SQL theory questions and answers.

1. **What is SQL?**
   A. It stands for Structured Query Language, and it's a programming language used for interaction with relational database management systems (RDBMS). This includes fetching, updating, inserting, and removing data from tables.

2. **What are SQL dialects? Give some examples.**
   A. The various versions of SQL, both free and paid, are also called SQL dialects. All the flavors of SQL have a very similar syntax and vary insignificantly only in additional functionality. Some examples are Microsoft SQL Server, PostgreSQL, MySQL, SQLite, T-SQL, and Oracle.

3. **What are the main applications of SQL?**
   Using SQL, we can:

   - create, delete, and update tables in a database
   - access, manipulate, and modify data in a table
   - retrieve and summarize the necessary information from a table or several tables
   - add or remove certain rows or columns from a table
   - All in all, SQL allows querying a database in multiple ways. In addition, SQL easily integrates with other programming languages, such as Python or R, so we can use their combined power.

4. **What is an SQL statement? Give some examples.**
   Also known as an SQL command. It's a string of characters interpreted by the SQL engine as a legal command and executed accordingly. Some examples of SQL statements are SELECT, CREATE, DELETE, DROP, REVOKE, and so on.

5. **What types of SQL commands (or SQL subsets) do you know?**
   - Data Definition Language (DDL) – to define and modify the structure of a database.
   - Data Manipulation Language (DML) – to access, manipulate, and modify data in a database.
   - Data Control Language (DCL) – to control user access to the data in the database and give or revoke privileges to a specific user or a group of users.
   - Transaction Control Language (TCL) – to control transactions in a database.
   - Data Query Language (DQL) – to perform queries on the data in a database to retrieve the necessary information from it.

6. **Give some examples of common SQL commands of each type.**
   - DDL: CREATE, ALTER TABLE, DROP, TRUNCATE, and ADD COLUMN
   - DML: UPDATE, DELETE, and INSERT
   - DCL: GRANT and REVOKE

- TCL: COMMIT, SET TRANSACTION, ROLLBACK, and SAVEPOINT
- DQL: – SELECT

7. What is a database?

A structured storage space where the data is kept in many tables and organize that the necessary information can be easily fetched, manipulated, and summarized.

8. What is DBMS, and what types of DBMS do you know?

It stands for Database Management System, a software package used to perform various operations on the data stored in a database, such as accessing, updating, wrangling, inserting, and removing data. There are various types of DBMS, such as relational, hierarchical, network, graph, or object-oriented. These types are based on the way the data is organized, structured, and stored in the system.

9.What is RDBMS? Give some examples of RDBMS.

It stands for Relational Database Management System. It's the most common type of DBMS used for working with data stored in multiple tables related to each other by means of shared keys. The SQL programming language is designed to interact with RDBMS. Some examples of RDBMS are MySQL, PostgreSQL, Oracle, MariaDB, etc.

10. What are tables and fields in SQL?

A table is an organized set of related data stored in a tabular form, i.e., in rows and columns. A field is another term for a column of a table.

11. What is an SQL query, and what types of queries do you know?

A query is a piece of code written in SQL to access or modify data from a database.

There are two types of SQL queries: select and action queries. The first ones are used to retrieve the necessary data (this also includes limiting, grouping, ordering the data, extracting the data from multiple tables, etc.), while the second ones are used to create, add, delete, update, rename the data, etc.

10. What is a subquery?

Also called an inner query, a query placed inside another query, or an outer query. A subquery may occur in the clauses such as SELECT, FROM, WHERE, UPDATE, etc. It's also possible to have a subquery inside another subquery. The innermost subquery is run first, and its result is passed to the containing query (or subquery).

11. What types of SQL subqueries do you know?

- Single-row – returns at most one row.
- Multi-row – returns at least two rows.
- Multi-column – returns at least two columns.
- Correlated – a subquery related to the information from the outer query.
- Nested – a subquery inside another subquery.

12. **What is a constraint, and why use constraints?**

A set of conditions defining the type of data that can be input into each column of a table. Constraints ensure data integrity in a table and block undesired actions.

