Análise Exploratória de Dados

Pet Love - Talents Data Case

Matheus Miranda Brandão

Imports

In [ ]:
```
import numpy as np
import datetime
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style='darkgrid')
sns.set_theme(style="ticks", rc={"axes.spines.right": False, "axes.spines
PALETTE = 'mako'
```

Observação inicial dos dados

In [ ]:
```
df = pd.read_pickle('../datasets/processed/data-test-analytics.pkl')
df.head()
```

Out[ ]:

	created_at	updated_at	deleted_at	birth_date	status	version	city	state	neighbo
0	2017-08-15 07:05:00	2021-01-14 11:23:00	NaT	1974-07-10	active	2.31.7	Peixoto da Praia	AM	Apare
1	2019-12-31 21:53:00	2021-01-08 11:23:00	NaT	1940-07-06	paused	3.30.12	Fernandes	RR	Santa
2	2019-03-07 23:46:00	2021-01-07 11:23:00	NaT	1963-03-18	active	3.28.9	Lopes	RR	
3	2018-07-21 10:17:00	2021-01-10 11:23:00	NaT	1980-11-21	active	3.34.3	Campos do Campo	PE	C
4	2018-06-08 12:09:00	2021-01-18 11:23:00	NaT	1959-07-07	active	3.19.8	das Neves	RJ	Vila S Se

In [ ]:
```
# Entendimento inicial dos dados
df.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype         
---  ------              --------------  -----         
 0   created_at          10000 non-null  datetime64[ns]
 1   updated_at          10000 non-null  datetime64[ns]
 2   deleted_at          505 non-null    datetime64[ns]
 3   birth_date          10000 non-null  datetime64[ns]
 4   status              10000 non-null  category      
 5   version             10000 non-null  object        
 6   city                10000 non-null  object        
 7   state               10000 non-null  category      
 8   neighborhood        10000 non-null  object        
 9   last_date_purchase  10000 non-null  datetime64[ns]
 10  average_ticket      10000 non-null  float32       
 11  items_quantity      10000 non-null  int8          
 12  all_revenue         10000 non-null  float32       
 13  all_orders          10000 non-null  int8          
 14  recency             10000 non-null  int16         
 15  marketing_source    10000 non-null  category      
dtypes: category(3), datetime64[ns](5), float32(2), int16(1), int8(2), ob
ject(3)
memory usage: 2.6 MB
```

In [ ]:
```
df.describe(include='number').T
```

Out[ ]:

	count	mean	std	min	25%	50%	
average_ticket	10000.0	216.894699	22.757214	131.378677	201.398853	217.019478	232.
items_quantity	10000.0	8.499800	3.026040	1.000000	6.000000	8.000000	11.
all_revenue	10000.0	1174.888550	763.141968	0.000000	494.873558	1172.751953	1798.
all_orders	10000.0	5.415400	3.457577	0.000000	2.000000	5.000000	8.
recency	10000.0	67.192900	175.723276	1.000000	31.000000	35.000000	39.

In [ ]:
```
sorted(df.all_orders.unique().tolist())
```

Out[ ]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [ ]:
```
sorted(df.items_quantity.unique().tolist())
```

Out[ ]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [ ]:
```
df.describe(include=['object', 'category']).T
```

Out[ ]:

	count	unique	top	freq
status	10000	3	active	8524
version	10000	2905	4.26.7	24
city	10000	2406	Cardoso	161
state	10000	27	TO	409
neighborhood	10000	482	Vila Antena	36
marketing_source	10000	6	organic_search	3699

In [ ]:
```
def birth_date_to_age(birth_date):
    today = datetime.date.today()
    return today.year - birth_date.year - ((today.month, today.day) < (bi
```

Para facilitar a análise foi utilizada a coluna de data de nascimento para extrair a idade dos clientes.

A função utiliza o dia atual, e essa análise foi feita no dia 2023-05-04.

In [ ]:
```
df['age'] = df.birth_date.apply(birth_date_to_age)
df.drop(columns=['birth_date'], inplace=True)
del birth_date_to_age
```

Agrupamento em anos do somatório dos casos de Status, para facilitar a visualização.

Foi observado o grande aglomerado relacionado ao ano de 2021, nas colunas relacionadas a atualização de conta e data de ultima compra.

In [ ]:
```
for coluna in df.select_dtypes(include=['datetime64[ns]']).columns:
    print(f'{coluna}: ')
    number = 1
    for i, item in zip(df[coluna].dt.year.value_counts(dropna=False).sort_
        if isinstance(i, float) and not np.isnan(i): i = int(i)
        end = '\t'

        if number == len(df[coluna].dt.year.value_counts(dropna=False).so

        print(f'{i} - {item}', end=end)
        number += 1

    if coluna not in 'last_date_purchase': print('\n' + '-'*50 + '\n')
```

```
created_at:
2016 - 1770     2017 - 2001     2018 - 1995
2019 - 2035     2020 - 1946     2021 - 253
--------------------------------------------------

updated_at:
2016 - 9        2017 - 33       2018 - 59
2019 - 106      2020 - 315      2021 - 9478
--------------------------------------------------

deleted_at:
2016 - 9        2017 - 33       2018 - 59
2019 - 106      2020 - 239      2021 - 59       nan - 9495
--------------------------------------------------

last_date_purchase:
2016 - 27       2017 - 75       2018 - 100
2019 - 125      2020 - 234      2021 - 9439
```

Agora o agrupamento foi realizado separando entre os itens de status:

- Ativo
- Pausado
- Cancelado

Foi observado que os dados referentes a quando o cliente atualiza a conta ou compra um item estão altamente correlacionados entre si.

A criação dos dados nos traz um olhar mais geral do problema.

In [ ]:
```
pd.DataFrame(
    [
        df.status.groupby(df.created_at.dt.year).value_counts(),
        df.status.groupby(df.updated_at.dt.year).value_counts(),
        df.status.groupby(df.deleted_at.dt.year).value_counts(),
        df.status.groupby(df.last_date_purchase.dt.year).value_counts()
    ], index=['created_at', 'updated_at', 'deleted_at', 'last_date_purchas
).T
```

Out[ ]:

	status	created_at	updated_at	deleted_at	last_date_purchase
2016	**active**	1480	0	0	0
	paused	189	0	0	0
	canceled	101	9	9	27
2017	**active**	1743	0	0	0
	paused	174	0	0	0
	canceled	84	33	33	75
2018	**active**	1693	0	0	0
	paused	204	0	0	0
	canceled	98	59	59	100
2019	**active**	1736	0	0	0
	paused	195	0	0	0
	canceled	104	106	106	125
2020	**active**	1660	66	0	66
	paused	182	10	0	10
	canceled	104	239	239	158
2021	**active**	212	8458	0	8458
	paused	27	961	0	961
	canceled	14	59	59	20

