Integração de APIs sobre Aquecimento Global

Introdução

Este código tem como objetivo coletar dados de três APIs relacionadas ao aquecimento global, processá-los e gerar um banco de dados consolidado. Ele facilita o entendimento dos impactos climáticos por meio de análises de temperaturas globais, redução do gelo ártico e aumento de temperaturas oceânicas.

Público-alvo:
Este código é destinado a analistas de dados, pesquisadores e qualquer pessoa interessada em estudar o impacto das mudanças climáticas.

Saída:
A saída é um conjunto de DataFrames com dados estruturados, que podem ser exportados para arquivos CSV através do arquivo .db gerado.

Nível de privacidade:
Os dados utilizados são públicos e provenientes de APIs abertas.

Pré-requisitos

Ambiente:

- Python 3.8 ou superior.

Bibliotecas:

- pandas
- requests
- datetime
- plyer

Arquivos:
Nenhum arquivo externo é necessário para execução do código, além das bibliotecas previamente instaladas.

APIs Utilizadas

1. **Temperaturas**
   - URL: `https://global-warming.org/api/temperature-api`
   - Dados: Médias de temperaturas globais ao longo dos anos.
2. **Gelo Ártico**
   - URL: `https://global-warming.org/api/arctic-api`
   - Dados: Medidas da redução do gelo no ártico.
3. **Temperaturas Oceânicas**
   - URL: `https://global-warming.org/api/ocean-warming-api`
   - Dados: Dados de aumento da temperatura dos oceanos.

Chaves de acesso:
As APIs são públicas e não requerem autenticação.

Bibliotecas Utilizadas

1. **pandas**: Manipulação e análise de dados.
2. **requests**: Realiza requisições HTTP para acessar as APIs.
3. **plyer**: Notificações sobre erros ou eventos importantes.
4. **datetime**: Manipulação de datas e horários.

Funções Criadas

1. `alerta(api_nome, status_code)`
   - **Parâmetros:** Nome da API, código de status HTTP.
   - **Retorno:** Nenhum.
   - **Descrição:** Exibe uma notificação caso ocorra erro no carregamento de dados da API.
2. `carregar_api(url, nome_api)`
   - **Parâmetros:** URL da API, nome da API.
   - **Retorno:** DataFrame com os dados da API.
   - **Descrição:** Faz a requisição HTTP e transforma os dados em um DataFrame do pandas.
3. `substituir_outliers_por_media(df, coluna_valor, coluna_ano)`

- **Parâmetros:** DataFrame com os dados a serem processados, nome da coluna que contém valores negativos ou nulos, nome da coluna que indica o ano.
- **Retorno:** DataFrame com os valores negativos ou nulos substituídos pela média dos valores válidos agrupados pela coluna_ano.
- **Descrição:** Identifica valores outliers (negativos ou nulos) na coluna especificada e os substitui pela média dos outros valores do mesmo grupo - ano. Em seguida, retorna o DataFrame atualizado.

Tratamentos Aplicados

- **Erros:**
  Tratamento de erros no carregamento das APIs, com notificações exibidas ao usuário em caso de falha.
- **Limpeza de Dados:**
  Conversão dos dados JSON para DataFrames pandas.
  - Base Gelo: remoção das colunas não relevantes para o trabalho e função para identificar valores outliers e substituir pela média anual; cálculo de média anual;
  - Base Oceano: remoção da coluna ‘error’ do dataframe final por ser irrelevante ao trabalho;
  - Base Temperatura: conversão dos dados usando .tolist(), cálculo de média anual.

Método de Saída

Formato:
Os dados estruturados são organizados em DataFrames pandas e podem ser exportados para CSV.

O resultado final foi disposto no arquivo bd1.db, a partir da junção dos dataframes obtidos das três tabelas analisadas.

Estrutura:
Cada DataFrame contém colunas especificadas de acordo com os dados brutos retornados pelas APIs.

É possível realizar queries SQL no arquivo .db, no formato:

```
query = "SELECT * FROM table_merged"
```

```
df = pd.read_sql(query, conn)

print(df)
```

Exemplo de saída de consulta SQL:

year oceano - temp anomala média_anual temp_media_station \

0 1851 -0.05 NaN NaN

1 1852 0.01 NaN NaN

2 1853 0.05 NaN NaN

3 1854 0.00 NaN NaN

4 1855 -0.01 NaN NaN

.. ..

169 2020 0.75 21.741667 1.310833

170 2021 0.70 22.103333 1.145000

171 2022 0.64 21.367500 1.160833

172 2023 0.67 20.303333 1.435000

173 2024 0.94 20.770909 1.552727

temp_media_land

0 NaN

1 NaN

2 NaN

3 NaN

4 NaN

.. ...

169 1.005833

170	0.844167
171	0.890000
172	1.170833
173	1.285455

[174 rows x 5 columns]

Exemplo de Consulta

Consulta:
Código de exemplo para carregar e visualizar os dados:

```
url = "https://global-warming.org/api/temperature-api"
nome_api = "temperaturas"
dados = carregar_api(url, nome_api)
print(dados.head())
```


Resultado:
Primeiras linhas do DataFrame exibidas no console.

Versionamento

Bibliotecas:

```
pandas==1.3.3
requests==2.26.0
plyer==2.0.0
```

Referências

- Global Warming API

CODERHOUSE