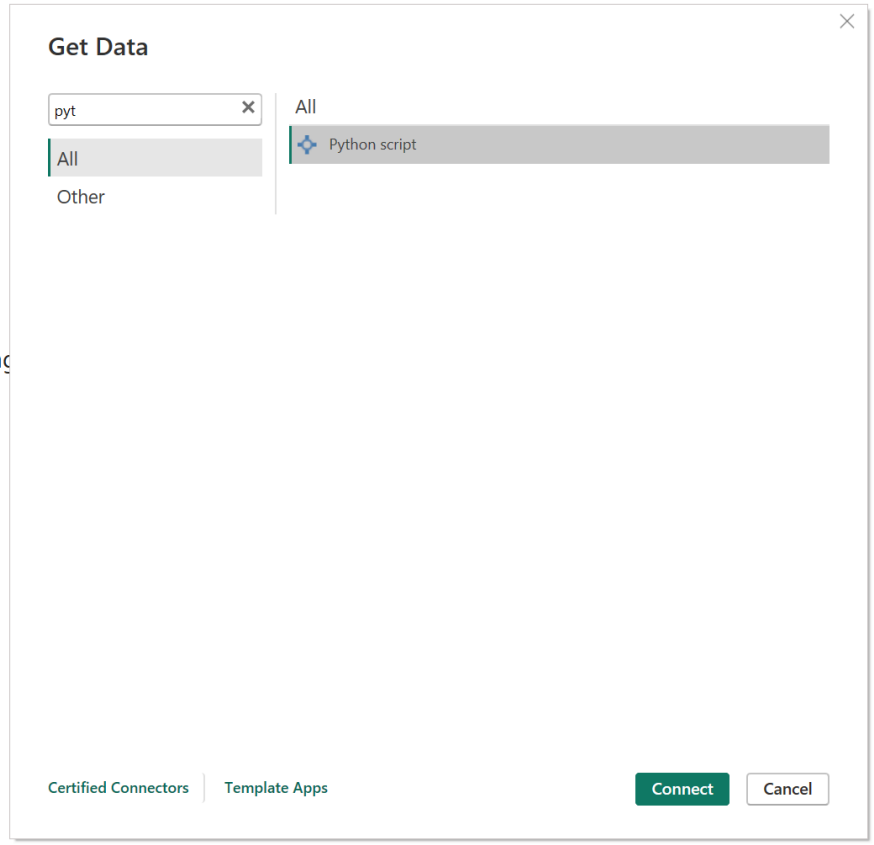
SPRINT 8.2

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.

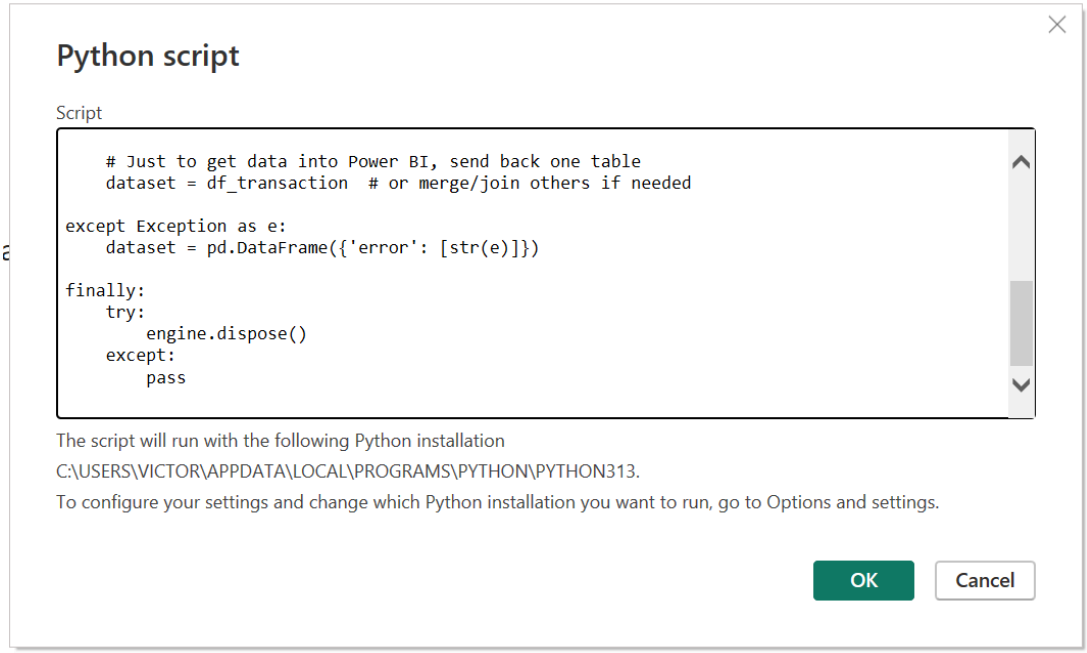
Objectius

- Integrar els scripts de Python desenvolupats en la Tasca 1 amb Power BI per a la creació de visualitzacions avançades.
- Documentar cada pas del procés de creació de l'informe amb scripts per a facilitar la reproducibilidad i manteniment.

First, we have to select the way in which we will get the data, in this case through a python script.

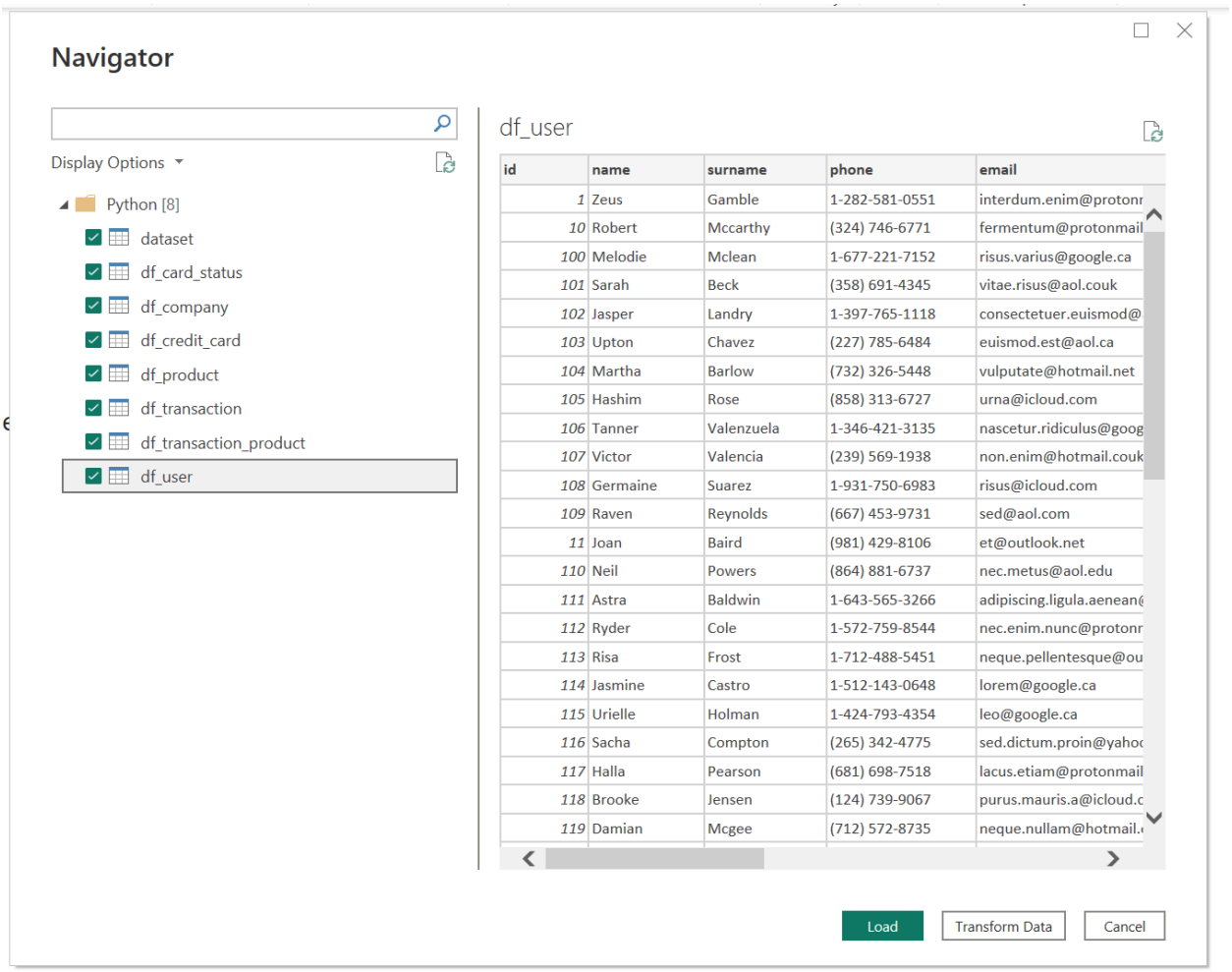


Then we will run the script that will connect with MYSQL workbench, where the database lives and extract the information to Power BI.
Importing all the libraries that will be used to create the visuals, and proper tables.

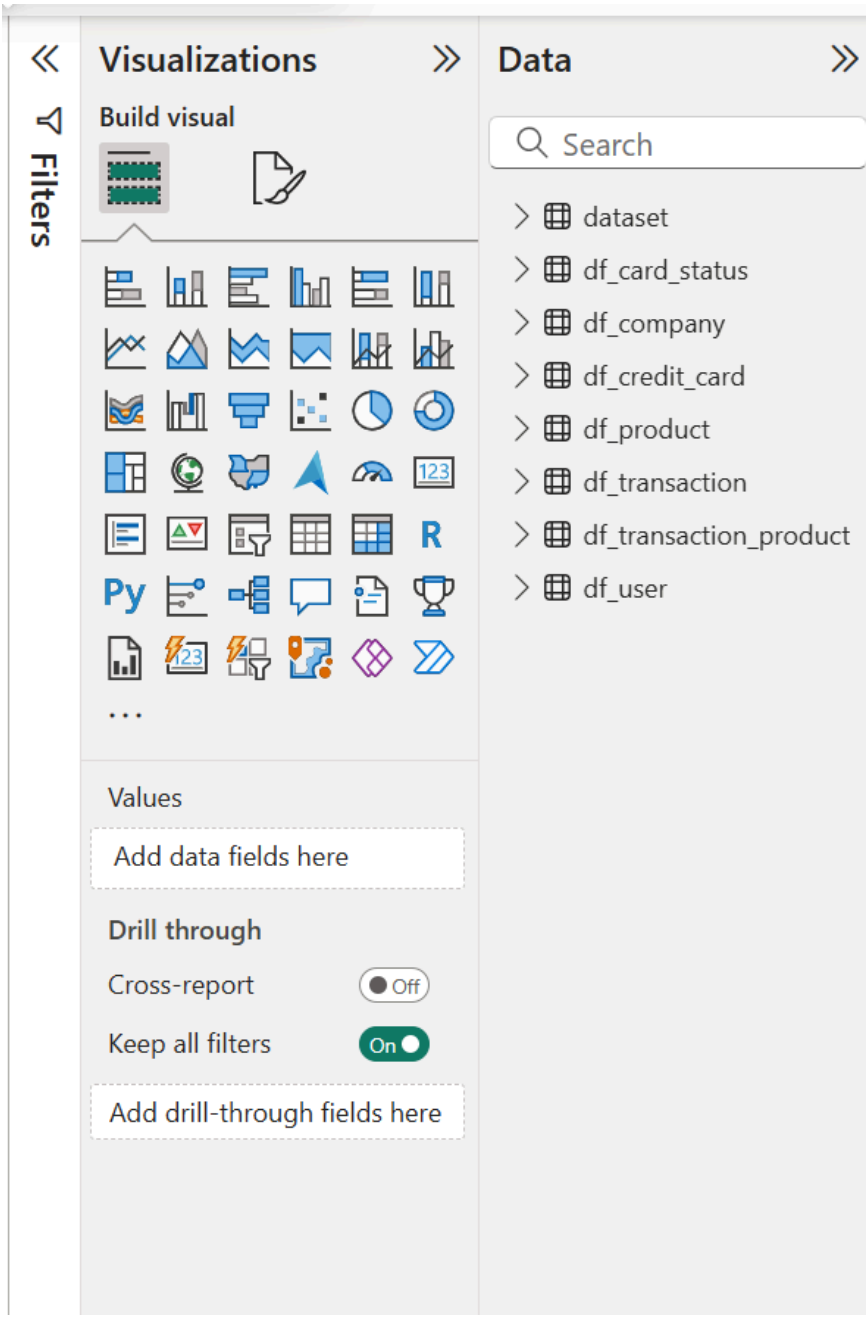



```python
from sqlalchemy import create_engine
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

try:
    engine = create_engine("mysql+pymysql://root:0000@localhost:3306/empresa")

    df_transaction = pd.read_sql("SELECT * FROM transaction;", engine)
    df_product = pd.read_sql("SELECT * FROM product;", engine)
    df_user = pd.read_sql("SELECT * FROM user;", engine)
    df_credit_card = pd.read_sql("SELECT * FROM credit_card;", engine)
    df_card_status = pd.read_sql("SELECT * FROM card_status;", engine)
    df_company = pd.read_sql("SELECT * FROM company;", engine)
    df_transaction_product = pd.read_sql("SELECT * FROM transaction_product;", engine)

    # Just to get data into Power BI, send back one table
    dataset = df_transaction  # or merge/join others if needed

except Exception as e:
    dataset = pd.DataFrame({'error': [str(e)]})

finally:
    try:
        engine.dispose()
    except:
        pass
```

