**The correct order of execution in SQL is**

1. **FROM/JOIN**
2. **WHERE**
3. **GROUP BY**
4. **HAVING**
5. **SELECT**
6. **DISTINCT**
7. **ORDER BY**
8. **LIMIT/OFFSET**

**Q 1) What is Normalization?**

Normalization is a database design technique to remove redundant data.

Normalization is used to minimize **redundancy** and **dependency** by organizing fields and table of a database.

There are some rules of database normalization, which is commonly known as Normal From, and they are:

* First normal form(1NF)
* Second normal form(2NF)
* Third normal form(3NF)
* Boyce-Codd normal form (BCNF)

Using these steps, the redundancy, anomalies, inconsistency of the data in the database can be removed.

**What is the primary use of Normalization?**

Normalization is mainly used to add, delete or modify a field that can be made in a single table. The primary use of Normalization is to remove redundancy and remove the insert, delete and update distractions. Normalization breaks the table into small partitions and then links them using different 1relationships to avoid the chances of redundancy.

**What are the disadvantages of not performing database Normalization?**

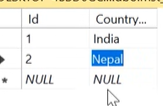
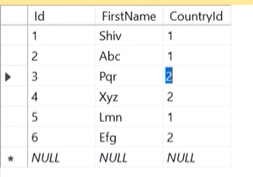
The major disadvantages are:

The occurrence of redundant terms in the database causes the **waste of space in the disk**.

Due to redundant terms, inconsistency may also occur. If any change is made in the data of one table but not made in the same data of another table, then **inconsistency will occur**. This inconsistency will lead to the maintenance problem and effects the ACID properties as well.

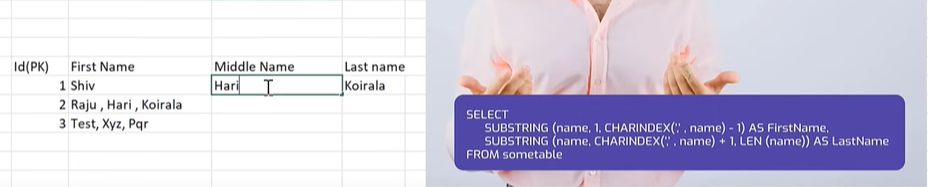
**How to implement normalization?**

Normalization is implemented by splitting tables into two, one with reference data (master table) and other transaction data.

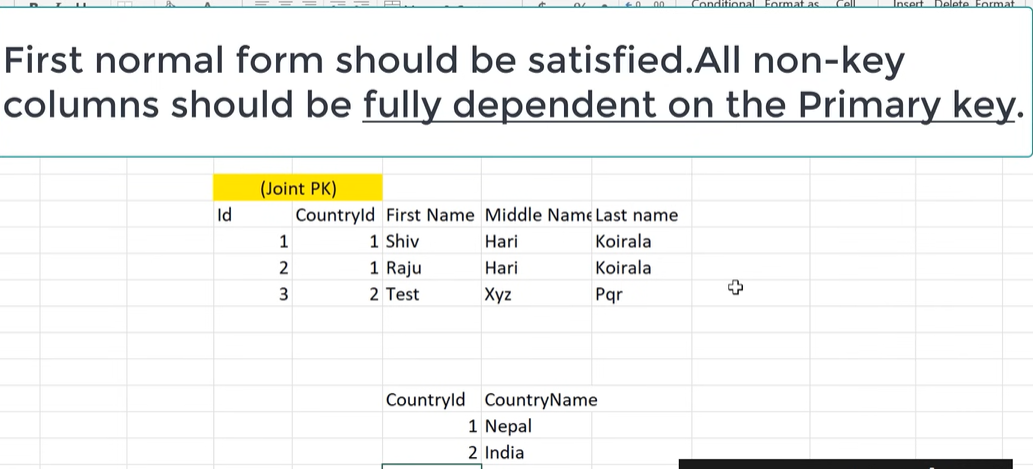
 

**Explain 1st, 2nd and 3rd Normal form?**

1st normal form: A table is in first normal form when the columns have atomic values. It should not have repeating groups.

