SQL OPERATORS

Every database administrator and user use SQL queries for manipulating and accessing the data of database tables and views.

The manipulation and retrieving of the data are performed with the help of reserved words and characters, which are used to perform arithmetic operations, logical operations, comparison operations, compound operations, etc.

What is SQL Operator?

The SQL reserved words and characters are called operators, which are used with a WHERE clause in a SQL query. In SQL, an operator can either be a unary or binary operator. The unary operator uses only one operand for performing the unary operation, whereas the binary operator uses two operands for performing the binary operation.

Syntax of Unary SQL Operator

1. Operator SQL_Operand

Syntax of Unary SQL Operator

1. Operand1 SQL_Operator Operand2

Note: SQL operators are used for filtering the table's data by a specific condition in the SQL statement.

What is the Precedence of SQL Operator?

The precedence of SQL operators is the sequence in which the SQL evaluates the different operators in the same expression. Structured Query Language evaluates those operators first, which have high precedence.

In the following table, the operators at the top have high precedence, and the operators that appear at the bottom have low precedence.

SQL Operator Symbols	Operators
**	Exponentiation operator
+, -	Identity operator, Negation operator
*, /	Multiplication operator, Division operator
+, -, ||	Addition (plus) operator, subtraction (minus) operator, String Concatenation operator

=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN	Comparison Operators
NOT	Logical negation operator
&& or AND	Conjunction operator
OR	Inclusion operator

For Example,

```
1. UPDATE employee
2. SET salary = 20 - 3 * 5 WHERE Emp_Id = 5;
```

In the above SQL example, salary is assigned 5, not 85, because the * (Multiplication)

Operator has higher precedence than the - (subtraction) operator, so it first gets multiplied with 3*5 and then subtracts from 20.

Types of Operators

SQL operators are categorized in the following categories:

1. **SQL Arithmetic Operators**
2. **SQL Comparison Operators**
3. **SQL Logical Operators**
4. **SQL Set Operators**
5. **SQL Bit-wise Operators**
6. **SQL Unary Operators**

Let's discuss each operator with their types.

SQL Arithmetic Operators

The **Arithmetic Operators** perform the mathematical operation on the numerical data of the SQL tables. These operators perform addition, subtraction, multiplication, and division operations on the numerical operands.

Following are the various arithmetic operators performed on the SQL data:

1. SQL Addition Operator (+)
2. SQL Subtraction Operator (-)
3. SQL Multiplication Operator (+)

4. SQL Division Operator (-)
5. SQL Modulus Operator (+)

SQL Addition Operator (+)

The **Addition Operator** in SQL performs the addition on the numerical data of the database table. In SQL, we can easily add the numerical values of two columns of the same table by specifying both the column names as the first and second operand. We can also add the numbers to the existing numbers of the specific column.

Syntax of SQL Addition Operator:

1. SELECT operand1 + operand2;

Let's understand the below example which explains how to execute Addition Operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_Monthlybonus.**

Emp Id	Emp Name	Emp Salary	Emp Monthlybonus
101	Tushar	25000	4000
102	Anuj	30000	200

- Suppose, we want to add **20,000** to the salary of each employee specified in the table. Then, we have to write the following query in the SQL:

1. SELECT Emp_Salary + 20000 as Emp_New_Salary FROM Employee_details;

In this query, we have performed the SQL addition operation on the single column of the given table.

- Suppose, we want to add the Salary and monthly bonus columns of the above table, then we have to write the following query in SQL:

1. SELECT Emp_Salary + Emp_Monthlybonus as Emp_Total_Salary FROM Employee_det ails;

In this query, we have added two columns with each other of the above table.

SQL Subtraction Operator (-)

The Subtraction Operator in SQL performs the subtraction on the numerical data of the database table. In SQL, we can easily subtract the numerical values of two columns of the same table by specifying both the column names as the first and second operand. We can also subtract the number from the existing number of the specific table column.

Syntax of SQL Subtraction Operator:

1. SELECT operand1 - operand2;

Let's understand the below example which explains how to execute Subtraction Operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_Monthlybonus.**

Emp Id	Emp Name	Emp Salary	Penalty
201	Abhay	25000	200
202	Sumit	30000	500

- Suppose we want to subtract 5,000 from the salary of each employee given in the **Employee_details** table. Then, we have to write the following query in the SQL:

1. SELECT Emp_Salary - 5000 as Emp_New_Salary FROM Employee_details;

In this query, we have performed the SQL subtraction operation on the single column of the given table.

