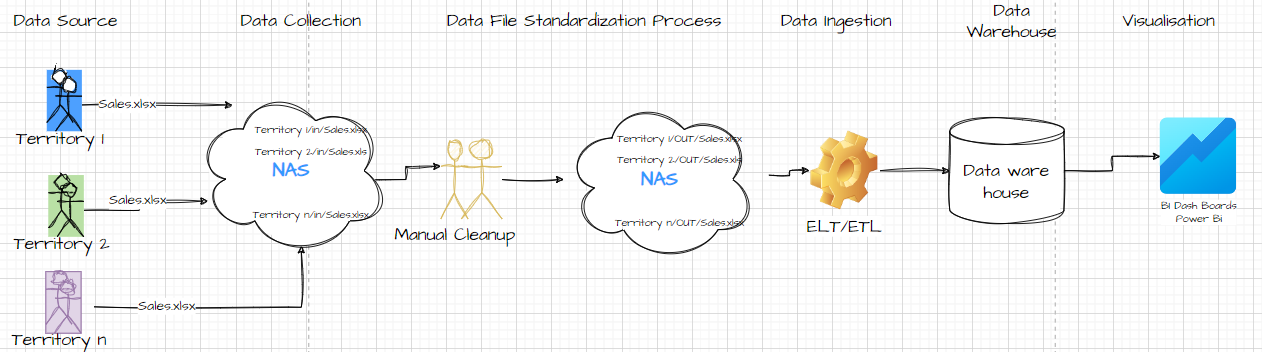
Python Program for Automation – (Unpivoting and Transformations)

Problem Statement

For one of our Business Analytics Dashboards Projects, we get Sales data from different countries (100 +) around the globe, and we consolidate this data and show the trend analysis, such as MTD, YTD, and ITD KPIs.

The data we receive is in an Excel file in a crosstab style (Pivot) format, the data file format is not standard, and each country may have Sales Volume for different months., For example, the USA may have 12 months of Sales, Canada may have 8 months of Sales, and London may have 5 months of Sales.

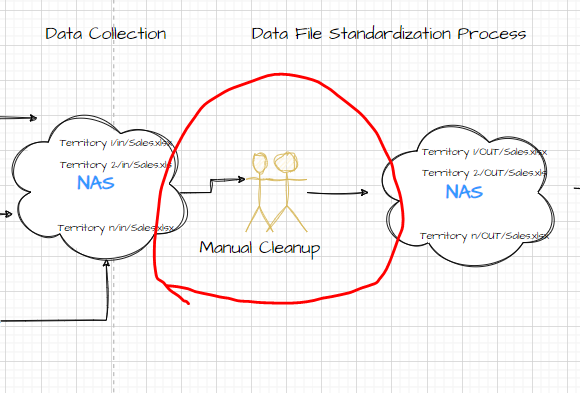
To process this data and load it into the Data warehouse, the data ingestion process (ELT/ETL) expects properly formatted standard files with constant columns and valid data in each column domain. To make this happen, we have a team of multiple team members who manually convert pivot files to un- pivots files and do the data transformation, such as converting string values from Jun’23 to 01-31-2023.



Up above is a diagram of the existing solution design

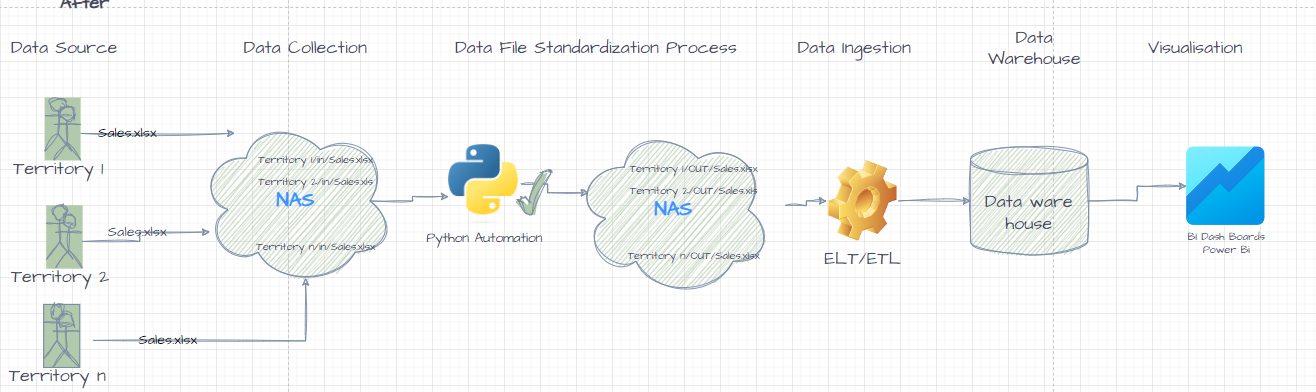
The problem with performing the aforesaid standardization activities is that 1) we spend a good amount of effort and time and 2) human errors in the process.

