**Union**

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

**To use 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.**

## Syntax

**The basic syntax of a UNION clause is as follows −**

**SELECT column1 [, column2 ]**

**FROM table1 [, table2 ]**

**[WHERE condition]**

**UNION**

**SELECT column1 [, column2 ]**

**FROM table1 [, table2 ]**

**[WHERE condition]**

## 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.**

### Syntax

**The basic syntax of the UNION ALL is as follows.**

**SELECT column1 [, column2 ]**

**FROM table1 [, table2 ]**

**[WHERE condition]**

**UNION ALL**

**SELECT column1 [, column2 ]**

**FROM table1 [, table2 ]**

**[WHERE condition]**

**Soundex**

**SOUNDEX converts an alphanumeric string to a four-character code that is based on how the string sounds when spoken in English. The first character of the code is the first character of *character\_expression*, converted to upper case. The second through fourth characters of the code are numbers that represent the letters in the expression. The letters A, E, I, O, U, H, W, and Y are ignored unless they are the first letter of the string. Zeroes are added at the end if necessary to produce a four-character code. For more information about the SOUNDEX code, see** [**The Soundex Indexing System**](https://www.archives.gov/research/census/soundex.html)**.**

**SOUNDEX codes from different strings can be compared to see how similar the strings sound when spoken. The DIFFERENCE function performs a SOUNDEX on two strings, and returns an integer that represents how similar the SOUNDEX codes are for those strings.**

**The following example shows the SOUNDEX function and the related DIFFERENCE function. In the first example, the standard SOUNDEX values are returned for all consonants. Returning the SOUNDEX for Smith and Smythe returns the same SOUNDEX result because all vowels, the letter y, doubled letters, and the letter h, are not included.**

**-- Using SOUNDEX**

**SELECT SOUNDEX ('Smith'), SOUNDEX ('Smythe');**

**Here is the result set. Valid for a Latin1\_General collation.**

**S530 S530**

**The DIFFERENCE function compares the difference of the SOUNDEX pattern results. The following example shows two strings that differ only in vowels. The difference returned is 4, the lowest possible difference.**

**- Using DIFFERENCE**

**SELECT DIFFERENCE('Smithers', 'Smythers');**

**GO**

**4**

**CROSS JOIN**

**The main idea of the CROSS JOIN is that it returns the Cartesian product of the joined tables. In the following tip, we will briefly explain the Cartesian product;**

***The Cartesian Product is a multiplication operation in the set theory that generates all ordered pairs of the given sets. Suppose that, A is a set and elements are {a,b} and B is a set and elements are {1,2,3}. The Cartesian Product of these two A and B is denoted AxB and the result will be like the following.***

***AxB ={(a,1), (a,2), (a,3), (b,1), (b,2), (b,3)}***

**SELECT ColumnName\_1,**

**ColumnName\_2,**

**ColumnName\_N**

**FROM [Table\_1]**

**CROSS JOIN [Table\_2]**

**Or we can use the following syntax instead of the previous one. This syntax does not include the CROSS JOIN keyword; only we will place the tables that will be joined after the FROM clause and separated with a comma.**

| **1**  **2**  **3**  **4** | **SELECT ColumnName\_1,**  **ColumnName\_2,**  **ColumnName\_N**  **FROM [Table\_1],[Table\_2]** |
| --- | --- |

**CREATE TABLE Meals(MealName VARCHAR(100))**

**CREATE TABLE Drinks(DrinkName VARCHAR(100))**

**INSERT INTO Drinks**

**VALUES('Orange Juice'), ('Tea'), ('Cofee')**

**INSERT INTO Meals**

**VALUES('Omlet'), ('Fried Egg'), ('Sausage')**

**SELECT \***

**FROM Meals;**

**SELECT \***

**FROM Drinks**

**SELECT \* FROM Meals**

**CROSS JOIN Drinks**