SQL Joins

SQL joins are used to query data from two or more tables, based on a relationship between certain columns in these tables.

SQL JOIN

The JOIN keyword is used in an SQL statement to query data from two or more tables, based on a relationship between certain columns in these tables.

Tables in a database are often related to each other with keys.

A primary key is a column (or a combination of columns) with a unique value for each row. Each primary key value must be unique within the table. The purpose is to bind data together, across tables, without repeating all of the data in every table.

Different SQL JOINs

Before we continue with examples, we will list the types of JOIN you can use, and the differences between them.

Self Join

A self-join is simply a normal SQL join that joins one table to itself. This is accomplished by using table name aliases to give each instance of the table a separate name. Joining a table to itself can be useful when you want to compare values in a column to other values in the same column. A join in which records from a table are combined with other records from the same table when there are matching values in the joined fields.

Inner Join

The INNER JOIN keyword return rows when there is at least one match in both tables.

Note: Inner Join is the same as the term 'Join'

The **INNER JOIN** keyword return rows when there is at least one match in both tables. If there are rows in "Persons" that do not have matches in "Orders", those rows will NOT be listed.

Left Join (sometimes called Left Outer Join)

The LEFT JOIN keyword returns all rows from the left table (table_name1), even if there are no matches in the right table (table_name2).

The LEFT JOIN keyword returns all the rows from the left table (Persons), even if there are no matches in the right table (Orders).

Right Join (sometimes called Right Outer Join)

The RIGHT JOIN keyword returns all the rows from the right table (table_name2), even if there are no matches in the left table (table_name1).

Full Outer Join

A FULL OUTER JOIN is neither "left" nor "right"— it's both! It includes all the rows from both of the tables or result sets participating in the JOIN. When no matching rows exist for rows on the "left" side of the JOIN, you see Null values from the result set on the "right." Conversely, when no matching rows exist for rows on the "right" side of the JOIN, you see Null values from the result set on the "left."

SQL INNER JOIN Keyword

The INNER JOIN keyword returns rows when there is at least one match in both tables.

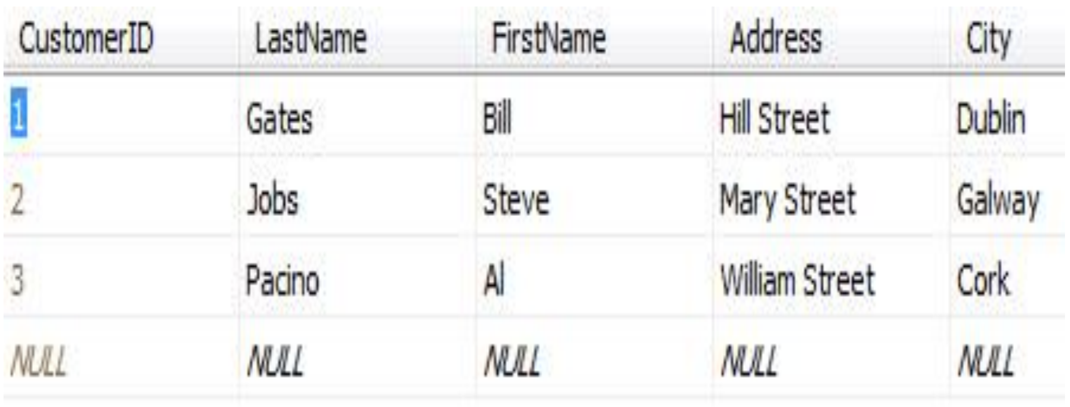
SQL INNER JOIN Syntax

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

PS: INNER JOIN is the same as JOIN.