The challenge area in the solution design is highlighted below



To solve the above said concerns, we have developed a Python program that is more efficient and economical.

## Proposed Solution Design

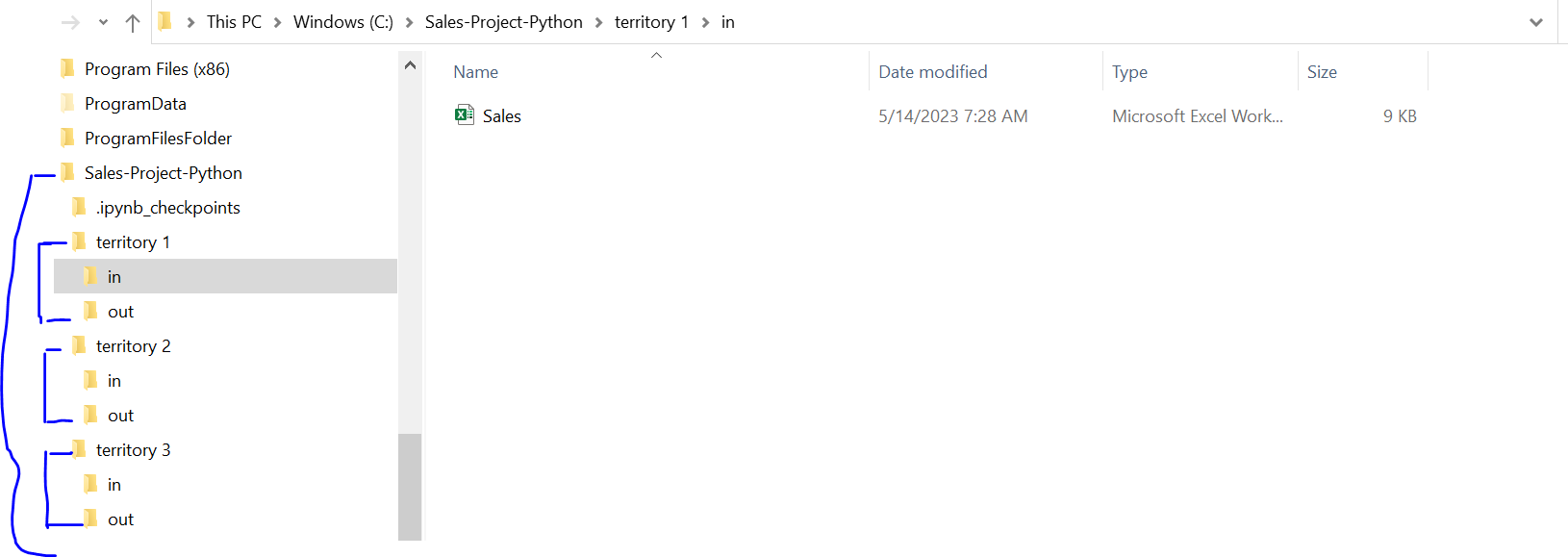


## Solution Implementation Steps

The Python code has been designed using Jupyter Notebook to see the result at each step. The same code can be put into a Python file and can be scheduled as needed.

