

Session VII

This week we will discuss Variables.

Naming your variables

1. Variable names can contain characters, numbers, and punctuation marks
2. Can't contain, # \$ & % @ or !
3. Can't begin with a number or a space
4. Don't use reserve words (reserve words are words that have a special meaning to Access operations)
5. Must be less than 255 characters

Variable Data Types

If you don't understand all of the code don't worry, we will eventually get to the code later in the class. The main thing to learn is what data types there are and with which data you would select a specific data type.

A variable length string is represented as "" which means it is a zero length string. A fixed length string is filled with the ASCII Char 0. Variant variables are initialized to empty. An empty variable is represented by 0 in a numeric context.

Scope

The **lifetime** of variables determines how long a variable retains its value. Some variables only live while they are being accessed in a function. Other variables have a life of their own and can be accessed throughout your application. When you call a function the procedure level variables in that function only exist while that function is running. Once the function is complete the variables are removed from memory.

Indenting

In working with decision structure procedures they are much easier to understand if you use standard indenting procedures. It is also helpful to provide blank lines between groups of statements that perform a specific task. Indentations and blank lines do not affect the compilation or performance of your code; they simply make it easier to read and to understand.

Option Explicit

The **Option Explicit** statement tells Access to remind you to always declare your variables. The declaration of variables is referred to as an **Explicit declaration**.

If you don't declare a variable it is referred to as an **Implicit** variable and Access assigns the variant data type. The variant data type can accept any type of data.

Comment Line

Note the use of the single (') quote at the beginning of the comment lines. This is the comment indicator. If a line begins with a comment indicator it is called internal documentation and the purpose of the line is to help you, as the programmer, or someone else who is maintaining your program to understand what you are doing. Sometimes if you haven't worked with a program for a long period of time it's easy to forget what a section of code is all about and how it works.

Reading Assignment:**Variables:**

For an understanding of what variables are refer to:

Korol – Pg. 54

Declaring Variables:

Korol – Pg. 55

Variants:

Korol – Pg. 52

Constants:

Korol – Pg. 81-82

Enum:

Korol – Pg. 83-84

Global Variables:

Korol – Pg. 71 - 72

Comments:

Korol – Pg. 41 - 42

Scope:

Korol Pg. 65 - 75

Option Explicit:

Korol Pg. 67

Understanding and Using Static Variables

Korol Pg 75

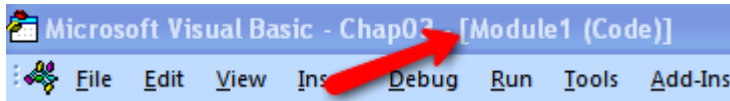
Indenting

Korol Pg 40

The following Hands-On exercises are from the Korol book:

Hands-On 2-2 **Using the Indent/Outdent Features** Pg. 41

Hands-On 3-1 **Using Variables** Pg. 57

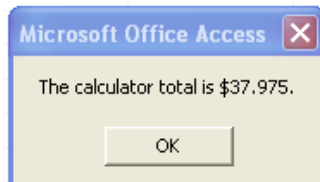


Hands-On 3-2 **Understanding the Data Type of a Variable** Pg. 60.

Hands-On 3-3 **Using Type Declaration Characters in Variable Names** Pg. 61.

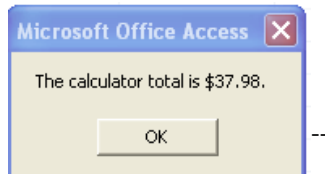
Note that in this code the author does not write the name of the type of data but instead uses **Type Declaration Characters**. (see table 3-2)

Hands-On 3-4 **Assigning Values to Variables** Pg. 62 - 63.



Hands-On 3-5 **Forcing Declaration of Variables** Pg. 65.





Hands-On 3-6 **Understanding Module-Level Variables** Pg. 69

Hands-On 3-7 **Using Static Variables** Module 5 Pg. 76

Hands-On 3-8 **Working with Object Variables** Module 6 Pg. 78

Hands-On 3-11 **Exploring Access's Constants** Pg. 83

Hands-On 3-12 **Using Intrinsic Constants in a VBA Procedure** Pg. 84

Helpful Hint for the quiz: Know the variables for 2, 4, 8, and 14 bytes.

Project

Amanda has met with the managers who will be using the MovieCam application to determine the variations of reports that they will need. Amanda does not want managers to have access to the Navigation Pane, but she wants you to design a switchboard form that will give managers access to the reports they need. She wants you to use VBA procedures to execute various actions from the switchboard form.

Richard Jenkins and personnel want to be able to print a report that is based on whether the employee is terminated or not, another based on whether an employee is salaried or hourly, and a third according to the department. He also wants to be able to generate these reports using filters. Martin Woodward and production want to be able to print job reports that are based on a job number. He wants to be able to specify the first few digits and job number such as "92" or "99" so that you can print a report of a job that start with these numbers. He also needs to run reports that are based on time cards date and on a range of dates, such as a month or a year.

