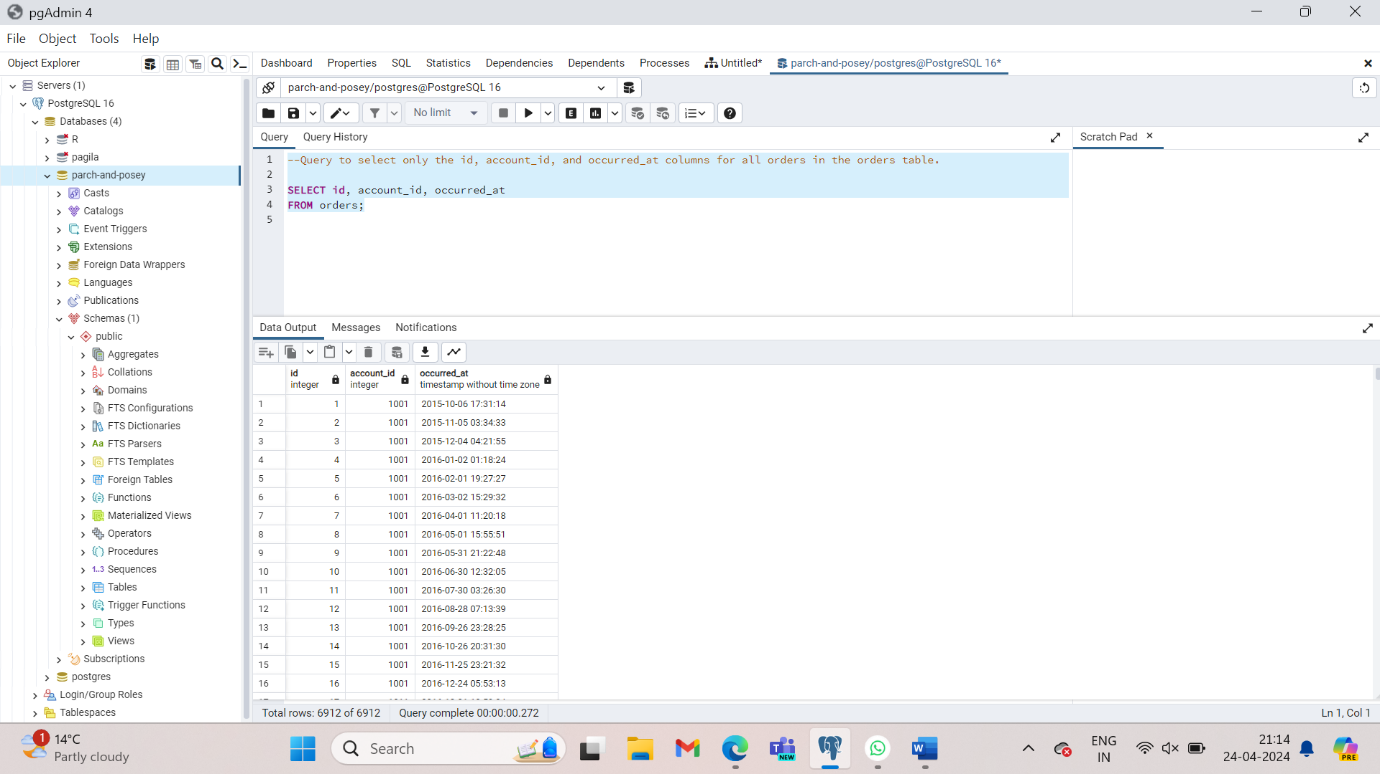
**Basic SQL**

1. Query to select only the id, account\_id, and occurred\_at columns for all orders in the **orders** table.

SELECT id, account\_id, occurred\_at

FROM orders;

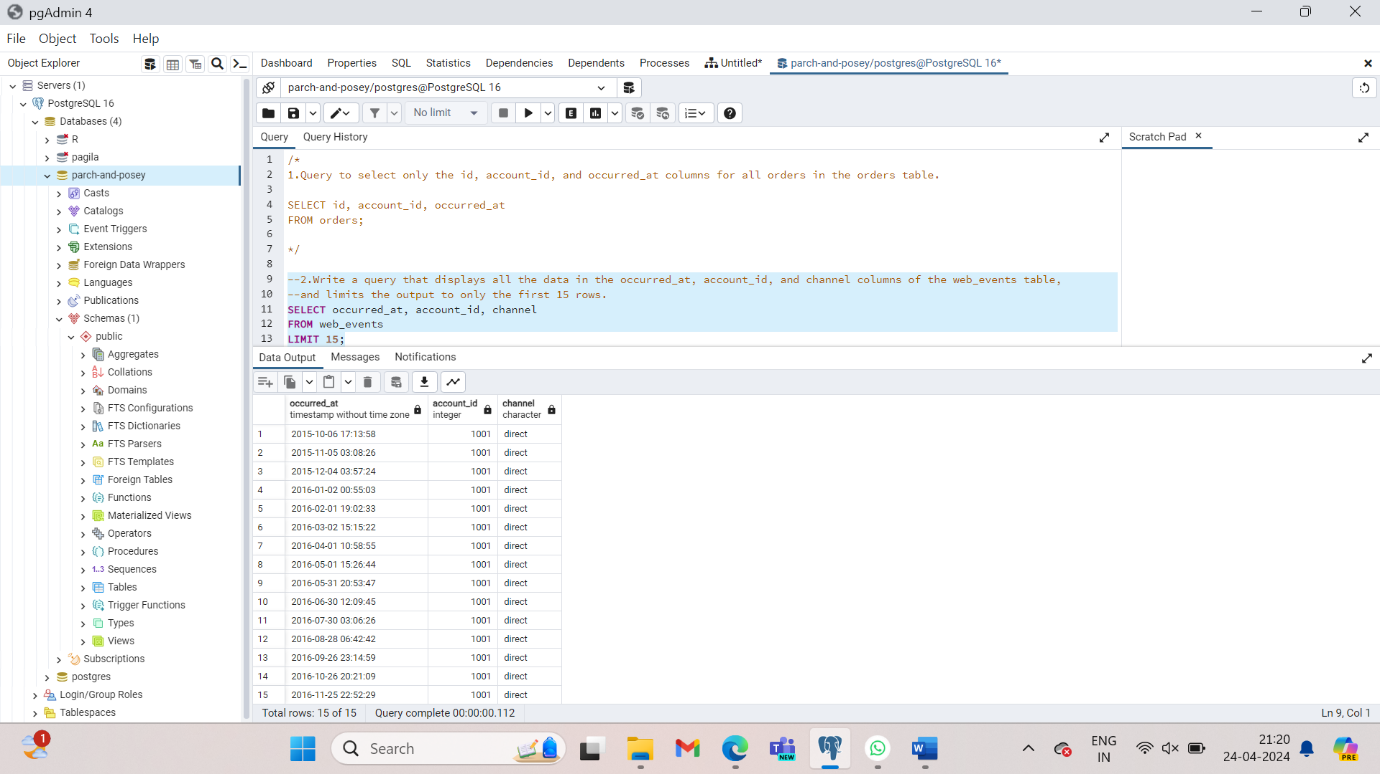


1. Write a query that displays all the data in the occurred\_at, account\_id, and channel columns of the web\_events table, and limits the output to only the first 15 rows.

