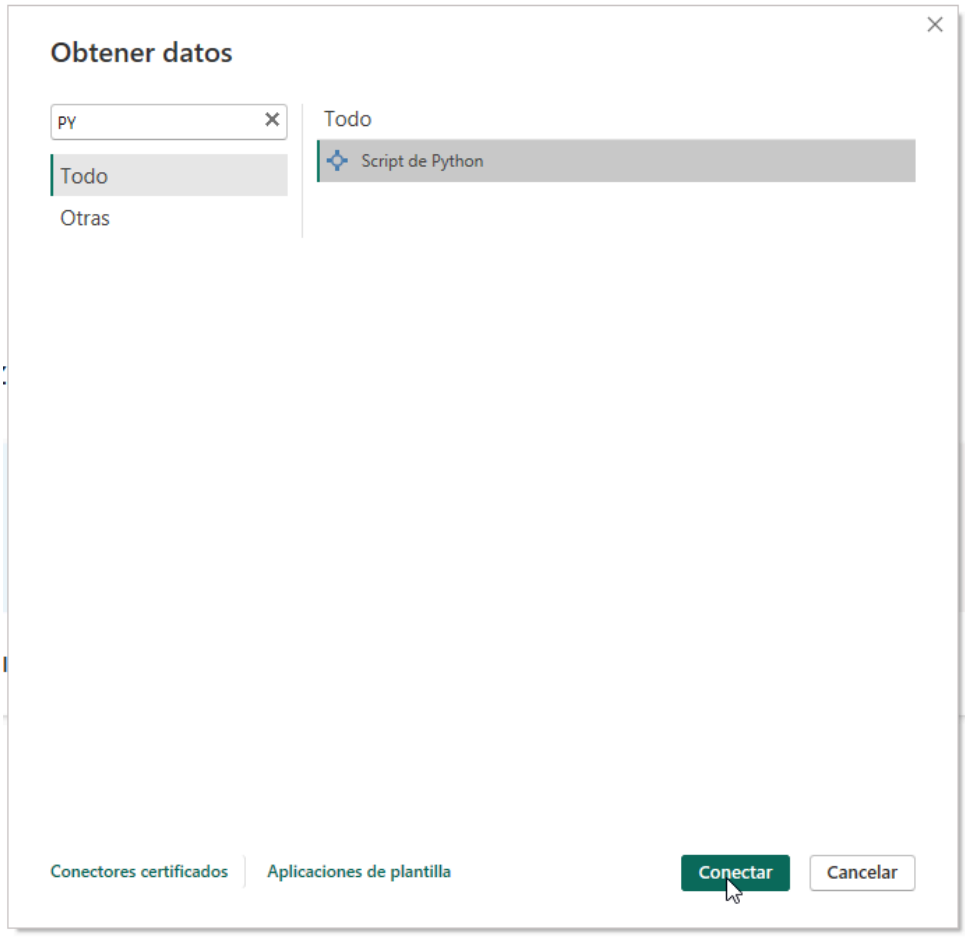
Mailin Adriana Villán Villán
Fecha entrega: 18/03/2025
Revisor: Thais Alcaide Delgado

