Criação das tabelas no PostgreSQL e imputação das informações do Banco de Dados

July 18, 2021

Trabalho: Projeto Integrador 2

Discentes: Bruna Mattioli de Oliveira e Gabriel Andrade Varga

Descrição: Criação do Banco de Dados no PostgreSQL e imputação dos dados oriundos de um arquivo CSV diretamente no Banco de Dados

1 Pacotes Necessários

[1]:
```
import numpy as np
import pandas as pd
import psycopg2
import csv
import pandas.io.sql as sqlio
from matplotlib import pyplot as plt
from matplotlib import ticker as ticker
import locale
locale.setlocale(locale.LC_NUMERIC, "pt_BR.UTF-8")
```

[1]:
```
'pt_BR.UTF-8'
```

2 Importação do Banco de Dados em CSV e Criação das Tabelas no BD

[2]:
```
# Importação do conjunto de dados completo contendo as informações de␣
 ↪desmatamento em CSV
desmatamento = pd.read_csv("G:/My Drive/Especialização/Disciplinas/Módulo 2/
 ↪Projeto Integrador 2/Bases/DesmatamentoMunicipios.csv", sep=';')

# Conexão no postgre com o banco de dados Projeto_Integrador_2
conn = psycopg2.connect("dbname=Projeto_Integrador_2 user=postgres␣
 ↪password=vagan9ch")
cur = conn.cursor()

# Criação das tabelas modeladas no Postgre
cur.execute("CREATE TABLE uf (id_uf SERIAL PRIMARY KEY, sigla_uf VARCHAR);")
```

```
cur.execute("CREATE TABLE municipio (id_municipio SERIAL PRIMARY KEY,␣
 ↪nome_municipio VARCHAR, id_uf INTEGER);")
cur.execute("CREATE TABLE desmatamento (id_desmatamento SERIAL PRIMARY KEY,␣
 ↪km_hidrografia NUMERIC(30,2), km_area_total NUMERIC(30,2), km_desmatado␣
 ↪NUMERIC(30,2), km_n_floresta NUMERIC(30,2), km_nuvem NUMERIC(30,2), km_n_obs␣
 ↪NUMERIC(30,2), km_floresta NUMERIC(30,2), id_municipio INTEGER, id_ano␣
 ↪INTEGER);")
cur.execute("CREATE TABLE ano (id_ano SERIAL PRIMARY KEY, ano INTEGER);")

# Atribuição das chaves estrangeiras
cur.execute("ALTER TABLE municipio ADD FOREIGN KEY (id_uf) REFERENCES uf␣
 ↪(id_uf);")
cur.execute("ALTER TABLE desmatamento ADD FOREIGN KEY (id_municipio) REFERENCES␣
 ↪municipio (id_municipio);")
cur.execute("ALTER TABLE desmatamento ADD FOREIGN KEY (id_ano) REFERENCES ano␣
 ↪(id_ano);")

# Execução das atualizações
conn.commit()
```

3 Imputação dos Dados no PostgreSQL

[3]:
```
#-----------------------------------------------------
# Imputando os dados da tabela / entidade UF no BD
#-----------------------------------------------------

# Filtro do campo necessário
uf = desmatamento[['Estado']]

# Retirada dos valores duplicados. Não é necessário ter os dados repetidos no␣
 ↪BD.
uf2 = uf.drop_duplicates(subset = 'Estado')

# Criação de uma lista com os dados da coluna
sigla_uf = uf2['Estado'].tolist()

#Criação de uma lista de dicionários contendo os dados da coluna estado e com a␣
 ↪chave sigla_uf
uf_dicionario = {}
uf_lista = []

for i in range(0, len(sigla_uf)):

    uf_dicionario[i] = dict({'sigla_uf':sigla_uf[i]})

    uf_lista.append(uf_dicionario[i])
```

```
# Imputação dos dados na tabela do BD estado
cur.executemany("""INSERT INTO uf (sigla_uf) VALUES (%(sigla_uf)s);""",␣
 ↪uf_lista)

# Execução das atualizações
conn.commit()
```

[4]:
```
#-----------------------------------------------------
# Imputando os dados da tabela / entidade ano no BD
#-----------------------------------------------------

# Filtro do campo necessário
ano = desmatamento[['Ano']]

# Retirada dos valores duplicados. Não é necessário ter os dados repetidos no␣
 ↪BD.
ano2 = ano.drop_duplicates(subset = 'Ano')

# Criação de uma lista com os dados da coluna
coluna_ano = ano2['Ano'].tolist()

#Criação de uma lista de dicionários contendo os dados da coluna estado e com a␣
 ↪chave sigla_uf
ano_dicionario = {}
ano_lista = []

for i in range(0, len(coluna_ano)):

    ano_dicionario[i] = dict({'ano':coluna_ano[i]})

    ano_lista.append(ano_dicionario[i])

# Imputação dos dados na tabela do BD estado
cur.executemany("""INSERT INTO ano (ano) VALUES (%(ano)s);""", ano_lista)

# Execução das atualizações
conn.commit()
```

[5]:
```
#-----------------------------------------------------------
# Imputando os dados da tabela / entidade municipio no BD
#-----------------------------------------------------------

# Imputando os dados via um looping no CSV linha por linha
with open('G:/My Drive/Especialização/Disciplinas/Módulo 2/Projeto Integrador 2/
 ↪Bases/Lista_Municipios.csv',newline='', encoding='utf8', errors='ignore') as␣
 ↪csvfile:
```

```
    reader = csv.DictReader(csvfile, delimiter=';')
    for row in reader:
        if row['Estado'] == 'PA' or row['Estado'] == 'MA' or row['Estado'] ==␣
↪'MT' or row['Estado'] == 'RO' or row['Estado'] == 'TO' or row['Estado'] ==␣
↪'AC' or row['Estado'] == 'AM' or row['Estado'] == 'RR' or row['Estado'] ==␣
↪'AP':
            id_uf = 0
            query = "SELECT id_uf FROM uf WHERE sigla_uf = '%s'"
            var1 = row['Estado']
            cur.execute(query % var1)
            id_uf = cur.fetchone()
            insert = "INSERT INTO municipio(nome_municipio, id_uf) VALUES␣
↪(%s,%s)"
            parametros = (row['Municipio'], id_uf[0])
            cur.execute(insert, parametros)

# Execução das atualizações
conn.commit()
```

[6]:
```
#-----------------------------------------------------------------
# Imputando os dados da tabela / entidade desmatamento no BD
#-----------------------------------------------------------------

# Imputando os dados via um looping no CSV linha por linha
with open('G:/My Drive/Especialização/Disciplinas/Módulo 2/Projeto Integrador 2/
↪Bases/DesmatamentoMunicipios.csv',newline='', encoding='utf8',␣
↪errors='ignore') as csvfile:
    reader = csv.DictReader(csvfile, delimiter=';')
    for row in reader:
        if row['Estado'] == 'PA' or row['Estado'] == 'MA' or row['Estado'] ==␣
↪'MT' or row['Estado'] == 'RO' or row['Estado'] == 'TO' or row['Estado'] ==␣
↪'AC' or row['Estado'] == 'AM' or row['Estado'] == 'RR' or row['Estado'] ==␣
↪'AP':
            id_municipio = 0
            id_ano = 0
            query1 = "SELECT id_municipio FROM municipio WHERE nome_municipio =␣
↪'%s'"
            var1 = row['Municipio']
            query2 = "SELECT id_ano FROM ano WHERE ano = '%s'"
            var2 = row['Ano']
            cur.execute(query1 % var1)
            id_municipio = cur.fetchone()
            cur.execute(query2 % var2)
            id_ano = cur.fetchone()
            insert = "INSERT INTO desmatamento(km_hidrografia, km_area_total,␣
↪km_desmatado, km_n_floresta, km_nuvem, km_n_obs, km_floresta, id_municipio,␣
↪id_ano) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)"
```

```
            parametros = (row['Hidrografia'], row['AreaKm2'], row['Desmatado'],␣
 ↪row['NaoFloresta'], row['Nuvem'], row['NaoObservado'], row['Floresta'],␣
 ↪id_municipio[0], id_ano[0])
            cur.execute(insert, parametros)

# Execução das atualizações
conn.commit()
```