Microsoft Office Access provides an Access Switchboard Manager to generate generic switchboards for you. Although this is useful for creating a simple switchboard to open other forms and reports, you can develop a much more extensible user interface by creating your own switchboards and utilizing your programming skills. Switchboards are typically the initial point of entry for the user, and when designed properly, multiple switchboards can help control workflow and simplify a series of tasks.

In this session and future sessions, you will create a switchboard form that looks like the one shown below: it contains two Option Groups: one for employee reports and one for

job reports. You anticipate that you will add more reports to each option group. The Switchboard will contain check box, combo box, and text box controls from which users can select the criteria for the report that they need to run. One set of controls will apply to Employee Reports, and another set of controls will apply to Job reports. Because the reports in each category are based on the same tables or queries, the controls for selecting the criteria will be nearly the same for each report in a category.

Reports Switchboard

MovieCam Technologies

Employee Reports

- ☒ Employee List
- ☒ Employees By Department

Terminated ☐

Hourly/Salaried

Department

Job Reports

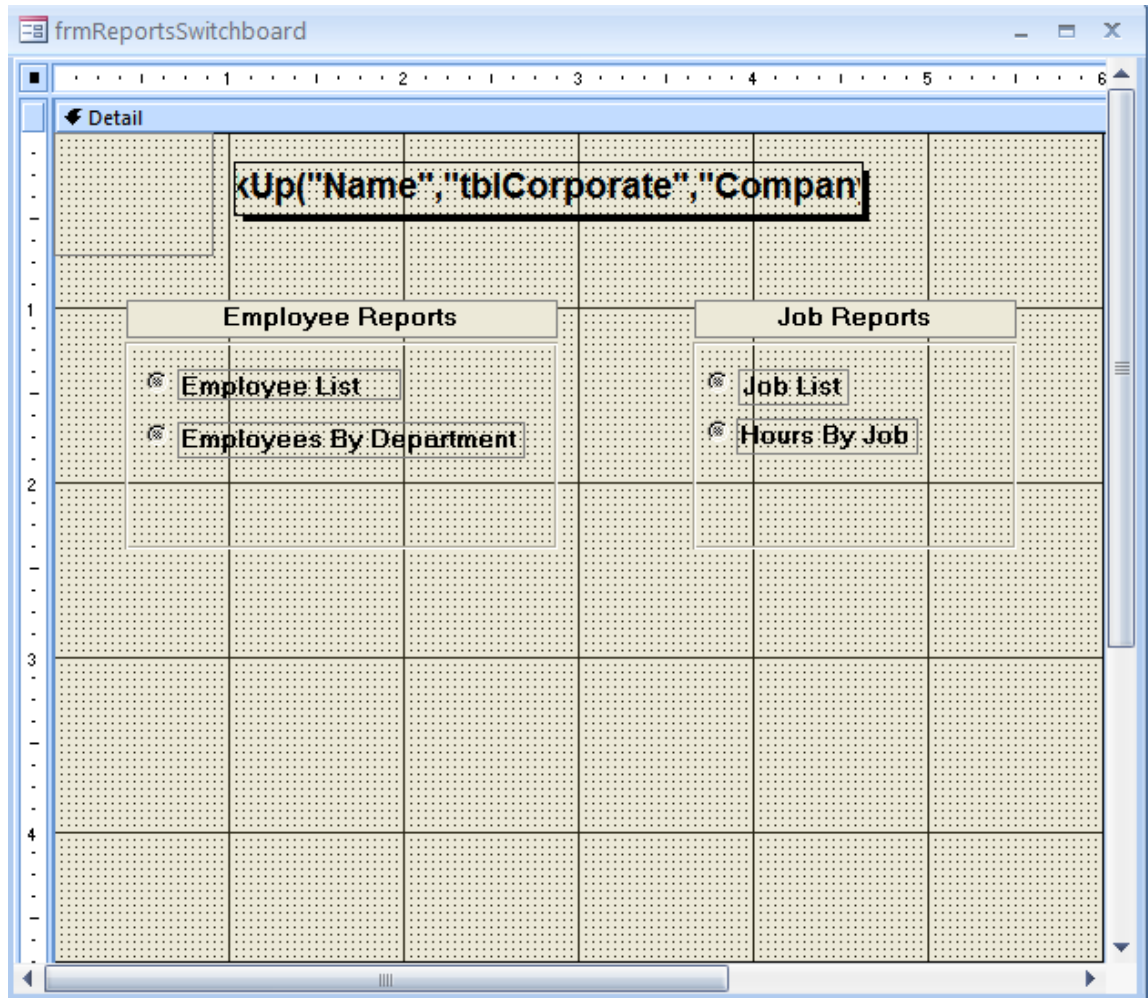
- ☒ Job List
- ☒ Hours By Job

Job ID

Beginning & Ending Dates:


Preview **Close**

Jason had previously worked on a report switchboard. It currently looks like the one shown below: it contains two option groups, grpEmployees and grpJobs, and each option group contains two option buttons. Each option button represents a report.



