JOINS

By
Vijaya Nandini M

SQL

- Welcome to this section on JOINS.
- JOINS will allow us to combine information from multiple tables!
- Let’s see what we will learn in this section

SQL

- Section Overview
  - Creating an alias with the AS clause
  - Understanding different kinds of JOINs
    - INNER JOINS
    - OUTER JOINS
    - FULL JOINS
    - UNIONS
  - Challenge Tasks

AS

SQL

- Before we learn about JOINs , let’s quickly cover the AS clause which allows us to create an “alias” for a column or result.
- Let’s see the example syntax

SQL

- SELECT column AS new_name
  FROM table

SQL

- SELECT SUM(column) AS new_name FROM table

SQL

- Example

Query Editor | Query History

```
1  SELECT customer_id
2  FROM payment;
```

Data Output | Explain | Messages | Notifications

	customer_id smallint
1	341

SQL

- Example

Query Editor | Query History

```
1  SELECT amount AS rental_price
2  FROM payment;
```

Data Output | Explain | Messages | Notifications

	rental_price numeric (5,2)
1	7.99
2	1.99

SQL

- Example

Query Editor | Query History

```
1  SELECT SUM(amount) AS net_revenue
2  FROM payment;
```

Data Output | Explain | Messages | Notifications

	net_revenue numeric
1	61312.04

SQL

- The AS operator gets executed at the very end of a query, meaning that we can not use the ALIAS inside a WHERE operator.

INNER JOIN

SQL

- There are several types of JOINs, let’s go through the simplest JOIN type, which is an **INNER JOIN**

SQL

- What is a JOIN operation?
- JOINs allow us to combine multiple tables together.
- The main reason for the different JOIN types is to decide how to deal with information only present in one of the joined tables.

SQL

- Let’s imagine a simple example.
- Our company is holding a conference for people in the movie rental industry.
- We’ll have people register online beforehand and then login the day of the conference.

SQL

- After the conference we have these tables

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- The respective id columns indicate what order they registered or logged in on site.

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- For the sake of simplicity, we will assume the names are unique.

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- To help you keep track, Registrations names' first letters go A,B,C,D

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- An INNER JOIN will result with the set of records that match in both tables.

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- An INNER JOIN will result with the set of records that **match in both** tables.

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

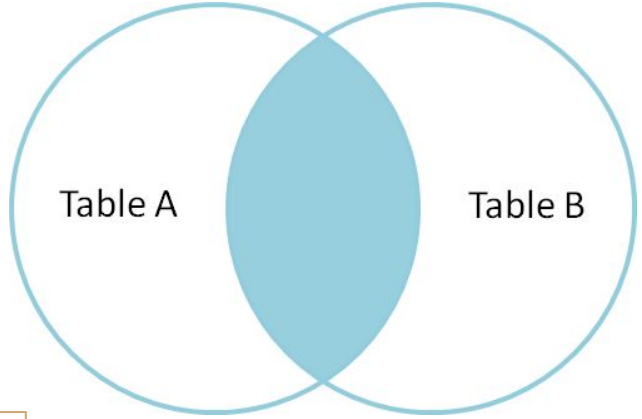
LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM **TableA**
  INNER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match

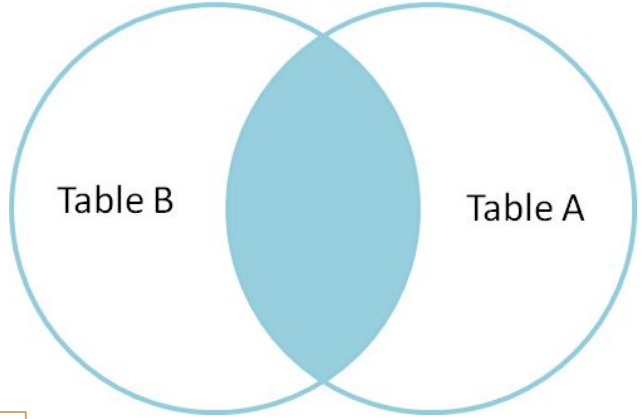
SQL

- SELECT * FROM **TableA**
  INNER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match



SQL

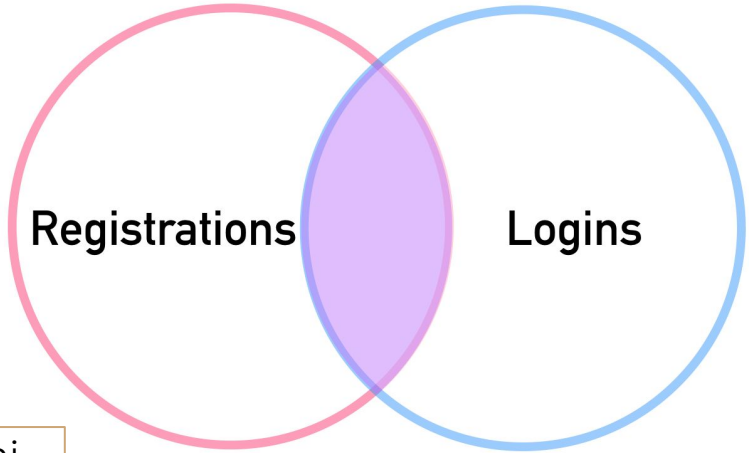
- SELECT * FROM **TableB**
  INNER JOIN **TableA**
  ON **TableA**.col_match = **TableB**.col_match



