Database :

**A database is a collection of organized and structured data that can be accessed, managed, and updated electronically.**

**Relational database stored data in form of ta bles that can easily managed and updated.**

**database is control By DBMS.**

There are many popular databases used in the industry:

**MySQL :** Open-source relational database management system used in web applications and powering many of the world's largest websites

**Oracle**: Commercial relational database management system used in enterprise applications and large-scale systems.

**Microsoft SQL Server:** Relational database management system developed by Microsoft for Windows-based systems, widely used in enterprise applications.

**PostgreSQL**: Open-source object-relational database management system known for its extensibility and advanced features.

**MongoDB:** Document-oriented NoSQL database used for handling large volumes of unstructured data, often used in real-time applications.

**Cassandra:** Distributed NoSQL database designed to handle large amounts of data across multiple commodity servers and data centers.

**Redis :** In-memory key-value store used as a database, cache, and message broker, popular for its high performance and simplicity.

**SQLite:** Self-contained, serverless, public-domain SQL database engine, often used in embedded systems or mobile applications.

**IBM DB2:** Relational database management system developed by IBM, used in enterprise-level systems and mainframe environments

…..

**What is SQL?**

**SQL is the lang. to communicate with databases .SQL commands help you to store ,process ,analyse ,and manipulate databases**.

**SQL (Structured Query Language) was developed by IBM Corporation in the 1970s**

**In SQL, a field refers to a single unit of data stored in a table, which corresponds to a column in the table and represents a specific attribute or property of the table's data.**

| **SQL** | **NoSQL** |
| --- | --- |
| Structured Query Language | Not only SQL |
| Relational databases with fixed schema | Non-relational databases with flexible schema |
| Suitable for structured data | Suitable for semi-structured and unstructured data |
| ACID transactions (Atomicity, Consistency, Isolation, Durability) | BASE properties (Basically Available, Soft state, Eventual consistency) |
|  |  |
| Supports JOIN operations | Does not support JOIN operations |
| Good for complex queries | Good for simple queries and frequent write operations |
| Examples: MySQL, Oracle, SQL Server | Examples: MongoDB, Cassandra, Redis |
| Data stored in tables | Data is stored in various formats such as key-value pairs, documents, graphs, etc. |
| Suitable for structured data | Suitable for semi-structured and unstructured data |
|  |  |
| Good for complex queries including JOINs | Good for simple queries and frequent write operations |
| Uses a structured data model | Uses a flexible data model |

| **Type of SQL Command** | **Description** |
| --- | --- |
| DDL (Data Definition Language) | Used to define and manage the database structure, including creating and modifying tables, indexes, and constraints. Common commands include CREATE, ALTER, DROP, TRUNCATE, and RENAME. |
| DML (Data Manipulation Language) | Used to manipulate the data stored in the database, including adding, updating, and deleting records. Common commands include INSERT, UPDATE, DELETE, and MERGE. |
| DQL (Data Query Language) | Used to retrieve data from one or more tables in the database. The most common DQL command is SELECT. |
| DCL (Data Control Language) | Used to control access to the database, including granting and revoking user privileges and permissions. Common commands include GRANT and REVOKE. |
| TCL (Transaction Control Language) | Used to manage transactions within the database. Common commands include COMMIT, ROLLBACK, and SAVEPOINT. |

| **Data Type** | **Description** | **Group** |
| --- | --- | --- |
| INTEGER | A whole number value, typically stored as a 32-bit signed integer. | Numeric |
| BIGINT | A large integer value, typically stored as a 64-bit signed integer. | Numeric |
| DECIMAL, NUMERIC | A fixed-point number value with a specified precision and scale. | Numeric |
| FLOAT | A floating-point number value with a specified precision. | Numeric |
| REAL | A floating-point number value with a precision of 7 digits. | Numeric |
| CHAR | A fixed-length character string with a specified length. | Character |
| VARCHAR | A variable-length character string with a maximum length specified. | Character |
| TEXT | A variable-length character string with no maximum length. | Character |
| NCHAR | A fixed-length Unicode character string with a specified length. | Character |
| DATE | A date value in the format 'YYYY-MM-DD'. | Date/Time |
| TIME | A time value in the format 'HH:MI:SS'. | Date/Time |
| DATETIME | A date and time value in the format 'YYYY-MM-DD HH:MI:SS'. | Date/Time |
| TIMESTAMP | A date and time value with a higher precision than DATETIME. | Date/Time |
| BOOLEAN | A boolean value with two possible values: TRUE or FALSE. | Logical |
| BINARY | A fixed-length binary string with a specified length. | Binary |
| VARBINARY | A variable-length binary string with a maximum length specified. | Binary |
| BLOB | A variable-length binary string with no maximum length. | Binary |

