Tasca S8.02. Power BI amb Python

Aquesta tasca consisteix en l'elaboració d'un informe de Power BI, aprofitant les capacitats analítiques de Python. S'utilitzaran els scripts de Python creats prèviament en la Tasca 1 per a generar visualitzacions personalitzades amb les biblioteques Seaborn i Matplotlib. Aquestes visualitzacions seran integrades en l'informe de Power BI per a oferir una comprensió més profunda de la capacitat del llenguatge de programació en l'eina Power BI.

Configuració Power BI i connexió amb script de Python

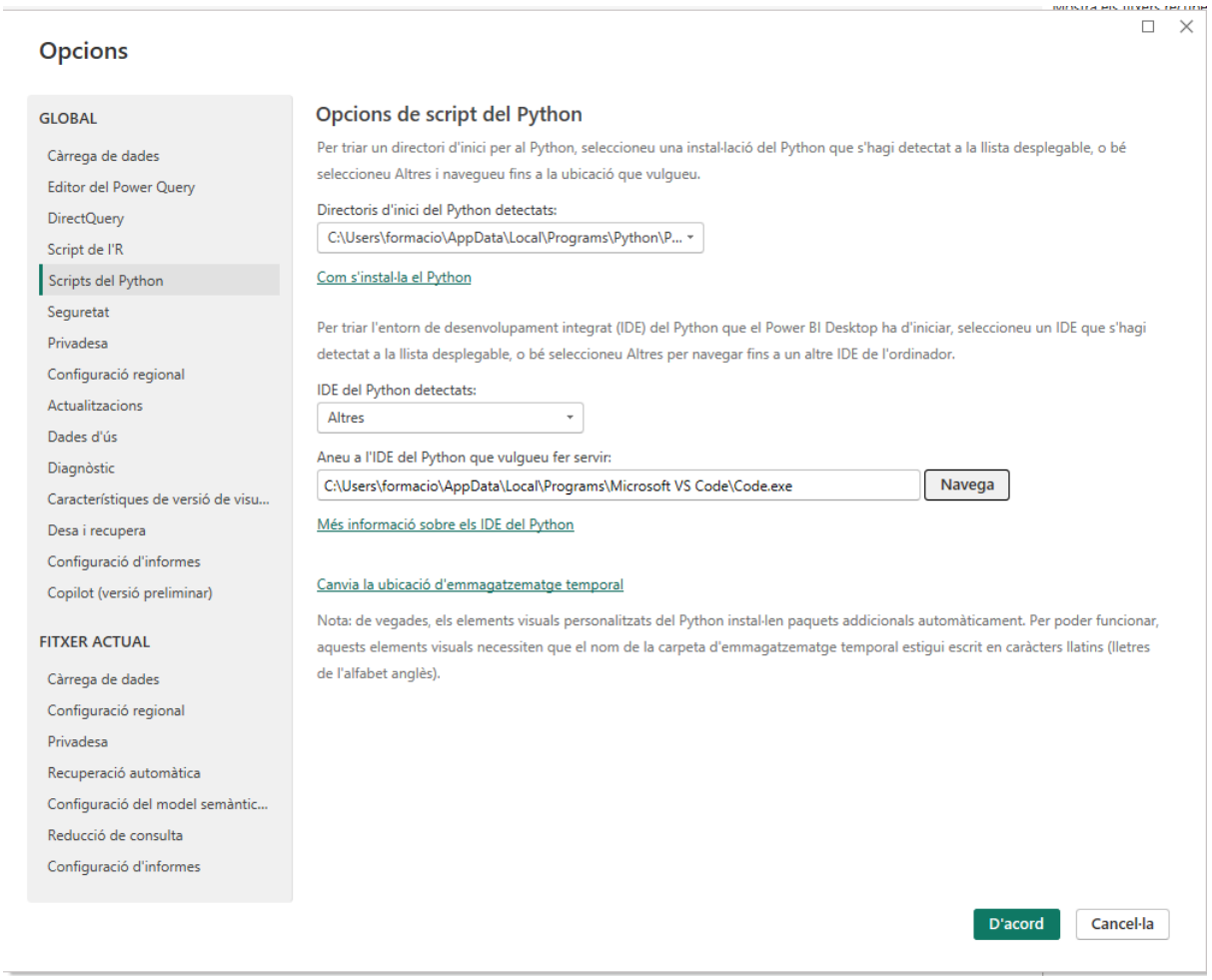
Nos aseguramos de tener las librerías pandas (para el manejo de dataframes), matplotlib y seaborn (para la creación de gráficos) en el equipo local.

En una instancia del terminal ejecuto el Python Launcher y le pido una lista de paquetes instalados mediante pip:

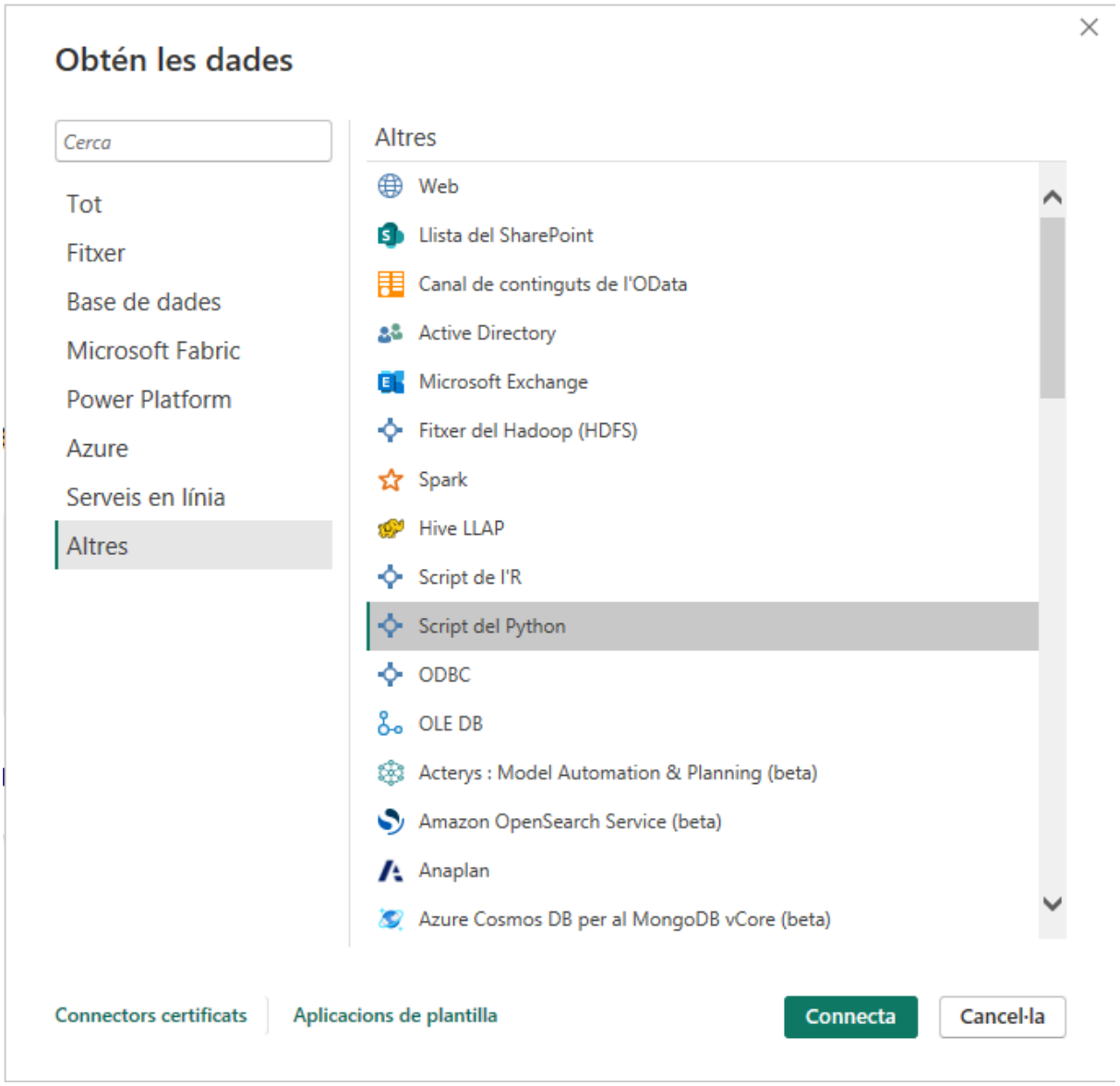
```
PS C:\Users\formacio\Documents\DataAnalyticsAlex> py --list
 -V:3.12 *        Python 3.12 (64-bit)
PS C:\Users\formacio\Documents\DataAnalyticsAlex> py -m pip list
Package                Version
---------------------- -----------
asttokens              2.4.1
certifi                2024.12.14
charset-normalizer     3.4.1
colorama               0.4.6
comm                   0.2.2
contourpy              1.3.0
cycler                 0.12.1
debugpy                1.8.5
decorator              5.1.1
et-xmlfile             1.1.0
executing              2.1.0
fontawesomefree        6.6.0
fonttools              4.53.1
greenlet               3.1.1
highlight-text         0.2
idna                   3.10
ipykernel              6.29.5
ipython                8.27.0
jedi                   0.19.1
jupyter_client         8.6.2
jupyter_core           5.7.2
kiwisolver             1.4.7
matplotlib             3.10.0
matplotlib-inline      0.1.7
mysql-connector-python 9.1.0
nest-asyncio           1.6.0
numpy                  2.1.1
openpyxl               3.1.5
packaging              24.1
panda                  0.3.1
pandas                 2.2.2
parso                  0.8.4
pillow                 10.4.0
```

```
pillow                  10.4.0
pip                     25.0
platformdirs            4.2.2
prompt_toolkit          3.0.47
psutil                  6.0.0
pure_eval               0.2.3
Pygments                2.18.0
pypalettes              0.1.5
pyparsing               3.1.4
python-dateutil         2.9.0.post0
pytz                    2024.1
pywaffle                1.1.1
pywin32                 306
pyzmq                   26.2.0
requests                2.32.3
seaborn                 0.13.2
setuptools              75.8.0
six                     1.16.0
SQLAlchemy              2.0.37
squarify                0.4.4
stack-data              0.6.3
tornado                 6.4.1
traitlets               5.14.3
typing_extensions       4.12.2
tzdata                  2024.1
urllib3                 2.3.0
wcwidth                 0.2.13
PS C:\Users\formacio\Documents\DataAnalyticsAlex>
```