Tasca S8.02. Power BI con Python

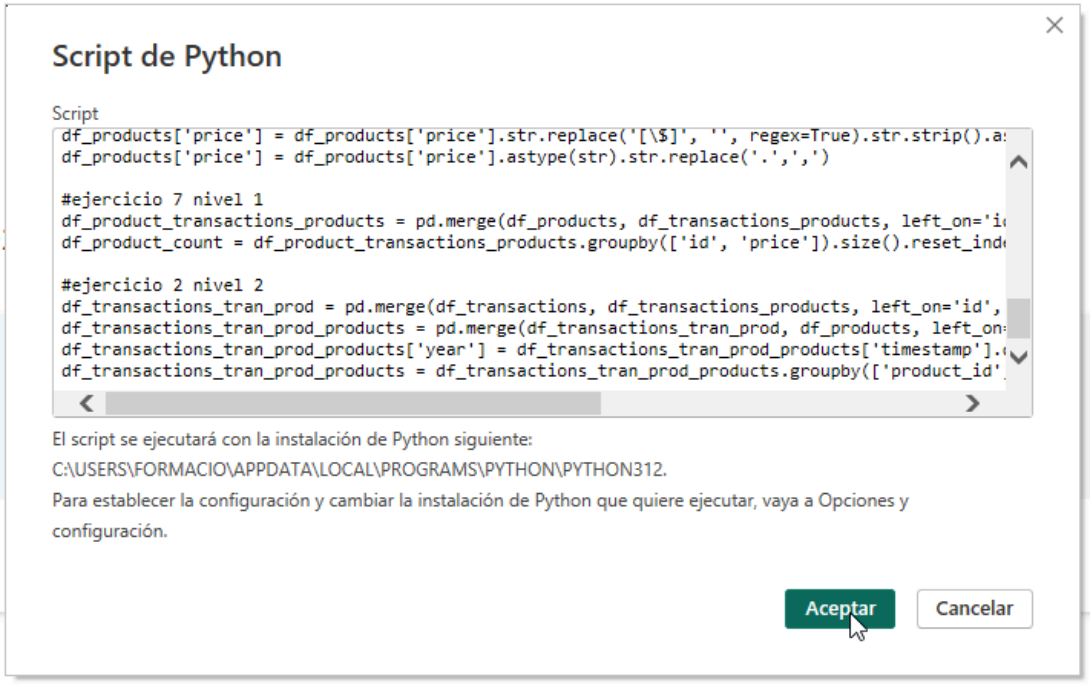
Esta labor consiste en la elaboración de un informe de Power BI, aprovechando las capacidades analíticas de Python. Se utilizarán los scripts de Python creados previamente en la Tarea 1 para generar visualizaciones personalizadas con las bibliotecas Seaborn y Matplotlib. Estas visualizaciones estarán integradas en el informe de Power BI para ofrecer una comprensión más profunda de la capacidad del lenguaje de programación en la herramienta Power BI.

Ajustes previos

Lo primero que hice fue configurar PBI para que sea compatible con Visual Studio Code. Posteriormente realicé la carga de los Dataframes. Para ello, en el módulo inicio de PBI, seleccione la opción obtener datos, busqué python y elegí la opción Script de Python y pulsé sobre conectar.



Luego se abrió una nueva ventana que me permitió cargar el script de python



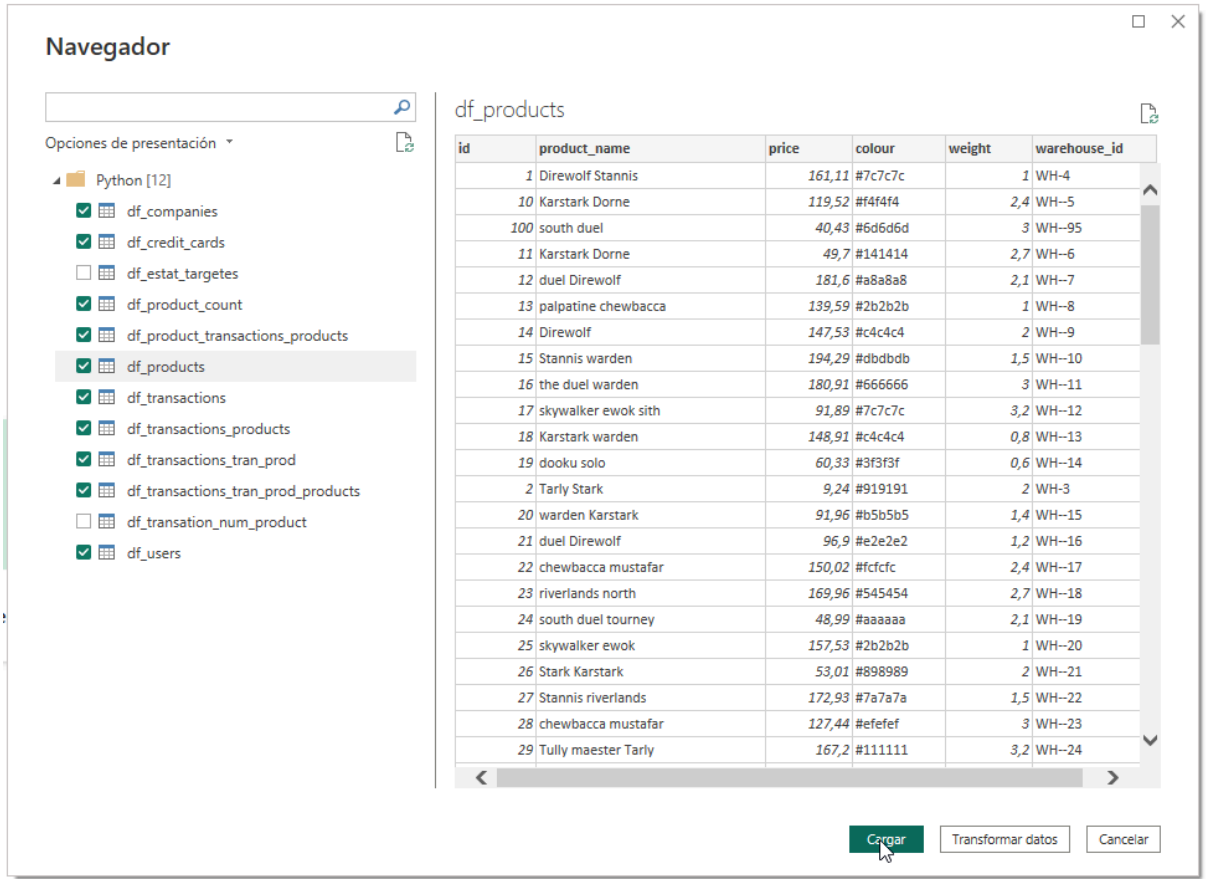
A continuación muestro el script cargado en PBI. Para realizar este script tome como base el script realizado en la tasca S8.01, en el cual he tenido que realizar algunas modificaciones para garantizar su ejecución sin errores dentro de PBI. Las modificaciones realizadas incluyen ajustes en la conexión a MySQL, transformación de datos, cambio de tipo de datos y la inclusión de los DataFrames creados en python.

