Teste técnico Engenheiro de Dados Pleno

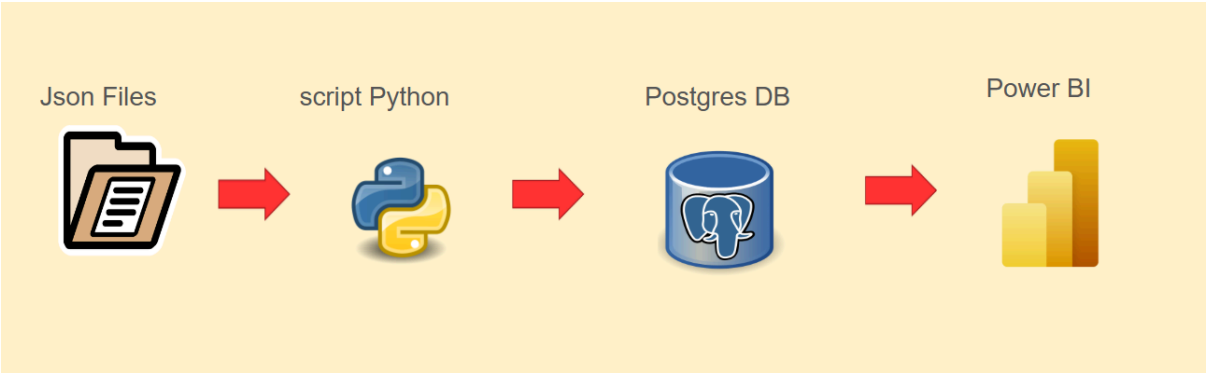
Cliente: Secretaria de Estado de Saúde de Goiás

Candidato: Luciano Martins dos Santos
Data: 21/03/2025

Objetivo: O trabalho de integração de dados feito teve o objetivo de, com código mais enxuto possível, extrair apenas os dados necessários para obter o máximo de performance tanto na extração dos dados dos arquivos json como nas consultas analíticas finais em SQL no banco de dados PostgreSQL.

As tecnologias utilizadas (Python, PostgreSQL e Power BI) são open source, foram criadas e testadas em ambiente On-Premisse.

Diagrama da solução:



Análises solicidadas:

1 - Top 10 Condições

2 - Top 10 Prescrições de Medicamentos

3 - Quantidade de gêneros masculinos

1 - Criar tabelas no postgreSQL:

```
CREATE TABLE patients (
    id SERIAL PRIMARY KEY,
    resource_type varchar(20),
    gender varchar(10)
);

CREATE TABLE medications (
    id SERIAL PRIMARY KEY,
    resource_type varchar(20),
    medication text
);

CREATE TABLE conditions (
    id SERIAL PRIMARY KEY,
    resource_type varchar(20),
    condition varchar(100)
);
```

2 - script Python teste_tecnico_luciano.ipynp.

2.1 Import de bibliotecas necessário:

```
import pandas as pd
import os
import glob
import psycopg2
```

pandas: criação de dataframes
glob e os: usados em conjunto para a leitura de todos arquivos json de uma vez num único dataframe pandas
psycopg2: necessário para abrir conexão com o PostgreSQL

2.2 Carregamento dos arquivos json dentro de um dataframe pandas

```
path = "C:\\Users\\lucia\\Documents\\teste_engenheiro-main\\teste_engenheiro-main\\data"
```

```
# a função glob faz com que eu consiga carregar todos os arquivos json do path de uma só vez para uma variável
json_files = glob.glob(os.path.join(path, '*.json'))
```

```
# esta estrutura, método concat combinado com a List Comprehension, faz com que eu transforme todo conteúdo num único dataframe.
pacientes_source =  pd.concat([pd.read_json(file) for file in json_files], ignore_index=True)
```

2.3 Abre conexão com o PostgreSQL

```
# Abre conexão com o PosrgresSQL. Pode ser reproduzido facilmente em qualuer instalação do PgAdmin
conn = psycopg2.connect(host='localhost', database='postgres', user='postgres', password='admin')
cur = conn.cursor()
```

2.4 Carrega a tabela ***patients*** somente com dos dados necessários para a análise: *resourceType = 'Patient' e gender*

```
# Carrega a tabela do patients no PostgresSQL
for i in range(len(pacientes_source)):
    if pacientes_source.iloc[i].entry.get('resource').get('resourceType') == 'Patient':
        cur.execute("""
            INSERT INTO patients (resource_type, gender)
            VALUES (%s, %s)
        """, (
            pacientes_source.iloc[i].entry.get('resource').get('resourceType'),
            pacientes_source.iloc[i].entry.get('resource').get('gender')
        ))
conn.commit()
```

2.5 Carrega a tabela ***conditions*** somente com dos dados necessários para a análise: *resourceType = 'Condition' e a descrição das conditions*

```
# Carrega a tabela do conditions no PostgresSQL
for i in range(len(pacientes_source)):
    if pacientes_source.iloc[i].entry.get('resource').get('resourceType') == 'Condition':
        cur.execute("""
            INSERT INTO conditions (resource_type, condition)
            VALUES (%s, %s)
        """, (
            pacientes_source.iloc[i].entry.get('resource').get('resourceType'),
            pacientes_source.iloc[i].entry.get('resource').get('code').get('text')
        ))
conn.commit()
```

2.6 Carrega a tabela ***medications*** somente com dos dados necessários para a análise: *resourceType = 'MedicationRequest' e a descrição das medications*

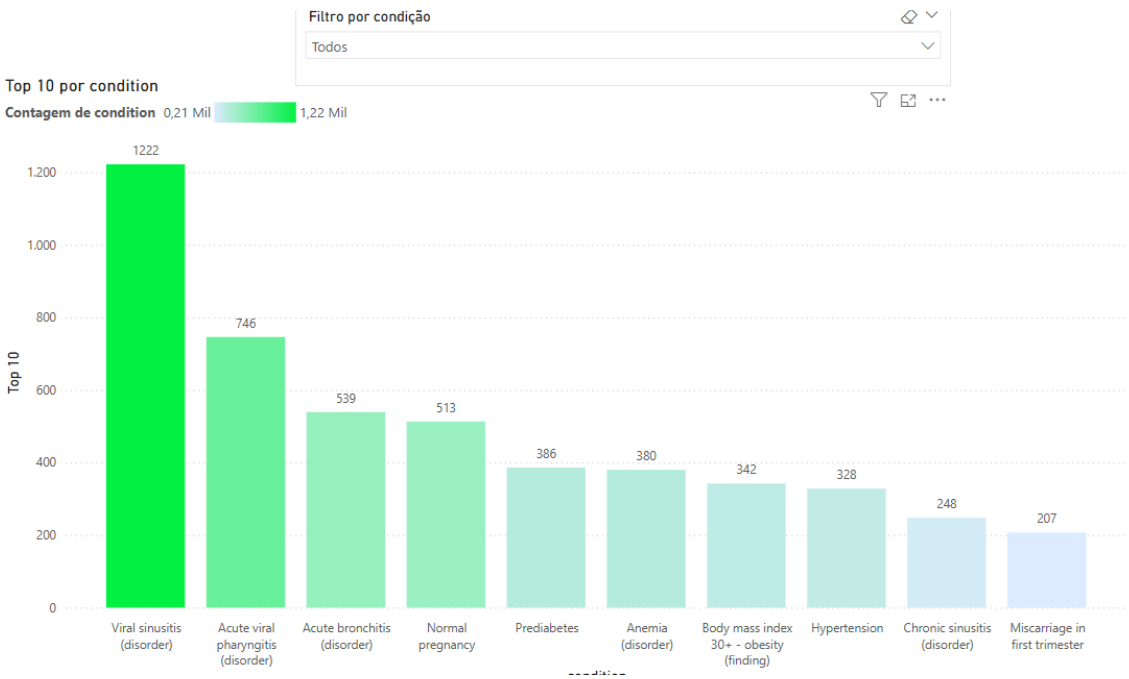
```
# Carrega a tabela do medications no PostgresSQL
for i in range(len(pacientes_source)):
    if pacientes_source.iloc[i].entry.get('resource').get('resourceType') == 'MedicationRequest':
        cur.execute("""
            INSERT INTO medications (resource_type, medication)
            VALUES (%s, %s)
        """, (
            pacientes_source.iloc[i].entry.get('resource').get('resourceType'),
            pacientes_source.iloc[i].entry.get('resource').get('medicationCodeableConcept').get('text')
        ))
conn.commit()
```

3. Executar Queries no banco de dados PostgresSQL para análise de dados e performance

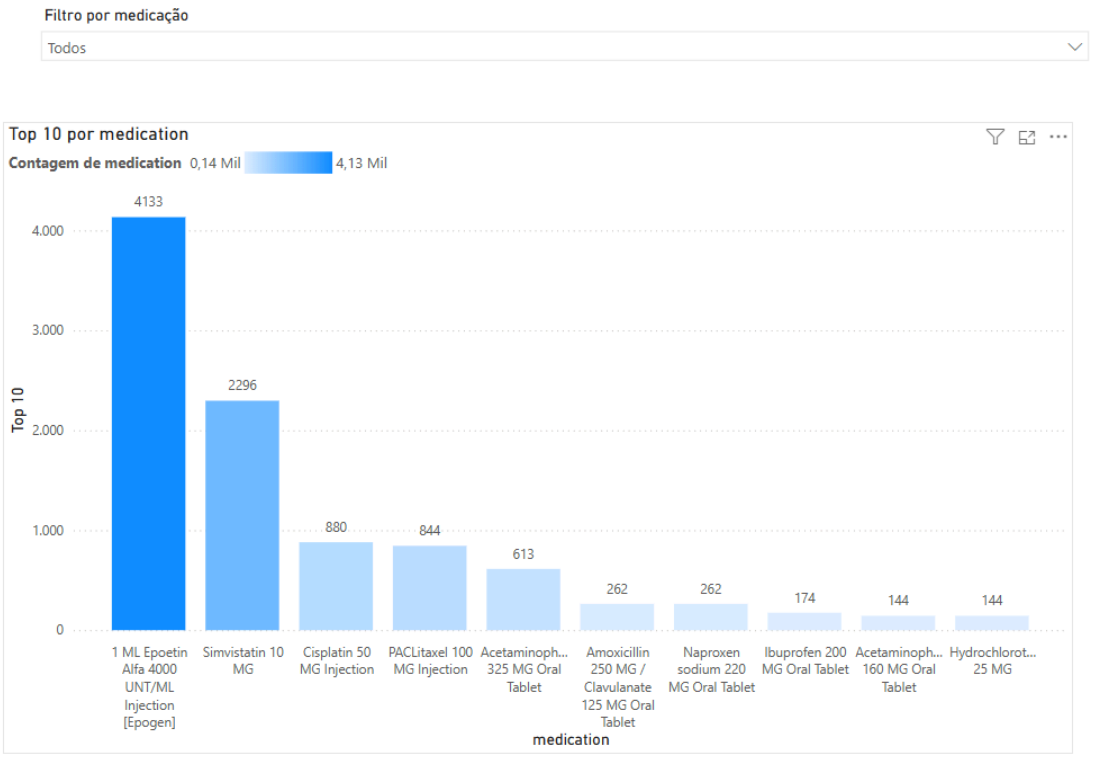
```
select medication,count(*)
from medications
group by medication
order by 2 desc
LIMIT 10;

SELECT gender, COUNT(*) AS qtd
FROM patients
GROUP BY gender;

select condition, count(*)
from conditions
group by condition
order by 2 desc
limit 10;
```

4. Abrir arquivo teste_tecnico_luciano.pbix

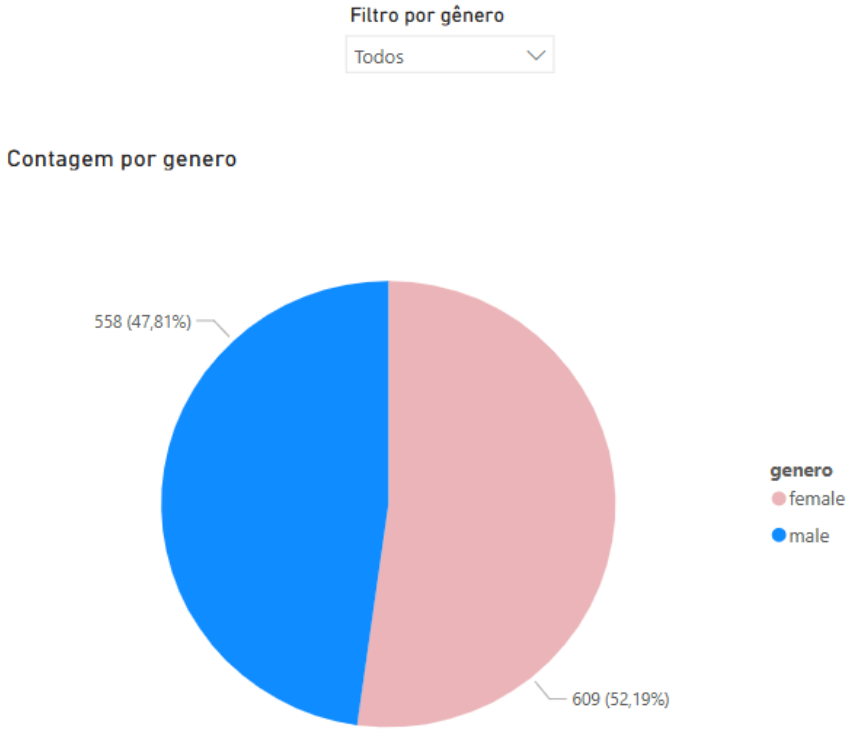
4.1 Top 10 Condições



4.2 Top 10 Medicações



4.3 Quantidade por gênero



Obs: o arquivo com as consultas SQL para rodar no postgreSQL estão juntas com o pacote.

Conclusão: as extrações de dados dos arquivos json e as consultas SQL estão bem otimizadas, levam segundos para rodarem.