SQL

- SELECT * FROM Registrations
  INNER JOIN Logins
  ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

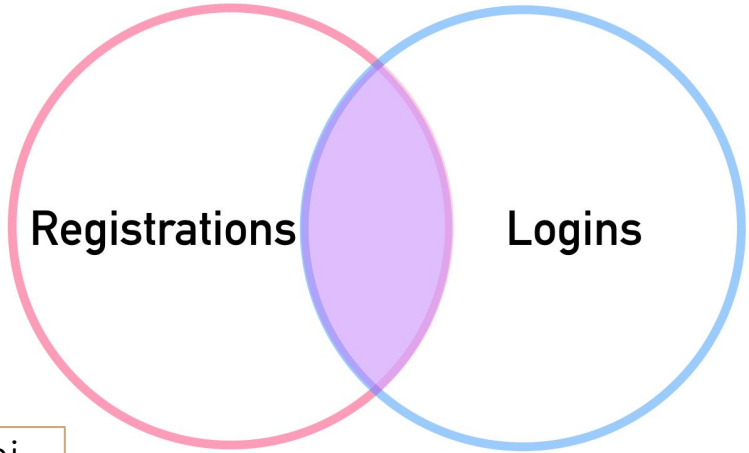


LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations
  INNER JOIN Logins
  ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

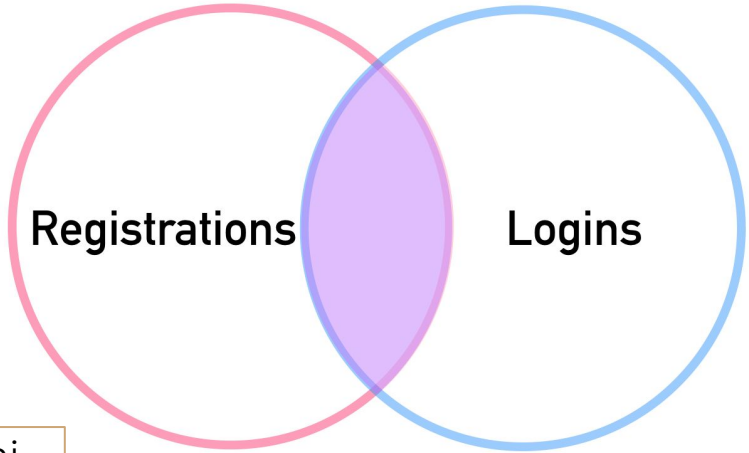


LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations
  INNER JOIN Logins
  ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David



LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations
  INNER JOIN Logins
  ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
1	Andrew	2	Andrew
2	Bob	4	Bob

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT reg_id,Logins.name,log_id
  FROM Registrations
  INNER JOIN Logins
  ON Registrations.name = Logins.name

RESULTS		
reg_id	name	log_id
1	Andrew	2
2	Bob	4

SQL

- SELECT reg_id,Logins.name, og_id
  FROM Registrations
  INNER JOIN Logins
  ON Registrations.name = Logins.name

RESULTS		
reg_id	name	log_id
1	Andrew	2
2	Bob	4

SQL

- Remember that table order won’t matter in an INNER JOIN
- Also if you see just JOIN without the INNER, PostgreSQL will treat it as an INNER JOIN.
- Let’s see some examples in pgAdmin!

OUTER JOINS

SQL

- There are few different types of OUTER JOINs
- They will allow us to specify how to deal with values only present in one of the tables being joined.
- These are the more complex JOINs, take your time when trying to understand them!

SQL

- Types of OUTER JOIN:
  - FULL OUTER JOIN
    - Clarifying WHERE null
  - LEFT OUTER JOIN
    - Clarifying WHERE null
  - RIGHT OUTER JOIN
    - Clarifying WHERE null

FULL OUTER JOIN

SQL

- Let’s review our two example tables from the previous lectures.

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- We know we would join these tables together on the name column

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- Recall we match Andrew and Bob in both tables

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- But we have names that only appear in one table!

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- Let’s see how the different OUTER JOINs deal with this discrepancy.

