SQL Operators

varshachoudhary

Pre-requisites: What is SQL?

Structured Query Language is a computer language that we use to interact with a relational database. In this article we will see all types of SQL operators.

In simple operator can be defined as an entity used to perform operations in a table.

Operators are the foundation of any programming language. We can define operators as symbols that help us to perform specific mathematical and logical computations on operands. In other words, we can say that an operator operates the operands. SQL operators have three different categories.

Types of SQL Operators

- Arithmetic operator
- Comparison operator
- Logical operator

Arithmetic Operators

We can use various arithmetic operators on the data stored in the tables. Arithmetic Operators are:

Operator	Description
+	The addition is used to perform an addition operation on the data values.

Operator	Description
_	This operator is used for the subtraction of the data values.
/	This operator works with the 'ALL' keyword and it calculates division operations.
*	This operator is used for multiplying data values.
%	Modulus is used to get the remainder when data is divided by another.

Example Query:

```
SELECT * FROM employee WHERE emp_city NOT LIKE 'A%';
```

Output:

Results Messages

	emp_id	emp_name	emp_city	emp_country
1	101	Utkarsh Tripathi	Varanasi	India
2	102	Abhinav Singh	Varanasi	India
3	103	Utkarsh Raghuvanshi	Varanasi	India
4	106	Ashutosh Kumar	Patna	India

Comparison Operators

Another important operator in SQL is a comparison operator, which is used to compare one expression's value to other expressions. SQL supports different types of comparison operator, which is described below:

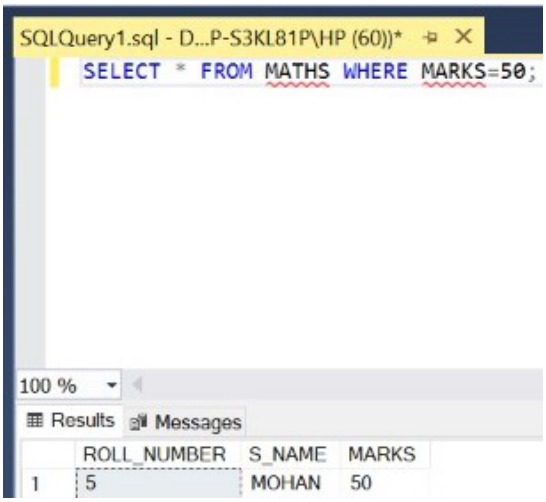
Operator	Description
=	Equal to.
>	Greater than.
<	Less than.
>=	Greater than equal to.
<=	Less than equal to.

Operator	Description
<>	Not equal to.

Example Query:

```
SELECT * FROM MATHS WHERE MARKS=50;
```

Output:



Logical Operators

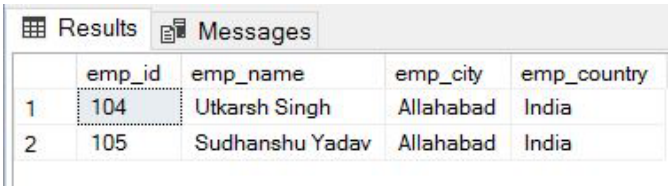
The Logical operators are those that are true or false. They return true or false values to combine one or more true or false values.

Operator	Description
AND	Logical AND compares two Booleans as expressions and returns true when both expressions are true.
OR	Logical OR compares two Booleans as expressions and returns true when one of the expressions is true.
NOT	Not takes a single Boolean as an argument and change its value from false to true or from true to false.

Example Query:

```
SELECT * FROM employee WHERE emp_city =
'Allahabad' AND emp_country = 'India';
```

Output:



Results | Messages

	emp_id	emp_name	emp_city	emp_country
1	104	Utkarsh Singh	Allahabad	India
2	105	Sudhanshu Yadav	Allahabad	India

Special Operators

Operators	Description
ALL	ALL is used to select all records of a SELECT STATEMENT. It compares a value to every value in a list of results from a query. The ALL must be preceded by the comparison operators and evaluated to TRUE if the query returns no rows.
ANY	ANY compares a value to each value in a list of results from a query and evaluates to true if the result of an inner query contains at least one row.
BETWEEN	The SQL BETWEEN operator tests an expression against a range. The range consists of a beginning, followed by an AND keyword and an end expression.
IN	The IN operator checks a value within a set of values separated by commas and retrieves the rows from the table that match.
EXISTS	The EXISTS checks the existence of a result of a subquery. The EXISTS subquery tests whether a subquery fetches at least one row. When no data is returned then this operator returns 'FALSE'.
SOME	SOME operator evaluates the condition between the outer and inner tables and evaluates to true if the final result returns any one row. If not, then it evaluates to false.
UNIQUE	The UNIQUE operator searches every unique row of a specified table.

Example Query:

```
SELECT * FROM employee WHERE emp_id BETWEEN 101 AND 104;
```

Output:

Results | Messages

	emp_id	emp_name	emp_city	emp_country
1	101	Utkarsh Tripathi	Varanasi	India
2	102	Abhinav Singh	Varanasi	India
3	103	Utkarsh Raghuvanshi	Varanasi	India
4	104	Utkarsh Singh	Allahabad	India

Last Updated : 06 Apr, 2023

5

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL AND and OR operators
4. SQL | Wildcard operators
5. SQL | Arithmetic Operators
6. SQL - Logical Operators
7. Comparison Operators in SQL
8. SQL | Procedures in PL/SQL
9. SQL | Difference between functions and stored procedures in PL/SQL
10. SQL SERVER – Input and Output Parameter For Dynamic SQL

Previous Next

Article Contributed By :

varshachoudhary
varshachoudhary

Vote for difficulty

Current difficulty : Basic

SQL | Arithmetic Operators

classyallrounder

Prerequisite: Basic Select statement, Insert into clause, Sql Create Clause, 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

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.

References: Oracle Docs

Last Updated : 21 Mar, 2018

30

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL
4. Spatial Operators, Dynamic Spatial Operators and Spatial Queries in DBMS
5. SQL AND and OR operators
6. SQL | Wildcard operators

Comparison Operators in SQL

abhisri459

In SQL, there are six comparison operators available which help us run queries to perform various operations. We will use the WHERE command along with the conditional operator to achieve this in SQL. For this article, we will be using the Microsoft SQL Server as our database.

Syntax:

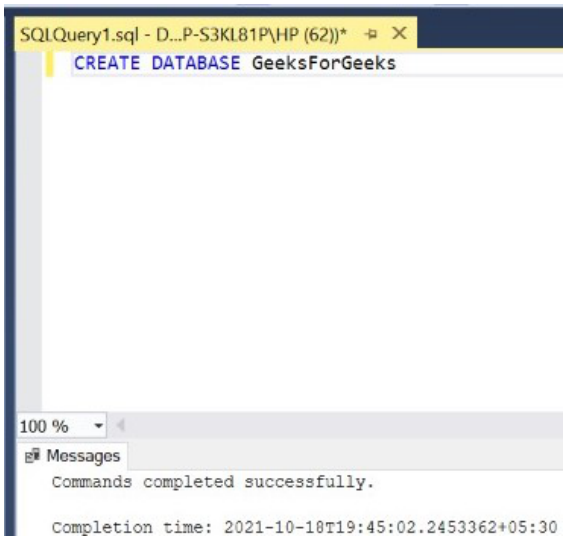
```
SELECT * FROM TABLE_NAME WHERE
ATTRIBUTE CONDITION_OPERATOR GIVEN_VALUE;
```

Step 1: Create a Database. For this use the below command to create a database named GeeksForGeeks.

Query:


```
CREATE DATABASE GeeksForGeeks
```

Output:

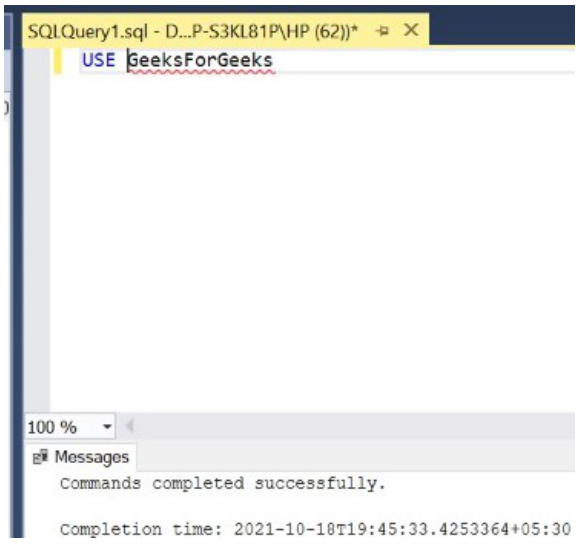


Step 2: Use the GeeksForGeeks database. For this use the below command.

