Aggregate Function

COUNT()

The COUNT function returns the number of values present in a particular column.

Syntax:
SELECT ***COUNT***(column_name)
FROM table_name
WHERE condition;

SUM()

The SUM function returns the sum of values in the specified column.

Syntax:
SELECT ***SUM***(column_name)
FROM table_name
WHERE condition;

AVG()

The AVG function returns the average of values in a specified column.

Syntax:
SELECT ***AVG***(column_name)
FROM table_name
WHERE condition;

MIN()

The MIN function returns the smallest value of a specified column.

Syntax:
SELECT ***MIN***(column_name)
FROM table_name
WHERE condition;

MAX()

The MAX function returns the maximum value of a specified column.

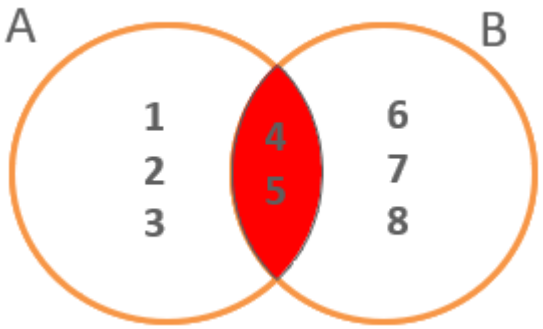
Syntax:
SELECT ***MAX***(column_name)
FROM table_name
WHERE condition;

Set Operators

UNION

The UNION operator returns all rows from both tables, after eliminating duplicates.

Syntax:
SELECT column1 FROM table 1
UNION
SELECT column 2 FROM table 2;



The result of listing all elements in A and B eliminating duplicates is {1, 2, 3, 4, 5, 6, 7, 8}.

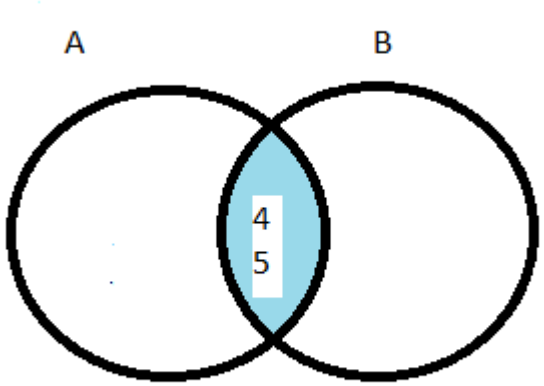
If you joined A and B you would get only {4, 5}. You would have to perform a full outer join to get the same list as above.

INTERSECTION

The INTERSECT operator returns all rows common to both tables.

Syntax:

```
SELECT a_id FROM a INTERSECT
SELECT b_id FROM b;
```



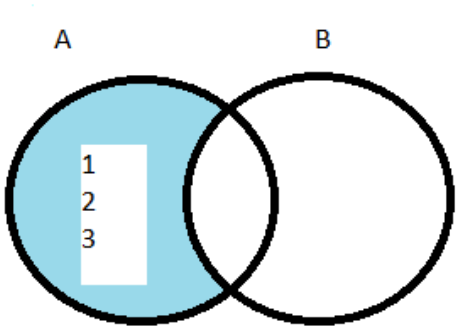
The result of listing all elements found in both A and B is {4, 5}.

EXCEPT

EXCEPT returns distinct rows from the left input query that aren't output by the right input query.

Syntax:

```
SELECT a_id FROM a EXCEPT
SELECT b_id FROM b;
```



The result of listing all elements found in A but not B is {1, 2, 3}.
The result of B MINUS A would give {6, 7, 8}.

SQL Joins

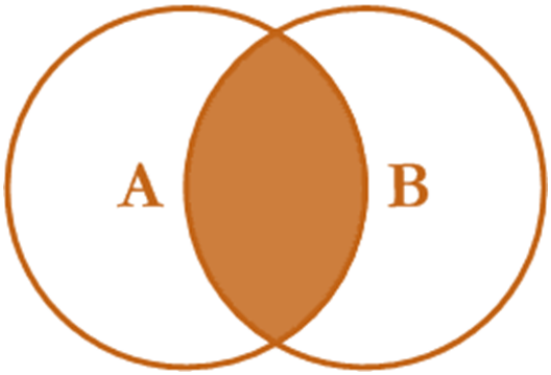
An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

Types of SQL join operations

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- OUTER JOIN

INNER JOIN

Inner join only takes those rows from the Cartesian Product Table where the join elements match fully.
Inner join condition will create result by combining all rows from both tables where the value of common filled will be the same.



Example:

Product

PID	PName
1	Shirt
2	Punjabi
3	Lungi

Sale

SID	ProductID	Price
101	1	1000
102	2	800
103	5	400
104	2	600

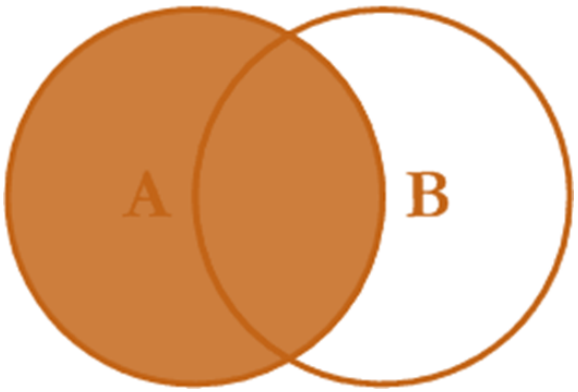
Query: Select * from Product p ***Inner join*** Sale s on p.PID = s.ProductID;

PID	Pname	SID	ProductID	Price
1	Shirt	101	1	1000
2	Panjabi	102	2	800

2	Panjabi	104	2	600

LEFT JOIN

Left join takes those rows which are in the inner join output. And it also looks for the rows in the left table which are not in the inner join output. The rows are added to OUTPUT with null in right columns.



Example:

Product

PID	**PName**
1	Shirt
2	Punjabi
3	Lungi

Sale

SID	**ProductID**	**Price**
101	1	1000
102	2	800
103	5	400
104	2	600

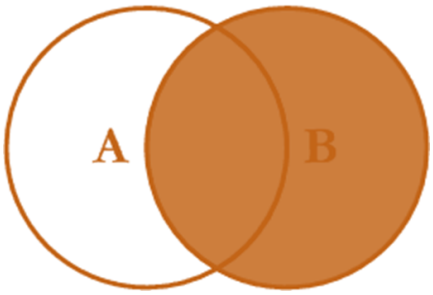
Query: Select * from Product p ***Left join*** Sale s on p.PID = s.ProductID;

So the output of the join table is shown below.
In the left table Product |PID = 3 | Pname = Lungi|
row could not be joined with any row of Sale table. So it is added with null value in right columns

PID	Pname	SID	ProductID	Price
1	Shirt	101	1	1000
2	Panjabi	102	2	800
2	Panjabi	104	2	600
3	Lungi	null	null	null

RIGHT JOIN

Right join takes those rows which are in the inner join output. Also looks for the rows in the right table which are not in the inner join output. The rows are added to OUTPUT with null in the left columns.



Example:

Product

PID	**PName**
1	Shirt
2	Punjabi
3	Lungi

Sale

SID	**ProductID**	**Price**
101	1	1000
102	2	800
103	5	400
104	2	600

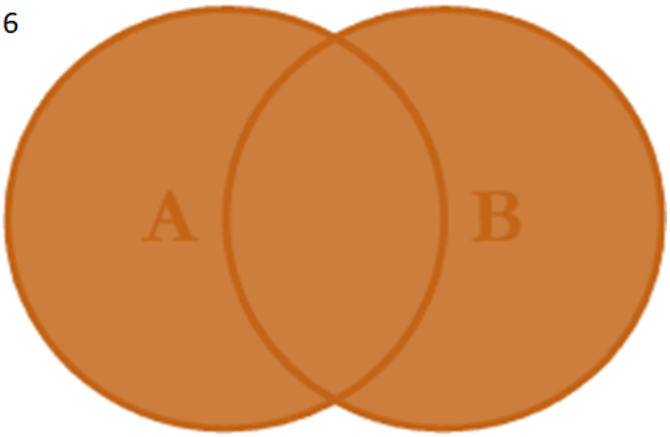
Query: Select * from Product p ***Right join*** Sale s on p.PID = s.ProductID;

So the output of the join table is shown below.
In the right table Sale |SID = 103|ProductID = 5|Price = 400|row could not be joined with any row of Product table. So it is added with null value in the left columns.

PID	Pname	SID	ProductID	Price
1	Shirt	101	1	1000
2	Panjabi	102	2	800
2	Panjabi	104	2	600
null	null	103	5	400

OUTER JOIN

Aside from inner join output Outer join looks for the rows in the left table which are not in inner join output. The rows are added to OUTPUT with null in right columns. Similarly the rows from the right table not in the inner join output are added to OUTPUT with null values in left columns.



Example:

Product

PID	PName
1	Shirt
2	Punjabi
3	Lungi

Sale

SID	ProductID	Price
101	1	1000
102	2	800
103	5	400
104	2	600

Query: Select * from Product p ***Outer join*** Sale s on p.PID = s.ProductID

PID	Pname	SID	ProductID	Price
1	Shirt	101	1	1000
2	Punjabii	102	2	800
2	Punjabi	104	2	600
3	Lungi	null	null	null
null	null	103	5	400

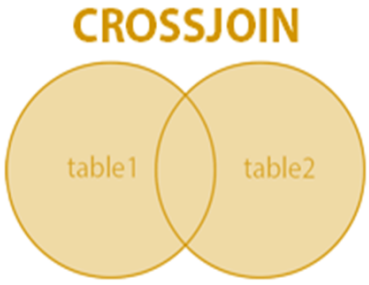
In the 4th row there is no joinable row on the right. So the right values are null. Similarly in the 5th row there is no joinable row in the left. So left values are null.

CROSS JOINS

The CROSS JOIN is used to generate a paired combination of each row of the first table with each row of the second table. This join type is also known as cartesian join.
A cross join produces a cartesian product between the two tables, returning all possible combinations of all rows.

Syntax:

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



Example:

Product

PID	**PName**
1	Shirt
2	Punjabi
3	Lungi

Sale

SID	**ProductID**	**Price**
101	1	1000
102	2	800
103	5	400
104	2	600

Query: Select * from table 1 ***CROSS JOIN*** table 2;

PID	Pname	SID	ProductID	Price
1	Shirt	101	1	1000
1	Shirt	102	2	800
1	Shirt	103	5	400
1	Shirt	104	2	600
2	Panjabi	101	1	1000
2	Panjabi	102	2	800
2	Panjabi	103	5	400
2	Panjabi	104	2	600
3	Lungi	101	1	1000

3	Lungi	102	2	800
3	Lungi	103	5	400
3	Lungi	104	2	600