RTP - guia

March 3, 2023

[ ]:

[1]:
```
# Caraganos todas las librerias necesarias

import os
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:

1 1. Manejo de Python

1. Rutas archivos y display
2. Lectura de archivos: Excel, CSV y TXT
3. Tipo de datos en pyuthon y Numpy
4. Condionales y Capture
5. Bucles
6. Funciones
7. Group By, Merge y Concat

1.0.1 1.1 Rutas archivos y display

[2]:
```
# Directorio en el que estoy

print(os.getcwd())
```

```
/Users/adrian_gr/Desktop/4.cnmv/04.practica
```

[4]:
```
# Cambio a otro directorio

os.chdir("/Users/adrian_gr/Desktop/4.cnmv/04.practica/01. python_avanzado") #␣
 ↪cambiamos el directorio
print(os.getcwd()) # ruta actual
```

```
os.chdir("/Users/adrian_gr/Desktop/4.cnmv/04.practica") # cambiamos el␣
 ↪directorio
print(os.getcwd()) # ruta actual
```

```
/Users/adrian_gr/Desktop/4.cnmv/04.practica/01. python_avanzado
/Users/adrian_gr/Desktop/4.cnmv/04.practica
```

[40]:
```
# Todos los archivos de una carpeta y que los liste especificando condicion de␣
 ↪nombre y tipo archivo
# Y que me lo guarde en una lista

ruta = "/Users/adrian_gr/Desktop/4.cnmv/04.practica/01. python_avanzado"
archivos = os.listdir(ruta)

l_archivos = []
for archivo in archivos:
    ruta_completa = ruta + "/" + archivo # como en stata, le digo toda la ruta
    if os.path.isfile(ruta_completa) and archivo.startswith("") and "xlsx" in␣
 ↪archivo and not "python" in archivo: # solo excel
        print(archivo)
        l_archivos = l_archivos + [archivo]

print(l_archivos)
# Ver lo sensillo que es esto. Condiones de nombre es archivo.startswith/
 ↪archivo.endswith o in
# Ver que facil decirle que no contenga la palabnra python
```

```
prueba_1.xlsx
prueBa_4.xlsx
prueba_3.xlsx
prueba_2.xlsx
['prueba_1.xlsx', 'prueBa_4.xlsx', 'prueba_3.xlsx', 'prueba_2.xlsx']
```

[26]:
```
# Creamos una carpeta y la borramos

os.chdir("/Users/adrian_gr/Desktop/4.cnmv/04.practica") # decimos en que ruta

os.mkdir("000.xlsx") # creamos una nueva carpeta y vemos si esta
print(os.listdir(os.getcwd())) # esta
os.rmdir("000.xlsx") # vemos que está y le eliminamos
print(os.listdir(os.getcwd())) # ya no esta
```

```
['RTP - Oposicon Tecnico Analista de Datos CNMV.ipynb', '.DS_Store', '05.
modelos', '.ipynb_checkpoints', '04. tratamiento', '02. visualizacion', '01.
python_avanzado', '06. NLP y Web Scraping', '03. SQL']
```

[33]:
```
# Cambiamos el nombre de un archivo y luego lo volvemos a dejar igual

ruta = "/Users/adrian_gr/Desktop/4.cnmv/04.practica/01. python_avanzado"
archivos = os.listdir(ruta)

for archivo in archivos:
    ruta_completa = ruta + "/" + archivo
    if "prueva" in archivo:
        nuevo_nombre = archivo.replace("prueva", "prueBa")
        nueva_ruta = ruta + "/" + nuevo_nombre
        os.rename(ruta_completa, nueva_ruta)
print(os.listdir(ruta))

for archivo in archivos:
    ruta_completa = ruta + "/" + archivo
    if "prueva" in archivo:
        nuevo_nombre = archivo.replace("prueBa", "prueva") # con replace cambia el nombre
        nueva_ruta = ruta + "/" + nuevo_nombre # define la nueva ruta
        os.rename(ruta_completa, nueva_ruta) #  Y os.rename cambia el nopmbre de facto
print(os.listdir(ruta))
```

```
['prueba_1.xlsx', 'archivo_csv.csv', '.DS_Store', 'prueba_python.xlsx',
'texto.txt', 'prueBa_4.xlsx', '.ipynb_checkpoints', 'prueba_3.xlsx',
'01.python_avanzado.ipynb', 'prueba_2.xlsx']
['prueba_1.xlsx', 'archivo_csv.csv', '.DS_Store', 'prueba_python.xlsx',
'texto.txt', 'prueBa_4.xlsx', '.ipynb_checkpoints', 'prueba_3.xlsx',
'01.python_avanzado.ipynb', 'prueba_2.xlsx']
```

[41]:
```
# Manejamos un poco el display dentro de un bucle de archivos

ruta = "/Users/adrian_gr/Desktop/4.cnmv/04.practica/01. python_avanzado"
archivos = os.listdir(ruta)

contador = 0
for archivo in archivos:
    ruta_completa = ruta + "/" + archivo # como en stata, le digo toda la ruta
    if os.path.isfile(ruta_completa) and archivo.startswith("") and "xlsx" in archivo and not "python" in archivo:
        contador = contador + 1
        print(f'El archivo numero {contador} es el mágnifico {archivo}')
```

```
El archivo numero 1 es el mágnifico prueba_1.xlsx
El archivo numero 2 es el mágnifico prueBa_4.xlsx
El archivo numero 3 es el mágnifico prueba_3.xlsx
El archivo numero 4 es el mágnifico prueba_2.xlsx
```

[ ]:

1.0.2 1.2 Lectura de archivos: Excel, CSV y TXT

[54]:
```
# Definimos el dataframe

datos = {
    'Nombre': ['Juan', 'Maria', 'Carlos', 'Laura'],
    'Apellido': ['Perez', 'Garcia', 'Sanchez', 'Fernández'],
    'Edad': [25, 30, 40, 35]
}


df = pd.DataFrame(datos)
df
```

[54]:

	Nombre	Apellido	Edad
0	Juan	Perez	25
1	Maria	Garcia	30
2	Carlos	Sanchez	40
3	Laura	Fernández	35

[64]:
```
tipo = ["csv", "xlsx", "txt"]

for j in tipo:
    salida = j
    if j == "xlsx":
        salida = "excel"
    elif j == "txt":
        salida = "string"
    print(salida)

    ruta_salida = f"/Users/adrian_gr/Desktop/4.cnmv/04.practica/df.{j}"
    export = f"df.to_{salida}(ruta_salida, index=False)"
    eval(export) # eval interpreta texto como una sentencia python
```

```
csv
excel
string
```

```
<string>:1: UserWarning: Pandas requires version '1.4.3' or newer of
'xlsxwriter' (version '1.2.9' currently installed).
```

[106]:
```
# Abrimos un los difernetes archivos con pandas y mediante un bucle

ruta = "/Users/adrian_gr/Desktop/4.cnmv/04.practica"
```

```
archivos = os.listdir(ruta)

for j in archivos:
    ruta_ar = ruta + "/" + j
    if os.path.isfile(ruta_ar) & j.startswith("df."):
        if "xlsx" in j:
            df_xlsx = pd.read_excel(ruta_ar)
        elif "csv" in j:
            df_csv = pd.read_csv(ruta_ar, sep=",")
        elif "txt" in j:
            df_txt = pd.read_table(ruta_ar, sep = "\t") # separado por tabulador

print(len(df_xlsx))
print(len(df_csv))
print(len(df_txt))
```

```
4
4
4
```

[107]:
```
# Forma de crear nosotros mismos un dataframe

nombres = ["Ana", "Juan", "María", "Pedro", "Lucía"]
edades = range(20, 41, 5)
ciudades = ["Madrid", "Barcelona", "Sevilla", "Valencia", "Bilbao"]

df = pd.DataFrame({
    "Nombre": nombres,
    "Edad": edades,
    "Ciudad": ciudades
})

df
```

[107]:

	Nombre	Edad	Ciudad
0	Ana	20	Madrid
1	Juan	25	Barcelona
2	María	30	Sevilla
3	Pedro	35	Valencia
4	Lucía	40	Bilbao

[ ]:
```

```

1.0.3 3. Tipo de datos en pyuthon y Numpy

1. Números: En Python, existen tres tipos de números: enteros (int), flotantes (float) y complejos (complex). Los enteros son números enteros sin decimales. Se pueden crear asignando un valor numérico a una variable: x = 5. Los flotantes son números decimales. Se pueden crear asignando un valor decimal a una variable: x = 3.14. Los complejos son números que tienen una parte real y una parte imaginaria. Se pueden crear especificando la parte real y la parte imaginaria: x = 3 + 4j.
2. En Python, una cadena de texto es una secuencia de caracteres encerrados entre comillas simples o dobles. Se pueden crear asignando un valor de cadena a una variable: x = "Hola Mundo".
3. En Python, una lista es una colección de elementos ordenados y modificables. Los elementos pueden ser de cualquier tipo de dato, y se separan por comas y se encierran entre corchetes. Se pueden crear una lista vacía y agregar elementos posteriormente: x = [] y x.append(5). También se pueden crear una lista con elementos predefinidos: x = [1, 2, 3].
4. En Python, una tupla es una colección de elementos ordenados e inmodificables. Los elementos pueden ser de cualquier tipo de dato, y se separan por comas y se encierran entre paréntesis. Se pueden crear una tupla vacía y agregar elementos posteriormente: x = () y x = x + (5,). También se pueden crear una tupla con elementos predefinidos: x = (1, 2, 3).
5. En Python, un conjunto es una colección no ordenada y sin elementos duplicados. Se pueden crear un conjunto vacío y agregar elementos posteriormente: x = set() y x.add(5). También se pueden crear un conjunto con elementos predefinidos: x = {1, 2, 3}.
6. En Python, un diccionario es una colección de pares clave-valor, donde las claves son únicas e inmodificables, y los valores pueden ser de cualquier tipo de dato. Se pueden crear un diccionario vacío y agregar pares clave-valor posteriormente: x = {} y x["clave"] = 5. También se pueden crear un diccionario con pares clave-valor predefinidos: x = {"clave1": 1, "clave2": 2}.

Es importante mencionar que Python es un lenguaje de programación de tipado dinámico, lo que significa que no es necesario declarar el tipo de dato de una variable al momento de su creación, sino que este se infiere automáticamente a partir del valor asignado a la variable.

[120]:
```
# Creamos una lista

a = [1, "a", 7, "b"]
b = [1 , 7, "hola"]
c = a + b
print(c)
print(c[1:5])
```

```
[1, 'a', 7, 'b', 1, 7, 'hola']
['a', 7, 'b', 1]
```

[132]:
```
# Hacemos cosas con munpy y tuplas

lista = list(np.random.randint(20, 30, size=10))
```

```
print(lista)

lista = list(np.zeros(10))
print(lista)

lista_missing = list(np.full(10, np.nan))
print(lista_missing)

lista_unos = list(np.ones(10))
print(lista_unos)
```

```
[21, 25, 25, 25, 28, 22, 24, 28, 23, 22]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```

[133]:
```
# Y ahora generamos un dataframe con ellos

df = pd.DataFrame({
    'lista_0': lista,
    'lista_missing': lista_missing,
    'lista_1': lista_unos
})
df
```

[133]:

	lista_0	lista_missing	lista_1
0	0.0	NaN	1.0
1	0.0	NaN	1.0
2	0.0	NaN	1.0
3	0.0	NaN	1.0
4	0.0	NaN	1.0
5	0.0	NaN	1.0
6	0.0	NaN	1.0
7	0.0	NaN	1.0
8	0.0	NaN	1.0
9	0.0	NaN	1.0

[124]:
```
# Creamos una tupla

a = (1, "hola")
b = ("adios")
```

[125]:
```
# Creamos un set

# Crear un conjunto vacío
conjunto_vacio = set()
```

```
# Crear un conjunto con elementos
mi_conjunto = {1, 2, 3, "Hola", True}

# Agregar elementos a un conjunto
mi_conjunto.add("nuevo elemento")
mi_conjunto.update([4, 5])
mi_conjunto
```

[125]:
```
{1, 2, 3, 4, 5, 'Hola', 'nuevo elemento'}
```

[126]:
```
# Creamos un diccionario

# Crear un diccionario vacío
diccionario_vacio = {}

# Crear un diccionario con elementos
mi_diccionario = {"nombre": "Juan", "edad": 30, "ciudad": "Madrid"}

# Agregar un elemento a un diccionario
mi_diccionario["profesion"] = "Ingeniero"
mi_diccionario
```

[126]:
```
{'nombre': 'Juan', 'edad': 30, 'ciudad': 'Madrid', 'profesion': 'Ingeniero'}
```

[ ]:
```

```

1.0.4 1.4 Condicional y Capture

[ ]:
```

```

1.0.5 1.4 Bucles

[ ]:
```

```

1.0.6 1.4 Funciones

[ ]:
```

```

1.0.7 1.4 Group by, Merge y Concat

[209]:
```
vendedores = ['Juan', 'Pedro', 'Ana', 'María', 'Pablo', 'Luisa', 'Sara',␣
↪'David', 'Elena', 'Mario'] * 10
dias = [1]*100
meses = np.random.randint(1, 13, size=100)
anios = np.random.randint(2010, 2021, size=100)
```

```
productos = ['Producto1', 'Producto2', 'Producto3', 'Producto4', 'Producto5',␣
 ↪'Producto6', 'Producto7', 'Producto8', 'Producto9', 'Producto10']* 10
importes = np.random.randint(100, 10000, size=100)
empresas = ['Empresa1', 'Empresa2', 'Empresa3', 'Empresa4']
nacionalidades = ['Español', 'Italiano', 'Alemán', 'Frances', 'Ruso', 'Chino',␣
 ↪'Japonés', 'Argentino', 'Mexicano', 'Colombiano']

# Crear DataFrame
data = {'Vendedor': vendedores,
        'Dia': dias,
        'Mes': meses,
        'Año': anios,
        'Producto': productos,
        'Importe': importes,
        'Empresa': np.random.choice(empresas, size=100),
        'Nacionalidad': np.random.choice(nacionalidades, size=100)}

df = pd.DataFrame(data)

# Mostrar las primeras 5 filas del DataFrame
df.head()
```

[209]:

	Vendedor	Dia	Mes	Año	Producto	Importe	Empresa	Nacionalidad
0	Juan	1	8	2020	Producto1	6106	Empresa2	Colombiano
1	Pedro	1	8	2011	Producto2	5778	Empresa1	Mexicano
2	Ana	1	9	2013	Producto3	7870	Empresa3	Japonés
3	María	1	9	2018	Producto4	1968	Empresa1	Mexicano
4	Pablo	1	3	2016	Producto5	8436	Empresa2	Ruso

[486]:
```
# duplicados

df = df.drop_duplicates()
```

[210]:
```
# Basics
print(df.describe())
print(df.info())
print(df.shape)
```

```
         Dia         Mes          Año      Importe
count  100.0  100.000000   100.000000   100.000000
mean     1.0    5.800000  2015.170000  5329.010000
std      0.0    3.437758     2.895678  2927.221058
min      1.0    1.000000  2010.000000   192.000000
25%      1.0    3.000000  2013.000000  2667.000000
50%      1.0    5.000000  2015.000000  5726.500000
75%      1.0    9.000000  2018.000000  8012.500000
max      1.0   12.000000  2020.000000  9871.000000
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Vendedor      100 non-null    object
 1   Dia           100 non-null    int64
 2   Mes           100 non-null    int64
 3   Año           100 non-null    int64
 4   Producto      100 non-null    object
 5   Importe       100 non-null    int64
 6   Empresa       100 non-null    object
 7   Nacionalidad  100 non-null    object
dtypes: int64(4), object(4)
memory usage: 6.4+ KB
None
(100, 8)
```

[211]:
```
# Cuanto tipos tiene cada variable
# asi se filtran las columnas
columnas = df.columns[df.columns.str.contains('Mes') == False]


for i in df.columns:
    if i != "Importe":
        print(df[f'{i}'].unique())
```

```
['Juan' 'Pedro' 'Ana' 'María' 'Pablo' 'Luisa' 'Sara' 'David' 'Elena'
 'Mario']
[1]
[ 8  9  3 12 11  2  1  5  4  6 10  7]
[2020 2011 2013 2018 2016 2014 2012 2015 2019 2017 2010]
['Producto1' 'Producto2' 'Producto3' 'Producto4' 'Producto5' 'Producto6'
 'Producto7' 'Producto8' 'Producto9' 'Producto10']
['Empresa2' 'Empresa1' 'Empresa3' 'Empresa4']
['Colombiano' 'Mexicano' 'Japonés' 'Ruso' 'Italiano' 'Frances' 'Argentino'
 'Chino' 'Español' 'Alemán']
```

[212]:
```
# Poner todos los nombres de las variables en minusculas

for i in df.columns:
    n = i.lower()
    df.rename(columns={i: n}, inplace=True)

df.columns
```

[212]:
```
Index(['vendedor', 'dia', 'mes', 'año', 'producto', 'importe', 'empresa',
       'nacionalidad'],
      dtype='object')
```

[213]:
```
# Tratamos las variables string

col_o = df.select_dtypes(include='object').columns

for j in col_o:
    df[f'{j}'] = df[f'{j}'].str.lower()
    df[f'{j}'] = df[f'{j}'].str.replace('  ', ' ')
    df[f'{j}'] = df[f'{j}'].str.strip()
    df[f'{j}'] = df[f'{j}'].apply(lambda x: unidecode.unidecode(x))

df.head()
```

[213]:

	vendedor	dia	mes	año	producto	importe	empresa	nacionalidad
0	juan	1	8	2020	producto1	6106	empresa2	colombiano
1	pedro	1	8	2011	producto2	5778	empresa1	mexicano
2	ana	1	9	2013	producto3	7870	empresa3	japones
3	maria	1	9	2018	producto4	1968	empresa1	mexicano
4	pablo	1	3	2016	producto5	8436	empresa2	ruso

[214]:
```
# Vamos a hacer que producto y empresa solo tengan los numeros

df['n_emp'] = df['empresa'].apply(lambda x: x[-1:]).astype(int) # importnte␣
 ↪pasar a numeric "str" al reves
df['n_pro'] = df['producto'].apply(lambda x: x[-1:]).astype(int)
df.drop(['empresa', 'producto'], axis = 1, inplace = True)
df
```

[214]:

	vendedor	dia	mes	año	importe	nacionalidad	n_emp	n_pro
0	juan	1	8	2020	6106	colombiano	2	1
1	pedro	1	8	2011	5778	mexicano	1	2
2	ana	1	9	2013	7870	japones	3	3
3	maria	1	9	2018	1968	mexicano	1	4
4	pablo	1	3	2016	8436	ruso	2	5
..	…	…	…	…	…	…	…	…
95	luisa	1	10	2014	6113	colombiano	3	6
96	sara	1	10	2015	2309	colombiano	4	7
97	david	1	1	2012	2249	argentino	2	8
98	elena	1	3	2012	3101	mexicano	4	9
99	mario	1	9	2013	5442	argentino	2	0

[100 rows x 8 columns]

[215]:
```
# Generamos la variable fecha. La ponemos y quitamos de indice

df['fecha'] = pd.to_datetime(df['año'].astype(str) + '-' + df['mes'].
 ↪astype(str) + '-' + df['dia'].astype(str))
df.info()

df = df.set_index('fecha') # lo ponemos como indice
df.reset_index(inplace=True) # lo quitamos
df.rename(columns={'index': 'fecha'}, inplace=True) # lo quitamos
df
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   vendedor      100 non-null    object
 1   dia           100 non-null    int64
 2   mes           100 non-null    int64
 3   año           100 non-null    int64
 4   importe       100 non-null    int64
 5   nacionalidad  100 non-null    object
 6   n_emp         100 non-null    int64
 7   n_pro         100 non-null    int64
 8   fecha         100 non-null    datetime64[ns]
dtypes: datetime64[ns](1), int64(6), object(2)
memory usage: 7.2+ KB
```

[215]:

	fecha	vendedor	dia	mes	año	importe	nacionalidad	n_emp	n_pro
0	2020-08-01	juan	1	8	2020	6106	colombiano	2	1
1	2011-08-01	pedro	1	8	2011	5778	mexicano	1	2
2	2013-09-01	ana	1	9	2013	7870	japones	3	3
3	2018-09-01	maria	1	9	2018	1968	mexicano	1	4
4	2016-03-01	pablo	1	3	2016	8436	ruso	2	5
..	…	…	…	…	…	…	…	…	…
95	2014-10-01	luisa	1	10	2014	6113	colombiano	3	6
96	2015-10-01	sara	1	10	2015	2309	colombiano	4	7
97	2012-01-01	david	1	1	2012	2249	argentino	2	8
98	2012-03-01	elena	1	3	2012	3101	mexicano	4	9
99	2013-09-01	mario	1	9	2013	5442	argentino	2	0

[100 rows x 9 columns]

[216]:
```
# Que me saque las obs entre dos periodos

print(df[df['fecha'].between('2015-01-01', '2016-01-31')].count()[1])
df[df['fecha'].between('2015-01-01', '2016-01-31')].head()
```

```
# hay 20 obs
```

```
12
```

[216]:

	fecha	vendedor	dia	mes	año	importe	nacionalidad	n_emp	n_pro
14	2015-05-01	pablo	1	5	2015	8563	ruso	1	5
30	2015-02-01	juan	1	2	2015	7040	japones	4	1
31	2015-10-01	pedro	1	10	2015	5404	argentino	1	2
35	2015-05-01	luisa	1	5	2015	3564	japones	4	6
38	2015-02-01	elena	1	2	2015	1425	frances	4	9

[220]:
```
# Que me cuente solo la empresa 1 y el producto 1.

df[(df['n_emp'] == 1) & (df['n_pro'] == 1)].count()[1]
```

[220]: 1

[228]:
```
# Cuantas veces se repite cada valor de una variable y uno en especifico

print(df['vendedor'].value_counts())

print(df[df['vendedor'] == "juan"].count()[1])
```

```
juan      10
pedro     10
ana       10
maria     10
pablo     10
luisa     10
sara      10
david     10
elena     10
mario     10
Name: vendedor, dtype: int64
10
```

[229]:
```
# Generar una variable con el numero de veces que se repite una variable

df['n_obs_vend'] = df.groupby('vendedor')['vendedor'].transform('count')
df
```

[229]:

	fecha	vendedor	dia	mes	año	importe	nacionalidad	n_emp	n_pro	\
0	2020-08-01	juan	1	8	2020	6106	colombiano	2	1	
1	2011-08-01	pedro	1	8	2011	5778	mexicano	1	2	
2	2013-09-01	ana	1	9	2013	7870	japones	3	3	
3	2018-09-01	maria	1	9	2018	1968	mexicano	1	4	
4	2016-03-01	pablo	1	3	2016	8436	ruso	2	5	

```
..         …      …  …  …   …      …          …    …    …
95 2014-10-01    luisa    1   10  2014     6113   colombiano      3      6
96 2015-10-01     sara    1   10  2015     2309   colombiano      4      7
97 2012-01-01    david    1    1  2012     2249    argentino      2      8
98 2012-03-01    elena    1    3  2012     3101     mexicano      4      9
99 2013-09-01    mario    1    9  2013     5442    argentino      2      0

    n_obs_vend
0           10
1           10
2           10
3           10
4           10
..         …
95          10
96          10
97          10
98          10
99          10

[100 rows x 10 columns]
```

[255]:
```
# Generar una variable que sea la suma de otra por una categoria de otra␣
↪variable

df['imp_vend'] = df.groupby(['vendedor', 'año'])['importe'].transform('sum')
df
```

[255]:
```
        fecha vendedor  dia  mes   año  importe nacionalidad  n_emp  n_pro  \
62 2016-05-01      ana    1    5  2016     6301      espanol      1      3
72 2020-12-01      ana    1   12  2020      752     mexicano      3      3
32 2019-01-01      ana    1    1  2019      192       aleman      2      3
42 2016-04-01      ana    1    4  2016     3104   colombiano      1      3
82 2012-02-01      ana    1    2  2012     7545        chino      3      3
..         …      …  …  …   …      …          …    …    …
36 2017-07-01     sara    1    7  2017     6253    argentino      2      7
76 2014-07-01     sara    1    7  2014     1511       aleman      4      7
46 2018-05-01     sara    1    5  2018     5675    argentino      2      7
86 2012-01-01     sara    1    1  2012     8694      japones      1      7
16 2011-02-01     sara    1    2  2011     1397         ruso      3      7

    n_obs_vend  imp_vend
62          10      9405
72          10      6969
32          10      8589
42          10      9405
82          10      8115
```

```
..          …       …
36          10      6253
76          10      1511
46          10     13522
86          10      8694
16          10      1397

[100 rows x 11 columns]
```

[248]:
```
# Poder tener señalado solo la primera observacion de cada vendedor y producto

df.sort_values('vendedor', inplace=True) # ordenamos por una variable
df['aux'] = df.groupby(['vendedor', 'año'])['vendedor'].transform(lambda x: x.
 ↪index == x.index.min())
df
df.drop('aux', axis = 1, inplace = True)
df
```

[248]:
```
        fecha vendedor  dia  mes   año  importe nacionalidad  n_emp  n_pro  \
62 2016-05-01      ana    1    5  2016     6301      espanol      1      3
72 2020-12-01      ana    1   12  2020      752     mexicano      3      3
32 2019-01-01      ana    1    1  2019      192       aleman      2      3
42 2016-04-01      ana    1    4  2016     3104   colombiano      1      3
82 2012-02-01      ana    1    2  2012     7545        chino      3      3
..        …       …   …   …   …   …       …          …    …     …
36 2017-07-01     sara    1    7  2017     6253    argentino      2      7
76 2014-07-01     sara    1    7  2014     1511       aleman      4      7
46 2018-05-01     sara    1    5  2018     5675    argentino      2      7
86 2012-01-01     sara    1    1  2012     8694      japones      1      7
16 2011-02-01     sara    1    2  2011     1397         ruso      3      7

    n_obs_vend  imp_vend
62          10     42748
72          10     42748
32          10     42748
42          10     42748
82          10     42748
..         …       …
36          10     50802
76          10     50802
46          10     50802
86          10     50802
16          10     50802

[100 rows x 11 columns]
```

[241]:
```
df.drop('aux', axis = 1, inplace = True)
```

[244]:
```
df
```

[244]:
```
        fecha vendedor  dia  mes   año  importe nacionalidad  n_emp  n_pro  \
0  2020-08-01     juan    1    8  2020     6106   colombiano      2      1
1  2011-08-01    pedro    1    8  2011     5778     mexicano      1      2
2  2013-09-01      ana    1    9  2013     7870      japones      3      3
3  2018-09-01    maria    1    9  2018     1968     mexicano      1      4
4  2016-03-01    pablo    1    3  2016     8436         ruso      2      5
..        …       …  …  …  …   …       …          …    …     …
95 2014-10-01    luisa    1   10  2014     6113   colombiano      3      6
96 2015-10-01     sara    1   10  2015     2309   colombiano      4      7
97 2012-01-01    david    1    1  2012     2249    argentino      2      8
98 2012-03-01    elena    1    3  2012     3101     mexicano      4      9
99 2013-09-01    mario    1    9  2013     5442    argentino      2      0

    n_obs_vend  imp_vend
0           10     59540
1           10     59796
2           10     42748
3           10     50654
4           10     65074
..         …       …
95          10     52020
96          10     50802
97          10     49096
98          10     31352
99          10     71819

[100 rows x 11 columns]
```

[254]:
```
# Hacemos GB de vendedor y empresa

df.groupby(['vendedor', 'n_emp'])['importe'].sum().reset_index().head()
```

[254]:
```
  vendedor  n_emp  importe
0      ana      1     9975
1      ana      2     6409
2      ana      3    26364
3    david      1    18750
4    david      2     9935
```

[257]:
```
# Hacemos GB lo que se ha vendido de cada producto

df.groupby('n_pro')['importe'].sum().reset_index()
```

[257]:
```
   n_pro  importe
0      0    71819
```

```
1      1    59540
2      2    59796
3      3    42748
4      4    50654
5      5    65074
6      6    52020
7      7    50802
8      8    49096
9      9    31352
```