The different types of constrains are-

- DEFAULT – provides a default value for a column.
- UNIQUE – allows only unique values.
- NOT NULL – allows only non-null values.
- PRIMARY KEY – allows only unique and strictly non-null values (NOT NULL and UNIQUE).
- FOREIGN KEY – provides shared keys between two or more tables.

13. What is a join?

A clause used to combine and retrieve records from two or multiple tables. SQL tables can be joined based on the relationship between the columns of those tables.

14. What types of joins do you know?

- (INNER) JOIN – returns only those records that satisfy a defined join condition in both (or all) tables. It's a default SQL join.
- LEFT (OUTER) JOIN – returns all records from the left table and those records from the right table that satisfy a defined join condition.
- RIGHT (OUTER) JOIN – returns all records from the right table and those records from the left table that satisfy a defined join condition.
- FULL (OUTER) JOIN – returns all records from both (or all) tables. It can be considered as a combination of left and right joins.

15. What are the different types of `JOINS` in SQL? Explain the difference between `INNER JOIN` and `LEFT JOIN`?

Joins are used to combine rows from two or more tables based on a related column between them. The main types are `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, and `FULL OUTER JOIN`.

- **`INNER JOIN`**: when the tables are combined, returns only the rows that have matching values in both tables. It's the most common type.
- **`LEFT JOIN`**: when the tables are combined, returns all rows from the left table and the matching rows from the right table. If there is no match, `NULL` values are returned for columns from the right table.

16. What is a `PRIMARY KEY`? How is it different from a `FOREIGN KEY`?

A. A `PRIMARY KEY` uniquely identifies each record in a table. It must contain unique values and cannot be `NULL`. A table can only have one primary key.

A `FOREIGN KEY` is a field in one table that refers to the primary key of another table. It's used to establish a link between two tables. A table can have multiple foreign keys.

17. Explain the difference between `WHERE` and `HAVING`.

Both **`where`** and **`Having`** are used to filtering the data.

`Where` is used to filter individual rows based on a condition. It operates on the original dataset and is executed before any grouping.

`Having` is used to filter grouped data based on a condition. It operates on the results of an aggregation. It cannot be used without a GROUP BY clause.

18. What is a `UNION`? How is it different from `UNION ALL`?

Both `UNION` and `UNION ALL` are used to combine the result sets of two or more `SELECT` statements. The queries must have the same number of columns and same data types to combine.

`UNION` removes duplicate rows from the final result set.

`UNION ALL` keeps all rows, including duplicates, making it faster and more efficient as it doesn't have to perform a de-duplication step.

19. What is the purpose of the `GROUP BY` clause?

A. The `GROUP BY` clause is used to arrange identical data into groups. It is most often used with aggregate functions like `SUM()`, `COUNT()`, and `AVG()` to perform calculations on each group of data rather than on the entire dataset.

Example: select ID, sum(amount) as total_sales from sales group by ID;

20. What is a `subquery`? When would you use one?

A. A subquery, or inner query, is a query nested inside another query. It's used when a query needs a value that can only be determined by another query, often to return a single value or a list of values to be used in the outer query's `WHERE` or `FROM` clauses.

21. What is a Common Table Expression (CTE)? Why might you use a CTE instead of a subquery?

A. CTE is common table expression, it is a named temporary result set that you can reference within a `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statement. They are often used to make complex queries more readable and can be referenced multiple times within a single query, unlike a subquery.

22. What is the difference between `DELETE` , `TRUNCATE` and drop ?

A. Delete, truncate, and drop are all used to remove rows of data from the database.

- Delete removes all the rows specified in the where clause. It does not remove the table structure. It can usually be rolled back.
- Truncate removes all the rows of the table. It does not remove the table structure. Once truncated it cannot be rolled back. Truncate is faster than delete
- Drop removes all the rows as well as the table structure. It deletes the whole table and cannot be rolled back.

23. What is a `View`? What is the benefit of using one?

A. A view in SQL is a virtual table that is based on the result of a stored query. It doesn't store the actual data itself but presents the data stored in one or more tables in a structured way. Views are great for simplifying complex queries and restricting access to sensitive data.