Agora foi criada uma função para agrupar em dias e mês, para facilitar a visualização.
Além disso calcularemos o churn rate utilizando a fórmula:

$$\text{churnRate} = \text{round}\left(\frac{\text{canceled}}{\text{active} + \text{paused}} \cdot 100, 2\right) \tag{1}$$

In [ ]:
```
def group_day_and_month(df_=df, column: str = ''):
    if column not in df_.select_dtypes(include=['datetime64[ns]']):
        return None

    df_['dt_ano_mes'] = (df_[column].dt.year*100 + df_[column].dt.month)

    df_['dt_ano_mes'] = pd.to_datetime(df_.dt_ano_mes, format='%Y%m')
    dummies = pd.get_dummies(df_.status)

    return pd.concat([df_, dummies], axis=1)

def calculate_churn_rate(df_=df, date_column: str = '', column_to_groupby
    df_ = group_day_and_month(df_, date_column)

    if isinstance(df_, type(None)):
        return None

    df_ = df_.groupby([
        column_to_groupby
    ]).agg({
        'active': 'sum',
        'canceled': 'sum',
        'paused': 'sum'
    })

    df_['churn_rate'] = np.round(df_['canceled'] / (df_['active'] + df_['p

    return df_
```

Desta vez observaremos os meses que mais ocorreram os cancelamentos.

In [ ]:
```
# Vamos agrupar os dados por mês e calcular a taxa de churn
df_churn = calculate_churn_rate(df_=df, date_column='created_at')
df_churn = df_churn.groupby(df_churn.index.month).agg({
    'active': 'sum',
    'canceled': 'sum',
    'paused': 'sum'
})

df_churn['churn_rate'] = np.round(df_churn['canceled'] / (df_churn['active

df_churn.sort_values(by='churn_rate', ascending=False)
```

Out[ ]:

dt_ano_mes	active	canceled	paused	churn_rate
11	682	49	79	6.44
4	674	47	79	6.24
2	707	47	80	5.97
3	696	46	89	5.86
8	681	44	82	5.77
1	716	44	79	5.53
10	712	42	92	5.22
9	666	38	80	5.09
12	736	39	67	4.86
6	759	40	80	4.77
7	744	37	76	4.51
5	751	32	88	3.81

Visualizaremos agora a taxa de cancelamento por mês ao longo dos anos, ordenando-os de maneira decrescente.

In [ ]:
```
calculate_churn_rate(df, 'created_at').sort_values(by='churn_rate', ascen
```

Out[ ]:

dt_ano_mes	active	canceled	paused	churn_rate
2016-02-01	58	8	4	12.90
2019-10-01	145	16	18	9.82
2020-04-01	152	16	13	9.70
2020-11-01	116	12	14	9.23
2016-07-01	150	14	13	8.59
...	...	...	...	...
2018-10-01	141	4	15	2.56
2020-12-01	148	4	11	2.52
2019-07-01	150	4	12	2.47
2018-07-01	149	4	17	2.41
2017-08-01	144	3	16	1.88

61 rows × 4 columns

Analogamente ao caso anterior, agora observaremos os estados que mais ocorreram os cancelamentos.

In [ ]:
```
calculate_churn_rate(df, 'created_at', 'state').sort_values(by='churn_rate
```

Out[ ]:

state	active	canceled	paused	churn_rate
SE	305	24	38	7.00
RS	317	25	40	7.00
MT	308	24	36	6.98
MA	303	23	28	6.95
PA	284	22	45	6.69
AL	301	22	30	6.65
AM	326	23	31	6.44
TO	356	24	29	6.23
RR	327	23	46	6.17
PE	309	20	38	5.76
BA	314	20	34	5.75
MS	305	20	49	5.65
GO	344	21	38	5.50
MG	320	19	34	5.37
CE	325	19	34	5.29
SP	308	16	30	4.73
AC	297	16	44	4.69
RO	329	17	35	4.67
PR	330	17	38	4.62
RN	306	15	32	4.44
RJ	323	16	38	4.43
DF	306	15	36	4.39
AP	334	15	32	4.10
PB	331	14	27	3.91
ES	303	12	37	3.53
PI	290	11	37	3.36
SC	323	12	35	3.35

Mais uma vez, analogamente ao caso anterior, agora observaremos a taxa de cancelamento por meio de marketing, ordenando-os de maneira decrescente.

In [ ]:
```
calculate_churn_rate(df, 'created_at', 'marketing_source').sort_values(by=
```

Out[ ]:

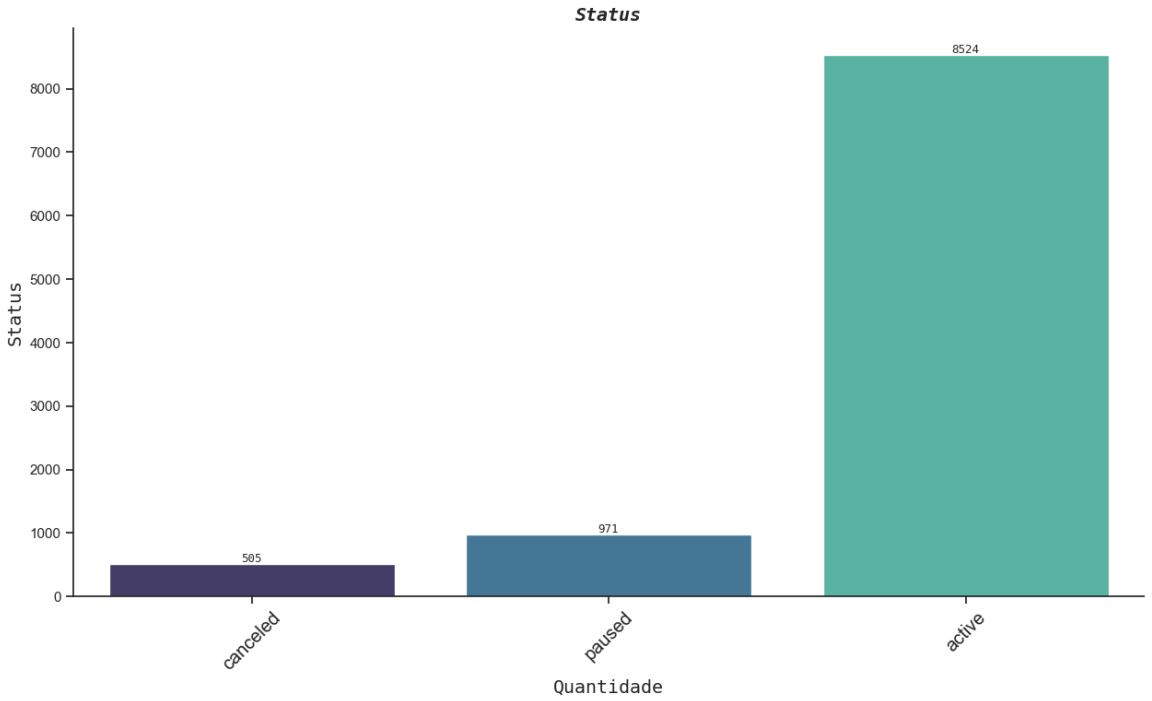
marketing_source	active	canceled	paused	churn_rate
none	439	34	56	6.87
telegram_whatsapp	914	66	88	6.59
organic_search	3118	196	385	5.60
paid_search	1307	70	149	4.81
direct	1872	96	181	4.68
crm	874	43	112	4.36