- If we want to subtract the penalty from the salary of each employee, then we have to write the following query in SQL:

1. SELECT Emp_Salary - Penalty as Emp_Total_Salary FROM Employee_details;

SQL Multiplication Operator (*)

The Multiplication Operator in SQL performs the Multiplication on the numerical data of the database table. In SQL, we can easily multiply the numerical values of two columns of the same table by specifying both the column names as the first and second operand.

Syntax of SQL Multiplication Operator:

1. SELECT operand1 * operand2;

Let's understand the below example which explains how to execute Multiplication Operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_Monthlybonus.**

Emp Id	Emp Name	Emp Salary	Penalty

201	Abhay	25000	200
202	Sumit	30000	500

- Suppose, we want to double the salary of each employee given in the **Employee_details** table. Then, we have to write the following query in the SQL:

```
1. SELECT Emp_Salary * 2 as Emp_New_Salary FROM Employee_details;
```

In this query, we have performed the SQL multiplication operation on the single column of the given table.

- If we want to multiply **the Emp_Id** column to **Emp_Salary** column of that employee whose **Emp_Id** is **202,** then we have to write the following query in SQL:

```
1. SELECT Emp_Id * Emp_Salary as Emp_Id * Emp_Salary FROM Employee_details WHE
   RE Emp_Id = 202;
```

In this query, we have multiplied the values of two columns by using the WHERE clause.

SQL Division Operator (/)

The Division Operator in SQL divides the operand on the left side by the operand on the right side.

Syntax of SQL Division Operator:

```
1. SELECT operand1 / operand2;
```

In SQL, we can also divide the numerical values of one column by another column of the same table by specifying both column names as the first and second operand.

We can also perform the division operation on the stored numbers in the column of the SQL table.

Let's understand the below example which explains how to execute Division Operator in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, and Emp_Salary.**

Emp Id	**Emp Name**	**Emp Salary**
201	Abhay	25000

202	Sumit	30000

- Suppose, we want to half the salary of each employee given in the Employee_details table. For this operation, we have to write the following query in the SQL:

```
1. SELECT Emp_Salary / 2 as Emp_New_Salary FROM Employee_details;
```

In this query, we have performed the SQL division operation on the single column of the given table.

SQL Modulus Operator (%)

The Modulus Operator in SQL provides the remainder when the operand on the left side is divided by the operand on the right side.

Syntax of SQL Modulus Operator:

```
1. SELECT operand1 % operand2;
```

Let's understand the below example which explains how to execute Modulus Operator in SQL query:

This example consists of a **Division** table, which has three columns **Number, First_operand, and Second_operand.**

Number	First operand	Second operand
1	56	4
2	32	8
3	89	9
4	18	10
5	10	5

- If we want to get the remainder by dividing the numbers of First_operand column by the numbers of Second_operand column, then we have to write the following query in SQL:

```
1. SELECT First_operand % Second_operand as Remainder FROM Employee_details;
```

SQL Comparison Operators

The **Comparison Operators** in SQL compare two different data of SQL table and check whether they are the same, greater, and lesser. The SQL comparison operators are used with the WHERE clause in the SQL queries

Following are the various comparison operators which are performed on the data stored in the SQL database tables:

1. SQL Equal Operator (=)
2. SQL Not Equal Operator (!=)
3. SQL Greater Than Operator (>)
4. SQL Greater Than Equals to Operator (>=)
5. SQL Less Than Operator (<)\
6. SQL Less Than Equals to Operator (<=)

SQL Equal Operator (=)

This operator is highly used in SQL queries. The **Equal Operator** in SQL shows only data that matches the specified value in the query.

This operator returns TRUE records from the database table if the value of both operands specified in the query is matched.

Let's understand the below example which explains how to execute Equal Operator in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, and Emp_Salary.**

Emp Id	Emp Name	Emp Salary
201	Abhay	30000
202	Ankit	40000
203	Bheem	30000
204	Ram	29000
205	Sumit	30000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose salary is 30000. Then, we have to write the following query in the SQL database:

1. SELECT * FROM Employee_details WHERE Emp_Salary = 30000;

In this example, we used the SQL equal operator with WHERE clause for getting the records of those employees whose salary is 30000.

SQL Equal Not Operator (!=)

The **Equal Not Operator** in SQL shows only those data that do not match the query's specified value.

