Análise Exploratória

Compreensão inicial dos dados, identificando variáveis, tipos e distribuições. Isso envolve carregar os dados, inspecionar as primeiras linhas, verificar tipos de dados, resumir estatísticas descritivas e lidar com valores ausentes.

Instalação das Bibliotecas

In [8]:
```
%pip install pandas matplotlib seaborn
```

```
Requirement already satisfied: pandas in c:\python312\lib\site-packages (2.2.2)No
te: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Requirement already satisfied: matplotlib in c:\python312\lib\site-packages (3.9.
2)
Requirement already satisfied: seaborn in c:\python312\lib\site-packages (0.13.2)
Requirement already satisfied: numpy>=1.26.0 in c:\python312\lib\site-packages (f
rom pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\python312\lib\site-pa
ckages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\python312\lib\site-packages (fr
om pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\python312\lib\site-packages
(from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\python312\lib\site-packages
(from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\python312\lib\site-packages (fr
om matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\python312\lib\site-package
s (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\python312\lib\site-package
s (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in c:\users\salom\appdata\roaming
\python\python312\site-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in c:\python312\lib\site-packages (from
matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\python312\lib\site-packages
(from matplotlib) (3.1.4)
Requirement already satisfied: six>=1.5 in c:\python312\lib\site-packages (from p
ython-dateutil>=2.8.2->pandas) (1.16.0)
```

Importação das tabelas

In [9]:
```
import pandas as pd

# Carregar os datasets
customers = pd.read_csv("olist_customers_dataset.csv")
geolocation = pd.read_csv("olist_geolocation_dataset.csv")
order_items = pd.read_csv("olist_order_items_dataset.csv")
order_payments = pd.read_csv("olist_order_payments_dataset.csv")
order_reviews = pd.read_csv("olist_order_reviews_dataset.csv")
orders = pd.read_csv("olist_orders_dataset.csv")
```

```
products = pd.read_csv("olist_products_dataset.csv")
sellers = pd.read_csv("olist_sellers_dataset.csv")
product_category_translation = pd.read_csv("product_category_name_translation.cs

print("-----------------------------------------------------------------------
print(customers.head())  # Primeiras 5 linhas
print(customers.info())   # Informações sobre as colunas
print(customers.describe()) # Estatísticas descritivas
print("-----------------------------------------------------------------------

print("-----------------------------------------------------------------------
print(geolocation.head())  # Primeiras 5 linhas
print(geolocation.info())   # Informações sobre as colunas
print(geolocation.describe()) # Estatísticas descritivas
print("-----------------------------------------------------------------------

print("-----------------------------------------------------------------------
print(order_items.head())  # Primeiras 5 linhas
print(order_items.info())   # Informações sobre as colunas
print(order_items.describe()) # Estatísticas descritivas
print("-----------------------------------------------------------------------

print("-----------------------------------------------------------------------
print(order_payments.head())  # Primeiras 5 linhas
print(order_payments.info())   # Informações sobre as colunas
print(order_payments.describe()) # Estatísticas descritivas
print("-----------------------------------------------------------------------

print("-----------------------------------------------------------------------
print(order_reviews.head())  # Primeiras 5 linhas
print(order_reviews.info())   # Informações sobre as colunas
print(order_reviews.describe()) # Estatísticas descritivas
print("-----------------------------------------------------------------------

print("-----------------------------------------------------------------------
print(orders.head())  # Primeiras 5 linhas
print(orders.info())   # Informações sobre as colunas
print(orders.describe()) # Estatísticas descritivas
print("-----------------------------------------------------------------------

print("-----------------------------------------------------------------------
print(products.head())  # Primeiras 5 linhas
print(products.info())   # Informações sobre as colunas
print(products.describe()) # Estatísticas descritivas
print("-----------------------------------------------------------------------

print("-----------------------------------------------------------------------
print(sellers.head())  # Primeiras 5 linhas
print(sellers.info())   # Informações sobre as colunas
print(sellers.describe()) # Estatísticas descritivas
print("-----------------------------------------------------------------------

print("-----------------------------------------------------------------------
print(product_category_translation.head())  # Primeiras 5 linhas
print(product_category_translation.info())   # Informações sobre as colunas
print(product_category_translation.describe()) # Estatísticas descritivas
print("-----------------------------------------------------------------------
```

```
----------------------------------------------------------------------------
                        customer_id                customer_unique_id  \
0  06b8999e2fba1a1fbc88172c00ba8bc7  861eff4711a542e4b93843c6dd7febb0
1  18955e83d337fd6b2def6b18a428ac77  290c77bc529b7ac935b93aa66c333dc3
2  4e7b3e00288586ebd08712fdd0374a03  060e732b5b29e8181a18229c7b0b2b5e
3  b2b6027bc5c5109e529d4dc6358b12c3  259dac757896d24d7702b9acbbff3f3c
4  4f2d8ab171c80ec8364f7c12e35b23ad  345ecd01c38d18a9036ed96c73b8d066

   customer_zip_code_prefix          customer_city customer_state
0                     14409                 franca             SP
1                      9790  sao bernardo do campo             SP
2                      1151              sao paulo             SP
3                      8775        mogi das cruzes             SP
4                     13056               campinas             SP
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 5 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   customer_id               99441 non-null  object
 1   customer_unique_id        99441 non-null  object
 2   customer_zip_code_prefix  99441 non-null  int64 
 3   customer_city             99441 non-null  object
 4   customer_state            99441 non-null  object
dtypes: int64(1), object(4)
memory usage: 3.8+ MB
None
       customer_zip_code_prefix
count              99441.000000
mean               35137.474583
std                29797.938996
min                 1003.000000
25%                11347.000000
50%                24416.000000
75%                58900.000000
max                99990.000000
----------------------------------------------------------------------------
----------------------------------------------------------------------------
   geolocation_zip_code_prefix  geolocation_lat  geolocation_lng  \
0                         1037       -23.545621       -46.639292
1                         1046       -23.546081       -46.644820
2                         1046       -23.546129       -46.642951
3                         1041       -23.544392       -46.639499
4                         1035       -23.541578       -46.641607

  geolocation_city geolocation_state
0        sao paulo                SP
1        sao paulo                SP
2        sao paulo                SP
3        sao paulo                SP
4        sao paulo                SP
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000163 entries, 0 to 1000162
Data columns (total 5 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   geolocation_zip_code_prefix  1000163 non-null  int64  
 1   geolocation_lat              1000163 non-null  float64
 2   geolocation_lng              1000163 non-null  float64
 3   geolocation_city             1000163 non-null  object 
```

```
 4   geolocation_state            1000163 non-null  object
dtypes: float64(2), int64(1), object(2)
memory usage: 38.2+ MB
None
       geolocation_zip_code_prefix  geolocation_lat  geolocation_lng
count                 1.000163e+06     1.000163e+06     1.000163e+06
mean                  3.657417e+04    -2.117615e+01    -4.639054e+01
std                   3.054934e+04     5.715866e+00     4.269748e+00
min                   1.001000e+03    -3.660537e+01    -1.014668e+02
25%                   1.107500e+04    -2.360355e+01    -4.857317e+01
50%                   2.653000e+04    -2.291938e+01    -4.663788e+01
75%                   6.350400e+04    -1.997962e+01    -4.376771e+01
max                   9.999000e+04     4.506593e+01     1.211054e+02
----------------------------------------------------------------------------
----------------------------------------------------------------------------
                           order_id  order_item_id  \
0  00010242fe8c5a6d1ba2dd792cb16214              1
1  00018f77f2f0320c557190d7a144bdd3              1
2  000229ec398224ef6ca0657da4fc703e              1
3  00024acbcdf0a6daa1e931b038114c75              1
4  00042b26cf59d7ce69dfabb4e55b4fd9              1

                         product_id                         seller_id  \
0  4244733e06e7ecb4970a6e2683c13e61  48436dade18ac8b2bce089ec2a041202
1  e5f2d52b802189ee658865ca93d83a8f  dd7ddc04e1b6c2c614352b383efe2d36
2  c777355d18b72b67abbeef9df44fd0fd  5b51032eddd242adc84c38acab88f23d
3  7634da152a4610f1595efa32f14722fc  9d7a1d34a5052409006425275ba1c2b4
4  ac6c3623068f30de03045865e4e10089  df560393f3a51e74553ab94004ba5c87

   shipping_limit_date   price  freight_value
0  2017-09-19 09:45:35   58.90          13.29
1  2017-05-03 11:05:13  239.90          19.93
2  2018-01-18 14:48:30  199.00          17.87
3  2018-08-15 10:10:18   12.99          12.79
4  2017-02-13 13:57:51  199.90          18.14
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112650 entries, 0 to 112649
Data columns (total 7 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   order_id             112650 non-null  object
 1   order_item_id        112650 non-null  int64
 2   product_id           112650 non-null  object
 3   seller_id            112650 non-null  object
 4   shipping_limit_date  112650 non-null  object
 5   price                112650 non-null  float64
 6   freight_value        112650 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 6.0+ MB
None
       order_item_id          price  freight_value
count  112650.000000  112650.000000  112650.000000
mean        1.197834     120.653739      19.990320
std         0.705124     183.633928      15.806405
min         1.000000       0.850000       0.000000
25%         1.000000      39.900000      13.080000
50%         1.000000      74.990000      16.260000
75%         1.000000     134.900000      21.150000
max        21.000000    6735.000000     409.680000
----------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------
                           order_id  payment_sequential payment_type  \
0  b81ef226f3fe1789b1e8b2acac839d17                   1  credit_card
1  a9810da82917af2d9aefd1278f1dcfa0                   1  credit_card
2  25e8ea4e93396b6fa0d3dd708e76c1bd                   1  credit_card
3  ba78997921bbcdc1373bb41e913ab953                   1  credit_card
4  42fdf880ba16b47b59251dd489d4441a                   1  credit_card

   payment_installments  payment_value
0                     8          99.33
1                     1          24.39
2                     1          65.71
3                     8         107.78
4                     2         128.45
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103886 entries, 0 to 103885
Data columns (total 5 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   order_id              103886 non-null  object 
 1   payment_sequential    103886 non-null  int64  
 2   payment_type          103886 non-null  object 
 3   payment_installments  103886 non-null  int64  
 4   payment_value         103886 non-null  float64
dtypes: float64(1), int64(2), object(2)
memory usage: 4.0+ MB
None
       payment_sequential  payment_installments  payment_value
count       103886.000000         103886.000000  103886.000000
mean             1.092679              2.853349     154.100380
std              0.706584              2.687051     217.494064
min              1.000000              0.000000       0.000000
25%              1.000000              1.000000      56.790000
50%              1.000000              1.000000     100.000000
75%              1.000000              4.000000     171.837500
max             29.000000             24.000000   13664.080000
--------------------------------------------------------------------------
--------------------------------------------------------------------------
                          review_id                          order_id  \
0  7bc2406110b926393aa56f80a40eba40  73fc7af87114b39712e6da79b0a377eb
1  80e641a11e56f04c1ad469d5645fdfde  a548910a1c6147796b98fdf73dbeba33
2  228ce5500dc1d8e020d8d1322874b6f0  f9e4b658b201a9f2ecdecbb34bed034b
3  e64fb393e7b32834bb789ff8bb30750e  658677c97b385a9be170737859d3511b
4  f7c4243c7fe1938f181bec41a392bdeb  8e6bfb81e283fa7e4f11123a3fb894f1

   review_score review_comment_title  \
0             4                  NaN
1             5                  NaN
2             5                  NaN
3             5                  NaN
4             5                  NaN

                              review_comment_message review_creation_date  \
0                                                NaN  2018-01-18 00:00:00
1                                                NaN  2018-03-10 00:00:00
2                                                NaN  2018-02-17 00:00:00
3              Recebi bem antes do prazo estipulado.  2018-04-21 00:00:00
4  Parabéns lojas lannister adorei comprar pela I...  2018-03-01 00:00:00

  review_answer_timestamp
```

```
0     2018-01-18 21:46:59
1     2018-03-11 03:05:13
2     2018-02-18 14:36:24
3     2017-04-21 22:02:06
4     2018-03-02 10:26:53
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99224 entries, 0 to 99223
Data columns (total 7 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   review_id                99224 non-null  object
 1   order_id                 99224 non-null  object
 2   review_score             99224 non-null  int64
 3   review_comment_title     11568 non-null  object
 4   review_comment_message   40977 non-null  object
 5   review_creation_date     99224 non-null  object
 6   review_answer_timestamp  99224 non-null  object
dtypes: int64(1), object(6)
memory usage: 5.3+ MB
None
       review_score
count  99224.000000
mean       4.086421
std        1.347579
min        1.000000
25%        4.000000
50%        5.000000
75%        5.000000
max        5.000000
----------------------------------------------------------------------------
----------------------------------------------------------------------------
                           order_id                       customer_id  \
0  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d
1  53cdb2fc8bc7dce0b6741e2150273451  b0830fb4747a6c6d20dea0b8c802d7ef
2  47770eb9100c2d0c44946d9cf07ec65d  41ce2a54c0b03bf3443c3d931a367089
3  949d5b44dbf5de918fe9c16f97b45f8a  f88197465ea7920adcdbec7375364d82
4  ad21c59c0840e6cb83a9ceb5573f8159  8ab97904e6daea8866dbdbc4fb7aad2c

  order_status order_purchase_timestamp    order_approved_at  \
0    delivered      2017-10-02 10:56:33  2017-10-02 11:07:15
1    delivered      2018-07-24 20:41:37  2018-07-26 03:24:27
2    delivered      2018-08-08 08:38:49  2018-08-08 08:55:23
3    delivered      2017-11-18 19:28:06  2017-11-18 19:45:59
4    delivered      2018-02-13 21:18:39  2018-02-13 22:20:29

  order_delivered_carrier_date order_delivered_customer_date  \
0          2017-10-04 19:55:00           2017-10-10 21:25:13
1          2018-07-26 14:31:00           2018-08-07 15:27:45
2          2018-08-08 13:50:00           2018-08-17 18:06:29
3          2017-11-22 13:39:59           2017-12-02 00:28:42
4          2018-02-14 19:46:34           2018-02-16 18:17:02

  order_estimated_delivery_date
0           2017-10-18 00:00:00
1           2018-08-13 00:00:00
2           2018-09-04 00:00:00
3           2017-12-15 00:00:00
4           2018-02-26 00:00:00
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
```

```
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   order_id                       99441 non-null  object
 1   customer_id                    99441 non-null  object
 2   order_status                   99441 non-null  object
 3   order_purchase_timestamp       99441 non-null  object
 4   order_approved_at              99281 non-null  object
 5   order_delivered_carrier_date   97658 non-null  object
 6   order_delivered_customer_date  96476 non-null  object
 7   order_estimated_delivery_date  99441 non-null  object
dtypes: object(8)
memory usage: 6.1+ MB
None
                                order_id                       customer_id  \
count                              99441                             99441   
unique                             99441                             99441   
top     e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d   
freq                                   1                                 1   

       order_status order_purchase_timestamp    order_approved_at  \
count         99441                    99441                99281   
unique            8                    98875                90733   
top       delivered      2018-04-11 10:48:14  2018-02-27 04:31:10   
freq          96478                        3                    9   

       order_delivered_carrier_date order_delivered_customer_date  \
count                         97658                         96476   
unique                        81018                         95664   
top             2018-05-09 15:48:00           2018-05-08 23:38:46   
freq                             47                             3   

       order_estimated_delivery_date  
count                          99441  
unique                           459  
top              2017-12-20 00:00:00  
freq                             522  
----------------------------------------------------------------------------
----------------------------------------------------------------------------
                         product_id  product_category_name  \
0  1e9e8ef04dbcff4541ed26657ea517e5             perfumaria   
1  3aa071139cb16b67ca9e5dea641aaa2f                  artes   
2  96bd76ec8810374ed1b65e291975717f          esporte_lazer   
3  cef67bcfe19066a932b7673e239eb23d                  bebes   
4  9dc1a7de274444849c219cff195d0b71  utilidades_domesticas   

   product_name_lenght  product_description_lenght  product_photos_qty  \
0                 40.0                       287.0                 1.0   
1                 44.0                       276.0                 1.0   
2                 46.0                       250.0                 1.0   
3                 27.0                       261.0                 1.0   
4                 37.0                       402.0                 4.0   

   product_weight_g  product_length_cm  product_height_cm  product_width_cm  
0             225.0               16.0               10.0              14.0  
1            1000.0               30.0               18.0              20.0  
2             154.0               18.0                9.0              15.0  
3             371.0               26.0                4.0              26.0  
4             625.0               20.0               17.0              13.0  
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 32951 entries, 0 to 32950
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   product_id                  32951 non-null  object 
 1   product_category_name       32341 non-null  object 
 2   product_name_lenght         32341 non-null  float64
 3   product_description_lenght  32341 non-null  float64
 4   product_photos_qty          32341 non-null  float64
 5   product_weight_g            32949 non-null  float64
 6   product_length_cm           32949 non-null  float64
 7   product_height_cm           32949 non-null  float64
 8   product_width_cm            32949 non-null  float64
dtypes: float64(7), object(2)
memory usage: 2.3+ MB
None
       product_name_lenght  product_description_lenght  product_photos_qty  \
count         32341.000000                32341.000000        32341.000000   
mean             48.476949                  771.495285            2.188986   
std              10.245741                  635.115225            1.736766   
min               5.000000                    4.000000            1.000000   
25%              42.000000                  339.000000            1.000000   
50%              51.000000                  595.000000            1.000000   
75%              57.000000                  972.000000            3.000000   
max              76.000000                 3992.000000           20.000000   

       product_weight_g  product_length_cm  product_height_cm  \
count      32949.000000       32949.000000       32949.000000   
mean        2276.472488          30.815078          16.937661   
std         4282.038731          16.914458          13.637554   
min            0.000000           7.000000           2.000000   
25%          300.000000          18.000000           8.000000   
50%          700.000000          25.000000          13.000000   
75%         1900.000000          38.000000          21.000000   
max        40425.000000         105.000000         105.000000   

       product_width_cm  
count      32949.000000  
mean          23.196728  
std           12.079047  
min            6.000000  
25%           15.000000  
50%           20.000000  
75%           30.000000  
max          118.000000  
----------------------------------------------------------------------------
----------------------------------------------------------------------------
                          seller_id  seller_zip_code_prefix  \
0  3442f8959a84dea7ee197c632cb2df15                   13023   
1  d1b65fc7debc3361ea86b5f14c68d2e2                   13844   
2  ce3ad9de960102d0677a81f5d0bb7b2d                   20031   
3  c0f3eea2e14555b6faeea3dd58c1b1c3                    4195   
4  51a04a8a6bdcb23deccc82b0b80742cf                   12914   

         seller_city seller_state  
0           campinas           SP  
1         mogi guacu           SP  
2     rio de janeiro           RJ  
3          sao paulo           SP  
4  braganca paulista           SP  
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3095 entries, 0 to 3094
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   seller_id               3095 non-null   object
 1   seller_zip_code_prefix  3095 non-null   int64 
 2   seller_city             3095 non-null   object
 3   seller_state            3095 non-null   object
dtypes: int64(1), object(3)
memory usage: 96.8+ KB
None
       seller_zip_code_prefix
count             3095.000000
mean             32291.059451
std              32713.453830
min               1001.000000
25%               7093.500000
50%              14940.000000
75%              64552.500000
max              99730.000000
----------------------------------------------------------------------------
----------------------------------------------------------------------------
    product_category_name product_category_name_english
0            beleza_saude                 health_beauty
1  informatica_acessorios         computers_accessories
2              automotivo                          auto
3         cama_mesa_banho                bed_bath_table
4        moveis_decoracao               furniture_decor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 71 entries, 0 to 70
Data columns (total 2 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   product_category_name          71 non-null     object
 1   product_category_name_english  71 non-null     object
dtypes: object(2)
memory usage: 1.2+ KB
None
       product_category_name product_category_name_english
count                     71                            71
unique                    71                            71
top             beleza_saude                 health_beauty
freq                       1                             1
----------------------------------------------------------------------------
```

Exploração básica de algumas tabelas

In [10]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Função para explorar um dataset
def explore_data(df, name):
    print(f"\nExplorando o dataset {name}:")
    print(df.head())
    print(f"\nInformações sobre {name}:")
    print(df.info())
    print(f"\nEstatísticas descritivas de {name}:")
```

```python
    print(df.describe())
    print(f"\nValores ausentes em {name}:")
    print(df.isnull().sum())


# Explorar cada dataset
explore_data(customers, "Clientes")
explore_data(geolocation, "Geolocalização")
explore_data(order_items, "Itens do Pedido")
explore_data(order_payments, "Pagamentos")
explore_data(order_reviews, "Avaliações")
explore_data(orders, "Pedidos")
explore_data(products, "Produtos")
explore_data(sellers, "Vendedores")
explore_data(product_category_translation, "Tradução de Categorias")


# Visualizações iniciais (exemplos)
plt.figure(figsize=(10, 5))
sns.countplot(x='order_status', data=orders)
plt.title('Distribuição dos Status dos Pedidos')
plt.show()


plt.figure(figsize=(12, 6))
sns.histplot(order_items['price'], bins=50, kde=True)
plt.title('Distribuição do Preço dos Itens')
plt.show()


# Visualizações adicionais
plt.figure(figsize=(12, 6))
sns.boxplot(x='payment_type', y='payment_value', data=order_payments)
plt.title('Valor do Pagamento por Tipo de Pagamento')
plt.xticks(rotation=45, ha='right') # Rotaciona os labels do eixo x para melhor
plt.show()

plt.figure(figsize=(10, 5))
sns.countplot(x='payment_installments', data=order_payments)
plt.title('Número de Parcelas')
plt.show()


print("\nFim da Exploração Inicial.")
```

```
Explorando o dataset Clientes:
                        customer_id                customer_unique_id  \
0  06b8999e2fba1a1fbc88172c00ba8bc7  861eff4711a542e4b93843c6dd7febb0
1  18955e83d337fd6b2def6b18a428ac77  290c77bc529b7ac935b93aa66c333dc3
2  4e7b3e00288586ebd08712fdd0374a03  060e732b5b29e8181a18229c7b0b2b5e
3  b2b6027bc5c5109e529d4dc6358b12c3  259dac757896d24d7702b9acbbff3f3c
4  4f2d8ab171c80ec8364f7c12e35b23ad  345ecd01c38d18a9036ed96c73b8d066

   customer_zip_code_prefix          customer_city customer_state
0                     14409                 franca             SP
1                      9790  sao bernardo do campo             SP
2                      1151              sao paulo             SP
3                      8775        mogi das cruzes             SP
4                     13056               campinas             SP

Informações sobre Clientes:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 5 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   customer_id               99441 non-null  object
 1   customer_unique_id        99441 non-null  object
 2   customer_zip_code_prefix  99441 non-null  int64
 3   customer_city             99441 non-null  object
 4   customer_state            99441 non-null  object
dtypes: int64(1), object(4)
memory usage: 3.8+ MB
None

Estatísticas descritivas de Clientes:
       customer_zip_code_prefix
count              99441.000000
mean               35137.474583
std                29797.938996
min                 1003.000000
25%                11347.000000
50%                24416.000000
75%                58900.000000
max                99990.000000

Valores ausentes em Clientes:
customer_id                 0
customer_unique_id          0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64

Explorando o dataset Geolocalização:
   geolocation_zip_code_prefix  geolocation_lat  geolocation_lng  \
0                         1037       -23.545621       -46.639292
1                         1046       -23.546081       -46.644820
2                         1046       -23.546129       -46.642951
3                         1041       -23.544392       -46.639499
4                         1035       -23.541578       -46.641607

  geolocation_city geolocation_state
0        sao paulo                SP
1        sao paulo                SP
```

```
2        sao paulo              SP
3        sao paulo              SP
4        sao paulo              SP

Informações sobre Geolocalização:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000163 entries, 0 to 1000162
Data columns (total 5 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   geolocation_zip_code_prefix  1000163 non-null  int64
 1   geolocation_lat              1000163 non-null  float64
 2   geolocation_lng              1000163 non-null  float64
 3   geolocation_city             1000163 non-null  object
 4   geolocation_state            1000163 non-null  object
dtypes: float64(2), int64(1), object(2)
memory usage: 38.2+ MB
None

Estatísticas descritivas de Geolocalização:
       geolocation_zip_code_prefix  geolocation_lat  geolocation_lng
count                 1.000163e+06     1.000163e+06     1.000163e+06
mean                  3.657417e+04    -2.117615e+01    -4.639054e+01
std                   3.054934e+04     5.715866e+00     4.269748e+00
min                   1.001000e+03    -3.660537e+01    -1.014668e+02
25%                   1.107500e+04    -2.360355e+01    -4.857317e+01
50%                   2.653000e+04    -2.291938e+01    -4.663788e+01
75%                   6.350400e+04    -1.997962e+01    -4.376771e+01
max                   9.999000e+04     4.506593e+01     1.211054e+02

Valores ausentes em Geolocalização:
geolocation_zip_code_prefix    0
geolocation_lat                0
geolocation_lng                0
geolocation_city               0
geolocation_state              0
dtype: int64

Explorando o dataset Itens do Pedido:
                           order_id  order_item_id  \
0  00010242fe8c5a6d1ba2dd792cb16214              1
1  00018f77f2f0320c557190d7a144bdd3              1
2  000229ec398224ef6ca0657da4fc703e              1
3  00024acbcdf0a6daa1e931b038114c75              1
4  00042b26cf59d7ce69dfabb4e55b4fd9              1

                         product_id                         seller_id  \
0  4244733e06e7ecb4970a6e2683c13e61  48436dade18ac8b2bce089ec2a041202
1  e5f2d52b802189ee658865ca93d83a8f  dd7ddc04e1b6c2c614352b383efe2d36
2  c777355d18b72b67abbeef9df44fd0fd  5b51032eddd242adc84c38acab88f23d
3  7634da152a4610f1595efa32f14722fc  9d7a1d34a5052409006425275ba1c2b4
4  ac6c3623068f30de03045865e4e10089  df560393f3a51e74553ab94004ba5c87

   shipping_limit_date   price  freight_value
0  2017-09-19 09:45:35   58.90          13.29
1  2017-05-03 11:05:13  239.90          19.93
2  2018-01-18 14:48:30  199.00          17.87
3  2018-08-15 10:10:18   12.99          12.79
4  2017-02-13 13:57:51  199.90          18.14
```

```
Informações sobre Itens do Pedido:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112650 entries, 0 to 112649
Data columns (total 7 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   order_id             112650 non-null  object 
 1   order_item_id        112650 non-null  int64  
 2   product_id           112650 non-null  object 
 3   seller_id            112650 non-null  object 
 4   shipping_limit_date  112650 non-null  object 
 5   price                112650 non-null  float64
 6   freight_value        112650 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 6.0+ MB
None

Estatísticas descritivas de Itens do Pedido:
       order_item_id          price  freight_value
count  112650.000000  112650.000000  112650.000000
mean        1.197834     120.653739      19.990320
std         0.705124     183.633928      15.806405
min         1.000000       0.850000       0.000000
25%         1.000000      39.900000      13.080000
50%         1.000000      74.990000      16.260000
75%         1.000000     134.900000      21.150000
max        21.000000    6735.000000     409.680000

Valores ausentes em Itens do Pedido:
order_id               0
order_item_id          0
product_id             0
seller_id              0
shipping_limit_date    0
price                  0
freight_value          0
dtype: int64

Explorando o dataset Pagamentos:
                           order_id  payment_sequential payment_type  \
0  b81ef226f3fe1789b1e8b2acac839d17                   1  credit_card   
1  a9810da82917af2d9aefd1278f1dcfa0                   1  credit_card   
2  25e8ea4e93396b6fa0d3dd708e76c1bd                   1  credit_card   
3  ba78997921bbcdc1373bb41e913ab953                   1  credit_card   
4  42fdf880ba16b47b59251dd489d4441a                   1  credit_card   

   payment_installments  payment_value  
0                     8          99.33  
1                     1          24.39  
2                     1          65.71  
3                     8         107.78  
4                     2         128.45  

Informações sobre Pagamentos:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103886 entries, 0 to 103885
Data columns (total 5 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   order_id              103886 non-null  object 
```

```
 1   payment_sequential    103886 non-null  int64  
 2   payment_type          103886 non-null  object 
 3   payment_installments  103886 non-null  int64  
 4   payment_value         103886 non-null  float64
dtypes: float64(1), int64(2), object(2)
memory usage: 4.0+ MB
None

Estatísticas descritivas de Pagamentos:
       payment_sequential  payment_installments  payment_value
count       103886.000000         103886.000000  103886.000000
mean             1.092679              2.853349     154.100380
std              0.706584              2.687051     217.494064
min              1.000000              0.000000       0.000000
25%              1.000000              1.000000      56.790000
50%              1.000000              1.000000     100.000000
75%              1.000000              4.000000     171.837500
max             29.000000             24.000000   13664.080000

Valores ausentes em Pagamentos:
order_id                0
payment_sequential      0
payment_type            0
payment_installments    0
payment_value           0
dtype: int64

Explorando o dataset Avaliações:
                          review_id                          order_id  \
0  7bc2406110b926393aa56f80a40eba40  73fc7af87114b39712e6da79b0a377eb   
1  80e641a11e56f04c1ad469d5645fdfde  a548910a1c6147796b98fdf73dbeba33   
2  228ce5500dc1d8e020d8d1322874b6f0  f9e4b658b201a9f2ecdecbb34bed034b   
3  e64fb393e7b32834bb789ff8bb30750e  658677c97b385a9be170737859d3511b   
4  f7c4243c7fe1938f181bec41a392bdeb  8e6bfb81e283fa7e4f11123a3fb894f1   

   review_score review_comment_title  \
0             4                  NaN   
1             5                  NaN   
2             5                  NaN   
3             5                  NaN   
4             5                  NaN   

                              review_comment_message review_creation_date  \
0                                                NaN  2018-01-18 00:00:00   
1                                                NaN  2018-03-10 00:00:00   
2                                                NaN  2018-02-17 00:00:00   
3              Recebi bem antes do prazo estipulado.  2017-04-21 00:00:00   
4  Parabéns lojas lannister adorei comprar pela I...  2018-03-01 00:00:00   

  review_answer_timestamp  
0     2018-01-18 21:46:59  
1     2018-03-11 03:05:13  
2     2018-02-18 14:36:24  
3     2017-04-21 22:02:06  
4     2018-03-02 10:26:53  

Informações sobre Avaliações:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99224 entries, 0 to 99223
Data columns (total 7 columns):
```

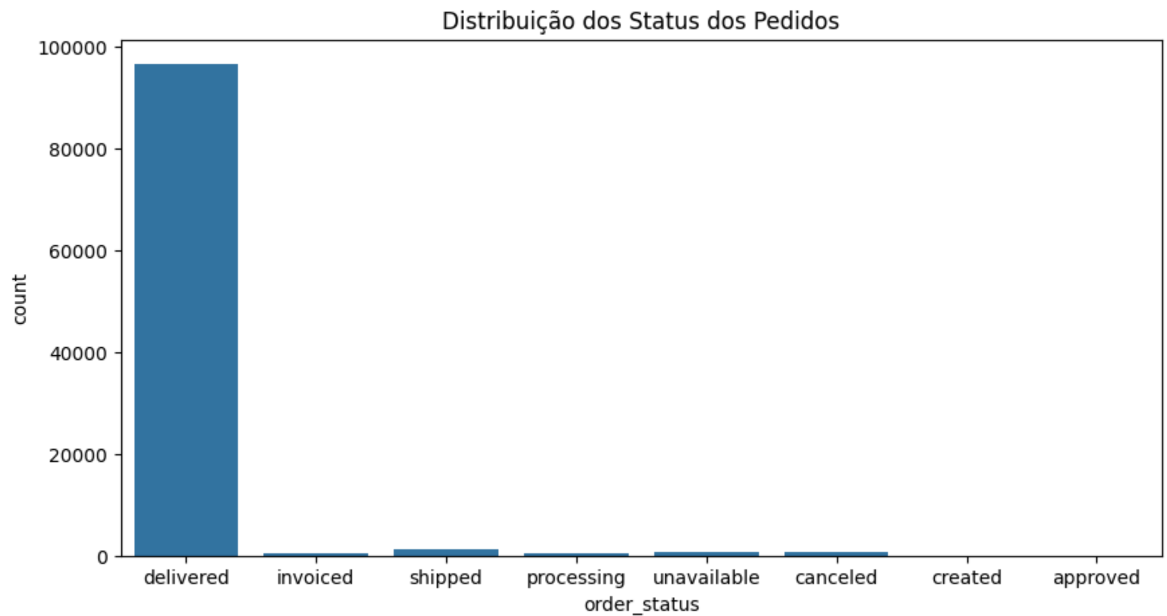
```
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   review_id                99224 non-null  object
 1   order_id                 99224 non-null  object
 2   review_score             99224 non-null  int64 
 3   review_comment_title     11568 non-null  object
 4   review_comment_message   40977 non-null  object
 5   review_creation_date     99224 non-null  object
 6   review_answer_timestamp  99224 non-null  object
dtypes: int64(1), object(6)
memory usage: 5.3+ MB
None

Estatísticas descritivas de Avaliações:
       review_score
count  99224.000000
mean       4.086421
std        1.347579
min        1.000000
25%        4.000000
50%        5.000000
75%        5.000000
max        5.000000

Valores ausentes em Avaliações:
review_id                      0
order_id                       0
review_score                   0
review_comment_title       87656
review_comment_message     58247
review_creation_date           0
review_answer_timestamp        0
dtype: int64

Explorando o dataset Pedidos:
                           order_id                       customer_id  \
0  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d   
1  53cdb2fc8bc7dce0b6741e2150273451  b0830fb4747a6c6d20dea0b8c802d7ef   
2  47770eb9100c2d0c44946d9cf07ec65d  41ce2a54c0b03bf3443c3d931a367089   
3  949d5b44dbf5de918fe9c16f97b45f8a  f88197465ea7920adcdbec7375364d82   
4  ad21c59c0840e6cb83a9ceb5573f8159  8ab97904e6daea8866dbdbc4fb7aad2c   

  order_status order_purchase_timestamp    order_approved_at  \
0    delivered      2017-10-02 10:56:33  2017-10-02 11:07:15   
1    delivered      2018-07-24 20:41:37  2018-07-26 03:24:27   
2    delivered      2018-08-08 08:38:49  2018-08-08 08:55:23   
3    delivered      2017-11-18 19:28:06  2017-11-18 19:45:59   
4    delivered      2018-02-13 21:18:39  2018-02-13 22:20:29   

  order_delivered_carrier_date order_delivered_customer_date  \
0          2017-10-04 19:55:00           2017-10-10 21:25:13   
1          2018-07-26 14:31:00           2018-08-07 15:27:45   
2          2018-08-08 13:50:00           2018-08-17 18:06:29   
3          2017-11-22 13:39:59           2017-12-02 00:28:42   
4          2018-02-14 19:46:34           2018-02-16 18:17:02   

  order_estimated_delivery_date  
0           2017-10-18 00:00:00  
1           2018-08-13 00:00:00  
2           2018-09-04 00:00:00  
```

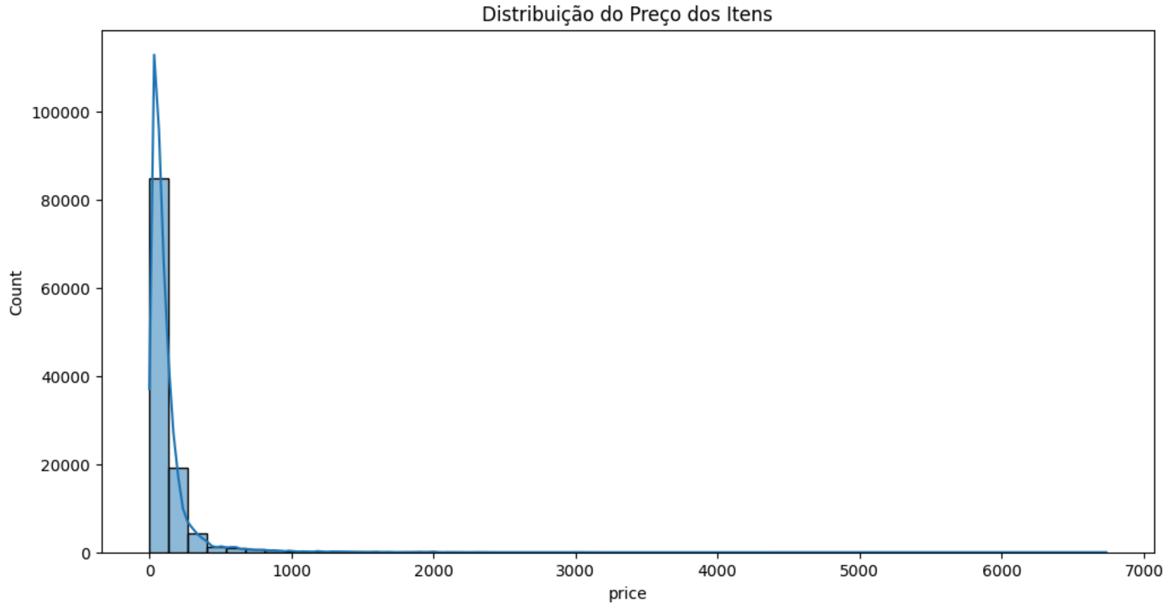
```
3          2017-12-15 00:00:00
4          2018-02-26 00:00:00

Informações sobre Pedidos:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   order_id                       99441 non-null  object
 1   customer_id                    99441 non-null  object
 2   order_status                   99441 non-null  object
 3   order_purchase_timestamp       99441 non-null  object
 4   order_approved_at              99281 non-null  object
 5   order_delivered_carrier_date   97658 non-null  object
 6   order_delivered_customer_date  96476 non-null  object
 7   order_estimated_delivery_date  99441 non-null  object
dtypes: object(8)
memory usage: 6.1+ MB
None

Estatísticas descritivas de Pedidos:
                                order_id                       customer_id  \
count                              99441                             99441   
unique                             99441                             99441   
top     e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d   
freq                                   1                                 1   

       order_status order_purchase_timestamp    order_approved_at  \
count         99441                    99441                99281   
unique            8                    98875                90733   
top       delivered      2018-04-11 10:48:14  2018-02-27 04:31:10   
freq          96478                        3                    9   

       order_delivered_carrier_date order_delivered_customer_date  \
count                         97658                         96476   
unique                        81018                         95664   
top             2018-05-09 15:48:00           2018-05-08 23:38:46   
freq                             47                             3   

       order_estimated_delivery_date  
count                          99441  
unique                           459  
top              2017-12-20 00:00:00  
freq                             522  

Valores ausentes em Pedidos:
order_id                            0
customer_id                         0
order_status                        0
order_purchase_timestamp            0
order_approved_at                 160
order_delivered_carrier_date     1783
order_delivered_customer_date    2965
order_estimated_delivery_date       0
dtype: int64

Explorando o dataset Produtos:
                         product_id  product_category_name  \
0  1e9e8ef04dbcff4541ed26657ea517e5             perfumaria   
```

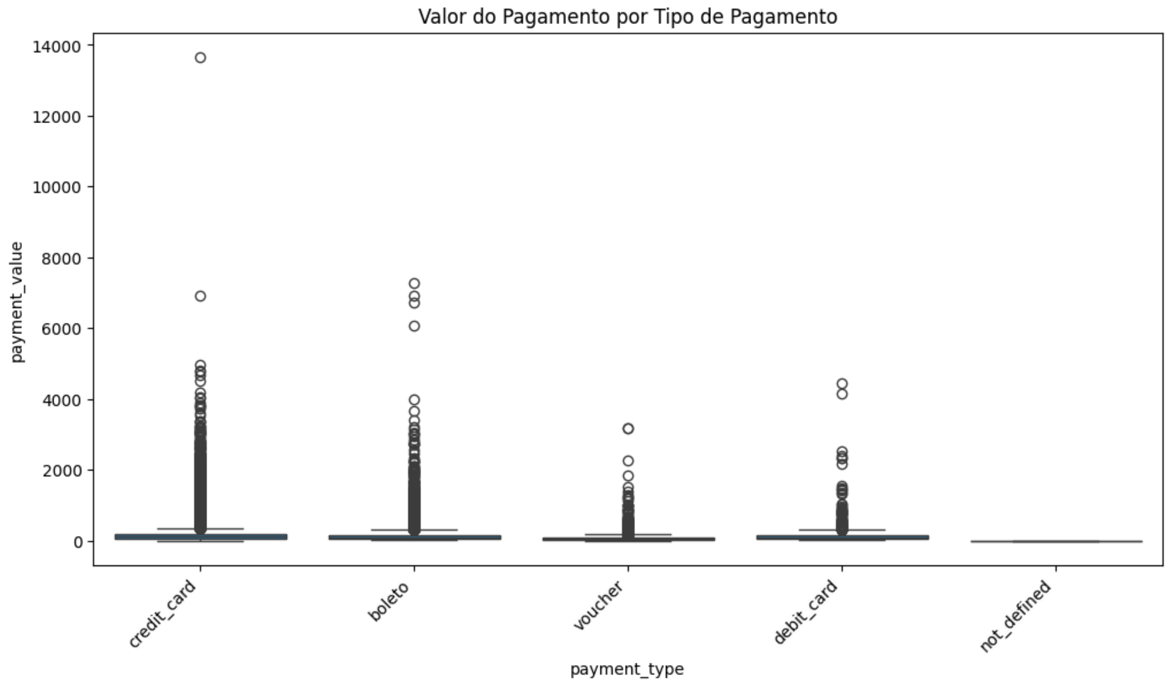
```
1  3aa071139cb16b67ca9e5dea641aaa2f                 artes
2  96bd76ec8810374ed1b65e291975717f         esporte_lazer
3  cef67bcfe19066a932b7673e239eb23d                 bebes
4  9dc1a7de274444849c219cff195d0b71  utilidades_domesticas

   product_name_lenght  product_description_lenght  product_photos_qty  \
0                 40.0                       287.0                 1.0
1                 44.0                       276.0                 1.0
2                 46.0                       250.0                 1.0
3                 27.0                       261.0                 1.0
4                 37.0                       402.0                 4.0

   product_weight_g  product_length_cm  product_height_cm  product_width_cm
0             225.0               16.0               10.0              14.0
1            1000.0               30.0               18.0              20.0
2             154.0               18.0                9.0              15.0
3             371.0               26.0                4.0              26.0
4             625.0               20.0               17.0              13.0

Informações sobre Produtos:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32951 entries, 0 to 32950
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   product_id                  32951 non-null  object
 1   product_category_name       32341 non-null  object
 2   product_name_lenght         32341 non-null  float64
 3   product_description_lenght  32341 non-null  float64
 4   product_photos_qty          32341 non-null  float64
 5   product_weight_g            32949 non-null  float64
 6   product_length_cm           32949 non-null  float64
 7   product_height_cm           32949 non-null  float64
 8   product_width_cm            32949 non-null  float64
dtypes: float64(7), object(2)
memory usage: 2.3+ MB
None

Estatísticas descritivas de Produtos:
       product_name_lenght  product_description_lenght  product_photos_qty  \
count         32341.000000                32341.000000        32341.000000
mean             48.476949                  771.495285            2.188986
std              10.245741                  635.115225            1.736766
min               5.000000                    4.000000            1.000000
25%              42.000000                  339.000000            1.000000
50%              51.000000                  595.000000            1.000000
75%              57.000000                  972.000000            3.000000
max              76.000000                 3992.000000           20.000000

       product_weight_g  product_length_cm  product_height_cm  \
count      32949.000000       32949.000000       32949.000000
mean        2276.472488          30.815078          16.937661
std         4282.038731          16.914458          13.637554
min            0.000000           7.000000           2.000000
25%          300.000000          18.000000           8.000000
50%          700.000000          25.000000          13.000000
75%         1900.000000          38.000000          21.000000
max        40425.000000         105.000000         105.000000

       product_width_cm
```

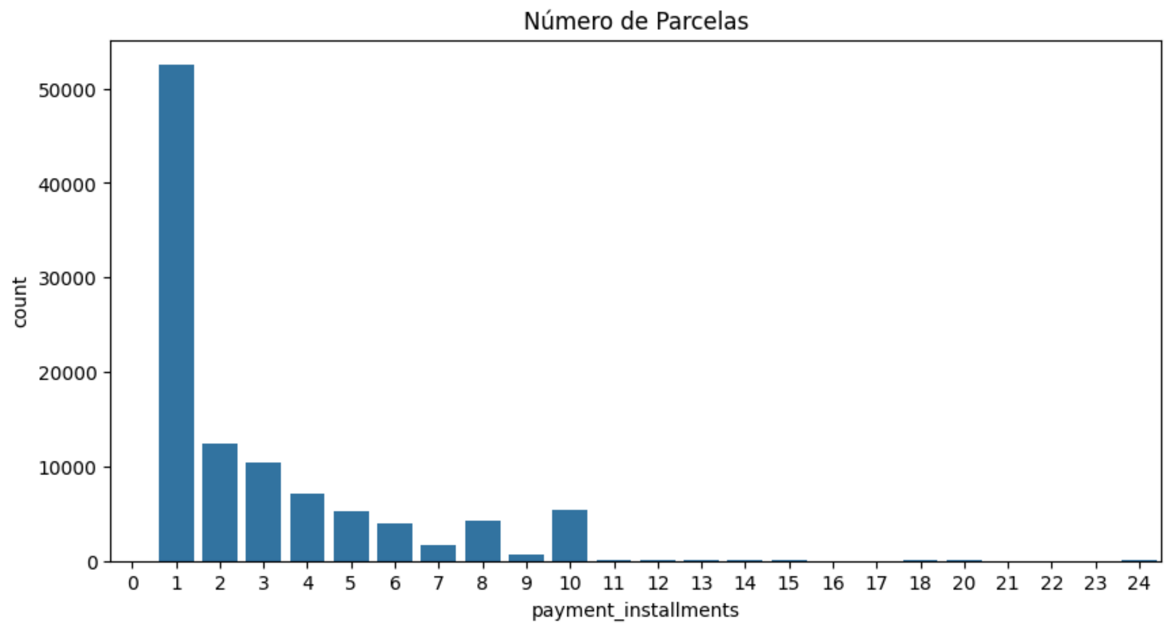
```
count      32949.000000
mean          23.196728
std           12.079047
min            6.000000
25%           15.000000
50%           20.000000
75%           30.000000
max          118.000000

Valores ausentes em Produtos:
product_id                      0
product_category_name         610
product_name_lenght           610
product_description_lenght    610
product_photos_qty            610
product_weight_g                2
product_length_cm               2
product_height_cm               2
product_width_cm                2
dtype: int64

Explorando o dataset Vendedores:
                          seller_id  seller_zip_code_prefix  \
0  3442f8959a84dea7ee197c632cb2df15                   13023
1  d1b65fc7debc3361ea86b5f14c68d2e2                   13844
2  ce3ad9de960102d0677a81f5d0bb7b2d                   20031
3  c0f3eea2e14555b6faeea3dd58c1b1c3                    4195
4  51a04a8a6bdcb23deccc82b0b80742cf                   12914

         seller_city seller_state
0           campinas           SP
1         mogi guacu           SP
2     rio de janeiro           RJ
3          sao paulo           SP
4  braganca paulista           SP

Informações sobre Vendedores:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3095 entries, 0 to 3094
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   seller_id               3095 non-null   object
 1   seller_zip_code_prefix  3095 non-null   int64
 2   seller_city             3095 non-null   object
 3   seller_state            3095 non-null   object
dtypes: int64(1), object(3)
memory usage: 96.8+ KB
None

Estatísticas descritivas de Vendedores:
       seller_zip_code_prefix
count             3095.000000
mean             32291.059451
std              32713.453830
min               1001.000000
25%               7093.500000
50%              14940.000000
75%              64552.500000
max              99730.000000
```

```
Valores ausentes em Vendedores:
seller_id                 0
seller_zip_code_prefix    0
seller_city               0
seller_state              0
dtype: int64

Explorando o dataset Tradução de Categorias:
    product_category_name product_category_name_english
0            beleza_saude                 health_beauty
1  informatica_acessorios         computers_accessories
2              automotivo                          auto
3         cama_mesa_banho                bed_bath_table
4        moveis_decoracao               furniture_decor

Informações sobre Tradução de Categorias:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 71 entries, 0 to 70
Data columns (total 2 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   product_category_name          71 non-null     object
 1   product_category_name_english  71 non-null     object
dtypes: object(2)
memory usage: 1.2+ KB
None

Estatísticas descritivas de Tradução de Categorias:
       product_category_name product_category_name_english
count                     71                            71
unique                    71                            71
top             beleza_saude                 health_beauty
freq                       1                             1

Valores ausentes em Tradução de Categorias:
product_category_name            0
product_category_name_english    0
dtype: int64
```










```
Fim da Exploração Inicial.
```

Análise descritiva

Calcular medidas de tendência central (média, mediana, moda) e dispersão (desvio padrão, variância). Criar visualizações como histogramas, gráficos de barras e dispersão para representar os dados e obter insights sobre vendas, comportamento do cliente e desempenho de marketing.

Instalação das bibliotecas

In [9]:
```
%pip install pandas matplotlib seaborn
```

```
Requirement already satisfied: pandas in c:\python312\lib\site-packages (2.2.2)No
te: you may need to restart the kernel to use updated packages.

Requirement already satisfied: matplotlib in c:\python312\lib\site-packages (3.9.
2)
Requirement already satisfied: seaborn in c:\python312\lib\site-packages (0.13.2)
Requirement already satisfied: numpy>=1.26.0 in c:\python312\lib\site-packages (f
rom pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\python312\lib\site-pa
ckages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\python312\lib\site-packages (fr
om pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\python312\lib\site-packages
(from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\python312\lib\site-packages
(from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\python312\lib\site-packages (fr
om matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\python312\lib\site-package
s (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\python312\lib\site-package
s (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in c:\users\salom\appdata\roaming
\python\python312\site-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in c:\python312\lib\site-packages (from
matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\python312\lib\site-packages
(from matplotlib) (3.1.4)
Requirement already satisfied: six>=1.5 in c:\python312\lib\site-packages (from p
ython-dateutil>=2.8.2->pandas) (1.16.0)
```

```
[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Leitura das tabelas

In [10]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Carregar os datasets
customers = pd.read_csv("olist_customers_dataset.csv")
```

```
geolocation = pd.read_csv("olist_geolocation_dataset.csv")
order_items = pd.read_csv("olist_order_items_dataset.csv")
order_payments = pd.read_csv("olist_order_payments_dataset.csv")
order_reviews = pd.read_csv("olist_order_reviews_dataset.csv")
orders = pd.read_csv("olist_orders_dataset.csv")
products = pd.read_csv("olist_products_dataset.csv")
sellers = pd.read_csv("olist_sellers_dataset.csv")
product_category_translation = pd.read_csv("product_category_name_translation.cs
```

Primeiras Análises

In [11]:

```
# Análise descritiva para variáveis numéricas
print("\nAnálise Descritiva de 'price' em 'order_items':")
print(order_items['price'].describe())

print("\nAnálise Descritiva de 'freight_value' em 'order_items':")
print(order_items['freight_value'].describe())



# Análise descritiva por categoria
# 1. Merge order_items com products para obter a categoria
order_items_with_category = pd.merge(order_items, products[['product_id', 'produ

# 2. Calcula o preço médio por categoria traduzida
preco_medio_por_categoria = order_items_with_category.groupby('product_category_

# Imprime o resultado
print("\nPreço médio por categoria de produto:")
print(preco_medio_por_categoria)

# Contagem de pedidos por status
print("\nContagem de pedidos por status:")
print(orders['order_status'].value_counts())
```

```
Análise Descritiva de 'price' em 'order_items':
count    112650.000000
mean        120.653739
std         183.633928
min           0.850000
25%          39.900000
50%          74.990000
75%         134.900000
max        6735.000000
Name: price, dtype: float64

Análise Descritiva de 'freight_value' em 'order_items':
count    112650.000000
mean         19.990320
std          15.806405
min           0.000000
25%          13.080000
50%          16.260000
75%          21.150000
max         409.680000
Name: freight_value, dtype: float64

Preço médio por categoria de produto:
product_category_name
agro_industria_e_comercio    342.124858
alimentos                     57.634137
alimentos_bebidas             54.602446
artes                        115.802105
artes_e_artesanato            75.583750
                                ...
sinalizacao_e_seguranca      108.086583
tablets_impressao_imagem      90.703735
telefonia                     71.213978
telefonia_fixa               225.693182
utilidades_domesticas         90.788148
Name: price, Length: 73, dtype: float64

Contagem de pedidos por status:
order_status
delivered      96478
shipped         1107
canceled         625
unavailable      609
invoiced         314
processing       301
created            5
approved           2
Name: count, dtype: int64
```

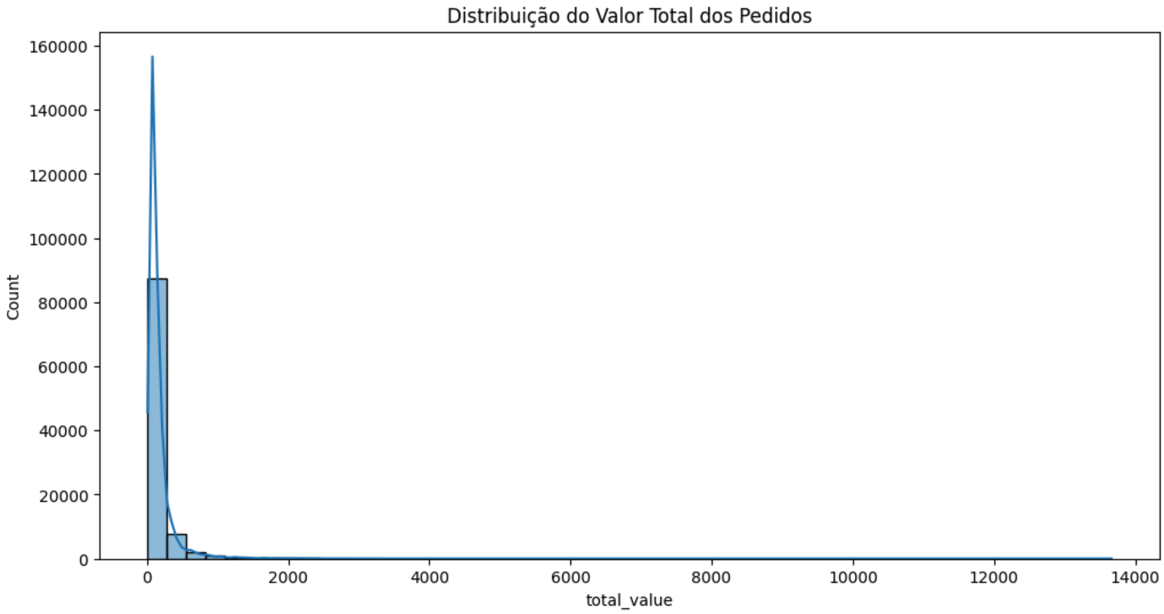
Análise do Valor Total dos Pedidos

In [12]:

```python
# Calcula o valor total de cada pedido
order_items['total_value'] = order_items['price'] + order_items['freight_value']

# Agrupa por pedido e soma o valor total dos itens
order_totals = order_items.groupby('order_id')['total_value'].sum().reset_index(

# Junta com o dataset de pedidos para ter mais informações
orders = pd.merge(orders, order_totals, on='order_id', how='left')
```

```python
# Analisa a distribuição do valor total dos pedidos
print(orders['total_value'].describe())

# Visualiza a distribuição
plt.figure(figsize=(12, 6))
sns.histplot(orders['total_value'], bins=50, kde=True)
plt.title('Distribuição do Valor Total dos Pedidos')
plt.show()

# Pedidos com maior valor
print("\nPedidos com maior valor:")
print(orders.sort_values('total_value', ascending=False).head())
```

```
count    98666.000000
mean       160.577638
std        220.466087
min          9.590000
25%         61.980000
50%        105.290000
75%        176.870000
max      13664.080000
Name: total_value, dtype: float64
```



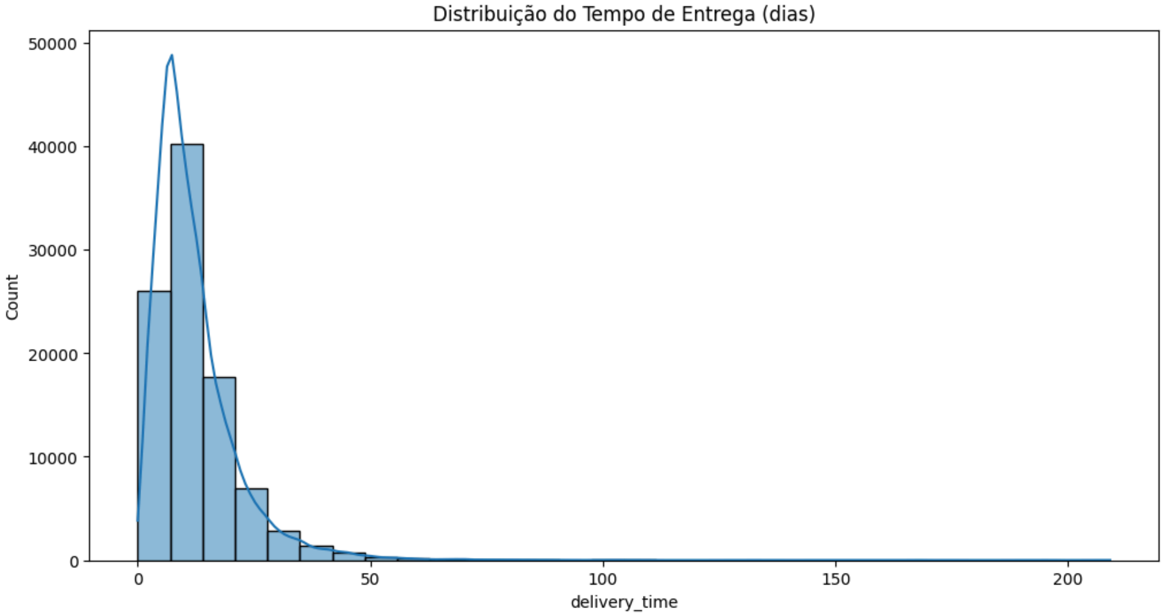
```
Pedidos com maior valor:
                               order_id                       customer_id  \
13390  03caa2c082116e1d31e67e9ae3700499  1617b1357756262bfa56ab541c47bc16
66599  736e1922ae60d0d6a89247b851902527  ec5b2ba62e574342386871631fafd3fc
22171  0812eb902a67711a1cb742b3cdaa65ae  c6e2731c5b391845f6800c97401a43a9
28326  fefacc66af859508bf1a7934eab1e97f  f48d464a0baaea338cb25f816991ab1f
3508   f5136e38d1a14a4dbd87dff67da82701  3fd6777bbce08a352fddd04e4a7cc8f6

      order_status order_purchase_timestamp    order_approved_at  \
13390    delivered      2017-09-29 15:24:52  2017-10-02 15:28:20
66599    delivered      2018-07-15 14:49:44  2018-07-17 04:31:36
22171    delivered      2017-02-12 20:37:36  2017-02-12 20:45:12
28326    delivered      2018-07-25 18:10:17  2018-07-27 04:05:13
3508     delivered      2017-05-24 18:14:34  2017-05-26 02:45:17

      order_delivered_carrier_date order_delivered_customer_date  \
13390          2017-10-10 15:43:17           2017-10-17 18:22:29
66599          2018-07-20 13:09:00           2018-07-26 22:03:06
22171          2017-02-16 09:23:13           2017-03-03 14:23:18
28326          2018-08-03 14:42:00           2018-08-15 14:57:50
3508           2017-05-26 11:20:47           2017-06-05 17:09:48

      order_estimated_delivery_date  total_value
13390           2017-10-23 00:00:00     13664.08
66599           2018-08-02 00:00:00      7274.88
22171           2017-03-09 00:00:00      6929.31
28326           2018-08-10 00:00:00      6922.21
3508            2017-06-28 00:00:00      6726.66
```

Análise do Tempo de Entrega

In [13]:

```python
# Converte as colunas de data para o tipo datetime
orders['order_purchase_timestamp'] = pd.to_datetime(orders['order_purchase_times
orders['order_delivered_customer_date'] = pd.to_datetime(orders['order_delivered
orders['order_estimated_delivery_date'] = pd.to_datetime(orders['order_estimated


# Calcula o tempo de entrega em dias
orders['delivery_time'] = (orders['order_delivered_customer_date'] - orders['ord

# Calcula a diferença entre a data estimada e a data real de entrega
orders['delivery_diff'] = (orders['order_estimated_delivery_date'] - orders['ord

# Analisa o tempo de entrega
print("\nAnálise do Tempo de Entrega:")
print(orders['delivery_time'].describe())


# Visualiza a distribuição do tempo de entrega
plt.figure(figsize=(12, 6))
sns.histplot(orders['delivery_time'], bins=30, kde=True)
plt.title('Distribuição do Tempo de Entrega (dias)')
plt.show()
```

```
Análise do Tempo de Entrega:
count    96476.000000
mean        12.094086
std          9.551746
min          0.000000
25%          6.000000
50%         10.000000
75%         15.000000
max        209.000000
Name: delivery_time, dtype: float64
```

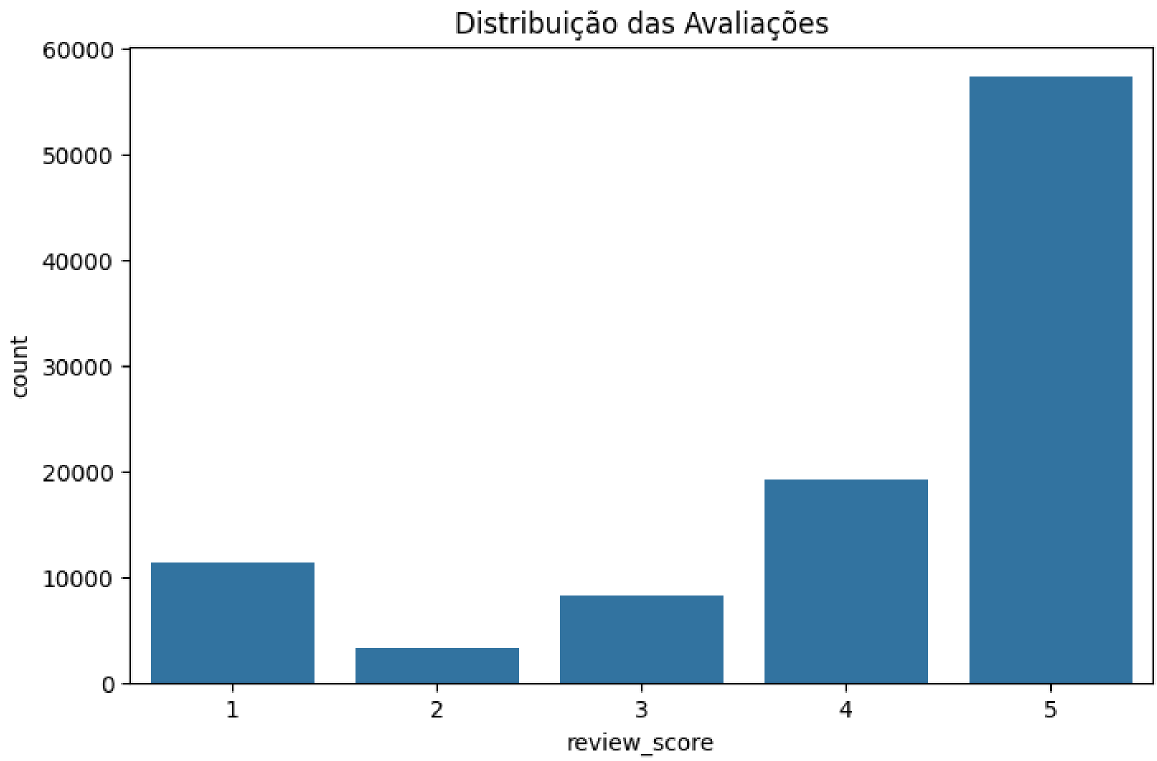


Análise das Avaliações

In [14]:

```python
# Junta as avaliações com os dados dos pedidos
orders = pd.merge(orders, order_reviews, on='order_id', how='left')


# Analisa a distribuição das avaliações
print("\nAnálise das Avaliações:")
print(order_reviews['review_score'].describe())

plt.figure(figsize=(8,5))
sns.countplot(x='review_score', data=order_reviews)
plt.title('Distribuição das Avaliações')
plt.show()
```

```
Análise das Avaliações:
count    99224.000000
mean         4.086421
std          1.347579
min          1.000000
25%          4.000000
50%          5.000000
75%          5.000000
max          5.000000
Name: review_score, dtype: float64
```

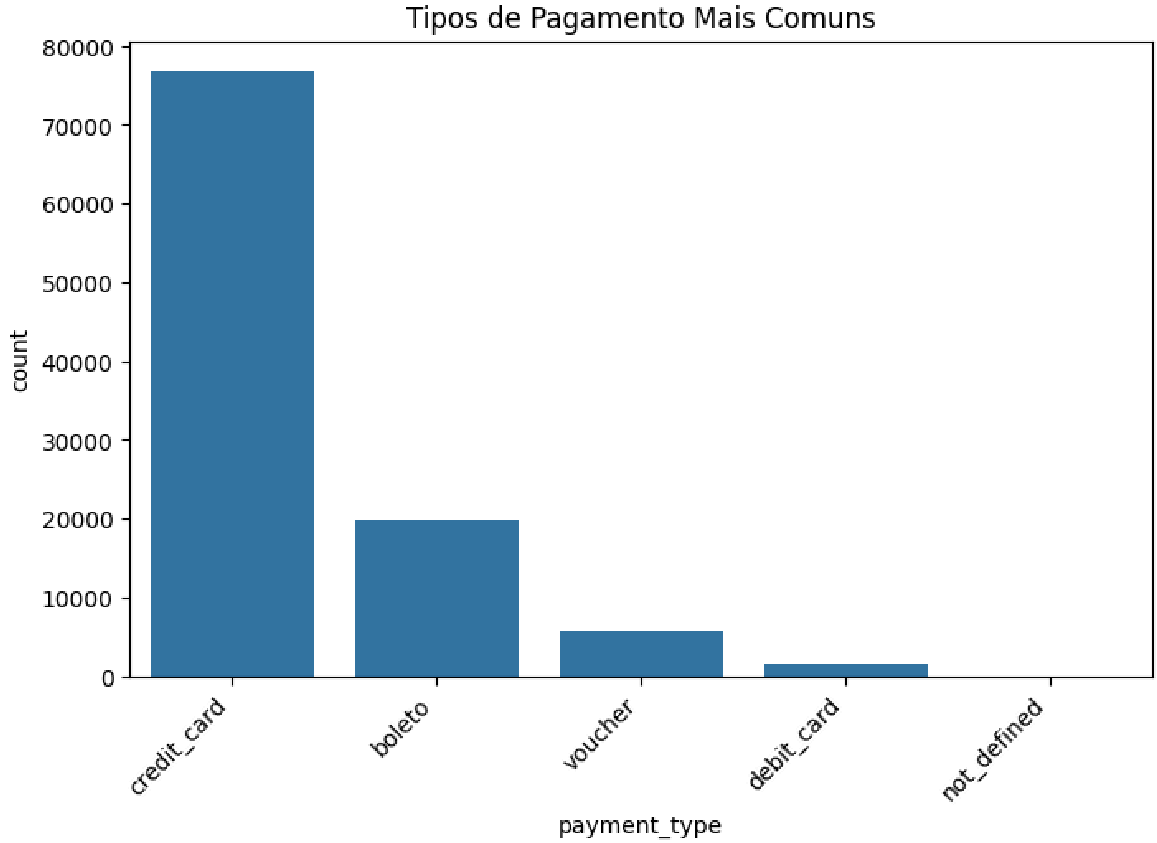


Análise dos Pagamentos

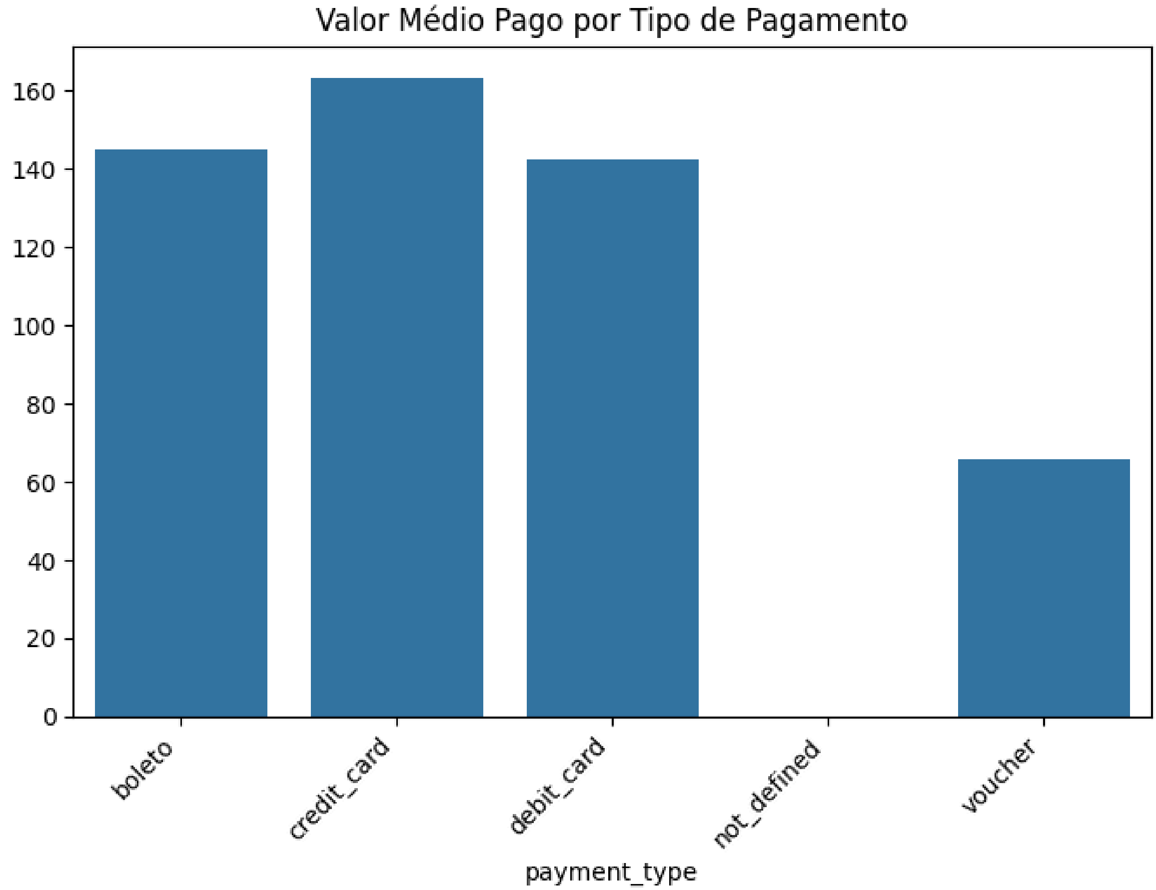
In [16]:
```
print("\nAnálise dos Pagamentos:")

# 1. Tipos de pagamento mais comuns
payment_types = order_payments['payment_type'].value_counts(normalize=True)
print("\nTipos de Pagamento Mais Comuns (Proporção):")
print(payment_types)

plt.figure(figsize=(8, 5))
sns.countplot(x='payment_type', data=order_payments, order=payment_types.index)
plt.title('Tipos de Pagamento Mais Comuns')
plt.xticks(rotation=45, ha='right')
plt.show()


# 2. Valor médio pago por tipo de pagamento
mean_payment_by_type = order_payments.groupby('payment_type')['payment_value'].m
print("\nValor Médio Pago por Tipo de Pagamento:")
print(mean_payment_by_type)

plt.figure(figsize=(8, 5))
sns.barplot(x=mean_payment_by_type.index, y=mean_payment_by_type.values)
plt.title('Valor Médio Pago por Tipo de Pagamento')
plt.xticks(rotation=45, ha='right')
plt.show()


# 3. Distribuição do valor pago por tipo de pagamento
plt.figure(figsize=(10, 6))
sns.boxplot(x='payment_type', y='payment_value', data=order_payments, showfliers
plt.title('Distribuição do Valor Pago por Tipo de Pagamento (sem outliers)')
plt.xticks(rotation=45, ha='right')
plt.show()
```

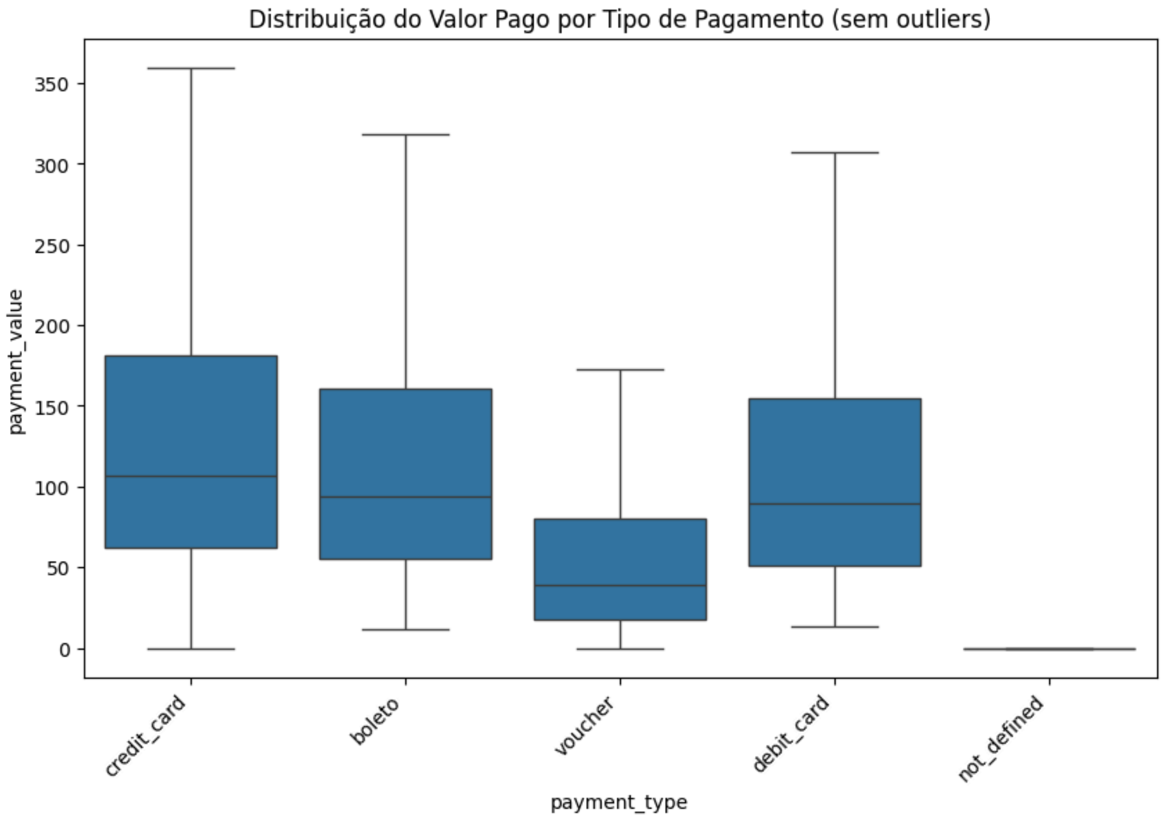
```
# 4. Número de parcelas mais comuns
installments = order_payments['payment_installments'].value_counts(normalize=Tru
print("\nNúmero de Parcelas Mais Comuns (Proporção):")
print(installments)

plt.figure(figsize=(10, 5))
sns.countplot(x='payment_installments', data=order_payments, order=installments.
plt.title('Número de Parcelas Mais Comuns')
plt.show()


# 5. Valor pago por número de parcelas (remover a parcela 0 que indica pagamento
installments_no_zero = order_payments[order_payments['payment_installments'] !=

plt.figure(figsize=(12,6))
sns.boxplot(x='payment_installments', y='payment_value', data=installments_no_ze
plt.title('Valor Pago x Número de Parcelas (sem outliers e sem parcela 0)')
plt.show()



# 6. Correlação entre valor da parcela e número de parcelas
correlation = installments_no_zero['payment_value'].corr(installments_no_zero['p
print(f"\nCorrelação entre valor da parcela e número de parcelas: {correlation}"
```

```
Análise dos Pagamentos:

Tipos de Pagamento Mais Comuns (Proporção):
payment_type
credit_card    0.739224
boleto         0.190440
voucher        0.055590
debit_card     0.014718
not_defined    0.000029
Name: proportion, dtype: float64
```

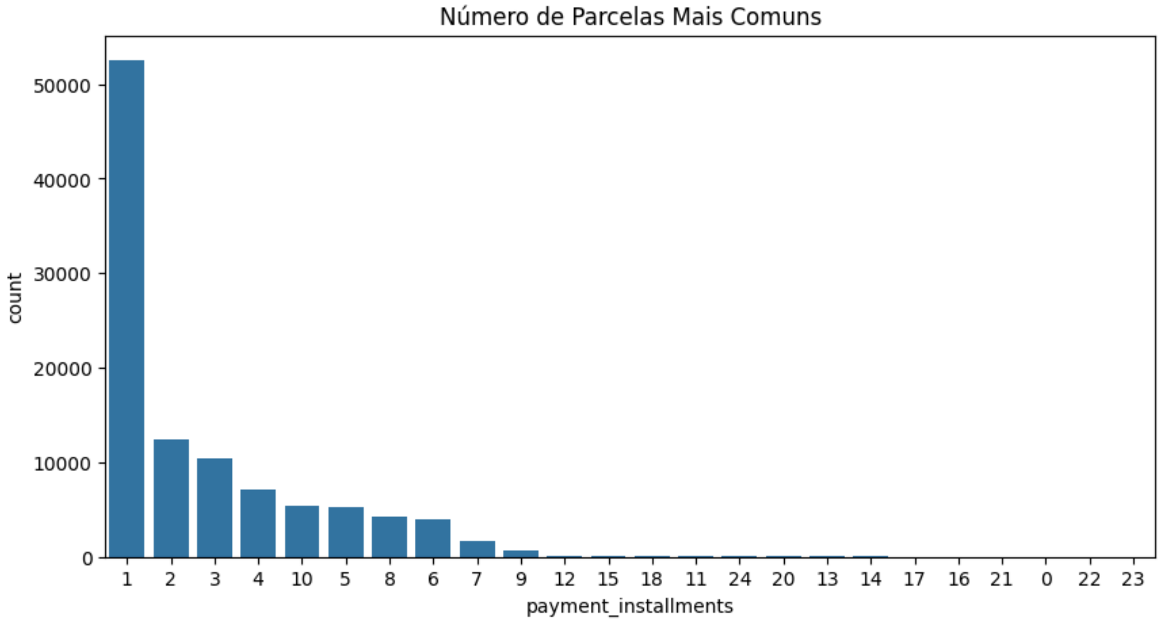


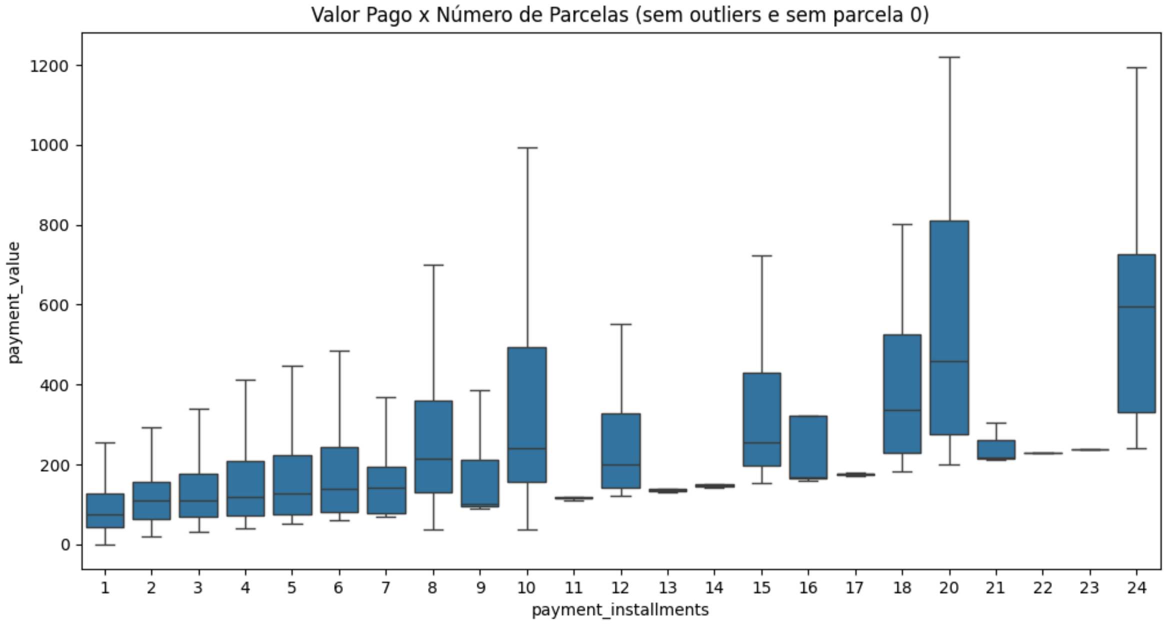

