Joining Data in SQL

Data across multiple tables

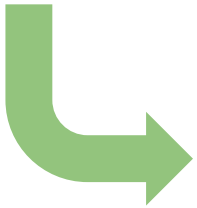
Very often data will be spread across **multiple tables**. In order to perform the analyses you need to it will be important to **combine the data**. There are many different ways to combine the data and each serves a slightly different purpose.

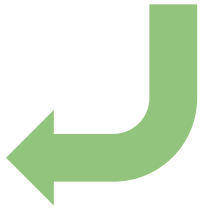
Table A

id	col_1	col_2
1	23-B	12
2	435	45
3	AB145	23
4	BB	56
5	435	123

Table B

id	lookup_id	col_a
a1	1	a
a2	2	a
b1	3	b
b2	4	b
a3	5	a





merge_table

id	col_1	col_2	col_a
1	23-B	12	a
2	435	45	a
3	AB145	23	b
4	BB	56	b
5	435	123	a

Keywords

- JOIN
  - INNER
  - LEFT
  - RIGHT
  - FULL
  - CROSS
  - self
  - semi
  - anti
- USING

- UNION
- UNION ALL
- INTERSECT
- EXCEPT

Let’s walk through a few examples using the **olympics** data

countries

id	country	region
1	AFG - Afghanistan	ASIA (EX. NEAR EAST)
2	ALB - Albania	EASTERN EUROPE
3	ALG - Algeria	NORTHERN AFRICA

Athletes

id	name	gender	age	height	weight
51	Nstor Abad Sanjun	M	23	167	64
55	Antonio Abadia Beci	M	26	170	65
110	Abubakar Abbas Abbas	M	20	175	66

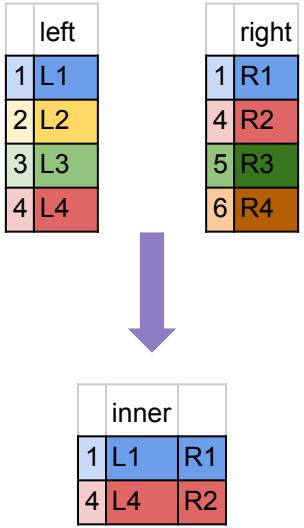
winter_sports

sport	event	year	athlete_id	country_id	bronze	silver	gold
Alpine Skiing	Alpine Skiing Women's Slalom	2014-01-01	126	89	NULL	NULL	NULL
Alpine Skiing	Alpine Skiing Women's Super G	2014-01-01	463	102	NULL	NULL	NULL
Alpine Skiing	Alpine Skiing Women's Giant Slalom	2014-01-01	463	102	NULL	NULL	NULL

INNER JOIN

Horizontally combining data from a **left** and **right** table is called **JOIN**ing. There are multiple kinds of joins and each one combines the data in a different way.

An **INNER JOIN** keeps all rows that have matching values in both tables



INNER JOIN

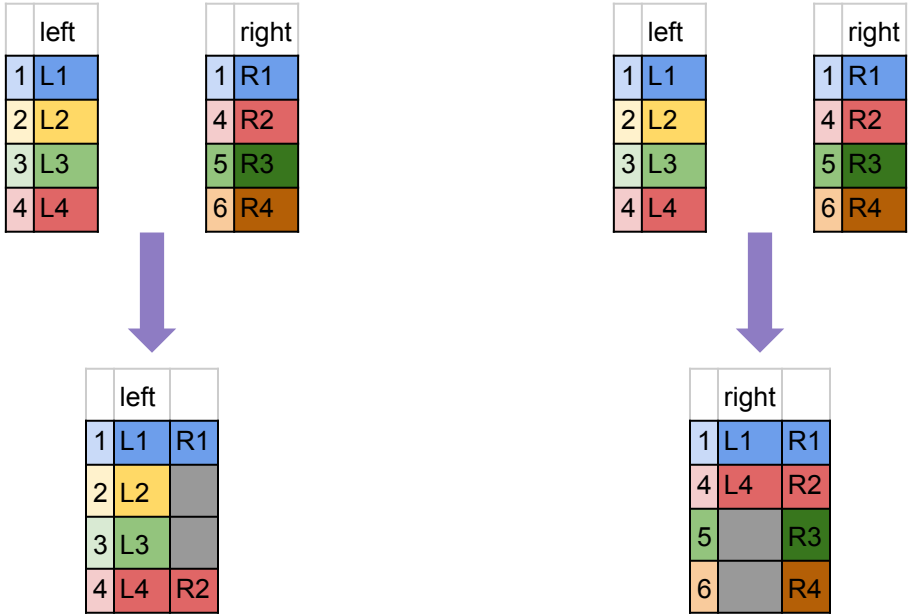
Add the athlete name to the **winter_sports** table.

```
SELECT sport, year, name, age
FROM winter_sports INNER JOIN athletes
ON winter_sports.athlete_id = athletes.id;
```

sport	year	name	age
Alpine Skiing	2014-01-01	Forough Abbasi	20
Alpine Skiing	2014-01-01	Agnese boltia	17
Alpine Skiing	2014-01-01	Agnese boltia	17
Alpine Skiing	2014-01-01	Agnese boltia	17
Alpine Skiing	2014-01-01	Iason Abramashvili	25
Alpine Skiing	2014-01-01	Iason Abramashvili	25

LEFT JOIN and RIGHT JOIN

A **LEFT JOIN** keeps all rows from the left table and all matching rows from the right table. A **RIGHT JOIN**, except all rows from the right table are kept.