This operator returns those records or rows from the database views and tables if the value of both operands specified in the query is not matched with each other.

Let's understand the below example which explains how to execute Equal Not Operator in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, and Emp_Salary.**

Emp Id	Emp Name	Emp Salary
201	Abhay	45000
202	Ankit	45000
203	Bheem	30000
204	Ram	29000
205	Sumit	29000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose salary is not 45000. Then, we have to write the following query in the SQL database:

1. SELECT * FROM Employee_details WHERE Emp_Salary != 45000;

In this example, we used the SQL equal not operator with WHERE clause for getting the records of those employees whose salary is not 45000.

SQL Greater Than Operator (>)

The **Greater Than Operator** in SQL shows only those data which are greater than the value of the right-hand operand.

Let's understand the below example which explains how to execute Greater ThanOperator (>) in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, and Emp_Salary.**

Emp Id	Emp Name	Emp Salary
201	Abhay	45000
202	Ankit	45000
203	Bheem	30000
204	Ram	29000
205	Sumit	29000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose employee id is greater than 202. Then, we have to write the following query in the SQL database:

```
1. SELECT * FROM Employee_details WHERE Emp_Id > 202;
```

Here, SQL greater than operator displays the records of those employees from the above table whose Employee Id is greater than 202.

SQL Greater Than Equals to Operator (>=)

The **Greater Than Equals to Operator** in SQL shows those data from the table which are greater than and equal to the value of the right-hand operand.

Let's understand the below example which explains how to execute greater than equals to the operator (>=) in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, and Emp_Salary.**

Emp Id	Emp Name	Emp Salary
201	Abhay	45000

202	Ankit	45000
203	Bheem	30000
204	Ram	29000
205	Sumit	29000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose employee id is greater than and equals to 202. For this, we have to write the following query in the SQL database:

```
1. SELECT * FROM Employee_details WHERE Emp_Id >= 202;
```

Here,**'SQL greater than equals to operator'** with WHERE clause displays the rows of those employees from the table whose Employee Id is greater than and equals to 202.

SQL Less Than Operator (<)

The **Less Than Operator** in SQL shows only those data from the database tables which are less than the value of the right-side operand.

This comparison operator checks that the left side operand is lesser than the right side operand. If the condition becomes true, then this operator in SQL displays the data which is less than the value of the right-side operand.

Let's understand the below example which explains how to execute less than operator (<) in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, and Emp_Salary.**

Emp Id	Emp Name	Emp Salary
201	Abhay	45000
202	Ankit	45000
203	Bheem	30000
204	Ram	29000
205	Sumit	29000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose employee id is less than 204. For this, we have to write the following query in the SQL database:

```
1. SELECT * FROM Employee_details WHERE Emp_Id < 204;
```

Here,**SQL less than operator** with WHERE clause displays the records of those employees from the above table whose Employee Id is less than 204.

SQL Less Than Equals to Operator (<=)

The **Less Than Equals to Operator** in SQL shows those data from the table which are lesser and equal to the value of the right-side operand.

This comparison operator checks that the left side operand is lesser and equal to the right side operand.

Let's understand the below example which explains how to execute less than equals to the operator (<=) in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, and Emp_Salary.**

Emp Id	Emp Name	Emp Salary
201	Abhay	45000
202	Ankit	45000
203	Bheem	30000
204	Ram	29000
205	Sumit	29000

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose employee id is less and equals **203.** For this, we have to write the following query in the SQL database:

```
1. SELECT * FROM Employee_details WHERE Emp_Id <= 203;
```

Here, SQL **less than equals to the operator** with WHERE clause displays the rows of those employees from the table whose Employee Id is less than and equals 202.

SQL Logical Operators

The **Logical Operators** in SQL perform the Boolean operations, which give two results **True and False.** These operators provide **True** value if both operands match the logical condition.

Following are the various logical operators which are performed on the data stored in the SQL database tables:

1. SQL ALL operator
2. SQL AND operator
3. SQL OR operator
4. SQL BETWEEN operator
5. SQL IN operator
6. SQL NOT operator
7. SQL ANY operator
8. SQL LIKE operator

SQL ALL Operator

The ALL operator in SQL compares the specified value to all the values of a column from the sub-query in the SQL database.