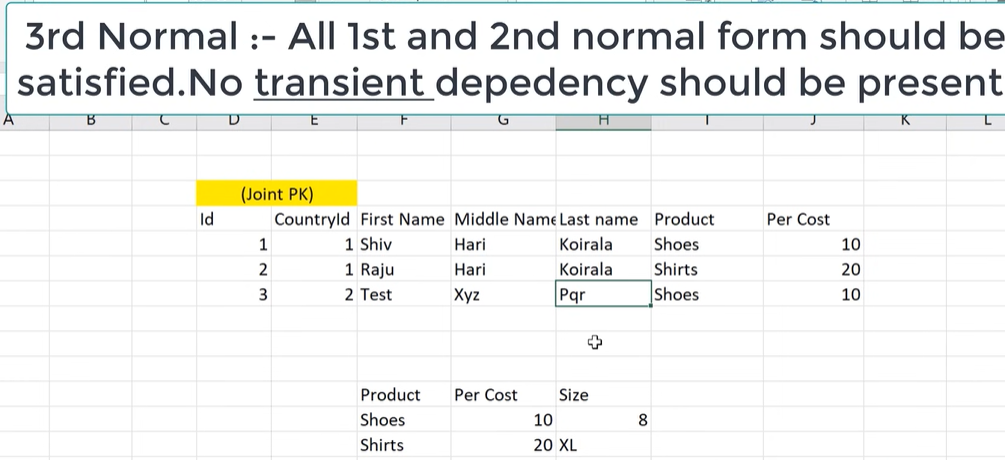


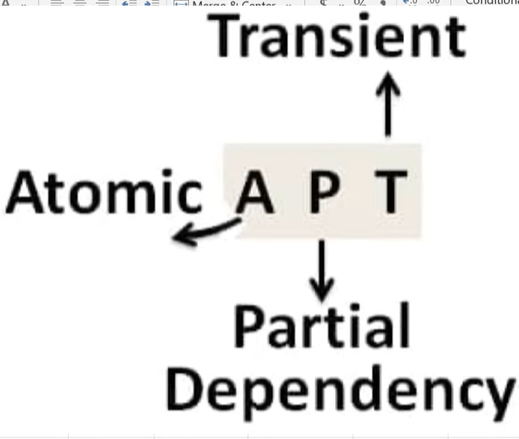
2nd normal form: First normal form should be satisfied. All non-key columns should be fully dependent on the primary key.



3nd normal form:

Any non key column should not depend on the other non key columns.



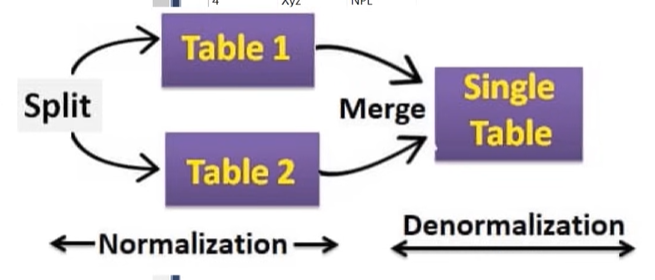


**Q 2) what is Denormalization?**

Denormalization is a database design technique to improve search performance.

We merge tables.

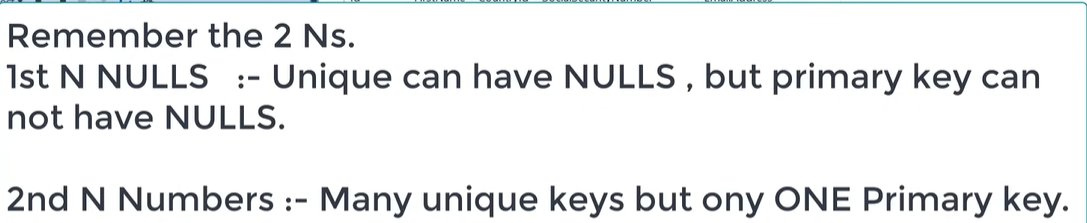
Duplicate data is present in denormalization.



