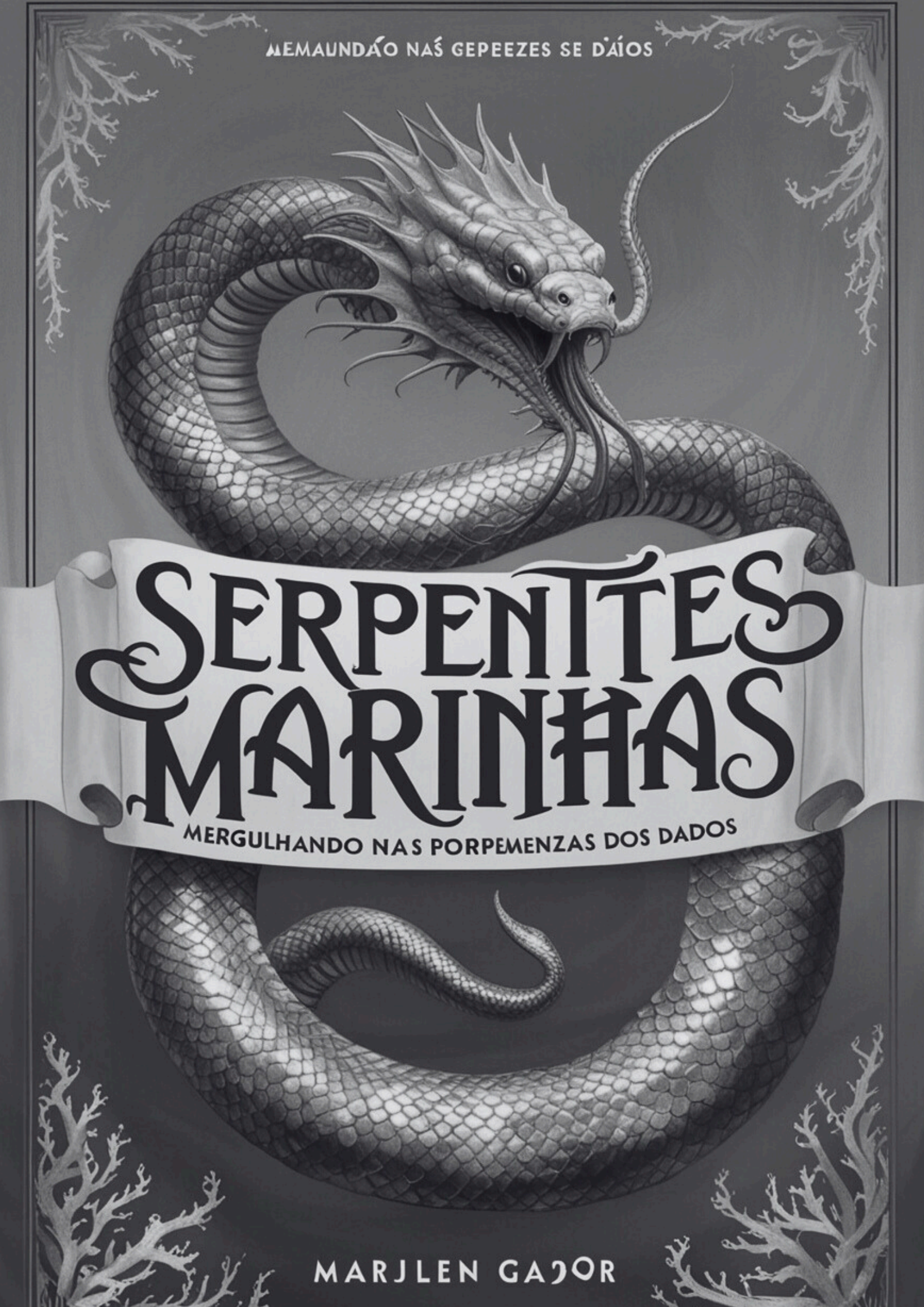
MEMAUNDÁO NAŚ GEPEEZES SE DÁIOS

SERPENTES MARINHAS

MERGULHANDO NAS PORPEMENZAS DOS DADOS

MARJLEN GAJOR

MEMAUNDÃO NAS GEPEEZES SE DÁIOS



SERPENTES MARINHAS

MERGULHANDO NAS PORPEMENZAS DOS DADOS

MARJLEN GAJOR

01

Entre Serpentes e Marés:

Navegando no Oceano de Dados com Python

02

Serpentes no Comando:

Conquistando o Mar de Dados com Scripts Python

03

Navegando pelo Labirinto das Marés:

Técnicas de Análise e Visualização de Dados

04

Entre Serpentes e Algoritmos:

Manipulação Avançada de Dados com Pandas

05

Ondas e Visualizações:

Transformando Números em Histórias no Oceano de Dados

06

Serpentes e Consultas

Dominando o SQL nas Profundezas do Mar de Dados

01

Entre Serpentes e Marés:

Navegando no Oceano de Dados com Python

Bem-vindos, exploradores!
Neste capítulo, vamos iniciar uma jornada emocionante pelo oceano de dados com a ajuda da nossa serpente marinha, Python. Imagine o oceano como um vasto mar de informações, e a serpente Python como nossa guia astuta, pronta para nos conduzir por entre as marés de dados e descobertas.

1. Conhecendo a Serpente Python

Python é uma linguagem de programação poderosa e amigável, perfeita para explorar e analisar dados. Ela nos ajuda a navegar pelas águas complexas dos dados de forma eficiente e intuitiva.
Instalando o Python: Primeiro, precisamos ter a serpente Python pronta para a jornada. Vá a python.org e baixe a versão mais recente.

2. Preparando o Barco: Configurando o Ambiente

Para começar, precisamos configurar nosso ambiente de trabalho. Abra um editor de código, como o VSCode ou PyCharm, e crie um novo arquivo Python.

```python
# Importando a biblioteca pandas para trabalhar com dados
import pandas as pd

# Carregando um arquivo de dados
dados = pd.read_csv('dados_exemplo.csv')

# Mostrando as primeiras linhas dos dados
print(dados.head())
```

Neste código, usamos pandas, uma biblioteca que é como um mapa que nos ajuda a explorar os dados. O pd.read_csv('dados_exemplo.csv') carrega nossos dados e dados.head() exibe as primeiras linhas, permitindo uma visão geral inicial.

3. Navegando nas Correntes: Analisando os Dados

Agora que temos nossos dados, vamos começar a analisá-los. Vamos calcular a média de uma coluna de dados.

Código para Calcular a Média:

```python
# Calculando a média da coluna 'valor'
media = dados['valor'].mean()
print(f'A média dos valores é: {media}')
```

Aqui, mean() nos dá a média dos valores na coluna, ajudando-nos a entender o comportamento dos dados.

4. Explorando os Tesouros: Visualizando os Dados

Para ver padrões nos dados, podemos criar gráficos. Vamos usar a biblioteca matplotlib para isso.
Código para Criar um Gráfico:

```python
import matplotlib.pyplot as plt

# Criando um gráfico simples
plt.plot(dados['valor'])
plt.title('Gráfico de Valores')
plt.xlabel('Índice')
plt.ylabel('Valor')
plt.show()
```

Os gráficos são como mapas do tesouro, revelando padrões e insights ocultos nos dados.

5. Conclusão da Navegação

Parabéns, navegadores! Vocês deram os primeiros passos na exploração do oceano de dados com Python. Com a serpente Python ao seu lado, vocês estão prontos para enfrentar as marés e descobrir novos segredos no mundo dos dados. Continuem explorando e experimentando!
Nos próximos capítulos, mergulharemos em técnicas mais avançadas e ferramentas que vão tornar sua jornada ainda mais emocionante. Até lá, boa exploração!

