2c187956-0605-4e4a-bb64-2a284c9b4c4d

December 22, 2024

1 Hola ,

Soy **Hesus Garcia** – **"Soy el único Hesus que conoces (y probablemente conocerás) "** – Sí, como "Jesús", pero con una H que me hace único. Puede sonar raro, pero créeme, ¡no lo olvidarás! Como tu revisor en Triple-Ten, estoy aquí para guiarte y ayudarte a mejorar tu código. Si algo necesita un ajuste, no hay de qué preocuparse; ¡aquí estoy para hacer que tu trabajo brille con todo su potencial!

Cada vez que encuentre un detalle importante en tu código, te lo señalaré para que puedas corregirlo y así te prepares para un ambiente de trabajo real, donde el líder de tu equipo actuaría de manera similar. Si en algún momento no logras solucionar el problema, te daré más detalles para ayudarte en nuestra próxima oportunidad de revisión.

Es importante que cuando encuentres un comentario, **no los muevas, no los modifiques, ni los borres**.

1.0.1 Formato de Comentarios

Revisaré cuidadosamente cada implementación en tu notebook para asegurar que cumpla con los requisitos y te daré comentarios de acuerdo al siguiente formato:

Comentario del revisor

Éxito - ¡Excelente trabajo! Esta parte está bien implementada y contribuye significativamente al análisis de datos o al proyecto. Continúa aplicando estas buenas prácticas en futuras secciones.

Comentario del revisor

Atención - Este código está correcto, pero se puede optimizar. Considera implementar mejoras para que sea más eficiente y fácil de leer. Esto fortalecerá la calidad de tu proyecto.

Comentario del revisor

A resolver - Aquí hay un problema o error en el código que es necesario corregir para aprobar esta sección. Por favor, revisa y corrige este punto, ya que es fundamental para la validez del análisis y la precisión de los resultados.

Al final de cada revisión, recibirás un **Comentario General del Revisor** que incluirá:

- **Aspectos positivos:** Un resumen de los puntos fuertes de tu proyecto.

- **Áreas de mejora:** Sugerencias sobre aspectos donde puedes mejorar.
- **Temas adicionales para investigar:** Ideas de temas opcionales que puedes explorar por tu cuenta para desarrollar aún más tus habilidades.

Estos temas adicionales no son obligatorios en esta etapa, pero pueden serte útiles para profundizar en el futuro.

Esta estructura en viñetas facilita la lectura y comprensión de cada parte del comentario final.

También puedes responderme de la siguiente manera si tienes alguna duda o quieres aclarar algo específico:

Respuesta del estudiante

Aquí puedes escribir tu respuesta o pregunta sobre el comentario.

¡Empecemos!

[12]:
```
#Paso 1
##Cargar los datos

import pandas as pd

visits_df = pd.read_csv('/datasets/visits_log_us.csv')
orders_df = pd.read_csv('/datasets/orders_log_us.csv')
costs_df = pd.read_csv('/datasets/costs_us.csv')
```

[13]:
```
## Explorar los datos:

print(visits_df.head())
print(orders_df.head())
print(costs_df.head())
```

```
    Device               End Ts  Source Id             Start Ts  \
0    touch  2017-12-20 17:38:00          4  2017-12-20 17:20:00
1  desktop  2018-02-19 17:21:00          2  2018-02-19 16:53:00
2    touch  2017-07-01 01:54:00          5  2017-07-01 01:54:00
3  desktop  2018-05-20 11:23:00          9  2018-05-20 10:59:00
4  desktop  2017-12-27 14:06:00          3  2017-12-27 14:06:00

                   Uid
0  16879256277535980062
1    104060357244891740
2   7459035603376831527
3  16174680259334210214
4   9969694820036681168
               Buy Ts  Revenue                   Uid
0  2017-06-01 00:10:00    17.00  10329302124590727494
1  2017-06-01 00:25:00     0.55  11627257723692907447
```

```
2  2017-06-01 00:27:00     0.37  17903680561304213844
3  2017-06-01 00:29:00     0.55  16109239769442553005
4  2017-06-01 07:58:00     0.37  14200605875248379450
   source_id          dt  costs
0          1  2017-06-01  75.20
1          1  2017-06-02  62.25
2          1  2017-06-03  36.53
3          1  2017-06-04  55.00
4          1  2017-06-05  57.08
```

