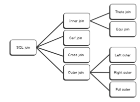
What is a SQL join?

A SQL Join is a special form of generating a meaningful data by combining multiple tables relate to each other using a "Key". Typically, relational tables must be designed with a unique column and this column is used to create relationships with one or more other tables. When you need a result-set that includes related rows from multiple tables, you'll need to use SQL join on this column

The various SQL join types are as follows

1. SQL inner join
   a. Equi join
   b. Non-equi join (Theta join)
2. SQL outer join
   a. SQL left join or left outer join
   b. SQL right join or right outer join
   c. SQL full join or full outer join
3. SQL cross join
4. SQL self join



> ***Note:*** *The keyword outer is optional. It means you can specify the keyword "outer" or not makes no difference to the query execution.*

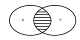
For example,

SQL join types

SQL inner join

The simplest and most common form of a join is the SQL inner join the default of the SQL join types used in most database management systems. It's the default SQL join you get when you use the join keyword by itself.

The result of the SQL inner join includes rows from both the tables where the join conditions are met.



Syntax:

```
SELECT ColumnList from LeftTable L
INNER join  RightTable R
ON L.Column=R.Column
```

> ***Note:*** *It is very easy to visualize a join query as a Venn diagram, where each of the tables is represented by intersecting shapes. The intersection of the shapes, where the tables overlap, are the rows where a condition is met. Unique columns (ID) are often used for this purpose, where the condition to be met is matching the ids of rows.*

Equi join:

An equi join is the most common form of SQL inner join used in practice. If the join contains an equality operator e.g. =, then it's an *equi-join.*

The following example returns all matching state names and stateProvinceIDs.

```
SELECT DISTINCT A.StateProvinceID,S.Name
FROM Person.Address A
inner join Person.StateProvince S
On A.StateProvinceID=S.StateProvinceID
```

	StateProvinceID	Name
1	1	Alberta
2	3	Alabama
3	6	Arizona
4	7	British Columbia
5	8	Bayern
6	9	California
7	10	Colorado
8	11	Connecticut
9	14	England
10	15	Florida
11	17	Georgia
12	19	Hessen
13	20	Hamburg
14	23	Idaho
15	24	Illinois
16	25	Indiana
17	27	Kentucky
18	30	Massachusetts
19	31	Manitoba

Theta join (Non-equi join):

In general, this a Theta join used to specify operators or conditions (the ON clause in SQL). In practice, this is a rarely used SQL join types. In most cases, the join will use a non-equality condition e.g. >

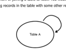
```
SELECT p1.FirstName, p2. FirstName
FROM PErson.Person p1
INNER join PErson.Person p2
ON len(p1.FirstName) > len(p2.FirstName);
```

Results | Messages

	FirstName	FirstName
3...	Marco	Anna
3...	Marco	Beth
3...	Marco	Dana
3...	Marco	Emma
3...	Marco	Eric
3...	Marco	Erik
3...	Marco	Gary
3...	Marco	Gina
3...	Marco	Jack
3...	Marco	Jill

SQL self join

A SQL Self join is a mechanism of joining a table to itself. You would use a self join when you wanted to create a result set joining records in the table with some other records from the same table.



For a SQL self join example, consider an Employee table where managers are listed because they are also employees, and we would like to take a look at a result set that returns all of the employees and indicating who their managers are

```
SELECT e.ename, e.empno, m.ename as manager, e.mgr
FROM
    emp e, emp m
WHERE e.mgr = m.empno
```

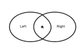
Results | Messages

	ename	empno	manager	mgr
1	SMITH	7369	FORD	7902
2	ALLEN	7499	BLAKE	7698
3	WARD	7521	BLAKE	7698
4	JONES	7566	KING	7839
5	MARTIN	7654	BLAKE	7698
6	BLAKE	7698	KING	7839
7	CLARK	7782	KING	7839
8	SCOTT	7788	JONES	7566
9	TURNER	7844	BLAKE	7698
10	ADAMS	7876	SCOTT	7788
11	JAMES	7900	BLAKE	7698
12	FORD	7902	JONES	7566
13	MILLER	7934	CLARK	7782

SQL cross join

A CROSS join returns all rows for all possible combinations of two tables. It generates all the rows from the left table which is then combined with all the rows from the right table. This type of join is also known as a Cartesian product(A*B).

For example, if the left table has 100 rows and the right table has 100 then the cross join result will yield 10,000 rows.



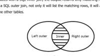

```
SELECT e.BusinessEntityID, d.Name AS Department
FROM HumanResources.Employee AS e
CROSS join HumanResources.Department AS d
```

Results | Messages

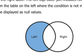
	BusinessEntityID	Department
1	263	Document Control
2	78	Document Control
3	242	Document Control
4	125	Document Control
5	278	Document Control
6	239	Document Control
7	184	Document Control
8	87	Document Control
9	174	Document Control
10	284	Document Control
11	211	Document Control
12	115	Document Control
13	241	Document Control
14	7	Document Control
15	222	Document Control
16	232	Document Control
17	280	Document Control
18	287	Document Control
19	51	Document Control

SQL outer join

On joining tables with a SQL inner join, the output returns only matching rows from both the tables. When using a SQL outer join, not only it will list the matching rows, it will also list the unmatched rows from the other tables.



