**AVALIAÇÃO – CIENTISTA DE DADOS I**

**Nome**: Natali Lourenço Nascimento

1. **Integração e manutenção da base de comércio exterior de 143 países:**

**1.1**

**1.2**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import os

import requests

import time

paises = ["BRA", "ARG", "USA", "CHN", "AFG", ...]

anos = [2019, 2020, 2021]

def consultar\_api(pais\_origem, pais\_destino, ano):

try:

url = f"https://api.comercioexterior.com/{pais\_origem}/{pais\_destino}/{ano}"

response = requests.get(url)

response.raise\_for\_status()

return response.json()

except requests.exceptions.RequestException as e:

print(f"Erro ao consultar API: {e}")

return None

base\_dir = "comercio\_exterior"

os.makedirs(base\_dir, exist\_ok=True)

def salvar\_dados(pais\_origem, pais\_destino, ano, dados):

caminho = os.path.join(base\_dir, pais\_origem, pais\_destino)

os.makedirs(caminho, exist\_ok=True)

arquivo = os.path.join(caminho, f"{ano}.txt")

with open(arquivo, 'w') as f:

f.write(str(dados))

def realizar\_consultas():

consultas\_diarias = 0

limite\_diario = 2500

for ano in anos:

for pais\_origem in paises:

for pais\_destino in paises:

if pais\_origem != pais\_destino and pais\_destino != "WLD" and pais\_origem != "WLD":

dados = consultar\_api(pais\_origem, pais\_destino, ano)

if dados:

salvar\_dados(pais\_origem, pais\_destino, ano, dados)

consultas\_diarias += 1

if consultas\_diarias >= limite\_diario:

print("Limite diário atingido. Aguardando até o próximo dia...")

time.sleep(86400)

consultas\_diarias = 0

print("Todas as consultas foram realizadas.")

if \_\_name\_\_ == "\_\_main\_\_":

realizar\_consultas()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Quantidade de soja o Mundo importou do Brasil em 2020**
* Localizar o arquivo correspondente na estrutura de diretórios criada para armazenar os dados de comércio exterior;
* O arquivo em questão, corresponde à importação/exportação entre Brasil (BRA) e Mundo (WLD) no ano de 2020 (BRA/WLD.txt);

O código acima foi desenvolvido em Python, mas a requisição dos dados poderia ter sido feita também através de VBA (Visual Basic for Applications), incluindo no código o tratamento de erros, segurança e testes de validação. Dependendo do caso, seria mais prático para apresentação de relatórios e manutenção dos dados da API para outros setores que não tem familiaridade com programação e até mesmo para o envio dos arquivos para manipulação em Power BI.

* 1. **Consultas do comércio exterior por mês e ano**

Primeiramente, poderia ser desenvolvido um script para automatização das tarefas envolvendo periodicidade (semanalmente, diariamente, por hora etc.). Em seguida, o tratamento de exceções buscando evitar erros na conexão com a API ou outros problemas de rede e arquivo. É imperativo também pensar na segurança dos dados no processo de atualização e armazenamento, além de sua validação mantendo sua integridade. E por fim, considerar a escalabilidade do processo para lidar com o grande volume de dados à medida que o número de consultas aumentar.

1. **Manter o dataset sempre atualizado evitando erros**
2. **Baixar:** Implementar um script para baixar os arquivos do site da Receita Federal, utilizando a biblioteca “requests” em Python para realizar as solicitações HTTP e baixar os arquivos.
3. **Verificar**: a integridade do arquivo, para certificação de que os arquivos não estão corrompidos.
4. **Extrair:** Durante a extração dos arquivos, caso os arquivos estejam compactados (por exemplo, em formato ZIP), usar a biblioteca “zipfile” para extrair os conteúdos dos arquivos baixados.
5. **Empilhar :** Utilizar “pandas” para carregar os dados dos arquivos extraídos e empilhá-los em um único dataset, se necessário, usando “pandas.concat” para combinar os datasets de diferentes arquivos. Ex:

import pandas as pd

arquivo1 = 'dados\_empresas/arquivo1.csv'

arquivo2 = 'dados\_empresas/arquivo2.csv'

df1 = pd.read\_csv(arquivo1)

df2 = pd.read\_csv(arquivo2)

1. **Validar e Limpar Dados:** Verificar se os dados estão no formato esperado, remover duplicatas ou tratar valores ausentes.
2. **Armazenar os dados atualizados:** salvar o dataset resultante em um formato adequado (como CSV ou banco de dados).
3. **Automatizar o processo de atualização:** configurar o script para rodar periodicamente (por exemplo, mensal, semestral ou anual) para baixar os novos arquivos do site da Receita Federal.
4. **Monitorar e Registrar Eventos:** implementação de um sistema de logging para registrar eventos importantes durante o processo de atualização, como download de novos arquivos, tratamento de erros, e atualização do dataset.
5. **Consulta SQL**

SELECT

e.employee\_name AS employee\_name,

first\_dept.department\_name AS first\_department,

current\_dept.department\_name AS current\_department,

COUNT(DISTINCT edh2.department\_id) AS num\_departments\_worked

FROM

employees e

LEFT JOIN

employee\_department\_history edh1 ON e.employee\_id = edh1.employee\_id

LEFT JOIN

departments first\_dept ON edh1.department\_id = first\_dept.department\_id

AND edh1.start\_date = (SELECT MIN(edh2.start\_date)

FROM employee\_department\_history edh2

WHERE edh2.employee\_id = e.employee\_id)

LEFT JOIN

employee\_department\_history edh3 ON e.employee\_id = edh3.employee\_id

LEFT JOIN

departments current\_dept ON edh3.department\_id = current\_dept.department\_id

AND edh3.end\_date IS NULL

GROUP BY

e.employee\_id, e.employee\_name, first\_dept.department\_name, current\_dept.department\_name

ORDER BY

e.employee\_name;

1. **Exemplo1.parquet com 7GB**

Dentre os procedimentos adotados para otimização de consulta e leitura, estão: pré-processamento do arquivo, envolvendo o particionamento (dividindo o arquivo Parquet em partes menores baseadas em colunas frequentemente consultadas ou em intervalos de tempo), compressão (como Snappy ou GZIP), armazenamento e indexação (criação de índices e armazenamento em cache) e otimização da API (carregamento dos dados sob demanda, uso de armazenamento local, e uso de bibliotecas como pyarrow ou fastparquet, que são mais eficientes do que algumas soluções padrão).

Além disso, é necessário pensar na atualização do arquivo, agendando os processos incrementais programando as atualizações para momentos de baixa demanda, garantindo que a API continue respondendo rapidamente enquanto o arquivo é atualizado. Por fim, é imperativo, sempre manter o monitoramento de desempenho, assim como os testes e se atentar a escalabilidade do modelo.

1. **Otimização de modelos de regressão**

A otimização iria depende da natureza do modelo, mas formas gerais de otimizá-lo podem incluir: usar técnicas como “Grid Search” ou “Random Search” para encontrar os melhores hiperparâmetros (e.g., *eta, max\_depth, nrounds, gamma*) para o modelo; criar features a partir das existentes para melhorar a performance do modelo; implementar a técnica de validação cruzada k-fold; adicionar mais dados de treinamento para melhorar a robustez do modelo; e, por fim, experimentar com diferentes tipos de modelos, como Random Forest, Gradient Boosting, Bagging etc. Além do desempenho computacional e escalabilidade do modelo, os parâmetros de erro também devem ser avaliados na escolha do melhor modelo de otimização, tais como: CV (Coeficiente de Variação), ME (Erro Médio), RMSE (Raiz do Erro Quadrático Médio), MAPE (Erro Percentual Absoluto Médio), MAE (Erro Absoluto Médio) e MPE (Erro Percentual Médio). Uma forma de utilizar essas métricas de erro para avaliar o melhor modelo seria usar uma matriz de pesos e pontuação agregada, onde os pesos seriam inversamente proporcionais ao CV, ME, RMSE, MAPE, MAE e MPE.

Nesse caso, os pesos seriam normalizados para garantir que a soma de todos os pesos fosse igual a 1. Isso seria feito dividindo cada peso pelo somatório de todos os pesos. Para o cálculo da pontuação agregada de cada modelo, seria necessário somar o produto de cada métrica pelo peso correspondente, assim, as pontuações agregadas seriam normalizadas para obter os pesos finais, dividindo cada pontuação agregada pelo somatório de todas as pontuações agregadas.

1. **Sobre esses procedimentos (ELT e ETL) , é correto afirmar que:**

Estrutura de dados em nuvem são ideais para a adoção de estratégias ELT, devido a maior rapidez no carregamento dos dados e a adequada capacidade de processamento posterior.