When the report switchboard is completed, clicking the Employee List button in the Employee Reports group will open the rptEmployeeslist report shown below: You plan to design the switchboard so that the user can open a report that shows records for active employees or terminated employees. To do that you need to add a checkbox to the form rptSwitchboard form. You will test the value of this checkbox in the event procedure that you will write later in this session and use its value to filter the records for the report when it opens. Recall that an event procedure is a procedure that is triggered by an event in a form or report and is stored in the form or report class module.

Employee List



MovieCam Technologies
Employee List

Emp No.	Name	Department	Job Title	Labor Category	Hourly Rate	Hourly/ Salary
10	Thomas Arquette	Engineering	Engineering Manager	Engineering	\$59.50	S
20	Juan Gerardo	Engineering	Engineering Associate	Engineering	\$25.80	S
99	Janice Smitty	Accounting	Accountant	Overhead	\$24.00	S
150	Carolyn Valdez	Accounting	Accountant	Overhead	\$45.25	S
200	Dan Sylvester	Production	Electronic Assembler	Production	\$21.00	H
210	Martin Woodward	Product Management	Product Manager	Production	\$39.33	S
230	Sandra Miller	Production	Production/Assembly	Production	\$36.50	S
300	Jack Huft	Engineering	Mechanical Engineer	Engineering	\$39.99	S
420	Todd Combs	Production	Production/Assembly	Production	\$14.21	S
500	Alan Cook	Production	Production/Assembly	Production	\$19.60	H
550	Michael Eichman	Production	Production/Assembly	Production	\$26.03	S
600	Gloria Cauldwell	Engineering	Engineering Associate	Engineering	\$39.22	H
700	Ernest Gold	Production	Production/Assembly	Production	\$17.00	H
800	Ann Garcia	Accounting	Accounting Clerk	Overhead	\$15.30	H
900	Jason Hart	Production	Production/Assembly	Production	\$12.90	H
Total Employees			15			

Page: 1 Unfiltered

Adding a Checkbox to the frmReportSwitchboard Form:

Step 1

Start Access and open the Session 7 Movie database.

Step 2

Open the frmReportSwitchboard form in design view, and then make sure the Controls section is available in the Ribbon.

Step 3

Click the Check Box button in the ribbon, and then click the form below the grpEmployees option group to insert the check box as shown below. If your new check box is not named Check27 as in the figure do not be concerned as we will be giving it a new name.

Step 4

Click the **Properties Sheet** in the Ribbon (if it is not already open), click the **All** tab if it is not already displayed, change the name property of the checkbox to **chkTerm**, and then change the Default Value property to **=No**. This sets the checkbox so that it will default to No. Later in this tutorial when you write code to test the value of the checkbox, you want it to default to No if nothing has been checked on the form.

