**Task 5: SQL Joins (Inner, Left, Right, Full)**

Objective: Learn to combine data from multiple tables

Tools :DB Browser for SQLite / MySQL Workbench

Deliverables: SQL queries using all join types

Hints/Mini Guide:

1.Create two related tables (e.g., Customers, Orders)

2.Use INNER, LEFT, RIGHT, FULL JOIN

Outcome: Mastery of merging data Interview Questions

**Creating Related Tables**

We’ll create **CUSTOMERS** and **ORDERS** tables in Oracle:

-- Create CUSTOMERS table

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(50),

city VARCHAR2(50)

);

-- Create ORDERS table

CREATE TABLE orders (

order\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

order\_date DATE,

amount NUMBER(10,2),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

-- Insert sample data into CUSTOMERS

INSERT INTO customers VALUES (1, 'Ravi Kumar', 'Hyderabad');

INSERT INTO customers VALUES (2, 'Anita Sharma', 'Mumbai');

INSERT INTO customers VALUES (3, 'John Paul', 'Delhi');

INSERT INTO customers VALUES (4, 'Priya Singh', 'Chennai');

-- Insert sample data into ORDERS

INSERT INTO orders VALUES (101, 1, DATE '2025-08-01', 5000);

INSERT INTO orders VALUES (102, 1, DATE '2025-08-05', 2000);

INSERT INTO orders VALUES (103, 2, DATE '2025-08-07', 3000);

INSERT INTO orders VALUES (104, 5, DATE '2025-08-09', 1500); -- Orphan record (no matching customer)

**2. INNER JOIN (Default JOIN in Oracle)**

**Definition:** Returns only rows with matching values in both tables.  
**Rule:** If there’s no match, the row is excluded.

**Explanation:**

* Only customers **with orders** will appear.
* Oracle **INNER JOIN** is ANSI SQL standard.
* The older Oracle join syntax was:

SELECT c.customer\_id, c.customer\_name, o.order\_id, o.amount

FROM customers c, orders o

WHERE c.customer\_id = o.customer\_id;

**3. LEFT OUTER JOIN**

**Definition:** Returns all rows from the left table (CUSTOMERS), and matching rows from the right table (ORDERS). If there’s no match, right side columns are NULL.

**Rules & Notes:**

* All customers will be listed, even if they don’t have orders.
* Orders will show NULL for customers without orders.
* Oracle syntax:

SELECT c.customer\_id, c.customer\_name, o.order\_id, o.amount

FROM customers c, orders o

WHERE c.customer\_id = o.customer\_id(+);

The (+) is **right-side outer join** notation.

**4. RIGHT OUTER JOIN**

**Definition:** Returns all rows from the right table (ORDERS), and matching rows from the left table (CUSTOMERS). If there is no match, left side columns are NULL.

**Rules & Notes:**

* All orders will be shown, even if the customer ID doesn’t exist in CUSTOMERS (e.g., orphan order with customer\_id = 5).
* Oracle syntax:

SELECT c.customer\_id, c.customer\_name, o.order\_id, o.amount

FROM customers c, orders o

WHERE c.customer\_id(+) = o.customer\_id;

**5. FULL OUTER JOIN**

**Definition:** Returns all rows from both tables, with NULLs where there’s no match.

**Rules & Notes:**

* Not available in old Oracle (+) syntax — only ANSI JOIN syntax supports it.
* Shows:
  1. Matching customers with orders.
  2. Customers without orders.
  3. Orders without customers.
* Oracle syntax:

SELECT c.customer\_id, c.customer\_name, o.order\_id, o.amount

FROM customers c, orders o

WHERE c.customer\_id = o.customer\_id(+)

Union

SELECT c.customer\_id, c.customer\_name, o.order\_id, o.amount

FROM customers c, orders o

WHERE c.customer\_id(+) = o.customer\_id;

**6. Extra Oracle-Specific Points & Rules**

* **Order of tables matters** for LEFT and RIGHT joins.
* **FULL JOIN** is expensive in large datasets because it processes both unmatched sets.
* Oracle **supports ANSI SQL joins** (preferred for clarity) and **old Oracle join syntax** (+).
* Avoid mixing ANSI and (+) join syntax in the same query — Oracle will throw ORA-01468: a predicate may reference only one outer-joined table.
* When filtering:
  + For **INNER JOIN**, put conditions in ON or WHERE — same effect.
  + For **OUTER JOIN**, filtering in WHERE can turn it into an INNER JOIN accidentally, so use ON for join conditions and WHERE for additional filtering that shouldn’t remove null-matched rows.