Query:

```
USE GeeksForGeeks
```

Output:

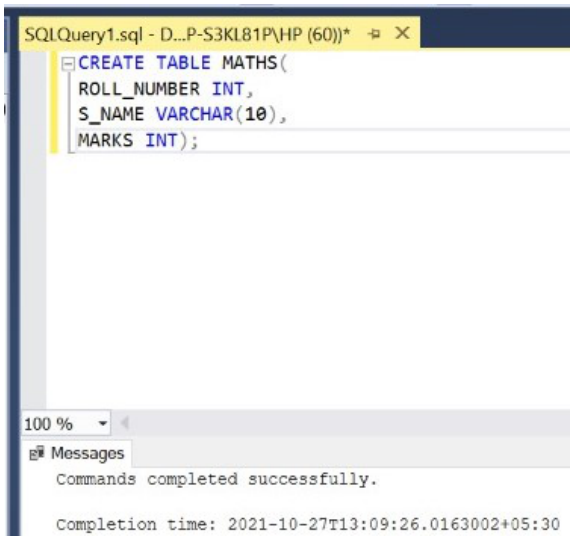


Step 3: Create a table MATHS inside the database GeeksForGeeks. This table has 3 columns namely ROLL_NUMBER, S_NAME and MARKS containing roll number, student name, and marks obtained in math's subject by various students.

Query:

```
CREATE TABLE MATHS(
ROLL_NUMBER INT,
S_NAME VARCHAR(10),
MARKS INT);
```

Output:

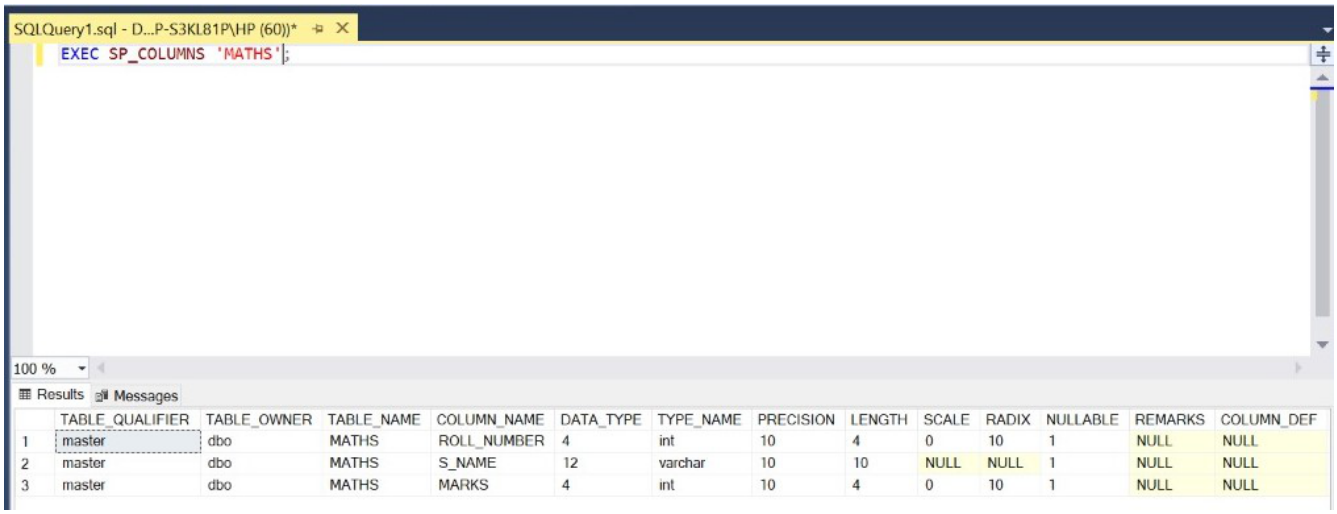


Step 4: Display the structure of the MATHS table.

Query:

```
EXEC SP_COLUMNS 'MATHS';
```

Output:



	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	PRECISION	LENGTH	SCALE	RADIX	NULLABLE	REMARKS	COLUMN_DEF
1	master	dbo	MATHS	ROLL_NUMBER	4	int	10	4	0	10	1	NULL	NULL
2	master	dbo	MATHS	S_NAME	12	varchar	10	10	NULL	NULL	1	NULL	NULL
3	master	dbo	MATHS	MARKS	4	int	10	4	0	10	1	NULL	NULL

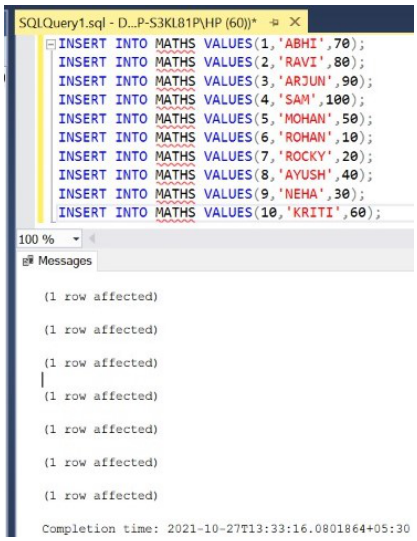
Step 5: Insert 10 rows into the MATHS table.

Query:

```
INSERT INTO MATHS VALUES(1,'ABHI',70);
INSERT INTO MATHS VALUES(2,'RAVI',80);
INSERT INTO MATHS VALUES(3,'ARJUN',90);
INSERT INTO MATHS VALUES(4,'SAM',100);
INSERT INTO MATHS VALUES(5,'MOHAN',50);
INSERT INTO MATHS VALUES(6,'ROHAN',10);
INSERT INTO MATHS VALUES(7,'ROCKY',20);
INSERT INTO MATHS VALUES(8,'AYUSH',40);
```

```
INSERT INTO MATHS VALUES(9,'NEHA',30);
INSERT INTO MATHS VALUES(10,'KRITI',60);
```

Output:

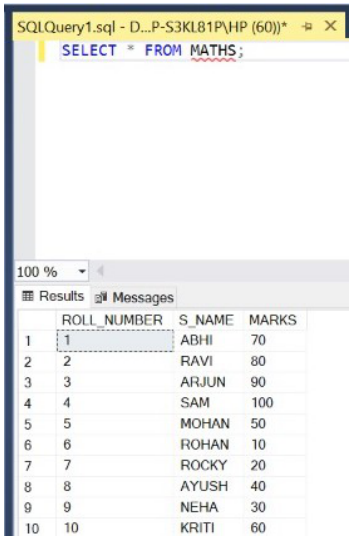


Step 6: Display all the rows of the MATHS table.

Query:

```
SELECT * FROM MATHS;
```

Output:



	ROLL_NUMBER	S_NAME	MARKS
1	1	ABHI	70
2	2	RAVI	80
3	3	ARJUN	90
4	4	SAM	100
5	5	MOHAN	50
6	6	ROHAN	10
7	7	ROCKY	20
8	8	AYUSH	40
9	9	NEHA	30
10	10	KRITI	60

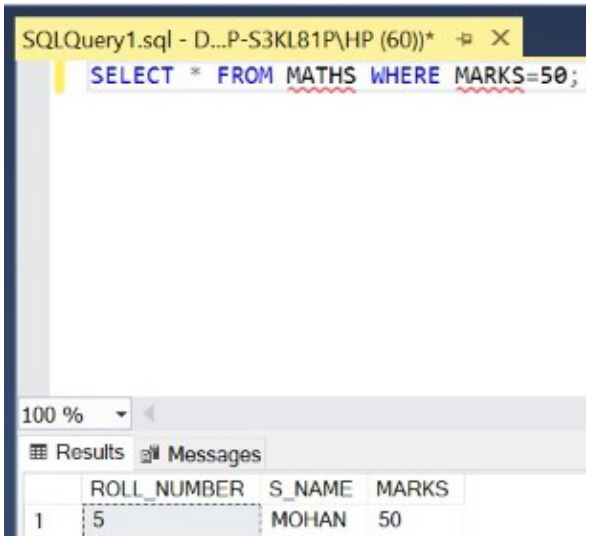
Demonstration of various Comparison Operators in SQL:

- **Equal to (=) Operator**: It returns the rows/tuples which have the value of the attribute equal to the given value.

