AUTOMATIZACION DE INFORMES DE VENTAS CON PYTHON

MORO MARÍA EMILIA

DNI: 39570398

INTRODUCCIÓN

La elección de la temática del presente trabajo surgió debido a la necesidad periódica de realizar informes de gestión de ventas, tanto para los usuarios internos de la empresa donde trabajo como para usuarios externos (entidades financieras).

Si bien es una tarea que realizaba una vez al mes, al cerrarse el período mensual de ventas, requería de un tiempo considerable ya que, a partir de un reporte de comprobantes de ventas que descargaba desde el sistema de gestión "Tango", en archivo de Excel, debía:

- Clasificar los códigos de artículos (productos y conceptos) en segmentos de actividad
- Excluir los conceptos que no me interesaban para armar los informes
- Totalizar las ventas de los segmentos por importes y por volumen

Y a su vez, me encontraba con el problema de que las ventas estaban en valores nominales, es decir, expresados en moneda heterogénea, a la fecha de cada factura. Esto no me permitía poder realizar análisis de las ventas por importes y debía recurrir a expresarlas en moneda homogénea, reexpresando todos los importes, según los índices de precios, a un momento determinado.

Para lograr automatizar esta tarea se me ocurrió desarrollar una serie de scripts en Python que me permita, a partir de 3 bases de datos (el reporte de ventas que me proporciona el sistema, un archivo con la clasificación de los productos y conceptos y el archivo con los índices de precios), crear un DataFrame que contenga las ventas ya clasificadas en segmentos y con los importes reexpresados a moneda homogénea. Con esta base de datos consolidada, armando distintas tablas dinámicas (también en Python) se pueden crear diversos informes de gestión, visualizarlos y analizarlos.

FUNCIONES, MÉTODOS Y OTRAS HERRAMIENTAS DE PYTHON UTILIZADAS

- read_excel(), merge(), concat(), to_datetime, groupby(), pivot_table() y to_excel de la librería pandas
- BeautifulSoup() y find_all() de la librería bs4,
- get() de la librería requests
- split(), loc[], len(), rage(), drop()
- bucle for, if, try
- entre otros.

DESARROLLO DEL CÓDIGO

- Parto utilizando la librería **pandas** y su función **read_excel()** para leer y trabajar con el reporte de ventas en archivo de Excel que genera el sistema de gestión (Tango), asignando, al mismo tiempo, nuevos nombres a las columnas del reporte para no tener problema con los espacios ni los paréntesis al manipular el DataFrame. También leo el archivo de Excel que contiene la clasificación en segmentos de actividad de los artículos de ventas.

```
import pandas as pd
directorio_origen="C:/Users/Usuario/Downloads/"
nuevos_nombres_col=[
                    "Fecha_de_emision",
                    "Tipo_comprobante",
                    "Nro._comprobante",
                    "Cod._vendedor",
                    "Nombre_Vendedor",
                    "Moneda",
                    "Cod._cliente",
                    "Razon_social",
                    "Nombre_provincia",
                    "Cod._pais",
                    "Nombre_pais",
                    "Cod._Articulo",
                    "Descripcion",
                    "Cantidad",
                    "Total",
]

df_REPORTE_VENTAS_2023=pd.read_excel(directorio_origen+"REPORTE_VENTAS_2023.xlsx", decimal=',', names=nuevos_nombres_col)
df_Clasificacion_articulos=pd.read_excel(directorio_origen+"Clasificacion articulos.xlsx")

df_REPORTE_VENTAS_2023['Total'] = round(df_REPORTE_VENTAS_2023['Total'],2)

print(df_REPORTE_VENTAS_2023)
print(df_Clasificacion_articulos)
```

- Luego creo un nuevo df, para agregarle al df de ventas la clasificación. Para hacer la unión de los datos de estos dos DataFrames utilizo la función **merge()**, tomando como columna en común para unir los dos objetos las que contienen los códigos de los artículos.

```
df_VENTAS_2023_CLASIFICADAS=pd.merge(df_REPORTE_VENTAS_2023,
                                     df_Clasificacion_articulos,
                                     on="Cod._Articulo",
                                     how="left")
print(df_VENTAS_2023_CLASIFICADAS)
```

