

Generated Bookmap

**INSTRUCTOR HANDBOOK
INSTRUCTOR-LED TRAINING**

Course Version:
Course Duration:
Material Number:

Contents

vii **Course Overview**

1 **Unit 1: Generated Learning Group**

2	Lesson: Generated Learning Object	
13	Exercise 1: Create a Dynamic Drop-down List Using Microsoft Excel VBA	10 Minutes
21	Exercise 2: Create a Button to Transfer Selections Using Microsoft Excel VBA	10 Minutes
29	Exercise 3: Using Microsoft Excel VBA with Check Boxes for Filtering (Optional)	10 Minutes

Course Overview

TARGET AUDIENCE

This course is intended for the following audiences:



UNIT 1

Generated Learning Group

Lesson 1

Generated Learning Object	2
Exercise 1: Create a Dynamic Drop-down List Using Microsoft Excel VBA	13
Exercise 2: Create a Button to Transfer Selections Using Microsoft Excel VBA	21
Exercise 3: Using Microsoft Excel VBA with Check Boxes for Filtering (Optional)	29



UNIT OBJECTIVES

- Extend workbook display options with functions and Microsoft Excel VBA



Generated Learning Object

LESSON OVERVIEW



LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Extend workbook display options with functions and Microsoft Excel VBA

Functions and VBA

The Add-in also contains a set of functions that you can use to build formulas. You can use these functions to include data and metadata of data sources into your analysis. For example, you can insert information fields on data source properties, display the measure filter or list the variables of a data source. With the SAPGetData function, you can also define measure values for certain member combinations.

A Microsoft Excel formula for Analysis consists of a function and references to the data source, measures and/or dimensions. You can use the text or the key of an object to use it as reference. You can also use a cell value like B10 as reference.

The formula alias of a data source is displayed and can be changed in the data source properties on the Components tab in the design panel. For measures, dimensions and their members text references are better to read, but if you want to create a multi-language enabled analysis or there are duplicate texts in the meta data of your data source, you should reference these objects with their keys.

You can also use these functions in VBA macros.

Macros

Analysis contains API (Application Programming Interface) methods that can be used in VBA macros that are embedded in Microsoft Office Excel workbooks. Macros are created in the Visual Basic Editor. The Visual Basic Editor can be used to write and edit a macro that is attached to a Microsoft Office Excel macro enabled workbook. The macros can be connected to UI elements that are available on the Developer tab in the menu.



Note:

In Microsoft Excel 2003, you can find the UI elements in the Forms toolbar. You can also use the Analysis functions for creating formulas in VBA macros. Microsoft Office documentation provides information about creating and using VBA macros. The following section describes the Analysis API methods.

The following API methods are available in Analysis.



API Methods in Analysis

- **SAPAddMessage:** Define messages and add them to the standard message dialog
- **SAPCallMemberSelector:** Call the standard input help (member selector) from the worksheet
- **SAPExecuteCommand:** Use this API method as a generic command to execute commands such as hide/show ribbon components, refresh, and so on.
- **SAPExecutePlanningFunction:** Execute Integrated Planning functions (See the BOAN20 class)
- **SAPExecutePlanningSequence:** Execute Integrated Planning sequences
- **SAPGetCellInfo:** Define a command to get information on a crosstab cell such as dimensions and selections
- **SAPGetProperty:** Read properties of a data source or a workbook such as whether the data source is input ready, for example
- **SAPLogon:** Trigger a logon to a system for a specified data source
- **SAPMoveDimension:** Define the position of a dimension in the crosstab, swap the dimension with another one or position a dimension relative to another one.
- **SAPSetFilter:** Define which members of a dimension should be filtered.
- **SAPSetRefreshBehaviour:** Use different API methods one after the other without the result set being refreshed and without the message dialog being shown
- **SAPSetVariable:** Define values for input-ready BW variables (prompts). If you want to set multiple variables, you can use the PauseVariableSubmit command with the SAPExecuteCommand method.

SAPCallMemberSelector

With this API method, you can call the standard input help (member selector) from the worksheet. The standard input help is the dialog that is used for filtering in Analysis. If you define the ReferenceCell parameter, the inplace selector is called. The inplace selector is a smaller version of the member selector. You can use it to select members of a dimension, but you cannot change how members are displayed in the selector, for example to change the display from key to text.

SAPCallMemberSelectorFilter Example

```
Dim lResult As Variant
lResult= Application.Run("SAPCallMemberSelector", "DS_1", "FILTER",
"OMATERIAL")
```

The member selector for dimension OMATERIAL is displayed, and you can select the members you need. You can also select a range of members or change how the members are displayed in the selector.

SAPCallMemberSelectorInplace with Single Selection Example

```
Dim lResult As Variant
lResult= Application.Run("SAPCallMemberSelector", "DS_1", "FILTER",
"OMATERIAL",
ActiveCell, "SINGLE")
```

The inplace selector for dimension OMATERIAL is displayed in single selection mode in the active cell, and you can select the member you need.

SAPExecuteCommand

You can use this API method as a generic command to execute the commands described below. (numbered 1-5)

The system returns one of the following output parameters for each command execution:

- 0 = execution failed
- 1 = execution successful

1. PauseVariableSubmit

Use this command to pause the execution of variables using the parameter "On". To end the pausing, use the command with parameter "Off".

When PauseVariableSubmit is in mode "On", all Analysis formulas and most of the Analysis macros will not be executed. The following macros will be executed: SAPAddMessage, SAPSetVariable, PauseVariableSubmit "Off".

2. Hide/Show

Use this command to hide and show Analysis UI components for a workbook. You can use the command for the types "Ribbon", "ContextMenu" and "TaskPane".

You can hide the complete ribbon ("All") and show it again ("Default"). For the context menu, you can hide the single refresh or the complete menu and show the complete menu again. For the design panel, you can hide the Analysis and the Components tab and show the complete design panel again.

The following commands are supported:

- `Application.Run("SAPExecuteCommand", "Hide", "Ribbon", "All")`
- `Application.Run("SAPExecuteCommand", "Show", "Ribbon", "Default")`
- `Application.Run("SAPExecuteCommand", "Hide", "ContextMenu", "Refresh_Single")`
- `Application.Run("SAPExecuteCommand", "Hide", "ContextMenu", "All")`
- `Application.Run("SAPExecuteCommand", "Show", "ContextMenu", "Default")`
- `Application.Run("SAPExecuteCommand", "Hide", "TaskPane", "Analysis")`
- `Application.Run("SAPExecuteCommand", "Hide", "TaskPane", "Components")`
- `Application.Run("SAPExecuteCommand", "Show", "TaskPane", "Default")`

3. ShowPrompts

Use this command to display the prompts dialog. You can call the workbook or data source prompts dialog. To call the dialog, the data source(s) need to be refreshed.

