CHAPTER 1 – Foundations of T-SQL

***Listing 1-1.*** C# Snippet to Read a Flat File

StreamReader sr = new StreamReader("c:\\Person\_Person.txt");

string FirstName = null;

while ((FirstName = sr.ReadLine()) != null) {

Console.WriteLine(s); } sr.Dispose();

***Listing 1-2.*** SQL Query to Retrieve Names from a Table

SELECT FirstName FROM Person.Person;

***Listing 1-3.*** The HumanResources.vEmployee View from the AdventureWorks2012 Database

SELECT e.BusinessEntityID, p.Title, p.FirstName, p.MiddleName, p.LastName, p.Suffix, e.JobTitle, pp.PhoneNumber, pnt.Name AS PhoneNumberType, ea.EmailAddress,

p.EmailPromotion, a.AddressLine1, a.AddressLine2, a.City, sp.Name AS StateProvinceName, a.PostalCode, cr.Name AS CountryRegionName, p.AdditionalContactInfo

FROM HumanResources.Employee AS e INNER JOIN Person.Person AS p ON p.BusinessEntityID = e.BusinessEntityID INNER JOIN Person.BusinessEntityAddress AS bea ON bea.BusinessEntityID = e.BusinessEntityID INNER JOIN Person.Address AS a ON a.AddressID = bea.AddressID INNER JOIN Person.StateProvince AS sp ON sp.StateProvinceID = a.StateProvinceID INNER JOIN Person.CountryRegion AS cr ON cr.CountryRegionCode = sp.CountryRegionCode LEFT OUTER JOIN Person.PersonPhone AS pp ON pp.BusinessEntityID = p.BusinessEntityID LEFT OUTER JOIN Person.PhoneNumberType AS pnt ON pp.PhoneNumberTypeID = pnt.PhoneNumberTypeID LEFT OUTER JOIN Person.EmailAddress AS ea ON p.BusinessEntityID = ea.BusinessEntityID

***Listing 1-4.*** The HumanResources.vEmployee View Reformatted for Readability

SELECT

e.BusinessEntityID,

p.Title,

p.FirstName,

p.MiddleName,

p.LastName,

p.Suffix,

e.JobTitle,

pp.PhoneNumber,

pnt.Name AS PhoneNumberType,

ea.EmailAddress,

p.EmailPromotion,

a.AddressLine1,

a.AddressLine2,

a.City,

sp.Name AS StateProvinceName,

a.PostalCode,

cr.Name AS CountryRegionName,

p.AdditionalContactInfo

FROM HumanResources.Employee AS e INNER JOIN Person.Person AS p

ON p.BusinessEntityID = e.BusinessEntityID

INNER JOIN Person.BusinessEntityAddress AS bea

ON bea.BusinessEntityID = e.BusinessEntityID

INNER JOIN Person.Address AS a

ON a.AddressID = bea.AddressID

INNER JOIN Person.StateProvince AS sp

ON sp.StateProvinceID = a.StateProvinceID

INNER JOIN Person.CountryRegion AS cr

ON cr.CountryRegionCode = sp.CountryRegionCode

LEFT OUTER JOIN Person.PersonPhone AS pp

ON pp.BusinessEntityID = p.BusinessEntityID

LEFT OUTER JOIN Person.PhoneNumberType AS pnt

ON pp.PhoneNumberTypeID = pnt.PhoneNumberTypeID

LEFT OUTER JOIN Person.EmailAddress AS ea

ON p.BusinessEntityID = ea.BusinessEntityID;

***Listing 1-5.*** All-Capital SELECT Query

SELECT P.BUSINESSENTITYID, P.FIRSTNAME, P.LASTNAME, S.SALESYTD

FROM PERSON.PERSON P INNER JOIN SALES.SALESPERSON SP

ON P.BUSINESSENTITYID = SP.BUSINESSENTITYID;

***Listing 1-6.*** Reformatted, Easy-on-the-Eyes Query

SELECT

p.BusinessEntityID,

p.FirstName,

p.LastName,

sp.SalesYTD

FROM Person.Person p INNER JOIN Sales.SalesPerson sp

ON p.BusinessEntityID = sp.BusinessEntityID;

***Listing 1-7.*** Stored Procedure Example with One Entry and Multiple Exits

CREATE PROCEDURE dbo.GetOrAdd\_ContactType

(

@Name NVARCHAR(50),

@ContactTypeID INT OUTPUT

)

AS

DECLARE @Err\_Code AS INT;

SELECT @Err\_Code = 0;

SELECT @ContactTypeID = ContactTypeID

FROM Person.ContactType

WHERE [Name] = @Name;

IF @ContactTypeID IS NOT NULL

RETURN; -- Exit 1: if the ContactType exists

INSERT

INTO Person.ContactType ([Name], ModifiedDate)

SELECT @Name, CURRENT\_TIMESTAMP;

SELECT @Err\_Code = 'error';

IF @Err\_Code <> 0

RETURN @Err\_Code; -- Exit 2: if there is an error on INSERT

SELECT @ContactTypeID = SCOPE\_IDENTITY();

RETURN @Err\_Code; -- Exit 3: after successful INSERT

GO

***Listing 1-8.*** Stored Procedure with One Entry and One Exit

CREATE PROCEDURE dbo.GetOrAdd\_ContactType

(

@Name NVARCHAR(50),

@ContactTypeID INT OUTPUT

)

AS

DECLARE @Err\_Code AS INT;

SELECT @Err\_Code = 0;

SELECT @ContactTypeID = ContactTypeID

FROM Person.ContactType

WHERE [Name] = @Name;

IF @ContactTypeID IS NULL

BEGIN

INSERT

INTO Person.ContactType ([Name], ModifiedDate)

SELECT @Name, CURRENT\_TIMESTAMP;

SELECT @Err\_Code = @@error;

IF @Err\_Code = 0 -- If there's an error, skip next

SELECT @ContactTypeID = SCOPE\_IDENTITY();

END

RETURN @Err\_Code; -- Single exit point

GO

***Listing 1-9.*** Sample Code Using an Uninitialized Variable

DECLARE @i INT; SELECT @i = @i + 5; SELECT @i;

***Listing 1-10.*** Sample Code Using an Initialized Variable

DECLARE @i INT = 0; -- Changed this statement to initialize @i to 0

SELECT @i = @i + 5;

SELECT @i;

CHAPTER 3 – Procedural Code and CASE Expressions

***Listing 3-1.*** Simple IF . . . ELSE Example

DECLARE @i int = NULL;

IF @i = 10

PRINT 'TRUE.';

ELSE IF NOT (@i = 10)

PRINT 'FALSE.';

ELSE

PRINT 'UNKNOWN.';

***Listing 3-2.*** IF . . . ELSE with Statement Blocks

DECLARE @direction NVARCHAR(20) = N'DESCENDING';

IF @direction = N'ASCENDING'

BEGIN

PRINT 'Start at the top!';

SELECT TOP (10)

LastName,

FirstName,

MiddleName

FROM Person.Person

ORDER BY LastName ASC;

END

ELSE IF @direction = N'DESCENDING'

BEGIN

PRINT 'Start at the bottom!';

SELECT TOP (10)

LastName,

FirstName,

MiddleName

FROM Person.Person

ORDER BY LastName DESC;

END

ELSE

PRINT '@direction was not recognized!';

***Listing 3-3.*** WHILE Statement Example

DECLARE @i int = 1;

WHILE @i < = 10

BEGIN

PRINT @i;

SET @i = @i + 1;

END

***Listing 3-4.*** WHILE . . . CONTINUE Example

DECLARE @i int = 1;

WHILE @i < = 10

BEGIN

PRINT @i;

SET @i = @i + 1;

CONTINUE; -- Force the WHILE loop to restart

PRINT 'The CONTINUE keyword ensures that this will never be printed.';

END

***Listing 3-5.*** WHILE . . . BREAK Example

DECLARE @i int = 1;

WHILE @i < = 10

BEGIN

PRINT @i;

SET @i = @i + 1; CHAPTER 3 ■ Proce dural Code and CASE Expres ions8

BREAK; -- Force the WHILE loop to terminate

PRINT 'The BREAK keyword ensures that this will never be printed.';

END

***Listing 3-6.*** Simple GOTO Example

PRINT 'Step 1 Begin.';

GOTO Step3\_Label;

PRINT 'Step 2 will not be printed.';

Step3\_Label:

PRINT 'Step 3 End.';

***Listing 3-7.*** WAITFOR Example

PRINT 'Step 1 complete. ';

GO

DECLARE @time\_to\_pass nvarchar(8);

SELECT @time\_to\_pass = N'00:00:03';

WAITFOR DELAY @time\_to\_pass;

PRINT 'Step 2 completed three seconds later. ';

***Listing 3-8.*** Counting West Coast Customers with a Simple CASE Expression

WITH EmployeesByRegion(Region)

AS

(

SELECT

CASE sp.StateProvinceCode

WHEN 'CA' THEN 'West Coast'

WHEN 'WA' THEN 'West Coast'

WHEN 'OR' THEN 'West Coast'

ELSE 'Elsewhere'

END

FROM HumanResources.Employee e

INNER JOIN Person.Person p

ON e.BusinessEntityID = p.BusinessEntityID

INNER JOIN Person.BusinessEntityAddress bea

ON bea.BusinessEntityID = e.BusinessEntityID

INNER JOIN Person.Address a

ON a.AddressID = bea.AddressID

INNER JOIN Person.StateProvince sp

ON sp.StateProvinceID = a.StateProvinceID

WHERE sp.CountryRegionCode = 'US'

)

SELECT COUNT(Region) AS NumOfEmployees, Region

FROM EmployeesByRegion

GROUP BY Region;

***Listing 3-9.*** Counting Employees by Region with a Searched CASE Expression

WITH EmployeesByRegion(Region)

AS

(

SELECT

CASE WHEN sp.StateProvinceCode IN ('CA', 'WA', 'OR') THEN 'West Coast'

