SPRINT S8.02 – PYTHON: POWER BI CON PYTHON

Esta tarea 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 01 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.

NIVEL 1

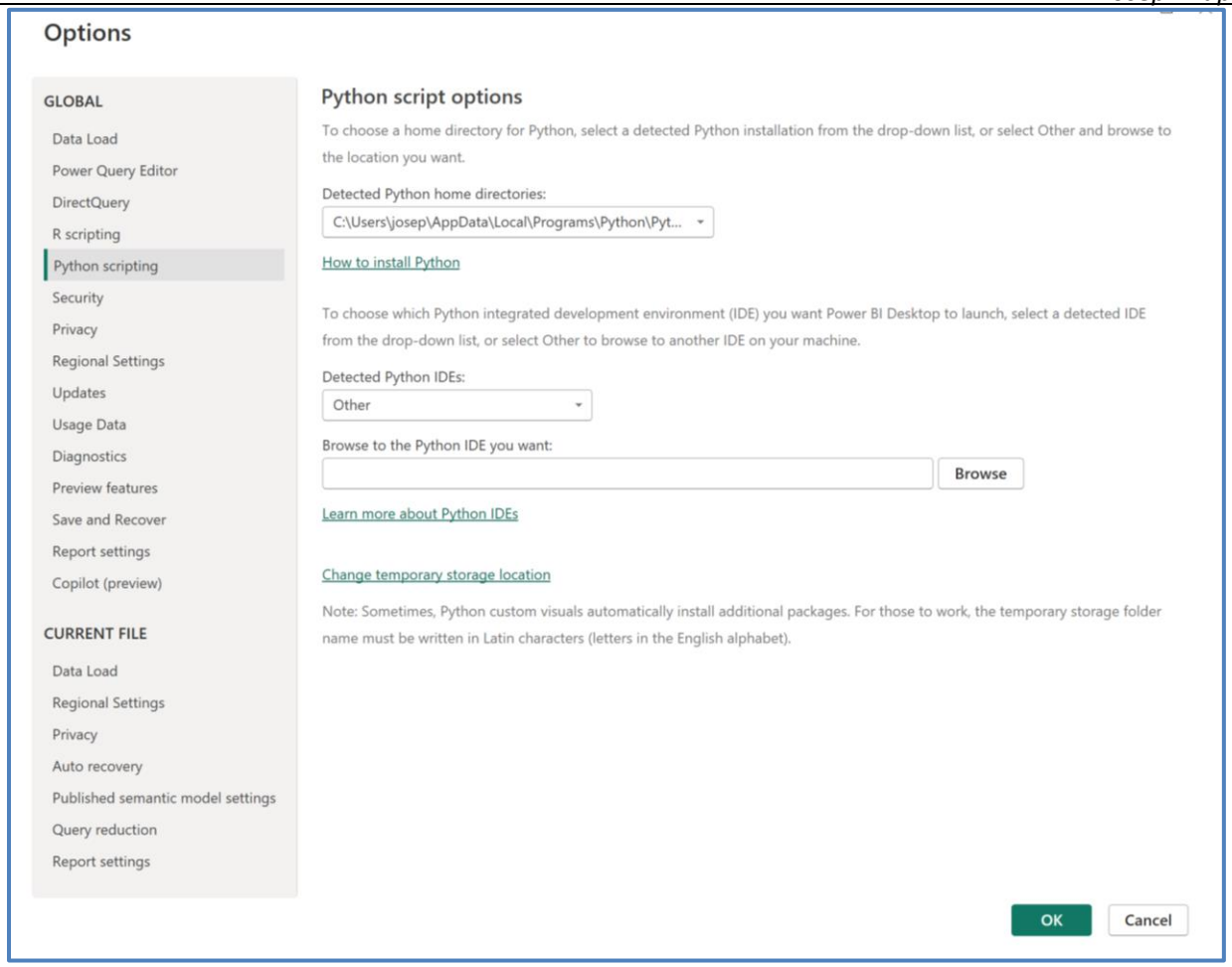
Realizar los 7 ejercicios del Nivel 1 de la Tarea 01.

Pasos previos:

El objetivo es conectar Python con Power BI y que todo esté bien configurado para poder correr el código desde Power BI y generar las gráficas. Con esa finalidad, a continuación se detalla el paso a paso.

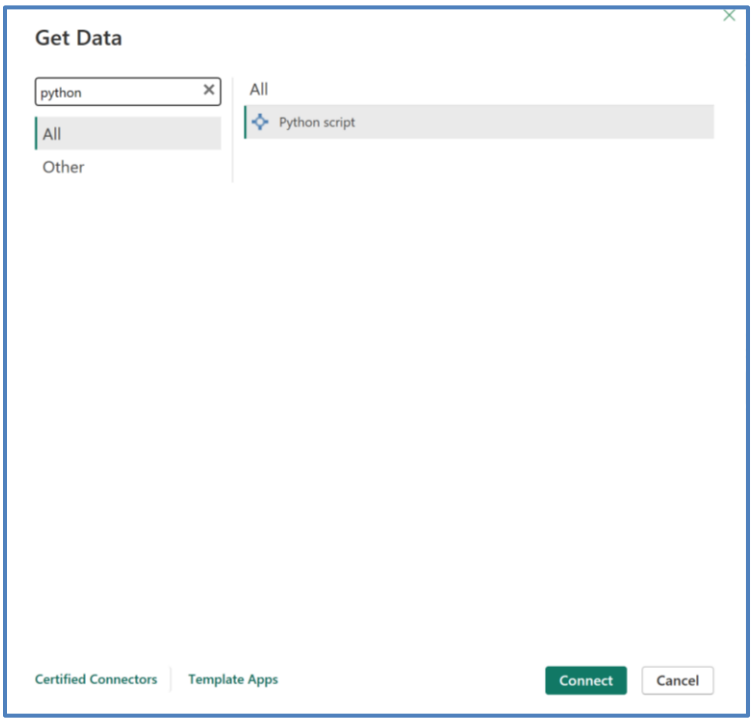
Primer Paso: Se verifica la carpeta asignada por defecto para los scripts de Python dentro del Power BI.

