Sprint 8_S01 – EXPLICACIÓN

Nivell 1

Aquesta pràctica permetrà explorar com es poden utilitzar eines de visualització avançades en Python per a interpretar i presentar dades de manera efectiva, proporcionant una oportunitat valuosa per a millorar les habilitats analítiques i tècniques.

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.

EXERCICI NUM 1, _ PART 01

In [2]:
```
import pandas as pd
import mysql.connector
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
# S8.01 -CONNEXIÓ AMB MYSQL - SPRINT 8 EJERCICIO 1

mydb = mysql.connector.connect(
  host="127.0.0.1",
  user="root",
  password="Plus7070",
  database="transactions"
)

# Usa la conexción
cur = mydb.cursor()

# Ejecuta la consulta

cur.execute("""
        SELECT *
        FROM transaction
        ORDER BY id ASC
""")

# Gurada la consulta en un objeto
myresult = cur.fetchall()
```

In [4]:
```
# Close cursor and connection
cur.close()
mydb.close()
```

In [5]:
```
df = pd.DataFrame(myresult, columns=[i[0] for i in cur.description])
```

In [6]:
```
df.head()
```

Out[6]:

	id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
0	02C6201E-D90A-1859-B4EE-88D2986D3B02	CcU-2938	b-2362	92	81.9185	-12.5276	2021-08-28 23:42:24	466.92	0

Para empezar voy a explicar el proceso de conectar Jupyter Notebook con MySQL Workbench y cómo cargar toda la información de nuestras tablas en Python. Esta práctica no solo nos permitirá obtener los datos de nuestras tablas, sino que también nos dará la oportunidad de explorar herramientas de visualización avanzadas en Python para interpretar y presentar datos de manera

efectiva. Este ejercicio es una excelente manera de mejorar nuestras habilidades analíticas y técnicas.

Paso 1: Conexión con MySQL Workbench

El primer paso que realicé fue establecer una conexión con la base de datos MySQL Workbench desde Python. Para ello, utilicé la librería mysql.connector, que es una herramienta poderosa para conectarse y ejecutar consultas SQL desde un entorno Python. Configuré los parámetros de conexión, incluyendo el host, usuario, contraseña y la base de datos específica con la que íbamos a trabajar. Esto fue crucial para asegurarme de que estaba accediendo a la base de datos correcta.

Paso 2: Creación de un Cursor

Una vez establecida la conexión, el siguiente paso fue crear un cursor. El cursor es esencial porque actúa como un controlador que nos permite interactuar directamente con la base de datos. Utilicé este cursor para ejecutar la consulta SQL que necesitaba.

Paso 3: Ejecución de la Consulta

Con el cursor listo, ejecuté una consulta SQL para seleccionar todos los registros de la tabla transaction, ordenándolos por el campo id en orden ascendente. Esto me permitió obtener todos los datos necesarios de manera organizada y estructurada.

Paso 4: Recuperación y Almacenamiento de Resultados

Después de ejecutar la consulta, recuperé los resultados utilizando el método fetchall() del cursor. Este método me permitió almacenar todos los registros obtenidos de la base de datos en una variable que posteriormente utilizaría para análisis y visualización.

Paso 5: Cierre del Cursor y la Conexión

Para mantener una buena práctica de programación y gestión de recursos, cerré el cursor y la conexión una vez que terminé de usarlos. Esto es importante para liberar los recursos que ya no necesitamos y evitar posibles bloqueos en la base de datos.

Paso 6: Conversión de Resultados a un DataFrame de Pandas

Finalmente, convertí los resultados obtenidos en un DataFrame de Pandas. Utilicé los nombres de las columnas del cursor para definir las columnas del DataFrame, lo que me permitió trabajar con los datos de una manera mucho más fácil y eficaz para el análisis y la visualización.

Visualización de los Datos

Para cada ejercicio, realizaré una visualización de los datos cargados. La elección del tipo de gráfico dependerá de la naturaleza de los datos que esté visualizando. Por ejemplo, si quiero mostrar la distribución de transacciones a lo largo del tiempo, podría optar por un gráfico de líneas. Si deseo comparar cantidades entre diferentes categorías, podría elegir un gráfico de barras.

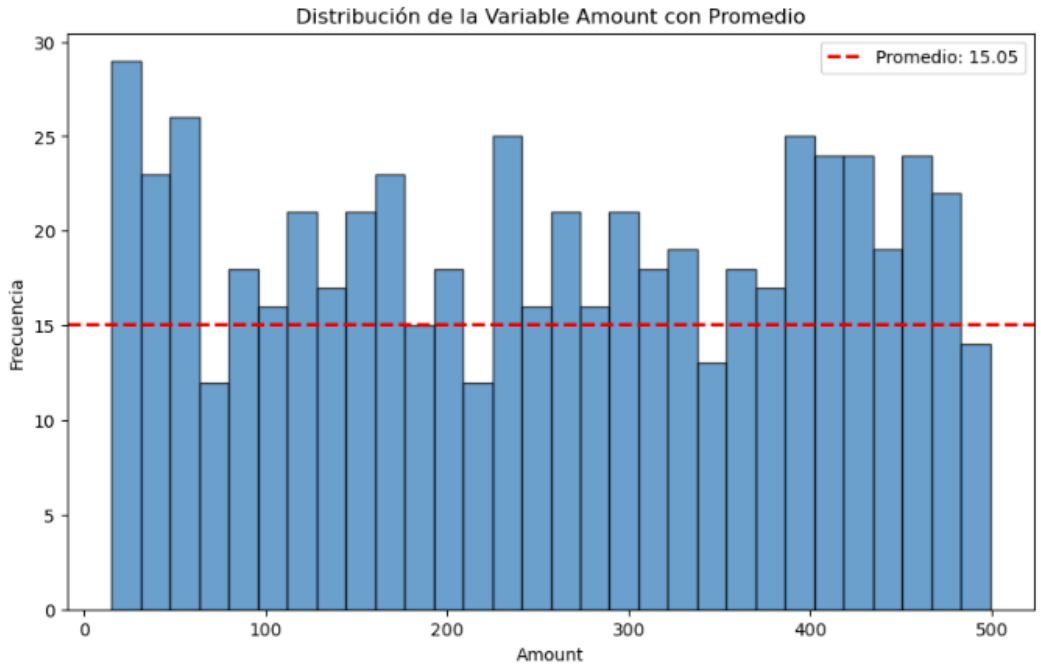
Cada visualización será cuidadosamente comentada, destacando lo que me llama la atención de graficar esa variable específica. Justificaré la elección del gráfico explicando por qué es la mejor opción para esos datos en particular y proporcionaré una interpretación de los resultados basándome en la información visualizada. Esto me permitirá no solo presentar los datos de manera efectiva, sino también obtener insights valiosos que puedan guiar futuras decisiones.

Conclusión

En resumen, esta práctica no solo me ha permitido conectar y extraer datos de una base de datos MySQL, sino que también me ha dado la oportunidad de mejorar mis habilidades en visualización y análisis de datos. Estoy entusiasmado por seguir explorando estas herramientas y ver qué insights puedo obtener de mis datos. ¡Espero que esta explicación haya sido clara y útil para todos! Si tienen alguna pregunta o necesitan más detalles, estaré encantado de ayudar.

EJERCICIO 2, SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN

In [7]:
```
# Sprint 8- Exercici 2 - VARIABLES NUMÉRICA
# COMPARAMOS LA CANTIDAD  AMOUNT CON UN VALOR CALCULADO (MIN,MAX,AVERAGE, ETC...)
# VALOR NUMÉRICO CALCULADO REPRESENTADO POR UNA LÍNA ROJA

# Conexión a MySQL y carga de datos en un DataFrame
mydb = mysql.connector.connect(
  host="127.0.0.1",
  user="root",
  password="Plus7070",
  database="transactions"
)

cur = mydb.cursor()

# Ejecutar la consulta para calcular el monto promedio
cur.execute("""
        SELECT MIN(amount) as average_amount
        FROM transaction
""")

# Obtener el resultado de la consulta
average_amount_result = cur.fetchone()
average_amount = average_amount_result[0]

# Cerrar el cursor
cur.close()

# Ejecutar la consulta para obtener todos los datos de la tabla transaction
cur = mydb.cursor()
cur.execute("""
        SELECT *
        FROM transaction
        ORDER BY id ASC
""")

myresult = cur.fetchall()
column_names = [i[0] for i in cur.description]
cur.close()
mydb.close()

df = pd.DataFrame(myresult, columns=column_names)

# Convertir la columna 'amount' a float
df['amount'] = df['amount'].astype(float)

# Crear el histograma de la variable numérica con una línea horizontal del promedio
plt.figure(figsize=(10, 6))
plt.hist(df['amount'], bins=30, edgecolor='black', alpha=0.7)
plt.axhline(y=average_amount, color='r', linestyle='--', linewidth=2, label=f'Promedio: {average_amount:.2f}')
plt.title('Distribución de la Variable Amount con Promedio')
plt.xlabel('Amount')
plt.ylabel('Frecuencia')
plt.legend()

plt.show()
```



Presentación del Ejercicio: Análisis de Variables Numéricas en MySQL Workbench y Jupyter Notebook

Introducción

En este ejercicio del Sprint 8, me centraré en el análisis de una variable numérica específica de la base de datos transactions. Utilizaré Python y las capacidades de visualización de Pandas y Matplotlib para comparar la cantidad amount con un valor calculado, representado gráficamente. El objetivo es proporcionar una comprensión profunda del proceso de análisis y visualización de datos.

Cálculo del Valor Mínimo de la Variable amount

Después de establecer la conexión con la base de datos (como se explicó en el ejercicio anterior), el primer paso fue calcular un valor de referencia para la variable amount. Para este ejercicio, decidí calcular el valor mínimo de amount en la tabla transaction.

Primero, creé un cursor, que es un objeto que permite ejecutar consultas SQL en la base de datos. Utilizando este cursor, ejecuté una consulta SQL que selecciona el valor mínimo de la columna amount y le asigna un alias para una referencia más fácil. Este proceso de ejecución de la consulta implica enviar la instrucción SQL al servidor de la base de datos y esperar a que el servidor procese y devuelva el resultado.

Una vez que la consulta se ejecutó correctamente, recuperé el resultado. Utilicé un método específico del cursor que devuelve el primer registro del conjunto de resultados de la consulta. En este caso, como la consulta solo devuelve un valor (el valor mínimo de amount), este método es adecuado. El resultado devuelto es una tupla, y extraje el valor numérico específico de esta tupla y lo almacené en una variable para su uso posterior en el análisis y la visualización.

Obtención de Todos los Datos de la Tabla transaction

A continuación, ejecuté una segunda consulta para recuperar todos los registros de la tabla transaction. Esta consulta selecciona todos los campos de la tabla y ordena los resultados por el campo id en orden ascendente. Este ordenamiento asegura que los datos estén organizados de manera coherente y sean fáciles de interpretar y analizar.

Utilicé nuevamente el cursor para ejecutar esta consulta. Después de ejecutar la consulta, recuperé todos los resultados utilizando un método del cursor que devuelve todos los registros del conjunto de resultados de la consulta. Este método devuelve una lista de tuplas, donde cada tupla representa un registro de la tabla transaction.

Almacenamiento de los Resultados y Preparación del DataFrame

Para facilitar el análisis y la manipulación de los datos, convertí los resultados recuperados en un DataFrame de Pandas. Un DataFrame es una estructura de datos bidimensional que facilita la manipulación y análisis de datos tabulares.

Para crear el DataFrame, utilicé los nombres de las columnas de la tabla transaction, que extraje del cursor.

Luego, me aseguré de que la columna amount estuviera en el formato adecuado para el análisis numérico. Convertí esta columna a tipo float, lo que es esencial para realizar operaciones matemáticas y estadísticas precisas.

Visualización de los Datos

Para la visualización, elegí crear un histograma de la variable amount. Un histograma es una representación gráfica de la distribución de un conjunto de datos numéricos, donde los datos se dividen en intervalos (bins) y la frecuencia de los datos en cada intervalo se representa mediante barras.

Además del histograma, añadí una línea horizontal que representa el valor mínimo calculado de amount. Esta línea actúa como una referencia visual que permite comparar la distribución de los valores de amount con el valor mínimo. Utilicé Matplotlib para crear esta visualización. Especifiqué el tamaño de la figura para asegurarme de que la visualización sea clara y legible. Configuré el histograma para que tenga 30 bins, lo que proporciona un equilibrio entre la granularidad y la legibilidad de la distribución de los datos.

