'This is the code for the User Form

Option Explicit 'makes variable declaration & definition required - doing this can reduce human error

Private Sub UserForm\_Initialize**()**

' --- Populate controls for the first chart ---

' Populate the state selection list box with all state tabs and "national"

With lstState

**.**AddItem "arizona"

**.**AddItem "arkansas"

**.**AddItem "colorado"

**.**AddItem "connecticut"

**.**AddItem "delaware"

**.**AddItem "districtOfColumbia"

**.**AddItem "illinois"

**.**AddItem "indiana"

**.**AddItem "iowa"

**.**AddItem "kansas"

**.**AddItem "kentucky"

**.**AddItem "louisiana"

**.**AddItem "maine"

**.**AddItem "maryland"

**.**AddItem "massachusetts"

**.**AddItem "michigan"

**.**AddItem "mississippi"

**.**AddItem "montana"

**.**AddItem "nevada"

**.**AddItem "newHampshire"

**.**AddItem "newJersey"

**.**AddItem "newYork"

**.**AddItem "northCarolina"

**.**AddItem "ohio"

**.**AddItem "oregon"

**.**AddItem "pennsylvania"

**.**AddItem "rhodeIsland"

**.**AddItem "southDakota"

**.**AddItem "tennessee"

**.**AddItem "vermont"

**.**AddItem "virginia"

**.**AddItem "westVirginia"

**.**AddItem "wyoming"

**.**AddItem "national"

End With

' Populate the measure selection list box

With lstMeasure

**.**AddItem "Handle"

**.**AddItem "Revenue"

**.**AddItem "Taxes"

End With

' Set default selections for the first chart

lstState.ListIndex **=** **0**

lstMeasure.ListIndex **=** **0**

' --- Populate controls for the second chart (bubble chart) ---

' Populate lstState2 with only state names (no "national")

With lstState2

**.**Clear

**.**AddItem "arizona"

**.**AddItem "arkansas"

**.**AddItem "colorado"

**.**AddItem "connecticut"

**.**AddItem "delaware"

**.**AddItem "districtOfColumbia"

**.**AddItem "illinois"

**.**AddItem "indiana"

**.**AddItem "iowa"

**.**AddItem "kansas"

**.**AddItem "kentucky"

**.**AddItem "louisiana"

**.**AddItem "maine"

**.**AddItem "maryland"

**.**AddItem "massachusetts"

**.**AddItem "michigan"

**.**AddItem "mississippi"

**.**AddItem "montana"

**.**AddItem "nevada"

**.**AddItem "newHampshire"

**.**AddItem "newJersey"

**.**AddItem "newYork"

**.**AddItem "northCarolina"

**.**AddItem "ohio"

**.**AddItem "oregon"

**.**AddItem "pennsylvania"

**.**AddItem "rhodeIsland"

**.**AddItem "southDakota"

**.**AddItem "tennessee"

**.**AddItem "vermont"

**.**AddItem "virginia"

**.**AddItem "westVirginia"

**.**AddItem "wyoming"

**.**MultiSelect **=** fmMultiSelectMulti

**.**ListIndex **=** **0**

End With

End Sub

Private Sub btnCreateChart\_Click**()** 'this sub defines what happens when the first "Go" button is clicked

Dim i As Long

Dim selStates As New Collection

Dim selMeasures As New Collection

' Gather selected states from lstState

For i **=** **0** To lstState.ListCount **-** **1**

If lstState.Selected**(**i**)** Then

selStates.Add lstState.List**(**i**)**

End If

Next i

' Gather selected measures from lstMeasure

For i **=** **0** To lstMeasure.ListCount **-** **1**

If lstMeasure.Selected**(**i**)** Then

selMeasures.Add lstMeasure.List**(**i**)**

End If

Next i

'this message will appear if the use clicks Go without selecting at least one state and one measure

If selStates.Count **=** **0** Or selMeasures.Count **=** **0** Then

MsgBox "Please select at least one state and one measure."**,** vbExclamation

Exit Sub

End If

' Call the helper procedure to create a multi-series line chart

CreateLineChartMulti selStates**,** selMeasures

End Sub

Private Sub btnCreateChart2\_Click**()** ' Triggered when the second "Go" button is clicked

Dim selStates As New Collection ' Collection to store selected states

Dim i As Long

' Loop through the states in lstState2 list box

For i **=** **0** To lstState2.ListCount **-** **1**

If lstState2.Selected**(**i**)** Then ' Check if the state is selected

selStates.Add lstState2.List**(**i**)** ' Add selected state to collection

End If

Next i

' If no states are selected, show a message and exit

If selStates.Count **=** **0** Then

MsgBox "Please select at least one state."**,** vbExclamation

Exit Sub

End If

' Call the module procedure to create the correlation line chart.