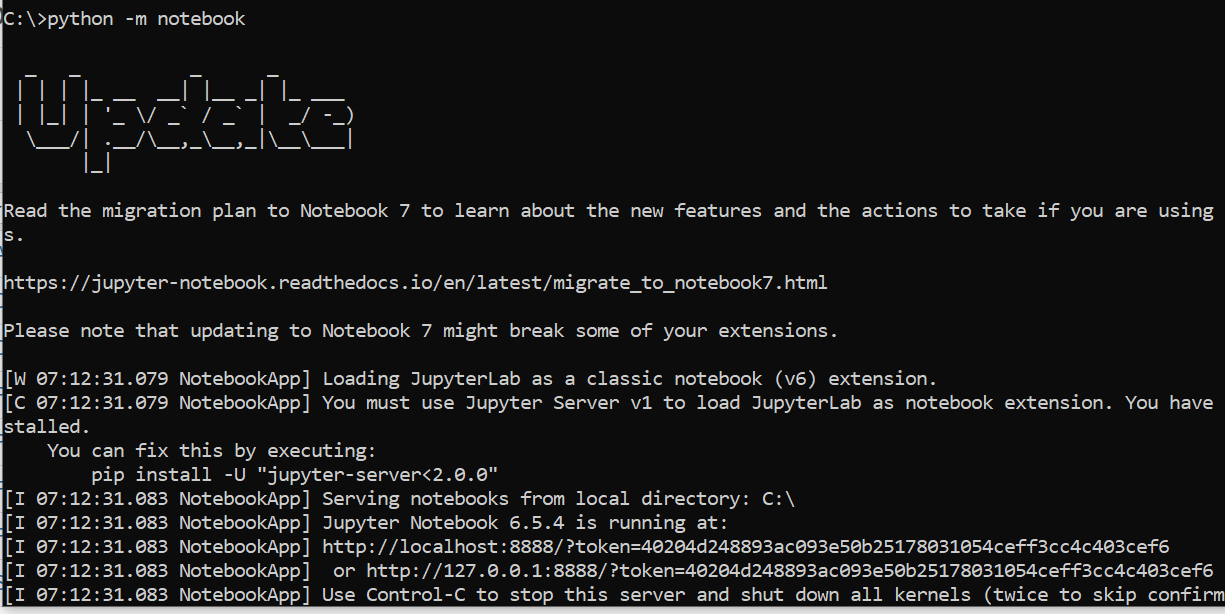
#### Preparing the project structure

1. Create a project directory that says “Sales-Project-Python.”
2. Create 3 subfolders under the project directory named Territory 1, Territory 2, and Territory 3.
3. Create 2 subfolders under each territory with the name in and out as shown in the below picture
4. The original raw files can be placed in the “in” folder, and the result files are placed in the “out” folder



#### Invoke the Jupyter Notebook.

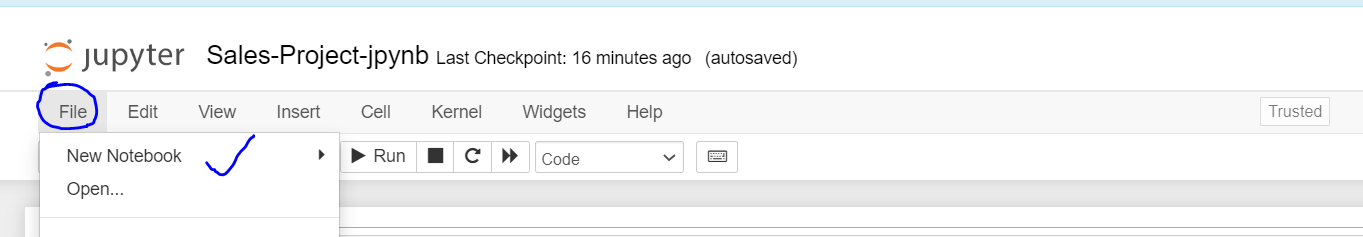
Invoke the Jupyter notebook using the “python **-**m notebook.” Command.



#### Create a New Jupyter notebook.

After invoking the jupyter notebook, the browser opens with the project’s root.

Now we will see the Jupyter notebook opened and now create a new Jupyter notebook. File 🡪 New Notebook



Now that we have a new window lets write the code !

