NuvoRetail Assignment

Requirements

- ☑ Load data using PowerQuery and automate the process using VBA.
- ☑ Using VBA and Macros automate dashboard functions and report generation.
- ☑ Use python to automate and productionize the entire process.
- ☑ Use alternate of substring in excel (Check 3rd bullet point Under Section Automating Data Loading & Cleaning and subsub-section Data transformation)

DataSet

NuvoRetail empowers e-commerce customers using advanced analytics, so I found fit to use amazon e-commerce sales data for this assignment. The data is publicly available on kaggle.

DataSet link
https://www.kaggle.com/datasets/thedevastator/unlock-profits-with-e-commerce-sales-data/data?select=Amazon+Sale+Report.csv

The dataset provides detailed insights into Amazon sales data, including SKU Code, Design Number, Stock, Category, Size and Color, to help optimize product profitability

- Category: Type of product. (String)
- Size: Size of the product. (String)
- Date: Date of the sale. (Date)
- Status: Status of the sale. (String)
- Fulfilment: Method of fulfilment. (String)
- Style: Style of the product. (String)
- SKU: Stock Keeping Unit. (String)
- ASIN: Amazon Standard Identification Number. (String)
- Courier Status: Status of the courier. (String)
- Qty: Quantity of the product. (Integer)
- Amount: Amount of the sale. (Float)
- B2B: Business to business sale. (Boolean)
- Currency: The currency used for the sale. (String)
- Untitled: 22: Not Useful