CreateCorrelationLineChart selStates

End Sub

'clicking the last "Go" button simply calls the CreateBubbleChartMap procedure

Private Sub btnCreateBubbleChart\_Click**()**

CreateBubbleChartMap

End Sub

'This will close the User Form so the User can see their results

Private Sub CloseUserForm\_Click**()**

Unload Me

End Sub

'Module 1 - This module is tied to the Step 2 button to import state-specific gambling data only

Option Explicit 'must declare all variables and types

Sub ImportCSVFiles\_States**()**

'all the Dim statements are declaring and defining our variables

Dim FolderPath As String

Dim FileName As String

Dim wbCSV As Workbook

Dim wsDest As Worksheet

Dim sheetName As String

'FolderPath is where it will pull the csv files from - you can edit this to work on your pc

FolderPath **=** "C:\Users\ctole\OneDrive\Desktop\ProjData\States\"

FileName **=** Dir**(**FolderPath **&** "\*.csv"**)**

'now we start a loop to make sure all the csv files are imported

' Loop to process all files in the folder (FileName holds the file name, FolderPath is the folder location)

Do While FileName **<>** ""

' Open the current CSV file

Set wbCSV **=** Workbooks.Open**(**FileName**:=**FolderPath **&** FileName**)**

' Extract the sheet name from the CSV file name (remove the file extension)

sheetName **=** Left**(**FileName**,** InStrRev**(**FileName**,** "."**)** **-** **1)**

' Create a new worksheet in the current workbook after the last existing worksheet

Set wsDest **=** ThisWorkbook.Worksheets.Add**(**After**:=**ThisWorkbook.Worksheets**(**ThisWorkbook.Worksheets.Count**))**

' Rename the newly created worksheet to match the CSV file name (without extension)

wsDest.Name **=** sheetName

' Copy the used range of the first sheet in the CSV file and paste it into the new worksheet

wbCSV.Sheets**(1).**UsedRange.Copy Destination**:=**wsDest.Range**(**"A1"**)**

' Close the CSV workbook without saving changes

wbCSV.Close SaveChanges**:=**False

' Get the next file name in the folder (Dir() will return the next file, or "" if no more files)

FileName **=** Dir**()**

Loop

End Sub

'Module 2 - This module is tied to the Step 1 button to import supporting data csv files from the ProjData folder.

'This is the same folder as the application/workbook is located.

'These include # of teams per state, census data, lat & long of each state, etc.

Option Explicit 'must declare all variables and types

Sub ImportCSVFiles\_ProjData**()**

'all the Dim statements are declaring and defining our variables

Dim FolderPath As String

Dim FileName As String

Dim wbCSV As Workbook

Dim wsDest As Worksheet

Dim sheetName As String

'FolderPath is where it will pull the csv files from - you can edit this to work on your pc

FolderPath **=** "C:\Users\ctole\OneDrive\Desktop\ProjData\"

FileName **=** Dir**(**FolderPath **&** "\*.csv"**)**

'now we start a loop to make sure all the csv files are imported

' Loop through all the files in the folder (FileName holds the current file name, FolderPath is the folder location)

Do While FileName **<>** ""

' Open the current file (CSV file) located at FolderPath with the current FileName

Set wbCSV **=** Workbooks.Open**(**FileName**:=**FolderPath **&** FileName**)**

' Extract the sheet name from the CSV file name by removing the file extension (everything after the last period)

sheetName **=** Left**(**FileName**,** InStrRev**(**FileName**,** "."**)** **-** **1)**

' Add a new worksheet to the current workbook, placing it after the last existing worksheet

Set wsDest **=** ThisWorkbook.Worksheets.Add**(**After**:=**ThisWorkbook.Worksheets**(**ThisWorkbook.Worksheets.Count**))**

' Rename the newly created worksheet to the sheet name extracted from the CSV file

wsDest.Name **=** sheetName

' Copy the entire used range from the first sheet in the opened CSV file and paste it into the new worksheet starting at cell A1

wbCSV.Sheets**(1).**UsedRange.Copy Destination**:=**wsDest.Range**(**"A1"**)**

' Close the CSV file without saving any changes (as it's no longer needed)

wbCSV.Close SaveChanges**:=**False

' Get the next file name in the folder (Dir() returns the next file name or "" if no more files are found)

FileName **=** Dir**()**

Loop

End Sub

'Module 3

'The purpose of this code is to include state date as it gets updated over time.

'It is tied to the button that says it will refresh the state data.

Option Explicit 'must declare all variables and types

Sub RefreshCSVFiles\_States**()**

'all the Dim statements are declaring and defining our variables

Dim FolderPath As String

Dim FileName As String

Dim sheetName As String

Dim wbCSV As Workbook

Dim validStates As Object

Dim missingList As String

Dim key As Variant

'FolderPath is where it will pull the csv files from - you can edit this to work on your pc

FolderPath **=** "C:\Users\ctole\OneDrive\Desktop\ProjData\States\"

Set validStates **=** CreateObject**(**"Scripting.Dictionary"**)**

' Build dictionary of CSV file names (without the extension) from the States folder.

FileName **=** Dir**(**FolderPath **&** "\*.csv"**)**

Do While FileName **<>** ""

sheetName **=** Left**(**FileName**,** InStrRev**(**FileName**,** "."**)** **-** **1)**

validStates**(**sheetName**)** **=** FolderPath **&** FileName

FileName **=** Dir**()**

Loop

missingList **=** ""

' For each expected state worksheet (based on the CSV file names stored in validStates dictionary)

For Each key In validStates.Keys

Dim ws As Worksheet

On Error Resume Next ' Temporarily ignore errors to check if the worksheet exists

Set ws **=** ThisWorkbook.Worksheets**(**key**)** ' Try to set the worksheet object for the state

On Error GoTo **0** ' Re-enable normal error handling

' If the worksheet does not exist (ws is Nothing), add it to the missing list

If ws Is Nothing Then

missingList **=** missingList **&** key **&** vbCrLf ' Append missing state to the list