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import squarify
import mysql.connector

conexion_bd = mysql.connector.connect(host='localhost',
                                      database='transactions',
                                      user='root',
                                      password='root')

cursor = conexion_bd.cursor()
cursor.execute("SHOW TABLES")
MySQL_tables = [table[0] for table in cursor.fetchall()]

for table in MySQL_tables:
    cursor.execute(f"SELECT * FROM {table}") # Ejecuta la consulta "Select * From {tabla}" para extraer los datos de c/tabla
    columnas = [columna[0] for columna in cursor.description] # Obtención de lista de nombres de las columnas de la tabla
    globals()[f'df_{table}'] = pd.DataFrame(cursor.fetchall(), columns=columnas) #Creación y nombramiento dinamico de pd.df
    print(f'df_{table}')

cursor.close()
conexion_bd.close()


# Limpieza de datos

df_products = df_products.astype({'price': object, 'weight': object})
df_transactions = df_transactions.astype({'amount': object, 'declined': bool})

df_transactions['amount'] = df_transactions['amount'].astype(str).str.replace('.',',')
df_products['weight'] = df_products['weight'].astype(str).str.replace('.',',')
df_products['price'] = df_products['price'].str.replace('[\$]', '', regex=True).str.strip().astype(float)
df_products['price'] = df_products['price'].astype(str).str.replace('.',',')

#ejercicio 7 nivel 1
df_product_transactions_products = pd.merge(df_products, df_transactions_products, left_on='id', right_on='product_id', how='outer')
df_product_count = df_product_transactions_products.groupby(['id', 'price']).size().reset_index(name='veces_vendido')

