bf2511dd-3b02-44db-ab4a-27df4aa26269

April 21, 2025

0.1 Precision Budget Allocation /

[1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

[2]:
```
visits = pd.read_csv('/datasets/visits_log_us.csv')
orders = pd.read_csv('/datasets/orders_log_us.csv')
costs = pd.read_csv('/datasets/costs_us.csv')
```

[3]:
```
visits.info()
print()
visits.head()
```

```
<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  object
 4   Uid        359400 non-null  uint64
dtypes: int64(1), object(3), uint64(1)
memory usage: 13.7+ MB
```

[3]:

	Device	End Ts	Source Id	Start Ts	Uid
0	touch	2017-12-20 17:38:00	4	2017-12-20 17:20:00	16879256277535980062
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	

```
1    104060357244891740
2   7459035603376831527
3  16174680259334210214
4    9969694820036681168
```

[4]:
```
visits['Start Ts'] = pd.to_datetime(visits['Start Ts'])
visits['End Ts'] = pd.to_datetime(visits['End Ts'])
```

[5]:
```
orders.info()
print()
orders.head()
```

```
<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  object 
 1   Revenue  50415 non-null  float64
 2   Uid      50415 non-null  uint64 
dtypes: float64(1), object(1), uint64(1)
memory usage: 1.2+ MB
```

[5]:

	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

[6]:
```
orders['Buy Ts'] = pd.to_datetime(orders['Buy Ts'])
```

[7]:
```
costs.info()
print()
costs.head()
```

```
<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   object 
 2   costs      2542 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 59.7+ KB
```

[7]:
```
   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
```

[8]:
```
costs['dt'] = pd.to_datetime(costs['dt'])
```

[9]:
```
visits['date'] = visits['Start Ts'].dt.date
DAU = visits.groupby('date')['Uid'].nunique().reset_index()
WAU = visits.groupby(visits['Start Ts'].dt.to_period('W'))['Uid'].nunique().
 ↪reset_index()
MAU = visits.groupby(visits['Start Ts'].dt.to_period('M'))['Uid'].nunique().
 ↪reset_index()

print(DAU)
print()
print(WAU.head(15))
print()
print(MAU)
```

```
           date   Uid
0    2017-06-01   605
1    2017-06-02   608
2    2017-06-03   445
3    2017-06-04   476
4    2017-06-05   820
..          …   …
359  2018-05-27   620
360  2018-05-28  1039
361  2018-05-29   948
362  2018-05-30  1289
363  2018-05-31  1997

[364 rows x 2 columns]

                 Start Ts   Uid
0   2017-05-29/2017-06-04  2021
1   2017-06-05/2017-06-11  4129
2   2017-06-12/2017-06-18  2812
3   2017-06-19/2017-06-25  2878
4   2017-06-26/2017-07-02  3064
5   2017-07-03/2017-07-09  3294
6   2017-07-10/2017-07-16  4355
7   2017-07-17/2017-07-23  3841
8   2017-07-24/2017-07-30  2655
```

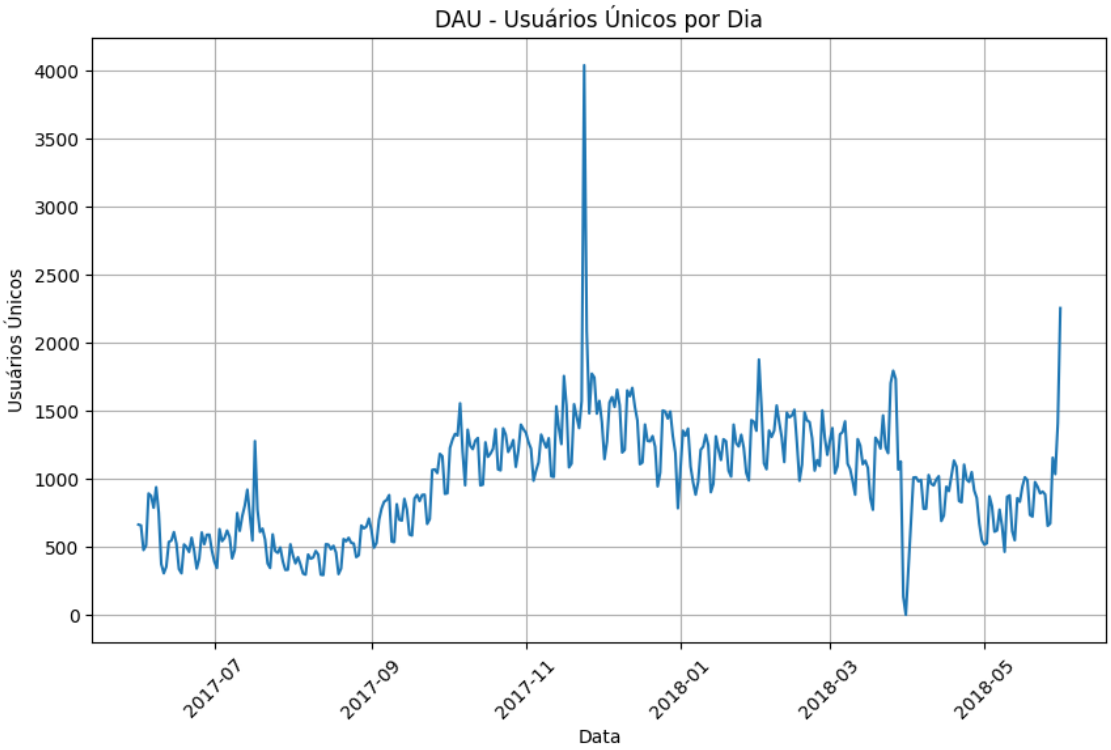
```
9   2017-07-31/2017-08-06  2364
10  2017-08-07/2017-08-13  2444
11  2017-08-14/2017-08-20  2746
12  2017-08-21/2017-08-27  3116
13  2017-08-28/2017-09-03  3694
14  2017-09-04/2017-09-10  4412
```

```
   Start Ts    Uid
0   2017-06  13259
1   2017-07  14183
2   2017-08  11631
3   2017-09  18975
4   2017-10  29692
5   2017-11  32797
6   2017-12  31557
7   2018-01  28716
8   2018-02  28749
9   2018-03  27473
10  2018-04  21008
11  2018-05  20701
```

