

# DAY 6 – SQL JOINS & MULTI-TABLE ANALYSIS

**Dataset:** Superstore Sales Dataset

**Tool Used:** MySQL 8.0

**Analysis Type:** SQL Joins and Relational Analysis

## 1. Objective of Day 6

The objective of Day 6 was to understand and apply SQL joins to combine data from multiple tables, analyze relationships between datasets, and perform multi-table business analysis. This day focused on both **conceptual understanding** and **practical implementation** of joins.

## 2. Introduction to SQL Joins

In real-world databases, data is rarely stored in a single table. Instead, it is distributed across multiple related tables to reduce redundancy and improve scalability. **SQL joins** are used to combine rows from two or more tables based on a related column.

SQL joins allow analysts to:

- Combine transactional data with reference data
- Perform multi-table analysis
- Validate data relationships
- Build business-ready reports and dashboards

## 3. Why SQL Joins Are Important

SQL joins are essential because:

- Most real-world databases follow a **relational model**
- Business data is stored across **fact tables** and **dimension tables**
- Analytical queries almost always require joins
- Joins are a **core topic in SQL interviews**

Without joins, it is not possible to perform meaningful analysis on structured databases.

## 4. Types of SQL Joins

### 4.1 INNER JOIN

**Definition:**

An INNER JOIN returns only the rows that have matching values in both tables.

**Syntax:**

```
SELECT *
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

### Key Points:

- Returns only matching records
- Non-matching rows are excluded
- Most commonly used join type

### Business Use Case:

- Fetch orders that have valid category information
- Generate clean reports with confirmed relationships

## 4.2 LEFT JOIN (LEFT OUTER JOIN)

### Definition:

A LEFT JOIN returns all rows from the left table and matching rows from the right table. If no match exists, NULL values are returned for the right table.

### Syntax:

```
SELECT *
FROM table1
LEFT JOIN table2
ON table1.column = table2.column;
```

### Key Points:

- Preserves all records from the left table
- Displays NULL for missing matches
- Useful for data completeness checks

### Business Use Case:

- Identify missing reference data
- Validate data quality
- Detect unmatched records

## 4.3 RIGHT JOIN (RIGHT OUTER JOIN)

### Definition:

A RIGHT JOIN returns all rows from the right table and matching rows from the left table.

### Syntax:

```
SELECT *
FROM table1
RIGHT JOIN table2
ON table1.column = table2.column;
```

### **Key Points:**

- Less commonly used
- Can usually be replaced by a LEFT JOIN by switching table order

### **Business Use Case:**

- Useful when reference table records must be preserved

## **4.4 FULL OUTER JOIN**

### **Definition:**

A FULL OUTER JOIN returns all rows when there is a match in either table.

### **Key Points:**

- Includes both matching and non-matching rows
- Not directly supported in MySQL
- Can be simulated using LEFT JOIN + UNION + RIGHT JOIN

### **Business Use Case:**

- Dataset comparison
- Identifying mismatches across systems

## **4.5 SELF JOIN**

### **Definition:**

A SELF JOIN is a join where a table is joined with itself.

### **Syntax:**

```
SELECT a.column, b.column  
FROM table a  
JOIN table b  
ON a.common_column = b.common_column;
```

### **Key Points:**

- Requires table aliases
- Used for comparison or hierarchical analysis

### **Business Use Case:**

- Comparing product performance within the same category
- Employee-manager relationship analysis

## **4.6 CROSS JOIN**

### **Definition:**

A CROSS JOIN produces the Cartesian product of two tables.

### **Syntax:**

```
SELECT *
FROM table1
CROSS JOIN table2;
```

### Key Points:

- Returns all possible combinations
- Can generate large result sets
- Should be used cautiously

### Business Use Case:

- Scenario analysis
- Simulation and modeling
- 

## SQL Joins Applied in Analysis

---

### 5.1 INNER JOIN – Combine Orders with Categories

```
SELECT
    o.order_id,
    o.sales,
    o.profit,
    c.category_name
FROM orders o
INNER JOIN categories c
ON o.category = c.category_name;
```

	order_id	sales	profit	category_name
▶	CA-2016-152156	261.96	41.91	Furniture
	CA-2016-152156	731.94	219.58	Furniture
	CA-2016-138688	14.62	6.87	Office Supplies
	US-2015-108966	957.58	-383.03	Furniture
	US-2015-108966	22.37	2.52	Office Supplies
	CA-2014-115812	48.86	14.17	Furniture
	CA-2014-115812	7.28	1.97	Office Supplies
	CA-2014-115812	907.15	90.72	Technology
	CA-2014-115812	18.50	5.78	Office Supplies
	CA-2014-115812	114.90	34.47	Office Supplies
	CA-2014-115812	1706.18	85.31	Furniture
	CA-2014-115812	911.42	68.36	Technology
	CA-2017-114412	15.55	5.44	Office Supplies
	CA-2016-161389	407.98	132.59	Office Supplies
	US-2015-118983	68.81	-123.86	Office Supplies
	US-2015-118983	2.54	-3.82	Office Supplies
	CA-2014-105893	665.88	13.32	Office Supplies
	CA-2014-167164	55.50	9.99	Office Supplies
	CA-2014-143336	8.56	2.48	Office Supplies
	CA-2014-143336	213.48	16.01	Technology
	CA-2014-143336	22.72	7.38	Office Supplies

### 5.2 LEFT JOIN – Preserve All Orders

