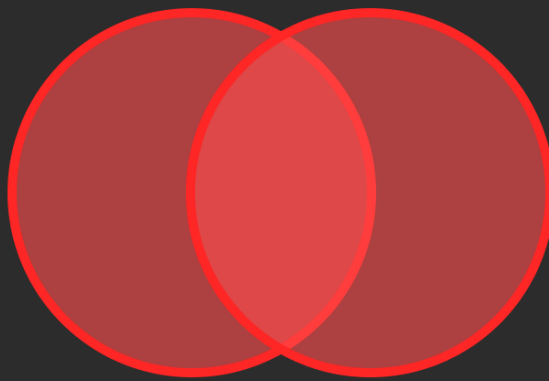




What are UNION, MINUS, and INTERSECT commands in DBMS?



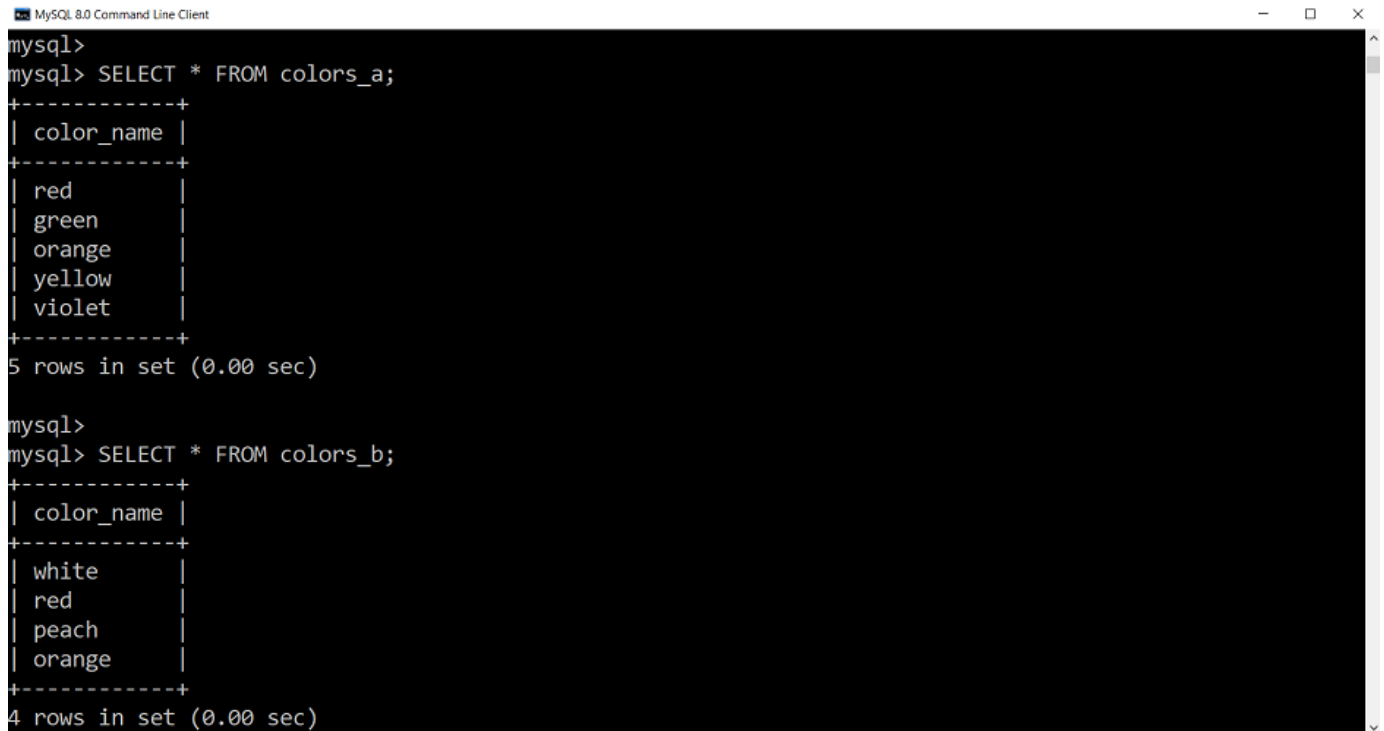
UNION, MINUS, and INTERSECT commands in DBMS

afteracademy.com

In DBMS, we deal with multiple entities and perform various operations on them. Some of the most commonly used database operations are union, minus, and intersect. So in this blog, we'll learn about these DBMS commands in detail with their implementations.

