Conformidade - Grupo 6

October 26, 2021

1 Trabalho de Conformidade do Curso de Dados Abertos para Controle Social

O objetivo do presente trabalho é verificar quais empresas de pessoas politicamente expostas que receberam recursos dos próprios municípios em que têm mandatos.

Para isso serão utilizado dados do Portal de Transparência Municipal do TCESP, base de dados públicos de CNPJ da SRF e dados do Portal da Transparência da CGU.

Atenção: Como o presente trabalho é para fins educacionais, os nomes, CPFs e CNPJs foram alterados, mas as estruturas são idênticas às das tabelas originais.

1.1 Importar as bibliotecas utilizadas

[1]:
```
import os
import shutil

import pandas as pd
import sqlalchemy
```

1.2 Carregar Dados

[2]:
```
diretorio_dados = 'dados'
```

1.2.1 Carregar dados de banco

O banco de dados possui três tabelas: despesas_consolidado, empresas e sócios

[3]:
```
arquivo_sqlite = 'banco_trabalho06.db'
filepath_arquivo_sqlite = os.path.join(diretorio_dados, arquivo_sqlite)
```

[4]:
```
url_banco = "sqlite:///{}".format(filepath_arquivo_sqlite)
engine = sqlalchemy.create_engine(url_banco)
```

Despesas É a totalização de valores pagos a empresas por município, valores aproximadamente acima de 100 mil reais.

A consolidação foi realizada a partir da tabela despesas do TCE/SP. Os nomes de empresas, cnpjs e municípios foram alterados.

A coluna **ds__municipio** é o nome do município fictício, **nr__identificador__despesa** o CNPJ da empresa que recebeu o valor da coluna soma e **ds__despesa** a razão social fictícia da empresa.

[5]:
```
sql = 'SELECT * FROM despesas_consolidado'
despesas = pd.read_sql(sql, engine, index_col=None)
```

Aqui podemos visualizar uma amostra dos dados

[6]:
```
despesas.sample(5)
```

[6]:
```
      ds_municipio nr_identificador_despesa              ds_despesa  \
16980  KINDZIERSKI           26318430000189           ALEIX CALCADOS
10288      ROZARIO           26074393000172  GIOVANNI CONSTRUCOES SA
20817      FELIPPE           37373134000133      MADISON TRADING EPP
22332     ANZILAGO           46057911000196          IZAN ESPORTIVOS
6718       ZIEMATH           57686073000152       SOPHIA CONTABIL SA

            soma
16980  189032.64
10288  382607.19
20817  133119.31
22332  123720.51
6718   548056.01
```

Com o comando abaixo, podemos ver que a tabela possui 25.885 registros; não possui dados nulos e há apenas uma coluna com dados numéricos.

[7]:
```
despesas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25885 entries, 0 to 25884
Data columns (total 4 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ds_municipio             25885 non-null  object
 1   nr_identificador_despesa  25885 non-null  object
 2   ds_despesa               25885 non-null  object
 3   soma                     25885 non-null  float64
dtypes: float64(1), object(3)
memory usage: 809.0+ KB
```

Empresas Tabela de empresas da receita federal.

Possui dados alterados, mas a estrutura das colunas é semelhante a base pública da Receita Federal.

[8]:
```
sql = 'SELECT * FROM empresas'
empresas = pd.read_sql(sql, engine, index_col=None)
```