#ejercicio 2 nivel 2
df_transactions_tran_prod = pd.merge(df_transactions, df_transactions_products, left_on='id', right_on='transaction_id')
df_transactions_tran_prod_products = pd.merge(df_transactions_tran_prod, df_products, left_on='product_id', right_on='id')
df_transactions_tran_prod_products['year'] = df_transactions_tran_prod_products['timestamp'].dt.year
df_transactions_tran_prod_products = df_transactions_tran_prod_products.groupby(['product_id', 'price', 'year']).size().reset_index(name='cantidad_productos')
```

En el código anterior se realizan los siguientes pasos:

1. Se importan las librerías necesarias para manipulación de datos (pandas, numpy), visualización (matplotlib.pyplot, seaborn, squarify), y conexión a MySQL (mysql.connector).
2. Se realiza la conexión a la BD MySQL llamada transactions, se crea un cursor para ejecutar consultas SQL y se obtiene una lista con los nombres de todas las tablas dentro de la BD.
3. Se cargan los datos desde MySQL a Pandas DataFrames y se cierra el cursor y la conexión a la BD.
4. Limpieza y transformación de datos
    - Se convierten las columnas de df_products(price, weight) y df_transactions(amount, declined) a los tipos de datos adecuados.
    - Se reemplazan los puntos (.) por comas (,).
    - Se eliminan los signos de dólar ($) en la columna price y se convierten los valores a tipo float, luego se convierten nuevamente a string con comas como separadores decimales.
5. Inclusión de los DataFrames creados en python para el análisis de datos de los ejercicios: 7 (nivel 1) y 2 (nivel 2)

Al ejecutar este código en PBI se abre una ventana de selección de datos donde elegí los DataFrames necesarios para realizar los ejercicios de los 3 niveles.



id	product_name	price	colour	weight	warehouse_id
1	Direwolf Stannis	161,11	#7c7c7c	1	WH-4
10	Karstark Dorne	119,52	#f4f4f4	2,4	WH--5
100	south duel	40,43	#6d6d6d	3	WH--95
11	Karstark Dorne	49,7	#141414	2,7	WH--6
