untitled125

August 25, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
from sklearn.preprocessing import StandardScaler
import datetime as dt
```

[3]:
```
#Carregando dados
df = pd.read_csv("Data - data (2).csv.csv.csv")
```

[4]:
```
#Lendos os dados
df.head()
```

[4]:
```
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

       InvoiceDate  UnitPrice  CustomerID         Country
0  12/1/2010 08:26       2.55     17850.0  United Kingdom
1  12/1/2010 08:26       3.39     17850.0  United Kingdom
2  12/1/2010 08:26       2.75     17850.0  United Kingdom
3  12/1/2010 08:26       3.39     17850.0  United Kingdom
4  12/1/2010 08:26       3.39     17850.0  United Kingdom
```

[5]:
```
#Vendo informações gerais
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
```

```
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

[6]:
```
#Vendo infos estatisticas
df.describe()
```

[6]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611121	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

[7]:
```
#Verificando dados nulos
df.isna().sum()
```

[7]:
```
InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID     135080
Country            0
dtype: int64
```

[8]:
```
#Vendo dados nulos na coluna CustomerID
df[df['CustomerID'].isna()]
```

[8]:

	InvoiceNo	StockCode	Description	Quantity	\
622	536414	22139	NaN	56	
1443	536544	21773	DECORATIVE ROSE BATHROOM BOTTLE	1	
1444	536544	21774	DECORATIVE CATS BATHROOM BOTTLE	2	
1445	536544	21786	POLKADOT RAIN HAT	4	
1446	536544	21787	RAIN PONCHO RETROSPOT	2	
…	…	…	…	…	
541536	581498	85099B	JUMBO BAG RED RETROSPOT	5	
541537	581498	85099C	JUMBO BAG BAROQUE BLACK WHITE	4	

```
541538    581498     85150    LADIES & GENTLEMEN METAL SIGN         1
541539    581498     85174                S/4 CACTI CANDLES         1
541540    581498       DOT                   DOTCOM POSTAGE         1

            InvoiceDate  UnitPrice  CustomerID         Country
622     12/1/2010 11:52       0.00         NaN  United Kingdom
1443    12/1/2010 14:32       2.51         NaN  United Kingdom
1444    12/1/2010 14:32       2.51         NaN  United Kingdom
1445    12/1/2010 14:32       0.85         NaN  United Kingdom
1446    12/1/2010 14:32       1.66         NaN  United Kingdom
…                   …          …           …              …
541536  12/9/2011 10:26       4.13         NaN  United Kingdom
541537  12/9/2011 10:26       4.13         NaN  United Kingdom
541538  12/9/2011 10:26       4.96         NaN  United Kingdom
541539  12/9/2011 10:26      10.79         NaN  United Kingdom
541540  12/9/2011 10:26    1714.17         NaN  United Kingdom

[135080 rows x 8 columns]
```

[9]:
```
#Dropando valores nulos CustomerID
df.dropna(subset=['CustomerID'],inplace=True)
```

Identidificando dados nulos ou menores que 0

[10]:
```
#Filtrando dados nulos ou <0 na coluna UnitPrice
df.filter(like='UnitPrice')
filtred_UnitedPrice = df[df['UnitPrice'].isna() | (df['UnitPrice'] < 0)]
filtred_UnitedPrice.head()
```

[10]:
```
Empty DataFrame
Columns: [InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice,
CustomerID, Country]
Index: []
```

Filtrando valores da coluna UnitPrice acima de 0

[11]:
```
#Filtrando preços somente aciima de 0
df = df[df['UnitPrice'] > 0]
df.head()
```

[11]:
```
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

       InvoiceDate  UnitPrice  CustomerID         Country
```

```
0  12/1/2010 08:26       2.55     17850.0  United Kingdom
1  12/1/2010 08:26       3.39     17850.0  United Kingdom
2  12/1/2010 08:26       2.75     17850.0  United Kingdom
3  12/1/2010 08:26       3.39     17850.0  United Kingdom
4  12/1/2010 08:26       3.39     17850.0  United Kingdom
```

[12]:
```
#Filtrando dados nulos ou menor que 0 na coluna Quantity
df[df['Quantity'].isna() | (df['Quantity'] < 0)]

#Deixando o dataset somente com valores em Quantity acima de 0
df = df[df['Quantity'].isna() | (df['Quantity'] > 0)]
df.head()
```

[12]:
```
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

       InvoiceDate  UnitPrice  CustomerID         Country
0  12/1/2010 08:26       2.55     17850.0  United Kingdom
1  12/1/2010 08:26       3.39     17850.0  United Kingdom
2  12/1/2010 08:26       2.75     17850.0  United Kingdom
3  12/1/2010 08:26       3.39     17850.0  United Kingdom
4  12/1/2010 08:26       3.39     17850.0  United Kingdom
```