Análise Gráfica

In [ ]:
```
def create_count_plot(df=df, column: str = '', title: str = '', xlabel: st
    _, ax = plt.subplots(figsize=(15, 8))
    sns.countplot(data=df, x=column, order=df[column].value_counts().index
    for value in ax.containers:
        ax.bar_label(value, label_type='edge', fontsize=9, family='monospa
    ax.set_xlabel(ylabel, fontsize='large', family='monospace')
    ax.set_ylabel(xlabel, fontsize='large', family='monospace')
    ax.set_title(title, fontsize='large', fontweight='bold', style='italic
    ax.tick_params(axis='x', labelsize='large', rotation=45)
    plt.show()
```

Vemos a grande quantidade de clientes ativos no programa da **PetLove**.

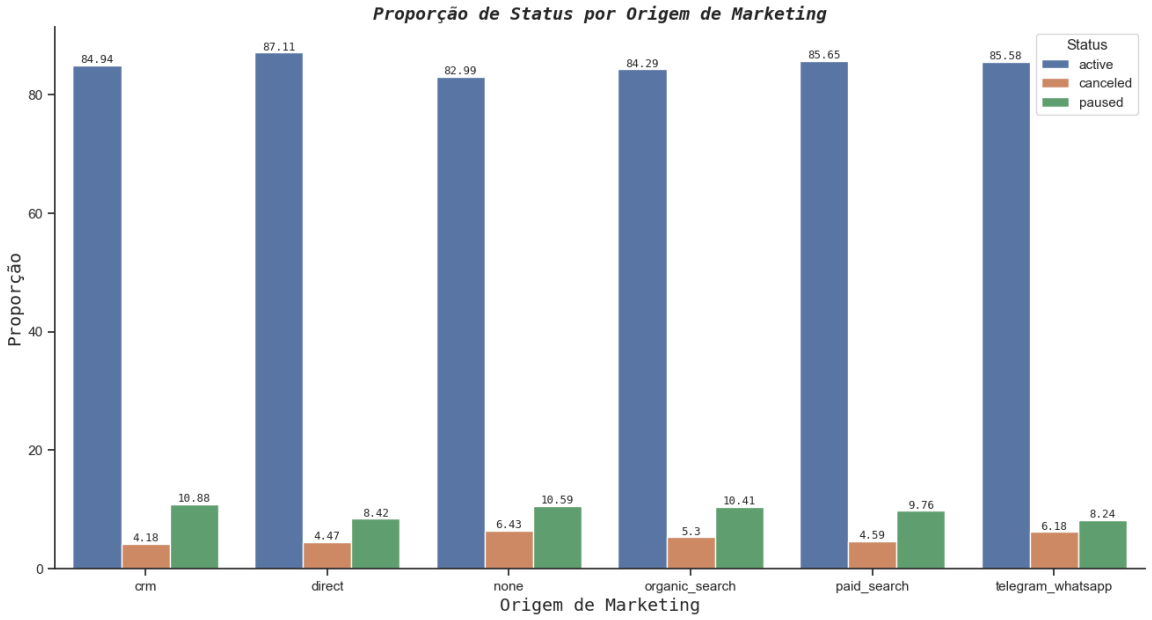
In [ ]:
```
create_count_plot(df, 'status', 'Status', 'Status', 'Quantidade')
```



In [ ]:
```
def create_bar_plot(df: pd.DataFrame, column: str, title: str, x_label: st
    _, ax = plt.subplots(figsize=(16, 8))
    df_ = df.groupby(column)['status'].value_counts(normalize=True).rename
    df_.prop = np.round(df_.prop * 100, 2)
    sns.barplot(data=df_, x=column, y='prop', hue='status', ax=ax)
    ax.legend(title="Status")
    for value in ax.containers:
        ax.bar_label(value, label_type='edge', fontsize=9, family='monospa
    ax.set_xlabel(x_label, fontsize='large', family='monospace')
    ax.set_ylabel('Proporção', fontsize='large', family='monospace')
    ax.set_title(title, fontsize='large', fontweight='bold', style='italic
```

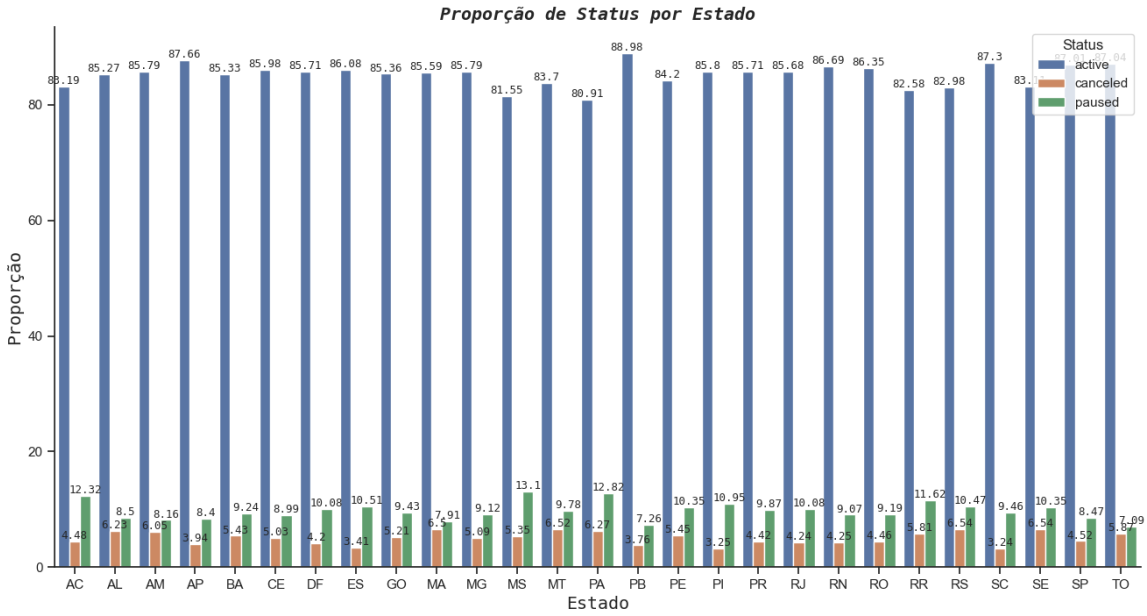
A partir do marketing vamos analisar a proporção de clientes ativos, pausados e cancelados.

