What is MySQL?

MySQL is a **Relational Database Management System (RDBMS)** that uses **SQL (Structured Query Language)** to store, manage, and retrieve data.

MySQL stores data in **tables** (rows and columns).

Example:

Students table

id name age course
1 John 20 Java
2 Alice 22 MySQL

- **Database** → Collection of tables
- **Table** → Collection of rows and columns
- **Row** → One record
- **Column** → One attribute/field
- **SQL** → Language used to communicate with MySQL

---

1. Creating a Database

Create Database

```
CREATE DATABASE college;
```

Select Database

```
USE college;
```

Delete Database

```
DROP DATABASE college;
```

---

2. Creating Tables

Create Student Table

```
CREATE TABLE students (
    id INT,
    name VARCHAR(50),
    age INT,
    course VARCHAR(50)
);
```

View tables:

```
SHOW TABLES;
```

View table structure:

```
DESC students;
```

---

3. CRUD Operations in MySQL

CRUD means:

Operation	SQL Command
Create	INSERT
Read	SELECT
Update	UPDATE
Delete	DELETE

CREATE (INSERT DATA)

Insert one record

```
INSERT INTO students
(id, name, age, course)
VALUES
(1, 'John', 20, 'Java');
```

Insert multiple records

```
INSERT INTO students
VALUES
```

```
(2,'Alice',22,'MySQL'),
(3,'Bob',21,'Python');
```

READ (SELECT DATA)

Select all data

```
SELECT * FROM students;
```

Output:

id name age course
1 John 20 Java
2 Alice 22 MySQL

Select specific columns

```
SELECT name, course
FROM students;
```

Using WHERE condition

```
SELECT *
FROM students
WHERE age > 20;
```

Output:

name age
Alice 22
Bob 21

Operators

Equal

```
SELECT *
FROM students
WHERE course='Java';
```

Greater than

```
WHERE age > 18;
```

Less than

```
WHERE age < 25;
```

Between

```
SELECT *
FROM students
WHERE age BETWEEN 20 AND 25;
```

IN

```
SELECT *
FROM students
WHERE course IN ('Java','Python');
```

LIKE

Search pattern:

```
SELECT *
FROM students
WHERE name LIKE 'J%';
```

Output:

```
John
```

`%` means any number of characters.

UPDATE DATA

Change existing records.

```
UPDATE students
SET age=23
WHERE id=2;
```

Before:

id name age
2 Alice 22

After:

id name age
2 Alice 23

DELETE DATA

Delete a record:

```
DELETE FROM students
WHERE id=3;
```

Delete all records:

```
DELETE FROM students;
```

---

4. Primary Key

A **Primary Key** is a column that uniquely identifies each record.

Rules:

- Cannot contain duplicate values
- Cannot contain NULL values
- Only one primary key per table

Example:

```
CREATE TABLE students (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT
);
```

Insert:

```
INSERT INTO students
VALUES
(1,'John',20);
```

Wrong:

```
INSERT INTO students
VALUES
(1,'Alice',22);
```

Error:

```
Duplicate primary key
```

Auto Increment Primary Key

Automatically generates IDs.

```
CREATE TABLE students (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50),
    age INT
);
```

Insert:

```
INSERT INTO students(name,age)
VALUES
('John',20);
```

MySQL creates:

```
id = 1
```

---

5. Foreign Key

A **Foreign Key** connects two tables.

Example:

Department Table

```
CREATE TABLE departments(
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(50)
);
```

Insert:

```
INSERT INTO departments
```

```
VALUES
(1,'Computer Science'),
(2,'Mechanical');
```

Student Table

```
CREATE TABLE students(
    id INT PRIMARY KEY,
    name VARCHAR(50),
    dept_id INT,

    FOREIGN KEY(dept_id)
    REFERENCES departments(dept_id)
);
```

Relationship:

```
departments
-------------
dept_id (Primary Key)
dept_name


students
-------------
id (Primary Key)
name
dept_id (Foreign Key)
```

Example data:

Departments:

dept_id	dept_name
1	Computer Science
2	Mechanical

Students:

id	name	dept_id
1	John	1
2	Alex	2

---

6. SQL Joins

Joins combine data from multiple tables.

We use:

Students

id	name	dept_id
1	John	1
2	Alice	2
3	Bob	1

Departments

dept_id	dept_name
1	Computer
2	Mechanical

INNER JOIN

Returns matching records from both tables.

```
SELECT
students.name,
departments.dept_name

FROM students

INNER JOIN departments
```

```
ON students.dept_id = departments.dept_id;
```

Output:

name	dept_name
John	Computer
Alice	Mechanical
Bob	Computer

LEFT JOIN

Returns all records from left table and matching records from right table.

```
SELECT
students.name,
departments.dept_name

FROM students

LEFT JOIN departments

ON students.dept_id = departments.dept_id;
```

Example:

Students without departments will also appear.

RIGHT JOIN

Returns all records from right table.

```
SELECT
students.name,
departments.dept_name

FROM students

RIGHT JOIN departments

ON students.dept_id = departments.dept_id;
```

FULL JOIN

MySQL does not directly support FULL JOIN.

Use:

```
SELECT *
FROM students

LEFT JOIN departments
ON students.dept_id=departments.dept_id

UNION

SELECT *
FROM students

RIGHT JOIN departments
ON students.dept_id=departments.dept_id;
```

---

7. Aggregate Functions

COUNT()

Count rows:

```
SELECT COUNT(*)
FROM students;
```

SUM()

```
SELECT SUM(salary)
FROM employees;
```

AVG()

```
SELECT AVG(age)
FROM students;
```

MAX()

```
SELECT MAX(age)
FROM students;
```

MIN()

```
SELECT MIN(age)
FROM students;
```

---

8. GROUP BY

Groups similar values.

Example:

```
SELECT course, COUNT(*)
FROM students
GROUP BY course;
```

Output:

course count
Java 5
Python 3

---

9. HAVING

Filters groups.

```
SELECT course, COUNT(*)
```

```
FROM students
GROUP BY course
HAVING COUNT(*) > 2;
```

---

10. ORDER BY

Sort data.

Ascending:

```
SELECT *
FROM students
ORDER BY age ASC;
```

Descending:

```
SELECT *
FROM students
ORDER BY age DESC;
```

---

11. LIMIT

Get limited records.

```
SELECT *
FROM students
LIMIT 5;
```

---

12. ALTER TABLE

Add column:

```
ALTER TABLE students
ADD email VARCHAR(100);
```

Modify column:

```
ALTER TABLE students
MODIFY name VARCHAR(100);
```

Delete column:

```
ALTER TABLE students
```

```
DROP COLUMN email;
```

---

13. Constraints in MySQL

NOT NULL

Value cannot be empty.

```
name VARCHAR(50) NOT NULL
```

UNIQUE

No duplicate values.

```
email VARCHAR(100) UNIQUE
```

DEFAULT

Default value.

```
city VARCHAR(50) DEFAULT 'Mumbai'
```

CHECK

Validate data.

```
age INT CHECK(age>=18)
```

---

14. Views

A view is a virtual table.

Create:

```
CREATE VIEW student_view AS
```

```
SELECT name, course
FROM students;
```

Use:

```
SELECT *
FROM student_view;
```

Delete:

```
DROP VIEW student_view;
```

---

15. Stored Procedure

Reusable SQL code.

Create:

```
DELIMITER //

CREATE PROCEDURE getStudents()

BEGIN

SELECT * FROM students;

END //

DELIMITER ;
```

Run:

```
CALL getStudents();
```

---

16. Index

Improves search speed.

Create:

```
CREATE INDEX idx_name
ON students(name);
```

Remove:

```
DROP INDEX idx_name
ON students;
```

Complete Database Example

```
CREATE DATABASE company;

USE company;


CREATE TABLE department(
id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(50)
);


CREATE TABLE employee(
id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(50),
salary INT,
department_id INT,

FOREIGN KEY(department_id)
REFERENCES department(id)
);


INSERT INTO department(name)
VALUES
('IT'),
('HR');


INSERT INTO employee(name,salary,department_id)
VALUES
('John',50000,1),
('Alice',40000,2);
```

```
SELECT
employee.name,
department.name

FROM employee

INNER JOIN department

ON employee.department_id = department.id;
```