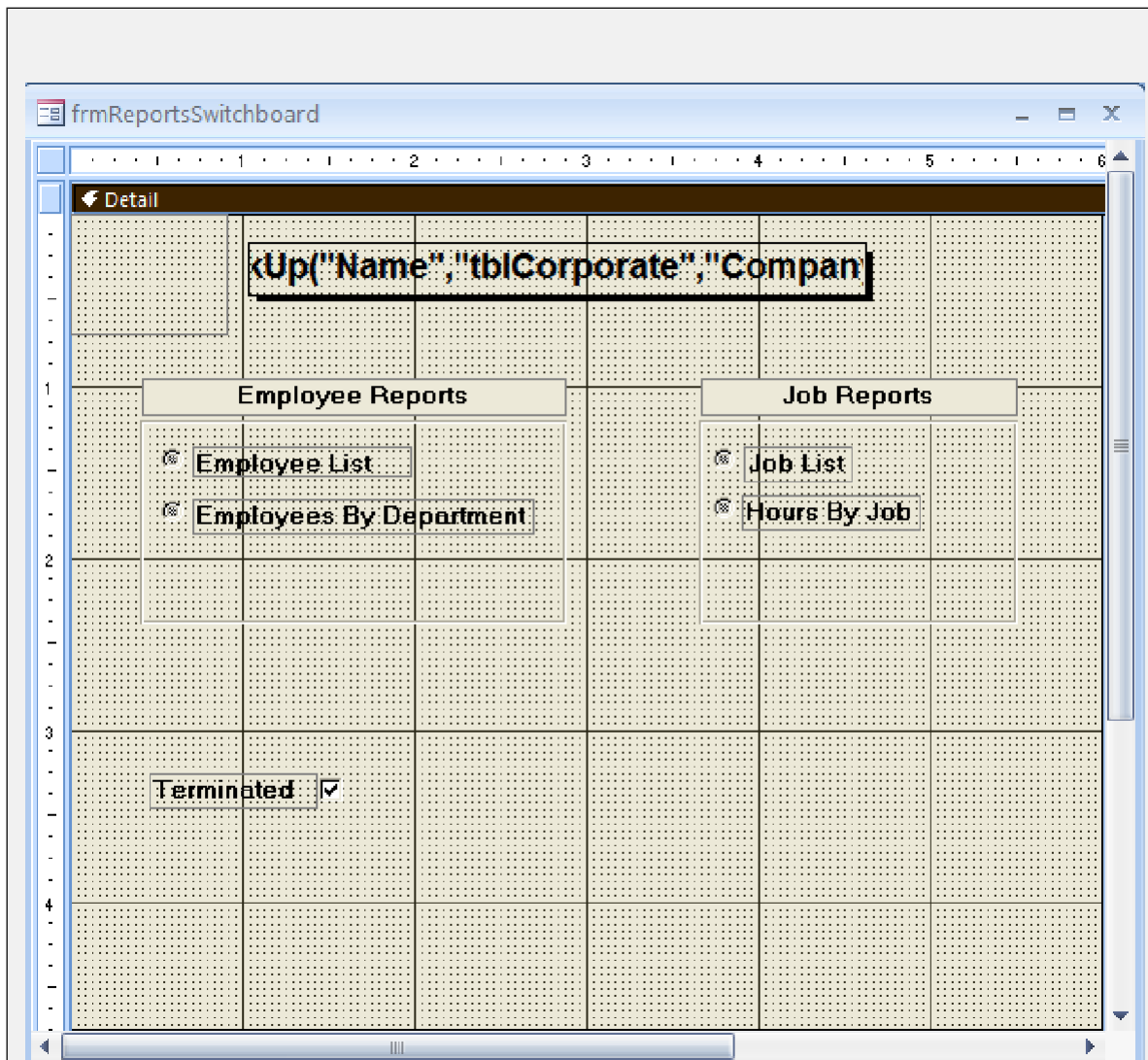
Step 5

Click the **label control** for the checkbox to display its properties, change the checkbox label's **Caption property** to **Terminated**, move the label to the left of the checkbox, and then resize the label so that the entire label text is visible.

Note: To move the label to the left of the checkbox you must click on the gray square in the top left corner of the label.

Step 6

Close the properties sheet, and then click an empty area of the form to deselect the control. The Design view of the form should appear as below:



You will use the OpenReport method of the DoCmd object to open reports from the reports switchboard. The OpenReport method contains a where condition argument that will be used to filter the records of the report that is being opened. The syntax of this method is as follows:

DoCmd.OpenReport (Report name, View, Filter name, Where condition, Windows mode)

The OpenReport method arguments are as follows:

- Report name: Opens the Report.
- View: Indicates the view in which the argument is opened
 - AC view Design: design view

- AC viewed Normal: prints a report immediately and is the default.
- AC view Preview: print preview

Filter name: Specifies a filter to be used with the report

- Where condition: Specifies a where condition
- Windows mode: window mode is optional and can be Normal, Hidden, Minimized, or Dialog.

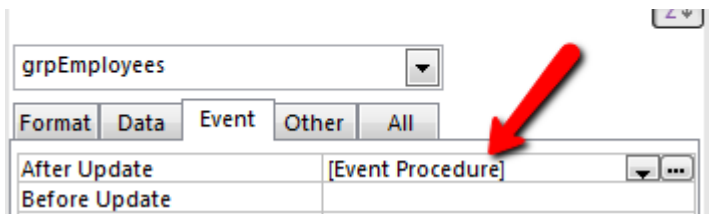
Open report, you can leave an optional argument blank in the middle of the syntax, but you must include the arguments comma. For example, the following statement opens the rptEmployeesList report in Print Preview and displays the data for all employees that were terminated by placing double quotes "Terminated =yes" in the designated placeholder for the Where Condition Argument:

DoCmd.OpenReport "rptEmployeeList", acViewPreview,, "Terminated = Yes"

Adding an Event Procedure to the frmReportsSwitchboard Form:

Step 1

Click the **grpEmployees** option group, click the **Properties** button in the ribbon, click the **Event** tab, click in the **After Update** text box, and then click its **Build** button (...). The Visual Basic window opens, and the insertion point is positioned between the first and last lines of the procedure.