[13]:
```
df.isna().sum()
```

[13]:
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

Tratando dados duplicados

[14]:
```
#Identificar linhas duplicadas
df[df.duplicated()]

#Excluindo valores duplicados
df = df.drop_duplicates()
```

[15]:
```
#Vendo tipos dos dados
df.dtypes
```

[15]:
```
InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID     float64
Country         object
dtype: object
```

Alterando os tipos de dados de colunas específicas

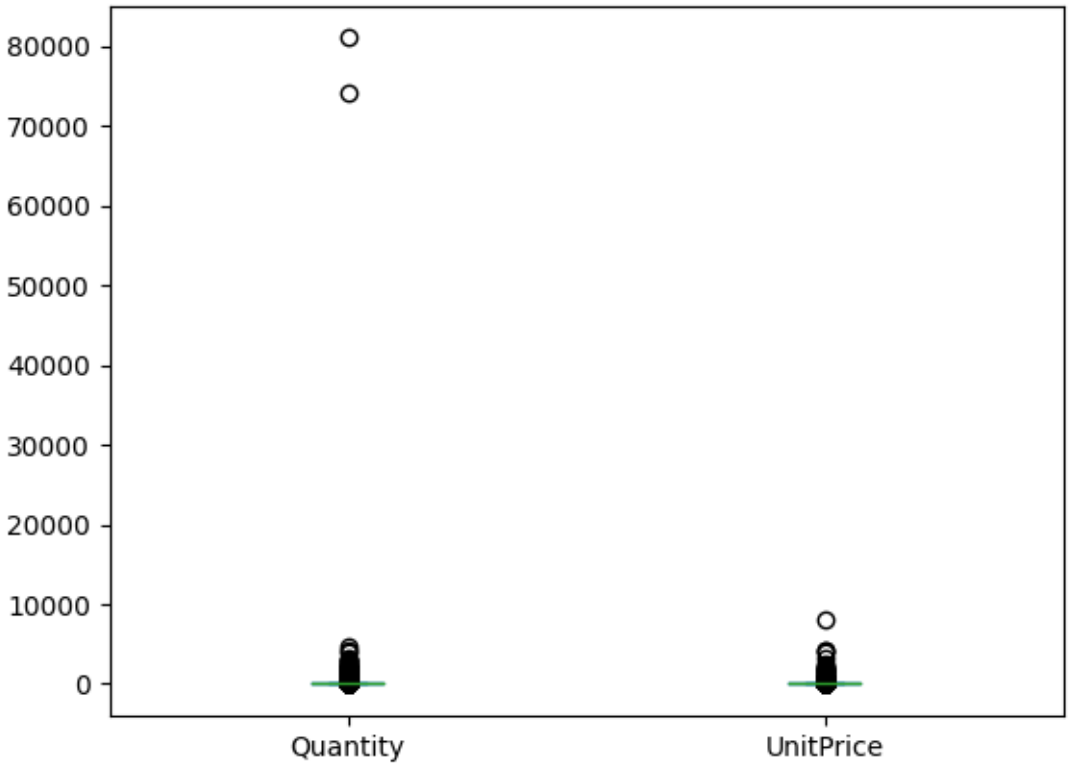
[16]:
```
#Mudando o tipo de dado da coluna InvoiceDate para data
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'],format="%m/%d/%Y %H:%M")
```

[17]:
```
#Mudando o tipo de dado da coluna CustomerID para int
df = df.astype({"CustomerID": object})
```

