## Instructions:

* The assignment aims to reinforce SQL query skills using the **WideWorldImporters** database. You will write and execute SQL queries to retrieve and manipulate data using various SQL constructs covered in Chapter 14 of our SQL text.
* Write SQL queries that fulfill the requirements listed below.
* Use appropriate column names, functions, and sorting techniques where necessary.
* Ensure queries return meaningful results based on the dataset.
* Save your work as a .sql file and upload it to D2L Dropbox folder for the assignment. Use the following file naming convention: *yourFirstName\_yourLastName\_SQL\_HW\_09.sql*
* Include DROP VIEW IF EXISTS statements before each CREATE VIEW to avoid errors on re-runs.
* Do not modify any original records in the database.
* For update/insert/delete exercises, use only test data that you insert yourself.

1. Declare and Use Scalar Variables

Objective: Demonstrate the use of scalar variables in a script.

Description: Declare a scalar variable named @AvgUnitPrice and assign it the average UnitPrice from the Sales.InvoiceLines table. Display the value of the variable using a SELECT statement. This exercise helps you understand scalar variable declaration, assignment, and output.

2. Use Table Variables

Objective: Use table variables to store and manipulate data.

Description: Define a table variable to capture the top 5 customers based on the sum of ExtendedPrice. Join the Sales.InvoiceLines table with Sales.Invoices to get the CustomerID. Group by CustomerID and use an ORDER BY clause with OFFSET FETCH to select only the top 5. Then display the results. This will help you learn how to work with table variables and aggregate joins.

3. Use Temporary Tables

Objective: Work with temporary tables and populate data from an existing table.

Description: Create a temporary table named #HeavyStockItems that includes StockItemID, StockItemName, and TypicalWeightPerUnit for all stock items where the typical weight per unit is greater than 500 from the Warehouse.StockItems table. Display the contents and drop the table afterward. This reinforces how to use SELECT INTO for temp table creation and cleanup.

4. Conditional Script Execution

Objective: Demonstrate conditional logic using IF...ELSE statements.

Description: Count how many invoices exist in the Sales.Invoices table. If the count is greater than 1000, print a message stating that. Otherwise, print a message indicating fewer invoices. This teaches you to use IF...ELSE logic with scalar variables and PRINT statements.

5. Existence Check for Object

Objective: Use IF EXISTS to test for an object before creating or dropping it.

Description: Check for the existence of a temporary table named #TestTable using OBJECT\_ID. If it exists, drop it. Then create a new temporary table with columns ID (int) and Name (nvarchar). This is useful for ensuring repeatable scripts.

6. Repetitive Processing with WHILE

Objective: Demonstrate the use of WHILE loop for repetition.

Description: Use a WHILE loop to print the numbers 1 to 5. Store a counter variable and increment it in each loop iteration. Use PRINT statements to output each number. This demonstrates repetitive logic and basic loop constructs.

7. Cursor Usage

Objective: Use a cursor to iterate through a result set.

Description: Declare and use a cursor to fetch and print customer names one-by-one from the Sales.Customers table. This exercise demonstrates cursor declaration, fetching, iteration, and cleanup using CLOSE and DEALLOCATE.

8. Error Handling

Objective: Use TRY...CATCH to handle errors in scripts.

Description: Use a TRY...CATCH block to attempt a division by zero operation. If an error occurs, catch it and print an informative error message. This reinforces error detection and graceful handling of unexpected issues.

9. Session Settings

Objective: Use SET statements to configure session options.

Description: Use SET NOCOUNT ON to prevent the row count message from being returned during insert operations. Create a temporary table, insert a value, then display the contents. Drop the table and turn SET NOCOUNT OFF afterward.

10. Dynamic SQL

Objective: Demonstrate use of dynamic SQL for flexibility.

Description: Declare a variable @TableName with a value of 'Sales.Invoices'. Build a SQL string to select the top 5 rows from that table and execute it using sp\_executesql. This shows how to use dynamic SQL for flexible scripting.

**Grading Rubric**

Each query is worth \*\*10 points\*\*, evaluated based on the following criteria:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criteria** | **Excellent (10 - 9 pts)** | **Good (8 - 7 pts)** | **Satisfactory (6 - 5 pts)** | **Needs Improvement (4 - 0 pts)** |
| **Correctness of Query (50%)** | Query executes without errors and returns the expected result set. | Query executes with minor logical errors but mostly correct. | Query contains errors affecting correctness. | Query does not execute properly or is incorrect. |
| **Use of Required SQL Concepts (30%)** | Successfully applies the required SQL concepts. | Uses most required SQL concepts correctly. | Uses some SQL concepts but omits key elements. | SQL concepts are misused or omitted. |
| **Query Formatting & Readability (10%)** | Query is well-structured and properly formatted. | Query is readable but could use better formatting. | Query is difficult to read with inconsistent formatting. | Query lacks readability and proper structure. |
| **Use of Proper Column Naming and Aliasing (10%)** | All columns are named meaningfully. | Most columns are appropriately named. | Some columns are meaningfully named. | Column naming and aliasing are unclear or missing. |