### What is MySQL?

SQL Interview Question

MySQL is an open-source, Relational Database Management System that stores data in a structured format using rows and columns. It’s software that enables users to create, manage, and manipulate databases.

**Relational Database:** Relational Database is type of database collection of data items that store data in form of table ,row amd column

Example: MySQL , Oracle, SQLite

**Database Commands :**

1.**Create Database :**Create command is used to create database and table in sql

CREATE DATABASE databaseName;

2.

**Show Databases :** Show command retrive information of database and show all list of database

SHOW DATABASES;

3.

**Drop Database :** Drop command used of remove existing database object such as table

DROP DATABASE database\_Name;

4.

**Select Database and Query :**

USE my\_databaseName;

SELECT \* FROM my\_table;

**1. Numeric Data Types:**

* INT: Integer, a whole number without fraction parts.
* FLOAT: Floating-point number.
* DOUBLE: Double-precision floating-point number.
* DECIMAL: Fixed-point number.

**2. Date and Time Data Types:**

* DATE: Date in 'YYYY-MM-DD' format.
* TIME: Time in 'HH:MM format.
* DATETIME: Date and time combination in 'YYYY-MM-DD HH:MM format.
* TIMESTAMP: Timestamp, typically 'YYYY-MM-DD HH:MM format, but also auto-updated to the current timestamp.

**3. String Data Types:**

* CHAR: Fixed-length string, maximum length specified.
* VARCHAR: Variable-length string, maximum length specified (n).
* TEXT: Suitable to store large blocks of text.

**4. Binary Data Types:**

* BINARY: Fixed-length binary string.

**5. ENUM Data Type:**

* ENUM: Enumeration, list of permitted values.

**\* How can create Table** CREATE TABLE tablename ( PersonID int,

FirstName varchar(255), LastName varchar(255), int age,

Dob date,Address varchar(255), City varchar(255)

);

# \* Database Insertion

INSERT INTO table\_name(customer\_id,first\_name,last\_name,age,country) VALUES(1,"priyanshu","Singh",20,"India"), (2,"Anshu","Kumar",20,"India"), (3,"piyush","Rajput",20,"India") // multi value insertion (4,"priya," Singh",20,"India");

or

insert into table\_name values(1,"Priyanshu",20); // Single value insertion insert into table\_name values(1,"Priyanshu",20);

# \* Database Select/Show data

SELECT \* FROM table\_name;

# \* Commands: Drop Table

DROP TABLE table\_name;

# \* Commands: Delete Data in table

delete from table\_name

where id=1; // where clause condition can can anything using name ,id,other unique data

\* **Command : Delete Column in table**

alter table table\_name

drop column column\_name;

# \* TRUNCATE TABLE:

**Note:** The TRUNCATE TABLE statement is used to delete all rows from a table, but it keeps the table structure as it’s TRUNCATE TABLE table\_name

## **\* Alter :** Alter command is used to modify structure existing table and add new column in table

ALTER TABLE table\_name

ADD COLUMN rollno INT; // rollno new coumn name Int dataype

# \* Copy table

CREATE TABLE new\_table\_name AS SELECT \* FROM old\_table\_name;

# \* Rename table name

RENAME TABLE current\_table\_name TO new\_table\_name;

**Select :** The SELECT command in SQL is used to retrieve data from a database. we can access any row or column in table

# Q .SQL SELECT Statement:

### This selects all columns from a table named users:

SELECT \* FROM table\_name;

### SQL SELECT TOP:

**This selects the top 5 rows from the users table:**

SELECT **TOP 5 \*** FROM table\_name;

### SQL SELECT Desecnding/reverse Order:

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

### SQL SELECT RANDOM:

SELECT \* FROM **table\_name ORDER** BY RAND() LIMIT 1;

### Q.SQL SELECT IN:

SELECT \* FROM users WHERE id IN (1, 2, 3);

### Q.SQL SELECT Multiple:

SELECT name, email FROM users;

# Update :

## The UPDATE command in SQL is used to modify existing records in a table. It allows you to change the values of one or more columns in one or more rows based on specified conditions.

update database\_name Set amount=50000

where id=3; // condition according to you

## **Rollback:** ROLLBACK in SQL is a transactional control language that is used to undo the transactions that have not been saved in the database.If you make a mistake in an UPDATE query and accidentally modify incorrect data, you may need to use ROLLBACK to revert the changes and restore the database to its previous state.

1. **Update salary of employee then rollback**  update table\_name

set Salary=10000000 where id=1; RollBack;

### Commit : Commit use to pramanent save query in database if you applly rollback No, a query cannot be reverted after a COMMIT operation.

CREATE TABLE ExampleTable ( ID INT PRIMARY KEY, Name VARCHAR(255) );

INSERT INTO ExampleTable (ID, Name) VALUES (1, 'John'); INSERT INTO ExampleTable (ID, Name) VALUES (2, 'Jane'); COMMIT;

# Logical Operator :

•AND Operator • OR Operator • NOT Operator

**Comparison Operators**

• LIKE Operator • IN Operator • IS NULL Operator • Equal Operator • Not equal(<>) Operator • Greater than • Less than • Greater than equal • Less than equal • Between • IS NOT NULL

**Arithmetic Operators**

•+: Addition • -: Subtraction •\*: Multiplication •/: Divisio •%: Modulus (remainder)

**Set Operators**

•Union •Union All •Intersection •Except •Distinct •All

**Logical Operator :**

### SQL AND Operator: Return true if all condition must be true

SELECT \* FROM Customers(table\_name) WHERE age>=18 AND age<=28;

### SQL OR Operator: Returns true if at least one of the conditions should be true

SELECT \* FROM Customers(table\_name) WHERE country="UK" OR country="UAE";

### SQL NOT Operator: Returning true if the condition is false.

SELECT \* FROM Customers where not country="UAE";

**Comparison Operators**

**•SQL LIKE Operator : is used for pattern matching in WHERE clauses, like string find**

Select \* from Customers(table\_name)

where first\_name like "J%"; // Select name **stating J alphabet**

Select \* from Customers(table\_name)

**where first\_name like "%J"; // Select name ending J alphabet**

Select \* from Customers

where first\_name LIKE "B\_tt\_"; //select name starting name **start B—tt-- alphabet**

SELECT \* FROM Customers

WHERE first\_name LIKE 'J%' OR first\_name LIKE 'D%'; // using operator

**•SQL IN/ not in Operator**

SELECT \* FROM Customers WHERE country in('USA','UAE'); //check whether a value matches any value in a specified list or subquery.

