

SQL Operators

SQL Arithmetic Operators

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo

SQL Comparison Operators

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

SQL Logical Operators

- These operators return a Boolean value which can be either a TRUE or FALSE.
- Logical operators available in SQL are,
 - ANY and ALL
 - AND, OR and NOT
 - BETWEEN
 - EXISTS
 - IN
 - LIKE
 - IS NULL

ANY Operator

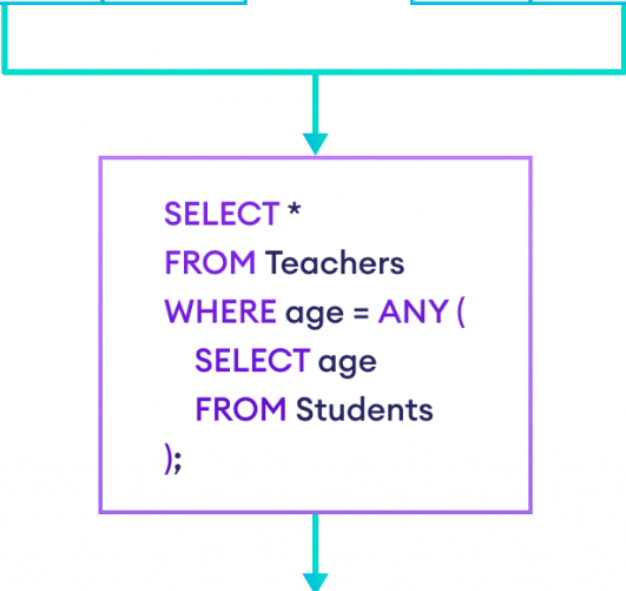
- SQL ANY compares a value of the first table with all values of the second table and returns the row if there is a match with any value.
- It has the following syntax:
- `SELECT column FROM table1 WHERE column OPERATOR ANY (SELECT column FROM table2);`
- `SELECT * FROM Teachers WHERE age = ANY (SELECT age FROM Students);`

Table: Teachers

id	name	age
1	Peter	32
2	Megan	43
3	Rose	29
4	Linda	30
5	Mary	41

Table: Students

id	name	age
1	Harry	23
2	Jack	42
3	Joe	32
4	Dent	23
5	Bruce	40



```
SELECT *  
FROM Teachers  
WHERE age = ANY (  
  SELECT age  
  FROM Students  
);
```

id	name	age
1	Peter	32

ALL Operator

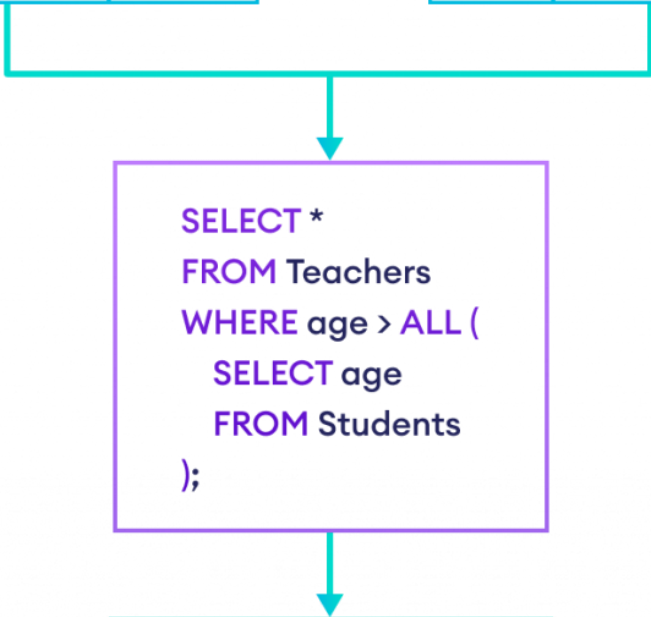
- SQL ALL compares a value of the first table with all values of the second table and returns the row if there is a match with all values.
- It has the following syntax:
- *SELECT column FROM table1 WHERE column OPERATOR ALL (SELECT column FROM table2);*
- `SELECT * FROM Teachers WHERE age > ALL (SELECT age FROM Students);`

Table: Teachers

id	name	age
1	Peter	32
2	Megan	43
3	Rose	29
4	Linda	30
5	Mary	41

Table: Students

id	name	age
1	Harry	23
2	Jack	42
3	Joe	32
4	Dent	23
5	Bruce	40



```
SELECT *  
FROM Teachers  
WHERE age > ALL (  
  SELECT age  
  FROM Students  
);
```

id	name	age
2	Megan	43

AND Operator

- The SQL AND operator selects data if all conditions are TRUE

Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE



```
SELECT first_name, last_name  
FROM Customers  
WHERE country = 'USA' AND last_name = 'Doe';
```



first_name	last_name
John	Doe

OR Operator

- The SQL OR operator selects data if any one condition is TRUE.

Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

SELECT first_name, last_name
FROM Customers
WHERE country = 'USA' **OR** last_name = 'Doe';

first_name	last_name
John	Doe
Robert	Luna
Betty	Doe

NOT Operator

- The SQL NOT operator selects data if the given condition is FALSE.

Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

SELECT first_name, last_name
FROM Customers
WHERE NOT country = 'USA';

first_name	last_name
David	Robinson
John	Reinhardt
Betty	Doe

BETWEEN Operator

- In SQL, the BETWEEN operator with the WHERE clause selects values within a given range.

