RELATIONAL DATA MODELS

What is the Relational Model?

- The relational model represents **how data is stored in Relational Databases.** A relational database consists **of a collection of tables**, each of which is **assigned a unique name.**

Example :

Consider a relation **STUDENT** with **attributes ROLL_NO, NAME, ADDRESS, PHONE, and AGE**

Cont..

Table Student

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

- **Attribute:** Attributes are the properties that define an entity. e.g.; **ROLL_NO**, **NAME, ADDRESS**
- **Relation Schema:** A relation **schema defines the structure** of the relation.

Example: ROLL_NO, NAME, ADDRESS, PHONE, and AGE is the relation schema for STUDENT. If a **schema has more than 1 relation**, it is called **Relational Schema.** (if teacher comes it is RS)

Tuple: Each row in the relation is known as a tuple. The above relation contains 4 tuples,

Cont..

- **Column:** The column represents **the set of values for a particular attribute.**
- The column **ROLL_NO** is extracted from the relation STUDENT.

ROLL_NO
1
2
3
4

- **NULL Values:** The value which is not known or unavailable is called a NULL value. It is represented by blank space.
- **Eg:** PHONE of STUDENT having ROLL_NO 4 is NULL.

- **Relation Key:** These are basically the **keys that are used to identify the rows uniquely** in the table.
- These are of the following types.
- Primary Key
- Candidate Key
- Super Key
- Foreign Key
- Alternate Key
- Composite Key

Characteristics of the Relational Model

- Data is represented **in rows and columns** called **relations**.
- Data is **stored in tables having relationships between them** called the Relational model.
- The relational model supports the operations **like Data definition, Data manipulation, and Transaction management.**
- Each column **has a different name** and they are representing attributes.
- Each row represents a **single entity**.(person)

Data Definition Language (DDL)

- Data definition language is used to store the information

- **Create:** It is used to create tables in the database.
- **Alter:** It is used to alter **the structure of the database.(changes/edit) table .**
- **Drop:** It is used to delete table from the database.

(structure also deleted)

- **Truncate:** It is used to remove all records from a table.(only data)
- **Rename:** It is used to rename an table.

Data Manipulation Language (DML)

- It is used for accessing and manipulating data in a database. It handles user requests.

Some task under DML☹query

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.(new)
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.(eg :name) FN & LN →stuname

Data Control Language (DCL)

- **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored data.
- **Grant:** It is used to give user access privileges to a database.(**giving the permission to access the db**)
- **Revoke:** It is used to take back permissions from the user. (**getting back the permission is called revoke)**

Transaction Control Language (TCL)

- TCL is used to run the changes made by the DML statement.
- **Commit:** It is used to save the transaction on the database.(permanent save)
- **Save point :** You can create save points within a transaction.(temporary) If the transaction rolls back, **changes are undone to the specified save point.**
- **Rollback:** It is used to **restore the database to original** since the last Commit. **(back to original form)**

Example

Table :student

REGNO	NAME	DEPT	YEAR
101	X	Bca	II
102	Y	Bcom	II

Commit;

- insert into student values(103,’z’,’IT’,2);

REGNO	NAME	DEPT	YEAR
101	X	Bca	II
102	Y	Bcom	II
103	Z	IT	II

SAVEPOINT A; (temporary save)

- insert into student values(104’G’,’Interior design’,2);

REGNO	NAME	DEPT	YEAR
101	X	Bca	II
102	Y	Bcom	II
103	Z	IT	II
104	G	Interior design	II

SAVEPOINT B; (temporary save)

ROLL BACK

- Rollback to A;

REGNO	NAME	DEPT	YEAR
101	X	Bca	II
102	Y	Bcom	II
103	Z	IT	II

- Rollback to student ;

REGNO	NAME	DEPT	YEAR
101	X	Bca	II
102	Y	Bcom	II

RELATIONAL MODEL CONSTRAINS

INTEGRITY CONSTRAINTS

❖ Integrity constraints are **pre-defined set of rules that are applied on the table fields(columns) or relations** to ensure that the overall validity, integrity, and consistency of the data present in the database/table is maintained.

❖ It is used to **maintain the quality of information**.

