SQL - UNIONS CLAUSE

The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.

- While using this UNION clause, each SELECT statement must have:
  ‣ The same number of columns selected
  ‣ The same number of column expressions
  ‣ The same data type and
  ‣ Have them in the same order

But they need not have to be in the same length.

Example

Consider the following two tables.

Table 1 – customers table is as follows:

```
+----+----------+-----+-----------+----------+
| id | name     | age | address   | salary   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Table 2 – orders table is as follows:

```
+-----+---------------------+-------------+--------+
| oid | date                | customer_id | amount |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
+-----+---------------------+-------------+--------+
```

Now, let us join these two tables in our SELECT statement as follows:

```sql
SELECT id, name, amount, date
   FROM customer
   LEFT JOIN orders
```

```
   ON customers.id = orders.customer_id
UNION
   SELECT id, name, amount, date
   FROM customer
   RIGHT JOIN orders
   ON customers.id = orders.customer_id
```

This would produce the following result:

The UNION ALL Clause

The UNION ALL operator is used to combine the results of two SELECT statements including duplicate rows.

The same rules that apply to the UNION clause will apply to the UNION ALL operator.

Example - Consider the following two tables:

- Table 1 – customers table is as follows:

```
+----+----------+-----+-----------+----------+
| id | name     | age | address   | salary   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

- Table 2 – orders table is as follows:

```
+-----+---------------------+-------------+--------+
| oid | date                | customer_id | amount |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
+-----+---------------------+-------------+--------+
```

Now, let us join these two tables in our SELECT statement as follows :

```
SELECT id, name, amount, date
   FROM customers
   LEFT JOIN orders
```

```
      ON customers.id = order.customer_id
UNION ALL
      SELECT id, name, amount, date
      FROM customers
      RIGHT JOIN orders
      ON customers.id = orders.customer_id;
```

This would produce the following result:

```
+------+----------+--------+---------------------+
| id   | name     | amount | date                |
+------+----------+--------+---------------------+
|    1 | Ramesh   |   NULL | NULL                |
|    2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|    3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|    3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|    4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
|    5 | Hardik   |   NULL | NULL                |
|    6 | Komal    |   NULL | NULL                |
|    7 | Muffy    |   NULL | NULL                |
|    3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|    3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|    2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|    4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
+------+----------+--------+---------------------+
```

Note : **There are two other clauses (i.e., operators), which are like the UNION clause.**