Note: In this blog, We will take all the examples using the MySQL database only. We'll be using the below-mentioned colors_a and colors_b table for

demonstrating the examples. The tables are as follows:



```
mysql>
mysql> SELECT * FROM colors_a;
+-----+
| color_name |
+-----+
| red        |
| green      |
| orange     |
| yellow     |
| violet     |
+-----+
5 rows in set (0.00 sec)

mysql>
mysql> SELECT * FROM colors_b;
+-----+
| color_name |
+-----+
| white      |
| red        |
| peach      |
| orange     |
+-----+
4 rows in set (0.00 sec)
```

Now let us learn about these database operations one-by-one.

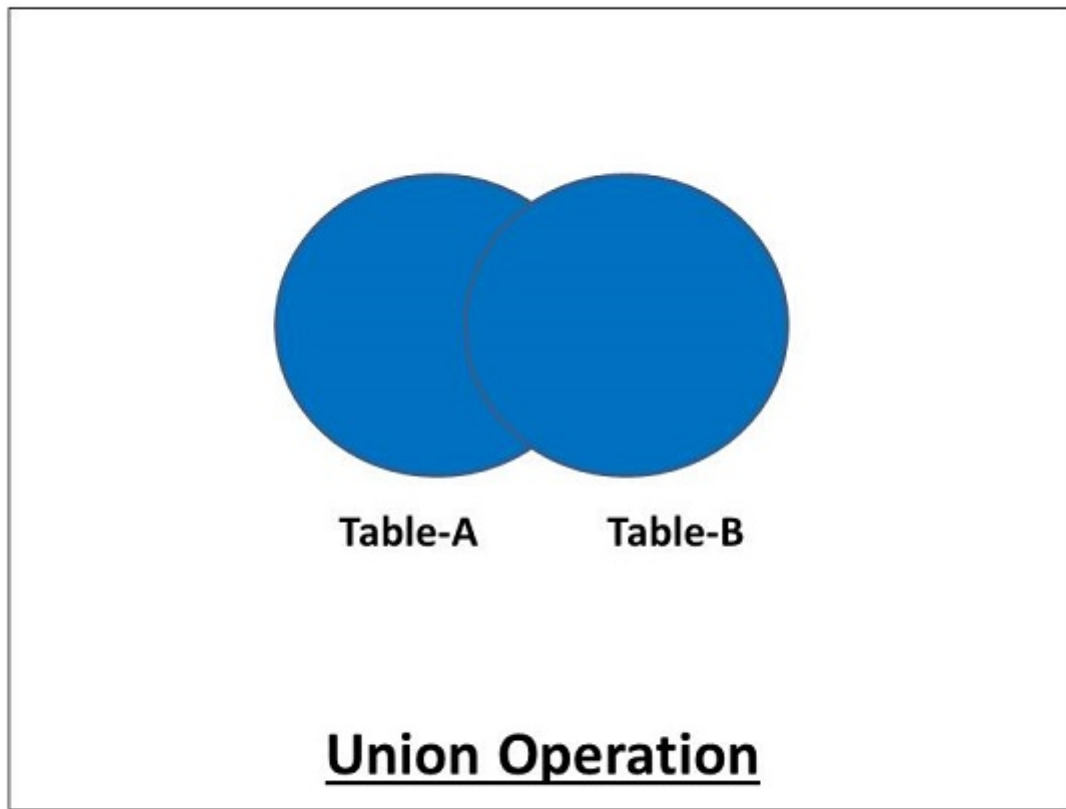
UNION

The Union is a binary set operator in DBMS. It is used to combine the result set of two select queries. Thus, It combines two result sets into one. In other words, the result set obtained after union operation is the collection of the result set of both the tables.

But two necessary conditions need to be fulfilled when we use the union command. These are:

1. Both SELECT statements should have an equal number of fields in the same order.
2. The data types of these fields should either be the same or compatible with each other.