Else

' If the worksheet exists, refresh its data by clearing old content and loading new data

ws.Cells.Clear ' Clear any existing content from the worksheet

Set wbCSV **=** Workbooks.Open**(**FileName**:=**validStates**(**key**))** ' Open the corresponding CSV file based on the state

wbCSV.Sheets**(1).**UsedRange.Copy Destination**:=**ws.Range**(**"A1"**)** ' Copy the CSV data to the worksheet starting from cell A1

wbCSV.Close SaveChanges**:=**False ' Close the CSV file without saving changes

End If

Next key

' After processing all states, check if there were any missing data sources

If missingList **<>** "" Then

' Notify the user with a message box listing the states that could not be found

MsgBox "The following States data sources were not found:" **&** vbCrLf **&** missingList**,** vbExclamation

End If

End Sub

' Module 4 is connected to the User Form. It defines the CreateLineChartMulti procedure that the User Form calls

' in the second frame. This procedure is responsible for creating a multi-series line chart based on selected states and measures.

Option Explicit

Public Sub CreateLineChartMulti**(**selStates As Collection**,** selMeasures As Collection**)**

' Declare variables for worksheets, counters, and other necessary parameters.

Dim wsChartData As Worksheet

Dim wsData As Worksheet

Dim i As Long**,** j As Long

Dim lastRow As Long**,** dataRows As Long

Dim outCol As Long

Dim sheetName As String

' Use the first selected state for the X-axis data (Month values) by setting the sheetName from the first selected state.

sheetName **=** selStates.Item**(1)**

On Error Resume Next ' Ignore errors temporarily while checking for the sheet's existence

Set wsData **=** ThisWorkbook.Worksheets**(**CStr**(**sheetName**))** ' Attempt to set the worksheet corresponding to the selected state

On Error GoTo **0** ' Re-enable normal error handling

' If the worksheet does not exist, display an error message and exit the procedure

If wsData Is Nothing Then

MsgBox "Data sheet '" **&** sheetName **&** "' not found."**,** vbExclamation

Exit Sub

End If

' Determine the last used row in Column A of the selected state sheet to find the range of data.

lastRow **=** wsData.Cells**(**wsData.Rows.Count**,** "A"**).**End**(**xlUp**).**Row

' Create (or clear) a worksheet named "ChartOutput" for assembling chart data.

On Error Resume Next ' Again, ignore errors while checking for the worksheet

Set wsChartData **=** ThisWorkbook.Worksheets**(**"ChartOutput"**)** ' Check if the "ChartOutput" sheet exists

On Error GoTo **0** ' Re-enable normal error handling

' If "ChartOutput" sheet doesn't exist, create a new one and name it "ChartOutput"

If wsChartData Is Nothing Then

Set wsChartData **=** ThisWorkbook.Worksheets.Add

wsChartData.Name **=** "ChartOutput"

Else

' If the sheet already exists, clear any old content to prepare for new data.

wsChartData.Cells.Clear

End If

' Write the X-axis header ("Month") in cell A1 of the ChartOutput sheet.

wsChartData.Range**(**"A1"**).**Value **=** "Month"

' Instead of copying directly, recalculate each Month cell by adjusting the date format and extracting year/month.

Dim r As Long

For r **=** **2** To lastRow

Dim cellText As String**,** newYear As Long**,** monthName As String**,** monthNum As Long**,** newDate As Date

cellText **=** wsData.Cells**(**r**,** "A"**).**Text ' Extract the date string from the current cell (e.g., "18-Jun")

' If the cell text has a valid format (at least 5 characters), extract and process the date

If Len**(**cellText**)** **>=** **5** Then

' Extract the two-digit year, convert it to a full year (e.g., "18" -> 2018)

newYear **=** **2000** **+** CLng**(**Left**(**cellText**,** **2))**

' Extract the month abbreviation from the date string

monthName **=** Mid**(**cellText**,** **4)**

' Convert the month abbreviation to its corresponding month number

Select Case LCase**(**monthName**)**

Case "jan"**:** monthNum **=** **1**

Case "feb"**:** monthNum **=** **2**

Case "mar"**:** monthNum **=** **3**

Case "apr"**:** monthNum **=** **4**

Case "may"**:** monthNum **=** **5**

Case "jun"**:** monthNum **=** **6**

Case "jul"**:** monthNum **=** **7**

Case "aug"**:** monthNum **=** **8**

Case "sep"**:** monthNum **=** **9**

Case "oct"**:** monthNum **=** **10**

Case "nov"**:** monthNum **=** **11**

Case "dec"**:** monthNum **=** **12**

Case Else**:** monthNum **=** **1** ' Default to January if the month name is invalid

End Select

' Create a full date using the extracted year and month, setting the day to the 1st

newDate **=** DateSerial**(**newYear**,** monthNum**,** **1)**

wsChartData.Cells**(**r**,** "A"**).**Value **=** newDate

' Format the cell to show only the year (for the X-axis)

wsChartData.Cells**(**r**,** "A"**).**NumberFormat **=** "yyyy"

Else

' If the cell doesn't match the expected format, copy it as is.

wsChartData.Cells**(**r**,** "A"**).**Value **=** wsData.Cells**(**r**,** "A"**).**Value

End If

Next r

' Set up the output column for the series data. We'll start placing data from column B onward.

outCol **=** **2**

' Loop through each selected state and measure to add each series to the chart data.