#### Let us implement the logic for Territory 1 and then expand this logic to Multiple Territories.

1. Load the territory 1 raw data into a data frame using pandas and then transform.

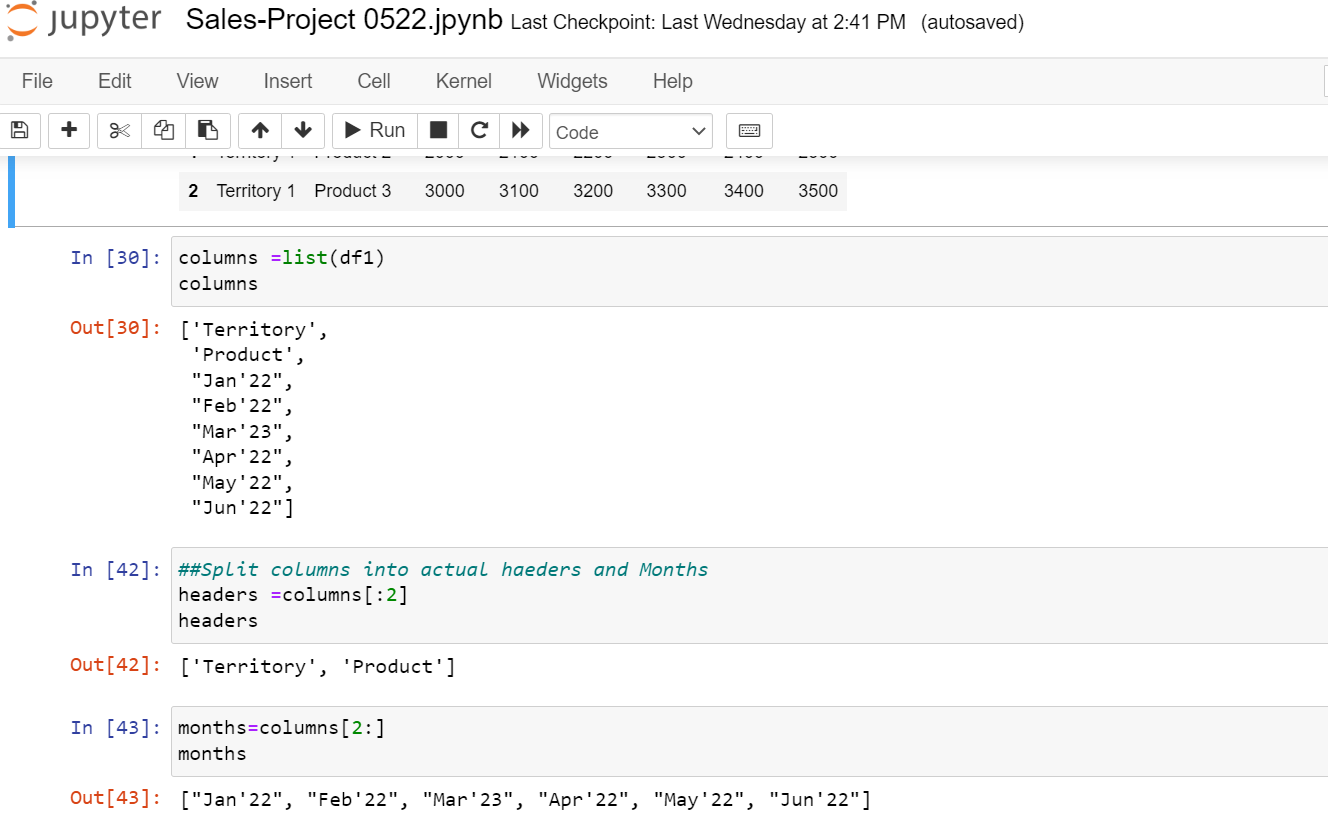
### 

1. Import the data into the data frame using the pandas**.**read\_excle**()**method. This method expects the Excel file name with the path and returns the data frame, i.e., df1, as shown in the above picture.
2. Now that the data is available in data frame df1, let’s print it. Just place it in the cell and click on execute. The output is shown in the above picture

#### Separate header columns and variable columns

*Separate the header columns,* such as Territory Name and Product Name, and the variable columns, such as Jan’22, Feb’22, etc., to transform the data from pivot to unpivot and make it a fixed structure

* + 1. Let’s use the **list()** function to extract all the columns from a data frame and store it in the variable “Columns.”
    2. Now let’s separate fixed headers and variable columns into 2 different Lists from variable “Columns”, one for fixed columns and the second for variable months and store them into respective variables “headers” and “months.” As shown below picture

