**Module 12: Arrays & Collections**

* **Declaring and Using Arrays**

An Array is a group of variables that share the same name and data type, used to store multiple values.

Example 1: Fixed Array

Sub FixedArrayExample()

Dim scores(1 To 5) As Integer

Dim i As Integer

'Assign values

For i = 1 To 5

scores(i) = i \* 10

Next i

'Print values in Immediate Window (Ctrl + G to view)

For i = 1 To 5

Debug.Print "Score " & i & ": " & scores(i)

Next i

End Sub

📘 *Explanation:*

* Dim scores(1 To 5) → Creates an array of 5 integers.
* Loops are used to assign and print values.
* **Dynamic Arrays with ReDim**

**Definition:**  
When you don’t know the number of elements in advance, use a **dynamic array**.  
Use ReDim to set or resize the array at runtime.

Example 2: Dynamic Array

Sub DynamicArrayExample()

Dim students() As String

Dim count As Integer

Dim i As Integer

count = InputBox("Enter number of students:")

ReDim students(1 To count)

For i = 1 To count

students(i) = InputBox("Enter student name " & i & ":")

Next i

Debug.Print "List of Students:"

For i = 1 To count

Debug.Print students(i)

Next i

End Sub

📘 *Explanation:*

* The size of the array is decided during execution.
* ReDim changes the array size dynamically.

Preserve Keyword (Retain old data)

ReDim Preserve students(1 To count + 1)

📘 Use when you want to expand the array and keep existing values.

* **Using Collections**

**Definition:**  
A **Collection** is an object that stores related items (similar to a list).  
It automatically resizes and stores mixed data types.

Example 3: Collection Example

Sub CollectionExample()

Dim fruits As New Collection

Dim item As Variant

'Add items

fruits.Add "Apple"

fruits.Add "Banana"

fruits.Add "Cherry"

'Access items

Debug.Print "First fruit: " & fruits(1)

'Loop through collection

For Each item In fruits

Debug.Print item

Next item

'Remove item

fruits.Remove 2

Debug.Print "After removing 2nd item:"

For Each item In fruits

Debug.Print item

Next item

End Sub

📘 Collections are easy to use and flexible.

* **Using Dictionary (from Scripting Library)**

**Definition:**  
A **Dictionary** is like a Collection but with **key–value pairs** (like Python’s dictionary).

To use it, go to **VBA Editor → Tools → References → check “Microsoft Scripting Runtime”**.

Example 4: Dictionary Example

Sub DictionaryExample()

Dim dict As Object

Dim key As Variant

Set dict = CreateObject("Scripting.Dictionary")

'Add items

dict.Add "Amit", 50000

dict.Add "Ravi", 45000

dict.Add "Priya", 55000

'Access value

Debug.Print "Amit's salary: " & dict("Amit")

'Change value

dict("Amit") = 60000

'Loop through keys

For Each key In dict.Keys

Debug.Print key & " earns " & dict(key)

Next key

'Check if key exists

If dict.Exists("Ravi") Then

Debug.Print "Ravi found in dictionary"

End If

End Sub

📘 Dictionaries are powerful for lookups, unique lists, and fast data access.

**Real life project:- Employee Salary Manager**

**🎯 Goal:**

Manage employee data (Name, Department, Salary) using:

* **Array** → store initial records
* **Collection** → store new employees dynamically
* **Dictionary** → lookup salaries quickly

**⚙️ Setup in Excel**

1. Open a **new Excel file**.
2. Rename **Sheet1** to EmployeeData.
3. In row 1, type headers:

| **A** | **B** | **C** |
| --- | --- | --- |
| Employee Name | Department | Salary |

1. Press **Alt + F11** → open **VBA Editor**.
2. Insert → **Module** → paste the full code below 👇

Sub EmployeeSalaryManager()

' --------------------------------------

' Employee Salary Manager Project

' --------------------------------------

Dim employees(1 To 5, 1 To 3) As Variant

Dim i As Integer

Dim ws As Worksheet

Dim empCollection As New Collection

Dim empDict As Object

Set empDict = CreateObject("Scripting.Dictionary")

Set ws = ThisWorkbook.Sheets("EmployeeData")

ws.Cells.Clear

ws.Range("A1:C1").Value = Array("Employee Name", "Department", "Salary")

