➢ Keys in Database

1) Super key

A super key is a set of one or more attributes that allow us to identify each tuple uniquely in a relation. For example, the enrollment_no of a student is sufficient to distinguish one student tuple from another.

2) Candidate key

Candidate key is a super key for which no proper subset is a super key.

3) Primary Key

A Primary key is a candidate key that is chosen by database designer to identify tuples uniquely in a relation.

4) Alternate key

An Alternate key is a candidate key that is not chosen by database designer to identify tuples uniquely in a relation.

5) Foreign key

A foreign key is a set of one or more attributes whose values are derived from the primary key attribute of another relation.

Example of keys

Table-1:Atmiya_Student_Info

Enrollment No.	Roll No.	Name	Semester	Department
130070702009	1	ABC	5	I.T
140070702009	1	PQR	3	E.C.
120070702009	2	XYZ	7	C.E.
120070702010	3	SRT	7	C.E.

Table-2:Atmiya_Result_Info

Enrollment No.	Result(%)
130070702009	70
140070702009	80
120070702009	90
120070702010	92

Keys	Attribute
Superkey: enrollment_no of a student is sufficient to distinguish one student tuple from another. combination of any attribute with enrollment no. is also sufficient to distinguish one student tuple from another. Combination of Roll No, Department and Semester can also be used as superkey.	1. {Enrollment No.} 2. { Enrollment No., Roll No./Name/Semester/Department} 3. {Roll No., Department, Semester}
Candidate Key Although Combination of enrollment_no with any attribute is sufficient to distinguish one student tuple from another. But enrollment_no alone is also sufficient to distinguish one student tuple from another. So Enrollment No can be candidate key combination of roll_no and department and semester is sufficient to distinguish one student tuple from another. But either rollno, department or and semester alone is not sufficient to distinguish one student tuple from another. So {rollno, department, semester} is candidate key.	1. {Enrollment No.} 2. { Roll No., Department, Semester}
Primary Key There is always only one primary key in one table. So database designer choose one candidate key from the list of candidate keys. Here enrollmen No. is choosen.	1. {Enrollment No.}
Alternate Key Database designer will select only 1 candidate key. So remaining will be canceled. In this case enrollment No is selected as primary key from two candidate keys. So remaining candidate keys are alternate key.	1. {Roll No., Department, Semester}
Foreign Key Here we have Enrollment No. in table Atmiya_student_info as well as in Atmiya Atmiya _Result_info. Enrollment No. of Atmiya _Result_info can contain any value which is available in Enrollment No of Atmiya_student_info. So it is foreign key which reference to Enrollment No of Atmiya_student_info.	1.{Enrollment No.} (Atmiya_Result_info) For this Primary key is {Enrollment No.}(Atmiya_Student_Info)

Relational Algebra

- The **Relational Model** uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name.
- The relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type.
- The **Relational Algebra** is a *procedural* query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result. The fundamental operations in the relational algebra are *select, project*, *union*, *set difference*, *Cartesian product,* and *rename*. In addition to the fundamental operations, there are several other operations—namely, set intersection, natural join, division.

➢ Fundamental Operations of Relational Algebra

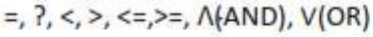
1) Selection:-

Operation: Selects tuples from a relation that satisfy a given condition. It is used to select particular tuples from a relation. It selects particular tuples but all attribute from a relation.

Symbol: σ (Sigma)

Notation: $\sigma_{(condition)}$ <Relation>

Operators: The following operators can be used in a condition.



=, ?, <, >, <=,>=, Λ(AND), V(OR)

Consider following table

Student

Rno	Name	Dept	CPI
101	Ramesh	CE	8
108	Mahesh	EC	6
109	Amit	CE	7
125	Chetan	CI	8
138	Mukesh	ME	7
128	Reeta	EC	6
133	Anita	CE	9

Example: Find out all the students of CE department.

