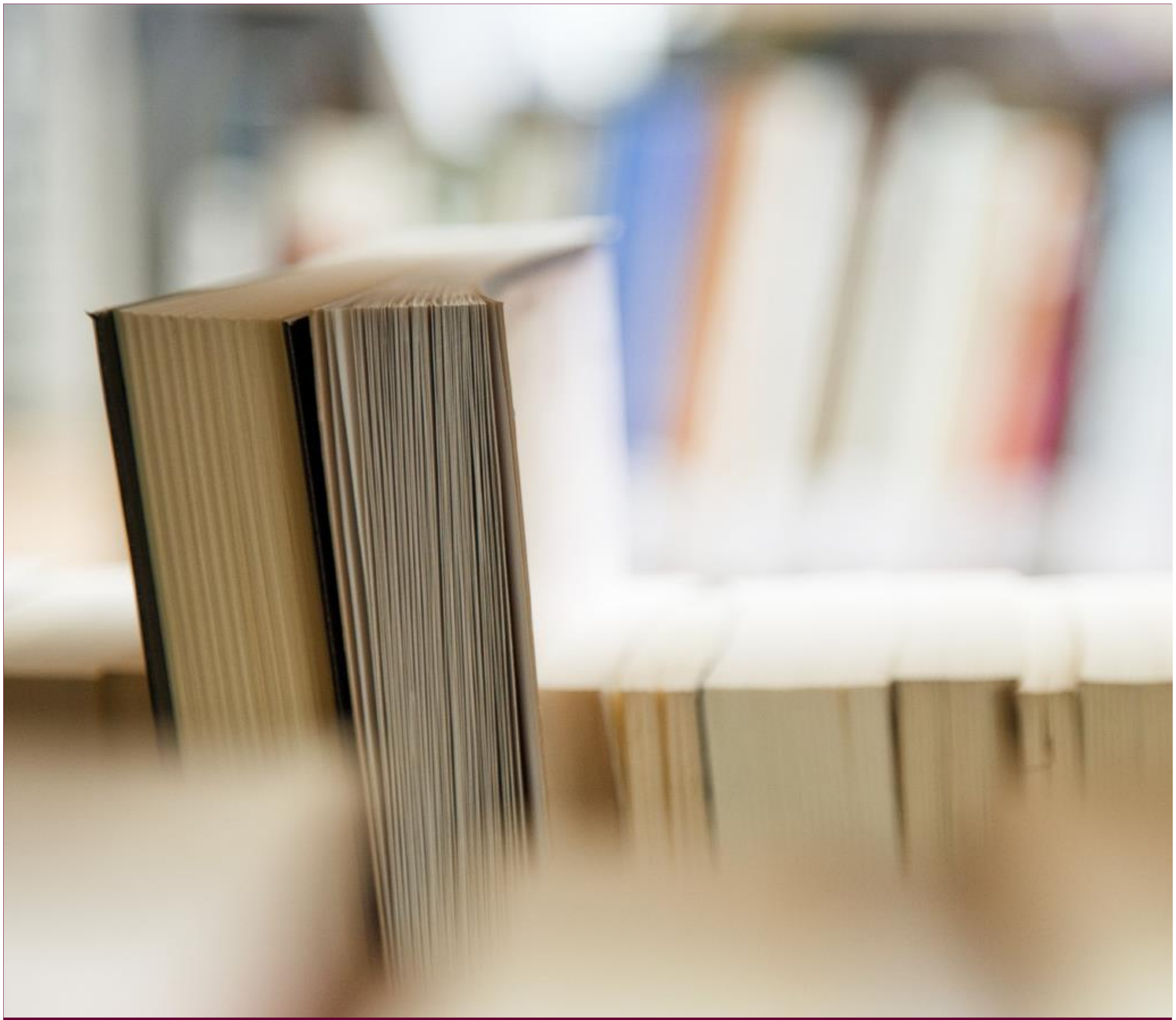


UNIT-03

DBMS

Prof.Akshara Vilas Chavan SYKKSU

RELATIONAL MODEL UNIT-03

INTRODUCTION TO SQL

- The relational Model was proposed by E.F. Codd to model data in the form of relations or tables.
- After designing the conceptual model of the Database using ER diagram, we need to convert the conceptual model into a relational model which can be implemented using any RDBMS language like Oracle SQL, MySQL, etc.
- The relational model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables).
- Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE, and AGE shown in Table 1.

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	978264856	18

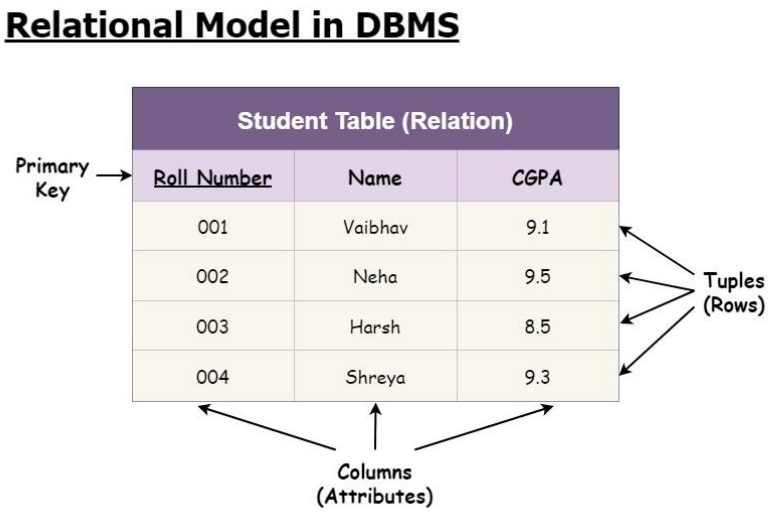
IMPORTANT TERMINOLOGIES

- **Attribute:** Attributes are the properties that define a relation. e.g.; ROLL_NO, NAME
- **Relation Schema:** A relation schema represents the name of the relation with its attributes. e.g.; STUDENT (ROLL_NO, NAME, ADDRESS, PHONE, and AGE) is the relation schema for STUDENT.
- **Tuple**: Each row in the relation is known as a tuple. The above relation contains 4 tuples, one of which is shown as:

1	**RAM**	**DELHI**	**9455123451**	**18**

- **Relation Instance:** The set of tuples of a relation at a particular instance of time is called a relation instance.
- **Degree:** The number of attributes in the relation is known as the degree of the relation. The **STUDENT** relation defined above has degree 5.
- **Cardinality:** The number of tuples in a relation is known as cardinality. The **STUDENT** relation defined above has cardinality 4.
- **Column:** The column represents the set of values for a particular attribute.