  Como parámetro how asigno “left” para que el nuevo DataFrame unido contenga todos los registros del primer df (df_REPORTE_DE_VENTAS_2023), independientemente de si están todos los códigos de los artículos clasificados o no. Esto lo hago así para poder detectar cuando se incorporen nuevos productos y tenga que clasificarlos.
  Anexo al final del trabajo el desarrollo de un script para identificar la incorporación de nuevos productos y actualizar el DataFrame.

- Agrego al nuevo DataFrame dos columnas con el año y el mes al que corresponden las ventas:

```
df_VENTAS_2023_CLASIFICADAS["anio"]=df_VENTAS_2023_CLASIFICADAS.Fecha_de_emision.dt.year
df_VENTAS_2023_CLASIFICADAS["mes"]=df_VENTAS_2023_CLASIFICADAS.Fecha_de_emision.dt.month
print(df_VENTAS_2023_CLASIFICADAS)
```

- Para obtener los índices a utilizar para la reexpresión de los importes de las ventas, utilizando las funciones **BeautifulSoup()** de la librería bs4, **get()** de la librería requests y **read_excel()** de la librería pandas y el método **find_all()**. Con estas herramientas accedo a un archivo de Excel que se encuentra publicado en la página de la FACPCE y creo un DataFrame. Este contiene los meses de todos los años desde enero de 1993 y el respectivo IPC.

```
import requests
import pandas as pd    #no es necesario porque ya lo importé antes
from bs4 import BeautifulSoup

url_indices_facpce="https://www.facpce.org.ar/indices-facpce/"

soup = BeautifulSoup(requests.get(url_indices_facpce).text,features="lxml")

for link in soup.find_all('a'):
    try:
        url_enlaces = link.get('href')
        url_array = url_enlaces.split(".")
        extension = url_array[-1]
        if extension == "xlsx":
            archivo_indices_RT6 = requests.get(url_enlaces).content
            DF_INDICES_RT6FACPCE = pd.read_excel(archivo_indices_RT6, names=["FECHA","IPC"])
    except:
        print()

print(DF_INDICES_RT6FACPCE)
```

- Cambio el formato de las fechas contenidas en la primera columna de este nuevo df y agrego dos columnas más con el año y el mes al que corresponden los índices. Antes de esto, con la función **iloc[]** excluyo las filas que tienen valores que no me permiten manejar el formato de fecha en esa columna.

```
DF_INDICES_RT6FACPCE=DF_INDICES_RT6FACPCE.iloc[2:-3]

DF_INDICES_RT6FACPCE["FECHA"]=pd.to_datetime(DF_INDICES_RT6FACPCE['FECHA'])
DF_INDICES_RT6FACPCE["ANIO"]=DF_INDICES_RT6FACPCE.FECHA.dt.year
DF_INDICES_RT6FACPCE["MES"]=DF_INDICES_RT6FACPCE.FECHA.dt.month
print(DF_INDICES_RT6FACPCE)
```

- Creo un nuevo DataFrame, para agregarle al df de ventas clasificadas los índices de reexpresión correspondientes. Para realizar la unión de los datos de estos dos df utilizo nuevamente la función **merge()**, tomando como condición dos columnas de cada df, las que contienen el año y el mes.
  Así, creo una lista para cada df indicando estas dos columnas, y las paso como parámetro left_on y right_on para indicar los objetos de unión.

```
columnas_df_ventas= ["anio", "mes"]
columnas_df_indices= ["ANIO", "MES"]
DF_VENTAS_2023_INDICES=pd.merge(df_VENTAS_2023_CLASIFICADAS,
                                DF_INDICES_RT6FACPCE,
                                left_on=columnas_df_ventas,
                                right_on=columnas_df_indices,
                                how='inner')
print(DF_VENTAS_2023_INDICES)
```

