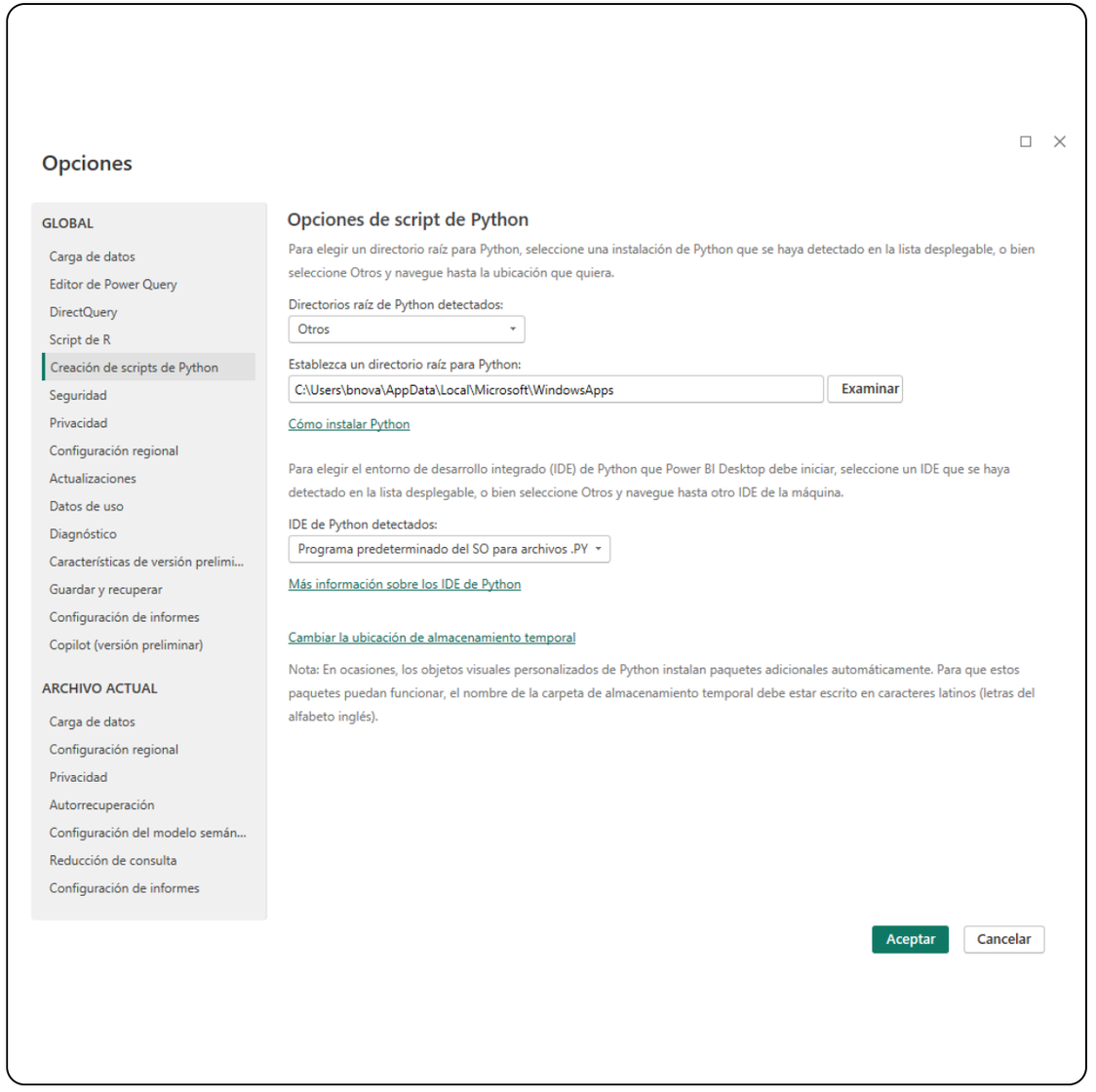
SPRINT 8

TAREA 2

VISUALIZACIONES SCRIPTS PYTHON EN POWER BI

Marcos Novalvos Gómez - Especialización Data Analytics

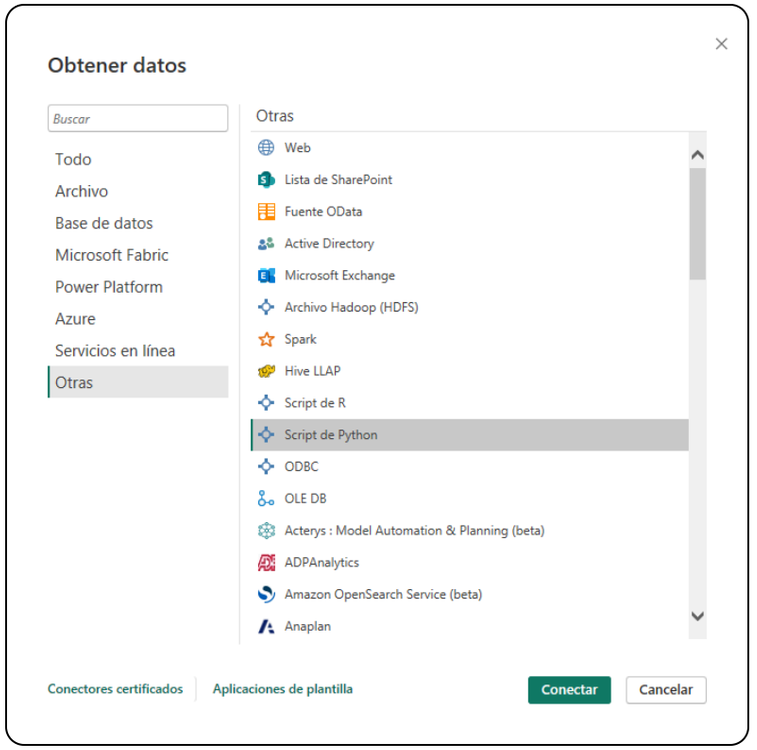


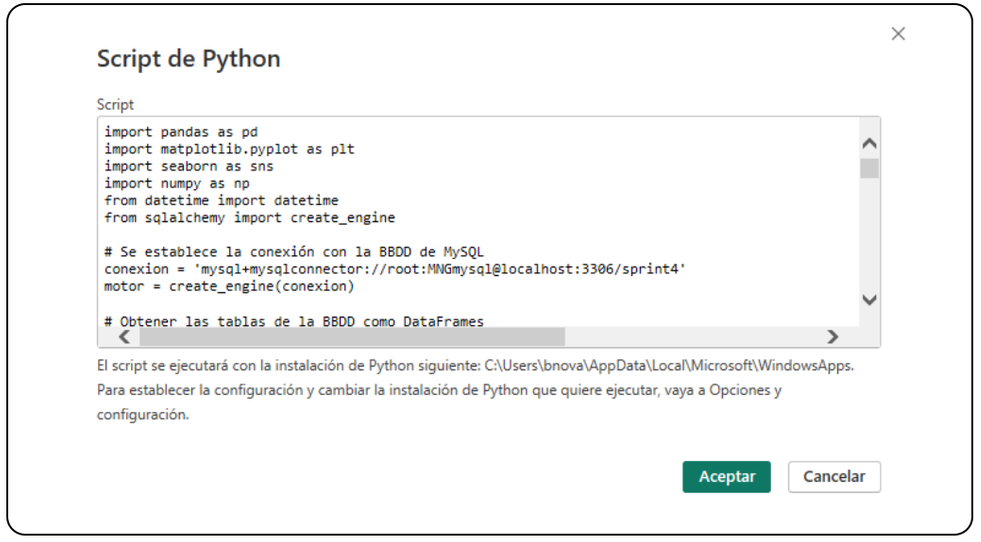
HABILITAR SCRIPTS DE PYTON EN POWER BI

Desde el menu ***Opciones y configuración*** de Power BI acceder al menú ***Creación de scripts de Python.***

Para habilitar los scripts de Python dentro de Power BI, debemos indicar la ruta donde tenemos instalado el propio ***Python***.

También es necesario indicar la ruta donde se ubica el ejecutable del cliente Python, en este caso ***Visual Studio Code.***





CARGAR DATAFRAMES DE PYTON EN POWER BI

Desde el menu ***Inicio*** de Power BI acceder al menú ***Obtener datos.***

En el sub-menú ***Otras***, seleccionar ***Script de Python***.

