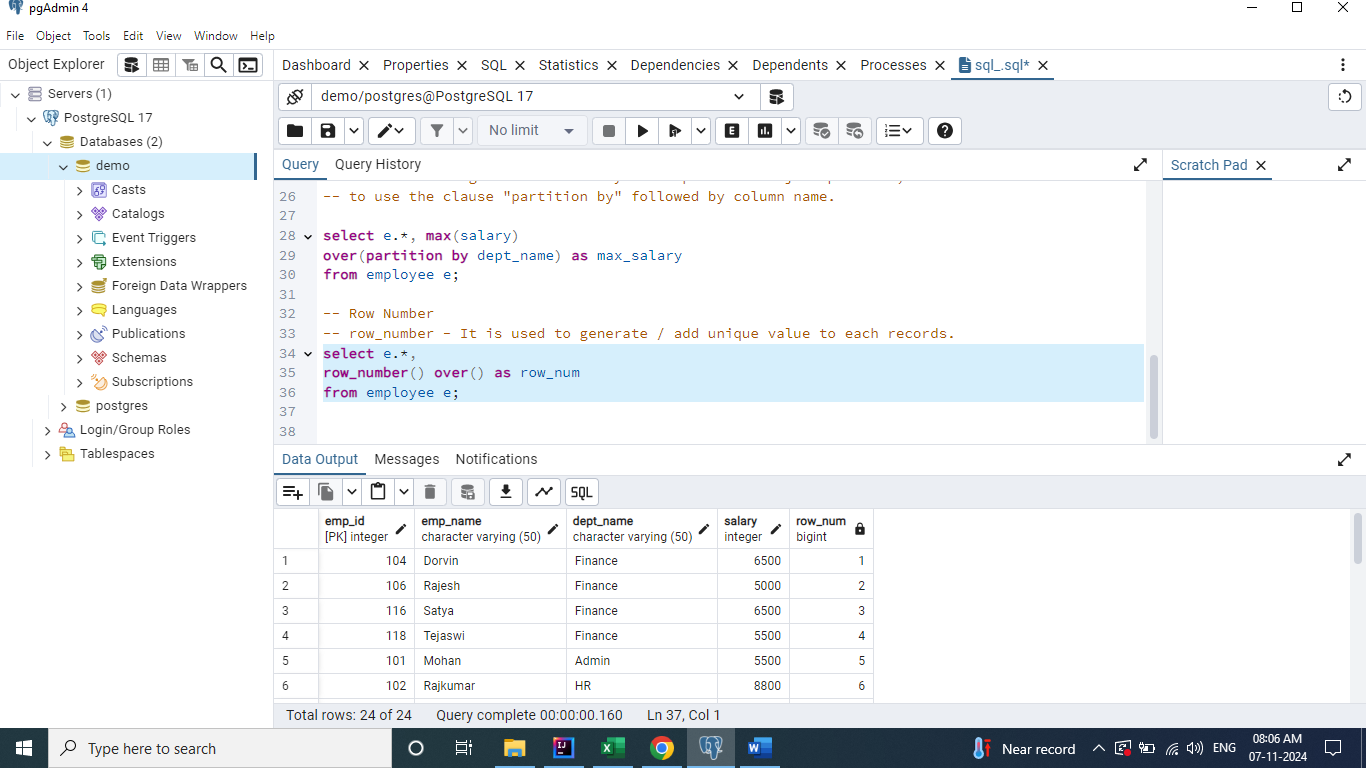
# Row Number

**It is used to generate / add unique value to each record.**

**select e.\*,**

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

**from employee e;**

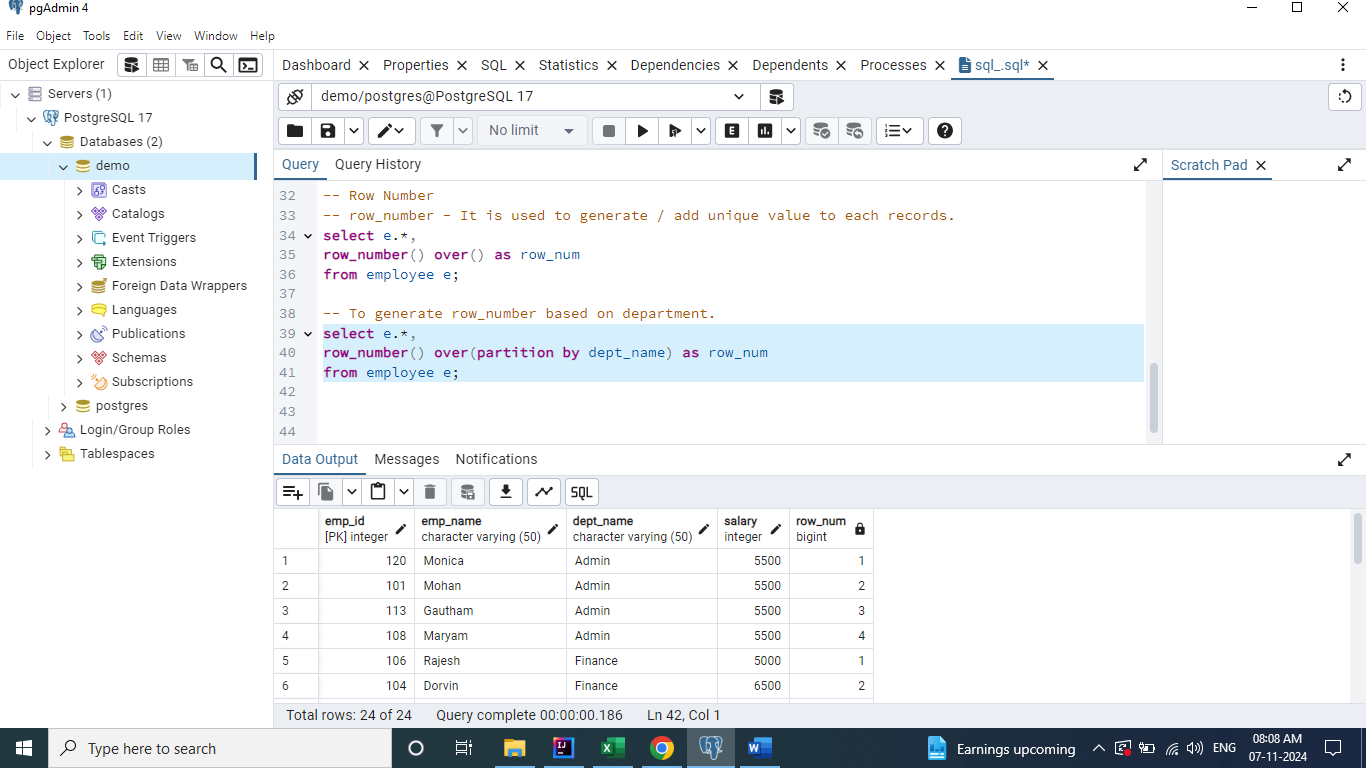


