

03

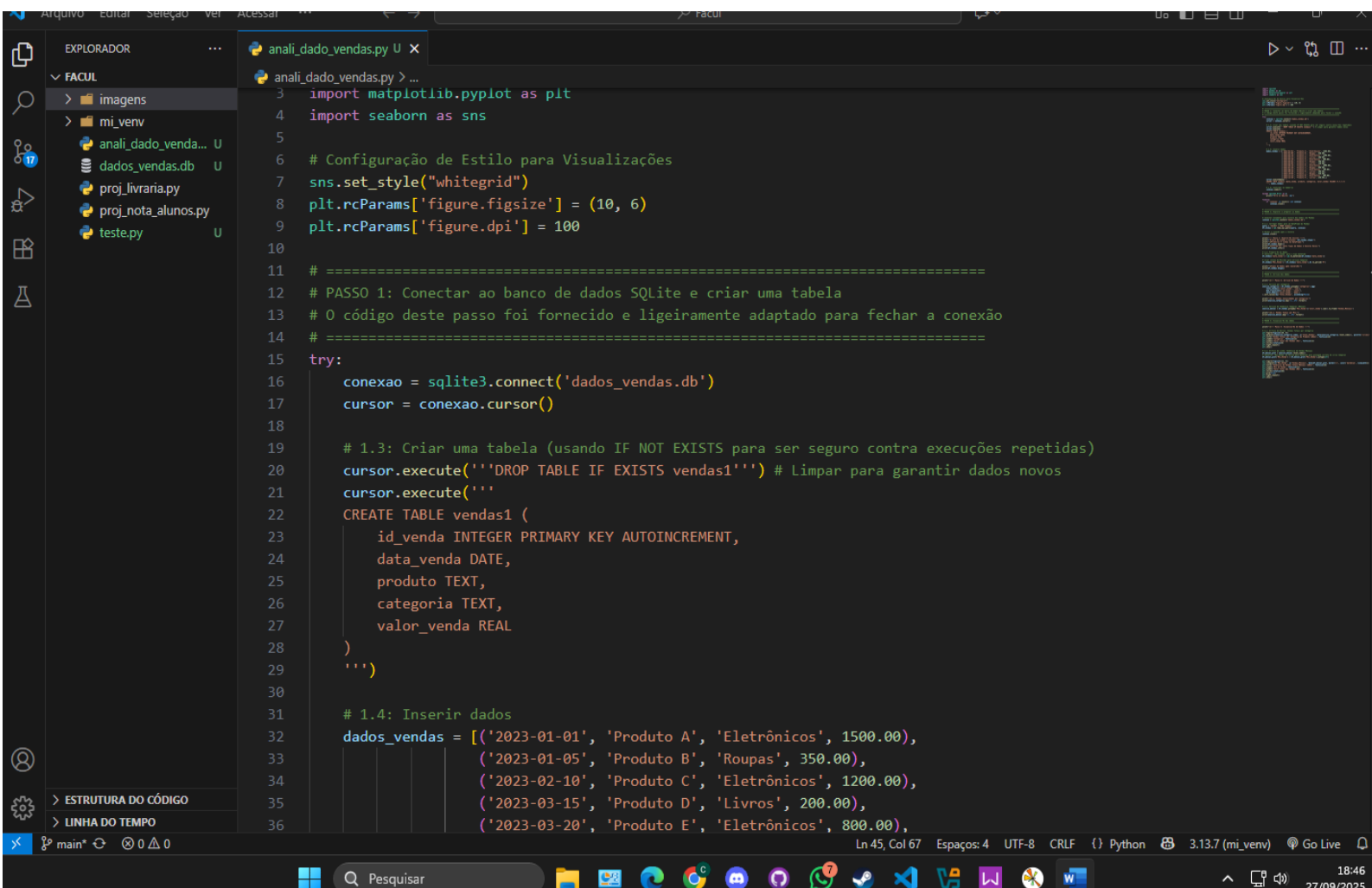
# **TRABALHO DE LINGUAGEM DE PROGRAMAÇÃO EM PYTHON**

## Objetivo Geral

O objetivo deste projeto é analisar os dados de vendas do último ano para identificar padrões, tendências e insights valiosos que possam ser utilizados para melhorar o desempenho e a estratégia de vendas da empresa.

