**Conestoga**

14

**Results**

**ASQL Final Project**

**Nick W, Constantine G, Jim R**

Table of Contents

[Transformation Requirements 3](#_Toc405990264)

[Overall 3](#_Toc405990265)

[i. Fahrenheit to Celsius 3](#_Toc405990266)

[ii. Inches to Millimeters 3](#_Toc405990267)

[iii. Parse YearMonth 3](#_Toc405990268)

[Data Visualization 4](#_Toc405990269)

[Libraries 4](#_Toc405990270)

[General Approach 4](#_Toc405990271)

[Series - PCP, CDD/HDD, TAVG/TMIN/TMAX 4](#_Toc405990272)

[Region 4](#_Toc405990273)

[Summary Data - Yearly, Quarterly, Monthly 4](#_Toc405990274)

[X/Y Range 4](#_Toc405990275)

[Time Slider 5](#_Toc405990276)

[Printing 5](#_Toc405990277)

[Schema Diagram 6](#_Toc405990278)

[Schema with a Single User 6](#_Toc405990279)

[Appendix A - SQL Scripts 7](#_Toc405990280)

# Transformation Requirements

## Overall

The ETL was done within the asp.net application, which was then bulk uploaded into the database.

## Fahrenheit to Celsius

In order to convert the Fahrenheit to Celsius, we use a single line within the application.

return ((fah - 32) \* 5.0M / 9.0M);

## Inches to Millimeters

In order to convert the Inches to Millimeters, we use a single line within the application.

return inches / 0.039370M;

## Parse YearMonth

In order to parse the month and the year from the same string, I use a try parse for each on a substring of the input.

bool returnVal = true;

if(!Int16.TryParse(input.Substring(0, 4), out year))

{

returnVal = false;

}

if(!byte.TryParse(input.Substring(4, 2), out month))

{

returnVal = false;

}

return returnVal;

# Data Visualization

## Libraries

* Google.Visualization
* Google Charts and Controls
* Google Dashboard
* ASP.NET
* MSSQL

## General Approach

### Series - PCP, CDD/HDD, TAVG/TMIN/TMAX

In order to plot a series you need to pass the stored procedure a reference to which graph you want to put on the screen. Depending on which type of graph is being requested the amount of data series changes. There are 3 different types of charts displayed below.

|  |  |  |
| --- | --- | --- |
| Chart Type | Variable being displayed | Description |
| Precipitation | 1 Series  PCP(y), Date(x) | The precipitation chart displays Rainfall In Millimeters. |
| Heating and Cooling  Statistics | 2 Series  CDD(y), Date(x) - Cooling  HDD(y), Date(x) - Heating | The heating and cooling rates in different regions. |
| Temperature | 3 Series  TAVG(Y), Date(x) – the AVG temp  TMIN(Y), Date(x) – the MIN temp  TMAX(Y), Date(x) – the MAX temp | The temperature statistics graphed out with average min and max temperature for the period requested |

### Region

In order to plot a series for a specified region, you must select a region from a dropdown menu of possible regions you can view, this is passed in along with the series requested.

Regions are represented by their respective region codes in the dropdown window.

### Summary Data - Yearly, Quarterly, Monthly

In order to plot summary data, you must select the corresponding radio button. This is passed in along with the series and region requested.

For the different compounding periods grouping is done differently as well as the individual ticks on the graph

### X/Y Range

Google. Visualization provides a chart that dynamically sets the X and Y ranges to reasonable numbers.

### Time Slider

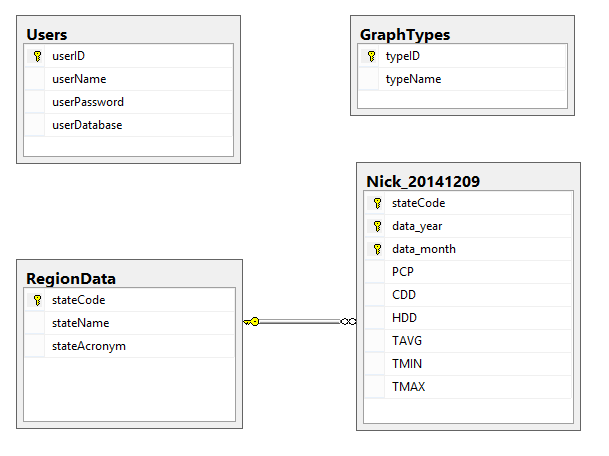
The time slider is provided by Google using Google.Visualization.DateRangeFilter. It allows the user to select a start and an end date. The start and end ranges are set using the two ends of the slider.