SELECT \* FROM Customers WHERE country not in(<>) ('USA','UAE'); //NOT IN returns TRUE if the column value does't match any value in the list.

**SQL NOT EQUAL Operator:**

SELECT \* FROM products(table\_name) WHERE category != 'Electronics';

**SQL EQUAL Operator:**

Select \* from table\_name where id=2;

**SQL Greater than Operator:**

# Select \* from table\_name where age>21;

**SQL Less than Operator:**

# Select \* from table\_name where age>21;

**SQL Greater and less than equal Operator:**

# Select \* from table\_name where age>=21;

# Select \* from table\_name where age<=21;

**SQL Between Operators**

# Select \* from Betech(table\_name) where age between 20 and 22;

**SQL IS NULL / IS NOT NULL Operators**

# SELECT \* FROM Employees WHERE manager\_id IS NULL;

# SELECT \* FROM Employees WHERE manager\_id IS NOT NULL;

# Set Operators

**UNION Operator: The UNION operator is used to combine the result sets of two or more SELECT statements.**

SELECT product\_name FROM products WHERE category = 'Electronics' UNION

SELECT product\_name FROM products WHERE category = 'Appliances';

**INTERSECT Operator:**

**The INTERSECT operator is used to return the common rows between two SELECT statements.**

SELECT product\_name FROM products(table name) WHERE category = 'Electronics'

INTERSECT

SELECT product\_name FROM products (tbale name) WHERE brand = 'Samsung';

**SQL EXISTS Operator:**

SELECT \* FROM employees WHERE EXISTS (

SELECT 1 FROM orders

WHERE orders.employee\_id = employees.employee\_id

);

# SELECT \* FROM Employees WHERE manager\_id IS NULL;

**Arithmetic Operators**

**Addition Operator**

UPDATE Betech SET Salary = Salary + 500 WHERE Sno = 1;

**Subtraction Operator**

UPDATE Betech SET Salary = Salary -500 WHERE Sno = 1;

**Multiply Operator**

UPDATE Betech SET Salary = Salary \*2 WHERE Sno = 1;

**Division Operator**

UPDATE Betech SET Salary = Salary /50 WHERE Sno = 1;

**Modulo Operator**

UPDATE Betech SET Salary = Salary %500 WHERE Sno = 1;

# SQL Clauses

**• WHERE Clause • WITH Clause • HAVING Clause • ORDER By Clause • Group By Clause • LIMIT Clause**

**SQL WHERE Clause:**

### The WHERE clause keyword in SQL is used to filter rows returned by a SELECT, UPDATE, or DELETE statement based on specified conditions.

1. **GREATER Than**

Select \* from table\_name where age>=20;

**2. Like**

Select \* from table\_name where country Like 'Usa';

**3. IN**

Select \* from table\_name where country in ('Usa','UK');

**4. EUQAL OPERATION**

Select \* from table\_name where id=1;

**5.Between OPERATION**

Select \* from table\_name where age between 20 and 30;

**6. SELECT Statement:**

This selects all columns from a table named users:

SELECT \* FROM users;

* **SQL LIMIT Clause:** Limits the number of rows returned by a query

SELECT \* FROM Customers where age>=25 limit 3; Select from table\_name limit 2;

### SQL ORDER By Clause

### The ORDER BY clause in SQL is used to sort the result set of a query in either ascending (ASC) or descending (DESC) order

**ORDER BY**

SELECT \* FROM Customers(tablename) ORDER BY last\_name;

**ORDER BY ASC**

SELECT \* FROM Customers ORDER BY age ASC;

**ORDER BY DESC**

SELECT \* FROM Customers ORDER BY age DESC;

**ORDER BY ASC and DESC**

SELECT \* FROM Customers ORDER BY age Asc , customer\_id DESC;

**Sql Group by Clause**

* Groups rows based on specified columns, often used with aggregate functions.

SELECT Column\_name, COUNT(\*) FROM table\_name GROUP BY Column\_name;

### SQL WITH Clause

### Sql Limit Clause

The LIMIT clause in SQL is used to restrict the number of rows returned by a query.

Select \* from Table\_name group by Name ASC Limit 5;

**SQL Having Clause**

Sql allow to filtering of query result based on aggregate function and grouping

SELECT Course, sum(Salary) FROM Betech  
GROUP BY Course HAVING SUM(Salary) >= 50000;

**Aggragate function**

Used to perform calculations on a set of values and return a single value.

**• count() • min() • max() • avg() • sum()**

### min()

select Min(Salary) from table\_name;

**max()**

select Max(Salary) from table\_name;

**count()**

select Count(Name) from table\_name;

### avg()

select AVG(Age) from table\_name;

**Q.select max(Salary),min(Salary),avg(Salary) ,sum(Salary) from employe\_data group by DepartmentID having sum(Salary)>60000**

* + **SQL Alias Command:**  The ALIAS command in SQL is used to give a temporary name to a table or a column in a query result,

SELECT name AS employee\_name, salary AS monthly\_salary FROM Employees;

**SQL Data Constraints**

A constraint in SQL is a rule applied to a column or set of columns in a table ensuring that data meets specific conditions, such as uniqueness, non-null values, or referential integrity.

* **NOT NULL Constraints • UNIQUE Constraint • Primary Key Constraint •Foreign Key Constraint •Composite Key •Alternate Key**

**DEFAULT Constraint :**

CREATE TABLE vivek (

ID INT UNIQUE, name VARCHAR(28), age INT DEFAULT 18

)

**NOT NULL Constraint : we can't kept empty id nll bacause constraint is not null**

CREATE TABLE vivek (

ID INT not null, name VARCHAR(28), age INT DEFAULT 18

)

**Unique Constraint :** all values in a column are unique, no duplicate values are allowed.

CREATE TABLE vivek (

ID INT UNIQUE, name VARCHAR(28), age INT DEFAULT 18

**)**

**Primary key :** Ensures that each row in a table is uniquely identified by a specified column its not null its combination of no null + unique

CREATE TABLE vivek (

ID INT primary, name VARCHAR(28), age INT DEFAULT 18

**)**

**SQL CHECK Constraints: all values in a column satisfy a specified condition.**

CREATE TABLE Employees ( EmpID int PRIMARYKEY, Age int CHECK (Age >= 18)

