✅ MySQL – Essential Guide for Developers & DevOps

📌 What is MySQL?

MySQL is a **Relational Database Management System (RDBMS)** based on **SQL (Structured Query Language)**.
It's used to **store, manage, and retrieve data** in relational format (tables with rows and columns).

🧩 Why We Use MySQL?

- To persist data for your application (user details, orders, products, etc.)
- Easily query, update, delete, and join data
- Works well with **Java + Spring Boot** stack
- Open-source and widely supported

🧱 Basic MySQL Architecture

css

CopyEdit

[Client] ⇄ [MySQL Server] ⇄ [Databases]

- **Client**: Terminal, Workbench, or Java app
- **Server**: MySQL engine that processes queries
- **Databases**: Collections of tables, views, and stored procedures

---

📚 Core Concepts (Required)

1️⃣ Database

- Logical container of tables.
- Create with:

sql

CopyEdit

CREATE DATABASE studentdb;

2️⃣ Table

- Stores data in rows and columns.

sql

CopyEdit

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

3️⃣ CRUD Operations

- **C**reate: INSERT INTO table VALUES(...)
- **R**ead: SELECT * FROM table
- **U**pdate: UPDATE table SET col = val WHERE ...
- **D**elete: DELETE FROM table WHERE ...

4️⃣ Data Types

- INT, VARCHAR, DATE, BOOLEAN, FLOAT, TEXT, etc.

5️⃣ Constraints

- PRIMARY KEY, NOT NULL, UNIQUE, FOREIGN KEY, DEFAULT, CHECK

🔧 JOINs in SQL

Used to combine rows from two or more tables based on a related column.

Type	Description
INNER JOIN	Match in both tables
LEFT JOIN	All from left, match from right
RIGHT JOIN	All from right, match from left

Type	Description
FULL JOIN	All records when there's a match in one

🔐 User & Access Management

sql

CopyEdit

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON dbname.* TO 'username'@'localhost';

FLUSH PRIVILEGES;
```

🧪 Useful Queries for Developers

sql

CopyEdit

```
-- List all databases

SHOW DATABASES;


-- List all tables in current DB

SHOW TABLES;


-- Describe table

DESC table_name;


-- Drop table or database

DROP TABLE table_name;

DROP DATABASE db_name;


-- Search records
```

SELECT * FROM users WHERE name LIKE '%john%';

🔄 MySQL with Java (Spring Boot)

- Use JDBC/MySQL connector
- Add in application.properties:

properties

CopyEdit

```
spring.datasource.url=jdbc:mysql://localhost:3306/studentdb
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update
```

🛠 DevOps Tips with MySQL

Use Case	Tool
Local MySQL setup	apt, brew, or Docker
MySQL in Docker	docker run mysql
Backup/Restore	mysqldump
Monitoring	Prometheus MySQL Exporter
Production Deployment	AWS RDS, GCP SQL

📌 MySQL + Docker (Quick Setup)

bash

CopyEdit

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=root \
 -e MYSQL_DATABASE=studentdb -p 3306:3306 -d mysql:latest
```

🎯 Real-Life Examples

Scenario	MySQL Usage
Login System	Store users, passwords, roles
E-commerce	Products, orders, payments
Student Management System	Students, courses, marks
Inventory Tracking	Items, quantities, vendors

📖 Interview Questions (MySQL)

Q1. What is the difference between WHERE and HAVING?

- WHERE filters rows before grouping; HAVING filters after GROUP BY.

Q2. What is normalization?

- Process of organizing data to avoid redundancy (1NF, 2NF, 3NF).

Q3. What are transactions?

- A set of SQL operations executed as a unit (ACID properties).

Q4. What is indexing?

- Improves query speed by creating quick access paths.

Q5. How do you connect MySQL with Java (Spring Boot)?

- Via JDBC or Spring Data JPA using MySQL Driver and properties.

✅ Summary

Concept	Must Know
CREATE/DROP/SELECT/UPDATE	✅
JOINs and Subqueries	✅
Indexes, Keys, Constraints	✅
MySQL with Spring Boot	✅
Docker + MySQL	✅

Concept	Must Know
Backup & Permissions	👍

Here is a **MySQL + Docker Compose setup** — easy to run, perfect for local development and testing with Spring Boot or any backend app.

🐬 MySQL + Docker Compose Setup

✅ 1. Create docker-compose.yml

yaml

CopyEdit

```
version: '3.8'


services:
  mysql:
    image: mysql:8.0
    container_name: mysql-container
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: studentdb
      MYSQL_USER: devuser
      MYSQL_PASSWORD: devpass
    ports:
      - "3306:3306"
    volumes:
      - mysql-data:/var/lib/mysql
```

volumes:

mysql-data:

✅ 2. Run the MySQL Container

bash

CopyEdit

```
docker-compose up -d
```

- MySQL will be accessible on localhost:3306
- DB name: studentdb
- Username: devuser
- Password: devpass

✅ 3. Connect from Spring Boot (application.properties)

properties

CopyEdit

```
spring.datasource.url=jdbc:mysql://localhost:3306/studentdb
spring.datasource.username=devuser
spring.datasource.password=devpass
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

✅ 4. Verify MySQL is Running

bash

CopyEdit

```
docker ps
docker logs mysql-container
```

To access the DB:

bash

CopyEdit

```
docker exec -it mysql-container mysql -u root -p
```

📝 Optional Enhancements

✅ phpMyAdmin for UI

Add this to your docker-compose.yml:

yaml

CopyEdit

```yaml
 phpmyadmin:
   image: phpmyadmin/phpmyadmin
   container_name: phpmyadmin
   restart: always
   ports:
     - "8081:80"
   environment:
     PMA_HOST: mysql
```

Access phpMyAdmin at: http://localhost:8081

🧪 Summary of Config

Component	Access Info
MySQL DB	localhost:3306
Database Name	studentdb
User/Pass	devuser / devpass
Admin	root / root
UI (optional)	http://localhost:8081