**1️⃣ Setup: CollegeDB Tables and Sample Data**

CREATE DATABASE CollegeDB;

USE CollegeDB;

CREATE TABLE Department (

DeptID INT PRIMARY KEY AUTO\_INCREMENT,

DeptName VARCHAR(50) NOT NULL UNIQUE,

Location VARCHAR(50)

);

CREATE TABLE Student (

StudentID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(50) NOT NULL,

Gender CHAR(1) CHECK (Gender IN ('M','F')),

DOB DATE NOT NULL,

DeptID INT,

Email VARCHAR(100) UNIQUE,

Phone VARCHAR(15),

Marks INT,

FOREIGN KEY (DeptID) REFERENCES Department(DeptID)

);

-- Insert Departments

INSERT INTO Department (DeptName, Location)

VALUES ('Computer Science','Block A'),

('Mechanical','Block B'),

('Electronics','Block C');

-- Insert Students

INSERT INTO Student (Name, Gender, DOB, DeptID, Email, Phone, Marks)

VALUES ('Mayuri Kasetty','F','2000-05-10',1,'mayuri@example.com','9876543210', 85),

('Ravi Kumar','M','1999-08-15',2,'ravi@example.com','9123456780', 72),

('Anil Sharma','M','2001-02-20',3,'anil@example.com','9001122334', 65),

('Sita Reddy','F','2002-03-12',1,'sita@example.com','9012345678', 90),

('Deepak Verma','M','2000-11-01',2,'deepak@example.com','9011223344', 58),

('Priya Singh','F','2001-06-22',3,'priya@example.com','9009988776', 77);

✅ We now have two tables: **Department** and **Student** with sample data, including Marks to demonstrate filtering.

**2️⃣ Basic Filtering with WHERE**

-- Simple filtering: students with marks > 70

SELECT \* FROM Student

WHERE Marks > 70;

**3️⃣ Logical Operators: AND, OR, NOT**

-- AND: Female students in Computer Science with marks > 80

SELECT \* FROM Student

WHERE Gender='F' AND DeptID=1 AND Marks > 80;

-- OR: Students in Mechanical OR Electronics departments

SELECT \* FROM Student

WHERE DeptID=2 OR DeptID=3;

-- NOT: Students NOT in Computer Science

SELECT \* FROM Student

WHERE NOT DeptID=1;

**4️⃣ IN / NOT IN**

-- Students in departments 1 or 3

SELECT \* FROM Student

WHERE DeptID IN (1,3);

-- Students NOT in department 2

SELECT \* FROM Student

WHERE DeptID NOT IN (2);

**5️⃣ ANY / ALL**

-- Marks greater than ANY student in department 2

SELECT \* FROM Student

WHERE Marks > ANY (SELECT Marks FROM Student WHERE DeptID=2);

-- Marks greater than ALL students in department 2

SELECT \* FROM Student

WHERE Marks > ALL (SELECT Marks FROM Student WHERE DeptID=2);

**6️⃣ LIKE (Pattern Matching)**

-- Names starting with 'A'

SELECT \* FROM Student

WHERE Name LIKE 'A%';

-- Names ending with 'i'

SELECT \* FROM Student

WHERE Name LIKE '%i';

-- Names containing 'ree'

SELECT \* FROM Student

WHERE Name LIKE '%ree%';

**7️⃣ BETWEEN**

-- Students with marks between 70 and 90

SELECT \* FROM Student

WHERE Marks BETWEEN 70 AND 90;

-- DOB between 2000-01-01 and 2001-12-31

SELECT \* FROM Student

WHERE DOB BETWEEN '2000-01-01' AND '2001-12-31';

**8️⃣ EXISTS / NOT EXISTS**

-- Students in a department that exists in Department table

SELECT \* FROM Student s

WHERE EXISTS (

SELECT 1 FROM Department d

WHERE s.DeptID = d.DeptID

);

-- Students NOT in any department (hypothetical case)

SELECT \* FROM Student s

WHERE NOT EXISTS (

SELECT 1 FROM Department d

WHERE s.DeptID = d.DeptID

);

**9️⃣ Combined Example**

-- Female students in Computer Science with marks > 80 OR students in Electronics with marks < 70

SELECT \* FROM Student

WHERE (Gender='F' AND DeptID=1 AND Marks > 80)

OR (DeptID=3 AND Marks < 70)

ORDER BY Marks DESC;

✅ **Explanation:**

* You can combine **AND, OR, NOT** with **IN, BETWEEN, LIKE, EXISTS**
* Useful for **complex filtering** in reports

If you want, I can **draw a big visual example with all these filters applied to CollegeDB** showing which students match each query—it makes it super easy to understand.

Do you want me to do that?

**You said:**

definition table for all

**ChatGPT said:**

Perfect! Let’s summarize **all the filtering keywords and operators** in SQL in one **comprehensive definition table** with **syntax and purpose**. I’ll make it detailed for clarity.

| **Keyword / Operator** | **Definition / Purpose** | **Syntax Example** |
| --- | --- | --- |
| **WHERE** | Filters rows based on a condition | SELECT \* FROM Student WHERE Marks > 70; |
| **AND** | Combines multiple conditions; all must be true | WHERE Gender='F' AND Marks>80; |
| **OR** | Combines multiple conditions; any one can be true | WHERE DeptID=2 OR DeptID=3; |
| **NOT** | Negates a condition | WHERE NOT DeptID=1; |
| **IN** | Matches a value in a list | WHERE DeptID IN (1,3); |
| **NOT IN** | Matches a value not in a list | WHERE DeptID NOT IN (2); |
| **ANY** | Compares a value to **any** value in a subquery | WHERE Marks > ANY (SELECT Marks FROM Student WHERE DeptID=2); |
| **ALL** | Compares a value to **all** values in a subquery | WHERE Marks > ALL (SELECT Marks FROM Student WHERE DeptID=2); |
| **LIKE** | Pattern matching for strings | WHERE Name LIKE 'A%'; |
| **BETWEEN** | Checks if a value is in a range (inclusive) | WHERE Marks BETWEEN 70 AND 90; |
| **EXISTS** | Returns TRUE if subquery returns any row | WHERE EXISTS (SELECT 1 FROM Department d WHERE s.DeptID=d.DeptID); |
| **NOT EXISTS** | Returns TRUE if subquery returns no rows | WHERE NOT EXISTS (SELECT 1 FROM Department d WHERE s.DeptID=d.DeptID); |
| **ORDER BY** | Sorts result set ascending (ASC) or descending (DESC) | SELECT \* FROM Student ORDER BY Marks DESC; |
| **IS NULL** | Checks for NULL values | WHERE Phone IS NULL; |
| **IS NOT NULL** | Checks for non-NULL values | WHERE Email IS NOT NULL; |

**✅ Quick Notes:**

* **AND / OR / NOT** are **logical operators**.
* **IN / NOT IN / ANY / ALL / EXISTS / NOT EXISTS** are **set comparison operators**.
* **LIKE / BETWEEN** are **pattern/range operators**.
* Always use **parentheses** () when combining multiple conditions for clarity.