Task-4 (next)

1. What is a Database?

A **database** is an organized collection of structured data that can be easily accessed, managed, and updated.
Examples: MySQL, PostgreSQL, SQL Server, Oracle.
Types:

Relational Database (RDBMS): Stores data in tables (MySQL, PostgreSQL).
NoSQL Database: Stores unstructured/semi-structured data (MongoDB, Cassandra).

What is SQL?

SQL (Structured Query Language) is the standard language to **create, read, update, and delete (CRUD)** data in a relational database.
DDL (Data Definition Language): CREATE, ALTER, DROP
DML (Data Manipulation Language): SELECT, INSERT, UPDATE, DELETE
DCL (Data Control Language): GRANT, REVOKE

TCL (Transaction Control Language): COMMIT, ROLLBACK, SAVEPOINT

3. Basic SQL Operations

a. Create Database and Tables

#NAME?

```
CREATE DATABASE CompanyDB;
```

```
#NAME?
USE CompanyDB;

#NAME?
CREATE TABLE Employees (
  EmployeeID INT AUTO_INCREMENT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  Department VARCHAR(50),
  Salary DECIMAL(10,2),
  HireDate DATE
);

#NAME?
CREATE TABLE Departments (
  DepartmentID INT AUTO_INCREMENT PRIMARY KEY,
  DepartmentName VARCHAR(50),
  Manager VARCHAR(50)
);
```

b. Insert Data

```
INSERT INTO Employees (FirstName, LastName, Department, Salary, HireDate)
VALUES
('Alice', 'Smith', 'HR', 50000, '2023-01-15'),
('Bob', 'Johnson', 'IT', 60000, '2022-07-10');

INSERT INTO Departments (DepartmentName, Manager)
VALUES ('HR', 'John Doe'), ('IT', 'Jane Doe');
```

c. Query Data (SELECT)

```
#NAME?
SELECT * FROM Employees;

-- Select specific columns
SELECT FirstName, LastName, Salary FROM Employees;

-- Filter employees by department
SELECT * FROM Employees WHERE Department='IT';

-- Sort employees by salary
SELECT * FROM Employees ORDER BY Salary DESC;
```

```
-- Aggregate data (average salary per department)
SELECT Department, AVG(Salary) AS AvgSalary
FROM Employees
GROUP BY Department;
```

d. JOINS (Combine Tables)

```
#NAME?
SELECT e.FirstName, e.LastName, e.Department, d.Manager
FROM Employees e
INNER JOIN Departments d
ON e.Department = d.DepartmentName;

-- Left join (all employees, even if department info is missing)
SELECT e.FirstName, e.LastName, d.Manager
FROM Employees e
LEFT JOIN Departments d
ON e.Department = d.DepartmentName;
```

e. Subqueries

```
-- Employees earning more than the average salary
SELECT * FROM Employees
WHERE Salary > (SELECT AVG(Salary) FROM Employees);
```

f. Views

```
#NAME?
CREATE VIEW IT_Employees AS
SELECT FirstName, LastName, Salary
FROM Employees
WHERE Department='IT';

#NAME?
SELECT * FROM IT_Employees;
```

g. Indexes (Performance)

```
-- Create index on LastName to speed up searches
CREATE INDEX idx_lastname ON Employees(LastName);
```

h. Aggregate Functions

SUM()
– Total salary
AVG()
– Average salary
COUNT()
– Number of employees
MAX()
/
MIN()
– Highest/lowest salary

Example:

```
SELECT Department, COUNT(*) AS EmployeeCount, AVG(Salary) AS AvgSalary
FROM Employees
GROUP BY Department;
```

4. Summary

Database: Organized collection of data.

SQL: Language to manage databases.

Tables: Store data in rows and columns.

Queries: Retrieve and manipulate data.
Joins/Subqueries/Views: Combine, filter, or save queries.
Indexes: Improve performance for searching and sorting.

Database analysis

Database Basics

A **database** is a structured collection of data. It allows multiple users to **store, retrieve, and manage data efficiently**.
Example: A company might have a database to store employee, department, and payroll data.

Analysis: Without a database, data would be stored in files, making it harder to query, maintain, or secure. Databases provide:

Data Integrity: Ensures correctness of data.
Security: Users can be restricted to specific operations.
Efficiency: Optimized for storage, retrieval, and updates.

