Joins in SQL

Joins on Multiple Tables:

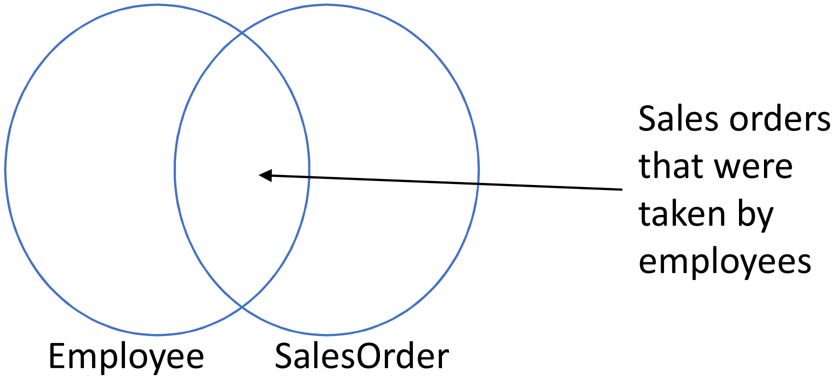
- SQL provides a convenient operation to retrieve information from multiple tables.

- This operation is called **join**.

- The join operation will **combine** the tables into one large table with all possible combinations (Math: Cartesian Product), and then it will filter the rows of this combined table to yield useful information.

