

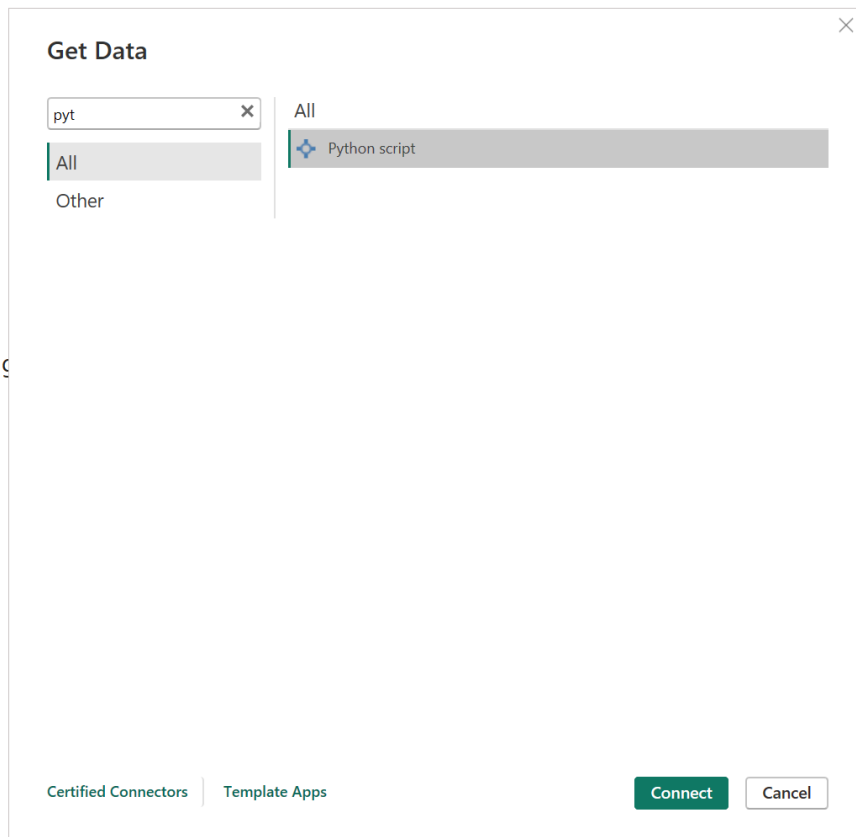
SPRINT 8.2

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.