- Finalmente, creo una nueva columna en el informe de ventas donde irán los importes de las ventas reexpresados. Comienzo asignándole como valor 0 en todas las filas, defino el índice que voy a utilizar para ajustar las ventas (índice del mes al que las quiero reexpresar) y, luego,

utilizo la función **range()** y **len()** para definir el rango que voy a reexpresar, que va desde el primer renglón hasta el largo de la tabla.
Así, con el bucle **for** y función **loc[]** completo las filas del total reexpresado.
Este DataFrame es la base para crear, a partir de tablas dinámicas, distintos informes de ventas, según las necesidades que se tengan.

```
DF_VENTAS_2023_INDICES["Total_Reexpresado"]=0
IPC_BASE = 3533.1922  #Diciembre 2023
largo = range(0,len(DF_VENTAS_2023_INDICES))
for i in largo:
    DF_VENTAS_2023_INDICES.loc[i,'Total_Reexpresado'] = round(DF_VENTAS_2023_INDICES.loc[i,'Total']/DF_VENTAS_2023_INDICES.loc[i,'IPC']*IPC_BASE,2)
print(DF_VENTAS_2023_INDICES)
```

INFORMES DE VENTAS

- INFORMES DE VENTAS NETAS Y EN VOLUMEN MENSUALES, POR SEGMENTOS DE ACTIVIDAD:

Se pueden realizar informes que visualicen las ventas mensuales, tanto en unidades monetarias como en volumen.

Para esto, comienzo limpiando DataFrame de los datos que no me interesan para este informe, ya sea porque no son conceptos de ventas/bonificaciones (ej: recupero de cheques rechazados, recomposición de saldos de clientes, etc) o porque su “cantidad” no implica una variación en el volumen de productos vendidos.

```
DF_VTAS_NETAS_DATOS_LIMPIOS=DF_VENTAS_2023_INDICES.drop(DF_VENTAS_2023_INDICES[(DF_VENTAS_2023_INDICES["Clasificacion"]=="Otros")].index)


valores_a_eliminar = ["Bonif. C. de frutas y vegetales",
                      "Bonif. Conservas de pescados",
                      "Bonif. Espumosos frutados y otras bebidas",
                      "Otros",
                      "Otros descuentos y bonificaciones"]

DF_VTAS_VOLUMEN_DATOS_LIMPIOS = DF_VENTAS_2023_INDICES[~DF_VENTAS_2023_INDICES['Clasificacion'].isin(valores_a_eliminar)]
```

De esta forma, obtengo un DataFrame limpio para el informe de ventas netas y otro para el informe de ventas en volumen.

Probé realizar estos informes con el método **groupby()** de pandas, agrupando por mes y por clasificación los totales reexpresados y las cantidades, pero el resultado obtenido no me resultaba práctico visualmente.

Recurrí entonces al método **pivot_table()** de pandas, que me permitió resumir los datos de una forma más sencilla de comparar, agrupando los valores (ventas reexpresadas y cantidades) por segmento de actividad en el eje de las filas y por mes en el eje de las columnas. Además, completé los valores nulos con 0, pasando como parámetro: fill_value=0, e indexé las filas según un orden específico con el método **reindex().**

INFORME DE VENTAS NETAS MENSUALES POR SEGMENTO:

```
pd.options.display.float_format = '{:.2f}'.format #para que los Nros se muestren con dos decimales y no con notacion exponencial


orden_clasificaciones_vn = [
    'Conservas de pescados',
    'Bonif. Conservas de pescados',
    'Conservas de frutas y vegetales',
    'Bonif. C. de frutas y vegetales',
    'Exportación de conservas de frutas',
    'Espumosos frutados y otras bebidas',
    'Bonif. Espumosos frutados y otras bebidas',
    'Otras ventas',
    'Otros descuentos y bonificaciones',
    'Total',
]

VTAS_NETAS_MENSUALES_POR_SEGMENTO=DF_VTAS_NETAS_DATOS_LIMPIOS.pivot_table(index = "Clasificacion",
                                                columns = "MES",
                                                values = "Total_Reexpresado",
                                                aggfunc = "sum",
                                                fill_value=0,
                                                margins=True,
                                                margins_name='Total').reindex(orden_clasificaciones_vn).fillna(value=0)

VTAS_NETAS_MENSUALES_POR_SEGMENTO
```