**);**

* **SQL Foreign Key Constraints : its refernce of another table link 2 table each other**

CREATE TABLE Orders (

OrderID int PRIMARY KEY, CustomerID int,

OrderDate date,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

**String Function**

**CONCAT:** Concatenates two or more strings together.

SELECT CONCAT (first\_name, ' ', last\_name) AS full\_name FROM employees;

**UPPER:** Converts all characters in a string to uppercase.

SELECT UPPER('hello') AS uppercase\_result; -- Output: "HELLO"

**LOWER:** Converts all characters in a string to lowercase.

SELECT LOWER('HELLO') AS lowercase\_result; -- Output: "hello"

**LENGTH (or LEN):** Returns the length of a string. SELECT LENGTH('Hello') AS string\_length; -- Output: 5

**LEFT:** Returns a specified number of characters from the start of a string. SELECT LEFT('Hello, World', 5) AS left\_string; -- Output: "Hello"

**RIGHT:** Returns a specified number of characters from the end of a string. SELECT RIGHT('Hello, World', 5) AS right\_string; -- Output: "World"

**Trim :** Remove leading and trailing space

SELECT TRIM(' Hello World ') AS trimmed\_string;

### Check Ascii value using sql

select ascii('a'); select ascii('A');

### character to ascii aplpha numeric

SELECT CHAR(97);

### Find character index

SELECT CHAR\_LENGTH("Priyanshu");

### Concat 2 sring

**select concat("Priya","anshu");**

### Comment in sql : 1. single-line comment 2. multi-line comment

-- This is a single-line comment

/\* This is a multi-line comment that spans across multiple lines \*/

**Indexing:** Indexing is a technique used to improve the performance of queries by creating data structures that allow for faster retrieval of data.

/\* Search any data before indexing \*/ CREATE CLUSTERED INDEX

idx\_clustered ON Empdatas(emp\_id); select \* from Empdatas where emp\_id=6

**JOIN**

A JOIN clause is used to combine rows from two or more tables based on a related column between them. It allows you to retrieve data from multiple tables in a single query

* **LEFT JOIN • RIGHT JOIN • CROSS JOIN • NATURAL JOIN • SELF JOIN • INNER JION • FULL JOIN • SELF JOIN**

**INNER JOIN :** The INNER JOIN returns only the rows that have matching values in both tables.

CREATE TABLE Employ (

EmpID INT PRIMARY KEY, EmpName VARCHAR(50), DepartmentID INT );

CREATE TABLE Departs(

DepartmentID INT PRIMARY KEY, DepartmentName VARCHAR(50) );

-- Insert data into Departs table

INSERT INTO Departs (DepartmentID, DepartmentName)

VALUES(1, 'Human Resources'), (2, 'Engineering'), (3, 'Marketing');

-- Insert data into Employ table

INSERT INTO Employ (EmpID, EmpName, DepartmentID)

VALUES (101, 'Alice Johnson', 1), (102, 'Bob Smith', 2), (103, 'Charlie Brown', 3), (104, 'Diana Green', 4), (105, 'Cameron Green', 5);

SELECT Employ.EmpID, Employ.EmpName, Departs.DepartmentName

FROM Employ

INNER JOIN Departs ON Employ.DepartmentID = Departs.DepartmentID;

**LEFT JOIN**

A LEFT JOIN in SQL is used to retrieve all records from the left table and the matching records from the right table If there is no match, the result will contain NULL values for the columns of the right table.

SELECT Employ.EmpID, Employ.EmpName, Departs.DepartmentName

FROM Employ

Left JOIN Departs ON Employ.DepartmentID = Departs.DepartmentID;

**RIGHT JOIN :**

A RIGHT JOIN in SQL is similar to a LEFT JOIN, but it retrieves all records from the right table and the matching records from the left table If no match is found, the result will contain NULL values for the columns of the left table.

SELECT Employ.EmpID, Employ.EmpName, Departs.DepartmentName

FROM Employ

Right JOIN Departs ON Employ.DepartmentID = Departs.DepartmentID;

**FULL OUTER JOIN**

A FULL JOIN in SQL combines the results of both LEFT JOIN and RIGHT JOIN. It retrieves all records when there is a match in either the left table, the right table, or both. If there is no match, NULL values are returned for columns where there is no data.

SELECT Employ.EmpID, Employ.EmpName, Departs.DepartmentName

FROM Employ FULL JOIN Departs ON Employ.DepartmentID = Departs.DepartmentID;

**CROSS JOIN :**

A CROSS JOIN in SQL produces a Cartesian product of the two tables, meaning that every row from the first table is combined with every row from the second table.

SELECT Employ.EmpID, Employ.EmpName, Departs.DepartmentName  
FROM Employ  
CROSS JOIN Departs;

**Self Join :**  SELF JOIN is a type of join where a table is joined with itself.

SELECT E1.EmpID AS EmployeeID, E1.EmpName AS EmployeeName, E2.EmpName AS ManagerName  
FROM Employ E1  
JOIN Employ E2 ON E1.DepartmentID = E2.DepartmentID;

**Natural Join :** A NATURAL JOIN in SQL automatically joins tables based on columns with the same name and data type in both tables.

SELECT E1.EmpID AS EmployeeID, E1.EmpName AS EmployeeName, E2.EmpName AS ManagerName  
FROM Employ E1  
JOIN Employ E2 ON E1.DepartmentID = E2.DepartmentID;

**SQL Keys**

A key is a set of attributes that can identify each tuple uniquely in the given relation

**Primary Key**: Uniquely identifies each record in a table.You can create only one primary key in a table

**Unique Key:** Ensures that all values in a column are unique.

**Foreign Key:** Links one table to another by referencing the primary key in the related table.

**Composite Key:** A combination of two or more columns that uniquely identify a record.

**Candidate Key:** Any column or set of columns that could be a primary key., But You can create muliple Cnadidate key in a table

**Super Key:** A set of one or more columns that uniquely identify a record in a table.

**Alternate Key:** A candidate key that is not chosen as the primary key.

**Primary key :** Ensures that each row in a table is uniquely identified by a specified column its not null its combination of no null +unique +Single primary key

