**Introduction to SQL and Database Concepts**

**1. File Storage Overview**

**Definition:**  
File storage is a hierarchical method of storing data in files, organized within folders and directories.

**Advantages:**

* Easy to implement for small-scale use
* Cost-effective (no specialized software required)
* Direct access to files
* Suitable for static data like images, logs, and configuration files

**Common Use Cases:**

* Media storage (photos, videos, audio)
* Application log files
* Configuration settings
* Simple data records (CSV, TXT)

**2. Database Management System (DBMS)**

**Definition:**  
A DBMS is software that manages structured data, enabling efficient storage, retrieval, and manipulation.

**Benefits Over File Storage:**

* Eliminates data redundancy
* Enhances consistency and integrity
* Supports concurrent access
* Provides security and controlled access
* Enables efficient querying
* Offers data independence

**SQL Essentials**

**Table Creation**

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(40) NOT NULL,

designation VARCHAR(60) NOT NULL,

email VARCHAR(60) NOT NULL

);

CREATE TABLE e\_commerce\_orders (

order\_id INT PRIMARY KEY,

customer\_id INT NOT NULL,

city VARCHAR(40) NOT NULL,

status VARCHAR(50) NOT NULL,

order\_date DATE NOT NULL,

amount DECIMAL NOT NULL

);

**Data Insertion**

INSERT INTO employee VALUES (101, "Sayali", "STE", "powarsayali2002@gmail.com");

INSERT INTO customer VALUES

(101, "Raj", "raj@mail.com"),

(102, "Meera", "meera@mail.com"),

(103, "Arjun", "arjun@mail.com");

**Basic Queries**

* SELECT \* FROM e\_commerce\_orders;
* SELECT order\_id AS ID, customer\_id AS Customer FROM e\_commerce\_orders;
* SELECT \* FROM e\_commerce\_orders WHERE order\_id = 3;
* SELECT \* FROM e\_commerce\_orders ORDER BY order\_date DESC;

**Aggregate Functions**

SELECT COUNT(\*) AS total\_orders,

SUM(amount) AS total\_revenue,

AVG(amount) AS avg\_order\_value,

MAX(amount) AS max\_order,

MIN(amount) AS min\_order

FROM e\_commerce\_orders

WHERE status = "PAID";

**Grouping by City**

SELECT city, SUM(amount) AS revenue

FROM e\_commerce\_orders

WHERE status = "PAID"

GROUP BY city

HAVING SUM(amount) > 2500;

**Top Orders and Tax Calculation**

SELECT order\_id, city, amount

FROM e\_commerce\_orders

ORDER BY amount DESC

LIMIT 3;

SELECT order\_id, amount, amount \* 0.18 AS gst\_tax

FROM e\_commerce\_orders;

**Join Operations**

* Inner Join
* Left Join
* Right Join
* Cross Join

Example:

SELECT o.order\_id, c.name, o.amount

FROM e\_commerce\_orders o

JOIN customer c ON o.customer\_id = c.customer\_id

WHERE o.status = "PAID";

**Filtering and Pattern Matching**

* WHERE city IN ("Mumbai", "Delhi")
* WHERE city LIKE "M%"
* WHERE amount BETWEEN 1000 AND 3000

**Subqueries**

SELECT name, salary

FROM employee

WHERE salary > (SELECT AVG(salary) FROM employee);

**Views**

CREATE VIEW high\_salary\_employees AS

SELECT name, salary

FROM employee

WHERE salary > 40000;

**Indexing**

CREATE INDEX idx\_name ON employee(name);