EXERCÍCIO 1

Para fazer a inserção dos dados, farei em Python, pois estou estudando para aprimorar e tentando colocar em prática com cases reais.

EXERCÍCIO 1.1

Python

import os #primeiro passo, criar arquivo root, caso não exista

def criar\_est\_paises(paises):

root = "root"

if not os.path.exists(root):

os.mkdir(root)

#existirá um diretório para cada país da lista

for pais\_origem in paises:

origem\_path = os.path.join(root, pais\_origem)

if not os.path.exists(origem\_path):

os.mkdir(origem\_path)

#verifica a lista de destino e cria diretórios, combinando destino e origem

for pais\_destino in paises:

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

destino\_path = os.path.join(origem\_path, f"{pais\_destino}.txt")

if not os.path.exists(destino\_path):

with open(destino\_path, "w") as f:

f.write("") # Cria o arquivo vazio

#só continua se país de destino e origem não forem iguais

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

if pais\_origem == pais\_destino or pais\_destino == "WLD":

print("Consulta inválida")

return

root = "root"

file\_path = os.path.join(root, pais\_origem, f"{pais\_destino}.txt")

#o arquivo tem adicionado informações de ano e dados

with open(file\_path, "a") as f:

f.write(f"Ano: {ano}, Dados: {dados}\n")

# Lista de países

paises = ["BRA", "ARG", "CHN", "WLD", "AFG", "ZAF" , "ALA", "ALB", "DEU", "AND", "AGO", "AIA", "ATA", "ATG", "SAU", "DZA", "ARM", "ABW", "AUS", "AUT", "AZE", "BHS", "BHR", "BGD", "BRB", "BEL", "BLZ", "BEN", "BMU", "BLR", "BTN", "BOL", "BIH", "BAW", "BVT", "BRN", "BGR", "BFA", "BDI", "KHM", "CMR", "CAN", "CPV", "CYN", "KAZ", "TCD", "CHL", "CYP", "CXR", "CCK", "COL", "COM", "COG", "COK", "PRK","KOR", "CRI","CIV", "HRV", "CUB", "CUW", "DNK", "DJI", "DMA", "ECU", "EGY", "SLV", "ARE", "ERI", "SCO", "SVK", "SVN", "ESP", "USA", "UMI", "EST","ETH", "FRO", "FJI", "PHL", "FIN", "FRA", "ATF", "GAB", "GHA", "GMB", "GEO", "SGS", "GIB", "GRD", "GRC", "GRL", "GLP", "GUM", "GTM", "GGY", "GIN", "GNB", "GNQ", "GUY", "GUF", "HTI", "HMD", "HND", "HKG", "HUN", "YEM", "IND", "ENG", "IRN", "IRQ", "IRL", "NIR", "IMN", "ISL", "ISR", "ITA", "JAM", "JPN", "JEY", "JOR", "KIR", "KOS", "KWT", "LAO", "LVA", "LSO", "LBN", "LBR", "LBY", "LIE", "LTU", "LUX", "MAC", "MKD", "MDG", "MYS", "MWI", "MDV", "MLI", "MLT", "FLK", "MHL", "MNP", "MAR", "MTQ", "MRT", "MUS", "MYT", "MEX", "MMR", "FSM", "MDA", "MCO", "MNG", "MNE", "MSR", "MOZ", "NAM", "NRU", "NPL", "NCL", "NZL", "NIC", "NER", "NGA", "NIU", "NFK", "NOR", "OMN", "WAL", "NLD", "PLW", "PSE", "PAN", "PNG", "PAK", "PRY", "PER", "PCN", "PYF", "POL", "PRT", "PRI", "QAT", "KEN", "KGZ", "GBR", "IOT", "VGB", "CAF", "CZE", "COD", "DOM", "TWN", "REU", "ROU", "RUS", "RWA", "ESH", "SLB", "WSM", "ASM", "BLM", "SHN", "LCA", "MAF", "KNA", "SPM", "VCT", "SMR", "STP", "SEN", "SLE", "SRB", "SYC", "SGP", "SYR", "SOM", "LKA", "SDN", "SSD", "SWE", "SWZ", "CHE", "SUR", "SJM", "TJK", "THA", "TZA", "TLS", "TGO", "TKL", "TON", "TTO", "TUN","TKM", "TUR", "TCA", "TUV", "UKR", "UGA", "VIR", "URY", "UZB", "VUT", "VAT", "VEN","VNM", "WLF", "ZMB", "ZWE"]

# Cria a estrutura de diretórios e arquivos

criar\_estrutura\_paises(paises)