WHEN sp.StateProvinceCode IN ('HI', 'AK') THEN 'Pacific'

WHEN sp.StateProvinceCode IN ('CT', 'MA', 'ME', 'NH', 'RI', 'VT')

THEN 'New England'

ELSE 'Elsewhere'

END

FROM HumanResources.Employee e

INNER JOIN Person.Person p

ON e.BusinessEntityID = p.BusinessEntityID

INNER JOIN Person.BusinessEntityAddress bea

ON bea.BusinessEntityID = e.BusinessEntityID

INNER JOIN Person.Address a

ON a.AddressID = bea.AddressID

INNER JOIN Person.StateProvince sp

ON sp.StateProvinceID = a.StateProvinceID

WHERE sp.CountryRegionCode = 'US'

)

SELECT COUNT(Region) AS NumOfCustomers, Region

FROM EmployeesByRegion

GROUP BY Region;

***Listing 3-10.*** CASE-Style Pivot Table

SELECT

t.CountryRegionCode,

SUM

(

CASE WHEN t.Name = 'Northwest' THEN 1

ELSE 0

END

) AS Northwest,

SUM

(

CASE WHEN t.Name = 'Northeast' THEN 1

ELSE 0

END

) AS Northeast,

SUM

(

CASE WHEN t.Name = 'Southwest' THEN 1

ELSE 0

END

) AS Southwest,

SUM

(

CASE WHEN t.Name = 'Southeast' THEN 1

ELSE 0

END

) AS Southeast,

SUM

(

CASE WHEN t.Name = 'Central' THEN 1

ELSE 0

END

) AS Central

FROM Sales.SalesOrderHeader soh

INNER JOIN Sales.SalesTerritory t

ON soh.TerritoryID = t.TerritoryID

WHERE t.CountryRegionCode = 'US'

GROUP BY t.CountryRegionCode;

***Listing 3-11.*** PIVOT Operator Pivot Table

SELECT

CountryRegionCode,

Northwest,

Northeast,

Southwest,

Southeast,

Central

FROM

(

SELECT

t.CountryRegionCode,

t.Name

FROM Sales.SalesOrderHeader soh

INNER JOIN Sales.SalesTerritory t

ON soh.TerritoryID = t.TerritoryID

WHERE t.CountryRegionCode = 'US'

) p

PIVOT

(

COUNT (Name)

FOR Name

IN

(

Northwest,

Northeast,

Southwest,

Southeast,

Central

)

) AS pvt;

***Listing 3-12.*** Dynamic Pivot Table Query

-- Declare variables

DECLARE @sql nvarchar(4000);

DECLARE @temp\_pivot table

(

TerritoryID int NOT NULL PRIMARY KEY,

CountryRegion nvarchar(20) NOT NULL,

CountryRegionCode nvarchar(3) NOT NULL

);

-- Get column names from source table rows

INSERT INTO @temp\_pivot

(

TerritoryID,

CountryRegion,

CountryRegionCode

)

SELECT

TerritoryID,

Name,

CountryRegionCode

FROM Sales.SalesTerritory

GROUP BY

TerritoryID,

Name,

CountryRegionCode;

-- Generate dynamic SQL query

SET @sql = N'SELECT' +

SUBSTRING(

(

SELECT N', SUM(CASE WHEN t.TerritoryID = ' +

CAST(TerritoryID AS NVARCHAR(3)) +

N' THEN 1 ELSE 0 END) AS ' + QUOTENAME(CountryRegion) AS "\*"

FROM @temp\_pivot

FOR XML PATH('')

), 2, 4000) +

N' FROM Sales.SalesOrderHeader soh ' +

N' INNER JOIN Sales.SalesTerritory t ' +

N' ON soh.TerritoryID = t.TerritoryID; ' ;

-- Print and execute dynamic SQL

PRINT @sql;

EXEC (@sql);

***Listing 3-13.*** Autogenerated Dynamic SQL Pivot Table Query

SELECT SUM

(

CASE WHEN t.TerritoryID = 1 THEN 1

ELSE 0

END

) AS [Northwest],

SUM

(

CASE WHEN t.TerritoryID = 2 THEN 1

ELSE 0

END

) AS [Northeast],

SUM

(

CASE WHEN t.TerritoryID = 3 THEN 1

ELSE 0

END

) AS [Central],

SUM

(

CASE WHEN t.TerritoryID = 4 THEN 1

ELSE 0

END

) AS [Southwest],

SUM

(

CASE WHEN t.TerritoryID = 5 THEN 1

ELSE 0

END

) AS [Southeast],

SUM

(

CASE WHEN t.TerritoryID = 6 THEN 1

ELSE 0

END

) AS [Canada],

SUM

(

CASE WHEN t.TerritoryID = 7 THEN 1

ELSE 0

END

) AS [France],

SUM

(

CASE WHEN t.TerritoryID = 8 THEN 1

ELSE 0

END

) AS [Germany],

SUM

(

CASE WHEN t.TerritoryID = 9 THEN 1

ELSE 0

END

) AS [Australia],

SUM

(

CASE WHEN t.TerritoryID = 10 THEN 1

ELSE 0

END

) AS [United Kingdom]

FROM Sales.SalesOrderHeader soh

INNER JOIN Sales.SalesTerritory t

ON soh.TerritoryID = t.TerritoryID;

***Listing 3-14.*** Examples Using the IIF statement

--Example 1. IIF Statement Using Variables

DECLARE @valueA int = 85

DECLARE @valueB int = 45

SELECT IIF (@valueA < @valueB, 'True', 'False') AS Result

--Example 2. IIF Statement Using Table Column

SELECT IIF (Name in ('Alberta', 'British Columbia'), 'Canada', Name)

FROM [Person].[StateProvince];

***Listing 3-15.*** Example Using the CHOOSE Statement

SELECT p.FirstName,

pp.PhoneNumber,

CHOOSE(pp.PhoneNumberTypeID, 'Cell', 'Home', 'Work') 'Phone Type'

FROM Person.Person p

JOIN Person.PersonPhone pp

ON p.BusinessEntityID = pp.BusinessEntityID;

***Listing 3-16.*** Using COALESCE with NULLIF

DECLARE @s varchar(10);

SELECT @s = 'A';

SELECT COALESCE(NULLIF(@s, 'A'), 'This is NULL or A');

***Listing 3-17.*** Sample Administrative Task Performed with a Cursor

CREATE PROCEDURE dbo.RebuildIndexes

@ShowOrRebuiId nvarchar(10) = N'show',

@MaxFrag decimal(20, 2) = 20.0

AS

BEGIN

-- Declare variables

SET NOCOUNT ON;

DECLARE

@Schema nvarchar(128),

@Table nvarchar(128),

@Index nvarchar(128),

@Sql nvarchar(4000),

@DatabaseId int,

@SchemaId int,

@TableId int,

@lndexId int;

-- Create the index list table

DECLARE @IndexList TABLE

(

DatabaseName nvarchar(128) NOT NULL,

DatabaseId int NOT NULL,

SchemaName nvarchar(128) NOT NULL,

SchemaId int NOT NULL,

TableName nvarchar(128) NOT NULL,

TableId int NOT NULL,

IndexName nvarchar(128),

IndexId int NOT NULL,

Fragmentation decimal(20, 2),

PRIMARY KEY (DatabaseId, SchemaId, TableId, IndexId) );

-- Populate index list table

INSERT INTO @IndexList

(

DatabaseName,

DatabaseId,

SchemaName,

SchemaId,

TableName,

TableId,

IndexName,

IndexId,

Fragmentation

)

SELECT

db\_name(),

db\_id(),

s.Name,

s.schema\_id,

t.Name,

t.object\_id,

i.Name,

i.index\_id,

MAX(ip.avg\_fragmentation\_in\_percent) FROM sys.tables t INNER JOIN sys.schemas s

ON t.schema\_id = s.schema\_id INNER JOIN sys.indexes i

ON t.object\_id = i.object\_id INNER JOIN sys.dm\_db\_index\_physical\_stats (db\_id(), NULL, NULL, NULL, NULL) ip

ON ip.object\_id = t.object\_id AND ip.index\_id = i.index\_id WHERE ip.database\_id = db\_id()

GROUP BY

s.Name,

s.schema\_id,

t.Name,

t.object\_id,

i.Name,

i.index\_id;

-- If user specified rebuiId, use a cursor to loop through all indexes

-- rebuiId them

IF @ShowOrRebuiId = N'rebuiId'

BEGIN

-- Declare a cursor to create the dynamic SQL statements

DECLARE Index\_Cursor CURSOR FAST\_FORWARD

FOR SELECT SchemaName, TableName, IndexName

FROM @IndexList

WHERE Fragmentation > @MaxFrag

ORDER BY Fragmentation DESC, TableName ASC, IndexName ASC;

-- Open the cursor for reading

OPEN Index\_Cursor;

-- Loop through all the tables in the database

FETCH NEXT FROM Index\_Cursor

INTO @Schema, @Table, @Index;

WHILE @@FETCH\_STATUS = 0 BEGIN -- Create ALTER INDEX statement to rebuiId inddex

SET @Sql = N'ALTER INDEX ' +

QUOTENAME(RTRIM(@Index)) + N' ON ' + QUOTENAME(RTRIM(@Table)) + N'.' +

QUOTENAME(RTRIM(@Table)) + N' REBUIId WITH (ONLINE = OFF); ';

PRINT @Sql;

-- Execute dynamic SQL

EXEC (@Sql);

-- Get the next index

FETCH NEXT FROM Index\_Cursor

INTO @Schema, @Table, @Index;

END

-- Close and deallocate the cursor.

