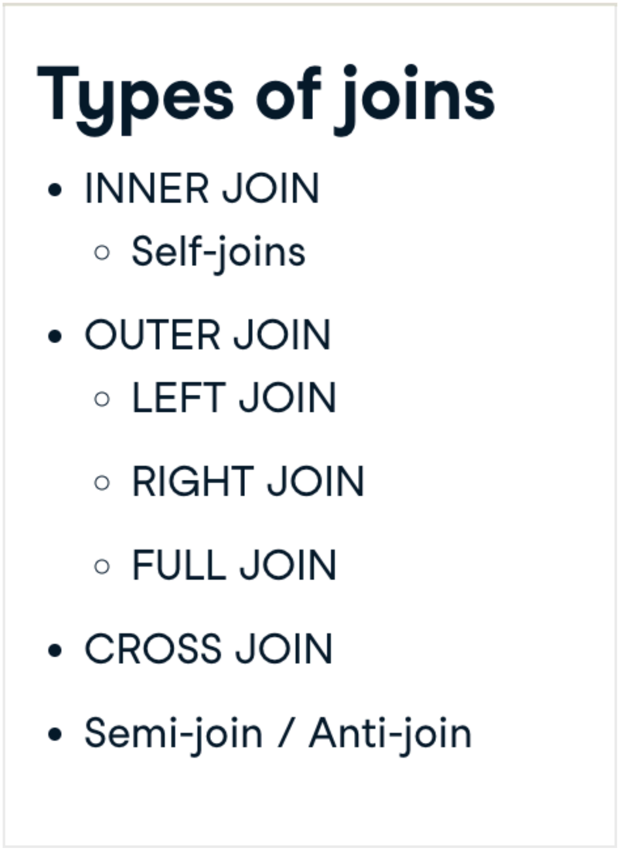
JOINS IN SQL



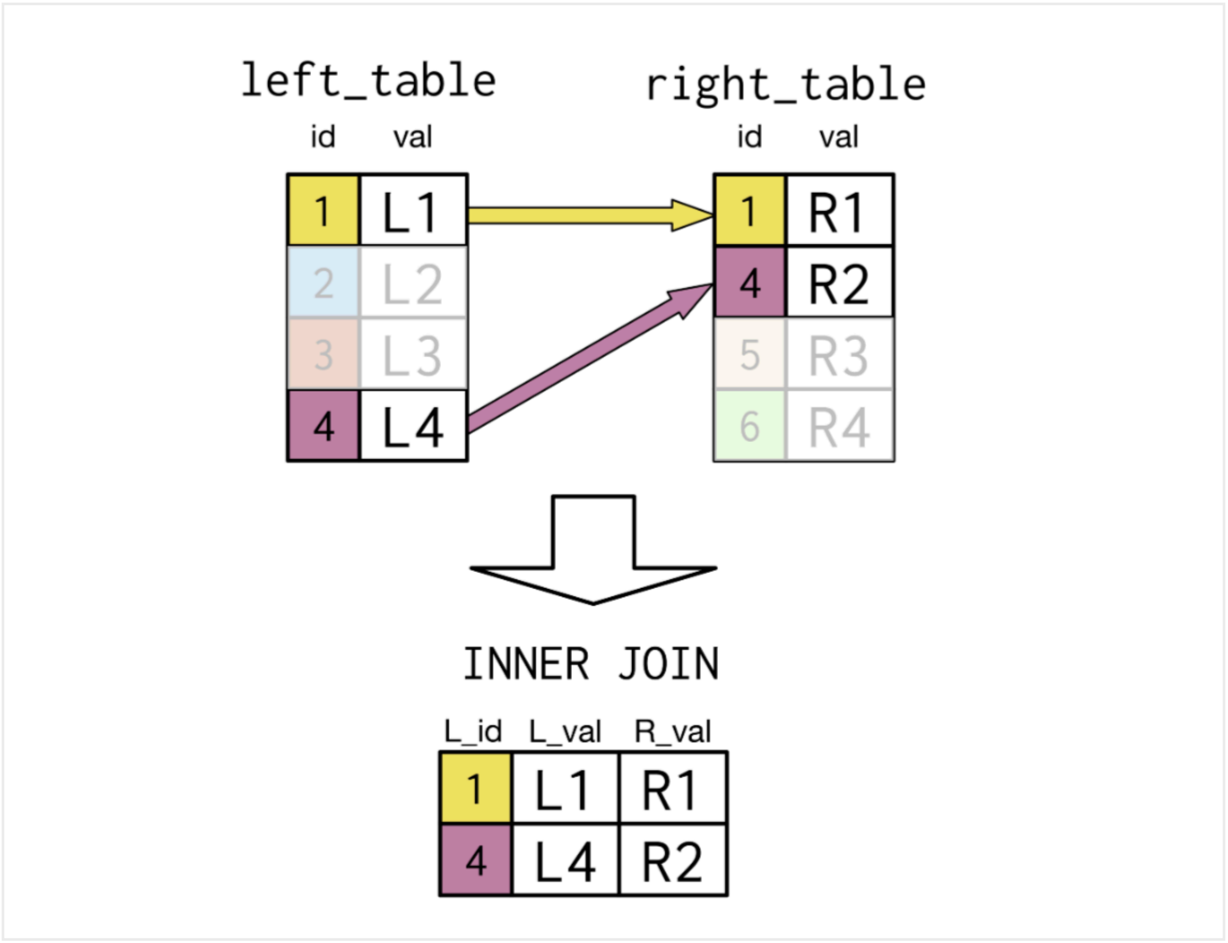
Types of joins

- INNER JOIN
  - Self-joins
- OUTER JOIN
  - LEFT JOIN
  - RIGHT JOIN
  - FULL JOIN
- CROSS JOIN
- Semi-join / Anti-join