SELECT occurred\_at, account\_id, channel

FROM web\_events

LIMIT 15;

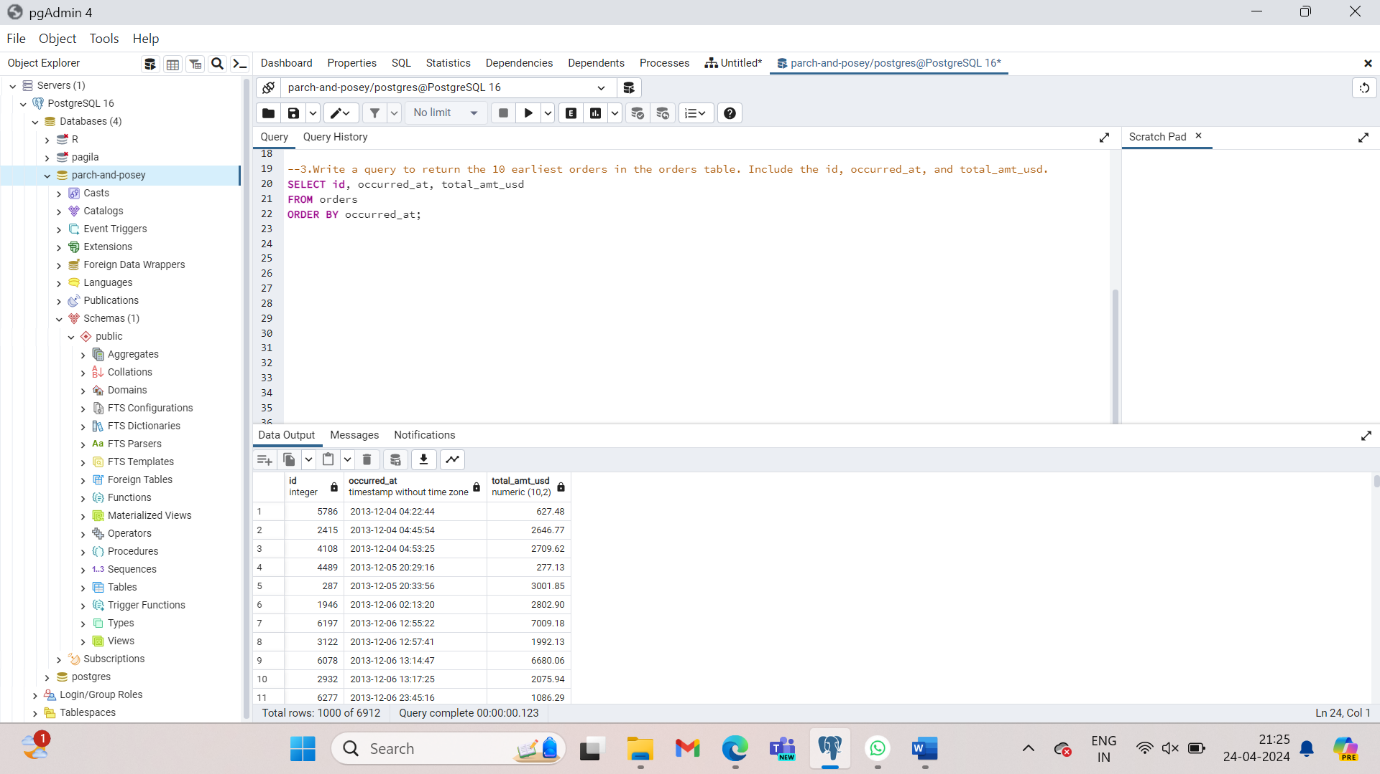


1. Write a query to return the 10 earliest orders in the **orders** table. Include the id, occurred\_at, and total\_amt\_usd.

SELECT id, occurred\_at, total\_amt\_usd

FROM orders

ORDER BY occurred\_at;



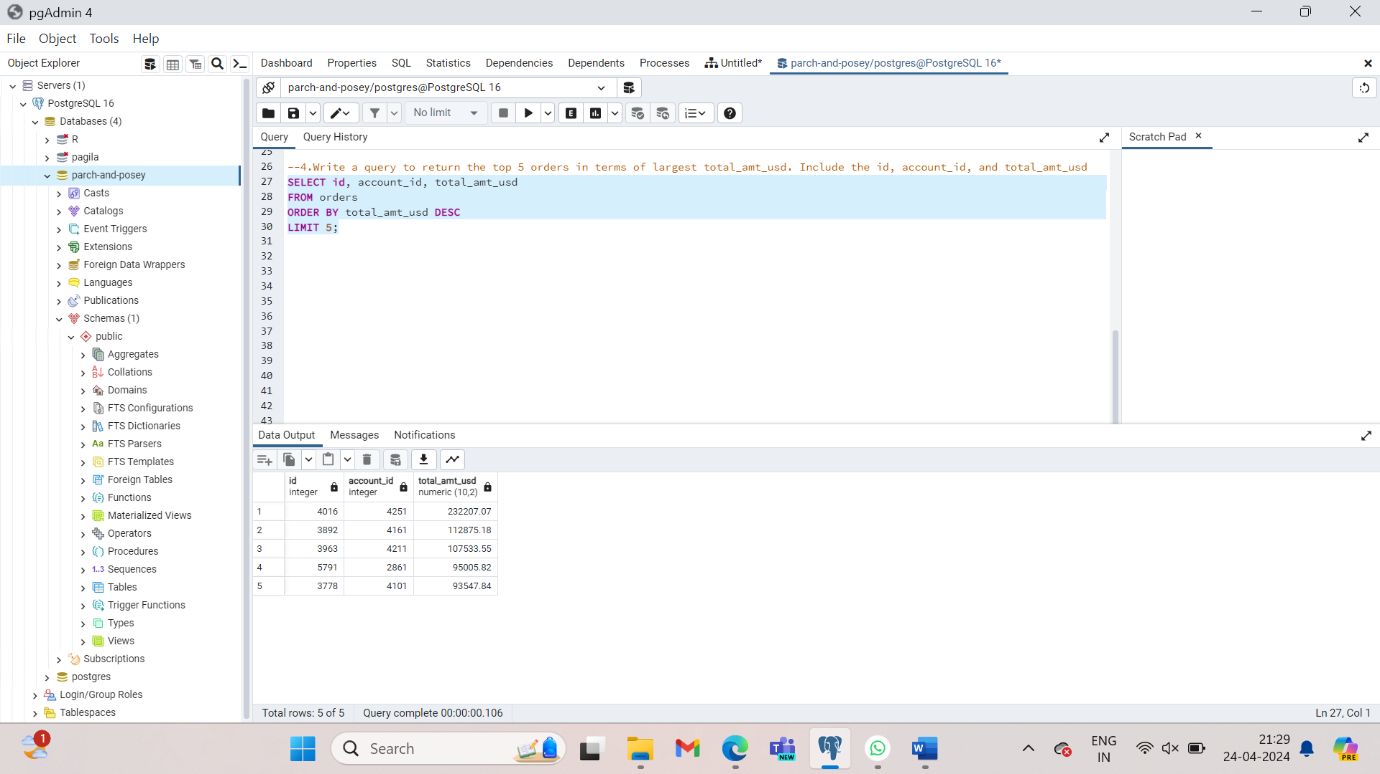
1. Write a query to return the top 5 **orders** in terms of largest total\_amt\_usd. Include the id, account\_id, and total\_amt\_usd

SELECT id, account\_id, total\_amt\_usd

FROM orders

ORDER BY total\_amt\_usd DESC

LIMIT 5;



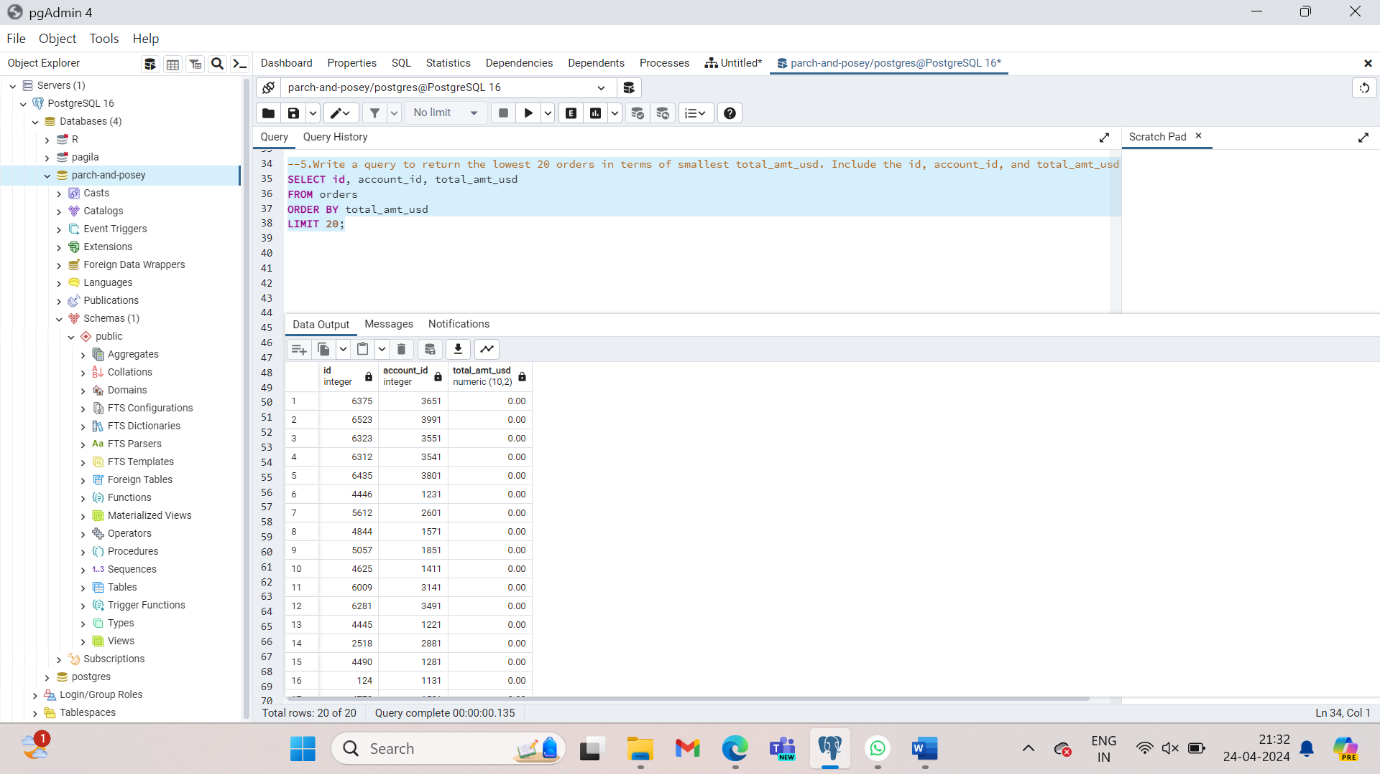
1. Write a query to return the lowest 20 **orders** in terms of smallest total\_amt\_usd. Include the id, account\_id, and total\_amt\_usd.

