

Multiple Tables

Inner Join

The **JOIN** clause allows for the return of results from more than one table by joining them together with other results based on common column values specified using an **ON** clause. **INNER JOIN** is the default **JOIN** and it will only return results matching the condition specified by **ON**.

```
SELECT *
FROM books
JOIN authors
ON books.author_id = authors.id;
```

Outer Join

An outer join will combine rows from different tables even if the join condition is not met. In a **LEFT JOIN**, every row in the *left* table is returned in the result set, and if the join condition is not met, then **NULL** values are used to fill in the columns from the *right* table.

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name =
table2.column_name;
```


Primary Key


A *primary key* column in a SQL table is used to uniquely identify each record in that table. A primary key cannot be **NULL**. In the example, **customer_id** is the primary key. The same value cannot re-occur in a primary key column. Primary keys are often used in **JOIN** operations.

 customer_id	f_name	l_name
1	Abby	Caren
2	Aaron	Paul
3	Gratian	Joseph

Foreign Key

A *foreign key* is a reference in one table's records to the primary key of another table. To maintain multiple records for a specific row, the use of foreign key plays a vital role. For instance, to track all the orders of a specific customer, the table **order** (illustrated at the bottom of the image) can contain a foreign key.

 customer_id	f_name	l_name
1	Abby	Caren
2	Aaron	Paul
3	Gratian	Joseph

 order_id	customer_id	order_qty
1	2	5
2	2	6
3	1	2

CROSS JOIN Clause

The **CROSS JOIN** clause is used to combine each row from one table with each row from another in the result set. This **JOIN** is helpful for creating all possible combinations for the records (rows) in two tables.

The given query will select the **shirt_color** and **pants_color** columns from the result set, which will contain all combinations of combining the rows in the **shirts** and **pants** tables. If there are 3 different shirt colors in the **shirts** table and 5 different pants colors in the **pants** table then the result set will contain $3 \times 5 = 15$ rows.

```
SELECT shirts.shirt_color,  
       pants.pants_color  
FROM shirts  
CROSS JOIN pants;
```

UNION Clause

The **UNION** clause is used to combine results that appear from multiple **SELECT** statements and filter duplicates.

For example, given a **first_names** table with a column **name** containing rows of data "James" and "Hermione", and a **last_names** table with a column **name** containing rows of data "James", "Hermione" and "Cassidy", the result of this query would contain three **name** s: "Cassidy", "James", and "Hermione".

```
SELECT name  
FROM first_names  
UNION  
SELECT name  
FROM last_names
```

WITH Clause

The **WITH** clause stores the result of a query in a temporary table (**temporary_movies**) using an alias.

Multiple temporary tables can be defined with one instance of the **WITH** keyword.

```
WITH temporary_movies AS (  
    SELECT *  
    FROM movies  
)  
SELECT *  
FROM temporary_movies  
WHERE year BETWEEN 2000 AND 2020;
```