**Assignment instructions**

Please follow the instructions below to complete the assignment:

1. Connect to the AdventureWorks database:
   * If you don't have the AdventureWorks database installed, you can download it from Microsoft's website.
   * Use the appropriate connection string or method to connect to the database.
   * Add a column called CalculatedTotalDue of type Money to **SalesOrderHeader**.
2. Create a new stored procedure named **sp\_OrderProcessing**:
   * The stored procedure should declare a cursor that selects all the sales orders from the **Sales.SalesOrderHeader** table.
   * Order the results by the **OrderDate** column in ascending order.
   * Make sure to define appropriate variables to store the necessary data from the cursor.
3. Fetch and process the cursor rows:
   * Use a loop structure to fetch each row from the cursor.
   * Store the relevant data from each row into the previously defined variables.
   * Within the loop, perform the following tasks for each row: a. Calculate the total order amount (sum of **SubTotal** and **TaxAmt** columns) for the current order.
   * b. Update the **CalculatedTotaleDue** column in the **SalesOrderHeader** table with the calculated total order amount for the current order.
4. Close and deallocate the cursor:
   * After processing all the rows, close and deallocate the cursor to release resources.
5. Test the stored procedure:
   * Execute the stored procedure **sp\_OrderProcessing** to ensure it runs without errors.
   * Verify that the **SalesOrderHeader** column in the **SalesOrderHeader** table has been updated correctly.
6. Write a short summary:
   * Write a brief summary explaining the purpose of using cursors in this scenario and discuss any potential drawbacks or performance considerations.

**Questions for this assignment**

1. Create a new stored procedure named sp\_OrderProcessing

Insert your answer here:

2. Write a brief summary explaining the purpose of using cursors in this scenario and discuss any potential drawbacks or performance considerations.

Insert your answer here:

**How did you do?**

Compare the instructor's example to your own

**Instructor example**

A person with his arms crossed

Description automatically generated

[Trevoir Williams](https://www.udemy.com/user/trevoirwilliams/)

Create a new stored procedure named sp\_OrderProcessing

CREATE PROCEDURE sp\_OrderProcessing

AS

BEGIN

-- Declare variables

DECLARE @OrderId INT;

DECLARE @SubTotal MONEY;

DECLARE @TaxAmt MONEY;

DECLARE @TotalDue MONEY;

-- Declare and open the cursor

DECLARE OrderCursor CURSOR FOR

SELECT SalesOrderID, SubTotal, TaxAmt

FROM Sales.SalesOrderHeader

ORDER BY OrderDate ASC;

OPEN OrderCursor;

-- Fetch and process the cursor rows

FETCH NEXT FROM OrderCursor INTO @OrderId, @SubTotal, @TaxAmt;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- Calculate the total order amount

SET @TotalDue = @SubTotal + @TaxAmt;

-- Update the CalculatedTotalDue column in the SalesOrderHeader table

UPDATE Sales.SalesOrderHeader

SET CalculatedTotalDue = @TotalDue

WHERE SalesOrderID = @OrderId;

FETCH NEXT FROM OrderCursor INTO @OrderId, @SubTotal, @TaxAmt;

END

-- Close and deallocate the cursor

CLOSE OrderCursor;

DEALLOCATE OrderCursor;

END

Write a brief summary explaining the purpose of using cursors in this scenario and discuss any potential drawbacks or performance considerations.

* Cursors are useful when you need to perform row-by-row operations on a result set. In this scenario, the **sp\_OrderProcessing** stored procedure uses a cursor to iterate over the sales orders in the **SalesOrderHeader** table, calculates the total order amount for each order, and updates the **TotalDue** column accordingly.
* However, it's important to note that cursors can have drawbacks in terms of performance, especially when dealing with large result sets. Cursors operate row by row, which can be slower compared to set-based operations. It's essential to consider alternative approaches and evaluate the potential impact on performance before opting for cursors.
* In many cases, it's recommended to explore set-based operations or other SQL constructs like JOINs, subqueries, or CTEs to achieve the desired results more efficiently