INTEGRITY CONSTRAINTS

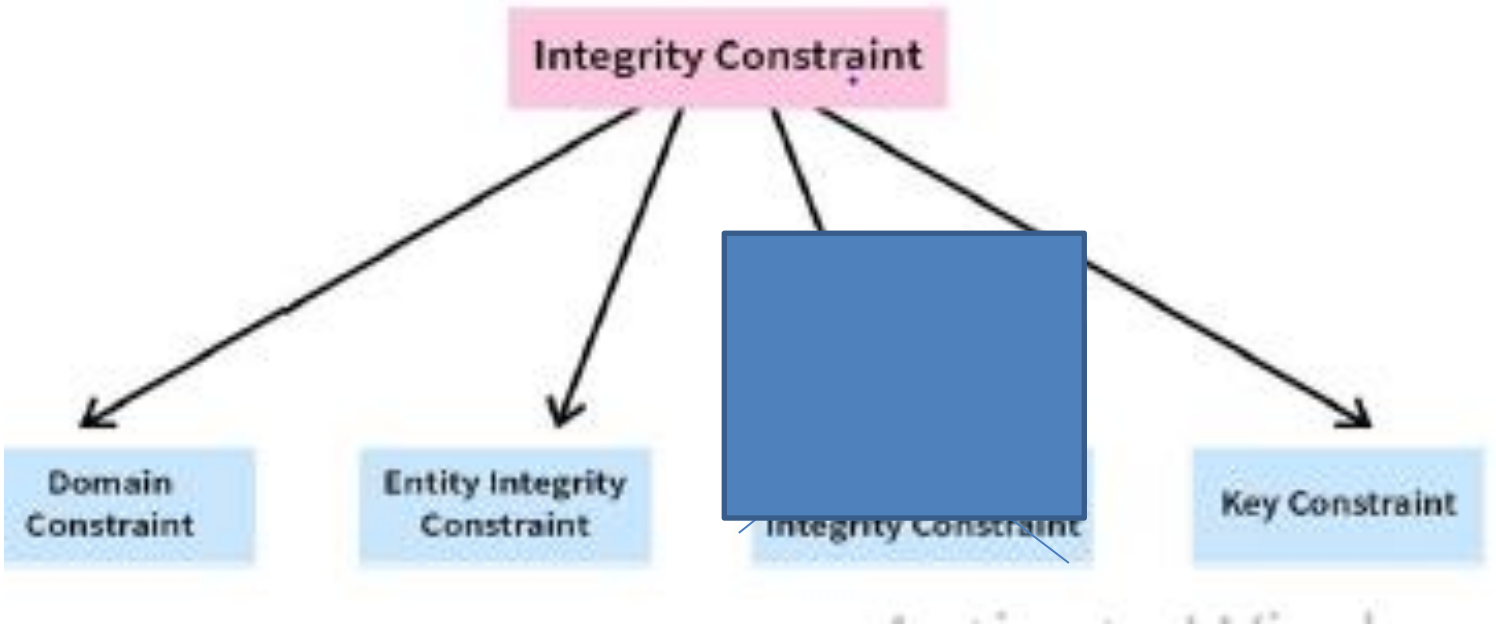
Constraints Used In Database:

Constraints are used in a database to specify the rules for data in a table. The following are the different types of constraints:

- ***NOT NULL:*** *This constraint ensures that a column cannot have a NULL value.*
- ***UNIQUE:*** *This constraint ensures that all the values in a column are unique.*
- ***CHECK:*** *This constraint ensures that all the values in a column satisfy a specific condition.*
- ***DEFAULT:*** *This constraint consists of a set of default values for a column when no value is specified*
- ***INDEX:*** *This constraint is used to create indexes in the table, through which you can create and retrieve data from the database very quickly.*

all values in a column that contains meaningful check under specific conditions

CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHAR(50), Salary DECIMAL(10, 2) **CHECK (Salary >= 0));**



Domain constraints

- The data type of domain includes varchar, character, integer, time, date, currency, etc. The **value of the attribute must be available in the corresponding domain.**

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

Entity integrity constraints:

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

ID	*NAME*	*SEMESTER*	*AGE*
10001	*PRAJWAL*	*2*	*19*
10002	*SUNNY*	*1*	*18*
	SHAILU	*3*	*20*

Not allowed as primary key can't contain a NULL value

Key constraints: Keys are the entity set that is used to identify an entity within its entity set uniquely.

