**Materialized View**

# Materialized View

**It is a database object.**

**It is created over a query.**

**Whenever we create MView, it will do 2 things.**

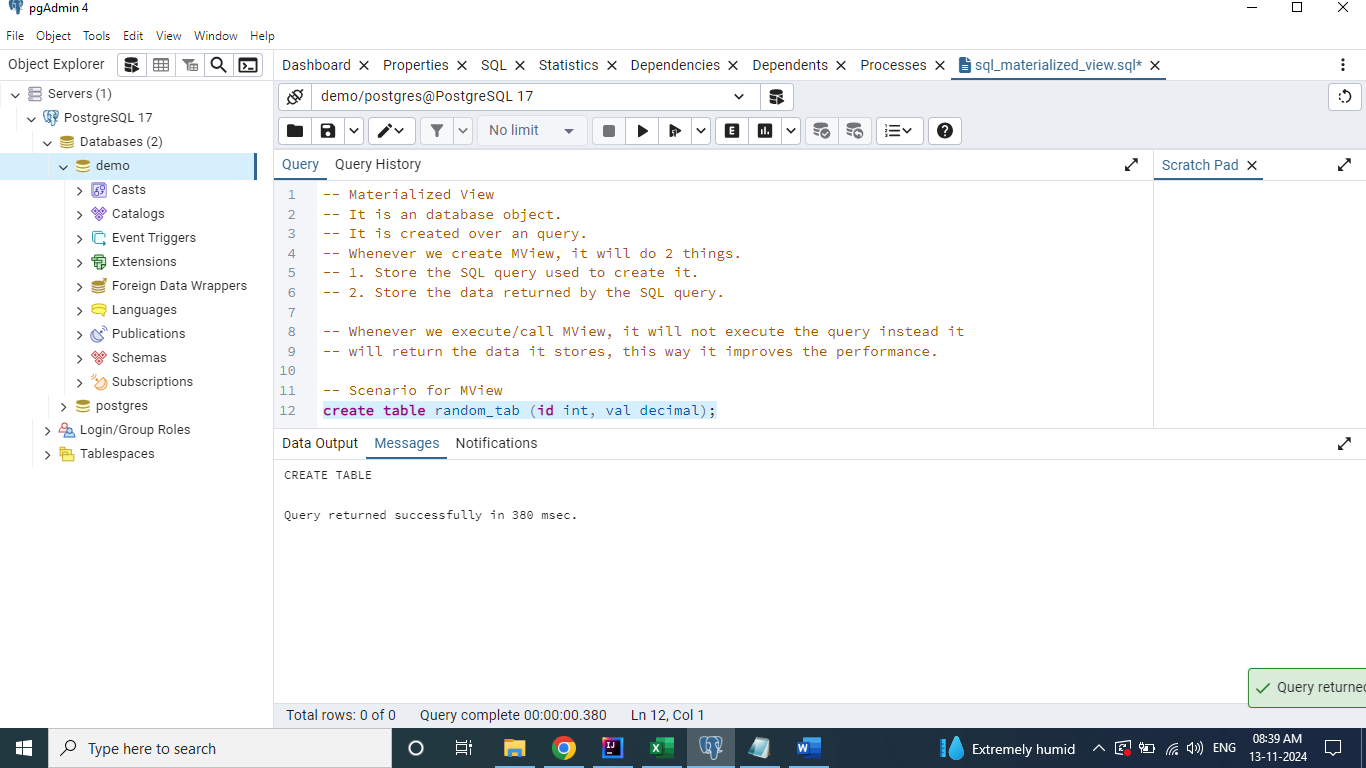
**1. Store the SQL query used to create it.**

**2. Store the data returned by the SQL query.**

**Whenever we execute/call MView, it will not execute the query instead it will return the data it stores, this way it improves the performance.**

