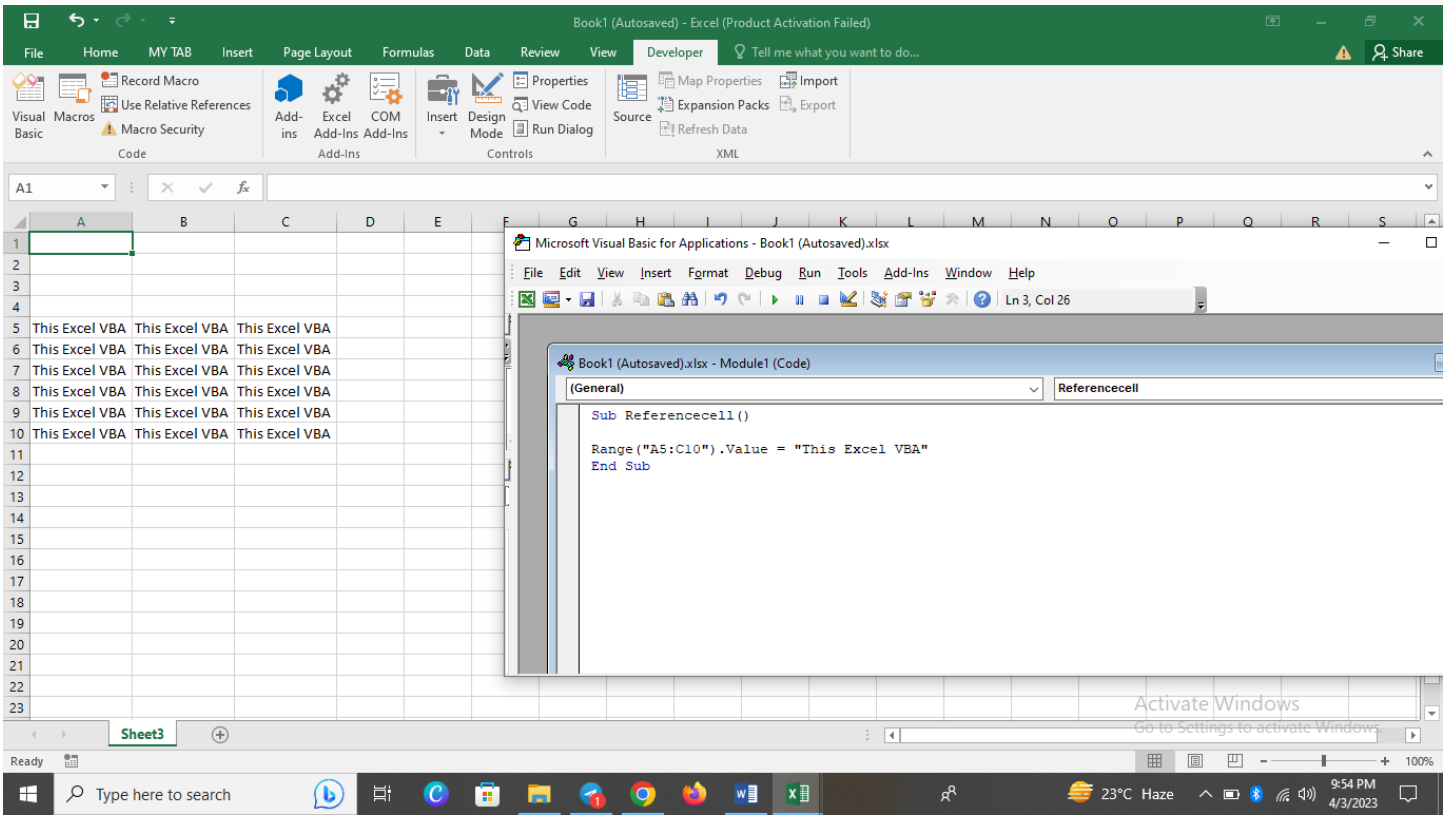
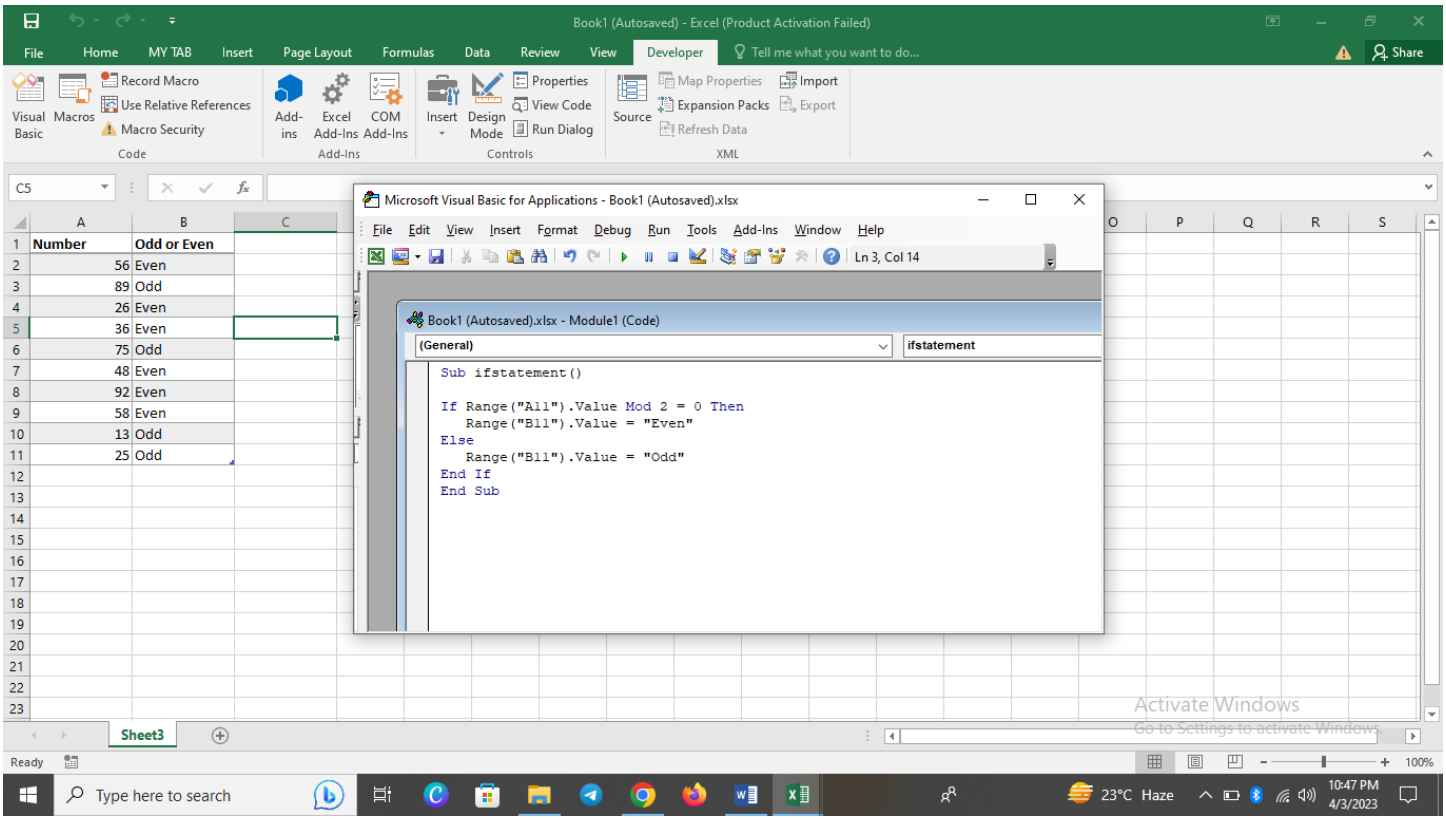
Advance Excel Assignment 20

