**Tornado**

**A Python Project**

**Submitted By:**

Devvrat Vaidya

Dolly Bagaria

Ishant Tiwari

Pramodh Naryan L

# Problem Statement

To design a python application which will be able to open up 2 Excel files. These files will both be data tables laid out in a similar format as the below mentioned examples.

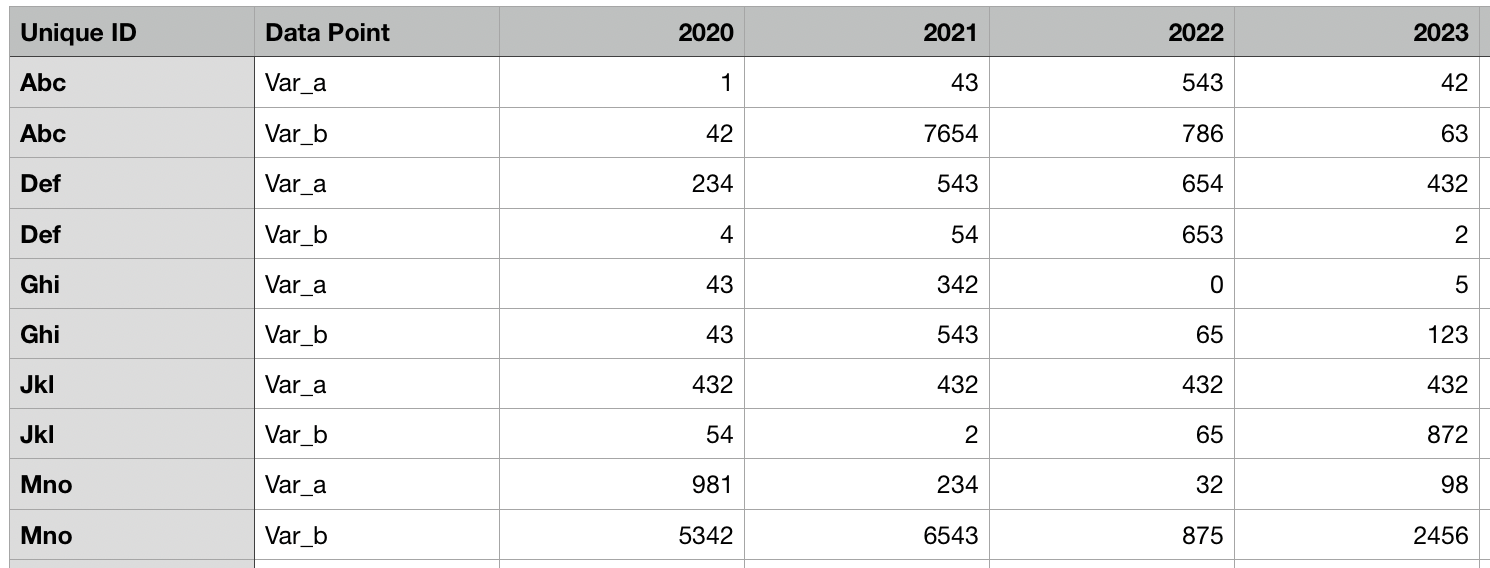


