**SQL | Arithmetic Operators**

**Arithmetic Operators are:**

**+ [Addition]**

**- [Subtraction]**

**/ [Division]**

**\* [Multiplication]**

**% [Modulus]**

**Addition (+) :**

**It is used to perform addition operation on the data items, items include either single column or multiple columns.**

**Implementation:**

**SELECT employee\_id, employee\_name, salary, salary + 100**

**AS "salary + 100" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY+100** |
| **1** | **alex** | **25000** | **25100** |
| **2** | **rr** | **55000** | **55100** |
| **3** | **jpm** | **52000** | **52100** |
| **4** | **ggshmr** | **12312** | **12412** |

**Here we have done addition of 100 to each Employee’s salary i.e, addition operation on single column.**

**Let’s perform addition of 2 columns:**

**SELECT employee\_id, employee\_name, salary, salary + employee\_id**

 **AS "salary + employee\_id" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY+EMPLOYEE\_ID** |
| **1** | **alex** | **25000** | **25001** |
| **2** | **rr** | **55000** | **55002** |
| **3** | **jpm** | **52000** | **52003** |
| **4** | **ggshmr** | **12312** | **12316** |

**Here we have done addition of 2 columns with each other i.e, each employee’s employee\_id is added with its salary.**

**Subtraction (-) :**

**It is use to perform subtraction operation on the data items, items include either single column or multiple columns.**

**Implementation:**

**SELECT employee\_id, employee\_name, salary, salary - 100**

 **AS "salary - 100" FROM subtraction;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY-100** |
| **12** | **Finch** | **15000** | **14900** |
| **22** | **Peter** | **25000** | **24900** |
| **32** | **Warner** | **5600** | **5500** |
| **42** | **Watson** | **90000** | **89900** |

**Here we have done subtraction of 100 to each Employee’s salary i.e, subtraction operation on single column.**

**Let’s perform subtraction of 2 columns:**

**SELECT employee\_id, employee\_name, salary, salary - employee\_id**

 **AS "salary - employee\_id" FROM subtraction;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY – EMPLOYEE\_ID** |
| **12** | **Finch** | **15000** | **14988** |
| **22** | **Peter** | **25000** | **24978** |
| **32** | **Warner** | **5600** | **5568** |
| **42** | **Watson** | **90000** | **89958** |

**Here we have done subtraction of 2 columns with each other i.e, each employee’s employee\_id is subtracted from its salary.**

**Multiplication (\*) :**

**It is use to perform multiplication of data items.**

**Implementation:**

**SELECT employee\_id, employee\_name, salary, salary \* 100**

 **AS "salary \* 100" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY \* 100** |
| **1** | **Finch** | **25000** | **2500000** |
| **2** | **Peter** | **55000** | **5500000** |
| **3** | **Warner** | **52000** | **5200000** |
| **4** | **Watson** | **12312** | **1231200** |

**Here we have done multiplication of 100 to each Employee’s salary i.e, multiplication operation on single column.**

**Let’s perform multiplication of 2 columns:**

**SELECT employee\_id, employee\_name, salary, salary \* employee\_id**

 **AS "salary \* employee\_id" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY \* EMPLOYEE\_ID** |
| **1** | **Finch** | **25000** | **25000** |
| **2** | **Peter** | **55000** | **110000** |
| **3** | **Warner** | **52000** | **156000** |
| **4** | **Watson** | **12312** | **49248** |

**Here we have done multiplication of 2 columns with each other i.e, each employee’s employee\_id is multiplied with its salary.**

**Modulus ( % ) :**

**It is use to get remainder when one data is divided by another.**

**Implementation:**

**SELECT employee\_id, employee\_name, salary, salary % 25000**

 **AS "salary % 25000" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY % 25000** |
| **1** | **Finch** | **25000** | **0** |
| **2** | **Peter** | **55000** | **5000** |
| **3** | **Warner** | **52000** | **2000** |
| **4** | **Watson** | **12312** | **12312** |

**Here we have done modulus of 100 to each Employee’s salary i.e, modulus operation on single column.**

**Let’s perform modulus operation between 2 columns:**

**SELECT employee\_id, employee\_name, salary, salary % employee\_id**

 **AS "salary % employee\_id" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY % EMPLOYEE\_ID** |
| **1** | **Finch** | **25000** | **0** |
| **2** | **Peter** | **55000** | **0** |
| **3** | **Warner** | **52000** | **1** |
| **4** | **Watson** | **12312** | **0** |