* Numeric: Data types that store numerical values.
* Character: Data types that store character or text values.
* Date/Time: Data types that store date and/or time values.
* Logical: Data types that store true/false or yes/no values.
* Binary: Data types that store binary data, such as images or audio files.
* Arithmetic: Operators used for mathematical operations on numeric values.
* Comparison: Operators used to compare two values and return a boolean result.
* Logical: Operators used to combine or negate conditions and return a boolean result.
* String: Operators used for string manipulation and pattern matching

| **Type** | **Operator** | **Description** |
| --- | --- | --- |
| Arithmetic | + | Addition |
|  | - | Subtraction |
|  | \* | Multiplication |
|  | / | Division |
|  | % | Modulo (remainder) |
| Comparison | = | Equal to |
|  | <> or != | Not equal to |
|  | < | Less than |
|  | > | Greater than |
|  | <= | Less than or equal to |
|  | >= | Greater than or equal to |
| Logical | AND | Returns true if both conditions are true |
|  | OR | Returns true if either condition is true |
|  | NOT | Negates a condition |
| String | CONCAT | Concatenates two strings |
|  | LIKE | Searches for a pattern within a string |
|  | IN | Specifies multiple possible values for a column |
|  | NOT IN | Specifies multiple excluded values for a column |

🔴🔴🔴

What is SQL?

SQL is the lang. to communicate with databases .SQL commands help you to store ,process ,analyse ,and manipulate databases.

SQL (Structured Query Language) was developed by IBM Corporation in the 1970s

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

In SQL, a field refers to a single unit of data stored in a table, which corresponds to a column in the table and represents a specific attribute or property of the table's data.

.............................................

🔴🔴🔴FEATURES :

1) SQL lets you access any data within the relational database.

2) SQL is very fast in retrieving large amount of data very efficiently

3)SQL is versatile aas it works with database systems from oracle , ibm and microsoft

4)SQL helps you manage databases without knowing lot of coding

5)SQL is extensively used as client /Server language to connect the front end wiht back -end thus supporting the client /Server architecture

What is ER diagram?

An ER diagram describes the relationship of entities that need to be stored in db.

ER diagram is mainly a structural design for the database .It is a framework ,using specialized symbols to define the relation between entites.

ER diagram is created based on three components entites ,attributes and relationships.

..................

why use ER diagram:

1)Helps us conceptualize the db and help us to know which filed need to be embedded for a particular entity.

2)ER diagram gives a better understanding of the info to be stored in a db

3)Reduces complexity and saves time which follows you to build db quickly

4)Helps to describe elements using ER model

/..............................................

Entities: Rectangles

Attributes: Ovals

Relationships: Diamonds

Cardinality: Lines

Participation: Marks

Primary Key: Underline

Foreign Key: Dashed underline

Multivalued attributes: Double ovals

Derived Attributes: dashed oval

Weak entities: Double rectangles

Identifying relationship: Double diamond

Superclass/Subclass relationship: Triangle

Union Type: Circle

Optional relationship: Dashed line

Mandatory relationship: Solid line

An ER (Entity-Relationship) diagram is a graphical representation of entities and their relationships to each other in a database.

💯💯💯💯💯💯💯💯💯💯

Entities: The objects or concepts that are represented in the database are called entities. Entities are usually represented as rectangles with the name of the entity inside

**1)Strong Entity:** A strong entity is an entity that exists independently of any other entities in the database. It has its own primary key and can be uniquely identified. Example: A "Person" entity in a database that stores information about individual

**2)Weak Entity:** A weak entity is an entity that cannot exist without a relationship with a strong entity. It does not have its own unique primary key, but instead relies on the primary key of the related strong entity. Example: A "Room" entity in a hotel management database that cannot exist without being associated with a "Hotel" entity.

**3) Associative Entity**: An associative entity is an entity that is created to represent a many-to-many relationship between two or more other entities. It has its own attributes and a composite primary key made up of the primary keys of the related entities. Example: A "Sales" entity in a database that tracks sales transactions between "Customer" and "Product" entities.

...................................................

Attributes: An attribute is a characteristic or property of an entity. Attributes are represented as ovals connected to the entity rectangle by a line. The name of the attribute is written inside the oval.

1)**Simple Attribute:** A simple attribute is an attribute that cannot be further subdivided into smaller parts. Example: A "Name" attribute for a "Person" entity.

