Álvaro Míguez
Sprint 08.1. Visualitzacions en Python

Realitza la connexió en Python amb el MySQL Workbench per a carregar tota la informació que tens en les taules.

Realitzaràs una visualització per a cada exercici. Comenta el que et crida l'atenció de graficar aquesta variable, justifica l'elecció del gràfic i interpreta els resultats en funció de les teves dades.

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

# Connect to the database
mydb = mysql.connector.connect(
    host="127.0.0.1",
    user="root",
    password="root",
    database="sprint"
)
query = """
    SELECT *
    FROM companies
"""

query2 = """
    SELECT *
    FROM transactions
"""
query3 = """
    SELECT *
    FROM credit_cards
"""
query4 = """
    SELECT *
    FROM users
"""
query5 = """
    SELECT *
    FROM products
"""
df_comp = pd.read_sql(query, con=mydb)
df_comp.head

df_trans = pd.read_sql(query2, con=mydb)
df_trans.head

df_users = pd.read_sql(query4, con=mydb)
df_trans.head
```

✓ 0.4s

Descargando mysql.connector podemos hacer la conexión a la base de datos de mysql, necesitamos tener host, user, contraseña y database.

Aprovecho además para crear los dataframes con los que voy a trabajar en el resto de ejercicios, utilizo un head para que me muestre la información y así confirmo que todo está bien.

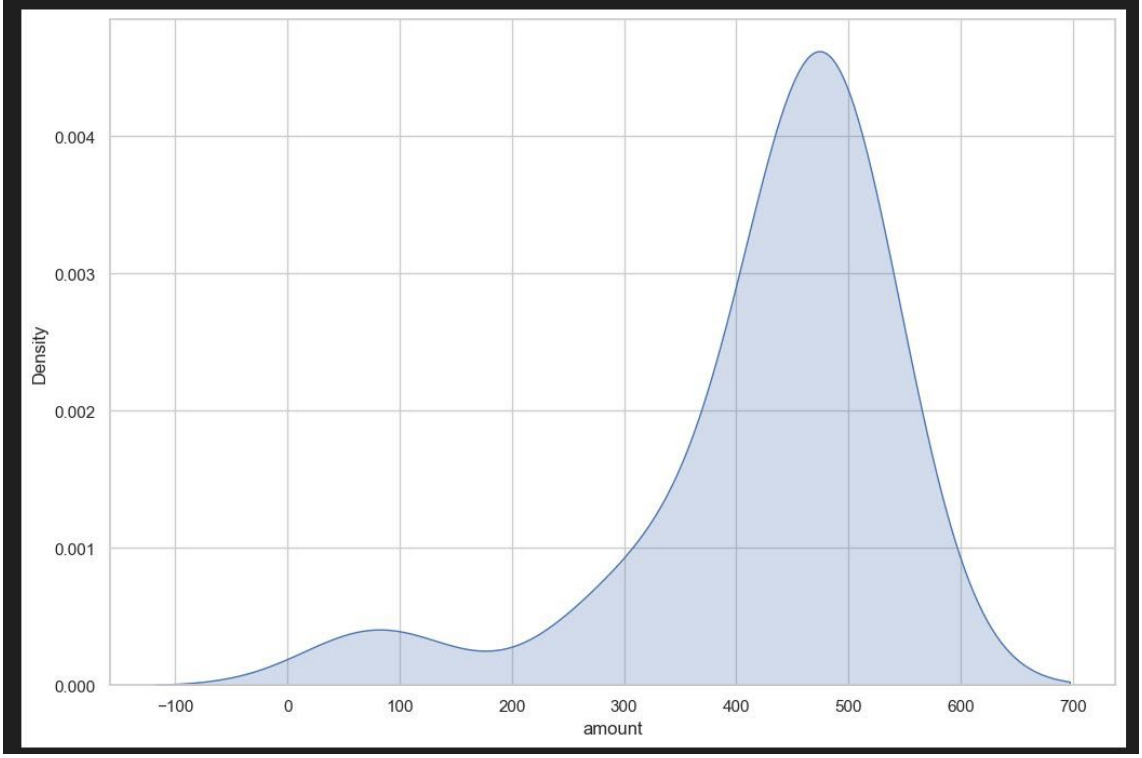
Álvaro Míguez
Sprint 08.1. Visualitzacions en Python

- Exercici 1 - Una variable numèrica.

```
#ej 1 variable numerica
import sqlalchemy as sa
df_trans = pd.read_sql(query2, con=mydb)
resulte1 = pd.merge(df_trans, df_comp, left_on = "business_id", right_on = "company_id", how="inner")
dresulte1 = resulte1.groupby("country").max("amount")
plt.figure(figsize=(12, 8))
sns.set(style="whitegrid")
sns.kdeplot(dresulte1,x="amount", fill=True)
```

```
C:\Users\amigu\AppData\Local\Temp\ipykernel_14808\1500318629.py:3: UserWarning: pandas only supports SQLAlchemy connectable (engine/con
  df_trans = pd.read_sql(query2, con=mydb)
```