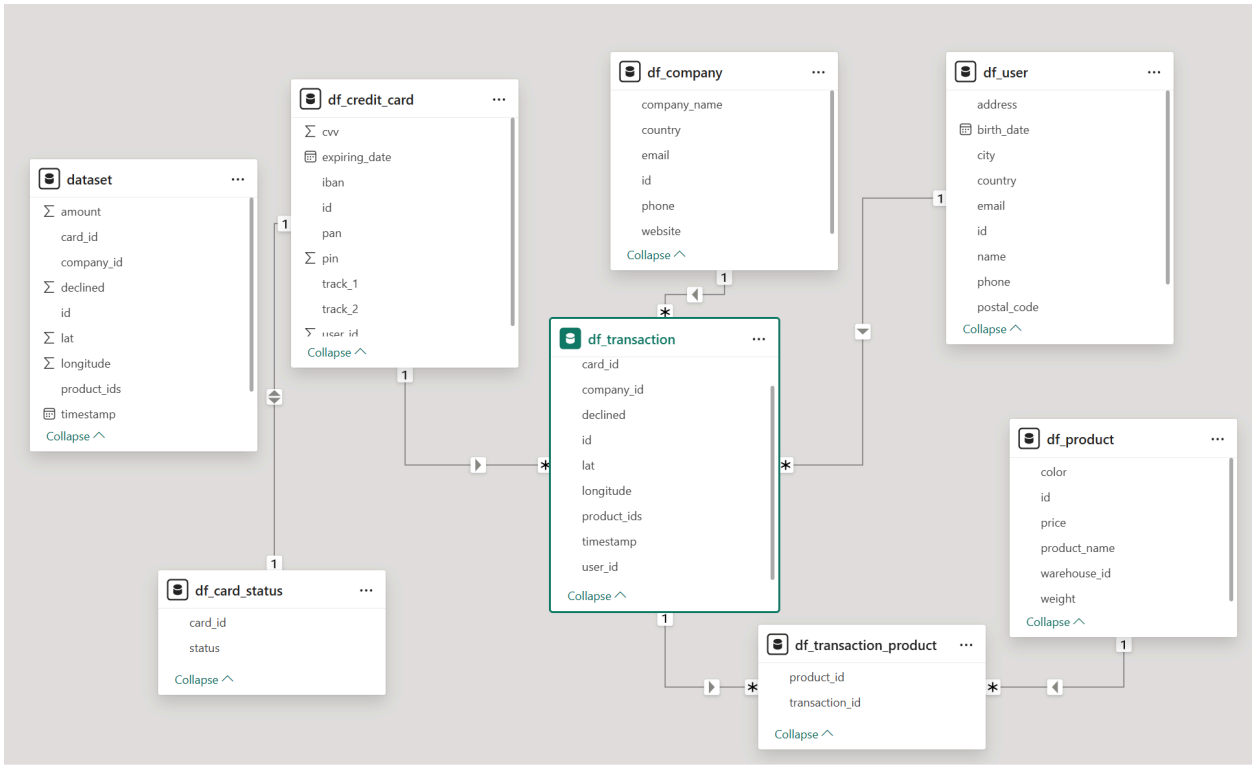
After the script is run, we select the tables from the database, in this case I select all the tables.



Here, we can see the tables are showing.



The tables are created with no relationship between them, so we have to do it manually.



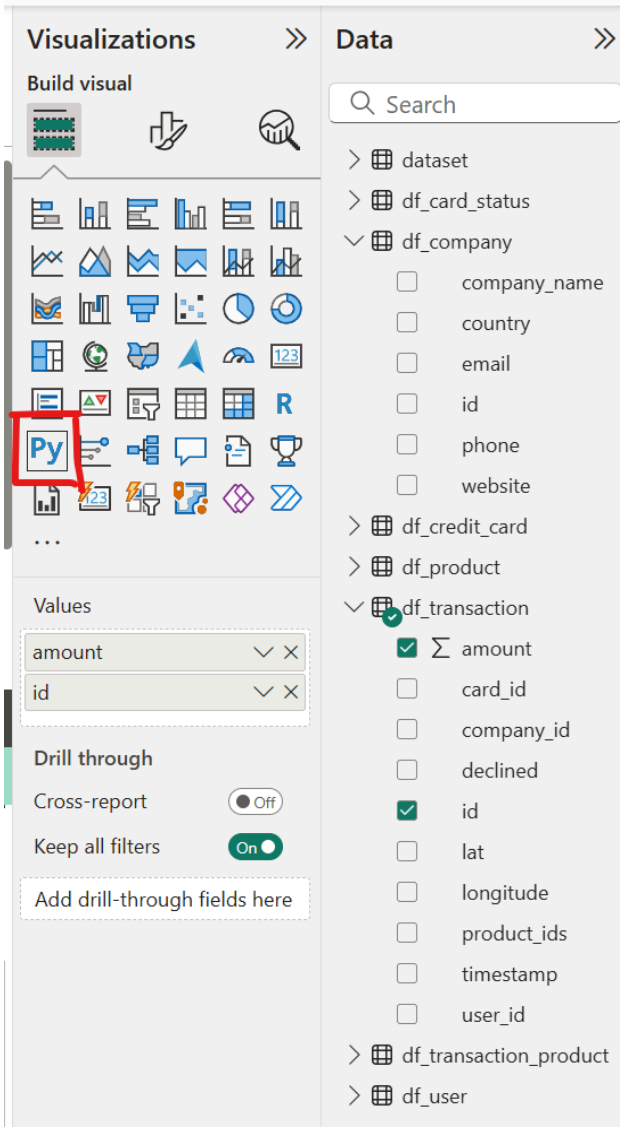
NIVELL 1

Els 7 exercicis del nivell 1 de la tasca 01

- Exercici 1

Una variable numèrica.

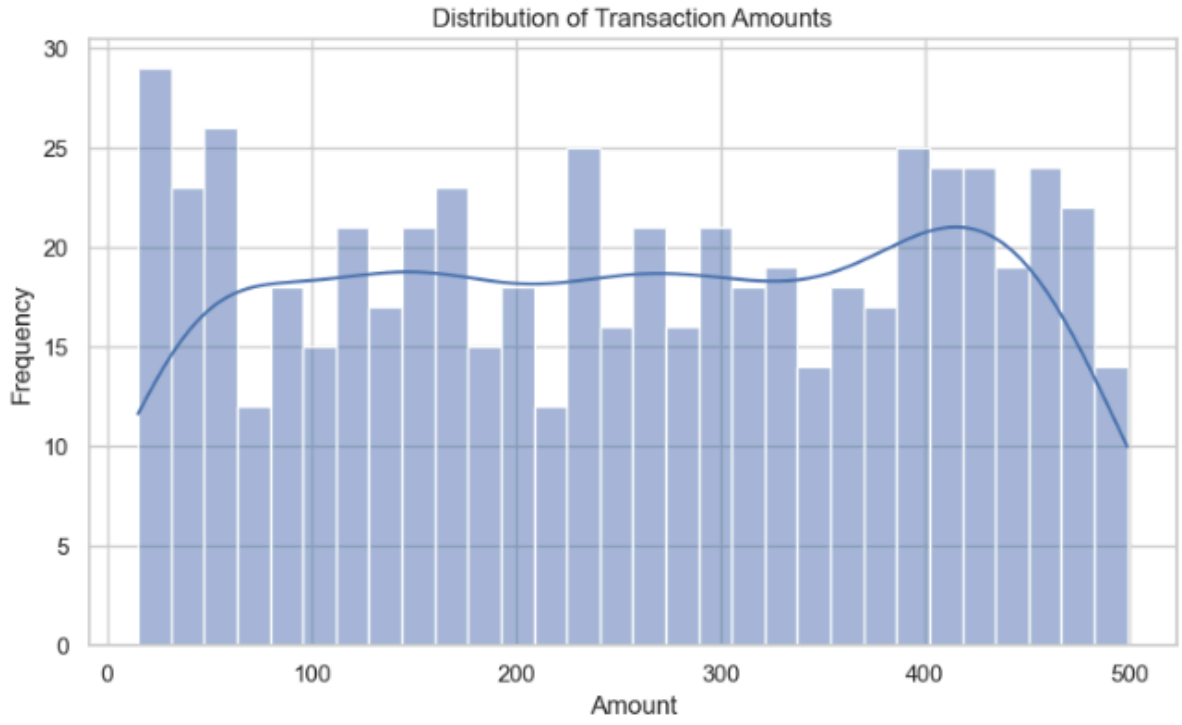
To create the visuals we will be selecting the python visual, that allows python scripts to build the visuals, and then select the columns that will be used in the graphic.
Power BI by default erases duplicate lines with the same values, so we have to add a column that has distinct values to prevent incorrect display of data.
In this exercise we added the column id from the table transaction.



Script used to create the visual from exercise 1.

```
# Paste or type your script code here:
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Remove rows with missing values in relevant columns
dataset = dataset.dropna(subset=['id', 'amount'])

# Set Seaborn style
sns.set(style='whitegrid')

plt.figure(figsize=(8, 5))
sns.histplot(data=dataset, x='amount', bins=30, kde=True)

plt.title('Distribution of Transaction Amounts')
plt.xlabel('Amount')
plt.ylabel('Frequency')
plt.grid(True)
```

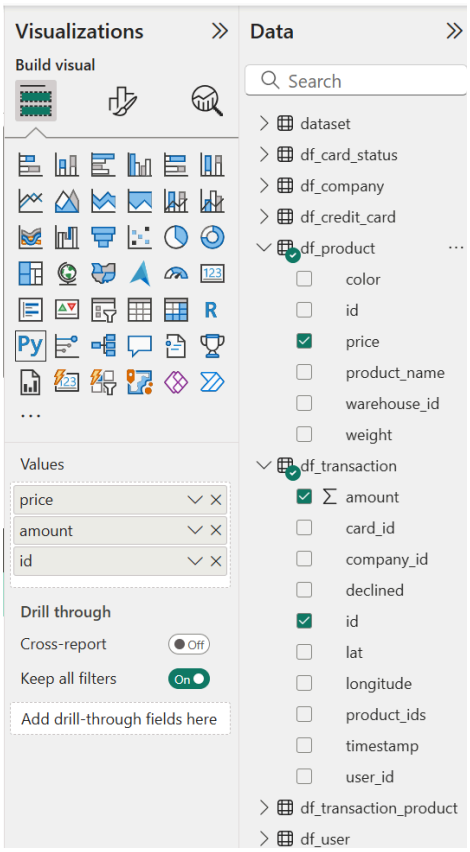
Plot the graphic.



- Exercici 2

Dues variables numèriques.

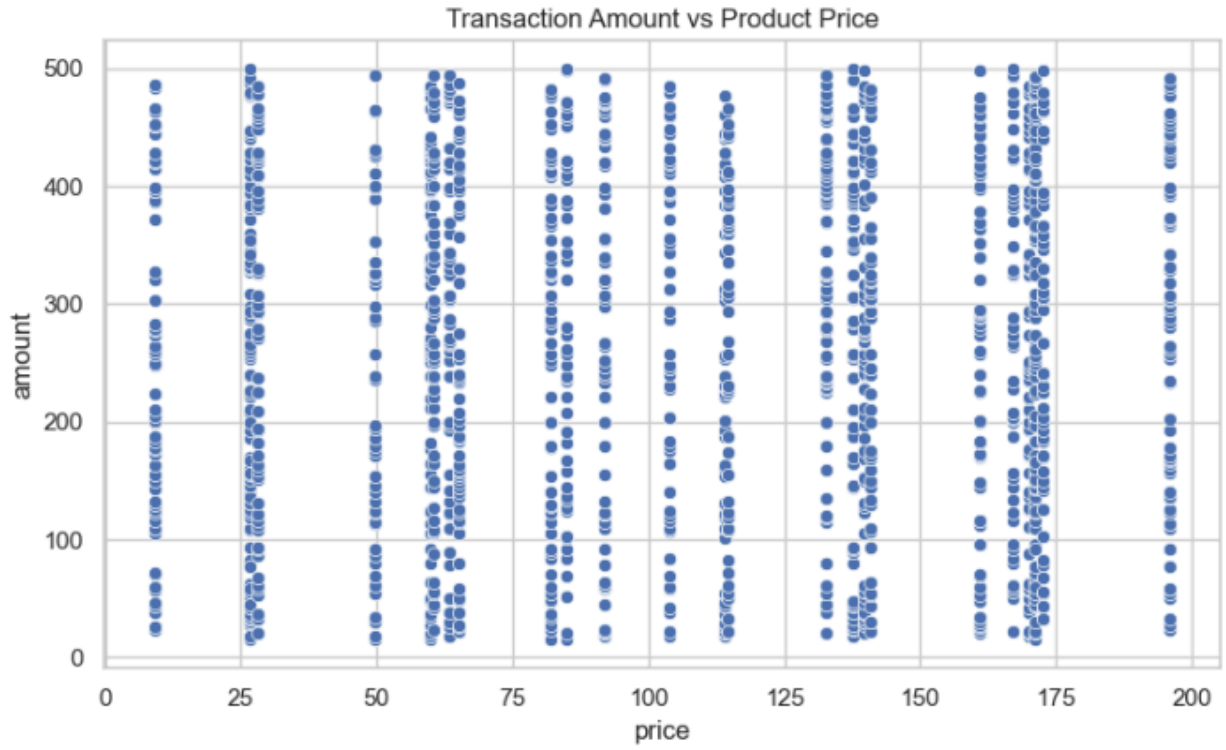
Select the columns we will be using. Here we also selected the column id from the transaction table so we can add a unique value and prevent lines from being deleted.



Run the python script

```
 5
 6 # Paste or type your script code here:
 7 import pandas as pd
 8 import matplotlib.pyplot as plt
 9 import seaborn as sns
10
11 plt.figure(figsize=(8, 5))
12 sns.set(style='whitegrid')
13 sns.scatterplot(data=dataset, x='price', y='amount')
14 plt.title('Transaction Amount vs Product Price')
15 plt.tight_layout()
16 plt.show()
```

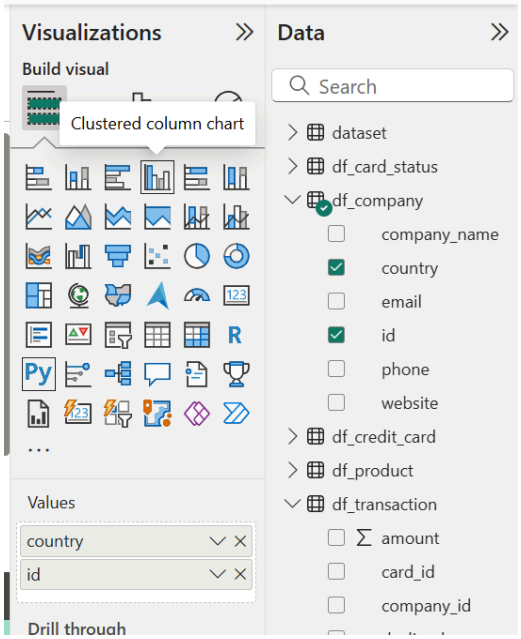
Plot the graphic



- Exercici 3

Una variable categòrica.

Selecting the columns. In this case we also had to select the column id, so the graphic could work.



