Sql stand for the structure query language. That is the used for the access and manipulate data in the database.

A **database** is an **electronically stored**, systematic collection of data that can include **words**, **numbers**, **images**, **videos**, and other types of files.

A **Database Management System (DBMS)** is a software solution designed to efficiently **manage**, **organize**, and **retrieve data** in a structured manner.

Data is the core component of any database, representing the **actual information stored.**

The schema is the blueprint or structure of the database. It defines how data is organized and includes details like **tables**, **columns**, **data types**, and relationships between entities.

A relational database's contents are arranged as a **collection of tables** with rows and columns.

Important command for the mysql

1 CREATE DATABASE *databasename*;

2 DROP DATABASE *databasename*;

2.1 use databasesname;

3 CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);

4 DROP TABLE *table_name*;

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

5 ALTER TABLE Customers
ADD Email varchar(255);

6 ALTER TABLE Customers
DROP COLUMN Email;

7 ALTER TABLE *table_name*
RENAME COLUMN *old_name* to *new_name*;

8 ALTER TABLE *table_name*
MODIFY COLUMN *column_name datatype*;

SQL constraints are used to specify rules for data in a table.

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

```
9 CREATE TABLE Persons (
   ID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int,
   CHECK (Age>=18)
);
```

Indexes are used to retrieve data from the database more quickly than otherwise

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

```
11 CREATE TABLE Persons (
   Personid int NOT NULL AUTO_INCREMENT,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int,
   PRIMARY KEY (Personid)
);
```

- DATE - format YYYY-MM-DD
- DATETIME - format: YYYY-MM-DD HH:MI:SS
- TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
- YEAR - format YYYY or YY

In SQL, a **view** is a **virtual table** based on the result of a SQL query. It does not store data physically like a real table but instead provides a way to look at data from one or more tables as if it were a single table.

- **Numeric Data Types**

Data Type	**Description**
INT or INTEGER	Whole numbers (e.g., 1, 100, -25)
SMALLINT	Smaller range of whole numbers
BIGINT	Large range of whole numbers
DECIMAL(p, s) / NUMERIC(p, s)	Fixed-point number. p = precision, s = scale (e.g., DECIMAL(5,2) stores 999.99)
FLOAT	Approximate floating-point number

Data Type	Description
REAL	Less precise floating-point number
DOUBLE or DOUBLE PRECISION	More precise floating-point number

---

🔹 **String/Text Data Types**

Data Type	Description
CHAR(n)	Fixed-length string (e.g., CHAR(10) always stores 10 characters)
VARCHAR(n)	Variable-length string, up to n characters
TEXT	Large amount of text (unlimited or very large in size)

🔹 **Date and Time Data Types**

Data Type	Description
DATE	Stores date only (e.g., '2025-05-03')
TIME	Stores time only (e.g., '13:45:00')
DATETIME	Stores date and time
TIMESTAMP	Date and time, often used to track record changes
YEAR	Stores a year value (e.g., 2025)

🔹 **Boolean Data Type**

Data Type	Description
BOOLEAN or BOOL	Stores TRUE or FALSE

🔹 **Binary Data Types**

Data Type	Description
BINARY(n)	Fixed-length binary data
VARBINARY(n)	Variable-length binary data
BLOB	Binary Large Object (e.g., images, files)

12 SELECT * FROM Customers;

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

14 SELECT DISTINCT *column1*, *column2, ...*
FROM *table_name*;

15 SELECT * FROM Customers
WHERE CustomerID=1;

a **clause** is a component of a SQL statement that performs a specific function. Clauses help structure queries and specify operations to be performed on data.

In MySQL, a **clause** is a component of a SQL statement that performs a specific function. Clauses help structure queries and specify operations to be performed on data.

Common Clauses in MySQL:

1. **SELECT clause**
   Specifies which columns of data to retrieve.
2. SELECT name, age FROM students;

3. **FROM clause**
   Indicates the table(s) to retrieve data from.
4. SELECT * FROM employees;
5. **WHERE clause**
   Filters rows based on conditions.
6. SELECT * FROM orders WHERE status = 'pending';
7. **GROUP BY clause**
   Groups rows sharing a property so aggregate functions can be applied.
8. SELECT department, COUNT(*) FROM employees GROUP BY department;
9. **HAVING clause**
   Filters groups created by GROUP BY.
10. SELECT department, COUNT(*) FROM employees GROUP BY department HAVING COUNT(*) > 5;
11. **ORDER BY clause**
    Sorts the result set.
12. SELECT name, salary FROM employees ORDER BY salary DESC;
13. **LIMIT clause**
    Limits the number of rows returned.
14. SELECT * FROM products LIMIT 10;

The `ORDER BY` keyword is used to sort the result-set in ascending or descending order.

** SELECT *column1*, *column2, ...*
FROM *table_name*
ORDER BY *column1, column2, ...* ASC|DESC;

** INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

** UPDATE *table_name*
SET *column1 = value1*, *column2 = value2*, ...
WHERE *condition*;

// get the top element

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
LIMIT *number*;

An aggregate function is a function that performs a calculation on a set of values, and returns a single value.

- MIN() - returns the smallest value within the selected column
- MAX() - returns the largest value within the selected column
- COUNT() - returns the number of rows in a set
- SUM() - returns the total sum of a numerical column
- AVG() - returns the average value of a numerical column

// The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;

Operator

Operator	Description
ALL	TRUE if all of the subquery values meet the condition
AND	TRUE if all the conditions separated by AND is TRUE
ANY	TRUE if any of the subquery values meet the condition
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if the subquery returns one or more records
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Displays a record if the condition(s) is NOT TRUE
OR	TRUE if any of the conditions separated by OR is TRUE
SOME	TRUE if any of the subquery values meet the condition