2)**Composite Attribute:** A composite attribute is an attribute that can be further subdivided into smaller parts. Example: An "Address" attribute for a "Person" entity that consists of multiple parts, such as "Street," "City," and "Zip Code."

3)**Derived Attribute:** A derived attribute is an attribute that can be calculated or derived from other attributes in the database. Example: A "Total Price" attribute for an "Order" entity that is calculated by multiplying the "Price" attribute by the "Quantity" attribute.

4)**Multivalued Attribute:** A multivalued attribute is an attribute that can have multiple values for a single instance of an entity. Example: A "Phone Number" attribute for a "Person" entity that can have multiple phone numbers.

5)**Stored Attribute:** A stored attribute is an attribute that is physically stored in the database. Example: A "Date of Birth" attribute for a "Person" entity

6)**Null Attribute**: A null attribute is an attribute that does not have a value for a particular instance of an entity. Example: A "Middle Name" attribute for a "Person" entity that is not filled out for some individuals.

**A simple attribute is a non-key attribute in a database table, whereas a key attribute is a column or set of columns that uniquely identifies each row in a table.**

**A key attribute is an attribute or combination of attributes that uniquely identifies an instance of an entity in a database (roll no)**

**the text of key attribute is underlined.**

**🔴Primary Key: A primary key is a single attribute or combination of attributes that uniquely identifies each instance of an entity in a database. Example: In a "Customer" entity, the "CustomerID" attribute could be used as the primary key to uniquely identify each customer.**

**🔴Candidate Key:** A candidate key is an attribute or combination of attributes that could be used as a primary key for an entity, but is not currently being used as the primary key. Example: In a "Person" entity, both the "Social Security Number" and "Passport Number" attributes could be used as candidate keys to uniquely identify each person.

**🔴Foreign Key: A foreign key is an attribute in one entity that refers to the primary key of another entity. Example: In an "Order" entity, the "CustomerID" attribute could be a foreign key that refers to the "CustomerID" attribute in a "Customer" entity.**

Relationships: The connections between entities are called relationships. Relationships are represented as diamonds connected by lines to the entities they relate. The name of the relationship is written inside the diamond.

1)One-to-One (1:1) Relationship: This is a relationship between two entities where one instance of an entity is associated with only one instance of another entity. Example: A "Person" entity and a "Driver's License" entity, where each person can have only one driver's license

2)One-to-Many (1:N) Relationship: This is a relationship between two entities where one instance of an entity is associated with one or more instances of another entity. Example: A "Department" entity and an "Employee" entity, where one department can have many employees.

3)Many-to-Many (M:M) Relationship: This is a relationship between two entities where one instance of an entity is associated with many instances of another entity, and vice versa. Example: A "Student" entity and a "Course" entity, where each student can take multiple courses and each course can have multiple students.

4)Recursive Relationship: This is a relationship between a single entity and itself. Example: A "Manager" entity and an "Employee" entity, where an employee can be a manager of other employees.

5)Associative Relationship: This is a relationship between three or more entities, where an associative entity is created to represent the relationship. Example: A "Sales" entity that connects a "Customer" entity, a "Product" entity, and a "Store" entity to track sales transactions.

6)A many-to-one relationship is a type of relationship in which multiple records from one table can be associated with a single record in another table. This relationship is also known as a "belong-to" or "parent-child" relationship.

In an online bookstore database, an author can write multiple books, but each book is written by only one author (many books to one author relationship).

student enroll for course (many to one)

Type of SQL Command Description:

......................................

DDL (Data Definition Language) :Used to define and manage the database structure, including creating and modifying tables, indexes, and constraints. Common commands include CREATE, ALTER, DROP, TRUNCATE, and RENAME.

DDL (Data Definition Language) is a set of SQL commands used to define and manage the structure of a database. Here are some properties of DDL:

Used to define database structure: DDL commands are used to create, modify, and delete database objects such as tables, indexes, and constraints.

Permanent changes: DDL commands make permanent changes to the database structure and are not easily reversible. They are typically used by database administrators to manage the schema of a database.

Can impact data: DDL commands can impact the data stored in the database, as changes to the structure can affect how data is stored and retrieved. It is important to use DDL commands carefully and test their impact before executing them on a production database.

Requires higher privileges: DDL commands require higher privileges than other SQL commands, as they can modify the structure of the database. Typically, only database administrators and users with special privileges are allowed to execute DDL commands.

Examples of DDL commands: Some examples of DDL commands include CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, ALTER INDEX, DROP INDEX, CREATE VIEW, ALTER VIEW, and DROP VIEW.

........................

....

DML (Data Manipulation Language) :Used to manipulate the data stored in the database, including adding, updating, and deleting records. Common commands include SELECT(DQL),INSERT, UPDATE, DELETE, and MERGE.

DML (Data Manipulation Language) is a set of SQL commands used to manipulate the data stored in a database. Here are some properties of DML:

Used to manipulate data: DML commands are used to insert, update, delete, and retrieve data from tables in a database.

Temporary changes: DML commands make temporary changes to the data stored in the database, and the changes can be undone using the appropriate DML command.

Data integrity: DML commands can affect data integrity, so it is important to use them carefully and ensure that the changes made are consistent with the constraints and rules defined in the database schema.

Requires lower privileges: DML commands typically require lower privileges than DDL commands, as they do not modify the database structure. Users with appropriate privileges can manipulate data in tables, subject to the constraints and rules defined in the schema.

Examples of DML commands: Some examples of DML commands include SELECT, INSERT, UPDATE, DELETE, and MERGE. These commands are used to manipulate data in tables, and they can be used in combination with other SQL commands to perform complex operations on the data stored in a database.

.................................

.......................................

DQL (Data Query Language): Used to retrieve data from one or more tables in the database. The most common DQL command is SELECT.

DQL (Data Query Language) is a set of SQL commands used to retrieve data from one or more tables in a database. Here are some properties of DQL:

Used to retrieve data: DQL commands are used to retrieve data from tables in a database, based on specified conditions and criteria.

Read-only commands: DQL commands are read-only commands, meaning they do not modify the data stored in the database.

Powerful queries: DQL commands allow for the creation of powerful queries that can retrieve data from multiple tables and join them based on specified conditions.

Data manipulation: While DQL commands are primarily used to retrieve data, they can also be used to manipulate the data retrieved using aggregate functions, sorting, grouping, and other techniques.

Examples of DQL commands: Some examples of DQL commands include SELECT, JOIN, UNION, INTERSECT, and EXCEPT. These commands are used to retrieve data from tables and join multiple tables based on specified conditions, creating complex queries to retrieve data from a database.

..........................

DCL (Data Control Language) :Used to control access to the database, including granting and revoking user privileges and permissions. Common commands include GRANT and REVOKE.

DCL (Data Control Language) is a set of SQL commands used to control access to data stored in a database. Here are some properties of DCL:

Used to control access: DCL commands are used to control access to data stored in a database by creating users and granting or revoking privileges.

Higher-level commands: DCL commands are higher-level commands than DML commands, as they deal with access control and user management rather than data manipulation.

Permanent changes: DCL commands make permanent changes to the database by creating users and granting or revoking privileges, and these changes can only be undone using the appropriate DCL command.

Security management: DCL commands are used to manage the security of a database by creating users with appropriate privileges and ensuring that only authorized users have access to the data.

Examples of DCL commands: Some examples of DCL commands include GRANT, REVOKE, CREATE USER, DROP USER, and ALTER USER. These commands are used to create and manage users and to grant or revoke privileges to access data in a database.

........................................

................................

TCL (Transaction Control Language) :Used to manage transactions within the database. Common commands include COMMIT, ROLLBACK, and SAVEPOINT.

TCL (Transaction Control Language) is a set of SQL commands used to manage transactions in a database. Here are some properties of TCL:

Used to manage transactions: TCL commands are used to manage transactions in a database by controlling the flow of transactions, committing or rolling back changes, and setting savepoints.

Higher-level commands: TCL commands are higher-level commands than DML commands, as they deal with transaction management rather than data manipulation.

Temporary changes: TCL commands make temporary changes to the database by managing transactions, and the changes can be undone using the appropriate TCL command.

Ensures data integrity: TCL commands ensure data integrity by managing transactions and ensuring that changes made to the database are consistent with the rules and constraints defined in the database schema.

Examples of TCL commands: Some examples of TCL commands include COMMIT, ROLLBACK, SAVEPOINT, and SET TRANSACTION. These commands are used to manage transactions in a database and ensure that changes made to the database are consistent and reliable.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*🔴🔴🔴🔴🔴🔴🔴🔴🔴🔴🔴

//////////////////////////////////////////////////////////////

SELECT column\_name1, column\_name2, ...