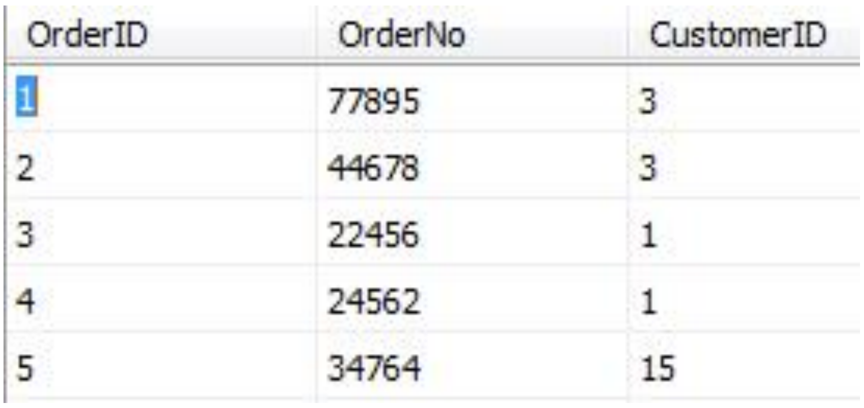
SQL INNER JOIN Example

The "Persons" table:



CustomerID	LastName	FirstName	Address	City
1	Gates	Bill	Hill Street	Dublin
2	Jobs	Steve	Mary Street	Galway
3	Pacino	Al	William Street	Cork
NULL	NULL	NULL	NULL	NULL

The "Orders" table:



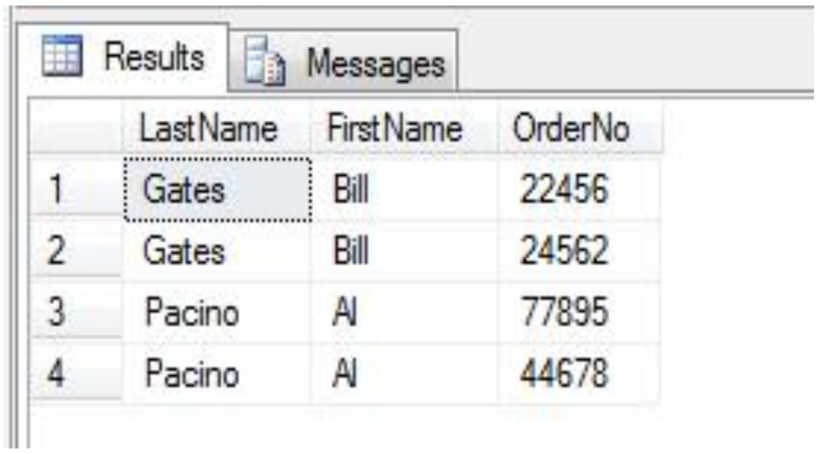
OrderID	OrderNo	CustomerID
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Now we want to list all the persons with any orders.

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
INNER JOIN Orders
ON Persons.CustomerID=Orders.CustomerID
ORDER BY Persons.LastName
```

The result-set will look like this:



	LastName	FirstName	OrderNo
1	Gates	Bill	22456
2	Gates	Bill	24562
3	Pacino	Al	77895
4	Pacino	Al	44678

The INNER JOIN keyword returns rows when there is at least one match in both tables. If there are rows in "Persons" that do not have matches in "Orders", those rows will NOT be listed.

SQL LEFT JOIN Keyword

The LEFT JOIN keyword returns all rows from the left table (table_name1), even if there are no matches in the right table (table_name2).

SQL LEFT JOIN Syntax

```
SELECT column_name(s)
FROM table_name1
LEFT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

PS: In some databases LEFT JOIN is called LEFT OUTER JOIN.