La línea horizontal se añadió utilizando una función específica de Matplotlib que permite dibujar líneas horizontales en una gráfica. Especifiqué el color, estilo y ancho de la línea para que sea fácilmente distinguible en la gráfica. También añadí una etiqueta a la línea para que quede claro que representa el valor mínimo calculado.

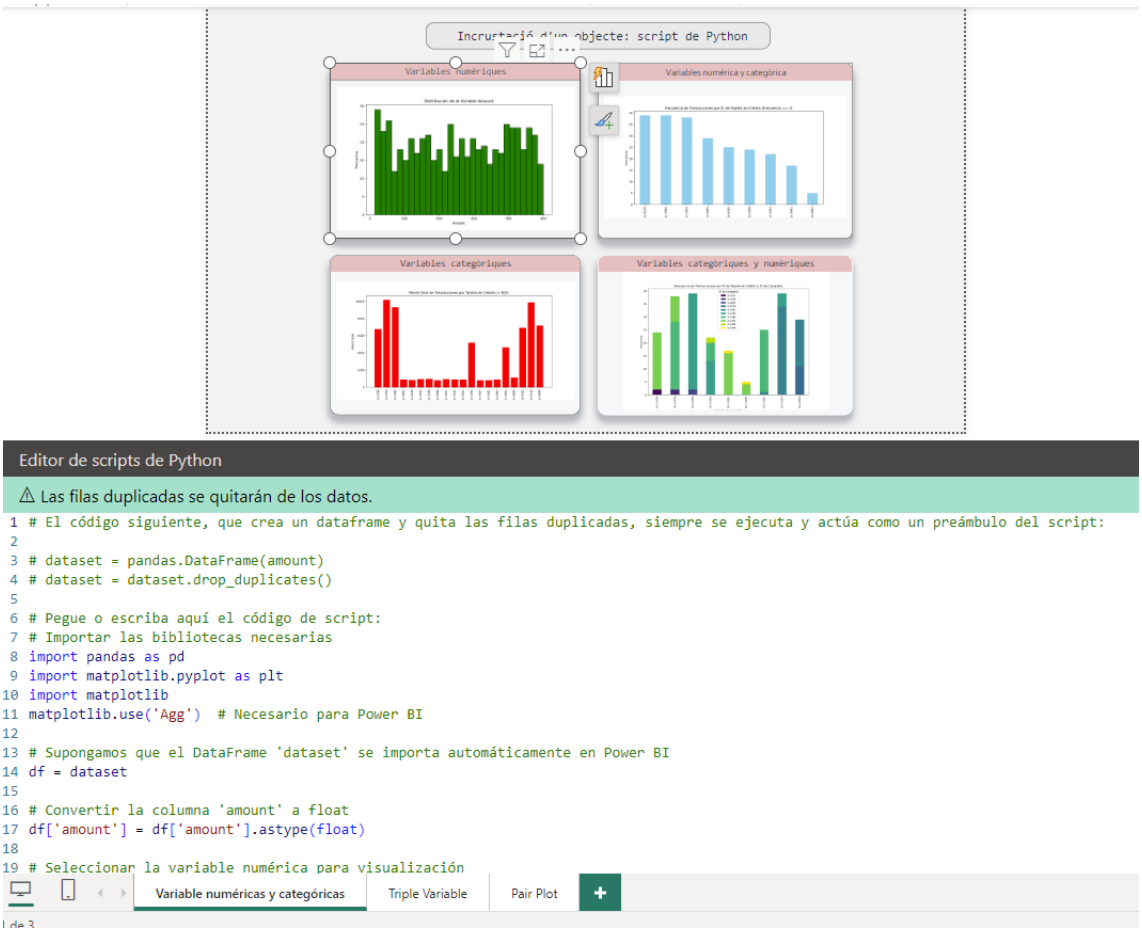
Finalmente, añadí títulos y etiquetas a la gráfica para proporcionar contexto y facilitar la interpretación. El título de la gráfica describe lo que se muestra, y las etiquetas de los ejes indican las unidades y la naturaleza de los datos representados. Añadí una leyenda para explicar qué representa la línea horizontal.

Conclusión

Este ejercicio me permitió realizar un análisis profundo de una variable numérica utilizando Python y MySQL. El proceso incluyó la ejecución de consultas SQL para obtener datos específicos, la manipulación de estos datos en un DataFrame de Pandas, y la creación de visualizaciones efectivas utilizando Matplotlib. La visualización mediante histogramas y líneas de referencia proporciona una herramienta poderosa para interpretar y presentar datos, facilitando la identificación de patrones y tendencias significativas.

PARA SER PRÁCTICOS, EXPLICO AQUÍ TAMBIEN LA SEGUNDA PARTE DE LOS EJERCICIOS DE ESTE SPRINT 8 QUE HACE REFRENCIA A POWER BI, PUES ESENCIALMENTE SE TRATA DEL MISMO CODIGO.

EJERCICIO 2, SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN



Explicación del Código para Incrustación en Power BI

Introducción

Este ejercicio consiste en incrustar un script de Python en Power BI para realizar un análisis visual de la variable numérica amount de un conjunto de datos. El script utiliza bibliotecas de Python para manipular los datos y generar una visualización, específicamente un histograma. A continuación, detallo lo que hace el código y cómo he solucionado los problemas específicos de esta implementación en Power BI.

Preámbulo del Script

El preámbulo del script en Power BI prepara el conjunto de datos creando un DataFrame y eliminando filas duplicadas. Esto asegura que los datos sean únicos y estén limpios antes de cualquier análisis o visualización. En Power BI, el objeto dataset representa los datos importados en el entorno, y este preámbulo actúa automáticamente para preparar estos datos.

Importación de Bibliotecas Necesarias

El script comienza importando las bibliotecas necesarias para el análisis y la visualización de datos. pandas se utiliza para la manipulación de datos, mientras que matplotlib se emplea para generar la visualización.

Configuración de Matplotlib para Power BI

A diferencia de Jupyter Notebook, Power BI requiere una configuración especial para que matplotlib funcione correctamente. Específicamente, se debe configurar matplotlib para usar el backend Agg. Este backend es necesario en Power BI para evitar problemas de renderizado gráfico, ya que Power BI no soporta los backends de matplotlib que requieren una interfaz gráfica.

Importación del DataFrame

En Power BI, se asume que el DataFrame dataset se importa automáticamente. Esto significa que los datos ya están disponibles en el entorno de Power BI sin necesidad de realizar una carga manual, permitiendo que el script se enfoque directamente en el análisis.

Conversión de la Columna amount a Tipo Float

Para asegurarse de que los valores numéricos se procesen correctamente, el script convierte la columna amount a tipo float. Esta conversión es crucial para realizar operaciones matemáticas y estadísticas precisas sobre los datos.

Selección de la Variable Numérica para Visualización

Se define amount como la variable numérica de interés para la visualización. Este enfoque modular permite reutilizar el script fácilmente para otras variables numéricas simplemente cambiando el nombre de la variable.

Creación del Histograma

El script crea un histograma para visualizar la distribución de la variable amount. El tamaño de la figura se configura para asegurar que la visualización sea clara y legible. El histograma se divide en 30 intervalos (bins), lo que proporciona un buen equilibrio entre granularidad y legibilidad de la distribución de datos. Las barras del histograma tienen bordes negros para mejorar la visibilidad y el color de las barras es verde.

Para proporcionar contexto y facilitar la interpretación, el histograma incluye un título y etiquetas para los ejes X e Y. El título describe el contenido del gráfico,

mientras que las etiquetas de los ejes indican la variable y la frecuencia de los valores en cada intervalo.

Visualización del Gráfico

Finalmente, el script muestra el gráfico generado. En el entorno de Power BI, este comando renderiza el histograma, haciendo posible que los usuarios vean la distribución de la variable amount.

Diferencias con Jupyter Notebook

Configuración de Matplotlib: En Power BI, se requiere configurar matplotlib para usar el backend Agg, lo que no es necesario en Jupyter Notebook.

Importación Automática del DataFrame: En Power BI, el DataFrame dataset se importa automáticamente, mientras que en Jupyter Notebook, el DataFrame debe crearse manualmente a partir de los resultados de la consulta SQL.

Preámbulo del Script: En Power BI, el preámbulo que elimina filas duplicadas del DataFrame asegura que los datos estén limpios antes de la visualización, algo que no se maneja automáticamente en Jupyter Notebook.

Conclusión

Este script demuestra cómo se puede utilizar Python en Power BI para realizar análisis y visualización de datos. La configuración especial de matplotlib para Power BI y la manipulación adecuada del DataFrame aseguran que los datos se procesen y visualicen correctamente. Esta integración de capacidades avanzadas de análisis y visualización de Python en Power BI mejora significativamente la capacidad de interpretar y presentar datos.

EJERCICIO 3 , SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN

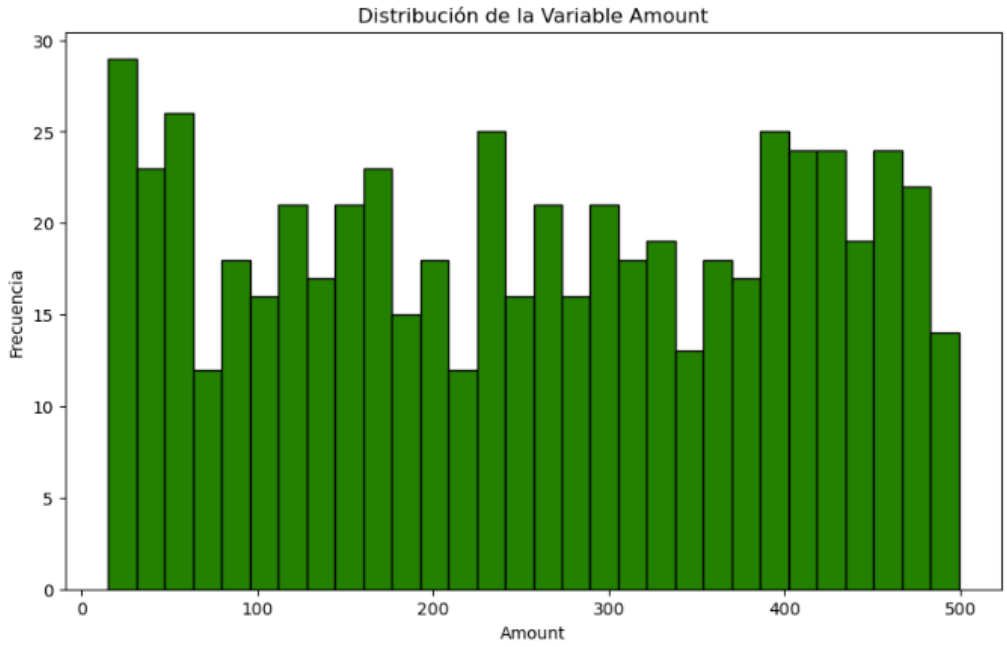
Dues variables numèriques.

