

✓ Contexto

Uma empresa do ramo de e-commerce solicitou um levantamento sobre os indicadores de recência, frequência e ticket médio (RFM) dos seus clientes.

A saber RFM:

- R (Recency): Tempo que o cliente realizou a última compra (em dias)
- F (Frequency): Quantidade de compras realizadas pelo cliente
- M (Monetary): Valor do ticket médio gasto pelo cliente

onde ticket médio = média do total gasto por pedido para cada cliente.

Para isso, a empresa disponibilizou uma base de dados (arquivo csv). Inicialmente, foi identificado o Python como ferramenta de utilização, gerando um output seguro e satisfatório para o cliente. Para a segurança da empresa, o arquivo será nomeado contendo apenas a identificação do cliente e métricas RFM.

Sobre os dados

A tabela contém informações de compras de um e-commerce em 37 países. Contém a identificação do cliente e os dados da compra.

Coluna	Descrição
CustomerID	Código de identificação do cliente
Description	Descrição do produto
InvoiceNo	Código da fatura
StockCode	Código de estoque do produto
Quantity	Quantidade do produto
InvoiceDate	Data do faturamento (compra)
UnitPrice	Preço unitário do produto
Country	País da compra

Como começar?

1. Importar o dataset para o colab
2. Entender os dados
3. Tratar os dados nulos
4. Tratar os outliers

Dessa forma, vamos desenvolver o algoritmo para receber o arquivo csv de entrada e retornar um algoritmo de saída com as seguintes colunas:

- **CustomerID:** Código do cliente
- **R: Recência**
- **F: Frequência**
- **M: Ticket médio**

✓ 1 - Análise Descritiva

✓ 1.1 - Upload do arquivo (desafio_5.csv)

```
1 from google.colab import files
2 uploaded = files.upload()
```



✓ 1.2 - Importar bibliotecas

```
1 import pandas as pd
2
```

✓ 1.3 - Análise do DataFrame (df)

```
1 # O df = pd.read_csv não está funcionando
2
3 df = pd.read_csv("desafio_5.csv", encoding="latin1") # Codificação mais comum para caracteres especiais
4 # ou
5 df = pd.read_csv("desafio_5.csv", encoding="ISO-8859-1")
```

```
6 # ou
7 df = pd.read_csv("desafio_5.csv", encoding="cp1252")
```

```
1 df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom

```
1 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
```

```
1 # InvoiceNo - object / str
2 # StockCode - object / str
3 # Description - object / str
4 # Quantity - int
5 # InvoiceDate - datetime
6 # UnitPrice - float
7 # CustomerID - int
8 # Country - object / str
9
10
11 df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'], format='%m/%d/%Y %H:%M') # datetime64[ns]
12
13 df = df.dropna(subset=['CustomerID'])
14 df['CustomerID'] = df['CustomerID'].astype(int)
15 # int
16
17 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   InvoiceNo        406829 non-null object
1   StockCode       406829 non-null object
2   Description     406829 non-null object
3   Quantity        406829 non-null int64
4   InvoiceDate     406829 non-null datetime64[ns]
5   UnitPrice       406829 non-null float64
6   CustomerID     406829 non-null int64
7   Country         406829 non-null object
dtypes: datetime64[ns](1), float64(1), int64(2), object(4)
memory usage: 27.9+ MB
<ipython-input-69-73dc9b945281>:14: 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
df['CustomerID'] = df['CustomerID'].astype(int)
```

```
1 df.describe()
```

	Quantity	InvoiceDate	UnitPrice	CustomerID
count	406829.000000	406829	406829.000000	406829.000000
mean	12.061303	2011-07-10 16:30:57.879207424	3.460471	15287.690570
min	-80995.000000	2010-12-01 08:26:00	0.000000	12346.000000
25%	2.000000	2011-04-06 15:02:00	1.250000	13953.000000
50%	5.000000	2011-07-31 11:48:00	1.950000	15152.000000
75%	12.000000	2011-10-20 13:06:00	3.750000	16791.000000
max	80995.000000	2011-12-09 12:50:00	38970.000000	18287.000000
std	248.693370	NaN	69.315162	1713.600303

- Os valores negativos nas colunas 'Quantity' e 'UnitPrice' possuem chances consideráveis de existirem valores com erro de inserção, e/ou possíveis transações que precisam ser melhor aprofundadas, ex.: devoluções, etc. Além disso, as colunas estão com alta dispersão na maioria delas, indicando alta variabilidade dos dados.