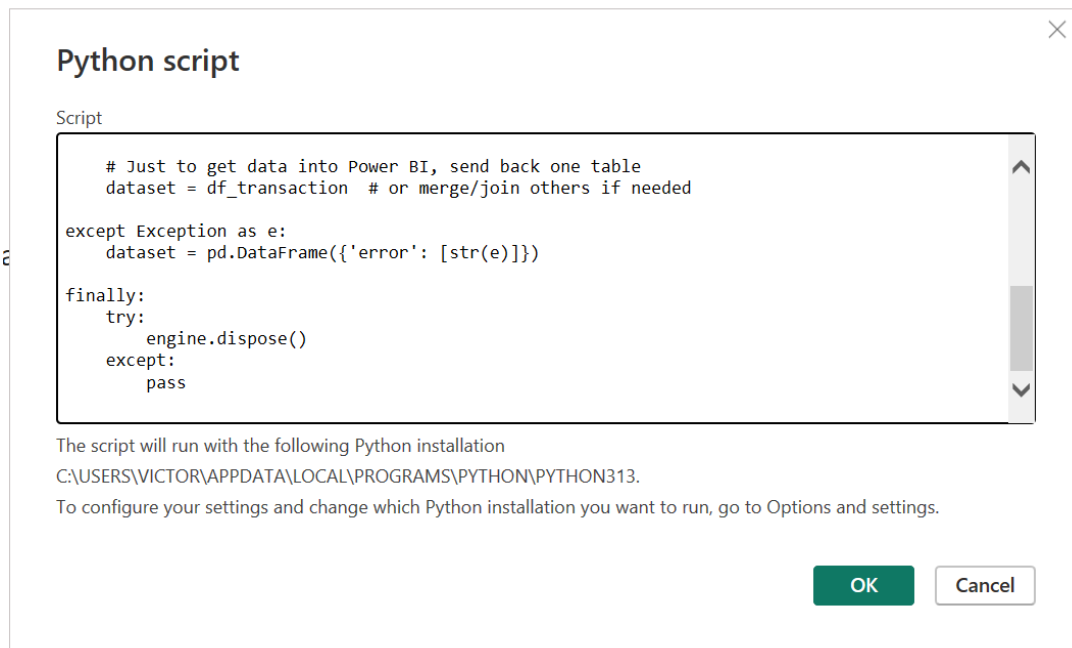
Objectius

- 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 reproduccibilitat 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.



```
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.

Navigator

Display Options ▾

Python [8]

☒

dataset

☒

df_card_status

☒

df_company

☒

df_credit_card

☒

df_product

☒

df_transaction

☒

df_transaction_product

☒

df_user

df_user

id	name	surname	phone	email
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonr
10	Robert	Mccarthy	(324) 746-6771	fermentum@protonmail
100	Melodie	Mclean	1-677-221-7152	risus.varius@google.ca
101	Sarah	Beck	(358) 691-4345	vitae.risus@aol.couk
102	Jasper	Landry	1-397-765-1118	consectetur.euismod@
103	Upton	Chavez	(227) 785-6484	euismod.est@aol.ca
104	Martha	Barlow	(732) 326-5448	vulputate@hotmail.net
105	Hashim	Rose	(858) 313-6727	urna@icloud.com
106	Tanner	Valenzuela	1-346-421-3135	nascetur.ridiculus@goo
107	Victor	Valencia	(239) 569-1938	non.enim@hotmail.couk
108	Germaine	Suarez	1-931-750-6983	risus@icloud.com
109	Raven	Reynolds	(667) 453-9731	sed@aol.com
11	Joan	Baird	(981) 429-8106	et@outlook.net
110	Neil	Powers	(864) 881-6737	nec.metus@aol.edu
111	Astra	Baldwin	1-643-565-3266	adipiscing.ligula.aenean@
112	Ryder	Cole	1-572-759-8544	nec.enim.nunc@protonr
113	Risa	Frost	1-712-488-5451	neque.pellentesque@ou
114	Jasmine	Castro	1-512-143-0648	lorem@google.ca
115	Urielle	Holman	1-424-793-4354	leo@google.ca
116	Sacha	Compton	(265) 342-4775	sed.dictum.proin@yaho