You can use the **USING** keyword instead of **ON** if column names are the same in both tables.

LEFT JOIN

For **athletes** that competed in the **winter_games** what were their events in the **summer_games**, if any.

```
SELECT winter_games.athlete_id, winter_games.event, summer_games.event
FROM winter_games LEFT JOIN summer_games
USING(athlete_id);
```

athlete_id	event	event
127594	Cross Country Skiing Women's 4 x 5 kilometres Relay	Women's Marathon
127594	Cross Country Skiing Women's 15 km Skiathlon	Women's Marathon
127594	Cross Country Skiing Women's 30 kilometres	Women's Marathon
127594	Cross Country Skiing Women's 10 kilometres	Women's Marathon

RIGHT JOIN

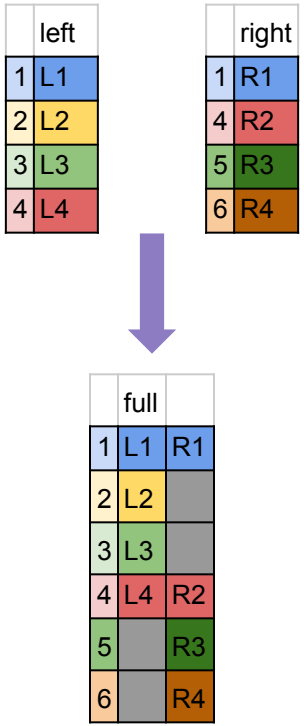
What **events** only have **athletes** that compete in the **summer_games**?

```
SELECT winter_games.athlete_id, winter_games.event, summer_games.event
FROM winter_games RIGHT JOIN summer_games
USING(athlete_id);
```

athlete_id	event	event
NULL	NULL	Gymnastics Men's Individual All-Around
NULL	NULL	Gymnastics Men's Floor Exercise
NULL	NULL	Gymnastics Men's Parallel Bars
NULL	NULL	Gymnastics Men's Horizontal Bar

FULL JOIN

A **FULL JOIN** keeps all rows from both tables:



FULL JOIN

What are all the **events** across both **winter_games** and **summer_games** and the **athletes** that competed in them?

```
SELECT winter_games.athlete_id, summer_games.athlete_id,
winter_games.event, summer_games.event
FROM winter_games FULL JOIN summer_games
USING(athlete_id);
```

athlete_id	athlete_id	event	event
NULL	135547	NULL	Women's 200 metres
NULL	135547	NULL	Women's 4 x 100 metres Relay
27520	NULL	Alpine Skiing Men's Slalom	NULL
27520	NULL	Alpine Skiing Men's Giant Slalom	NULL

self JOIN

There are other joins that are useful but don't have specific keywords associated with them. To use this, you will instead create a query that establishes the logic behind the join.

One example is a **self JOIN**, which merges a table with itself, usually to quickly make combinations and/or subset data within a table. you will need to use **aliasing** to distinguish each instance of the table and most likely will use some form of **aggregation**.

self JOIN

Find instances where the **gdp** for a **country** was lower in a subsequent **year**.

```
SELECT c1.country_id, c1.year, c1.gdp, c2.year, c2.gdp
FROM country_stats AS c1 INNER JOIN country_stats AS c2
USING(country_id)
WHERE c1.year < c2.year
AND c1.gdp > c2.gdp;
```

country_id	year	gdp	year	gdp
3	2000-01-01	54790245601	2001-01-01	54744714396
6	2000-01-01	9129594819	2001-01-01	8936063723
7	2000-01-01	830158777.8	2002-01-01	814615333.3
7	2000-01-01	830158777.8	2001-01-01	800740259.3

Set theory clauses

Set theory clauses stack data vertically or filter data. They involve two **SELECT** statements, specifying what tables to consider and how to combine them. The number of columns considered have to be the same.

