Database Fundamentals

Cybersecurity Department

Course Code: ***CBS 213***

Practical Lecture 5: SQL Select Queries & Data Filtering

Halal Abdulrahman Ahmed

Outlines

- Recap: databases, tables, keys, relationships
- Introduction to SELECT
- Selecting columns vs SELECT *
- Using DISTINCT
- Filtering with WHERE
- Comparison & logical operators
- Sorting with ORDER BY
- Grouping with GROUP BY
- Hands-on SQL practice

Learning Outcomes

By the end of this lecture, students will be able to:

- Use the SELECT statement to retrieve specific data from MySQL tables.
- Apply DISTINCT, WHERE, and comparison operators to filter query results.
- Combine multiple conditions using logical operators.
- Sort and organize output using ORDER BY.
- Group rows and summarize data using GROUP BY.

The MySQL SELECT Statement

- The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

SELECT Syntax

```
SELECT column1, column2
FROM table_name;
```

Here, column1, column2 are the field names of the table you want to select data from.

```
SELECT * FROM table_name;
```

Select all the fields available in the table.

```
SELECT DISTINCT column1, column2
FROM table_name;
```

The SELECT DISTINCT statement is used to return only distinct (different) values.
Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

```
SELECT Country FROM Customers;
```

Selects all (including the duplicates) values from the "Country" column in the "Customers" table.

SELECT Columns Example

```sql
-- Create database
CREATE DATABASE UniversityDB;

-- Use database
USE UniversityDB;

-- Create table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(100),
    City VARCHAR(50),
    Country VARCHAR(50),
    Phone VARCHAR(20)
);
```

This command creates a new database named UniversityDB.

This tells MySQL to switch to UniversityDB so all next commands run inside it.

This creates a table named Customers with 5 attributes (columns).

```sql
-- Insert sample data
INSERT INTO Customers (CustomerID, CustomerName, City, Country, Phone)
VALUES
(1, 'Alice Johnson', 'Paris', 'France', '123-4567'),
(2, 'Mohammed Ali', 'Erbil', 'Iraq', '444-8899'),
(3, 'Sara Smith', 'London', 'UK', '567-2233');

-- Select specific columns
SELECT CustomerName, City, Country
FROM Customers;
```

These rows are sample records inserted into the Customers table.

This retrieves only CustomerName, City, and Country from the table.

SELECT * Example

```sql
-- Select all columns
SELECT *
FROM Customers;
```

This retrieves all columns and all rows from the Customers table, including:

CustomerID
CustomerName
City
Country
Phone

SELECT * Example

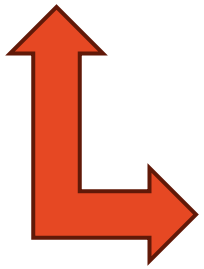
```sql
-- Select all columns
SELECT *
FROM Customers;
```

This retrieves all columns and all rows from the Customers table, including:

CustomerID
CustomerName
City
Country
Phone

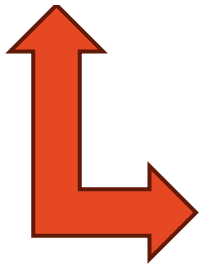
SELECT DISTINCT Examples

```
SELECT DISTINCT Country FROM Customers;
```



Selects only the DISTINCT values from the "country" column in the "customers" table

```
SELECT COUNT(DISTINCT Country) FROM Customers;
```



SQL statement counts and returns the number of different (distinct) countries in the "Customers" table

Filter Query in MySQL

Introduction to Filtering in MySQL

- **MySQL is one of the most popular relational database management systems (RDBMS). It is** known for its **high performance**, **reliability**, and **ease of use**.

- **Filtering data is one of the most important skills in SQL.** Filtering allows you to extract only the **specific rows** you need from a large dataset. It enables more **focused analysis**, improves **accuracy**, and supports **better decision-making**.
- Essential for real-world tasks such as searching students by name, finding employees by department, or selecting transactions within a date range.

