DBMS &SQL Interview Question

1. What Do You Mean by Database?

This is one of the most common SQL interview questions being asked. A database is an organized collection of structured data that can be stored and accessed on a computer system. It is managed by software called a Database Management System (DBMS), which efficiently handles and manages the data.

2. What Is a Relational Database?

In a relational database, data is systematically arranged with specific relationships for easy access. The model distinguishes between data structures (tables, indexes, views) and physical storage so administrators can change storage settings without disrupting the logical organization of data.

3. What Is RDBMS?

This question is one of the most commonly listed and asked SQL interview questions for beginners. RDBMS, or Relational Database Management System, is software designed to manage relational databases. It organizes data into tables with rows and columns, allowing for efficient storage and retrieval. Examples include SQL, MySQL, and Oracle.

4. What Are the Differences Between SQL and MYSQL?

Differentiating between SQL and MySQL is a topic frequently covered in SQL interview questions. Below are the key differences between SQL and MySQL:

Aspect	SQL	MySQL
Definition	Structured Query Language.	An open-source Relational Database Management System.
Type	A language	A software application
Purpose	To manage and query relational databases.	To implement SQL and manage databases.
Scope	The standard language for all relational databases.	A specific RDBMS implementation.
Origin	Developed by IBM in the 1970s	Created by MySQL AB in 1995, now owned by Oracle.
Usage	Used across various database systems.	A specific database system that uses SQL.
Functionality	Defines operations like SELECT, INSERT, UPDATE, and DELETE.	Provides a platform to execute SQL commands and manage databases.
Customization	Standard syntax with minor variations.	Has its own extensions and specific features.

Licensing	Not applicable (it's a language).	Open-source with commercial options available.
Performance	Not applicable (performance depends on implementation).	Known for high performance and reliability.

5. What Do the Terms Table and Field Refer to in SQL?

The term table and field in SQL are referred to as:

- A table in SQL is a structured collection of related data organized into rows (records) and columns (fields). Each row represents an individual record, and each column represents an attribute of that record.
- A field refers to a specific column in a table containing individual pieces of data for each record. It has a defined data type (e.g., integer, string) and constraints.

6. Can You Explain What Relationships Exist Between Tables and List Their Types?

Relationships between tables in a relational database refer to the logical connections between tables based on common data fields. These relationships allow for data integrity, efficient querying, and the ability to retrieve related information across multiple tables.

There are three main types of relationships between tables:

- **One-to-One (1:1) Relationship:**
    - Each record in the first table corresponds to exactly one record in the second table, and vice versa.
- **One-to-Many (1:N) Relationship:**
    - A record in one table may correspond to multiple records in another table, but each record in the second table is associated with only one record in the first table.
    - This is the most common type of relationship.
- **Many-to-Many (M:N) Relationship:**
    - Multiple records in one table can be related to multiple records in another table.
    - This type of relationship is implemented using a junction table (also called a bridge table or associative entity).

7. What Is an Entity in a Database?

An entity in database design represents a distinct object, concept, or item from the real world about which we want to store information. It's essentially any "thing" that can be uniquely identified and described.

Entities can be:

- **Concrete:** Physical objects like a person, car, or building.
- **Abstract:** Concepts or events such as a course, transaction, or appointment.

In a database, entities become tables, with each instance represented as a row in that table. The attributes of the entity become the columns of the table.

8. What Is an Attribute?

You might encounter this topic in SQL interview questions. In SQL, an attribute is a key concept you'll need to understand. It represents a specific characteristic or property of the data stored in a table. Think of an attribute as a column in the table, where each column defines what kind of information can be recorded.

Each attribute has a unique name and is associated with a data type, such as text, numbers, dates, or binary data. This data type tells you what kind of information the attribute can hold. Knowing how attributes work will help you grasp how data is organized and queried in SQL.

9. What Are the Types of Attributes?

The term table and field in SQL are referred to as:

- **Simple Attributes:** These are atomic attributes that cannot be divided further. Examples include Age, Gender, and Phone Number.
- **Composite Attributes:** These can be broken down into smaller subparts, each representing independent attribute values. For example, an Address can be divided into Street, City, State, and ZIP.
- **Single-valued Attributes:** These hold only one value for each entity instance. Examples include Social Security Number and Date of Birth.
- **Multi-valued Attributes:** These can hold multiple values for a single entity instance. Examples include Phone Numbers (a person may have multiple) and Skills.
- **Derived Attributes:** These are calculated from other attributes rather than explicitly stored. Examples include Age (derived from Date of Birth) and Total Order Amount (derived from individual item prices).
- **Key Attributes:** These uniquely identify an entity instance within the entity set. Examples include Student ID and Employee Number.
- **Null Attributes:** These can have a null value, indicating that the data is unknown or not applicable. Examples include Middle Name (not everyone has one).

10. What Are the Differences Between the CHAR and VARCHAR2 Data Types in SQL?

When working with string data in SQL, it's essential to understand the differences between the CHAR and VARCHAR2 data types. Both are used to store character strings but handle data storage and manipulation differently.

The following are the differences between CHAR and VARCHAR2 data types in SQL.

Aspect	CHAR	VARCHAR2
Storage	Fixed-length	Variable-length
Maximum size	Two thousand bytes (Oracle), 8000 characters (SQL Server).	Four thousand bytes (Oracle), 8000 characters (SQL Server).
Padding	Padded with spaces to the defined length.	No padding
Storage efficiency	Less efficient for variable-length data.	More efficient for variable-length data

Performance	Slightly faster for fixed-length data.	Slightly slower due to the length indicator.
Usage	Best for fixed-length data (e.g., state codes).	Best for variable-length data (e.g., names, addresses)
Trailing spaces	Retained	Trimmed when stored
Comparison behavior	Blank-padded comparison.	Nonpadded comparison.
Memory allocation	Always uses fully defined length.	Uses only the space needed plus a small overhead.
Indexing	It can be slightly faster.	It can be slightly slower due to variable length.

11. Explain SQL Server and Its Core Components

SQL Server is a relational database management system (RDBMS) developed by Microsoft, designed to store, retrieve, and manage structured data. Its core components are:

- **Database Engine:** Manages data storage, processing, and security, providing controlled access and rapid transaction processing.
- **Relational Engine:** Handles SQL query processing, including syntax checking, query normalization, query optimization, and execution.
- **Storage Engine:** Manages physical data storage and retrieval, using memory buffers for efficient access, and ensures data consistency and reliability through ACID properties.
- **SQLOS:** Provides services for thread management, memory allocation, and input/output operations to support the Database Engine.

12. Explain Transaction in SQL Server and Its Modes

A transaction in SQL Server is a logical unit of work that includes one or more database operations, treated as a single operation to ensure data integrity and consistency. Transactions follow the ACID properties: Atomicity, Consistency, Isolation, and Durability.

SQL Server supports the following transaction modes:

- **Autocommit Transactions:** Each T-SQL statement is treated as a transaction and is automatically committed or rolled back upon completion.
- **Explicit Transactions:** Defined by the user with a BEGIN TRANSACTION statement and ended with either a COMMIT or ROLLBACK statement.
- **Implicit Transactions:** Enabled with SET IMPLICIT_TRANSACTIONS ON, where a new transaction begins automatically after the previous one completes and must be explicitly committed or rolled back.

13. What Are User-Defined Functions?

User-defined functions in PL/SQL or Java can add functionality that SQL or its built-in functions lack. Just like SQL functions, these functions can be placed anywhere expressions are used.

14. What Is a Transaction Log, and Why Is It Important?

A transaction log is a component of a database that tracks all actions executed by the database management system. It records all changes made to the database to support data integrity and recovery.

