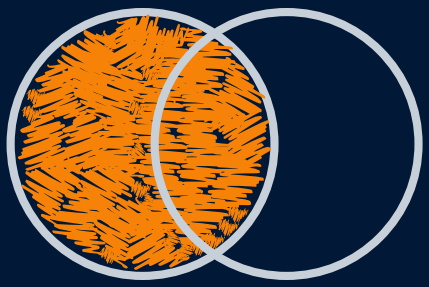
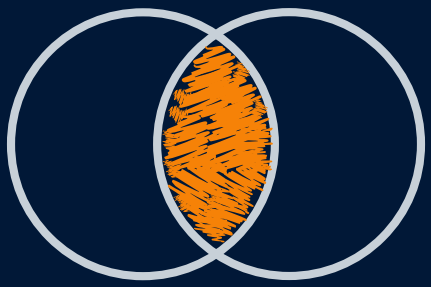
DATA SCIENCE INFINITY

SQL JOINS

A VISUAL GUIDE FOR ALL BRAINS



VS.



AAA				
BBB				

VS.

AAA				

VS.

DATA SCIENCE INFINITY

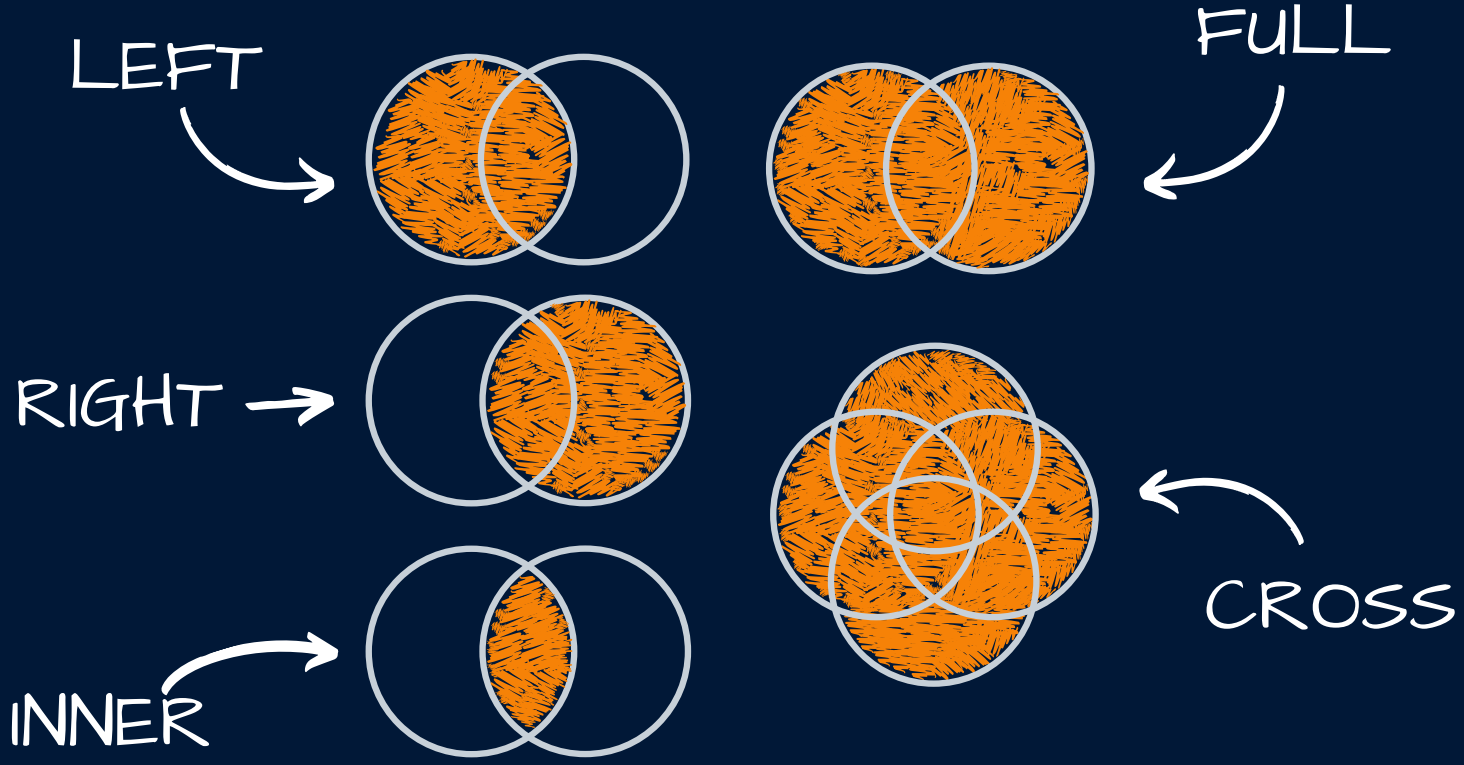
What is a Join?

A **JOIN** is used in SQL to merge together certain rows from two (or more) tables. This joining is done based upon one or more columns that are **shared** by each table.