Denormalization is a technique used by database administrators to optimize the efficiency of their database infrastructure. The denormalization concept is based on Normalization, which is defined as arranging a database into tables correctly for a particular purpose. This method allows us to add redundant data into a normalized database to alleviate issues with database queries that merge data from several tables into a single table. It adds redundant terms into the tables to avoid complex joins and many other complex operations.

Denormalization doesn't mean that normalization will not be done. It is an optimization strategy that takes place after the normalization process.

**Q 3) Unique key & Primary Key:**



|  |  |
| --- | --- |
| **Primary Key** | **Unique Key** |
| The primary key act as a unique identifier for each record in the table. | The unique key is also a unique identifier for records when the primary key is not present in the table. |
| We cannot store NULL values in the primary key column. | We can store NULL value in the unique key column, but only one NULL is allowed. |
| We cannot change or delete the primary key column values. | We can modify the unique key column values. |

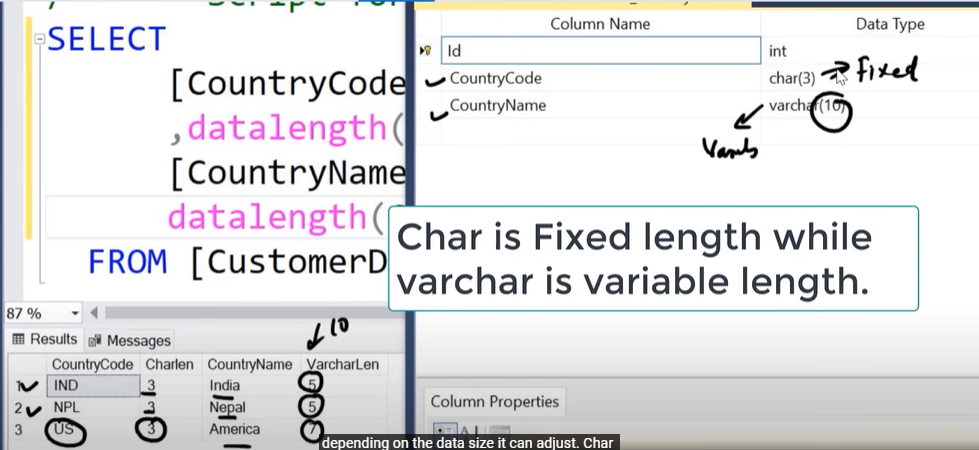
**What is a primary key?**

A primary key is a field or the combination of fields that uniquely identify each record in the table. It is one of a special kind of unique key. If the column contains a primary key, it cannot be null or empty. A table can have duplicate columns, but it cannot have more than one primary key. It always stores unique values into a column. **For example,** the ROLL Number can be treated as the primary key for a student in the university or college.