Aqui podemos visualizar uma amostra dos dados

[9]:
```
empresas.sample(5)
```

[9]:
```
                cnpj matriz_filial            razao_social nome_fantasia  \
4580  09886216000298             2         RIKO COMERCIO ME
1688  08760737000242             2              KSENIA BENS
5472  09803104000173             1  BOTOND SUPERMERCADO EPP
5775  01390361000158             1         BALINT ALIMENTOS
3787  01744979000285             2           ARINA MADEIRAS

     situacao data_situacao motivo_situacao nm_cidade_exterior cod_pais  \
4580       02      20200101              01
1688       02      20200101              01
5472       02      20200101              63
5775       02      20200101              00
3787       02      20200101              37

     nome_pais  … email qualif_resp capital_social porte opc_simples  \
4580           …                              0.0    01           6
1688           …                              0.0    05           0
5472           …                              0.0    01           0
5775           …                              0.0    05           0
3787           …                              0.0    05           0

     data_opc_simples data_exc_simples opc_mei sit_especial data_sit_especial
4580
1688
5472                                        N
5775                                        N
3787

[5 rows x 38 columns]
```

Com o comando abaixo, podemos ver que a tabela possui 5.779 registros; não possui dados nulos e há apenas uma coluna com dados numéricos.

[10]:
```
empresas.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5779 entries, 0 to 5778
Data columns (total 38 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   cnpj                5779 non-null   object
```

```
 1   matriz_filial       5779 non-null   object
 2   razao_social        5779 non-null   object
 3   nome_fantasia       5779 non-null   object
 4   situacao            5779 non-null   object
 5   data_situacao       5779 non-null   object
 6   motivo_situacao     5779 non-null   object
 7   nm_cidade_exterior  5779 non-null   object
 8   cod_pais            5779 non-null   object
 9   nome_pais           5779 non-null   object
 10  cod_nat_juridica    5779 non-null   object
 11  data_inicio_ativ    5779 non-null   object
 12  cnae_fiscal         5779 non-null   object
 13  tipo_logradouro     5779 non-null   object
 14  logradouro          5779 non-null   object
 15  numero              5779 non-null   object
 16  complemento         5779 non-null   object
 17  bairro              5779 non-null   object
 18  cep                 5779 non-null   object
 19  uf                  5779 non-null   object
 20  cod_municipio       5779 non-null   object
 21  municipio           5779 non-null   object
 22  ddd_1               5779 non-null   object
 23  telefone_1          5779 non-null   object
 24  ddd_2               5779 non-null   object
 25  telefone_2          5779 non-null   object
 26  ddd_fax             5779 non-null   object
 27  num_fax             5779 non-null   object
 28  email               5779 non-null   object
 29  qualif_resp         5779 non-null   object
 30  capital_social      5779 non-null   float64
 31  porte               5779 non-null   object
 32  opc_simples         5779 non-null   object
 33  data_opc_simples    5779 non-null   object
 34  data_exc_simples    5779 non-null   object
 35  opc_mei             5779 non-null   object
 36  sit_especial        5779 non-null   object
 37  data_sit_especial   5779 non-null   object
dtypes: float64(1), object(37)
memory usage: 1.7+ MB
```

Sócios Tabela de sócios da receita federal.

Possui dados alterados, mas a estrutura das colunas é semelhante a base pública da Receita Federal.

[11]:
```
sql = 'SELECT * FROM socios'
socios = pd.read_sql(sql, engine, index_col=None)
```

Aqui podemos visualizar uma amostra dos dados

[12]:
```
socios.sample(5)
```

[12]:
```
                cnpj tipo_socio                              nome_socio  \
5030  27977304000107          2                          ZAPPAZ ANIENTO
2574  56776669000107          2                     IDEVAL DELU STRAPPA
2747  41290096000177          2       JOAO YANKOUS JOANNES DE MOLIMAROLI
3535  16688097000163          2                  JULIANO BENDOCCHI TAKEZI
4070  33226260000114          2  SANTISSIMO CAUME TIERSON SCHIEFERDECKER

     cnpj_cpf_socio cod_qualificacao  perc_capital data_entrada cod_pais_ext  \
5030    ***193438**               65           0.0     20200101
2574    ***078588**               16           0.0     20200101
2747    ***810538**               49           0.0     20200101
3535    ***109298**               49           0.0     20200101
4070    ***868328**               22           0.0     20200101

     nome_pais_ext cpf_repres nome_repres cod_qualif_repres
5030                                                    00
2574                                                    00
2747                                                    00
3535                                                    00
4070                                                    00
```

Com o comando abaixo, podemos ver que a tabela possui 5.825 registros; não possui dados nulos e há apenas uma coluna com dados numéricos.

