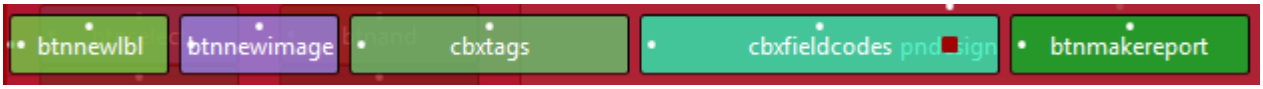
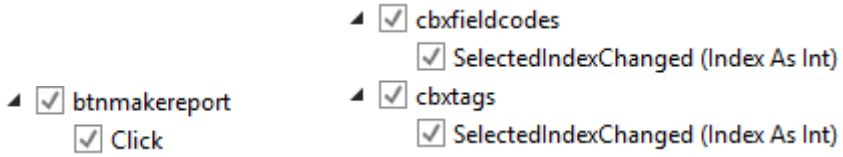
Step 5 : add report tags, field codes and make reports

Changes in the layout:



- Add the B4XComboBox views to the pndesign pane: set the **cbxtags** B4XComboBox width to 140 and the height to 30, set the **cbxfieldcodes** B4XComboBox width to 180 and the height to 30.
- Add a button btnmakereport next to the 2 B4XComboBox views. The width of the button is 120 and the height is 30.
- Generate the members and check the methods:



Changes in the B4XMainPage page:

- In the Class_Globals subroutine you should have the variables for the views and you can add the variables for the B4XComboBox lists:

```
Public btnmakereport As Button
Public cbxtags As B4XComboBox
Public tagslst As List
Public cbxfieldcodes As B4XComboBox
Public fieldcodeslst As List
```

- In the B4XPage_Created subroutine you can add code to fill the B4XComboBox views. Initialize the lists, add an array of selections, set the items of the B4XComboBox and set the selected index to 0.

```
tagslst.Initialize
tagslst.AddAll(Array As String("Select a tag", "FTR=footer", "HDR=header",
"RST=resultset", "TBL=table"))
cbxtags.SetItems(tagslst)
cbxtags.SelectedIndex = 0
fieldcodeslst.Initialize
fieldcodeslst.AddAll(Array As String("Select a fieldcode", "DATE", "NUMPAGES",
"PAGE"))
cbxfieldcodes.SetItems(fieldcodeslst)
cbxfieldcodes.SelectedIndex = 0
```

- Add code to the cbxtags_SelectedIndexChanged subroutine. The first selection acts as a prompt and is not used. The part before the = sign is tested and the tags are added to the panemain pane with the add_tags_to_grid subroutine. When the SQL textarea is filled with a sql statement then the get_fieldlabels_from_sql subroutine generates a tag for each field after the SQL SELECT keyword.

```
Private Sub cbxtags_SelectedIndexChanged (Index As Int)
    If cbxtags.SelectedIndex = 0 Then Return
    Dim seltag As String = cbxtags.SelectedItem
    Log(seltag)
```

```
	Dim tag As String = seltag.SubString2(0,seltag.IndexOf("="))
	Select tag
		Case "FTR"		' footer [FTR][/FTR]
			add_tags_to_grid("FTR","FTR")
		Case "HDR"		' header [HDR][/HDR]
			add_tags_to_grid("HDR","HDR")
		Case "RST"		' resultset [RST][/RST]
			add_tags_to_grid("RST","RST")
			If tasql.IsInitialized And tasql.Text <> "" And tasql.Text <> "SELECT "
Then
				get_fieldlabels_from_sql(tasql.text)
			End If
		Case "TBL"		' table [TBL][/TBL]
			add_tags_to_grid("TBL","TBL")
	End Select
	cbxtags.SelectedIndex = 0
End Sub
```