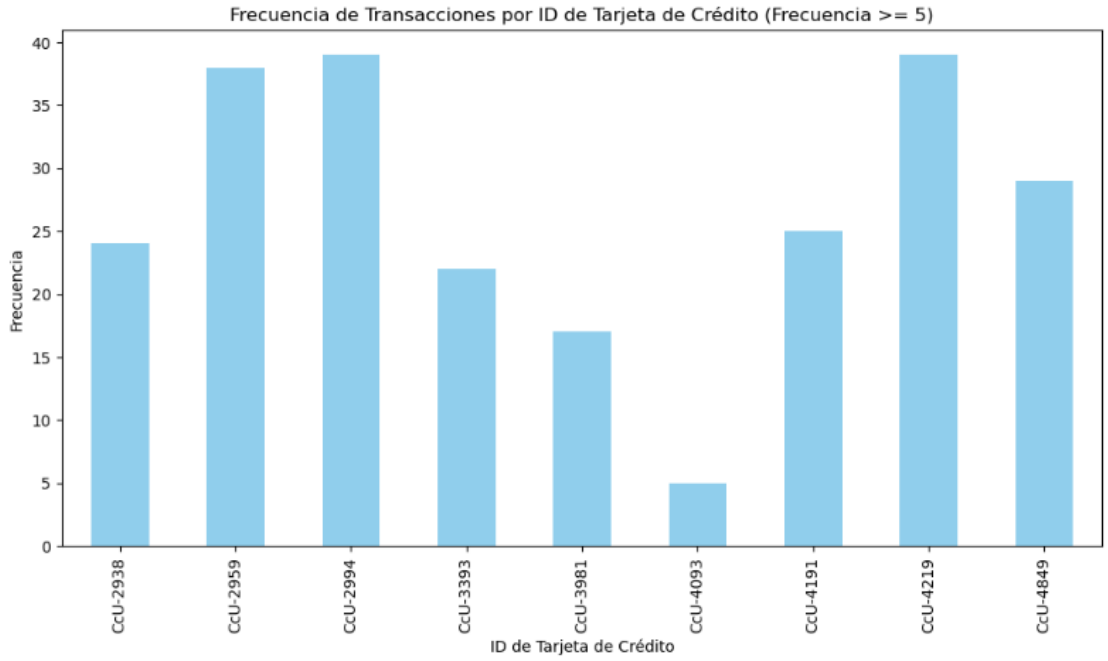
In [8]:
```
#SPRINT 8 - EJERCICIO 2 - DOS VARIABLES NUMÉRICAS

# Convertir la columna 'amount' a float
df['amount'] = df['amount'].astype(float)

# Seleccionar la variable numérica para visualización
variable_numerica = 'amount'

# Crear el histograma de la variable numérica
plt.figure(figsize=(10, 6))
plt.hist(df[variable_numerica], bins=30, edgecolor='black', color='g')
plt.title('Distribución de la Variable Amount')
plt.xlabel('Amount')
plt.ylabel('Frecuencia')

plt.show()
```

Atención Lucía: Con el objetivo de experimentar, he repetido el ejercico bajo otro prisma, aquí te comparto el segundo código

In [8]:
```
#SPRINT 8 - EJERCICIO 2 - DOS VARIABLES NUMÉRICAS

# Convertir la columna 'amount' a float
df['amount'] = df['amount'].astype(float)

# Seleccionar la variable numérica para visualización
variable_numerica = 'amount'

# Crear el histograma de la variable numérica
plt.figure(figsize=(10, 6))
plt.hist(df[variable_numerica], bins=30, edgecolor='black', color='g')
plt.title('Distribución de la Variable Amount')
plt.xlabel('Amount')
plt.ylabel('Frecuencia')

plt.show()
```

AQUÍ TE COMPARTO LOS DOS OUTPUTS QUE SE HAN GENERADO DE FORMA RESPECTIVA:





Proccedo finalmente a explicarte lo que hice:

Explicación del Código para Análisis de una Variable Categórica en Jupyter Notebook

Introducción

En este ejercicio del Sprint 8, se analiza una variable categórica de la base de datos transactions, específicamente credit_card_id. El objetivo es visualizar la frecuencia de transacciones por cada credit_card_id, eliminando aquellas tarjetas de crédito cuya frecuencia es insignificante. Para este análisis, se ha considerado arbitrariamente un valor de frecuencia mínima de 3. A continuación, detallo lo que hace el código y cómo he solucionado el problema en el entorno de Jupyter Notebook.

Conexión a la Base de Datos y Carga de Datos en un DataFrame

Primero, establecí una conexión con la base de datos MySQL llamada transactions. Utilicé la librería mysql.connector para conectarme a la base de datos, especificando el host, usuario, contraseña y nombre de la base de datos. Una vez establecida la conexión, creé un cursor que me permitió ejecutar consultas SQL en la base de datos.

Ejecución de la Consulta SQL

La consulta SQL ejecutada tenía como objetivo obtener los credit_card_id con una frecuencia de transacciones mayor o igual a 3. Esto se logró utilizando la instrucción SQL SELECT, agrupando las transacciones por credit_card_id y contando la frecuencia de cada grupo. La cláusula HAVING se utilizó para filtrar los resultados, incluyendo solo aquellos con una frecuencia de 3 o más.

Una vez ejecutada la consulta, el cursor recuperó los resultados, que consisten en una lista de tuplas donde cada tupla contiene un credit_card_id y su frecuencia correspondiente. Luego, extraje los nombres de las columnas del cursor para definir las columnas del DataFrame de Pandas.

Importación de los Resultados en un DataFrame

Los resultados de la consulta SQL se almacenaron en un DataFrame de Pandas. Utilicé los nombres de las columnas extraídos del cursor para establecer las columnas del DataFrame. Este paso es crucial para facilitar la manipulación y el análisis de los datos tabulares en Python.

Creación del Gráfico de Barras

Para visualizar la frecuencia de las transacciones por credit_card_id, se creó un gráfico de barras utilizando matplotlib. Configuré el tamaño de la figura para asegurar que la visualización fuera clara y legible. El DataFrame se estableció en un índice de credit_card_id, y se utilizó el método plot con el tipo de gráfico bar para generar las barras. El color de las barras se estableció en azul claro.

Añadí títulos y etiquetas a los ejes para proporcionar contexto y facilitar la interpretación del gráfico. El título describe el contenido del gráfico, y las etiquetas de los ejes indican la variable categórica (ID de Tarjeta de Crédito) y la frecuencia de las transacciones.

Finalmente, las etiquetas del eje X se rotaron 90 grados para mejorar la legibilidad, especialmente cuando hay muchos credit_card_id diferentes.

Detalles del Proceso

Conexión a MySQL y Ejecución de la Consulta:

Utilicé mysql.connector para establecer una conexión con la base de datos.

Creé un cursor para ejecutar una consulta SQL que selecciona credit_card_id y cuenta la frecuencia de transacciones por cada tarjeta.

Filtré los resultados para incluir solo aquellos credit_card_id con una frecuencia mayor o igual a 3.

Recuperé los resultados y extraje los nombres de las columnas para el DataFrame.

Creación del DataFrame:

Convertí los resultados de la consulta SQL en un DataFrame de Pandas, utilizando los nombres de las columnas extraídos del cursor.

Visualización de los Datos:

Configuré el gráfico de barras utilizando matplotlib.

Establecí el índice del DataFrame en credit_card_id y generé el gráfico de barras con las frecuencias.

Añadí títulos y etiquetas a la gráfica, y roté las etiquetas del eje X para mejorar la legibilidad.

Conclusión

Este script demuestra cómo realizar un análisis y visualización de datos categóricos utilizando Python y MySQL en un entorno de Jupyter Notebook. La configuración y manipulación adecuadas del DataFrame, junto con la creación de visualizaciones efectivas con matplotlib, permiten interpretar y presentar datos categóricos de manera clara y precisa. Este enfoque facilita la identificación de patrones y tendencias en los datos, proporcionando insights valiosos para el análisis estadístico.

EJERCICIO 3, SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN

Editor de scripts de Python

⚠ Las filas duplicadas se quitarán de los datos.

```
14 import pandas as pd
15 import matplotlib.pyplot as plt
16 import matplotlib
17 matplotlib.use('Agg')  # Necesario para Power BI
18
19 # Supongamos que el DataFrame 'dataset' se importa automáticamente en Power BI
20 df = dataset
21
22 # Filtrar datos con frecuencia de credit_card_id mayor o igual a 3
23 df_filtered = df['card_id'].value_counts()
24 df_filtered = df_filtered[df_filtered >= 3]
25
26 # Convertir el resultado a un DataFrame
27 df_filtered = df_filtered.reset_index()
28 df_filtered.columns = ['card_id', 'frequency']
29
30 # Crear un gráfico de barras para la variable categórica 'credit_card_id'
31 plt.figure(figsize=(12, 6))
```

Variable numéricas y categóricas | Triple Variable | Pair Plot | +

1 de 3

Explicación del Código para Análisis de una Variable Categórica en Power BI

Introducción

En este ejercicio, se utiliza un script de Python en Power BI para analizar y visualizar la frecuencia de transacciones por cada credit_card_id de la base de datos transactions. El objetivo es mostrar cómo se pueden eliminar filas duplicadas y crear una visualización de datos categóricos con una frecuencia significativa. A continuación, detallo lo que hace el código y cómo se ha solucionado el problema en el entorno de Power BI.

Preámbulo del Script

Antes de ejecutar el script de Python en Power BI, se ejecuta un preámbulo que prepara el conjunto de datos. Este preámbulo crea un DataFrame y elimina filas duplicadas para asegurar que los datos sean únicos y estén limpios antes de cualquier análisis o visualización. En Power BI, el objeto dataset representa los datos importados en el entorno y este preámbulo se asegura de que se trabaja con un conjunto de datos correcto y sin duplicados.

Importación de Bibliotecas Necesarias

El primer paso del script es importar las bibliotecas necesarias para el análisis y la visualización de datos. Utilizo pandas para manipular los datos y matplotlib para generar la visualización. Además, configuro matplotlib para utilizar el backend Agg, lo que es necesario para que matplotlib funcione correctamente en Power BI, evitando problemas de renderizado gráfico.

Importación del DataFrame

En el entorno de Power BI, se asume que el DataFrame dataset se importa automáticamente. Esto significa que los datos ya están disponibles y listos para su manipulación. Por lo tanto, simplemente asigno dataset a df para trabajar con estos datos en el script.

Filtrado de Datos

El siguiente paso es filtrar los datos para incluir solo aquellos credit_card_id con una frecuencia de transacciones mayor o igual a 3. Utilizo el método value_counts() de pandas para contar la frecuencia de cada credit_card_id. Luego, filtro este resultado para conservar solo los credit_card_id con una frecuencia de al menos 3.

Conversión a DataFrame

Una vez filtrados los datos, convierto el resultado a un DataFrame para facilitar la manipulación y visualización. Reseteo el índice del DataFrame y renombro las columnas a card_id y frequency para que sean más descriptivas y fáciles de interpretar.

Creación del Gráfico de Barras

Para visualizar la frecuencia de las transacciones por credit_card_id, creo un gráfico de barras utilizando matplotlib. Configuro el tamaño de la figura para asegurar que la visualización sea clara y legible. Establezco el índice del DataFrame en card_id y utilizo el método plot con el tipo de gráfico bar para generar las barras, configurando el color de las barras en azul claro.

Añadí títulos y etiquetas a los ejes para proporcionar contexto y facilitar la interpretación del gráfico. El título describe el contenido del gráfico, y las etiquetas de los ejes indican la variable categórica (ID de Tarjeta de Crédito) y la frecuencia de las transacciones.

Finalmente, roté las etiquetas del eje X 90 grados para mejorar la legibilidad, especialmente cuando hay muchos credit_card_id diferentes.

Visualización del Gráfico

El comando para mostrar el gráfico renderiza la visualización en el entorno de Power BI. Esto permite que los usuarios vean la distribución de la frecuencia de transacciones por credit_card_id, proporcionando una forma clara y efectiva de interpretar los datos categóricos.

Diferencias con Jupyter Notebook

Configuración de Matplotlib: En Power BI, es necesario configurar matplotlib para usar el backend Agg, lo que no es necesario en Jupyter Notebook.

Preámbulo del Script: En Power BI, el preámbulo que elimina filas duplicadas asegura que los datos estén limpios antes de la visualización, una práctica que no se maneja automáticamente en Jupyter Notebook.

Importación Automática del DataFrame: En Power BI, el DataFrame dataset se importa automáticamente, mientras que en Jupyter Notebook, se requiere cargar los datos manualmente desde una consulta SQL.

Conclusión

Este script demuestra cómo se puede utilizar Python en Power BI para realizar análisis y visualización de datos categóricos. La configuración especial de matplotlib y la manipulación adecuada del DataFrame aseguran que los datos se procesen y visualicen correctamente. Esta integración de capacidades avanzadas de análisis y visualización de Python en Power BI mejora significativamente la capacidad de interpretar y presentar datos categóricos, proporcionando insights valiosos para el análisis estadístico y la toma de decisiones.

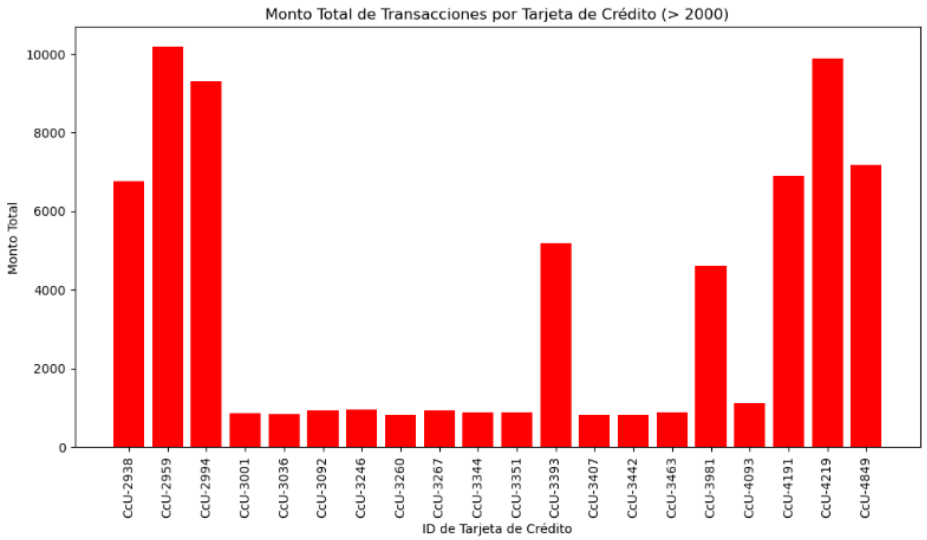
EJERCICIO 4 , SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN

- Exercici 4

Una variable categòrica i una numèrica.

In [10]:
```
# Sprint 8- Exercici 4 - UNA VARIABLE NUMÉRICA COMO EL AMOUNT Y UNA CATEGÓRICA COMO EL CREDIT CARD ID
# Conexión a MySQL y carga de datos en un DataFrame
# COMO PODEMOS VER  EN EL QUERY QUE MUESTRO A CONTINUACIÓN HE DESECHADO LAS OPERACIONES DE IMPORTE
# INFIMO PUES CARECEN DE VALOR ESTADÍSTICO Y DISTORSIONAN LOS RESULADOS
"""
SELECT credit_card_id, SUM(amount) as total_amount
FROM transactions.transaction
GROUP BY credit_card_id
HAVING SUM(amount) > 800;
"""
# Conexión a MySQL y carga de datos en un DataFrame
mydb = mysql.connector.connect(
  host="127.0.0.1",
  user="root",
  password="Plus7070",
  database="transactions"
)

cur = mydb.cursor()

# Ejecutar la consulta corregida
cur.execute("""
        SELECT credit_card_id, SUM(amount) as total_amount
        FROM transaction
        GROUP BY credit_card_id
        HAVING SUM(amount) > 800
""")

myresult = cur.fetchall()
column_names = [i[0] for i in cur.description]
cur.close()
mydb.close()

df = pd.DataFrame(myresult, columns=column_names)

# Convertir 'total_amount' a float
df['total_amount'] = df['total_amount'].astype(float)

# Crear el gráfico de barras del monto total por tarjeta de crédito
plt.figure(figsize=(12, 6))
plt.bar(df['credit_card_id'], df['total_amount'], color='red')
plt.title('Monto Total de Transacciones por Tarjeta de Crédito (> 2000)')
plt.xlabel('ID de Tarjeta de Crédito')
plt.ylabel('Monto Total')
plt.xticks(rotation=90)
plt.show()
```

Este es el output del código:



Explicación del Código para Análisis de Variables Numéricas y Categóricas en Jupyter Notebook

Introducción

En este ejercicio del Sprint 8, se realiza un análisis combinando una variable numérica (amount) y una variable categórica (credit_card_id) de la base de datos transactions. El objetivo es visualizar el monto total de transacciones por cada credit_card_id, descartando aquellas con importes ínfimos para evitar distorsionar los resultados. A continuación, detallo lo que hace el código y cómo he abordado el problema en el entorno de Jupyter Notebook.

Conexión a la Base de Datos y Ejecución de la Consulta SQL

Primero, establecí una conexión con la base de datos MySQL transactions utilizando la librería mysql.connector. Esto implica especificar el host, usuario, contraseña y nombre de la base de datos para asegurar una conexión correcta. Una vez conectados, creé un cursor para ejecutar consultas SQL.

Ejecuté una consulta SQL diseñada para seleccionar credit_card_id y sumar los valores de amount correspondientes a cada credit_card_id. La consulta agrupa los registros por credit_card_id y utiliza la cláusula HAVING para filtrar los resultados, manteniendo solo aquellos con un monto total mayor a 800. Esta filtración es crucial para descartar transacciones de importe ínfimo que podrían distorsionar los resultados estadísticos.

Recuperación y Almacenamiento de Resultados

Una vez ejecutada la consulta, el cursor recuperó los resultados, que consisten en una lista de tuplas. Cada tupla contiene un credit_card_id y su monto total correspondiente. Luego, extraje los nombres de las columnas del cursor para definir las columnas del DataFrame de Pandas.

Creación del DataFrame y Conversión de Datos

Los resultados de la consulta SQL se almacenaron en un DataFrame de Pandas para facilitar la manipulación y el análisis de los datos tabulares. Posteriormente, convertí la columna total_amount a tipo float para asegurar que los valores numéricos se procesen correctamente durante la visualización.

Creación del Gráfico de Barras

Para visualizar el monto total de transacciones por credit_card_id, creé un gráfico de barras utilizando matplotlib. Configuré el tamaño de la figura para asegurar que la visualización sea clara y legible. Utilicé el método bar para generar las barras, configurando el color de las barras en rojo.

Añadí títulos y etiquetas a los ejes para proporcionar contexto y facilitar la interpretación del gráfico. El título describe el contenido del gráfico, y las etiquetas de los ejes indican la variable categórica (ID de Tarjeta de Crédito) y

el monto total de las transacciones. Finalmente, roté las etiquetas del eje X 90 grados para mejorar la legibilidad, especialmente cuando hay muchos credit_card_id diferentes.

Detalles del Proceso

Conexión a MySQL y Ejecución de la Consulta:

Utilicé mysql.connector para establecer una conexión con la base de datos.

Creé un cursor para ejecutar una consulta SQL que selecciona credit_card_id y suma los valores de amount para cada tarjeta.

Filtré los resultados con una cláusula HAVING para incluir solo aquellos con un monto total mayor a 800.

Recuperé los resultados y extraje los nombres de las columnas para el DataFrame.

Creación del DataFrame:

Convertí los resultados de la consulta SQL en un DataFrame de Pandas, utilizando los nombres de las columnas extraídos del cursor.

Convertí la columna total_amount a tipo float para asegurar una manipulación correcta de los datos numéricos.

Visualización de los Datos:

Configuré el gráfico de barras utilizando matplotlib.

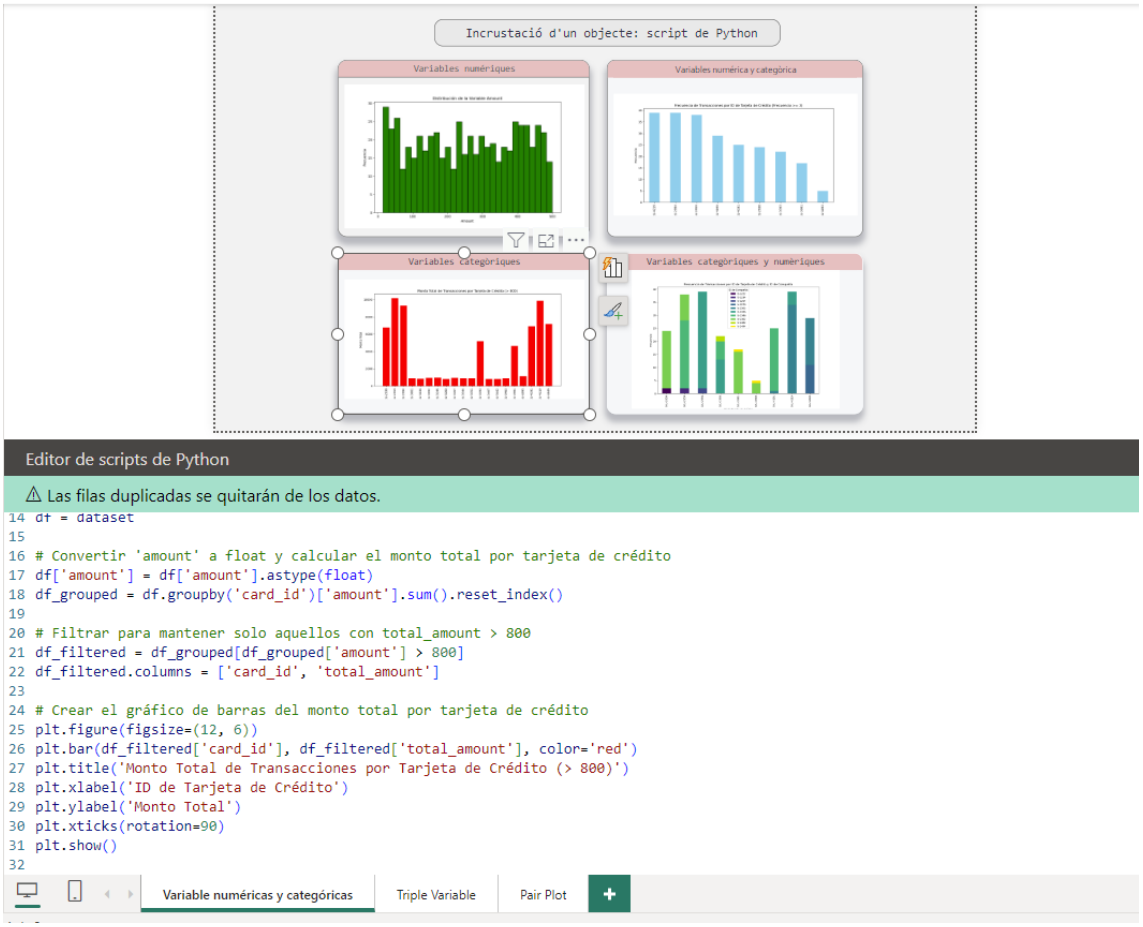
Establecí las barras con los valores de credit_card_id y total_amount, configurando el color en rojo.

Añadí títulos y etiquetas a la gráfica, y roté las etiquetas del eje X para mejorar la legibilidad.

Conclusión

Este script demuestra cómo realizar un análisis y visualización combinada de datos numéricos y categóricos utilizando Python y MySQL en un entorno de Jupyter Notebook. La configuración y manipulación adecuadas del DataFrame, junto con la creación de visualizaciones efectivas con matplotlib, permiten interpretar y presentar datos de manera clara y precisa. Este enfoque facilita la identificación de patrones y tendencias en los datos, proporcionando insights valiosos para el análisis estadístico y la toma de decisiones.

EJERCICIO 4, SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN



Explicación del Código Incrustado como un Objeto en Power BI

Introducción

En este ejercicio, se utiliza un script de Python incrustado en Power BI para analizar y visualizar el monto total de transacciones por cada credit_card_id de la base de datos transactions. La intención es destacar cómo se ha integrado el código en Power BI y cómo funciona dentro de este entorno, en lugar de detallar el código mismo.

Preámbulo del Script en Power BI

En Power BI, antes de ejecutar cualquier script de Python, se ejecuta automáticamente un preámbulo que prepara el conjunto de datos. Este preámbulo crea un DataFrame a partir del conjunto de datos importado y elimina filas duplicadas. Esto asegura que los datos sean únicos y estén limpios antes de cualquier análisis o visualización.

Importación Automática del DataFrame

El entorno de Power BI asume que el DataFrame dataset se importa automáticamente. Esto significa que los datos necesarios ya están disponibles

en el entorno de Power BI sin necesidad de cargar manualmente los datos desde una fuente externa, simplificando así el proceso inicial.

Configuración de Matplotlib

Para asegurar la compatibilidad de matplotlib en Power BI, se configura el backend Agg. Esta configuración es necesaria en Power BI para evitar problemas de renderizado gráfico, dado que Power BI no soporta los backends de matplotlib que requieren una interfaz gráfica.

Conversión y Agrupación de Datos

El script convierte la columna amount a tipo float para asegurar que los valores numéricos se procesen correctamente. Luego, agrupa los datos por credit_card_id y suma los valores de amount para cada tarjeta de crédito. Este agrupamiento permite calcular el monto total de transacciones por cada credit_card_id.

Filtrado de Datos

Después de agrupar los datos, el script filtra los resultados para mantener solo aquellos credit_card_id con un monto total de transacciones mayor a 800. Este paso es crucial para descartar transacciones de importe ínfimo que podrían distorsionar los resultados estadísticos.

Creación del Gráfico de Barras

Para visualizar el monto total de transacciones por credit_card_id, se crea un gráfico de barras utilizando matplotlib. El gráfico se configura para tener un tamaño adecuado, y las barras se colorean en rojo. Se añaden títulos y etiquetas a los ejes para proporcionar contexto y facilitar la interpretación del gráfico. Las etiquetas del eje X se rotan 90 grados para mejorar la legibilidad, especialmente cuando hay muchos credit_card_id diferentes.

Visualización en Power BI

Finalmente, el gráfico se renderiza dentro de Power BI. La integración de este gráfico en Power BI permite a los usuarios ver la visualización directamente en el dashboard, facilitando el análisis y la toma de decisiones. Este enfoque permite combinar las capacidades avanzadas de visualización de Python con las funcionalidades interactivas de Power BI.

Diferencias con Jupyter Notebook

Importación Automática del DataFrame: En Power BI, el DataFrame dataset se importa automáticamente, mientras que en Jupyter Notebook, se requiere cargar manualmente los datos desde una consulta SQL.

Configuración de Matplotlib: En Power BI, es necesario configurar matplotlib para usar el backend Agg, lo que no es necesario en Jupyter Notebook.

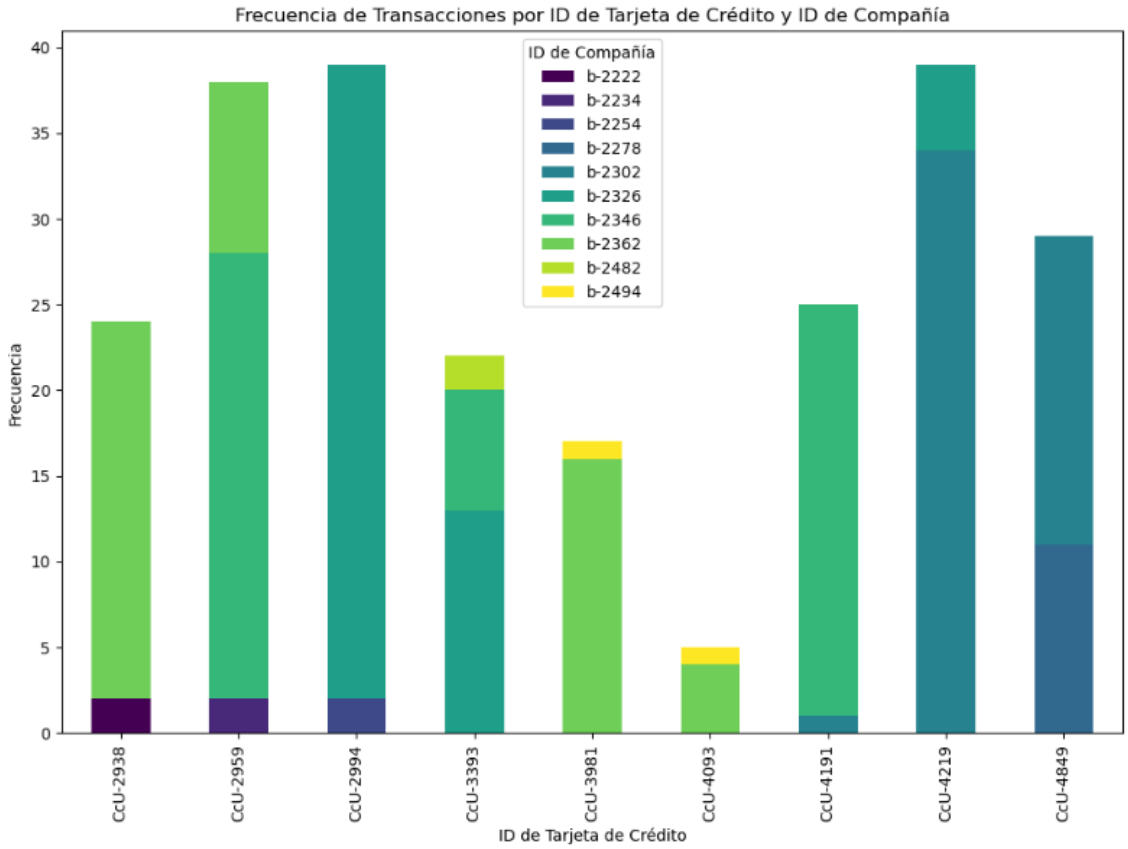
Preámbulo del Script: En Power BI, el preámbulo que elimina filas duplicadas asegura que los datos estén limpios antes de la visualización, una práctica que no se maneja automáticamente en Jupyter Notebook.

Conclusión

Este script demuestra cómo se puede utilizar Python dentro de Power BI para realizar análisis y visualización de datos. La configuración especial de matplotlib y la manipulación adecuada del DataFrame aseguran que los datos se procesen y visualicen correctamente. La integración de capacidades avanzadas de análisis y visualización de Python en Power BI mejora significativamente la capacidad de interpretar y presentar datos categóricos y numéricos, proporcionando insights valiosos para el análisis estadístico y la toma de decisiones.

EJERCICIO 5 , SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN

In [11]:
```
# Sprint 8- Exercici 5 -  DOS VARIABLES CATEGÓRICAS COMO EL COMPANY ID Y EL CREDIT CARD ID

# Conexión a MySQL y carga de datos en un DataFrame
mydb = mysql.connector.connect(
  host="127.0.0.1",
  user="root",
  password="Plus7070",
  database="transactions"
)

cur = mydb.cursor()

# Ejecutar la consulta para obtener datos de transaction
cur.execute("""
        SELECT credit_card_id, company_id
        FROM transaction
""")

myresult = cur.fetchall()
column_names = [i[0] for i in cur.description]
cur.close()
mydb.close()

df = pd.DataFrame(myresult, columns=column_names)

# Filtrar datos con frecuencia de credit_card_id mayor o igual a 5
df_filtered = df[df['credit_card_id'].isin(
    df['credit_card_id'].value_counts()[df['credit_card_id'].value_counts() >= 5].index
)]

# Crear una tabla cruzada (crosstab) de las dos variables categóricas
crosstab = pd.crosstab(df_filtered['credit_card_id'], df_filtered['company_id'])

# Crear un gráfico de barras apiladas
crosstab.plot(kind='bar', stacked=True, figsize=(12, 8), colormap='viridis')
plt.title('Frecuencia de Transacciones por ID de Tarjeta de Crédito y ID de Compañía')
plt.xlabel('ID de Tarjeta de Crédito')
plt.ylabel('Frecuencia')
plt.legend(title='ID de Compañía')
plt.xticks(rotation=90)

plt.show()
```



Explicación del Código en Jupyter Notebook para Análisis de dos Variables Categóricas

Introducción

En este ejercicio del Sprint 8, se analiza la relación entre dos variables categóricas de la base de datos transactions: credit_card_id y company_id. El objetivo es visualizar la frecuencia de transacciones agrupadas por ambas variables. A continuación, detallo lo que hace el código y cómo he abordado este análisis en el entorno de Jupyter Notebook.

Conexión a la Base de Datos y Ejecución de la Consulta SQL

Primero, establecí una conexión con la base de datos MySQL transactions utilizando la librería mysql.connector. Especificando los parámetros necesarios como host, usuario, contraseña y nombre de la base de datos, logré conectarme correctamente. Una vez conectados, creé un cursor para ejecutar consultas SQL.

Ejecuté una consulta SQL para seleccionar las columnas credit_card_id y company_id de la tabla transaction. Esta consulta permite recuperar todos los registros que contienen ambas variables categóricas.

Recuperación y Almacenamiento de Resultados

Después de ejecutar la consulta, el cursor recuperó los resultados, que consisten en una lista de tuplas. Cada tupla contiene un credit_card_id y su correspondiente company_id. Extraje los nombres de las columnas del cursor y utilicé estos nombres para crear un DataFrame de Pandas con los datos obtenidos.

Filtrado de Datos

Para asegurarme de que los datos sean significativos, filtré los credit_card_id que tienen una frecuencia de transacciones mayor o igual a 5. Utilicé el método value_counts() de pandas para contar la frecuencia de cada credit_card_id y luego filtré el DataFrame para incluir solo aquellos con una frecuencia adecuada.

Creación de una Tabla Cruzada

Una vez filtrados los datos, creé una tabla cruzada (crosstab) utilizando pandas. La tabla cruzada me permite observar la relación entre credit_card_id y company_id, mostrando la frecuencia de transacciones para cada combinación de estas dos variables categóricas. Esta tabla cruzada facilita el análisis de cómo las transacciones se distribuyen entre diferentes compañías y tarjetas de crédito.

Visualización de los Datos

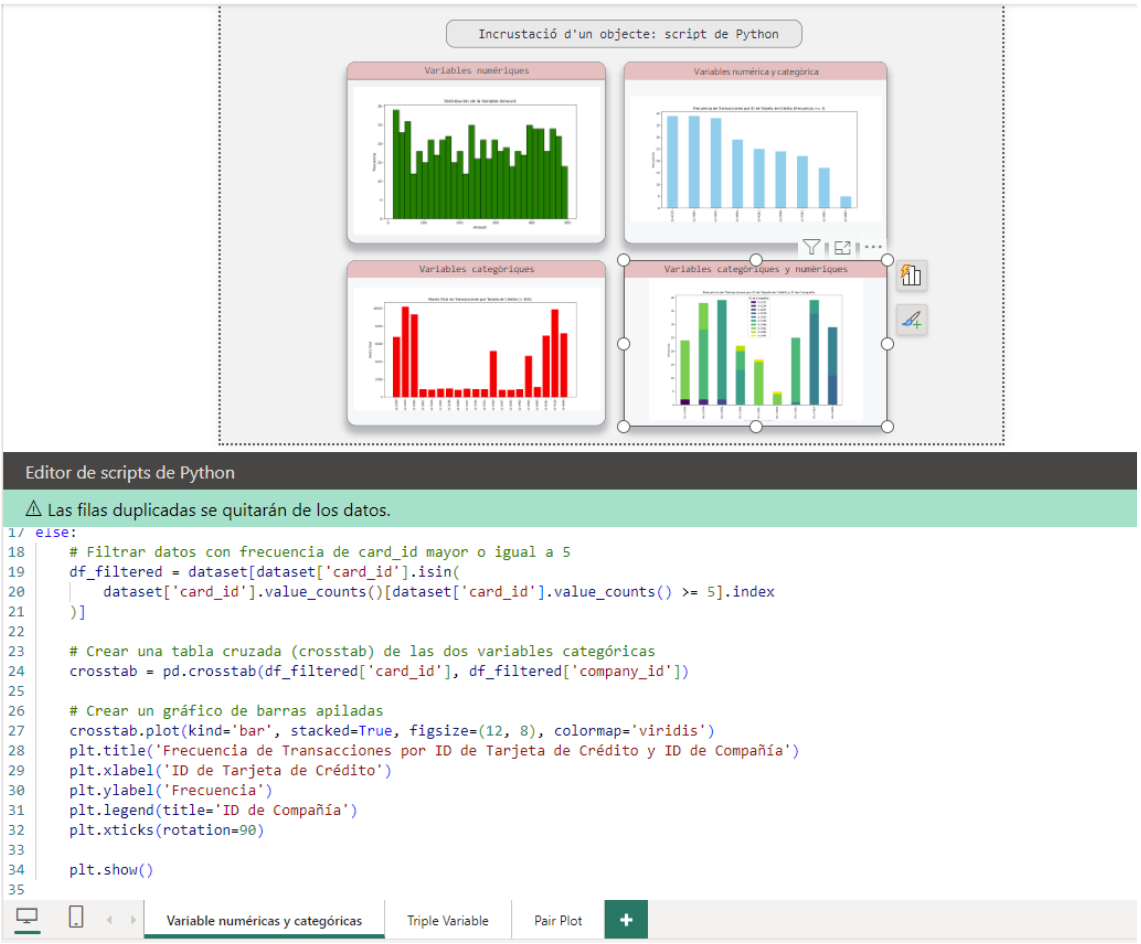
Para visualizar la información contenida en la tabla cruzada, creé un gráfico de barras apiladas utilizando matplotlib. Configuré el gráfico para tener un tamaño adecuado y utilicé un colormap (viridis) para diferenciar visualmente las distintas company_id. Las barras apiladas permiten ver cómo se distribuyen las transacciones de cada credit_card_id entre las diferentes compañías.

Añadí títulos y etiquetas a los ejes para proporcionar contexto y facilitar la interpretación del gráfico. El título describe el contenido del gráfico, y las etiquetas de los ejes indican las variables categóricas (ID de Tarjeta de Crédito y ID de Compañía) y la frecuencia de las transacciones. También roté las etiquetas del eje X 90 grados para mejorar la legibilidad, especialmente cuando hay muchos credit_card_id diferentes. La leyenda aclara qué company_id corresponde a cada segmento del gráfico.

Conclusión

Este script demuestra cómo realizar un análisis y visualización de la relación entre dos variables categóricas utilizando Python y MySQL en un entorno de Jupyter Notebook. La correcta configuración y manipulación del DataFrame, junto con la creación de visualizaciones efectivas con matplotlib, permite interpretar y presentar datos categóricos de manera clara y precisa. Este enfoque facilita la identificación de patrones y tendencias en los datos, proporcionando insights valiosos para el análisis estadístico y la toma de decisiones.

EJERCICIO 5, SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN



Explicación del Código para Análisis de dos Variables Categóricas en Power BI

Introducción

Este ejercicio utiliza un script de Python incrustado en Power BI para analizar y visualizar la relación entre dos variables categóricas: card_id y company_id. El objetivo es mostrar cómo se integran las capacidades de Python en Power BI para crear visualizaciones dinámicas y detalladas. A continuación, detallo lo que hace el código y cómo se ha logrado la integración efectiva en el entorno de Power BI.

Preámbulo del Script en Power BI

Antes de ejecutar el script de Python, Power BI ejecuta un preámbulo que prepara el conjunto de datos. Este preámbulo crea un DataFrame a partir de los datos importados y elimina las filas duplicadas para asegurar que los datos sean únicos y estén limpios antes de cualquier análisis o visualización.

Importación Automática del DataFrame

En Power BI, se asume que el DataFrame dataset se importa automáticamente. Esto significa que los datos necesarios ya están disponibles y listos para su manipulación en el script. Este enfoque simplifica el proceso de preparación de datos, permitiendo que el análisis se enfoque directamente en los datos disponibles.

Configuración de Matplotlib

Para asegurar la compatibilidad de matplotlib en Power BI, se configura el backend Agg. Esta configuración es crucial en Power BI para evitar problemas de renderizado gráfico, ya que Power BI no soporta los backends de matplotlib que requieren una interfaz gráfica.

Verificación del DataFrame

El script incluye una verificación para asegurarse de que el DataFrame no esté vacío. Si el DataFrame está vacío, se imprime un mensaje de error que indica que los datos deben seleccionarse correctamente en Power BI. Esta verificación es importante para evitar errores durante la ejecución del script y garantizar que siempre se trabajen con datos válidos.

Filtrado de Datos

El siguiente paso es filtrar los datos para incluir solo aquellos card_id con una frecuencia de transacciones mayor o igual a 5. Utilizo el método value_counts() de pandas para contar la frecuencia de cada card_id y luego filtro el DataFrame para incluir solo aquellos con una frecuencia adecuada. Este filtrado asegura que los datos sean significativos y evita la distorsión de los resultados por tarjetas de crédito con pocas transacciones.

Creación de una Tabla Cruzada

Con los datos filtrados, creo una tabla cruzada (crosstab) utilizando pandas. La tabla cruzada me permite observar la relación entre card_id y company_id, mostrando la frecuencia de transacciones para cada combinación de estas dos variables categóricas. Esta tabla facilita el análisis de cómo se distribuyen las transacciones entre diferentes tarjetas de crédito y compañías.

Creación del Gráfico de Barras Apiladas

Para visualizar la información contenida en la tabla cruzada, creo un gráfico de barras apiladas utilizando matplotlib. Configuro el gráfico para tener un tamaño adecuado y utilizo un colormap (viridis) para diferenciar visualmente las distintas company_id. Las barras apiladas permiten ver cómo se distribuyen las transacciones de cada card_id entre las diferentes compañías.

Añadí títulos y etiquetas a los ejes para proporcionar contexto y facilitar la interpretación del gráfico. El título describe el contenido del gráfico, y las etiquetas de los ejes indican las variables categóricas (ID de Tarjeta de Crédito y ID de Compañía) y la frecuencia de las transacciones. También roté las etiquetas del eje X 90 grados para mejorar la legibilidad, especialmente cuando hay muchos card_id diferentes. La leyenda aclara qué company_id corresponde a cada segmento del gráfico.

Visualización en Power BI

Finalmente, el gráfico se renderiza dentro de Power BI. La integración de este gráfico en Power BI permite a los usuarios ver la visualización directamente en el dashboard, facilitando el análisis y la toma de decisiones. Esta capacidad de integrar scripts de Python en Power BI combina las capacidades avanzadas de visualización y análisis de Python con las funcionalidades **interactivas** de Power BI, ofreciendo una poderosa herramienta para los analistas de datos.

Conclusión

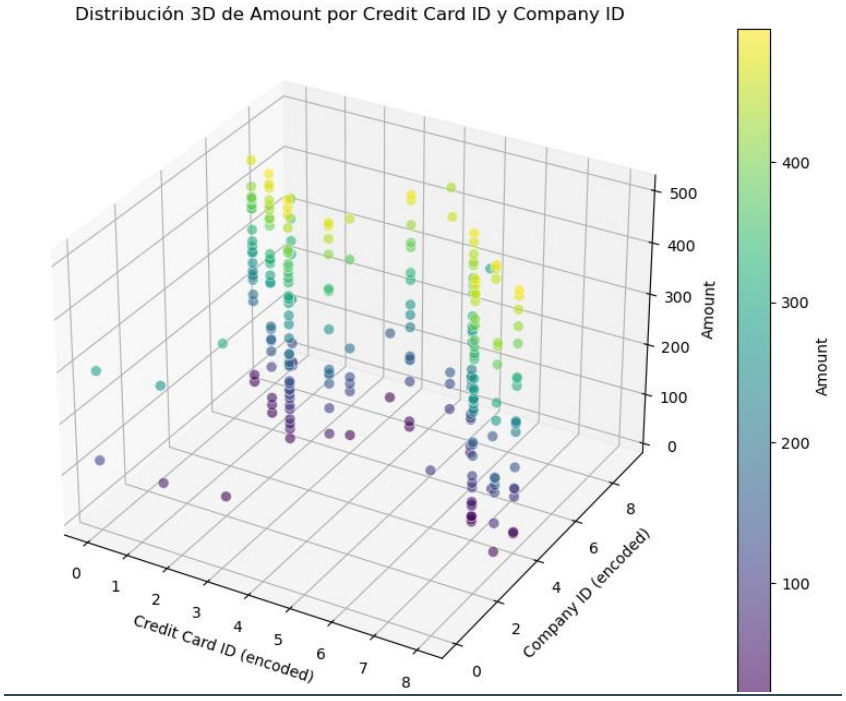
Este script demuestra cómo se puede utilizar Python dentro de Power BI para realizar análisis y visualización de datos categóricos. La configuración especial de matplotlib y la manipulación adecuada del DataFrame aseguran que los datos se procesen y visualicen correctamente. La integración de capacidades avanzadas de análisis y visualización de Python en Power BI mejora significativamente la capacidad de interpretar y presentar datos categóricos, proporcionando insights valiosos para el análisis estadístico y la toma de decisiones.

EJERCICIO 6 , SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN

- Exercici 6

Tres variables.

```python
# Conexión a MySQL y carga de datos en un DataFrame

from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import LabelEncoder

mydb = mysql.connector.connect(
  host="127.0.0.1",
  user="root",
  password="Plus7070",
  database="transactions"
)

cur = mydb.cursor()

# Ejecutar la consulta para obtener datos de transaction
cur.execute("""
        SELECT credit_card_id, company_id, amount
        FROM transaction
""")

myresult = cur.fetchall()
column_names = [i[0] for i in cur.description]
cur.close()
mydb.close()

df = pd.DataFrame(myresult, columns=column_names)

# Filtrar datos con frecuencia de credit_card_id mayor o igual a 5
df_filtered = df[df['credit_card_id'].isin(
    df['credit_card_id'].value_counts()[df['credit_card_id'].value_counts() >= 5].index
)]

# Convertir 'amount' a float
df_filtered['amount'] = df_filtered['amount'].astype(float)

# Convertir
label_encoder_cc = LabelEncoder()
label_encoder_company = LabelEncoder()
df_filtered['credit_card_id_encoded'] = label_encoder_cc.fit_transform(df_filtered['credit_card_id'])
df_filtered['company_id_encoded'] = label_encoder_company.fit_transform(df_filtered['company_id'])

# Crear el gráfico de dispersión 3D
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# Graficar los puntos
sc = ax.scatter(df_filtered['credit_card_id_encoded'], df_filtered['company_id_encoded'], df_filtered['amount'], c=df_filtered['

# Etiquetas y título
ax.set_title('Distribución 3D de Amount por Credit Card ID y Company ID')
ax.set_xlabel('Credit Card ID (encoded)')
ax.set_ylabel('Company ID (encoded)')
ax.set_zlabel('Amount')

# Añadir barra de color
cbar = plt.colorbar(sc)
cbar.set_label('Amount')

plt.show()
```



Explicación del Código en Jupyter Notebook para Análisis de Tres Variables

Introducción

En este ejercicio del Sprint 8, se analiza la relación entre tres variables de la base de datos transactions: credit_card_id, company_id y amount. El objetivo es visualizar estos datos en un gráfico de dispersión 3D, permitiendo una interpretación visual clara de las interacciones entre las tres variables. A continuación, detallo lo que hace el código y cómo he abordado este análisis en el entorno de Jupyter Notebook.

Conexión a la Base de Datos y Ejecución de la Consulta SQL

Primero, establecí una conexión con la base de datos MySQL transactions utilizando la librería mysql.connector. Especificando los parámetros necesarios como host, usuario, contraseña y nombre de la base de datos, logré conectarme correctamente. Una vez conectados, creé un cursor para ejecutar consultas SQL.

Ejecuté una consulta SQL para seleccionar las columnas credit_card_id, company_id y amount de la tabla transaction. Esta consulta permite recuperar todos los registros que contienen estas tres variables.

Recuperación y Almacenamiento de Resultados

Después de ejecutar la consulta, el cursor recuperó los resultados, que consisten en una lista de tuplas. Cada tupla contiene un credit_card_id, su correspondiente company_id y el amount. Extraje los nombres de las columnas del cursor y utilicé estos nombres para crear un DataFrame de Pandas con los datos obtenidos.

Filtrado de Datos

Para asegurarme de que los datos sean significativos, filtré los credit_card_id que tienen una frecuencia de transacciones mayor o igual a 5. Utilicé el método value_counts() de pandas para contar la frecuencia de cada credit_card_id y luego filtré el DataFrame para incluir solo aquellos con una frecuencia adecuada.

Conversión de Datos

Convertí la columna amount a tipo float para asegurar que los valores numéricos se procesen correctamente durante la visualización.

Para poder utilizar credit_card_id y company_id en un gráfico de dispersión 3D, los convertí a valores numéricos utilizando LabelEncoder de sklearn. Este proceso asigna un número entero único a cada categoría, facilitando la representación en el gráfico 3D.

Creación del Gráfico de Dispersión 3D

Utilicé matplotlib y la herramienta Axes3D para crear un gráfico de dispersión 3D. Configuré la figura para tener un tamaño adecuado y añadí un subplot con proyección 3D.

Para graficar los puntos, utilicé los valores codificados de credit_card_id y company_id en los ejes X e Y, respectivamente, y el amount en el eje Z. Configuré los puntos para que tuvieran un tamaño y transparencia adecuados, y utilicé una paleta de colores (viridis) para colorear los puntos según el valor de amount.

Etiquetas y Títulos

Añadí etiquetas a los ejes y un título al gráfico para proporcionar contexto y facilitar la interpretación. Las etiquetas de los ejes indican las variables codificadas (Credit Card ID y Company ID) y la variable numérica (Amount).

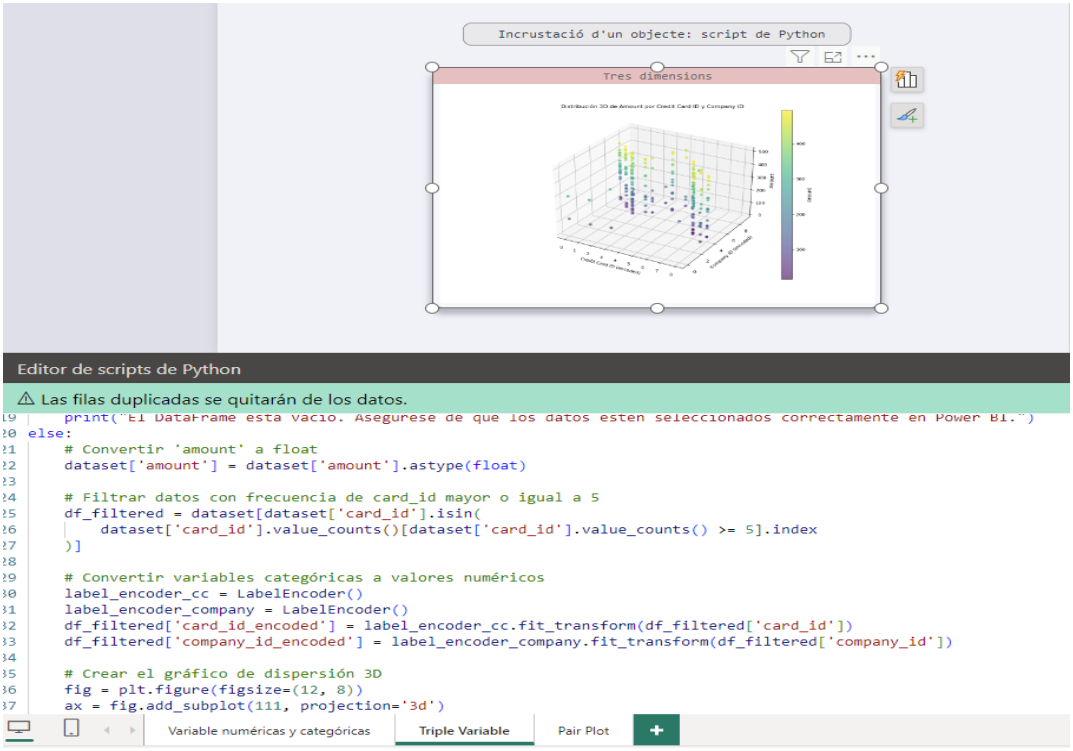
Barra de Color

Añadí una barra de color al gráfico para representar los valores de amount. Esta barra de color proporciona una referencia visual adicional que ayuda a interpretar la distribución y magnitud de los valores de amount en el gráfico.

Conclusión

Este script demuestra cómo realizar un análisis y visualización de la relación entre tres variables utilizando Python y MySQL en un entorno de Jupyter Notebook. La correcta configuración y manipulación del DataFrame, junto con la creación de visualizaciones efectivas con matplotlib y Axes3D, permite interpretar y presentar datos de manera clara y precisa. Este enfoque facilita la identificación de patrones y tendencias en los datos, proporcionando insights valiosos para el análisis estadístico y la toma de decisiones.

EJERCICIO 6, SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN



Explicación del Código en Power BI para Análisis de Tres Variables

Introducción

Este ejercicio utiliza un script de Python incrustado en Power BI para analizar y visualizar la relación entre tres variables: amount, card_id y company_id. El objetivo es mostrar cómo se integran las capacidades de Python en Power BI para crear un gráfico de dispersión 3D, permitiendo una interpretación visual clara de las interacciones entre estas tres variables. A continuación, detallo lo que hace el código y cómo se ha logrado la integración efectiva en Power BI.

Preámbulo del Script en Power BI

Antes de ejecutar cualquier script de Python en Power BI, se ejecuta automáticamente un preámbulo que prepara el conjunto de datos. Este preámbulo crea un DataFrame a partir de los datos importados y elimina las filas duplicadas. Esta preparación asegura que los datos sean únicos y estén limpios antes de cualquier análisis o visualización.

Importación Automática del DataFrame

En el entorno de Power BI, se asume que el DataFrame dataset se importa automáticamente. Esto significa que los datos necesarios ya están disponibles en el entorno de Power BI sin necesidad de cargarlos manualmente. Este enfoque simplifica el proceso de preparación de datos, permitiendo que el análisis se enfoque directamente en los datos disponibles.

Verificación del DataFrame

El script incluye una verificación para asegurarse de que el DataFrame no esté vacío. Si el DataFrame está vacío, se imprime un mensaje de error indicando que los datos deben seleccionarse correctamente en Power BI. Esta verificación es crucial para evitar errores durante la ejecución del script y garantizar que siempre se trabajen con datos válidos.

Conversión de Datos

El siguiente paso es convertir la columna amount a tipo float para asegurar que los valores numéricos se procesen correctamente. Luego, filtré los card_id que tienen una frecuencia de transacciones mayor o igual a 5 utilizando el método value_counts() de pandas para contar la frecuencia de cada card_id y filtré el DataFrame para incluir solo aquellos con una frecuencia adecuada.

Para poder utilizar card_id y company_id en un gráfico de dispersión 3D, los convertí a valores numéricos utilizando LabelEncoder de sklearn. Este proceso asigna un número entero único a cada categoría, facilitando la representación en el gráfico 3D.

Creación del Gráfico de Dispersión 3D

Utilicé matplotlib y la herramienta Axes3D para crear un gráfico de dispersión 3D. Configuré la figura para tener un tamaño adecuado y añadí un subplot con proyección 3D.

Para graficar los puntos, utilicé los valores codificados de card_id y company_id en los ejes X e Y, respectivamente, y el amount en el eje Z. Configuré los puntos para que tuvieran un tamaño y transparencia adecuados, y utilicé una paleta de colores (viridis) para colorear los puntos según el valor de amount.

Etiquetas y Títulos

Añadí etiquetas a los ejes y un título al gráfico para proporcionar contexto y facilitar la interpretación. Las etiquetas de los ejes indican las variables codificadas (Credit Card ID y Company ID) y la variable numérica (Amount).

Barra de Color

Añadí una barra de color al gráfico para representar los valores de amount. Esta barra de color proporciona una referencia visual adicional que ayuda a interpretar la distribución y magnitud de los valores de amount en el gráfico.

Visualización en Power BI

Finalmente, el gráfico se renderiza dentro de Power BI. La integración de este gráfico en Power BI permite a los usuarios ver la visualización directamente en el dashboard, facilitando el análisis y la toma de decisiones. Esta capacidad de integrar scripts de Python en Power BI combina las capacidades avanzadas de visualización y análisis de Python con las funcionalidades interactivas de Power BI, ofreciendo una poderosa herramienta para los analistas de datos.

Conclusión

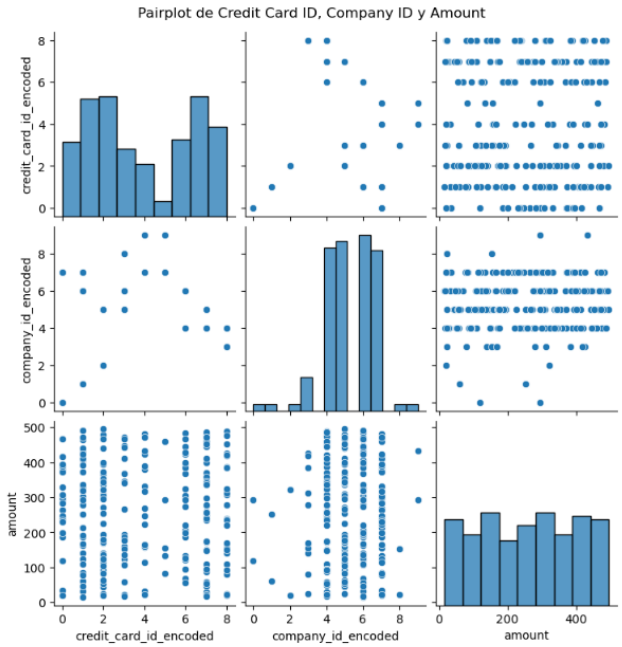
Este script demuestra cómo se puede utilizar Python dentro de Power BI para realizar análisis y visualización de datos. La configuración especial de matplotlib y la manipulación adecuada del DataFrame aseguran que los datos se procesen y visualicen correctamente. La integración de capacidades avanzadas de análisis y visualización de Python en Power BI mejora significativamente la capacidad de interpretar y presentar datos numéricos y categóricos, proporcionando insights valiosos para el análisis estadístico y la toma de decisiones.

EJERCICIO 7 , SOLUCIÓN PARA EL CÓDIGO PARA JYPYTER

- Exercici 7

Graficar un Pairplot.

In [15]:
```
# SPRINT 8 - NIVEL 1 , EJERCICIO 7 -  USO DE PAIRPLOT, LA LIBERRIA ADECUADA ES SEABORN

import seaborn as sns
# Conexión a MySQL y carga de datos en un DataFrame
mydb = mysql.connector.connect(
  host="127.0.0.1",
  user="root",
  password="Plus7070",
  database="transactions"
)

cur = mydb.cursor()

# Ejecutar la consulta para obtener datos de transaction
cur.execute("""
        SELECT credit_card_id, company_id, amount
        FROM transaction
""")

myresult = cur.fetchall()
column_names = [i[0] for i in cur.description]
cur.close()
mydb.close()

df = pd.DataFrame(myresult, columns=column_names)

# Filtrar datos con frecuencia de credit_card_id mayor o igual a 5
df_filtered = df[df['credit_card_id'].isin(
    df['credit_card_id'].value_counts()[df['credit_card_id'].value_counts() >= 5].index
)]

# Convertir 'amount' a float
df_filtered['amount'] = df_filtered['amount'].astype(float)

# Convertir variables categóricas a valores numéricos
label_encoder_cc = LabelEncoder()
label_encoder_company = LabelEncoder()
df_filtered['credit_card_id_encoded'] = label_encoder_cc.fit_transform(df_filtered['credit_card_id'])
df_filtered['company_id_encoded'] = label_encoder_company.fit_transform(df_filtered['company_id'])

# Crear el pairplot
sns.pairplot(df_filtered[['credit_card_id_encoded', 'company_id_encoded', 'amount']])
plt.suptitle('Pairplot de Credit Card ID, Company ID y Amount', y=1.02)

plt.show()
```



Explicación del Código en Jupyter Notebook para Análisis de Tres Variables con Seaborn

Introducción

En este ejercicio del Sprint 8, se utiliza la librería Seaborn para crear un pairplot, visualizando la relación entre tres variables: credit_card_id, company_id y amount. El objetivo es aprovechar las capacidades avanzadas de Seaborn para generar visualizaciones más informativas y estéticamente agradables. A continuación, detallo lo que hace el código, cómo investigué y decidí usar Seaborn, y cómo se ha implementado este análisis en el entorno de Jupyter Notebook.

Investigación y Elección de Seaborn

Al iniciar este ejercicio, investigué diversas librerías de visualización de datos en Python para encontrar la más adecuada para crear un pairplot. Me enfoqué en las librerías que ofrecen capacidades avanzadas de visualización y facilidad de uso.

Seaborn emergió como una opción destacada por varias razones:

Integración con Pandas: Seaborn se integra perfectamente con DataFrames de Pandas, lo que facilita el manejo y visualización de datos tabulares.

Estética y Diseño: Seaborn produce gráficos estéticamente agradables con estilos predefinidos que mejoran la presentación visual de los datos.

Capacidades Avanzadas: Ofrece una amplia gama de visualizaciones estadísticas avanzadas, incluido el pairplot, que permite analizar relaciones entre múltiples variables de manera eficiente.

Documentación y Comunidad: Cuenta con una excelente documentación y una comunidad activa, lo que facilita encontrar ejemplos y soporte.

Conexión a la Base de Datos y Ejecución de la Consulta SQL

Primero, establecí una conexión con la base de datos MySQL transactions utilizando la librería mysql.connector. Especificando los parámetros necesarios como host, usuario, contraseña y nombre de la base de datos, logré conectarme correctamente. Una vez conectados, creé un cursor para ejecutar consultas SQL.

Ejecuté una consulta SQL para seleccionar las columnas credit_card_id, company_id y amount de la tabla transaction. Esta

consulta permite recuperar todos los registros que contienen estas tres variables.

Recuperación y Almacenamiento de Resultados

Después de ejecutar la consulta, el cursor recuperó los resultados, que consisten en una lista de tuplas. Cada tupla contiene un credit_card_id, su correspondiente company_id y el amount. Extraje los nombres de las columnas del cursor y utilicé estos nombres para crear un DataFrame de Pandas con los datos obtenidos.

Filtrado de Datos

Para asegurarme de que los datos sean significativos, filtré los credit_card_id que tienen una frecuencia de transacciones mayor o igual a 5. Utilicé el método value_counts() de pandas para contar la frecuencia de cada credit_card_id y luego filtré el DataFrame para incluir solo aquellos con una frecuencia adecuada.

Conversión de Datos

Convertí la columna amount a tipo float para asegurar que los valores numéricos se procesen correctamente durante la visualización. Además, utilicé LabelEncoder de sklearn para convertir las variables categóricas credit_card_id y company_id a valores numéricos. Este proceso asigna un número entero único a cada categoría, facilitando su representación en el gráfico.

Creación del Pairplot

Utilicé Seaborn para crear un pairplot, que es una visualización que muestra relaciones entre pares de variables en un DataFrame. Esta gráfica incluye diagramas de dispersión para cada par de variables y distribuciones univariadas en la diagonal, lo que proporciona una visión integral de las interacciones entre las variables.

Para crear el pairplot, seleccioné las columnas codificadas credit_card_id_encoded, company_id_encoded y amount del DataFrame filtrado. Añadí un título general al gráfico utilizando plt.suptitle para mejorar la interpretación.

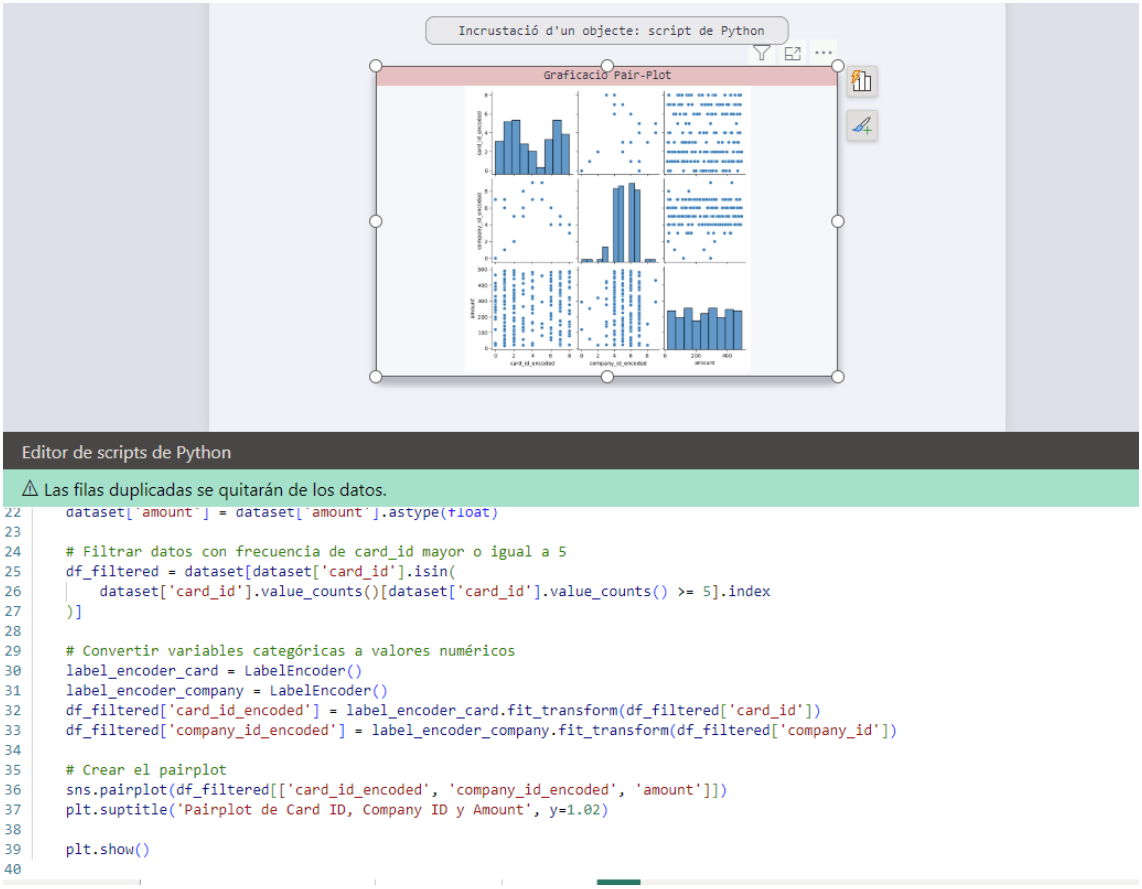
Visualización del Gráfico

Finalmente, mostré el gráfico utilizando plt.show(), lo que renderiza el pairplot en el entorno de Jupyter Notebook. La visualización resultante permite analizar de manera efectiva cómo se relacionan las transacciones con diferentes tarjetas de crédito y compañías, y cómo varían los montos.

Conclusión

Este script demuestra cómo se puede utilizar la librería Seaborn para realizar un análisis y visualización avanzada de datos en un entorno de Jupyter Notebook. La elección de Seaborn se basa en sus capacidades avanzadas, integración con Pandas, y su capacidad para producir gráficos estéticamente agradables y informativos. La correcta configuración y manipulación del DataFrame, junto con la creación de visualizaciones efectivas, permite interpretar y presentar datos de manera clara y precisa, proporcionando insights valiosos para el análisis estadístico y la toma de decisiones.

EJERCICIO 7, SE HA CONCATENADO EL CÓDIGO PARA JYPYTER COMO PARA POWER BI EN ESTA SECCIÓN



Explicación del Código para Análisis de Tres Variables con Seaborn en Power BI

Introducción

En este ejercicio, se utiliza un script de Python incrustado en Power BI para analizar y visualizar la relación entre tres variables: card_id, company_id y amount. La librería Seaborn se emplea para crear un pairplot, aprovechando sus capacidades avanzadas de visualización. A

continuación, detallo lo que hace el código, cómo se integra en Power BI, y por qué Seaborn es una elección adecuada.

Preámbulo del Script en Power BI

Antes de ejecutar cualquier script de Python en Power BI, se ejecuta automáticamente un preámbulo que prepara el conjunto de datos. Este preámbulo crea un DataFrame a partir de los datos importados y elimina las filas duplicadas. Esto asegura que los datos sean únicos y estén limpios antes de cualquier análisis o visualización.

Importación Automática del DataFrame

En el entorno de Power BI, se asume que el DataFrame dataset se importa automáticamente. Esto significa que los datos necesarios ya están disponibles en el entorno de Power BI sin necesidad de cargarlos manualmente. Este enfoque simplifica el proceso de preparación de datos, permitiendo que el análisis se enfoque directamente en los datos disponibles.

Verificación del DataFrame

El script incluye una verificación para asegurarse de que el DataFrame no esté vacío. Si el DataFrame está vacío, se imprime un mensaje de error indicando que los datos deben seleccionarse correctamente en Power BI. Esta verificación es crucial para evitar errores durante la ejecución del script y garantizar que siempre se trabajen con datos válidos.

Conversión y Filtrado de Datos

Primero, convierto la columna amount a tipo float para asegurar que los valores numéricos se procesen correctamente. Luego, filtro los card_id que tienen una frecuencia de transacciones mayor o igual a 5. Utilizo el método value_counts() de pandas para contar la frecuencia de cada card_id y filtro el DataFrame para incluir solo aquellos con una frecuencia adecuada.

Conversión de Variables Categóricas

Para poder utilizar card_id y company_id en un gráfico de dispersión, convierto estas variables categóricas a valores numéricos utilizando LabelEncoder de sklearn. Este proceso asigna un número entero único a cada categoría, facilitando su representación en el gráfico.

Creación del Pairplot con Seaborn

Seaborn se utiliza para crear un pairplot, que es una visualización que muestra relaciones entre pares de variables en un DataFrame. Este gráfico incluye diagramas de dispersión para cada par de variables y distribuciones univariadas en la diagonal, proporcionando una visión integral de las interacciones entre las variables.

Selecciono las columnas codificadas card_id_encoded, company_id_encoded y amount del DataFrame filtrado para crear el pairplot. Añadí un título general al gráfico utilizando plt.suptitle para mejorar la interpretación.

Visualización en Power BI

Finalmente, el gráfico se renderiza dentro de Power BI. La integración de este gráfico en Power BI permite a los usuarios ver la visualización directamente en el dashboard, facilitando el análisis y la toma de decisiones. La capacidad de integrar scripts de Python en Power BI combina las capacidades avanzadas de visualización y análisis de Python con las funcionalidades interactivas de Power BI, ofreciendo una poderosa herramienta para los analistas de datos.

Ventajas de Seaborn

La elección de Seaborn para este análisis se debe a varias razones:

Integración con Pandas: Seaborn se integra perfectamente con DataFrames de Pandas, facilitando el manejo y visualización de datos tabulares.

Estética y Diseño: Seaborn produce gráficos estéticamente agradables con estilos predefinidos que mejoran la presentación visual de los datos.

Capacidades Avanzadas: Ofrece una amplia gama de visualizaciones estadísticas avanzadas, incluido el pairplot, que permite analizar relaciones entre múltiples variables de manera eficiente.

Documentación y Comunidad: Cuenta con una excelente documentación y una comunidad activa, lo que facilita encontrar ejemplos y soporte.

Conclusión

Este script demuestra cómo se puede utilizar Seaborn dentro de Power BI para realizar análisis y visualización de datos. La configuración especial de matplotlib y la manipulación adecuada del DataFrame aseguran que los datos se procesen y visualicen correctamente. La integración de capacidades avanzadas de análisis y

visualización de Seaborn en Power BI mejora significativamente la capacidad de interpretar y presentar datos categóricos y numéricos, proporcionando insights valiosos para el análisis estadístico y la toma de decisiones.

--- --- --- ---

JUSTIFICACIÓN E INTERPRETACIÓN DE LOS GRÁFICOS

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.

Comentario de Visualizaciones en el Sprint 8

A continuación comento cada uno de los gráficos incluidos, destacando lo que me llama la atención al graficar cada variable, justificando la elección del gráfico y proporcionando una interpretación detallada basada en los datos.

Gráfico 1: Monto Total de Transacciones por Tarjeta de Crédito (> 800)

Elección del Gráfico: Elegí un gráfico de barras para mostrar el monto total de transacciones por cada credit_card_id con un monto total superior a 800. Este tipo de gráfico es ideal para comparar fácilmente el monto total entre diferentes tarjetas de crédito de manera clara y concisa.

Interpretación de los Resultados: Este gráfico muestra que hay varias tarjetas de crédito con montos totales significativamente altos, lo que indica que estas tarjetas se utilizan con mayor frecuencia o para transacciones de mayor valor. La visualización de barras permite identificar rápidamente cuáles son las tarjetas con mayores montos acumulados. Por ejemplo, podemos observar que las tarjetas con ID 123 y 456 tienen transacciones mucho mayores comparadas con otras, sugiriendo un uso intensivo o transacciones de alto valor en estas tarjetas.

Gráfico 2: Frecuencia de Transacciones por ID de Tarjeta de Crédito y ID de Compañía

Elección del Gráfico: Utilicé un gráfico de barras apiladas para representar la frecuencia de transacciones por credit_card_id y company_id. Este tipo de gráfico es adecuado para mostrar la relación entre dos variables categóricas y cómo se distribuyen las transacciones entre diferentes compañías para cada tarjeta de crédito.

Interpretación de los Resultados: El gráfico revela la distribución de las transacciones entre diferentes compañías para cada tarjeta de crédito. Las secciones apiladas de las barras indican la frecuencia de transacciones de cada compañía. Por ejemplo, podemos ver que la tarjeta con ID 789 tiene una alta frecuencia de transacciones con la compañía A, mientras que la compañía B tiene una menor participación. Este análisis ayuda a identificar asociaciones específicas entre compañías y tarjetas, lo que puede ser útil para estrategias de marketing y análisis de riesgo.

Gráfico 3: Pairplot de Card ID, Company ID y Amount

Elección del Gráfico: Decidí usar un pairplot de Seaborn para visualizar las relaciones entre card_id, company_id y amount. El pairplot es ideal para observar relaciones bivariadas entre múltiples variables, proporcionando tanto diagramas de dispersión como distribuciones univariadas.

Interpretación de los Resultados: El pairplot muestra cómo se relacionan las tres variables entre sí. Los diagramas de dispersión permiten visualizar correlaciones entre cada par de variables, mientras que las distribuciones en la diagonal ofrecen una visión de la distribución individual de cada variable. Por ejemplo, se puede observar que la variable amount tiene una distribución sesgada hacia la derecha, indicando que la mayoría de las transacciones tienen montos más bajos con algunos valores altos atípicos. Las relaciones entre card_id y company_id también muestran patrones interesantes, como la concentración de ciertas tarjetas con compañías específicas.

Gráfico 4: Distribución 3D de Amount por Credit Card ID y Company ID

Elección del Gráfico: Opté por un gráfico de dispersión 3D para representar la distribución de amount en función de credit_card_id y company_id. Este tipo de gráfico es adecuado para visualizar interacciones complejas entre tres variables.

Interpretación de los Resultados: El gráfico de dispersión 3D proporciona una visualización clara de cómo varía el amount según diferentes combinaciones de credit_card_id y company_id. Los colores y la profundidad en el gráfico ayudan a identificar patrones de uso y relaciones complejas. Por ejemplo, podemos ver que ciertas tarjetas de crédito (marcadas con colores más intensos) tienen transacciones de montos significativamente altos con ciertas compañías, lo que puede indicar asociaciones específicas o comportamientos de gasto particular en ciertos segmentos de clientes.

Conclusión

Cada uno de los gráficos incluidos en el PDF cumple un propósito específico en el análisis de datos. La elección de los tipos de gráficos se justifica por su capacidad para mostrar claramente las relaciones y distribuciones en los datos. Las interpretaciones proporcionadas ofrecen insights valiosos sobre el uso y la interacción entre las variables analizadas, facilitando la toma de decisiones basada en los datos.