[13]:
```
socios.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5825 entries, 0 to 5824
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   cnpj               5825 non-null   object
 1   tipo_socio         5825 non-null   object
 2   nome_socio         5825 non-null   object
 3   cnpj_cpf_socio     5825 non-null   object
 4   cod_qualificacao   5825 non-null   object
 5   perc_capital       5825 non-null   float64
 6   data_entrada       5825 non-null   object
 7   cod_pais_ext       5825 non-null   object
 8   nome_pais_ext      5825 non-null   object
 9   cpf_repres         5825 non-null   object
 10  nome_repres        5825 non-null   object
 11  cod_qualif_repres  5825 non-null   object
dtypes: float64(1), object(11)
memory usage: 546.2+ KB
```

1.2.2 Carregar dados de Prefeitos e Vereadores de São Paulo

[14]:
```
arquivo_peps = 'PEP_SP.csv'
filepath_arquivo_pep = os.path.join(diretorio_dados, arquivo_peps)
```

[15]:
```
peps = pd.read_csv(filepath_arquivo_pep, sep=';')
```

Aqui podemos visualizar uma amostra dos dados

[16]:
```
peps.sample(5)
```

[16]:

	CPF	Nome_PEP	Sigla_Função	\
4185	***.534.708-**	VANIA BRAZ DE STACHUCK QUINES	VEREAD	
7416	***.963.388-**	CAUME GERALDO PENUELA	PREFEI	
5619	***.727.458-**	JOAO UNAIDE OZILA	VEREAD	
981	***.123.328-**	SPERETA UNAIDE FIDELIA GONDOREK	PREFEI	
5143	***.665.818-**	EDIVALDO ZUBELZO	VEREAD	

	Descrição_Função	Nível_Função	Nome_Órgão	\
4185	VEREADOR	NaN	MUN. DE FEDRI-SP	
7416	PREFEITO	NaN	MUN. DE ZOCULO-SP	
5619	VEREADOR	NaN	MUN. DE LAVAQUIAL-SP	
981	PREFEITO	NaN	MUN. DE FELETO-SP	
5143	VEREADOR	NaN	MUN. DE TUSUTIYA-SP	

	Data_Início_Exercício	Data_Fim_Exercício	Data_Fim_Carência	uf_orgao
4185	01/01/2017	31/12/2020	31/12/2025	SP
7416	01/01/2017	31/12/2020	31/12/2025	SP
5619	01/01/2017	31/12/2020	31/12/2025	SP
981	01/01/2017	31/12/2020	31/12/2025	SP
5143	01/01/2017	31/12/2020	31/12/2025	SP

Com o comando abaixo, podemos ver que a tabela possui 7.690 registros; possui apenas o coluna **Nível_Função** com dados nulos.

[17]:
```
peps.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7690 entries, 0 to 7689
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CPF                   7690 non-null   object
 1   Nome_PEP              7690 non-null   object
 2   Sigla_Função          7690 non-null   object
 3   Descrição_Função      7690 non-null   object
 4   Nível_Função          0 non-null      float64
 5   Nome_Órgão            7690 non-null   object
```

```
 6   Data_Início_Exercício  7690 non-null   object
 7   Data_Fim_Exercício     7690 non-null   object
 8   Data_Fim_Carência      7690 non-null   object
 9   uf_orgao               7690 non-null   object
dtypes: float64(1), object(9)
memory usage: 600.9+ KB
```

1.3 Execução

[18]:
```
# Definir diretorio saida
diretorio_saida_dados = 'saida'

# Remover diretorio saida, se houver
shutil.rmtree(diretorio_saida_dados, ignore_errors=True)

# Criar diretorio saida
os.makedirs(diretorio_saida_dados, exist_ok=True)
```

1.3.1 Tratamento dos dados

Alguns dados precisam ser tratados antes da análise.