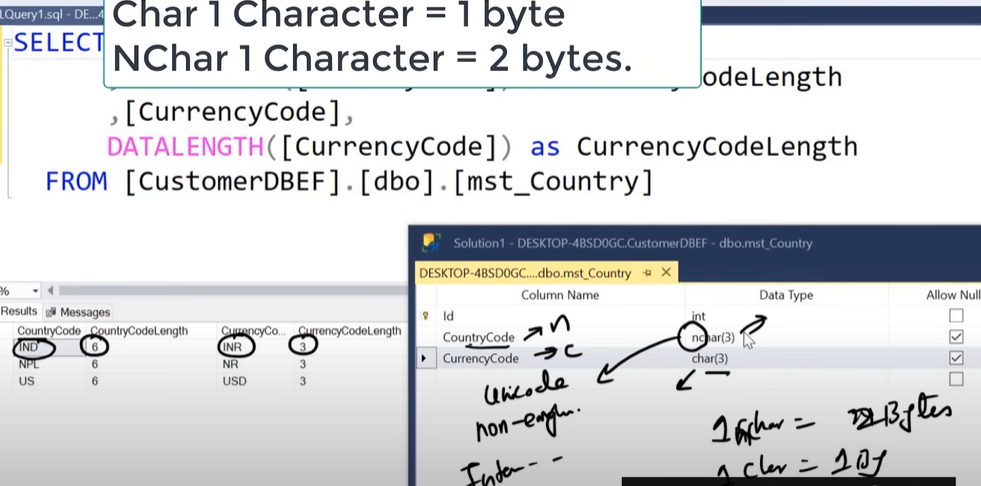
**unique key**

A unique key is a single or combination of fields that ensure all values stores in the column will be unique. It means a column cannot stores duplicate values. This key provides uniqueness for the column or set of columns. **For example,** the email addresses and roll numbers of student's tables should be unique. It can accept a null value but only one null value per column. It ensures the integrity of the column or group of columns to store different values into a table.

**Q 4) Char & Varchar:**



**Q 5) Char & NChar:**



**Q 6) Indexes:**

Indexes increases search performance

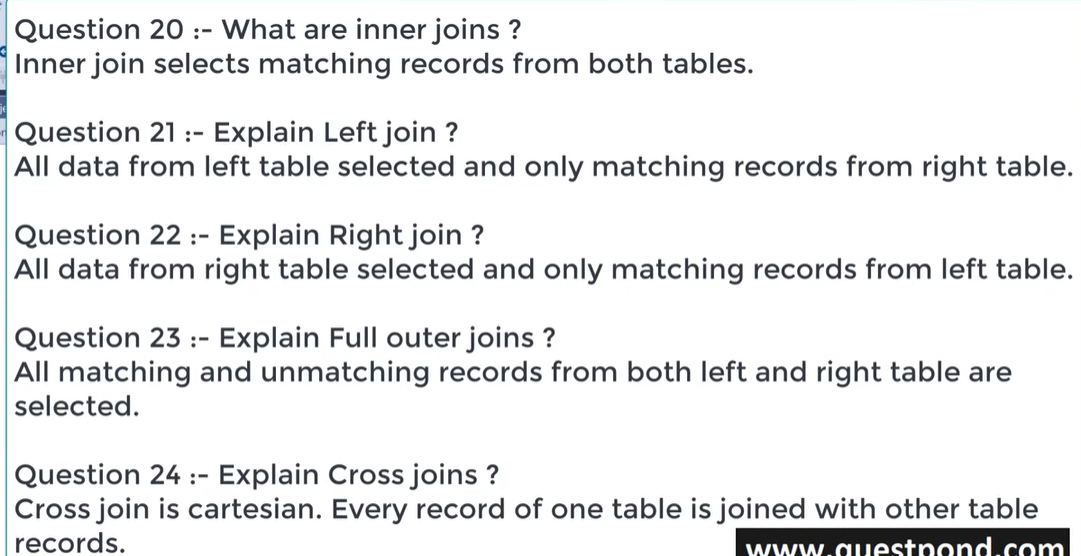
Indexing reduces the number of data pages we need to visit to find a particular data page. It also has a unique value meaning that the index cannot be duplicated.

An index is used to improve the performance of queries by allowing for faster data retrieval.

**Q 7) Identity:**



**Q 8) Joins:**



**Q 9) What is an inconsistent dependency?**

An Inconsistent dependency refers to the difficulty of getting relevant data due to a missing or broken path to the data. It leads users to search the data in the wrong table, resulting in an error as an output.

**Q 10) Explain OLTP Vs OLAP.**

OLTP: - Online Transaction Processing.

OLAP: - Online Analytical Processing.

OLTP Normalization avoids redundancy and we follow normalization design (1st, 2nd and 3rd normal).

OLAP Denormalization improve search performance and we follow denormalization design.

OLAP Stands for Online Analytical Processing. OLAP systems have the capability to analyse database information of multiple systems at the current time. The primary goal of OLAP Service is data analysis and not data processing.

OLTP stands for Online Transaction Processing. OLTP has the work to administer day-to-day transactions in any organization. The main goal of OLTP is data processing not data analysis.

