Mastering SQL: Analytical Patterns for LeetCode and Beyond

A Comprehensive Guide to SQL Problem-Solving Patterns

Table of Contents

---

1. Introduction

This guide provides a structured approach to solving SQL problems, moving beyond simple syntax and focusing on the underlying analytical patterns. By understanding these patterns, you can tackle a wide range of SQL challenges more effectively. Remember, SQL isn’t just about writing queries; it’s about translating business problems into logical steps.

2. The Core Process

Core Analytical Task

- Before writing any SQL, understand exactly what the problem asks you to analyze or find
- Identify if it’s a straightforward question or needs breaking down
- Look for keywords and focus on the intent

Data Relationships

- Identify relevant tables and their relationships
- Determine necessary joins
- Identify unique identifiers linking tables
- Understand the underlying data model

SQL Techniques

- Choose specific techniques based on analytical intent
- Consider: aggregations, filtering, self-joins
- Match the right tool to the analytical task

CTEs For Structure

- Consider using Common Table Expressions (CTEs)
- Break down queries into logical, manageable parts
- Improve query structure and readability
- Consider CTEs even for seemingly simple queries

Execution Plan

- Develop logical progression for problem-solving
- Start with a plan before coding
- Essential for complex problems

3. Specific Analytical Patterns and Techniques

A. Aggregation and Grouping

Analytical Intent: To summarize data over groups

Techniques:

- GROUP BY
- COUNT(), SUM(), AVG(), MIN(), MAX()
- HAVING

Example (184 - Department Highest Salary):

```
WITH MaxSalaries AS (
  SELECT DepartmentId, MAX(COALESCE(Salary, 0)) AS MaxSalary
  FROM Employee
  GROUP BY DepartmentId
)
SELECT d.Name AS Department,
       e.Name AS Employee,
       e.Salary
FROM Employee e
JOIN MaxSalaries ms
  ON e.DepartmentId = ms.DepartmentId
  AND COALESCE(e.Salary, 0) = ms.MaxSalary
JOIN Department d ON e.DepartmentId = d.Id;
```

B. Filtering and Selection

Analytical Intent: Retrieve specific subsets based on conditions

Techniques:

- WHERE, AND, OR, NOT
- IN, NOT IN, BETWEEN, LIKE

Example (183 - Customers Who Never Order):

```
WITH OrderedCustomers AS (
  SELECT CustomerId FROM Orders
)
SELECT Name
FROM Customers
WHERE Id NOT IN (
  SELECT COALESCE(CustomerId, 0)
  FROM OrderedCustomers
);
```

C. Self-Joins

Analytical Intent: Compare rows within a single table

Example (180 - Consecutive Numbers):

```
WITH LaggedLogs AS (
  SELECT
    Id,
    Num,
    LAG(Num, 1, NULL) OVER (ORDER BY Id) AS prev_num1,
    LAG(Num, 2, NULL) OVER (ORDER BY Id) AS prev_num2
  FROM Logs
)
SELECT DISTINCT Num
FROM LaggedLogs
WHERE Num = prev_num1
  AND Num = prev_num2
  AND Num IS NOT NULL;
```

D. Ranking and Ordering

Analytical Intent: Find relative position of records

Example (176 - Second Highest Salary):

```
WITH RankedSalaries AS (
  SELECT
    Salary,
    DENSE_RANK() OVER (
      ORDER BY COALESCE(Salary, 0) DESC
    ) as rank_num
  FROM Employee
)
SELECT
  CASE
    WHEN (SELECT COUNT(*) FROM RankedSalaries) < 2 THEN NULL
    ELSE (SELECT Salary FROM RankedSalaries WHERE rank_num = 2)
  END AS SecondHighestSalary
FROM (SELECT 1 as dummy_column) as dummy_table;
```

4. Common LeetCode SQL Problems by Pattern

Pattern-Based Quick Reference

Pattern & Problem	Difficulty	Key Concepts	Common Pitfalls
Aggregation			
184 - Dept Highest Salary	Medium	GROUP BY, MAX	NULL handling
185 - Dept Top 3 Salaries	Hard	Window funcs	Duplicates
262 - Trips and Users	Hard	Multi-join	Date filtering
Filtering			
183 - Customers No Orders	Easy	NOT IN	NULL in subquery
196 - Delete Duplicates	Easy	Self-JOIN	Row deletion
1251 - Avg Selling Price	Easy	JOIN	Date ranges
Self-Joins			
180 - Consecutive Numbers	Medium	LAG	Row sequence
197 - Rising Temperature	Easy	Self-JOIN	Date compare
1270 - Manager Hierarchy	Medium	Multi-join	Recursion
Ranking			
176 - Second High Salary	Medium	DENSE_RANK	NULL result
177 - Nth High Salary	Medium	ROW_NUMBER	Variable N
178 - Rank Scores	Medium	DENSE_RANK	Tie handling

5. Tips and Practice Recommendations

Pattern Recognition Tips

1. **Aggregation Problems**
    - Look for: "highest," "average," "total"
    - Consider NULL handling in aggregations
    - Watch for grouping requirements
2. **Filtering Problems**
    - Keywords: "never," "not in," "exclude"
    - Consider date range conditions
    - Watch for multiple conditions
3. **Self-Join Problems**
    - Look for consecutive values
    - Consider hierarchical relationships
    - Watch for row comparisons
4. **Ranking Problems**
    - Keywords: "nth highest," "top k"
    - Consider tie handling
    - Watch for NULL values

Practice Strategy

1. **Start Simple**
    - Begin with easy problems in each pattern
    - Master basic patterns before combinations
    - Practice NULL handling consistently
2. **Build Complexity**

- Move to medium difficulty
- Combine multiple patterns
- Focus on performance optimization

3. **Review and Reflect**

- Document common mistakes
- Create pattern templates
- Build a personal problem-solving framework

Common Mistakes to Avoid

1. **Technical Mistakes**

- Forgetting NULL handling
- Incorrect join conditions
- Missing edge cases

2. **Logical Mistakes**

- Misinterpreting requirements
- Overlooking business rules
- Incorrect aggregation logic

3. **Performance Mistakes**

- Unnecessary subqueries
- Inefficient joins
- Missing indexes consideration

End of Guide