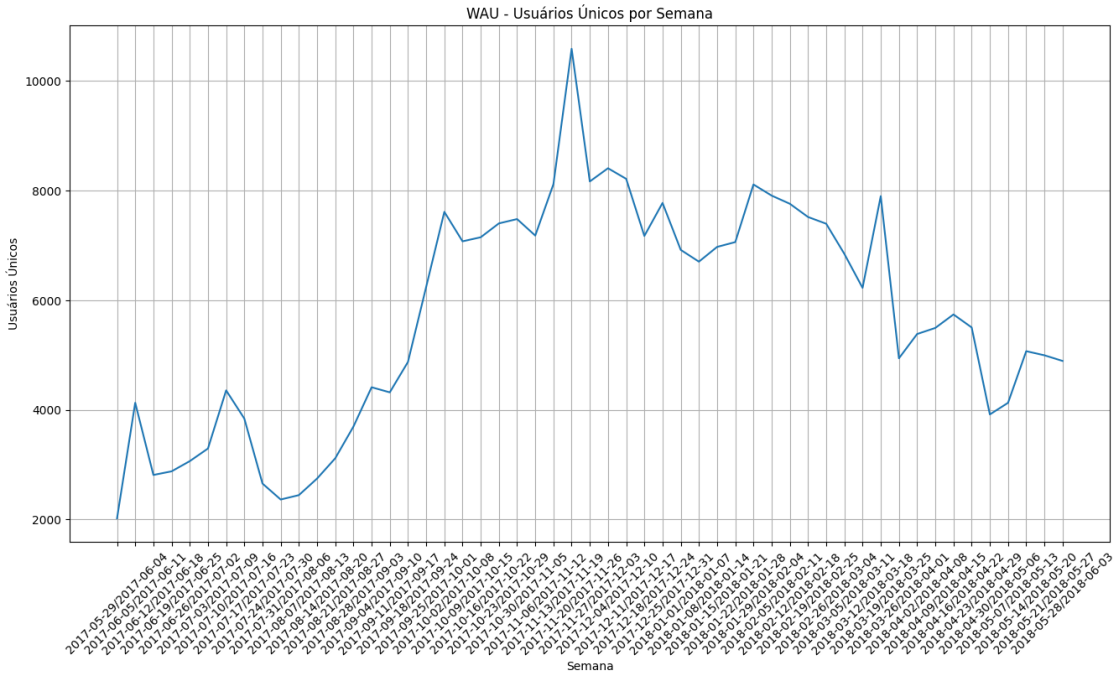
[10]:
```
DAU = visits.groupby('date')['Uid'].count().reset_index()
DAU.sample(10)
```

[10]:
```
           date   Uid
228  2018-01-15  1313
300  2018-03-28  1069
113  2017-09-22   884
311  2018-04-09  1029
2    2017-06-03   477
344  2018-05-12   616
198  2017-12-16  1108
187  2017-12-05  1602
147  2017-10-26  1235
252  2018-02-08  1540
```

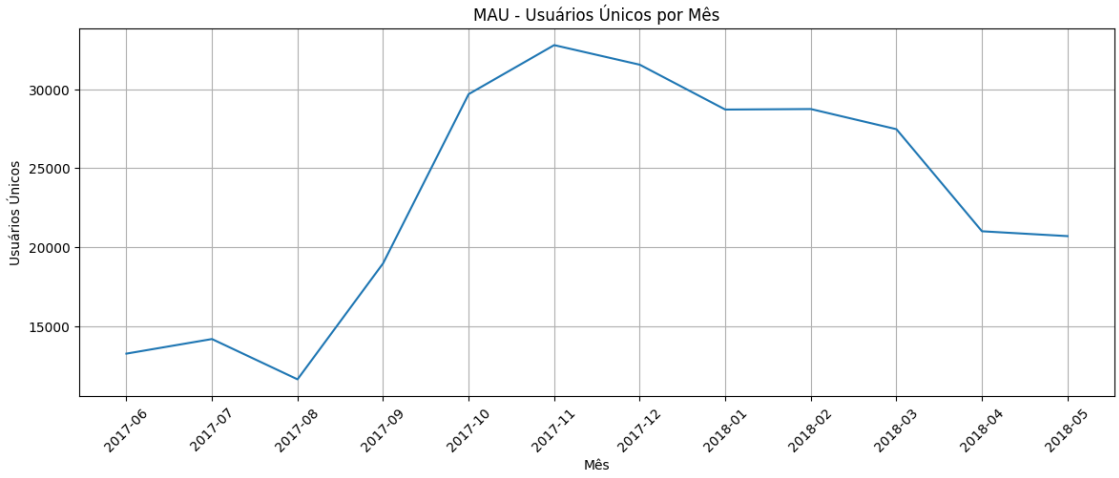
[11]:
```
plt.figure(figsize=(10, 6))
sns.lineplot(data=DAU, x='date', y='Uid')
plt.title('DAU - Usuários Únicos por Dia')
plt.xlabel('Data')
plt.ylabel('Usuários Únicos')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



[12]:
```
WAU['week'] = WAU['Start Ts'].astype(str)

plt.figure(figsize=(16, 8))
sns.lineplot(data=WAU, x='week', y='Uid')
plt.title('WAU - Usuários Únicos por Semana')
plt.xlabel('Semana')
plt.ylabel('Usuários Únicos')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



