**SQL UNIONs – What is a UNION?**

A **UNION** in SQL is used to combine **rows** from two or more **SELECT queries** into a **single result set**. It’s important to note that the queries being combined must have:

* The **same number of columns**.
* The columns must be of **compatible data types**.

**UNION** removes **duplicate rows** by default. If you want to include duplicates, you can use **UNION ALL**.

**Types of UNIONs**

1. **UNION**:
   * Combines the result sets from two or more queries, removing any duplicate rows.
2. **UNION ALL**:
   * Combines the result sets from two or more queries and **keeps all rows**, including duplicates.

**Example of a UNION**

Let’s say we have two tables:

**1. customers table:**

| **customer\_id** | **customer\_name** |
| --- | --- |
| 1 | Alice |
| 2 | Bob |
| 3 | Charlie |

**2. suppliers table:**

| **supplier\_id** | **supplier\_name** |
| --- | --- |
| 101 | Dave |
| 102 | Eve |
| 103 | Alice |

**SQL Query Using UNION:**

SELECT customer\_name FROM customers

UNION

SELECT supplier\_name FROM suppliers;

**Result After UNION (removes duplicates):**

| **name** |
| --- |
| Alice |
| Bob |
| Charlie |
| Dave |
| Eve |

**Explanation:**

* The **UNION** operator combines the results of the two queries: one selecting **customer\_name** from the customers table and the other selecting **supplier\_name** from the suppliers table.
* Since the names are the same for Alice in both tables, **UNION** removes the **duplicate** Alice from the result set.

**SQL Query Using UNION ALL:**

SELECT customer\_name FROM customers

UNION ALL

SELECT supplier\_name FROM suppliers;

**Result After UNION ALL (keeps duplicates):**

| **name** |
| --- |
| Alice |
| Bob |
| Charlie |
| Dave |
| Eve |
| Alice |

**Explanation:**

* This time, we use **UNION ALL** which **keeps all rows**, including duplicates.
* **Alice** appears twice because she exists in both the customers and suppliers tables.

**Key Differences Between UNION and JOIN:**

| **Feature** | **JOIN** | **UNION** |
| --- | --- | --- |
| Combines... | **Columns side by side** | **Rows on top of each other** |
| Based on... | A **related column** (usually keys) | **Structure** (same number & type of columns) |
| Returns... | **Wider rows** (more columns) | **Taller result set** (more rows) |
| Removes duplicates? | No | Yes, unless you use **UNION ALL** |
| Use Case Example | Combine employee data with department data | Combine customer and supplier lists |

**When to Use JOIN vs. UNION:**

* **JOIN**: Use when you need to **combine columns** from two or more tables based on a **common key**.
* **UNION**: Use when you want to **combine rows** from two or more queries with **compatible structures** (same number and type of columns).

That’s a basic rundown of **UNIONs**. Does that make sense, or would you like more examples?

**You said:**

So, UNIONs do not require a common column, i.e. foreign key, because it's combining rows. Yes?

**ChatGPT said:**

Yes, exactly! You’ve got it.

**Key Point about UNIONs:**

Unlike **JOINs**, which require a **common column** (typically a key) to match rows between tables, **UNIONs** only require that the **queries** you're combining have the same **number of columns** and **compatible data types**.

Here’s why:

* **UNION** is focused on **combining rows** from two or more SELECT queries, not on matching specific data between tables like **JOIN** does.
* You’re essentially stacking the results vertically, one on top of the other.

**To clarify:**

* **JOIN** matches rows based on a common column (like a foreign key).
* **UNION** simply combines rows with the same structure (same number of columns, same data types) without any need for matching columns.

**Example Recap:**

If you want to combine **customer names** and **supplier names**, they don’t need to be related by anything like a foreign key. You just ensure that both the customers and suppliers queries return the **same type and number of columns** (in this case, just name), and then you can use a **UNION** to combine them.

For example:

SELECT customer\_name FROM customers

UNION

SELECT supplier\_name FROM suppliers;

Here, the **two tables don’t need a common column** — you're just stacking the names from both tables into one result set.

Let me know if that clears things up!

**You said:**

SELECT customer\_name FROM customers UNION SELECT supplier\_name FROM suppliers; What's the output?

**ChatGPT said:**

Let’s break it down step by step based on the **UNION** operation, which combines rows from two queries and **removes duplicates** by default.

**Given Tables:**

**customers table:**

| **customer\_id** | **customer\_name** |
| --- | --- |
| 1 | Alice |
| 2 | Bob |
| 3 | Charlie |

**suppliers table:**

| **supplier\_id** | **supplier\_name** |
| --- | --- |
| 101 | Dave |
| 102 | Eve |
| 103 | Alice |

**SQL Query:**

SELECT customer\_name FROM customers

UNION

SELECT supplier\_name FROM suppliers;

**How the UNION Works:**

* **First Query:** SELECT customer\_name FROM customers;
  + This will return:
    - Alice
    - Bob
    - Charlie
* **Second Query:** SELECT supplier\_name FROM suppliers;
  + This will return:
    - Dave
    - Eve
    - Alice
* **UNION** combines the results from both queries, but removes duplicates.

**Output After UNION:**

| **customer\_name** |
| --- |
| Alice |
| Bob |
| Charlie |
| Dave |
| Eve |

**Why this output?**

* **Alice** appears only **once** in the result set, even though she exists in both customers and suppliers tables (because **UNION** removes duplicates).
* **Bob, Charlie, Dave, and Eve** each appear only once because they appear only in one of the tables.

If you were to use **UNION ALL** instead, then **Alice** would appear **twice** (once for each table).

Let me know if you'd like further clarification or more examples!