24. What is an `index` in a database, and how does it improve query performance?

A. An index is a data structure that improves the speed of data retrieval operations. It's like a physical index in a book; it allows the database engine to quickly find the data without having to scan the entire table, which is very useful for optimizing queries with `WHERE` or `JOIN` clauses. There are two main types:

- Clustered Index: Reorders the physical storage of the table's rows to match the order of the index. A table can only have one clustered index.
- Non-clustered Index: Does not change the physical order of the rows; it's a separate object that stores the column value and a pointer to the row's location. A table can have multiple non-clustered indexes.

25. Explain the difference between `ROW_NUMBER()`, `RANK()`, and `DENSE_RANK()`.

A. **rank()**-- rank is used for giving each row a numerical value based on a parameter. if there are repetions of the value, then, the rank is skipped. 1 2 2 4

dense_rank()-- this is also similar to rank() gives each row a numercial value based on another value. however when the data repeates, dense_rank does not skip the rank 1 2 2 3

row_number()-- each row is given a numerical value irrespective of the value. it is used when we want a serial number for the data. it does not skip values, gives unique number to each row 1 2 3 4

26. What is a `window function`? How is it different from an aggregate function?

A. window functions are functions used to create calculations on the values given for the specified rows. while aggreegate functions groups the values, window functions does not group it, but keeps it as induvidual rows.

27. How would you handle duplicate records in a query?

A. we can either remove all the duplicates if the data does not skew. or we can keep them as it is if required.

You can identify duplicates by grouping data and using `COUNT(*) > 1`. To remove them, you can use a CTE with `ROW_NUMBER()` to assign a unique number to each row within a partition. Then, you can delete or select only the rows where `ROW_NUMBER()` equals 1.

If you simply want to remove duplicates we can use the `DISTINCT` keyword.

28. What are `NULL` values, and how do you handle them in SQL queries?

A. NULL values in SQL represent missing or unknown data. They are not the same as empty strings or zero, they mean the value is simply not available.

To handle NULLs, we have different approaches depending on the requirement:

- If you want to **replace NULLs with a default value**, we can use functions like `COALESCE()` or `IFNULL()`.
- If you want to **filter out rows with NULLs**, you use conditions like `WHERE column IS NOT NULL`.
- If you want to specifically identify or compare NULLs, you can use `IS NULL` or `IS NOT NULL` since `= NULL` doesn't work.
- In calculations or reporting, you can use `NULLIF()` to return `NULL` when two values are equal, or replace NULLs before performing aggregations.
- From a data modeling perspective, you can also try to minimize unnecessary NULLs by applying constraints like `NOT NULL` where appropriate.

29. What is the difference between full outer join and cross join?

A. **FULL OUTER JOIN** returns all the rows from both tables, matching them where possible, and filling in NULLs where there is no match. In other words, it combines

the results of a LEFT JOIN and a RIGHT JOIN. It is useful when you want to see all records from both tables regardless of whether they match.

A **CROSS JOIN**, on the other hand, returns the **Cartesian product** of the two tables—every row from the first table is combined with every row from the second table. It doesn't require any join condition and is rarely used in practical scenarios, except when generating combinations or test data.

30. What is a primary key?

A column (or multiple columns) of a table to which the PRIMARY KEY constraint was imposed to ensure unique and non-null values in that column. In other words, a primary key is a combination of the NOT NULL and UNIQUE constraints. The primary key uniquely identifies each record of the table. Each table can define at most one PRIMARY KEY (which may be composite). A PRIMARY KEY is strongly recommended but not strictly required by all engines.

31. What is a unique key?

A column (or multiple columns) of a table to which the UNIQUE constraint was imposed to ensure unique values in that column, including a possible NULL value (the only one).

32. What is a foreign key?

A column (or multiple columns) of a table to which the FOREIGN KEY (or UNIQUE key) constraint was imposed to link this column to the primary key in another table (or several tables). The purpose of foreign keys is to keep connected various tables of a database.

33. What is a schema?

A collection of database structural elements such as tables, stored procedures, indexes, functions, and triggers. It shows the overall database architecture, specifies the relationships between various objects of a database, and defines different access permissions for them.

34. What is a SQL operator?

A reserved character, a combination of characters, or a keyword used in SQL queries to perform a specific operation. SQL operators are commonly used with the WHERE clause to set a condition (or conditions) for filtering the data.