```
Valor Médio Pago por Tipo de Pagamento:
payment_type
boleto         145.034435
credit_card    163.319021
debit_card     142.570170
not_defined      0.000000
voucher         65.703354
Name: payment_value, dtype: float64
```


Valor Médio Pago por Tipo de Pagamento
160
140
120
100
80
60
40
20
0
boleto
credit_card
debit_card
not_defined
voucher
payment_type


Distribuição do Valor Pago por Tipo de Pagamento (sem outliers)
payment_value
350
300
250
200
150
100
50
0
credit_card
boleto
voucher
debit_card
not_defined
payment_type

```
Número de Parcelas Mais Comuns (Proporção):
payment_installments
1     0.505804
2     0.119487
3     0.100697
4     0.068325
10    0.051287
5     0.050430
8     0.041083
6     0.037734
7     0.015652
9     0.006199
12    0.001280
15    0.000712
18    0.000260
11    0.000221
24    0.000173
20    0.000164
13    0.000154
14    0.000144
17    0.000077
16    0.000048
21    0.000029
0     0.000019
22    0.000010
23    0.000010
Name: proportion, dtype: float64
```






```
Correlação entre valor da parcela e número de parcelas: 0.3308091417006476
```

Segmentação de clientes

Usar o algoritmo K-Means para agrupar clientes com base em seus comportamentos de compra. Descrever as características de cada grupo.

Instalação de bibliotecas

In [1]:
```
%pip install pandas matplotlib seaborn sklearn
```

```
Requirement already satisfied: pandas in c:\python312\lib\site-packages (2.2.2)
Requirement already satisfied: matplotlib in c:\python312\lib\site-packages (3.9.
2)
Requirement already satisfied: seaborn in c:\python312\lib\site-packages (0.13.2)
Collecting sklearn
  Using cached sklearn-0.0.post12.tar.gz (2.6 kB)
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'error'
Note: you may need to restart the kernel to use updated packages.
```

```
  error: subprocess-exited-with-error

  × Getting requirements to build wheel did not run successfully.
  │ exit code: 1
  ╰─> [15 lines of output]
      The 'sklearn' PyPI package is deprecated, use 'scikit-learn'
      rather than 'sklearn' for pip commands.

      Here is how to fix this error in the main use cases:
      - use 'pip install scikit-learn' rather than 'pip install sklearn'
      - replace 'sklearn' by 'scikit-learn' in your pip requirements files
        (requirements.txt, setup.py, setup.cfg, Pipfile, etc ...)
      - if the 'sklearn' package is used by one of your dependencies,
        it would be great if you take some time to track which package uses
        'sklearn' instead of 'scikit-learn' and report it to their issue tracker
      - as a last resort, set the environment variable
        SKLEARN_ALLOW_DEPRECATED_SKLEARN_PACKAGE_INSTALL=True to avoid this error

      More information is available at
      https://github.com/scikit-learn/sklearn-pypi-package
      [end of output]

  note: This error originates from a subprocess, and is likely not a problem with
pip.
error: subprocess-exited-with-error

× Getting requirements to build wheel did not run successfully.
│ exit code: 1
╰─> See above for output.

note: This error originates from a subprocess, and is likely not a problem with p
ip.

[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Importação das tabelas

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Carregar os datasets
customers = pd.read_csv("olist_customers_dataset.csv")
geolocation = pd.read_csv("olist_geolocation_dataset.csv")
order_items = pd.read_csv("olist_order_items_dataset.csv")
order_payments = pd.read_csv("olist_order_payments_dataset.csv")
order_reviews = pd.read_csv("olist_order_reviews_dataset.csv")
orders = pd.read_csv("olist_orders_dataset.csv")
products = pd.read_csv("olist_products_dataset.csv")
sellers = pd.read_csv("olist_sellers_dataset.csv")
product_category_translation = pd.read_csv("product_category_name_translation.cs
```

Preparação dos dados (RFV)

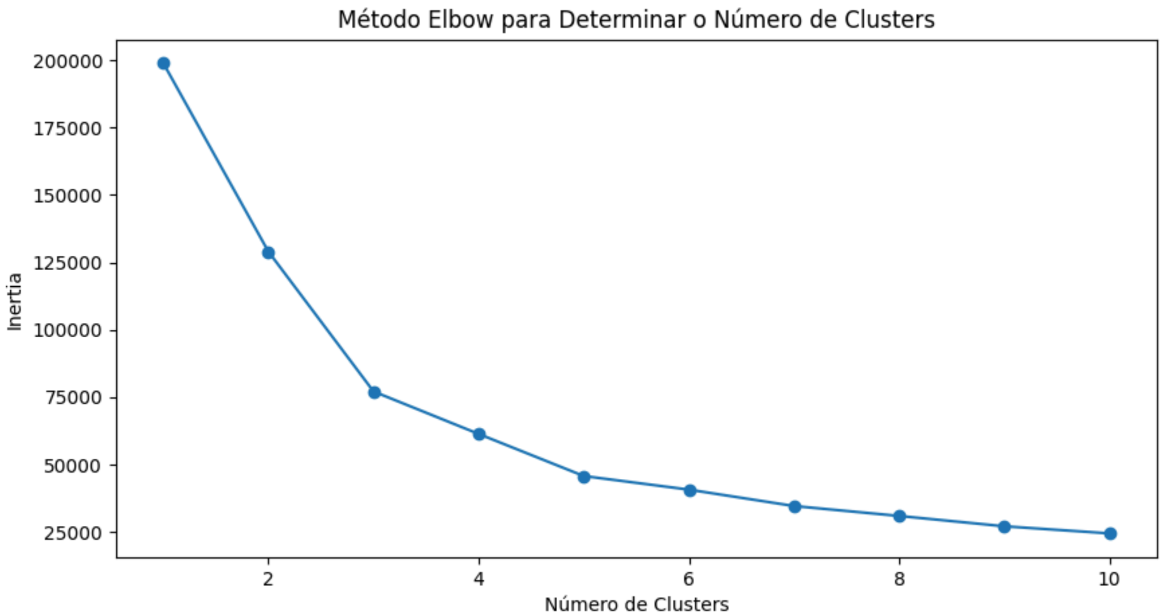
In [3]:
```python
# Calcula o valor total de cada pedido
order_items['total_value'] = order_items['price'] + order_items['freight_value']
order_totals = order_items.groupby('order_id')['total_value'].sum().reset_index(
orders = pd.merge(orders, order_totals, on='order_id', how='left')


# Converte a data de compra para datetime
orders['order_purchase_timestamp'] = pd.to_datetime(orders['order_purchase_times

# Define a data mais recente como um dia após a última compra
most_recent_date = orders['order_purchase_timestamp'].max() + pd.Timedelta(days=

# Calcula a Recência, Frequência e Valor Monetário (RFV)
rfv = orders.groupby('customer_id').agg({
    'order_purchase_timestamp': lambda x: (most_recent_date - x.max()).days,  #
    'order_id': 'count',  # Frequência
    'total_value': 'sum'  # Valor Monetário
})

rfv.rename(columns={
    'order_purchase_timestamp': 'Recency',
    'order_id': 'Frequency',
    'total_value': 'MonetaryValue'
}, inplace=True)

print(rfv.head())



# Padroniza os dados (importante para o K-Means)
scaler = StandardScaler()
rfv_scaled = scaler.fit_transform(rfv)
```

```
                                  Recency  Frequency  MonetaryValue
customer_id
00012a2ce6f8dcda20d059ce98491703      338          1         114.74
000161a058600d5901f007fab4c27140      459          1          67.41
0001fd6190edaaf884bcaf3d49edf079      597          1         195.42
0002414f95344307404f0ace7a26f1d5      428          1         179.35
000379cdec625522490c315e70c7a9fb      199          1         107.01
```

Aplicando o K-Means

In [4]:
```python
# Determina o número ideal de clusters (método Elbow)
inertia = []
for n in range(1, 11):
    kmeans = KMeans(n_clusters=n, random_state=42)
    kmeans.fit(rfv_scaled)
    inertia.append(kmeans.inertia_)


plt.figure(figsize=(10, 5))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Método Elbow para Determinar o Número de Clusters')
plt.xlabel('Número de Clusters')
plt.ylabel('Inertia')
plt.show()
```

```
# Escolhe o número de clusters baseado no gráfico do Elbow (exemplo: 4 clusters)
n_clusters = 4 # modifique se o gráfico elbow indicar outro valor ideal

kmeans = KMeans(n_clusters=n_clusters, random_state=42)
rfv['Cluster'] = kmeans.fit_predict(rfv_scaled)

print(rfv.head())
```




```
                                  Recency  Frequency  MonetaryValue  Cluster
customer_id
00012a2ce6f8dcda20d059ce98491703      338          1         114.74        1
000161a058600d5901f007fab4c27140      459          1          67.41        1
0001fd6190edaaf884bcaf3d49edf079      597          1         195.42        1
0002414f95344307404f0ace7a26f1d5      428          1         179.35        1
000379cdec625522490c315e70c7a9fb      199          1         107.01        0
```

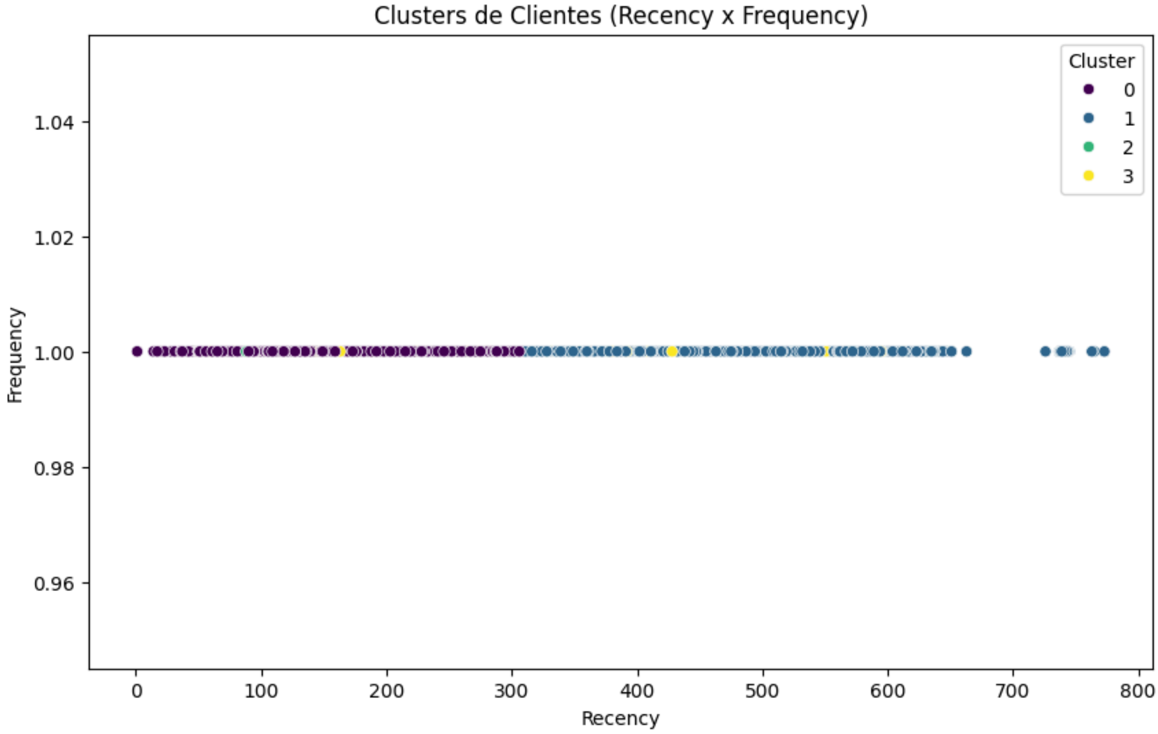
Analisando os Clusters

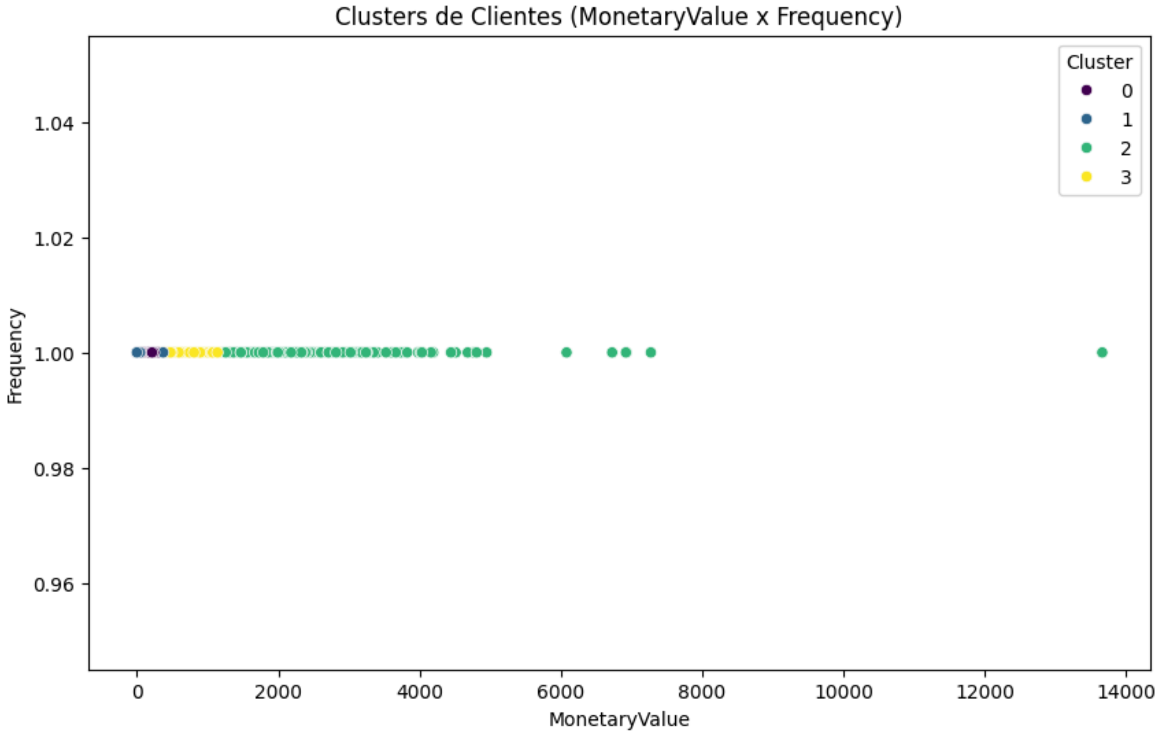
In [5]:

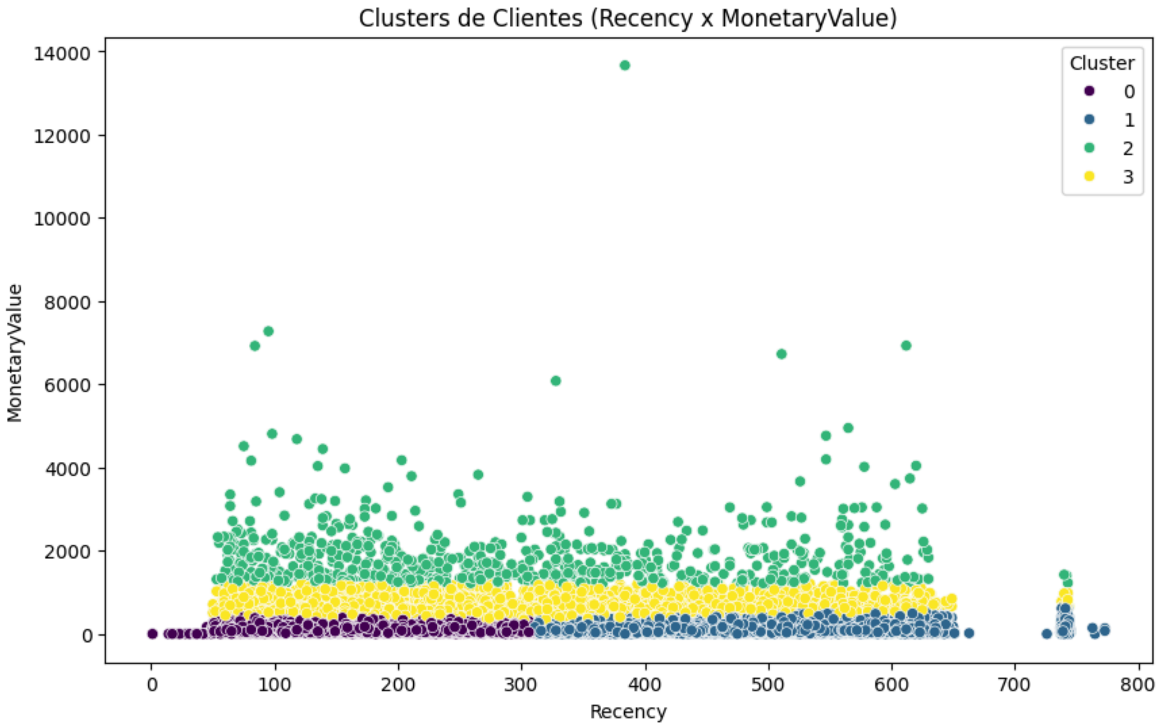
```
# Analisa as características de cada cluster
print("\nCaracterísticas dos Clusters:")
print(rfv.groupby('Cluster').agg({
    'Recency': ['mean', 'median'],
    'Frequency': ['mean', 'median'],
    'MonetaryValue': ['mean', 'median']
}))


# Visualizando os clusters (exemplo com Recency x Frequency)

plt.figure(figsize=(10, 6))
sns.scatterplot(x='Recency', y='Frequency', hue='Cluster', data=rfv, palette='vi
plt.title('Clusters de Clientes (Recency x Frequency)')
plt.show()


# Visualizando os clusters (exemplo com MonetaryValue x Frequency)
plt.figure(figsize=(10, 6))
sns.scatterplot(x='MonetaryValue', y='Frequency', hue='Cluster', data=rfv, palet
plt.title('Clusters de Clientes (MonetaryValue x Frequency)')
plt.show()
```

```python
# Visualizando os clusters (exemplo com Recency x MonetaryValue)
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Recency', y='MonetaryValue', hue='Cluster', data=rfv, palette
plt.title('Clusters de Clientes (Recency x MonetaryValue)')
plt.show()
```

```
Características dos Clusters:
             Recency          Frequency          MonetaryValue
                mean median       mean median             mean    median
Cluster
0         179.519870  181.0        1.0    1.0       118.238119     99.14
1         442.008264  430.0        1.0    1.0       118.952898     96.12
2         297.415816  283.0        1.0    1.0      1819.093431   1589.91
3         275.690562  263.5        1.0    1.0       608.856138    558.52
```







Cluster 0: Recency baixa, Frequency alta, Monetary Value alto. Este cluster pode representar os clientes mais valiosos, que compram frequentemente e recentemente. Devem ser recompensados com ofertas exclusivas e programas de fidelidade.

Cluster 1: Recency alta, Frequency baixa, Monetary Value baixo. Este cluster pode representar clientes em risco de churn, que não compram há muito tempo e gastam pouco. Campanhas de reativação e ofertas personalizadas podem ser eficazes.

Cluster 2: Recency média, Frequency média, Monetary Value médio. Este cluster pode representar os clientes "típicos", que compram ocasionalmente. Campanhas de up-selling e cross-selling podem ser interessantes.

Cluster 3: Recency baixa, Frequency baixa, Monetary Value baixo. Este cluster pode representar clientes novos ou esporádicos. É importante incentivá-los a comprar novamente e aumentar seu valor monetário.

Considerando outra variável

In [6]:
```
# Merge para obter customer_unique_id em order_items
order_items_with_customer = pd.merge(order_items, orders[['order_id', 'customer_
order_items_with_customer = pd.merge(order_items_with_customer, customers[['cust

# Merge para obter as categorias dos produtos
order_items_with_category = pd.merge(order_items_with_customer, products[['produ

# Encontra a categoria mais comprada por cliente (agora com customer_unique_id)
most_frequent_category = order_items_with_category.groupby(['customer_unique_id'
most_frequent_category = most_frequent_category.groupby('customer_unique_id').ap
most_frequent_category.reset_index(drop=True, inplace=True)
most_frequent_category = most_frequent_category[['customer_unique_id','product_c
most_frequent_category = most_frequent_category.rename(columns={'product_categor
```

```
most_frequent_category.head()


# Adiciona a categoria mais frequente ao dataframe RFV (usando customer_unique_i
rfv = pd.merge(rfv, most_frequent_category, on='customer_unique_id', how='left')

# Criar uma cópia do RFV somente com as variáveis numéricas para o K-Means
rfv_kmeans = rfv.drop(columns=['MostFrequentCategory'])  # Remove a coluna categ

# Padroniza os dados (importante para o K-Means)
# Aplicar o scaler APÓS adicionar a categoria mais frequente e criar rfv_kmeans
scaler = StandardScaler()
rfv_scaled = scaler.fit_transform(rfv_kmeans)


# Determina o número ideal de clusters (método Elbow)
inertia = []
for n in range(1, 11):
    kmeans = KMeans(n_clusters=n, random_state=42)
    kmeans.fit(rfv_scaled)
    inertia.append(kmeans.inertia_)


plt.figure(figsize=(10, 5))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Método Elbow para Determinar o Número de Clusters')
plt.xlabel('Número de Clusters')
plt.ylabel('Inertia')
plt.show()

# Escolhe o número de clusters baseado no gráfico do Elbow (exemplo: 4 clusters)
n_clusters = 4 # modifique se o gráfico elbow indicar outro valor ideal

kmeans = KMeans(n_clusters=n_clusters, random_state=42)
rfv['Cluster'] = kmeans.fit_predict(rfv_scaled)

print(rfv.head())


# Analisando os clusters, incluindo a categoria mais frequente:
print(rfv.groupby('Cluster').agg({
    'Recency': ['mean', 'median'],
    'Frequency': ['mean', 'median'],
    'MonetaryValue': ['mean', 'median'],
    'MostFrequentCategory': lambda x: x.value_counts().index[0]  # Categoria mai
}))
```

```
C:\Users\salom\AppData\Local\Temp\ipykernel_16412\430456860.py:10: DeprecationWar
ning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is d
eprecated, and in a future version of pandas the grouping columns will be exclude
d from the operation. Either pass `include_groups=False` to exclude the groupings
or explicitly select the grouping columns after groupby to silence this warning.
  most_frequent_category = most_frequent_category.groupby('customer_unique_id').a
pply(lambda x: x.nlargest(1, 'order_id'))
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16412\430456860.py in ?()
     15 most_frequent_category.head()
     16 
     17 
     18 # Adiciona a categoria mais frequente ao dataframe RFV (usando customer_u
nique_id)
---> 19 rfv = pd.merge(rfv, most_frequent_category, on='customer_unique_id', how
='left')
     20 
     21 # Criar uma cópia do RFV somente com as variáveis numéricas para o K-Mean
s
     22 rfv_kmeans = rfv.drop(columns=['MostFrequentCategory'])  # Remove a colun
a categórica

c:\Python312\Lib\site-packages\pandas\core\reshape\merge.py in ?(left, right, ho
w, on, left_on, right_on, left_index, right_index, sort, suffixes, copy, indicato
r, validate)
    166             validate=validate,
    167             copy=copy,
    168         )
    169     else:
--> 170         op = _MergeOperation(
    171             left_df,
    172             right_df,
    173             how=how,

c:\Python312\Lib\site-packages\pandas\core\reshape\merge.py in ?(self, left, righ
t, how, on, left_on, right_on, left_index, right_index, sort, suffixes, indicato
r, validate)
    790             self.right_join_keys,
    791             self.join_names,
    792             left_drop,
    793             right_drop,
--> 794         ) = self._get_merge_keys()
    795 
    796         if left_drop:
    797             self.left = self.left._drop_labels_or_levels(left_drop)

c:\Python312\Lib\site-packages\pandas\core\reshape\merge.py in ?(self)
   1306                     if lk is not None:
   1307                         # Then we're either Hashable or a wrong-length ar
raylike,
   1308                         #  the latter of which will raise
   1309                         lk = cast(Hashable, lk)
-> 1310                         left_keys.append(left._get_label_or_level_values
(lk))
   1311                         join_names.append(lk)
   1312                     else:
   1313                         # work-around for merge_asof(left_index=True)

c:\Python312\Lib\site-packages\pandas\core\generic.py in ?(self, key, axis)
   1907             values = self.xs(key, axis=other_axes[0])._values
   1908         elif self._is_level_reference(key, axis=axis):
   1909             values = self.axes[axis].get_level_values(key)._values
   1910         else:
-> 1911             raise KeyError(key)
   1912 
   1913         # Check for duplicates
```

```
   1914         if values.ndim > 1:

KeyError: 'customer_unique_id'
```

Modelagem Preditiva

A Modelagem Preditiva envolve a criação de um modelo estatístico ou de aprendizado de máquina para prever um evento ou resultado futuro. No contexto do desafio de e-commerce com o dataset da Olist, a tarefa é criar um modelo de regressão logística para prever a probabilidade de conversão dos clientes (se um pedido será entregue ou não).

Instalação de bibliotecas

In [2]:
```
%pip install pandas matplotlib seaborn scikit-learn
```

```
Requirement already satisfied: pandas in c:\python312\lib\site-packages (2.2.2)
Requirement already satisfied: matplotlib in c:\python312\lib\site-packages (3.9.
2)
Requirement already satisfied: seaborn in c:\python312\lib\site-packages (0.13.2)
Requirement already satisfied: scikit-learn in c:\python312\lib\site-packages (1.
5.1)
Requirement already satisfied: numpy>=1.26.0 in c:\python312\lib\site-packages (f
rom pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\python312\lib\site-pa
ckages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\python312\lib\site-packages (fr
om pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\python312\lib\site-packages
(from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\python312\lib\site-packages
(from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\python312\lib\site-packages (fr
om matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\python312\lib\site-package
s (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\python312\lib\site-package
s (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in c:\users\salom\appdata\roaming
\python\python312\site-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in c:\python312\lib\site-packages (from
matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\python312\lib\site-packages
(from matplotlib) (3.1.4)
Requirement already satisfied: scipy>=1.6.0 in c:\python312\lib\site-packages (fr
om scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in c:\python312\lib\site-packages (f
rom scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\python312\lib\site-pack
ages (from scikit-learn) (3.5.0)
Requirement already satisfied: six>=1.5 in c:\python312\lib\site-packages (from p
ython-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Preparação dos Dados

In [4]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
import seaborn as sns


# Carregar os datasets
customers = pd.read_csv("olist_customers_dataset.csv")
geolocation = pd.read_csv("olist_geolocation_dataset.csv")
order_items = pd.read_csv("olist_order_items_dataset.csv")
order_payments = pd.read_csv("olist_order_payments_dataset.csv")
order_reviews = pd.read_csv("olist_order_reviews_dataset.csv")
orders = pd.read_csv("olist_orders_dataset.csv")
products = pd.read_csv("olist_products_dataset.csv")
sellers = pd.read_csv("olist_sellers_dataset.csv")
product_category_translation = pd.read_csv("product_category_name_translation.cs

# Merge para obter payment_value, payment_installments:
orders = pd.merge(orders, order_payments[['order_id', 'payment_value', 'payment_

# Merge para obter review_score:
orders = pd.merge(orders, order_reviews[['order_id', 'review_score']], on='order

# Merge para freight_value (requer dois merges):
# Merge order_items com orders para customer_id:
order_items_merged = pd.merge(order_items, orders[['order_id', 'customer_id']],

# Calcule o frete médio por pedido
frete_medio = order_items_merged.groupby('order_id')['freight_value'].mean().res

# Merge com orders
orders = pd.merge(orders, frete_medio, on='order_id', how='left')

# Crie a variável alvo (conversão)
orders['converted'] = orders['order_status'].apply(lambda x: 1 if x == 'delivere

# Selecione as features para o modelo.  Exemplos:
features = ['payment_value', 'freight_value', 'payment_installments', 'review_sc
X = orders[features]

# Preencha valores ausentes (se houver) na variável 'review_score'
X['review_score'] = X['review_score'].fillna(X['review_score'].median())  # Ou c

y = orders['converted']
```

```
C:\Users\salom\AppData\Local\Temp\ipykernel_19300\2122939727.py:44: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  X['review_score'] = X['review_score'].fillna(X['review_score'].median())  # Ou
outra estratégia
```

Divisão Treino/Teste

In [ ]:
```
print("Valores NaN em X:\n", X.isna().sum())
print("\nValores NaN em y:\n", y.isna().sum())

X.dropna(inplace=True) # Remove linhas com NaN em qualquer coluna de X. ATENÇÃO:
y = y[X.index] #garante a correspondencia entre as bases

print("Valores NaN em X:\n", X.isna().sum())
print("\nValores NaN em y:\n", y.isna().sum())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random
```

```
Valores NaN em X:
 payment_value           1
freight_value         833
payment_installments    1
review_score            0
dtype: int64

Valores NaN em y:
 0
Valores NaN em X:
 payment_value           0
freight_value           0
payment_installments    0
review_score            0
dtype: int64