Visualizando outliers

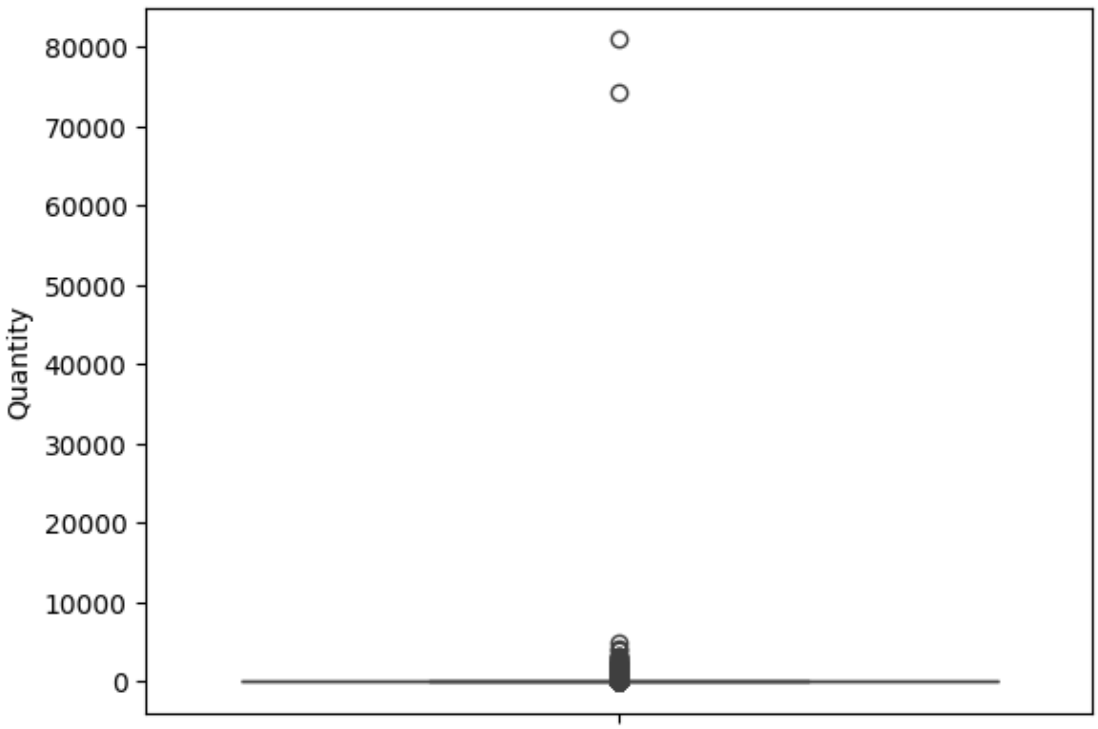
[18]:
```
# Visualizando outliers
df.plot.box()
```

[18]:
```
<Axes: >
```



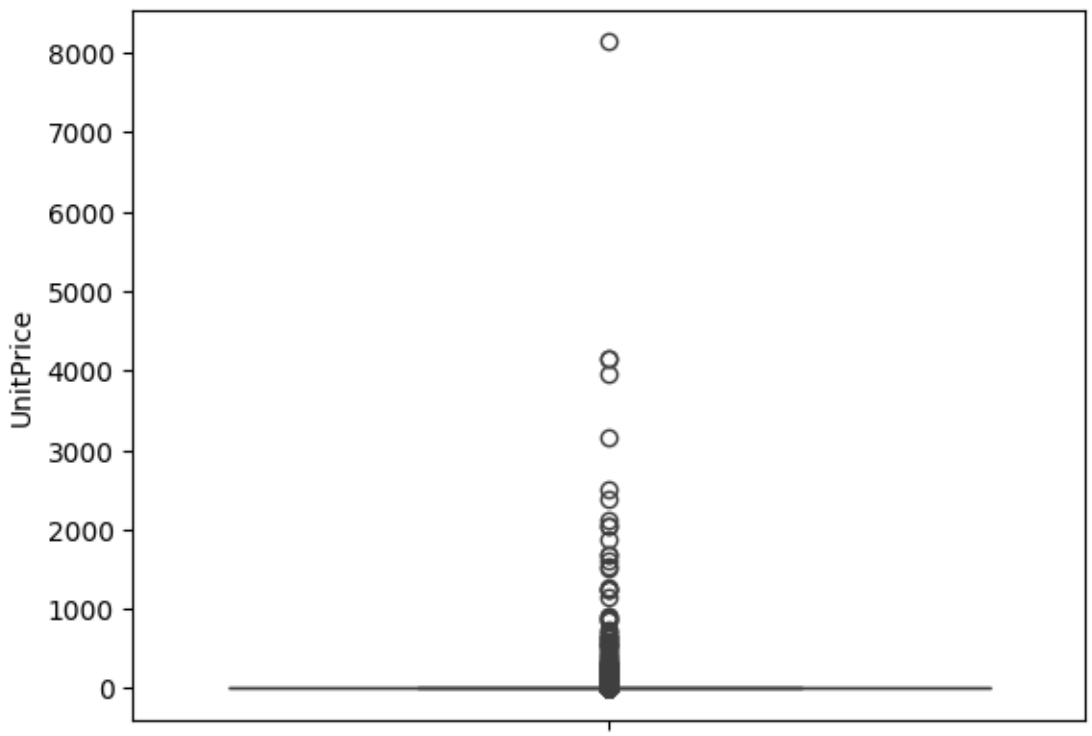
[19]:
```
# Visualizando outliers na coluna Quantity
sns.boxplot(df['Quantity'])
```