Pegar el script que hayamos desarrollado en ***Visual Studio Code***, para la creación de los ***DataFrames (DF)*** así como las transformaciones necesarias que hayamos realizado.

DATAFRAMES ORIGINALES DE LA BBDD TRANSACTIONS

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from datetime import datetime
from sqlalchemy import create_engine

# Se establece la conexión con la BBDD de MySQL
conexion = 'mysql+mysqlconnector://root:MNGmysql@localhost:3306/sprint4'
motor = create_engine(conexion)

# Obtener las tablas de la BBDD como DataFrames
df_companies = pd.read_sql("SELECT * FROM companies", conexion)
df_credit_cards = pd.read_sql("SELECT * FROM credit_cards", conexion)
df_estado_tarjetas = pd.read_sql("SELECT * FROM estado_tarjetas", conexion)
df_products = pd.read_sql("SELECT * FROM products", conexion)
df_products_transactions = pd.read_sql("SELECT * FROM products_transactions", conexion)
df_transactions = pd.read_sql("SELECT * FROM transactions", conexion)
df_users = pd.read_sql("SELECT * FROM users", conexion)
```

NIVEL1 - DATAFRAMES NUEVOS Y/O TRANSAFORMADOS

```python
# Nivel 1 - Ejercicio 2
df_products = df_products.assign(precio = df_products["price"])
df_products['precio'] = df_products['precio'].str.replace('$','')
df_products['precio'] = df_products['precio'].astype(float)

# Nivel 1 - Ejercicio 4
df_transacionesCompañias=pd.merge(df_transactions, df_companies, left_on='business_id', right_on='company_id', how='inner')

# Nivel 1 - Ejercicio 5
df_estadoTransactionPais=pd.crosstab(df_companies["country"],df_transactions["declined"])
df_estadoTransactionPais.columns.name = None
df_estadoTransactionPais = df_estadoTransactionPais.reset_index()
df_estadoTransactionPais

# Nivel 1 - Ejercicio 6
df_transacionesCompañias=pd.merge(df_transactions, df_companies, left_on='business_id', right_on='company_id', how='inner')
def_paisesTOP= df_transacionesCompañias.groupby("country")["amount"].sum().nlargest(3).index
df_paisesTOPFiltrado=df_transacionesCompañias[df_transacionesCompañias["country"].isin(def_paisesTOP)]

# Nivel 1 - Ejercicio 7
df_numProductosTransacion = df_products_transactions.groupby('transactions_id', as_index=False).count()
df_numProductosTransacion.rename(columns={'products_id':'numProductos'}, inplace=True)
df_numProductosTransacion.rename(columns={'transactions_id':'transaction_id'}, inplace=True)
df_users.rename(columns={'id':'user_id'}, inplace=True)
df_users["birth_date"] = pd.to_datetime(df_users["birth_date"])
df_users["edad"] = (datetime.now().year - df_users["birth_date"].dt.year)
df_transactions.rename(columns={'id':'transaction_id'}, inplace=True)
df_transactionsConNumeroProductos=pd.merge(df_transactions, df_numProductosTransacion, on='transaction_id', how='inner')
df_transactionsFull=pd.merge(df_transactionsConNumeroProductos, df_users, on='user_id', how='inner')
df_datos_interes = df_transactionsFull[['transaction_id','numProductos','declined','amount','user_id','edad','country']]
```

NIVEL2 - DATAFRAMES NUEVOS Y/O TRANSAFORMADOS

```
# Nivel 2 - Ejercicio 1
df_transactionsPeso=pd.merge(df_products_transactions, df_products, left_on='products_id', right_on='id', how='inner')
df_datosTransactionPeso = df_transactionsPeso[['transactions_id','weight']]
df_transactionPeso = df_datosTransactionPeso.groupby('transactions_id', as_index=False).sum()
df_transactionsResumenVarNumericas=pd.merge(df_datos_interes, df_transactionPeso, left_on='transaction_id', right_on='transactions_id', how='inner')
df_transactionsResumenVarNumericas.rename(columns={'weight':'peso'}, inplace=True)
df_transactionsResumenVarNumericas[['transaction_id','numProductos','amount','edad','peso']]

# Nivel 2 - Ejercicio 2
df_edadNumTransacciones=df_transactionsResumenVarNumericas[['amount','edad',]]
df_edadAmount = df_edadNumTransacciones.groupby('edad', as_index=False).mean()
df_edadAmount.rename(columns={'amount':'importe medio'}, inplace=True)
```

NIVEL3 - DATAFRAMES NUEVOS Y/O TRANSAFORMADOS

```
# Nivel 3 - Ejercicio 1
df_paisesTOPResumido=df_paisesTOPFiltrado[['id','country','amount','declined']]

