SQL Aggregate Functions & GROUP BY - Interview Revision Notes

🎯 5 Core Aggregate Functions (Memorize!)

1. COUNT() - Counting Rows

What it does: Counts the number of rows

```
sql

-- Count all orders
SELECT COUNT(*) FROM orders;
-- Result: 150

-- Count non-null user_ids
SELECT COUNT(user_id) FROM orders;
```

Interview Tip: Always use `COUNT(*)` to count all rows, `COUNT(column)` excludes NULLs!

---

2. SUM() - Adding Numbers

What it does: Adds up all values in a column

```
sql

-- Total revenue
SELECT SUM(price) FROM sales;
-- Result: 45,890.50
```

Real Example: "What's our total revenue this month?" → Use SUM()

---

3. AVG() - Finding Average

What it does: Calculates the mean of values

```
sql

-- Average order value
SELECT AVG(order_amount) FROM orders;
-- Result: 127.45
```

Interview Tip: AVG() ignores NULL values automatically!

---

4. MIN() - Smallest Value

What it does: Finds the minimum value

```sql
-- Cheapest product
SELECT MIN(price) FROM products;
-- Result: 9.99
```

---

5. MAX() - Largest Value

What it does: Finds the maximum value

```sql
-- Most expensive product
SELECT MAX(price) FROM products;
-- Result: 999.99
```

📊 GROUP BY - The Game Changer

What is GROUP BY?

Think of it as: Excel's Pivot Table in SQL **What it does:** Splits data into groups, then aggregates each group separately

Basic Syntax Pattern

```sql
SELECT
    category,          -- What to group by
    SUM(sales)         -- What to calculate
FROM products
GROUP BY category;     -- Must match non-aggregated columns
```

Real Example - Sales by Category

```sql
```

```sql
-- WITHOUT GROUP BY (wrong for this question)
SELECT SUM(sales) FROM products;
-- Result: 5000 (total only)

-- WITH GROUP BY (correct!)
SELECT
    category,
    SUM(sales) AS total_sales
FROM products
GROUP BY category;

-- Result:
-- Electronics | 2000
-- Appliances  | 1800
-- Furniture   | 1200
```

🔥 GROUP BY Multiple Columns

sql

```sql
-- Sales by category AND year
SELECT
    category,
    EXTRACT(YEAR FROM date) AS year,
    SUM(sales)
FROM products
GROUP BY category, year;  -- or GROUP BY 1, 2 (shorthand)

-- Result:
-- Electronics | 2023 | 1200
-- Electronics | 2024 | 800
-- Appliances  | 2023 | 900
-- Appliances  | 2024 | 900
```

Interview Tip: Every column in SELECT that's NOT aggregated MUST be in GROUP BY!

⚡ HAVING - Filtering After Aggregation

WHERE vs HAVING (Critical Interview Question!)

Aspect	WHERE	HAVING
Filters	Individual rows BEFORE grouping	Aggregated results AFTER grouping
Works with	Regular columns	Aggregate functions

Aspect	WHERE	HAVING
Position	Before GROUP BY	After GROUP BY

Visual Example

```sql
-- ❌ WRONG - Can't use aggregate in WHERE
SELECT category, AVG(price)
FROM products
WHERE AVG(price) > 100  -- ERROR!
GROUP BY category;

-- ✅ CORRECT - Use HAVING for aggregates
SELECT category, AVG(price)
FROM products
GROUP BY category
HAVING AVG(price) > 100;

-- Result: Only categories with avg price > 100
-- Electronics | 250.50
-- Appliances  | 180.75
```

WHERE + HAVING Together

```sql
-- Find categories with avg price > 100,
-- but only for products in stock
SELECT
    category,
    AVG(price) AS avg_price
FROM products
WHERE stock > 0          -- Filter rows FIRST
GROUP BY category
HAVING AVG(price) > 100;  -- Filter groups SECOND
```

🕌 Multiple Conditions in HAVING

```sql
```

```sql
-- Stocks with high average AND high minimum
SELECT
    ticker,
    AVG(price) AS avg_price,
    MIN(price) AS min_price
FROM stocks
GROUP BY ticker
HAVING AVG(price) > 200 AND MIN(price) > 100;

-- Result: Only tickers meeting BOTH conditions
-- NFLX | 420.69 | 176.49
-- MSFT | 254.08 | 153.00
```

📋 SQL Query Order (CRITICAL!)