Fig 1 : Table 1 for comparison

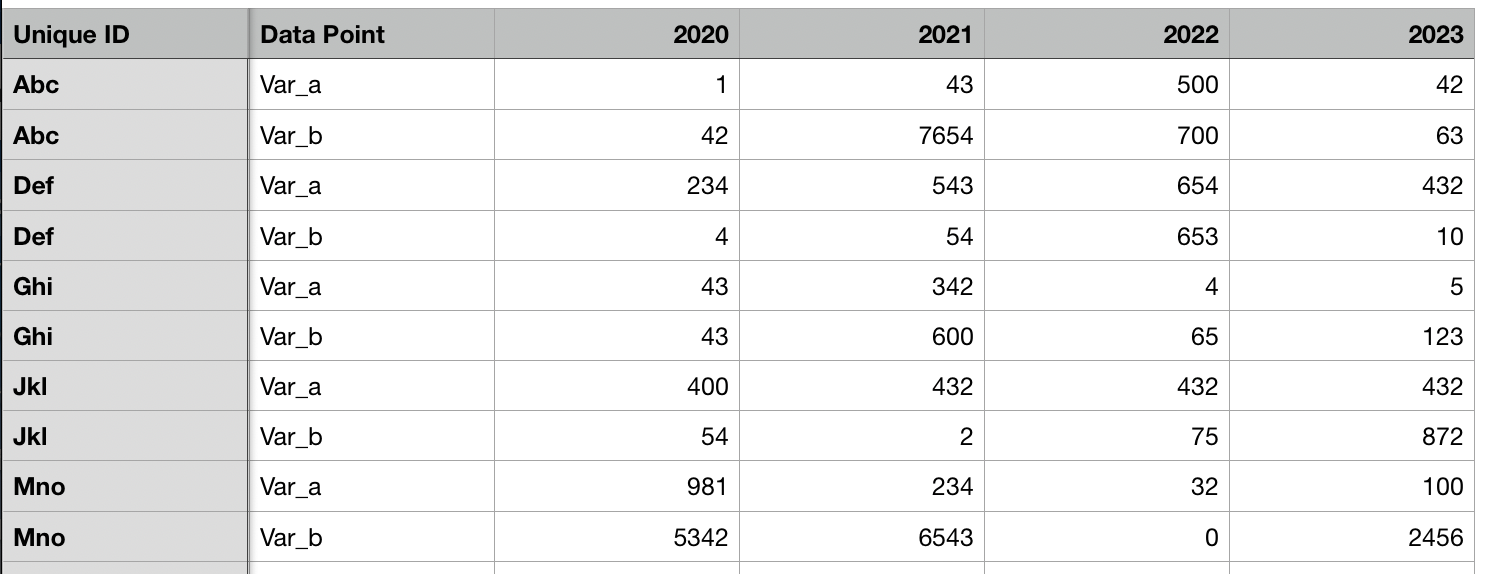


Fig 2 : Table 2 for comparison

It should to be able to tell for each ID what Data Point what has changed with respect to each other. We will be highlighting all the updated rows and columns using a special formatting (in this case the color changes to red).

