



JOINS



- 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



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



AS



- 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



- `SELECT column AS new_name
FROM table`



- `SELECT SUM(column) AS new_name
FROM table`



- Example

Query Editor

Query History

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

Data Output

Explain

Messages

Notifications

	customer_id smallint	
1	341	




- 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	



- 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	



- The AS operator gets executed at the very end of a query, meaning that we can not use the ALIAS inside a WHERE operator.
- Let's walk through a few examples in pgAdmin!



INNER JOIN



- There are several types of JOINS, in this lecture we will go through the simplest JOIN type, which is an **INNER JOIN**



- 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.



- 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.



- 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



- 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



- 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



- 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



- 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



- 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

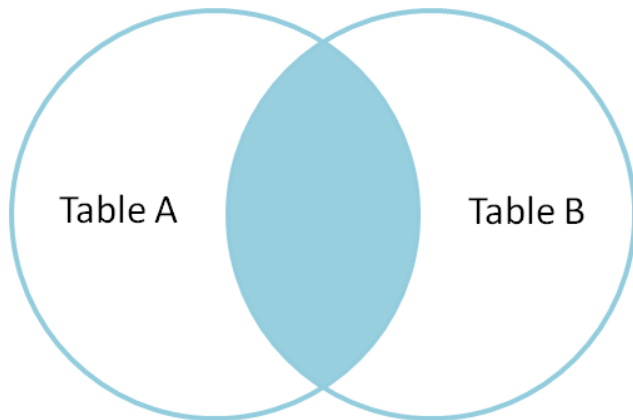
LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob



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

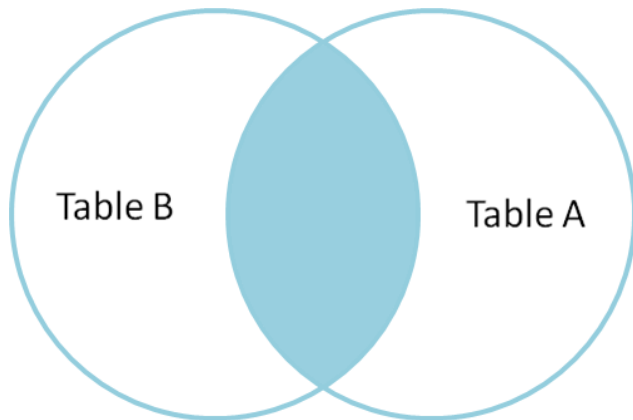


- SELECT * FROM **TableA**
INNER JOIN **TableB**
ON **TableA**.col_match = **TableB**.col_match





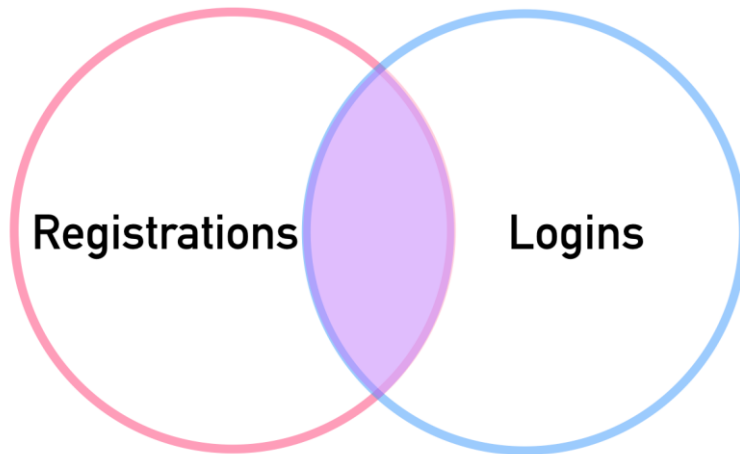
- SELECT * FROM **TableB**
INNER JOIN **TableA**
ON **TableA**.col_match = **TableB**.col_match





