**Day – 4**

* Views, Stored Procedures, Functions
* Triggers, Cursors
* Transactions & ACID
* Indexing
* Query Optimization (EXPLAIN plans)
* Window Functions (ROW\_NUMBER, RANK, LAG, LEAD)

**✅ 1. View – A Saved SELECT Query**

**📌 What is it?**

A **View** is like a **virtual table** that shows data from real tables.

It doesn't store data — it just **shows a saved query result**.

**💡 Easy Use Cases:**

* Show only **some columns** from a big table
* Make a **complex query easy to reuse**

And you want to show only HR employees:

CREATE VIEW HR\_Employees AS

SELECT Name, Salary

FROM Employee

WHERE Department = 'HR';

Now just use:

SELECT \* FROM HR\_Employees;

✅ It will always give you updated HR data!

**✅ Where and How a View is Saved?**

**🔹 A View is saved in the database itself, just like a table or a stored procedure.**

But —

* It **does not store actual data**
* It only stores the **SQL query** (like: SELECT ... FROM ... WHERE ...)

**📦 Where is it stored?**

👉 In the **system catalog** or **metadata** of the database.

**✅ What is a Stored Procedure?**

**📌 Simple Definition:**

A **Stored Procedure** is a **saved block of SQL code** (like SELECT, INSERT, UPDATE, DELETE) that you can run anytime to **do a task**.

It's like a **reusable button** inside the database.

**🧠 Why Use a Stored Procedure?**

* To **save time** by not writing the same query again
* To **automate a task** (like inserting today's sales)
* To **accept input values** (e.g., show employees from a selected department)
* To **handle logic** (like IF, LOOP, etc.)

**Created first (historically and conceptually)**

**✅ View:**

Just saves a **SELECT query**  
Used to **read** data only  
Like a **shortcut to a query**  
You create it to avoid writing the **same SELECT** again and again

**Came later, for more powerful and flexible use**

**✅ Stored Procedure:**

Saves **multiple SQL commands** (SELECT, INSERT, DELETE, logic, etc.)  
Can **take inputs**, **do actions**, and return results  
Like a **mini-program** stored in the database  
You create it to **do a task** without repeating the same code

**(CREATE PROCEDURE GetTopEmployees**

**AS**

**BEGIN**

**SELECT \* FROM Employees WHERE Rating > 4;**

**END**

**-- Call it**

**EXEC GetTopEmployees;)**

**So your thought is 💯 correct:**

🧾 View = Created to **store and reuse SELECT queries**, avoid writing same query again and again.  
⚙️ Stored Procedure = Created to **store big, complete sets of SQL code** with logic, so full jobs/processes can run without repeating steps.

**✅ What is a Function in SQL?**

**📌 Simple Definition:**

A **Function** is a mini calculator in SQL that **takes input**, does a **calculation or logic**, and **returns a result** (one value or a table).

**CREATE FUNCTION GetAnnualSalary(@Monthly INT)**

**RETURNS INT**

**AS**

**BEGIN**

**RETURN @Monthly \* 12**

**END**

**-- Use it inside SELECT**

**SELECT Name, dbo.GetAnnualSalary(Salary) FROM Employees;**

**🎯 Why Do We Use Functions?**

* To **reuse calculations or logic**
* To **simplify complex expressions**
* To make queries **clean and modular**
* **Use Inside SELECT, WHERE, JOIN, etc.**

✅ Unlike **Stored Procedures**, **Functions** can be used directly **inside**:

* SELECT
* WHERE
* JOIN
* ORDER BY

**🧠 Real-Life Analogy:**

* View = Saved **SELECT** query
* Stored Procedure = Saved **job or task**
* Function = Saved **formula or calculator**

**📂 Types of Functions:**

| **Type** | **What it does** | **Example** |
| --- | --- | --- |
| **Scalar Function** | Returns **one value** | E.g. returns full name, tax, salary total |
| **Table-Valued Function (TVF)** | Returns a **table** | E.g. returns all employees from one department |

**🔍 Examples:**

**📂 Types of Functions:**

| **Type** | **What it does** | **Example** |
| --- | --- | --- |
| **Scalar Function** | Returns **one value** | E.g. returns full name, tax, salary total |
| **Table-Valued Function (TVF)** | Returns a **table** | E.g. returns all employees from one department |

✔ **All 3** (View, Function, Stored Proc) are part of **SQL optimization and modular design**.  
✔ Using them properly **improves**:

* 🔍 **Query readability**
* ♻️ **Reusability**
* 🧹 **Clean logic separation**
* ✅ **Data accuracy & performance**