# Adicionando dados entre todos os países da lista

for pais\_origem in paises:

for pais\_destino in paises:

if pais\_origem != pais\_destino :

ano = random.randint(2000, 2023) # Gera um ano aleatório entre 2000 e 2023

adicionar\_dados(pais\_origem, pais\_destino, ano, f"Dados comerciais de {pais\_origem} para {pais\_destino} em {ano}")

EXERCÍCIO 1.2

import os

import random

# Função para criar a estrutura de diretórios e arquivos

def criar\_estrutura\_paises(paises, imprimir\_estrutura=False):

root = "root"

if not os.path.exists(root):

os.mkdir(root)

# Cria um diretório para cada país na lista

for pais\_origem in paises:

origem\_path = os.path.join(root, pais\_origem)

if not os.path.exists(origem\_path):

os.mkdir(origem\_path)

# Verifica a lista de destino e cria arquivos de destino

for pais\_destino in paises:

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

destino\_path = os.path.join(origem\_path, f"{pais\_destino}.txt")

if not os.path.exists(destino\_path):

with open(destino\_path, "w") as f:

f.write("") # Cria o arquivo vazio

if imprimir\_estrutura:

print(f"{pais\_origem}/{pais\_destino}.txt")

# Função para adicionar dados

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

if pais\_origem == pais\_destino or pais\_destino == "WLD":

print(f"Consulta inválida para {pais\_origem} -> {pais\_destino}")

return

root = "root"

file\_path = os.path.join(root, pais\_origem, f"{pais\_destino}.txt")

# Adiciona informações de ano e dados ao arquivo

with open(file\_path, "a") as f:

f.write(f"Ano: {ano}, Dados: {dados}\n")

# Lista de países

paises = ["BRA", "ARG", "CHN", "WLD", "AFG", "ZAF", "ALA", "ALB", "DEU", "AND", "AGO", "AIA", "ATA", "ATG", "SAU", "DZA", "ARM", "ABW", "AUS", "AUT", "AZE", "BHS", "BHR", "BGD", "BRB", "BEL", "BLZ", "BEN", "BMU", "BLR", "BTN", "BOL", "BIH", "BAW", "BVT", "BRN", "BGR", "BFA", "BDI", "KHM", "CMR", "CAN", "CPV", "CYN", "KAZ", "TCD", "CHL", "CYP", "CXR", "CCK", "COL", "COM", "COG", "COK", "PRK","KOR", "CRI","CIV", "HRV", "CUB", "CUW", "DNK", "DJI", "DMA", "ECU", "EGY", "SLV", "ARE", "ERI", "SCO", "SVK", "SVN", "ESP", "USA", "UMI", "EST","ETH", "FRO", "FJI", "PHL", "FIN", "FRA", "ATF", "GAB", "GHA", "GMB", "GEO", "SGS", "GIB", "GRD", "GRC", "GRL", "GLP", "GUM", "GTM", "GGY", "GIN", "GNB", "GNQ", "GUY", "GUF", "HTI", "HMD", "HND", "HKG", "HUN", "YEM", "IND", "ENG", "IRN", "IRQ", "IRL", "NIR", "IMN", "ISL", "ISR", "ITA", "JAM", "JPN", "JEY", "JOR", "KIR", "KOS", "KWT", "LAO", "LVA", "LSO", "LBN",

"LBR", "LBY", "LIE", "LTU", "LUX", "MAC", "MKD", "MDG", "MYS", "MWI", "MDV", "MLI", "MLT", "FLK", "MHL", "MNP", "MAR", "MTQ", "MRT", "MUS", "MYT", "MEX", "MMR", "FSM", "MDA", "MCO", "MNG", "MNE", "MSR", "MOZ", "NAM", "NRU", "NPL", "NCL", "NZL", "NIC", "NER", "NGA", "NIU", "NFK", "NOR", "OMN", "WAL", "NLD", "PLW", "PSE", "PAN", "PNG", "PAK", "PRY", "PER", "PCN", "PYF", "POL", "PRT", "PRI", "QAT", "KEN", "KGZ", "GBR", "IOT", "VGB", "CAF", "CZE", "COD", "DOM", "TWN", "REU", "ROU", "RUS", "RWA", "ESH", "SLB", "WSM", "ASM", "BLM", "SHN", "LCA", "MAF", "KNA", "SPM", "VCT", "SMR", "STP", "SEN", "SLE", "SRB", "SYC", "SGP", "SYR", "SOM", "LKA", "SDN", "SSD", "SWE", "SWZ", "CHE", "SUR", "SJM", "TJK", "THA", "TZA", "TLS", "TGO", "TKL", "TON", "TTO", "TUN","TKM", "TUR", "TCA", "TUV", “UKR", "UGA", "VIR", "URY", "UZB", "VUT", "VAT", "VEN","VNM", "WLF", "ZMB", "ZWE"]