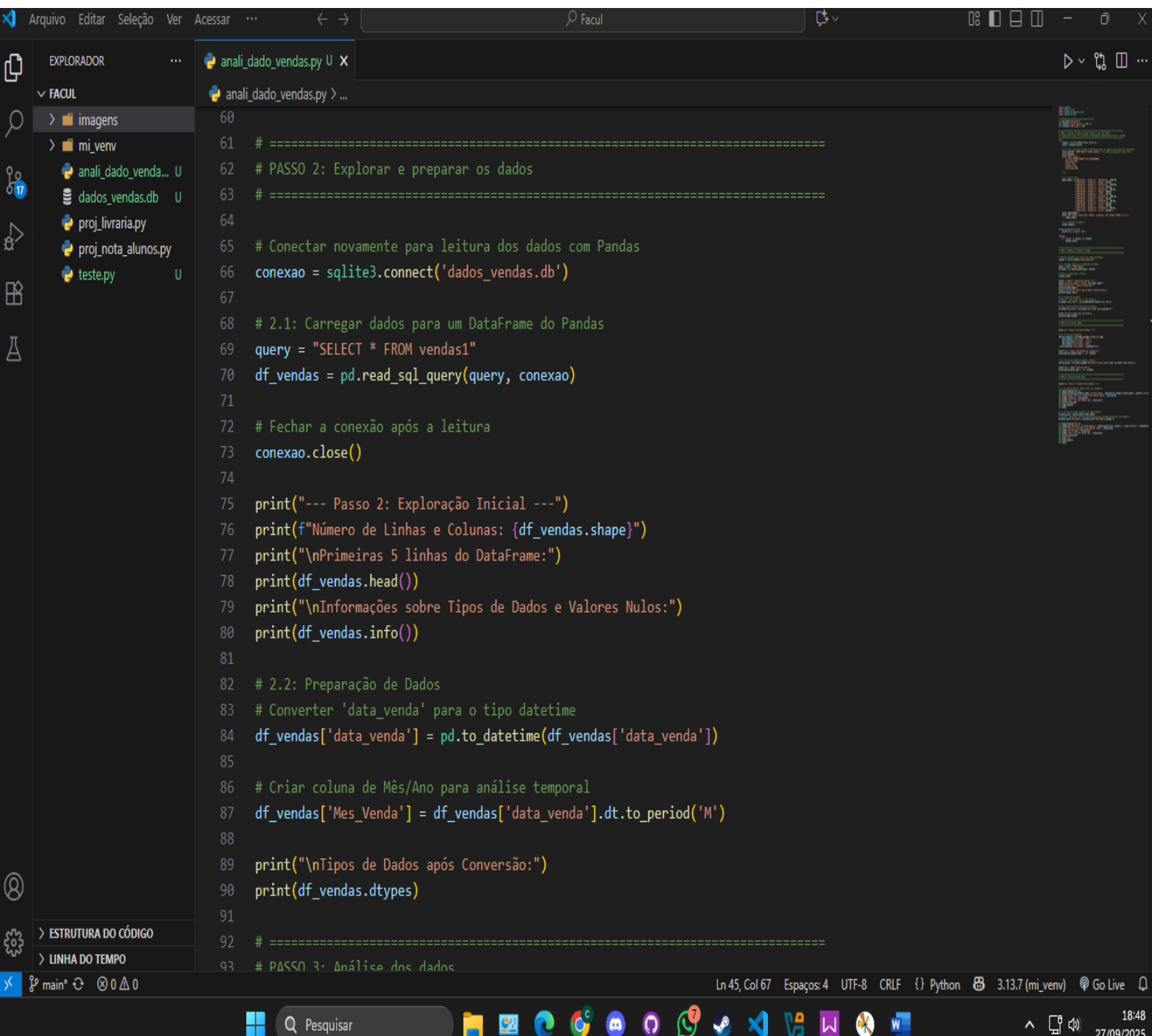
## Ferramentas e Bibliotecas Utilizadas

- **Python:** Linguagem de programação principal.
- **SQLite:** Banco de dados relacional para armazenar os dados de vendas.
- **Pandas:** Biblioteca de manipulação e análise de dados em Python. Essencial para transformar e processar o conjunto de dados.
- **Matplotlib:** Biblioteca de visualização de dados para criação de gráficos estáticos.
- **Seaborn:** Biblioteca de visualização baseada no Matplotlib, oferecendo uma interface de alto nível para criar gráficos estatísticos atraentes e informativo.



```
anali_dado_vendas.py U x
anali_dado_vendas.py > ...
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 # Configuração de Estilo para Visualizações
7 sns.set_style("whitegrid")
8 plt.rcParams['figure.figsize'] = (10, 6)
9 plt.rcParams['figure.dpi'] = 100
10
11 # =====
12 # PASSO 1: Conectar ao banco de dados SQLite e criar uma tabela
13 # O código deste passo foi fornecido e ligeiramente adaptado para fechar a conexão
14 # =====
15 try:
16     conexao = sqlite3.connect('dados_vendas.db')
17     cursor = conexao.cursor()
18
19     # 1.3: Criar uma tabela (usando IF NOT EXISTS para ser seguro contra execuções repetidas)
20     cursor.execute('DROP TABLE IF EXISTS vendas1') # Limpar para garantir dados novos
21     cursor.execute('
22     CREATE TABLE vendas1 (
23         id_venda INTEGER PRIMARY KEY AUTOINCREMENT,
24         data_venda DATE,
25         produto TEXT,
26         categoria TEXT,
27         valor_venda REAL
28     )
29     ')
30
31     # 1.4: Inserir dados
32     dados_vendas = [('2023-01-01', 'Produto A', 'Eletrônicos', 1500.00),
33                     ('2023-01-05', 'Produto B', 'Roupas', 350.00),
34                     ('2023-02-10', 'Produto C', 'Eletrônicos', 1200.00),
35                     ('2023-03-15', 'Produto D', 'Livros', 200.00),
36                     ('2023-03-20', 'Produto E', 'Eletrônicos', 800.00),
```

## Passo 1: Conexão e Preparação do Banco de Dados

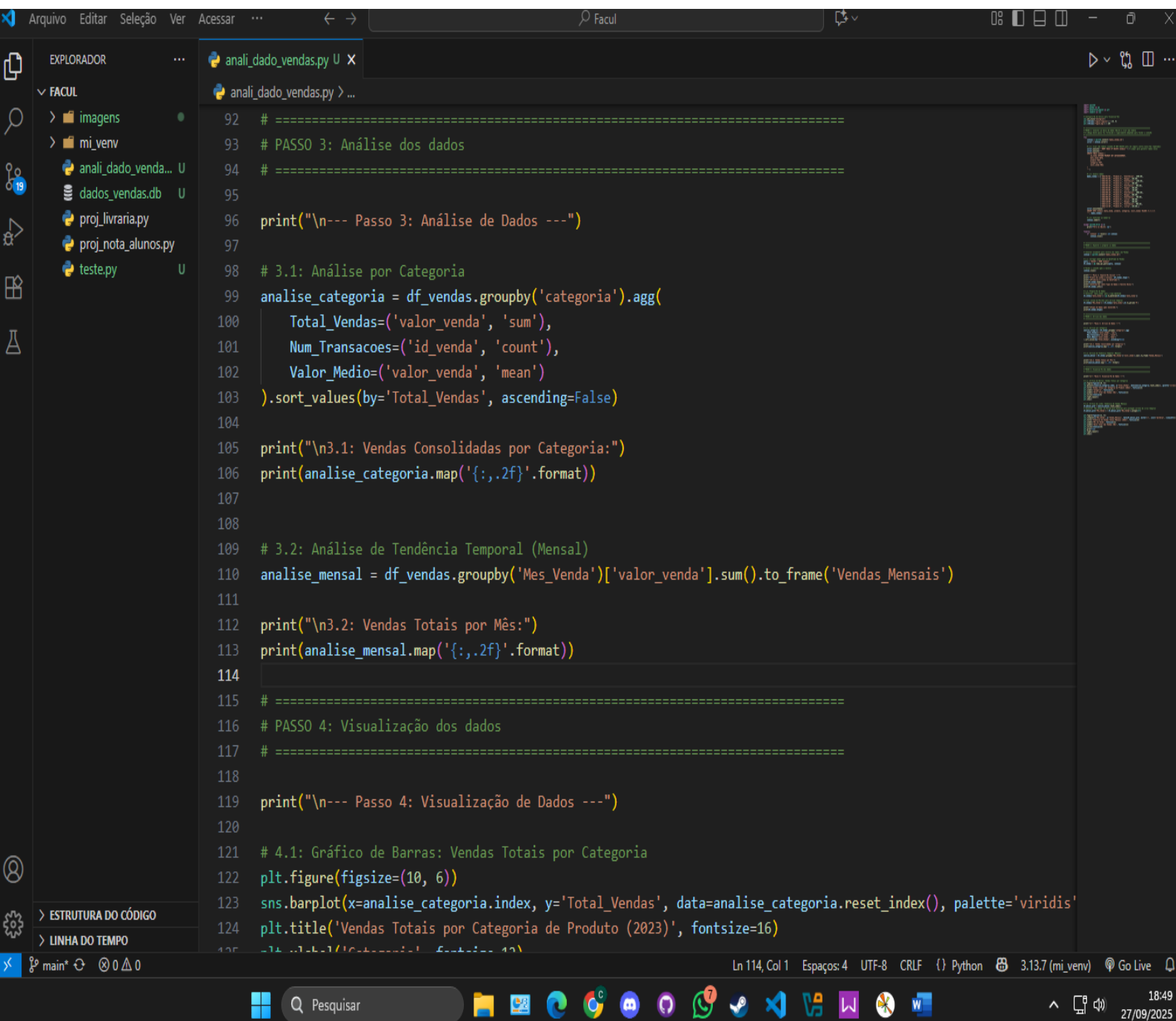


The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'FACUL' with subfolders 'imagens' and 'mi\_venv'. The code editor shows a file named 'anali\_dado\_vendas.py' with the following code:

```
60
61 # =====
62 # PASSO 2: Explorar e preparar os dados
63 # =====
64
65 # Conectar novamente para leitura dos dados com Pandas
66 conexao = sqlite3.connect('dados_vendas.db')
67
68 # 2.1: Carregar dados para um DataFrame do Pandas
69 query = "SELECT * FROM vendas1"
70 df_vendas = pd.read_sql_query(query, conexao)
71
72 # Fechar a conexão após a leitura
73 conexao.close()
74
75 print("--- Passo 2: Exploração Inicial ---")
76 print(f"Número de Linhas e Colunas: {df_vendas.shape}")
77 print("\nPrimeiras 5 linhas do DataFrame:")
78 print(df_vendas.head())
79 print("\nInformações sobre Tipos de Dados e Valores Nulos:")
80 print(df_vendas.info())
81
82 # 2.2: Preparação de Dados
83 # Converter 'data_venda' para o tipo datetime
84 df_vendas['data_venda'] = pd.to_datetime(df_vendas['data_venda'])
85
86 # Criar coluna de Mês/Ano para análise temporal
87 df_vendas['Mes_Venda'] = df_vendas['data_venda'].dt.to_period('M')
88
89 print("\nTipos de Dados após Conversão:")
90 print(df_vendas.dtypes)
91
92 # =====
93 # PASSO 3: Análise dos dados
```

The status bar at the bottom shows the current line and column (Ln 45, Col 67), the number of spaces (Espaços: 4), the encoding (UTF-8), the line ending (CRLF), the Python version (Python 3.13.7), the environment (mi\_venv), and the Go Live status.