# Scenario for MView

**create table random\_tab (id int, val decimal);**

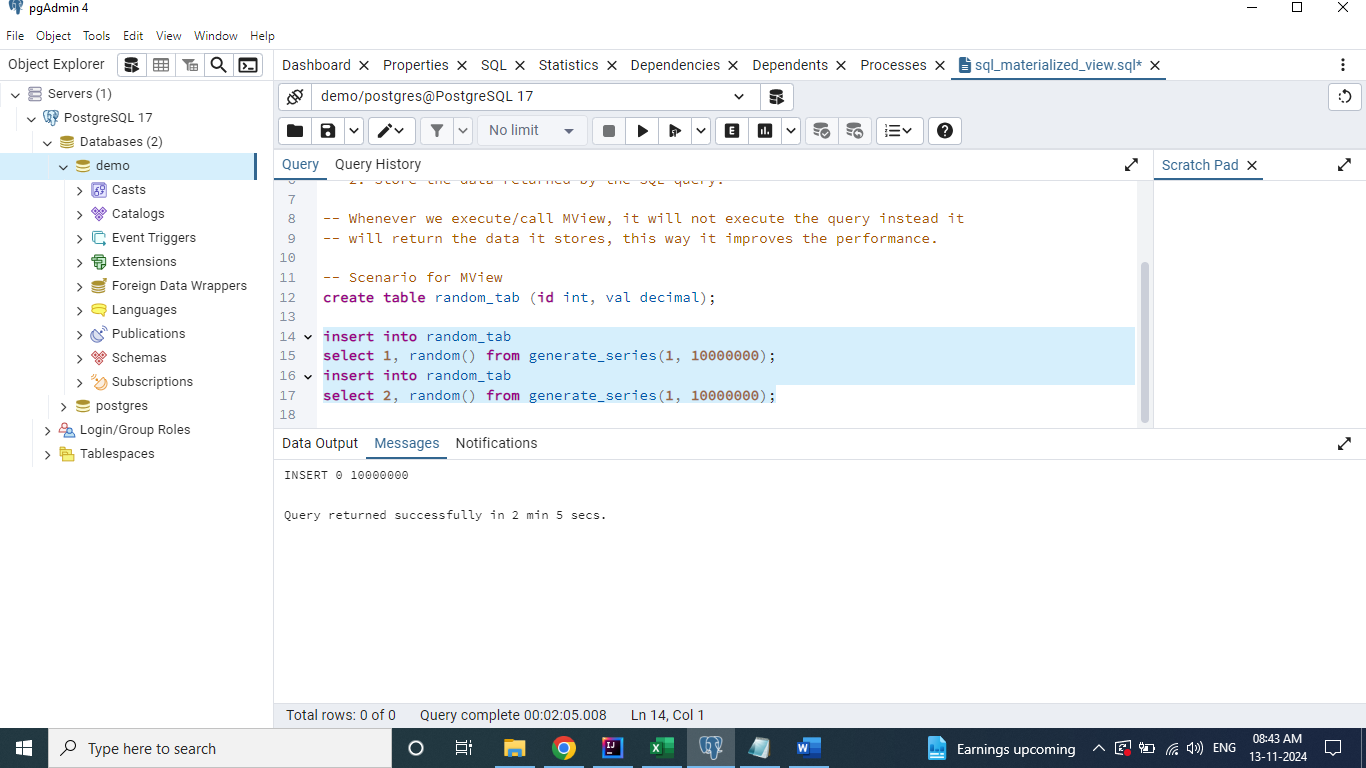


**insert into random\_tab**

**select 1, random() from