Empresa A coluna **data_situacao** está no formato YYYMMDD, para facilitar a análise, vamos criar a coluna **data_situacao_normalizado** com o formato YYYY-MM-DD

[19]:
```
empresas['data_situacao'].sample(5)
```

[19]:
```
2458    20200101
3353    20200101
4676    20200101
2766    20200101
766     20200101
Name: data_situacao, dtype: object
```

[20]:
```
empresas['data_situacao_normalizado'] = pd.
↪to_datetime(empresas['data_situacao']).dt.date
```

[21]:
```
empresas['data_situacao_normalizado'].sample(5)
```

[21]:
```
5046    2020-01-01
2450    2020-01-01
2646    2020-01-01
132     2020-01-01
2739    2020-01-01
Name: data_situacao_normalizado, dtype: object
```

Socios A coluna **data_entrada** está no formato YYYMMDD, para facilitar a análise, vamos criar a coluna **data_entrada_normalizado** com o formato YYYY-MM-DD

[22]:
```
socios['data_entrada'].sample(5)
```

[22]:
```
3668    20200101
2107    20200101
4108    20200101
3034    20200101
2152    20200101
Name: data_entrada, dtype: object
```

[23]:
```
socios['data_entrada_normalizado'] = pd.to_datetime(socios['data_entrada']).dt.
↪date
```

[24]:
```
socios['data_entrada_normalizado'].sample(5)
```

[24]:
```
2184    2020-01-01
195     2020-01-01
3245    2020-01-01
5027    2020-01-01
645     2020-01-01
Name: data_entrada_normalizado, dtype: object
```

1.3.2 Pessoas Politicamente Expostas

A coluna **CPF** possuem pontuação, vamos criar a coluna **CPF_normalizado** sem elas.

[25]:
```
peps['CPF'].sample(5)
```

[25]:
```
5989    ***.772.418-**
7529    ***.980.228-**
693     ***.088.278-**
2535    ***.310.278-**
6104    ***.786.918-**
Name: CPF, dtype: object
```

[26]:
```
peps['CPF_normalizado'] = peps['CPF'].str.replace(".", "", regex=False).str.
↪replace("-", "", regex=False)
```

[27]:
```
peps['CPF_normalizado'].sample(5)
```

[27]:
```
410     ***051178**
7214    ***937508**
534     ***065848**
7185    ***934308**
2831    ***344928**
Name: CPF_normalizado, dtype: object
```

Na coluna **Nome_Órgão** vamos filtrar dados extras e manter apenas o nome do município na coluna **Municipio_normalizado**

[28]:
```
peps['Nome_Órgão'].sample(5)
```

[28]:
```
5705    MUN. DE FANZERES-SP
6474     MUN. DE MANHANE-SP
4800    MUN. DE ESTRUTTI-SP
5081    MUN. DE FRANSANI-SP
2814     MUN. DE CHOMSKI-SP
Name: Nome_Órgão, dtype: object
```

[29]:
```
peps['Municipio_normalizado'] = peps['Nome_Órgão'].str.replace("MUN. DE ", "",␣
↪regex=False).str.replace("-SP", "", regex=False)
```

[30]:
```
peps['Municipio_normalizado'].sample(5)
```

[30]:
```
3236        MAYAN
3069    FUJICHIMA
5653       CINQUE
3972     PODEWILS
4762         TOME
Name: Municipio_normalizado, dtype: object
```

