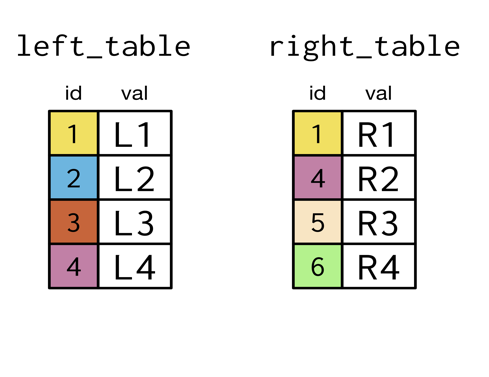
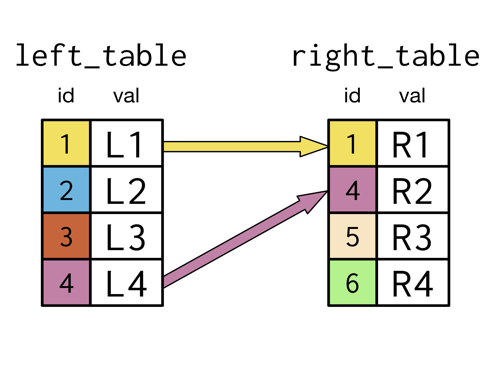
**Joining Data in PostgreSQL**

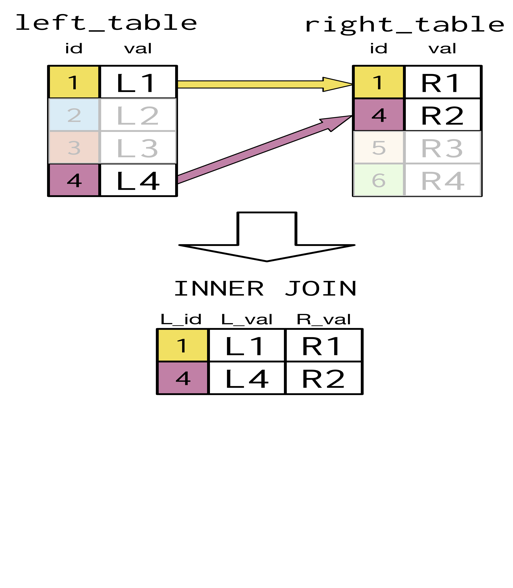
**Chapter 1:**

Introduction to Inner Join

* Mechanics of different joins

* In inner join, only includes the records in which the key is in both tables

 rthe Resultant table looks lik

INNER JOIN in SQL

**SELECT** p1.country, p1.continent,

prime\_minister, president

**FROM** prime\_ministers **AS** p1

**INNER** **JOIN** presidents **AS** p2

**ON** p1.country = p2.country;

+-----------+---------------+--------------------+-------------------------+

| country | continent | prime\_minister | president |

|-----------+---------------+--------------------+-------------------------|

| Egypt | Africa | Sherif Ismail | Abdel Fattah el-Sisi |

| Portugal | Europe | Antonio Costa | Marcelo Rebelo de Sousa |

| Vietnam | Asia | Nguyen Xuan Phuc | Tran Dai Quang |

| Haiti | North America | Jack Guy Lafontant | Jovenel Moise |

+-----------+---------------+--------------------+-------------------------+

Exercise 01

# Inner join

PostgreSQL was mentioned in the slides but you'll find that these joins and the material here applies to different forms of SQL as well.

Throughout this course, you'll be working with the countriesdatabase containing information about the most populous world cities as well as country-level economic data, population data, and geographic data. This countries database also contains information on languages spoken in each country.

You can see the different tables in this database by clicking on the tabs on the bottom right below **query.sql**. Click through them to get a sense for the types of data that each table contains before you continue with the course! Take note of the fields that appear to be shared across the tables.