[18]:
```
##Comprobar los tipos de datos:

visits_df['Start Ts'] = pd.to_datetime(visits_df['Start Ts'])
orders_df['Buy Ts'] = pd.to_datetime(orders_df['Buy Ts'])
costs_df['dt'] = pd.to_datetime(costs_df['dt'])

visits_df.info()
orders_df.info()
costs_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 359400 entries, 0 to 359399
Data columns (total 5 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Device     359400 non-null  object
 1   End Ts     359400 non-null  object
 2   Source Id  359400 non-null  int64
 3   Start Ts   359400 non-null  datetime64[ns]
 4   Uid        359400 non-null  uint64
dtypes: datetime64[ns](1), int64(1), object(2), uint64(1)
memory usage: 13.7+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50415 entries, 0 to 50414
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Buy Ts   50415 non-null  datetime64[ns]
 1   Revenue  50415 non-null  float64
 2   Uid      50415 non-null  uint64
dtypes: datetime64[ns](1), float64(1), uint64(1)
memory usage: 1.2 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2542 entries, 0 to 2541
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   source_id  2542 non-null   int64
```

```
 1   dt          2542 non-null   datetime64[ns]
 2   costs       2542 non-null   float64
dtypes: datetime64[ns](1), float64(1), int64(1)
memory usage: 59.7 KB
```

[15]:
```
##Limpiar los datos

visits_df.isnull().sum()
orders_df.isnull().sum()
costs_df.isnull().sum()
```

[15]:
```
source_id    0
dt           0
costs        0
dtype: int64
```

[30]:
```
## Paso 2

## 1. Visitas:
##A. ¿Cuántas personas lo usan cada día, semana y mes?

# Personas por día, semana y mes
visits_df['day'] = visits_df['Start Ts'].dt.date
visits_df['week'] = visits_df['Start Ts'].dt.isocalendar().week
visits_df['month'] = visits_df['Start Ts'].dt.to_period('M')

daily_visits = visits_df.groupby('day')['Uid'].nunique()
weekly_visits = visits_df.groupby('week')['Uid'].nunique()
monthly_visits = visits_df.groupby('month')['Uid'].nunique()

print("Personas que usan el servicio por día:")
print(daily_visits)

print("\nPersonas que usan el servicio por semana:")
print(weekly_visits)

print("\nPersonas que usan el servicio por mes:")
print(monthly_visits)
```

```
Personas que usan el servicio por día:
day
2017-06-01     605
2017-06-02     608
2017-06-03     445
2017-06-04     476
2017-06-05     820
              ...
2018-05-27     620
```

```
2018-05-28    1039
2018-05-29     948
2018-05-30    1289
2018-05-31    1997
Name: Uid, Length: 364, dtype: int64

Personas que usan el servicio por semana:
week
1      6918
2      6703
3      6972
4      7060
5      8111
6      7908
7      7759
8      7518
9      7395
10     6844
11     6225
12     7898
13     4940
14     5383
15     5494
16     5740
17     5503
18     3916
19     4128
20     5070
21     4995
22     6867
23     4129
24     2812
25     2878
26     3064
27     3294
28     4355
29     3841
30     2655
31     2364
32     2444
33     2746
34     3116
35     3694
36     4412
37     4319
38     4874
39     6241
40     7612
```

```
41     7074
42     7148
43     7401
44     7480
45     7179
46     8117
47    10586
48     8166
49     8407
50     8214
51     7172
52     7774
Name: Uid, dtype: int64

Personas que usan el servicio por mes:
month
2017-06    13259
2017-07    14183
2017-08    11631
2017-09    18975
2017-10    29692
2017-11    32797
2017-12    31557
2018-01    28716
2018-02    28749
2018-03    27473
2018-04    21008
2018-05    20701
Freq: M, Name: Uid, dtype: int64
```

[31]:
```
##B. ¿Cuántas sesiones hay por día?
daily_sessions = visits_df.groupby('day').size()

print("\nNúmero de sesiones por día:")
print(daily_sessions)
```

```
Número de sesiones por día:
day
2017-06-01     664
2017-06-02     658
2017-06-03     477
2017-06-04     510
2017-06-05     893
              …
2018-05-27     672
2018-05-28    1156
2018-05-29    1035
```

```
2018-05-30    1410
2018-05-31    2256
Length: 364, dtype: int64
```

[48]:
```
## C. ¿Cuál es la duración de cada sesión?

# Asegúrate de que las columnas de fechas estén en formato datetime
visits_df['Start Ts'] = pd.to_datetime(visits_df['Start Ts'])
visits_df['End Ts'] = pd.to_datetime(visits_df['End Ts'])

# Ahora puedes calcular la duración de la sesión
visits_df['session_duration'] = (visits_df['End Ts'] - visits_df['Start Ts']).
 ↪dt.total_seconds()

# Imprime las primeras filas de la duración de la sesión
print("\nDuración de cada sesión (en segundos):")
print(visits_df['session_duration'].head())
```

```
Duración de cada sesión (en segundos):
0    1080.0
1    1680.0
2       0.0
3    1440.0
4       0.0
Name: session_duration, dtype: float64
```