**Q 11) What is a view in SQL?**

A view is a database object that has no values. It is a virtual table that contains a subset of data within a table. It looks like an actual table containing rows and columns, but it takes less space because it is not present physically. It is operated similarly to the base table but does not contain any data of its own. Its name is always unique. A view can have data from one or more tables. If any changes occur in the underlying table, the same changes reflected in the views also.

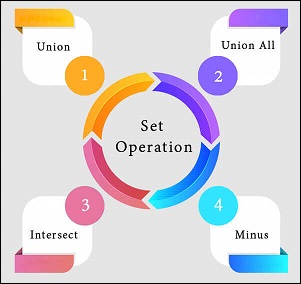
**Q 12) What is a "TRIGGER" in SQL?**

A trigger is a set of SQL statements that reside in a system catalog. It is a special type of stored procedure that is invoked automatically in response to an event. It allows us to execute a batch of code when an insert, update or delete command is run against a specific table because the trigger is the set of activated actions whenever DML commands are given to the system.

SQL triggers have two main components one is action, and another is an event. When certain actions are taken, an event occurs as a result of those actions.

**Q 13) What are the set operators in SQL?**

We use the set operators to merge data from one or more tables of the same kind. Although the set operators are like SQL joins, there is a significant distinction. SQL joins combine columns from separate tables, whereas SQL set operators combine rows from different queries. SQL queries that contain set operations are called compound queries. The set operators in SQL are categories into four different types:



**A. UNION:** It combines two or more results from multiple SELECT queries into a single result set. It has a default feature to remove the duplicate rows from the tables. The following syntax illustrates the Union operator:

SELECT columns FROM table1

UNION

SELECT columns FROM table2;

**B. UNION ALL:** This operator is similar to the Union operator, but it does not remove the duplicate rows from the output of the SELECT statements. The following syntax illustrates the UNION ALL operator:

SELECT columns FROM table1

UNION ALL

SELECT columns FROM table2;

**C. INTERSECT:** This operator returns the common records from two or more SELECT statements. It always retrieves unique records and arranges them in ascending order by default. Here, the number of columns and data types should be the same. The following syntax illustrates the INTERSECT operator:

SELECT columns FROM table1

INTERSECT

SELECT columns FROM table2;

**D. MINUS:** This operator returns the records from the first query, which is not found in the second query. It does not return duplicate values. The following syntax illustrates the MINUS operator:

SELECT columns FROM table1

MINUS

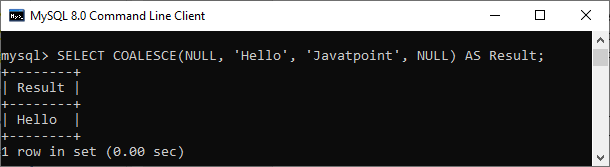
SELECT columns FROM table2;

**Q 14) What is a constraint? Tell me about its various levels.**

The constraint is used to specify the **rule and regulations** that allows or restricts what values/data will be stored in the table. It ensures data accuracy and integrity inside the table. It enforces us to store valid data and prevents us from storing irrelevant data. If any interruption occurs between the constraint and data action, the action is failed. Some of the most commonly used constraints are NOT NULL, PRIMARY KEY, FOREIGN KEY, AUTO\_INCREMENT, UNIQUE KEY, etc.

**Q 15) What are the syntax and use of the COALESCE function?**

The COALESCE() function evaluates the arguments in sequence and returns the first NON-NULL value in a specified number of expressions. If it evaluates arguments as NULL or not found any NON-NULL value, it returns the NULL result.



**Q 16) What is the difference between the WHERE and HAVING clauses?**

The main difference is that the WHERE clause is used to filter records before any groupings are established, whereas the HAVING clause is used to filter values from a group. The below comparison chart explains the most common differences:

|  |  |
| --- | --- |
| **WHERE** | **HAVING** |
| This clause is implemented in row operations. | This clause is implemented in column operations. |
| It does not allow to work with aggregate functions. | It can work with aggregate functions. |
| This clause can be used with the SELECT, UPDATE, and DELETE statements. | This clause can only be used with the SELECT statement. |

**Q 17) How many Aggregate functions are available in SQL?**

The aggregate function is used to determine and calculate several values in a table and return the result as a single number

**SQL provides seven (7) aggregate functions**

**Q 18) What is a database?**

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

**Q 19) What is a schema?**

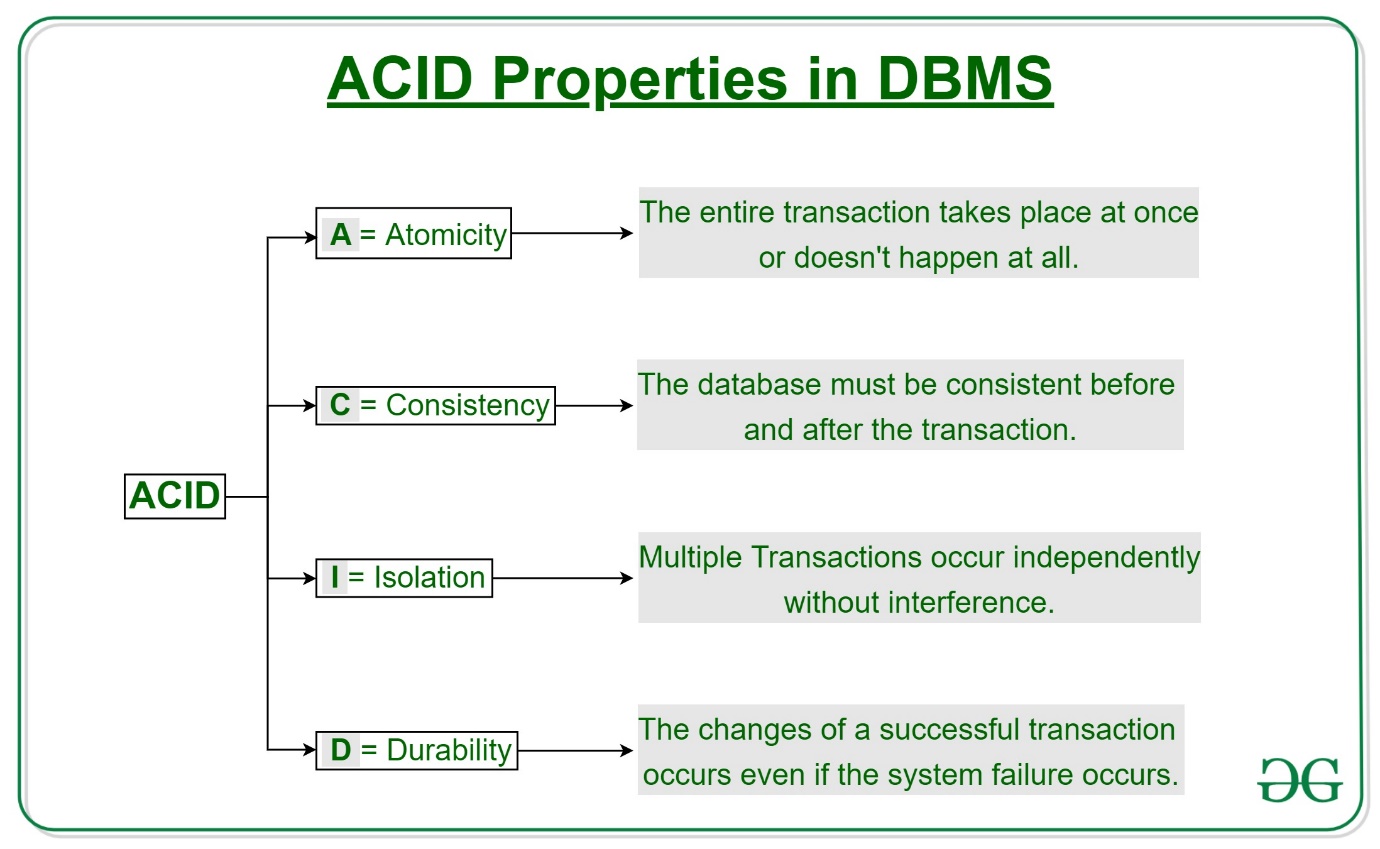
A collection of database structural elements such as tables, stored procedures, indexes, functions, and triggers. It shows the overall database architecture, specifies the relationships between various objects of a database, and defines different access permissions for them.