# Nivel 3 - Ejercicio 2
df_varNumericas = sns.FacetGrid(df_transactionsResumenVarNumericas, col='numProductos', row='declined', margin_titles=True)
```

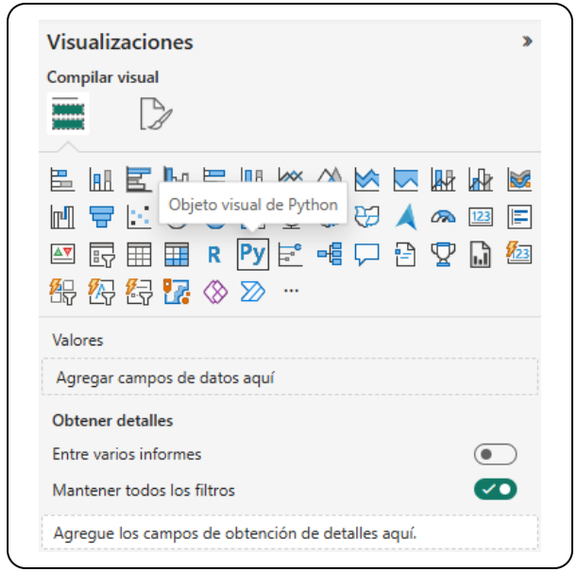
COMO CREAR UN GRÁFICO A PARTIR DE UN SCRIPT DE PYTON

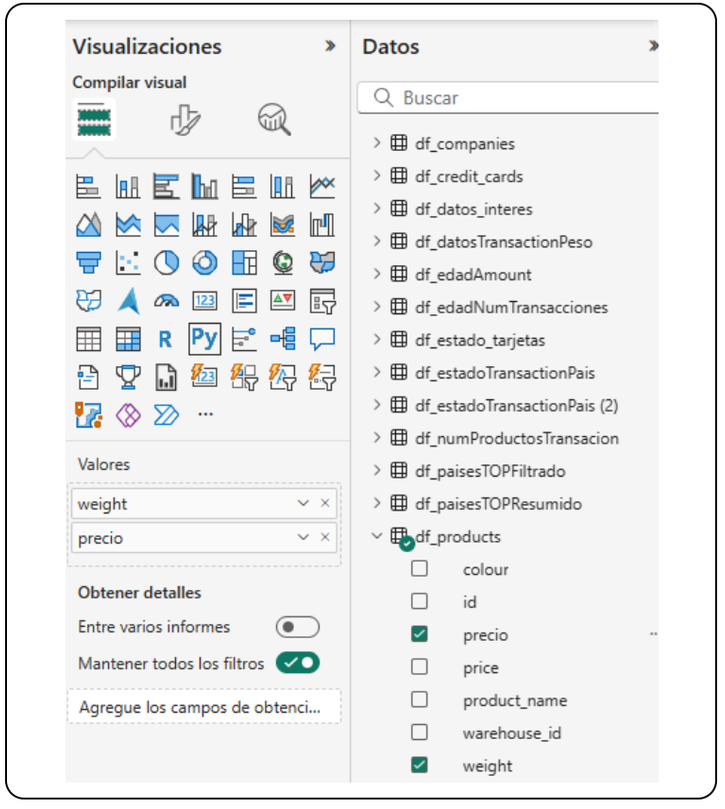
Seleccionar la visualización ***Objeto visual de Python***.

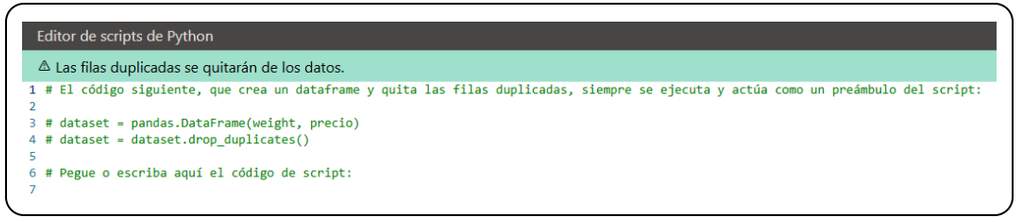
Seleccionar las ***variables*** que necesitamos para la gráfica desde el ***DataFrame*** necesario (todas las***variables*** deben estar recogidas en el mismo ***DataFrame***.

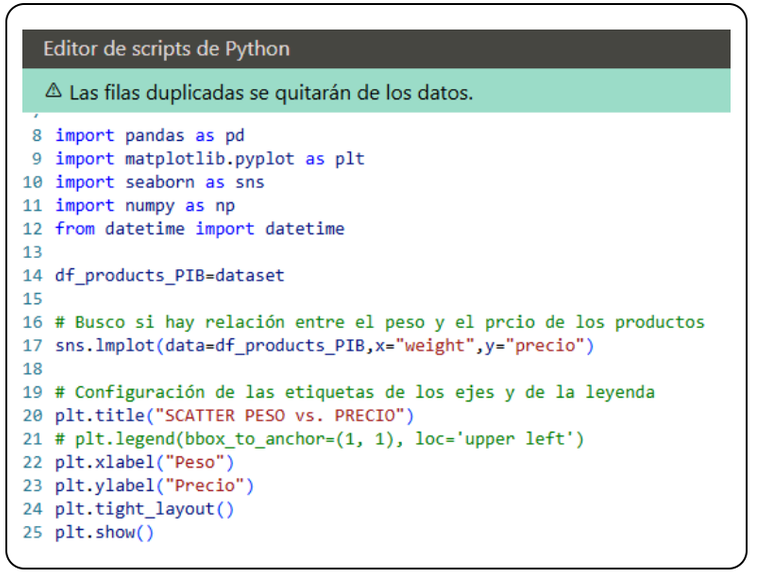
Copiar en el ***DataFrame*** el ***dataset*** generado automáticamente por Power BI a partir de las ***variables*** seleccionadas anteriormente.

Pegar el ***script*** generado en ***Visual Studio Code*** de la visualización a realizar.





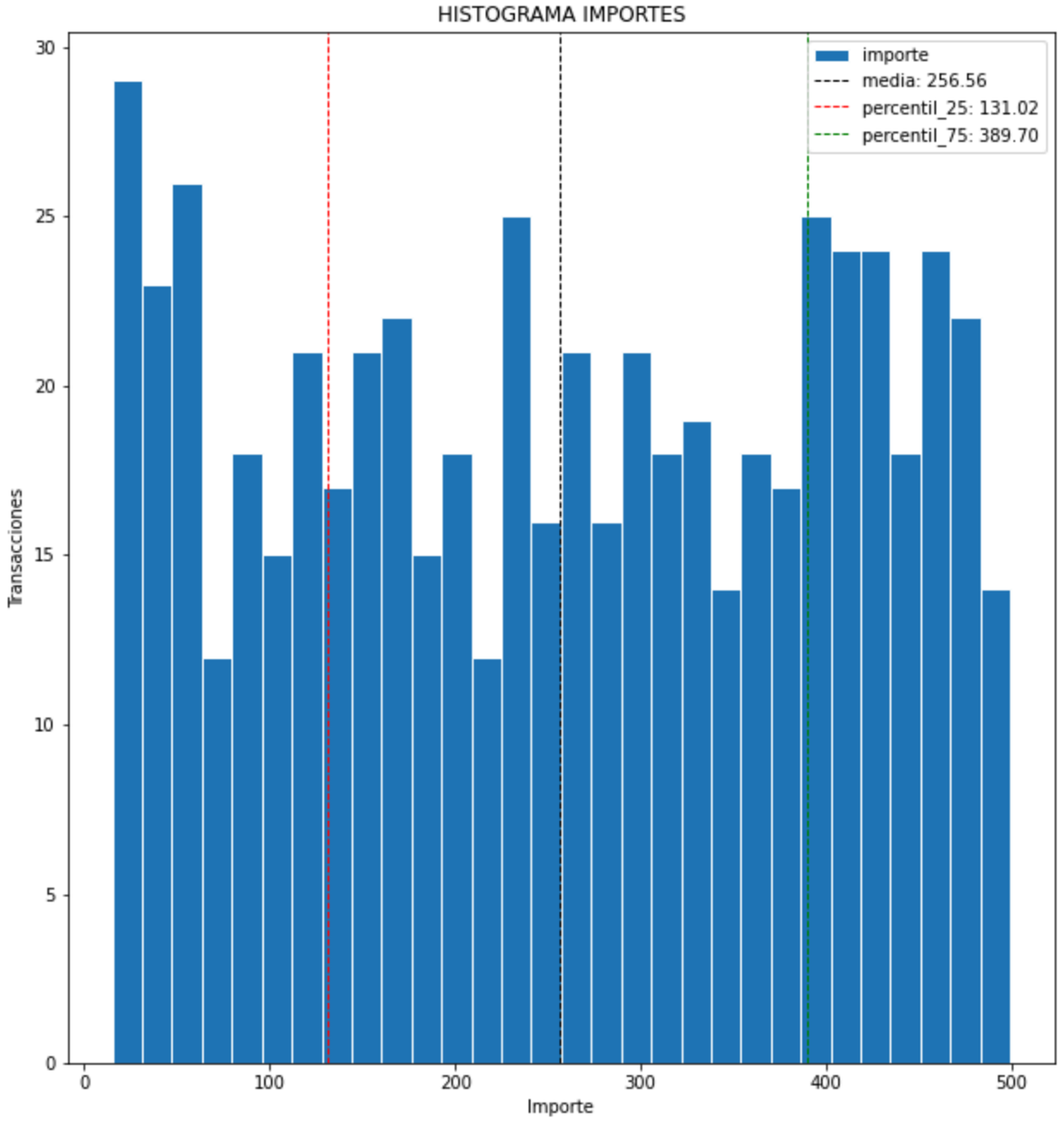




NIVEL 1

1 VARIABLE NUMÉRICA

HISTOGRAMA - Mostrar la frecuencia con la que se repite un rango de importes

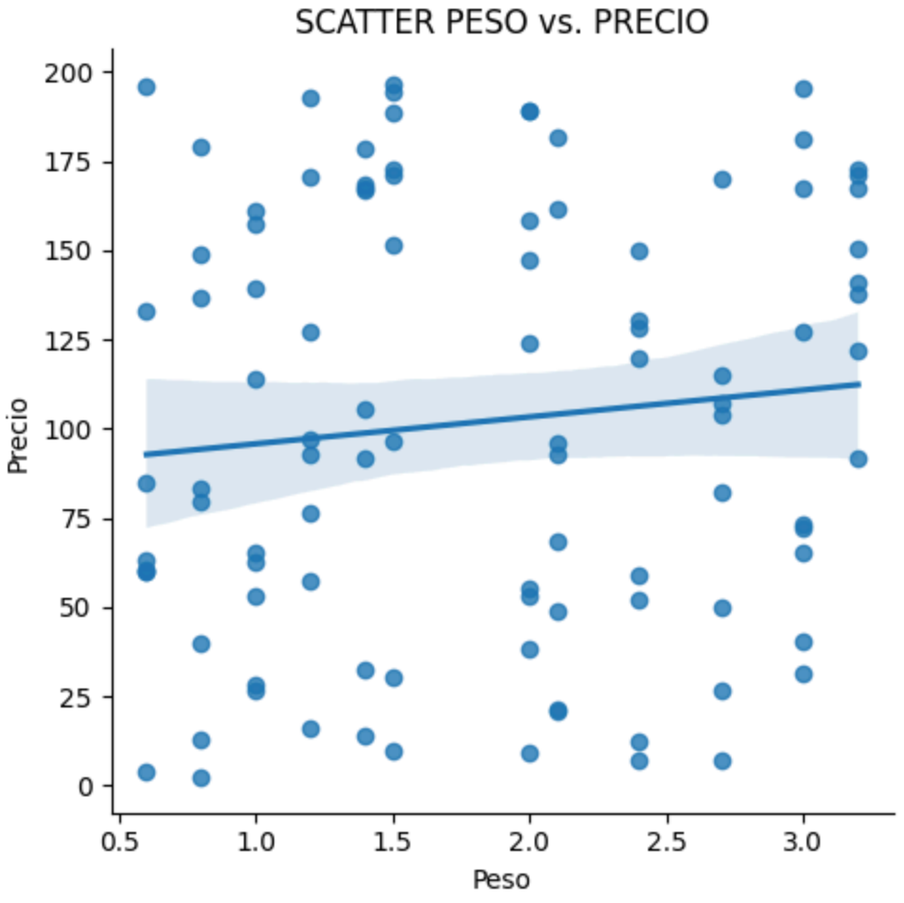


SCRIPT

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

df_transactions_PIB=dataset

# Calculo las 4 medidas estadísticas básicas a partir del importe por transacción
amount = df_transactions_PIB["amount"]

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

# Gráfica en la que se muestran el histograma que representa a la variable numérica "amount"
df_transactions_PIB["amount"].plot.hist(bins=30,edgecolor="white",label="importe")

# Lineas auxilliares sobre el gráfico
plt.axvline(media, color='black', linestyle='dashed', linewidth=1, label=f'media: {media:.2f}')
plt.axvline(percentil_25, color='red', linestyle='dashed', linewidth=1, label=f'percentil_25: {percentil_25:.2f}')
plt.axvline(percentil_75, color='green', linestyle='dashed', linewidth=1, label=f'percentil_75:
{percentil_75:.2f}')

# Configuración de las etiquetas de los ejes y de la leyenda
plt.title("HISTOGRAMA IMPORTES")
plt.legend(bbox_to_anchor=(1, 1), loc='upper right')
plt.xlabel("Importe")
plt.ylabel("Transacciones");
plt.show()
```

