Conformidade - Grupo 6

October 26, 2021

1 Trabalho de Conformidade do

2 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.

2.1 Importar as bibliotecas utilizadas

[1]:
```
import os
import shutil

import pandas as pd
import sqlalchemy
```

2.2 Carregar Dados

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

2.2.1 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	soma
11920	SACCOL	03829447000172	ERIC CONSULTORIA	270435.88
7724	BAROUCH	07242413000116	AYANE SEGUROS SA	520539.72
11778	LOTTO	11299478000143	ROSALIE MOVEIS	302115.20
16488	NAKIRI	27071953000108	CELIA SISTEMAS SA	206873.03
24834	VALLIS	01433461000133	HANA ENGENHARIA	98952.78

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	\
5283	06982866000104	1	HIMARI MARKETING ME		

```
3783  01791646000378             2  AYAKA ADMINISTRATIVOS ME
994   04536152000678             2   RIKUTO CONSULTORIA LTDA
2479  02145121000215             2        MARGARITA SERVICOS
109   34834117000310             2              URI SHOPPING

     situacao data_situacao motivo_situacao nm_cidade_exterior cod_pais  \
5283       02      20200101              00
3783       02      20200101              37
994        02      20200101              00
2479       02      20200101              37
109        02      20200101              00

     nome_pais  … email qualif_resp capital_social porte opc_simples  \
5283           …                    486768.124845    05           0
3783           …                         0.000000    05           0
994            …                         0.000000    05           0
2479           …                         0.000000    05           0
109            …                         0.000000    05           0

     data_opc_simples data_exc_simples opc_mei sit_especial data_sit_especial
5283                                        N
3783
994
2479
109

[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	\
3579	38830942000179	2	ESTANISLAO BONAVOGLIA ZIELAK	
811	15598532000187	2	SPERETA JUNOR SHUMURANI	
794	25946534000192	2	SPERETA SALARINE	
1954	03676077000120	2	ERIVALDO GHUERREN PRANTEL	
4067	12427749000247	2	SANTISSIMO GUSTAVO BEDETTE HAUACH	

```
     cnpj_cpf_socio cod_qualificacao  perc_capital data_entrada cod_pais_ext  \
3579    ***324688**               49           0.0     20200101
811     ***243038**               59           0.0     20200101
794     ***724528**               59           0.0     20200101
1954    ***789858**               28           0.0     20200101
4067    ***439728**               65           0.0     20200101

     nome_pais_ext cpf_repres nome_repres cod_qualif_repres
3579                                                    00
811                                                     00
794                                                     00
1954                                                    00
4067                                                    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
```

2.2.2 Carregar dados de Prefeitos e Vereadores de São Paulo

[14]:
```
arquivo_prefeitos_vereadores = 'PEP_SP.csv'
filepath_arquivo_prefeitos_vereadores = os.path.join(diretorio_dados,␣
 ↪arquivo_prefeitos_vereadores)
```

[15]:
```
prefeitos_vereadores = pd.read_csv(filepath_arquivo_prefeitos_vereadores, sep=';
↪')
```

Aqui podemos visualizar uma amostra dos dados

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

[16]:

	CPF	Nome_PEP
6705	***.871.128-**	VARJAL DE NIRISMAR LAINY
5506	***.712.385-**	JOAO DE MOLIMAROLI NETO
2144	***.264.288-**	JANETE PIRASSOLLI OSARTCHUK ACHINELIS ALVARIA
6893	***.896.878-**	OTAIR SALARINE
1384	***.170.228-**	CAMPEZINO DINEFAR DO MORISSAWA

	Sigla_Função	Descrição_Função	Nível_Função	Nome_Órgão
6705	VEREAD	VEREADOR	NaN	MUN. DE LEFFA-SP
5506	VEREAD	VEREADOR	NaN	MUN. DE APOLINARIO-SP
2144	VEREAD	VEREADOR	NaN	MUN. DE CHIECON-SP
6893	VEREAD	VEREADOR	NaN	MUN. DE HATIDA-SP
1384	VEREAD	VEREADOR	NaN	MUN. DE CAZAROTTI-SP

	Data_Início_Exercício	Data_Fim_Exercício	Data_Fim_Carência	uf_orgao
6705	01/01/2017	31/12/2020	31/12/2025	SP
5506	01/01/2017	31/12/2020	31/12/2025	SP
2144	01/01/2017	31/12/2020	31/12/2025	SP
6893	01/01/2017	31/12/2020	31/12/2025	SP
1384	01/01/2017	31/12/2020	31/12/2025	SP

Com o comando abaixo, podemos ver que a tabela possui 7.690 registros; não possui dados nulos e há apenas uma coluna com dados numéricos.

[17]:
```
prefeitos_vereadores.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
```

2.3 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)
```