The transaction log is important because:

- It ensures atomicity by allowing the rollback of changes if a transaction fails.
- It enables recovery of the database to a consistent state after a system crash or power failure.
- It allows the database to be restored to a specific point in time for disaster recovery.
- It supports replication by identifying and transferring changes to subscribing databases.
- It helps in synchronizing secondary copies of the database in technologies like Always On Availability Groups and Database Mirroring.

15. Explain ACID Properties

ACID properties are characteristics that ensure reliable processing of database transactions:

- **Atomicity:** A transaction is treated as a single, indivisible unit. All operations must be completed successfully, or none will be applied. If any part fails, the entire transaction is rolled back.
- **Consistency:** A transaction brings the database from one valid state to another valid state. All data must comply with defined rules and constraints, maintaining database consistency before and after the transaction.
- **Isolation:** Concurrent transactions must not affect each other. Each transaction should appear to execute in isolation from others, even if they are processed simultaneously.
- **Durability:** Once a transaction is committed, it remains so, even in the case of power loss, crashes, or errors. The committed data is preserved and available after a system failure or restart.

16. What Do You Mean by a Stored Procedure in a Database?

A stored procedure is a group of SQL queries saved and reused multiple times. It performs one or more DML (Data Manipulation Language) operations on a database, accepts input parameters, performs tasks, and optionally returns values.

Syntax for creating a stored procedure:

```
CREATE PROCEDURE procedure_name (parameter1 data_type, parameter2 data_type, ...)
AS
BEGIN
   -- SQL statements to be executed
END
```

Syntax for executing a stored procedure:

```
EXEC procedure_name parameter1_value, parameter2_value, ..
```

17. Can You List the Various Types of User-Defined Functions?

The main types of user-defined functions are:

- Scalar functions.
- Table-valued functions.
- Aggregate functions.
- Inline table-valued functions.
- Multi-statement table-valued functions.

18. What Is the Purpose of the ALIAS Command in SQL?

The purpose of ALIAS command in SQL are:

- Rename columns.
- Rename tables.
- Create temporary names for columns or tables.
- Simplify complex queries.
- Improve query readability.

19. What Is a Recursive Stored Procedure?

A recursive stored procedure is a stored procedure that includes a call to itself within its own body. This allows it to perform operations on hierarchical or tree-structured data or to address problems that can be broken down into similar, smaller tasks.

In simpler terms, it repeats its process to solve complex problems more efficiently by dividing them into smaller parts. This approach is useful in scenarios like traversing organizational charts or directory structures.

20. Explain Database Normalization and Denormalization

Below is a detailed explanation of Normalisation and Denormalisation.

Database Normalization: It is the process of designing a database schema to reduce redundancy and improve data integrity. It involves organizing data into tables and defining relationships to minimize duplication and dependencies. Key normal forms include:

- **First Normal Form (1NF):** Eliminates repeating groups.
- **Second Normal Form (2NF):** Removes partial dependencies.
- **Third Normal Form (3NF):** Eliminates transitive dependencies.
- **Boyce-Codd Normal Form (BCNF):** A stricter version of 3NF.
- **Fourth Normal Form (4NF):** Deals with multivalued dependencies.
- **Fifth Normal Form (5NF):** Addresses join dependencies.

Database Denormalization: It is the process of intentionally introducing redundancy into a database to improve performance. It simplifies queries and speeds up data retrieval by incorporating redundant data. Techniques include:

- Introducing redundant columns.
- Using mirrored tables.
- Table splitting.
- Adding derived columns.
- Creating materialized views.

21. What Is a JOIN, and Mention Its Types?

SQL Joins are used to obtain data from multiple tables by applying a join condition. This condition establishes a relationship between columns in the data tables participating in the join. The tables in a database are interconnected through SQL keys, and these key relationships are utilized in SQL Joins to fetch and combine the relevant data.

Types of joins:

Join Type	Description
INNER JOIN	Returns only the matching rows from both tables.
LEFT (OUTER) JOIN	Returns all rows from the left table and matching rows from the right table.
RIGHT (OUTER) JOIN	Returns all rows from the right table and matching rows from the left table.
FULL (OUTER) JOIN	Returns all rows when there's a match in either the left or right table.
CROSS JOIN	Returns the Cartesian product of both tables (all possible combinations).
SELF JOIN	Joins a table to itself.

22. What Is Subquery?

A subquery is a query within another SQL query. It returns data that will be used in the main query as a condition to restrict the data further to be retrieved.

Subqueries can be used in various parts of a SQL statement:

- In the SELECT clause.
- In the FROM clause.
- In the WHERE clause.
- In the HAVING clause.

23. What Are the Types of Subquery in SQL?

SQL Server supports various types of subqueries based on their usage and the type of data they return. Here are some common types:

- **Scalar Subqueries:**

A Scalar subquery returns a single data value consisting of one row and one column.

- **Single or Multiple Row Subqueries:**
    - **Single Row:** This subquery returns a single row of data or multiple rows of values from the subquery specified in the outer query's 'WHERE' clause.
    - **Multiple Row:** Returns multiple rows from the inner query.
- **Correlated Subqueries:**

A correlated subquery references columns from the outer query and is executed repeatedly for each row of the outer query.

- **Nested Subqueries:**

Nested subqueries involve SQL SELECT queries within other subqueries. SQL Server supports up to 32 levels of nested subqueries in a single outer or main query statement.

24. What Are the Differences Between a Primary Key and a Unique Key?

Here's the difference between a Primary Key and a Unique Key:

Characteristic	Primary Key	Unique Key
Purpose	Uniquely identifies each record in a table.	Ensures uniqueness of values in a column or set of columns.
Nullability	It cannot contain NULL values.	It can contain NULL values (usually one NULL per column).
Number per table	Only one per table	Multiple unique keys are allowed per table.
Index	Automatically creates a clustered index (by default).	Creates a non-clustered index.
Foreign Key reference	A foreign key can reference it.	A foreign key can reference it.
Uniqueness	Ensures both uniqueness and identity.	Ensures uniqueness but not necessarily identity.
Default constraint	It cannot have a default constraint.	It can have a default constraint.
Modification	is not modified once set.	It can be modified more freely.
Use in relationships	The primary method for defining relationships between tables.	It can be used in relationships, but it is less common.
Implicit constraints	Implies both NOT NULL and UNIQUE constraints.	It only implies a UNIQUE constraint.

25. What Is Data Integrity?

Data integrity refers to the accuracy, consistency, and reliability of data stored in a database. It ensures that data remains unchanged and valid throughout its lifecycle unless modified by authorized processes.

To maintain data integrity, databases follow specific rules or constraints set by the DBMS, such as unique keys, foreign keys, and data validation checks. These mechanisms help ensure the data remains correct, consistent, and trustworthy, preserving its integrity over time.

26. What Are the Defaults in the SQL Server?

In SQL Server, defaults are predefined values automatically applied to a column when no value is provided during an INSERT operation. They ensure that a column has a value, even if the user does not explicitly provide one. Defaults can be defined using the DEFAULT constraint during table creation or added later to a column.

27. What Are Cursors, and Mention Their Types?

A cursor is a temporary workspace or memory area allocated by the database server during DML (Data Manipulation Language) operations on a table conducted by the user. It is used to handle and store data from database tables.

There are two main types of cursors:

- **Implicit Cursors:** Known as default cursors in SQL Server, Implicit Cursors are allocated by SQL Server automatically when users perform DML (Data Manipulation Language) operations.
- **Explicit Cursors:** Explicit Cursors are user-defined and created as needed. They are utilized to fetch data from a table, row by row.