- Add code to the cbxfieldcodes_SelectedIndexChanged subroutine. The fieldcode label gets the default label properties and the tagdata properties of the label are changed. The label has a labeltype “FCO”.

```
Private Sub cbxfieldcodes_SelectedIndexChanged (Index As Int)
	If cbxfieldcodes.SelectedIndex = 0 Then Return
	Dim seltag As String = cbxfieldcodes.SelectedItem
	Log(seltag)
	Dim lbl As Label =
set_default_label_properties(seltag,fx.Colors.Black,fx.Colors.ARGB(255,255,200,200))
	Dim td As tagdata = set_tagdata_properties(lbl,"FCO")
	td.row = 0
	td.col = 0
	td.text = seltag
	lbl.Tag = td
	panemain.AddNode(lbl,0,0,lbl.Width,lbl.Height)
	cbxfieldcodes.SelectedIndex = 0
End Sub
```

- Add the subroutine add_tags_to_grid. The default label properties are set and the tagdata are changed. The tags come in pairs of two, one opening tag [...] and one closing tag [/...]. The resultset opening tag gets some extra database information in the set_tagdata_properties subroutine. The position of the opening tag in the grid is row 0 and column 0. The position of the closing tag in the grid is row 2 and column 0. Move the tags to the appropriate place in the grid.

```
Private Sub add_tags_to_grid(text As String, tag As String)
	Dim lbl As Label = set_default_label_properties("[" & text &
"]",fx.Colors.Black,fx.Colors.ARGB(255,202,255,187))
	Dim td As tagdata = set_tagdata_properties(lbl,tag)
	td.row = 0
	td.col = 0
	td.text = "[" & text & "]"
```

```
        lbl.Tag = td
        panemain.AddNode(lbl,0,0,lbl.Width,lbl.Height)
        Dim lbl As Label = set_default_label_properties("[/" & text &
"]",fx.Colors.Black,fx.Colors.ARGB(255,202,255,187))
        Dim td As tagdata = set_tagdata_properties(lbl,"/" & tag)
        td.row = 2
        td.col = 0
        td.text = "[/" & text & "]"
        lbl.Tag = td
        panemain.AddNode(lbl,0,40,lbl.Width,lbl.Height)
End Sub
```

- Add the subroutine set_tagdata_properties.

```
Private Sub set_tagdata_properties(lbl As Label, labeltype As String) As tagdata
        Dim td As tagdata
        td.Initialize
        td.labeltype = labeltype
        td.row = lbl.Top/gridsize
        td.col = lbl.Left/gridsize
        Select labeltype
                Case "FTR"          ' footer [FTR][/FTR]
                Case "HDR"          ' header [HDR][/HDR]
                Case "RST"          ' resultset [RST][/RST]
                        td.dbfile = ""
                        td.dbpath = ""
                        td.dbname = ""
                        td.tblname = ""
                        Select currtab
                                Case 0
                                        If tasqlitefile.IsInitialized And tasqlitefile.Text <> "" Then
                                                td.dbfile = tasqlitefile.Text
                                                td.dbpath = tapath.Text
                                                td.tblname = tatable.Text
                                        End If
                                Case 1
                                        If tamysqldb.IsInitialized And tamysqldb.Text <> "" Then
                                                td.dbname = tamysqldb.Text
                                                td.tblname = tamysqltable.text
                                        End If
                        End Select
                        If tasql.IsInitialized And tasql.Text <> "" Then
                                td.strsql = tasql.Text.Replace(",","|")
                        End If
                Case "TBL"          ' table [TBL][/TBL]
        End Select
        Return td
End Sub
```