**Here we have done modulus of 2 columns with each other i.e, each employee’s salary is divided with its id and corresponding remainder is shown.**

**Basically, modulus is use to check whether a number is Even or Odd. Suppose a given number if divided by 2 and gives 1 as remainder, then it is an *odd number* or if on dividing by 2 and gives 0 as remainder, then it is an *even number*.**

**Concept of NULL :**

**If we perform any arithmetic operation on NULL, then answer is *always* null.**

**Implementation:**

**SELECT employee\_id, employee\_name, salary, type, type + 100**

 **AS "type+100" FROM addition;**

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **TYPE** | **TYPE + 100** |
| **1** | **Finch** | **25000** | **NULL** | **NULL** |
| **2** | **Peter** | **55000** | **NULL** | **NULL** |
| **3** | **Warner** | **52000** | **NULL** | **NULL** |
| **4** | **Watson** | **12312** | **NULL** | **NULL** |

**Here output always came null, since performing any operation on null will always result in a *null value*.**

**SQL | BETWEEN & IN Operator**

**BETWEEN**

**The SQL BETWEEN condition allows you to easily test if an expression is within a range of values (inclusive). The values can be text, date, or numbers. It can be used in a SELECT, INSERT, UPDATE, or DELETE statement. The SQL BETWEEN Condition will return the records where expression is within the range of value1 and value2.**

**Syntax:**

**SELECT column\_name(s)**

**FROM table\_name**

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

**Examples:
Consider the following Employee Table,
**

**Queries**

* **Using BETWEEN with Numeric Values:
List all the Employee Fname, Lname who is having salary between 30000 and 45000.**
* **SELECT Fname, Lname**
* **FROM Employee**
* **WHERE Salary BETWEEN 30000 AND 45000;**

**Output:
**

* **Using BETWEEN with Date Values:
Find all the Employee having Date of Birth Between 01-01-1985 and 12-12-1990.**
* **SELECT Fname, Lname**
* **FROM Employee**
* **where DOB BETWEEN '1985-01-01' AND '1990-12-30';**

**Output:
**

* **Using NOT operator with BETWEEN
Find all the Employee name whose salary is not in the range of 30000 and 45000.**
* **SELECT Fname, Lname**
* **FROM Emplyoee**
* **WHERE Salary NOT BETWEEN 30000 AND 45000;**

**Output:
**

**IN**

**IN operator allows you to easily test if the expression matches any value in the list of values. It is used to remove the need of multiple OR condition in SELECT, INSERT, UPDATE or DELETE. You can also use NOT IN to exclude the rows in your list.
Syntax:**

**SELECT column\_name(s)**

**FROM table\_name**

**WHERE column\_name IN (list\_of\_values);**

**Queries**

* **Find the Fname, Lname of the Employees who have Salary equal to 30000, 40000 or 25000.**
* **SELECT Fname, Lname**
* **FROM Employee WHERE Salary IN (30000, 40000, 25000);**

**Output:
**

* **Find the Fname, Lname of all the Employee who have Salary not equal to 25000 or 30000.**
* **SELECT Fname, Lname**
* **FROM Employee WHERE Salary NOT IN (25000, 30000);**

**Output:
**

# SQL | Arithmetic Operators

**Prerequisite:**[**Basic Select statement**](https://www.geeksforgeeks.org/sql-select-clause/)**,**[**Insert into clause**](https://www.geeksforgeeks.org/sql-insert-statement/)**,**[**Sql Create Clause**](https://www.geeksforgeeks.org/sql-create/)**,**[**SQL Aliases**](https://www.geeksforgeeks.org/sql-aliases/)

**We can use various Arithmetic Operators on the data stored in the tables.**

**Arithmetic Operators are:**

**+ [Addition]**

**- [Subtraction]**

**/ [Division]**

**\* [Multiplication]**

**% [Modulus]**

#### **Addition** (+) :

**It is used to perform addition operation on the data items, items include either single column or multiple columns.**

**Implementation:**

**SELECT employee\_id, employee\_name, salary, salary + 100**

 **AS "salary + 100" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY+100** |
| **1** | **alex** | **25000** | **25100** |
| **2** | **rr** | **55000** | **55100** |
| **3** | **jpm** | **52000** | **52100** |
| **4** | **ggshmr** | **12312** | **12412** |