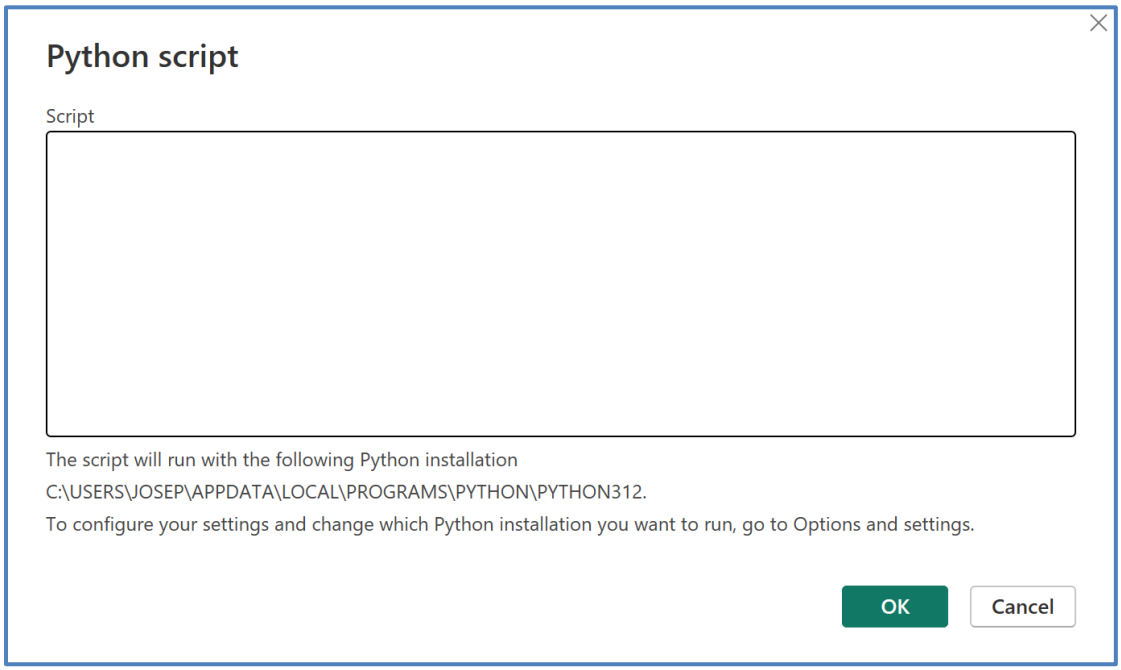
File → Options and settings → Options → Python scripting



Segundo Paso: A través del Python script se importarán los datos.

File → Get data → Get data to get started → Python script → Connect





Ingresamos el siguiente script de Python y lo corremos a través de → **OK**

```python
# Sección i:
import pandas as pd
from sqlalchemy import create_engine

# Sección ii:
host ='localhost'
database='transactionsts4'
user='root'
password='MIcuenta123'

conexion = f'mysql+mysqlconnector://{user}:{password}@{host}/{database}' # Creación de la cadena de la conexión

motor = create_engine(conexion) # Creación del motor de conexión

# Sección iii:
nombres_tablas = ['transactions', 'credit_cards', 'companies', 'users']

# Sección iv:
for nombre in nombres_tablas:
    query = f'SELECT * FROM {nombre}'
    df_name = f'df_{nombre}'
    globals()[df_name] = pd.read_sql(query, con=motor)

# Sección v:
transactions = df_transactions
creditcards = df_credit_cards
companies = df_companies
users = df_users
```