' --------------------------------------

' 1?? Using ARRAY – initial records

' --------------------------------------

employees(1, 1) = "Amit": employees(1, 2) = "IT": employees(1, 3) = 55000

employees(2, 1) = "Ravi": employees(2, 2) = "HR": employees(2, 3) = 48000

employees(3, 1) = "Priya": employees(3, 2) = "Finance": employees(3, 3) = 60000

employees(4, 1) = "Sita": employees(4, 2) = "Sales": employees(4, 3) = 52000

employees(5, 1) = "John": employees(5, 2) = "Marketing": employees(5, 3) = 50000

' Write Array data to sheet

For i = 1 To 5

ws.Cells(i + 1, 1).Value = employees(i, 1)

ws.Cells(i + 1, 2).Value = employees(i, 2)

ws.Cells(i + 1, 3).Value = employees(i, 3)

Next i

' --------------------------------------

' 2?? Using COLLECTION – add new employees

' --------------------------------------

empCollection.Add Array("Raj", "IT", 58000)

empCollection.Add Array("Neha", "HR", 49000)

Dim emp

For Each emp In empCollection

ws.Cells(ws.Rows.count, 1).End(xlUp).Offset(1, 0).Resize(1, 3).Value = emp

Next emp

' --------------------------------------

' 3?? Using DICTIONARY – salary lookup

' --------------------------------------

Dim lastRow As Long

Dim empName As String, empSalary As Double

lastRow = ws.Cells(ws.Rows.count, 1).End(xlUp).Row

' Add all employees to Dictionary

For i = 2 To lastRow

empName = ws.Cells(i, 1).Value

empSalary = ws.Cells(i, 3).Value

empDict(empName) = empSalary

Next i

' Ask user for lookup

Dim searchName As String

searchName = InputBox("Enter employee name to check salary:")

If empDict.Exists(searchName) Then

MsgBox searchName & "'s salary is ?" & empDict(searchName), vbInformation, "Salary Lookup"

Else

MsgBox "Employee not found!", vbExclamation, "Lookup Error"

End If

' --------------------------------------

' 4?? Summary Output

' --------------------------------------

MsgBox "? Employee Salary Manager Completed!" & vbCrLf & \_

"• Initial data from Array" & vbCrLf & \_

"• New employees via Collection" & vbCrLf & \_

"• Salary lookup using Dictionary", vbInformation

End Sub

**🧩 Concept Integration**

| **Concept** | **Purpose in Project** |
| --- | --- |
| **Array** | Stored base data (fixed-size) |
| **Collection** | Added new employees dynamically |
| **Dictionary** | Quick salary lookup (key = name) |

**Real-life, professional-grade Excel VBA project** that actually uses **Arrays + Collections + Dictionary**

**roject Title: Sales Performance Dashboard Automation**

**🎯 Goal**

Build an **Excel VBA automation tool** that:

* Imports sales data from Excel (using **Arrays**)
* Groups and summarizes totals by region & salesperson (using **Dictionary**)
* Automatically generates a **Top Performer Report** (using **Collections**)

**Business Scenario**

You are managing a sales file with thousands of records like this:

| **Date** | **Salesperson** | **Region** | **Amount** |
| --- | --- | --- | --- |
| 01-Jan-2025 | Amit | North | 25000 |
| 01-Jan-2025 | Ravi | East | 18000 |
| 02-Jan-2025 | Neha | West | 21000 |
| 03-Jan-2025 | Amit | North | 30000 |

* 1. Calculate **total sales by salesperson and region**
  2. Find the **top 3 performers**
  3. Generate a summary report automatically on a new sheet

**⚙️ Setup in Excel**

1. Open a **new Excel workbook**.
2. Create a sheet named SalesData.
3. Add headers in row 1:

| **A** | **B** | **C** | **D** |
| --- | --- | --- | --- |
| Date | Salesperson | Region | Amount |

Add some sample data (10–20 rows).

Press **Alt + F11** → open **VBA Editor**, Insert → **Module**.

Paste the full code below 👇

Option Explicit

Sub SalesPerformanceDashboard()

Dim wsData As Worksheet, wsReport As Worksheet

Dim salesArray As Variant, resultArray() As Variant

Dim salesDict As Object

Dim i As Long, j As Long, lastRow As Long

Dim key As Variant

Dim region As String, rep As String

Dim totalKey As String, amount As Double

Dim tempRegion As String, tempRep As String, tempTotal As Double

Application.ScreenUpdating = False

'----------------------------------

' 1?? Load data into ARRAY

'----------------------------------

On Error Resume Next

Set wsData = ThisWorkbook.Sheets("SalesData")