As coluna **Data_Início_Exercício**, **Data_Fim_Exercício** e **Data_Fim_Carência** estão no formato YYYY/MM/DD, para facilitar a análise, vamos criar a colunas **Data_Inicio_Exercicio_normalizado**, **Data_Fim_Exercicio_normalizado** e **Data_Fim_Carência_normalizado** com o formato YYYY-MM-DD

[31]:
```
peps['Data_Início_Exercício'].sample(5)
```

[31]:
```
361     01/01/2017
4650    01/01/2017
1611    01/01/2017
5866    01/01/2017
4463    01/01/2017
Name: Data_Início_Exercício, dtype: object
```

[32]:
```
peps['Data_Fim_Exercício'].sample(5)
```

[32]:
```
7654    31/12/2020
3565    31/12/2020
1815    31/12/2020
3147    31/12/2020
6603    31/12/2020
Name: Data_Fim_Exercício, dtype: object
```

[33]:
```
peps['Data_Fim_Carência'].sample(5)
```

[33]:
```
415     31/12/2025
5718    31/12/2025
197     31/12/2025
5534    31/12/2025
6562    31/12/2025
Name: Data_Fim_Carência, dtype: object
```

[34]:
```
peps['Data_Inicio_Exercicio_normalizado'] = pd.
 ↪to_datetime(peps['Data_Início_Exercício']).dt.date
peps['Data_Fim_Exercicio_normalizado'] = pd.
 ↪to_datetime(peps['Data_Fim_Exercício']).dt.date
peps['Data_Fim_Carência_normalizado'] = pd.
 ↪to_datetime(peps['Data_Fim_Carência']).dt.date
```

[35]:
```
peps['Data_Inicio_Exercicio_normalizado'].sample(5)
```

[35]:
```
7070    2017-01-01
3207    2017-01-01
6726    2017-01-01
14      2017-01-01
5940    2017-01-01
Name: Data_Inicio_Exercicio_normalizado, dtype: object
```

[36]:
```
peps['Data_Fim_Exercicio_normalizado'].sample(5)
```

[36]:
```
4527    2020-12-31
467     2020-12-31
1922    2020-12-31
6469    2020-12-31
665     2020-12-31
Name: Data_Fim_Exercicio_normalizado, dtype: object
```

[37]:
```
peps['Data_Fim_Carência_normalizado'].sample(5)
```

[37]:
```
818     2025-12-31
1685    2025-12-31
5123    2025-12-31
5610    2025-12-31
4088    2025-12-31
Name: Data_Fim_Carência_normalizado, dtype: object
```

1.3.3 Gerar Banco de Dados

Copiar banco original para diretorio de saida

[38]:
```
arquivo_banco_saida = 'output.db'
filepath_arquivo_banco_saida = os.path.join(diretorio_saida_dados,␣
 ↪arquivo_banco_saida)
```

```
shutil.copyfile(filepath_arquivo_sqlite, filepath_arquivo_banco_saida)
```

[38]:
```
'saida/output.db'
```

Gravar tabela de prefeitos_vereadores em banco gerado

[39]:
```
url_banco = "sqlite:///{}".format(filepath_arquivo_banco_saida)
engine_banco_saida = sqlalchemy.create_engine(url_banco)
```