- **NULL Values:** The value which is not known or unavailable is called a NULL value. It is represented by blank space.



Student Table (Relation)

Roll Number	Name	CGPA
001	Vaibhav	9.1
002	Neha	9.5
003	Harsh	8.5
004	Shreya	9.3

ADVANTAGES OF USING THE RELATIONAL MODEL

The advantages and reasons due to which the relational model in DBMS is widely accepted as a standard are:

- **Simple and Easy to Use** - Storing data in tables is much easier to understand and implement as compared to other storage techniques.
- **Manageability** - Because of the independent nature of each relation in a relational database, it is easy to manipulate and manage. This improves the performance of the database.
- **Query capability** - With the introduction of relational algebra, relational databases provide easy access to data via high-level query language like SQL.
- **Data integrity** - With the introduction and implementation of relational constraints, the relational model can maintain data integrity in the database.

DISADVANTAGES OF USING THE RELATIONAL MODEL

The main disadvantages of relational model in DBMS occur while dealing with a huge amount of data as:

- The performance of the relational model depends upon the number of relations present in the database.
- Hence, as the number of tables increases, the requirement of physical memory increases.
- The structure becomes complex and there is a decrease in the response time for the queries.
- Because of all these factors, the cost of implementing a relational database increase.

RELATIONAL ALGEBRA

- 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.
- It uses operators to perform queries. An operator can be either unary or binary.
- The most general operations in relational algebra are grouped into two categories as follows:
- Relation-based operations
  - Selection
  - Projection
  - Join
  - Division
- Set-based operations
  - UNION (υ)
  - INTERSECTION ( ),
  - DIFFERENCE (-)
  - CARTESIAN PRODUCT (x)

FUNDAMENTAL OPERATIONS IN RELATIONAL ALGEBRA

RELATION-BASED OPERATIONS

1. SELECT OPERATION (σ)

- The SELECT operation is used for selecting a subset of the tuples according to a given selection condition.
- Sigma(σ) Symbol denotes it.
- It is used as an expression to choose tuples which meet the selection condition. Select operator selects tuples that satisfy a given predicate.
- **Notation – $\sigma_p(r)$**
- Where σ stands for selection predicate and r stands for relation. p is prepositional logic formula which may use connectors like and, or, and not. These terms may use relational operators like − =, ≠, ≥, < , >, ≤.
- **For example:**
  1. **$\sigma_{subject = "database"}$(Books)**
     Output − Selects tuples from books where subject is 'database'.
  2. **$\sigma_{subject = "database" \text{ and } price = "450"}$(Books)**
     Output − Selects tuples from books where subject is 'database' and 'price' is 450.
  3. **$\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}$(Books)**
     Output − Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

- **Queries:**
  1. Select the EMPLOYEE tuples whose department number is 4:

     $\sigma_{DNO = 4}$ (EMPLOYEE)
  2. Select the employee tuples whose salary is greater than $30,000:

     $\sigma_{SALARY > 30,000}$ (EMPLOYEE)
  3. Select the instructors in Physics with a salary greater than $90,000, (and (∧), or (∨), and not (¬))

     $\sigma_{deptname = \text{"Physics"} \wedge salary > 90000}$ (INSTRUCTOR)

2. PROJECTION

- Project operation is used to project only a certain set of attributes of a relation.
- In simple words, If you want to see only the names of all the students in the Student table, then you can use Project Operation.
- It will only project or show the columns or attributes asked for, and will also remove duplicate data from the columns.
- where π (pi) is the symbol used to represent the PROJECT operation
- **NOTATION-$\Pi_{A1, A2...}$(r)**

  Where A1, A2 , An are attribute names of relation r
- In SQL, the PROJECT attribute list is specified in the SELECT clause of a query.
- For example, the following operation: πname, Salary(EMPLOYEE) would correspond to the following SQL query:

```
 SELECT DISTINCT name, Salary
FROM EMPLOYEE
```

- Notice that if we remove the keyword DISTINCT from this SQL query, then duplicates will not be eliminated.
- **For Example:**

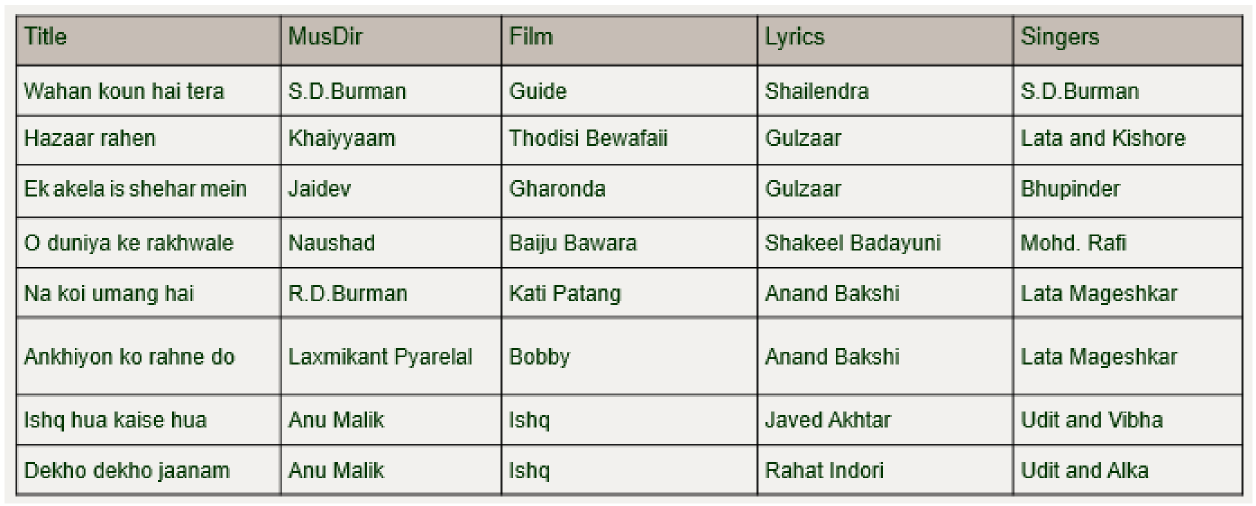
CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