A **SQL left outer join** will return all the records from the left table in the join clause, regardless of matching records in the right table. The left SQL outer join includes rows where the condition is met plus all the rows from the table on the left where the condition is not met. Fields from the right table with no match will be displayed as null values.



Syntax:

```
SELECT ColumnList from LeftTable L
LEFT join  RightTable R
ON L.Column=R.Column
Where R.Column is NULL
```

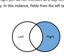
The following example joins two tablesProduct and SalesOrderDetail on ProductID and preserves the unmatched rows from the left table. The Product table is matched with the SalesOrderDetail table on the ProductID columns in each table. All products, ordered and not ordered, appear in the result set.

```
SELECT p.Name, so.SalesOrderID
FROM Production.Product  p
LEFT OUTER join Sales.SalesOrderDetail so
ON p.ProductID = so.ProductID
ORDER BY p.Name ;
```

Results | Messages

	Name	SalesOrderID
1	Adjustable Race	NULL
2	All-Purpose Bike Stand	51179
3	All-Purpose Bike Stand	51488
4	All-Purpose Bike Stand	51520
5	All-Purpose Bike Stand	51558
6	All-Purpose Bike Stand	51882
7	All-Purpose Bike Stand	51903
8	All-Purpose Bike Stand	51970
9	All-Purpose Bike Stand	52010
10	All-Purpose Bike Stand	52032
11	All-Purpose Bike Stand	52095
12	All-Purpose Bike Stand	52309
13	All-Purpose Bike Stand	52357
14	All-Purpose Bike Stand	52372
15	All-Purpose Bike Stand	52380
16	All-Purpose Bike Stand	52417
17	All-Purpose Bike Stand	52558
18	All-Purpose Bike Stand	52620
19	All-Purpose Bike Stand	52705

A **right outer join** will return all the records in the right table in the join clause, regardless of matching records in the left table. Using the right SQL outer join includes all the rows from the table on the right. The right SQL outer join is considered a special case and many databases don't support right joins. Generally, a SQL right join can be rewritten as a SQL left join by simply changing the order of the tables in the query. In this instance, fields from the left table with no match will display null values



Syntax:

```
SELECT ColumnList from LeftTable L
RIGHT join  RightTable R
ON L.Column=R.Column
Where L.Column is NULL
```

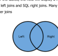
The following example joins two tables on TerritoryID(SalesTerritory) and preserves the unmatched rows from the right table(SalesPerson). The SalesTerritory table is matched with the SalesPerson table on the TerritoryID column in each table. All salespersons appear in the result set, whether or not they are assigned a territory.

```
SELECT s.Name AS Territory, p.BusinessEntityID
FROM Sales.SalesTerritory  s
RIGHT OUTER join Sales.SalesPerson p
ON s.TerritoryID = p.TerritoryID ;
```

Results | Messages

	Territory	BusinessEntityID
1	NULL	274
2	Northeast	275
3	Southwest	276
4	Central	277
5	Canada	278
6	Southeast	279
7	Northwest	280
8	Southwest	281
9	Canada	282
10	Northwest	283
11	Northwest	284
12	NULL	285
13	Australia	286
14	NULL	287
15	Germany	288
16	United Kingdom	289
17	France	290

A **SQL outer join**, as you might expect by now, will return all the rows in both tables. When rows don't have a match in one of the tables, the field will display a null value. A full SQL outer join combines the effects of the SQL left joins and SQL right joins. Many databases do not support the implementation of full SQL outer joins



Syntax:

```
SELECT ColumnList from LeftTable L
FULL OUTER join  RightTable R
ON L.Column=R.Column
```

The following example returns the name of the product name any corresponding sales orders in the SalesOrderDetail table from the AdventureWorks2014 database. It also returns any sales orders that have no product listed in the Product table, and any products with a sales order other than the one listed in the Product table.

```
SELECT p.Name, s.SalesOrderID
FROM Production.Product p
FULL OUTER join Sales.SalesOrderDetail s
ON p.ProductID = s.ProductID
ORDER BY p.Name ;
```

Results | Messages

	Name	SalesOrderID
1	Adjustable Race	NULL
2	All-Purpose Bike Stand	51179
3	All-Purpose Bike Stand	51488
4	All-Purpose Bike Stand	51520
5	All-Purpose Bike Stand	51558
6	All-Purpose Bike Stand	51882
7	All-Purpose Bike Stand	51903
8	All-Purpose Bike Stand	51970
9	All-Purpose Bike Stand	52010
10	All-Purpose Bike Stand	52032
11	All-Purpose Bike Stand	52095
12	All-Purpose Bike Stand	52309
13	All-Purpose Bike Stand	52357
14	All-Purpose Bike Stand	52372
15	All-Purpose Bike Stand	52380
16	All-Purpose Bike Stand	52417
17	All-Purpose Bike Stand	52558
18	All-Purpose Bike Stand	52620
19	All-Purpose Bike Stand	52705