On Error GoTo 0

If wsData Is Nothing Then

MsgBox "? Sheet 'SalesData' not found!", vbCritical

Exit Sub

End If

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

If lastRow < 2 Then

MsgBox "? No sales data found!", vbExclamation

Exit Sub

End If

salesArray = wsData.Range("A2:D" & lastRow).Value

'----------------------------------

' 2?? Use DICTIONARY to group totals

'----------------------------------

Set salesDict = CreateObject("Scripting.Dictionary")

For i = 1 To UBound(salesArray, 1)

rep = Trim(salesArray(i, 2))

region = Trim(salesArray(i, 3))

amount = CDbl(salesArray(i, 4))

totalKey = region & "|" & rep

If salesDict.Exists(totalKey) Then

salesDict(totalKey) = salesDict(totalKey) + amount

Else

salesDict(totalKey) = amount

End If

Next i

'----------------------------------

' 3?? Transfer to 2D ARRAY for sorting

'----------------------------------

ReDim resultArray(1 To salesDict.count, 1 To 3)

i = 0

For Each key In salesDict.Keys

i = i + 1

resultArray(i, 1) = Split(key, "|")(0) ' Region

resultArray(i, 2) = Split(key, "|")(1) ' Salesperson

resultArray(i, 3) = salesDict(key) ' Total Sales

Next key

'----------------------------------

' 4?? Sort the 2D ARRAY (Descending by Total Sales)

'----------------------------------

For i = 1 To UBound(resultArray, 1) - 1

For j = i + 1 To UBound(resultArray, 1)

If resultArray(i, 3) < resultArray(j, 3) Then

tempRegion = resultArray(i, 1)

tempRep = resultArray(i, 2)

tempTotal = resultArray(i, 3)

resultArray(i, 1) = resultArray(j, 1)

resultArray(i, 2) = resultArray(j, 2)

resultArray(i, 3) = resultArray(j, 3)

resultArray(j, 1) = tempRegion

resultArray(j, 2) = tempRep

resultArray(j, 3) = tempTotal

End If

Next j

Next i

'----------------------------------

' 5?? Output report

'----------------------------------

On Error Resume Next

Set wsReport = ThisWorkbook.Sheets("SalesReport")

On Error GoTo 0

If wsReport Is Nothing Then

Set wsReport = ThisWorkbook.Sheets.Add

wsReport.Name = "SalesReport"

Else

wsReport.Cells.Clear

End If

wsReport.Range("A1:C1").Value = Array("Region", "Salesperson", "Total Sales")

wsReport.Range("A1:C1").Font.Bold = True

wsReport.Range("A2").Resize(UBound(resultArray, 1), 3).Value = resultArray

wsReport.Columns("A:C").AutoFit

wsReport.Range("A2:C4").Interior.Color = RGB(198, 239, 206)

wsReport.Range("A2:C4").Font.Bold = True

'----------------------------------

' 6?? Dashboard message

'----------------------------------

MsgBox "? Sales Dashboard Generated!" & vbCrLf & \_

"Total Records: " & (lastRow - 1) & vbCrLf & \_

"Unique Salespersons: " & salesDict.count & vbCrLf & \_

"Top Performer: " & resultArray(1, 2) & " (" & resultArray(1, 1) & ")", vbInformation

Application.ScreenUpdating = True

End Sub

**How It Works**

| **Step** | **Component** | **Purpose** |
| --- | --- | --- |
| 1 | **Array** | Loads all Excel data into memory for fast processing |
| 2 | **Dictionary** | Groups sales totals by region + salesperson |
| 3 | **Collection** | Stores results for sorting and final report |
| 4 | **Sorting** | Manual sort of collection to get top performers |
| 5 | **Report Sheet** | Generates clean summary table automatically |

**Output Example (on “SalesReport” sheet)**

| **Region** | **Salesperson** | **Total Sales** |
| --- | --- | --- |
| North | Amit | 55000 |
| West | Neha | 49000 |
| East | Ravi | 46000 |
| South | Priya | 38000 |

✅ Top 3 performers highlighted in green.

**🧩 What You Learn**

* Use of **Arrays** for fast Excel data handling
* **Dictionary** for grouping and aggregation (like SQL “GROUP BY”)
* **Collections** for sorting and dynamic lists

**Real-world version of the Sales Dashboard** with a **UserForm Filter**.

**🎯 Goal**

You’ll build a **Sales Dashboard Tool** that lets you:

✅ Select a **Region** from a dropdown (auto-filled from data)  
✅ Instantly view all **Salespersons + Total Sales** for that region  
✅ Click a button to **generate a report sheet**

**🧩 Step 1: Create the UserForm**

1. Press **Alt + F11** → open VBA editor
2. Go to **Insert → UserForm**
3. Set these properties (in the Properties window below the Project Explorer):

| **Property** | **Value** |
| --- | --- |
| Name | frmSalesDashboard |
| Caption | Sales Dashboard Filter |
| Step 2: Add Controls to the Form   |  |  |  | | --- | --- | --- | | **Control** | **Name** | **Caption / Purpose** | | **ComboBox** | cmbRegion | for selecting region | | **ListBox** | lstSales | to display salesperson + sales | | **CommandButton** | btnGenerate | Generate Report | | **CommandButton** | btnClose | Close | | **Label** | (optional) | “Select Region” (above ComboBox) |   **⚙️ Step 3: Add Code Behind the Form**  Double-click the **UserForm** and paste this code inside:  Option Explicit  Private salesDict As Object  Private regionList As Collection  '---------------------------------  ' Load event - populate dropdown  '---------------------------------  Private Sub UserForm\_Initialize()  Dim ws As Worksheet  Dim dataArr As Variant  Dim lastRow As Long  Dim i As Long  Dim region As String  Dim rep As String  Dim amt As Double  Dim key As String    Set ws = ThisWorkbook.Sheets("SalesData")  lastRow = ws.Cells(ws.Rows.count, "A").End(xlUp).Row  dataArr = ws.Range("A2:D" & lastRow).Value    ' Dictionary for totals  Set salesDict = CreateObject("Scripting.Dictionary")  Set regionList = New Collection    On Error Resume Next  For i = 1 To UBound(dataArr, 1)  region = Trim(dataArr(i, 3))  rep = Trim(dataArr(i, 2))  amt = CDbl(dataArr(i, 4))    key = region & "|" & rep    If salesDict.Exists(key) Then  salesDict(key) = salesDict(key) + amt  Else  salesDict(key) = amt  End If    ' Build unique region list  regionList.Add region, region  Next i  On Error GoTo 0    ' Populate Region ComboBox  For i = 1 To regionList.count  cmbRegion.AddItem regionList(i)  Next i  End Sub  '---------------------------------  ' When Region selected  '---------------------------------  Private Sub cmbRegion\_Change()  Dim key As Variant  Dim arr() As String  Dim reg As String    lstSales.Clear  reg = cmbRegion.Value    For Each key In salesDict.Keys  arr = Split(key, "|")  If arr(0) = reg Then  lstSales.AddItem arr(1) & " - " & Format(salesDict(key), "#,##0")  End If  Next key  End Sub  '---------------------------------  ' Generate report button  '---------------------------------  Private Sub btnGenerate\_Click()  Dim ws As Worksheet  Dim region As String  Dim key As Variant  Dim arr() As String  Dim rowNum As Long    If cmbRegion.ListIndex = -1 Then  MsgBox "Please select a region first!", vbExclamation  Exit Sub  End If    region = cmbRegion.Value    On Error Resume Next  Set ws = ThisWorkbook.Sheets("RegionReport")  On Error GoTo 0    If ws Is Nothing Then  Set ws = ThisWorkbook.Sheets.Add  ws.Name = "RegionReport"  Else  ws.Cells.Clear  End If    ws.Range("A1:C1").Value = Array("Region", "Salesperson", "Total Sales")  ws.Range("A1:C1").Font.Bold = True    rowNum = 2  For Each key In salesDict.Keys  arr = Split(key, "|")  If arr(0) = region Then  ws.Cells(rowNum, 1).Value = arr(0)  ws.Cells(rowNum, 2).Value = arr(1)  ws.Cells(rowNum, 3).Value = salesDict(key)  rowNum = rowNum + 1  End If  Next key    ws.Columns("A:C").AutoFit  MsgBox "? Report generated for region: " & region, vbInformation  End Sub  Private Sub btnClose\_Click()  Unload Me  End Sub  **Step 4: Add a Shortcut Macro**  In a **standard VBA module** (not the form), add this simple launcher:  Sub OpenSalesDashboard()  frmSalesDashboard.Show  End Sub  Now in Excel:  Press Alt + F8  Run OpenSalesDashboard  ✅ You’ll get your live dashboard form! |  |

**🖼️ Final Output**

* Dropdown shows all unique regions (auto-filled)
* When you select a region → the list updates instantly
* Clicking **Generate Report** creates a filtered sheet with that region’s totals

**🧠 Skills You’ve Just Combined**

✔ Arrays (loaded sales data)  
✔ Dictionary (grouping totals)  
✔ Collection (unique regions)  
✔ UserForm (interactive filter UI)  
✔ Dynamic Report Creation