Tarea 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.

Entrega:

Almacena en un repositorio de tu GitHub una carpeta que contenga:

- El archivo .pbix de Power BI con el informe solicitado.
- Un PDF del informe exportado.
- Un archivo .ipynb de Jupyter Notebook que contenga cada script correspondiente a cada visualización del informe junto al enunciado correspondiente.

Nivel 1

Los 7 ejercicios del nivel 1 de la tarea 01

Ejercicio 1
Una variable numérica.

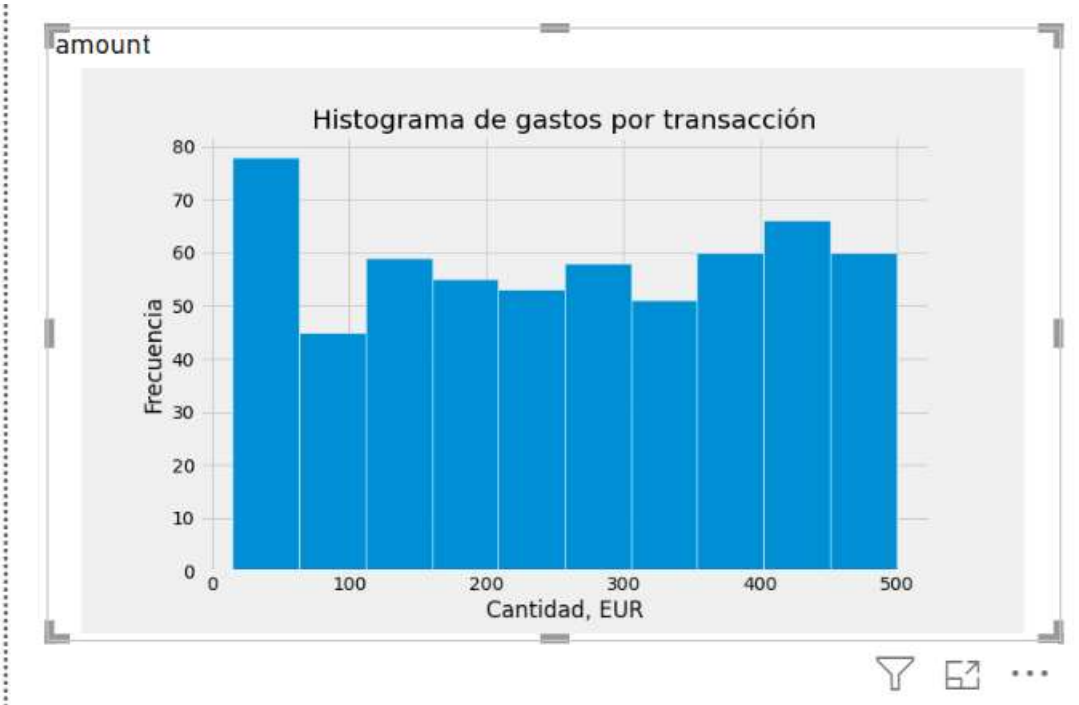
Procedimiento seguido:

- Se importan en Power BI las tablas deseadas. Con esto se evita la necesidad de acceder a la base de datos con un conector de Python.
- Se crea un objeto de tipo Python
- Se arrastra en este objeto el campo ‘amount’ de la tabla ‘transactions’. Al hacer esto se genera automáticamente en segundo plano un archivo csv y un dataframe de pandas, que se pueden ver al editar el código en VScode:
- dataset = pandas.read_csv('input_df_a8a8192f-07b7-4329-a75e-f43713f6afdb.csv')
- En el editor de scripts se introduce y ejecuta el siguiente script, similar al original, salvo la linea data = dataset['amount'] que se usa para acceder al campo deseado y la linea matplotlib.pyplot.figure(figsize=(10,6), dpi=72) que se usa para ajustar el tamaño del gráfico:

```
import matplotlib.pyplot as plt
data = dataset['amount']

matplotlib.pyplot.figure(figsize=(10,6), dpi=72)
plt.style.use('fivethirtyeight')
```

```
plt.hist(data, bins=10, edgecolor='white')
plt.xlabel('Cantidad, EUR')
plt.ylabel('Frecuencia')
plt.title('Histograma de gastos por transacción')

plt.show()
```