28. What Is Auto Increment?

Auto Increment is a feature used in database design to automatically generate a unique numerical value for each new record in a table. This is especially useful for setting a Primary Key when manually assigning unique values would be tedious or difficult. With Auto Increment, the database ensures that each new record receives a unique identifier without requiring manual input.

In SQL Server, the IDENTITY(starting_value, increment_value) property is used to implement the auto-increment feature, where the starting_value defines the initial number and increment_value specifies how much to increase the value with each new record.

29. What Is a Trigger in SQL?

A trigger in SQL is a database object that contains a set of SQL statements, which are automatically executed (or "triggered") when a specific event occurs within the database. Triggers are usually associated with a particular table and are activated by certain operations such as INSERT, UPDATE, or DELETE.

30. What Is T-SQL?

T-SQL stands for Transact-SQL, a Microsoft product that extends the SQL language for interacting with relational databases. It is optimized for use with Microsoft SQL Server and is used for performing database transactions. T-SQL is crucial because all communications with an SQL Server instance are conducted through Transact-SQL statements. Additionally, users can define functions using T-SQL.

31. What Is an ER Diagram and Its Essential Aspects?

The Entity-Relationship (ER) Model is a framework for identifying entities that need to be represented in a database and describing the relationships between those entities. The ER data model outlines the enterprise schema, which graphically depicts the overall logical structure of a database.

Essential aspects of ER Diagrams:

- **Entities:** These are the main objects or concepts in the system, typically represented by rectangles.
- **Attributes:** These are properties or characteristics of entities, usually shown as ovals connected to their respective entities.
- **Relationships:** These show how entities are connected, represented by diamond shapes or lines connecting entities.

- **Cardinality:** This indicates the numerical nature of relationships between entities, such as one-to-one, one-to-many, or many-to-many.
- **Primary Keys:** These are unique identifiers for each instance of an entity, often underlined in the diagram.
- **Foreign Keys:** These are attributes that link to primary keys in other entities, representing relationships.

32. What Is a SQL Common Table Expression (CTE)?

A Common Table Expression (CTE) in SQL is a temporary result set that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement. Defined using the WITH keyword, a CTE allows you to create a named subquery that can be reused within the same query. CTEs simplify complex queries by breaking them into smaller, more manageable parts, making the SQL code easier to read and maintain.

33. What Is a Sparse Column?

A sparse column in SQL Server is a type of column that optimizes storage for null values, using significantly less space than a regular column. By using sparse columns, you can reduce the storage requirements when a large number of null values are expected. However, this comes with a slight increase in overhead when retrieving non-null values. They are most beneficial when they provide at least 20 to 40 percent in space savings.

34. Can You Explain the Concepts of Shared, Exclusive, and Updated Locks?

In SQL Server, locks are used to manage concurrency and maintain data integrity during transactions. Here are the key types:

- **Shared (S) Locks:** Allow multiple transactions to read the same data concurrently. However, no transaction can modify the data while shared locks are held. Multiple transactions can hold shared locks on the same resource at the same time.
- **Exclusive (X) Locks:** Used when a transaction modifies data. An exclusive lock prevents any other transaction from reading or modifying the data. Only one transaction can hold an exclusive lock on a resource at a time.
- **Update (U) Locks:** Serve as an intermediate step between shared and exclusive locks. A transaction acquires an update lock before upgrading to an exclusive lock. This is primarily used to prevent deadlocks in scenarios where multiple transactions may attempt to update the same resource.

35. What Is SQL Server Profiler?

SQL Server Profiler is a diagnostic tool used to trace, monitor, and troubleshoot activities within Microsoft SQL Server, Microsoft's Relational Database Management System (RDBMS). It allows developers and Database Administrators (DBAs) to capture and analyze detailed events happening within the server, such as query executions, login activity, and transaction performance.

With SQL Server Profiler, you can create and manage traces, replay trace results, and gain insights into the performance and health of the SQL Server instance, helping identify and resolve issues.

36. What Do You Mean by Check Constraints?

Check constraints in SQL Server are used to enforce data integrity by ensuring that the data in a table meets specific conditions. They define a rule or condition that must be evaluated as TRUE or UNKNOWN for any INSERT or UPDATE operation to succeed. If the condition evaluates to FALSE, the operation is rejected. Check constraints can refer to multiple columns within the same table but cannot reference columns in other tables.

37. What Is an SQL Server Agent?

SQL Server Agent is a Windows service that manages and executes scheduled administrative tasks, known as jobs, in Microsoft SQL Server.

Key features of SQL Server Agent include:

- **Job Scheduling:** Run jobs at specified times or in response to specific events.
- **Notifications:** Alert administrators about server events or job statuses via email, pager, or net send messages.
- **Operator Management:** Define which users can receive notifications.
- **Job Steps:** Break complex jobs into multiple steps with flow control between them.
- **Security Integration:** Work with SQL Server security to control who can create and execute jobs.

38. What Is COALESCE in an SQL Server, and Describe Its Properties?

The COALESCE function in SQL Server returns the first non-null expression from a list of arguments. It is useful for handling null values in queries by providing an alternative value when encountering nulls.

Properties of COALESCE:

- **Multiple Arguments:** It can accept two or more arguments of different data types.
- **Data Type Precedence:** It returns the data type of the argument with the highest precedence among the provided arguments.
- **All Nulls:** If all arguments are NULL, it returns NULL.
- **Flexibility:** It is often used as a more versatile alternative to the ISNULL function.

39. What Is ETL in SQL?

ETL, which stands for Extract, Transform, and Load, is a crucial process in data warehousing and database management. It involves three main steps:

- **Extract:** Data is retrieved from various source systems.
- **Transform:** The extracted data is then transformed into a staging area to fit the desired format or structure.
- **Load:** Finally, the transformed data is loaded into a data warehouse or target database.

ETL tools integrate these three functions to streamline the process of transferring and integrating data between different systems and databases.

40. What Do You Mean by Collation?

Collations in SQL Server set the rules for sorting data and determine how case and accent sensitivity are handled. They define the bit patterns for each character in the database's metadata. SQL Server supports the inclusion of objects with diverse collations within the same database.

Collations are important because they affect how queries with ORDER BY, GROUP BY, and comparison operators behave. They can be set at various levels:

- Server level
- Database level
- Column level
- Expression level

41. What Is the Use of the UPDATE_STATISTICS Command?

The UPDATE_STATISTICS command is used to refresh the statistics for a table or indexed view in SQL Server. These statistics provide information about the distribution of data, which helps the query optimizer make more informed decisions about the most efficient way to execute queries.

42. What Is a Filtered Index?

A filtered index is a type of non-clustered index that includes a WHERE clause to index only a subset of the rows in a table. This specialized index is designed to optimize queries that target specific data subsets. By indexing only a portion of the data, a filtered index offers advantages over a full-table index, including reduced size and lower maintenance costs.

43. How to Copy Tables in SQL?

Copying a table can be useful for creating duplicates for testing or other purposes without affecting the original table. In MySQL, you can copy a table's structure or both its structure and data.

To create an exact copy of an existing table's structure (without copying the data), use the following syntax:

```
CREATE TABLE new_table_name LIKE original_table_name;
```

44. Which Is Generally Faster, a Table Variable or a Temporary Table?

The performance between table variables and temporary tables can vary based on the scenario. Generally:

Table Variables:

- Faster for small datasets (less than 100 rows).
- Stored in tempdb, with limited indexing and no automatic statistics.

Temporary Tables:

- Better for larger datasets.
- Stored in tempdb, supports indexes and automatic statistics.

In general, table variables are faster for small datasets, while temporary tables are more efficient for larger datasets.

45. What Do You Mean by Scheduled Tasks in the SQL Server?

Scheduled Tasks in SQL Server, managed through SQL Server Agent, are pre-defined operations that run automatically at specified times or intervals. They're used to automate routine database maintenance and administrative tasks.