Table: Orders

order_id	item	amount	customer_id
1	Keyboard	400	4
2	Mouse	300	4
3	Monitor	12000	3
4	Keyboard	400	1
5	Mousepad	250	2



```
SELECT item, amount  
FROM Orders  
WHERE amount BETWEEN 300 AND 500;
```



item	amount
Keyboard	400
Mouse	300
Keyboard	400

IN Operators

- We use the IN operator with the WHERE clause to match values in a list.
- *SELECT column1, column2 FROM table WHERE column IN (value1, value2, ...);*

Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

SELECT first_name, last_name
FROM Customers
WHERE country **NOT IN** ('UK', 'UAE');

first_name	country
John	USA
Robert	USA

LIKE Operator

- We use the SQL LIKE operator with the WHERE clause to get a result set that matches the given string pattern.
- Syntax
- *SELECT column1, column2 FROM table_name WHERE columnN LIKE pattern;*
- There are two wildcards often used in conjunction with the LIKE operator:
- The **percent sign %** represents zero, one, or multiple characters
- The **underscore sign _** represents one, single character

Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE


SELECT *
FROM Customers
WHERE country **LIKE** 'UK';

customer_id	first_name	last_name	age	country
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK

- **SQL LIKE With the % Wildcard**
- The SQL LIKE query is often used with the % wildcard to match a pattern of a string.

Table: Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE



```
SELECT *  
FROM Customers  
WHERE last_name LIKE 'R%';
```



customer_id	first_name	last_name	age	country
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK

Starts With

- To return records that starts with a specific letter or phrase, add the % at the end of the letter or phrase.
- Example1
- Return all customers that starts with 'La':
- *SELECT * FROM Customers WHERE CustomerName LIKE 'La%';*
- Example2
- Return all customers that starts with 'a' or starts with 'b':
- *SELECT * FROM Customers WHERE CustomerName LIKE 'a%' OR CustomerName LIKE 'b%';*

Ends With

- To return records that ends with a specific letter or phrase, add the % at the beginning of the letter or phrase.
- Example
- Return all customers that ends with 'a':
- *SELECT * FROM Customers WHERE CustomerName LIKE '%a';*

Contains

- To return records that contains a specific letter or phrase, add the % both before and after the letter or phrase.
- Example
- Return all customers that contains the phrase 'or'
- *SELECT * FROM Customers WHERE CustomerName LIKE '%or%';*

Example

- Return all customers that starts with "b" and ends with "s":
- *SELECT * FROM Customers WHERE CustomerName LIKE 'b%s';*

Combine Wildcards

- Any wildcard, like % and _ , can be used in combination with other wildcards.
- Example
- Return all customers that starts with "a" and are at least 3 characters in length:
- *SELECT * FROM Customers WHERE CustomerName LIKE 'a__%';*

ORDER BY

- The ORDER BY clause in SQL is a powerful feature used to sort query results in either ascending or descending order based on one or more columns.
- The ORDER BY command sorts the result set in ascending order by default. To sort the records in descending order, use the DESC keyword.
- The DESC command is used to sort the data returned in descending order.

Syntax:

- *SELECT column1, column2, ...*
FROM table_name
ORDER BY column1, column2, ... ASC/DESC;

Example:

1.Sort the products by price:

```
SELECT * FROM Products  
ORDER BY Price;
```

2.Sort the products from highest to lowest price:

```
SELECT * FROM Products  
ORDER BY Price DESC;
```

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

Order Alphabetically

- For string values the ORDER BY keyword will order alphabetically:

Example

1. Sort the products alphabetically by ProductName:

```
SELECT * FROM Products  
ORDER BY ProductName;
```

2. Sort the products by ProductName in reverse order:

```
SELECT * FROM Products  
ORDER BY ProductName DESC;
```

roll_no	age	name	address	phone
1	18	Shubham Thakur	123 Main St, Mumbai	9876543210
2	18	Mohit Thakur	321 Main St, Mumbai	9876543201
3	19	Abhishek	567 New Way, Mumbai	9876543219
4	19	Aman Chopra	456 Park Ave, Delhi	9876543211
5	20	Naveen Tulasi	789 Broadway, Ahmedabad	9876543212
6	21	Aditya arpan	246 5th Ave, Kolkata	9876543213
7	22	Nishant Jain	369 3rd St, Bengaluru	9876543214

roll_no	age	name	address	phone
7	22	Nishant Jain	369 3rd St, Bengaluru	9876543214
6	21	Aditya arpan	246 5th Ave, Kolkata	9876543213
5	20	Naveen Tulasi	789 Broadway, Ahmedabad	9876543212
4	19	Aman Chopra	456 Park Ave, Delhi	9876543211
3	19	Abhishek	567 New Way, Mumbai	9876543219
2	18	Mohit Thakur	321 Main St, Mumbai	9876543201
1	18	Shubham Thakur	123 Main St, Mumbai	9876543210

Query:

SELECT * FROM students ORDER BY ROLL_NO DESC;

- `SELECT * FROM students ORDER BY age DESC , name ASC;`
- Output:

roll_no	age	name	address	phone
7	22	Nishant Jain	369 3rd St, Bengaluru	9876543214
6	21	Aditya arpan	246 5th Ave, Kolkata	9876543213
5	20	Naveen Tulasi	789 Broadway, Ahmedabad	9876543212
3	19	Abhishek	567 New Way, Mumbai	9876543219
4	19	Aman Chopra	456 Park Ave, Delhi	9876543211
2	18	Mohit Thakur	321 Main St, Mumbai	9876543201
1	18	Shubham Thakur	123 Main St, Mumbai	9876543210

In the above output, we can see that first the result is sorted in descending order according to Age. There are multiple rows of having the same Age. Now, sorting further this result-set according to name will sort the rows with the same Age according to name in ascending order.