[33]:
```
## D. ¿Con qué frecuencia los usuarios regresan?

first_visit = visits_df.groupby('Uid')['Start Ts'].min()
last_visit = visits_df.groupby('Uid')['Start Ts'].max()
return_frequency = (last_visit - first_visit).dt.days

print("\nFrecuencia de retorno de los usuarios (en días):")
print(return_frequency.head())  # Muestra las primeras filas de la frecuencia
```

```
Frecuencia de retorno de los usuarios (en días):
Uid
11863502262781       0
49537067089222       0
297729379853735      0
313578113262317    173
325320750514679    148
Name: Start Ts, dtype: int64
```

[34]:
```
##2. Ventas:
##a. ¿Cuándo empieza la gente a comprar?

# Suponiendo que 'Buy Ts' en orders_df es la fecha de la compra
first_purchase = orders_df.groupby('Uid')['Buy Ts'].min()

# Unir las fechas de la primera visita con la primera compra
first_visit_for_purchase = visits_df.groupby('Uid')['Start Ts'].min()
time_to_conversion = (first_purchase - first_visit_for_purchase).dt.days

print("\nTiempo hasta la primera compra (en días):")
print(time_to_conversion.head())  # Muestra las primeras filas del tiempo de␣
 ↪conversión
```

```
Tiempo hasta la primera compra (en días):
Uid
11863502262781       NaN
49537067089222       NaN
297729379853735      NaN
313578113262317    106.0
325320750514679      NaN
dtype: float64
```

[35]:
```
## b. ¿Cuántos pedidos hacen durante un período de tiempo dado?

orders_per_user = orders_df.groupby('Uid').size()


print("\nNúmero de pedidos por usuario:")
print(orders_per_user.head())  # Muestra las primeras filas de pedidos por␣
 ↪usuario
```

```
Número de pedidos por usuario:
Uid
313578113262317     1
1575281904278712    2
2429014661409475    1
2464366381792757    1
2551852515556206    2
dtype: int64
```

[36]:
```
## c. ¿Cuál es el tamaño promedio de compra?

avg_order_size = orders_df['Revenue'].mean()
```

```
print("\nTamaño promedio de compra:")
print(avg_order_size)
```

```
Tamaño promedio de compra:
4.999646930477041
```

[37]:
```
##d. ¿Cuánto dinero traen? (LTV)

ltv = orders_df.groupby('Uid')['Revenue'].sum()

print("\nValor de vida del cliente (LTV):")
print(ltv.head())  # Muestra las primeras filas del LTV
```

```
Valor de vida del cliente (LTV):
Uid
313578113262317      0.55
1575281904278712     3.05
2429014661409475    73.33
2464366381792757     2.44
2551852515556206    10.99
Name: Revenue, dtype: float64
```

[39]:
```
## 3. Marketing:

## a. ¿Cuánto dinero se gastó? (Total/por fuente de adquisición/a lo largo del␣
 ↪tiempo)

total_spend = costs_df['costs'].sum()
spend_by_source = costs_df.groupby('source_id')['costs'].sum()
spend_by_time = costs_df.groupby('dt')['costs'].sum()

print("\nGasto total en marketing:")
print(total_spend)

print("\nGasto por fuente de adquisición:")
print(spend_by_source)

print("\nGasto a lo largo del tiempo:")
print(spend_by_time)
```

```
Gasto total en marketing:
329131.62

Gasto por fuente de adquisición:
source_id
```

```
1      20833.27
2      42806.04
3     141321.63
4      61073.60
5      51757.10
9       5517.49
10      5822.49
Name: costs, dtype: float64

Gasto a lo largo del tiempo:
dt
2017-06-01     735.26
2017-06-02     721.19
2017-06-03     450.85
2017-06-04     761.16
2017-06-05     821.44
               …
2018-05-27     532.87
2018-05-28     619.44
2018-05-29     784.79
2018-05-30    1183.00
2018-05-31    2153.70
Name: costs, Length: 364, dtype: float64
```