46. What Is the Use of the SIGN Function?

The SIGN function in SQL Server returns an indicator of a number's sign. It's used to determine whether a numeric expression is positive, negative, or zero. The SIGN function returns:

- 1 if the expression is positive.
- -1 if the expression is negative.
- 0 if the expression is zero.

Syntax:

```
SIGN ( numeric_expression )
```

47. What Is Database Mirroring?

Database mirroring is a technique used in relational database management systems (RDBMS) to ensure data consistency despite high availability requirements by creating redundant copies of a dataset. In other words, it maintains two copies of a single database on separate SQL Server Database Engine instances.

48. What Is a Unique Key, and How Can We Create It?

A unique key is a column or set of columns in a database that uniquely identifies each row in a table, ensuring no two rows have the same values for these columns.

To Create a Unique Key:

- **Using CREATE TABLE:** Define the unique key with the UNIQUE constraint in the CREATE TABLE statement.
- **Using ALTER TABLE:** Add a unique key to an existing table using the ALTER TABLE statement with the UNIQUE constraint.

49. What Is the ORDER BY Clause?

The ORDER BY clause in SQL is used to sort the result set of a query in ascending or descending order based on one or more columns. It's the last clause in a SELECT statement.

50. Explain Aggregate Functions

SQL Aggregate functions are used to process the values from multiple rows according to specific criteria, combining them to produce a single, more meaningful result. These functions summarize data by merging multiple values into one comprehensive outcome.

51. What Is a Constraint in SQL?

Constraints are rules applied to the type of data within a table. They allow us to specify limits on the kinds of data that can be stored in a particular column of a table.

Types of constraints in SQL:

- **PRIMARY KEY:** Ensures each row in a table is uniquely identifiable.
- **FOREIGN KEY:** Ensures referential integrity between two tables.
- **UNIQUE:** Ensures all values in a column are unique.
- **CHECK:** Ensures all values in a column satisfy a specific condition.
- **NOT NULL:** Ensures a column cannot have NULL values.
- **DEFAULT:** Provides a default value for a column when none is specified.

52. What Is a Foreign Key?

A foreign key is used to link two tables by referring to the primary key of one table from columns in another table. This means that a column in one table points to the primary key attribute of another table. Thus, an attribute designated as a primary key in one table can be a foreign key in another. However, it is important to note that a foreign key is independent of the primary key.

53. What Are the Different Types of Commands in SQL Server?

The different types of commands in SQL Server are:

Data Definition Language (DDL):

- **CREATE:** Creates database objects (tables, views, etc.).
- **ALTER:** Modifies the structure of database objects.
- **DROP:** Deletes database objects.
- **TRUNCATE:** Removes all records from a table.

Data Manipulation Language (DML):

- **SELECT:** Retrieves data from tables.

- **INSERT:** Adds new records to a table.
- **UPDATE:** Modifies existing records.
- **DELETE:** Removes records from a table.

Data Control Language (DCL):

- **GRANT:** Gives specified privileges to users.
- **REVOKE:** Removes specified privileges from users.
- **DENY:** Prevents a user from performing certain operations.

Transaction Control Language (TCL):

- **BEGIN TRANSACTION:** Starts a transaction.
- **COMMIT:** Saves the changes of a transaction.
- **ROLLBACK:** Undoes changes made in a transaction.
- **SAVE TRANSACTION:** Creates a savepoint within a transaction.

The SQL interview questions covered above are fundamental and essential for any fresher to know, as they form the basic foundation of SQL and database management. Understanding these basics is crucial for building a strong SQL skill set and performing well in interviews.

As you progress, you'll explore intermediate-level SQL interview questions to deepen your knowledge and enhance your expertise in SQL. This will help you tackle more complex scenarios and advance your skills in the field.

Intermediate-Level SQL Interview Questions

These SQL interview questions cover advanced topics and are ideal for candidates with some experience in SQL. They are designed to test your ability to handle complex queries and optimize performance, helping you further enhance your skills.

54. What Is an Index?

A database index is a data structure that improves the speed of data retrieval operations on a database table. This comes at the cost of extra writes and more storage space to maintain the additional copy of the data. Data can be stored in only one order on a disk, so indexes are created on tables to support faster access by different values, enabling quicker searches such as binary searches. These indexes occupy additional disk space but provide faster searches for frequently accessed values.

55. What Are Clustered Indexes?

A clustered index is established only when both of the following criteria are fulfilled:

- The data or file being moved to secondary memory should be in sequential or sorted order.
- There must be a key value, meaning it cannot have duplicate values.

Applying a clustered index to a table will result in sorting that table only. You can create only one clustered index per table, similar to a primary key. A clustered index is comparable to a dictionary where data is arranged in alphabetical order.

56. What Are Non-clustered Indexes?

A non-clustered index is an index structure independent of the data stored in a table, reordering one or more selected columns. This index is designed to improve the performance of frequently used queries that are not covered by a clustered index.

57. What Are the Differences Between Stored Procedures and Functions in SQL?

Stored procedures and functions are both types of database objects used to encapsulate and execute SQL code. While they both allow for reusable code and encapsulation, they serve different purposes and have distinct characteristics in SQL.

Below are the differences between Stored Procedures and Functions in SQL:

Aspect	Stored Procedures	Functions
Purpose	Perform actions that can modify data.	Compute and return values.
Return value	It can return multiple result sets.	Must return a single value or table.
Parameters	It can have input, output, and input/output parameters.	It can only have input parameters.
Transaction control	It can contain transaction control statements.	It cannot contain transaction control statements.
Calling method	EXECUTE or EXEC keyword.	It is called a function.
DML operations	Can perform INSERT, UPDATE, and DELETE.	Cannot perform DML operations (except table-valued functions).
Performance	Generally faster for complex operations.	Generally faster for simple computations.
Default parameters	Supports default parameter values.	Does not support default parameter values.
Table variables	Can declare and use table variables.	Cannot declare table variables (except table-valued functions).
Determinism	Non-deterministic by default.	Can be deterministic or non-deterministic.

58. Write Some of the Most Common SQL Queries

Common SQL queries are fundamental tools for interacting with databases, allowing users to retrieve, manipulate, and manage data. Understanding and mastering these queries is essential for effective database management and analysis.

Here are some of the most common SQL queries:

Query Type	Purpose	Example

SELECT	Retrieve data from one or more tables.	SELECT column1, column2 FROM table_name;
INSERT	Add new rows to a table.	INSERT INTO table_name (column1, column2) VALUES (value1, value2);
UPDATE	Modify existing data in a table.	UPDATE table_name SET column1 = value1 WHERE condition;
DELETE	Remove rows from a table.	DELETE FROM table_name WHERE condition;
WHERE	Filter results based on a condition.	SELECT * FROM table_name WHERE condition;
ORDER BY	Sort the result set.	SELECT * FROM table_name ORDER BY column1 ASC/DESC;
GROUP BY	Group rows that have the same values.	SELECT column1, COUNT(*) FROM table_name GROUP BY column1;
HAVING	Specify a search condition for a group.	SELECT column1, COUNT(*) FROM table_name GROUP BY column1 HAVING COUNT(*) > 5;
JOIN	Combine rows from two or more tables.	SELECT * FROM table1 INNER JOIN table2 ON table1.column = table2.column;
UNION	Combine result sets of two or more SELECT statements.	SELECT column FROM table1 UNION SELECT column FROM table2;
CREATE TABLE	Create a new table in the database.	CREATE TABLE table_name (column1 datatype, column2 datatype);
ALTER TABLE	Modify an existing table structure.	ALTER TABLE table_name ADD column_name datatype;
DROP TABLE	Remove a table from the database.	DROP TABLE table_name;
CREATE INDEX	Create an index on table columns.	CREATE INDEX index_name ON table_name (column1, column2);
LIKE	Search for a specified pattern in a column.	SELECT * FROM table_name WHERE column1 LIKE 'pattern';

59. What Are the Differences Between UNION and UNION ALL?

UNION and UNION ALL are SQL operations used to combine results from multiple queries into a single result set. While both are used for merging data, they handle duplicate rows differently and have distinct performance implications.

