Excel Assignment - 20

1. Write a VBA code to select the cells from A5 to C10. Give it a name “Data Analytics” and fill the cells with the following cells “This is Excel VBA”

Number	Odd or even
56	
89	
26	
36	
75	
48	
92	
58	
13	
25	

Ans.

```
Sub FillDataAnalyticsRange()
   With Range("A5:C10")
      .Name = "Data Analytics" ' Name the range as "Data Analytics"
      .Value = Array("This is Excel VBA", "Number", "Odd or", "even", 56, 89, 26, 36, 75, 48, 92, 58,
```

```
13, 25)
   End With
End Sub
```

2. Use the above data and write a VBA code using the following statements to display in the next column if the number is odd or even

a. IF ELSE statement
b. Select Case statement
c. For the Next Statement

Ans.

a. IF ELSE statement:

```
Sub OddEven_IFELSE()
   Dim rng As Range
   Dim cell As Range

   Set rng = Range("A5:A16") ' Assuming the data is in column A

   For Each cell In rng
      If IsNumeric(cell.Value) Then
         If cell.Value Mod 2 = 0 Then
            cell.Offset(0, 1).Value = "Even"
         Else
            cell.Offset(0, 1).Value = "Odd"
         End If
      End If
   Next cell
End Sub
```

b. Select Case Statement:

```
Sub OddEven_SelectCase()
   Dim rng As Range
   Dim cell As Range

   Set rng = Range("A5:A16") ' Assuming the data is in column A

   For Each cell In rng
      If IsNumeric(cell.Value) Then
```

```
            Select Case cell.Value Mod 2
                Case 0
                    cell.Offset(0, 1).Value = "Even"
                Case 1
                    cell.Offset(0, 1).Value = "Odd"
            End Select
        End If
    Next cell
End Sub
```

C. For-Next Loop:

```
Sub OddEven_ForNext()
    Dim rng As Range
    Dim cell As Range
    Dim lastRow As Long

    Set rng = Range("A5:A16") ' Assuming the data is in column A
    lastRow = rng.Rows.Count + rng.Row - 1

    For i = rng.Row To lastRow
        Set cell = Cells(i, rng.Column)
        If IsNumeric(cell.Value) Then
            If cell.Value Mod 2 = 0 Then
                cell.Offset(0, 1).Value = "Even"
            Else
                cell.Offset(0, 1).Value = "Odd"
            End If
        End If
    Next i
End Sub
```

3. What types of errors do you usually see in VBA? 4. How

do you handle Runtime errors in VBA?

Ans. Common types of errors in VBA include syntax errors, runtime errors,

logical errors, and object errors. You can use error handling techniques such as On Error Resume Next and On Error GoTo [label] to handle runtime errors. The Err object provides information about the error, which you can use to display custom messages or take appropriate actions.

5. Write some good practices to be followed by VBA users for handling errors

Ans.

- Use error handling routines like On Error GoTo [label] or Error Resume Next.
- Provide specific error messages using Err. Number and Err.Description.
- Utilize debugging tools to identify and troubleshoot errors.
- Validate input and data to prevent errors caused by invalid data.
- Handle expected errors with conditional statements or error-handling blocks.
- Implement error logging to track and analyze runtime errors.
- Thoroughly test and review code for potential errors before deployment.
- Document error handling approach and known issues.

6. What is UDF? Why are UDF's used? Create a UDF to multiply 2 numbers in VBA

Ans.

UDF stands for User-Defined Function. UDFs are used to create custom functions in VBA for specific calculations or tasks not available in Excel's built-in functions. Here's a short example of a UDF in VBA that multiplies two numbers:

```
Function MultiplyNumbers(num1 As Double, num2 As Double) As Double
    MultiplyNumbers = num1 * num2
End Function
```

You can use this UDF in Excel by typing =MultiplyNumbers(A1, B1) in a cell, where A1 and B1 are the cell references containing the numbers you want to multiply.


iNeuron