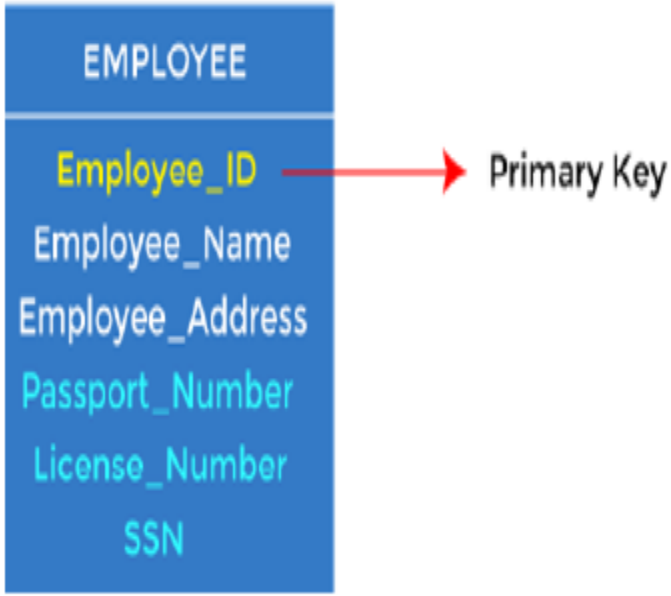
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table..

ID	NAME	SEMESTER	AGE
10001	PRAJWAL	2	19
10002	SUNNY	1	18
10002	SHAILU	3	20

Not allowed. Because all row must be unique

Primary key:

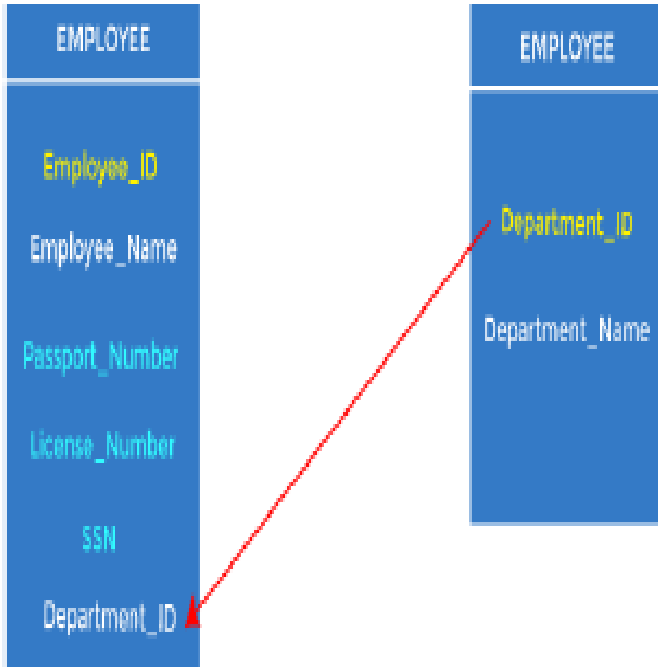
- ✓ It is a field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values.
- ✓ A primary key column cannot have NULL values.
- ✓ Ex: In the EMPLOYEE table, ID can be the primary key since it is unique for each employee.




```
1.CREATE TABLE SAMPLE_TABLE
2.(COL1 integer,
3.COL2 nvarchar(30),
4.COL3 nvarchar(50),
5.PRIMARY KEY (COL1));
```

Foreign key:

- ✓ A key used to link two tables together is called a foreign key, also called as referencing key.
- ✓ Foreign key is a field that matches the primary key column of another table.
- ✓ In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.

