

Tarea S8.02. Power BI con Python

Esta tarea consiste en la elaboración de un informe de Power BI, aprovechando las capacidades analíticas de Python. Se utilizarán los scripts de Python creados previamente en la tarea 1 para generar visualizaciones personalizadas con las bibliotecas Seaborn y Matplotlib. Estas visualizaciones serán integradas en el informe de Power BI para ofrecer una comprensión más profunda de la capacidad del lenguaje de programación den la herramienta Power BI.

Nivel 1

Los 7 ejercicios del nivel 1 de la tarea 01

El primer paso que debemos hacer una vez esté configurado Power BI es asegurarnos que tenemos instaladas las bibliotecas que necesitaremos.

Nos conectamos a través de Python a la base de datos:


Obtener datos

Py

Todo

Otras

Todo

 Script de Python

Y utilizamos el siguiente script:

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

# Vamos a crear la conexión:
# Parámetros de conexión
username = 'root'
password = 'Erethiel00.'
hostname = 'localhost'
database = 'transactionsv2'

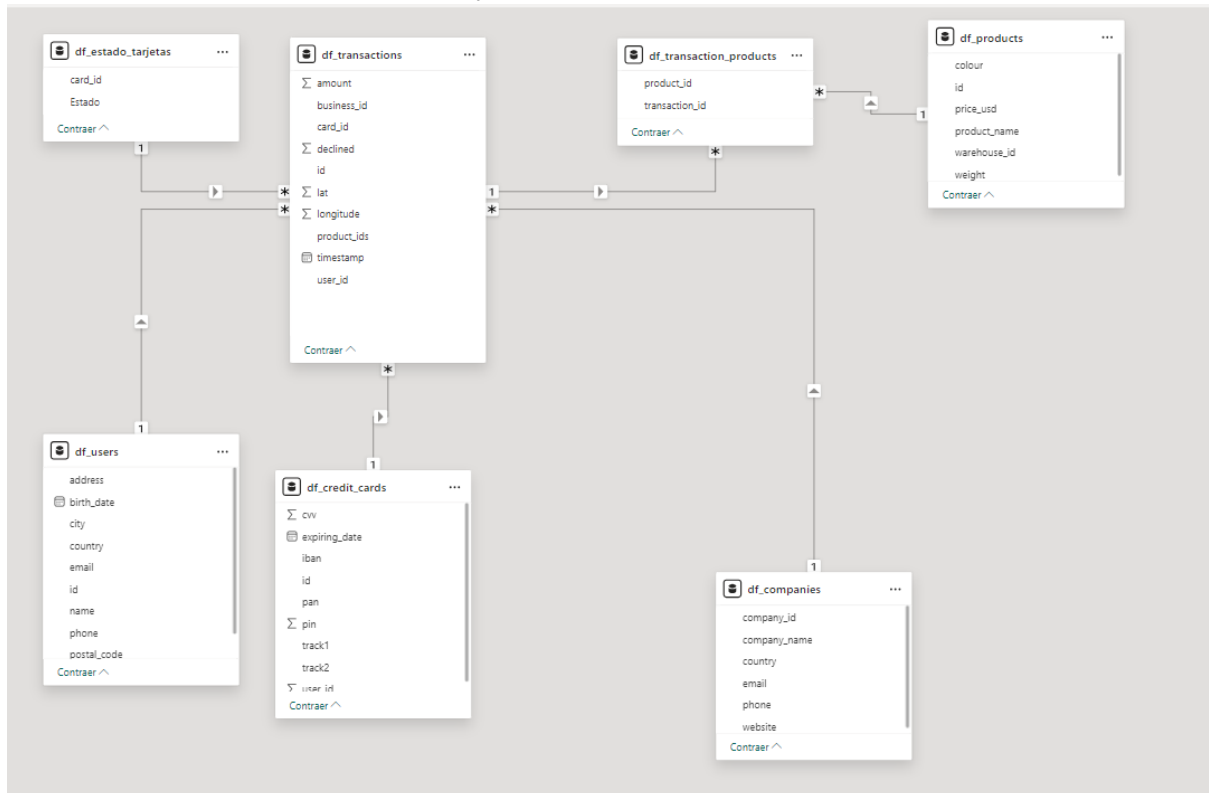
conexion = f'mysql+pymysql://{username}:{password}@{hostname}/{database}'
# Crear el motor de la base de datos
engine = create_engine(conexion)