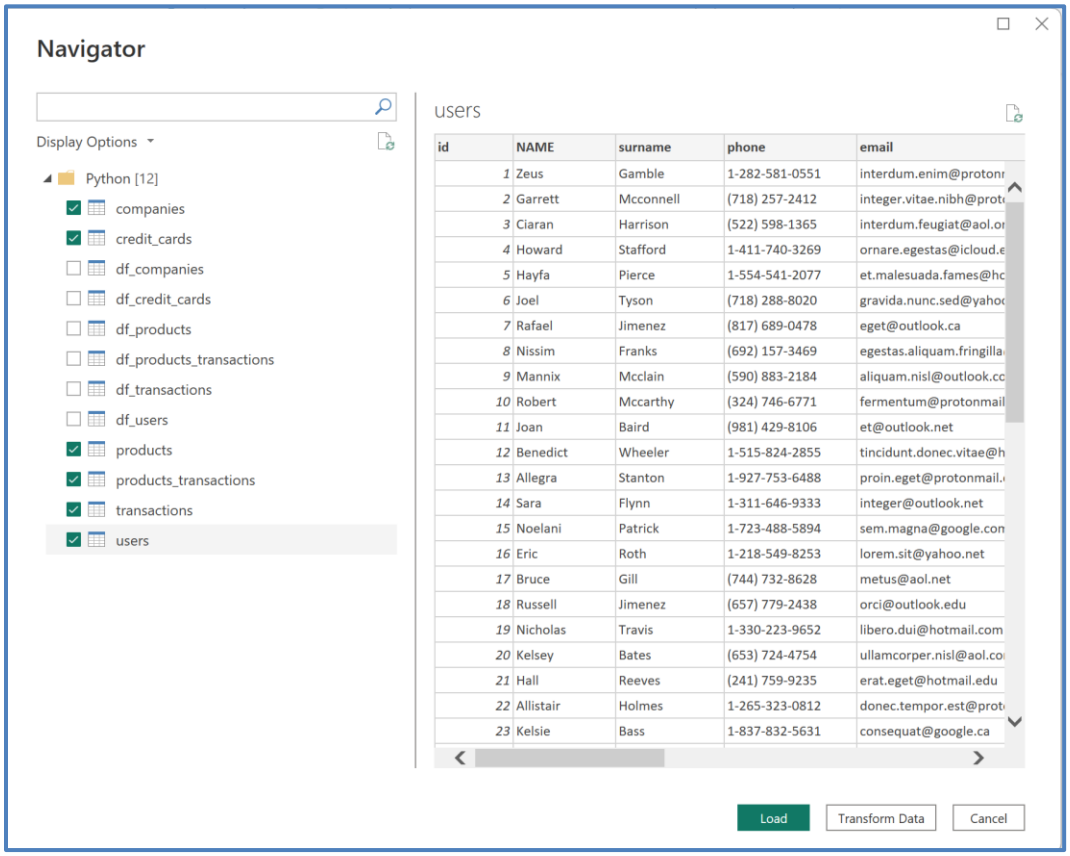
✓ 0.3s

Explicación del código de Python ingresado:

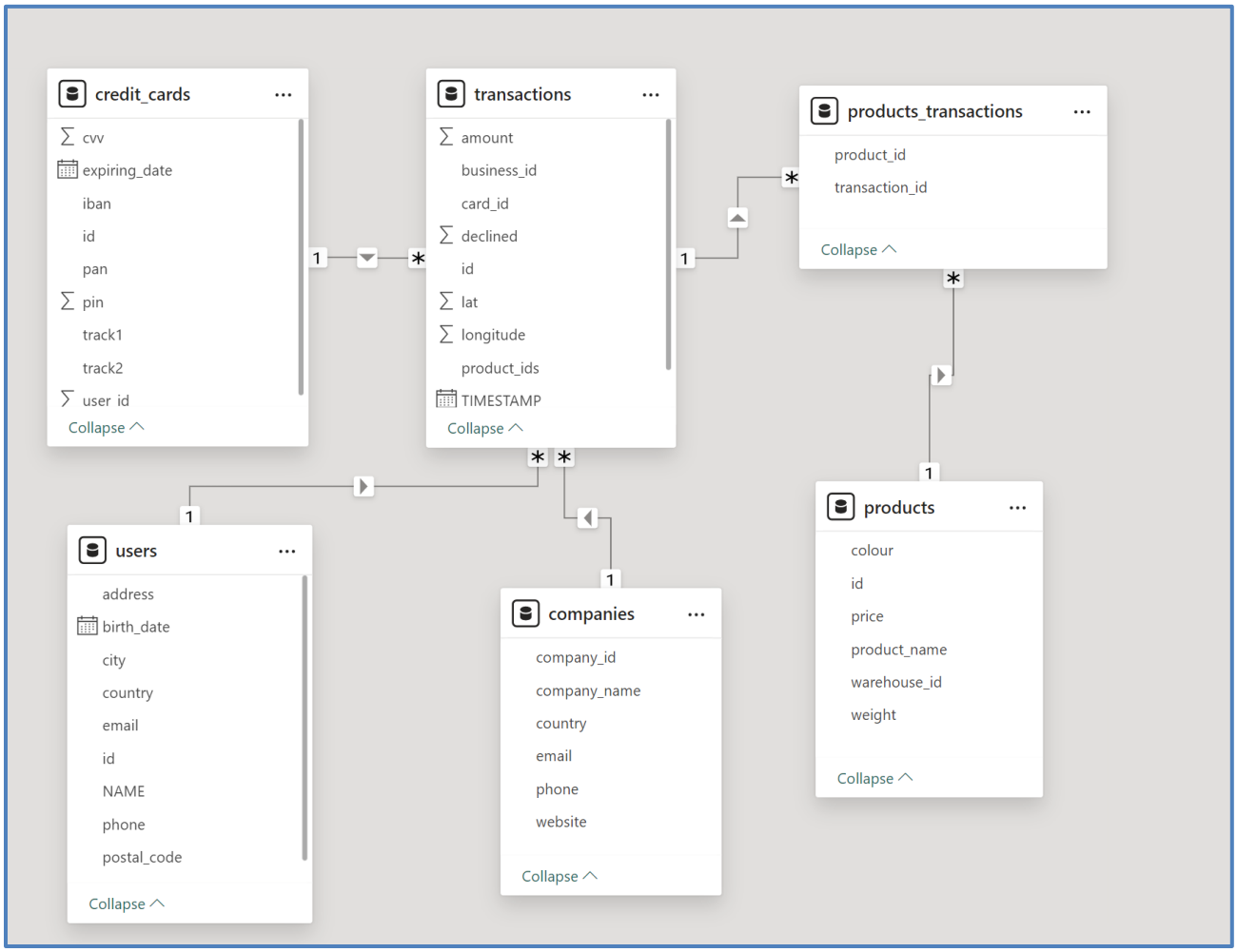
```
# Explicación:

A través del siguiente código se realiza la conexión con la base de datos MySQL para leer varias tablas y cargarlas en diferentes
Frames de Pandas, asignándoles nombres dinámicos basados en los nombres de las tablas.

- Sección i:
    Con Pandas se manipulará los datos en DataFrames, mientras que con SQLAlchemy estableceremos la conexión con la base de datos. Por ello
    importamos ambas librerías.

- Sección ii:
    Se realiza la conexión con la base de datos. A través de create_engine se crea un objeto de motor de conexión para poder ejecutar
    consultas SQL y transferir datos entre Python y la base de datos.

- Sección iii:
    Creación de la lista de las tablas que se cargan.

- Sección iv:
    Se utiliza el método globals() y con un bucle for se lee cada tabla en un DataFrame para almacenarlo en el diccionario.

- Sección v:
    Se asigna nombres predefinidos (transactions, creditcards,companies y users) a los DataFrames dinámicos creados previamente.
    Luego de ello se podrá trabajar con los DataFrames directamente.
```