Show prompts dialog for one data source

```
lResult= Application.Run("SAPExecuteCommand", "ShowPrompts", "DS_1")
```

If you execute this command, the prompts dialog for DS_1 is displayed. This is only possible if the variables are not merged in the workbook.

Show prompts dialog for all data sources

```
lResult= Application.Run("SAPExecuteCommand", "ShowPrompts", "ALL")
```

If you execute this command, the prompts dialog for all data sources in the workbook is displayed.

4. Refresh

Use this command to initially refresh the data in the workbook. You can specify one data source or one planning object as a parameter that should be refreshed. If you do not enter a parameter or you enter the string 'ALL' as parameter, all data sources and planning objects will be refreshed. If you execute this command for a data source which is already refreshed, all corresponding crosstabs are redrawn.

Additionally, you can refresh a list of data sources.

Refresh one data source

```
Dim lResult As Long
lResult= Application.Run("SAPExecuteCommand", "Refresh", "DS_1")
```

If you execute this command, the data for data source DS_1 is refreshed.

Refresh all data sources

```
Dim lResult As Long
lResult= Application.Run("SAPExecuteCommand", "Refresh")
```

If you execute this command, the data of all data sources are refreshed.

Refresh a list of data sources

```
Dim lResult As Long
lResult= Application.Run("SAPExecuteCommand", "Refresh"; "DS_1;DS_2")
```

If you execute this command, the data of data source DS_1 and DS_2 is refreshed.

5. Refresh Data

Use this command to refresh all or defined data sources in the workbook. You can specify the data sources that should be refreshed. If you do not enter a parameter or you enter the string 'ALL' as parameter, all data sources will be refreshed. If you execute this command for a data source, the corresponding data is updated from the server and the crosstabs are redrawn.

Refresh two defined data sources

```
Dim lResult As Long
lResult= Application.Run("SAPExecuteCommand", "RefreshData", "DS_1;DS_3")
```

If you execute this command, the data for data source DS_1 and DS_3 is refreshed from the server.

SAPMoveDimension

With this API method, you can define the position of a dimension in the crosstab, swap the dimension with another one or position a dimension relative to another one.

To call the method, use `Application.Run` and specify the input parameters shown in the examples that follow.

Example:

```
Dim lResult As Long
lResult= Application.Run("SAPMoveDimension", "DS_1", "OCOUNTRY", "AFTER",
"OCALMONTH")
```

If you execute this command, dimension OCOUNTRY will be inserted in the crosstab to the position after dimension OCALMONTH.

Example

```
Dim lResult As Long
lResult= Application.Run("SAPMoveDimension", "DS_1", "OCOUNTRY", "ROWS",
"2")
```

If you execute this command, dimension OCOUNTRY is added as second entry to the rows in the crosstab.

SAPSetVariable

With this API method, you can define values for input-ready BW variables (prompts). If you want to set multiple variables, you can use the `PauseVariableSubmit` command with the `SAPExecuteCommand` method.

To call the method, use `Application.Run` and specify the following input parameters:

- **Prompt Name:** Name or technical name of the BW variable to be filtered.
- **Prompt Value:** String that represents the value for the prompt, for example the technical name. Note the syntax rules for entering values.
- **Value Format:**
 - **Text:** Single member as text.
 - **Key:** Single member as key.
 - **INTERNAL_KEY:** Single member with its internal key.
 - **INPUT_STRING:** Complex selection of members.
 - **INPUT_STRING_AS_ARRAY:** Returns the input string as string as array. Note that the **KEY** and **INTERNAL_KEY** depend on the InfoObject modeling in SAP NetWeaver BW.
- **Formula Alias:** Enter the formula alias for the data source. You can set the alias when configuring the data source on the *Components* tab in the design panel. You have to define this parameter if the variables in the workbook are not merged. If the variables in the workbook are merged, you can define this parameter but you do not have to define it.

The system returns one of the following output parameters for each function execution:

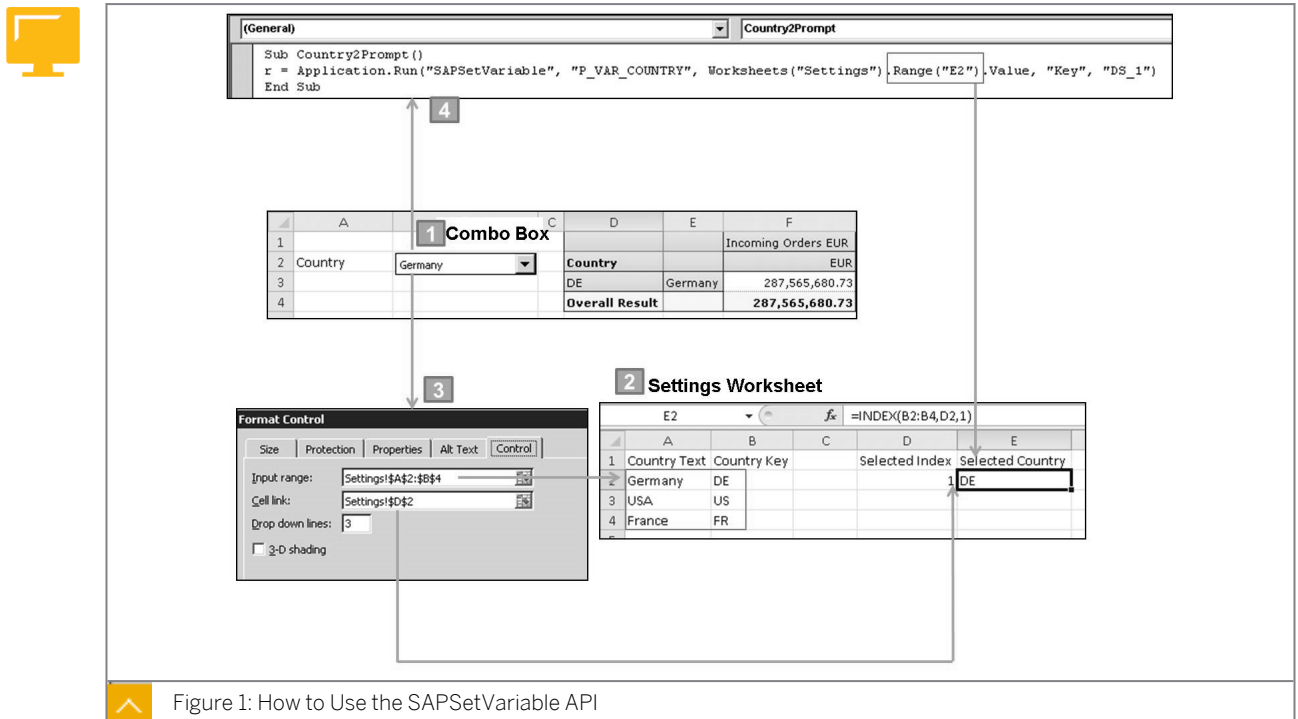
0 = execution failed.

