Armazenamento de Dados

O que fazer com dados retornados por uma API?

Após receber os dados retornados por uma API, existem diversas coisas que podem ser feitas, dependendo do contexto e objetivo do projeto. Algumas das ações mais comuns incluem:

Armazenar os dados em uma variável ou estrutura de dados para uso posterior;

Limpar os dados para remover valores duplicados ou inconsistentes;

Analisar os dados para extrair insights ou informações relevantes;

Visualizar os dados usando gráficos ou tabelas para tornar a análise mais fácil e intuitiva; e

Transformar os dados para um formato mais adequado para o projeto.

Por exemplo:

import requests

# Faz a requisição para a API

response = requests.get("https://api.example.com/data")

# Verifica se a resposta foi bem-sucedida

if response.status\_code == 200:

# Armazena os dados em uma variável

data = response.json()

# Limpa os dados para remover valores duplicados ou inconsistentes

cleaned\_data = [item for item in data if item["value"] > 0]

# Analisa os dados para extrair informações relevantes

total = sum(item["value"] for item in cleaned\_data)

average = total / len(cleaned\_data)

# Imprime o resultado da análise

print(f"Média dos valores: {average}")

Como persistir esses dados

Os dados retornados por uma API podem ser persistidos de diversas formas, dependendo do contexto e objetivo do projeto. Algumas das opções mais comuns incluem:

Armazenamento em arquivos (CSV, JSON, TXT, etc.);

Armazenamento em bancos de dados (SQL ou NoSQL); e

Armazenamento em sistemas de armazenamento distribuído (Hadoop, etc.).

Por exemplo:

import csv

# Armazena os dados em um arquivo CSV

with open("data.csv", "w", newline="") as file:

writer = csv.writer(file)

writer.writerow(["name", "value"])

for item in cleaned\_data:

writer.writerow([item["name"], item["value"]])

# Armazena os dados em um banco de dados SQL

import sqlite3

conn = sqlite3.connect("data.db")

cursor = conn.cursor()

cursor.execute("CREATE TABLE IF NOT EXISTS data (name text, value real)")

for item in cleaned\_data:

cursor.execute("INSERT INTO data (name, value) VALUES (?, ?)", (item["name"], item["value"]))

conn.commit()

conn.close()

Utilizando Parquet

O Parquet é um formato de arquivo de dados de coluna que permite armazenar e compartilhar grandes quantidades de dados estruturados de maneira eficiente. É amplamente utilizado em aplicações de Big Data e Análise de Dados por sua capacidade de armazenar grandes quantidades de dados de forma compacta e acessível.

Aqui está um exemplo de como usar o Parquet em Python:

import pandas as pd

# Cria um DataFrame de exemplo

data = {'col1': [1, 2, 3], 'col2': ['A', 'B', 'C']}

df = pd.DataFrame(data)

# Salva o DataFrame como um arquivo Parquet

df.to\_parquet("data.parquet", index=False)

# Carrega o arquivo Parquet como um novo DataFrame

parquet\_df = pd.read\_parquet("data.parquet")

print(parquet\_df)

Este exemplo cria um DataFrame simples, salva o DataFrame como um arquivo Parquet e, em seguida, carrega o arquivo como um novo DataFrame. O resultado será um DataFrame que contém os dados armazenados no arquivo Parquet.

Parquet x CSV

Podemos comparar o uso do Parquet e o uso do CSV da seguinte forma:

Armazenamento de dados: O Parquet é um formato de armazenamento de dados de colunas otimizado para o processamento de grandes quantidades de dados em sistemas distribuídos, enquanto o CSV é um formato de arquivo simples e amplamente utilizado para armazenar dados em tabelas, com suporte a tipos de dados simples, como inteiros, reais e strings;

Eficiência de armazenamento: O Parquet oferece compressão eficiente e armazenamento de dados estruturados de forma eficiente, enquanto o CSV não oferece compressão e armazenamento de dados estruturados pode ser ineficiente para grandes quantidades de dados;

Leitura e escrita: O Parquet é otimizado para realizar leituras rápidas de subconjuntos de colunas, enquanto o CSV é mais adequado para a leitura e escrita de dados em uma tabela simples; e

Integração com ferramentas: O Parquet é suportado por diversas ferramentas de análise de dados, enquanto o CSV é compatível com a maioria das ferramentas de análise de dados e de planilhas.

Em resumo, o Parquet é uma boa opção para o armazenamento de grandes quantidades de dados em sistemas distribuídos, enquanto o CSV é uma opção mais adequada para o armazenamento de dados simples em tabelas em uma única máquina.

Projeto - Extração de dados da API gratuita Mercado Bitcoin

Objetivo: O objetivo deste exercício é extrair dados de cotação de Bitcoin da API gratuita Mercado Bitcoin e armazená-los em um formato apropriado para análise posterior.

Contexto: Bitcoin é uma criptomoeda digital descentralizada, que permite realizar transações financeiras sem a necessidade de intermediários, como bancos. A cotação do Bitcoin é influenciada por diversos fatores, como a oferta e a demanda, notícias econômicas e políticas governamentais.

Passos:

Acesse a documentação da API Mercado Bitcoin em https://www.mercadobitcoin.com.br/api-doc/;

Crie uma conta de API gratuita na plataforma Mercado Bitcoin;

Utilize a biblioteca requests do Python para realizar uma requisição à API e obter os dados de cotação de Bitcoin;

Armazene os dados obtidos em um formato apropriado, como CSV ou Parquet; e

Realize uma análise exploratória dos dados armazenados, como a criação de gráficos de tendência de preço ao longo do tempo.

Resultado esperado: Ao final deste exercício, você deverá ter extraído os dados de cotação de Bitcoin da API Mercado Bitcoin e armazená-los de forma adequada para análise posterior. Além disso, você deverá ter realizado uma análise exploratória dos dados obtidos e compreender o contexto básico em que o Bitcoin se encontra no mercado financeiro.

Referências

CSV: https://docs.python.org/3/library/csv.html

SQLite: https://docs.python.org/3/library/sqlite3.html

Parquet: https://arrow.apache.org/docs/python/parquet.html e https://butecotecnologico.com.br/apache-parquet-explicado/

Parquet com Pandas: https://pandas.pydata.org/docs/user\_guide/io.html#parquet

Tipos de Arquivos em BigData: https://medium.com/rescuepoint/tipos-de-dados-em-bigdata-6ab0debec30a