12	duel Direwolf	181,6	#a8a8a8	2,1	WH--7
13	palpatine chewbacca	139,59	#2b2b2b	1	WH--8
14	Direwolf	147,53	#c4c4c4	2	WH--9
15	Stannis warden	194,29	#dbdbdb	1,5	WH--10
16	the duel warden	180,91	#666666	3	WH--11
17	skywalker ewok sith	91,89	#7c7c7c	3,2	WH--12
18	Karstark warden	148,91	#c4c4c4	0,8	WH--13
19	dooku solo	60,33	#3f3f3f	0,6	WH--14
2	Tarly Stark	9,24	#919191	2	WH-3
20	warden Karstark	91,96	#b5b5b5	1,4	WH--15
21	duel Direwolf	96,9	#e2e2e2	1,2	WH--16
22	chewbacca mustafar	150,02	#fcfcfc	2,4	WH--17
23	riverlands north	169,96	#545454	2,7	WH--18
24	south duel tourney	48,99	#aaaaaa	2,1	WH--19
25	skywalker ewok	157,53	#2b2b2b	1	WH--20
26	Stark Karstark	53,01	#898989	2	WH--21
27	Stannis riverlands	172,93	#7a7a7a	1,5	WH--22
28	chewbacca mustafar	127,44	#efefef	3	WH--23
29	Tully maester Tarly	167,2	#111111	3,2	WH--24

Tras cargar los DataFrames he realizado lo siguiente:

- En la vista tabla en el df_transactions añadí una columna nueva “year” con una expresión DAX, para realizar algunos ejercicios especificados posteriormente en los que necesité dicha información

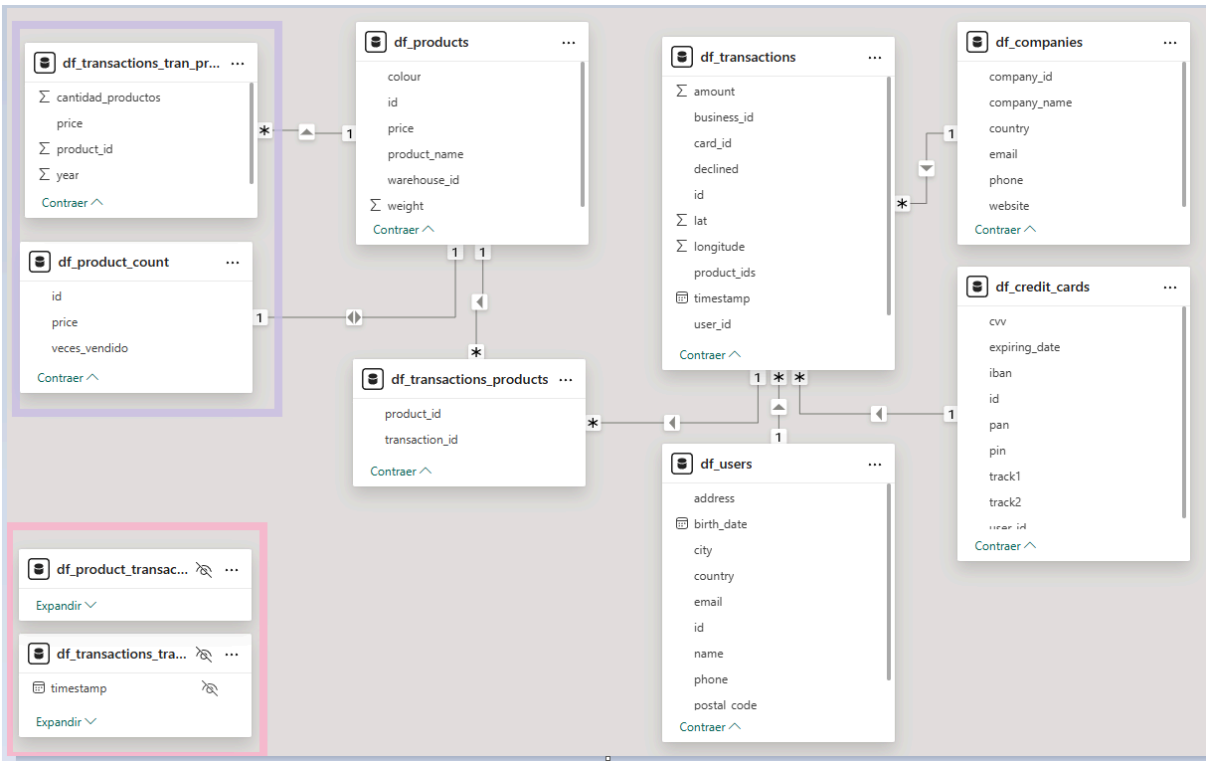
```
1 year = df_transactions[timestamp].[Año]
```

id	card_id	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude	year
0466A42E-47CF-8D24-FD01-C0B689713128	CcU-4219	b-2302	26/7/2021 7:29:18	49,53 €	False	47, 97, 43	170	-439695	-117525	2021
0A476ED9-0C13-1962-F87B-D3563924B539	CcU-4359	b-2302	26/2/2022 20:33:54	430,49 €	False	29, 41, 11	221	-564901	114801	2022
122DC333-E19F-D629-DCD8-9C54CF1EBB9A	CcU-4366	b-2302	9/6/2021 6:04:14	172,01 €	False	1, 67, 19	221	296372	-166173	2021
135267BA-2E7D-957C-C42C-6450A2B3ED54	CcU-4520	b-2302	29/12/2021 20:38:23	17,97 €	False	11, 71	210	206724	149732	2021

- En la vista tabla en el df_transactions cambie amount a formato moneda y le agregue el símbolo €

id	card_id	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude	year