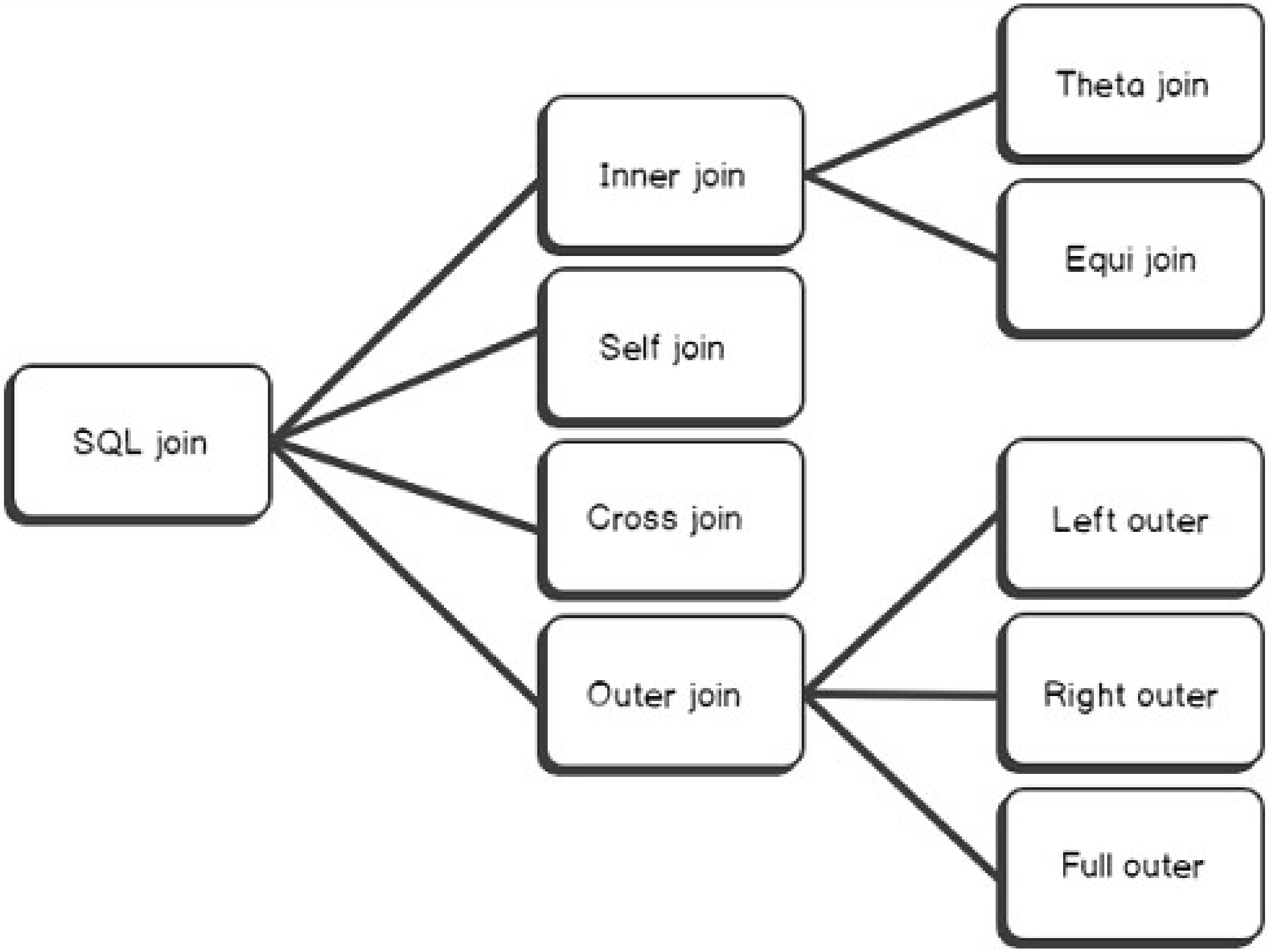
Join Concepts

- Combine rows from multiple tables by specifying matching criteria
  - Usually based on primary key – foreign key relationships
  - For example, return rows that combine data from the **Employee** and **SalesOrder** tables by matching the **Employee.EmployeeID** primary key to the **SalesOrder.EmployeeID** foreign key
- It helps to think of the tables as sets in a Venn diagram

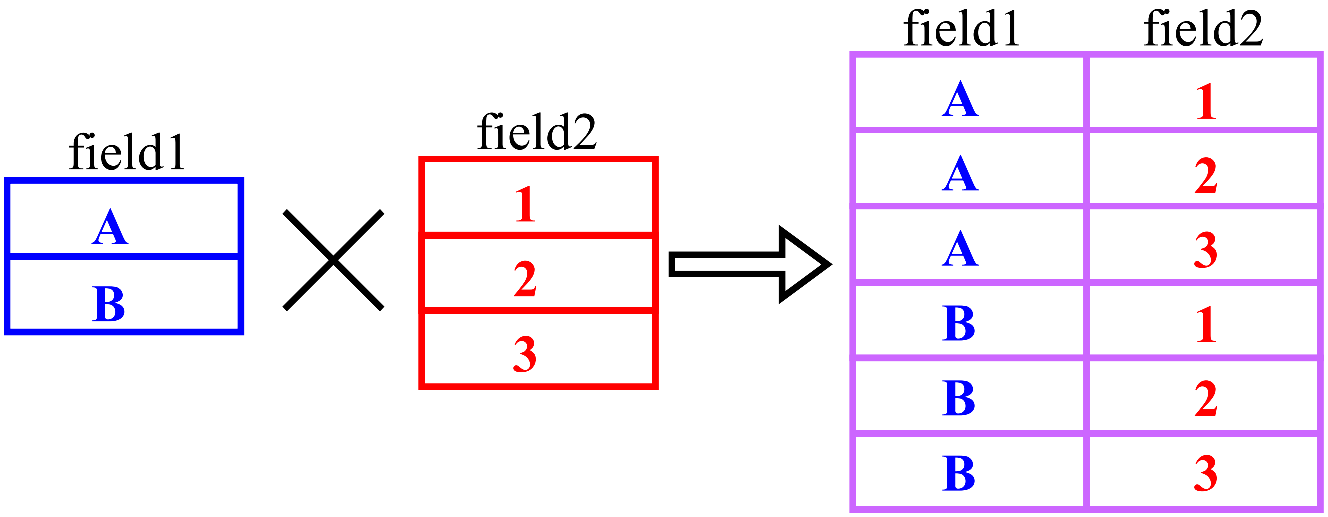


Different type of Joins

1. Cross Join
2. Inner Join
    - i. Equi/Equality join (‘=‘)
    - ii. Natural join
    - iii. Theta join (or non-equi join (<, <=, >, >=, <>))
3. Outer Join
    - i. Left outer join
    - ii. Right outer join
    - iii. Full outer join
4. Self Join
5. Semi Join