- **Query- Π $_{\text{CustomerName, Status}}$ (Customers)**
  Output:

CustomerName	Status
Google	Active
Amazon	Active
Apple	Inactive
Alibaba	Active

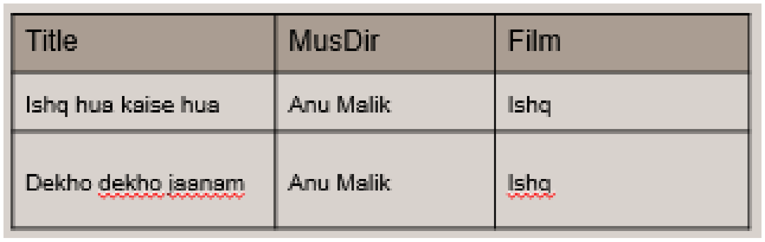
3. COMBINATION OF 'SELECT' AND 'PROJECT' OPERATION

- It is possible to club the operations 'Select' and 'Project' in order to provide more effective results.
- The combination is possible in any order. It is possible because the result of either of the operation is again a relation; further that relation is used as operand for the operation.
- For example
- 'Select' operation is applied on a relation 'R' to produce the resulting relation 'Result1'. Then a 'Project' operation can be applied on 'Result1' to produce another resulting relation say 'Result11'.
- **Example:**
  - Consider the same relation Song (Title, MusDir, Film, Lyrics, Singers) represented with some tuples as follows:
  - Song:



Title	MusDir	Film	Lyrics	Singers
Wahan koun hai tera	S.D.Burman	Guide	Shailendra	S.D.Burman
Hazaar rahen	Khaiyyaam	Thodisi Bewafaii	Gulzaar	Lata and Kishore
Ek akela is shehar mein	Jaidev	Gharonda	Gulzaar	Bhupinder
O duniya ke rakhwale	Naushad	Baiju Bawara	Shakeel Badayuni	Mohd. Rafi
Na koi umang hai	R.D.Burman	Kati Patang	Anand Bakshi	Lata Mageshkar
Ankhiyon ko rahne do	Laxmikant Pyarelal	Bobby	Anand Bakshi	Lata Mageshkar
Ishq hua kaise hua	Anu Malik	Ishq	Javed Akhtar	Udit and Vibha
Dekho dekho jaanam	Anu Malik	Ishq	Rahat Indori	Udit and Alka

- Write the symbolic expression to obtain the following results given in the form of tables:
  - Result1:

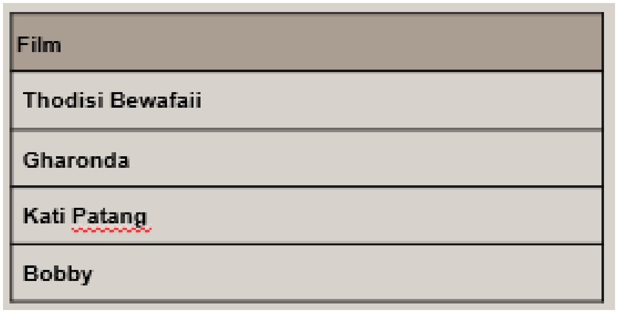


Title	MusDir	Film
Ishq hua kaise hua	Anu Malik	Ishq
Dekho dekho jaanam	Anu Malik	Ishq

  - Query:

    $\Pi_{\text{Title, MusDir, Film}} (\sigma_{\text{Film='Ishq'}}(\text{Song}))$

  - Result 2:



Film
Thodisi Bewafaii
Gharonda
Kati Patang
Bobby

  - Query:

    $\Pi_{\text{Film}}(\sigma_{\text{Lyrics='Gulzaar' AND Lyrics='Anand Bakshi'}}(\text{song}))$

4. JOIN OPERATIONS

- 'Join' operation requires TWO relations or operands for its operation. The 'Join' operation joins or clubs the tuples from two relations to form a single relation.
- It is a very good operation when two different relations are to be joined on the basis of some similar (having same domain) attributes.
- The operator used to represent the join operation is ⋈.
- Types of JOIN:
- Inner Joins:
  1. Theta join
  2. EQUI join
  3. Natural join
- Outer join:
  1. Left Outer Join
  2. Right Outer Join
  3. Full Outer Join

❖ Inner Join

1. Theta Join

- **THETA JOIN** allows you to merge two tables based on the condition represented by theta. Theta joins work for all comparison operators. It is denoted by symbol **θ**. The general case of JOIN operation is called a Theta join.
- Syntax: $A \bowtie_{\theta} B$
- Theta join can use any conditions in the selection criteria.
- Consider the following tables.

Table A

column 1	column 2
1	1
1	2

Table B

column 1	column 2
1	1
1	3

- **For example:**

  Query: $A \bowtie_{\text{A.column 2} > \text{B.column 2}} (B)$

  - **Output:**

A ⋈ A.column 2 > B.column 2 (B)			
column 1	column 2	Column 1	Column 2
1	2	1	1

- **For example:**

Emp:

ENo	EName	HNo
1001	R. A. Sharma	E-105
1002	Raghav Deshpande	F-293
1004	Arun Sharma	D-108/C
1005	Susheel Sharma	C-115
1010	Ramesh Wadhawan	C-119

Dept:

ENo	Department	Salary
1002	Administration	29000
1004	Computer	16500
1005	Administration	8500
1006	Computer	9500
1011	Electronics	10200

- Query: $\text{Emp} \bowtie_{\text{Salary}>10000} \text{Dept}$

- Output:

Emp.ENo	EName	HNo	Department	Salary
1002	Raghav Deshpande	F-293	Administration	29000
1004	Arun Sharma	D-108/C	Computer	16500

2. EQUI Join

- EQUI JOIN is done when a Theta join uses only the equivalence condition. EQUI join is the most difficult operation to implement efficiently in an RDBMS, and one reason why RDBMS have essential performance problems.
- For example:
- Query:

$$A \bowtie_{A.column\ 2\ =\ B.column\ 2} (B)$$

- Output:

A ⋈ A.column 2 = B.column 2 (B)			
column 1	column 2	column 1	column 2
1	1	1	1

- **For example:**

Emp:

ENo	EName	HNo
1001	R. A. Sharma	E-105
1002	Raghav Deshpande	F-293
1004	Arun Sharma	D-108/C
1005	Susheel Sharma	C-115
1010	Ramesh Wadhawan	C-119

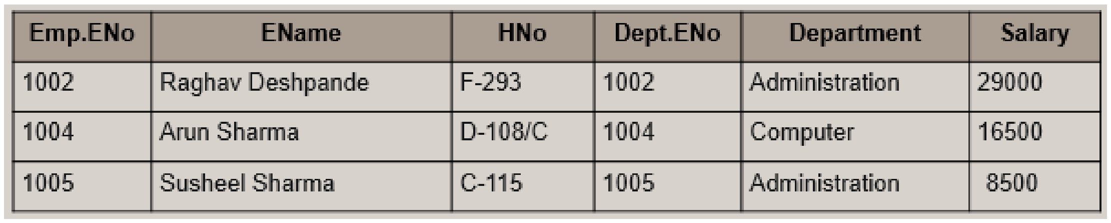
Dept:

ENo	Department	Salary
1002	Administration	29000
1004	Computer	16500
1005	Administration	8500
1006	Computer	9500
1011	Electronics	10200

- Query:

$$\text{Emp} \bowtie_{\text{Emp.ENo=Dept.ENo}} \text{Dept}$$

- Output:



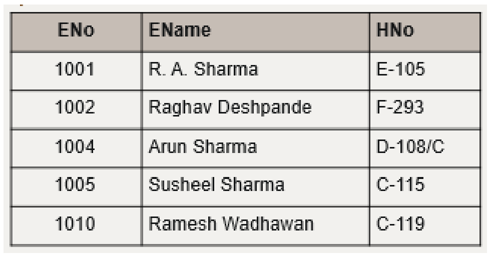
Emp.ENo	EName	HNo	Dept.ENo	Department	Salary
1002	Raghav Deshpande	F-293	1002	Administration	29000
1004	Arun Sharma	D-108/C	1004	Computer	16500
1005	Susheel Sharma	C-115	1005	Administration	8500

3. Natural Join (⋈)

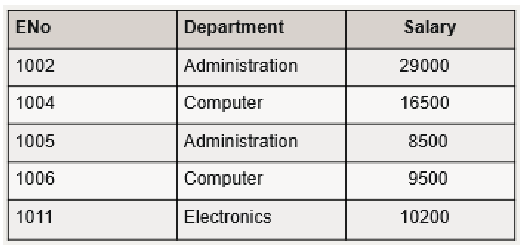
- Natural join is a join operation in which the tuples of one relation are clubbed (joined horizontally) with the tuples of the other relation, on the basis of the equal value of common attributes.
- The equality condition is not at all specified on the common attribute or attributes. It is naturally done. That is why this operation is called as natural join operation
- Symbolically the syntax of natural join operation is represented as follows:

$$\text{Relation1} \bowtie \text{Relation2}$$

- The natural join operation is performed on the basis of the following sequence:
    1. The combination of tuples from First relation to Second relation is formed
    2. Selection operation is performed with the condition of equality on common attribute/s of both the relations
    3. The duplicate attributes are removed from the resulting relation.
- **For example:**
    - Emp: Dept:



ENo	EName	HNo
1001	R. A. Sharma	E-105
1002	Raghav Deshpande	F-293
1004	Arun Sharma	D-108/C
1005	Susheel Sharma	C-115
1010	Ramesh Wadhawan	C-119

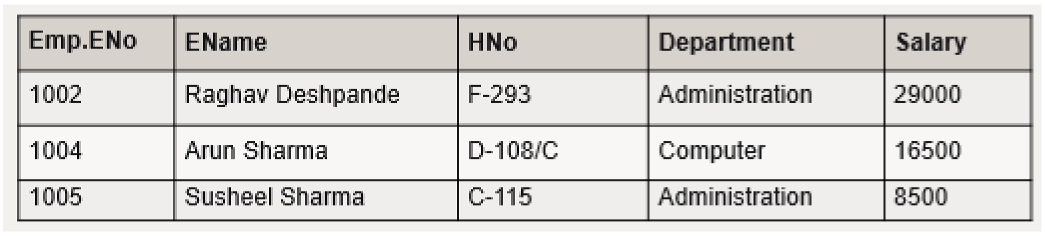


ENo	Department	Salary
1002	Administration	29000
1004	Computer	16500
1005	Administration	8500
1006	Computer	9500
1011	Electronics	10200

- Query:

$$\text{Emp} \bowtie \text{Dept}$$

- Output:

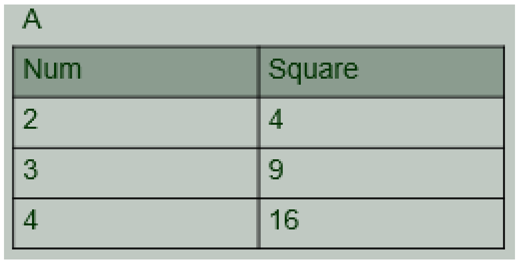


Emp.ENo	EName	HNo	Department	Salary
1002	Raghav Deshpande	F-293	Administration	29000
1004	Arun Sharma	D-108/C	Computer	16500
1005	Susheel Sharma	C-115	Administration	8500

