EXPLORE AI ACADEMY

Joins and unions

Joins and set operations

Introduction

Join operations

Joins are used to combine data from two or more tables based on **related column(s)** between them. The result of a join is a **new table** that includes columns from both tables, arranged side by side.

Types of join operations:

- `INNER JOIN`
- `LEFT JOIN` (or `LEFT OUTER JOIN`)
- `RIGHT JOIN` (or `RIGHT OUTER JOIN`)
- `FULL JOIN` (or `FULL OUTER JOIN`)

Set operations

Set operations are used to combine or exclude data based on the result of two SELECT queries. The columns in each **SELECT** query must have the same data types, and the number of columns in each query must be the same.

Types of set operations:

- `UNION`
- `INTERSECT`
- `EXCEPT`

The datasets

> To better understand join and union operations, we use two datasets that are SDG 6 related: **`Countries`** table which has a **`Country_id`**, **`Country_name`**, and the country's **`Population`**, and the **`Water_usage`** table which has data on **`Water_consumed_m3`** listed per **`Country_id`**.

`Countries`

Country_id	Country_name	Population
1	Country A	1000000
2	Country B	1500000
3	Country C	5000000
4	Country D	2000000

`Water_usage`

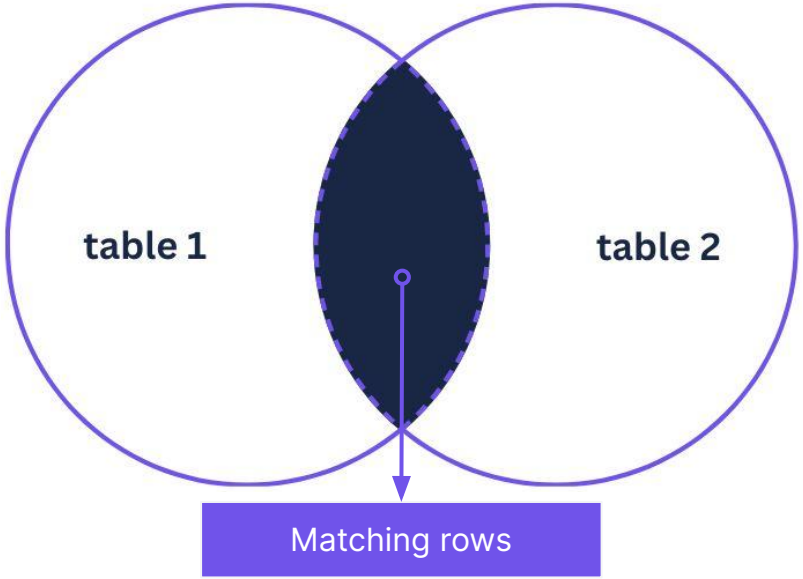
Country_id	Water_consumed_m3
1	500000
2	800000
3	3000000
5	1000000

INNER JOIN

An **`INNER JOIN`** is a type of join that **returns only the rows from both tables where there is a match between the specified columns in each table**. It filters out the rows that do not have corresponding matches in both tables.

```
SELECT
     columns
FROM
     table1
INNER JOIN
     table2
ON
     table1.column_name = table2.column_name;
```

Keyword that specifies the join condition, which is used to match rows from **`table1`** with rows from **`table2`** based on the specified column(s).



INNER JOIN

> To create a dataset with columns **Country_name**, **Population**, and **Water_consumed_m3**, containing only the information for countries where there is data available in both the **Countries** and **Water_usage** tables, we would utilise **INNER JOIN**.