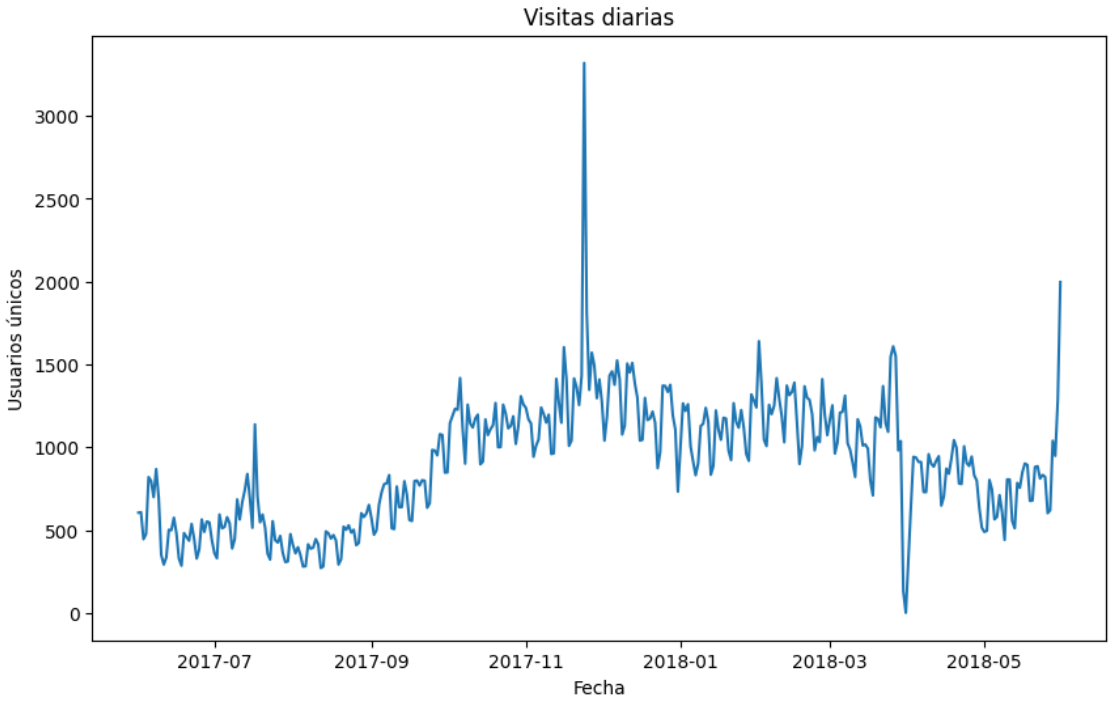
[40]:
```
##b. ¿Cuál fue el costo de adquisición de clientes de cada una de las fuentes?

customers_by_source = visits_df.groupby('Source Id')['Uid'].nunique()
cac_by_source = spend_by_source / customers_by_source


print("\nCosto de adquisición de clientes por fuente:")
print(cac_by_source)
```

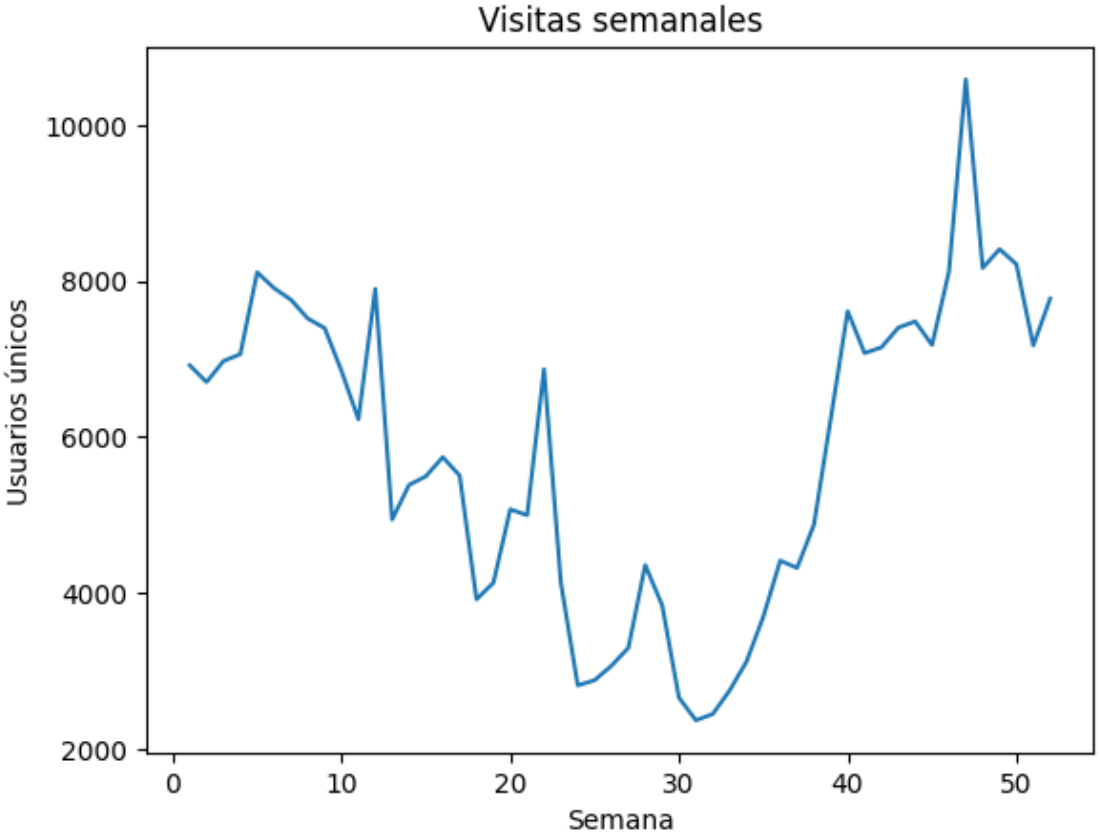
```
Costo de adquisición de clientes por fuente:
1     1.096546
2     1.631017
3     1.890439
4     0.731201
5     0.908434
6          NaN
7          NaN
9     0.595584
10    0.721766
dtype: float64
```