The Union operation can be demonstrated as follows:



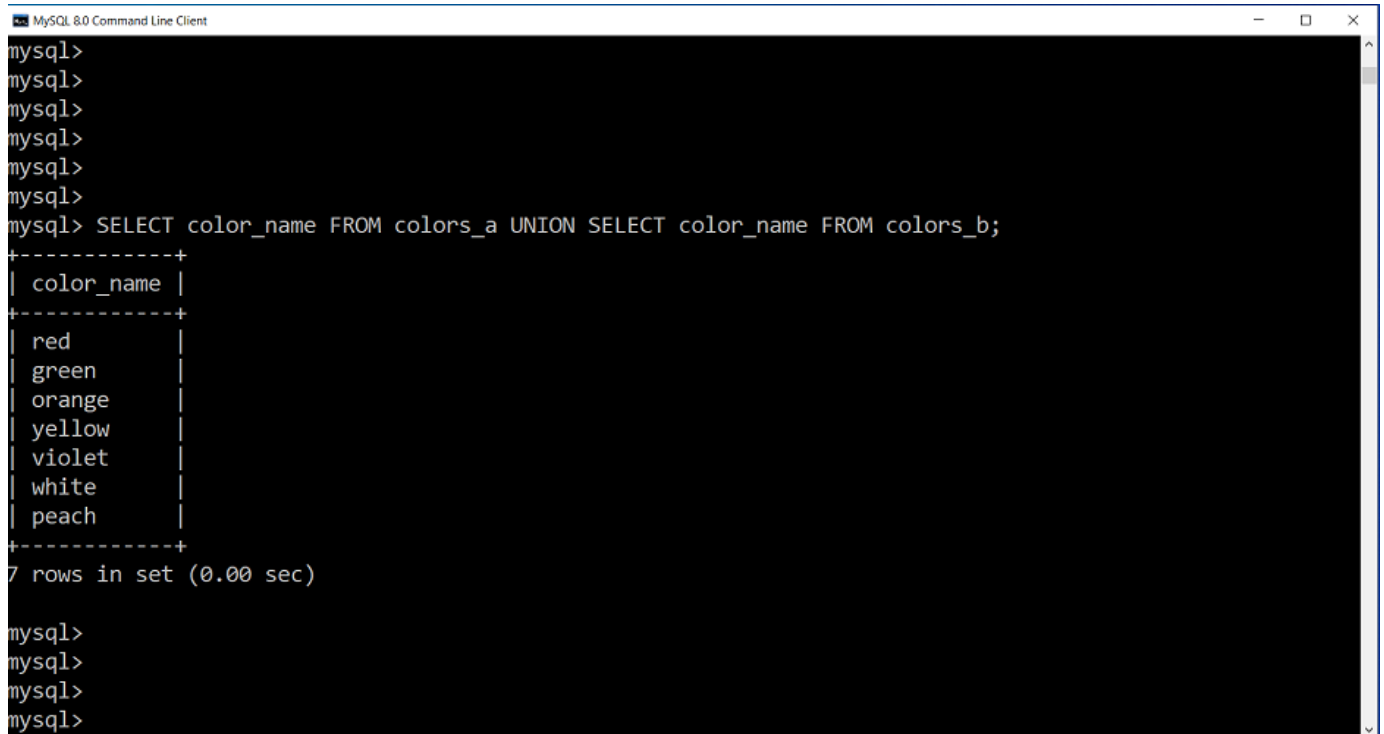
The syntax for the union operation is as follows:

```
SELECT (column_names) from table1 [WHERE condition] UNION SELECT (co
```

The MySQL query for the union operation can be as follows:

```
SELECT color_name FROM colors_a UNION SELECT color_name FROM colors_b
```

The returned values for the above query is as follows:

A screenshot of the MySQL 8.0 Command Line Client window. The window has a title bar with the text "MySQL 8.0 Command Line Client" and standard window controls. The command prompt shows a series of "mysql>" prompts. The final command entered is "SELECT color_name FROM colors_a UNION SELECT color_name FROM colors_b;". The output is a table with one column, "color_name", containing seven rows of color names: red, green, orange, yellow, violet, white, and peach. The output is formatted with a header row and vertical bars separating the column name from the data. Below the table, it says "7 rows in set (0.00 sec)".

```
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> SELECT color_name FROM colors_a UNION SELECT color_name FROM colors_b;
+-----+
| color_name |
+-----+
| red        |
| green      |
| orange     |
| yellow     |
| violet     |
| white      |
| peach      |
+-----+
7 rows in set (0.00 sec)

mysql>
mysql>
mysql>
mysql>
```

The Union operation gives us distinct values. If we want to allow the duplicates in our result set, we'll have to use the 'Union-All' operation.