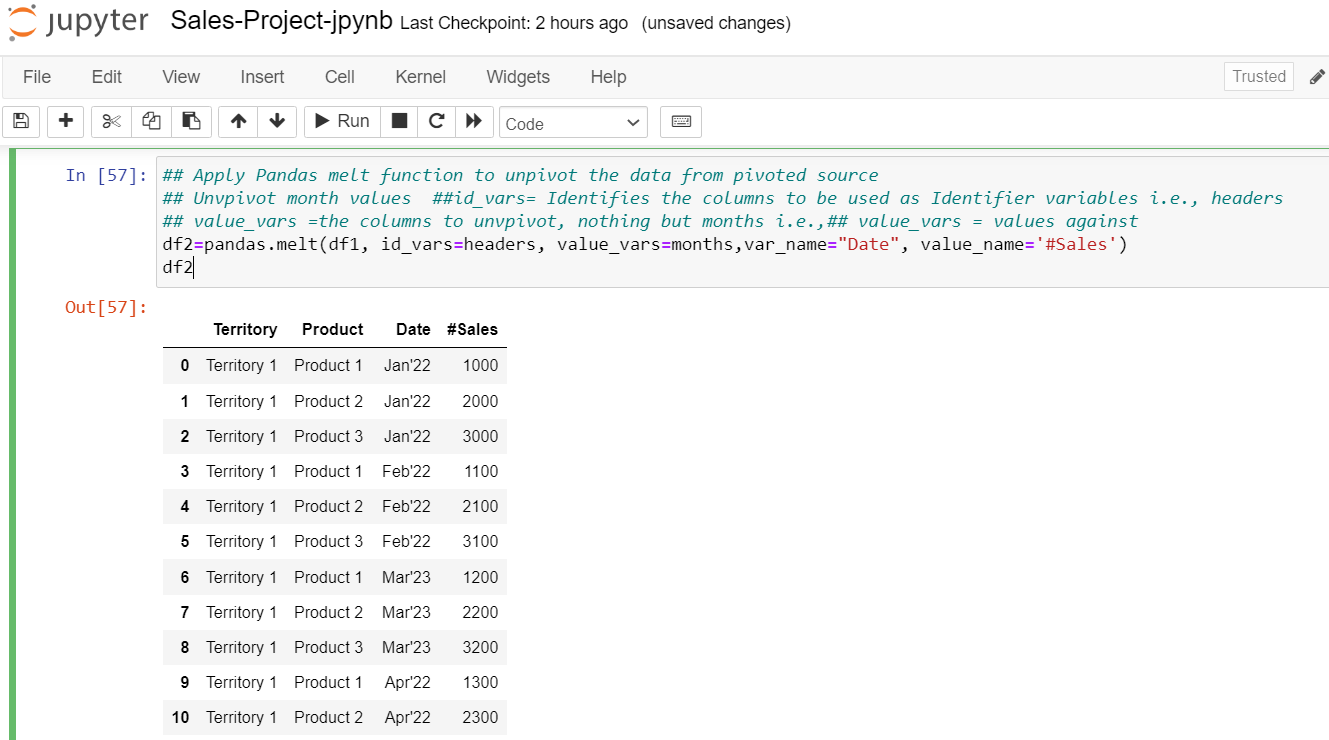


#### Unpivoting the data using the Pandas Melt function

*To unpivot the data, padas has provided a function called* pandas**.** melt **().** *The signature of this function is*