```
1 neg_quantity = df[df['Quantity'] < 0]
2 print(neg_quantity)
```

	InvoiceNo	StockCode	Description	Quantity	\
141	C536379	D	Discount	-1	
154	C536383	35004C	SET OF 3 COLOURED FLYING DUCKS	-1	
235	C536391	22556	PLASTERS IN TIN CIRCUS PARADE	-12	
236	C536391	21984	PACK OF 12 PINK PAISLEY TISSUES	-24	
237	C536391	21983	PACK OF 12 BLUE PAISLEY TISSUES	-24	
...	
540449	C581490	23144	ZINC T-LIGHT HOLDER STARS SMALL	-11	
541541	C581499	M	Manual	-1	
541715	C581568	21258	VICTORIAN SEWING BOX LARGE	-5	
541716	C581569	84978	HANGING HEART JAR T-LIGHT HOLDER	-1	
541717	C581569	20979	36 PENCILS TUBE RED RETROSPOT	-5	
	InvoiceDate	UnitPrice	CustomerID	Country	
141	2010-12-01 09:41:00	27.50	14527	United Kingdom	
154	2010-12-01 09:49:00	4.65	15311	United Kingdom	
235	2010-12-01 10:24:00	1.65	17548	United Kingdom	
236	2010-12-01 10:24:00	0.29	17548	United Kingdom	
237	2010-12-01 10:24:00	0.29	17548	United Kingdom	
...	
540449	2011-12-09 09:57:00	0.83	14397	United Kingdom	
541541	2011-12-09 10:28:00	224.69	15498	United Kingdom	
541715	2011-12-09 11:57:00	10.95	15311	United Kingdom	
541716	2011-12-09 11:58:00	1.25	17315	United Kingdom	
541717	2011-12-09 11:58:00	1.25	17315	United Kingdom	

[8905 rows x 8 columns]

```
1 # Contagem de valores negativos
2 qt_neg_quantity = (df['Quantity'] <= 0).sum()
3 qt_neg_unitprice = (df['UnitPrice'] <= 0).sum()
4
5 print(f"Quantidade de valores negativos em 'Quantity': {qt_neg_quantity}")
6 print(f"Quantidade de valores negativos em 'UnitPrice': {qt_neg_unitprice}")
7
```

```
Quantidade de valores negativos em 'Quantity': 8905
Quantidade de valores negativos em 'UnitPrice': 40
```

```
1 # Valores nulos por coluna
2
3 print(df.isnull().sum())
```

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

- Grande quantidade de valores nulos, impossibilitando o tratamento dos dados para este caso, pois, na coleta, não foram evidenciadas tratativas de solução ou alternativas para os valores nulos em questão.

```

1 nulos_customerid = df[df['CustomerID'].isnull()]
2 print(f' Linhas nulas da coluna CustomerID:')
3 print(nulos_customerid)

```

```

↳ Linhas nulas da coluna CustomerID:
Empty DataFrame
Columns: [InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country]
Index: []

```

2 - Análise Exploratória

2.1 - Removendo dados nulos e afins

```

1 # Removendo os dados nulos das colunas: CustomerID e Description, sendo:
2
3 # Description      1454
4 # CustomerID      135080
5
6 df = df.dropna(subset=['CustomerID', 'Description'])
7 df.info()
8

```

```

↳ <class 'pandas.core.frame.DataFrame'>
Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
#   Column      Non-Null Count  Dtype
---  -
0   InvoiceNo    406829 non-null  object
1   StockCode   406829 non-null  object
2   Description  406829 non-null  object
3   Quantity    406829 non-null  int64
4   InvoiceDate  406829 non-null  datetime64[ns]
5   UnitPrice   406829 non-null  float64
6   CustomerID  406829 non-null  int64
7   Country     406829 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(2), object(4)
memory usage: 27.9+ MB

```

```

1 # Confirmação da ausência de valores nulos e na - pós tratamento:
2 print(f'Quantidade de valores null (nulos):\n{df.isnull().sum()}')
3 print('\n')
4 print(f'Quantidade de valores na (não aplicáveis):\n{df.isna().sum()}')

```

```

↳ Quantidade de valores null (nulos):
InvoiceNo      0
StockCode      0
Description     0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64

```

```

Quantidade de valores na (não aplicáveis):
InvoiceNo      0
StockCode      0
Description     0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64

```

2.2 - Filtrando valores

```

1 df_filter_quantity = df[df['Quantity'] <= 0]
2 print(df_filter_quantity)
3
4 # 8905 valores menores ou iguais a zero, da coluna 'Quantity'

```

```

↳
   InvoiceNo  StockCode  Description  Quantity  \
141    C536379        D      Discount      -1
154    C536383    35004C  SET OF 3 COLOURED  FLYING DUCKS      -1
235    C536391    22556  PLASTERS IN TIN CIRCUS PARADE      -12
236    C536391    21984  PACK OF 12 PINK PAISLEY TISSUES      -24
237    C536391    21983  PACK OF 12 BLUE PAISLEY TISSUES      -24
...      ...      ...      ...      ...
540449  C581490    23144  ZINC T-LIGHT HOLDER STARS SMALL      -11
541541  C581499        M      Manual      -1

```

541715	C581568	21258	VICTORIAN SEWING BOX LARGE	-5
541716	C581569	84978	HANGING HEART JAR T-LIGHT HOLDER	-1
541717	C581569	20979	36 PENCILS TUBE RED RETROSPOT	-5

	InvoiceDate	UnitPrice	CustomerID	Country
141	2010-12-01 09:41:00	27.50	14527	United Kingdom
154	2010-12-01 09:49:00	4.65	15311	United Kingdom
235	2010-12-01 10:24:00	1.65	17548	United Kingdom
236	2010-12-01 10:24:00	0.29	17548	United Kingdom
237	2010-12-01 10:24:00	0.29	17548	United Kingdom
...
540449	2011-12-09 09:57:00	0.83	14397	United Kingdom
541541	2011-12-09 10:28:00	224.69	15498	United Kingdom
541715	2011-12-09 11:57:00	10.95	15311	United Kingdom
541716	2011-12-09 11:58:00	1.25	17315	United Kingdom
541717	2011-12-09 11:58:00	1.25	17315	United Kingdom

[8905 rows x 8 columns]