**Here we have done addition of 100 to each Employee’s salary i.e, addition operation on single column.**

**Let’s perform addition of 2 columns:**

**SELECT employee\_id, employee\_name, salary, salary + employee\_id**

 **AS "salary + employee\_id" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY+EMPLOYEE\_ID** |
| **1** | **alex** | **25000** | **25001** |
| **2** | **rr** | **55000** | **55002** |
| **3** | **jpm** | **52000** | **52003** |
| **4** | **ggshmr** | **12312** | **12316** |

**Here we have done addition of 2 columns with each other i.e, each employee’s employee\_id is added with its salary.**

#### **Subtraction** (-) :

**It is use to perform subtraction operation on the data items, items include either single column or multiple columns.**

**Implementation:**

**SELECT employee\_id, employee\_name, salary, salary - 100**

 **AS "salary - 100" FROM subtraction;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY-100** |
| **12** | **Finch** | **15000** | **14900** |
| **22** | **Peter** | **25000** | **24900** |
| **32** | **Warner** | **5600** | **5500** |
| **42** | **Watson** | **90000** | **89900** |

**Here we have done subtraction of 100 to each Employee’s salary i.e, subtraction operation on single column.**

**Let’s perform subtraction of 2 columns:**

**SELECT employee\_id, employee\_name, salary, salary - employee\_id**

 **AS "salary - employee\_id" FROM subtraction;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY – EMPLOYEE\_ID** |
| **12** | **Finch** | **15000** | **14988** |
| **22** | **Peter** | **25000** | **24978** |
| **32** | **Warner** | **5600** | **5568** |
| **42** | **Watson** | **90000** | **89958** |

**Here we have done subtraction of 2 columns with each other i.e, each employee’s employee\_id is subtracted from its salary.**

**Division (/) : For Division refer this link- [Division in SQL](https://www.geeksforgeeks.org/sql-division/)**

#### **Multiplication** (\*) :

**It is use to perform multiplication of data items.**

**Implementation:**

**SELECT employee\_id, employee\_name, salary, salary \* 100**

 **AS "salary \* 100" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY \* 100** |
| **1** | **Finch** | **25000** | **2500000** |
| **2** | **Peter** | **55000** | **5500000** |
| **3** | **Warner** | **52000** | **5200000** |
| **4** | **Watson** | **12312** | **1231200** |

**Here we have done multiplication of 100 to each Employee’s salary i.e, multiplication operation on single column.**

**Let’s perform multiplication of 2 columns:**

**SELECT employee\_id, employee\_name, salary, salary \* employee\_id**

 **AS "salary \* employee\_id" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY \* EMPLOYEE\_ID** |
| **1** | **Finch** | **25000** | **25000** |
| **2** | **Peter** | **55000** | **110000** |
| **3** | **Warner** | **52000** | **156000** |
| **4** | **Watson** | **12312** | **49248** |

**Here we have done multiplication of 2 columns with each other i.e, each employee’s employee\_id is multiplied with its salary.**

#### **Modulus** ( % ) :

**It is use to get remainder when one data is divided by another.**

**Implementation:**

**SELECT employee\_id, employee\_name, salary, salary % 25000**

 **AS "salary % 25000" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY % 25000** |
| **1** | **Finch** | **25000** | **0** |
| **2** | **Peter** | **55000** | **5000** |
| **3** | **Warner** | **52000** | **2000** |
| **4** | **Watson** | **12312** | **12312** |

**Here we have done modulus of 100 to each Employee’s salary i.e, modulus operation on single column.**

**Let’s perform modulus operation between 2 columns:**

**SELECT employee\_id, employee\_name, salary, salary % employee\_id**

 **AS "salary % employee\_id" FROM addition;**

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **SALARY % EMPLOYEE\_ID** |
| **1** | **Finch** | **25000** | **0** |
| **2** | **Peter** | **55000** | **0** |
| **3** | **Warner** | **52000** | **1** |
| **4** | **Watson** | **12312** | **0** |

**Here we have done modulus of 2 columns with each other i.e, each employee’s salary is divided with its id and corresponding remainder is shown.**

**Basically, modulus is use to check whether a number is Even or Odd. Suppose a given number if divided by 2 and gives 1 as remainder, then it is an odd number or if on dividing by 2 and gives 0 as remainder, then it is an even number.**