Seldom will all our data be held in one giant table, so this logic is extremely useful!

There are **many different types of joins** we can utilise depending on what we need from each of the initial tables.

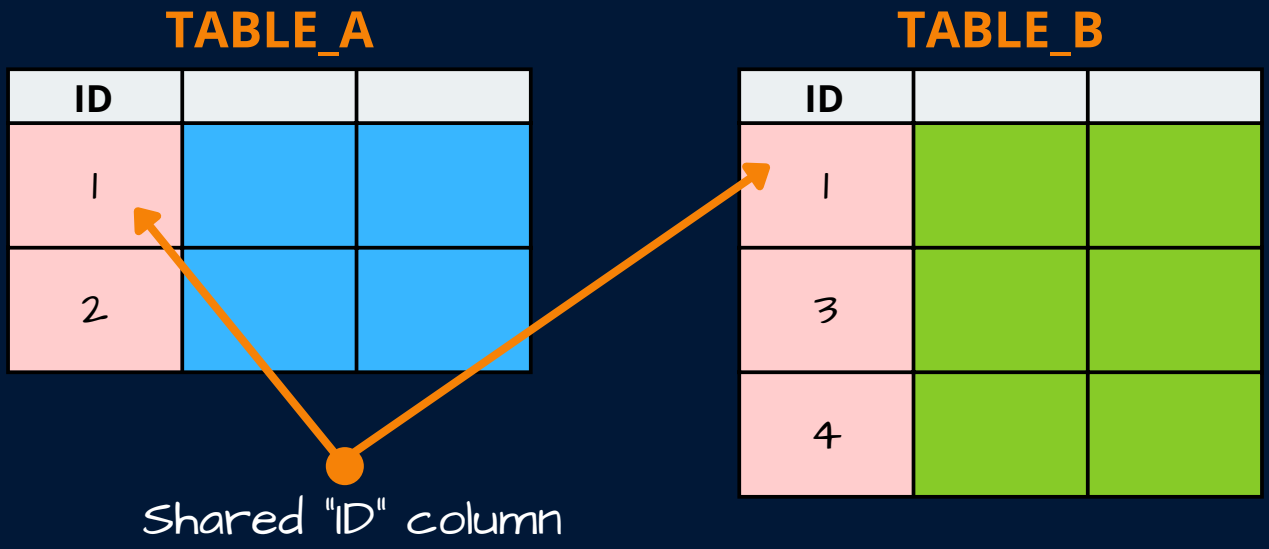
Commonly used joins are:



Our 2 base tables...

Below we have two tables, **TABLE_A** and **TABLE_B**.

Both tables contain the "ID" column. **TABLE_A** contains rows for ID's 1 & 2 whereas **TABLE_B** contains rows for ID's 1, 3, and 4



We could also represent these two tables more simply, like so...

Or...

We could *also* represent our two tables using my face, like below.

I'm not sure why we'd want to, but we could...

TABLE_A

Table A contains data about my face, and about some glorious hair on top

TABLE_B

Table B contains data about my face, and about a strong beard

INNER JOIN (Table)

An **INNER JOIN** returns only the rows where there is a match (on the join condition) in **both** tables!

In our example below, if we join on the ID column we are returned only the row for ID = 1 as it is the only row found in **both TABLE_A & TABLE_B**