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

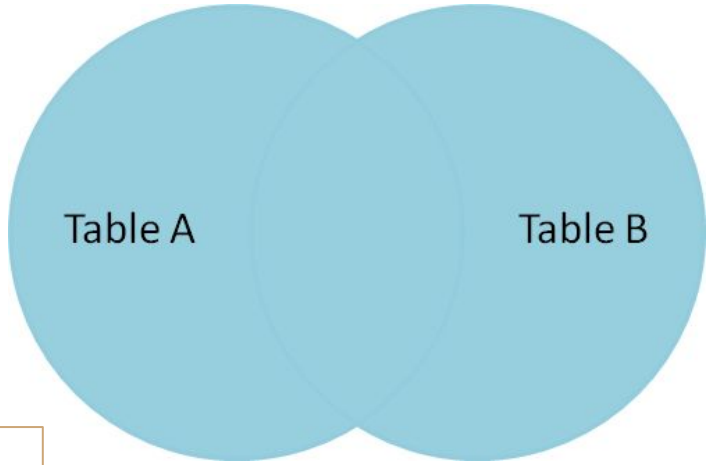
We will first take a look at the simplest, which is a **FULL OUTER JOIN**

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

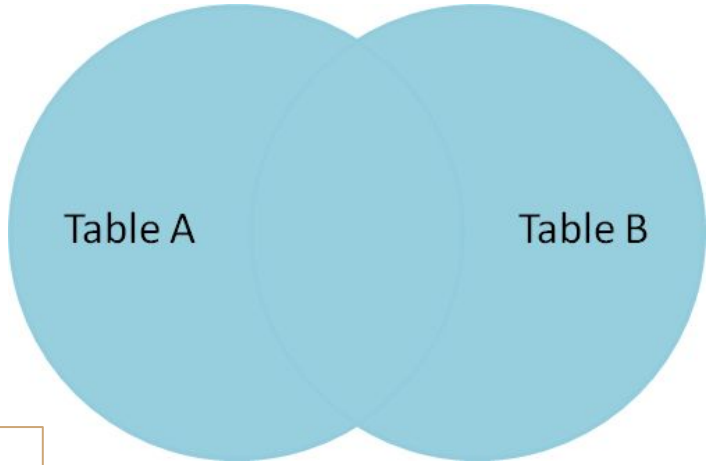
SQL

- SELECT * FROM **TableA**
  FULL OUTER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match



SQL

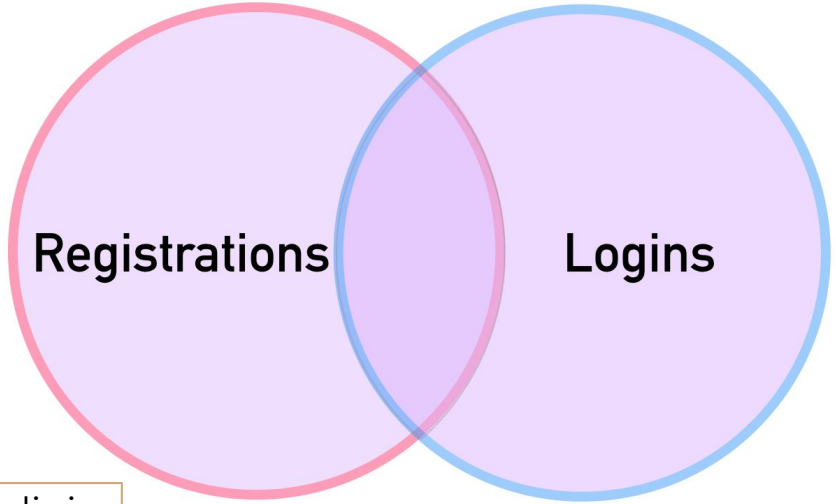
- SELECT * FROM **TableB**
  FULL OUTER JOIN **TableA**
  ON **TableA**.col_match = **TableB**.col_match



SQL

- SELECT * FROM Registrations
  FULL OUTER JOIN Logins
  ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David



LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations FULL OUTER JOIN Logins ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations FULL OUTER JOIN Logins ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
1	Andrew		
2	Bob		
3	Charlie		
4	David		

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations FULL OUTER JOIN Logins ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
1	Andrew	2	Andrew
2	Bob	4	Bob
3	Charlie		
4	David		

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations FULL OUTER JOIN Logins ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
1	Andrew	2	Andrew
2	Bob	4	Bob
3	Charlie	null	null
4	David	null	null

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations FULL OUTER JOIN Logins ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
1	Andrew	2	Andrew
2	Bob	4	Bob
3	Charlie	null	null
4	David	null	null
null	null	1	Xavier
null	null	3	Yolanda

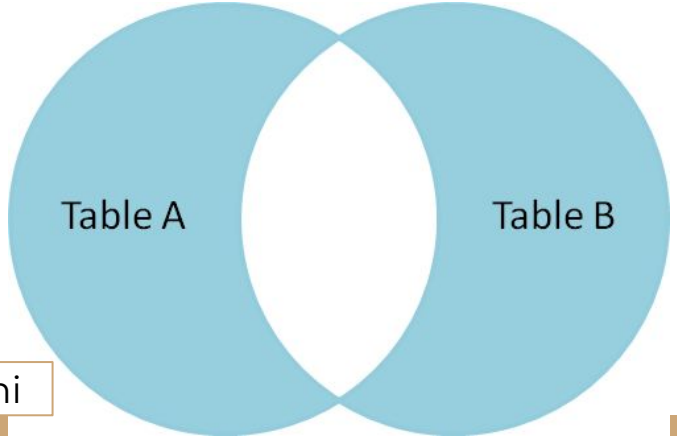
LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