1 = execution successful.

The following is an example of how to use the `SAPSetVariable` API.

We are using a query with a variable for Country. The user wants to select a country via a drop down box (combo box) to select the country. To do this we need to:

1. Insert the combo box. When a new country is selected, the system will perform a refresh with the selected member.
2. Create a *Settings* worksheet with the following:
 - The range of countries text and keys in A2:B4
 - The index function in E2 which converts index values to members
3. Configure the *Format Control* to read the range of members in the *Settings* sheet and generate an index value for the selected country.
4. Create a macro with the following:
 - The SAPSetVariable API calls the P_VAR_COUNTRY variable
 - The Settings worksheet cell E2 provides the selected country key value
 - The DS_1 is the alias for the query with the variable



SAPSetRefreshBehaviour

With this API method, you can use different API methods one after the other without the result set being refreshed and without the message dialog being shown.

To call the method, use `Application.Run` and specify the following input parameters:

- **Mode:** Define whether the mode is On or Off.
- **Message Display:** Define whether the message display is On or Off. The default setting is Off.

The system returns one of the following output parameters for each function execution:

0 = execution failed.

1 = execution successful.

The following is an example of how to use SAPSetRefreshBehaviour. The set up from the previous SAPSetVariable is still in place except the country drop down is not assigned a macro any longer (now we want to use the *Transfer Selection* button to trigger a refresh).

The business requirement is to provide a button to transfer multiple data selections to reduce the overall wait time for data refreshes.

To do this we need to:

1. Insert a button.
2. Create a *Settings* worksheet with the following:
 - The range of divisions text and keys in A8:B10
 - The index function in E8 which converts index values to members
3. Configure the *Format Control* to read the range of members in the *Settings* sheet and generate an index value for the selected division.
4. Create a macro with the following:
 - Use SAPSetRefreshBehaviour to turn refresh off
 - Determine the country id for the P_VAR_COUNTRY variable
 - Determine the division member id for the SAPSetFilter command
 - Turn refresh behavior back on



The figure shows three main components and their interactions:

- Macro (Transfer_Selection):** A VBA sub procedure that:
 - Turns off SAPSetRefreshBehaviour.
 - Runs SAPSetVariable for P_VAR_COUNTRY using the value from cell E2 in the Settings worksheet.
 - Runs SAPSetFilter for DS_1 using the value from cell E8 in the Settings worksheet.
 - Turns SAPSetRefreshBehaviour back on.
- Settings Worksheet:** A worksheet with columns for Division Text, Division Key, Selected Index, and Selected Division.

Division Text	Division Key	Selected Index	Selected Division
Pumps	01		
High Tech	07	2	07
Service	08		
- Format Control:** Configured to read the range of members in the Settings sheet (A8:B10) and generate an index value for the selected division.
- Main Worksheet:** Contains a Country dropdown (USA), a Division dropdown (High Tech), and a Transfer Selections button. It also displays data from SAP, including Incoming Orders EUR and Overall Result (17,486,150.98).

Numbered callouts (1-4) indicate the flow of data and actions between these components.

Figure 2: How to Use the SAPSetRefreshBehaviour API

SAPSetFilter

With this API method, you can define which members of a dimension should be filtered. You can also use this method to define the filter for a planning function.

To call the method, use `Application.Run` and specify the input parameters shown in the examples below.

Example

```
Dim lResult as long
lResult= Application.Run("SAPSetFilter", "DS_1", "0SOLD_TO__OCOUNTRY",
"CA;US;DE", "INPUT_STRING")
```

With this example, you set the filter for dimension `0SOLD_TO__OCOUNTRY` of data source `DS_1` to the countries USA, Canada and Germany using member format `INPUT_STRING`.

Example

The following is an example of how to use `SAPSetFilter`. The business requirement is to provide check boxes to select one or more members and a create filter push button to transfer the selections.

To do this we need to:

1. Insert the checkboxes.
2. Create a *Settings* worksheet with the following:
 - The range of sales organizations text and keys in B4:C7
 - The count function in C9 to determine the number of sales organizations selected
3. Configure the *Format Control* for each check box to read the members in the *Settings* sheet.
4. Create a macro with the following code:

```
Sub Create_Filter()
(This is the sub routine)
Dim Act_true As Integer
(this declares the local variable Act_true as an integer)
Dim Number_true As Integer
(this declares the local variable Number_true as an integer)
Dim Filter_Value As String
(this declares the local variable as a string)
Number_true = Worksheets("Settings").Range("C9").Value
(C9 is the source for Number_true)
For i = 4 To 7
(this is the loop ... i=rows 4 through 7)
If Worksheets("Settings").Cells(i, 3).Value = "True" Then
(if cell C4 ... in loop #1 ... is true then)
Act_true = Act_true + 1
(Act_true is 0 initially ... this makes it 1 in loop #1)
Filter_Value = Filter_Value & Worksheets("Settings").Cells(i, 2).Value
(the Filter_Value = B4 in loop #1)
If Act_true < Number_true Then Filter_Value = Filter_Value & ";"
(in loop #1, if Act_true > C9 ... then concatenate with ; as a separator)
End If
(this ends the if statement)
Next
ret = Application.Run("SAPSetFilter", "DS_1", "0SALESORG", Filter_Value,
"Input_String")
```


This formula returns a list with the dimensions of data source DS_1. If the data source contains only one dimension, the returned array is one-dimensional.

To make your programming easier, you can create an additional function to ensure that the array is always two-dimensional, for example the function GetAsTwoDimArray.



How to Create a List and Use Microsoft Excel VBA to Transfer the Value to a Prompt

For the demonstration steps and data, see the [Create a Dynamic Drop-down List Using Microsoft Excel VBA](#) and [Create a Button to Transfer Selections Using Microsoft Excel VBA](#) exercises.



Create a Dynamic Drop-down List Using Microsoft Excel VBA

Business Example

As an analyst, you often have requirements that call for VBA. Now, you need to use VBA in Analysis to create a dynamic drop-down box that will be used to provide a member value to a variable.

In this exercise, where you see the characters ##, replace them with the number your instructor assigned you.