Dim measureColNum As Long

For i **=** **1** To selStates.Count

For j **=** **1** To selMeasures.Count

Dim currState As String**,** currMeasure As String

currState **=** selStates.Item**(**i**)** ' Get the current state

currMeasure **=** selMeasures.Item**(**j**)** ' Get the current measure

' Get the data worksheet for the current state.

On Error Resume Next ' Temporarily ignore errors while accessing the worksheet

Set wsData **=** ThisWorkbook.Worksheets**(**currState**)** ' Try to get the state's worksheet

On Error GoTo **0** ' Restore normal error handling

' If the worksheet isn't found, skip to the next measure and continue the loop.

If wsData Is Nothing Then

GoTo NextMeasure

End If

' Determine the column number corresponding to the measure.

' Assume: Column B = Handle, Column C = Revenue, Column E = Taxes.

Select Case currMeasure

Case "Handle"**:** measureColNum **=** **2**

Case "Revenue"**:** measureColNum **=** **3**

Case "Taxes"**:** measureColNum **=** **5**

Case Else**:** measureColNum **=** **2** ' Default to Column B for unrecognized measures

End Select

' Copy the measure data (from rows 2 to lastRow) into the ChartOutput sheet.

wsData.Range**(**wsData.Cells**(2,** measureColNum**),** wsData.Cells**(**lastRow**,** measureColNum**)).**Copy \_

Destination**:=**wsChartData.Cells**(2,** outCol**)**

' Set the header for this series (combining the state and measure).

wsChartData.Cells**(1,** outCol**).**Value **=** currState **&** " - " **&** currMeasure

' Move to the next output column for the next series.

outCol **=** outCol **+** **1**

NextMeasure**:** ' Label to skip to the next measure if necessary

Next j

Next i

' Determine the new last row for the data (should match the count of Month rows).

dataRows **=** wsChartData.Cells**(**wsChartData.Rows.Count**,** "A"**).**End**(**xlUp**).**Row

' Define the range for the chart data (from A1 to the last used column).

Dim rngChart As Range

Set rngChart **=** wsChartData.Range**(**wsChartData.Cells**(1,** **1),** wsChartData.Cells**(**dataRows**,** outCol **-** **1))**

' Create a new worksheet for the chart, with a unique and meaningful name based on the current time.

Dim wsChartSheet As Worksheet

Dim chObj As ChartObject

Dim chartSheetName As String

chartSheetName **=** "Chart\_" **&** Format**(**Now**,** "hhmmss"**)** ' Generate a unique chart sheet name using current time

Set wsChartSheet **=** ThisWorkbook.Worksheets.Add ' Add a new worksheet for the chart

wsChartSheet.Name **=** chartSheetName ' Set the name of the new chart worksheet

' Add a chart object to the new worksheet, specifying its position and size.

Set chObj **=** wsChartSheet.ChartObjects.Add**(**Left**:=50,** Top**:=10,** Width**:=600,** Height**:=400)**

' Configure the chart with the data range, chart type, and title.

With chObj.Chart

**.**SetSourceData Source**:=**rngChart**,** PlotBy**:=**xlColumns ' Set the source data for the chart and configure for column-based plotting

**.**ChartType **=** xlLine ' Set the chart type to Line

**.**HasTitle **=** True ' Enable the chart title

**.**ChartTitle.Text **=** "Multi-Series Chart" ' Set the chart title

' Force the Category (X) axis to be a time scale with yearly ticks.

With **.**Axes**(**xlCategory**)**

**.**CategoryType **=** xlTimeScale ' Set the X-axis to represent time (years)

**.**TickLabels.NumberFormat **=** "yyyy" ' Display the year format on the X-axis

**.**MajorUnitScale **=** xlYears ' Set the major unit to represent one year

**.**MajorUnit **=** **1** ' Set the major unit to 1 year

**.**HasTitle **=** True ' Enable title for the X-axis

**.**AxisTitle.Text **=** "Year" ' Set the X-axis title to "Year"

End With

' Configure the Y-axis with a title.

With **.**Axes**(**xlValue**)**

**.**HasTitle **=** True ' Enable title for the Y-axis

**.**AxisTitle.Text **=** "USD" ' Set the Y-axis title to "USD"

End With

End With

' Notify the user that the chart has been created with a message box, including the name of the new chart sheet.

MsgBox "Chart created on sheet '" **&** chartSheetName **&** "'."**,** vbInformation

End Sub

'Module 5 is tied to the step 3. Open User Form button on the MASTER worksheet.

' This subroutine opens and shows the Chart Builder user form.

