Assignment #6

Instructor: Richard Frank (rfrank@sfu.ca)
TA: Ankit Gupta (aga53@sfu.ca)

Total Marks: 50 (5% of the Individual Assignments)
Due Date: Oct 28, 14:30

For all questions, use the AdventureWorksLT database.

Question 1) Provide the SQL commands to accomplish the below. Note that each question could contain multiple queries.

a) Add a new customer John Doe who lives at 456 Main Street, Burnaby, BC, V1V2V3. For the password-hash and salt, just put 'password' and 'salt', respectively. [10 marks]

```
DECLARE @CustID INTEGER
INSERT INTO Customer
SELECT @CustID = CustomerID FROM Customer WHERE…
INSERT INTO Address(…) VALUES (…)

DECLARE @AddressID INTEGER
SELECT @AddressID = AddressID FROM Address WHERE …
INSERT INTO CustomerAddress(AddressID, CustomerID, …) VALUES (@AddressID,
@CustID, …)
```

```
As an alternative, you can also assume that the ID of the previous INSERT
is somehow handed in an application. In this case, the SQL will be:

INSERT INTO Customer(…) VALUES (…)
SELECT CustomerID FROM Customer WHERE…
**Save the CustomerID from your .NET program
INSERT INTO Address(…) VALUES (…)

SELECT AddressID FROM Address WHERE …
**Save the AddressID from your .NET program
INSERT INTO CustomerAddress(AddressID, CustomerID, …) VALUES (**AddressID
from program**, **CustomerID from program**, …)
```

```
As an alternative, you can do the following:

INSERT INTO Customer(…) VALUES (…)
INSERT INTO Address(…) VALUES (…)
INSERT INTO CustomerAddress(AddressID, CustomerID, …) VALUES (SELECT
AddressID FROM Address WHERE …, SELECT CustomerID FROM Customer WHERE …
, …)
```

b) Apply a $10 tax-rebate to each order. If a 10$ tax-rebate is not possible, apply the largest rebate you can. Reduce the total due by the corresponding amount. [5 marks]

```
UPDATE SalesLT.SalesOrderHeader SET TotalDue = TotalDue – TaxAmt, SubTotal
= SubTotal – TaxAmt, TaxAmt = 0 WHERE TaxAmt < 10
```

```
UPDATE SalesLT.SalesOrderHeader SET TotalDue = TotalDue – 10, SubTotal =
SubTotal – 10, TaxAmt = TaxAmt – 10
WHERE TaxAmt > 10
```

c) Count the number of addresses for each customer. [2 marks]

```
SELECT CustomerID, COUNT(*) From SalesOrderHeader GROUP BY CustomerID
```

d) Calculate the average, minimum and maximum prices of all products in the database. [3 marks]

```
SELECT AVG(ListPrice) as AVER, MIN(ListPrice) as MINI, MAX(ListPrice) as
MAXI
FROM Product
```

e) Display all product model names and their descriptions (from the ProductDescription table), display the product model name even if the description does not exist. [8 marks]

```
SELECT P.Name, PD.Description
FROM SalesLT.Product P LEFT JOIN SalesLT.ProductModel PM
      ON P.ProductModelID = PM.ProductModelID
      LEFT JOIN SalesLT.ProductModelProductDescription PMPD
            ON PM.ProductModelID = PMPD.ProductModelID
            LEFT JOIN SalesLT.ProductDescription PD
                  ON PMPD.ProductDescriptionID = PD.ProductDescriptionID
```

f) Count the number of NULL descriptions from the above question e). [2 marks]

```
SELECT COUNT(*) FROM (
      SELECT P.Name, PD.Description
      FROM SalesLT.Product P LEFT JOIN SalesLT.ProductModel PM
            ON P.ProductModelID = PM.ProductModelID
            LEFT JOIN SalesLT.ProductModelProductDescription PMPD
                  ON PM.ProductModelID = PMPD.ProductModelID
                  LEFT JOIN SalesLT.ProductDescription PD
                        ON PMPD.ProductDescriptionID =
PD.ProductDescriptionID ) A
WHERE Description IS NULL
```

