Olinda 24 de Fevereiro de 2024
Edson Henrique Ferreira da Silva
Desafio 01 parte B

EXTRAÇÃO DE DADOS EM SQL COM PYTHON

Para realizar a consulta ao banco de dados e transformar os resultados em um arquivo CSV utilizando Python, você pode utilizar a biblioteca `pandas` em conjunto com a biblioteca `sqlalchemy` para a conexão ao banco de dados. Aqui está um exemplo básico de como pode ser feito:

```
import pandas as pd
from sqlalchemy import create_engine

# Configurações de conexão ao banco de dados
db_url = "mysql://seu_usuario:senha@localhost/seu_banco_de_dados"  # Substitua pelos seus dados
engine = create_engine(db_url)

# Consultas SQL
consulta_pedidos = "SELECT * FROM orders;"
consulta_produtos = "SELECT * FROM products;"
consulta_itens_pedido = "SELECT * FROM order_items;"

# Realizando as consultas e armazenando os resultados em DataFrames
df_pedidos = pd.read_sql(consulta_pedidos, engine)
df_produtos = pd.read_sql(consulta_produtos, engine)
df_itens_pedido = pd.read_sql(consulta_itens_pedido, engine)

# Realizando a transformação dos dados conforme o script SQL anterior

# Agregação dos dados
df_agregado = pd.merge(df_itens_pedido, df_produtos, on='product_id')
df_agregado['total_item_price'] = df_agregado['quantity'] * df_agregado['prince']
df_agregado['total_order_price'] = df_agregado.groupby('order_id')
['total_item_price'].transform('sum')

# Salvando o DataFrame resultante em um arquivo CSV
df_agregado.to_csv('data_lake_sales.csv', index=False)
```

Para este exemplo se assume que estamos usando um banco de dados MySQL. Certifique-se de substituir as variáveis de configuração (`db_url`, por exemplo) pelos seus próprios detalhes de conexão.

Certifique-se de ter as bibliotecas `pandas` e `sqlalchemy` instaladas em seu ambiente Python:

```
pip install pandas sqlalchemy
```

Este é um exemplo básico, podemos precisar adaptar conforme a estrutura real do banco de dados e requisitos específicos. Além disso, ajuste a lógica de agregação conforme necessário. Se tiver mais perguntas ou precisar de esclarecimentos adicionais, fique à vontade para perguntar!

A estrutura já abrange a consulta ao banco de dados, a transformação dos dados conforme o script SQL anterior e a exportação para um arquivo CSV. No entanto, aqui estão algumas sugestões adicionais podemos considerar dependendo dos requisitos específicos do seu projeto:

1. Tratamento de Erros:

- Adicione tratamentos de erros para lidar com possíveis exceções durante a execução, como problemas de conexão ao banco de dados ou erros na execução de consultas.

2. Configuração Segura de Senhas:

- Evite incluir senhas diretamente no código. Considere o uso de variáveis de ambiente ou arquivos de configuração seguros para armazenar informações sensíveis.

3. Logging:

- Adicione logs para registrar informações úteis durante a execução, facilitando a depuração e monitoramento.

4. Paralelização:

- Se o volume de dados for muito grande, você pode considerar estratégias de paralelização para acelerar o processo. Isso pode ser feito dividindo a carga de trabalho em tarefas menores que podem ser executadas em paralelo.

5. Testes Unitários:

- Se este script for parte de um projeto maior, considere a inclusão de testes unitários para garantir a robustez do código.

6. Otimização de Desempenho:

- Dependendo da complexidade das consultas e do volume de dados, pode ser útil otimizar o desempenho da consulta SQL. Analise os planos de execução para garantir que os índices estejam sendo utilizados eficientemente.

Lembre-se de que estas são sugestões gerais e a aplicabilidade delas dependerá dos requisitos específicos do seu projeto. Se tiver mais detalhes ou requisitos específicos, ficarei feliz em fornecer orientações mais precisas.

O uso de DataFrames, como os proporcionados pela biblioteca Pandas em Python, é particularmente benéfico em diversas situações, especialmente quando você está lidando com manipulação e análise de dados tabulares. Abaixo estão alguns cenários em que o uso de DataFrames pode ser aplicado:

1. **Leitura e Escrita de Dados:**
    - DataFrames são úteis para ler dados de diferentes formatos (CSV, Excel, SQL, etc.) e escrever os resultados de volta para esses formatos.

    Exemplo:

```
import pandas as pd

# Leitura de dados
df = pd.read_csv('dados.csv')
```

```
# Escrita de dados
df.to_excel('resultados.xlsx', index=False)
```

2,Manipulação de Dados:

- Pandas oferece uma ampla variedade de métodos para manipulação de dados, incluindo seleção, filtragem, ordenação e agregação.

Exemplo:

```
# Filtrando dados
df_filtrado = df[df['coluna'] > 10]

# Agrupamento e agregação
df_agregado = df.groupby('categoria').agg({'valor': 'sum'})
```

3.Limpeza e Transformação de Dados:

- DataFrames facilitam a aplicação de transformações nos dados, como preenchimento de valores ausentes, renomeação de colunas, e outras operações de limpeza.

Exemplo:

```
# Preenchimento de valores nulos
df = df.fillna(0)

# Renomeação de colunas
df = df.rename(columns={'antiga_coluna': 'nova_coluna'})
```


```
# Preenchimento de valores nulos
df = df.fillna(0)

# Renomeação de colunas
df = df.rename(columns={'antiga_coluna': 'nova_coluna'})
```


4.Análise Exploratória de Dados (EDA):

- Ao realizar EDA, você pode usar DataFrames para criar visualizações e resumos estatísticos para entender melhor a distribuição dos dados.

Exemplo:

```
# Criando um gráfico de barras
df['categoria'].value_counts().plot(kind='bar')
```

5. Integração com Outras Bibliotecas:

- DataFrames do Pandas podem ser facilmente integrados com outras bibliotecas populares para análises mais avançadas, como Matplotlib, Seaborn, Scikit-Learn, entre outras.

Exemplo:

```
import matplotlib.pyplot as plt

# Plotando um gráfico de dispersão
plt.scatter(df['idade'], df['salario'])
```

6. Manipulação de Séries Temporais:

- Se os dados possuem informações de séries temporais, Pandas oferece funcionalidades específicas para manipulação e análise desse tipo de dados.

Exemplo:

```
# Convertendo coluna de datas para formato datetime
df['data'] = pd.to_datetime(df['data'])

# Criando uma nova coluna com o ano
df['ano'] = df['data'].dt.year
```

DataFrames são uma ferramenta poderosa e versátil para trabalhar com dados em Python, oferecendo uma gama de funcionalidades para facilitar a análise e manipulação de dados.

Para exibição de gráficos, você pode utilizar bibliotecas como Matplotlib ou Seaborn em conjunto com Pandas para visualizar dados de DataFrames de forma mais intuitiva. Aqui estão algumas maneiras de exibir gráficos a partir de DataFrames:

1. **Gráficos Básicos com Pandas:**
    - Pandas tem alguns métodos integrados para criar gráficos básicos diretamente do DataFrame.

    Exemplo:

```
import pandas as pd

# Criando um DataFrame de exemplo
data = {'Categoria': ['A', 'B', 'C'], 'Valor': [10, 20, 15]}
df = pd.DataFrame(data)

# Gráfico de barras
df.plot(kind='bar', x='Categoria', y='Valor', title='Gráfico de Barras')
```

Matplotlib e Seaborn:

- Matplotlib é uma biblioteca de visualização mais flexível e Seaborn é baseada em Matplotlib, fornecendo estilos adicionais e funções simplificadas.

Exemplo:

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


# Gráfico de dispersão com Seaborn
sns.scatterplot(data=df, x='ColunaX', y='ColunaY')
plt.title('Gráfico de Dispersão')
plt.show()
```

3. Visualizações Avançadas:

- Para visualizações mais avançadas e personalizadas, você pode usar funções específicas do Matplotlib e Seaborn.

Exemplo:

```
# Criando um gráfico de linha com Matplotlib
plt.plot(df['Data'], df['Valor'], marker='o', linestyle='-', color='b')
plt.title('Gráfico de Linha')
plt.xlabel('Data')
plt.ylabel('Valor')
plt.show()
```

4.Gráficos Interativos:

- Se precisar de gráficos interativos, pode considerar bibliotecas como Plotly ou Bokeh.

Exemplo (Plotly):

```
import plotly.express as px

# Gráfico de dispersão interativo com Plotly
fig = px.scatter(df, x='ColunaX', y='ColunaY', title='Gráfico Interativo')
fig.show()
```

As opções de visualização podem variar dependendo do tipo de dados que você está lidando e da história que deseja contar com seus gráficos. Explore as funcionalidades dessas bibliotecas para criar visualizações atraentes e informativas.

A implementação de extração de dados via Python, acesso ao banco de dados, tratamento e manutenção de uma base em CSV para consultas e análises posteriores é uma abordagem bastante comum e eficaz. Essa prática oferece diversas vantagens:

1. **Automatização do Processo:**
    - Utilizando scripts em Python, é possível automatizar o processo de extração e transformação de dados. Isso é especialmente útil para lidar com atualizações frequentes no banco de dados.

2. **Flexibilidade na Manipulação de Dados:**

- Python, juntamente com bibliotecas como Pandas, proporciona uma ampla gama de funcionalidades para manipulação de dados. Você pode realizar limpezas, transformações e agregações conforme necessário.

3. **Portabilidade e Compatibilidade:**
    - O formato CSV é amplamente suportado e pode ser facilmente compartilhado entre diferentes ferramentas e plataformas. Isso torna a base de dados mais acessível para a equipe de ciência de dados.

4. **Controle sobre o Formato dos Dados:**
    - Ao manter uma base em CSV, você tem controle total sobre o formato dos dados. Pode adaptar a estrutura do arquivo CSV conforme as necessidades específicas da equipe de ciência de dados.

5. **Desempenho:**
    - Trabalhar com um arquivo CSV local muitas vezes é mais rápido do que consultar diretamente um banco de dados, especialmente para operações simples e consultas frequentes.

6. **Reprodutibilidade:**
    - Scripts Python fornecem uma maneira reprodutível de executar o processo de extração e transformação de dados. Isso é valioso para garantir consistência nas análises.

7. **Integração com Ferramentas de Visualização:**
    - Dados em formato CSV podem ser facilmente importados em ferramentas de visualização como Excel, Tableau, ou integrados diretamente em scripts de visualização em Python.

8. **Facilidade de Compartilhamento:**
    - A equipe de ciência de dados pode compartilhar os arquivos CSV gerados, facilitando a colaboração e o compartilhamento de resultados.

Devemos considerar fatores como segurança e privacidade ao lidar com dados sensíveis. implementar práticas adequadas para proteger a integridade e a confidencialidade dos dados durante todo o processo.