**LAB 02 Operators, Filtering and Sorting**

# OBJECTIVE(S)

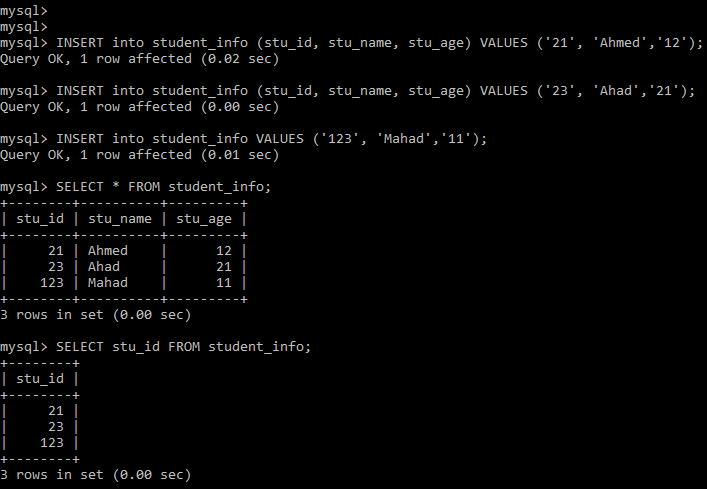
* Learn Operators
* Learn how to filter data
* Learn how to sort data

# QUERIES

## SELECT

In SQL, **SELECT** statement is used to select data from database tables. Often, you want to select subset of rows or columns or both. The result of **SELECT** statement is called a result set.

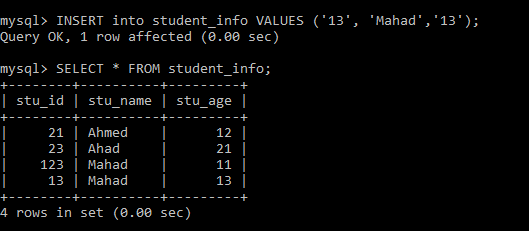
* **SELECT** \* **FROM** tb\_name;
* **SELECT** col1\_name, col2\_name **FROM** tb\_name;

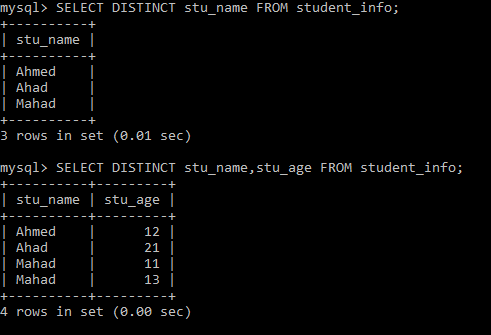


## SELECT DISTINCT

In SQL, you can use **DISTINCT** clause with **SELECT** statement to select only unique values or removes duplicates.

* **SELECT DISTINCT** col\_name **FROM** tb\_name;
* **SELECT DISTINCT** col1\_name, col2\_name **FROM** tb\_name;

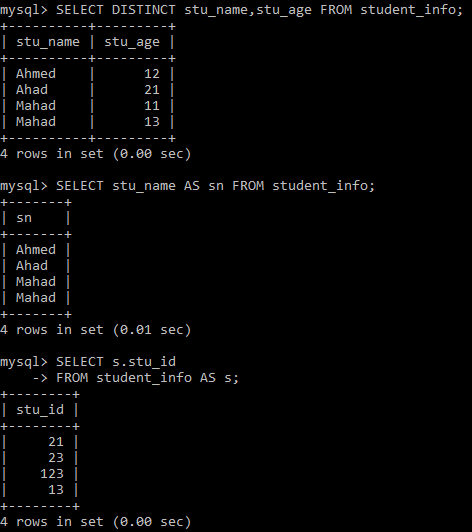




## SQL Aliases

MySQL supports two kinds of aliases, columns alias and table alias. Alias are used to give a table or columns a temporary name. Alias can also be used to make columns names more readable.