[312]:
```
# Vamos a generarnos de forma artificial otro df para hacer un append == concat

df.sort_index(inplace = True)
df2 = df[['n_pro', 'importe', 'vendedor']].iloc[0:10]
df2 = pd.concat([df2, df], ignore_index=True) # los vamos apendeando
df2

# esto es un append y no tiene más
```

[312]:
```
     n_pro  importe vendedor      fecha  dia   mes     año nacionalidad  \
0        1     6106     juan        NaT  NaN   NaN     NaN         NaN
1        2     5778    pedro        NaT  NaN   NaN     NaN         NaN
2        3     7870      ana        NaT  NaN   NaN     NaN         NaN
3        4     1968    maria        NaT  NaN   NaN     NaN         NaN
4        5     8436    pablo        NaT  NaN   NaN     NaN         NaN
..     …       …       …          …   …    …     …          …
105      6     6113    luisa 2014-10-01  1.0  10.0  2014.0   colombiano
106      7     2309     sara 2015-10-01  1.0  10.0  2015.0   colombiano
107      8     2249    david 2012-01-01  1.0   1.0  2012.0    argentino
108      9     3101    elena 2012-03-01  1.0   3.0  2012.0     mexicano
109      0     5442    mario 2013-09-01  1.0   9.0  2013.0    argentino

     n_emp  n_obs_vend  imp_vend
0      NaN         NaN       NaN
1      NaN         NaN       NaN
2      NaN         NaN       NaN
3      NaN         NaN       NaN
4      NaN         NaN       NaN
..     …          …       …
105    3.0        10.0    9012.0
106    4.0        10.0    2309.0
107    2.0        10.0   18335.0
108    4.0        10.0    3101.0
109    2.0        10.0   13525.0

[110 rows x 11 columns]
```

[316]:
```
# Vamos a generar un df para hacer un merge

df_m = df.groupby('año')['importe'].mean().reset_index()
df_m = df_m.merge(df, on='año', how='left')
df_m.rename(columns={'importe_x': 'media_anual'}, inplace=True)
df_m
```

[316]:

	año	media_anual	fecha	vendedor	dia	mes	importe_y	nacionalidad	\
0	2010	5759.000000	2010-03-01	elena	1	3	8642	chino	
1	2010	5759.000000	2010-06-01	mario	1	6	2876	aleman	
2	2011	3642.555556	2011-08-01	pedro	1	8	5778	mexicano	
3	2011	3642.555556	2011-08-01	luisa	1	8	2166	italiano	
4	2011	3642.555556	2011-02-01	sara	1	2	1397	ruso	
..	…	…	…	…	…	…	…	…	
95	2020	4055.571429	2020-01-01	ana	1	1	6217	colombiano	
96	2020	4055.571429	2020-04-01	david	1	4	498	italiano	
97	2020	4055.571429	2020-07-01	sara	1	7	9554	japones	
98	2020	4055.571429	2020-12-01	ana	1	12	752	mexicano	
99	2020	4055.571429	2020-09-01	pablo	1	9	1782	colombiano	

	n_emp	n_pro	n_obs_vend	imp_vend
0	1	9	10	8642
1	1	0	10	2876
2	1	2	10	5778
3	3	6	10	10476
4	3	7	10	1397
..	…	…	…	…
95	2	3	10	6969
96	2	8	10	498
97	4	7	10	9554
98	3	3	10	6969
99	4	5	10	1782

[100 rows x 12 columns]

[325]:
```
# comprobamos que sale lo mismo haciendo con el complemnete del bys

df_m['aux'] = df_m.groupby('año')['importe_y'].transform('mean')
df_m['aux_1'] = np.where(df_m['aux'] == df_m['media_anual'], 1, 0)
df_m['aux_1'].unique() # siempre coincide
```

[325]:
```
array([1])
```

[ ]:
```

```

[275]:
```
# Vamos a hacer un reshape wide de vendedor importe y año
# nos aseguramos que solo haya un valor por año
```

```
df_w = df[['vendedor', 'año', 'importe']]
df_w = df.groupby(['vendedor', 'año'])['importe'].sum().reset_index()
df_w = df_w.pivot(index='año', columns='vendedor', values='importe')
df_w
```

[275]:
```
vendedor     ana    david   elena     juan    luisa    maria    mario  \
año
2010         NaN      NaN  8642.0      NaN      NaN      NaN   2876.0
2011      1800.0      NaN  3983.0   1439.0  10476.0    825.0   7085.0
2012      8115.0  18335.0  3101.0      NaN   9451.0      NaN   9017.0
2013      7870.0      NaN  1165.0  21490.0   6971.0   9448.0  13525.0
2014         NaN  14196.0     NaN   8487.0   9012.0   7063.0      NaN
2015         NaN   1437.0  4740.0   7040.0   3564.0      NaN   8875.0
2016      9405.0      NaN     NaN    602.0   1083.0      NaN  22088.0
2017         NaN      NaN  2592.0      NaN      NaN   8493.0      NaN
2018         NaN   7442.0  3649.0   8836.0   4274.0  16958.0      NaN
2019      8589.0   7188.0     NaN   5540.0   7189.0   7867.0   8353.0
2020      6969.0    498.0  3480.0   6106.0      NaN      NaN      NaN

vendedor    pablo    pedro     sara
año
2010          NaN      NaN      NaN
2011          NaN   5778.0   1397.0
2012          NaN  11293.0   8694.0
2013       9501.0   9745.0      NaN
2014       8068.0   2091.0   1511.0
2015      10589.0  20510.0   2309.0
2016       8436.0   6070.0      NaN
2017       9075.0      NaN   6253.0
2018       9629.0   4309.0  13522.0
2019       7994.0      NaN   7562.0
2020       1782.0      NaN   9554.0
```

[ ]:
```

```

2 2. SQL

[ ]:
```

```

3 3. Visualización

[326]:
```
# Trabajaremos con este df

df
```

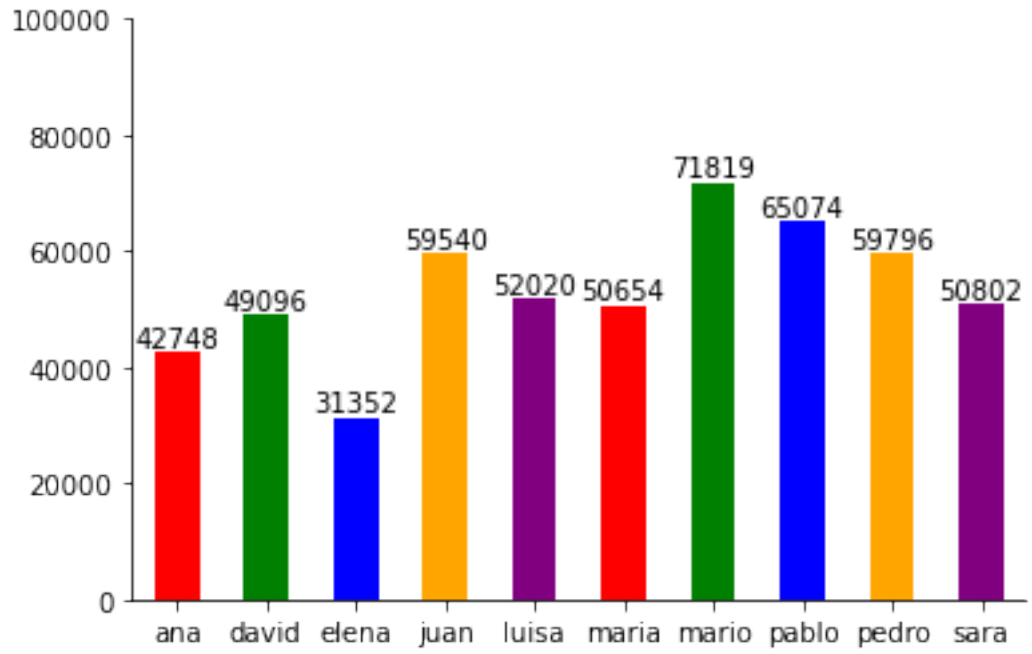
[326]:
```
        fecha vendedor  dia  mes   año  importe nacionalidad  n_emp  n_pro  \
0  2020-08-01     juan    1    8  2020     6106   colombiano      2      1
1  2011-08-01    pedro    1    8  2011     5778     mexicano      1      2
2  2013-09-01      ana    1    9  2013     7870      japones      3      3
3  2018-09-01    maria    1    9  2018     1968     mexicano      1      4
4  2016-03-01    pablo    1    3  2016     8436         ruso      2      5
..        …       …  …  …  …   …       …           …    …     …
95 2014-10-01    luisa    1   10  2014     6113   colombiano      3      6
96 2015-10-01     sara    1   10  2015     2309   colombiano      4      7
97 2012-01-01    david    1    1  2012     2249    argentino      2      8
98 2012-03-01    elena    1    3  2012     3101     mexicano      4      9
99 2013-09-01    mario    1    9  2013     5442    argentino      2      0

    n_obs_vend  imp_vend
0           10      6106
1           10      5778
2           10      7870
3           10     16958
4           10      8436
..         …       …
95          10      9012
96          10      2309
97          10     18335
98          10      3101
99          10     13525

[100 rows x 11 columns]
```

[ ]:
```

```

3.0.1 3.1 Barras: Importe total por vendedor

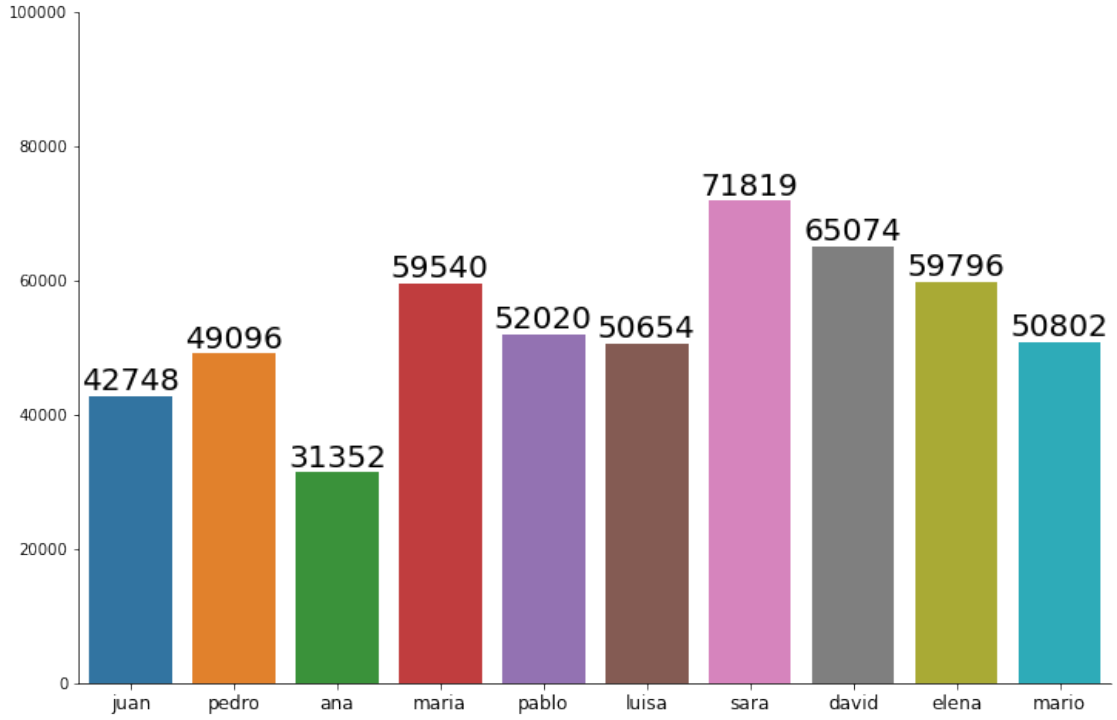
[363]:
```
# MAT

colors = ['red', 'green', 'blue', 'orange', 'purple']

fig, ax = plt.subplots()
gh = df.groupby('vendedor')['importe'].sum() # ponemos bien los datos
gh.plot(kind='bar', color=colors, ax=ax) # tipo de grafico
ax.set_xticklabels(gh.index, rotation=0) # eje x
plt.ylim(0, 100000) # eje y
```

```
ax.set_xlabel('')# sin nombre eje x
ax.spines['right'].set_visible(False) # quitar marcos
ax.spines['top'].set_visible(False)

# Añadir etiquetas con el valor de cada barra
for i, v in enumerate(gh.values):
    ax.text(i, v+1, str(v), ha='center', va='bottom')

#plt.savefig('pib_paises.png', bbox_inches='tight')  # exportamos
plt.show()
```



[367]:
```
# SEABORN

gh = df.groupby('vendedor')['importe'].sum().reset_index() # ponemos bien los␣
↪datos

fig, ax = plt.subplots(figsize=(12,8))
sns.barplot(data=gh, x='vendedor', y='importe', ci=None, ax=ax) # importante el␣
↪ci para que no salgan intervalos de confianza
ax.set_xticklabels(df['vendedor'], rotation=0, fontsize = 12)
ax.set_xlabel('')
ax.set_ylabel('')
ax.set_ylim(0, 100000)
sns.despine(top=True, right=True)  # quitar marcos de arriba y derecha
```

```
for i, v in enumerate(gh['importe']):
    ax.annotate(str(v), xy=(i, v), ha='center', va='bottom', fontsize = 20)

plt.show()
```