02

Serpentes no Comando:

Conquistando o Mar de Dados com Scripts Pytho

Bem-vindos ao segundo capítulo!
Agora que já conhecemos a nossa serpente Python e o oceano de dados, é hora de ver como podemos usar Python para dominar esse mar de informações com scripts poderosos. Vamos explorar como criar e usar scripts para manipular e analisar dados de maneira eficiente.

1. O Poder dos Scripts

Os scripts Python são como mapas que nos guiam em nossa jornada pelos dados. Eles nos permitem automatizar tarefas e processar informações de maneira rápida e eficiente.

2. Criando o Primeiro Script

Vamos começar criando um script simples que lê um arquivo de dados e faz uma análise básica. Crie um arquivo chamado analise_dados.py e adicione o seguinte código:

```python
# Importando a biblioteca pandas
import pandas as pd

# Carregando os dados de um arquivo CSV
dados = pd.read_csv('dados_exemplo.csv')

# Exibindo as primeiras linhas dos dados
print(dados.head())
```

Este script carrega um arquivo CSV com dados e exibe as primeiras linhas, ajudando a ter uma visão geral do conteúdo.

3. Processando Dados com Scripts

Vamos adicionar mais funcionalidades ao nosso script para processar os dados. Por exemplo, podemos calcular a soma de uma coluna:

O método to_csv() salva o DataFrame filtrado em um novo arquivo, tornando os resultados disponíveis para futuras consultas.

6. Conclusão

Parabéns! Agora vocês sabem como usar scripts Python para dominar o mar de dados. Desde carregar e processar dados até filtrar e salvar resultados, vocês têm as ferramentas necessárias para realizar análises poderosas. Continuem explorando e aprimorando suas habilidades para enfrentar desafios ainda maiores!

Nos próximos capítulos, exploraremos técnicas avançadas para levar suas habilidades a um nível superior. Até lá, continue navegando e conquistando novos horizontes!

03

Navegando pelo Labirinto das Marés:

Técnicas de Análise e Visualização de Dados

Bem-vindos ao terceiro capítulo!
Agora que já sabemos como usar scripts Python para manipular dados, é hora de explorar técnicas de análise e visualização. Assim como navegadores habilidosos usam mapas para entender melhor as águas, nós usaremos Python para revelar padrões e insights nos dados.

1. Explorando Dados com Pandas

A biblioteca pandas é nossa aliada principal para análise de dados. Vamos começar carregando um conjunto de dados e explorando algumas técnicas básicas de análise.

Carregando Dados:

```python
import pandas as pd

# Carregando o arquivo CSV
dados = pd.read_csv('dados_exemplo.csv')

# Exibindo informações básicas sobre os dados
print(dados.info())
print(dados.describe())
```

O método info() nos dá uma visão geral da estrutura dos dados, enquanto describe() fornece estatísticas descritivas, como média, mediana e desvio padrão.

2. Limpando e Preparando Dados

Dados reais frequentemente precisam ser limpos antes de serem analisados. Vamos ver como lidar com valores ausentes:

```python
# Verificando valores ausentes
print(dados.isnull().sum())

# Preenchendo valores ausentes com a média da coluna
dados['valor'].fillna(dados['valor'].mean(), inplace=True)
```

Aqui, usamos isnull().sum() para identificar colunas com valores ausentes e fillna() para preenchê-los com a média.

3. Visualizando Dados com Matplotlib

Visualizações ajudam a entender melhor os dados. Vamos criar alguns gráficos básicos usando matplotlib.

Histograma:

```python
import matplotlib.pyplot as plt

# Criando um histograma da coluna 'valor'
plt.hist(dados['valor'], bins=20, edgecolor='black')
plt.title('Distribuição de Valores')
plt.xlabel('Valor')
plt.ylabel('Frequência')
plt.show()
```

Histograma nos mostra a distribuição dos dados, revelando padrões importantes.

