



give core concepts of SQL

Got it! Here are the **core concepts of SQL (Structured Query Language)** that you should know:

1. Databases and Tables

- **Database:** A structured collection of data.
 - **Table:** Stores data in **rows (records)** and **columns (fields)**, like a spreadsheet.
 - Each column has a **data type** (e.g., INT, VARCHAR, DATE).
-

2. SQL Statements

SQL is divided into categories:

- **DDL (Data Definition Language)** → Defines structure
 - CREATE, ALTER, DROP, TRUNCATE
 - **DML (Data Manipulation Language)** → Manages data
 - SELECT, INSERT, UPDATE, DELETE
 - **DCL (Data Control Language)** → Controls permissions
 - GRANT, REVOKE
 - **TCL (Transaction Control Language)** → Manages transactions
 - COMMIT, ROLLBACK, SAVEPOINT
-

3. CRUD Operations

The four essential actions for working with data:

- **Create** → INSERT INTO
 - **Read** → SELECT
 - **Update** → UPDATE
 - **Delete** → DELETE
-

4. Filtering Data

- **WHERE** → Apply conditions (**WHERE age > 18**)
 - **Operators:** =, !=, >, <, BETWEEN, IN, LIKE, IS NULL
-

5. Sorting & Limiting

- **ORDER BY** → Sort results (**ORDER BY salary DESC**)
 - **LIMIT / TOP** → Restrict number of rows
-

6. Joins

Combine data from multiple tables:

- **INNER JOIN** → Only matching rows
 - **LEFT JOIN** → All rows from left table + matches
 - **RIGHT JOIN** → All rows from right table + matches
 - **FULL OUTER JOIN** → All rows from both tables
 - **CROSS JOIN** → Cartesian product
-

7. Aggregation & Grouping

- **Aggregate Functions:** **COUNT()**, **SUM()**, **AVG()**, **MIN()**, **MAX()**
 - **GROUP BY** → Groups rows by column
 - **HAVING** → Filters groups (like **WHERE** but after grouping)
-

8. Constraints

Rules to maintain data integrity:

- **PRIMARY KEY** → Unique + not null
 - **FOREIGN KEY** → References another table
 - **UNIQUE** → Ensures uniqueness
 - **NOT NULL** → Must have a value
 - **CHECK** → Custom condition
 - **DEFAULT** → Auto value if none given
-

9. Indexes

- Improve query speed by indexing columns
 - **CREATE INDEX idx_name ON employees(salary);**
-

10. Transactions

- Group multiple operations into one unit:
 - **BEGIN TRANSACTION**
 - **COMMIT** → Save changes

- ROLLBACK → Undo changes

👉 Would you like me to also create a **visual cheat sheet / diagram** of these concepts for quick revision?



create a product management system of [dell laptops], where we use all the above things

Perfect! 🚀 Let's design a **Product Management System for Dell Laptops** using **all the SQL concepts** I listed above.

1. Database & Tables (DDL)

```
sql

-- Create Database
CREATE DATABASE DellProductDB;
USE DellProductDB;

-- Create Table: Categories
CREATE TABLE Categories (
  category_id INT PRIMARY KEY AUTO_INCREMENT,
  category_name VARCHAR(50) UNIQUE NOT NULL
);

-- Create Table: Laptops
CREATE TABLE Laptops (
  laptop_id INT PRIMARY KEY AUTO_INCREMENT,
  model_name VARCHAR(100) NOT NULL,
  processor VARCHAR(50),
  ram_size INT CHECK (ram_size > 0),
  storage_size INT,
  price DECIMAL(10,2) CHECK (price > 0),
  category_id INT,
  release_date DATE DEFAULT (CURRENT_DATE),
  FOREIGN KEY (category_id) REFERENCES Categories(category_id)
);

-- Create Table: Customers
CREATE TABLE Customers (
  customer_id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  phone VARCHAR(15)
);

-- Create Table: Orders
CREATE TABLE Orders (
  order_id INT PRIMARY KEY AUTO_INCREMENT,
  customer_id INT,
  laptop_id INT,
  order_date DATE DEFAULT (CURRENT_DATE),
  quantity INT CHECK (quantity > 0),
  FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
  FOREIGN KEY (laptop_id) REFERENCES Laptops(laptop_id)
);
```

2. Insert Sample Data (DML – CREATE)

```
sql
```

```
-- Insert Categories
INSERT INTO Categories (category_name) VALUES
('Gaming'), ('Business'), ('Ultrabook');

-- Insert Laptops
INSERT INTO Laptops (model_name, processor, ram_size, storage_size, price, category_id)
VALUES
('Dell XPS 15', 'Intel i7', 16, 512, 1500.00, 3),
('Dell Alienware M15', 'Intel i9', 32, 1024, 2500.00, 1),
('Dell Latitude 7420', 'Intel i5', 8, 256, 1200.00, 2);

-- Insert Customers
INSERT INTO Customers (name, email, phone)
VALUES
('Alice Johnson', 'alice@example.com', '1234567890'),
('Bob Smith', 'bob@example.com', '9876543210');

-- Insert Orders
INSERT INTO Orders (customer_id, laptop_id, quantity)
VALUES
(1, 1, 2),
(2, 2, 1);
```

3. Read Data (DML – SELECT)

```
sql
```

```
-- Get all laptops with price > 1000
SELECT model_name, price FROM Laptops WHERE price > 1000;

-- Get orders with customer and laptop details (JOIN)
SELECT o.order_id, c.name, l.model_name, o.quantity, l.price, (o.quantity * l.price) AS
total amount
FROM Orders o
INNER JOIN Customers c ON o.customer_id = c.customer_id
INNER JOIN Laptops l ON o.laptop_id = l.laptop_id;
```

4. Update & Delete (DML)

```
sql
```

```
-- Update price of Dell Latitude
UPDATE Laptops SET price = 1100 WHERE model_name = 'Dell Latitude 7420';

-- Delete a customer
DELETE FROM Customers WHERE name = 'Bob Smith';
```

5. Sorting & Limiting

```
sql
```

```
-- Get top 2 most expensive laptops
SELECT model_name, price FROM Laptops ORDER BY price DESC LIMIT 2;
```

6. Grouping & Aggregation

```
sql

-- Count laptops by category
SELECT c.category_name, COUNT(*) AS laptop_count
FROM Laptops l
INNER JOIN Categories c ON l.category_id = c.category_id
GROUP BY c.category_name
HAVING COUNT(*) > 0;
```

7. Indexes

```
sql

-- Create index on laptop price for faster search
CREATE INDEX idx_price ON Laptops(price);
```

8. Transactions (TCL)

```
sql

-- Example transaction
START TRANSACTION;

INSERT INTO Orders (customer_id, laptop_id, quantity) VALUES (1, 3, 1);

-- Oops, rollback
ROLLBACK;

-- Try again and commit
INSERT INTO Orders (customer_id, laptop_id, quantity) VALUES (1, 3, 1);
COMMIT;
```

9. Permissions (DCL)

```
sql

-- Grant read-only access to an analyst user
GRANT SELECT ON DellProductDB.* TO 'analyst'@'localhost';

-- Revoke permission
REVOKE SELECT ON DellProductDB.* FROM 'analyst'@'localhost';
```

✅ Now you have a **fully functional Dell Laptop Product Management System** that uses **DDL, DML, DCL, TCL, CRUD, Joins, Aggregation, Constraints, Indexes, Transactions** – everything tied together.

Do you want me to also **draw an ER Diagram (Entity-Relationship)** for this system so you can see how the tables connect visually?