Backslash

Python Project

Avirup Roy Chowdhury

Pranathi Varma M.

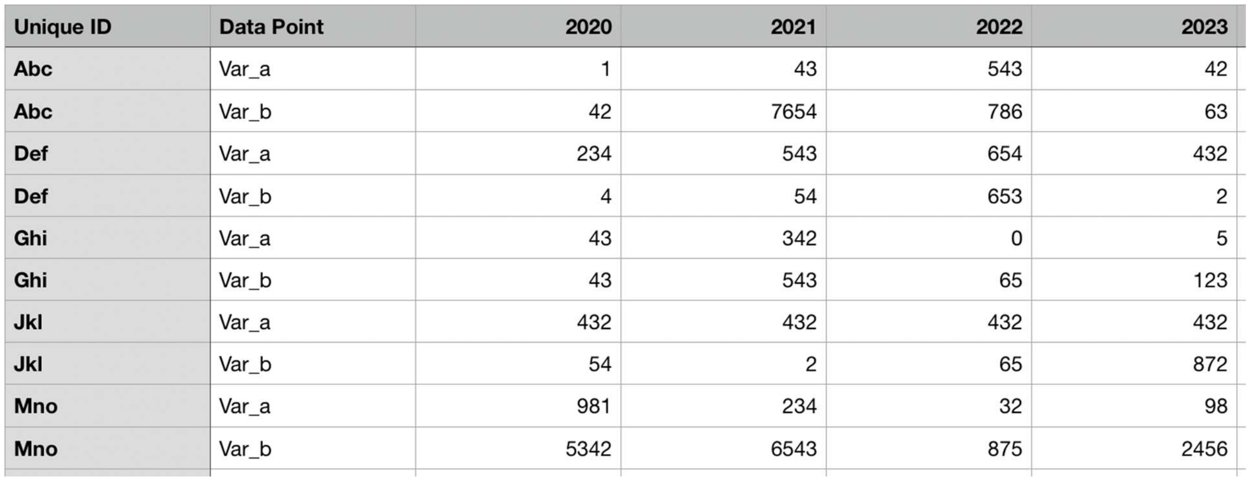
Niharika Gupta

Chaitanya Mohite

Diptajit Chaurangi

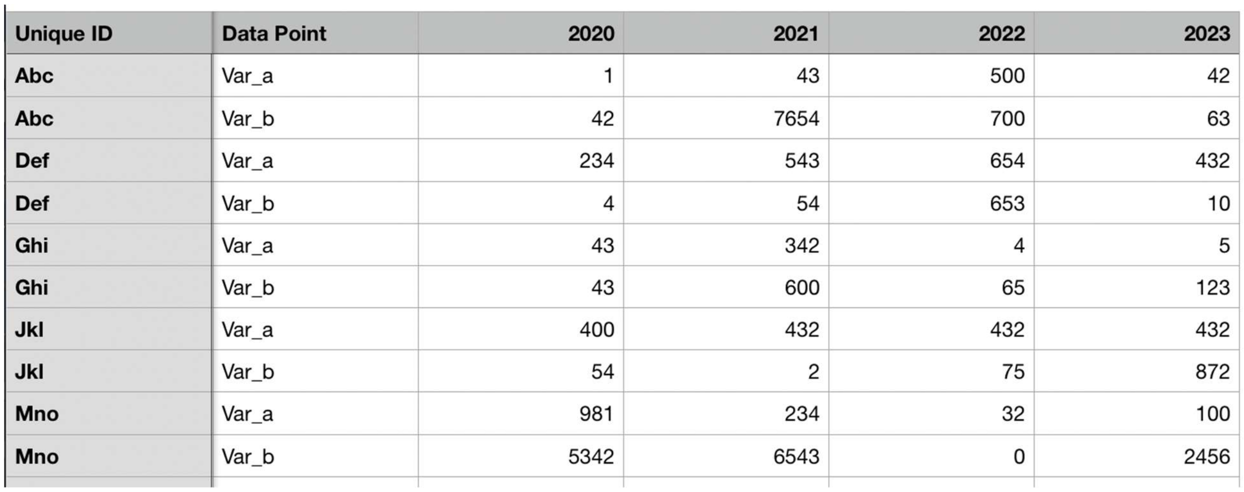
Problem Statement

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.