Tercer Paso: Se cargan las tablas: companies, credit_cards, products, products_transactions, transactions y users.

→ **Load**



Cuarto Paso: Se establecen las relaciones entre las tablas de la BBDD de manera correcta, como se ha realizado en los *sprints* anteriores.

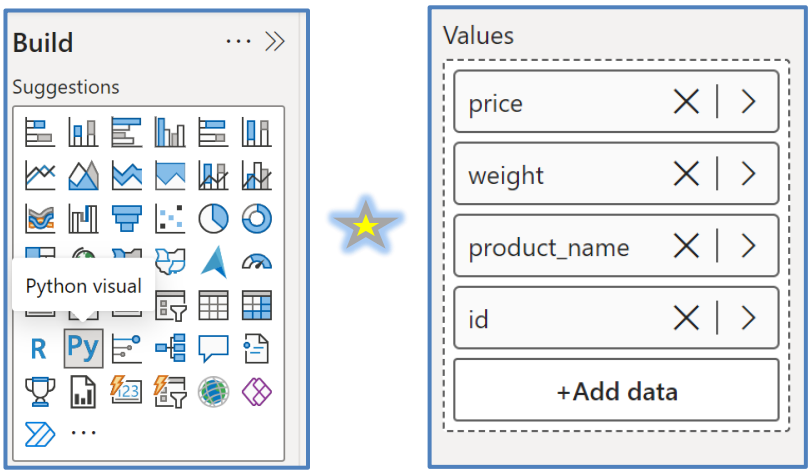


Consideraciones adicionales:

- Se han utilizado exactamente las mismas gráficas que en la tarea S08_01; sin embargo se han realizado pequeñas modificaciones para poder adaptar el formato de las gráficas y darles cierta homogeneidad para mostrarlos dentro del Dashboard del Power BI.

- Se ha tenido que incluir la importación de las librerías para correr cada uno de los scripts de los ejercicios. Las librerías ya habían sido instaladas para la tarea S08_01, por lo que solo fue necesario importarlas. Librerías utilizadas: Pandas, Numpy, Matplotlib, Seaborn, etc.

- Para generar cada gráfica se ha tenido que escoger ***Python Visual*** como construcción de gráfica y luego en *Values*, ingresar los campos necesarios de las tablas correspondientes. A continuación una pequeña muestra del ejercicio 1.2.