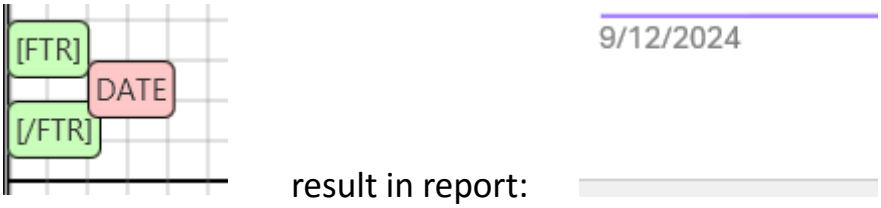
- Add code to generate the field labels from the SQL statement.

```
Private Sub get_fieldlabels_from_sql(strsql As String)
```

```
        Dim sqlparts As List = Regex.Split(" ",strsql)
        Log(sqlparts.Get(1))
        Dim fldnames As List = Regex.Split(",",sqlparts.Get(1))
        Log(fldnames)
        Dim tblname As String = sqlparts.Get(3)
        Dim rowcnt As Int = 1
        For i = 0 To fldnames.Size -1
            Dim lbl As Label =
set_default_label_properties(fldnames.Get(i),fx.Colors.Blue,fx.Colors.ARGB(255,216,238,2
55))
            Dim td As tagdata = set_tagdata_properties(lbl,"FLD")
            td.row = rowcnt
            td.col = 4
            td.tblname = tblname
            td.fldname = fldnames.Get(i)
            td.text = fldnames.Get(i)
            lbl.Text = fldnames.Get(i)
            lbl.Tag = td
            panemain.AddNode(lbl,80,i*40,lbl.prefWidth,lbl.prefHeight)
            rowcnt = rowcnt + 2
        Next
End Sub
```

You can run the application now and test adding the report tags and the fieldcode tags. The report tags appear in row 0 and 2 and column 0. The fieldcode tag appears in row 0 and column 0.

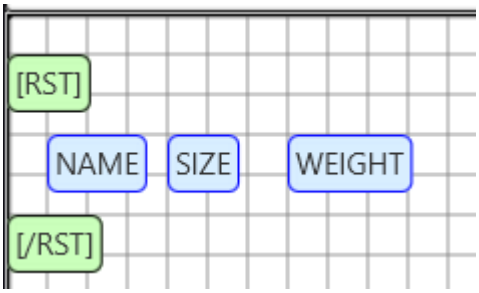
Examples:



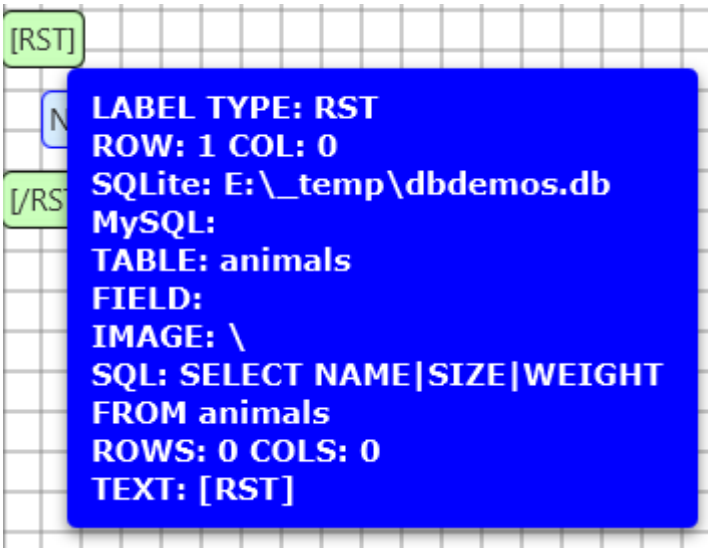
result in report:

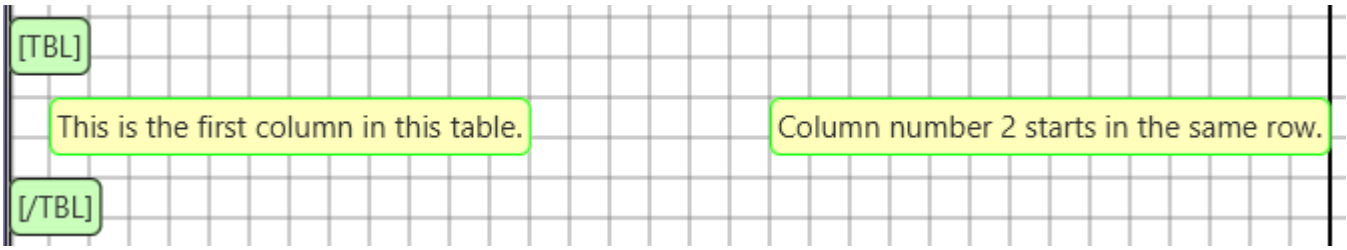
result in report:

Example of a header 1 of 1



NAME	SIZE	WEIGHT
Angel Fish	2	2
Boa	10	8
Critters	30	20





result in report:

This is the first column in this table.		Column number 2 starts in the same row.

