That's quite a comprehensive list of SQL questions! I'll provide detailed answers to each of these questions. Let's get started:

### **Basic SQL Questions**

1. **What is SQL?** SQL (Structured Query Language) is a standard programming language used for managing and manipulating relational databases. It allows users to create, read, update, and delete database records.
2. **What are the different types of SQL statements?**
   1. **DDL (Data Definition Language):** CREATE, ALTER, DROP, TRUNCATE
   2. **DML (Data Manipulation Language):** SELECT, INSERT, UPDATE, DELETE
   3. **DCL (Data Control Language):** GRANT, REVOKE
   4. **TCL (Transaction Control Language):** COMMIT, ROLLBACK, SAVEPOINT
3. **What is a primary key?** A primary key is a unique identifier for a record in a table. It ensures that each record is unique and cannot contain NULL values.
4. **Explain the difference between primary key and unique key.**
   1. **Primary Key:** Uniquely identifies each record in a table and cannot be NULL.
   2. **Unique Key:** Ensures all values in a column are unique but can contain a single NULL value.
5. **What is a foreign key?** A foreign key is a column or a set of columns in one table that references the primary key columns of another table, establishing a relationship between the two tables.
6. **What are joins in SQL? Name different types of joins.** Joins are used to combine rows from two or more tables based on a related column. Types of joins:
   1. INNER JOIN
   2. LEFT JOIN (or LEFT OUTER JOIN)
   3. RIGHT JOIN (or RIGHT OUTER JOIN)
   4. FULL JOIN (or FULL OUTER JOIN)
   5. CROSS JOIN
   6. SELF JOIN
7. **What is the difference between INNER JOIN and OUTER JOIN?**
   1. **INNER JOIN:** Returns only the rows that have matching values in both tables.
   2. **OUTER JOIN:** Returns all rows from one table and the matched rows from the other table. If there is no match, NULL values are returned for columns from the table without a match.
8. **What is a self join?** A self join is a regular join but the table is joined with itself.
9. **What is a subquery in SQL?** A subquery is a query nested inside another query. It can be used in SELECT, INSERT, UPDATE, or DELETE statements.
10. **What is the difference between DELETE and TRUNCATE?**
    1. **DELETE:** Removes rows from a table based on a condition. It can be rolled back.
    2. **TRUNCATE:** Removes all rows from a table without logging individual row deletions. It cannot be rolled back.
11. **Explain the difference between CHAR and VARCHAR.**
    1. **CHAR:** Fixed-length character data type. It pads extra spaces for shorter values.
    2. **VARCHAR:** Variable-length character data type. It does not pad extra spaces.
12. **What are indexes in SQL?** Indexes are used to speed up the retrieval of data from a table by creating a data structure that allows for faster searches.
13. **What are constraints in SQL?** Constraints are rules enforced on data columns to ensure data integrity. Examples include PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, and CHECK.
14. **Explain the concept of normalization.** Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller, related tables.
15. **What is denormalization?** Denormalization is the process of combining normalized tables to improve read performance by reducing the number of joins.
16. **What are the different normal forms?**
    1. **1NF (First Normal Form):** Ensures each column contains atomic values.
    2. **2NF (Second Normal Form):** Meets 1NF and all non-key attributes are fully functional dependent on the primary key.
    3. **3NF (Third Normal Form):** Meets 2NF and all attributes are functionally dependent only on the primary key.
    4. **BCNF (Boyce-Codd Normal Form):** A stricter version of 3NF.
17. **Explain the difference between a clustered and non-clustered index.**
    1. **Clustered Index:** Determines the physical order of data in a table. Only one clustered index per table.
    2. **Non-Clustered Index:** Does not alter the physical order of data. Multiple non-clustered indexes can exist per table.
18. **What is a stored procedure?** A stored procedure is a precompiled collection of SQL statements stored in the database that can be executed as a single unit.
19. **What are views in SQL?** Views are virtual tables created by a query that selects data from one or more tables.
20. **What is a unique constraint?** A unique constraint ensures that all values in a column or a set of columns are unique.
21. **What is the difference between HAVING and WHERE clauses?**
    1. **WHERE:** Filters rows before any groupings are made.
    2. **HAVING:** Filters groups after the GROUP BY clause has been applied.
22. **What is the difference between UNION and UNION ALL?**
    1. **UNION:** Combines the result sets of two queries and removes duplicate rows.
    2. **UNION ALL:** Combines the result sets of two queries without removing duplicates.
23. **What is the purpose of the GROUP BY clause?** The GROUP BY clause groups rows that have the same values in specified columns into summary rows.
24. **What are aggregate functions? Provide examples.** Aggregate functions perform calculations on a set of values and return a single value. Examples: COUNT(), SUM(), AVG(), MAX(), MIN().
25. **Explain the difference between COUNT(), COUNT(column), and COUNT(\*).**
    1. **COUNT():** Counts all rows.
    2. **COUNT(column):** Counts non-NULL values in a column.
    3. **COUNT(\*):** Counts all rows, including rows with NULL values.
26. **What is a composite key?** A composite key is a primary key composed of two or more columns.
27. **Explain the difference between a natural join and a cross join.**
    1. **Natural Join:** Automatically joins tables based on columns with the same name and data type.
    2. **Cross Join:** Returns the Cartesian product of two tables, combining all rows from both tables.
28. **What is a default constraint in SQL?** A default constraint sets a default value for a column when no value is specified.
29. **What is a cursor in SQL?** A cursor is a database object used to retrieve, manipulate, and navigate through a result set row by row.
30. **What are the ACID properties of a transaction?**
    1. **Atomicity:** Ensures all operations within a transaction are completed successfully or none are.
    2. **Consistency:** Ensures the database remains in a consistent state before and after the transaction.
    3. **Isolation:** Ensures transactions are isolated from each other.
    4. **Durability:** Ensures that once a transaction is committed, it remains permanent.

### **Intermediate SQL Questions**

1. **What is a common table expression (CTE)?** A CTE is a temporary result set defined within the execution scope of a single SELECT, INSERT, UPDATE, or DELETE statement.
2. **Explain recursive CTE with an example.** A recursive CTE references itself to produce a result set. Example:

WITH RECURSIVE cte AS (  
 SELECT 1 AS n  
 UNION ALL  
 SELECT n + 1  
 FROM cte  
 WHERE n < 10  
)  
SELECT \* FROM cte;

1. **What is a correlated subquery?** A correlated subquery is a subquery that references columns from the outer query.
2. **Explain the difference between RANK(), DENSE*RANK(), and ROW*NUMBER().**
   1. **RANK():** Assigns a rank to each row within a partition, with gaps in ranking for ties.
   2. **DENSE\_RANK():** Similar to RANK() but without gaps in ranking for ties.
   3. **ROW\_NUMBER():** Assigns a unique sequential number to each row within a partition.
3. **How do you use the CASE statement in SQL?** The CASE statement allows conditional logic in SQL queries. Example:

SELECT  
 employee\_id,  
 salary,  
 CASE  
 WHEN salary > 50000 THEN 'High'  
 WHEN salary > 30000 THEN 'Medium'  
 ELSE 'Low'  
 END AS salary\_level  
FROM employees;

1. **What is the difference between UNION and JOIN?**
   1. **UNION:** Combines result sets of two queries.
   2. **JOIN:** Combines rows from two or more tables based on a related column.
2. **What is a trigger? Explain types of triggers.** A trigger is a set of SQL statements that automatically execute in response to certain events on a table. Types:
   1. **BEFORE Trigger:** Executes before an operation.
   2. **AFTER Trigger:** Executes after an operation.
   3. **INSTEAD OF Trigger:** Executes instead of an operation.
3. **What are transaction isolation levels?** Isolation levels define the degree to which transactions are isolated from each other. Levels:
   1. **Read Uncommitted**
   2. **Read Committed**
   3. **Repeatable Read**
   4. **Serializable**
4. **Explain the use of the COALESCE() function.** The COALESCE

Sure, let's continue with the SQL questions:

1. **Explain the use of the COALESCE() function.** The COALESCE() function returns the first non-null value in a list of arguments. It's useful for handling null values. Example:

SELECT COALESCE(column1, column2, 'default\_value') AS result  
FROM table\_name;

1. **What is a pivot table in SQL?** A pivot table is used to transform rows into columns. It is often used for reporting and data analysis. Example using the PIVOT operator:

SELECT \*  
FROM (  
 SELECT Year, Quarter, Sales  
 FROM SalesData  
) AS SourceTable  
PIVOT (  
 SUM(Sales)  
 FOR Quarter IN ([Q1], [Q2], [Q3], [Q4])  
) AS PivotTable;

1. **What are window functions?** Window functions perform calculations across a set of table rows related to the current row. Examples include ROW*NUMBER(), RANK(), DENSE*RANK(), and SUM() OVER().
2. **What is the difference between DELETE and DROP?**
   1. **DELETE:** Removes rows from a table based on a condition. The table structure remains.
   2. **DROP:** Removes the entire table or database, including its structure and data.
3. **How do you optimize SQL queries for performance?**
   1. Use indexes appropriately.
   2. Avoid using SELECT \*.
   3. Use joins instead of subqueries.
   4. Optimize WHERE clauses with proper indexing.
   5. Limit the number of rows returned using LIMIT or TOP.
   6. Analyze and optimize query execution plans.
4. **What is the difference between EXISTS and IN?**
   1. **EXISTS:** Checks for the existence of rows in a subquery. It stops processing once a match is found.
   2. **IN:** Checks if a value matches any value in a list or subquery result set. It processes all values.
5. **Explain the difference between logical and physical database design.**
   1. **Logical Design:** Focuses on the abstract structure of the database, including tables, relationships, and constraints.
   2. **Physical Design:** Focuses on the actual implementation of the database on the storage system, including indexing, partitioning, and storage optimization.
6. **What is the difference between a scalar subquery and a table subquery?**
   1. **Scalar Subquery:** Returns a single value.
   2. **Table Subquery:** Returns a result set that can be used as a table.
7. **What is the WITH clause?** The WITH clause, also known as Common Table Expressions (CTEs), allows you to define temporary result sets that can be referenced within a SELECT, INSERT, UPDATE, or DELETE statement.
8. **How do you create a temporary table?** Temporary tables are created using the CREATE TEMPORARY TABLE statement. Example:

CREATE TEMPORARY TABLE temp\_table (  
 id INT,  
 name VARCHAR(50)  
);

1. **What are derived tables?** Derived tables are subqueries in the FROM clause that provide a temporary result set for the main query.
2. **Explain indexing strategies for performance optimization.**
   1. Use indexes on columns frequently used in WHERE clauses.
   2. Use composite indexes for queries involving multiple columns.
   3. Avoid over-indexing, as it can slow down write operations.
   4. Regularly update statistics and rebuild indexes.

### **Advanced SQL Scenario-Based Questions**

1. **Write a query to find the second highest salary of employees.**

SELECT MAX(salary) AS SecondHighestSalary  
FROM employees  
WHERE salary < (SELECT MAX(salary) FROM employees);

1. **How would you find duplicate records in a table?**

SELECT column1, column2, COUNT(\*)  
FROM table\_name  
GROUP BY column1, column2  
HAVING COUNT(\*) > 1;

1. **How do you delete duplicate rows from a table without using a temporary table?**

DELETE FROM table\_name  
WHERE id NOT IN (  
 SELECT MIN(id)  
 FROM table\_name  
 GROUP BY column1, column2  
);

1. **Write a query to get the nth highest salary.**

SELECT DISTINCT salary  
FROM employees  
ORDER BY salary DESC  
LIMIT 1 OFFSET n-1;

1. **How do you find the employees who have not placed any orders?**

SELECT e.\*  
FROM employees e  
LEFT JOIN orders o ON e.employee\_id = o.employee\_id  
WHERE o.order\_id IS NULL;

1. **Given a table of student marks, write a query to find the top three scorers.**

SELECT student\_id, marks  
FROM student\_marks  
ORDER BY marks DESC  
LIMIT 3;

1. **Write a query to find the department with the highest number of employees.**

SELECT department\_id, COUNT(\*) AS employee\_count  
FROM employees  
GROUP BY department\_id  
ORDER BY employee\_count DESC  
LIMIT 1;

1. **How would you update a column value based on a condition in another table?**

UPDATE employees  
SET salary = salary \* 1.1  
WHERE department\_id IN (  
 SELECT department\_id  
 FROM departments  
 WHERE location = 'New York'  
);

1. **Write a query to display records with even row numbers.**

SELECT \*  
FROM (  
 SELECT \*, ROW\_NUMBER() OVER (ORDER BY column\_name) AS row\_num  
 FROM table\_name  
) AS temp  
WHERE row\_num % 2 = 0;

1. **Write a query to retrieve the last 5 records from a table.**

SELECT \*  
FROM table\_name  
ORDER BY id DESC  
LIMIT 5;

1. **How do you find the total salary department-wise?**

SELECT department\_id, SUM(salary) AS total\_salary  
FROM employees  
GROUP BY department\_id;

1. **How would you calculate running totals in a result set?**

SELECT employee\_id, salary,  
 SUM(salary) OVER (ORDER BY employee\_id) AS running\_total  
FROM employees;

1. **Write a query to reverse a string in SQL.**

SELECT REVERSE('your\_string') AS reversed\_string;

1. **How would you swap values of two columns in a single query?**

UPDATE table\_name  
SET column1 = column2,  
 column2 = column1;

1. **Write a query to get the cumulative sum in a result set.**

SELECT employee\_id, salary,  
 SUM(salary) OVER (ORDER BY employee\_id) AS cumulative\_sum  
FROM employees;

1. **Given a table of orders, write a query to find customers with more than five orders.**

SELECT customer\_id, COUNT(\*) AS order\_count  
FROM orders  
GROUP BY customer\_id  
HAVING COUNT(\*) > 5;

1. **Write a query to find the third last record in a table.**

SELECT \*  
FROM table\_name  
ORDER BY id DESC  
LIMIT 1 OFFSET 2;

1. **How do you find missing numbers in a sequence?**

SELECT t1.id + 1 AS missing\_number  
FROM table\_name t1  
LEFT JOIN table\_name t2 ON t1.id + 1 = t2.id  
WHERE t2.id IS NULL;

1. **How would you find the first non-null value in a list of columns?**

SELECT COALESCE(column1, column2, column3, 'default\_value') AS first\_non\_null  
FROM table\_name;

1. **Write a query to find the most frequently ordered product.** sql SELECT product\_id, COUNT(\*) AS order\_count FROM orders GROUP BY product\_id ORDER BY order\_count DESC LIMIT 1;

### **Complex SQL Scenario-Based Questions**

1. **Explain how to design a database for an online bookstore.**
   1. **Tables:** Books, Authors, Customers, Orders, OrderDetails, Categories, Publishers
   2. **Relationships:**
      1. Books to Authors (Many-to-Many)
      2. Books to Categories (Many-to-Many)
      3. Books to Publishers (Many-to-One)
      4. Orders to Customers (Many-to-One)
      5. Orders to OrderDetails (One-to-Many)
      6. OrderDetails to Books (Many-to-One)
   3. **Constraints:** Primary keys, foreign keys, unique constraints, and indexes for optimization.
2. **Describe steps to normalize a poorly designed table with redundant data.**
   1. **Step 1:** Identify repeating groups and create separate tables for them.
   2. **Step 2:** Ensure each table has a primary key.
   3. **Step 3:** Remove partial dependencies (2NF).
   4. **Step 4:** Remove transitive dependencies (3NF).
   5. **Step 5:** Apply BCNF if necessary.
3. **Given a sales table, how do you retrieve monthly sales trends?** ```sql SELECT DATE*FORMAT(sale*date, '%Y-%m') AS month