Query:

```
SELECT * FROM MATHS WHERE MARKS=50;
```

Output:

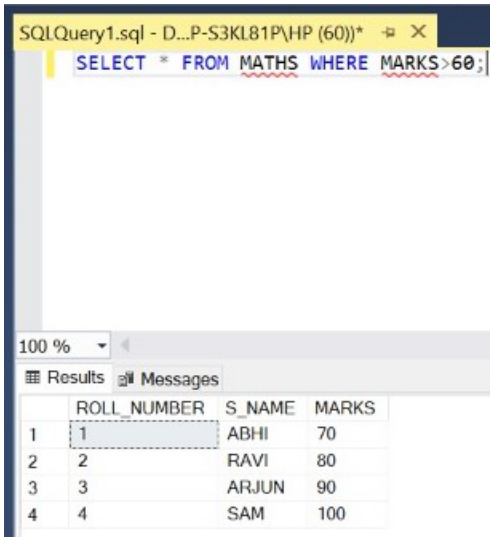


- **Greater than (>) Operator:** It returns the rows/tuples which have the value of the attribute greater than the given value.

Query:

```
SELECT * FROM MATHS WHERE MARKS>60;
```

Output:

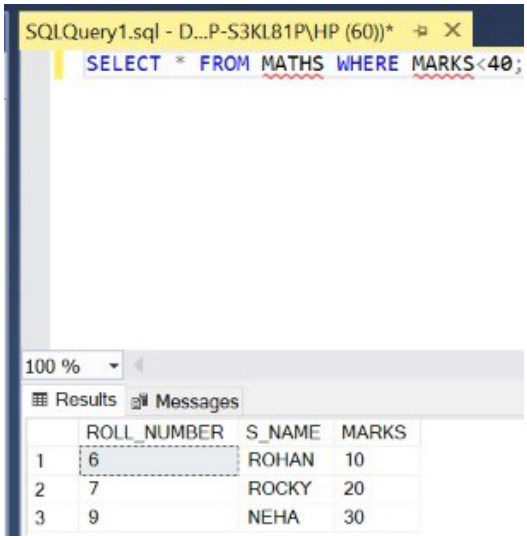


- **Less than (<) Operator:** It returns the rows/tuples which have the value of the attribute lesser than the given value.

Query:

```
SELECT * FROM MATHS WHERE MARKS<40;
```

Output:

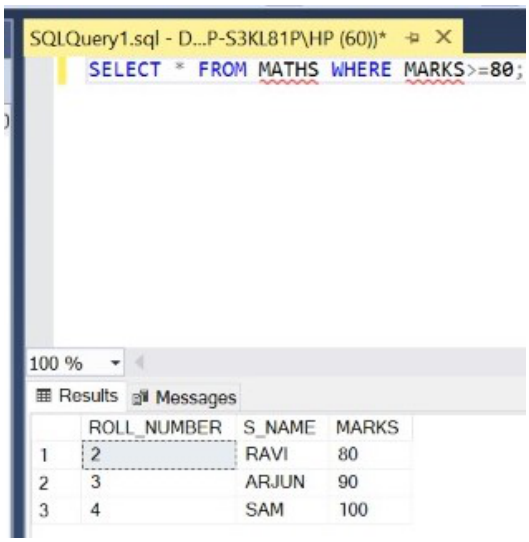


- **Greater than or equal to (>=) Operator:** It returns the rows/tuples which have the value of the attribute greater or equal to the given value.

Query:

```
SELECT * FROM MATHS WHERE MARKS>=80;
```

Output:

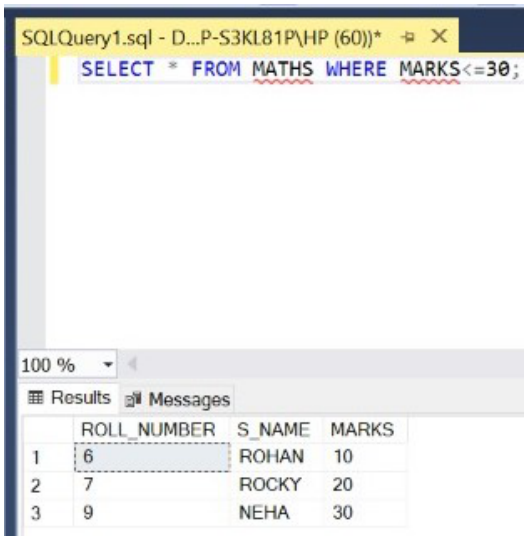


- **Less than or equal to (<=) Operator**: It returns the rows/tuples which have the value of the attribute lesser or equal to the given value.

Query:

```
SELECT * FROM MATHS WHERE MARKS<=30;
```

Output:

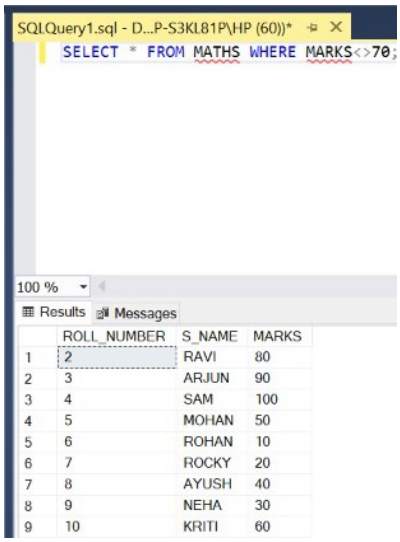


- **Not equal to (<>) Operator:** It returns the rows/tuples which have the value of the attribute not equal to the given value.

Query:

```
SELECT * FROM MATHS WHERE MARKS<>70;
```

Output:



Last Updated : 14 Nov, 2021

2

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL AND and OR operators
4. SQL | Wildcard operators

SQL AND and OR operators

In SQL, the AND & OR operators are used for filtering the data and getting precise results based on conditions. The SQL **AND** & **OR** operators are also used to combine multiple conditions. These two operators can be combined to test for multiple conditions in a SELECT, INSERT, UPDATE, or DELETE statement.

When combining these conditions, it is important to use parentheses so that the database knows what order to evaluate each condition.

- The AND and OR operators are used with the WHERE clause.
- These two operators are called conjunctive operators.

AND Operator:

This operator displays only those records where both the conditions condition1 and condition2 evaluates to True.

Syntax:


```
SELECT * FROM table_name WHERE condition1 AND condition2 and ...conditionN;

table_name: name of the table
condition1,2,..N : first condition, second condition and so on
```

OR Operator:

This operator displays the records where either one of the conditions condition1 and condition2 evaluates to True. That is, either condition1 is True or condition2 is True.

Syntax:

```
SELECT * FROM table_name WHERE condition1 OR condition2 OR... conditionN;
```

table_name: name of the table
condition1,2,..N : first condition, second condition and so on

Now, we consider a table database to demonstrate AND & OR operators with multiple cases:

Student

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	xxxxxxxxxx	18
2	RAMESH	GURGAON	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
4	SURESH	Delhi	xxxxxxxxxx	18
3	SUJIT	ROHTAK	xxxxxxxxxx	20
2	RAMESH	GURGAON	xxxxxxxxxx	18

If suppose we want to fetch all the records from the Student table where Age is 18 and ADDRESS is Delhi. then the query will be:

Query:

```
SELECT * FROM Student WHERE Age = 18 AND ADDRESS = 'Delhi';
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18
4	SURESH	Delhi	XXXXXXXXXX	18

Take another example, to fetch all the records from the Student table where NAME is Ram and Age is 18.

Query:

```
SELECT * FROM Student WHERE Age = 18 AND NAME = 'Ram';
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18

To fetch all the records from the Student table where NAME is Ram or NAME is SUJIT.

Query:

```
SELECT * FROM Student WHERE NAME = 'Ram' OR NAME = 'SUJIT';
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18
3	SUJIT	ROHTAK	XXXXXXXXXX	20
3	SUJIT	ROHTAK	XXXXXXXXXX	20

To fetch all the records from the Student table where NAME is Ram or Age is 20.

Query:

```
SELECT * FROM Student WHERE NAME = 'Ram' OR Age = 20;
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18

ROLL_NO	NAME	ADDRESS	PHONE	Age
3	SUJIT	ROHTAK	XXXXXXXXXX	20
3	SUJIT	ROHTAK	XXXXXXXXXX	20

Combining AND and OR:

We can combine AND and OR operators in the below manner to write complex queries.