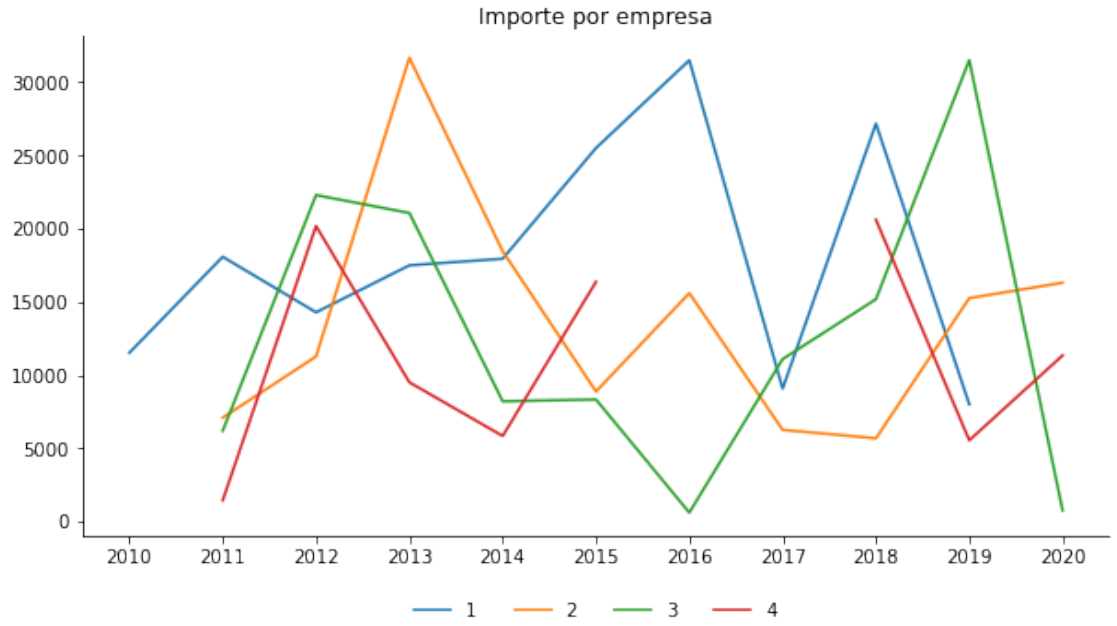


[ ]:

3.0.2 3.2 Lineas: Importe por empresa

[368]:
```
# MAT

df_w = df.groupby(['n_emp', 'año'])['importe'].sum().reset_index()
df_w = df_w.pivot(index = 'año', columns = 'n_emp', values = 'importe')
df_w

fig, ax = plt.subplots(figsize=(10, 5)) # definimos el tamaño del grafico
for emp in df_w.columns:
    ax.plot(df_w.index, df_w[emp], label=emp)

ticks = np.arange(2010, 2021, 1) # perosnalizar eje x
ax.set_xticks(ticks) # para que salgan todos los años
ax.set_xticklabels(df_w.index, rotation=0)  # por si queremos poner otra␣
 ↪rotación
```

```
ax.legend(loc='upper center', frameon=False, ncol=4, bbox_to_anchor=(0.5, -0.1))


ax.set_title('Importe por empresa')
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

#plt.savefig('pib_paises.png', bbox_inches='tight')
plt.show()
```



[370]:
```
df_w = df.groupby(['n_emp', 'año'])['importe'].sum().reset_index()
df_w = df_w.pivot(index = 'año', columns = 'n_emp', values = 'importe')
df_w
```

[370]:

n_emp	1	2	3	4
año				
2010	11518.0	NaN	NaN	NaN
2011	18071.0	7085.0	6188.0	1439.0
2012	14281.0	11266.0	22287.0	20172.0
2013	17489.0	31663.0	21062.0	9501.0
2014	17939.0	18449.0	8204.0	5836.0
2015	25499.0	8875.0	8326.0	16364.0
2016	31493.0	15589.0	602.0	NaN
2017	9075.0	6253.0	11085.0	NaN
2018	27164.0	5675.0	15177.0	20603.0
2019	7994.0	15247.0	31501.0	5540.0

```
2020       NaN  16301.0    752.0  11336.0
```

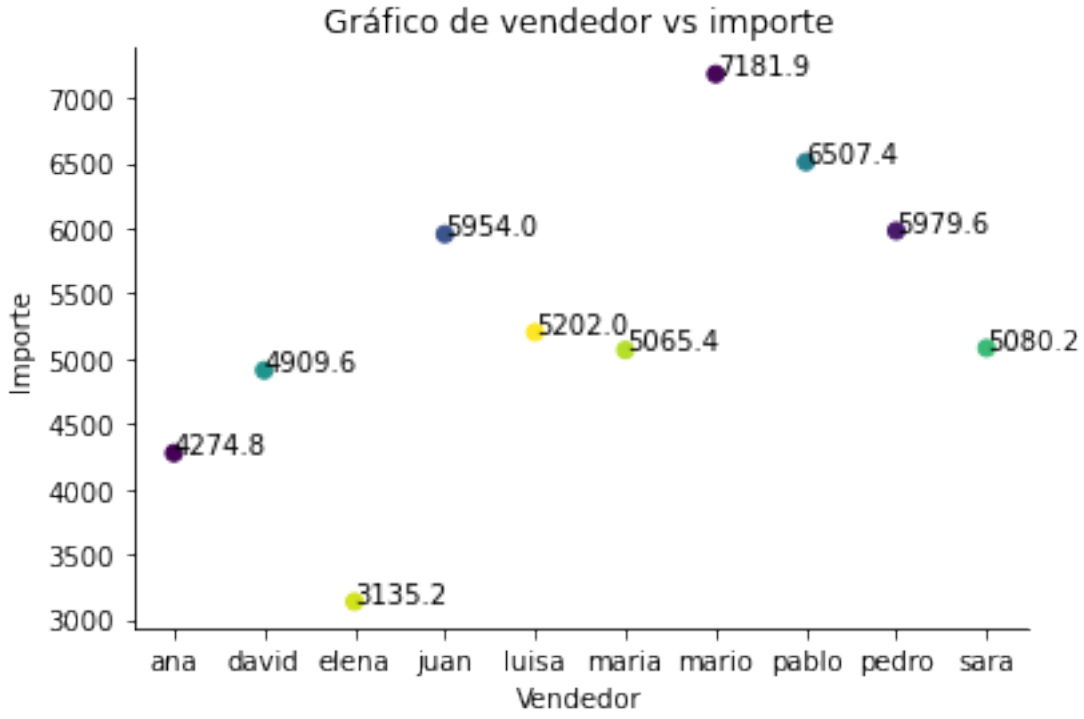
[381]:
```
# SEABORN

df_w = df.groupby(['n_emp', 'año'])['importe'].sum().reset_index()
df_w = df_w.pivot(index = 'año', columns = 'n_emp', values = 'importe')
df_w

sns.set_style("white") # qu no salga el grid
fig, ax = plt.subplots(figsize=(12, 8))

sns.lineplot(data=df_w, x=df_w.index, y=1, color="red", linewidth=2, label="1")
sns.lineplot(data=df_w, x=df_w.index, y=2, color="green", linewidth=1,␣
 ↪linestyle="--", label="2")
sns.lineplot(data=df_w, x=df_w.index, y=3, color="pink", linewidth=1,␣
 ↪linestyle="--", label="3")

ax.set_xticks(range(2010, 2021, 1))
ax.set_yticks(range(0, 50000, 10000))

ax.spines["right"].set_visible(False)
ax.spines["top"].set_visible(False)
ax.set_ylabel("")

ax.legend(loc="upper center", frameon=False, ncol=3, bbox_to_anchor=(0.5, -0.1))
#ax.text(0.15, -0.2, "Fuente: datos de ejemplo", ha="center", transform=ax.
 ↪transAxes, fontsize=12)
#ax.axvline(x=2014, color='black', linestyle='--')

#plt.savefig("pib_paises_seaborn.png", bbox_inches="tight")
plt.show()
```

[ ]:

3.0.3 3.1 Scatter: Importe medio vendido por vendedor

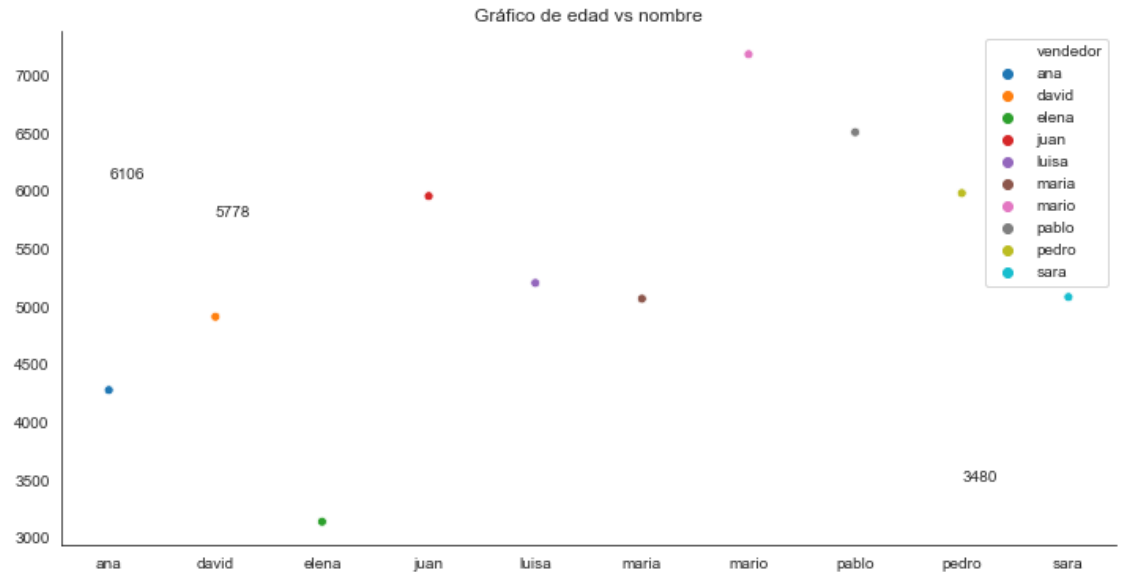
[354]:
```
# MAT.

gh = df.groupby('vendedor')['importe'].mean().reset_index()


diferentes = gh['vendedor'].nunique() # contamos cuantos nombres diferentes hay
colors = np.random.rand(diferentes)

fig, ax = plt.subplots()
ax.scatter(gh['vendedor'], gh['importe'], c=colors)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
#ax.set_ylim(0, 50)
ax.set_xlabel('Vendedor')
ax.set_ylabel('Importe')
ax.set_title('Gráfico de vendedor vs importe')
```

```
#ax.text(0.5, -0.2, 'Fuente: datos de ejemplo', ha='center', transform=ax.
 ↪transAxes, fontsize=12)

for x, y, val in zip(gh['vendedor'], gh['importe'], gh['importe']):
    ax.annotate(str(val), xy=(x, y), textcoords='data')
plt.show()
```



[386]:
```
# SEABORN

gh = df.groupby('vendedor')['importe'].mean().reset_index()
diferentes = gh['vendedor'].nunique() # contamos cuantos nombres diferentes hay
colors = np.random.rand(diferentes)


# Creamos la figura y el objeto de ejes
fig, ax = plt.subplots(figsize=(12,6))

# Creamos el scatterplot
sns.scatterplot(data=gh, x='vendedor', y='importe', hue='vendedor', ax=ax)

# Personalizamos el eje y
#ax.set_ylim(0, 50)
```

```
# Quitamos los marcos de arriba y derecha
sns.despine(top=True, right=True)

# Añadimos el título, las etiquetas de los ejes y la fuente
ax.set_title('Gráfico de edad vs nombre')
ax.set_xlabel('')
ax.set_ylabel('')


# Añadimos los valores de edad encima de cada punto
for x, y, val in zip(gh['vendedor'], df['importe'], df['importe']):
    ax.annotate(str(val), xy=(x, y), textcoords='data')
#ax.legend(loc='upper center', frameon=False, ncol=4, bbox_to_anchor=(0.5, -0.
 ↪1), fontsize=15)
plt.show()
```



[ ]:

4 4 Tratamiento de Datos

[10]:
```
# Vamos a cagar un excel

ruta = "/Users/adrian_gr/Desktop/4.cnmv/04.practica/datos_tratamiento.xlsx"
datos = pd.read_excel(ruta, sheet_name = 'aqui df', header = 3)
datos
```

[10]:
```
    Unnamed: 0  Unnamed: 1      fecha vendedor  dia  mes   año  importe  \
0          NaN         NaN 2020-08-01     juan    1    8  2020   6106.0
1          NaN         NaN 2011-08-01    pedro    1    8  2011   5778.0
2          NaN         NaN 2013-09-01      ana    1    9  2013   7870.0
3          NaN         NaN 2018-09-01    maria    1    9  2018   1968.0
4          NaN         NaN 2016-03-01    pablo    1    3  2016   8436.0
..         …           …        …       …  …  …   …      …
95         NaN         NaN 2014-10-01    luisa    1   10  2014   6113.0
96         NaN         NaN 2015-10-01     sara    1   10  2015   2309.0
97         NaN         NaN 2012-01-01    david    1    1  2012   2249.0
98         NaN         NaN 2012-03-01    elena    1    3  2012      NaN
99         NaN         NaN 2013-09-01    mario    1    9  2013   5442.0

   nacionalidad  n_emp  n_pro  n_obs_vend  salario
0    colombiano      2      1          10  59540.0
1      mexicano      1      2          10  59796.0
2       japones      3      3          10  42748.0
3      mexicano      1      4          10  50654.0
4          ruso      2      5          10      NaN
..          …    …    …          …       …
95   colombiano      3      6          10  52020.0
96   colombiano      4      7          10  50802.0
97    argentino      2      8          10      NaN
98     mexicano      4      9          10  31352.0
99    argentino      2      0          10  71819.0

[100 rows x 13 columns]
```

[11]:
```
# borramos las columnas mal

col = datos.columns

for j in col:
    if "Unna" in j:
        datos.drop(j, axis = 1, inplace = True)
datos
```

[11]:
```
        fecha vendedor  dia  mes   año  importe nacionalidad  n_emp  n_pro  \
0  2020-08-01     juan    1    8  2020   6106.0   colombiano      2      1
```

```
1  2011-08-01    pedro    1    8  2011   5778.0     mexicano      1      2
2  2013-09-01      ana    1    9  2013   7870.0      japones      3      3
3  2018-09-01    maria    1    9  2018   1968.0     mexicano      1      4
4  2016-03-01    pablo    1    3  2016   8436.0         ruso      2      5
..        …      …  …  …   …      …          …    …     …
95 2014-10-01    luisa    1   10  2014   6113.0   colombiano      3      6
96 2015-10-01     sara    1   10  2015   2309.0   colombiano      4      7
97 2012-01-01    david    1    1  2012   2249.0    argentino      2      8
98 2012-03-01    elena    1    3  2012      NaN     mexicano      4      9
99 2013-09-01    mario    1    9  2013   5442.0    argentino      2      0

    n_obs_vend  salario
0           10  59540.0
1           10  59796.0
2           10  42748.0
3           10  50654.0
4           10      NaN
..         …      …
95          10  52020.0
96          10  50802.0
97          10      NaN
98          10  31352.0
99          10  71819.0

[100 rows x 11 columns]
```