CREATE TABLE vivek (

ID INT primary, name VARCHAR(28), age INT DEFAULT 18

**)**

**Unique key : all value must be unique and its allow one null value and you and create multiple unique**

CREATE TABLE Employees ( EmpID int PRIMARYKEY, Age int CHECK (Age >= 18) );

* **Foreign Key Constraints :** its refernce of another table link 2 table each otherits take reference by primary key

CREATE TABLE Customers (  
 CustomerID INT PRIMARY KEY, CustomerName VARCHAR(100)

);

CREATE TABLE Orders (  
 OrderID INT PRIMARY KEY,OrderDate DATE, CustomerID INT,  
 FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  
);

INSERT INTO Customers (CustomerID, CustomerName) VALUES  
(1, 'Alice Smith'),(2, 'Bob Johnson');

INSERT INTO Orders (OrderID, OrderDate, CustomerID) VALUES  
(101, '2024-08-15', 1),(102, '2024-08-16', 2);

SELECT o.OrderID, o.OrderDate, c.CustomerName  
FROM Orders o // retrived data into database  
JOIN Customers c ON o.CustomerID = c.CustomerID;

**Super Key:** A single key or a combination of multiple keys that can uniquely identify a tuple (row) in a relation (table) is called a super key. Ex= id, id+name , id+name+email

CREATE TABLE Employees (

FirstName varchar(50), LastName varchar(50),DateOfBirth date,

PRIMARY KEY (FirstName, LastName, DateOfBirth)

);

**Candidate Key :** A candidate key is a column or a set of columns in a table that can uniquely identify each row in that table.

its not null value but you can create mutiple candidate key Ex- id,Aadhar,Mobile no

CREATE TABLE Employees (

EmpID int,Email varchar(100),SSN varchar(20), PhoneNumber varchar(15),

PRIMARY KEY (EmpID), -- Candidate Key 1

UNIQUE (Email), -- Candidate Key 2

UNIQUE (SSN), -- Candidate Key 3

UNIQUE (PhoneNumber) -- Candidate Key 4

);

**Composite Key :** A composite key is a combination of two or more columns that together uniquely identify each row in a table.

CREATE TABLE Enrollments (

StudentID int,CourseID int,EnrollmentDate date,

PRIMARY KEY (StudentID, CourseID) -- Composite Key

);

**Alternate Key :** An **alternate key** is any candidate key that is not chosen as the **primary key** of a table.

CREATE TABLE Employees (

EmpID int, Email varchar(100), SSN varchar(20),

PhoneNumber varchar(15),

PRIMARY KEY (EmpID), -- Primary Key

UNIQUE (Email), -- Alternate Key 1

UNIQUE (SSN), -- Alternate Key 2

UNIQUE (PhoneNumber) -- Alternate Key 3

);

**Q. What is a database?**  **Theory Question**

A database is a structured collection of related data stored on a computer, organized in a way that allows easy store , access and manipulation of data. It includes schemas, tables,row , column and other objects,

### Application of database:

**Ans:** Company Information, Account information, manufacturing, banking, finance transactions, telecommunications,Heathcare

### SQL:

**Ans** : SQL stands for Structured Query Language and is a computer language that we use to interact with a relational database. SQL is a tool for *organizing*, *managing*, and *retrieving* archived data from a computer database.

### We use SQL for CRUD Operations:

* CREATE - To create databases, tables, insert tuples in tables etc
* READ - To read data present in the database.
* UPDATE - Modify already inserted data.
* DELETE - Delete database, table or specific data point/tuple/row or multiple row

**What is DBMS:** A Database Management System (DBMS) is software that enables users to create, manage, and organize databases. It provides an interface for users and applications to interact with the database by performing tasks such as storing, retrieving, updating, and deleting data

### Types of SQL Commands:

1. **DQL** (Data Query Language): Used to retrieve data from databases. (**SELECT**)
2. **DDL** (Data Definition Language): Used to create, alter, and delete database objects (C**REATE, DROP, ALTER, RENAME, TRUNCATE**)
3. **DML** (Data Manipulation Language): Used to modify the database. (**INSERT, UPDATE, DELETE**)
4. **DCL** (Data Control Language): Used to grant & revoke permissions. (**GRANT, REVOKE**)
5. **TCL** (Transaction Control Language): Used to manage transactions (**COMMIT, ROLLBACK, START TRANSACTIONS**)

### Data Definition Language (DDL)

Data Definition Language (DDL) is a subset of SQL (Structured Query Language) responsible for defining and managing the structure of databases DDL commands enable you to **create, modify, and delete, Alter, Drop, database** objects like tables,

### DATA QUERY/RETRIEVAL LANGUAGE (DQL or DRL)

DQL (Data Query Language) is a subset of SQL focused on retrieving data from databases. It includes commands such as **SELECT,** which are used to query and retrieve data from database tables. DQL allows users to specify the data they want to retrieve, apply filtering and sorting

### DATA MANIPULATION LANGUAGE

Data Manipulation Language (DML) is a subset of SQL (Structured Query Language) that allows users to manipulate data stored in a database. DML statements are used to perform operations such as **inserting, updating, deleting,** and querying data within database tables.

### Data Control Language (DCL)

Data Control Language (DCL) is a part of SQL (Structured Query Language) that deals with controlling access to data within a database.

**GRANT**: This command allows users to give specific permissions to other users or roles. For example, a database administrator can grant SELECT permission on a table to a specific user, allowing them to retrieve data from that table.

**REVOKE:** This command is like taking back a permission slip. For example, if you let someone borrow your book but then decide you need it back, you revoke their borrowing permission. Similarly, in a database,

### Transaction Control Language (TCL)

Transaction Control Language (TCL) deals with the management of transactions within a database. TCL commands are used to control the initiation, execution, and termination of transactions, which are sequences of one or more SQL statements that are executed as a single unit of work

**COMMIT:**

The COMMIT command is used to permanently save the changes made during a transaction

Note :If you make changes to the database (like adding, updating, or deleting data) without using commit those changes are temporary. in database

**ROLLBACK:**

The ROLLBACK command is used to undo changes made during a transaction. It reverts all the changes applied to the database since the transaction began.

**JOINS**

join is a fundamental operation used to combine rows from two or more tables based on a related column between them. It allows you to retrieve data from multiple tables simultaneously,

### Types of Joins:

1. Inner Join 2. Outer Join 3. Cross Join 4. Self-Join

**1. Inner Join:** only the rows from the participating tables where the join condition is satisfied for both tables are included in the result set. This means that if there is no matching row in one of the tables based on the specified join condition, that row will not appear in the final result.