generate\_series(1, 10000000);**

**insert into random\_tab**

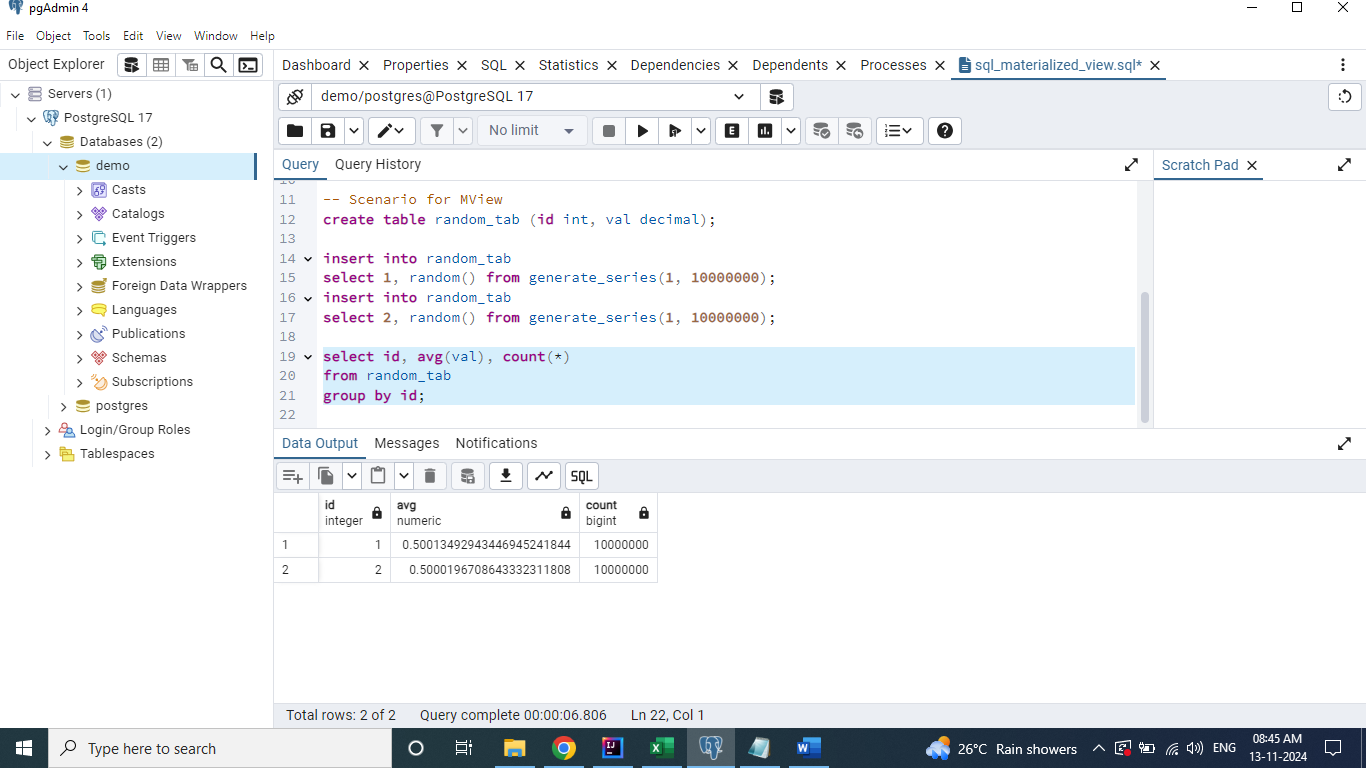
**select 2, random() from generate\_series(1, 10000000);**