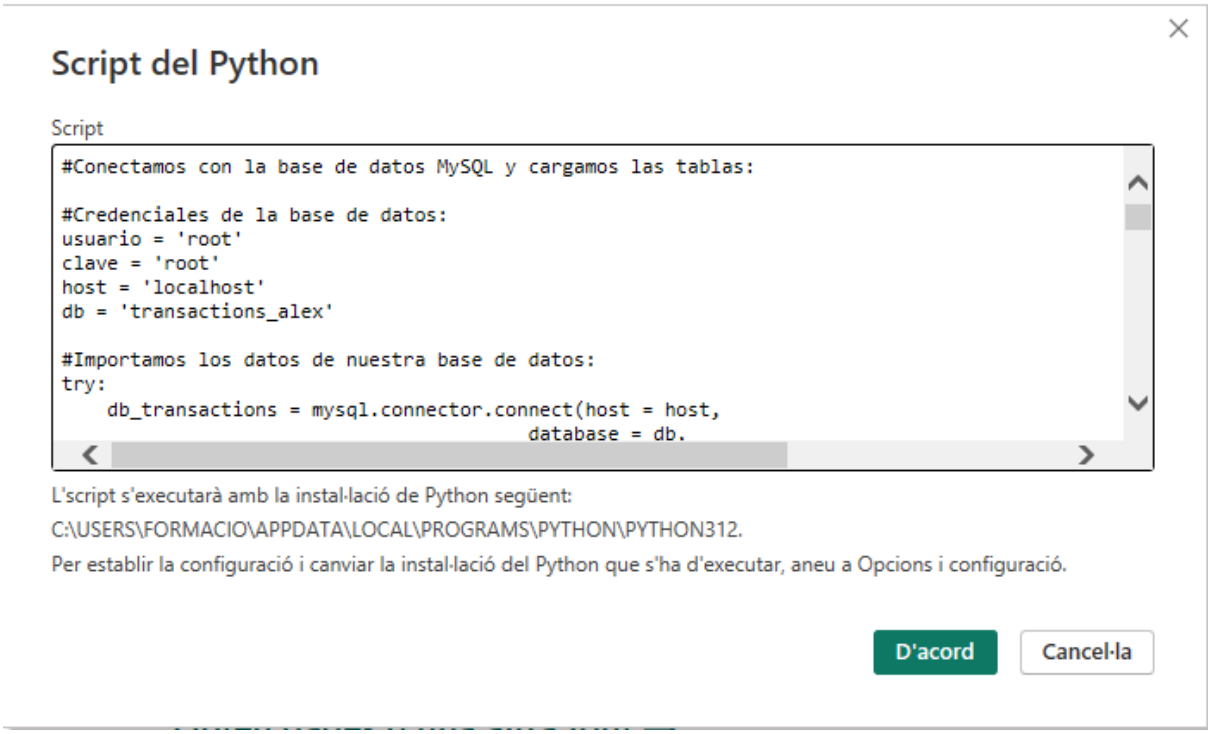
En Power BI, habilitamos la ejecución de scripts de Python:



Escogeremos la opción **Obtener datos / Script de Python**:



Y en el cuadro habilitado para el script cargaremos los dataframes que hemos creado en el script de la TascaS801, tanto los creados a partir de la base de datos **transaction_alex** que importamos a través de MySQL Connector para Python como los dataframes creados a partir de la función merge() de la librería pandas con las transformaciones pertinentes.



Script:

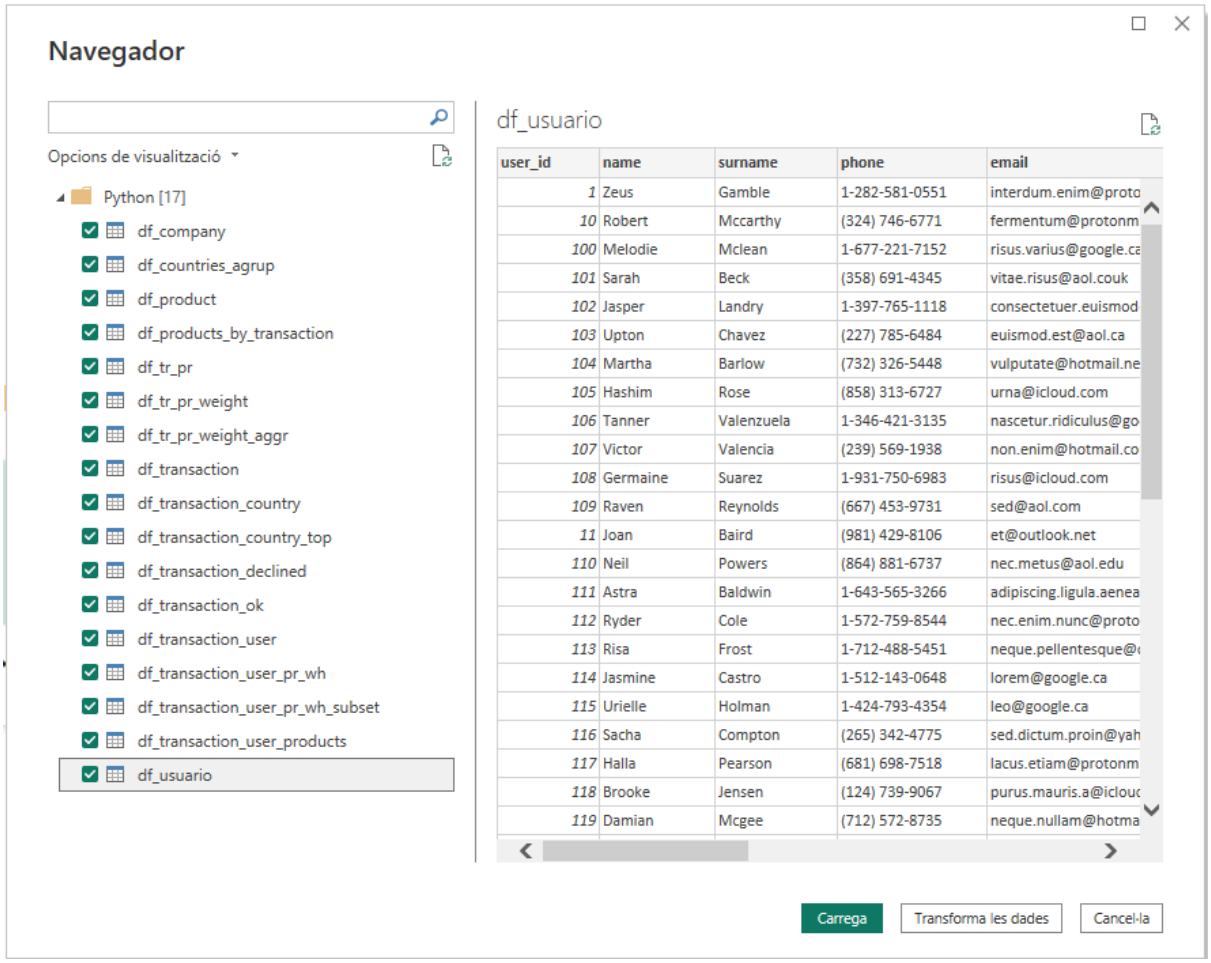
```
#Importamos las librerías necesarias para manejar los datos y crear las
gráficas:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Importamos las librerías necesarias para la conexión con la base de
datos:

import mysql.connector
from mysql.connector import Error

from sqlalchemy import create_engine

#Conectamos con la base de datos MySQL y cargamos las tablas:

#Credenciales de la base de datos:
usuario = 'root'
clave = 'root'
host = 'localhost'
db = 'transactions_alex'

#Importamos los datos de nuestra base de datos:
try:
    db_transactions = mysql.connector.connect(host = host,
                                              database = db,
                                              user = usuario,
                                              password = clave)
```

```
    #Nos aseguramos de que se realiza la conexión con MySQL Server y
con la base de datos:
    if db_transactions.is_connected():
        db_Info = db_transactions.get_server_info()
        print(f"Conectado con la version {db_Info} de MySQL Server")
        cursor = db_transactions.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print(f"Conexión establecida con la base de datos {record[0]}")

    #Recuperamos el nombre de las tablas de la base de datos y los
guardamos en una lista:
    query_tablas = """SHOW tables"""
    cursor.execute(query_tablas)
    db_tablas = cursor.fetchall()

    lista_tablas = []
    for tabla in db_tablas:
        lista_tablas.append(tabla[0])

    #Vamos a cargar la información de cada tabla en sendos dataframes y
crearemos un diccionario
    #cuyas claves serán los nombres de las tablas almacenadas en la
lista y sus valores, el dataframe correspondiente a cada tabla:
    conexion =
create_engine(f'mysql+mysqlconnector://{usuario}:{clave}@{host}/{db}')
    dataframe = {}

    for tabla in lista_tablas:
        query = f"SELECT * FROM {tabla}"
        dataframe[tabla] = pd.read_sql(query, conexion)

    #Comprobamos que hemos cargado correctamente el diccionario
    for tabla in lista_tablas:
        print(f'Tabla {tabla}: \n', dataframe[tabla].head(), '\n')

#Manejamos los posibles errores de sesión
except Error as e:
    print("Error en la conexión a MySQL:", e)

#Cerramos el cursor y la conexión
finally:
    if db_transactions.is_connected():
        cursor.close()
        db_transactions.close()
        print("Conexión a MySQL cerrada")

#Incorporamos los dataframes del script S801 a Power BI (y aprovechamos
para limpiar el campo amount de la tabla product)

df_product = dataframe['product']
df_product['price'] = df_product['price'].str.replace('$','')
df_product['price'] = df_product['price'].astype(float)

#Nivel 1
#Ejercicio 1
df_transaction = dataframe['transaction']
df_transaction_ok = df_transaction[df_transaction['declined'] == 0]
```

```
#Ejercicio 2
df_usuario = dataframe['user']
df_usuario['birth_date'] = pd.to_datetime(df_usuario['birth_date'])
df_usuario['age'] = (pd.Timestamp('now') -
df_usuario['birth_date']).dt.days // 365
df_usuario = df_usuario.rename(columns={'id': 'user_id'})
df_transaction_user = pd.merge(df_transaction, df_usuario,
on='user_id', how='inner')

#Ejercicio 3
df_company = dataframe['company']

#Ejercicio 4
df_transaction['period'] =
df_transaction['timestamp'].dt.to_period('M')
transacciones_mensuales =
df_transaction['amount'].groupby([df_transaction['period']]).sum()
ventas_mensuales =
df_transaction_ok['amount'].groupby([df_transaction['period']]).sum()

#Ejercicios 5 y 6
df_company.rename(columns={'id': 'company_id'}, inplace=True)
df_transaction_country = pd.merge(df_transaction, df_company,
on='company_id', how='inner')
df_countries_agrup = df_transaction_country.groupby('country',
as_index=False).size().sort_values('size', ascending=False)
lista_countries_top = df_countries_agrup[df_countries_agrup['size'] >
20]['country']
df_transaction_country_top =
df_transaction_country[df_transaction_country['country'].isin(lista_cou
ntries_top)]

#Ejercicio 7
df_tr_pr = dataframe['transaction_products']
df_tr_pr.rename(columns={'transaction_id': 'id'}, inplace=True)
df_products_by_transaction = df_tr_pr.groupby('id',
as_index=False).count()
df_products_by_transaction.rename(columns={'product_id':'prods_x_trans'
}, inplace=True)
df_transaction_user_products = pd.merge(df_transaction_user,
df_products_by_transaction, on='id', how='inner')

#Nivel 2
#Ejercicio 1
df_product.rename(columns={'id': 'product_id'}, inplace=True)
df_tr_pr_weight = pd.merge(df_tr_pr, df_product, on='product_id',
how='left')
group_df_tr_pr_weight = df_tr_pr_weight.groupby('id')
df_tr_pr_weight_aggr = df_tr_pr_weight.pivot_table(index='id',
values=['price', 'weight'], aggfunc=['sum'])
df_transaction_user_pr_wh = pd.merge(df_transaction_user_products,
df_tr_pr_weight_aggr['sum'], on='id', how='left')
df_transaction_user_pr_wh_subset = df_transaction_user_pr_wh[['amount',
'age', 'prods_x_trans', 'price', 'weight']]

#Nivel 3
#Ejercicio 2
```