FROM table\_name

WHERE condition

GROUP BY column\_name

HAVING condition

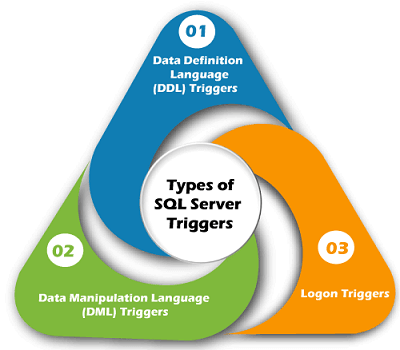
ORDER BY column\_name ASC/DESC;

//////////////////////////////////////////////

| **WHERE Clause** | **HAVING Clause** |
| --- | --- |
| **Purpose** | **Filters individual rows before grouping** | **Filters grouped results after grouping** |
| **Used with** | **SELECT, UPDATE, DELETE statements** | **SELECT statement with GROUP BY clause** |
| **Filtering** | **Filters individual rows** | **Filters groups of rows** |
| **Placement** | **Comes before the GROUP BY clause** | **Comes after the GROUP BY clause** |
| **Columns** | **Can filter based on any column** | **Can filter based on columns used in GROUP BY and aggregates** |
| **Aggregates** | **Cannot use aggregate functions** | **Can use aggregate functions (COUNT, SUM, AVG, etc.)** |
| **Comparison** | **Compares values row by row** | **Compares values across groups** |
| **Syntax** | **WHERE condition** | **HAVING condition** |
| **Example** | **SELECT \* FROM table WHERE column = value** | **SELECT column, COUNT(\*) FROM table GROUP BY column HAVING COUNT(\*) > 5** |

A subquery is a query that appears inside another query statement. Subqueries are also referred to as sub- SELECT s or nested SELECT s

In SQL, a subquery is a query that is nested within another query. It allows you to use the result of one query as a part of another query. Subqueries can be used in various parts of a SQL statement, including the SELECT, FROM, WHERE, and HAVING clauses.



|  |  |
| --- | --- |
| **UNION** | **UNION ALL** |
| It combines the result set from multiple tables and **returns distinct records** into a single result set. | It combines the result set from multiple tables and **returns all records** into a single result set. |
| Following is the basic **syntax** of UNION operator: SELECT column\_list FROM table1 UNION SELECT column\_list FROM table2; | Following is the basic **syntax** of UNION ALL operator: SELECT column\_list FROM table1 UNION ALL SELECT column\_list FROM table2; |
| It has a default feature to eliminate the duplicate rows from the output. | It has no feature to eliminate the duplicate rows from the output. |
| Its **performance is slow** because it takes time to find and then remove duplicate records. | Its **performance is fast** because it does not eliminate the duplicate rows. |
| Most database users prefer to use this operator. | Most database users do not prefer to use this operator. |

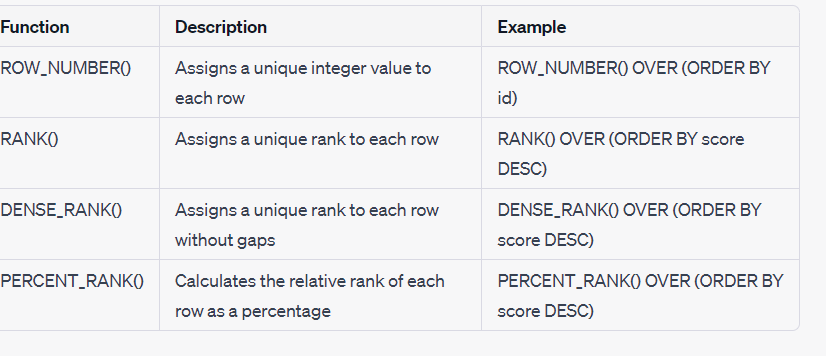
|  |  |  |
| --- | --- | --- |
| **Features** | **Power BI** | **Excel** |
| Tabular Reports | It creates relatively limited tabular reports. | It’s ideal for making tabular reports. |
| Duplicate Tables | Can’t display duplicated tables | Allows you to display duplicated tables |
| Reports | More visually appealing, customized, appealing, and interactive reporting. | Reports are simpler and less appealing than those in Power BI. |
| Crossed Filters | Supports powerful chart cross-filtering features. | There is no advanced graphics cross-filtering. |
| Charts and Visuals | Dashboards, alarms, and KPIs work best. Includes richer graphics than Excel and allows for visual data analysis. | It contains the most powerful and cutting-edge charting tools, however it isn’t compatible with data models. |
| Automatic Updates | Data is automatically updated. | Data is not automatically updated. |
| Availability | Repots can be worked on by a huge number of people, regardless of their expertise. | The number of users who can see a report is limited. |
| Analytics | Power BI has fewer data analysis possibilities. | Excel has more advanced analytical capabilities. |
| Data Model | Ideal for quickly creating complex data models. | Works with simple and structured data models. |
| Separate Tables | Separate tables can be linked together easily. | Connecting various tables is tough. |
| Tools | It is a more complex version of the data analysis tool, with more options for working with data. | It’s a standard spreadsheet tool with a lot of options. |
| Collaborative Work | Power BI makes it simple to share data and reports. | Sharing documents and working with others is complex. |
| Big Data | Allows working with significantly bigger data sets. | Can only handle a certain amount of info. |
| Dashboards | More advanced features for creating custom dashboards. | Users have limited features for creating dashboards. |
| Processing | Faster processing. | Slower processing. |
| Utility | Dashboards can be created and shared, as well as advanced data visualizations. | Typically, it is used to arrange data, execute calculations, and build more complex tabular reports. |
| Data Model Language | DAX language | MDX language |
| Connectivity | Data can be extracted from any virtual platform, software, or application. | Connectivity to other apps and systems is limited. |
| Price | It has [a free version](https://dynamics.folio3.com/blog/difference-between-power-bi-pro-vs-free-vs-premium/) and a payment version. | Payment Tool. |
| Usability | Easy to use compared to Excel | More difficult to use than Power BI |