[40]:
```
peps.to_sql('peps', con=engine_banco_saida)
```

1.3.4 Consultar Empresas Relacionadas com Pessoas Politicamente Expostas que Receberam Recursos

[41]:
```
sql = 'SELECT \
  des.ds_municipio AS [Nome_Municipio_Ficticio], \
  des.nr_identificador_despesa AS [CNPJ_Empresa_Beneficiaria], \
  des.ds_despesa AS [Razao_Social_Empresa_Beneficiaria], \
  des.soma AS [Valor_Total], \
  emp.cnpj AS [CNPJ_RFB], \
  emp.razao_social AS [Razao_Social_RFB], \
  CASE emp.situacao \
    WHEN "01" THEN "NULA" \
    WHEN "02" THEN "ATIVA" \
    WHEN "03" THEN "SUSPENSA" \
    WHEN "04" THEN "INAPTA" \
    WHEN "08" THEN "BAIXADA" \
  END AS [Situacao_Cadastral_RFB], \
  CAST(emp.data_situacao AS DATE) AS [Data_Situacao_Cadastral_RFB], \
  emp.cod_nat_juridica AS [Codigo_Natureza_Juridica_RFB], \
  CAST(emp.data_inicio_ativ AS DATE) AS [Data_Inicio_Atividade_RFB], \
  emp.cnae_fiscal AS [CNAE_Fiscal_RFB], \
  emp.municipio AS [Municipio_RFB], \
  emp.uf AS [UF_RFB], \
  soc.nome_socio AS [Nome_Socio_RFB], \
  soc.cnpj_cpf_socio AS [CPF_Socio_RFB], \
  CAST(soc.data_entrada AS DATE) AS [Data_Entrada_Socio_RFB], \
  pep.Nome_PEP AS [Nome_PEP], \
  pep.CPF_normalizado AS [CPF_PEP], \
  pep.Descrição_Função AS [Funcao_PEP], \
  pep.Data_Inicio_Exercicio_normalizado AS [Data_Inicio_PEP], \
  pep.Data_Fim_Exercicio_normalizado AS [Data_Fim_PEP], \
  pep.Data_Fim_Carência_normalizado AS [Data_Carencia_PEP], \
  pep.Municipio_normalizado AS [Municipio_PEP], \
  pep.uf_orgao AS [UF_PEP] \
FROM despesas_consolidado des \
JOIN empresas emp ON emp.cnpj = des.nr_identificador_despesa \
```

```
JOIN socios soc ON soc.cnpj = emp.cnpj \
JOIN peps pep ON pep.CPF_normalizado = soc.cnpj_cpf_socio AND pep.Nome_PEP =␣
↪soc.nome_socio'
```

[42]:
```
empresas_peps = pd.read_sql(sql, engine_banco_saida, index_col=None)
```

[43]:
```
empresas_peps.sample(5)
```

[43]:

```
    Nome_Municipio_Ficticio CNPJ_Empresa_Beneficiaria  \
4                    MARTEL            49057710000187
44                 NIELAND            06273922000124
66                 VIECILI            08262075000137
63                CAMPALDI            10240793000137
38                PODEWILS            05605785000141

   Razao_Social_Empresa_Beneficiaria  Valor_Total        CNPJ_RFB  \
4                     IZAN INDUSTRIAL   4777742.82  49057710000187
44            MOMOKA ARTISTICAS LTDA    195474.65  06273922000124
66           MARIUS CONTABIL EIRELLI    112998.16  08262075000137
63         JOAO LUCAS ELETRONICOS SA    122131.48  10240793000137
38          NIKITA PRESENTES EIRELLI    238680.05  05605785000141

            Razao_Social_RFB Situacao_Cadastral_RFB  \
4             IZAN INDUSTRIAL                ATIVA
44    MOMOKA ARTISTICAS LTDA                 ATIVA
66   MARIUS CONTABIL EIRELLI                BAIXADA
63  JOAO LUCAS ELETRONICOS SA                ATIVA
38   NIKITA PRESENTES EIRELLI                ATIVA

    Data_Situacao_Cadastral_RFB Codigo_Natureza_Juridica_RFB  \
4                      20200101                         2143
44                     20200101                         3999
66                     20200101                         3999
63                     20200101                         3999
38                     20200101                         2062

    Data_Inicio_Atividade_RFB  … CPF_Socio_RFB Data_Entrada_Socio_RFB  \
4                    20200713  …   ***529608**               20200101
44                   20000214  …   ***923478**               20200101
66                   20030308  …   ***341758**               20200101
63                   20130111  …   ***502138**               20200101
38                   19970922  …   ***952898**               20200101

                                Nome_PEP      CPF_PEP Funcao_PEP  \
4                    RADANOVIC ASCENCIANO  ***529608**   VEREADOR
44           GEOVANISA TERSINHA BERTASSONI  ***923478**   PREFEITO
66                GOSSON LEGNANNE GONDOREK  ***341758**   PREFEITO
```

```
63  CAUME UNAIDE ABASILIA PHREDERICO ZIELAK  ***502138**   PREFEITO
38           ANDERSON PHREDERICO ZIELAK RAGO  ***952898**   VEREADOR

    Data_Inicio_PEP Data_Fim_PEP Data_Carencia_PEP Municipio_PEP UF_PEP
4        2017-01-01   2020-12-31        2025-12-31      MORANDIM     SP
44       2017-01-01   2020-12-31        2025-12-31        SACCOL     SP
66       2017-01-01   2020-12-31        2025-12-31       ROZARIO     SP
63       2017-01-01   2020-12-31        2025-12-31         BREIA     SP
38       2017-01-01   2020-12-31        2025-12-31     GONSALVES     SP

[5 rows x 24 columns]
```