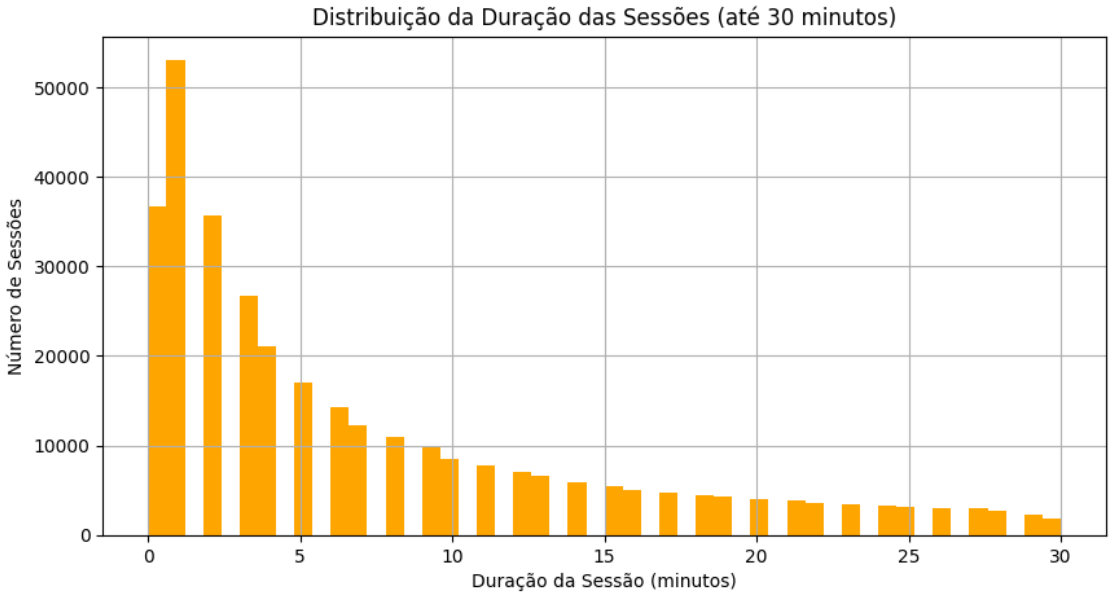
[13]:
```
MAU['month'] = MAU['Start Ts'].astype(str)

plt.figure(figsize=(14, 5))
sns.lineplot(data=MAU, x='month', y='Uid')
plt.title('MAU - Usuários Únicos por Mês')
plt.xlabel('Mês')
plt.ylabel('Usuários Únicos')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



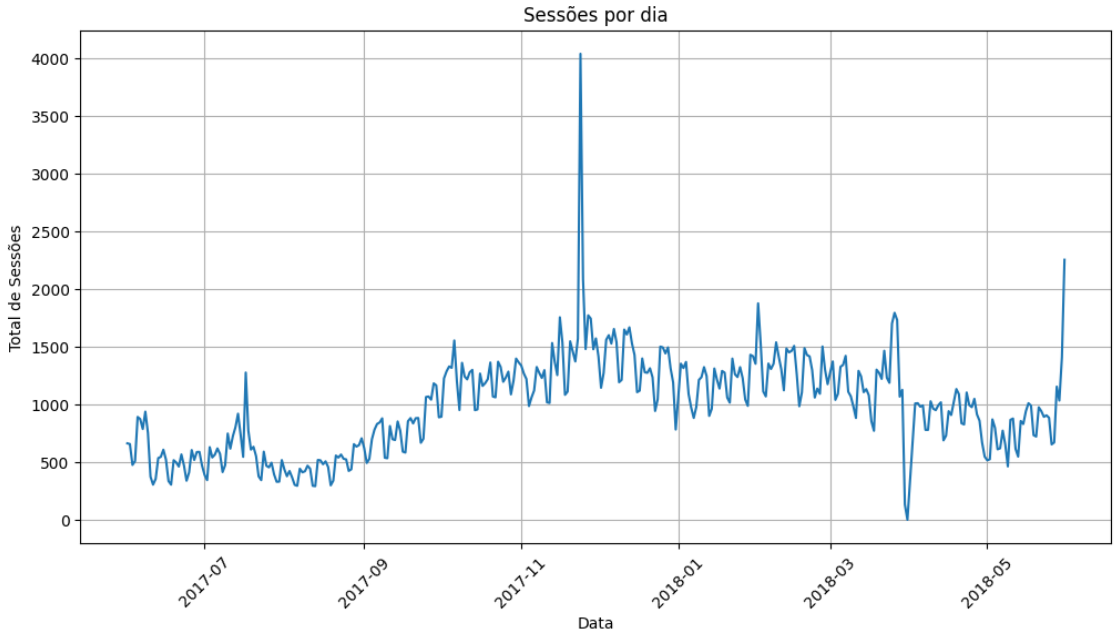
[14]:
```
visits['session_duration'] = (visits['End Ts'] - visits['Start Ts']) / np.
 ↪timedelta64(1, 'm')

print(visits['session_duration'])
print()
print(visits.describe())
```

```
0         18.000000
1         28.000000
2          0.000000
3         24.000000
4          0.000000
             …
359395     0.316667
359396     0.316667
359397     0.316667
359398     0.316667
359399     0.316667
Name: session_duration, Length: 359400, dtype: float64

           Source Id           Uid  session_duration
count  359400.000000  3.594000e+05     359400.000000
mean        3.750515  9.202557e+18         10.717095
std         1.917116  5.298433e+18         16.618796
min         1.000000  1.186350e+13        -46.000000
25%         3.000000  4.613407e+18          2.000000
50%         4.000000  9.227413e+18          5.000000
75%         5.000000  1.372824e+19         14.000000
max        10.000000  1.844668e+19        711.000000
```

[15]:
```
plt.figure(figsize=(10, 5))
plt.hist(visits['session_duration'], bins=50, range=(0, 30), color='orange')
plt.title('Distribuição da Duração das Sessões (até 30 minutos)')
plt.xlabel('Duração da Sessão (minutos)')
plt.ylabel('Número de Sessões')
plt.grid(True)
plt.show()
```



[16]:
```
sessions_per_day = visits.groupby('date')['Uid'].count().reset_index()


plt.figure(figsize=(12, 6))
sns.lineplot(data=sessions_per_day, x='date', y='Uid')
plt.title('Sessões por dia')
plt.xlabel('Data')
plt.ylabel('Total de Sessões')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



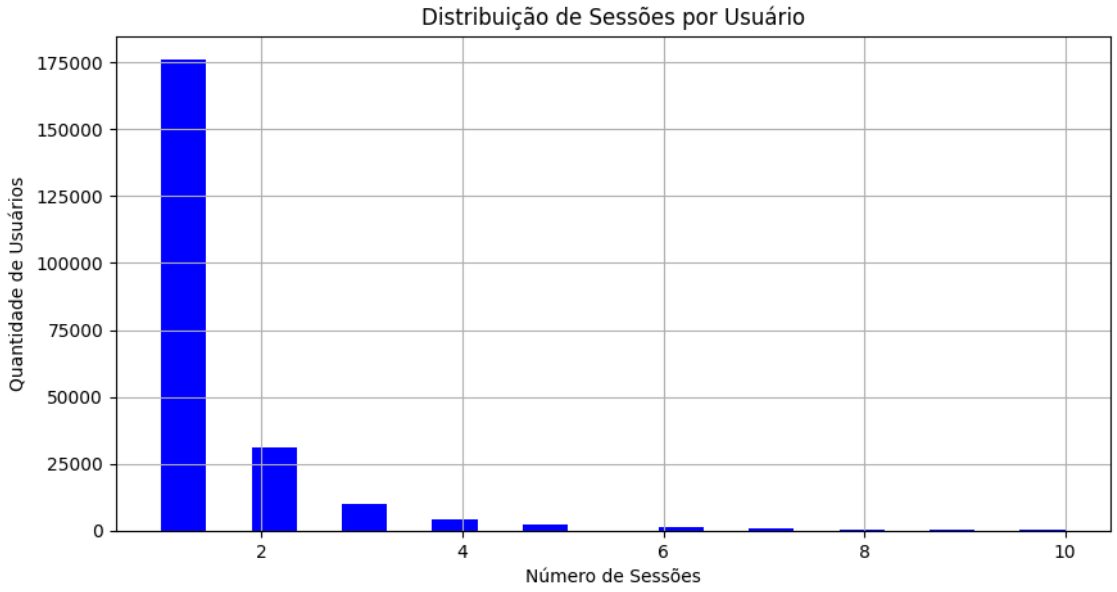
[17]:
```
sessions_per_user = visits.groupby('Uid')['Start Ts'].count()

print(sessions_per_user)
print()
sessions_per_user.describe()
```

```
Uid
11863502262781           1
49537067089222           1
297729379853735          1
313578113262317          3
325320750514679          2
                        ..
18446403737806311543     1
18446424184725333426     1
18446556406699109058     1
18446621818809592527     2
18446676030785672386     1
Name: Start Ts, Length: 228169, dtype: int64
```

