Problem Set: Modifying Data

There is a lot to learn when querying data, but you may also need to modify data as well. This is especially true if you develop T-SQL code for an application, not just for reports. To avoid messing up your copy of AdventureWorks, this section starts with creating temporary tables. Temporary tables are stored in a special system database called Tempdb. They last as long as the connection that created them is open or until they are purposefully dropped.

There are three things you can do to manipulate data: insert new rows, modify existing rows, or delete existing rows.

Exercise 1

1.1 Create a copy of the Sales.SalesOrderHeader table into a temp table called #Sales and populate it with data in one step. Include only the SalesOrderID, OrderDate, TotalDue, and CustomerID columns.

1.2 Create a temp table called #Customers. It must have a CustomerID. It must also have a FirstName and LastName column. Figure out which data types are needed by looking at the Person.Person and Sales.Customer tables. Make the CustomerID a primary key.

Exercise 2

2.1 Write a query to populate the #Customers table. You will need to join the Person.Person and Sales.Customer tables. The PersonID will join to the BusinessEntityID. Be sure to run the query to populate the #Customers table.

2.2 Add three new customers to the #Customers table. They can be any names you wish, but make sure that you do not duplicate the CustomerID column.

2.3 Add an order for each of the new customers you added in 2.2. Make sure you do not duplicate existing SalesOrderID values. You can choose any date and total due you wish.

Exercise 3

3.1 Modify the #Sales rows so that every order below $10 is changed to $10.

3.2 You can also use existing columns to modify data. Write a query that adds one day to every OrderDate in the #Sales table. Use the DATEADD function.

3.3 Delete any rows where the order was placed in 2011.

Modifying Data: Solution

Exercise 1

1.3 Create a copy of the Sales.SalesOrderHeader table into a temp table called #Sales and populate it with data in one step. Include only the SalesOrderID, OrderDate, TotalDue, and CustomerID columns.

```
SELECT SalesOrderID, OrderDate, TotalDue, CustomerID  INTO #Sales  FROM
Sales.SalesOrderHeader;
```

1.4 Create a temp table called #Customers. It must have a CustomerID. It must also have a FirstName and LastName column. Figure out which data types are needed by looking at the Person.Person and Sales.Customer tables. Make the CustomerID a primary key.

```
CREATE TABLE #Customers (
        CustomerID INT PRIMARY KEY,
        FirstName NVARCHAR(50),
        LastName NVARCHAR(50)
);
```

Exercise 2

2.1 Write a query to populate the #Customers table. You will need to join the Person.Person and Sales.Customer tables. The PersonID will join to the BusinessEntityID. Be sure to run the query to populate the #Customers table.

```
INSERT INTO #Customers (CustomerID, FirstName, LastName)
SELECT C.CustomerID, P.FirstName, P.LastName
FROM Sales.Customer AS C  JOIN Person.Person AS P ON P.BusinessEntityID = C.PersonID;
```

2.2 Add three new customers to the #Customers table. They can be any names you wish, but make sure that you do not duplicate the CustomerID column.

```
SELECT MAX(CustomerID) AS MaxCustomerID FROM #Customers;
--Answers will vary!
INSERT INTO #Customers ( CustomerID, FirstName, LastName )
VALUES (30119, 'Kathi','Morgan'),
        (30120,'Charlie','Brown'),
        (30121, 'Snoopy','Brown');
```

2.3 Add an order for each of the new customers you added in 2.2. Make sure you do not duplicate existing SalesOrderID values. You can choose any date and total due you wish.

```
SELECT MAX(SalesOrderID) AS MaxSalesOrderID  FROM #Sales;
 --Answers will vary
INSERT INTO #Sales( SalesOrderID, OrderDate, TotalDue, CustomerID)
VALUES  (75124,'2015-08-31',100,30119),
            (75125,'2015-09-01',200,30120),
            (75126,'2015-09-02',300,30121);
```

Exercise 3

3.1 Modify the #Sales rows so that every order below $10 is changed to $10.

```
UPDATE #Sales
SET TotalDue = 10
WHERE TotalDue < 10;
```

3.2 You can also use existing columns to modify data. Write a query that adds one day to every OrderDate in the #Sales table. Use the DATEADD function.

```
UPDATE #Sales
SET OrderDate = DATEADD(DAY, 1, OrderDate);
```

3.3 Delete any rows where the order was placed in 2011.

```
DELETE FROM #Sales
WHERE OrderDate >= '2011-01-01' AND   OrderDate < '2012-01-01';
```