1. Start Analysis for Microsoft Excel.
2. Log on to the BI Platform via Enterprise authentication, using the following credentials:
User: **train-##**
Password: **train-##**
Authentication: **Enterprise**
3. Log on to the RWB BW server using the following credentials:
Client: **800**
User: **user##**
Password: **user##**
Language: Preferred language key such as **EN** for English, **DE** for German, and so on.
4. Open the Analysis of Special Countries query.
5. Move the crosstab to D1 and turn the *Developers* tab on.
6. Enter a label of Country in A2 with a column width of 20.
7. Add a combo box(drop-down) in B2.
8. Add information in Sheet2 that will allow communication between the drop-down and the macro, and rename Sheet2 to Settings. Enter the information as shown in the *Settings* sheet below. You also need to add the following parameters into the combo box's Form Control:
 - Input range: **Settings!\$A\$2:\$A\$4**
 - Cell link field: **Settings!\$D\$2**
 - Drop-down lines: **3**
 - In the Settings Worksheet, add the following function in cell E2: **=INDEX(B2:B4,D2,1)**



	A	B	C	D	E
1	Country Text	Country Key		Selected Index	Selected Country
2	Germany	DE			
3	USA	US			
4	France	FR			

Figure 4: Settings Sheet

9. Use a macro with the command `SAPSetVariable` to transfer the selected value (in cell `Settings!E2`) to the prompt (`P_VAR_COUNTRY`).
- Name the macro **Country2Prompt**. The macro code for this example (with 3 lines of code), follows:

```
Sub Country2Prompt()  
r = Application.Run("SAPSetVariable",  
"P_VAR_COUNTRY",Worksheets("Settings").Range("E2").Value, "Key","DS_1")  
End Sub
```

There is no line break in line 2 above.

10. Test the macro by selecting France.
11. Save the workbook as a macro enabled workbook to the remote desktop with a Name of **VBA Dynamic Drop-down Workbook ##**.



Create a Dynamic Drop-down List Using Microsoft Excel VBA

Business Example

As an analyst, you often have requirements that call for VBA. Now, you need to use VBA in Analysis to create a dynamic drop-down box that will be used to provide a member value to a variable.

In this exercise, where you see the characters ##, replace them with the number your instructor assigned you.

1. Start Analysis for Microsoft Excel.
 - a) From the *Windows* Start menu, choose *All Programs* → *SAP Business Intelligence* → *SAP BusinessObjects Analysis* → *Analysis for Microsoft Excel*.
 - b) Choose the *Analysis* tab.
 - c) Choose *Insert* → *Select Data Source...*
 - d) Choose *Options* from the lower right.
2. Log on to the BI Platform via Enterprise authentication, using the following credentials:
User: **train-##**
Password: **train-##**
Authentication: **Enterprise**
 - a) Enter the credentials shown above.
 - b) Choose *OK*.
The *Select Data Source* dialog box appears.
3. Log on to the RWB BW server using the following credentials:
Client: **800**
User: **user##**
Password: **user##**
Language: Preferred language key such as **EN** for English, **DE** for German, and so on.
 - a) Double click the *RWB System Connection*.
 - b) Choose *Options*.
 - c) Enter the credentials shown above.
If you have not reset your password, use a password of **INITIAL** and reset it to **user##**.

- d) Choose *OK*.
4. Open the Analysis of Special Countries query.
 - a) Next to *Search For*, enter **Country Sales Analysis**.
 - b) Press Enter.
 - c) Locate *Analysis of Special Countries (P_QD_COUNTRY_SPEC_ANALYSIS)*.
 - d) Choose *OK*.
 - e) In the *Prompt* dialog box, choose *Germany (DE)*.
 - f) Choose *OK*.
5. Move the crosstab to D1 and turn the *Developers* tab on.
 - a) Place your cursor in cell D1.
 - b) Choose the *Components* tab on the lower right.
 - c) Expand the *Analysis of Special Countries* query.
 - d) Right click on Crosstab 1 and select *Move to*.
 - e) Choose *OK*.
 - f) From the *File* menu, choose *Options*.
 - g) In the left pane, select *Customize Ribbon*.
 - h) In the *Main Tabs* pane on the right, select *Developer*.
 - i) Choose *OK*.
6. Enter a label of Country in A2 with a column width of 20.
 - a) In cell A2, enter **Country**.
 - b) In Sheet1, right click on column B.
 - c) Choose *Column Width* and enter **20**.
 - d) Choose *OK*.
7. Add a combo box(drop-down) in B2.
 - a) Choose the *Developer* tab.
 - b) Choose *Insert*.
 - c) Under *Form Controls*, choose the *Combo Box* icon.
Your cursor turns into a cross hair that you can use to place the combo box item.
 - d) Draw a box around cell B2.
A drop-down now appears in cell B2.
8. Add information in Sheet2 that will allow communication between the drop-down and the macro, and rename Sheet2 to Settings. Enter the information as shown in the *Settings* sheet below. You also need to add the following parameters into the combo box's Form Control:

- Input range: **Settings!\$A\$2:\$A\$4**
- Cell link field: **Settings!\$D\$2**
- Drop-down lines: **3**
- In the Settings Worksheet, add the following function in cell E2: **=INDEX (B2 : B4 , D2 , 1)**



	A	B	C	D	E
1	Country Text	Country Key		Selected Index	Selected Country
2	Germany	DE			
3	USA	US			
4	France	FR			

Figure 4: Settings Sheet

- Right click on Sheet2.
- Choose *Rename*.
- Enter **Settings**.
- Enter the information in the *Settings* sheet as shown above and in the figure below.
- Choose Sheet1.
- Right click the Combo Box in cell B2 and choose *Format Control*.
- Input the *Form Control* values as shown in the figure below.



Hint:

Instead of typing the cell ranges, simply use your mouse to click on the *Settings* sheet and highlight the range you want.



Format Control	
Size	Protection
Input range:	Settings!\$A\$2:\$A\$4
Cell link:	Settings!\$D\$2
Drop down lines:	3
<input type="checkbox"/> 3-D shading	

Figure 5: Combo Box Format Control Settings for Country

- Choose *OK*.
- Choose the *Settings* worksheet.
- In cell E2 enter **=INDEX (B2 : B4 , D2 , 1)** as shown below.



Figure 6: Country Index Function

- k) Choose Sheet1.
- l) Use the drop-down to select a *Country* such as *Germany*.
- m) Choose the *Settings* sheet.



Figure 7: Settings Sheet with Germany Selected

Since Germany is chosen, the Selected Country is DE and the Selected Index is 1 (ie Germany is the 1st country in the list).

9. Use a macro with the command `SAPSetVariable` to transfer the selected value (in cell `Settings!E2`) to the prompt (`P_VAR_COUNTRY`).
- Name the macro **Country2Prompt**. The macro code for this example (with 3 lines of code), follows:

```
Sub Country2Prompt()
r = Application.Run("SAPSetVariable",
"P_VAR_COUNTRY",Worksheets("Settings").Range("E2").Value, "Key","DS_1")
End Sub
```

There is no line break in line 2 above.

- a) In Sheet1, right click the *Combo Box*.
- b) In cell B2, choose *Assign Macro*.
The *Assign Macro* dialog box displays.
- c) Under *Macro name* enter **Country2Prompt**.
- d) Choose *New*.

