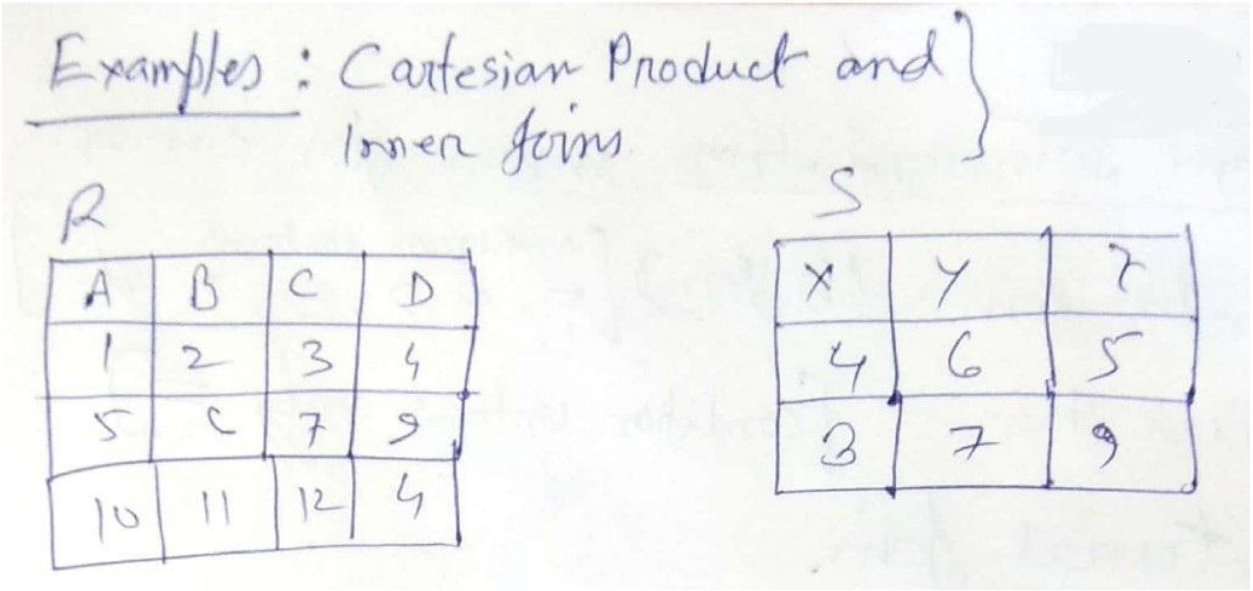
Relational Algebra Operations

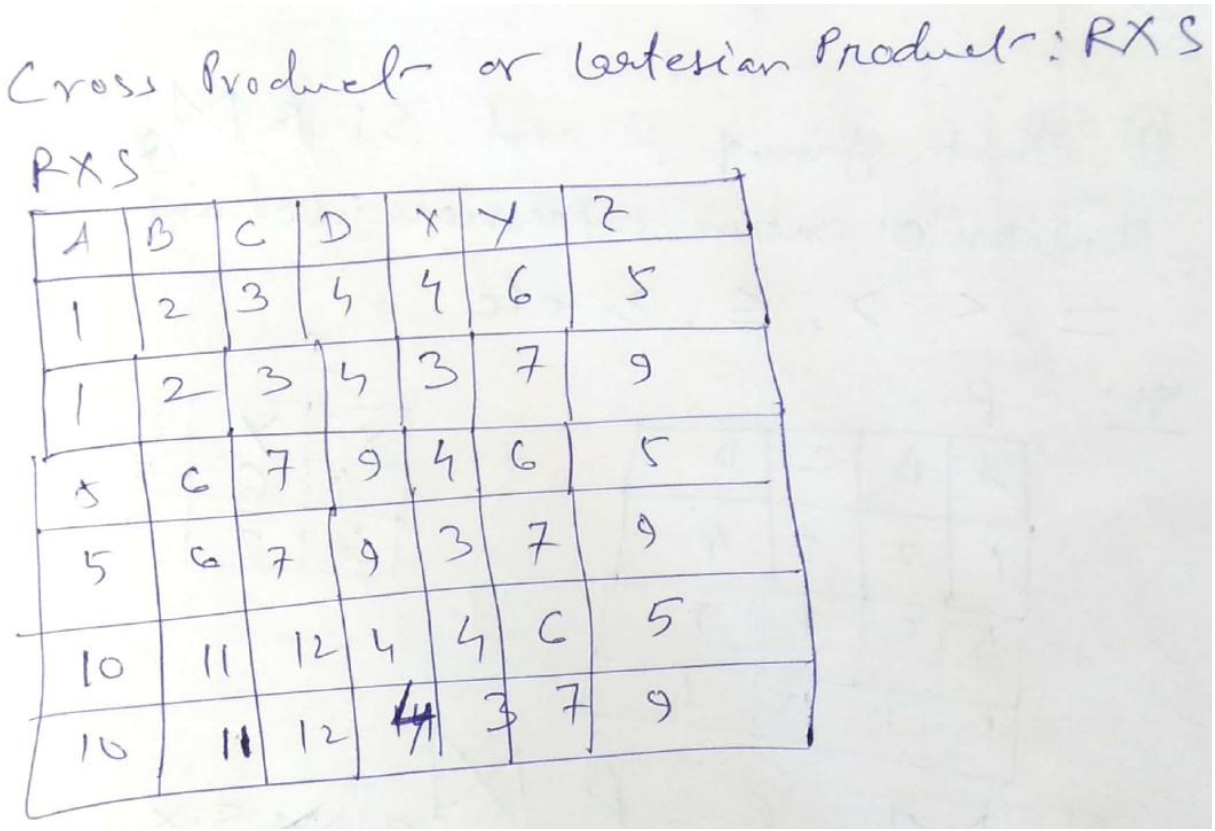
CARTESIAN PRODUCT and Different types of JOIN operations