#### **Concept of NULL** :

**If we perform any arithmetic operation on NULL, then answer is always null.**

**Implementation:**

**SELECT employee\_id, employee\_name, salary, type, type + 100**

 **AS "type+100" FROM addition;**

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMPLOYEE\_ID** | **EMPLOYEE\_NAME** | **SALARY** | **TYPE** | **TYPE + 100** |
| **1** | **Finch** | **25000** | **NULL** | **NULL** |
| **2** | **Peter** | **55000** | **NULL** | **NULL** |
| **3** | **Warner** | **52000** | **NULL** | **NULL** |
| **4** | **Watson** | **12312** | **NULL** | **NULL** |

**Here output always came null, since performing any operation on null will always result in a null value.**

**Note: Make sure that NULL is unavailable, unassigned, unknown. Null is not same as blank space or zero.
To get in depth understanding of NULL, refer [THIS link](https://www.geeksforgeeks.org/sql-null/%22%20%5Ct%20%22_blank).**

**SQL AND, OR, NOT – SQL Logical Operators**

SQL provides logical operators which helps in filtering the result set based on some condition. SQL logical operators that we will be discussing are AND, OR and NOT. These three are the most commonly used [logical operators](https://www.journaldev.com/18216/sql-operators) in SQL queries.

**1. SQL AND Operator**

SQL AND operator are used when we want to combine multiple conditions as part of the WHERE clause. The result set will be filtered based on the satisfaction of both the condition. So, if both the conditions are true then only the result will be filtered. To combine multiple conditions, we can use more than one AND as part of the [WHERE](https://www.journaldev.com/18411/sql-where-clause) clause.

**1.1) SQL single AND operator example**

SQL AND operator syntax is:

SELECT column(s) FROM table\_name WHERE condition1 AND condition2;

As mentioned in the syntax above for combining two conditions we can use one AND operator.

We will now try to understand one AND operator through some example.

Let’s consider the following Student table for example purpose.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RollNo | StudentName | StudentGender | StudentAge | StudentPercent |
| 1 | George | M | 14 | 85 |
| 2 | Monica | F | 13 | 88 |
| 3 | Jessica | F | 14 | 84 |
| 4 | Tom | M | 13 | 78 |

**Scenario**: Get the percentage of students whose age is more than 12 years and gender is female.

SELECT StudentPercent FROM Student WHERE StudentAge>12 AND StudentGender = "F";

Output:

|  |
| --- |
| StudentPercent |
| 88 |
| 84 |

In the example above, we have used one AND operator to combine two conditions, StudentAge is greater than 12 and StudentGender is equal to “F”.

Let’s try to see an example for multiple AND operator.

**1.2) SQL multiple AND operator example**

SELECT column(s) FROM table\_name WHERE condition1 AND condition2 AND condition3…AND conditionN;

As mentioned in the syntax above for combining more than two conditions we can use more than one AND operator.

We will now try to understand more than one AND operator through some example. Let’s consider the earlier defined Student table for example purpose.

**Scenario**: Get the percentage of students whose age is more than 12 years and gender are female and the percentage is more than 80.

SELECT StudentPercent FROM Student WHERE StudentAge>12 AND StudentGender = "F" AND StudentPercent>80;

Output:

|  |
| --- |
| StudentPercent |
| 85 |
| 88 |
| 84 |

In the example above, we have used one AND operator to combine three conditions, StudentAge is greater than 12, StudentGender is equal to “F” and StudentPercent is greater than 80.

**2. SQL OR Operator**

OR operator is used when we want to combine multiple conditions as part of the WHERE clause. The result set will be filtered based on satisfaction of at least one of the conditions. So, if at least one of the conditions are true than only the result will be filtered. To combine multiple conditions, we can use more than one OR as part of the WHERE clause.

**2.1) SQL single OR operator example**

**Syntax**:

SELECT column(s) FROM table\_name WHERE condition1 OR condition2;

As mentioned in the syntax above for combining two conditions we can use one OR operator.
We will now try to understand one OR operator through some example. Let’s reuse the earlier defined Student table for example purpose.

**Scenario**: Get the percentage of students whose age is more than 12 years or gender are female.

**Query**:

SELECT StudentPercent FROM Student WHERE StudentAge>12 OR StudentGender = "F";

**Output**:

|  |
| --- |
| StudentPercent |
| 85 |
| 88 |
| 84 |
| 78 |

In the example above, we have used one OR operator to combine two conditions, StudentAge is greater than 12 and StudentGender is equal to “F”.