Salvar dados

[44]:
```
arquivo_saida = 'empresas_peps.csv'
filepath = os.path.join(diretorio_saida_dados, arquivo_saida)
empresas_peps.to_csv(filepath, index=False)
```

1.3.5 Consultar Empresas Relacionadas com Pessoas Politicamente Expostas que Receberam Recursos dos Municípios que Possuem Mandatos

[45]:
```
sql = 'SELECT \
  des.ds_municipio AS [Nome_Municipio_Ficticio], \
  des.nr_identificador_despesa AS [CNPJ_Empresa_Beneficiaria], \
  des.ds_despesa AS [Razao_Social_Empresa_Beneficiaria], \
  des.soma AS [Valor_Total], \
  emp.cnpj AS [CNPJ_RFB], \
  emp.razao_social AS [Razao_Social_RFB], \
  CASE emp.situacao \
    WHEN "01" THEN "NULA" \
    WHEN "02" THEN "ATIVA" \
    WHEN "03" THEN "SUSPENSA" \
    WHEN "04" THEN "INAPTA" \
    WHEN "08" THEN "BAIXADA" \
  END AS [Situacao_Cadastral_RFB], \
  CAST(emp.data_situacao AS DATE) AS [Data_Situacao_Cadastral_RFB], \
  emp.cod_nat_juridica AS [Codigo_Natureza_Juridica_RFB], \
  CAST(emp.data_inicio_ativ AS DATE) AS [Data_Inicio_Atividade_RFB], \
  emp.cnae_fiscal AS [CNAE_Fiscal_RFB], \
  emp.municipio AS [Municipio_RFB], \
  emp.uf AS [UF_RFB], \
  soc.nome_socio AS [Nome_Socio_RFB], \
  soc.cnpj_cpf_socio AS [CPF_Socio_RFB], \
  CAST(soc.data_entrada AS DATE) AS [Data_Entrada_Socio_RFB], \
  pep.Nome_PEP AS [Nome_PEP], \
  pep.CPF_normalizado AS [CPF_PEP], \
  pep.Descrição_Função AS [Funcao_PEP], \
```

```
  pep.Data_Inicio_Exercicio_normalizado AS [Data_Inicio_PEP], \
  pep.Data_Fim_Exercicio_normalizado AS [Data_Fim_PEP], \
  pep.Data_Fim_Carência_normalizado AS [Data_Carencia_PEP], \
  pep.Municipio_normalizado AS [Municipio_PEP], \
  pep.uf_orgao AS [UF_PEP] \
FROM despesas_consolidado des \
JOIN empresas emp ON emp.cnpj = des.nr_identificador_despesa \
JOIN socios soc ON soc.cnpj = emp.cnpj \
JOIN peps pep ON pep.CPF_normalizado = soc.cnpj_cpf_socio AND pep.Nome_PEP =␣
 ↪soc.nome_socio \
WHERE emp.uf = pep.uf_orgao AND emp.municipio = pep.Municipio_normalizado'
```