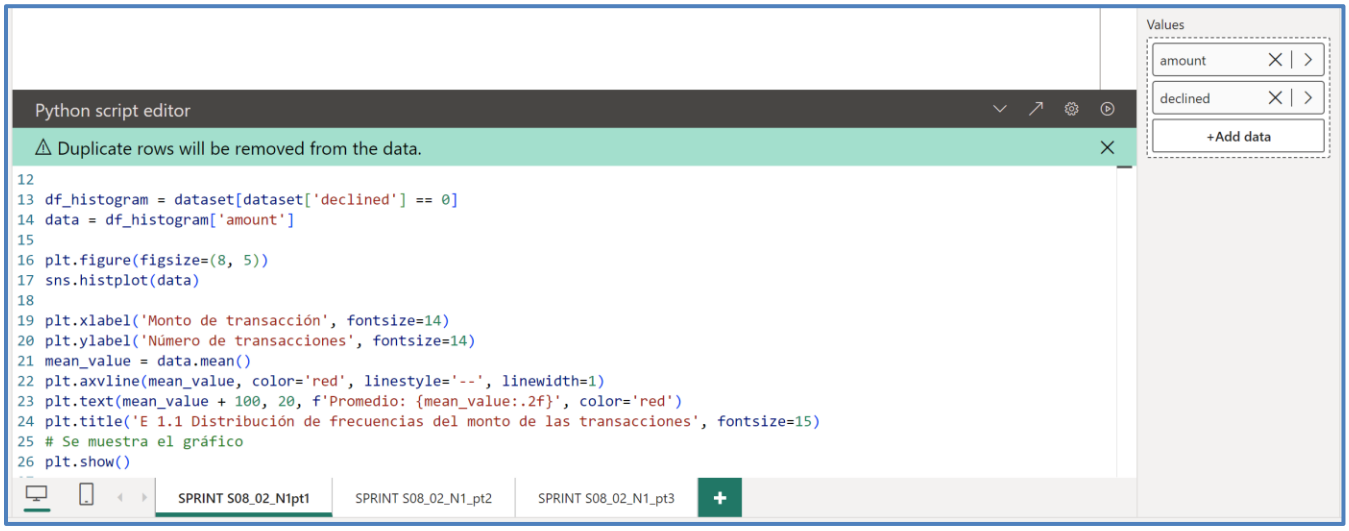


Con todo ello, ahora sí procedemos con la resolución de los ejercicios.

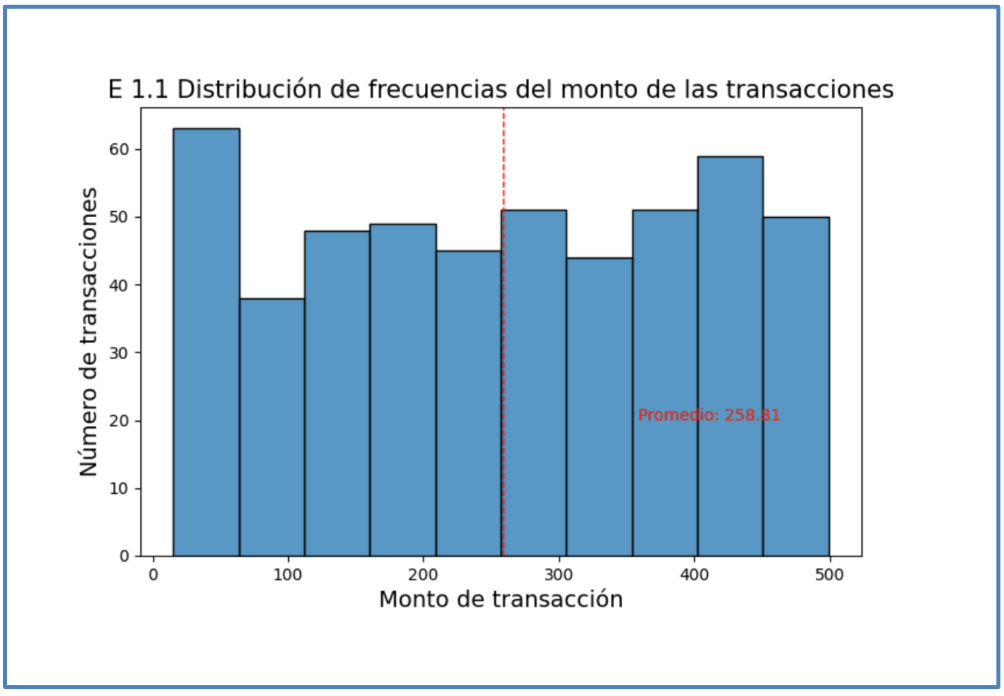
Ejercicio 1.1

Una variable numérica `--> Se utiliza el campo 'amount' de la tabla 'transactions'`

Código ingresado:



Gráfica:



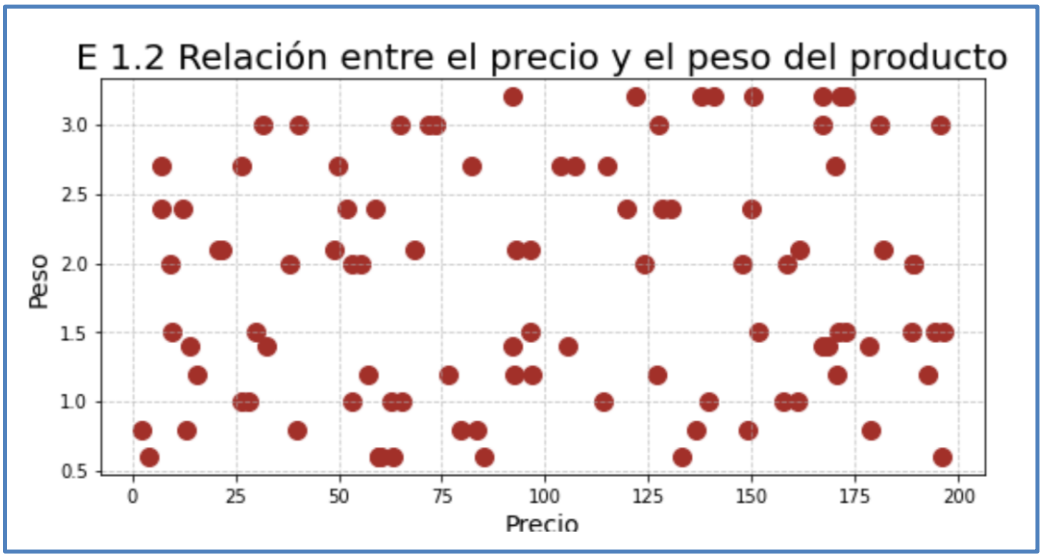
Ejercicio 1.2

Dos variables numéricas --> Se utilizan los campos 'price' y 'weight' de la tabla 'products'.