**Q 20) 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.

**Q 21) What is ASID properties in sql server**

If a database operation has these ACID properties, it can be called an ACID transaction, and data storage systems that apply these operations are called transactional systems.



**Atomicity =** Either all OR None.Failed transaction will restart.

**Consistency =** Before trans. start and after trans completed sum of money should be same (e.g. bank trans, A= 1000, B=3000, sum = 4000, after trans sum ll’be 4000)

**Isolation =** parallel to serial schedule conversion.

**Durability =** changes should be permanent.

<https://medium.com/@rganesh0203/sql-interview-questions-asked-in-a-product-based-company-df6c9f625bbe>

**Q 22) What is the use of the GROUP BY clause in SQL?**

The GROUP BY clause in SQL is used to group rows that share the same values in specified columns into summary rows. It is typically used with aggregate functions like COUNT(), SUM(), AVG(), etc., to perform operations on these grouped data.

**Q 23) What is ETL in SQL?**

**ETL (Extract, Transform, Load)** is a common process used in data warehousing and business intelligence to move data from various sources into a data warehouse or database. The process involves three steps:

* **Extract**: In this step, data is extracted from various sources such as databases, files, or web services. This may involve using SQL queries to extract data from databases, APIs, or web scraping tools to extract data from web services or files.
* **Transform:** Once the data has been extracted, it is transformed or cleaned to make it suitable for storage and analysis. This may involve applying filters, aggregating data, or converting data types. SQL is commonly used to transform data as part of the ETL process.
* **Load:**The final step is to load the transformed data into a data warehouse or database. This may involve loading data into tables, creating indexes, or performing other database operations.

The ETL process is critical for data integration, as it allows organizations to collect data from various sources, transform it into a consistent format, and store it in a central location for analysis and reporting. ETL tools such as Microsoft SQL **Server Integration Services (SSIS)**or Talend can automate much of the ETL process and provide a visual interface for designing and managing data flows.

**Q 24) What are Aggregate and Scalar functions?**

In database management systems, there are two main types of functions used to manipulate data:**aggregate functions and scalar functions**.

Aggregate functions perform calculations on a set of values and return a single value representing a summary of that set.

**Scalar functions**, on the other hand, operate on a single value and also return a single value. They can manipulate data in various ways, such as performing string operations, date calculations, or mathematical computations. Examples of scalar functions include CONCAT

**Q 25) List different Types of Index in SQL?**

In SQL, different indexes can be created to improve query performance. Here are some of the most common types of indexes:

* **Clustered Index:**
* **Non-Clustered Index:**
* **Unique Index:**
* **Composite Index:**
* **Full-Text Index**
* **Spatial Index**

**Q 26) What are OLAP and OLTP?**

**OLAP (Online analytical processing) and OLTP (Online Transaction Processing)**are two different types of database systems that are used for different purposes in the data processing. OLTP systems are mainly used for recording and processing transactions in real-time. These transactions are usually brief and involve updating, inserting, or deleting data in the database. OLTP systems are built to handle a high volume of concurrent transactions while ensuring data consistency, accuracy, and availability.

In contrast, OLAP systems are used for analytical processing, which involves complex queries and data aggregation for decision-making and business intelligence. OLAP systems typically store a large amount of historical data and allow for sophisticated data analysis, such as trend analysis, forecasting, and data mining. OLAP systems are designed for quick query performance and data retrieval, even when handling large amounts of data.

To sum up, OLTP systems are primarily used for real-time transaction processing, while OLAP systems are used for complex analysis of historical data to support decision-making and business intelligence.

**Q 27) What is a Self Join in SQL?**

A self-join in SQL occurs when a table is joined with itself. This is typically used when a relationship exists within the same table and you need to compare rows within that table. For instance, if you have an employee table where each employee has a manager, you can use a self-join to find all employees with the same manager.  
A self-join uses table aliases to distinguish between the original table and the copy of the table:

SELECT A.employee\_name, B.emplotrayee\_name AS manager\_name

FROM employees A

JOIN employees B ON A.manager\_id = B.employee\_id;

**Q 28) Explain the use of the COALESCE function in SQL.**

SELECT COALESCE(first\_name, 'Unknown') AS display\_name

FROM users;

In this example, if the first\_name is NULL, the query will return 'Unknown'. The COALESCE function can accept multiple arguments, making it more versatile than ISNULL, which can only evaluate two arguments.

**Q 29) How do you delete duplicate rows in SQL?**

WITH CTE AS (

SELECT column\_name,

ROW\_NUMBER() OVER (PARTITION BY column\_name ORDER BY column\_name) AS row\_num

FROM table\_name

)

DELETE FROM CTE WHERE row\_num > 1;

**Q 30) What is a CTE (Common Table Expression) in SQL?**

A **Common Table Expression (CTE)** is a temporary result set in SQL that is defined using the WITH keyword and can be referenced within a SELECT, INSERT, UPDATE, or DELETE query. CTEs make queries easier to read and maintain, especially when working with complex joins, recursive queries, or hierarchical data.

A CTE is particularly useful for breaking down complicated queries into simpler steps or reusing a common subquery in multiple parts of the main query. For example:

WITH CTE AS (

SELECT employee\_id, manager\_id, salary

FROM employees

WHERE salary > 50000

)

SELECT \* FROM CTE WHERE manager\_id IS NOT NULL;

**Q 31) Explain how to perform pagination in SQL.**

**Answer:**  
**Pagination** in SQL refers to the process of dividing large sets of data into smaller chunks or pages to improve performance and provide easier data navigation. Pagination is commonly used in web applications to display data in a user-friendly way across multiple pages.

**ISNULL Function**

The ISNULL function in SQL Server allows you to replace NULL values with a specified alternative value.

SELECT EmployeeID, ISNULL(FirstName, 'N/A') AS FirstName, ISNULL(LastName, 'N/A') AS LastName

FROM Employees;

Expected Output:

EmployeeID | FirstName | LastName

---------------------------------

1 | John | Doe

2 | Jane | N/A

3 | N/A | N/A

4 | N/A | Johnson

**COALESCE Function**

The COALESCE function in SQL Server allows you to evaluate multiple expressions and return the first non-NULL value.

SELECT EmployeeID, COALESCE(FirstName, LastName, 'Unknown') AS FullName

FROM Employees;

Expected Output:

EmployeeID | FullName

---------------------

1 | John

2 | Jane

3 | Unknown

4 | Johnson

**Explain transactions and how to implement it?**

Transaction helps to implement series of activity treat as one single unit. Either everything is successful and or everything rollbacks.

begin try

begin tran

insert into tblCustomer(FirstName) values('shiv1')

insert into tblCustomer(FirstName) values('shiv2')

--delete

--update

commit tran

end try

begin catch

rollback

end catch

<https://www.simplilearn.com/top-sql-interview-questions-and-answers-article>

**What are the different types of triggers?**

* DML triggers: These triggers fire in response to data manipulation language (DML) operations such as INSERT, UPDATE, or DELETE.
* DDL triggers: These triggers fire in response to data definition language (DDL) events, such as table or view creation.

**How is data integrity ensured in SQL?**

Data integrity in SQL is ensured through various means, including constraints (e.g., primary keys, foreign keys, check constraints), normalization, transactions, and referential integrity constraints. These mechanisms prevent invalid or inconsistent data from being stored in the database.

**What is an SQL injection?**

SQL injection is a cybersecurity attack method that involves the insertion of malicious SQL code into an application's input fields or parameters. This unauthorized action enables attackers to illicitly access a database, extract confidential information, or manipulate data.