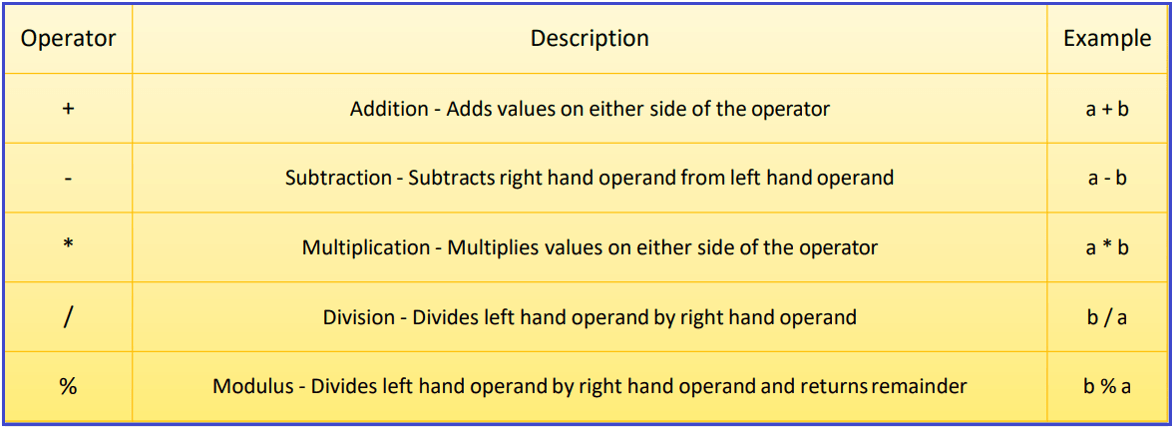
Arithmetic operator



SELECT 150 + 250; -- O/P = 400

SELECT 145 - 75; -- O/P = 70

SELECT 17 \* 5; -- O/P = 85

SELECT 49 / 7; -- O/P = 7.0000

SELECT 21 % 5; -- O/P = 1

CREATE DATABASE Company;

USE Company;

CREATE TABLE Employee (

Id INT PRIMARY KEY,

Name VARCHAR(45) NOT NULL,

Department VARCHAR(45) NOT NULL,

Salary FLOAT NOT NULL,

Gender VARCHAR(45) NOT NULL,

Age INT NOT NULL,

City VARCHAR(45) NOT NULL

);

INSERT INTO Employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1001, 'John Doe', 'IT', 35000, 'Male', 25, 'London');

INSERT INTO Employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1002, 'Mary Smith', 'HR', 45000, 'Female', 27, 'London');

INSERT INTO Employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1003, 'James Brown', 'Finance', 50000, 'Male', 28, 'London');

INSERT INTO Employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1004, 'Mike Walker', 'Finance', 50000, 'Male', 28, 'London');

INSERT INTO Employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1005, 'Linda Jones', 'HR', 75000, 'Female', 26, 'London');

INSERT INTO Employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1006, 'Anurag Mohanty', 'IT', 35000, 'Male', 25, 'Mumbai');

INSERT INTO Employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1007, 'Priyanla Dewangan', 'HR', 45000, 'Female', 27, 'Mumbai');

INSERT INTO Employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1008, 'Sambit Mohanty', 'IT', 50000, 'Male', 28, 'Mumbai');

INSERT INTO Employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1009, 'Pranaya Kumar', 'IT', 50000, 'Male', 28, 'Mumbai');

INSERT INTO Employee (Id, `Name`, Department, Salary, Gender, Age, City) VALUES (1010, 'Hina Sharma', 'HR', 75000, 'Female', 26, 'Mumbai');

Example: Our requirement is to calculate the salaries of each employee by adding 10% as a bonus. The following SQL Query will calculate the employee’s salary after the addition of 10% and show it in the Bonus along with the Id, Name, Salary column.

**SELECT ID, Name, Salary, Salary \* 1.1 AS Bonus FROM Employee;**

It is also possible to perform multiple mathematical calculations as shown in the below example.

**SELECT ID, Name, Salary, Salary \* 0.11 / 2.54 + 27.36 AS Bonus FROM Employee;**

**If you want to provide the order precedence of the arithmetic operators then simply use round brackets as shown in the below example.**

**SELECT ID, Name, Salary, Salary \* 0.11 / (2.54 + 27.36) AS Bonus FROM Employee;**

**What are SET Operators in MySQL?**

The SET Operators in MySQL are basically used to combine the result of more than 1 select statement and return the output as a single result set. In SQL, 4 types of set operators are. They are as follows:

1. **UNION**: It is used to combine two or more result sets into a single set, without duplicates.
2. **UNION ALL**: It is used to combine two or more result sets into a single set, including duplicates.
3. **INTERSECT**: It is used to combine two result sets and returns the data which are common in both the result set.
4. **EXCEPT**: It is used to combine two result sets and returns the data from the first result set which is not present in the second result set.

**Points to Remember while working with Set Operations:**

1. Every SELECT statement involved in the query must have a similar number of columns.
2. The columns in the SELECT statement must be in the same order and have similar data types.
3. In order to sort the result, an ORDER BY clause should be part of the last select statement. The column names or aliases must be found out by the first select statement.

