Business Challenge: Forex Data Analysis Challenge

En este ejercicio, nos embarcaremos en un desafío de análisis de datos de Forex, simulando un escenario donde necesitas analizar datos históricos de tasas de cambio extranjero (Forex) para un par de divisas. Se te ha proporcionado un conjunto de datos que contiene tasas de Forex diarias para un par de divisas específico durante un mes. Tu objetivo es aprovechar NumPy para extraer valiosas percepciones de estos datos financieros.

Descripción del Dataset

El conjunto de datos comprende tres arrays de NumPy:

- `dates`: Un array que contiene valores de fecha que representan cada día de negociación del mes.

```
dates = np.array(['2023-10-01', '2023-10-02', '2023-10-03', '2023-10-
04',
                  '2023-10-05', '2023-10-06', '2023-10-07', '2023-10-
08',
                  '2023-10-09', '2023-10-10', '2023-10-11', '2023-10-
12',
                  '2023-10-13', '2023-10-14', '2023-10-15', '2023-10-
16',
                  '2023-10-17', '2023-10-18', '2023-10-19', '2023-10-
20',
                  '2023-10-21', '2023-10-22', '2023-10-23', '2023-10-
24',
                  '2023-10-25', '2023-10-26', '2023-10-27', '2023-10-
28',
                  '2023-10-29', '2023-10-30', '2023-10-31'])
```

- `exchange_rates`: Un array de valores de tasas de cambio para un par de divisas específico correspondiente a cada fecha.

```
exchange_rates = np.array([1.10, 1.12, 1.15, 1.11, 1.14, 1.13, 1.10,
1.12,
                           1.13, 1.11, 1.15, 1.16, 1.18, 1.20, 1.19,
1.17,
                           1.16, 1.14, 1.12, 1.13, 1.11, 1.10, 1.12,
1.13,
                           1.14, 1.15, 1.11, 1.10, 1.08, 1.09, 1.11,
1.13])
```

- `trade_volumes`: Un array que representa los volúmenes de negociación para cada día de negociación.

```
trade_volumes = np.array([1000, 1200, 1100, 1300, 1500, 1400, 1600,
1700,
                          1800, 1900, 2100, 2200, 2000, 2300, 2400,
2500,
                          2600, 2700, 2800, 2900, 3000, 3100, 3200,
3300,
                          3400, 3500, 3600, 3700, 3800, 3900, 4000,
4100,
                          4200])
```

- `external_variable`: Un array que representa un índice económico (por ejemplo, el índice de precios al consumidor) que puede ser utilizado para calcular la correlación.

```
external_variable = np.array([102, 103, 101, 104, 105, 106, 107, 108,
                              109, 110, 111, 112, 113, 114, 115, 116,
                              117, 118, 119, 120, 121, 122, 123, 124,
                              125, 126, 127, 128, 129, 130, 131, 132,
                              133])
```

Tus tareas involucrarán:

1. Calcular la tasa de cambio promedio para todo el mes.
2. Identificar la fecha con la tasa de cambio más alta.
3. Determinar el volumen total de negociación para el mes.
4. Calcular la tasa de cambio mínima y máxima del mes.
5. Determinar la volatilidad diaria de las tasas de cambio (diferencias entre días consecutivos).
6. Graficar la evolución de las tasas de cambio a lo largo del mes.
7. Comparar el rendimiento de las tasas de cambio en la primera y última semana del mes.
8. Calcular la correlación de las tasas de cambio con respecto a una variable externa (por ejemplo, un índice económico).
9. Determinar el número de días en los que la tasa de cambio superó un valor específico (por ejemplo, 1.15).
10. Crear un resumen estadístico que incluya la media, mediana y desviación estándar de las tasas de cambio.

Al final de este ejercicio, ganarás experiencia práctica en el uso de NumPy para el análisis de datos financieros, una habilidad valiosa para profesionales en el mundo del trading de Forex y las finanzas.

¡Comencemos calculando la tasa de cambio promedio para todo el mes!

¡Pista!: Revisa la documentación aquí.

```
import numpy as np

dates = np.array(['2023-10-01', '2023-10-02', '2023-10-03', '2023-10-
04',
                  '2023-10-05', '2023-10-06', '2023-10-07', '2023-10-
```

```
08',
                   '2023-10-09', '2023-10-10', '2023-10-11', '2023-10-
12',
                   '2023-10-13', '2023-10-14', '2023-10-15', '2023-10-
16',
                   '2023-10-17', '2023-10-18', '2023-10-19', '2023-10-
20',
                   '2023-10-21', '2023-10-22', '2023-10-23', '2023-10-
24',
                   '2023-10-25', '2023-10-26', '2023-10-27', '2023-10-
28',
                   '2023-10-29', '2023-10-30', '2023-10-31'])

exchange_rates = np.array([1.10, 1.12, 1.15, 1.11, 1.14, 1.13, 1.10,
1.12,
                           1.13, 1.11, 1.15, 1.16, 1.18, 1.20, 1.19,
1.17,
                           1.16, 1.14, 1.12, 1.13, 1.11, 1.10, 1.12,
1.13,
                           1.14, 1.15, 1.11, 1.10, 1.08, 1.09, 1.11,
1.13])

trade_volumes = np.array([1000, 1200, 1100, 1300, 1500, 1400, 1600,
1700,
                          1800, 1900, 2100, 2200, 2000, 2300, 2400,
2500,
                          2600, 2700, 2800, 2900, 3000, 3100, 3200,
3300,
                          3400, 3500, 3600, 3700, 3800, 3900, 4000,
4100,
                          4200])

external_variable = np.array([102, 103, 101, 104, 105, 106, 107, 108,
                              109, 110, 111, 112, 113, 114, 115, 116,
                              117, 118, 119, 120, 121, 122, 123, 124,
                              125, 126, 127, 128, 129, 130, 131, 132,
                              133])
```

