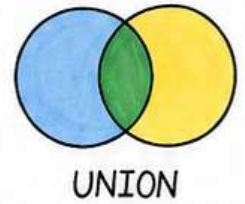


Chapter 15 : MySQL Set Operations

15.1 UNION

- * UNION is used to combine the result sets of two or more SELECT statements into a single result set.
- * It removes duplicates rows from the final result.
- * The number of columns and their data types must be the same in all SELECT statements.
- * The columns are combined based on their position, not on name.



Syntax :

```
SELECT column1, column2, ...  
FROM table1  
UNION  
SELECT column1, column2, ...  
FROM table2;
```

Example :

```
SELECT name FROM employees;  
UNION  
SELECT name FROM customers;
```



Example Tables :

employees

id	name
1	Alice
2	Bob
3	Charlie

customers

id	name
1	David
2	Eve
3	Alice

Result (UNION):

name
Alice
Bob
Charlie
David
Eve

Note :
"Alice" appears only once because UNION removes duplicates.

15.2 UNION ALL

- * UNION ALL is used to combine the result sets of two or more SELECT statements.
- * It includes all rows from both result sets, including duplicates.
- * The number of columns and their data types must be the same.
- * The columns are combined based on their position.



Syntax :

```
SELECT column1, column2, ...  
FROM table1  
UNION ALL  
SELECT column1, column2, ...  
FROM table2;
```

Example :

```
SELECT name FROM employees;  
UNION ALL  
SELECT name FROM customers;
```



Example Tables :

employees

id	name
1	Alice
2	Bob
3	Charlie

customers

id	name
1	David
2	Eve
3	Alice

Result (UNION ALL):

name
Alice
Bob
Charlie
David
Eve
Alice

Note :
All rows are included, even duplicates.



Key Takeaway :

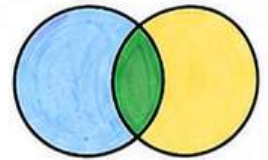
Use UNION when you want unique results.

Use UNION ALL when you want all results including duplicates.



15.3 INTERSECT

- * **INTERSECT** is used to return only the **common rows** between two result sets.
- * It removes all rows that are **not present in both** result sets.
- * The number of columns and their data types must be the same in all **SELECT** statements.
- * The columns are combined based on their **position**, not on name.



INTERSECT

Syntax :

```
SELECT column1, column2, ...  
FROM table1  
INTERSECT  
SELECT column1, column2, ...  
FROM table2;
```

Example :

```
SELECT name FROM employees;  
INTERSECT  
SELECT name FROM customers;
```



Example Tables :

employees

id	name
1	Alice
2	Bob
3	Charlie
4	David

customers

id	name
1	David
2	Eve
3	Alice
4	Frank

Result (INTERSECT):

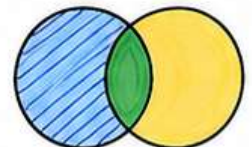
name
Alice
David

Note :

Only names that exist in both tables are returned.

15.4 EXCEPT

- * **EXCEPT** is used to return rows from the first result set that are **not present in the second** result set.
- * It removes the common rows and keeps only the **unique rows** from the first result set.
- * The number of columns and their data types must be the same in all **SELECT** statements.
- * The columns are combined based on their **position**, not on name.



EXCEPT

Syntax :

```
SELECT column1, column2, ...  
FROM table1  
EXCEPT  
SELECT column1, column2, ...  
FROM table2;
```

Example :

```
SELECT name FROM employees;  
EXCEPT  
SELECT name FROM customers;
```



Example Tables :

employees

id	name
1	Alice
2	Bob
3	Charlie
4	David

customers

id	name
1	David
2	Eve
3	Alice
4	Frank

Result (EXCEPT):

name
Bob
Charlie

Note :

Names that are in employees but not in customers.



Key Takeaway :

- ☒ Use **INTERSECT** to get only common rows.
- ☒ Use **EXCEPT** to get rows from the first set that are not in the second set.



15.5 Combining Set Operations

- * You can combine **multiple set operations** in a single query.
- * The operations are evaluated from **left to right**, unless parentheses are used to change the order.
- * Using parentheses gives you control over the order of operations.
- * The number of columns and their data types must be the same in all **SELECT** statements.



COMBINING SET OPERATIONS

Syntax :

```
SELECT column1, column2, ...  
FROM table1  
UNION / UNION ALL / INTERSECT / EXCEPT  
SELECT column1, column2, ...  
FROM table2  
UNION / UNION ALL / INTERSECT / EXCEPT  
SELECT column1, column2, ...  
FROM table3;
```

Example :

```
SELECT name FROM employees  
UNION  
SELECT name FROM customers  
INTERSECT  
SELECT name FROM suppliers;
```



Example Tables :

employees

name
Alice
Bob
Charlie
David

customers

name
Bob
David
Eve
Frank

suppliers

name
David
Frank
Grace
Harry

Result (Query):

name
David
Frank

Note :

First UNION gives {Alice, Bob, Charlie, David, Eve, Frank}.
INTERSECT with suppliers returns {David, Frank}.

15.6 ORDER BY with Set Operations

- * **ORDER BY** is used to **sort** the final result of the set operation.
- * It must be placed at the **end** of the entire query.
- * You cannot use **ORDER BY** in individual **SELECT** statements when using set operations.
- * It works with **UNION**, **UNION ALL**, **INTERSECT** and **EXCEPT**.



SORTED RESULTS

Syntax :

```
SELECT column1, column2, ...  
FROM table1  
UNION / UNION ALL / INTERSECT / EXCEPT  
SELECT column1, column2, ...  
FROM table2  
ORDER BY column1 [ASC | DESC];
```

Example :

```
SELECT name FROM employees  
UNION  
SELECT name FROM customers  
ORDER BY name ASC;
```



Result (ORDER BY name ASC):

name
Alice
Bob
Charlie
David
Eve

Note :

ORDER BY sorts the final combined result in ascending (ASC) or descending (DESC) order.

Example Tables :

employees

name
Charlie
Alice
David

customers

name
Bob
Eve
Alice

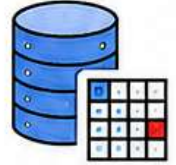
Key Takeaway :

- ☒ You can combine **multiple set operations** for complex queries.
- ☒ Always place **ORDER BY** at the **end** to sort the final result.



15.7 Practical Set Operation Queries

- * Let us understand set operations with practical examples.
- * We will use three tables: **employees**, **department** and **contractors**.
- * These queries demonstrate **UNION**, **UNION ALL**, **INTERSECT** and **EXCEPT** in real scenarios.
- * All **SELECT** statements must have the same number of columns with compatible data types.



SET OPERATIONS

Tables Used :

employees

id	name
1	Alice
2	Bob
3	Charlie
4	David

department

id	name
10	Bob
20	Eve
30	Frank
40	Grace

contractors

id	name
100	Charlie
200	David
300	Harry
400	Eve

Note :

We will use only the **name** column for set operations.

1. Using UNION

```
SELECT name FROM employees
UNION
SELECT name FROM department;
```

Result :

name
Alice
Bob
Charlie
David
Eve
Frank
Grace

UNION

removes duplicates.

2. Using UNION ALL

```
SELECT name FROM employees
UNION ALL
SELECT name FROM department;
```

Result :

name
Alice
Bob
Charlie
David
Bob
Eve
Frank
Grace

UNION ALL

includes all rows, including duplicates.

3. Using INTERSECT

```
SELECT name FROM employees
INTERSECT
SELECT name FROM contractors;
```

Result :

name
Charlie
David

INTERSECT

returns only common rows.

4. Using EXCEPT

```
SELECT name FROM employees
EXCEPT
SELECT name FROM contractors;
```

Result :

name
Alice
Bob

EXCEPT

returns rows from first set that are not in second set.

Key Takeaway :

- ✓ Use **UNION** to combine unique results from multiple sources.
- ✓ Use **UNION ALL** when you need all rows, including duplicates.
- ✓ Use **INTERSECT** to get only common rows between result sets.
- ✓ Use **EXCEPT** to get rows from first set that do not exist in second set.
- ✓ **ORDER BY** can be used at the end to sort the final result.