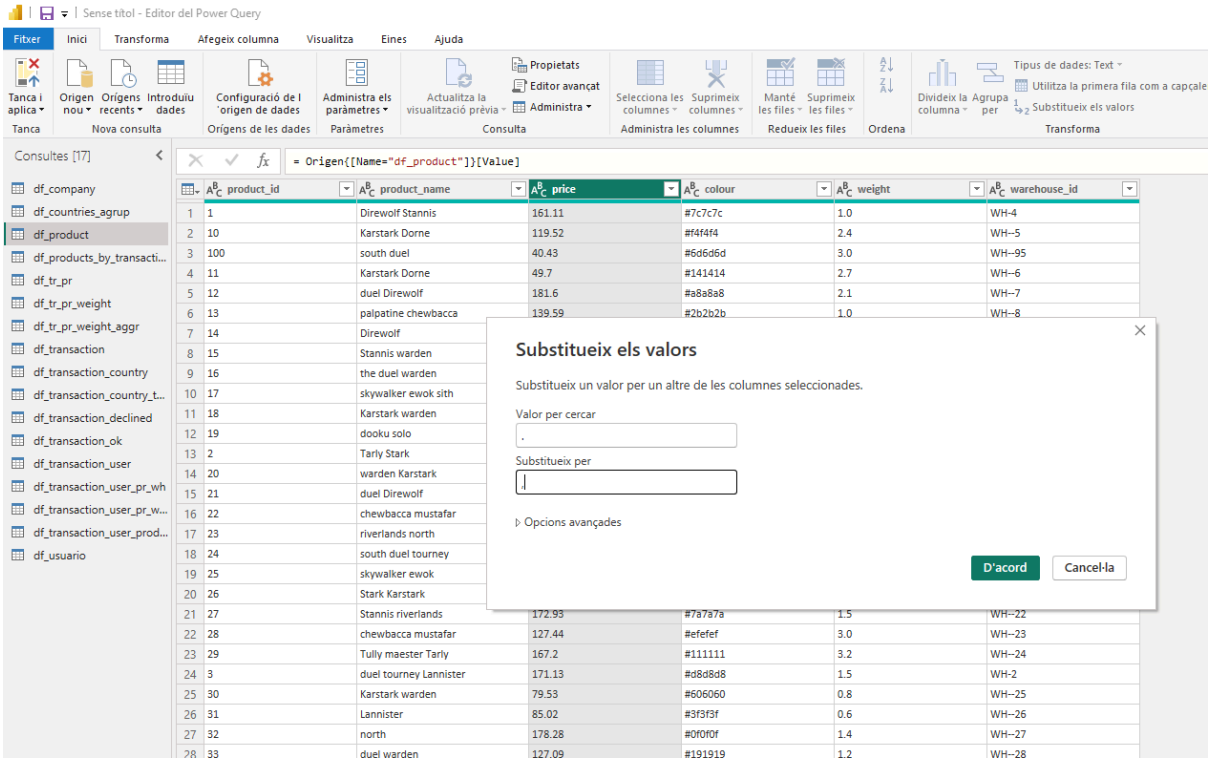
```
df_transaction_user_pr_wh['period'] =
df_transaction_user_pr_wh['timestamp'].dt.to_period('M')
df_transaction_user_pr_wh['month'] =
df_transaction_user_pr_wh['timestamp'].dt.month
df_transaction_user_pr_wh['year'] =
df_transaction_user_pr_wh['timestamp'].dt.year
```

Una vez cargados los dataframes, pasamos a revisar los datos y realizar las transformaciones pertinentes:



Las columnas **price** y **weight** de los dataframes **df_product, df_tr_pr_weight** aparecen sin el separador de decimales, así como la columna **amount** en los dataframes **df_transaction, df_transaction_country, df_transaction_country_top, df_transaction_ok, df_transaction_user, df_transaction_user_pr_wh** y **df_transaction_user_pr_wh_subset**. En estos dos últimos dataframes, fruto de la unión con la tabla product, vemos que el resultado de la agregación multiplica los resultados en el script, a causa de la desaparición del separador de decimales.

Procedemos a transformar estos datos antes de cargar los dataframes. Por ejemplo, en el caso de la columna **price**, eliminamos la transformación automática de texto a número decimal, sustituimos el signo '.' por ',' y transformamos a decimal fijo. También eliminaremos la conversión de algunos id que Power BI ha transformado de string a número entero.



Nivell 1

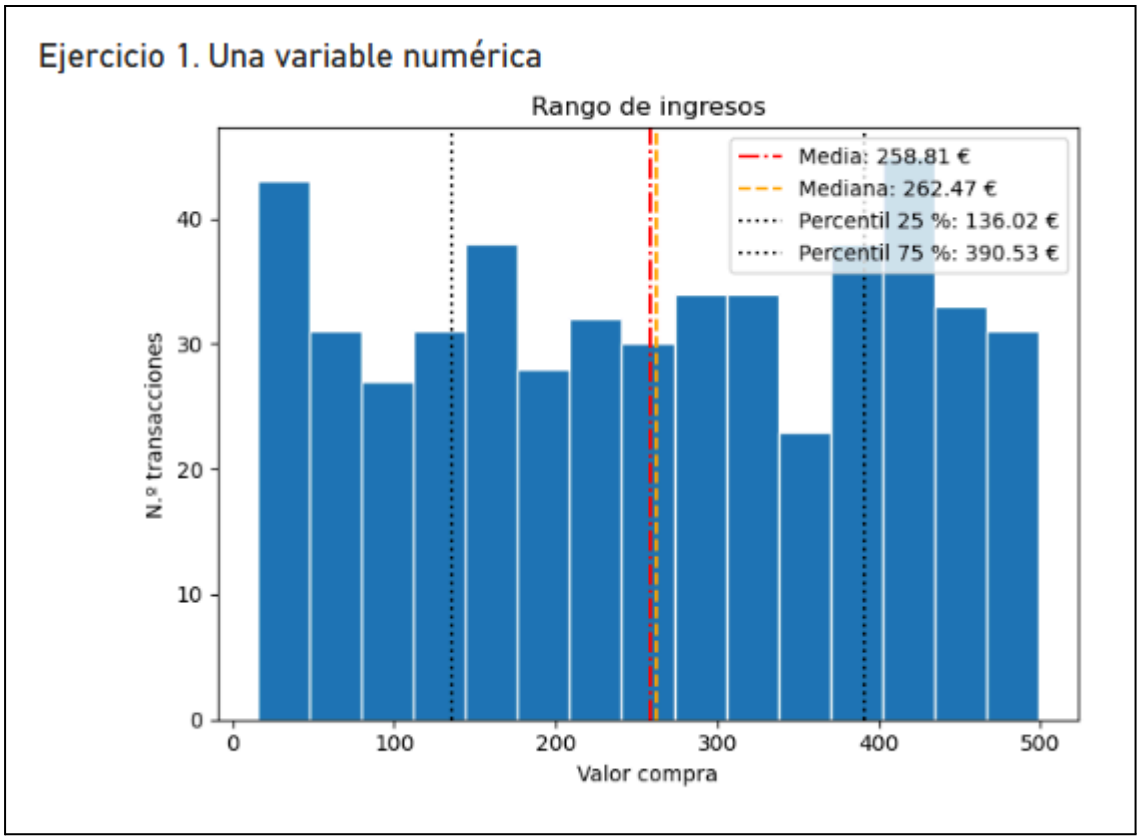
Els 7 exercicis del nivell 1 de la tasca 01

En las representaciones de las gráficas de los tres niveles seguiremos el mismo mecanismo: escogemos la herramienta **Elemento visual de Python**, seleccionamos en la columna **Datos** el dataframe que contiene la(s) columna(s) con los datos que queremos graficar, los añadimos al campo **Datos** de la pestaña **Compilación** y, a continuación, copiamos y adaptamos el script correspondiente del archivo **Tasca S801.ipynb** en el recuadro inferior para generar el gráfico en Power BI.

