

SQL JOINS

The real power of SQL comes from working with data across multiple tables at once.

The term relational database refers to the fact that tables within it relate to one another. They contain common identifiers that allow information from multiple tables to be easily combined. Keeping all of company's data, from purchasing transactions, to employee job satisfaction to inventory in a single Excel dataset doesn't make a ton of sense. The table would hold a ton of information and it'd be hard to determine a row column structure for so many different types of data. **Databases give us the flexibility to keep things organized neatly in their own tables, so they are easy to find and work with while allowing us to combine tables as needed to solve problems that require several types of data.**

We will try to understand joins using the parch and posey database.

Why would we want to split data into separate tables?

If we look at the orders data table of parch and posey, we notice that none of the orders say the name of the customer. Instead, the table refers to customers by numerical values in the account id column. We need to join another table to connect this data to names.

		id	account_id	occurred_at	standard_qty	gloss_qty	poster_qty	total	standard_amt_usd	gloss_amt_usd	poster_amt_usd	total_amt_usd
1	1	1001		2015-10-06 17:31:14	123	22	24	169	613.77	164.78	194.88	973.43
2	2	1001		2015-11-05 03:34:33	190	41	57	288	948.1	307.09	462.84	1718.03
3	3	1001		2015-12-04 04:21:55	85	47	0	132	424.15	352.03	0	776.18
4	4	1001		2016-01-02 01:18:24	144	32	0	176	718.56	239.68	0	958.24

But first why isn't the customer's name in this table in the first place?

Orders and accounts are different types of objects and will be easier to organize if kept separate.

This multi-table structure allows queries to execute more quickly.



The accounts and orders tables store fundamentally different types of objects.

Parch and Posey probably only wants one account per company, and they want it to be up to date with the latest information. Orders on the other hand are likely to stay the same once they are

entered, and certainly once they are filled. A given customer might have multiple orders, and rather than change a past order, parch and posey might just add a new one. Because these objects operate differently, it makes sense for them to live in different tables.

Another reason to store this information separately has to do with the speed at which database can modify data. When we write a query its execution speed depends on the amount of data we're asking the database to read and the number and type of calculations you're asking it to make.

Database Normalization

Normalization is a technique that allows us to think about the way data will be stored when creating a database.

There are essentially three ideas that are aimed at database normalization:

- a. Are the tables storing logical groupings of the data?
- b. Can I make changes in a single location, rather than in many tables for the same information?
- c. Can I access and manipulate data quickly and efficiently?

The primary key (pkey) is an attribute (column) is a table that acts as a unique identifier for each row in a table. A foreign key (fkey) is an attribute that forms an implied link between two tables that are in a 1:M relationship. The fkey, which is a column in the table of the many, is usually a pkey in the table of the one.

Introduction to joins

1. The whole purpose of Join is to allow us to pull data from more than one table at a time.

SELECT orders.*

FROM orders

JOIN accounts

ON orders.account_id = accounts.id;

2. If we want to pull only the **account name** and the dates in which that account placed an order, but none of the other columns, we can do this with the following query:

SELECT accounts.name, orders.occurred_at

FROM orders

JOIN accounts

ON orders.account_id = accounts.id;

pgAdmin 4

File Object Tools Help

Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes

Query Query History

```

1 SELECT orders.*
2 FROM orders
3 JOIN accounts
4 ON orders.account_id = accounts.id;
5
6 SELECT accounts.name, orders.occurred_at
7 FROM orders
8 JOIN accounts
9 ON orders.account_id = accounts.id;
10

```

Data Output Messages Notifications

	name	occurred_at
1	Walmart	2015-10-06 17:31:14
2	Walmart	2015-11-05 03:34:33
3	Walmart	2015-12-04 04:21:55
4	Walmart	2016-01-02 01:18:24
5	Walmart	2016-02-01 19:27:27
6	Walmart	2016-03-02 15:29:32
7	Walmart	2016-04-01 11:20:18
8	Walmart	2016-05-01 15:55:51
9	Walmart	2016-05-31 21:22:48
10	Walmart	2016-06-30 12:32:05
11	Walmart	2016-07-30 03:26:30
12	Walmart	2016-08-28 07:13:39
13	Walmart	2016-09-26 23:28:25
14	Walmart	2016-10-26 20:31:30
15	Walmart	2016-11-25 23:21:32
16	Walmart	2016-12-24 05:58:13

Total rows: 1000 of 6912 Query complete 00:00:00.114

23°C Mostly sunny Search ENG IN 18:24 25-04-2024 Ln 6, Col 1

3. This query pulls all the columns from *both* the **accounts** and **orders** table.

SELECT *

FROM orders

JOIN accounts

ON orders.account_id = accounts.id;

pgAdmin 4

File Object Tools Help

Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes

Query Query History

```

1 SELECT orders.*;
2 FROM orders
3 JOIN accounts
4 ON orders.account_id = accounts.id;
5
6 SELECT *
7 FROM orders
8 JOIN accounts
9 ON orders.account_id = accounts.id;
10

```

Data Output Messages Notifications

	id	account_id	occurred_at	standard_qty	gloss_qty	poster_qty	total	standard_amt_usd	gloss_amt_usd	poster_amt_usd	total_amt_usd	id	name
1	1	1001	2015-10-06 17:31:14	123	22	169	613.77	164.78	194.88	973.43	1001	Walmart	
2	2	1001	2015-11-05 03:34:33	190	41	57	288	948.10	307.09	462.84	1718.03	1001	Walmart
3	3	1001	2015-12-04 04:21:55	85	47	0	132	424.15	352.03	0.00	776.18	1001	Walmart
4	4	1001	2016-01-02 01:18:24	144	32	0	176	718.56	239.68	0.00	958.24	1001	Walmart
5	5	1001	2016-02-01 19:27:27	108	29	28	165	538.92	217.21	227.36	983.49	1001	Walmart
6	6	1001	2016-03-02 15:29:32	103	24	46	173	518.97	179.76	373.52	1067.25	1001	Walmart
7	7	1001	2016-04-01 11:20:18	101	33	92	226	503.99	247.17	747.04	1498.20	1001	Walmart
8	8	1001	2016-05-01 15:55:51	95	47	151	293	474.05	352.03	1226.12	2052.20	1001	Walmart
9	9	1001	2016-05-31 21:22:48	91	16	22	129	454.09	119.84	178.64	752.57	1001	Walmart
10	10	1001	2016-06-30 12:32:05	94	46	8	148	469.06	344.54	64.96	878.56	1001	Walmart
11	11	1001	2016-07-30 03:26:30	101	36	0	137	503.99	269.64	0.00	773.63	1001	Walmart
12	12	1001	2016-08-28 07:13:39	124	33	39	196	618.76	247.17	316.68	1182.61	1001	Walmart
13	13	1001	2016-09-26 23:28:25	104	10	44	158	518.96	74.90	357.28	951.14	1001	Walmart
14	14	1001	2016-10-26 20:31:30	97	143	54	294	484.03	1071.07	438.48	1993.58	1001	Walmart

Total rows: 1000 of 6912 Query complete 00:00:00.174

DC - GT Game score Search ENG IN 18:27 25-04-2024 Ln 11, Col 1

4. Try pulling all the data from the **accounts** table, and all the data from the **orders** table.

SELECT *

FROM accounts

JOIN orders

ON accounts.id = orders.accounts_id;

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'Tables' section, the 'accounts' table is selected, showing 7 columns: id, name, website, lat, long, primary_poc, and sales_rep_id. The 'orders' table is also listed with 11 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.

In the main window, a query is being run:

```
1 SELECT orders.*  
2 FROM orders  
3 JOIN accounts  
4 ON orders.account_id = accounts.id;  
5  
6 SELECT accounts.name, orders.occurred_at  
7 FROM orders  
8 JOIN accounts  
9 ON orders.account_id = accounts.id;  
10  
11 SELECT *  
12 FROM orders  
13 JOIN accounts  
14 ON orders.account_id = accounts.id;  
15  
16 --4. Try pulling all the data from the accounts table, and all the data from the orders table.  
17  
18 SELECT *  
19 FROM accounts  
20 JOIN orders  
21 ON accounts.id = orders.account_id;
```

The results show 1000 rows of data from the accounts table and 6912 rows from the orders table.

5. Try pulling **standard_qty**, **gloss_qty**, and **poster_qty** from the **orders** table, and the **website** and the **primary_poc** from the **accounts** table.

SELECT orders.standard_qty, orders.gloss_qty, orders.poster_qty, accounts.website, accounts.primary_poc
FROM orders
JOIN accounts
ON orders.account_id = accounts.id;

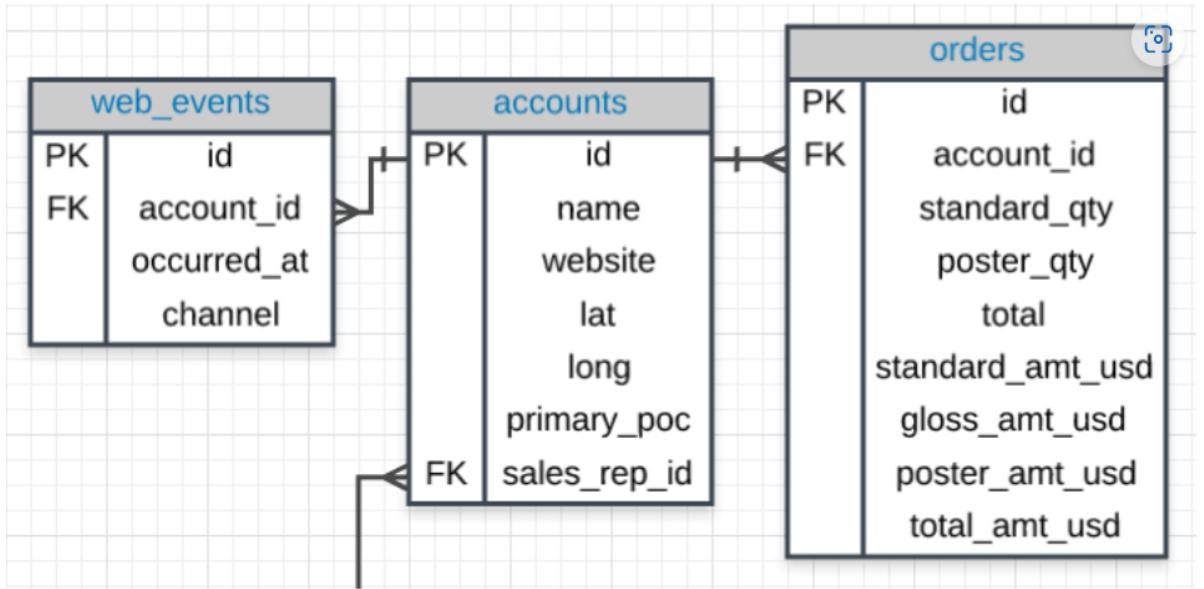
The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'Tables' section, the 'accounts' table is selected, showing 7 columns: id, name, website, lat, long, primary_poc, and sales_rep_id. The 'orders' table is also listed with 11 columns: id, account_id, occurred_at, standard_qty, gloss_qty, poster_qty, total, standard_amt_usd, gloss_amt_usd, poster_amt_usd, and total_amt_usd.

In the main window, a query is being run:

```
21  
22 --5. Try pulling standard_qty, gloss_qty, and poster_qty from the orders table, and the website and the primary_poc from the accounts table.  
23 SELECT orders.standard_qty, orders.gloss_qty, orders.poster_qty, accounts.website, accounts.primary_poc  
24 FROM orders  
25 JOIN accounts  
26 ON orders.account_id = accounts.id;
```

The results show 1000 rows of data from the accounts table and 6912 rows from the orders table.

6. If you were to join the **web_events**, **accounts** and **orders** table, how would you do it?



```

SELECT *
FROM web_events
JOIN accounts
ON web_events.account_id = accounts.id
JOIN orders
ON accounts.id = orders.account_id;
  
```

Screenshot of pgAdmin 4 interface showing the execution of the query:

- Object Explorer:** Shows the database schema with tables `web_events`, `accounts`, and `orders`.
- Query Editor:** Contains the SQL query:


```

28 | --, if you were to join the web_events, accounts and orders table how would you do it?
29 | SELECT *
30 | FROM web_events
31 | JOIN accounts
32 | ON web_events.account_id = accounts.id
33 | JOIN orders
34 | ON accounts.id = orders.account_id;
      
```
- Data Output:** Displays the results of the query, showing 1000 rows of data from the joined tables.

- Pull specific columns from any of the three tablesee tables
- ```

SELECT web_events.channel, accounts.name, orders.total
FROM web_events
JOIN accounts
ON web_events.account_id = accounts.id
JOIN orders
ON accounts.id = orders.account_id;

```

```

36 --7. Pull specific columns from any of the three table see tables
37 SELECT web_events.channel, accounts.name, orders.total
38 FROM web_events
39 JOIN accounts
40 ON web_events.account_id = accounts.id
41 JOIN orders
42 ON accounts.id = orders.account_id;
43
44

```

| channel | name    | total |
|---------|---------|-------|
| direct  | Walmart | 1205  |
| direct  | Walmart | 1347  |
| direct  | Walmart | 1384  |
| direct  | Walmart | 1238  |
| direct  | Walmart | 1343  |
| direct  | Walmart | 1254  |
| direct  | Walmart | 1267  |
| direct  | Walmart | 1307  |
| direct  | Walmart | 1280  |
| direct  | Walmart | 1405  |
| direct  | Walmart | 1410  |
| direct  | Walmart | 1321  |
| direct  | Walmart | 269   |
| direct  | Walmart | 210   |
| direct  | Walmart | 294   |
| direct  | Walmart | 158   |
| direct  | Walmart | 196   |
| direct  | Walmart | 137   |

Total rows: 1000 of 321483 Query complete 00:00:00.325

When performing JOINS, it is easiest to give table names aliases. The aliases for a table will be created in the from and join clauses. Once we have given these aliases, we can refer to columns in the select clause by using those alias names. Generally, the first character of the table name is chosen as the alias.

```

SELECT o.* , a.*
FROM orders o
JOIN accounts a
ON o.account_id = a.id;

```

```

43
44 /*When performing JOINS, it is easiest to give table names aliases. The aliases for a table will be created in the from
45 and join clauses. Once we have given these aliases, we can refer to columns in the select clause by using those alias names.
46 Generally, the first character of the table name is chosen as the alias.*/
47 SELECT o.* , a.*
48 FROM orders o
49 JOIN accounts a
50 ON o.account_id = a.id;
51

```

| id | account_id | occurred_at         | standard_qty | gloss_qty | poster_qty | total  | standard_amt_usd | gloss_amt_usd | poster_amt_usd | total_amt_usd | id      | name        |
|----|------------|---------------------|--------------|-----------|------------|--------|------------------|---------------|----------------|---------------|---------|-------------|
| 1  | 1001       | 2015-10-06 17:31:14 | 123          | 22        | 169        | 613.77 | 164.78           | 194.88        | 973.43         | 1001          | Walmart |             |
| 2  | 2          | 2015-11-05 03:34:33 | 190          | 41        | 57         | 288    | 948.10           | 307.09        | 462.84         | 1718.03       | 1001    | Walmart     |
| 3  | 3          | 2015-12-04 04:21:55 | 85           | 47        | 0          | 132    | 424.15           | 352.03        | 0.00           | 776.18        | 1001    | Walmart     |
| 4  | 4          | 2016-01-02 01:18:24 | 144          | 32        | 0          | 176    | 718.55           | 239.68        | 0.00           | 958.24        | 1001    | Walmart     |
| 5  | 5          | 2016-02-01 19:27:27 | 108          | 29        | 28         | 165    | 538.92           | 217.21        | 227.36         | 963.49        | 1001    | Walmart     |
| 6  | 6          | 2016-03-20 15:55:51 | 103          | 24        | 46         | 173    | 513.97           | 179.76        | 373.52         | 1067.25       | 1001    | Walmart     |
| 7  | 7          | 2016-04-01 11:20:18 | 101          | 39        | 92         | 226    | 503.99           | 247.17        | 747.04         | 1498.20       | 1001    | Walmart     |
| 8  | 8          | 2016-05-01 15:55:51 | 95           | 47        | 151        | 293    | 474.05           | 352.03        | 1226.12        | 2052.20       | 1001    | Walmart     |
| 9  | 9          | 2016-05-31 21:22:48 | 91           | 16        | 22         | 129    | 454.09           | 119.84        | 178.64         | 752.57        | 1001    | Walmart     |
| 10 | 10         | 2016-06-30 12:32:05 | 94           | 46        | 8          | 148    | 469.06           | 344.54        | 64.96          | 878.56        | 1001    | Walmart     |
| 11 | 11         | 2016-07-30 03:26:30 | 101          | 36        | 0          | 137    | 503.99           | 269.64        | 0.00           | 773.63        | 1001    | Walmart     |
| 12 | 12         | 2016-08-28 07:13:39 | 124          | 33        | 39         | 196    | 618.76           | 247.17        | 316.68         | 1182.41       | 1001    | Walmart     |
| 13 | 13         | 2016-09-26 23:28:25 | 104          | 10        | 44         | 158    | 518.96           | 74.90         | 357.28         | 951.14        | 1001    | Walmart     |
| 14 | 14         | 2016-10-26 20:31:30 | 97           | 143       | 54         | 294    | 484.03           | 1071.07       | 436.40         | 1993.58       | 1001    | Walmart     |
| 15 | 15         | 2016-11-25 23:21:32 | 127          | 39        | 44         | 210    | 633.73           | 292.11        | 357.28         | 1283.12       | 1001    | Walmart     |
| 16 | 16         | 2016-12-24 05:53:13 | 123          | 127       | 19         | 269    | 613.77           | 951.23        | 154.28         | 1719.28       | 1001    | Walmart     |
| 17 | 17         | 2017-01-10 22:10:59 | 527          | 14        | 0          | 541    | 2629.73          | 104.86        | 0.00           | 2734.59       | 1011    | Exxon Mobil |
| 18 | 18         | 2017-01-10 22:15:56 | 516          | 23        | 0          | 539    | 2574.84          | 172.27        | 0.00           | 2747.11       | 1021    | Apple       |

Total rows: 1000 of 6912 Query complete 00:00:00.220

- Provide a table for all **web\_events** associated with account **name** of Walmart. There should be three columns. Be sure to include the primary\_poc, time of the event, and the channel for each event. Additionally, you might choose to add a fourth column to assure only Walmart events were chosen.

```

SELECT accounts.name, accounts.primary_poc, web_events.occurred_at,
web_events.channel
FROM accounts
JOIN web_events
ON accounts.id = web_events.account_id;

```

The screenshot shows the pgAdmin 4 interface. The left sidebar has an 'Object Explorer' tree with nodes like Functions, Materialized Views, Operators, Procedures, Sequences, Tables (5), and accounts. The 'accounts' node is expanded, showing columns: id, name, website, lat, long, primary\_poc, and sales\_rep\_id. The main area is a query editor with the following SQL code:

```

24 /*
25 8. Provide a table for all web_events associated with account name of Walmart. There should be three columns. Be sure to include
26 only Walmart events were chosen.
27 */
28 SELECT accounts.name, accounts.primary_poc, web_events.occurred_at, web_events.channel
29 FROM accounts
30 JOIN web_events
31 ON accounts.id = web_events.account_id
32 WHERE accounts.name LIKE 'Walmart';

```

Below the code is a 'Data Output' tab showing the results of the query:

|    | name    | primary_poc | occurred_at         | channel  |
|----|---------|-------------|---------------------|----------|
| 1  | Walmart | Tamara Tuma | 2015-10-06 17:13:58 | direct   |
| 2  | Walmart | Tamara Tuma | 2015-11-05 08:08:26 | direct   |
| 3  | Walmart | Tamara Tuma | 2015-12-04 03:57:24 | direct   |
| 4  | Walmart | Tamara Tuma | 2016-01-02 00:55:03 | direct   |
| 5  | Walmart | Tamara Tuma | 2016-02-01 19:02:33 | direct   |
| 6  | Walmart | Tamara Tuma | 2016-03-02 15:15:22 | direct   |
| 7  | Walmart | Tamara Tuma | 2016-04-01 10:58:55 | direct   |
| 8  | Walmart | Tamara Tuma | 2016-05-01 15:26:44 | direct   |
| 9  | Walmart | Tamara Tuma | 2016-05-31 20:53:47 | direct   |
| 10 | Walmart | Tamara Tuma | 2016-06-30 12:09:45 | direct   |
| 11 | Walmart | Tamara Tuma | 2016-07-30 03:09:26 | direct   |
| 12 | Walmart | Tamara Tuma | 2016-08-28 06:42:42 | direct   |
| 13 | Walmart | Tamara Tuma | 2016-09-26 23:14:59 | direct   |
| 14 | Walmart | Tamara Tuma | 2016-10-26 20:21:09 | direct   |
| 15 | Walmart | Tamara Tuma | 2016-11-25 22:52:29 | direct   |
| 16 | Walmart | Tamara Tuma | 2016-12-24 05:35:14 | direct   |
| 17 | Walmart | Tamara Tuma | 2015-10-06 04:22:11 | facebook |

Total rows: 39 of 39    Query complete 00:00:00.081

The system tray at the bottom shows a weather icon (16°C, Mostly clear), a search bar, and various system icons.

- Provide a table that provides the **region** for each **sales\_rep** along with their associated **accounts**. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

```

SELECT regions.name, sales_reps.name, accounts.name
FROM regions
JOIN sales_reps
ON regions.id = sales_reps.region_id
JOIN accounts
ON sales_reps.id = accounts.sales_rep_id
ORDER BY accounts.name;

```

pgAdmin 4

File Object Tools Help

Object Explorer

```

63 /*
64 9. Provide a table that provides the region for each sales_rep along with their associated accounts. Your final table should include
65 three columns: the region name, the sales rep name, and the account name. Sort the accounts alphabetically (A-Z) according to account name.
66 */
67 SELECT region.name AS region_name, sales_reps.name AS sales_reps_name, accounts.name AS accounts_name
68 FROM region
69 JOIN sales_reps
70 ON region.id = sales_reps.region_id
71 JOIN accounts
72 ON sales_reps.id = accounts.sales_rep_id
73 ORDER BY accounts.name;

```

Data Output

| region_name | sales_reps_name     | accounts_name            |
|-------------|---------------------|--------------------------|
| Northeast   | Sibyl Lauria        | 3M                       |
| Midwest     | Chau Rowles         | Abbott Laboratories      |
| Midwest     | Julie Starr         | AbbVie                   |
| Southeast   | Earlie Schleusner   | ADP                      |
| West        | Marquette Laycock   | Advance Auto Parts       |
| Southeast   | Moon Torian         | AECOM                    |
| Southeast   | Calvin Ellison      | AES                      |
| Northeast   | Renetta Carew       | Aetna                    |
| Midwest     | Cliff Meints        | Aflac                    |
| Northeast   | Elba Felder         | AIG                      |
| West        | Georgianna Chisholm | Air Products & Chemicals |
| Midwest     | Chau Rowles         | Alcoa                    |
| Northeast   | Julia Behrman       | Allstate                 |
| West        | Georgianna Chisholm | Ally Financial           |
| Northeast   | Tia Amato           | Alphabet                 |
| Midwest     | Charles Bidwell     | Altria Group             |

Total rows: 351 of 351 Query complete 00:00:00.081

15°C Mostly cloudy 21:46 27-04-2024 ENG IN

10. Provide the **name** for each region for every **order**, as well as the **account name** and the **unit price** they paid (**total\_amt\_usd/total**) for the order. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. A few accounts have 0 for **total**, so I divided by (**total + 0.01**) to assure not dividing by zero.

```

SELECT region.name AS region_name, accounts.name AS account_name,
(orders.total_amt_usd/(orders.total+ 0.01)) AS unit_price
FROM region
JOIN sales_reps ON region.id = sales_reps.region_id
JOIN accounts ON sales_reps.id = accounts.sales_rep_id
JOIN orders ON accounts.id = orders.account_id;

```

pgAdmin 4

File Object Tools Help

Object Explorer

```

74 /*
75 10. Provide the name for each region for every order, as well as the account name and the unit price they paid (total_amt_usd/total) for
76 the order. Your final table should have 3 columns: region name, account name, and unit price. A few accounts have 0 for total, so I divided
77 by (total + 0.01) to assure not dividing by zero.
78 */
79
80 SELECT region.name AS region_name, accounts.name AS account_name, (orders.total_amt_usd/(orders.total+ 0.01)) AS unit_price
81 FROM region
82 JOIN sales_reps ON region.id = sales_reps.region_id
83 JOIN accounts ON sales_reps.id = accounts.sales_rep_id
84 JOIN orders ON accounts.id = orders.account_id;

```

Data Output

| region_name | account_name | unit_price          |
|-------------|--------------|---------------------|
| Northeast   | Walmart      | 5.7596000236672386  |
| Northeast   | Walmart      | 5.965174820318789   |
| Northeast   | Walmart      | 5.8797060828725907  |
| Northeast   | Walmart      | 5.4442381229475998  |
| Northeast   | Walmart      | 5.9601842312587116  |
| Northeast   | Walmart      | 6.1687185711908566  |
| Northeast   | Walmart      | 6.62899102252112738 |
| Northeast   | Walmart      | 7.0038545236681342  |
| Northeast   | Walmart      | 5.833423765995959   |
| Northeast   | Walmart      | 5.9358151476251605  |
| Northeast   | Walmart      | 5.6465221516677615  |
| Northeast   | Walmart      | 6.0334166624151829  |
| Northeast   | Walmart      | 6.0194924371875198  |
| Northeast   | Walmart      | 6.7806537192612496  |
| Northeast   | Walmart      | 6.1098042950335998  |
| Northeast   | Walmart      | 6.3911378759153935  |

Total rows: 1000 of 6912 Query complete 00:00:00.062

15°C Cloudy 21:57 27-04-2024 ENG IN

**Every JOIN we have done up to this point has been an INNER JOIN.** That is, we have always pulled rows only if they exist as a match across two tables.

**full outer join** - will return the inner join result set, as well as any unmatched rows from either of the two tables being joined.

- Suppose you want to check all the orders that were brought by one particular sales rep.

Imagine yourself as a sales rep at Parch and Posey. You'd like to make it easy for a sales rep to find her own data deals rather than having to sift through all the orders and try and remember which were hers. sales\_rep\_id isn't in the orders table so join is necessary to get that information. The first thing to do this is by using the WHERE clause:

```
SELECT orders.*, accounts.*
```

```
FROM orders
```

```
LEFT JOIN accounts ON orders.account_id = accounts.id
```

```
WHERE accounts.sales_rep_id = 321500;
```

