Trabalho de Conformidade do Curso de Dados Abertos para Controle Social

October 29, 2021

Responsáveis:

Alan Taranti

Antônio Carlos

Natan Falbo

Renato Canegusuco Akamine

Orientadores:

Marcelo L. Perucci

Ricardo Tomita

O objetivo do presente trabalho consistem verificar quais empresas de PEPs (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 Importar as bibliotecas utilizadas

[1]:
```
import os
import shutil

import folium
import pandas as pd
import sqlalchemy
import seaborn as sns
```

2 Carregar Dados

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

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)
```

2.1.1 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.

Os dados originais podem ser obtidos em: https://transparencia.tce.sp.gov.br/conjunto-de-dados

[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
6398	LUND	62593700000124	CARL EQUIPAMENTOS	585659.09
3552	TOPFER	50180419000153	LENNARD SEGUROS	1119101.62
11930	ZACCAI	27173839000153	GYULA AGROPECUARIA	319774.64
7474	BOTCHER	08100800000136	CLARA LOGISTICA	531647.89
24635	ROZARIO	08506128000199	BENETT SERVICOS	115093.61

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
```

Múnicipios Únicos

[8]:
```
municipios_sao_paulo = 645
quantidade_municipios_unicos = len(despesas['ds_municipio'].unique())
proporcao = round(quantidade_municipios_unicos/municipios_sao_paulo*100,1)

print('O estado de São Paulo possui {} municipios'.format(municipios_sao_paulo))
print('A amostra utilizada, possui {} municipios unicos'.
 ↪format(quantidade_municipios_unicos))
print('Representando {}% dos municipios de SP'.format(proporcao))
```

```
O estado de São Paulo possui 645 municipios
A amostra utilizada, possui 644 municipios unicos
Representando 99.8% dos municipios de SP
```

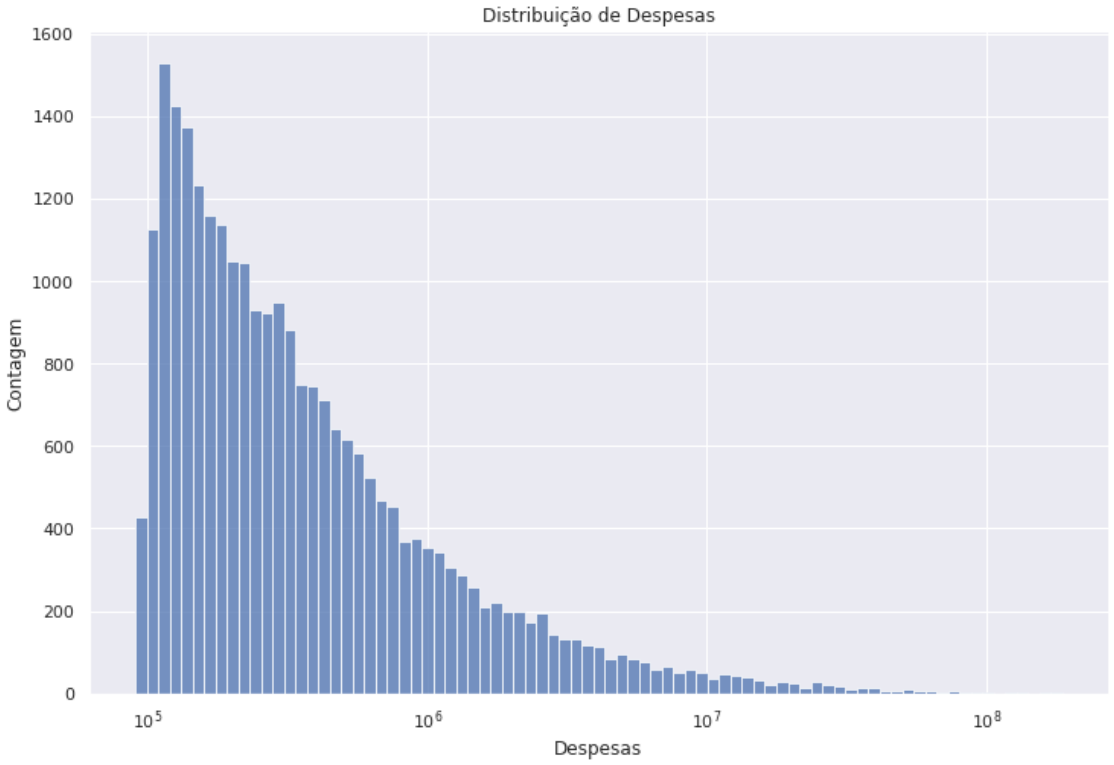
Distribuição Despesas Município

[9]:
```
sns.set(rc={'figure.figsize':(12, 8)})

ax = sns.histplot(despesas, x='soma', log_scale=True)

ax.set(title='Distribuição de Despesas')
ax.set(xlabel='Despesas', ylabel='Contagem')
```

[9]:
```
[Text(0.5, 0, 'Despesas'), Text(0, 0.5, 'Contagem')]
```



Municípios Que Mais Gastaram

[10]:
```
municipios_despesas = despesas.groupby('ds_municipio').sum()
municipios_despesas = municipios_despesas.sort_values('soma', ascending=False)
top_10_municipio_despesas = municipios_despesas.head(10).reset_index()
```

[11]:
```
top_10_municipio_despesas
```

[11]:

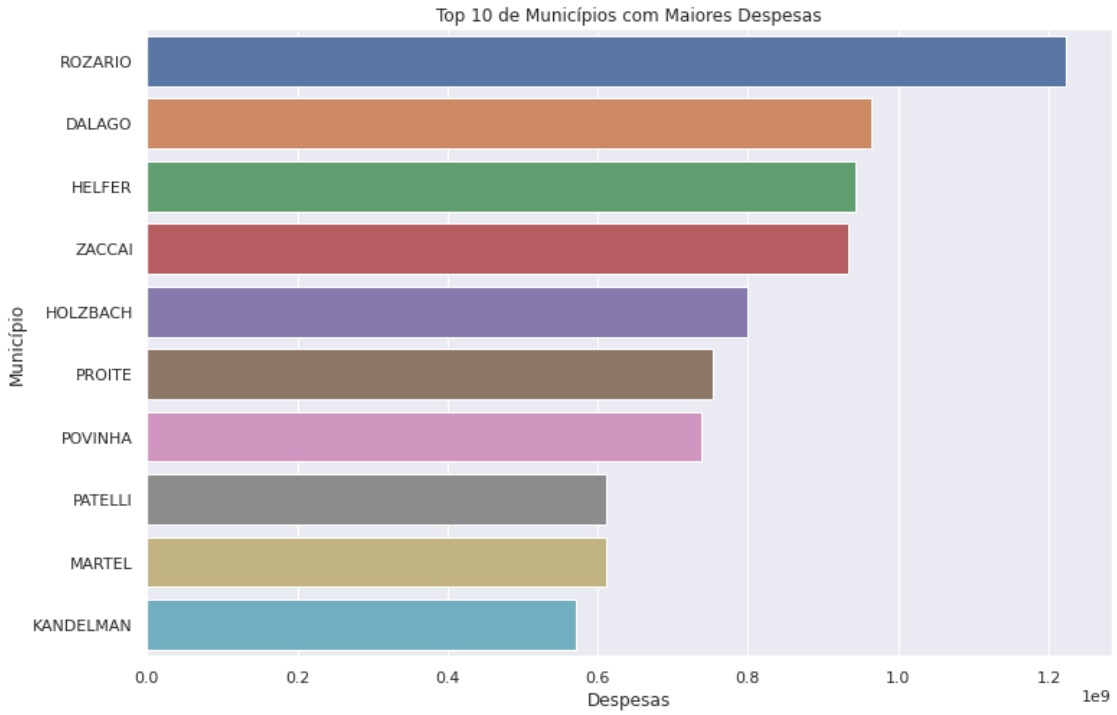
	ds_municipio	soma
0	ROZARIO	1.223180e+09
1	DALAGO	9.643243e+08
2	HELFER	9.434388e+08
3	ZACCAI	9.342326e+08
4	HOLZBACH	7.990991e+08
5	PROITE	7.534733e+08
6	POVINHA	7.382816e+08
7	PATELLI	6.120559e+08
8	MARTEL	6.112997e+08
9	KANDELMAN	5.709025e+08