Automating Data Loading & Cleaning

I Used VBA to automate the process of data loading and cleaning from raw csv. Below is the VBA script used to automate the process.

```
Sub TransformAmazonSaleData()
    Dim wsSource As Worksheet
    Dim wsTarget As Worksheet
    Dim lastRow As Long
    Dim i As Long
    Dim targetRow As Long
    Dim lastProcessedRow As Long

    ' Set the source and target worksheets
    On Error GoTo ErrHandler
    Set wsSource = ThisWorkbook.Sheets("AmazonSaleData")
    Set wsTarget = ThisWorkbook.Sheets("TransformedSaleData")
    On Error GoTo 0

    ' Optimize performance
    Application.ScreenUpdating = False
    Application.Calculation = xlCalculationManual

    ' Get the last processed row from a hidden sheet or named range
    On Error Resume Next
    lastProcessedRow = ThisWorkbook.Names("LastProcessedRow").RefersToRange.Value
    On Error GoTo 0

    If lastProcessedRow = 0 Then lastProcessedRow = 1 ' If not set, start from the
first row

    ' Get the last row of the source data
    lastRow = wsSource.Cells(wsSource.Rows.Count, "A").End(xlUp).Row

    ' Clear the target sheet except for headers
    If wsTarget.Cells(2, 1).Value <> "" Then
        wsTarget.Rows("2:" & wsTarget.Rows.Count).ClearContents
    End If

    ' Initialize the target row counter
    targetRow = 2

    ' Add headers to the target sheet if it's empty
```

```
    If wsTarget.Cells(1, 1).Value = "" Then
        Dim headers As Variant


headers = Array("index", "Order ID", "Date", "Status", "Fulfilment", "Sales
Channel", "ship-service-level", "Style", "SKU", "Category", "Size", "ASIN",
"Courier Status", "Qty", "currency", "Amount", "ship-city", "ship-state",
"ship-postal-code", "ship-country", "promotion-ids", "B2B", "fulfilled-by")

        For i = LBound(headers) To UBound(headers)
            wsTarget.Cells(1, i + 1).Value = headers(i)
        Next i
    End If

    ' Loop through the source data starting from the last processed row + 1
    For i = lastProcessedRow + 1 To lastRow
        ' Copy relevant data to the target sheet and replace blanks with "Null"
        wsTarget.Cells(targetRow, 1).Value = IIf(wsSource.Cells(i, 1).Value = "",
"Null", wsSource.Cells(i, 1).Value) ' index
        wsTarget.Cells(targetRow, 2).Value = IIf(wsSource.Cells(i, 2).Value = "",
"Null", wsSource.Cells(i, 2).Value) ' Order ID
        wsTarget.Cells(targetRow, 3).Value = IIf(wsSource.Cells(i, 3).Value = "",
"Null", wsSource.Cells(i, 3).Value) ' Date
        ' Handle Status column with hyphen
        statusValue = wsSource.Cells(i, 4).Value
        hyphenPos = InStr(statusValue, "-")
        If hyphenPos > 0 Then
            statusValue = Mid(statusValue, hyphenPos + 1)
        End If
        wsTarget.Cells(targetRow, 4).Value = IIf(statusValue = "", "Null",
statusValue) ' Status
        wsTarget.Cells(targetRow, 5).Value = IIf(wsSource.Cells(i, 5).Value = "",
"Null", wsSource.Cells(i, 5).Value) ' Fulfilment
        wsTarget.Cells(targetRow, 6).Value = IIf(wsSource.Cells(i, 6).Value = "",
"Null", wsSource.Cells(i, 6).Value) ' Sales Channel
        wsTarget.Cells(targetRow, 7).Value = IIf(wsSource.Cells(i, 7).Value = "",
"Null", wsSource.Cells(i, 7).Value) ' ship-service-level
        wsTarget.Cells(targetRow, 8).Value = IIf(wsSource.Cells(i, 8).Value = "",
"Null", wsSource.Cells(i, 8).Value) ' Style
        wsTarget.Cells(targetRow, 9).Value = IIf(wsSource.Cells(i, 9).Value = "",
"Null", wsSource.Cells(i, 9).Value) ' SKU
        wsTarget.Cells(targetRow, 10).Value = IIf(wsSource.Cells(i, 10).Value = "",
"Null", wsSource.Cells(i, 10).Value) ' Category
        wsTarget.Cells(targetRow, 11).Value = IIf(wsSource.Cells(i, 11).Value = "",
"Null", wsSource.Cells(i, 11).Value) ' Size
        wsTarget.Cells(targetRow, 12).Value = IIf(wsSource.Cells(i, 12).Value = "",
"Null", wsSource.Cells(i, 12).Value) ' ASIN
```

```
        wsTarget.Cells(targetRow, 13).Value = IIf(wsSource.Cells(i, 13).Value = "",
"Null", wsSource.Cells(i, 13).Value) ' Courier Status
        wsTarget.Cells(targetRow, 14).Value = IIf(wsSource.Cells(i, 14).Value = "",
"Null", wsSource.Cells(i, 14).Value) ' Qty



' Handle currency and Amount based on Status
        If wsSource.Cells(i, 4).Value = "Cancelled" Then
            wsTarget.Cells(targetRow, 15).Value = "Null" ' currency
            wsTarget.Cells(targetRow, 16).Value = "Null" ' Amount
        Else
            wsTarget.Cells(targetRow, 15).Value = IIf(wsSource.Cells(i, 15).Value =
"", "Null", wsSource.Cells(i, 15).Value) ' currency
            wsTarget.Cells(targetRow, 16).Value = IIf(wsSource.Cells(i, 16).Value =
"", "Null", wsSource.Cells(i, 16).Value) ' Amount
        End If

        wsTarget.Cells(targetRow, 17).Value = IIf(wsSource.Cells(i, 17).Value = "",
"Null", wsSource.Cells(i, 17).Value) ' ship-city
        wsTarget.Cells(targetRow, 18).Value = IIf(wsSource.Cells(i, 18).Value = "",
"Null", wsSource.Cells(i, 18).Value) ' ship-state
        wsTarget.Cells(targetRow, 19).Value = IIf(wsSource.Cells(i, 19).Value = "",
"Null", wsSource.Cells(i, 19).Value) ' ship-postal-code
        wsTarget.Cells(targetRow, 20).Value = IIf(wsSource.Cells(i, 20).Value = "",
"Null", wsSource.Cells(i, 20).Value) ' ship-country
        wsTarget.Cells(targetRow, 21).Value = IIf(wsSource.Cells(i, 21).Value = "",
"Null", wsSource.Cells(i, 21).Value) ' promotion-ids
        wsTarget.Cells(targetRow, 22).Value = IIf(wsSource.Cells(i, 22).Value = "",
"Null", wsSource.Cells(i, 22).Value) ' B2B
        wsTarget.Cells(targetRow, 23).Value = IIf(wsSource.Cells(i, 23).Value = "",
"Null", wsSource.Cells(i, 23).Value) ' fulfilled-by

        ' Increment the target row counter
        targetRow = targetRow + 1
    Next i

    ' Save the last processed row
    ThisWorkbook.Names.Add Name:="LastProcessedRow", RefersTo:=lastRow

    ' Restore settings
    Application.ScreenUpdating = True
    Application.Calculation = xlCalculationAutomatic

    MsgBox "Data transformation complete!"
    Exit Sub
```

```
ErrHandler:
    MsgBox "Error: " & Err.Description & ". Please check the sheet names and data
structure."
    Application.ScreenUpdating = True
    Application.Calculation = xlCalculationAutomatic
End Sub
```

