**Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology**

**(Deemed to be University Estd. u/s 3 of UGC Act, 1956)**

**School of Computing**

**B.Tech. – Computer Science and Engineering**

**VTR UGE2021- (CBCS)**

Academic Year: 2025–2026

SUMMER SEMESTER - SS2526

Course Code : 10211CS207

Course Name: Database Management Systems

Slot No : S2L5

DBMS TASK - 5 REPORT

**Title:** Writing Join Queries, Equivalent, and/or Recursive Queries

**Submitted by:**

|  |  |  |
| --- | --- | --- |
| **VTUNO** | **REGISTER NUMBER** | **STUDENT NAME** |
| VTU27661 | 24UECS1159 | C Venugopal reddy |

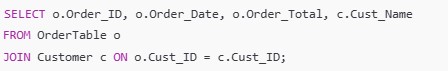
**Task 5: Writing Join Queries, Equivalent, and/or Recursive Queries**

**Case Study**: Online Food Ordering System

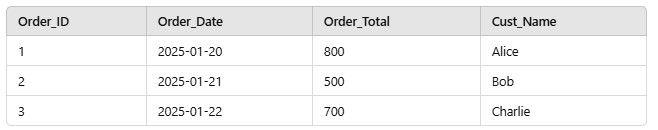
**Objective**: To perform advanced query processing and test its heuristics by designing optimized complex queries and their equivalent forms, including recursive queries, for the Online Food Ordering System.

# Join Queries

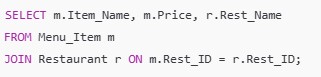
**Query 1: Retrieve all orders along with the corresponding customer's name.**



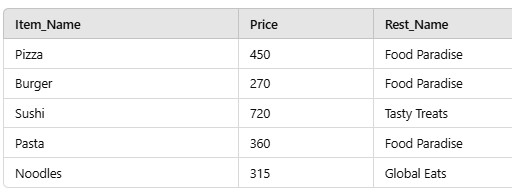
**Expected Output:**



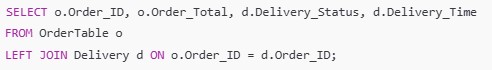
**Query 2: Retrieve all menu items along with the restaurant name that offers them.**



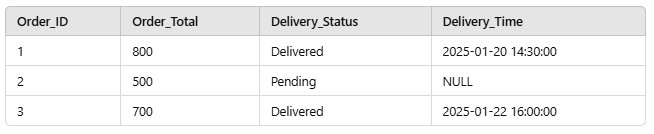
**Expected Output:**



**Query 3: Retrieve all orders and their delivery status.**



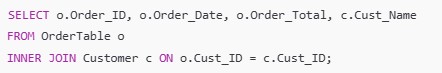
**Expected Output:**



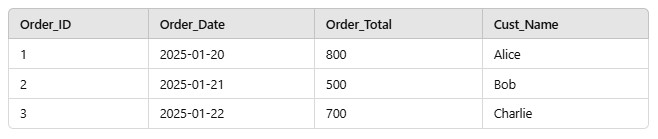
# INNER JOIN

An **INNER JOIN** retrieves records that have matching values in both tables.

**Query: Retrieve all orders along with their customer names.**



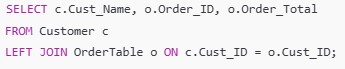
**Expected Output:**



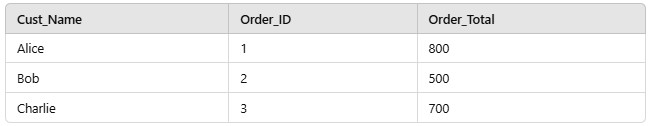
# LEFT OUTER JOIN

A **LEFT OUTER JOIN** retrieves all records from the left table and the matched records from the right table. If no match is found, NULL is returned for columns from the right table.

**Query: Retrieve all customers, even those who haven’t placed any orders.**



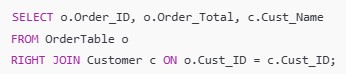
**Expected Output:**



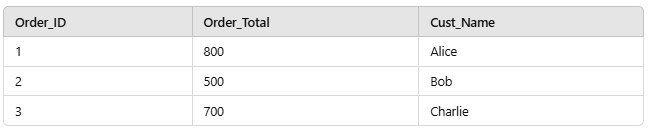
# RIGHT OUTER JOIN

A **RIGHT OUTER JOIN** retrieves all records from the right table and the matched records from the left table. If no match is found, NULL is returned for columns from the left table.

**Query: Retrieve all orders and the names of customers who placed them. Include orders even if the customer details are missing.**



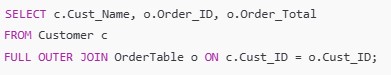
**Expected Output:**



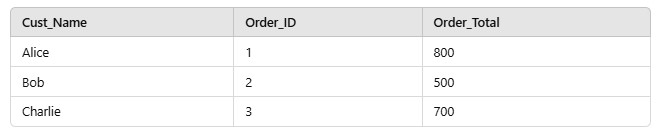
# FULL OUTER JOIN

A **FULL OUTER JOIN** retrieves all records from both tables. If no match is found, NULL is returned for unmatched rows from either table.