Gráfico de Dispersão:

```python
# Criando um gráfico de dispersão
plt.scatter(dados['coluna_x'], dados['coluna_y'])
plt.title('Gráfico de Dispersão')
plt.xlabel('Coluna X')
plt.ylabel('Coluna Y')
plt.show()
```

Gráficos de dispersão são ótimos para visualizar a relação entre duas variáveis

.

4. Analisando Correlações

Entender as correlações entre variáveis é essencial. Vamos calcular a matriz de correlação:

python

```python
# Calculando a matriz de correlação
correlacao = dados.corr()
print(correlacao)

# Criando um mapa de calor da matriz de correlação
import seaborn as sns
sns.heatmap(correlacao, annot=True, cmap='coolwarm')
plt.title('Matriz de Correlação')
plt.show()
```

O heatmap de seaborn nos ajuda a visualizar a força e a direção das correlações entre variáveis.

5. Conclusão

Excelente trabalho, exploradores! Agora vocês sabem como analisar e visualizar dados usando Python. Com pandas e matplotlib, vocês têm ferramentas poderosas para navegar pelo labirinto das marés de dados, revelando padrões e insights valiosos.

Nos próximos capítulos, continuaremos a expandir nossas habilidades, explorando técnicas mais avançadas de análise e visualização. Até lá, mantenham-se curiosos e continuem navegando pelos mares de dados!

04

Entre Serpentes e Algoritmos:

Manipulação Avançada de Dados com Pandas

Bem-vindos ao quarto capítulo!

Agora que vocês já estão familiarizados com as técnicas básicas de análise e visualização, é hora de mergulhar mais fundo nas águas da manipulação avançada de dados com a nossa fiel serpente, Python, usando a poderosa biblioteca pandas.

1. Transformações e Limpezas Avançadas

Manipular dados envolve muitas vezes transformações complexas. Vamos começar limpando e transformando nossos dados de maneira mais avançada.

Removendo Duplicatas:

```python
# Removendo linhas duplicadas
dados = pd.read_csv('dados_exemplo.csv')
dados = dados.drop_duplicates()
print(dados)
```

O método drop_duplicates() remove linhas duplicadas do DataFrame, garantindo que nossos dados estejam limpos.

Manipulando Datas:

```python
# Convertendo a coluna 'data' para datetime
dados['data'] = pd.to_datetime(dados['data'])

# Extraindo ano, mês e dia em colunas separadas
dados['ano'] = dados['data'].dt.year
dados['mes'] = dados['data'].dt.month
dados['dia'] = dados['data'].dt.day
print(dados.head())
```

Transformar colunas de data em diferentes partes (ano, mês, dia) pode ser útil para análises temporais mais detalhadas:

2. Agrupamentos e Agregações

Agrupar e agregar dados são técnicas essenciais para sumarizar informações.

Agrupando Dados:

```python
# Agrupando dados por uma coluna e calculando a média
agrupados = dados.groupby('categoria')['valor'].mean()
print(agrupados)
```

O método groupby() nos permite agrupar dados por categorias e aplicar funções de agregação, como mean(), para calcular médias.

3. Aplicando Funções Personalizadas

Podemos aplicar funções personalizadas para transformar os dados de maneiras específicas.

Função Personalizada:

```python
# Definindo uma função personalizada para normalizar dados
def normalizar(x):
    return (x - x.min()) / (x.max() - x.min())

# Aplicando a função na coluna 'valor'
dados['valor_normalizado'] = dados['valor'].apply(normalizar)
print(dados.head())
```

O método apply() permite aplicar funções personalizadas a colunas do DataFrame, possibilitando transformações complexas.

4. Mesclando e Combinando Dados

Mesclar diferentes conjuntos de dados é uma habilidade crucial para análises abrangentes.