[12]:
```
ax = sns.barplot(data=top_10_municipio_despesas, x='soma', y='ds_municipio')

ax.set(title='Top 10 de Municípios com Maiores Despesas')
```

```
ax.set(xlabel='Despesas', ylabel='Município')
```

[12]:
```
[Text(0.5, 0, 'Despesas'), Text(0, 0.5, 'Município')]
```



[ ]:
```

```

2.1.2 Empresas

Tabela de empresas da receita federal.

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

Os dados originais podem ser obtidos em: https://www.gov.br/receitafederal/pt-br/assuntos/orientacao-tributaria/cadastros/consultas/dados-publicos-cnpj

[13]:
```
sql = 'SELECT * FROM empresas'
empresas = pd.read_sql(sql, engine, index_col=None)
```

Aqui podemos visualizar uma amostra dos dados

[14]:
```
empresas.sample(5)
```

[14]:

	cnpj	matriz_filial	razao_social	nome_fantasia	situacao	\
2723	01906093000778	2	ZSELYKE GAS SA		02	
112	08179943000160	1	MATS GERAL SA		02	

```
3726  01780119000179             1  THEODOR SHOPPING                    02
612   00916501000179             1         ROMY MODAS                   02
2745  33831577000196             1    SOPHIE TURISMO                    02

     data_situacao motivo_situacao nm_cidade_exterior cod_pais nome_pais  …  \
2723      20200101              37                                        …
112       20200101              21                                        …
3726      20200101              00                                        …
612       20200101              73                                        …
2745      20200101              00                                        …

     email qualif_resp capital_social porte opc_simples data_opc_simples  \
2723                            0.0    05           0
112                             0.0    01           0
3726                            0.0    05           0
612                             0.0    05           0
2745                            0.0    05           0

     data_exc_simples opc_mei sit_especial data_sit_especial
2723
112                       N
3726                      N
612                       N
2745                      N

[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.

[15]:
```
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.

Os dados originais podem ser obtidos em: https://www.gov.br/receitafederal/pt-br/assuntos/orientacao-tributaria/cadastros/consultas/dados-publicos-cnpj

[16]:
```
sql = 'SELECT * FROM socios'
socios = pd.read_sql(sql, engine, index_col=None)
```

Aqui podemos visualizar uma amostra dos dados

[17]:
```
socios.sample(5)
```

[17]:

	cnpj	tipo_socio	nome_socio	\
579	54015456000174	2	SPERETA UNAIDE DE PRANTEL ABUBAKIR	
440	05810717000502	2	ANDRE WEISMANN RAGO NULMAN	
5710	25943738000179	2	RAIFUR ALEXANDRE SALARINE	
2887	01848331000198	2	JOAO ZACHHUBER NETO	

```
4559  00031146000161          2        NORBERTO DE DANNIELLY JUNIOR

     cnpj_cpf_socio cod_qualificacao  perc_capital data_entrada cod_pais_ext  \
579     ***844638**               49           0.0     20200101
440     ***914177**               22           0.0     20200101
5710    ***738748**               49           0.0     20200101
2887    ***052228**               59           0.0     20200101
4559    ***176878**               22           0.0     20200101

     nome_pais_ext cpf_repres nome_repres cod_qualif_repres
579                                                     00
440                                                     00
5710                                                    00
2887                                                    00
4559                                                    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.

[18]:
```
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 Carregar dados de Pessoas Expostas Politicamente

As Pessoas Expostas Politicamente são os agentes públicos que desempenham ou tenham desempenhado, nos cinco anos anteriores, no Brasil ou em países, territórios e dependências estrangeiros, cargos, empregos ou funções públicas relevantes, assim como seus representantes, familiares e outras pessoas de seu relacionamento próximo.

Possui dados alterados, mas a estrutura das colunas é semelhante a base pública do Portal da

Transparência.

O arquivo original pode ser obtido em http://www.portaldatransparencia.gov.br/download-de-dados/pep

[19]:
```
arquivo_peps = 'PEP_SP.csv'
filepath_arquivo_pep = os.path.join(diretorio_dados, arquivo_peps)
```

[20]:
```
peps = pd.read_csv(filepath_arquivo_pep, sep=';')
```

Aqui podemos visualizar uma amostra dos dados

[21]:
```
peps.sample(5)
```

[21]:

	CPF	Nome_PEP	\
18	***.002.608-**	SOEL BONAVOGLIA ZUKOWISKI PHREDERICO ZIELAK	
2703	***.327.798-**	GOSSON MOSCHETO PENUELA FILHO	
1922	***.233.248-**	RAPPO JUNOR ROVILSOM	
1153	***.143.178-**	VANDIERENDONCK GRIZOLIA CARNAZ	
1054	***.132.788-**	RONALDO DA SUSAKO KRASIUK	

	Sigla_Função	Descrição_Função	Nível_Função	Nome_Órgão	\
18	VEREAD	VEREADOR	NaN	MUN. DE BEUST-SP	
2703	VEREAD	VEREADOR	NaN	MUN. DE GUIRRA-SP	
1922	PREFEI	PREFEITO	NaN	MUN. DE BUFFULIN-SP	
1153	VEREAD	VEREADOR	NaN	MUN. DE SCLAVI-SP	
1054	VEREAD	VEREADOR	NaN	MUN. DE JAMUR-SP	

	Data_Início_Exercício	Data_Fim_Exercício	Data_Fim_Carência	uf_orgao
18	01/01/2017	31/12/2020	31/12/2025	SP
2703	01/01/2017	31/12/2020	31/12/2025	SP
1922	01/01/2017	31/12/2020	31/12/2025	SP
1153	01/01/2017	31/12/2020	31/12/2025	SP
1054	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.