In [ ]:
```
create_bar_plot(df, 'marketing_source', 'Proporção de Status por Origem de
plt.show()
```



Analogamente ao caso anterior, agora observaremos a proporção de clientes ativos, pausados e cancelados por estado, infelizmente a visualização foi comprometida pela quantidade de itens.

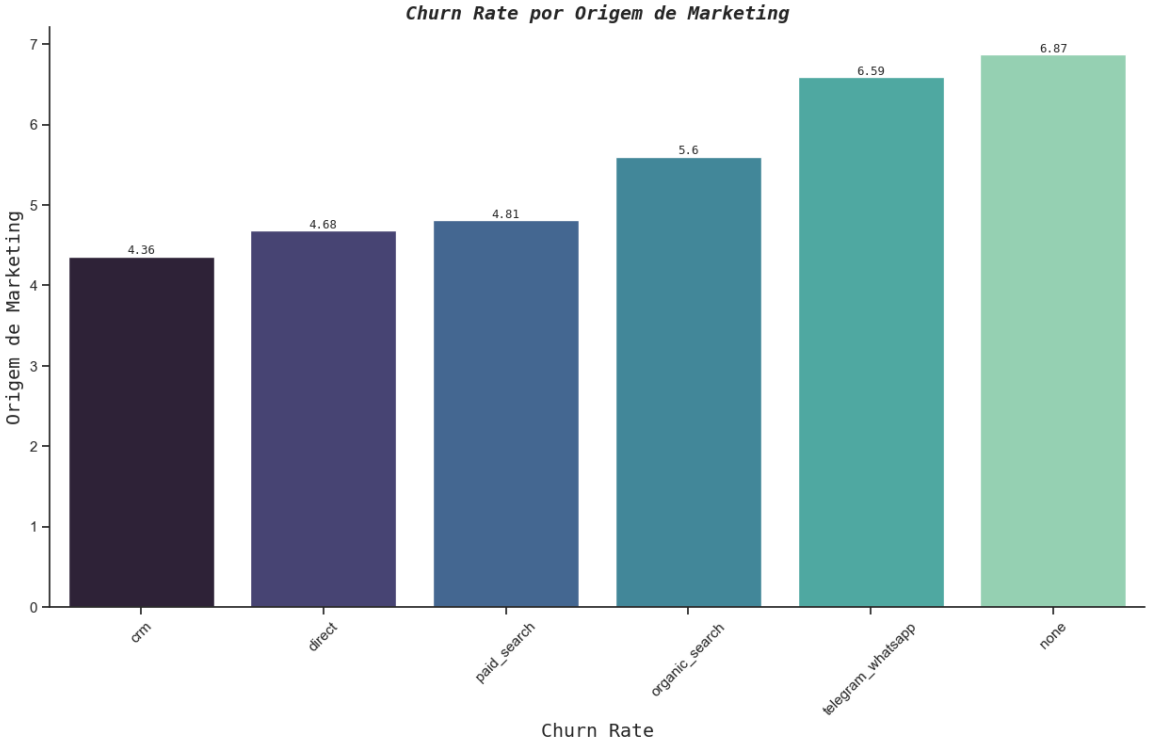
In [ ]:
```
create_bar_plot(df, 'state', 'Proporção de Status por Estado', 'Estado')
plt.show()
```



In [ ]:
```
def create_bar_plot(df=df, y: str = '', title: str = '', xlabel: str = ''
    _, ax = plt.subplots(figsize=(15, 8))
    df = df.sort_values(by=y, ascending=True)[y]
    sns.barplot(data=df, x=df.index, y=df.values, palette=PALETTE)
    for value in ax.containers:
        ax.bar_label(value, label_type='edge', fontsize=9, family='monospa
    ax.set_xlabel(ylabel, fontsize='large', family='monospace')
    ax.set_ylabel(xlabel, fontsize='large', family='monospace')
    ax.set_title(title, fontsize='large', fontweight='bold', style='italic
    ax.set_xticks(np.arange(df.index.shape[0]))
    ax.set_xticklabels(df.index, rotation=rot)
    plt.show()
```

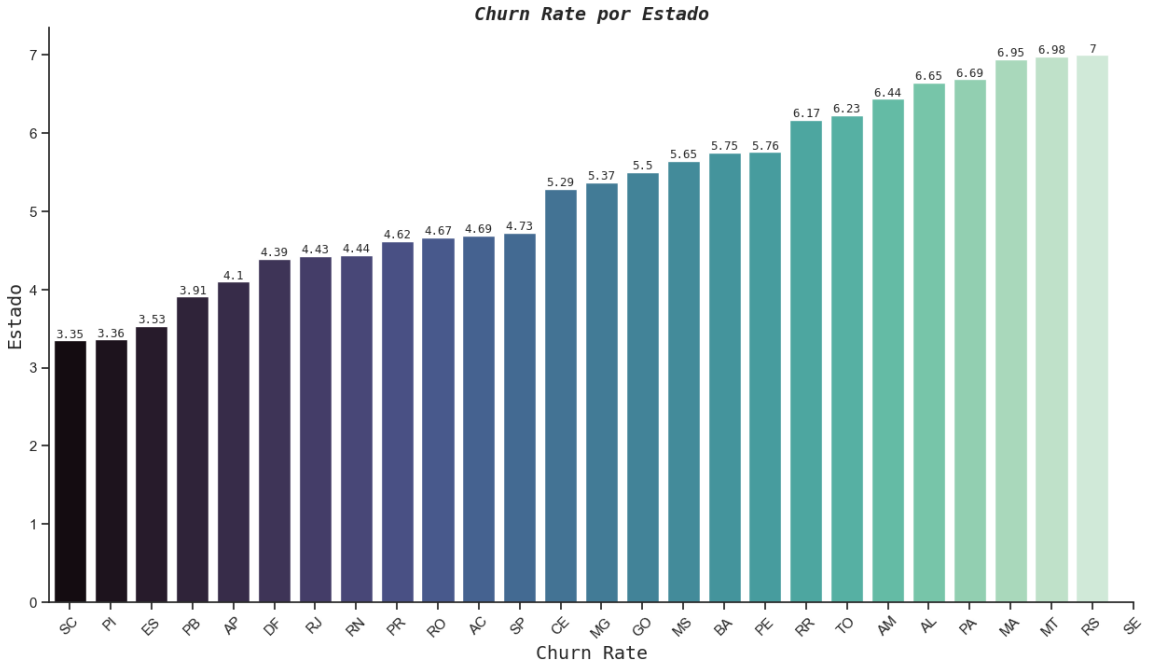
Como o foco do case é a análise de churn, vamos analisar a proporção de taxa de churn por meio de marketing.

In [ ]:
```
df_grouped = calculate_churn_rate(df, 'created_at', 'marketing_source')
create_bar_plot(df_grouped, y='churn_rate', title='Churn Rate por Origem c
```



Desta vez a análise será feita por estado.

In [ ]:
```
df_grouped = calculate_churn_rate(df, 'created_at', 'state')
create_bar_plot(df_grouped, y='churn_rate', title='Churn Rate por Estado'
```

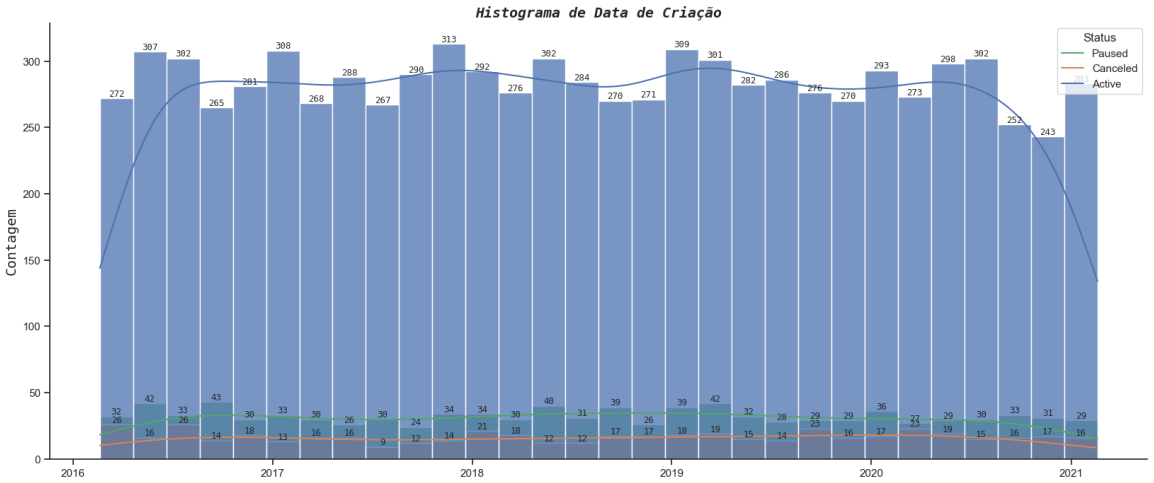


Análise de distribuição com histogramas

In [ ]:
```
def create_histogram(df: pd.DataFrame, column: str, title: str, bins: int
    _, ax = plt.subplots(figsize=(20, 8))
    sns.histplot(data=df, x=column, hue='status', bins=bins, kde=True, alp
    ax.legend(title="Status", loc="upper right", labels=["Paused", "Cancel
    for value in ax.containers:
        ax.bar_label(value, label_type='edge', fontsize=9, family='monospa
    ax.set_xlabel(None)
    ax.set_ylabel('Contagem', fontsize='large', family='monospace')
    ax.set_title(title, fontsize='large', fontweight='bold', style='italic
```

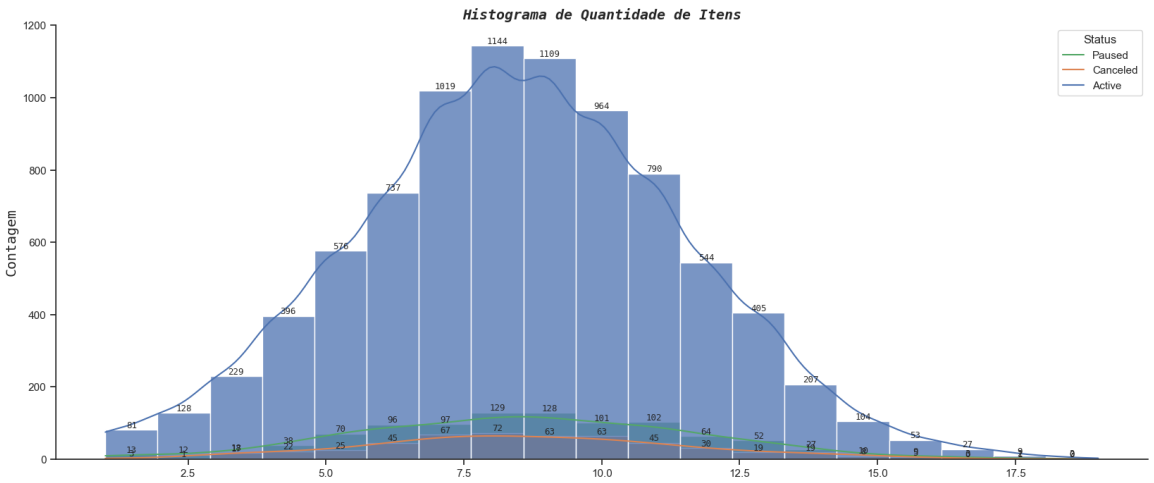
Análise da distribuição dos casos de cancelamento pelos anos (considerando o ano de criação da conta).

In [ ]:
```
create_histogram(df, 'created_at', 'Histograma de Data de Criação', 30)
plt.show()
```



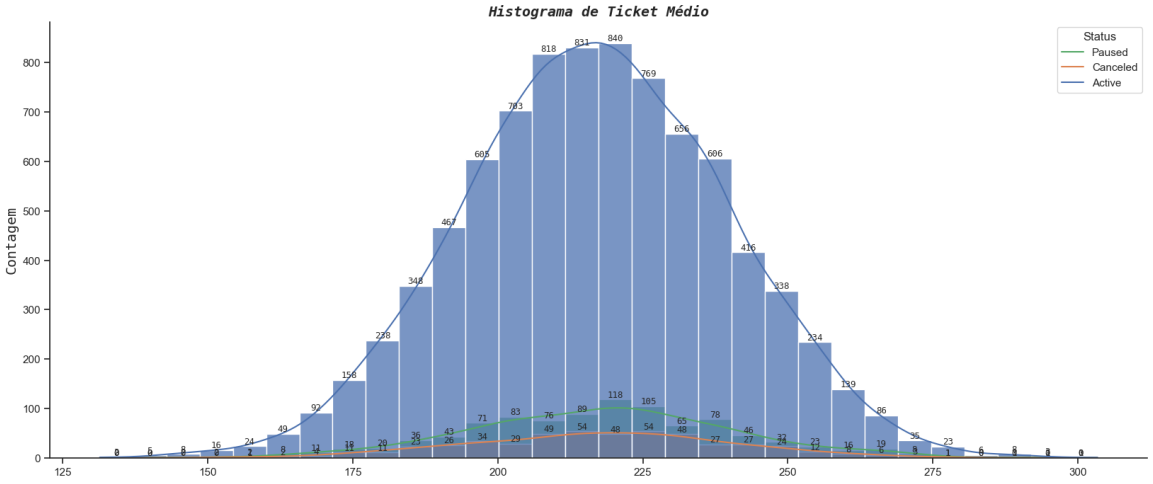
Análise da distribuição dos casos de cancelamento pela quantidade de itens comprados (vemos que se comporta exatamente como uma distribuição normal).