TABLE_A

ID		
1		
2		

TABLE_B

ID		
1		
3		
4		

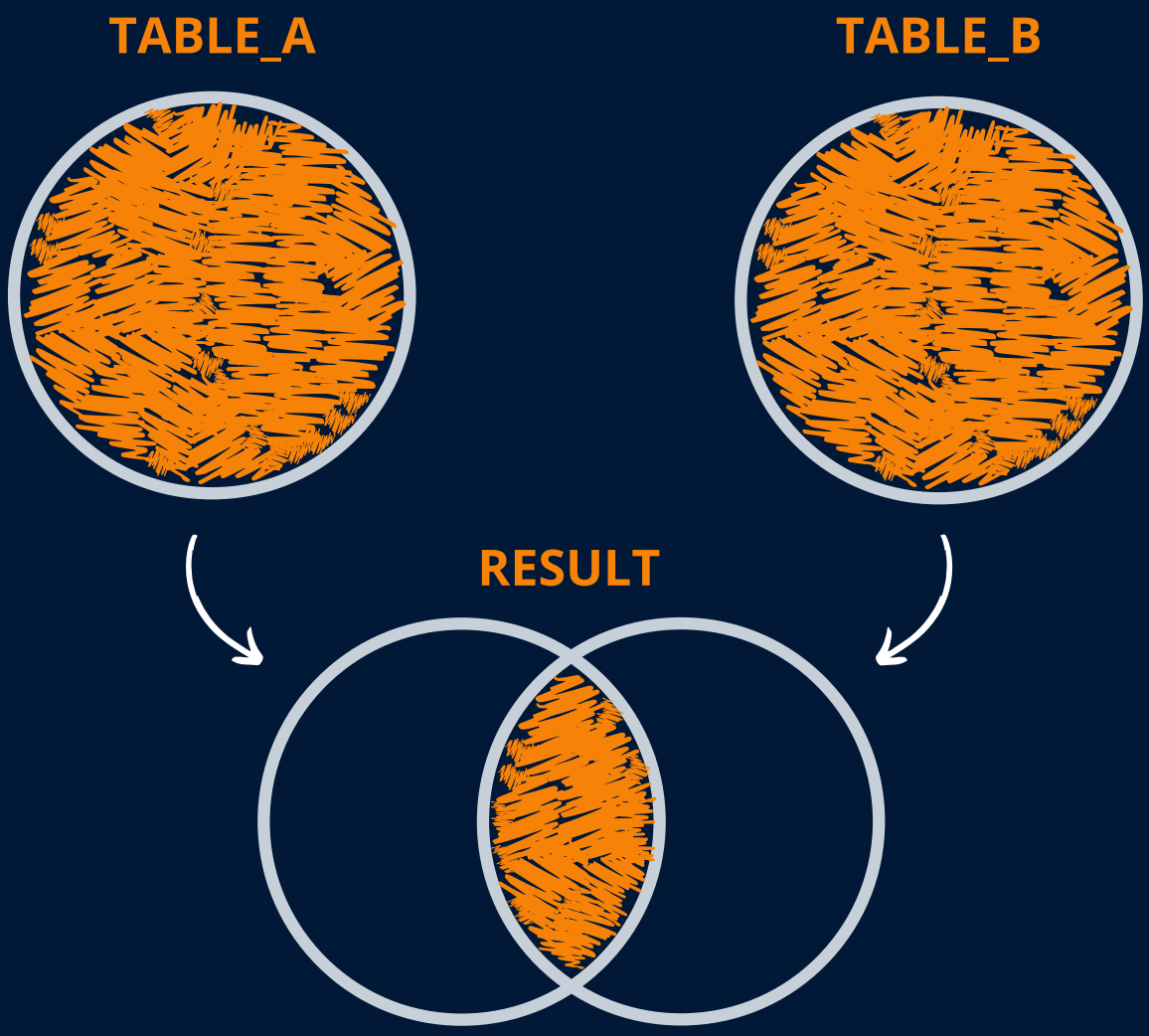
RESULT

ID				
1				

We are returned only **rows** where a match for ID was found but we will be returned all **columns** from each table (unless specified otherwise)

INNER JOIN (Venn)

Using a Venn Diagram, we can visualise the result of our **INNER JOIN** like so...



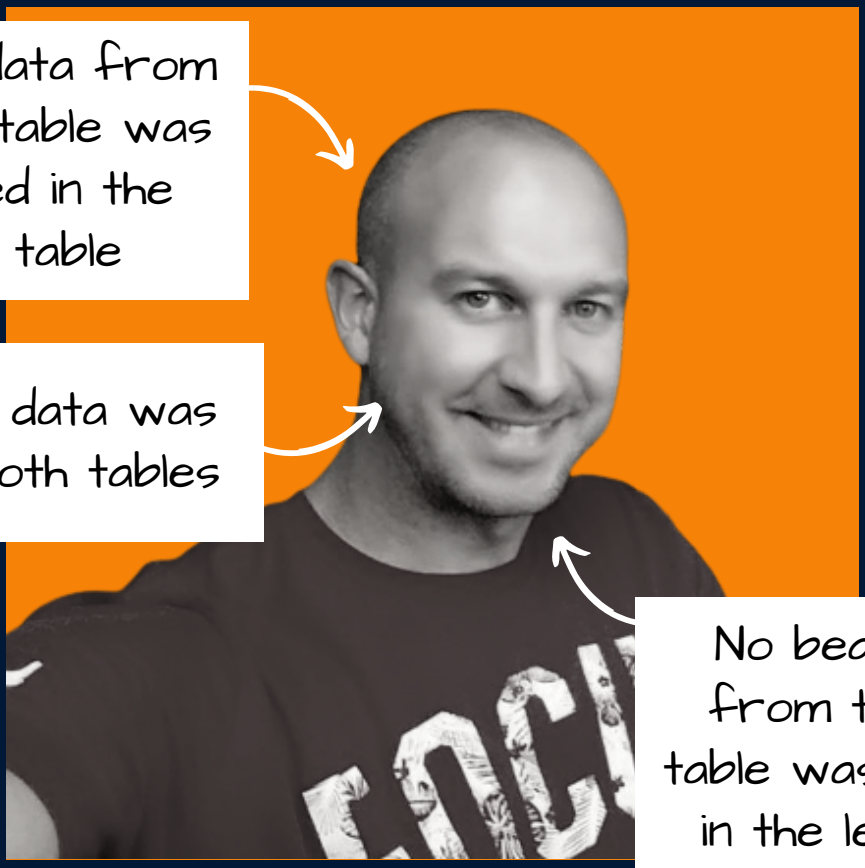
As we see, the **INNER JOIN** is only returning us rows of data where there is a match on the join condition in **both tables!**

INNER JOIN (My Face)

While the other methods are interesting - you really must *burn* the concepts into your mind...