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

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

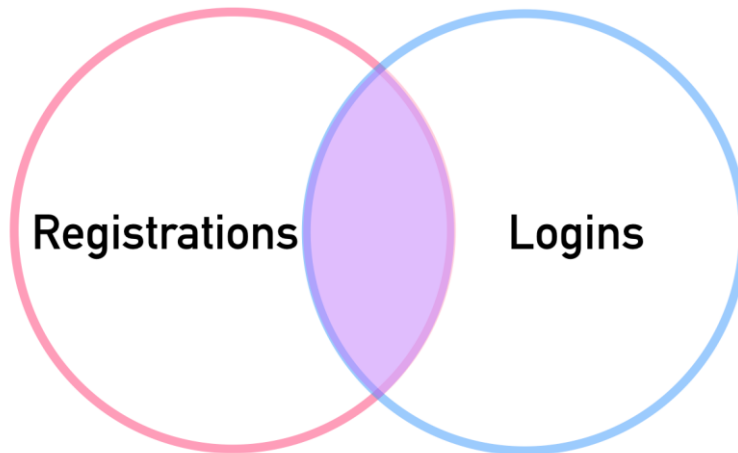


LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob



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

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

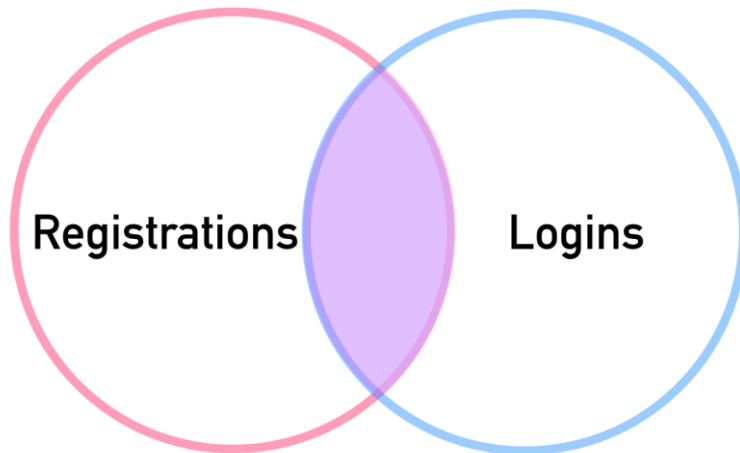


LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob



- `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



- 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



- `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



- `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



- 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



- 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!



- In these lectures we will explain:
 - FULL OUTER JOIN
 - Clarifying WHERE null
 - LEFT OUTER JOIN
 - Clarifying WHERE null
 - RIGHT OUTER JOIN
 - Clarifying WHERE null



FULL OUTER JOIN



- 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



- 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



- 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



- 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



- 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



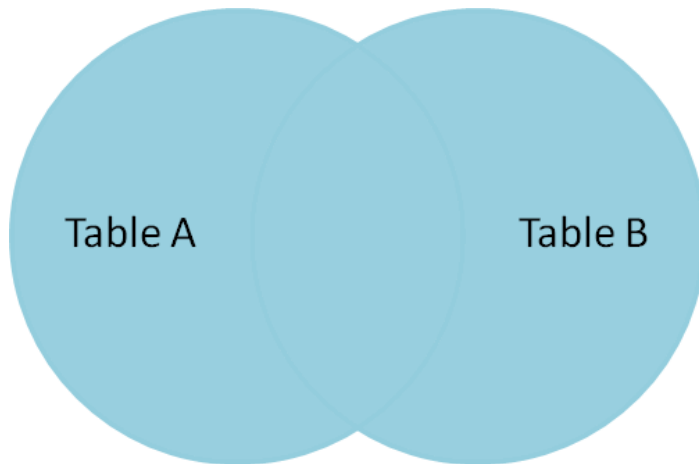
We will first take a look at the simplest,
which is a **FULL OUTER JOIN**

REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob

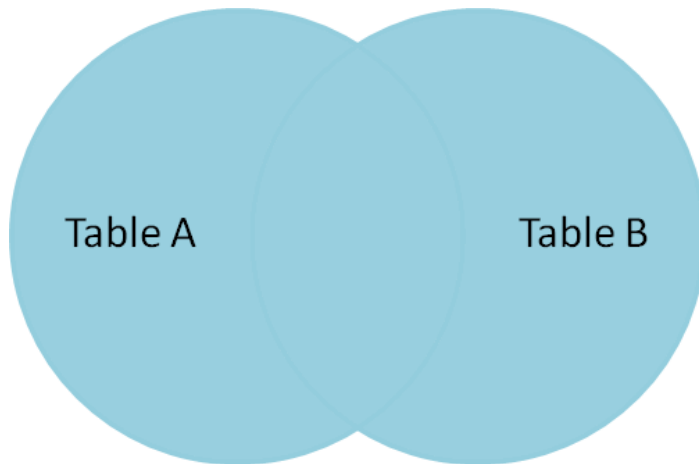


- `SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.col_match = TableB.col_match`