El resultado es:



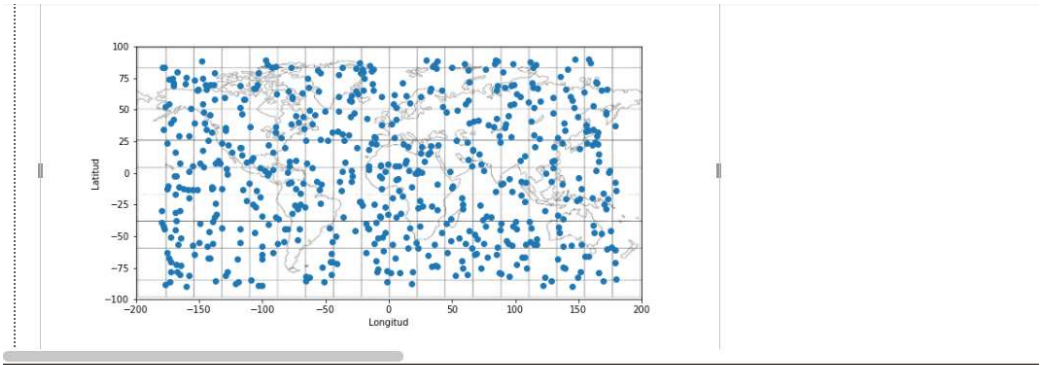
Editor de scripts de Python

⚠ Las filas duplicadas se quitarán de los datos.

```
1  import matplotlib.pyplot as plt
2  cantidad = dataset['amount']
3
4  matplotlib.pyplot.figure(figsize=(10,6), dpi=72)
5  plt.style.use('fivethirtyeight')
6  plt.hist(cantidad, bins=10, edgecolor='white')
7  plt.xlabel('Cantidad, EUR')
8  plt.ylabel('Frecuencia')
9  plt.title('Histograma de gastos por transacción')
10
11 plt.show()
```

Ejercicio 2
Dos variables numéricas.

Se sigue el mismo procedimiento del Ejercicio 1 con la única diferencia que se arrastran las variables lat y longitude y se modifica el código original para acceder a ambas ellas. El resultado es:



Editor de scripts de Python

```python
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

fondo = mpimg.imread('C:\Python & datascience\Cursos\Data Analysys especialización - ITactiva\Tarea S8.02. Power BI con Python\map.jpg')

latitud = dataset['lat']
longitud = dataset['longitude']
matplotlib.pyplot.figure(figsize=(10,5), dpi=72)
plt.imshow(fondo, extent=[-200, 200, -100, 100])

plt.scatter(longitud, latitud)

plt.xlabel('Longitud')
plt.ylabel('Latitud')
plt.grid(False)

plt.show()
```