```
SELECT
    o.order_id,
    o.sales,
    o.profit,
    s.sub_category_name
FROM orders o
LEFT JOIN sub_categories s
ON o.sub_category = s.sub_category_name;
```

	order_id	sales	profit	sub_category_name
▶	CA-2016-152156	261.96	41.91	Bookcases
	CA-2016-152156	731.94	219.58	Chairs
	CA-2016-138688	14.62	6.87	Labels
	US-2015-108966	957.58	-383.03	Tables
	US-2015-108966	22.37	2.52	Storage
	CA-2014-115812	48.86	14.17	Furnishings
	CA-2014-115812	7.28	1.97	Art
	CA-2014-115812	907.15	90.72	Phones
	CA-2014-115812	18.50	5.78	Binders
	CA-2014-115812	114.90	34.47	Appliances
	CA-2014-115812	1706.18	85.31	Tables
	CA-2014-115812	911.42	68.36	Phones
	CA-2017-114412	15.55	5.44	Paper
	CA-2016-161389	407.98	132.59	Binders
	US-2015-118983	68.81	-123.86	Appliances
	US-2015-118983	2.54	-3.82	Binders
	CA-2014-105893	665.88	13.32	Storage
	CA-2014-167164	55.50	9.99	Storage

## 5.3 Identify Missing Matches Using LEFT JOIN

```
SELECT
    o.order_id,
    o.sub_category
FROM orders o
LEFT JOIN sub_categories s
ON o.sub_category = s.sub_category_name
WHERE s.sub_category_name IS NULL;
```

order_id	sub_category
----------	--------------

## 5.4 JOIN with Aggregation

```
SELECT
    c.category_name,
    ROUND(SUM(o.sales),2) AS total_sales,
    ROUND(SUM(o.profit),2) AS total_profit
FROM orders o
JOIN categories c
ON o.category = c.category_name
GROUP BY c.category_name;
```

	category_name	total_sales	total_profit
▶	Furniture	741999.98	18451.25
	Office Supplies	719046.99	122490.88
	Technology	836154.10	145455.66

## 5.5 SELF JOIN (Benchmark Comparison Using Subquery)

```
SELECT
    o.sub_category,
    ROUND(SUM(o.profit),2) AS product_profit,
    cat.avg_profit
FROM orders o
JOIN (
    SELECT
        category,
        ROUND(AVG(profit),2) AS avg_profit
    FROM orders
    GROUP BY category
) cat
ON o.category = cat.category
GROUP BY o.sub_category, cat.avg_profit;
```

	sub_category	product_profit	avg_profit
▶	Bookcases	-3472.56	8.70
	Chairs	26590.15	8.70
	Labels	5546.18	20.33
	Tables	-17725.59	8.70
	Storage	21279.05	20.33
	Furnishings	13059.25	8.70
	Art	6527.96	20.33
	Phones	44516.25	78.75
	Binders	30221.64	20.33
	Appliances	18138.07	20.33
	Paper	34053.34	20.33
	Accessories	41936.78	78.75
	Envelopes	6964.10	20.33
	Fasteners	949.53	20.33
	Supplies	-1188.99	20.33
	Machines	3384.73	78.75
	Copiers	55617.90	78.75

## 5.6 JOIN + WINDOW FUNCTION – Rank Products Within Categories

```
WITH profit_data AS (
    SELECT
        c.category_name,
        o.sub_category,
        ROUND(SUM(o.profit),2) AS total_profit
    FROM orders o
    JOIN categories c
        ON o.category = c.category_name
    GROUP BY c.category_name, o.sub_category
)
SELECT
    category_name,
    sub_category,
    total_profit,
    DENSE_RANK() OVER (
        PARTITION BY category_name
        ORDER BY total_profit DESC
    ) AS category_rank
FROM profit_data;
```

	category_name	sub_category	total_profit	category_rank
▶	Furniture	Chairs	26590.15	1
	Furniture	Furnishings	13059.25	2
	Furniture	Bookcases	-3472.56	3
	Furniture	Tables	-17725.59	4
	Office Supplies	Paper	34053.34	1
	Office Supplies	Binders	30221.64	2
	Office Supplies	Storage	21279.05	3
	Office Supplies	Appliances	18138.07	4
	Office Supplies	Envelopes	6964.10	5
	Office Supplies	Art	6527.96	6
	Office Supplies	Labels	5546.18	7
	Office Supplies	Fasteners	949.53	8
	Office Supplies	Supplies	-1188.99	9
	Technology	Copiers	55617.90	1
	Technology	Phones	44516.25	2
	Technology	Accessories	41936.78	3
	Technology	Machines	3384.73	4