Query

```
SELECT
      Countries.Country_name,
      Countries.Population,
      Water_usage.Water_consumed_m3
FROM
      Countries
INNER JOIN
      Water_usage
ON
      Countries.Country_id =
      Water_usage.Country_id;
```

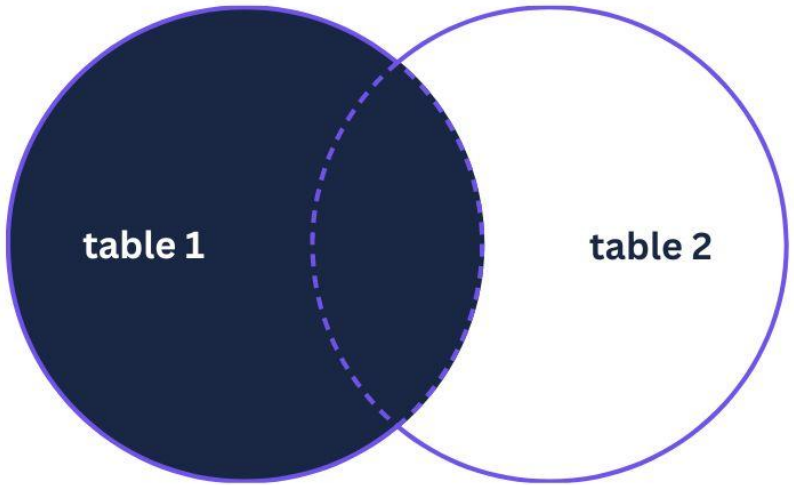
Output

Country_name	Population	Water_consumed_m3
Country A	1000000	500000
Country B	1500000	800000
Country C	5000000	3000000

LEFT JOIN

A **LEFT JOIN**, also known as a **LEFT OUTER JOIN**, returns all the **rows from the left table and only the matching rows from the right table**. If there is no match in the right table, **NULL** values are returned for the columns of the right table.

```
SELECT
     columns
FROM
     table1
LEFT JOIN
     table2
ON
     table1.column_name = table2.column_name;
```



LEFT JOIN

To generate a merged dataset with the columns **Country_name**, **Population**, and **Water_consumed_m3**, with details for all countries in the **Countries** table, and incorporating data from the **Water_usage** table (when available), you can employ the **LEFT JOIN** operation.

Query

```
SELECT
      Countries.Country_name,
      Countries.Population,
      Water_usage.Water_consumed_m3
FROM
      Countries
LEFT JOIN
      Water_usage
ON
      Countries.Country_id =
      Water_usage.Country_id;
```

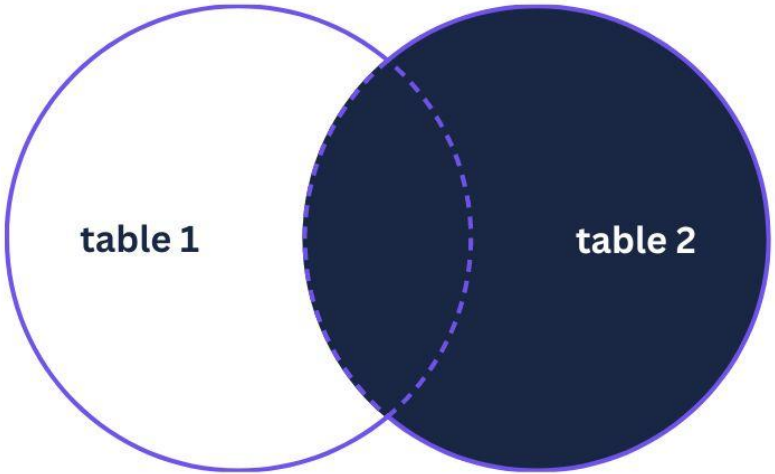
Output

Country_name	Population	Water_consumed_m3
Country A	1000000	500000
Country B	1500000	800000
Country C	5000000	3000000
Country D	2000000	NULL

RIGHT JOIN

A **`RIGHT JOIN`**, also known as a **`RIGHT OUTER JOIN`**, returns all the **rows from the right table and the matching rows from the left table**. If there is no match in the left table, **`NULL`** values are returned for the columns of the left table.

```
SELECT
     columns
FROM
     table1
RIGHT JOIN
     table2
ON
     table1.column_name = table2.column_name;
```



RIGHT JOIN

To create a dataset comprising the columns **Country_name**, **Population**, and **Water_consumed_m3**, it includes data for all water usage records from the **Water_usage** table, along with any available associated country details from the **Countries** table.