The Microsoft Visual Basic editor opens for Module1.

- e) Enter the 3 lines of code as shown above and in the figure below.



```

[General] Country2Prompt
Sub Country2Prompt ()
x = Application.Run("SAPSetVariable", "P_VAR_COUNTRY", Worksheets("Settings").Range("E2").Value, "Key", "DS_1")
End Sub

```

Figure 8: VBA Editor with 3 Lines of Code for the Country Prompt

The macro is contained in Module1.

- f) Keep the VBA editor open.
10. Test the macro by selecting France.
- a) Go back to Excel in Sheet1.
- b) Use the *Country* drop-down to select *France*.



Hint:

You may need to click in cell A4, for example, to get the drop-down out of edit mode.



	A	B	C	D	E	F
1						Incoming Orders EUR
2	Country	France		Country		EUR
3				FR	France	1,057,002.26
4				Overall Result		1,057,002.26
5						

Figure 9: Result of Drop-down and VBA for France

Your report now displays France.

11. Save the workbook as a macro enabled workbook to the remote desktop with a Name of **VBA Dynamic Drop-down Workbook ##**.
- a) Choose *File* → *Save As*.
- b) Choose *Desktop*.
- c) Next to *File Name*, enter the name shown above.
- d) Next to *Save as Type*, choose *Excel Macro-Enabled Workbook (*.xlsm)*.
- e) Choose *Save*.
- f) Keep the workbook open for the next exercise.

Unit 1

Exercise 2



Create a Button to Transfer Selections Using Microsoft Excel VBA

Business Example

As an analyst, your users often need to do multiple member selections. Rather than performing a data refresh after each selection, you want to postpone the refresh until all selections are made, in order to reduce the overall wait time.

In this exercise, where you see the characters ##, replace them with the number your instructor assigned you.

1. Open the workbook VBA Dynamic Drop-down Workbook ##, if necessary.
2. Turn the *Developers* tab on, if necessary.
3. Add a combo box(drop-down) in B4 with a label of Division in A4.
4. Add information in the *Settings* sheet that will allow communication between the drop-down and the macro. Use the following three parameters for the combo box's form control:
 - Input range: **Settings!\$A\$8:\$A\$10**
 - Cell link field: **Settings!\$D\$8**
 - Drop-down lines: **3**
 - In the Settings Worksheet, input the following in cell E8: **=INDEX(B8:B10,D8,1)**
 - In the Settings Worksheet, input the Division Text and Keys of Pumps 01, High Tech 07, Service 08



	A	B	C	D	E
1	Country Text	Country Key		Selected Index	Selected Country
2	Germany	DE			1 DE
3	USA	US			
4	France	FR			
5					
6					
7	Division Text	Division Key		Selected Index	Selected Division
8	Pumps	01			
9	High Tech	07			
10	Service	08			

Figure 10: Settings Sheet with Division Text and Key

5. Delete the assigned macro from the *Country* drop-down box.
6. Add a button in cell B6 with a macro to transfer the data selections. Name the new macro **Transfer_Selection**.

The macro for this example (with 6 lines of code), is as follows:

```
Sub Transfer_Selection()  
r = Application.Run("SAPSetRefreshBehaviour", "Off")  
r = Application.Run("SAPSetVariable", "P_VAR_COUNTRY",  
Worksheets("Settings").Range("E2").Value, "Key", "DS_1")  
r = Application.Run("SAPSetFilter", "DS_1", "0DIVISION",  
Worksheets("Settings").Range("E8").Value, "Key")  
End Sub
```

7. Test the button by selecting Germany and Pumps.
8. View the filter for *Division* from the *Analysis* tab on the lower right.
9. Save the workbook as a macro enabled workbook to the remote desktop with a *Name* of **VBA button to Transfer Selections Workbook ##**.



Create a Button to Transfer Selections Using Microsoft Excel VBA

Business Example

As an analyst, your users often need to do multiple member selections. Rather than performing a data refresh after each selection, you want to postpone the refresh until all selections are made, in order to reduce the overall wait time.

In this exercise, where you see the characters ##, replace them with the number your instructor assigned you.

1. Open the workbook VBA Dynamic Drop-down Workbook ##, if necessary.
 - a) Choose *File* → *Open*.
 - b) Choose the Remote Desktop.
 - c) Choose the workbook named *VBA Dynamic Drop-down Workbook ##*.
2. Turn the *Developers* tab on, if necessary.
 - a) From the *File* menu, choose *Options*.
 - b) In the left pane, choose *Customize Ribbon*.
 - c) In the *Main Tabs* pane, choose *Developer*.
 - d) Choose *OK*.
3. Add a combo box(drop-down) in B4 with a label of Division in A4.
 - a) In cell A4, enter **Division**.
 - b) Choose the *Developer* tab.
 - c) Choose *Insert*.
 - d) Under *Form Controls*, choose the *Combo Box* icon.
Your cursor turns into a cross hair that you can use to place the combo box item.
 - e) Draw a box around cell B4.
A drop-down now is displayed in cell B4.
4. Add information in the *Settings* sheet that will allow communication between the drop-down and the macro. Use the following three parameters for the combo box's form control:
 - Input range: **Settings!\$A\$8:\$A\$10**
 - Cell link field: **Settings!\$D\$8**

- Drop-down lines: 3
- In the Settings Worksheet, input the following in cell E8: **=INDEX (B8 : B10 , D8 , 1)**
- In the Settings Worksheet, input the Division Text and Keys of Pumps 01, High Tech 07, Service 08



	A	B	C	D	E
1	Country Text	Country Key		Selected Index	Selected Country
2	Germany	DE			1 DE
3	USA	US			
4	France	FR			
5					
6					
7	Division Text	Division Key		Selected Index	Selected Division
8	Pumps	01			
9	High Tech	07			
10	Service	08			

Figure 10: Settings Sheet with Division Text and Key

- Choose the *Settings* sheet.
- Starting in cell A7, enter the information in the *Settings* sheet shown in the figure above.



Caution:

If 01, 07, and 08 appear as 1, 7, and 8 set the format to text for cells B8-10.

- Choose Sheet1.
- Right click the *Combo Box* in cell B4.
- Choose *Format Control*.
- Input the *Form Control* values listed in the figure below:



Format Control	
Size	Protection
Properties	Alt Text
Control	
Input range:	Settings!\$A\$8:\$B\$10
Cell link:	Settings!\$D\$8
Drop down lines:	3
<input type="checkbox"/> 3-D shading	

Figure 11: Combo Box Format Control Settings for Division

- Choose *OK*.
- Choose the *Settings* worksheet.
- In cell E8 enter **=INDEX (B8 : B10 , D8 , 1)**.
If you get a message, continue through it.
- Choose Sheet1.
- Select a *Division* such as *High Tech*.