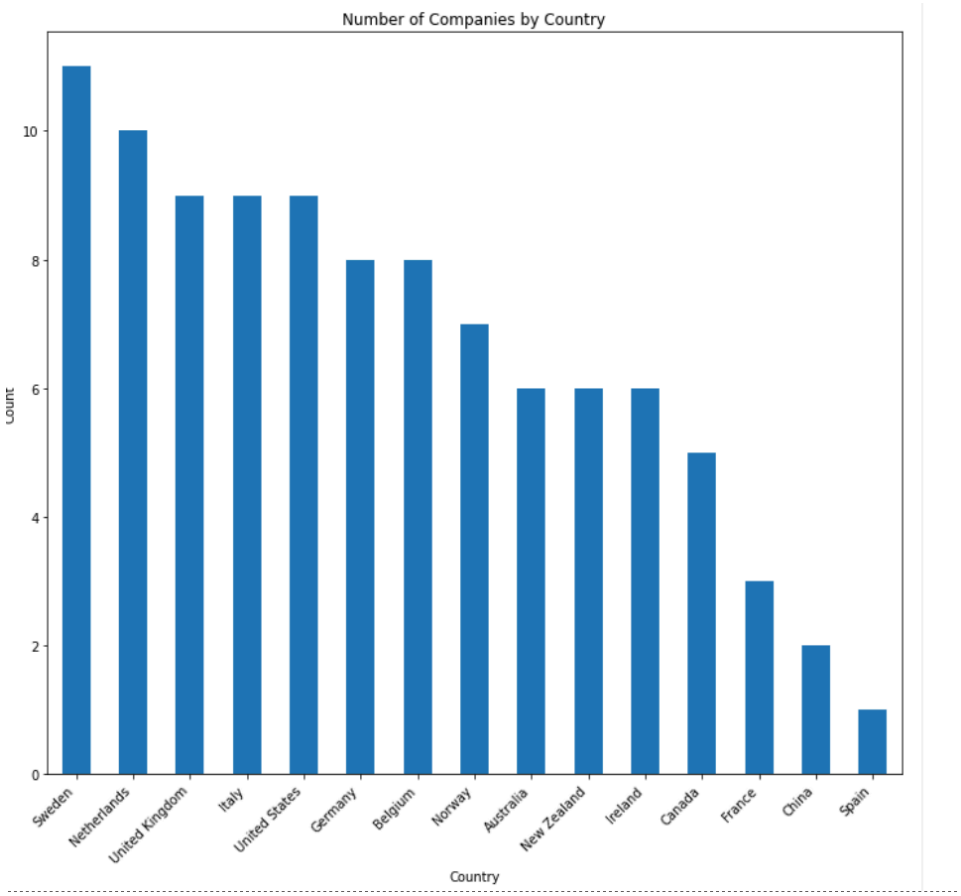
Then we run the python script adapted to work in Power BI

Python script editor

⚠ Duplicate rows will be removed from the data.

```
# dataset = pandas.DataFrame(country, id)
# dataset = dataset.drop_duplicates()

# Paste or type your script code here:
import pandas as pd
import matplotlib.pyplot as plt

# Count number of companies per country
country_counts = dataset.groupby('country').size().sort_values(ascending=False)

# Plot
country_counts.plot(kind='bar')
plt.title('Number of Companies by Country')
plt.xlabel('Country')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

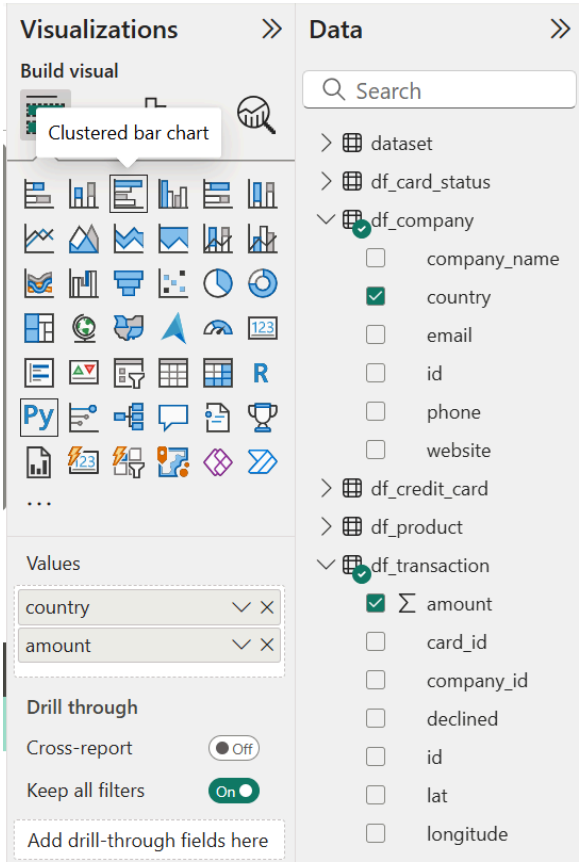
Plot the graphic



- Exercici 4

Una variable categòrica i una numèrica.

Select the columns



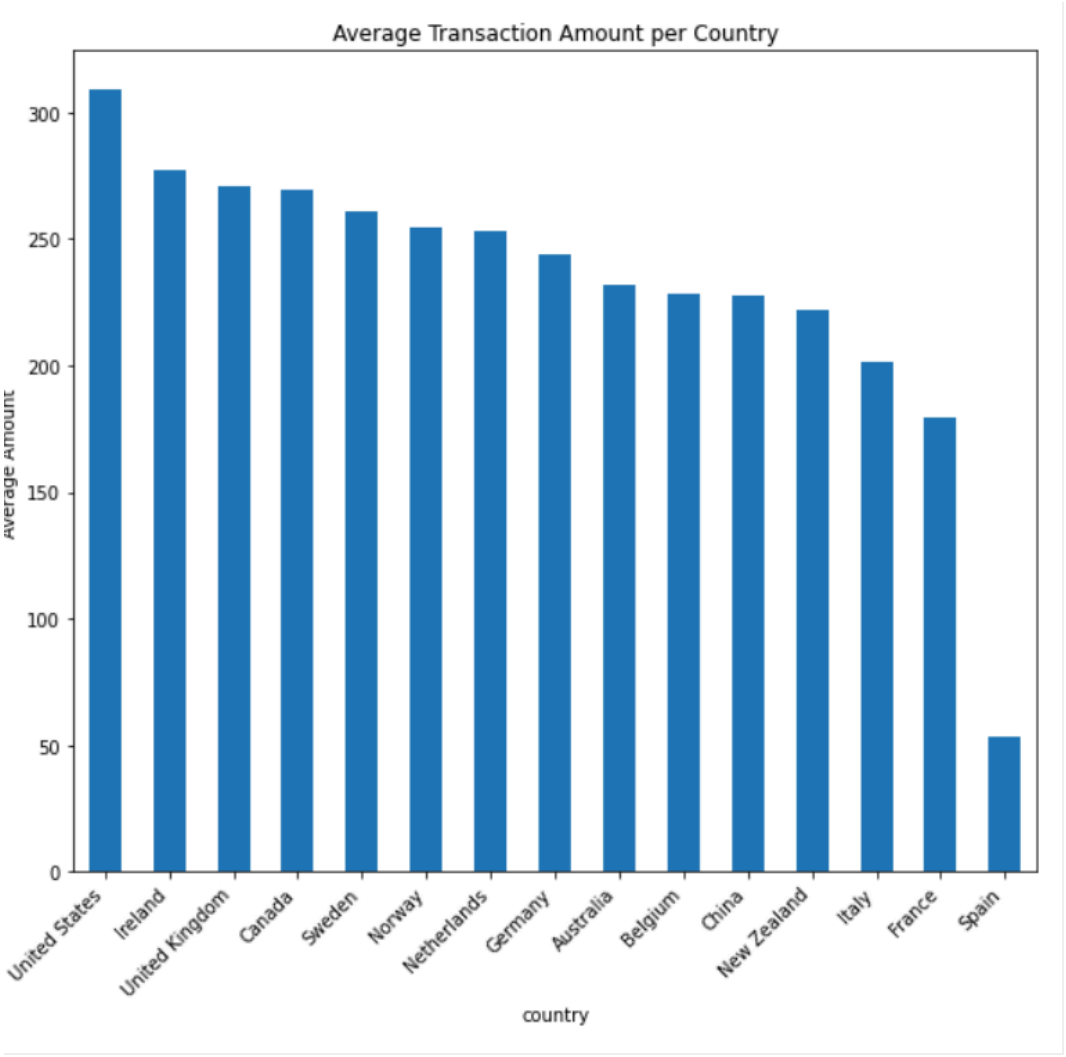
Run the script

Python script editor

⚠ Duplicate rows will be removed from the data.

```
# Paste or type your script code here:
import pandas as pd
import matplotlib.pyplot as plt

# Drop missing data
dataset = dataset.dropna(subset=['country', 'amount'])

# Drop rows with failed conversion
dataset = dataset.dropna(subset=['amount'])

# Group and calculate average
avg_amount = dataset.groupby('country')['amount'].mean().sort_values(ascending=False)

avg_amount.plot(kind='bar', title='Average Transaction Amount per Country')
plt.ylabel('Average Amount')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

```

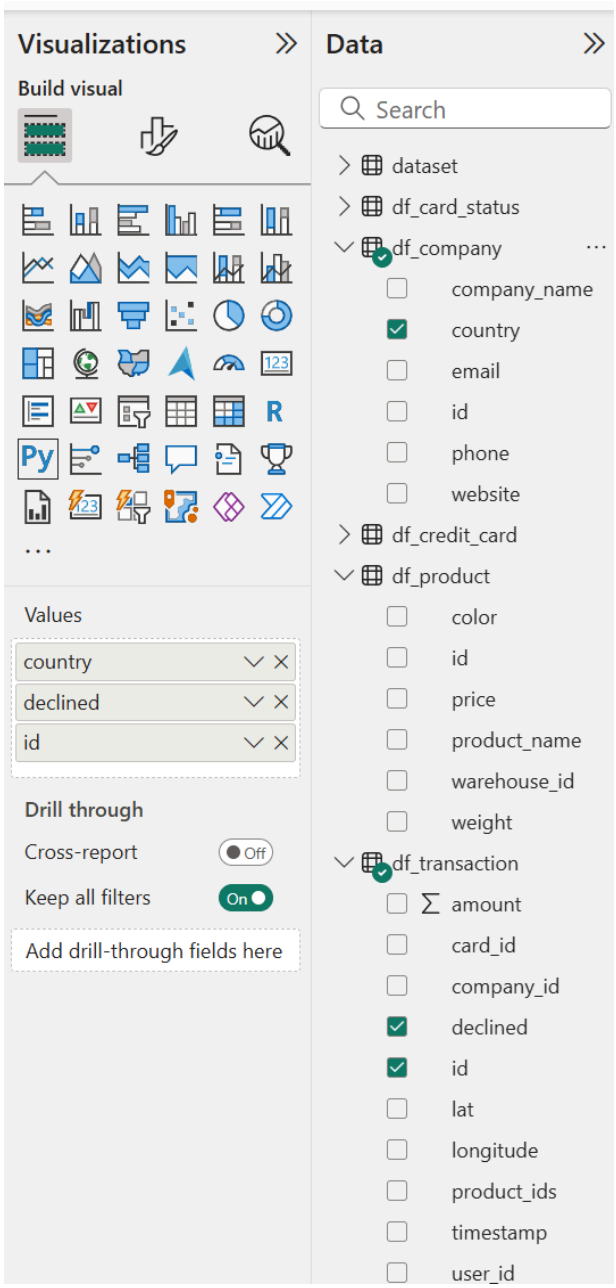
Plot the graphic



- Exercici 5

Dues variables categòriques.

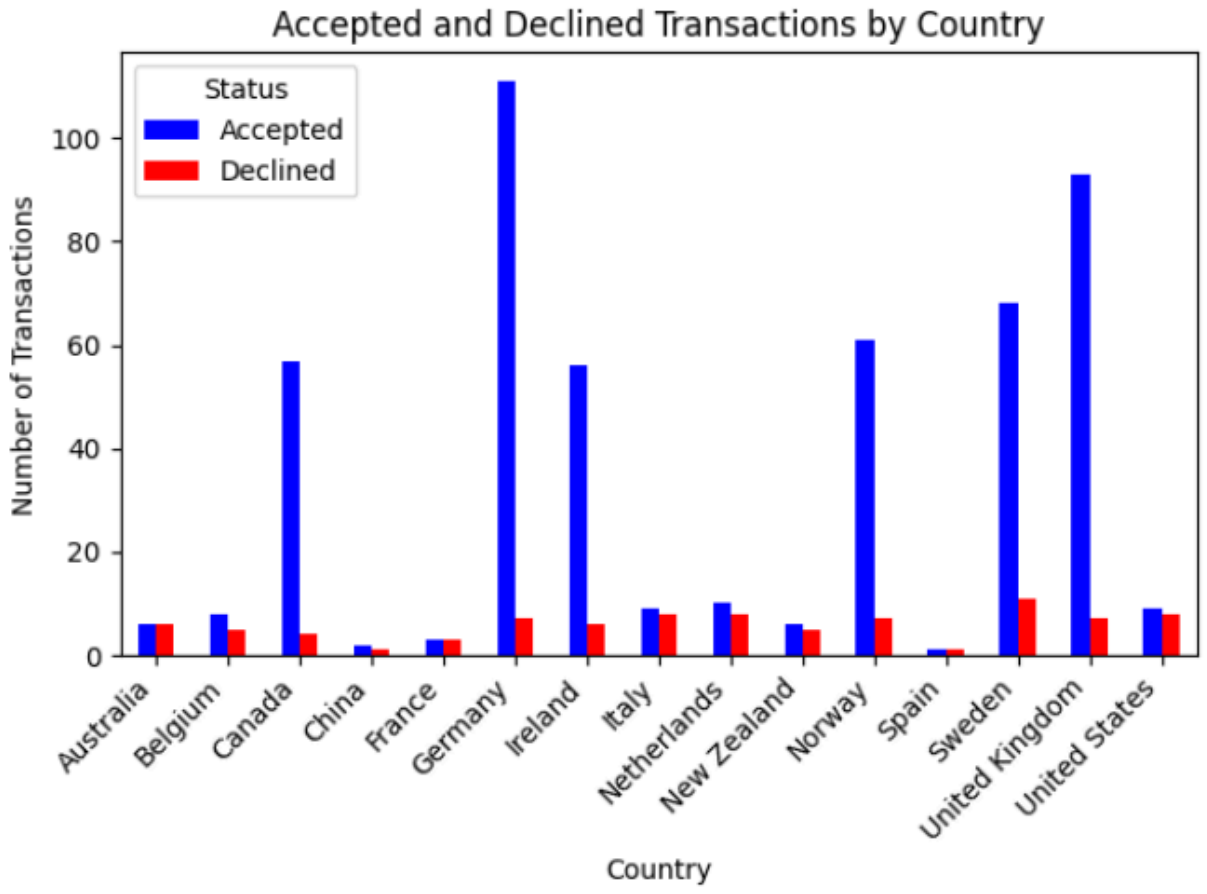
Select the columns to be used.



Run the script

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

# Replace numeric with categorical status
dataset['status'] = dataset['declined'].apply(lambda x: 'Declined' if x == 1 else 'Accepted')

# Group and pivot
df_country_status = dataset.groupby(['country', 'status']).size().unstack(fill_value=0)

# Plot
ax = df_country_status.plot(kind='bar', color=['blue', 'red'])

plt.title('Accepted and Declined Transactions by Country')
plt.xlabel('Country')
plt.ylabel('Number of Transactions')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Status', loc='upper left')
plt.tight_layout()
plt.show()
```

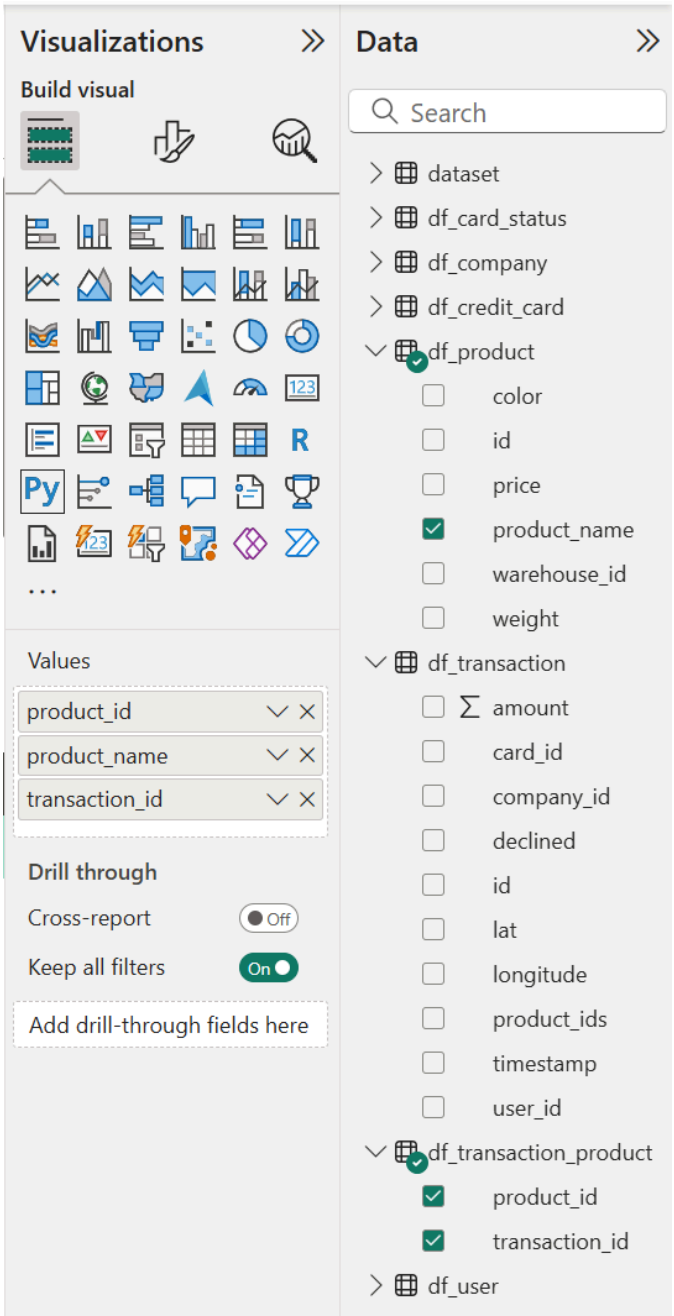
Plot the graphic



- Exercici 6

Tres variables.

Select the columns to be used in the graphic.

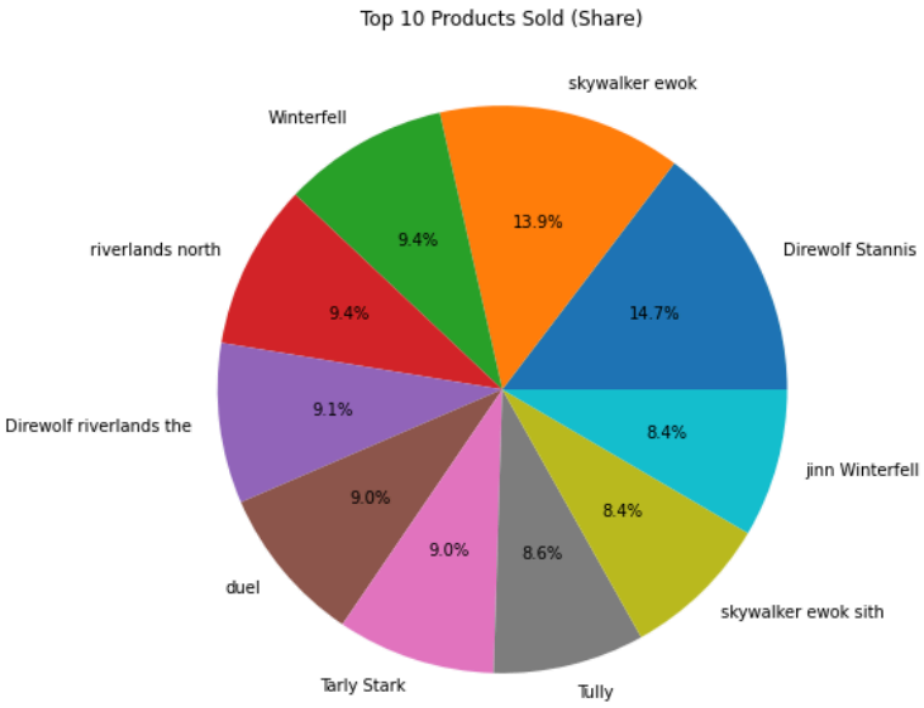


Run the script to plot the graphic

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

# Drop rows with missing data
dataset = dataset.dropna(subset=['product_id', 'transaction_id', 'product_name', 'amount'])

# Group by product name and sum the amount
product_sales = dataset.groupby('product_name')['amount'].sum().reset_index()

# Sort and get top 10
top_10 = product_sales.sort_values(by='amount', ascending=False).head(10)
```

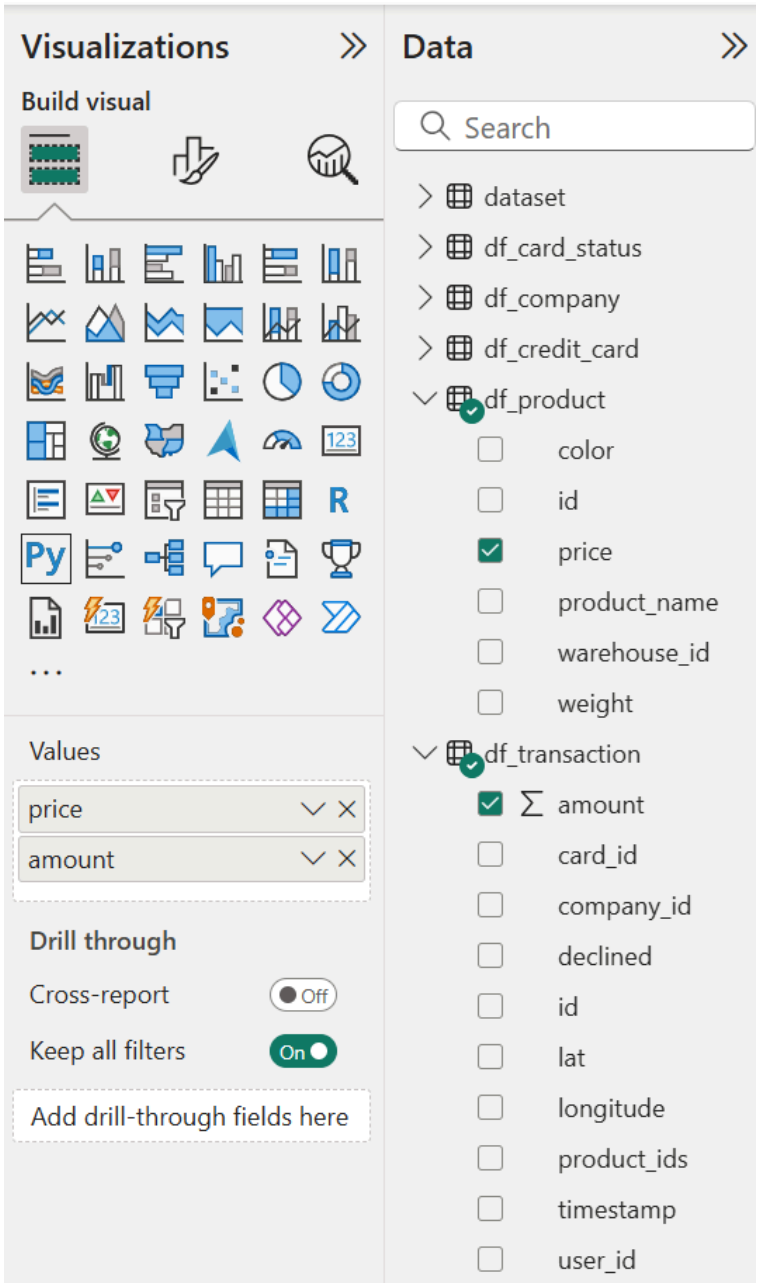
```python

top_10.set_index('product_name')['amount'].plot.pie(
    autopct='%1.1f%%',
    figsize=(8, 8),
    ylabel=''
)

plt.title("Top 10 Products by Total Amount Sold")
plt.tight_layout()
plt.show()
```

Plot the graphic



- Exercici 7

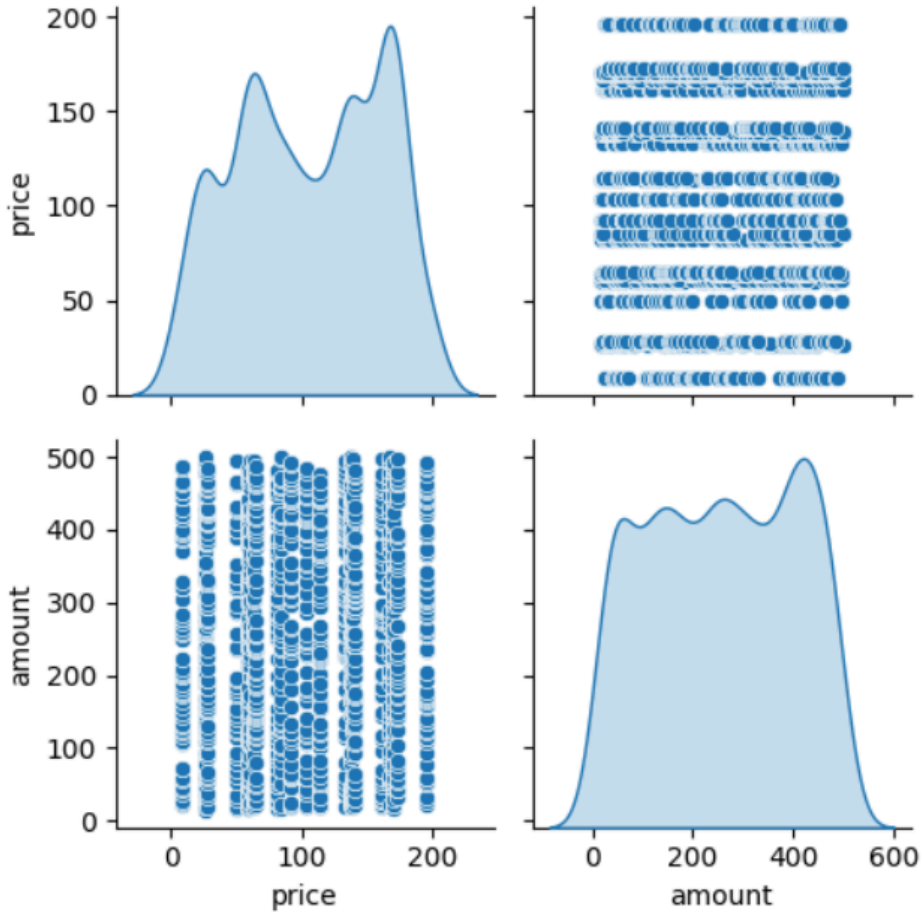
Graficar un Pairplot.



Run the script to create the pairplot.

```
 7
 8 import pandas as pd
 9 import seaborn as sns
10 import matplotlib.pyplot as plt
11
12 df_clean = dataset.dropna(subset=['price', 'amount'])
13
14 sns.pairplot(df_clean, vars=['price', 'amount'], diag_kind='kde')
15 plt.tight_layout()
16 plt.show()
```

Plot the graphic.

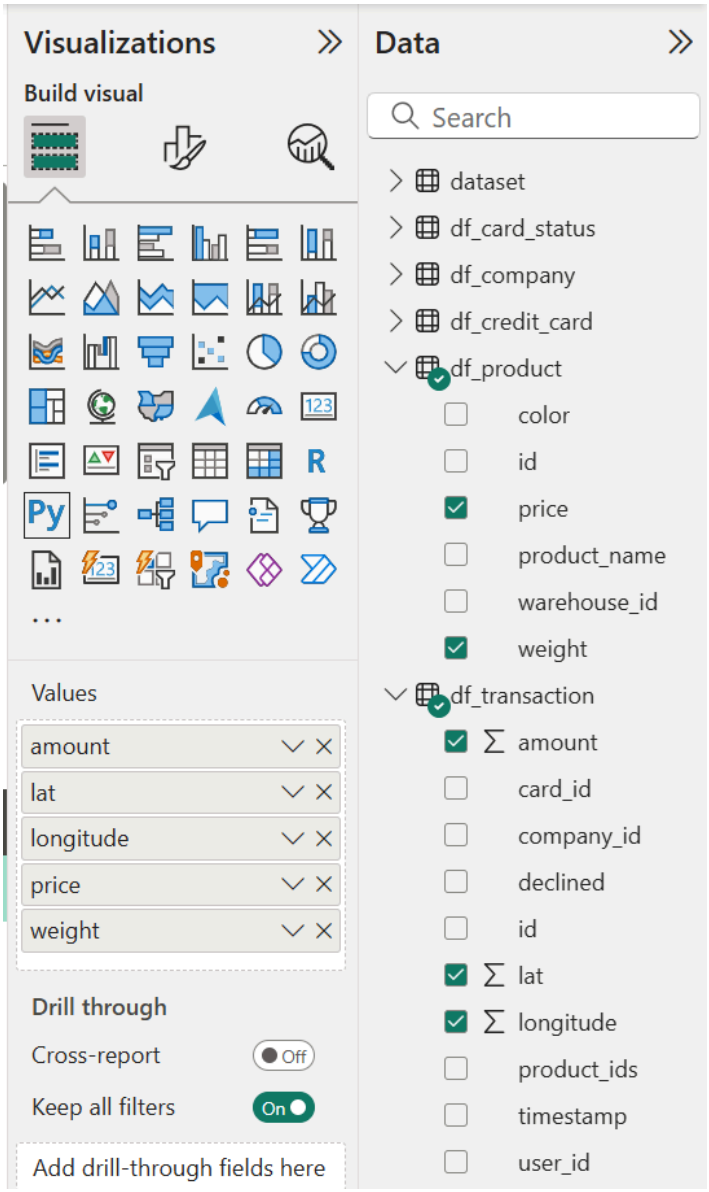


NIVELL 2

- Exercici 1

Correlació de totes les variables numèriques.

Select the numeric columns to be used



Run the script

```
 8 import pandas as pd
 9 import seaborn as sns
10 import matplotlib.pyplot as plt
11
12 dataset = dataset.dropna(subset=['amount', 'lat', 'longitude', 'price', 'weight'])
13
14 # Select the numeric columns
15 numeric_cols = dataset[['amount', 'lat', 'longitude', 'price', 'weight']]
16
17 # Compute the correlation matrix
18 corr = numeric_cols.corr()
19
20 # Plot heatmap
21 sns.heatmap(corr, annot=True, cmap='coolwarm')
22 plt.title('Correlation Between Numeric Variables')
23 plt.tight_layout()
24 plt.show()
```

Plot the graphic