**To generate row\_number based on department.**

**select e.\*,**

**row\_number() over(partition by dept\_name) as row\_num**

**from employee e;**



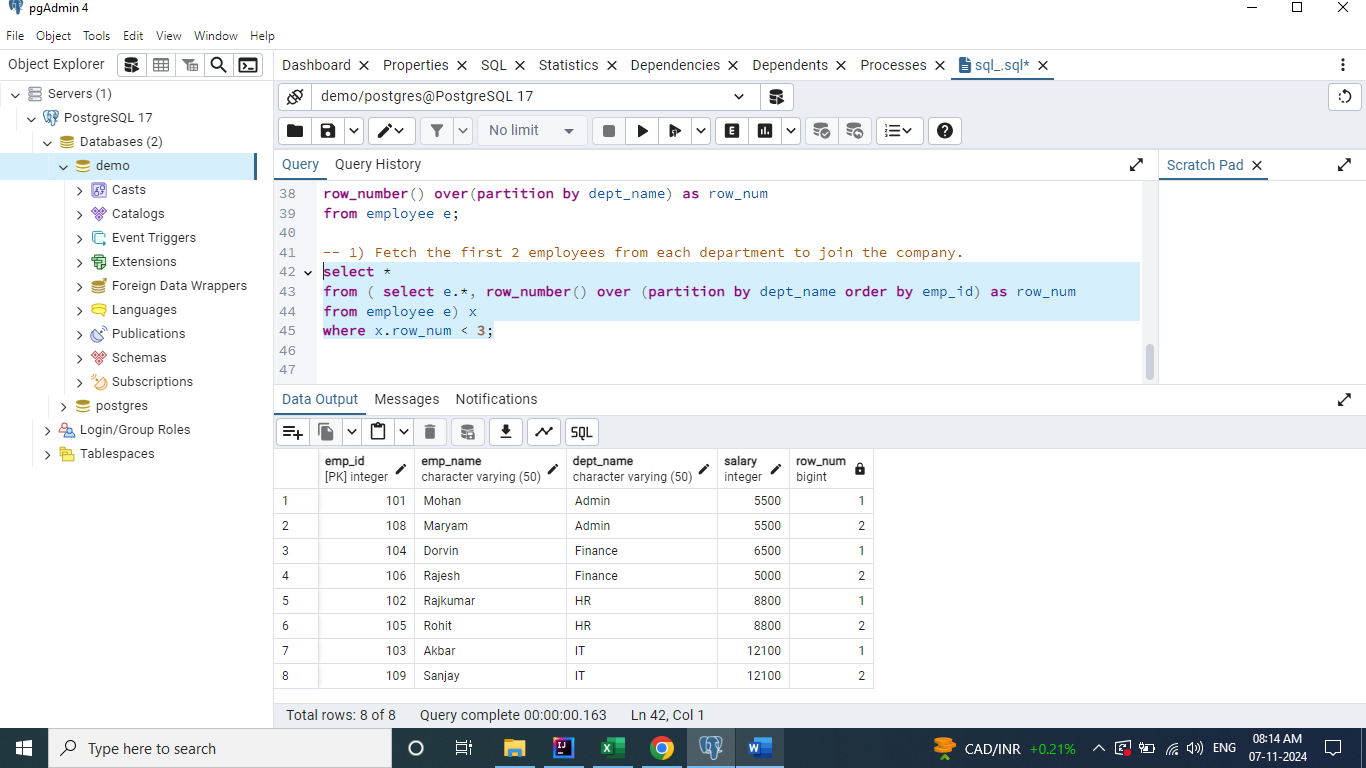
## 1) Fetch the first 2 employees from each department to join the company.

**select \***

**from ( select e.