Assignment_3

Name: Shikha Singh
Roll no: 25

ASSIGNMENT - 3

Name :- Shikha Singh
Roll no :- 25

1. What is join ? Explain types of join ? Demonstrate any one example.

→ In SQL, a "join" is an operation used to combine rows from two or more tables based on a related column between them. Joins are fundamental for querying relational databases where data is often distributed across multiple tables. By using joins you can retrieve and combine this data into single result set.

Types of Joins :-

1. INNER JOIN : This join returns only the rows where there is a match between the columns in both tables. If a row in either table does not have a corresponding match, it will not appear in the result set.
2. LEFT JOIN (or LEFT OUTER JOIN) : This join returns all rows from the ~~left~~ ~~left~~ right table and the matched rows from the right table. If there is no match, the result is NULL on the side of the right table.
3. RIGHT JOIN (or RIGHT OUTER JOIN) : This join returns all rows from the right table and the matched rows from the left table. If there is no match

the result is NULL on the side of the left table.

4. FULL JOIN (or FULL OUTER JOIN) : This join returns rows when there is a match in either table. If there is no match, the result will include NULLs in columns from the table without a match.

5. CROSS JOIN: This join returns the Cartesian product of the two tables, meaning it will return all possible combinations of rows from both tables. It does not require any condition to join.

6. SELF JOIN : This is a special case where a table is joined with itself. It is useful for querying hierarchical data or comparing rows within the same data.

Example :-

→ Employees table :-

EmployeeID	Name	DepartmentID
1	Alice	10
2	Bob	20
3	Charlie	10

→ Departments table :-

DepartmentID	Departname
10	HR
20	Engineering

SQL :-

```
SELECT Employees.Name , Departments.DepartmentName
FROM Employees
INNER JOIN Departments ON Employees.DepartmentID
    = Departments.DepartmentID
```

(2.) What is subqueries ? Explain its types and Demonstrate any one example.

→ A subquery, also known as a nested query, is a query embedded within another SQL query. subqueries are used to perform operations that require multiple steps or to retrieve data that is then used by the outer query. They can be used in various clauses like SELECT, WHERE, FROM, and HAVING.

* Types of Subqueries :-

1. Single-Row Subquery :- Returns a single row and a single column. Typically used with comparison operators like =, <, >, etc.
2. Multiple-Row Subquery : Returns multiple rows but typically only one column.

rows and multiple columns. Used when comparing against a set of rows and columns.

4. Correlated Subquery: The subquery references columns from the outer query. It is executed once for each row processed by the outer query.

5. Uncorrelated Subquery: The subquery does not reference any columns from the outer query and can be executed independently.

Example:-

Orders table:

OrderID	CustomerID	Order Amount
1	101	250
2	102	150
3	101	300

Customer Table:

CustomerID	Customer Name
101	John Doe
102	Jane Smith

Sql :-

```
SELECT      CustomerName
FROM  Customers
WHERE  CustomerID IN (
    SELECT  CustomerID
```

Result:

CustomerName

John Doe.

In this example, the subquery retrieves CustomerID s of orders with OrderAmount greater than 200. The outer query then uses these IDs to find the corresponding customer names.

3. What is view ? explain in detail?

→ A view is a virtual table in SQL that is defined by a query. It does not store data physically but provides a way to present data from one or more tables in a specific format. Views can simplify complex queries, provide security by restricting access to specific data, and present aggregated or derived data.

key points:

→ Definition: A view is defined by a SELECT query and can include data from one or multiple tables.

→ Usage: You can use views just like tables in queries. They can be used to simplify complex joins and aggregations

SQL :-

```
CREATE VIEW HighValueOrders AS
SELECT Customers.CustomerName, Orders.OrderAmount
FROM Orders
JOIN Customers ON orders.CustomerID =
                  Customers.CustomerID
WHERE Orders.OrderAmount > 200;
```

* Use the view :-

```
SELECT * FROM HighValueOrders;
```

(4) What is Trigger and stored procedure explain in detail.

→ Trigger :

A trigger is a special type of stored procedure that is automatically executed in response to certain events on a table or view, such as INSERT, UPDATE, or DELETE operations. Triggers are used to enforce business rules, validate data, or maintain audit logs.

Key points :-

→ Types : There are three main types of

→ AFTER Trigger : Executes after the triggering event.

→ INSTEAD OF Trigger : Executes in place of the triggering ~~es~~ event.

- Usage : Triggers are useful for tasks like enforcing referential integrity, logging changes and automatically updating related tables.

Example :-

```
CREATE TRIGGER LogOrderInsertion
AFTER INSERT ON Orders
FOR EACH ROW
BEGIN
    INSERT INTO OrderLog (OrderID,
                          changeTime, Action)
    VALUES (NEW.OrderID, NOW(), 'INSERT');
END;
```

2. Stored Procedure :-

A store procedure is a precomplied collection of one or more SQL Statements ~~that~~ that can be executed as a single unit. Stored procedures can accept parameters, perform operations, and return results.

logic, improve performance, and simplify application development.

Characteristics :-

→ Encapsulation :- stored procedures encapsulate business logic within the database.

→ Reusability :- They can be reused across different applications and reduce redundancy.

→ Performance : Since they are precomplied, they often execute more efficiently than sending multiple individual SQL statements.

→ Security :- They can help to secure database operations by limiting direct access to the database tables.

Example use cases :-

→ Data retrieval :- Execute complex queries with parameters to fetch data.

→ Data manipulation : Perform multiple INSERT, UPDATE, or DELETE operations.

→ Transaction Management : Control transactions and ensure data consistency.

Example :

```
CREATE PROCEDURE GetEmployeeById (IN empId INT)
BEGIN
    SELECT * FROM Employees
    WHERE EmployeeID = empId;
END;
```