

SELECT DISTINCT clause in use

In this reading, you'll explore the usage of SELECT DISTINCT to retrieve a unique set of values in a SELECT statement. You've learned about the purpose and the syntax of SELECT DISTINCT and how it behaves in a SELECT statement. The main objective of this reading is to present some more examples and practical scenarios that use the DISTINCT keyword in the SELECT statement.

The DISTINCT keyword

DISTINCT is useful for retrieving a set of unique values when there are duplicate column values in a table. It is used with the SELECT statement, so it's commonly referred to as SELECT DISTINCT. In short, what DISTINCT does is to find unique values within a column, or columns, of a table.

Let's look at some examples of how the DISTINCT keyword behaves using a few data retrieval scenarios from the table in the sample database.

Using SELECT DISTINCT on a single column

If there's a table named invoices with the same BillingCountry repeated in many instances, you can run the following query to identify what they are:

```
1  SELECT BillingCountry
2
3  FROM invoices
4
5  ORDER BY BillingCountry;
```

Run

Reset

When you look at the result, you'll notice that there are duplicate values in the BillingCountry column. How can you obtain a list of unique billing countries where the invoices have been raised? Let's change the SELECT statement by adding the DISTINCT keyword and then run it again.

```
1  SELECT DISTINCT BillingCountry
2
3  FROM invoices
4
5  ORDER BY BillingCountry;
```

Run

Reset

This time, the duplicate values are gone and only a unique set of billing countries are returned as the result. Where there are repeating values in the BillingCountry column, for example for Argentina, Australia and Austria. The above SELECT DISTINCT query will eliminate those duplicate rows and generate the result as a unique set of values.

Using SELECT DISTINCT on multiple columns

If you inspect the values in the BillingCountry and BillingCity columns, you'll notice that the same billing City repeats for a single billing country. You can run the following code to verify this.

```
1  SELECT BillingCountry, BillingCity
2
3  FROM invoices;
```

Run

Reset

So how can you generate list of unique billing cities within the billing countries?

You can run a query that adds the DISTINCT keyword to the SELECT statement.

```
1  SELECT DISTINCT BillingCountry, BillingCity
```

```
2
3 FROM invoices
4
5 ORDER BY BillingCountry, BillingCity;
```

[Run](#)[Reset](#)

Note: The ORDER BY clause is added here to sort the values for easy reference.

The result is a unique set of billing cities retrieved for the billing countries. Basically, there are no duplicate values in the BillingCity column. In other words, when you do a DISTINCT of multiple columns, it looks for a combination of unique values in all those columns. In this example, all combinations of BillingCountry and BillingCity in the result are unique.

NULL values in a DISTINCT column

Let's say there are NULL values in a DISTINCT column(s). For example, in the BillingCity column. You can run the same query as before to get the unique billing cities within the billing countries.

```
1 SELECT DISTINCT BillingCountry, BillingCity
2
3 FROM invoices
4
5 ORDER BY BillingCountry, BillingCity;
```

[Run](#)[Reset](#)

Provided that for some records the BillingCity column has NULL values, you'll receive records with a combination of some value for BillingCountry and NULL for BillingCity.

So, it's important to know that SELECT DISTINCT treats any NULL values in the DISTINCT column(s) as unique. Therefore, in this case, it looks for a combination of unique BillingCountry and BillingCity values. Any NULL values in the BillingCity column are considered unique values. For example, **Argentina – NULL** could be one unique combination and **Australia – NULL** could be another.

Using DISTINCT with SQL aggregate functions

DISTINCT can also be used with SQL aggregate functions like COUNT, AVG, MAX and so on. In this case, you must specify an expression that's written using some aggregate function. Therefore, it's not only column names that you can use DISTINCT with but also with expressions.

What if you want to find out the number of unique countries of the customers in the customer table? Run a SELECT statement that uses the aggregate function COUNT on the country column along with DISTINCT.

For example:

```
1 SELECT COUNT(DISTINCT country)
2
3 FROM customers;
```

[Run](#)[Reset](#)


The result that you get is the number of unique countries that the customers come from. Using DISTINCT on the country column/field gives a unique list of countries and the COUNT aggregate function counts the number of results. Here are some important points to remember in terms of SELECT DISTINCT:

When only one column or expression is provided in the DISTINCT clause, the query will return the unique values for that column.

When more than one column or expression is provided in the DISTINCT clause, the query will retrieve unique combinations for those columns.

The DISTINCT clause doesn't ignore NULL values in DISTINCT column(s). NULL values are considered as unique values by DISTINCT.

Mark as completed

 Like  Dislike 

Report an
issue