[22]:
```
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
```

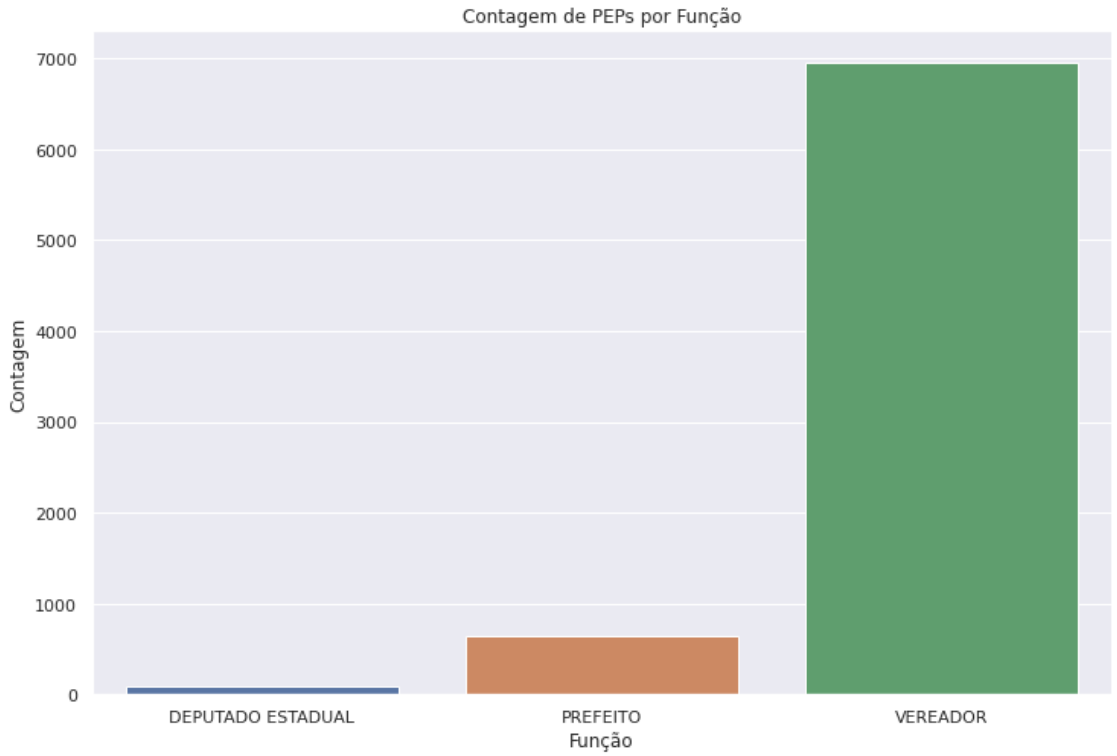
```
 4   Nível_Função            0 non-null      float64
 5   Nome_Órgão              7690 non-null   object
 6   Data_Início_Exercício   7690 non-null   object
 7   Data_Fim_Exercício      7690 non-null   object
 8   Data_Fim_Carência       7690 non-null   object
 9   uf_orgao                7690 non-null   object
dtypes: float64(1), object(9)
memory usage: 600.9+ KB
```

2.2.1 Contagem de PEPs por Função

[23]:
```
ax = sns.countplot(data=peps.sort_values('Descrição_Função'),␣
↪x='Descrição_Função')

ax.set(title='Contagem de PEPs por Função')
ax.set(xlabel='Função', ylabel='Contagem')
```

[23]:
```
[Text(0.5, 0, 'Função'), Text(0, 0.5, 'Contagem')]
```



3 Execução

[24]:
```
# 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)
```

3.1 Tratamento dos dados

Alguns dados precisam ser tratados antes da análise.

3.1.1 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

[25]:
```
empresas['data_situacao'].sample(5)
```

[25]:
```
4851    20200101
504     20200101
3107    20200101
4507    20200101
3997    20200101
Name: data_situacao, dtype: object
```

[26]:
```
empresas['data_situacao_normalizado'] = pd.
↪to_datetime(empresas['data_situacao']).dt.date
```

[27]:
```
empresas['data_situacao_normalizado'].sample(5)
```

[27]:
```
5305    2020-01-01
735     2020-01-01
3612    2020-01-01
3348    2020-01-01
1467    2020-01-01
Name: data_situacao_normalizado, dtype: object
```

3.1.2 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

[28]:
```
socios['data_entrada'].sample(5)
```

[28]:
```
4644    20200101
4834    20200101
4455    20200101
2529    20200101
2806    20200101
Name: data_entrada, dtype: object
```

[29]:
```
socios['data_entrada_normalizado'] = pd.to_datetime(socios['data_entrada']).dt.
↪date
```

[30]:
```
socios['data_entrada_normalizado'].sample(5)
```

[30]:
```
5374    2020-01-01
881     2020-01-01
5378    2020-01-01
1891    2020-01-01
2192    2020-01-01
Name: data_entrada_normalizado, dtype: object
```

3.1.3 Pessoas Politicamente Expostas

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

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

[31]:
```
5501    ***.711.738-**
1173    ***.145.688-**
6141    ***.793.328-**
5969    ***.769.808-**
4318    ***.554.258-**
Name: CPF, dtype: object
```

[32]:
```
peps['CPF_normalizado'] = peps['CPF'].str.replace(".", "", regex=False).str.
↪replace("-", "", regex=False)
```

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

[33]:
```
3972    ***506688**
2978    ***362368**
3150    ***388848**
4023    ***515168**
5783    ***747468**
Name: CPF_normalizado, dtype: object
```

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

[34]:
```
peps['Nome_Órgão'].sample(5)
```

[34]:
```
3272    MUN. DE WAIHRICH-SP
5321      MUN. DE ARNOLD-SP
6814       MUN. DE TEBON-SP
7457     MUN. DE BOARINI-SP
5826         MUN. DE RIM-SP
Name: Nome_Órgão, dtype: object
```

[35]:
```
peps['Municipio_normalizado'] = peps['Nome_Órgão'].str.replace("MUN. DE ", "",␣
↪regex=False).str.replace("-SP", "", regex=False)
```

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

[36]:
```
2910           KNAB
4003         ARIZIO
6234    TARABORRELLI
4135       ZHENGJIAN
2521         CETOLIN
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

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

[37]:
```
4463    01/01/2017
3065    01/01/2017
5584    01/01/2017
7269    01/01/2017
2833    01/01/2017
Name: Data_Início_Exercício, dtype: object
```

[38]:
```
peps['Data_Fim_Exercício'].sample(5)
```

[38]:
```
1826    31/12/2020
535     31/12/2020
7497    31/12/2020
5947    31/12/2020
1584    31/12/2020
Name: Data_Fim_Exercício, dtype: object
```

[39]:
```
peps['Data_Fim_Carência'].sample(5)
```

[39]:
```
1715    31/12/2025
364     31/12/2025
282     31/12/2025
```

