**Short Questions.**

1. Define SQL.

ANS::

SQL (Structured Query Language) is a language to operate databases; it includes Database Creation, Database Deletion, Fetching Data Rows, Modifying & Deleting Data rows, etc.

SQL stands for Structured Query Language which is a computer language for storing, manipulating and retrieving data stored in a relational database. SQL was developed in the 1970s by IBM Computer Scientists and became a standard of the American National Standards Institute (ANSI) in 1986, and the International Organization for Standardization (ISO) in 1987.

SQL is the standard language to communicate with Relational Database Systems. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their Standard Database Language

1. Sate the features of Good Relational Designs.

ANS::

**1. Normalization**

* **Definition**: Normalization involves organizing data into tables to minimize redundancy and dependency.
* **Objective**: Ensure that each table focuses on a single concept, and relationships between tables are well-defined.
* **Benefits**:
  + Reduces data duplication.
  + Simplifies data updates and ensures consistency.
  + Improves query performance.

**2. Minimal Redundancy**

* Redundant data increases storage requirements and the risk of inconsistencies.
* Good design ensures data is stored only once unless duplication is necessary for performance (denormalization in specific cases).

**3.Well-Defined Relationships**

* Properly designed relationships between tables (one-to-one, one-to-many, many-to-many) ensure logical connections.
* Relationships should avoid circular dependencies or unnecessary complexity.

**Scalability**

* A relational design should accommodate future growth, allowing for the addition of new tables, columns, or data without significant restructuring.

**Flexibility**

* The design should be flexible enough to handle changes in requirements without disrupting existing functionality.
* For example, changes in schema or additional attributes should be easy to integrate.

**Security**

* Data should be protected using access controls to ensure only authorized users can read or modify it.
* Sensitive information should be encrypted.

**Long Questions.**

1. Explain Aggregate Functions.

ANS::

**Aggregate Functions**

SQL Aggregate Functions are used to perform calculations on a set of rows and return a single value. They are often used with the GROUP BY clause in SQL to summarize data for each group. Commonly used aggregate functions include COUNT(), SUM(), AVG(), MIN(), and MAX().

**What Are SQL Aggregate Functions?**

SQL Aggregate Functions are essential for summarizing and analysing large datasets. These functions operate on multiple rows of a table, combining them into a single, more meaningful result. SQL Aggregate functions are mostly used with the GROUP BY clause of the SELECT statement.

**Various Aggregate Functions**

1. Count()
2. Sum()
3. Avg()
4. Min()
5. Max()

**Common SQL Aggregate Functions**

Below is the list of [SQL](https://www.geeksforgeeks.org/sql-tutorial/) aggregate functions, with examples

**Count()**

* ***Count(\*):*** Returns the total number of records .i.e 6.
* ***Count(salary):*** Return the number of Non-Null values over the column salary. i.e 5.
* ***Count(Distinct Salary):***  Return the number of distinct Non-Null values over the column salary .i.e 5.

**Sum():**

* ***sum(salary):***  Sum all Non-Null values of Column salary i.e., 3120.
* ***sum(Distinct salary):*** Sum of all distinct Non-Null values i.e., 3120..

**Avg():**

* ***Avg(salary)*** = Sum(salary) / count(salary) = 3120 / 5 = 624
* ***Avg(Distinct salary)*** = sum(Distinct salary) / Count(Distinct Salary) = 3120 / 5 = 624

**Min():**

* ***Min(salary):*** Minimum value in the salary column except NULL i.e., 403.

**Max():**

* ***Max(salary):*** Maximum value in the salary i.e., 802.

**SQL Aggregate Functions with Examples**

Let’s consider a demo Employee table for our examples. This table contains employee details such as their ID, Name, and Salary.

**Query:**

CREATE TABLE Employee (

Id INT PRIMARY KEY,

Name CHAR(1), -- Adjust data type and length if names can be longer than a single character

Salary DECIMAL(10,2) -- Adjust precision and scale if needed for salaries

);

INSERT INTO Employee (Id, Name, Salary)

VALUES (1, 'A', 802),

(2, 'B', 403),

(3, 'C', 604),

(4, 'D', 705),

(5, 'E', 606),

(6, 'F', NULL);

**Output:**

|  |  |  |
| --- | --- | --- |
| **Id** | **Name** | **Salary** |
| 1 | A | 802 |
| 2 | B | 403 |

1. Explain different SQL data types.

**SQL Data Types**

SQL data types define the **type of data** that can be stored in a column of a table. Choosing the right data type is important for **efficient data storage** and **accurate operations**.