```
1 df_filter_unitprice = df[df['UnitPrice'] <= 0]
2 print(df_filter_unitprice)
3
4 # 40 valores menores ou iguais a zero, da coluna 'UnitPrice'
```

314746	564651	23268	SET OF 2 CERAMIC CHRISTMAS REINDEER	192
314747	564651	22955	36 FOIL STAR CAKE CASES	144
314748	564651	21786	POLKADOT RAIN HAT	144
358655	568158	PADS	PADS TO MATCH ALL CUSHIONS	1
361825	568384	M	Manual	1
379913	569716	22778	GLASS CLOCHE SMALL	2
395529	571035	M	Manual	1
420404	572893	21208	PASTEL COLOUR HONEYCOMB FAN	5
436428	574138	23234	BISCUIT TIN VINTAGE CHRISTMAS	216
436597	574175	22065	CHRISTMAS PUDDING TRINKET POT	12
436961	574252	M	Manual	1
439361	574469	22385	JUMBO BAG SPACEBOY DESIGN	12
446125	574879	22625	RED KITCHEN SCALES	2
446793	574920	22899	CHILDREN'S APRON DOLLY GIRL	1
446794	574920	23480	MINI LIGHTS WOODLAND MUSHROOMS	1
454463	575579	22437	SET OF 9 BLACK SKULL BALLOONS	20
454464	575579	22089	PAPER BUNTING VINTAGE PAISLEY	24
479079	577129	22464	HANGING METAL HEART LANTERN	4
479546	577168	M	Manual	1
480649	577314	23407	SET OF 2 TRAYS HOME SWEET HOME	2
485985	577696	M	Manual	1
502122	578841	84826	ASSTD DESIGN 3D PAPER STICKERS	12540

	InvoiceDate	UnitPrice	CustomerID	Country
9302	2010-12-05 14:02:00	0.0	12647	Germany
33576	2010-12-16 14:36:00	0.0	16560	United Kingdom
40089	2010-12-21 13:45:00	0.0	14911	EIRE
47068	2011-01-06 16:41:00	0.0	13081	United Kingdom
47070	2011-01-06 16:41:00	0.0	13081	United Kingdom
56674	2011-01-13 15:10:00	0.0	15107	United Kingdom
86789	2011-02-10 13:08:00	0.0	17560	United Kingdom
130188	2011-03-23 10:25:00	0.0	13239	United Kingdom
139453	2011-03-30 12:45:00	0.0	13113	United Kingdom
145208	2011-04-04 14:42:00	0.0	14410	United Kingdom
157042	2011-04-14 18:57:00	0.0	12457	Switzerland
187613	2011-05-12 15:21:00	0.0	17667	United Kingdom
198383	2011-05-20 14:13:00	0.0	12415	Australia
279324	2011-07-26 12:24:00	0.0	16818	United Kingdom
282912	2011-07-28 17:09:00	0.0	12507	Spain
285657	2011-08-01 11:44:00	0.0	15581	United Kingdom
298054	2011-08-11 11:42:00	0.0	14911	EIRE
314745	2011-08-26 14:19:00	0.0	14646	Netherlands
314746	2011-08-26 14:19:00	0.0	14646	Netherlands
314747	2011-08-26 14:19:00	0.0	14646	Netherlands
314748	2011-08-26 14:19:00	0.0	14646	Netherlands
358655	2011-09-25 12:22:00	0.0	16133	United Kingdom
361825	2011-09-27 09:46:00	0.0	12748	United Kingdom
379913	2011-10-06 08:17:00	0.0	15804	United Kingdom
395529	2011-10-13 12:50:00	0.0	12446	RSA
420404	2011-10-26 14:36:00	0.0	18059	United Kingdom
436428	2011-11-03 11:26:00	0.0	12415	Australia
436597	2011-11-03 11:47:00	0.0	14110	United Kingdom
436961	2011-11-03 13:24:00	0.0	12437	France
439361	2011-11-04 11:55:00	0.0	12431	Australia
446125	2011-11-07 13:22:00	0.0	13014	United Kingdom
446793	2011-11-07 16:34:00	0.0	13985	United Kingdom
446794	2011-11-07 16:34:00	0.0	13985	United Kingdom
454463	2011-11-10 11:49:00	0.0	13081	United Kingdom

