# Module 13

## What to learn

UDF
Create User-defined Functions
Limitation and restriction of User-defined Functions
Scalar Functions
Table-Valued Functions
Drop Function
Alter Function
Built-in system functions
Best Practices

## Practice Exercise

**Practice 1**
Do the hands on the things provided in videos and ppt and on msdn
**Practice 2**

### Fetching Data with Row_number() Function

Using a sample table named `Products` with the schema: **ProductID (int), ProductName (varchar), Category (varchar), Price (decimal)**, write an SQL query to assign a unique row number to each product within its category based on ascending price order. This will help identify the relative ranking of products in each category.

**Things Covered:** Ranking Functions - Row_number()

**Practice 3**

### Finding Top Performers with Rank() Function

Given a table named `StudentScores` with the schema: **StudentID (int), StudentName (varchar), Subject (varchar), Score (int)**, write an SQL query to rank students within each subject based on their scores. If two or more students have the same score, they should share the same rank, and the next rank should be skipped.

**Things Covered:** Ranking Functions - Rank()

**Practice 4**

### Calculating Densely Packed Ranks with Dense_rank() Function

Using an example table named `EmployeePerformance` having the schema: **EmployeeID (int), EmployeeName (varchar), Department (varchar), PerformanceScore (int)**, write a query to assign dense ranks to employees within each department based on their performance scores. Ensure that no rank numbers are skipped when there are ties.

**Things Covered:** Ranking Functions - Dense_rank()

Practice 5

## Aggregating and Analyzing Sales Data with Aggregate Functions

Use a table named `SalesData` having the schema: **SaleID (int), SaleAmount (decimal), SaleDate (date), Region (varchar)**. Write SQL queries to calculate the total sales, count the number of sales transactions, find the average sale amount, and determine the maximum and minimum sales for the entire dataset.

**Things Covered:** Aggregate Function - SUM/COUNT/AVG/MAX/MIN

Practice 6

## Grouping Sales by Region and Filtering by Minimal Criteria

Use the `SalesData` table with the schema: **SaleID (int), SaleAmount (decimal), SaleDate (date), Region (varchar)**. Write an SQL query to group the sales data by region. Calculate the total sales for each region and include only those regions where the total sales exceed 5000 in the result.

**Things Covered:** Group by/Having

Practice 7

## Generating Aggregated Subtotals and Grand Totals Using Rollup

Given the same `SalesData` table, write an SQL query to use the ROLLUP feature to generate subtotals for each region and a grand total for all regions.

**Things Covered:** Group by/ROLLUP

### Assignment Exercise

Assignment 1

Create a scaler Function to compute PF which will accept parameter basicsalary and compute PF. PF is 12% of the basic salary.

Assignment 2

Create a scaler Function which will compute PT which will accept MontlyEarning. Criteria as below. 1 Less Than Rs. 6,000/- NIL 2 Rs. 6,000/- or Above but less than

Rs. 9,000/- 80/- 3 Rs. 9,000/- or Above but less than Rs. 12,000/- 150/- 4 Rs.12,000/- or Above 200/

Assignment 3

## Customer Sales Data Management and Reporting

You have been hired as a database analyst to manage and analyze sales data for a retail company. Your task is to create the necessary tables, populate them with data, and perform analytical queries. Below are the details of the requirements:

### Schema Details:

`Customers`: CustomerID (Auto Increment, Primary Key), CustomerName (varchar), CustomerRegion (varchar), RegistrationDate (date)
`Sales`: SaleID (Auto Increment, Primary Key), CustomerID (Foreign Key), SaleAmount (decimal), SaleDate (date), ProductCategory (varchar)
`Payments`: PaymentID (Auto Increment, Primary Key), CustomerID (Foreign Key), PaymentAmount (decimal), PaymentDate (date), PaymentMethod (varchar)

### Tasks:

Create and populate the `Customers`, `Sales`, and `Payments` tables with at least 10 rows each.
Using the Ranking Function `ROW_NUMBER()`, list the top 5 highest-value sales transactions for each product category.
Using the Ranking Function `RANK()`, rank customers in each region based on their total sales. If two customers within the same region have the same sales total, they should share the same rank, with the next rank skipped.
Using the `ROLLUP` operator, calculate subtotals for total sales grouped by region and a grand total for all regions.
Generate a report of customers' total sales and their average sales amount using Aggregate Functions (`SUM` and `AVG`). Display the results grouped by the customer.
Create a view that shows each customer's total payment transactions and the number of purchases they made. Ensure no duplicate records appear.
Create a view that lists customers who made an online payment and bought products in the 'Electronics' category after a specific date (e.g., '2023-01-01').
Update the records of a customer who changed their region or sales details (e.g., Update a customer's region from 'North' to 'South').

### Functional and Business Logic Overview:

The solution should allow easy insights into customer behaviors and help identify profitable customers, high-value sales transactions, and effective payment methods. Use appropriate primary and foreign key constraints to model relationships correctly and ensure data integrity. Use indexes where necessary to improve query performance.

Online Reference

https://docs.microsoft.com/en-us/sql/relational-databases/user-defined-functi…