0. Analizar relacion de los datos mediante dispersión

```
import matplotlib.pyplot as plt

print("dates: ", dates.size)
print("exchange_rates: ", exchange_rates.size)
print("trade_volumes: ", trade_volumes.size)
print("external_variable: ", external_variable.size)
```

```
dates:  31
exchange_rates:  32
```

```
trade_volumes:  33
external_variable:  33

valor_minimo = np.min([dates.size, exchange_rates.size,
trade_volumes.size, external_variable.size])

dates = dates[:valor_minimo]
exchange_rates = exchange_rates[:valor_minimo]
trade_volumes = trade_volumes[:valor_minimo]
external_variable = external_variable[:valor_minimo]

tabla = []

for (j,k,l,m) in zip(dates, exchange_rates, trade_volumes,
external_variable):
    tabla.append([j, k, l, m])

tabla

[['2023-10-01', 1.1, 1000, 102],
 ['2023-10-02', 1.12, 1200, 103],
 ['2023-10-03', 1.15, 1100, 101],
 ['2023-10-04', 1.11, 1300, 104],
 ['2023-10-05', 1.14, 1500, 105],
 ['2023-10-06', 1.13, 1400, 106],
 ['2023-10-07', 1.1, 1600, 107],
 ['2023-10-08', 1.12, 1700, 108],
 ['2023-10-09', 1.13, 1800, 109],
 ['2023-10-10', 1.11, 1900, 110],
 ['2023-10-11', 1.15, 2100, 111],
 ['2023-10-12', 1.16, 2200, 112],
 ['2023-10-13', 1.18, 2000, 113],
 ['2023-10-14', 1.2, 2300, 114],
 ['2023-10-15', 1.19, 2400, 115],
 ['2023-10-16', 1.17, 2500, 116],
 ['2023-10-17', 1.16, 2600, 117],
 ['2023-10-18', 1.14, 2700, 118],
 ['2023-10-19', 1.12, 2800, 119],
 ['2023-10-20', 1.13, 2900, 120],
 ['2023-10-21', 1.11, 3000, 121],
 ['2023-10-22', 1.1, 3100, 122],
 ['2023-10-23', 1.12, 3200, 123],
 ['2023-10-24', 1.13, 3300, 124],
 ['2023-10-25', 1.14, 3400, 125],
 ['2023-10-26', 1.15, 3500, 126],
 ['2023-10-27', 1.11, 3600, 127],
 ['2023-10-28', 1.1, 3700, 128],
 ['2023-10-29', 1.08, 3800, 129],
 ['2023-10-30', 1.09, 3900, 130],
 ['2023-10-31', 1.11, 4000, 131]]
```

```
tabla = np.array(tabla)
tabla
```

```
array([['2023-10-01', '1.1', '1000', '102'],
       ['2023-10-02', '1.12', '1200', '103'],
       ['2023-10-03', '1.15', '1100', '101'],
       ['2023-10-04', '1.11', '1300', '104'],
       ['2023-10-05', '1.14', '1500', '105'],
       ['2023-10-06', '1.13', '1400', '106'],
       ['2023-10-07', '1.1', '1600', '107'],
       ['2023-10-08', '1.12', '1700', '108'],
       ['2023-10-09', '1.13', '1800', '109'],
       ['2023-10-10', '1.11', '1900', '110'],
       ['2023-10-11', '1.15', '2100', '111'],
       ['2023-10-12', '1.16', '2200', '112'],
       ['2023-10-13', '1.18', '2000', '113'],
       ['2023-10-14', '1.2', '2300', '114'],
       ['2023-10-15', '1.19', '2400', '115'],
       ['2023-10-16', '1.17', '2500', '116'],
       ['2023-10-17', '1.16', '2600', '117'],
       ['2023-10-18', '1.14', '2700', '118'],
       ['2023-10-19', '1.12', '2800', '119'],
       ['2023-10-20', '1.13', '2900', '120'],
       ['2023-10-21', '1.11', '3000', '121'],
       ['2023-10-22', '1.1', '3100', '122'],
       ['2023-10-23', '1.12', '3200', '123'],
       ['2023-10-24', '1.13', '3300', '124'],
       ['2023-10-25', '1.14', '3400', '125'],
       ['2023-10-26', '1.15', '3500', '126'],
       ['2023-10-27', '1.11', '3600', '127'],
       ['2023-10-28', '1.1', '3700', '128'],
       ['2023-10-29', '1.08', '3800', '129'],
       ['2023-10-30', '1.09', '3900', '130'],
       ['2023-10-31', '1.11', '4000', '131']], dtype='<U32')
```

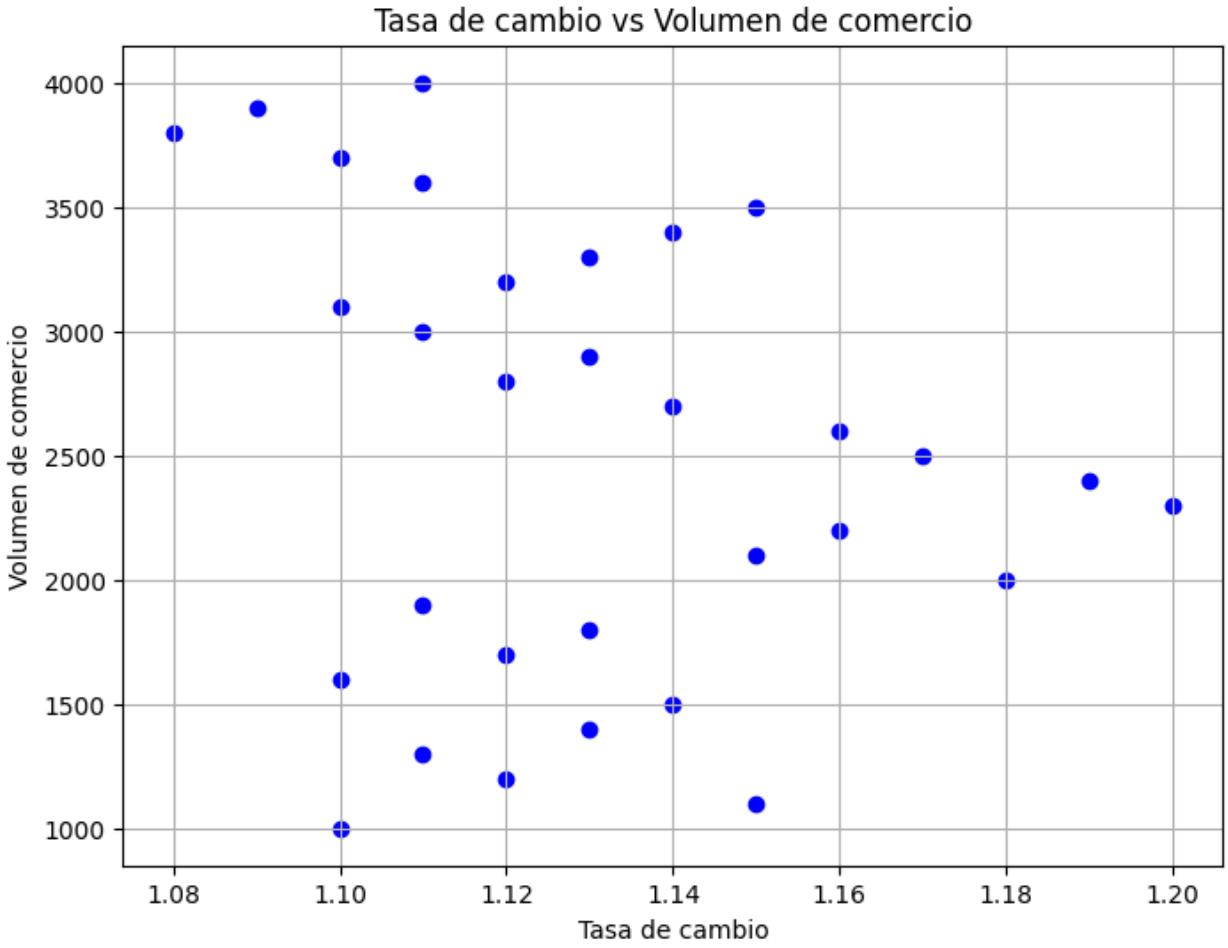
```
dates2 = tabla[:, 0]
dates == dates2
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True])
```