**The LIMIT clause is used in SQL to specify the number of rows to be returned in a query result. It is commonly used to control the result set size .**

**SELECT column1, column2, ...**

**FROM table\_name**

**LIMIT number\_of\_rows;**

****

In SQL, CTE stands for Common Table Expression. It is a temporary named result set that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement. CTEs are particularly useful when you want to break down a complex query into smaller, more manageable parts.

**…………………………………………………………………………………………………………….**

**BY RISHABH MISHRA**

**Q) What is SQL ?**

SQL (Structured Query Language) is a programming language used for managing and manipulating relational databases.

……………………………………………………………………………………………………………………………………………..

**Q)** **What are the subsets of SQL?**

T-SQL(Transact-SQL): Proprietary extension of SQL used by Microsoft SQL Server and Sybase.

PL/SQL (Procedural Language/SQL): Proprietary extension of SQL used by Oracle Database.

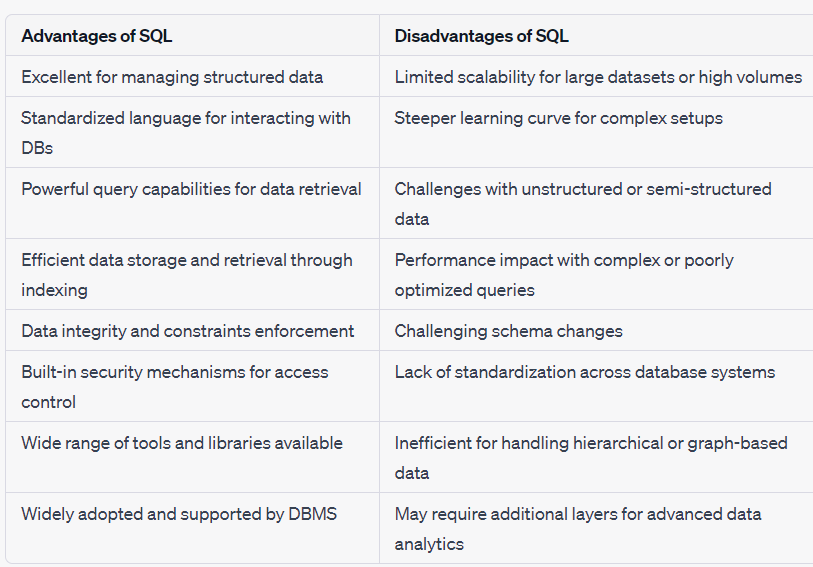
MySQL: Open-source relational database management system with its variant of SQL.

PostgreSQL: Open-source relational database management system that supports SQL.

SQLite: Lightweight, file-based relational database engine with a self-contained subset of SQL.

……………………………………………………………………………………………………………………………………………….

**Q)** A**dvantages & disadvantage of SQL?**

****

**Q) What is Datbase ?**

DBMS stands for Database Management System, which is software that enables users to interact with databases, manage data, and ensure secure and efficient storage and retrieval. It provides features like data definition, manipulation, security, integrity, transaction management, and concurrency control.

………………………………………………………………………………………………..

**Q) What are Tables and Fields?**

**Tables:** Tables are the main building blocks of a database. They represent logical structures that hold data in rows and columns. A table is comprised of rows (also known as records or tuples) and columns (also known as fields or attributes). Each row in a table represents a specific entity or record, while each column represents a specific attribute or piece of data associated with that entity. Tables provide the structure to store and organize data in a tabular format, making it easier to query and manipulate.

**Fields:** Fields, also known as columns or attributes, are the individual data elements within a table. Each field represents a specific type of information related to the entities or records stored in the table. For example, in a "Customers" table, fields may include "Customer ID," "Name," "Email," and "Phone Number." Fields define the type of data that can be stored (such as numeric, string, date, etc.) and provide the labels for the data stored in each column of the table.

…………………………………………………………………………………………………………………………….

**Q) What are Constraints?**

Constraints in SQL are rules or conditions that are applied to the columns of a table to enforce data integrity and maintain consistency. They define the allowable values and relationships within a table. Constraints ensure that data entered into a table adheres to specified rules, preventing the insertion of invalid or inconsistent data.

Here are some commonly used constraints in SQL:

1. **Primary Key Constraint**: A primary key constraint ensures that each row in a table has a unique identifier. It prevents duplicate or null values in the specified column(s). Only one primary key constraint can be defined per table.
2. **Foreign Key Constraint:** A foreign key constraint establishes a relationship between two tables based on a column(s) in each table. It ensures that values in the foreign key column(s) in one table exist as primary key values in another table. This maintains referential integrity and enforces data consistency between related tables.
3. **Unique Constraint:** A unique constraint ensures that the values in a specified column(s) are unique and distinct. It prevents duplicate values but allows null values.
4. **Check Constraint:** A check constraint defines a condition that must be satisfied for data in a column(s). It restricts the values allowed in the column(s) based on a specified condition or expression. For example, a check constraint can limit the range of acceptable values or enforce specific patterns.
5. **Not Null Constraint**: A not null constraint specifies that a column(s) cannot contain null values. It ensures that the specified column(s) must have a value for every row in the table

**By using constraints, SQL databases can maintain the integrity and consistency of data, enforce relationships between tables, and prevent the insertion of invalid or inconsistent data. Constraints play a crucial role in ensuring data reliability and accuracy within a database.**

**………………………………………………………………………………………….**

****

**……………………………………………………………………………………….**

**Q)What is Trigger ?**

In SQL, a trigger is a database object that is associated with a table and automatically executes a set of actions or statements in response to a specific event or data modification operation (such as INSERT, UPDATE, or DELETE) on that table. Triggers are used to enforce data integrity, implement business logic, or perform additional tasks before or after a data operation.

In short, a trigger in SQL is a mechanism that allows you to define automatic actions to be executed when certain events occur on a table, providing a way to enforce rules and automate processes in response to data changes

**Q) What is Normalization in SQL?**

Normalization in SQL is the process of organizing and structuring a relational database to minimize data redundancy and dependency issues. It involves breaking down larger tables into smaller, more manageable tables, and establishing relationships between them using primary keys and foreign keys. The goal of normalization is to eliminate data anomalies, improve data integrity, and optimize database performance.

Normalization is typically achieved through a series of normal forms, including:

1. First Normal Form (1NF): Ensures that each column contains atomic (indivisible) values, and there are no repeating groups.
2. Second Normal Form (2NF): Builds upon 1NF and requires that all non-key columns depend on the entire primary key, eliminating partial dependencies.
3. Third Normal Form (3NF): Builds upon 2NF and eliminates transitive dependencies, ensuring that non-key columns depend only on the primary key and not on other non-key columns.

There are additional normal forms beyond 3NF, such as Boyce-Codd Normal Form (BCNF) and Fourth Normal Form (4NF), which provide further guidelines for database design and normalization.

………………………………………………………………………………………………………………………………………………

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

In SQL, aggregate functions and scalar functions are used to perform calculations or operations on data within a query. Here's an explanation of each:

1. **Aggregate Functions:** Aggregate functions operate on a set of values and return a single value as the result. They are typically used with the GROUP BY clause to perform calculations on groups of rows. Some commonly used aggregate functions include:

* **COUNT**: Returns the number of rows or non-null values in a column.
* SUM: Calculates the sum of values in a column.
* AVG: Calculates the average of values in a column.
* MAX: Returns the maximum value in a column.
* MIN: Returns the minimum value in a column.

