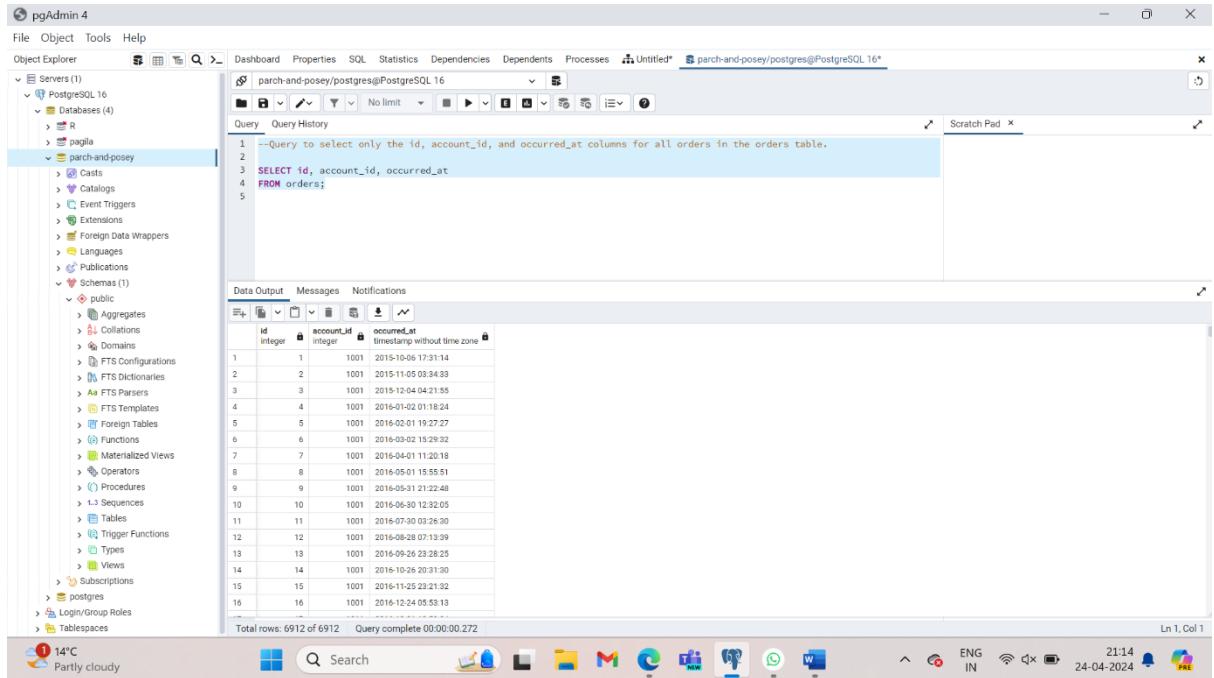


## Basic SQL

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;
```



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the Object Explorer with the 'Servers' node expanded, showing 'PostgreSQL 16' and its 'Databases' (R, pagila, parch-and-posey). The 'parch-and-posey' database is selected. The main window contains a query editor with the following content:

```
--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;
```

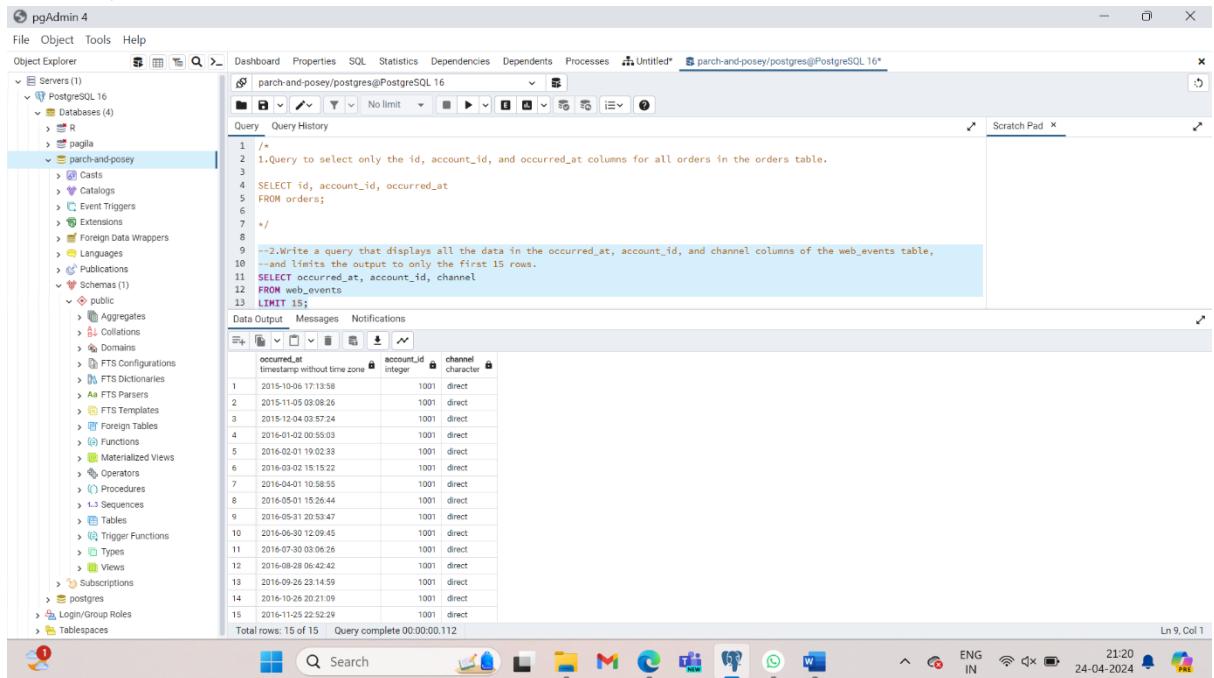
Below the query editor is a data grid showing the results of the query. The columns are id, account\_id, and occurred\_at. The data consists of 6912 rows, with the first few rows being:

|    | id | account_id | occurred_at         |
|----|----|------------|---------------------|
| 1  | 1  | 1001       | 2015-10-06 17:31:14 |
| 2  | 2  | 1001       | 2015-11-05 03:34:33 |
| 3  | 3  | 1001       | 2015-12-04 04:21:55 |
| 4  | 4  | 1001       | 2016-01-02 01:18:24 |
| 5  | 5  | 1001       | 2016-02-01 19:27:27 |
| 6  | 6  | 1001       | 2016-03-02 15:29:32 |
| 7  | 7  | 1001       | 2016-04-01 11:20:18 |
| 8  | 8  | 1001       | 2016-05-01 15:55:51 |
| 9  | 9  | 1001       | 2016-06-31 21:22:48 |
| 10 | 10 | 1001       | 2016-08-30 12:32:05 |
| 11 | 11 | 1001       | 2016-07-30 03:26:30 |
| 12 | 12 | 1001       | 2016-08-28 07:13:39 |
| 13 | 13 | 1001       | 2016-09-26 23:28:25 |
| 14 | 14 | 1001       | 2016-10-26 20:31:30 |
| 15 | 15 | 1001       | 2016-11-25 23:21:32 |
| 16 | 16 | 1001       | 2016-12-24 05:53:13 |

Total rows: 6912 Query complete 00:00:00.272

2. 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;
```