Let’s try to understand for multiple OR operator.

**2.2) SQL multiple OR operator example**

**Syntax**:

SELECT column(s) FROM table\_name WHERE condition1 OR condition2 OR condition3 ... OR conditionN;

As mentioned in the syntax above for combining more than two conditions we can use more than one OR operator.

We will now try to understand more than one OR operator through some example.

**Scenario**: Get the percentage of students whose age is more than 12 years or gender is female or percentage is more than 80.

**Query**:

SELECT StudentPercent FROM Student WHERE StudentAge>12 OR StudentGender = "F" OR StudentPercent>80;

In the example above, we have used one OR operator to combine three conditions, StudentAge is greater than 12, StudentGender is equal to “F” and StudentPercent is greater than 80.

**3. SQL NOT Operator**

SQL NOT operator is used when we want to filter result set when the condition is not satisfied in the WHERE clause.

Let’s try to understand NOT operator in detail with some examples.

**3.1) SQL NOT operator example**

**Syntax**:

SELECT column(s) FROM table\_name WHERE NOT condition;

As mentioned in the syntax above we use NOT operator along with WHERE clause. We will now try to understand NOT operator through some example.

**Scenario**: Get the percentage of students whose gender is not female.

**Query**:

SELECT StudentPercent FROM Student WHERE NOT StudentGender = "F";

**Output**:

|  |
| --- |
| StudentPercent |
| 85 |
| 78 |

In the example above, we have used NOT operator to identify if the gender of the student is not female.

.

**SQL Comparison operator**

Comparison operator

A comparison (or relational) operator is a mathematical symbol which is used to compare two values.

Comparison operators are used in conditions that compares one expression with another. The result of a comparison can be TRUE, FALSE, or UNKNOWN (an operator that has one or two NULL expressions returns UNKNOWN).

The following table describes different types of comparison operators -

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Operates on** |
| = | Equal to. | Any compatible data types |
| > | Greater than. | Any compatible data types |
| < | Less than. | Any compatible data types |
| >= | Greater than equal to. | Any compatible data types |
| <= | Less than equal to. | Any compatible data types |
| <> | Not equal to. | Any compatible data types |

**Syntax :**

SELECT[column\_name| \* |expression]<comparison operator>

[column\_name | \* | expression ]

FROM <table\_name>

WHERE <expression>[comparison operator]<expression>;

**Parameters:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| column\_name | Name of the column of a table. |
| \* | Indicates all the columns of a table. |
| expression | Expression made up of a single constant, variable, scalar function, or column name and can also be the pieces of a SQL query that compare values against other values or perform arithmetic calculations. |
| table\_name | Name of the table. |
| comparison operator | Equal to (=), not equal to(<>), greater than(>), less than(<), greater than or equal to (>=), less than or equal to (<=). |

**Contents:**

* [Equal to Operator](https://www.w3resource.com/sql/comparison-operators/sql-comparison-operators.php%22%20%5Cl%20%22EQUAL)
* [Greater than Operator](https://www.w3resource.com/sql/comparison-operators/sql-comparison-operators.php%22%20%5Cl%20%22GREATER)
* [Less than Operator](https://www.w3resource.com/sql/comparison-operators/sql-comparison-operators.php%22%20%5Cl%20%22LESS)
* [Greater than or equal to Operator](https://www.w3resource.com/sql/comparison-operators/sql-comparison-operators.php%22%20%5Cl%20%22GE)
* [Less than or equal to Operator](https://www.w3resource.com/sql/comparison-operators/sql-comparison-operators.php%22%20%5Cl%20%22LE)
* [Not equal to Operator](https://www.w3resource.com/sql/comparison-operators/sql-comparison-operators.php%22%20%5Cl%20%22NOTEQ)

**Example: SQL Comparison operator**

To get a comparison between two numbers from the [DUAL](https://www.w3resource.com/sql-dual-table.php) table, the following SQL statement can be used :

SELECT 15>14 FROM dual;

Copy

SQL Equal to ( = ) operator

The equal to operator is used for equality test within two numbers or expressions.