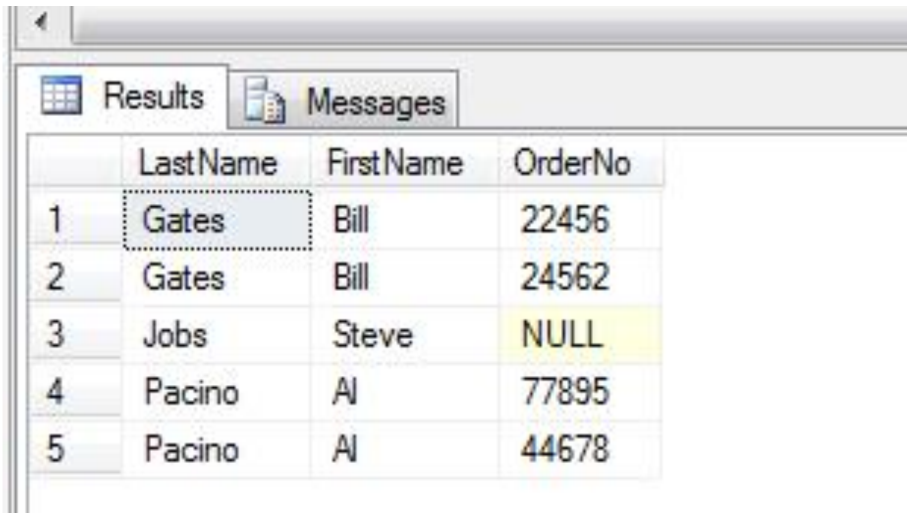
SQL LEFT JOIN Example

Now we want to list all the persons and their orders - if any, from the tables above.

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
LEFT JOIN Orders
ON Persons.CustomerID=Orders.CustomerID
ORDER BY Persons.LastName
```

The result-set will look like this:



	LastName	FirstName	OrderNo
1	Gates	Bill	22456
2	Gates	Bill	24562
3	Jobs	Steve	NULL
4	Pacino	Al	77895
5	Pacino	Al	44678

The LEFT JOIN keyword returns all the rows from the left table (Persons), even if there are no matches in the right table (Orders).

SQL RIGHT JOIN Keyword

The RIGHT JOIN keyword returns all the rows from the right table (table_name2), even if there are no matches in the left table (table_name1).

SQL RIGHT JOIN Syntax

```
SELECT column_name(s)
FROM table_name1
RIGHT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

PS: In some databases RIGHT JOIN is called RIGHT OUTER JOIN.

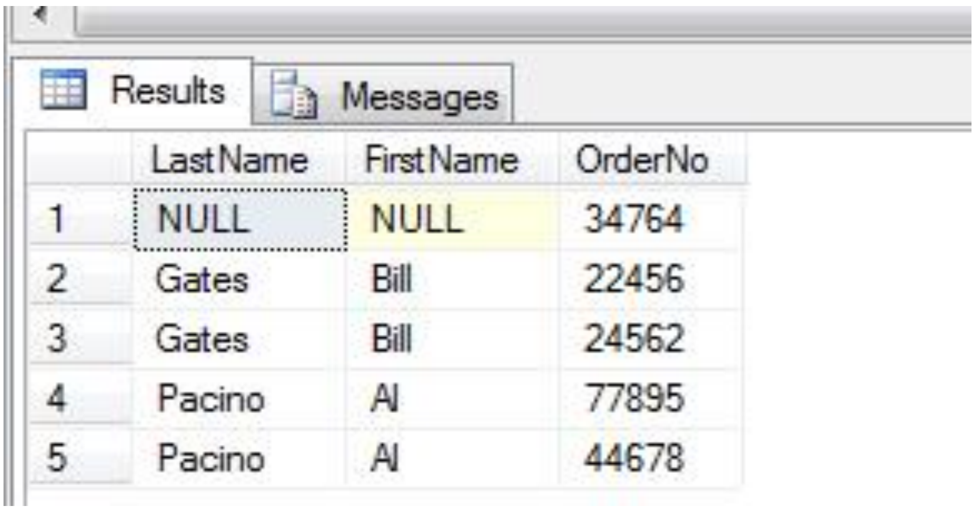
SQL RIGHT JOIN Example

Now we want to list all the orders with containing persons - if any, from the tables above.

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
RIGHT JOIN Orders
ON Persons.CustomerID =Orders.CustomerID
ORDER BY Persons.LastName
```

The result-set will look like this:



	LastName	FirstName	OrderNo
1	NULL	NULL	34764
2	Gates	Bill	22456
3	Gates	Bill	24562
4	Pacino	Al	77895
5	Pacino	Al	44678

The RIGHT JOIN keyword returns all the rows from the right table (Orders), even if there are no matches in the left table (Persons).

SQL FULL JOIN Keyword

The FULL JOIN keyword return rows when there is a match in one of the tables.

SQL FULL JOIN Syntax

```
SELECT column_name(s)
FROM table_name1
FULL JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