This operator is always used with the following statement:

1. SELECT,
2. HAVING, and
3. WHERE.

Syntax of ALL operator:

1. SELECT column_Name1, ..., column_NameN FROM table_Name WHERE column Co mparison_operator ALL (SELECT column FROM tablename2)

Let's understand the below example which explains how to execute ALL logical operators in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Gurgaon
202	Ankit	45000	Delhi

203	Bheem	30000	Jaipur
204	Ram	29000	Mumbai
205	Sumit	40000	Kolkata

- If we want to access the employee id and employee names of those employees from the table whose salaries are greater than the salary of employees who lives in Jaipur city, then we have to type the following query in SQL.

1. SELECT Emp_Id, Emp_Name FROM Employee_details WHERE Emp_Salary **>** ALL (SEL ECT Emp_Salary FROM Employee_details WHERE Emp_City = Jaipur)

Here, we used the **SQL ALL operator** with greater than the operator.

SQL AND Operator

The **AND operator** in SQL would show the record from the database table if all the conditions separated by the AND operator evaluated to True. It is also known as the conjunctive operator and is used with the WHERE clause.

Syntax of AND operator:

1. SELECT column1, ..., columnN FROM table_Name WHERE condition1 AND condition 2 AND condition3 AND ..... AND conditionN;

Let's understand the below example which explains how to execute AND logical operator in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	**Emp Name**	**Emp Salary**	**Emp City**
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to access all the records of those employees from the **Employee_details** table whose salary is 25000 and the city is Delhi. For this, we have to write the following query in SQL:
    - SELECT * FROM Employee_details WHERE Emp_Salary = 25000 OR Emp_City = 'Delhi';

Here,**SQL AND operator** with WHERE clause shows the record of employees whose salary is 25000 and the city is Delhi.

SQL OR Operator

The OR operator in SQL shows the record from the table if any of the conditions separated by the OR operator evaluates to True. It is also known as the conjunctive operator and is used with the WHERE clause.

Syntax of OR operator:

1. SELECT column1, ..., columnN FROM table_Name WHERE condition1 OR condition2 OR condition3 OR ..... OR conditionN;

Let's understand the below example which explains how to execute OR logical operator in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

- If we want to access all the records of those employees from the **Employee_details** table whose salary is 25000 or the city is Delhi. For this, we have to write the following query in SQL:

1. SELECT * FROM Employee_details WHERE Emp_Salary = 25000 OR Emp_City = 'Delhi' ;

Here, **SQL OR operator** with WHERE clause shows the record of employees whose salary is 25000 or the city is Delhi.

SQL BETWEEN Operator

The **BETWEEN operator** in SQL shows the record within the range mentioned in the SQL query. This operator operates on the numbers, characters, and date/time operands.

If there is no value in the given range, then this operator shows NULL value.

Syntax of BETWEEN operator:

1. SELECT column_Name1, column_Name2 ...., column_NameN FROM table_Name WH ERE column_nameBETWEEN value1 and value2;

Let's understand the below example which explains how to execute BETWEEN logical operator in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to access all the information of those employees from the **Employee_details** table who is having salaries between 20000 and 40000. For this, we have to write the following query in SQL:

1. SELECT * FROM Employee_details WHERE Emp_Salary BETWEEN 30000 AND 45000;

Here, we used the **SQL BETWEEN operator** with the Emp_Salary field.

SQL IN Operator

The **IN operator** in SQL allows database users to specify two or more values in a WHERE clause. This logical operator minimizes the requirement of multiple OR conditions.

This operator makes the query easier to learn and understand. This operator returns those rows whose values match with any value of the given list.

Syntax of IN operator:

1. SELECT column_Name1, column_Name2 ...., column_NameN FROM table_Name WH ERE column_name IN (list_of_values);

Let's understand the below example which explains how to execute IN logical operator in SQL query:

This example consists of an **Employee_details** table, which has three columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to show all the information of those employees from the **Employee_details** table whose **Employee Id** is 202, 204, and 205. For this, we have to write the following query in SQL:

1. SELECT * FROM Employee_details WHERE Emp_Id IN (202, 204, 205);

Here, we used the **SQL IN operator** with the Emp_Id column.

- Suppose, we want to show all the information of those employees from the **Employee_details** table whose **Employee Id** is not equal to 202 and 205. For this, we have to write the following query in SQL:

1. SELECT * FROM Employee_details WHERE Emp_Id NOT IN (202,205);
2.

Here, we used the **SQL NOT IN operator** with the Emp_Id column.

