Realizando a instalação das bibliotecas

```
!pip install sidetable
!pip install pandas-profiling==3.3.0
!pip install pandas
!pip install datasets
!pip install --upgrade matplotlib
```

Carregando dados

```
import pandas as pd
tabela = pd.read_csv('/content/data.csv', encoding='latin1')
display(tabela)
```

Analisando dados nulos

```
# Verificar valores nulos
print(tabela.isnull().sum())
```

```
#Removendos dados nulos da coluna 'Description'
tabela = pd.read_csv('/content/data.csv', encoding='latin1')
tabela = tabela.drop(columns=['Description'])

print("Dados após a exclusão das colunas:")
display(tabela)
```

Removendo valores menores ou igual a zeo

```
# Excluir linhas com valores de 'UnitPrice' menores ou iguais a zero
tabela = tabela[tabela['UnitPrice'] > 0]

# Exibir a tabela atualizada
display(tabela)
```

Analisando valoes duplicados

```
#identificar valores duplicados
tabela.duplicated().sum()
```

```
#identificando valores duplicado para cada Linha
(tabela
 .groupby(tabela.columns.tolist(), dropna=False)
 .size()
 .to_frame("n_duplicates")
 .query("n_duplicates>1")
 .sort_values("n_duplicates", ascending=False)
 .head(10)
)
```

```
#removendo dados duplicados
registros_antes = tabela.shape[0]

tabela = tabela_sem_duplicatas = tabela.drop_duplicates()

registros_depois = tabela_sem_duplicatas.shape[0]

registros_removidos = registros_antes - registros_depois

print("Número removidos:", registros_removidos)
```

```
#verificando valores duplicado para cada Linha
(tabela
 .groupby(tabela.columns.tolist(), dropna=False)
 .size()
 .to_frame("n_duplicates")
 .query("n_duplicates>1")
 .sort_values("n_duplicates", ascending=False)
 .head(10)
)


#confirmando valores duplicados
tabela_sem_duplicatas = tabela
tabela.duplicated().sum()
```

Verificando valores nulos restantes

```
# Verificar valores nulos
print(tabela.isnull().sum())


# Excluir as linhas com valores nulos na coluna 'CustomerID'
tabela = tabela.dropna(subset=['CustomerID'])

# Exibir o dataframe após as alterações
print(tabela.isnull().sum())
```

Analisando os tipos de dados

```
# Verificando os tipos de dados
print(tabela.dtypes)


# Convertendo os Tipos dados das colunas 'InvoiceDate' e 'CustomerID'

tabela['InvoiceDate'] = pd.to_datetime(tabela['InvoiceDate'])
tabela['CustomerID'] = tabela['CustomerID'].astype('Int64')


# Verificando os tipos de dados após a conversão
print(tabela.dtypes)
```

Analsando os Outliers

```
# verificando outliers
import matplotlib.pyplot as plt
tabela.plot.box()
plt.xticks(rotation=60, ha='right')
plt.show()


import numpy as np
#Função
def is_outliers(row, qty_limit=10000, price_limit=5.0):

    # Checando colunas quantity e Uniprice
    qty_outlier = row['Quantity'] > qty_limit

    price_outlier = row['UnitPrice'] > price_limit

    return qty_outlier or price_outlier
```

```
# Aplicando a função
outliers_mask = tabela.apply(is_outliers, axis=1)

# Filtro baseado na função
tabela_nova = tabela[~outliers_mask]
outliers = tabela[outliers_mask]

# verificando outliers
tabela_nova.plot.box()
plt.xticks(rotation=60, ha='right')
plt.show()
```

Criando uma nova coluna Faturamento

```
# Criar a nova coluna 'Faturamento'
tabela_nova['invoicing'] = tabela_nova['Quantity'] * tabela['UnitPrice']
display(tabela_nova.head())
```

Função Max

```
# Função Max para utima dataVamos supor que 'Date' seja a coluna que contém as datas das compras
data_ultima_compra = tabela_nova['InvoiceDate'].max()

print("Data da última compra no dataset:", data_ultima_compra)
```

Top 10 países com maior valor em vendas

```
# Calculando o valor total de vendas por país
vendas_country = tabela_nova.groupby('Country')['invoicing'].sum().reset_index()

# Ordenando os países pelo valor total de vendas em ordem decrescente
top_countries = vendas_country.sort_values(by='invoicing', ascending=False)

# Obtendo os Top 10 países com maior valor em vendas
top_10 = top_countries.head(10)

# Plotando o gráfico de barras
plt.figure(figsize=(8, 5))
plt.bar(top_10['Country'], top_10['invoicing'], color='skyblue')

plt.xlabel('País')
plt.ylabel('Valor de Vendas')
plt.title('Top 10 Países com Maior Valor em Vendas')

plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Top 10 produtos mais vendidos

```
# Calculando a quantidade total vendida por produto
quantity_by_product = tabela_nova.groupby('StockCode')['Quantity'].sum().reset_index()

# Ordenando os produtos pela quantidade total vendida em ordem decrescente
top_products = quantity_by_product.sort_values(by='Quantity', ascending=False)

# Obtendo os Top 10 produtos mais vendidos
top_10_products = top_products.head(10)

# Plotando o gráfico de barras
plt.figure(figsize=(8, 5))
plt.bar(top_10_products['StockCode'], top_10_products['Quantity'], color='lightgreen')
plt.xlabel('Código do Produto')
plt.ylabel('Quantidade Vendida')
plt.title('Top 10 Produtos Mais Vendidos')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Valor de venda total por mês

```
import matplotlib.pyplot as plt

# Extrair o mês da coluna 'InvoiceDate' e criar uma nova coluna 'Month'
tabela_nova['Month'] = tabela_nova['InvoiceDate'].dt.to_period('M')

# Calculando o valor total de vendas por mês
vendas_mes = tabela_nova.groupby('Month')['invoicing'].sum()

# Plotando o gráfico de linha
plt.figure(figsize=(8, 5))
vendas_mes.plot(marker='o', color='orange', linestyle='-')

plt.xlabel('Mês')
plt.ylabel('Valor de Vendas')
plt.title('Valor de Venda Total por Mês')

plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.grid(True)
plt.show()
```

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

+ Código + Texto

```
import matplotlib.pyplot as plt

# Calculando o valor total de vendas por país
sales_by_country = tabela_nova.groupby('Country')['invoicing'].sum().reset_index()

# Ordenando os países pelo valor total de vendas em ordem decrescente
top_countries = sales_by_country.sort_values(by='invoicing', ascending=False)

# Obtendo os Top 10 países com maior valor em vendas
top_10_countries = top_countries.head(10)

# Filtrando as linhas dos Top 10 países
df_top_10_countries = tabela_nova[tabela_nova['Country'].isin(top_10_countries['Country'])]

# Calculando o valor total de vendas por mês e por país
sales_by_month_country = df_top_10_countries.groupby(['Month', 'Country'])['invoicing'].sum().unstack()

# Plotando o gráfico de linha
plt.figure(figsize=(12, 8))
sales_by_month_country.plot(marker='o', linestyle='-')
plt.xlabel('Mês')
plt.ylabel('Valor de Vendas')
plt.title('Valor Total de Venda de País por Mês')
plt.xticks(rotation=45, ha='right')
plt.legend(title='País top 10', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.grid(True)
plt.show()
```

Cálculo do RFM

```
# Agrupar os dados por cliente e pedido para obter a data e o preço total do pedido
grouped_data = tabela_nova.groupby(['CustomerID', 'InvoiceNo']).agg({'InvoiceDate': 'max', 'UnitPrice': 'sum'})
print(grouped_data)
```

```
# Calcular a recência para cada cliente
recencia = (data_ultima_compra - grouped_data.groupby('CustomerID')['InvoiceDate'].max()).dt.days
print(recencia)
```

```
# Calcular a frequência para cada cliente
frequencia = grouped_data.groupby('CustomerID').size()
print(frequencia)
```

```
# Calcular o ticket médio para cada cliente
ticket_medio = grouped_data.groupby('CustomerID')['UnitPrice'].mean()
print(ticket_medio)
```

```
# Criar um DataFrame com os resultados do RFM
rfm_data = pd.DataFrame({'Recência': recencia,
                         'Frequência': frequencia,
                         'Ticket Médio': ticket_medio})

# Exibir o DataFrame com os resultados do RFM
print(rfm_data)
```