SELECT id, account\_id, total\_amt\_usd

FROM orders

ORDER BY total\_amt\_usd

LIMIT 20;

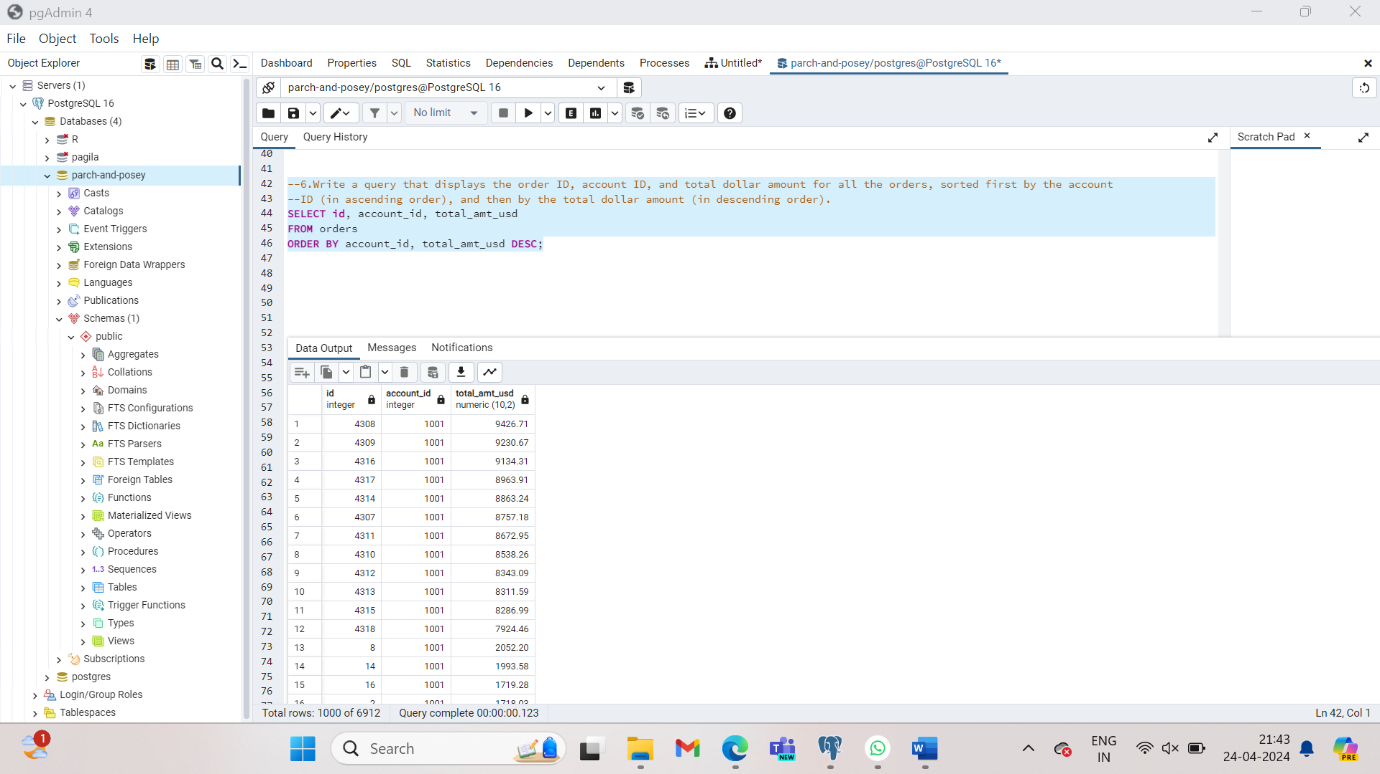


1. Write a query that displays the order ID, account ID, and total dollar amount for all the orders, sorted first by the account ID (in ascending order), and then by the total dollar amount (in descending order).

SELECT id, account\_id, total\_amt\_usd

FROM orders

ORDER BY account\_id, total\_amt\_usd DESC;

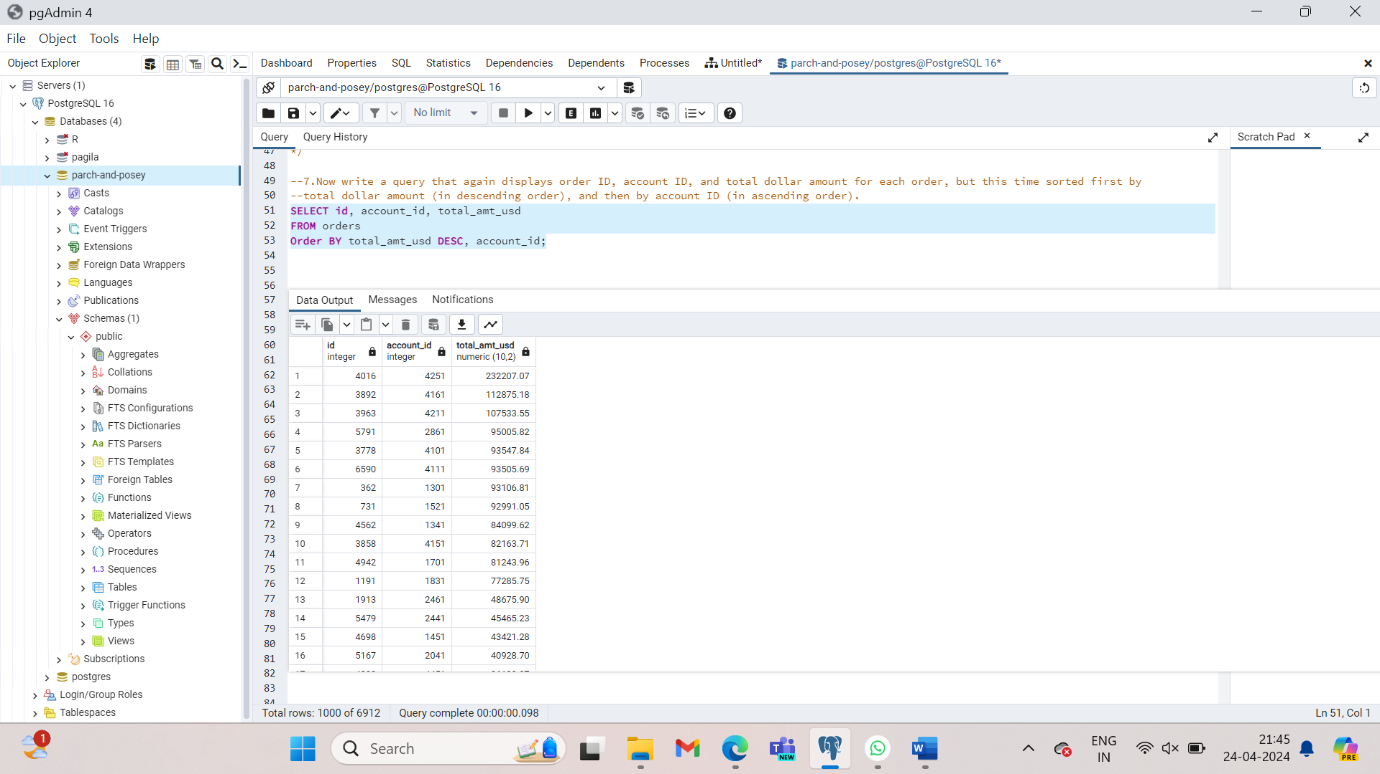


1. Now write a query that again displays order ID, account ID, and total dollar amount for each order, but this time sorted first by total dollar amount (in descending order), and then by account ID (in ascending order).