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the Object Explorer with the 'Servers' node expanded, showing 'PostgreSQL 16' and its 'Databases' (R, pagila, parch-and-posey). The 'parch-and-posey' database is selected. The main window contains a query editor with the following content:

```
/*
1.Query to select only the id, account_id, and occurred_at columns for all orders in the orders table.
2.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.
*/
--2.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;
```

Below the query editor is a data grid showing the results of the query. The columns are occurred\_at, account\_id, and channel. The data consists of 15 rows, with the first few rows being:

|    | occurred_at         | account_id | channel |
|----|---------------------|------------|---------|
| 1  | 2015-10-06 17:13:58 | 1001       | direct  |
| 2  | 2015-11-05 03:08:26 | 1001       | direct  |
| 3  | 2015-12-04 03:57:24 | 1001       | direct  |
| 4  | 2016-01-02 00:55:03 | 1001       | direct  |
| 5  | 2016-02-01 19:02:33 | 1001       | direct  |
| 6  | 2016-03-02 15:15:22 | 1001       | direct  |
| 7  | 2016-04-01 10:58:55 | 1001       | direct  |
| 8  | 2016-05-01 15:26:44 | 1001       | direct  |
| 9  | 2016-05-31 20:53:47 | 1001       | direct  |
| 10 | 2016-06-30 12:09:45 | 1001       | direct  |
| 11 | 2016-07-30 03:06:26 | 1001       | direct  |
| 12 | 2016-08-28 06:42:42 | 1001       | direct  |
| 13 | 2016-09-26 23:14:59 | 1001       | direct  |
| 14 | 2016-10-26 20:21:09 | 1001       | direct  |
| 15 | 2016-11-25 22:52:29 | 1001       | direct  |

Total rows: 15 of 15 Query complete 00:00:00.112

3. 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;
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database structure under "Servers (1) > PostgreSQL 16 > Databases (4) > parch-and-posey".
- Query Editor:** Contains the SQL query:
 

```
18 --3. Write a query to return the 10 earliest orders in the orders table. Include the id, occurred_at, and total_amt_usd.
19 SELECT id, occurred_at, total_amt_usd
20 FROM orders
21 ORDER BY occurred_at;
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
```
- Data Output:** Displays the results of the query, showing 10 rows of data:
 

|    | id   | occurred_at         | total_amt_usd |
|----|------|---------------------|---------------|
| 1  | 5786 | 2013-12-04 04:22:44 | 627.48        |
| 2  | 2415 | 2013-12-04 04:45:54 | 2648.77       |
| 3  | 4108 | 2013-12-04 04:53:23 | 2709.62       |
| 4  | 4489 | 2013-12-05 20:29:16 | 277.13        |
| 5  | 287  | 2013-12-05 20:35:56 | 3001.85       |
| 6  | 1946 | 2013-12-06 02:13:20 | 2802.90       |
| 7  | 6197 | 2013-12-06 12:58:22 | 7009.18       |
| 8  | 3122 | 2013-12-06 12:57:41 | 1992.13       |
| 9  | 6078 | 2013-12-06 13:14:47 | 6680.06       |
| 10 | 2932 | 2013-12-06 13:17:25 | 2075.94       |
| 11 | 6277 | 2013-12-06 02:45:25 | 1086.29       |
- System Bar:** Shows the date (24-04-2024), time (21:25), and system status.

4. 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;
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database structure under "Servers (1) > PostgreSQL 16 > Databases (4) > parch-and-posey".
- Query Editor:** Contains the SQL query:
 

```
26 --4. 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
27 SELECT id, account_id, total_amt_usd
28 FROM orders
29 ORDER BY total_amt_usd DESC
30 LIMIT 5;
31
32
33
34
35
36
37
38
39
40
41
42
43
```
- Data Output:** Displays the results of the query, showing 5 rows of data:
 

|   | id   | account_id | total_amt_usd |
|---|------|------------|---------------|
| 1 | 4016 | 4251       | 232207.07     |
| 2 | 3892 | 4161       | 112875.18     |
| 3 | 3963 | 4211       | 107533.55     |
| 4 | 5791 | 2861       | 95005.82      |
| 5 | 3778 | 4101       | 93547.84      |
- System Bar:** Shows the date (24-04-2024), time (21:29), and system status.

5. 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;

```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the Object Explorer with a tree view of databases, schemas, and objects. The main area shows a query editor with the following SQL code:

```

--5.-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;

```

The results pane shows the output of the query, which contains 20 rows of order details. The columns are id, account\_id, and total\_amt\_usd. The data includes various order IDs and account IDs, with total amounts ranging from 0.00 to 3551.00.

6. 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;

```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the Object Explorer with a tree view of databases, schemas, and objects. The main area shows a query editor with the following SQL code:

```

--6.-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;

```

The results pane shows the output of the query, which contains 6912 rows of order details. The columns are id, account\_id, and total\_amt\_usd. The data is sorted by account\_id (1001, 1002, 1003) and then by total\_amt\_usd in descending order. The total number of rows is 6912.

7. 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;
```

|      | id   | account_id | total_amt_usd |
|------|------|------------|---------------|
| 1    | 4016 | 4251       | 232207.07     |
| 2    | 3892 | 4161       | 112375.18     |
| 3    | 3963 | 4211       | 107532.55     |
| 4    | 5791 | 2861       | 95005.62      |
| 5    | 3778 | 4101       | 93547.84      |
| 6    | 6590 | 4111       | 93505.69      |
| 7    | 362  | 1301       | 93106.81      |
| 8    | 731  | 1521       | 92991.05      |
| 9    | 4562 | 1341       | 84099.62      |
| 10   | 3958 | 4151       | 82165.71      |
| 11   | 4942 | 1701       | 81243.95      |
| 12   | 1191 | 1831       | 77285.75      |
| 13   | 1913 | 2461       | 48675.90      |
| 14   | 5479 | 2441       | 45465.23      |
| 15   | 4698 | 1451       | 43421.28      |
| 16   | 5167 | 2041       | 40928.70      |
| ...  | ...  | ...        | ...           |
| 1000 | ...  | ...        | ...           |

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.)

8. 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;
```

pgAdmin 4

File Object Tools Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes Untitled\* parch-and-posey@PostgreSQL 16\*

Query History

```
55 --8. 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
56 -- greater than or equal to 1000.
57
58 SELECT *
59 FROM orders
60 WHERE gloss_amt_usd >=1000;
61
```

Data Output Messages Notifications

|    | <b>id</b> | <b>account_id</b> | <b>occurred_at</b>  | <b>standard_qty</b> | <b>gloss_qty</b> | <b>poster_qty</b> | <b>total</b> | <b>standard_amt_usd</b> | <b>gloss_amt_usd</b> | <b>poster_amt_usd</b> | <b>total_amt_usd</b> |
|----|-----------|-------------------|---------------------|---------------------|------------------|-------------------|--------------|-------------------------|----------------------|-----------------------|----------------------|
| 1  | 14        | 1001              | 2016-10-26 20:31:30 | 97                  | 143              | 54                | 294          | 484.03                  | 1071.07              | 438.48                | 1993.58              |
| 2  | 62        | 1091              | 2014-10-13 12:12:55 | 146                 | 196              | 3                 | 345          | 728.54                  | 1468.04              | 24.36                 | 2220.94              |
| 3  | 88        | 1101              | 2015-05-24 13:08:15 | 182                 | 339              | 17                | 538          | 908.18                  | 2393.11              | 138.04                | 3585.33              |
| 4  | 121       | 1131              | 2016-08-10 23:47:41 | 273                 | 134              | 0                 | 407          | 1362.27                 | 1003.66              | 0.00                  | 2365.93              |
| 5  | 129       | 1141              | 2016-12-11 15:52:58 | 143                 | 1045             | 2157              | 3345         | 713.57                  | 7827.05              | 17514.84              | 26055.46             |
| 6  | 187       | 1191              | 2016-03-23 21:33:31 | 485                 | 134              | 0                 | 619          | 2420.15                 | 1003.66              | 0.00                  | 3423.81              |
| 7  | 196       | 1191              | 2016-10-15 20:49:47 | 0                   | 171              | 24                | 195          | 0.00                    | 1280.79              | 194.88                | 1475.67              |
| 8  | 214       | 1221              | 2016-04-04 20:17:54 | 485                 | 1345             | 21                | 1851         | 2420.15                 | 10074.05             | 170.52                | 12664.72             |
| 9  | 262       | 1261              | 2015-08-16 12:33    | 277                 | 566              | 35                | 878          | 1382.23                 | 4239.34              | 284.20                | 5905.77              |
| 10 | 301       | 1281              | 2015-01-23 07:36:27 | 515                 | 342              | 26                | 883          | 2569.85                 | 2561.58              | 211.12                | 5342.55              |
| 11 | 320       | 1281              | 2016-09-11 04:09:37 | 3337                | 1729             | 12                | 5078         | 16651.63                | 12950.21             | 97.44                 | 29699.28             |
| 12 | 458       | 1401              | 2014-12-02 11:31:30 | 340                 | 918              | 14                | 1272         | 1696.60                 | 6875.82              | 113.68                | 8686.10              |
| 13 | 498       | 1411              | 2015-05-19 21:15:13 | 549                 | 542              | 0                 | 1091         | 2739.51                 | 4059.58              | 0.00                  | 6799.09              |
| 14 | 503       | 1411              | 2015-05-13 10:23:01 | 508                 | 303              | 0                 | 811          | 2534.92                 | 2269.47              | 0.00                  | 4804.39              |
| 15 | 506       | 1411              | 2015-11-08 09:35:32 | 493                 | 182              | 4                 | 679          | 2460.07                 | 1363.18              | 32.48                 | 3855.73              |
| 16 | 510       | 1411              | 2016-03-05 23:01:52 | 499                 | 158              | 46                | 703          | 2490.01                 | 1183.42              | 373.52                | 4046.95              |
| 17 | 521       | 1421              | 2014-02-28 19:24:53 | 322                 | 216              | 6                 | 544          | 1606.78                 | 1617.84              | 48.72                 | 3273.34              |
| 18 | 579       | 1441              | 2016-06-10 02:00:06 | 189                 | 222              | 11                | 422          | 943.11                  | 1662.78              | 89.32                 | 2695.21              |
| 19 | 623       | 1451              | 2016-07-10 11:04:03 | 291                 | 197              | 29                | 517          | 1452.09                 | 1475.53              | 235.48                | 3163.10              |