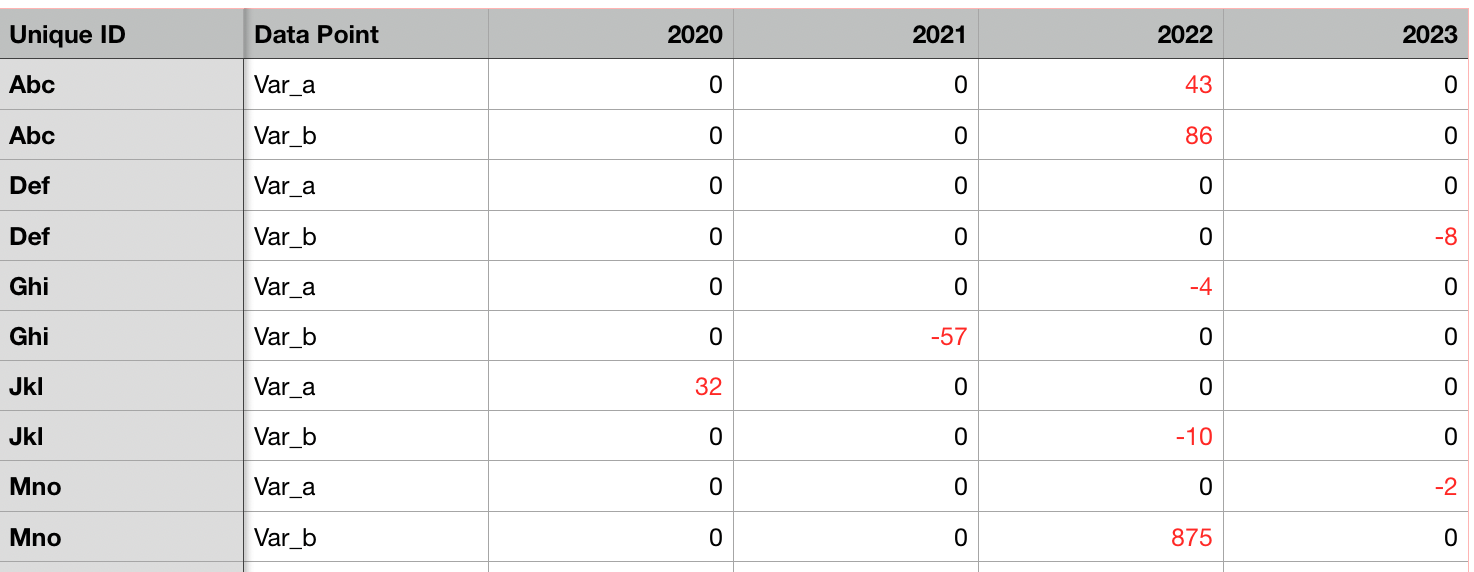


Fig 3 : Comparison Table

# Technologies Used

The project uses **Python 3** as the coding platform and uses two libraries i.e

1. **Pandas -**  an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming
2. **OpenPYXL -**  is a Python library to read/write Excel 2010 xlsx/xlsm/xltx/xltm files. It was born from lack of existing library to read/write natively from Python

# Scope of the Project

The project will revolve around comparing two excel files and highlighting out the differences to compare values in multiple scenarios, and making a real world application out of it.

# Source Code

## Implementation using Pandas

1. **Import**

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

1. **Assigning the excel sheets as inputs in python**

old=pd.read\_excel('excel\_old.xlsx')

new=pd.read\_excel('excel\_new.xlsx')

final=pd.read\_excel('excel\_new.xlsx')

1. **A boolean matrix is generated using the comparison in-built function between the two generated DataFrames**

comparison\_values = old.values == new.values

print (comparison\_values)

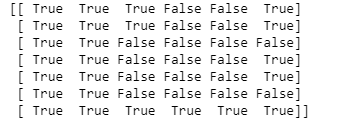


Fig 4 : Boolean Matrix

1. **Looking for rows and columns with ‘False’ values**

rows,cols=np.where(comparison\_values==**False**)

1. In [5]:

**for** item **in** zip(rows,cols):

final.iloc[item[0], item[1]] = '**{}**'.format(new.iloc[item[0], item[1]]-old.iloc[item[0], item[1]])

1. **After subtracting the false column values**

Final

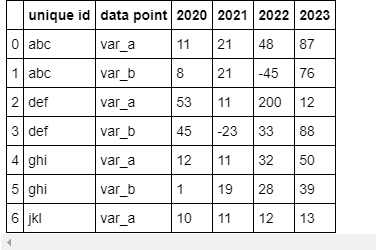


Fig 5: modified excel sheet

1. **Looking for ‘True’ values**

rows1,cols1=np.where(comparison\_values==**True**)

**for** item **in** zip(rows1,cols1):

**if** (type(old.iloc[item[0],item[1]])!=str):

final.iloc[item[0], item[1]] = '**{}**'.format(old.iloc[item[0], item[1]]-new.iloc[item[0], item[1]])

final

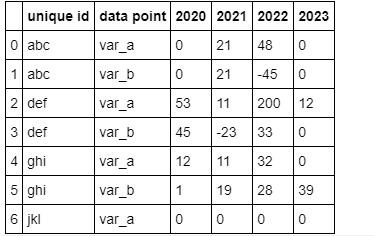


Fig 6: Modified excel sheet

1. **Colouring the values**

def color\_negative\_red(value):

color = 'red' if (value!=0 and type(value)!=str) else 'black'

return 'color: %s' %color

df\_display=final.style.applymap(color\_negative\_red)