Steps followed in the script

Sub Procedure TransformAmazonSaleData, transforms data from one worksheet to another within the same workbook, ensuring data cleaning and optimization for further analysis. Here's a summary of its key steps:

Setup:

- It defines the source (wsSource) and target (wsTarget) worksheets, specifically named "AmazonSaleData" and "TransformedSaleData".
- Error handling ensures the code stops gracefully if the specified sheets aren't found.

Optimization:

- To speed up the macro, it turns off screen updating and automatic calculation.
- Last Processed Row: Retrieves the last processed row to avoid reprocessing the same data each time the script runs. It stores this value in a hidden named range (LastProcessedRow).

Clearing Target Sheet:

- If there is existing data, it clears everything except the headers.

Header Setup:

- If headers are missing in the target sheet, it adds a predefined set of headers.

Data Transformation:

- Loops through each unprocessed row in the source sheet and copies relevant columns to the target sheet.
- Empty cells are filled with "Null" to ensure data consistency.

- For the "Status" column, if there is a hyphen (-), only the part after the hyphen is retained as the part before is Shipped which is not required for our particular usecase. **** This Fulfills the requirement to find and substring in excel ****
- If the "Status" is "Cancelled", both "currency" and "Amount" are set to "Null".

Saving Progress:

- After processing, it saves the last processed row for reference in future runs.

Restoration and Cleanup:

- Restores screen updating and calculation settings and displays a completion message.

Error Handling:

- If an error occurs (e.g., sheet names don't match), it shows an error message and re-enables screen updating and calculations.

Automatically trigger the above script on new row addition

I have also written a trigger that will call the script when a new row is added in original sheet i.e "AmazonSaleData"
Below is the script

```
Private Sub Worksheet_Change(ByVal Target As Range)
    ' Check if the change is within the used range of the sheet
    If Not Intersect(Target, Me.UsedRange) Is Nothing Then
        ' Run the transformation macro
        Call TransformAmazonSaleData
    End If
End Sub
```

Automating Dashboard Refresh

There are three metrics I have tracked and visualized:

1. Total Sales Amount by Date
2. Order Status Distribution
3. Sales Amount by Fulfilment Type

Created a new sheet for the dashboards and added graphs for each metric.

Metric 1: Total Sales Amount by Date

1. Created a new sheet for the dashboard.
2. Added a pivot table to summarize the data in TransformedSaleData sheet.
3. Created a line chart to represent the sales amount by date

