1. App Installations

When starting development, some essential apps/tools need to be installed:

- Python / Java / Node.js → depending on your stack.
- Git → version control system.
- MySQL Server & Workbench → database.
- IDE / Code Editor → VS Code, PyCharm, IntelliJ, Eclipse, etc.
- Package Managers:
  - pip for Python
  - npm for Node.js
  - maven/gradle for Java
- Docker (Optional) → containerization for deployment.
- Postman → for API testing.

👉 Always ensure:

- Install the correct version (LTS preferred).
- Configure environment variables (PATH) properly.
- Verify installation with commands like:
  - git –version
  - python –version
  - mysql –version

---

2. Git Repository Creation and Commands

Git is a Version Control System (VCS) that helps track changes in code.

Steps to create a repository:

1. Initialize

   bash git init

   → creates a new Git repository in your folder.

2. Add remote repository (if using GitHub/GitLab)

   bash git remote add origin

3. Basic Git Workflow

   bash git status # check modified files git add file.py # add specific file git add . # add all files git commit -m "Initial commit" # save changes with message git push origin main # push to GitHub

4. Other useful commands

   - git clone → copy repository.
   - git pull origin main → fetch & merge changes.
   - git branch → list branches.
   - git checkout -b featureX → create & switch to new branch.
   - git merge featureX → merge branch into main.
   - git log → history of commits.

---

3. Introduction to MySQL

MySQL is a Relational Database Management System (RDBMS) used to store and manage data in structured format (tables).

- Tables = rows (records) + columns (fields).
- SQL (Structured Query Language) = used to interact with MySQL.

---

4. Keys in MySQL

- Primary Key: uniquely identifies a row (cannot be null).
- Foreign Key: establishes relationship between two tables.
- Unique Key: ensures all values in a column are unique.
- Composite Key: combination of multiple columns as primary key.
- Candidate Key: all possible keys that can be primary.
- Super Key: any set of attributes that uniquely identifies rows.

---

5. Data Types in MySQL

- String types:
  - CHAR(n) → fixed length (e.g. CHAR(5) stores "abc ").
  - VARCHAR(n) → variable length (saves space).
  - TEXT, BLOB → large text/binary data.
- Numeric types:

  - INT, BIGINT, SMALLINT
  - DECIMAL(m,n) → exact numbers (money).
  - FLOAT, DOUBLE → approximate decimals.

- Date/Time types:
  - DATE (YYYY-MM-DD)
  - DATETIME (YYYY-MM-DD HH:MM:SS)
  - TIMESTAMP (time-zone aware)
  - TIME (HH:MM:SS)
  - YEAR

---

6. MySQL Libraries (Functions)

- String Functions:
  - LENGTH('abc') → 3
  - UPPER('abc') → ABC
  - LOWER('ABC') → abc
  - CONCAT('Hello', 'World') → HelloWorld
  - SUBSTRING('Hello', 2, 3) → ell
- Date Functions:
  - NOW() → current date & time.
  - CURDATE() → current date.
  - DAYNAME('2025-09-22') → Monday
  - DATEDIFF('2025-09-22', '2025-09-20') → 2

---

7. Normalization Forms

Normalization = organizing data to reduce redundancy & improve integrity.

- 1NF: No repeating groups, atomic values only.
- 2NF: Must be in 1NF + no partial dependency on primary key.
- 3NF: Must be in 2NF + no transitive dependency.
- BCNF: Stronger version of 3NF.

---

8. ACID Properties

ACID = important for transactions in DB.

- Atomicity → all or nothing execution.
- Consistency → maintains valid state.
- Isolation → transactions execute independently.
- Durability → committed data is permanent.

---

9. Types of SQL Commands

- DDL (Data Definition Language): CREATE, ALTER, DROP, TRUNCATE
- DML (Data Manipulation Language): INSERT, UPDATE, DELETE
- TCL (Transaction Control Language): COMMIT, ROLLBACK, SAVEPOINT
- DRL (Data Retrieval Language) aka DQL: SELECT

---

10. Group By

Used to group rows with same values, often with aggregate functions.

sql SELECT department, COUNT(*) FROM employees GROUP BY department;

👉 Gives number of employees per department.

---

11. Joins

Joins combine rows from multiple tables.

- INNER JOIN → only matching rows.
- LEFT JOIN → all from left + matching from right.
- RIGHT JOIN → all from right + matching from left.
- FULL JOIN → all rows, match where possible.
- SELF JOIN → table joins with itself.
- CROSS JOIN → cartesian product.

Example:

sql SELECT e.name, d.department_name FROM employees e INNER JOIN departments d ON e.dept_id = d.id;