SELECT id, account\_id, total\_amt\_usd

FROM orders

Order BY total\_amt\_usd DESC, account\_id;



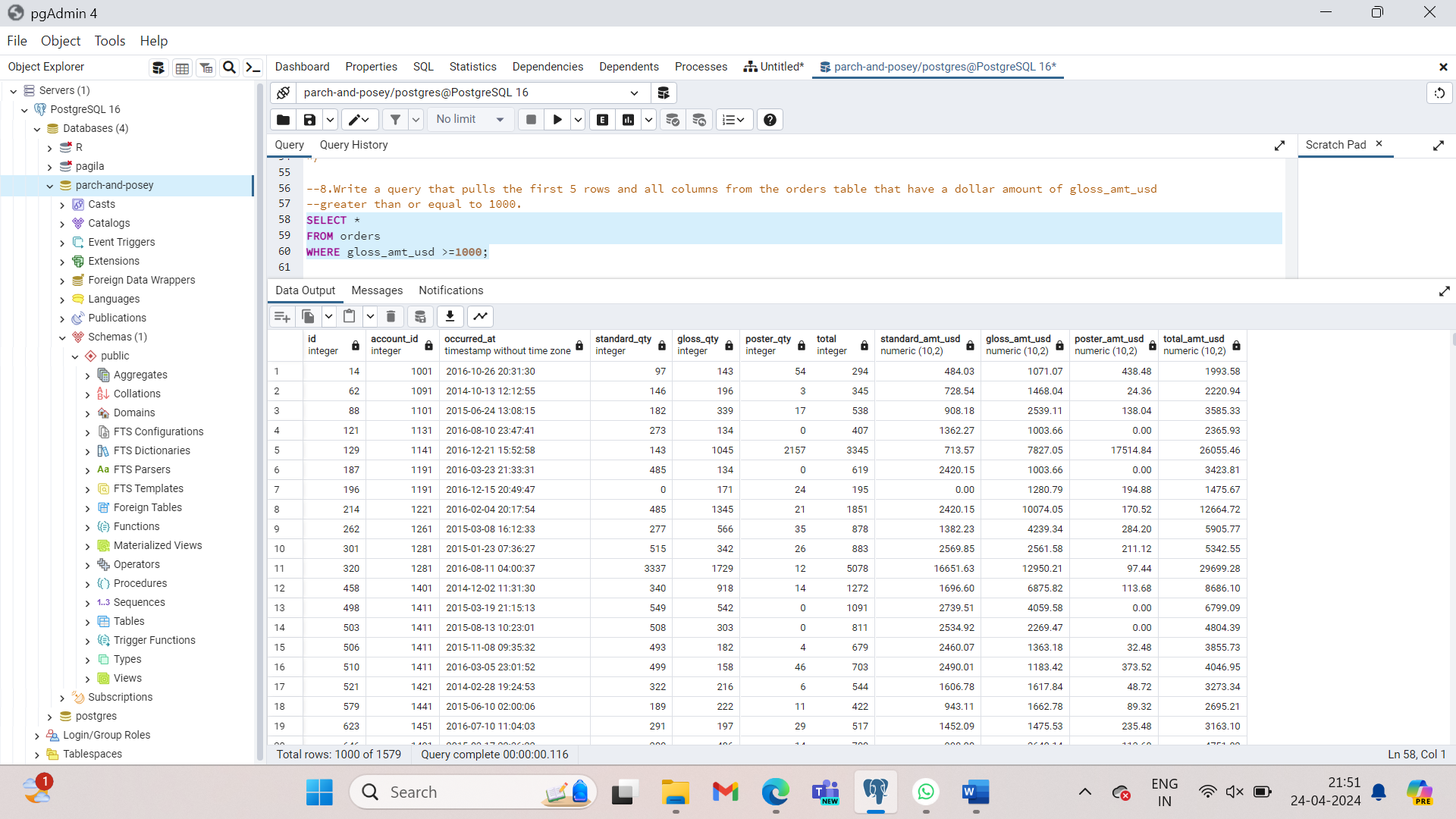
**In query #6, all of the orders for each account ID are grouped together, and then within each of those groupings, the orders appear from the greatest order amount to the least. In query #7, since we sorted by the total dollar amount first, the orders appear from greatest to least regardless of which account ID they were from. Then they are sorted by account ID next. (The secondary sorting by account ID is difficult to see here, since only if there were two orders with equal total dollar amounts would there need to be any sorting by account ID.)**

1. Write a query that pulls the first 5 rows and all columns from the **orders** table that have a dollar amount of gloss\_amt\_usd greater than or equal to 1000.

SELECT \*

FROM orders

WHERE gloss\_amt\_usd >=1000;



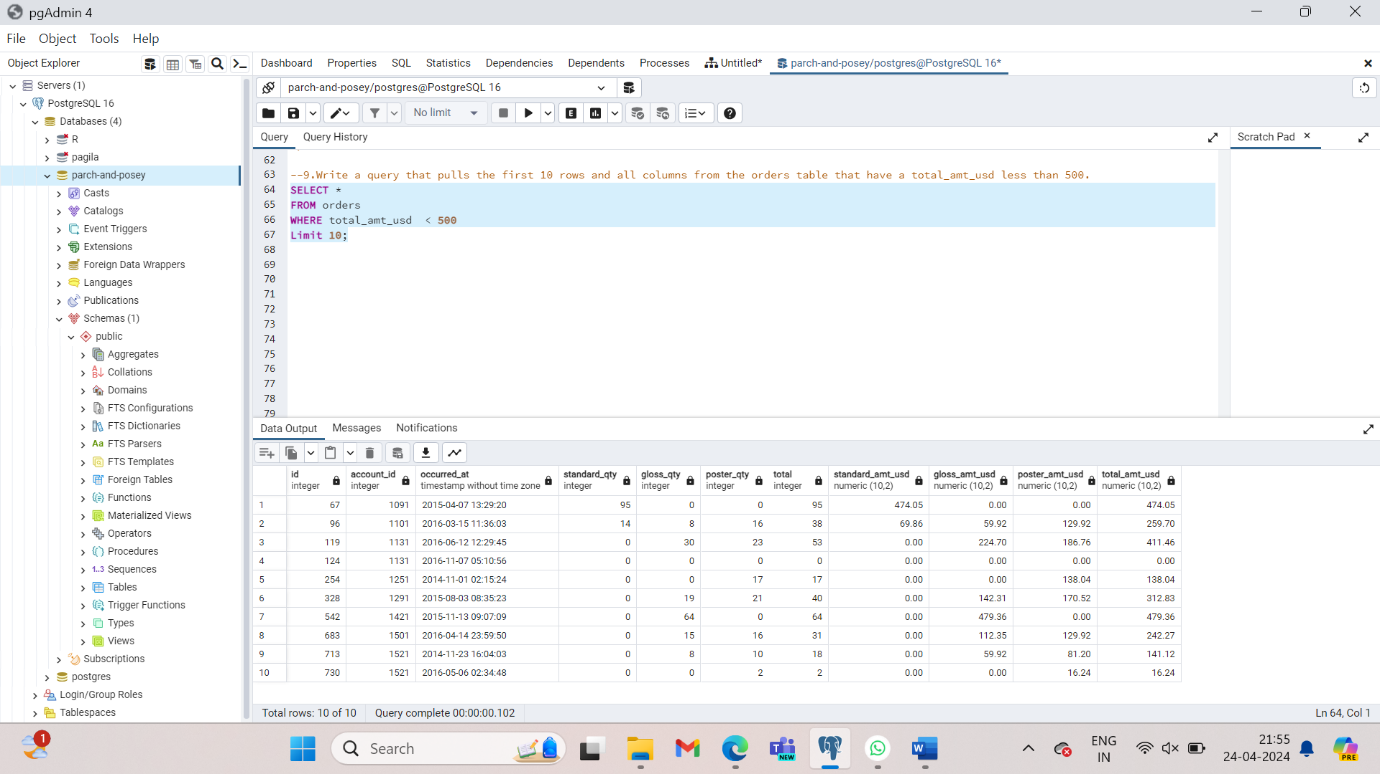
1. Write a query that pulls the first 10 rows and all columns from the **orders** table that have a total\_amt\_usd less than 500.