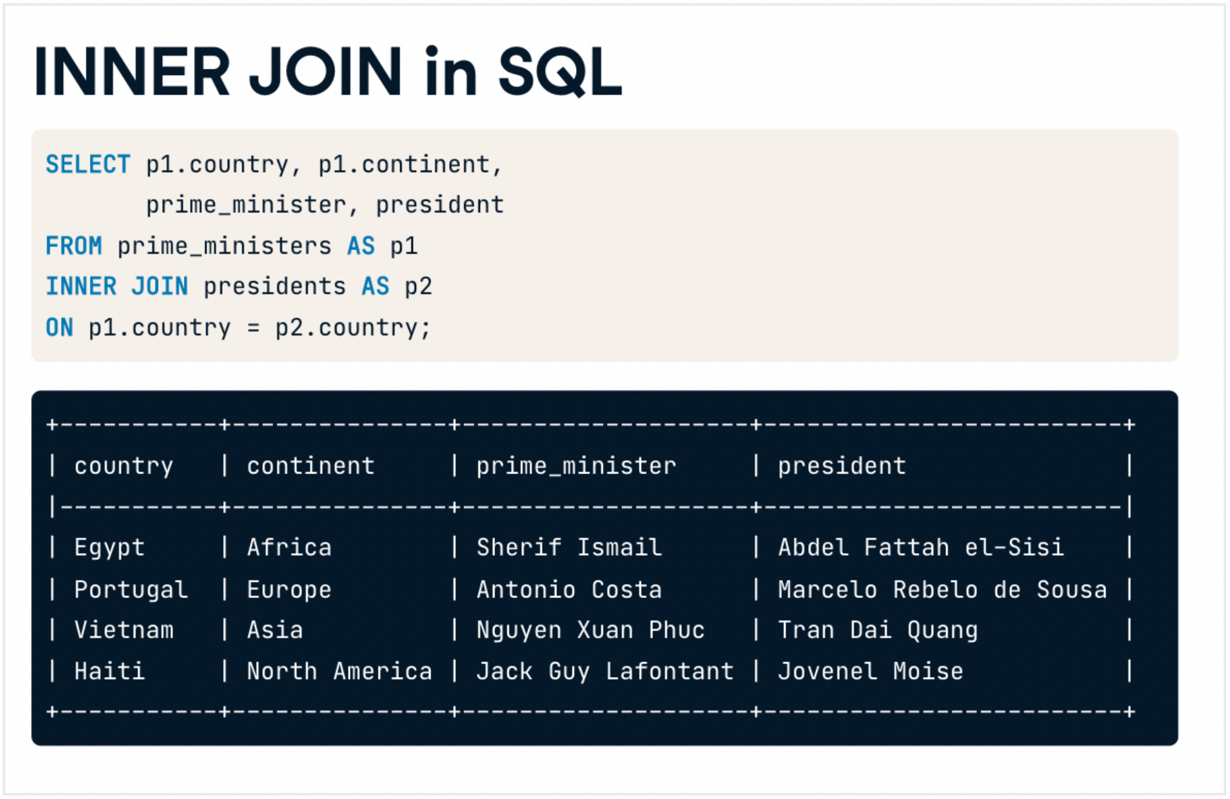
INNER JOIN

Key Word - Inner join table_name
on col1 = col2;

- **Inner join selects only the rows that select the matching join column rows between the left and right table**



– Returns only those rows with the matching key fields from the left and right tables.



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          |
+-----------+---------------+-------------------+-------------------------+
```

Similar to
prime_ministers.merge(president, on = 'country', how = 'inner')

```
select p1.country, p2.continent, p1.prime_minister, p2.president
from prime_ministers as p1
inner join president as p2
on p1.country = p2.country;
```

```
-- Select fields
SELECT c.code, name, region, e.year, fertility_rate, unemployment_rate
  -- From countries (alias as c)
  FROM countries AS c
  -- Join to populations (as p)
  INNER JOIN populations AS p
    -- Match on country code
    ON c.code = p.country_code
  -- Join to economies (as e)
  INNER JOIN economies AS e
    -- Match on country code and year
    on c.code = e.code and p.year = e.year;
```

USING vs ON when joining

- **ON keyword will be used when we having different column names in the left and right table**

- **USING keyword will be used when we have the same column name in the left and right table**

```
SELECT left_table.id AS L_id,
       left_table.val AS L_val,
       right_table.val AS R_val
FROM left_table
INNER JOIN right_table
USING (id);
```

Note: When making use of the using key word the column name should be encased within a parenthesis

Self-join

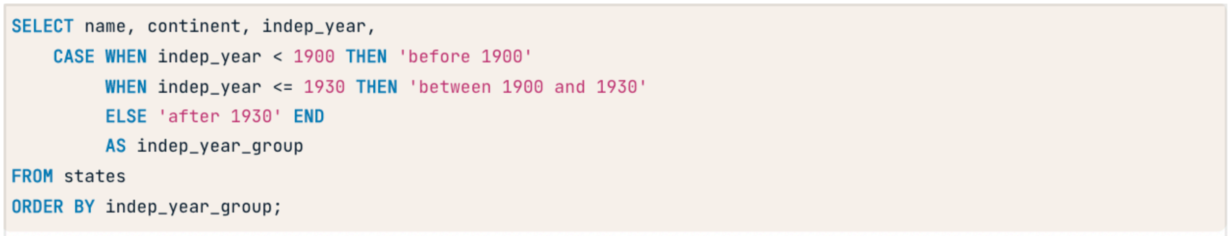
```
SELECT p1.country AS country1, p2.country AS country2, p1.continent
FROM prime_ministers AS p1
INNER JOIN prime_ministers AS p2
ON p1.continent = p2.continent AND p1.country <> p2.country
LIMIT 13;
```

```
+------------+------------+-------------+
| country1   | country2   | continent   |
|------------+------------+-------------|
| Portugal   | Spain      | Europe      |
| Portugal   | Norway     | Europe      |
| Vietnam    | Oman       | Asia        |
| Vietnam    | Brunei     | Asia        |
| Vietnam    | India      | Asia        |
| India      | Oman       | Asia        |
| India      | Brunei     | Asia        |
| India      | Vietnam    | Asia        |
| Norway     | Spain      | Europe      |
| Norway     | Portugal   | Europe      |
| Brunei     | Oman       | Asia        |
| Brunei     | India      | Asia        |
| Brunei     | Vietnam    | Asia        |
+------------+------------+-------------+
```

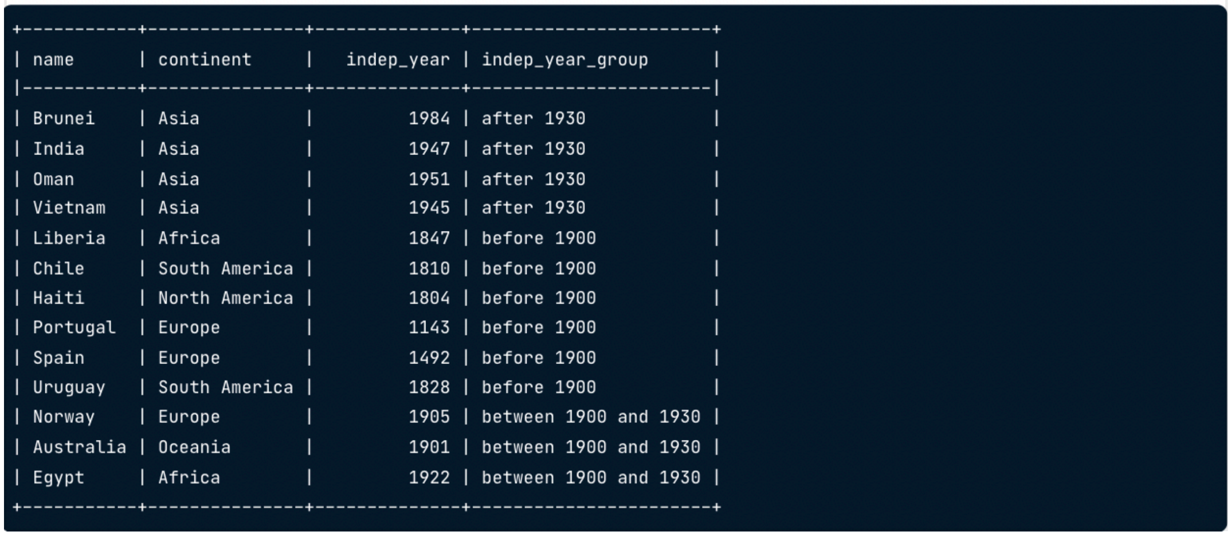
CASE
CASE, WHEN , THEN, ELSE, END

- Case is a simplified version of placing many if then else statements inside of SQL
- We use case when we have to group things in a manner that is not specified by the Group by clause,
- Suppose we want to group years before 1990 as one group, b/n 1990 and

2000 as another and beyond 2000 as another, the case keyword will allow us to perform this.



```
SELECT name, continent, indep_year,
    CASE WHEN indep_year < 1900 THEN 'before 1900'
         WHEN indep_year <= 1930 THEN 'between 1900 and 1930'
         ELSE 'after 1930' END
         AS indep_year_group
FROM states
ORDER BY indep_year_group;
```



```
+-----------+--------------+-------------+----------------------+
| name      | continent    |  indep_year | indep_year_group     |
|-----------+--------------+-------------+----------------------|
| Brunei    | Asia         |        1984 | after 1930           |
| India     | Asia         |        1947 | after 1930           |
| Oman      | Asia         |        1951 | after 1930           |
| Vietnam   | Asia         |        1945 | after 1930           |
| Liberia   | Africa       |        1847 | before 1900          |
| Chile     | South America |        1810 | before 1900          |
| Haiti     | North America |        1804 | before 1900          |
| Portugal  | Europe       |        1143 | before 1900          |
| Spain     | Europe       |        1492 | before 1900          |
| Uruguay   | South America |        1828 | before 1900          |
| Norway    | Europe       |        1905 | between 1900 and 1930 |
| Australia | Oceania      |        1901 | between 1900 and 1930 |
| Egypt     | Africa       |        1922 | between 1900 and 1930 |
+-----------+--------------+-------------+----------------------+
```

SELECT name, continent, indep_year,
    CASE WHEN indep_year < 1900 **THEN** 'before_1900'
        WHEN indep_year <= 1930 **THEN** 'between 1900 and 1930'
        ELSE 'after 1930' **END**
        AS indep_year_group
FROM states
ORDER BY indep_year_group;

- From the above Postresql query we can note the following things,
  - The case statement is used to create a new column where we perform multiple conditional statements, we can use these values to filter rows further in the where clause
  - the case column starts with the keyword CASE and the end is marked by the END keyword.
  - To start a conditional statement we will specify the WHEN keyword then place a boolean condition and then use the THEN keyword to specify the value that should be placed in the column when the boolean condition is met in the THEN clause.
  - WHEN bool_condition THEN 'val_to_use' END
  - At the end we can use aliasing to specify the new column name.

```
SELECT name, continent, code, surface_area,
    -- First case
    CASE WHEN surface_area > 2000000 THEN 'large'
        -- Second case
        WHEN surface_area > 350000 AND surface_area < 2000000 THEN 'medium'
        -- Else clause + end
        ELSE 'small' END
        -- Alias name
        AS geosize_group
-- From table
FROM countries;
```

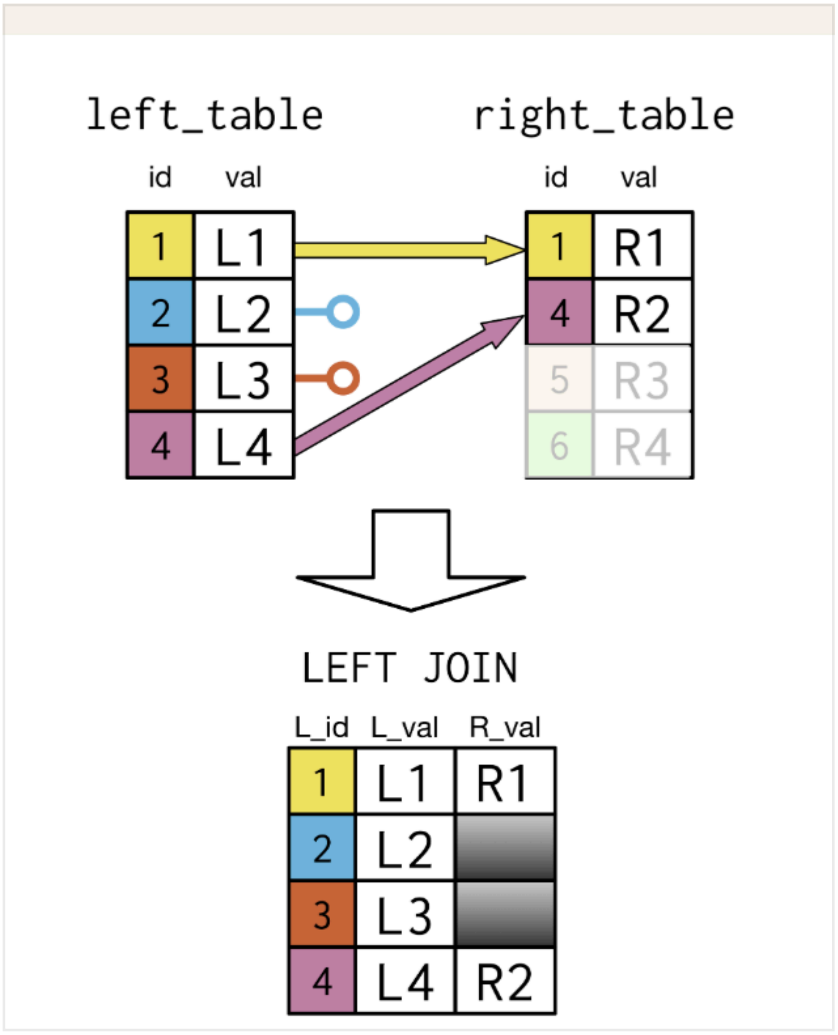
INTO

- The into keyword is used to store the result of a query into a new table, we can then access the new table using another query.
- In the below code snipped we can notice that before accessing the pop_plus table there is a semicolon which marks that 1st query is ended and the second query is beginning.

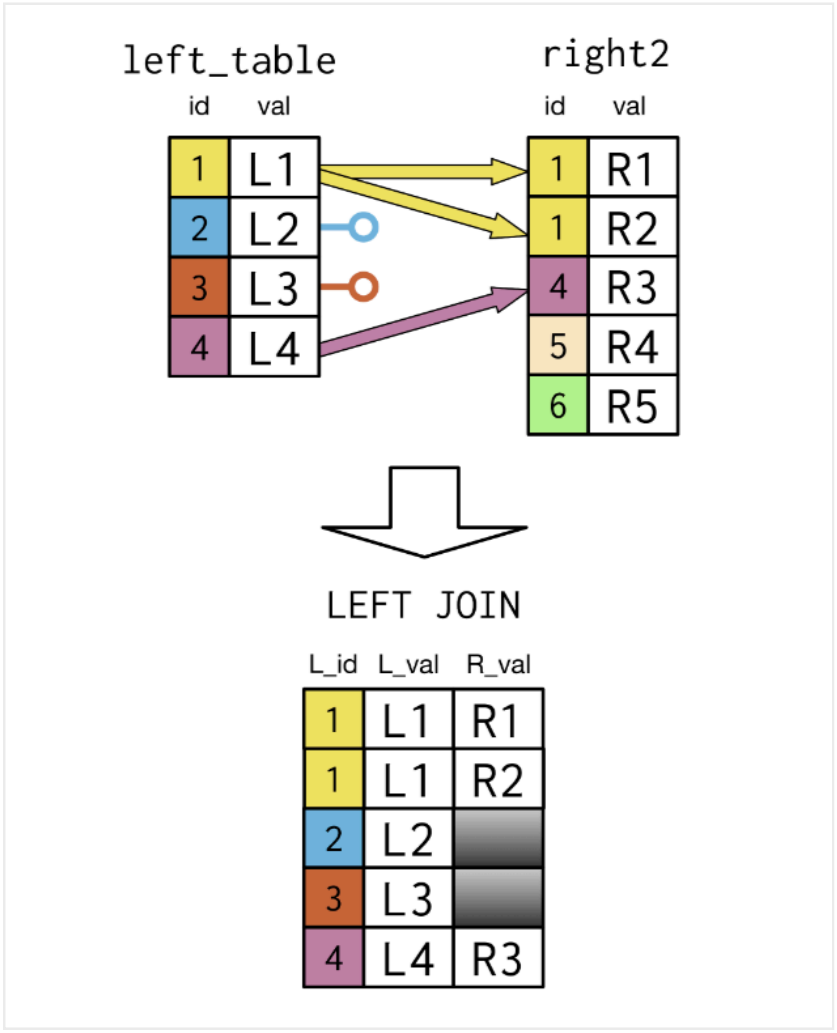
```
SELECT country_code, size,
    CASE WHEN size > 50000000 THEN 'large'
        WHEN size > 1000000 THEN 'medium'
        ELSE 'small' END
        AS popsize_group
-- Into table
Into pop_plus
FROM populations
WHERE year = 2015;

-- Select all columns of pop_plus
Select *
FROM pop_plus;
```

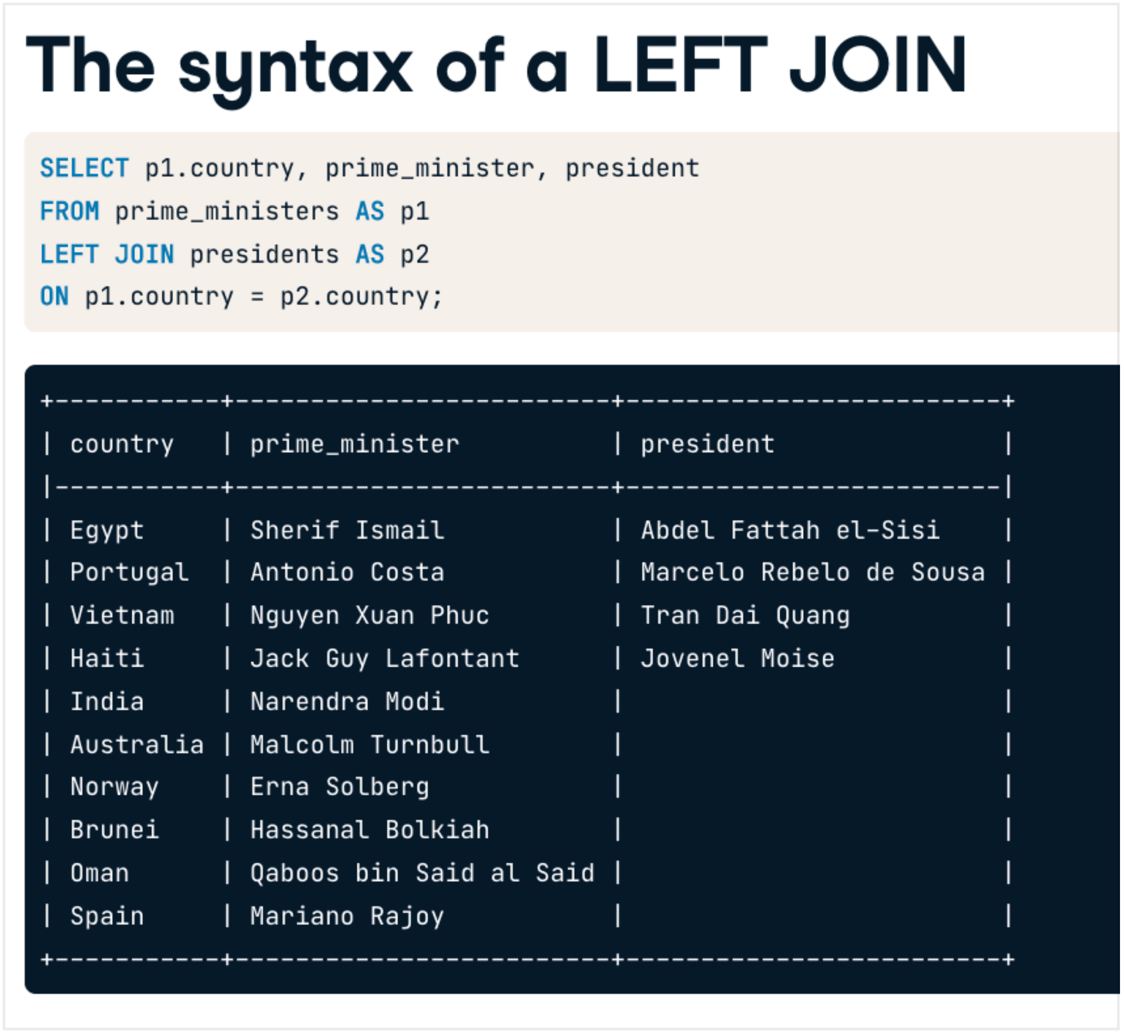
LEFT JOIN



Left join keeps all the records from the left table and marks the values as missing (in the right table column)that are not present in the right table.
left.merge(right, on = ‘id’, how = ‘left’)



One to many relationships. We get many relationship rows when we have more than one matches in the right row.



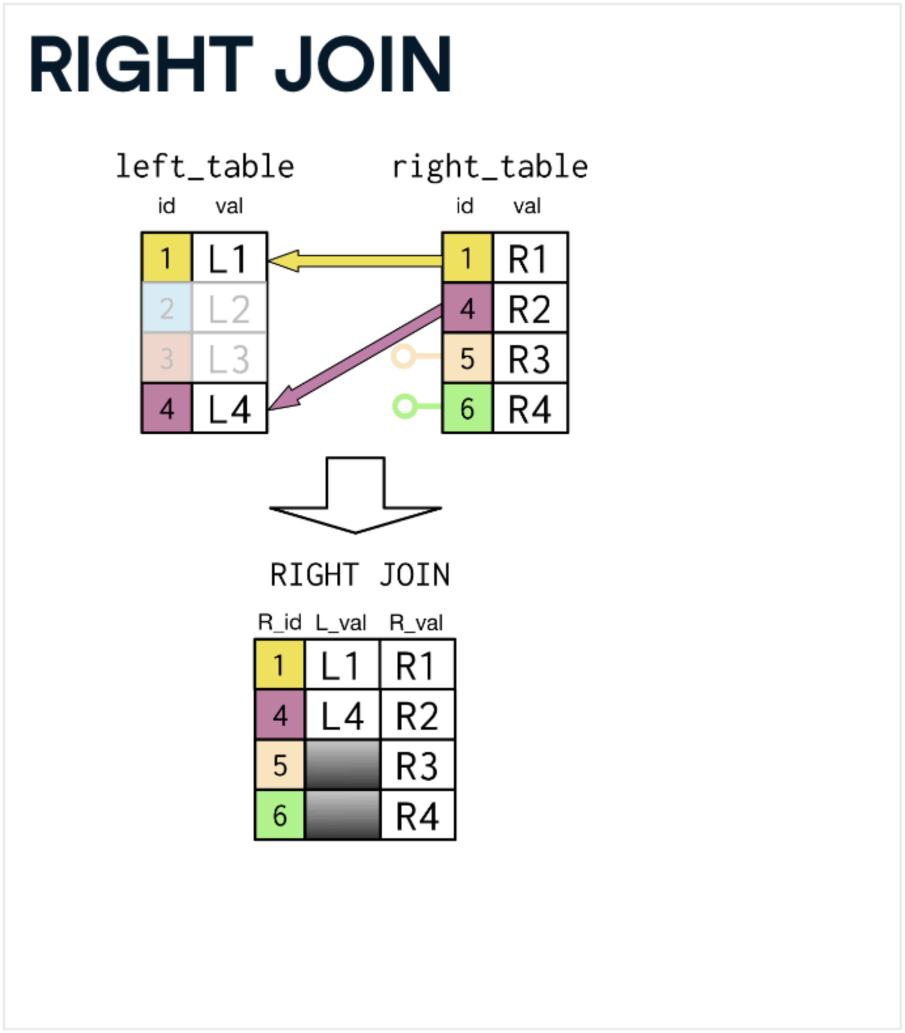
The syntax of a LEFT JOIN

```sql
SELECT p1.country, prime_minister, president
FROM prime_ministers AS p1
LEFT JOIN presidents AS p2
ON p1.country = p2.country;
```

```
+-----------+------------------------+------------------------+
| country   | prime_minister         | president              |
|-----------+------------------------+------------------------|
| Egypt     | Sherif Ismail          | Abdel Fattah el-Sisi   |
| Portugal  | Antonio Costa          | Marcelo Rebelo de Sousa |
| Vietnam   | Nguyen Xuan Phuc       | Tran Dai Quang         |
| Haiti     | Jack Guy Lafontant     | Jovenel Moise          |
| India     | Narendra Modi          |                        |
| Australia | Malcolm Turnbull       |                        |
| Norway    | Erna Solberg           |                        |
| Brunei    | Hassanal Bolkiah       |                        |
| Oman      | Qaboos bin Said al Said |                       |
| Spain     | Mariano Rajoy          |                        |
+-----------+------------------------+------------------------+
```

Note: From the above query we can observe that the left table is place in the from clause and the right table is placed after the left join keyword, in the below right join query we can also observe the same.

RIGHT JOIN

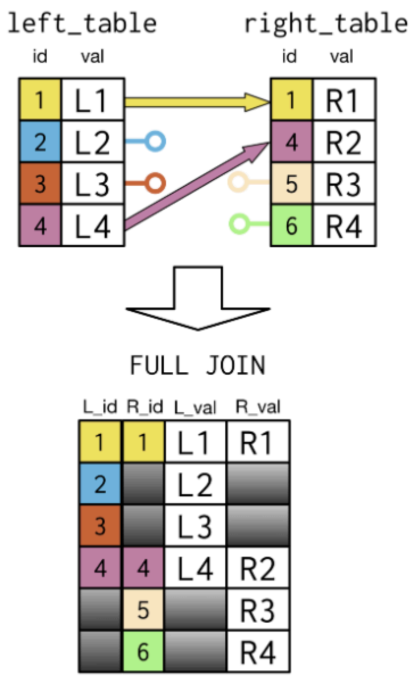


- All entries in the right table are kept, we have a missing value in the merged column where we do not have a match in the left column.

```
SELECT right_table.id AS R_id,
       left_table.val AS L_val,
       right_table.val AS R_val
FROM left_table
RIGHT JOIN right_table
ON left_table.id = right_table.id;
```

FULL JOIN

FULL JOIN diagram

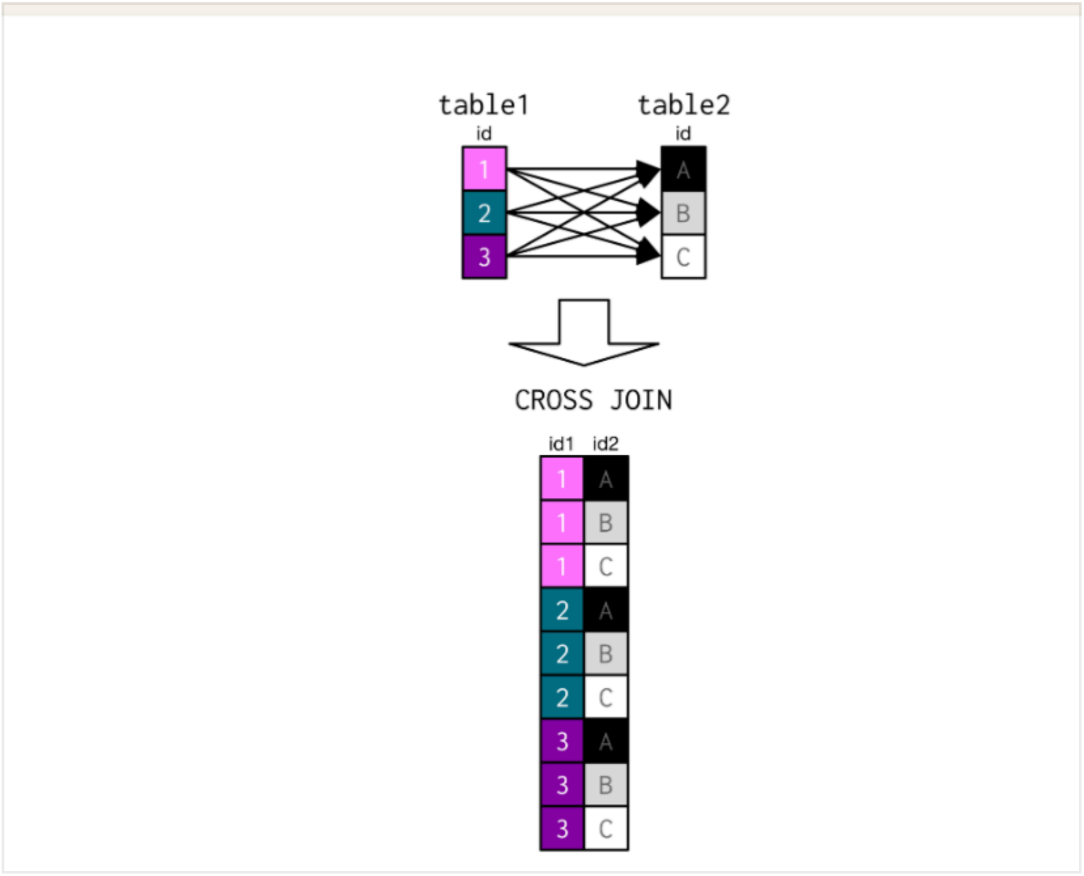


```
SELECT left_table.id AS L_id,
       right_table.id AS R_id,
       left_table.val AS L_val,
       right_table.val AS R_val
FROM left_table
FULL JOIN right_table
USING (id);
```

In full join, we retain all the records from both the left and right tables and we fill in values for the merged columns only in those rows where we have a common value.

CROSS JOIN

- Create all possible combinations

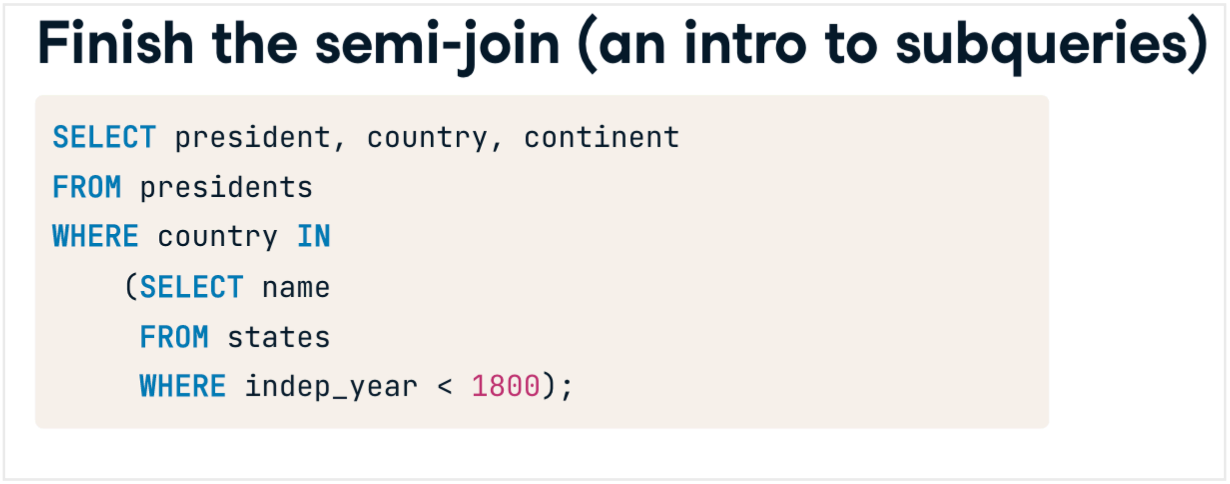


Pairing prime ministers with presidents

```sql
SELECT prime_minister, president
FROM prime_ministers AS p1
CROSS JOIN presidents AS p2
WHERE p1.continent IN ('North America', 'Oceania');
```

```
+--------------------+------------------------+
| prime_minister     | president              |
|--------------------+------------------------|
| Jack Guy Lafontant | Abdel Fattah el-Sisi   |
| Malcolm Turnbull   | Abdel Fattah el-Sisi   |
| Jack Guy Lafontant | Marcelo Rebelo de Sousa |
| Malcolm Turnbull   | Marcelo Rebelo de Sousa |
| Jack Guy Lafontant | Jovenel Moise          |
| Malcolm Turnbull   | Jovenel Moise          |
| Jack Guy Lafontant | Jose Mujica            |
| Malcolm Turnbull   | Jose Mujica            |
| Jack Guy Lafontant | Ellen Johnson Sirleaf  |
| Malcolm Turnbull   | Ellen Johnson Sirleaf  |
| Jack Guy Lafontant | Michelle Bachelet      |
| Malcolm Turnbull   | Michelle Bachelet      |
| Jack Guy Lafontant | Tran Dai Quang         |
| Malcolm Turnbull   | Tran Dai Quang         |
+--------------------+------------------------+
```

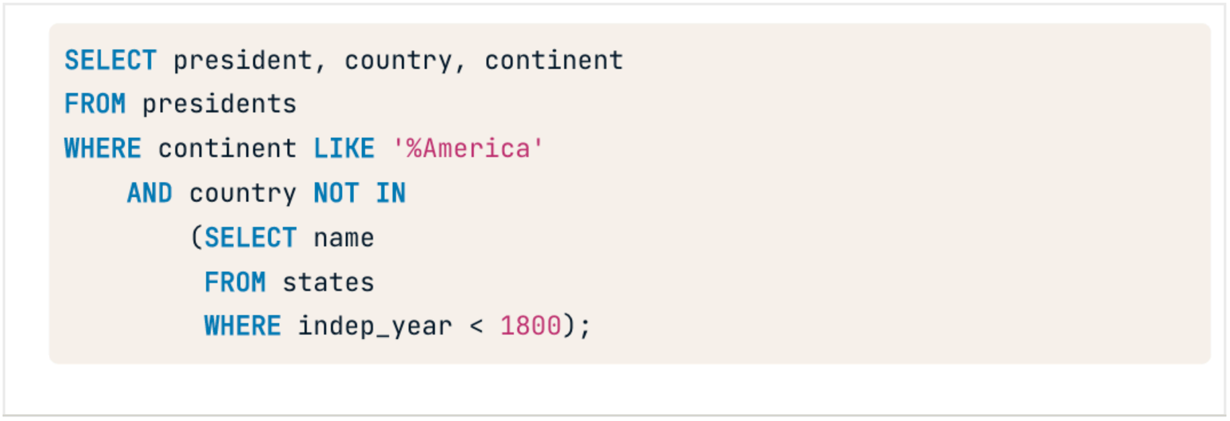
Semi-joins and Anti-joins



Finish the semi-join (an intro to subqueries)

```
SELECT president, country, continent
FROM presidents
WHERE country IN
    (SELECT name
     FROM states
     WHERE indep_year < 1800);
```

Semi join selects rows where the condition is met is met in the second table.
Anti Join selects the rows where the condition is not met in the second table.



```
SELECT president, country, continent
FROM presidents
WHERE continent LIKE '%America'
    AND country NOT IN
        (SELECT name
         FROM states
         WHERE indep_year < 1800);
```