```
7453    31/12/2025
5197    31/12/2025
Name: Data_Fim_Carência, dtype: object
```

[40]:
```
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
```

[41]:
```
peps['Data_Inicio_Exercicio_normalizado'].sample(5)
```

[41]:
```
4819    2017-01-01
2062    2017-01-01
6564    2017-01-01
7289    2017-01-01
6215    2017-01-01
Name: Data_Inicio_Exercicio_normalizado, dtype: object
```

[42]:
```
peps['Data_Fim_Exercicio_normalizado'].sample(5)
```

[42]:
```
2920    2020-12-31
3818    2020-12-31
6888    2020-12-31
6803    2020-12-31
5365    2020-12-31
Name: Data_Fim_Exercicio_normalizado, dtype: object
```

[43]:
```
peps['Data_Fim_Carência_normalizado'].sample(5)
```

[43]:
```
7246    2025-12-31
334     2025-12-31
4953    2025-12-31
1437    2025-12-31
2944    2025-12-31
Name: Data_Fim_Carência_normalizado, dtype: object
```

3.2 Gerar Banco de Dados

3.2.1 Copiar banco original para diretorio de saida

[44]:
```
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)
```

[44]:
```
'saida/output.db'
```

3.2.2 Gravar tabela de prefeitos_vereadores em banco gerado

[45]:
```
url_banco = "sqlite:///{}".format(filepath_arquivo_banco_saida)
engine_banco_saida = sqlalchemy.create_engine(url_banco)
```

[46]:
```
peps.to_sql('peps', con=engine_banco_saida)
```

3.3 Consultar Empresas Relacionadas com Pessoas Politicamente Expostas que Receberam Recursos

[47]:
```
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'
```

[48]:
```
empresas_peps = pd.read_sql(sql, engine_banco_saida, index_col=None)
```

[49]:
```
empresas_peps.sample(5)
```

[49]:

	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	Data_Fim_PEP	Data_Carencia_PEP	Municipio_PEP
69	ASSUB	49524688000199	IBAI VESTUARIO	113162.42	49524688000199	IBAI VESTUARIO	ATIVA	20200101	2038	20201019	…	***271878**	20200101	JAIME BONAVOGLIA DA TEVEIRA	***271878**	PREFEITO	2017-01-01	2020-12-31	2025-12-31	ASSUB
8	VALLIS	55815300000213	AMINE IMOBILIARIOS	747080.25	55815300000213	AMINE IMOBILIARIOS	ATIVA	20200101	3999	19870728	…	***890658**	20200101	AMARILDO SABATE KAILER	***890658**	PREFEITO	2017-01-01	2020-12-31	2025-12-31	SOLIZ
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	2020-12-31	2025-12-31	HACKEL
30	DALAGO	01379160000172	ALEKSANDRA INDUSTRIAS	285251.53	01379160000172	ALEKSANDRA INDUSTRIAS	ATIVA	20200101	2062	20070307	…	***063078**	20200101	SANDRA PLANCKE DALTOM DIFANTE ITUE	***063078**	VEREADOR	2017-01-01	2020-12-31	2025-12-31	SAO PAULO
3	POSE	03195017000178	JONA SPE	8875293.18	03195017000178	JONA SPE	ATIVA	20200101	2240	20200423	…	***294438**	20200101	TOSHIO MARIOSI	***294438**					

```
3    PREFEITO       2017-01-01    2020-12-31        2025-12-31     SUCHOBKOW

   UF_PEP
69     SP
8      SP
17     SP
30     SP
3      SP

[5 rows x 24 columns]
```

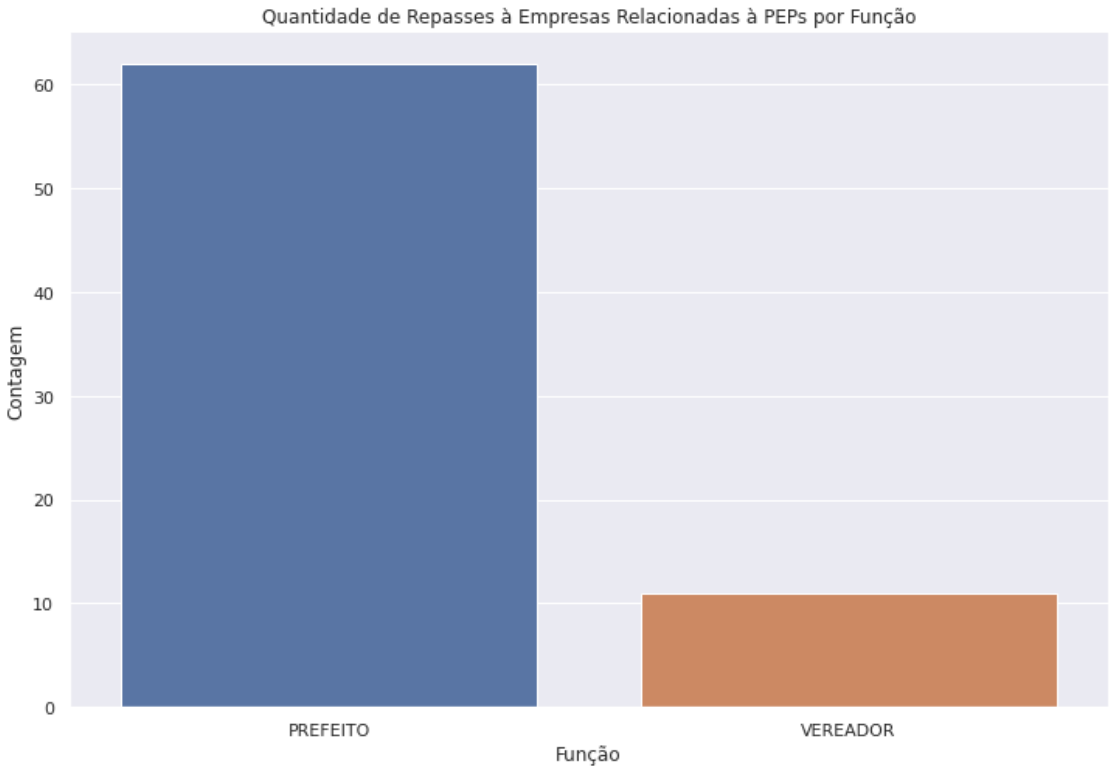
3.3.1 Visualização

Quantidade de Repasses à Empresas Relacionadas à PEPs por Função

[50]:
```
ax = sns.countplot(data=empresas_peps.sort_values('Funcao_PEP'), x='Funcao_PEP')

