User-Defined Functions (UDFs) in SQL are custom functions that allow you to encapsulate logic for reuse across multiple queries. These functions can be **scalar functions** (return a single value) or **table-valued functions** (return a table). Here’s an explanation with a real-life example:

**Scenario: E-Commerce Platform**

Imagine an e-commerce platform where customers earn loyalty points based on their purchase amount. For simplicity:

* Customers earn 1 point for every $10 spent.
* If the total exceeds $500, they get a 10% bonus on loyalty points.

Instead of calculating loyalty points in every query, you can create a **scalar UDF**.

CREATE FUNCTION CalculateLoyaltyPoints(@PurchaseAmount DECIMAL(10, 2))

RETURNS INT

AS

BEGIN

DECLARE @Points INT;

DECLARE @Bonus INT;

-- Calculate base points

SET @Points = FLOOR(@PurchaseAmount / 10);

-- Calculate bonus points if applicable

IF @PurchaseAmount > 500

BEGIN

SET @Bonus = FLOOR(@Points \* 0.1); -- 10% bonus

SET @Points = @Points + @Bonus;

END

RETURN @Points;

END;

**Orders Table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| OrderID | INT | Unique identifier for each order. |
| CustomerID | INT | Unique identifier for the customer. |
| PurchaseAmount | DECIMAL(10, 2) | Total amount spent in the order. |
| OrderDate | DATE | Date the order was placed. |

**Sample Data**

| **OrderID** | **CustomerID** | **PurchaseAmount** | **OrderDate** |
| --- | --- | --- | --- |
| 1 | 101 | 120.00 | 2025-01-01 |
| 2 | 102 | 540.00 | 2025-01-02 |
| 3 | 103 | 30.00 | 2025-01-03 |
| 4 | 101 | 600.00 | 2025-01-04 |
| 5 | 104 | 480.00 | 2025-01-05 |

**Use the Function in Queries**

You can now use the CalculateLoyaltyPoints function to compute loyalty points dynamically:

SELECT

CustomerID,

PurchaseAmount,

dbo.CalculateLoyaltyPoints(PurchaseAmount) AS LoyaltyPoints

FROM

Orders;

**Output**

| **OrderID** | **CustomerID** | **PurchaseAmount** | **LoyaltyPoints** |
| --- | --- | --- | --- |
| 1 | 101 | 120.00 | 12 |
| 2 | 102 | 540.00 | 59 |
| 3 | 103 | 30.00 | 3 |
| 4 | 101 | 600.00 | 66 |
| 5 | 104 | 480.00 | 48 |