Here we see the **INNER JOIN** - remember, the left table contains hair and face data, the right table contains face and beard data.

At this point, you should be fearful of what is to come later in this document...



DATA SCIENCE INFINITY

LEFT JOIN (Table)

A **LEFT JOIN** returns **all rows from the left table** (TABLE_A) even if there are no matches in the **right table** (TABLE_B)

TABLE_A

ID		
1		
2		

TABLE_B

ID		
1		
3		
4		

RESULT

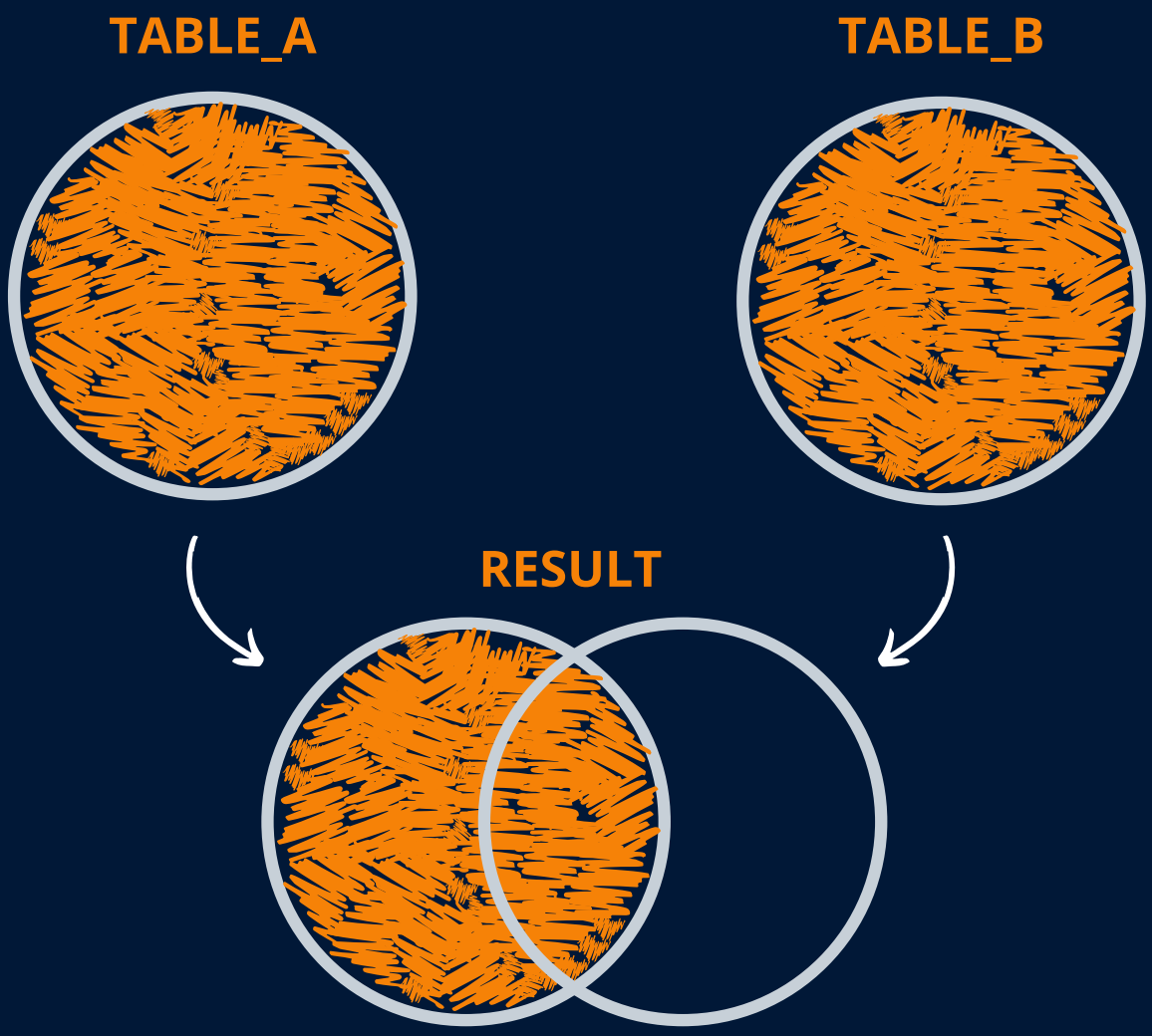
ID				
1				
2				

Where an ID match is found between the **left table** (TABLE_A) and the **right table** (TABLE_B), data from the right table is populated...

...where **no match is found**, we are returned null values!

LEFT JOIN (Venn)

Using a Venn Diagram, we can visualise the result of our **LEFT JOIN** like so...