SQL NOT Operator

The **NOT operator** in SQL shows the record from the table if the condition evaluates to false. It is always used with the WHERE clause.

Syntax of NOT operator:

1. SELECT column1, column2 ...., columnN FROM table_Name WHERE NOT condition;

Let's understand the below example which explains how to execute NOT logical operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to show all the information of those employees from the **Employee_details** table whose Cityis not Delhi. For this, we have to write the following query in SQL:

1. SELECT * FROM Employee_details WHERE NOT Emp_City = 'Delhi' ;

In this example, we used the **SQL NOT operator** with the Emp_City column.

- Suppose, we want to show all the information of those employees from the **Employee_details** table whose Cityis not Delhi and Chandigarh. For this, we have to write the following query in SQL:

1. SELECT * FROM Employee_details WHERE NOT Emp_City = 'Delhi' AND NOT Emp_Cit y = 'Chandigarh';

In this example, we used the **SQL NOT operator** with the Emp_City column.

SQL ANY Operator

The **ANY operator** in SQL shows the records when any of the values returned by the sub-query meet the condition.

The ANY logical operator must match at least one record in the inner query and must be preceded by any SQL comparison operator.

Syntax of ANY operator:

1. SELECT column1, column2 ...., columnN FROM table_Name WHERE column_name co mparison_operator ANY (SELECT column_name FROM table_name WHERE conditio n(s)) ;

SQL LIKE Operator

The **LIKE operator** in SQL shows those records from the table which match with the given pattern specified in the sub-query.

The percentage (%) sign is a wildcard which is used in conjunction with this logical operator.

This operator is used in the WHERE clause with the following three statements:

1. SELECT statement
2. UPDATE statement
3. DELETE statement

Syntax of LIKE operator:

1. SELECT column_Name1, column_Name2 ...., column_NameN FROM table_Name WH ERE column_name LIKE pattern;

Let's understand the below example which explains how to execute LIKE logical operator in SQL query:

This example consists of an **Employee_details** table, which has four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Chandigarh
203	Saket	30000	Delhi
204	Abhay	25000	Delhi
205	Sumit	40000	Kolkata

- If we want to show all the information of those employees from the **Employee_details** whose name starts with ''s''. For this, we have to write the following query in SQL:

1. SELECT * FROM Employee_details WHERE Emp_Name LIKE 's%' ;

In this example, we used the SQL LIKE operator with **Emp_Name** column because we want to access the record of those employees whose name starts with s.

- If we want to show all the information of those employees from the **Employee_details**whose name ends with ''y''. For this, we have to write the following query in SQL:

```
1. SELECT * FROM Employee_details WHERE Emp_Name LIKE '%y' ;
```

- If we want to show all the information of those employees from the **Employee_details**whose name starts with ''S'' and ends with ''y''. For this, we have to write the following query in SQL:

```
1. SELECT * FROM Employee_details WHERE Emp_Name LIKE 'S%y' ;
```

SQL Set Operators

The **Set Operators** in SQL combine a similar type of data from two or more SQL database tables. It mixes the result, which is extracted from two or more SQL queries, into a single result.

Set operators combine more than one select statement in a single query and return a specific result set.

Following are the various set operators which are performed on the similar data stored in the two SQL database tables:

1. SQL Union Operator
2. SQL Union ALL Operator
3. SQL Intersect Operator
4. SQL Minus Operator

SQL Union Operator

The SQL Union Operator combines the result of two or more SELECT statements and provides the single output.

The data type and the number of columns must be the same for each SELECT statement used with the UNION operator. This operator does not show the duplicate records in the output table.

Syntax of UNION Set operator:

```
1. SELECT column1, column2 ...., columnN FROM table_Name1 [WHERE conditions]
2. UNION
3. SELECT column1, column2 ...., columnN FROM table_Name2 [WHERE conditions];
```

Let's understand the below example which explains how to execute Union operator in Structured Query Language:

In this example, **we used two tables.** Both tables have four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Delhi
203	Saket	30000	Aligarh

Table: Employee_details1

Emp Id	Emp Name	Emp Salary	Emp City
203	Saket	30000	Aligarh
204	Saurabh	40000	Delhi
205	Ram	30000	Kerala
201	Sanjay	25000	Delhi

Table: Employee_details2

- Suppose, we want to see the employee name and employee id of each employee from both tables in a single output. For this, we have to write the following query in SQL:

```
1. SELECT Emp_ID, Emp_Name FROM Employee_details1
2. UNION
3. SELECT Emp_ID, Emp_Name FROM Employee_details2 ;
```

SQL Union ALL Operator

The SQL Union Operator is the same as the UNION operator, but the only difference is that it also shows the same record.