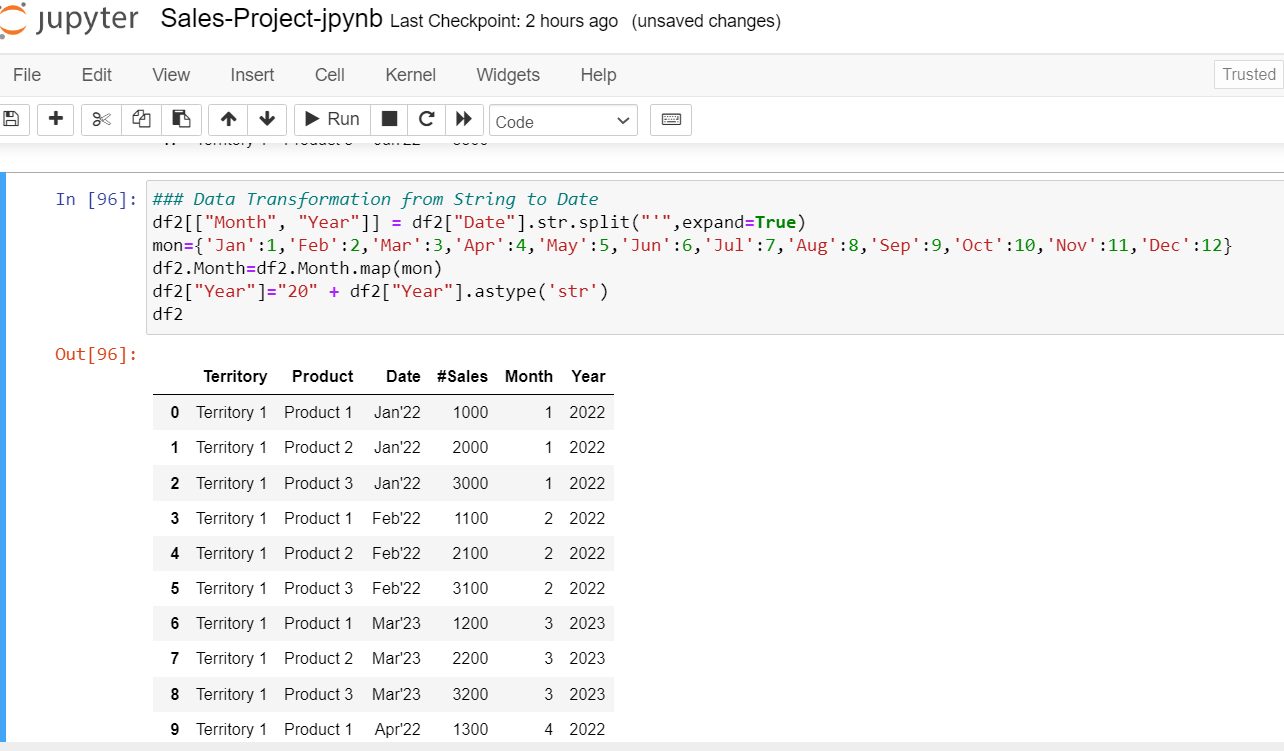
pandas**.** melt **(**frame**,** id\_vars**=None,** value\_vars**=None,** var\_name**=None,** value\_name**=**'value'**,** col\_level**=None,** ignore\_index**=True)**

Here the first parameter is data frame df1; the second parameter, id\_vars, is nothing but headers, the third parameter, value\_vars is dynamic months, the fourth parameter, var\_name is nothing but a label for the dynamic month which is a “Date”. The fifth parameter, value\_name is nothing but a measure value nothing but “# Sales”, and then execute the code you will see the below results.