FULL OUTER JOIN with WHERE

Get rows unique to either table
(rows not found in both tables)

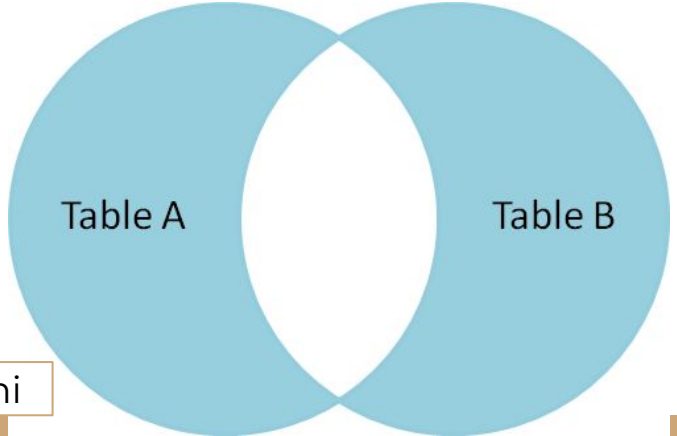
SQL

- SELECT * FROM **TableA**
  FULL OUTER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match
  WHERE **TableA**.id IS null OR
  TableB.id IS null



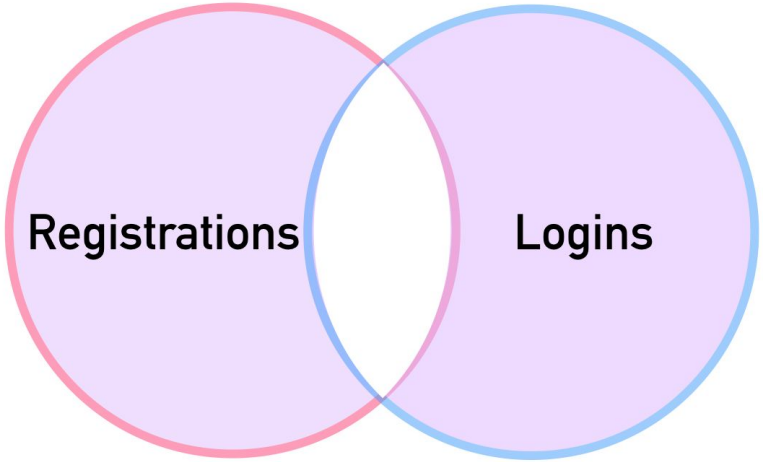
SQL

- SELECT * FROM **TableB**
  FULL OUTER JOIN **TableA**
  ON **TableA**.col_match = **TableB**.col_match
  WHERE **TableA**.id IS null OR
  TableB.id IS null



SQL

```
SELECT * FROM Registrations FULL OUTER
JOIN Logins
ON Registrations.name = Logins.name
WHERE Registrations.reg_id IS null OR
Logins.log_id IS null
```



SQL

- SELECT * FROM Registrations FULL OUTER JOIN Logins ON Registrations.name = Logins.name

REGISTRATIONS

reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS

reg_id	name	log_id	name
1	Andrew	2	Andrew
2	Bob	4	Bob
3	Charlie	null	null
4	David	null	null
null	null	1	Xavier
null	null	3	Yolanda

LOGINS

log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations FULL OUTER JOIN Logins ON Registrations.name = Logins.name WHERE Registrations.reg_id IS null OR Logins.log_id IS null

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
3	Charlie	null	null
4	David	null	null
null	null	1	Xavier
null	null	3	Yolanda

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

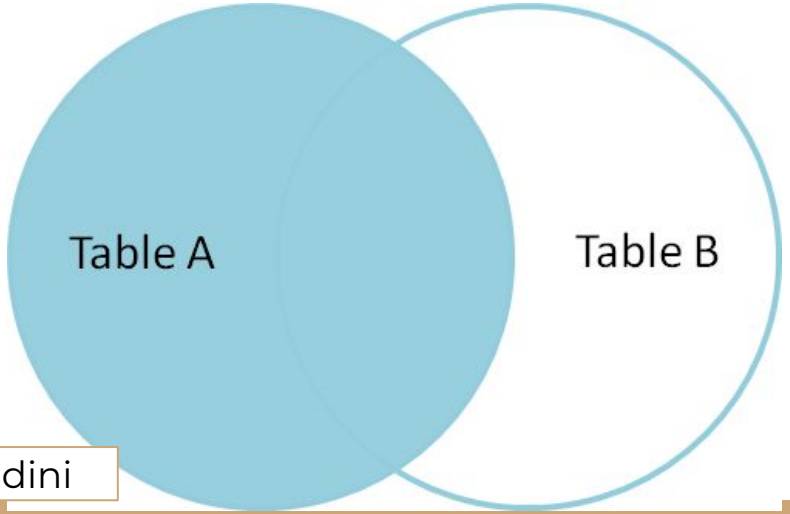
LEFT OUTER JOIN

SQL

- A LEFT OUTER JOIN results in the set of records that are in the left table, if there is no match with the right table, the results are null.
- Later on we will learn how to add WHERE statements to further modify a LEFT OUTER JOIN