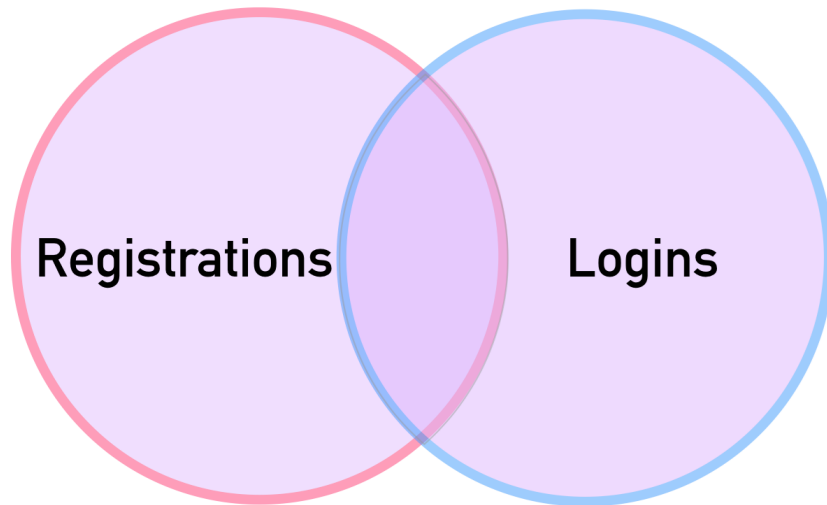
- SELECT * FROM **TableB**
FULL OUTER JOIN **TableA**
ON **TableA**.col_match = **TableB**.col_match





- `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



- `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



- 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



- 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



- 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



- 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

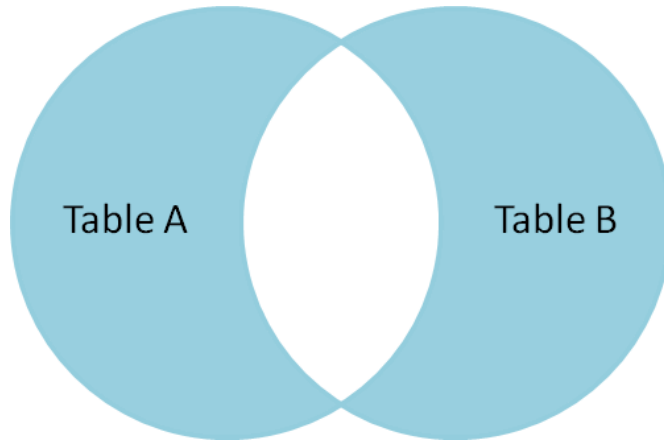


FULL OUTER JOIN with WHERE

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

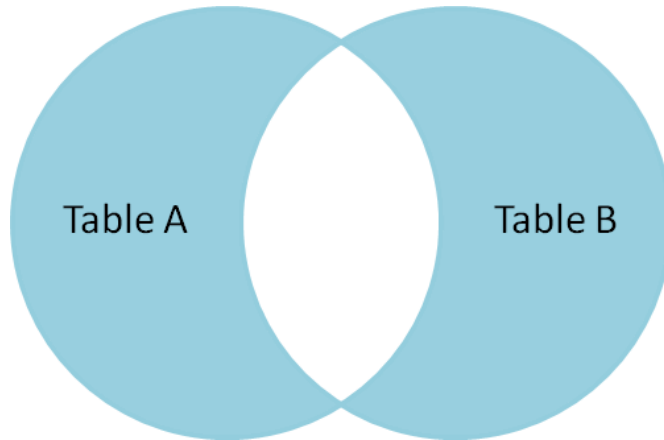


- `SELECT * FROM TableA
FULL OUTER JOIN TableB
ON TableA.col_match = TableB.col_match
WHERE TableA.id IS null OR
TableB.id IS null`



