1. Write a VBA code to enter your name in A1 cell using Input Box and
once you enter the name display a message box that says the name
has been entered.
Ans:Certainly! Here's a VBA code that prompts the user to enter their name using an input box and then enters the name into cell A1 of the active worksheet. After entering the name, it displays a message box confirming that the name has been entered.

```vba
Sub EnterName()
Dim userName As String

' Prompt the user to enter their name using an input box
userName = InputBox("Please enter your name:", "Enter Name")

' Check if the user has entered a name
If userName <> "" Then
' Enter the name into cell A1 of the active
```

```
worksheet
ThisWorkbook.ActiveSheet.Range("A1").Value =
userName

' Display a message box confirming the name
has been entered
MsgBox "Name " & userName & " has been
entered in cell A1.", vbInformation, "Name
Entered"
Else
' Display a message box if the user cancels or
does not enter a name
MsgBox "No name entered. Please try again.",
vbExclamation, "No Name Entered"
End If
End Sub
```

2. What are Userforms? Why are they used? How to fill a list box using for loop.

Ans:Userforms in VBA (Visual Basic for Applications) are custom dialog boxes that you can create and design to interact with users.

They allow you to create a graphical user interface (GUI) for your VBA applications, providing a more user-friendly experience. Userforms can contain various controls such as text boxes, labels, buttons, list boxes, combo boxes, etc., which users can interact with to input data, make selections, or perform actions.

Why Userforms are Used:

1. **Enhanced User Interaction**: Userforms provide a more interactive and intuitive way for users to interact with your VBA applications compared to using only message boxes or input boxes.

2. **Customization**: You have full control over the design and layout of the user interface, allowing you to tailor it to the specific needs and preferences of your application and users.

3. **Input Validation**: Userforms allow you to

implement input validation to ensure that users enter data in the correct format or within certain bounds.

4. **Data Presentation**: Userforms can be used to display data in a structured and visually appealing manner, making it easier for users to interpret and analyze information.

5. **Error Handling**: You can incorporate error handling mechanisms into userforms to provide users with helpful feedback in case of errors or invalid inputs.

Filling a List Box Using a For Loop:

To fill a list box using a for loop, you can iterate over a collection of items and add each item to the list box. Here's an example of how to fill a list box with numbers from 1 to 10 using a for loop:

```vba
Sub FillListBoxWithNumbers()
```

```
Dim i As Integer

' Clear the existing items in the list box
ListBox1.Clear

' Fill the list box with numbers from 1 to 10 using a for loop
For i = 1 To 10
ListBox1.AddItem i
Next i
End Sub
```

In this example:
- `ListBox1` refers to the name of the list box control on the userform.
- `Clear` method is used to remove any existing items from the list box before adding new items.
- `AddItem` method is used inside the for loop to add each number to the list box.

3. What is an array? Write a VBA code to enter

students and their marks
from the below table.
Ans:An array is a data structure that can hold multiple values of the same data type under a single variable name. In VBA, arrays are used to store collections of data elements, which can be accessed and manipulated using index values.

Here's a VBA code to enter students and their marks from a table into an array:

```vba
Sub EnterStudentsAndMarks()
Dim students() As Variant
Dim marks() As Variant
Dim numStudents As Integer
Dim i As Integer

' Define the number of students
numStudents = 5 ' You can change this value according to the number of students

' Resize the arrays to accommodate the data
```

```
ReDim students(1 To numStudents)
ReDim marks(1 To numStudents)

' Enter students and their marks into the
arrays
For i = 1 To numStudents
students(i) = Cells(i + 1, 1).Value ' Assuming
student names are in column A starting from
row 2
marks(i) = Cells(i + 1, 2).Value ' Assuming
marks are in column B starting from row 2
Next i

' Display the entered data (optional)
For i = 1 To numStudents
MsgBox "Student: " & students(i) & ", Marks: "
& marks(i)
Next i
End Sub
```

In this code:
- We declare two arrays: `students()` to store student names and `marks()` to store

their corresponding marks.
- We define the number of students (`numStudents`) and resize the arrays accordingly.
- We use a loop to iterate over the rows of the table (assuming student names are in column A and marks are in column B starting from row 2) and populate the arrays with the data from the table.