**Query: Retrieve all customers and all orders, even if there is no match.**



**Expected Output:**

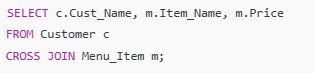


*Note: Oracle doesn’t support FULL OUTER JOIN directly. Use UNION of LEFT JOIN and RIGHT JOIN.*

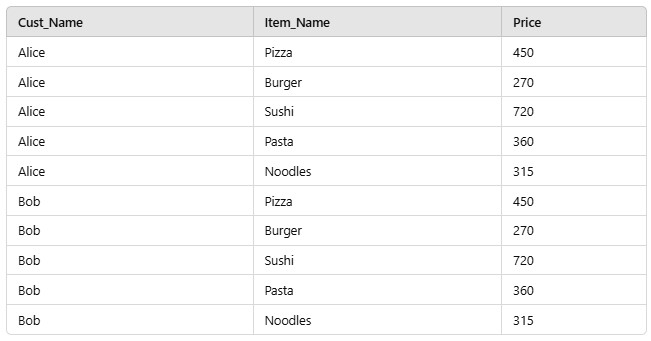
# CROSS JOIN

A **CROSS JOIN** returns the Cartesian product of the two tables. Every row from the first table is combined with every row from the second table.

**Query: Retrieve all possible combinations of customers and menu items.**



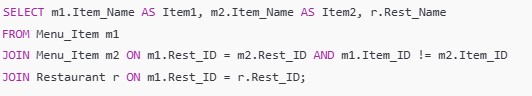
**Expected Output:**



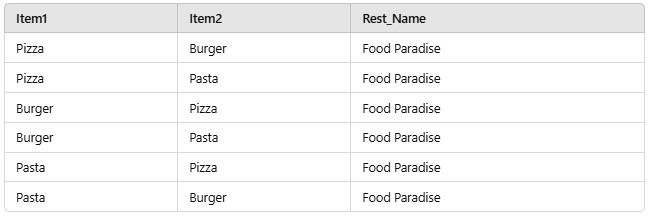
# SELF JOIN

A **SELF JOIN** joins a table with itself. It is useful for hierarchical or comparison data.

**Query: Retrieve all menu items that belong to the same restaurant as another item.**



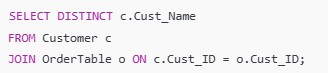
**Expected Output:**



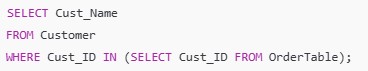
# Equivalent Queries

**Query 1: Retrieve all customers who placed orders using a join (equivalent to a subquery).**

**Using Join:**

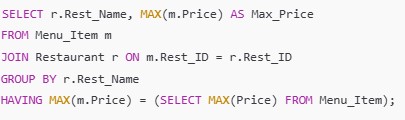


**Equivalent Subquery:**

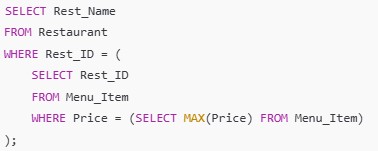


**Query 2: Retrieve the restaurant offering the most expensive menu item.**

**Using Join:**



**Equivalent Subquery:**

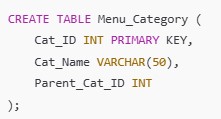


# Recursive Queries

Oracle SQL supports recursion using the **WITH** clause for hierarchical data.

**Query 1: Generate a recursive query to find all ancestors of a given category in a hypothetical "Menu Category" table.**

Assume we have a table:



Sample Data:

INSERT INTO Menu\_Category (Cat\_ID, Cat\_Name, Parent\_Cat\_ID) VALUES (1, 'Food', NULL);

INSERT INTO Menu\_Category (Cat\_ID, Cat\_Name, Parent\_Cat\_ID) VALUES (2, 'Italian',

1);

INSERT INTO Menu\_Category (Cat\_ID, Cat\_Name, Parent\_Cat\_ID) VALUES (3, 'Chinese', 1);

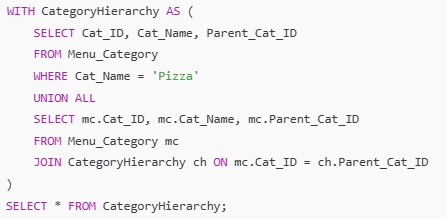
INSERT INTO Menu\_Category (Cat\_ID, Cat\_Name, Parent\_Cat\_ID) VALUES (4, 'Pizza',

2);

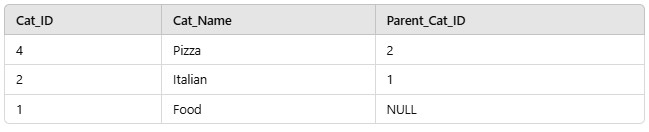
INSERT INTO Menu\_Category (Cat\_ID, Cat\_Name, Parent\_Cat\_ID) VALUES (5, 'Pasta',

2);

**Recursive Query:**

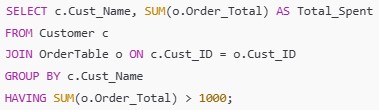


**Expected Output for 'Pizza':**

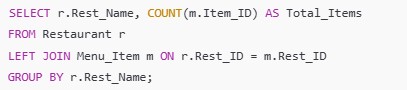


# Optimizing Complex Queries

**Query 1: Find customers who placed orders totaling more than 1000 across all their orders.**



**Query 2: Retrieve all restaurants and their total number of menu items using a join.**



**RESULT:**Thus the task has been executed and verified successfully.