Syntax of UNION ALL Set operator:

```
1. SELECT column1, column2 ...., columnN FROM table_Name1 [WHERE conditions]
```

2. UNION ALL
3. SELECT column1, column2 ...., columnN FROM table_Name2 [WHERE conditions];

Let's understand the below example which explains how to execute Union ALL operator in Structured Query Language:

In this example, **we used two tables.** Both tables have four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Delhi
203	Saket	30000	Aligarh

Table: Employee_details1

Emp Id	Emp Name	Emp Salary	Emp City
203	Saket	30000	Aligarh
204	Saurabh	40000	Delhi
205	Ram	30000	Kerala
201	Sanjay	25000	Delhi

Table: Employee_details2

- If we want to see the employee name of each employee of both tables in a single output. For this, we have to write the following query in SQL:

1. SELECT Emp_Name FROM Employee_details1
2. UNION ALL
3. SELECT Emp_Name FROM Employee_details2 ;

SQL Intersect Operator

The SQL Intersect Operator shows the common record from two or more SELECT statements. The data type and the number of columns must be the same for each SELECT statement used with the INTERSECT operator.

Syntax of INTERSECT Set operator:

1. SELECT column1, column2 ...., columnN FROM table_Name1 [WHERE conditions]
2. INTERSECT
3. SELECT column1, column2 ...., columnN FROM table_Name2 [WHERE conditions];

Let's understand the below example which explains how to execute INTERSECT operator in Structured Query Language:

In this example, **we used two tables.** Both tables have four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Delhi
203	Saket	30000	Aligarh

Table: Employee_details1

Emp Id	Emp Name	Emp Salary	Emp City
203	Saket	30000	Aligarh
204	Saurabh	40000	Delhi
205	Ram	30000	Kerala
201	Sanjay	25000	Delhi

Table: Employee_details2

Suppose, we want to see a common record of the employee from both the tables in a single output. For this, we have to write the following query in SQL:

1. SELECT Emp_Name FROM Employee_details1
2. INTERSECT
3. SELECT Emp_Name FROM Employee_details2 ;

SQL Minus Operator

The SQL Minus Operator combines the result of two or more SELECT statements and shows only the results from the first data set.

Syntax of MINUS operator:

1. SELECT column1, column2 ...., columnN FROM First_tablename [WHERE conditions]

2. MINUS
3. SELECT column1, column2 ...., columnN FROM Second_tablename [WHERE condition s];

Let's understand the below example which explains how to execute INTERSECT operator in Structured Query Language:

In this example, **we used two tables.** Both tables have four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Delhi
203	Saket	30000	Aligarh

Table: Employee_details1

Emp Id	Emp Name	Emp Salary	Emp City
203	Saket	30000	Aligarh
204	Saurabh	40000	Delhi
205	Ram	30000	Kerala
201	Sanjay	25000	Delhi

Table: Employee_details2

Suppose, we want to see the name of employees from the first result set after the combination of both tables. For this, we have to write the following query in SQL:

1. SELECT Emp_Name FROM Employee_details1

2. MINUS
3. SELECT Emp_Name FROM Employee_details2 ;

SQL Unary Operators

The **Unary Operators** in SQL perform the unary operations on the single data of the SQL table, i.e., these operators operate only on one operand.

These types of operators can be easily operated on the numeric data value of the SQL table.

Following are the various unary operators which are performed on the numeric data stored in the SQL table:

1. SQL Unary Positive Operator
2. SQL Unary Negative Operator
3. SQL Unary Bitwise NOT Operator

SQL Unary Positive Operator

The SQL Positive (+) operator makes the numeric value of the SQL table positive.

Syntax of Unary Positive Operator

1. SELECT +(column1), +(column2) ...., +(columnN) FROM table_Name [WHERE conditio ns] ;

Let's understand the below example which explains how to execute a Positive unary operator on the data of SQL table:

This example consists of an**Employee_details** table, which has four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Chandigarh
203	Saket	30000	Delhi
204	Abhay	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to see the salary of each employee as positive from the Employee_details table. For this, we have to write the following query in SQL:

1. SELECT +Emp_Salary Employee_details ;

SQL Unary Negative Operator

The SQL Negative (-) operator makes the numeric value of the SQL table negative.