1. **Scalar Functions:** Scalar functions operate on a single value and return a modified or derived value. They can be used within SELECT statements, WHERE clauses, or in other SQL expressions. Scalar functions can perform various operations, such as manipulating strings, performing mathematical calculations, or extracting portions of a date.

Examples of scalar functions include:

* UPPER: Converts a string to uppercase.
* LOWER: Converts a string to lowercase.
* ROUND: Rounds a numeric value to a specified decimal place.
* CONCAT: Concatenates two or more strings together.
* DATEPART: Extracts a specific part (e.g., year, month, day) from a date
* …………………………………………………………………………………………………………………

**Q) What does a window function do in SQL?**

In SQL, a window function is a type of function that performs calculations across a set of rows, called a window, based on a specified window or partitioning criteria. Unlike aggregate functions that summarize data into a single result per group, window functions return a value for each row within the window, while maintaining the original result set.

**ROW\_NUMBER:** Assigns a unique sequential number to each row within a partition.

**RANK and DENSE\_RANK:** Rank the rows based on specified criteria within a partition.

**LAG and LEAD:** Access the values of the previous or next row within a partition.

**SUM, AVG, MAX, MIN:** Calculate aggregate values over a window

……………………………………………………………………………………………………………………………………….

Q) **Difference between rank, dense\_rank and row\_number in sql?**

In SQL, the functions RANK, DENSE\_RANK, and ROW\_NUMBER are used to assign a unique sequence number to each row in a result set. However, there are subtle differences in how these functions handle ties or duplicate values. Here's an explanation of each:

1. **ROW\_NUMBER:**
   * ROW\_NUMBER function assigns a unique sequential number to each row in the result set.
   * It does not handle ties or duplicate values. Each row receives a distinct number, regardless of any ties in the data.
   * The numbering starts from 1 for the first row and increments by 1 for each subsequent row.
   * ROW\_NUMBER does not guarantee any specific order unless an ORDER BY clause is used.
2. **RANK:**
   * RANK function assigns a unique rank to each distinct value in the result set.
   * It handles ties by assigning the same rank to rows with identical values. The next rank is then skipped for the tied rows.
   * The ranking is determined by the ORDER BY clause. Rows with the same values receive the same rank, and the next rank is skipped accordingly.
   * RANK values are not consecutive. If two rows have the same rank, the next rank is skipped, resulting in non-consecutive ranks.
3. **DENSE\_RANK:**
   * DENSE\_RANK function assigns a unique rank to each distinct value in the result set, similar to RANK.
   * It handles ties by assigning the same rank to rows with identical values, just like RANK.
   * However, DENSE\_RANK does not skip the next rank for tied rows. It assigns consecutive ranks to tied rows.
   * DENSE\_RANK values are consecutive, even if tied ranks are encountered. No ranks are skipped.

……………………………………………………………………………………………….

**Q) What are clustered and non-clustered index in SQL?.**

In SQL, indexes are used to improve the performance of database queries by allowing faster data retrieval. Two commonly used types of indexes are clustered indexes and non-clustered indexes. Here's an explanation of each:

**Clustered Index:**

A clustered index determines the physical order of data in a table.

There can be only one clustered index per table.

The data in the table is physically sorted and stored based on the values of the clustered index.

The leaf nodes of a clustered index contain the actual data rows.

Clustered indexes are particularly useful for optimizing queries that involve range-based searches or require sequential data access.

**Non-Clustered Index:**

A non-clustered index is a separate structure from the actual table data.

It contains a copy of selected columns from the table along with a pointer to the actual data row.

A table can have multiple non-clustered indexes.

Non-clustered indexes are generally smaller in size compared to the actual table data.

Non-clustered indexes are helpful for optimizing queries that involve searching, filtering, and sorting operations.

Here's an example to illustrate the difference between clustered and non-clustered indexes:

Consider a table "Employees" with columns "EmployeeID" (primary key), "FirstName", "LastName", and "Salary". If a clustered index is created on the "EmployeeID" column, the actual data in the table will be physically sorted and stored based on the values of "EmployeeID". In this case, accessing data based on "EmployeeID" would be faster because the data is organized sequentially.

On the other hand, a non-clustered index can be created on the "LastName" column. The non-clustered index will contain a copy of the "LastName" column values along with pointers to the actual data rows. This allows faster searching and filtering based on last names, as the index can be used to quickly locate the desired rows.

In summary, clustered indexes determine the physical order of data in a table, while non-clustered indexes provide a separate structure for faster data access based on selected columns. Both types of indexes play a crucial role in optimizing database performance for different types of queries.

………………………………………………………………………………………………………………………………………….