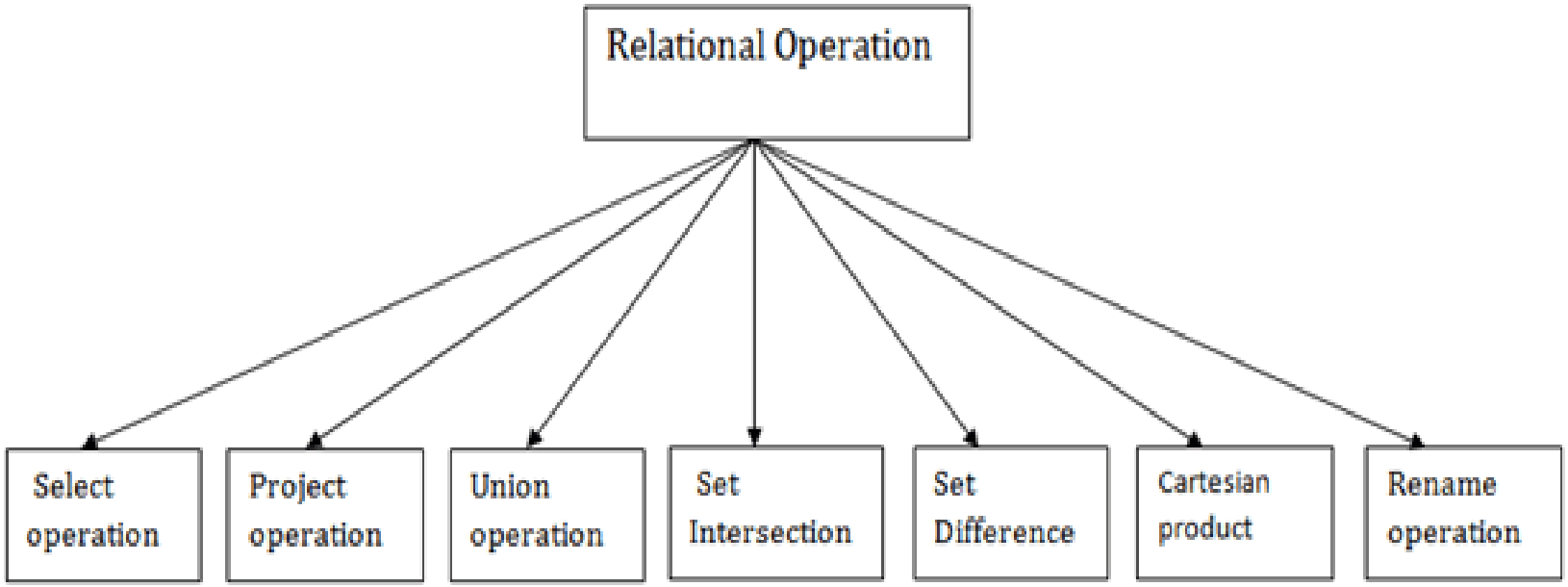



```
CREATE TABLE Employee
( Emp_Id int NOT NULL PRIMAY KEY,
EName Varchar(255) NOT NULL,
P_No int NOT NULL,
Dept_Id int FOREIGN KEY REFERENCES
Department(Dept_Id)
)
```

Relational Algebra

- Relational algebra operations are used to perform **various tasks such as retrieving, updating, and manipulating data stored in relational databases.**
- Relational algebra is a **procedural query language.**

Types of Relational operation



1. Select Operation

- The selection operation is used to retrieve rows from a relation that satisfy a given condition.
- It is denoted by sigma (σ).
- syntax is σ<condition>(relation).

Example:

scss Copy code

```
σ_salary > 50000(Employee)
```

Projection (π):

- The projection operation is used to retrieve specific columns from a relation.
- The syntax is π<column-list>(relation).
- It only shows empname, dept column

Example:

```scss
π_employeeName, department(Employee)
```

Union (∪):

- The union operation combines two relations and removes duplicate tuples.
- The relations must have the same set of attributes.
- The syntax is relation1 ∪ relation2.

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

Input:

∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.
- Notation: R ∩ S

Input:

∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

Set Difference:

- Suppose there are two tuples R and S. The set Difference operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).
- Notation: R - S

Input:

∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

Input:

```
∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)
```

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.
- Notation: E X D

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

Rename Operation:

- The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

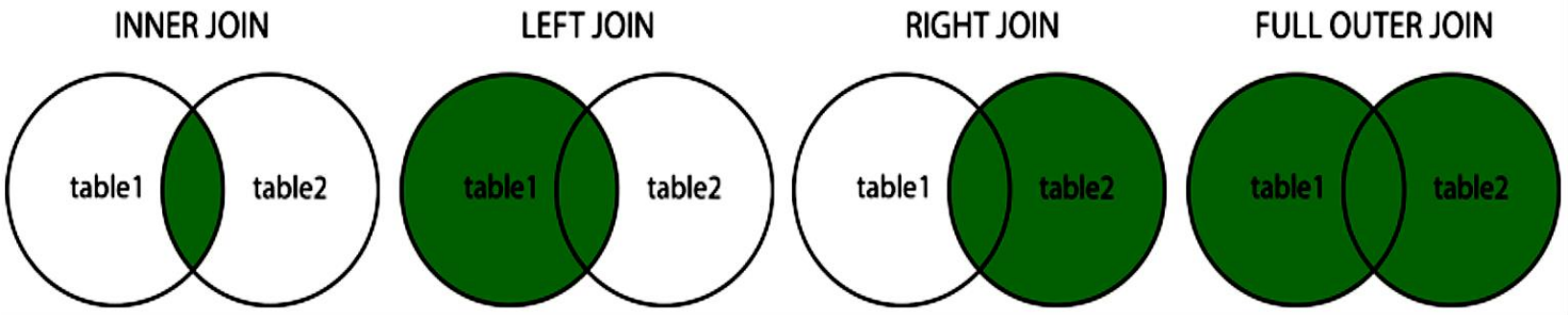
Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

ρ(STUDENT1, STUDENT)

JOIN OPERATION :

Joins in Database Management System are used in relational algebra and SQL to ***join/combine more than one table to get some specific results out of those tables.***

- A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ⋈.
- **Different types of the JOINs in SQL:**
  - ***(INNER) JOIN***
  - ***LEFT (OUTER) JOIN***
  - ***RIGHT (OUTER) JOIN***
  - ***FULL (OUTER) JOIN***



- **Different types of the JOINs in SQL:**
  - ***(INNER) JOIN:*** Returns records that have matching values in both tables
  - ***LEFT (OUTER) JOIN:*** Returns **all records from the left table,** and **the matched records from the right table**
  - ***RIGHT (OUTER) JOIN:*** Returns **all records from the right table,** and the **matched records from the left table**
  - ***FULL (OUTER) JOIN:*** **Returns all records when there is a match in either left or right table**

Sample table

EMPLOYEE

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	Angelina	Chicago	200000	30
2	Robert	Austin	300000	26
3	Christian	Denver	100000	42
4	Kristen	Washington	500000	29
5	Russell	Los angels	200000	36
6	Marry	Canada	600000	48

PROJECT

PROJECT_NO	EMP_ID	DEPARTMENT
101	1	Testing
102	2	Development
103	3	Designing
104	4	Development

INNER JOIN

Query

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
INNER JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development

LEFT OUTER JOIN

Query

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT

FROM EMPLOYEE

LEFT JOIN PROJECT

ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development
Russell	NULL
Marry	NULL

RIGHT OUTER JOIN

Query

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT
FROM EMPLOYEE
RIGHT JOIN PROJECT
ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

Output

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development

FULL OUTER JOIN

Query

```
SELECT EMPLOYEE.EMP_NAME, PROJECT.DEPARTMENT

FROM EMPLOYEE

FULL JOIN PROJECT

ON PROJECT.EMP_ID = EMPLOYEE.EMP_ID;
```

EMP_NAME	DEPARTMENT
Angelina	Testing
Robert	Development
Christian	Designing
Kristen	Development
Russell	NULL
Marry	NULL

NULL	Backend developer

AGGREGATE FUNCTIONS

- Aggregate functions in a DBMS operate on a set of values and return a single value as a result. These functions are often used in conjunction with the SELECT statement.

Here are some common aggregate functions:

- COUNT()
- SUM()
- AVG()
- MIN()
- MAX()

COUNT():

- Counts the number of rows in a set.

SUM():

- Calculates the sum of numeric values in a set.

AVG():

- Calculates the average of numeric values in a set.

MIN():

- Returns the minimum value in a set.

MAX():

- Returns the maximum value in a set.

QUERY

- SELECT **COUNT**(subject) FROM student;
- SELECT **SUM**(mark) FROM student;
- SELECT **AVG**(mark/ salary) FROM student;
- SELECT **MIN**(mark) FROM student;
- SELECT **MAX**(mark) FROM student;

- These functions can be used to perform calculations on data within specified columns in a table

GROUPING

- They are often used in combination with the GROUP BY clause to aggregate data based on specific criteria.
- Which is used group the records.
- Group By Clause is executed by row by row

- First we need to separate the data based on the group.

- **WAQToDisplay count of the employee in each department ?**

Select count(*),deptno from emp GROUP BY deptno;
(groupby expression)

EID	NAME	DEPTNO
1	A	10
2	B	20
3	C	30
4	D	30
5	E	20

OUTPUT

Count	Deptno
1	10
2	20
2	30

10		
1	A	10

20		
2	B	20
5	E	20

30		
3	C	30
4	D	30

Nested SubQuery

- A subquery inside another query is called a nested subquery.
- It is used when you need to filter or manipulate data from multiple tables and when the outcome of one query is based on the results of another.
- Upto 255 nested Sub queries can be used
- It executes innermost subquery first then next level.

Subquery example

Customer:

CID	NAME	AGE	COUNTRY
1	Safiya	18	India
2	Sandhiya	17	Usa
3	Sumiya	20	India
4	John	17	Uk
5	david	22	UAE

Select * from Customer where age = (Select min(Age) from customer);

CID	NAME	AGE	COUNTRY
2	Sandhiya	17	Usa
4	John	17	Uk

Subquery with operator

- Customer

CID	FIRST NAME
1	John
2	Robert
3	Sumaiya
4	Kavin

- Orders

ORDERID	CID	AMOUNT
101	2	200
102	1	100
103	10	500
104	12	400

Select * from customer WHERE cid IN (select cid from orders);

- Output

CID	FIRST NAME
1	John
2	Robert

Employee

EMPID	NAME	SALARY	DESIGNATION
301	James	2975	Salesman
302	Smith	5000	Manager
303	King	3000	Analyst
304	Adams	800	clerk

Find Second maximum salary?

```
Select max(sal) from emp
where sal < (select max(sal) from emp);
```

Output : 3000

```
+-----+-------+
| emp |  sal  |
+-----+-------+
| 101 | 5000  |
| 102 | 6000  |
| 103 | 5500  |
| 104 | 7000  |
| 105 | 6500  |
+-----+-------+
```

```
SELECT MAX(sal) FROM emp WHERE sal <
   (SELECT MAX(sal) FROM emp);
```

Subquery: SELECT MAX(sal) FROM emp;

Result: The maximum salary in the entire "emp" table is 7000.

Main Query: SELECT MAX(sal) FROM emp

WHERE sal (5000 ,6000,6500) < 7000;

Result: The query filters rows where the salary is less than 7000. **So, it considers the salaries 5000, 6000, and 6500.** The **maximum salary among these is 6500.**

```
+------------+
| MAX(sal)   |
+------------+
| 6500       |
+------------+
```

- **Find Third maximum salary?**

```
Select max(sal) from emp
where sal < (select max(sal) from emp
 where sal < (Select max(Sal) from emp ));
```

Output :

2975

Views

- Views in SQL are considered as a **virtual table.** A view also contains **rows and columns.**
- To create the view, we can select the fields **from one or more tables present in the database.**
- A view can **either have specific rows based** on certain condition or **all the rows of a table.**

STUDENT DETAILS

STU_ID	NAME	ADDRESS
1	Stephan	Delhi
2	Kathrin	Noida
3	David	Ghaziabad
4	Alina	Gurugram

STUDENT MARKS

STU_ID	NAME	MARKS	AGE
1	Stephan	97	19
2	Kathrin	86	21
3	David	74	18
4	Alina	90	20
5	John	96	18