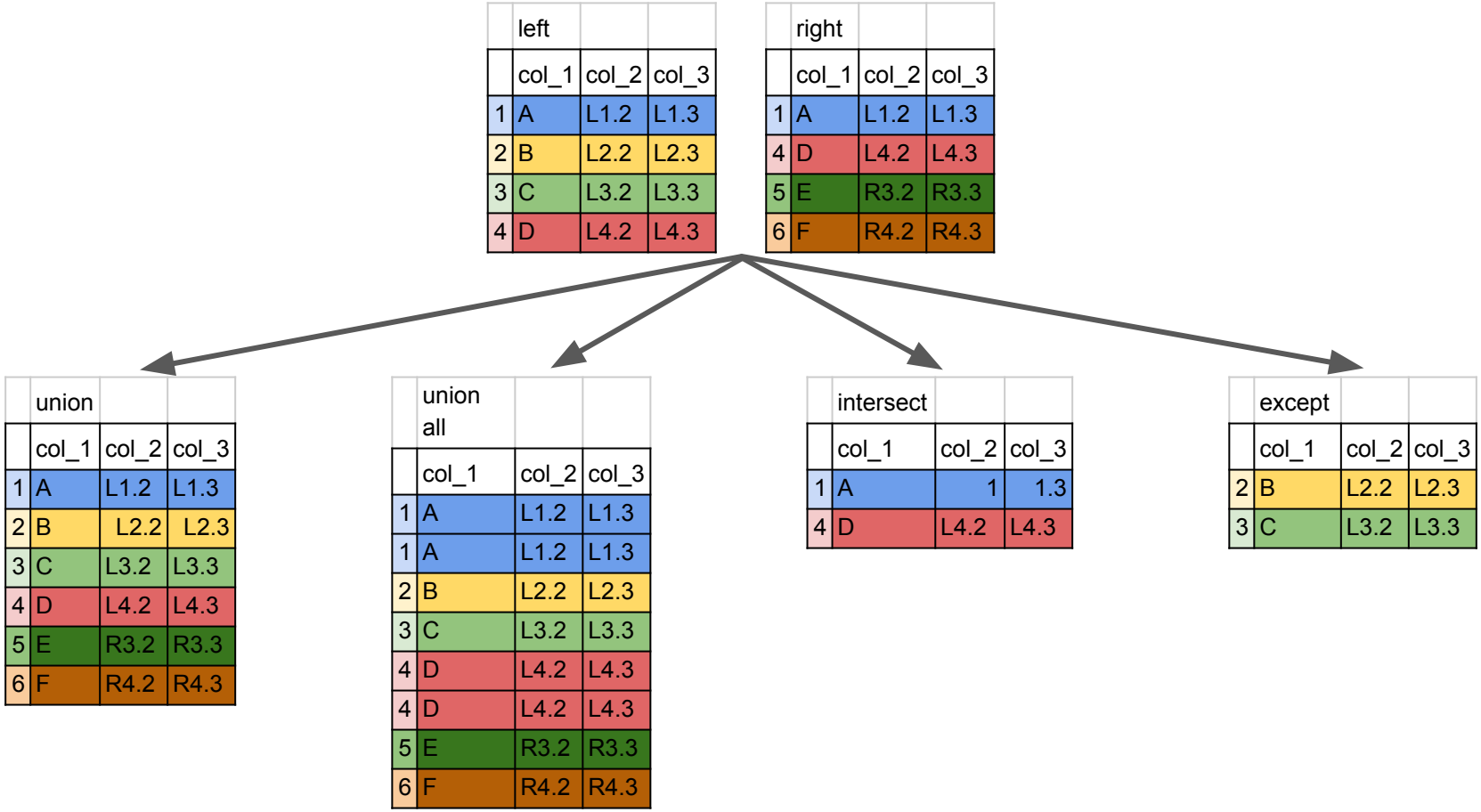
UNION takes all **unique** rows and stacks them on top of each other.

UNION ALL groups identical rows and stacks them on top of each other.

INTERSECT keeps rows that are identical in both **SELECT** statements.

EXCEPT keeps rows from the first **SELECT** that are not in the second.

Set theory clauses



UNION

Which countries competed in an olympic games?

```
SELECT country_id
FROM summer_games
UNION
SELECT country_id
FROM winter_games;
```

country_id
70
148
16
190
141

UNION ALL

Which countries competed in an olympic games? Repeats are different events.

```
SELECT country_id
FROM summer_games
UNION ALL
SELECT country_id
FROM winter_games;
```

country_id
173
173
173
15
19
19

INTERSECT

Which countries competed in both summer and winter games?

```
SELECT country_id
FROM summer_games
INTERSECT
SELECT country_id
FROM winter_games;
```

country_id
8
120
171
129
195

EXCEPT

Which countries competed in summer but not winter games?

```
SELECT country_id
FROM summer_games
EXCEPT
SELECT country_id
FROM winter_games;
```

country_id
80
106
7
141
202

Exercises

1. Count the number of **athletes** by **country name** in the **summer games**.

2. How many **events** did each **country** compete in across both **summer and winter games** (just use **country_id**, not country name).

3. Find **country names** where the **population** decreased from 2000 to 2006.