As this shows, a **LEFT JOIN** gives us everything from **TABLE_A** and only returns data from **TABLE_B** where a match is found in the join logic!

LEFT JOIN (My Face)

Here we see the **LEFT JOIN**. Remember, the left table contains hair and face data, the right table contains face and beard data.

The face data & hair data are returned, but as the beard does not exist in the left, it is not returned.

Should I grow out my hair?!

RIGHT JOIN (Table)

A **RIGHT JOIN** is simply a reversed LEFT JOIN. It returns **all rows from the right table** (TABLE_B) even if there are no matches in the **left table** (Table_A)

TABLE_A

ID		
1		
2		

TABLE_B

ID		
1		
3		
4		

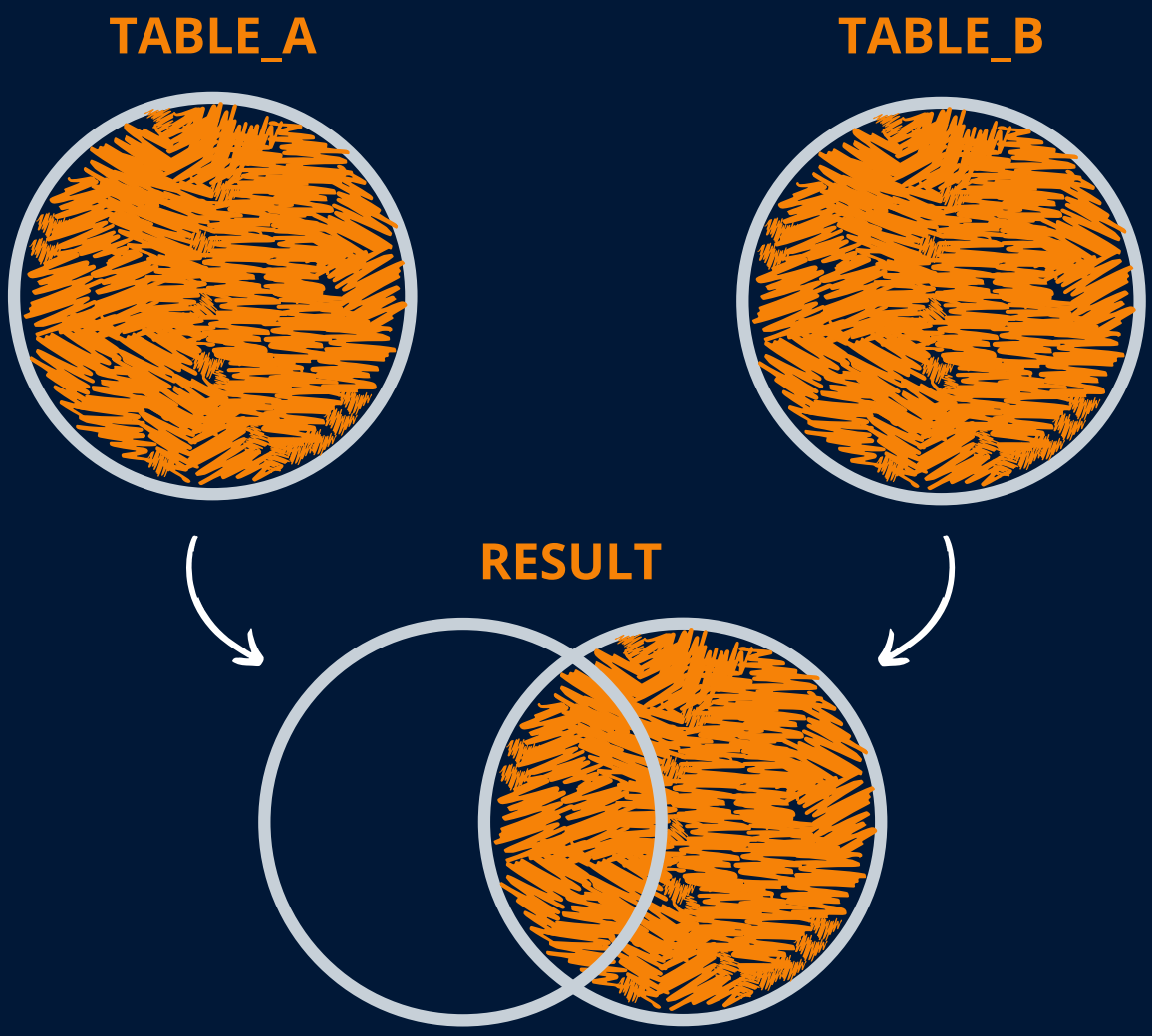
RESULT

ID				
1				
3				
4				

In practice, a **RIGHT JOIN** is seldom used as we could simply change TABLE_B to be the **left table**, and use a **LEFT JOIN** instead

RIGHT JOIN (Venn)

Using a Venn Diagram, we can visualise the result of our **RIGHT JOIN** like so...