Unique ID	Data Point	2020	2021	2022	2023
Abc	Var_a	1	43	543	42
Abc	Var_b	42	7654	786	63
Def	Var_a	234	543	654	432
Def	Var_b	4	54	653	2
Ghi	Var_a	43	342	0	5
Ghi	Var_b	43	543	65	123
Jkl	Var_a	432	432	432	432
Jkl	Var_b	54	2	65	872
Mno	Var_a	981	234	32	98
Mno	Var_b	5342	6543	875	2456

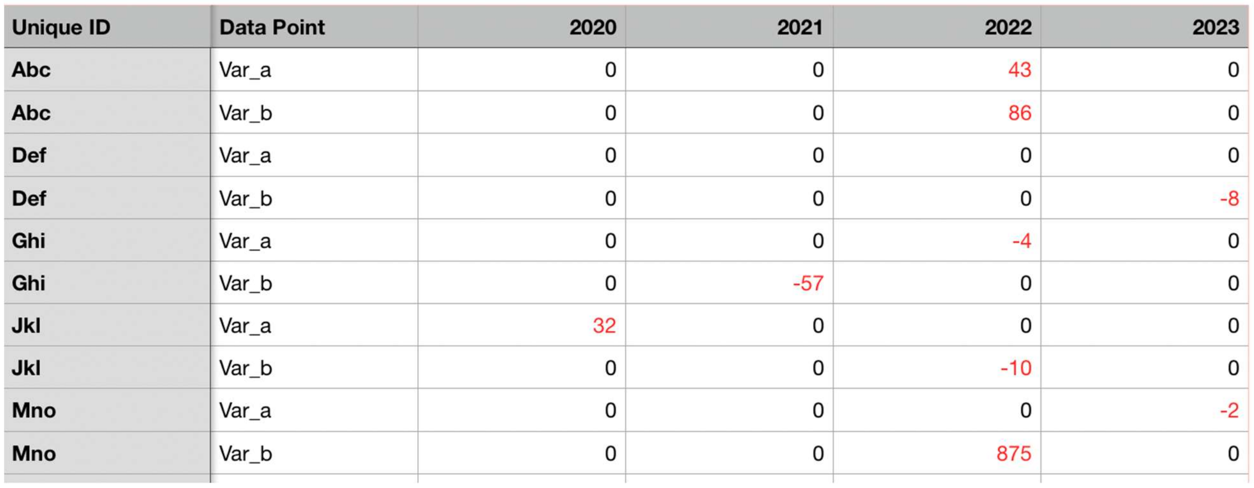
Fig 1 : Table 1 for comparison



Unique ID	Data Point	2020	2021	2022	2023
Abc	Var_a	1	43	500	42
Abc	Var_b	42	7654	700	63
Def	Var_a	234	543	654	432
Def	Var_b	4	54	653	10
Ghi	Var_a	43	342	4	5
Ghi	Var_b	43	600	65	123
Jkl	Var_a	400	432	432	432
Jkl	Var_b	54	2	75	872
Mno	Var_a	981	234	32	100
Mno	Var_b	5342	6543	0	2456

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).



Unique ID	Data Point	2020	2021	2022	2023
Abc	Var_a	0	0	43	0
Abc	Var_b	0	0	86	0
Def	Var_a	0	0	0	0
Def	Var_b	0	0	0	-8
Ghi	Var_a	0	0	-4	0
Ghi	Var_b	0	-57	0	0
Jkl	Var_a	32	0	0	0
Jkl	Var_b	0	0	-10	0
Mno	Var_a	0	0	0	-2
Mno	Var_b	0	0	875	0

Fig 3 : Comparison Table

Technologies Used

The project base and structure will be revolving around **Python 3** and will be implemented using 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.

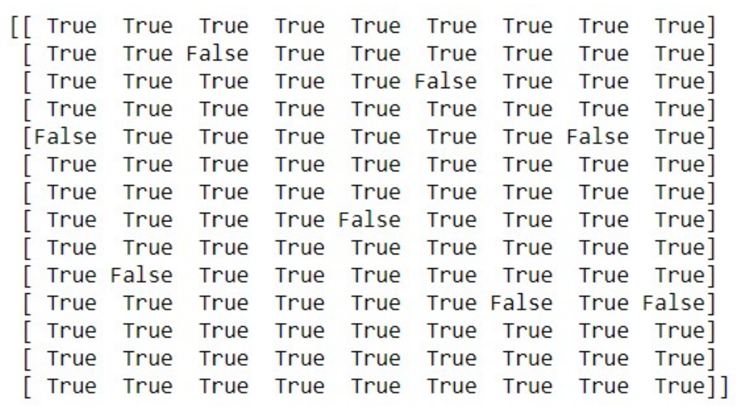
Source Code

1. Implementation using Pandas

a) Import

```
                        #importing excel to DataFrame
df1=pd.read_excel('excel_file_old.xlsx') #Df1 will represent the old file
df2=pd.read_excel('excel_file_new.xlsx') #Df2 will represent the new file
```

b) A boolean matrix is generated using the comparison in-built function between the two generated DataFrames



```
[[ True  True  True  True  True  True  True  True  True]
 [ True  True False  True  True  True  True  True  True]
 [ True  True  True  True  True False  True  True  True]
 [ True  True  True  True  True  True  True  True  True]
 [False  True  True  True  True  True  True False  True]
 [ True  True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True  True]
 [ True  True  True  True False  True  True  True  True]
 [ True  True  True  True  True  True  True  True  True]
 [ True False  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True False  True False]
 [ True  True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True  True]
 [ True  True  True  True  True  True  True  True  True]]
```

Fig 4 : Boolean Matrix

c) For different values the value is given as False and for similar values it is given as True

a. If value is True, we subtract the values and it becomes 0

```
rows,cols=np.where(comparison_values==True)
for item in zip(rows,cols):
   if (type(df1.iloc[item[0],item[1]])!= str):
   df3.iloc[item[0],item[1]]=df1.iloc[item[0],
```

b. If value is False, we subtract the values and it returns a non Zero value

```
rows,cols=np.where(comparison_values==False)
for item in zip(rows,cols):
  df3.iloc[item[0],item[1]]=df1.iloc[item[0],item[1]]
```

d) Finally the updates values are formatted for better clarity using color_negative_red()

```
def color_negative_red(val):
    """
    Takes a scalar and returns a string with
```

```
                the css property `'color: red'` for negative
                     strings, black otherwise.
                                """
    color = 'red' if (val != 0  and type(val) !=str) else 'black'
                     return 'color: %s' % color
```

e) It is exported into a new file

```
s.to_excel('./Excel_diff.xlsx',index=False,header=True)
```

2. Implementation using openpyxl

a) Setting FilePath for Three Excel Files (Comparison1, Comparison2, NewFile)

```
#set file path for the three excel files
filepath1="excel_file_old.xlsx"
filepath2="excel_file_new.xlsx"
filepath3="excel_file_diff.xlsx"
```

b) **Activate All the Sheets in the File**

```
#selecting the three excel sheets
sheet1=wb1.active
sheet2=wb2.active
sheet3=wb3.active
```

c) Will Evaluate Maximum number of rows and columns to generate operation size

```
#evaluating the maximum rows and columns
max_row1=sheet1.max_row
max_column1=sheet1.max_column
```

d) Will print the header(title) of each excel File

```
for i in range(1,2):
   for j in range(1,max_column1+1):
     cell_obj1=sheet1.cell(row=i,column=j)
     sheet3.cell(row=i, column=j).value = cell_obj1.v
                 alue
     wb3.save(filepath3)
```

e) **Will compare cell by cell for all the rows and columns and update accordingly**

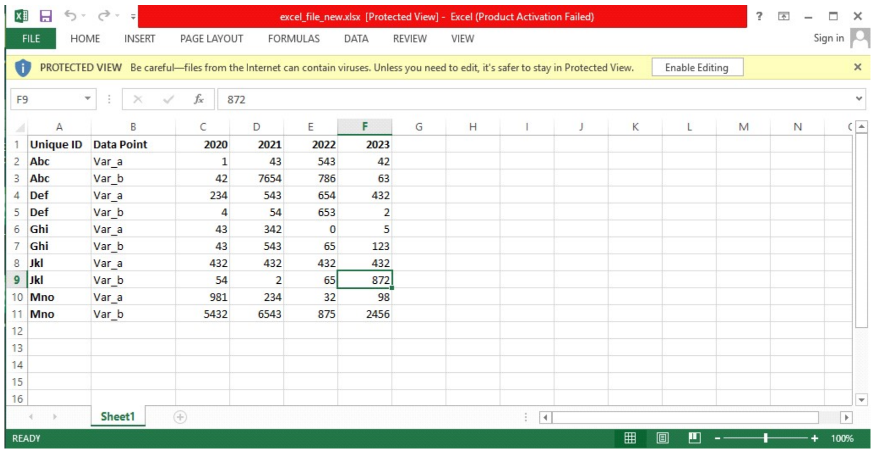
```
for i in range(2,max_row1+1):
        for j in range(1,max_column1+1):
             cell_obj1=sheet1.cell(row=i,column=j)
             cell_obj2=sheet2.cell(row=i,column=j)
             if type(cell_obj1.value) == str and type(cell_ob
                  j2.value) == str   :
                   sheet3.cell(row=i, column=j).value = cell_o
                         bj1.value
```

```
                    else:
                        temp=cell_obj1.value-cell_obj2.value
                        sheet3.cell(row=i, column=j).value = temp
```

f) Now we will format the updated values with openpyxl.styles

```
                    if temp!=0:
                            sheet3.cell(row=i, column=j).font = F
                 ont(color=colors.RED)
```

Screenshots



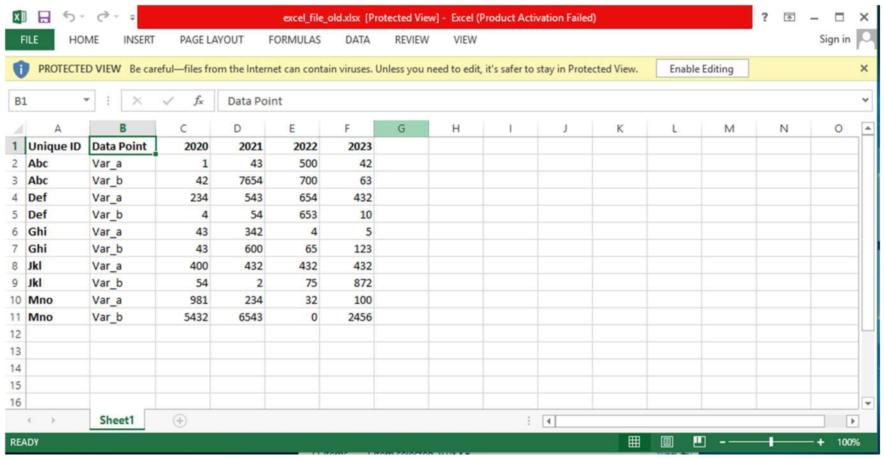
excel_file_new.xlsx [Protected View] - Excel (Product Activation Failed)
FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW
Sign in
Enable Editing
Sheet1
READY
100%

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

F9 872

Unique ID	Data Point	2020	2021	2022	2023
Abc	Var_a	1	43	543	42
Abc	Var_b	42	7654	786	63
Def	Var_a	234	543	654	432
Def	Var_b	4	54	653	2
Ghi	Var_a	43	342	0	5
Ghi	Var_b	43	543	65	123
Jkl	Var_a	432	432	432	432
Jkl	Var_b	54	2	65	872
Mno	Var_a	981	234	32	98
Mno	Var_b	5432	6543	875	2456

Fig 5 : excel_file_new.xlsx



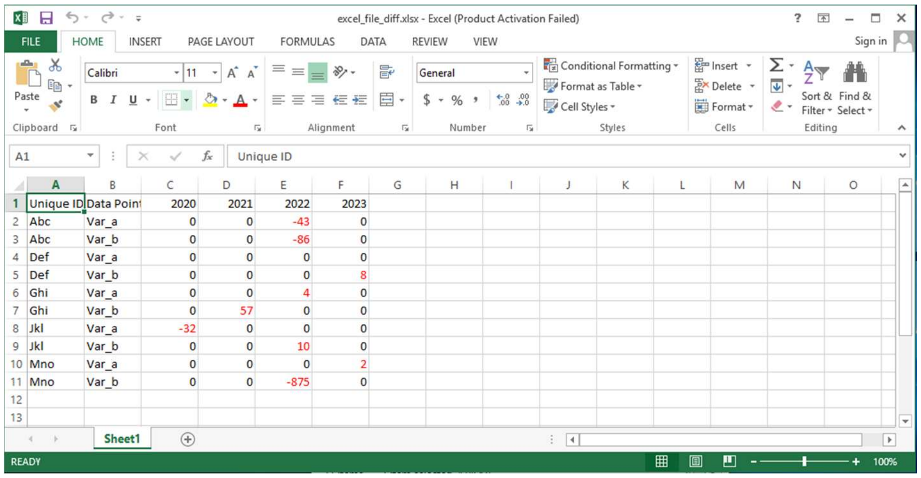
excel_file_old.xlsx [Protected View] - Excel (Product Activation Failed)
FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW
Sign in
Enable Editing
Sheet1
READY
100%

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

B1 Data Point

Unique ID	Data Point	2020	2021	2022	2023
Abc	Var_a	1	43	500	42
Abc	Var_b	42	7654	700	63
Def	Var_a	234	543	654	432
Def	Var_b	4	54	653	10
Ghi	Var_a	43	342	4	5
Ghi	Var_b	43	600	65	123
Jkl	Var_a	400	432	432	432
Jkl	Var_b	54	2	75	872
Mno	Var_a	981	234	32	100
Mno	Var_b	5432	6543	0	2456

Fig 6 : excel_file_old.xlsx



A1 | Unique ID

Unique ID	Data Point	2020	2021	2022	2023
Abc	Var_a	0	0	-43	0
Abc	Var_b	0	0	-86	0
Def	Var_a	0	0	0	0
Def	Var_b	0	0	0	8
Ghi	Var_a	0	0	4	0
Ghi	Var_b	0	57	0	0
Jkl	Var_a	-32	0	0	0
Jkl	Var_b	0	0	10	0
Mno	Var_a	0	0	0	2
Mno	Var_b	0	0	-875	0

Sheet1

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.

Libraries and Functions Used :

1. **Pandas** –
    a. **read_excel** - Read an Excel file into a pandas DataFrame.
    b. **iloc()** - Purely integer-location based indexing for selection by position.
    c. **style() –** to apply conditional formatting
        i. **applymap()** - Apply a function to a Dataframe elementwise
    d. **to_excel()** - Write object to an Excel sheet.

2. **Numpy –**
    a. **where()** - Return elements chosen from *x* or *y* depending on *condition*.

3. **Python –**
    a. ***copy()*** - This module provides generic shallow and deep copy operations
    b. ***zip()*** - Make an iterator that aggregates elements from each of the iterables.
    c. ***type()*** - With one argument, return the type of an *object*.
    d. ***range()*** - Rather than being a function, range is actually an immutable sequence type

4. **User-defined Functions –**
    a. ***color_negative_red() –*** adds red colour to the font.

5. **openpyxl**
    a. ***Workbook -*** A workbook is always created with at least one worksheet.
    b. ***Load_Workbook –*** Loading a workbook into an object for i/o operations
    c. ***Styles() -*** Styles are used to change the look of your data while displayed on screen.
        i. **Fill -** fill to set a pattern or color gradient
        ii. **Font -** font to set font size, color, underlining, etc.
        iii. **Color –** to import colour codes
        iv. **Colors –** to import colour codes
    d. ***Save() -*** The simplest and safest way to save a workbook
    e. ***Active() –*** To activate a workbook
    f. ***Max_row –*** count maximum rows in a worksheet
    g. ***Max_column –*** count maximum columns in a worksheet
    h. ***Cell(row,column) –*** returns cell in the actual excel sheet