$\boldsymbol{\sigma}_{\textbf{Dept=“CE”}}$ **(Student)**

Output: The above query returns all tuples which contain CE as department name.
Output of above query is as follows

Student

Rno	Name	Dept	CPI
101	Ramesh	CE	8
109	Amit	CE	7
133	Anita	CE	9

2) Projection:-

Operation: Selects specified attributes of a relation. It selects particular attributes but all tuples from a relation.

Symbol: ∏ (Pi)

Notation: ∏ $_{\text{(attribute set)}}$ <Relation>

Consider following table

Student

Rno	**Name**	**Dept**	**CPI**
101	Ramesh	CE	8
108	Mahesh	EC	6
109	Amit	CE	7
125	Chetan	CI	8
138	Mukesh	ME	7
128	Reeta	EC	6
133	Anita	CE	9

Example: List out all students with their roll no, name and department name.

∏Rno, Name, Dept $^{\textbf{(Student)}}$

Output: The above query returns all tuples with three attributes roll no, name and department name.

Output of above query is as follows

Student

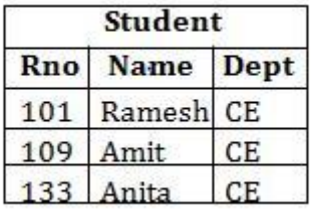
Rno	**Name**	**Dept**
101	Ramesh	CE
108	Mahesh	EC
109	Amit	CE
125	Chetan	CI
138	Mukesh	ME
128	Reeta	EC
133	Anita	CE

3) Composition of Relational Operations

Example: List out all students of CE department with their roll no, name and department name.

∏Rno, Name, Dept (σDept="CE" $^{\textbf{(Student)}}$**)**

Output: The above query returns tuples which contain CE as department with three attributes roll no, name and department name. Output of above query is as follows



Student

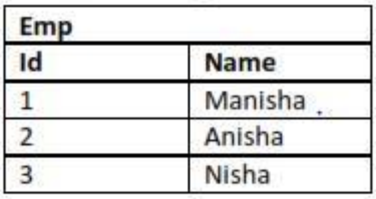
Rno	**Name**	**Dept**
101	Ramesh	CE
109	Amit	CE
133	Anita	CE

4) Union

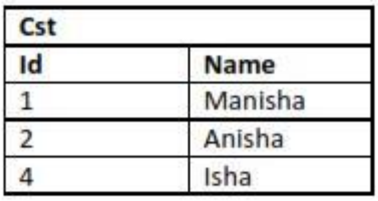
Operation: Selects tuples those are in either or both of the relations.
Symbol : U(union)
Notation : Relation1 U Relation2
Requirement: Union must be taken between compatible relations. Relations R and S are compatible, if

- Both have same arity, i.e. total numbers of attributes, and
- Domains of ith attribute of R and S are same type.

Example :
Consider following tables

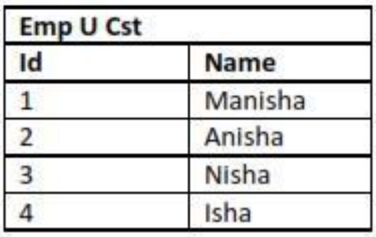


Emp	
Id	**Name**
1	Manisha
2	Anisha
3	Nisha



Cst	
Id	**Name**
1	Manisha
2	Anisha
4	Isha

xample:



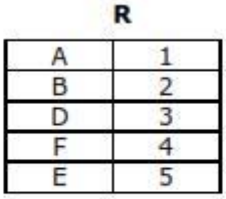
Emp U Cst	
Id	**Name**
1	Manisha
2	Anisha
3	Nisha
4	Isha

5) Difference

Operation: Selects tuples those are in first (left) relation but not in second (right) relation.
Symbol : — (minus)
Notation : Relation1 — Relation2
Requirement: Set-difference must be taken between compatible relations. Relations R and S are compatible, if

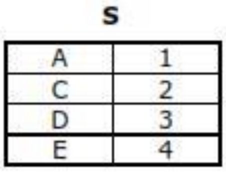
- Both have same arity, i.e. total numbers of attributes, and
- Domains of ith attribute of R and S are same type.