2.3.1 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]:
```
4124    20200101
1576    20200101
1899    20200101
5653    20200101
4851    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]:
```
198     2020-01-01
2505    2020-01-01
39      2020-01-01
4035    2020-01-01
5469    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]:
```
362     20200101
2471    20200101
1218    20200101
4503    20200101
2060    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]:
```
2921    2020-01-01
4847    2020-01-01
2752    2020-01-01
4300    2020-01-01
5489    2020-01-01
Name: data_entrada_normalizado, dtype: object
```

2.3.2 Prefeitos e Vereadores de SP

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

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

[25]:
```
4767    ***.614.888-**
4744    ***.612.198-**
4713    ***.607.358-**
2667    ***.323.768-**
5606    ***.725.928-**
Name: CPF, dtype: object
```

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

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

[27]:
```
3464    ***435598**
5169    ***669348**
2935    ***357908**
1517    ***184928**
2169    ***267718**
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]:
```
prefeitos_vereadores['Nome_Órgão'].sample(5)
```

[28]:
```
6173    MUN. DE DALPONTE-SP
462      MUN. DE CAMPEAN-SP
3687    MUN. DE CENCIANI-SP
1914     MUN. DE CSILLAG-SP
6107    MUN. DE PAULIQUE-SP
Name: Nome_Órgão, dtype: object
```

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

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

[30]:
```
2850      SAO PAULO
1908      SAO PAULO
4859    CARVALHEIRA
3536       BURGADAO
3989       FRANSANI
Name: Municipio_normalizado, dtype: object
```

As coluna **Data_Início_Exercício**, **Data_Fim_Exercício** e **Data_Fim_Carência** estão no formato YYY/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]:
```
prefeitos_vereadores['Data_Início_Exercício'].sample(5)
```

[31]:
```
5471    01/01/2017
2834    01/01/2017
1326    01/01/2017
390     01/01/2017
4284    01/01/2017
Name: Data_Início_Exercício, dtype: object
```

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

[32]:
```
3356    31/12/2020
6301    31/12/2020
673     31/12/2020
2485    31/12/2020
2852    31/12/2020
Name: Data_Fim_Exercício, dtype: object
```

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

[33]:
```
2503    31/12/2025
6409    31/12/2025
```

```
6967    31/12/2025
4872    31/12/2025
2363    31/12/2025
Name: Data_Fim_Carência, dtype: object
```

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

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

[35]:
```
2493    2017-01-01
3578    2017-01-01
788     2017-01-01
1622    2017-01-01
652     2017-01-01
Name: Data_Inicio_Exercicio_normalizado, dtype: object
```

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

[36]:
```
6959    2020-12-31
5696    2020-12-31
6487    2020-12-31
7312    2020-12-31
6682    2020-12-31
Name: Data_Fim_Exercicio_normalizado, dtype: object
```

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

[37]:
```
2887    2025-12-31
5933    2025-12-31
1428    2025-12-31
3854    2025-12-31
1197    2025-12-31
Name: Data_Fim_Carência_normalizado, dtype: object
```

2.3.3 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]:
```
prefeitos_vereadores.to_sql('peps', con=engine_banco_saida)
```

2.3.4 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_prefeitos_vereadores = pd.read_sql(sql, engine_banco_saida,␣
↪index_col=None)
```

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

[43]:

	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	Nome_PEP	CPF_PEP	Funcao_PEP	Data_Inicio_PEP
33	VIGANICO	49524688000199	IBAI VESTUARIO	290768.45	49524688000199	IBAI VESTUARIO	ATIVA	20200101	2038	20201019	…	***271878**	20200101	JAIME BONAVOGLIA DA TEVEIRA	***271878**	PREFEITO	2017-01-01
71	ASSUB	49524688000199	IBAI VESTUARIO	113162.42	49524688000199	IBAI VESTUARIO	ATIVA	20200101	2038	20201019	…	***460278**	20200101	SAVIOLO LUYTEN SMERIERI	***460278**	PREFEITO	2017-01-01
17	HACKEL	04578574000175	SOMA TURISMO EPP	513335.60	04578574000175	SOMA TURISMO EPP	ATIVA	20200101	2062	19960117	…	***149208**	20200101	CLOZOE SPERETA DANIEL	***149208**	PREFEITO	2017-01-01
42	VEJAM	13753624000188	MOHAMED DECORACOES EPP	213973.22	13753624000188	MOHAMED DECORACOES EPP	ATIVA	20200101	2062	20080529	…	***323128**	20200101				
28	SITA	49524688000199	IBAI VESTUARIO	339078.90	49524688000199	IBAI VESTUARIO	ATIVA	20200101	2038	20201019	…	***460278**	20200101				