ax.set(title='Quantidade de Repasses à Empresas Relacionadas à PEPs por Função')
ax.set(xlabel='Função', ylabel='Contagem')
```

[50]:
```
[Text(0.5, 0, 'Função'), Text(0, 0.5, 'Contagem')]
```

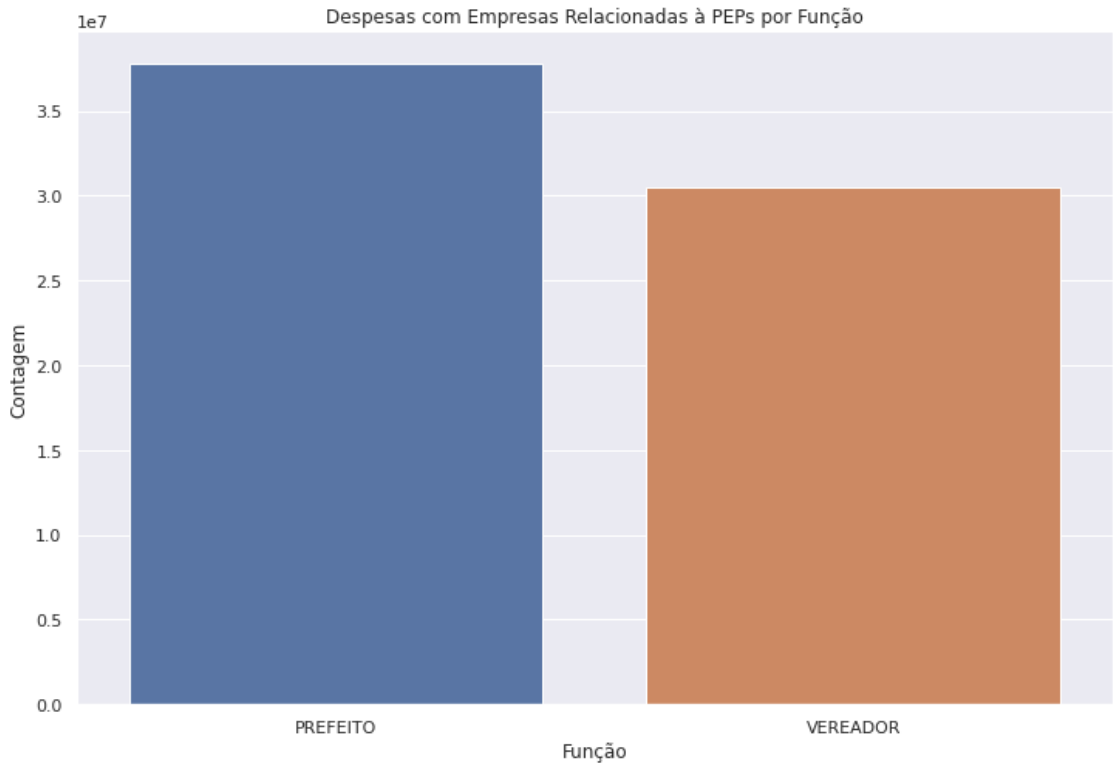


Despesas com Empresas Relacionadas à PEPs por Função

[51]:
```
soma_valores_empresas_peps_funcao = empresas_peps.
 ↪groupby('Funcao_PEP')['Valor_Total'].sum().reset_index()
soma_valores_empresas_peps_funcao = soma_valores_empresas_peps_funcao.
 ↪sort_values('Funcao_PEP')
ax = sns.barplot(data=soma_valores_empresas_peps_funcao, x='Funcao_PEP',␣
 ↪y='Valor_Total')

ax.set(title='Despesas com Empresas Relacionadas à PEPs por Função')
ax.set(xlabel='Função', ylabel='Contagem')
```

[51]:
```
[Text(0.5, 0, 'Função'), Text(0, 0.5, 'Contagem')]
```

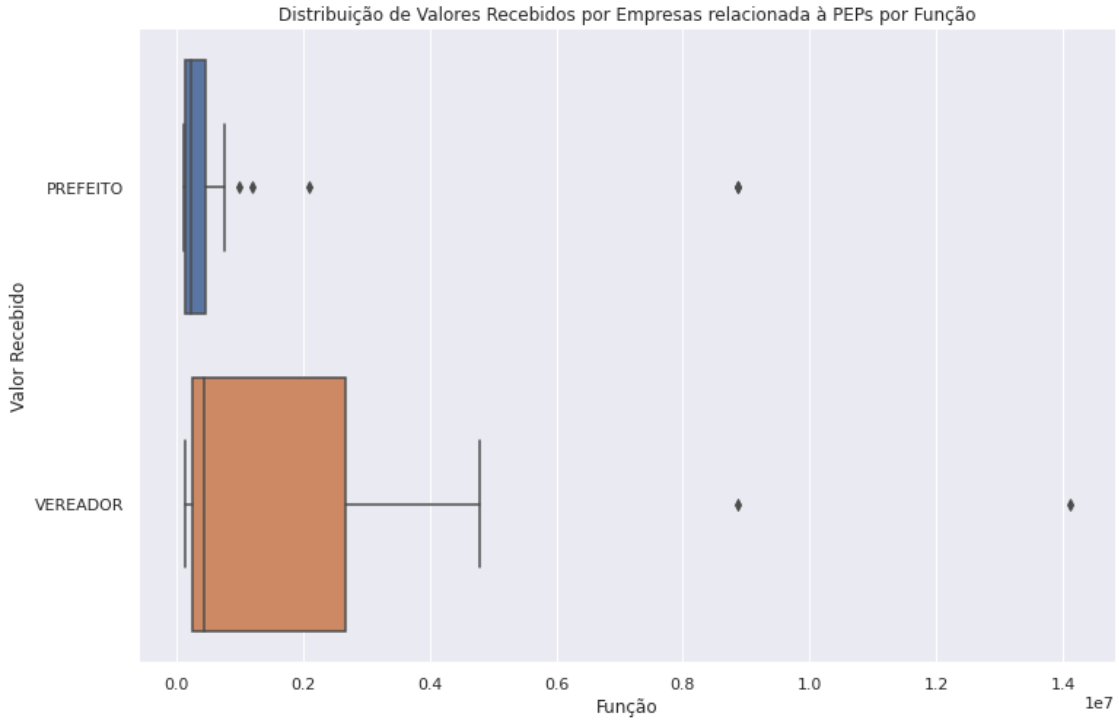


Distribuição de Valores Recebidos por Empresas relacionada à PEPs por Função

[52]:
```
ax = sns.boxplot(data=empresas_peps.sort_values('Funcao_PEP'), x="Valor_Total",␣
 ↪y="Funcao_PEP")

ax.set(title='Distribuição de Valores Recebidos por Empresas relacionada à PEPs␣
 ↪por Função')
ax.set(xlabel='Função', ylabel='Valor Recebido')
```

[52]:
```
[Text(0.5, 0, 'Função'), Text(0, 0.5, 'Valor Recebido')]
```



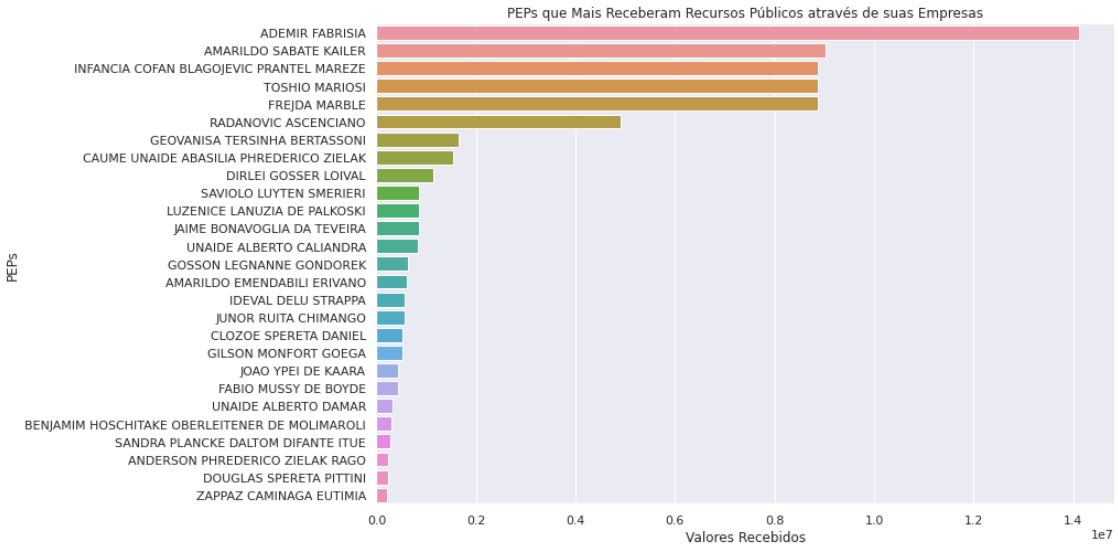
PEPS com maiores Recebimentos

[53]:
```
valores_por_pep = empresas_peps.groupby('Nome_PEP')['Valor_Total'].sum().
↪sort_values(ascending=False).reset_index()
```

[54]:
```
ax = sns.barplot(data=valores_por_pep, x='Valor_Total', y='Nome_PEP')

ax.set(title='PEPs que Mais Receberam Recursos Públicos através de suas␣
↪Empresas')
ax.set(xlabel='Valores Recebidos', ylabel='PEPs')
```

[54]:
```
[Text(0.5, 0, 'Valores Recebidos'), Text(0, 0.5, 'PEPs')]
```



3.3.2 Salvar dados

[55]:
```
arquivo_saida = 'empresas_peps.csv'
filepath = os.path.join(diretorio_saida_dados, arquivo_saida)
empresas_peps.to_csv(filepath, index=False)
```

3.4 Consultar Empresas Relacionadas com Pessoas Politicamente Expostas que Receberam Recursos dos Municípios que Atuam

[56]:
```
empresas_peps_mesmo_municipio = empresas_peps[empresas_peps['Municipio_PEP'] ==␣
↪empresas_peps['Nome_Municipio_Ficticio']]
```

[57]:
```
empresas_peps_mesmo_municipio.shape
```

[57]:
```
(11, 24)
```

[58]:
```
empresas_peps_mesmo_municipio.sample(5)
```

[58]:
```
   Nome_Municipio_Ficticio CNPJ_Empresa_Beneficiaria  \
52                  ROZARIO            08262075000137
54                   SACCOL            06273922000124
6                     SOLIZ            55815300000213
16                SUCHOBKOW            56776669000107
17                   HACKEL            04578574000175

   Razao_Social_Empresa_Beneficiaria  Valor_Total        CNPJ_RFB  \
52           MARIUS CONTABIL EIRELLI    135193.38  08262075000137
54            MOMOKA ARTISTICAS LTDA    141497.58  06273922000124
```

```
6                AMINE IMOBILIARIOS   1193598.13  55815300000213
16                  FINN TECNOLOGIA    559375.94  56776669000107
17                SOMA TURISMO EPP    513335.60  04578574000175

          Razao_Social_RFB Situacao_Cadastral_RFB  \
52  MARIUS CONTABIL EIRELLI                BAIXADA
54   MOMOKA ARTISTICAS LTDA                  ATIVA
6        AMINE IMOBILIARIOS                  ATIVA
16          FINN TECNOLOGIA                  ATIVA
17        SOMA TURISMO EPP                   ATIVA

    Data_Situacao_Cadastral_RFB Codigo_Natureza_Juridica_RFB  \
52                     20200101                         3999
54                     20200101                         3999
6                      20200101                         3999
16                     20200101                         3999
17                     20200101                         2062

    Data_Inicio_Atividade_RFB  … CPF_Socio_RFB Data_Entrada_Socio_RFB  \
52                   20030308  …   ***341758**               20200101
54                   20000214  …   ***923478**               20200101
6                    19870728  …   ***890658**               20200101
16                   19860918  …   ***078588**               20200101
17                   19960117  …   ***149208**               20200101

                        Nome_PEP      CPF_PEP Funcao_PEP  Data_Inicio_PEP  \
52       GOSSON LEGNANNE GONDOREK  ***341758**   PREFEITO       2017-01-01
54  GEOVANISA TERSINHA BERTASSONI  ***923478**   PREFEITO       2017-01-01
6           AMARILDO SABATE KAILER  ***890658**   PREFEITO       2017-01-01
16             IDEVAL DELU STRAPPA  ***078588**   VEREADOR       2017-01-01
17           CLOZOE SPERETA DANIEL  ***149208**   PREFEITO       2017-01-01

   Data_Fim_PEP Data_Carencia_PEP Municipio_PEP UF_PEP
52   2020-12-31        2025-12-31       ROZARIO     SP
54   2020-12-31        2025-12-31        SACCOL     SP
6    2020-12-31        2025-12-31         SOLIZ     SP
16   2020-12-31        2025-12-31     SUCHOBKOW     SP
17   2020-12-31        2025-12-31        HACKEL     SP