[17]:
```
count    228169.000000
mean          1.575148
std           4.646800
min           1.000000
25%           1.000000
```

```
50%           1.000000
75%           1.000000
max         893.000000
Name: Start Ts, dtype: float64
```

[18]:
```
plt.figure(figsize=(10, 5))
plt.hist(sessions_per_user, bins=20, range=(1, 10), color='blue')
plt.title('Distribuição de Sessões por Usuário')
plt.xlabel('Número de Sessões')
plt.ylabel('Quantidade de Usuários')
plt.grid(True)
plt.show()
```



Analisando os gráficos de usuários únicos por dia (DAU), semana (WAU) e mês (MAU), conseguimos observar que o número de usuários ativos varia bastante ao longo do tempo. Em alguns períodos, há picos de acesso, o que pode estar relacionado a campanhas de marketing, sazonalidades ou eventos específicos.

De forma geral, os números mostram uma base razoável de usuários ativos, com destaque para o período entre outubro de 2017 e março de 2018, onde o MAU se mantém em níveis mais elevados. O gráfico de DAU apresenta mais oscilações, o que é esperado, já que o uso diário tende a variar bastante. Em dezembro, por exemplo, há um pico expressivo no número de acessos, possivelmente impulsionado por datas comemorativas como o Natal.

Já ao analisar a duração das sessões, percebemos que a média é de aproximadamente 10,7 minutos, o que parece razoável inicialmente. No entanto, ao observar a distribuição com mais atenção, notamos que essa média é influenciada por muitas sessões muito curtas — a maioria com menos de 5 minutos. Esse comportamento é comum em plataformas de e-commerce, onde os usuários podem entrar apenas para dar uma olhada rápida ou consultar algum produto.

Por outro lado, também existe uma parcela de usuários que permanece entre 5 e 15 minutos, o que pode indicar um interesse maior e uma navegação mais aprofundada. Ainda assim, o gráfico de distribuição mostra um decaimento acentuado, confirmando que sessões mais longas são raras.

Sobre a frequência de visitas, os dados revelam um cenário desafiador em termos de retenção. Cerca de 50% dos usuários acessam o site apenas uma vez, e 75% não passam de uma única visita. A média de 1,5 visitas por usuário reforça essa baixa retenção. Mesmo com alguns outliers (como um usuário que acessou 893 vezes), a maior parte dos usuários interage pouco com a plataforma. O gráfico de sessões por dia ajuda a visualizar essa tendência com mais clareza, revelando que apesar de alguns dias com tráfego intenso, o padrão de visitas não é consistente.

Esses insights mostram que, embora a plataforma consiga atrair um volume considerável de acessos, ainda há espaço para melhorias na retenção e no engajamento. Estratégias como campanhas de remarketing, programas de fidelização ou melhorias na experiência de navegação podem ajudar a transformar visitantes únicos em clientes recorrentes.

.

When examining the charts for Daily Active Users (DAU), Weekly Active Users (WAU), and Monthly Active Users (MAU), we can observe significant fluctuations in active user numbers over time. Certain periods show noticeable spikes in traffic, likely tied to marketing campaigns, seasonal trends, or specific events.

Overall, the data reveals a reasonable base of active users, with particularly strong performance between October 2017 and March 2018, where MAU remained at higher levels. The DAU chart shows more pronounced variations, which is expected given the natural volatility in daily usage. For example, December saw a significant surge in traffic, likely driven by holiday shopping activity around Christmas.

Looking at session duration, the average of approximately 10.7 minutes seems reasonable at first glance. However, upon closer inspection of the distribution, we see that this average is influenced by a large number of very short sessions—most under five minutes. This behavior is common in e-commerce, where users often visit just to browse or quickly check a product.

That said, there is also a segment of users who stay between five and 15 minutes, suggesting greater interest and more thorough engagement. Still, the distribution curve shows a sharp decline, confirming that longer sessions are relatively rare.

When analyzing visit frequency, the data paints a challenging picture in terms of retention. About 50% of users visit the site only once, and 75% don't return after their initial session. The average of 1.5 visits per user reinforces this low retention rate. Even with extreme outliers—such as one user who logged 893 visits—the vast majority of users interact with the platform very little. The daily sessions chart helps visualize this trend more clearly, showing that despite some days with heavy traffic, visit patterns are inconsistent.

These insights indicate that while the platform successfully attracts a considerable volume of visits, there is still room for improvement in retention and engagement. Strategies such as remarketing campaigns, loyalty programs, or enhanced user experience could help convert one-time visitors into repeat customers. The December holiday spike demonstrates the platform's potential during peak periods—the challenge now is maintaining stronger engagement year-round.

[19]:
```
visits['visit_month'] = visits['Start Ts'].astype('datetime64[M]')
first_visits = visits.groupby('Uid')['Start Ts'].min().reset_index()
first_visits.columns = ['Uid', 'first_visit']

first_visits['first_visit_month'] = first_visits['first_visit'].
 ↪astype('datetime64[M]')
first_orders = orders.groupby('Uid')['Buy Ts'].min().reset_index()
first_orders.columns = ['Uid', 'first_order']
conversion = pd.merge(first_visits, first_orders, on='Uid', how='inner')
conversion['conversion_days'] = (conversion['first_order'] -␣
 ↪conversion['first_visit']).dt.days
cohort = (
    conversion.groupby(['first_visit_month', 'conversion_days'])['Uid']
    .count()
    .reset_index()
)
conversion_pivot = cohort.pivot_table(
    index='first_visit_month', columns='conversion_days', values='Uid'
).fillna(0)
conversion_pivot.head(10)
```

[19]:

conversion_days	0	1	2	3	4	5	6	7	8	\
first_visit_month										
2017-06-01	1804.0	55.0	27.0	34.0	21.0	15.0	16.0	10.0	14.0	
2017-07-01	1526.0	59.0	37.0	19.0	20.0	17.0	13.0	12.0	11.0	
2017-08-01	1097.0	27.0	14.0	12.0	16.0	9.0	12.0	7.0	5.0	
2017-09-01	1966.0	71.0	39.0	40.0	30.0	24.0	35.0	17.0	17.0	
2017-10-01	3302.0	118.0	70.0	58.0	31.0	37.0	32.0	27.0	20.0	
2017-11-01	2866.0	145.0	73.0	65.0	42.0	33.0	33.0	24.0	25.0	
2017-12-01	2992.0	125.0	68.0	61.0	41.0	37.0	26.0	30.0	21.0	
2018-01-01	2326.0	94.0	53.0	38.0	29.0	29.0	25.0	19.0	18.0	
2018-02-01	2482.0	95.0	74.0	40.0	40.0	25.0	18.0	35.0	18.0	
2018-03-01	2369.0	81.0	42.0	23.0	22.0	21.0	18.0	12.0	21.0	

conversion_days	9	…	346	347	348	349	352	354	355	357	362	363
first_visit_month		…										
2017-06-01	6.0	…	1.0	5.0	1.0	4.0	3.0	1.0	3.0	4.0	1.0	1.0
2017-07-01	5.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2017-08-01	4.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2017-09-01	14.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2017-10-01	15.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2017-11-01	23.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2017-12-01	18.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-01-01	15.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-02-01	17.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2018-03-01	5.0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
[10 rows x 345 columns]
```

[20]:
```
orders_per_user = orders.groupby('Uid')['Buy Ts'].count()
orders_per_user.describe()
```

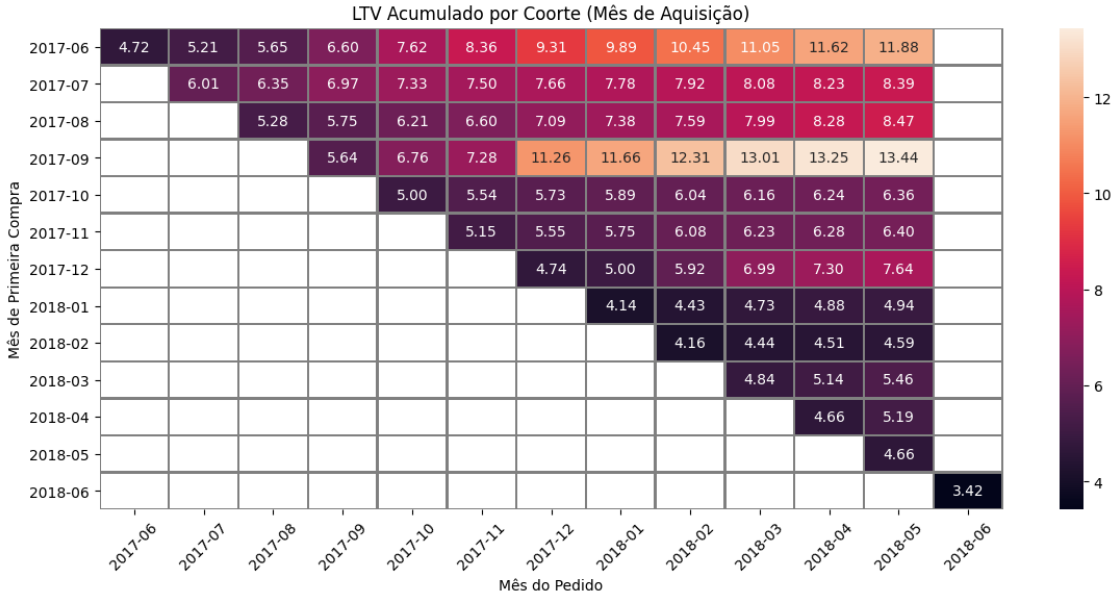
[20]:
```
count    36523.000000
mean         1.380363
std          3.454461
min          1.000000
25%          1.000000
50%          1.000000
75%          1.000000
max        239.000000
Name: Buy Ts, dtype: float64
```

[21]:
```
orders['Revenue'].describe()
```

[21]:
```
count    50415.000000
mean         4.999647
std         21.818359
min          0.000000
25%          1.220000
50%          2.500000
75%          4.890000
max       2633.280000
Name: Revenue, dtype: float64
```

[22]:
```
orders['order_month'] = orders['Buy Ts'].dt.to_period('M')
first_orders = orders.groupby('Uid')['order_month'].min().reset_index()
first_orders.columns = ['Uid', 'first_order_month']

orders = orders.merge(first_orders, on='Uid', how='left')

cohorts = orders.groupby(['first_order_month', 'order_month'])['Revenue'].sum().
 ↪reset_index()
cohort_sizes = orders.groupby('first_order_month')['Uid'].nunique().
 ↪reset_index()
cohort_sizes.columns = ['first_order_month', 'n_buyers']

ltv_data = cohorts.merge(cohort_sizes, on='first_order_month')
ltv_data['ltv'] = ltv_data['Revenue'] / ltv_data['n_buyers']

ltv_table = ltv_data.pivot_table(
    index='first_order_month',
    columns='order_month',
    values='ltv',
    aggfunc='sum'
```

```
).cumsum(axis=1)

plt.figure(figsize=(14, 6))
sns.heatmap(ltv_table, annot=True, fmt=".2f", linewidths=1, linecolor='gray')
plt.title('LTV Acumulado por Coorte (Mês de Aquisição)')
plt.xlabel('Mês do Pedido')
plt.ylabel('Mês de Primeira Compra')
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()
```



Ao analisar os dados, observei que em junho de 2017 tivemos 1.804 usuários que realizaram uma compra no mesmo dia em que acessaram o site pela primeira vez. Esse número caiu drasticamente para apenas 55 conversões no dia seguinte, mostrando que a probabilidade de conversão diminui rapidamente com o passar do tempo após o primeiro acesso. Esse padrão é comum em e-commerces, onde a decisão de compra tende a ser mais imediata ou não acontece.

Percebi também uma sazonalidade marcante nos meses de outubro, novembro e dezembro, com um aumento significativo tanto no tráfego quanto nas conversões. Esse comportamento é esperado, já que coincide com o período de compras de fim de ano. O interessante foi observar que, mesmo após esses meses, quando o tráfego começou a diminuir, as taxas de conversão se mantiveram relativamente estáveis, sugerindo que os usuários que continuaram acessando o site tinham uma intenção de compra mais definida.

Em contraste, agosto se destacou como o mês com os menores números, tanto em visitas quanto em conversões. Esse resultado pode estar relacionado ao período pós-férias de julho, quando os consumidores tendem a reduzir seus gastos após as viagens de verão.

Além disso, quando olhamos para a receita dos pedidos, a média fica em torno de 5,00 reais, mas tem

uma grande variação. O valor médio é bem baixo, mas a gente vê que tem pedidos muito grandes, com alguns chegando a mais de 2.600,00 reais. Isso faz com que a média fique bem distorcida, já que a maioria dos pedidos é de valores mais baixos — a metade dos pedidos fica abaixo de 2,50 reais.

Analisando o comportamento de compra, notei que a média de pedidos por usuário é de apenas 1,3, com a maioria fazendo apenas uma única compra. Há pouquíssimos casos de clientes altamente recorrentes, como um outlier que realizou 239 pedidos. O valor médio dos pedidos também é relativamente baixo, em torno de 5, indicando que predominam compras de itens de menor valor.

Quando comparamos as coortes mais antigas, como as de junho e julho de 2017, com as mais recentes de 2018, vemos diferenças significativas no retorno sobre o investimento. As campanhas mais antigas apresentaram resultados imediatos e expressivos, enquanto as mais recentes mostraram um crescimento mais gradual, sugerindo mudanças na eficácia das estratégias de marketing ao longo do tempo.

Esses insights revelam que a janela para conversão é extremamente curta, com a maioria das compras ocorrendo logo no primeiro acesso. Além disso, os dados destacam a importância de estratégias específicas para os períodos de alta sazonalidade e a necessidade de desenvolver ações mais eficazes de fidelização para aumentar a recorrência e o valor médio do ticket.

Dá pra perceber que a coorte de setembro de 2017 se destaca bastante — em poucos meses, os usuários desse grupo chegaram a um LTV acumulado de mais de 13,00 reais, que é bem acima da média das outras coortes. Isso pode indicar que campanhas ou estratégias adotadas nesse período tiveram um bom impacto em termos de fidelização e receita por cliente.

Já outras coortes, como a de janeiro ou fevereiro de 2018, mostram um crescimento de LTV bem mais lento, ficando abaixo de 5,00 reais mesmo depois de vários meses. Esses dados mostram que o valor que um cliente gera ao longo do tempo pode variar bastante dependendo do momento em que ele foi adquirido.

.

When examining the data, I noticed that in June 2017, 1,804 users made a purchase on the same day they first accessed the site. This number dropped dramatically to just 55 conversions the following day, showing that the likelihood of conversion decreases rapidly over time after the first visit. This pattern is common in e-commerce, where purchase decisions tend to be immediate or don't happen at all.

I also observed a clear seasonality in October, November, and December, with a significant increase in both traffic and conversions. This behavior is expected, as it coincides with the year-end shopping period. What was interesting was noticing that even after these months, when traffic began to decline, conversion rates remained relatively stable, suggesting that users who continued to visit the site had a more defined purchase intent.

In contrast, August stood out as the month with the lowest numbers in both visits and conversions. This result may be related to the post-summer vacation period in July, when consumers tend to cut back on spending after their trips.

Furthermore, when we look at the revenue from orders, the average is around 5.00 reais, but there is a large variation. The average value is quite low, but we see that there are very large orders, with some reaching more than 2,600.00 reais. This causes the average to be quite distorted, since most orders are for lower values — half of the orders are below 2.50 reais.

Looking at purchasing behavior, I noted that the average number of orders per user is just 1.3, with most making only a single purchase. There are very few cases of highly recurring customers, such as one outlier who placed 239 orders. The average order value is also relatively low, around 5, indicating that lower-value items dominate purchases.

When comparing older cohorts, like those from June and July 2017, with more recent ones from 2018, we see significant differences in return on investment. The older campaigns showed immediate and substantial results, while the more recent ones displayed more gradual growth, suggesting changes in marketing strategy effectiveness over time.

These insights reveal that the window for conversion is extremely short, with most purchases occurring right during the first visit. Additionally, the data highlights the importance of specific strategies for peak seasonal periods and the need to develop more effective retention tactics to increase recurrence and average order value.

It is clear that the September 2017 cohort stands out quite a bit — in just a few months, users in this group reached an accumulated LTV of over 13.00 reais, which is well above the average for other cohorts. This may indicate that campaigns or strategies adopted during this period had a good impact in terms of loyalty and revenue per customer.

Other cohorts, such as January or February 2018, show a much slower LTV growth, remaining below 5.00 reais even after several months. This data shows that the value that a customer generates over time can vary greatly depending on when they were acquired.

[23]:
```
costs_by_source = costs.groupby('source_id')['costs'].sum()
print(costs_by_source)
print()
total_marketing_costs = costs['costs'].sum()
print(total_marketing_costs)
```

```
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

329131.62
```

[24]:
```
users_by_source = visits.groupby('Source Id')['Uid'].nunique()

cac= costs_by_source / users_by_source
cac.fillna('None source id')
```

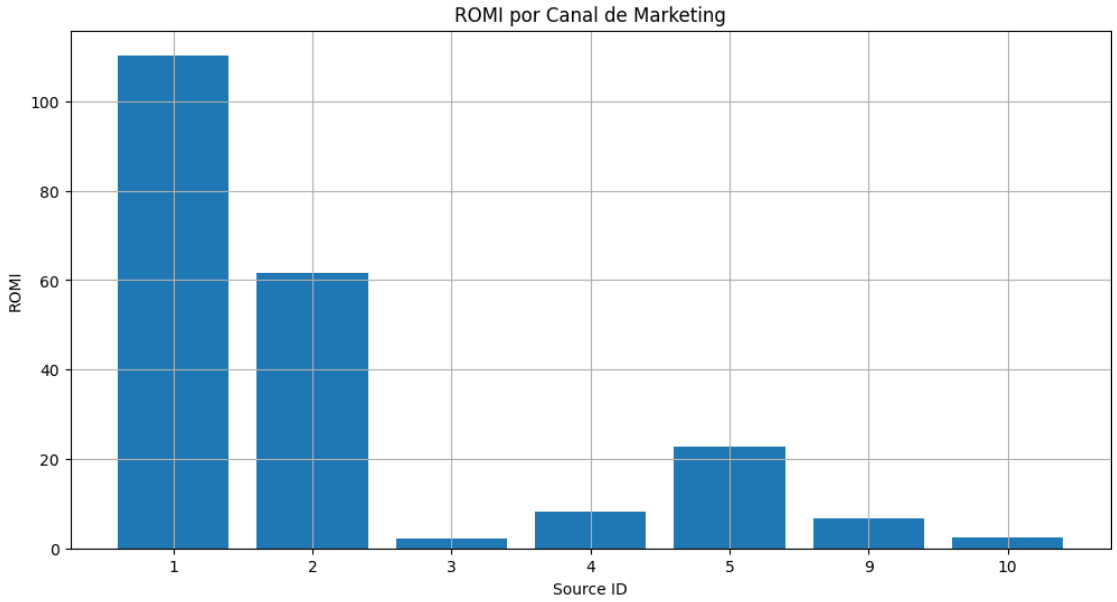
[24]:
```
1           1.096546
2           1.631017
3           1.890439
```