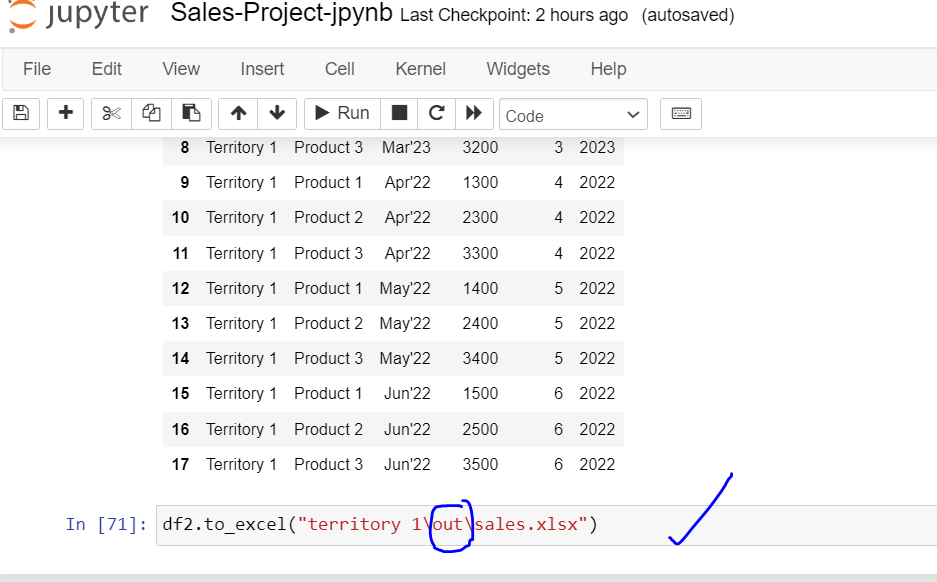


#### Data Transformation

* To transform the month and year composite column to separate month and year columns, we can use the string split **()** function, as shown in the picture with the delimiter “ ’ ”.
* To convert sting month values to integer values, let’s create a dictionary object mapping Jan to 1, Feb to 2, etc. Example Mon\_Dictionary ={‘Jan’:1, “Feb”:2, …} as shown below
* Now, to overwrite the month string value (Jan, Feb, etc.) with Integer Value (1,2, etc.) we can use the Python **map()function as shown below**
* The last transformation is to add century values to two-digit years using concatenation symbol (+) as shown below



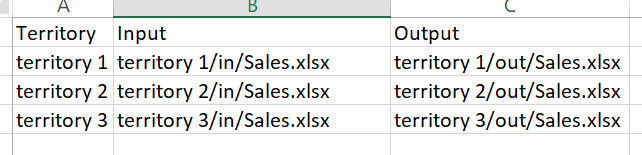
## Save final result in the Target folder.



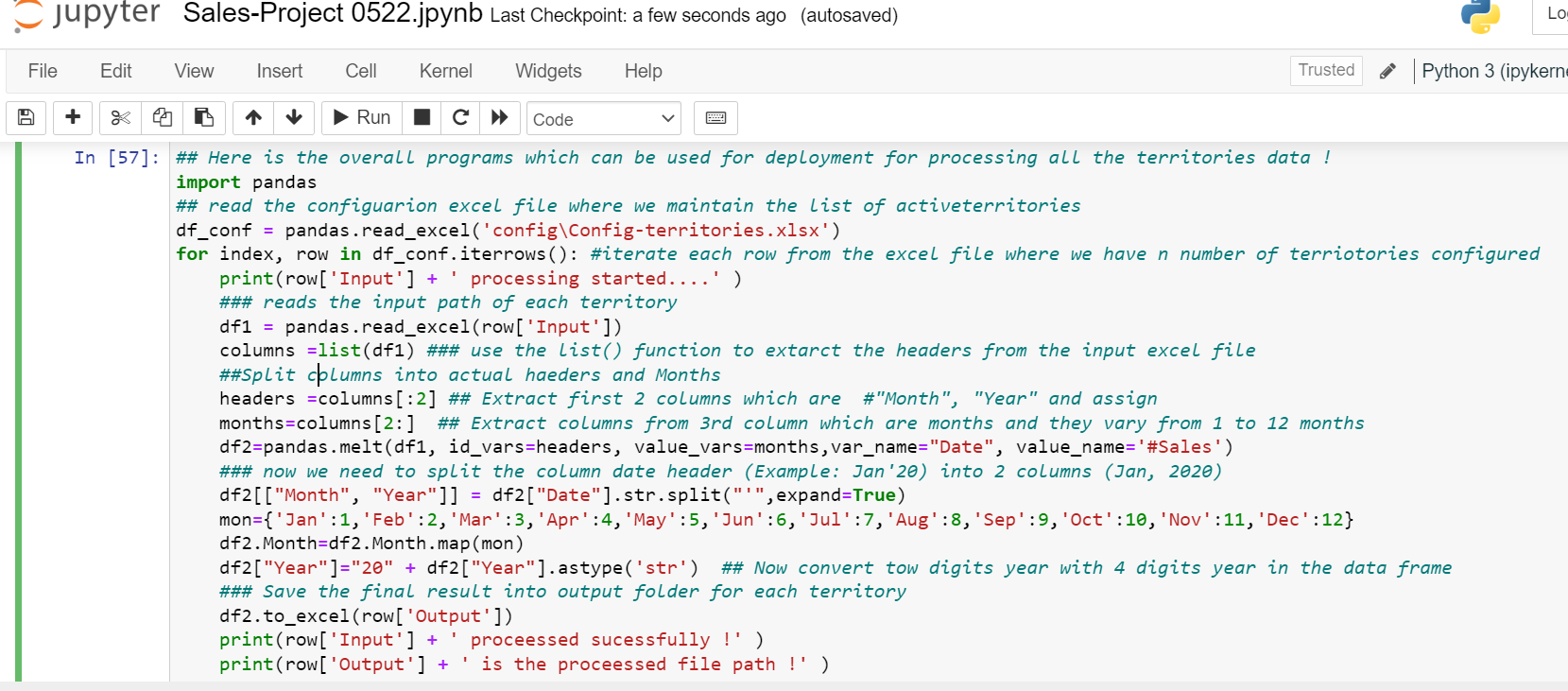
## Extending the prior logic to all territories