```
<Axes: xlabel='amount', ylabel='Density'>
```

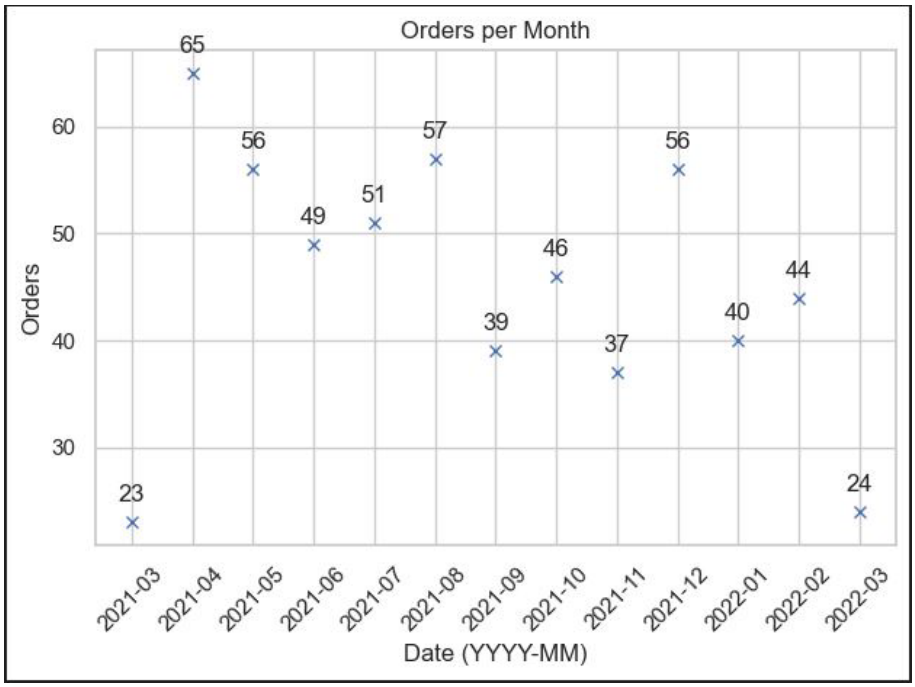


Pongo el dataframe para ver de cual emiezo y que sea fácil de seguir. Hago un merge entre dataframe de transactions y el de companies para poder hacer un gráfico de densidad por max amount. Utilizo un kdeplot para que salga la gráfica de densidad. Nos muestra que los pedidos más altos, aunque pocos están cerca de los 500.

Álvaro Míguez
Sprint 08.1. Visualitzacions en Python

- Exercici 2 - Dues variables numèriques.

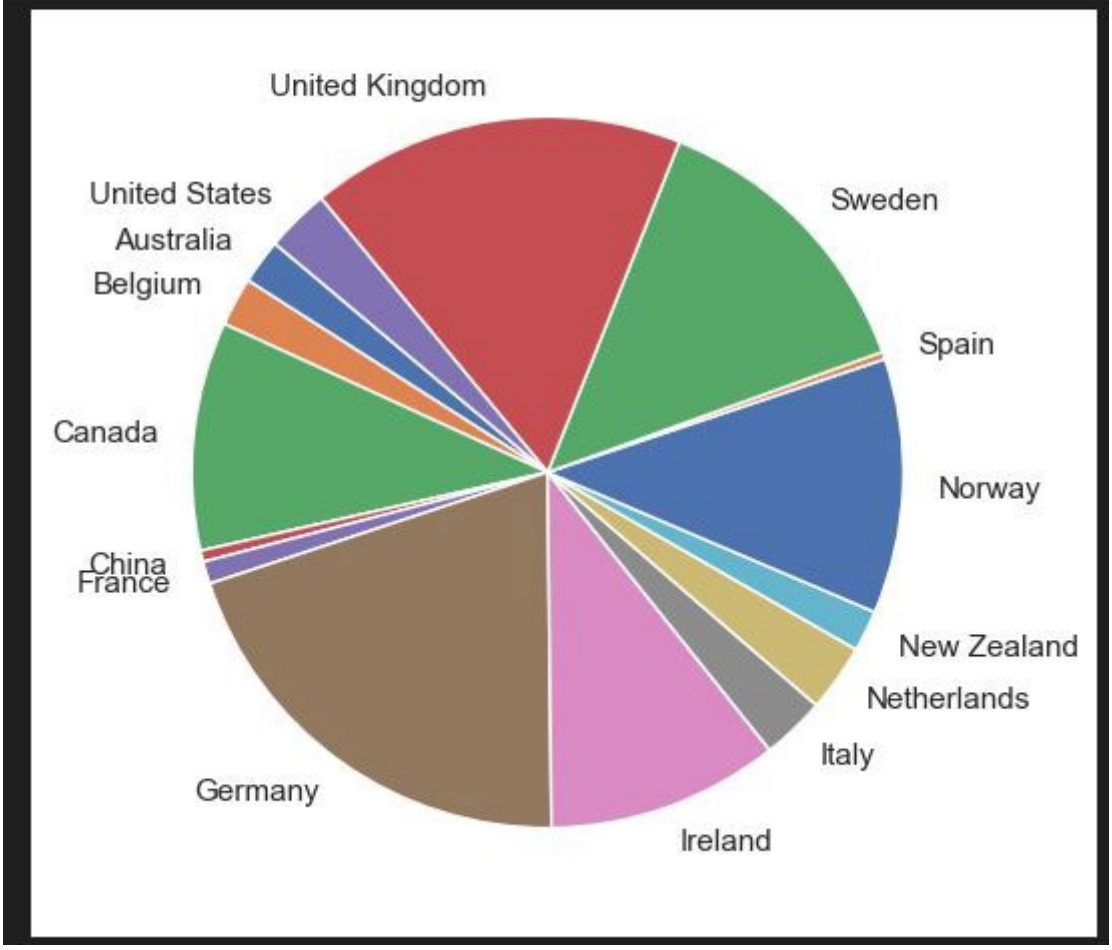
```python
#2 variables numericas

q2 = pd.read_sql(query2, con=mydb)

# Close the database connection
mydb.close()

# Convert the timestamp column to datetime format
q2['timestamp'] = pd.to_datetime(q2['timestamp'])

# Extract year and month from the timestamp
q2['year_month'] = q2['timestamp'].dt.to_period('M')

# Aggregate data: count orders per year_month
agg_df = q2.groupby('year_month').size().reset_index(name='orders')

# Convert year_month back to string for plotting
agg_df['year_month'] = agg_df['year_month'].astype(str)

plt.plot(agg_df['year_month'], agg_df['orders'], linestyle='none', marker='x')
for i in range(len(agg_df)):
    plt.annotate(
        agg_df['orders'][i],
        (agg_df['year_month'][i], agg_df['orders'][i]),
        textcoords="offset points",
        xytext=(0, 10),
        ha='center')
plt.xlabel('Date (YYYY-MM)')
plt.ylabel('Orders')
plt.title('Orders per Month')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout for better fit
plt.show()
```



Aquí creo otro frame (Q2) que hace referencia a mi dataframe query2 que referencia a transacciones. Utilizo como variables la cantidad de pedidos por mes y la fecha dividida por mes y año. Convierto timestamp a fecha y de ahí lo paso a un formato mensual. Utilizo la función size para ver cuántas veces sale mencionado cada mes y así poder contar los pedidos por mes y le doy el nombre “orders”. Por último, con ayuda de chatgpt hag un annotate para que salga el número de pedidos por mes.

- Exercici 3 - Una variable categòrica.

```
#ej 1 variable categorica
a=resulte1.groupby("country").size()
a1=resulte1.groupby("country")
a1.groups.keys()
a1l=list(a1.groups)
plt.pie(a, labels=a1l, startangle=140)
plt.tight_layout()
plt.show()
```

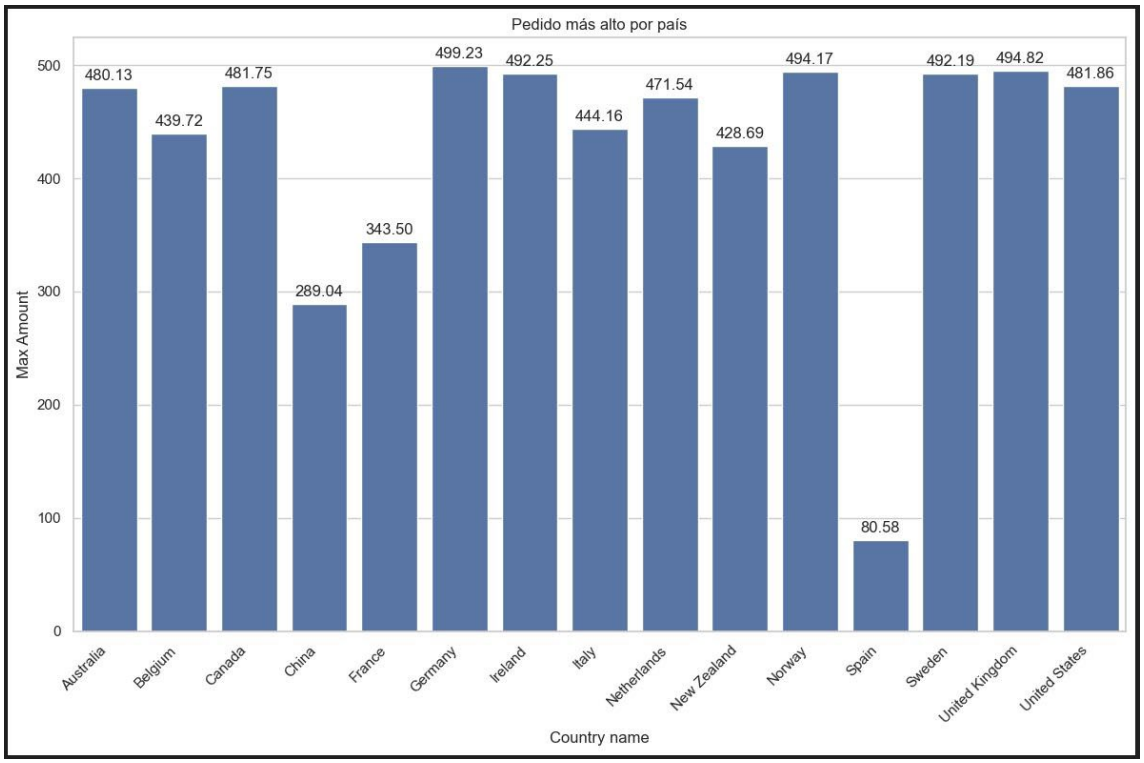


Álvaro Míguez
Sprint 08.1. Visualitzacions en Python

Resulte1 se puede ver en el primer ejercicio es el merge entre df_trans y df comp. Realmente con usar solo df_comp debería de funcionar. Lo único que hago es contar las empresas por país viendo cuantas veces son mencionadas en la database usando size(). De esta manera podemos ver cuánto ocupa cada país y los vemos mencionados.

- Exercici 4 - Una variable categòrica i una numèrica.

```
#ej  variable numerica y categorica

df_comp = pd.read_sql(query, con=mydb)
df_trans = pd.read_sql(query2, con=mydb)
resulte1 = pd.merge(df_trans, df_comp, left_on = "business_id", right_on = "company_id", how="inner")
dresulte1 = resulte1.groupby("country").max("amount")



plt.figure(figsize=(12, 8))
sns.set(style="whitegrid")
barplot = sns.barplot(x="country", y="amount",data = dresulte1)
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.2f'),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),  # Offset text by 9 points vertically
                     textcoords = 'offset points')
plt.xlabel("Country name")
plt.ylabel("Max Amount")
plt.title("Pedido más alto por país")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()  # Adjust layout to fit labels
plt.show()
```

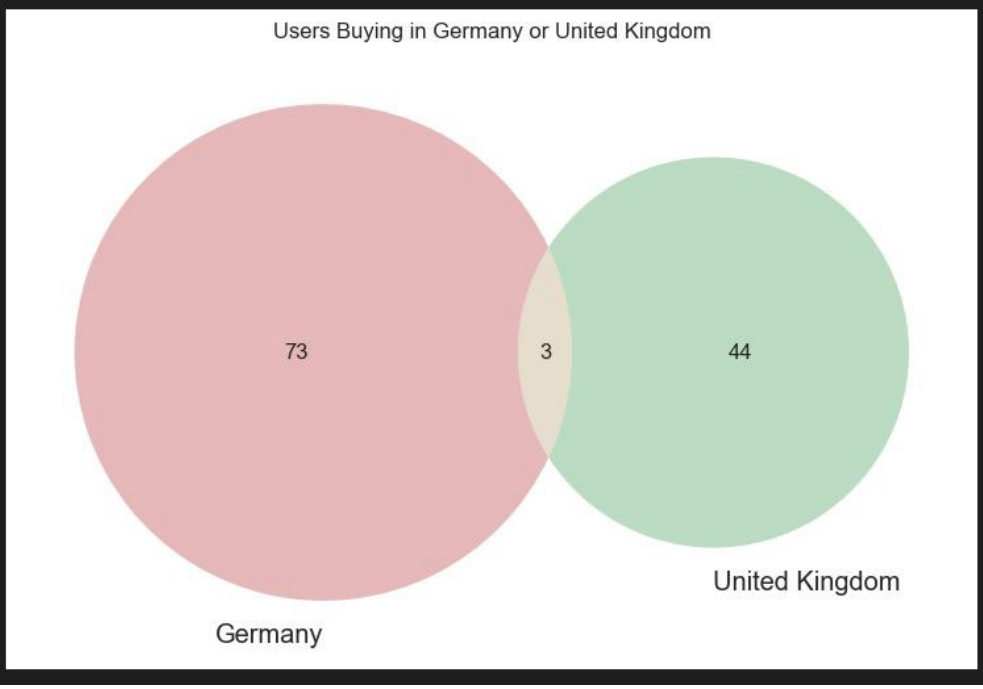


Aquí partimos de nuevo del merge resulte1. Creamos el groupby para agrupar por país y max amount. Creamos un barplot para obtener el gráfico por columnas y usamos la data del groupby (dresulte1). Con la ayuda de chatgpt usamos el annotate para que se vean las cantidades por columna.

Alemania es el país con el valor más alto, 499.23 y España el más bajo 80.58.

- Exercici 5 - Dues variables categòriques.

```
# dos variables categoricas
from matplotlib_venn import venn2
resulte1 = pd.merge(df_trans, df_comp, left_on = "business_id", right_on = "company_id", how="inner")


users_germany = set(resulte1[resulte1['country'] == 'Germany']['user_id'])
users_uk = set(resulte1[resulte1['country'] == 'United Kingdom']['user_id'])

# Plot the Venn diagram
plt.figure(figsize=(10, 6))
venn2([users_germany, users_uk], ('Germany', 'United Kingdom'))
plt.title('Users Buying in Germany or United Kingdom')
plt.show()
```



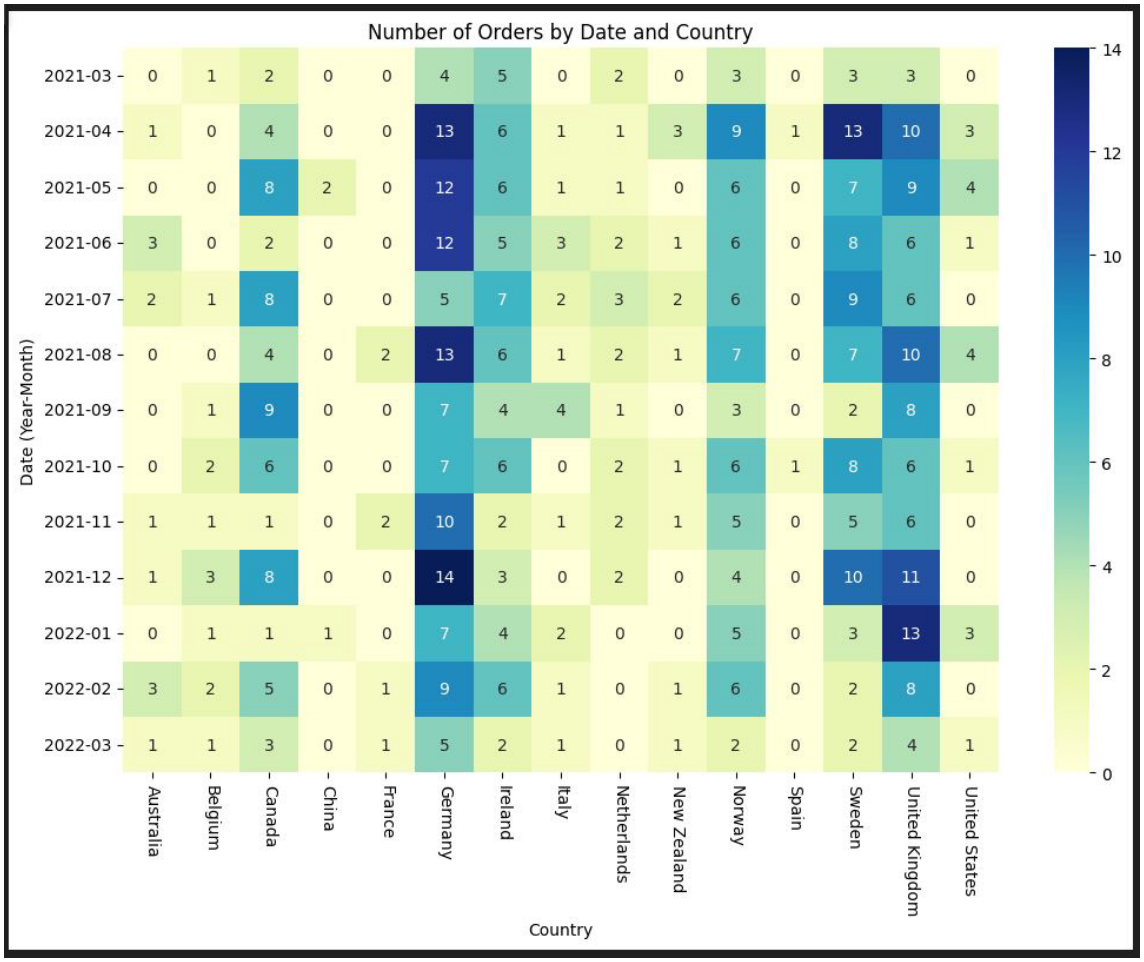
Para hacer este diagram de venn tenemos que instalar matplotlib_venn e importar venn2. Partimos de resulte1 y de ahí sacamos los users que compran en empresas de Alemania o de Inglaterra. Lo conseguimos porque al leer el dataframe df_trans podemos ver a que id de empresa están haciendo el pedido y de ahí podemos sacar el país de la compañía.

Álvaro Míguez
Sprint 08.1. Visualitzacions en Python

- Exercici 6 - Tres variables.

```
# variables
resulte1 = pd.merge(df_trans, df_comp, left_on = "business_id", right_on = "company_id", how="inner")
# Convert the timestamp column to datetime
resulte1['timestamp'] = pd.to_datetime(resulte1['timestamp'])

# Extract the date (year-month) for aggregation
resulte1['year_month'] = resulte1['timestamp'].dt.to_period('M')

# Aggregate the data by year_month and country to count orders
heatmap_data = resulte1.groupby(['year_month', 'country']).size().unstack(fill_value=0)

# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(heatmap_data, annot=True, fmt="d", cmap="YlGnBu")
plt.title('Number of Orders by Date and Country')
plt.xlabel('Country')
plt.ylabel('Date (Year-Month)')
plt.xticks(rotation=-90)  # Rotate x-axis labels for better readability
plt.yticks(rotation=0)  # Keep y-axis labels horizontal
plt.show()
```

✓ 0.7s



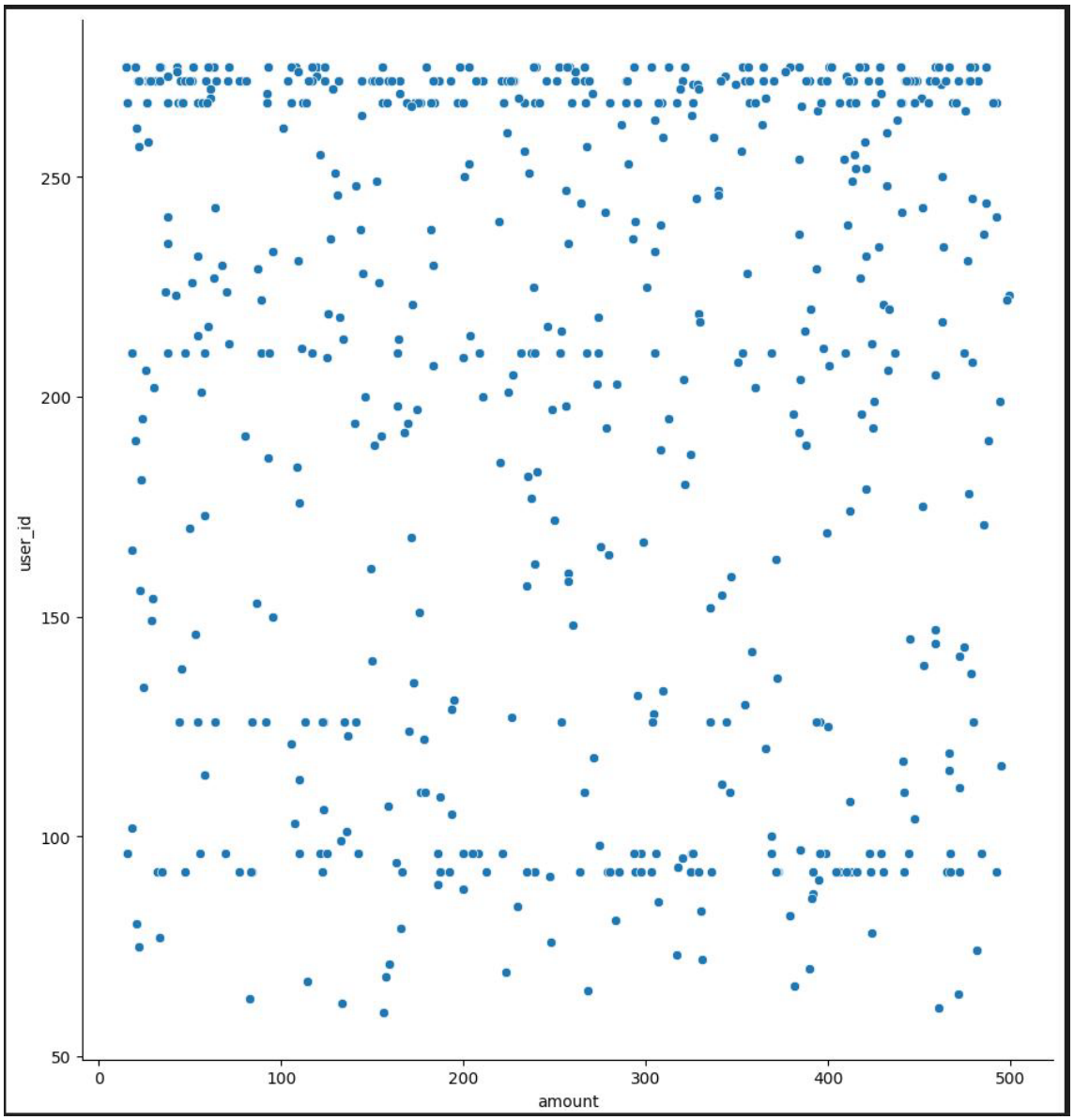
Partiendo de resulte1, hacemos lo mismo que en el ejercicio 2 y pasamos el timestamp a mes. Luego para la data del heatmap agrupamos por el nuevo timestamp mensual (year-month), contamos la presencia de cada mes usando la función size() y así saber el número de pedidos por mes. Por último, le damos estos datos para que se haga el mapa de calor por mes, país y cantidad de pedidos.

Álvaro Míguez
Sprint 08.1. Visualitzacions en Python

- Exercici 7 - Graficar un Pairplot.

```
#nivel 1, graficar pairplot
df_qp = pd.read_sql(query2, con=mydb)

gra = sns.pairplot(df_qp,
    x_vars=["amount"],
    y_vars=["user_id"])
gra.fig.set_size_inches(12,12)
```

✓ 0.3s



Creamos el frame df_qp que referencia a query2 que referencia a transactions. Para hacer el pairplot solo tengo que alimentar la tabla el dataframe que quiero que me desglose, para que no me desglose toda la tabla, trabajo solo con amount y user_id y vemos como se reparten todos los pedidos por amount y user_id

Álvaro Míguez
Sprint 08.1. Visualitzacions en Python

Nivel 2

Exercici 1 - Correlació de totes les variables numèriques.

```
#nivel 2, correlacion variables numericas
df_c = pd.read_sql(query2, con=mydb)


df_c.corr(numeric_only=True)
```

```
C:\Users\amigu\AppData\Local\Temp\ipykernel_24492\3086146024.py
  df_c = pd.read_sql(query2, con=mydb)
```

	amount	declined	user_id	lat	longitude
amount	1.000000	-0.037882	-0.035354	0.078116	0.043439
declined	-0.037882	1.000000	0.414293	0.010402	-0.002523
user_id	-0.035354	0.414293	1.000000	-0.004486	0.017145
lat	0.078116	0.010402	-0.004486	1.000000	-0.006894
longitude	0.043439	-0.002523	0.017145	-0.006894	1.000000

Aquí vemos la correlación entre todos los valores númericos entre todas las columnas del dataframe query2 que hace referencia a la tabla de transacciones. Necesario poner “numeric_only = True”, ya que sino intentará hacer relaciones con variables categóricas.

Corrección Lucía

```
df_c = pd.read_sql(query2, con=mydb)
d_matrix = pd.DataFrame(df_c, columns=['amount', 'lat', 'longitude'])
matrix = d_matrix.corr()
matrix.style.background_gradient(cmap='coolwarm', axis=None)
```

✓ 0.0s

```
D:\users\ciberconnecta_11\AppData\Local\Temp\ipykernel_18780\1323706851.py:1: User
  df_c = pd.read_sql(query2, con=mydb)
```

	amount	lat	longitude
amount	1.000000	0.078116	0.043439
lat	0.078116	1.000000	-0.006894
longitude	0.043439	-0.006894	1.000000

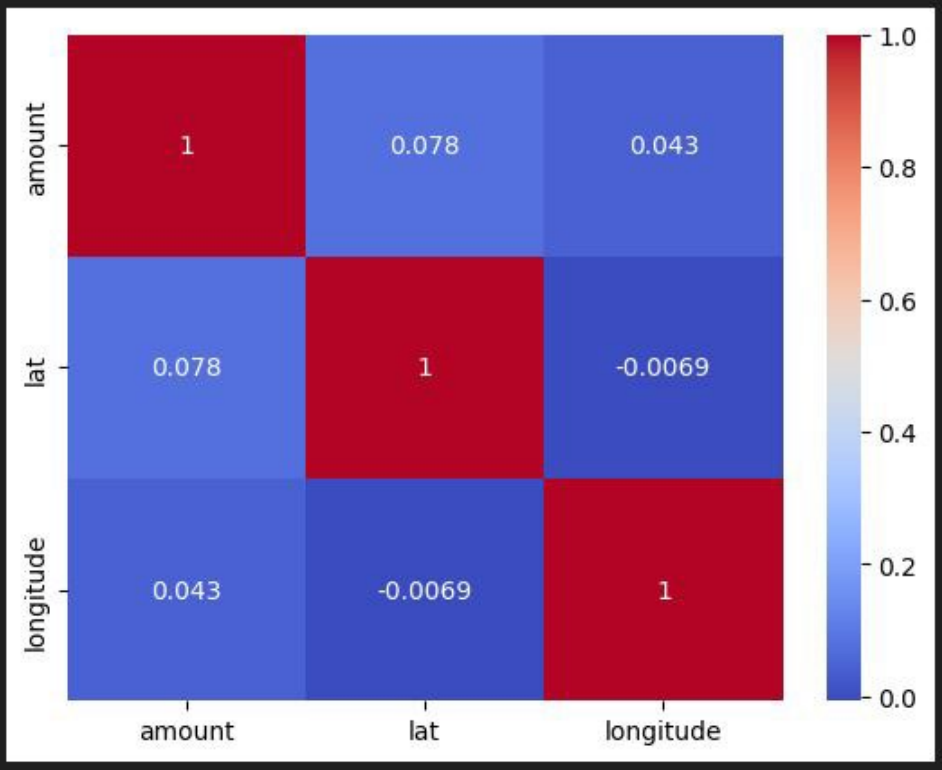
El primer paso de jugar con la matrix para poder crear el mapa de calor fue jugar con el estilo y ver si podía acotar las columnas que estaba usando ya que así no usaba las categóricas de declined y user_id.

```
df_c = pd.read_sql(query2, con=mydb)
d_matrix = pd.DataFrame(df_c, columns=['amount', 'lat', 'longitude'])
matrix = d_matrix.corr()
sns.heatmap(matrix, annot=True, cmap="coolwarm")
```

✓ 0.2s

```
D:\users\ciberconnecta_11\AppData\Local\Temp\ipykernel_18780\1003287095.py:1: Us
  df_c = pd.read_sql(query2, con=mydb)
```

```
<Axes: >
```



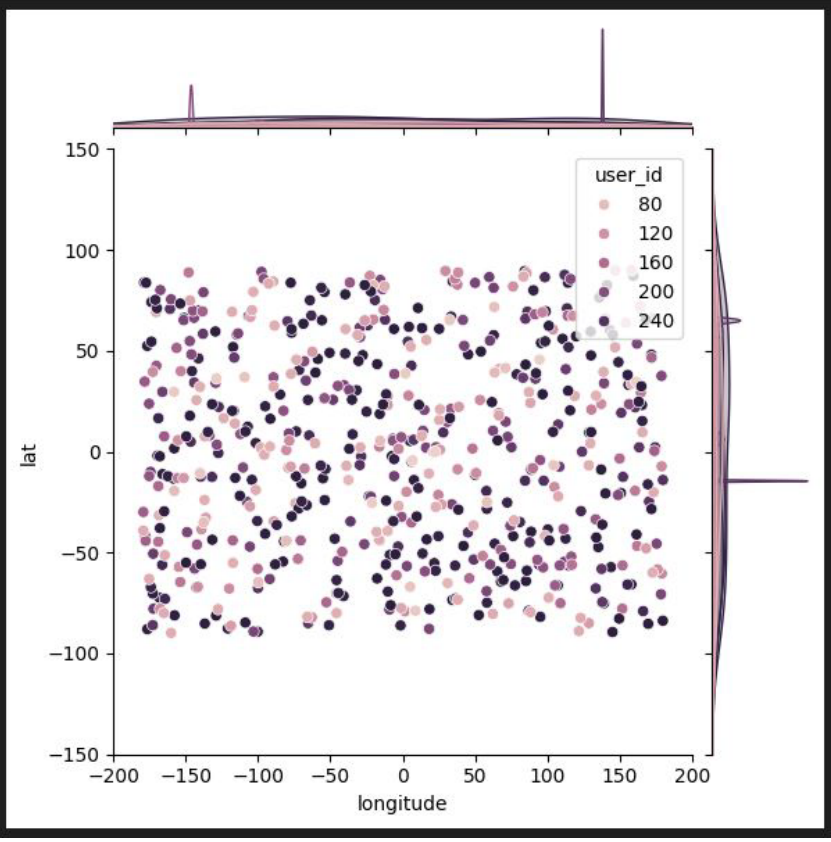
Al final después de trastear puedo crear el heatmap con solo las variables numéricas acordadas, amount, lat y longitud.

Álvaro Míguez
Sprint 08.1. Visualitzacions en Python

Exercici 2 - Implementa un jointplot.

```
#nivel 2, jointplot
df_pair = pd.read_sql(query2, con=mydb)


ad=sns.jointplot(df_pair, x="longitude", y="lat", hue="user_id",xlim=(-200, 200), ylim=(-150, 150))
```

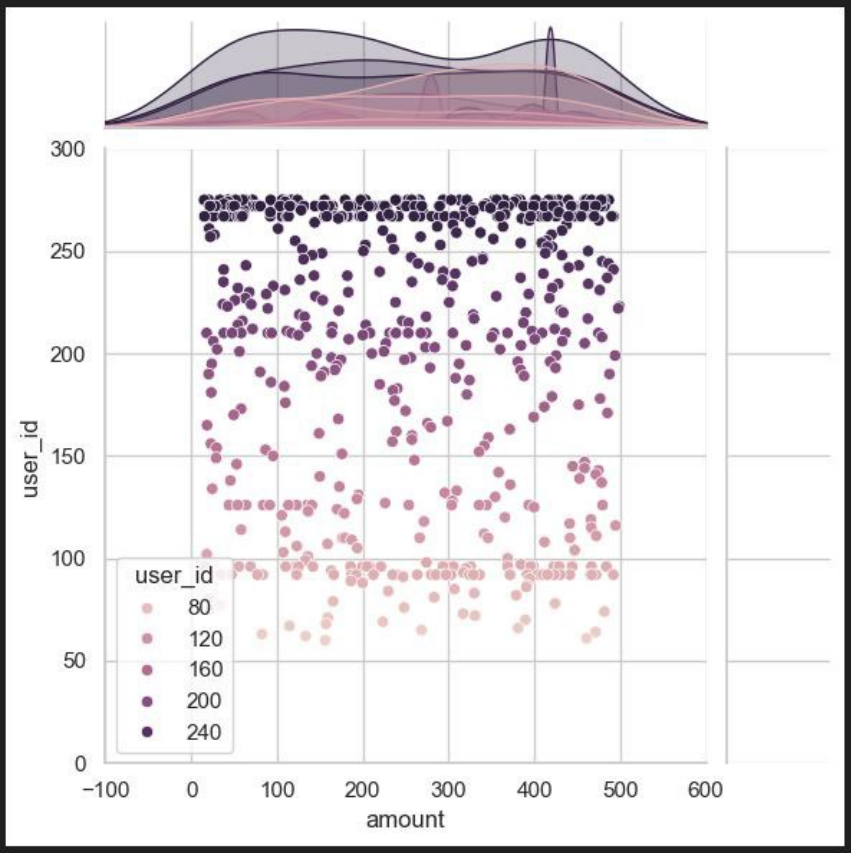
```
D:\users\ciberconnecta_11\AppData\Local\Temp\ipykernel_12992\4073064569.py:2: UserWarning: pandas only su
  df_pair = pd.read_sql(query2, con=mydb)
```



Este fue mi jointplot original donde utilizo longitud y latitude por utilizar valores diferentes al resto de ejercicios. Utilizar longitud y latitud no tiene mucho sentido, pero quería utilizar otros valores a ver cómo funcionaba.

```
df_pair = pd.read_sql(query2, con=mydb)

ad=sns.jointplot(df_pair, x="amount", y="user_id", hue="user_id",xlim=(-100, 600), ylim=(0, 300))
```

```
C:\Users\amigu\AppData\Local\Temp\ipykernel_14808\431213741.py:1: UserWarning: pandas only supports S
  df_pair = pd.read_sql(query2, con=mydb)
```



Este es mi jointplot definitivo, viendo el amount por user_id para ver el valor de los pedidos por usuario.

Álvaro Míguez
Sprint 08.1. Visualitzacions en Python

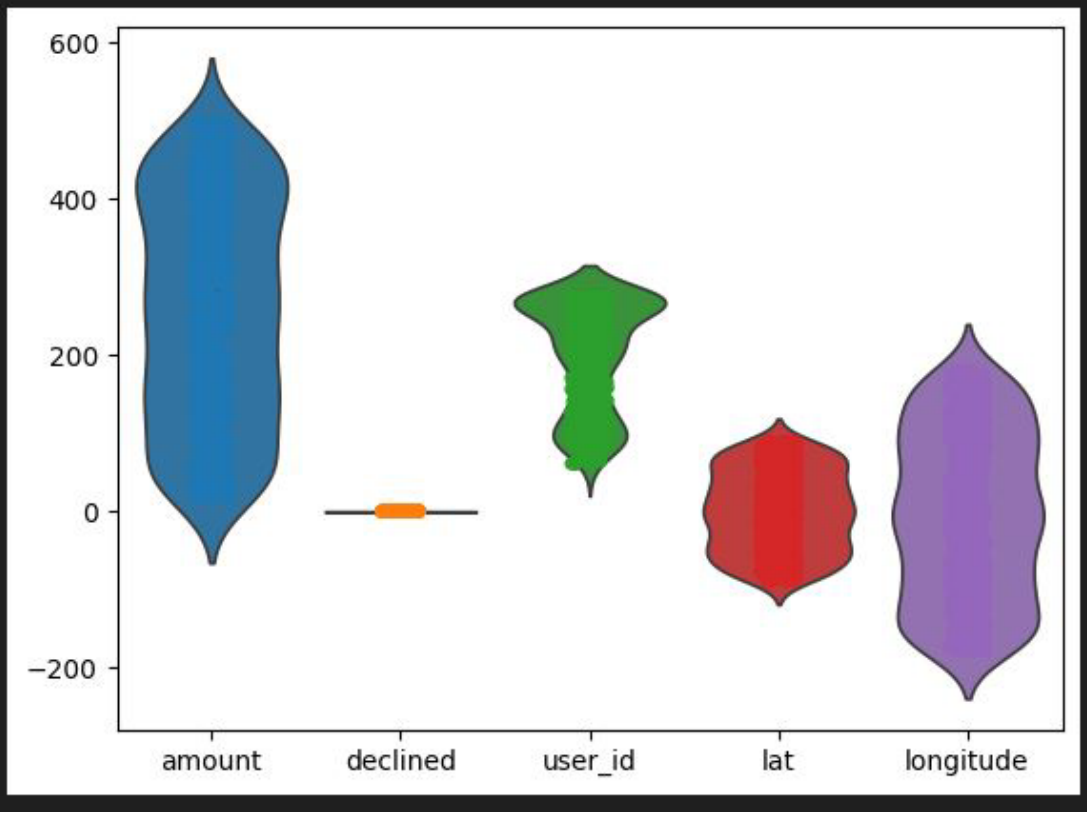
Nivel 3

Exercici 1 - Implementa un violinplot combinat amb un altre tipus de gràfic.

```
#nivel 3, violinplot
df_trans = pd.read_sql(query2, con=mydb)

sns.violinplot(df_trans)
sns.stripplot(df_trans)
```

```
D:\users\ciberconnecta_11\AppData\Local\Temp\ipykernel_5012\4016078039.py:2:
  df_trans = pd.read_sql(query2, con=mydb)
```

```
<Axes: >
```



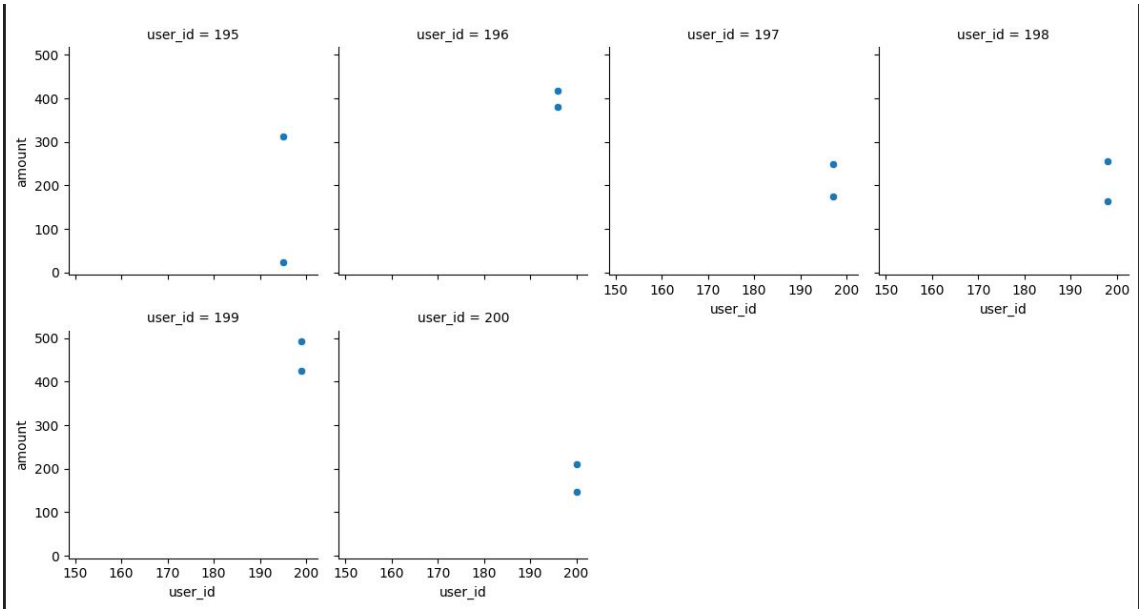
Tuve que investigar mucho para poder juntar dos gráficos diferentes, acabe usando un stripplot para que nos de esta especie de cuerpo dentro del violinplot.

Álvaro Míguez
Sprint 08.1. Visualitzacions en Python

Exercici 2 - Genera un FacetGrid per a visualitzar múltiples aspectes de les dades simultàniament.

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

# Assuming df_trans, df_comp, and df_users are already defined

# Merge dataframes

l3m = pd.merge(df_trans, df_users, left_on="user_id", right_on="id", how="inner")

# Groupby and aggregate
l3max = l3m.groupby("user_id").agg({'amount': 'max'}).reset_index()
l3min = l3m.groupby("user_id").agg({'amount': 'min'}).reset_index()

# Filter for UK users
l3_uk = l3m[l3m['country'] == 'United Kingdom']

# Plotting using FacetGrid
g = sns.FacetGrid(l3_uk, col="user_id", col_wrap=4)
g.map_dataframe(sns.scatterplot, x="user_id", y="amount")

plt.show()
```

✓ 13.9s



Para poder trabajar el facetgrid decidí que el gráfico me mostrase el pedido más alto y el más bajo de todos los usuarios del Reino Unido, utilizo el Reino Unido ya que es el país con menos usuarios (50).

Hago la unión entre df_trans y df_users para poder sacar la información de los usuarios y de sus transacciones, agrupo por amount máximo y minimo y por último filtro por país = United Kingdom.