[19]:
```
<Axes: ylabel='Quantity'>
```

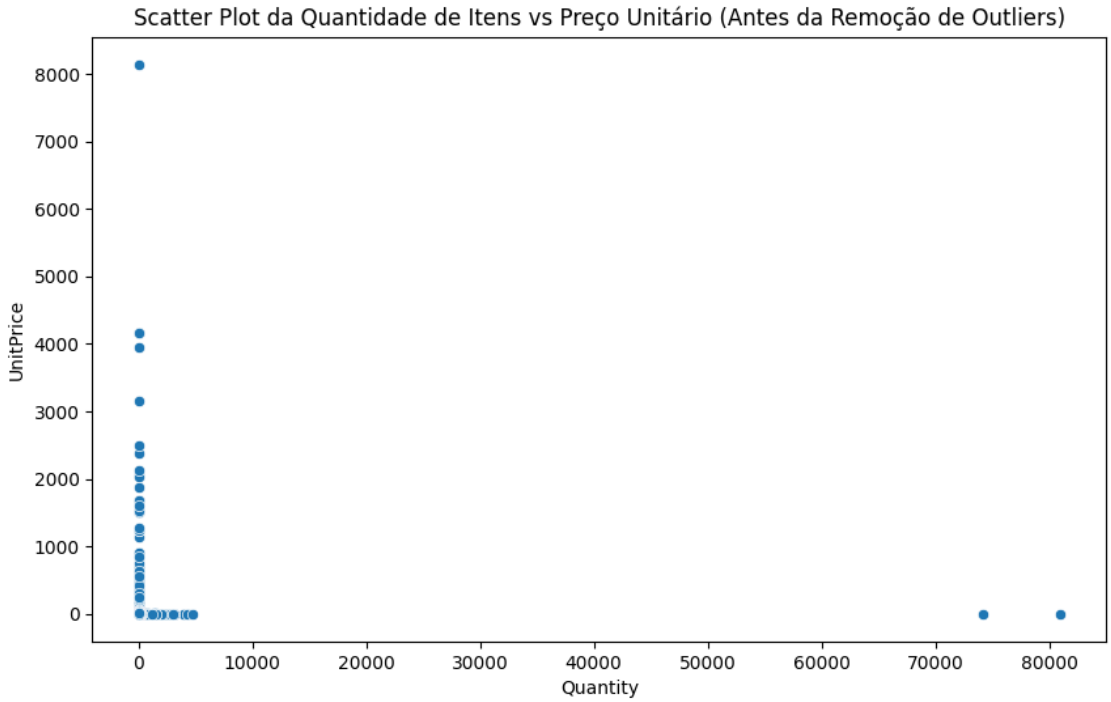


[20]:
```
# Visualizando outliers na coluna UnitedPrice
sns.boxplot(df['UnitPrice'])
```

[20]:
```
<Axes: ylabel='UnitPrice'>
```



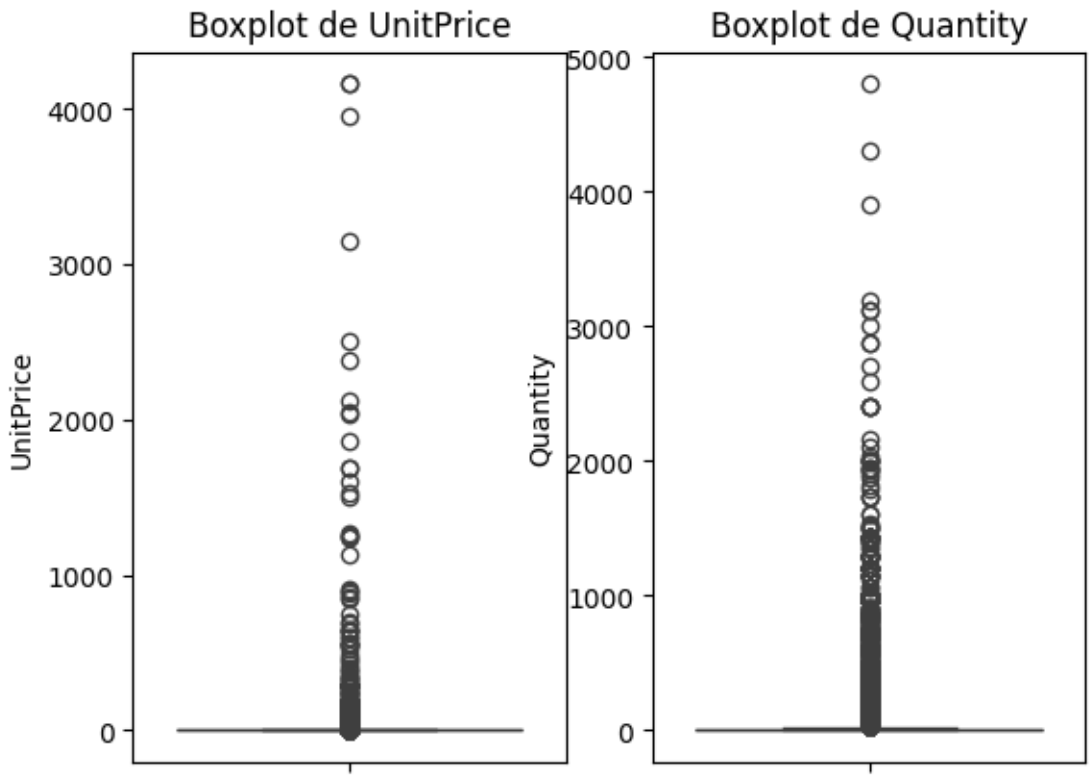
[21]:
```
# Visualizar os outliers usando scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Quantity', y='UnitPrice', data=df)
plt.title('Scatter Plot da Quantidade de Itens vs Preço Unitário (Antes da␣
 ↪Remoção de Outliers)')
plt.show()
```