117	Halla	Pearson	(681) 698-7518	lacus.etiam@protonmail
118	Brooke	Jensen	(124) 739-9067	purus.mauris.a@icloud.c
119	Damian	Mcgee	(712) 572-8735	neque.nullam@hotmail.i

Load

Transform Data

Cancel

Here, we can see the tables are showing.

The screenshot displays a user interface for a data visualization tool, divided into two main sections: **Visualizations** and **Data**.

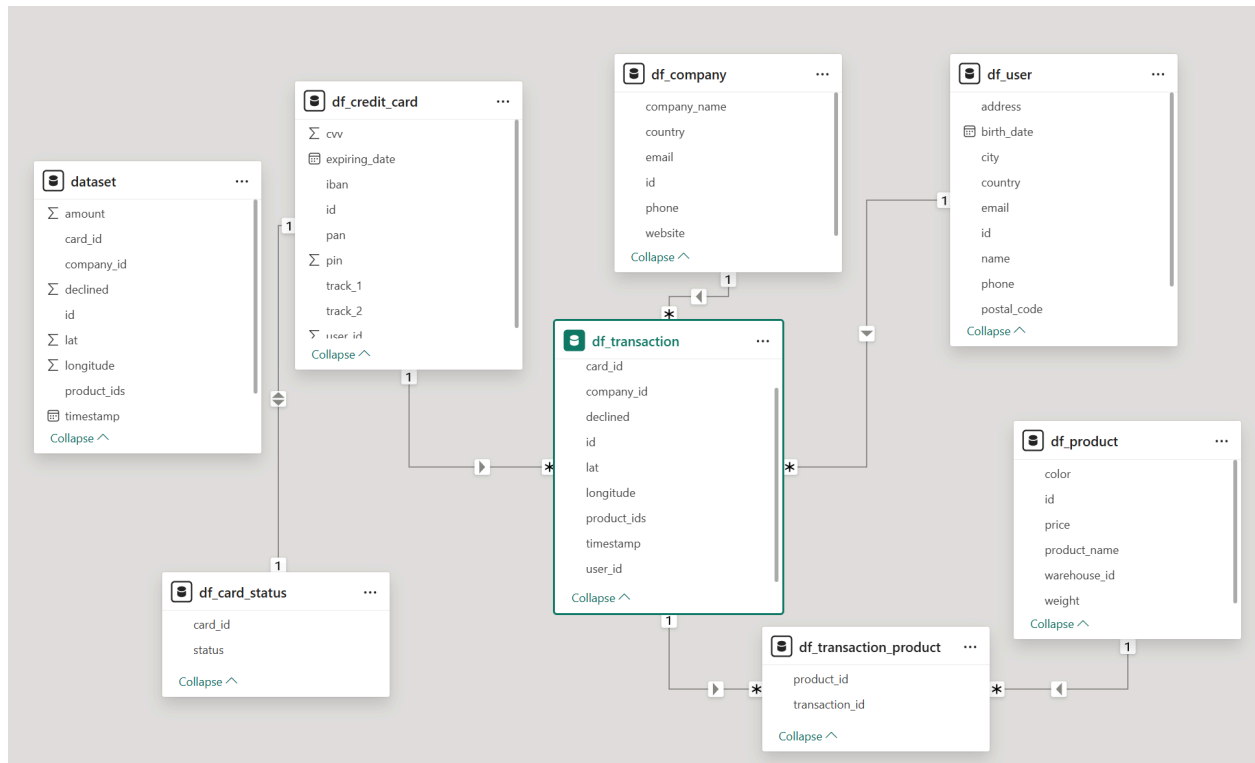
Visualizations Panel (Left):

- Build visual:** A section with a grid icon and a pencil icon.
- Visual Type Grid:** A collection of icons representing various chart types, including bar charts, line charts, pie charts, and maps.
- Values:** A dashed box containing the text "Add data fields here".
- Drill through:** A section with two toggle switches:
 - Cross-report:** Set to **Off**.
 - Keep all filters:** Set to **On**.
- Add drill-through fields here:** A dashed box for additional drill-through fields.

Data Panel (Right):

- Search:** A search bar with a magnifying glass icon.
- Table List:** A list of data tables, each preceded by a table icon and a greater-than symbol (>):
 - dataset
 - df_card_status
 - df_company
 - df_credit_card
 - df_product
 - df_transaction
 - df_transaction_product
 - df_user

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

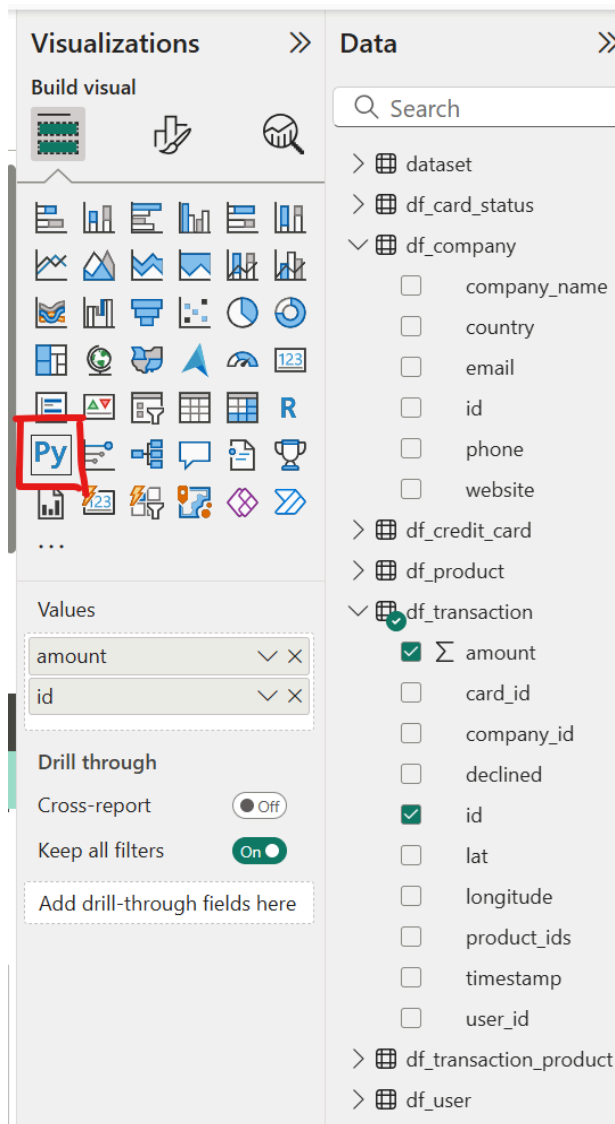
- Exercici 1

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.

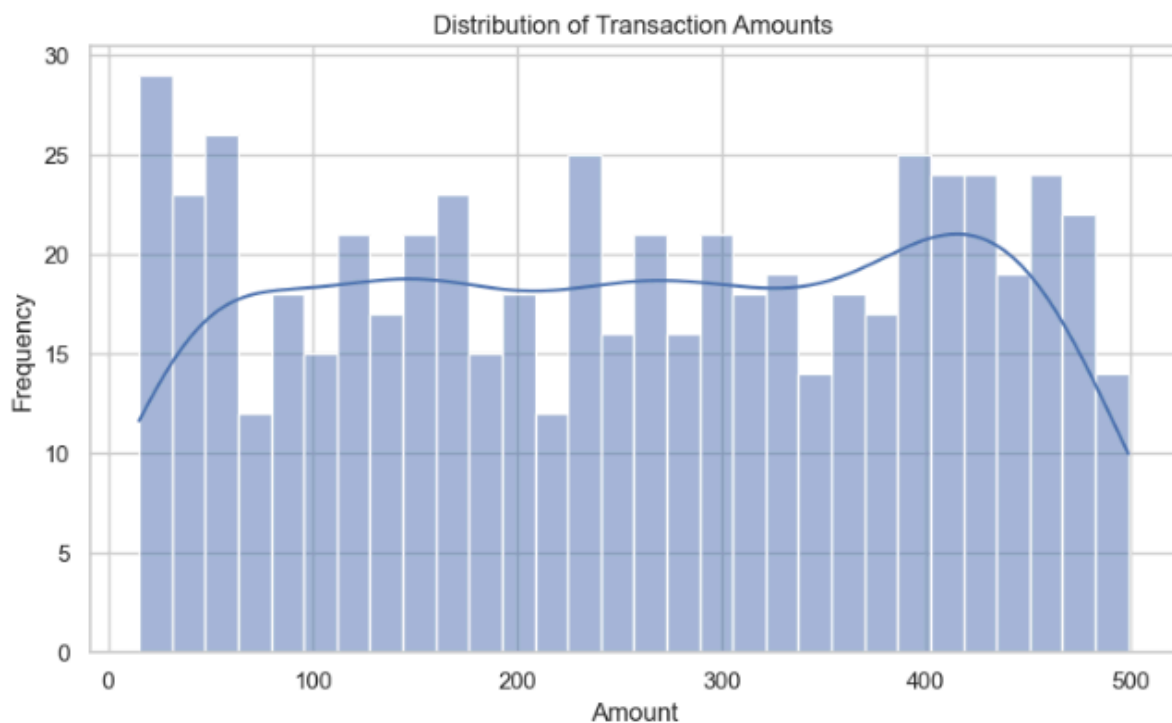
In this exercise we added the column id from the table transaction.



Script used to create the visual from exercise 1.

```
6 # Paste or type your script code here:
7 import seaborn as sns
8 import matplotlib.pyplot as plt
9 import pandas as pd
10
11 # Remove rows with missing values in relevant columns
12 dataset = dataset.dropna(subset=['id', 'amount'])
13
14 # Set Seaborn style
15 sns.set(style='whitegrid')
16
17 plt.figure(figsize=(8, 5))
18 sns.histplot(data=dataset, x='amount', bins=30, kde=True)
19
20 plt.title('Distribution of Transaction Amounts')
21 plt.xlabel('Amount')
22 plt.ylabel('Frequency')
23 plt.grid(True)
```

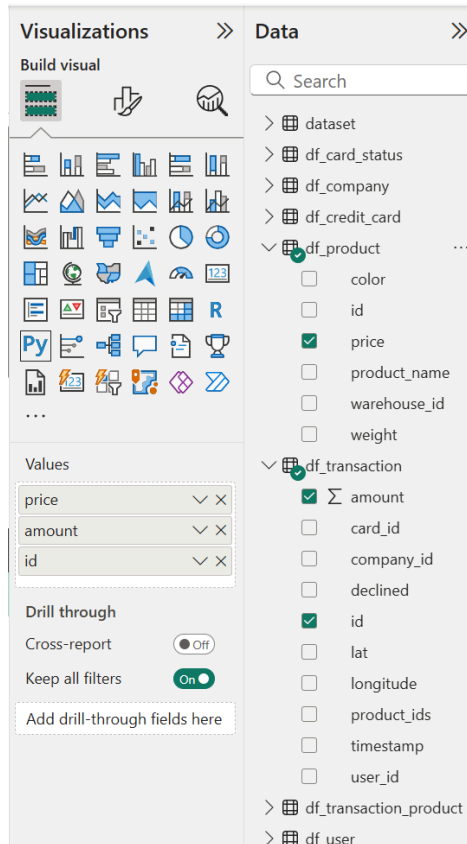
Plot the graphic.



- Exercici 2

Dues variables numèriques.

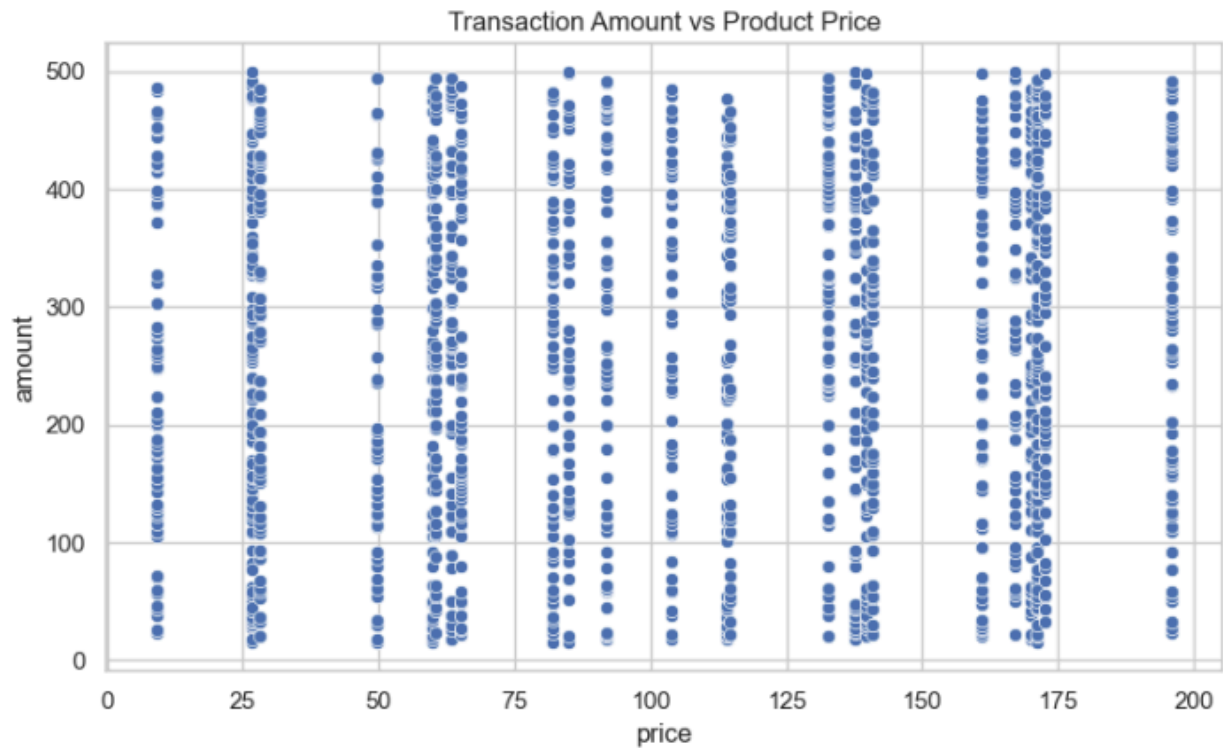
Select the columns we will be using. Here we also selected the column id from the transaction table so we can add a unique value and prevent lines from being deleted.



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()
17
```

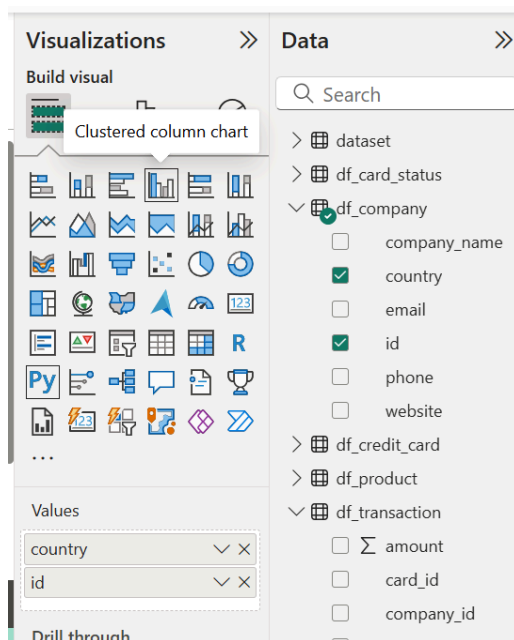

Plot the graphic



- Exercici 3

Una variable categòrica.

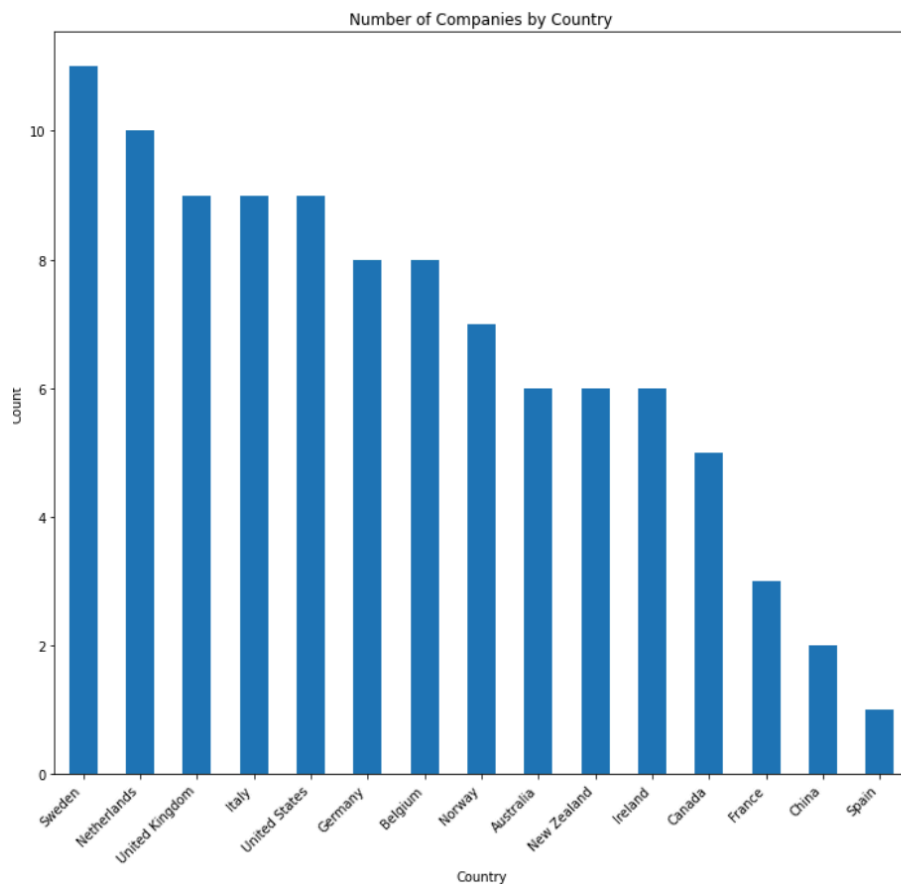
Selecting the columns. In this case we also had to select the column id, so the graphic could work.



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

```
Python script editor
⚠ Duplicate rows will be removed from the data.
3 # dataset = pandas.DataFrame(country, id)
4 # dataset = dataset.drop_duplicates()
5
6 # Paste or type your script code here:
7 import pandas as pd
8 import matplotlib.pyplot as plt
9
10 # Count number of companies per country
11 country_counts = dataset.groupby('country').size().sort_values(ascending=False)
12
13 # Plot
14 country_counts.plot(kind='bar')
15 plt.title('Number of Companies by Country')
16 plt.xlabel('Country')
17 plt.ylabel('Count')
18 plt.xticks(rotation=45, ha='right')
19 plt.tight_layout()
20 plt.show()
```

Plot the graphic



- Exercici 4

Una variable categòrica i una numèrica.

Select the columns

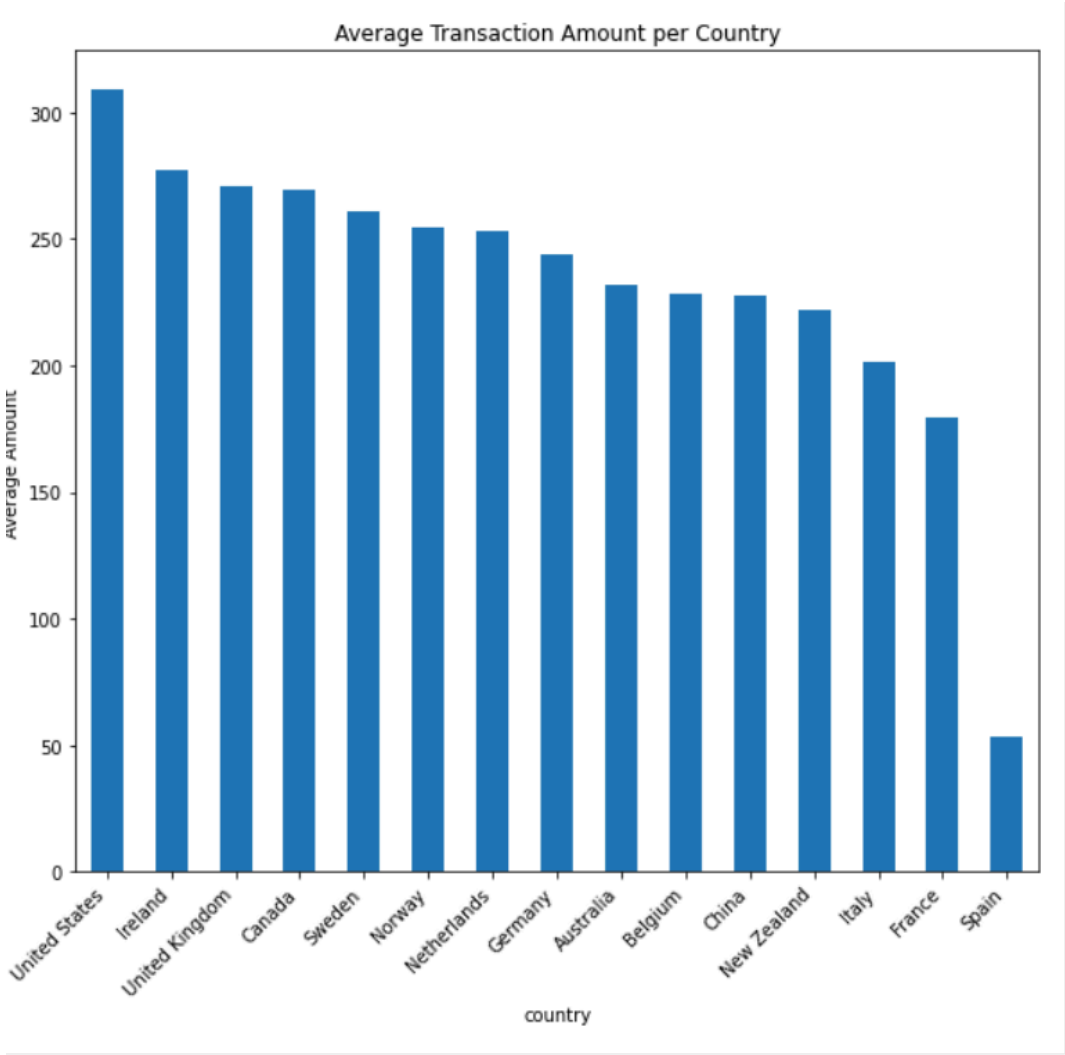
The screenshot shows a data visualization tool interface. On the left, the 'Visualizations' panel has a 'Build visual' section with a 'Clustered bar chart' icon selected. Below this is a 'Values' section with a list containing 'country' and 'amount'. At the bottom of this panel are 'Drill through' and 'Cross-report' (set to 'Off') options, and a 'Keep all filters' (set to 'On') toggle. On the right, the 'Data' panel shows a tree structure of datasets: 'dataset', 'df_card_status', 'df_company' (expanded), 'df_credit_card', 'df_product', and 'df_transaction' (expanded). Under 'df_company', 'country' is selected with a checkmark. Under 'df_transaction', 'amount' is selected with a checkmark and a summation symbol (Σ). Other fields like 'company_name', 'email', 'id', 'phone', 'website', 'card_id', 'company_id', 'declined', 'lat', and 'longitude' are listed but not selected.

Run the script

```
Python script editor

⚠ Duplicate rows will be removed from the data.
6 # Paste or type your script code here:
7 import pandas as pd
8 import matplotlib.pyplot as plt
9
10 # Drop missing data
11 dataset = dataset.dropna(subset=['country', 'amount'])
12
13 # Drop rows with failed conversion
14 dataset = dataset.dropna(subset=['amount'])
15
16 # Group and calculate average
17 avg_amount = dataset.groupby('country')['amount'].mean().sort_values(ascending=False)
18
19 avg_amount.plot(kind='bar', title='Average Transaction Amount per Country')
20 plt.ylabel('Average Amount')
21 plt.xticks(rotation=45, ha='right')
22 plt.tight_layout()
23 plt.show()
24
```

Plot the graphic



- Exercici 5

Dues variables categòriques.

Select the columns to be used.

The screenshot shows a data visualization tool interface with two main panels: 'Visualizations' and 'Data'.

Visualizations Panel:

- Build visual:** Contains icons for various chart types (bar, line, pie, etc.).
- Values:** A list of selected fields: 'country', 'declined', and 'id'. Each field has a dropdown arrow and a close button (X).
- Drill through:** Includes a 'Cross-report' toggle (set to 'Off') and a 'Keep all filters' toggle (set to 'On'). Below these is a text box labeled 'Add drill-through fields here'.

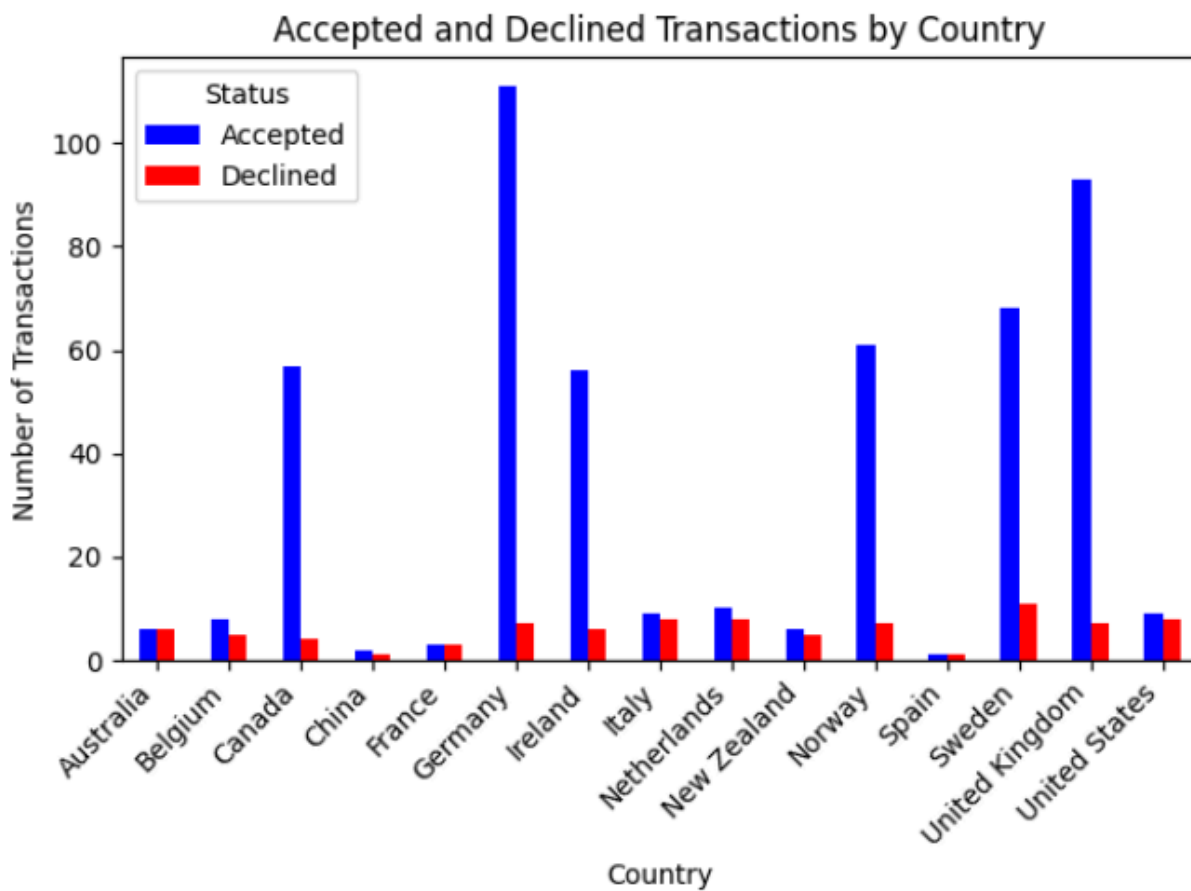
Data Panel:

- Search:** A search bar at the top.
- Dataset List:** A list of datasets with expandable/collapsible arrows:
 - dataset** (expanded)
 - df_card_status** (expanded)
 - df_company** (expanded, marked with a green checkmark):
 - ☐ company_name
 - ☒ country
 - ☐ email
 - ☐ id
 - ☐ phone
 - ☐ website
 - df_credit_card** (expanded)
 - df_product** (expanded):
 - ☐ color
 - ☐ id
 - ☐ price
 - ☐ product_name
 - ☐ warehouse_id
 - ☐ weight
 - df_transaction** (expanded, marked with a green checkmark):
 - ☐ Σ amount
 - ☐ card_id
 - ☐ company_id
 - ☒ declined
 - ☒ id
 - ☐ lat
 - ☐ longitude
 - ☐ product_ids
 - ☐ timestamp
 - ☐ user_id

Run the script

```
/ import pandas as pd
8 import matplotlib.pyplot as plt
9
10 # Replace numeric with categorical status
11 dataset['status'] = dataset['declined'].apply(lambda x: 'Declined' if x == 1 else 'Accepted')
12
13 # Group and pivot
14 df_country_status = dataset.groupby(['country', 'status']).size().unstack(fill_value=0)
15
16 # Plot
17 ax = df_country_status.plot(kind='bar', color=['blue', 'red'])
18
19 plt.title('Accepted and Declined Transactions by Country')
20 plt.xlabel('Country')
21 plt.ylabel('Number of Transactions')
22 plt.xticks(rotation=45, ha='right')
23 plt.legend(title='Status', loc='upper left')
24 plt.tight_layout()
25 plt.show()
```

Plot the graphic



- Exercici 6
Tres variables.

Select the columns to be used in the graphic.

The image shows a user interface for building a visualization. It is divided into two main sections: 'Visualizations' on the left and 'Data' on the right.

Visualizations Panel:

- Build visual:** Contains icons for different chart types (bar, line, pie, etc.).
- Values:** A list of fields selected for the visualization: 'product_id', 'product_name', and 'transaction_id'. Each field has a dropdown arrow and a close button.
- Drill through:** Includes a 'Cross-report' toggle (set to 'Off') and a 'Keep all filters' toggle (set to 'On').
- Add drill-through fields here:** A dashed box indicating where to add additional drill-through fields.

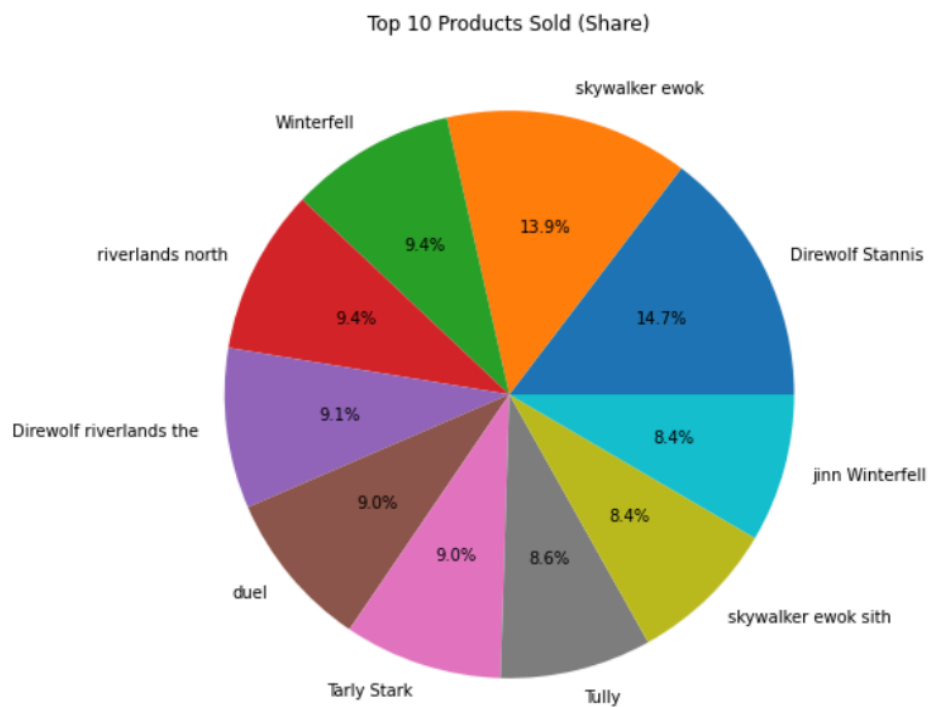
Data Panel:

- Search:** A search bar at the top.
- Dataset List:** A list of datasets with expandable/collapsible arrows:
 - dataset** (expanded)
 - df_card_status** (expanded)
 - df_company** (expanded)
 - df_credit_card** (expanded)
 - df_product** (expanded, selected with a green checkmark):
 - ☐ color
 - ☐ id
 - ☐ price
 - ☒ product_name
 - ☐ warehouse_id
 - ☐ weight
 - df_transaction** (expanded):
 - ☐ Σ amount
 - ☐ card_id
 - ☐ company_id
 - ☐ declined
 - ☐ id
 - ☐ lat
 - ☐ longitude
 - ☐ product_ids
 - ☐ timestamp
 - ☐ user_id
 - df_transaction_product** (expanded, selected with a green checkmark):
 - ☒ product_id
 - ☒ transaction_id
 - df_user** (expanded)

Run the script to plot the graphic

```
8 import pandas as pd
9 import matplotlib.pyplot as plt
10
11 # Drop rows with missing data
12 dataset = dataset.dropna(subset=['product_id', 'transaction_id', 'product_name', 'amount'])
13
14 # Group by product name and sum the amount
15 product_sales = dataset.groupby('product_name')['amount'].sum().reset_index()
16
17 # Sort and get top 10
18 top_10 = product_sales.sort_values(by='amount', ascending=False).head(10)
19
20 top_10.set_index('product_name')['amount'].plot.pie(
21     autopct='%1.1f%%',
22     figsize=(8, 8),
23     ylabel=''
24 )
25
26 plt.title("Top 10 Products by Total Amount Sold")
27 plt.tight_layout()
28 plt.show()
29
```

Plot the graphic



- Exercici 7

Graficar un Pairplot.

The image shows a user interface for creating data visualizations, divided into two main sections: 'Visualizations' and 'Data'.

Visualizations Panel:

- Build visual:** Contains icons for various chart types (bar, line, area, pie, etc.) and a search icon.
- Values:** A list of selected fields for the visualization. Currently, 'price' and 'amount' are listed, each with a dropdown arrow and a close button (X).
- Drill through:** A section with two toggle switches: 'Cross-report' (set to 'Off') and 'Keep all filters' (set to 'On'). Below these is a text input field labeled 'Add drill-through fields here'.

Data Panel:

- Search:** A search bar at the top.
- Dataset List:** A list of available datasets with expandable/collapsible arrows:
 - > dataset
 - > df_card_status
 - > df_company
 - > df_credit_card
 - ▼ df_product (selected with a green checkmark)
 - ☐ color
 - ☐ id
 - ☒ price
 - ☐ product_name
 - ☐ warehouse_id
 - ☐ weight
 - ▼ df_transaction (selected with a green checkmark)
 - ☒ Σ amount
 - ☐ card_id
 - ☐ company_id
 - ☐ declined
 - ☐ id
 - ☐ lat
 - ☐ longitude
 - ☐ product_ids
 - ☐ timestamp
 - ☐ user_id

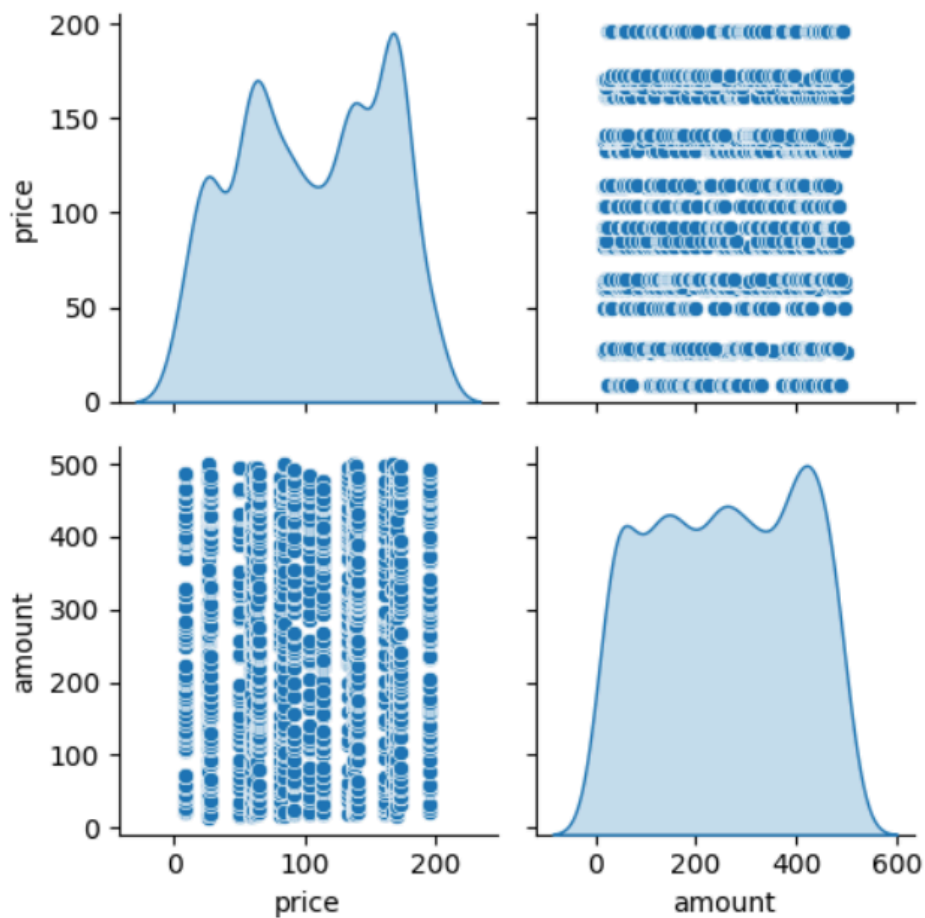
Run the script to create the pairplot.

```

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

The screenshot shows a data visualization tool interface with two main panels: 'Visualizations' and 'Data'.

Visualizations Panel:

- Build visual:** A section with icons for different chart types (bar, line, pie, etc.) and a search bar.
- Values:** A list of selected fields: amount, lat, longitude, price, and weight. Each field has a dropdown arrow and a close button (X).
- Drill through:** A section with two toggle switches: 'Cross-report' (set to Off) and 'Keep all filters' (set to On). Below these is a text input field labeled 'Add drill-through fields here'.

Data Panel:

- Search:** A search bar at the top.
- Dataset List:** A list of datasets with expand/collapse arrows and checkboxes:
 - dataset (expanded)
 - df_card_status (expanded)
 - df_company (expanded)
 - df_credit_card (expanded)
 - df_product (expanded, checked with a green checkmark)
 - color (unchecked)
 - id (unchecked)
 - price (checked with a green checkmark)
 - product_name (unchecked)
 - warehouse_id (unchecked)
 - weight (checked with a green checkmark)
 - df_transaction (expanded, checked with a green checkmark)
 - Σ amount (checked with a green checkmark)
 - card_id (unchecked)
 - company_id (unchecked)
 - declined (unchecked)
 - id (unchecked)
 - Σ lat (checked with a green checkmark)
 - Σ longitude (checked with a green checkmark)
 - product_ids (unchecked)
 - timestamp (unchecked)
 - user_id (unchecked)

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