Union All operation is also similar to the union operation. The only difference is that it allows duplicate values in the result set.

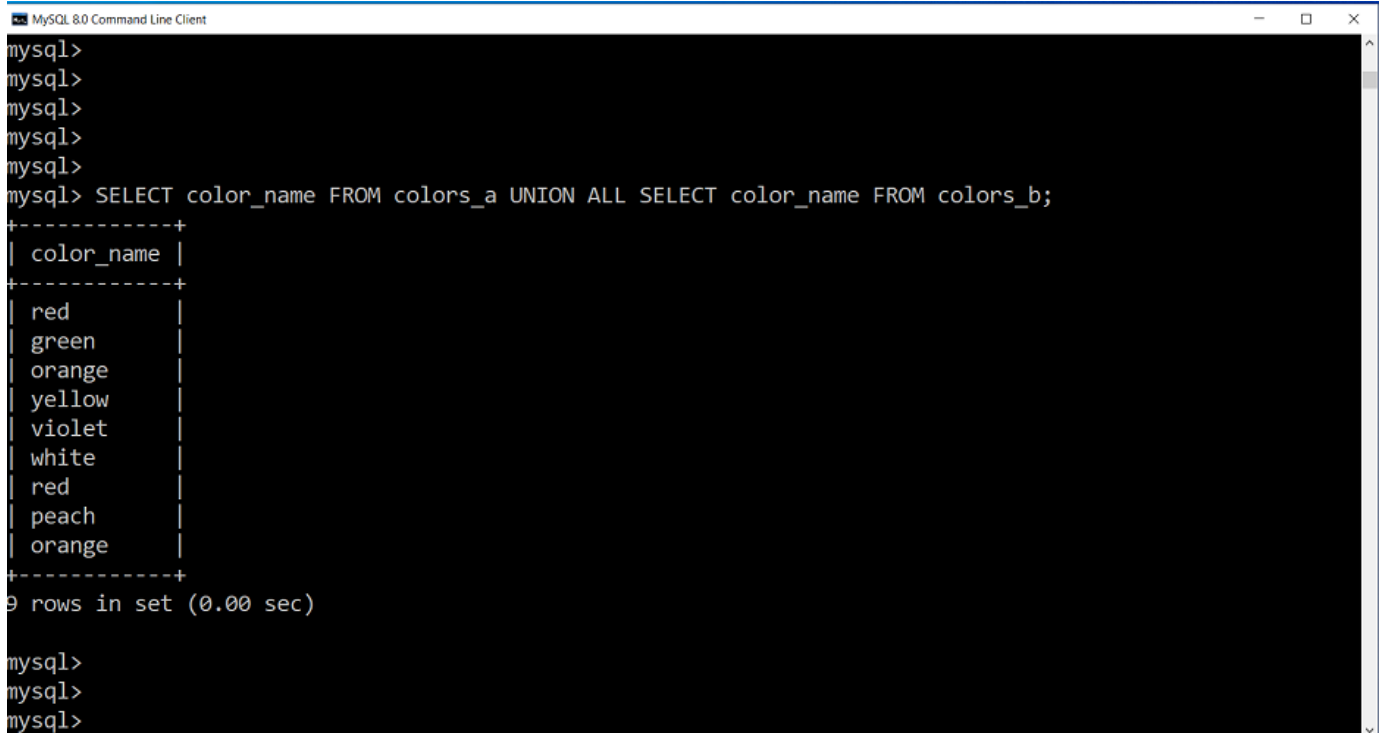
The syntax for the union all operation is as follows:

```
SELECT (column_names) from table1 [WHERE condition] UNION ALL SELECT
```