[12]:
```
datos.salario.describe()
```

[12]:
```
count       86.000000
mean     53055.662791
std      11355.790682
min      31352.000000
25%      49096.000000
50%      50802.000000
75%      59796.000000
max      71819.000000
Name: salario, dtype: float64
```

4.0.1 4.1 Tratamientos de missing

[13]:
```
# Cuantos missing hay

print(datos.isna().sum())
datos['salario'].isna().sum()
```

```
fecha            0
vendedor         0
```

```
dia              0
mes              0
año              0
importe         25
nacionalidad     0
n_emp            0
n_pro            0
n_obs_vend       0
salario         14
dtype: int64
```

[13]:
```
14
```

[14]:
```
# Una primera prueba es crear una variable nueva y remplzar el missing por 0
# Ya vemos como es el replace de stata

datos['aux'] = datos['salario']
datos.loc[(datos['salario'].isna()) & (datos['dia'] == 1), 'aux'] = 0
datos
datos.drop('aux', axis = 1, inplace = True)
```

[15]:
```
datos
```

[15]:

	fecha	vendedor	dia	mes	año	importe	nacionalidad	n_emp	n_pro	\
0	2020-08-01	juan	1	8	2020	6106.0	colombiano	2	1	
1	2011-08-01	pedro	1	8	2011	5778.0	mexicano	1	2	
2	2013-09-01	ana	1	9	2013	7870.0	japones	3	3	
3	2018-09-01	maria	1	9	2018	1968.0	mexicano	1	4	
4	2016-03-01	pablo	1	3	2016	8436.0	ruso	2	5	
..	…	…	…	…	…	…	…	…	…	
95	2014-10-01	luisa	1	10	2014	6113.0	colombiano	3	6	
96	2015-10-01	sara	1	10	2015	2309.0	colombiano	4	7	
97	2012-01-01	david	1	1	2012	2249.0	argentino	2	8	
98	2012-03-01	elena	1	3	2012	NaN	mexicano	4	9	
99	2013-09-01	mario	1	9	2013	5442.0	argentino	2	0	

	n_obs_vend	salario
0	10	59540.0
1	10	59796.0
2	10	42748.0
3	10	50654.0
4	10	NaN
..	…	…
95	10	52020.0
96	10	50802.0
97	10	NaN
98	10	31352.0

```
99          10  71819.0

[100 rows x 11 columns]
```

[16]:
```
# Otra opción borramos las observaciones que esas variables tiene missing si %
 ↪menos a 25%
obs = len(datos)

col = ['importe', "salario"]

for j in col:

    if (datos[j].isna().sum() / obs) < 0.24:
        print(j)
        datos = datos.dropna(subset=[j])
len(datos)
```

```
salario
```

[16]:
```
86
```

[17]:
```
# Generamos una variable para identificar a los missing de importe

#datos['aux'] = datos['importe'].isna().astype(int)
#datos
```

[18]:
```
# CON LOS SIGUIENTES PASOS HACEMOS TODO
datos['v2'] = 1
datos
```

```
<ipython-input-18-a67f779153fd>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  datos['v2'] = 1
```

[18]:

	fecha	vendedor	dia	mes	año	importe	nacionalidad	n_emp	n_pro	\
0	2020-08-01	juan	1	8	2020	6106.0	colombiano	2	1	
1	2011-08-01	pedro	1	8	2011	5778.0	mexicano	1	2	
2	2013-09-01	ana	1	9	2013	7870.0	japones	3	3	
3	2018-09-01	maria	1	9	2018	1968.0	mexicano	1	4	
5	2011-08-01	luisa	1	8	2011	2166.0	italiano	3	6	
..	…	…	…	…	…	…	…	…	…	
94	2013-05-01	pablo	1	5	2013	9501.0	aleman	4	5	
95	2014-10-01	luisa	1	10	2014	6113.0	colombiano	3	6	

```
96 2015-10-01     sara    1   10  2015   2309.0   colombiano      4      7
98 2012-03-01    elena    1    3  2012      NaN     mexicano      4      9
99 2013-09-01    mario    1    9  2013   5442.0    argentino      2      0

    n_obs_vend  salario  v2
0           10  59540.0   1
1           10  59796.0   1
2           10  42748.0   1
3           10  50654.0   1
5           10  52020.0   1
..         …      …  ..
94          10  65074.0   1
95          10  52020.0   1
96          10  50802.0   1
98          10  31352.0   1
99          10  71819.0   1

[86 rows x 12 columns]
```

[19]:
```
# count if missing(v1) & v2 == 1

sum((datos['importe'].isnull()) & (datos['v2'] == 1))
```

[19]:
```
19
```

[20]:
```
# drop if missing(v1) & v2 == 1

datos2 = datos.drop(datos[(datos['importe'].isnull()) & (datos['v2'] == 1)].
 ↪index)
datos2
```

[20]:
```
         fecha vendedor  dia  mes   año  importe nacionalidad  n_emp  n_pro  \
0   2020-08-01     juan    1    8  2020   6106.0   colombiano      2      1
1   2011-08-01    pedro    1    8  2011   5778.0     mexicano      1      2
2   2013-09-01      ana    1    9  2013   7870.0      japones      3      3
3   2018-09-01    maria    1    9  2018   1968.0     mexicano      1      4
5   2011-08-01    luisa    1    8  2011   2166.0     italiano      3      6
..         …      …  …  …  …    …       …          …    …     …
91  2014-02-01    pedro    1    2  2014   2091.0         ruso      3      2
94  2013-05-01    pablo    1    5  2013   9501.0       aleman      4      5
95  2014-10-01    luisa    1   10  2014   6113.0   colombiano      3      6
96  2015-10-01     sara    1   10  2015   2309.0   colombiano      4      7
99  2013-09-01    mario    1    9  2013   5442.0    argentino      2      0

    n_obs_vend  salario  v2
0           10  59540.0   1
1           10  59796.0   1
```

```
2           10  42748.0   1
3           10  50654.0   1
5           10  52020.0   1
..         …      …  ..
91          10  59796.0   1
94          10  65074.0   1
95          10  52020.0   1
96          10  50802.0   1
99          10  71819.0   1

[67 rows x 12 columns]
```

[21]:
```
# keep if missing(v1) & v2 == 1

datos2 = datos[(datos['importe'].isnull()) & (datos['v2'] == 1)]
datos2
```

[21]:
```
        fecha vendedor  dia  mes   año  importe nacionalidad  n_emp  n_pro  \
10 2013-01-01     juan    1    1  2013      NaN      frances      3      1
13 2018-03-01    maria    1    3  2018      NaN   colombiano      4      4
21 2016-09-01    pedro    1    9  2016      NaN     italiano      2      2
25 2014-03-01    luisa    1    3  2014      NaN     italiano      2      6
28 2013-09-01    elena    1    9  2013      NaN     italiano      3      9
29 2016-05-01    mario    1    5  2016      NaN     italiano      1      0
30 2015-02-01     juan    1    2  2015      NaN      japones      4      1
31 2015-10-01    pedro    1   10  2015      NaN    argentino      1      2
44 2015-01-01    pablo    1    1  2015      NaN    argentino      4      5
45 2013-01-01    luisa    1    1  2013      NaN     italiano      2      6
46 2018-05-01     sara    1    5  2018      NaN    argentino      2      7
55 2019-07-01    luisa    1    7  2019      NaN      japones      3      6
56 2018-02-01     sara    1    2  2018      NaN    argentino      4      7
57 2018-03-01    david    1    3  2018      NaN   colombiano      1      8
77 2014-02-01    david    1    2  2014      NaN        chino      4      8
78 2011-07-01    elena    1    7  2011      NaN     mexicano      1      9
85 2011-04-01    luisa    1    4  2011      NaN        chino      1      6
93 2014-03-01    maria    1    3  2014      NaN       aleman      2      4
98 2012-03-01    elena    1    3  2012      NaN     mexicano      4      9

    n_obs_vend  salario  v2
10          10  59540.0   1
13          10  50654.0   1
21          10  59796.0   1
25          10  52020.0   1
28          10  31352.0   1
29          10  71819.0   1
30          10  59540.0   1
31          10  59796.0   1
```

```
44          10  65074.0   1
45          10  52020.0   1
46          10  50802.0   1
55          10  52020.0   1
56          10  50802.0   1
57          10  49096.0   1
77          10  49096.0   1
78          10  31352.0   1
85          10  52020.0   1
93          10  50654.0   1
98          10  31352.0   1
```

[22]:
```
# bys vendedor año : egen aux = mean(importe)
datos['aux'] = datos.groupby(['vendedor', 'año'])['importe'].transform('mean')

# replace v1 = aux if missing(v1) & v2 == 1
datos.loc[(datos['importe'].isnull()) & (datos['v2'] == 1), 'importe'] =␣
↪datos['aux']
datos
```

```
<ipython-input-22-5012a1300c0a>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  datos['aux'] = datos.groupby(['vendedor', 'año'])['importe'].transform('mean')
```

[22]:
```
        fecha vendedor  dia  mes   año  importe nacionalidad  n_emp  n_pro  \
0  2020-08-01     juan    1    8  2020   6106.0   colombiano      2      1
1  2011-08-01    pedro    1    8  2011   5778.0     mexicano      1      2
2  2013-09-01      ana    1    9  2013   7870.0      japones      3      3
3  2018-09-01    maria    1    9  2018   1968.0     mexicano      1      4
5  2011-08-01    luisa    1    8  2011   2166.0     italiano      3      6
..        …       …   …  …  …    …       …          …    …     …
94 2013-05-01    pablo    1    5  2013   9501.0       aleman      4      5
95 2014-10-01    luisa    1   10  2014   6113.0   colombiano      3      6
96 2015-10-01     sara    1   10  2015   2309.0   colombiano      4      7
98 2012-03-01    elena    1    3  2012      NaN     mexicano      4      9
99 2013-09-01    mario    1    9  2013   5442.0    argentino      2      0

    n_obs_vend  salario  v2     aux
0           10  59540.0   1  6106.0
1           10  59796.0   1  5778.0
2           10  42748.0   1  7870.0
3           10  50654.0   1  3440.0
5           10  52020.0   1  2166.0
```

```
..          …      …  ..     …
94          10  65074.0   1  9501.0
95          10  52020.0   1  6113.0
96          10  50802.0   1  2309.0
98          10  31352.0   1     NaN
99          10  71819.0   1  6762.5

[86 rows x 13 columns]
```

[23]:
```
# Vemos cuantos missing quedan y borramos

print(datos['importe'].isna().sum())

datos.dropna(subset = 'importe', inplace =  True)

print(datos['importe'].isna().sum())
```

```
8
0

<ipython-input-23-9be172608ff2>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  datos.dropna(subset = 'importe', inplace =  True)
```

[24]:
```
# Cambiar un valor por una condcion

datos.loc[(datos['vendedor'] == "david") & (datos['año'] > 2010),␣
 ↪'nacionalidad'] = "españa"
datos[datos['vendedor'] == "david"]
```

[24]:
```
         fecha vendedor  dia  mes   año  importe nacionalidad  n_emp  n_pro  \
7   2014-11-01    david    1   11  2014   9871.0       españa      1      8
17  2020-04-01    david    1    4  2020    498.0       españa      2      8
37  2012-02-01    david    1    2  2012   1344.0       españa      4      8
47  2019-11-01    david    1   11  2019   7188.0       españa      2      8
67  2012-07-01    david    1    7  2012   7279.0       españa      3      8
77  2014-02-01    david    1    2  2014   9871.0       españa      4      8
87  2015-06-01    david    1    6  2015   1437.0       españa      1      8

    n_obs_vend  salario  v2     aux
7           10  49096.0   1  9871.0
17          10  49096.0   1   498.0
37          10  49096.0   1  4311.5
47          10  49096.0   1  7188.0
67          10  49096.0   1  4311.5
```

```
77          10  49096.0   1  9871.0
87          10  49096.0   1  1437.0
```

[25]:
```
# Generar una varible nueva en base a una condición

datos.loc[datos['nacionalidad'] == 'españa', 'continente'] = "europa"
datos[datos['nacionalidad'] == 'españa']
```

```
<ipython-input-25-167fe78afbec>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  datos.loc[datos['nacionalidad'] == 'españa', 'continente'] = "europa"
```

[25]:
```
        fecha vendedor  dia  mes   año  importe nacionalidad  n_emp  n_pro  \
7  2014-11-01    david    1   11  2014   9871.0       españa      1      8
17 2020-04-01    david    1    4  2020    498.0       españa      2      8
37 2012-02-01    david    1    2  2012   1344.0       españa      4      8
47 2019-11-01    david    1   11  2019   7188.0       españa      2      8
67 2012-07-01    david    1    7  2012   7279.0       españa      3      8
77 2014-02-01    david    1    2  2014   9871.0       españa      4      8
87 2015-06-01    david    1    6  2015   1437.0       españa      1      8

    n_obs_vend  salario  v2     aux continente
7           10  49096.0   1  9871.0     europa
17          10  49096.0   1   498.0     europa
37          10  49096.0   1  4311.5     europa
47          10  49096.0   1  7188.0     europa
67          10  49096.0   1  4311.5     europa
77          10  49096.0   1  9871.0     europa
87          10  49096.0   1  1437.0     europa
```

[26]:
```
# Tema de keep and drop.

# keep solo david y españa

datos[(datos['vendedor'] == "david") & (datos['nacionalidad'] == "españa")]


# drop

datos.drop(datos[datos['importe'].isnull()].index, inplace=True)

# para tema missing mejor hacerlo asi
```

```
datos.dropna(subset=['importe'], inplace=True)

# borrar por dos condicones

datos.drop(datos[(datos['vendedor'] == "david") & (datos['nacionalidad'] ==␣
 ↪"españa")].index, inplace = True)
```

```
<ipython-input-26-e5048050f309>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  datos.drop(datos[datos['importe'].isnull()].index, inplace=True)
<ipython-input-26-e5048050f309>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  datos.dropna(subset=['importe'], inplace=True)
<ipython-input-26-e5048050f309>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  datos.drop(datos[(datos['vendedor'] == "david") & (datos['nacionalidad'] ==
"españa")].index, inplace = True)
```