Removendo outliers extremos, filtrando os valores

[22]:
```
# Remover outliers extremos (quantidade superior a 10.000 e preço unitário
↪maior que 5.000)
df = df[(df['Quantity'] <= 10000) & (df['UnitPrice'] <= 5000)]
```

[23]:
```
# Gráfico UnitPrice
plt.subplot(1, 2, 1)
sns.boxplot(y=df['UnitPrice'])
plt.title('Boxplot de UnitPrice')

# Gráfico Quantity
plt.subplot(1, 2, 2)
sns.boxplot(y=df['Quantity'])
plt.title('Boxplot de Quantity')

plt.show()
```



Nova coluna - Preço total de compra

[24]:
```
# Criando coluna adicional do preço total da compra
df['Total_price'] = df['Quantity']*df['UnitPrice']
df.head()
```

```
<ipython-input-24-e04bdf41b82a>:2: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Total_price'] = df['Quantity']*df['UnitPrice']
```

[24]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Total_price
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6					
1	536365	71053	WHITE METAL LANTERN	6					
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8					
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6					
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6					

```
0 2010-12-01 08:26:00       2.55    17850.0  United Kingdom        15.30
1 2010-12-01 08:26:00       3.39    17850.0  United Kingdom        20.34
2 2010-12-01 08:26:00       2.75    17850.0  United Kingdom        22.00
3 2010-12-01 08:26:00       3.39    17850.0  United Kingdom        20.34
4 2010-12-01 08:26:00       3.39    17850.0  United Kingdom        20.34
```

Ultima data de compra do dataset

[25]:
```
# Calculando a data da ultima compra do df
df['InvoiceDate'].max()
```

[25]:
```
Timestamp('2011-12-09 12:50:00')
```

Plotando gráficos

[26]:
```
#Top 10 países com maior valor em vendas
top10_sales_country = df.groupby('Country')['Total_price'].sum().
 ↪sort_values(ascending=False).head(10)
top10_sales_country

#Top 10 produtos mais vendidos
top10_sales_product = df.groupby('Description')['Total_price'].sum().
 ↪sort_values(ascending=False).head(10)
top10_sales_product
```

[26]:
```
Description
REGENCY CAKESTAND 3 TIER              142264.75
WHITE HANGING HEART T-LIGHT HOLDER    100392.10
JUMBO BAG RED RETROSPOT                85040.54
POSTAGE                                69661.21
PARTY BUNTING                          68785.23
ASSORTED COLOUR BIRD ORNAMENT          56413.03
Manual                                 53419.93
RABBIT NIGHT LIGHT                     51251.24
CHILLI LIGHTS                          46265.11
PAPER CHAIN KIT 50'S CHRISTMAS         42584.13
Name: Total_price, dtype: float64
```

