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”

Ans: VBA code to select cells A5 to C10, name the range "Data Analytics" and fill the cells with the text "This is Excel VBA":

```
Sub SetRange()

    Range("A5:C10").Select

    Selection.Name = "Data Analytics"

    Selection.Value = "This is Excel VBA"

End Sub
```

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

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 Next Statement

Ans: VBA code to display if a number is odd or even for the data in the next column, using different control structures:

a. **IF ELSE statement:**

```
Sub OddOrEvenIfElse()
  Dim currentCell As Range
  For Each currentCell In Range("D2:D11")
    If currentCell.Value Mod 2 = 0 Then
      currentCell.Offset(0, 1).Value = "Even"
    Else
      currentCell.Offset(0, 1).Value = "Odd"
    End If
  Next currentCell
End Sub
```

b. **Select Case statement:**

```
Sub OddOrEvenSelectCase()
  Dim currentCell As Range
  For Each currentCell In Range("D2:D11")
```

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

c. **For Next statement:**

```
    Sub OddOrEvenForNext()
      Dim i As Long
      For i = 2 To 11
        If Cells(i, 4).Value Mod 2 = 0 Then
          Cells(i, 5).Value = "Even"
        Else
          Cells(i, 5).Value = "Odd"
        End If
      Next i
    End Sub
```

3. What are the types of errors that you usually see in VBA?

Ans: In VBA, errors can be broadly classified into three categories:

1. Syntax errors: These are the errors that occur due to incorrect use of syntax in the code. For example, missing a closing bracket, using an incorrect function name, etc.

2. Runtime errors: These errors occur while the code is running. It can be due to an unexpected condition, such as dividing by zero, attempting to access a non-existent object, etc.

3. Logical errors: These errors are also known as semantic errors and occur due to incorrect logic in the code. For example, if a loop is not correctly implemented, or if the program is not executing the intended steps. These errors may not cause the code to fail, but they can produce incorrect results.

4. How do you handle Runtime errors in VBA?

Ans: Runtime errors are errors that occur while the code is running. To handle these errors in VBA, you can use error handling techniques. There are three types of error handling statements in VBA: On Error Resume Next, On Error GoTo, and On Error GoTo 0.

- On Error Resume Next: This statement tells VBA to continue executing the code even if an error occurs. It is often used when you want to skip over an error and continue with the code execution. However, it is important to note that this statement can hide errors that should be addressed.

- On Error GoTo: This statement directs VBA to go to a specific label when an error occurs. You can specify the label that you want the code to go to using the syntax "On Error GoTo LabelName". This statement is useful when you want to handle specific errors in a specific way.

- On Error GoTo 0: This statement disables any error handling that was previously enabled by an On Error statement. It is used when you want to turn off error handling for a specific section of code.

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

Ans: Some good practices for handling errors in VBA:

1. Use error handling statements: Use the On Error statements to handle errors that may occur in your code. This will help you catch errors and handle them in a more graceful manner.

2. Use specific error handling: Use specific error handling techniques to handle different types of errors that may occur in your code. This will help you provide more accurate error messages to the user.

3. Use descriptive error messages: When an error occurs, provide a descriptive error message that tells the user what went wrong and how to fix it. This will help the user understand the problem and take appropriate action.

4. Use logging: Use a logging mechanism to track errors that occur in your code. This will help you diagnose and fix problems more easily.

5. Test your code: Before deploying your code, test it thoroughly to make sure that it is error-free. This will help you catch errors before they become a problem for your users.

6. Document your code: Document your code to make it easier for other developers to understand how it works. This will help them diagnose and fix errors more easily.

7. Handle unexpected errors: Handle unexpected errors in your code by providing a default error handler. This will help you catch errors that you may not have anticipated.

By following these good practices, you can ensure that your VBA code is robust, reliable, and easy to maintain.

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 in VBA to create custom functions that can be used in Excel formulas, just like built-in Excel functions. UDFs allow users to create functions that perform specific calculations that are not available in the built-in Excel functions.

Here is an example of a UDF to multiply two numbers in VBA:

```
Function MultiplyNumbers(num1 As Double, num2 As Double) As Double

  MultiplyNumbers = num1 * num2

End Function
```

To use this UDF in Excel, you would enter the following formula in a cell:

=MultiplyNumbers(A1, B1)

where A1 and B1 are the cells containing the numbers you want to multiply.

In this example, the UDF is called "MultiplyNumbers" and it takes two arguments: "num1" and "num2". The function multiplies these two numbers and returns the result using the "MultiplyNumbers" variable.