[27]:
```
# no hay nada

datos[(datos['vendedor'] == "david") & (datos['nacionalidad'] == "españa")]
```

[27]:
```
Empty DataFrame
Columns: [fecha, vendedor, dia, mes, año, importe, nacionalidad, n_emp, n_pro,
n_obs_vend, salario, v2, aux, continente]
Index: []
```

[ ]:
```

```

4.0.2 4.2 Estandarizacíon

[49]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
```

[38]:
```
# Traemos un df de los de antes

vendedores = ['Juan', 'Pedro', 'Ana', 'María', 'Pablo', 'Luisa', 'Sara',␣
 ↪'David', 'Elena', 'Mario'] * 10
```

```
dias = [1]*100
meses = np.random.randint(1, 13, size=100)
anios = np.random.randint(2010, 2021, size=100)
productos = ['Producto1', 'Producto2', 'Producto3', 'Producto4', 'Producto5',␣
↪'Producto6', 'Producto7', 'Producto8', 'Producto9', 'Producto10']* 10
importes = np.random.randint(100, 10000, size=100)
empresas = ['Empresa1', 'Empresa2', 'Empresa3', 'Empresa4']
nacionalidades = ['Español', 'Italiano', 'Alemán', 'Frances', 'Ruso', 'Chino',␣
↪'Japonés', 'Argentino', 'Mexicano', 'Colombiano']

# Crear DataFrame
data = {'Vendedor': vendedores,
        'Dia': dias,
        'Mes': meses,
        'Año': anios,
        'Producto': productos,
        'Importe': importes,
        'Empresa': np.random.choice(empresas, size=100),
        'Nacionalidad': np.random.choice(nacionalidades, size=100)}

df = pd.DataFrame(data)

# Mostrar las primeras 5 filas del DataFrame

print(df.shape)
df.head()
```

```
(100, 8)
```

[38]:

	Vendedor	Dia	Mes	Año	Producto	Importe	Empresa	Nacionalidad
0	Juan	1	4	2013	Producto1	2442	Empresa2	Ruso
1	Pedro	1	5	2020	Producto2	4162	Empresa4	Español
2	Ana	1	9	2018	Producto3	1835	Empresa2	Chino
3	María	1	12	2016	Producto4	4556	Empresa1	Argentino
4	Pablo	1	9	2016	Producto5	7568	Empresa1	Frances

[39]:
```
salario = np.random.randint(30000, 80001, size=len(df))
salario
```

[39]:
```
array([44138, 61019, 32515, 64894, 44464, 52214, 51299, 62602, 71314,
       57167, 59186, 51771, 52137, 36369, 54677, 70393, 44554, 70944,
       38327, 49496, 35800, 37833, 50716, 40714, 51329, 53666, 32128,
       66160, 74950, 71094, 71732, 68255, 56987, 36802, 47188, 64420,
       48750, 46841, 32707, 65700, 36751, 58797, 51992, 54682, 68586,
       64848, 68413, 64664, 69424, 73270, 32945, 42013, 63645, 70357,
       70937, 45883, 66275, 46510, 62022, 54080, 42303, 58178, 32529,
       64898, 78578, 62214, 43822, 52551, 41498, 38578, 40047, 79741,
```

```
        74063, 66391, 49924, 44215, 56727, 78179, 64897, 39459, 42225,
        72101, 57236, 76945, 49799, 60959, 35007, 52819, 70028, 55393,
        66956, 36787, 57965, 75048, 31763, 64634, 49354, 76025, 72387,
        42033])
```

[40]:
```
# Creamos dos variablesnueva que metemos al dd

salario = np.random.randint(30000, 80001, size=len(df))
puesto = np.random.choice(['empleado', 'jefe', 'director'], size=len(df))

df['salario'] = salario
df['puesto'] = puesto
df
```

[40]:
```
   Vendedor  Dia  Mes   Año    Producto  Importe   Empresa Nacionalidad  \
0      Juan    1    4  2013   Producto1     2442  Empresa2         Ruso
1     Pedro    1    5  2020   Producto2     4162  Empresa4      Español
2       Ana    1    9  2018   Producto3     1835  Empresa2        Chino
3     María    1   12  2016   Producto4     4556  Empresa1    Argentino
4     Pablo    1    9  2016   Producto5     7568  Empresa1      Frances
..      …  …  …   …         …       …        …           …
95    Luisa    1    9  2020   Producto6      139  Empresa1      Japonés
96     Sara    1   11  2011   Producto7     9918  Empresa4    Argentino
97    David    1    1  2017   Producto8     5717  Empresa1    Argentino
98    Elena    1    5  2012   Producto9     8215  Empresa4    Argentino
99    Mario    1    2  2019  Producto10     6977  Empresa3     Mexicano

    salario    puesto
0     40218      jefe
1     57692  director
2     64085  empleado
3     53962  director
4     73980  director
..      …       …
95    59133  empleado
96    30312      jefe
97    55590  director
98    57834  director
99    67304  empleado

[100 rows x 10 columns]
```

[47]:
```
# Vamos a estandarizar las variable importe y salario

scaler = StandardScaler()
df['importe2'] = scaler.fit_transform(df[['Importe']])
df
```

[47]:
```
   Vendedor  Dia  Mes   Año    Producto  Importe   Empresa Nacionalidad  \
0      Juan    1    4  2013   Producto1     2442  Empresa2         Ruso
1     Pedro    1    5  2020   Producto2     4162  Empresa4      Español
2       Ana    1    9  2018   Producto3     1835  Empresa2        Chino
3     María    1   12  2016   Producto4     4556  Empresa1    Argentino
4     Pablo    1    9  2016   Producto5     7568  Empresa1      Frances
..      …  …  …   …         …       …         …         …
95    Luisa    1    9  2020   Producto6      139  Empresa1      Japonés
96     Sara    1   11  2011   Producto7     9918  Empresa4    Argentino
97    David    1    1  2017   Producto8     5717  Empresa1    Argentino
98    Elena    1    5  2012   Producto9     8215  Empresa4    Argentino
99    Mario    1    2  2019  Producto10     6977  Empresa3     Mexicano

    salario    puesto  importe2
0     40218      jefe -0.883226
1     57692  director -0.260226
2     64085  empleado -1.103087
3     53962  director -0.117515
4     73980  director  0.973460
..      …       …       …
95    59133  empleado -1.717395
96    30312      jefe  1.824652
97    55590  director  0.303010
98    57834  director  1.207809
99    67304  empleado  0.759394

[100 rows x 11 columns]
```

[50]:
```
# Normalizacion

scaler = MinMaxScaler()
df['importe3'] = scaler.fit_transform(df[['Importe']])
df
```

[50]:
```
   Vendedor  Dia  Mes   Año    Producto  Importe   Empresa Nacionalidad  \
0      Juan    1    4  2013   Producto1     2442  Empresa2         Ruso
1     Pedro    1    5  2020   Producto2     4162  Empresa4      Español
2       Ana    1    9  2018   Producto3     1835  Empresa2        Chino
3     María    1   12  2016   Producto4     4556  Empresa1    Argentino
4     Pablo    1    9  2016   Producto5     7568  Empresa1      Frances
..      …  …  …   …         …       …         …         …
95    Luisa    1    9  2020   Producto6      139  Empresa1      Japonés
96     Sara    1   11  2011   Producto7     9918  Empresa4    Argentino
97    David    1    1  2017   Producto8     5717  Empresa1    Argentino
98    Elena    1    5  2012   Producto9     8215  Empresa4    Argentino
99    Mario    1    2  2019  Producto10     6977  Empresa3     Mexicano
```

```
    salario    puesto  importe2  importe3
0     40218      jefe -0.883226  0.237299
1     57692  director -0.260226  0.412773
2     64085  empleado -1.103087  0.175372
3     53962  director -0.117515  0.452969
4     73980  director  0.973460  0.760253
..      …       …        …        …
95    59133  empleado -1.717395  0.002346
96    30312      jefe  1.824652  1.000000
97    55590  director  0.303010  0.571414
98    57834  director  1.207809  0.826260
99    67304  empleado  0.759394  0.699959

[100 rows x 12 columns]
```

[ ]:

4.0.3 4.3 Variables categóricas

[51]:
```
from sklearn.preprocessing import OneHotEncoder
```

[55]:
```
# Crear una instancia de OneHotEncoder
encoder = OneHotEncoder(sparse=False)

# Ajustar y transformar el codificador a la columna "Ciudad"
vendedor_encoded = encoder.fit_transform(df[["Vendedor"]])

# Crear un DataFrame con las variables codificadas
df_encoded = pd.DataFrame(vendedor_encoded, columns=encoder.
 ↪get_feature_names(['Vendedor']))

# Hacemos el concat horizonta

df = pd.concat([df, df_encoded], axis = 1)
df.drop('Vendedor', axis = 1, inplace = True)
df
```

[55]:
```
    Dia  Mes   Año    Producto  Importe   Empresa Nacionalidad  salario  \
0     1    4  2013   Producto1     2442  Empresa2         Ruso    40218
1     1    5  2020   Producto2     4162  Empresa4      Español    57692
2     1    9  2018   Producto3     1835  Empresa2        Chino    64085
3     1   12  2016   Producto4     4556  Empresa1    Argentino    53962
4     1    9  2016   Producto5     7568  Empresa1      Frances    73980
..  …  …    …          …       …          …           …       …
95    1    9  2020   Producto6      139  Empresa1      Japonés    59133
96    1   11  2011   Producto7     9918  Empresa4    Argentino    30312
97    1    1  2017   Producto8     5717  Empresa1    Argentino    55590
```

```
98    1    5  2012   Producto9     8215  Empresa4    Argentino    57834
99    1    2  2019  Producto10     6977  Empresa3     Mexicano    67304

      puesto  importe2  …  Vendedor_Ana  Vendedor_David  Vendedor_Elena  \
0       jefe -0.883226  …           0.0             0.0             0.0
1   director -0.260226  …           0.0             0.0             0.0
2   empleado -1.103087  …           1.0             0.0             0.0
3   director -0.117515  …           0.0             0.0             0.0
4   director  0.973460  …           0.0             0.0             0.0
..       …       …  …           …               …               …
95  empleado -1.717395  …           0.0             0.0             0.0
96      jefe  1.824652  …           0.0             0.0             0.0
97  director  0.303010  …           0.0             1.0             0.0
98  director  1.207809  …           0.0             0.0             1.0
99  empleado  0.759394  …           0.0             0.0             0.0

    Vendedor_Juan  Vendedor_Luisa  Vendedor_Mario  Vendedor_María  \
0             1.0             0.0             0.0             0.0
1             0.0             0.0             0.0             0.0
2             0.0             0.0             0.0             0.0
3             0.0             0.0             0.0             1.0
4             0.0             0.0             0.0             0.0
..            …               …               …               …
95            0.0             1.0             0.0             0.0
96            0.0             0.0             0.0             0.0
97            0.0             0.0             0.0             0.0
98            0.0             0.0             0.0             0.0
99            0.0             0.0             1.0             0.0

    Vendedor_Pablo  Vendedor_Pedro  Vendedor_Sara
0              0.0             0.0            0.0
1              0.0             1.0            0.0
2              0.0             0.0            0.0
3              0.0             0.0            0.0
4              1.0             0.0            0.0
..             …               …              …
95             0.0             0.0            0.0
96             0.0             0.0            1.0
97             0.0             0.0            0.0
98             0.0             0.0            0.0
99             0.0             0.0            0.0

[100 rows x 21 columns]
```

[57]:
```
# Hcaemos una variable continua

df['n_emp'], _ = pd.factorize(df['Empresa'])
```

```
df
```

[57]:

	Dia	Mes	Año	Producto	Importe	Empresa	Nacionalidad	salario	\
0	1	4	2013	Producto1	2442	Empresa2	Ruso	40218	
1	1	5	2020	Producto2	4162	Empresa4	Español	57692	
2	1	9	2018	Producto3	1835	Empresa2	Chino	64085	
3	1	12	2016	Producto4	4556	Empresa1	Argentino	53962	
4	1	9	2016	Producto5	7568	Empresa1	Frances	73980	
..	…	…	…	…	…	…	…	…	
95	1	9	2020	Producto6	139	Empresa1	Japonés	59133	
96	1	11	2011	Producto7	9918	Empresa4	Argentino	30312	
97	1	1	2017	Producto8	5717	Empresa1	Argentino	55590	
98	1	5	2012	Producto9	8215	Empresa4	Argentino	57834	
99	1	2	2019	Producto10	6977	Empresa3	Mexicano	67304	

	puesto	importe2	…	Vendedor_David	Vendedor_Elena	Vendedor_Juan	\
0	jefe	-0.883226	…	0.0	0.0	1.0	
1	director	-0.260226	…	0.0	0.0	0.0	
2	empleado	-1.103087	…	0.0	0.0	0.0	
3	director	-0.117515	…	0.0	0.0	0.0	
4	director	0.973460	…	0.0	0.0	0.0	
..	…	…	…	…	…	…	
95	empleado	-1.717395	…	0.0	0.0	0.0	
96	jefe	1.824652	…	0.0	0.0	0.0	
97	director	0.303010	…	1.0	0.0	0.0	
98	director	1.207809	…	0.0	1.0	0.0	
99	empleado	0.759394	…	0.0	0.0	0.0	

	Vendedor_Luisa	Vendedor_Mario	Vendedor_María	Vendedor_Pablo	\
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	1.0	0.0	
4	0.0	0.0	0.0	1.0	
..	…	…	…	…	
95	1.0	0.0	0.0	0.0	
96	0.0	0.0	0.0	0.0	
97	0.0	0.0	0.0	0.0	
98	0.0	0.0	0.0	0.0	
99	0.0	1.0	0.0	0.0	

	Vendedor_Pedro	Vendedor_Sara	n_emp
0	0.0	0.0	0
1	1.0	0.0	1
2	0.0	0.0	0
3	0.0	0.0	2
4	0.0	0.0	2

```
..              …              …    …
95              0.0            0.0      2
96              0.0            1.0      1
97              0.0            0.0      2
98              0.0            0.0      1
99              0.0            0.0      3

[100 rows x 22 columns]
```

[ ]:

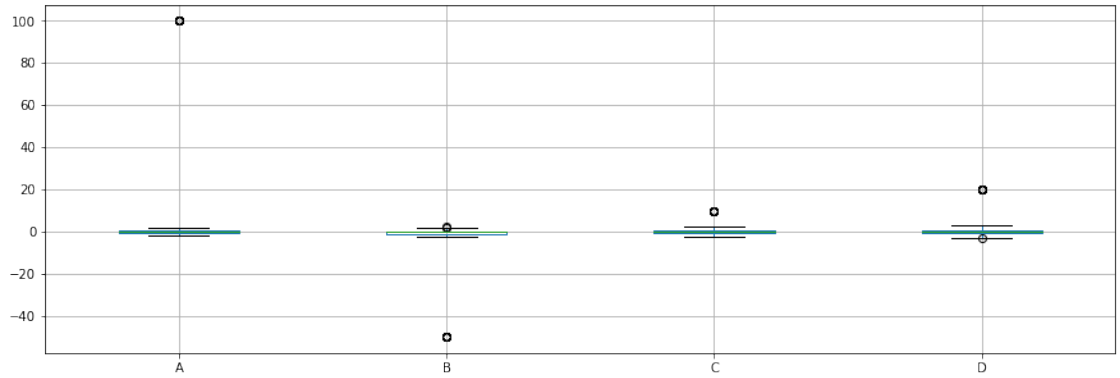
[ ]:

4.0.4 4.4 Valores atipicos

[58]:
```
# Generamos el dataframe

# Creamos un DataFrame con 4 columnas: 'A', 'B', 'C' y 'D'
np.random.seed(123)
data = pd.DataFrame(np.random.randn(100, 4), columns=['A', 'B', 'C', 'D'])

# Agregamos algunos valores atípicos
data.iloc[0:10, 0] = 100
data.iloc[10:20, 1] = -50
data.iloc[20:30, 2] = 10
data.iloc[30:40, 3] = 20
data
```

[58]:
```
              A         B         C         D
0   100.000000  0.997345  0.282978 -1.506295
1   100.000000  1.651437 -2.426679 -0.428913
2   100.000000 -0.866740 -0.678886 -0.094709
3   100.000000 -0.638902 -0.443982 -0.434351
4   100.000000  2.186786  1.004054  0.386186
..         …       …         …         …
95    0.789828 -1.007509 -1.305786 -0.882829
96   -0.346090  0.109403 -0.772584  0.744819
97    0.251464 -0.694798  0.888993  1.161068
98   -0.098685 -0.214983 -1.773771 -0.407513
99   -0.291507  0.245379 -0.168426  0.244027

[100 rows x 4 columns]
```

[59]:
```
# Visualizamos la distribución
plt.figure(figsize=(15, 5))
data.boxplot()
plt.show() # vemos que hay outliers
```



[60]:
```
# Detección por mas de 2 devia iones tipicaS. Sería mejor poner 3 pero en este caso no hay

for col in data.columns:
    mean = data[col].mean()
    std = data[col].std()
    data[f"{col}_outlier"] = ((data[col] < mean - 2 * std) | (data[col] > mean + 2 * std)).astype(int)
data[data['A_outlier'] == 1]
```

[60]:

	A	B	C	D	A_outlier	B_outlier	C_outlier	D_outlier
0	100.0	0.997345	0.282978	-1.506295	1	0	0	0
1	100.0	1.651437	-2.426679	-0.428913	1	0	0	0
2	100.0	-0.866740	-0.678886	-0.094709	1	0	0	0
3	100.0	-0.638902	-0.443982	-0.434351	1	0	0	0
4	100.0	2.186786	1.004054	0.386186	1	0	0	0
5	100.0	1.490732	-0.935834	1.175829	1	0	0	0
6	100.0	-0.637752	0.907105	-1.428681	1	0	0	0
7	100.0	-0.861755	-0.255619	-2.798589	1	0	0	0
8	100.0	-0.699877	0.927462	-0.173636	1	0	0	0
9	100.0	0.688223	-0.879536	0.283627	1	0	0	0

[61]:
```
# Forma muy elegante de generar variable binaria en base a algo. Voy a hacerlo␣
↪más simple
# si C es menor de -2 y D menos que 0 valga 1

data['aux'] = ((data.C < -2) & (data.D < 0)).astype(int)
data[data['aux'] == 1]
```

[61]:

	A	B	C	D	A_outlier	B_outlier	C_outlier	\
1	100.000000	1.651437	-2.426679	-0.428913	1	0	0	
92	0.419568	0.206928	-2.251535	-0.588971	0	0	0	

	D_outlier	aux
1	0	1
92	0	1

[69]:
```
# Borramos las obs donde hay outlier en A y en B a la vez

print(len(data[(data['A_outlier'] == 1) & (data['B_outlier'] == 1)])) # esto␣
↪nos dice que hay 0


data.drop(data[(data['A_outlier'] == 1) & (data['B_outlier'] == 1)].index,␣
↪inplace=True)
data
```

```
0
```

[69]:

	A	B	C	D	A_outlier	B_outlier	C_outlier	\
0	100.000000	0.997345	0.282978	-1.506295	1	0	0	
1	100.000000	1.651437	-2.426679	-0.428913	1	0	0	
2	100.000000	-0.866740	-0.678886	-0.094709	1	0	0	
3	100.000000	-0.638902	-0.443982	-0.434351	1	0	0	
4	100.000000	2.186786	1.004054	0.386186	1	0	0	
..	…	…	…	…	…	…	…	
95	0.789828	-1.007509	-1.305786	-0.882829	0	0	0	
96	-0.346090	0.109403	-0.772584	0.744819	0	0	0	
97	0.251464	-0.694798	0.888993	1.161068	0	0	0	
98	-0.098685	-0.214983	-1.773771	-0.407513	0	0	0	
99	-0.291507	0.245379	-0.168426	0.244027	0	0	0	

	D_outlier	aux
0	0	0
1	0	1
2	0	0
3	0	0
4	0	0
..	…	…

```
95          0    0
96          0    0
97          0    0
98          0    0
99          0    0

[100 rows x 9 columns]
```

[70]:
```
# Borramos las obs donde A es outlier
data.drop(data[data['A_outlier'] == 1].index, inplace=True)
data.drop('A_outlier', axis =1, inplace=True)
data
```

[70]:
```
           A          B         C         D  B_outlier  C_outlier  D_outlier  \
10 -0.805367 -50.000000 -0.390900  0.573806          1          0          0
11  0.338589 -50.000000  2.392365  0.412912          1          0          0
12  0.978736 -50.000000 -1.294085 -1.038788          1          0          0
13  1.743712 -50.000000  0.029683  1.069316          1          0          0
14  0.890706 -50.000000  1.495644  1.069393          1          0          0
..       …        …        …        …         …          …          …
95  0.789828  -1.007509 -1.305786 -0.882829          0          0          0
96 -0.346090   0.109403 -0.772584  0.744819          0          0          0
97  0.251464  -0.694798  0.888993  1.161068          0          0          0
98 -0.098685  -0.214983 -1.773771 -0.407513          0          0          0
99 -0.291507   0.245379 -0.168426  0.244027          0          0          0

    aux
10    0
11    0
12    0
13    0
14    0
..  …
95    0
96    0
97    0
98    0
99    0

[90 rows x 8 columns]
```

[74]:
```
# Ahora queremos borrar todo donde haya outlier

cols_out = data.filter(like='out').columns # seleccion de columna

data['out'] = data[cols_out].sum(axis=1)
data.drop(data[data['out'] > 0].index, inplace=True)
```

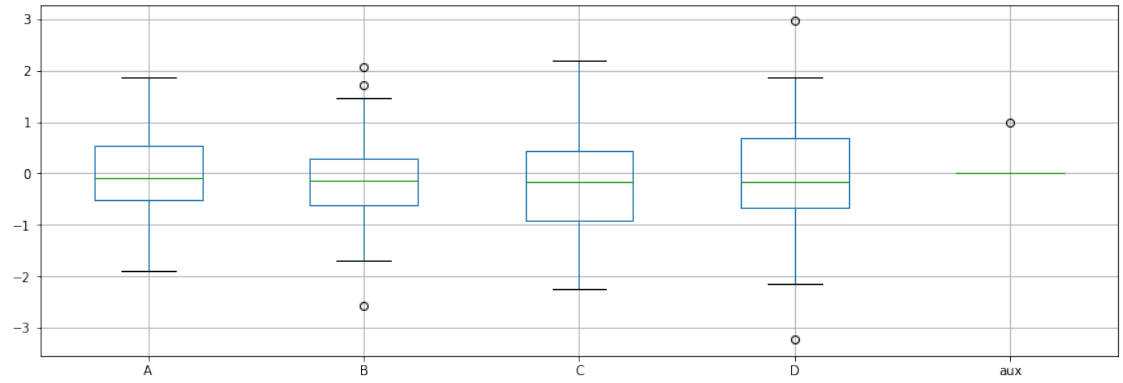
```
data.head()
```

[74]:

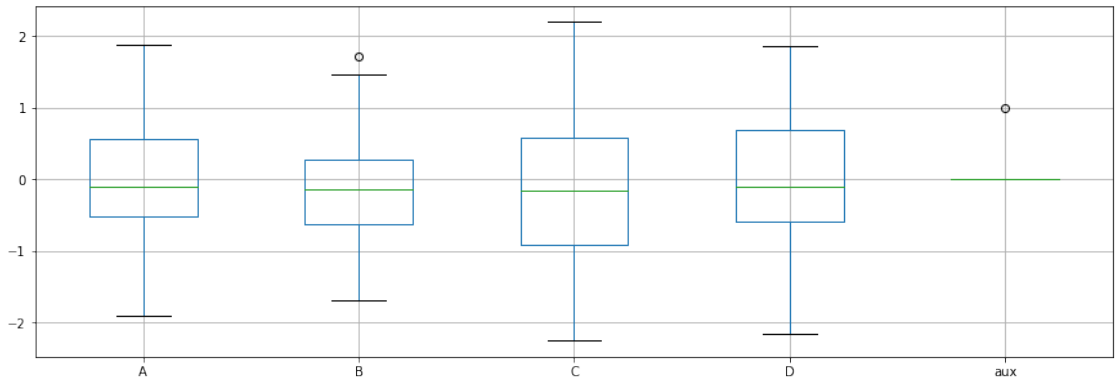
	A	B	C	D	B_outlier	C_outlier	D_outlier	\
40	0.020316	-0.193964	0.134027	0.704474	0	0	0	
41	0.665653	-0.898423	1.523664	-1.095026	0	0	0	
42	0.079227	-0.274397	-1.048992	-0.075121	0	0	0	
43	-0.740814	0.072907	0.403086	1.471929	0	0	0	
44	0.307384	-0.611225	-0.391620	0.139978	0	0	0	

	aux	out
40	0	0
41	0	0
42	0	0
43	0	0
44	0	0

[76]:
```
# Visualizamos la distribución
plt.figure(figsize=(15, 5))
data.boxplot()
plt.show() # vemos que hay outliers
```



[77]:
```
## Vemos que sigue habiendo puntos fuera en B y C

q = data[['B', 'D']].quantile(0.99)
p = data[['B', 'D']].quantile(0.01)
data = data[(data['B'] < q[0]) & (data['B'] > p[0] )]
data = data[(data['D'] < q[1]) & (data['D'] > p[1] )]
# Visualizamos la distribución
plt.figure(figsize=(15, 5))
data.boxplot()
plt.show() # vemos que hay outliers
```



[ ]:

5 EXAMEN

[78]:
```
# Decir en que dir estas

os.getcwd()
```

[78]: '/Users/adrian_gr/Desktop/4.cnmv/04.practica'

[81]:
```
# Ver todos los archivos de este directorio y haga una lista solo de los df

print(os.listdir())

ar = os.listdir()

df = []
for j in ar:
    if "df" in j:
        df = df + [j]
df
```

['RTP - Oposicon Tecnico Analista de Datos CNMV.ipynb', '.DS_Store', 'df.xlsx', 'df.csv', '05. modelos', '.ipynb_checkpoints', 'df_general.xlsx', '04. tratamiento', '02. visualizacion', '01. python_avanzado', '06. NLP y Web Scraping', 'df.txt', 'RTP - guia.ipynb', 'datos_tratamiento.xlsx', '03. SQL']

[81]: ['df.xlsx', 'df.csv', 'df_general.xlsx', 'df.txt']

[82]:
```
# Cambiar el directorio

os.chdir('/Users/adrian_gr/Desktop/4.cnmv/04.practica')
```

[94]:
```
# Hacenos basic s con este df

vendedores = ['Juan', 'Pedro', 'Ana', 'María', 'Pablo', 'Luisa', 'Sara',␣
 ↪'David', 'Elena', 'Mario'] * 10
dias = [1]*100
meses = np.random.randint(1, 13, size=100)
anios = np.random.randint(2010, 2021, size=100)
productos = ['Producto1', 'Producto2', 'Producto3', 'Producto4', 'Producto5',␣
 ↪'Producto6', 'Producto7', 'Producto8', 'Producto9', 'Producto10']* 10
importes = np.random.randint(100, 10000, size=100)
empresas = ['Empresa1', 'Empresa2', 'Empresa3', 'Empresa4']
nacionalidades = ['Español', 'Italiano', 'Alemán', 'Frances', 'Ruso', 'Chino',␣
 ↪'Japonés', 'Argentino', 'Mexicano', 'Colombiano']

# Crear DataFrame
data = {'Vendedor': vendedores,
        'Dia': dias,
        'Mes': meses,
        'Año': anios,
        'Producto id': productos,
        'Importe': importes,
        'Empresa': np.random.choice(empresas, size=100),
        'Nacionalidad': np.random.choice(nacionalidades, size=100)}

df = pd.DataFrame(data)