\*, row\_number() over (partition by dept\_name order by emp\_id) as row\_num**

**from employee e) x**

**where x.row\_num < 3;**



# Rank

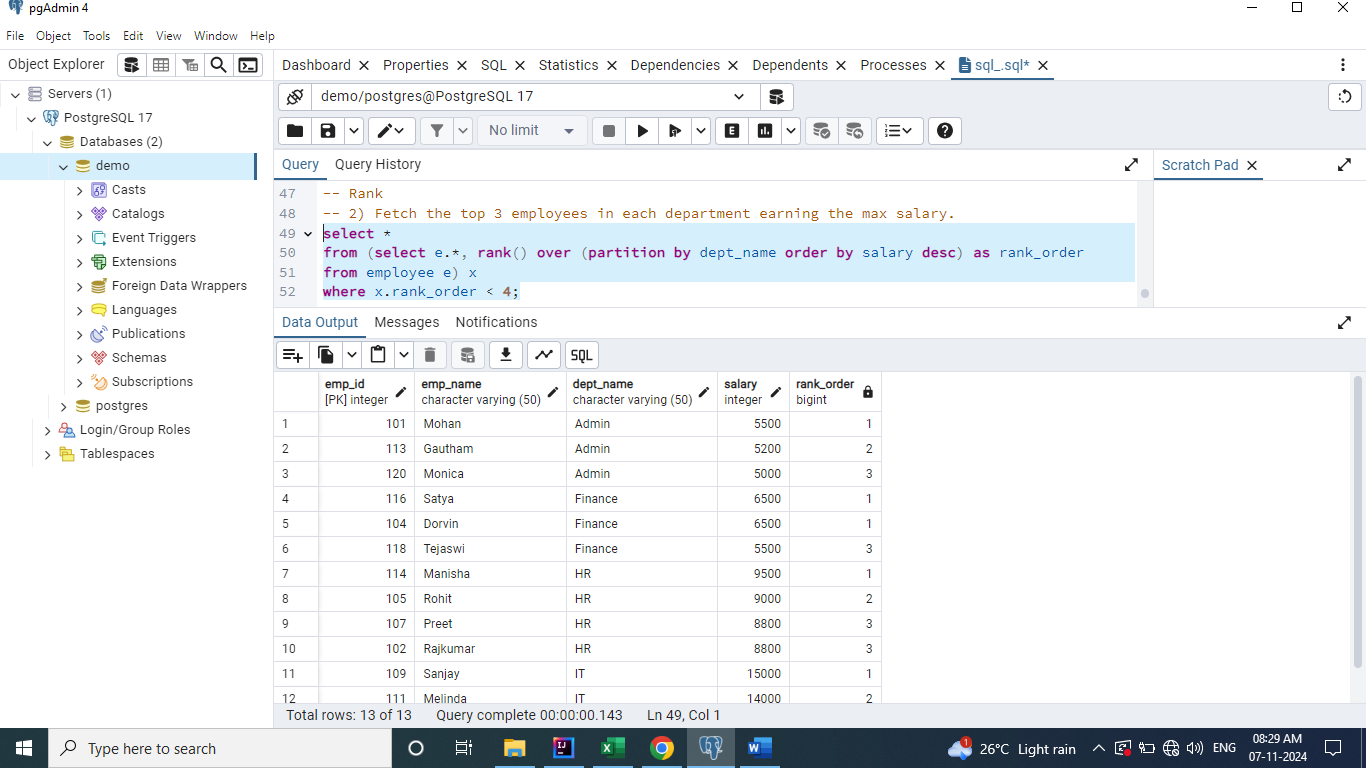
## 2) Fetch the top 3 employees in each department earning the max salary.