Metric 2: Order Status Distribution

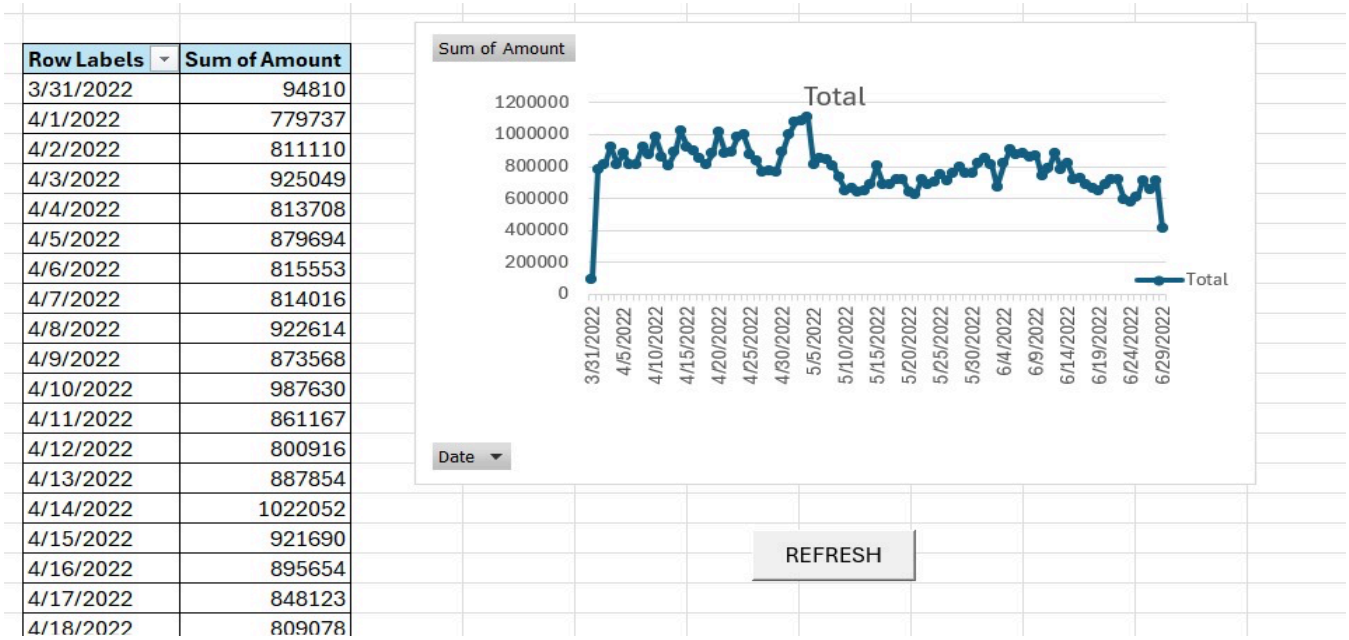
1. Added another pivot table using the TransformedSaleData sheet.
2. Created a pie chart representing the order status distribution.

Metric 3: Sales Amount by Fulfilment Type

1. Added another pivot table using “Fulfilment” and “Amount”.
2. Created a bar chart using the above pivot table.

Now to automate and and refresh the graphs and charts on demand recorded one macro each for each graph and assigned them to a refresh on the page.

Macro to refresh of line chart for Metric 1



Row Labels	Sum of Amount
3/31/2022	94810
4/1/2022	779737
4/2/2022	811110
4/3/2022	925049
4/4/2022	813708
4/5/2022	879694
4/6/2022	815553
4/7/2022	814016
4/8/2022	922614
4/9/2022	873568
4/10/2022	987630
4/11/2022	861167
4/12/2022	800916
4/13/2022	887854
4/14/2022	1022052
4/15/2022	921690
4/16/2022	895654
4/17/2022	848123
4/18/2022	809078

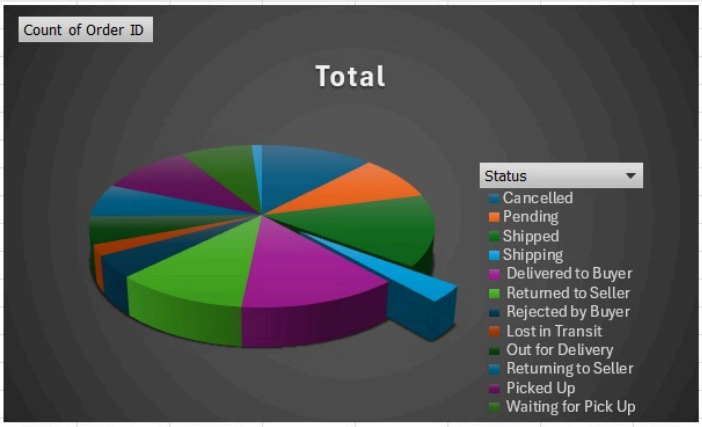

```
Sub Linechart()
'
' Linechart Macro
'

'
    Range("K1").Select
```

```
    ActiveSheet.ChartObjects("Chart 1").Activate
    ActiveChart.PlotArea.Select
    ActiveChart.ChartArea.Select
    ActiveChart.PivotLayout.PivotTable.PivotCache.Refresh
End Sub
```

Macro to refresh of Pie chart for Metric 2

Row Labels	Count of Order ID
Cancelled	11
Pending	8
Shipped	13
Shipping	3
Delivered to Buyer	12
Returned to Seller	10
Rejected by Buyer	4
Lost in Transit	2
Out for Delivery	5
Returning to Seller	6
Picked Up	9
Waiting for Pick Up	7
Damaged	1
Grand Total	