**select id, avg(val), count(\*)**

**from random\_tab**

**group by id;**



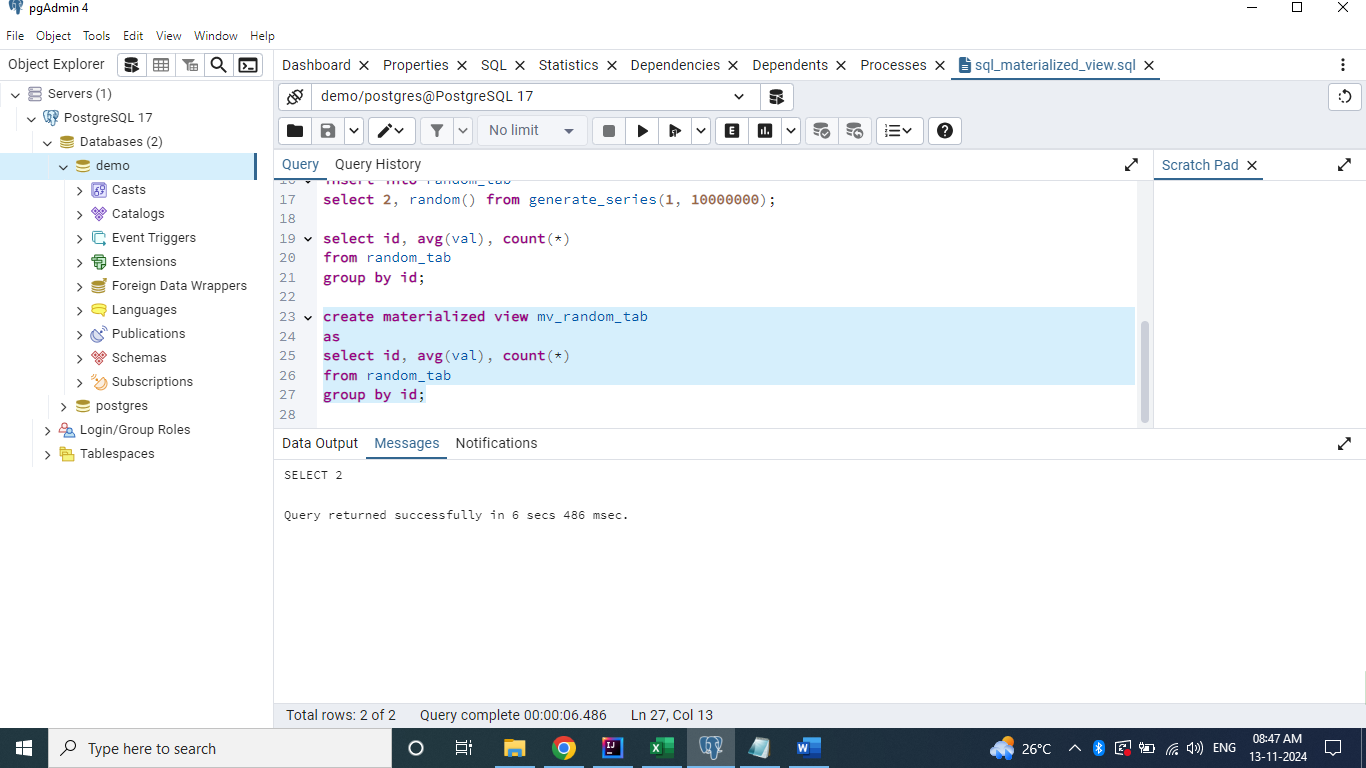
**create materialized view mv\_random\_tab**

