SQL Interview Revision Notes

1. SELECT - Retrieving Data 🔦

Purpose: Tells the database what columns to display

Theory: SELECT is the foundation of SQL - it retrieves data from tables but doesn't modify anything in the database

Basic Syntax

```sql
SELECT column1, column2, column3
FROM table_name;
```

Select All Columns

```sql
SELECT *
FROM reviews;
```

Real Example - Amazon Reviews

```sql
SELECT review_id, submit_date, stars
FROM reviews;
```

Key Points:

- No trailing comma after last column
- `SELECT *` retrieves all columns
- Always specify table with `FROM`

---

2. WHERE - Filtering Rows 📍

Purpose: Filters rows based on conditions

Theory: WHERE clause filters rows BEFORE they're returned - reducing data processed and improving query performance

Basic Syntax

```sql
```

```
SELECT column1, column2
FROM table_name
WHERE condition;
```

Comparison Operators

Operator	Meaning	Example
`=`	Equals	`stars = 5`
`!=` or `<>`	Not equals	`stars != 5`
`<`	Less than	`stars < 4`
`>`	Greater than	`stars > 3`
`<=`	Less than or equal	`stars <= 4`
`>=`	Greater than or equal	`stars >= 3`

Examples

```sql
-- Single condition
SELECT *
FROM reviews
WHERE stars < 4;

-- Multiple conditions with AND
SELECT *
FROM reviews
WHERE stars < 4 AND user_id = 362;

-- Filter by text
SELECT *
FROM reviews
WHERE product_id = 12580;
```

Interview Tip: WHERE speeds up queries and reduces costs by processing less data!

---

3. AND, OR, NOT - Logical Operators ❌

Theory: Logical operators combine multiple conditions - AND requires ALL conditions true, OR requires ANY condition true, NOT negates a condition

AND Operator

Both conditions must be TRUE

```sql
SELECT *
FROM reviews
WHERE product_id = 50001 AND stars > 3;
```

Multiple AND Conditions

```sql
SELECT *
FROM reviews
WHERE stars > 3
  AND stars < 5
  AND product_id != 50001;
```

OR Operator

At least one condition must be TRUE

```sql
SELECT *
FROM reviews
WHERE stars > 3 OR product_id = 50001;
```

Combining AND with OR

```sql
SELECT *
FROM reviews
WHERE (stars = 3 OR stars = 4)
  AND review_id > 5000;
```

Note: Use parentheses to control evaluation order!

NOT Operator

Negates a condition

```sql
```

```sql
SELECT *
FROM reviews
WHERE NOT stars = 5;
-- Same as: WHERE stars != 5

-- Useful with BETWEEN
SELECT *
FROM reviews
WHERE stars NOT BETWEEN 2 AND 4;
```

---

4. BETWEEN - Range Filtering 🖐

Purpose: Filter values within a range (INCLUSIVE on both ends)

Theory: BETWEEN simplifies range queries and is inclusive on both boundaries - saves writing two separate comparison operators

Syntax

sql

```sql
SELECT column_name
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Examples

sql

```sql
-- Olympics gold medals from 2000s
SELECT *
FROM medals
WHERE type = 'GOLD'
  AND year BETWEEN 2000 AND 2010;

-- Medicine sales between 100k and 105k units
SELECT *
FROM pharmacy_sales
WHERE units_sold BETWEEN 100000 AND 105000;
```

BETWEEN is Inclusive!

sql

```sql
-- These are DIFFERENT:
WHERE price > 10000 AND price <= 20000  -- Excludes 10000
WHERE price BETWEEN 10000 AND 20000     -- Includes 10000
```

Better than multiple ORs:

```sql
-- Instead of:
WHERE year = 2000 OR year = 2004 OR year = 2008

-- Use:
WHERE year BETWEEN 2000 AND 2010
```

---

5. IN - List Membership 🔎

Purpose: Check if value matches ANY in a list

Theory: IN operator checks if a value exists in a specified list - more efficient and readable than chaining multiple OR conditions

Syntax

```sql
SELECT column_name
FROM table_name
WHERE column_name IN (value1, value2, value3);
```

Example - CVS Pharmacy

```sql
-- Find drugs from specific manufacturers
SELECT drug, manufacturer, units_sold
FROM pharmacy_sales
WHERE manufacturer IN ('Biogen', 'Bayer', 'Eli Lilly');
```

Much cleaner than:

```sql
WHERE manufacturer = 'Biogen'
  OR manufacturer = 'Bayer'
  OR manufacturer = 'Eli Lilly';
```

Combining with NOT

```sql
SELECT *
FROM pharmacy_sales
WHERE manufacturer IN ('Roche', 'Bayer', 'AstraZeneca')
  AND units_sold NOT BETWEEN 55000 AND 550000;
```

---

6. LIKE - Pattern Matching ❤️

Purpose: Match strings against patterns using wildcards

Theory: LIKE enables pattern matching with wildcards - % matches zero or more characters, _ matches exactly one character

Wildcards

- `%` - Represents zero or more characters
- `_` - Represents exactly one character

Pattern Examples

Pattern	Matches	Example
`'a%'`	Starts with 'a'	apple, analytics, awesome
`'%a'`	Ends with 'a'	data, banana, camera
`'%Relief%'`	Contains 'Relief'	Pain Relief, Relief Plus
`'_b%'`	'b' in 2nd position	abc, obelisk
`'a%o'`	Starts with 'a', ends with 'o'	avocado, amigo
`'a___'`	Starts with 'a', 4 letters total	able, aunt

Real Examples

```sql
```

```sql
-- Find drugs with "Relief" anywhere in name
SELECT product_id, manufacturer, drug
FROM pharmacy_sales
WHERE drug LIKE '%Relief%';

-- Find employees whose name starts with 'ke' and ends with 'y'
SELECT *
FROM employees
WHERE first_name LIKE 'ke%y';

-- Find 4-letter words starting with 'f', 3rd letter is 'c'
SELECT *
FROM dictionary_words
WHERE word LIKE 'f_c_';

-- Customers starting with 'F', ending with 'ck'
SELECT *
FROM customers
WHERE name LIKE 'F%ck';

-- Names where 2nd and 3rd letters are both 'e'
SELECT *
FROM customers
WHERE name LIKE '_ee%';
```

---

7. ORDER BY - Sorting Results ↕️

Purpose: Sort query results

Theory: ORDER BY sorts the final result set AFTER all filtering - default is ASC (ascending), database row order is not guaranteed without ORDER BY

Syntax

```sql
SELECT column1, column2
FROM table_name
WHERE condition
ORDER BY column1 [ASC|DESC];
```

Ascending (Default)

```sql
```

```sql
SELECT product_id, drug, units_sold
FROM pharmacy_sales
ORDER BY drug;  -- Alphabetically A-Z
```

Descending

sql

```sql
SELECT product_id, drug, units_sold
FROM pharmacy_sales
ORDER BY units_sold DESC;  -- Highest first
```

Multiple Columns

sql

```sql
SELECT policy_holder_id, call_category, call_received
FROM callers
ORDER BY policy_holder_id, call_received DESC;
```

Using Column Numbers

sql

```sql
SELECT policy_holder_id, call_category, call_received
FROM callers
ORDER BY 1, 3 DESC;
-- 1 = policy_holder_id (1st column in SELECT)
-- 3 = call_received (3rd column in SELECT)
```

LIMIT and OFFSET

sql

```sql
-- Get top 5 most recent calls
SELECT *
FROM callers
ORDER BY call_received DESC
LIMIT 5;

-- Skip first 10, get next 5
SELECT *
FROM callers
ORDER BY call_received DESC
OFFSET 10
LIMIT 5;
```

8. Complete Filtering Examples

Complex Multi-Condition Query

sql

```sql
-- Australian customers: ages 18-22, specific states,
-- not n/a gender, name starts with A or B
SELECT customer_id, customer_name, gender, age, state
FROM customers
WHERE age BETWEEN 18 AND 22
  AND state IN ('Victoria', 'Tasmania', 'Queensland')
  AND gender != 'n/a'
  AND (customer_name LIKE 'A%' OR customer_name LIKE 'B%')
ORDER BY age, customer_name;
```

Medicine Sales Analysis

sql

```sql
-- Find medicines: 100k-105k units sold,
-- specific manufacturers
SELECT manufacturer, drug, units_sold
FROM pharmacy_sales
WHERE units_sold BETWEEN 100000 AND 105000
  AND manufacturer IN ('Biogen', 'AbbVie', 'Eli Lilly')
ORDER BY units_sold DESC;
```

Quick Reference Table

Command	Purpose	Example
SELECT	Choose columns	`SELECT name, age`
FROM	Specify table	`FROM users`
WHERE	Filter rows	`WHERE age > 18`
AND	Both conditions true	`age > 18 AND age < 65`
OR	Any condition true	`city = 'NYC' OR city = 'LA'`
NOT	Negate condition	`NOT age = 25`
BETWEEN	Range (inclusive)	`age BETWEEN 18 AND 25`
IN	Match list	`city IN ('NYC', 'LA', 'SF')`
LIKE	Pattern match	`name LIKE 'J%'`
ORDER BY	Sort results	`ORDER BY age DESC`

Command	Purpose	Example
LIMIT	Limit rows returned	`LIMIT 10`

Interview Tips 💡

1. **Start with SELECT * to explore data** before writing complex queries
2. **Use parentheses with AND/OR** to make logic clear: `(A OR B) AND C`
3. **BETWEEN is inclusive** - includes both boundary values
4. **IN is cleaner than multiple ORs** for categorical data
5. **Use LIKE for fuzzy matching** when you don't know exact values
6. **Always ORDER BY when using LIMIT** to get consistent results
7. **WHERE before ORDER BY** in query structure
8. **Comments help in interviews:** `-- This filters active users`

Common Query Pattern

sql

```sql
SELECT column1, column2, column3          -- 1. Choose what to display
FROM table_name                           -- 2. From which table
WHERE condition1                          -- 3. Filter rows
  AND/OR condition2                       -- 4. Multiple conditions
  AND column3 BETWEEN value1 AND value2   -- 5. Range filtering
  AND column4 IN ('val1', 'val2')         -- 6. List filtering
  AND column5 LIKE 'pattern%'             -- 7. Pattern matching
ORDER BY column1 DESC, column2 ASC        -- 8. Sort results
LIMIT 100;                                -- 9. Limit output
```

Practice Makes Perfect! 🎯

Remember: SQL is about **retrieving** (SELECT), **filtering** (WHERE), and **organizing** (ORDER BY) data. Master these basics and you're ready for interviews!

Good luck! 🚀