So now it's time to write some code to make a report like the results you see above.

- Create a region and name it report.
  #Region report
- Move the btnmakereport_Click subroutine into the region report and call the make_report subroutine. The Try Catch structure is commented out for testing purposes.

```
Private Sub btnmakereport_Click
'       Try
                make_report
'       Catch
'               Log(LastException)
'               xui.MsgboxAsync("There was an error in making the report","Make report")
'       End Try
End Sub
```

- The report is assembled by using the labels from the grid. The grid is scanned from left to right and from top to bottom.
- The gridmap will contain the label tagdata.
- The gridlist will be the sequence of the labels.
- The make_report subroutine will create a Word document with the content of the page. The document is saved in a file and then opened in the Word application.
- The Word document can be converted to a PDF file if you remove the comment.
- The Word tables will have no borders at this time. For this to work you have to add the following Java code below the B4J code (code from Erel):

```
#if java
import org.apache.poi.xwpf.usermodel.*;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.*;
public static void removeBorders(XWPFTable table) {
    CTTblPr tblpro = table.getCTTbl().getTblPr();
    CTTblBorders borders = tblpro.addNewTblBorders();
    borders.addNewBottom().setVal(STBorder.NONE);
    borders.addNewLeft().setVal(STBorder.NONE);
    borders.addNewRight().setVal(STBorder.NONE);
    borders.addNewTop().setVal(STBorder.NONE);
    //also inner borders
    borders.addNewInsideH().setVal(STBorder.NONE);
    borders.addNewInsideV().setVal(STBorder.NONE);
}
#End If
```

- And here's the make_report subroutine:

```
Private Sub make_report
        fill_gridmap
        If gridmap.Size = 0 Then Return
        fill_gridlist
        Dim docstring As String = compose_page
'       Log(docstring)
        Dim Document As WordDocument = Wutils.CreateDocument
        Document.Append($"
        ${docstring}
        "$)
        ' all Word tables have no borders
        Dim tables As List = Document.XWPFDocument.RunMethod("getTables", Null)
        For Each table As JavaObject In tables
                Me.as(JavaObject).RunMethod("removeBorders", Array(table))
        Next
        ' the result is saved in a Word document called Report.docx
        Dim f As String = Document.SaveAs(xui.DefaultFolder, "Report.docx", True)
'       Wait For (Wutils.PowerShellConvertToPdf(f, File.Combine(xui.DefaultFolder,
"Report.pdf"), True)) Complete (Success As Boolean)
        Wait For (Wutils.OpenWord(f)) Complete (Success As Boolean)
End Sub
```