CARTESIAN PRODUCT operation—also known as **CROSS PRODUCT** or **CROSS JOIN**—which is denoted by ×. This is also a binary set operation, but the relations on which it is applied do *not* have to be union compatible. In its binary form, this set operation produces a new element by combining every member (tuple) from one relation (set) with every member (tuple) from the other relation (set).

In general, the result of $R(A_1, A_2, ..., A_n) \times S(B_1, B_2, ..., B_m)$ is a relation Q with degree $n + m$ attributes $Q(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)$, in that order.

The resulting relation Q has one tuple for each combination of tuples—one from R and one from S. Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then $R \times S$ will have $n_R * n_S$ tuples.





Binary Relational Operation: JOIN (or INNER JOIN)

The **JOIN** operation, denoted by ⋈ , is used to combine *related tuples* from two relations into single "longer" tuples. This operation is very important for any relational database with more than a single relation because it allows us to process relationships among relations.

The general form of a JOIN operation on two relations $R(A_1, A_2, ..., A_n)$ and $S(B_1, B_2, ..., B_m)$ is

$$R \bowtie_{\text{<join condition>}} S$$

The result of the JOIN is a relation Q with $n + m$ attributes $Q(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)$ in that order; Q has one tuple for each combination of

tuples—one from *R* and one from *S*—*whenever the combination satisfies the join condition*. This is the main difference between CARTESIAN PRODUCT and JOIN.

This type of join, where tuples with no match are eliminated, is also known as an **inner join**.

In JOIN, only combinations of tuples *satisfying the join condition* appear in the result, whereas in the CARTESIAN PRODUCT *all* combinations of tuples are included in the result. The join condition is specified on attributes from the two relations *R* and *S* and is evaluated for each combination of tuples. Each tuple combination for which the join condition evaluates to TRUE is included in the resulting relation *Q as a single combined tuple*.

- **Types of JOIN operations**
  - **Inner join**
  - **Outer join**

- **Types of Inner join**
  - **Theta JOIN**
  - **EQUI JOIN**
  - **Natural JOIN**

Theta JOIN

A general join condition is of the form <condition> **AND** <condition> **AND**...**AND** <condition> where, each <condition> is of the form $A_i\ \theta\ B_j$, A_i is an attribute of *R*, B_j is an attribute of *S*, A_i and B_j have the same domain, and θ (theta) is one of the comparison operators {=, <, ≤, >, ≥, ≠}. A JOIN operation with such a general join condition is called a **THETA JOIN**.

eg:

R

A	B	C	D
1	2	3	4
5	6	7	9
10	11	12	4

S

X	Y	Z
4	6	5
3	7	9

$R \bowtie_{R.D > S.X} S = R \bowtie S$

Result-

A	B	C	D	X	Y	Z
1	2	3	4	3	7	9
5	6	7	9	4	6	5
5	6	7	9	3	7	9
10	11	12	4	3	7	9

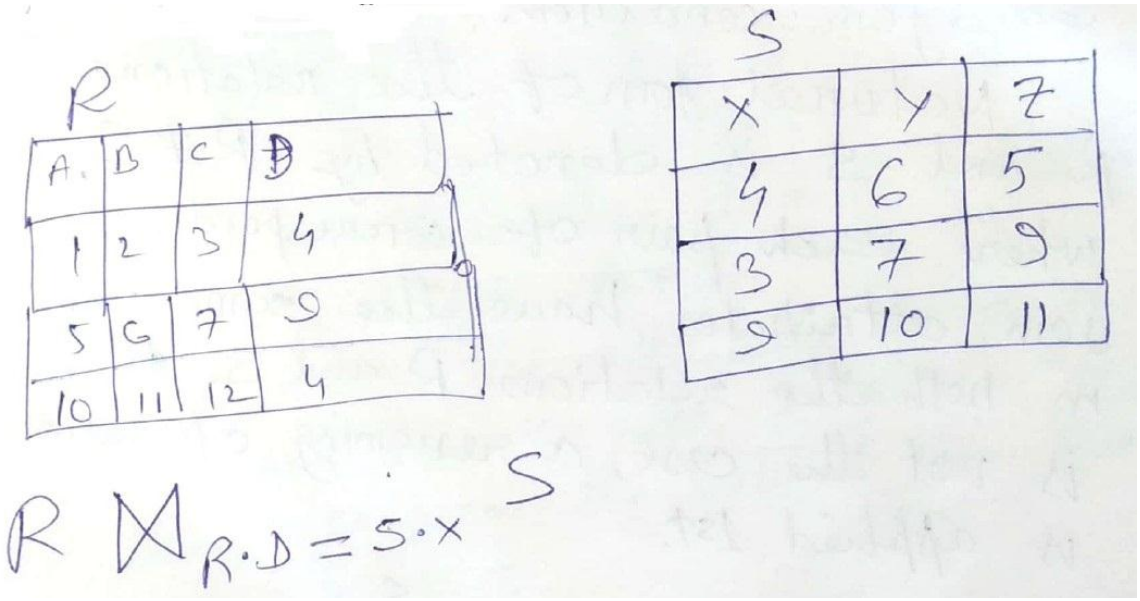
Tuples whose join attributes are NULL or for which the join condition is FALSE *do not* appear in the result. In that sense, the JOIN operation does *not* necessarily preserve all of the information in the participating relations, because tuples that do not get combined with matching ones in the other relation do not appear in the result.

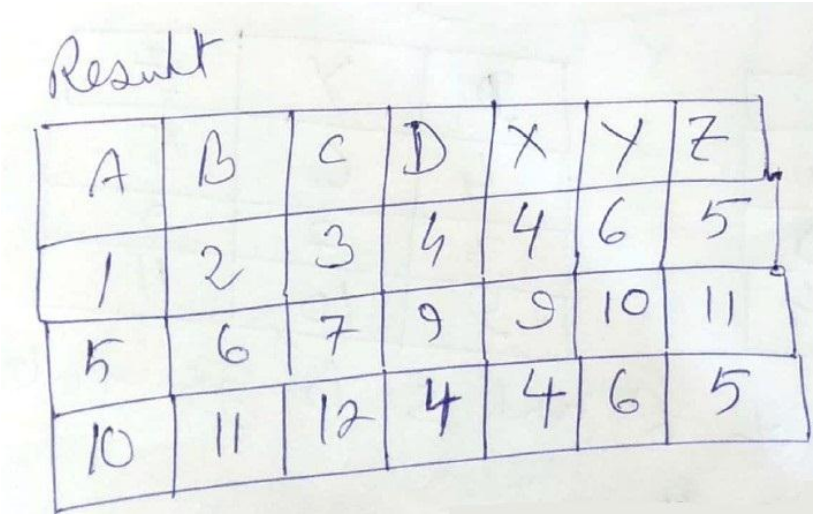
The EQUI JOIN

The most common use of JOIN involves join conditions with equality comparisons only. Such a JOIN, where the only comparison operator used is =, is called an **EQUIJOIN.**

Notice that in the result of an EQUIJOIN we always have one or more pairs of attributes that have *identical values* in every tuple.

- Equi join produces duplicate columns (superfluous or redundant or extra column)





In the above result of Equi join operation one of each pair of attributes with identical values is superfluous. Here either attribute D or attribute X is superfluous (since both the columns have identical values)

NATURAL JOIN

Because one of each pair of attributes with identical values is superfluous in EQUI JOIN, a new operation called **NATURAL JOIN**—denoted by R*S was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition. The standard definition of NATURAL JOIN requires that the two join attributes (or each pair of join attributes) have the same name in both relations. If this is not the case, a renaming operation is applied first.

Example

R

A	B	C	D
1	2	3	4
5	6	7	9
10	11	12	4

S

D	Y	Z
4	6	5
3	7	9
9	10	11

Natural Join of R and S is given by R*S.

Result

A	B	C	D	Y	Z
1	2	3	4	6	5
5	6	7	9	10	11
10	11	12	4	6	5

Note:
No superfluous columns here

Example (ii) Let

R

A	B	C	D
1	2	3	4
5	6	7	9
10	11	12	4

Relation (1)

S

X	Y	Z
4	6	5
3	7	9
9	10	11

Relation (2)

Let, Join attributes be D in R and X in S in the natural join R * S.

Now, Rename the attribute X to D in S (or rename the attribute D to X in R) Ist, then perform the natural join R * S

i.e Perform:

$\rho_{(D,Y,Z)}(S)$ → S

D	Y	Z
4	6	5
3	7	9
9	10	11

Relation (3)

(Ref. to rename operation ρ (Rho))

Now, Perform: R * S. which is same as example (1) i.e.

(using Relations (1) and Relation (3))

A	B	C	D	Y	Z
1	2	3	4	6	5
5	6	7	9	10	11
10	11	12	4	6	5