MES	1	2	3	4	5	6	7	8	9	10	11	12	Total
Clasificacion													
Conservas de pescados	938433971.66	746405314.10	713051299.68	745268445.44	552806317.75	457006408.19	346675634.83	613673509.75	407800205.63	193283904.05	78284469.85	668823420.24	6461512901.17
Bonif. Conservas de pescados	-7486134.23	-9215067.57	-2690598.95	-2134869.76	-833316.24	-5130753.24	-838322.04	-1313683.38	-473490.71	-107340.61	-587777.19	-16688891.04	-47500244.96
Conservas de frutas y vegetales	484507456.17	327513176.76	286254913.77	232628661.23	302930898.87	252293915.48	300665636.88	466867875.61	395791373.12	467174025.26	661525048.58	522763164.30	4700916146.03
Bonif. C. de frutas y vegetales	-4471032.01	-9992769.68	-9766183.21	-7531867.98	-11757869.32	-3828322.00	-1164872.54	-7188342.13	-3679044.04	-7509771.41	-5120594.25	-12252301.79	-84262970.36
Exportación de conservas de frutas	0.00	0.00	0.00	0.00	0.00	9584691.33	34118255.69	12294195.70	0.00	24860913.77	0.00	0.00	80858056.49
Espumosos frutados y otras bebidas	22815848.27	13474701.10	11202390.01	9120830.96	40057362.41	26759118.43	39675361.74	558418175.71	598534688.32	730860038.65	199181004.77	45029436.58	2295128956.95
Bonif. Espumosos frutados y otras bebidas	-547003.60	-767279.68	-2967853.03	-17139.54	-984891.66	-5217.29	-89403.33	-116354046.12	-10334617.85	-115281474.14	-2621835.48	-963430.94	-250934192.66
Otras ventas	0.00	0.00	0.00	0.00	2529174.23	0.00	0.00	2011773.97	0.00	0.00	0.00	858904.00	5399852.20
Otros descuentos y bonificaciones	-31035698.09	-28500839.70	-27303949.90	-24576192.63	-28588051.99	-22317927.12	-11931120.44	-22510260.17	-57685715.26	-24086066.26	-15426363.21	-24695900.42	-318658085.19
Total	1402217408.17	1038917235.33	967780018.37	952757867.72	856159624.05	714361913.78	707111170.79	1505899198.94	1329953399.21	1269194229.31	915233953.07	1182874400.93	12842460419.67

INFORME DE VENTAS EN VOLUMEN MENSUALES POR SEGMENTO:

```
orden_clasificaciones_vv = [
    'Conservas de pescados',
    'Conservas de frutas y vegetales',
    'Exportación de conservas de frutas',
    'Espumosos frutados y otras bebidas',
    'Otras ventas',
    'Total',
]

VTAS_VOLUMEN_MENSUALES_POR_SEGMENTO=DF_VTAS_VOLUMEN_DATOS_LIMPIOS.pivot_table(index = "Clasificacion",
                                    columns = "MES",
                                    values = "Cantidad",
                                    aggfunc = "sum",
                                    fill_value=0,
                                    margins=True,
                                    margins_name='Total').reindex(orden_clasificaciones_vv).astype("int64")

VTAS_VOLUMEN_MENSUALES_POR_SEGMENTO
```

MES	1	2	3	4	5	6	7	8	9	10	11	12	Total
Clasificacion													
Conservas de pescados	25669	20565	21012	20339	14790	13101	9233	13722	8400	4932	2135	9772	163670
Conservas de frutas y vegetales	37089	26351	24284	18372	18176	19469	23318	26586	22795	36702	46809	28588	328539
Exportación de conservas de frutas	0	0	0	0	0	1080	2716	1590	0	2288	0	0	7674
Espumosos frutados y otras bebidas	2351	1044	871	1118	3328	2852	4356	30788	43429	43702	15475	4809	154123
Otras ventas	0	0	0	0	3726	0	0	3000	0	0	0	2126	8852
Total	65109	47960	46167	39829	40020	36502	39623	75686	74624	87624	64419	45295	662858

- RANKING DE VENTAS:

También se pueden elaborar informes de ranking de ventas, utilizando el método groupby() de pandas. Ejemplos visualizando las 10 primeras posiciones:

RANKING DE VENTAS ANUALES POR VENDEDOR, EN UNIDADES MONETARIAS:

```
RANKING_VTAS_VENDEDOR=DF_VTAS_NETAS_DATOS_LIMPIOS.groupby(["Cod._vendedor"])["Total_Reexpresado"].agg("sum").sort_values(ascending=False).reset_index()
```

```
RANKING_VTAS_VENDEDOR.rename(columns={'Cod._vendedor':'Vendedor', 'Total_Reexpresado':'Ventas Netas 2023'},inplace=True)
```

```
RANKING_VTAS_VENDEDOR.head(10)
```

	Vendedor	Ventas Netas 2023
0	103	1867363178.28
1	7	1782825823.08
2	14	1568460118.97
3	5	1048801535.84
4	11	990874330.18
5	105	960643297.37
6	102	880801087.45
7	6	753994592.93
8	104	582433617.90
9	109	479027574.84

PRINCIPALES PRODUCTOS VENDIDOS EN EL AÑO, EXPRESADO EN CAJAS:

```
RANKING_VTAS_PRODUCTOS=DF_VTAS_NETAS_DATOS_LIMPIOS.groupby(["Cod._Articulo"])["Cantidad"].agg("sum").sort_values(ascending=False).astype("int64").reset_index()
```

```
RANKING_VTAS_PRODUCTOS.rename(columns={'Cod._Articulo':'Producto', 'Cantidad':'Cajas vendidas 2023'},inplace=True)
```

```
RANKING_VTAS_PRODUCTOS.head(10)
```

	Producto	Cajas vendidas 2023
0	70323	119514
1	20314	58090
2	12013	48598
3	20315	35822
4	20702	35499
5	20807	34196
6	42001	32826
7	10112	22081
8	10111	21130
9	10173	19849

RANKING DE VENTAS ANUALES POR CLIENTES, EN UNIDADES MONETARIAS:

```
pd.options.display.float_format = '{:.2f}'.format
```

```
RANKING_VTAS_CLIENTES=DF_VTAS_NETAS_DATOS_LIMPIOS.groupby(["Cod._cliente"])["Total_Reexpresado"].agg("sum").sort_values(ascending=False).reset_index()
```

```
RANKING_VTAS_CLIENTES.rename(columns={'Cod._cliente':'Cliente', 'Total_Reexpresado':'Ventas Netas 2023'},inplace=True)

RANKING_VTAS_CLIENTES.head(10)
```

	Cliente	Ventas Netas 2023
0	5822	1082448928.28
1	7453	364427370.15
2	4319	339570719.34
3	6307	277952575.77
4	1730	277171545.39
5	6531	258297451.62
6	5231	154360801.67
7	2193	153046701.51
8	2103	148536954.75
9	4880	145593024.03

RANKING DE VENTAS ANUALES POR PROVINCIA, EN UNIDADES MONETARIAS:

```
pd.options.display.float_format = '{:.2f}'.format

RANKING_VTAS_PROVINCIAS=DF_VTAS_NETAS_DATOS_LIMPIOS.groupby(["Nombre_provincia"])["Total_Reexpresado"].agg("sum").sort_values(ascending=False).reset_index()

RANKING_VTAS_PROVINCIAS.rename(columns={'Nombre_provincia':'Provincia', 'Total_Reexpresado':'Ventas Netas 2023'},inplace=True)

RANKING_VTAS_PROVINCIAS.head(10)
```