| id                                                 | order_qty | gloss_qty | poster_qty | total | standard_amt_usd | gloss_amt_usd | poster_amt_usd | total_amt_usd | id      | name            | website         | lat             | long         | primary_poc  | salesrep_id |        |
|----------------------------------------------------|-----------|-----------|------------|-------|------------------|---------------|----------------|---------------|---------|-----------------|-----------------|-----------------|--------------|--------------|-------------|--------|
| 101                                                | 123       | 22        | 24         | 169   | 613.77           | 164.78        | 973.43         | 1001          | Walmart | www.walmart.com | 40.23849561     | -75.10329704    | Tamara Tuma  | 321500       |             |        |
| 102                                                | 190       | 41        | 57         | 288   | 948.10           | 307.09        | 462.84         | 1718.03       | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 103                                                | 85        | 47        | 0          | 132   | 424.15           | 352.03        | 0.00           | 776.18        | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 104                                                | 144       | 32        | 0          | 176   | 718.56           | 239.68        | 0.00           | 958.24        | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 105                                                | 108       | 29        | 28         | 165   | 538.92           | 217.21        | 227.36         | 983.49        | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 106                                                | 103       | 24        | 46         | 173   | 513.97           | 179.76        | 373.52         | 1067.25       | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 107                                                | 101       | 33        | 92         | 226   | 503.99           | 247.17        | 747.04         | 1498.20       | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 108                                                | 95        | 47        | 151        | 293   | 474.05           | 352.03        | 1226.12        | 2052.20       | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 109                                                | 91        | 16        | 22         | 129   | 454.09           | 119.84        | 178.64         | 792.57        | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 110                                                | 94        | 46        | 8          | 148   | 469.06           | 344.54        | 64.96          | 878.56        | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 111                                                | 101       | 36        | 0          | 137   | 503.99           | 269.64        | 0.00           | 773.63        | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 112                                                | 124       | 33        | 39         | 196   | 618.76           | 247.17        | 316.68         | 1182.61       | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 113                                                | 104       | 10        | 44         | 158   | 518.96           | 74.90         | 357.28         | 951.14        | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 114                                                | 97        | 143       | 54         | 294   | 484.03           | 1071.07       | 438.48         | 1993.58       | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 115                                                | 127       | 39        | 44         | 210   | 633.73           | 292.11        | 357.28         | 1283.12       | 1001    | Walmart         | www.walmart.com | 40.23849561     | -75.10329704 | Tamara Tuma  | 321500      |        |
| 116                                                | 16        | 173       | 17         | 16    | 769              | 619.77        | 691.93         | 142.98        | 1719.98 | 1001            | Walmart         | www.walmart.com | 40.23849561  | -74.10329704 | Tamara Tuma | 321500 |
| Total rows: 134 of 134 Query complete 00:00:00.051 |           |           |            |       |                  |               |                |               |         |                 |                 |                 | Ln 86, Col 1 |              |             |        |

As you can see the query will have only the orders that have 321500 in the sales\_rep\_id. But what if we need to keep all the other results order as well (BY changing the WHERE to AND)

- Provide a table that provides the region for each sales\_rep along with their associated accounts. This time only for the Midwest region. Your final table should include three columns: the region name, the sales rep name, and the account name. Sort the accounts alphabetically (A-Z) according to account name.

```
SELECT region.name AS region_name, sales_reps.name AS sales_rep_name, accounts.name AS accounts_name
FROM region
JOIN sales_reps ON region.id = sales_reps.region_id
JOIN accounts ON sales_reps.id = accounts.sales_rep_id
WHERE region.name = 'Midwest'
ORDER BY accounts.name;
```

pgAdmin 4

File Object Tools Help

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

Query Query History

```

96
97 /*
98 12. Provide a table that provides the region for each sales_rep along with their associated accounts. This time only for the Midwest region.
99 Your final table should include three columns: the region name, the sales rep name, and the account name. Sort the accounts alphabetically (A-Z)
100 according to account name.
101 */
102 SELECT region.name AS region_name, sales_reps.name AS sales_reps_name, accounts.name AS accounts_name
103 FROM region
104 JOIN sales_reps ON region.id = sales_reps.region_id
105 JOIN accounts ON sales_reps.id = accounts.sales_rep_id
106 WHERE region.name = 'Midwest'
107 ORDER BY accounts.name;
108
109 Data Output Messages Notifications
110
111
112

region_name	sales_reps_name	accounts_name
1	Midwest	Chas Rowles
2	Midwest	Julie Starr
3	Midwest	Cliff Meints
4	Midwest	Chas Rowles
5	Midwest	Charles Bidwell
6	Midwest	Delilah Krum
7	Midwest	Charles Bidwell
8	Midwest	Delilah Krum
9	Midwest	Delilah Krum
10	Midwest	Cordell Rieder
11	Midwest	Sherlene Wetherington
12	Midwest	Delilah Krum
13	Midwest	Carletta Kosinski
14	Midwest	Carletta Kosinski
15	Midwest	Cordell Rieder
16	Midwest	Cliff Meints

113 Total rows: 48 of 48 Query complete 00:00:00.047

```

15°C Cloudy Search ENG IN 22:50 27-04-2024

13. Provide a table that provides the **region** for each **sales\_rep** along with their associated **accounts**. This time only for accounts where the sales rep has a first name starting with S and in the Midwest region. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

```

SELECT region.name AS region_name, sales_reps.name AS sales_reps_name, accounts.name
AS accounts_name
FROM sales_reps
JOIN region ON sales_reps.region_id = region.id
JOIN accounts ON accounts.sales_rep_id = sales_reps.id
WHERE sales_reps.name LIKE 'S%' AND region.name = 'Midwest'
ORDER BY accounts.name;

```

pgAdmin 4

File Object Tools Help

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

Query Query History

```

109 /*
110 13. Provide a table that provides the region for each sales_rep along with their associated accounts. This time only for accounts where the sales rep
111 has a first name starting with S and in the Midwest region. Your final table should include three columns: the region name, the sales rep name, and the
112 account name. Sort the accounts alphabetically (A-Z) according to account name.
113 */
114 SELECT region.name AS region_name, sales_reps.name AS sales_reps_name, accounts.name AS accounts_name
115 FROM sales_reps
116 JOIN region ON sales_reps.region_id = region.id
117 JOIN accounts ON accounts.sales_rep_id = sales_reps.id
118 WHERE sales_reps.name LIKE 'S%' AND region.name = 'Midwest'
119 ORDER BY accounts.name;
120
121 Data Output Messages Notifications
122
123
124

region_name	sales_reps_name	accounts_name
1	Midwest	Sherlene Wetherington
2	Midwest	Sherlene Wetherington
3	Midwest	Sherlene Wetherington
4	Midwest	Sherlene Wetherington
5	Midwest	Sherlene Wetherington

125 Total rows: 5 of 5 Query complete 00:00:00.112

```

15°C Cloudy Search ENG IN 23:13 27-04-2024

14. Provide a table that provides the **region** for each **sales\_rep** along with their associated **accounts**. This time only for accounts where the sales rep has a **last name** starting with K and in the Midwest region. Your final table should include three columns: the region **name**, the sales rep **name**, and the account **name**. Sort the accounts alphabetically (A-Z) according to account name.

```
SELECT region.name, sales_reps.name, accounts.name
FROM sales_reps
JOIN region ON sales_reps.region_id = region.id
JOIN accounts ON sales_reps.id = accounts.sales_rep_id
WHERE sales_reps.name LIKE '%K%' and region.name = 'Midwest'
ORDER BY accounts.name;
```

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

- File Object Tools Help**
- Dashboard Properties SQL Statistics Dependencies Dependents Processes**
- parch-and-posey/postgres@PostgreSQL 16\***
- Query History**
- Query** (selected)
- No limit**
- Scratch Pad**
- Query** (containing the provided SQL code)
- Data Output** (selected)
- Messages Notifications**
- region\_name sales\_rep\_name accounts\_name**
- character character character**

|    | region_name | sales_rep_name    | accounts_name                     |
|----|-------------|-------------------|-----------------------------------|
| 1  | Midwest     | Dellah Krum       | Amgen                             |
| 2  | Midwest     | Dellah Krum       | AutoNation                        |
| 3  | Midwest     | Dellah Krum       | Capital One Financial             |
| 4  | Midwest     | Dellah Krum       | Cummins                           |
| 5  | Midwest     | Carletta Kosinski | Danaher                           |
| 6  | Midwest     | Carletta Kosinski | Dollar General                    |
| 7  | Midwest     | Kathleen Lalonde  | EMC                               |
| 8  | Midwest     | Dellah Krum       | Hartford Financial Services Group |
| 9  | Midwest     | Carletta Kosinski | International Paper               |
| 10 | Midwest     | Dellah Krum       | Kimberly-Clark                    |
| 11 | Midwest     | Kathleen Lalonde  | Kraft Heinz                       |
| 12 | Midwest     | Carletta Kosinski | McDonald's                        |
| 13 | Midwest     | Carletta Kosinski | Northrop Grumman                  |
| 14 | Midwest     | Kathleen Lalonde  | Penske Automotive Group           |
| 15 | Midwest     | Dellah Krum       | Plains GP Holdings                |
| 16 | Midwest     | Dellah Krum       | Southwest Airlines                |
| 17 | Midwest     | Kathleen Lalonde  | US Foods Holding                  |

Total rows: 17 of 17    Query complete 00:00:00.100

System tray icons: 15°C Cloudy, Search, File Explorer, Mail, Task View, Power, ENG IN, WiFi, Battery, 23:27, 27-04-2024, Notifications.

15. Provide the **name** for each region for every **order**, as well as the account **name** and the **unit price** they paid (total\_amt\_usd/total) for the order. However, you should only provide the results if the **standard order quantity** exceeds 100. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. In order to avoid a division by zero error, adding .01 to the denominator here is helpful total\_amt\_usd/(total+0.01).

```
SELECT region.name AS region_name, accounts.name AS accounts_name,
orders.total_amt_usd/(orders.total+0.01) AS unit_price
FROM region
JOIN sales_reps ON sales_reps.region_id = region.id
JOIN accounts ON accounts.sales_rep_id = sales_reps.id
JOIN orders on orders.account_id = accounts.id
WHERE orders.standard_qty > 100;
```

```

133 /*+
134 15. Provide the name for each region for every order, as well as the account name and the unit price they paid (total_amt_usd/total) for the order.
135 However, you should only provide the results if the standard order quantity exceeds 100. Your final table should have 3 columns: region name, account name,
136 and unit price. In order to avoid a division by zero error, adding .01 to the denominator here is helpful: total_amt_usd/(total+0.01).
137
138 SELECT region.name AS region_name, accounts.name AS accounts_name, orders.total_amt_usd/(orders.total+0.01) AS unit_price
139 FROM region
140 JOIN sales_reps ON sales_reps.region_id = region.id
141 JOIN accounts ON accounts.sales_rep_id = sales_reps.id
142 JOIN orders ON orders.account_id = accounts.id
143 WHERE orders.standard_qty > 100;
144
145
146 Data Output Messages Notifications
147
148
149 region_name account_name unit_price
150 character character numeric
151 1 Northeast Walmart 5.7594000236672386
152 2 Northeast Walmart 5.9651748203187389
153 3 Northeast Walmart 5.4442351229475998
154 4 Northeast Walmart 5.9601842312587116
155 5 Northeast Walmart 6.1687185711698566
156 6 Northeast Walmart 6.628910225212738
157 7 Northeast Walmart 6.6465321516677615
158 8 Northeast Walmart 6.0394166624151829
159 9 Northeast Walmart 6.01949424371875198
160 10 Northeast Walmart 6.1098042950335698
161 11 Northeast Walmart 6.3911378759153935
162 12 Northeast Exxon Mobil 5.054601578520475
163 13 Northeast Apple 5.0965844789521593
164 14 Northeast Apple 5.263203164020000
165 15 Northeast Apple 5.1203150731136287
166 16 Northeast Apple 5.17609588687976
Total rows: 1000 of 4509 Query complete 00:00:00.114

