In [ ]:
```
import pandas as pd
import requests
import sqlite3
import os
import time
```

Conexão com o Banco de Dados

In [ ]:
```
db_path = os.path.abspath("../datasets/data.db")
conn = sqlite3.connect(db_path)
cursor = conn.cursor()
```

Criação das tabelas

In [ ]:
```
conn.execute("PRAGMA foreign_keys = ON;")

cursor.executescript(
    """
    CREATE TABLE IF NOT EXISTS deputados_56_detalhes (
        id INTEGER NOT NULL PRIMARY KEY,
        nomeCivil TEXT NOT NULL,
        cpf TEXT NOT NULL,
        dataNascimento TEXT,
        dataFalecimento TEXT,
        escolaridade TEXT,
        profissoes TEXT,
        redeSocial TEXT -- Salvar lista como uma str separada por ;
    );
    CREATE INDEX IF NOT EXISTS idx_deputados_56_detalhes_nomeCivil ON dep

    CREATE TABLE IF NOT EXISTS partidos (
        id INTEGER NOT NULL PRIMARY KEY,
        sigla TEXT NOT NULL UNIQUE,
        nome TEXT NOT NULL,
        urlLogo TEXT,
        uri TEXT NOT NULL
    );

    CREATE TABLE IF NOT EXISTS deputados_56 (
        id INTEGER NOT NULL PRIMARY KEY,
        nome TEXT NOT NULL,
        siglaUf TEXT NOT NULL,
        siglaPartido TEXT NOT NULL,
        urlFoto TEXT NOT NULL,
        uri TEXT NOT NULL,
        FOREIGN KEY (id) REFERENCES deputados_56_detalhes(id) ON DELETE C
    );

    CREATE TABLE IF NOT EXISTS despesas (
        id_deputado INTEGER NOT NULL,
        ano INTEGER NOT NULL,
        mes INTEGER NOT NULL,
        tipoDespesa TEXT NOT NULL,
        codDocumento INTEGER NOT NULL,
        tipoDocumento TEXT NOT NULL,
```

```
        codTipoDocumento INTEGER NOT NULL,
        dataDocumento TEXT NOT NULL,
        numDocumento TEXT NOT NULL,
        valorDocumento REAL NOT NULL,
        valorLiquido REAL NOT NULL,
        urlDocumento TEXT,
        nomeFornecedor TEXT NOT NULL,
        cnpjCpfFornecedor TEXT NOT NULL,
        codLote INTEGER NOT NULL,
        FOREIGN KEY (id_deputado) REFERENCES deputados_56_detalhes(id) ON
    );
    CREATE INDEX IF NOT EXISTS idx_despesas_deputados ON despesas(numDocu
    """
)

conn.commit()
conn.close()
```

1) Deputados da 56° Legislatura 2019-2022

In [ ]:
```
base_url = "https://dadosabertos.camara.leg.br/api/v2/deputados"
params = {
    "idLegislatura": 56,
    "dataInicio": "2021-01-01",
    "dataFim": "2021-12-31",
    "ordem": "ASC",
    "ordenarPor": "siglaUF"
}


with sqlite3.connect(db_path) as conn:
    all_deputados = []
    url = base_url

    while url:
        try:
            response = requests.get(url, params=params if url == base_url
            response.raise_for_status()

            data = response.json()
            if "dados" in data and data["dados"]:
                df = pd.DataFrame(data["dados"])
                df = df[["id", "nome", "siglaUf", "siglaPartido", "urlFot
                all_deputados.append(df)


            url = None
            if "links" in data:
                for link in data["links"]:
                    if link.get("rel") == "next":
                        url = link.get("href")
                        break

        except requests.RequestException as e:
            print(f"Erro na requisição: {e}")
            break

        time.sleep(0.5)
```

```
    if all_deputados:
        deputados_df = pd.concat(all_deputados, ignore_index=True)
        deputados_df.to_sql("deputados_56", conn, if_exists="replace", in
        print(f"Salvos {len(deputados_df)} registros no banco.")

    else:
        print("Nenhum dado encontrado.")
```

2) Detalhes de cada Deputado

In [ ]:
```
# Função para converter listas de redes sociais para string
def convert_rede_social(rede_social):
    if isinstance(rede_social, list):
        return '; '.join(rede_social)
    return rede_social
```

In [ ]:
```
try:
    with sqlite3.connect(db_path) as conn:
        cursor = conn.cursor()
        cursor.execute("SELECT id, uri FROM deputados_56")
        uris = cursor.fetchall()

        for id, uri in uris:
            response = requests.get(uri)
            response.raise_for_status()
            data = response.json()


            if "dados" in data and data["dados"]:
                dados = data["dados"]
                detalhes = {
                    "id": id,
                    "nomeCivil": dados.get("nomeCivil", ""),
                    "cpf": dados.get("cpf", ""),
                    "dataNascimento": dados.get("dataNascimento", ""),
                    "dataFalecimento": dados.get("dataFalecimento", ""),
                    "escolaridade": dados.get("escolaridade", ""),
                    "profissoes": dados.get("profissoes", ""),
                    "redeSocial": convert_rede_social(dados.get("redeSoci
                }



                # Verificando se o deputado já está na tabela
                cursor.execute("SELECT id FROM deputados_56_detalhes WHER
                result = cursor.fetchone()


                if not result:
                    cursor.execute(
                        """
                        INSERT INTO deputados_56_detalhes
                        (id, nomeCivil, cpf, dataNascimento, dataFalecime
                        VALUES (?, ?, ?, ?, ?, ?, ?, ?)
                        """,
                        (
```

```
                            detalhes["id"],
                            detalhes["nomeCivil"],
                            detalhes["cpf"],
                            detalhes["dataNascimento"],
                            detalhes["dataFalecimento"],
                            detalhes["escolaridade"],
                            detalhes["profissoes"],
                            detalhes["redeSocial"]
                        )
                    )
                    conn.commit()

                time.sleep(0.5)
                print(f"Descrição do Deputado {detalhes['id']} inserido c

            else:
                print("Nenhum dado encontrado para a URI:", uri)

except requests.RequestException as e:
    print(f"Erro na requisição: {e}")
```

2.1) Inserindo profissão de cada Deputado

In [ ]:
```
def convert_profissoes(profissoes):
    if isinstance(profissoes, list):
        # Garante que só strings válidas sejam utilizadas e ignora valore
        return "; ".join(
            prof["titulo"] for prof in profissoes if "titulo" in prof and
        )
    return ""
```

In [ ]:
```
with sqlite3.connect(db_path) as conn:
    cursor = conn.cursor()

    for id, uri in uris:
        try:
            response = requests.get(
                f"https://dadosabertos.camara.leg.br/api/v2/deputados/{id
            )
            response.raise_for_status()
            data = response.json()

            if "dados" in data and data["dados"]:
                lista_profissoes = data["dados"]
                profissao = convert_profissoes(lista_profissoes)

                cursor.execute(
                    "UPDATE deputados_56_detalhes SET profissoes = ? WHER
                    (profissao, id)
                )
                conn.commit()

                time.sleep(0.5)
                print(f"Profissões inseridas com sucesso para ID {id}: {p
            else:
                print(f"Nenhuma profissão encontrada para ID {id}")

        except requests.RequestException as e:
```

```
            print(f"Erro na requisição para ID {id}: {e}")
```

4) Detalhes do partido referente ao(s) Deputado(s)

In [ ]:
```
url = (
    "https://dadosabertos.camara.leg.br/api/v2/partidos?"
    "dataInicio=2022-01-01&"
    "dataFim=2022-12-31&"
    "ordem=ASC&"
    "ordenarPor=sigla"
)

todos_partidos = []


while url:
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()

        for partido in data["dados"]:
            partido_info = {
                "id": partido.get("id", ""),
                "sigla": partido.get("sigla", ""),
                "nome": partido.get("nome", ""),
                "urlLogo": partido.get("urlLogo", ""),
                "uri": partido.get("uri", "")
            }
            todos_partidos.append(partido_info)

        url = None
        for link in data["links"]:
            if link.get("rel") == "next":
                url = link.get("href")
                break


    else:
        print(f"Erro na requisição ({response.status_code})")
        break


if todos_partidos:
    df_final = pd.DataFrame(todos_partidos)

    with sqlite3.connect(db_path) as conn:
        df_final.to_sql("partidos", conn, if_exists="replace", index=Fals

    print("✅ Dados dos partidos armazenados com sucesso no banco de dado
else:
    print("❌ Nenhum dado foi coletado.")
```

4.1) Consultando uri dos partidos e atualizando urlLogo

In [ ]:
```
conn = sqlite3.connect(db_path)
cursor = conn.cursor()

cursor.execute("SELECT id, uri FROM partidos")
partidos = cursor.fetchall()

for id, uri in partidos:
    response = requests.get(uri)
    response.raise_for_status()
    data = response.json()

    if "dados" in data and data["dados"]:
        partido = data["dados"]
        url_logo = partido.get("urlLogo", "")

        cursor.execute("UPDATE partidos SET urlLogo = ? WHERE id = ?", (u
        conn.commit()

        print("URL da logo atualizada com sucesso:", id)

    else:
        print("Nenhum dado encontrado para a URI:", uri)
```

5) Despesas de cada Deputado

In [ ]:
```
with sqlite3.connect(db_path) as conn:
    deputados = pd.read_sql_query("SELECT id FROM deputados_56_detalhes",

deputados_ids = deputados["id"].tolist()
despesas_nao_encontradas = []

for deputado in deputados_ids:
    url_despesas = (
        f"https://dadosabertos.camara.leg.br/api/v2/deputados/{deputado}/
        f"idLegislatura=56&ano=2022&itens=100&ordem=ASC&ordenarPor=dataDo
    )

    while url_despesas:
        try:
            response = requests.get(url_despesas)
            response.raise_for_status()
            data = response.json()

            if "dados" in data and data["dados"]:
                despesas = pd.DataFrame(data["dados"])


                if not despesas.empty:
                    despesas["id_deputado"] = deputado  # Adicionar o ID
                    despesas = despesas[[
                        "id_deputado", "ano", "mes", "tipoDespesa", "codD
                        "codTipoDocumento", "dataDocumento", "numDocument
                        "valorLiquido", "urlDocumento", "nomeFornecedor",
                    ]]


                    with sqlite3.connect(db_path) as conn:
                        despesas.to_sql("despesas", conn, if_exists="appe
```

```
                    time.sleep(0.5)

            else:
                break


            url_despesas = None
            for link in data.get("links", []):
                if link.get("rel") == "next":
                    url_despesas = link.get("href")
                    break

        except requests.RequestException as e:
            print(f"Erro de requisição para deputado {deputado}: {e}")
            despesas_nao_encontradas.append(deputado)
            break
        except Exception as e:
            print(f"Erro inesperado ao processar deputado {deputado}: {e}
            despesas_nao_encontradas.append(deputado)
            break


if despesas_nao_encontradas:
    print("Deputados com falha na requisição de despesas:", despesas_nao_
```