### Printing

To print, we used a PDF writer and java scripts ability to print div s on an html page. The application uses document. Print from the JavaScript DOM mapped to a button handler provide a simple easy way of doing this.

# Schema Diagram

## Schema with a Single User



# Appendix A - SQL Scripts

## dbSetup.sql

-- #############################################

-- Create custom dataTypes

-- #############################################

CREATE TYPE yearType FROM smallint NOT NULL;

CREATE TYPE monthType FROM tinyint NOT NULL;

CREATE TYPE uInt FROM int NOT NULL;

CREATE TYPE uFloat FROM float NOT NULL;

-- #############################################

-- Create and populate tables with

-- required values

-- #############################################

-- =============================================

-- RegionData table

-- =============================================

----If the table already exists delete it.--

IF OBJECT\_ID('dbo.RegionData') IS NOT NULL

DROP TABLE dbo.RegionData

GO

----Create the table--

CREATE TABLE dbo.RegionData

(

stateCode int NOT NULL,

stateName nVarChar(100) NOT NULL,

stateAcronym nVarChar(4) NOT NULL,

CONSTRAINT PK\_RegionData PRIMARY KEY (stateCode)

);

GO

INSERT INTO dbo.RegionData VALUES

(1,'Alabama','AL'),

(2,'Arizona','AZ'),

(3,'Arkansas','AR'),

(4,'California','CA'),

(5,'Colorado','CO'),

(6,'Connecticut','CT'),

(7,'Delaware','DE'),

(8,'Florida','FL'),

(9,'Georgia','GA'),

(10,'Idaho','ID'),

(11,'Illinois','IL'),

(12,'Indiana','IN'),

(13,'Iowa','IA'),

(14,'Kansas','KS'),

(15,'Kentucky','KY'),

(16,'Louisiana','LA'),

(17,'Maine','ME'),

(18,'Maryland','MD'),

(19,'Massachusetts','MA'),

(20,'Michigan','MI'),

(21,'Minnesota','MN'),

(22,'Mississippi','MS'),

(23,'Missouri','MO'),

(24,'Montana','MT'),

(25,'Nebraska','NE'),

(26,'Nevada','NV'),

(27,'New Hampshire','NH'),

(28,'New Jersey','NJ'),

(29,'New Mexico','NM'),

(30,'New York','NY'),

(31,'North Carolina','NC'),

(32,'North Dakota','ND'),

(33,'Ohio','OH'),

(34,'Oklahoma','OK'),

(35,'Oregon','OR'),

(36,'Pennsylvania','PA'),

(37,'Rhode Island','RI'),

(38,'South Carolina','SC'),

(39,'South Dakota','SD'),

(40,'Tennessee','TN'),

(41,'Texas','TX'),

(42,'Utah','UT'),

(43,'Vermont','VT'),

(44,'Virginia','VA'),

(45,'Washington','WA'),

(46,'West Virginia','WV'),

(47,'Wisconsin','WI'),

(48,'Wyoming','WY'),

(101,'Northeast Region','NER'),

(102,'East North Central Region','ENCR'),

(103,'Central Region','CR'),

(104,'Southeast Region','SER'),

(105,'West North Central Region','WNCR'),

(106,'South Region','SR'),

(107,'Southwest Region','SWR'),

(108,'Northwest Region','NWR'),

(109,'West Region','WR'),

(110,'National','US')

-- =============================================

-- Users table

-- =============================================

----If the table already exists delete it.--

IF OBJECT\_ID('dbo.Users') IS NOT NULL

DROP TABLE dbo.Users

GO

----Create the table--

CREATE TABLE dbo.Users

(

userID int NOT NULL IDENTITY(1,1),

userName nVarChar(50) NOT NULL UNIQUE,

userPassword nVarChar(25),

userDatabase nVarChar(50),

CONSTRAINT PK\_Users PRIMARY KEY (userID)

);

GO

-- =============================================

-- GraphTypes Table

-- =============================================

----If the table already exists delete it.--

IF OBJECT\_ID('dbo.GraphTypes') IS NOT NULL

DROP TABLE dbo.GraphTypes

GO

----Create the table--

CREATE TABLE dbo.GraphTypes

(

typeID int NOT NULL IDENTITY(1,1),

typeName nVarChar(50) NOT NULL UNIQUE,

CONSTRAINT PK\_GraphTypes PRIMARY KEY (typeID)

);