Step 2

Press the Tab key to indent the first line of code that you will type, and then type the following code with additional indenting as indicted (note: do not wrap the text for each

DoCmd.OpenReport method to a second line in the code window, but rather type it all on one line):

```
Select Case grpEmployees
    Case 1
        DoCmd.OpenReport "rptActiveEmployees", acViewPreview, , "Terminated = " & chkTerm
    Case 2
        DoCmd.OpenReport "rptEmployeesByDept", acViewPreview, , "Terminated = " & chkTerm

End Select
grpEmployees = Null
```

Your code should appear as the code below:

```
Private Sub grpEmployees_AfterUpdate()
    Select Case grpEmployees
        Case 1
            DoCmd.OpenReport "rptActiveEmployees", acViewPreview, , "Terminated = " & ChkTerm
        Case 2
            DoCmd.OpenReport "rptEmployeesByDept", acViewPreview, , "Terminated = " & ChkTerm

    End Select
    grpEmployees = Null
End Sub
```

Compiling and Testing the Code in the frmReportsSwitchboard Form:

Step 1

Click **Debug** on the menu bar, and then click **Compile MovieCam**.

Step 2

Click the **Save** button on the toolbar, and then close the Visual Basic Window.

Step 3

Switch to form view.

Step 4

Test the code by clicking the Terminated check box, and then click the Employee List button. The report opens with only four records, representing the four employees whose employment has been terminated.

Step 5

Close the report; **clear** the terminated checkbox, and then click the Employee list button again. The report opens with 13 records, representing the 13 employees who are actively employed at the company.

Step 6

Close the report to return to the switchboard.

Adding a Comment to the Code in the frmReportsSwitchboard form:

Step 1

Switch to Design view and then click the **View Code** button which is under Design in the Tools section to open the Visual Basic window.

Step 2

Position the insertion point at the beginning of Select Case and then press the enter key twice.

Step 3

Press the **Tab** key to indent the line, type **'Open rptEmployeesList or rptEmployeesByDept** and then press the **Enter** key.

Step 4

Type **'based on criteria entered into the form by the user.** (Be sure to include two spaces following the apostrophe at the beginning of the comment line to indicate that the text is a continuation of the comment started on the preceding line; also be sure to include the period at the end of the comment), and then press the **Enter** key. Your code should look like the code shown below: notice that the comments are a different color from the code itself.

```
grpEmployees      AfterUpdate
Option Compare Database

Private Sub grpEmployees_AfterUpdate()
    'Open rptEmployeesList or rptEmployeesByDept
    ' based on criteria entered into the form by the user

    Select Case grpEmployees
    |   Case 1
        DoCmd.OpenReport "rptEmployeesList", acViewPreview, , "Terminated = " & chkTerm
    |   Case 2
        DoCmd.OpenReport "rptEmployeesByDept", acViewPreview, , "Terminated = " & chkTerm
    End Select
    grpEmployees = Null
End Sub
```

Modifying the code in the SRM reports switchboard form:

Step 1

Position the insertion point at the end of the **Option Compare Database** statement in the Declarations section, and then press the **Enter** key.

Step 2

Type **Option Explicit.**

```
Option Compare Database
Option Explicit
|

Private Sub grpEmployees_AfterUpdate()
```

Step 3

Under AfterUpdate() line press indent and type **Dim strSQL as String** and press Enter.

```
grpEmployees  AfterUpdate

Option Compare Database
Option Explicit

Private Sub grpEmployees_AfterUpdate()
    Dim strSQL As String

    'Open rptEmployeesList or rptEmployeesByDept
    ' based on criteria entered into the form by the user.

    Select Case grpEmployees
        Case 1
            DoCmd.OpenReport "rptEmployeesList", acViewPreview, , "Terminated = " & chkTerm
        Case 2
            DoCmd.OpenReport "rptEmployeesByDept", acViewPreview, , "Terminated = " & chkTerm
    End Select
    grpEmployees = Null
End Sub
```

Step 4

Position the insertion point at the end of the “based on criteria entered into the form by the user. (After the period), Press the **Enter** key twice, type **strSQL = “Terminated = “& chkTerm** to set the value of the strSQL variable. Press the **Enter** key.

Step 5

Delete the string “**Terminated = “& chkTerm** in each of the DoCmd.OpenReport statements, and type **strSQL** in its place. Your code should look like that below:

```
grpEmployees  AfterUpdate

Option Compare Database
Option Explicit

Private Sub grpEmployees_AfterUpdate()
    Dim strSQL As String

    'Open rptEmployeesList or rptEmployeesByDept
    ' based on criteria entered into the form by the user.

    strSQL = "Terminated = " & chkTerm

    Select Case grpEmployees
        Case 1
            DoCmd.OpenReport "rptEmployeesList", acViewPreview, , strSQL
        Case 2
            DoCmd.OpenReport "rptEmployeesByDept", acViewPreview, , strSQL
    End Select
    grpEmployees = Null
End Sub
```

Step 6

Click the **Debug** on the menu bar, and then click **compile MovieCam** to compile the code and check for errors.