Top 10 países com maior valor em vendas

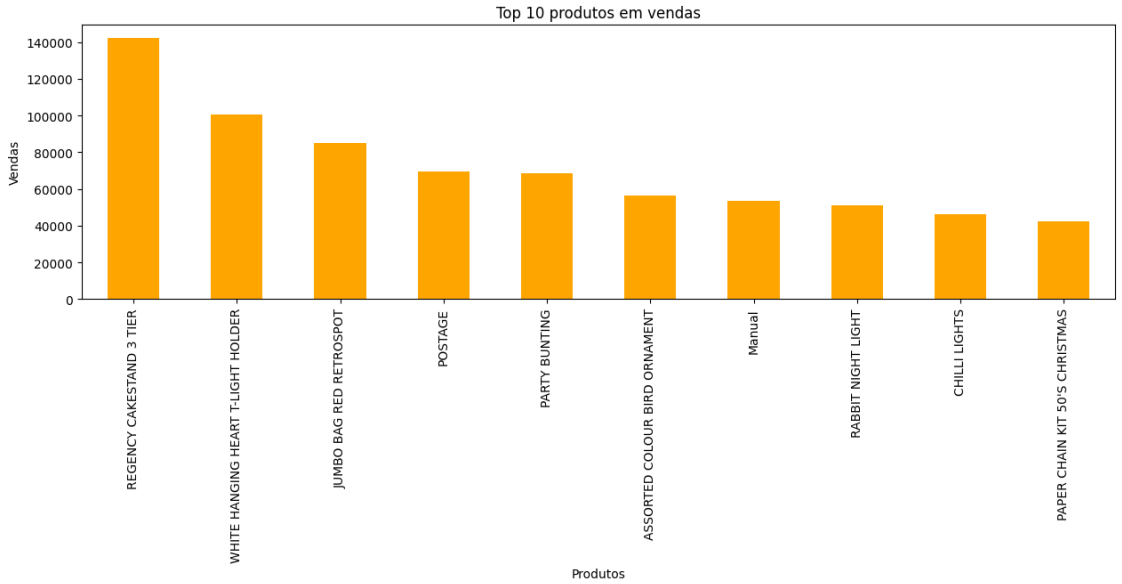
[27]:
```
#Gráfico top países produtos em vendas
fig = plt.figure(figsize=(15,4))

top10_sales_country.plot(kind='bar',color='blue')

plt.title('Top 10 vendas por país')
plt.xlabel('Países')
plt.ylabel('Vendas')
```

```
plt.tight_layout()

plt.xticks(rotation=45);
```

Top 10 produtos mais vendidos

[28]:
```
#Gráfico top 10 produtos em vendas
fig = plt.figure(figsize=(15,4))

top10_sales_product.plot(kind='bar',color='orange')

plt.title('Top 10 produtos em vendas')
plt.xlabel('Produtos')
plt.ylabel('Vendas')
```

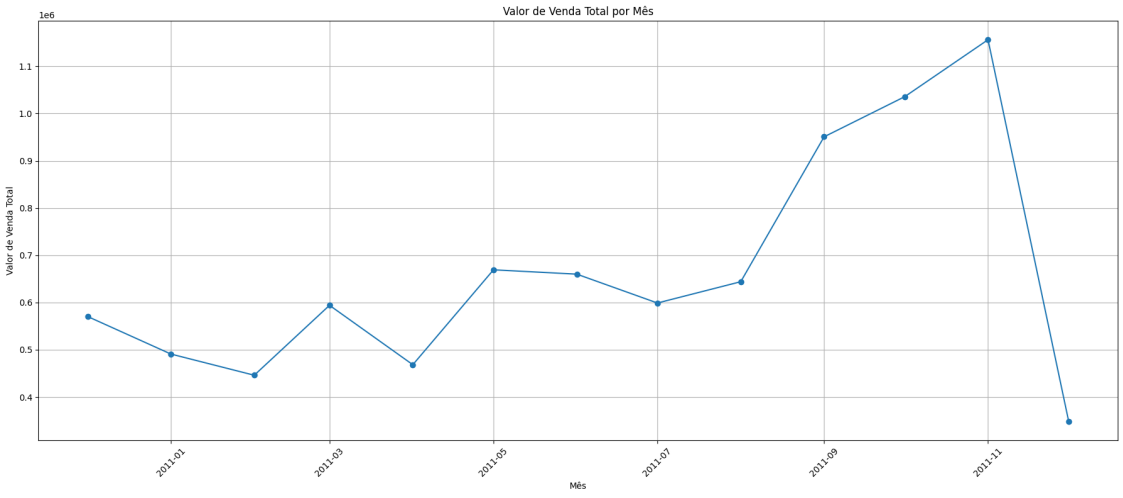
[28]:
```
Text(0, 0.5, 'Vendas')
```



Valor de venda total por mês

[29]:
```
#Extrair o ano e o mês de InvoiceDate
df['YearMonth'] = df['InvoiceDate'].dt.to_period('M')

# Valor total de vendas por mês

df.head()
sales_total_month = df.groupby('YearMonth')['Total_price'].sum().head(15);

#Convertendo o indice de periodo para datetime
sales_total_month.index = sales_total_month.index.to_timestamp()


# Criando o gráfico de linha
plt.figure(figsize=(18, 8))
plt.plot(sales_total_month.index, sales_total_month.values, marker='o')
plt.title('Valor de Venda Total por Mês')
plt.xlabel('Mês')
plt.ylabel('Valor de Venda Total')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()

# Exibindo o gráfico
plt.show()
```

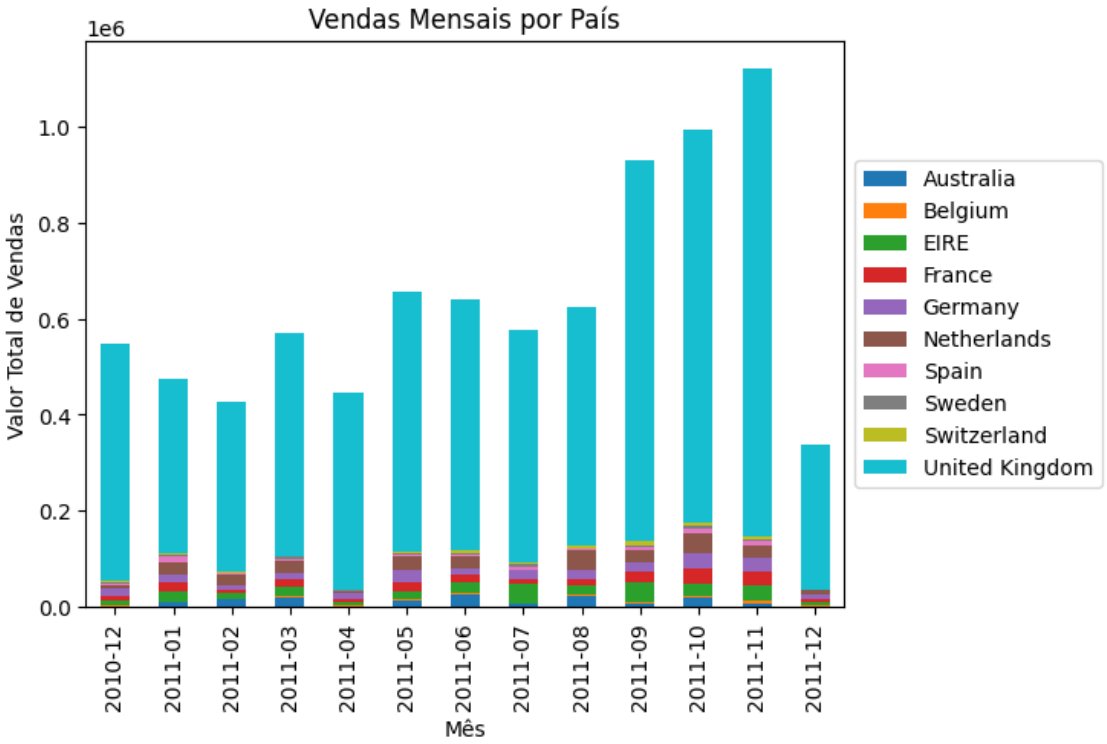
```
<ipython-input-29-9482956622ec>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['YearMonth'] = df['InvoiceDate'].dt.to_period('M')
```



Valor de venda total por mês e por país (considere apenas os top 10)

[30]:
```
#Valor de venda total por mês e por país (considere apenas os top 10)

#Extrair o ano e o mês de InvoiceDate
df['YearMonth'] = df['InvoiceDate'].dt.to_period('M')

#Agrupando
top10_sales_countries = df.groupby('Country')['Total_price'].sum().
 ↪sort_values(ascending=False).head(10);

#Filtrar o DataFrame original para incluir apenas os top 10 países
top_countries = top10_sales_countries.index
df_top_countries = df[df['Country'].isin(top_countries)]

#Agrupar por YearMonth e Country, somando os valores de Total_price
monthly_country_sales = df_top_countries.groupby(['YearMonth',␣
 ↪'Country'])['Total_price'].sum().unstack().fillna(0)

# Criar figura e eixos
fig, ax = plt.subplots()

# Plotar o gráfico de barras
monthly_country_sales.plot(kind='bar', ax=ax, stacked=True)

# Definir rótulos e título
ax.set_xlabel('Mês')
ax.set_ylabel('Valor Total de Vendas')
ax.set_title('Vendas Mensais por País')

# Adicionar legenda
plt.legend(monthly_country_sales.columns, loc='center left', bbox_to_anchor=(1,␣
 ↪0.5))

# Mostrar o gráfico
plt.show()
```

```
<ipython-input-30-37b275cd8fe8>:4: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['YearMonth'] = df['InvoiceDate'].dt.to_period('M')
```



[31]:
```
df.head()
```

[31]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Total_price	YearMonth
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30	2010-12
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	2010-12
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00	2010-12
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	2010-12
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	2010-12