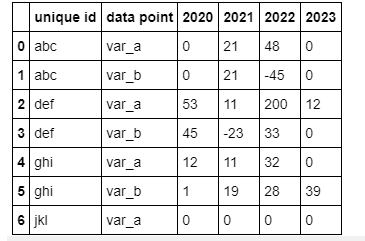


Fig 7: Modified excel sheet

1. **Comparing values**

comparison\_values = old == new

comparison\_values

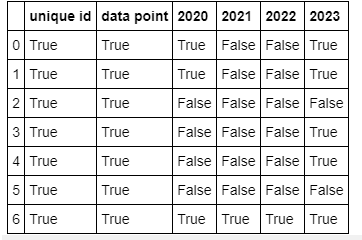


Fig 8: Modified excel sheet

1. **List**

lista=list(old.columns)

listb=list(new.columns)

lista

['unique id', 'data point', 2020, 2021, 2022, 2023]

listb

['unique id', 'data point', 2020, 2021, 2022, 2023]

lista==listb

True

old=old.sort\_values(by=['unique id','data point'])

new=new.sort\_values(by=['unique id','data point'])

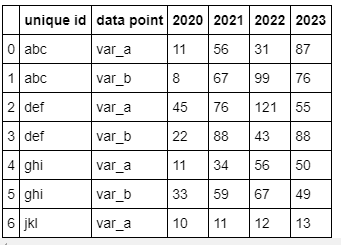


Fig 9: Final excel sheet

## Implementation using openpyxl

1. **Importing pyxl libraries**

from openpyxl import Workbook

from openpyxl import load\_workbook

from openpyxl.styles import Fill,Font,Color,colors

1. **Creating a workbook object from old excel sheet**

wold = load\_workbook('excel\_old.xlsx')

1. **Creating a workbook object from new excel sheet**

wnew = load\_workbook('excel\_new.xlsx')

1. **Creating a workbook object from final excel sheet**

wfinal = Workbook()

1. **Activating the worksheet and giving the detail as sheet1**

wfinalsheet=wfinal.active

wfinalsheet.title='sheet1'

**f) Finding the maximum rows and columns of old sheets**

sheet1=wold.active

sheet2=wnew.active

max\_row1=sheet1.max\_row

max\_column1=sheet1.max\_column

1. **Output of max rows and max columns**

max\_row1

Out[69]:

8

max\_column1

6

1. **Getting the headers of old sheet and putting it in final sheet and saving it**

**for** i **in** range(1,2):

**for** j **in** range(1,max\_column1+1):

cell\_obj1=sheet1.cell(row=i,column=j)

wfinalsheet.cell(row=i, column=j).value = cell\_obj1.value

wfinal.save('final\_excel.xlsx')

1. **When the values are non-string,adding the values of new excel from old excel to final sheet**

**for** i **in** range(2,max\_row1+1):

**for** j **in** range(1,max\_column1+1):

cell\_obj1=sheet1.cell(row=i,column=j)

cell\_obj2=sheet2.cell(row=i,column=j)

**if** type(cell\_obj1.value) == str **and** type(cell\_obj2.value) == str :

wfinalsheet.cell(row=i, column=j).value = cell\_obj1.value

**else**:

temp=cell\_obj2.value-cell\_obj1.value

**if**(temp!=0):

wfinalsheet.cell(row=i, column=j).font = Font(color=colors.RED)

wfinalsheet.cell(row=i, column=j).value = temp

1. **Saving the file**

wfinal.save('final\_excel.xlsx')

# Screenshots

Fig 5 : excel\_file\_new.xlsx

Fig 6 : excel\_file\_old.xlsx

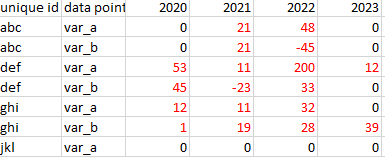


Fig 7 : excel\_file\_diff.xlsx

# Result

The files have been compared using execution of two technologies and a new file has been generated in all instances.