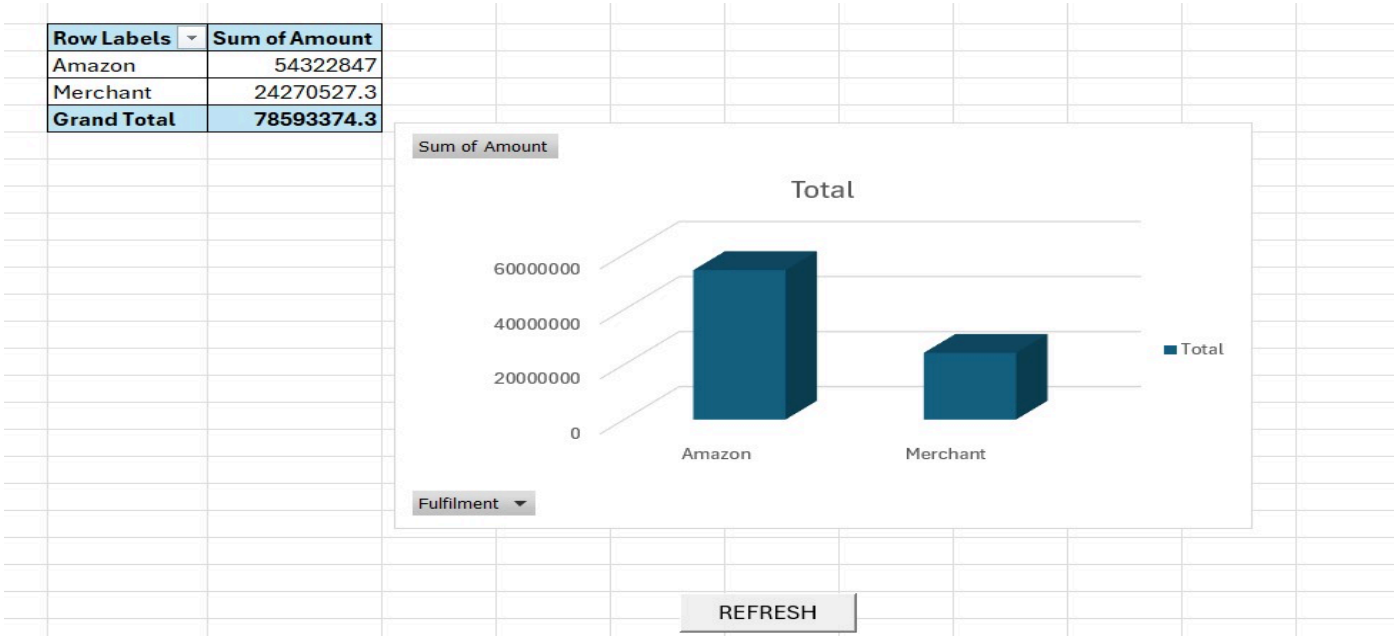


REFRESH

```
Sub Piechart()
'
' Piechart Macro
'

'
    ActiveSheet.ChartObjects("Chart 2").Activate
    ActiveChart.PivotLayout.PivotTable.PivotCache.Refresh
End Sub
```

Macro to refresh of Bar graph for Metric 3




```
Sub Barchart()
'
' Barchart Macro
'

'
    ActiveSheet.ChartObjects("Chart 1").Activate
    ActiveChart.PlotArea.Select
    ActiveChart.ChartArea.Select
    ActiveChart.PivotLayout.PivotTable.PivotCache.Refresh
End Sub
```

Automating Cumulative Report Generation

Here I have created a VBA module to create a report out of all the three sheets that have the dashboards mentioned in the last section.

Steps

1. The VBA script will first refresh all the graphs in all three sheets by calling the macro mentioned in the previous section.
2. It will then take all the charts from all three sheets and put it in a sheet named report.
3. Save the sheet with a name appended by the timestamp. This is done so that no two reports will have the same name and can be easily ordered.
4. The reports are generated in a dir named reports in the same dir as the excel workbook.