Recall from the video the basic syntax for an INNER JOIN, here including all columns in **both** tables:

Recall from the video the basic syntax for an INNER JOIN, here including all columns in **both** tables:

SELECT \*

FROM left\_table

INNER JOIN right\_table

ON left\_table.id = right\_table.id;

You'll start off with a SELECT statement and then build up to an inner join with the cities and countries tables. Let's get to it!

Chapter 1.1

Using Key word

When the column names are with same name, then ‘USING’ clause can be used instead on ‘ON’ clause.

Chapter 1.2

Self-ish Join, just in case

Inner Join – join with itself (Self-Join)

* Compare the field with other field in the same table

CASE When and THEN

* CASE
* - suppose to group an year
* CASE - if-when-then-else
* Syntax :

SELECT name, continent, indep\_year,

CASE WHEN \_ \_ \_< \_ \_ \_ THEN ‘before 1990’

WHEN indep\_year <= 1930 THEN ‘---‘

ELSE ‘---‘ END

AS indep\_year\_group

FROM states

ORDER BY indep\_year\_group

SELECT name, continent, indep\_year,

CASE WHEN indep\_year < 1900 THEN ‘before 1990’

WHEN indep\_year <= 1930 THEN ‘in between 1900 and 1930‘

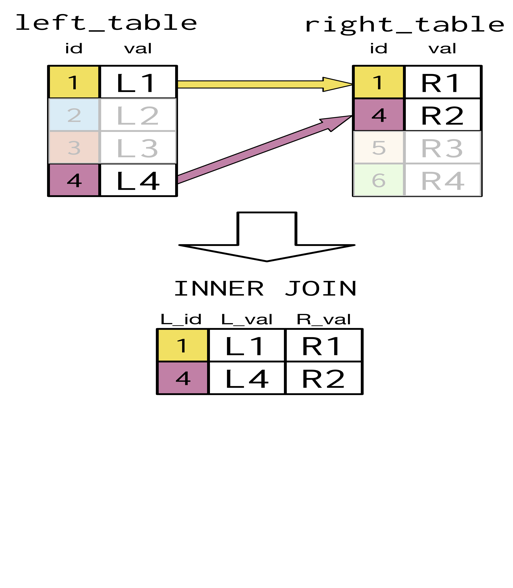
ELSE ‘after 1930‘ END

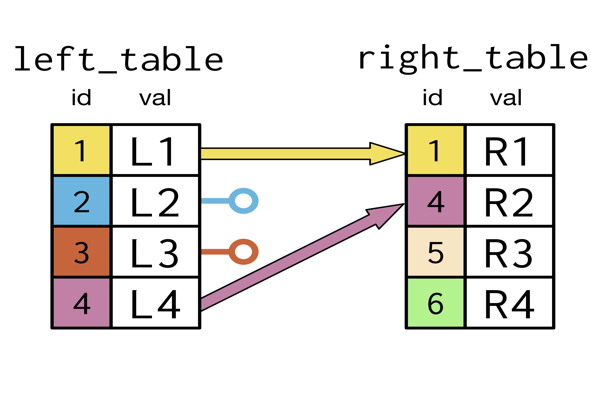
AS indep\_year\_group

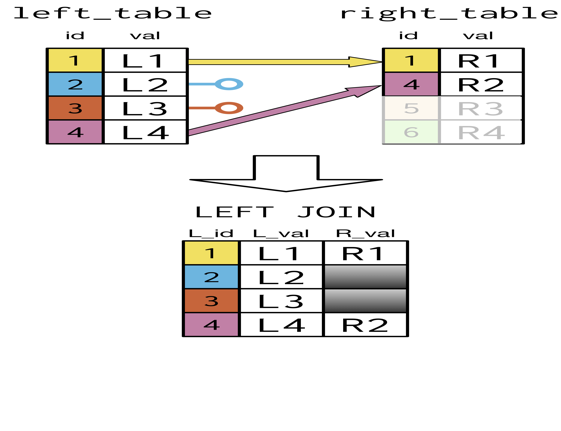
FROM states

ORDER BY indep\_year\_group:

* LEFT JOIN and RIGHT JOIN and JOIN
* LEFT JOIN

Inner Join 

Left Join 



Syntax :

select **p1.country, prime\_minister, president**

From

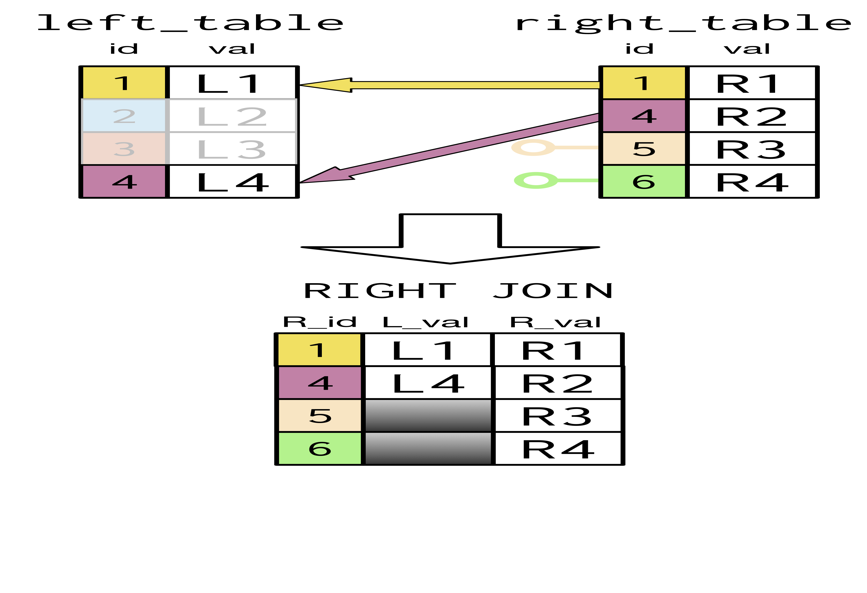
**prime\_minister** as p1

left join

**president** as p2

on **p1.country = p2.country**

**RIGHT JOIN**



SELECT

**right\_table.id AS R\_id,**

**left\_table.val AS L\_val,**

**right\_table.val AS R\_val**

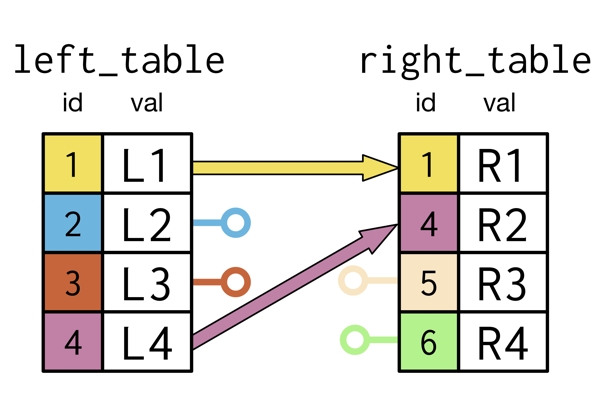
FROM l**eft\_table**

RIGHT JOIN **right\_table**

ON **left\_table.id = right\_table.id;**

**Chapter 2.2**

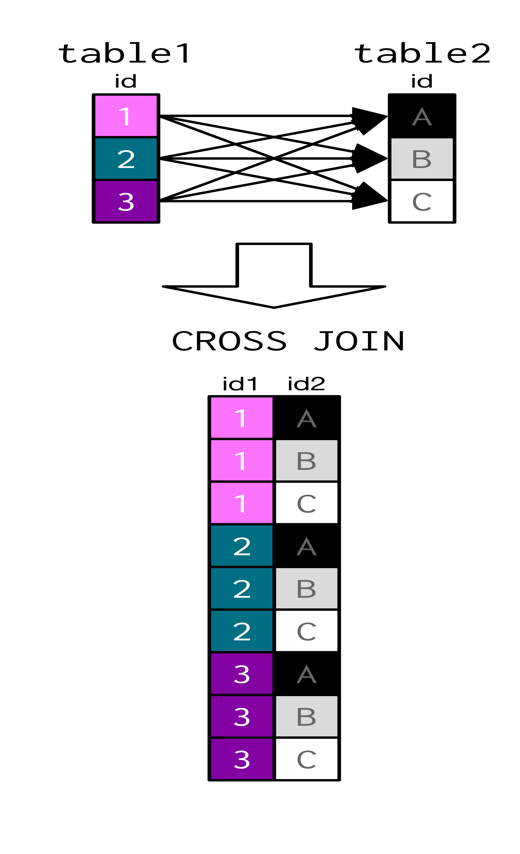
**Full Join : combination of Left join and right join**



**Select** left\_table.id as L\_id, right\_table.id as R\_id, left\_table.val as L\_val, right\_table.val as R\_val **FROM** left\_table FULL JOIN right\_table USING(id).

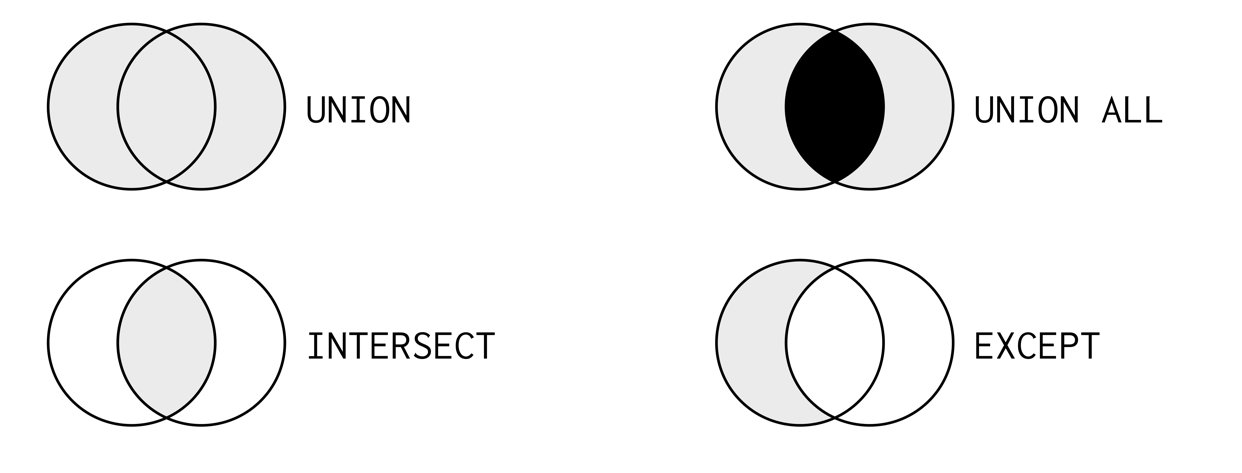
Select p1.country as pm\_co, p2.country as pres\_co, prime\_minister, president from prime\_minister as P1 FULL JOIN presidents AS p2 ON p1.country = p2.country

Crossing the Rubicon (Cross Joins)

* 
* Cross joins return all the valid keys of table 1 with keys of table 2
* SELECT prime\_minister, president from prime\_minister AS p1 CROSS JOIIN presidents AS p2 where p1.continent IN (‘North America’, ‘Oceania’)

State of UNION

Venn Diagrams



* UNION is attaching two files with records
  + All the fields should be same
  + Doesn’t include duplciateas
* UNION ALL
  + All the fields should be same
  + Does include duplicates

Select prime\_minister AS leader, country FROM prime\_minister UNION SELECT monarch, country from monarch ORDER by country