[46]:
```
empresas_peps_mesmo_municipio = pd.read_sql(sql, engine_banco_saida,␣
 ↪index_col=None)
```

[47]:
```
empresas_peps_mesmo_municipio.sample(5)
```

[47]:

	Nome_Municipio_Ficticio	CNPJ_Empresa_Beneficiaria	Razao_Social_Empresa_Beneficiaria	Valor_Total	CNPJ_RFB	Razao_Social_RFB	Situacao_Cadastral_RFB	Data_Situacao_Cadastral_RFB	Codigo_Natureza_Juridica_RFB	Data_Inicio_Atividade_RFB	…	CPF_Socio_RFB	Data_Entrada_Socio_RFB
16	BOTCHER	10240793000137	JOAO LUCAS ELETRONICOS SA	102755.90	10240793000137	JOAO LUCAS ELETRONICOS SA	ATIVA	20200101	3999	20130111	…	***502138**	20200101
15	MILLNITZ	12480500000115	VICENTE MODAS	131687.01	12480500000115	VICENTE MODAS	BAIXADA	20200101	2305				
5	PATRUCCO	12480500000115	VICENTE MODAS	306932.06	12480500000115	VICENTE MODAS	BAIXADA	20200101	2305				
12	TUMARKIN	10240793000137	JOAO LUCAS ELETRONICOS SA	140473.55	10240793000137	JOAO LUCAS ELETRONICOS SA	ATIVA	20200101	3999				
7	PANIS	10240793000137	JOAO LUCAS ELETRONICOS SA	231545.38	10240793000137	JOAO LUCAS ELETRONICOS SA	ATIVA	20200101	3999				

```
15                  20140218  …   ***178358**              20200101
5                   20140218  …   ***178358**              20200101
12                  20130111  …   ***502138**              20200101
7                   20130111  …   ***502138**              20200101

                                    Nome_PEP      CPF_PEP Funcao_PEP  \
16  CAUME UNAIDE ABASILIA PHREDERICO ZIELAK  ***502138**   PREFEITO
15                      JOAO YPEI DE KAARA  ***178358**   PREFEITO
5                       JOAO YPEI DE KAARA  ***178358**   PREFEITO
12  CAUME UNAIDE ABASILIA PHREDERICO ZIELAK  ***502138**   PREFEITO
7   CAUME UNAIDE ABASILIA PHREDERICO ZIELAK  ***502138**   PREFEITO

    Data_Inicio_PEP Data_Fim_PEP Data_Carencia_PEP Municipio_PEP UF_PEP
16       2017-01-01   2020-12-31        2025-12-31         BREIA     SP
15       2017-01-01   2020-12-31        2025-12-31        GAZETA     SP
5        2017-01-01   2020-12-31        2025-12-31        GAZETA     SP
12       2017-01-01   2020-12-31        2025-12-31         BREIA     SP
7        2017-01-01   2020-12-31        2025-12-31         BREIA     SP

[5 rows x 24 columns]
```

Salvar dados

[48]:
```
arquivo_saida = 'empresas_peps_mesmo_municipio.csv'
filepath = os.path.join(diretorio_saida_dados, arquivo_saida)
empresas_prefeitos_vereadores_mesmo_municipio.to_csv(filepath, index=False)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_1182898/99507733.py in <module>
      1 arquivo_saida = 'empresas_peps_mesmo_municipio.csv'
      2 filepath = os.path.join(diretorio_saida_dados, arquivo_saida)
----> 3 empresas_prefeitos_vereadores_mesmo_municipio.to_csv(filepath,␣
 ↪index=False)

NameError: name 'empresas_prefeitos_vereadores_mesmo_municipio' is not defined
```

[ ]: