DBMS and SQL: 200 Questions Covering All Concepts

Database Fundamentals (20 questions)

1. What is a database?
2. What are the advantages of DBMS over file systems?
3. Explain the three-schema architecture of DBMS.
4. What is data independence? Explain physical and logical data independence.
5. What are the different types of database users?
6. What is a data model? Name different types of data models.
7. Explain the entity-relationship model.
8. What is normalization and why is it important?
9. What are ACID properties in DBMS?
10. Explain the client-server architecture in DBMS.
11. What is a database schema?
12. What is data abstraction in DBMS?
13. What is a data dictionary?
14. Explain the different types of database languages.
15. What is a database instance?
16. What are the responsibilities of a DBA?
17. Explain the difference between logical and physical database design.
18. What are the different types of database interfaces?
19. What is a database system environment?
20. Explain the difference between centralized and distributed database systems.

SQL Basics (30 questions)

21. What is SQL?
22. What are the different subsets of SQL?
23. Explain the difference between DDL, DML, and DCL.
24. How do you create a database in SQL?
25. How do you create a table in SQL?
26. What are SQL constraints? Name different types.
27. What is the difference between DELETE, TRUNCATE, and DROP commands?
28. How do you add a column to an existing table?
29. How do you modify a column in a table?
30. What is the difference between CHAR and VARCHAR data types?
31. What is the difference between VARCHAR and VARCHAR2?
32. How do you rename a table in SQL?
33. What is the purpose of the ALTER command?
34. How do you add a primary key to an existing table?
35. How do you remove a constraint from a table?
36. What is the difference between a primary key and a unique key?
37. What is a composite primary key?
38. How do you create an index in SQL?
39. What is the difference between a clustered and non-clustered index?
40. How do you insert data into a table?
41. How do you insert multiple rows in a single SQL query?
42. What is the difference between INSERT and INSERT INTO?
43. How do you update data in a table?
44. How do you delete all records from a table?
45. What is the difference between a table and a view?
46. How do you create a view in SQL?
47. Can you update a view? What are the limitations?
48. What is a materialized view?
49. How do you drop a view?
50. What is the purpose of the WITH CHECK OPTION in view creation?

SQL Queries (40 questions)

51. What is the SELECT statement used for?
52. How do you select all columns from a table?
53. How do you select specific columns from a table?
54. What is the WHERE clause used for?
55. What are the different comparison operators in SQL?
56. How do you use the LIKE operator?
57. What are wildcards in SQL? Explain % and _.
58. How do you sort results in SQL?
59. What is the difference between ORDER BY ASC and DESC?
60. How do you limit the number of rows returned in a query?
61. What is the difference between TOP, LIMIT, and ROWNUM?
62. How do you use the DISTINCT keyword?
63. What is the difference between COUNT(*) and COUNT(column_name)?
64. How do you use GROUP BY in SQL?
65. What is the HAVING clause used for?
66. What is the difference between WHERE and HAVING?
67. How do you concatenate strings in SQL?
68. What are SQL functions? Name different types.
69. How do you use aggregate functions in SQL?
70. Explain the difference between single-row and multiple-row functions.
71. What is the difference between UNION and UNION ALL?
72. How do you use the INTERSECT operator?
73. What is the MINUS (EXCEPT) operator used for?
74. How do you perform joins in SQL?
75. What are the different types of joins?
76. Explain INNER JOIN with an example.
77. Explain LEFT OUTER JOIN with an example.
78. Explain RIGHT OUTER JOIN with an example.

79. Explain FULL OUTER JOIN with an example.
80. What is a self join? Provide an example.
81. What is a cross join?
82. What is the difference between INNER JOIN and OUTER JOIN?
83. What is a natural join?
84. How do you use subqueries in SQL?
85. What are correlated subqueries?
86. What is the difference between a subquery and a join?
87. How do you use the EXISTS operator?
88. What is the difference between IN and EXISTS?
89. How do you use the CASE statement in SQL?
90. What are common table expressions (CTEs)?

Advanced SQL (30 questions)

91. What are stored procedures?
92. How do you create a stored procedure?
93. What are the advantages of stored procedures?
94. What are parameters in stored procedures?
95. What is the difference between input and output parameters?
96. How do you execute a stored procedure?
97. What are functions in SQL?
98. What is the difference between stored procedures and functions?
99. How do you create a function in SQL?
100. What are triggers in SQL?
101. What are the different types of triggers?
102. How do you create a trigger?
103. What is the difference between AFTER and INSTEAD OF triggers?
104. What are cursors in SQL?
105. How do you use cursors?

