Session 7:

Topic1: Views

VIEW is a virtual table, through which a selective portion of the data from one or more tables. Views in SQL offer advantages like data security, simplified complexity, and data consistency, but can also lead to performance issues.

A view can be created from one or many tables. Unless indexed, a view does not exist in a database.

The SQL CREATE VIEW Statement

Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2....
FROM table_name
WHERE [condition];
```

Example: created a table named CUSTOMERS using the **CREATE TABLE** statement

```
CREATE TABLE CUSTOMERS(
  ID   INT              NOT NULL,
  NAME VARCHAR (20)     NOT NULL,
  AGE  INT              NOT NULL,
  ADDRESS  CHAR (25) ,
  SALARY   DECIMAL (18, 2),
  PRIMARY KEY (ID)
);
```

insert values into this table using the INSERT statement

```
INSERT INTO CUSTOMERS VALUES
(1, 'Ramesh', 32, 'Ahmedabad', 2000.00 ),
(2, 'Khilan', 25, 'Delhi', 1500.00 ),
(3, 'Kaushik', 23, 'Kota', 2000.00 ),
(4, 'Chaitali', 25, 'Mumbai', 6500.00 ),
(5, 'Hardik', 27, 'Bhopal', 8500.00 ),
```

(6, 'Komal', 22, 'Hyderabad', 4500.00),

(7, 'Muffy', 24, 'Indore', 10000.00);

Creates a view based on the above created table –

CREATE VIEW CUSTOMERS_VIEW AS SELECT * FROM CUSTOMERS;

verify the contents of a view using the select query

SELECT * FROM CUSTOMERS_VIEW;

Advantages of views

1.Security

Each user can be given permission to access the database only through a small set of views that contain the specific data the user is authorized to see, thus restricting the user's access to stored data

2.Query Simplicity

A view can draw data from several different tables and present it as a single table, turning multi-table queries into single-table queries against the view.

3.Structural simplicity

Views can give a user a "personalized" view of the database structure, presenting the database as a set of virtual tables that make sense for that user.

4.Consistency

A view can present a consistent, unchanged image of the structure of the database, even if the underlying source tables are split, restructured, or renamed.

5.Data Integrity

If data is accessed and entered through a view, the DBMS can automatically check the data to ensure that it meets the specified integrity constraints.

6.Logical data independence.

View can make the application and database tables to a certain extent independent. If there is no view, the application must be based on a table. With the view, the program can be established in view of above, to view the program with a database table to be separated.

Disadvantages of views

1.Performance

Views create the appearance of a table, but the DBMS must still translate queries against the view into queries against the underlying source tables. If the view is defined by a complex, multi-table query then simple queries on the views may take considerable time.

2.Update restrictions

When a user tries to update rows of a view, the DBMS must translate the request into an update on rows of the underlying source tables. This is possible for simple views, but more complex views are often restricted to read-only.

Topic2: Triggers

A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

Triggers are used to enforce business rules, maintain data integrity, and automate certain actions within a database. They can be triggered by various events, such as inserting, updating, or deleting data in a table, and they allow you to perform additional operations based on those events.

Because a trigger cannot be called directly, unlike a stored procedure, it is referred to as a special procedure. A trigger is automatically called whenever a data modification event against a table takes place, which is the main distinction between a trigger and a procedure.

The differences between triggers and stored procedures:

1. Triggers cannot be manually invoked or executed.
2. There is no chance that triggers will receive parameters.
3. A transaction cannot be committed or rolled back inside a trigger.

Syntax:

```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]
```

TYPES OF TRIGGERS

1. **DDL Trigger**

   The Data Definition Language (DDL) command events such as Create_table, Create_view, drop_table, Drop_view, and Alter_table cause the DDL triggers to be activated.

2. **DML Trigger**

3. **Logon Triggers**

Advantage of Triggers

Data integrity: Triggers allow you to enforce complex business rules and constraints at the database level, ensuring that data remains consistent and accurate.

Automation: Triggers can automate repetitive or complex tasks by executing predefined actions whenever a specified event occurs. This reduces the need for manual intervention and improves efficiency.

Audit trails: Triggers can be used to track changes made to data, such as logging modifications in a separate audit table. This helps in auditing and maintaining a history of data changes.

Data validation: Triggers can perform additional validation checks on data before it is inserted, updated, or deleted, ensuring that only valid and conforming data is stored in the database.

Disadvantage of Triggers

A. Only triggers permit the use of extended validations.
B. Automatic triggers are used, and the user is unaware of when they are being executed. Consequently, it is difficult to troubleshoot issues that arise in the database layer.
C. The database server's overhead may increase as a result of triggers.
D. In a single CREATE TRIGGER statement, we can specify the same trigger action for multiple user actions, such as INSERT and UPDATE.
E. Only the current database is available for creating triggers, but they can still make references to objects outside the database.

Topic3: Window Functions

windows functions are SQL functions that enable us to perform operations on a window, a set of records. The window word represents the group of rows on which the function will be operated. SQL Server categorizes the window functions into mainly three types:

Types of Window Functions

A. Aggregate window functions

These functions operated on multiple rows and Examples of such functions are SUM(), MAX(), MIN(), AVG(), COUNT(), etc.