How to Filter Query Results in MySQL

Filtering involves selecting subsets of data from a **database** based on specified conditions or criteria. It enables users to narrow down the scope of their queries, focusing only on the relevant information. so there are four methods discussed in this lecture to filter query results:

- **Using WHERE Clause**
- **Using Comparison Operator**
- **Using Logical Operators**
- **Using ORDER BY Clause**
- **Using GROUP BY Clause**

Example

```
CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    age INT,
    grade CHAR(1)
);

INSERT INTO students (name, age, grade) VALUES
('John', 20, 'A'),
('Emily', 22, 'B'),
('Michael', 21, 'A'),
('Sophia', 19, 'C'),
('William', 23, 'B');
```

Output

```
+----+---------+-----+-------+
| id | name    | age | grade |
+----+---------+-----+-------+
|  1 | John    |  20 | A     |
|  2 | Emily   |  22 | B     |
|  3 | Michael |  21 | A     |
|  4 | Sophia  |  19 | C     |
|  5 | William |  23 | B     |
+----+---------+-----+-------+
```

Using WHERE Clause

The **WHERE clause** is the fundamental tool for filtering data in MySQL queries. It allows you to specify conditions that the retrieved rows must meet.

Example

```
SELECT * FROM students WHERE age > 20;
```

Retrieve students who are older than 20 years.

Output

```
+----+---------+-----+-------+
| id | name    | age | grade |
+----+---------+-----+-------+
|  2 | Emily   |  22 | B     |
|  3 | Michael |  21 | A     |
|  5 | William |  23 | B     |
+----+---------+-----+-------+
```

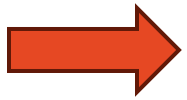
Returns students' records where age is over 20: '**Emily**', '**Michael**', '**William**' with ages 22, 21, and 23.

Using Comparison Operator

Comparison Operators in MySQL provides a range of comparison operators to refine your WHERE clause conditions. These include "=", "<>", ">", "<", ">=", "<=", etc.

Example

```
SELECT * FROM students WHERE grade = 'B';
```



Fetch students who have a grade of 'B'

Output

```
+----+--------+-----+-------+
| id | name   | age | grade |
+----+--------+-----+-------+
|  2 | Emily  |  22 | B     |
|  5 | William|  23 | B     |
+----+--------+-----+-------+
```

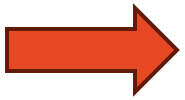
Returns students with grade 'B': Emily (age 22) and William (age 23) from the students table.

Using Logical Operators

Logical operators such as **AND**, **OR**, and **NOT** allow us to combine multiple conditions in a **WHERE clause**.

Example

```
SELECT * FROM students WHERE age < 20 OR grade = 'A';
```



Finds students who are either younger than 20 years old or have a grade of 'A'.

Output

```
+----+---------+-----+-------+
| id | name    | age | grade |
+----+---------+-----+-------+
|  1 | John    |  20 | A     |
|  3 | Michael |  21 | A     |
|  4 | Sophia  |  19 | C     |
+----+---------+-----+-------+
```

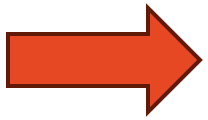
Selects all columns from 'students' where age is under 20 or grade is 'A'.

Using ORDER BY CLAUSE

While not strictly a filtering technique, the **ORDER BY** clause allows us to sort query results based on specified columns, making it easier to analyze the data.

Example

```
SELECT * FROM students ORDER BY age DESC;
```



Sorts students by age in descending order.

Output

```
+----+---------+-----+-------+
| id | name    | age | grade |
+----+---------+-----+-------+
|  5 | William |  23 | B     |
|  2 | Emily   |  22 | B     |
|  3 | Michael |  21 | A     |
|  1 | John    |  20 | A     |
|  4 | Sophia  |  19 | C     |
+----+---------+-----+-------+
```

Orders students by age in descending order: William (23), Emily (22), Michael (21), John (20), Sophia (19).

Using GROUP BY Clause

The GROUP BY clause aggregates rows that have the same values into summary rows, often used with aggregate functions (like COUNT, SUM, AVG, etc.).

