Tarea S8.02. Power BI amb Python

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.

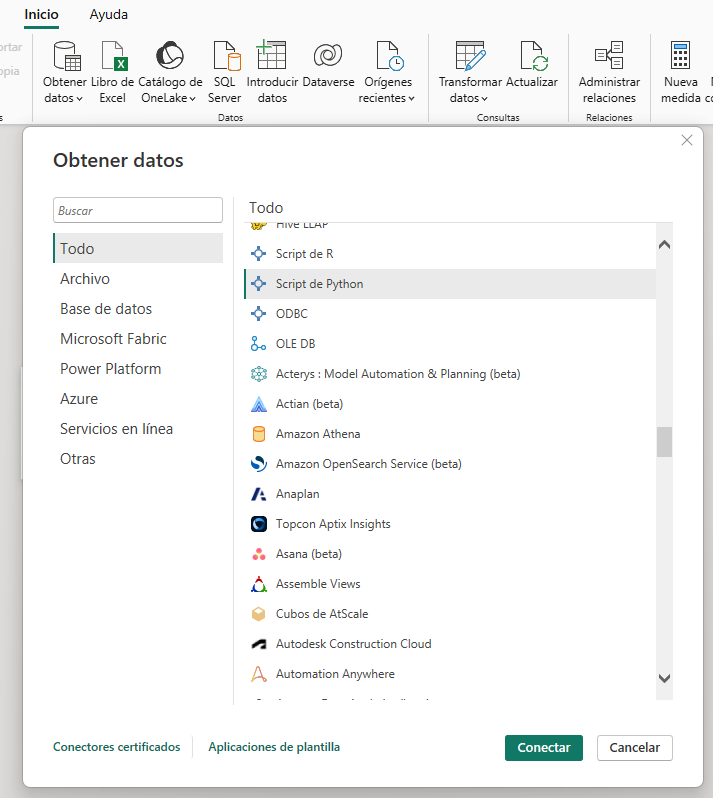
Para conectar la base de datos a Power BI mediante scripts de Python, primero guardo las tablas obtenidas al conectar Python con MySQL como archivos .csv, y así poder cargarlos directamente en Power BI con un script directo de Python, sin necesidad de conectar Power BI a Python y Python a MySQL.

*# Para poder trabajar con scripts de python en PowerBI, guardo los dataframes en un csv*

for table\_name, df in dict\_transactions.items():

    df.to\_csv(f"{table\_name}.csv", index=False)

Conexión en Power BI mediante scripts de Python:



Script de Python para cargar los DataFrames:

*#En PowerBI:*

import pandas as pd

*# Cargar los archivos CSV como DataFrames*

transactions = pd.read\_csv(r"C:\users\maria\ITAcademy\Sprint 8 - Visualitzacions en Python Scripts en Power BI\data\transactions.csv")

card\_status = pd.read\_csv(r"C:\Users\maria\ITAcademy\Sprint 8 - Visualitzacions en Python Scripts en Power BI\data\card\_status.csv")

companies = pd.read\_csv(r"C:\Users\maria\ITAcademy\Sprint 8 - Visualitzacions en Python Scripts en Power BI\data\companies.csv")

credit\_cards = pd.read\_csv(r"C:\Users\maria\ITAcademy\Sprint 8 - Visualitzacions en Python Scripts en Power BI\data\credit\_cards.csv")

data\_users = pd.read\_csv(r"C:\Users\maria\ITAcademy\Sprint 8 - Visualitzacions en Python Scripts en Power BI\data\data\_users.csv")

products = pd.read\_csv(r"C:\Users\maria\ITAcademy\Sprint 8 - Visualitzacions en Python Scripts en Power BI\data\products.csv")

trans\_prod = pd.read\_csv(r"C:\Users\maria\ITAcademy\Sprint 8 - Visualitzacions en Python Scripts en Power BI\data\trans\_prod.csv")

*# Asegurar que los datos se leen correctamente*

print(transactions.head(2))

print(card\_status.head(2))

print(companies.head(2))

print(credit\_cards.head(2))

print(data\_users.head(2))

print(products.head(2))

print(trans\_prod.head(2))

# Nivel 1

Els 7 exercicis del nivell 1 de la tasca 01.

**Ejercicio 1** –

Una variable numèrica.