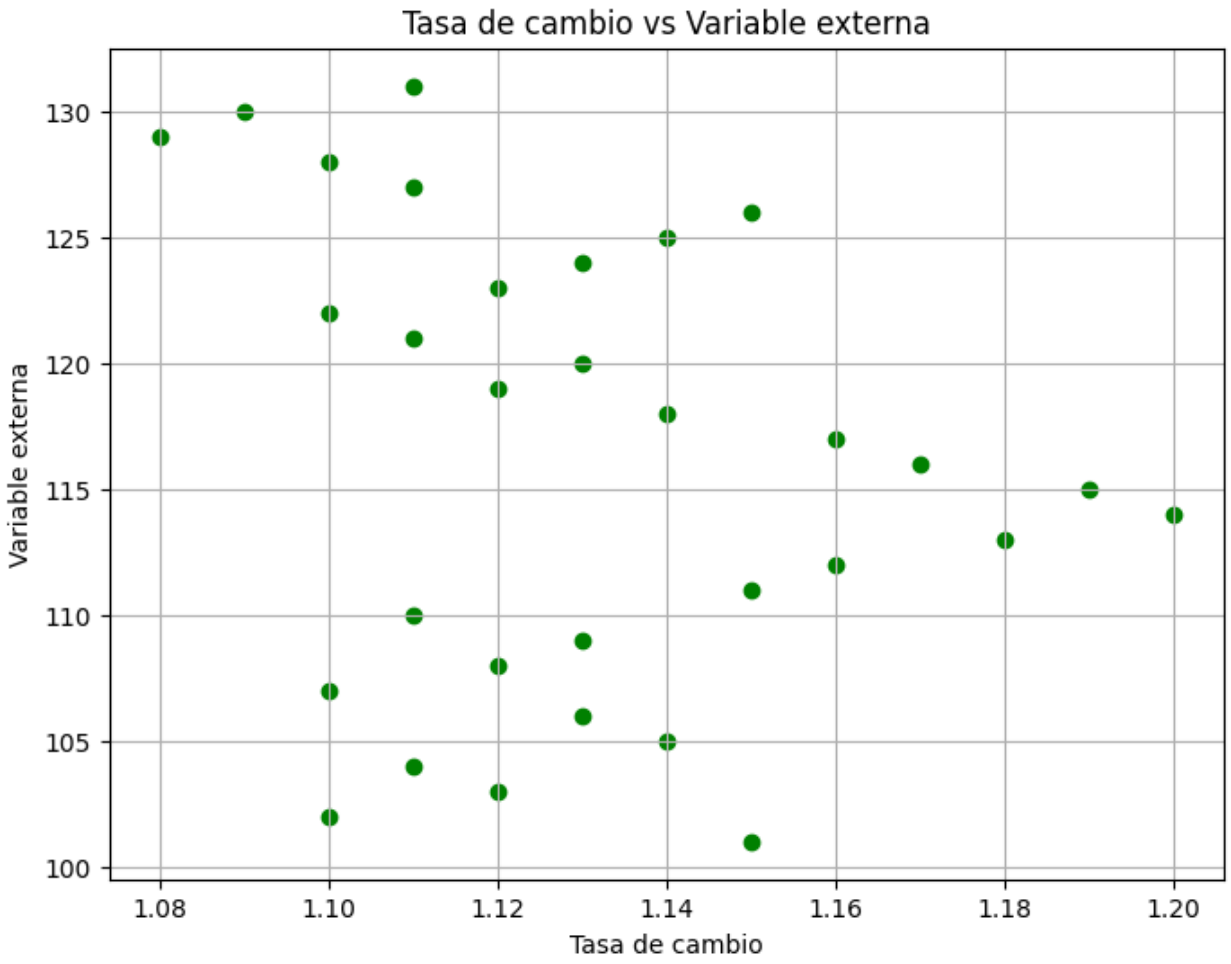
```
exchange_rates2 = tabla[:, 1]
exchange_rates == exchange_rates2.astype(float)
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True])
```

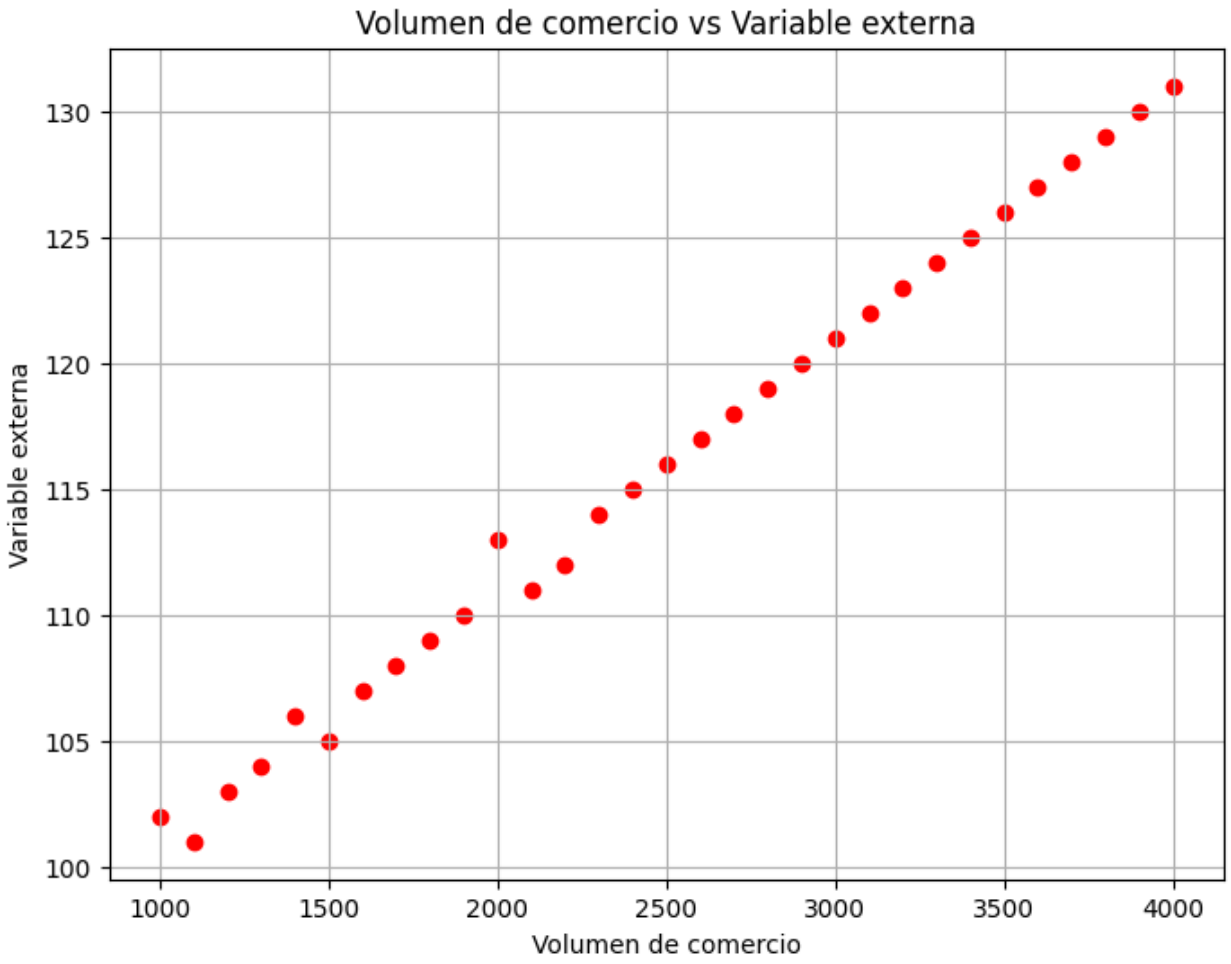
```
# 1. Gráfico de dispersión: Tasa de cambio vs Volumen de comercio
plt.figure(figsize=(8, 6))
plt.scatter(exchange_rates, trade_volumes, color='blue')
plt.title('Tasa de cambio vs Volumen de comercio')
plt.xlabel('Tasa de cambio')
plt.ylabel('Volumen de comercio')
plt.grid(True)
plt.show()
```




```
# 2. Gráfico de dispersión: Tasa de cambio vs Variable externa
plt.figure(figsize=(8, 6))
plt.scatter(exchange_rates, external_variable, color='green')
plt.title('Tasa de cambio vs Variable externa')
plt.xlabel('Tasa de cambio')
plt.ylabel('Variable externa')
plt.grid(True)
plt.show()
```




```
# 3. Gráfico de dispersión: Volumen de comercio vs Variable externa
plt.figure(figsize=(8, 6))
plt.scatter(trade_volumes, external_variable, color='red')
plt.title('Volumen de comercio vs Variable externa')
plt.xlabel('Volumen de comercio')
plt.ylabel('Variable externa')
plt.grid(True)
plt.show()
```



1. Calcular la tasa de cambio promedio para todo el mes

```
average_exchange_rate = np.mean(exchange_rates)
print("Tasa de cambio promedio:", average_exchange_rate)
```

```
Tasa de cambio promedio: 1.1306451612903228
```

```
1.1306451612903228 == 1.1306451612903228
```

```
True
```

2. Identificar la fecha con la tasa de cambio más alta

```
max_exchange_rate = np.max(exchange_rates)
max_exchange_rate_date = dates[np.argmax(exchange_rates)]
print("Tasa de cambio más alta:", max_exchange_rate)
print("Fecha con la tasa de cambio más alta:", max_exchange_rate_date)
```

```
Tasa de cambio más alta: 1.2
Fecha con la tasa de cambio más alta: 2023-10-14
```

3. Determinar el volumen total de negociación para el mes

```
total_trade_volume = np.sum(trade_volumes)
print("Volumen total de negociación:", total_trade_volume)
```

```
Volumen total de negociación: 77500
```

4. Calcular la tasa de cambio mínima y máxima del mes

```
min_exchange_rate = np.min(exchange_rates)
max_exchange_rate = np.max(exchange_rates)
print("Tasa de cambio mínima:", min_exchange_rate)
print("Tasa de cambio máxima:", max_exchange_rate)
```

```
Tasa de cambio mínima: 1.08
Tasa de cambio máxima: 1.2
```

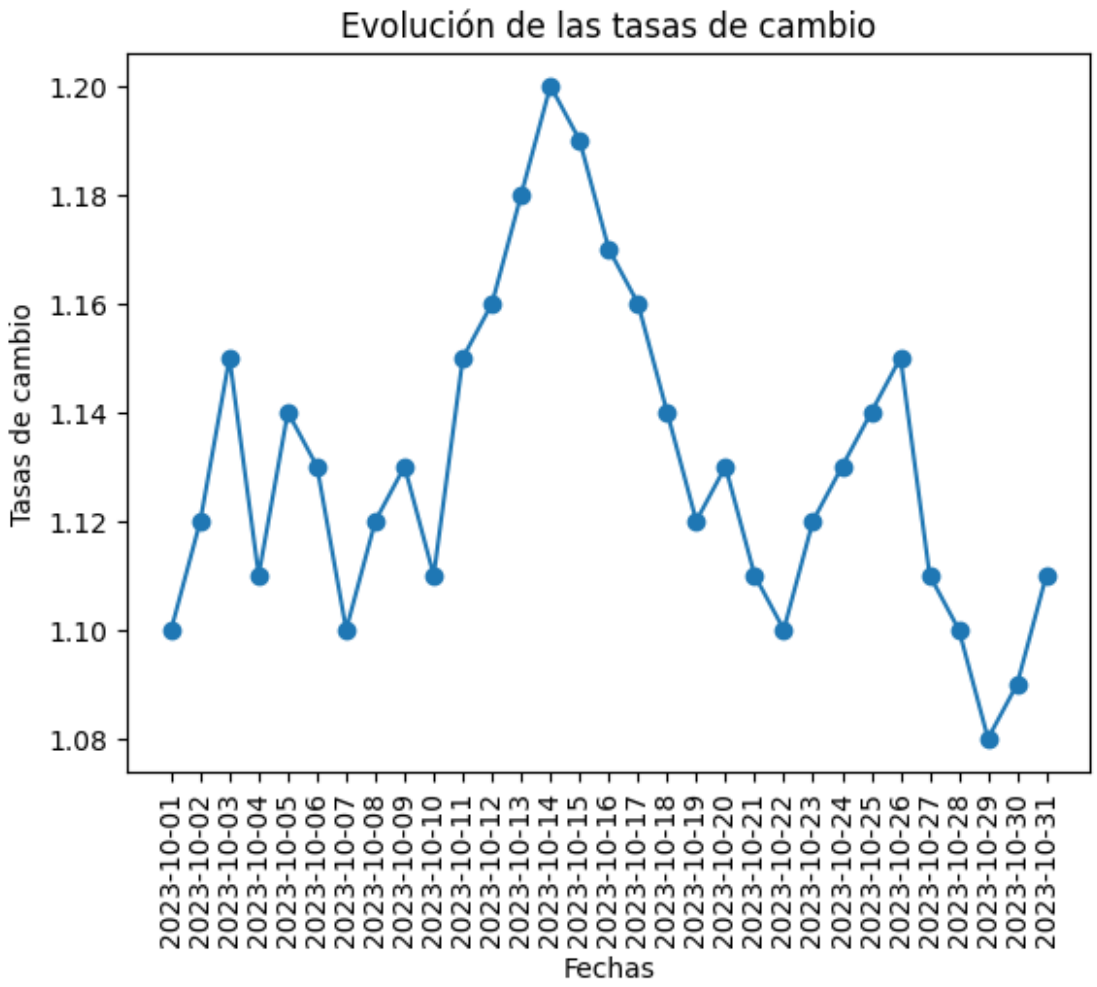
5. Determinar la volatilidad diaria de las tasas de cambio

```
volatility = np.diff(exchange_rates)
print("Volatilidad diaria:", volatility)
```

```
Volatilidad diaria: [ 0.02  0.03 -0.04  0.03 -0.01 -0.03  0.02  0.01 -
0.02  0.04  0.01  0.02
  0.02 -0.01 -0.02 -0.01 -0.02 -0.02  0.01 -0.02 -0.01  0.02  0.01
0.01
  0.01 -0.04 -0.01 -0.02  0.01  0.02]
```

6. Graficar la evolución de las tasas de cambio a lo largo del mes

```
plt.plot(dates, exchange_rates, marker='o')
plt.xticks(rotation=90)
plt.xlabel('Fechas')
plt.ylabel('Tasas de cambio')
plt.title('Evolución de las tasas de cambio')
plt.show()
```



7. Comparar el rendimiento de las tasas de cambio en la primera y última semana del mes

```
first_week_avg = np.mean(exchange_rates[:7])
last_week_avg = np.mean(exchange_rates[-7:])
print("Promedio de la primera semana:", first_week_avg)
print("Promedio de la última semana:", last_week_avg)

if last_week_avg > first_week_avg:
    print("El rendimiento ha mejorado en la última semana.")
elif last_week_avg < first_week_avg:
    print("El rendimiento ha disminuido en la última semana.")
else:
    print("El rendimiento se ha mantenido igual entre la primera y la
última semana.")
```

```
Promedio de la primera semana: 1.1214285714285714
Promedio de la última semana: 1.1114285714285714
El rendimiento ha disminuido en la última semana.
```

8. Calcular la correlación de las tasas de cambio con respecto a una variable externa

```
correlation = np.corrcoef(exchange_rates, external_variable)[0, 1]
print("Correlación entre tasas de cambio y variable externa:",
correlation)

Correlación entre tasas de cambio y variable externa: -
0.23044484958251135
```

9. Determinar el número de días en los que la tasa de cambio superó un valor específico

```
specific_value = 1.15
days_above_value = np.sum(exchange_rates > specific_value)
print("Número de días con tasas superiores a", specific_value, ":",
days_above_value)

Número de días con tasas superiores a 1.15 : 6
```

10. Crear un resumen estadístico que incluya la media, mediana y desviación estándar de las tasas de cambio

```
mean_rate = np.mean(exchange_rates)
median_rate = np.median(exchange_rates)
std_deviation = np.std(exchange_rates)

print("Media de las tasas de cambio:", mean_rate)
print("Mediana de las tasas de cambio:", median_rate)
print("Desviación estándar de las tasas de cambio:", std_deviation)

Media de las tasas de cambio: 1.1306451612903228
Mediana de las tasas de cambio: 1.13
Desviación estándar de las tasas de cambio: 0.02895329764169913
```

Business Challenge: E-commerce Customer Data Analysis

Trabajas para una empresa de comercio electrónico, y tu gerente te ha pedido analizar los datos de los clientes para obtener insights sobre el comportamiento de los clientes. Tu tarea es calcular algunas métricas clave y generar informes basados en los datos proporcionados.

Datos de Clientes

Se te proporciona un array de NumPy llamado `customer_data`, que contiene información sobre cada cliente. Cada fila en el array representa a un cliente, y las columnas contienen la siguiente información:

1. ID del Cliente
2. Total de Pedidos Realizados

3. Total Gastado (en dólares)
4. Días Desde la Última Compra
5. Suscrito al Correo Electrónico (1 para suscrito, 0 para no suscrito)

Aquí tienes un ejemplo de cómo podría verse el array `customer_data` con 30 registros:

```
import numpy as np

customer_data = np.array([
    [1, 5, 500, 10, 1],
    [2, 3, 300, 20, 1],
    [3, 2, 200, 30, 0],
    [4, 7, 700, 5, 1],
    [5, 4, 400, 15, 1],
    [6, 1, 100, 40, 0],
    [7, 8, 800, 2, 1],
    [8, 6, 600, 25, 0],
    [9, 5, 500, 50, 1],
    [10, 3, 300, 35, 1],
    [11, 0, 0, 60, 0],
    [12, 9, 900, 1, 1],
    [13, 4, 400, 12, 1],
    [14, 2, 200, 18, 0],
    [15, 10, 1000, 0, 1],
    [16, 1, 100, 29, 0],
    [17, 3, 300, 22, 1],
    [18, 5, 500, 14, 1],
    [19, 6, 600, 10, 1],
    [20, 4, 400, 30, 0],
    [21, 7, 700, 4, 1],
    [22, 8, 800, 3, 1],
    [23, 0, 0, 50, 0],
    [24, 2, 200, 45, 1],
    [25, 9, 900, 2, 1],
    [26, 1, 100, 33, 0],
    [27, 6, 600, 7, 1],
    [28, 3, 300, 17, 1],
    [29, 4, 400, 28, 0],
    [30, 5, 500, 19, 1]
])
```

Tu objetivo es crear un conjunto de funciones para analizar estos datos y generar informes para tu gerente.

Funciones a Implementar

Necesitas implementar las siguientes funciones:

1. **`average_order_value(data)`**: Calcular el valor promedio de pedido (AOV) para todos los clientes. El AOV se calcula como el total gastado dividido por el total de pedidos.

2. **`customer_lifetime_value(data)`**: Calcular el valor de vida útil del cliente

(CLV) para cada cliente. El CLV se calcula como el producto del valor promedio de pedido y el número de días desde la última compra.

1. **`high_value_customers(data, threshold)`**: Identificar a los clientes de alto valor cuyo CLV excede un umbral dado.

2. **`email_subscription_rate(data)`**: Calcular la tasa de suscripción al correo electrónico, que es el porcentaje de clientes que se han suscrito a los correos electrónicos.

3. **`inactive_customers(data, days)`**: Identificar a los clientes que no han realizado una compra en un número específico de días.

Actividades Adicionales

Además de las funciones anteriores, se te pide realizar las siguientes tareas:

1. **`most_frequent_customers(data)`**: Identificar a los clientes que han realizado la mayor cantidad de pedidos. Esto puede ayudar a centrar las estrategias de marketing en los clientes más activos.

2. **`total_revenue(data)`**: Calcular el ingreso total generado por todos los clientes durante el período analizado.

3. **`average_days_since_last_purchase(data)`**: Calcular el promedio de días desde la última compra para todos los clientes. Esto puede indicar la lealtad y satisfacción del cliente.

4. **`customer_retention_rate(data)`**: Calcular la tasa de retención de clientes, que es el porcentaje de clientes que han realizado más de un pedido en comparación con el total de clientes.

5. **`top_spenders(data, n)`**: Identificar a los `n` principales clientes que han gastado más dinero en total. Esto puede ser útil para estrategias de marketing dirigidas.

Instrucciones

Se te proporciona un array de NumPy llamado `customer_data` que contiene información del cliente. Tu tarea es implementar las funciones mencionadas anteriormente para realizar las siguientes tareas:

- Calcular el valor promedio de pedido (AOV) para todos los clientes.
- Calcular el valor de vida útil del cliente (CLV) para cada cliente.

- Identificar a los clientes de alto valor cuyo CLV excede un umbral dado. (Toma 300 como umbral)
- Calcular la tasa de suscripción al correo electrónico.
- Identificar a los clientes que no han realizado una compra en un número específico de días. (Toma 30 días)
- Identificar a los clientes que han realizado la mayor cantidad de pedidos.
- Calcular el ingreso total generado por todos los clientes.
- Calcular el promedio de días desde la última compra.
- Calcular la tasa de retención de clientes.
- Identificar a los `n` principales clientes que han gastado más dinero.

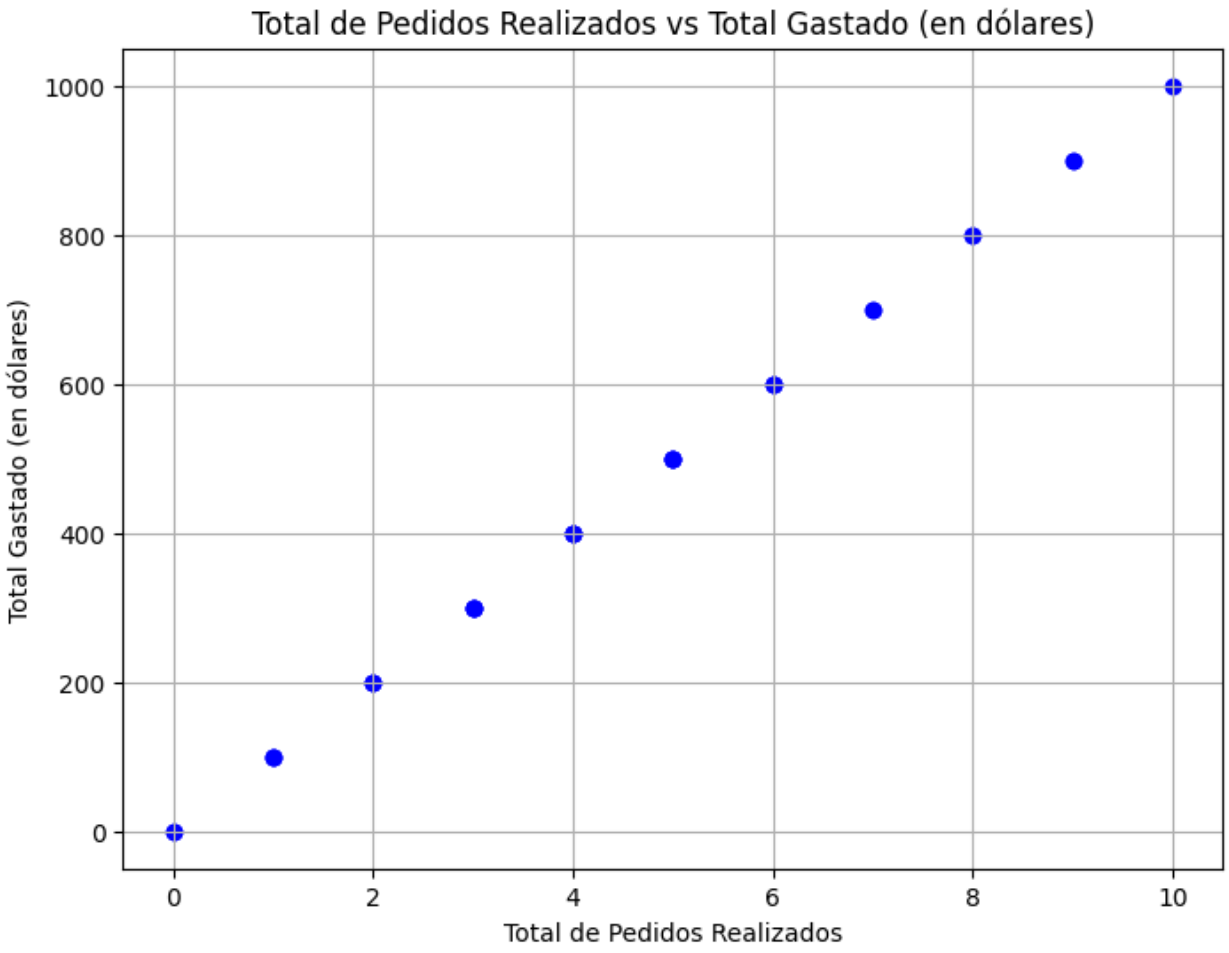
Debes probar tus funciones con los datos proporcionados y mostrar los resultados de manera clara y concisa.

```
import numpy as np

customer_data = np.array([
    [1, 5, 500, 10, 1],
    [2, 3, 300, 20, 1],
    [3, 2, 200, 30, 0],
    [4, 7, 700, 5, 1],
    [5, 4, 400, 15, 1],
    [6, 1, 100, 40, 0],
    [7, 8, 800, 2, 1],
    [8, 6, 600, 25, 0],
    [9, 5, 500, 50, 1],
    [10, 3, 300, 35, 1],
    [11, 0, 0, 60, 0],
    [12, 9, 900, 1, 1],
    [13, 4, 400, 12, 1],
    [14, 2, 200, 18, 0],
    [15, 10, 1000, 0, 1],
    [16, 1, 100, 29, 0],
    [17, 3, 300, 22, 1],
    [18, 5, 500, 14, 1],
    [19, 6, 600, 10, 1],
    [20, 4, 400, 30, 0],
    [21, 7, 700, 4, 1],
    [22, 8, 800, 3, 1],
    [23, 0, 0, 50, 0],
    [24, 2, 200, 45, 1],
    [25, 9, 900, 2, 1],
```

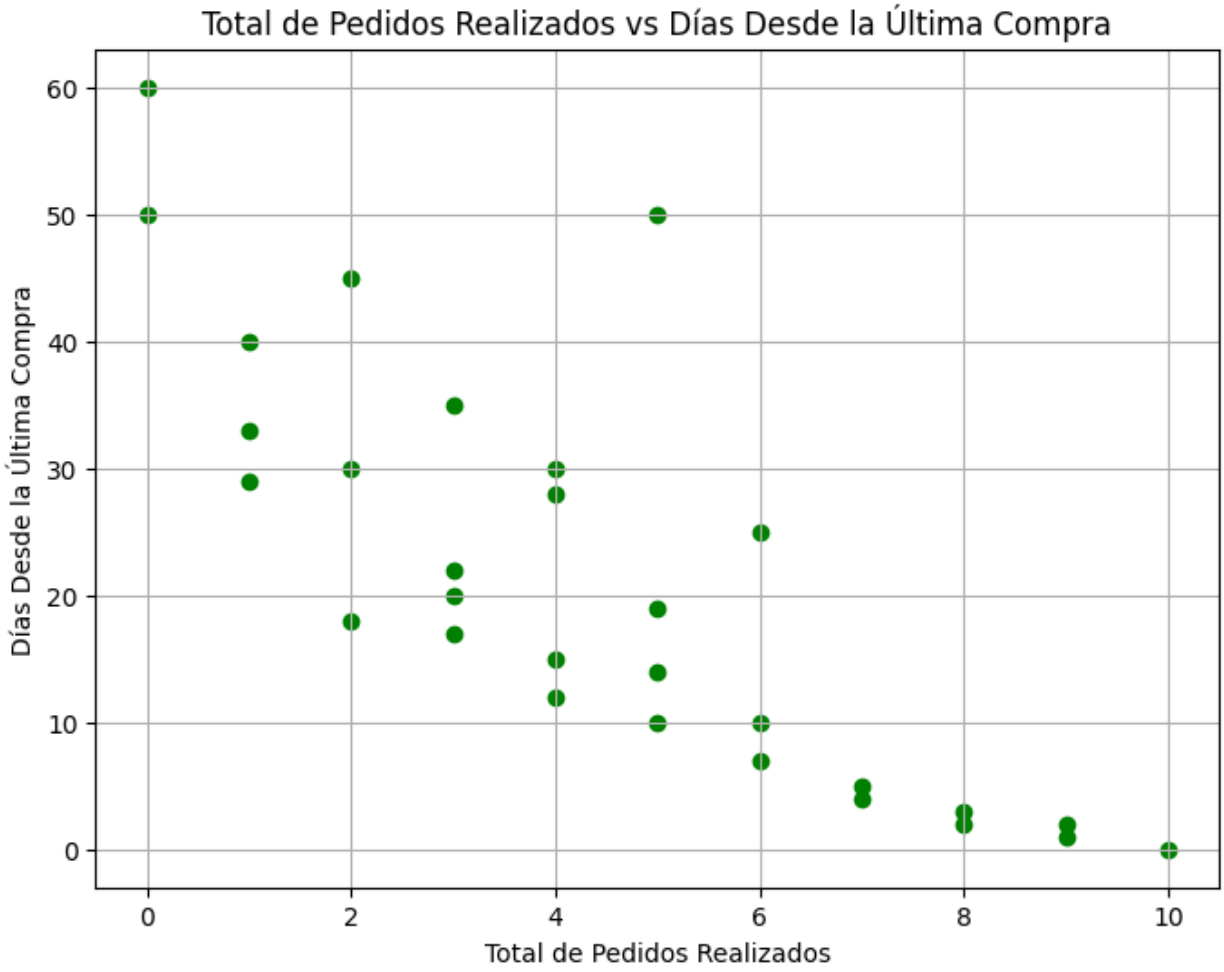
```
    [26, 1, 100, 33, 0],
    [27, 6, 600, 7, 1],
    [28, 3, 300, 17, 1],
    [29, 4, 400, 28, 0],
    [30, 5, 500, 19, 1]
])
```

0. Analizar relacion de los datos mediante dispersión

```
# 1. Gráfico de dispersión: Total de Pedidos Realizados vs Total
Gastado (en dólares)


plt.figure(figsize=(8, 6))
plt.scatter(customer_data[:,1], customer_data[:,2], color='blue')
plt.title('Total de Pedidos Realizados vs Total Gastado (en dólares)')
plt.xlabel('Total de Pedidos Realizados')
plt.ylabel('Total Gastado (en dólares)')
plt.grid(True)
plt.show()
```

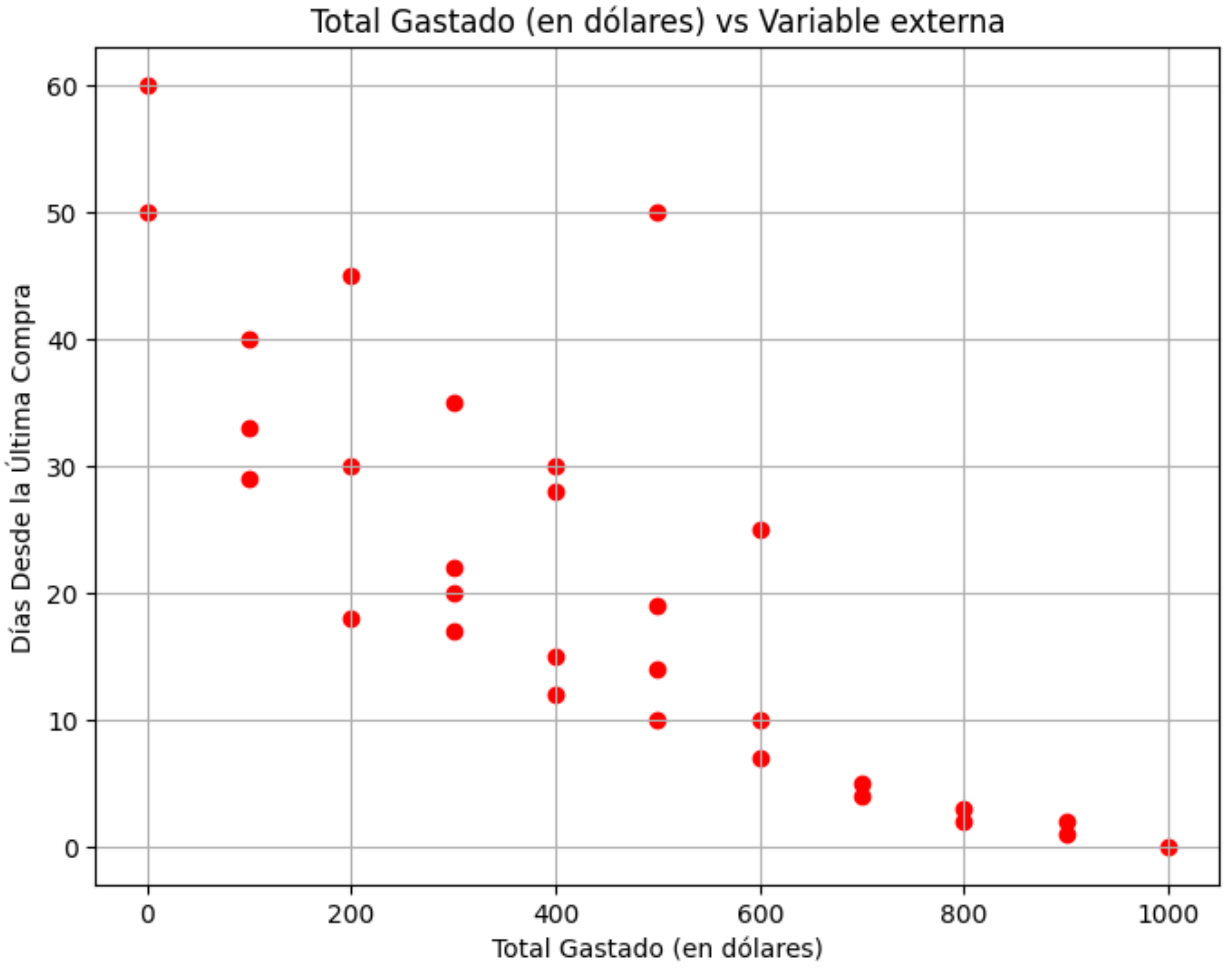


```
# 2. Gráfico de dispersión: Total de Pedidos Realizados vs Días Desde la Última Compra
plt.figure(figsize=(8, 6))
plt.scatter(customer_data[:,1], customer_data[:,3], color='green')
plt.title('Total de Pedidos Realizados vs Días Desde la Última Compra')
plt.xlabel('Total de Pedidos Realizados')
plt.ylabel('Días Desde la Última Compra')
plt.grid(True)
plt.show()
```




```
# 3. Gráfico de dispersión: Total Gastado (en dólares) vs Días Desde la Última Compra
plt.figure(figsize=(8, 6))
plt.scatter(customer_data[:,2], customer_data[:,3], color='red')
plt.title('Total Gastado (en dólares) vs Variable externa')
plt.xlabel('Total Gastado (en dólares)')
plt.ylabel('Días Desde la Última Compra')
```

```
plt.grid(True)
plt.show()
```



1. Calcular el valor promedio de pedido (AOV)

```
def average_order_value(data):
    total_orders = data[:, 1]
    total_spent = data[:, 2]
    valid_indices = total_orders > 0
    aov = np.mean(total_spent[valid_indices] /
total_orders[valid_indices])
    return aov

aov = average_order_value(customer_data)
print("Valor promedio de pedido (AOV):", aov)
```

```
Valor promedio de pedido (AOV): 100.0
```

2. Calcular el valor de vida útil del cliente (CLV)

```
def customer_lifetime_value(data):
    total_spent = data[:, 2]
    orders = data[:,1]
    indexs = orders > 0
    mean_spent = total_spent[indexs ]/orders[indexs]
    days_since_last_purchase = data[:,3]
    clv = mean_spent * days_since_last_purchase[indexs]
    return clv

clv = customer_lifetime_value(customer_data)
print("Valor de vida útil del cliente (CLV) por cliente: \n", clv)
```

```
Valor de vida útil del cliente (CLV) por cliente:
 [1000. 2000. 3000.  500. 1500. 4000.  200. 2500. 5000. 3500.  100.
1200.
 1800.    0. 2900. 2200. 1400. 1000. 3000.  400.  300. 4500.  200.
3300.
  700. 1700. 2800. 1900.]
```

3. Identificar clientes de alto valor cuyo CLV excede un umbral

```
def high_value_customers(data, threshold):
    clv = customer_lifetime_value(data)
    high_value_indices = np.where(clv > threshold)
    return data[high_value_indices]

high_value = high_value_customers(customer_data, 300)
print("Clientes de alto valor (CLV > 300): \n", high_value)
```

```
Clientes de alto valor (CLV > 300):
 [[   1    5  500   10    1]
 [   2    3  300   20    1]
 [   3    2  200   30    0]
 [   4    7  700    5    1]
 [   5    4  400   15    1]
 [   6    1  100   40    0]
 [   8    6  600   25    0]
 [   9    5  500   50    1]
 [  10    3  300   35    1]
 [  12    9  900    1    1]
 [  13    4  400   12    1]
 [  15   10 1000    0    1]
 [  16    1  100   29    0]
 [  17    3  300   22    1]
 [  18    5  500   14    1]
 [  19    6  600   10    1]
 [  20    4  400   30    0]
 [  22    8  800    3    1]
 [  24    2  200   45    1]
```

```
 [  25    9  900    2    1]
 [  26    1  100   33    0]
 [  27    6  600    7    1]
 [  28    3  300   17    1]]
```

4. Calcular la tasa de suscripción al correo electrónico

```
def email_subscription_rate(data):
    subscriptions = data[:, 4]
    subscription_rate = np.mean(subscriptions) * 100
    return subscription_rate

subscription_rate = email_subscription_rate(customer_data)
print("Tasa de suscripción al correo electrónico (%):",
subscription_rate)
```

```
Tasa de suscripción al correo electrónico (%): 66.66666666666666
```

5. Identificar clientes inactivos por un número específico de días

```
def inactive_customers(data, days):
    days_since_last_purchase = data[:, 3]
    inactive_indices = np.where(days_since_last_purchase > days)
    return data[inactive_indices]

inactive = inactive_customers(customer_data, 30)
print("Clientes inactivos (más de 30 días sin comprar): \n", inactive)
```

```
Clientes inactivos (más de 30 días sin comprar):
 [[  6   1 100  40   0]
 [  9   5 500  50   1]
 [ 10   3 300  35   1]
 [ 11   0   0  60   0]
 [ 23   0   0  50   0]
 [ 24   2 200  45   1]
 [ 26   1 100  33   0]]
```

6. Identificar a los clientes que han realizado la mayor cantidad de pedidos

```
def most_frequent_customers(data):
    total_orders = data[:, 1]
    max_orders = np.max(total_orders)
    frequent_customers_indices = np.where(total_orders == max_orders)
    return data[frequent_customers_indices]

frequent_customers = most_frequent_customers(customer_data)
print("Clientes más frecuentes:", frequent_customers)
```

```
Clientes más frecuentes: [[  15   10 1000    0    1]]
```

7. Calcular el ingreso total generado por todos los clientes

```
def total_revenue(data):
    total_spent = data[:, 2]
    return np.sum(total_spent)

revenue = total_revenue(customer_data)
print("Ingreso total generado:", revenue)
```

```
Ingreso total generado: 13300
```

8. Calcular el promedio de días desde la última compra

```
def average_days_since_last_purchase(data):
    days_since_last_purchase = data[:, 3]
    return np.mean(days_since_last_purchase)

avg_days = average_days_since_last_purchase(customer_data)
print("Promedio de días desde la última compra:", avg_days)
```

```
Promedio de días desde la última compra: 21.2
```

9. Calcular la tasa de retención de clientes

```
def customer_retention_rate(data):
    total_orders = data[:, 1]
    retained_customers = np.sum(total_orders > 1)
    retention_rate = (retained_customers / len(data)) * 100
    return retention_rate

retention_rate = customer_retention_rate(customer_data)
print("Tasa de retención de clientes (%):", retention_rate)
```

```
Tasa de retención de clientes (%): 83.33333333333334
```

10. Identificar los n principales clientes que han gastado más dinero

```
def top_spenders(data, n):
    total_spent = data[:, 2]
    top_spenders_indices = np.argsort(total_spent)[-n:][::-1]
    return data[top_spenders_indices]

top_5_spenders = top_spenders(customer_data, 5)
print("Top 5 clientes que más han gastado: \n", top_5_spenders)
```

```
Top 5 clientes que más han gastado:
 [[  15   10 1000    0    1]
 [  25    9  900    2    1]
 [  12    9  900    1    1]
 [   7    8  800    2    1]
 [  22    8  800    3    1]]
```