Hint:

You may need to click in cell A6, for example, to get the drop down out of edit mode.

l) Choose the *Settings* worksheet.



	A	B	C	D	E
7	Division Text	Division Key		Selected Index	Selected Division
8	Pumps	01		2	07
9	High Tech	07			
10	Service	08			

Figure 12: Settings Sheet with an Index Set To High Tech (07)

The selected *Division* is *High Tech*.

5. Delete the assigned macro from the *Country* drop-down box.
 - a) Go to Sheet1.
 - b) In cell B2, right click on the *Country* drop-down box.
 - c) Choose *Assign Macro*.
 - d) Under *Macro name*, use your delete key to delete the macro.

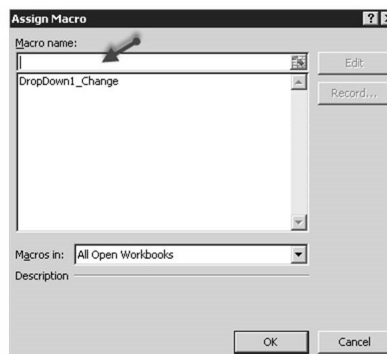


Figure 13: Country Drop-down Without a Macro

The drop-down is no longer attached to a macro.

- e) Choose OK.
6. Add a button in cell B6 with a macro to transfer the data selections.
Name the new macro **Transfer_Selection**.

The macro for this example (with 6 lines of code), is as follows:

```
Sub Transfer_Selection()
r = Application.Run("SAPSetRefreshBehaviour", "Off")
r = Application.Run("SAPSetVariable", "P_VAR_COUNTRY",
Worksheets("Settings").Range("E2").Value, "Key", "DS_1")
```

```
r = Application.Run("SAPSetFilter", "DS_1", "0DIVISION",
Worksheets("Settings").Range("E8").Value, "Key")
End Sub
```

- a) In Sheet1, choose the *Developer* tab.
- b) Choose *Insert* → *Button*.

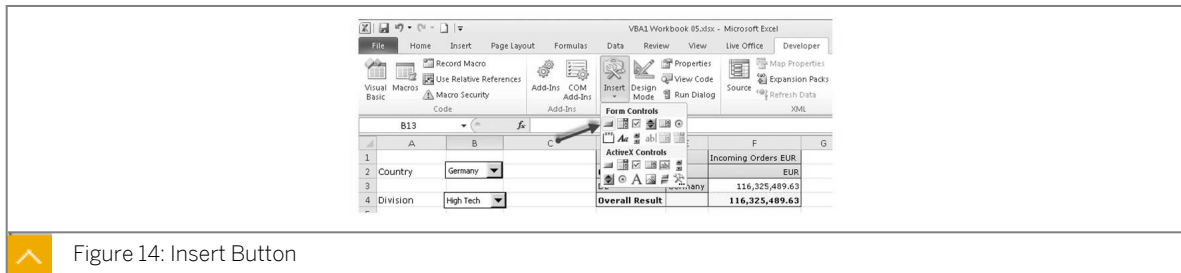


Figure 14: Insert Button

- c) Place the button in cell B6:C6.
The *Assign Macro* window opens.
- d) Under *Macro name*, enter **Transfer_Selection**.
- e) Choose *New*.
- f) Modify your code as shown.

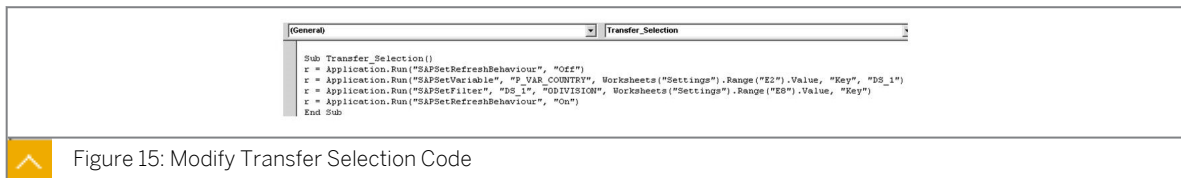


Figure 15: Modify Transfer Selection Code

- g) Go back to Excel.
 - h) Left click the button in B6 and enter a description of **Transfer Selections**.
7. Test the button by selecting Germany and Pumps.
 - a) Use the *Country* drop-down to select Germany.
Your report is not affected.
 - b) Use the *Division* drop-down to select Pumps.
Your report is not affected.
 - c) Press the *Transfer Selections* button.
Your report now displays DE (Germany) in the rows and is restricted to the Pumps Division.
 8. View the filter for *Division* from the *Analysis* tab on the lower right.
 - a) Select the *Analysis* tab on the lower right.
 - b) Expand *Division* in the *Background Filter* window.

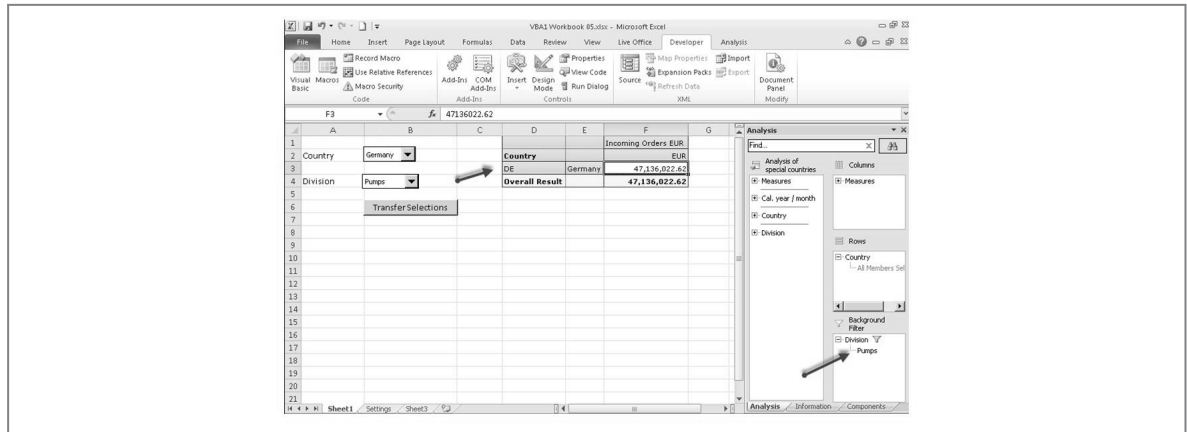


Figure 16: Result of Using the Transfer Selections Button

Division is restricted to Pumps.

9. Save the workbook as a macro enabled workbook to the remote desktop with a Name of **VBA button to Transfer Selections Workbook ##**.
 - a) Choose *Desktop*.
 - b) Next to *File Name*, enter the name shown above.
 - c) Next to *Save as Type*, choose *Excel Macro-Enabled Workbook(*.xlsm)*.
 - d) Choose *Save*.
 - e) Keep *Analysis* open for the next exercise.