CLOSE Index\_Cursor;

DEALLOCATE Index\_Cursor;

END

-- Show results, including oId fragmentation and new fragmentation

-- after index rebuiId

SELECT

il.DatabaseName,

il.SchemaName,

il.TableName,

il.IndexName,

il.Fragmentation AS FragmentationStart,

MAX(

CAST(ip.avg\_fragmentation\_in\_percent AS DECIMAL(20, 2))

) AS FragmentationEnd

FROM @IndexList il

INNER JOIN sys.dm\_db\_index\_physical\_stats(@DatabaseId, NULL, NULL, NULL, NULL) ip

ON DatabaseId = ip.database\_id

AND TableId = ip.object\_id

AND IndexId = ip.index\_id

GROUP BY

il.DatabaseName,

il.SchemaName,

il.TableName,

il.IndexName,

il.Fragmentation ORDER BY

Fragmentation DESC,

TableName ASC,

IndexName ASC;

RETURN;

END

GO

-- Execute index rebuild stored procedure

EXEC dbo.RebuildIndexes N'rebuild', 30;

CHAPTER 4 – User Defined Functions

***Listing 4-1.*** Simple Scalar UDF

CREATE FUNCTION dbo.CalculateCircleArea (@Radius float =1.0)

RETURNS float

WITH RETURNS NULL ON NULL INPUT

AS

BEGIN

RETURN PI() \* POWER(@Radius, 2);

END;

***Listing 4-2.*** Recursive Scalar UDF

CREATE FUNCTION dbo.CalculateFactorial (@n int = 1)

RETURNS decimal(38, 0)

WITH RETURNS NULL ON NULL INPUT

AS

BEGIN

RETURN

(CASE

WHEN @n <= 0 THEN NULL

WHEN @n > 1 THEN CAST(@n AS float) \* dbo.CalculateFactorial (@n - 1)

WHEN @n = 1 THEN 1

END);

END;

***Listing 4-3.*** Creating the NYSIIS Replacement Rules Table

-- Create the NYSIIS replacement rules table

CREATE TABLE dbo.NYSIIS\_Replacements

(Location nvarchar(10) NOT NULL,

NGram nvarchar(10) NOT NULL,

Replacement nvarchar(10) NOT NULL,

PRIMARY KEY (Location, NGram));

***Listing 4-4.*** INSERT Statement to Populate NYSIIS Replacement Rules Table

INSERT INTO NYSIIS\_Replacements (Location, NGram, Replacement)

VALUES(N'End', N'DT', N'DD'),

(N'End', N'EE', N'YY'),

(N'End', N'lE', N'YY'),

(N'End', N'ND', N'DD'),

(N'End', N'NT', N'DD'),

(N'End', N'RD', N'DD'),

(N'End', N'RT', N'DD'),

(N'Mid', N'A', N'A'),

(N'Mid', N'E', N'A'),

(N'Mid', N'T', N'A'),

(N'Mid', N'K', N'C'),

(N'Mid', N'M', N'N'),

(N'Mid', N'O', N'A'),

(N'Mid', N'Q', N'G'),

(N'Mid', N'U', N'A'),

(N'Mid', N'Z', N'S'),

(N'Mid', N'AW', N'AA'),

(N'Mid', N'EV', N'AF'),

(N'Mid', N'EW', N'AA'),

(N'Mid', N'lW', N'AA'),

(N'Mid', N'KN', N'NN'),

(N'Mid', N'OW', N'AA'),

(N'Mid', N'PH', N'FF'),

(N'Mid', N'UW', N'AA'),

(N'Mid', N'SCH', N'SSS'),

(N'Start', N'K', N'C'),

(N'Start', N'KN', N'NN'),

(N'Start', N'PF', N'FF'),

(N'Start', N'PH', N'FF'),

(N'Start', N'MAC', N'MCC'),

(N'Start', N'SCH', N'SSS');

GO

***Listng 4-5.*** Function to Encode Strings Using NYSIIS

CREATE FUNCTION dbo.EncodeNYSIIS

(

@String nvarchar(100)

)

RETURNS nvarchar(6)

WITH RETURNS NULL ON NULL INPUT

AS

BEGIN

DECLARE @Result nvarchar(100);

SET @Result = UPPER(@String);

-- Step 1: Remove All Nonalphabetic Characters

WITH Numbers (Num)

AS

(

SELECT 1

UNION ALL

SELECT Num + 1

FROM Numbers

WHERE Num < LEN(@Result)

)

SELECT @Result = STUFF

(

@Result,

Num,

1,

CASE WHEN SUBSTRING(@Result, Num, 1) >= N'A'

AND SUBSTRING(@Result, Num, 1) <= N'Z'

THEN SUBSTRING(@Result, Num, 1)

ELSE N'.'

END )

FROM Numbers;

SET @Result = REPLACE(@Result, N'.', N'');

-- Step 2: Replace the Start N-gram

SELECT TOP (1) @Result = STUFF

(

@Result,

1,

LEN(NGram),

Replacement

)

FROM dbo.NYSIIS\_Replacements

WHERE Location = N'Start'

AND SUBSTRING(@Result, 1, LEN(NGram)) = NGram

ORDER BY LEN(NGram) DESC;

-- Step 3: Replace the End N-gram

SELECT TOP (1) @Result = STUFF

(

@Result,

LEN(@Result) - LEN(NGram) + 1,

LEN(NGram),

Replacement

)

FROM dbo.NYSIIS\_Replacements

WHERE Location = N'End'

AND SUBSTRING(@Result, LEN(@Result) - LEN(NGram) + 1, LEN(NGram)) = NGram

ORDER BY LEN(NGram) DESC;

-- Step 4: Save the First Letter of the Name

DECLARE @FirstLetter nchar(1);

SET @FirstLetter = SUBSTRING(@Result, 1, 1);

-- Step 5: Replace All Middle N-grams

DECLARE @Replacement nvarchar(10);

DECLARE @i int;

SET @i = 1;

WHILE @i <= LEN(@Result)

BEGIN

SET @Replacement = NULL;

-- Grab the middle-of-name replacement n-gram

SELECT TOP (1) @Replacement = Replacement

FROM dbo.NYSIIS\_Replacements

WHERE Location = N'Mid'

AND SUBSTRING(@Result, @i, LEN(NGram)) = NGram

ORDER BY LEN(NGram) DESC;

SET @Replacement = COALESCE(@Replacement, SUBSTRING(@Result, @i, 1));

-- If we found a replacement, apply it

SET @Result = STUFF(@Result, @i, LEN(@Replacement), @Replacement)

-- Move on to the next n-gram

SET @i = @i + COALESCE(LEN(@Replacement), 1);

END;

-- Replace the first character with the first letter we saved at the start

SET @Result = STUFF(@Result, 1, 1, @FirstLetter);

-- Here we apply our special rules for the 'H' character. Special handling for 'W'

-- characters is taken care of in the replacement rules table

WITH Numbers (Num)

AS

(

SELECT 2 -- Don't bother with the first character

UNION ALL

SELECT Num + 1

FROM Numbers

WHERE Num < LEN(@Result)

)

SELECT @Result = STUFF

(

@Result,

Num,

1,

CASE SUBSTRING(@Result, Num, 1)

WHEN N'H' THEN

CASE WHEN SUBSTRING(@Result, Num + 1, 1)

NOT IN (N'A', N'E', N'I', N'O', N'U')

OR SUBSTRING(@Result, Num - 1, 1)

NOT IN (N'A', N'E', N'I', N'O', N'U')

THEN SUBSTRING(@Result, Num - 1, 1)

ELSE N'H'

END

ELSE SUBSTRING(@Result, Num, 1)

END

)

FROM Numbers;

-- Step 6: Reduce All Side-by-side Duplicate Characters

-- First replace the first letter of any sequence of two side-by-side

-- duplicate letters with a period

WITH Numbers (Num)

AS

(

SELECT 1

UNION ALL

SELECT Num + 1

FROM Numbers

WHERE Num < LEN(@Result)

)

SELECT @Result = STUFF

(

@Result,

Num,

1,

CASE SUBSTRING(@Result, Num, 1)

WHEN SUBSTRING(@Result, Num + 1, 1) THEN N'.'

ELSE SUBSTRING(@Result, Num, 1)

END

)

FROM Numbers;

-- Next replace all periods '.' with an empty string ''

SET @Result = REPLACE(@Result, N'.', N'');

-- Step 7: Remove Trailing 'S' Characters

WHILE RIGHT(@Result, 1) = N'S' AND LEN(@Result) > 1

SET @Result = STUFF(@Result, LEN(@Result), 1, N'');

-- Step 8: Remove Trailing 'A' Characters

WHILE RIGHT(@Result, 1) = N'A' AND LEN(@Result) > 1

SET @Result = STUFF(@Result, LEN(@Result), 1, N'');

-- Step 9: Replace Trailing 'AY' Characters with 'Y'

IF RIGHT(@Result, 2) = 'AY'

SET @Result = STUFF(@Result, LEN(@Result) - 1, 1, N'');

-- Step 10: Truncate Result to 6 Characters

RETURN COALESCE(SUBSTRING(@Result, 1, 6), '');

END;

GO

***Listing 4-6.*** Using NYSIIS to Phonetically Encode All AdventureWorks Contacts

SELECT LastName,

dbo.EncodeNYSIIS(LastName) AS NYSIIS

FROM Person.Person

GROUP BY LastName;

*L****isting 4-7.*** Retrieving All Contact Phonetic Matches for Liu

SELECT

BusinessEntityID,

LastName,

FirstName,

MiddleName,

dbo.EncodeNYSIIS(LastName) AS NYSIIS

FROM Person.Person

WHERE dbo.EncodeNYSIIS(LastName) = dbo.EncodeNYSIIS(N' Liu');

***Listing 4-8.*** Pre-encoding AdventureWorks Contact Names with NYSIIS

CREATE TABLE Person.ContactNYSIIS

(

BusinessEntityID int NOT NULL,

NYSIIS nvarchar(6) NOT NULL,

PRIMARY KEY(NYSIIS, BusinessEntityID)

);

GO

INSERT INTO Person.ContactNYSIIS

(

BusinessEntityID,

NYSIIS

)

SELECT

BusinessEntityID,

dbo.EncodeNYSIIS(LastName)

FROM Person.Person;