```
42      ZAPPAZ CAMINAGA EUTIMIA  ***323128**   VEREADOR       2017-01-01
28      SAVIOLO LUYTEN SMERIERI  ***460278**   PREFEITO       2017-01-01

   Data_Fim_PEP Data_Carencia_PEP Municipio_PEP UF_PEP
33   2020-12-31        2025-12-31         ASSUB     SP
71   2020-12-31        2025-12-31          SITA     SP
17   2020-12-31        2025-12-31        HACKEL     SP
42   2020-12-31        2025-12-31       TSUKUDA     SP
28   2020-12-31        2025-12-31          SITA     SP

[5 rows x 24 columns]
```

Salvar dados

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

2.3.5 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_prefeitos_vereadores_mesmo_municipio = pd.read_sql(sql,␣
 ↪engine_banco_saida, index_col=None)
```

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

[47]:

	Nome_Municipio_Ficticio	CNPJ_Empresa_Beneficiaria	\
1	KNOLL	18941225000172	
12	TUMARKIN	10240793000137	
4	HANDA	07815964000138	
6	ALVERNE	47817109002513	
3	HACKEL	04578574000175	

	Razao_Social_Empresa_Beneficiaria	Valor_Total	CNPJ_RFB	\
1	CSENGE CONFECCOES EIRELLI	555209.93	18941225000172	
12	JOAO LUCAS ELETRONICOS SA	140473.55	10240793000137	
4	AMINE LIMPEZA EPP	425778.13	07815964000138	
6	CLAIRE SUPERMERCADO	311980.65	47817109002513	
3	SOMA TURISMO EPP	513335.60	04578574000175	

	Razao_Social_RFB	Situacao_Cadastral_RFB	\
1	CSENGE CONFECCOES EIRELLI	ATIVA	
12	JOAO LUCAS ELETRONICOS SA	ATIVA	
4	AMINE LIMPEZA EPP	ATIVA	
6	CLAIRE SUPERMERCADO	ATIVA	
3	SOMA TURISMO EPP	ATIVA	

	Data_Situacao_Cadastral_RFB	Codigo_Natureza_Juridica_RFB	\
1	20200101	2232	
12	20200101	3999	
4	20200101	2305	
6	20200101	2062	
3	20200101	2062	

	Data_Inicio_Atividade_RFB	…	CPF_Socio_RFB	Data_Entrada_Socio_RFB	\
1	20120324	…	***969658**	20200101	

```
12                  20130111  …   ***502138**              20200101
4                   20200127  …   ***396958**              20200101
6                   20050723  …   ***199388**              20200101
3                   19960117  …   ***149208**              20200101

                                    Nome_PEP      CPF_PEP Funcao_PEP  \
1                      JUNOR RUITA CHIMANGO  ***969658**   VEREADOR
12  CAUME UNAIDE ABASILIA PHREDERICO ZIELAK  ***502138**   PREFEITO
4                       FABIO MUSSY DE BOYDE  ***396958**   VEREADOR
6                       UNAIDE ALBERTO DAMAR  ***199388**   VEREADOR
3                      CLOZOE SPERETA DANIEL  ***149208**   PREFEITO

    Data_Inicio_PEP Data_Fim_PEP Data_Carencia_PEP Municipio_PEP UF_PEP
1        2017-01-01   2020-12-31        2025-12-31         KNOLL     SP
12       2017-01-01   2020-12-31        2025-12-31         BREIA     SP
4        2017-01-01   2020-12-31        2025-12-31       VIJARVA     SP
6        2017-01-01   2020-12-31        2025-12-31       ALVERNE     SP
3        2017-01-01   2020-12-31        2025-12-31        HACKEL     SP

[5 rows x 24 columns]
```

Salvar dados

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

[ ]:
```

```