# Cria a estrutura de diretórios e arquivos

criar\_estrutura\_paises(paises, imprimir\_estrutura=True)

# Adicionando dados entre todos os países da lista

for pais\_origem in paises:

for pais\_destino in paises:

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

ano = random.randint(2000, 2023) # Gera um ano aleatório entre 2000 e 2023

adicionar\_dados(pais\_origem, pais\_destino, ano, f"Dados comerciais de {pais\_origem} para {pais\_destino} em {ano}")

EXERCÍCIO 1.3

import os

# Função para encontrar e imprimir a quantidade de soja importada pelo Mundo do Brasil em 2020

def encontrar\_dado\_soja():

root = "root"

pais\_origem = "WLD"

pais\_destino = "BRA"

ano = 2020

file\_path = os.path.join(root, pais\_origem, f"{pais\_destino}.txt")

if os.path.exists(file\_path):

with open(file\_path, "r") as f:

for line in f:

if f"Ano: {ano}" in line and "soja" in line.lower():

print(f"Dado encontrado: {line.strip()}")

return line.strip()

print("Dados não encontrados.")

return None

# Chamar a função para encontrar e imprimir o dado desejado

encontrar\_dado\_soja()

EXERCÍCIO 1.4

1- Primeiramente verificar a estrutura da base de dados para que os países tenham diretórios mensais e anuais

2- montar um cronograma para a atualização e implementar no script dentro de um loop

3- fazer a coleta dos dados no inicio de cada mês atualizando no diretório como exemplo root/BRA/CHN\_2025\_M.txt

4- Da mesma forma, após 12 meses, coletar anualmente os dados, por exemplo root/BRA/CHN\_2024\_A.txt

EXERCÍCIO 2

1- Configurar o ambiente para fazer os requests e criar agendamentos para que isso seja feito de tempos em tempos usando função schedule

2- definir o caminho da URL que precisa ser acessada: salvar caminho numa variável

3- criar uma função para realizar o download dos dados usando get e verificar a integridade dos dados para evitar arquivo corrompido

4- gerar uma função para que a partir dos dados obtidos, extrair, ler e empilhar eles

EXERCÍCIO 3

WITH FirstDepartment AS (

SELECT

edh.employee\_id,

edh.department\_id,

MIN(edh.start\_date) AS first\_start\_date

FROM

employee\_department\_history edh

GROUP BY

edh.employee\_id,

edh.department\_id

QUALIFY

ROW\_NUMBER() OVER (PARTITION BY edh.employee\_id ORDER BY MIN(edh.start\_date)) = 1

),

CurrentDepartment AS (

SELECT

edh.employee\_id,

edh.department\_id

FROM

employee\_department\_history edh

WHERE

edh.end\_date IS NULL

),

DepartmentCount AS (

SELECT

edh.employee\_id,

COUNT(DISTINCT edh.department\_id) AS dept\_count

FROM

employee\_department\_history edh

GROUP BY

edh.employee\_id

)

SELECT

e.employee\_name,

dep1.department\_name AS first\_department,

dep2.department\_name AS current\_department,

dc.dept\_count

FROM

employees e

JOIN FirstDepartment fd ON e.employee\_id = fd.employee\_id

JOIN departments dep1 ON fd.department\_id = dep1.department\_id

JOIN CurrentDepartment cd ON e.employee\_id = cd.employee\_id

JOIN departments dep2 ON cd.department\_id = dep2.department\_id

JOIN DepartmentCount dc ON e.employee\_id = dc.employee\_id

ORDER BY

e.employee\_name;

Descrição

- Encontrar o departamento do funcionário, filtrando por data de início

- Encontrar o departamento do funcionário filtrando por data de término

- Contar número de departamentos distintos onde cada funcionário passou

- Fazer join das tabelas para acessar todas as informações

EXERCÍCIO 4

Algumas ações que podem ser feitas

- Criar indices para facilitar a busca por linhas especificas

- Particionar o arquivo em arquivos menos, como por exemplo por mês

- Armazenar em caches resultados de consultas frequentes da API

EXERCÍCIO 5

Para melhorar o resultado do modelo, algumas coisas podem ser feitas:

1. Verificar se o modelo escolhido é o melhor para os dados;
2. Verificar se as variaveis relevantes estão sendo utilizadas e as não relevantes foram removidas
3. Verificar se as transformações necessárias para as variáveis foram feitas
4. Verificar a homocedasticidade, normalidade de residuos e autocorrelação dos residuos
5. Verificar se possíveis outliers podem estar causando o resultado obtido

EXERCÍCIO 6

Alternativa d. Pois ao se trabalhar com ETL, utilizar a nuvem facilita a rapidez de acesso, facilidade de armazenamento e sua transformação