Consolidando dados e calculando Outliers

O seu segundo desafio será analisar os dados de vendas desde 2019 e responder as perguntas abaixo utilizando a linguagem de programação Python para preparar e analisar os dados:

2) Analise os dados na perspectiva da coluna quantidade. Existem outliers nos dados disponibilizados? É possível identificar algo em relação às vendas associadas a estes outliers? Justifique sua resposta. Calcule uma estimativa de variabilidade que ignore o efeito desses outliers.

```
import os
import csv

# Caminho para a pasta que contém os arquivos CSV

#Aqui é preciso importar os dados dentro do Collab caso queriam testar
pasta_csv = 'C:/Users/luish/OneDrive/Área de Trabalho/FIAP Projetos/FASE 5/ASSETS_PBL_FIAP_ON_1TSC_FASE_5'

# Lista para armazenar as linhas de todos os arquivos CSV
linhas_combinadas = []

# Iterar sobre os arquivos na pasta
for arquivo in os.listdir(pasta_csv):
    if arquivo.endswith('.csv'):
        caminho_arquivo = os.path.join(pasta_csv, arquivo)
        with open(caminho_arquivo, 'r', newline='', encoding='latin1') as arquivo_csv:
            leitor_csv = csv.reader(arquivo_csv)
            linhas = list(leitor_csv)
            if linhas_combinadas:
                linhas_combinadas.extend(linhas[1:])
            else:
                linhas_combinadas.extend(linhas)

# Salvar as linhas combinadas em um novo arquivo CSV
caminho_saida = 'resultado_final.csv'
with open(caminho_saida, 'w', newline='', encoding='latin1') as arquivo_saida:
    escritor_csv = csv.writer(arquivo_saida)
    escritor_csv.writerows(linhas_combinadas)

print("Arquivo CSV final gerado com sucesso.")
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
import plotly.express as px

warnings.filterwarnings('ignore')
arquivo = "resultado_final.csv"
```

```
try:
    df = pd.read_csv(arquivo, encoding='latin-1', error_bad_lines=False, delimiter=';')
except pd.errors.ParserError as e:
    print(f"Erro na leitura do arquivo CSV: {e}")
```

```
if 'df' in locals():
    df.info()
```

```
Skipping line 15017: expected 14 fields, saw 20
Skipping line 22522: expected 14 fields, saw 17
Skipping line 29992: expected 14 fields, saw 16
Skipping line 44905: expected 14 fields, saw 17
Skipping line 52384: expected 14 fields, saw 17

Skipping line 67272: expected 14 fields, saw 21

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74456 entries, 0 to 74455
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
```

```
 0   cod_pedido           74456 non-null  int64
 1   regiao_pais          74456 non-null  object
 2   produto              74456 non-null  object
 3   valor                74455 non-null  object
 4   quantidade           74290 non-null  object
 5   valor_total_bruto    74455 non-null  object
 6   data                 74455 non-null  object
 7   estado               74455 non-null  object
 8   formapagto           74455 non-null  object
 9   centro_distribuicao  74455 non-null  object
 10  responsavelpedido    74455 non-null  object
 11  valor_comissao       74455 non-null  object
 12  lucro_liquido        74454 non-null  object
 13  categoriaprod        74454 non-null  object
dtypes: int64(1), object(13)
memory usage: 8.0+ MB
```

```
df.head()
```

	cod_pedido	regiao_pais	produto	valor	quantidade	valor_total_bruto	d
0	1	Norte	Biscoito True Champion 300g	22	2	44	13/06/2
1	2	Norte	Biscoito True Champion 300g	21	2	42	3/1/2
2	3	Norte	Biscoito True Champion 300g	22	NaN	44	18/01/2
3	4	Norte	Biscoito True Champion 300g	19	4	88	19/08/2
4	5	Norte	Biscoito True Champion 300g	22	2	44	18/10/2

```
#GERANDO UM CSV COM OS OUTLIERS

import pandas as pd
import numpy as np

df['quantidade'] = pd.to_numeric(df['quantidade'], errors='coerce', downcast='integer')

# Passo 1: Calcule o IQR (Intervalo Interquartil)
Q1 = df['quantidade'].quantile(0.25)
Q3 = df['quantidade'].quantile(0.75)
IQR = Q3 - Q1

# Passo 2: Defina limites para identificar outliers
limite_inferior = Q1 - 1.5 * IQR
limite_superior = Q3 + 1.5 * IQR

# Passo 3: Identifique os outliers
outliers = df[(df['quantidade'] < limite_inferior) | (df['quantidade'] > limite_superior)]

outliers.to_csv('Outliers.csv', index=False)



#GERANDO UM CSV SEM OS OUTLIERS

import pandas as pd
import numpy as np

df['quantidade'] = pd.to_numeric(df['quantidade'], errors='coerce', downcast='integer')

# Passo 1: Calcule o IQR (Intervalo Interquartil)
Q1 = df['quantidade'].quantile(0.25)
Q3 = df['quantidade'].quantile(0.75)
```

```
IQR = Q3 - Q1

# Passo 2: Defina limites para identificar outliers
limite_inferior = Q1 - 1.5 * IQR
limite_superior = Q3 + 1.5 * IQR

# Passo 3: Identifique os outliers
outliers = df[(df['quantidade'] < limite_inferior) | (df['quantidade'] > limite_superior)]

# Crie um novo DataFrame excluindo os outliers
df_sem_outliers = df[~df.index.isin(outliers.index)]

# Agora, df_sem_outliers contém a planilha original excluindo os outliers
df_sem_outliers.to_csv('Sem_Outliers.csv', index=False)
```

Calculando medidas de variabilidade descontando os Outliers

```
#CALCULANDO MEDIDAS DE VARIABILIDADE SEM OUTLIERS

import pandas as pd
import numpy as np

# Carregue o DataFrame original
arquivo = "Sem_Outliers.csv"  # Substitua pelo nome do seu arquivo original

# Agora, calcule estimativas de variabilidade sem considerar outliers
media_sem_outliers = df_sem_outliers['quantidade'].mean()
mediana_sem_outliers = df_sem_outliers['quantidade'].median()
desvio_padrao_sem_outliers = df_sem_outliers['quantidade'].std()
variancia_sem_outliers = df_sem_outliers['quantidade'].var()
iqr_sem_outliers = df_sem_outliers['quantidade'].quantile(0.75) - df_sem_outliers['quantidade'].quantile(0.25)

# Imprima as estimativas de variabilidade
print("Média de quantidade sem outliers:", media_sem_outliers)
print("Mediana sem outliers:", mediana_sem_outliers)
print("Desvio Padrão sem outliers:", desvio_padrao_sem_outliers)
print("Variância sem outliers:", variancia_sem_outliers)
print("IQR (Intervalo Interquartil) sem outliers:", iqr_sem_outliers)
```

```
Média de quantidade sem outliers: 1.5755300353356891
Mediana sem outliers: 1.0
Desvio Padrão sem outliers: 0.7747788957866744
Variância sem outliers: 0.6002823373564183
IQR (Intervalo Interquartil) sem outliers: 1.0
```

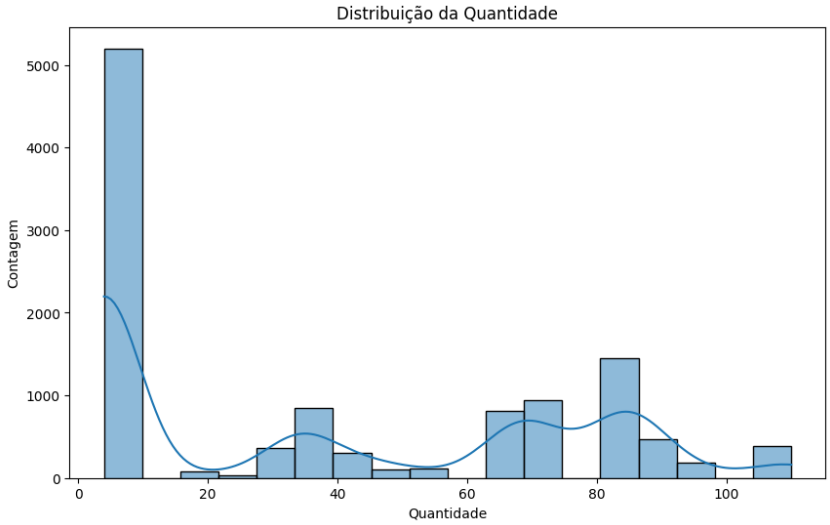
Análise dos Outliers

```
warnings.filterwarnings('ignore')
arquivo = "Outliers.csv"


try:
    df = pd.read_csv(arquivo, encoding='latin-1', error_bad_lines=False, delimiter=',')
except pd.errors.ParserError as e:
    print(f"Erro na leitura do arquivo CSV: {e}")

if 'df' in locals():
    import matplotlib.pyplot as plt
    import seaborn as sns

    # Histograma da coluna 'quantidade'
    plt.figure(figsize=(10, 6))
    sns.histplot(df['quantidade'], kde=True)
    plt.title('Distribuição da Quantidade')
    plt.xlabel('Quantidade')
    plt.ylabel('Contagem')
    plt.show()
```




```
warnings.filterwarnings('ignore')
arquivo = "Outliers.csv"


try:
    df = pd.read_csv(arquivo, encoding='latin-1', error_bad_lines=False, delimiter=',')
except pd.errors.ParserError as e:
    print(f"Erro na leitura do arquivo CSV: {e}")

if 'df' in locals():
  # Exemplo de segmentação por região
  segmented_data = df.groupby('regiao_pais')['quantidade'].mean()
  pd.DataFrame(segmented_data)

segmented_data
```

```
regiao_pais
Centro Oeste    34.918537
Nordeste        43.491833
Norte           42.147281
Sudeste         32.901627
Sul             37.647973
Name: quantidade, dtype: float64
```

```
warnings.filterwarnings('ignore')
arquivo = "Outliers.csv"


try:
    df = pd.read_csv(arquivo, encoding='latin-1', error_bad_lines=False, delimiter=',')
except pd.errors.ParserError as e:
    print(f"Erro na leitura do arquivo CSV: {e}")

if 'df' in locals():
  # Resumo estatístico das colunas numéricas
  summary = df[['valor', 'valor_total_bruto', 'valor_comissao', 'lucro_liquido']].describe()
  pd.DataFrame(summary)

summary
```

```
             valor  valor_total_bruto  valor_comissao  lucro_liquido
```

```
warnings.filterwarnings('ignore')
arquivo = "Outliers.csv"

try:
    df = pd.read_csv(arquivo, encoding='latin-1', error_bad_lines=False, delimiter=',')
except pd.errors.ParserError as e:
    print(f"Erro na leitura do arquivo CSV: {e}")

if 'df' in locals():
  # Exemplo de segmentação por região
  segmented_data = df.groupby('formapagto')['quantidade'].mean()
  segmented_data = segmented_data.sort_values(ascending=True)
  pd.DataFrame(segmented_data)

segmented_data
```

```
formapagto
Dinheiro            36.949624
Boleto BancÃ¡rio    38.297760
Pix                 38.570659
Cartao DÃ©bito      38.811877
Cartao CrÃ©dito     39.086090
Name: quantidade, dtype: float64
```

Análise

Chegamos a conclusão que existem algumas relações entre as vendas consideradas como Outliers. Primeiramente verificamos através da biblioteca seaborn e matplotlib, que em sua grande maioria das vendas classificadas como outliers, estão na parte inferior do boxplot, ou seja, são bem inferiores às 20 quantidades, conforme mostra o gráfico acima.

Além disso, verificamos que a região que possui mais vendas classificadas como Outliers, é a região Nordeste e a forma de pagamento mais comum para esses outliers, é a de Cartões de Crédito, seguido pelos de Débito.

3) Em relação à média de preço, há diferença estatisticamente significativa entre a média de preço de alguma região e a média da população? E em relação à média de preço de alguma modalidade de pagamento e à média da população? Jusitifique a hipótese

```
import pandas as pd
from scipy import stats

# Carregando os dados
data = pd.read_csv("Sem_Outliers.csv")



# Convertendo a coluna 'valor' para números (ou substituindo não numéricos por NaN)
data['valor'] = pd.to_numeric(data['valor'], errors='coerce')

# Calculando a média da coluna 'valor' após a correção
media_populacao = data['valor'].mean()

print(f"A média da população é: {media_populacao}")
```

```
A média da população é: 88.9893939883124
```

```
regioes = data['regiao_pais'].unique()
resultados = []

for regiao in regioes:
    grupo = data[data['regiao_pais'] == regiao]
    media_regiao = grupo['valor'].mean()
    resultados.append({'Região': regiao, 'Média de Preço': media_regiao})

resultados
```

```
[{'Região': 'Norte', 'Média de Preço': 89.20597854151127},
 {'Região': 'Centro Oeste', 'Média de Preço': 89.12121212121212},
 {'Região': 'Nordeste', 'Média de Preço': 88.30821371610845},
 {'Região': 'Sudeste', 'Média de Preço': 89.3678292046936},
 {'Região': 'Sul', 'Média de Preço': 89.06233098972417}]
```

```
for resultado in resultados:
    regiao = resultado['Região']
    media_regiao = resultado['Média de Preço']

    # Realize o teste t-student
    t_stat, p_valor = stats.ttest_1samp(data[data['regiao_pais'] == regiao]['valor'], media_populacao)

    if p_valor < 0.05:
        print(f"Há uma diferença estatisticamente significativa na média de preço da região {regiao} em relação à média da população.")
    else:
        print(f"A média de preço da região {regiao} não é estatisticamente diferente da média da população.")
```

```
 A média de preço da região Norte não é estatisticamente diferente da média da população.
 A média de preço da região Centro Oeste não é estatisticamente diferente da média da população.
 A média de preço da região Nordeste não é estatisticamente diferente da média da população.
 A média de preço da região Sudeste não é estatisticamente diferente da média da população.
 A média de preço da região Sul não é estatisticamente diferente da média da população.
```

Desse modo conseguimos conclui que não existe uma difereça estatística entre a média de preços das regiões.

Teste de Hipótese para Média de Preço por Modalidade de Pagamento:

O processo é semelhante ao anterior. Primeiro, dividimos os dados em grupos com base na modalidade de pagamento e calculamos a média de preço para cada grupo:

```
modalidades_pagamento = data['formapagto'].unique()
resultados_modalidade = []

for modalidade in modalidades_pagamento:
    if pd.notna(modalidade) and modalidade != 'Neide':
        grupo = data[data['formapagto'] == modalidade]
        media_modalidade = grupo['valor'].mean()
        resultados_modalidade.append({'Modalidade de Pagamento': modalidade, 'Média de Preço': media_modalidade})

resultados_modalidade
```

```
 [{'Modalidade de Pagamento': 'Dinheiro', 'Média de Preço': 90.47111184373198},
  {'Modalidade de Pagamento': 'Pix', 'Média de Preço': 88.15321078629849},
  {'Modalidade de Pagamento': 'Boleto Bancário',
   'Média de Preço': 89.10365173018657},
  {'Modalidade de Pagamento': 'Cartao Crédito',
   'Média de Preço': 86.95458883410211},
  {'Modalidade de Pagamento': 'Cartao Débito',
   'Média de Preço': 90.32572076449627}]
```

```
for resultado_modalidade in resultados_modalidade:
    modalidade = resultado_modalidade['Modalidade de Pagamento']
    media_modalidade = resultado_modalidade['Média de Preço']

    # Realize o teste t-student
    t_stat, p_valor = stats.ttest_1samp(data[data['formapagto'] == modalidade]['valor'], media_populacao)

    if p_valor < 0.05:
        print(f"Há uma diferença estatisticamente significativa na média de preço da modalidade de pagamento {modalidade} em relação à média
    else:
        print(f"A média de preço da modalidade de pagamento {modalidade} não é estatisticamente diferente da média da população.")
```

```
 A média de preço da modalidade de pagamento Dinheiro não é estatisticamente diferente da média da população.
 A média de preço da modalidade de pagamento Pix não é estatisticamente diferente da média da população.
 A média de preço da modalidade de pagamento Boleto Bancário não é estatisticamente diferente da média da população.
 A média de preço da modalidade de pagamento Cartao Crédito não é estatisticamente diferente da média da população.
 A média de preço da modalidade de pagamento Cartao Débito não é estatisticamente diferente da média da população.
```

Com base nesse teste de hipótese, concluimos que não existe uma diferença estatisticamente siginificativa nas formas de pagamento e a média de preço

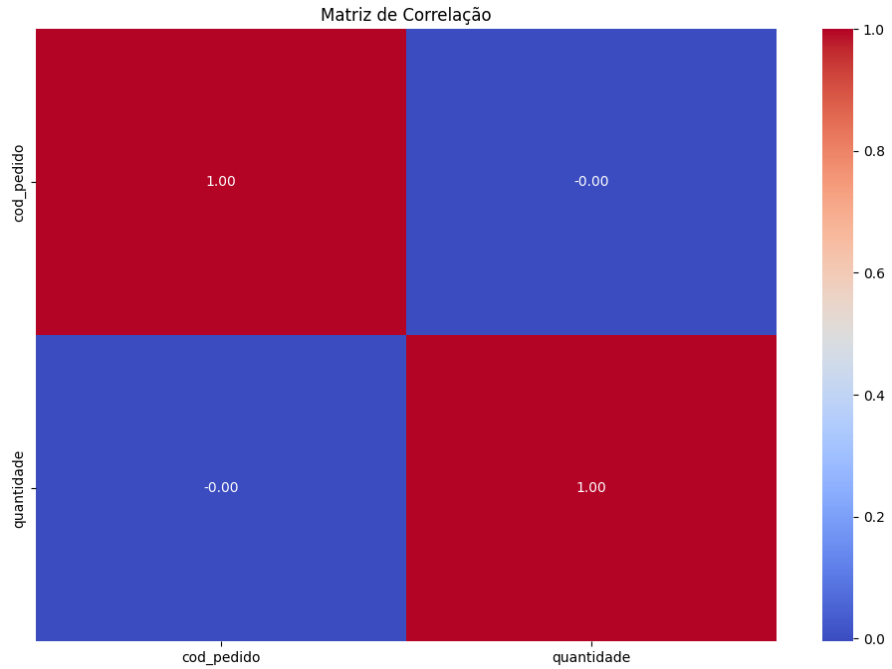
4) Calcule a matriz de correlação dos dados fornecidos. Quais as variáveis que apresentam forte correlação positiva ou negativa? Acrescente a matriz de correlação como uma imagem e anexe-a ao seu relatório.

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

# Carregando os dados
data = pd.read_csv("Sem_Outliers.csv")

# Calculando a matriz de correlação
correlation_matrix = data.corr()

# Plotando a matriz de correlação como um gráfico de calor
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Matriz de Correlação")
plt.show()
```



✓ 0s completed at 9:30 PM