Example

```
SELECT grade, COUNT(*) AS count_students
FROM students
GROUP BY grade;
```

This query groups students by grade, counting the number of students in each grade ('A', 'B', 'C'). Outputs include counts: 'A' has 2, 'B' has 2, and 'C' has 1 student.

Output

```
+-------+---------------+
| grade | count_students|
+-------+---------------+
|   A   |       2       |
|   B   |       2       |
|   C   |       1       |
+-------+---------------+
```

Conclusion

Filtering search results in MySQL is one of the most important abilities. With the help of techniques such as **WHERE clause**, **comparison** and **logical operators**, and specially designed filtering methods, users can gain the accuracy and timeliness they need to draw inferences from their data. By doing this, we learn how to construct a sharp database, conduct a smart analysis, and follow our decision-making with wisdom of our datasets.

Implementation

```sql
-- 1. Create a new database
CREATE DATABASE UniversityDB;

-- 2. Switch to the database
USE UniversityDB;

-- 3. Create a table named Customers
CREATE TABLE Customers (
    CustomerID INT,
    CustomerName VARCHAR(50),
    City VARCHAR(50),
    Country VARCHAR(50),
    Phone VARCHAR(20)
);
```

```sql
-- 4. Insert sample records
INSERT INTO Customers VALUES (1, 'John', 'Berlin', 'Germany', '123456');
INSERT INTO Customers VALUES (2, 'Emily', 'Paris', 'France', '987654');
INSERT INTO Customers VALUES (3, 'Michael', 'Berlin', 'Germany', '567890');
INSERT INTO Customers VALUES (4, 'Sophia', 'Rome', 'Italy', '998877');
INSERT INTO Customers VALUES (5, 'William', 'Paris', 'France', '110022');
```

Output

```sql
-- 5. SELECT specific columns
SELECT CustomerName, City, Country
FROM Customers;

-- 6. SELECT all columns
SELECT *
FROM Customers;
```

CustomerName	City	Country
John	Berlin	Germany
Emily	Paris	France
Michael	Berlin	Germany
Sophia	Rome	Italy
William	Paris	France

CustomerID	CustomerName	City	Country	Phone
1	John	Berlin	Germany	123456
2	Emily	Paris	France	987654
3	Michael	Berlin	Germany	567890
4	Sophia	Rome	Italy	998877
5	William	Paris	France	110022

```sql
-- 7. SELECT DISTINCT values
SELECT DISTINCT Country
FROM Customers;

-- 8. COUNT DISTINCT values
SELECT COUNT(DISTINCT Country)
FROM Customers;

-- 9. Filtering with WHERE
SELECT *
FROM students
WHERE age > 20;

-- 10. Comparison Operator Example
SELECT *
FROM students
WHERE grade = 'B';
```

Output

Country
Germany
France
Italy

COUNT(DISTINCT Country)
3

Name	Age	Grade
Sophia	19	A
Michael	21	A

Name	Age	Grade
Emily	22	B
William	23	B

Output

```sql
-- 11. Logical Operators
SELECT *
FROM students
WHERE age < 20 OR grade = 'A';

-- 12. ORDER BY (sort by age descending)
SELECT *
FROM students
ORDER BY age DESC;

-- 13. GROUP BY (count students by grade)
SELECT grade, COUNT(*)
FROM students
GROUP BY grade;
```

Name	Age	Grade
Sophia	19	A
Michael	21	A

Name	Age	Grade
William	23	B
Emily	22	B
Michael	21	A
John	20	C
Sophia	19	A

Grade	COUNT(*)
A	2
B	2
C	1

References

- Budy, S., Reese, G., & Tahaghoghi, S. M. M. (Year unknown). *Learning MySQL*. O'Reilly Media.
- Beaulieu, A. (Year unknown). *MySQL Cookbook* (4th ed.). O'Reilly Media.
- GeeksforGeeks. (n.d.). *How to filter query results in MySQL?* Retrieved from https://www.geeksforgeeks.org/mysql/how-to-filter-query-results-in-mysql/
- TutorialsPoint. (n.d.). *MySQL – SELECT Query*. Retrieved from https://www.tutorialspoint.com/mysql/mysql_select_query.htm


Any
Question