As we see, a **RIGHT JOIN** gives us everything from **TABLE_B** and only returns data from **TABLE_A** where a match is found in the join logic!

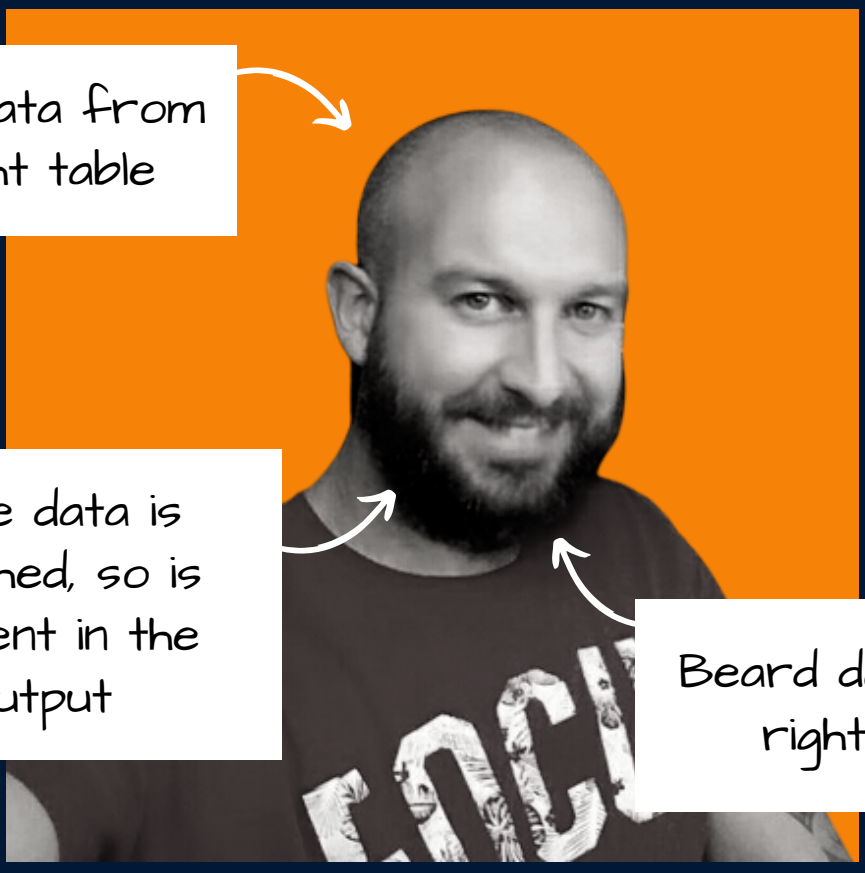
DATA SCIENCE INFINITY

RIGHT JOIN (My Face)

Here we see the **RIGHT JOIN**. Remember, the left table contains hair and face data, the right table contains face and beard data.

The face data & beard data are returned, but as the hair does not exist in the right, it is not returned.

I think the long hair looked better...



FULL JOIN (Table)

A **FULL JOIN** (often referred to as a FULL OUTER JOIN) returns all possible rows from each table. Where a match is found (based on the join condition) we are returned the values from the other table, otherwise we are returned null values...

