МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский технологический университет «МИСиС»

КАФЕДРА	Автоматизированных систем управления
НАПРАВЛЕНИЕ	09.03.01 «Информатика и вычислительная техника»
ПРОФИЛЬ	Интеллектуальные системы обработки информации и управления

ЛАБОРАТОРНАЯ РАБОТА

по дисциплине

Базы данных

на тему: «Программирование на Transact-SQL»

Студент(ка) ак. группы БИВТ-18-1 _____ Гузев В. Н.
аббревиатура подпись И.О. Фамилия

Оценка с учетом защиты _____ _____
оценка дата

Преподаватель _____ _____
подпись И.О. Фамилия

Москва 2021

Обзор

На этом семинаре Вы будете использовать базовые конструкции программирования на языке Transact-SQL для работы с данными в базе данных AdventureWorksLT.

Задача 1: Создание скриптов для добавления заказов

Вам необходимо написать скрипты, которые упрощают добавление информации по новым заказам в базу данных. Вы планируете создать скрипт для добавления записи заголовка заказа и отдельный скрипт для добавления записей товаров заказа для указанного заголовка заказа. Оба скрипта должны использовать переменные, чтобы облегчить их повторное использование.

1. **Напишите код для добавления заголовка заказа**

   Ваш скрипт для добавления заголовка заказа должен позволять пользователям указывать значения для даты заказа (столбец ***OrderDate****), срока платежа (столбец* ***DueDate****) и идентификатора клиента (столбец* ***CustomerID****). Идентификатор* ***SalesOrderID*** *должен быть сгенерирован автоматически – необходимо использовать следующее значение для последовательности* ***SalesLT.SalesOrderNumber*** *и присваиваться переменной. Затем скрипт должен добавить запись в таблицу* ***SalesLT.SalesOrderHeader*** *с использованием этих значений и жестко запрограммированного значения «CARGO TRANSPORT 5» для способа доставки и со значениями по умолчанию или NULL для всех остальных столбцов.*

   После того, как скрипт добавил запись, он должен отобразить использованное значение SalesOrderID с помощью команды PRINT.

   Проверьте свой код со следующими значениями:

Order Date	***Due Date***	***Customer ID***
< сегодня >	*< сегодня + 7 дней >*	*1*

Команда:

```
DECLARE @OrderDate datetime = GETDATE();
DECLARE @DueDate datetime = DATEADD(dd, 7, GETDATE());
DECLARE @CustomerID int = 1;
DECLARE @OrderID int;
SET @OrderID = NEXT VALUE FOR SalesLT.SalesOrderNumber;
INSERT INTO SalesLT.SalesOrderHeader (SalesOrderID, OrderDate, DueDate, CustomerID,
ShipMethod)
VALUES
(@OrderID, @OrderDate, @DueDate, @CustomerID, 'CARGO TRANSPORT 5');
PRINT @OrderID;
SELECT * FROM SalesLT.SalesOrderHeader WHERE SalesOrderID=@ID
```

2. **Напишите код, добавляющий товар к заказу**

   Скрипт для добавления товара в заказ должен позволять указывать идентификатор заказа, идентификатор товара, проданное количество товара и цену за единицу товара. Затем скрипт должен проверить, существует ли указанный идентификатор заказа в таблице ***SalesLT.SalesOrderHeader****. Если существует, то скрипт должен добавить данные по товару в таблицу* ***SalesLT.SalesOrderDetail*** *(используя значения по умолчанию или NULL для неуказанных столбцов). Если идентификатор заказа не существует в таблице* ***SalesLT.SalesOrderHeader****, то скрипт должен выводить сообщение «Заказ не существует». Вы можете*

проверить наличие записи в таблице с использованием предиката EXISTS. Проверьте свой код со следующими значениями:

Sales Order ID	Product ID	Quantity	Unit Price
<SalesOrderID полученный в предыдущей задаче по добавлению заголовка заказа >	760	1	782,99

Затем проверьте свой код вновь со следующими значениями:

Sales Order ID	Product ID	Quantity	Unit Price
0	760	1	782,99

Команда:

```
DECLARE @SalesOrderID int
DECLARE @ProductID int = 760;
DECLARE @Quantity int = 1;
DECLARE @UnitPrice money = 782.99;
SET @SalesOrderID = 0;
IF EXISTS (SELECT * FROM SalesLT.SalesOrderHeader WHERE SalesOrderID =
@SalesOrderID)
BEGIN
  INSERT INTO SalesLT.SalesOrderDetail (SalesOrderID, OrderQty, ProductID, UnitPrice)
  VALUES
  (@SalesOrderID, @Quantity, @ProductID, @UnitPrice)
END
ELSE
BEGIN
```

```
  PRINT N'Заказ не существует'
END
```

Задача 2: Обновление цен на велосипеды (категория «Bikes»)

В компании Adventure Works определили, что средняя рыночная цена на велосипед составляет 2000 долларов США, а потребительское исследование показало, что максимальная цена, которую клиент может заплатить за велосипед, составляет 5000 долларов США. Вам необходимо написать скрипт на Transact-SQL, который постепенно увеличивает цену (в столбце ListPrice) для всех велосипедов на 10%, пока средняя цена велосипеда не будет по крайней мере такой же, как средняя по рынку, или пока самый дорогой велосипед не будет стоить выше приемлемой максимальной цены, указанной в потребительском исследовании.

1. **Напишите цикл WHILE, чтобы обновить цены на велосипеды**

   Цикл должен:

   - *Выполняться только в том случае, если средняя цена (в столбце ListPrice) товаров в родительской категории «Bikes» меньше средней по рынку. Обратите внимание, что категории товаров в родительской категории «Bikes» можно определить из представления SalesLT.vGetAllCategories.*
   - *Обновить все товары, находящиеся в родительской категории «Bikes», увеличив цену (в столбце ListPrice) на 10%.*
   - *Определить новую среднюю и максимальную отпускную цену для товаров, находящихся в родительской категории «Bikes».*
   - *Если новая максимальная цена больше или равна максимально допустимой цене (5000 долларов США), выйти из цикла, а в противном случае – продолжить выполнение.*

После завершения выполнения цикла выведите на экран методом PRINT среднюю цену продаваемых велосипедов («Новая средняя цена на велосипед») и максимальную цену велосипеда («Новая максимальная цена велосипеда»).

Команда:

```
BEGIN TRAN

DECLARE @MaxPrice money = 5000
DECLARE @AvgPrice money = 2000

WHILE
(@AvgPrice > (SELECT AVG(ListPrice) FROM SalesLT.Product
WHERE ProductCategoryID IN (SELECT ProductCategoryID
FROM SalesLT.vGetAllCategories WHERE ParentProductCategoryName = 'Bikes')))
AND
(@MaxPrice > (SELECT MAX(ListPrice) FROM SalesLT.Product WHERE ProductCategoryID
IN (SELECT ProductCategoryID
FROM SalesLT.vGetAllCategories WHERE ParentProductCategoryName = 'Bikes')))

BEGIN
UPDATE SalesLT.Product SET ListPrice = 1.1*ListPrice WHERE ProductCategoryID IN
(SELECT ProductCategoryID
FROM SalesLT.vGetAllCategories WHERE ParentProductCategoryName = 'Bikes')
END

DECLARE @a money = (SELECT MAX(ListPrice) FROM SalesLT.Product WHERE
ProductCategoryID IN (SELECT ProductCategoryID
FROM SalesLT.vGetAllCategories WHERE ParentProductCategoryName = 'Bikes'))
DECLARE @b money = (SELECT AVG(ListPrice) FROM SalesLT.Product AS p
WHERE
(SELECT ParentProductCategoryName
```

```
FROM SalesLT.vGetAllCategories AS v WHERE v.ProductCategoryID =
p.ProductCategoryID) = 'Bikes')

print @a
print @b

ROLLBACK TRAN
```

Вывод

В ходе выполнения данной работы мы познакомились с базовыми конструкциями программирования на языке Transact-SQL для работы с данными в базе данных.