# Para almacenar las tablas en data frames, primero necesitaremos el nombre de las tablas:
tabla_nombres= pd.read_sql('SHOW TABLES',engine)
tabla_nombres

# Utilizamos un bucle para lamacenar las tablas en data frames y crear variables globales:
for nombres in tabla_nombres['Tables_in_transactionsv2']:
    print(nombres)
    query= f"SELECT * FROM {nombres}"
    df= pd.read_sql(query,engine)
    globals()[f'df_{nombres}']=df

engine.dispose()
```

A continuación revisamos los datos y creamos las relaciones entre las tablas:



Habilitamos los objetos visuales de script de Python para poder hacer los gráficos. Para cada gráfico deberemos arrastrar los campos que utilizaremos y adaptar el código del script, ya que aquí trabajaremos con el “dataset” que Power BI crea al arrastrar estos campos.

Ejercicio 1

Una variable numérica: Cantidad de veces que se compró cada producto.

```
# El código siguiente, que crea un dataframe y quita las filas duplicadas, siempre se ejecuta y actúa como

# dataset = pandas.DataFrame(price_usd, id, weight)
# dataset = dataset.drop_duplicates()

# Importamos librerías
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# Hacemos el conteo y cambiamos el nombre a las columnas
product_counts = dataset['product_name'].value_counts().reset_index()
product_counts.columns = ['product_name', 'num_ventas'] #cambiamos el nombre a las columnas

# Creamos una variable para guardar los colores de las barras basados en el número de ventas
colors = ['#F9B1B3' if x < 50 else '#8DD3C7' for x in product_counts['num_ventas']]

plt.bar(product_counts['product_name'], product_counts['num_ventas'], color = colors)

plt.axhline(y=50, color='blue', linestyle='--', linewidth=2, label='Mínimo de 50 ventas')

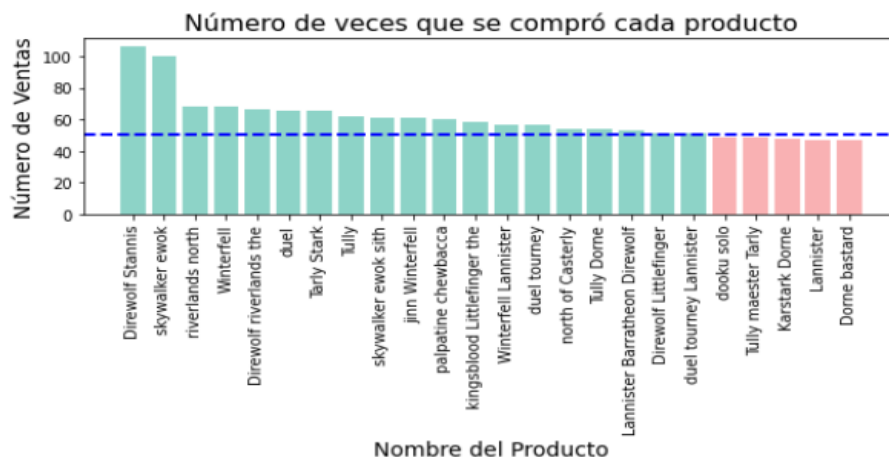
plt.title('Número de veces que se compró cada producto', fontsize=16)
plt.xlabel('Nombre del Producto', fontsize=14)
plt.ylabel('Número de Ventas', fontsize=14)

# Rotamos las etiquetas del eje x para que no se solapen
plt.xticks(rotation=90)
plt.tight_layout()

# Mostrar el gráfico
plt.show();
```

Gráfico:

N1 E1: Una variable numérica



Ejercicio 2

Dos variables numéricas: Precio y peso:

```
# El código siguiente, que crea un dataframe y quita las filas duplicadas, siempre se ejecuta y actúa como  
  
# dataset = pandas.DataFrame(price_usd, id, weight)  
# dataset = dataset.drop_duplicates()  
  
import matplotlib.pyplot as plt  
import pandas as pd  
import numpy as np  
  
# Ordenamos los datos por 'peso' y luego por 'precio'  
data = dataset.sort_values(by=['weight', 'price_usd'])  
  
# Creamos el gráfico de dispersión  
  
scatter = plt.scatter(data['weight'], data['price_usd'], s=100)  
  
plt.xlabel('Peso')  
plt.ylabel('Precio')  
plt.title('Relación entre Precio y Peso')  
plt.tight_layout()  
  
# Mostramos el gráfico  
plt.show();
```

Gráfico:

N1 E2: Dos variables numéricas



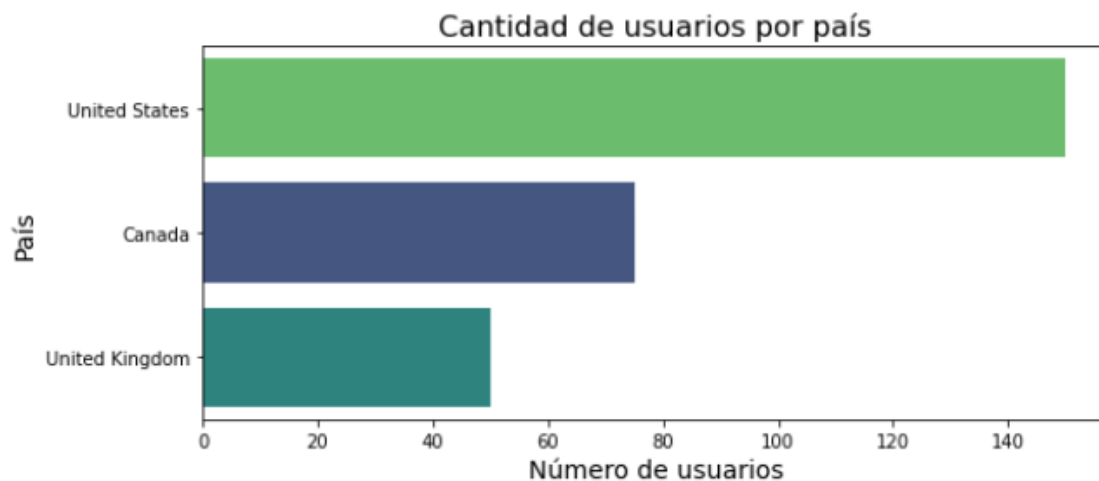
Ejercicio 3

Una variable categórica: Cantidad de usuarios por país

```
# El código siguiente, que crea un dataframe y quita las filas duplicadas, siempre se ejecuta y actúa como  
  
# dataset = pandas.DataFrame(amount, id)  
# dataset = dataset.drop_duplicates()  
  
# Importar bibliotecas necesarias  
import matplotlib.pyplot as plt  
import pandas as pd  
import seaborn as sns  
  
sns.countplot(data=dataset, order=dataset['country'].value_counts().index, y='country', hue='country', pale  
  
plt.title('Cantidad de usuarios por país', fontsize=16)  
plt.xlabel('Número de usuarios', fontsize=14)  
plt.ylabel('País', fontsize=14)  
plt.tight_layout()  
plt.show();
```

Gráfico:

N1 E3: Una variable categórica



Ejercicio 4

Una variable categórica y una numérica

```
# El código siguiente, que crea un dataframe y quita las filas duplicadas, siempre se ejecuta y actúa como

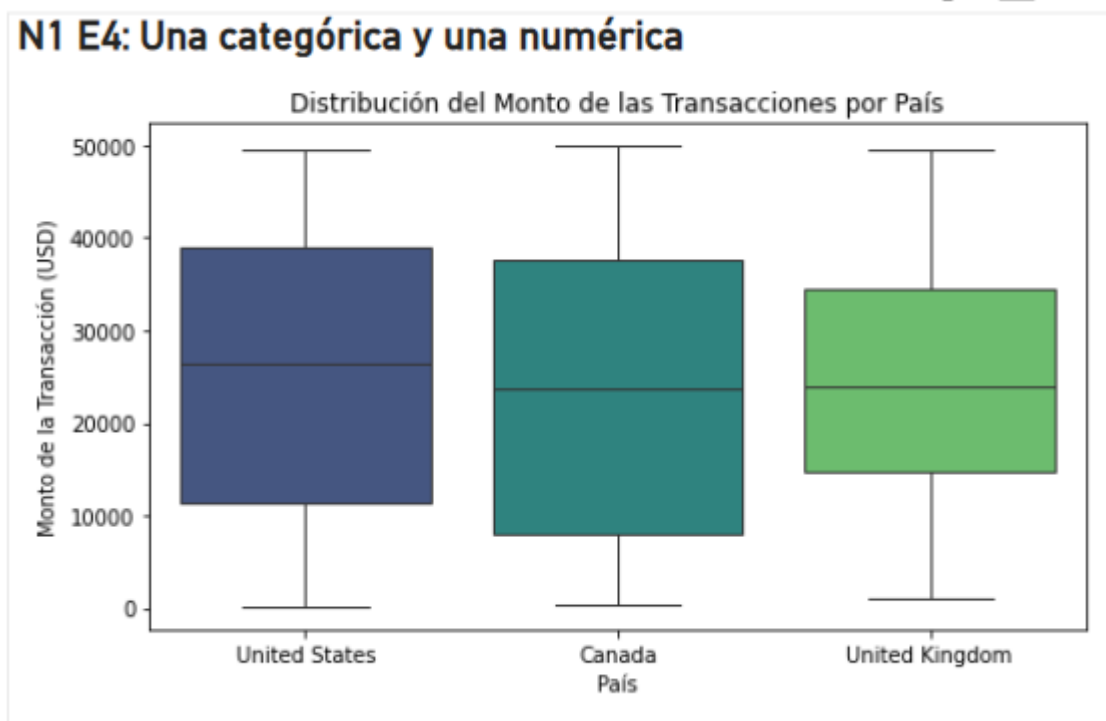
# dataset = pandas.DataFrame(amount, country, id, user_id)
# dataset = dataset.drop_duplicates()

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

sns.boxplot(x='country', y='amount', data=dataset, hue='country', palette='viridis')
plt.title('Distribución del Monto de las Transacciones por País')
plt.xlabel('País')
plt.ylabel('Monto de la Transacción (USD)')
plt.tight_layout()

plt.show();
```

Gráfico:



Ejercicio 5

Dos variables categóricas: Cantidad de transacciones por país y declined

```
# El código siguiente, que crea un dataframe y quita las filas duplicadas, siempre se ejecuta y actúa como .

# dataset = pandas.DataFrame(declined, id, business_id, company_id, country)
# dataset = dataset.drop_duplicates()

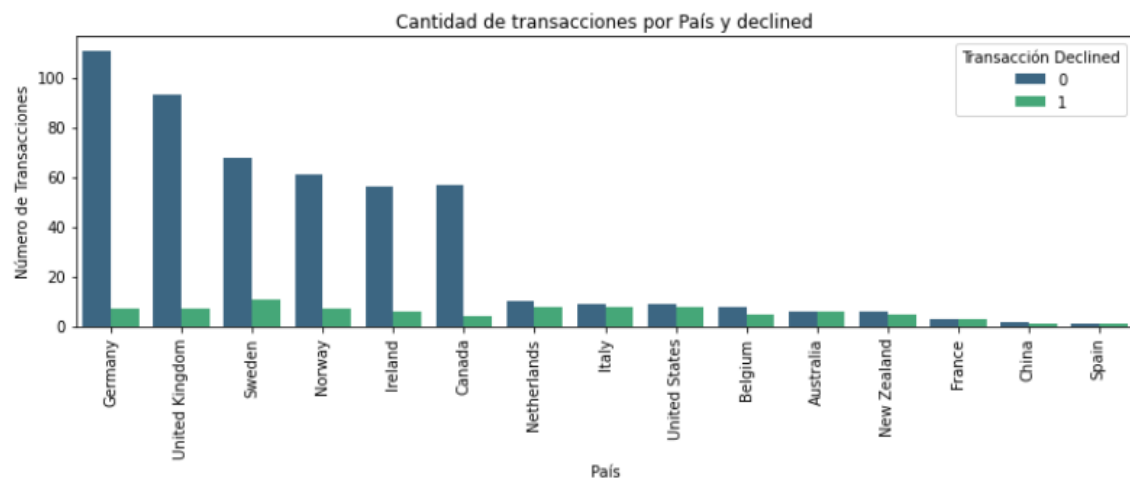
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

sns.countplot(data=dataset, x='country', hue='declined', order=dataset['country'].value_counts().index, palette='magma')

plt.title('Cantidad de transacciones por País y declined')
plt.xlabel('País')
plt.ylabel('Número de Transacciones')
plt.xticks(rotation=90)
plt.legend(title='Transacción Declined', loc='upper right')
plt.tight_layout()

plt.show();
```

Gráfico:



Ejercicio 6

Tres variables: Transacciones a través del tiempo por país:

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

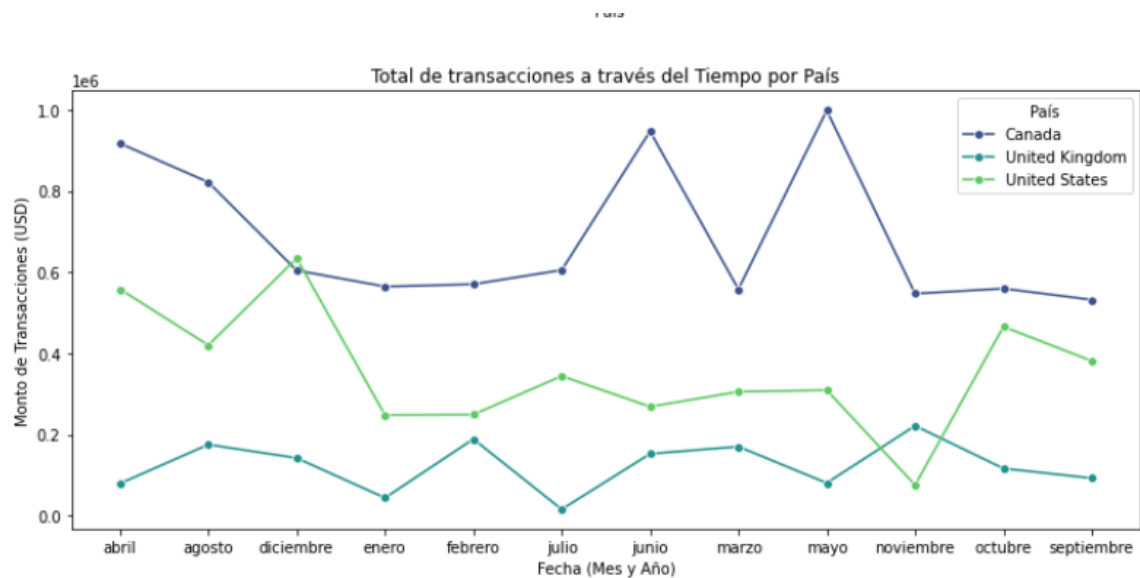
# Agrupar los datos por mes y país, calculando el monto total de transacciones por mes
grouped_data = dataset.groupby(['Mes', 'country'])['amount'].sum().reset_index()

sns.lineplot(data=grouped_data, x='Mes', y='amount', hue='country', marker='o', palette='viridis')

plt.title('Total de transacciones a través del Tiempo por País')
plt.xlabel('Fecha (Mes y Año)')
plt.ylabel('Monto de Transacciones (USD)')
plt.legend(title='País', loc='upper right')
plt.tight_layout()

plt.show()
```

Gráfico:



Ejercicio 7

Pairplot

```
# El código siguiente, que crea un dataframe y quita las filas duplicadas, siempre se ejecuta y actúa como  
  
# dataset = pandas.DataFrame(amount, price_usd, weight, id, product_id, transaction_id, id.1)  
# dataset = dataset.drop_duplicates()  
  
import matplotlib.pyplot as plt  
import pandas as pd  
import seaborn as sns  
  
# Crear un pairplot para 'amount' y 'country'  
sns.pairplot(data=dataset, vars=['price_usd', 'weight'], diag_kind='kde', kind='reg')  
  
plt.show();
```

Gráfico:

N1 E7: Pairplot