Following are the differences between UNION and UNION ALL in SQL Server:

Aspect	UNION	UNION ALL
Duplicate Rows	Removes duplicate rows.	Retains all rows, including duplicates.
Performance	Generally slower due to duplicate removal.	It is faster as it doesn't remove duplicates.
Result Set	Returns only distinct rows.	Returns all rows from all queries.
Sorting	Implicitly sorts the result set.	It does not sort the result set.
Resource Usage	It is more CPU intensive due to duplicate checking.	Less CPU intensive.
Number of Columns	Each SELECT statement must have the same number of columns.	Each SELECT statement must have the same number of columns.
Use Case	When you need unique results.	When duplicates are acceptable or expected.

60. What Is the Difference Between JOIN and UNION?

JOIN and UNION are SQL operations used to combine data from multiple tables or queries, but they do so in different ways. JOIN is used to combine rows from two or more tables based on a related column, while UNION combines the result sets of two or more queries into a single result set.

Here is the difference between JOIN and UNION in SQL Server:

Aspect	JOIN	UNION
Purpose	Combines rows from two or more tables based on a related column between them.	Combines the result sets of two or more SELECT statements.
Result	Creates a new result set by combining columns from multiple tables.	Creates a new result set by combining rows from multiple queries.
Number of columns	Can increase (combines columns from joined tables).	Remains the same (must match in all unioned queries).

Number of rows	Depending on the join type, it can increase, decrease, or remain the same.	Increases (combines rows from all queries).
Duplicates	Retains duplicates	Removes duplicates by default (unless UNION ALL is used).
Column names	Retains original column names from all tables.	Uses column names from the first SELECT statement
Data types	Joined columns must be comparable; other columns can differ.	Corresponding columns in all queries must have compatible data types.
Usage	For querying related data across multiple tables.	This is for combining similar data from different tables or queries.
Syntax	table1 JOIN table2 ON condition	query1 UNION query2

61. Explain WITH Clause in SQL

The WITH clause defines a temporary relationship, which is available only to the query where the WITH clause appears. SQL applies predicates in the WITH clause after groups have been formed, allowing the use of aggregate functions.

62. How Do You Use the SCOPE_IDENTITY() Function in SQL Server?

The *SCOPE_IDENTITY()* function returns the last identity value inserted into an identity column within the same scope. It is specific to the current session and execution scope, making it useful for retrieving the most recent identity value generated during an insert operation.

63. What Is the Purpose of Using WITH TIES?

The WITH TIES option is used with the TOP clause in a SELECT statement to include additional rows that have the same value as the last row in the result set. This ensures that all rows with tied values are included, not just the number specified by TOP.

64. How Can Deadlocks in SQL Server Be Resolved?

Deadlocks in SQL Server can be resolved using several strategies:

- **Using the UPDLOCK Hint:** Apply the UPDLOCK hint in queries to request an update lock on a table, preventing other transactions from obtaining conflicting locks and reducing the risk of deadlocks.
- **Handling Deadlocks with TRY…CATCH:** Implement TRY…CATCH blocks to catch and handle deadlocks. In the CATCH block, you can rerun the transaction chosen as the deadlock victim.
- **Setting Deadlock Priority:** Use the DEADLOCK_PRIORITY setting to determine which transaction should be rolled back when a deadlock occurs, with SQL Server rolling back the transaction with the lower priority.
- **Keeping Transactions Short:** Minimize the duration of transactions to reduce the time locks are held on resources, lowering the likelihood of deadlocks.
- **Avoiding User Interaction in Transactions:** Prevent user interactions during transactions to shorten their duration and decrease deadlock chances.

- **Using Snapshot Isolation:** Enable the READ_COMMITTED_SNAPSHOT isolation level to avoid deadlocks by ensuring read operations do not block write operations.
- **Fine-Tuning MAXDOP Settings:** Adjust the MAXDOP (Maximum Degree of Parallelism) setting to avoid deadlocks caused by parallel query execution, particularly in cases of parallelism deadlocks.

65. What Is the Difference Between Local and Global Temporary Tables?

This topic is frequently asked in SQL interview questions, making it important for interview preparation. Local and global temporary tables in SQL Server both store temporary data but differ in scope and visibility. Understanding these differences is essential for selecting the right table type in database operations.

Below are the differences between local and global temporary tables:

Aspect	Local Temporary Tables	Global Temporary Tables
Naming Convention	Start with a single # (e.g., #TempTable).	Start with double ## (e.g., ##TempTable).
Scope	Visible only to the connection that created it.	Visible to all connections.
Lifetime	Dropped automatically when the creating connection closes.	It dropped when the last connection referencing it closed.
Accessibility	Accessible only within the creating session.	Accessible by any session.
Naming Conflicts	You can have multiple tables with the same name in different sessions.	Must have unique names across all sessions.
Use Case	Temporary storage for session-specific data.	Sharing temporary data across multiple sessions
Performance	Generally better for single-session use.	It may have more overhead due to global visibility.
Concurrency	No concurrency issues.	It may require concurrency handling.

66. What Is a Livelock?

Livelock occurs when two or more processes continuously change their state in response to each other, but none of them make progress. Unlike a deadlock, where processes are stuck waiting for each other, in a livelock, the processes are actively running but fail to achieve any productive work.

67. How Can You Differentiate Between the SUBSTR and CHARINDEX?

The SUBSTR and CHARINDEX functions serve different purposes in SQL. SUBSTR extracts a substring from a string, while CHARINDEX locates the position of a substring within a string. Understanding the distinction between these functions is crucial for effectively manipulating and querying string data. This topic often appears in SQL interview questions, making it important for those preparing for interviews.

Here is the table comparing SUBSTR and CHARINDEX to understand the difference between Them.

Aspect	SUBSTR	CHARINDEX
Purpose	Extracts a substring from a string.	Finds the starting position of a substring within a string.
Return Value	Returns the extracted substring.	Returns the numeric position of the substring.
Syntax	SUBSTR(string, start, [length])	CHARINDEX(substring, string, [start_location]).
Parameters	string, start position, optional length.	Substring to find, string to search, optional start position.
Case Sensitivity	Case-sensitive by default.	Case-insensitive by default.
Usage in WHERE clause	It can be used directly.	It is often used with other functions like SUBSTRING.
Functionality	Extraction	Searching

68. When Should You Use the COMMIT and ROLLBACK Commands in SQL?

The COMMIT and ROLLBACK commands are crucial for managing transactions in SQL, as they help maintain data integrity. Knowing when to use each command is important for effectively controlling the outcome of your data transactions.

- You can use the COMMIT command to save all changes made during a transaction permanently to the database. It should be executed when all operations in the transaction have been completed, making the changes visible to other sessions.
- You can use the ROLLBACK command to undo all changes made during the current transaction. This is necessary if an error occurs or if you decide not to proceed with the changes, restoring the database to its state before the transaction begins.

Understanding these commands is considered helpful in SQL interview questions, making them a key topic for interview preparation.

69. Explain SSMA in an SQL Server

SQL Server Migration Assistant (SSMA) is a Microsoft tool designed to simplify the migration of databases to SQL Server. The latest version of SSMA is v9.2, and it supports migration from MySQL, SQL Server, Access, Db2, SAP ASE, and Oracle to SQL Server.

SSMA helps with schema conversion, data migration, and validation. It also generates detailed reports about the migration process and allows users to compare SQL code between the original and migrated databases. SSMA is compatible with Windows Server and Windows desktop versions and supports both on-premises and Azure SQL Server targets.

70. Can You Explain What SQL Server Integration Services Are and Describe Their Primary Functions?

SQL Server Integration Services (SSIS) is a comprehensive platform for developing enterprise-level data integration and transformation solutions. Integration Services can be utilized to address complex business challenges. For example:

- Copy or download files.
- Load data into data warehouses.
- Cleanse and mine data.
- Manage SQL Server objects and data.

SSIS is primarily used for two functionalities:

- **Data Integration:** Merges data from multiple sources into a unified dataset.
- **Workflow Automation:** Automates tasks like file transfers, data loading, and SQL Server maintenance.

71. What Are the Differences Between Derived Attributes, Derived Persistent Attributes, and Computed Attributes?

Derived attributes, derived persistent attributes, and computed attributes refer to different ways of calculating and storing data in a database. Each serves a unique purpose depending on whether the value is calculated on demand or stored for future use.

Here are the differences between Derived attribute, Derived persistent attribute, and computed attribute:

Aspect	Derived Attribute	Derived Persistent Attribute	Computed Attribute
Storage	Not stored	Stored in database	Not stored
Performance	It may impact query performance.	Better read performance.	It may impact query performance.
Update Frequency	Always current	Needs manual or triggered updates.	Always current.
Disk Space	Doesn't use extra space.	Uses additional disk space	Doesn't use extra space.
Use Case	When real-time calculation is needed.	When the calculation is expensive and data doesn't change often.	When calculation is simple and data changes frequently

72. What Are the Different Levels of Normalization, and Can You Briefly Explain Each One?

Normalization is a process in SQL used to organize data in a database to reduce redundancy and improve data integrity. It's a topic frequently covered in SQL interview questions to test your understanding of database design principles.

Levels of Normalization:

Level	Description
1NF	Eliminate repeating groups and identify the primary key.
2NF	Meet 1NF and remove partial dependencies.
3NF	Meet 2NF and remove transitive dependencies.
BCNF	Meet 3NF, and every determinant must be a candidate key.
4NF	Meet BCNF and remove multi-valued dependencies.
5NF	Meet 4NF and remove join dependencies.

73. What Is the Difference Between DELETE and TRUNCATE Commands?

The DELETE and TRUNCATE commands are both used to remove data from tables in SQL, but they operate differently. Understanding these key differences is important for answering SQL interview questions related to data manipulation.

Let's see the primary differences between DELETE and TRUNCATE command:

Aspect	DELETE	TRUNCATE
Logging	Fully logged operation.	Minimally logged.
Speed	Slower for large data.	Faster for large data.
WHERE Clause	Supports WHERE clause.	Doesn't support the WHERE clause.
Triggers	Fires trigger.	Doesn't fire triggers.
IDENTITY Reset	Doesn't reset IDENTITY.	Resets IDENTITY to seed value.
Transaction	It can be rolled back.	It can't be rolled back (in most cases).
Permissions	Requires DELETE permission.	Requires ALTER TABLE permission.

74. Compare Local Variables and Global Variables

Local and global variables differ in their scope and accessibility within a program. In SQL, local variables are accessible only within the block or procedure they are defined, while global variables are available across the entire session or connection.

Let's compare local and global variables:

Aspect	Local Variables	Global Variables
Scope	Limited to the batch or procedure.	Server-wide scope
Declaration	Declared with a DECLARE statement.	Predefined by SQL Server.
Naming	Start with @	Start with @@
Lifetime	It exists for the duration of the batch or procedure.	Exists for the duration of the server session.
Modification	The user can modify it.	Most are read-only, and some can be modified.
Example	@VariableName	@@SERVERNAME

75. What Is the Difference Between OLAP and OLTP?

OLAP (Online Analytical Processing) and OLTP (Online Transaction Processing) are two distinct systems designed for different purposes in data management. OLAP is used for complex data analysis and reporting, while OLTP focuses on managing day-to-day transaction processing.

Below are the differences between OLAP and OLTP, and it's one of the most common topics in SQL interview questions, testing your knowledge of database management systems and their use cases.

Aspect	OLAP	OLTP
Full Form	Online Analytical Processing.	Online Transaction Processing.
Purpose	Analysis and decision support.	Day-to-day transactions.
Data Model	Multidimensional	Normalized
Query Type	Complex, aggregated	Simple, standardized
Data Updates	Periodic, bulk updates.	Continuous updates
Data Volume	Large amounts of historical data.	Current operational data.

Users	Analysts, executives	Customer-facing applications, clerks.
Response Time	It can be slower (seconds to minutes).	Fast (milliseconds to seconds).
Backup and Recovery	Periodic	Continuous
Data Redundancy	Often denormalized for performance.	Normalized to avoid redundancy.

76. What Is the FLOOR Function in the SQL Server?

The FLOOR function in SQL Server returns the largest integer value that is less than or equal to a given number.

The syntax for the FLOOR function in SQL Server is:

FLOOR(number)

Where *number* is the numeric value for which you want to find the largest integer that is less than or equal to it.

The FLOOR function in SQL Server works as follows:

- It takes a numeric input value.
- It then returns the largest integer value that is less than or equal to the input value.

For example,

- FLOOR(25) returns 25 since 25 is the largest integer less than or equal to 25.
- FLOOR(25.7) returns 25, as 25 is the largest integer less than or equal to 25.7.

77. What Is the Use of SQL Server Locks, and What Resources Can Be Locked by Server Locks?

SQL Server uses locks to manage concurrent access to database resources and ensure data integrity during transactions. Locks help prevent conflicts and ensure that multiple transactions do not interfere with each other, maintaining the consistency and accuracy of the data.

Resources that can be locked include:

- Rows: Specific rows in a table.
- Pages: Data pages within a table.
- Tables: Entire tables.
- Database: The entire database.

These locks prevent other transactions from accessing or modifying the locked resources until the lock is released.

78. What Is SQL Latch Contention, and How Can It Be Avoided?

When multiple threads attempt to acquire incompatible latches on the same in-memory structure, latch contention occurs. The SQL engine automatically manages latch usage to preserve memory consistency. In cases of latch contention, the SQL server queues conflicting latch requests until the active ones are completed.

79. What Do You Mean by Magic Tables?

Magic Tables, also known as pseudo-tables, are virtual tables used by SQL Server in triggers to access data modified by DML operations. They include:

- **INSERTED Table:** Contains new or updated rows for INSERT or UPDATE operations.
- **DELETED Table:** Contains old or removed rows for UPDATE or DELETE operations.

80. How Can You Prevent SQL Injection Vulnerabilities?

To prevent SQL injection, use the following techniques:

- **Use Parameterized Queries:** Employ parameterized statements or prepared statements to separate user inputs from SQL queries.
- **Utilize Stored Procedures:** Opt for stored procedures that accept parameters rather than embedding user inputs directly into SQL queries.
- **Escape User Inputs:** Apply database-specific escaping functions to neutralize special characters in user inputs.
- **Conduct Regular Audits:** Perform both automated and manual security audits to identify and address potential vulnerabilities.
- **Apply Least Privilege:** Grant minimal permissions to application users to limit the impact of a potential SQL injection attack.
- **Implement a Web Application Firewall (WAF):** Use a WAF to monitor and filter traffic, detecting and blocking known SQL injection attack patterns.

81. What Do You Mean by the Recovery Model in SQL Server and Its Types?

The recovery model in SQL Server determines how transaction logs are managed and how a database can be restored in the event of failure or data loss. There are three primary recovery models:

- **Simple Recovery Model:** Automatically truncates transaction logs, which means it does not retain logs for point-in-time recovery or log shipping. This model supports only full and bulk-logged backups and is suitable for environments where some data loss is acceptable.
- **Full Recovery Model:** Retains transaction logs to enable point-in-time recovery. It supports full, differential, and log backups, providing comprehensive protection against data loss. Regular log backups are necessary to manage storage and maintain performance.
- **Bulk-Logged Recovery Model:** Similar to the Full Recovery Model but optimized for bulk operations, like large data imports. It retains transaction logs but reduces the log space used for bulk operations. It supports full, differential, and log backups but does not support point-in-time recovery as effectively as the Full Recovery Model.

82. What Are the Different Types of Backups Used in SQL Servers?

The different types of backups used in SQL Server are:

- **Full Backup:** Captures the entire database, including all data and log files. It is a complete snapshot of the database at a specific point in time.
- **Differential Backup:** Includes only the changes made since the last full backup. It is smaller and faster than a full backup but requires a full backup to restore.
- **Transaction Log Backup:** Records all transactions that have occurred since the last transaction log backup. It is essential for point-in-time recovery and maintaining the log chain.
- **Tail-Log Backup:** Captures any remaining log records from the tail of the transaction log before performing a restore operation. It is used to recover the most recent transactions not included in the last log backup.

- **Copy-Only Backup:** A full backup that does not affect the backup sequence or log chain. It is used for special purposes, such as creating a backup for testing.
- **File and Filegroup Backup:** Allows for backing up individual files or filegroups within a database. This can be useful for large databases where you want to manage backups more granularly.
- **Partial Backup:** Backs up the primary filegroup and any read/write filegroups but not the read-only filegroups. It is useful for large databases with multiple filegroups.

83. How Can You Use HAVING and WHERE Clauses in a Single Query?

Here's an example of using the HAVING and WHERE clauses together:

SELECT technology, COUNT(*) **AS** num_projects, AVG(budget) **AS** avg_budget

FROM projects

WHERE project_duration > 6

GROUP BY technology

HAVING COUNT(*) > 2 **AND** AVG(budget) > 100000;

Let's go through the execution steps:

1. **WHERE clause: WHERE project_duration > 6**
    - The WHERE clause filters the rows, keeping only the projects with a duration of more than 6 months.
2. **GROUP BY clause: GROUP BY technology**
    - The technology column groups the data after the WHERE clause has been applied.
3. **HAVING clause: HAVING COUNT(*) &#62 2 AND AVG(budget) &#62 100000**
    - The HAVING clause filters the grouped rows, keeping only the technologies with more than 2 projects and an average budget greater than $100,000.
    - Since the HAVING clause uses aggregate functions (COUNT and AVG), these functions must also be present in the SELECT clause.
4. **SELECT clause: SELECT technology, COUNT(*) AS num_projects, AVG(budget) AS avg_budget**
    - The SELECT clause retrieves the technology, the COUNT of projects, and the AVG of budgets for the groups that passed the HAVING clause filter.

84. What Are the Uses of Stored Procedures?

Stored procedures are used for various purposes in SQL Server, including data validation, access control, and encapsulating complex business logic. They help in managing large volumes of data efficiently by centralizing logic, which can enhance execution speed and reduce network traffic by limiting the amount of data sent between the client and the server. Additionally, stored procedures improve security by controlling access to the underlying database and its operations.

85. What Types of Stored Procedures Are in an SQL Server?

There are two primary types of stored procedures in SQL servers.

- **System-defined Stored Procedures:** These are built-in procedures provided by SQL Server that perform administrative tasks or help manage the SQL Server environment. Examples include sp_help, sp_who, and sp_configure.

- **User-defined Stored Procedures:** These are created by users to encapsulate business logic or complex queries for reuse. They can be further classified into:
    - **Transact-SQL (T-SQL) Procedures:** Written in T-SQL, these are the most common type of user-defined stored procedures.
    - **CLR (Common Language Runtime) Procedures:** Written in .NET languages such as C# or VB.NET, these procedures allow for more complex operations and can be integrated with SQL Server using the CLR integration feature.

86. What Are the Benefits of Using Stored Procedures in an SQL Server?

One of the primary benefits of using stored procedures is their ability to process information directly on the database server, which minimizes network usage between servers.

Some of the other benefits are:

- **Minimize Network Usage:** Stored procedures execute on the database server, reducing the amount of data transferred between the client and the server, which can improve performance and reduce network congestion.
- **Enhanced Security:** By controlling access to data and encapsulating logic, stored procedures can limit the exposure of data and protect against unauthorized access. They also help in guarding against SQL injection attacks by using parameterized queries.
- **Reduced Development Costs and Improved Reliability:** Stored procedures can encapsulate complex logic, making it reusable and reducing duplication of code. This encapsulation simplifies maintenance and reduces the risk of errors in application code.
- **Performance Improvement:** Stored procedures are precompiled and optimized by the database engine, which can enhance execution speed compared to sending individual SQL queries from client applications.
- **Encapsulation and Code Modifications:** Changes to business logic can be made within the stored procedure without affecting the client applications, reducing the risk of data corruption or errors from client-side code.

87. What Is the Use of ISNULL()?

The *ISNULL()* function in SQL Server is used to replace NULL values with a specified alternative value. This function takes two arguments: the expression to be checked for NULL and the value to return if the expression is NULL. If the expression is NULL, *ISNULL()* returns the second argument; otherwise, it returns the first argument.

Syntax:

```
ISNULL(check_expression, replacement_value)
```

88. What Are Aggregate Functions, and What Are the Different Types of Functions in SQL Server?

An aggregate function in SQL Server performs a calculation on a set of values and returns a single result. These functions are commonly used to summarize or analyze data across multiple rows.

SQL Server offers various types of functions, including aggregate functions, scalar functions, and table-valued functions. Each type serves different purposes, such as summarizing data, performing operations on individual values, or returning tables.

Below are some common aggregate functions, along with their descriptions:

Aggregate Function	Description
AVG()	Computes the average of a column's values.
COUNT()	Counts the number of rows in a column
FIRST()	Returns the first value in an ordered set.
LAST()	Returns the last value in an ordered set.
MAX()	Retrieves the maximum value from a column.
MIN()	Retrieves the minimum value from a column.
SUM()	Calculates the sum of values in a numeric column.

89. What Are Scalar Functions, and What Are the Different Types of Functions in SQL Server?

In SQL Server, a scalar function is a type of function that returns a single value based on the input provided. Scalar functions are used to perform operations that produce a single result, such as mathematical calculations or string manipulations.

In addition to scalar functions, SQL Server includes several other types of functions, such as table-valued functions, which return a table, and aggregate functions, which perform calculations on a set of values. Each type serves a specific purpose and can be utilized in various scenarios to enhance SQL queries.

Below are some common scalar functions, along with their descriptions:

Scalar Function	Description
UCASE()	Converts a string to uppercase.
LCASE()	Converts a string to lowercase.
MID()	Extracts a substring from a string.
LEN()	Returns the length of a string.
ROUND()	Rounds a number to a specified decimal place.
NOW()	Returns the current date and time.

FORMAT()	Formats a value according to a specified format.

The intermediate-level SQL interview questions listed above are designed to help both beginners and those with some experience prepare effectively for interviews. As you proceed further, you will encounter more challenging SQL interview questions that are particularly relevant for experienced professionals.

Advanced-Level SQL Interview Questions

Here, the focus shifts to advanced topics essential for experienced SQL professionals. By exploring these advanced SQL interview questions, you will gain a comprehensive understanding of complex database features and concepts, equipping you to handle intricate data management scenarios effectively.

90. How Important Is Database Design in SQL Servers?

Database design is crucial in SQL Servers because it impacts performance, data integrity, and efficiency. Proper design ensures optimal table structures, efficient relationships between tables, and minimal data redundancy. This leads to better performance, easier maintenance, and reliable data management.

91. Write a Query to Display Employee Details With Their Department Names

To effectively showcase your skills during SQL interview questions, you might be asked to write a query that displays employee details along with their department names. Here's how you can approach it:

Below is a demonstration of the query asked by the interviewer.

```
SELECT emp.EmployeeID, emp.FirstName, emp.LastName, emp.Salary, dept.DepartmentName
FROM Employees emp
JOIN Departments dept ON emp.DepartmentID = dept.DepartmentID;
```

This query joins the Employees table with the Departments table using the DepartmentID, and it retrieves employee details along with their department names.

92. Write a Query to Display Employee Details Along With Department_name and Their Age Between 21 to 25

During SQL interview questions, you might be asked to write a query that not only displays employee details and their department names but also filters the results based on specific criteria. For example, you might need to retrieve employees whose ages are between 21 and 25.

Below is a demonstration of the query asked by the interviewer.

```
SELECT emp.EmployeeID, emp.FirstName, emp.LastName, emp.Salary, dept.DepartmentName,
    DATEDIFF(YEAR, emp.BirthDate, GETDATE()) AS Age
FROM Employees emp
JOIN Departments dept ON emp.DepartmentID = dept.DepartmentID
WHERE DATEDIFF(YEAR, emp.BirthDate, GETDATE()) BETWEEN 21 AND 25;
```

This query joins the Employees table with the Departments table to get the department names and filters the results to include only employees aged between 21 and 25.

93. Write a Query to Display the Details of the Employee Whose Salary Is Above 23000, Whose Age Is Above 22, and Who Is Working in the CSE Department

During the SQL interview questions, you may be asked to write a query that filters employee details based on multiple conditions. For instance, you might need to find employees whose salary exceeds 23,000, whose age is above 22, and who work in the CSE department.

Below is a demonstration of the query asked by the interviewer.

```
SELECT emp.EmployeeID, emp.FirstName, emp.LastName, emp.Salary,
    DATEDIFF(YEAR, emp.BirthDate, GETDATE()) AS Age
FROM Employees emp
JOIN Departments dept ON emp.DepartmentID = dept.DepartmentID
WHERE emp.Salary > 23000
  AND DATEDIFF(YEAR, emp.BirthDate, GETDATE()) > 22
  AND dept.DepartmentName = 'CSE';
```

This query joins the Employees table with the Departments table, filtering for employees with a salary greater than 23,000, an age over 22, and who are in the CSE department.

94. Write an SQL Query to Find All the Employees Whose Salaries Are Between 50000 and 100000

In SQL interview questions, you might be asked to write a query that retrieves employees based on specific salary ranges. For example, you could be asked to find all employees whose salaries fall between 50,000 and 100,000.

Below is a demonstration of the query asked by the interviewer.

```
SELECT emp.EmployeeID, emp.FirstName, emp.LastName, emp.Salary
FROM Employees emp
WHERE emp.Salary BETWEEN 50000 AND 100000;
```

This query selects employees from the Employees table whose salary is within the specified range.

95. Write a SQL Query to Fetch the Department-Wise Employees Sorted by the Department's Count in Ascending Order

In SQL interview questions, you may be required to write a query that organizes employees by their respective departments and sorts these departments based on the number of employees they have. This ensures you can demonstrate your ability to aggregate and sort data effectively.

Below is a demonstration of the query asked by the interviewer.

```
SELECT dept.DepartmentName, COUNT(emp.EmployeeID) AS EmployeeCount
FROM Departments dept
LEFT JOIN Employees emp ON dept.DepartmentID = emp.DepartmentID
GROUP BY dept.DepartmentName
ORDER BY EmployeeCount ASC;
```

This query joins the Employees and Departments tables, groups the results by department and employee, and sorts the departments based on the number of employees in ascending order.

96. Write an SQL Query to Fetch All the Employees Who Are Also the Managers From the Employee Table

In SQL interview questions, you might be asked to write a query to identify employees who hold the position of manager within the same employee table. This tests your ability to filter data based on hierarchical relationships.

Below is a demonstration of the query asked by the interviewer.

SELECT emp.EmployeeID, emp.FirstName, emp.LastName

FROM Employees emp

WHERE emp.EmployeeID **IN** (**SELECT DISTINCT** ManagerID **FROM** Employees **WHERE** ManagerID **IS NOT NULL**);

This query selects employees who are also managers by checking if their EmployeeID exists in the list of distinct ManagerIDs.

97. What Are the Differences Between Clustered and Non-clustered Indexes in SQL?

When discussing indexing in SQL, it's essential to understand the distinctions between clustered and non-clustered indexes. These types of indexes affect data retrieval performance and structure differently, impacting how data is stored and accessed.

The following are the key differences between the clustered Index and the non-clustered Index:

Aspect	Clustered Index	Non-Clustered Index
Data Storage	Determines the physical order of data in a table.	Separate structure from data rows.
Number per Table	Only one per table.	Multiple allowed per table.
Leaf Level	Contains actual data pages.	Contains row pointers.
Speed	Generally faster for range queries.	It can be faster for selective queries.
Size	No additional storage is needed.	Requires extra storage.
Default	The primary Key constraint is created by default.	It must be explicitly created.
Table Structure	Alters the table structure.	Doesn't alter the table structure.

Ideal Use	For columns frequently used to sort or range query.	For columns often used in WHERE clauses or joins.

98. How Can You Hide an Instance of the SQL Server Database Engine?

To hide an instance of the SQL Server Database Engine:

- Open SQL Server Configuration Manager and expand the "SQL Server Network Configuration" section. Right-click on "Protocols for [InstanceName]" and select "Properties."
- Go to the "Flags" tab, find the "Hide Instance" option, set it to "Yes," and click "OK" to apply the changes. The modification will take effect immediately for new connections.

99. What Is SSRS?

SSRS stands for SQL Server Reporting Services. It is a server-based reporting tool from Microsoft that provides a unified and scalable platform for creating, managing, and delivering business reports. SSRS allows users to design interactive and web-based reports, replacing traditional paper-based reporting. It supports multiple distribution methods, including file sharing and email delivery, and can generate reports in various formats like CSV, Microsoft Excel, and HTML.

It integrates with SharePoint for enhanced report delivery and management and is a key component of Microsoft Business Intelligence, aiding in effective data analysis across enterprises.

100. How Can You Alter a Table Schema in an SQL Server?

To alter a table schema in SQL Server, use the ALTER SCHEMA statement. This statement is used to transfer an object (such as a table, view, or function) from one schema to another within the same database.

Syntax:

ALTER SCHEMA target_schema_name

TRANSFER [object_category ::] object_name;

102. How Can You Optimize Query Performance in SQL Server Using Techniques Like Indexing?

Optimizing query performance in SQL Server through techniques like indexing is essential for efficient data retrieval and improved execution times. To ensure these optimizations are effective across various environments and configurations, thorough testing is crucial.

You can test your SQL Server setups in diverse environments, simulate different loads, and verify that your indexing strategies and performance optimizations are effective. This comprehensive testing ensures your database performance remains consistently high, leading to more efficient data management and faster response times.

In addition to indexing, performance can be further optimized by analyzing query execution plans, updating statistics to ensure the query optimizer has accurate information, and avoiding the retrieval of unnecessary columns in SELECT statements. Regular monitoring of index performance, including rebuilding and reorganizing indexes as needed, also helps maintain optimal performance over time.