- Create the variables for the gridmap and gridlist.

```
Public gridmap As Map
```

```
Public gridlist As List
```
- Initialize the variables in the B4XPage_Created subroutine.

```
gridmap.Initialize
gridlist.Initialize
```
- In the fill_gridmap subroutine the GetAllViewsRecursive method from the panemain pane is used to fill the gridmap. The key is the row and column from the tagdata of the label.

```
Private Sub fill_gridmap
    gridmap.Clear
    For Each vw As B4XView In panemain.GetAllViewsRecursive
        If vw Is Label Then
            Dim td As tagdata = vw.Tag
            gridmap.Put(td.row & "|" & td.col, td)
        End If
    Next
End Sub
```
- Two loops are used to go over the rows and columns in the fill_gridlist. For testing purposes only the first 27 rows and 33 columns are used (= 1 Word page with normal margins).
  NOTE: change the pagewidth variable in the draw_grid subroutine from 800 to 660 to show the pagewidth line after 33 columns.

```
Private Sub fill_gridlist
    gridlist.Clear
    For r = 0 To 26
        For c = 0 To 32
            If gridmap.ContainsKey(r & "|" & c) Then
                Dim td As tagdata = gridmap.Get(r & "|" & c)
                gridlist.Add(td)
            End If
        Next
    Next
    Log(gridlist)
End Sub
```
- In the compose_page subroutine a string is build that holds all the paragraphs and tables. Empty paragraphs are inserted above the first label if the row of that label is greater than the line count.
- The labeltype is tested to call the specific write subroutine (write_footer, write_header, write_resultset_table, write_table, write_paragraph)
- Declare the public variables maxtablerows and maxtablecols and initialize them at 1.

```
Public maxtablerows As Int = 1
Public maxtablecols As Int = 1
```
-

```
Private Sub compose_page As String
    Dim sb As StringBuilder
    sb.Initialize
    Dim linecnt As Int = 0
    For i = 0 To gridlist.Size - 1
```

```
        Dim td As tagdata = gridlist.Get(i)
        Log(i & " " & td.labeltype & " " & td.row & " " & td.col)
        If i = 0 Then
            For j = 0 To td.row -1
                sb.Append(write_paragraph(Null))
                linecnt = linecnt + 1
            Next
        Else
            If td.row > linecnt Then
                For k = linecnt To td.row -1
                    sb.Append(write_paragraph(Null))
                Next
                linecnt = td.row
            End If
        End If
        Select td.labeltype
            Case "FTR"
                sb.Append(write_footer(td.row,i))
                i = i + maxtablecols + 1
                Log("i: " & i)
                linecnt = linecnt + maxtablerows
            Case "HDR"
                sb.Append(write_header(td.row,i))
                i = i + maxtablecols + 1
                Log("i: " & i)
                linecnt = linecnt + maxtablerows
            Case "RST"
                sb.Append(write_resultset_table(td.row,i))
                i = i + maxtablecols + 1
                Log("i: " & i)
                linecnt = linecnt + maxtablerows
            Case "TBL"
                sb.Append(write_table(td.row,i))
                i = i + maxtablecols + 1
                Log("i: " & i)
                linecnt = linecnt + maxtablerows
            Case "TXT"
                sb.Append(write_paragraph(td))
                linecnt = linecnt + td.numrows
            Case "IMG"
                sb.Append(write_paragraph(td))
                linecnt = linecnt + td.numrows
            Case Else
                sb.Append(write_paragraph(Null))
        End Select
    Next
    Log("line count: " & linecnt)
    Return sb.ToString
```

```
End Sub
```

- The write_footer subroutine will be called when the labeltype is “FTR” and the closing tag is “/FTR”. The content of the footer will be added to a Word table with 1 row and all the labels between the starting and ending “FTR” tag. The count of the columns is performed in the count_columns subroutine. By default a blue line is placed above the footer table. You can put this line in comment if you don’t want to use it. The set_one_row_table subroutine will create the footer table.

```
Private Sub write_footer(row As Int, gridlistindex As Int) As String
    Dim sb As StringBuilder
    sb.Initialize
    Dim colcnt As Int = 0
    Dim strcolcnt As String = count_columns(gridlistindex, "FTR")
    row = strcolcnt.SubString2(0,strcolcnt.IndexOf("|"))
    Dim colcnt As Int = strcolcnt.SubString(strcolcnt.IndexOf("|")+1)
    maxtablecols = colcnt
    sb.Append($"[footer][p]"$)
    ' default footer line
    sb.Append($"[p][img dir=assets filename="blue_line.png" width=470
height=2/][/p]"$)
    sb.Append(set_one_row_table(row,colcnt,"FTR"))
    sb.Append($"[/p][/footer]"$)
    Return sb.ToString
End Sub
```

- The write_header subroutine: a header one row table is created. The columns are counted to set in the Word table.

```
Private Sub write_header(row As Int, gridlistindex As Int) As String
    Dim sb As StringBuilder
    sb.Initialize
    Dim colcnt As Int = 0
    Dim strcolcnt As String = count_columns(gridlistindex, "HDR")
    row = strcolcnt.SubString2(0,strcolcnt.IndexOf("|"))
    Dim colcnt As Int = strcolcnt.SubString(strcolcnt.IndexOf("|")+1)
    maxtablecols = colcnt
    sb.Append($"[header][p]"$)
    sb.Append(set_one_row_table(row,colcnt,"HDR"))
    sb.Append($"[/p][/header]"$)
    Return sb.ToString
End Sub
```