**1º** Histograma que refleja la distribución de las transacciones realizadas. Las transacciones se agrupan en rangos y no se observa una distribución normal de los valores.

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))

plt.hist(x=dataset["amount"], bins=15, color="skyblue", edgecolor="black")

plt.title("Total accepted transactions")

plt.xlabel("Total sales ($)")

plt.ylabel("Frecuency")

plt.grid(axis='y', linestyle='--', alpha=0.5)

plt.show()

A graph of sales

AI-generated content may be incorrect.

**2º** Boxplot del total de transacciones aceptadas. Este gráfico se realizó con los mismos datos del anterior, permite visualizar la dispersión de los valores, detectar la presencia de valores atípicos o outliers, e identifica la mediana y los rangos intercuartílicos.

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(5, 5))

sns.boxplot(data=dataset, y="amount", color="darkseagreen")

plt.title("Total accepted transactions")

plt.ylabel("Total sales ($)")

plt.ylim(0, 600)

plt.grid(True, linestyle='--', alpha=0.25)

plt.show()

A graph with a green rectangle

AI-generated content may be incorrect.

**Ejercicio 2** –

Dues variables numèriques.

En primer lugar, se realizó un scatterplot para representar la relación entre el valor total de las transacciones y el precio de los productos implicados en las mismas. La dispersión de los puntos es muy elevada y no se observa ninguna correlación entre el precio del producto y el total de la transacción.

Debido a una incompatibilidad entre las aplicaciones, Power BI no es capaz de representar los gráficos realizados con Plotly, aunque estos se generan y visualizan automáticamente en el navegador web.

import pandas as pd

import plotly.graph\_objects as go

*# Calcular el monto total de la transacción correctamente*

total\_transaction\_amount = dataset.groupby('transaction\_id')['amount'].sum() / dataset.groupby('transaction\_id').size()

*# Calcular el precio promedio del producto*

product\_price = dataset.groupby('product\_id')['price'].mean()

*# Mapear valores a dataset*

dataset['total\_transaction\_amount'] = dataset['transaction\_id'].map(total\_transaction\_amount)

dataset['product\_price'] = dataset['product\_id'].map(product\_price)

*# Eliminar posibles NaN generados en el mapeo*

dataset = dataset.dropna(subset=['total\_transaction\_amount', 'product\_price'])

*# Definir X e Y*

X = dataset['total\_transaction\_amount']

Y = dataset['product\_price']

*# Crear el gráfico*

fig = go.Figure()

fig.add\_trace(go.Scatter(

    x=X,

    y=Y,

    mode='markers',

    marker=dict(

        size=10,

        color='darkseagreen',

        opacity=0.7,

        line=dict(width=1, color='black') # Bordes

    ),

    text=[f"Total sale = $ {sale:.2f}, Product price = $ {price:.2f}" for sale, price in zip(X, Y)]

))

*# Personalizar el diseño*

fig.update\_layout(

    title='Product price ($) vs. Total sales amount ($)',

    xaxis\_title='Sales amount by transaction ($)',

    yaxis\_title='Product price ($)',

    width=700,

    height=700)

*# Mostrar el gráfico*

fig.show()

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

Al adaptar la visualización del gráfico a Matplotlib, este se genera correctamente en Power BI.

import pandas as pd

import matplotlib.pyplot as plt

*# Unir transacciones con productos - está todo en dataset*

total\_transaction\_amount = dataset.groupby(by=['transaction\_id'])['amount'].sum()/dataset['transaction\_id'].value\_counts()

product\_price = dataset.groupby('product\_id')['price'].mean()

*# Tengo que relacionar cada id con su valor total:*

dataset['total\_transaction\_amount'] = dataset['transaction\_id'].map(total\_transaction\_amount)

dataset['product\_price'] = dataset['product\_id'].map(product\_price)

plt.scatter(dataset['total\_transaction\_amount'], dataset['product\_price'], color='darkseagreen')

plt.title('Product price ($) vs. Total sales amount ($)')

plt.xlabel('Sales amount by transaction ($)')

plt.ylabel('Product price ($)')

plt.show()

A graph with green dots

AI-generated content may be incorrect.

**Ejercicio 3** –

Una variable categòrica.

Debido a los problemas de compatibilidad entre Plotly y Power BI, una vez más, el script se ejecuta directamente en el navegador web.

import pandas as pd

import numpy as np

import plotly.express as px

*# Agrupo por país y sumo las ventas*

country\_sum\_accepted\_sales = dataset.groupby('country')['amount'].sum().reset\_index()

total\_accepted\_amount = country\_sum\_accepted\_sales['amount'].sum()

country\_sum\_accepted\_sales['percentage'] = round(country\_sum\_accepted\_sales['amount'] / total\_accepted\_amount \* 100, 2)

*# Crear el treemap con plotly*

fig2 = px.treemap(country\_sum\_accepted\_sales,

                 path=['country'],

                 values='amount',

                 title='Total sales by country',

                 labels={'amount': 'Total sales ($)'},

                 hover\_data={'amount': True, 'percentage': True})

*# Para modificar el texto del hover, encontré este ejemplo que lo adapto a mis datos*

fig2.update\_traces(hovertemplate=

    "<b>%{label}</b><br>" +

    "Total sales: $ %{value}<br>" +

    "Percentage: %{customdata[1]} %")

*# Mostrar el gráfico*

fig2.show()

A screenshot of a computer

AI-generated content may be incorrect.

Por otro lado, cuando el gráfico se realiza como “Squarify” de Matplotlib, se visualiza correctamente.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import squarify

*# Agrupo por país y sumo las ventas*

country\_sum\_accepted\_sales = dataset.groupby('country')['amount'].sum().reset\_index()

total\_accepted\_amount = country\_sum\_accepted\_sales['amount'].sum()

percentage = [amount / total\_accepted\_amount \* 100 for amount in country\_sum\_accepted\_sales['amount']]     *# list comprehension*

S = country\_sum\_accepted\_sales['amount']

L = country\_sum\_accepted\_sales['country']

*# Gráfico*

fig, ax = plt.subplots(figsize=(20,5))

*# Treemap*

squarify.plot(

   sizes=S,

   label= [f"{country}\n{pct:.2f}%" for country, pct in zip(L, percentage)],     *# tengo que unir el nombre del país con su porcentaje*

   ax=ax,

   ec= 'white',

   text\_kwargs={'color': 'white', 'fontsize': 9})

plt.axis('off')

plt.show()

A green and blue rectangles

AI-generated content may be incorrect.

En ambos casos, el treemap representa la contribución de cada país a las ventas totales, y permite identificar los países con mayor volumen de ventas y su proporción en comparación con el resto.

**Ejercicio 4** –

Una variable categòrica i una numèrica.

**1º** Distribución de las compras realizadas por cada país, representación con un gráfico de barras de Seaborn. Este gráfico nos permite analizar el valor total de compras (variable numérica) de cada país (variable categórica).

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(7, 5))

*# Agrupo por país y calculo la suma de las transacciones aceptadas*

country\_sum\_accepted\_transactions = dataset.groupby('country')['amount'].sum().reset\_index()

X = country\_sum\_accepted\_transactions['country']

Y = country\_sum\_accepted\_transactions['amount']

*# Compras por país*

sns.barplot(x= X, y= Y, color="slateblue")

plt.title("Distribution of accepted purchases amount")

plt.xlabel("Countries that buy")

plt.ylabel("Purchases amount ($)")

plt.ylim(0, 100000)

plt.grid(True, linestyle='--', alpha=0.25)

*# Etiquetas de valores*

for i, v in enumerate(Y):

    plt.text(i, v + 1000, f"$ {v:,.0f}", ha='center', fontsize=9, color='darkblue')

plt.show()

A graph of a distribution of purchases

AI-generated content may be incorrect.

**2º** Distribución de las ventas realizadas por país. De forma análoga al anterior, este gráfico nos permite analizar el valor total de ventas (variable numérica) de cada país (variable categórica).

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(14, 5))

*# Agrupo por país y calculo la media de las transacciones aceptadas*

country\_sum\_accepted\_sales = dataset.groupby('country')['amount'].sum().reset\_index()

X = country\_sum\_accepted\_sales['country']

Y = country\_sum\_accepted\_sales['amount']

*# Ventas por país*

sns.barplot(x= X, y= Y, color="darkseagreen")

plt.title("Distribution of accepted sales amounts by country")

plt.xlabel("Countries that sell products")

plt.ylabel("Sales amount ($)")

plt.xticks(rotation=90)

plt.ylim(0, 40000)

plt.grid(True, linestyle='--', alpha=0.25)

*# Etiquetas de valores*

for i, v in enumerate(Y):

    plt.text(i, v + 500, f"$ {v:,.0f}", ha='center', fontsize=9, color='darkgreen')

plt.show()

A graph with green bars

AI-generated content may be incorrect.