SELECT \*

FROM orders

WHERE total\_amt\_usd < 500

Limit 10;



When we are using **WHERE** with non-numeric data fields, we use the **LIKE**, **NOT**, or **IN** operators.

1. Filter the accounts table to include the company name, website, and the primary point of contact (primary\_poc) just for the Exxon Mobil company in the **accounts** table.

SELECT name, website, primary\_poc

FROM accounts

WHERE name = ‘Exxon Mobil’;

A screenshot of a computer

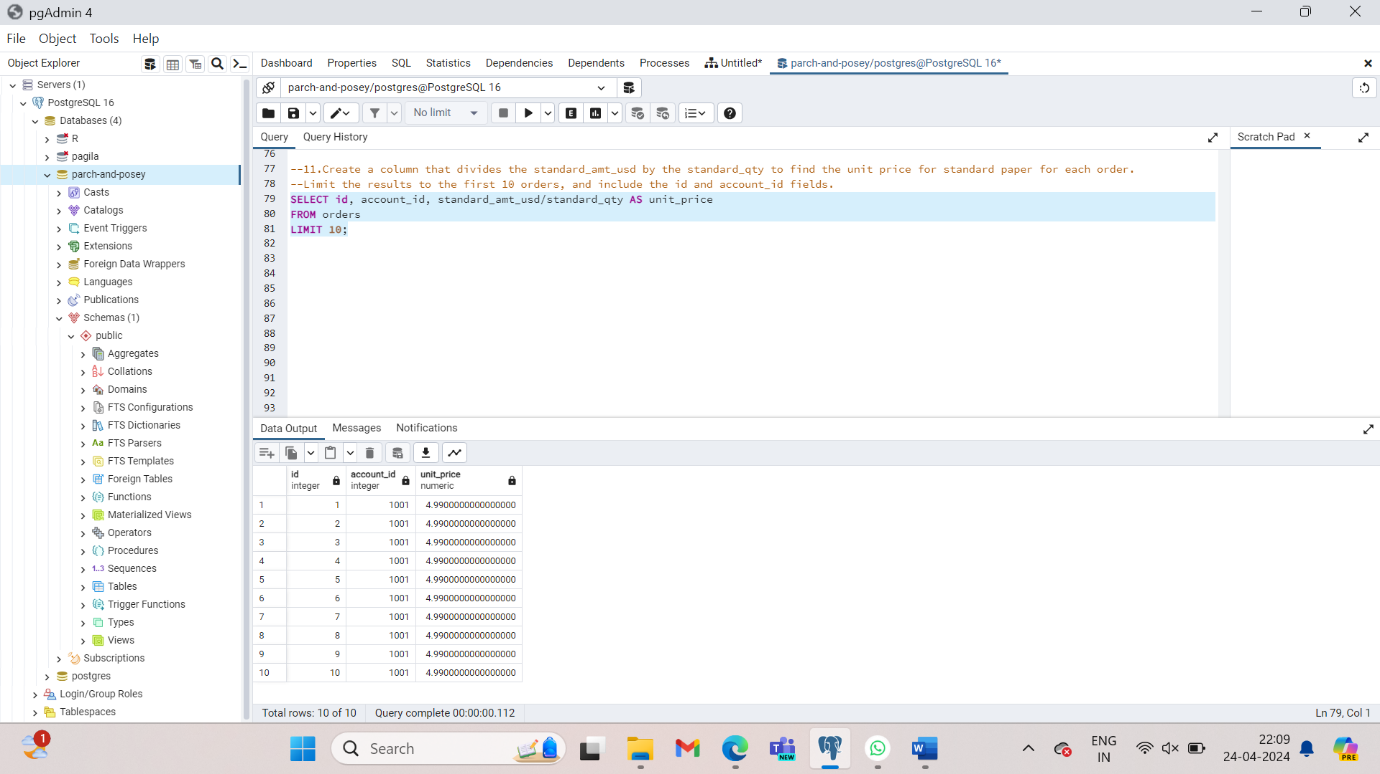
Description automatically generated

1. Create a column that divides the standard\_amt\_usd by the standard\_qty to find the unit price for standard paper for each order. Limit the results to the first 10 orders, and include the id and account\_id fields.

SELECT id, account\_id, standard\_amt\_usd/standard\_qty **AS** unit\_price

FROM orders

LIMIT 10;



1. Write a query that finds the percentage of revenue that comes from poster paper for each order. You will need to use only the columns that end with \_usd. (Try to do this without using the total column.) Display the id and account\_id fields also. Limit the output to 10 rows.

SELECT id, account\_id, poster\_amt\_usd/(poster\_amt\_usd+standard\_amt\_usd+gloss\_amt\_usd) AS post\_per

FROM orders

LIMIT 10;

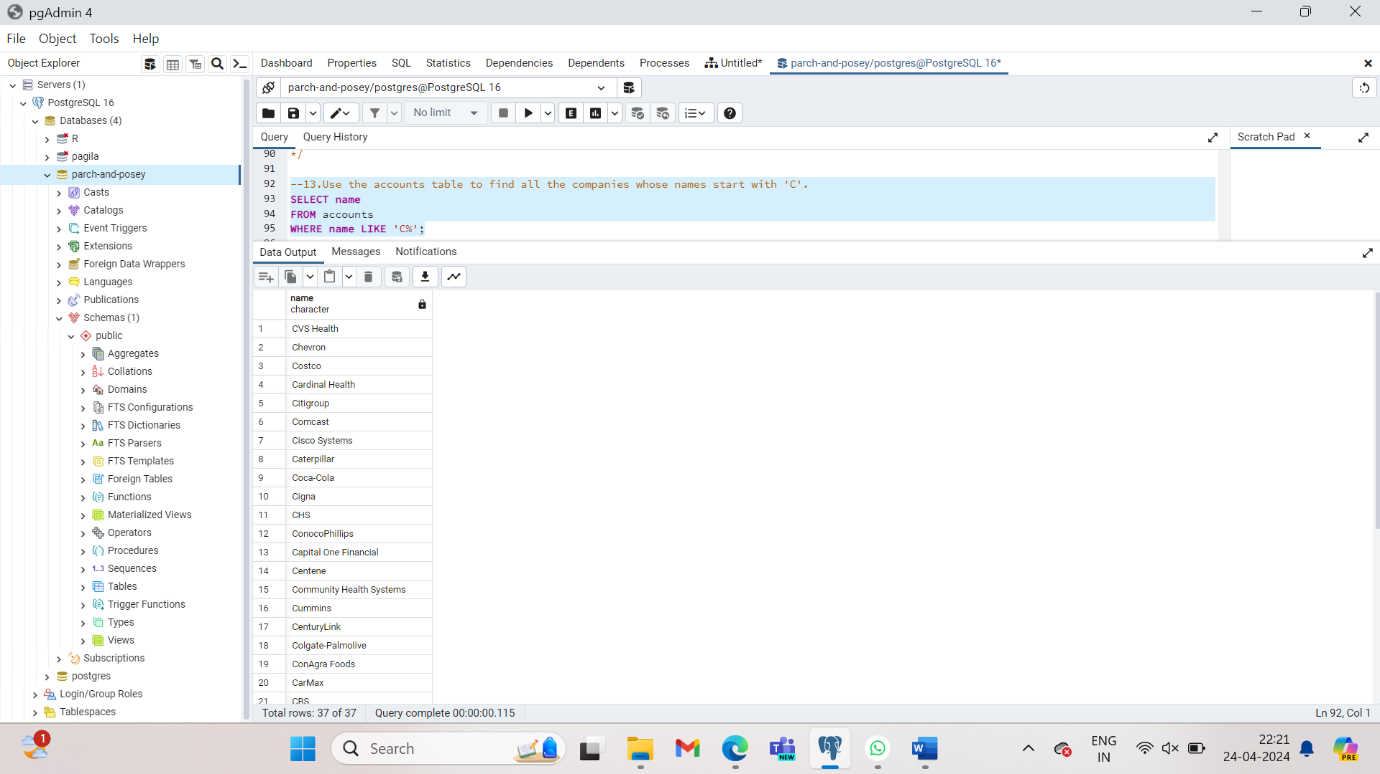


1. Use the **accounts** table to find all the companies whose names start with 'C'.

SELECT name

FROM accounts

WHERE name LIKE ‘C%’;

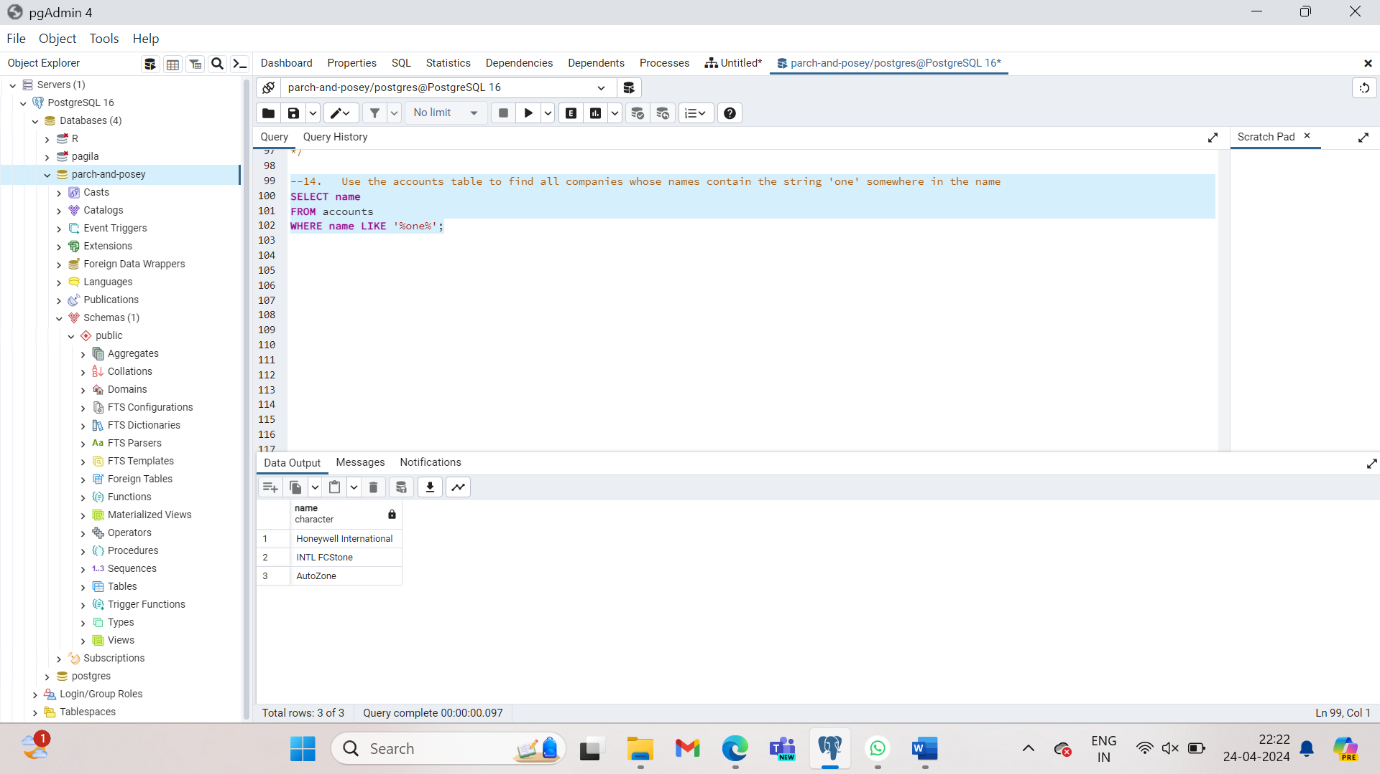


1. Use the **accounts** table to find all companies whose names contain the string 'one' somewhere in the name

SELECT name

FROM accounts

WHERE name LIKE ‘%one%’;

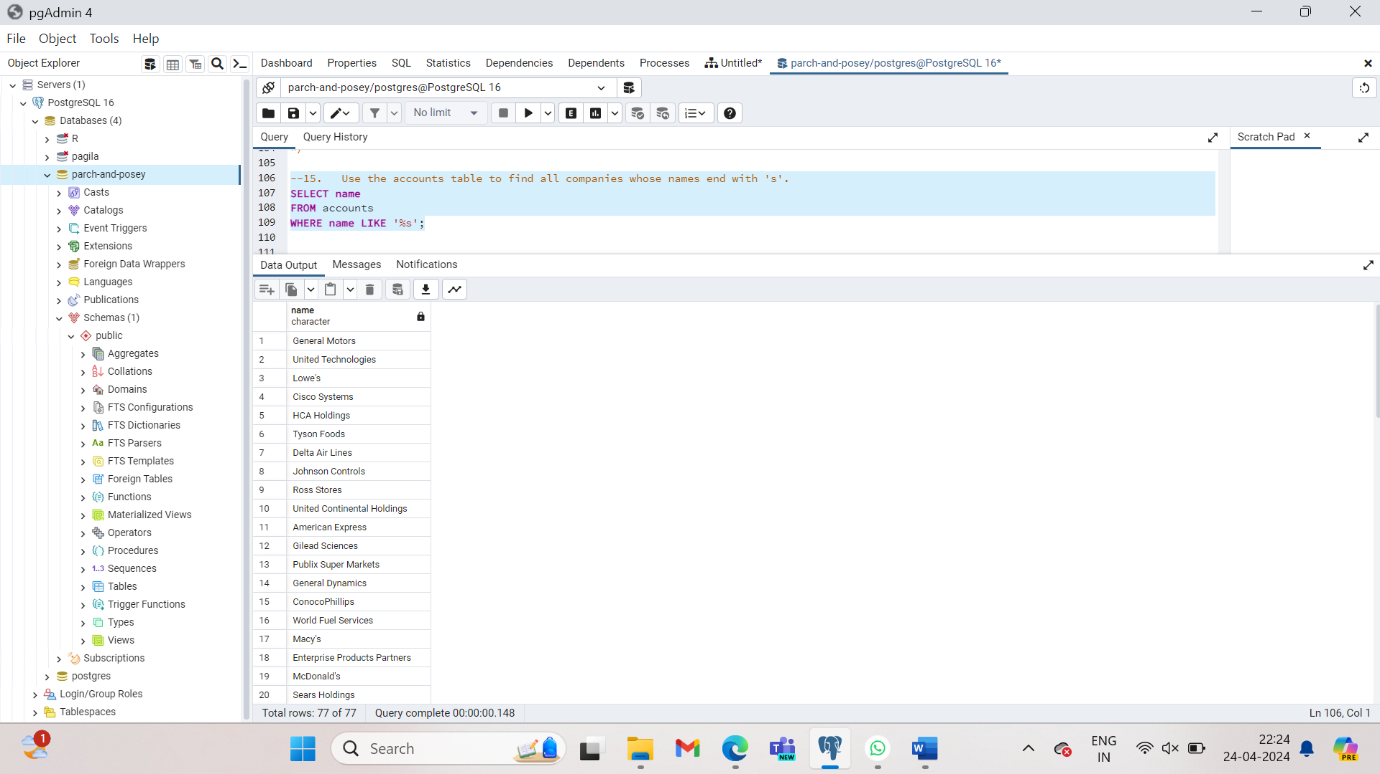


1. Use the **accounts** table to find all companies whose names end with 's'.

SELECT name

FROM accounts

WHERE name LIKE ‘%s’;