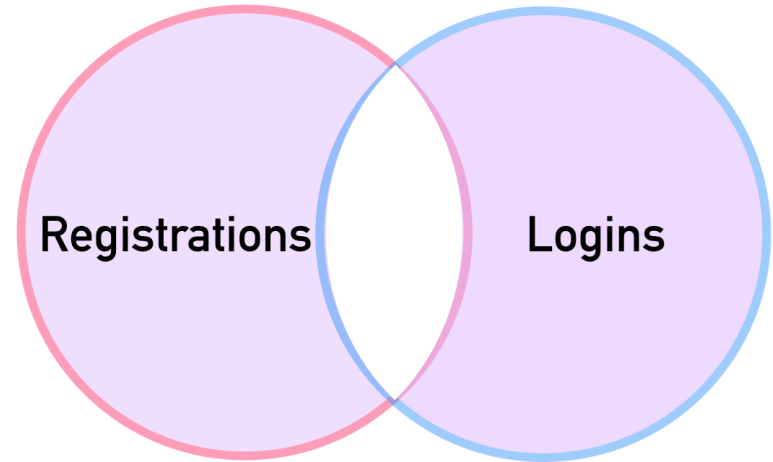


- `SELECT * FROM TableB
FULL OUTER JOIN TableA
ON TableA.col_match = TableB.col_match
WHERE TableA.id IS null OR
TableB.id IS null`





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





- 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



- 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



- Let's see a quick example in pgAdmin of FULL OUTER JOINS!



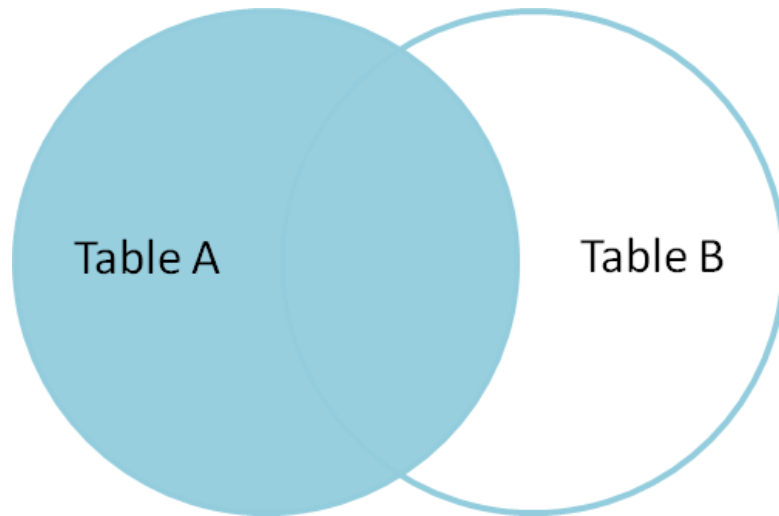
LEFT OUTER JOIN



- 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

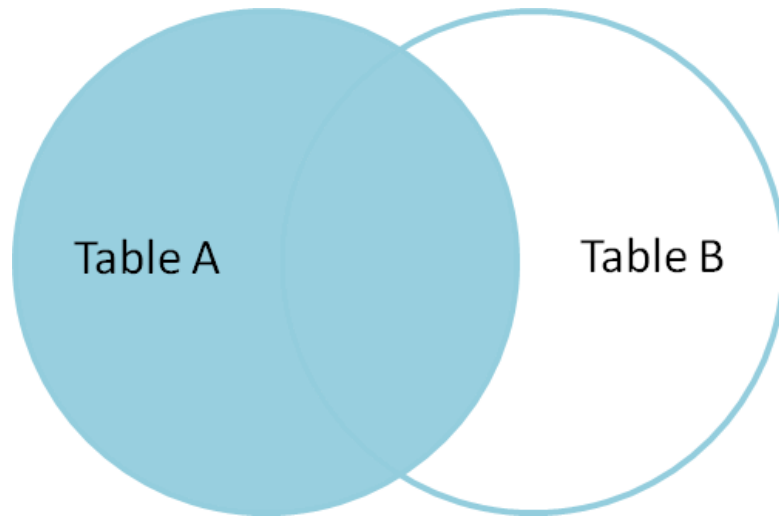


- `SELECT * FROM TableA
LEFT OUTER JOIN TableB
ON TableA.col_match = TableB.col_match`



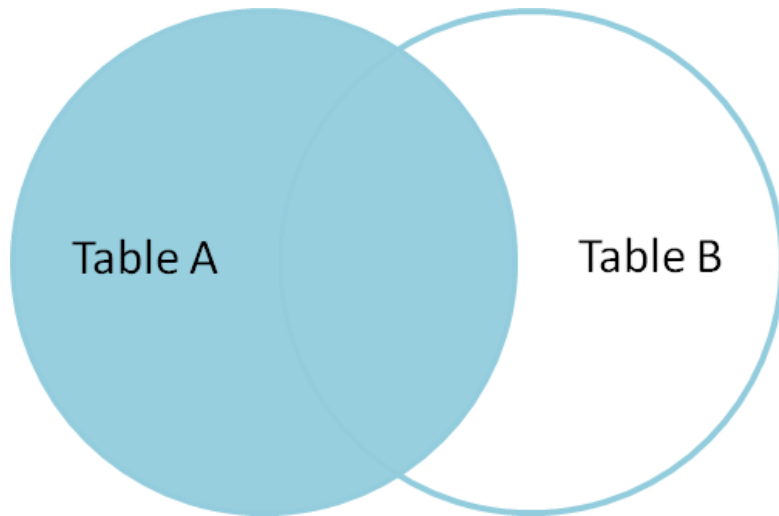


- `SELECT * FROM TableA
LEFT JOIN TableB
ON TableA.col_match = TableB.col_match`





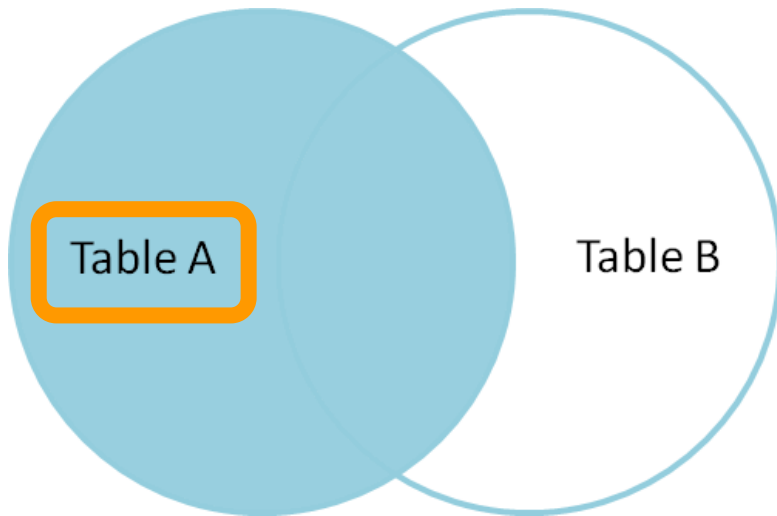
- SELECT * FROM **TableA**
LEFT OUTER JOIN **TableB**
ON **TableA**.col_match = **TableB**.col_match





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

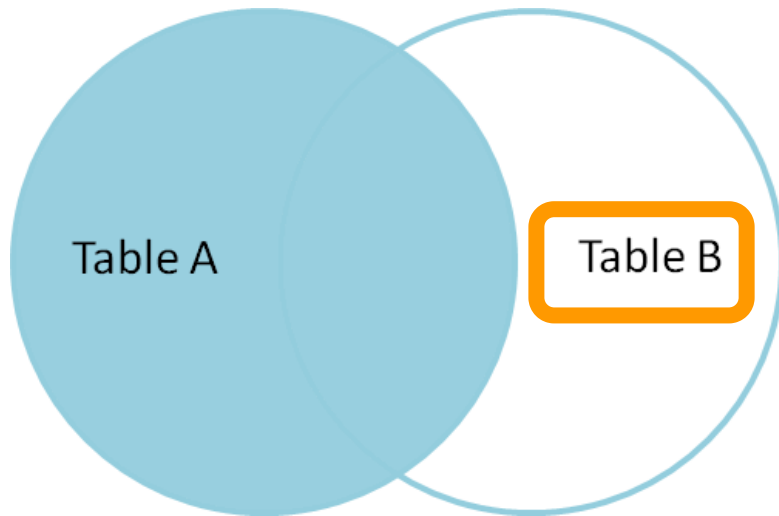
**ORDER
MATTERS FOR
LEFT OUTER JOIN!**





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

ORDER
MATTERS FOR
LEFT OUTER JOIN!





