In SQL, both UNION and UNION ALL are used to combine the results of two or more SELECT statements. However, there are key differences between them in how they handle duplicate records.

**Key Differences:**

* **UNION**: Removes duplicate records from the result set.
* **UNION ALL**: Includes all records, including duplicates.

Let’s go through detailed explanations and examples.

**1. UNION (Removes Duplicates)**

The UNION operator combines the results of two or more SELECT queries and returns only **distinct** rows (i.e., removes duplicates).

**Syntax:**

SELECT column\_list FROM table1

UNION

SELECT column\_list FROM table2;

**Example:**

Suppose we have two tables, Employees\_A and Employees\_B:

* **Employees\_A**:

|  |  |
| --- | --- |
| **EmployeeID** | **Name** |
| 101 | Alice |
| 102 | Bob |
| 103 | Charlie |

* **Employees\_B**:

|  |  |
| --- | --- |
| **EmployeeID** | **Name** |
| 102 | Bob |
| 104 | David |
| 105 | Eve |

Now, let's combine these two tables using UNION:

SELECT Name FROM Employees\_A

UNION

SELECT Name FROM Employees\_B;

**Result** (Note that duplicates, like Bob, are removed):

|  |
| --- |
| **Name** |
| Alice |
| Bob |
| Charlie |
| David |
| Eve |

**2. UNION ALL (Keeps Duplicates)**

The UNION ALL operator combines the results of two or more SELECT queries and returns **all** rows, including duplicates.

**Syntax:**

SELECT column\_list FROM table1

UNION ALL

SELECT column\_list FROM table2;

**Example:**

Using the same Employees\_A and Employees\_B tables as before, let’s combine the data using UNION ALL:

SELECT Name FROM Employees\_A

UNION ALL

SELECT Name FROM Employees\_B;

**Result** (Duplicates are kept):

|  |
| --- |
| **Name** |
| Alice |
| Bob |
| Charlie |
| Bob |
| David |
| Eve |

As you can see, **Bob** appears twice in the result set because UNION ALL does not remove duplicates.

**Key Considerations:**

1. **Performance**:
   * UNION performs a **deduplication** step, which can be more resource-intensive, especially with large data sets.
   * UNION ALL is generally faster because it does not perform duplicate elimination.
2. **Column Data Types**:
   * The number of columns and their data types in all SELECT statements must match for both UNION and UNION ALL.
   * If the columns differ in number or type, SQL will throw an error.

**Example: Combining Different Tables with UNION and UNION ALL**

Consider two tables, Sales\_2023 and Sales\_2024:

* **Sales\_2023**:

|  |  |
| --- | --- |
| **SaleID** | **Amount** |
| 1 | 100 |
| 2 | 200 |
| 3 | 300 |

* **Sales\_2024**:

|  |  |
| --- | --- |
| **SaleID** | **Amount** |
| 2 | 200 |
| 4 | 400 |
| 5 | 500 |

**Using UNION to Combine Sales:**

SELECT SaleID, Amount FROM Sales\_2023

UNION

SELECT SaleID, Amount FROM Sales\_2024;

**Result** (Distinct rows only):

|  |  |
| --- | --- |
| **SaleID** | **Amount** |
| 1 | 100 |
| 2 | 200 |
| 3 | 300 |
| 4 | 400 |
| 5 | 500 |

**Using UNION ALL to Combine Sales:**

SELECT SaleID, Amount FROM Sales\_2023

UNION ALL

SELECT SaleID, Amount FROM Sales\_2024;

**Result** (All rows, including duplicates):

|  |  |
| --- | --- |
| **SaleID** | **Amount** |
| 1 | 100 |
| 2 | 200 |
| 3 | 300 |
| 2 | 200 |
| 4 | 400 |
| 5 | 500 |

**Summary of Differences:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **UNION** | **UNION ALL** |
| Duplicates | Removes duplicates | Keeps duplicates |
| Performance | Slower (because of deduplication) | Faster (no deduplication) |
| Use Case | When you need distinct values | When duplicates are allowed/needed |