**Example:**

Sample table: agents

To get data of all columns from the 'agents' table with the following condition -

1. 'commission' is equal to .15,

the following SQL statement can be used :

**SQL Code:**

SELECT \* FROM agents

WHERE commission = 0.15;

Output:

AGENT\_ AGENT\_NAME WORKING\_AREA COMMISSION PHONE\_NO COUNTRY

------ ------------------------------ ----------------- ---------- --------------- --------

A007 Ramasundar Bangalore .15 077-25814763

A011 Ravi Kumar Bangalore .15 077-45625874

A006 McDen London .15 078-22255588

A004 Ivan Torento .15 008-22544166

SQL Greater than ( > ) operator

The greater than operator is used to test whether an expression (or number) is greater than another one.

**Example:**

To get data of all columns from the 'agents' table with the following condition -

1. 'commission' is greater than .14,

the following SQL statement can be used :

**SQL Code:**

SELECT \*

FROM agents

WHERE commission> 0.14;

Copy

Output:

AGENT\_ AGENT\_NAME WORKING\_AREA COMMISSION PHONE\_NO COUNTRY

------ -------------- ----------------------------------- ---------- --------------- -------

A007 Ramasundar Bangalore .15 077-25814763

A011 Ravi Kumar Bangalore .15 077-45625874

A006 McDen London .15 078-22255588

A004 Ivan Torento .15 008-22544166

SQL Less than ( < ) operator

The less than operator is used to test whether an expression (or number) is less than another one.

**Example:**

To get data of all columns from the 'agents' table with the following condition -

1. 'commission' is less than .12,

the following SQL statement can be used :

**SQL Code:**

SELECT \*

FROM agents

WHERE commission < 0.12;

Copy

Output:

AGENT\_ AGENT\_NAME WORKING\_AREA COMMISSION PHONE\_NO COUNTRY

------ -------------- ----------------------------------- ---------- --------------- ---------

A009 Benjamin Hampshair .11 008-22536178

A002 Mukesh Mumbai .11 029-12358964

SQL Greater than or equal to ( >= ) operator

The greater than equal to operator is used to test whether an expression (or number) is either greater than or equal to another one.

**Example:**

To get data of all columns from the 'agents' table with the following condition -

1. 'commission' is greater than or equal to .14,

the following SQL statement can be used :

**SQL Code:**

SELECT \*

FROM agents

WHERE commission >= 0.14;

Copy

Output:

AGENT\_ AGENT\_NAME WORKING\_AREA COMMISSION PHONE\_NO COUNTRY

------ --------------- ----------------------------------- ---------- --------------- --------

A001 Subbarao Bangalore .14 077-12346674

A007 Ramasundar Bangalore .15 077-25814763

A011 Ravi Kumar Bangalore .15 077-45625874

A010 Santakumar Chennai .14 007-22388644

A006 McDen London .15 078-22255588

A004 Ivan Torento .15 008-22544166

SQL Less than or equal to ( <= ) operator

The less than equal to operator is used to test whether an expression (or number) is either less than or equal to another one.

**Example:**

To get data of all columns from the 'agents' table with the following condition -

1. commission is less than or equal to .12,

the following SQL statement can be used :

**SQL Code:**

SELECT \*

FROM agents

WHERE commission <= 0.12;

Copy

Output:

AGENT\_ AGENT\_NAME WORKING\_AREA COMMISSION PHONE\_NO COUNTRY

------ --------------- ---------------------------------- ---------- --------------- ---------

A009 Benjamin Hampshair .11 008-22536178

A008 Alford New York .12 044-25874365

A012 Lucida San Jose .12 044-52981425

A002 Mukesh Mumbai .11 029-12358964

SQL Not equal to ( <> ) operator

The not equal to operator is used for inequality test between two numbers or expression.

**Example:**

To get data of all columns from the 'agents' table with the following condition -

1. commission is not equal to .15,

the following SQL statement can be used :

SQL Code:

SELECT \*

FROM agents

WHERE commission <> 0.15;

Copy

Output:

AGENT\_ AGENT\_NAME WORKING\_AREA COMMISSION PHONE\_NO COUNTRY

------ ------------- ----------------------------------- ---------- --------------- --------

A003 Alex London .13 075-12458969

A001 Subbarao Bangalore .14 077-12346674

A009 Benjamin Hampshair .11 008-22536178

A008 Alford New York .12 044-25874365

A010 Santakumar Chennai .14 007-22388644

A012 Lucida San Jose .12 044-52981425

A005 Anderson Brisban .13 045-21447739

A002 Mukesh Mumbai .11 029-12358964