Código ingresado:

```
#Ejercicio 1.2: Scatterplot para variables precio y peso del producto

import matplotlib.pyplot as plt
import pandas as pd

import seaborn as sns

price = dataset['price']
weight = dataset['weight']

plt.scatter(x=price, y=weight, color='brown', s=100)
plt.xlabel('Precio', fontsize=14)  # Tamaño de la fuente
plt.ylabel('Peso', fontsize=14)
plt.title("E 1.2 Relación entre el precio y el peso del producto", fontsize=20)

plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

Gráfica:



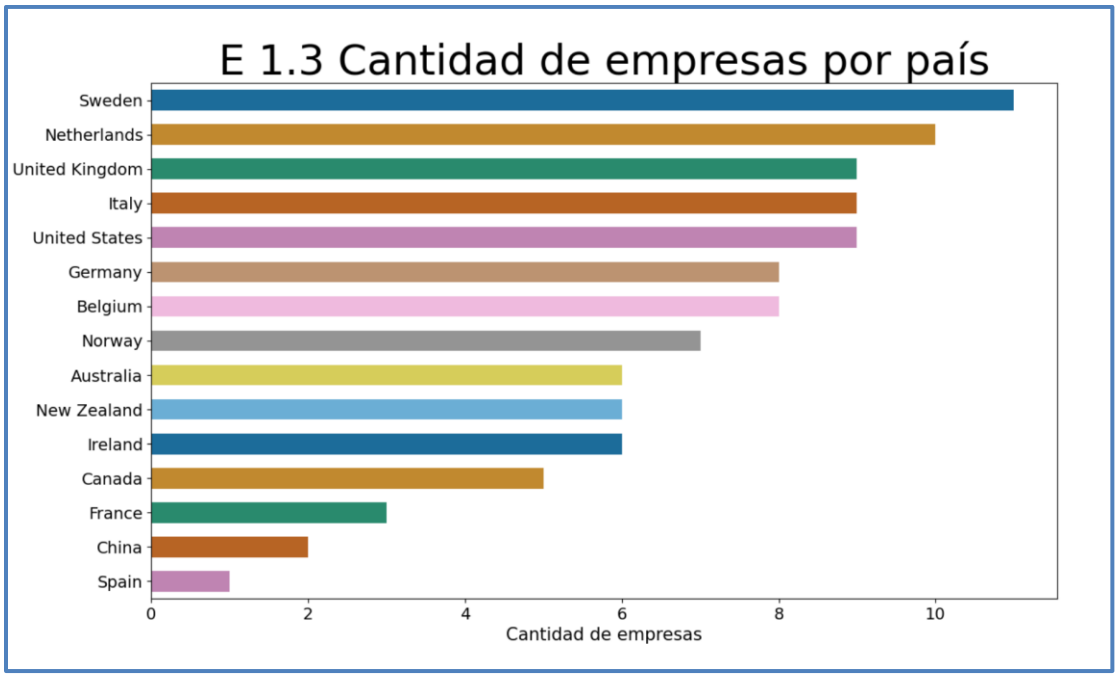
Ejercicio 1.3

Una variable categórica --> Se utiliza el campo 'country' de la tabla 'companies'.

Código ingresado:

```
# Ejercicio 1.3:

# Se importan las librerías:
import seaborn as sns
import matplotlib.pyplot as plt

# Se crea el gráfico para contar y mostrar en barras:

orden_country = dataset['country'].value_counts().index

plt.figure(figsize=(14, 8))
sns.countplot(y=dataset['country'], data=dataset, order = orden_country , palette='colorblind', width=0.6)

# Se añaden las etiquetas y el título:
plt.xlabel('Cantidad de empresas', fontsize=15)
plt.ylabel('País', fontsize=11)
plt.title('E 1.3 Cantidad de empresas por país', fontsize=36)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)