```
1 df_filter_quantity.info() # 8905 valores menores ou iguais a zero
2 print('\n')
3 df_filter_unitprice.info() # 40 valores menores ou iguais a zero
4
5 # 8945 valores menores ou iguais a zero (total)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8905 entries, 141 to 541717
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   InvoiceNo        8905 non-null   object
1   StockCode        8905 non-null   object
2   Description      8905 non-null   object
3   Quantity         8905 non-null   int64
4   InvoiceDate       8905 non-null   datetime64[ns]
5   UnitPrice        8905 non-null   float64
6   CustomerID       8905 non-null   int64
7   Country          8905 non-null   object
dtypes: datetime64[ns](1), float64(1), int64(2), object(4)
memory usage: 626.1+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 40 entries, 9302 to 502122
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   InvoiceNo        40 non-null     object
1   StockCode        40 non-null     object
2   Description      40 non-null     object
3   Quantity         40 non-null     int64
4   InvoiceDate       40 non-null     datetime64[ns]
5   UnitPrice        40 non-null     float64
6   CustomerID       40 non-null     int64
7   Country          40 non-null     object
dtypes: datetime64[ns](1), float64(1), int64(2), object(4)
memory usage: 2.8+ KB
```

```
1 # Filtro de ambas as columnas, com valores maiores que zero
2
3 df_filter_qt_price = df[(df['Quantity'] > 0) | (df['UnitPrice'] > 0)]
4 print(df_filter_qt_price)
5
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   InvoiceNo        406829 non-null  object
1   StockCode        406829 non-null  object
2   Description      406829 non-null  object
3   Quantity         406829 non-null  int64
4   InvoiceDate       406829 non-null  datetime64[ns]
5   UnitPrice        406829 non-null  float64
6   CustomerID       406829 non-null  int64
7   Country          406829 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(2), object(4)
memory usage: 1.1+ GB
```

	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	
...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	
...
0	2010-12-01 08:26:00	2.55	17850	United Kingdom	
1	2010-12-01 08:26:00	3.39	17850	United Kingdom	
2	2010-12-01 08:26:00	2.75	17850	United Kingdom	
3	2010-12-01 08:26:00	3.39	17850	United Kingdom	
4	2010-12-01 08:26:00	3.39	17850	United Kingdom	
...
541904	2011-12-09 12:50:00	0.85	12680	France	
541905	2011-12-09 12:50:00	2.10	12680	France	
541906	2011-12-09 12:50:00	4.15	12680	France	
541907	2011-12-09 12:50:00	4.15	12680	France	
541908	2011-12-09 12:50:00	4.95	12680	France	

[406829 rows x 8 columns]

```
1 # 5225 valores duplicados
2
3 df_duplicated_removed = df_filter_qt_price[df_filter_qt_price.duplicated()]
4 print(df_duplicated_removed)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, 517 to 541701
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   InvoiceNo        10 non-null     object
1   StockCode        10 non-null     object
2   Description      10 non-null     object
3   Quantity         10 non-null     int64
4   InvoiceDate       10 non-null     datetime64[ns]
5   UnitPrice        10 non-null     float64
6   CustomerID       10 non-null     int64
7   Country          10 non-null     object
dtypes: datetime64[ns](1), float64(1), int64(2), object(4)
memory usage: 1.1+ GB
```

	InvoiceNo	StockCode	Description	Quantity	\
517	536409	21866	UNION JACK FLAG LUGGAGE TAG	1	
527	536409	22866	HAND WARMER SCOTTY DOG DESIGN	1	
537	536409	22900	SET 2 TEA TOWELS I LOVE LONDON	1	
539	536409	22111	SCOTTIE DOG HOT WATER BOTTLE	1	
555	536412	22327	ROUND SNACK BOXES SET OF 4 SKULLS	1	
...
541675	581538	22068	BLACK PIRATE TREASURE CHEST	1	
541689	581538	23318	BOX OF 6 MINI VINTAGE CRACKERS	1	
541692	581538	22992	REVOLVER WOODEN RULER	1	
541699	581538	22694	WICKER STAR	1	
541701	581538	23343	JUMBO BAG VINTAGE CHRISTMAS	1	
...
517	2010-12-01 11:45:00	1.25	17908	United Kingdom	
527	2010-12-01 11:45:00	2.10	17908	United Kingdom	

```

537    2010-12-01 11:45:00    2.95    17908    United Kingdom
539    2010-12-01 11:45:00    4.95    17908    United Kingdom
555    2010-12-01 11:49:00    2.95    17920    United Kingdom
...
541675 2011-12-09 11:34:00    0.39    14446    United Kingdom
541689 2011-12-09 11:34:00    2.49    14446    United Kingdom
541692 2011-12-09 11:34:00    1.95    14446    United Kingdom
541699 2011-12-09 11:34:00    2.10    14446    United Kingdom
541701 2011-12-09 11:34:00    2.08    14446    United Kingdom

```

[5225 rows x 8 columns]

```

1 # DataFrame após a remoção dos valores duplicados
2
3 df_cleaned = df_filter_qt_price.drop_duplicates()
4 print(df_cleaned)

```

```

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
...
541904    581587    22613      PACK OF 20 SPACEBOY NAPKINS       12
541905    581587    22899  CHILDREN'S APRON DOLLY GIRL          6
541906    581587    23254  CHILDRENS CUTLERY DOLLY GIRL          4
541907    581587    23255  CHILDRENS CUTLERY CIRCUS PARADE         4
541908    581587    22138      BAKING SET 9 PIECE RETROSPOT        3

```