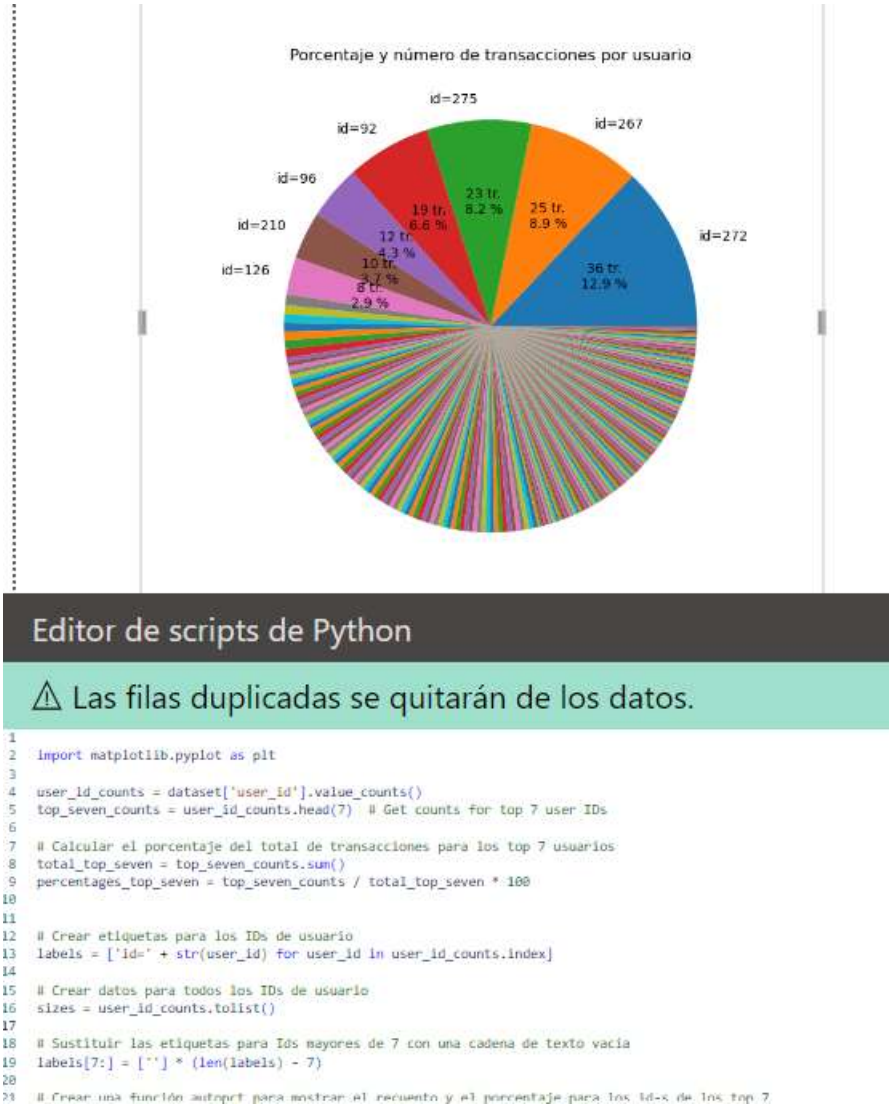
Ej. 3 Una variable categórica.

Power BI elimina por defecto las filas duplicadas. La variable categórica escogida (user_id) tiene duplicados y de hecho el objetivo es contar el número de repeticiones y representarla en un piechart. Para evitar la eliminación de duplicados de este campo, se ha añadido otro campo (id) cuyos valores son todos diferentes, de modo que ya no hay filas duplicadas y se mantienen todos los id-s de usuario. El código introducido ha sido idéntico al original, salvo el cambio del nombre del dataframe de df a dataset:

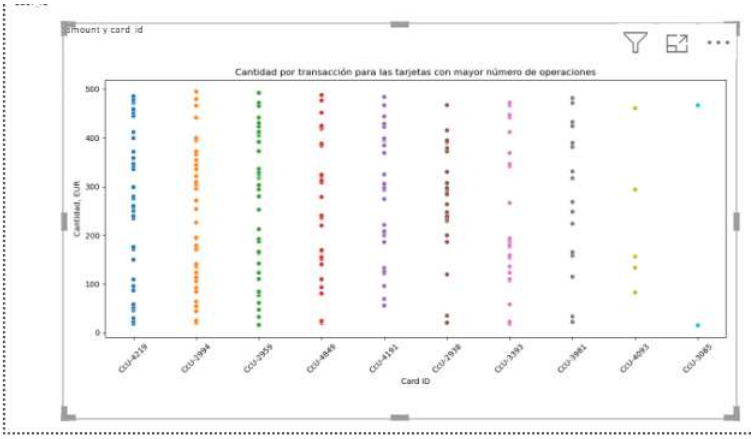
```
import matplotlib.pyplot as plt

user_id_counts = dataset['user_id'].value_counts()
top_seven_counts = user_id_counts.head(7)

# Calcular el porcentaje del total de transacciones para los top 7 usuarios
total_top_seven = top_seven_counts.sum()
percentages_top_seven = top_seven_counts / total_top_seven * 100


# Crear etiquetas para los IDs de usuario
labels = ['id=' + str(user_id) for user_id in user_id_counts.index]

# Crear datos para todos los IDs de usuario
sizes = user_id_counts.tolist()

# Sustituir las etiquetas para Ids mayores de 7 con una cadena de texto vacía
labels[7:] = [''] * (len(labels) - 7)
```

```python
# Crear una función autopct para mostrar el recuento y el porcentaje para los id-s de los top 7
def autopct_func(pct):
    if pct > 2.8:
        count = total_top_seven * pct / 100
        return '{:.0f} tr.\n{:.1f} %'.format(count, pct)
    else:
        return ''

plt.figure(figsize=(7, 7))
plt.pie(sizes, labels=labels, autopct=autopct_func)
plt.title('Porcentaje y número de transacciones por usuario')

plt.show()
```

De este modo el resultado obtenido es idéntico al obtenido en S8.01:



Ejercicio 4 Una variable categórica y una numérica.



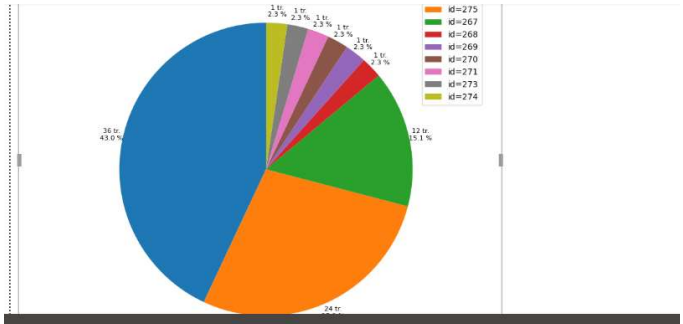
Editor de scripts de Python

⚠ Las filas duplicadas se quitarán de los datos.

```python
#Debido a que en Tarea S8.02. no se puede representar el plot interractivo,
# se genera un plot no interractivo, solo con los card_id-s que tienen el mayor número operraciones.

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

#Top N card_id-s por número de transacciones
top_n = 10
top_card_ids = dataset['card_id'].value_counts().nlargest(top_n).index

# Filtrado del DataFrame dejando solo las filas que corresponden a los top N
df_top_n = dataset[dataset['card_id'].isin(top_card_ids)]
df_top_n
# Número de puntos(transacciones) por cada card_id
```

Ejercicio 5 Dos variables categóricas.

En este caso también se arrastra un campo extra para evitar la eliminación de los duplicados.
De nuevo, se utiliza el código original, sustituyendo el nombre del dataframe a ‘dataset’.
El resultado es:



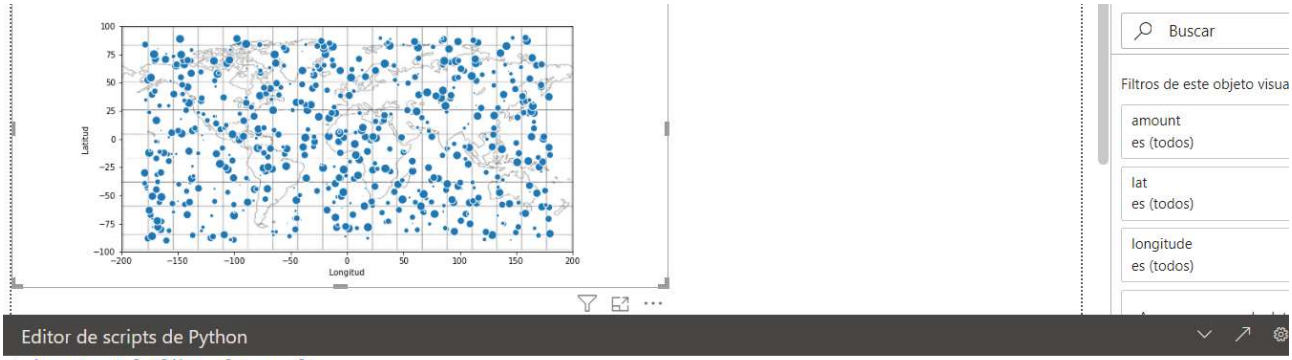
Editor de scripts de Python

⚠ Las filas duplicadas se quitarán de los datos.

```python
import matplotlib.pyplot as plt

# Recuento de transacciones rechazadas
user_id_counts = dataset.loc[dataset['declined'] == 1, 'user_id'].value_counts()

# Etiquetas para los user_id
labels = ['id=' + str(user_id) for user_id in user_id_counts.index]

#Etiquetas con porcentaje y numero de transacciones rechazadas
def autopct_func(pct):
    count = int(pct / 100 * user_id_counts.sum())
    return '{:.0f} tr.\n{:.1f} %'.format(count, pct)
```

Ejercicio 6 Tres variables.

Se emplea el código original de la tarea 8.01, sin la parte de conversión de object a float, ya que al tomar los datos del dataframe llamado dataset ya están en el formato correcto. El resultado es:

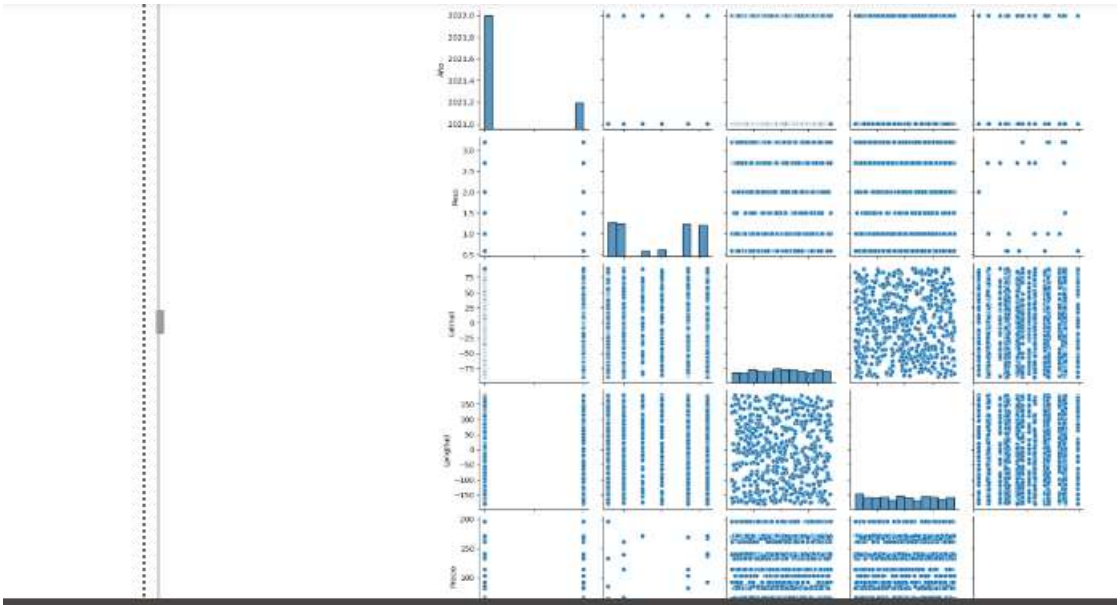


Editor de scripts de Python

```python
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pandas as pd

fondo = mpimg.imread('C:\Python & datascience\Cursos\Data Analysys especialización - ITactiva\Tarea S8.02. Power BI con Python\map.jpg')
matplotlib.pyplot.figure(figsize=(10,5), dpi=72)
plt.imshow(fondo, extent=[-200, 200, -100, 100])

plt.scatter(dataset['longitude'], dataset['lat'], s=dataset['amount']/4, edgecolor='white')

plt.xlabel('Longitud')
plt.ylabel('Latitud')
plt.grid(False)
plt.show()
```

Ejercicio 7 Graficar un Pairplot.

Para que funcione el código original que genera el pairplot, tan solo hay que eliminar el símbolo de $ del campo de precio y convertir el campo a numérico. También se cambian los nombres de los campos que están en inglés para que todos estén en castellano.



Editor de scripts de Python

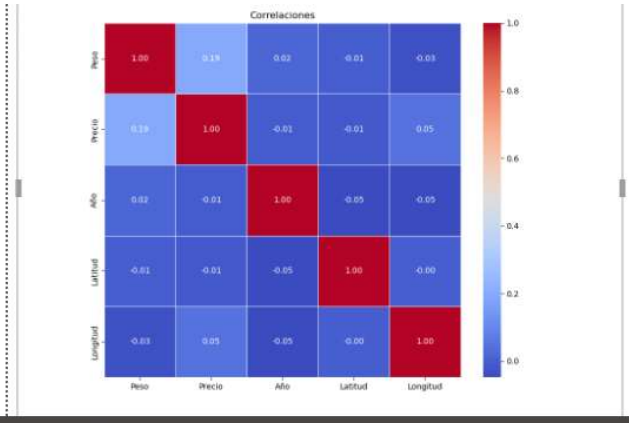
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

dataset['Precio'] = dataset['price'].str.replace('$', '')
dataset['Precio'] = dataset['Precio'].apply(pd.to_numeric)


sns.pairplot(dataset)
plt.show()
```

Nivel 2
Los 2 ejercicios del nivel 2 de la tarea 01

Ejercicio 1 Correlación de todas las variables numéricas.

La variable precio que está en formato texto se ha convertido en número mediante Power Query, ya que la conversión con Python fallaba por alguna razón difícil de aclarar.

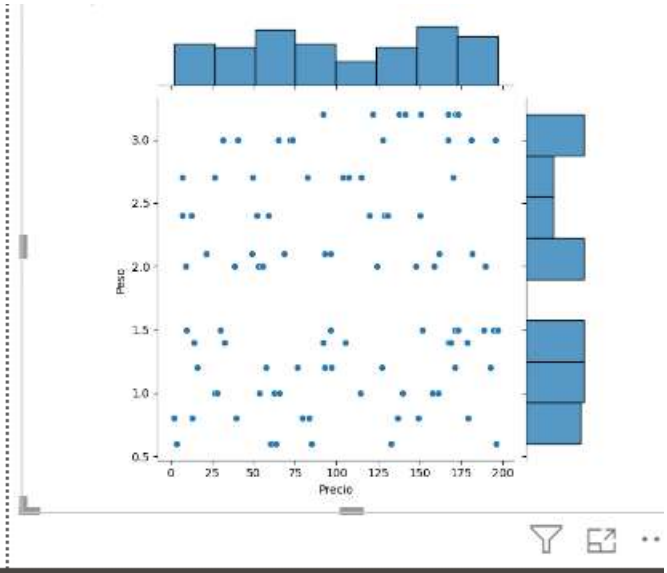
El resultado es:



Editor de scripts de Python

