* **Introduction to MySQL**

**What is MySQL?**

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to manage and manipulate data. It is widely used for web applications and is known for its reliability, speed, and ease of use.

**Features of MySQL:**

* Open-source and free to use under the GNU General Public License.
* Cross-platform: runs on Windows, Linux, MacOS, and more.
* High performance and scalability for large databases.
* Secure with user management and access privileges.
* Supports transactions and ACID compliance.
* Replication and clustering for high availability.

**RDBMS Concepts:**

* Data is organized in tables (rows and columns).
* Supports relationships between tables using keys.
* Enforces data integrity with constraints.
* Supports transactions for reliable data operations.

**Normalization:**

Normalization is a process in RDBMS to organize data efficiently by reducing redundancy and ensuring data integrity. It involves dividing tables into smaller, related tables and defining relationships. The main normal forms are:

* **1NF**: Each column contains atomic values; no repeating groups.
* **2NF**: In 1NF and all non-key columns fully depend on the entire primary key.
* **3NF**: In 2NF and all columns depend only on the primary key, removing transitive dependencies.
* **Database Operations**
* **CREATE DATABASE**: Creates a new database. Example: CREATE DATABASE school;
* **SHOW DATABASES**: Lists all databases on the server. Example: SHOW DATABASES;
* **USE database\_name**: Selects a database to work with. Example: USE school;
* **DROP DATABASE**: Deletes a database and all its data. Example: DROP DATABASE school;
* **Table Operations**
* **CREATE TABLE**: Defines a new table with columns and data types.

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

* **ALTER TABLE**: Modifies an existing table (add, drop, or modify columns).

Example: ALTER TABLE students ADD email VARCHAR(100);

* **DROP TABLE**: Removes a table and its data.

Example: DROP TABLE students;

* **RENAME TABLE**: Changes the name of a table.

Example: RENAME TABLE students TO pupils;

* **TRUNCATE TABLE**: Deletes all rows in a table but keeps its structure.

Example: TRUNCATE TABLE students;

* **Data Types**
* **Numeric**:
  + INT for integers, FLOAT and DOUBLE for floating-point numbers.
* **String**:
  + CHAR (fixed length), VARCHAR (variable length), TEXT (large text).
* **Date & Time**:
  + DATE (YYYY-MM-DD), DATETIME (YYYY-MM-DD HH:MM:SS), TIMESTAMP (auto-updated).
* **Others**:
  + JSON for JSON data, ENUM for predefined values, SET for multiple values.
* **Constraints**
* **PRIMARY KEY**: Uniquely identifies each row in a table.
* **FOREIGN KEY**: Links a column to the primary key of another table, enforcing referential integrity.
* **UNIQUE**: Ensures all values in a column are different.
* **NOT NULL**: Prevents NULL values in a column.
* **CHECK**: Restricts values based on a condition.
* **DEFAULT**: Sets a default value for a column if none is provided.
* **Data Manipulation (DML)**
* **INSERT**: Adds new rows to a table.

Example: INSERT INTO students (id, name, age) VALUES (1, 'Alice', 20);

* **SELECT**: Retrieves data from one or more tables.

Example: SELECT \* FROM students;

* **UPDATE**: Modifies existing data.

Example: UPDATE students SET age = 21 WHERE id = 1;

* **DELETE**: Removes rows from a table.

Example: DELETE FROM students WHERE id = 1;

* **SQL Clauses**
* **WHERE**: Filters rows based on a condition.

Example: SELECT \* FROM students WHERE age > 18;

* **ORDER BY**: Sorts results by one or more columns.

Example: SELECT \* FROM students ORDER BY name ASC;

* **GROUP BY**: Groups rows sharing a property for aggregate functions.

Example: SELECT age, COUNT(\*) FROM students GROUP BY age;

* **HAVING**: Filters groups created by GROUP BY.

Example: SELECT age, COUNT(\*) FROM students GROUP BY age HAVING COUNT(\*) > 1;

* **LIMIT**: Restricts the number of rows returned.

Example: SELECT \* FROM students LIMIT 5;

* **SQL Functions**
* **Aggregate**: COUNT(), SUM(), AVG(), MIN(), MAX()
* **String**: CONCAT(), LENGTH(), UPPER(), LOWER()
* **Date**: NOW(), CURDATE(), DATEDIFF(), DATE\_FORMAT()
* **Numeric**: ROUND(), FLOOR(), CEIL()
* **Joins**
* **INNER JOIN**: Returns rows with matching values in both tables.
* **LEFT JOIN**: Returns all rows from the left table and matched rows from the right.
* **RIGHT JOIN**: Returns all rows from the right table and matched rows from the left.
* **FULL OUTER JOIN**: Returns all rows when there is a match in either table (emulated with UNION in MySQL).
* **CROSS JOIN**: Returns the Cartesian product of both tables.
* **SELF JOIN**: Joins a table to itself.

**Example:**

SELECT students.name, courses.title FROM students INNER JOIN courses ON students.course\_id = courses.id;

* **Subqueries**
* **Scalar Subquery**: Returns a single value.
* **Correlated Subquery**: References columns from the outer query.
* **Nested Subquery**: Subquery inside another subquery.

**Example:** SELECT name FROM students WHERE age = (SELECT MAX(age) FROM students);

* **Views**
* **CREATE VIEW**:

CREATE VIEW student\_names AS SELECT name FROM students;

* **UPDATE VIEW**:

CREATE OR REPLACE VIEW student\_names AS SELECT name, age FROM students;

* **DROP VIEW**:

DROP VIEW student\_names;

* **Indexes**
* **CREATE INDEX**:

CREATE INDEX idx\_name ON students(name);

* **UNIQUE INDEX**:

CREATE UNIQUE INDEX idx\_email ON students(email);

* **FULLTEXT INDEX**:

CREATE FULLTEXT INDEX idx\_bio ON students(bio);

* **DROP INDEX**:

DROP INDEX idx\_name ON students;

* **aahhsh**