GO

***Listing 4-9.*** Efficient NYSIIS Query Using Pre-encoded Data

SELECT

cn.BusinessEntityID,

c.LastName,

c.FirstName,

c.MiddleName,

cn.NYSIIS

FROM Person.ContactNYSIIS cn

INNER JOIN Person.Person c

ON cn.BusinessEntityID = c.BusinessEntityID

WHERE cn.NYSIIS = dbo.EncodeNYSIIS(N'Liu');

***Listing 4-10.*** Creating a Numbers Table

USE [AdventureWorks2012]

GO

IF EXISTS (SELECT \* FROM sys.objects

WHERE object\_id = OBJECT\_ID(N'[dbo].[Numbers]')

AND type in (N'U'))

DROP TABLE [dbo].[Numbers];

-- Create a numbers table to allow the product pull list to be

-- created using set-based logic

CREATE TABLE dbo.Numbers (Num int NOT NULL PRIMARY KEY);

GO

-- Fill the numbers table with numbers from 0 to 30,000

WITH NumCTE (Num)

AS

(

SELECT 0

UNION ALL

SELECT Num + 1

FROM NumCTE

WHERE Num < 30000

)

INSERT INTO dbo.Numbers (Num) SELECT Num FROM NumCTE

OPTION (MAXRECURSION 0);

GO

***Listing 4-11.*** Creating a Product Pull List

CREATE FUNCTION dbo.GetProductPullList()

RETURNS @result table

(

SalesOrderID int NOT NULL,

ProductID int NOT NULL,

LocationID smallint NOT NULL,

Shelf nvarchar(10) NOT NULL,

Bin tinyint NOT NULL,

QuantityInBin smallint NOT NULL,

QuantityOnOrder smallint NOT NULL,

QuantityToPull smallint NOT NULL,

PartialFillFlag nchar(1) NOT NULL,

PRIMARY KEY (SalesOrderID, ProductID, LocationID, Shelf, Bin)

)

AS

BEGIN

INSERT INTO @result

(

SalesOrderID,

ProductID,

LocationID,

Shelf,

Bin,

QuantityInBin,

QuantityOnOrder,

QuantityToPull,

PartialFillFlag

)

SELECT

Order\_Details.SalesOrderID,

Order\_Details.ProductID,

Inventory\_Details.LocationID,

Inventory\_Details.Shelf,

Inventory\_Details.Bin,

Inventory\_Details.Quantity,

Order\_Details.OrderQty,

COUNT(\*) AS PullQty,

CASE WHEN COUNT(\*) < Order\_Details.OrderQty

THEN N'Y'

ELSE N'N'

END AS PartialFillFlag

FROM

(

SELECT ROW\_NUMBER() OVER

(

PARTITION BY p.ProductID

ORDER BY p.ProductID,

p.LocationID,

p.Shelf,

p.Bin

) AS Num,

p.ProductID,

p.LocationID,

p.Shelf,

p.Bin,

p.Quantity

FROM Production.ProductInventory p

INNER JOIN dbo.Numbers n

ON n.Num BETWEEN 1 AND Quantity

) Inventory\_Details

INNER JOIN

(

SELECT ROW\_NUMBER() OVER

(

PARTITION BY o.ProductID

ORDER BY o.ProductID,

o.SalesOrderID

) AS Num,

o.ProductID,

o.SalesOrderID,

o.OrderQty

FROM Sales.SalesOrderDetail o

INNER JOIN dbo.Numbers n

ON n.Num BETWEEN 1 AND o.OrderQty

) Order\_Details

ON Inventory\_Details.ProductID = Order\_Details.ProductID

AND Inventory\_Details.Num = Order\_Details.Num

GROUP BY

Order\_Details.SalesOrderID,

Order\_Details.ProductID,

Inventory\_Details.LocationID,

Inventory\_Details.Shelf,

Inventory\_Details.Bin,

Inventory\_Details.Quantity,

Order\_Details.OrderQty;

RETURN;

END;

GO

-- Fill the numbers table with numbers from 0 to 30,000

WITH NumCTE (Num)

AS

(

SELECT 0

UNION ALL

SELECT Num + 1

FROM NumCTE

WHERE Num < 30000

)

INSERT INTO dbo.Numbers (Num) SELECT Num FROM NumCTE

OPTION (MAXRECURSION 0);

GO

***Listing 4-12.*** Retrieving a Product Pull List with Product Names

SELECT

p.Name AS ProductName,

p.ProductNumber,

p.Color,

ppl.SalesOrderID,

ppl.ProductID,

ppl.LocationID,

ppl.Shelf,

ppl.Bin,

ppl.QuantityInBin,

ppl.QuantityOnOrder,

ppl.QuantityToPull,

ppl.PartialFillFlag

FROM Production.Product p

INNER JOIN dbo.GetProductPullList() ppl

ON p.ProductID = ppl.ProductID;

***Listing 4-13.*** Comma-Separated String-Splitting Function

CREATE FUNCTION dbo.GetCommaSplit (@String nvarchar(max))

RETURNS table

AS

RETURN

(

WITH Splitter (Num, String)

AS

(

SELECT Num, SUBSTRING(@String,

Num,

CASE CHARINDEX(N',', @String, Num)

WHEN 0 THEN LEN(@String) - Num + 1

ELSE CHARINDEX(N',', @String, Num) - Num

END

) AS String

FROM dbo.Numbers

WHERE Num <= LEN(@String)

AND (SUBSTRING(@String, Num - 1, 1) = N',' OR Num = 0)

)

SELECT

ROW\_NUMBER() OVER (ORDER BY Num) AS Num,

RTRIM(LTRIM(String)) AS Element

FROM Splitter

WHERE String <> ''

);

GO

***Listing 4-15.*** Using the FnCommaSplit Function

SELECT n.Num,

p.Name,

p.ProductNumber,

p.Color,

p.Size,

p.SizeUnitMeasureCode,

p.StandardCost,

p.ListPrice

FROM Production.Product p

INNER JOIN dbo.GetCommaSplit('FR-R38R-52,FR-M94S-52,FR-M94B-44,BK-M68B-38') n

ON p.ProductNumber = n.Element;

CHAPTER 7 – Encryption

***Listing 7-1.*** BACKUP and RESTORE SMK Examples

-- Back up the SMK to a file

BACKUP SERVICE MASTER KEY TO FILE = 'c:\CH07\S0L2012.SMK'

ENCRYPTION BY PASSWORD = 'p@$$w0rd';

-- Restore the SMK from a file

RESTORE SERVICE MASTER KEY FROM FILE = 'c:\CH07\S0L2012.SMK'

DECRYPTION BY PASSWORD = 'p@$$w0rd';

***Listing 7-2.*** Creating a Master Key

USE AdventureWorks2012;

GO

CREATE MASTER KEY

ENCRYPTION BY PASSWORD = 'p@$$w0rd' ;

***Listing 7-3.*** Turning off DMK Encryption by the SMK

ALTER MASTER KEY

DROP ENCRYPTION BY SERVICE MASTER KEY;

***Listing 7-4.*** Backing up and Restoring a DMK

USE AdventureWorks2012;

GO

OPEN MASTER KEY DECRYPTION BY PASSWORD = 'p@$$w0rd' ;

BACKUP MASTER KEY

TO FILE = 'c:\CH07\AdventureWorks2012.DMK'

ENCRYPTION BY PASSWORD = 'p@$$wOrd';

-- Restore DMK from backup

RESTORE MASTER KEY

FROM FILE = 'c:\CH07\AdventureWorks2012.DMK'

DECRYPTION BY PASSWORD = 'p@$$wOrd'

ENCRYPTION BY PASSWORD = '3rt = d4uy';

CLOSE MASTER KEY;

***Listing 7-5.*** Creating a Certificate on SQL Server

-- Back up DMK

CREATE CERTIFICATE TestCertificate

ENCRYPTION BY PASSWORD = 'p@$$wOrd'

WITH SUBJECT = 'Adventureworks2012 Test Certificate',

EXPIRY\_DATE = '2026-10-31';

***Listing 7-6.*** Backing up a Certificate

BACKUP CERTIFICATE TestCertificate

TO FILE = 'c:\CH07\TestCertificate.CER'

WITH PRIVATE KEY

(

FILE = 'c:\CH07\TestCertificate.PVK',

ENCRYPTION BY PASSWORD = ' 7&rtOxp2',

DECRYPTION BY PASSWORD = 'p@$$wOrd'

);

***Listing 7-7.*** Sample Encryption and Decryption by Certificate

-- Create a DMK

CREATE MASTER KEY

ENCRYPTION BY PASSWORD = 'P@55w0rd';

-- Create a certificate

CREATE CERTIFICATE TestCertificate

WITH SUBJECT = N'Adventureworks Test Certificate',

EXPIRY\_DATE = '2026-10-31';

-- Create the plain text data to encrypt

DECLARE @plaintext nvarchar(58) =

N'This is a test string to encrypt';

SELECT 'Plain text = ', @plaintext;

-- Encrypt the plain text by certificate

DECLARE @ciphertext varbinary(128) =

EncryptByCert(Cert\_ID('TestCertificate'), @plaintext);

SELECT 'Cipher text = ', @ciphertext;

-- Decrypt the cipher text by certificate

DECLARE @decryptedtext nvarchar(58) =

DecryptByCert(Cert\_ID('TestCertificate'), @ciphertext);

SELECT 'Decrypted text = ', @decryptedtext;

-- Drop the test certificate

DROP CERTIFICATE TestCertificate;

-- Drop the DMK

DROP MASTER KEY;

***Listing 7-8.*** Signing a Message with the SignByCert Function

-- Create a DMK

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'P@55w0rd';

-- Create a certificate

CREATE CERTIFICATE TestCertificate

WITH SUBJECT = 'Adventureworks Test Certificate',

EXPIRY\_DATE = '2026-10-31';

-- Create message

DECLARE @message nvarchar(4000) = N'Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation, so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this. ';

-- Sign the message by certificate

SELECT SignByCert(Cert\_ID(N'TestCertificate'), @message);