**Examples to understand SET Operators in MySQL:**

**We are going to use the following EmployeeUK and EmployeeUSA tables to understand the SET Operators in MySQL.**

CREATE DATABASE EmployeeDB;

USE EmployeeDB;

CREATE TABLE EmployeeUK

(

EmployeeId INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Gender VARCHAR(10),

Department VARCHAR(20)

);

INSERT INTO EmployeeUK VALUES(1, 'Pranaya', 'Rout', 'Male','IT');

INSERT INTO EmployeeUK VALUES(2, 'Priyanka', 'Dewangan', 'Female','IT');

INSERT INTO EmployeeUK VALUES(3, 'Preety', 'Tiwary', 'Female','HR');

INSERT INTO EmployeeUK VALUES(4, 'Subrat', 'Sahoo', 'Male','HR');

INSERT INTO EmployeeUK VALUES(5, 'Anurag', 'Mohanty', 'Male','IT');

INSERT INTO EmployeeUK VALUES(6, 'Rajesh', 'Pradhan', 'Male','HR');

INSERT INTO EmployeeUK VALUES(7, 'Hina', 'Sharma', 'Female','IT');

CREATE TABLE EmployeeUSA

(

EmployeeId INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Gender VARCHAR(10),

Department VARCHAR(20)

);

INSERT INTO EmployeeUSA VALUES(1, 'James', 'Pattrick', 'Male','IT');

INSERT INTO EmployeeUSA VALUES(2, 'Priyanka', 'Dewangan', 'Female','IT');

INSERT INTO EmployeeUSA VALUES(3, 'Sara', 'Taylor', 'Female','HR');

INSERT INTO EmployeeUSA VALUES(4, 'Subrat', 'Sahoo', 'Male','HR');

INSERT INTO EmployeeUSA VALUES(5, 'Sushanta', 'Jena', 'Male','HR');

INSERT INTO EmployeeUSA VALUES(6, 'Mahesh', 'Sindhey', 'Female','HR');

INSERT INTO EmployeeUSA VALUES(7, 'Hina', 'Sharma', 'Female','IT');

**MySQL UNION Operator Example:**

The following query combines two select statements using the UNION operator. In our example, both the EmployeeUK and EmployeeUSA tables having seven records.

SELECT FirstName, LastName, Gender, Department FROM EmployeeUK

UNION

SELECT FirstName, LastName, Gender, Department FROM EmployeeUSA;

UNION ALL

SELECT FirstName, LastName, Gender, Department FROM EmployeeUK

UNION ALL

SELECT FirstName, LastName, Gender, Department FROM EmployeeUSA;

* + **UNION/UNION ALL with ORDER BY Clause in MySQL**

**The UNION/UNION ALL Operator can be used with the ORDER BY clause to sort the result returned from the query. Suppose we want to sort the employees by First Name column values. ORDER BY clause should be part of the last select statement. The SQL statement will be:**

**SELECT FirstName, LastName, Gender, Department FROM EmployeeUK**

**UNION**

**SELECT FirstName, LastName, Gender, Department FROM EmployeeUSA**

**ORDER BY FirstName;**

**MySQL EXCEPT Operator:**

**Using NOT IN Operator to achieve EXCEPT functionality:**

**Here, we are checking the FirstName column value only. Following is the SQL Query using the NOT IN Operator which returns the employees from the first EmployeeUK table that are not present in the EmployeeUSA table.**

**SELECT \* FROM EmployeeUK**

**WHERE FirstName NOT IN (SELECT FirstName FROM EmployeeUSA);**

**Using Join to achieve EXCEPT functionality in MySQL:**

**We can use LEFT JOIN to achieve the functionality of EXCEPT Operator. Here, the join clause needs to contain all 4 columns FirstName, LastName, Gender, and Department. The where clause picks null values in EmployeeId in EmployeeUSA, which limits to rows that exist in EmployeeUK only.**

**SELECT t1.\* FROM EmployeeUK AS t1**

**LEFT JOIN EmployeeUSA AS t2 ON**

**t1.FirstName=t2.FirstName**

**AND t1.LastName=t2.LastName**

**AND t1.Gender=t2.Gender**

**AND t1.Department=t2.Department**

**WHERE t2.EmployeeId IS NULL;**

**INTERSECT Operator in MySQL**

**Using IN Operator to achieve INTERSECT functionality:**

**Here, we are checking the FirstName column value only. Following is the SQL Query using the IN Operator which returns the common employees i.e. the employees which are present in both t EmployeeUK and EmployeeUSA tables. Here, we are checking common based on the First Name column value.**

**SELECT \* FROM EmployeeUK**

**WHERE FirstName IN (SELECT FirstName FROM EmployeeUSA);**

**Using Join to achieve INTERSECT functionality in MySQL:**

**We can use INNER JOIN to achieve the functionality of INTERSECT Operator. Here, the join clause needs to contain all 4 columns FirstName, LastName, Gender, and Department.**

**SELECT t1.\* FROM EmployeeUK AS t1**

**INNER JOIN EmployeeUSA AS t2 ON**

**t1.FirstName=t2.FirstName**

**AND t1.LastName=t2.LastName**

**AND t1.Gender=t2.Gender**

**AND t1.Department=t2.Department;**