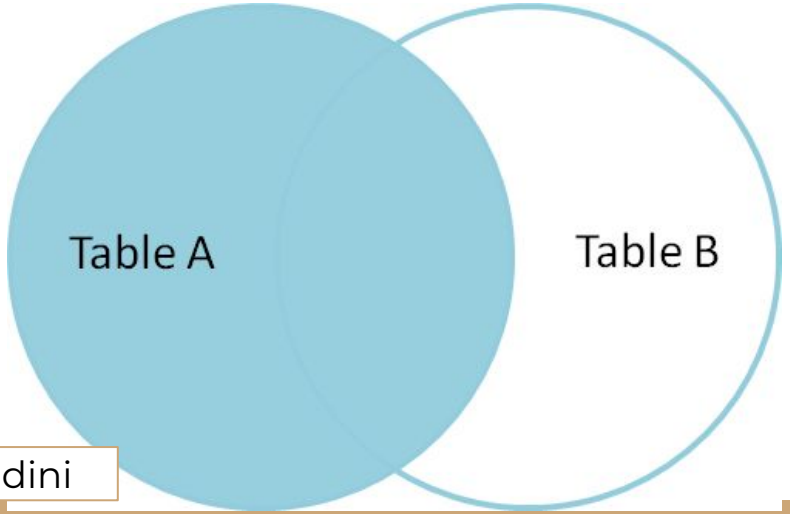
SQL

- SELECT * FROM **TableA**
  LEFT OUTER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match



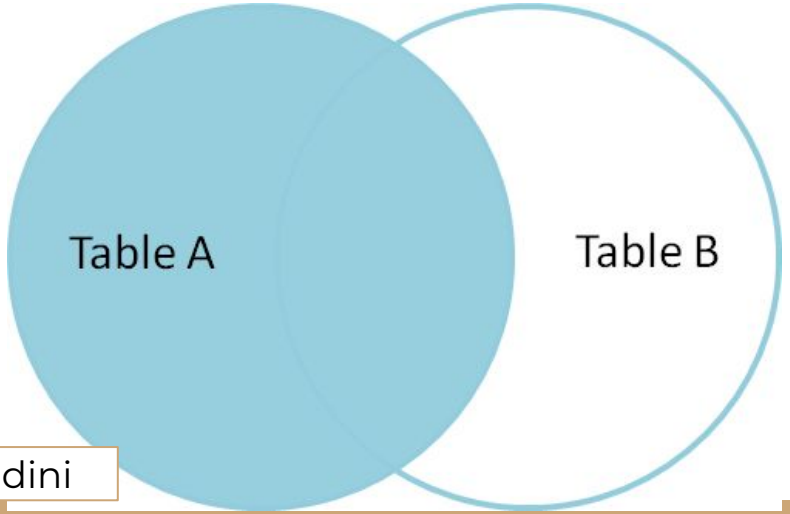
SQL

- SELECT * FROM **TableA**
  LEFT JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match



SQL

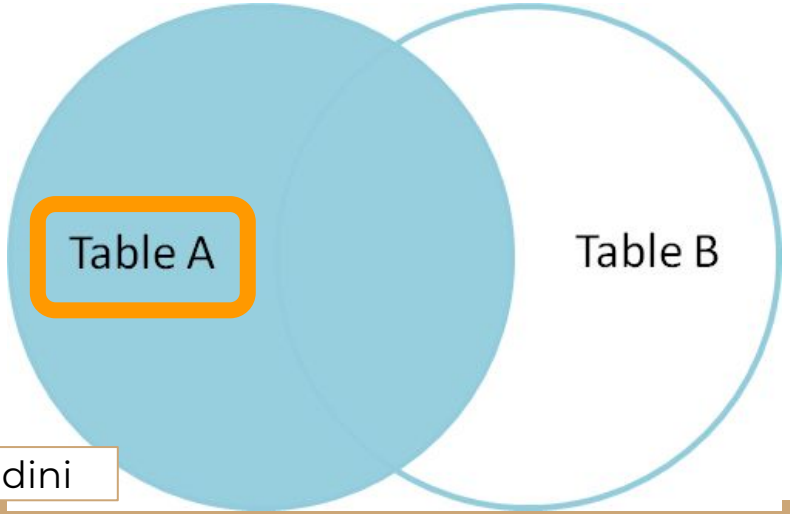
- SELECT * FROM **TableA**
  LEFT OUTER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match



SQL

ORDER MATTERS FOR LEFT OUTER JOIN!