**2) Outer Join:.** An outer join is a method of combining two or more tables where the result includes unmatched rows from one or both tables, depending on the type of outer join used.

### Types:

1. **Left Outer Join (Left Join):** A left outer join returns all the rows from the left table and the matching rows from the right table If there is no match in the right table, the result will still include the left table's row with NULL values in the right table's columns

### Right Outer Join (Right Join):

A right outer join is similar to a left outer join, but it returns all rows from the right table and the matching rows from the left table. If there is no match in the left table, the result will still include the right table's row with NULL values in the left table's columns.

### Full Outer Join (Full Join):

A full outer join returns all rows from both the left and right tables, including matches and non\_matche If there's no match, NULL values appear in columns from the table where there's no corresponding value.

### 3.Cross Join

A cross join, also known as a Cartesian product, is a type of join operation in a Database Management System (DBMS) that combines every row from one table with every row from another table.

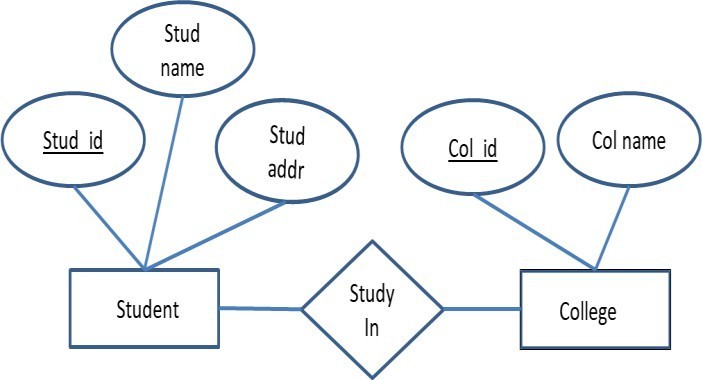
### 4) Self Join:

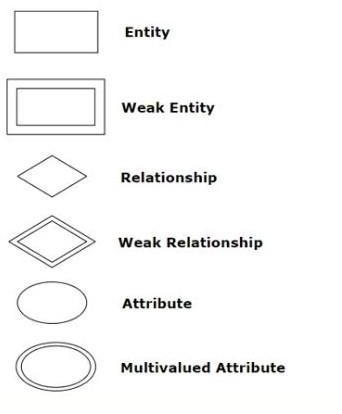
A self-join is a type of join operation where a table is joined by itself. This can be useful when you have a table that contains hierarchical or recursive data,

**Need of Joining table :**Joining tables is necessary in databases when one table doesn't have all the information needed for a query. By merging data from different tables using shared columns, joints help users get a complete dataset, revealing how different things in the database are related.

### ER diagram:

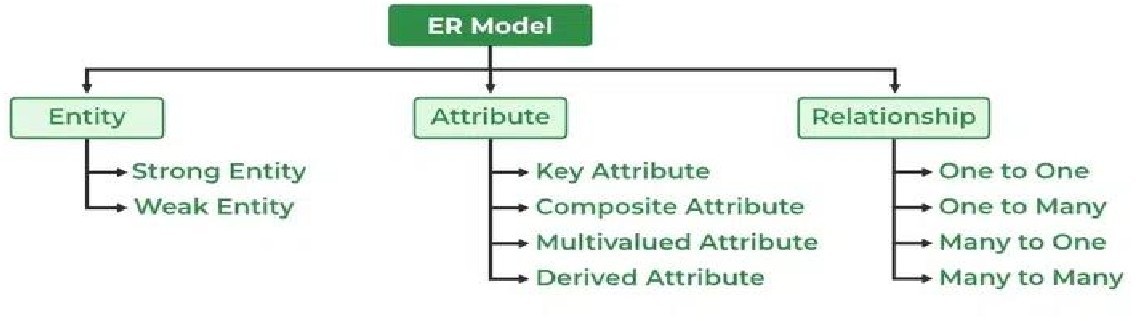
ER diagram or Entity Relationship diagram is a conceptual model that gives the graphical representation of the logical structure of the database. It shows all the constraints and relationships that exist among the different components

ER diagrams are used during the database design phase to visualize and understand the database schema.



### Components of ER Diagram:

**Why Use ER Diagrams In DBMS?**

ER diagrams are used to represent the E-R model in a database, which makes them easy to convert into relations (tables).

ER diagrams provide the purpose of real-world modeling of objects which makes them intently useful. ER diagrams require no technical knowledge and no hardware support.

These diagrams are very easy to understand and easy to create even for a naive user. It gives a standard solution for visualizing the data logically.

**Entity**

An Entity may be an object with a physical existence – a particular person, **car, house, or employee** – an entity can be represented as rectangles. it may be an object with a conceptual existence – a company, a job, or a university course.

### Entity Set :

1. **Strong Entity :** A **Strong Entity** is an entity that has a **primary key** and can exist independently of other entities in a database schema. Its primary key uniquely identifies each instance of the entity Ex- id,Dob,Aadhar no , Gender
2. **Week Entity:** A **Weak Entity** is an entity that cannot be uniquely identified by its own attributes and instead depends on a **strong entity** for its existence and identification. Ex- House no ,city , Loaction

### Attributes:

### Attributes are properties or characteristics of an entity. Attributes are used to describe the entity.

### Ex- **Student** is the **entity** **Name**, **Class**, and **Age** are the **attributes** of the Student entity

* **Simple Attributes -** Simple attributes are those attributes which cannot be divided further.its single value **Ex-**Age,Dod,Id

### Composite Attribute :An attribute composed of more than one value called a composite attribute. **For example,** the Address attribute of the student Entity type consists of Street, City, State, and Country.

* **Multi Valued** **Attributes** - Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set. Ex. Mobile No, EmailID , Passbook
* **Derived Attributes -** Derived attributes are those attributes which can be derived from other attribute(s). Ex. Age can be derived from DOB.
* **Key Attributes :** The attribute which uniquely identifies each entity in the entity set is called the key attribute. For example, Roll\_No , Student Id will be unique for each student.

### Relationship

A Relationship Type represents the association between entity types. For example, ‘Enrolled in’ is a relationship type that exists between entity type Student and Course.