Calculo do RFM

[32]:
```
# Criando uma data de referencia
data_referencia = df['InvoiceDate'].max() + dt.timedelta(days=1)

# Agrupar os dados por cliente, e aplicando os cálculos às colunas data␣
 ↪(diferença de dias para Recência), número do pedido (contagem para␣
 ↪frequência) e Preço total (soma para Valor Monetário)
rfm = df.groupby(['CustomerID']).agg({'InvoiceDate': lambda x :␣
 ↪(data_referencia - x.max()).days,
                                      'InvoiceNo':'count','Total_price':'mean'})

# Renomeando as colunas
rfm = rfm.rename(columns={'InvoiceDate':'Recência','InvoiceNo':
 ↪'Frequência','Total_price':'TicketMédio'})
rfm


# Calculando os níveis RFM
rfm['RecênciaRank'] = pd.qcut(rfm['Recência'], q=3, labels=["Baixa", "Média",␣
 ↪"Alta"])
rfm['FrequênciaRank'] = pd.qcut(rfm['Frequência'], q=3, labels=["Baixa",␣
 ↪"Média", "Alta"])
rfm['TicketMédioRank'] = pd.qcut(rfm['TicketMédio'], q=3, labels=["Baixa",␣
 ↪"Média", "Alta"])
```

[33]:
```
rfm.head(20)
```

[33]:

CustomerID	Recência	Frequência	TicketMédio	RecênciaRank	FrequênciaRank
12347.0	2	182	23.681319	Baixa	Alta
12348.0	75	31	57.975484	Média	Média
12349.0	19	73	24.076027	Baixa	Alta
12350.0	310	17	19.670588	Alta	Baixa
12352.0	36	85	29.482824	Média	Alta
12353.0	204	4	22.250000	Alta	Baixa
12354.0	232	58	18.610345	Alta	Média
12355.0	214	13	35.338462	Alta	Baixa
12356.0	23	59	47.651356	Baixa	Média
12357.0	33	131	47.386794	Média	Alta
12358.0	2	19	61.476842	Baixa	Baixa
12359.0	58	245	25.755224	Média	Alta
12360.0	52	129	20.636124	Média	Alta
12361.0	287	10	18.990000	Alta	Baixa
12362.0	3	266	19.647481	Baixa	Alta
12363.0	110	23	24.000000	Alta	Baixa
12364.0	8	85	15.448235	Baixa	Alta

```
12365.0          291           22    29.153636         Alta          Baixa
12367.0            4           11    15.354545        Baixa          Baixa
12370.0           51          166    21.336988        Média           Alta
```

```
           TicketMédioRank
CustomerID
12347.0               Alta
12348.0               Alta
12349.0               Alta
12350.0              Média
12352.0               Alta
12353.0               Alta
12354.0              Média
12355.0               Alta
12356.0               Alta
12357.0               Alta
12358.0               Alta
12359.0               Alta
12360.0              Média
12361.0              Média
12362.0              Média
12363.0               Alta
12364.0              Média
12365.0               Alta
12367.0              Média
12370.0               Alta
```

[34]:
```
rfm.tail(20)
```

[34]:
```
             Recência  Frequência  TicketMédio RecênciaRank FrequênciaRank  \
CustomerID
18259.0            25          42    55.680952        Baixa          Média
18260.0           173         133    19.762030         Alta           Alta
18261.0            43          21    15.440000        Média          Baixa
18262.0           140          13    11.498462         Alta          Baixa
18263.0            26          61    19.887869        Média          Média
18265.0            72          46    17.424130        Média          Média
18268.0           134           1    25.500000         Alta          Baixa
18269.0           366           7    24.085714         Alta          Baixa
18270.0            38          11    25.740909        Média          Baixa
18272.0             3         166    18.545663        Baixa           Alta
18273.0             2           3    68.000000        Baixa          Baixa
18274.0            30          11    15.992727        Média          Baixa
18276.0            44          14    23.990000        Média          Baixa
18277.0            58           8    13.797500        Média          Baixa
18278.0            74           9    19.322222        Média          Baixa
18280.0           278          10    18.060000         Alta          Baixa
```

18281.0	181	7	11.545714	Alta	Baixa
18282.0	8	12	14.837500	Baixa	Baixa
18283.0	4	721	2.837074	Baixa	Alta
18287.0	43	70	26.246857	Média	Média

CustomerID	TicketMédioRank
18259.0	Alta
18260.0	Média
18261.0	Média
18262.0	Baixa
18263.0	Média
18265.0	Média
18268.0	Alta
18269.0	Alta
18270.0	Alta
18272.0	Média
18273.0	Alta
18274.0	Média
18276.0	Alta
18277.0	Baixa
18278.0	Média
18280.0	Média
18281.0	Baixa
18282.0	Baixa
18283.0	Baixa
18287.0	Alta