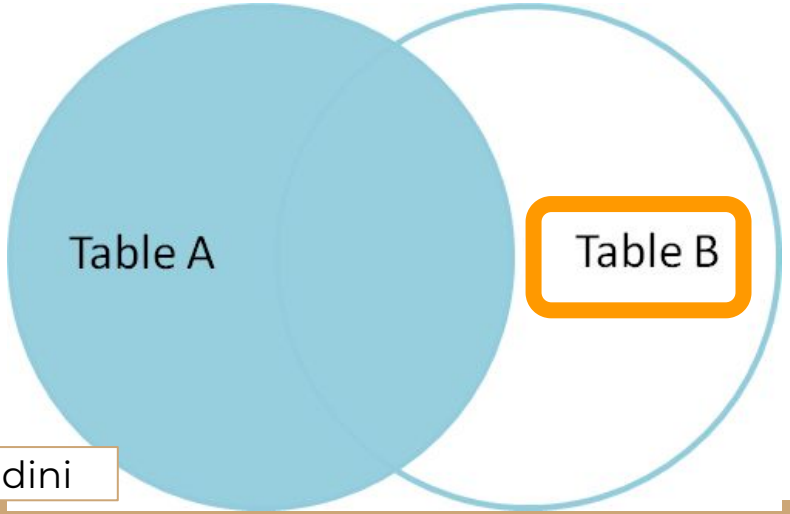
- SELECT * FROM **TableA**
  LEFT OUTER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match



SQL

ORDER MATTERS FOR LEFT OUTER JOIN!

- SELECT * FROM **TableA**
  LEFT OUTER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match



SQL

Let's explore a **LEFT OUTER JOIN** with our two example tables.

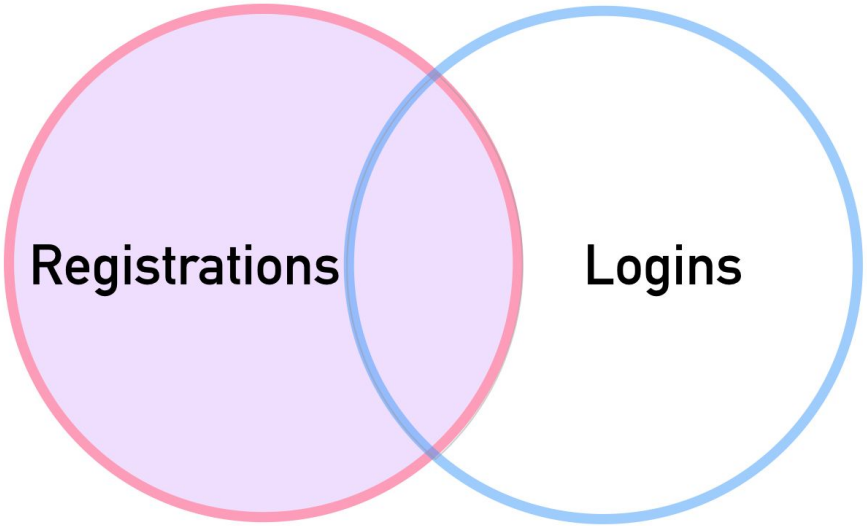
REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations
  LEFT OUTER JOIN Logins
  ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David



LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations
  LEFT OUTER JOIN Logins
  ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
1	Andrew	2	Andrew
2	Bob	4	Bob
3	Charlie	null	null
4	David	null	null

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

- SELECT * FROM Registrations
  LEFT OUTER JOIN Logins
  ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
1	Andrew	2	Andrew
2	Bob	4	Bob
3	Charlie	null	null
4	David	null	null

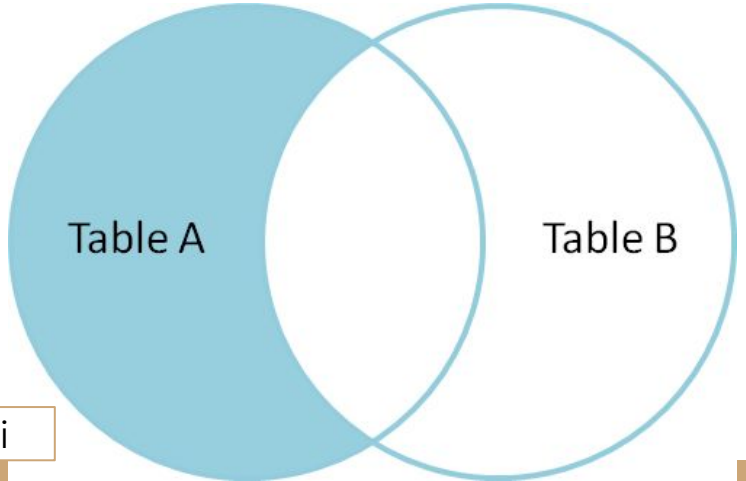
LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

LEFT OUTER JOIN With WHERE

Get rows unique to left table

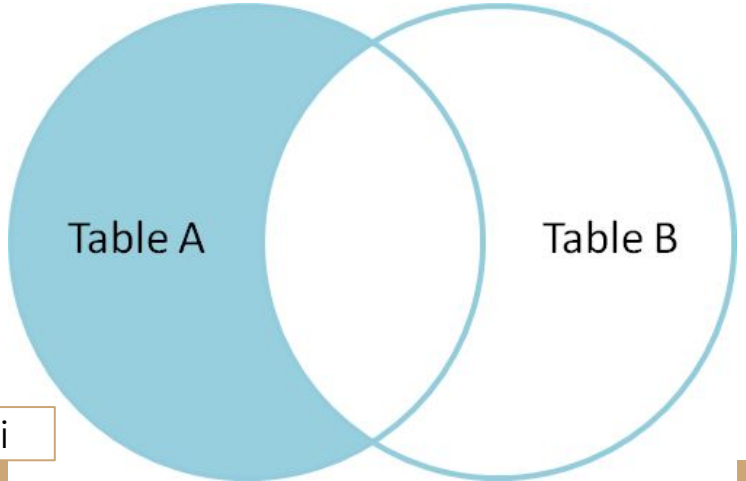
SQL

- What if we only wanted entries unique to Table A? Those rows found in Table A and **not** found in Table B.



SQL

- SELECT * FROM **TableA**
  LEFT OUTER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match
  WHERE **TableB**.id IS null



SQL

- SELECT * FROM Registrations
  LEFT OUTER JOIN Logins
  ON Registrations.name = Logins.name

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
1	Andrew	2	Andrew
2	Bob	4	Bob
3	Charlie	null	null
4	David	null	null

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

SQL

```
SELECT * FROM Registrations
LEFT OUTER JOIN Logins
ON Registrations.name = Logins.name
WHERE Logins.log_id IS null
```

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

RESULTS			
reg_id	name	log_id	name
3	Charlie	null	null
4	David	null	null

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

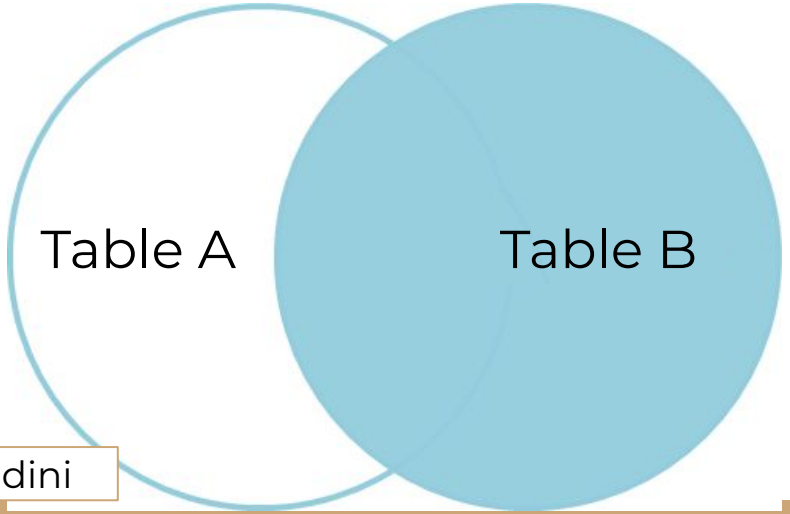
RIGHT JOINS

SQL

- A RIGHT JOIN is essentially the same as a LEFT JOIN, except the tables are switched.
- This would be the same as switching the table order in a LEFT OUTER JOIN.
- Let’s quickly see some examples of a RIGHT JOIN.

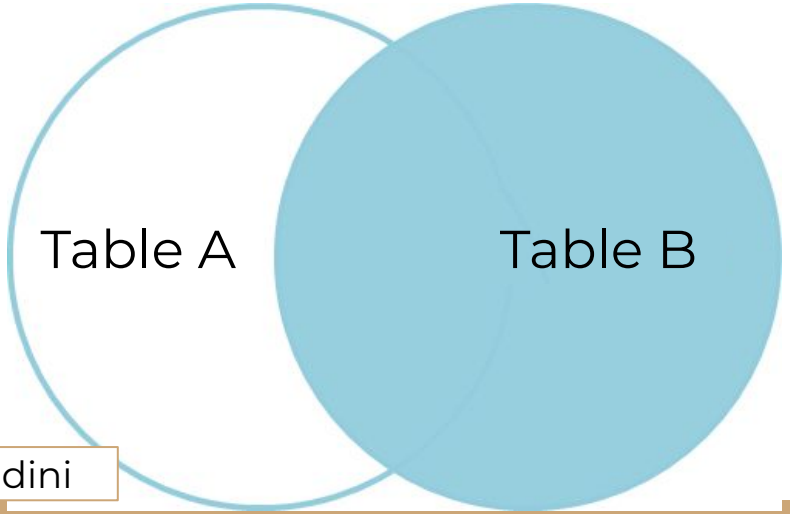
SQL

- SELECT * FROM **TableA**
  RIGHT OUTER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match



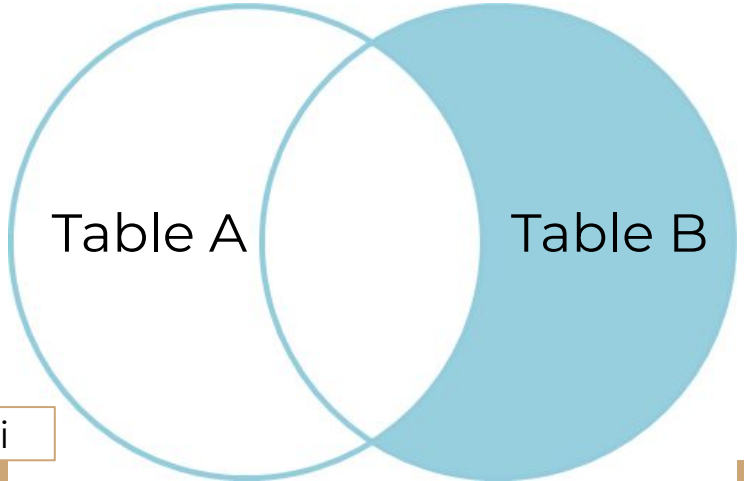
SQL

- SELECT * FROM **TableA**
  RIGHT JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match



SQL

- SELECT * FROM **TableA**
  RIGHT OUTER JOIN **TableB**
  ON **TableA**.col_match = **TableB**.col_match
  WHERE **TableA**.id IS null



SQL

- It is up to you and how you have the tables organized “in your mind” when it comes to choosing a LEFT vs RIGHT join, since depending on the table order you specify in the JOIN, you can perform duplicate JOINs with either method.

UNIONS

SQL

- The UNION operator is used to combine the result-set of two or more SELECT statements.
- It basically serves to directly concatenate two results together, essentially “pasting” them together.

SQL

- SELECT column_name(s) FROM table1
  UNION
  SELECT column_name(s) FROM table2;

SQL

Let’s explore a **UNION** with two example tables.

Sales2021_Q1	
name	amount
David	100
Claire	50

Sales2021_Q2	
name	amount
David	200
Claire	100

SQL

- SELECT * FROM Sales2021_Q1
  UNION
  SELECT * FROM Sales2021_Q2;

name	amount
David	100
Claire	50
David	200
Claire	100

SQL

- SELECT * FROM Sales2021_Q1
  UNION
  SELECT * FROM Sales2021_Q2
  ORDER BY name;

name	amount
David	100
David	200
Claire	50
Claire	100

JOIN

CHALLENGE TASKS

SQL

- California sales tax laws have changed and we need to alert our customers to this through email.
- What are the emails of the customers who live in California?

SQL

- Expected Results

	district character varying (20)	email character varying (50)
1	California	patricia.johnson@sakilacust...
2	California	betty.white@sakilacustomer...
3	California	alice.stewart@sakilacustom...
4	California	rosa.reynolds@sakilacusto...
5	California	renee.lane@sakilacustomer....
6	California	kristin.johnston@sakilacust...
7	California	cassandra.walters@sakilacu...
8	California	jacob.lance@sakilacustome...
9	California	rene.mcalister@sakilacusto...

SQL

- Hints
  - You will need to use the address and customer tables.
  - Look at the district column

SQL

- Solution
  - SELECT district,email FROM address INNER JOIN customer ON address.address_id = customer.address_id WHERE district = 'California'

SQL

- Challenge
  - A customer walks in and is a huge fan of the actor “Nick Wahlberg” and wants to know which movies he is in.
  - Get a list of all the movies “Nick Wahlberg” has been in.

SQL

- Expected Results

	title character varying (255)	first_name character varying (45)	last_name character varying (45)
1	Adaptation Holes	Nick	Wahlberg
2	Apache Divine	Nick	Wahlberg
3	Baby Hall	Nick	Wahlberg
4	Bull Shawshank	Nick	Wahlberg
5	Chainsaw Uptown	Nick	Wahlberg
...	...	...	...
21	Mask Peach	Nick	Wahlberg
22	Roof Champion	Nick	Wahlberg
23	Rushmore Mermaid	Nick	Wahlberg
24	Smile Earring	Nick	Wahlberg
25	Wardrobe Phantom	Nick	Wahlberg

SQL

- HINTS
  - You will need to do 2 JOINs in a row to do this in a single query.
  - Try using the online documentation or a simple google search to see how this is done.
  - Tables to use: actor, film, film_actor

SQL

- Solution
  - SELECT title,first_name,last_name
    FROM film_actor INNER JOIN actor
    ON film_actor.actor_id = actor.actor_id
    INNER JOIN film
    ON film_actor.film_id = film.film_id
    WHERE first_name = 'Nick'
    AND last_name = 'Wahlberg'