The MySQL query for the union all operation can be as follows:

```
SELECT color_name FROM colors_a UNION ALL SELECT color_name FROM colo
```

The returned values for the above query is as follows:



```
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> SELECT color_name FROM colors_a UNION ALL SELECT color_name FROM colors_b;
+-----+
| color_name |
+-----+
| red        |
| green      |
| orange     |
| yellow     |
| violet     |
| white      |
| red        |
| peach      |
| orange     |
+-----+
9 rows in set (0.00 sec)

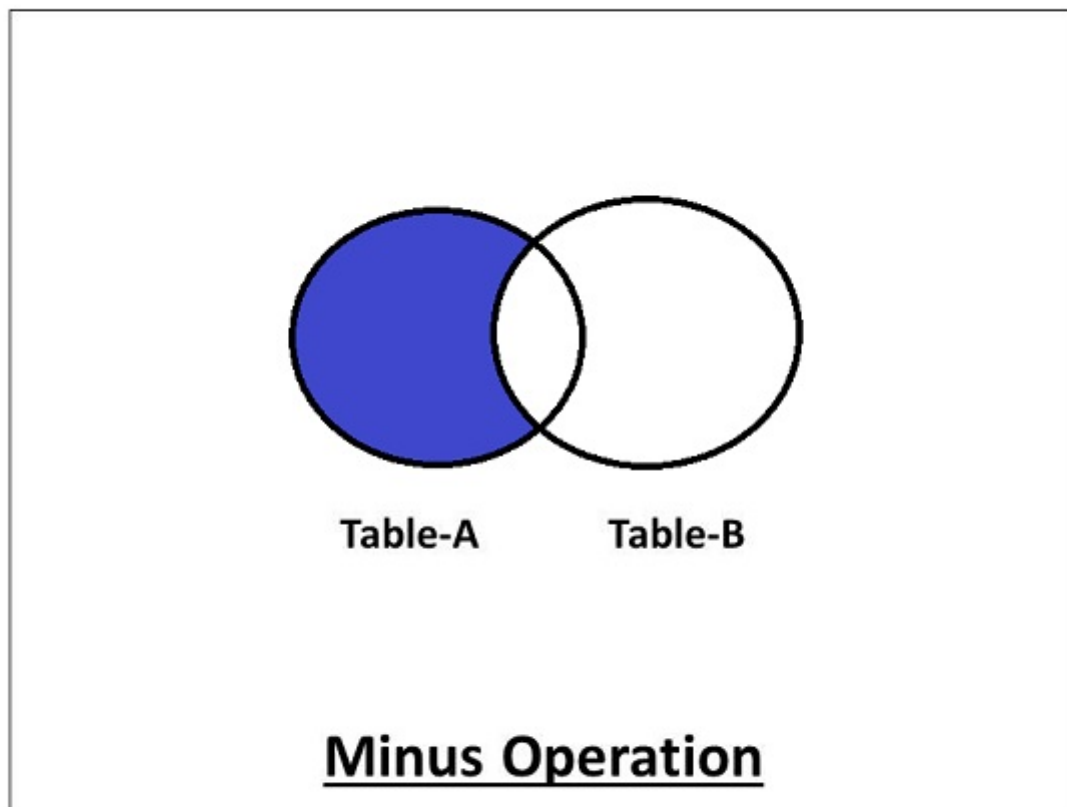
mysql>
mysql>
mysql>
```

MINUS

Minus is a binary set operator in DBMS. **The minus operation between two selections returns the rows that are present in the first selection but not in the second selection.** The Minus operator returns only the distinct rows from the first table.

It is a must to follow the above conditions that we've seen in the union, i.e., the number of fields in both the SELECT statements should be the same, with the same data type, and in the same order for the minus operation.