Unit 1

Exercise 3



Using Microsoft Excel VBA with Check Boxes for Filtering (Optional)

Business Example

As an analyst, you know users spend a lot of time waiting for the system to refresh while selecting members. In this scenario the user needs checkboxes to select one or more sales organizations and they do not want a system refresh until they have finalized their selections. You decide to add a button and macro to turn off the system refresh, determine the sales organizations selected, and then refresh the data.

In this exercise, where you see the characters ##, replace them with the number your instructor assigned you.

1. Start Analysis for Microsoft Excel.
2. Log on to the BI Platform via Enterprise authentication, using the following credentials:
User: **train-##**
Password: **train-##**
Authentication: **Enterprise**
3. Log on to the RWB BW server using the following credentials:
Client: **800**
User: **user##**
Password: **user##**
Language: Preferred language key such as **EN** for English, **DE** for German, and so on.
4. Open the Country Sales Analysis query.
5. Move the crosstab to D1.
6. Remove *Sold-toPartyCountry* from the *Rows*, add *Sales Organization* into the *Rows*.
7. Turn the *Developers* tab on, if necessary.
8. Add four check boxes and label them as shown in the figure below.



Hint:

The checkboxes do not have to be in the exact cells as shown in the figure below.



	A	B	C	D	E	F
1	<input type="checkbox"/> Frankfurt					Sales Volume EUR
2	<input type="checkbox"/> Denver			Sales Organization		EUR
3	<input type="checkbox"/> Rotterdm			1000	Germany Frankfurt	14,217,648.73
4	<input type="checkbox"/> Milan			2200	France, Paris	1,057,002.26
5				2400	Italy, Milan	10,785,700.00
6				2500	Netherlands/Rotterdm	15,872,750.00
7				3000	USA Philadelphia	7,462,291.61
8				3020	USA Denver	15,240,181.94
9				R300	Retail USA	23,280,974.21
10				S300	Services USA	0.00
11				Overall Result		87,916,548.75

Figure 17: Workbook with Four Checkboxes

- Re-name Sheet2 to Settings. Add information in the Settings sheet that will allow communication between the checkbox and the macro as shown below.

C9 Cell Value: **=COUNTIF (C4 : C7 , "TRUE")**



	A	B	C
1			
2			
3		Sales Org	Selected
4	Frankfurt	1000	
5	Denver	3020	
6	Rotterdm	2500	
7	Milan	2400	
8			
9			0

Figure 18: Settings Sheet for Check Boxes

- In the format control for each checkbox, configure the cell link as shown below.
 - Frankfurt: **Settings!\$C\$4**
 - Denver: **Settings!\$C\$5**
 - Rotterdm: **Settings!\$C\$6**
 - Milan: **Settings!\$C\$7**
- Do a preliminary test of the check boxes for Frankfurt and Milan. In the Settings sheet you should see the following:



	A	B	C
1			
2			
3		Sales Org	Selected
4	Frankfurt	1000	TRUE
5	Denver	3020	
6	Rotterdam	2500	
7	Milan	2400	TRUE
8			
9			2

Figure 19: Preliminary Test with Frankfurt and Milan Selected

12. In Sheet1, add a button in A6 and A7 called Create Filter with a macro that will identify the selected members separated by a ";", and transfer them via the SAPSetFilter command. The macro code is shown below:

```
Sub Create_Filter()
Dim Act_true As Integer
Dim Number_true As Integer
Dim Filter_Value As String
For i = 4 To 7
If Worksheets("Settings").Cells(i, 3).Value = "True" Then Act_true =
Act_true + 1 Filter_Value = Filter_Value &
Worksheets("Settings").Cells(i, 2).Value
If Act_true < Number_true Then Filter_Value = Filter_Value & ";"
End If
Next
ret = Application.Run("SAPSetFilter", "DS_1", "0SALESORG", Filter_Value,
"Input_String")
End Sub
```

13. Test the macro by selecting *Denver* and *Rotterdam*.
14. Save the workbook as a macro enabled workbook to the remote desktop with a Name of **VBA Checkboxes Workbook ##**.



Using Microsoft Excel VBA with Check Boxes for Filtering (Optional)

Business Example

As an analyst, you know users spend a lot of time waiting for the system to refresh while selecting members. In this scenario the user needs checkboxes to select one or more sales organizations and they do not want a system refresh until they have finalized their selections. You decide to add a button and macro to turn off the system refresh, determine the sales organizations selected, and then refresh the data.

In this exercise, where you see the characters ##, replace them with the number your instructor assigned you.

1. Start Analysis for Microsoft Excel.
 - a) From the *Windows Start* menu, choose *All Programs* → *SAP Business Intelligence* → *SAP BusinessObjects Analysis* → *Analysis for Microsoft Excel*.
 - b) Choose the *Analysis* tab.
 - c) Place your cursor in cell D1.
 - d) Choose *Insert* → *Select Data Source...*
 - e) Choose *Options* from the lower right.
2. Log on to the BI Platform via Enterprise authentication, using the following credentials:
User: **train-##**
Password: **train-##**
Authentication: **Enterprise**
 - a) Enter the credentials shown above.
 - b) Choose *OK*.
The *Select Data Source* dialog box appears.
3. Log on to the RWB BW server using the following credentials:
Client: **800**
User: **user##**
Password: **user##**
Language: Preferred language key such as **EN** for English, **DE** for German, and so on.
 - a) Double click the *RWB System Connection*.

- b) Choose *Options*.
 - c) Enter the credentials shown above.
If you have not reset your password, use a password of **INITIAL** and reset it to **user##**.
 - d) Choose *OK*.
4. Open the Country Sales Analysis query.
 - a) Next to *Search For*, enter **Country Sales Analysis**.
 - b) Press Enter.
 - c) Locate *Country Sales Analysis(P_QD_COUNTRY_SALES)*.
 - d) Choose *OK*.
5. Move the crosstab to D1.
 - a) Place your cursor in cell D1.
 - b) Choose the *Components* tab on the lower right.
 - c) Expand the *Country Sales Analysis* query.
 - d) Right click on Crosstab 1.
 - e) Choose *Move to*.
 - f) Choose *OK*.
6. Remove *Sold-toPartyCountry* from the *Rows*, add *Sales Organization* into the *Rows*.
 - a) Choose the *Analysis* tab on the lower right.
 - b) Drag *Sold-toPartyCountry* out of the *Rows*.
 - c) Drag *Sales Organization* into the *Rows*.
7. Turn the *Developers* tab on, if necessary.
 - a) From the *File* menu, choose *Options*.
 - b) In the left pane, select *Customize Ribbon*.
 - c) In the *Main Tabs* pane on the right, select *Developer*.
 - d) Choose *OK*.
8. Add four check boxes and label them as shown in the figure below.



Hint:

The checkboxes do not have to be in the exact cells as shown in the figure below.



	A	B	C	D	E	F
1	<input type="checkbox"/> Frankfurt					Sales Volume EUR
2	<input type="checkbox"/> Denver			Sales Organization		EUR
3	<input type="checkbox"/> Rotterdm			1000	Germany Frankfurt	14,217,648.73
4	<input type="checkbox"/> Milan			2200	France, Paris	1,057,002.26
5				2400	Italy, Milan	10,785,700.00
6				2500	Netherlands/Rotterdm	15,872,750.00
7				3000	USA Philadelphia	7,462,291.61
8				3020	USA Denver	15,240,181.94
9				R300	Retail USA	23,280,974.21
10				S300	Services USA	0.00
11				Overall Result		87,916,548.75

Figure 17: Workbook with Four Checkboxes

- a) Choose the *Developer* tab.
- b) Choose *Insert*.
- c) Choose the *Check Box* icon.
Your cursor turns into a cross hair that you can use to place the check box.
- d) Draw a box around cell A1.



Hint:
Just after dropping the checkbox, simply type the name of the city since you are in edit mode.

A drop-down now appears in cell A1 and is ready for input.

- e) Enter **Frankfurt**.
 - f) Repeat steps b, through e for **Denver**, **Rotterdm**, and **Milan**.
9. Re-name Sheet2 to Settings. Add information in the Settings sheet that will allow communication between the checkbox and the macro as shown below.
C9 Cell Value: **=COUNTIF (C4 : C7 , "TRUE")**



	A	B	C
1			
2			
3		Sales Org	Selected
4	Frankfurt	1000	
5	Denver	3020	
6	Rotterdm	2500	
7	Milan	2400	
8			
9			0

Figure 18: Settings Sheet for Check Boxes

- a) In cell C9 enter the function shown above.

- b) Right click on Sheet2.
- c) Choose *Rename*.
- d) Enter **Settings**.
- e) Choose *OK*.
- f) Enter the information in the *Settings* sheet as shown in the figure above.
- g) In cell C9 enter **=COUNTIF (C4 ; C7 , "TRUE")**.



Hint:
Continue through any Excel messages.

10. In the format control for each checkbox, configure the cell link as shown below.

- Frankfurt: **Settings!\$C\$4**
- Denver: **Settings!\$C\$5**
- Rotterdam: **Settings!\$C\$6**
- Milan: **Settings!\$C\$7**

- a) Go to Sheet1.
- b) Right click on the *Frankfurt* checkbox in cell A1.
- c) Choose *Format Control*.
- d) Choose the *Control* tab.
- e) Next to *Cell link*, enter the cell reference shown above.



Hint:
Instead of typing, simply left click in the field next to Cell Link and then move your cursor to the *Settings* sheet cell C4 and press enter.

- f) Repeat step a, through d for the *Denver*, *Rotterdam*, and *Milan* check boxes using the cell references shown above.

11. Do a preliminary test of the check boxes for Frankfurt and Milan. In the *Settings* sheet you should see the following:



	A	B	C
1			
2			
3		Sales Org	Selected
4	Frankfurt	1000	TRUE
5	Denver	3020	
6	Rotterdam	2500	
7	Milan	2400	TRUE
8			
9			2

Figure 19: Preliminary Test with Frankfurt and Milan Selected

a) In Sheet1, select *Frankfurt* and *Milan*.

In the *Settings* sheet, you should see the result shown in the figure above.

12. In Sheet1, add a button in A6 and A7 called Create Filter with a macro that will identify the selected members separated by a ";", and transfer them via the SAPSetFilter command. The macro code is shown below:

```
Sub Create_Filter()
Dim Act_true As Integer
Dim Number_true As Integer
Dim Filter_Value As String
For i = 4 To 7
If Worksheets("Settings").Cells(i, 3).Value = "True" Then Act_true =
Act_true + 1 Filter_Value = Filter_Value &
Worksheets("Settings").Cells(i, 2).Value
If Act_true < Number_true Then Filter_Value = Filter_Value & ";"
End If
Next
ret = Application.Run("SAPSetFilter", "DS_1", "0SALESORG", Filter_Value,
"Input_String")
End Sub
```

a) In Sheet1, choose the *Developer* tab.

b) Choose *Insert* → *Button*.

c) Place the button in cell A6 and A7. The *Assign Macro* window opens.

d) Under *Macro name*, enter **Create_Filter**.

e) Choose *New*.

f) Enter the macro code shown above and in the figure below, and keep the VBA editor open when finished.



```

[General] Create_Filter
Sub Create_Filter()
Dim Act_true As Integer
Dim Number_true As Integer
Dim Filter_Value As String
Number_true = Worksheets("Settings").Range("C9").Value

For i = 4 To 7
If Worksheets("Settings").Cells(i, 3).Value = "True" Then
Act_true = Act_true + 1
Filter_Value = Filter_Value & Worksheets("Settings").Cells(i, 2).Value
If Act_true < Number_true Then Filter_Value = Filter_Value & ";"
End If
Next
ret = Application.Run("SAPSetFilter", "DS_1", "OSALESORG", Filter_Value, "Input_String")
End Sub

```

Figure 20: VBA Code for Check Boxes

- g) Go back to Sheet1 in Excel.
 - h) In the button, enter a text of **Create Filter**.
13. Test the macro by selecting *Denver* and *Rotterdam*.
- a) Select only *Denver* and *Rotterdam*.
 - b) Press the *Create Filter* button.



	A	B	C	D	E	F
1	<input type="checkbox"/> Frankfurt					Sales Volume EUR
2	<input checked="" type="checkbox"/> Denver			Sales Organization		EUR
3	<input checked="" type="checkbox"/> Rotterdam			3020	USA Denver	15,240,181.94
4	<input type="checkbox"/> Milan			2500	Netherlands/Rotterdam	15,872,750.00
5				Overall Result		31,112,931.94
6						
7						

Figure 21: Result of Check Boxes and Button

Your report now displays *Denver* and *Rotterdam*.

14. Save the workbook as a macro enabled workbook to the remote desktop with a *Name* of **VBA Checkboxes Workbook ##**.
- a) Choose *File* → *Save As*.
 - b) Choose *Desktop*.
 - c) Next to *File Name*, enter the name shown above.
 - d) Next to *Save as Type*, choose *Excel Macro-Enabled Workbook(*.xlsm)*.
 - e) Choose *Save*.
 - f) Choose *File* → *Exit*.



LESSON SUMMARY

You should now be able to:

- Extend workbook display options with functions and Microsoft Excel VBA



39

Learning Assessment

1. *Choose the correct answers.*

A



Learning Assessment - Answers

1. *Choose the correct answers.*

A