Querying with Transact-SQL

Lab 10 – Programming with Transact-SQL

Overview

In this lab, you will use some basic Transact-SQL programming logic to work with data in the **AdventureWorksLT** database.

What You'll Need

- An SQL Server Database instance with the **AdventureWorksLT** sample database.

Challenge 1: Creating scripts to insert sales orders

You want to create reusable scripts that make it easy to insert sales orders. You plan to create a script to insert the order header record, and a separate script to insert order detail records for a specified order header. Both scripts will make use of variables to make them easy to reuse.

Tip: Review the documentation for variables and the IF...ELSE block in the Transact-SQL Language Reference.

1. Write code to insert an order header

Your script to insert an order header must enable users to specify values for the order date, due date, and customer ID. The script should insert a record into the **SalesLT.SalesOrderHeader** table using these values and a hard-coded value of 'CARGO TRANSPORT 5' for the shipping method with default or NULL values for all other columns.

After the script has inserted the record, it should display the inserted **SalesOrderID** using the PRINT command.

Test your code with the following values:

Order Date	Due Date	Customer ID
Today's date	7 days from now	1

2. Write code to insert an order detail

The script to insert an order detail must enable users to specify a sales order ID, a product ID, a quantity, and a unit price. It must then check to see if the specified sales order ID exists in the **SalesLT.SalesOrderHeader** table. If it does, the code should insert the order details into the **SalesLT.SalesOrderDetail** table (using default values or NULL for unspecified columns). If the sales order ID does not exist in the **SalesLT.SalesOrderHeader** table, the code should print the message 'The order does not exist'.

Test your code with the following values:

Sales Order ID	Product ID	Quantity	Unit Price
The sales order ID returned by your previous code to insert a sales order header.	760	1	782.99

Then test it again with the following values:

Sales Order ID	Product ID	Quantity	Unit Price
0	760	1	782.99

Challenge 2: Updating Bike Prices

Adventure Works has determined that the market average price for a bike is $2,000, and consumer research has indicated that the maximum price any customer would be likely to pay for a bike is $5,000. You must write some Transact-SQL logic that incrementally increases the list price for all bike products by 10% until the average list price for a bike is at least the same as the market average, or until the most expensive bike is priced above the acceptable maximum indicated by the consumer research.

Tip: Review the documentation for WHILE in the Transact-SQL Language Reference.

1. Write a WHILE loop to update bike prices

The loop should:

- Execute only if the average list price of a product in the 'Bikes' parent category is less than the market average. Note that the product categories in the Bikes parent category can be determined from the **SalesLT.vGetAllCategories** view.
- Update all products that are in the 'Bikes' parent category, increasing the list price by 10%.
- Determine the new average and maximum selling price for products that are in the 'Bikes' parent category.

- If the new maximum price is greater than or equal to the maximum acceptable price, exit the loop; otherwise continue.