**Conexão:** É estabelecida uma conexão com o banco de dados **SQLite (dados\_vendas.db)**. Se o arquivo não existir, ele é criado automaticamente.



```
92 # =====
93 # PASSO 3: Análise dos dados
94 # =====
95
96 print("\n--- Passo 3: Análise de Dados ---")
97
98 # 3.1: Análise por Categoria
99 analise_categoria = df_vendas.groupby('categoria').agg(
100     Total_Vendas=('valor_venda', 'sum'),
101     Num_Transacoes=('id_venda', 'count'),
102     Valor_Medio=('valor_venda', 'mean')
103 ).sort_values(by='Total_Vendas', ascending=False)
104
105 print("\n3.1: Vendas Consolidadas por Categoria:")
106 print(analise_categoria.map('{:,.2f}'.format))
107
108 # 3.2: Análise de Tendência Temporal (Mensal)
109 analise_mensal = df_vendas.groupby('Mes_Venda')['valor_venda'].sum().to_frame('Vendas_Mensais')
110
111 print("\n3.2: Vendas Totais por Mês:")
112 print(analise_mensal.map('{:,.2f}'.format))
113
114 # =====
115 # PASSO 4: Visualização dos dados
116 # =====
117
118 print("\n--- Passo 4: Visualização de Dados ---")
119
120 # 4.1: Gráfico de Barras: Vendas Totais por Categoria
121 plt.figure(figsize=(10, 6))
122 sns.barplot(x=analise_categoria.index, y='Total_Vendas', data=analise_categoria.reset_index(), palette='viridis')
123 plt.title('Vendas Totais por Categoria de Produto (2023)', fontsize=16)
124 plt.xlabel('Categoria', fontsize=12)
125 plt.ylabel('Total_Vendas', fontsize=12)
```

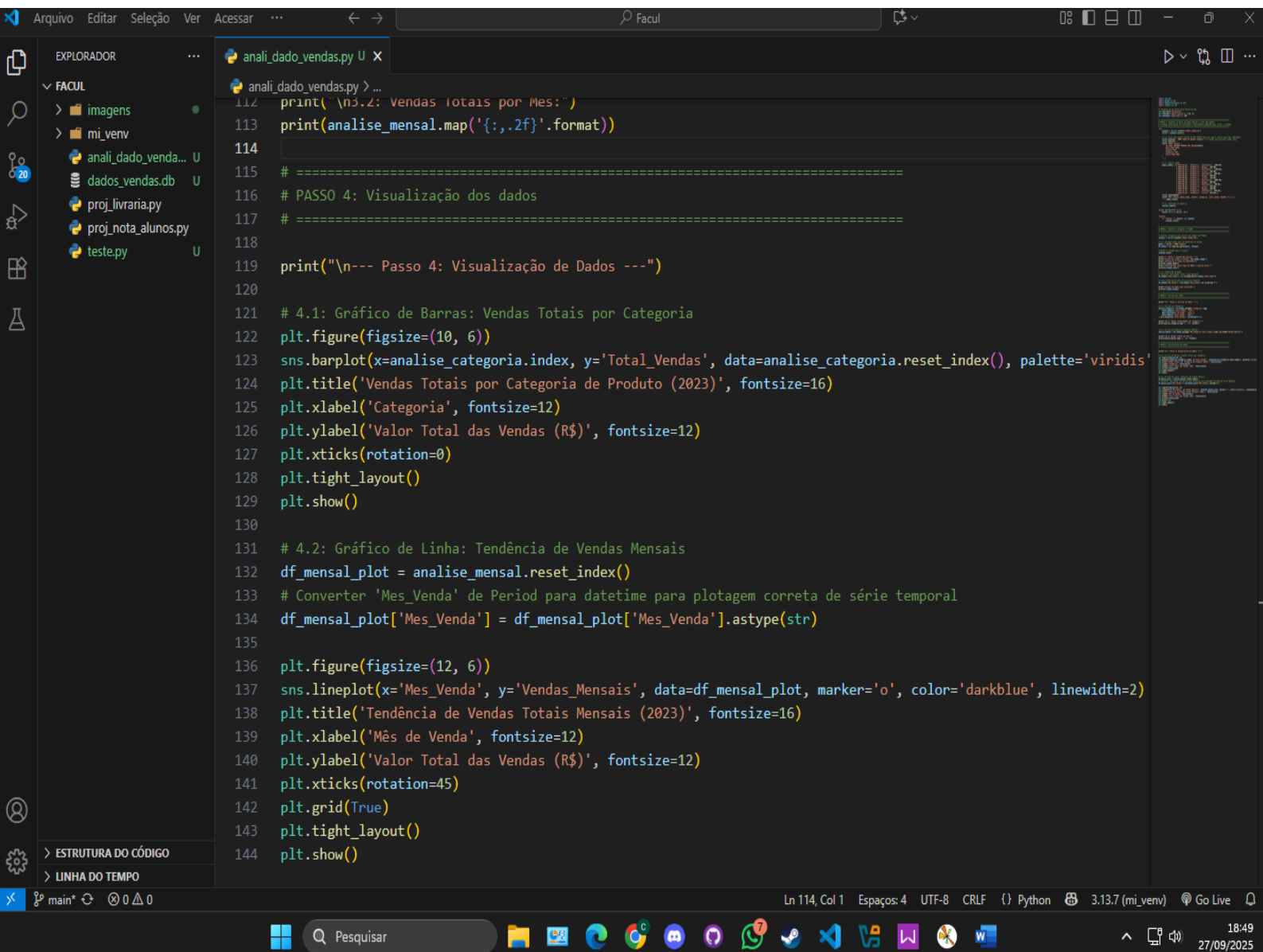
## Visualização dos Dados

As visualizações criadas com **Matplotlib e Seaborn** ajudam a apresentar os insights de forma clara e intuitiva.

**Vendas Totais por Categoria (Gráfico de Barras):** Um gráfico de barras é utilizado para comparar visualmente o desempenho de vendas de cada categoria de produto.

**Receita Mensal (Gráfico de Linhas):** Um gráfico de linhas mostra a receita ao longo do tempo, destacando tendências e padrões sazonais.

**Distribuição dos Valores de Venda (Histograma):** Um histograma é usado para visualizar a frequência dos valores de venda, ajudando a entender a distribuição do ticket médio.



```
112 print('\n3.2: Vendas Totais por Mes:')
113 print(analise_mensal.map('{:,.2f}'.format))
114
115 # =====
116 # PASSO 4: Visualização dos dados
117 # =====
118
119 print("\n--- Passo 4: Visualização de Dados ---")
120
121 # 4.1: Gráfico de Barras: Vendas Totais por Categoria
122 plt.figure(figsize=(10, 6))
123 sns.barplot(x=analise_categoria.index, y='Total_Vendas', data=analise_categoria.reset_index(), palette='viridis')
124 plt.title('Vendas Totais por Categoria de Produto (2023)', fontsize=16)
125 plt.xlabel('Categoria', fontsize=12)
126 plt.ylabel('Valor Total das Vendas (R$)', fontsize=12)
127 plt.xticks(rotation=0)
128 plt.tight_layout()
129 plt.show()
130
131 # 4.2: Gráfico de Linha: Tendência de Vendas Mensais
132 df_mensal_plot = analise_mensal.reset_index()
133 # Converter 'Mes_Venda' de Period para datetime para plotagem correta de série temporal
134 df_mensal_plot['Mes_Venda'] = df_mensal_plot['Mes_Venda'].astype(str)
135
136 plt.figure(figsize=(12, 6))
137 sns.lineplot(x='Mes_Venda', y='Vendas_Mensais', data=df_mensal_plot, marker='o', color='darkblue', linewidth=2)
138 plt.title('Tendência de Vendas Totais Mensais (2023)', fontsize=16)
139 plt.xlabel('Mês de Venda', fontsize=12)
140 plt.ylabel('Valor Total das Vendas (R$)', fontsize=12)
141 plt.xticks(rotation=45)
142 plt.grid(True)
143 plt.tight_layout()
144 plt.show()
```

## Conclusões e Insights

Nesta etapa, os resultados da análise e das visualizações são interpretados para formular conclusões e recomendações estratégicas.

- **Principais Insights:** Apresentação dos achados mais relevantes, como as categorias mais lucrativas, os meses de maior e menor venda, e o ticket médio das transações.
- **Recomendações:** Sugestões práticas para a empresa, como a alocação de mais recursos para as categorias de alto desempenho, planejamento de promoções em meses de baixa sazonalidade ou a criação de campanhas de marketing direcionadas.

Com essa documentação, o projeto se torna mais organizado e fácil de ser compreendido e replicado. Você consegue visualizar a ordem lógica do processo, desde a coleta dos dados até a tomada de decisões estratégicas.