35.**What types of SQL operators do you know?**

- Arithmetic (+, -, *, /, etc.)
- Comparison (>, <, =, >=, etc.)
- Compound (+=, -=, *=, /=, etc.)
- Logical (AND, OR, NOT, BETWEEN, etc.)
- String (%, _, +, ^, etc.)
- Set (UNION, UNION ALL, INTERSECT, and MINUS (or EXCEPT))

36.What is an alias?

A temporary name given to a table (or a column in a table) while executing a certain SQL query. Aliases are used to improve the code readability and make the code more compact. An alias is introduced with the AS keyword

37.What is a clause?

A condition imposed on a SQL query to filter the data to obtain the desired result. Some examples are WHERE, LIMIT, HAVING, LIKE, AND, OR, ORDER BY, etc.

38.What are some common statements used with the SELECT query?

The most common ones are FROM, GROUP BY, JOIN, WHERE, ORDER BY, LIMIT, and HAVING.

39.What is the DISTINCT statement and how do you use it?

This statement is used with the SELECT statement to filter out duplicates and return only unique values from a column of a table.

40.What are relationships? Give some examples.

Relationships are the connections and correlations between entities, basically meaning how two or more tables of a database are related to one another. For example, we can find an ID of the same client in a table on sales data and in a customer table.

41.What is a NULL value? How is it different from zero or a blank space?

A NULL value indicates the absence of data for a certain cell of a table. Instead, zero is a valid numeric value, and an empty string is a legal string of zero length.

42.What is the difference between SQL and NoSQL?

SQL databases are relational, structured, and use tables with predefined schemas, while NoSQL databases are non-relational, schema-less, and designed to handle unstructured or semi-structured data.

43. What are some common challenges when working with SQL databases?

Challenges include performance tuning for large datasets, managing indexing strategies, ensuring data integrity with constraints, handling concurrent transactions, and optimizing query execution.

44. **What is a function in SQL, and why use functions?**

A database object representing a set of SQL statements frequently used for a certain task. A function takes in some input parameters, performs calculations or other manipulations on them, and returns the result. Functions help improve code readability and avoid repetition of the same code snippets.

45.What types of SQL functions do you know?

- Aggregate functions – work on multiple, usually grouped records for the provided columns of a table, and return a single value (usually by group).
- Scalar functions – work on each individual value and return a single value.

46. What aggregate functions do you know?

- AVG() – returns the average value
- SUM() – returns the sum of values
- MIN() – returns the minimum value
- MAX() – returns the maximum value
- COUNT() – returns the number of rows, including those with null values
- FIRST() – returns the first value from a column
- LAST()– returns the last value from a column

47.What is a stored procedure, and how is it different from a function?

A stored procedure is a precompiled set of SQL statements executed as a unit to perform a task. Procedures can modify data or schema objects, manage transactions, and return zero or more result sets. Functions, on the other hand, are typically used in SQL expressions, must return a value (scalar or table-valued), and in many databases are restricted from side effects. Exact behavior differs by DB (e.g., T-SQL has scalar and table-valued functions; PostgreSQL distinguishes between functions and procedures).

48. What set operators do you know?

- UNION – returns the records obtained by at least one of two queries (excluding duplicates)
- UNION ALL – returns the records obtained by at least one of two queries (including duplicates)
- INTERSECT – returns the records obtained by both queries
- EXCEPT (called MINUS in MySQL and Oracle) – returns only the records obtained by the first query but not the second one

49. What operator is used in the query for pattern matching?

The LIKE operator in combination with the % and _ wildcards. The % wildcard represents any number of characters including zero, while _ – strictly one character.

50.What is a composite primary key?

The primary key of a table, based on multiple columns.

51. **In which order does the interpreter execute the common statements in the SELECT query?**

Here is the SQL order of execution:
FROM → ON → JOIN → WHERE → GROUP BY → HAVING → SELECT → ORDER BY → LIMIT/OFFSET (FETCH)

52. What types of SQL relationships do you know?

- One-to-one – each record in one table corresponds to only one record in another table
- One-to-many – each record in one table corresponds to several records in another table
- Many-to-many – each record in both tables corresponds to several records in another table

53. What is the difference between nested and correlated subqueries?

A correlated subquery is an inner query nested in a bigger (outer) query that refers to the values from the outer query for its execution, meaning that a correlated subquery depends on its outer query. Instead, a non-correlated subquery doesn't rely on the data from the outer query and can be run independently of it.