Syntax:

```
SELECT * FROM table_name WHERE condition1 AND (condition2 OR condition3);
```

Take an example to fetch all the records from the Student table where Age is 18 NAME is Ram or RAMESH.

Query:

```
SELECT * FROM Student WHERE Age = 18 AND (NAME = 'Ram' OR NAME = 'RAMESH');
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	Ram	Delhi	XXXXXXXXXX	18
2	RAMESH	GURGAON	XXXXXXXXXX	18

Last Updated : 04 Oct, 2021

36

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL

SQL | ALL and ANY

ALL & ANY are logical operators in SQL. They return boolean value as a result.

ALL

ALL operator is used to select all tuples of SELECT STATEMENT. It is also used to compare a value to every value in another value set or result from a subquery.

- The ALL operator returns TRUE if all of the subqueries values meet the condition. The ALL must be preceded by comparison operators and evaluates true if all of the subqueries values meet the condition.
- ALL is used with SELECT, WHERE, HAVING statement.

ALL with SELECT Statement:


```
Syntax:
SELECT ALL field_name
FROM table_name
WHERE condition(s);
```

ALL with WHERE or HAVING Statement:

```
Syntax:
SELECT column_name(s)
FROM table_name
WHERE column_name comparison_operator ALL
(SELECT column_name
```

```
FROM table_name
WHERE condition(s));
```

Example: Consider the following Products Table and OrderDetails Table,**Products Table**

ProductID	ProductName	SupplierID	CategoryID	Price
1	Chais	1	1	18
2	Chang	1	1	19
3	Aniseed Syrup	1	2	10
4	Chef Anton's Cajun Seasoning	2	2	22
5	Chef Anton's Gumbo Mix	2	2	21
6	Boysenberry Spread	3	2	25
7	Organic Dried Pears	3	7	30
8	Northwoods Cranberry Sauce	3	2	40
9	Mishi Kobe Niku	4	6	97

OrderDetails Table

OrderDetailsID	OrderID	ProductID	Quantity
1	10248	1	12
2	10248	2	10
3	10248	3	15
4	10249	1	8
5	10249	4	4
6	10249	5	6
7	10250	3	5
8	10250	4	18
9	10251	5	2
10	10251	6	8
11	10252	7	9
12	10252	8	9
13	10250	9	20
14	10249	9	4

Queries

- **Find the name of the all the product.**

```
SELECT ALL ProductName
FROM Products
WHERE TRUE;
```

ProductName
Chais
Chang
Aniseed Syrup
Chef Anton's Cajun Seasoning
Chef Anton's Gumbo Mix
Boysenberry Spread
Organic Dried Pears
Northwoods Cranberry Sauce
Mishi Kobe Niku

- Output:
- **Find the name of the product if all the records in the OrderDetails has Quantity either equal to 6 or 2.**

```
SELECT ProductName
FROM Products
WHERE ProductID = ALL (SELECT ProductId
                       FROM OrderDetails
                       WHERE Quantity = 6 OR Quantity = 2);
```

- Output:

ProductName
Chef Anton's Gumbo Mix

- **Find the OrderID whose maximum Quantity among all product of that OrderID is greater than average quantity of all OrderID.**

```
SELECT OrderID
FROM OrderDetails
GROUP BY OrderID
HAVING max(Quantity) > ALL (SELECT avg(Quantity)
                            FROM OrderDetails
                            GROUP BY OrderID);
```

- Output:

OrderID
10248
10250

ANY

ANY compares a value to each value in a list or results from a query and evaluates to true if the result of an inner query contains at least one row.

- ANY return true if any of the subqueries values meet the condition.
- ANY must be preceded by comparison operators. **Syntax:**

```
SELECT column_name(s)
FROM table_name
WHERE column_name comparison_operator ANY
(SELECT column_name
FROM table_name
WHERE condition(s));
```

Queries

- Find the Distinct CategoryID of the products which have any record in OrderDetails Table.

```
SELECT DISTINCT CategoryID
FROM Products
WHERE ProductID = ANY (SELECT ProductID
                       FROM OrderDetails);
```

- Output:

CategoryID
1
2
7
6

- Finds any records in the OrderDetails table that Quantity = 9.

```
SELECT ProductName
FROM Products
WHERE ProductID = ANY (SELECT ProductID
                       FROM OrderDetails
                       WHERE Quantity = 9);
```

-

ProductName
Organic Dried Pears
Northwoods Cranberry Sauce

This article is contributed by **Anuj Chauhan**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Last Updated : 28 Nov, 2022

44

Similar Reads

SQL | Wildcard operators

Prerequisite: SQL | WHERE Clause

In the above-mentioned article WHERE Clause is discussed in which the LIKE operator is also explained, where you must have encountered the word wildcards now let's get deeper into Wildcards. Wildcard operators are used with the LIKE operator, there are four basic operators:

Operator Table

Operator	Description
%	It is used in substitute of zero or more characters.
_	It is used as a substitute for one character.
–	It is used to substitute a range of characters.
[range_of_characters]	It is used to fetch a matching set or range of characters specified inside the brackets.

Syntax:

SELECT column1,column2 FROM table_name

WHERE column LIKE wildcard_operator;

column1,column2: fields in the table

table_name: name of table

column: name of field used for filtering data

CREATE Table:

```
CREATE TABLE Customer(
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(50),
    LastName VARCHAR(50),
    Country VARCHAR(50),
    Age int(2),
  Phone int(10)
);
-- Insert some sample data into the Customers table
INSERT INTO Customer (CustomerID, CustomerName, LastName, Country, Age, Phone)
VALUES (1, 'Shubham', 'Thakur', 'India','23','xxxxxxxxxx'),
       (2, 'Aman ', 'Chopra', 'Australia','21','xxxxxxxxxx'),
       (3, 'Naveen', 'Tulasi', 'Sri lanka','24','xxxxxxxxxx'),
       (4, 'Aditya', 'Arpan', 'Austria','21','xxxxxxxxxx'),
       (5, 'Nishant. Salchichas S.A.', 'Jain', 'Spain','22','xxxxxxxxxx');
       Select * from Customer;
```

Output:

Customer

CustomerID	CustomerName	LastName	Country	Age	Phone
1	Shubham	Thakur	India	23	xxxxxxxxxx
2	Aman	Chopra	Australia	21	xxxxxxxxxx
3	Naveen	Tulasi	Sri lanka	24	xxxxxxxxxx
4	Aditya	Arpan	Austria	21	xxxxxxxxxx
5	Nishant. Salchichas S.A.	Jain	Spain	22	xxxxxxxxxx

1. To fetch records from the Customer table with NAME starting with the letter ‘A’.

Query:

```
SELECT * FROM Customer WHERE CustomerName LIKE 'A%';
```

Output:

CustomerID	CustomerName	LastName	Country	Age	Phone
2	Aman	Chopra	Australia	21	xxxxxxxxxx
4	Aditya	Arpan	Austria	21	xxxxxxxxxx

2. To fetch records from the Customer table with NAME ending with the letter ‘A’.

Query:

```
SELECT * FROM Customer WHERE CustomerName LIKE '%A';
```

Output:

CustomerID	CustomerName	LastName	Country	Age	Phone
4	Aditya	Arpan	Austria	21	xxxxxxxxxx

3. To fetch records from the Customer table with NAME with the letter ‘A’ at any position.

Query:

```
SELECT * FROM Customer WHERE CustomerName LIKE '%A%';
```

Output:

CustomerID	CustomerName	LastName	Country	Age	Phone
1	Shubham	Thakur	India	23	xxxxxxxxxx
2	Aman	Chopra	Australia	21	xxxxxxxxxx
3	Naveen	Tulasi	Sri lanka	24	xxxxxxxxxx
4	Aditya	Arpan	Austria	21	xxxxxxxxxx
5	Nishant. Salchichas S.A.	Jain	Spain	22	xxxxxxxxxx

4. To fetch records from the Student table with NAME ending any letter but starting from ‘Nav’.

Query:

```
SELECT * FROM Customer WHERE CustomerName LIKE 'Nav___';
```

Output:

CustomerID	CustomerName	LastName	Country	Age	Phone
3	Naveen	Tulasi	Sri lanka	24	234565663

5. To fetch records from the Student table with LastName containing letters ‘a, ‘b’, or ‘c’.

Query:

```
SELECT * FROM Student WHERE LastName REGEXP '[A-C]';
```

Output:

```
CustomerID      CustomerName    LastName        Country Age     Phone
1       Shubham Thakur  India   23      862357843
2       Aman    Chopra  Australia       21      436577545
3       Naveen  Tulasi  Sri lanka       24      234565663
4       Aditya  Arpan   Austria 21      234565676
5       Nishant. Salchichas S.A.        Jain    Spain   22      234567346
```

6. To fetch records from the Student table with LastName not containing letters ‘y’, or ‘z’.

Query:

```
SELECT * FROM Students WHERE LastName NOT LIKE '%[y-z]%';
```

Output:

```
CustomerID      CustomerName    LastName        Country Age     Phone
1       Shubham Thakur  India   23      862357843
2       Aman    Chopra  Australia       21      436577545
3       Naveen  Tulasi  Sri lanka       24      234565663
4       Aditya  Arpan   Austria 21      234565676
5       Nishant. Salchichas S.A.        Jain    Spain   22      234567346
```

7. To fetch records from the Student table with the PHONE field having an ‘8’ in the 1st position and a ‘3’ in the 3rd position.

Query:

```
SELECT * FROM Student WHERE PHONE LIKE '8__3%';
```

Output:

```
CustomerID      CustomerName    LastName        Country Age     Phone
1       Shubham Thakur  India   23      862357843
```

8. To fetch records from the Student table with Country containing a total of 7 characters.

Query:

```
SELECT * FROM Students WHERE Country LIKE '_______';
```

Output:

```
CustomerID      CustomerName    LastName        Country Age     Phone
4       Aditya  Arpan   Austria 21      234565676
```

9. To fetch records from the Student table with the LastNamecontaining ‘ra’ at any position, and the result set should not contain duplicate data.

Query:

```
SELECT DISTINCT * FROM Students WHERE Country LIKE '%ra%';
```

Output:

```
CustomerID      CustomerName    LastName        Country Age     Phone
2       Aman    Chopra  Australia       21      436577545
```

Frequently Asked Question

Question: What is a wildcard operator in SQL?

Ans: The LIKE operator makes use of wildcard characters. The LIKE operator is used in a WHERE clause to look for a specific pattern in a column.

This article is contributed by **Pratik Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please comment if you find anything incorrect or if you want to share more information about the topic discussed above.

Last Updated : 06 Apr, 2023

42

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL

SQL | Concatenation Operator

Prerequisite: Basic Select statement, Insert into clause, SQL Create Clause, SQL Aliases

|| or concatenation operator is use to **link columns** or **character strings**. We can also use a **literal**. A literal is a **character**, **number** or **date** that is included in the SELECT statement.

Let's demonstrate it through an example:

Syntax:


```
SELECT id, first_name, last_name, first_name || last_name,
              salary, first_name || salary FROM myTable
```

```
Output (Third and Fifth Columns show  values concatenated by operator ||)
id  first_name  last_name    first_name||last_name  salary   first_name||salary
1   Rajat        Rawat          RajatRawat          10000    Rajat10000
2   Geeks        ForGeeks      GeeksForGeeks        20000    Geeks20000
3   Shane        Watson         ShaneWatson         50000    Shane50000
4   Kedar        Jadhav         KedarJadhav         90000     Kedar90000
```

Note: Here above we have used **||** which is known as **Concatenation operator** which is used to link 2 or as **many** columns as you want in your select query and it is **independent of the datatype** of column. Here above we have linked 2 columns i.e, **first_name+last_name** as well as **first_name+salary**.

We can also use **literals** in the **concatenation** operator. Let's see:

Example 1: Using character literal
Syntax:

```
SELECT id, first_name, last_name, salary,
      first_name||' has salary '||salary as "new"  FROM myTable

Output : (Concatenating three values and giving a name 'new')
id  first_name  last_name  salary           new
1   Rajat        Rawat     10000      Rajat has salary 10000
2   Geeks        ForGeeks  20000      Geeks has salary 20000
3   Shane        Watson    50000      Shane has salary 50000
4   Kedar        Jadhav    90000      Kedar has salary 90000
```

Note: Here above we have used **has salary** as a character literal in our select statement. Similarly we can use number literal or date literal according to our requirement.

Example 2: Using character as well as number literal
Syntax:

```
SELECT id, first_name, last_name, salary, first_name||100||'
                        has id '||id AS "new" FROM myTable

Output (Making readable output by concatenating a string
with values)

id  first_name   last_name     salary     new
1    Rajat       Rawat         10000    Rajat100 has id 1
2    Geeks       ForGeeks      20000    Geeks100 has id 2
3    Shane       Watson        50000    Shane100 has id 3
4    Kedar       Jadhav        90000    Kedar100 has id 4
```

Here above we have used **has salary** as a character literal as well as **100** as number literal in our select statement.

References:
1) About Concatenation operator: Oracle Docs
2) Performing SQL Queries Online: Oracle Live SQL

Note: For performing SQL Queries online you must have account on Oracle, if you don't have then you can make by opening above link.

Last Updated : 21 Mar, 2018

17

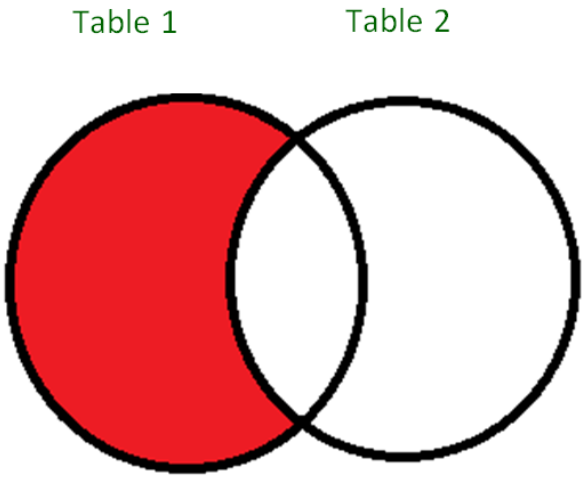
Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)

SQL | MINUS Operator

The Minus Operator in SQL is used with two SELECT statements. The MINUS operator is used to subtract the result set obtained by first SELECT query from the result set obtained by second SELECT query. In simple words, we can say that MINUS operator will return only those rows which are unique in only first SELECT query and not those rows which are common to both first and second SELECT queries.

Pictorial Representation:



As you can see is in the above diagram, the MINUS operator will return only those rows which are present in the result set from Table1 and not present in the result set of Table2.

Basic Syntax:

```
SELECT column1 , column2 , ... columnN
FROM table_name
WHERE condition
MINUS
SELECT column1 , column2 , ... columnN
FROM table_name
WHERE condition;
```

columnN: column1, column2.. are the name of columns of the table.

Important Points:

- The WHERE clause is optional in the above query.
- The number of columns in both SELECT statements must be same.
- The data type of corresponding columns of both SELECT statement must be same.

Sample Tables:

Table1

Table 1

Name	Address	Age	Grade
Harsh	Delhi	20	A
Gaurav	Jaipur	21	B
Pratik	Mumbai	21	A
Dhanraj	Kolkata	22	B

Table 2

Name	Age	Phone	Grade
Akash	20	xxxxxxxxxx	A
Dhiraj	21	xxxxxxxxxx	B
Vaibhav	21	xxxxxxxxxx	A
Dhanraj	22	xxxxxxxxxx	B

Queries:

```
SELECT NAME, AGE , GRADE
FROM Table1
MINUS
SELECT NAME, AGE, GRADE
FROM Table2
```

Output:
The above query will return only those rows which are unique in 'Table1'. We can clearly see that values in the fields NAME, AGE and GRADE for the last row in both tables are same. Therefore, the output will be the first three rows from Table1. The obtained output is shown below:

Name	Age	Grade
Harsh	20	A
Gaurav	21	B
Pratik	21	A

Note: The MINUS operator is not supported with all databases. It is supported by Oracle database but not SQL server or PostgreSQL.

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Last Updated : 15 Jul, 2022

20

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL | BETWEEN & IN Operator
4. SQL | NOT Operator
5. SQL | Concatenation Operator
6. SQL | Alternative Quote Operator
7. Difference between = and IN operator in SQL
8. SQL | UNION Operator
9. Inequality Operator in SQL

SQL | DIVISION

Division is typically required when you want to find out entities that are interacting with **all entities** of a set of different type entities.
The division operator is used when we have to evaluate queries which contain the keyword 'all'.

Some instances where division operator is used are:

- Which person has account in all the banks of a particular city?
- Which students have taken all the courses required to graduate?

In all these queries, the description after the keyword 'all' defines a set which contains some elements and the final result contains those units who satisfy these requirements.

Important: Division is not supported by SQL implementations. However, it can be represented using other operations.(like cross join, Except, In)

SQL Implementation of Division

Given two relations(tables): R(x,y) , S(y).
R and S : tables
x and y : column of R
y : column of S

R(x,y) div S(y) means gives all distinct values of x from R that are associated with all values of y in S.
Computation of Division : R(x,y) div S(y)
Steps:

- Find out all possible combinations of S(y) with R(x) by computing R(x) x(cross join) S(y), say r1

- Subtract actual R(x,y) from r1, say r2
- x in r2 are those that are not associated with every value in S(y); therefore R(x)-r2(x) gives us x
  that are associated with all values in S

Queries

1. **Implementation 1:**

```
SELECT * FROM R
WHERE x not in ( SELECT x FROM (
(SELECT x , y FROM (select y from S ) as p cross join
(select distinct x from R) as sp)
EXCEPT
(SELECT x , y FROM R) ) AS r );
```

2. **Implementation 2 :** Using correlated subquery

```
SELECT * FROM R as sx
WHERE NOT EXISTS (
(SELECT p.y FROM S as p )
EXCEPT
(SELECT sp.y FROM  R as sp WHERE sp.x = sx.x ) );
```

Relational algebra

Using steps which is mention above:
All possible combinations
r1 ← πx(R) x S
x values with “incomplete combinations”,
r2x ← πx(r1-R)
and
result ← πx(R)-r2x

R div S = πx(R)- πx((πx(R) x S) – R)

Examples

Supply Schema

sid (integer)	pid (integer)
101	1
102	1
101	3
103	2
102	2
102	3
102	4
102	5

pid (integer)
1
2
3
4
5

Here **sid** means **supplierID** and **pid** means **partsID**.

Tables: suppliers(sid,pid) , parts(pid)

1. Find suppliers that supply all parts.

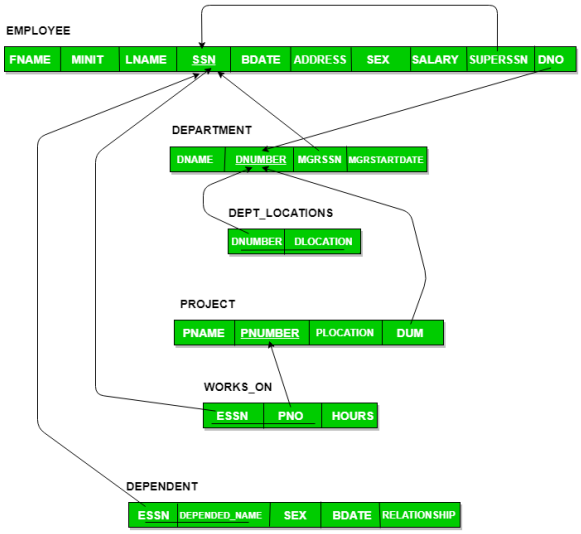
Ans 1 : Using implementation 1

```
SELECT * FROM suppliers
WHERE sid not in ( SELECT sid FROM ( (SELECT sid, pid FROM (select pid from
parts) as p
cross join
(select distinct sid from supplies) as sp)
EXCEPT
(SELECT sid, pid FROM supplies)) AS r );
```

Ans 2: Using implementation 2

```
SELECT * FROM suppliers as s
WHERE NOT EXISTS (( SELECT p.pid FROM parts as p )
EXCEPT
 (SELECT sp.pid FROM supplies sp WHERE sp.sid = s.sid ) );
```

Company schema



2. List employees who work on all projects controlled by dno=4.

Ans 1. Using implementation 1

```
SELECT * FROM employee AS e
WHERE ssn NOT IN (
SELECT essn FROM (
(SELECT essn, pno FROM (select pno from project where dno=4)
as p cross join (select distinct essn from works_on) as w)
EXCEPT (SELECT essn, pno FROM works_on)) AS r );
```

Ans 2. Using implementation 2

```
SELECT * FROM employee AS e
WHERE NOT EXISTS (
 (SELECT pno FROM project WHERE dno = 4)
 EXCEPT
 (SELECT pno FROM works_on WHERE essn = e.ssn) );
```

Important : For division correlated query seems simpler to write but may expensive to execute.

Some more Examples.

1. List supplier who supply all ‘Red’ Parts.(supply schema)
2. Retrieve the names of employees, who work on all the projects that ‘John Smith’ works (company schema)

This article is contributed by Kadam Patel. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

SQL | NOT Operator

NOT Syntax SELECT column1, colomn2, ... FROM table_name WHERE NOT condition; **Demo Database** Below is a selection from the “Customers” table in the Northwind sample database:

Customer ID	Customer Name	City	PostalCode	Country
1	John Wick	New York	1248	USA
2	Around the Horn	London	WA1 1DP	UK
3	Rohan	New Delhi	100084	India

NOT Example The following SQL statement selects all fields from “Customers” where country is not “UK” SELECT * FROM Customers WHERE NOT Country=’UK’;

Customer ID	Customer Name	City	PostalCode	Country
1	John Wick	New York	1248	USA
3	Rohan	New Delhi	100084	India

Combining AND, OR and NOT You can also combine the AND, OR, and NOT operators. Example: 1.) SELECT * FROM Customers WHERE NOT Country=’USA’ AND NOT Country=’UK’;

Customer ID	Customer Name	City	PostalCode	Country

3	Rohan	New Delhi	100084	India

Alternatively you can use <> (Not Operator) to get the desired result :-


```
SELECT * FROM Customer WHERE Country <>'USA';
```

Output :-

	CUSTOMER_ID	CUSTOMER_NAME	CITY	POSTALCODE	COUNTRY
1	2	Around the Horn	London	WA1 1DP	UK
2	3	Rohan	New Delhi	100084	India

Last Updated : 24 Mar, 2023

36

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL | BETWEEN & IN Operator
4. SQL | MINUS Operator
5. SQL | Concatenation Operator
6. SQL | Alternative Quote Operator
7. Difference between = and IN operator in SQL
8. SQL | UNION Operator
9. Inequality Operator in SQL

SQL | BETWEEN & IN Operator

Pre-requisites: SQL Operators

Operators are the foundation of any programming language. We can define operators as symbols that help us to perform specific mathematical and logical computations on operands. In other words, we can say that an operator operates the operands.

In this article, we will see BETWEEN & IN Operator of SQL.

Between Operator

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 the expression is within the range of value1 and value2.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Let's create a database to understand BETWEEN & IN Operator in SQL.

Query:

```
CREATE TABLE Emp(
    EmpID INT PRIMARY KEY,
    Name VARCHAR(50),
    Country VARCHAR(50),
    Age int(2),
  Salary int(10)
);
-- Insert some sample data into the Customers table
INSERT INTO Emp (EmpID, Name,Country, Age, Salary)
VALUES (1, 'Shubham',  'India','23','30000'),
       (2, 'Aman ',  'Australia','21','45000'),
       (3, 'Naveen', 'Sri lanka','24','40000'),
       (4, 'Aditya',  'Austria','21','35000'),
       (5, 'Nishant', 'Spain','22','25000');
       Select * from Emp;
```

Output:

Emp

EmpID	Name	Country	Age	Salary
1	Shubham	India	23	30000
2	Aman	Australia	21	45000
3	Naveen	Sri lanka	24	40000
4	Aditya	Austria	21	35000
5	Nishant	Spain	22	25000

Using BETWEEN with Numeric Values

List all the Employee's Names who is having salary between 30000 and 45000.

Query:

```
SELECT Name
FROM Emp
WHERE Salary
BETWEEN 30000 AND 45000;
```

Output:

Name
Shubham
Aman
Naveen
Aditya

Using BETWEEN with Date Values

Find all the Employees an Age Between 22 to 24.

Query:

```
SELECT Name
FROM Emp
where Age
BETWEEN '22' AND '24';
```

Output:

Name
Shubham
Naveen
Nishant

Using the NOT Operator with BETWEEN

Find all the Employee names whose salary is not in the range of 30000 and 45000.

Query:

```
SELECT Name
FROM Emp
WHERE Salary
NOT BETWEEN 30000 AND 45000;
```

Output:

Name
Nishant

IN Operator

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 for multiple OR conditions in SELECT, INSERT, UPDATE, or DELETE. You can also use NOT IN to exclude the rows in your list. We should note that any kind of duplicate entry will be retained.

Syntax:

SELECT column_name(s)

FROM table_name

WHERE column_name IN (list_of_values);

Find the Fname, and Lname of the Employees who have a Salary equal to 30000, 40000, or 25000.

Query:

```
SELECT Name
FROM Emp
WHERE Salary IN (30000, 40000, 25000);
```

Output:

Name
Shubham
Naveen
Nishant

Find the Fname and Lname of all the Employees who has a Salary not equal to 25000 or 30000.

Query:

```
SELECT Name
FROM Emp
WHERE Salary NOT IN (25000, 30000);
```

Output:

Name
Aman
Naveen
Aditya

This article is contributed by **Anuj Chauhan**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 05 Apr, 2023

31

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL
4. Difference between = and IN operator in SQL
5. SQL | Difference between functions and stored procedures in PL/SQL
6. Difference between T-SQL and PL-SQL
7. Difference between SQL and T-SQL
8. SQL | MINUS Operator
9. SQL | NOT Operator

SQL LIKE

Sometimes we may require tuples from the database which match certain patterns. For example, we want to retrieve all columns where the tuples start with the letter 'y', or start with 'b' and end with 'l', or even more complicated and restrictive string patterns. This is where the **SQL LIKE Clause** comes to the rescue, often coupled with the WHERE Clause in SQL.

In SQL, the LIKE operator is mainly used in the WHERE clause to search for a enumerate pattern in a column.

Two barriers are often used in conjunction with the LIKE :

1. %: Used to match zero or more characters. (Variable Length)
2. _: Used to match exactly one character. (Fixed Length)

SQL Like Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

The following are the rules for pattern matching with the LIKE Clause :

Pattern	Meaning
'a%'	Match strings that start with 'a'
'%a'	Match strings with end with 'a'

Pattern	Meaning
‘a%t’	Match strings that contain the start with ‘a’ and end with ‘t’.
‘%wow%’	Match strings that contain the substring ‘wow’ in them at any position.
‘_wow%’	Match strings that contain the substring ‘wow’ in them at the second position.
‘_a%’	Match strings that contain ‘a’ at the second position.
‘a_ _%’	Match strings that start with ‘a and contain at least 2 more characters.

Example: Say we have a relation, Supplier. We want to test various patterns using the LIKE clause:

Supplier Table

SupplierID	Name	Address
S1	Paragon Suppliers	21-3, Okhla, Delhi
S2	Mango Nation	21, Faridabad, Haryana
S3	Canadian Biz	6/7, Okhla Phase II, Delhi
S4	Caravan Traders	2-A, Pitampura, Delhi
S5	Harish and Sons	Gurgaon, NCR
S6	Om Suppliers	2/1, Faridabad, Haryana

SQL LIKE – Sample Queries and Outputs

Query 1:

```
SELECT SupplierID, Name, Address
FROM Suppliers
WHERE Name LIKE 'Ca%';
```

Output:

S3	Canadian Biz	6/7, Okhla Phase II, Delhi
S4	Caravan Traders	2-A, Pitampura, Delhi

Query 2:

```
SELECT *
FROM Suppliers
WHERE Address LIKE '%Okhla%';
```

Output:

S1	Paragon Suppliers	21-3, Okhla, Delhi
S3	Canadian Biz	6/7, Okhla Phase II, Delhi

Query 3:

```
SELECT SupplierID, Name, Address
FROM Suppliers
WHERE Name LIKE '_ango%';
```

Output:

S2	Mango Nation	21, Faridabad, Haryana

SQL LIKE Application

The LIKE operator is extremely resourceful in situations such as address filtering wherein we know only a segment or a portion of the entire address (such as locality or city) and would like to retrieve results based on that. The wildcards can be resourcefully exploited to yield even better and more filtered tuples based on the requirement.

Important to know about SQL LIKE

One important thing to note about the LIKE operator is that it is **case-insensitive** by default in most database systems. This means that if you search for “apple” using the LIKE operator, it will return results that include “Apple”, “APPLE”, “aPpLe”, and so on.

For making the LIKE operator case-sensitive, you can use the "**BINARY" keyword in MySQL or the "COLLATE" keyword in other database systems**.

For example:

XML

```
SELECT * FROM products WHERE name LIKE BINARY 'apple%'
```

This following query will only return products whose name starts with "apple" and is spelled exactly like that, without capital letters.

This article is contributed by **Anannya Uberoi**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 04 May, 2023

29

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. How to Escape Square Brackets in a LIKE Clause in SQL Server?
4. SQL | Procedures in PL/SQL
5. SQL | Difference between functions and stored procedures in PL/SQL
6. SQL SERVER – Input and Output Parameter For Dynamic SQL
7. Difference between SQL and T-SQL
8. SQL Server | Convert tables in T-SQL into XML
9. SQL SERVER | Bulk insert data from csv file using T-SQL command
10. SQL - SELECT from Multiple Tables with MS SQL Server

Previous Next

SQL | SOME

SQL | ALL and ANY

SOME operator evaluates the condition between the outer and inner tables and evaluates to true if the final result returns **any one** row. If not, then it evaluates to false.

- The SOME and ANY comparison conditions are similar to each other and are completely interchangeable.
- SOME must match at least one row in the subquery and must be preceded by comparison operators.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE expression comparison_operator SOME (subquery)
```

Instructor Table:

Name	Department	Salary
Chandra	Computational Biology	1
Visweswaran	Electronics	1.5
Abraham	Computer Science	1.3
John	Electronics	1.2
Samantha	Computer Science	2
Jyoti	Electronics	1.2
Debarka	Computer Science	2

Ganesh	Computational Biology	0.9

Sample Queries and Outputs:


```
select name
from instructor
where Salary > some(select Salary
from instructor
where dept='Computer Science');
```

Output:

Visweswaran
Samantha
Debarka

Explanation

The instructors with salary > (salary of some instructor in the 'Computer Science' department) get returned. The salaries in the 'Computer Science' department are 1.3, 2 and 2. This implies any instructor with a salary greater than 1.3 can be included in the final result.

Exercise: Try to write same query using ANY clause.

This article is contributed by **Anannya Uberoi**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Last Updated : 21 Mar, 2018

16

Similar Reads

SQL | EXISTS

The EXISTS condition in SQL is used to check whether the result of a correlated nested query is empty (contains no tuples) or not. The result of EXISTS is a boolean value True or False. It can be used in a SELECT, UPDATE, INSERT or DELETE statement.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
  (SELECT column_name(s)
   FROM table_name
   WHERE condition);
```

Examples:
Consider the following two relation “Customers” and “Orders”.

Customers

customer_id	lname	fname	website
401	Singh	Dolly	abc.com
402	Chauhan	Anuj	def.com
403	Kumar	Niteesh	ghi.com
404	Gupta	Shubham	jkl.com
405	Walecha	Divya	abc.com
406	Jain	Sandeep	jkl.com
407	Mehta	Rajiv	abc.com
408	Mehra	Anand	abc.com

Orders

order_id	c_id	order_date
1	407	2017-03-03
2	405	2017-03-05
3	408	2017-01-18
4	404	2017-02-05

Queries

1. **Using EXISTS condition with SELECT statement**

   To fetch the first and last name of the customers who placed atleast one order.

```
SELECT fname, lname
FROM Customers
WHERE EXISTS (SELECT *
              FROM Orders
              WHERE Customers.customer_id = Orders.c_id);
```

Output:

fname	lname
Shubham	Gupta
Divya	Walecha
Rajiv	Mehta
Anand	Mehra

2. **Using NOT with EXISTS**

Fetch last and first name of the customers who has not placed any order.

```
SELECT lname, fname
FROM Customer
WHERE NOT EXISTS (SELECT *
                  FROM Orders
                  WHERE Customers.customer_id = Orders.c_id);
```

Output:

lname	fname
Singh	Dolly
Chauhan	Anuj
Kumar	Niteesh
Jain	Sandeep

3. **Using EXISTS condition with DELETE statement**

Delete the record of all the customer from Order Table whose last name is 'Mehra'.

```
DELETE
FROM Orders
WHERE EXISTS (SELECT *
              FROM customers
              WHERE Customers.customer_id = Orders.cid
              AND Customers.lname = 'Mehra');
```

```
SELECT * FROM Orders;
```

Output:

order_id	c_id	order_date
1	407	2017-03-03
2	405	2017-03-05
4	404	2017-02-05

4. **Using EXISTS condition with UPDATE statement**

   Update the lname as 'Kumari' of customer in Customer Table whose customer_id is 401.

```
UPDATE Customers
SET lname = 'Kumari'
WHERE EXISTS (SELECT *
              FROM Customers
              WHERE customer_id = 401);


SELECT * FROM Customers;
```

Output:

customer_id	lname	fname	website
401	Kumari	Dolly	abc.com
402	Chauhan	Anuj	def.com
403	Kumar	Niteesh	ghi.com
404	Gupta	Shubham	jkl.com
405	Walecha	Divya	abc.com
406	Jain	Sandeep	jkl.com
407	Mehta	Rajiv	abc.com
408	Mehra	Anand	abc.com

This article is contributed by Anuj Chauhan. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Last Updated : 27 Apr, 2017

48

Similar Reads

SQL | Aliases

Pre-Requisites:SQL table

Aliases are the temporary names given to tables or columns for the purpose of a particular SQL query. It is used when the name of a column or table is used other than its original name, but the modified name is only temporary.

- Aliases are created to make table or column names more readable.
- The renaming is just a temporary change and the table name does not change in the original database.
- Aliases are useful when table or column names are big or not very readable.
- These are preferred when there is more than one table involved in a query.

Syntax for Column Alias

SELECT column as alias_name FROM table_name;

column: fields in the table

alias_name: temporary alias name to be used in

replacement of original column name

table_name: name of table

Parameter Explanation

The following table explains the arguments in detail:

- Column_Name: The column name can be defined as the column on which we are going to create an alias name.
- Alias_Name: It can be defined as a temporary name that we are going to assign for the column or table.
- AS: It is optional. If you have not specified it, there is no effect on the query execution.

Syntax for Table Alias

SELECT column FROM table_name as alias_name;

column: fields in the table

table_name: name of table

alias_name: temporary alias name to be used in replacement

of original table name

Lets see examples for SQL Aliases.

```
CREATE TABLE Customer(
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(50),
    LastName VARCHAR(50),
    Country VARCHAR(50),
    Age int(2),
  Phone int(10)
);
-- Insert some sample data into the Customers table
INSERT INTO Customer (CustomerID, CustomerName, LastName, Country, Age, Phone)
VALUES (1, 'Shubham', 'Thakur', 'India','23','xxxxxxxxxx'),
       (2, 'Aman ', 'Chopra', 'Australia','21','xxxxxxxxxx'),
       (3, 'Naveen', 'Tulasi', 'Sri lanka','24','xxxxxxxxxx'),
       (4, 'Aditya', 'Arpan', 'Austria','21','xxxxxxxxxx'),
       (5, 'Nishant. Salchichas S.A.', 'Jain', 'Spain','22','xxxxxxxxxx');
       Select * from Customer;
```

Output:

CustomerID	CustomerName	LastName	Country	Age	Phone
1	Shubham	Thakur	India	23	xxxxxxxxxx
2	Aman	Chopra	Australia	21	xxxxxxxxxx
3	Naveen	Tulasi	Sri lanka	24	xxxxxxxxxx
4	Aditya	Arpan	Austria	21	xxxxxxxxxx
5	Nishant. Salchichas S.A.	Jain	Spain	22	xxxxxxxxxx

Example 1: Column Alias

To fetch SSN from the customer table using CustomerID as an alias name.

Query:

```
SELECT CustomerID AS SSN FROM Customer;
```

Output:

SSN
1
2
3
4
5

Example 2: Table Alias

Generally, table aliases are used to fetch the data from more than just a single table and connect them through field relations.

To fetch the CustomerName and Country of the customer with Age = 21.

Query:

```
SELECT s.CustomerName, d.Country
FROM Customer AS s, Customer
AS d WHERE s.Age=21 AND
s.CustomerID=d.CustomerID;
```

Output:

CustomerName	Country
Aman	Australia
Aditya	Austria

Advantages of SQL Alias

1. It is useful when you use the function in the query.
2. It can also allow us to combine two or more columns.
3. It is also useful when the column names are big or not readable.
4. It is used to combine two or more columns.

This article is contributed by **Pratik Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please comment if you find anything incorrect or want to share more information about the topic discussed above.

Last Updated : 05 Apr, 2023

24

Similar Reads

1. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
2. Configure SQL Jobs in SQL Server using T-SQL
3. SQL vs NO SQL vs NEW SQL
4. SQL | Procedures in PL/SQL
5. SQL | Difference between functions and stored procedures in PL/SQL
6. SQL SERVER – Input and Output Parameter For Dynamic SQL
7. Difference between T-SQL and PL-SQL
8. Difference between SQL and T-SQL
9. SQL Server | Convert tables in T-SQL into XML
10. SQL SERVER | Bulk insert data from csv file using T-SQL command

SQL | Alternative Quote Operator

classyallrounder

This post is a continuation of the SQL Concatenation Operator.

Now, suppose we want to use **apostrophe** in our literal value but we can't use it directly.

See **Incorrect** code:
SELECT id, first_name, last_name, salary,
first_name||' has salary's '||salary
AS "new" FROM one

So above we are getting error, because Oracle server thinking that the **first apostrophe** is for the **starting literal** and the **second apostrophe** is for the **ending literal**, so what about the **third apostrophe**???. That's why we get error.

Alternative Quote Operator(*q*)

:
To **overcome** the above problem Oracle introduce an operator known as **Alternative Quote Operator**(q).

Let's see through an example:

```
Query that uses Alternative Quote Operator(q)
SELECT id, first_name, last_name, salary,
first_name||q'{ has salary's }'||salary
AS "new" FROM myTable
```

```
Output:
See, we are able to use apostrophe in the
new column as a literal value of myTable

ID  FIRST_NAME LAST_NAME  SALARY           new
3    Shane     Watson     50000    Shane has salary's 50000
1    Rajat     Rawat      10000    Rajat has salary's 10000
2    Geeks     ForGeeks   20000    Geeks has salary's 20000
3    MS        Dhoni      90000    MS    has salary's 90000
```

Here above see, **q'{** indicates starting of our literal value and then we use **}'** which indicates end of our literal value. So see here we have used apostrophe in our literal value easily(means we easily use **'s** in salary) without any error that's why we get output as Rajat has salary**'s** 50000.

So to use apostrophe in literal we first need to use **q** which is known as **alternative quote operator** after that we need to use an apostrophe ' and after that we need to use a **delimiter** and after delimiter we write our literal value, when we finished writing our literal value then again we need to **close the delimiter** which we have **opened before** and after that we need to put an **apostrophe again** and hence in this way we can use apostrophe in our literal value. This concept is known as **Alternative Quote Operator**(q).

We can use **any character** such as **{**, **<**, **(**, **[**, **!** or any character as **delimiter**. These characters are known as **delimiters**.

1 another example

:

Without using Quote Operator:

```
Here we get Error since we are using apostrophe in our literal value directly.

Error code below:
SELECT id, name, dept, name||' work's in '||dept||'
 department' AS "work" FROM myTable2
```

Using Quote Operator:
SELECT id, name, dept, name||q'[work's in ']'||dept||'
department' AS "work" FROM myTable2

```
Output:
See, we are able to use apostrophe in the
work column as a literal value of myTable2
```

```
ID NAME      DEPT           work
1  RR        Executive  RR work's in 'Executive department
2  GFG       HR         GFG work's in 'HR department
3  Steve     Sales      Steve work's in 'Sales department
4  Bhuvi     CSE        Bhuvi work's in 'CSE department
```

Here above see, **q'[** indicates starting of our literal value and the we use **]'** which indicate end of our literal value. So see here we have used **apostrophe** in our literal value easily(means we easily use **'s** in work) without any error that's why we get output as RR **work's** in Executive Department.]

Here above we use **[** as delimiter so it is **not** a limitation in using delimiter means we can use any character as delimiter.

References:

About Alternative Quote Operator,

Performing SQL Queries Online

Last Updated : 21 Mar, 2018

6

Similar Reads

1. What is Alternative to _N_ in PROC SQL?
2. QUOTE () function in MySQL
3. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
4. Configure SQL Jobs in SQL Server using T-SQL
5. SQL | BETWEEN & IN Operator
6. SQL | MINUS Operator
7. SQL | NOT Operator
8. SQL | Concatenation Operator
9. Difference between = and IN operator in SQL
10. SQL | UNION Operator

Previous Next