1. CREATING VIEW From Single Table

- A view can be created using the **CREATE VIEW** statement.
- We can create a view from a **single table or multiple tables.**
- **CREATE VIEW** DetailsView **AS SELECT** NAME, ADDRESS **FROM** Student_Details **WHERE** STU_ID < 4;

Output:

Output:

NAME	ADDRESS
Stephan	Delhi
Kathrin	Noida
David	Ghaziabad

CREATING VIEW FROM MULTIPLE TABLES

- View from multiple tables can be created by simply include **multiple tables in the SELECT statement.**

- **CREATE VIEW** MarksView **AS SELECT**

Student_Detail.NAME, Student_Detail.ADDRESS

Student_Marks.MARKS **FROM** Student_Detail,

Student_Mark **WHERE**

Student_Detail.**NAME** =Student_Marks.**NAME**;

OUTPUT

NAME	ADDRESS	MARKS
Stephan	Delhi	97
Kathrin	Noida	86
David	Ghaziabad	74
Alina	Gurugram	90

Deleting View :

A view can be deleted using the Drop View statement.

Syntax : **DROP VIEW** view_name;

Eg: **DROP VIEW** MarksView;

What is PL/SQL?

- PL/SQL stands for Procedural Language extension of SQL.
- It was developed by Oracle Corporation in the late 1980s to enhance the capabilities of SQL.
- PL/SQL is a combination of SQL along with the procedural features of programming languages
- Every PL/SQL statement will be followed by semicolon (;).
- PL/SQL blocks can be nested.

PL/SQL block structure:

```
DECLARE
   Declaration statements;
BEGIN
   Execution statements;
EXCEPTION
   Exception handling statements;
END;
/
```

/ (slash): This symbol signifies the end of the PL/SQL block and executes the block.

PL/SQL Block sections

1. Declaration section (optional).
2. Execution section (mandatory).
3. Exception handling section (optional).

DECLARATION SECTION:

- It is an optional section and starts with DECLARE keyword. It **is used to declare the variables, constants, records and cursors etc.**

EXECUTION SECTION:

- Execution section starts with BEGIN keyword and ends with END keyword.
- It is a mandatory section. **It is used to write the program logic code.**

Note: Execution section must have one statement.

EXCEPTION HANDLING SECTION:

- Execution section starts with EXCEPTION keyword. It is an optional section.
- It is used to **handle the exceptions occurred in execution section.**

Advantages of PL/SQL:

- PL/SQL is a procedural language.
- PL/SQL is a block structure language
- PL/SQL handles the exceptions.
- PL/SQL engine can process the multiple SQL statements simultaneously as a single block.
- It provides better performance.

```
-- PL/SQL program to calculate the square of a number

-- Declaration section
DECLARE
   -- Declare variables
   v_number NUMBER;
   v_square NUMBER;

-- Execution section
BEGIN
   -- Get user input for a number
   DBMS_OUTPUT.PUT_LINE('Enter a number: ');
   v_number := &user_input; -- &user_input is a placeholder for user input

   -- Calculate the square of the number
   v_square := v_number * v_number;

   -- Display the result
   DBMS_OUTPUT.PUT_LINE('The square of ' || v_number || ' is: ' || v_square);

-- Exception handling section (optional)
EXCEPTION
   WHEN OTHERS THEN
      -- Handle exceptions (errors) if they occur
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

PL/SQL PROGRAM TO CALCULATE THE AREA OF A RECTANGLE

```
-- Declare variables to store length, width, and area
DECLARE
  l_length NUMBER := 5; -- Length of the rectangle
  l_width  NUMBER := 3; -- Width of the rectangle
  l_area   NUMBER;     -- Area of the rectangle
BEGIN
  -- Calculate the area of the rectangle
  l_area := l_length * l_width;

  -- Display the input parameters and the calculated area
  DBMS_OUTPUT.PUT_LINE('Length: ' || l_length);
  DBMS_OUTPUT.PUT_LINE('Width: ' || l_width);
  DBMS_OUTPUT.PUT_LINE('Area: ' || l_area);
END;
/
```

OUTPUT

Length: 5

Width: 3

Area: 15