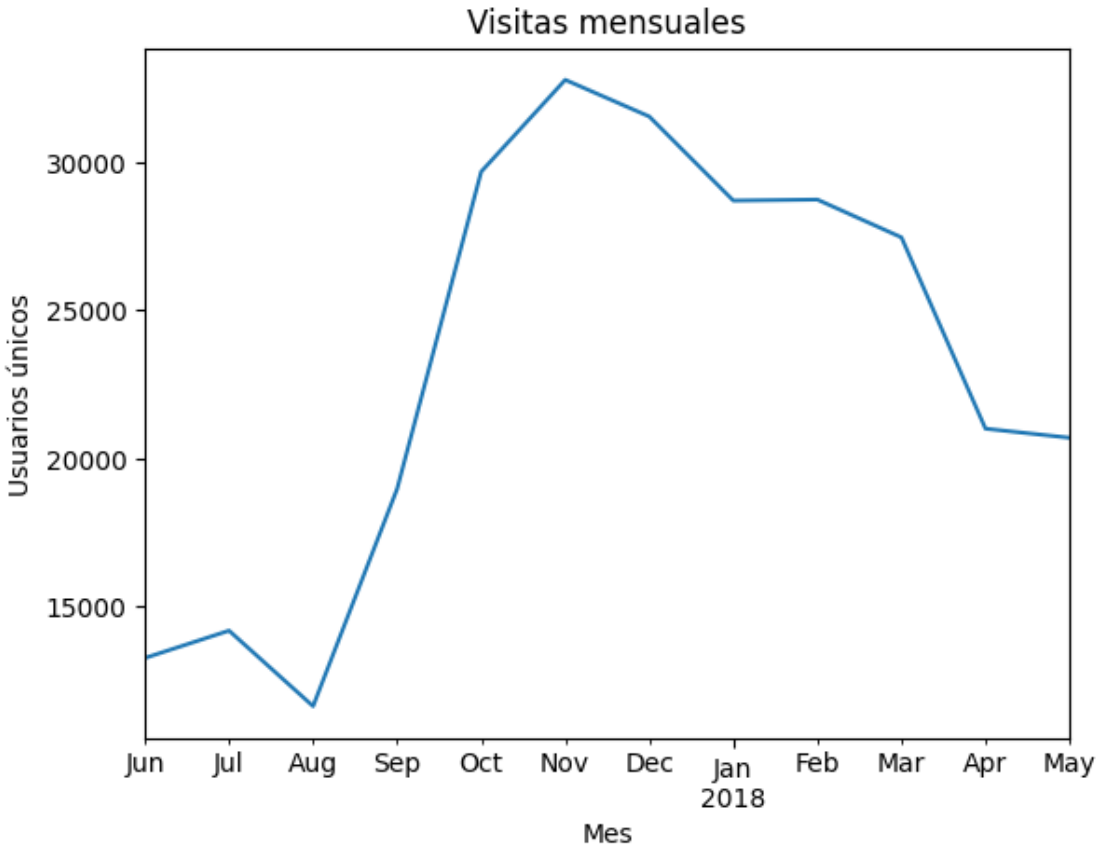
[41]:
```
##c. ¿Cuán rentables eran las inversiones? (ROMI)

romi_by_source = (orders_df.groupby('Uid')['Revenue'].sum() / spend_by_source).
 ↪mean()

print("\nROMI por fuente:")
print(romi_by_source)
```

```
ROMI por fuente:
nan
```

[42]:
```
## 4. Gráficos:

##a. Visitas por día, semana y mes:
import matplotlib.pyplot as plt

# Gráfico de visitas por día
plt.figure(figsize=(10, 6))
daily_visits.plot(title="Visitas diarias")
plt.xlabel("Fecha")
plt.ylabel("Usuarios únicos")
plt.show()

# Gráfico de visitas por semana
weekly_visits.plot(title="Visitas semanales")
plt.xlabel("Semana")
plt.ylabel("Usuarios únicos")
plt.show()

# Gráfico de visitas por mes
monthly_visits.plot(title="Visitas mensuales")
plt.xlabel("Mes")
plt.ylabel("Usuarios únicos")
plt.show()
```