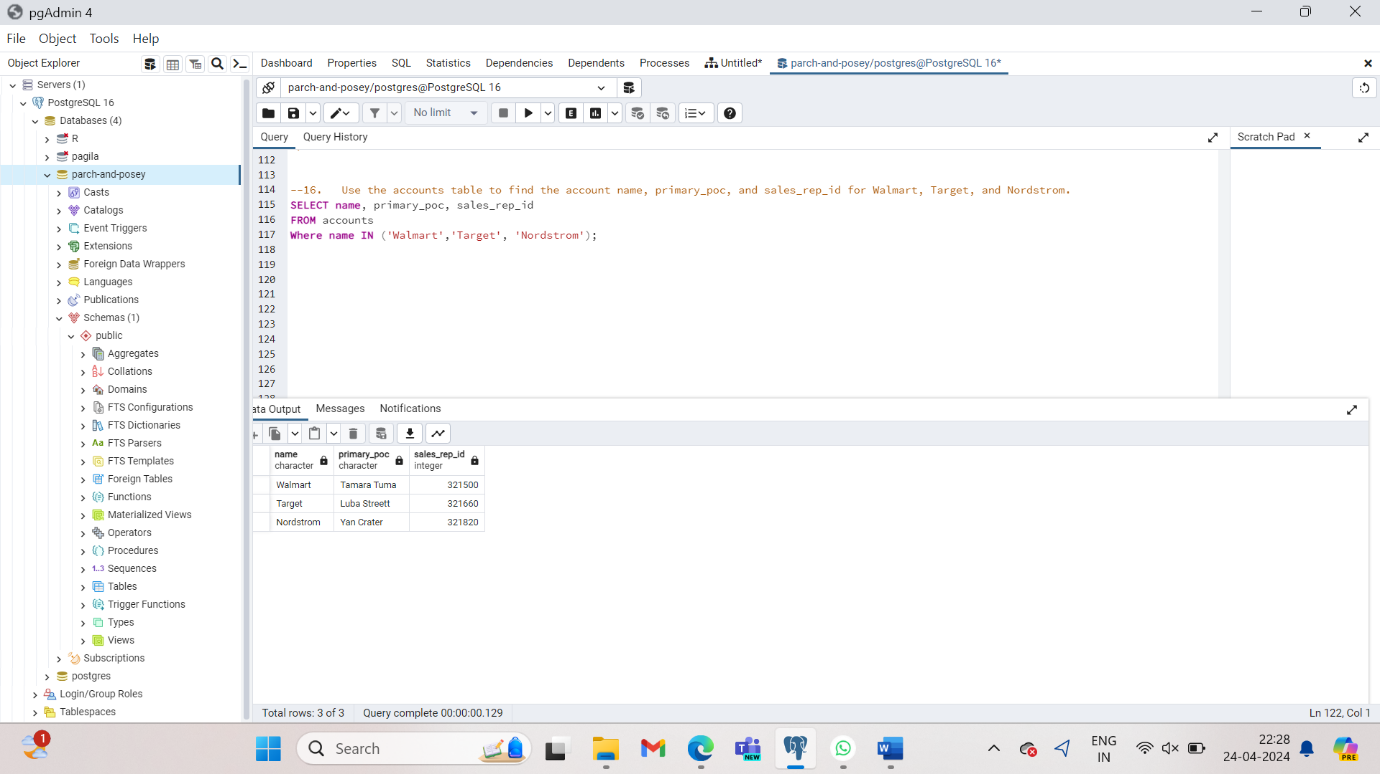


1. Use the **accounts** table to find the account name, primary\_poc, and sales\_rep\_id for Walmart, Target, and Nordstrom.

SELECT name, primary\_poc, sales\_rep\_id

FROM accounts

Where name IN (‘Walmart’,’Target’, ‘Nordstrom’);

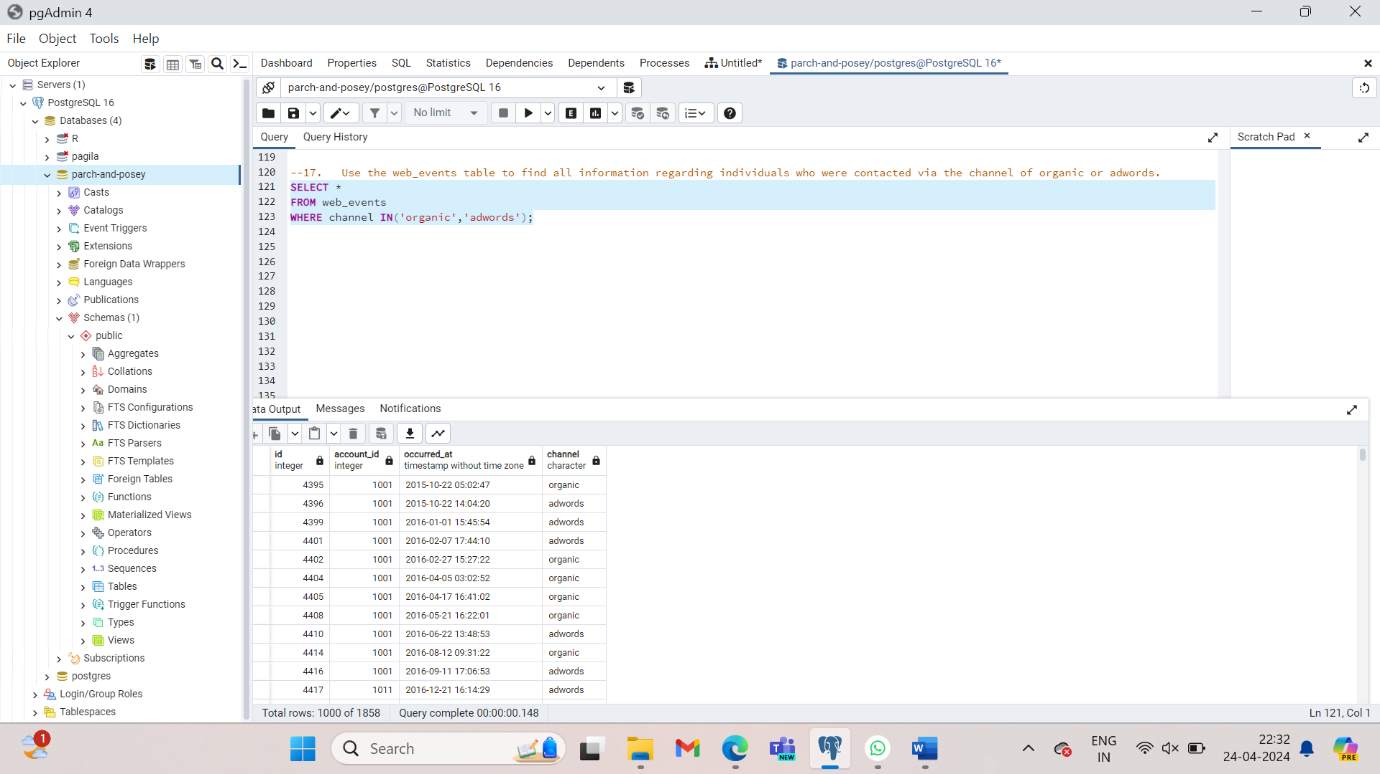


1. Use the **web\_events** table to find all information regarding individuals who were contacted via the **channel** of organic or adwords.