1. **Unary Relationship:** When there is only ONE entity set participating in a relation, the relationship is called a unary relationship
2. **Binary Relationship:** When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.
3. **Ternary Relationship:** When there are n entities set participating in a relation, the relationship is called an n- ary relationship.

**Cardinality :**The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:

1. **One-to-One:** Each record in the first entity is related to a single record in the second entity **Example**: Each Person has one Passport,

**2.One-to-Many**: Each record in the first entity can be related to multiple records in the second entity, but each record in the second entity is related to only one record in the first entity.  **Ex- :** A Department can have multiple Employees, but each Employee belongs to only one Department.

**3.Many - to-one**

Multiple records in the first entity can be related to a single record in the second entity**, Ex-**Each Customer can place multiple Orders. Each Order can only be associated with a single Customer

**4.Many-to-Many:** When multiple records in the first entity are associated with multiple records in the second

**Ex-** Each **Student** can enroll in multiple **Courses**. Each **Course** can have multiple **Students** enrolled.

**Keys**

A key is a set of attributes that can identify each tuple uniquely in the given relation

**Type of key: 1. Candidate Key 2. Composite Key 3. Foreign Key 4. Super Key 5. Primary Key 6.Alternate Key**

**Primary Key:** The Primary Key uniquely identifies each record in a table. It must contain unique values and cannot have NULL values. There can be only one primary key in a table. **Example: Student'd, Aadhar number**

**Candidate Key :** A candidate key is a set of one or more columns in a table that can uniquely identify each record within that table. **Example:** Email, Student'd

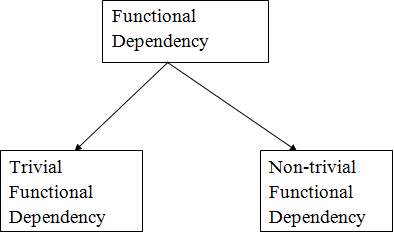
**Foreign Key :** A Foreign Key is a column or a set of columns on a table that establishes a relationship with a Primary Key or a Unique Key in another table. It enforces referential integrity between the two related tables.

**Super Key :** A Super Key is a set of one or more columns in a table whose combined values can uniquely identify each row in the table. It's a superset of a Candidate Key. **Example:** Student'd +Phone email, Rollno+name

**Composite Key :** A Composite Key, also known as a composite primary key or concatenated key, is a combination of two or more columns in a table that together uniquely identify each row in the table.

**Functional Dependency**

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table. **α → β**

****

* **Trivial Functional Dependencies**
* **Non-Trivial Functional Dependencies –**

**Normalization**

Normalization is a method for arranging data in a database to reduce duplication and errors. It involves splitting large tables into smaller ones to make data storage more efficient and maintain data accuracy

To avoid insertion , deletion ,update anomaly

1. Minimizing Redundancy
2. Minimizing the Insertion, Deletion, And Update Anomalies Relation

### Normal Forms:

* **First Normal Form (1NF) -** First Normal Form (1NF) if each cell of the table contains only an atomic value not repating value and an attribute can't be hold mutiple value
* **Second Normal Form (2NF) -** It should from of 1nf Second Normal Form (2NF) ensures that all non-prime attributes in a table are fully dependent on the entire primary key,
* **Third Normal Form (3NF) -**t should from of 2nf Third Normal Form (3NF), every non-prime attribute is directly dependent on the primary key, and there are no transitive dependencies between non-prime attributes.
* **BCNF:** Boyce-Codd Normal Form (BCNF) it eliminates all non-trivial functional dependencies where the determinant is not a superkey. This helps to further reduce redundancy and anomalies in the database schema, ensuring data integrity and consistency.

**Transaction**

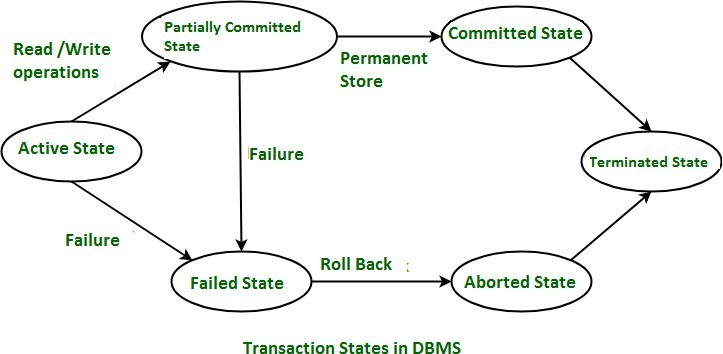
Transaction Based on Acid property A Database Transaction is a set of database operations that must be treated as a whole, which means either all operations are executed or none of them

**Example :**An example can be a bank transaction from one account to another account. Either both debit and credit

operations must be executed or neither of them.

### Operations in Transaction:

* **Read Operation** - Read(A) instruction will read the value of ‘A’ from the database and will store it in the buffer in main memory.
* **Write Operation** – Write(A) will write the updated value of ‘A’ from the buffer to the database



.

* **Partially Committed State –**
* After the last instruction of the transaction has been executed, it enter into a partially committed state.
* After entering this state, the transaction is considered to be partially committed.It is not considered fully committed because all the changes made by the transaction are still stored in the buffer in main memory.

### Committed State –

oAfter all the changes made by the transaction have been successfully store into the database it enters into a committed state

* Now, the transaction is considered to be fully committed.
* **Failed State –**
* When a transaction is getting executed in the active state or partially committed state and some failure occur due to which it becomes impossible to continue