GO

----Populate the table--

INSERT INTO dbo.GraphTypes VALUES

('Precipitation'),

('Cooling Days/Heating Days'),

('Temperature (Min/Max/Avg)');

GO

## NicksNutHouse.sql

-- ===============================================

-- Create stored procedure with OUTPUT parameters

-- ===============================================

-- Drop stored procedure if it already exists

--USE ASQLGroup

-- CREATE TABLE

IF OBJECT\_ID('CreateTable', 'P') IS NOT NULL

DROP PROCEDURE CreateTable;

GO

CREATE PROCEDURE CreateTable

@userName nVarChar(50),

@userPassword nVarChar(25),

@overWrite bit = 0,

@successful bit OUTPUT

WITH EXECUTE AS Owner

AS

-- table naming username\_YYYYMMDD

DECLARE @TableName nvarchar(50);

SET @successful = 1

-- check if user has table

SELECT @TableName = [userDatabase]

FROM [ASQLGroup].[dbo].[Users]

WHERE [ASQLGroup].[dbo].[Users].[userName] = @userName AND

[ASQLGroup].[dbo].[Users].[userPassword] = @userPassword;

IF @TableName IS NOT NULL

BEGIN

-- table exists

-- if overwrite

-- drop table

-- else

-- select 0

IF @overWrite = 1

BEGIN

DECLARE @SQL VARCHAR(max) =

'DROP TABLE ASQLGroup.dbo.' + @TableName

EXEC(@SQL);

END

ELSE

BEGIN

set @successful = 0

END

END

if @successful = 1

BEGIN

-- generate table name

DECLARE @NewTableName nvarchar(50);

SELECT @NewTableName = @userName + '\_' + CONVERT(varchar,GETDATE(),112);

-- generate table

DECLARE @tableCreate VARCHAR(max) =

'CREATE TABLE ' + @NewTableName +

'(

stateCode int NOT NULL,

data\_year yearType NOT NULL,

data\_month monthType NOT NULL,

PCP decimal,

CDD int,

HDD int,

TAVG money,

TMIN money,

TMAX money

CONSTRAINT FK\_' + @NewTableName + '\_stateCode FOREIGN KEY(stateCode)

REFERENCES RegionData(stateCode),

CONSTRAINT PK\_' + @NewTableName + ' PRIMARY KEY

(stateCode,data\_year,data\_month)

)'

EXEC(@tableCreate)

-- assign table name

UPDATE [ASQLGroup].[dbo].[Users]

SET [ASQLGroup].[dbo].[Users].[userDatabase] = @NewTableName

WHERE [ASQLGroup].[dbo].[Users].[userName] = @userName

AND [ASQLGroup].[dbo].[Users].[userPassword] = @userPassword

SET @successful = 1

END

GO

-- TABLE EXISTS

IF OBJECT\_ID('TableExists', 'P') IS NOT NULL

DROP PROCEDURE TableExists;

GO

CREATE PROCEDURE TableExists

@userName nVarChar(50),

@userPassword nVarChar(25),

@dataCount int OUTPUT

AS

DECLARE @TableName nvarchar(50);

SET @dataCount = 0;

-- check if user has table

SELECT @TableName = [userDatabase]

FROM [ASQLGroup].[dbo].[Users]

WHERE [ASQLGroup].[dbo].[Users].[userName] = @userName AND

[ASQLGroup].[dbo].[Users].[userPassword] = @userPassword

IF @TableName IS NULL

BEGIN

SET @dataCount = -1

END

IF @dataCount = 0

BEGIN

DECLARE @Count int

DECLARE @sqlCommand nvarchar(1000)

DECLARE @params nvarchar(1000)

SET @sqlCommand =

'SELECT @cnt = COUNT(\*) FROM' + quotename(@TableName)

SET @params =

N'@cnt int OUTPUT'

EXECUTE sp\_executesql @sqlCommand, @params, @Count OUTPUT

IF @Count = 0

BEGIN

SET @dataCount = 0

END

ELSE

BEGIN

SET @dataCount = @Count

END

END

GO

-- CREATE USER

IF OBJECT\_ID('CreateUser', 'P') IS NOT NULL

DROP PROCEDURE CreateUser;

GO

CREATE PROCEDURE CreateUser

@userName nVarChar(50),

@rowCount int output

AS

INSERT INTO Users(userName,userPassword)

VALUES (@userName, 'Incorrect')

SET @rowCount = @@ROWCOUNT

GO

-- UPDATE USER

IF OBJECT\_ID('UpdateUser', 'P') IS NOT NULL

DROP PROCEDURE UpdateUser;