Query

```
SELECT
      Countries.Country_name,
      Countries.Population,
      Water_usage.Water_consumed_m3
FROM
      Countries
RIGHT JOIN
      Water_usage
ON
      Countries.Country_id =
      Water_usage.Country_id;
```

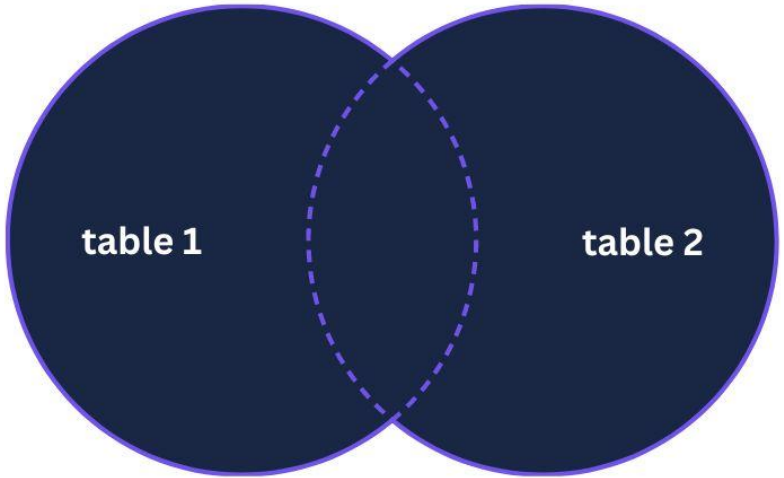
Output

Country_name	Population	Water_consumed_m3
Country A	1000000	500000
Country B	1500000	800000
Country C	5000000	3000000
NULL	NULL	1000000

FULL OUTER JOIN

> A **`FULL OUTER JOIN`** returns all the **rows from both tables, including unmatched rows from both the left and right tables.** If there is no match in either table, **`NULL`** values are returned for the columns of the respective table.

```
SELECT
     columns
FROM
     table1
FULL OUTER JOIN
     table2
ON
     table1.column_name = table2.column_name;
```



FULL OUTER JOIN

FULL OUTER JOIN can be used to create a dataset that combines columns **Country_name**, **Population**, and **Water_consumed_m3**, encompassing data for all countries from the **Countries** table as well as all water usage records from the **Water_usage** table.

Query

```
SELECT
      Countries.Country_name,
      Countries.Population,
      Water_usage.Water_consumed_m3
FROM
      Countries
FULL OUTER JOIN
      Water_usage
ON
      Countries.Country_id =
      Water_usage.Country_id;
```

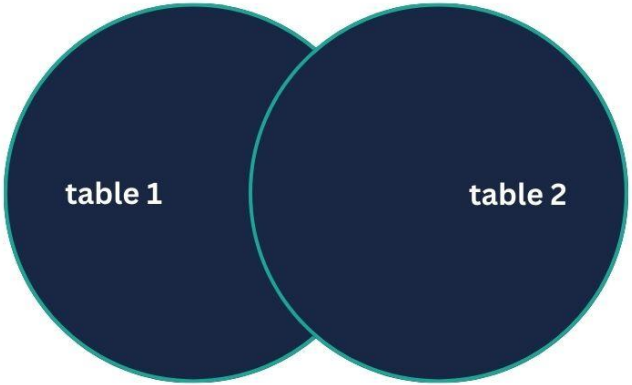
Output

Country_name	Population	Water_consumed_m3
Country A	1000000	500000
Country B	1500000	800000
Country C	5000000	3000000
Country D	2000000	NULL
NULL	NULL	1000000

UNION

> The **UNION** operator is **used to combine the results of two or more SELECT queries into a single result set**. It merges the rows from all queries while removing duplicates.

```
SELECT
     columns
FROM
     table1
UNION
SELECT
     columns
FROM
     table2;
```



The main difference between **FULL OUTER JOIN** and **UNION** is:
FULL OUTER JOIN combines rows from two tables, keeping all records and matching them when there is a common key, while including **NULLs** for non-matching rows. **UNION** combines the results of multiple **SELECT** queries, removing duplicates and returning only distinct rows. It does not include all unmatched rows from both sources like **FULL OUTER JOIN** does.

UNION

> **UNION** can be used to include the data from the **Water_usage** table in a combined result set while preserving the structure of the **Countries** table, with **NULL** values for **Country_name** and **Population** in place of specific data from the **Countries** table.

Query

```
SELECT
      Country_id, Country_name, Population
FROM
      Countries
UNION
SELECT
      Country_id, NULL, NULL
FROM
      Water_usage;
```

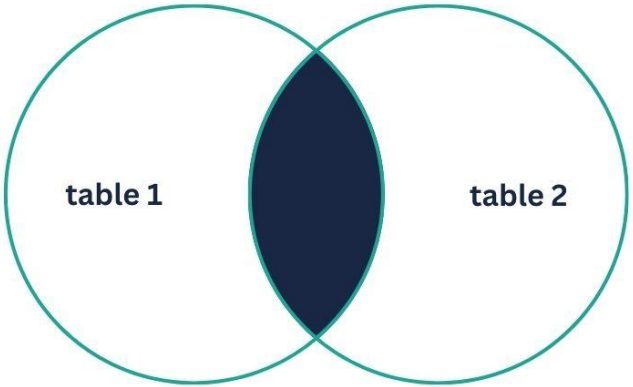
Output

Country_id	Country_name	Population
1	Country A	1000000
2	Country B	1500000
3	Country C	5000000
4	Country D	2000000
1	NULL	NULL
2	NULL	NULL
3	NULL	NULL
5	NULL	NULL

INTERSECT

> The **INTERSECT** operator is used to combine the **results of two or more SELECT queries and returns only the rows that appear in all the result sets**. It retrieves the common rows between the queries while eliminating any duplicate rows.

```
SELECT
     columns
FROM
     table1
INTERSECT
SELECT
     columns
FROM
     table2;
```



The main difference between **INNER JOIN** and **INTERSECT** is:
INNER JOIN combines rows from two or more tables based on a specified join condition, while **INTERSECT** combines the results of multiple queries and retrieves only the rows that are common to all the queries.

INTERSECT

> The **INTERSECT** operator can be used to find the common rows between two datasets.

Query

```
SELECT
      Country_id
FROM
      Countries
INTERSECT
SELECT
      Country_id
FROM
      Water_usage;
```

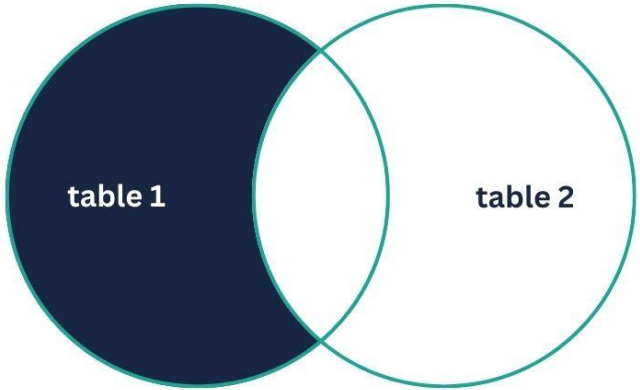
Output

Country_id
1
2
3

EXCEPT

> **EXCEPT** is used to retrieve the rows that appear in the first **SELECT** query but not in the second **SELECT** query. It is used to find the difference between the results of two queries.

```
SELECT
     columns
FROM
     table1
EXCEPT
SELECT
     columns
FROM
     table2;
```



The main difference between **LEFT JOIN** and **EXCEPT** is:
LEFT JOIN combines rows from two tables based on a join condition, ensuring all rows from the left table are included. **EXCEPT** does not involve any join and retrieves rows that are unique to the first query and not present in the second query.

EXCEPT

> **EXCEPT** is valuable for obtaining distinct **Country_id** values from the **Countries** table that are absent in the **Water_usage** table.

Query

```
SELECT
      Country_id
FROM
      Countries
EXCEPT
SELECT
      Country_id
FROM
      Water_usage;
```

Output

Country_id
4