- The write_table subroutine: a one row table is created. The columns are counted and set in the Word table. This report tag can be used to create a multiple column layout with text labels, images and field codes. The Word columns feature is not implemented (yet) and requires some special Java code. You have to take care of the text continuation in the next column yourself.

```
Private Sub write_table(row As Int, gridlistindex As Int) As String
    Dim sb As StringBuilder
    sb.Initialize
    Dim colcnt As Int = 0
```

```
        Dim strcolcnt As String = count_columns(gridlistindex, "TBL")
        row = strcolcnt.SubString2(0,strcolcnt.IndexOf("|"))
        Dim colcnt As Int = strcolcnt.SubString(strcolcnt.IndexOf("|")+1)
        maxtablecols = colcnt
        sb.Append(set_one_row_table(row,colcnt,"TBL"))
        Return sb.ToString
End Sub
```

- The write_paragraph subroutine: an empty paragraph is added if the parameter is Null. The paragraph will be indented by 14 pixels per grid column.

```
Private Sub write_paragraph(td As tagdata) As String
        Dim sb As StringBuilder
        sb.Initialize
        If td = Null Then
                sb.Append($"[p] [/p]"$)
        Else
                Dim indent As Int = td.col*14
                Select td.labeltype
                        Case "TXT"
                                sb.Append($"[p Indentationleft=${indent}]${td.text}[/p]"$)
                        Case "IMG"
                                sb.Append($"[p Indentationleft=${indent}][img
dir="${td.imgdir}" filename="${td.imgfile}" width=100 height=100 /][/p]"$)
                End Select
        End If
        Return sb.ToString
End Sub
```

- The write_resultset_table: for the resultset table the number of labels are counted between the starting “RST” tag and the ending “/RST” tag. In the set_resultset_table subroutine the table is written to the document string.

```
Private Sub write_resultset_table(row As Int, gridlistindex As Int) As String
        Dim sb As StringBuilder
        sb.Initialize
        Dim colcnt As Int = 0
        Dim strcolcnt As String = count_columns(gridlistindex, "RST")
        row = strcolcnt.SubString2(0,strcolcnt.IndexOf("|"))
        Dim colcnt As Int = strcolcnt.SubString(strcolcnt.IndexOf("|")+1)
        maxtablecols = colcnt
        sb.Append(set_resultset_table(gridlistindex))
        Return sb.ToString
End Sub
```

- The count_columns subroutine uses the index of the starting report tag in the gridlist. The loop goes over all the labels in the gridlist starting with the index of the starting report tag and stopping when the ending report tag is reached.
- Create and initialize the gridfields list.

```
Public gridfields As List
gridfields.Initialize
```

- The gridfields list contains the visible fields in the grid per RST tag. This list will be checked when creating a resultset_table. Only these fields will be used in the Word table.

```
Private Sub count_columns(startindex As Int, labeltype As String) As String
	gridfields.Clear
	Dim colcnt As Int = 0
	Dim actualrow As Int
	For j = startindex To gridlist.size - 1
		Dim td As tagdata = gridlist.Get(j)
		If td.labeltype = labeltype Then rstsqlstring = td.strsql
		If td.labeltype = "/" & labeltype Then
			Exit
		End If
		If td.labeltype <> labeltype Then
			colcnt = colcnt + 1
			gridfields.Add(td.fldname)
			actualrow = td.row
		End If
	Next
	Return actualrow & "|" & colcnt
End Sub
```

- In the set_one_row_table subroutine the 33 columns from the row are scanned. If a label is in the gridmap and it's not one of the report tags then a table cell is added to the document string. A table cell can contain a text label, an image or a fieldcode tag.

```
Private Sub set_one_row_table(row As Int,colcnt As Int,labeltype As String) As String
	Dim sb As StringBuilder
	sb.Initialize
	Dim ccnt As Int = 0
	Dim oldcol As Int
	sb.Append($"[table Alignment=left rows=${1} cols=${colcnt+(colcnt-1)}]"$)
	sb.Append($"[row]"$)
	For c = 0 To 32
		If gridmap.ContainsKey(row & "|" & c) Then
			Dim td As tagdata = gridmap.Get(row & "|" & c)
			If td.labeltype <> labeltype And td.labeltype <> "/" & labeltype Then
				Dim tabstring As String = ""
				Dim tabcnt As Int = (td.col - oldcol) / 3
				For j = 1 To tabcnt + 1
					tabstring = tabstring & $"${TAB}"$
				Next
				If tabcnt = 0 Then tabstring = $" "$
				If ccnt > 0 Then sb.Append($"[cell][p
alignment=left]${tabstring}[/p][/cell]"$)
				ccnt = ccnt + 1
				Select td.labeltype
					Case "TXT"
						sb.Append($"[cell][p
alignment=left]${td.text}[/p][/cell]"$)
```

```
                                Case "IMG"
                                        sb.Append($"[cell width=120][p
alignment=center][img dir="${td.imgdir}" filename="${td.imgfile}" width=100 height=100
/][/p][/cell]"$)
                                Case "FCO"
                                        sb.Append($"[cell][p
alignment=left][FieldCode=${td.text}/][/p][/cell]"$)
                        End Select
                        ccnt = ccnt + 1
                        oldcol = td.col + td.numcols
                        If td.numrows > maxtablerows Then maxtablerows = td.numrows
                    End If
                End If
            Next
            sb.Append($"[/row]"$)
            sb.Append($"[/table]"$)
            Return sb.ToString
End Sub
```

- In the set_sqlstring_from_grid_fields a SQL string is created using the “FLD” labels that are between the “RST” and “/RST” report tags. The tblname from the tagdata of the first “FLD” label is used in the FROM clause of the SQL statement. There can only be one table name!

```
Private Sub set_sqlstring_from_grid_fields(gridlistindex As Int) As String
        Dim sqlstring As String = ""
        Dim fldstring As String
        Dim tblname As String
        For i = gridlistindex To gridlist.Size -1
            Dim td As tagdata = gridlist.Get(i)
            If td.labeltype = "/RST" Then Exit
            If td.labeltype = "FLD" Then
                If tblname <> "" And tblname <> td.tblname Then Return "more than 1
table"
                If tblname = "" Then tblname = td.tblname
                fldstring = fldstring & td.fldname & ","
            End If
        Next
        fldstring = fldstring.SubString2(0,fldstring.Length-1)
        sqlstring = "SELECT " & fldstring & " FROM " & tblname
        Return sqlstring
End Sub
```

- In the set_resultset_table the sqlstring variable can contain the SQL statement from the SQL textarea or the SQL string from the tagdata of the “RST” report tag or the created SQL string in the set_sqlstring_from_grid_fields subroutine.
- Create a public variable rstsqlstring. This variable is initialized in the count_columns subroutine.

```
Public rstsqlstring As String = ""
```

- The record list (reclst) is filled by calling the ExecQuery from the selected database in the current tab.

```
•
Private Sub set_resultset_table(gridlistindex As Int) As String
	Select currtab
		Case 0
			If tasqlitefile.Text = "" Then Return "ERROR"
		Case 1
			If tamysqldb.Text = "" Then Return "ERROR"
	End Select
	Dim sb As StringBuilder
	sb.Initialize
	Dim sqlstring As String = tasql.Text
	If rstsqlstring <> "" Then sqlstring = rstsqlstring.Replace("|",",")		' from RST tag
	If sqlstring <> "" And sqlstring <> "SELECT " Then
		Dim sqllines As List = Regex.Split(CRLF,sqlstring)
		Dim fldstring As String = sqllines.Get(0)
		fldstring = fldstring.SubString(fldstring.IndexOf("SELECT ")+7).Trim
		Dim fldslst As List = Regex.Split(",",fldstring)
	End If
	If sqlstring = "" Or sqlstring = "SELECT " Then
		If maxtablecols > 0 Then
			Log("maxtablecols: " & maxtablecols)
			sqlstring = set_sqlstring_from_grid_fields(gridlistindex)
			Log("resultset sqlstring: " & sqlstring)
			If sqlstring = "more than 1 table" Then Return "ERROR: more than 1
table"
			Dim sqlparts As List = Regex.Split(" ",sqlstring.Trim)
			Dim fldslst As List = Regex.Split(",",sqlparts.Get(1))
		End If
	End If
	Log("resultset sqlstring: " & sqlstring)
	If sqlstring = "" Then Return ""
	Dim reclst As List
	reclst.Initialize
	Select currtab
		Case 0
			Private rs1 As ResultSet = dbsqlite.ExecQuery(sqlstring)
			Do While rs1.NextRow
				Dim rec As List
				rec.Initialize
				For i = 0 To fldslst.Size -1
					Dim fldvalue As String = rs1.GetString2(i)
					If fldvalue <> Null Then
						rec.Add(fldvalue)
					Else
						rec.Add("NULL")
					End If
				Next
				reclst.Add(rec)
```

```
                Loop
                rs1.Close
            Case 1
                Dim dbname As String = tamysqldb.text
                dbmysql.ExecQuery("USE " & dbname)
                Dim rs As ResultSet = dbmysql.ExecQuery(sqlstring)
                Do While rs.NextRow
                    Dim rec As List
                    rec.Initialize
                    For i = 0 To fldslst.Size -1
                        Dim fldvalue As String = rs.GetString2(i)
                        If fldvalue <> Null Then
                            rec.Add(fldvalue)
                        Else
                            rec.Add("NULL")
                        End If
                    Next
                    reclst.Add(rec)
                Loop
                rs.Close
        End Select
        sb.Append($"[table Alignment=Center rows=${reclst.size+1}
cols=${gridfields.Size}]"$)
        sb.Append($"[row RepeatHeader=True]"$)
        For i = 0 To fldslst.Size -1
            If gridfields.IndexOf(fldslst.Get(i)) <> -1 Then
                sb.Append($"[cell color=blue width=100][p
alignment=left][color=white][b]${fldslst.Get(i)}[/b][/color][/p][/cell]"$)
            End If
        Next
        sb.Append($"[/row]"$)
        For i = 0 To reclst.size -1
            sb.Append($"[row]"$)
            Dim rec As List = reclst.Get(i)
            For j = 0 To fldslst.Size -1
                If gridfields.IndexOf(fldslst.Get(j)) <> -1 Then
                    sb.Append($"[cell width=100][p
alignment=left]${rec.get(j)}[/p][/cell]"$)
                End If
            Next
            sb.Append($"[/row]"$)
        Next
        sb.Append($"[/table]"$)
        maxtablerows = reclst.Size
        Return sb.ToString
End Sub
```

Changes to existing code:

- Make sure there is a space before the SQL keywords in the button click events subroutines:

```
tasql.Text = tasql.Text & " " & CRLF & btnfrom.Text
tasql.Text = tasql.Text & " " & CRLF & btnwhere.Text
tasql.Text = tasql.Text & " " & CRLF & btnorderby.Text
tasql.Text = tasql.Text & " " & CRLF & btngroupby.Text
tasql.Text = tasql.Text & " " & CRLF & btnhaving.Text
tasql.Text = tasql.Text & " " & CRLF & btnand.Text
tasql.Text = tasql.Text & " " & CRLF & btnor.Text
tasql.Text = tasql.Text & " " & btnnot.Text
tasql.Text = tasql.Text & " " & CRLF & btninnerjoin.Text
tasql.Text = tasql.Text & " " & CRLF & btnunion.Text
```

- You can move the grid down if the SQL textarea is active. Add a test in the tbtndesign_SelectedChange subroutine just below the **if Selected** test.

```
If tasql.Text <> "SELECT " And tasql.Text <> "" And tbtnsql.Selected = True Then
        pndesign.Top = 250
        pncanvas.Top = 300
        spgrid.Top = 300
End If
```

- And to reset the position of the grid add code below the **else** line of the If Selected test.

```
        pndesign.Top = 50
        pncanvas.Top = 100
        spgrid.Top = 100
```

In the next step you can find some code to save and load the grid labels.