**select \***

**from (select e.\*, rank() over (partition by dept\_name order by salary desc) as rank\_order**

**from employee e) x**

**where x.rank\_order < 4;**



# 3. Dense Rank

**Rank will skip a value for every duplicate record whereas Dense Rank will not skip a value.**

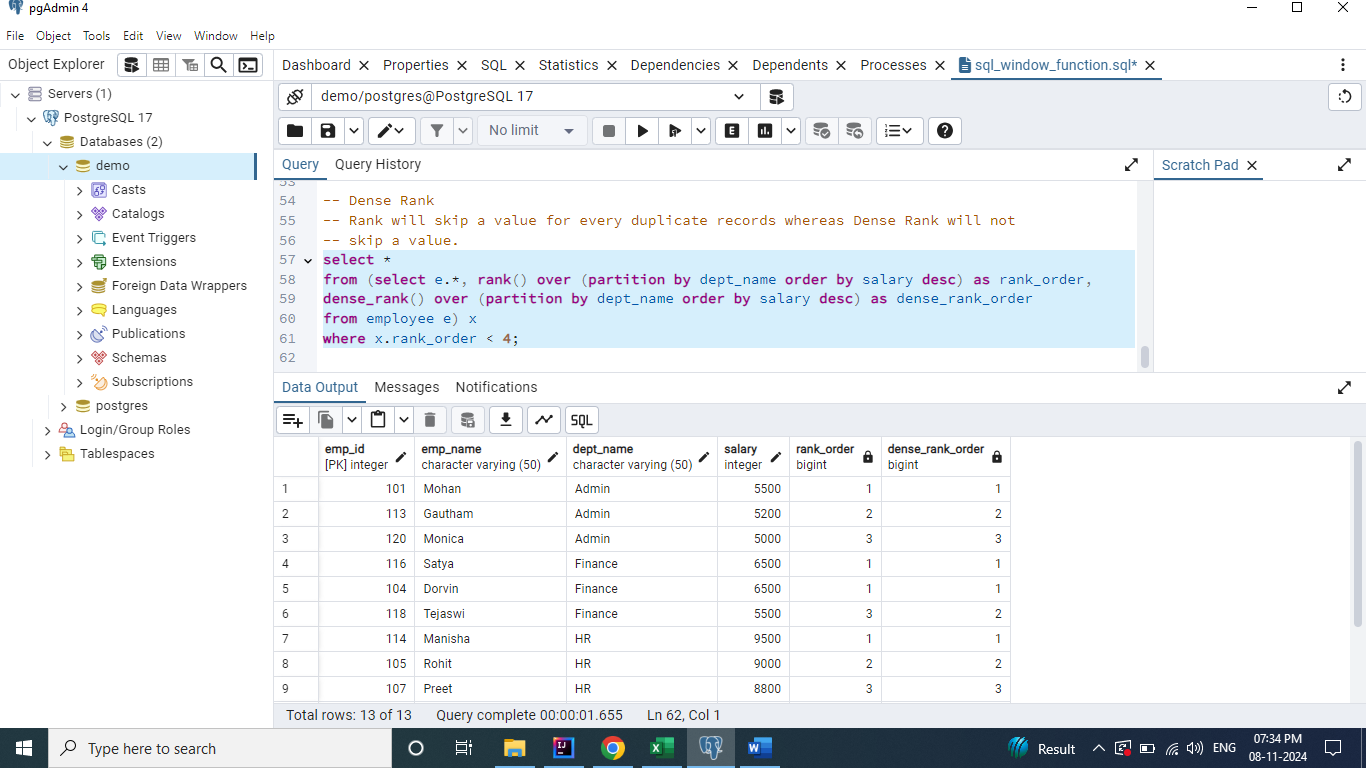