the execution, it enters into a failed state**.**

### Aborted State –

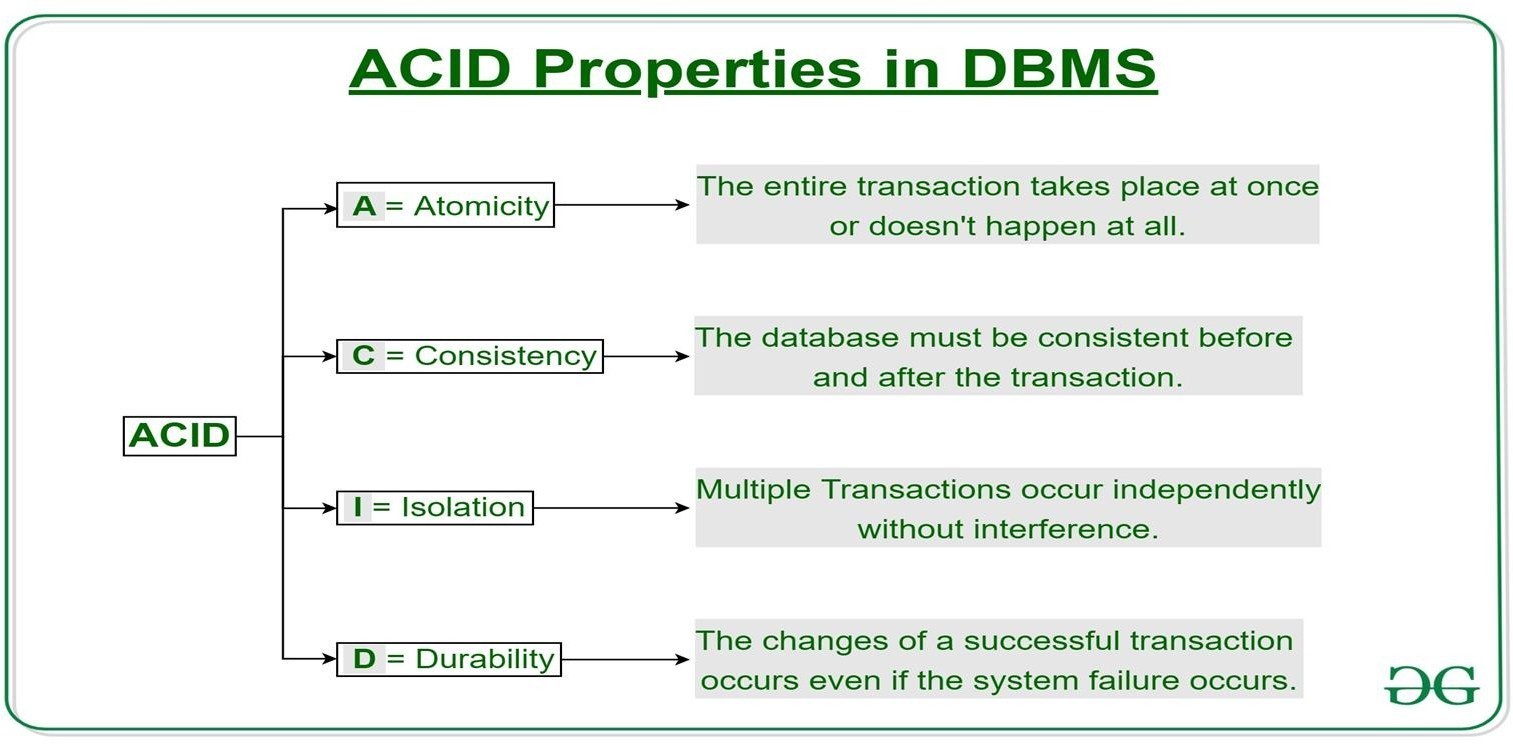
* After the transaction has failed and entered into a failed state, all the changes made by it have to be undone.

o to undo the changes made by the transaction, it becomes necessary to roll back the transaction.

* After the transaction has rolled back completely, it enters into an aborted sta**te.**

### Terminated State –

* This is the last state in the life cycle of a transaction.
* After entering the committed state or aborted state, the transaction finally enters into a terminated state where its life cycle finally comes to an end

**ACID Properties:** ACID is an acronym that stands for Atomicity, Consistency, Isolation, and Durability. These are the four key properties that ensure reliability and consistency of transactions in a database management system (DBMS)

### Atomicity –

o This property ensures that either the transaction occurs completely, or it does not occur at all.

o in other words, it ensures that no transaction occurs partially.

* + **Consistency –**Ensure Database properly change state upon a Sucessfully commited transaction
  + **Total before T occurs = 500 + 200 = 700.**

**Total after T occurs = 400 + 300 = 700.**

* **Isolation:** Transactions execute independently of each other, preventing interference between concurrent transactions.
* **Durability:** Once a transaction is committed, its changes are permanently saved, even in the event of system failures.

### Trigger in SQL

A **Trigger** in Structured Query Language is a set of procedural statements which are executed automatically when there is any response to certain events on the particular table in the database. Triggers are used to protect the data integrity in the database.

**Data Mining:** is the process of investigating hidden patterns of information to various perspectives for categorization into useful data, which is collected and assembled in particular areas such as data warehouses, efficient analysis, data mining algorithm, helping decision making

**Data Warehousing:** A data warehouse is a separate database system from the operational databases (DBMS) used by an organization. It is designed to store and manage large volumes of data collected from various sources such as operational databases, files, external data sources, etc. The primary goal of a data warehouse is to provide a consolidated, integrated, and historical view of an organization's data to support decision-making process**es.**

***Why is the use of DBMS recommended? Explain by listing some of its major advantages?***

* + **Controlled Redundancy:** DBMS Controlled redundancy in a database management system (DBMS) ensures that data is stored in a single place, preventing duplication. This integration of data into one central location eliminates redundancies,

### Data orgnized and Store in Database

* + **Handling multiple users simultaneously.**

### Data recovery and backup.

* + **Data Sharing:** Sharing of data among multiple users simultaneously can also be done in DBMS as the same database will be shared among all the users and by different application programs.
  + **Backup and Recovery Facility :**DBMS minimizes the pain of creating the backup of data again and again by providing a feature of ‘backup and recovery’ which automatically creates the data backup and restores the data whenever required.

**. What are indexes?** A database index is a specialized data structure that enhances the speed of data retrieval operations on a database table. It achieves this by creating a sorted copy of selected columns or fields from the table

**What is CLAUSE in SQL?** A clause in SQL is a part of a query that lets you filter or customize how you want your data to be queried to you.

**Database Schema :** A database schema is a logical representation of data that shows how the data in a database should be stored logically. It shows how the data is organized and the relationship between the tables.

* + Database schema contains table, field, views and relation between different keys like primary key, foreign key.

### There are 3 levels of data abstraction in the DBMS.

**View Level:** a view schema refers to the structure or definition of a view. A view is a virtual table created by a query that selects rows and columns from one or more tables. It presents the data in a customized way, often simplifying complex data structures

**Physical Level** The physical database schema describes how data is physically stored on the storage system. It includes details such as file organization and index creation. Essentially, it's the actual code or syntax used to create the database structure at the storage level.

**Logical Level:** The logical database schema defines the structure of the database at a conceptual level,ncluding tables, columns, relationships, and constraints,

**RDBMS:** RDBMS stands for Relational Database Management System. It's a software application used to manage relational databases, which store data in tables consisting of rows and columns.

### MySQL, Advantage:

Open Source: MySQL is freely available under the GNU General Public License (GPL). This makes it accessible to a wide range of users without any licensing costs.

High Performance: MySQL is optimized for fast data retrieval and processing. It offers high-performance indexing, caching mechanisms, and efficient query execution, making it suitable for demanding applications.

Cross-Platform Compatibility: MySQL is available for multiple platforms, including Windows, Linux, macOS, and Unix- like operating systems, ensuring compatibility with a wide range of environments.

Ease of Use: MySQL comes with user-friendly tools and utilities for database administration, management, and monitoring. Its simple installation process and intuitive interfaces make it easy for developers and administrators to work with.What is Serializability in DBMS?

A system is considered serializable if its outcome remains consistent regardless of how the operations are sequenced, as long as there is no overlap in their execution.

**Type of Serializability :** Conflict Equivalent Schedules: View Serializability:

### DBMS Integrity Constraints

Integrity constraints are the set of predefined rules that are used to maintain the quality of information. Integrity constraints ensure that the data insertion, data updating, data deleting, and other processes have to be performed in such a way that the data integrity is not affected.

**Entity Integrity Constraint**: This constraint ensures that each row (or entity) in a table has a unique and non-null primary key value, meaning that primary key attributes cannot contain null values and must be unique within the table.

1.

**Referential Integrity Constraint**: Also known as foreign key constraints, these constraints maintain the relationships

between tables by ensuring that values in a foreign key column (referencing another table's primary key) always correspond

to existing values in the referenced table's primary key column. This constraint prevents orphaned rows and maintains data consistency.

2.

**Domain Integrity Constraint**: Domain integrity constraints define the allowable values for columns in a table. These

constraints specify the data type, format, and range of values that can be stored in a column, ensuring that only valid data is

inserted into the database.

3.

**Check Constraint**: Check constraints define conditions that data must meet for insertion or updating operations to be

successful. These conditions can involve comparisons, logical expressions, or predefined functions to validate data integrity.

4.

**Unique Constraint**: Unique constraints ensure that values in a specified column (or combination of columns) are

unique across all rows in a table, except for null values. This constraint prevents duplicate entries in the table.

5.

**Assertion Constraint**: Assertion constraints define conditions that must be true for the entire database, rather than for

individual tables or columns. These constraints are typically used to enforce complex business rules that involve multiple

tables or rows.

**Difference between where and having clause**

|  |  |  |
| --- | --- | --- |
| **Point** | **WHERE Clause** | **HAVING Clause** |
| 1. Filters rows before groups are aggregated | Yes, filters rows before aggregation process | No, filters groups after aggregation process |
| 2. Usage with GROUP BY Clause | Can be used without GROUP BY Clause | Can be used with GROUP BY Clause |
| 3. Scope of implementation | Implements in row operations | Implements in column operation |
| 4. Usage with SQL statements | Can be used with SELECT, UPDATE, DELETE statement | Can only be used with SELECT statement |
| 5. Usage with functions | Used with single row functions like UPPER, LOWER, etc. | Used with multiple row functions like SUM, COUNT, etc. |

# SQL vs NoSQL

SQL NoSQL

SQL database Called Relational Database NoSQL Called Non-Relational database SQL Database is table-based data store NoSQL Database is Document based data Stored Ex- Oracle, Postgres, SQL Server Ex – MongoDB, Cassandara

These databases have fixed or static or They have a dynamic schema

### Type of Database

**These databases have fixed or static or predefined schema Relational Databases (RDBMS):**

Relational databases store and manage data in tables with rows and columns. They use Structured Query Language (SQL) for querying and manipulating data.

Examples include MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, and SQLite.

### NoSQL Databases:

NoSQL databases are non-relational databases that provide flexible schema design and are suitable for handling unstructured or semi- structured data.

They can be document-oriented, key-value stores, column-family stores, or graph databases. Examples include MongoDB, Couchbase, Cassandra, Redis, and Neo4j.Object-Oriented Databases (OODBMS):

object-oriented databases store data as objects, allowing for complex data structures and relationships. They are suitable for applications with complex data models and object-oriented programming languages. Examples include db4o and Object DB

### Graph Databases:

Graph databases are optimized for storing and querying graph data structures, consisting of nodes, edges, and properties. They are suitable for applications that require traversing relationships between entities.

Examples include Neo4j, Amazon Neptune, and Janus Graph.

### Columnar Databases:

**In-Memory Databases:**

### Time-Series Databases:

Time-series databases specialize in storing and analyzing time-stamped data, such as sensor data, IoT data, and financial data. They optimize storage and retrieval for time-series data patterns.

Examples include Influx DB, Prometheus, and Timescale DB.

**Aggregation**: Aggregation in the context of databases refers to the process of applying a function (or operation) to multiple rows of data and returning a single summary value. This summary value represents some aspect of the dataset, such as a total, average, count, maximum, or minimum, Sum value.

### What is the difference between primary key and unique constraints?

The primary key cannot have NULL value, the unique constraints can have NULL values. There is only one primary key in a table, but there can be multiple unique constraints.

**Clause In Dbms :**

**.A clause in SQL is a part of a query that specifies certain conditions, actions, or constraints to customize the retrieval or manipulation of data from a database.**

**Example: Select, Where Order by, Group by, Having Clause**

### What is Denormalization?

**Denormalization is a database optimization technique in which we add redundant data to one or more tables.**

| ****Criteria**** | | | ****CHAR**** | | | | ****VARCHAR**** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Length** | | | | CHAR datatype is used to store character strings of fixed length | | | VARCHAR datatype is used to store character strings of variable length |
| **Padding** | | | Pads with spaces if shorter than set length it occupies extra space and unused space. | | | | No padding, stores as isefficiently uses space by releasing any unused portion |
| **Meaning** | Stands for "Character" | | | | | | Stands for "Variable Character" |
| **Performance** | | Faster for fixed-length data | | | | | Slightly slower due to variable length |
| **Performance** | | | | | Faster for fixed-length data | Slightly slower due to variable length | |