Question 2)

a) Consider Question 1a). Create a database procedure out of it. [10 marks]

```
CREATE PROCEDURE AddCustomer
   @FirstName VARCHAR(50),
   @LastName VARCHAR(50),
   @Address VARCHAR(60),
   @City VARCHAR(30),
   @StateProvince VARCHAR(30),
   @PostalCode VARCHAR(15),
   @Country VARCHAR(50),
   @OfficeType VARCHAR(50)
AS
   --Insert the customer record first
   INSERT INTO SalesLT.Customer(NameStyle, FirstName, LastName, PasswordHash,
PasswordSalt)
         VALUES (0,@FirstName, @LastName, 'password', 'salt');
```

```
    DECLARE @CustID INTEGER
    --Get the ID that was just inserted
    SELECT @CustID = CustomerID
    FROM SalesLT.Customer
    WHERE FirstName = @FirstName AND LastName = @LastName;
    --Can also use "SELECT @CustID = @@SCOPE_IDENTITY"
    --@@IDENTITY returns the last-inserted identity value within the
    --current session.

    --Insert the Address information
    INSERT INTO SalesLT.Address(AddressLine1,City,PostalCode, StateProvince,
CountryRegion)
          VALUES(@Address, @City, @PostalCode, @StateProvince, @Country);

    DECLARE @AddID INTEGER
    --Get the ID that was just inserted
    SELECT @AddID = AddressID
    FROM SalesLT.Address
    WHERE AddressLine1 = @Address AND City = @City
          AND PostalCode = @PostalCode AND CountryRegion = @Country;
    --Can also use "SELECT @AddID = @@ SCOPE_IDENTITY"

    --Insert the entry into the join table
    INSERT INTO SalesLT.CustomerAddress(CustomerID,AddressID,AddressType)
          VALUES(@CustID, @AddID, @OfficeType)

GO
```

b) Write a sample command to run the procedure. [2 marks]

```
EXECUTE AddCustomer 'Richar', 'Frank', '29 Mai', NULL, 'V1V1V1', 'BC',
'Canada', 'Main Office'
```

c) What happens if the database crashes in the middle of executing the procedure? What is rolled back? [3 marks]

Nothing is rolled back. All executed commands are permanent. The effect of this is that there will be tuples in the database that do not have corresponding tuples in other tables, which were supposed to have been created during the latter commands.

d) Modify the procedure so the entire thing is a transaction on its own. [7 marks]

```
CREATE PROCEDURE AddCustomer
      @FirstName VARCHAR(50),
      @LastName VARCHAR(50),
      @Address VARCHAR(60),
      @City VARCHAR(30),
      @StateProvince VARCHAR(30),
      @PostalCode VARCHAR(15),
      @Country VARCHAR(50),
      @OfficeType VARCHAR(50)
AS

      BEGIN TRANSACTION

      --Insert the customer record first
```

```
	INSERT INTO SalesLT.Customer(NameStyle, FirstName, LastName,
PasswordHash, PasswordSalt)
		VALUES (0,@FirstName, @LastName, 'password', 'salt');

	IF @@ERROR <> 0 ROLLBACK

	DECLARE @CustID INTEGER
	--Get the ID that was just inserted
	SELECT @CustID = CustomerID
	FROM SalesLT.Customer
	WHERE FirstName = @FirstName AND LastName = @LastName;

	--Insert the Address information
	INSERT INTO SalesLT.Address(AddressLine1,City,PostalCode,
StateProvince, CountryRegion)
		VALUES(@Address, @City, @PostalCode, @StateProvince, @Country);

	IF @@ERROR <> 0 ROLLBACK

	DECLARE @AddID INTEGER
	--Get the ID that was just inserted
	SELECT @AddID = AddressID
	FROM SalesLT.Address
	WHERE AddressLine1 = @Address AND City = @City
		AND PostalCode = @PostalCode AND CountryRegion = @Country;

	--Insert the entry into the join table
	INSERT INTO SalesLT.CustomerAddress(CustomerID,AddressID,AddressType)
		VALUES(@CustID, @AddID, @OfficeType)

	IF @@ERROR <> 0 ROLLBACK

	COMMIT
GO
```