**select \***

**from (select e.\*, rank() over (partition by dept\_name order by salary desc) as rank\_order,**

**dense\_rank() over (partition by dept\_name order by salary desc) as dense\_rank\_order**

**from employee e) x**

**where x.rank\_order < 4;**



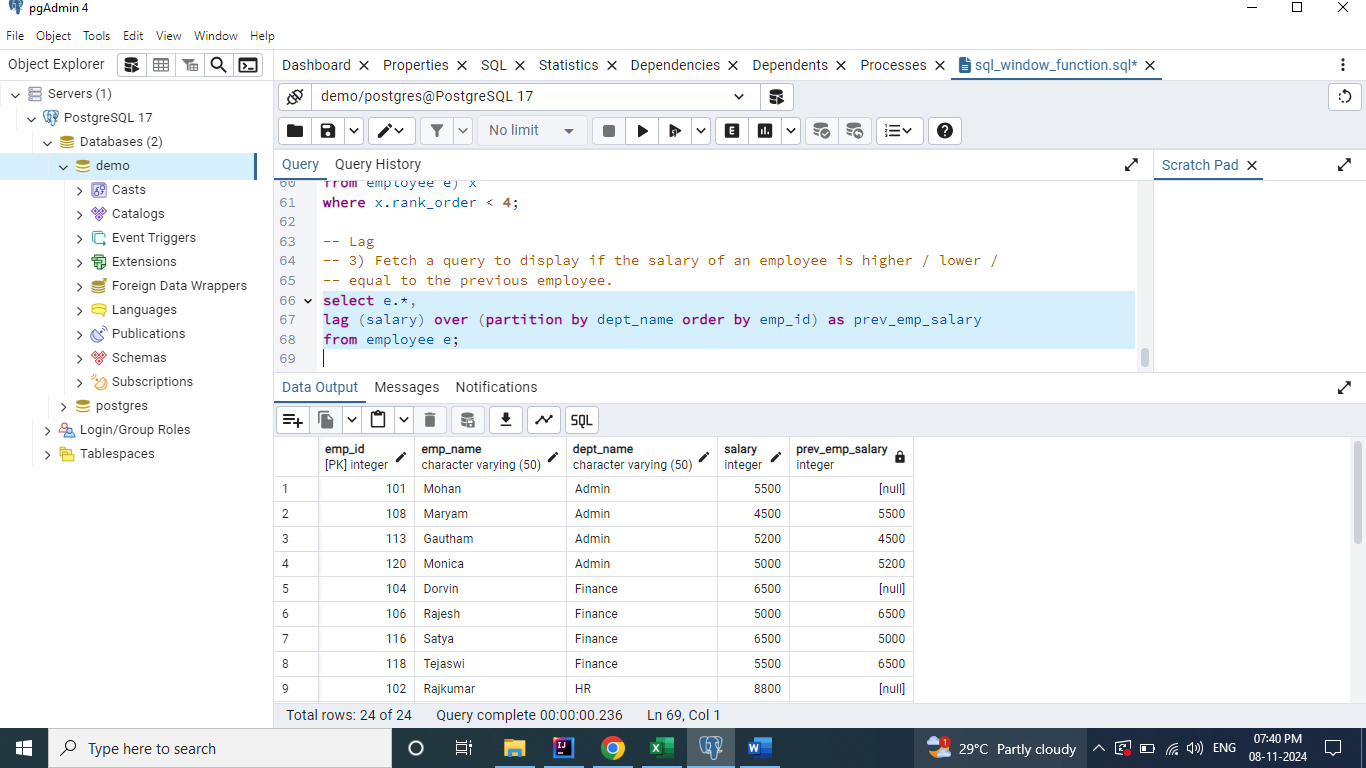
# 4. Lag

## 3) Fetch a query to display if the salary of an employee is higher / lower / equal to the previous employee.