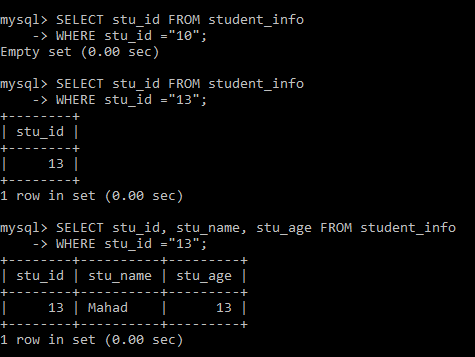
* **SELECT** col\_name **AS** aliased\_name **FROM** tb\_name;



## WHERE

The **WHERE** clause in **SELECT** statements can be used to filter the records. We can also use the **WHERE** clause with **UPDATE**, **DELETE** etc.

* **SELECT** col\_name **FROM** tb\_name **WHERE** condition;



|  |
| --- |
| TASK  * Alter the *student* table created in Lab 01 to have the following columns:   ID, Name, DOB, Semester, Department, GPA   * Insert/update the records with variations in DOB, semester, department, and GPA. * Display all the records. * Display only the ID, name, and department of all the students. * Display a list of all the departments (without any repetitions). * Display a list of all the departments and student name (without any repetitions). * Display the name as “Student Name”, department as “Department”, and DOB as “Date of Birth”. |

# OPERATORS

An operator is a character or reserved word in SQL. There are many operators that are primarily used with the **WHERE** clause to perform various operations, such as logical and comparisons operations.

* Arithmetic operators
* Comparison operators
* Logical operators

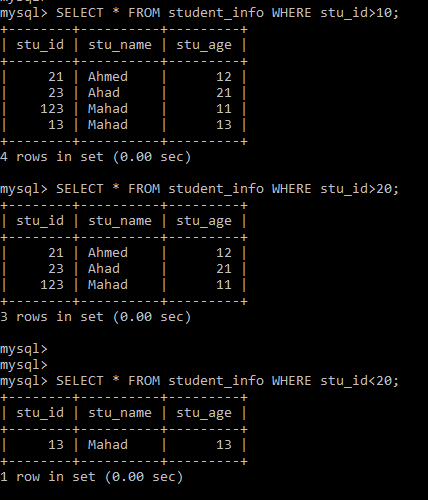
## Arithmetic Operators

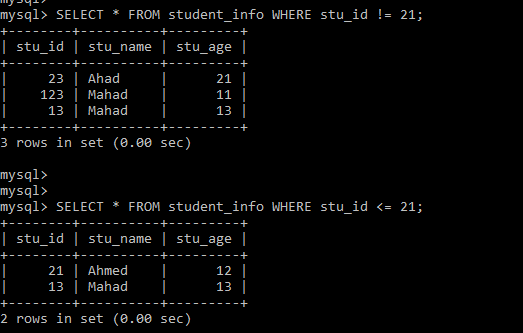
|  |  |
| --- | --- |
| Operator | Description |
| + (Addition) | Adds values on either side of the operator. |
| - (Subtraction) | Subtracts right hand operand from left hand operand. |
| \* (Multiplication) | Multiplies values on either side of the operator. |
| / (Division) | Divides left hand operand by right hand operand. |
| % (Modulus) | Divides left hand operand by right hand operand and returns remainder. |

## 

## Comparison Operators

|  |  |
| --- | --- |
| Operator | Description |
| = | Equal |
| <> or != | Not equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |





## Logical Operators

|  |  |
| --- | --- |
| Operator | Description |
| **AND** | The **AND** operator allows the existence of multiple conditions in an SQL statement's **WHERE** clause. The **AND** operator displays a result if all conditions in the **WHERE** clause are TRUE. |
| **OR** | The **OR** operator is used to combine multiple conditions in an SQL statement's **WHERE** clause. The **OR** operator displays a result if any one of the conditions in the **WHERE** clause is TRUE. |
| **NOT** | The **NOT** operator reverses the meaning of the logical operator with which it is used. The **NOT** operator displays a result if a condition is NOT TRUE. |
| **IN** | The **IN** operator is used to compare a value to a list of literal values that have been specified. The **IN** operator allows us to specify multiple values in the **WHERE** clause. |
| **BETWEEN** | The **BETWEEN** operator is used to search for values that are within a set of values, given the minimum value and the maximum value. The values can be numbers, text, and dates. |
| **LIKE** | The **LIKE** operator is used to **SELECT** data based on patterns. It is used to compare a value to similar values using wildcard operators. MySQL allows the use of two wildcards:   * Percentage (**%**) wildcard allows us to match any string of zero or more characters. * The Underscore (**\_**) wildcard allows us to match any single character. |
| **EXISTS** | The **EXISTS** operator is used to search for the presence of a row in a specified table that meets a certain criterion. |
| **ALL** | The **ALL** operator is used to compare a value to all values in another value set. |
| **ANY** | The **ANY** operator is used to compare a value to any applicable value in the list as per the condition. |
| **IS NULL** | The **IS** **NULL** operator is used to compare a value with a NULL value. |

### AND/OR/NOT/IN

* **SELECT** col\_name(s) **FROM** tb\_name

**WHERE** condition1 **AND** condition2;

* **SELECT** col\_name(s) **FROM** tb\_name

**WHERE** condition1 **OR** condition2;

* **SELECT** col\_name(s) **FROM** tb\_name

**WHERE** **NOT** condition1;

* **SELECT** col\_name(s) **FROM** tb\_name

**WHERE** col\_name **IN** (value1, value2);

### BETWEEN

* **SELECT** col\_name(s) **FROM** tb\_name

**WHERE** col\_name **BETWEEN** value1 **AND** value2;

* **SELECT** col\_name(s) **FROM** tb\_name

**WHERE** col\_name **NOT BETWEEN** value1 **AND** value2;

For string/text values:

* **SELECT** col\_name(s) **FROM** tb\_name

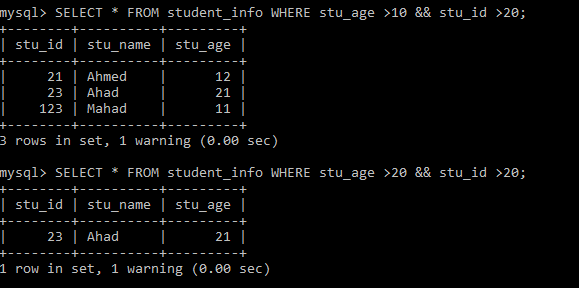
**WHERE** col\_name **NOT BETWEEN** “value1” **AND** “value2”;

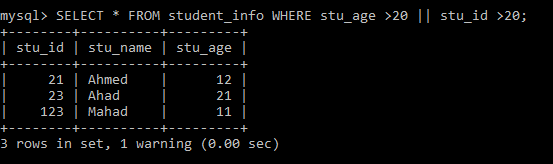
### LIKE

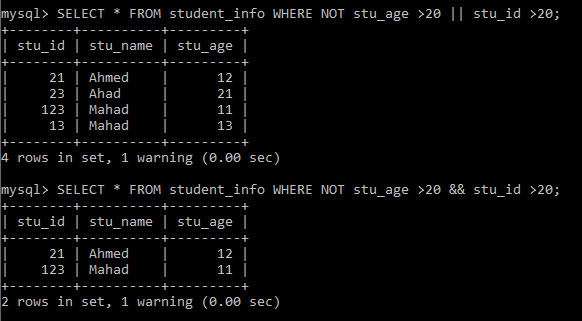
* **SELECT** col\_name(s) **FROM** tb\_name

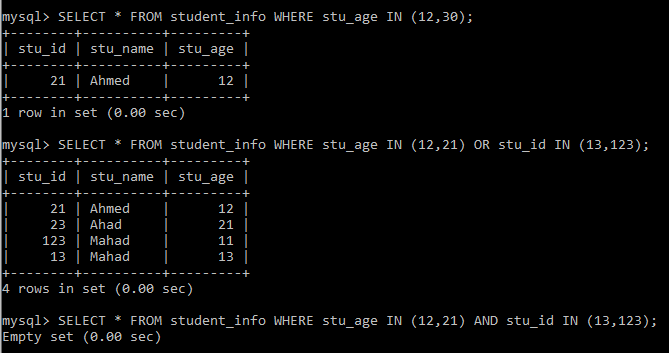
**WHERE** col\_name **LIKE** pattern;

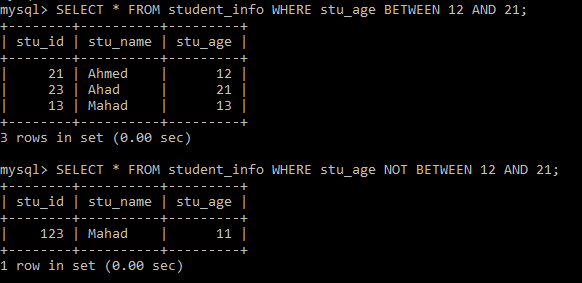
|  |  |
| --- | --- |
| LIKE “a%” | Values starts with ‘a’ |
| LIKE “%a” | Values ends with ‘a’ |
| LIKE “%a%” | Values having ‘a’ in any position |
| LIKE “\_a%” | Values having ‘a’ in the second position |
| LIKE “a%z” | Values starts with ‘a’ ends with ‘z’ |

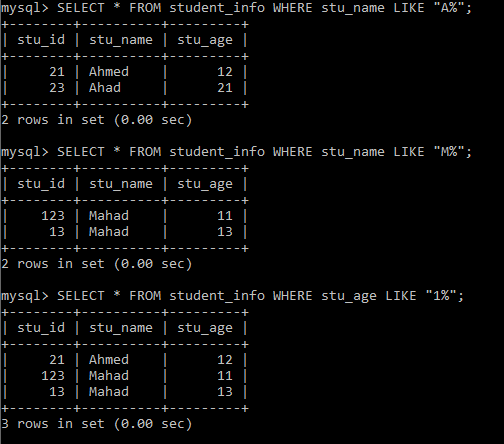


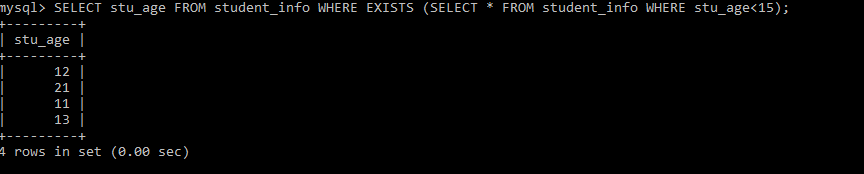












|  |
| --- |
| TASK  * Display all the information of all the students belonging to any one particular department. * Display only the name, semester, and GPA of all the students belonging to any one particular department. * Display the name, semester, and department of all the students who are not in a particular department. * Add a column marks and update the marks of all the students (out of 100). * Display the name, semester, and percentage of all the students belonging to a particular department. * Repeat the previous task by adding to the student’s marks and calculating the new percentage (out of 200). Display both the old percentage and new percentage. Label them accordingly. * Display the name, semester, GPA, and department of all the students whose GPA is less than 2.5 and who are in 5th semester or above. * Display the name and semester of students who belong to a particular department or whose GPA is greater than 3.5. * Display the name, DOB, and department of students born after 2000 or whose semester are not yet in the third semester. * Display the names and department of students who do not belong to any two departments without using AND. * Display the names and GPA of students whose GPA lies between 2.5 and 3.5. * Display a list of all the students with an “a” in their name. |

# LIMIT

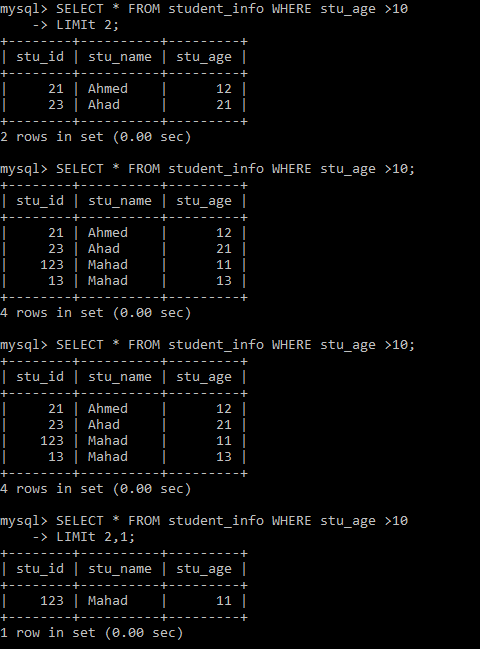
The **LIMIT** clause is used in the **SELECT** statement to force the number of records in a result set. The **LIMIT** clause accepts two arguments: **offset** and/or **count**. The values of both arguments must be zero or positive integer.

* **Offset** specifies the offset of the first row to return. The offset of the first row is 0, not 1.
* **Count** specifies the maximum number of rows to return.
* **SELECT** col\_names(s) **FROM** tb\_name

**LIMIT** offset, count;

* **SELECT** col\_names(s) **FROM** tb\_name

**LIMIT** count;

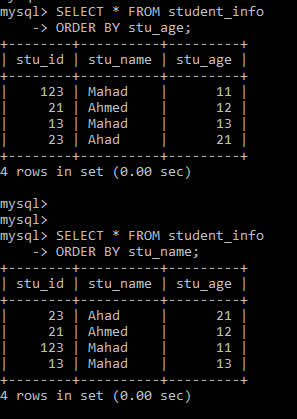


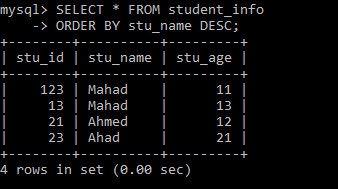
# ORDER BY

The **ORDER BY** clause in **SELECT** statement is used to sort the result set in ascending or descending order. The **ORDER BY** clause sorts the records in ascending order by default.

* **SELECT** col\_names(s) **FROM** tb\_name

**ORDER BY** col\_name(s) **ASC**/**DESC**;





|  |
| --- |
| TASK  * Display the names (in alphabetical order) and departments of all the students. * Display the name, department, and GPA of all the students ranked from highest to lowest GPA. * Display name, department, and GPA of the three students having the highest GPA. * Display the names and GPAs of the top three students of a particular department. * Display the names and GPAs of the 4th, 5th and 6th ranked students. * Display the names and GPAs of the students having the lowest three GPAs. |

# LAB ASSIGNMENT

1. Create a table **Employees** with following fields: ***employee\_id, first\_name, last\_name, designation, salary, birth\_date, hire\_date, department*** and ***city.***
2. Add at least 8 unique employees’ records (insert relevant data). Also, show structure of the table.
3. Display the names of all the employees along with their designation and department.
4. Display the name, salary and date of hiring of all the employees sorted by the date of hiring in descending order.
5. Display a list of all the departments in the organization (without any repetitions).
6. List all employees having a salary less than 40,000 and hired in 2018.
7. List all cities having the letter ‘r’ or ‘i’ in their name.
8. List all the departments with employees who have joined between February 2016 and October 2018 sorted by their date of hiring in descending order. For employees having the same hiring dates, sort records by salary in ascending order.
9. List all the employees who are Managers in Lahore.
10. Display the names of all Managers and Auditors along with their salaries. Also, show a 20% increment on their salaries under two new aliased fields, “Bonus” and “New Salary”.