**as**

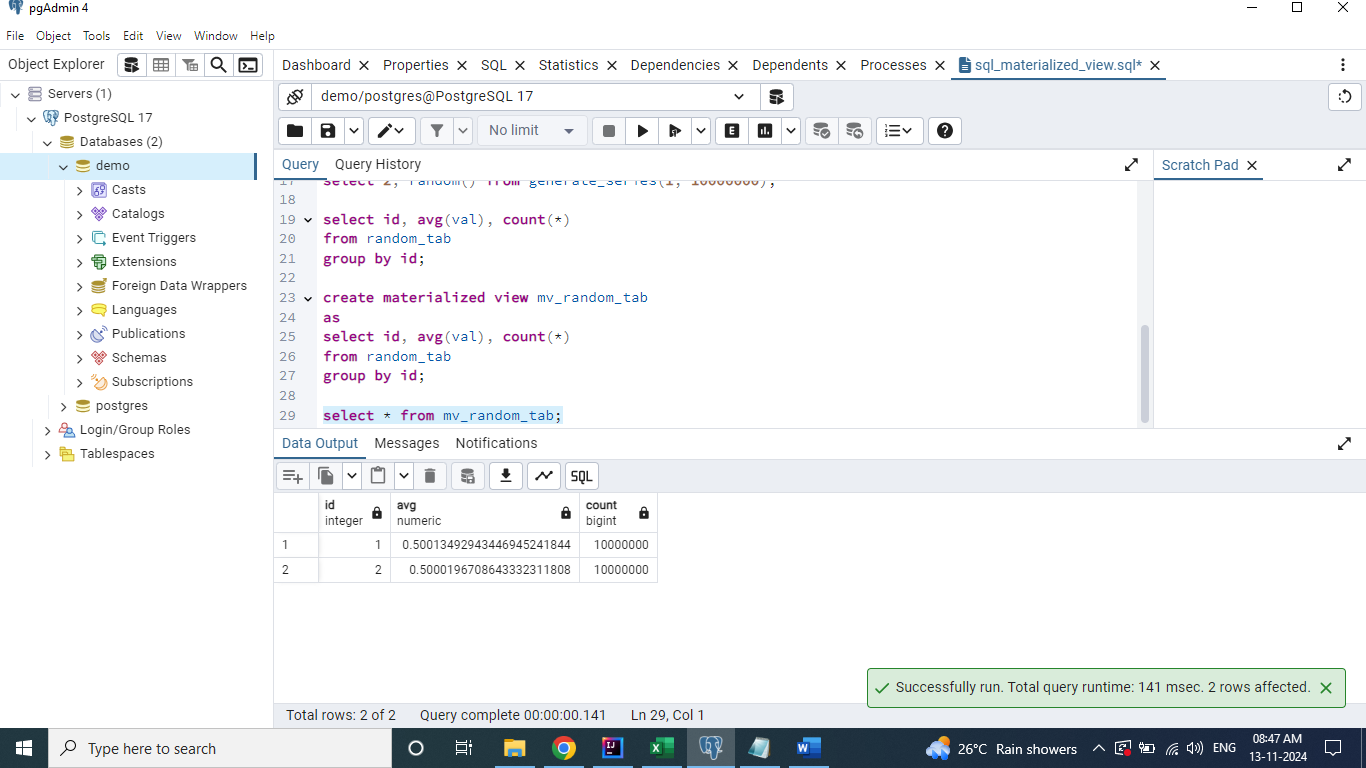
**select id, avg(val), count(\*)**

**from random\_tab**

**group by id;**



**select \* from mv\_random\_tab;**



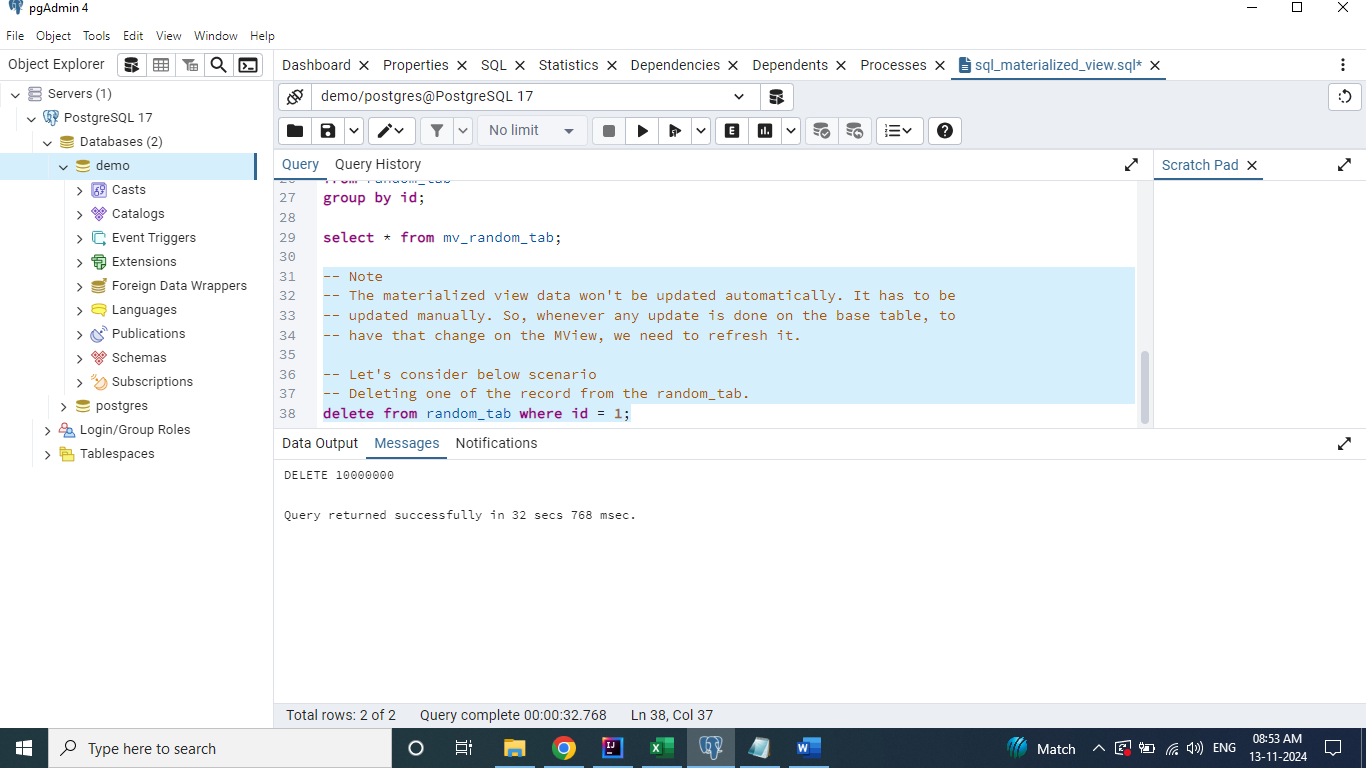
# Note

**The materialized view data won't be updated automatically. It has to be updated manually. So, whenever any update is done on the base table, to have that change on the MView, we need to refresh it.**