Example :
Consider following tables



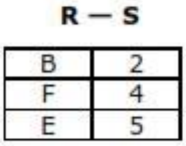
R

A	1
B	2
D	3
F	4
E	5



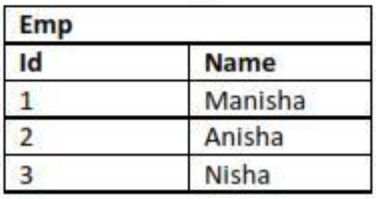
S

A	1
C	2
D	3
E	4

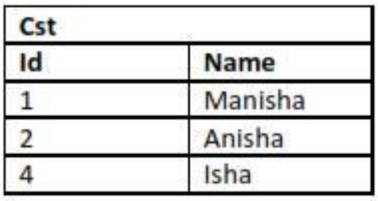


R — S

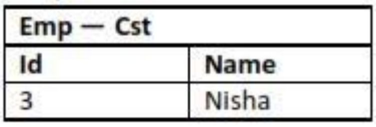
B	2
F	4
E	5



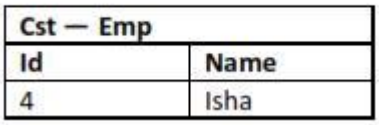
Emp	
Id	**Name**
1	Manisha
2	Anisha
3	Nisha



Cst	
Id	**Name**
1	Manisha
2	Anisha
4	Isha



Emp — Cst	
Id	Name
3	Nisha



Cst — Emp	
Id	Name
4	Isha

6) Rename:-

Operation: It is used to rename a relation or attributes.

Symbol: ρ (rho)

Notation: $\rho_A(B)$ Rename relation B to A.
$\rho_{A(X1,X2....Xn)}(B)$ Rename relation B to A and its attributes to X1, X2, …., Xn.
Consider following table

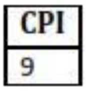
Student

Rno	Name	Dept	CPI
101	Ramesh	CE	8
108	Mahesh	EC	6
109	Amit	CE	7
125	Chetan	CI	8
138	Mukesh	ME	7
128	Reeta	EC	6
133	Anita	CE	9

Example: Find out highest CPI from student table.

$$\prod_{CPI} \textbf{(Student)} — \prod_{A.CPI} (\sigma_{A.CPI<B.CPI} (\rho_A \textbf{(Student)} \text{ X } \rho_B \textbf{(Student)}))$$

Output: The above query returns highest CPI. Output of above query is as follows



CPI
9

7) Cartesian product:-

Operation: Combines information of two relations. It will multiply each tuples of first relation to each tuples of second relation. It is also known as Cross product operation and similar to mathematical Cartesian product operation. **Symbol:** X (cross)

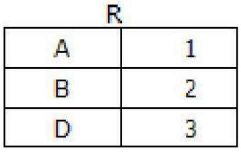
Notation: Relation1 X Relation2

Resultant Relation :

- If relation1 and relation2 have n1 and n2 attributes respectively, then resultant relation will have n1 + n2 attributes from both the input relations.
- If both relations have some attribute having same name, it can be distinguished by combing relation-name.attribute-name.
- If relation1 and relation2 have n1 and n2 tuples respectively, then resultant relation will have n1*n2 attributes, combining each possible pair of tuples from both the input relations.

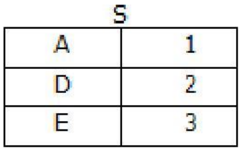
Example

Consider following relation



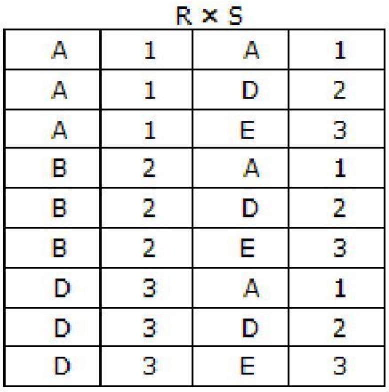
R

A	1
B	2
D	3



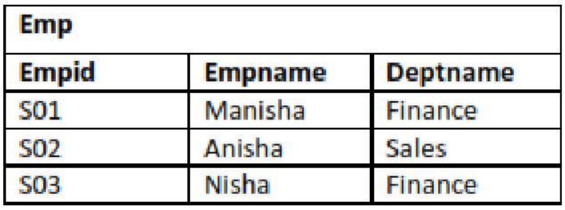
S

A	1
D	2
E	3



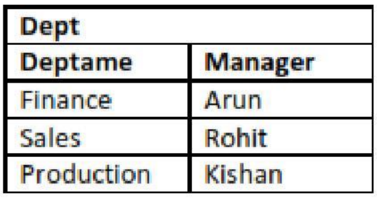
R × S

A	1	A	1
A	1	D	2
A	1	E	3
B	2	A	1
B	2	D	2
B	2	E	3
D	3	A	1
D	3	D	2
D	3	E	3



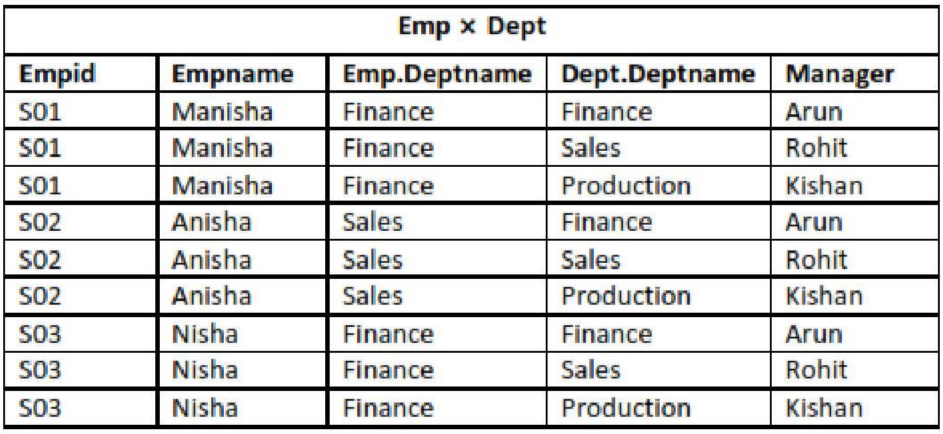
Emp

Empid	Empname	Deptname
S01	Manisha	Finance
S02	Anisha	Sales
S03	Nisha	Finance



Dept

Deptame	Manager
Finance	Arun
Sales	Rohit
Production	Kishan



Emp × Dept

Empid	Empname	Emp.Deptname	Dept.Deptname	Manager
S01	Manisha	Finance	Finance	Arun
S01	Manisha	Finance	Sales	Rohit
S01	Manisha	Finance	Production	Kishan
S02	Anisha	Sales	Finance	Arun
S02	Anisha	Sales	Sales	Rohit
S02	Anisha	Sales	Production	Kishan
S03	Nisha	Finance	Finance	Arun
S03	Nisha	Finance	Sales	Rohit
S03	Nisha	Finance	Production	Kishan

- **Other Operations**

1) Division:-

Operation: The division is a binary relation that is written as R÷S. The result consists of the header of R but not in the header of S, for which it holds that all the tuples in S are presented in R.

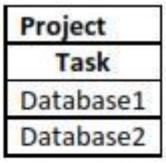
Symbol: ÷

Notation: R ÷ S

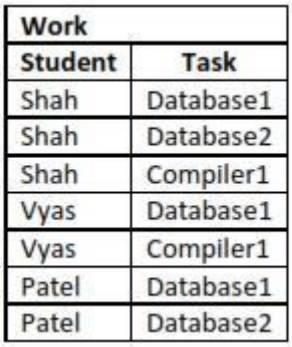
Consider following table

Example: Find out all students having both tasks Database1 as well as Database 2.

$$\prod_{(student)}(\mathbf{Work}) \div \prod_{(Task)}(\mathbf{Project})$$

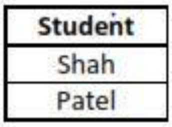


Project
Task
Database1
Database2



Work	
Student	**Task**
Shah	Database1
Shah	Database2
Shah	Compiler1
Vyas	Database1
Vyas	Compiler1
Patel	Database1
Patel	Database2

Output: It gives name of all students whose task is both Database1 as well as Database 2. Output of above query is as follows.



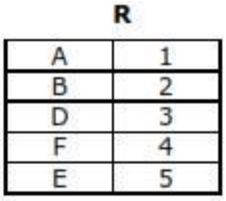
Student
Shah
Patel

2) Intersection

Operation: Selects tuples those are in both relations. **Symbol :** ∩ (intersection)
Notation : Relation1 ∩ Relation2
Requirement: Set-intersection must be taken between compatible relations. Relations R and S are compatible, if

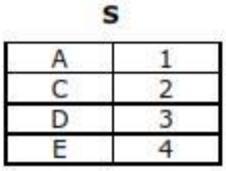
- Both have same arity, i.e. total numbers of attributes, and
- Domains of ith attributes of R and S are same type.

Example
Consider following tables



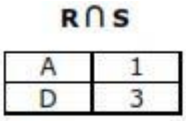
R

A	1
B	2
D	3
F	4
E	5



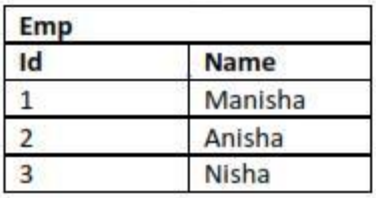
S

A	1
C	2
D	3
E	4

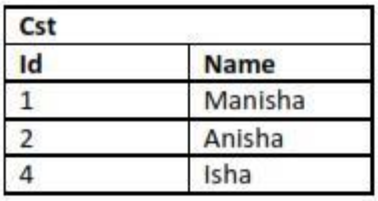


R∩S

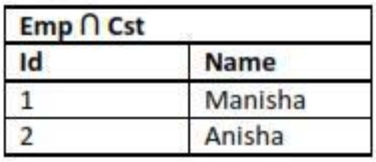
A	1
D	3



Emp	
Id	**Name**
1	Manisha
2	Anisha
3	Nisha



Cst	
Id	**Name**
1	Manisha
2	Anisha
4	Isha



Emp ∩ Cst	
Id	**Name**
1	Manisha
2	Anisha

Aggregate Function:-

Operation: It takes a more than one value as input and returns a single value as output (result).
Symbol: G

Notation: G $_{\text{function (attribute)}}$ (relation)
Aggregate functions: Sum, Count, Max, Min, Avg. Consider following table

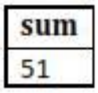
Student

Rno	Name	Dept	CPI
101	Ramesh	CE	8
108	Mahesh	EC	6
109	Amit	CE	7
125	Chetan	CI	8
138	Mukesh	ME	7
128	Reeta	EC	6
133	Anita	CE	9

Example: Find out sum of all students CPI.

G $_{\text{sum (CPI)}}$ **(Student)**

Output: The above query returns sum of CPI.Output of above query is as follows



sum
51

Example: Find out max and min CPI.

^{G} **max (CPI), min (CPI)** $^{\text{(Student)}}$

Output: The above query returns sum of CPI.Output of above query is as follows

max	min
9	6