Sub ShowChartBuilder**()**

' Display the Chart Builder form.

frmChartBuilder.Show

End Sub

'Module 6 creates the helper procedure which will make the second chart option of the User Form

Option Explicit

Public Sub CreateCorrelationLineChart**(**selStates As Collection**)**

' Declare variables for worksheets, rows, columns, and other data

Dim wsTemp As Worksheet**,** wsPop As Worksheet**,** wsChart As Worksheet

Dim outRow As Long**,** i As Long**,** rState As Long**,** yr As Long

Dim stateName As String**,** properState As String

Dim lastRowState As Long

Dim totalRev As Double**,** monthCount As Long**,** rev As Double**,** popVal As Double

Dim rngData As Range**,** chObj As ChartObject

Dim chartSheetName As String

' Create or clear temporary summary sheet "CorrelationData"

On Error Resume Next

Set wsTemp **=** ThisWorkbook.Worksheets**(**"CorrelationData"**)**

On Error GoTo **0**

If wsTemp Is Nothing Then

Set wsTemp **=** ThisWorkbook.Worksheets.Add

wsTemp.Name **=** "CorrelationData"

Else

wsTemp.Cells.Clear

End If

' Write headers in the first row for the summary data: State, Year, Population, Revenue

With wsTemp

**.**Range**(**"A1"**).**Value **=** "State"

**.**Range**(**"B1"**).**Value **=** "Year"

**.**Range**(**"C1"**).**Value **=** "Population"

**.**Range**(**"D1"**).**Value **=** "Revenue"

End With

outRow **=** **2** ' Start inserting data from row 2

' Set reference to the population census data sheet (popCensus).

Dim wsPopData As Worksheet

Set wsPopData **=** ThisWorkbook.Worksheets**(**"popCensus"**)**

' Loop through each selected state from the list.

Dim wsState As Worksheet

For i **=** **1** To selStates.Count

stateName **=** selStates**(**i**)** ' Get state name in lowercase (e.g., "arizona")

properState **=** WorksheetFunction.Proper**(**stateName**)** ' Capitalize state name properly (e.g., "Arizona")

' Get the state's worksheet (assumed to be named in lowercase).

On Error Resume Next

Set wsState **=** ThisWorkbook.Worksheets**(**stateName**)**

On Error GoTo **0**

If wsState Is Nothing Then GoTo NextState ' Skip if state sheet does not exist

' Determine the last row with data in the state's sheet.

lastRowState **=** wsState.Cells**(**wsState.Rows.Count**,** "A"**).**End**(**xlUp**).**Row

' Loop through the years 2020 to 2024 for each state's data.

For yr **=** **2020** To **2024**

totalRev **=** **0**

monthCount **=** **0**

' Loop through each row of the state's worksheet to calculate total revenue for the current year.

For rState **=** **2** To lastRowState

' Get the displayed date from Column A to extract the year.

Dim dispDate As String**,** displayYear As Long

dispDate **=** wsState.Cells**(**rState**,** "A"**).**Text ' e.g., "18-Nov"

displayYear **=** **2000** **+** Val**(**Left**(**dispDate**,** **2))** ' Extract year (e.g., "18" -> 2018)

' If the year matches the selected year, add the revenue for that month.

If displayYear **=** yr Then

If IsNumeric**(**wsState.Cells**(**rState**,** "C"**).**Value**)** Then

totalRev **=** totalRev **+** wsState.Cells**(**rState**,** "C"**).**Value

monthCount **=** monthCount **+** **1**

End If

End If

Next rState

' Skip to next year if no data was found for the current year.

If monthCount **=** **0** Then GoTo NextYear

rev **=** totalRev ' Set total revenue for the year (average could also be used)

' Lookup population for this state and year from the popCensus sheet.

Dim popRow As Variant

popRow **=** Application.Match**(**properState**,** wsPopData.Range**(**"E:E"**),** **0)**

If IsError**(**popRow**)** Then

popVal **=** **0** ' If population data is not found, set to 0.

Else

' Lookup population based on the year (columns G-K).

Select Case yr

Case **2020:** popVal **=** wsPopData.Cells**(**popRow**,** "G"**).**Value

Case **2021:** popVal **=** wsPopData.Cells**(**popRow**,** "H"**).**Value

Case **2022:** popVal **=** wsPopData.Cells**(**popRow**,** "I"**).**Value

Case **2023:** popVal **=** wsPopData.Cells**(**popRow**,** "J"**).**Value

Case **2024:** popVal **=** wsPopData.Cells**(**popRow**,** "K"**).**Value

End Select

End If

' Write the data for this year into the temporary summary sheet.

With wsTemp

**.**Cells**(**outRow**,** "A"**).**Value **=** properState

**.**Cells**(**outRow**,** "B"**).**Value **=** yr

**.**Cells**(**outRow**,** "C"**).**Value **=** popVal

**.**Cells**(**outRow**,** "D"**).**Value **=** rev

End With

outRow **=** outRow **+** **1** ' Move to the next row for the next data entry

NextYear**:**

Next yr

NextState**:**

Set wsState **=** Nothing ' Reset the worksheet variable before moving to the next state

Next i

' If no data was found, notify the user and exit the procedure.

If outRow **=** **2** Then

MsgBox "No data found for the selected states."**,** vbExclamation

Exit Sub

End If

' Define the data range for the chart (from A1 to last row of data).

Set rngData **=** wsTemp.Range**(**"A1:D" **&** outRow **-** **1)**

' Create or clear a worksheet for the chart named "CorrelationLineChart".

Dim wsChartExists As Boolean

On Error Resume Next

wsChartExists **=** Not ThisWorkbook.Worksheets**(**"CorrelationLineChart"**)** Is Nothing

On Error GoTo **0**

If wsChartExists Then

Set wsChart **=** ThisWorkbook.Worksheets**(**"CorrelationLineChart"**)**

wsChart.Cells.Clear ' Clear existing chart data

Else

Set wsChart **=** ThisWorkbook.Worksheets.Add

wsChart.Name **=** "CorrelationLineChart"

End If

' Create a line chart and set its properties.

Set chObj **=** wsChart.ChartObjects.Add**(**Left**:=50,** Top**:=10,** Width**:=800,** Height**:=400)**

Dim myChart As Chart

Set myChart **=** chObj.Chart

myChart.ChartType **=** xlLine

myChart.HasTitle **=** True

myChart.ChartTitle.Text **=** "Revenue vs Population Over Time"

myChart.HasLegend **=** True

' Remove any default series from the chart.

Do While myChart.SeriesCollection.Count **>** **0**

myChart.SeriesCollection**(1).**Delete

Loop

' Build a unique list of states from the summary data for charting.

Dim dictStates As Object

Set dictStates **=** CreateObject**(**"Scripting.Dictionary"**)**

Dim rTemp As Long

For rTemp **=** **2** To wsTemp.Cells**(**wsTemp.Rows.Count**,** "A"**).**End**(**xlUp**).**Row

Dim sName As String

sName **=** wsTemp.Cells**(**rTemp**,** "A"**).**Value

If Not dictStates.Exists**(**sName**)** Then

dictStates.Add sName**,** sName ' Add unique state to the dictionary

End If

Next rTemp

' For each state, add two series: one for Population (secondary axis) and one for Revenue.

Dim stKey As Variant

For Each stKey In dictStates.Keys

Dim arrYear**()** As Double**,** arrPop**()** As Double**,** arrRev**()** As Double

Dim cnt As Long**,** idx As Long

cnt **=** **0**

' Count how many years of data exist for the current state.

For rTemp **=** **2** To wsTemp.Cells**(**wsTemp.Rows.Count**,** "A"**).**End**(**xlUp**).**Row

If wsTemp.Cells**(**rTemp**,** "A"**).**Value **=** stKey Then cnt **=** cnt **+** **1**

Next rTemp

If cnt **=** **0** Then GoTo NextUnique

' Initialize arrays to store data for the current state.

ReDim arrYear**(1** To cnt**)**

ReDim arrPop**(1** To cnt**)**

ReDim arrRev**(1** To cnt**)**

idx **=** **1**

' Loop through the rows and populate the arrays with data for this state.

For rTemp **=** **2** To wsTemp.Cells**(**wsTemp.Rows.Count**,** "A"**).**End**(**xlUp**).**Row

If wsTemp.Cells**(**rTemp**,** "A"**).**Value **=** stKey Then

arrYear**(**idx**)** **=** wsTemp.Cells**(**rTemp**,** "B"**).**Value

arrPop**(**idx**)** **=** wsTemp.Cells**(**rTemp**,** "C"**).**Value

arrRev**(**idx**)** **=** wsTemp.Cells**(**rTemp**,** "D"**).**Value

idx **=** idx **+** **1**

End If

Next rTemp

' Add Population series to the chart (on the secondary Y-axis).

With myChart.SeriesCollection.NewSeries

**.**Name **=** stKey **&** " Population"

**.**XValues **=** arrYear

**.**Values **=** arrPop

**.**AxisGroup **=** xlSecondary

End With

' Add Revenue series to the chart (on the primary Y-axis).

With myChart.SeriesCollection.NewSeries

**.**Name **=** stKey **&** " Revenue"

**.**XValues **=** arrYear

**.**Values **=** arrRev

End With

NextUnique**:**

Next stKey

' Notify the user that the chart has been created successfully.

MsgBox "Correlation line chart created on sheet 'CorrelationLineChart'."**,** vbInformation

End Sub

'Module 7 is the most complex code we developed. We did some research and wanted to create a geographical

'map of the gambling data in some way, but that type of chart was not compatible.

'As a workaround, we imported the latitude and longitude of the approximate center of each state with legalized

'sport gambling. Then we used the bubbles to show the larger values of amount spent gambling per person in the state.

'To make the visualization more compelling, we used a simple formula to tie the R,G,B values to the size

'of each bubble. So dark and larger go together to make a more compelling graphic.

Option Explicit

Public Sub CreateBubbleChartMap**()**

Dim wsSummary As Worksheet**,** wsPop As Worksheet**,** wsCoords As Worksheet**,** stateSheet As Worksheet

Dim lastRow As Long**,** i As Long

Dim outRow As Long

Dim stName As String

Dim avgPop As Double**,** avgHandle As Double**,** hpc As Double

Dim totalHandle As Double**,** countMonths As Long

Dim rngData As Range

Dim chObj As ChartObject

Dim wsChart As Worksheet**,** chartSheetName As String

' Create (or clear) the summary worksheet "BubbleMapData".

On Error Resume Next

Set wsSummary **=** ThisWorkbook.Worksheets**(**"BubbleMapData"**)**

On Error GoTo **0**

If wsSummary Is Nothing Then

Set wsSummary **=** ThisWorkbook.Worksheets.Add

wsSummary.Name **=** "BubbleMapData"

Else

wsSummary.Cells.Clear

End If

' Write headers: State, HandlePerCapita, Latitude, Longitude.

With wsSummary

**.**Range**(**"A1"**).**Value **=** "State"

**.**Range**(**"B1"**).**Value **=** "HandlePerCapita"

**.**Range**(**"C1"**).**Value **=** "Latitude"

**.**Range**(**"D1"**).**Value **=** "Longitude"

End With

outRow **=** **2**

' Set references:

' popCensus: Column E (NAME) holds state names; Columns G:K hold population estimates.

Set wsPop **=** ThisWorkbook.Worksheets**(**"popCensus"**)**

' US\_States\_Coordinates: Column A = State; Column B = Latitude; Column C = Longitude.

Set wsCoords **=** ThisWorkbook.Worksheets**(**"US\_States\_Coordinates"**)**

' Define the list of states.

Dim stateList As Variant

stateList **=** Array**(**"Arizona"**,** "Arkansas"**,** "Colorado"**,** "Connecticut"**,** "Delaware"**,** \_

"District of Columbia"**,** "Illinois"**,** "Indiana"**,** "Iowa"**,** "Kansas"**,** \_

"Kentucky"**,** "Louisiana"**,** "Maine"**,** "Maryland"**,** "Massachusetts"**,** \_

"Michigan"**,** "Mississippi"**,** "Montana"**,** "Nevada"**,** "New Hampshire"**,** \_

"New Jersey"**,** "New York"**,** "North Carolina"**,** "Ohio"**,** "Oregon"**,** \_

"Pennsylvania"**,** "Rhode Island"**,** "South Dakota"**,** "Tennessee"**,** \_

"Vermont"**,** "Virginia"**,** "West Virginia"**,** "Wyoming"**)**

' Loop through each state.

For i **=** LBound**(**stateList**)** To UBound**(**stateList**)**

stName **=** stateList**(**i**)**

' Get the state's worksheet (assumes state worksheets are named in lowercase, e.g., "arizona").

On Error Resume Next

Set stateSheet **=** ThisWorkbook.Worksheets**(**LCase**(**stName**))**

On Error GoTo **0**

If stateSheet Is Nothing Then GoTo NextState

' Determine last used row in Column A of the state's sheet.

lastRow **=** stateSheet.Cells**(**stateSheet.Rows.Count**,** "A"**).**End**(**xlUp**).**Row

' Calculate average monthly Handle from Column B.

totalHandle **=** **0**

countMonths **=** **0**

Dim rState As Long

For rState **=** **2** To lastRow

If IsNumeric**(**stateSheet.Cells**(**rState**,** "B"**).**Value**)** Then

totalHandle **=** totalHandle **+** stateSheet.Cells**(**rState**,** "B"**).**Value

countMonths **=** countMonths **+** **1**

End If

Next rState

If countMonths **>** **0** Then

avgHandle **=** totalHandle **/** countMonths

Else

avgHandle **=** **0**

End If

' Look up average population from popCensus.

Dim popRow As Variant

popRow **=** Application.Match**(**stName**,** wsPop.Range**(**"E:E"**),** **0)**

If IsError**(**popRow**)** Then

avgPop **=** **0**

Else

avgPop **=** **(**wsPop.Cells**(**popRow**,** "G"**).**Value **+** \_

wsPop.Cells**(**popRow**,** "H"**).**Value **+** \_

wsPop.Cells**(**popRow**,** "I"**).**Value **+** \_

wsPop.Cells**(**popRow**,** "J"**).**Value **+** \_

wsPop.Cells**(**popRow**,** "K"**).**Value**)** **/** **5**

End If

' Calculate Handle per Capita.

If avgPop **>** **0** Then

hpc **=** avgHandle **/** avgPop

Else

hpc **=** **0**

End If

' Write summary row.

With wsSummary

**.**Cells**(**outRow**,** "A"**).**Value **=** stName

**.**Cells**(**outRow**,** "B"**).**Value **=** hpc

**.**Cells**(**outRow**,** "B"**).**NumberFormat **=** "0.000"

End With

' Look up coordinates from US\_States\_Coordinates.

Dim coordRow As Variant**,** latVal As Double**,** longVal As Double

coordRow **=** Application.Match**(**stName**,** wsCoords.Range**(**"A:A"**),** **0)**

If IsError**(**coordRow**)** Then

latVal **=** **0:** longVal **=** **0**

Else

latVal **=** wsCoords.Cells**(**coordRow**,** "B"**).**Value

longVal **=** wsCoords.Cells**(**coordRow**,** "C"**).**Value

End If

With wsSummary

**.**Cells**(**outRow**,** "C"**).**Value **=** latVal

**.**Cells**(**outRow**,** "D"**).**Value **=** longVal

End With

outRow **=** outRow **+** **1**

NextState**:**

Set stateSheet **=** Nothing

Next i

' Define the summary data range.

Set rngData **=** wsSummary.Range**(**"A1:D" **&** outRow **-** **1)**

' Create or clear a fixed worksheet named "BubbleMap".

Dim wsChartExists As Boolean

On Error Resume Next

wsChartExists **=** Not ThisWorkbook.Worksheets**(**"BubbleMap"**)** Is Nothing

On Error GoTo **0**

If wsChartExists Then

Set wsChart **=** ThisWorkbook.Worksheets**(**"BubbleMap"**)**

wsChart.Cells.Clear

Else

Set wsChart **=** ThisWorkbook.Worksheets.Add

wsChart.Name **=** "BubbleMap"

End If

' Add a bubble chart on wsChart.

Set chObj **=** wsChart.ChartObjects.Add**(**Left**:=50,** Top**:=10,** Width**:=600,** Height**:=400)**

With chObj.Chart

**.**ChartType **=** xlBubble

**.**HasTitle **=** True

**.**ChartTitle.Text **=** "Sports Gambling Per Person by State" **&** vbCrLf **&** "Bubble size & color = Handle per capita"

**.**HasLegend **=** False

' Remove any default series.

Do While **.**SeriesCollection.Count **>** **0**

**.**SeriesCollection**(1).**Delete

Loop

Dim srs As Series

Set srs **=** **.**SeriesCollection.NewSeries

With srs

' X values: Longitude (Column D)

**.**XValues **=** wsSummary.Range**(**"D2:D" **&** outRow **-** **1)**

' Y values: Latitude (Column C)

**.**Values **=** wsSummary.Range**(**"C2:C" **&** outRow **-** **1)**

' Bubble sizes: Handle per capita (Column B)

**.**BubbleSizes **=** wsSummary.Range**(**"B2:B" **&** outRow **-** **1)**

**.**HasDataLabels **=** True

Dim k As Long

For k **=** **1** To **.**DataLabels.Count

With **.**DataLabels**(**k**)**

**.**Text **=** wsSummary.Cells**(**k **+** **1,** "A"**).**Value

**.**Position **=** xlLabelPositionCenter

End With

Next k

End With

End With

' Adjust axis ranges and remove tick labels and gridlines.

With chObj.Chart.Axes**(**xlCategory**)**

**.**MinimumScale **=** **-140**

**.**MaximumScale **=** **-60**

**.**TickLabelPosition **=** xlTickLabelPositionNone

**.**HasMajorGridlines **=** False

**.**HasMinorGridlines **=** False

End With

With chObj.Chart.Axes**(**xlValue**)**

**.**MinimumScale **=** **25**

**.**MaximumScale **=** **50**

**.**TickLabelPosition **=** xlTickLabelPositionNone

**.**HasMajorGridlines **=** False

**.**HasMinorGridlines **=** False

End With

' Adjust bubble fill color based on bubble size (heat map).

' Bubble colors are assigned based on the Handle per Capita value.

Dim bubbleArray As Variant

bubbleArray **=** wsSummary.Range**(**"B2:B" **&** outRow **-** **1).**Value

Dim iPoint As Long**,** currentSize As Double**,** intensity As Long

Dim maxBubble As Double**,** minBubble As Double

' Get the maximum and minimum bubble sizes from the data.

maxBubble **=** Application.WorksheetFunction.Max**(**wsSummary.Range**(**"B2:B" **&** outRow **-** **1))**

minBubble **=** Application.WorksheetFunction.Min**(**wsSummary.Range**(**"B2:B" **&** outRow **-** **1))**

' Loop through each point to set the color intensity based on the bubble size.

' The loop ensures that each bubble gets a color from light gray (low value) to dark gray (high value).

For iPoint **=** **1** To UBound**(**bubbleArray**,** **1)**

currentSize **=** bubbleArray**(**iPoint**,** **1)**

If maxBubble **<>** minBubble Then

' Map so that the smallest bubble gets intensity 200 (light gray) and largest gets 128 (dark gray)

intensity **=** **200** **-** **((**currentSize **-** minBubble**)** **/** **(**maxBubble **-** minBubble**))** **\*** **(200** **-** **75)**

Else

intensity **=** **200** ' If all values are the same, use a default intensity (light gray).

End If

' Ensure the intensity is within valid RGB range.

If intensity **<** **0** Then intensity **=** **0**

If intensity **>** **255** Then intensity **=** **255**

' Set the bubble color to the calculated intensity (gray scale).

srs.Points**(**iPoint**).**Format.Fill.ForeColor.RGB **=** RGB**(**intensity**,** intensity**,** intensity**)**

Next iPoint

' Set chart title with subtitle in smaller font.

Dim mainTitle As String**,** subTitle As String

mainTitle **=** "Sports Gambling Per Person by State"

subTitle **=** "Bubble size & color = Handle per capita"

With chObj.Chart.ChartTitle

**.**Text **=** mainTitle **&** vbCrLf **&** subTitle

**.**Characters**(**Start**:=1,** Length**:=**Len**(**mainTitle**)).**Font.Size **=** **22**

**.**Characters**(**Start**:=**Len**(**mainTitle**)** **+** **3,** Length**:=**Len**(**subTitle**)).**Font.Size **=** **14**

End With

MsgBox "Bubble chart created on sheet 'BubbleMap'."**,** vbInformation

End Sub

'Module 8: This macro is assigned to the button called "Clear all tabs" on the MASTER worksheet.

'The purpose of this macro is to loop through all the worksheets in the active workbook and delete each one

'except for the "MASTER" worksheet. This macro effectively clears all non-MASTER worksheets from the workbook.

Sub DeleteNonMasterTabs**()**

Dim ws As Worksheet ' Declare a variable 'ws' to represent each worksheet in the workbook.

Application.DisplayAlerts **=** False ' Disable Excel's alert messages to avoid confirmation prompts when deleting sheets.

' Loop through each worksheet in the active workbook.

For Each ws In ActiveWorkbook.Worksheets

' Check if the worksheet is not the "MASTER" worksheet.

If ws.Name **<>** "MASTER" Then

ws.Delete ' If it's not the MASTER sheet, delete the worksheet.

End If

Next ws ' Move to the next worksheet in the workbook.

Application.DisplayAlerts **=** True ' Re-enable Excel's alert messages after deleting the sheets.

End Sub