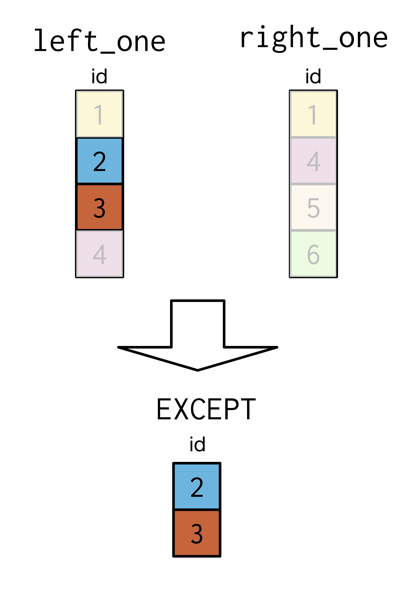
INTERSECT

* Only attaches the records common on both tables
  + SELECT id from left\_table INTERSECT SELECT id FROM right\_one
* Intersect with two columns
  + Select country, prime\_minister as leader from prime\_ministers intersect select country, president from presidents

Exceptional:

EXCEPT - allows to include the records in one table and not the others.

e.g SELECT monarch, country FROM monarchs EXCEPT SELECT prime\_minister, country FROM prime\_ministers;



* only the records in the left table and records in right table are not included.

**Semi-joins and Anti-Joins:**

**Six joins:** Inner, outer, self, left, right, cross

**Semi-join:**

* + Right-Join on Left-Join

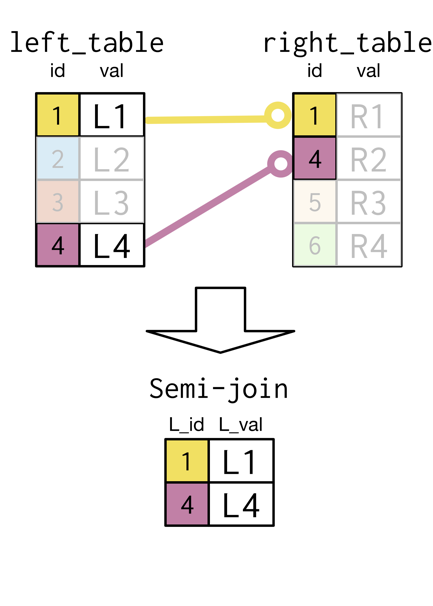
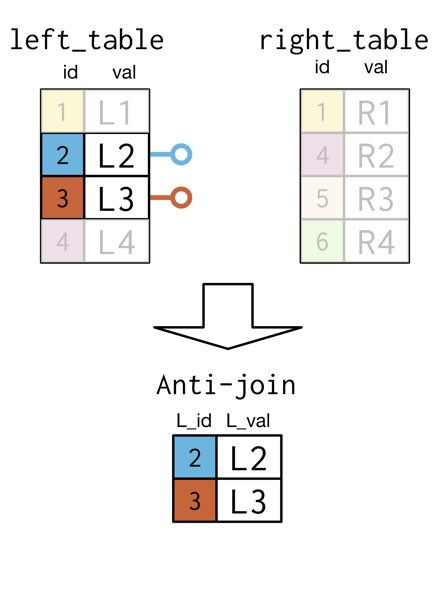
E.g. Select name from states where indep\_year < 1800

Sub-select

Select president, country, continent from presidents where country in (select name from states where indep\_year < 1800)

**Anti Join:**

* + Select president, country, continent from presidents where continent like ‘America’ and country in (select name from states where indep\_year < 1800)

pre { padding: 8px; line-height:

**Chapter 4**

**Sub-queries inside where and select clauses:**

**Average-Fertility Rate:**

* **Use fert\_rate as subquery**

**Sub-query :**

Select DISTINCT continent, (select count(\*) from states where prime\_ministers.continent = states.continent) AS countries\_num FROM prime\_ministers;

**Sub query inside a from clause:**

Select continent, MAX(women\_parli\_perc) AS max\_perc FROM states GROUP BY continent, ORDER BY continent.