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

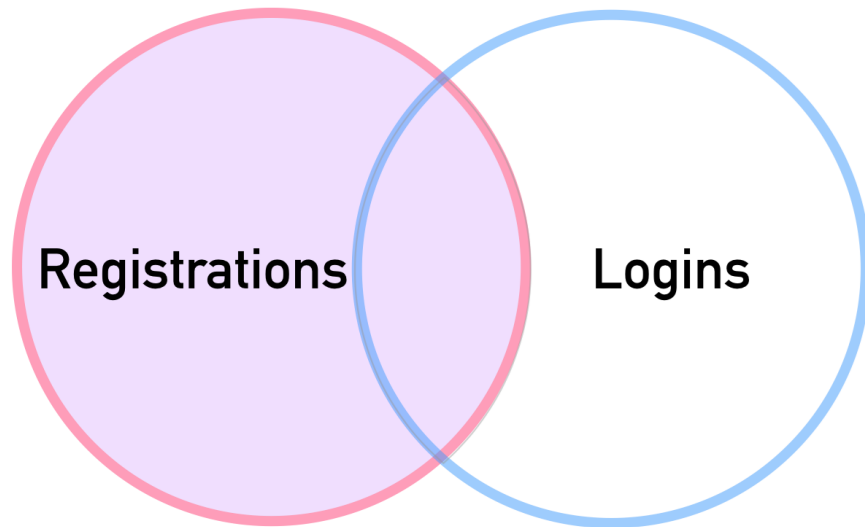
REGISTRATIONS	
reg_id	name
1	Andrew
2	Bob
3	Charlie
4	David

LOGINS	
log_id	name
1	Xavier
2	Andrew
3	Yolanda
4	Bob



- `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



- `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



- 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

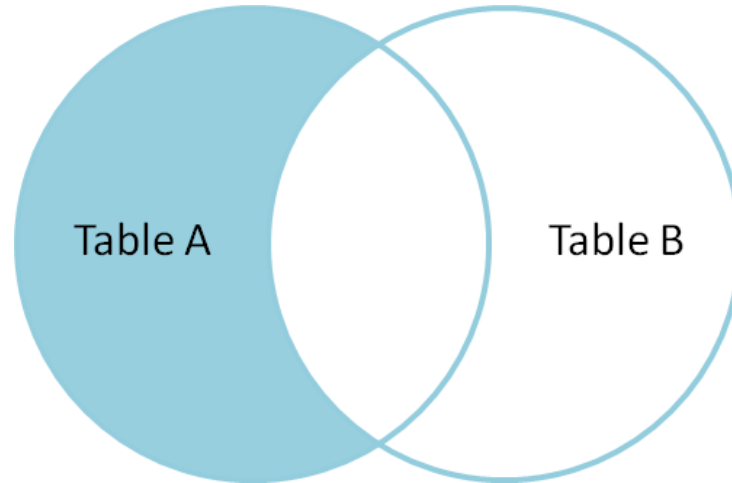


LEFT OUTER JOIN With WHERE

Get rows unique to left table

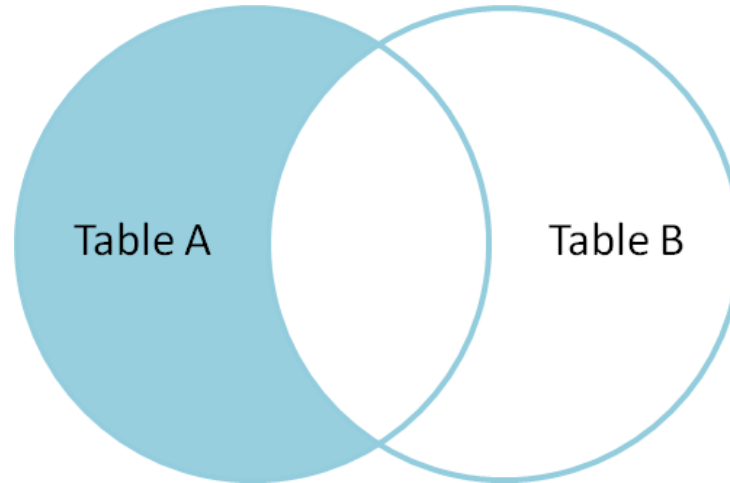


- What if we only wanted entries unique to Table A? Those rows found in Table A and **not** found in Table B.