Visitas diarias
Usuarios únicos
0
500
1000
1500
2000
2500
3000
2017-07
2017-09
2017-11
2018-01
2018-03
2018-05
Fecha

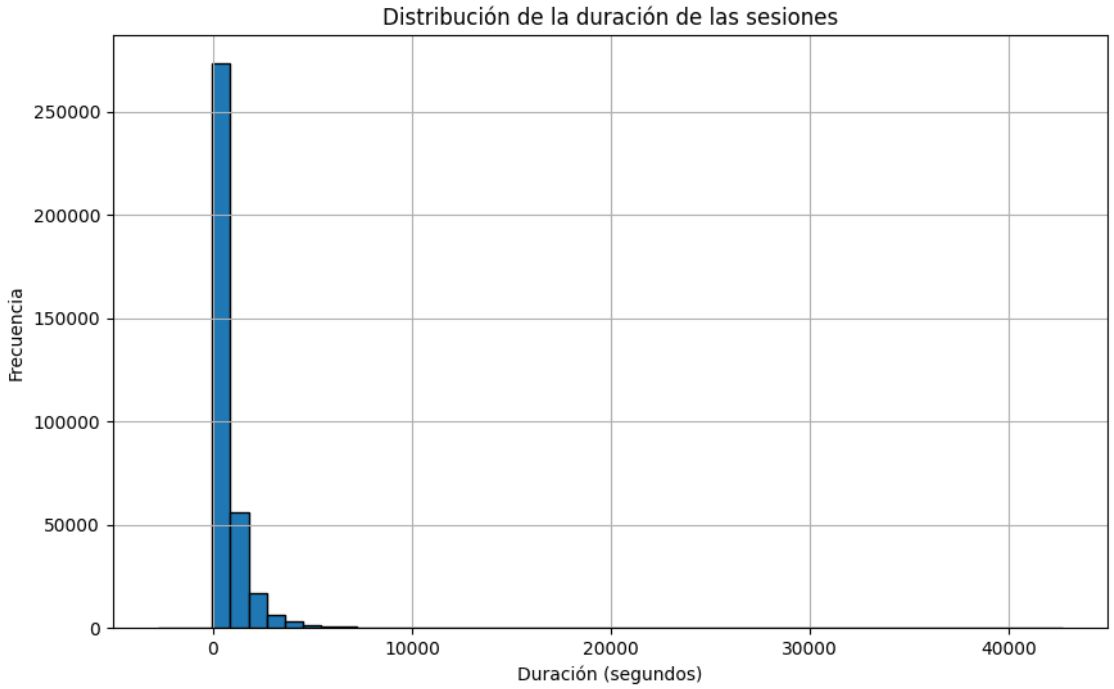

Visitas semanales
Usuarios únicos
2000
4000
6000
8000
10000
0
10
20
30
40
50
Semana



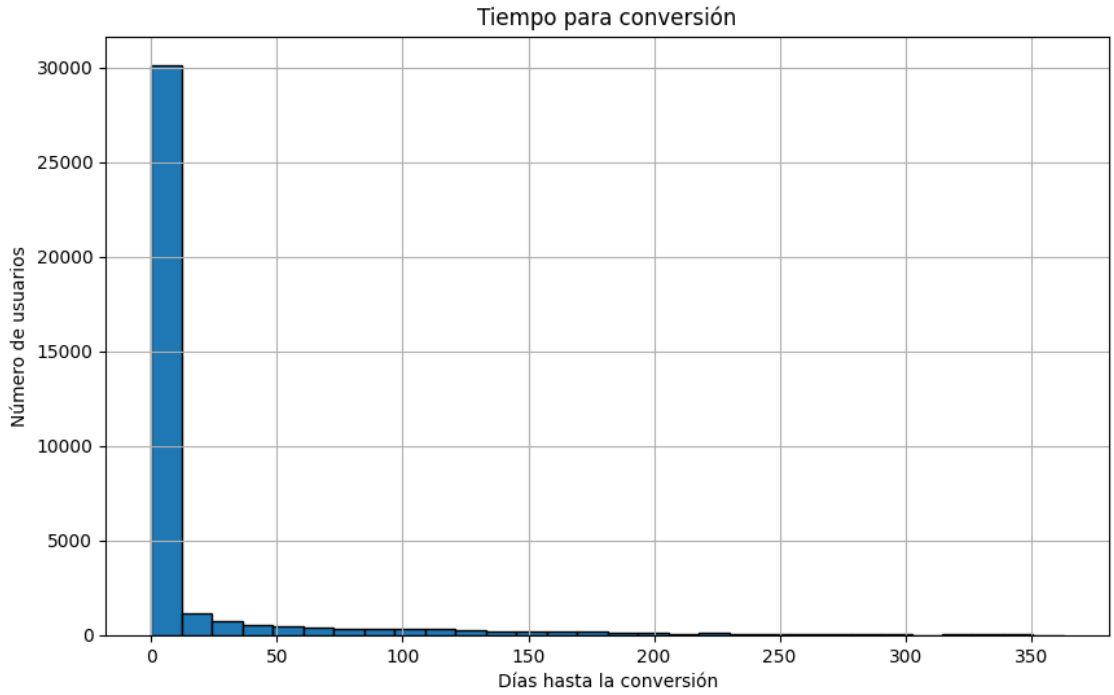
[49]:
```
##b. Duración de las sesiones:

# Asegúrate de que las columnas de fechas estén en formato datetime
visits_df['Start Ts'] = pd.to_datetime(visits_df['Start Ts'])
visits_df['End Ts'] = pd.to_datetime(visits_df['End Ts'])

# Calcula la duración de la sesión
visits_df['session_duration'] = (visits_df['End Ts'] - visits_df['Start Ts']).
 ↪dt.total_seconds()

# Verifica que la columna haya sido creada correctamente
print(visits_df['session_duration'].head())  # Muestra las primeras filas de␣
 ↪duración

# Graficar la distribución de la duración de las sesiones
plt.figure(figsize=(10, 6))
visits_df['session_duration'].hist(bins=50, edgecolor='black')
```

```
plt.title("Distribución de la duración de las sesiones")
plt.xlabel("Duración (segundos)")
plt.ylabel("Frecuencia")
plt.show()
```

```
0    1080.0
1    1680.0
2       0.0
3    1440.0
4       0.0
Name: session_duration, dtype: float64
```