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	

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

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

In Visual Basic, errors fall into one of three categories: syntax errors, run-time errors, and logic errors.

Syntax Errors

Syntax errors are those that appear while you write code. If you're using Visual Studio, Visual Basic checks your code as you type it in the Code Editor window and alerts you if you make a mistake, such as misspelling a word or using a language element improperly. If you compile from the command line, Visual Basic displays a compiler error with information about the syntax error. Syntax errors are the most common type of errors. You can fix them easily in the coding environment as soon as they occur.

Run-Time Errors

Run-time errors are those that appear only after you compile and run your code. These involve code that may appear to be correct in that it has no syntax errors, but that will not execute. For example, you might correctly write a line of code to open a file. But if the file does not exist, the

application cannot open the file, and it throws an exception. You can fix most run-time errors by rewriting the faulty code or by using exception handling, and then recompiling and rerunning it.

Logic Errors

Logic errors are those that appear once the application is in use. They are most often faulty assumptions made by the developer, or unwanted or unexpected results in response to user actions. For example, a mistyped key might provide incorrect information to a method, or you may assume that a valid value is always supplied to a method when that is not the case. Although logic errors can be handled by using exception handling (for example, by testing whether an argument is Nothing and throwing an ArgumentNullException), most commonly they should be addressed by correcting the error in logic and recompiling the application.

4. How do you handle Runtime errors in VBA?

Here are some best practices you can use when it comes to error handling in Excel VBA.

1. Use 'On Error Go [Label]' at the beginning of the code.
2. Use 'On Error Resume Next' ONLY when you're sure about the errors that can occur.
3. When using error handlers, make sure you're using Exit Sub before the handlers.

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

VBA Error Handling Best Practices

No matter how skilled you get a writing VBA code, errors are always going to be a part of it. The best coders are those who have the skills to handle these errors properly.

Here are some best practices you can use when it comes to error handling in Excel VBA.

1. Use ‘On Error Go [Label]’ at the beginning of the code. This will make sure any error that can happen from there is handled.
2. Use ‘On Error Resume Next’ ONLY when you’re sure about the errors that can occur. Use it with expected error only. In case you use it with unexpected errors, it will simply ignore it and move forward. You can use ‘On Error Resume Next’ with ‘Err.Raise’ if you want to ignore a certain type of error and catch the rest.
3. When using error handlers, make sure you’re using Exit Sub before the handlers. This will ensure that the error handler code is executed only when there is an error (else it will always be executed).

4. Use multiple error handlers to trap different kinds of errors. Having multiple error handler ensures that an error is properly addressed. For example, you would want to handle a ‘type mismatch’ error differently than a ‘Division by 0’ run-time error.

6. What is UDF? Why are UDF’s used?

User-defined functions (UDFs) are custom functions that extend the calculation and data-import capabilities of Excel. Developers create custom calculation packages to provide:

- Functions that are not built into Excel.
- Custom implementations to built-in functions.
- Custom data feeds for legacy or unsupported data sources, and application-specific data flows.

Users who create workbooks can call UDFs from a cell through formulas—for example, "=MyUdf(A1*3.42)"—just like they call built-in functions.

Excel Services UDFs give you the ability to use formulas in cells to call custom functions written in managed code and deployed to Microsoft SharePoint Server 2010. You can create UDFs to.

- Call custom mathematical functions.
- Get data from custom data sources into worksheets.
- Call Web services from the UDFs.

Excel Services UDFs give you the ability to use formulas in cells to call custom functions written in managed code and deployed to Microsoft SharePoint Server 2010. You can create UDFs to: Call custom mathematical functions. Get data from custom data sources into worksheets.