106. What are the disadvantages of cursors?
107. What is transaction management in SQL?
108. How do you begin and end a transaction?
109. What is the COMMIT command used for?
110. What is the ROLLBACK command used for?
111. What is the SAVEPOINT command used for?
112. What is a deadlock in DBMS?
113. How can deadlocks be prevented?
114. What is concurrency control in DBMS?
115. What are isolation levels in SQL?
116. Explain READ UNCOMMITTED isolation level.
117. Explain READ COMMITTED isolation level.
118. Explain REPEATABLE READ isolation level.
119. Explain SERIALIZABLE isolation level.
120. What is locking in DBMS? Explain different types of locks.

Database Design and Normalization (20 questions)

121. What is database design?
122. What is the difference between conceptual, logical, and physical database design?
123. What is an entity in DBMS?
124. What is an attribute? Explain different types.
125. What is a relationship? Explain different types.
126. What is cardinality in DBMS?
127. What is participation constraint?
128. What is normalization?
129. What is denormalization and when is it used?
130. Explain 1NF (First Normal Form).
131. Explain 2NF (Second Normal Form).
132. Explain 3NF (Third Normal Form).

133. Explain BCNF (Boyce-Codd Normal Form).
134. Explain 4NF (Fourth Normal Form).
135. Explain 5NF (Fifth Normal Form).
136. What is functional dependency?
137. What is transitive dependency?
138. What is partial dependency?
139. What is a surrogate key?
140. What is a candidate key?

Advanced Database Concepts (30 questions)

141. What is indexing in DBMS?
142. What are the different types of indexes?
143. What is a B-tree index?
144. What is a bitmap index?
145. What is a hash index?
146. What is query optimization?
147. What is an execution plan?
148. What are the different join algorithms?
149. What is database partitioning?
150. What are the different partitioning strategies?
151. What is database sharding?
152. What is replication in DBMS?
153. What are the different replication strategies?
154. What is database clustering?
155. What is a distributed database?
156. What are the advantages of distributed databases?
157. What is CAP theorem?
158. What is BASE in database systems?
159. What is NoSQL?

160. What are the different types of NoSQL databases?
161. What is a document database?
162. What is a key-value store?
163. What is a column-family store?
164. What is a graph database?
165. What is NewSQL?
166. What is a data warehouse?
167. What is OLTP vs OLAP?
168. What is ETL process?
169. What is data mining?
170. What is big data?

SQL Performance Tuning (15 questions)

171. What is SQL query optimization?
172. How do you analyze query performance?
173. What is the EXPLAIN PLAN statement?
174. How do indexes improve query performance?
175. When should you avoid using indexes?
176. What is query rewriting?
177. How does join order affect query performance?
178. What are the best practices for writing efficient SQL queries?
179. How do you handle large result sets efficiently?
180. What is parameter sniffing in SQL Server?
181. How do you avoid table scans?
182. What is a covering index?
183. How do you optimize subqueries?
184. What is the N+1 query problem?
185. How do you optimize LIKE queries?

Database Security (15 questions)

186. What is database security?
187. What are the different types of database security threats?
188. What is authentication in DBMS?
189. What is authorization in DBMS?
190. What are roles in database security?
191. How do you create a user in SQL?
192. How do you grant privileges to a user?
193. What is the difference between GRANT and REVOKE?
194. What are system privileges vs object privileges?
195. What is data encryption in DBMS?
196. What is data masking?
197. What is SQL injection?
198. How can you prevent SQL injection attacks?
199. What is auditing in DBMS?
200. What are the best practices for database security?

Basic SQL Problems (15)

1. Select all columns from a table

```sql
SELECT * FROM employees;
```

2. Select specific columns from a table

```sql
SELECT first_name, last_name, salary FROM employees;
```

3. Filter records using WHERE clause

```sql
SELECT * FROM products WHERE price > 100;
```

4. Sort results using ORDER BY

```sql
SELECT * FROM customers ORDER BY last_name ASC;
```

5. Limit the number of results returned

```sql
-- MySQL/PostgreSQL
SELECT * FROM orders LIMIT 10;

-- SQL Server
SELECT TOP 10 * FROM orders;

-- Oracle
SELECT * FROM orders WHERE ROWNUM <= 10;
```

6. Use DISTINCT to find unique values

```sql
SELECT DISTINCT department_id FROM employees;
```

7. Count the number of records

```sql
SELECT COUNT(*) FROM orders;
```

8. Group data using GROUP BY

```sql
SELECT department_id, COUNT(*)
FROM employees
GROUP BY department_id;
```

9. Filter groups using HAVING

```sql
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id
HAVING AVG(salary) > 50000;
```

10. Concatenate strings

```sql
SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM
   employees;
```

11. Use CASE expressions

```sql
SELECT product_name,
       CASE WHEN price > 100 THEN 'Expensive'
            WHEN price > 50 THEN 'Moderate'
            ELSE 'Cheap' END AS price_category
FROM products;
```

12. Calculate averages, sums, min, max

```
SELECT AVG(salary), SUM(salary), MIN(salary), MAX(salary)
FROM employees;
```

13. Find records with NULL values

```
SELECT * FROM employees WHERE manager_id IS NULL;
```

14. Find records without NULL values

```
SELECT * FROM employees WHERE manager_id IS NOT NULL;
```

15. Use BETWEEN for range queries

```
SELECT * FROM orders
WHERE order_date BETWEEN '2023-01-01' AND '2023-12-31';
```

Intermediate SQL Problems (20)

16. Inner join two tables

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id;
```

17. Left outer join

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id;
```

18. Self join (employees and their managers)

```
SELECT e.first_name AS employee, m.first_name AS manager
FROM employees e
LEFT JOIN employees m ON e.manager_id = m.employee_id;
```

19. Find duplicate records

```sql
SELECT email, COUNT(*)
FROM customers
GROUP BY email
HAVING COUNT(*) > 1;
```

20. Delete duplicate records

```sql
DELETE FROM customers
WHERE customer_id NOT IN (
    SELECT MIN(customer_id)
    FROM customers
    GROUP BY email
);
```

21. Find the nth highest salary

```sql
-- For 2nd highest salary
SELECT MAX(salary) FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);

-- Using window function (more flexible)
SELECT salary FROM (
    SELECT salary, DENSE_RANK() OVER (ORDER BY salary DESC) as
       rank
    FROM employees
) WHERE rank = 2;
```

22. Find employees with salary above average

```sql
SELECT * FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

23. Find departments with more than 5 employees

```sql
SELECT department_id, COUNT(*)
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 5;
```

24. Use IN with subquery

```
SELECT * FROM products
WHERE category_id IN (
    SELECT category_id FROM categories WHERE category_name LIKE '
       Electronics%'
);
```

25. Use EXISTS with subquery

```
SELECT * FROM departments d
WHERE EXISTS (
    SELECT 1 FROM employees e
    WHERE e.department_id = d.department_id
);
```

26. Find employees who don't have dependents

```
SELECT * FROM employees e
WHERE NOT EXISTS (
    SELECT 1 FROM dependents d
    WHERE d.employee_id = e.employee_id
);
```

27. Calculate running totals

```
SELECT order_date, amount,
       SUM(amount) OVER (ORDER BY order_date) AS running_total
FROM orders;
```

28. Find top 3 products by sales in each category

```
SELECT * FROM (
    SELECT p.product_name, c.category_name, s.sales_amount,
           DENSE_RANK() OVER (PARTITION BY c.category_id ORDER BY
               s.sales_amount DESC) as rank
    FROM products p
    JOIN categories c ON p.category_id = c.category_id
    JOIN sales s ON p.product_id = s.product_id
) WHERE rank <= 3;
```

29. Calculate month-over-month growth

```
SELECT
    EXTRACT(YEAR FROM order_date) AS year,
    EXTRACT(MONTH FROM order_date) AS month,
    SUM(amount) AS monthly_sales,
    LAG(SUM(amount), 1) OVER (ORDER BY EXTRACT(YEAR FROM
       order_date), EXTRACT(MONTH FROM order_date)) AS
       prev_month_sales,
    (SUM(amount) - LAG(SUM(amount), 1) OVER (ORDER BY EXTRACT(
       YEAR FROM order_date), EXTRACT(MONTH FROM order_date))) /
    LAG(SUM(amount), 1) OVER (ORDER BY EXTRACT(YEAR FROM
       order_date), EXTRACT(MONTH FROM order_date)) * 100 AS
       growth_percentage
FROM orders
GROUP BY EXTRACT(YEAR FROM order_date), EXTRACT(MONTH FROM
   order_date)
ORDER BY year, month;
```

30. Pivot data (rows to columns)

```
SELECT
    product_id,
    MAX(CASE WHEN quarter = 1 THEN sales END) AS Q1_sales,
    MAX(CASE WHEN quarter = 2 THEN sales END) AS Q2_sales,
    MAX(CASE WHEN quarter = 3 THEN sales END) AS Q3_sales,
    MAX(CASE WHEN quarter = 4 THEN sales END) AS Q4_sales
FROM quarterly_sales
GROUP BY product_id;
```

31. Unpivot data (columns to rows)

```
SELECT product_id, 'Q1' AS quarter, Q1_sales AS sales
FROM annual_sales
UNION ALL
SELECT product_id, 'Q2' AS quarter, Q2_sales AS sales
FROM annual_sales
UNION ALL
SELECT product_id, 'Q3' AS quarter, Q3_sales AS sales
FROM annual_sales
UNION ALL
SELECT product_id, 'Q4' AS quarter, Q4_sales AS sales
FROM annual_sales;
```

32. Find gaps in sequential data

```
SELECT prev_id + 1 AS start_gap, curr_id - 1 AS end_gap
FROM (
    SELECT
        id AS curr_id,
        LAG(id, 1) OVER (ORDER BY id) AS prev_id
    FROM sequence_table
)
WHERE curr_id > prev_id + 1;
```

33. Calculate cumulative distribution

```
SELECT
    employee_id,
    salary,
    CUME_DIST() OVER (ORDER BY salary) AS cumulative_dist
FROM employees;
```

34. Find median salary

```
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY salary) AS
   median_salary
FROM employees;
```

35. Generate a date series

```
-- PostgreSQL
SELECT generate_series(
    '2023-01-01'::date,
    '2023-12-31'::date,
    '1 day'::interval
) AS date;
```

Advanced SQL Problems (15)

36. Recursive query (find hierarchy)

```
WITH RECURSIVE employee_hierarchy AS (
    -- Base case
    SELECT employee_id, first_name, last_name, manager_id, 1 AS
       level
    FROM employees
    WHERE manager_id IS NULL

    UNION ALL

    -- Recursive case
    SELECT e.employee_id, e.first_name, e.last_name, e.manager_id
       , eh.level + 1
    FROM employees e
    JOIN employee_hierarchy eh ON e.manager_id = eh.employee_id
)
SELECT * FROM employee_hierarchy ORDER BY level;
```

37. Find longest consecutive sequence

```
WITH numbered_days AS (
    SELECT date, date - ROW_NUMBER() OVER (ORDER BY date) AS grp
    FROM attendance
)
SELECT MIN(date) AS start_date, MAX(date) AS end_date, COUNT(*)
   AS days
FROM numbered_days
GROUP BY grp
ORDER BY days DESC
LIMIT 1;
```

38. Calculate moving average

```
SELECT
    date,
    amount,
    AVG(amount) OVER (ORDER BY date ROWS BETWEEN 2 PRECEDING AND
       CURRENT ROW) AS moving_avg
FROM sales;
```

39. Find employees with the same salary

```
SELECT salary, STRING_AGG(first_name, ', ') AS employees
FROM employees
GROUP BY salary
HAVING COUNT(*) > 1;
```

40. Implement pagination

```
-- Offset pagination
SELECT * FROM products
ORDER BY product_name
LIMIT 10 OFFSET 20; -- Page 3 (rows 21-30)

-- Keyset pagination (better for large datasets)
SELECT * FROM products
WHERE product_id > 100  -- Last ID from previous page
ORDER BY product_id
LIMIT 10;
```

41. Calculate retention rate

```
SELECT
    a.signup_month,
    COUNT(DISTINCT a.user_id) AS cohort_size,
    COUNT(DISTINCT b.user_id) / COUNT(DISTINCT a.user_id)::FLOAT
       AS retention_rate
FROM user_signups a
LEFT JOIN user_activity b ON a.user_id = b.user_id
    AND b.activity_month = a.signup_month + 1
GROUP BY a.signup_month;
```

42. Find sessions with no conversion

```
SELECT s.session_id, s.start_time, s.end_time
FROM user_sessions s
LEFT JOIN conversions c ON s.session_id = c.session_id
WHERE c.session_id IS NULL;
```

43. Calculate session duration statistics

```
SELECT
    AVG(EXTRACT(EPOCH FROM (end_time - start_time))) AS
       avg_duration_seconds,
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY EXTRACT(EPOCH
       FROM (end_time - start_time))) AS median_duration_seconds
FROM user_sessions;
```

44. Find customers who purchased all products in a category

```
SELECT c.customer_id, c.customer_name
FROM customers c
WHERE NOT EXISTS (
    SELECT p.product_id
    FROM products p
    WHERE p.category_id = 5
    EXCEPT
    SELECT o.product_id
    FROM orders o
    WHERE o.customer_id = c.customer_id
);
```

45. Calculate RFM (Recency, Frequency, Monetary) analysis

```
WITH rfm AS (
    SELECT
        customer_id,
        MAX(order_date) AS last_order_date,
        COUNT(*) AS frequency,
        SUM(amount) AS monetary,
        NTILE(5) OVER (ORDER BY MAX(order_date) DESC) AS
           recency_score,
        NTILE(5) OVER (ORDER BY COUNT(*)) AS frequency_score,
        NTILE(5) OVER (ORDER BY SUM(amount)) AS monetary_score
    FROM orders
    GROUP BY customer_id
)
SELECT
    customer_id,
    recency_score,
    frequency_score,
    monetary_score,
    (recency_score + frequency_score + monetary_score) AS
       rfm_score
FROM rfm
ORDER BY rfm_score DESC;
```

46. Find mutually following pairs (social network)

```
SELECT a.follower_id AS user1, a.followee_id AS user2
FROM follows a
JOIN follows b ON a.follower_id = b.followee_id AND a.followee_id
    = b.follower_id
WHERE a.follower_id < a.followee_id;  -- Avoid duplicates
```

47. Calculate employee tenure distribution

```
SELECT
    tenure_bucket,
    COUNT(*) AS employee_count
FROM (
    SELECT
        employee_id,
        CASE
            WHEN tenure < 1 THEN '0-1 years'
            WHEN tenure < 3 THEN '1-3 years'
            WHEN tenure < 5 THEN '3-5 years'
            ELSE '5+ years'
        END AS tenure_bucket
    FROM (
        SELECT
            employee_id,
            EXTRACT(YEAR FROM age(current_date, hire_date)) AS
               tenure
        FROM employees
    ) t
) t2
GROUP BY tenure_bucket
ORDER BY tenure_bucket;
```

48. Find most recent record for each group

```
SELECT p.*
FROM product_prices p
INNER JOIN (
    SELECT product_id, MAX(effective_date) AS max_date
    FROM product_prices
    WHERE effective_date <= CURRENT_DATE
    GROUP BY product_id
) latest ON p.product_id = latest.product_id AND p.effective_date
    = latest.max_date;
```

49. Calculate employee turnover rate

```
SELECT
    department_id,
    COUNT(*) AS total_employees,
    SUM(CASE WHEN termination_date IS NOT NULL THEN 1 ELSE 0 END)
        AS terminated_employees,
    SUM(CASE WHEN termination_date IS NOT NULL THEN 1 ELSE 0 END)
        / COUNT(*)::FLOAT AS turnover_rate
FROM employees
GROUP BY department_id;
```

50. Find customers with increasing purchase amounts

```
WITH ordered_purchases AS (
    SELECT
        customer_id,
        order_date,
        amount,
        LAG(amount, 1) OVER (PARTITION BY customer_id ORDER BY
           order_date) AS prev_amount
    FROM orders
)
SELECT DISTINCT customer_id
FROM ordered_purchases
GROUP BY customer_id
HAVING COUNT(*) > 1
   AND COUNT(CASE WHEN amount <= prev_amount OR prev_amount IS
      NULL THEN 1 END) = 0;
```

These problems cover a wide range of SQL concepts from basic queries to advanced analytical functions, providing a comprehensive practice set for SQL proficiency.