```
CREATE TABLE Product_Sales(
   Emp_Name VARCHAR(45) NOT NULL,
   Year INT NOT NULL,
   Country VARCHAR(45) NOT NULL,
   Prod_name VARCHAR(45) NOT NULL,
   Sales DECIMAL(12,2) NOT NULL,
   PRIMARY KEY(Emp_Name, Year)
);
INSERT INTO Product_Sales(Emp_Name, Year, Country, Prod_name, Sales)
VALUES('Mike Johnson', 2017, 'Britain', 'Laptop', 10000),
('Mike Johnson', 2018, 'Britain', 'Laptop', 15000),
('Mike Johnson', 2019, 'Britain', 'TV', 20000),
('Mary Greenspan', 2017, 'Australia', 'Computer', 15000),
('Mary Greenspan', 2018, 'Australia', 'Computer', 10000),
('Mary Greenspan', 2019, 'Australia', 'TV', 20000),
('Nancy Jackson', 2017, 'Canada', 'Mobile', 20000),
('Nancy Jackson', 2018, 'Canada', 'Calculator', 1500),
('Nancy Jackson', 2019, 'Canada', 'Mobile', 25000);
```

Aggregate Window Function

SUM()

It is an aggregate function that performs the addition of the specified field for a specified group or the entire table when we have not specified any group.

```
SELECT Country, SUM(Sales) AS total_amount
FROM Product_Sales GROUP BY Country;
SELECT Emp_Name, Year, Country, Prod_name, Sales, SUM(Sales)
OVER(PARTITION BY Country) as grand_total
FROM Product_Sales;
```

B. Value window functionsss

C. Ranking window functions

Topic4: Case statement

The CASE is a statement that operates if-then-else type of logical queries. This statement returns the value when the specified condition evaluates to True. When no condition evaluates to True, it returns the value of the ELSE part.

In Structured Query Language, CASE statement is used in SELECT, INSERT, and DELETE statements with the following three clauses:

- WHERE Clause
- ORDER BY Clause
- GROUP BY Clause

The CASE statement is of two types in relational databases:

- Simple CASE statement
- Searched CASE statement

Syntax of CASE statement in SQL

CASE <expression>

WHEN condition_1 THEN statement_1

WHEN condition_2 THEN statement_2 …….

WHEN condition_N THEN statement_N

ELSE result

END;

Examples of CASE statement in SQL

Roll_No	Stu_Name	Stu_Subject	Stu_Marks	Stu_City
2001	Akshay	Science	92	Noida
2002	Ram	Math	49	Jaipur
2004	Shyam	English	52	Gurgaon
2005	Yatin	Hindi	45	Lucknow
2006	Manoj	Computer	70	Ghaziabad
2007	Sheetal	Math	82	Noida
2008	Parul	Science	62	Gurgaon
2009	Yogesh	English	42	Lucknow
2010	Ram	Computer	88	Delhi
2011	Shyam	Hindi	35	Kanpur

Ex1:

```
SELECT Roll_No, Stu_Name, Stu_Subject, Stu_marks,
CASE
WHEN Stu_Marks >= 50 THEN 'Student_Passed'
ELSE 'Student_Failed'
END AS Student_Result
FROM Student_Details;
```

Ex2:

```
SELECT Roll_No, Stu_Name, Stu_Subject, Stu_marks,
CASE
WHEN Stu_Marks >= 90 THEN 'Outstanding'
WHEN Stu_Marks >= 80 AND Stu_Marks < 90 THEN 'Excellent'
WHEN Stu_Marks >= 70 AND Stu_Marks < 80 THEN 'Good'
WHEN Stu_Marks >= 60 AND Stu_Marks < 70 THEN 'Average'
WHEN Stu_Marks >= 50 AND Stu_Marks < 60 THEN 'Bad'
WHEN Stu_Marks < 50 THEN 'Failed'
END AS Stu_Remarks
FROM Student_Details;
```