TABLE_A

ID		
1		
2		

TABLE_B

ID		
1		
3		
4		

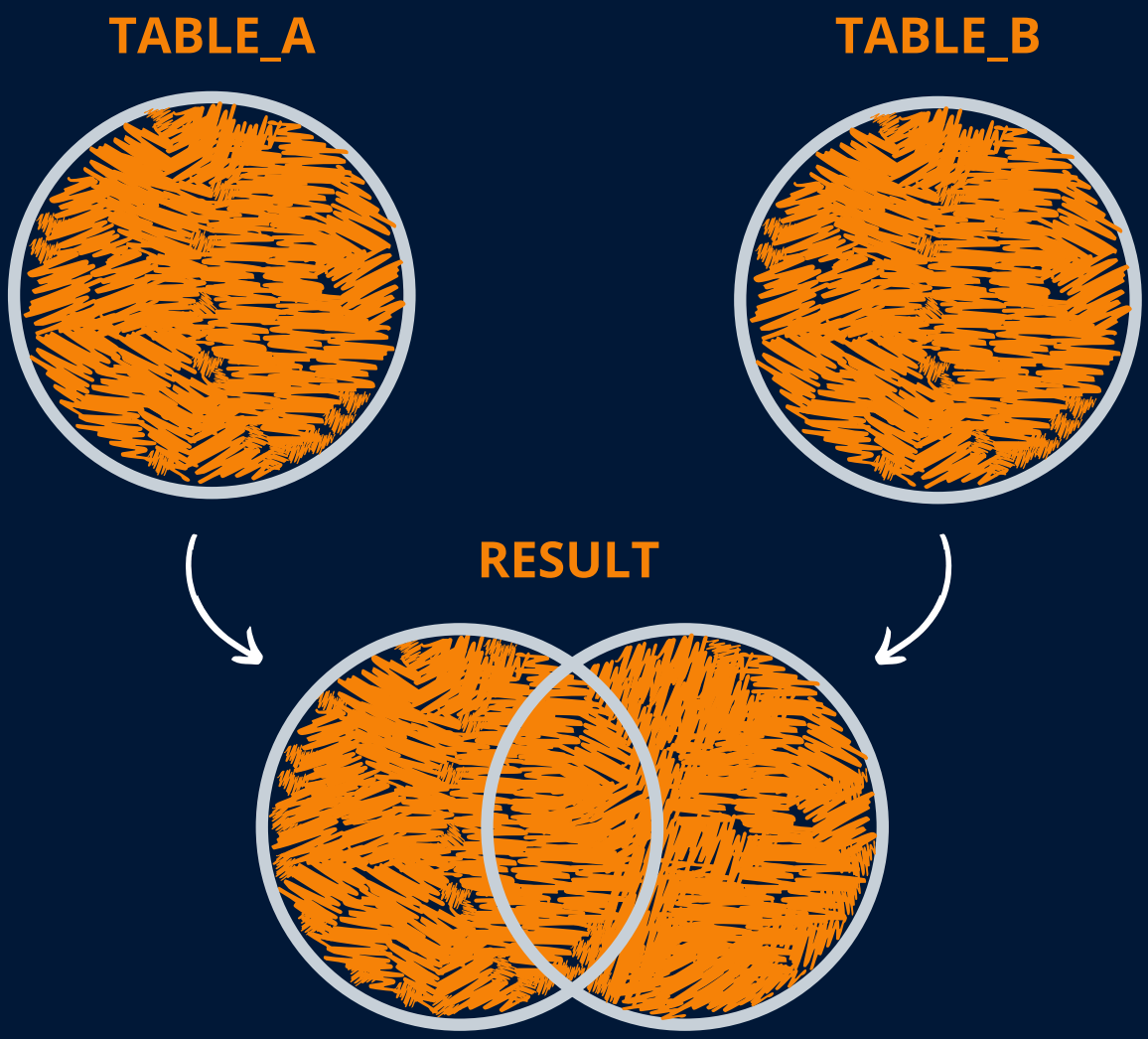
RESULT

ID				
1				
2				
3				
4				

A **FULL JOIN** is essentially like executing both a **LEFT JOIN** and **RIGHT JOIN** at the same time!

FULL JOIN (Venn)

Using a Venn Diagram, we can visualise the result of our **FULL JOIN** like so...



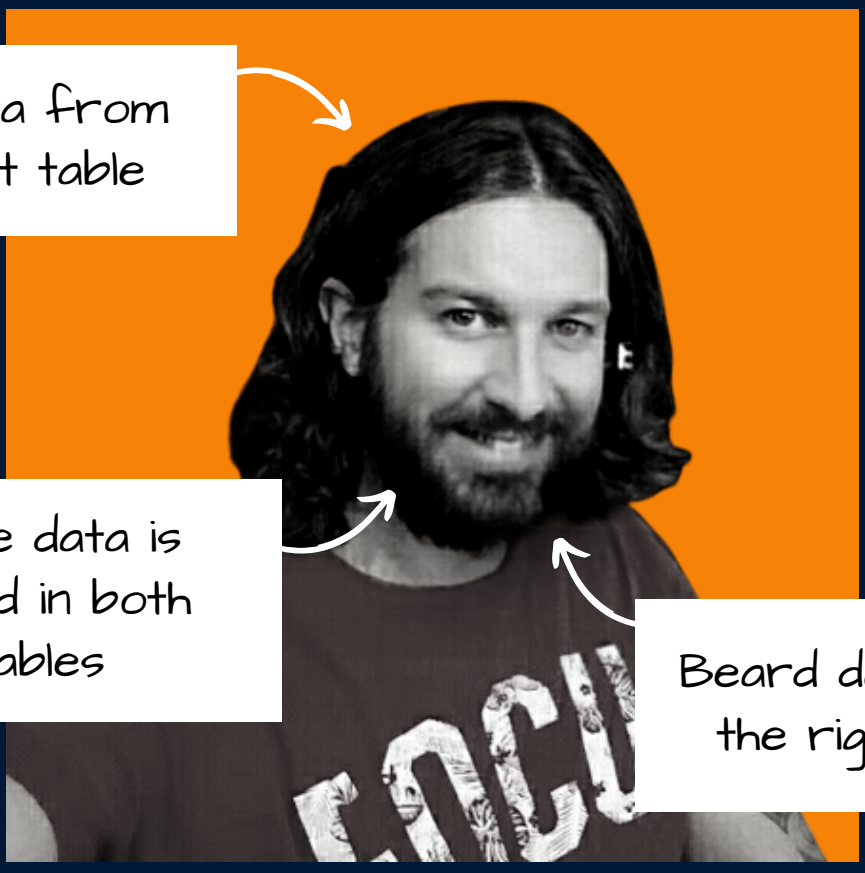
As we see, a **FULL JOIN** gives us everything from **TABLE_A** and everything from **TABLE_A**

FULL JOIN (My Face)

Here we see the **FULL JOIN**. Remember, the left table contains hair and face data, the right table contains face and beard data.