4. Use the following data to create a pie chart using VBA code. Use Font
- 'Times new Roman', Size -14, Bold, Title - Piechart' and you are per
to use colours as per your taste.
Ans:You can create a pie chart in VBA by adding a chart object to a worksheet and then specifying the data range and chart type. Here's how you can create a pie chart using the given data and customize its font and title:

```vba
```

```
Sub CreatePieChart()
Dim ws As Worksheet
Dim chartObj As ChartObject
Dim chartDataRange As Range
Dim chartTitle As String

' Set the worksheet where the chart will be
placed
Set ws = ThisWorkbook.Sheets("Sheet1") '
Change "Sheet1" to your worksheet name

' Define the data range for the chart
Set chartDataRange = ws.Range("A1:B4") '
Assuming data is in columns A and B

' Add a new chart object to the worksheet
Set chartObj = ws.ChartObjects.Add(Left:=100,
Width:=375, Top:=75, Height:=225)

' Set the chart type to pie chart
chartObj.Chart.ChartType = xlPie

' Set the data source for the chart
chartObj.Chart.SetSourceData
```

```
Source:=chartDataRange

' Set the title of the chart
chartTitle = "Pie Chart"
chartObj.Chart.HasTitle = True
chartObj.Chart.ChartTitle.Text = chartTitle

' Customize font properties for the chart title
With
chartObj.Chart.ChartTitle.Format.TextFrame2.
TextRange.Font
.Name = "Times New Roman"
.Size = 14
.Bold = msoTrue
End With

' Customize font properties for the legend
With chartObj.Chart.Legend.Font
.Name = "Times New Roman"
.Size = 14
End With

' Customize font properties for the data
labels
```

```
With chartObj.Chart.SeriesCollection(1).DataLabels.Format.TextFrame2.TextRange.Font
.Name = "Times New Roman"
.Size = 14
.Bold = msoTrue
End With
End Sub
```

In this code:
- Replace `"Sheet1"` with the name of the worksheet where your data is located.
- Adjust the data range (`chartDataRange`) to match the range of your data.
- You can customize the title of the pie chart by changing the `chartTitle` variable.
- The font properties for the title, legend, and data labels are customized according to the given specifications (Times New Roman, Size 14, Bold). You can adjust these properties as needed.

5. Check the dataset in the link given below and create a pivot table using VBA showing the sales for the year from stationary category.

Ans:

```
Sub CreatePivotTable()
Dim ws As Worksheet
Dim pt As PivotTable
Dim pc As PivotCache
Dim srcData As Range
Dim lastRow As Long

' Define the data range
Set ws = ThisWorkbook.Sheets("Sheet1") '
Change "Sheet1" to your worksheet name
lastRow = ws.Cells(ws.Rows.Count,
"A").End(xlUp).Row
Set srcData = ws.Range("A1:F" & lastRow)

' Add a new worksheet for the pivot table
Dim pivotSheet As Worksheet
Set pivotSheet = ThisWorkbook.Sheets.Add

' Create a pivot cache
```

```
Set pc = ThisWorkbook.PivotCaches.Create( _
SourceType:=xlDatabase, _
SourceData:=srcData)

' Create a pivot table
Set pt = pc.CreatePivotTable( _
TableDestination:=pivotSheet.Range("A3"), _
TableName:="StationarySalesPivotTable")

' Set row and column fields
With pt
.PivotFields("Date").Orientation = xlRowField
.PivotFields("Category").Orientation =
xlColumnField
End With

' Set data field
With pt.PivotFields("Amount")
.Orientation = xlDataField
.Function = xlSum
.NumberFormat = "₹#,##0.00"
End With

' Format pivot table
```

```
With pt
' Autofit columns
.Columns.AutoFit
' Set title
.Parent.Cells(1, 1).Value = "Stationary Sales Pivot
Table"
' Apply bold font
.Parent.Cells(1, 1).Font.Bold = True
End With
End Sub
```

6. Write step by step procedure to protect your workbook using a password.

Ans:Certainly! Here's a step-by-step procedure to protect your workbook using a password in Microsoft Excel:

1. **Open Your Workbook**: First, open the Excel workbook that you want to protect with a password.

2. **Navigate to the "File" Menu**: Click on the

"File" tab located in the top-left corner of the Excel window. This will open the backstage view.

3. **Select "Info"**: In the backstage view, select the "Info" option from the list on the left. This will display information about the current workbook.

4. **Click on "Protect Workbook"**: Under the "Info" section, you will see various options related to workbook protection. Click on the "Protect Workbook" dropdown arrow.

5. **Choose "Encrypt with Password"**: From the dropdown menu, select the option "Encrypt with Password".

6. **Enter the Password**: A dialog box will appear asking you to enter a password. Type the password that you want to use to protect the workbook. Make sure to choose a strong password that is difficult to guess.

7. **Confirm the Password**: After entering

the password, you will be prompted to confirm it. Re-enter the same password to confirm.

8. **Click "OK"**: Once you've entered and confirmed the password, click "OK" to apply the protection to the workbook.

9. **Save Your Workbook**: After setting the password protection, make sure to save your workbook to apply the changes. You can do this by clicking on the "Save" or "Save As" option from the File menu.

10. **Close and Reopen the Workbook**: To ensure that the password protection is working as expected, close the workbook and reopen it. You should be prompted to enter the password before you can access the contents of the workbook.