MSIN 616 Advanced Database Management Fall 2017

Solutions Assignment 10

III. Exercises 1 – 4 of Murach p. 447

1. Write a script that declares and sets a variable that is equal to the total outstanding balance due (InvoiceTotal – PaymentTotal – CreditTotal). If that balance due is greater than $10,000.0 the script should return a result set consisting of VendorName, InvoiceNumber, InvoiceDueDate and Balance for each invoice with balance due, sorted with the oldest due date first. If the total outstanding balance due is less than $10,000.00, the script should return the message “Balance due is less than $10,000.00.” How do you test the script to verify that both parts are working?

```
DECLARE @TotalInvoiceDue money;

SELECT @TotalInvoiceDue =
       SUM(InvoiceTotal - CreditTotal - PaymentTotal)
FROM Invoices
WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0;

IF @TotalInvoiceDue > 10000
    SELECT VendorName
           ,InvoiceNumber
            ,InvoiceDueDate
            ,Balance = InvoiceTotal - CreditTotal - PaymentTotal
    FROM  Invoices inv
      JOIN  Vendors  ven
       ON  ven.VendorID = inv.VendorID
    WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0
    ORDER BY InvoiceDueDate;
ELSE
  PRINT 'Balance due is less than $10,000.00.';
```

VendorName	InvoiceNumber	InvoiceDueDate	Balance
Data Reproductions Corp	39104	04/09/16	85.31
Ingram	31361833	04/10/16	579.42
Federal Express Corporation	963253264	04/17/16	52.25
Cardinal Business Media, Inc.	134116	04/17/16	90.36
Federal Express Corporation	263253268	04/20/16	59.97
Federal Express Corporation	263253270	04/21/16	67.92
Federal Express Corporation	263253273	04/21/16	30.75
Ford Motor Credit Company	9982771	04/23/16	503.20
Malloy Lithographing Inc	0-2436	04/30/16	10,976.06
Blue Cross	547480102	04/30/16	224.00

2. The following script uses a derived table to return the date and invoice total of the earliest invoice issued by each vendor. Write a script that generates the same result set but uses a temporary table, #FirstInvoice, in place of the derived table. Make sure your script tests for the existence of #FirstInvoice: **`SELECT OBJECT_ID('tempdb..#FirstInvoice')`**

```
USE AP;

SELECT VendorName
      ,FirstInvoiceDate
        ,InvoiceTotal
FROM Invoices inv
JOIN (SELECT VendorID,
             MIN(InvoiceDate) AS FirstInvoiceDate
      FROM Invoices
      GROUP BY VendorID) AS FrstInv
  ON FrstInv.VendorID = inv.VendorID
 AND FrstInv.FirstInvoiceDate = inv.InvoiceDate
JOIN Vendors ven
  ON inv.VendorID = ven.VendorID
ORdER BY VendorName, FirstInvoiceDate;
```

```
USE AP;

IF OBJECT_ID('tempdb..#FirstInvoice') IS NOT NULL
    DROP TABLE #FirstInvoice;

SELECT VendorID,
       FirstInvoiceDate = MIN(InvoiceDate)
INTO #FirstInvoice
FROM Invoices
GROUP BY VendorID;

SELECT VendorName
      ,FirstInvoiceDate
      ,InvoiceTotal
FROM Invoices inv
JOIN #FirstInvoice tmp
  ON  tmp.VendorID    = inv.VendorID AND
      inv.InvoiceDate = tmp.FirstInvoiceDate
JOIN Vendors ven
  ON ven.VendorID = inv.VendorID
ORDER BY VendorName, FirstInvoiceDate;
```

3. Write a script that generates the same result set as the code shown in exercise 2, but uses a view instead of a derived table. Also write the script that creates the vies. Make sure that your script test for the existence of the view. The view does not need to be redefined each time the script is executed.

```
USE AP;

IF OBJECT_ID('vFirstInvoice') IS NOT NULL
    DROP VIEW vFirstInvoice;
GO

CREATE VIEW vFirstInvoice
AS
SELECT VendorID
      ,FirstInvoiceDate = MIN(InvoiceDate)
FROM Invoices
GROUP BY VendorID;
GO

SELECT VendorName
      ,FirstInvoiceDate
   ,InvoiceTotal
FROM  Invoices inv
JOIN  vFirstInvoice vie
  ON  inv.VendorID = vie.VendorID AND
      inv.InvoiceDate = vie.FirstInvoiceDate
JOIN Vendors ven
  ON ven.VendorID = inv.VendorID
ORDER BY VendorName, FirstInvoiceDate;
```

4. Write a script that uses dynamic SQL to return a single column that represents the number of rows in the first table in the current database. The script should automatically choose the table that appears first alphabetically, and it should exclude tables named dtproperteds and sysdiagrams. Name the column CountOf*Table* Where *Table* is the chosen table name. Hint: Us the sys.tables catalog view.

```
DECLARE @TableName varchar(128);
DECLARE @Query     varchar(256);

SELECT @TableName = MIN(name)
FROM   sys.tables
WHERE  name <> 'dtproperties' AND name <> 'sysdiagrams';

SET @Query = CONCAT('SELECT COUNT(*) AS CountOf', @TableName,
                    ' FROM ', @TableName)

EXEC(@Query);
```

IV. Exercise 5 Murach p. 499

5.Create a scalar-valued function fnUnpaidInvoiceID that returns the InvoiceID of the earliest invoice with unpaid balance. Test the function in the following **SELECT** statement

```
SELECT VendorName
      ,InvoiceNumber
        ,InvoiceDueDate
        ,Balance = InvoiceTotal = PaymentTotal - CreditTotal
FROM   Vendors ven
JOIN   Invoices inv
  ON ven.VendorID = inv.VendorID
WHERE InvoiceID = dbo.fnUnpaidInvoiceID();
```

```
USE AP;
GO

CREATE FUNCTION fnUnpaidInvoiceID()
RETURNS int
BEGIN
    RETURN
    (SELECT MIN(InvoiceID)
     FROM Invoices
     WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0
       AND InvoiceDueDate = (SELECT MIN(InvoiceDueDate)
                             FROM Invoices
                             WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0));
END;
```