SQL Questions:

1. In SQL, there are 4 main types of joins:

1.Inner Join:

This is the most common type of join in SQL, it also known as join and it returns only the matching rows from both tables. You use an inner join when you want to retrieve the data that is common in both tables.

Example: If you have a customer table and an orders table, and you want to see both customer and order details together, you'd use an inner join.

2. Full Join:

A full join returns all the rows from both tables, with matching rows where possible. For any unmatched rows, it returns NULL.

Example: If you want to see all customer records and their orders, even if some customers haven't made any orders, you'd use a full join.

3. Left Join:

A left join returns all rows from the left table and the matching rows from the right table. If there are no matches, it will return NULL for the right table.

Example: If you want to see all customer data regardless of whether they have placed an order, you'd use a left join, selecting the customer table first, followed by the orders table.

4.Right Join:

This works opposite to the left join. It returns all rows from the right table and the matching rows from the left table.

Example: If you want to see all order details, even if some orders don't have corresponding customer data, you'd use a right join.

Additionally, there is:

5.Cross Join:

A cross join returns the Cartesian product of both tables, meaning it pairs every row from one table with every row from the other.

This is rarely used in practice.

2. Explain the difference between the WHERE and HAVING clauses in SQL.

These both clauses use to do filter in SQL, but they are applied at different stages.
The WHERE clause is used to filter rows before grouping, while the HAVING clause is used to filter rows after grouping.

3.What is the purpose of a primary key in a database?

A primary key is a constraint in SQL used for data validation in a table. It ensures that the values in the primary key column are unique and do not contain any NULL values. SQL also creates an index on the primary key to help speed up the data iteration. It helps to create relation with another table on foreign key

4. How do you optimize a slow query in SQL?

To optimize a slow SQL query, I follow these 5 basic steps:

1. Check the indexes: Indexes help speed up data retrieval.
2. Select specific columns: Instead of selecting all columns from a table, choose only the columns you need. For example, if you only need the `customer_id` and `order_id`, select just those columns and join the tables using the primary key and foreign key.
3. Optimize the joins: Make sure the joins between the tables are efficient.
4. Use LIMIT: If you only need a specific number of rows, use the LIMIT clause.

Finally, analyze the execution plans to check the query's performance.

5. What are aggregate functions in SQL? Can you provide examples of commonly used aggregate functions?

Here is a list of commonly used aggregate functions in SQL and their use:

SUM() - Adds up all the values.

COUNT() - Counts the number of rows or non-null values.

AVG() - Calculates the average of values.

MIN() - Finds the smallest value.

MAX() - Finds the largest value.

6. What is a subquery in SQL, and how is it different from a JOIN? Can you give an example of when to use a subquery?

A subquery is a query that is nested inside another SQL query, typically within a SELECT, INSERT, UPDATE, or DELETE statement. It is used to return data that will be used by the main query. Subqueries can be used in places where joins might not be suitable or if we need to perform a calculation or filter based on dynamic data.

Key Differences Between Subquery and JOIN:

Subquery: A subquery is used when you need to return a single value or a set of values to be used in the main query. It works within a statement (like SELECT, WHERE, etc.).

JOIN: A JOIN combines rows from two or more tables based on a related column between them. It works by bringing data from different tables together in one result set.

7. What is the difference between DELETE and TRUNCATE in SQL?

DELETE and **TRUNCATE** both remove rows from a table, but they work differently. DELETE allows you to remove specific rows using a condition and can be rolled back. However, it's slower. TRUNCATE is much faster and removes all rows from the table at once, but the data cannot be rolled back after truncation.

8. What is the difference between a UNION and a UNION ALL in SQL?

Both UNION and UNION ALL are used to combine the result sets of two or more SELECT statements. However, the key difference lies in how they handle duplicate values.

Duplicates: UNION removes duplicates, while UNION ALL keeps all rows, including duplicates.
Performance: UNION ALL is faster since it doesn’t need to remove duplicates.

9. What are indexes in SQL, and why are they used?

Indexes in SQL are special data structures that improve the speed of data retrieval operations on a database table
An index creates a sorted copy of specific columns in a table. When you run a query, the database looks at the index first to find the exact rows you need instead of scanning all rows.