GO

CREATE PROCEDURE UpdateUser

@userName nVarChar(50),

@oldPass varchar(25),

@newPass varchar(25),

@rowCount int output

AS

UPDATE Users

SET userPassword = @newPass

WHERE userName = @userName

AND userPassword = @oldPass

SET @rowCount = @@ROWCOUNT

GO

-- DROP USER

IF OBJECT\_ID('DropUser', 'P') IS NOT NULL

DROP PROCEDURE DropUser;

GO

CREATE PROCEDURE DropUser

@userName nVarChar(50),

@userPass nVarChar(25),

@rowCount int output

AS

DECLARE @TableName varChar(50)

SELECT @TableName = [userDatabase]

FROM [ASQLGroup].[dbo].[Users]

WHERE [ASQLGroup].[dbo].[Users].[userName] = @userName AND

[ASQLGroup].[dbo].[Users].[userPassword] = @userPass

IF @TableName IS NOT NULL

BEGIN

DECLARE @sql varchar(1000)

SET @sql = 'DROP TABLE ' + @TableName

EXEC(@sql)

END

DELETE FROM Users

WHERE userName = @userName

AND userPassword = @userPass

SET @rowCount = @@ROWCOUNT

GO

## JimsJungle.sql

IF OBJECT\_ID('verifyUser', 'P') IS NOT NULL

DROP PROCEDURE verifyUser;

GO

CREATE PROCEDURE verifyUser

@userName nVarchar(50),

@password nVarChar(50)

AS

IF(SELECT COUNT(\*) FROM Users

WHERE userName = @userName

AND userPassword = @password) = 0

BEGIN

RETURN(1)

END

RETURN(0)

GO

IF OBJECT\_ID('getDBName', 'P') IS NOT NULL

DROP PROCEDURE getDBName;

GO

CREATE PROCEDURE getDBName

@userName nVarchar(50),

@password nVarChar(50),

@userTable nVarChar(100) output

AS

DECLARE @retCode int;

EXECUTE @retCode = verifyUser @username, @password;

IF @retCode = 0

BEGIN

SELECT @userTable = userDatabase

FROM Users

WHERE userName = @userName;

RETURN 0

END

ELSE

BEGIN

RETURN (1)

END

GO

IF OBJECT\_ID('getMaxMin', 'P') IS NOT NULL

DROP PROCEDURE getMaxMin;

GO

Create Procedure getMaxMin

@userName nVarchar(50),

@password nVarChar(50),

@minDate date OUTPUT,

@maxDate date OUTPUT

AS

DECLARE @retCode int

Declare @tableName varChar(50)

EXEC @retCode = dbo.getDBName

@userName = @userName,

@password = @password,

@userTable = @tableName OUTPUT;

IF @retCode = 0

BEGIN

DECLARE @sql nvarchar(4000),@params nvarchar(4000)

SELECT @sql =

N'SELECT @minDate = MIN(DATEADD(YEAR, data\_year-1900, DATEADD(MM, data\_month, -1))), ' +

'@maxDate = MAX(DATEADD(YEAR, data\_year-1900, DATEADD(MM, data\_month, -1))) '+

'FROM ' + @tableName

SELECT @params =

N'@minDate date OUTPUT, ' +

N'@maxDate date OUTPUT'

EXEC sp\_executesql @sql, @params,

@minDate = @minDate OUTPUT,

@maxDate = @maxDate OUTPUT

RETURN 0

END

ELSE

BEGIN

RETURN (1)

END

GO

IF OBJECT\_ID('getRegions', 'P') IS NOT NULL

DROP PROCEDURE getRegions;

GO

CREATE PROCEDURE getRegions

@userName nVarchar(50),

@password nVarChar(50)

AS

DECLARE @retCode int

Declare @tableName varChar(50)

EXEC @retCode = dbo.getDBName

@userName = @userName,

@password = @password,

@userTable = @tableName OUTPUT;

DECLARE @sql nVarChar(max) = 'SELECT DISTINCT stateCode ' +

'FROM [ASQLGroup].[dbo].[' + @tableName + ']';

EXEC (@sql)

GO

## GenericSearchProc.sql

IF OBJECT\_ID('GenericSearchYearly', 'P') IS NOT NULL

DROP PROCEDURE GenericSearchYearly;

GO

CREATE PROCEDURE GenericSearchYearly(

@stateCodeIn int,

@dbName nVarChar(50),

@ReturnColumns nVarChar(100)