Always write clauses in this order:

1. **SELECT** - What columns to show
2. **FROM** - Which table
3. **WHERE** - Filter rows (before grouping)
4. **GROUP BY** - Create groups
5. **HAVING** - Filter groups (after grouping)
6. **ORDER BY** - Sort results

```sql
-- Perfect query structure
SELECT category, COUNT(*) as total
FROM products
WHERE price > 50
GROUP BY category
HAVING COUNT(*) > 10
ORDER BY total DESC;
```

💡 Interview Question Patterns

Pattern 1: "Find total/average/count for each..."

Keywords: "for each", "by category", "per user" **Solution:** GROUP BY + aggregate function

```sql
```

```sql
-- "Average salary for each department"
SELECT department, AVG(salary)
FROM employees
GROUP BY department;
```

Pattern 2: "Find only [groups] where [condition]..."

Keywords: "only", "where average is", "having more than" **Solution:** GROUP BY + HAVING

sql

```sql
-- "Find departments where average salary > 80k"
SELECT department, AVG(salary)
FROM employees
GROUP BY department
HAVING AVG(salary) > 80000;
```

Pattern 3: "Count how many candidates have each skill"

sql

```sql
SELECT skill, COUNT(*) as candidate_count
FROM candidates
GROUP BY skill
ORDER BY candidate_count DESC;
```

Pattern 4: "Find users with more than X items"

sql

```sql
SELECT user_id, COUNT(*) as item_count
FROM orders
GROUP BY user_id
HAVING COUNT(*) > 2;
```

🚨 Common Mistakes to Avoid

❌ Mistake 1: Aggregate in WHERE

sql

```sql
-- WRONG
SELECT category, AVG(price)
FROM products
WHERE AVG(price) > 100  -- ERROR!
GROUP BY category;

-- RIGHT
SELECT category, AVG(price)
FROM products
GROUP BY category
HAVING AVG(price) > 100;
```

❌ Mistake 2: Missing column in GROUP BY

sql

```sql
-- WRONG
SELECT category, brand, SUM(sales)
FROM products
GROUP BY category;  -- Missing 'brand'!

-- RIGHT
SELECT category, brand, SUM(sales)
FROM products
GROUP BY category, brand;
```

❌ Mistake 3: Wrong clause order

sql

```sql
-- WRONG
SELECT * FROM products
GROUP BY category
WHERE price > 50;  -- WHERE must come BEFORE GROUP BY

-- RIGHT
SELECT category, AVG(price)
FROM products
WHERE price > 50
GROUP BY category;
```

🎓 Quick Memory Tricks

1. **"GROUP BY = Pivot Table"** - If you'd make a pivot table in Excel, use GROUP BY in SQL

2. **"WHERE = Before, HAVING = After"** - WHERE filters before grouping, HAVING filters after

3. **"Can't aggregate in WHERE"** - If you need AVG/SUM/COUNT in filter → use HAVING

4. **"SELECT = GROUP BY"** - Every non-aggregated column in SELECT must be in GROUP BY

5. **"SFWGHO"** - SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY (query order)

🏆 Practice Scenarios

Scenario 1: E-commerce Analysis

Question: "Find total sales by category, show only categories with sales > $1000"

```sql
SELECT category, SUM(sales) as total_sales
FROM orders
GROUP BY category
HAVING SUM(sales) > 1000
ORDER BY total_sales DESC;
```

Scenario 2: User Engagement

Question: "Find users who posted more than 5 times"

```sql
SELECT user_id, COUNT(*) as post_count
FROM posts
GROUP BY user_id
HAVING COUNT(*) > 5;
```

Scenario 3: Product Performance

Question: "Average rating by product, only products with 10+ reviews"

```sql
SELECT product_id, AVG(rating) as avg_rating, COUNT(*) as review_count
FROM reviews
GROUP BY product_id
HAVING COUNT(*) >= 10
ORDER BY avg_rating DESC;
```

✅ Pre-Interview Checklist

- [ ] Can you name all 5 aggregate functions?
- [ ] Can you explain WHERE vs HAVING?
- [ ] Do you know the SQL clause order (SFWGHO)?
- [ ] Can you use GROUP BY with multiple columns?
- [ ] Know when to use HAVING with multiple conditions?

Final Tip: When you see "for each" or "by" in a question → think GROUP BY!