	Provincia	Ventas Netas 2023
0	Buenos Aires	3314199668.74
1	Santa Fe	2217168269.10
2	Capital Federal	2128867138.34
3	Córdoba	1629674982.79
4	Entre Ríos	370013781.35
5	Mendoza	358911504.84
6	Misiones	328597741.01
7	Chaco	311309704.42
8	Tucumán	274953850.93
9	Salta	240276432.59

EXPORTACIÓN DE LOS INFORMES A EXCEL

```
Directorio_Destino = "C:/Users/Usuario/Downloads/"


VTAS_NETAS_MENSUALES_POR_SEGMENTO.to_excel(Directorio_Destino+"VTAS_NETAS_MENSUALES_POR_SEGMENTO.xlsx", sheet_name="Vtas Netas")

VTAS_VOLUMEN_MENSUALES_POR_SEGMENTO.to_excel(Directorio_Destino+"VTAS_VOLUMEN_MENSUALES_POR_SEGMENTO.xlsx", sheet_name="Vtas Volumen")

RANKING_VTAS_VENDEDOR.to_excel(Directorio_Destino+"RANKING_VTAS_VENDEDOR.xlsx", sheet_name="Ranking Vendedores", index=False)
```

```
RANKING_VTAS_PRODUCTOS.to_excel(Directorio_Destino+"RANKING_VTAS_PRODUCTOS.xlsx", sheet_name="Ranking Productos", index=False)

RANKING_VTAS_CLIENTES.to_excel(Directorio_Destino+"RANKING_VTAS_CLIENTES.xlsx", sheet_name="Ranking Clientes", index=False)

RANKING_VTAS_PROVINCIAS.to_excel(Directorio_Destino+"RANKING_VTAS_PROVINCIAS.xlsx", sheet_name="Ranking Provincias", index=False)
```

ANEXO

IDENTIFICAR ELEMENTOS FALTANTES EN UN DATAFRAME E INCORPORARLOS

Al utilizar la función **merge()** de **pandas** con el parámetro how=”left”, el nuevo DataFrame de Ventas con clasificaciones va a incluir todas las ventas del reporte de ventas, independientemente de si estaban clasificados los artículos o no. Esto permite detectar cuando se incorporen nuevos productos y haya que clasificarlos.

Así, con un script que sume la cantidad de valores nulos que hay en la columna “Clasificacion”, se puede saber si hay productos nuevos por clasificar:

```
df_VENTAS_2023_CLASIFICADAS["Clasificacion"].isnull().sum()
```

Para identificar cuales son estos nuevos productos, se puede utilizar el siguiente código:

```
df_VENTAS_2023_CLASIFICADAS[df_VENTAS_2023_CLASIFICADAS["Clasificacion"].isnull()]["Cod._Articulo"]
```

Y así, una vez identificados los artículos a incorporar, se pueden agregar al DataFrame de clasificaciones con la función **concat()** de **pandas**:

```
nuevos_productos = {"Cod._Articulo": ["010999", "020999"],
                    "Clasificacion": ["Conservas de pescados", "Conservas de frutas y vegetales"]}
df_nuevos_productos = pd.DataFrame(nuevos_productos)
df_Clasificacion_articulos = pd.concat([df_Clasificacion_articulos, df_nuevos_productos], ignore_index=True)
```

Luego de esto, se vuelve a correr el script que une las ventas con las clasificaciones y, de esta forma, se vuelve a generar el nuevo DataFrame, quedando todos los artículos vendidos clasificados:

```
df_VENTAS_2023_CLASIFICADAS=pd.merge(df_REPORTE_VENTAS_2023,
                    df_Clasificacion_articulos,
                    on="Cod._Articulo",
                    how="left")
```

Además, se puede grabar el DataFrame de clasificaciones que incluye los nuevos productos en el archivo de Excel original de Clasificaciones para que el mismo quede actualizado:

```
df_Clasificacion_articulos.to_excel(directorio_origen+"Clasificacion articulos prueba.xlsx", index=False)
```