Mesclando DataFrame

```python
# Criando um segundo DataFrame para exemplo de mesclagem
novos_dados = pd.DataFrame({
    'categoria': ['A', 'B', 'C'],
    'descricao': ['Item A', 'Item B', 'Item C']
})

# Mesclando os DataFrames com base na coluna 'categoria'
dados_completos = pd.merge(dados, novos_dados, on='categoria', how='left')
print(dados_completos.head())
```

O método merge() combina DataFrames com base em colunas comuns, semelhante a um JOIN em SQL.

5. Pivot e Unpivot

Transformar dados de formatos longos para largos (pivot) e vice-versa (unpivot) é frequentemente necessário.

Criando uma Tabela Pivô:

```
# Criando uma tabela pivô
tabela_pivo = dados.pivot_table(values='valor', index='categoria', columns='ano', aggfunc='sum')
print(tabela_pivo)
```

O método pivot_table() reorganiza os dados, permitindo uma visão sumarizada e estruturada.

6. Conclusão

Parabéns, navegadores! Vocês agora dominam técnicas avançadas de manipulação de dados com pandas. Desde limpeza e transformação até agrupamento e mesclagem, vocês têm as habilidades necessárias para enfrentar desafios complexos no oceano de dados.

Nos próximos capítulos, continuaremos a explorar técnicas ainda mais avançadas para enriquecer suas análises e visualizações. Até lá, continuem praticando e aprimorando suas habilidades!

05

Ondas e Visualizações:

Transformando Números em Histórias no Oceano de Dados

Bem-vindos ao quinto capítulo!

Até agora, aprendemos como manipular e analisar dados. Agora, vamos explorar como transformar esses números em histórias visuais cativantes. As visualizações são como faróis que nos guiam pelo oceano de dados, revelando padrões e insights que os números sozinhos não conseguem mostrar.

1. A Importância das Visualizações

Visualizações tornam os dados mais compreensíveis e acessíveis. Elas ajudam a identificar tendências, padrões e outliers de forma rápida e intuitiva. Vamos usar a biblioteca matplotlib para criar gráficos informativos.

2. Criando Gráficos Básicos

Vamos começar com alguns gráficos básicos para visualizar nossos dados.

Gráfico de Linhas:

```python
import pandas as pd
import matplotlib.pyplot as plt

# Carregando os dados
dados = pd.read_csv('dados_exemplo.csv')

# Criando um gráfico de linhas
plt.plot(dados['data'], dados['valor'])
plt.title('Valores ao Longo do Tempo')
plt.xlabel('Data')
plt.ylabel('Valor')
plt.show()
```

O gráfico de linhas é ideal para visualizar mudanças ao longo do tempo, ajudando a identificar tendências.

Gráfico de Barras:

```python
# Criando um gráfico de barras
plt.bar(dados['categoria'], dados['valor'])
plt.title('Valores por Categoria')
plt.xlabel('Categoria')
plt.ylabel('Valor')
plt.show()
```

O gráfico de barras é ótimo para comparar valores entre diferentes categorias.

3. Gráficos Avançados com Seaborn

A biblioteca seaborn oferece visualizações mais avançadas e esteticamente agradáveis.

Gráfico de Dispersão com Regressão:

```python
import seaborn as sns

# Criando um gráfico de dispersão com linha de regressão
sns.lmplot(x='coluna_x', y='coluna_y', data=dados)
plt.title('Relação entre Coluna X e Coluna Y')
plt.xlabel('Coluna X')
plt.ylabel('Coluna Y')
plt.show()
```

O lmplot mostra a relação entre duas variáveis com uma linha de regressão, ajudando a visualizar correlações.

Gráfico de Caixa:

```python
# Criando um gráfico de caixa
sns.boxplot(x='categoria', y='valor', data=dados)
plt.title('Distribuição de Valores por Categoria')
plt.xlabel('Categoria')
plt.ylabel('Valor')
plt.show()
```

O gráfico de caixa mostra a distribuição dos dados, destacando medianas, quartis e outliers.