)

AS

DECLARE @bigSQL varChar(max) =

'SELECT data\_year as [Year]'

+ @returnColumns +

'FROM ' + @dbName + '

WHERE stateCode = ' + CONVERT(varChar(10),@stateCodeIn) +

' GROUP BY

data\_year

ORDER BY

[Year]

;'

EXEC(@bigSQL)

GO

IF OBJECT\_ID('GenericSearchQuarterly', 'P') IS NOT NULL

DROP PROCEDURE GenericSearchQuarterly;

GO

CREATE PROCEDURE GenericSearchQuarterly(

@stateCodeIn int,

@dbName nVarChar(50),

@ReturnColumns nVarChar(100)

)

AS

DECLARE @bigSQL varChar(max) =

'SELECT data\_year as [Year],

CASE

WHEN data\_month IN (1, 2, 3) THEN 1

WHEN data\_month IN (4, 5, 6) THEN 2

WHEN data\_month IN (7, 8, 9) THEN 3

ELSE 4

END As [Quarter]'

+ @returnColumns +

'FROM ' + @dbName + '

WHERE stateCode = ' + CONVERT(varChar(10),@stateCodeIn) +

' GROUP BY

data\_year,

CASE

WHEN data\_month IN (1, 2, 3) THEN 1

WHEN data\_month IN (4, 5, 6) THEN 2

WHEN data\_month IN (7, 8, 9) THEN 3

ELSE 4

END

ORDER BY

[Year],

[Quarter]

;'

EXEC(@bigSQL)

GO

IF OBJECT\_ID('GenericSearchMonthly', 'P') IS NOT NULL

DROP PROCEDURE GenericSearchMonthly;

GO

CREATE PROCEDURE GenericSearchMonthly(

@stateCodeIn int,

@dbName nVarChar(50),

@ReturnColumns nVarChar(100)

)

AS

DECLARE @bigSQL varChar(max) =

'SELECT data\_year as [Year],

data\_month as [Month]'

+ @returnColumns +

'FROM ' + @dbName + '

WHERE stateCode = ' + CONVERT(varChar(10),@stateCodeIn) +

' GROUP BY

data\_year,

data\_month

ORDER BY

[Year],

[Month]

;'

EXEC(@bigSQL)

GO

IF OBJECT\_ID('GenericSearch', 'P') IS NOT NULL

DROP PROCEDURE GenericSearch;

GO

CREATE PROCEDURE GenericSearch(

@userName nVarChar(50),

@password nVarChar(50),

@stateCodeIn int,

@searchType int,

@timeIncrement int

)

AS

--Declare @userName nVarChar(50) = N'Demo'

--Declare @password nVarChar(50) = N'Password'

--Declare @stateCodeIn int = 101

--Declare @timeIncrement int = 2

-- Declare @rangeStart Date = DATEADD(YEAR, 1953 - 1900, DATEADD(MM, 01, -1));

--Declare @rangeEnd Date = DATEADD(YEAR, 1960 - 1900, DATEADD(MM, 04, -1))

DECLARE @return\_value int,

@userTable nVarChar(50)

EXEC @return\_value = getDBName

@userName = @userName,

@password = @password,

@userTable = @userTable OUTPUT

IF @return\_value = 0

BEGIN

Declare @ReturnColumns nVarChar(100) =

CASE @searchType

WHEN 2 THEN ', AVG(CDD) AS CDD, AVG(HDD) AS HDD '

WHEN 3 THEN ', MAX(TMAX) AS TMAX, MIN(TMIN) AS TMIN, AVG(TAVG) AS TAVG '

ELSE ', AVG(PCP) AS PCP '

END

IF @timeIncrement = 1

BEGIN

EXEC GenericSearchMonthly

@stateCodeIn = @stateCodeIn,

@dbName = @userTable,

@ReturnColumns = @ReturnColumns

END

ELSE

BEGIN

IF @timeIncrement = 2

BEGIN

EXEC GenericSearchQuarterly

@stateCodeIn = @stateCodeIn,

@dbName = @userTable,

@ReturnColumns = @ReturnColumns

END

ELSE

BEGIN

IF @timeIncrement = 3

BEGIN

EXEC GenericSearchYearly

@stateCodeIn = @stateCodeIn,

@dbName = @userTable,

@ReturnColumns = @ReturnColumns

END

ELSE

BEGIN

Return(1)

END

END

END

END

ELSE

BEGIN

RETURN(1)

END

RETURN(0)

GO