```

16. Provide the **name** for each region for every **order**, as well as the **account name** and the **unit price** they paid (**total\_amt\_usd/total**) for the order. However, you should only provide the results if the **standard order quantity** exceeds 100 and the **poster order quantity** exceeds 50. Your final table should have 3 columns: **region name**, **account name**, and **unit price**. Sort for the smallest **unit price** first. In order to avoid a division by zero error, adding .01 to the denominator here is helpful (**total\_amt\_usd/(total+0.01)**).
- ```

SELECT region.name AS region_name, accounts.name AS accounts_name,
       orders.total_amt_usd/(orders.total+0.01) AS unit_price
  FROM region
  JOIN sales_reps ON sales_reps.region_id = region.id
  JOIN accounts ON accounts.sales_rep_id = sales_reps.id
  JOIN orders ON orders.account_id = accounts.id
 WHERE orders.standard_qty > 100 AND orders.poster_qty >50
 ORDER BY unit_price;

```

```

145 /*+
146 16. Provide the name for each region for every order, as well as the account name and the unit price they paid (total_amt_usd/total) for the order.
147 However, you should only provide the results if the standard order quantity exceeds 100 and the poster order quantity exceeds 50. Your final table
148 should have 3 columns: region name, account name, and unit price. Sort for the smallest unit price first. In order to avoid a division by zero error,
149 adding .01 to the denominator here is helpful (total_amt_usd/(total+.01)).
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181

```

region.name	accounts.name	unit_price
1 Northeast	State Farm Insurance Cos.	5.1192822502542913
2 Southeast	DISH Network	5.2318158475403638
3 Northeast	Travelers Cos.	5.2351813313106532
4 Northeast	Best Buy	5.2604264379300265
5 West	Stanley Black & Decker	5.266395560729988
6 Northeast	Citigroup	5.2730812163197040
7 Southeast	BlackRock	5.2782704047278773
8 Southeast	Nucor	5.2890939504788515
9 Northeast	Merck	5.2963914415268702
10 Midwest	Sears Holdings	5.3009122462935962
11 Southeast	BlackRock	5.301769894729081
12 Northeast	Fannie Mae	5.3034650542924098
13 Midwest	Aflac	5.3055579626003016
14 Northeast	Northwestern Mutual	5.3079691144825591

Total rows: 835 of 835 Query complete 00:00:00.112 Ln 145, Col 1

17. What are the different **channels** used by **account id 1001**? Your final table should have only 2 columns: **account name** and the different **channels**. You can try **SELECT DISTINCT** to narrow down the results to only the unique values.

```

SELECT DISTINCT web_events.channel, accounts.name
FROM accounts
JOIN web_events ON accounts.id = web_events.account_id
WHERE accounts.id ='1001';

```

```

158 /*
159 17. What are the different channels used by account id 1001? Your final table should have only 2 columns: account name and the different channels.
160 You can try SELECT DISTINCT to narrow down the results to only the unique values.
161 */
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194

```

channel	name
1 advords	Walmart
2 banner	Walmart
3 direct	Walmart
4 facebook	Walmart
5 organic	Walmart
6 twitter	Walmart

Total rows: 6 of 6 Query complete 00:00:00.080 Ln 159, Col 1

18. Find all the orders that occurred in 2015. Your final table should have 4 columns:

occurred_at, account name, order total, and order total_amt_usd.

```

SELECT orders.occurred_at, accounts.name, orders.total, orders.total_amt_usd
FROM accounts
JOIN orders ON orders.account_id = accounts.id

```

```
WHERE orders.occurred_at BETWEEN '01-01-2015' AND '31-12-2015'
ORDER BY orders.occurred_at DESC;
```

```

165 JOIN web_events ON accounts.id = web_events.account_id
166 WHERE accounts.id != 1001;
167 /*
168 18. Find all the orders that occurred in 2015. Your final table should have 4 columns: occurred_at, account name, order total, and order total_amt_usd.
169 */
170 SELECT orders.occurred_at, accounts.name, orders.total, orders.total_amt_usd
171 FROM accounts
172 JOIN orders ON orders.account_id = accounts.id
173 WHERE orders.occurred_at BETWEEN '01-01-2015' AND '31-12-2015'
174 ORDER BY orders.occurred_at DESC;
175
176
177 Data Output Messages Notifications
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
Total rows: 1000 of 1715 Query complete 00:00:00.152
Ln 168, Col 1

```

RECAP

Primary keys- are unique for every row in a table. These are generally the first column in our database

Foreign keys – are the primary keys appearing in another table, which allows the rows to be non-unique.

JOIN – an INNER JOIN that only pulls the data that exists in both tables.

LEFT JOIN- pulls all the data that exists in both tables, as well as all the rows from the table in the FROM even if they do not exist in the JOIN statement.

RIGHT JOIN- pulls all the data that exists in both tables, as well as all the rows from the table in the join even if they do not exist in the FROM statement.

ALIAS – We can alias tables and columns using AS or not using it. This allows you to be more efficient in the number of characters you need to write, while at the same time you can assume that the column headings are informative of the data in your table.

[UNION, UNION ALL, CROSS JOIN, SELF JOIN](#)