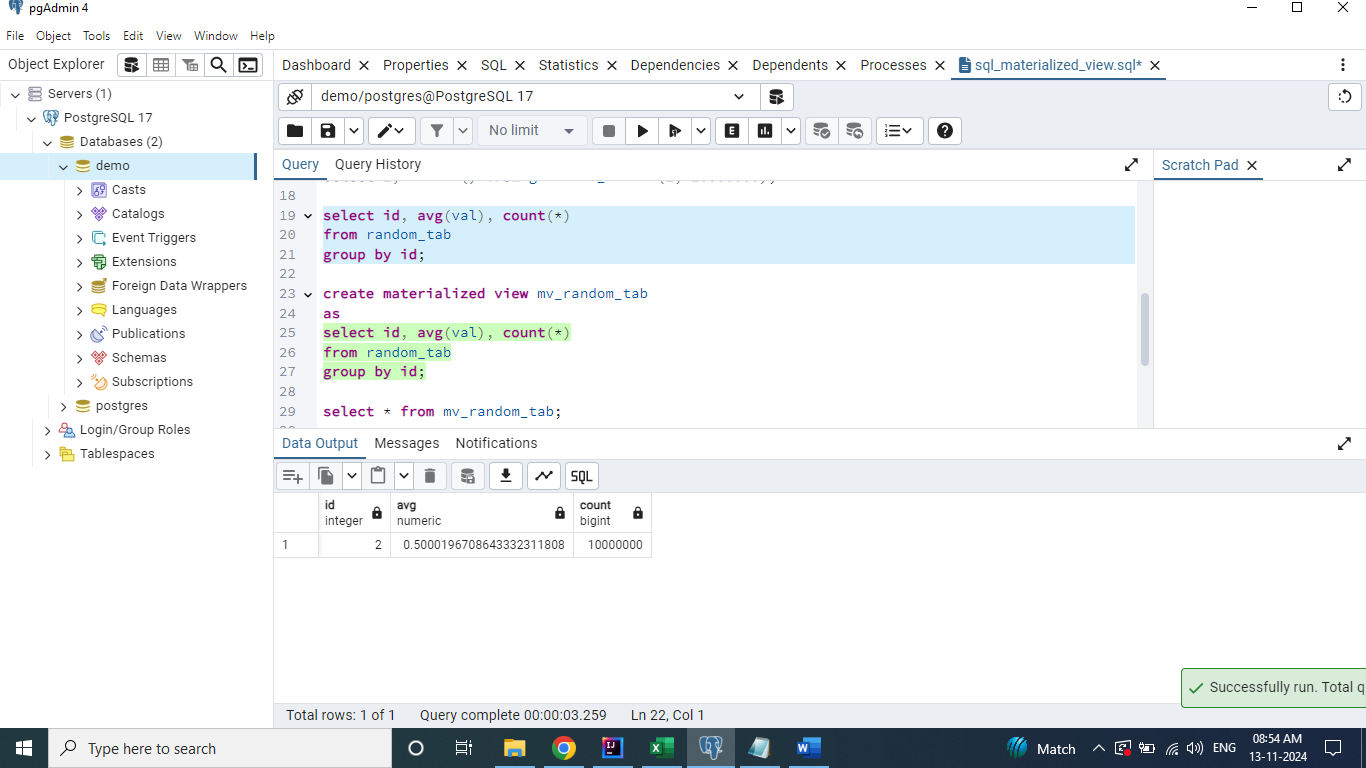
## Let's consider below scenario

**Deleting one of the records from the random\_tab.**

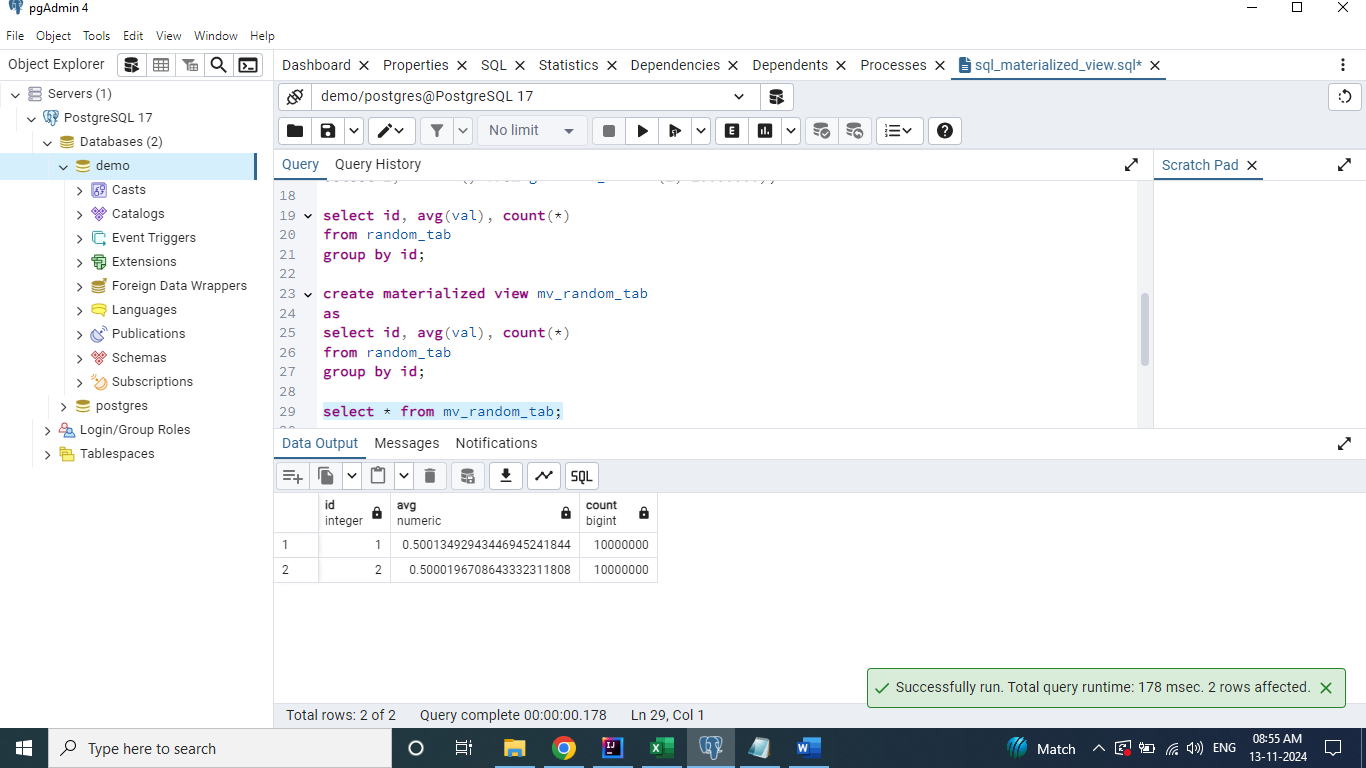
**delete from random\_tab where id = 1;**