Step 7

Click the **Save** button on the toolbar to save your changes, and then **Close** the Visual Basic window.

Step 8

Switch to form view and test the form by running the employee list report for terminated employees, closing it, and then running it again for active (non-terminated) employees. The form should produce the same results as when you ran the reports before: the list of terminated employees should include three records, and the list of active employees should include 13 employees.

Step 9

Switch to design view.

You are going to modify the code for the frmReportSwitchboard form. Richard Jenkins in the personnel department needs to run the employee reports for various conditions. He wants to be able to print reports of all active employees or all terminated employees. At other times, he might need reports on all terminated employees who are salaried, or all active employees who are salaried. And occasionally he needs to see all the salaried, active employees for one department only.


The report switchboard can be much more efficient for enabling this functionality when compared to writing predefined queries or even multiple reports. If you choose a predefined query approach, in order to provide for all the scenarios, the personnel department needs, you have to write several different queries and possibly several different reports. You can still use one report and change the record source property in code to point to the appropriate query, but that still requires several queries to be set up. Every query and report definition requires space in your database file which is part of the overall 2 GB file size limitations. You will learn how to provide for all these different scenarios without having to create multiple queries and reports, but rather by expanding the functionality of your user interface, specifically the report switchboard. The approach you will work with next often precludes the need for constant modifications or additional queries, because the reports switchboard can provide so many variations from one screen.

Before modifying the code for the form report switchboard form, you need to add some controls to the form. First you will add a combo box from which members of the personnel department can choose either hourly or salaried. You plan to use a combo box because the data is stored in this field as H for hourly or S for salaried, and that is what needs to be typed on the form. Richard or someone else in the personnel department might input the word "hourly" or the word "salaried" if you simply provide a text box, and such entries would not match what is stored in the tblEmployees table. The report would not display any records due to this mismatch. You use a combo box with a value list to ensure the users can only choose an appropriate value that will match a value in the hourly salaried field of the tblEmployees table. A combo box with the values list will

store the value selected in memory for later use, rather than storing it in an underlying table or query.

You will add a column box with the DeptName and DeptNo fields so that the appropriate department can be selected from the list, and the user does not misspell a department or type in a department that does not exist.

Detail



<Up("CompanyName","

Employee Reports

☐ Employee List

☐ Employees By Department

☒ Terminated

Hourly/Salaried: Unbound

Department: Unbound

Creating a combo box:

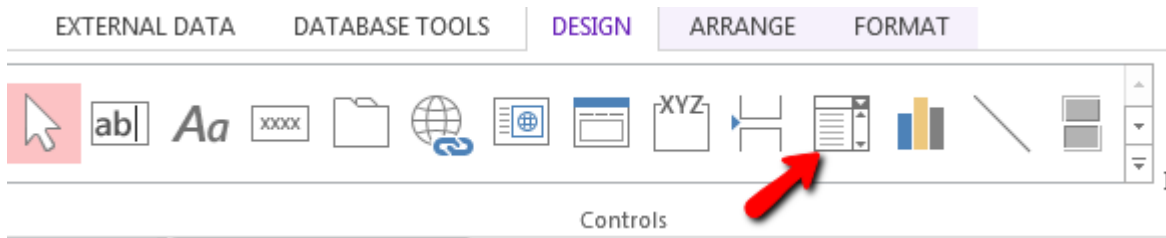
Step 1

Make certain that the frmReportsSwitchboard is open in design view.

Step 2

Make sure that the **Design** tab is selected and you can view the **Controls**.

Make certain the Control Wizard button has been selected:



Step 3

Click the **Combo Box** button in the Controls section and then click beneath the Terminated checkbox to insert a combo box. The combo box wizard opens.

Step 4

Select **I will type in the values that I want** option, and then click the **Next** button.

Step 5

Type **2** in the Number of columns text box, position the insertion point in the first row of Col1, and then type **H**.

Step 6

Press the **Tab** key to move to column 2, type **Hourly**, and then press the **Tab** key. This allows you to display the word "hourly," but to also store the "H" for use in the WhereCondition which is column 1.

Step 7

Type S, press the Tab key and then type Salaried as show below:

Combo Box Wizard

What values do you want to see in your combo box? Enter the number of columns you want in the list, and then type the values you want in each cell.

To adjust the width of a column, drag its right edge to the width you want, or double-click the right edge of the column heading to get the best fit.

Number of columns:

	Col1	Col2			
	H	Hourly			
	S	Salaried			
*					

Cancel < Back Next > Finish

Step 8

Click the **Next** button to move to the next combo box Wizard dialog box which you specify the column that contains the data you want to store for use later.

Step 9

Make sure that column 1 is selected, and then click the **Next** button. This will save “H” or “S” to use later. The last dialog box prompts you to enter a label for the combo box.

Step 10

Type **Hourly/Salaried** and then click the **Finish** button.