The minus operation can be demonstrated as follows:



The syntax for the minus operation is as follows:

```
SELECT (column_names) from table1 [WHERE condition] MINUS SELECT (co
```

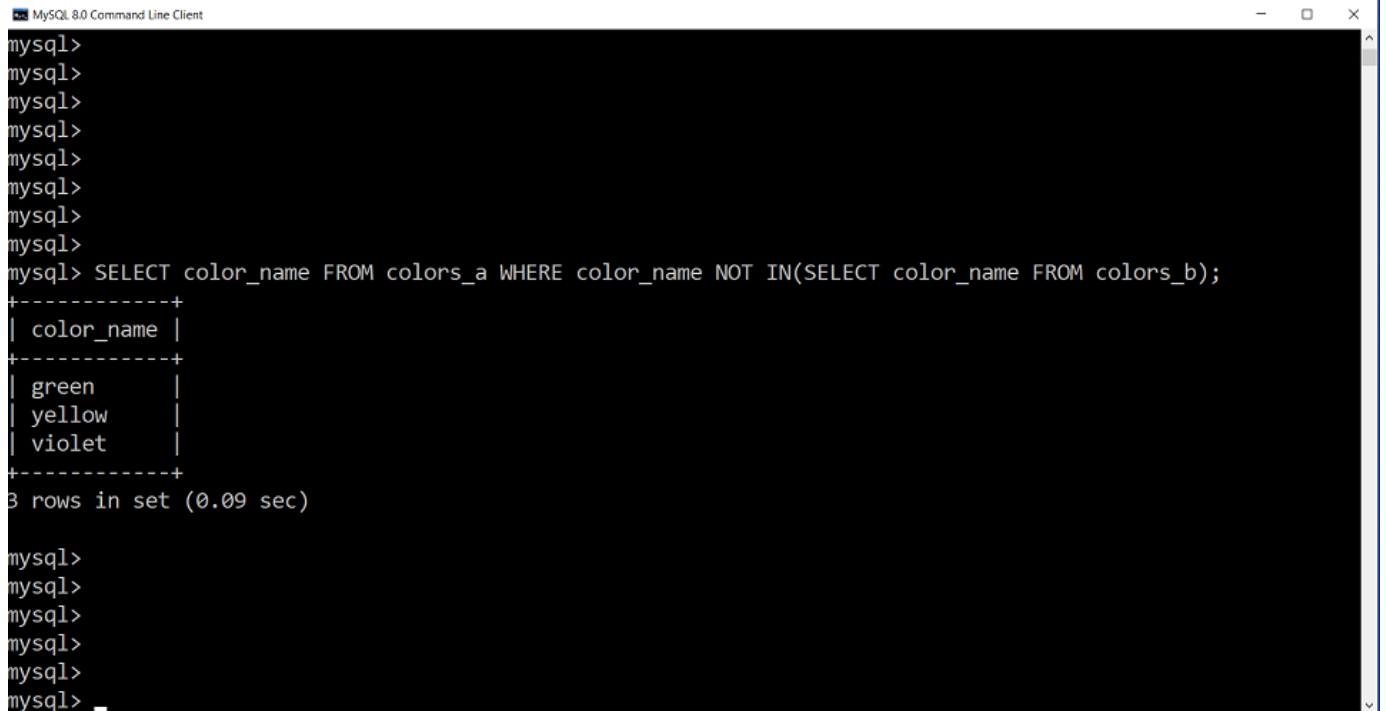
Note: It is to be noted that the minus operator is not present in MySQL. But we can make use of either 'NOT IN' operator or 'JOIN' for performing a minus operation in MySQL.

Here, we first see the 'NOT IN' operator for demonstrating the examples.

The MySQL query for the minus operation using the 'NOT IN' operator can be as follows:

```
SELECT color_name FROM colors_a WHERE color_name NOT IN(SELECT color_n
```

The returned values for the above query is as follows:

A screenshot of the MySQL 8.0 Command Line Client window. The window has a title bar with the text 'MySQL 8.0 Command Line Client' and standard window controls. The main area shows a series of 'mysql>' prompts. The last prompt is followed by the query: 'SELECT color_name FROM colors_a WHERE color_name NOT IN(SELECT color_name FROM colors_b);'. Below the query, the result is displayed in a table format with a header 'color_name' and three rows: 'green', 'yellow', and 'violet'. The output is preceded by a '+' separator and followed by '3 rows in set (0.09 sec)'.

```
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> SELECT color_name FROM colors_a WHERE color_name NOT IN(SELECT color_name FROM colors_b);
+-----+
| color_name |
+-----+
| green      |
| yellow     |
| violet     |
+-----+
3 rows in set (0.09 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
```