0466A42E-47CF-8D24-FD01-C0B689713128	CcU-4219	b-2302	26/7/2021 7:29:18	49,53 €	False	47, 97, 43	170	-439695	-117525	2021
0A476ED9-0C13-1962-F87B-D3563924B539	CcU-4359	b-2302	26/2/2022 20:33:54	430,49 €	False	29, 41, 11	221	-564901	114801	2022
122DC333-E19F-D629-DCD8-9C54CF1EBB9A	CcU-4366	b-2302	9/6/2021 6:04:14	172,01 €	False	1, 67, 19	221	296372	-166173	2021
135267BA-2E7D-957C-C42C-6450A2B3ED54	CcU-4520	b-2302	29/12/2021 20:38:23	17,97 €	False	11, 71	210	206724	149732	2021
14CAE5B5-8FB1-3E4A-4C85-0EA4167534F4	CcU-4849	b-2302	31/12/2021 0:29:42	388,04 €	False	2, 13, 53, 31	189	-536202	930533	2021

- En la vista modelo he creado las respectivas relaciones entre los DataFrames



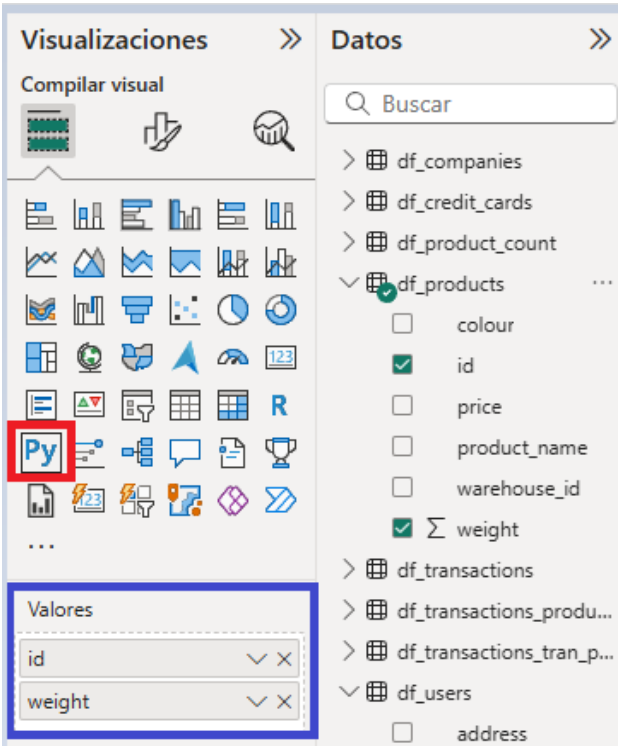
En la parte inferior izquierda de la imagen se encuentran 2 DataFrames ocultos y no relacionados con el modelo (color rosa) debido a que son el resultado de los merges que dan lugar a otros DataFrames que sí están relacionados con el modelo (color lila).

Una vez completados los pasos anteriores inicié con la realización de los ejercicios.

NIVELL 1

EXERCICI 1

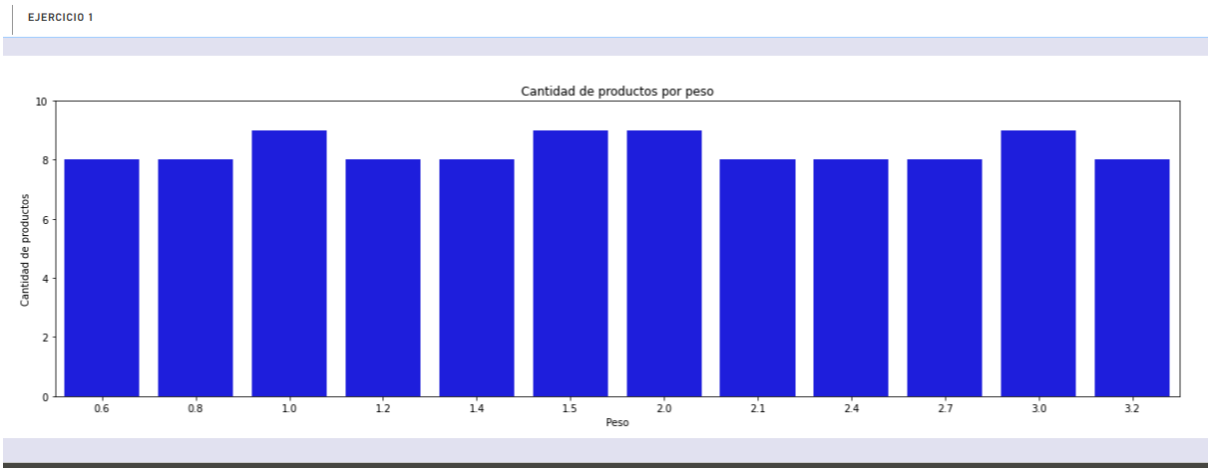
Para realizar este ejercicio seleccione en el panel de visualización el icono de PY (objeto visual de python - color rojo). Luego en la sección de valores arrastre los datos necesarios para la creación del gráfico, en este caso los campos id y weight de df_products (color azul).



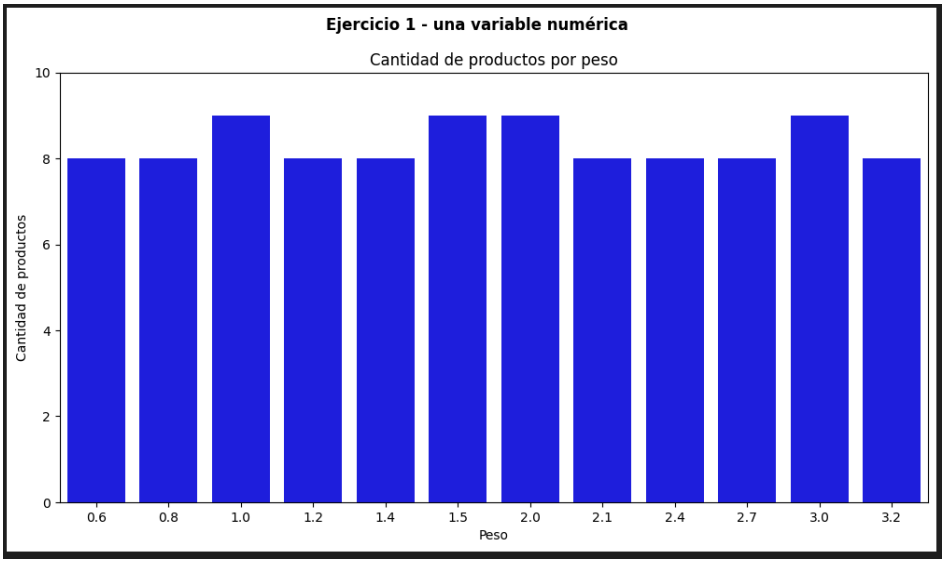
Posteriormente en el editor de script de python, ingresé el código empleado en el ejercicio 1 nivel 1 del S8.1. Es importante resaltar que en dicho código he reemplazado el nombre del DataFrame por dataset en todas sus referencias dentro del código.

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

sns.countplot(
    data=dataset,
    x='weight',
    color='blue'
)

plt.title("Cantidad de productos por peso")
plt.xlabel("Peso")
plt.ylabel("Cantidad de productos")
plt.ylim(0,10)

plt.show()
```