# Mostrar las primeras 5 filas del DataFrame
df.head()
```

[94]:

	Vendedor	Dia	Mes	Año	Producto id	Importe	Empresa	Nacionalidad
0	Juan	1	7	2016	Producto1	6930	Empresa2	Colombiano
1	Pedro	1	2	2015	Producto2	7785	Empresa2	Colombiano
2	Ana	1	7	2016	Producto3	3223	Empresa3	Alemán
3	María	1	9	2016	Producto4	852	Empresa4	Español
4	Pablo	1	1	2019	Producto5	756	Empresa4	Español

[95]:
```
# Todas las variables en minusculas

for j in df.columns:
    n = str.lower(j)
    df.rename(columns={j: n}, inplace=True)
df
```

[95]:

	vendedor	dia	mes	año	producto id	importe	empresa	nacionalidad
0	Juan	1	7	2016	Producto1	6930	Empresa2	Colombiano
1	Pedro	1	2	2015	Producto2	7785	Empresa2	Colombiano
2	Ana	1	7	2016	Producto3	3223	Empresa3	Alemán

```
3     María    1    9  2016   Producto4      852  Empresa4      Español
4     Pablo    1    1  2019   Producto5      756  Empresa4      Español
..      …  …  …   …         …       …        …          …
95    Luisa    1   10  2017   Producto6     2620  Empresa3      Español
96     Sara    1   11  2019   Producto7      862  Empresa2       Alemán
97    David    1   12  2020   Producto8     4915  Empresa4     Italiano
98    Elena    1   11  2012   Producto9     4920  Empresa3        Chino
99    Mario    1    5  2020  Producto10     5968  Empresa1        Chino

[100 rows x 8 columns]
```

[99]:
```
# quitamos los espacios

for j in df.columns:
    n = str.replace(j, " ", "_")
    df.rename(columns={j: n}, inplace=True)
df
```

[99]:
```
   vendedor  dia  mes   año producto_id  importe   empresa nacionalidad
0      Juan    1    7  2016   Producto1     6930  Empresa2   Colombiano
1     Pedro    1    2  2015   Producto2     7785  Empresa2   Colombiano
2       Ana    1    7  2016   Producto3     3223  Empresa3       Alemán
3     María    1    9  2016   Producto4      852  Empresa4      Español
4     Pablo    1    1  2019   Producto5      756  Empresa4      Español
..      …  …  …   …         …       …        …          …
95    Luisa    1   10  2017   Producto6     2620  Empresa3      Español
96     Sara    1   11  2019   Producto7      862  Empresa2       Alemán
97    David    1   12  2020   Producto8     4915  Empresa4     Italiano
98    Elena    1   11  2012   Producto9     4920  Empresa3        Chino
99    Mario    1    5  2020  Producto10     5968  Empresa1        Chino

[100 rows x 8 columns]
```

[92]:
```
# Convertimos dia a str y año a numerica

df['dia'] = df.dia.astype('str')
df['año'] = df.año.astype('int')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   vendedor      100 non-null    object
 1   dia           100 non-null    object
```

```
 2   mes           100 non-null    int64
 3   año           100 non-null    int64
 4   producto      100 non-null    object
 5   importe       100 non-null    int64
 6   empresa       100 non-null    object
 7   nacionalidad  100 non-null    object
dtypes: int64(3), object(5)
memory usage: 6.4+ KB
```

[93]:
```
# Posibles nomnbre

df.vendedor.unique()
```

[93]:
```
array(['Juan', 'Pedro', 'Ana', 'María', 'Pablo', 'Luisa', 'Sara', 'David',
       'Elena', 'Mario'], dtype=object)
```

[103]:
```
# Los nombres en minuscular

df['vendedor'] = df['vendedor'].str.lower()
```

[104]:
```
# Representatividad de cada uno

df.vendedor.value_counts()
```

[104]:
```
juan     10
pedro    10
ana      10
maría    10
pablo    10
luisa    10
sara     10
david    10
elena    10
mario    10
Name: vendedor, dtype: int64
```

[109]:
```
# Que cuente cuantas obs hay de david hay en 2020


len(df[(df['vendedor'] == "david") & (df['año'] == 2020)])
```

[109]:
```
4
```

[153]:
```
# que me muestre solo algunas variables

df[['vendedor', 'año']]
```

[153]:
```
   vendedor   año
0      juan  2016
1     pedro  2015
2       ana  2016
3     maría  2016
4     pablo  2019
..      …   …
91    pablo  2017
92    luisa  2017
93     sara  2019
94    elena  2012
95    mario  2020

[96 rows x 2 columns]
```

[110]:
```
# Que me muestre solo esas obs

df[(df['vendedor'] == "david") & (df['año'] == 2020)]
```

[110]:
```
   vendedor  dia  mes   año producto_id  importe   empresa nacionalidad
17    david    1    9  2020   Producto8     5407  Empresa3      Japonés
37    david    1    7  2020   Producto8     9758  Empresa3      Frances
87    david    1    5  2020   Producto8     8550  Empresa2       Alemán
97    david    1   12  2020   Producto8     4915  Empresa4     Italiano
```

[111]:
```
# Que me elimine estas obs

df.drop(df[(df['vendedor'] == "david") & (df['año'] == 2020)].index , inplace =␣
 ↪True)
df
```

[111]:
```
   vendedor  dia  mes   año producto_id  importe   empresa nacionalidad
0      juan    1    7  2016   Producto1     6930  Empresa2   Colombiano
1     pedro    1    2  2015   Producto2     7785  Empresa2   Colombiano
2       ana    1    7  2016   Producto3     3223  Empresa3       Alemán
3     maría    1    9  2016   Producto4      852  Empresa4      Español
4     pablo    1    1  2019   Producto5      756  Empresa4      Español
..      …   …   …   …          …       …         …
94    pablo    1    5  2017   Producto5     6895  Empresa1     Mexicano
95    luisa    1   10  2017   Producto6     2620  Empresa3      Español
96     sara    1   11  2019   Producto7      862  Empresa2       Alemán
98    elena    1   11  2012   Producto9     4920  Empresa3        Chino
99    mario    1    5  2020  Producto10     5968  Empresa1        Chino

[96 rows x 8 columns]
```

[117]:
```
# Hacemos algun GB

df[df['vendedor'] == "ana"].groupby(['vendedor', 'año'])['importe'].sum().
↪reset_index()
```

[117]:

	vendedor	año	importe
0	ana	2010	2309
1	ana	2011	6527
2	ana	2015	4766
3	ana	2016	11831
4	ana	2017	6376
5	ana	2019	7709
6	ana	2020	3041

[157]:
```
# Generamos columnas nuevas con valores

# Cuente cuantas veces se repite una combinacion de vendedor y año
df['aux'] = df.groupby(['vendedor', 'año'])['vendedor'].transform('count')

# Que me genere el total de importe por vendedor y año
df['aux2'] = df.groupby(['vendedor', 'año'])['importe'].transform('sum')

# Que me cuente en termino de todas las variables, por si hay duplicados

col = df.columns
df['aux3'] = df.groupby(list(df.columns))['vendedor'].transform('count')

df
```

[157]:

	vendedor	dia	mes	año	producto_id	importe	empresa	nacionalidad	aux	\
0	juan	1	7	2016	Producto1	6930	Empresa2	Colombiano	1	
1	pedro	1	2	2015	Producto2	7785	Empresa2	Colombiano	1	
2	ana	1	7	2016	Producto3	3223	Empresa3	Alemán	2	
3	maría	1	9	2016	Producto4	852	Empresa4	Español	1	
4	pablo	1	1	2019	Producto5	756	Empresa4	Español	1	
..	…	…	…	…	…	…	…	…	…	
91	pablo	1	5	2017	Producto5	6895	Empresa1	Mexicano	3	
92	luisa	1	10	2017	Producto6	2620	Empresa3	Español	1	
93	sara	1	11	2019	Producto7	862	Empresa2	Alemán	1	
94	elena	1	11	2012	Producto9	4920	Empresa3	Chino	1	
95	mario	1	5	2020	Producto10	5968	Empresa1	Chino	2	

	aux2	aux3
0	6930	1
1	7785	1
2	11831	1
3	852	1

```
4     756     1
..    …   …
91  10269     1
92   2620     1
93    862     1
94   4920     1
95  13245     1

[96 rows x 11 columns]
```

[137]:
```
# Hacemos un collpase

df.groupby('vendedor').agg({'importe' : 'sum', 'empresa' : 'first',␣
↪'nacionalidad' : 'last'}).reset_index()
```

[137]:
```
  vendedor  importe   empresa nacionalidad
0      ana    42559  Empresa3         Ruso
1    david    30505  Empresa3     Mexicano
2    elena    47439  Empresa4        Chino
3     juan    55501  Empresa2      Frances
4    luisa    36065  Empresa4      Español
5    mario    49482  Empresa2        Chino
6    maría    60910  Empresa4      Español
7    pablo    36471  Empresa4     Mexicano
8    pedro    41483  Empresa2        Chino
9     sara    51492  Empresa2       Alemán
```

[148]:
```
# Hacemos un collpase

df.groupby('vendedor').agg({'nacionalidad' : 'last'}).reset_index()
```

[148]:
```
  vendedor nacionalidad
0      ana         Ruso
1    david     Mexicano
2    elena        Chino
3     juan      Frances
4    luisa      Español
5    mario        Chino
6    maría      Español
7    pablo     Mexicano
8    pedro        Chino
9     sara       Alemán
```

[146]:
```
# Hacemos una prueba poniendo el vendedor como indice

df = df.set_index('vendedor') # lo ponemos como indice
df.head()
```

[146]:
```
           dia  mes   año producto_id  importe   empresa nacionalidad  aux  \
vendedor
juan         1    7  2016   Producto1     6930  Empresa2   Colombiano    1
pedro        1    2  2015   Producto2     7785  Empresa2   Colombiano    1
ana          1    7  2016   Producto3     3223  Empresa3       Alemán    2
maría        1    9  2016   Producto4      852  Empresa4      Español    1
pablo        1    1  2019   Producto5      756  Empresa4      Español    1

           aux2
vendedor
juan       6930
pedro      7785
ana       11831
maría       852
pablo       756
```

[147]:
```
# Lo sacamos del indice

df.reset_index(inplace=True) # lo quitamos
df.rename(columns={'index': 'vendedor'}, inplace=True) # lesto es por si el␣
 ↪indice viene sin nombre
df
```

[147]:
```
    vendedor  dia  mes   año producto_id  importe   empresa nacionalidad  aux  \
0       juan    1    7  2016   Producto1     6930  Empresa2   Colombiano    1
1      pedro    1    2  2015   Producto2     7785  Empresa2   Colombiano    1
2        ana    1    7  2016   Producto3     3223  Empresa3       Alemán    2
3      maría    1    9  2016   Producto4      852  Empresa4      Español    1
4      pablo    1    1  2019   Producto5      756  Empresa4      Español    1
..       …   …   …   …          …       …          …   …
91     pablo    1    5  2017   Producto5     6895  Empresa1     Mexicano    3
92     luisa    1   10  2017   Producto6     2620  Empresa3      Español    1
93      sara    1   11  2019   Producto7      862  Empresa2       Alemán    1
94     elena    1   11  2012   Producto9     4920  Empresa3        Chino    1
95     mario    1    5  2020  Producto10     5968  Empresa1        Chino    2

     aux2
0    6930
1    7785
2   11831
3     852
4     756
..    …
91  10269
92   2620
93    862
94   4920
```

```
95  13245

[96 rows x 10 columns]
```

[163]:
```
# Preparamops la estructura del concat

df_c1 = df[['año', 'vendedor', 'importe']][df['año'] >= 2015]
df_c2 = df[['año', 'importe', 'nacionalidad']][df['año'] < 2015]

df_c = pd.concat([df_c1, df_c2], axis = 0)
df_c[df_c['año'] <2015].head()
```

[163]:

	año	vendedor	importe	nacionalidad
6	2012	NaN	722	Ruso
7	2014	NaN	9043	Colombiano
9	2013	NaN	9556	Alemán
10	2014	NaN	7602	Italiano
12	2011	NaN	6527	Alemán

[178]:
```
# Preparamos la estructura del merge
# SOlo una obs por año y vendedor

df_m1 = df[['año', 'vendedor', 'importe']]
df_m1 = df_m1.groupby(['vendedor', 'año']).agg({'importe' : 'sum'})

# para m2 tenemos que hacer lo mismo, solo una nacionalidad por vendedor y año
df_m2 = df[['nacionalidad', 'vendedor', 'año']][df['año'] >= 2012]
df_m2 = df_m2.groupby(['vendedor', 'año']).agg({'nacionalidad' : 'first'})


df_m = df_m1.merge(df_m2, on = ['vendedor', 'año'], how = 'left').reset_index()

df_m[df_m['año'] < 2012]
```

[178]:

	vendedor	año	importe	nacionalidad
0	ana	2010	2309	NaN
1	ana	2011	6527	NaN
11	elena	2010	8435	NaN
12	elena	2011	3603	NaN
19	juan	2010	6280	NaN
20	juan	2011	12193	NaN
26	luisa	2010	1011	NaN
33	mario	2010	258	NaN
34	mario	2011	751	NaN
40	maría	2011	8727	NaN
49	pablo	2010	9832	NaN
55	pedro	2010	7724	NaN

```
62     sara  2010     5137          NaN
63     sara  2011     7731          NaN
```

[ ]:

[ ]:

5.1 FUNCIONES Y OTROS

[ ]:
```
## TIPS: casi siempre lo mejor es usa rfunciones numpy y pandas pq es lo que␣
 ↪mas rápido se ejecuta
#para hacer algo similar a lo de arriba pero además cambiar la variable edad:
df['sueldo'] = np.where(df['profesion'] == "Ingeniero", 40000, # si es ingeniro␣
 ↪sueldo vale 40k
                        np.where(df['profesion'] == "Abogado", 50000, 30000)) #␣
 ↪si es abogado 50j, y si no 30K
df['edad'] = np.where(df['sueldo'] > 41000, df['edad'] * 2, df['edad'])
df
```

[ ]:
```
def agrupar_nombre(df):
    df_agrupado = df.groupby('nombre').mean()
    return df_agrupado
```

[ ]:
```
# Crear una columna en base a su sueldo y dependiendo de ese sueldo hacer algo␣
 ↪con la edad
def asignar_sueldo(profesion):
    if profesion == "Ingeniero":
        return 40000
    elif profesion == "Abogado":
        return 50000
    else:
        return 30000

df['sueldo'] = df['profesion'].apply(asignar_sueldo)
df
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:
```
help(pd.DataFrame.merge)
```

[ ]:
```
help(pd.DataFrame.agg)
```