In [ ]:
```
create_histogram(df, 'items_quantity', 'Histograma de Quantidade de Itens
plt.show()
```



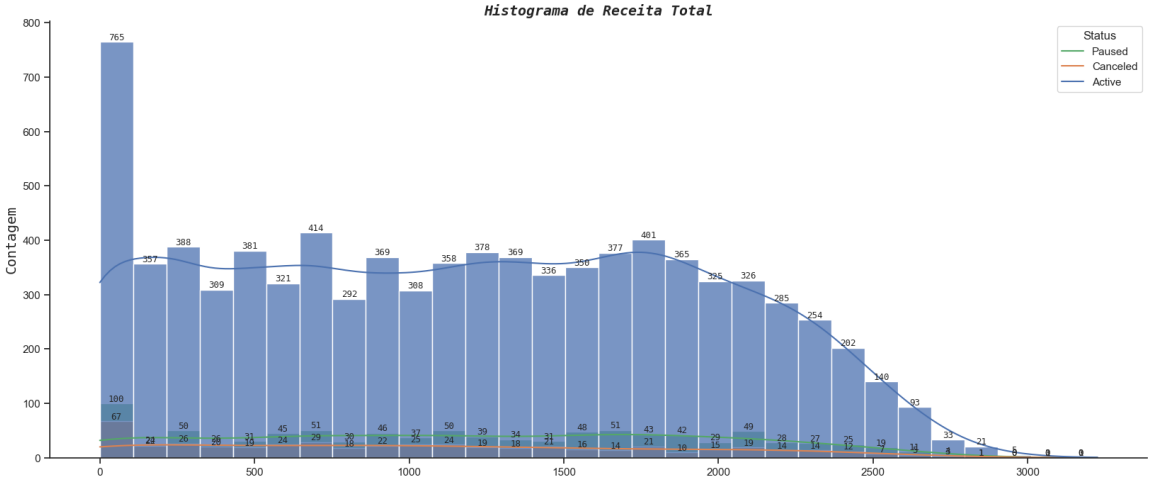
Análise da distribuição dos casos de cancelamento pelo preço médio dos tickets (vemos que se comporta exatamente como uma distribuição normal).

In [ ]:
```
create_histogram(df, 'average_ticket', 'Histograma de Ticket Médio', 30)
plt.show()
```



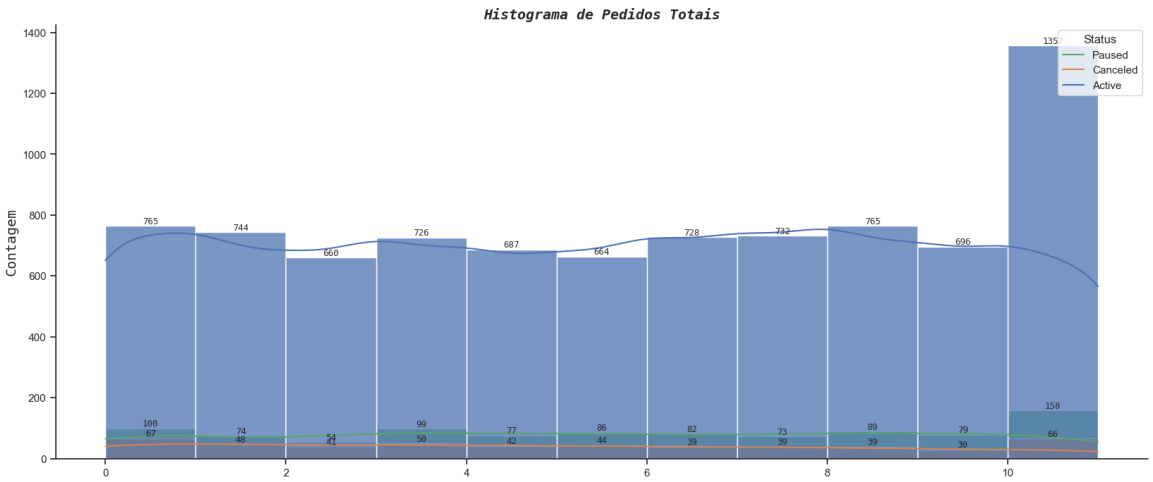
Análise do histograma pela receita total gasta, desta vez vemos que a distribuição não se comporta como uma normal.

In [ ]:
```
create_histogram(df, 'all_revenue', 'Histograma de Receita Total', 30)
plt.show()
```



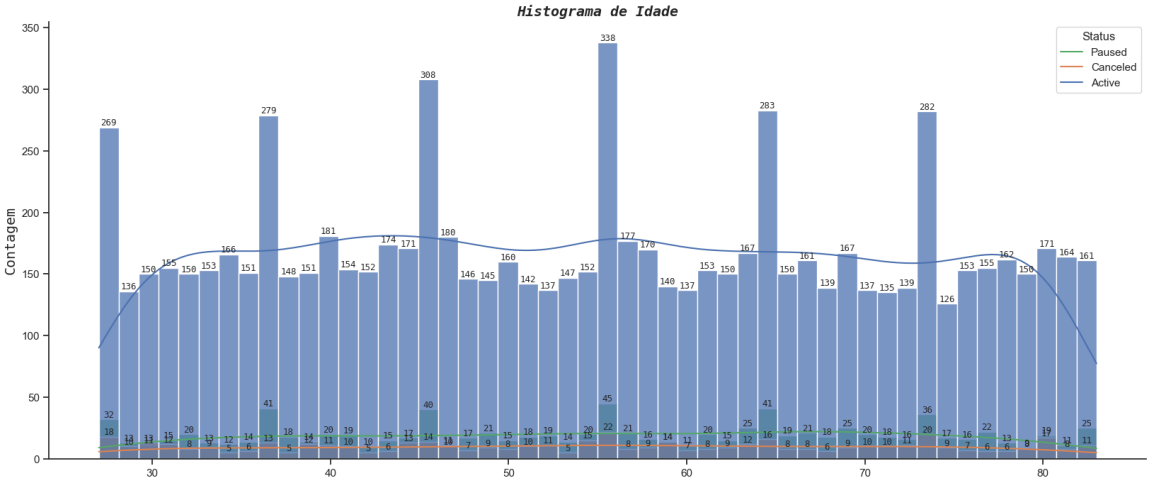
Análise do histograma por total de pedidos, vemos que se comporta de maneira constante.

In [ ]:
```
create_histogram(df, 'all_orders', 'Histograma de Pedidos Totais', 11)
plt.show()
```



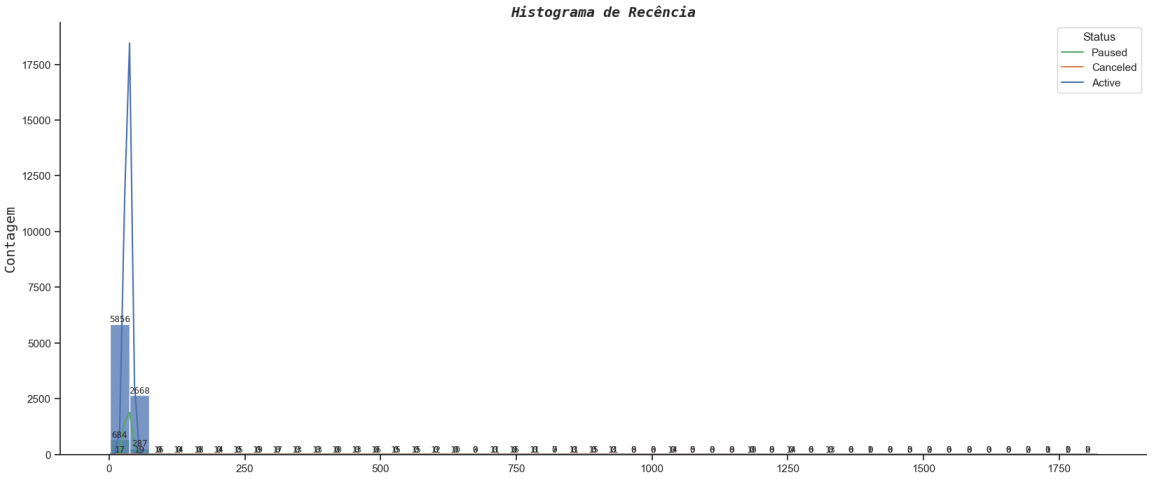
Análise do histograma por idade, vemos picos de idades com intervalos predefinidos.

In [ ]:
```
create_histogram(df, 'age', 'Histograma de Idade', 50)
plt.show()
```



Observamos que os clientes costumam comprar de maneira recorrente.

In [ ]:
```
create_histogram(df, 'recency', 'Histograma de Recência', 50)
plt.show()
```



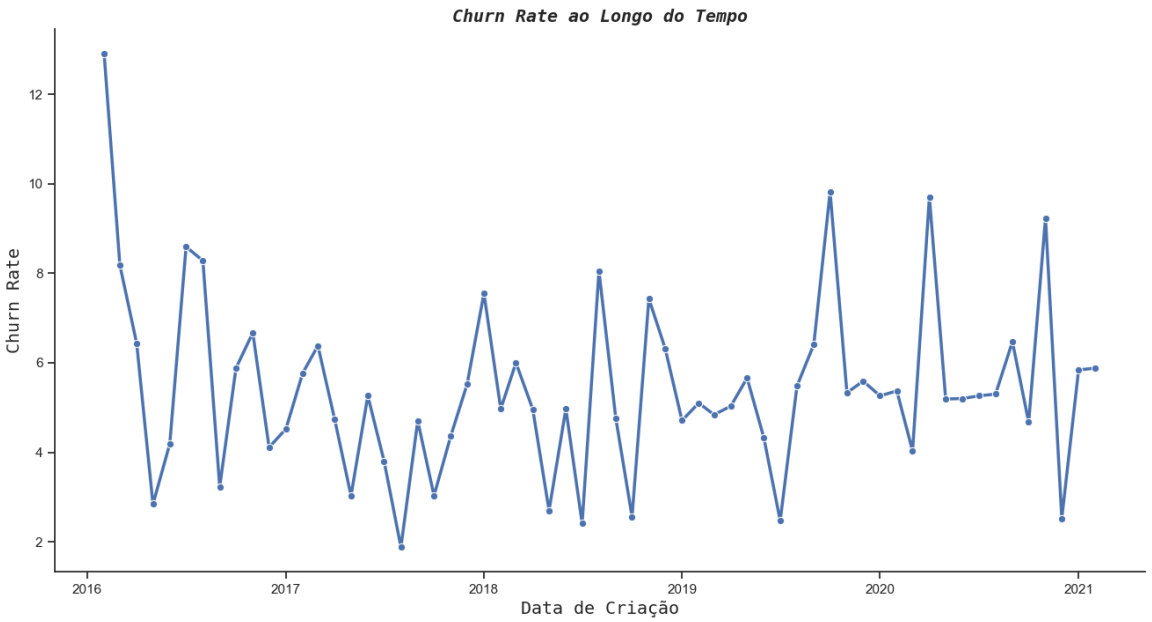
Análise Temporal

In [ ]:
```
def line_plot_for_temporal(df: pd.DataFrame, column: str, title: str, x_la
    _, ax = plt.subplots(figsize=(16, 8))
    sns.lineplot(data=df, x=df.index, y=column, marker='o', linewidth=2.5
    ax.set_xlabel(x_label, fontsize='large', family='monospace')
    ax.set_ylabel(y_label, fontsize='large', family='monospace')
    ax.set_title(title, fontsize='large', fontweight='bold', style='italic
```

Abaixo temos a análise temporal dos casos de churn, vemos que a quantidade de churn diminuiu ao longo dos anos.

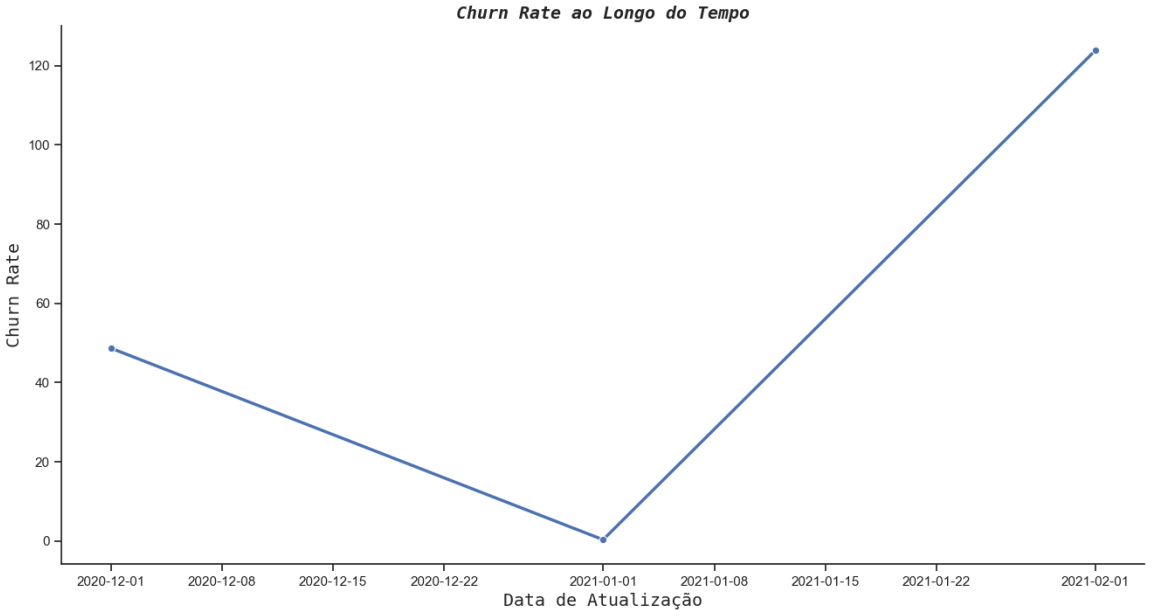
Lembrando que essa análise é baseada no ano de criação da conta.

In [ ]:
```
df_temporal = calculate_churn_rate(df, 'created_at')
line_plot_for_temporal(df_temporal, 'churn_rate', 'Churn Rate ao Longo do
plt.show()
```



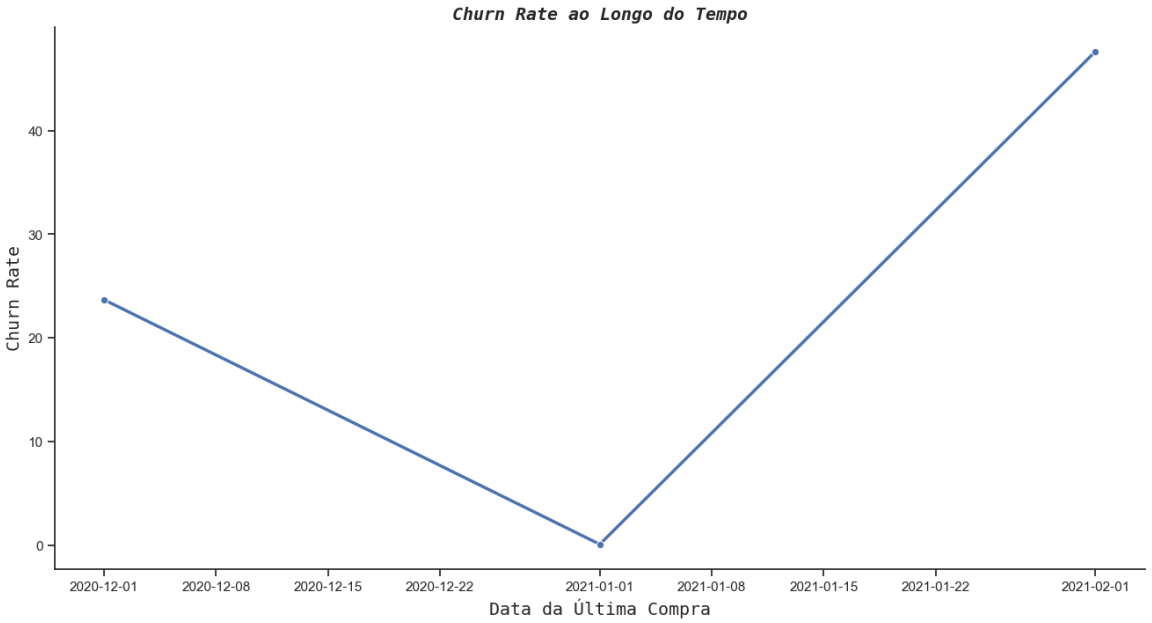
Vemos que a quantidade de churn diminuiu razoavelmente até o seu pico mais baixo (2021/01/01), então cresceu de maneira constante até o pico mais alto (2021/02/01).

In [ ]:
```
df_temporal = calculate_churn_rate(df, 'updated_at')
line_plot_for_temporal(df_temporal, 'churn_rate', 'Churn Rate ao Longo do
plt.show()
```



Como dito anteriormente nas tabulações, vamos que o tempo de atualização de conta e data de ultima compra estão altamente correlacionados.

In [ ]:
```
df_temporal = calculate_churn_rate(df, 'last_date_purchase')
line_plot_for_temporal(df_temporal, 'churn_rate', 'Churn Rate ao Longo do
plt.show()
```


Churn Rate ao Longo do Tempo
Churn Rate
40
30
20
10
0
2020-12-01
2020-12-08
2020-12-15
2020-12-22
2021-01-01
2021-01-08
2021-01-15
2021-01-22
2021-02-01
Data da Última Compra

Conclusão

Análise dos dados

A importância da análise da taxa de cancelamento é muito grande para uma empresa, pois, é através dela que é possível identificar os clientes que estão insatisfeitos com o serviço prestado e que possuem uma alta probabilidade de cancelar a assinatura.

Conseguimos identificar alguns padrões interessantes que deveriam ser melhor explorados pela empresa, como por exemplo, a alta taxa de cancelamento de clientes provenientes do canal de marketing "None" e "Telegram / Whatsapp".

Além disso alguns estados possuem uma taxa de cancelamento muito alta, como por exemplo, os estados de SE, RS, MT, MA, PA, AL, AM, TO, RR, que possuem uma taxa de cancelamento acima de 6%. Uma possível causa é que 7 dos 9 estados citados fazem parte das regiões Norte e Nordeste.

Além disso podemos observar um padrão da taxa de cancelamento relacionado a épocas do ano, onde os meses referentes ao meio do ano (Maio, Junho e Julho) possuem baixa taxa de cancelamento.

Apesar do canal de marketing Organic Search ser o que mais realiza a captação de clientes, ele possui uma taxa de cancelamento de 5.6%, sendo a terceira maior.

E por fim, é interessante salientar o período referente a pandemia, onde houve um crescimento exobritante de clientes, mas a taxa de cancelamento cresceu junto, o que pode ser um reflexo do período de incertezas que estavamos vivendo, pois conforme o tempo passou a taxa de cancelamento diminuiu.

Sugestões

- Avaliar a realização de pesquisas de satisfação com os clientes para identificar os principais motivos de cancelamento.
- Melhorar a experiência do usuário no site, para que o cliente consiga encontrar os produtos que deseja com mais facilidade.
- Investir em estratégias de marketing para o canal Organic Search, onde possui o maior número de clientes captados, buscando manter esses clientes satisfeitos.
- Identificar os problemas com os clientes captados via Telegram e Whatsapp.
- Um possível problema relacionado aos estados pode ser a logística, onde os produtos podem demorar mais para chegar, ou até mesmo chegar com avarias. Além do custo alto de frete para esses estados.
- Será interessante utilizar uma abordagem temporal para medição/predição de cancelamentos, onde será possível identificar os clientes que estão com uma alta probabilidade de cancelar a assinatura.