Los scripts adaptados se pueden consultar en el archivo **Tasca S802.ipynb**

Nota: como aprovechaba la característica de Visual Studio Code de interpretar los objetos de matplotlib y seaborn, a los scripts les añado las instrucciones **plt.tight_layout()** para evitar que la gráfica aparezca cortada en el contenedor de Power BI, y **plt.show()** para poder representarlos en Power BI.

Ejercicio 1. Gráfica con una variable numérica



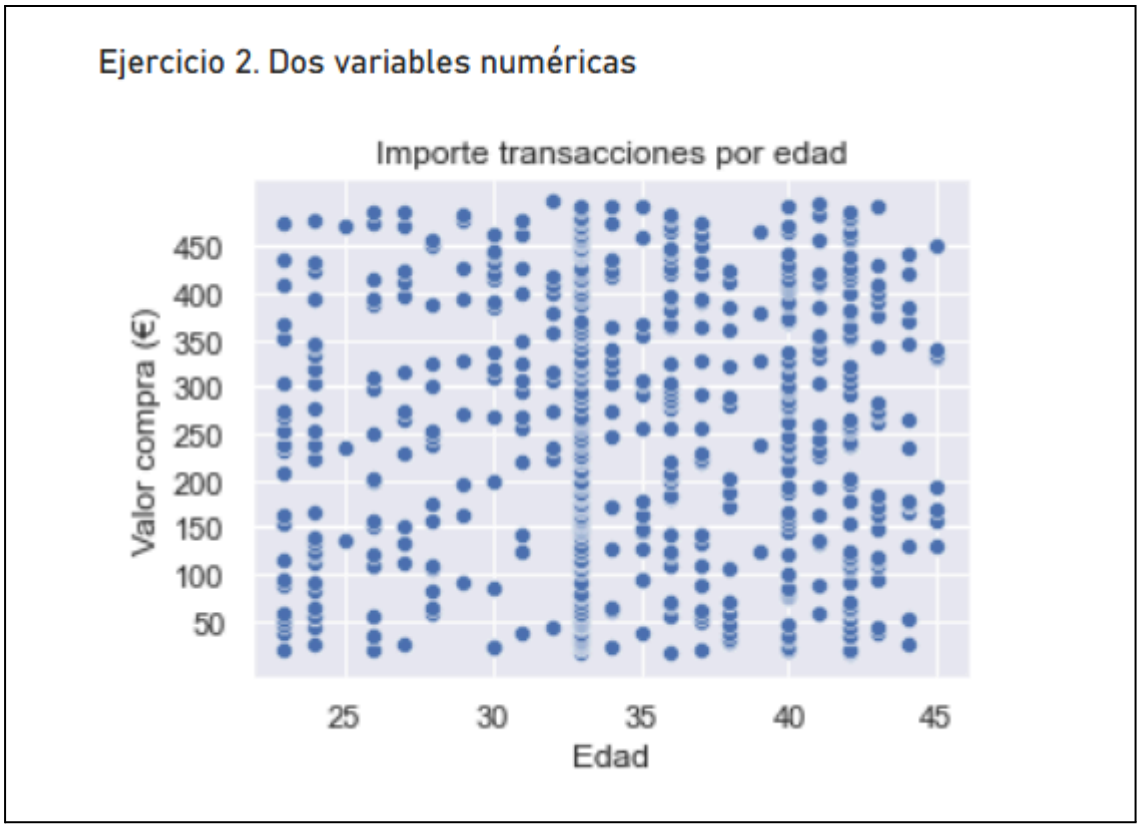
Script:

```
# El codi següent per crear un marc de dades i suprimir files
duplicades sempre s'executa i funciona com a preàmbul de l'script:

# dataset = pandas.DataFrame(amount)
# dataset = dataset.drop_duplicates()

# Enganxeu o escriviu el codi de l'script aquí:

df_transaction_ok = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Vamos a sacar algunas estadísticas sobre los ingresos y a graficarlas:
amount = df_transaction_ok['amount']

media = amount.mean()
mediana = amount.median()
percentil_25 = np.percentile(amount,25)
percentil_75 = np.percentile(amount,75)

fig, ax = plt.subplots()

ax.hist(dataset, bins=15, edgecolor="white")
ax.axvline(media, color="red", ls="-.", label=f"Media: {round(media,2)}
€")
```

```
ax.axvline(mediana, color="orange", ls="--", label=f"Mediana:
{round(mediana,2)} €")
ax.axvline(percentil_25, color="black", ls=":", label=f"Percentil 25 %:
{round(percentil_25,2)} €")
ax.axvline(percentil_75, color="black", ls=":", label=f"Percentil 75 %:
{round(percentil_75,2)} €")
ax.set_title('Rango de ingresos')
ax.set_xlabel('Valor compra')
ax.set_ylabel('N.° transacciones')
plt.legend(loc='upper right')
plt.tight_layout()

plt.show()
```

En esta visualización he sobreimpresionado la leyenda sobre el gráfico para mejorar la escala de la gráfica, puesto que la leyenda, usando los parámetros del ejercicio S8.01 sobresalía del marco creado por Power BI.

Ejercicio 2. Gráfica con dos variables numéricas

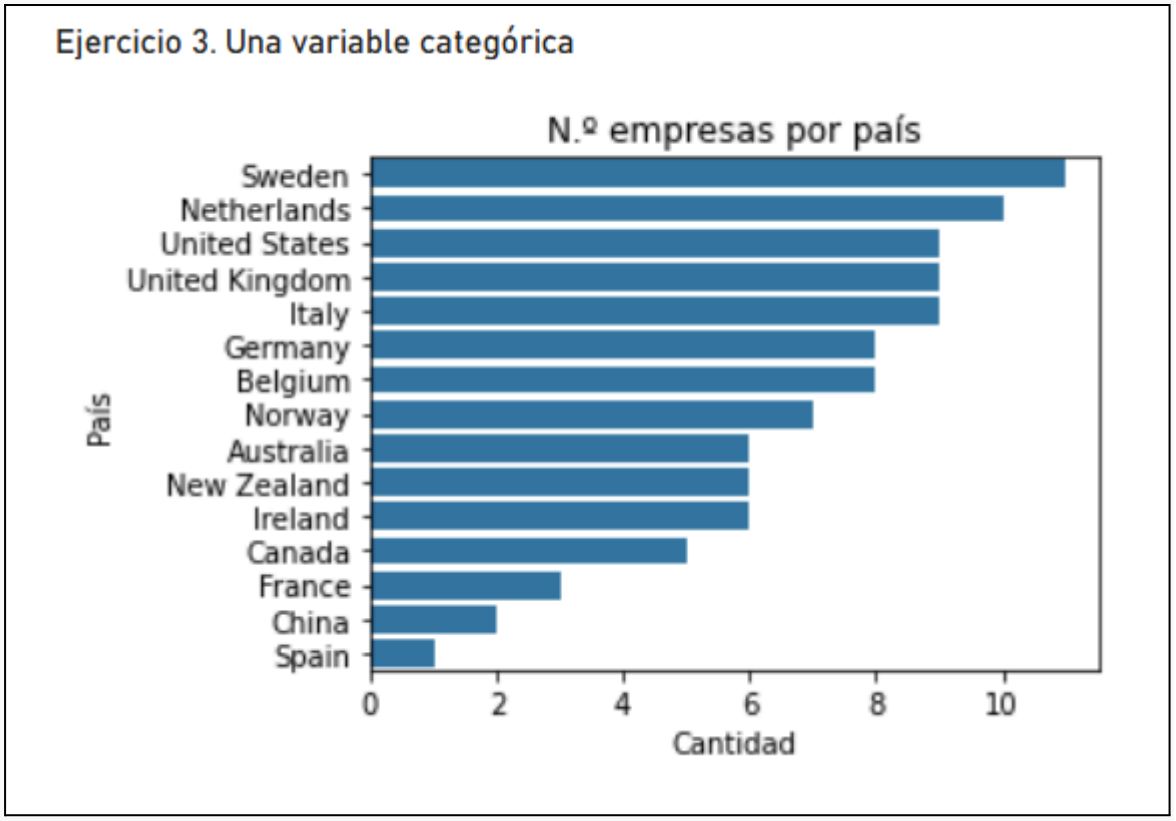


Script:

```
# El codi següent per crear un marc de dades i suprimir files
duplicades sempre s'executa i funciona com a preàmbul de l'script:

# dataset = pandas.DataFrame(amount, age)
# dataset = dataset.drop_duplicates()

# Enganxeu o escriviu el codi de l'script aquí:

df_transaction_user = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_theme(style="darkgrid")
sns.scatterplot(y=dataset['amount'], x=dataset['age'])
plt.xlabel('Edad')
plt.ylabel('Valor compra (€)')
plt.title('Importe transacciones por edad')
plt.yticks(range(50,500,50))
plt.tight_layout()

plt.show()
```

He mantenido los puntos correspondientes al usuario Hedwig Gilbert. Aun así, tal como señalabas en la corrección, efectivamente se ve una mayor concentración de compras entre los usuarios comprendidos entre los 30 y los 42 años. Identificado este rango, se pueden tomar acciones encaminadas a atraer a este sector de población.

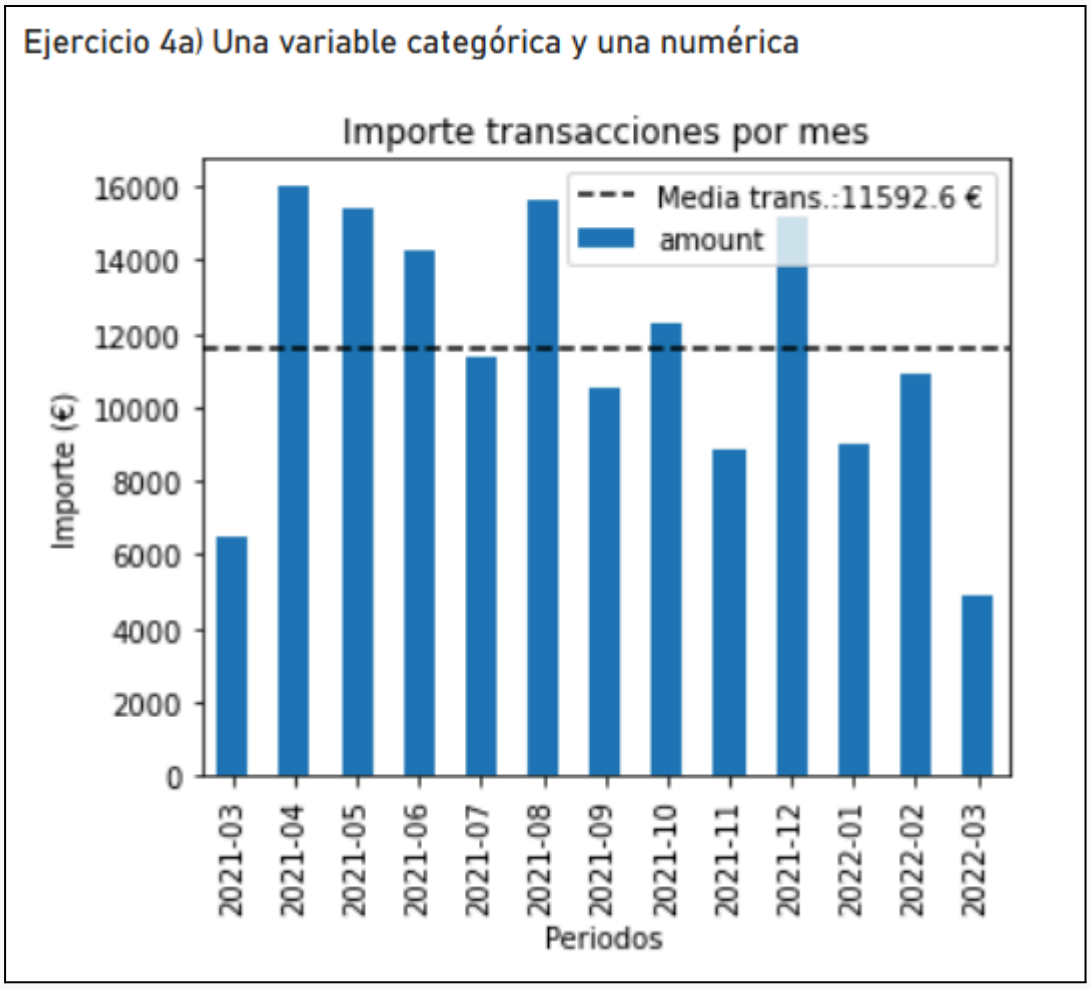
Ejercicio 3. Gráfica con una variables categórica



Script:

```
# El codi següent per crear un marc de dades i suprimir files
duplicades sempre s'executa i funciona com a preàmbul de l'script:

# dataset = pandas.DataFrame(company_id)
# dataset = dataset.drop_duplicates()

# Enganxeu o escriviu el codi de l'script aquí:

df_company = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.countplot(data=dataset, y='country', order =
dataset['country'].value_counts().index)

plt.title('N.º empresas por país')
plt.xlabel('Cantidad')
plt.ylabel('País')
plt.tight_layout()

plt.show()
```

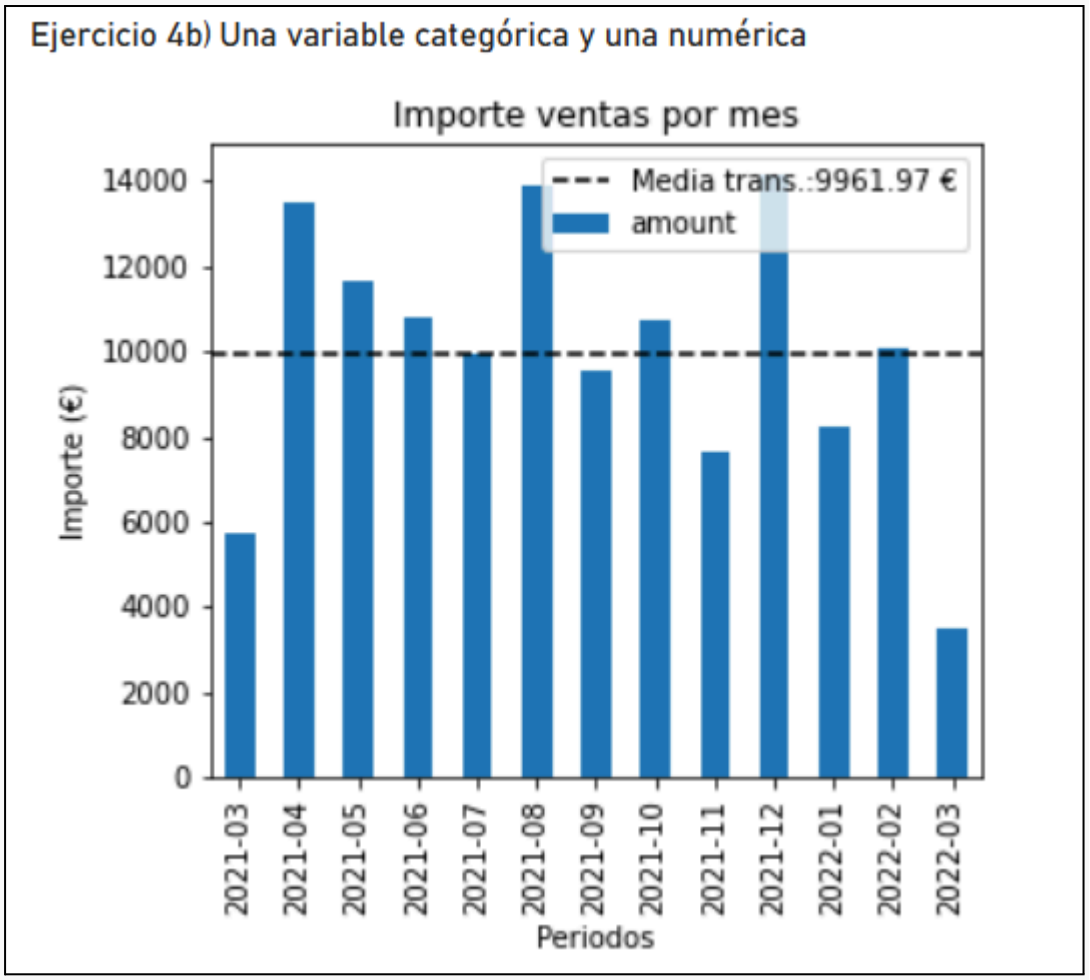
Ejercicio 4. Gráfica con una variable categórica y una numérica



Script:

```
# El código siguiente, que crea un dataframe y quita las filas
duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(amount, period, declined)
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:

df_transaction = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

transacciones_mensuales =
dataset['amount'].groupby([dataset['period']]).sum()

#Graficamos la evolución de los importes totales de las transacciones
por periodo:
media_transacciones = transacciones_mensuales.mean()

ax = transacciones_mensuales.plot(kind='bar')
```

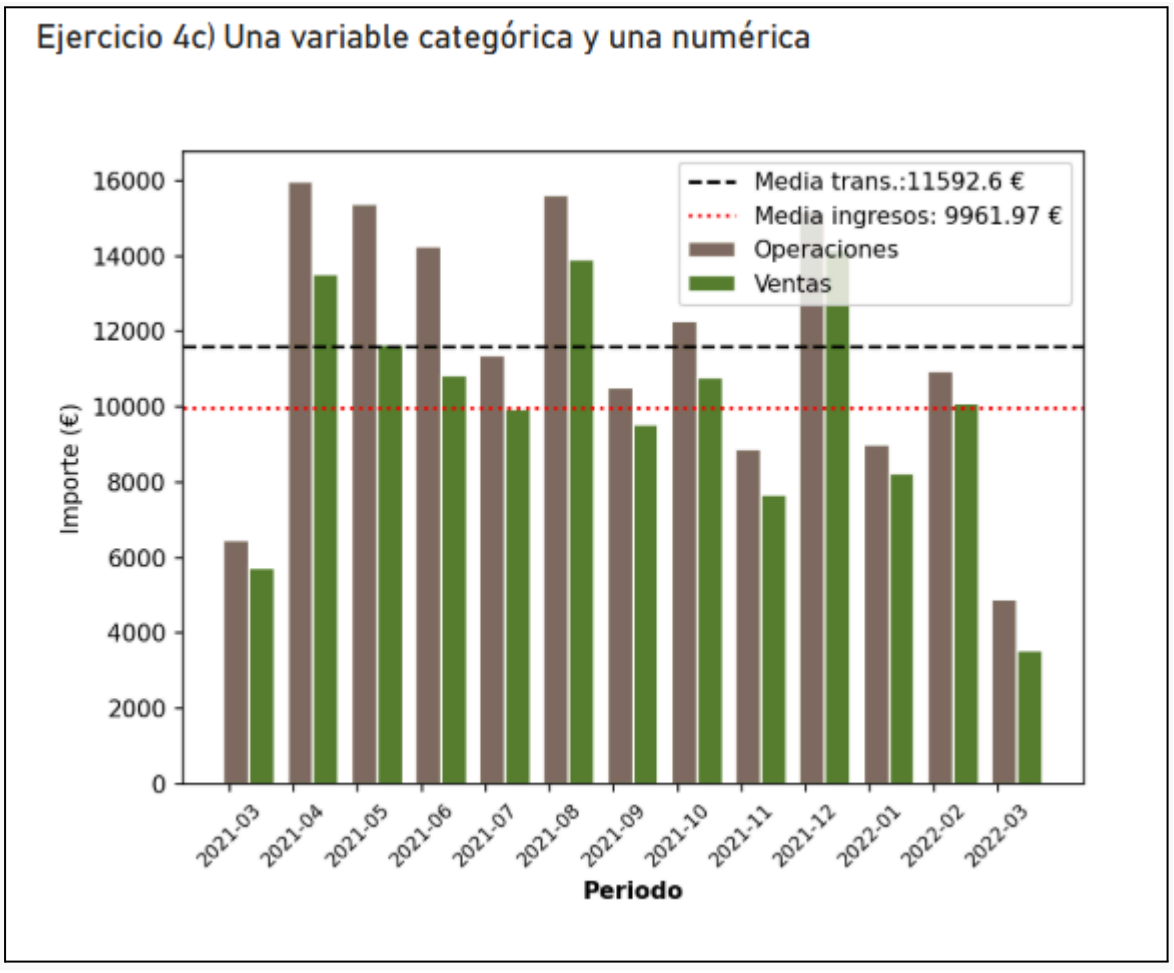
```
ax.axhline(y=media_transacciones, color='black', linestyle='--',
label=f'Media trans.:{round(media_transacciones,2)} €')
ax.set_title('Importe transacciones por mes')
ax.set_ylabel('Importe (€)')
ax.set_xlabel('Periodos')
ax.legend(loc='upper right')
plt.tight_layout()

plt.show()
```



Script:

```
# El código siguiente, que crea un dataframe y quita las filas
duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(amount, period, declined)
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:

df_transaction = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
ventas_mensuales =
dataset[dataset['declined']==0]['amount'].groupby([dataset['period']]).
sum()

#Graficamos la evolución de los importes totales de las transacciones
por periodo:
media_transacciones = ventas_mensuales.mean()

ax = ventas_mensuales.plot(kind='bar')
ax.axhline(y=media_transacciones, color='black', linestyle='--',
label=f'Media trans.:{round(media_transacciones,2)} €')
ax.set_title('Importe ventas por mes')
ax.set_ylabel('Importe (€)')
ax.set_xlabel('Periodos')
ax.legend(loc='upper right')
plt.tight_layout()

plt.show()
```



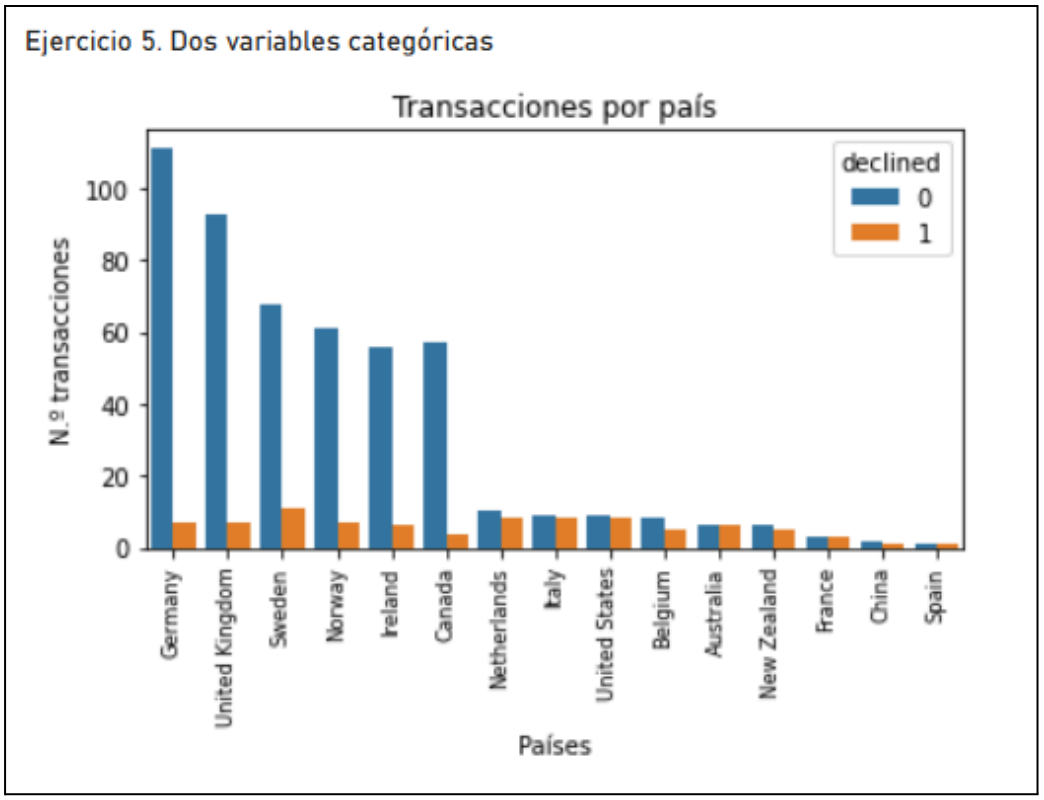
Script:

```
# El código siguiente, que crea un dataframe y quita las filas
duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(amount, period, declined)
```

```
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:

df_transaction = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Recuperamos las dos series de pandas con la suma de transacciones y
ventas:
transacciones_mensuales =
dataset['amount'].groupby([dataset['period']]).sum()
ventas_mensuales =
dataset[dataset['declined']==0]['amount'].groupby([dataset['period']]).
sum()

#Graficamos la evolución de los importes totales de las transacciones
por periodo:
media_transacciones = transacciones_mensuales.mean()
media_ventas = ventas_mensuales.mean()

# Datos
barWidth = 0.4
bars1 = transacciones_mensuales.values
bars2 = ventas_mensuales.values


# Posición de las barras
r = np.arange(len(bars1))
r2 = r + barWidth

# Gráfica
fig, ax = plt.subplots(dpi=150)
ax.bar(r, bars1, color='#7f6d5f', width=barWidth, edgecolor='white',
label='Operaciones')
ax.bar(r2, bars2, color='#557f2d', width=barWidth, edgecolor='white',
label='Ventas')

# Xticks
ax.set_xlabel('Periodo', fontweight='bold')
ax.set_ylabel('Importe (€)')
ax.set_xticks(r + barWidth -0.5)
ax.set_xticklabels(transacciones_mensuales.index)
ax.tick_params(axis='x', labelrotation=45, labelsize='small')

# Medias
ax.axhline(y=media_transacciones, color='black', linestyle='--',
label=f'Media trans.:{round(media_transacciones,2)} €')
ax.axhline(y=media_ventas, color='red', linestyle=':', label=f'Media
ingresos: {round(media_ventas,2)} €')

# Legend
ax.legend(ncol=1, loc='upper right')
plt.tight_layout()

plt.show()
```

Igual que en el ejercicio 1, he cambiado los parametros de la leyenda para sobreimpresionarlos encima de la gráfica y que no se perdiese más allá del marco.

Ejercicio 5. Gráfica con dos variables categóricas

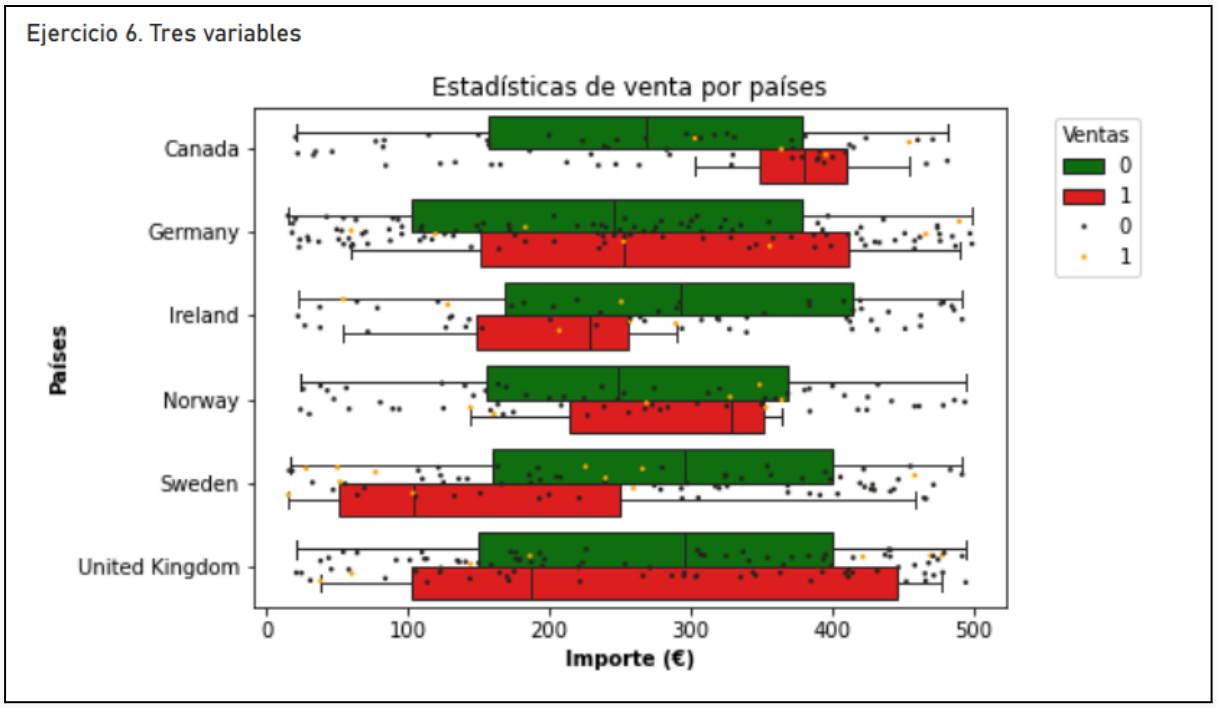


Script:

```
# El código siguiente, que crea un dataframe y quita las filas
duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(undefined, undefined.1, undefined.2,
undefined.3, undefined.4)
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:

df_transaction_country = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df_countries_agrup = df_transaction_country.groupby('country',
as_index=False).size().sort_values('size', ascending=False)


sns.countplot(data=dataset, x=dataset['country'], hue='declined',
stat='count', order=df_countries_agrup['country'])

plt.title('Transacciones por país')
plt.ylabel('N.° transacciones')
plt.xlabel('Países')
plt.xticks(size = 'small', rotation=90)
plt.tight_layout()
```

```
plt.show()
```

Ejercicio 6. Gráfica con tres variables

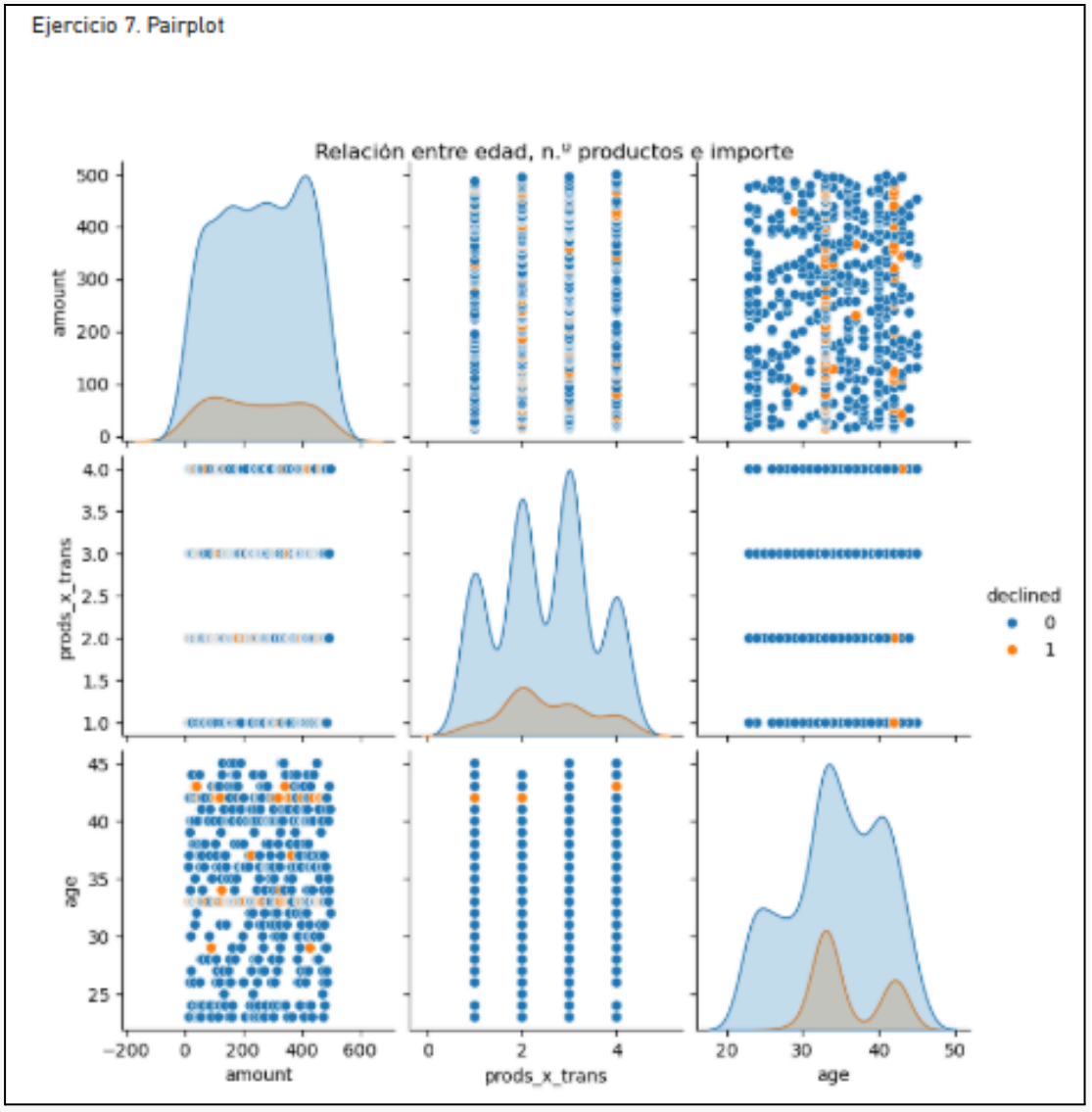


Script:

```
# El código siguiente, que crea un dataframe y quita las filas
duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(undefined, undefined.1, undefined.2)
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:

df_transaction_country_top = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Graficamos un boxplot para analizar las estadísticas por país y tipo
de operación. Por encima, marcamos las operaciones por país para tener
una referencia visual de la cantidad de operaciones de cada país:
custom_palette = {0: "green", 1: "red"}
sns.boxplot(y=dataset['country'], x=dataset['amount'],
hue=dataset['declined'], palette=custom_palette)

# Añadimos un stripplot:
sns.stripplot(y='country', x='amount', hue='declined', data=dataset,
palette="dark:orange", jitter=0.2, size=2.5)

plt.title("Estadísticas de venta por países", loc="center")
plt.ylabel('Países', fontweight='bold')
plt.xlabel('Importe (€)', fontweight='bold')
```

```
plt.legend(bbox_to_anchor=(1.05, 1), ncol=1, loc='upper left',
title='Ventas')
plt.tight_layout()

plt.show()
```

Ejercicio 7. Graficar un pairplot



Script:

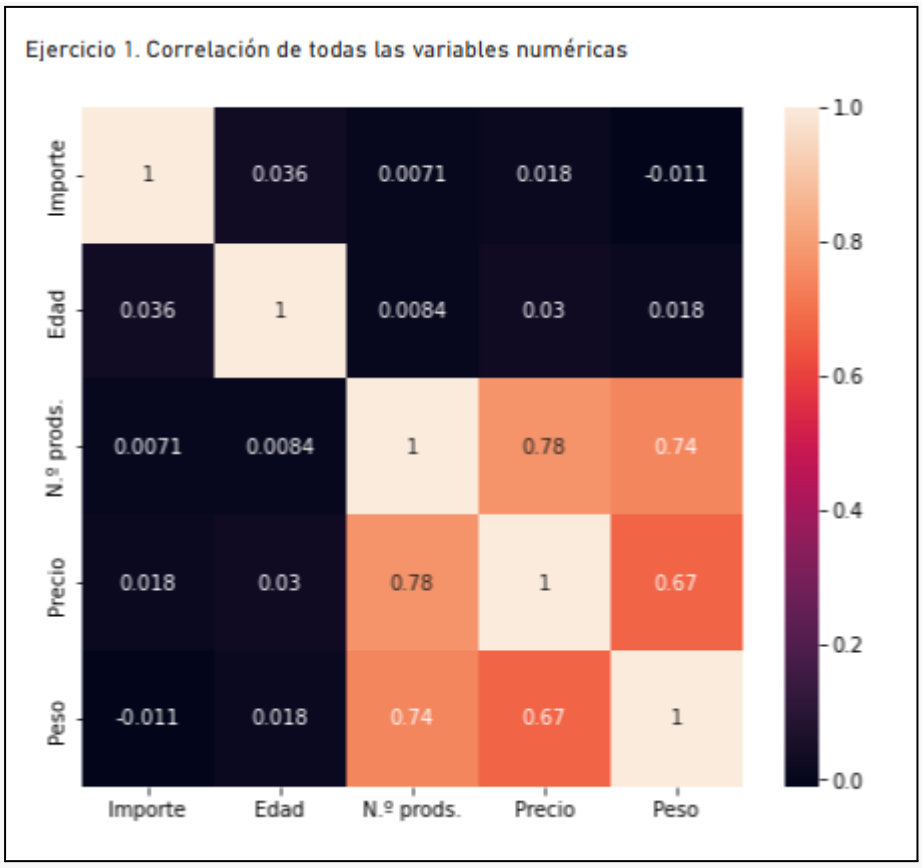
```
# El código siguiente, que crea un dataframe y quita las filas
duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(undefined, undefined.1, undefined.2)
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:

df_transaction_country_top = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Graficamos un boxplot para analizar las estadísticas por país y tipo
de operación. Por encima, marcamos las operaciones por país para tener
una referencia visual de la cantidad de operaciones de cada país:
custom_palette = {0: "green", 1: "red"}
sns.boxplot(y=dataset['country'], x=dataset['amount'],
hue=dataset['declined'], palette=custom_palette)

# Añadimos un stripplot:
# sns.stripplot(y='country', x='amount', hue='declined', data=dataset,
palette="dark:orange", jitter=0.2, size=2.5)

plt.title("Estadísticas de venta por países", loc="center")
plt.ylabel('Países', fontweight='bold')
plt.xlabel('Importe (€)', fontweight='bold')

plt.legend(bbox_to_anchor=(1.05, 1), ncol=1, loc='upper left',
title='Ventas')

plt.show()
```

Nivell 2

Els 2 exercicis del nivell 2 de la tasca 01

Ejercicio 1. Correlación de todas las variables numéricas

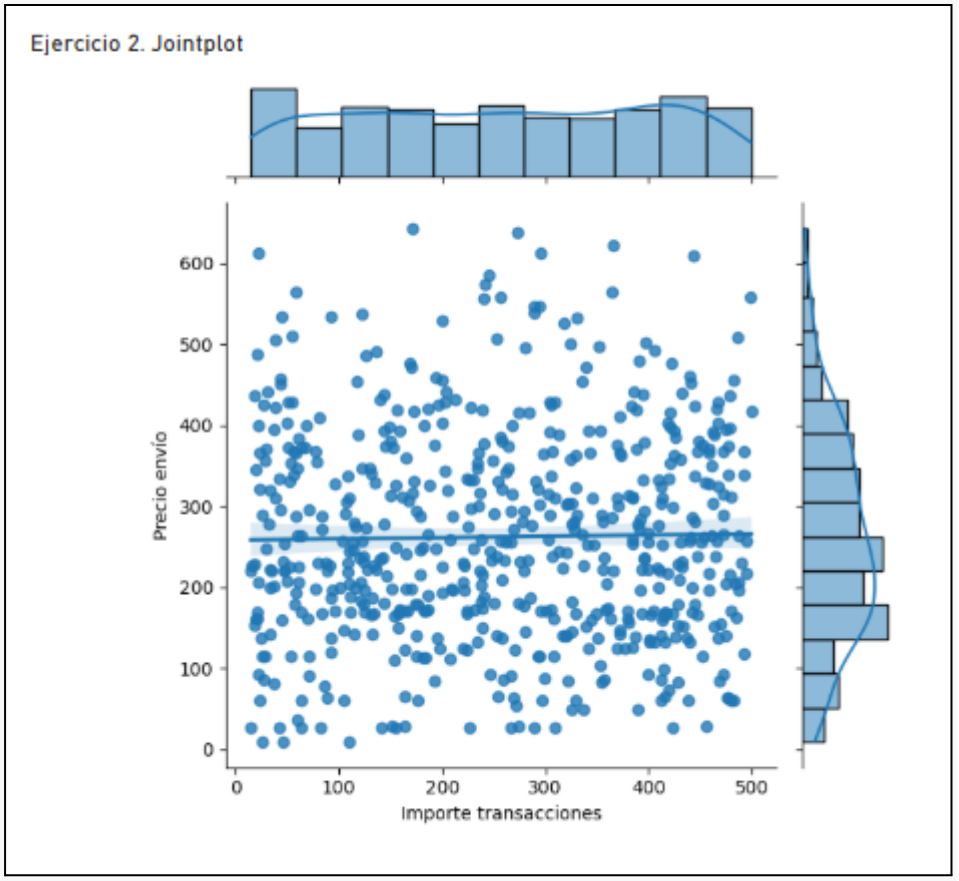


Script:

```
# El código siguiente, que crea un dataframe y quita las filas
duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(undefined, undefined.1, undefined.2,
undefined.3, undefined.4)
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:

df_transaction_user_pr_wh_subset = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

g = sns.heatmap(dataset[['amount','age','prods_x_trans','price',
'weight']].corr(), annot=True)
g.set_xticklabels(['Importe', 'Edad', 'N.° prods.', 'Precio', 'Peso'])
g.set_yticklabels(['Importe', 'Edad', 'N.° prods.', 'Precio', 'Peso'])
plt.tight_layout()
```

```
plt.show()
```

Ejercicio 2. Implementa un joinplot



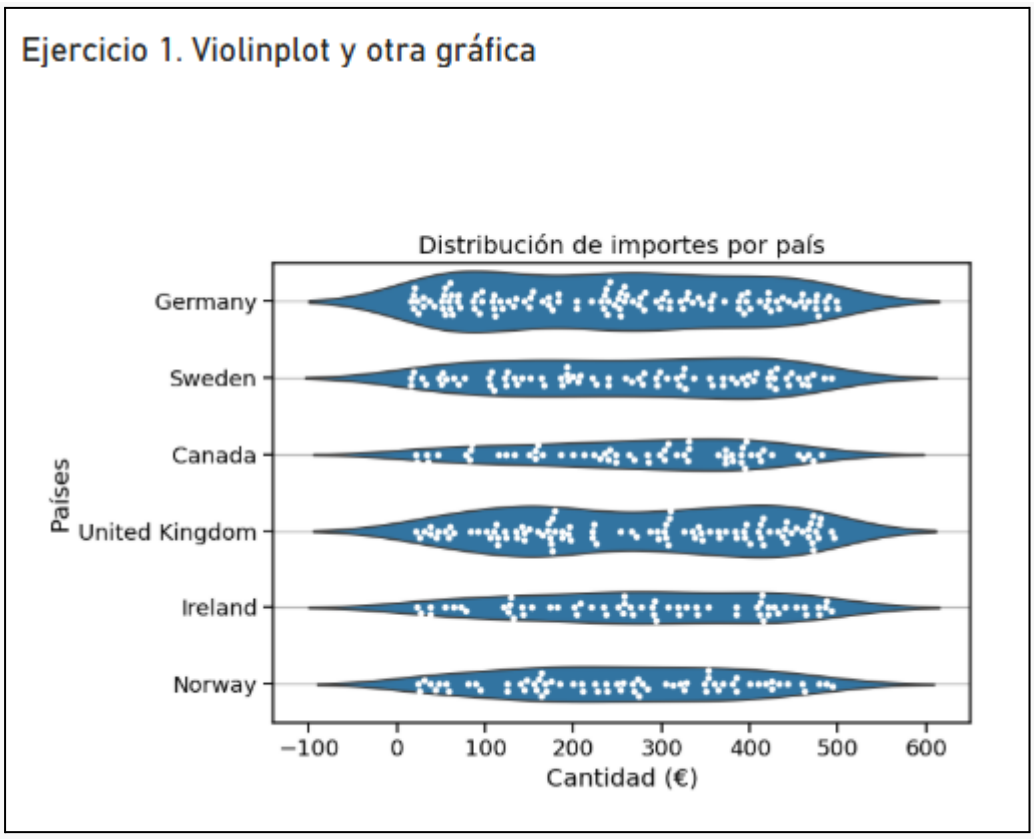
Script:

```
# El código siguiente, que crea un dataframe y quita las filas
duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(undefined, undefined.1)
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:

df_transaction_user_pr_wh = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.jointplot(data=dataset, x='amount', y='price', kind='reg')
plt.xlabel('Importe transacciones')
plt.ylabel('Precio envío')
plt.tight_layout()


plt.show()
```

Nivell 3

Els 2 exercicis del nivell 3 de la tasca 01

Ejercicio 1. Implementa un violinplot combinado con otro tipo de gráfica

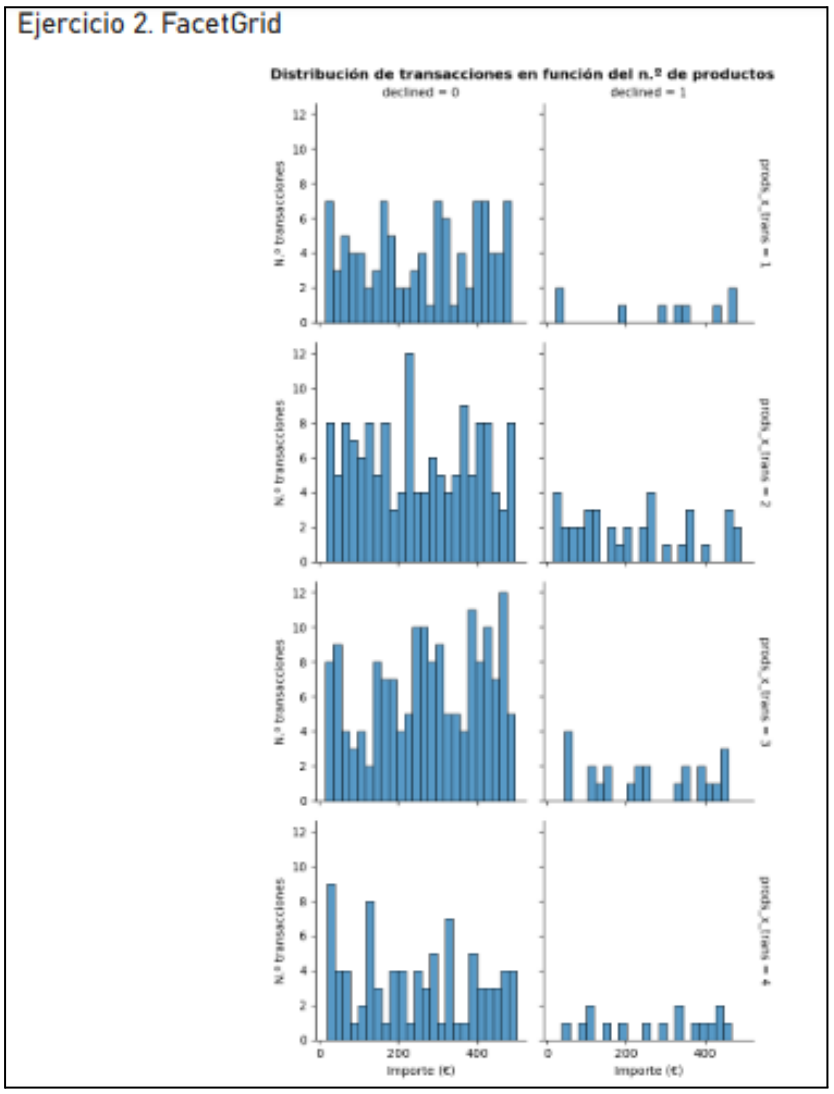


Script:

```
# El código siguiente, que crea un dataframe y quita las filas
duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(amount, country)
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:

df_transaction_country_top = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Vamos a añadir un swarmplot por encima para situar los valores de las
transacciones

# Creamos la figura:
sns.set_context('notebook', font_scale=1.2)
fig, ax = plt.subplots(figsize=(8,5))
```

```
# Dibujamos el violin
ax = sns.violinplot(y="country",
                    x="amount",
                    data=dataset,
                    density_norm='count',
                    inner=None
                )


# Situamos por encima el swarmplot
ax = sns.swarmplot(y="country",
                     x="amount",
                     data=dataset,
                     color="white",
                     edgecolor="auto",
                     s=4, # Circle size
                )


ax.set_xlabel('Cantidad (€)')
ax.set_ylabel('Países')


ax.grid(axis='y')
ax.set_axisbelow(True)

plt.title('Distribución de importes por país')
plt.tight_layout()

plt.show()
```

Ejercicio 2. Genera un FacetGrid para visualizar múltiples aspectos de los datos simultáneamente



Script:

```
# El código siguiente, que crea un dataframe y quita las filas
duplicadas, siempre se ejecuta y actúa como un preámbulo del script:

# dataset = pandas.DataFrame(amount, declined, prods_x_trans)
# dataset = dataset.drop_duplicates()

# Pegue o escriba aquí el código de script:

df_transaction_user_pr_wh = dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

g = sns.FacetGrid(dataset, col='declined', row='prods_x_trans',
margin_titles=True)
```

```
g.map_dataframe(sns.histplot, x='amount', binwidth=20)
g.fig.suptitle("Distribución de transacciones en función del n.° de
productos", fontsize=12, fontweight='bold')
g.fig.subplots_adjust(top=.93)
g.set_axis_labels('Importe (€)', 'N.° transacciones')
#g.set_titles(fontsize='small')

plt.show()
```