**select e.\*,**

**lag (salary) over (partition by dept\_name order by emp\_id) as prev\_emp\_salary**

**from employee e;**



**lag(salary, 2, 0)**

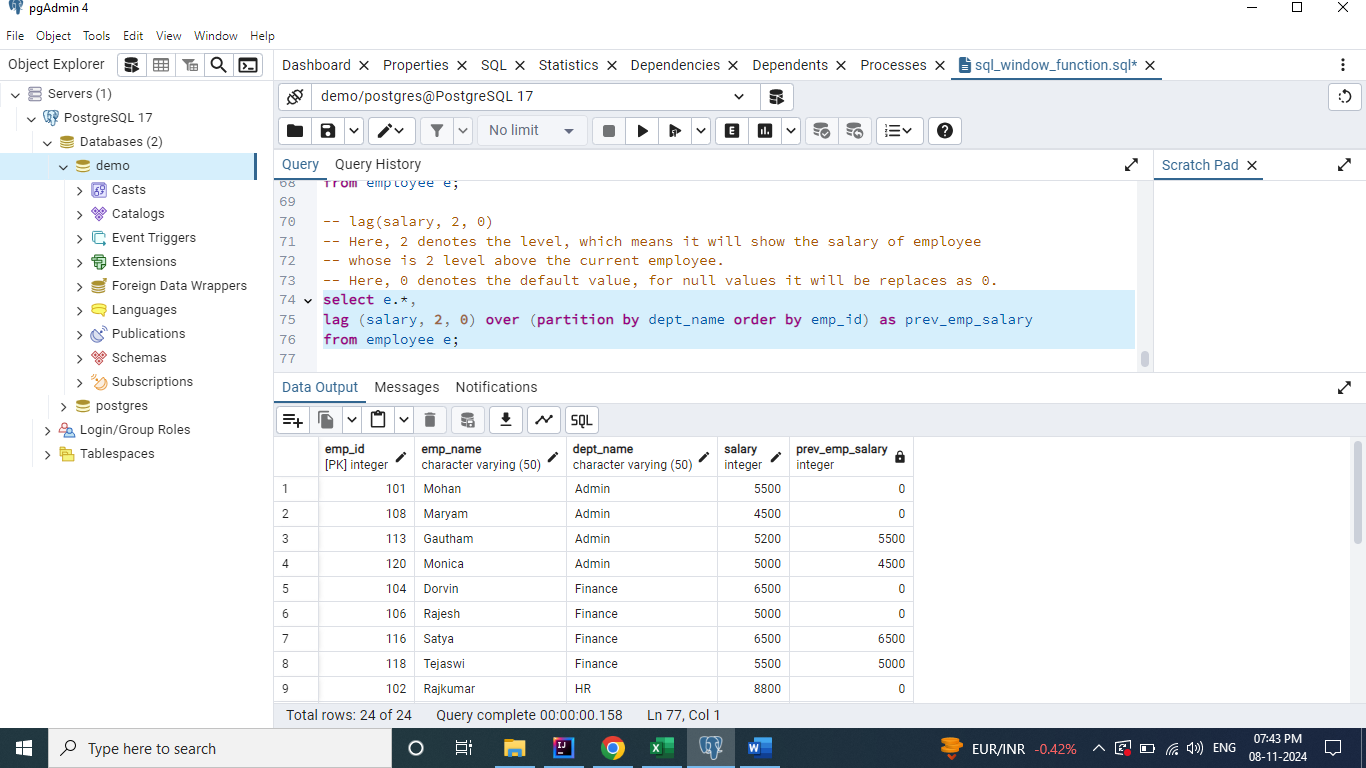
**Here, 2 denotes the level, which means it will show the salary of employee whose is 2 level above the current employee.**