To process all the required territories data, we shall create a common configuration file containing the list of territories to process and their input and output paths.

The sample file is placed in the config folder Example (Config/ Config-territories.xlsx), and the sample data is as shown below where each row contains each territory’s input and output paths.



Now we can iterate each row of the configuration file using **for** loop and read the data file and process as shown below. iterrows**()** being used to iterate each row in the **for** loop.

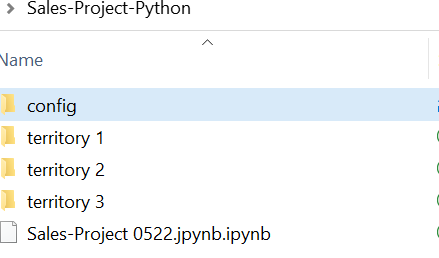


Conclusion:

1. Complex manual processes can be automated using Python libraries
2. Python library’s data frame melt functions can be used for unpivoting Data
3. transformation and converting data, such as unpivoting the data and data derivations, can be easily done

## *References*

The attached Sales-Project-Python.zip project contains the following folders and file.



* config: this folder contains the Excel file containing the list of territories that needs to be processed
* territory 1/2/3: contains in, and out folders. “In” folder should the raw data in a pivoted format
* Sales-Project 0522.jpynb is the Juypter note book to run the whole logic explained in this document

#### Input sample data



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Territory | Product | Jan'22 | Feb'22 | Mar'23 | Apr'22 | May'22 | Jun'22 |
| Territory 1 | Product 1 | 1000 | 1100 | 1200 | 1300 | 1400 | 1500 |
| Territory 1 | Product 2 | 2000 | 2100 | 2200 | 2300 | 2400 | 2500 |
| Territory 1 | Product 3 | 3000 | 3100 | 3200 | 3300 | 3400 | 3500 |

#### Python - Jupyter notebook for execution

#### Output sample file



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Territory** | **Product** | **Date** | **#Sales** | **Month** | **Year** |
| **0** | Territory 1 | Product 1 | Jan'22 | 1000 | 1 | 2022 |
| **1** | Territory 1 | Product 2 | Jan'22 | 2000 | 1 | 2022 |
| **2** | Territory 1 | Product 3 | Jan'22 | 3000 | 1 | 2022 |
| **3** | Territory 1 | Product 1 | Feb'22 | 1100 | 2 | 2022 |
| **4** | Territory 1 | Product 2 | Feb'22 | 2100 | 2 | 2022 |
| **5** | Territory 1 | Product 3 | Feb'22 | 3100 | 2 | 2022 |
| **6** | Territory 1 | Product 1 | Mar'23 | 1200 | 3 | 2023 |
| **7** | Territory 1 | Product 2 | Mar'23 | 2200 | 3 | 2023 |
| **8** | Territory 1 | Product 3 | Mar'23 | 3200 | 3 | 2023 |
| **9** | Territory 1 | Product 1 | Apr'22 | 1300 | 4 | 2022 |
| **..** | … | … | … | … | … | … |

#### Juypter Notebook Installation Reference

<https://jupyter.org/install>

#### Pandas Melt function

<https://pandas.pydata.org/docs/reference/api/pandas.melt.html>

#### Pandas Data frames

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html