Total rows: 1000 of 1579 Query complete 00:00:00.116 Ln 58, Col 1

9. 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;
```

pgAdmin 4

File Object Tools Help

Object Explorer

Dashboard Properties SQL Statistics Dependencies Dependents Processes Untitled\* parch-and-posey@PostgreSQL 16\*

Query History

```
62 --9. 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.
63
64 SELECT *
65 FROM orders
66 WHERE total_amt_usd < 500
67 Limit 10;
```

Data Output Messages Notifications

|    | <b>id</b> | <b>account_id</b> | <b>occurred_at</b>  | <b>standard_qty</b> | <b>gloss_qty</b> | <b>poster_qty</b> | <b>total</b> | <b>standard_amt_usd</b> | <b>gloss_amt_usd</b> | <b>poster_amt_usd</b> | <b>total_amt_usd</b> |
|----|-----------|-------------------|---------------------|---------------------|------------------|-------------------|--------------|-------------------------|----------------------|-----------------------|----------------------|
| 1  | 67        | 1091              | 2015-04-07 13:29:20 | 95                  | 0                | 0                 | 95           | 474.05                  | 0.00                 | 0.00                  | 474.05               |
| 2  | 96        | 1101              | 2016-03-15 11:36:03 | 14                  | 8                | 16                | 38           | 69.86                   | 59.92                | 129.92                | 259.70               |
| 3  | 119       | 1131              | 2016-06-12 12:29:45 | 0                   | 30               | 23                | 53           | 0.00                    | 224.70               | 186.76                | 411.46               |
| 4  | 124       | 1131              | 2016-10-07 05:10:56 | 0                   | 0                | 0                 | 0            | 0.00                    | 0.00                 | 0.00                  | 0.00                 |
| 5  | 254       | 1251              | 2014-11-01 02:15:24 | 0                   | 0                | 17                | 17           | 0.00                    | 0.00                 | 138.04                | 138.04               |
| 6  | 328       | 1291              | 2015-09-03 08:35:23 | 0                   | 19               | 21                | 40           | 0.00                    | 142.31               | 170.52                | 312.83               |
| 7  | 542       | 1421              | 2015-11-13 09:07:09 | 0                   | 64               | 0                 | 64           | 0.00                    | 479.36               | 0.00                  | 479.36               |
| 8  | 683       | 1501              | 2016-04-14 23:59:50 | 0                   | 15               | 16                | 31           | 0.00                    | 112.35               | 129.92                | 242.27               |
| 9  | 713       | 1521              | 2014-11-23 16:04:03 | 0                   | 8                | 10                | 18           | 0.00                    | 59.92                | 81.20                 | 141.12               |
| 10 | 730       | 1521              | 2015-05-06 02:34:48 | 0                   | 0                | 2                 | 2            | 0.00                    | 0.00                 | 16.24                 | 16.24                |

Total rows: 10 of 10 Query complete 00:00:00.102 Ln 64, Col 1

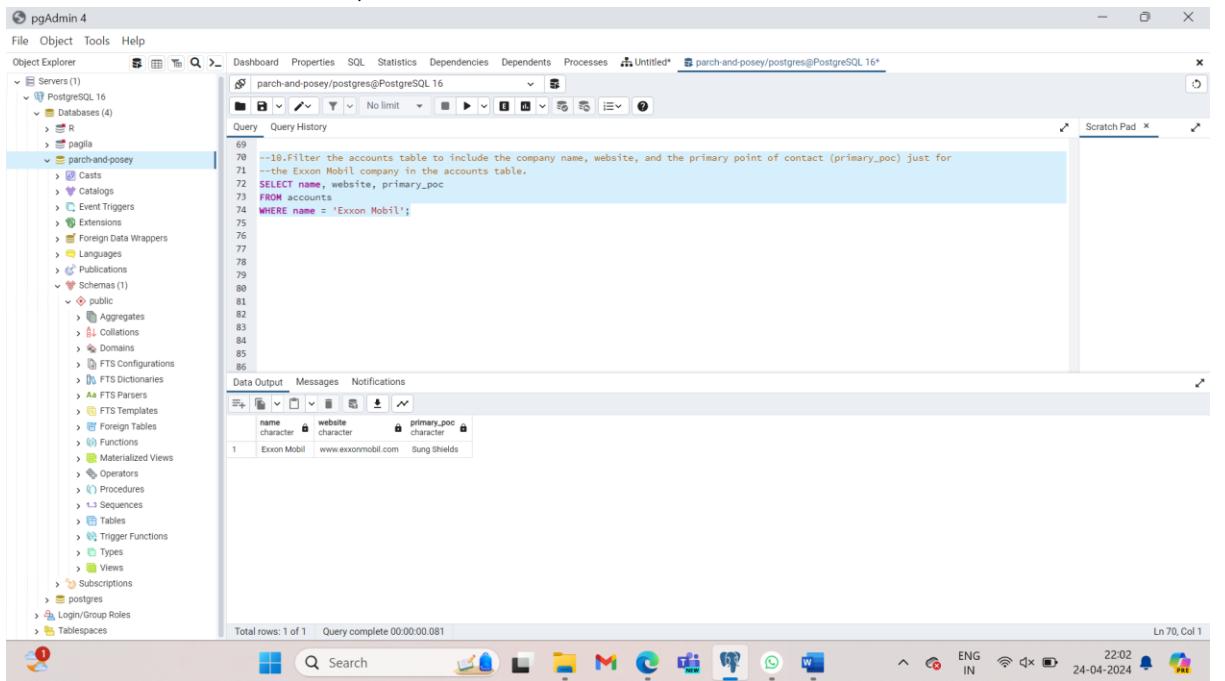
When we are using **WHERE** with non-numeric data fields, we use the **LIKE**, **NOT**, or **IN** operators.

10. 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';



The screenshot shows the pgAdmin 4 interface. In the Object Explorer, the 'Servers' node is expanded, showing 'PostgreSQL 16' and its sub-nodes: 'Databases', 'Casts', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Publications', 'Schemas', 'public', 'Aggregates', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Operators', 'Procedures', 'Sequences', 'Tables', 'Trigger Functions', 'Types', 'Views', and 'Subscriptions'. The 'Databases' node has three entries: 'R', 'pagila', and 'parch-and-posey', with 'parch-and-posey' being the selected database. In the main Query Editor window, the following SQL code is written:

```
69 --18. Filter the accounts table to include the company name, website, and the primary point of contact (primary_poc) just for
70 --the Exxon Mobil company in the accounts table.
71
72 SELECT name, website, primary_poc
73 FROM accounts
74 WHERE name = 'Exxon Mobil';
75
76
77
78
79
80
81
82
83
84
85
86
```

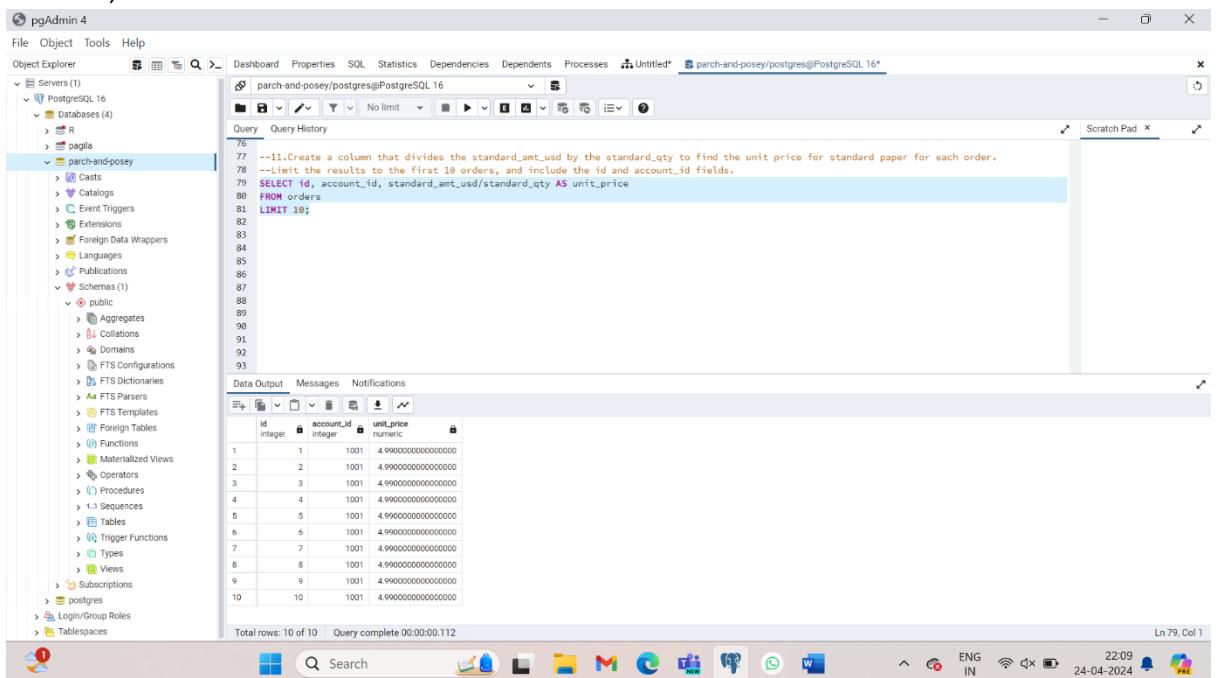
The Data Output tab shows the results of the query:

|   | name        | website            | primary_poc  |
|---|-------------|--------------------|--------------|
| 1 | Exxon Mobil | www.exxonmobil.com | Sung Shields |

Total rows: 1 of 1 Query complete 00:00:00.081

11. 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;
```



The screenshot shows the pgAdmin 4 interface. The Object Explorer is identical to the previous one. In the main Query Editor window, the following SQL code is written:

```
76 --11.Create a column that divides the standard_amt_usd by the standard_qty to find the unit price for standard paper for each order.
77 --Limit the results to the first 10 orders, and include the id and account_id fields.
78
79 SELECT id, account_id, standard_amt_usd/standard_qty AS unit_price
80 FROM orders
81 LIMIT 10;
82
83
84
85
86
87
88
89
90
91
92
93
```

The Data Output tab shows the results of the query:

|    | id | account_id | unit_price        |
|----|----|------------|-------------------|
| 1  | 1  | 1001       | 4.990000000000000 |
| 2  | 2  | 1001       | 4.990000000000000 |
| 3  | 3  | 1001       | 4.990000000000000 |
| 4  | 4  | 1001       | 4.990000000000000 |
| 5  | 5  | 1001       | 4.990000000000000 |
| 6  | 6  | 1001       | 4.990000000000000 |
| 7  | 7  | 1001       | 4.990000000000000 |
| 8  | 8  | 1001       | 4.990000000000000 |
| 9  | 9  | 1001       | 4.990000000000000 |
| 10 | 10 | 1001       | 4.990000000000000 |

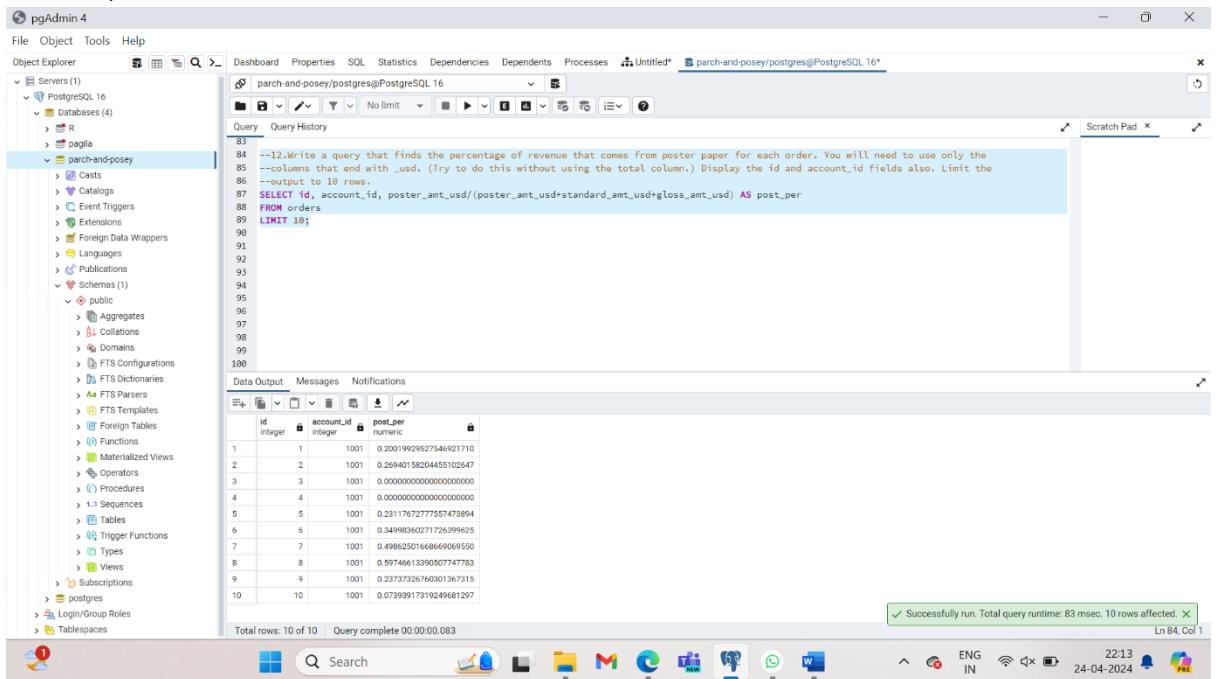
Total rows: 10 of 10 Query complete 00:00:00.112

12. 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;**



The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
--12. 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;
```

The results table has columns: id, account\_id, and post\_per. The data is:

|    | id | account_id | post_per                           |
|----|----|------------|------------------------------------|
| 1  | 1  | 1001       | 0.2001992952754692170              |
| 2  | 2  | 1001       | 0.269401582044515102647            |
| 3  | 3  | 1001       | 0.00000000000000000000000000000000 |
| 4  | 4  | 1001       | 0.00000000000000000000000000000000 |
| 5  | 5  | 1001       | 0.2311762777557473894              |
| 6  | 6  | 1001       | 0.3499836027172599625              |
| 7  | 7  | 1001       | 0.4986250166866906950              |
| 8  | 8  | 1001       | 0.59746613390507747783             |
| 9  | 9  | 1001       | 0.23737325760001367315             |
| 10 | 10 | 1001       | 0.0793991731924961297              |

Total rows: 10 of 10 | Query complete 00:00:00.083 | Ln 84, Col 1

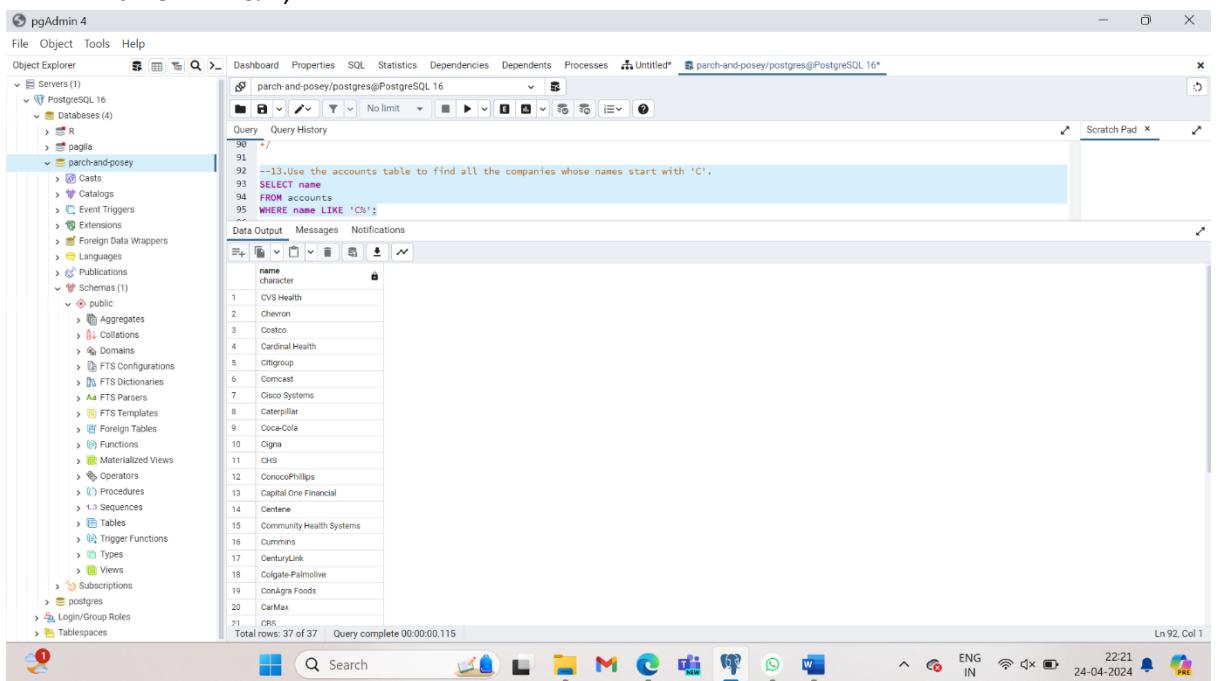
Successfully run. Total query runtime: 83 msec. 10 rows affected.

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

**SELECT name**

**FROM accounts**

**WHERE name LIKE 'C%';**



The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
--13. Use the accounts table to find all the companies whose names start with 'C'.
SELECT name
FROM accounts
WHERE name LIKE 'C%';
```

The results table has a single column: name. The data is:

| name                     |
|--------------------------|
| CVS Health               |
| Chevron                  |
| Costco                   |
| Cardinal Health          |
| Citigroup                |
| Concast                  |
| Cisco Systems            |
| Caterpillar              |
| Coca-Cola                |
| Digna                    |
| CHS                      |
| ConocoPhillips           |
| Capital One Financial    |
| Centene                  |
| Community Health Systems |
| Cummins                  |
| CenturyLink              |
| Colgate-Palmolive        |
| ConAgra Foods            |
| CarMax                   |
| CRS                      |

Total rows: 37 of 37 | Query complete 00:00:00.115 | Ln 92, Col 1

14. 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%';**

The screenshot shows the pgAdmin 4 interface with a query window containing the following SQL code:

```
--14. 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%';
```

The results pane displays three rows of data:

| name                    | character |
|-------------------------|-----------|
| Honeywell International |           |
| INTL FCStone            |           |
| AutoZone                |           |

Total rows: 3 of 3    Query complete 00:00:00.097

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

```
SELECT name
FROM accounts
WHERE name LIKE '%s';
```

The screenshot shows the pgAdmin 4 interface with a query window containing the following SQL code:

```
--15. Use the accounts table to find all companies whose names end with 's'.
SELECT name
FROM accounts
WHERE name LIKE '%s';
```

The results pane displays 20 rows of data:

| name                         | character |
|------------------------------|-----------|
| General Motors               |           |
| United Technologies          |           |
| Lowe's                       |           |
| Cisco Systems                |           |
| HCA Holdings                 |           |
| Tyson Foods                  |           |
| Delta Air Lines              |           |
| Johnson Controls             |           |
| Ross Stores                  |           |
| United Continental Holdings  |           |
| American Express             |           |
| Oilead Sciences              |           |
| Publix Super Markets         |           |
| General Dynamics             |           |
| ConocoPhillips               |           |
| World Fuel Services          |           |
| Macy's                       |           |
| Enterprise Products Partners |           |
| McDonald's                   |           |
| Sears Holdings               |           |

Total rows: 77 of 77    Query complete 00:00:00.148

16. 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');
```

```

--16. 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');

name          primary_poc    sales_rep_id
Walmart        Tamara Tuna   321500
Target        Luba Street   321660
Nordstrom     Yan Crater   321620

```

17. 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');

```

```

--17. 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');

account_id  account_nm  occurred_at  channel
4095        1001       2015-10-22 05:02:47  organic
4096        1001       2015-10-22 14:04:20  adwords
4099        1001       2016-01-01 15:49:54  adwords
4401        1001       2016-07-07 17:44:10  adwords
4402        1001       2016-06-27 15:27:22  organic
4404        1001       2016-04-05 03:02:52  organic
4405        1001       2016-04-17 16:41:02  organic
4408        1001       2016-06-21 16:22:01  organic
4410        1001       2016-06-22 13:48:53  adwords
4414        1001       2016-06-12 09:31:22  organic
4416        1001       2016-06-11 17:06:53  adwords
4417        1011       2016-12-21 16:14:29  adwords

```

18. 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');

```

```

126 --18. Use the accounts table to find the account name, primary poc, and sales rep id for all stores except Walmart, Target, and Nordstrom.
127
128   SELECT name, primary_poc, sales_rep_id
129   FROM accounts
130   WHERE name NOT IN ('Walmart', 'Target', 'Nordstrom');
131
132
133
134
135
136
137
138

```

| name               | primary_poc       | sales_rep_id |
|--------------------|-------------------|--------------|
| Exxon Mobil        | Sung Shields      | 321510       |
| Apple              | Jodee Lupo        | 321520       |
| Berkshire Hathaway | Serafina Banda    | 321530       |
| Mckesson           | Angelis Crusti    | 321540       |
| UnitedHealth Group | Savanna Gayman    | 321550       |
| CVS Health         | Anabel Haskell    | 321560       |
| General Motors     | Barnie O'meara    | 321570       |
| Ford Motor         | Kym Hagerman      | 321580       |
| AT&T               | Jamel Mosqueda    | 321590       |
| General Electric   | Parker Hoggar     | 321600       |
| AmerisourceBergen  | Tuan Trainer      | 321610       |
| Verizon            | Chantell Drescher | 321620       |
| Chevron            | Paige Bartos      | 321630       |
| Costco             | Dominique Favala  | 321640       |
| Fannie Mae         | Terilyn Kesler    | 321650       |
| Kroger             | Nanette Brookman  | 321660       |

19. 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');

```

20. 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';

```

21. 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 gloss_qty = 0;

```

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```

132 --21. 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.
134 SELECT *
135 FROM orders
136 WHERE standard_qty > 1000 AND poster_qty = 0 AND gloss_qty = 0;
137
138

```

The results table has the following columns: id, account\_id, occurred\_at, standard\_qty, gloss\_qty, poster\_qty, total, standard\_amt\_usd, gloss\_amt\_usd, poster\_amt\_usd, and total\_amt\_usd. The data is:

|   | <b>id</b> | <b>account_id</b> | <b>occurred_at</b>  | <b>standard_qty</b> | <b>gloss_qty</b> | <b>poster_qty</b> | <b>total</b> | <b>standard_amt_usd</b> | <b>gloss_amt_usd</b> | <b>poster_amt_usd</b> | <b>total_amt_usd</b> |
|---|-----------|-------------------|---------------------|---------------------|------------------|-------------------|--------------|-------------------------|----------------------|-----------------------|----------------------|
| 1 | 2613      | 2951              | 2015-08-15 00:06:12 | 1171                | 0                | 0                 | 1171         | 5843.29                 | 0.00                 | 0.00                  | 5843.29              |
| 2 | 3260      | 8491              | 2014-08-29 22:48:00 | 1552                | 0                | 0                 | 1552         | 7744.48                 | 0.00                 | 0.00                  | 7744.48              |

22. 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;

```

23. 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;

```

24. 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);

```

25. 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%');

```

pgAdmin 4

File Object Tools Help

Object Explorer

Servers (1)  
PostgreSQL 16  
Databases (4)  
R  
pagila  
parch-and-posey  
Casts  
Catalogs  
Event Triggers  
Extensions  
Foreign Data Wrappers  
Languages  
Publications  
Schemas (1)  
public  
Aggregates  
Collations  
Domains  
FTS Configurations  
FTS Dictionaries  
FTS Parsers  
FTS Templates  
Foreign Tables  
Functions  
Materialized Views  
Operators  
Procedures  
Sequences  
Tables  
Trigger Functions  
Types  
Views  
Subscriptions  
postges  
Login/Group Roles  
Tablespaces

Dashboard Properties SQL Statistics Dependencies Dependents Processes Untitled\* parch-and-posey/postgres@PostgreSQL 16\*

Query Query History

```
--22.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 'seana'.
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 '%seana%');
```

Data Output Messages Notifications

|   | <b>id</b> | <b>name</b> | <b>website</b>             | <b>lat</b>  | <b>long</b>  | <b>primary_poc</b> | <b>sales_vp_id</b> |
|---|-----------|-------------|----------------------------|-------------|--------------|--------------------|--------------------|
| 1 | 1061      | CV9 Health  | www.cv9health.com          | 41.46779585 | -73.76763638 | Anabel Haskell     | 321560             |
| 2 | 1361      | Comcast     | www.comcastcorporation.com | 42.54154764 | -76.24992387 | Shana Sanborn      | 321650             |

Total rows: 2 of 2    Query complete 00:00:00.122

Scratch Pad

LN 139, Col 1

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, the 'parch-and-posey' database is selected. The main pane displays a SQL query to find company names starting with 'C' or 'W' where the primary contact contains 'ana' or 'Ana' but not 'seana'. The results show two companies: CV9 Health and Comcast, each with their respective coordinates and sales VP ID.