**3º** Boxplot de ventas por país. También nos permite analizar las ventas por países, pero nos da información sobre la media de ventas, el valor máximo y el mínimo, los cuartiles, y la presencia de valores atípicos (por ej. Francia).

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Obtener lista única de países ordenados alfabéticamente

sorted\_countries = sorted(dataset['country'])

*# Gráfico*

plt.figure(figsize=(14, 8))

sns.boxplot(x='country', y='amount', data=dataset, color='darkseagreen', order=sorted\_countries)

plt.xticks(rotation=45, fontsize=9)

plt.title("Distribution of accepted sales amounts by country")

plt.xlabel("Country")

plt.ylabel("Sales amount ($)")

plt.grid(True, linestyle='--', alpha=0.25)

plt.show()

A graph with green rectangular bars

AI-generated content may be incorrect.

**4º** Gráfico de líneas que representa el total de ventas (variable numérica) a lo largo del tiempo (variable categórica año\_mes)

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

*# Agrupar por mes y sumar las cantidades*

monthly\_total\_accepted\_transactions = round(dataset.groupby(["year\_month"], as\_index=False)["amount"].sum(), 2)

x = monthly\_total\_accepted\_transactions["year\_month"].astype(str)       # Necesito convertirlo a string para representarlo

y = monthly\_total\_accepted\_transactions["amount"]

*# Gráfico*

plt.figure(figsize=(12, 7))

plt.plot(x, y, color="darkseagreen",  marker='o' )

plt.title("Total Sales Per Month")

plt.xlabel("Year and Month")

plt.ylabel("Total Sales ($)")

plt.xticks(rotation=45)

plt.ylim(0, 16000)

plt.grid(True, linestyle='--', alpha=0.25)

*# Etiquetas de valores*

for i, v in enumerate(y):

    plt.text(x[i], v + 750, f"$ {v:,.0f}", ha='center', fontsize=9, color='darkgreen')

*# Mostrar*

plt.show()

A graph with green lines and numbers

AI-generated content may be incorrect.

**5º** Mapa coroplético que representa el total de ventas según su distribución geográfica.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

*# Agrupo por país, año y calculo la suma de las transacciones aceptadas*

country\_year\_sum\_accepted\_transactions = dataset.groupby(['country', 'year\_month'], as\_index=False)['amount'].sum()

country\_year\_sum\_accepted\_transactions = country\_year\_sum\_accepted\_transactions.groupby(["country","year\_month"])[["amount"]].sum().reset\_index()

map = px.choropleth(country\_year\_sum\_accepted\_transactions,

                    color="amount",

                    locationmode= "country names",

                    locations="country",

                    hover\_name="country",

                    height=600,

                    width= 900)

*# Hover text*

map.update\_traces(hovertemplate=

                "<b>%{location}</b><br>" +

                "Total sales: $ %{z}")

map.show()

A screenshot of a computer

AI-generated content may be incorrect.

**Ejercicio 5** –

Dues variables categòriques.

**1º** Gráfico de barras para representar el valor total de las transacciones aceptadas (verde) y de las transacciones rechazadas (rojo). Nos permite comparar la tasa de rechazo con las transacciones aceptadas. Por ejemplo, Estados Unidos, Francia y Australia cuentan con transacciones rechazadas con un valor mayo que las transacciones efectivas.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

*# Agrupar por país y calcular las transacciones aceptadas y rechazadas*

accepted\_transactions\_by\_country = dataset[dataset['declined'] == 0].groupby('country')['amount'].sum()

declined\_transactions\_by\_country = dataset[dataset['declined'] == 1].groupby('country')['amount'].sum()

*# Crear un DataFrame con las transacciones aceptadas y rechazadas por país*

transactions\_by\_country = pd.DataFrame({

    'amount\_accepted': accepted\_transactions\_by\_country,

    'amount\_declined': declined\_transactions\_by\_country})