[5 rows x 24 columns]
```

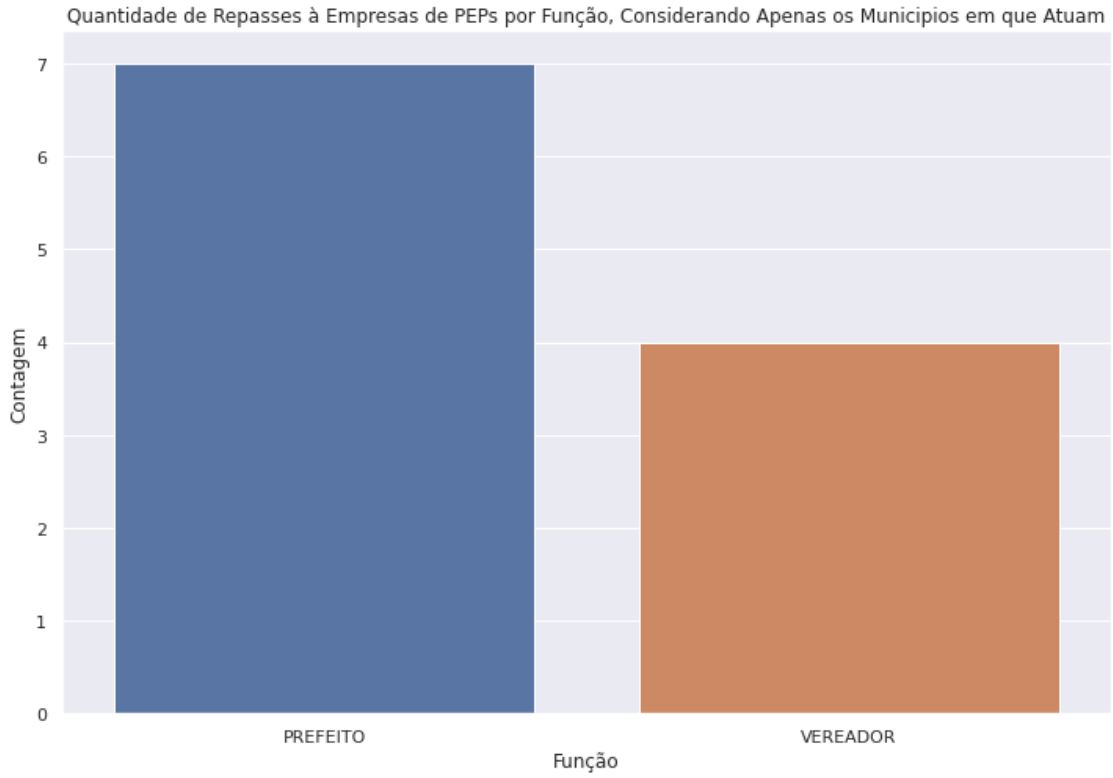
3.4.1 Visualização

Quantidade de Repasses à Empresas Relacionadas à PEPs por Função Considerando Apenas os Municipios em que Atuam

[59]:
```
ax = sns.countplot(data=empresas_peps_mesmo_municipio.
 ↪sort_values('Funcao_PEP'), x='Funcao_PEP')

ax.set(title='Quantidade de Repasses à Empresas de PEPs por Função,␣
 ↪Considerando Apenas os Municipios em que Atuam')
ax.set(xlabel='Função', ylabel='Contagem')
```

[59]:
```
[Text(0.5, 0, 'Função'), Text(0, 0.5, 'Contagem')]
```

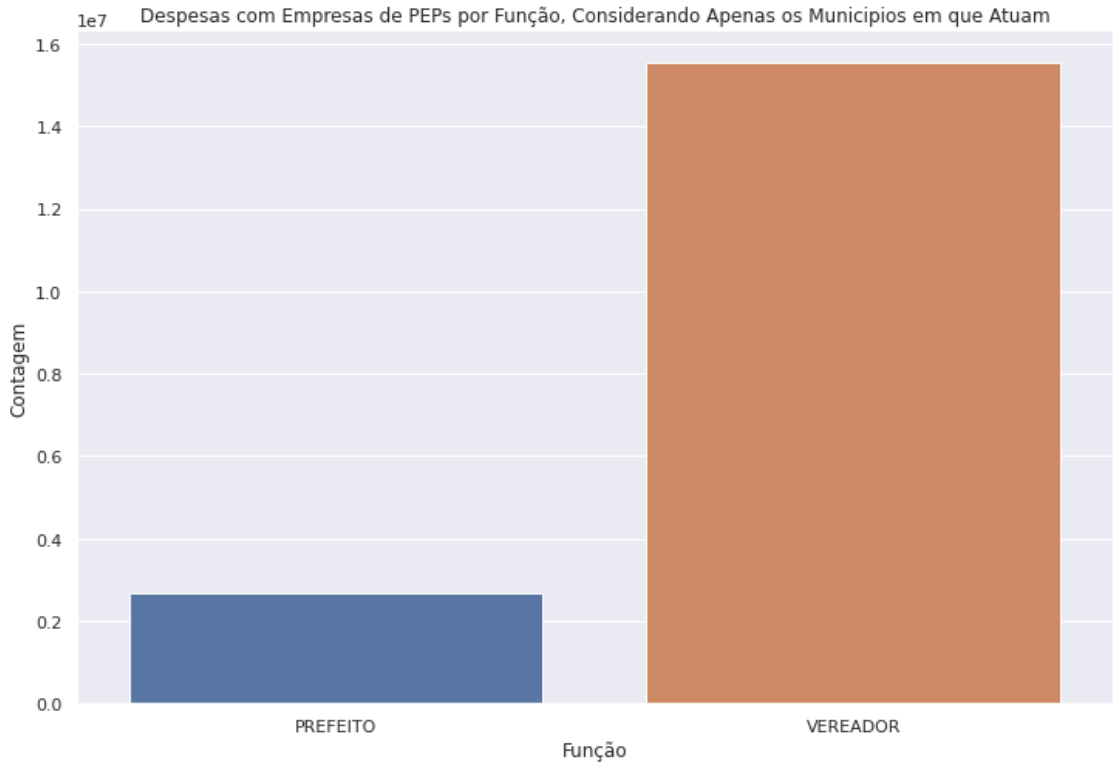


Despesas com Empresas Relacionadas à PEPs por Função, Considerando Apenas os Municipios em que Atuam

[60]:
```
soma_valores_empresas_peps_funcao = empresas_peps_mesmo_municipio.
 ↪groupby('Funcao_PEP')['Valor_Total'].sum().reset_index()
soma_valores_empresas_peps_funcao = soma_valores_empresas_peps_funcao.
 ↪sort_values('Funcao_PEP')
ax = sns.barplot(data=soma_valores_empresas_peps_funcao, x='Funcao_PEP',␣
 ↪y='Valor_Total')

ax.set(title='Despesas com Empresas de PEPs por Função, Considerando Apenas os␣
 ↪Municipios em que Atuam')
ax.set(xlabel='Função', ylabel='Contagem')
```

[60]: [Text(0.5, 0, 'Função'), Text(0, 0.5, 'Contagem')]

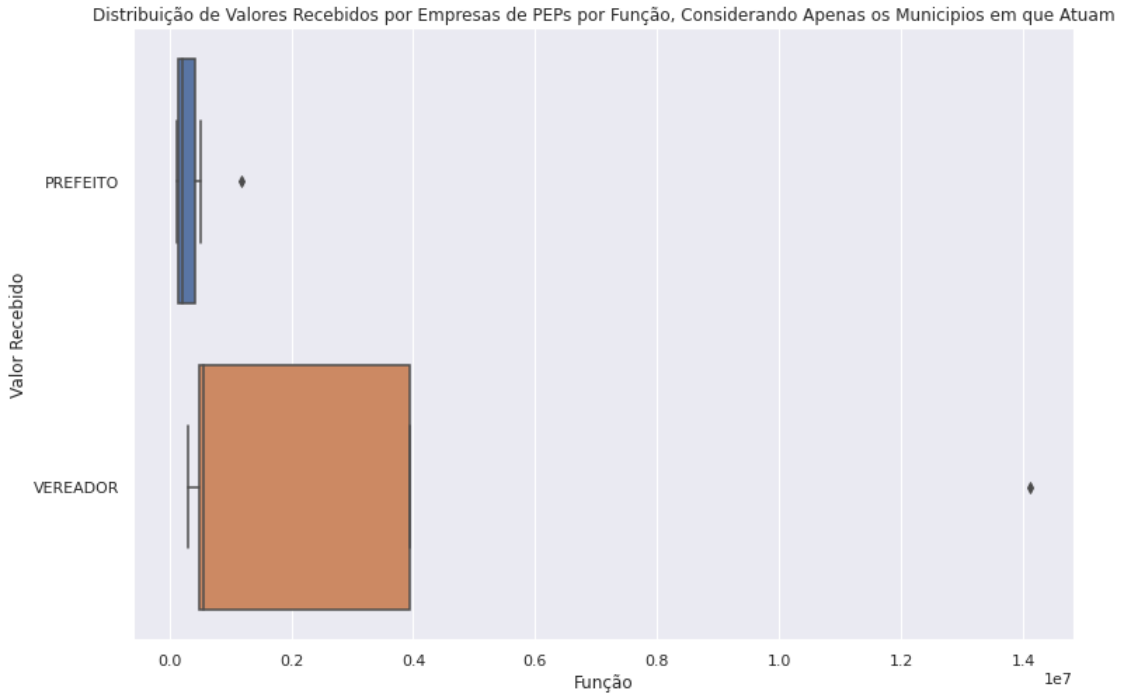


Distribuição de Valores Recebidos por Empresas relacionada à PEPs por Função

[61]:
```
ax = sns.boxplot(data=empresas_peps_mesmo_municipio.sort_values('Funcao_PEP'),␣
↪x="Valor_Total", y="Funcao_PEP")

ax.set(title='Distribuição de Valores Recebidos por Empresas de PEPs por␣
↪Função, Considerando Apenas os Municipios em que Atuam')
ax.set(xlabel='Função', ylabel='Valor Recebido')
```

[61]: [Text(0.5, 0, 'Função'), Text(0, 0.5, 'Valor Recebido')]



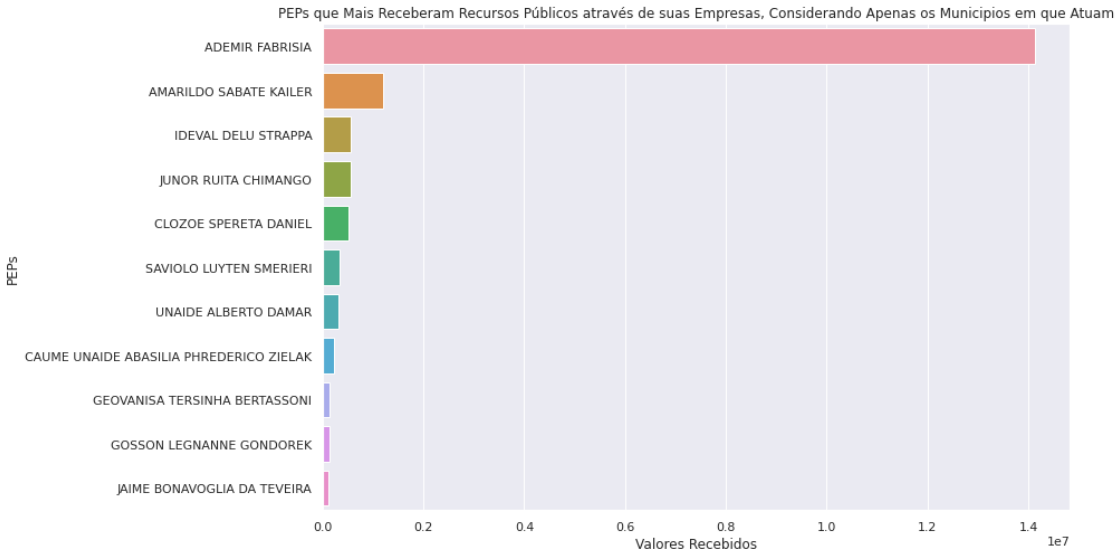
PEPS com maiores Recebimentos

[62]:
```
valores_por_pep = empresas_peps_mesmo_municipio.
↪groupby('Nome_PEP')['Valor_Total'].sum().sort_values(ascending=False).
↪reset_index()
```

[63]:
```
ax = sns.barplot(data=valores_por_pep, x='Valor_Total', y='Nome_PEP')

ax.set(title='PEPs que Mais Receberam Recursos Públicos através de suas␣
↪Empresas, Considerando Apenas os Municipios em que Atuam')
ax.set(xlabel='Valores Recebidos', ylabel='PEPs')
```

[63]:
```
[Text(0.5, 0, 'Valores Recebidos'), Text(0, 0.5, 'PEPs')]
```



3.4.2 Salvar dados

[64]:
```
arquivo_saida = 'empresas_peps_mesmo_municipio.csv'
filepath = os.path.join(diretorio_saida_dados, arquivo_saida)
empresas_peps_mesmo_municipio.to_csv(filepath, index=False)
```

4 Visualização

4.1 Mapa de Localização de Empresas relacionadas à PEPs

Essa visualização não é possível via PDF

[65]:
```
arquivo_municipios = 'Municipios_SP_paridade.csv'
filepath_arquivo_municipios = os.path.join(diretorio_dados, arquivo_municipios)
municipios = pd.read_csv(filepath_arquivo_municipios, sep='|')

empresas_peps_coord = pd.merge(empresas_peps, municipios, how="inner",␣
↪left_on='Nome_Municipio_Ficticio', right_on='Municipio_ficticio')
```

[66]:
```
mapa_plot = folium.Map(location=[empresas_peps_coord.Latitude.mean(),␣
↪empresas_peps_coord.Longitude.mean()], zoom_start=7, control_scale=True)
for index, location_info in empresas_peps_coord.iterrows():
    description = "A empresa " +␣
↪location_info["Razao_Social_Empresa_Beneficiaria"] + " (CNPJ " +␣
↪location_info["CNPJ_Empresa_Beneficiaria"] + "), relacionada ao PEP " +␣
↪location_info["Nome_PEP"] + ", recebeu R$ " +␣
↪str(location_info["Valor_Total"])
    description.splitlines(True)
```

```
    folium.Marker(
        [location_info["Latitude"], location_info["Longitude"]],
        popup=description,
        tooltip=description,
        icon=folium.Icon(color="red", icon="info-sign")
    ).add_to(mapa_plot)
title_html = '''
             <h3 align="center" style="font-size:20px"><b>Localização de␣
 ↪Empresas Relacionadas â PEPs</b></h3>
             '''
mapa_plot.get_root().html.add_child(folium.Element(title_html))
mapa_plot
```

[66]:
```
<folium.folium.Map at 0x7f3ca7d5bc70>
```

[ ]:
```

```