Step 11

Click the **Properties** button in Ribbon to open the properties sheet, click the **All** tab if necessary, change the Name property to **cboHS**, change the column's width property to **0"; 1"**, change the list width property to **1"**, then close the property sheet these changes will hide the first column, which contains H and S, although the first column values will be retained for later use.

Step 12

Resize a label to the left of the column box so that all of the text is visible.

Step 13

Switch to Form view and test the combo box by clicking the hourly/salaried list arrow. The options Hourly and Salaried should appear in the drop-down list.

Reports Switchboard



MovieCam Technologies

Employee Reports

- Employee List
- Employees By Department

Job Reports

- Job List
- Hours By Job

Terminated ☒

Hourly/Salaried Salaried ▼

Creating the CBO department combo box:

Step 1

Switch to Design view, click the **Combo Box** button on the Ribbon, and then click below the cboHS combo box control to create a new combo box.

Step 2

Click the **Next** button to accept the default option of **I want the combo box to look up the values in a table or query**, click **Table: tblDepartments** in the next dialog box, and then click the **Next** button.

Step 3

Add the **DeptNo** and **DeptName** fields to the Selected Fields list box, click the **Next** button, choose the **DeptName** for ascending sort order, and then click the **Next** button. Widen the DeptName column by dragging the edge of the column to the right or by double-clicking the right edge of the column so the width accommodates the widest entry, making the entire department name for each department visible. Note that the

checkbox above the department name titled "Hide key column (recommended)" should already be Checked; if it is not, check it.

Step 4

Click the **Next** button, type **Department** as the label for the combo box, and then click the **Finish** button.

Step 5

Click the **Properties** button on the Ribbon, change the name property of the combo box to **cboDept**, and then close the property sheet.

Step 6

Save the form and then size and align the controls on the form so they look like those shown below: remember that you can use the Align Size commands available on the Quick toolbar or in the Ribbon. Save the form again when finished with all adjustments.

You will now write the code so that the user can choose any combination of the three options on the form. To do this, you will first test to see if each of the controls is null. You will use the IsNull function.

The IsNull function tests the expression to see whether it is null or not. In this case, you want to know if the expression is not null, that is, whether it contains a value, so you will include the Not operator before the function. Then you will set a variable called strWhere. At the beginning of the procedure you will declare that variable equal to whatever is already contained in the strWhere, and equal to the part of the Where condition string that applies to the control you are testing.

You will then use a series of **If... Then** statements to test each of the three controls on the form.

Writing the Code for Building the Where Condition:

Step 1

Under the Design tab in the Click the View Code button on the Ribbon to open the Visual Basic window, position the insertion point at the end of the Dim strSQL As String, and then press the enter key.

Step 2

Type **Dim strWhere As String** to declare the other string variable. It will be used to build the WhereCondition. The variable strSQL will be used to trim off leading characters after you build the WhereCondition.

Step 3

Press the enter key twice, press the tab key, and type the following comment: **'Build the where condition to use in the open report method.**

Step 4

Press the enter key twice, and then type the following: Remember cboDept is a number and is not enclosed in quotes.

```
IfnotIsNull(cboDept)Then
    strWhere = strWhere & " AND DeptNo = " & cboDept & ""
End if
```

Step 5

Select the **strSQL = "terminated = " & chkTerm** line of code, press the **Delete** key, and then delete any extra blank lines until there is only one blank line separating the comment and the select case line of code.

You cannot be certain from which of the three controls the user will choose. To allow for this, you need to construct the where condition so that it always concatenates whatever it already contains and the strWhere variable to the keyword and then to the remaining string expression for that particular control. This way, no matter which or how many of the three controls are chosen, they are joined together with the keyword AND. Once you have finished building strWhere, you would use a function to trim off the leading AND keyword as well as any unneeded spacing before and after AND.

Step 7

Position the insertion point at the end of the End If statement, press the enter key twice, and then type the following:

```
If not IsNull (cboHS) Then
    strWhere = strWhere & " AND HourlySalaried = "&"'"
    & cboHS & "'"
End If
```

Step 8

Press the enter key twice, and then type the following:

```
If not IsNull (chkTerm) Then
    strWhere = strWhere & " AND Terminated = "&chkTerm
End If
```

```
Private Sub grpEmployees_AfterUpdate()
    Dim strSQL As String
    Dim strWhere As String

    'Build the where condition to use in the open report method.

    If Not IsNull(cboDept) Then
        strWhere = strWhere & " AND DeptNo = " & cboDept & ""
    End If
    If Not IsNull(cboHS) Then
        strWhere = strWhere & " AND HourlySalaried = " & "'" & cboHS & "'"
    End If

    If Not IsNull(chkTerm) Then
        strWhere = strWhere & " AND Terminated = " & chkTerm
    End If
```

Technically, because you have previously forced the default value for chkTerm, this checkbox should never be null; however, it is always good code practice be sure that the value in a field or variable is not null before you use it in a calculation or as part of a WhereCondition.

Step 9

Click the save button on the toolbar.

Adding Code to Trim Leading Characters and Spaces from the strWhere Variable:

Step 1

Press the **Enter** key twice to add a blank line after the third End If statement, and then type the following comment: **'Trim the leading "AND" off strWhere and store the new string in strSQL.**

Step 2

Press the **Enter** key twice, and then type the following code:

```
If Not is Null(strWhere) Then
    strSQL = mid$(strWhere, 6)
End if
```

The start argument in the mid-function is 6 because "AND" is a total of five characters, and you want the string to now begin at the six character. The strSQL variable now contains the contents of strWhere without unnecessary leading characters.

Step 3

Change the comment before the DoCmd.OpenReport statement to the following as shown below:

'Open rptEmployeesList or rptEmployeesByDept based on the contents of strSQL.

Step 4

Position the insertion point at the end of the End Select statement, and then press the Enter key twice.

Step 5

Type the following comment: **'set the option group and combo boxes back to null, and clear the check box.** And then press the **Enter** key.

Step 6

Position the insertion point at the end of the grpEmployees = Null statement, press the **Enter** key, and then type the following:

cboDept = Null
cboHS = Null
chkterm = False

```
'Open rptEmployeesList or rptEmployeesByDept based on the contents of strSQL
Select Case grpEmployees
Case 1
    DoCmd.OpenReport "rptActiveEmployees", acViewPreview, , strSQL
Case 2
    DoCmd.OpenReport "rptEmployeesByDept", acViewPreview, , strSQL
End Select

'set the option group and combo boxes back to null, and clear the checkbox
grpEmployees = Null
cboDept = Null
cboHS = Null
ChkTerm = False

End Sub
```

These statements set the option group and combo boxes back to Null and clear the check box by setting its value to False after the report opens. This way the user can make a difference choice without any residual value left in any of the controls. Setting the value of the chkTerm checkbox to false in VBA is the equivalent of setting its default value to No, which is an unchecked checkbox.

Step 7

Press the Enter key to add a blank line between the statements and the End Sub statement.
See Below

Option Compare Database

Option Explicit

Private Sub grpEmployees_AfterUpdate ()

Dim strSQL As String

Dim strWhere As String

'Build the where condition to use in the open report method.

If Not IsNull(cboDept) Then

 strWhere = strWhere & " AND DeptNo = " & "" & cboDept & ""

 strWhere = strWhere & " AND DeptNo = " & cboDept & ""

End If

If Not IsNull(cboHS) Then

 strWhere = strWhere & " AND HourlySalaried = " & "" & cboHS & ""

End If

If Not IsNull(chkTerm) Then

 strWhere = strWhere & " AND Terminated = " & chkTerm

End If

'Trim the leading "AND" off strWhere and store the new string in strSQL.

If Not IsNull(strWhere) Then

 strSQL = Mid\$(strWhere, 6)

End If

'Open rptEmployeesList or rptEmployeesByDept based on

' the contents of strSQL.

Select Case grpEmployees

Case 1

 DoCmd.OpenReport "rptEmployeesList", acViewPreview, , strSQL

Case 2

 DoCmd.OpenReport "rptEmployeesByDept", acViewPreview, , strSQL

 DoCmd.OpenReport "rptEmployeesByDept", acViewPreview, , "Terminated = " &
chkTerm

End Select

'set the option group and combo boxes back to null, and clear the check box.

```
grpEmployees = Null  
cboDept = Null  
cboHS = Null  
chkTerm = False
```

End Sub

Step 8

Click **Debug** on the menu bar, and then click compile MovieCam. If any of your variables are misspelled, you should correct them now.

Step 9

Click the **Save** button on the toolbar.

Step 10

Close the Visual Basic window, switch to Form view, and test the reports.

This concludes the assignment for Project 7. Please submit your work and read the Power Point slides for Session 7.

The screenshot shows a Visual Basic form titled 'frmReportsSwitchboard' in Design View. The form is divided into two main sections: 'Employee Reports' and 'Job Reports'. The 'Employee Reports' section contains two report controls: 'Employee List' and 'Employees By Department'. The 'Job Reports' section contains two report controls: 'Job List' and 'Hours By Job'. At the bottom of the form, there are three controls: a 'Terminated' checkbox (checked), an 'Hourly/Salaried' dropdown menu (set to 'Unbound'), and a 'Department' dropdown menu (set to 'Unbound'). A red arrow points to the 'Employees By Department' report. The top of the form has a label '<Up(\"Name\",\"tblCorporate\",\"Compan)'.

Note The Employee list doesn't work with the Department because the query on which it is based doesn't have

DeptNo as part of it's fields.