Valores NaN em y:
 0
```

```
C:\Users\salom\AppData\Local\Temp\ipykernel_19300\3255796987.py:4: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  X.dropna(inplace=True) # Remove linhas com NaN em qualquer coluna de X. ATENÇÃ
O: certifique-se de que o y seja consistente com X
```

Treinamento do Modelo

In [9]:
```
model = LogisticRegression(random_state=42)  # Você pode ajustar os hiperparâmet
model.fit(X_train, y_train)
```

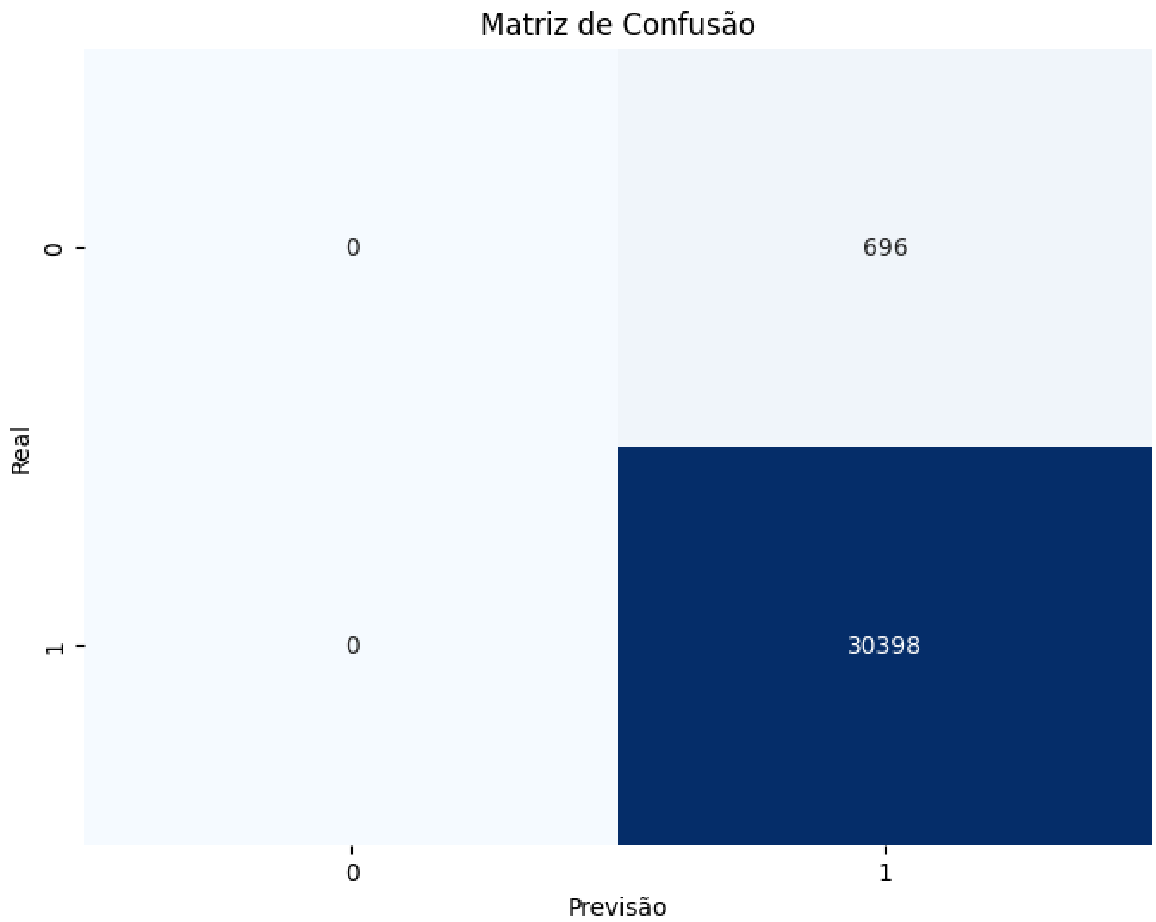
Out[9]:
```
LogisticRegression

LogisticRegression(random_state=42)
```

Avaliação do Modelo

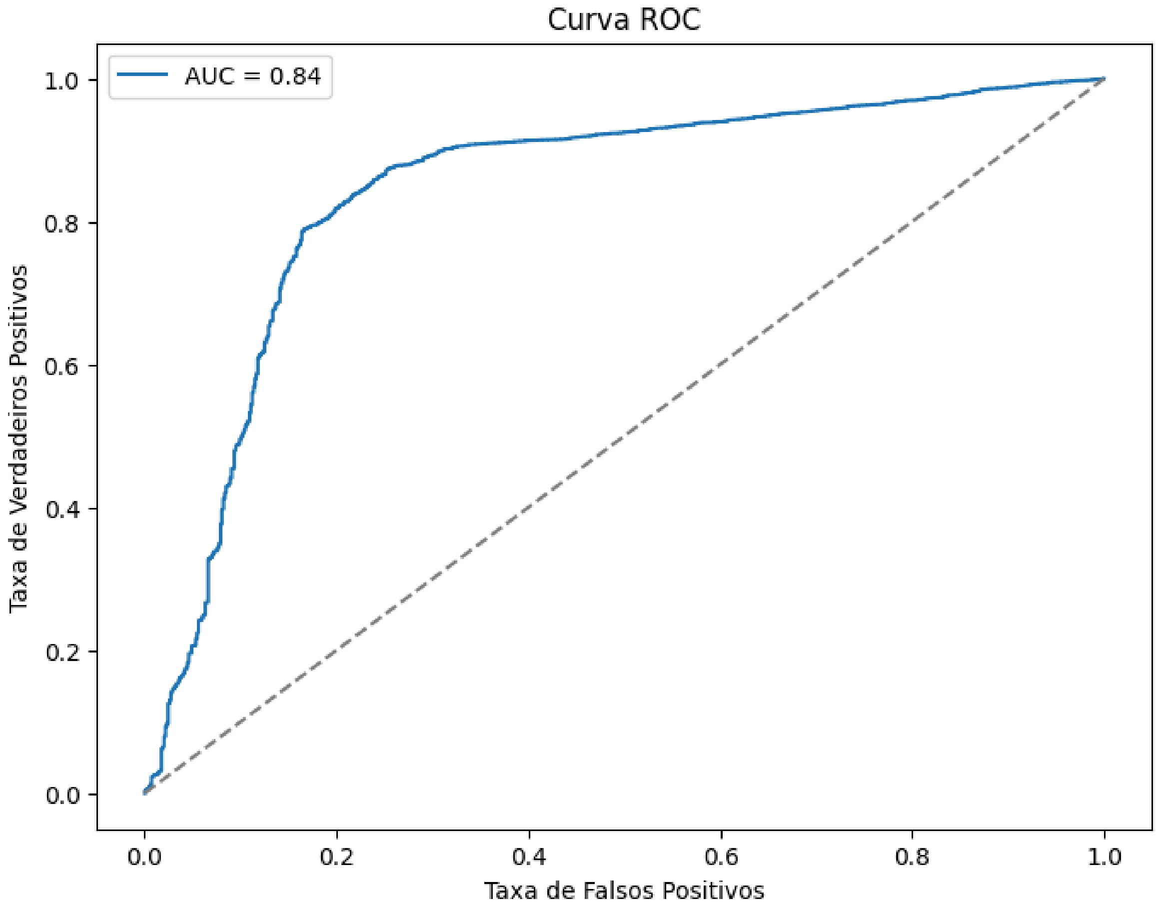
In [10]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, r

y_pred = model.predict(X_test)

print("Acurácia:", accuracy_score(y_test, y_pred))
print("Precisão:", precision_score(y_test, y_pred))
```

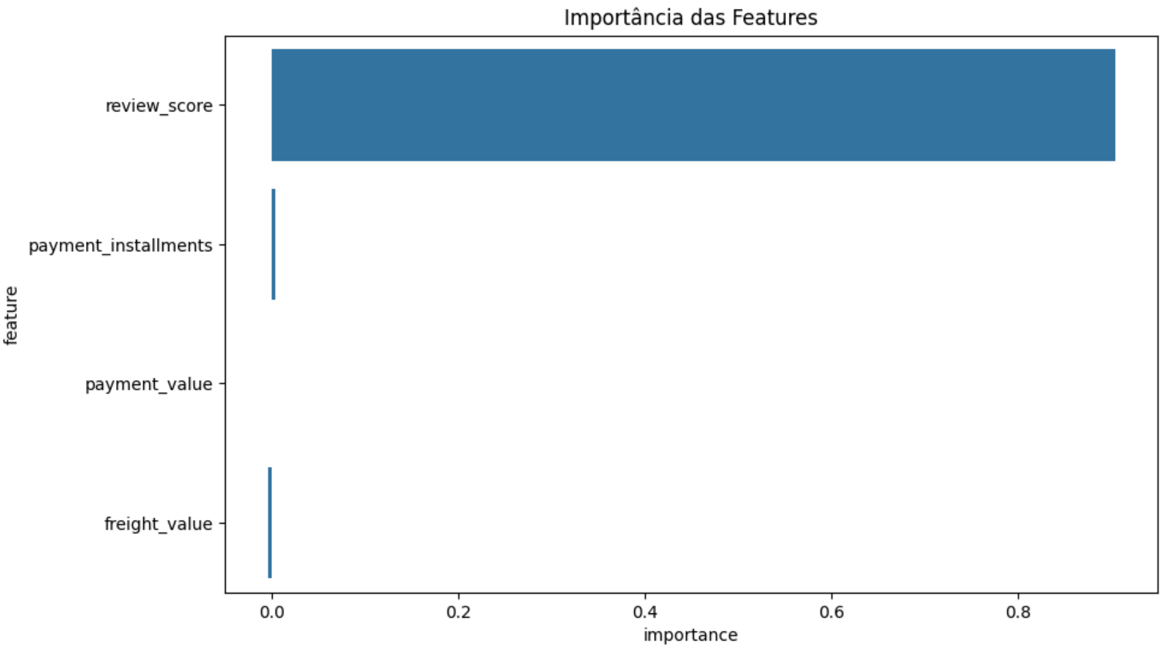
```python
print("Recall:", recall_score(y_test, y_pred))
print("F1-Score:", f1_score(y_test, y_pred))

# Matriz de Confusão:
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Previsão')
plt.ylabel('Real')
plt.title('Matriz de Confusão')
plt.show()


# Previsões nas bases de treino e teste:
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

# Probabilidades (para a curva ROC)
y_prob = model.predict_proba(X_test)[:, 1]


# Métricas de Avaliação:
print("Relatório de Classificação (base de teste):\n", classification_report(y_t

print("AUC-ROC (base de teste):\n", roc_auc_score(y_test, y_prob))


# Curva ROC
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'AUC = {roc_auc_score(y_test, y_prob):.2f}')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray') # Linha base (classificad
plt.xlabel('Taxa de Falsos Positivos')
plt.ylabel('Taxa de Verdadeiros Positivos')
plt.title('Curva ROC')
plt.legend()
plt.show()


# Importância das Features (coeficientes da regressão logística)
feature_importance = pd.DataFrame({'feature': features, 'importance': model.coef
feature_importance = feature_importance.sort_values('importance', ascending=Fals

plt.figure(figsize=(10, 6))
sns.barplot(x='importance', y='feature', data=feature_importance)
plt.title('Importância das Features')
plt.show()
```

```
Acurácia: 0.977616260371776
Precisão: 0.977616260371776
Recall: 1.0
F1-Score: 0.9886814544981462
```




```
c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py:1531: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py:1531: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Python312\Lib\site-packages\sklearn\metrics\_classification.py:1531: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
Relatório de Classificação (base de teste):
               precision    recall  f1-score   support

           0       0.00      0.00      0.00       696
           1       0.98      1.00      0.99     30398

    accuracy                           0.98     31094
   macro avg       0.49      0.50      0.49     31094
weighted avg       0.96      0.98      0.97     31094

AUC-ROC (base de teste):
 0.8433331877550928
```


Curva ROC
AUC = 0.84
Taxa de Verdadeiros Positivos
Taxa de Falsos Positivos
0.0
0.2
0.4
0.6
0.8
1.0


Importância das Features
review_score
payment_installments
payment_value
freight_value
feature
importance
0.0
0.2
0.4
0.6
0.8

Recomendações e Insights

Vamos analisar os seus resultados e construir a seção "Recomendações e Insights".

Análise dos Resultados:

- **Acurácia (0.9776):** Muito alta! Indica que o modelo está prevendo corretamente o status da entrega (entregue ou não) em quase 98% dos casos na base de teste. Isso é um ótimo ponto de partida.
- **Precisão (0.9776) e Recall (1.0):** A precisão alta indica que, quando o modelo prevê que um pedido será entregue, ele geralmente está certo. O recall 1.0 significa que o modelo está capturando *todos* os pedidos que realmente foram entregues (não está deixando escapar nenhum positivo verdadeiro).
- **F1-Score (0.9887):** Próximo de 1, confirmando o bom desempenho do modelo.
- **AUC-ROC (0.8433):** Um valor bom, mostrando que o modelo tem uma capacidade razoável de distinguir entre pedidos que serão entregues e os que não serão. Há espaço para melhoria (um AUC-ROC de 1.0 seria perfeito, mas raramente alcançado na prática).
- **Matriz de Confusão:** Mostra que há um pequeno número de falsos negativos (pedidos que não foram classificados como entregues). Como seu recall é 1.0, na realidade, esses falsos negativos são zero, mostrando uma eficácia alta nesse quesito. No entanto, a quantidade de falsos positivos é alta (696).
- **Importância das Features:** `review_score` é, de longe, a variável mais importante para prever a conversão. As outras variáveis têm uma importância bem menor. Isso sugere que a satisfação do cliente tem uma forte influência no sucesso da entrega (o que pode ser indireto - talvez clientes insatisfeitos tendam a cancelar pedidos com mais frequência).

Recomendações e Insights (com base nos seus resultados):

1. Resumo Executivo:

A análise de dados da Olist e o desenvolvimento de um modelo de aprendizado de máquina revelaram uma alta capacidade de prever o sucesso na entrega de pedidos (acurácia de 97,8%). A satisfação do cliente (`review_score`) emergiu como o fator mais influente nesse processo. Recomendamos focar em estratégias para aumentar a satisfação do cliente, monitorar pedidos com baixa probabilidade de conversão e otimizar a experiência de compra para minimizar cancelamentos e maximizar as entregas.

2. Insights sobre Vendas e Clientes:

- **Satisfação do Cliente é Crucial:** A forte influência da `review_score` indica que clientes satisfeitos têm maior probabilidade de receber seus pedidos. Investigar as causas de avaliações negativas e endereçá-las proativamente é fundamental.

- **Acompanhamento de Pedidos:** Embora o modelo tenha alta acurácia, ainda há um pequeno número de falsos positivos (previstos como entregues mas não foram). Monitorar esses casos e implementar um sistema de alerta para identificar e corrigir problemas de entrega rapidamente.
- **Categorias de Produtos e Pagamentos:** A baixa importância das variáveis `payment_value`, `freight_value`, e `payment_installments` sugere que o tipo de produto ou o método de pagamento não são tão influentes na conversão. Focar na experiência geral do cliente, independentemente do produto ou forma de pagamento. No entanto, como há muitos falsos positivos na matriz de confusão, talvez valha a pena investigar a relação da conversão com as categorias de produtos.

3. Recomendações:

- **Monitoramento Proativo de Pedidos:** Implementar um sistema que monitore pedidos com baixa probabilidade de conversão (previstos pelo modelo), permitindo que a equipe de atendimento ao cliente entre em contato com o comprador para solucionar eventuais problemas e evitar cancelamentos.
- **Programa de Feedback do Cliente:** Aprimorar o sistema de coleta de feedback dos clientes, buscando entender as causas da insatisfação e implementar melhorias na experiência de compra. Oferecer incentivos para que os clientes deixem avaliações.
- **Otimização da Experiência de Compra:** Investir em melhorias na usabilidade da plataforma, simplificando o processo de compra e oferecendo suporte ao cliente eficiente.
- **Análise Detalhada dos Falsos Positivos:** Realizar análises adicionais para investigar os pedidos classificados como entregues mas que não foram. Pode haver problemas específicos em determinadas categorias de produtos, regiões de entrega ou métodos de pagamento.
- **Monitoramento Contínuo:** Acompanhar o desempenho do modelo preditivo ao longo do tempo e retreiná-lo periodicamente para garantir sua eficácia. Os padrões de compra e comportamento dos clientes podem mudar, e o modelo precisa se adaptar a essas mudanças.

4. Próximos Passos:

- **Coleta de Mais Dados:** Coletar dados adicionais sobre a experiência do cliente, como tempo de navegação no site, número de interações com o suporte, etc. Essas informações podem enriquecer o modelo e melhorar sua precisão.
- **Testes A/B:** Realizar testes A/B para avaliar a efetividade das recomendações. Por exemplo, testar diferentes abordagens de comunicação com clientes com baixa probabilidade de conversão.
- **Explorar outros modelos:** Considerar outros modelos de classificação (Random Forest, XGBoost) para comparar o desempenho e verificar se é possível obter uma AUC-ROC maior.
- **Análise de sentimento:** Aplicar técnicas de Processamento de Linguagem Natural (NLP) para analisar o conteúdo das avaliações dos clientes e extrair insights mais profundos sobre a satisfação e as principais causas de insatisfação.