54. What is the CASE() function?

The way to implement the if-then-else logic in SQL. This function sequentially checks the provided conditions in the WHEN clauses and returns the value from the corresponding THEN clause when the first condition is satisfied. If none of the conditions is satisfied, the function returns the value from the ELSE clause in case it's provided, otherwise, it returns NULL.

55.Explain the GROUP BY clause and provide an example.

The GROUP BY clause is used to aggregate rows that have the same values in specified columns.

Example: "SELECT product_category, SUM(amount) FROM sales GROUP BY product_category;"

56. How do you sort the results of a query?

Use the ORDER BY clause to sort results in ascending or descending order.

Example: "SELECT * FROM sales ORDER BY amount DESC;"

57. Explain the concept of a CROSS JOIN.

A CROSS JOIN returns the Cartesian product of two tables, where each row in the first table is combined with each row in the second table.

Example: "SELECT * FROM products CROSS JOIN categories;"

58. How do you handle NULL values in SQL queries?

Use functions like IS NULL, IS NOT NULL, or COALESCE() to handle NULL values.

Example: "SELECT COALESCE(discount, 0) FROM sales;"

59. What is the LIMIT clause used for? Provide an example.

The LIMIT clause restricts the number of rows returned by a query.

Example: "SELECT * FROM sales LIMIT 10;"

60. What is a window function? Provide an example.

Window functions perform calculations across a set of table rows related to the current row, such as running totals or rankings.

Example: "SELECT product_name, amount, RANK() OVER (ORDER BY amount DESC) AS rank FROM sales;"

61.How do you calculate a running total in SQL?

Use the SUM() window function with an appropriate OVER clause.

Example: "SELECT order_date, amount, SUM(amount) OVER (ORDER BY order_date) AS running_total FROM orders;"

62.How do you use the EXISTS clause in SQL?

The EXISTS clause checks if a subquery returns any rows.

Example: "SELECT product_name FROM sales WHERE EXISTS (SELECT * FROM returns WHERE returns.product_id = sales.product_id);”

63. What are some common performance optimization techniques in SQL?

Common techniques include using indexes, optimizing queries, reducing the use of subqueries, and ensuring efficient joins.

64.What is the purpose of the COALESCE() function?

The COALESCE() function returns the first non-NULL value from a list of arguments.
Example: "SELECT COALESCE(discount, 0) FROM sales;"

65.Explain the DATEPART() function with an example.

DATEPART() extracts a specific part (e.g., year, month) from a date.
Example: "SELECT DATEPART(year, sale_date) AS sale_year FROM sales;"

66.What is a Recursive Query?

A recursive query is a query that calls itself. In SQL, recursive queries are typically written using a Common Table Expression (CTE). They are useful for hierarchical data, like organizational charts or folder structures.

67. What distinguishes CHAR from VARCHAR?

CHAR is a fixed-length data type, while VARCHAR is variable-length.CHAR adds spaces to the string to reach the specified length, while VARCHAR only allocates space for the actual length of the string.

68.What is a database warehouse?

A database warehouse is a centralized repository that stores data from various sources for analytical and reporting purposes. It is optimized for querying and analysis and often contains historical data.

69.What are the challenges in handling big data in SQL?

Handling big data in SQL involves dealing with large volumes of data that exceed the capabilities of traditional database systems. Challenges include data storage, processing, scalability, and efficient querying. Solutions may include distributed databases and big data technologies like Hadoop and Spark.

70. What is a star schema vs. snowflake schema?

- Star Schema has a central fact table connected directly to denormalized dimension tables. It is simple, fast for queries, but uses more storage.
- Snowflake Schema normalizes the dimension tables into multiple related tables. It saves storage and improves data integrity but requires more joins, making queries slower.
- Star is best for simplicity and performance; Snowflake is best for large warehouses needing space efficiency and strict normalization.