2. SQL (Structured Query Language)

SQL is the language used to interact with relational databases.
Subcategories of SQL:

Category	Commands
DDL (Data Definition)	CREATE, ALTER, DROP
DML (Data Manipulation)	SELECT, INSERT, UPDATE, DELETE
DCL (Data Control)	GRANT, REVOKE
TCL (Transaction Control)	COMMIT, ROLLBACK, SAVEPOINT

Analysis:

SQL ensures **standardization** in database operations, making it easier to maintain and scale applications.

3. Tables and Structure

Tables store data in rows and columns. Each **row** is a record, each **column** is a field/attribute.
Example: Employees Table

EmployeeID	FirstName
1	Alice

Analysis:

Tables organize data logically.
Normalization ensures minimal redundancy and maintains data integrity.

4. Insert and Query Data

Insert: Adds data to tables.

```
INSERT INTO Employees
(FirstName, LastName,
Department, Salary, HireDate)
VALUES
('Alice','Smith','HR',50000,'20
23-01-15');
```

Query: Retrieves data.

```
SELECT * FROM Employees;
```

Analysis:

Queries allow **data filtering, sorting, and analysis**.

Example: A manager can get **all IT employees with salary > 60000** using a WHERE clause.

5. Filtering and Sorting (WHERE, ORDER BY)

WHERE filters rows based on conditions.
ORDER BY sorts results ascending/descending.

```
SELECT * FROM Employees WHERE
Department='IT' ORDER BY
Salary DESC;
```

Analysis:

Helps **focus on relevant data**.
Critical for dashboards, reporting, and decision-making.

6. Aggregates and GROUP BY

SUM(), AVG(), COUNT(), MAX(), MIN() allow **summarization** of data.
GROUP BY aggregates rows by a column (e.g., department).

```
SELECT Department, AVG(Salary)
AS AvgSalary
FROM Employees
GROUP BY Department;
```

Analysis:
Enables **quick insights**: average salary per department, employee count, total expenditure.
Useful for **HR, finance, and analytics**.

7. Joins (INNER, LEFT, RIGHT)

Combine data from multiple tables.
INNER JOIN: Only matching rows.

```
SELECT e.FirstName, d.Manager
FROM Employees e
INNER JOIN Departments d
ON e.Department =
d.DepartmentName;
```

LEFT JOIN: All employees, even if no department info.
RIGHT JOIN: All departments, even if no employees.
Analysis:
Critical for **relational data**.
Example: Match employees with their managers for reporting.

8. Subqueries

A query inside another query.

```
SELECT * FROM Employees
WHERE Salary > (SELECT
AVG(Salary) FROM Employees);
```

Analysis:
Allows **dynamic filtering** based on other data.

Useful for **comparative analytics**, e.g., employees earning above average.

9. Views

Virtual tables from queries.

```
CREATE VIEW IT_Employees AS
SELECT FirstName, LastName,
Salary
FROM Employees
WHERE Department='IT';
```

Analysis:

Simplifies **repeated queries**.

Improves **security**: restrict access to only certain columns.

10. Indexes

Speeds up data retrieval.

```
CREATE INDEX idx_lastname ON
Employees(LastName);
```

Analysis:

Critical for **large datasets**.

Improves performance of searches and sorts, but may slow **inserts/updates** slightly.

11. Practical SQL Analysis Example

Scenario: A company wants:

1. Employees earning above the department average.
2. Total salary expenditure per department.

```
SELECT e.FirstName,
e.LastName, e.Salary,
e.Department
FROM Employees e
WHERE e.Salary > (
    SELECT AVG(Salary)
    FROM Employees
    WHERE Department =
e.Department
);


SELECT Department, SUM(Salary)
AS TotalSalary
FROM Employees
GROUP BY Department;
```

Analysis:

Managers can **identify high earners**.

Finance can **budget salaries** by department.

✅ Conclusion

Database + SQL = Powerful tool for structured data management.
SQL features (queries, joins, subqueries, views, aggregates) allow **data-driven decisions**.
Indexes and proper table design ensure **efficient performance**.

If you want, I can **create a complete MySQL database with Employees/Departments and generate an Excel sheet with all query results** so you can **see the full analysis live**.
Do you want me to do that next?

Purpose

Define or modify database structure
Manipulate data within tables
Control access and permissions

Manage transactions for consistency

LastName	Department	Salary	HireDate
Smith	HR	50000	15/01/2023