SQL Operators

Click to Continue

SQL Operators are used in processing data values which are stored in columns of tables and returns a result. The data values are called *operands*.

The following are MYSQL operators

- Arithmetic Operators
- Single Row Comparison Operators
- Multiple Row Comparison Operators
- Logical Operators
- Set Operators

Arithmetic Operators

Click to Continue

Arithmetic operators are used to perform arithmetic operations of numeric operands (or) data stored in columns with numeric data type.

Operator	Example	
*	SELECT Age * 10 FROM Student;	Multiplication.
+	SELECT Age+ 1000 FROM Student;	Addition.
-	SELECT Age- 5 FROM Student;	Subtraction.
+(unary)	SELECT + Age FROM Student;	Makes operand positive
-(Unary)	SELECT - Age FROM Student;	Makes operand negative
/	SELECT Age/ 10 FROM Student;	Division

Single Row Comparison Operators

Click to Continue

Single Row Comparison operators are used to compare one operand against the other and the result is either a true/false.

Operator	Example
=	SELECT MARKS FROM STUDENT WHERE STUDENT_ID=3;
!=, <>	SELECT MARKS FROM STUDENT WHERE STUDENT_ID!=7;
>	SELECT MARKS FROM STUDENT WHERE STUDENT_ID>3;
<	SELECT MARKS FROM EMPLOYEE WHERE STUDENT_ID<3;
>=	SELECT MARKS FROM STUDENT WHERE STUDENT_ID >= 20;
<=	SELECT MARKS FROM STUDENT WHERE STUDENT_ID <= 150;

Equality Test.

Inequality Test.

Greater than test

Less than test

Greater than or equal to test.

Less than or equal to test.

Multiple Row Comparison Operators

Click to Continue

Multiple Row Comparison operators are used to perform comparison of one operand value with multiple values in a list .

Operator	Example

IN/NOT IN	SELECT STUDENT_ID FROM STUDENT WHERE NAME IN ('ADAM', 'JACK');
	
	SELECT STUDENT_ID FROM STUDENT WHERE NAME NOT IN ('ADAM', 'JACK');

BETWEEN AND/NOT BETWEEN AND	SELECT NAME,AGE FROM STUDENT WHERE AGE BETWEEN 18 AND 21; // Retrieves student records whose age is between 18 and 21. it also includes the records whose age either 18 or 21.
	
	SELECT NAME,AGE FROM STUDENT WHERE AGE NOT BETWEEN 18 AND 21; // Retrieves student records whose age is not between 18 and 21. it also excludes the records whose age either 18 or 21.

Equivalent to comparing the operand value with a list of values and if any match happens it returns true.

Check whether the operand value is between a range, this includes the lower and higher limits.

Multiple Row Comparison Operators

Click to Continue

Operator	Example

LIKE/NOT LIKE	SELECT NAME FROM STUDENT WHERE NAME LIKE '%SH'; //Select students whose name ends with SH Example: Ramesh, Suresh etc.
	
	SELECT NAME FROM STUDENT WHERE NAME LIKE 'Ra_'; //Select students whose name starts with Ra and ends with one character after it. Example: Ram, Rajetc.

ALL	SELECT NAME FROM STUDENT WHERE AGE >= ALL (select max(age) from student); // Compares if age is greater than all the values in the list namely 21,12,33,44 if so returns all the records which matches the above mentioned condition. The records which do not satisfy the condition for one of the values will not be retrieved.

The LIKE operator is used for wild card matching.
_ is used for single character.
% used for multiple or no character.

Compares a operand value with every value in a list. Only if the the condition is met for all the values it returns a true.

Multiple Row Comparison Operators

Click to Continue

Operator	Example
ANY/SOME	SELECT first_NAME FROM STUDENT WHERE first_name= SOME (select last name from students); // Returns the records which has first name same as last name of other students.
IS NULL/ IS NOT NULL	SELECT NAME FROM STUDENT WHERE NAME IS NOT NULL AND AGE> 15; // Returns the records who has a age > 15 and name is not null.

Compares a value to each value in a list returns the records which matches the condition for at least one of the value in the list . Evaluates to FALSE if the query returns no rows.

Tests for nulls. This is the only operator that should be used to test for nulls.

Logical Operators

Click to Continue

Logical operators are used for processing the result based on one or more conditions.

Illustration: If height > 160 OR weight < 55. An OR operator is used to fetch all records which matches the conditions.

Operator	Example
AND	SELECT STUDENT_NAME FROM STUDENT WHERE MARKS < 50 AND CLASS='12' ; // Retrieves the students records who have scored marks < 50 and studying in class 12.
OR	SELECT STUDENT_NAME FROM STUDENT WHERE ENGLISH_MARKS < 50 OR MATHS_MARKS<60 ; // Retrieves the students records who have scored either english marks < 50 or maths <60.
NOT	SELECT STUDENT_NAME FROM STUDENT WHERE NOT (AGE IS NULL) // Retrieves the student records for which the age is not null.

Used to combine two conditions. Returns TRUE if both condition are met. Returns FALSE if either of it is FALSE.

Returns TRUE if one of the condition returns TRUE. Returns FALSE if both are FALSE.

Returns TRUE if the condition returns FALSE. Returns FALSE if the return values is TRUE.

Set Operators

Click to Continue

Set operators are used to combine the results of two queries into a single result.

The two queries can be a select query from a same table or from different tables.

Operator Types:

1. **UNION** - Returns all unique rows selected by both the queries
2. **UNION ALL** - Returns all rows selected by either query, including all duplicates
3. **INTERSECT** - Returns all records which are common between the two result sets. (Not supported in My SQL)
4. **MINUS** - Returns all unique rows selected by the first query but not the second. (Not supported in My SQL)

MySQL Does not support Intersect.

You need to use work arounds to achieve intersects

Basic Rules in using Set Operators

Click to Continue

1. Queries combined should retrieve the same number of columns.

2. The columns must be of the same data type.

3.The length and name of the columns may be different.

4. Column names of first query will be column headings of the retrieved records.

```
Select ID, Name, Salary  from employee <Set operator>
Select code, last name, salary from Employee
```

ID	NAME	Salary
Hamilton	Lewis	32

4. Null values from both the queries are treated as identical. Normally two null values are not equal. In SET operations if MYSQL encounters two null values it will treat them as identical.

Union

Click to Continue

The *UNION* operator returns all rows selected by the queries combined.

- By default the results are stored in the ascending order of the first column of the select clause.

Syntax:

```
select column1,column1, .....,columnN  from table1

UNION

 select column1,column1, .....,columnN  from table2;
```

Illustration : Union

Click to Continue

Duplicate records exists in the table.

Customer

Customer_ID	Customer Name
101	Adam
102	Jack
103	Tim

Customer_Details

Customer_ID	Customer_Alias
101	Adam
105	Ron
102	Jill
105	Ron

Duplicate records within the table.

```
Select Customer_ID, Customer_Name from  Customer
                          UNION
Select Customer_ID, Customer_Alias from  Orders;
```

Output:

Customer_ID	Customer_Name
101	Adam
102	Jill
102	Jack
103	Tim
105	Ron

All the distinct records from both the tables will be fetched.

Union All Operator

Click to Continue

UNION ALL operator combines the result sets of one or more select statements and returns all records including duplicate records.

Syntax:

```
select column1,column1, .....,columnN  from table1

UNION ALL

 select column1,column1, .....,columnN  from table2;
```

Illustration : Union All

Click to Continue

Duplicate records exists in the table.

Customer

Customer_ID	Customer Name
101	Adam
102	Jack
103	Tim

Customer_Details

Customer_ID	Customer_Alias
101	Adam
105	Ron
102	Jill
105	Ron

Duplicate records within the table.

```
Select Customer_ID, Customer_Name from  Customer
                    UNION ALL
Select Customer_ID, Customer_Alias from  Orders;
```

Customer_ID	Customer_Name
101	Adam
102	Jill
102	Jack
103	Tim
105	Ron
101	Adam
105	Ron

All the records from both the tables will be fetched including duplicates

You have come to the end of the session.