VBA Module

```
Sub GenerateReport()
    Dim wsReport As Worksheet
    Dim ws1 As Worksheet
    Dim ws2 As Worksheet
    Dim ws3 As Worksheet
    Dim chartObj As ChartObject
    Dim reportPath As String
    Dim nextRow As Long

    ' Set the worksheets
    Set ws1 = ThisWorkbook.Sheets("Fulfilment")
    Set ws2 = ThisWorkbook.Sheets("Datewise")
    Set ws3 = ThisWorkbook.Sheets("Statuswise")

    ' Refresh data in the sheets
    ws1.Activate
    Call Barchart
    ws2.Activate
    Call Linechart
    ws3.Activate
    Call Piechart

    ' Create a new sheet for the report
    On Error Resume Next
    Application.DisplayAlerts = False
    ThisWorkbook.Sheets("Report").Delete
    Application.DisplayAlerts = True
```

```
    On Error GoTo 0at(Now, "yyyymmdd_HHMMSS")

    ' Save the new workbook with the report sheet
    reportPath = ThisWorkbook.Path & "\reports\SalesReport_" & timestamp &
".xlsx"

    NewWorkbook.SaveAs Filename:=reportPath, FileFormat:=xlOpenXMLWorkbook
    NewWorkbook.Close SaveChanges:=False

    ' Delete the temporary report sheet from the original workbook
    Application.DisplayAlerts = False
    wsReport.Delete
    Application.DisplayAlerts = True

    MsgBox "Report generated and saved successfully at " & reportPath
End Sub
```

Automating Generating and Emailing Reports

Finally a python script has been written to link together the workflow. Below is the workflow and this run keeps generating and emailing the reports every hour. The reports are stored in the same directory as the python script and the workbook.

Steps

1. The script first asks for the list of email ids that you want to send the report to.
2. Then it asks for your email and password(this can be your mail password or app password shown in the next section I have used app password and gmail).
3. It next asks for the subject and the body of the email.
4. The script starts first checks if the dir named reports is present if not it will create the directory.
5. Opens excel and calls the macro to generate the report mentioned in the last section.
6. The report is generated and save under the reports directory.
7. The script takes the latest report that was created and sends email to all the email address mentioned in step 1.
8. Step 5 to 7 keeps repeating every hour till we stop the script.

We can configure the frequency of sending these reports in the code itself.

Generating app password in gmail to make this work.

To generate an **App Password** for Gmail, follow these steps. This password is a unique 16-character code that allows less secure applications to access your Google account without using your main Google password. You'll need this if you're using Gmail in a third-party app or service that doesn't support two-step verification directly.

Prerequisites

1. **Enable Two-Step Verification**: You need to have **two-step verification** enabled on your Google account to use app passwords.

Steps to Generate a Gmail App Password

1. **Go to Your Google Account**:
    - Open https://myaccount.google.com/ and sign in to your Google account.
2. **Navigate to Security**:
    - On the left sidebar, click on **Security**.
3. **Enable Two-Step Verification** (if not already enabled):
    - Scroll to **"Signing in to Google"** and click on **Two-Step Verification**.
    - Follow the instructions to set it up if you haven't already.
4. **Create an App Password**:
    - Once two-step verification is enabled, go back to the **Security** page.
    - Under **"Signing in to Google"**, click on **App Passwords**.
5. **Select the App and Device**:
    - In the App Passwords section, you'll be prompted to choose the app and device for which you need the password.
    - From the **Select App** dropdown, choose **Mail**.
    - From the **Select Device** dropdown, choose the device you're generating this for, or select **Other** if you want to label it manually.
6. **Generate the Password**:
    - Click **Generate**. Google will provide a 16-character password.
    - Copy this password (you'll need it to set up your app).
7. **Use the App Password**:
    - Enter this app password instead of your Google password in the application you're setting up (e.g., Outlook, Thunderbird, etc.)

How To Run

Prerequisites

1. Latest version of python is required I am using Python 3.13
2. Excel with macros enabled
3. Also make sure the following packages are installed or install them using and run the script fill in all the the user inputs and you should see

```
Pip install email
Pip install pywin32
Python automation.py
```

Ending Remarks

This Assignment pushed me to learn about automation using python and using event triggers in VBA. I also learned about the end to end workflow from my own research. Thank you, Any feedback from the NuvoRetail team would be highly appreciated.