10. What is normalization in SQL, and why is it important? Can you explain different normal forms?

Normalization is a technique in SQL and database design that organizes data to minimize redundancy and improve data integrity. It involves structuring a relational database in such a way that the relationships between data are clearly defined and the data is stored efficiently.

Reduce Data Redundancy: Avoid storing the same data in multiple places.
Improve Data Integrity: Ensure that the data is accurate and consistent.
Simplify Data Management: Make it easier to maintain and update data.

and there are three forms in normalization.
first normal forms
second normal forms
third normal form

12. What is a view in SQL, and how is it different from a table?

A view is like a virtual table that doesn't store data physically. Instead, it gets data from the real tables each time you use it. Because views don’t hold data, they need less storage space, but the underlying tables must still exist. Views also make it easier to work with complex queries by combining them into one object, so you don’t have to write complicated SQL statements repeatedly.

13. What is the difference between a primary key and a foreign key in SQL?

Primary Key and **Foreign Key** are both types of constraints in SQL that enforce data integrity in a relational database, but they serve different purposes.

Primary key: A primary key is a unique identifier for each record in a table. It ensures that each entry in the column (or combination of columns) is unique and not null.

Foreign key: A foreign key is a column (or a group of columns) in one table that refers to the primary key in another table. It establishes a relationship between the two tables.

14. What are the different types of SQL constraints? Can you explain each type?

We can think of a constraint as a rule for columns in a database. Here are a few types of constraints in SQL:

1. Primary Key: Ensures that a column has unique values, cannot be NULL, and establishes a relationship with a foreign key in another table.
2. Foreign Key: Creates a relationship with the primary key of another table.
3. Unique Key: Allows only unique values in a column and can accept NULL values.
4. Default Constraint: Sets a default value for a column when no value is provided.
5. Check Constraint: Ensures that the values in a column meet a specified condition.

15. What is the difference between a clustered index and a non-clustered index in SQL?

A clustered index is like an organized bookshelf where the books (data) are arranged in a specific order based on their titles (the index). There can only be one way to organize the shelf, so this type of index changes how the data is stored.

A non-clustered index is like a list of books in a library. The list tells you where to find the books but doesn't change how they are arranged on the shelves. You can have many lists (non-clustered indexes) pointing to the same books without changing their order.

16. What is a transaction in SQL, and what are the properties of a transaction (ACID)?

A transaction in SQL is a sequence of one or more operations that are executed as a single unit. It ensures that either all operations are completed successfully, or none are applied, maintaining data integrity. Transactions have four key properties known as ACID:

Atomicity: This means that all operations in a transaction must complete successfully. If any operation fails, the entire transaction is rolled back, and no changes are made to the database.

Consistency: A transaction must take the database from one valid state to another, ensuring that all data rules and constraints are followed. This guarantees that the database remains in a consistent state before and after the transaction.

Isolation: Transactions should operate independently of each other. This means that the changes made by one transaction should not be visible to others until the transaction is completed. This prevents issues like dirty reads or lost updates.

Durability: Once a transaction is committed, the changes it made are permanent, even if there's a system failure. The database ensures that committed data will not be lost.

17. What is the purpose of the GROUP BY clause in SQL? Can you provide an example?

The GROUP BY clause in SQL is used to summarize data by grouping rows that have the same values in specified columns. It helps in aggregating data to make it easier to analyze.

For example, if we want to know the number of customers in each city, we can select the city and count the customers, then use the GROUP BY clause to group the results by city.

18. What are stored procedures in SQL, and what are their advantages?

A stored procedure in SQL is a set of saved SQL statements that can be executed as a single unit. They are useful for running the same queries repeatedly, saving time and effort.

Advantages of stored procedures include:

- Reusability: They can be reused without rewriting code.

- Performance: They are precompiled, which makes them faster.

- Security: They control access to tables, enhancing security.

- Maintainability: Changes can be made without affecting application code.

- Reduced Network Traffic: Multiple SQL statements can be executed in one call.