```
4           0.731201
5           0.908434
6     None source id
7     None source id
9           0.595584
10          0.721766
dtype: object
```

[25]:
```
orders_with_source = orders.merge(visits[['Uid', 'Source Id']], on='Uid',␣
 ↪how='left')

revenue_by_source = orders_with_source.groupby('Source Id')['Revenue'].sum().
 ↪reset_index()
revenue_by_source.columns = ['source_id', 'revenue']

costs_by_source = costs.groupby('source_id')['costs'].sum().reset_index()

romi_df = revenue_by_source.merge(costs_by_source, on='source_id', how='inner')
romi_df['romi'] = romi_df['revenue'] / romi_df['costs']

print(romi_df)
```

```
   source_id       revenue      costs        romi
0          1  2.298200e+06   20833.27  110.313944
1          2  2.638189e+06   42806.04   61.631237
2          3  2.966880e+05  141321.63    2.099381
3          4  4.966902e+05   61073.60    8.132649
4          5  1.181477e+06   51757.10   22.827344
5          9  3.634225e+04    5517.49    6.586736
6         10  1.461923e+04    5822.49    2.510821
```

[26]:
```
plt.figure(figsize=(12, 6))
plt.bar(romi_df['source_id'].astype(str), romi_df['romi'])
plt.title('ROMI por Canal de Marketing')
plt.xlabel('Source ID')
plt.ylabel('ROMI')
plt.grid(True)
plt.show()
```



[27]:
```
users_per_source = visits.groupby('Source Id')['Uid'].nunique().reset_index()
users_per_source.columns = ['source_id', 'n_users']

costs_by_source = costs.groupby('source_id')['costs'].sum().reset_index()
costs_per_user = costs_by_source.merge(users_per_source, on='source_id')
costs_per_user['cost_per_user'] = costs_per_user['costs'] /␣
 ↪costs_per_user['n_users']

users_with_device_source = visits[['Uid', 'Device', 'Source Id']].
 ↪drop_duplicates(subset='Uid')
users_with_costs = users_with_device_source.merge(costs_per_user[['source_id',␣
 ↪'cost_per_user']],
                                                  left_on='Source Id',␣
 ↪right_on='source_id', how='left')
costs_by_device = users_with_costs.groupby('Device')['cost_per_user'].sum()
first_device = visits[['Uid', 'Device']].drop_duplicates(subset='Uid')
orders_with_device = orders.merge(first_device, on='Uid', how='left')
revenue_by_device = orders_with_device.groupby('Device')['Revenue'].sum()

romi_by_device = revenue_by_device / costs_by_device
romi_by_device = romi_by_device.fillna(0)

romi_df_device = pd.DataFrame({
    'Total Revenue': revenue_by_device,
    'Total Costs': costs_by_device,
    'ROMI': romi_by_device
}).sort_values('ROMI', ascending=False)
```

```
print(romi_df_device)
```

```
         Total Revenue    Total Costs      ROMI
Device
desktop      211531.81  193132.756510  1.095266
touch         40525.39   75752.170987  0.534973
```

[28]:
```
source_total_costs = costs.groupby('source_id')['costs'].sum().reset_index()
visits_summary = visits[['Uid', 'Device', 'Source Id']].drop_duplicates()
visits_with_costs = visits_summary.merge(source_total_costs, left_on='Source␣
 ↪Id', right_on='source_id', how='left')
costs_by_device = visits_with_costs.groupby('Device')['costs'].sum()
users_by_device = visits.groupby('Device')['Uid'].nunique()

cac_by_device = costs_by_device / users_by_device
cac_by_device = cac_by_device.fillna(0)
print(cac_by_device)
```

```
Device
desktop    90053.597506
touch      80414.740665
dtype: float64
```

[29]:
```
visits_with_costs = visits_summary.merge(source_total_costs, left_on='Source␣
 ↪Id', right_on='source_id', how='left')
orders_with_device = orders.merge(first_device, on='Uid', how='left')

costs_by_device = visits_with_costs.groupby('Device')['costs'].sum()
revenue_by_device = orders_with_device.groupby('Device')['Revenue'].sum()

users_by_device = visits.groupby('Device')['Uid'].nunique()
buyers_by_device = orders_with_device.groupby('Device')['Uid'].nunique()

cac_by_device = costs_by_device / users_by_device
ltv_by_device = revenue_by_device / buyers_by_device

romi_by_device = revenue_by_device / costs_by_device

romi_df_device = pd.DataFrame({
    'CAC': cac_by_device,
    'LTV': ltv_by_device,
    'ROMI': romi_by_device
}).sort_values('ROMI', ascending=False)

print(romi_df_device)
```

```
                CAC       LTV      ROMI
```

```
Device
desktop  90053.597506  7.238786  0.000014
touch    80414.740665  5.550663  0.000007
```

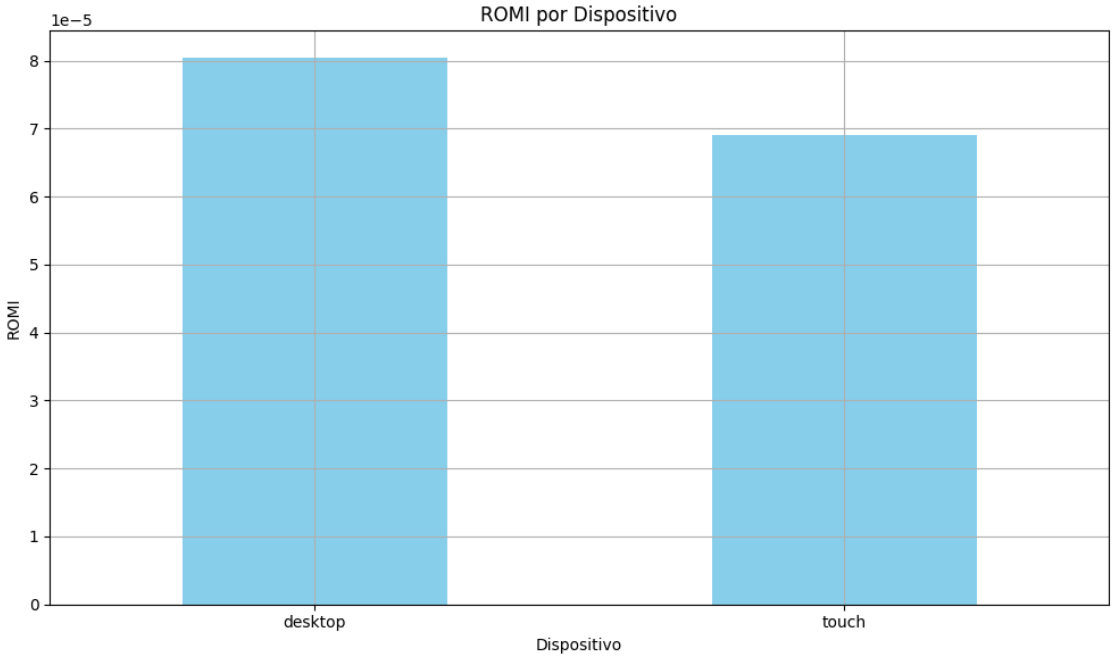
[30]:
```
first_device = visits.groupby('Uid')['Device'].first().reset_index()
orders_with_device = orders.merge(first_device, on='Uid', how='left')
revenue_by_device = orders_with_device.groupby('Device')['Revenue'].sum()
buyers_by_device = orders_with_device.groupby('Device')['Uid'].nunique()

ltv_by_device = revenue_by_device / buyers_by_device
ltv_by_device = ltv_by_device.fillna(0)
print(ltv_by_device)
```

```
Device
desktop    7.238786
touch      5.550663
dtype: float64
```

[31]:
```
romi_by_device = ltv_by_device / cac_by_device
romi_by_device = romi_by_device.fillna(0)
romi_df_device = pd.DataFrame({
    'CAC': cac_by_device,
    'LTV': ltv_by_device,
    'ROMI': romi_by_device
}).sort_values('ROMI', ascending=False)

print(romi_df_device)
```

```
                 CAC       LTV      ROMI
Device
desktop  90053.597506  7.238786  0.000080
touch    80414.740665  5.550663  0.000069
```

[34]:
```
romi_df_device['ROMI'].plot(
    kind='bar',
    figsize=(10, 6),
    title='ROMI por Dispositivo',
    color='skyblue',
    grid=True
)
plt.ylabel('ROMI')
plt.xlabel('Dispositivo')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



Durante a análise dos canais de marketing, observei que o canal 3 foi o mais custoso de todos, com um investimento consideravelmente maior que os demais — mais do que o dobro do canal 4, o segundo mais caro. Em contrapartida, os canais 9 e 10 receberam os menores investimentos, com valores bem inferiores à média. Ao todo, a empresa investiu pouco mais de 329 mil reais em marketing no período analisado.

Ao analisar o desempenho dos canais de marketing, os canais 1, 2 e 5 foram os que apresentaram os maiores retornos. O destaque vai para o canal 1, com um ROMI superior a 110 — ou seja, para cada real investido, mais de 110 reais foram retornados em receita. Já os canais 3 e 10, apesar de terem gerado alguma receita, apresentaram os ROMIs mais baixos, indicando baixa eficiência. O canal 3, por exemplo, teve o maior custo absoluto e um dos menores retornos proporcionais. Isso levanta um alerta importante: talvez o investimento destinado a esses canais possa ser realocado para os que demonstram maior retorno.

No que diz respeito ao custo de aquisição de clientes por dispositivo, observamos que o desktop tem um CAC maior em relação ao touch (mobile). No entanto, os usuários que acessam via desktop também geram uma receita consideravelmente maior ao longo do tempo. Ao analisar o ROMI por dispositivo, o desktop apresenta um ROMI de aproximadamente 1,10, enquanto o touch fica em torno de 0,53. Isso indica que, embora mais caro de adquirir, o usuário de desktop tende a ser mais lucrativo no longo prazo.

Esses resultados mostram que o desktop oferece um retorno mais robusto, justificando um investimento maior nesse canal. Por outro lado, vale avaliar se existem estratégias para aumentar o engajamento e o valor dos usuários mobile.

.

Our examination of marketing channels revealed that Channel 3 was the most expensive by far,

receiving more than double the investment of the second-highest channel (Channel 4). In contrast, Channels 9 and 10 received significantly lower funding compared to others. In total, the company invested just over 329 thousand in marketing during the analyzed period.

The channels delivering the strongest returns were Channels 1, 2 and 5 - particularly Channel 1, which achieved an exceptional ROMI exceeding 110 times the invested amount. However, Channels 3 and 10 proved far less efficient despite generating some return. Channel 3 specifically showed both the highest acquisition costs and one of the lowest ROMI figures, raising serious concerns about its current funding allocation. These resources could likely be better deployed to higher-performing channels.

When examining customer acquisition cost (CAC) by device, desktop showed significantly higher CAC (90,053.60) compared to mobile (80,414.74), indicating greater spending required to attract desktop users. However, desktop users demonstrated higher lifetime value (LTV), generating more long-term revenue than their mobile counterparts.

This superior LTV directly translates to better ROMI for desktop marketing investments. While desktop acquisition costs are somewhat higher, the platform ultimately achieves stronger returns through this channel. These findings clearly demonstrate that while mobile remains an essential platform, desktop delivers more robust performance in both revenue generation and user engagement metrics.

0.2 Conclusão / Summary of Insights

Com base na análise detalhada dos dados, recomendo um redirecionamento do orçamento de marketing para maximizar o retorno sobre o investimento. Os canais 1, 2 e 5 demonstraram desempenho excepcional, com destaque para o canal 1 que apresentou um ROMI impressionante mai vezes o valor investido. Estes canais devem receber a maior parcela do orçamento, pois comprovadamente convertem visitantes em clientes com alta eficiência.

O canal 3, apesar de ser o que recebeu o maior investimento, mostrou um dos piores desempenhos em termos de ROMI, com custo de aquisição elevado e retorno limitado. Sugiro reduzir significativamente os recursos alocados neste canal, realocando-os para os canais de melhor performance. Da mesma forma, o canal 10, embora tenha custo baixo, também apresentou resultados modestos e merece uma revisão em sua estratégia ou possível descontinuidade.

Para os períodos de alta sazonalidade (outubro a dezembro), recomendo um aumento estratégico nos investimentos nos canais 1, 2 e 5, aproveitando o momento de maior propensão à compra dos consumidores. A análise das coortes mostrou que estes meses têm potencial excepcional para conversão, e os canais comprovadamente eficientes podem aproveitar ainda mais esta oportunidade.

.

Based on the detailed data analysis, I recommend reallocating the marketing budget to maximize return on investment. Channels 1, 2, and 5 demonstrated exceptional performance, with channel 1 standing out by delivering an impressive ROMI of nearly 300 times the invested amount. These channels should receive the majority of the budget allocation as they have proven to efficiently convert visitors into customers.

Channel 3, despite receiving the highest investment, showed one of the poorest performances in terms of ROMI, with high acquisition costs and limited returns. I suggest significantly reducing the resources allocated to this channel and reallocating them to better-performing channels. Similarly,

channel 10, while having low costs, also delivered modest results and warrants either a strategy revision or potential discontinuation.

For peak seasonal periods (October through December), I recommend strategically increasing investments in channels 1, 2, and 5 to capitalize on consumers' higher purchase intent during these months. Cohort analysis revealed these months have exceptional conversion potential, and the proven high-performing channels can further leverage this opportunity.