-- Drop the certificate

DROP CERTIFICATE TestCertificate;

-- Drop the DMK DROP MASTER KEY;

***Listing 7-9.*** Creating an Asymmetric Key Pair

CREATE ASYMMETRIC KEY TempAsymmetricKey WITH ALGORITHM = RSA\_1024;

***Listing 7-10.*** Encrypting and Decrypting with Asymmetric Keys

-- Create DMK

CREATE MASTER KEY

ENCRYPTION BY PASSWORD = 'P@55wOrd';

-- Create asymmetric key

CREATE ASYMMETRIC KEY TestAsymmetricKey WITH ALGORITHM = RSA\_512;

--Assign a credit card number to encrypt

DECLARE @CreditCard nvarchar(26) = N'9000 1234 5678 9012';

SELECT @CreditCard;

--Encrypt the credit card number

DECLARE @EncryptedCreditCard varbinary(64) =

EncryptByAsymKey(AsymKey\_ID(N'TestAsymmetricKey'), @CreditCard);

SELECT @EncryptedCreditCard;

--Decrypt the encrypted credit card number

DECLARE @DecryptedCreditCard nvarchar(26) =

DecryptByAsymKey(AsymKey\_ID(N'TestAsymmetricKey'), @EncryptedCreditCard);

SELECT @DecryptedCreditCard;

-- Drop asymmetric key

DROP ASYMMETRIC KEY TestAsymmetricKey;

--Drop DMK

DROP MASTER KEY;

***Listing 7-11.*** Signing a Message by Asymmetric Key

-- Create DMK

CREATE MASTER KEY

ENCRYPTION BY PASSWORD = 'P@55wOrd';

-- Create asymmetric key

CREATE ASYMMETRIC KEY TestAsymmetricKey WITH ALGORITHM = RSA\_512;

-- Create message

DECLARE @message nvarchar(4000) = N'Alas, poor Yorick!';

SELECT @message;

-- Sign message by asymmetric key

SELECT SignByAsymKey(AsymKey\_ID(N'TestAsymmetricKey'), @message);

-- Drop asymmetric key

DROP ASYMMETRIC KEY TestAsymmetricKey;

-- Drop DMK

DROP MASTER KEY;

***Listing 7-12.*** Creating a Symmetric Key

CREATE SYMMETRIC KEY TestSymmetricKey WITH ALGORITHM = AES\_128 ENCRYPTION BY PASSWORD = 'p@55wOrd';

***Listing 7-13.*** Changing the Symmetric Key Encryption Method

-- Create certificate to protect symmetric key

CREATE CERTIFICATE TestCertificate

WITH SUBJECT = 'AdventureWorks Test Certificate',

EXPIRY\_DATE = '2026-10-31';

CREATE SYMMETRIC KEY TestSymmetricKey WITH ALGORITHM = AES\_128 ENCRYPTION BY PASSWORD = 'p@55wOrd';

OPEN SYMMETRIC KEY TestSymmetricKey

DECRYPTION BY PASSWORD = 'p@55wOrd';

ALTER SYMMETRIC KEY TestSymmetricKey

ADD ENCRYPTION BY CERTIFICATE TestCertificate;

ALTER SYMMETRIC KEY TestSymmetricKey

DROP ENCRYPTION BY PASSWORD = 'p@55wOrd';

CLOSE SYMMETRIC KEY TestSymmetricKey;

-- Drop the symmetric key

DROP SYMMETRIC KEY TestSymmetricKey;

-- Drop the certificate

DROP CERTIFICATE TestCertificate;

***Listing 7-14.*** *Encrypting Data with a Symmetric Key*

-- Create a temporary table to hold results

CREATE TABLE #TempNames

(

BusinessEntityID int PRIMARY KEY,

FirstName nvarchar(50),

MiddleName nvarchar(50),

LastName nvarchar(50),

EncFirstName varbinary(200),

EncMiddleName varbinary(200),

EncLastName varbinary(200)

);

-- Create DMK

CREATE MASTER KEY

ENCRYPTION BY PASSWORD = 'Test\_P@sswOrd';

-- Create certificate to protect symmetric key

CREATE CERTIFICATE TestCertificate

WITH SUBJECT = 'AdventureWorks Test Certificate',

EXPIRY\_DATE = '2026-10-31';

-- Create symmetric key to encrypt data

CREATE SYMMETRIC KEY TestSymmetricKey

WITH ALGORITHM = AES\_128

ENCRYPTION BY CERTIFICATE TestCertificate;

-- Open symmetric key

OPEN SYMMETRIC KEY TestSymmetricKey

DECRYPTION BY CERTIFICATE TestCertificate;

-- Populate temp table with 100 encrypted names from the Person.Person table

INSERT

INTO #TempNames

(

BusinessEntityID,

EncFirstName,

EncMiddleName,

EncLastName

)

SELECT TOP(100) BusinessEntityID,

EncryptByKey(Key\_GUID(N'TestSymmetricKey'), FirstName),

EncryptByKey(Key\_GUID(N'TestSymmetricKey'), MiddleName),

EncryptByKey(Key\_GUID(N'TestSymmetricKey'), LastName)

FROM Person.Person

ORDER BY BusinessEntityID;

-- Update the temp table with decrypted names

UPDATE #TempNames

SET FirstName = DecryptByKey(EncFirstName),

MiddleName = DecryptByKey(EncMiddleName),

LastName = DecryptByKey(EncLastName);

-- Show the results

SELECT BusinessEntityID,

FirstName,

MiddleName,

LastName,

EncFirstName,

EncMiddleName,

EncLastName

FROM #TempNames;

-- Close the symmetric key

CLOSE SYMMETRIC KEY TestSymmetricKey;

-- Drop the symmetric key

DROP SYMMETRIC KEY TestSymmetricKey;

-- Drop the certificate

DROP CERTIFICATE TestCertificate;

--Drop the DMK

DROP MASTER KEY;

--Drop the temp table

DROP TABLE #TempNames;

***Listing 7-15.*** Encryption and Decryption by Passphrase and Byte Hashing

DECLARE @cleartext nvarchar(256);

DECLARE @encrypted varbinary(512);

DECLARE @decrypted nvarchar(256);

SELECT @cleartext = N'To be, or not to be: that is the question: ' +

N'Whether ''tis nobler in the mind to suffer ' +

N'The slings and arrows of outrageous fortune, ' +

N'Or to take arms against a sea of troubles';

SELECT @encrypted = EncryptByPassPhrase(N'Shakespeare''s Donkey', @cleartext);

SELECT @decrypted = CAST

(

DecryptByPassPhrase(N'Shakespeare''s Donkey', @encrypted)

AS nvarchar(128)

);

SELECT @cleartext AS ClearText;

SELECT @encrypted AS Encrypted;

SELECT @decrypted AS Decrypted;

SELECT HashBytes ('SHA1', @ClearText) AS Hashed;

***Listing 7-16.*** Enabling EKM Providers

sp\_configure 'show advanced', 1;

GO

RECONFIGURE;

GO

sp\_configure 'EKM provider enabled', 1;

GO

RECONFIGURE;

GO

***Listing 7-17.*** Registering a Cryptographic Provider

CREATE CRYPTOGRAPHIC PROVIDER Eagle\_EKM\_Provider

FROM FILE = 'c:\Program Files\Eagle\_EKM\SQLEKM.DLL';

GO

***Listing 7-18.*** Creating an Asymmetric Key with HSM Protection

CREATE ASYMMETRIC KEY AsymKeyEKMProtected

FROM PROVIDER Eagle\_EKM\_Provider

WITH PROVIDER\_KEY\_NAME = 'EKM\_Key\_1',

CREATION\_DISPOSITION = OPEN\_EXISTING;

GO

***Listing 7-19.*** Creating a Server Certificate

CREATE CERTIFICATE ServerCert

WITH SUBJECT = 'Server Certificate for TDE',

EXPIRY\_DATE = '2022-12-31';

GO

***Listing 7-20.*** Creating a Database Encryption Key and Securing the Database

USE AdventureWorks2012;

GO

CREATE DATABASE ENCRYPTION KEY

WITH ALGORITHM = AES\_128

ENCRYPTION BY SERVER CERTIFICATE ServerCert;

GO

ALTER DATABASE AdventureWorks2012

SET ENCRYPTION ON;

GO

CHAPTER 10 – Full-Text Search

***Listing 10-1.*** Creating a Full-Text Catalog with T-SQL

CREATE FULLTEXT CATALOG AdventureWorksFTCat

WITH ACCENT\_SENSITIVITY = OFF

AS DEFAULT

AUTHORIZATION dbo;

***Listing 10-2.*** Creating a Full-Text Index with T-SQL Statements

CREATE FULLTEXT INDEX

ON Production.ProductModel

(

CatalogDescription LANGUAGE English,

Instructions LANGUAGE English,

Name LANGUAGE English

)

KEY INDEX PK\_ProductModel\_ProductModelID

ON

(

AdventureWorksFTCat

)

WITH

(

CHANGE\_TRACKING AUTO

);

GO

ALTER FULLTEXT INDEX

ON Production.ProductModel ENABLE;

GO

***Listing 10-3.*** Simple FREETEXT Full-Text Query

SELECT

ProductModelID,

Name,

CatalogDescription,

Instructions

FROM Production.ProductModel

WHERE FREETEXT(\*, N'sock');

***Listing 10-4.*** FREETEXT Query with Automatic Word Stemming

SELECT

ProductModelID,

Name,

CatalogDescription,

Instructions

FROM Production.ProductModel

WHERE FREETEXT(CatalogDescription, N'weld', LANGUAGE 1033);

***Listing 10-5.*** Simple CONTAINS Query

SELECT

ProductModelID ,

Name,

CatalogDescription,

Instructions

FROM Production.ProductModel

WHERE CONTAINS (\*, N’weld’);

***Listing 10-6.*** Sample CONTAINS Query with FORMSOF Inflectional Generation Term

SELECT

ProductModelID ,

Name,

CatalogDescription

FROM Production.ProductModel

WHERE CONTAINS

(

(

Name,

CatalogDescription

),

N'FORMSOF(INFLECTIONAL, sport)'

);

***Listing 10-7.*** Compound CONTAINS Search Term

SELECT

ProductModelID ,

Name,

CatalogDescription

FROM Production.ProductModel

WHERE CONTAINS

(

(

Name,

CatalogDescription

),

N'"tube" | FORMSOF (INFLECTIONAL, sport)'

);

***Listing 10-8.*** CONTAINS Prefix Search

SELECT

ProductModelID ,

Name

FROM Production.ProductModel

WHERE CONTAINS (Name, N'"bot\*"');

***Listing 10-9.*** CONTAINS Proximity Search

SELECT

ProductModelID ,

Name

FROM Production.ProductModel

WHERE CONTAINS (Instructions, N'aluminum NEAR jig');

***Listing 10-10.*** CONTAINS Custom Search

SELECT

ProductModelID ,

Name

FROM Production.ProductModel

WHERE CONTAINS(Instructions, 'NEAR((aluminum,jig), 3)');

***Listing 10-11.*** Custom Search with TRUE Clause

SELECT

ProductModelID ,

Name

FROM Production.ProductModel

WHERE CONTAINS(Instructions, 'NEAR((jig, aluminum),3, TRUE)');

SELECT

ProductModelID ,

Name

FROM Production.ProductModel

WHERE CONTAINS(Instructions, 'NEAR((aluminum, jig),3, TRUE)');

***Listing 10-12.*** FREETEXTTABLE Results Joined to Source Table

SELECT

ftt.[KEY],

ftt.[RANK],

pm.ProductModelID ,

pm.Name FROM FREETEXTTABLE

(

Production.ProductModel,

Instructions,

N'aluminum'

) ftt

INNER JOIN Production.ProductModel pm

ON ftt.[KEY] = pm.ProductModelID;

***Listing 10-13.*** Simple CONTAINSTABLE Query

SELECT

[KEY],

[RANK]

FROM CONTAINSTABLE (

Production.ProductModel,

[Name],

N'FORMSOF(INFLECTIONAL, tours)'

);

***Listing 10-14.*** ISABOUT in a CONTAINSTABLE Query

SELECT

ct.[RANK],

ct.[KEY],

pm.[Name]

FROM CONTAINSTABLE

(

Production.ProductModel,

Instructions,

N'aluminum OR polish'

) ct

INNER JOIN Production.ProductModel pm

ON ct.[KEY] = pm.ProductModelID

ORDER BY ct.[RANK] DESC;

SELECT

ct.[RANK],

ct.[KEY],

pm.[Name] FROM CONTAINSTABLE

(

Production.ProductModel,

Instructions,

N'ISABOUT(aluminum WEIGHT(1.0 ), polish WEIGHT(0.1))'

) ct

INNER JOIN Production.ProductModel pm

ON ct.[KEY] = pm.ProductModelID

ORDER BY ct.[RANK] DESC;

***Listing 10-15.*** *Creating a Full-Text Index*

CREATE FULLTEXT INDEX ON Production.Product

(

Name LANGUAGE English,

Color LANGUAGE English

)

KEY INDEX PK\_Product\_ProductID

ON (AdventureWorksFTCat)

WITH

(

CHANGE\_TRACKING AUTO,

STOPLIST = SYSTEM

);

GO

ALTER FULLTEXT INDEX ON Production.Product

ENABLE;

GO

***Listing 10-16.*** *Tsenu.xml US English XML Thesaurus File*

<XML ID = "Microsoft Search Thesaurus">

<thesaurus xmlns = "x-schema:tsSchema.xml">

<diacritics\_sensitive > 0</diacritics\_sensitive>

<expansion>

<sub > thin</sub>

<sub > flat</sub>

</expansion>

<replacement>

<pat > sapphire</pat>

<pat > indigo</pat>

<pat > navy</pat>

<sub > blue</sub>

</replacement>

</thesaurus>

</XML>

***Listing 10-17.*** Reloading US English XML Thesaurus

EXEC sys.sp\_fulltext\_load\_thesaurus\_file 1033;

GO

***Listing 10-18.*** FREETEXT Query with Thesaurus Expansion Sets

SELECT

ProductID,

Name

FROM Production.Product

WHERE FREETEXT(\*, N'flat');

***Listing 10-19.*** FREETEXT Query with Thesaurus Replacement Patterns

SELECT

ProductID,

Name,

Color

FROM Production.Product

WHERE FREETEXT(\*, N'navy');

***Listing 10-20.*** Creating a Full-Text Stoplist

CREATE FULLTEXT STOPLIST AWStoplist

FROM SYSTEM STOPLIST;

GO

***Listing 10-21.*** Adding the Word “Instructions” to the Stoplist

ALTER FULLTEXT STOPLIST AWStoplist

ADD N'instructions' LANGUAGE English;

GO

ALTER FULLTEXT INDEX ON Production.ProductModel

SET STOPLIST AWStoplist;

GO

***Listing 10-22.*** Full-Text Query with Newly Created Stoplist

SELECT

ProductModelID,

Name

FROM Production.ProductModel

WHERE FREETEXT(\*, N'instructions');

***Listing 10-23.*** Using Sys.dm\_fts\_parser to See Word Breaking and Stemming

SELECT

keyword,

group\_id,

phrase\_id,

occurrence,

special\_term,

display\_term,

expansion\_type,

source\_term

FROM sys.dm\_fts\_parser

(

N'FORMSOF(FREETEXT,had)',

1033,

NULL,

0

);

***Listing 10-24.*** Initializing the Statistical Semantics Database

EXEC sp\_fulltext\_semantic\_register\_language\_statistics\_db @dbname = N'semanticsdb';

***Listing 10-26.*** Using the Semantickeyphrasetable Function

SELECT TOP(10) KEYP\_TBL.keyphrase

FROM SEMANTICKEYPHRASETABLE

(

Production.ProductModel,

Name

) AS KEYP\_TBL

ORDER BY KEYP\_TBL.score DESC;

GO

CHAPTER 11 – XML

***Listing 11-1.*** Simple OPENXML Query

DECLARE @docHandle int;

DECLARE @xmlDocument nvarchar(max) = N' < Customers>

<Customer CustomerID = "1234" ContactName = "Larry" CompanyName = "APress">

<Orders>

<Order CustomerID = "1234" OrderDate = "2006-04-25T13:22:18"/>

<Order CustomerID = "1234" OrderDate = "2006-05-10T12:35:49"/>

</Orders>

</Customer>

<Customer CustomerID = "4567" ContactName = "Bill" CompanyName = "Microsoft">

<Orders>

<Order CustomerID = "4567" OrderDate = "2006-03-12T18:32:39"/>

<Order CustomerID = "4567" OrderDate = "2006-05-11T17:56:12"/>

</Orders>

</Customer>

</Customers > ';

EXECUTE sp\_xml\_preparedocument @docHandle OUTPUT, @xmlDocument;

SELECT

Id,

ParentId,

NodeType,

LocalName,

Prefix,

NameSpaceUri,

DataType,

Prev,

[Text]

FROM OPENXML(@docHandle, N'/Customers/Customer');

EXECUTE sp\_xml\_removedocument @docHandle;

GO

***Listing 11-2.*** OPENXML and WITH Clause, Explicit Schema

DECLARE @docHandle int;

DECLARE @xmlDocument nvarchar(max) = N' < Customers>

<Customer CustomerID = "1234" ContactName = "Larry" CompanyName = "APress">

<Orders>

<Order CustomerID = "1234" OrderDate = "2006-04-25T13:22:18"/>

<Order CustomerID = "1234" OrderDate = "2006-05-10T12:35:49"/>

</Orders>

</Customer>

<Customer CustomerID = "4567" ContactName = "Bill" CompanyName = "Microsoft">

<Orders>

<Order CustomerID = "4567" OrderDate = "2006-03-12T18:32:39"/>

<Order CustomerID = "4567" OrderDate = "2006-05-11T17:56:12"/>

</Orders>

</Customer>

</Customers > ';

EXECUTE sp\_xml\_preparedocument @docHandle OUTPUT, @xmlDocument;

SELECT

CustomerID,

CustomerName,

CompanyName,

OrderDate

FROM OPENXML(@docHandle, N'/Customers/Customer/Orders/Order')

WITH

(

CustomerID nchar(4) N'../../@CustomerID',

CustomerName nvarchar(50) N'../../@ContactName',

CompanyName nvarchar(50) N'../../@CompanyName',

OrderDate datetime

);

EXECUTE sp\_xml\_removedocument @docHandle;

***Listing 11-3.*** OPENXML with WITH Clause, Existing Table Schema

DECLARE @docHandle int;

DECLARE @xmlDocument nvarchar(max) = N' < Customers>

<Customer CustomerID = "1234" ContactName = "Larry" CompanyName = "APress">

<Orders>

<Order CustomerID = "1234" OrderDate = "2006-04-25T13:22:18"/>

<Order CustomerID = "1234" OrderDate = "2006-05-10T12:35:49"/>

</Orders>

</Customer>

<Customer CustomerID = "4567" ContactName = "Bill" CompanyName = "Microsoft">

<Orders>

<Order CustomerID = "4567" OrderDate = "2006-03-12T18:32:39"/>

<Order CustomerID = "4567" OrderDate = "2006-05-11T17:56:12"/>

</Orders>

</Customer>

</Customers > ';

EXECUTE sp\_xml\_preparedocument @docHandle OUTPUT, @xmlDocument;

CREATE TABLE #CustomerInfo

(

CustomerID nchar(4) NOT NULL,

ContactName nvarchar(50) NOT NULL,

CompanyName nvarchar(50) NOT NULL

);

CREATE TABLE #OrderInfo

(

CustomerID nchar(4) NOT NULL,

OrderDate datetime NOT NULL

);

INSERT INTO #CustomerInfo

(

CustomerID,

ContactName,

CompanyName

)

SELECT

CustomerID,

ContactName,

CompanyName

FROM OPENXML(@docHandle, N'/Customers/Customer')

WITH #CustomerInfo;

INSERT INTO #OrderInfo

(

CustomerID,

OrderDate

)

SELECT

CustomerID,

OrderDate

FROM OPENXML(@docHandle, N'//Order')

WITH #OrderInfo;

SELECT

c.CustomerID,

c.ContactName,

c.CompanyName,

o.OrderDate

FROM #CustomerInfo c

INNER JOIN #OrderInfo o

ON c.CustomerID = o.CustomerID;

DROP TABLE #OrderInfo;

DROP TABLE #CustomerInfo;

EXECUTE sp\_xml\_removedocument @docHandle;

GO

***Listing 11-4.*** Sample FOR XML RAW Query

USE AdventureWorks2012;

GO

SELECT

ProductID,

Name,

ProductNumber

FROM Production.Product

WHERE ProductID IN (770, 903)

FOR XML RAW;

***Listing 11-5.*** FOR XML AUTO Query on a Single Table

USE AdventureWorks2012;

GO

SELECT

ProductID,

Name,

ProductNumber

FROM Production.Product

WHERE ProductID IN (770, 903)

FOR XML AUTO;

***Listing 11-6.*** FOR XML AUTO Query with a Join

SELECT

Product.ProductID,

Product.Name,

Product.ProductNumber,

Inventory.Quantity

FROM Production.Product Product

INNER JOIN Production.ProductInventory Inventory

ON Product.ProductID = Inventory.ProductID

WHERE Product.ProductID IN (770, 3)

FOR XML AUTO;

***Listing 11-7.*** FOR XML AUTO Query with ELEMENTS Option

SELECT

ProductID,

Name,

ProductNumber

FROM Production.Product

WHERE ProductID = 770

FOR XML AUTO, ELEMENTS;

***Listing 11-8.*** FOR XML EXPLICIT Query

SELECT

1 AS Tag,

NULL AS Parent,

ProductID AS [Products!1!ProductID!element],

Name AS [Products!1!ProductName],

ProductNumber AS [Products!1!ProductNumber],

NULL AS [Products!2!Quantity]

FROM Production.Product

WHERE ProductID IN (770, 3)

UNION ALL

SELECT

2 AS Tag,

1 AS Parent,

NULL,

NULL,

NULL,

Quantity

FROM Production.ProductInventory

WHERE ProductID IN (770, 3)

FOR XML EXPLICIT;

***Listing 11-9.*** FOR XML PATH Query

SELECT

p.ProductID AS "Product/@ID",

p.Name AS "Product/Name",

p.ProductNumber AS "Product/Number",

i.Quantity AS "Product/Quantity"

FROM Production.Product p

INNER JOIN Production.ProductInventory i

ON p.ProductID = i.ProductID

WHERE p.ProductID = 770

FOR XML PATH;

***Listing 11-10.*** Untyped xml Variable and Column Declarations

DECLARE @x XML;

CREATE TABLE XmlPurchaseOrders

(

PoNum int NOT NULL PRIMARY KEY,

XmlPurchaseOrder xml );

***Listing 11-11.*** Populating an Untyped xml Variable

DECLARE @x xml = N' < ?xml version = "1.0" ?>

<Address>

<Latitude > 47.642737</Latitude>

<Longitude > −122.130395</Longitude>

<Street > ONE MICROSOFT WAY</Street>

<City > REDMOND</City>

<State > WA</State>

<Zip > 98052</Zip>

<Country > US</Country>

</Address > ';

SELECT @x;

***Listing 11-12.*** Creating a Typed xml Variable

CREATE XML SCHEMA COLLECTION AddressSchemaCollection

AS N'<?xml version="1.0" encoding="utf-16" ?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="Address">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="Latitude" type="xsd:decimal" />

<xsd:element name="Longitude" type="xsd:decimal" />

<xsd:element name="Street" type="xsd:string" />

<xsd:element name="City" type="xsd:string" />

<xsd:element name="State" type="xsd:string" />

<xsd:element name="Zip" type="xsd:string" />

<xsd:element name="Country" type="xsd:string" />

</xsd:sequence>

</xsd:complexType>

</xsd:element>

</xsd:schema>';

GO

DECLARE @x XML (CONTENT AddressSchemaCollection);

SELECT @x = N'<?xml version="1.0" ?>

<Address>

<Latitude>47.642737</Latitude>

<Longitude>-122.130395</Longitude>

<Street>ONE MICROSOFT WAY</Street>

<City>REDMOND</City>

<State>WA</State>

<Zip>98052</Zip>

<Country>US</Country>

</Address>';

SELECT @x;

DROP XML SCHEMA COLLECTION AddressSchemaCollection;

GO

***Listing 11-13.*** Using the Query Method on the HumanResources.JobCandidate Resume XML

SELECT Resume.query(N'declare namespace ns =

"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";

/ns:Resume/ns:Name') AS [NameXML]

FROM HumanResources.JobCandidate;

***Listing 11-14.*** xml Data Type Value Method Sample

SELECT Resume.value (N'declare namespace ns =

"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";

(/ns:Resume/ns:Name/ns:Name.Last)[1]',

'nvarchar(100)') AS [LastName]

FROM HumanResources.JobCandidate;

***Listing 11-15.*** xml Data Type Exist Method Example

SELECT Resume.value (N'declare namespace ns =

"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";

(/ns:Resume/ns:Name/ns:Name.Last) [1]',

'nvarchar(100)') AS [BachelorsCandidate]

FROM HumanResources.JobCandidate

WHERE Resume.exist (N'declare namespace ns =

"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";

/ns:Resume/ns:Education/ns:Edu.Level [ . = "Bachelor" ]') = 1;

***Listing 11-16.*** xml Data Type Nodes Example

SELECT

ProductModelID,

Name,

Specs.query('.') AS Result

FROM Production.ProductModel

CROSS APPLY CatalogDescription.nodes('declare namespace ns =

"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/ProductModelDescription";

/ns:ProductDescription/ns:Specifications/Material/text()

[ contains ( . , "Alloy" ) ]')

AS NodeTable(Specs);

***Listing 11-17.*** xml Data Type Modify Method Example

DECLARE @x xml = N' < ?xml version = "1.0" ?>

<Address>

<Street > l MICROSOFT WAY</Street>

<City > REDMOND</City>

<State > WA</State>

<Zip > 98052</Zip>

<Country > US</Country>

<Website>http://www.microsoft.com</Website>

</Address > ';

SELECT @x;

SET @x.modify ('insert

(

<CompanyName > Microsoft Corporation</CompanyName>,

<Url>http://msdn.microsoft.com</Url>,

<UrlDescription > Microsoft Developer Network</UrlDescription>

)

into (/Address)[1] ');

SET @x.modify('replace value of

(/Address/Street/text())[1]

with "ONE MICROSOFT WAY"

');

SET @x.modify('

delete /Address/Website

');

SELECT @x;

***Listing 11-18.*** Retrieving Job Candidates with Bachelor’s Degrees

SELECT Resume.query('declare namespace ns =

"http://schemas.microsoft.com/sqlserver/2004/07/adventure-works/Resume";

for $m in /ns:Resume

where $m/ns:Education/ns:Edu.Level[. = "Bachelor" ]

return < Name>

{

data(($m/ns:Name/ns:Name.Last)[1]),

data(($m/ns:Name/ns:Name.First)[1]),

data(($m/ns:Name/ns:Name.Middle)[1])

} </Name > ')

FROM HumanResources.JobCandidate;

GO

***Listing 11-19.*** Adding XML Indexes to the Resume Column

CREATE PRIMARY XML INDEX PXML\_JobCandidate

ON HumanResources.JobCandidate (Resume);

GO

CREATE XML INDEX IXML\_Education

ON HumanResources.JobCandidate (Resume)

USING XML INDEX PXML\_JobCandidate

FOR PATH;

GO

***Listing 11-20.*** Using FOR XML to Convert Relational Data to Populate an xml Variable

DECLARE @xml xml =

(

SELECT

p.ProductNumber AS "@Id",

p.Name AS "Name",

p.Color AS "Color",

p.ListPrice AS "ListPrice",

p.SizeUnitMeasureCode AS "Size/@UOM",

p.Size AS "Size",

p.WeightUnitMeasureCode AS "Weight/@UOM",

p.Weight AS "Weight",

(

SELECT COALESCE(SUM(i.Quantity), 0)

FROM Production.ProductInventory i

WHERE i.ProductID = p.ProductID

) AS "QuantityOnHand"

FROM Production.Product p

WHERE p.FinishedGoodsFlag = 1

ORDER BY p.Name

FOR XML PATH ('Product'),

ROOT ('Products')

);

SELECT @xml;

***Listing 11-21.*** XSLT Style Sheet to Convert Data to HTML

DECLARE @xslt xml = N' < ?xml version = "1.0" encoding = "utf-16"?>

<xsl:stylesheet version = "1.0"

xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

<xsl:template match = "/Products">

<html>

<head>

<title > AdventureWorks Product Listing Report</title>

<style type = "text/css">

tr.row-heading {

background-color: 000099;

color: ffffff;

font-family: tahoma, arial, helvetica, sans-serif;

font-size: 12px;

}

tr.row-light {

background-color: ffffff;

font-family: tahoma, arial, helvetica, sans-serif;

font-size: 12px;

}

tr.row-dark {

background-color: 00ffff;

font-family: tahoma, arial, helvetica, sans-serif;

font-size: 12px;

}

td.col-right {

text-align: right;

}

</style>

</head>

<body>

<table>

<tr class = "row-heading">

<th > ID</th>

<th > Product Name</th>

<th > On Hand</th>

<th > List Price</th>

<th > Color</th>

<th > Size</th>

<th > Weight</th>

</tr>

<xsl:for-each select = "Product">

<xsl:element name = "tr">

<xsl:choose>

<xsl:when test = "position() mod 2 = 0">

<xsl:attribute name = "class" > row-light</xsl:attribute>

</xsl:when>

<xsl:otherwise>

<xsl:attribute name = "class" > row-dark</xsl:attribute>

</xsl:otherwise>

</xsl:choose>

<td > <xsl:value-of select = "@Id"/></td>

<td > <xsl:value-of select = "Name"/></td>

<td class = "col-right">

<xsl:value-of select = "QuantityOnHand"/>

</td>

<td class = "col-right" > <xsl:value-of select = "ListPrice"/></td>

<td > <xsl:value-of select = "Color"/></td>

<td class = "col-right"> <xsl:value-of select = "Size"/>

<xsl:value-of select = "Size/@UOM"/>

</td>

<td class = "col-right">

<xsl:value-of select = "Weight"/>

<xsl:value-of select = "Weight/@UOM"/>

</td>

</xsl:element>

</xsl:for-each>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet > ';

***Listing 11-22.*** SQL CLR SP for XSL Transformations

using System.Data.SqlTypes;

using System.Xml;

using System.Xml.Xsl;

namespace Apress.Samples

{

public partial class XSLT

{

[Microsoft.SqlServer.Server.SqlProcedure]

public static void XmlToHtml

(

SqlXml RawXml,

SqlXml XslStyleSheet,

SqlString OutputPage

)

{

// Create and load the XslCompiledTransform object

XslCompiledTransform xslt = new XslCompiledTransform();

XmlDocument xmldoc1 = new XmlDocument();

xmldocl.LoadXml(XslStyleSheet.Value);

xslt.Load(xmldoc1);

// Create and load the Raw XML document

XmlDocument xml = new XmlDocument();

xml.LoadXml(RawXml.Value);

// Create the XmlTextWriter for output to HTML document

XmlTextWriter htmlout = new XmlTextWriter

(

OutputPage.Value,

System.Text.Encoding.Unicode

);

// Perform the transformation

xslt.Transform

(

xml,

htmlout

);

// Close the XmlTextWriter

htmlout.Close();

}

}

};

***Listing 11-23.*** Performing a SQL CLR XSL Transformation

EXECUTE XmlToHtml @xml,

gxslt,

'c:\Documents and SettingsXAll Users\Documents\adventureworks-inventory.html';

CHAPTER 17 – Error Handling and Dynamic SQL

***Listing 17-1.*** Incorrect Error Handling with @@error

CREATE PROCEDURE dbo.TestError (@e int OUTPUT)

AS

BEGIN

INSERT INTO Person.Person(BusinessEntityID)

VALUES (1);

PRINT N'Error code in procedure = ' + CAST(@@error AS nvarchar(10));

SET @e = @@error;

END

GO

DECLARE @ret int,

@e int;

EXEC @ret = dbo.TestError @e OUTPUT;

PRINT N'Returned error code = ' + CAST(@e AS nvarchar(10));

PRINT N'Return value = ' + CAST(@ret AS nvarchar(10));

***Listing 17-2.*** Corrected Error Handling with @@error

CREATE PROCEDURE dbo.TestError2 (@e int OUTPUT)

AS

BEGIN

INSERT INTO Person.Person(BusinessEntityID)

VALUES (1);

SET @e = @@error;

PRINT N'Error code in procedure = ' + CAST(@e AS nvarchar(10));

END

GO

DECLARE @ret int,

@e int;

EXEC @ret = dbo.TestError2 @e OUTPUT;

PRINT N'Returned error code = ' + CAST(@e AS nvarchar(10));

PRINT N'Return value = ' + CAST(@ret AS nvarchar(10));

***Listing 17-3.*** Raising a Custom Exception with RAISERROR

RAISERROR ('This is an exception.', 17, 127);

***Listing 17-4.*** Sample TRY . . . CATCH Error Handling

CREATE PROCEDURE dbo.TestError3 (@e int OUTPUT)

AS

BEGIN

SET @e = 0;

BEGIN TRY

INSERT INTO Person.Address (AddressID)

VALUES (1);

END TRY

BEGIN CATCH

SET @e = ERROR\_NUMBER();

PRINT N'Error Code = ' + CAST(@e AS nvarchar(10));

PRINT N'Error Procedure = ' + ERROR\_PROCEDURE();

PRINT N'Error Message = ' + ERROR\_MESSAGE();

END CATCH

END

GO

DECLARE @ret int,

@e int;

EXEC @ret = dbo.TestError3 @e OUTPUT;

PRINT N'Error code = ' + CAST(@e AS nvarchar(10));

PRINT N'Return value = ' + CAST(@ret AS nvarchar(10));

***Listing 17-5.*** Examples of *TRY\_PARSE*

DECLARE @fauxdate AS varchar(10)

DECLARE @realdate AS VARCHAR(10)

SET @fauxdate = 'iamnotadate'

SET @realdate = '01/05/2012'

SELECT TRY\_PARSE(@fauxdate AS DATE);

SELECT TRY\_PARSE(@realdate AS DATE);

SELECT TRY\_PARSE(@realdate AS DATE USING 'Fr-FR');

SELECT IIF(TRY\_PARSE(@realdate AS DATE) IS NULL, 'False', 'True')

***Listing 17-6.*** *TRY\_CONVERT* Examples

DECLARE @sampletext AS VARCHAR(10)

SET @sampletext = '123456'

SELECT TRY\_CONVERT(INT, @ sampletext);

SELECT TRY\_CONVERT(DATE, @ sampletext);

SELECT IIF(TRY\_CONVERT(binary, @ sampletext) IS NULL, 'FALSE', 'TRUE');

***Listing 17-7.*** Examples Using TRY\_CAST

DECLARE @sampletext AS VARCHAR(10)

SET @sampletext = '123456'

SELECT TRY\_CAST(@sampletext AS INT);

SELECT TRY\_CAST(@sampletext AS DATE);

SELECT IIF(TRY\_CAST(@sampletext AS BINARY) IS NULL, 'FALSE', 'TRUE');

***Listing 17-8.*** Examples of the THROW Statement

--1. Using THROW without paramaters

BEGIN TRY

INSERT INTO Person.Address (AddressID)

VALUES (1);

END TRY

BEGIN CATCH

PRINT 'This is an error';

THROW

END CATCH ;

--2. Using THROW with parameters

THROW 52000, 'This is also an error', 1

BEGIN TRY

INSERT INTO Person.Address (AddressID)

VALUES (1);

END TRY

BEGIN CATCH

THROW

END CATCH

***Listing 17-9.*** Debugging Script with PRINT

DECLARE @i int;

PRINT N'Initial value of @i = ' + COALESCE(CAST(@i AS nvarchar(10)), N'NULL');

SET @i + = 10;

PRINT N'Final value of @i = ' + COALESCE(CAST(@i AS nvarchar(10)), N'NULL');

***Listing 17-10.*** Turning Trace Flag 1204 On and Off

DBCC TRACEON (1204, -l);

GO

DBCC TRACEOFF (1204, -l);

GO

***Listing 17-11.*** Basic EXECUTE Statement

EXECUTE (N'SELECT ProductID FROM Production.Product');

***Listing 17-12.*** More Complex Dynamic SQL Example

DECLARE @min\_product\_id int = 500;

DECLARE @sql\_stmt nvarchar(128) =

N'SELECT ProductID ' +

N'FROM Production.Product ' +

N'WHERE ProductID > = ' + CAST(@min\_product\_id AS nvarchar(10));

EXECUTE (@sql\_stmt);

***Listing 17-13.*** Basic Dynamic SQL Query with a String Appended

DECLARE @product\_name nvarchar(50) = N'Mountain';

DECLARE @sql\_stmt NVARCHAR(128) = N'SELECT ProductID, Name ' +

N'FROM Production.Product ' +

N'WHERE Name LIKE ''' +

@product\_name + N'%''';

EXECUTE (@sql\_stmt);

***Listing 17-14.*** Simple T-SQL String Validation Function

CREATE FUNCTION dbo.ValidateString (@string nvarchar(4000))

RETURNS int

AS

BEGIN

DECLARE @result int = 0;

WITH Numbers (Num)

AS

(

SELECT 1

UNION ALL

SELECT Num + 1

FROM Numbers

WHERE Num <= LEN(@string)

)

SELECT @result = SUM

(

CASE

WHEN SUBSTRING(@string, n.Num, 1) LIKE N'[A-Z0-9\\_]' ESCAPE '\'

THEN 0

ELSE 1

END

)

FROM Numbers n

WHERE n.Num <= LEN(@string)

OPTION (MAXRECURSION 0);

RETURN @result;

END

GO

***Listing 17-15.*** *Dynamic SQL sp\_executesql Parameterized*

DECLARE @product\_name NVARCHAR(50) = N'Mountain%';

DECLARE @sql\_stmt NVARCHAR(128) = N'SELECT ProductID, Name ' +

N'FROM Production.Product ' +

N'WHERE Name LIKE @name';

EXECUTE sp\_executesql @sql\_stmt,

N'@name NVARCHAR(50)',

@name = @product\_name;

***Listing 17-16.*** Limited Scope of Dynamic SQL

DECLARE @sql\_stmt NVARCHAR(512) = N'CREATE TABLE #Temp\_ProductIDs ' +

N'(' +

N' ProductID int NOT NULL PRIMARY KEY' +

N'); ' +

N'INSERT INTO #Temp\_ProductIDs (ProductID) ' +

N'SELECT ProductID ' +

N'FROM Production.Product;' ;

EXECUTE (@sql\_stmt);

SELECT ProductID

FROM #Temp\_ProductIDs;

***Listing 17-17.*** Creating a Temp Table Accessible to Dynamic SQL

CREATE TABLE #Temp\_ProductIDs

(

ProductID int NOT NULL PRIMARY KEY

);

DECLARE @sql\_stmt NVARCHAR(512) = N'INSERT INTO #Temp\_ProductIDs (ProductID) ' +

N'SELECT ProductID ' +

N'FROM Production.Product;' ;

EXECUTE (@sql\_stmt);

SELECT ProductID

FROM #Temp\_ProductIDs;