SQL data types are mainly categorized into:

**1. Numeric Data Types:**

Used to store **numbers**, both integers and decimals.

| **Data Type** | **Description** |
| --- | --- |
| INT or INTEGER | Whole numbers (e.g., 1, 100, -5) |
| SMALLINT | Smaller range of integers |
| BIGINT | Very large integers |
| DECIMAL(p,s) or NUMERIC(p,s) | Fixed-point numbers with precision p and scale s |
| FLOAT, REAL, DOUBLE | Approximate (floating-point) numbers |

**2. Character/String Data Types:**

Used to store **text or strings**.

| **Data Type** | **Description** |
| --- | --- |
| CHAR(n) | Fixed-length string (e.g., CHAR(10) stores exactly 10 characters) |
| VARCHAR(n) | Variable-length string (up to n characters) |
| TEXT | Large string/text data |

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

Used to store **date and time** values.

| **Data Type** | **Description** |
| --- | --- |
| DATE | Stores only date (e.g., '2025-04-14') |
| TIME | Stores only time (e.g., '10:30:00') |
| DATETIME | Stores both date and time |
| TIMESTAMP | Stores date and time with automatic time zone updates |
| YEAR | Stores year in 2 or 4 digits |

**4. Boolean Data Type:**

Stores **logical values**.

| **Data Type** | **Description** |
| --- | --- |
| BOOLEAN | Stores TRUE or FALSE values (in some systems stored as 0 or 1) |

**5. Binary Data Types:**

Used to store **binary data** like images, files, etc.

| **Data Type** | **Description** |
| --- | --- |
| BINARY(n) | Fixed-length binary data |
| VARBINARY(n) | Variable-length binary data |
| BLOB | Large binary object (used to store files, images, etc.) |

1. Distinguish between Static and Dynamic Hashing.

**ANS::**

| **Point of Difference** | **Static Hashing** | **Dynamic Hashing** |
| --- | --- | --- |
| **1. Directory Size** | **Fixed size directory.** | **Directory grows or shrinks dynamically.** |
| **2. Flexibility** | **Less flexible; can lead to overflow.** | **More flexible; adjusts size as data grows.** |
| **3. Performance** | **Slower for large datasets due to overflow.** | **Faster for large data; avoids overflow.** |
| **4. Memory Usage** | **Uses less memory initially.** | **May use more memory due to dynamic resizing.** |
| **5. Rehashing** | **Difficult to rehash; needs manual resizing.** | **Automatic rehashing handled by the system.** |
| **6. Example Use Case** | **Suitable for small, fixed-size databases.** | **Suitable for large, growing databases.** |

1. Describe how an ER diagram represents entities, attributes, and relationships in a relational database model.

**ANS:::**

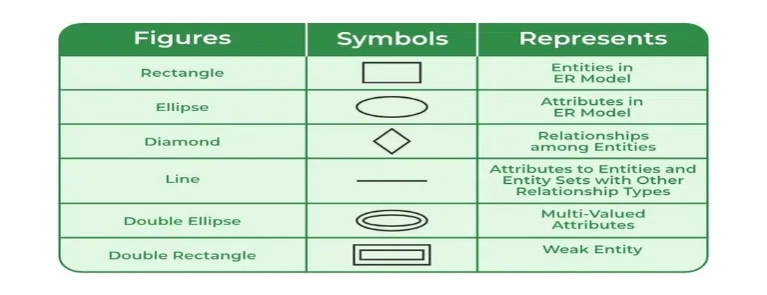
**Why Use ER Diagrams In DBMS?**

* ER diagrams represent the E-R model in a database, making 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 of the underlying DBMS used.
* It gives a standard solution for visualizing the data logically.

**Symbols Used in ER Model**

**ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:**

* Rectangles: Rectangles represent Entities in the ER Model.
* Ellipses: Ellipses represent Attributes in the ER Model.
* Diamond: Diamonds represent Relationships among Entities.
* Lines: Lines represent attributes to entities and entity sets with other relationship types.
* Double Ellipse: Double Ellipses represent Multi-Valued Attributes.
* Double Rectangle: Double Rectangle represents a Weak Entity.

****