```

InvoiceDate UnitPrice CustomerID Country
0    2010-12-01 08:26:00    2.55    17850    United Kingdom
1    2010-12-01 08:26:00    3.39    17850    United Kingdom
2    2010-12-01 08:26:00    2.75    17850    United Kingdom
3    2010-12-01 08:26:00    3.39    17850    United Kingdom
4    2010-12-01 08:26:00    3.39    17850    United Kingdom
...
541904 2011-12-09 12:50:00    0.85    12680      France
541905 2011-12-09 12:50:00    2.10    12680      France
541906 2011-12-09 12:50:00    4.15    12680      France
541907 2011-12-09 12:50:00    4.15    12680      France
541908 2011-12-09 12:50:00    4.95    12680      France

```

[401604 rows x 8 columns]

```

1 # Remoção dos outliers extremos em que a quantidade do item na compra é superior a 10.000, e o preço unitário é maior que 5.000.
2
3 df_cleaned_filter = df_cleaned[(df_cleaned['Quantity'] <= 10000) | (df_cleaned['UnitPrice'] <= 5000)]
4 print(df_cleaned_filter)
5 print('\n')
6 df_cleaned_filter.info()
7

```

```

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
...
541904    581587    22613      PACK OF 20 SPACEBOY NAPKINS       12
541905    581587    22899  CHILDREN'S APRON DOLLY GIRL          6
541906    581587    23254  CHILDRENS CUTLERY DOLLY GIRL          4
541907    581587    23255  CHILDRENS CUTLERY CIRCUS PARADE         4
541908    581587    22138      BAKING SET 9 PIECE RETROSPOT        3

```

```

InvoiceDate UnitPrice CustomerID Country
0    2010-12-01 08:26:00    2.55    17850    United Kingdom
1    2010-12-01 08:26:00    3.39    17850    United Kingdom
2    2010-12-01 08:26:00    2.75    17850    United Kingdom
3    2010-12-01 08:26:00    3.39    17850    United Kingdom
4    2010-12-01 08:26:00    3.39    17850    United Kingdom
...
541904 2011-12-09 12:50:00    0.85    12680      France
541905 2011-12-09 12:50:00    2.10    12680      France
541906 2011-12-09 12:50:00    4.15    12680      France
541907 2011-12-09 12:50:00    4.15    12680      France
541908 2011-12-09 12:50:00    4.95    12680      France

```

[401604 rows x 8 columns]

```

<class 'pandas.core.frame.DataFrame'>
Index: 401604 entries, 0 to 541908
Data columns (total 8 columns):
#   Column      Non-Null Count  Dtype
---  -

```

```

0 InvoiceNo 401604 non-null object
1 StockCode 401604 non-null object
2 Description 401604 non-null object
3 Quantity 401604 non-null int64
4 InvoiceDate 401604 non-null datetime64[ns]
5 UnitPrice 401604 non-null float64
6 CustomerID 401604 non-null int64
7 Country 401604 non-null object
dtypes: datetime64[ns](1), float64(1), int64(2), object(4)
memory usage: 27.6+ MB

```

3 - Modelagem

3.1 - Preparando os dados para análise mais avançadas

```

1 # Criação da coluna 'TotalPrice'
2
3 df_cleaned_filter['TotalPrice'] = df_cleaned_filter['Quantity'] * df_cleaned_filter['UnitPrice']
4
5 df_cleaned_filter = df_cleaned_filter[df_cleaned_filter['TotalPrice'] > 0]
6
7 df_cleaned_filter.info()

```

```

↗ <class 'pandas.core.frame.DataFrame'>
Index: 392692 entries, 0 to 541908
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   InvoiceNo        392692 non-null object
1   StockCode       392692 non-null object
2   Description      392692 non-null object
3   Quantity        392692 non-null int64
4   InvoiceDate      392692 non-null datetime64[ns]
5   UnitPrice       392692 non-null float64
6   CustomerID      392692 non-null int64
7   Country         392692 non-null object
8   TotalPrice      392692 non-null float64
dtypes: datetime64[ns](1), float64(2), int64(2), object(4)
memory usage: 30.0+ MB

```

```

1 # Data da última compra
2
3 data_primeira_compra = df_cleaned_filter['InvoiceDate'].min()
4 print(data_primeira_compra)
5
6 data_ultima_compra = df_cleaned_filter['InvoiceDate'].max()
7 print(data_ultima_compra)
8
9

```

