Let's go **step by step** and cover **Joins in SQL**, including types like **Natural, Cross, Self, Inner, Outer (Left, Right, Full)**. I’ll explain with examples so it’s crystal clear.

**1. What is a Join?**

A **join** combines rows from two or more tables based on a related column. It’s essential for querying **related data stored in multiple tables**.

**2. Types of Joins in SQL**

**A. INNER JOIN (IJ)**

* Returns **only matching rows** from both tables.
* Syntax:

SELECT a.column1, b.column2

FROM TableA a

INNER JOIN TableB b

ON a.common\_column = b.common\_column;

* **Example:**

SELECT Students.Name, Department.DeptName

FROM Students

INNER JOIN Department

ON Students.DeptID = Department.DeptID;

* **Result:** Only students who belong to a department.

**B. OUTER JOIN (OJ)**

* Returns **matching rows + unmatched rows** from one or both tables.
* Types:

**1. LEFT OUTER JOIN (LOJ)**

* Returns **all rows from left table**, matched rows from right table, NULL if no match.

SELECT Students.Name, Department.DeptName

FROM Students

LEFT OUTER JOIN Department

ON Students.DeptID = Department.DeptID;

* Students without a department will have NULL for DeptName.

**2. RIGHT OUTER JOIN (ROJ)**

* Returns **all rows from right table**, matched rows from left table.

SELECT Students.Name, Department.DeptName

FROM Students

RIGHT OUTER JOIN Department

ON Students.DeptID = Department.DeptID;

**3. FULL OUTER JOIN (FOJ)**

* Returns **all rows from both tables**, unmatched rows will show NULL.

SELECT Students.Name, Department.DeptName

FROM Students

FULL OUTER JOIN Department

ON Students.DeptID = Department.DeptID;

**C. CROSS JOIN**

* Returns **Cartesian product** of two tables (every row of Table A with every row of Table B).
* **No condition required.**

SELECT Students.Name, Department.DeptName

FROM Students

CROSS JOIN Department;

* **Result:** Total rows = rows in Students × rows in Department.

**D. SELF JOIN**

* A table joins with **itself**.
* Useful for hierarchical or comparative queries.

SELECT A.Name AS Employee, B.Name AS Manager

FROM Employees A

LEFT JOIN Employees B

ON A.ManagerID = B.EmployeeID;

* **Explanation:** Shows each employee with their manager.

**E. NATURAL JOIN**

* Automatically joins tables **using columns with the same name**.
* No ON clause required.

SELECT Students.Name, Department.DeptName

FROM Students

NATURAL JOIN Department;

* **Caution:** Column names must match; otherwise, it can produce unexpected results.

**F. COMPARISON**

| **Join Type** | **Result** |
| --- | --- |
| INNER JOIN | Only matching rows |
| LEFT OUTER JOIN | All rows from left table + matching right table rows |
| RIGHT OUTER JOIN | All rows from right table + matching left table rows |
| FULL OUTER JOIN | All rows from both tables |
| CROSS JOIN | Cartesian product |
| SELF JOIN | Table joined with itself |
| NATURAL JOIN | Joins automatically on columns with the same name |

If you want, I can make a **diagram showing all joins visually** with tables and how rows are selected—super useful for memorization.

Do you want me to do that?

**You said:**

equi and conditional

**ChatGPT said:**

Ah! Now you’re asking about **Equi Join** and **Conditional Join** (sometimes also called Non-Equi Join). Let’s break it down clearly.

**1. Equi Join**

* A join that **matches rows based on equality (=) of specified columns**.
* Most common type of join.
* Can be **Inner, Outer, or Self** join.
* Usually written using ON clause with = operator.

**Example:**

SELECT Students.Name, Department.DeptName

FROM Students

INNER JOIN Department

ON Students.DeptID = Department.DeptID;

✅ Here, Students.DeptID = Department.DeptID is the **equality condition**, so it’s an **Equi Join**.

**Notes:**

* Can use **Aliases** for clarity:

SELECT S.Name, D.DeptName

FROM Students S

JOIN Department D

ON S.DeptID = D.DeptID;

* **Natural Join** is also a type of equi join because it automatically matches columns with the same name.

**2. Conditional Join (Non-Equi Join)**

* A join where the **condition is NOT based on equality** (<, >, <=, >=, !=, BETWEEN, etc.).
* Used for **ranges, inequalities, or special conditions**.

**Example: Joining students with scholarship brackets:**

SELECT S.Name, B.ScholarshipAmount

FROM Students S

JOIN Scholarship B

ON S.Score BETWEEN B.MinScore AND B.MaxScore;

✅ Here, the join is **conditional** (based on a range), not simple equality.

**Other examples:**

* ON A.Salary > B.MinSalary
* ON A.Age <= B.MaxAge

**Comparison Table:**

| **Type** | **Condition Example** | **Use Case** |
| --- | --- | --- |
| Equi Join | A.ID = B.ID | Match rows with exact value |
| Conditional Join | A.Salary > B.MinSalary | Match rows based on inequality or range |

If you want, I can make a **full table of all joins (Inner, Outer, Cross, Self, Natural, Equi, Conditional) with diagrams and examples** so it’s super easy to memorize.

Do you want me to do that?

