Learning Objectives

Learners will be able to…

- Recognize CROSS JOIN
- Apply self JOIN
- Use UNION Operator
- Examine a Subquery

info

Make Sure You Know

Familiarity with different types of joins (INNER JOIN, LEFT JOIN, RIGHT JOIN) will be beneficial before introducing CROSS JOIN and self JOIN.

CROSS JOIN

The CROSS JOIN is used to generate a paired combination of each row of the first table with each row of the second table.

Syntax:

```
SELECT column_name(s)
FROM table1
CROSS JOIN table2;
```

important

CROSS JOIN Notes

Since this operation is the generation of all possible intersections of tables, it can generate just a huge array of data, use it only when necessary.

Example: All possible combinations between last names and first names of actors and customers.

```sql
SELECT a.first_name, a.last_name, c.first_name, c.last_name
FROM actor AS a
CROSS JOIN customer AS c
LIMIT 10;
```

Self JOIN

At times, there may be a need to join a table with itself instead of another table. The SQL syntax provides the capability to accomplish this self-join operation.

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

We can use a self join to retrieve all customers whose last name matches the first name of another customer.

To join, we need to use aliases. Copy and paste the code below into the terminal to see the results of the self join.

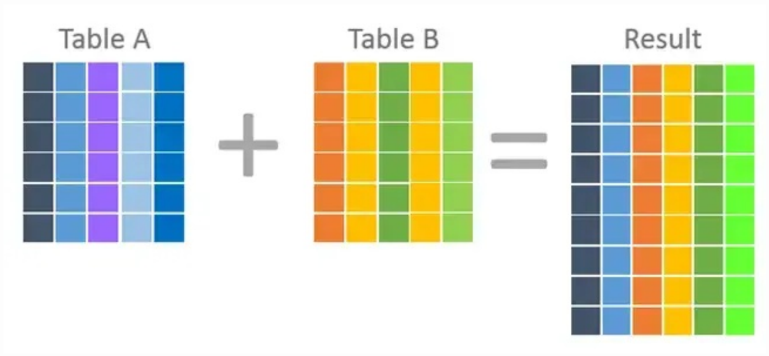
```sql
SELECT
  c1.customer_id, c1.first_name, c1.last_name,
  c2.customer_id, c2.first_name, c2.last_name
FROM customer c1, customer c2
WHERE c1.last_name = c2.first_name
ORDER BY c1.last_name;
```

The self join is often used to query hierarchical data or to compare a row with other rows within the same table.

UNION

Previously, our table connections were primarily horizontal, where we merged fields from different tables using keys or common fields. However, there is another approach where we need to vertically augment one table with another, establishing a connection in terms of their records or rows.

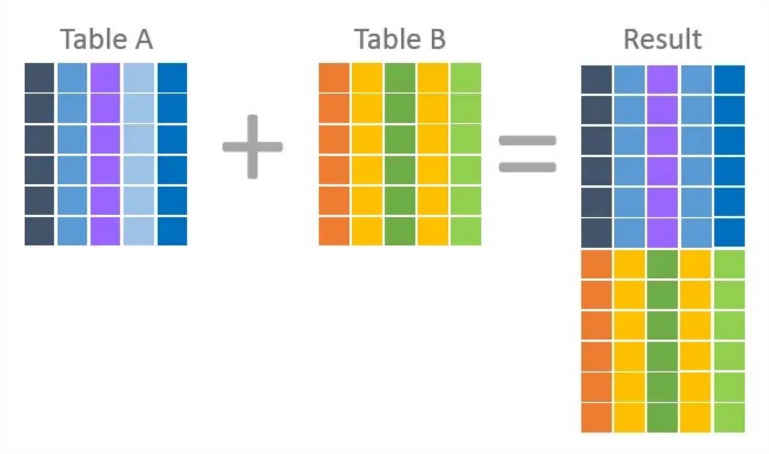
Here is a visual depiction of a join.



A picture of two tables, Table A and Table B and the results table shows a UNION between the two tables.

Picture Sources

In a union, each row within the result is from one table OR the other.



A picture of two tables, Table A and Table B and the results table shows a UNION between the two tables.

Union syntax.

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

The UNION operator is used to merge the result-set of two or more SELECT statements.

- Every SELECT statement within UNION must have an equal number columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order

```sql
SELECT a.first_name, a.last_name FROM actor AS a
UNION
SELECT c.first_name, c.last_name FROM customer AS c
ORDER BY 1
LIMIT 12;
```

By default, the union operator retrieves only unique values. To get all the values you need to **UNION ALL** operator.

```sql
SELECT a.first_name, a.last_name FROM actor AS a
UNION ALL
SELECT c.first_name, c.last_name FROM customer AS c
ORDER BY 1
LIMIT 12;
```

This operator can be used when reporting for different years, if the data is stored in separate tables.

Subqueries

Subqueries are a powerful tool in SQL that allows manipulation of the data generated by any query.

Subqueries can be used in the following places:
- SELECT clause
- FROM clause
- WHERE clause

For example, to select the languages used, you can run such a query.

```
SELECT *
FROM language
WHERE language_id IN (
    SELECT DISTINCT language_id FROM film
);
```

In this case, the subquery is this expression.

```
SELECT DISTINCT language_id FROM film
```

When a subquery is in a **FROM** clause, the subquery is called *derived table*.

```
SELECT *
FROM (
  SELECT actor.last_name, actor.first_name, film.title,
        film.film_id
  FROM film
  INNER JOIN film_actor
  ON film.film_id = film_actor.film_id
  INNER JOIN actor
  ON film_actor.actor_id = actor.actor_id
) AS t
WHERE t.last_name LIKE 'N%'
LIMIT 10;
```

In this case, we got a list of actors who starred in films with a subquery, and then filtered out those values that begin with the letter N.

info

Subqueries

Subqueries are a very powerful feature of the SQL language, but don't try to solve any problem with them, perhaps there is a better way.

The level of nesting is practically unlimited, so you can see a large number of subqueries within subqueries.

The query will find the full name and email address of all customers that have rented an sports movie.

```
SELECT first_name || ' ' || last_name AS name, email
FROM customer WHERE customer_id IN (
  SELECT customer_id FROM rental WHERE inventory_id IN (
    SELECT inventory_id FROM inventory WHERE film_id IN (
      SELECT film_id FROM film_category JOIN category ON
        film_category.category_id = category.category_id WHERE
        category.name='Sports'
    )
  )
)
ORDER BY 1
LIMIT 10;
```

In addition to the **SELECT** clause, subqueries can also be used in other language statements such as **DELETE**, **UPDATE** and **INSERT**.