2 VARIABLES NUMÉRICAS

SCATTER - Buscar si hay relación entre el peso y el precio de los productos

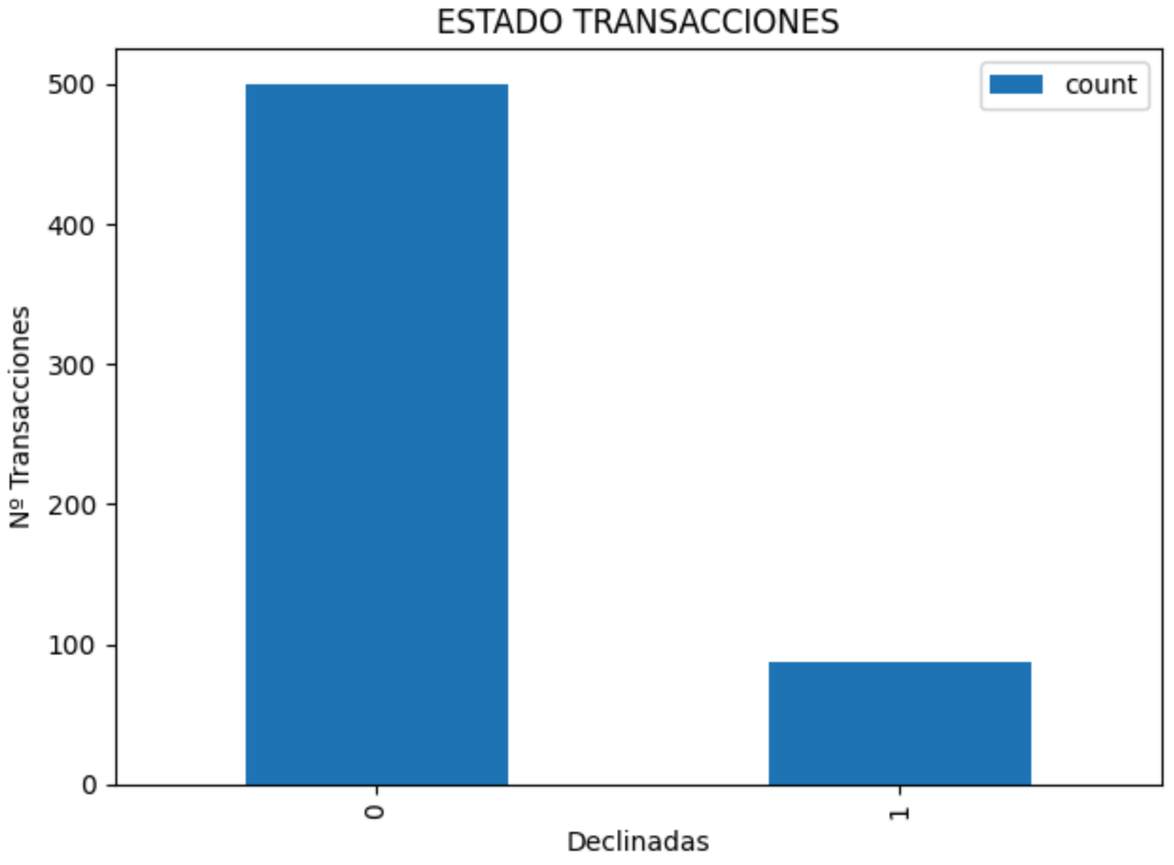


SCRIPT

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

df_products_PIB=dataset

# Busco si hay relación entre el peso y el precio de los productos
sns.lmplot(data=df_products_PIB,x="weight",y="precio")

# Configuración de las etiquetas de los ejes y de la leyenda
plt.title("SCATTER PESO vs. PRECIO")
plt.xlabel("Peso")
plt.ylabel("Precio")
plt.tight_layout()
plt.show()
```

1 VARIABLE CATEGÓRICA

BARRAS - Por ejemplo, visualizar el nº de transacciones que han sido declinadas y las que no

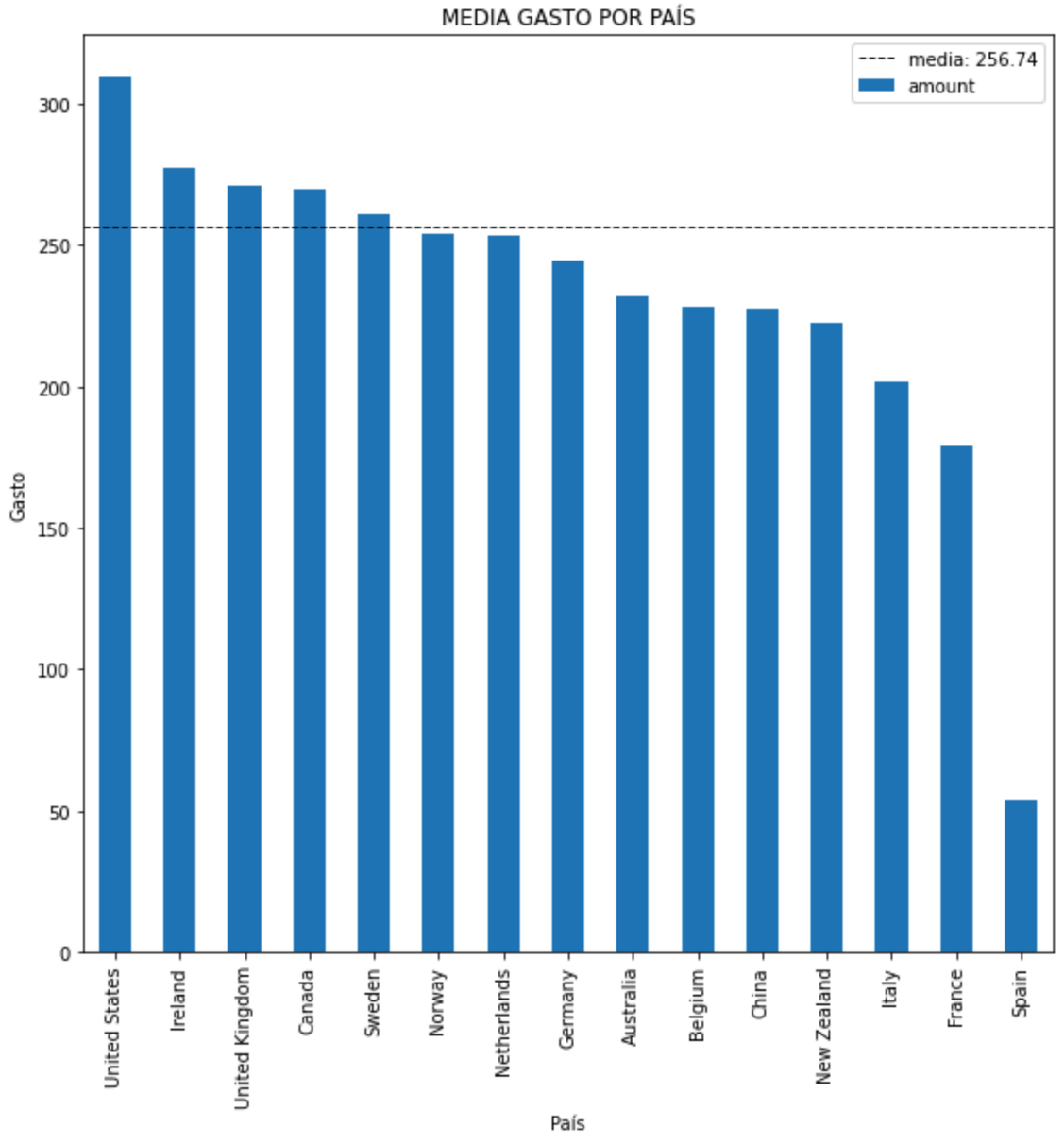


SCRIPT

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

df_transactions_PIB=dataset

# Relacionar el estado de las transacciones
pd.DataFrame(df_transactions_PIB["declined"].value_counts()).plot.bar()

# Configuración de las etiquetas de los ejes y de la leyenda
plt.title("ESTADO TRANSACCIONES")
plt.legend(bbox_to_anchor=(1, 1), loc='upper right')
plt.xlabel("Declinadas")
plt.ylabel("Nº Transacciones")
plt.show()
```

1 VARIABLE CATEGÓRICA Y 1 VARIABLE NUMÉRICA

BARRAS - Relación entre la media de gasto por país

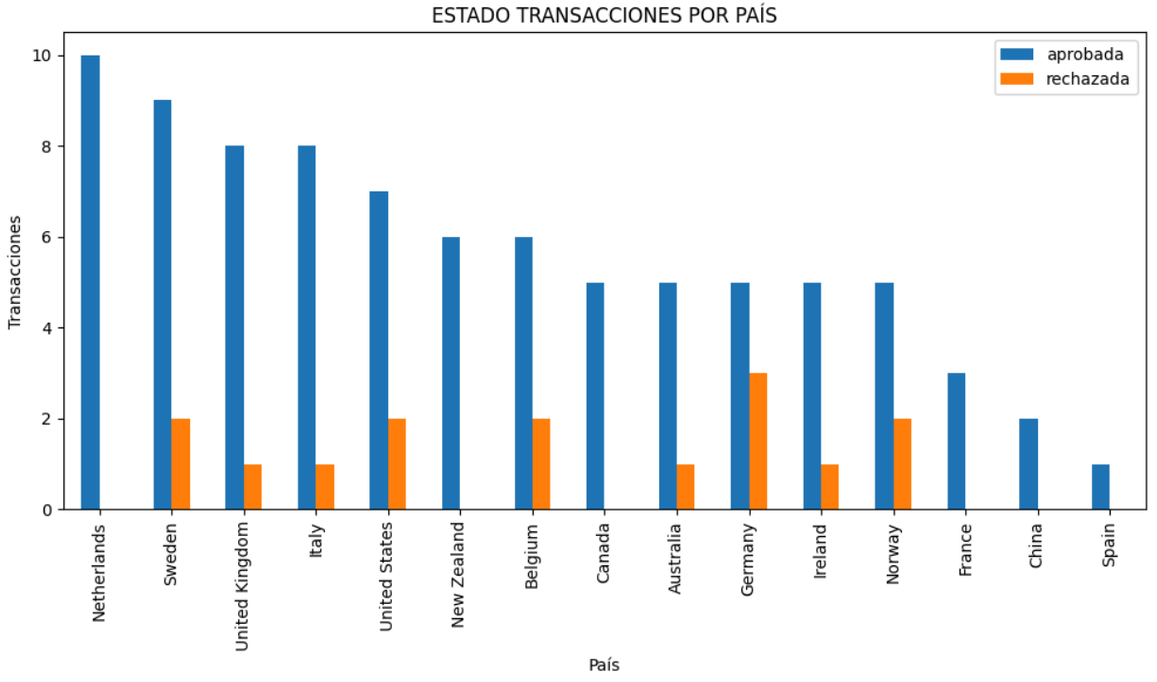


SCRIPT

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

df_transacionesCompañias_PIB=dataset

# Realizo un GROUP BY para representar la relación entre los países y la media de consumo de cada uno
df_transacionesCompañias_PIB.groupby("country")["amount"].mean().sort_values(ascending=False).plot.bar()

# Calculo la media total
amount = df_transacionesCompañias_PIB["amount"]
media = amount.mean()

# Lineas auxilliares sobre el gráfico
plt.axhline(media, color='black', linestyle='dashed', linewidth=1, label=f'media: {media:.2f}')

# Configuración de las etiquetas de los ejes y de la leyenda
plt.title("MEDIA GASTO POR PAÍS")
plt.legend(bbox_to_anchor=(1, 1), loc='upper right')
plt.xlabel("País")
plt.ylabel("Gasto")
plt.tight_layout()
plt.show()
```

2 VARIABLES CATEGÓRICAS

BARRAS - Nº de transacciones declinadas y no declinadas por país

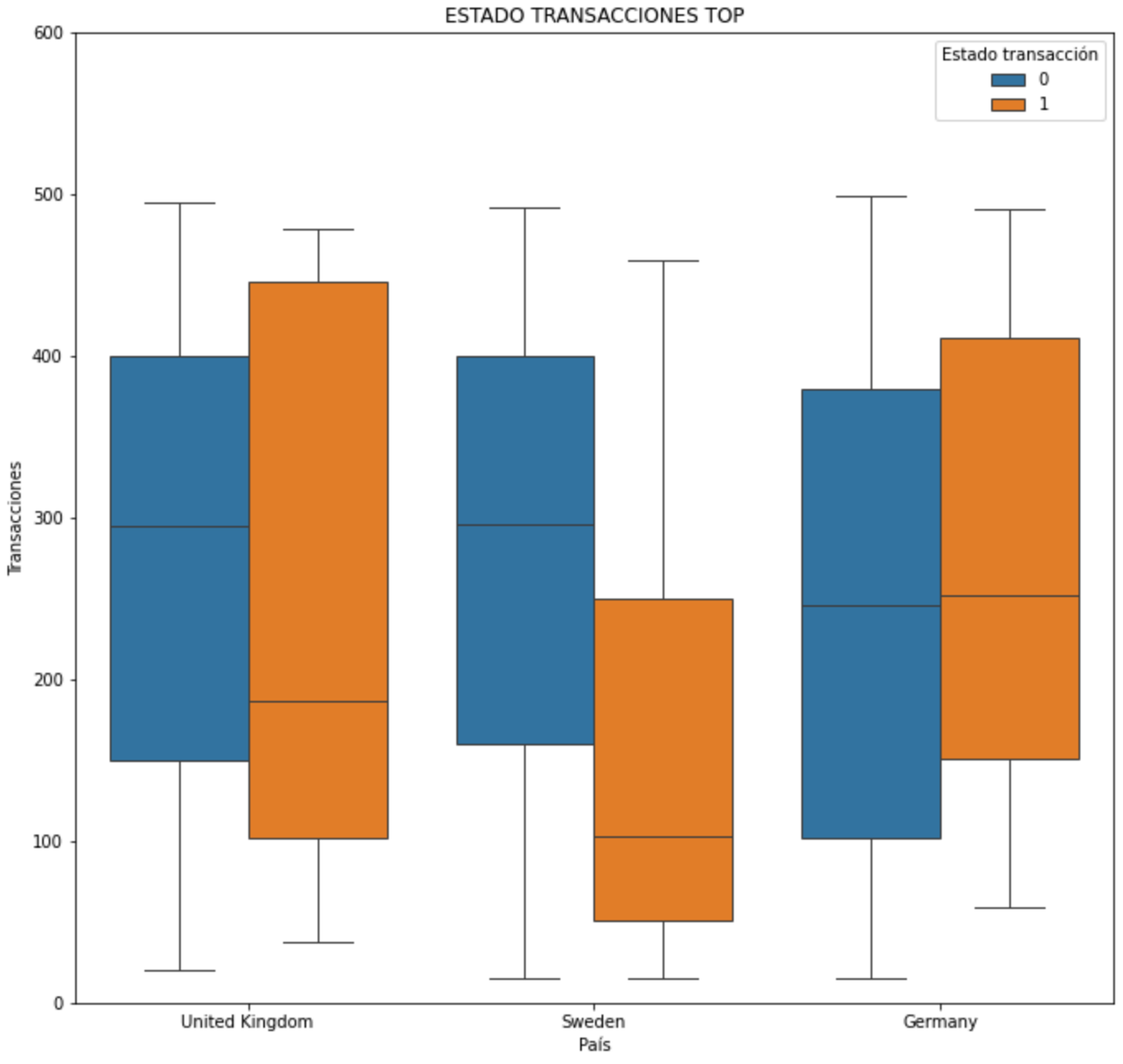


SCRIPT

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

df_15_PIB=dataset

# Agrupo por país y sumo el total de las transacciones según su estado
resumenPaisEstadoTransaciones = df_15_PIB.groupby('country')[['No Declined', 'Declined']].sum()

# Ordeno por la columna "No Declined", de mayor a menor
resumenPaisEstadoTransaciones = resumenPaisEstadoTransaciones.sort_values
                                        (by='No Declined', ascending=False)

#Ejecuto  gráfica de barras
resumenPaisEstadoTransaciones.plot(kind='bar', stacked=False, figsize=(10,6))

# Configuración de las etiquetas de los ejes y de la leyenda
plt.title("ESTADO TRANSACCIONES POR PAÍS")
plt.legend(["aprobada", "rechazada"],bbox_to_anchor=(1, 1), loc='upper right')
plt.xlabel("País")
plt.ylabel("Transacciones");
plt.tight_layout()
plt.show()
```

3 VARIABLES

BOXPLOT - Relacionar las ventas por país en función de si las transacciones han sido declinadas o no (empresas TOP)

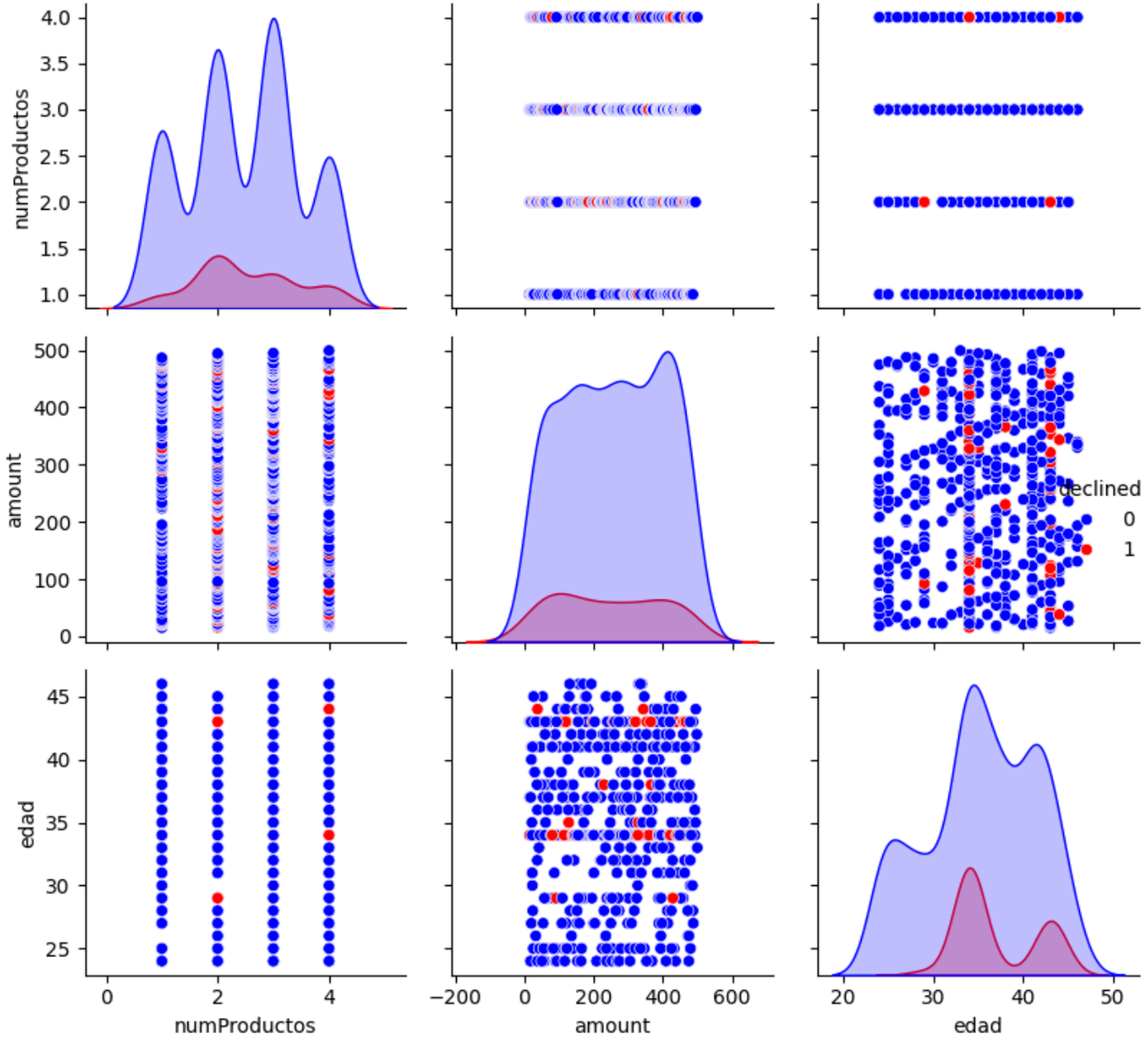


SCRIPT

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

df_paisesTOPFiltrado_PIB=dataset

# Hago un boxplot
sns.boxplot(data=df_paisesTOPFiltrado_PIB, x="country", y="amount", hue="declined")

# Configuración de las etiquetas de los ejes y de la leyenda
plt.title("ESTADO TRANSACCIONES TOP")
plt.legend(title="Estado transacción",bbox_to_anchor=(1, 1), loc='upper right')
plt.xlabel("País")
plt.ylabel("Transacciones")
plt.ylim(0, 600)
plt.tight_layout()
plt.show()
```

MAS DE 3 VARIABLES

PAIRPLOT - Analizar la relación que puede haber entre la edad de los usuarios y el tipo de transacciones que realizan (importe, productos, ...)



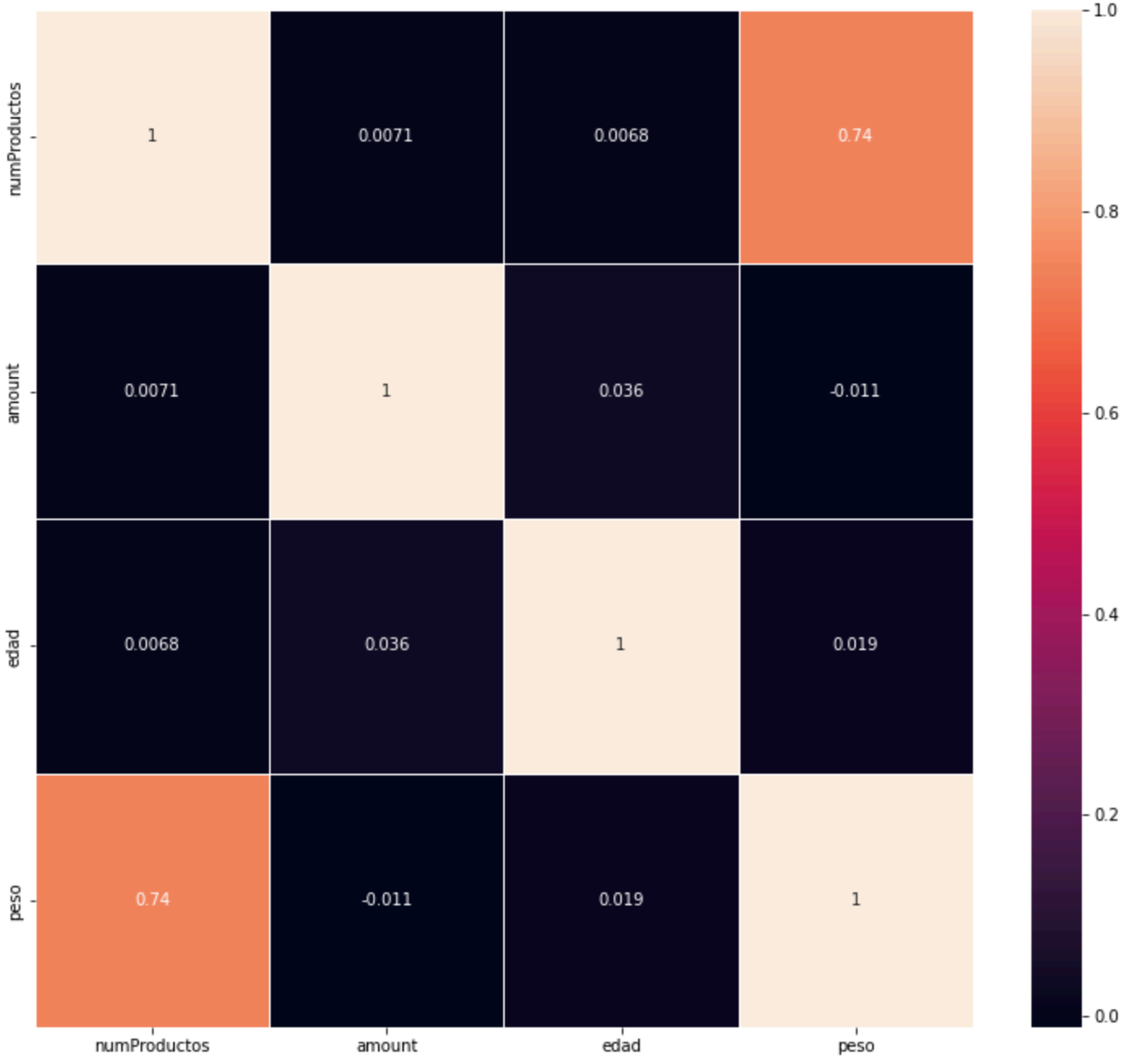
SCRIPT

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

df_datos_interes_PIB=dataset

sns.pairplot(
                df_datos_interes_PIB,
                vars=['numProductos', 'amount', 'edad'],
                hue='declined',
                diag_kind='kde',
                palette=['blue', 'red']
               )
plt.tight_layout()
plt.show()
```

NIVEL 2

CORRELACIÓN DE VARIABLES NUMÉRICAS

HEATMAP

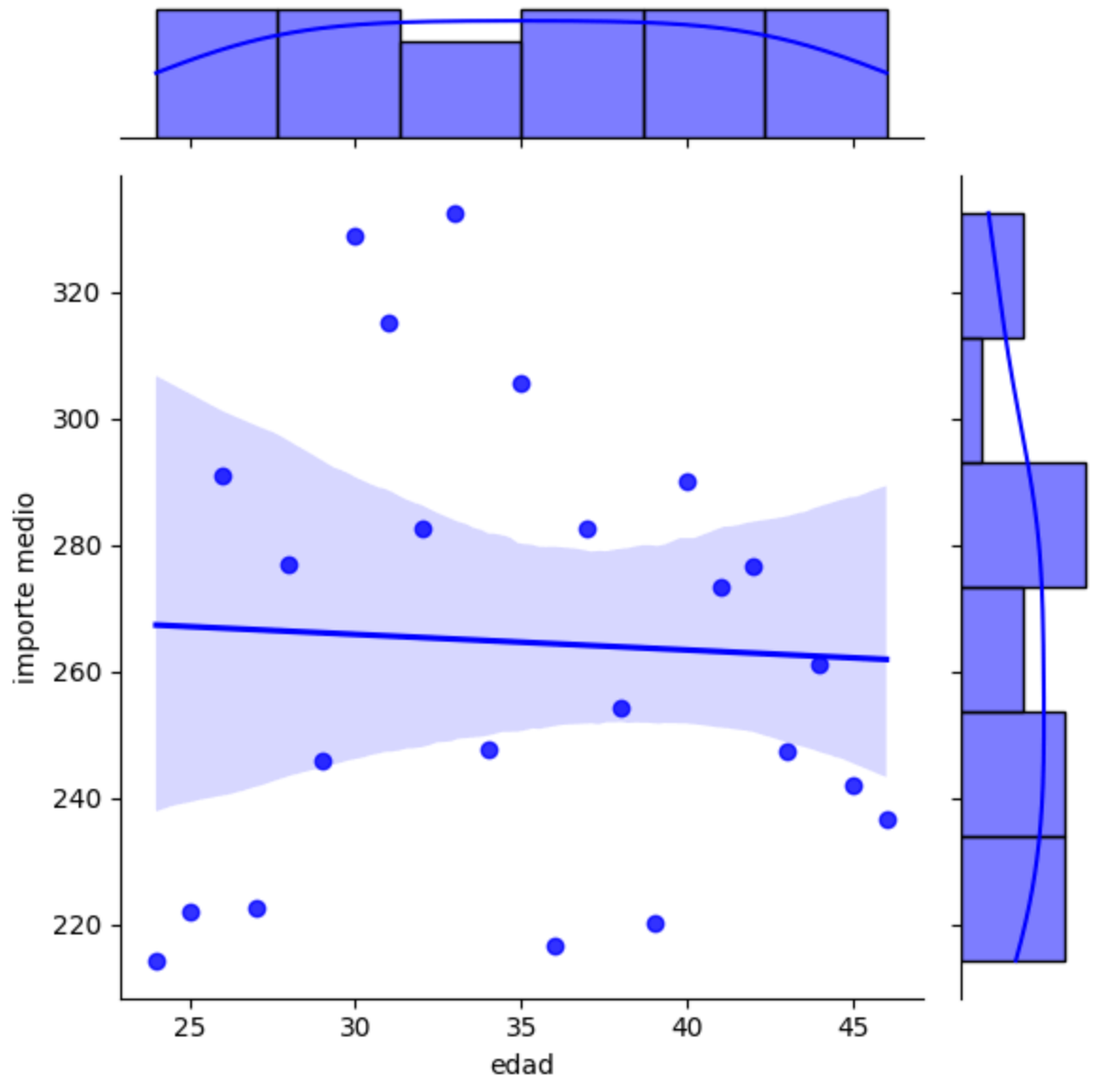


SCRIPT

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

df_transactionsResumenVarNumericas_PIB=dataset

sns.heatmap(
                df_transactionsResumenVarNumericas_PIB[['numProductos',
                'amount',
                'edad','peso']].corr(),
                 annot=True,
                 linewidths=0.5
                 )
plt.tight_layout()
plt.show()
```

JOINPLOT

Relación entre las edades de los usuarios y los importes medios de sus transacciones



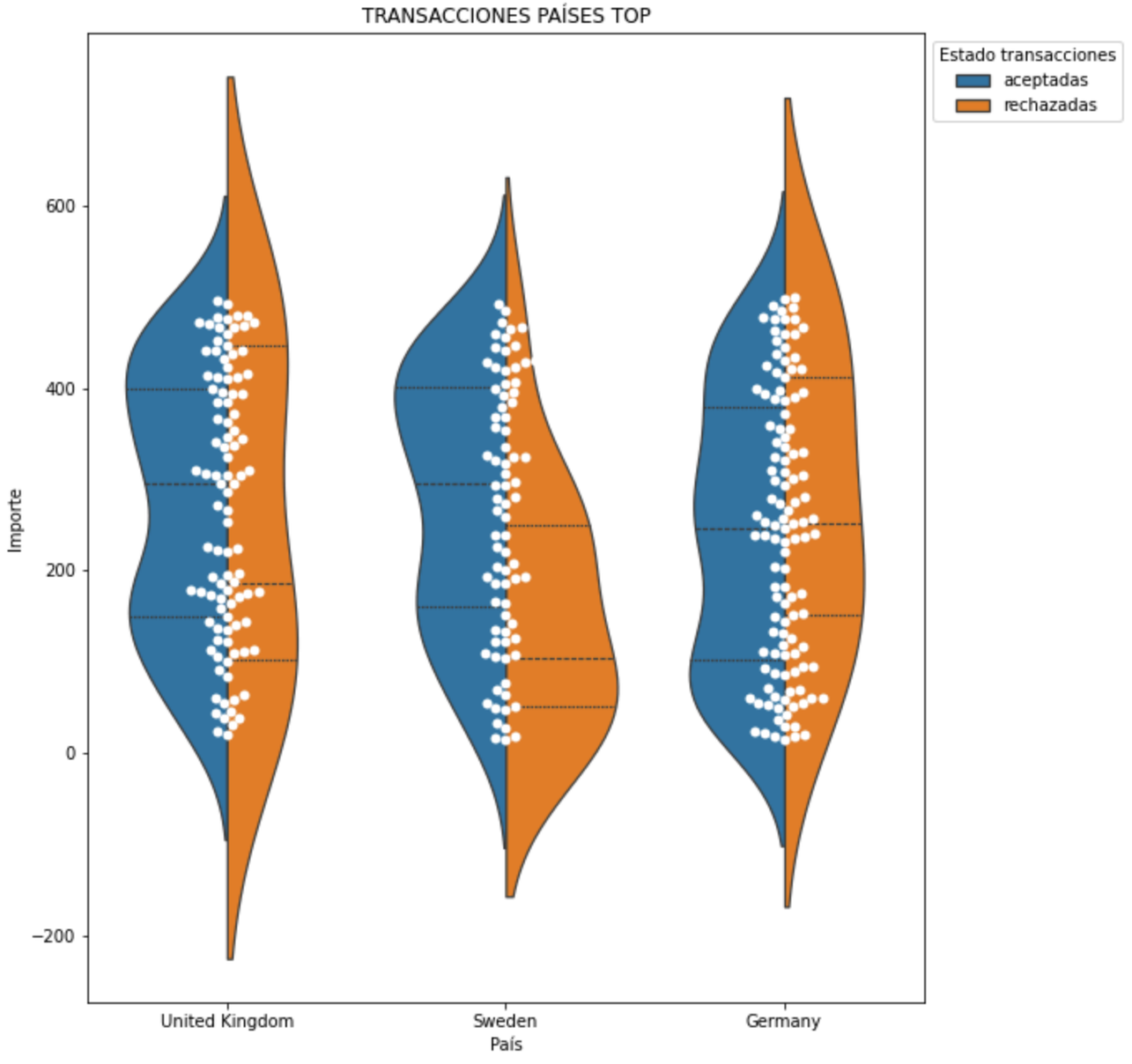
SCRIPT

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

df_edadAmount_PIB=dataset

sns.jointplot(x = "edad", y = "importe medio", data = df_edadAmount_PIB, kind = "reg",color="blue")
plt.tight_layout()
plt.show()
```

NIVEL 3

VIOLINPLOT Y OTRO TIPO DE GRÁFICO (SWARMPLOT)

Relación entre el importe de las transacciones y su estado, para los 3 países TOP en importe de transacciones.
Superpuesto con el estado de las transacciones a nivel individual.

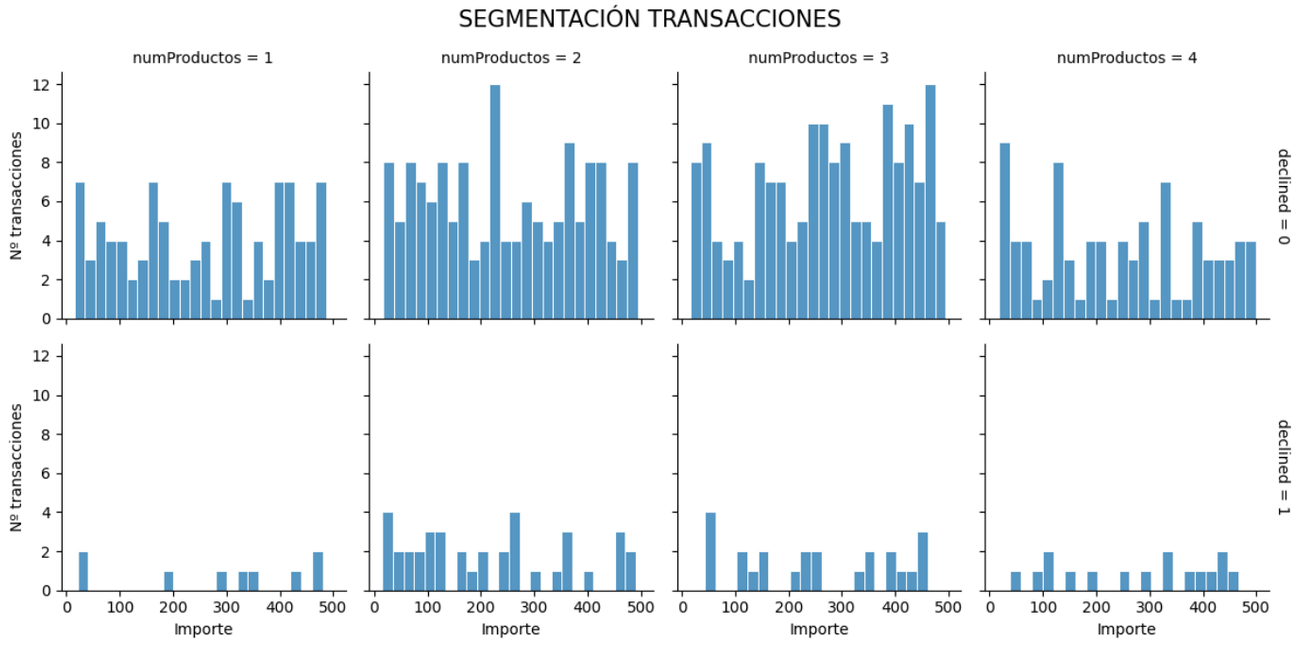


SCRIPT

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

df_paisesTOPResumido_PIB=dataset

# Saco la gráfica
ax = sns.violinplot(
      y=df_paisesTOPResumido_PIB['amount'],
      x=df_paisesTOPResumido_PIB['country'],
      hue=df_paisesTOPResumido_PIB['declined'].replace({0 : "aceptadas", 1 : "rechazadas"}),split=True,
      inner="quartile"
      )

# superpongo un swarmplot para que los puntos no se superpongan y se entienda un poco mejor
sns.swarmplot(
                  y="amount", x="country",
                  data=df_paisesTOPResumido_PIB,
                  color="white",
                  edgecolor="auto", s=6
                  )

# Configuración de las etiquetas de los ejes y de la leyenda
plt.title("TRANSACCIONES PAÍSES TOP")
plt.legend(title="Estado transacciones",bbox_to_anchor=(1, 1), loc='upper left',)
plt.xlabel("País")
plt.ylabel("Importe")
plt.tight_layout()
plt.show()
```

VISUALIZAR MÚLTIPLES ASPECTOS DE DATOS SIMULTÁNEAMENTE

FACETGRID - Visualizar a la vez según el nº de productos por transacción y si estás han sido aceptadas o rechazadas, el importe y el nº de las mismas



SCRIPT

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

df_transactionsResumenVarNumericas_PIB=dataset

# Implemento la gráfica
df_varNumericas = sns.FacetGrid(
                                df_transactionsResumenVarNumericas_PIB,
                                col='numProductos', row='declined',
                                margin_titles=True
                                )
df_varNumericas.map_dataframe(sns.histplot, x='amount', binwidth=20,bins=30,edgecolor="white");

# Configuración de las etiquetas de los ejes y de la leyenda
df_varNumericas.fig.suptitle("SEGMENTACIÓN TRANSACCIONES", fontsize=15)
df_varNumericas.fig.subplots_adjust(top=.88)
df_varNumericas.set_axis_labels('Importe', 'Nº transacciones')
plt.tight_layout()
plt.show()
```