The MySQL query for the minus operation using ' JOIN ' can be as follows:

```
SELECT color_name FROM colors_a LEFT JOIN colors_b USING (color_name)
```

The returned values for the above query is as follows:

```
MySQL 8.0 Command Line Client
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> SELECT color_name FROM colors_a LEFT JOIN colors_b USING (color_name) WHERE colors_b.color_name IS
NULL;
+-----+
| color_name |
+-----+
| green      |
| yellow     |
| violet     |
+-----+
3 rows in set (0.00 sec)

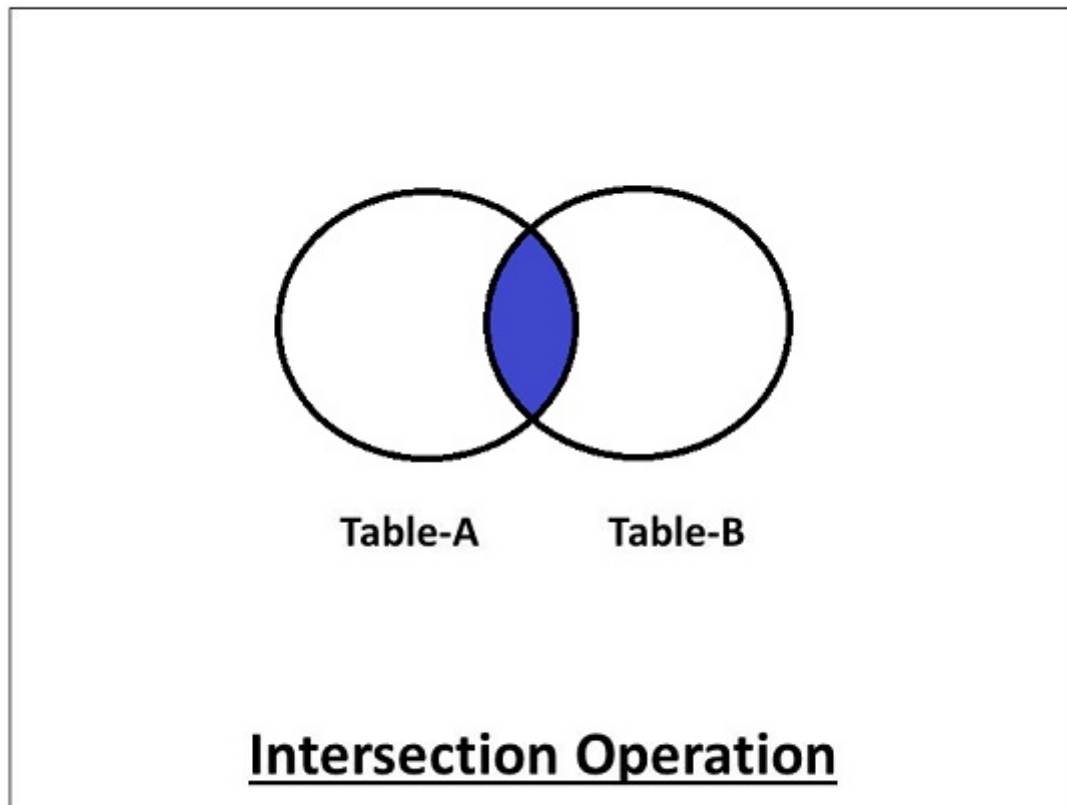
mysql>
mysql>
mysql>
mysql>
mysql>
```

INTERSECT

Intersect is a binary set operator in DBMS. **The intersection operation between two selections returns only the common data sets or rows between them.** It should be noted that the intersection operation always returns the distinct rows. The duplicate rows will not be returned by the intersect operator.

Here also, the above conditions of the union and minus are followed, i.e., the number of fields in both the SELECT statements should be the same, with the same data type, and in the same order for the intersection.

The intersection operation can be demonstrated as follows:



The syntax for the intersection operation is as follows:

```
SELECT (column_names) from table1[WHERE condition] INTERSECT SELECT
```

Note: It is to be noted that the intersect operator is not present in MySQL. But we can make use of either 'IN' or 'Exists' operator for performing an intersection operation in MySQL.