*# Crear eje X*

x\_labels = transactions\_by\_country.index

x = np.arange(len(x\_labels)) *# Posiciones en el eje X*

*# Configurar tamaño de figura*

plt.figure(figsize=(15, 9))

*# Crear barras con desplazamiento*

bar\_width = 0.45 *# Espaciado entre barras*

plt.bar(x - bar\_width/2, transactions\_by\_country['amount\_accepted'], width=bar\_width, color='darkseagreen', label='Accepted')

plt.bar(x + bar\_width/2, transactions\_by\_country['amount\_declined'], width=bar\_width, color='indianred', label='Declined')

*# Ajustar etiquetas del eje X*

plt.xticks(ticks=x, labels=x\_labels, rotation=45, ha='center', fontweight='bold')

*# Agregar etiquetas y título*

plt.xlabel("Country")

plt.ylabel("Transactions ($)")

plt.title("Accepted and Declined Transactions by Country")

plt.ylim(0, 32500)

plt.grid(True, linestyle='--', alpha=0.25)

plt.legend()

*# Agregar etiquetas de valores en las barras. Tengo que hacerlo para cada serie*

for i, v in enumerate(transactions\_by\_country['amount\_accepted']):

    plt.text(x[i] - bar\_width/2, v + 500, f"$ {v:,.0f}", ha='center', fontsize=9, color='darkgreen', rotation=90)

for i, v in enumerate(transactions\_by\_country['amount\_declined']):

    plt.text(x[i] + bar\_width/2, v + 500, f"$ {v:,.0f}", ha='center', fontsize=9, color='darkred', rotation=90)

*# Mostrar gráfico*

plt.show()

A graph of numbers and a number of numbers

AI-generated content may be incorrect.

**2º** Treemap agrupado que representa las compras realizadas en función del país (agrupación externa) y la ciudad (agrupación interna) de los usuarios. El tamaño y color de los cuadros va en función del valor de las ventas realizadas. En este caso es fácil detectar que Canadá cuenta con más de la mitad de las compras totales.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import plotly.express as px

*# Agrupar por país y ciudad sumando el monto de compras*

grouped\_data = dataset.groupby(['country', 'city'], as\_index=False)['amount'].sum()

*# Crear el gráfico Treemap*

fig = px.treemap(

    grouped\_data,

    path=['country', 'city'],  *# Jerarquía de datos*

    values='amount',  *# Tamaño basado en el valor de compras*

    color='amount',  *# Colorear por valor de compras*

    hover\_data=['amount'],  *# Información adicional al pasar el cursor*

    color\_continuous\_scale='Viridis')

*# Hover text*

fig.update\_traces(hovertemplate=

                "<b>%{parent}</b><br>" +

                "<b>%{label}</b><br>" +

                "Total sales: $ %{value}")

*# Mostrar*

fig.show()

A screenshot of a computer

AI-generated content may be incorrect.

**3º** Igual que ocurre en el gráfico anterior, este gráfico de proyección solar representa el total de las compras en función del país y la ciudad del usuario. En este caso se resaltan mejor los niveles de jerarquía, estando los países en el centro del círculo con una jerarquía mayor, rodeado por las ciudades que les pertenecen. En este gráfico también se refleja el total de las ventas en el color y tamaño de cada sección.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import plotly.express as px

*# Agrupar por país y ciudad sumando el monto de compras*

grouped\_data = dataset.groupby(['country', 'city'], as\_index=False)['amount'].sum()

fig = px.sunburst(

    grouped\_data,

    path=['country', 'city'],

    values='amount',

    color='amount',

    hover\_data='amount',

    color\_continuous\_scale='Blues',

    width= 600,

    height= 600)

*# Hover text*

fig.update\_traces(hovertemplate=

                "<b>%{parent}</b><br>" +

                "<b>%{label}</b><br>" +

                "Total sales: $ %{value}")

fig.show()

A screenshot of a computer

AI-generated content may be incorrect.

**Ejercicio 6** –

Tres variables.

En este caso se seleccionó un mapa animado de Plotly para mostrar el total de ventas realizada por cada país a lo largo del tiempo. El color de la burbuja representa cada país, el tamaño representa el total de las ventas, y la animación permite ver la evolución a través del tiempo.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import plotly.express as px

*# Agrupo por país, año y calculo la suma de las transacciones aceptadas*

country\_year\_sum\_accepted\_transactions = dataset.groupby(['country', 'year\_month'], as\_index=False)['amount'].sum()

country\_year\_sum\_accepted\_transactions = dataset.groupby(["country","year\_month"])[["amount"]].sum().reset\_index()

map = px.scatter\_geo(country\_year\_sum\_accepted\_transactions,

                    locations= 'country',

                    size='amount',

                    locationmode =  "country names",

                    color= 'country',

                    size\_max=60,

                    animation\_frame='year\_month',

                    projection="equirectangular",

                    width= 900,

                    height= 600)

*# Hover text*

map.update\_traces(hovertemplate=

                "<b>%{location}</b><br>" +

                "Total sales: $ %{marker.size:,.2f}")

map.show()

A screenshot of a computer

AI-generated content may be incorrect.

**Ejercicio 7** –

Graficar un Pairplot.

El siguiente gráfico nos permite observar la relación entre múltiples variables, destacando patrones y posibles correlaciones en los datos. En este caso se representó el total de ventas, el precio del producto y la cantidad de productos vendidos por transacción. El color de los gráficos viene definido en función del país al que se producen las ventas. Las ventas y el precio de los productos siguen una distribución multimodal, mientras que la cantidad de productos son valores discretos. No se observan relaciones claras entre las diferentes variables, pero sí se podría afirmar que Canadá tiene una mayor densidad de ventas que los otros países.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

*# Seleccionar variables para representar*

pairplot\_data = dataset[['amount', 'price', 'year\_month', 'country', 'product\_quantity']]

*# Adapto los nombres:*

pairplot\_data = pairplot\_data.rename(

columns={

'amount': 'Total Sales ($)',

'price': 'Product Price ($)',

'year\_month': 'Year-Month',

'product\_quantity': 'Product Quantity'})

*# Pairplot*

sns.pairplot(pairplot\_data,

kind="scatter",

diag\_kind="kde",

corner=False,

plot\_kws={'alpha':0.6, 's': 15},

hue="country",

palette="Set1",

aspect= 1.5)

plt.show()

A collage of graphs

AI-generated content may be incorrect.

# Nivel 2

Els 2 exercicis del nivell 2 de la tasca 01.

**Ejercicio 1** –

Correlació de totes les variables numèriques.

Este mapa de calor de correlaciones nos permite analizar las relaciones entre las variables numéricas, destacando qué características están más relacionadas entre sí. El coeficiente de correlación puede variar entre +1 y -1, cuanto más cerca está de +1 significa que las variables están directamente relacionadas entre sí, cuanto más cerca está de -1 significa que las variables están inversamente relacionadas entre sí, y cuando más cerca está del 0 significa que las variables no se correlacionan entre sí.

En este caso los coeficientes obtenidos son muy bajos y no se puede afirmar que ninguna de estas variables dependa de las otras.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

numerical\_cols = dataset[["amount", "product\_quantity", "price", "weight"]]

numerical\_cols = numerical\_cols.rename(

    columns={

            'amount': 'Total Sales ($)',

            'price': 'Product Price ($)',

            'product\_quantity': 'Product Quantity',

            'weight': 'Product Weight (kg)'

        })

correlation = numerical\_cols.corr()

sns.heatmap(correlation, cmap="coolwarm", annot=True, fmt=".2f", linewidths=0.5, square=False)

plt.title("Correlation Heatmap - Numerical Features", fontweight="bold")

plt.show()

A screenshot of a diagram

AI-generated content may be incorrect.

**Ejercicio 2** –

Implementa un jointplot.

El siguiente gráfico de distribución de dos variables muestra la estimación de densidad de kernel (KDE) entre el precio de las ventas y el precio medio de los productos. No se observa una relación lineal entre ambas variables, pero sí se puede ver que las curvas de nivel resaltan áreas con mayor concentración de datos, indicando patrones de densidad en distintas regiones sin una tendencia clara.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

total\_transaction\_amount = dataset.groupby(by=['transaction\_id'])['amount'].sum()/dataset['transaction\_id'].value\_counts()

product\_price = dataset.groupby('product\_id')['price'].mean()

*# Tengo que relacionar cada id con su valor total:*

dataset['total\_transaction\_amount'] = dataset['transaction\_id'].map(total\_transaction\_amount)

dataset['product\_price'] = dataset['product\_id'].map(product\_price)

fig = sns.jointplot(data=dataset, x='total\_transaction\_amount', y='product\_price', kind='kde')

plt.xlabel("Sales amount ($)")

plt.ylabel("Average product price ($)")

plt.xlim(-100, 650)

plt.ylim(-25, 225)

plt.suptitle("Relationship Between Sales Amount and Product Price", y=1)

plt.show()

A graph of sales and sales

AI-generated content may be incorrect.

# Nivel 3

Els 2 exercicis del nivell 3 de la tasca 01.

**Ejercicio 1** –

Implementa un violinplot combinat amb un altre tipus de gràfic.

**1º** La siguiente representación es un gráfico de violín combinado con un gráfico de dispersión. El violín nos permite ver la densidad de las ventas por país, mientras que la dispersión nos permite ver en detalle la situación de los puntos e identificar los valores atípicos.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(20, 7))

sns.violinplot(x="country", y="amount", data=dataset, inner=None, alpha=0.7)

sns.stripplot(x="country", y="amount", data=dataset, size=4, color="black", alpha=0.6, jitter=True)

plt.xticks(rotation=45)

plt.title("Total sales by country")

plt.xlabel("Country")

plt.ylabel("Sales ($)")

plt.ylim(-200, 800)

plt.show()

A graph showing different sizes of lines

AI-generated content may be incorrect.

**2º** Otra forma de combinar un gráfico de violín podría ser con un boxplot, que también nos da información de los valores atípicos como un gráfico de dispersión, pero además suma información sobre el valor medio, el mínimo, el máximo y los rangos intercuartílicos.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(20, 7))

sns.violinplot(x='country', y='amount', data=dataset, inner=None, color="lightblue")

sns.boxplot(x='country', y='amount', data=dataset, width=0.2, boxprops=dict(alpha=0.5))

plt.xticks(rotation=45)

plt.title("Total sales by country")

plt.xlabel("Country")

plt.ylabel("Sales ($)")

plt.ylim(-200, 800)

plt.show()

A graph of different sizes of points

AI-generated content may be incorrect.

**Ejercicio 2** –

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

**1º** El siguiente FacetGrid representa de forma paralela la distribución de las transacciones aceptadas y de los precios de los productos. En cada histograma podemos ver la frecuencia de los valores divididos en rangos y la curva de densidad superpuesta suaviza la distribución y nos permite ver más claramente la tendencia.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(12, 5))

*# Histograma de amount*

plt.subplot(1, 2, 1)        *# Dos gráficos al mismo nivel (uno al lado del otro), el de la izquierda*

sns.histplot(dataset['amount'], bins=30, kde=True, color="blue")

plt.title("Distribution of accepted transaction amounts")

plt.xlabel("Amount ($)")

plt.ylabel("Count")

plt.grid(True, linestyle='--', alpha=0.25)

*# Histograma de price*

plt.subplot(1, 2, 2)       *# Dos gráficos al mismo nivel (uno al lado del otro), el de la derecha*

sns.histplot(dataset['price'], bins=30, kde=True, color="green")

plt.title("Product price distribution")

plt.xlabel("Priced ($)")

plt.ylabel("Count")

plt.grid(True, linestyle='--', alpha=0.25)

plt.tight\_layout()

plt.show()

A comparison of graphs and charts

AI-generated content may be incorrect.

**2º** El siguiente FacetGrid presenta los mismos datos en gráficos de boxplot, lo que permite ver rápidamente la presencia de valores atípicos, los valores mínimo y máximo, la mediana y los cuartiles.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(12, 5))

*# Boxplot de amount*

plt.subplot(1, 2, 1)

sns.boxplot(x=dataset['amount'], color="slateblue")

plt.title("Distribution of transaction amounts")

plt.xlabel("Amount ($)")

plt.grid(True, linestyle='--', alpha=0.25)

*# Boxplot de price*

plt.subplot(1, 2, 2)

sns.boxplot(x=dataset['price'], color="darkseagreen")

plt.title("Product price distribution")

plt.xlabel("Price ($)")

plt.grid(True, linestyle='--', alpha=0.25)

plt.tight\_layout()

plt.show()

A comparison of a graph

AI-generated content may be incorrect.