```

↗ 2010-12-01 08:26:00
   2011-12-09 12:50:00

```

3.2 - Plotando Gráficos

- Top 10 países com maior valor em vendas
- Top 10 produtos mais vendidos
- Valor de venda total por mês
- Valor de venda total por mês e por país (considere apenas os top 10)

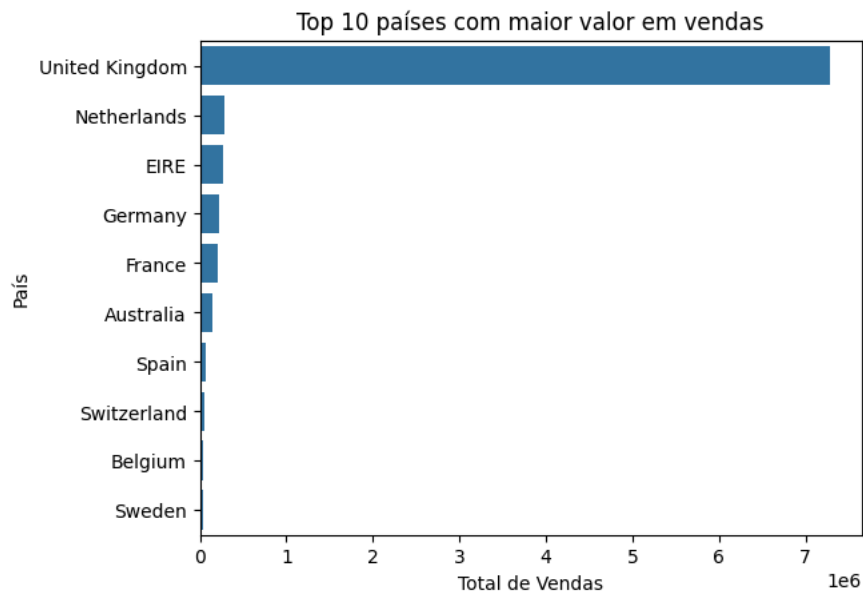
3.2.1 - Bibliotecas de Análises Gráficas

```

1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 import plotly.express as px
4 import altair as alt
5 import locale

1 top10_pais_vendas = df_cleaned_filter.groupby('Country')['TotalPrice'].sum().nlargest(10)
2
3 sns.barplot(x=top10_pais_vendas.values, y=top10_pais_vendas.index)
4 plt.xlabel('Total de Vendas')
5 plt.ylabel('País')
6 plt.title('Top 10 países com maior valor em vendas')
7 plt.show()

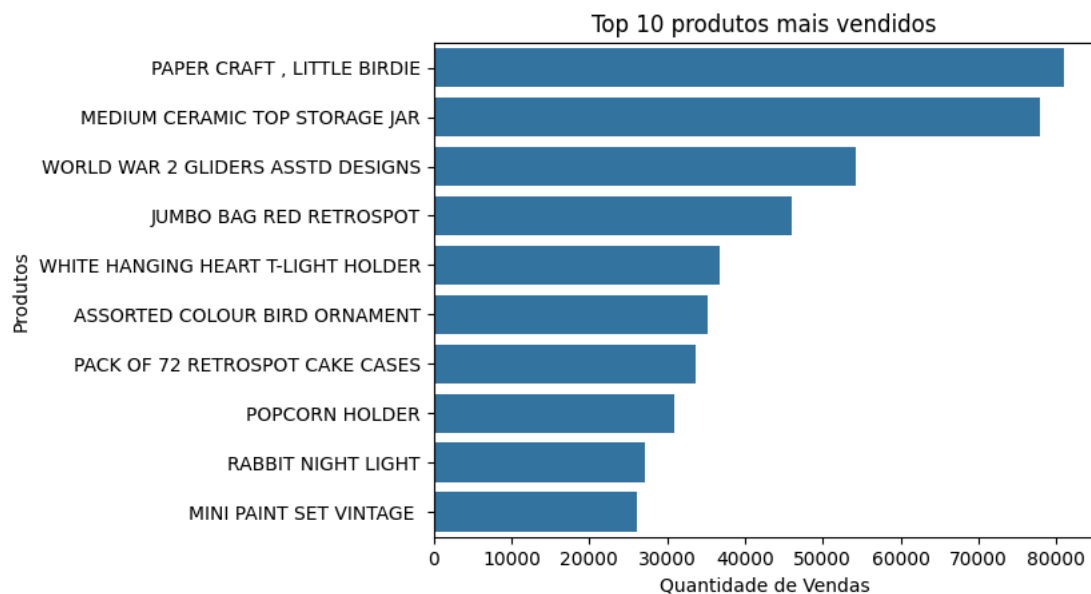
```

```

1 top10_produtos_vendas = df_cleaned_filter.groupby('Description')['Quantity'].sum().nlargest(10)
2
3 sns.barplot(x=top10_produtos_vendas.values, y=top10_produtos_vendas.index)
4 plt.xlabel('Quantidade de Vendas')
5 plt.ylabel('Produtos')
6 plt.title('Top 10 produtos mais vendidos')
7 plt.show()

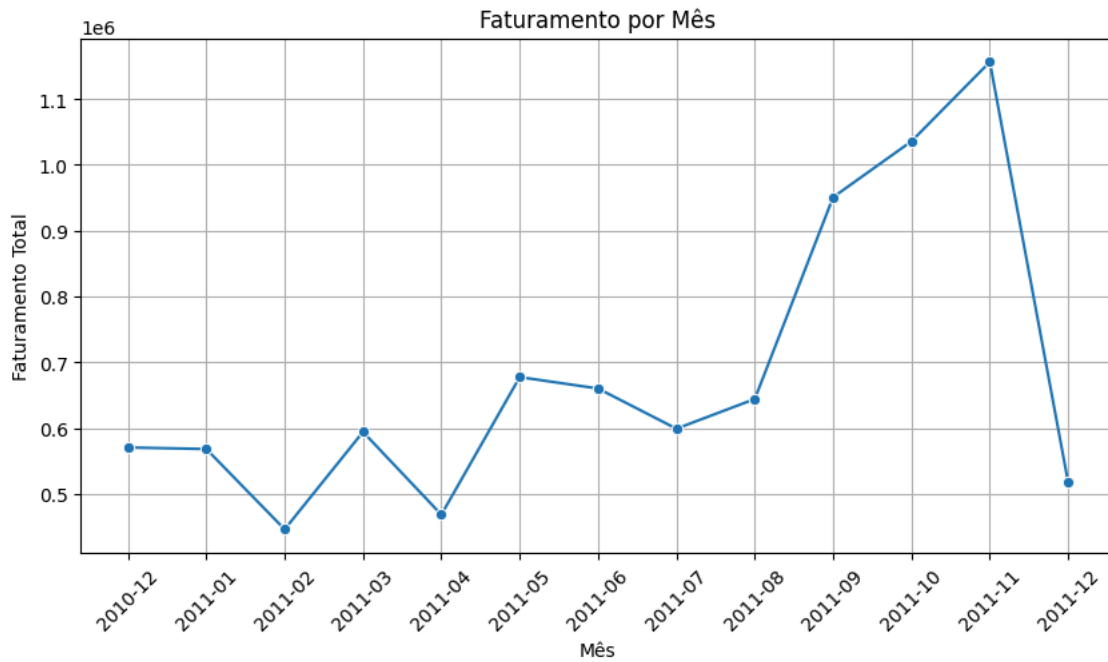
```



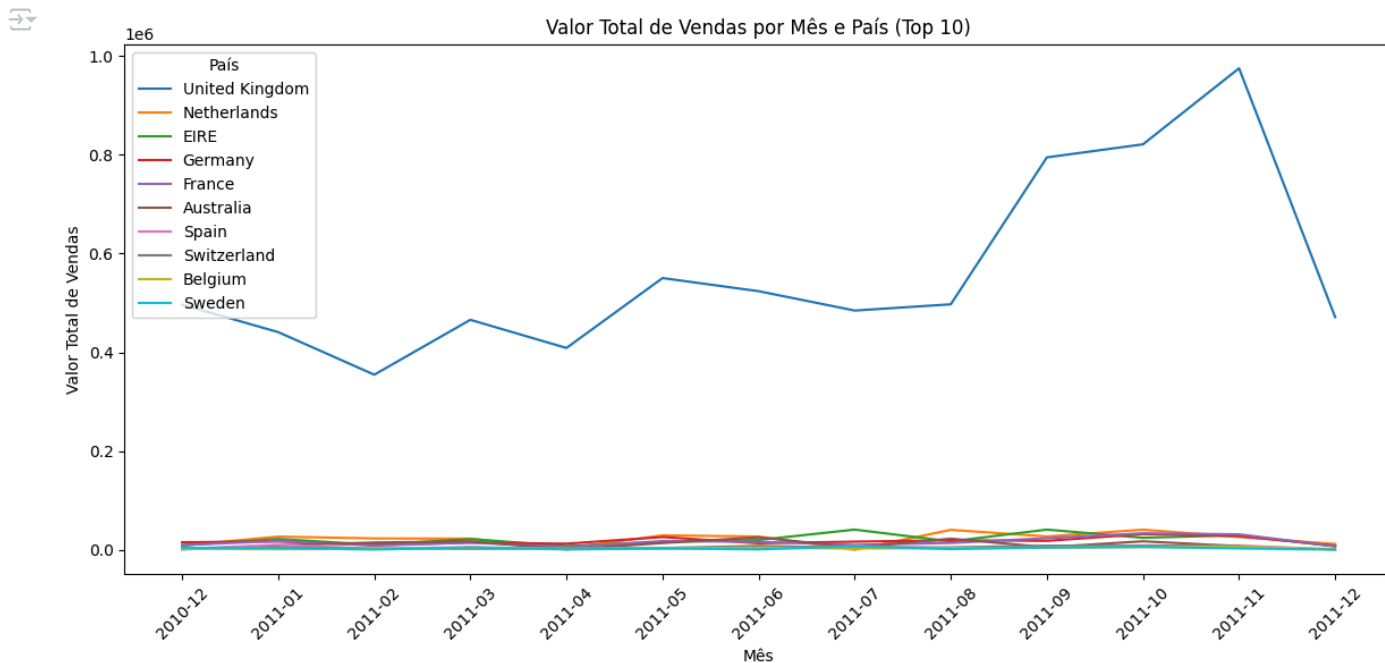
```

1 df_cleaned_filter['Month'] = df_cleaned_filter['InvoiceDate'].dt.strftime('%Y-%m')
2
3 # Faturamento/mês
4 sales_per_month = df_cleaned_filter.groupby('Month')['TotalPrice'].sum().reset_index()
5
6 sales_per_month = sales_per_month.sort_values('Month')
7
8 # Plotar gráfico
9 plt.figure(figsize=(10, 5))
10 sns.lineplot(x='Month', y='TotalPrice', data=sales_per_month, marker='o', linestyle='--')
11
12
13 plt.xticks(rotation=45) # Rotacionar os meses para melhor leitura
14 plt.xlabel('Mês')
15 plt.ylabel('Faturamento Total')
16 plt.title('Faturamento por Mês')
17 plt.grid(True)
18
19 # 7. Exibir gráfico
20 plt.show()
21

```



```
1 df_cleaned_filter['InvoiceDate'] = pd.to_datetime(df_cleaned_filter['InvoiceDate'])
2
3 df_cleaned_filter['Month'] = df_cleaned_filter['InvoiceDate'].dt.to_period('M')
4
5 total_sales_by_month_country = df_cleaned_filter.groupby(['Month', 'Country'])['TotalPrice'].sum().reset_index()
6
7 top_10_countries = total_sales_by_month_country.groupby('Country')['TotalPrice'].sum().nlargest(10).index
8
9 # Filtrar para considerar apenas os top 10 países
10 top_10_sales = total_sales_by_month_country[total_sales_by_month_country['Country'].isin(top_10_countries)]
11
12 # Criar gráfico
13 plt.figure(figsize=(12, 6))
14 for country in top_10_countries:
15     country_data = top_10_sales[top_10_sales['Country'] == country]
16     plt.plot(country_data['Month'].astype(str), country_data['TotalPrice'], label=country)
17
18 plt.xticks(rotation=45)
19 plt.xlabel('Mês')
20 plt.ylabel('Valor Total de Vendas')
21 plt.title('Valor Total de Vendas por Mês e País (Top 10)')
22 plt.legend(title='País')
23 plt.tight_layout()
24 plt.show()
```



4 - Calculo da Predição

Recomendação: Dados por cliente e pedido/compra (InvoiceNo) e obtenha a data e o preço total do pedido. Ou seja, importante agrupar novamente apenas por cliente e calcular o RFM, onde:

- R é a recência, diferença em dias da última compra do cliente e da última compra disponível no conjunto de dados, que calcularam previamente.
- F é a frequência, ou seja, a quantidade de compras feitas pelo cliente;
- M é o ticket médio, ou seja, a média das compras feitas pelo cliente.

```
1 rfm = df_cleaned_filter.groupby('CustomerID').agg(
2     Recency=('InvoiceDate', lambda x: (data_ultima_compra - x.max()).days),
3     Frequency=('InvoiceNo', 'count'),
4     Monetary=('TotalPrice', 'mean')
5 ).reset_index()
6
7 rfm.describe()
```

	CustomerID	Recency	Frequency	Monetary
count	4338.000000	4338.000000	4338.000000	4338.000000
mean	15300.408022	91.536422	90.523744	68.381590
std	1721.808492	100.014169	225.506968	1467.918294
min	12346.000000	0.000000	1.000000	2.136970
25%	13813.250000	17.000000	17.000000	12.393196
50%	15299.500000	50.000000	41.000000	17.755000
75%	16778.750000	141.000000	98.000000	24.878950
max	18287.000000	373.000000	7676.000000	77183.600000

4.1 - Explicação estatística dos dados obtidos

count → Número de clientes no conjunto de dados (4.338 clientes únicos).

mean → Média dos valores de cada métrica.

Recência média: 91 dias (tempo médio desde a última compra).

Frequência média: 90 compras por cliente.

Ticket médio: R\$ 68,38 por cliente.

std: A grande variação no Monetary (1467.92) indica que há clientes com valores de compra muito diferentes.

min - rfm mínimo temos:

- Recência = 0 dias (alguns clientes compraram na última data do dataset).
- Frequência = 1 (clientes que compraram apenas uma vez).
- Monetary mínimo = R\$ 2,14.

Quartis:

- 25% (Q1) → 25% dos clientes fizeram no máximo 17 compras e gastaram até R\$ 12,39.
- 50% (Mediana) → Metade dos clientes compraram até 41 vezes e gastaram até R\$ 17,76.
- 75% (Q3) → 75% dos clientes fizeram até 98 compras e gastaram até R\$ 24,88.

max - rfm máximo temos:

- Cliente mais ativo fez 7.676 compras! 🏆
- Maior valor gasto: R\$ 77.183,60! 💰

Outros valores:

- A Maior venda foi dos Estados Unidos, em novembro de 2011. Assim como, ganhou o título de maior faturamento histórico.
- Produto mais vendido: Paper Craft - Little Birdie

4.2 - Planos de ação (Conclusão)

Distribuição Desbalanceada

Alguns clientes compram muito, enquanto outros compram pouco ou raramente.

Soluções:

- Criar estratégias personalizadas para cada segmento.
- Segmentação de clientes com base no RFM para personalizar ofertas e comunicação.
- Criar programas de fidelidade para incentivar compras mais frequentes.