```python

import seaborn as sns
import matplotlib.pyplot as plt

corr_matrix=dataset.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlaciones')
plt.show()
```

Ejercicio 2 Implementa un jointplot.

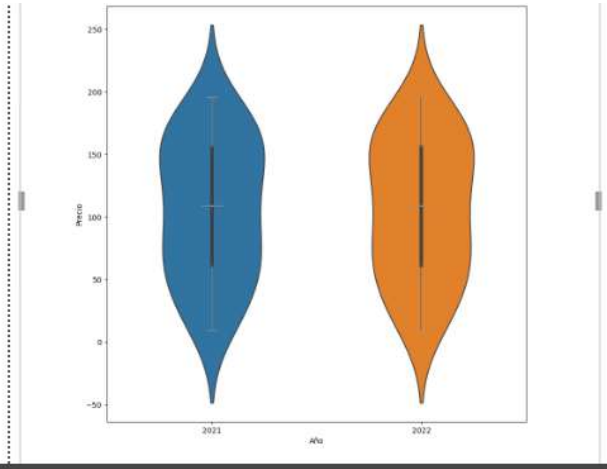


Editor de scripts de Python

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

sns.jointplot(x='Precio', y='Peso', data=dataset)

plt.show()
```

Nivel 3
Los 2 ejercicios del nivel 3 de la tarea 01

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



Editor de scripts de Python

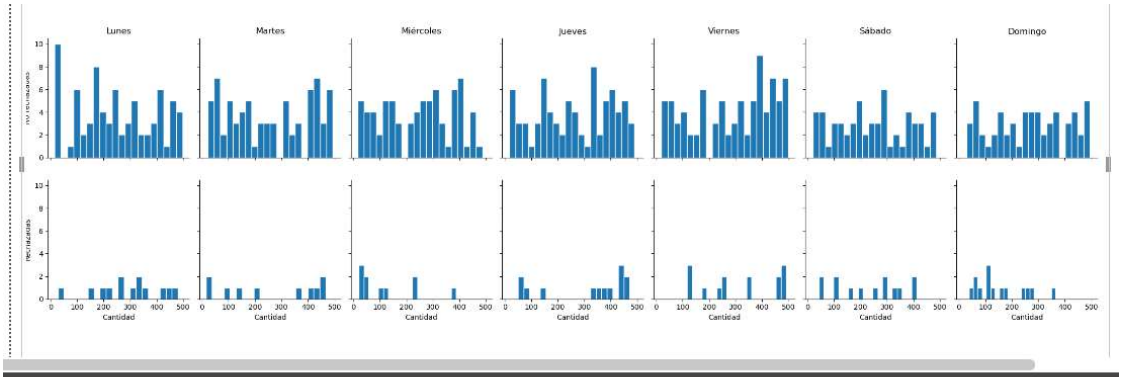
⚠ Las filas duplicadas se quitarán de los datos.

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

plt.figure(figsize=(10, 10))
sns.violinplot(x='Año', y='Precio', data=dataset, width=0.5)
sns.boxplot(x='Año', y='Precio', data=dataset, width=0.1, color='white', linewidth=1)

plt.show()
```

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

Primero se crea un campo de día de la semana mediante Power Query, ya que el dataframe llamado dataset contiene los elementos de la jerarquía y no el propio timestamp que se había empleado en el código original. El código M usado para generar el campo nuevo es el siguiente:
= Date.DayOfWeekName([Timestamp])

A continuación se adapta el código a los nombres de los campos y del dataframe (dataset). El resultado es:



Editor de scripts de Python

⚠ Las filas duplicadas se quitarán de los datos.

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

dataset['Rechazada'] = dataset['declined'].map({0: 'No', 1: 'Sí'})

g = sns.FacetGrid(dataset, col="Día de la semana", row='Rechazada')

g.map(plt.hist, "Cantidad", bins=20, edgecolor='white')

g.set_titles("", "")

for ax, label in zip(g.axes[:, 0], ['No rechazadas', 'Rechazadas']):
    ax.set_ylabel(label)
```