SQL join
Inner join
Self join
Cross join
Outer join
Theta join
Equi join
Left outer
Right outer
Full outer

Multiple Tables:



field1
A
B

×

field2
1
2
3

⇒

field1	field2
A	1
A	2
A	3
B	1
B	2
B	3

What kind of Join is this?

```
SELECT *
 FROM Students S ?? Enrolled E;
```

S

S.name	S.sid
Jones	11111
Smith	22222

E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150

S.name	S.sid	E.sid	E.classid
Jones	11111	11111	History105
Jones	11111	11111	DataScience194
Jones	11111	22222	French150
Smith	22222	11111	History105
Smith	22222	11111	DataScience194
Smith	22222	22222	French150

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**.

```
SELECT *
 FROM Students S CROSS JOIN Enrolled E;
```

S

S.name	S.sid
Jones	11111
Smith	22222

E

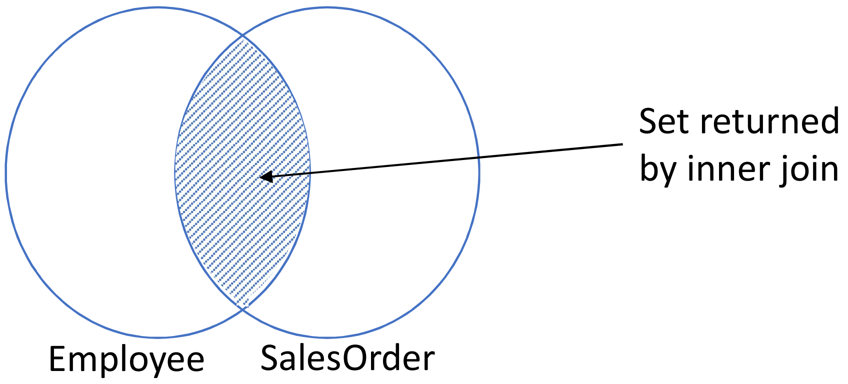
E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150

S.name	S.sid	E.sid	E.classid
Jones	11111	11111	History105
Jones	11111	11111	DataScience194
Jones	11111	22222	French150
Smith	22222	11111	History105
Smith	22222	11111	DataScience194
Smith	22222	22222	French150

Inner Joins

- Return only rows where a match is found in both input tables
- Match rows based on attributes supplied in predicate
- If join predicate operator is =, then it known as equi-join

```
SELECT emp.FirstName, ord.Amount
FROM Employee  emp
[INNER] JOIN SalesOrder  ord
ON emp.EmployeeID = ord.EmployeeID
```



SQL Inner Joins (Equi join)

```
SELECT S.name, E.classid
 FROM Students S (INNER) JOIN Enrolled E
 ON S.sid=E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222
Brown	33333

E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150
44444	English10

S.name	E.classid
Jones	History105
Jones	DataScience194
Smith	French150

Note the previous version of this query (with no join keyword) is an “Implicit join”

SQL Inner Joins (Equi join)

```
SELECT S.name, E.classid
 FROM Students S (INNER) JOIN Enrolled E
 ON S.sid=E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222
Brown	33333

E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150
44444	English10

S.name	E.classid
Jones	History105
Jones	DataScience194
Smith	French150

Unmatched keys

Joins and Inference

- Chaining relations together is the basic inference method in relational DBs. It produces new relations (effectively new facts) from the data:

```
SELECT S.name, M.mortality
 FROM Students S, Mortality M
 WHERE S.Race=M.Race
```

S

Name	Race
Socrates	Man
Thor	God
Barney	Dinosaur
Blarney stone	Stone

M

Race	Mortality
Man	Mortal
God	Immortal
Dinosaur	Mortal
Stone	Non-living

Joins and Inference

- Chaining relations together is the basic inference method in relational DBs. It produces new relations (effectively new facts) from the data:

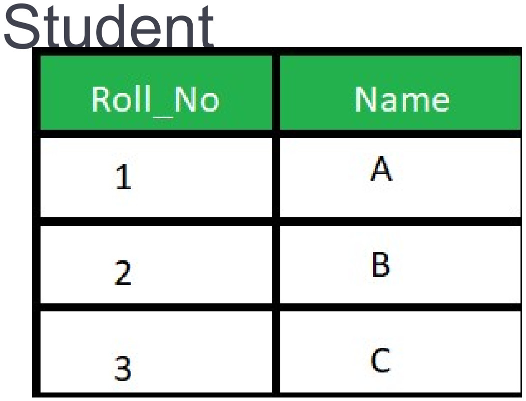
```
SELECT S.name, M.mortality
 FROM Students S, Mortality M
 WHERE S.Race=M.Race
```

Name	Mortality
Socrates	Mortal
Thor	Immortal
Barney	Mortal
Blarney stone	Non-living

Natural Join

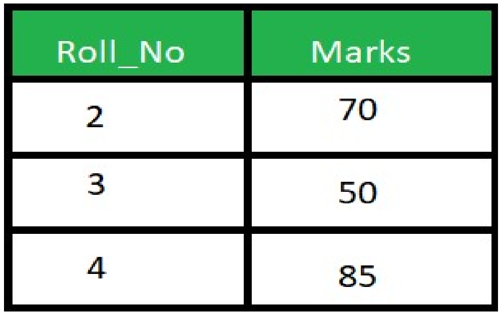
Natural Join joins two tables based on same attribute name and datatypes. The resulting table will contain all the attributes of both the table but keep only one copy of each common column.

Student



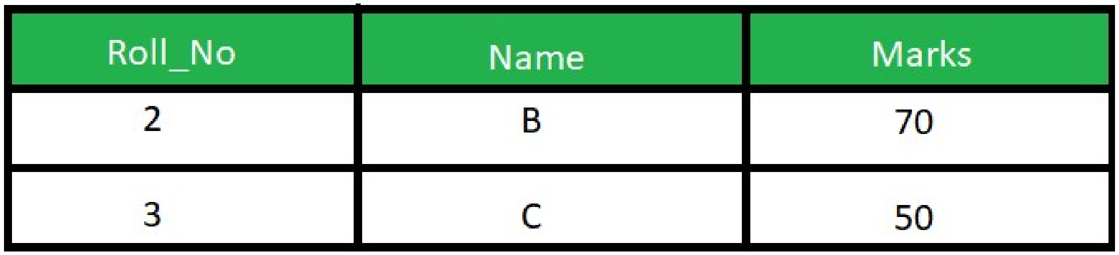
Roll_No	Name
1	A
2	B
3	C

Mark



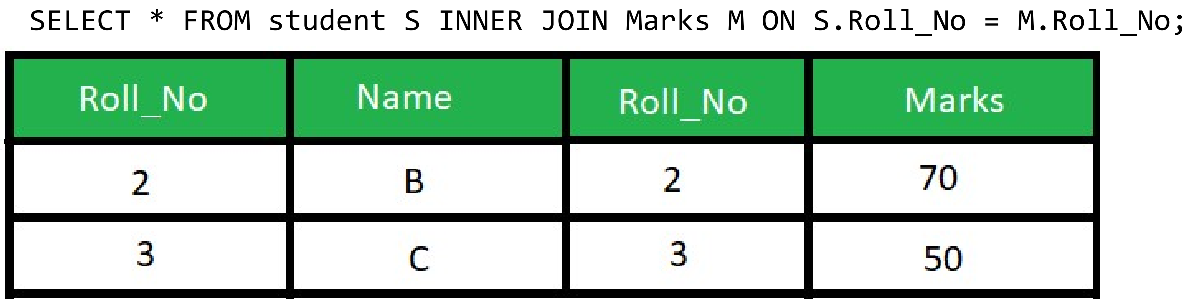
Roll_No	Marks
2	70
3	50
4	85

Select * from Student natural join Mark;



Roll_No	Name	Marks
2	B	70
3	C	50

Difference between Equi join and Natural join



```
SELECT * FROM student S INNER JOIN Marks M ON S.Roll_No = M.Roll_No;
```

Roll_No	Name	Roll_No	Marks
2	B	2	70
3	C	3	50

1.	Natural Join joins two tables based on same attribute name and datatypes.	Inner Join joins two table on the basis of the column which is explicitly specified in the ON clause.
2.	In Natural Join, The resulting table will contain all the attributes of both the tables but keep only one copy of each common column	In Inner Join, The resulting table will contain all the attribute of both the tables including duplicate columns also
3.	In Natural Join, If there is no condition specifies then it returns the rows based on the common column	In Inner Join, only those records will return which exists in both the tables

4.	SYNTAX:
	SELECT *
	FROM table1 NATURAL JOIN table2;
		SYNTAX:
		SELECT *
		FROM table1 INNER JOIN table2 ON table1.Column_Name = table2.Column_Name;

What kind of Join is this?

```
SELECT *
 FROM Students S, Enrolled E
 WHERE S.sid <= E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222

E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150

S.name	S.sid	E.sid	E.classid
Jones	11111	11111	History105
Jones	11111	11111	DataScience194
Jones	11111	22222	French150
Smith	22222	22222	French150

Theta Joins (<, <=, >, >=, <>)

```
SELECT *
 FROM Students S, Enrolled E
 WHERE S.sid <= E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222

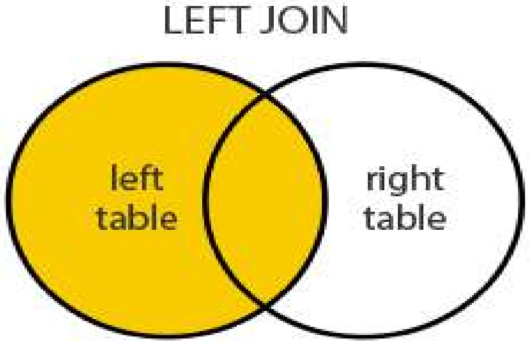
E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150

S.name	S.sid	E.sid	E.classid
Jones	11111	11111	History105
Jones	11111	11111	DataScience194
Jones	11111	22222	French150
Smith	22222	22222	French150

Left Outer Join

A LEFT JOIN performs a join starting with the first (left-most) table. Then, any matched records from the second table (right-most) will be included. LEFT JOIN and LEFT OUTER JOIN are the same. The result is 0 records from the right side, if no there is no match.



```
SELECT column-names
FROM table-name1 LEFT OUTER JOIN table-name2
ON column-name1 = column-name2
```

What kind of Join is this?

```
SELECT S.name, E.classid
 FROM Students S ?? Enrolled E
 ON S.sid=E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222
Brown	33333

E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150
44444	English10

S.name	E.classid
Jones	History105
Jones	DataScience194
Smith	French150
Brown	NULL

Query: List all students name and their classid irrespective whether they enrolled for that class or not.

```
SELECT S.name, E.classid
 FROM Students S LEFT OUTER JOIN Enrolled E
 ON S.sid=E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222
Brown	33333

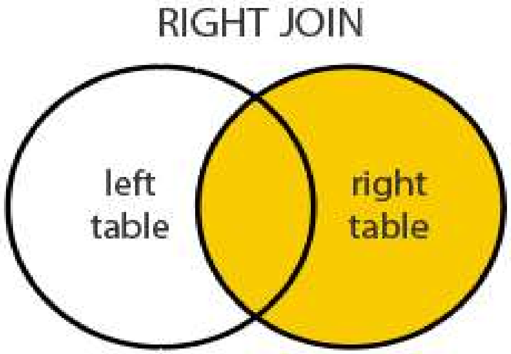
E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150
44444	English10

S.name	E.classid
Jones	History105
Jones	DataScience194
Smith	French150
Brown	NULL

Right outer join

A RIGHT JOIN performs a join starting with the second (right-most) table and then any matching first (left-most) table records. RIGHT JOIN and RIGHT OUTER JOIN are the same. The result is 0 records from the left side, if no there is no match.



SELECT column-names

FROM table-name1 RIGHT OUTER JOIN table-name2

ON column-name1 = column-name2

What kind of Join is this?

```
SELECT S.name, E.classid
 FROM Students S ?? Enrolled E
 ON S.sid=E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222
Brown	33333

E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150
44444	English10

S.name	E.classid
Jones	History105
Jones	DataScience194
Smith	French150
NULL	English10

Query: List all students name and their classid irrespective whether for a class any student enrolled or not.

```
SELECT S.name, E.classid
 FROM Students S RIGHT OUTER JOIN Enrolled E
 ON S.sid=E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222
Brown	33333

E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150
44444	English10

S.name	E.classid
Jones	History105
Jones	DataScience194
Smith	French150
NULL	English10

SQL Joins

```
SELECT S.name, E.classid
 FROM Students S ? JOIN Enrolled E
 ON S.sid=E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222
Brown	33333

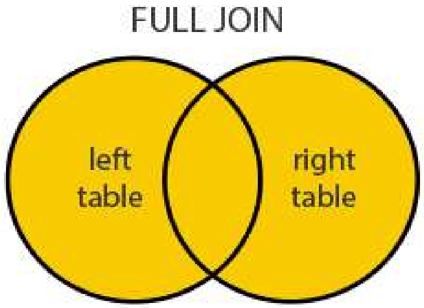
E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150
44444	English10

S.name	E.classid
Jones	History105
Jones	DataScience194
Smith	French150
NULL	English10
Brown	NULL

Full outer join

FULL JOIN returns all matching records from both tables whether the other table matches or not. Be aware that a FULL JOIN can potentially return very large datasets. These two: FULL JOIN and FULL OUTER JOIN are the same.



```
SELECT column-names
FROM table-name1 FULL OUTER JOIN table-name2
ON column-name1 = column-name2
```

Query: List all students name and their classid irrespective whether a student enrolled for a class or not, and irrespective whether for a class any student enrolled or not.

```
SELECT S.name, E.classid
 FROM Students S FULL OUTER JOIN Enrolled E
 ON S.sid=E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222
Brown	33333

E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150
44444	English10

S.name	E.classid
Jones	History105
Jones	DataScience194
Smith	French150
NULL	English10
Brown	NULL

Self Join

- A self JOIN occurs when a table takes a 'selfie', that is, it JOINs with itself. A self JOIN is a regular join but the table that it joins to is itself.
- **Joining** a table with itself means that each row of the table is combined with itself and with every other row of the table. SELF JOINs are also useful for comparisons within a table.

```
SELECT column-names
FROM table-name T1 JOIN table-name T2
WHERE condition
```

Emp-id	Name	Manager-id
111	Ravi	NULL
112	Rakul	111
113	Pal	111
114	Yuvi	113
115	Juri	112

Return all employees and the name of the employee’s manager

```
SELECT emp.FirstName AS Employee,  man.FirstName AS Manager
FROM Employee  emp JOIN Employee  man
ON emp.ManagerID = man.EmployeeID;
```

```
Output:        Employee                  Manager
               Ravi                        NULL
               Rakul                       Ravi
               Pal                         Ravi
               Yuvi                         Pal
               Juri                         Rakul
```

Semi-Join

- Semi join is a type of join whose result-set contains only the columns from one of the "*semi-joined*" tables. Each row from the first table(left table if Left Semi Join) will be returned maximum once, if matched in the second table. The duplicate rows from the first table will be returned, if matched once in the second table. A distinct row from the first table will be returned no matter how many times matched in a second table.
- The essential differences between a semi join and a regular join are:
- Semi join either returns each row from input A, or it does not. No row duplication can occur.
- Regular join duplicates rows if there are multiple matches on the join predicate.
- Semi join is defined to only return columns from input A.
- Regular join may return columns from either (or both) join inputs.

What kind of Join is this?

```
SELECT S.name, E.classid
 FROM Students S ?? Enrolled E
 ON S.sid=E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222
Brown	33333

E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150
44444	English10

S.name	E.classid
Jones	History105
Smith	French150

SQL Joins

```
SELECT S.name, E.classid
 FROM Students S LEFT SEMI JOIN Enrolled E
 ON S.sid=E.sid
```

S

S.name	S.sid
Jones	11111
Smith	22222
Brown	33333

E

E.sid	E.classid
11111	History105
11111	DataScience194
22222	French150
44444	English10

S.name	E.classid
Jones	History105
Smith	French150

Semi-join Vs. Anti semi-join

While a semi-join returns one copy of each row in the first table for which at least one match is found, an anti semi-join returns one copy of each row in the first table for which no match is found.

END