# Se muestra el gráfico:
plt.show()
```

Gráfica:



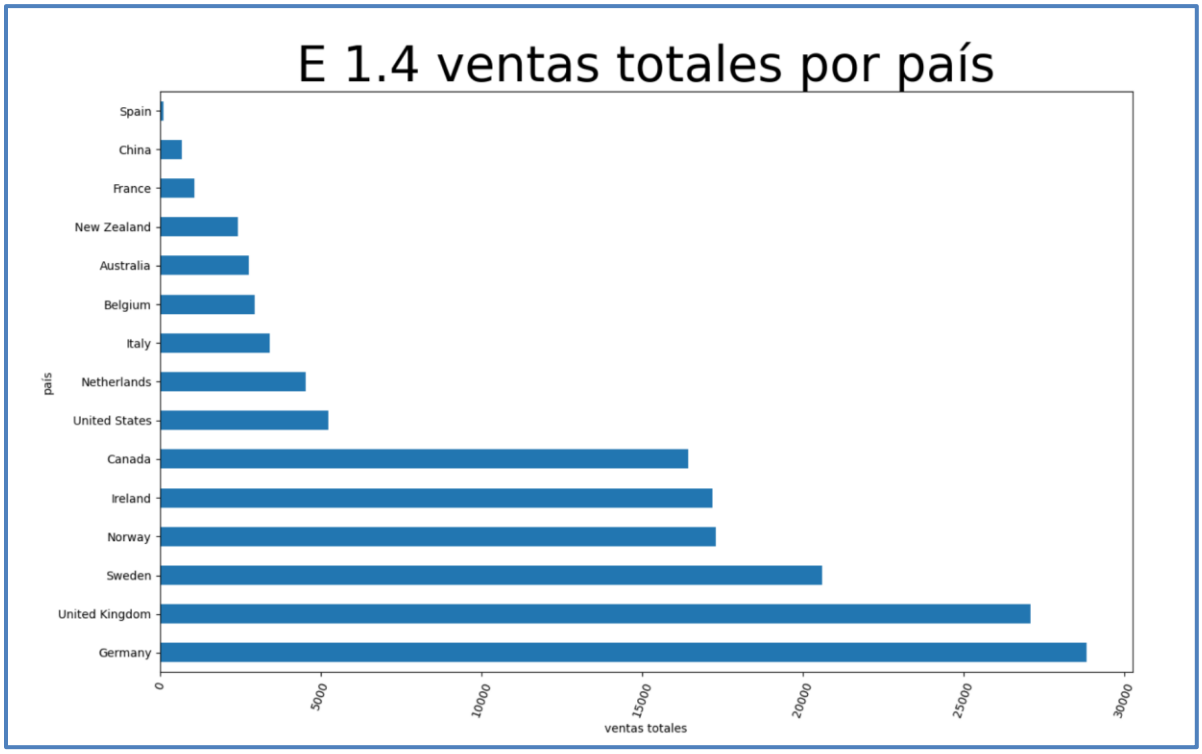
Ejercicio 1.4

Una variable categórica y una numérica --> Se utilizan los campos 'country' y 'amount' de las tablas 'companies' y 'transactions', respectivamente.

Código ingresado:

```
# Ejercicio 1.4: Una variable categórica y una numérica

import matplotlib.pyplot as plt

ventas_pais = dataset.groupby('country')['amount'].sum()
ventas_pais = ventas_pais.sort_values(ascending=False)

print(ventas_pais)

plt.figure(figsize=(15,9))
ventas_pais.plot(kind='barh')

plt.xlabel('ventas totales')
plt.ylabel('país')
plt.title('E 1.4 ventas totales por país', fontsize=42)
plt.xticks(rotation=70)
plt.show()
```

Gráfica:



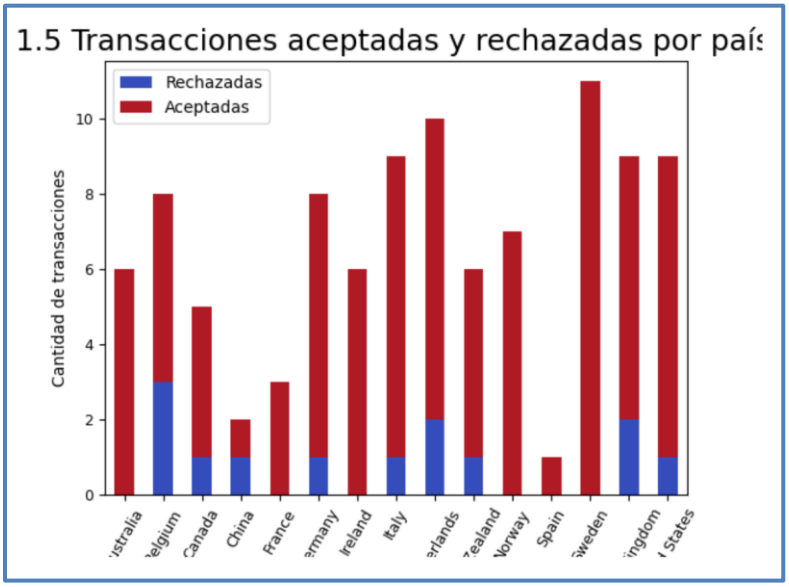
Ejercicio 1.5

Dos variables categóricas --> Se utilizan los campos 'declined' y 'país' de las tablas 'transactions' y 'companies', respectivamente.

Código ingresado:

```
# Ejercicio 1.5:

import pandas as pd
import matplotlib.pyplot as plt

companies = dataset[['company_id', 'country']]
transactions = dataset[['business_id', 'declined']]

# Se realiza un merge entre las tablas transactions y companies a través de los campos business_id y company_id
datos = pd.merge(transactions, companies, how='inner', left_on='business_id', right_on='company_id')

data_agrupada = datos.groupby('country')['declined'].value_counts().unstack()

data_agrupada.columns = ['Rechazadas', 'Aceptadas']

data_agrupada.plot(kind='bar', stacked=True, fontsize=9, colormap ='coolwarm' )

plt.title('E 1.5 Transacciones aceptadas y rechazadas por países', fontsize=18)
plt.xlabel('País')
plt.ylabel('Cantidad de transacciones')

plt.xticks(rotation=60)

plt.show()
```

Gráfica:



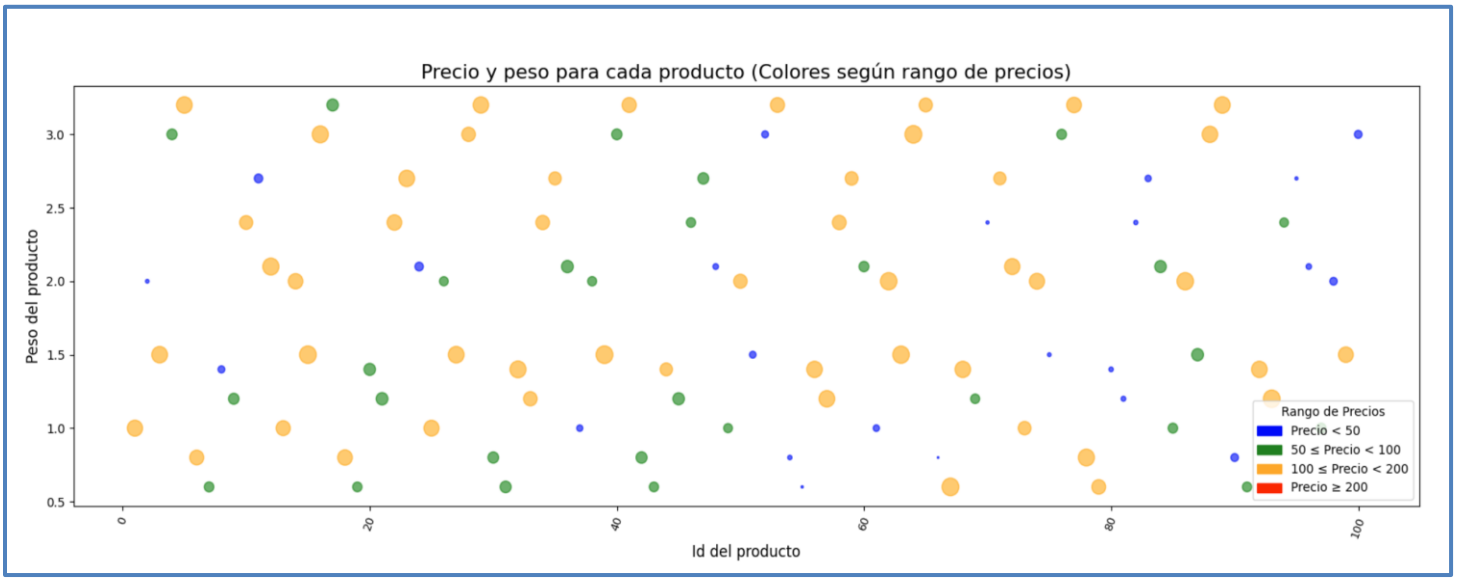
Ejercicio 1.6

Tres variables -- > Se utilizan los campos 'weight', 'price' e 'id' de la tabla 'products'.

Código ingresado:

```
# Ejercicio 1.6:

import matplotlib.pyplot as plt
import numpy as np

colores = np.where(dataset['price'] < 50, 'blue', np.where(dataset['price'] < 100, 'green', np.where(dataset['price'] < 200, 'orange', 'red')) )

plt.figure(figsize=(20, 6))
scatter = plt.scatter(dataset['id'], dataset['weight'], color=colores, alpha=0.6, s=dataset['price'])

plt.xlabel('Id del producto', fontsize=12)
plt.ylabel('Peso del producto', fontsize=12)
plt.title('Precio y peso para cada producto (Colores según rango de precios)', fontsize=16)
plt.xticks(rotation=70, fontsize=9)

# Se crea una leyenda manual para explicar los colores
import matplotlib.patches as mpatches
leyenda_azul = mpatches.Patch(color='blue', label='Precio < 50')
leyenda_verde = mpatches.Patch(color='green', label='50 ≤ Precio < 100')
leyenda_naranja = mpatches.Patch(color='orange', label='100 ≤ Precio < 200')
leyenda_rojo = mpatches.Patch(color='red', label='Precio ≥ 200')

plt.legend(handles=[leyenda_azul, leyenda_verde, leyenda_naranja, leyenda_rojo], title='Rango de Precios', fontsize=10)

# Mostrar el gráfico
plt.show()
```

Gráfica:



Ejercicio 1.7

Graficar un Pairplot --> Se han utilizado los campos 'price' y 'weight' de la tabla 'products'.

Código ingresado:

```
# Ejercicio 1.7 Pairplot --> Se ha utilizado los campos price y weight de la tabla products.

import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('classic')

pairplot = sns.pairplot(dataset, vars=['price', 'weight'], diag_kind='kde', kind='reg')

# Se agrega un título global a la figura del pairplot
pairplot.fig.suptitle(
    'E 1.7 Pairplot del precio y el peso de los productos', fontsize=14, y=1.10)

for ax in pairplot.axes.flatten():
    plt.setp(ax.get_xticklabels(), rotation=45)

plt.show()
```

Gráfica: