1. Difference between INNER and LEFT JOIN

INNER JOIN

- Returns **only matching rows** from both tables.
- Rows with no match are **excluded**.

LEFT JOIN

- Returns **all rows from the left table**, even if no match exists.
- Unmatched right-table columns appear as **NULL**.

---

2. What is a FULL OUTER JOIN?

A **FULL OUTER JOIN** returns:

- All rows from the **left table**
- All rows from the **right table**
- Matches where available
- **NULL** where no match exists

It is basically:
LEFT JOIN + RIGHT JOIN combined.

(If DB does not support FULL JOIN, use LEFT JOIN UNION RIGHT JOIN.)

---

3. Can joins be nested?

Yes.
Joins can be nested (chained) inside each other, for example:

SELECT *

FROM A

JOIN B ON A.id = B.id

JOIN C ON B.id = C.id;

Nested joins are common when working with 3 or more tables.

---

4. How to join more than 2 tables?

By chaining multiple JOINs:

SELECT *

FROM Table1

JOIN Table2 ON ...

JOIN Table3 ON ...

JOIN Table4 ON ...;

SQL allows joining **any number of tables** as long as proper conditions exist.

---

5. What is a CROSS JOIN?

A **CROSS JOIN** produces the **Cartesian product** of two tables:

- Every row of table A is paired with every row of table B.
- If A has 3 rows and B has 4 rows → output = **12 rows**

It does **not** use ON condition.

---

6. What is a NATURAL JOIN?

A **NATURAL JOIN** automatically joins tables using **all columns with the same name**.

Example: If both tables have emp_id column → it joins on that column.

⚠ **Not recommended** in real projects because:

- Automatic matching can cause unexpected results.
- Renaming of columns breaks queries.

---

7. Can you join tables without a foreign key?

Yes.
SQL does not require foreign keys to perform JOINs.

You can join using **any columns**, even if no relationship is defined:

SELECT *

FROM A

JOIN B

ON A.color = B.color;

Foreign keys help maintain data integrity, but joins work without them.

---

8. What is a self-join?

A **self-join** is when a table is joined with **itself**.

Common use: hierarchical data
(e.g., employee and their manager in the same table)

SELECT e.emp_name, m.emp_name AS manager

FROM Employees e

JOIN Employees m

ON e.manager_id = m.emp_id;

---

9. What causes Cartesian product?

A **Cartesian product** happens when:

- You use a **CROSS JOIN**, or
- You use a JOIN **without an ON condition**, or
- Your join condition is incorrect or missing.

This results in **every row** combining with **every other row**.

---

10. How to optimize joins?

Ways to optimize joins:

1. **Use indexes** on join columns (very important).
2. **Join on primary key or indexed fields**.
3. Select only required columns (SELECT specific_columns).
4. Use proper **JOIN conditions**.
5. Avoid unnecessary **nested JOINs**.
6. Use **EXPLAIN** query to understand performance.
7. Avoid joining extremely large tables without filters.
8. Use **WHERE** before JOIN if possible to reduce rows.