Al ejecutar el script obtuve la siguiente figura en PBI:



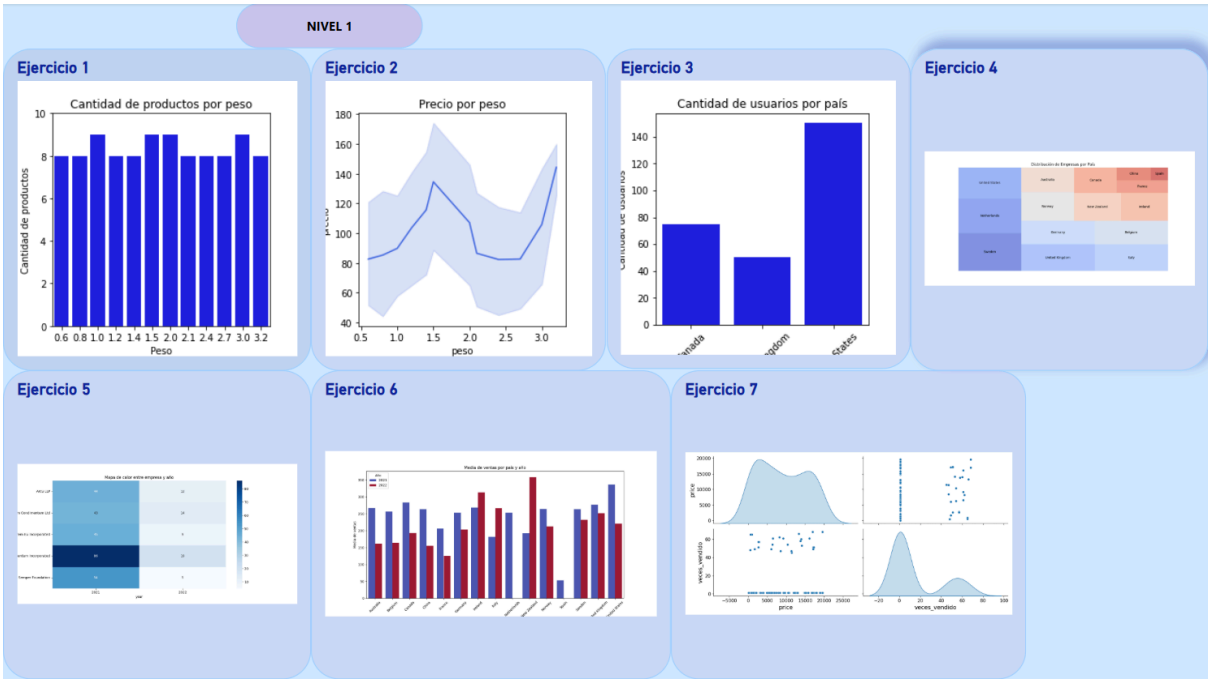
que es igual a la que había obtenido previamente en python:



Para la realización de los demás ejercicios de los 3 niveles aplique los mismos pasos descritos anteriormente, pero cabe resaltar los siguientes aspectos:

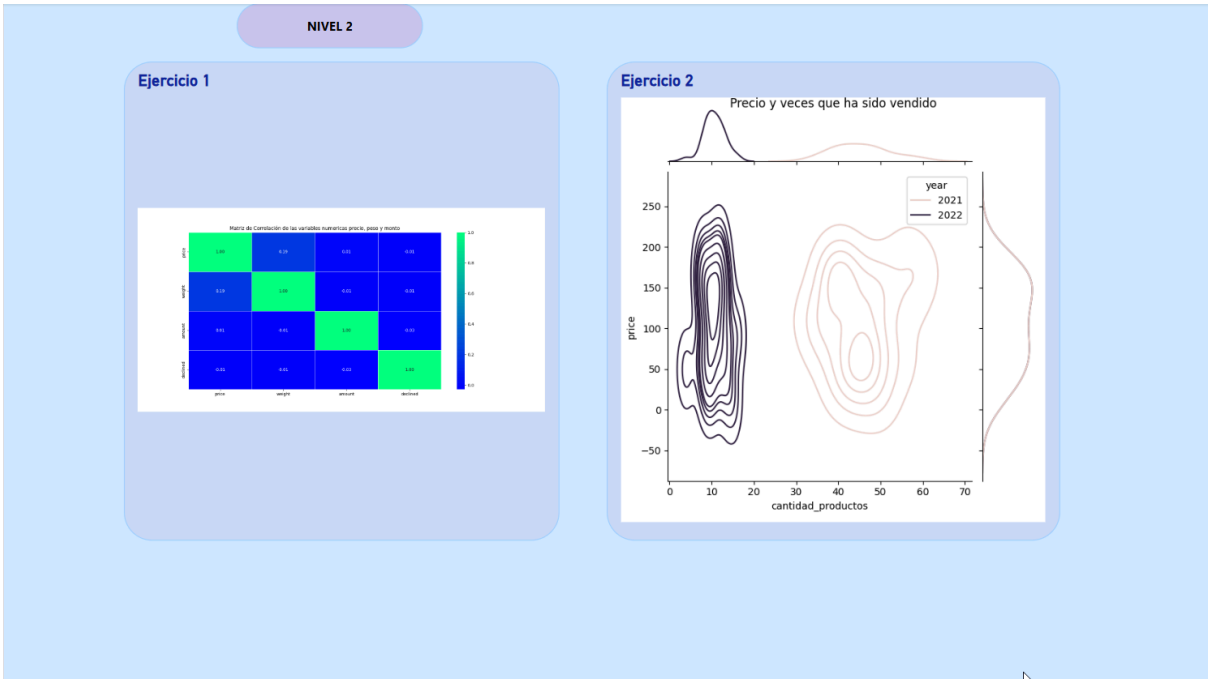
- Para realizar correctamente los ejercicios 5 y 6 del nivel 1 y el ejercicio 1 del nivel 3 utilice los datos de la nueva columna (year) creada y comentada previamente.
- Para los ejercicios 7 (nivel 1) y 2 (nivel 2) utilicé los los datos de los DataFrames creados en python y cargos en PBI en los ajustes previos a la realización de los ejercicios.
- En ciertos ejercicios tuve que incluir ajustes específicos en el editor de scripts de python dentro de PBI para garantizar que los gráficos se generan de manera adecuada. Sin embargo dichos casos fueron excepcionales y la mayoría de las visualizaciones pudieron reproducirse sin inconvenientes.

Visualizaciones de los ejercicios del 1 al 7



NIVELL 2

Visualizaciones de los ejercicios 1 y 2

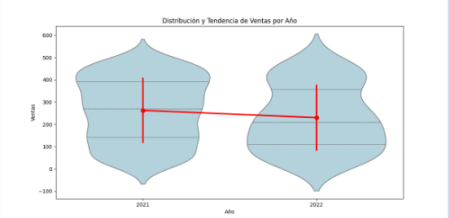


NIVELL 3

Visualizaciones de los ejercicios 1 y 2

NIVEL 3

Ejercicio 1



Ejercicio 2