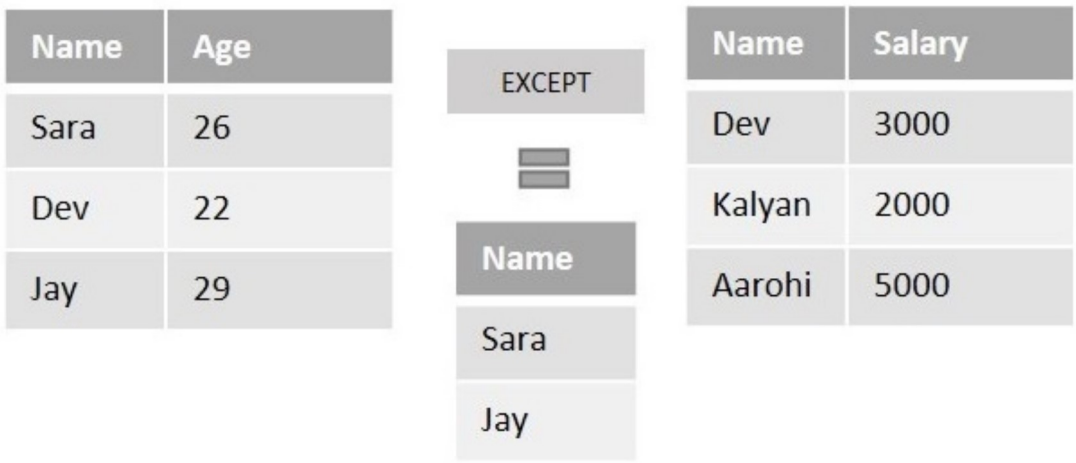
SQL - EXCEPT

The **EXCEPT** operator in SQL is used to retrieve the unique records that exist in the first table, not the common records of both tables. This operator acts as the opposite of the SQL UNION operator.

For better understanding consider two tables with records as shown in the following image –



If we perform the EXCEPT operator on the above two tables to retrieve the names, it will display the records only from the first table which are not in common with the records of the second table.

Here, "Dev" is common in both tables. So, the EXECPT operator will eliminate it and retrieves only "Sara" and "Jay" as output.

Following is the syntax of the EXCEPT operator in SQL –

```
SELECT column1, column2,..., columnN
FROM table1, table2,..., tableN
[Conditions] //optional
EXCEPT
SELECT column1, column2,..., columnN
FROM table1, table2,..., tableN
[Conditions] //optional
```

Note – The number and order of columns in both SELECT statements should be the same.

Example

First of all, let us create a table named "**STUDENTS**" using the following query –

```
SQL> CREATE TABLE STUDENTS(
   ID INT NOT NULL,
   NAME VARCHAR(20) NOT NULL,
   HOBBY VARCHAR(20) NOT NULL,
   AGE INT NOT NULL,
   PRIMARY KEY(ID)
);
```

Once the table is created, let us insert some values to the table using the query below –

```
SQL> INSERT INTO STUDENTS(ID, NAME, HOBBY, AGE) VALUES(1, 'Vijay', '
INSERT INTO STUDENTS(ID, NAME, HOBBY, AGE) VALUES(2, 'Varun', 'Footb
INSERT INTO STUDENTS(ID, NAME, HOBBY, AGE) VALUES(3, 'Surya', 'Crick
INSERT INTO STUDENTS(ID, NAME, HOBBY, AGE) VALUES(4, 'Karthik', 'Cri
INSERT INTO STUDENTS(ID, NAME, HOBBY, AGE) VALUES(5, 'Sunny', 'Footb
INSERT INTO STUDENTS(ID, NAME, HOBBY, AGE) VALUES(6, 'Dev', 'Cricket
```

```
SQL> SELECT * FROM STUDENTS;
```

As we can see in the below output, the table has been created in the database.

```
+-----+----------+-------------+-------+
| ID  | NAME     |   HOBBY     | AGE   |
+-----+----------+-------------+-------+
| 1   | Vijay    |  Cricket    |  18   |
| 2   | Varun    |  Football   |  26   |
| 3   | Surya    |  Cricket    |  19   |
| 4   | Karthik  |  Cricket    |  25   |
| 5   | Sunny    |  Football   |  26   |
| 6   | Dev      |  Cricket    |  23   |
+-----+----------+-------------+-------+
```

Let us create another table named "**ASSOCIATES**" using the following query –

```
SQL> CREATE TABLE ASSOCIATES(
   ID INT NOT NULL,
   NAME VARCHAR(20) NOT NULL,
   SUBJECT VARCHAR(20) NOT NULL,
   AGE INT NOT NULL,
   HOBBY VARCHAR(20) NOT NULL,
   PRIMARY KEY(ID)
);
```

Once the table is created, let us insert some values to the table using the query below –

```
SQL> INSERT INTO ASSOCIATES(ID, NAME, SUBJECT, AGE, HOBBY) VALUES(1,
INSERT INTO ASSOCIATES(ID, NAME, SUBJECT, AGE, HOBBY) VALUES(2, 'Var
INSERT INTO ASSOCIATES(ID, NAME, SUBJECT, AGE, HOBBY) VALUES(3, 'Dev
INSERT INTO ASSOCIATES(ID, NAME, SUBJECT, AGE, HOBBY) VALUES(4, 'Pri
```

Let us verify whether the table "**ASSOCIATES**" is created or not using the following query –

```
SQL> SELECT * FROM ASSOCIATES;
```

As we can see in the below output, the table has been created in the database.

```
+-----+----------+--------------+-------+----------+
| ID  |  NAME    |   SUBJECT    |  AGE  |  HOBBY   |
+-----+----------+--------------+-------+----------+
| 1   |  Naina   |   Mathematics |   24  | Cricket  |
| 2   |  Varun   |   Physics     |   26  | Football |
| 3   |  Dev     |   Mathematics |   23  | Cricket  |
| 4   |  Priya   |   Physics     |   25  | Cricket  |
| 5   |  Adithya |   Chemistry   |   21  | Cricket  |
| 6   |  Kalyan  |   Mathematics |   30  | Football |
+-----+----------+--------------+-------+-----------+
```

Let us retrieve the records that are **only** unique in the first table using the below query –

```
SQL> SELECT NAME, HOBBY, AGE
FROM STUDENTS

EXCEPT

SELECT NAME, HOBBY, AGE
FROM ASSOCIATES
```

Output

When we execute the above query, the output is obtained as follows –

```
+----------+-------------+-------+
```

```
|  Sunny   |  Football   |  26  |
|  Surya   |  Cricket    |  19  |
|  Vijay   |  Cricket    |  18  |
+----------+-------------+------+
```

EXCEPT with BETWEEN operator

As we discussed in the initial syntax, we can also use the EXCEPT operator along with conditional operators. We can use the EXCEPT operator with the BETWEEN operator in SQL to exclude rows that fall within a specified range.

Example

Let us retrieve the records that are **only** unique in the first table. In addition; we are retrieving the records who are aged between 20 and 30 using the following query.

```
SQL> SELECT NAME, HOBBY, AGE
FROM STUDENTS
WHERE AGE BETWEEN 20 AND 30

EXCEPT

SELECT NAME, HOBBY, AGE
FROM ASSOCIATES
WHERE AGE BETWEEN 20 AND 30
```

Output

When we execute the program query, the output is obtained as follows –

```
+---------+---------+-----+
|  NAME   |  HOBBY  | AGE |
+---------+---------+-----+
| Karthik | Cricket | 25  |
| Sunny   | Football | 26  |
```

Except with IN operator

We can also use the EXCEPT operator with the IN operator in SQL to exclude rows that have the specified values. The IN operator is used to filter a result set based on a list of specified values.

Example

Here, we are fetching the records that are **only** unique in the first table. In addition; we are using the **IN** operator to retrieve the records whose hobby is 'Cricket'.

```
SQL> SELECT NAME, HOBBY, AGE FROM STUDENTS
WHERE HOBBY IN('Cricket')

EXCEPT

SELECT NAME, HOBBY, AGE FROM ASSOCIATES
WHERE HOBBY IN('Cricket')
```

Output

When we execute the above query, the output is obtained as follows –

```
+----------+-------------+-------+
|  NAME    |   HOBBY     | AGE  |
+-----+----------+--------+-------+
|  Karthik |  Cricket    |  25  |
|  Surya   |  Cricket    |  19  |
|  Vijay   |  Cricket    |  18  |
+----------+-------------+-------+
```

EXCEPT with LIKE operator

The EXCEPT operator can also be used with the LIKE operator in SQL to exclude rows that matches with the specified pattern. The LIKE operator is

Example

Let us use the wildcard '**%**' with the **LIKE** operator to retrieve the names which starts with 'v' from the result set of first SELECT statement.

```
SQL> SELECT NAME, AGE, HOBBY FROM STUDENTS
WHERE NAME LIKE 'v%'

EXCEPT

SELECT NAME, AGE, HOBBY FROM ASSOCIATES
WHERE NAME LIKE 'v%'
```

Output

The output for the above query is produced as given below –

```
+----------+-------+----------+
|  NAME    | AGE  |  HOBBY   |
+----------+-------+----------+
|  Vijay   |  18  | Cricket  |
+----------+-------+----------+
```