## Sprint 8.2

Para poder realizar estos ejercicios, tuve que cambiar la conexión que usé en el sprint 8.1 pues no me conectaba a través de mysql.connector.
Lo que pasé a usar es:

```
import pandas as pd
from sqlalchemy import create_engine

connection =
create_engine(f'mysql+mysqlconnector://root:Mysql-1904@localhost/sprint
4')


tablas_db =
['transactions','companies','credit_card','user','estado_tarj',
'products']
df = {}

for tabla in tablas_db:
    query = f'SELECT * FROM {tabla}'
    df[tabla] = pd.read_sql(query, connection)

connection.dispose()


print(df)
```

Para que funcione tuve que llamar a cada dataframe como una nueva variable.

```
transaction = df['transactions']
companies = df['companies']
credit_card = df['credit_card']
user = df['user']
estado_tarj = df['estado_tarj']
products = df['products']
```

Al hacer la conexión y abrirla mediante Powerbi me llamó la atención que el formato de “amount” era de texto. Para poder convertirlo tuve que reemplazar puntos por comas y luego convertirlo a número. Lo mismo con “price”. Ambos cambios se realizaron mediante transformación de datos en powerquery.

Es importante chequear toda la información cuando se la abre desde el Powerbi pues sino los gráficos no reflejarán la misma información que hemos visto en Python.

Descripción de la tarea:

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.

## - Exercici 1

Una variable numèrica.

Esta primera parte de código la he copiado en cada gráfico, ya sea que en el ejercicio se utilice o no.

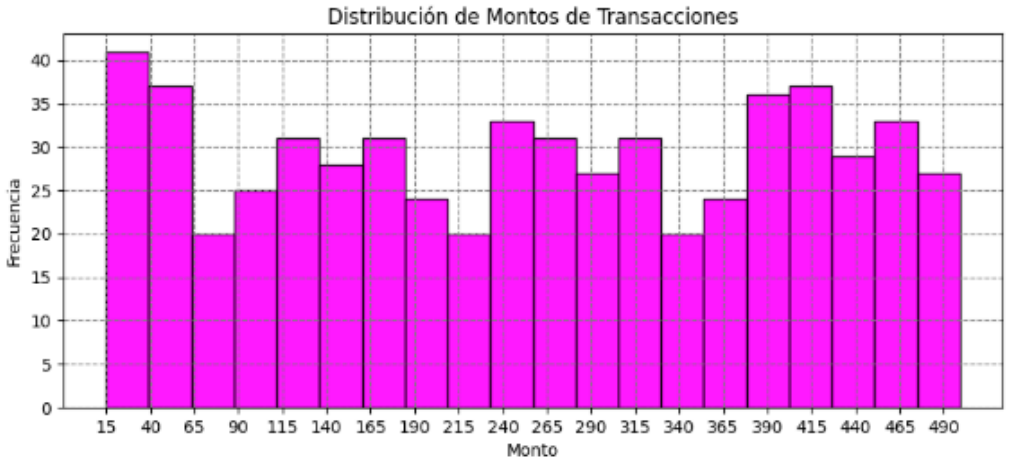
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import squarify
import plotly.express as px
```

Luego, en cada ejercicio he probado primero copiando el código desde Python (y cambiando por “dataset” todos los recorridos que llevaban hasta el origen de los datos). En algunos casos he tenido que realizar otras modificaciones pues los gráficos no se representan exactamente igual en una y otra herramienta.

En este caso no tuve que agregar nada al código:

```
plt.figure(figsize=(10, 4))
plt.hist(dataset, bins=20, edgecolor='k', color= "magenta", alpha=0.9)
plt.title('Distribución de Montos de Transacciones')
plt.xlabel('Monto')
plt.ylabel('Frecuencia')
plt.grid(True)
plt.grid(linestyle = "dashed", color = "grey")
plt.xticks(np.arange(min(dataset['amount']),
max(dataset['amount'])+1,25))
plt.show()
```

Nivel 1 - Ejercicio 1



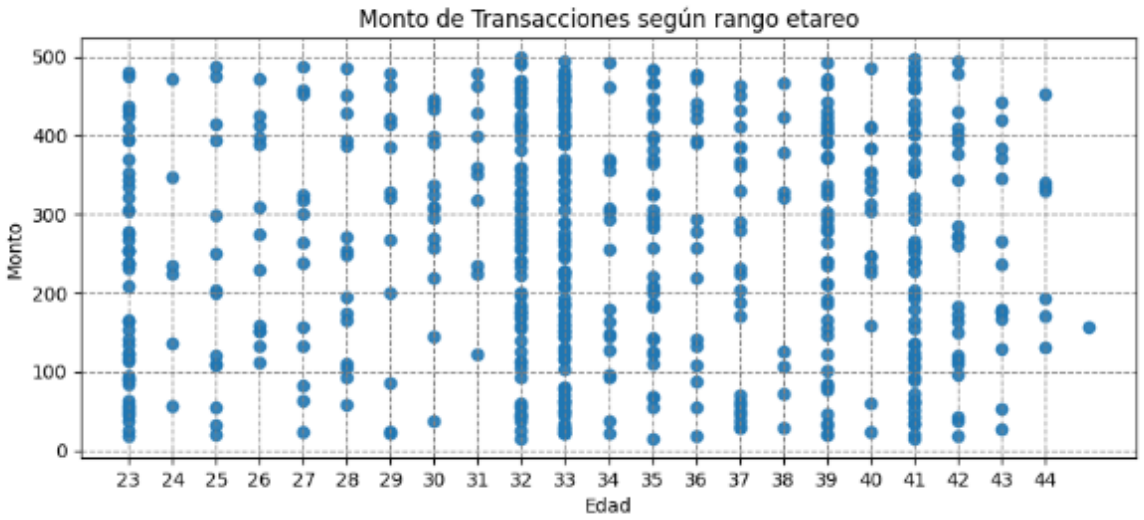

## - Exercici 2

Dues variables numèriques.

En esta oportunidad no he tenido que hacer el “merge” que he realizado en Python pues el programa lo realiza automáticamente.

```
plt.figure(figsize=(10, 4))
plt.scatter(dataset['edad_usuario'], dataset['amount'], alpha=0.9,)
plt.title('Monto de Transacciones según rango etareo')
plt.xlabel('Edad')
plt.ylabel('Monto')
plt.grid(True)
plt.grid(linestyle = "dashed", color = "grey")
plt.xticks(np.arange(min(dataset['edad_usuario']),
max(dataset['edad_usuario'])+0,1))
plt.show()
```

Nivel 1 - Ejercicio 2

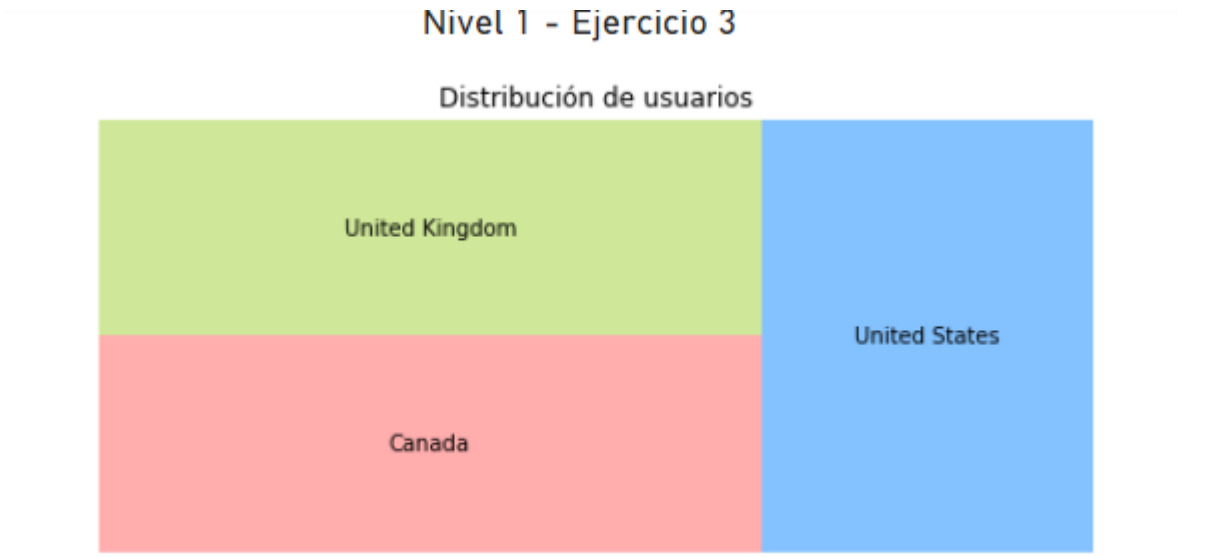



## - Exercici 3

Una variable categòrica.

En esta oportunidad no he cambiado nada pero el sistema ha cambiado solo los colores según el orden que ha establecido como prioritario. Considero que esto no altera las conclusiones que pueda sacar del gráfico, con lo cual he decidido dejarlo así.

```
country_counts = dataset['country'].value_counts()
colors = ['#ff9999','#c4e17f','#66b3ff']
squarify.plot(sizes=country_counts.values, label=country_counts.index,
alpha=0.8, color=colors[:len(country_counts)])
plt.title('Distribución de usuarios')
plt.axis('off')
plt.show()
```




## - Exercici 4

Una variable categòrica i una numèrica.

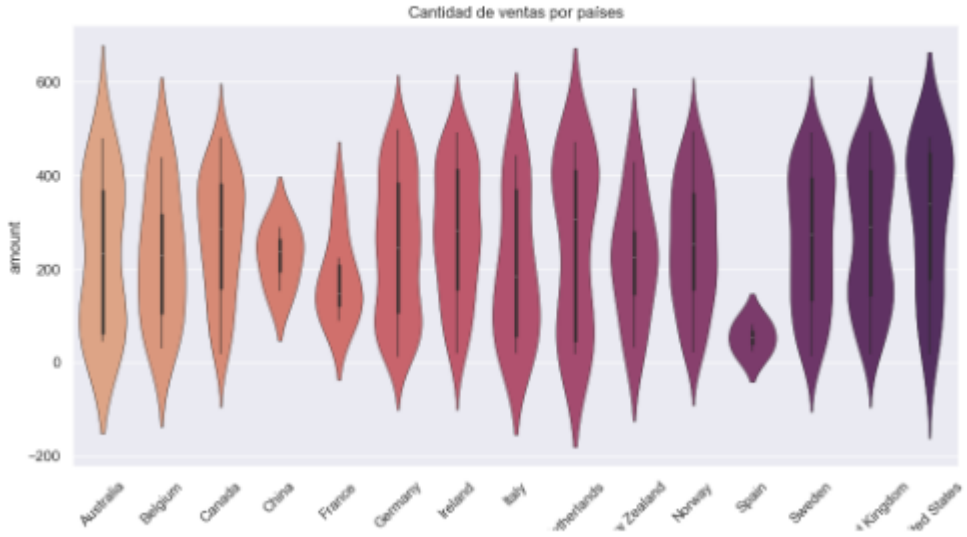
En esta oportunidad al traer el código de Python no se mostraba el gráfico, por lo que tuve que investigar el motivo. Principalmente era porque traía un error al cruzar los datos. Por eso tuve que probar imprimiendo la información.

```
print(dataset.info())
print(dataset.head())
```

Tampoco tuve que hacer el merge en este caso. Y aquí nuevamente el orden en el que se muestra el gráfico no es el mismo que en Python, pero la información sí.

```
plt.figure(figsize=(12,6))
sns.set_theme(style="darkgrid")

violin = sns.violinplot(x=dataset["country"], y=dataset["amount"],
linewidth=0.5, palette='flare', hue=dataset["country"], legend=False)
plt.title('Cantidad de ventas por países')
plt.xticks(rotation=45)
plt.show()
```

## Nivel 1 - Ejercicio 4




# - Exercici 5

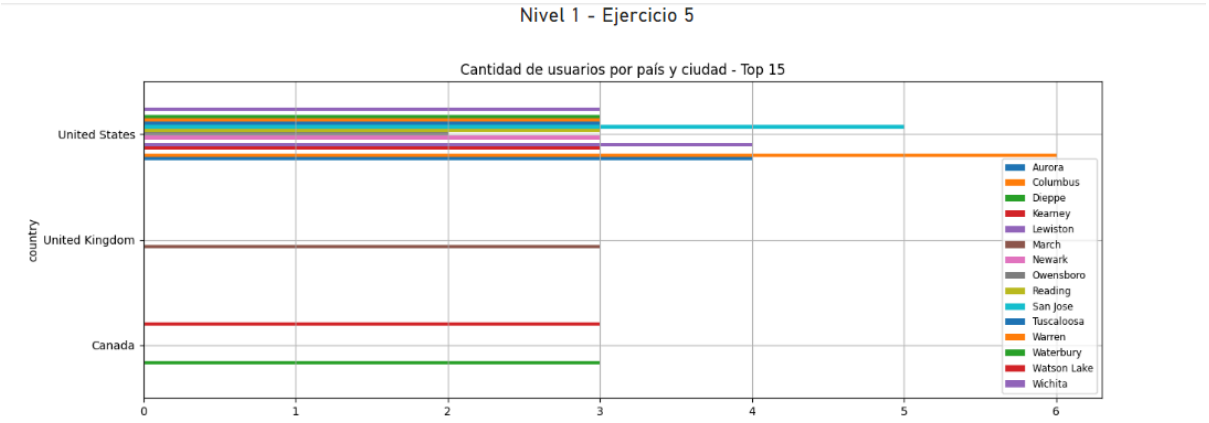
Dues variables categòriques.

Aquí cambié la forma de llamar a la nueva variable que agrupa para que siga con la línea de dataset.
También, como no podía utilizar el treemap de la librería Plotly (no está soportada por Power Bi), el gráfico no es el mismo que el que se muestra en Python.

Para este nuevo gráfico he utilizado las mismas variables, pero he mostrado la cantidad de usuarios por país y ciudad, mostrando el top 15 (con mayores usuarios).
Las conclusiones que se obtuvieron en Python pueden mantenerse en Power Bi.

```
dataset_counts = dataset.groupby(['country',
'city']).size().reset_index(name='count')
dataset = dataset_counts.sort_values('count').tail(15)

pivot_df = dataset.pivot(index='country', columns='city',
values='count')

ax = pivot_df.plot.barh(grid=True,figsize=(15,5))

# Add legend
plt.legend(loc='best', fontsize='small')

#Add title and label
```

```
ax.set_title('Cantidad de usuarios por país y ciudad - Top 15')

plt.show()
```



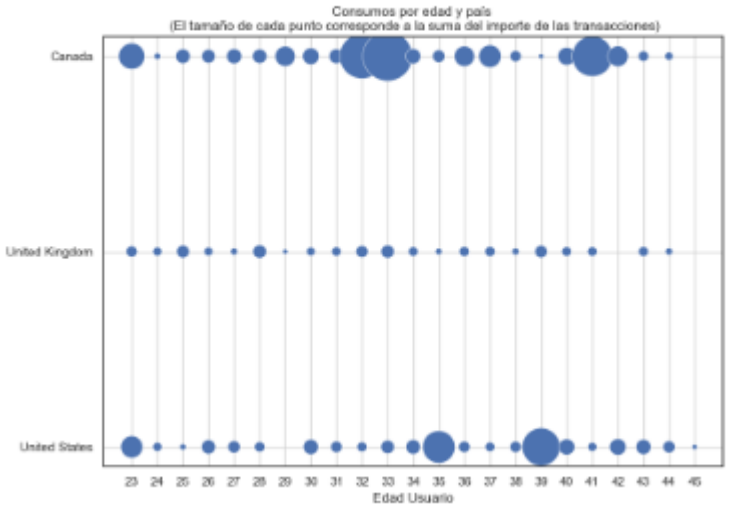

## - Exercici 6

Tres variables.

En esta oportunidad nuevamente no he realizado el merge por no ser necesario. La clave en este caso está en las variables que coloco en el cuadro de datos, pues hace la agregación automáticamente.

```
plt.figure(figsize=(10, 7))
sns.set_theme(style="white")
sns.scatterplot(data=dataset, x="edad_usuario", y="country",
size="amount", legend=False, sizes=(20, 2000))
plt.title('Consumos por edad y país \n (El tamaño de cada punto
corresponde a la suma del importe de las transacciones)')
plt.xlabel('Edad Usuario')
plt.ylabel('País')
plt.xticks(np.arange(min(dataset['edad_usuario']),
max(dataset['edad_usuario'])+1))
plt.grid(True)
plt.show()
```

## Nivel 1 - Ejercicio 6

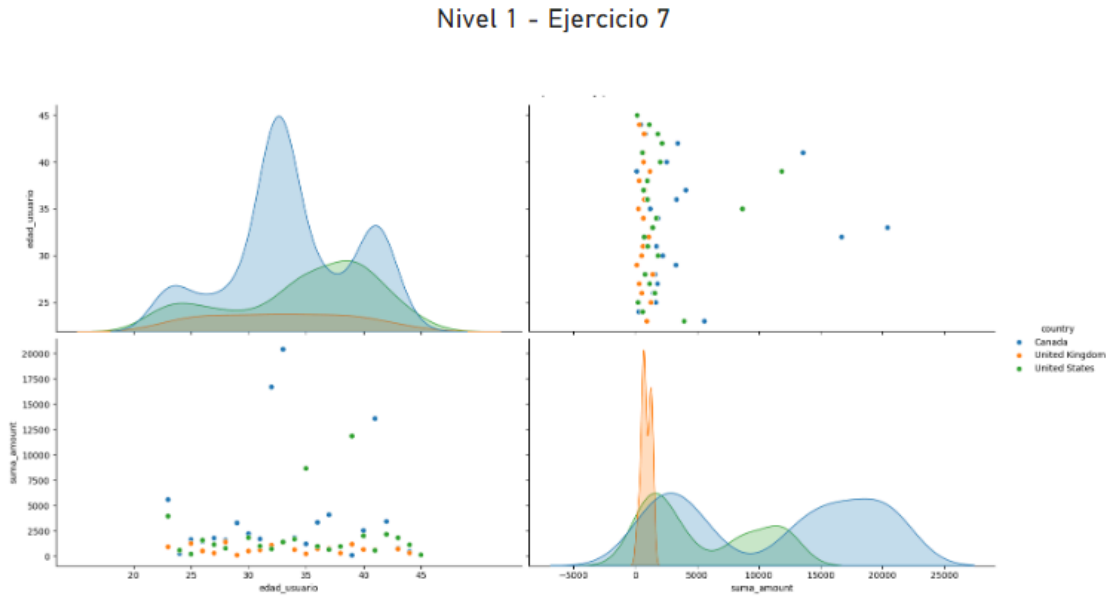



# - Exercici 7

Graficar un Pairplot.

Aquí realicé por fuera el cálculo que en el ejercicio anterior se realizaba automáticamente. El resto continúa igual.

```
dataset['suma_amount'] = dataset.groupby(['edad_usuario',
'country'])['amount'].transform('sum')
g = sns.pairplot(dataset[['edad_usuario', 'suma_amount', 'country']],
hue='country', height=4, aspect=2)
for ax in g.axes.flatten():
    if ax.get_xlabel() == 'edad_usuario':
        x_min, x_max = ax.get_xlim()
        new_xticks = range(int(20), int(45) + 5, 5)
        ax.set_xticks(new_xticks)
plt.suptitle('Consumos por edad y país', y=1.02)
plt.grid(True)
plt.show()
```



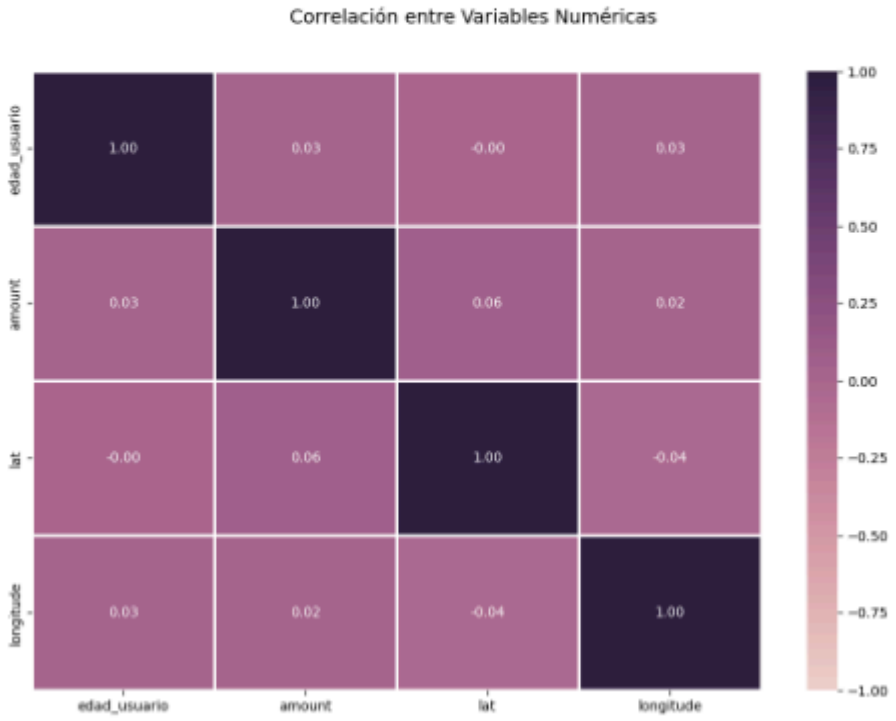

NIVEL 2

# Exercici 1

Correlació de totes les variables numèriques.

```
correlation_matrix =dataset.corr()
plt.figure(figsize=(12,8))
mapa = sns.heatmap(correlation_matrix,
cmap=sns.cubehelix_palette(as_cmap=True), annot=True, fmt='.2f', vmin=
-1, vmax= 1, center= 0, linewidths= 0.8)
mapa.figure.subplots_adjust(top=0.9)
mapa.figure.suptitle('Correlación entre Variables Numéricas',
fontsize=14)

plt.show()
```

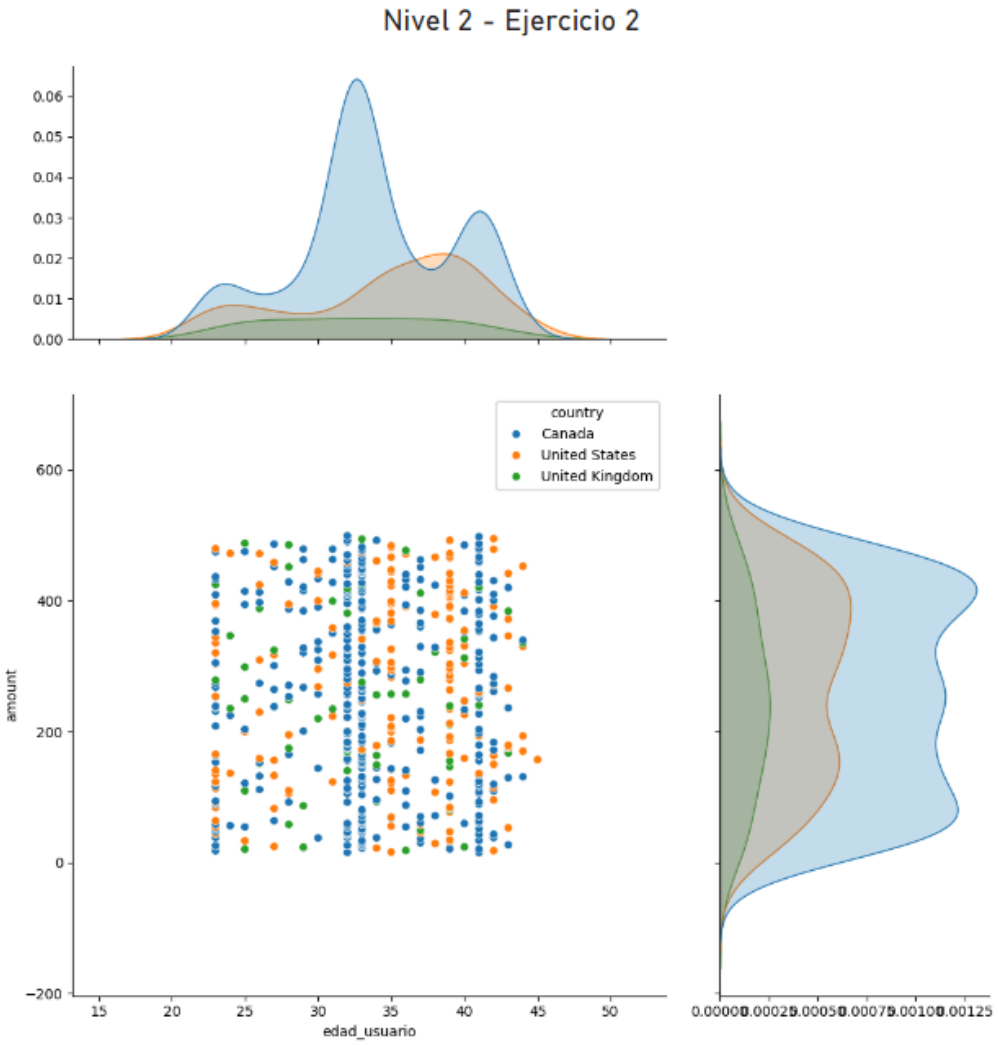
Nivel 2 - Ejercicio 1




## Exercici 2

Implementa un jointplot.

```
sns.jointplot(data=dataset, x="edad_usuario", y="amount",
hue="country", kind="scatter", height=10, ratio=2, marginal_ticks=True)
plt.show()
```



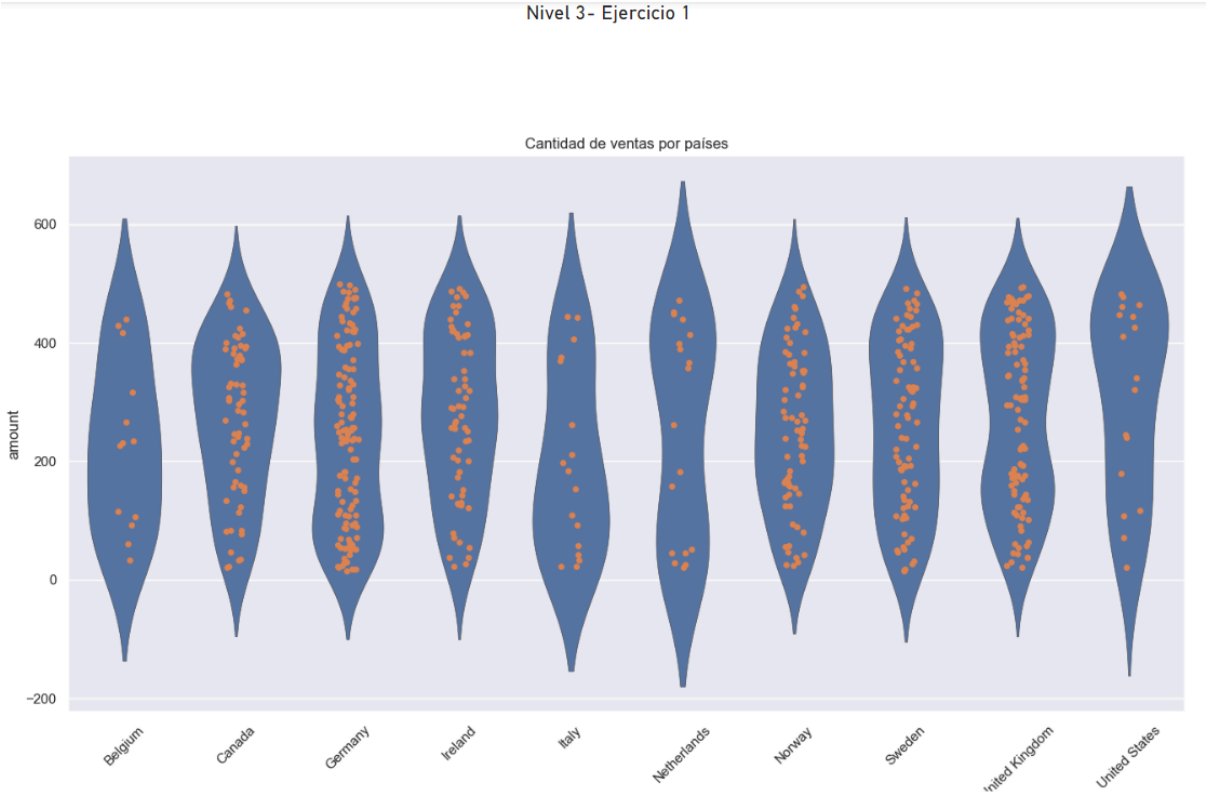

NIVEL 3

# Exercici 1

Implementa un violinplot combinat amb un altre tipus de gràfic.

```
top_countries = dataset['country'].value_counts().head(-5).index
dataset_filtered = dataset[dataset['country'].isin(top_countries)]
plt.figure(figsize=(16, 8))
sns.set_theme(style="darkgrid")
ax = sns.violinplot(x=dataset_filtered["country"],
y=dataset_filtered["amount"], linewidth=0.5, inner=None)
sns.stripplot(x='country', y='amount', data=dataset_filtered,
dodge=True, ax=ax)
plt.title('Cantidad de ventas por países')
```

```
plt.xticks(rotation=45)
plt.show()
```




## Exercici 2

Genera un FacetGrid per a visualitzar múltiples aspectes de les dades simultàniament.

En este caso ha cambiado el orden de los colores y la ubicación de los países, pero la información es la misma tanto en Power Bi como en Python.

```
transac_companies = dataset.rename(columns={'amount': 'Gasto',
'country': 'País', 'declined':'Situación'})
transac_companies['Situación'] =
transac_companies['Situación'].replace({0: 'Aceptada', 1: 'Rechazada'})
datos_seleccionados = transac_companies[['Gasto', 'Situación',
'País']].sort_values('Situación')

plt.figure(figsize=(12, 10))
grafico = sns.FacetGrid(datos_seleccionados, col='País', col_wrap = 5,
hue= 'Situación')
grafico.map(plt.hist, 'Gasto')
grafico.add_legend()
```

```
grafico.set_ylabels('Cantidad')
plt.subplots_adjust(top=0.9)
grafico.figure.suptitle('Gasto, Cantidad y Situación de las
Transacciones por País')
plt.show()
```