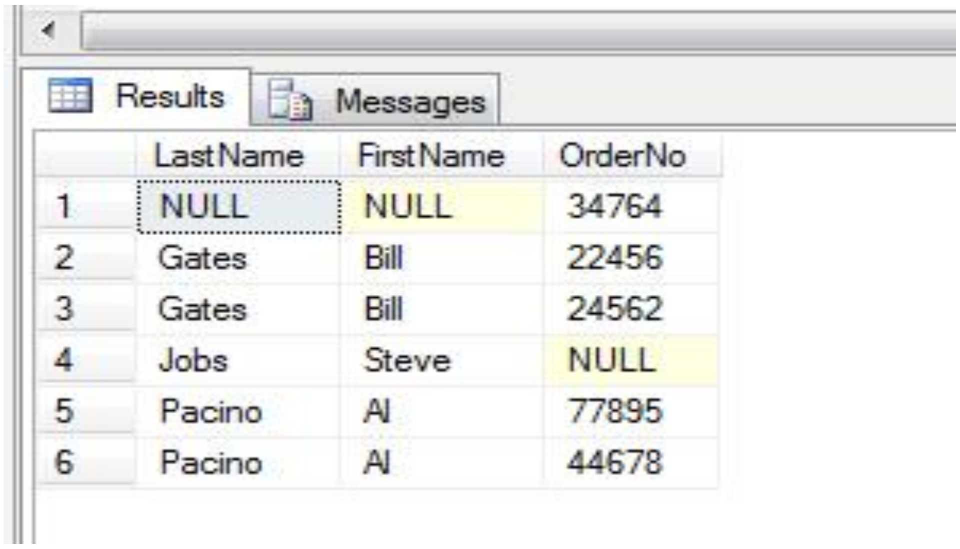
SQL FULL JOIN Example

Now we want to list all the persons and their orders, and all the orders with their persons.

We use the following SELECT statement:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
FULL JOIN Orders
ON Persons.CustomerID =Orders.CustomerID
ORDER BY Persons.LastName
```

The result-set will look like this:



	LastName	FirstName	OrderNo
1	NULL	NULL	34764
2	Gates	Bill	22456
3	Gates	Bill	24562
4	Jobs	Steve	NULL
5	Pacino	Al	77895
6	Pacino	Al	44678

The FULL JOIN keyword returns all the rows from the left table (Persons), and all the rows from the right table (Orders). If there are rows in "Persons" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Persons", those rows will be listed as well.

The SQL UNION Operator

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

Notice that each SELECT statement within the UNION must have the same number of columns. The columns must also have similar data types. Also, the columns in each SELECT statement must be in the same order.

SQL UNION Syntax

```
SELECT column_name(s) FROM table_name1
UNION
SELECT column_name(s) FROM table_name2
```

Note: The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL.

SQL UNION ALL Syntax

```
SELECT column_name(s) FROM table_name1
UNION ALL
SELECT column_name(s) FROM table_name2
```

PS: The column names in the result-set of a UNION are always equal to the column names in the first SELECT statement in the UNION.

SQL UNION Example

Look at the following tables:

"Employees_Norway":

E_ID	E_Name
01	Hansen, Ola
02	Svendson, Tove
03	Svendson, Stephen
04	Pettersen, Kari

"Employees_USA":

E_ID	E_Name
01	Turner, Sally
02	Kent, Clark
03	Svendson, Stephen
04	Scott, Stephen

Now we want to list **all the different** employees in Norway and USA.

We use the following SELECT statement:

```
SELECT E_Name FROM Employees_Norway
UNION
SELECT E_Name FROM Employees_USA
```

The result-set will look like this:

E_Name
Hansen, Ola
Svendson, Tove
Svendson, Stephen
Pettersen, Kari
Turner, Sally
Kent, Clark
Scott, Stephen

Note: This command cannot be used to list all employees in Norway and USA. In the example above we have two employees with equal names, and only one of them will be listed. The UNION command selects only distinct values.

SQL UNION ALL Example

Now we want to list **all** employees in Norway and USA:

```
SELECT E_Name FROM Employees_Norway
UNION ALL
SELECT E_Name FROM Employees_USA
```

Result

E_Name
Hansen, Ola
Svendson, Tove
Svendson, Stephen
Pettersen, Kari
Turner, Sally
Kent, Clark
Svendson, Stephen
Scott, Stephen