All data is returned here.

Is that me, or is that Cat Stevens?



CROSS JOIN (Table)

A **CROSS JOIN** returns all possible **combinations of rows** from both tables (also - there is no join condition applied)

TABLE_A

ID		
1		
2		

TABLE_B

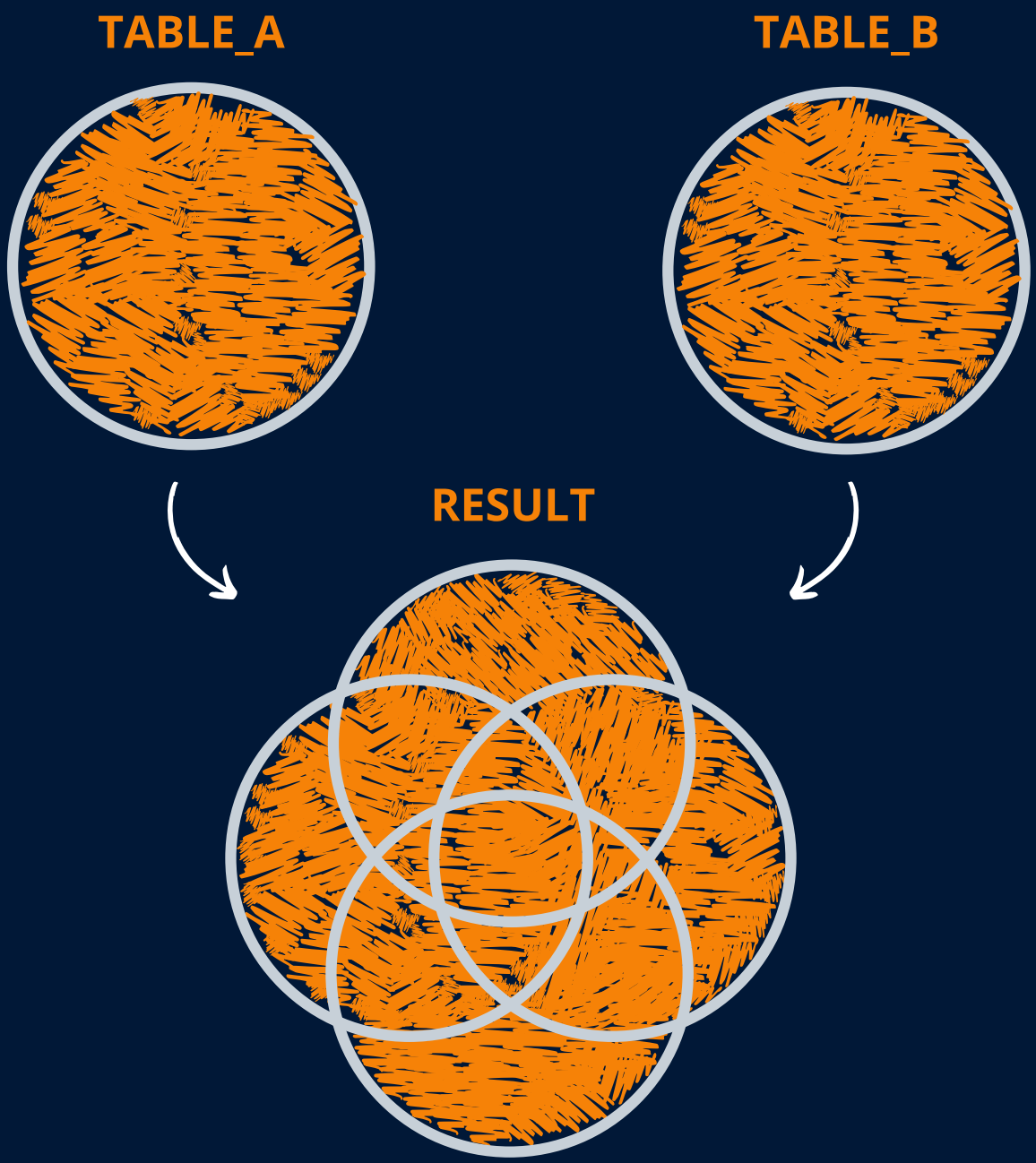
ID		
1		
3		
4		

RESULT

ID			ID		
1			1		
1			3		
1			4		
2			1		
2			3		
2			4		

CROSS JOIN (Venn)

Using a Venn Diagram, we can *somewhat* visualise the result of our **CROSS JOIN** like so...



CROSS JOIN (My Face)

Here we see the **CROSS JOIN**. Remember, the left table contains hair and face data, the right table contains face and beard data.

Our output contains a mix of them all!