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





- `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



```
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



- Let's see some examples of LEFT OUTER JOINS in pgAdmin!



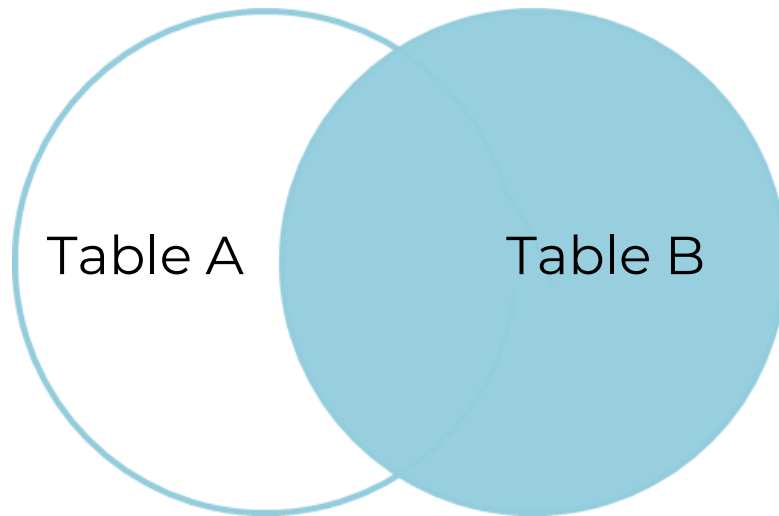
RIGHT JOINS



- 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.

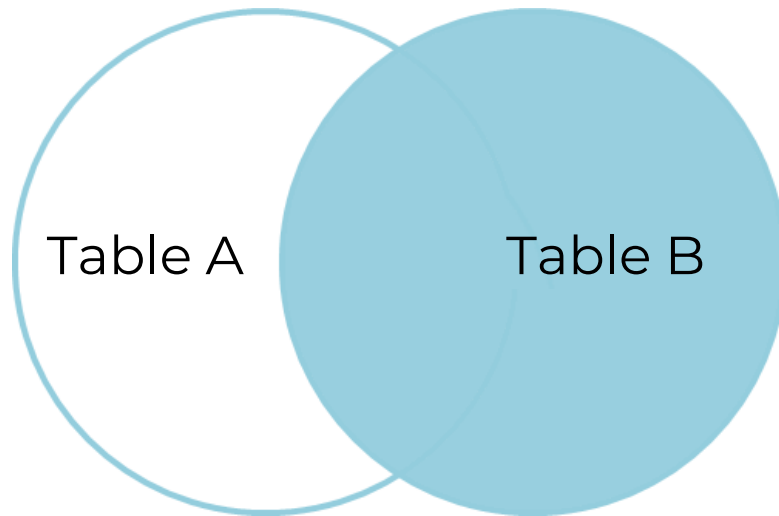


- SELECT * FROM **TableA**
RIGHT OUTER JOIN **TableB**
ON **TableA**.col_match = **TableB**.col_match



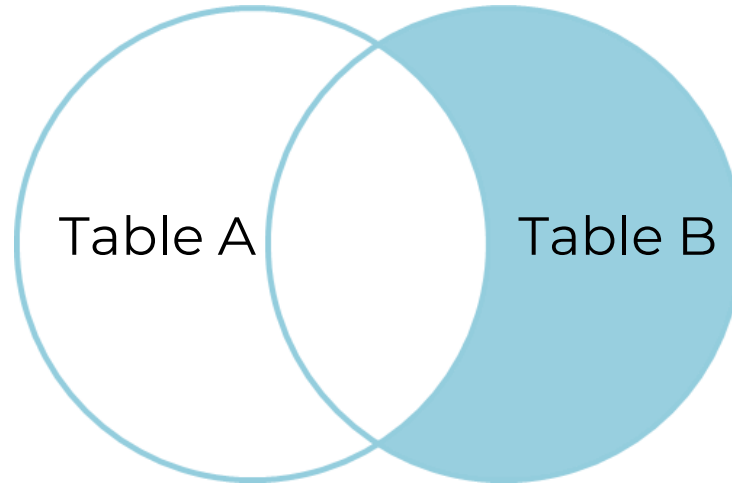


- `SELECT * FROM TableA
RIGHT JOIN TableB
ON TableA.col_match = TableB.col_match`





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





- 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.



JOIN

CHALLENGE TASKS



UNIONS



- 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.



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



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



- `SELECT * FROM Sales2021_Q1
UNION
SELECT * FROM Sales2021_Q2;`

name	amount
David	100
Claire	50
David	200
Claire	100



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

name	amount
David	100
David	200
Claire	50
Claire	100



JOIN Challenges



- 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?



- 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...



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



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



- 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.



- Expected Results

	<div><div>title</div><div>character varying (255)</div></div>	<div><div>first_name</div><div>character varying (45)</div></div>	<div><div>last_name</div><div>character varying (45)</div></div>
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



- 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



```
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'
```