SELECT \*

FROM web\_events

WHERE channel IN(‘organic’,’adwords’);

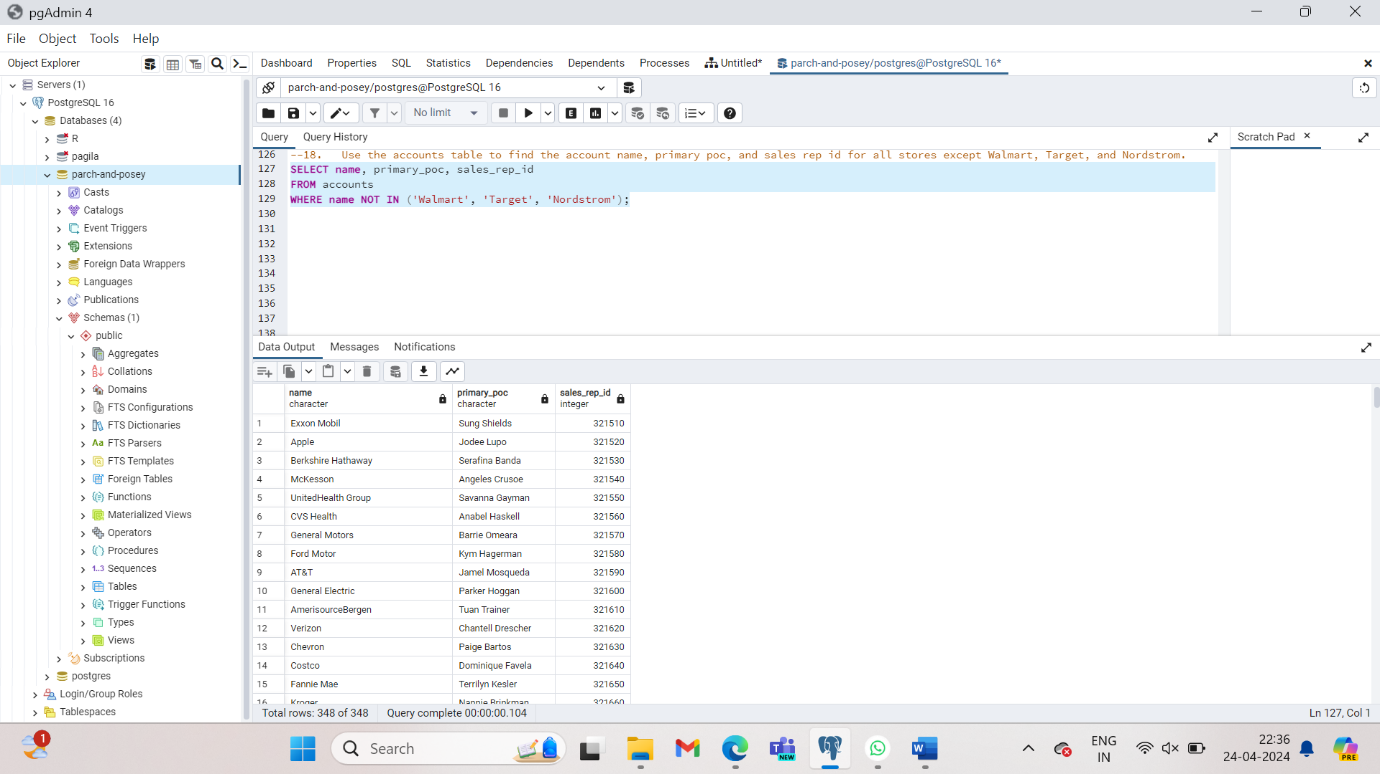


1. Use the **accounts** table to find the account name, primary poc, and sales rep id for all stores except Walmart, Target, and Nordstrom.

SELECT name, primary\_poc, sales\_rep\_id

FROM accounts

WHERE name NOT IN (‘Walmart’, ’Target’, ’Nordstrom’);



1. Use the **web\_events** table to find all information regarding individuals who were contacted via any method except using organic or adwords methods.

SELECT \*

FROM web\_events

WHERE channel NOT IN (‘organic’, ‘adwords’);

1. Use the **accounts** table to find all the companies whose names do not start with 'C'; all companies whose names do not contain the string 'one' somewhere in the name; all companies whose names do not end with 's'.

SELECT name

FROM accounts

WHERE name NOT LIKE ‘C%’;

SELECT name

FROM accounts

WHERE name NOT LIKE ‘%one%’;

SELECT name

FROM accounts

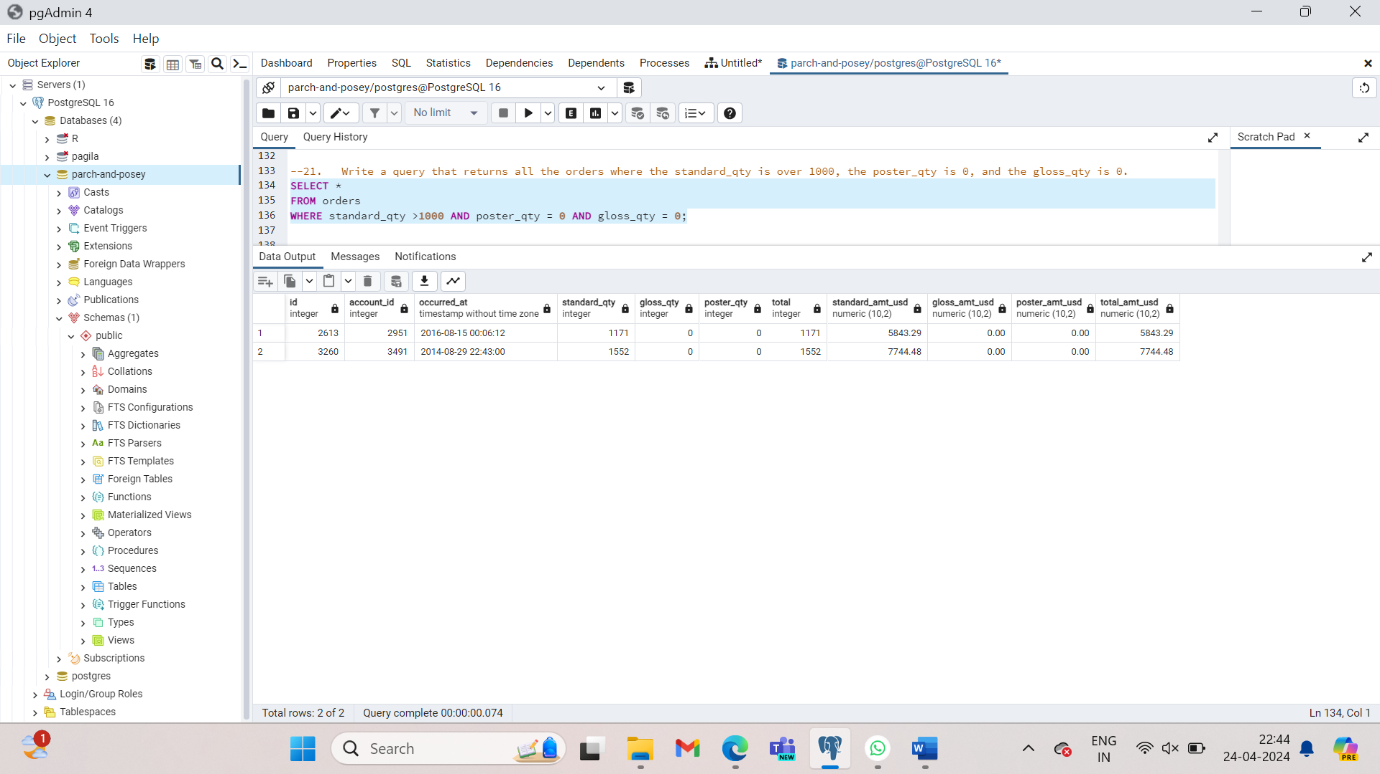
WHERE name NOT LIKE ‘%s’;

1. Write a query that returns all the **orders** where the standard\_qty is over 1000, the poster\_qty is 0, and the gloss\_qty is 0.

SELECT \*

FROM orders

WHERE standard\_qty >1000 AND poster\_qty = 0 AND ploss\_qty = 0;



1. Use the **web\_events** table to find all information regarding individuals who were contacted via the organic or adwords channels, and started their account at any point in 2016, sorted from newest to oldest.

SELECT \*

FROM web\_events

WHERE channel IN(‘organic’, ‘adwords’) AND occurred\_at BETWEEN '2016-01-01' AND '2017-01-01'

ORDER BY occurred\_at DESC;

1. Find list of **orders** ids where either gloss\_qty or poster\_qty is greater than 4000. Only include the id field in the resulting table.

SELECT id

FROM orders

WHERE gloss\_qty > 4000 or poster\_qty > 4000;

1. Write a query that returns a list of **orders** where the standard\_qty is zero and either the gloss\_qty or poster\_qty is over 1000.

SELECT \*

FROM orders

WHERE standard\_qty =0 AND (gloss\_qty >1000 OR poster\_qty >1000);

1. Find all the company names that start with a 'C' or 'W', and the primary contact **contains** 'ana' or 'Ana', but it doesn't contain 'eana'.

SELECT \*

FROM accounts

**WHERE** (name LIKE 'C%' OR name LIKE 'W%')

AND ((primary\_poc LIKE '%ana%' OR primary\_poc LIKE '%Ana%')

AND primary\_poc NOT LIKE '%eana%');