Syntax of Unary Negative Operator

1. SELECT -(column_Name1), -(column_Name2) ...., -(column_NameN) FROM table_Name [WHERE conditions] ;

Let's understand the below example which explains how to execute Negative unary operator on the data of SQL table:

This example consists of an **Employee_details** table, which has four columns **Emp_Id, Emp_Name, Emp_Salary, and Emp_City.**

Emp Id	Emp Name	Emp Salary	Emp City
201	Sanjay	25000	Delhi
202	Ajay	45000	Chandigarh
203	Saket	30000	Delhi
204	Abhay	25000	Delhi
205	Sumit	40000	Kolkata

- Suppose, we want to see the salary of each employee as negative from the Employee_details table. For this, we have to write the following query in SQL:

1. SELECT -Emp_Salary Employee_details ;

- Suppose, we want to see the salary of those employees as negative whose city is Kolkatain the Employee_details table. For this, we have to write the following query in SQL:

1. SELECT -Emp_Salary Employee_details WHERE Emp_City = 'Kolkata';

SQL Bitwise NOT Operator

The SQL Bitwise NOT operator provides the one's complement of the single numeric operand. This operator turns each bit of numeric value. If the bit of any numerical value is 001100, then this operator turns these bits into 110011.

Syntax of Bitwise NOT Operator

1. SELECT ~(column1), ~(column2) ...., ~(columnN) FROM table_Name [WHERE conditio ns] ;

Let's understand the below example which explains how to execute the Bitwise NOT operator on the data of SQL table:

This example consists of a**Student_details** table, which has four columns **Roll_No, Stu_Name, Stu_Marks, and Stu_City.**

Emp Id	Stu Name	Stu Marks	Stu City
101	Sanjay	85	Delhi
102	Ajay	97	Chandigarh
103	Saket	45	Delhi
104	Abhay	68	Delhi
105	Sumit	60	Kolkata

If we want to perform the Bitwise Not operator on the marks column of **Student_details,** we have to write the following query in SQL:

1. SELECT ~Stu_Marks Employee_details ;

SQL Bitwise Operators

The **Bitwise Operators** in SQL perform the bit operations on the Integer values. To understand the performance of Bitwise operators, you just knew the basics of Boolean algebra.

Following are the two important logical operators which are performed on the data stored in the SQL database tables:

1. Bitwise AND (&)
2. Bitwise OR(|)

Bitwise AND (&)

The Bitwise AND operator performs the logical AND operation on the given Integer values. This operator checks each bit of a value with the corresponding bit of another value.

Syntax of Bitwise AND Operator

1. SELECT column1 & column2 & .... & columnN FROM table_Name [WHERE conditions] ;

Let's understand the below example which explains how to execute Bitwise AND operator on the data of SQL table:

This example consists of the following table, which has two columns. Each column holds numerical values.

When we use the Bitwise AND operator in SQL, then SQL converts the values of both columns in binary format, and the AND operation is performed on the converted bits.

After that, SQL converts the resultant bits into user understandable format, i.e., decimal format.

Column1	Column2
1	1
2	5
3	4
4	2
5	3

- Suppose, we want to perform the Bitwise AND operator between both the columns of the above table. For this, we have to write the following query in SQL:

```
1. SELECT Column1 & Column2 From TABLE_AND ;
```

Bitwise OR (|)

The Bitwise OR operator performs the logical OR operation on the given Integer values. This operator checks each bit of a value with the corresponding bit of another value.

Syntax of Bitwise OR Operator

```
1. SELECT column1 | column2 | .... | columnN FROM table_Name [WHERE conditions] ;
```

Let's understand the below example which explains how to execute Bitwise OR operator on the data of SQL table:

This example consists of a table that has two columns. Each column holds numerical values.

When we used the Bitwise OR operator in SQL, then SQL converts the values of both columns in binary format, and the OR operation is performed on the binary bits. After that, SQL converts the resultant binary bits into user understandable format, i.e., decimal format.

Column1	Column2
1	1
2	5
3	4
4	2
5	3

- Suppose, we want to perform the Bitwise OR operator between both the columns of the above table. For this, we have to write the following query in SQL:

1. SELECT Column1 | Column2 From TABLE_OR ;