

SQL JOINS and UNIONS

SQL JOINS

JOINS in SQL are used to retrieve data from two or more tables based on a related column. There are different types of joins, each designed for a specific use case.

Types of SQL JOINS

a. INNER JOIN

- **Returns:** Only the rows with matching values in both tables.
- **Syntax:**

```
1 SELECT columns
2 FROM table1
3 INNER JOIN table2
4 ON table1.common_column = table2.common_column;
5
```

- **Example:**

```
1 SELECT employees.name, departments.department_name
2 FROM employees
3 INNER JOIN departments
4 ON employees.department_id =
   departments.department_id;
5
```

Result: Only employees assigned to a department.

b. LEFT JOIN (or LEFT OUTER JOIN)

- **Returns:** All rows from the left table and matching rows from the right table. If no match, NULL values are included for columns from the right table.
- **Syntax:**

```
1 SELECT columns
2 FROM table1
3 LEFT JOIN table2
4 ON table1.common_column = table2.common_column;
5
```

- **Example:**

```
SQL
1 SELECT employees.name, departments.department_name
2 FROM employees
3 LEFT JOIN departments
4 ON employees.department_id =
   departments.department_id;
5
```

Result: All employees, even those not assigned to a department (with NULL for department_name).

c. RIGHT JOIN (or RIGHT OUTER JOIN)

- **Returns:** All rows from the right table and matching rows from the left table. If no match, NULL values are included for columns from the left table.
- **Syntax:**

```
SQL
1 SELECT columns
2 FROM table1
3 RIGHT JOIN table2
4 ON table1.common_column = table2.common_column;
5
```

- **Example:**

```
SQL
1 SELECT employees.name, departments.department_name
2 FROM employees
3 RIGHT JOIN departments
4 ON employees.department_id =
   departments.department_id;
5
```

Result: All departments, even those without employees (with NULL for name).

d. FULL JOIN (or FULL OUTER JOIN)

- **Returns:** All rows from both tables. Rows without a match in one table will include NULL values for columns from the unmatched table.
- **Syntax:**

```
SQL  ▾  
1  SELECT columns  
2  FROM table1  
3  FULL JOIN table2  
4  ON table1.common_column = table2.common_column;  
5
```

- **Example:**

```
SQL  ▾  
1  SELECT employees.name, departments.department_name  
2  FROM employees  
3  FULL JOIN departments  
4  ON employees.department_id =  
   departments.department_id;  
5
```

Result: All employees and all departments, including unmatched rows from both tables.

e. CROSS JOIN

- **Returns:** A Cartesian product of both tables (all combinations of rows).
- **Syntax:**

```
SQL  ▾  
1  SELECT columns  
2  FROM table1  
3  CROSS JOIN table2;  
4
```

- **Example:**

```
SQL  ▾  
1  SELECT employees.name, projects.project_name
```

```
2 FROM employees
3 CROSS JOIN projects;
4
```

Result: Every employee paired with every project.

f. SELF JOIN

- **Description:** A table is joined with itself. Often used for hierarchical data.
- **Syntax:**

```
1 SELECT a.column1, b.column2
2 FROM table a
3 INNER JOIN table b
4 ON a.common_column = b.common_column;
5
```

- **Example:**

```
1 SELECT e1.name AS employee, e2.name AS manager
2 FROM employees e1
3 INNER JOIN employees e2
4 ON e1.manager_id = e2.employee_id;
5
```

Result: Matches employees with their managers.

2. SQL UNION

UNION is used to combine the result sets of two or more `SELECT` statements. Duplicate rows are removed by default.

Key Rules for UNION:

1. Each `SELECT` statement must have the same number of columns.
2. The columns must have compatible data types.
3. Column order matters.

a. UNION

- **Returns:** Combined result set with duplicates removed.

- **Syntax:**

```
1 SELECT column1, column2
2 FROM table1
3 UNION
4 SELECT column1, column2
5 FROM table2;
6
```

- **Example:**

```
1 SELECT name, city
2 FROM customers_us
3 UNION
4 SELECT name, city
5 FROM customers_uk;
6
```

Result: Unique customers from both the US and UK.

b. UNION ALL

- **Returns:** Combined result set with duplicates included.
- **Syntax:**

```
1 SELECT column1, column2
2 FROM table1
3 UNION ALL
4 SELECT column1, column2
5 FROM table2;
6
```

- **Example:**

```
1 SELECT name, city
2 FROM customers_us
3 UNION ALL
4 SELECT name, city
5 FROM customers_uk;
```

6

Result: All customers from both tables, including duplicates.

c. INTERSECT

- **Returns:** Rows that exist in both result sets.
- **Syntax:**

```
1 SELECT column1, column2
2 FROM table1
3 INTERSECT
4 SELECT column1, column2
5 FROM table2;
6
```

SQL



- **Example:**

```
1 SELECT name, city
2 FROM customers_us
3 INTERSECT
4 SELECT name, city
5 FROM customers_uk;
6
```

SQL



Result: Customers common to both the US and UK.

d. EXCEPT (or MINUS in some databases)

- **Returns:** Rows from the first result set that do not exist in the second result set.
- **Syntax:**

```
1 SELECT column1, column2
2 FROM table1
3 EXCEPT
4 SELECT column1, column2
5 FROM table2;
6
```

SQL



- **Example:**

```
1 SELECT name, city
2 FROM customers_us
3 EXCEPT
4 SELECT name, city
5 FROM customers_uk;
6
```

Result: Customers in the US but not in the UK.

Summary

- **JOINS** combine rows from multiple tables based on related columns.
- **UNION** combines rows from multiple result sets, either removing or retaining duplicates.
- **INTERSECT** finds common rows between result sets.
- **EXCEPT** finds rows in one result set but not in another.

Would you like practice queries or further clarification on a specific type?