✅ **Outcome:**  
With these examples, you’ll **master joining data in Oracle**, understand **ANSI vs Oracle-specific syntax**, and know **rules & pitfalls**.

**1. Difference between INNER and LEFT JOIN**

INNER JOIN

* Returns only matching rows from both tables based on the join condition.
* If there’s no match, the row is excluded.

LEFT JOIN

* Returns all rows from the left table and matches from the right table.
* If no match, right table columns will be NULL.

**Key Difference**: LEFT JOIN preserves all rows from the left table; INNER JOIN doesn’t.

**2. What is a FULL OUTER JOIN?**

* Returns all rows from both tables, whether they match or not.
* Missing values are filled with NULL.
* For Full join, we do union for left join and right join.

Oracle note: FULL OUTER JOIN is supported directly; you cannot use (+) syntax for it.

**3. Can joins be nested?**

Yes. You can join results of one join with another join, even in subqueries.

Example:

SELECT x.BookID, x.Title, a.AuthorName

FROM (

SELECT b.BookID, b.Title, ba.AuthorID

FROM Books b, BookAuthors ba

Where b.BookID = ba.BookID

) x, Authors a

Where x.AuthorID = a.AuthorID;

* Nested joins are common in complex reports.

**4. How to join more than 2 tables?**

* Add multiple JOIN clauses sequentially.
* Join order matters for performance, not for correctness (unless OUTER JOIN is involved).

SELECT b.Title, a.AuthorName, c.CategoryName

FROM Books b, BookAuthors ba, Authors a, Categories c

Where b.BookID = ba.BookID

And ba.AuthorID = a.AuthorID

And b.CategoryID = c.CategoryID;

**5. What is a CROSS JOIN?**

* Produces Cartesian product: every row from one table with every row from another.
* No ON condition is used.

SELECT a.AuthorName, b.Title

FROM Authors a

CROSS JOIN Books b;

* Avoid unless specifically needed — can explode result size.

**6. What is a NATURAL JOIN?**

* Automatically joins tables based on same column names and compatible data types.
* Avoid unless the column names are guaranteed consistent — can cause unexpected joins.

SELECT \*

FROM Books

NATURAL JOIN Categories;

* Dangerous in large schemas — explicit joins are safer.

**7. Can you join tables without foreign key?**

Yes. The join condition does not require a declared foreign key, just matching logic.

Example: Joining by Title (not recommended for production):

SELECT \*

FROM Books b, OldBooks ob

Where b.Title = ob.Title;

**8. What is a SELF JOIN?**

* Joining a table to itself, usually using aliases.
* Useful for hierarchical or related data in the same table.

Example: Authors recommending other authors

SELECT a1.AuthorName AS Author,

a2.AuthorName AS Recommends

FROM Authors a1, Authors a2

Where a1.RecommendedAuthorID = a2.AuthorID;

**9. What causes Cartesian product?**

* Missing join condition in INNER JOIN.
* Using CROSS JOIN intentionally.
* Accidentally writing multiple tables in FROM clause without WHERE filter.

Example of mistake:

SELECT \*

FROM Authors a, Books b; -- No WHERE => full Cartesian product

**10. How to optimize joins in Oracle?**

Best Practices:

1. Indexes
   * Ensure columns in ON condition are indexed.
   * Composite index if joining on multiple columns.
2. Join on numeric keys
   * Numbers compare faster than strings.
3. Filter early
   * Apply WHERE conditions before joins when possible.
4. Avoid functions on join columns
   * e.g., UPPER(col1) = UPPER(col2) disables index usage.
5. Use appropriate join type
   * INNER JOIN when outer joins are not needed (less overhead).
6. Check execution plan

**EXPLAIN PLAN FOR <your query>;**

**SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);**