Here, we are using the 'IN' clause for demonstrating the examples.

The MySQL query for the intersection operation using the 'IN' operator can be as follows:

```
SELECT color_name FROM colors_a WHERE color_name IN(SELECT color_name
```

The returned values for the above query is as follows:

A screenshot of the MySQL 8.0 Command Line Client window. The window title is "MySQL 8.0 Command Line Client". The terminal shows a series of "mysql>" prompts. The last prompt has the query "SELECT color_name FROM colors_a WHERE color_name IN(SELECT color_name FROM colors_b);". The output is a table with one column "color_name" and two rows: "red" and "orange". The output is formatted with a header row and a separator row. Below the table, it says "2 rows in set (0.05 sec)".

```
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> SELECT color_name FROM colors_a WHERE color_name IN(SELECT color_name FROM colors_b);
+-----+
| color_name |
+-----+
| red        |
| orange     |
+-----+
2 rows in set (0.05 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
```

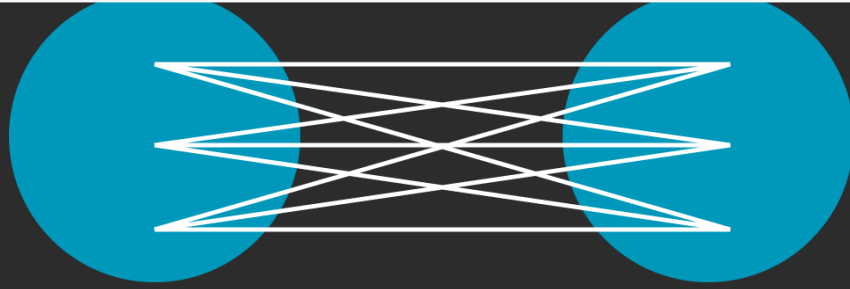
This is all about the UNION, MINUS, INTERSECT operators in DBMS. Hope you learned something new today. That's it for this blog.

Do share this blog with your friends to spread the knowledge. Visit our [YouTube channel](#) for more content. You can read more blogs from [here](#).

Keep Learning :)

Team AfterAcademy!

Recommended for You



What is Cross-Join in DBMS?

In this blog, we will learn how to cross-join two tables in DBMS, and what are the benefits of doing it with example.



Admin AfterAcademy
6 Apr 2020



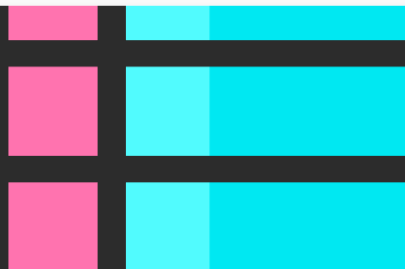
Difference between Trigger and

Difference between Trigger and Stored procedure in DBMS

In this blog, we will learn how triggers are different from stored procedures.



Admin AfterAcademy
16 Mar 2020



Stored Procedure in DBMS

What is a Stored Procedure in DBMS?

In this blog, we will learn what is a stored procedure in DBMS, and how it is executed depending on the number of parameters passed.



Admin AfterAcademy
21 Mar 2020

Self Join in DBMS

What is Self-Join in DBMS?

In this blog, we will learn how to join a table by itself, and what are the benefits of self-join with example.



Admin AfterAcademy
21 Mar 2020

Join in DBMS and its types

What is Join in DBMS and what are its types?

In this blog, we will learn how to join two tables in DBMS. We will also learn about the various types of joins, mainly the inner and the outer join.



Admin AfterAcademy
21 Mar 2020



Deadlock in DBMS, deadlock avoidance, detection, and

What is a Deadlock in DBMS and what are the deadlock avoidance, detection, and prevention techniques?

In this blog, we will learn what is a deadlock situation, what are various deadlock handling techniques like deadlock prevention, deadlock avoidance, deadlock detection, and deadlock ignorance.



Admin AfterAcademy
21 Mar 2020

Connect With Your Mentors



Janishar Ali

Founder | IIT-BHU | 10 Yrs Exp.



Amit Shekhar

Founder | IIT-BHU | 10 Yrs Exp.



© Copyright 2022, MindOrks Nextgen Private Limited