Work-8

Update the form from datasets attached in Outlook email

Installation

- Download and Install Anaconda

> NOTE: Don't forget to tick the checkbox corresponding to "Add to path". This will enable using `conda` in the terminal.

- Open the terminal and check for following packages

```
pandas
os
pywin32
```

- If not found, then run these 2 commands in terminal:
  - `pip install pandas`
  - `pip install os`
  - `pip install pywin32`

Coding

Modules

- Import packages

```
import win32com.client
import os
import pandas as pd
```

- Download mail attachments for subject (== "test")

```
outlook = win32com.client.Dispatch("Outlook.Application").GetNamespace("MAPI")
inbox = outlook.GetDefaultFolder(6) # "6" refers to the index of a folder - in this case the inbox. You can change that number to
messages = inbox.Items
message = messages.GetFirst()
subject = message.Subject

# path of directory where attached files are to be saved locally
get_path = 'I:\\github_repos\\work-8\\download_mail'
# print(len(messages))      # length of the messages in "Inbox" folder

# Loop in all the Inbox messages
for m in messages:
    if m.Subject == "test":     # check if the subject == "test"

        # print (message)
        attachments = message.Attachments
        num_attach = len([x for x in attachments])
        for x in range(1, num_attach+1):
            attachment = attachments.Item(x)
            attachment.SaveASFile(os.path.join(get_path,attachment.FileName))
        # print (attachment)
        message = messages.GetNext()

    else:
        message = messages.GetNext()
```

- Traverse in `download_mail` folder, fetch into `df_attachments` list, Define `df1` & `df2`

```
df_attachments = []
for dirname, dirpath, files in os.walk("./download_mail/"):
    for file in files:
        # print(file)
        df_attachments.append(pd.ExcelFile("./download_mail/" + file).parse(0))      # parse sheet_name - 0 i.e. 1st sheet

# define `df1` i.e. from downloaded mail attachments (in excel format)
df1 = df_attachments[0]
# print(df1.head())

# define `df2` i.e. output
df2 = pd.ExcelFile("./output/Attachment_1564702282.xlsx").parse(0)              # parse sheet_name - 0 i.e. 1st sheet
# print(df2)
```

- Create columns list `col_1` & `col_2`

```
col_1 = ["Order ID", "Billing First Name", "Shipping Address 1", "Shipping City",  "Shipping State",
        "Shipping Postcode", "Shipping Country", "SKU", "Purchased"]
col_2 = ["ClientOrderCode", "FirstName", "Address1", "City", "Province", "PostCode", "Country", "OrderSourceSKU", "ProductNum"]
```

- define `df1_col1` & `df2_col2`

```
# define `df1_col1` for `df1` as per the required columns i.e. col_1
df1_col1 = df1[col_1]
# print(df1_col1.head())

# define `df2_col2` for `df2` as per the required columns i.e. col_2
df2_col2 = df1_col1             # copy the content from `df1_col1` into `df2_col2`
df2_col2.columns = col_2        # add columns
# print(df2_col2.head())
```

- Clean `df2` and replace the contents of desired column's cells

```
# drop all rows of `df2`
df2.drop(df2.index, inplace=True)

"""
NOTE: Before applying this, `df2` needs to be cleared first.
and then copy respective columns data into from `df2_col2` into `df2`
"""
for c in col_2:
    df2[c] = df2_col2[c]

# print(df2)
```

- Display the output of `df2` into a separate excel

```
# print the `df2` (w/o index) into already present excel file
df2.to_excel("./output/Attachment_1564702282.xlsx", index= False)
```

Execution

There are 2 ways to run this:

- **M-1: Unix OS** - run the `run.sh`
- **M-2: Windows OS** - run the `run.bat`

Output

The file Output Excel File is modified after execution.