**You said:**

yes

**ChatGPT said:**

Perfect! Let’s create a **complete, organized guide** of **all SQL joins** including **Inner, Outer, Cross, Self, Natural, Equi, and Conditional joins**, with **examples and diagrams** for easy understanding.

**SQL Joins Complete Guide**

We’ll use **two example tables**:

**Students Table**

| **StudentID** | **Name** | **DeptID** | **Score** |
| --- | --- | --- | --- |
| 1 | Alice | 10 | 85 |
| 2 | Bob | 20 | 72 |
| 3 | Charlie | 10 | 60 |
| 4 | David | NULL | 90 |

**Department Table**

| **DeptID** | **DeptName** |
| --- | --- |
| 10 | Computer Sci |
| 20 | Mathematics |
| 30 | Physics |

**Scholarship Table**

| **MinScore** | **MaxScore** | **ScholarshipAmount** |
| --- | --- | --- |
| 70 | 100 | 5000 |
| 50 | 69 | 3000 |

**1. INNER JOIN (Equi Join)**

* Returns rows where the **columns match exactly**.

SELECT S.Name, D.DeptName

FROM Students S

INNER JOIN Department D

ON S.DeptID = D.DeptID;

**Result:**

| **Name** | **DeptName** |
| --- | --- |
| Alice | Computer Sci |
| Bob | Mathematics |
| Charlie | Computer Sci |

**2. LEFT OUTER JOIN**

* Returns **all rows from left table**, matched right table rows or NULL.

SELECT S.Name, D.DeptName

FROM Students S

LEFT JOIN Department D

ON S.DeptID = D.DeptID;

**Result:**

| **Name** | **DeptName** |
| --- | --- |
| Alice | Computer Sci |
| Bob | Mathematics |
| Charlie | Computer Sci |
| David | NULL |

**3. RIGHT OUTER JOIN**

* Returns **all rows from right table**, matched left table rows or NULL.

SELECT S.Name, D.DeptName

FROM Students S

RIGHT JOIN Department D

ON S.DeptID = D.DeptID;

**Result:**

| **Name** | **DeptName** |
| --- | --- |
| Alice | Computer Sci |
| Charlie | Computer Sci |
| Bob | Mathematics |
| NULL | Physics |

**4. FULL OUTER JOIN**

* Returns **all rows from both tables**, unmatched rows are NULL.

SELECT S.Name, D.DeptName

FROM Students S

FULL OUTER JOIN Department D

ON S.DeptID = D.DeptID;

**Result:**

| **Name** | **DeptName** |
| --- | --- |
| Alice | Computer Sci |
| Charlie | Computer Sci |
| Bob | Mathematics |
| David | NULL |
| NULL | Physics |

**5. CROSS JOIN**

* Cartesian product: every row in Students × every row in Department.

SELECT S.Name, D.DeptName

FROM Students S

CROSS JOIN Department D;

**Result:** 4×3 = 12 rows

| **Name** | **DeptName** |
| --- | --- |
| Alice | Computer Sci |
| Alice | Mathematics |
| Alice | Physics |
| Bob | Computer Sci |
| … | … |

**6. SELF JOIN**

* A table joins **with itself**. Often used for hierarchy.

SELECT A.Name AS Student, B.Name AS Mentor

FROM Students A

LEFT JOIN Students B

ON A.DeptID = B.DeptID AND A.StudentID <> B.StudentID;

**Explanation:** Shows **students in same department as peers/mentors**.

**7. NATURAL JOIN**

* Automatically joins on **columns with the same name** (DeptID in our case).

SELECT Name, DeptName

FROM Students

NATURAL JOIN Department;

**Result:** Same as inner join example.

**8. EQUI JOIN**

* Join **based on equality**.
* All INNER JOINs or NATURAL JOINs using = are **equi joins**.

SELECT S.Name, D.DeptName

FROM Students S

JOIN Department D

ON S.DeptID = D.DeptID;

**9. CONDITIONAL JOIN (Non-Equi Join)**

* Join **based on inequality or range**.
* Example: Assign scholarships based on score.

SELECT S.Name, S.Score, B.ScholarshipAmount

FROM Students S

JOIN Scholarship B

ON S.Score BETWEEN B.MinScore AND B.MaxScore;

**Result:**

| **Name** | **Score** | **ScholarshipAmount** |
| --- | --- | --- |
| Alice | 85 | 5000 |
| Bob | 72 | 5000 |
| Charlie | 60 | 3000 |
| David | 90 | 5000 |

**Visual Summary of Joins**

| **Join Type** | **Example Condition** | **Rows Returned** |
| --- | --- | --- |
| INNER JOIN | A.ID = B.ID | Only matching rows |
| LEFT OUTER JOIN | A.ID = B.ID | All left rows + matched right rows |
| RIGHT OUTER JOIN | A.ID = B.ID | All right rows + matched left rows |
| FULL OUTER JOIN | A.ID = B.ID | All rows from both tables |
| CROSS JOIN | No condition | Cartesian product |
| SELF JOIN | A.ID = B.ID (same table) | Table joined with itself |
| NATURAL JOIN | Automatically on same column | Only matching rows |
| EQUI JOIN | A.col = B.col | Only matching rows (uses =) |
| CONDITIONAL JOIN | A.col BETWEEN B.min AND B.max | Rows based on condition (inequality, ranges) |