**Here, 0 denotes the default value, for null values it will be replaces as 0.**

**select e.\*,**

**lag (salary, 2, 0) over (partition by dept\_name order by emp\_id) as prev\_emp\_salary**

**from employee e;**

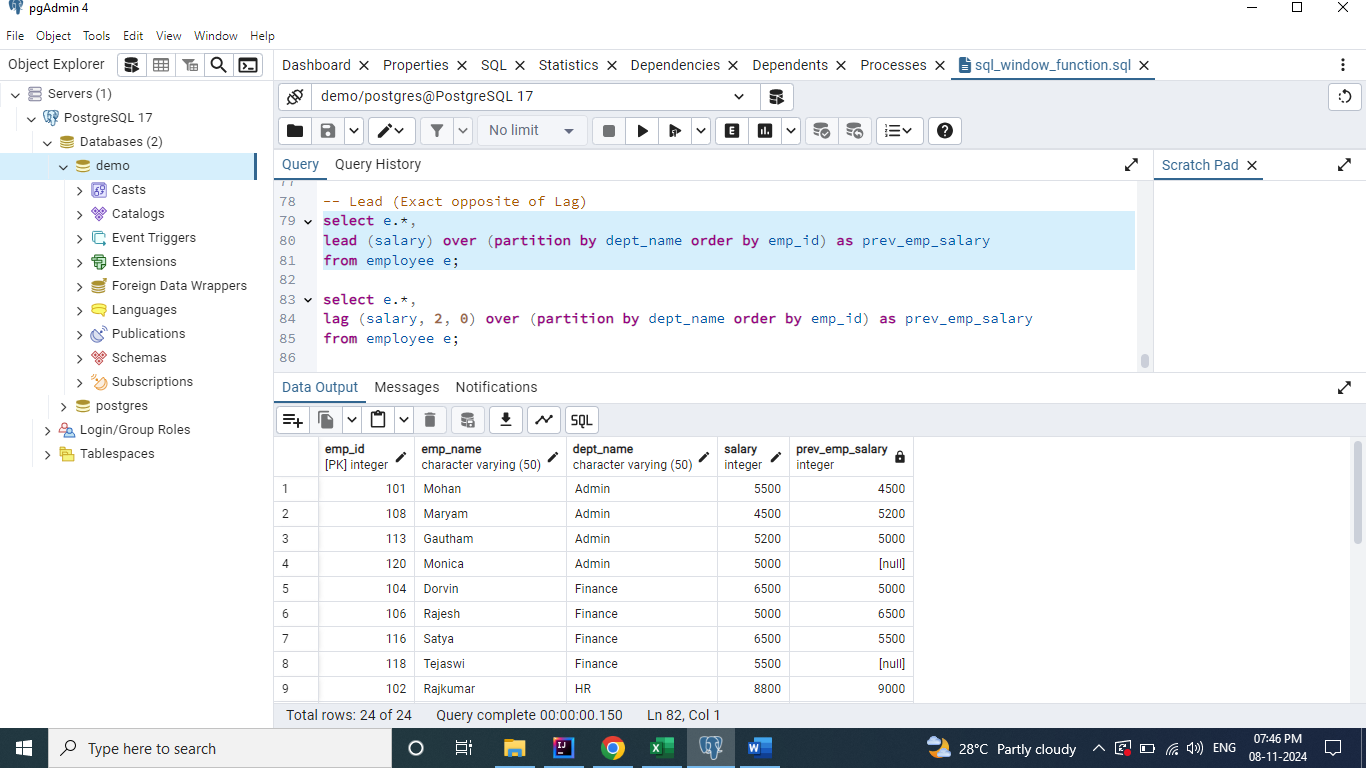


# 5. Lead (Exact opposite of Lag)

**select e.\*,**

**lead (salary) over (partition by dept\_name order by emp\_id) as next\_emp\_salary**

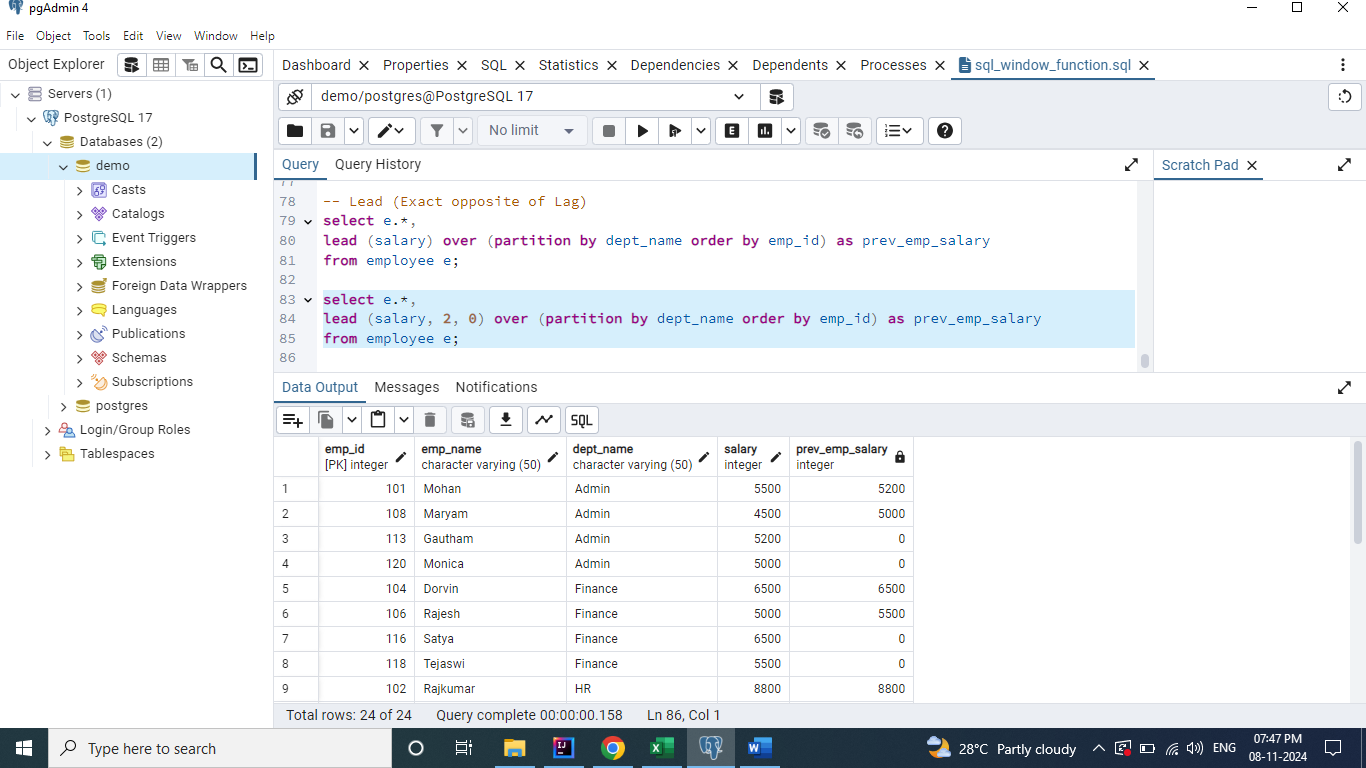
**from employee e;**



**select e.\*,**

**lead (salary, 2, 0) over (partition by dept\_name order by emp\_id) as next\_emp\_salary**

**from employee e;**



## Use Case

**select e.\*,**

**lag (salary) over (partition by dept\_name order by emp\_id) as prev\_emp\_salary,**

**case**

**when e.salary > lag(salary) over (partition by dept\_name order by emp\_id)**

**then 'Salary Higher than previous employee'**

**when e.salary < lag(salary) over (partition by dept\_name order by emp\_id)**

**then 'Salary lower than previous employee'**

**when e.salary = lag(salary) over (partition by dept\_name order by emp\_id)**

**then 'Salary same as previous employee'**

**end sal\_range**

**from employee e;**