4. Histórias com Visualizações Interativas

Visualizações interativas podem tornar a exploração de dados ainda mais envolvente. A biblioteca plotly permite criar gráficos interativos.

Gráfico Interativo:

```python
import plotly.express as px

# Criando um gráfico de linhas interativo
fig = px.line(dados, x='data', y='valor', title='Valores ao Longo do Tempo')
fig.show()
```

Gráficos interativos permitem explorar os dados de forma dinâmica, oferecendo uma experiência mais rica.

5. Contando Histórias com Dashboards

Dashboards combinam várias visualizações para contar uma história completa. Bibliotecas como dash podem ser usadas para criar dashboards interativos.

Exemplo de Dashboard Simples:

```python
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output

# Inicializando o app Dash
app = dash.Dash(__name__)
```

```python
# Layout do dashboard
app.layout = html.Div([
    dcc.Graph(id='grafico-linhas', figure=fig),
    dcc.Slider(id='slider-ano', min=dados['ano'].min(), max=dados['ano'].max(), value=dados['ano'].min())
])

# Callback para atualizar o gráfico com base no slider
@app.callback(
    Output('grafico-linhas', 'figure'),
    [Input('slider-ano', 'value')]
)
def update_figure(selected_year):
    filtered_df = dados[dados['ano'] == selected_year]
    fig = px.line(filtered_df, x='data', y='valor')
    return fig

if __name__ == '__main__':
    app.run_server(debug=True)
```

Este exemplo cria um dashboard simples onde o gráfico de linhas é atualizado com base na seleção de ano no slider.

6. Conclusão

Parabéns, exploradores! Agora vocês sabem como transformar números em histórias visuais poderosas. Usando matplotlib, seaborn, plotly e dash, vocês podem criar visualizações que não apenas informam, mas também envolvem e inspiram. Continuem praticando e experimentando para dominar a arte de contar histórias com dados.

Nos próximos capítulos, vamos explorar técnicas ainda mais avançadas e ferramentas para enriquecer suas habilidades analíticas. Até lá, continuem navegando e descobrindo novos horizontes no oceano de dados!

06

Serpentes e Consultas

Dominando o SQL nas Profundezas do Mar de Dados

Bem-vindos ao sexto e último capítulo!
Agora que vocês aprenderam a navegar e visualizar dados com Python, é hora de mergulhar nas profundezas do mar de dados com SQL, nosso golfinho amigável que nos ajudará a encontrar os tesouros escondidos. Python e SQL juntos formam uma dupla poderosa para explorar e analisar grandes volumes de dados de maneira eficiente.

1. Introdução ao SQL

SQL (Structured Query Language) é uma linguagem usada para gerenciar e manipular bancos de dados relacionais. Vamos começar conectando Python a um banco de dados SQL e executando algumas consultas básicas.
Conectando ao Banco de Dados:

```python
import sqlite3

# Conectando ao banco de dados SQLite (ou qualquer outro banco de dados)
conexao = sqlite3.connect('meu_banco_de_dados.db')

# Criando um cursor para executar comandos SQL
cursor = conexao.cursor()
```

Neste exemplo, usamos SQLite, um banco de dados leve e fácil de usar. Para outros bancos de dados, como PostgreSQL ou MySQL, bibliotecas específicas como psycopg2 ou mysql-connector-python são usadas.

2. Executando Consultas Básicas

Vamos criar uma tabela e inserir alguns dados para praticar.
Criando uma Tabela:

Criando uma Tabela:

```python
# Criando uma tabela
cursor.execute('''
CREATE TABLE IF NOT EXISTS produtos (
    id INTEGER PRIMARY KEY,
    nome TEXT,
    preco REAL
)
''')
conexao.commit()
```

Inserindo Dados:

```python
# Inserindo dados na tabela
cursor.execute('''
INSERT INTO produtos (nome, preco)
VALUES ('Produto A', 19.99), ('Produto B', 29.99), ('Produto C', 39.99)
''')
conexao.commit()
```

Consultando Dados:

```python
# Consultando dados da tabela
cursor.execute('SELECT * FROM produtos')
resultados = cursor.fetchall()
for linha in resultados:
    print(linha)
```

3. Consultas Avançadas

Agora vamos explorar consultas mais complexas, como filtragem, agrupamento e junção de tabelas.

Filtrando Dados:

```python
# Consultando produtos com preço maior que 20
cursor.execute('SELECT * FROM produtos WHERE preco > 20')
resultados = cursor.fetchall()
for linha in resultados:
    print(linha)
```

Agrupando Dados:

```python
# Agrupando produtos por faixa de preço e contando quantos produtos em cada faixa
cursor.execute('''
SELECT
    CASE
        WHEN preco < 20 THEN 'Barato'
        WHEN preco BETWEEN 20 AND 30 THEN 'Médio'
        ELSE 'Caro'
    END AS faixa_preco,
    COUNT(*) AS quantidade
FROM produtos
GROUP BY faixa_preco
''')
resultados = cursor.fetchall()
for linha in resultados:
    print(linha)
```

Juntando Tabelas:

```python
# Criando uma segunda tabela para junção
cursor.execute('''
CREATE TABLE IF NOT EXISTS vendas (
    id INTEGER PRIMARY KEY,
    produto_id INTEGER,
    quantidade INTEGER,
    FOREIGN KEY(produto_id) REFERENCES produtos(id)
)
''')
conexao.commit()

# Inserindo dados na tabela de vendas
cursor.execute('''
INSERT INTO vendas (produto_id, quantidade)
VALUES (1, 5), (2, 2), (3, 3)
''')
conexao.commit()
```

```python
# Consultando dados combinados das tabelas produtos e vendas
cursor.execute('''
SELECT
    produtos.nome,
    vendas.quantidade,
    produtos.preco,
    (vendas.quantidade * produtos.preco) AS total_vendido
FROM vendas
JOIN produtos ON vendas.produto_id = produtos.id
''')
resultados = cursor.fetchall()
for linha in resultados:
    print(linha)
```

4. Integrando Python e SQL para Análise de Dados

Vamos combinar a flexibilidade de Python com o poder de SQL para realizar análises mais complexas.
Usando pandas para Consultas SQL:

Usando pandas para Consultas SQL:

python Copiar código

```python
import pandas as pd

# Executando uma consulta SQL e carregando o resultado em um DataFrame do pandas
df = pd.read_sql_query('SELECT * FROM produtos', conexao)
print(df)

# Realizando uma análise no DataFrame
media_preco = df['preco'].mean()
print(f'Média de preço dos produtos: {media_preco}')
```

5. Conclusão

Parabéns, navegadores! Vocês agora sabem como dominar o SQL nas profundezas do mar de dados, com a ajuda do golfinho SQL, que nos guia com suas consultas ágeis e eficientes. Com as habilidades combinadas de Python e SQL, vocês têm o poder de explorar, manipular e analisar dados de maneira eficiente e eficaz.
Espero que esta jornada pelo oceano de dados tenha sido enriquecedora e que vocês estejam prontos para enfrentar novos desafios e descobrir novas possibilidades no mundo da programação e análise de dados.
Até a próxima aventura!

OBRIGADO POR LER ATÉ AQUI

Este e-book foi gerado por uma Inteligência Artificial e diagramado por mim, Lliliana Alencar para fins didáticos como atividade prática para o bootcamp IA para Devs da Santander pela Dio.me. Segue github com instruções do professor Felipe Aguiar que orientou essa atividade, lá terá instruções passo a passo.

Segue o meu Github, espero ter ajudado até a próxima.
https://github.com/LianaBarbosa