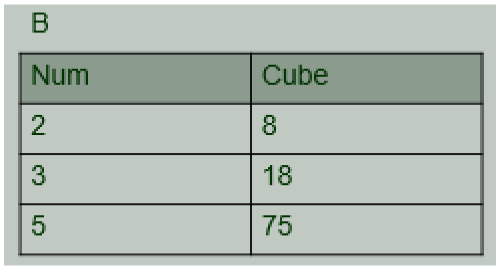
❖ Outer Join

- The outer join computes the natural join and adds extra tuples either from one relation or from both the relations substituting the NULL value for the missing data of the tuples.
- An OUTER JOIN doesn't require each record in the two join tables to have a matching record. In this type of join, the table retains each record even if no other matching record exists.
- Three types of Outer Joins are:
    - Left Outer Join
    - Right Outer Join
    - Full Outer Join

- **Left Outer Join (A ⟕B)**
    - LEFT JOIN returns all the rows from the table on the left even if no matching rows have been found in the table on the right. When no matching record found in the table on the right, NULL is returned.
    - Consider the following 2 Tables


A

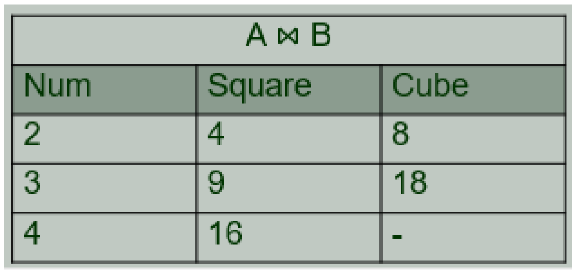
Num	Square
2	4
3	9
4	16


B

Num	Cube
2	8
3	18
5	75

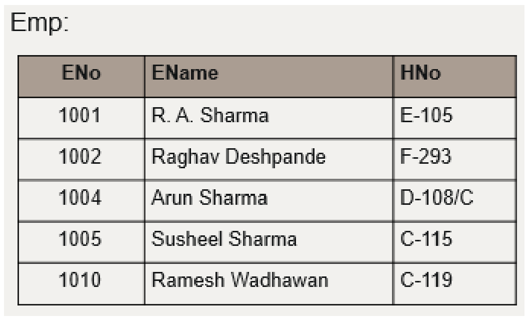
- **Query:** A ⟕ B

- **Output:**

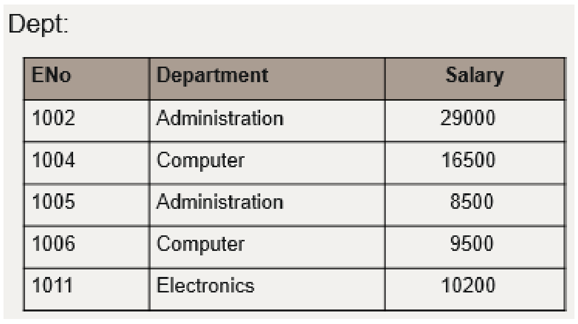

A ⋈ B

Num	Square	Cube
2	4	8
3	9	18
4	16	-

- **For Example**


Emp:

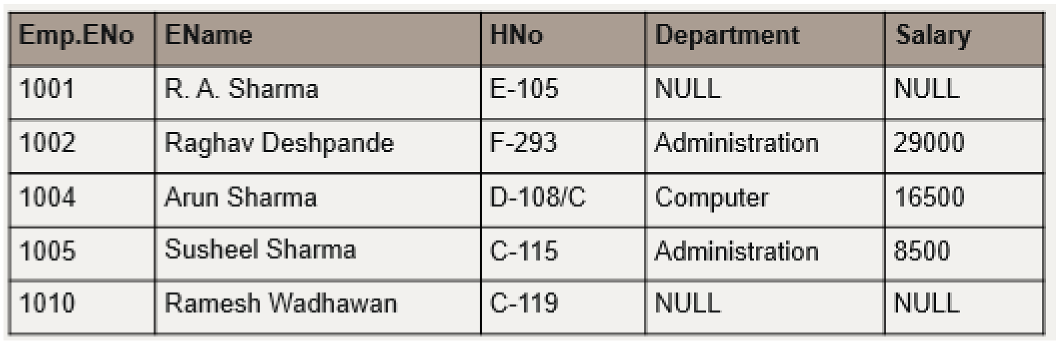
ENo	EName	HNo
1001	R. A. Sharma	E-105
1002	Raghav Deshpande	F-293
1004	Arun Sharma	D-108/C
1005	Susheel Sharma	C-115
1010	Ramesh Wadhawan	C-119


Dept:

ENo	Department	Salary
1002	Administration	29000
1004	Computer	16500
1005	Administration	8500
1006	Computer	9500
1011	Electronics	10200

- **Query:** emp ⟕ dept

- **Output:**

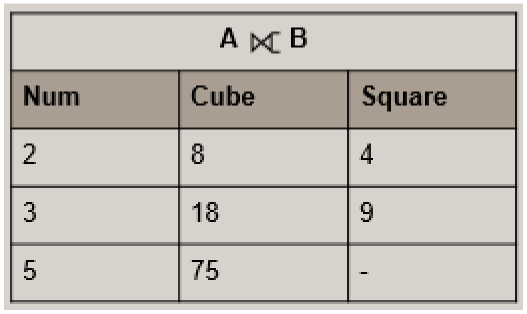


Emp.ENo	EName	HNo	Department	Salary
1001	R. A. Sharma	E-105	NULL	NULL
1002	Raghav Deshpande	F-293	Administration	29000
1004	Arun Sharma	D-108/C	Computer	16500
1005	Susheel Sharma	C-115	Administration	8500
1010	Ramesh Wadhawan	C-119	NULL	NULL

▪ Right Outer Join (A ⟖ B)

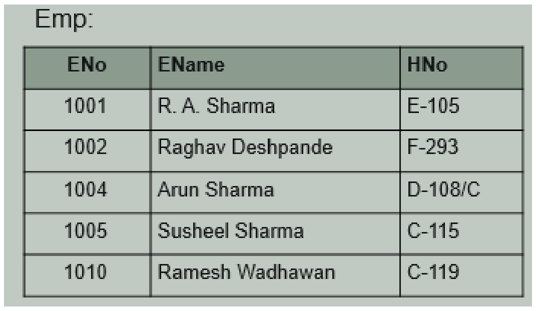
- RIGHT JOIN returns all the columns from the table on the right even if no matching rows have been found in the table on the left. Where no matches have been found in the table on the left, NULL is returned. RIGHT outer JOIN is the opposite of LEFT JOIN
- Consider the same table from left outer join example:
- **Query:** A ⟖ B

- **Output:**



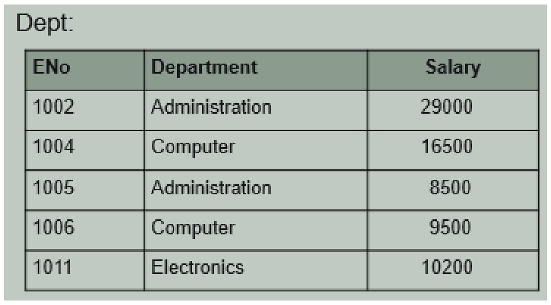
A ⟖ B		
Num	**Cube**	**Square**
2	8	4
3	18	9
5	75	-

- **For Example**



Emp:

ENo	EName	HNo
1001	R. A. Sharma	E-105
1002	Raghav Deshpande	F-293
1004	Arun Sharma	D-108/C
1005	Susheel Sharma	C-115
1010	Ramesh Wadhawan	C-119

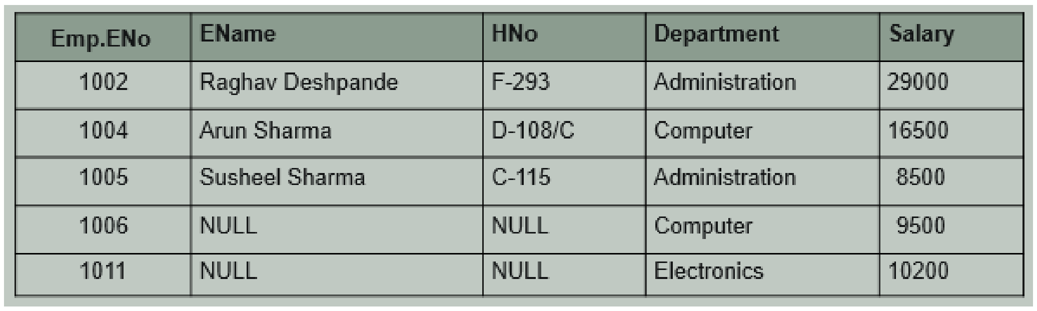


Dept:

ENo	Department	Salary
1002	Administration	29000
1004	Computer	16500
1005	Administration	8500
1006	Computer	9500
1011	Electronics	10200

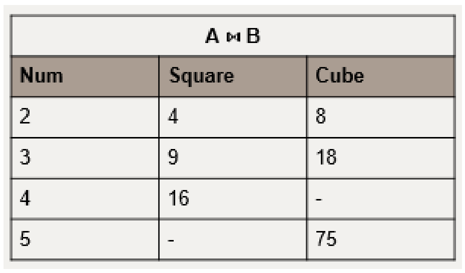
Query: A ⟖ B

Output:



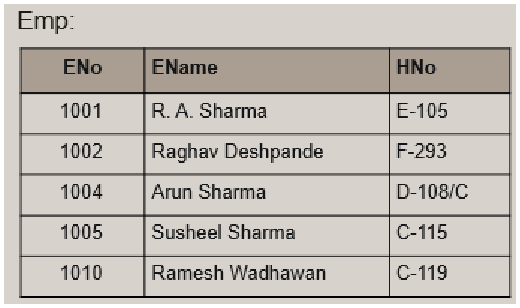
Emp.ENo	EName	HNo	Department	Salary
1002	Raghav Deshpande	F-293	Administration	29000
1004	Arun Sharma	D-108/C	Computer	16500
1005	Susheel Sharma	C-115	Administration	8500
1006	NULL	NULL	Computer	9500
1011	NULL	NULL	Electronics	10200

- **Full Outer Join (A ⟗ B)**
  - In a FULL OUTER JOIN , all tuples from both relations are included in the result, irrespective of the matching condition.
  - **Query**: A ⟗ B
  - **Output:**



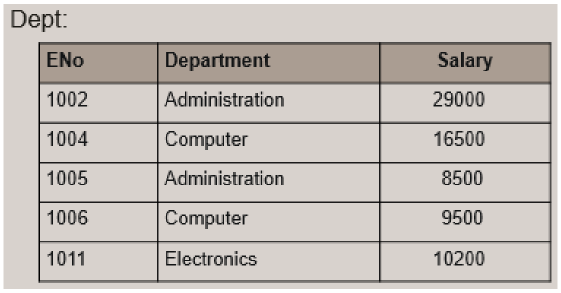
A ⋈ B		
Num	Square	Cube
2	4	8
3	9	18
4	16	-
5	-	75

  - **Example:**



Emp:

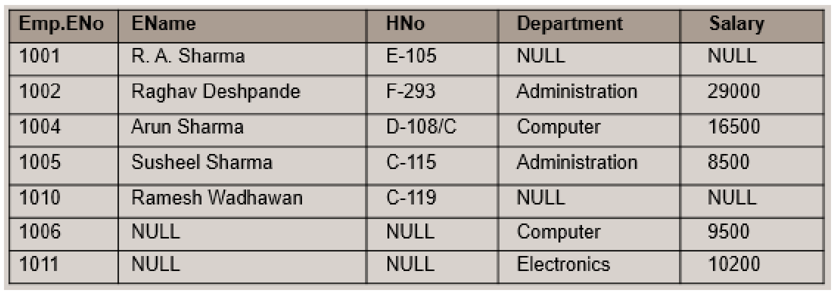
ENo	EName	HNo
1001	R. A. Sharma	E-105
1002	Raghav Deshpande	F-293
1004	Arun Sharma	D-108/C
1005	Susheel Sharma	C-115
1010	Ramesh Wadhawan	C-119



Dept:

ENo	Department	Salary
1002	Administration	29000
1004	Computer	16500
1005	Administration	8500
1006	Computer	9500
1011	Electronics	10200

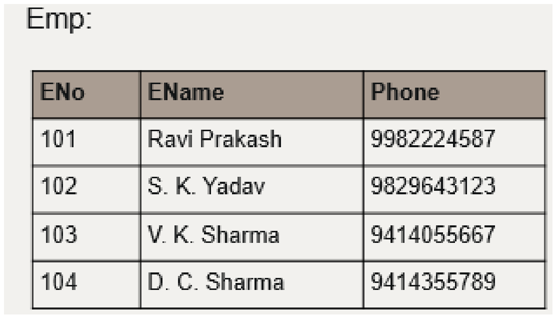
  - **Query:** Emp ⟗ Dept
  - **Output:**



Emp.ENo	EName	HNo	Department	Salary
1001	R. A. Sharma	E-105	NULL	NULL
1002	Raghav Deshpande	F-293	Administration	29000
1004	Arun Sharma	D-108/C	Computer	16500
1005	Susheel Sharma	C-115	Administration	8500
1010	Ramesh Wadhawan	C-119	NULL	NULL
1006	NULL	NULL	Computer	9500
1011	NULL	NULL	Electronics	10200

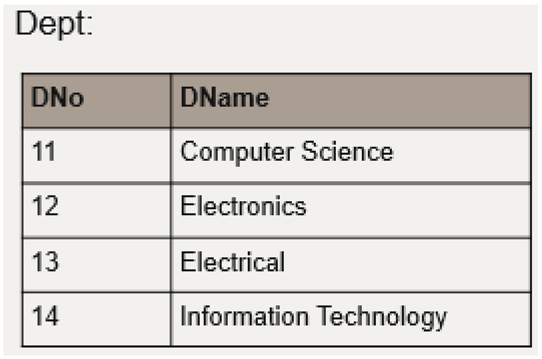
- **Cross Product or Cartesian Product(X) Operation**
  - The cross product operation is applied on two relations.
  - Every tuple from the first relation is combined or concatenated with every tuple of second relation and inserted in the resulting relation.
  - Number of tuples in the resulting relation is equal to the product of tuples of first relation and tuples of second relation.
  - The operator used for the cross product operation is same as cross product operator of sets. It is represented as 'X'. The syntax of the cross product operation is represented symbolically as:

Relation1 X Relation2



Emp:

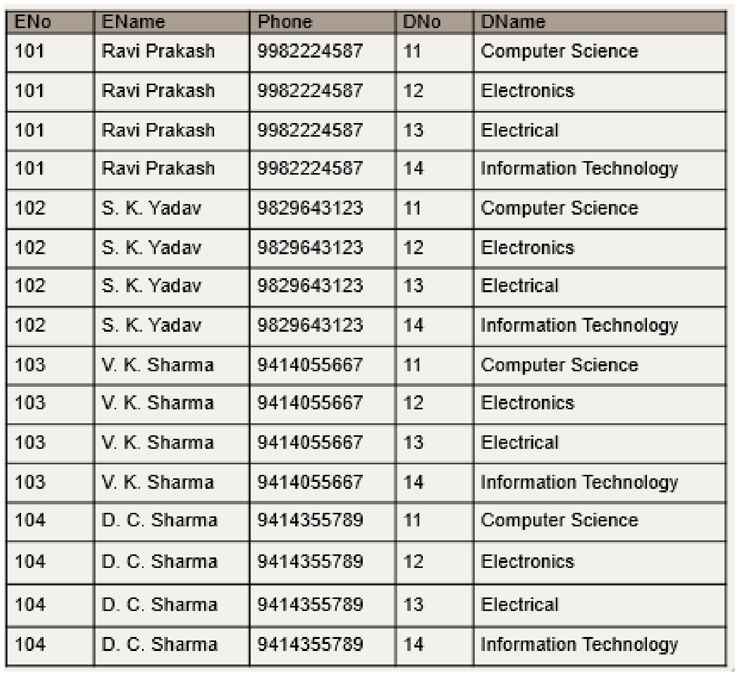
ENo	EName	Phone
101	Ravi Prakash	9982224587
102	S. K. Yadav	9829643123
103	V. K. Sharma	9414055667
104	D. C. Sharma	9414355789



Dept:

DNo	DName
11	Computer Science
12	Electronics
13	Electrical
14	Information Technology

  - **Example**
  - **Query:** Emp **X** Dept
  - **Output:**



ENo	EName	Phone	DNo	DName
101	Ravi Prakash	9982224587	11	Computer Science
101	Ravi Prakash	9982224587	12	Electronics
101	Ravi Prakash	9982224587	13	Electrical
101	Ravi Prakash	9982224587	14	Information Technology
102	S. K. Yadav	9829643123	11	Computer Science
102	S. K. Yadav	9829643123	12	Electronics
102	S. K. Yadav	9829643123	13	Electrical
102	S. K. Yadav	9829643123	14	Information Technology
103	V. K. Sharma	9414055667	11	Computer Science
103	V. K. Sharma	9414055667	12	Electronics
103	V. K. Sharma	9414055667	13	Electrical
103	V. K. Sharma	9414055667	14	Information Technology
104	D. C. Sharma	9414355789	11	Computer Science
104	D. C. Sharma	9414355789	12	Electronics
104	D. C. Sharma	9414355789	13	Electrical
104	D. C. Sharma	9414355789	14	Information Technology

SET-BASED OPERATIONS

1. Union operation (U)

- UNION is symbolized by ∪ symbol. It includes all tuples that are in tables A or in B. It also eliminates duplicate tuples. So, set A UNION set B would be expressed as :

A U B

- For a union operation to be valid, the following conditions must hold -
  - R and S must be the same number of attributes.
  - Attribute domains need to be compatible.
  - Duplicate tuples should be automatically removed.
- **Example**

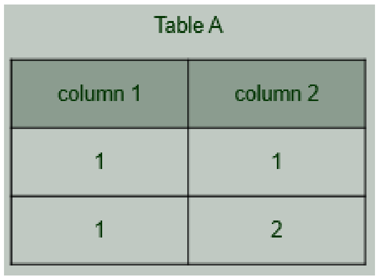


Table A

column 1	column 2
1	1
1	2

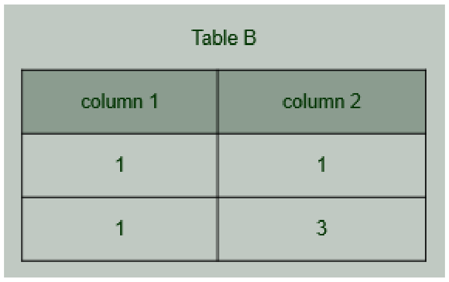


Table B

column 1	column 2
1	1
1	3

  - **Output:**

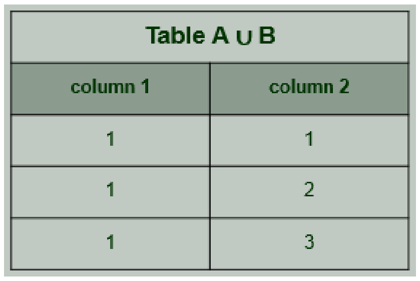


Table A ∪ B

column 1	column 2
1	1
1	2
1	3

2. Difference (-)

- Symbol denotes it. The result of A - B, is a relation which includes all tuples that are in A but not in B.
- The attribute name of A has to match with the attribute name in B.
- It should be defined relation consisting of the tuples that are in relation A, but not in B.
- **Example:**

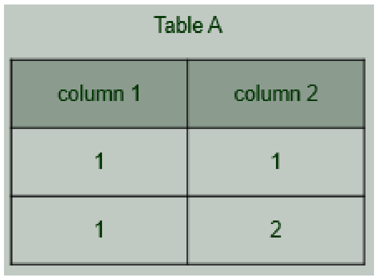


Table A

column 1	column 2
1	1
1	2

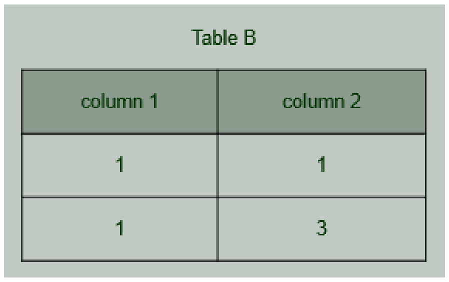


Table B

column 1	column 2
1	1
1	3

- **Output:**

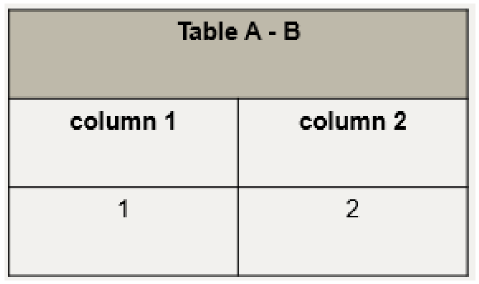


Table A - B	
column 1	column 2
1	2

3. Intersection

- An intersection is defined by the symbol ∩
- Defines a relation consisting of a set of all tuple that are in both A and B. However, A and B must be union compatible.

$$A \cap B$$

- **Example:**

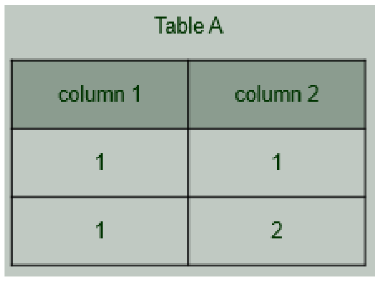


Table A	
column 1	column 2
1	1
1	2

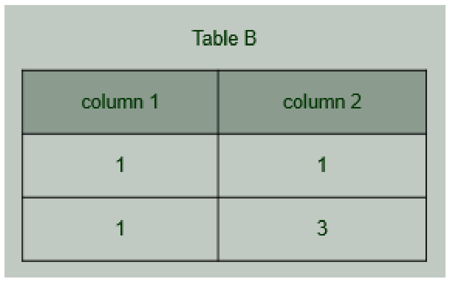


Table B	
column 1	column 2
1	1
1	3

- **Output:**

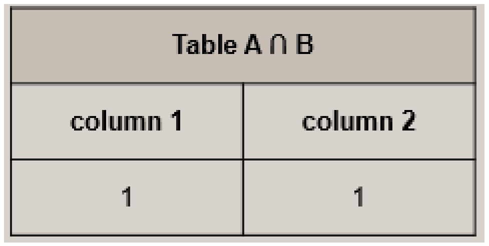


Table A ∩ B	
column 1	column 2
1	1

EXTENDED RELATIONAL ALGEBRA OPERATIONS

- Extended operators are those operators which can be derived from basic operators. There are mainly three types of extended operators in Relational Algebra:
    - Join
    - Intersection
    - Divide

(Refer above notes of the same)

AGGREGATE FUNCTIONS.

- Aggregate functions in DBMS take multiple rows from the table and return a value according to the query.
- All the aggregate functions are used in Select statement.
- Syntax:

  SELECT <FUNCTION NAME> (<PARAMETER>) FROM <TABLE NAME>

1. AVG Function

- This function returns the average value of the numeric column that is supplied as a parameter.
- Example: Write a query to select average salary from employee table.

  Select AVG(salary) from Employee

2. COUNT Function

- The count function returns the number of rows in the result. It does not count the null values.
- Example: Write a query to return number of rows where salary > 20000.

  Select COUNT(*) from Employee where Salary > 20000;

- Types –
  - COUNT(*): Counts all the number of rows of the table including null.
  - COUNT(COLUMN_NAME): count number of non-null values in column.
  - COUNT(DISTINCT COLUMN_NAME): count number of distinct values in a column.

3. MAX Function

- The MAX function is used to find maximum value in the column that is supplied as a parameter. It can be used on any type of data.
- Example – Write a query to find the maximum salary in employee table.

  Select MAX(salary) from Employee

4. SUM Function

- This function sums up the values in the column supplied as a parameter.
- Example: Write a query to get the total salary of employees.

  Select SUM(salary) from Employee

5. MIN Function

- MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.
- Example − Write a query to find the minimum salary in employee table

  Select MIN(salary) from Employee