Finding Matched and Unmatched Rows with FULL OUTER JOIN

You're not likely to use `FULL JOIN` (which can also be written as `FULL OUTER JOIN`) too often, but the syntax is worth practicing anyway. `LEFT JOIN` and `RIGHT JOIN` each return unmatched rows from one of the tables—`FULL JOIN` returns unmatched rows from both tables. `FULL JOIN` is commonly used in conjunction with aggregations to understand the amount of overlap between two tables.

Say you're an analyst at Parch & Posey and you want to see:

- each account who has a sales rep and each sales rep that has an account (all of the columns in these returned rows will be full)
- but also each account that does not have a sales rep and each sales rep that does not have an account (some of the columns in these returned rows will be empty)

This type of question is rare, but `FULL OUTER JOIN` is perfect for it. In the following SQL Explorer, write a query with `FULL OUTER JOIN` to fit the above described Parch & Posey scenario (selecting all of the columns in both of the relevant tables, `accounts` and `sales_reps`) then answer the subsequent multiple choice quiz.

Input HISTORY MENU

SCHEMA

accounts
orders
region
sales_reps
web_events

```
1 SELECT * FROM accounts a
2 FULL OUTER JOIN sales_reps s ON s.id =
  a.sales_rep_id
3 WHERE a.sales_rep_id is NULL or s.id is NULL
```

Success! EVALUATE

Output 0 results

id	name	website	lat	long	primary_poc	sales_rep_id	region_id

Inequality JOINs

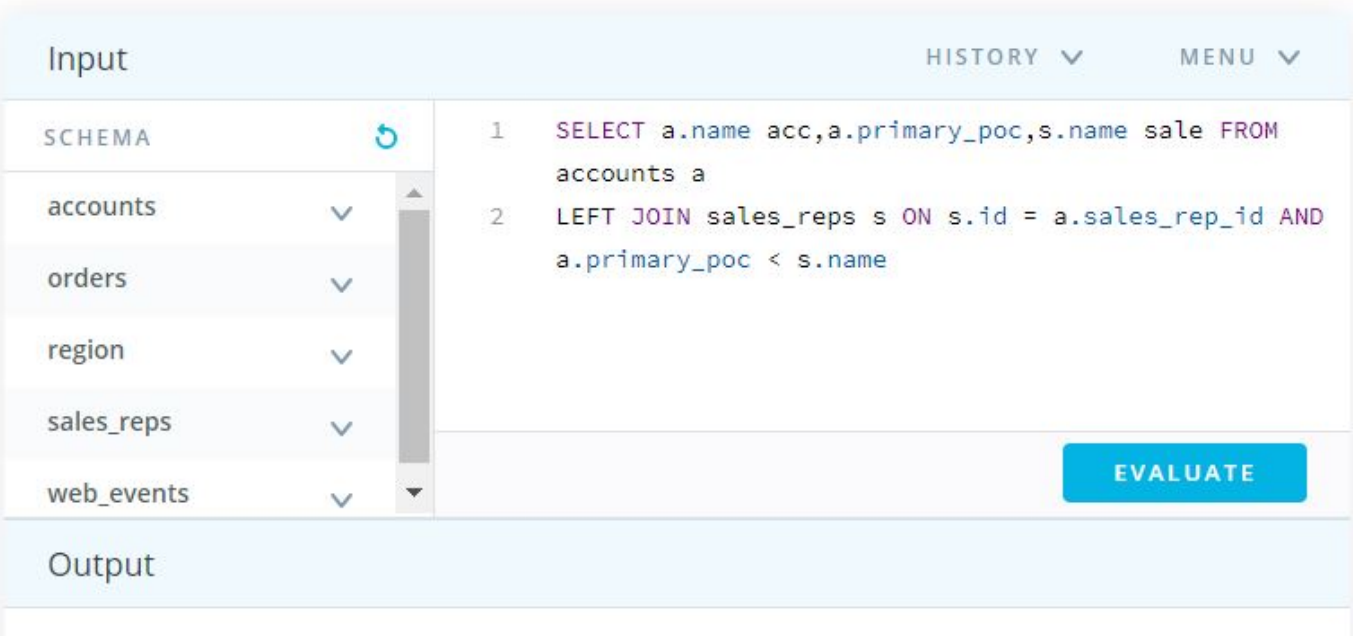
The query in Derek's video was pretty long. Let's now use a shorter query to showcase the power of joining with comparison operators.

Inequality operators (a.k.a. comparison operators) don't only need to be date times or numbers, they also work on strings! You'll see how this works by completing the following quiz, which will also reinforce the concept of joining with comparison operators.

In the following SQL Explorer, write a query that left joins the `accounts` table and the `sales_reps` tables on each sale rep's ID number *and* joins it using the `<` comparison operator on `accounts.primary_poc` and `sales_reps.name`, like so:

```
accounts.primary_poc < sales_reps.name
```

The query results should be a table with three columns: the account name (e.g. Johnson Controls), the primary contact name (e.g. Cammy Sosnowski), and the sales representative's name (e.g. Samuel Racine). Then answer the subsequent multiple choice question.



Appending Data via UNION

Write a query that uses `UNION ALL` on two instances (and selecting all columns) of the `accounts` table. Then inspect the results and answer the subsequent quiz.

Input

HISTORY MENU

SCHEMA

- accounts
- orders
- region
- sales_reps
- web_events

```
1  SELECT * FROM accounts a
2  UNION ALL
3  SELECT * FROM accounts a1
```

Success!

EVALUATE

Output 702 results

id	name	website	lat
1001	Walmart	www.walmart.com	40.2384
1011	Exxon Mobil	www.exxonmobil.com	41.1691
1021	Apple	www.apple.com	42.2904
1031	Berkshire Hathaway	www.berkshirehathaway.com	40.9490
1041	McKesson	www.mckesson.com	42.2170
1051	UnitedHealth Group	www.unitedhealthgroup.com	40.0879

Pretreating Tables before doing a UNION

Add a `WHERE` clause to each of the tables that you unioned in the query above, filtering the first table where `name` equals Walmart and filtering the second table where `name` equals Disney. Inspect the results then answer the subsequent quiz.

Input

HISTORY MENU

SCHEMA

- accounts
- orders
- region
- sales_reps
- web_events

```
SELECT * FROM accounts a
WHERE a.name = 'Walmart'
UNION ALL
SELECT * FROM accounts a1
WHERE a.name = 'Disney'
```

EVALUATE

Output 2 results

id	name	website	lat	long	primary_poc
1001	Walmart	www.walmart.com	40.23849561	-75.10329704	Tamara Tuma
1521	Disney	www.disney.com	41.87879976	-74.81102607	Timika Mistretta

SCHEMA

accounts

orders

region

sales_reps

web_events

```
WITH double_accounts AS (SELECT * FROM accounts a
UNION ALL
SELECT * FROM accounts a1)

SELECT COUNT(*), double_accounts.name FROM
double_accounts
GROUP BY 2
ORDER BY 1
```

EVALUATE

Output 351 results

2	Performance Food Group
2	Kellogg
2	Gap
2	BorgWarner
2	Disney
2	Whole Foods Market
2	HD Supply Holdings
2	Progressive
2	Level 3 Communications

↑ Menu Expand

Performing Operations on a Combined Dataset

Perform the union in your first query (under the **Appending Data via UNION** header) in a common table expression and name it `double_accounts`. Then do a `COUNT` the number of times a `name` appears in the `double_accounts` table. If you do this correctly, your query results should have a count of 2 for each `name`.