**After deletion, base table has only one record.**

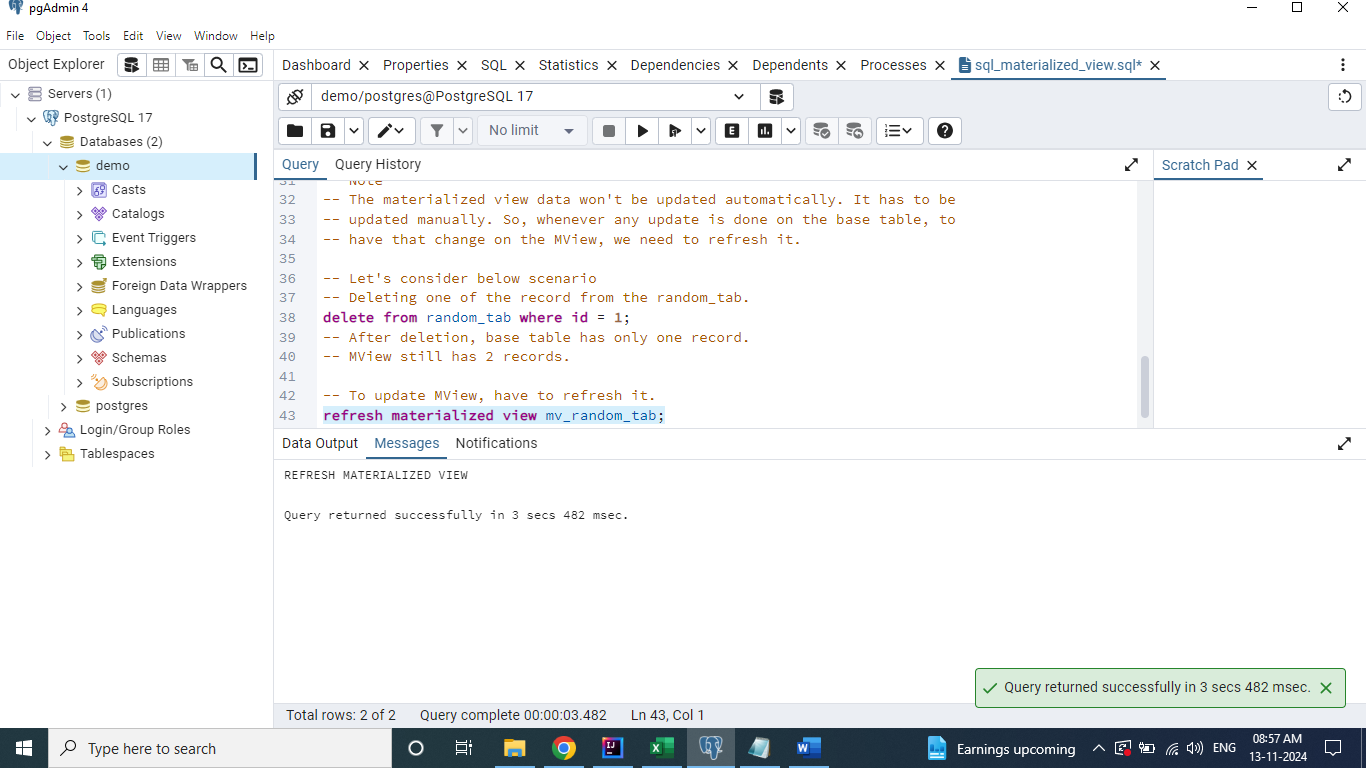


**MView still has 2 records.**



# Refresh MView

**refresh materialized view mv\_random\_tab;**



**After refresh**