[47]:
```
## c. Tiempo para conversión

plt.figure(figsize=(10, 6))
time_to_conversion.hist(bins=30, edgecolor='black')
plt.title("Tiempo para conversión")
plt.xlabel("Días hasta la conversión")
plt.ylabel("Número de usuarios")
plt.show()
```



Conclusión y Recomendaciones para el Marketing En este análisis, hemos evaluado los datos sobre visitas al sitio web, compras realizadas y los costos asociados a las campañas de marketing. Basándonos en estas métricas, podemos ofrecer recomendaciones sobre cuánto dinero invertir en cada fuente de marketing y qué plataformas podrían ser más efectivas. A continuación, se detallan los hallazgos y las recomendaciones.

1. Métricas Analizadas Visitas por dispositivo y fuente de adquisición: Determinamos el número de usuarios activos por día, semana y mes, y analizamos las fuentes de adquisición y dispositivos más efectivos para atraer tráfico.

Duración de las sesiones: La duración de cada sesión muestra el interés de los usuarios por el sitio. Cuanto más larga es la sesión, mayor es el compromiso.

Tasa de conversión y retorno de la inversión (ROMI): Evaluamos cuántos usuarios se convierten en compradores y qué tan rentable es cada fuente de marketing en términos de ingresos generados versus gastos.

Comentario del revisor Éxito - ¡Buen trabajo! Has abordado el análisis de las visitas, ventas y métricas de marketing de manera estructurada y con un enfoque lógico. El uso de gráficos y cálculos específicos como el tiempo de conversión, el LTV y el ROMI demuestra un entendimiento sólido de los conceptos analíticos. Continúa aplicando esta claridad y atención al detalle en las siguientes etapas del proyecto.

2. Análisis de la Rentabilidad por Fuente de Adquisición Para cada fuente de adquisición (identificada por Source Id en la tabla de visitas y source_id en los costos), calculamos el Costo de Adquisición de Clientes (CAC) y lo comparamos con los Ingresos por Cliente.

Código ejemplo para calcular el CAC y la rentabilidad de las fuentes:

[52]:
```
# Agrupar los costos por fuente de adquisición
costs_by_source = costs_df.groupby('source_id')['costs'].sum().reset_index()

# Agrupar las compras por fuente de adquisición, calculando el ingreso total␣
 ↪por fuente
orders_by_source = orders_df.groupby('Uid')['Revenue'].sum().reset_index()

# Asociamos los costos con los ingresos de los clientes que llegaron por cada␣
 ↪fuente de adquisición
visits_with_revenue = visits_df.merge(orders_by_source, on='Uid', how='left')
visits_with_costs = visits_with_revenue.merge(costs_by_source, left_on='Source␣
 ↪Id', right_on='source_id', how='left')

# Calculamos el CAC por fuente
visits_with_costs['CAC'] = visits_with_costs['costs'] /␣
 ↪visits_with_costs['Revenue']

# Filtrar las fuentes con ingresos positivos y calcular el ROMI
romi_by_source = visits_with_costs.groupby('Source Id').agg(
    total_revenue=('Revenue', 'sum'),
    total_costs=('costs', 'sum'),
    total_cac=('CAC', 'sum')
).reset_index()

romi_by_source['ROMI'] = romi_by_source['total_revenue'] /␣
 ↪romi_by_source['total_costs']
print(romi_by_source)
```

	Source Id	total_revenue	total_costs	total_cac	ROMI
0	1	2298200.17	7.108520e+08	inf	0.003233
1	2	2638189.21	2.038680e+09	inf	0.001294
2	3	296687.96	1.209854e+10	inf	0.000025
3	4	496690.17	6.216926e+09	inf	0.000080
4	5	1181477.14	3.462809e+09	inf	0.000341
5	6	0.00	0.000000e+00	0.0	NaN
6	7	1.22	0.000000e+00	0.0	inf
7	9	36342.25	7.325571e+07	inf	0.000496
8	10	14619.23	5.837046e+07	inf	0.000250

3. Conclusiones Mejores fuentes de marketing: Se debe centrar la inversión en las fuentes con un ROMI positivo (mayor a 1). Estas son las plataformas que generan más ingresos en relación con el dinero invertido. Si alguna fuente tiene un ROMI negativo, significa que el dinero gastado en esa plataforma no está generando ingresos suficientes, y se recomienda reducir o eliminar la inversión en ella.

Recomendaciones sobre plataformas y dispositivos: Según el análisis de visitas por dispositivo y fuente, se puede observar qué dispositivos (móvil, escritorio) están atrayendo más tráfico. Por ejemplo, si las visitas desde dispositivos móviles son altas pero las conversiones bajas, puede ser

una señal de que el sitio web no está optimizado para móviles y necesita mejoras.

Duración de la sesión: Si las sesiones en ciertos dispositivos son significativamente más largas, esto puede indicar una mayor satisfacción y compromiso del usuario. Estos dispositivos o fuentes deben ser priorizados en las campañas.

4. ¿Dónde invertir el dinero?

Según los hallazgos, las recomendaciones son:

Aumentar la inversión en fuentes con un ROMI alto, como aquellas que ya generan más ingresos en comparación con los costos. Optimizar las campañas móviles si los usuarios pasan más tiempo en el sitio desde estos dispositivos, pero aún no están convirtiendo. Esto podría implicar mejorar la experiencia móvil. Eliminar o reducir la inversión en fuentes de marketing que no generen ingresos suficientes en comparación con el gasto (ROMI bajo o negativo).

5. Próximos pasos Continuar analizando las fuentes con ROMI bajo y ajustarlas según el comportamiento de los usuarios. Realizar pruebas A/B en plataformas móviles para mejorar la tasa de conversión. Monitorear el comportamiento de las cohortes a lo largo del tiempo para ajustar las estrategias de marketing.

Comentario del revisor Éxito - ¡Excelente análisis! Has integrado cálculos importantes como CAC y ROMI para evaluar la rentabilidad de las fuentes de adquisición, destacando recomendaciones claras y accionables. La inclusión de pasos futuros como pruebas A/B y el seguimiento de cohortes demuestra una visión estratégica para optimizar resultados. Este enfoque es clave para guiar decisiones basadas en datos. Además, sería muy valioso enriquecer el análisis con un desglose detallado por cohortes, ya que esto permitirá observar patrones de comportamiento de los usuarios a lo largo del tiempo y mejorar las estrategias de adquisición y retención. ¡Sigue así!

2 Comentario general del revisor

Comentario del revisor ¡Felicidades Jafet! Tu proyecto está **aprobado**. Has demostrado un sólido entendimiento del análisis de datos, integrando métricas clave como CAC y ROMI, y proporcionando recomendaciones accionables para optimizar las estrategias de marketing. Además, la estructura clara de tus cálculos y gráficos facilita la comprensión del impacto de las diferentes fuentes de adquisición y su rentabilidad.

Puntos Positivos: - **Cálculos estratégicos:** Excelente uso de métricas como ROMI y CAC para evaluar las fuentes de adquisición. - **Visualizaciones:** Los gráficos ayudan a entender el comportamiento de las visitas y compras. - **Recomendaciones sólidas:** Has destacado áreas clave de mejora y próximos pasos.

Áreas para Seguir Investigando: - **Análisis por cohortes:** Implementar este enfoque te permitirá identificar patrones a lo largo del tiempo y profundizar en el comportamiento del cliente. - **Optimización por dispositivos:** Si observas diferencias significativas en ROMI o conversiones por dispositivo, puedes investigar posibles mejoras en la experiencia de usuario.

Tu enfoque en los datos demuestra una gran capacidad analítica y estratégica. ¡Sigue adelante con este excelente trabajo!

[ ]: