

Pesquisa em Métodos Inferenciais com Bases de Compras na categoria Informática (TIC) no portal compras.gov

Objetivo Geral Diagnosticar e modelar o comportamento dos contratos de TIC firmados pelos Ministérios Federais brasileiros (2020–2024) e a caracterização dos fornecedores, buscando realizar inferências e comparações com o cenário nacional a partir dos dados CETIC BR.

Objetivos Específicos Realizar análise exploratória dos contratos e fornecedores;

Estimar características da população de fornecedores a partir da amostra;

Aplicar métodos inferenciais (intervalos de confiança, testes de hipóteses);

Realizar modelagem multivariada para entender preditores relevantes (Ex.: Logistic Regression, Random Forest);

Comparar nossos achados com as bases CETIC BR (Empresas e Governo).

Análise Exploratória de Dados (AED) com bases tratadas compras.gov

1. arquivo 'contratos_ministerios_consolidados_separados.csv' - Base de contratos dos 25 ministérios do poder executivo federal obtidos no compras.gov.br e já consolidados e tratados (série histórica de 2020 a 2024);
2. arquivo 'fornecedores_formatado.csv' - Amostra de 150 fornecedores com dados enriquecidos pela base CNPJá

1. Importar Bibliotecas

In [154...]

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import warnings
warnings.filterwarnings('ignore')

# Configurar estilo dos gráficos
```

```
plt.style.use('ggplot')
sns.set_theme(style="whitegrid")
```

2. Carregar as Bases de Dados

In [155...]

```
# 2. Funções auxiliares para visualização elegante
def resumo_geral(df, nome_df):
    print(f"📄 {nome_df}: {df.shape[0]}:,} linhas x {df.shape[1]}:,} colunas")
    display(df.head(5).style.set_caption(f"Prévia de {nome_df}").set_table_styles(
        [{ 'selector': 'caption', 'props': [ ('caption-side', 'top'), ('font-size', '16px'), ('font-weight', 'bold') ] }]))
    info = pd.DataFrame({
        'Coluna': df.columns,
        'Tipo': df.dtypes.values,
        'Missing (%)': df.isnull().mean().round(4) * 100
    })
    display(info.style.background_gradient(subset=['Missing (%)'], cmap='Reds'))

# 3. Carregar as bases

# 3.1 Contratos Ministério (Compras.Gov.br)
df_contratos = pd.read_csv('contratos_ministerios_consolidados_separados.csv', sep=',', encoding='utf-8')

# 3.2 Fornecedores Amostra (Base CNPJá)
df_fornecedores = pd.read_csv('fornecedores_formatado.csv', sep=',', encoding='utf-8')

# 3.3 CETIC BR - Governo
df_cetic_gov = pd.read_excel('Consolidado_tic_egov_orgaos_federais_e_estaduais.xlsx')

# 3.4 CETIC BR - Empresas
df_cetic_empresas = pd.read_excel('TIC Empresas consoolidado historico.xlsx')

# 4. Exibir resumos
resumo_geral(df_contratos, "Contratos - Ministérios (Compras.gov.br)")
resumo_geral(df_fornecedores, "Fornecedores Amostra (Base CNPJá)")
resumo_geral(df_cetic_gov, "CETIC BR - Governo")
resumo_geral(df_cetic_empresas, "CETIC BR - Empresas")
```

📄 Contratos - Ministérios (Compras.gov.br): 1,088 linhas x 12 colunas

Prévia de Contratos - Ministérios (Compras.gov.br)

	Órgão	Unidade Gestora	Número Contrato	Fornecedor	Vig. Início	Vig. Fim	Valor Global	Núm. Parcelas	Valor Parcela	ministeric
0	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130025 - SFA/PE/ MAPA	00003/2019	64.799.539/0001-35 - TECNOSET INFORMATICA PRODUTOS E SERVICOS LTDA	07/05/2019	06/05/2023	R\$ 62.015,76	12	R\$ 5.167,98	Consulta Contratos: Contratos.gov.br Min Agricultura 22000
1	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130025 - SFA/PE/ MAPA	00006/2023	07.759.174/0001-81 - SOLUCOES SERVICOS DE LOCACAO DE MAQUINAS E EQUIPAMENTOS PARA ESCRITORIO LTDA	01/06/2023	31/05/2025	R\$ 102.850,00	12	R\$ 8.570,83	Consulta Contratos: Contratos.gov.br Min Agricultura 22000
2	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130067 - SFA/SP/ MAPA	00006/2023	07.432.517/0001-07 - SIMPRESS COMERCIO LOCACAO E SERVICOS LTDA	01/04/2023	01/04/2027	R\$ 720.238,67	48	R\$ 15.004,97	Consulta Contratos: Contratos.gov.br Min Agricultura 22000
3	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130069 - SFA/MA/ MAPA	00004/2022	08.951.049/0001-31 - CORESMA - COMERCIO DE EQUIPAMENTOS E SUPRIMENTOS LTDA	23/09/2022	22/09/2025	R\$ 58.860,00	12	R\$ 4.905,00	Consulta Contratos: Contratos.gov.br Min Agricultura 22000
4	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130094 - SFA/PA/ MAPA	00002/2023	03.117.534/0001-90 - BRADOK SOLUCOES CORPORATIVAS LTDA	21/11/2023	20/11/2027	R\$ 129.484,80	12	R\$ 10.790,40	Consulta Contratos: Contratos.gov.br Min Agricultura 22000

		Coluna	Tipo	Missing (%)
	Órgão	Órgão	object	0.000000
	Unidade Gestora	Unidade Gestora	object	0.000000
	Número Contrato	Número Contrato	object	0.000000
	Fornecedor	Fornecedor	object	0.000000
	Vig. Início	Vig. Início	object	0.000000
	Vig. Fim	Vig. Fim	object	0.000000
	Valor Global	Valor Global	object	0.000000
	Núm. Parcelas	Núm. Parcelas	int64	0.000000
	Valor Parcela	Valor Parcela	object	0.000000
	ministerio	ministerio	object	0.000000
	CNPJ	CNPJ	object	0.000000
	Nome Fornecedor	Nome Fornecedor	object	0.000000

 Fornecedores Amostra (Base CNPJá): 479 linhas × 14 colunas

Prévia de Fornecedores Amostra (Base CNPJá)

	CNPJ	Razão Social	Nome Fantasia	Natureza Jurídica	Capital Social	Data de Abertura	Idade da Empresa	Cidade	Estado	CEP	Por
0	113110000160	nan	nan	nan	nan	nan	nan	nan	nan	nan	n
1	258246000168	nan	nan	nan	nan	nan	nan	nan	nan	nan	n
2	306524000377	LEISTUNG COMERCIO E SERVICOS DE SISTEMAS DE ENERGIA LTDA.	Leistung	Sociedade Empresária Limitada	1230000.000000	2013-01-29	12.000000	Brasília	DF	70330520.000000	Dem
3	308141000176	nan	nan	nan	nan	nan	nan	nan	nan	nan	n
4	308141000923	nan	nan	nan	nan	nan	nan	nan	nan	nan	n

		Coluna	Tipo	Missing (%)
	CNPJ	CNPJ	int64	0.000000
	Razão Social	Razão Social	object	68.680000
	Nome Fantasia	Nome Fantasia	object	77.240000
	Natureza Jurídica	Natureza Jurídica	object	68.680000
	Capital Social	Capital Social	float64	69.310000
	Data de Abertura	Data de Abertura	object	68.680000
	Idade da Empresa	Idade da Empresa	float64	68.680000
	Cidade	Cidade	object	68.680000
	Estado	Estado	object	68.680000
	CEP	CEP	float64	68.680000
	Porte	Porte	object	68.680000
	Atividade Principal	Atividade Principal	object	68.680000
	Flag_CNEP	Flag_CNEP	int64	0.000000
	Flag_CEIS	Flag_CEIS	int64	0.000000

 CETIC BR - Governo: 286 linhas x 143 colunas

Column8	Column9	Column10	Column11	Column12	Column13	Column14	Column15	Column16	Column17	Column18
0	2021	b5a - órgãos públicos federais e estaduais com documento formalmente instituído de planejamento de tecnologia da informação, por tipo	nan	TOTAL	Total	832,0	492,0	28.000000	3,0	0.000000
1	2021	b5a - órgãos públicos federais e estaduais com documento formalmente instituído de planejamento de tecnologia da informação, por tipo	nan	PODER	Executivo	676,0	467,0	28.000000	3,0	0.000000
2	2021	b5a - órgãos públicos federais e estaduais com documento formalmente instituído de planejamento de tecnologia	nan	PODER	Legislativo	36,0	18,0	0.000000	0,0	0.000000

Column8	Column9	Column10	Column11	Column12	Column13	Column14	Column15	Column16	Column17	Column18
		da informação, por tipo								
3	2021	b5a - órgãos públicos federais e estaduais com documento formalmente instituído de planejamento de tecnologia da informação, por tipo	nan	PODER	Judiciário	95,0	1,0	0.000000	0.0	0.000000
4	2021	b5a - órgãos públicos federais e estaduais com documento formalmente instituído de planejamento de tecnologia da informação, por tipo	nan	PODER	Ministério Público	25,0	6,0	0.000000	0.0	0.000000

	Coluna	Tipo	Missing (%)
Column8	Column8	int64	0.000000
Column9	Column9	object	0.000000
Column10	Column10	float64	100.000000
Column11	Column11	object	0.000000
Column12	Column12	object	9.090000
Column13	Column13	object	16.080000
Column14	Column14	object	16.080000
Column15	Column15	float64	9.090000
Column16	Column16	object	19.580000
Column17	Column17	float64	40.560000
Column18	Column18	object	58.040000
Column19	Column19	object	58.040000
Column20	Column20	float64	58.040000
Column21	Column21	object	65.030000
Column22	Column22	float64	75.520000
Column23	Column23	object	61.540000
Column24	Column24	object	61.540000
Column25	Column25	object	61.540000
Column26	Column26	object	68.530000
Column27	Column27	float64	75.520000
Column28	Column28	object	65.030000
Column29	Column29	object	65.030000

	Coluna	Tipo	Missing (%)
Column30	Column30	object	65.030000
Column31	Column31	object	72.030000
Column32	Column32	float64	79.020000
Column33	Column33	object	79.020000
Column34	Column34	object	79.020000
Column35	Column35	object	79.020000
Column36	Column36	object	86.010000
Column37	Column37	float64	82.520000
Column38	Column38	object	86.010000
Column39	Column39	object	86.010000
Column40	Column40	object	86.010000
Column41	Column41	object	86.010000
Column42	Column42	float64	89.510000
Column43	Column43	object	93.010000
Column44	Column44	object	93.010000
Column45	Column45	float64	93.010000
Column46	Column46	object	93.010000
Column47	Column47	float64	93.010000
Column48	Column48	float64	100.000000
Column49	Column49	float64	100.000000
Column50	Column50	float64	100.000000
Column51	Column51	float64	100.000000

	Coluna	Tipo	Missing (%)
Column52	Column52	float64	100.000000
Column53	Column53	float64	100.000000
Column54	Column54	float64	100.000000
Column55	Column55	float64	100.000000
Column56	Column56	float64	100.000000
Column57	Column57	float64	100.000000
Column58	Column58	float64	100.000000
Column59	Column59	float64	100.000000
Column60	Column60	float64	100.000000
Column61	Column61	float64	100.000000
Column62	Column62	float64	100.000000
Column63	Column63	float64	100.000000
Column64	Column64	float64	100.000000
Column65	Column65	float64	100.000000
Column66	Column66	float64	100.000000
Column67	Column67	float64	100.000000
Column68	Column68	float64	100.000000
Column69	Column69	float64	100.000000
Column70	Column70	float64	100.000000
Column71	Column71	float64	100.000000
Column72	Column72	float64	100.000000
Column73	Column73	float64	100.000000

	Coluna	Tipo	Missing (%)
Column74	Column74	float64	100.000000
Column75	Column75	float64	100.000000
Column76	Column76	float64	100.000000
Column77	Column77	float64	100.000000
Column78	Column78	float64	100.000000
Column79	Column79	float64	100.000000
Column80	Column80	float64	100.000000
Column81	Column81	float64	100.000000
Column82	Column82	float64	100.000000
Column83	Column83	float64	100.000000
Column84	Column84	float64	100.000000
Column85	Column85	float64	100.000000
Column86	Column86	float64	100.000000
Column87	Column87	float64	100.000000
Column88	Column88	float64	100.000000
Column89	Column89	float64	100.000000
Column90	Column90	float64	100.000000
Column91	Column91	float64	100.000000
Column92	Column92	float64	100.000000
Column93	Column93	float64	100.000000
Column94	Column94	float64	100.000000
Column95	Column95	float64	100.000000

	Coluna	Tipo	Missing (%)
Column96	Column96	object	86.010000
Column97	Column97	float64	100.000000
Column98	Column98	float64	100.000000
Column99	Column99	float64	100.000000
Column100	Column100	float64	100.000000
Column101	Column101	float64	100.000000
Column102	Column102	float64	100.000000
Column103	Column103	float64	100.000000
Column104	Column104	float64	100.000000
Column105	Column105	float64	100.000000
Column106	Column106	float64	100.000000
Column107	Column107	float64	100.000000
Column108	Column108	float64	100.000000
Column109	Column109	float64	100.000000
Column110	Column110	float64	100.000000
Column111	Column111	float64	100.000000
Column112	Column112	float64	100.000000
Column113	Column113	float64	100.000000
Column114	Column114	float64	100.000000
Column115	Column115	float64	100.000000
Column116	Column116	float64	100.000000
Column117	Column117	float64	100.000000

	Coluna	Tipo	Missing (%)
Column118	Column118	float64	100.000000
Column119	Column119	float64	100.000000
Column120	Column120	float64	100.000000
Column121	Column121	float64	100.000000
Column122	Column122	float64	100.000000
Column123	Column123	float64	100.000000
Column124	Column124	float64	100.000000
Column125	Column125	float64	100.000000
Column126	Column126	float64	100.000000
Column127	Column127	float64	100.000000
Column128	Column128	float64	100.000000
Column129	Column129	float64	100.000000
Column130	Column130	float64	100.000000
Column131	Column131	float64	100.000000
Column132	Column132	float64	100.000000
Column133	Column133	float64	100.000000
Column134	Column134	float64	100.000000
Column135	Column135	float64	100.000000
Column136	Column136	float64	100.000000
Column137	Column137	float64	100.000000
Column138	Column138	object	93.010000
Column139	Column139	float64	93.010000

	Coluna	Tipo	Missing (%)
Column140	Column140	object	93.010000
Column141	Column141	object	96.500000
Column142	Column142	float64	100.000000
Column143	Column143	float64	100.000000
Column144	Column144	float64	100.000000
Column145	Column145	object	93.010000
Column146	Column146	object	93.010000
Column147	Column147	object	93.010000
Column148	Column148	object	93.010000
Column149	Column149	object	93.010000
Column150	Column150	object	93.010000

 CETIC BR - Empresas: 795 linhas x 105 colunas

			unnamed_0	unnamed_1	sim	não	não_sabe	não_respondeu	year	indicator	não_se_aplica	e_mail_em_nuvem	software_de_e
0	Total	Total	nan	nan	nan	nan	2019	b18 - empresas que pagaram por serviços em nuvem	nan	191597,0			
1	PORTE	De 10 a 49 pessoas ocupadas	nan	nan	nan	nan	2019	b18 - empresas que pagaram por serviços em nuvem	nan	158003,0			
2	PORTE	De 50 a 249 pessoas ocupadas	nan	nan	nan	nan	2019	b18 - empresas que pagaram por serviços em nuvem	nan	26994,0			
3	PORTE	De 250 pessoas ocupadas ou mais	nan	nan	nan	nan	2019	b18 - empresas que pagaram por serviços em nuvem	nan	6599,0			
4	REGIÃO	Norte	nan	nan	nan	nan	2019	b18 - empresas	nan	7893,0			

unnamed_0 unnamed_1 sim não não_sabe não_respondeu year indicator não_se_aplica e_mail_em_nuvem software_de_e
que
pagaram
por
serviços
em
nuvem

plataforma_de_computação_que_fornece_um_ambiente_hospitais_e_organizações_saúd

in

perso

serviço_de_chat_para_suporte_ao_cliente_como_um

anúncio_d

funções_e_responsabil

processos_para_posibil

processos_para_decidir_o_c

processos_para_posibil

me

discutiu_

deu_incentivos_de_desemp

promoveu_treinamento_sobre_a_gestão_de_risco_de_segurança_digital_como_cursos_on_line_workshops_seminários_co

promoveu_treinamento_sobre_a_gestão_de_risco_de_segurança_digital_como_cursos_online_workshops_seminários_co

serviço_de

serviço_d

pretensões_s

a_partir_de_dados_próprios_da_empresa_provenientes_de_dispositivos_inteligentes_ou_sensores_como_trocas_de_dados_entre_máquinas_se
a_partir_de_dados_de_geolocalização_provenientes_do_uso_de_dis
a_partir_de_dados_gerados_a_partir_de_mídias_sociais_como_redes_so

In [156...]

```
# 1. Importar bibliotecas
import pandas as pd
from IPython.display import display

# 2. Configurar Pandas para melhor visualização
pd.set_option('display.max_columns', None)    # Mostrar todas as colunas
pd.set_option('display.max_rows', 10)           # Limitar o número de linhas exibidas
```

```
pd.set_option('display.width', 1000)          # Largura da tela
pd.set_option('display.colheader_justify', 'center') # Centralizar nomes das colunas

# 3. Carregar os dados com caminhos ajustados
df_contratos = pd.read_csv('contratos_ministerios_consolidados_separados.csv', sep=',', encoding='utf-8')
df_fornecedores = pd.read_csv('fornecedores_formatado.csv', sep=',', encoding='utf-8')

# 4. Exibir com melhor formatação
print("📄 Base de Contratos:")
display(df_contratos.head(10)) # Mostra os 10 primeiros

print("📱 Base de Fornecedores:")
display(df_fornecedores.head(10)) # Mostra os 10 primeiros
```

📄 Base de Contratos:

	Órgão	Unidade Gestora	Número Contrato	Fornecedor	Vig. Início	Vig. Fim	Valor Global	Núm. Parcelas	Valor Parcela	mini
0	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130025 - SFA/PE/ MAPA	00003/2019	64.799.539/0001-35 - TECNOSET INFORMATICA PROD...	07/05/2019	06/05/2023	R\$ 62.015,76	12	R\$ 5.167,98	Cc Cor Contratos. Min A
1	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130025 - SFA/PE/ MAPA	00006/2023	07.759.174/0001-81 - SOLUCOES SERVICOS DE LOC...	01/06/2023	31/05/2025	R\$ 102.850,00	12	R\$ 8.570,83	Cc Cor Contratos. Min A
2	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130067 - SFA/SP/ MAPA	00006/2023	07.432.517/0001-07 - SIMPRESS COMERCIO LOCACAO...	01/04/2023	01/04/2027	R\$ 720.238,67	48	R\$ 15.004,97	Cc Cor Contratos. Min A
3	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130069 - SFA/MA/ MAPA	00004/2022	08.951.049/0001-31 - CORESMA - COMERCIO DE EQU...	23/09/2022	22/09/2025	R\$ 58.860,00	12	R\$ 4.905,00	Cc Cor Contratos. Min A
4	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130094 - SFA/PA/ MAPA	00002/2023	03.117.534/0001-90 - BRADOK SOLUCOES CORPORATI...	21/11/2023	20/11/2027	R\$ 129.484,80	12	R\$ 10.790,40	Cc Cor Contratos. Min A
5	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130100 - SFA/AP/ MAPA	00002/2022	34.941.930/0001-61 - DIGIMAQ INFORMATICA LTDA	04/08/2022	04/08/2025	R\$ 36.739,20	12	R\$ 3.061,60	Cc Cor Contratos. Min A
6	22000 - MINISTERIO DA	130005 - CGOEF/ DA/SE/	00030/2025	52.997.838/0001-03 - 52.997.838 IDES DE MORAIS...	05/05/2025	05/05/2026	R\$ 17.853,00	1	R\$ 17.853,00	Cc Cor Contratos.

	Órgão	Unidade Gestora	Número Contrato	Fornecedor	Vig. Início	Vig. Fim	Valor Global	Núm. Parcelas	Valor Parcela	mini
	AGRICULTURA E PECUARIA	MAPA								Min A
7	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130005 - CGOEF/ DA/SE/ MAPA	00009/2023	15.330.687/0001-09 - ATOM TECNOLOGIA EM INFORM...	01/07/2023	01/07/2026	R\$ 256.611,00	1	R\$ 256.611,00	Cc Cor Contratos. Min A
8	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130005 - CGOEF/ DA/SE/ MAPA	00003/2024	59.456.277/0003-38 - ORACLE DO BRASIL SISTEMAS...	24/01/2024	24/01/2026	R\$ 5.842.032,72	1	R\$ 5.842.032,72	Cc Cor Contratos. Min A
9	22000 - MINISTERIO DA AGRICULTURA E PECUARIA	130005 - CGOEF/ DA/SE/ MAPA	00003/2023	40.432.544/0001-47 - CLARO S.A.	27/02/2023	27/08/2025	R\$ 12.032.104,50	30	R\$ 401.070,15	Cc Cor Contratos. Min A



Base de Fornecedores:

	CNPJ	Razão Social	Nome Fantasia	Natureza Jurídica	Capital Social	Data de Abertura	Idade da Empresa	Cidade	Estado	CEP	Porte	Atividad Principal
0	113110000160	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
1	258246000168	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
2	306524000377	LEISTUNG COMERCIO E SERVICOS DE SISTEMAS DE EN...	Leistung	Sociedade Empresária Limitada	1230000.0	2013-01-29	12.0	Brasília	DF	70330520.0	Demais	Comércio atacadista de outras máquinas e equip...
3	308141000176	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
4	308141000923	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
5	330845000145	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
6	336701000104	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
7	394528000192	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
8	545482000165	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
9	665620000140	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal

No entanto, consultando o arquivo .csv
 "contratos_ministerios_consolidados_separados.csv" identificamos que:

- 5 contratos tinha início e fim de vigência anterior a 2020 e;
- 75 contratos tinha início de vigência em 2025.

Ambas as situações colocam esses contratos fora do intervalo temporal que buscamos analisar (2020 a 2025). Por isso, devemos excluir essas linhas (contratos) da nossa base.

In []:



Etapa 1: Pré-Processamento e Limpeza Inicial

🎯 Objetivo: Ajustar as datas dos contratos.

Garantir que os valores monetários (Valor Global) estejam como float.

Filtrar apenas o período de 2020 a 2024.

Remover ou lidar com valores faltantes.

Criar variáveis auxiliares como Ano, Mês, UF, Idade Empresa, etc.

📋 Script de Pré-Processamento Inicial python Copy Edit

Pré-Tratamento: Colunas Financeiras e Datas

```
In [69]: import pandas as pd
import numpy as np

# 1. Carregar a base original
df_contratos = pd.read_csv('contratos_ministerios_consolidados_separados.csv', sep=',', encoding='utf-8')

# 2. Função universal de conversão segura
def limpar_valores(valor):
    if pd.isna(valor):
        return np.nan
    if isinstance(valor, str):
        # Remove R$, pontos de milhar e troca vírgula decimal
        valor = valor.replace('R$', '').replace('.', '').replace(',', '.').strip()
    try:
        return float(valor)
    except ValueError:
        return np.nan

# 3. Aplicar a função apenas nas colunas financeiras
df_contratos['Valor Global'] = df_contratos['Valor Global'].apply(limpar_valores)
df_contratos['Valor Parcela'] = df_contratos['Valor Parcela'].apply(limpar_valores)
```

```
# 4. Converter datas
df_contratos['Vig. Início'] = pd.to_datetime(df_contratos['Vig. Início'], errors='coerce', dayfirst=True)
df_contratos['Vig. Fim'] = pd.to_datetime(df_contratos['Vig. Fim'], errors='coerce', dayfirst=True)

# 5. Criar colunas auxiliares
df_contratos['Ano Início'] = df_contratos['Vig. Início'].dt.year
df_contratos['Ano-Mês'] = df_contratos['Vig. Início'].dt.to_period('M').astype(str)

# 6. Visualização prévia
print(df_contratos.dtypes)
print(df_contratos[['Valor Global', 'Valor Parcela']].head())
```

```
Órgão                object
Unidade Gestora      object
Número Contrato      object
Fornecedor          object
Vig. Início         datetime64[ns]
...
ministerio          object
CNPJ                 object
Nome Fornecedor     object
Ano Início           int32
Ano-Mês              object
Length: 14, dtype: object
    Valor Global  Valor Parcela
0      62015.76      5167.98
1     102850.00     8570.83
2     720238.67    15004.97
3     58860.00      4905.00
4    129484.80     10790.40
```

Total de contratos válidos (2020–2024).

Total de fornecedores distintos (CNPJs únicos nos contratos).

Validação se todos os 150 CNPJs da amostra CNPJá estão nos contratos filtrados.

O resultado aponta que 7 fornecedores da nossa amostra da base CNPJá foram excluídos devido a estarem em contratos fora do nosso intervalo de datas alvo.

Trabalharemos com uma amostra de 143 fornecedores para nossas inferências.

- ✓ Base de contratos válidos (2020–2024) com 1008 contratos!
- ✓ 449 fornecedores distintos envolvidos nesses contratos!
- ✓ 143 dos 150 fornecedores da nossa amostra estão na base final (o que é ótimo para métodos inferenciais)!
- ⚠ Apenas 7 fornecedores da amostra fora — o que é natural em amostragens reais!

Próximos passos:

1. Diagnóstico de Qualidade e Estatística Descritiva Final

Explorar as variáveis financeiras (Valor Global, Valor Parcela) e categóricas.

Confirmar ausência de problemas graves de missing.

2. Análise Exploratória de Dados (EDA)

Distribuições, médias, outliers, gráficos.

Gráficos focando:

Volume de contratos por ministério e por fornecedor.

Evolução anual.

Distribuição de valores.

3. Formulação e Teste das Hipóteses (Inferência Estatística)

Testes de hipóteses como:

Diferença de médias (t-test).

Teste de associação (Qui-Quadrado, ANOVA).

Correlação entre variáveis (Spearman, Pearson).

4. Modelagem Multivariada (Se necessário)  

Regressão linear/múltipla.

Modelos de classificação ou clusterização (caso avance para machine learning).

5. Comparação com Bases Externas (CETIC BR) 

Validar tendências do mercado de TIC vs. dados da administração pública.

6. Relatório/Artigo Científico 

Conexão com a teoria de métodos inferenciais.

Apresentação dos achados e interpretações.

Adicionando as colunas "Flag_CNEP" e "Flag_CEIS" à nossa base de 1008 contratos do compras.gov

Criando o novo arquivo 'contratos_1008_com_flags.csv' com esse merge

```
In [81]: # Importar bibliotecas
import pandas as pd

# 1. Carregar as bases
df_contratos = pd.read_csv('contratos_ministerios_consolidados_separados.csv', sep=',', encoding='utf-8')
df_fornecedores_flags = pd.read_csv('fornecedores_formatado.csv', sep=',', encoding='utf-8')

# 2. Padronizar CNPJs
def padronizar_cnpj(cnpj):
    return ''.join(filter(str.isdigit, str(cnpj))).zfill(14)

df_contratos['CNPJ'] = df_contratos['CNPJ'].apply(padronizar_cnpj)
df_fornecedores_flags['CNPJ'] = df_fornecedores_flags['CNPJ'].astype(str).str.zfill(14)

# 3. Tratar datas
```

```
df_contratos['Vig. Início'] = pd.to_datetime(df_contratos['Vig. Início'], errors='coerce', dayfirst=True)
df_contratos['Vig. Fim'] = pd.to_datetime(df_contratos['Vig. Fim'], errors='coerce', dayfirst=True)

# 🚀 4. Filtro de Janela Temporal: 2020-2024
df_contratos_filtrado = df_contratos[
    (df_contratos['Vig. Início'].dt.year <= 2024) & (df_contratos['Vig. Fim'].dt.year >= 2020)
].copy()

# 🚀 5. Selecionar apenas as colunas necessárias
df_flags = df_fornecedores_flags[['CNPJ', 'Flag_CNEP', 'Flag_CEIS']].drop_duplicates()

# 🚀 6. Realizar o merge
df_contratos_com_flags = df_contratos_filtrado.merge(df_flags, on='CNPJ', how='left')

# 🚀 7. Preencher NaN das flags com 0
df_contratos_com_flags['Flag_CNEP'] = df_contratos_com_flags['Flag_CNEP'].fillna(0).astype(int)
df_contratos_com_flags['Flag_CEIS'] = df_contratos_com_flags['Flag_CEIS'].fillna(0).astype(int)

# 🚀 8. Salvar o novo arquivo CSV
df_contratos_com_flags.to_csv('contratos_1008_com_flags.csv', index=False, encoding='utf-8')

# ✅ 9. Confirmação
print(f"✅ Arquivo final gerado com {df_contratos_com_flags.shape[0]} contratos.")
print(df_contratos_com_flags[['CNPJ', 'Flag_CNEP', 'Flag_CEIS']].head())
```

✓ Arquivo final gerado com 1008 contratos.

	CNPJ	Flag_CNEP	Flag_CEIS
0	64799539000135	0	0
1	07759174000181	0	1
2	07432517000107	0	0
3	08951049000131	0	0
4	03117534000190	0	0



1. Diagnóstico de Qualidade e Estatística Descritiva Final

In [83]:

```
# 📈 Importar bibliotecas
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# 1. Carregar o arquivo final consolidado
```

```
df = pd.read_csv('contratos_1008_com_flags.csv', encoding='utf-8')

# 2. Diagnóstico de Qualidade 🖌
missing_info = pd.DataFrame({
    'Coluna': df.columns,
    'Tipo de Dado': df.dtypes.values,
    'Qtd Missing': df.isnull().sum().values,
    '% Missing': (df.isnull().mean().round(4) * 100).values
})

# 3. Exibir Diagnóstico
print("📊 Diagnóstico de Qualidade dos Dados")
display(missing_info.style.background_gradient(subset=['% Missing'], cmap='Reds'))

# 4. Estatística Descritiva ✎
# Separando variáveis numéricas
desc_num = df.describe().transpose()

# Exibir estatística descritiva
print("📈 Estatísticas Descritivas - Variáveis Numéricas")
display(desc_num.style.background_gradient(cmap='Blues'))

# 5. Estatísticas de Categóricas 📈
print("👉 Distribuição de Variáveis Categóricas")
for col in ['Órgão', 'Fornecedor', 'ministerio', 'Flag_CNEP', 'Flag_CEIS']:
    print(f"\n👉 {col}")
    print(df[col].value_counts(dropna=False))
```

📊 Diagnóstico de Qualidade dos Dados

	Coluna	Tipo de Dado	Qtd Missing	% Missing
0	Órgão	object	0	0.000000
1	Unidade Gestora	object	0	0.000000
2	Número Contrato	object	0	0.000000
3	Fornecedor	object	0	0.000000
4	Vig. Início	object	0	0.000000
5	Vig. Fim	object	0	0.000000
6	Valor Global	object	0	0.000000
7	Núm. Parcelas	int64	0	0.000000
8	Valor Parcela	object	0	0.000000
9	ministerio	object	0	0.000000
10	CNPJ	int64	0	0.000000
11	Nome Fornecedor	object	0	0.000000
12	Flag_CNEP	int64	0	0.000000
13	Flag_CEIS	int64	0	0.000000

 Estatísticas Descritivas - Variáveis Numéricas

	count	mean	std	min	25%	50%
Núm. Parcelas	1008.000000	7.032738	13.044950	1.000000	1.000000	1.0000
CNPJ	1008.000000	21551975599346.222656	23293055415725.222656	242160.000000	4602789000101.000000	9238496000100.0000
Flag_CNEP	1008.000000	0.000000	0.000000	0.000000	0.000000	0.0000
Flag_CEIS	1008.000000	0.031746	0.175410	0.000000	0.000000	0.0000

 Distribuição de Variáveis Categóricas Órgão

Órgão

24000 - MINISTERIO DA CIENCIA, TECNOLOGIA E INOVAC	228
36000 - MINISTERIO DA SAUDE	152
25000 - MINISTERIO DA FAZENDA	116
26000 - MINISTERIO DA EDUCACAO	62
52000 - MINISTERIO DA DEFESA	54
...	
47000 - MINISTERIO DO PLANEJAMENTO E ORCAMENTO	8
58000 - MINISTERIO DA PESCA E AQUICULTURA	4
54000 - MINISTERIO DO TURISMO	2
51000 - MINISTERIO DO ESPORTE	1
49000 - MINISTERIO DESENV.AGRARIO E AGRIC FAMILIAR	1

Name: count, Length: 23, dtype: int64

 Fornecedor

Fornecedor

04.198.254/0001-17 - MCR SISTEMAS E CONSULTORIA LTDA	43
33.683.111/0001-07 - SERVICO FEDERAL DE PROCESSAMENTO DE DADOS (SERPRO)	32
57.142.978/0001-05 - BRASOFTWARE INFORMATICA LTDA	23
04.602.789/0001-01 - DATEN TECNOLOGIA LTDA	22
07.432.517/0001-07 - SIMPRESS COMERCIO LOCACAO E SERVICOS LTDA	21
..	
02.543.216/0001-29 - PERFIL COMPUTACIONAL LTDA	1
05.633.420/0001-29 - VENUS WORLD COMERCIO DE EQUIPAMENTOS E MATERIAL PARA ESCRITORIO LTDA	1
04.238.297/0001-89 - 3CORP TECHNOLOGY INFRAESTRUTURA DE TELECOM LTDA.	1
48.411.373/0001-81 - TECHX INFORMATICA LTDA	1
01.628.251/0001-88 - ALUCOM LTDA	1

Name: count, Length: 449, dtype: int64

 ministerio

ministerio

Consulta Contratos Contratos.gov.br Min Ciencia 24000	228
Consulta Contratos Contratos.gov.br Min Saude 36000	152
Consulta Contratos Contratos.gov.br Min Fazenda 25000	116
Consulta Contratos Contratos.gov.br Min Educação 26000	62
Consulta Contratos Contratos.gov.br Min Rel Exteriores 35000	52
...	
Consulta Contratos Contratos.gov.br Min Planej Orc Ges 47000	4
Consulta Contratos Contratos.gov.br Min Planejamento 20113	4

```
Consulta Contratos Contratos.gov.br Min Turismo 54000          2
Consulta Contratos Contratos.gov.br Min Esporte 51000           1
Consulta Contratos Contratos.gov.br Min Des Agrario 49000         1
Name: count, Length: 25, dtype: int64
```

```
📌 Flag_CNEP
```

```
Flag_CNEP
```

```
0    1008
```

```
Name: count, dtype: int64
```

```
📌 Flag_CEIS
```

```
Flag_CEIS
```

```
0     976
```

```
1      32
```

```
Name: count, dtype: int64
```



2. Análise Exploratória de Dados (EDA)



Histograma do valor dos contratos (com escala log).



Evolução do número de contratos por ano.



Volume financeiro total por ano.



Top 15 Ministérios por número de contratos.



Top 15 Fornecedores por volume financeiro de contratos.

Etapa	Base recomendada
EDA (Distribuições, Boxplots, Outliers, Médias)	<input checked="" type="checkbox"/> Base completa e amostra
Testes de Hipóteses	<input checked="" type="checkbox"/> Apenas amostra
Modelos Estatísticos (Regressão, ANOVA, etc.)	<input checked="" type="checkbox"/> Apenas amostra
Validação com bases externas (CETIC.BR, ABES etc.)	<input checked="" type="checkbox"/> Apenas amostra , se possível comparar com indicadores externos

In [107...]

```
# 📦 Importar bibliotecas
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 🚀 1. Carregar os dados
df_contratos = pd.read_csv('contratos_1008_com_flags.csv', sep=',', encoding='utf-8')
df amostra = pd.read_csv('amostra_apenas_150_fornecedores.csv', sep=';', encoding='utf-8')

# ✎ 2. Padronizar CNPJs
def padronizar_cnpj(cnpj):
    return ''.join(filter(str.isdigit, str(cnpj))).zfill(14)

df_contratos['CNPJ_clean'] = df_contratos['CNPJ'].astype(str).apply(padronizar_cnpj)
df amostra['CNPJ_clean'] = df amostra['CNPJ'].astype(str).apply(padronizar_cnpj)

# 🔎 3. Filtrar contratos da amostra
df_amostra_contratos = df_contratos[df_contratos['CNPJ_clean'].isin(df_amostra['CNPJ_clean'])].copy()

# 💰 4. Converter valores monetários com segurança
def converter_valor(valor):
    if isinstance(valor, str):
        valor = valor.replace('R$', '').replace('.', '').replace(',', '.')
    try:
        return float(valor)
    except:
        return 0.0

df_contratos['Valor Global'] = df_contratos['Valor Global'].apply(converter_valor)
df_amostra_contratos['Valor Global'] = df_amostra_contratos['Valor Global'].apply(converter_valor)

# 📊 5. Agrupar e gerar top 15

# Ministérios (todos os contratos)
top_orgaos_qtd = df_contratos['Órgão'].value_counts().head(15)
top_orgaos_valor = df_contratos.groupby('Órgão')['Valor Global'].sum().nlargest(15)

# Fornecedores (todos os contratos)
top_fornecedores_qtd = df_contratos['Nome Fornecedor'].value_counts().head(15)
top_fornecedores_valor = df_contratos.groupby('Nome Fornecedor')['Valor Global'].sum().nlargest(15)

# Fornecedores (amostra)
```

```
top_amostra_fornecedores_qtd = df_amostra_contratos['Nome Fornecedor'].value_counts().head(15)
top_amostra_fornecedores_valor = df_amostra_contratos.groupby('Nome Fornecedor')['Valor Global'].sum().nlargest(15)

# 🎉 6. Exibir resultados
print("\nTOP Top 15 Ministérios por Quantidade de Contratos:")
print(top_orgaos_qtd)

print("\nTOP Top 15 Ministérios por Valor Total (R$):")
print(top_orgaos_valor)

print("\nTOP Top 15 Fornecedores por Quantidade de Contratos:")
print(top_fornecedores_qtd)

print("\nTOP Top 15 Fornecedores por Valor Total (R$):")
print(top_fornecedores_valor)

print("\nTOP Top 15 Fornecedores da Amostra por Quantidade de Contratos:")
print(top_amostra_fornecedores_qtd)

print("\nTOP Top 15 Fornecedores da Amostra por Valor Total (R$):")
print(top_amostra_fornecedores_valor)
```

 Top 15 Ministérios por Quantidade de Contratos:

Órgão

24000 - MINISTERIO DA CIENCIA, TECNOLOGIA E INOVAC	228
36000 - MINISTERIO DA SAUDE	152
25000 - MINISTERIO DA FAZENDA	116
26000 - MINISTERIO DA EDUCACAO	62
52000 - MINISTERIO DA DEFESA	54
...	
81000 - MINISTERIO DOS DIREITOS HUMANOS E CIDADANI	33
55000 - MIN. DESENV.E ASSIT.SOCIAL,FAM.E COMBATE FO	27
42000 - MINISTERIO DA CULTURA	24
53000 - MINIST. DA INTEGR. E DO DESENVOLV. REGIONA	21
44000 - MINIST. DO MEIO AMBIENTE E MUDANCA DO CLIM	17

Name: count, Length: 15, dtype: int64

 Top 15 Ministérios por Valor Total (R\$):

Órgão

41000 - MINISTERIO DAS COMUNICACOES	3.214012e+09
25000 - MINISTERIO DA FAZENDA	2.331131e+09
40000 - MINISTERIO DO TRABALHO E EMPREGO	1.103023e+09
36000 - MINISTERIO DA SAUDE	7.862045e+08
39000 - MINISTERIO DOS TRANSPORTES	4.434386e+08
...	
35000 - MINISTERIO DAS RELACOES EXTERIORES	1.574251e+08
52000 - MINISTERIO DA DEFESA	6.545486e+07
81000 - MINISTERIO DOS DIREITOS HUMANOS E CIDADANI	5.423958e+07
44000 - MINIST. DO MEIO AMBIENTE E MUDANCA DO CLIM	5.219403e+07
53000 - MINIST. DA INTEGR. E DO DESENVOLV. REGIONA	5.204011e+07

Name: Valor Global, Length: 15, dtype: float64

 Top 15 Fornecedores por Quantidade de Contratos:

Nome Fornecedor

MCR SISTEMAS E CONSULTORIA LTDA	43
SERVICO FEDERAL DE PROCESSAMENTO DE DADOS (SERPRO)	34
POSITIVO TECNOLOGIA S.A.	23
BRASOFTWARE INFORMATICA LTDA	23
DATEN TECNOLOGIA LTDA	22
..	
LAYER TECNOLOGIA DA INFORMACAO LTDA	12
EMPRESA DE TECNOLOGIA E INFORMACOES DA PREVIDENCIA S.A. - DATAPREV	12
CLARO S.A.	11
PERFIL COMPUTACIONAL LTDA	11

CENTRAL IT TECNOLOGIA DA INFORMACAO S/A

10

Name: count, Length: 15, dtype: int64

 Top 15 Fornecedores por Valor Total (R\$):

Nome Fornecedor

TELECOMUNICACOES BRASILEIRAS SA TELEBRAS	3.199984e+09
SERVICO FEDERAL DE PROCESSAMENTO DE DADOS (SERPRO)	1.603927e+09
EMPRESA DE TECNOLOGIA E INFORMACOES DA PREVIDENCIA S.A. - DATAPREV	1.293257e+09
SERPRO - SEDE - BRASILIA	7.850108e+08
BRASOFTWARE INFORMATICA LTDA	2.316736e+08
	...

DATEN TECNOLOGIA LTDA

6.688642e+07

RESOURCE TECNOLOGIA E INFORMATICA LTDA.

4.905684e+07

DIGISYSTEM SERVICOS ESPECIALIZADOS LTDA

4.879315e+07

HEPTA TECNOLOGIA E INFORMATICA LTDA

4.353070e+07

G4F SOLUCOES CORPORATIVAS LTDA

4.332259e+07

Name: Valor Global, Length: 15, dtype: float64

 Top 15 Fornecedores da Amostra por Quantidade de Contratos:

Nome Fornecedor

SIMPRESS COMERCIO LOCACAO E SERVICOS LTDA	21
XP ON CONSULTORIA LTDA	9
NORTHWARE COMERCIO E SERVICOS LTDA	9
ISH TECNOLOGIA S/A	6
NCT INFORMATICA LTDA	5
	..

WETALK TECNOLOGIA DA INFORMACAO LTDA

4

BRADOK SOLUCOES CORPORATIVAS LTDA

4

JAMC CONSULTORIA E REPRESENTACAO DE SOFTWARE LTDA

4

AMM TECNOLOGIA E SERVICOS DE INFORMATICA S/A

4

NP TECNOLOGIA E GESTAO DE DADOS LTDA

4

Name: count, Length: 15, dtype: int64

 Top 15 Fornecedores da Amostra por Valor Total (R\$):

Nome Fornecedor

DIGISYSTEM SERVICOS ESPECIALIZADOS LTDA	48793151.28
LANLINK SOLUCOES E COMERCIALIZACAO EM INFORMATICA S/A	34027809.10
ISH TECNOLOGIA S/A	27504282.99
SYDLE SISTEMAS LTDA	24884852.17
SIMPRESS COMERCIO LOCACAO E SERVICOS LTDA	24692690.04
	..
NORTHWARE COMERCIO E SERVICOS LTDA	16577000.00

AMM TECNOLOGIA E SERVICOS DE INFORMATICA S/A	14410738.59
JAMC CONSULTORIA E REPRESENTACAO DE SOFTWARE LTDA	13536400.00
RJR SERVICOS DE INFORMATICA LTDA	12985240.00
GLOBALWEB OUTSOURCING DO BRASIL S.A.	12749196.48
Name: Valor Global, Length: 15, dtype: float64	

```
In [99]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Configurar estilo de gráficos
sns.set(style="whitegrid")

# Carregar as bases
df_contratos = pd.read_csv("contratos_1008_com_flags.csv", sep=',', encoding='utf-8')
df amostra = pd.read_csv("amostra_apenas_150_fornecedores.csv", sep=';', encoding='utf-8')

# Padronizar CNPJs para realizar o merge
def padronizar_cnpj(cnpj):
    return ''.join(filter(str.isdigit, str(cnpj))).zfill(14)

df_contratos['CNPJ_clean'] = df_contratos['CNPJ'].astype(str).apply(padronizar_cnpj)
df amostra['CNPJ_clean'] = df amostra['CNPJ'].astype(str).apply(padronizar_cnpj)

# Filtrar contratos da amostra
df_amostra_contratos = df_contratos[df_contratos['CNPJ_clean'].isin(df_amostra['CNPJ_clean'])].copy()

print(f"✅ Número de contratos da amostra encontrados: {df_amostra_contratos.shape[0]}")
print(f"✅ Fornecedores únicos da amostra encontrados: {df_amostra_contratos['CNPJ_clean'].nunique()}")
```

- ✓ Número de contratos da amostra encontrados: 248
- ✓ Fornecedores únicos da amostra encontrados: 143

Evolução anual da quantidade de contratos e valores contratados

```
In [106...]: import pandas as pd
import matplotlib.pyplot as plt

# ✅ Carregue o dataframe final
df = pd.read_csv("contratos_1008_com_flags.csv", encoding='utf-8')
```

```
# ✅ Converta datas corretamente
df['Vig. Início'] = pd.to_datetime(df['Vig. Início'], errors='coerce', dayfirst=True)
df['Vig. Fim'] = pd.to_datetime(df['Vig. Fim'], errors='coerce', dayfirst=True)

# ✅ Tratar valores monetários
def converter_valor(valor):
    if isinstance(valor, str):
        valor = valor.replace('R$', '').replace('.', '').replace(',', '.')
    try:
        return float(valor)
    except:
        return 0.0

df['Valor Global'] = df['Valor Global'].apply(converter_valor)

# ✅ Criar DataFrame de evolução anual dos contratos vigentes
anos_analise = range(2020, 2025)
dados = []

for ano in anos_analise:
    contratos_ativos = df[
        (df['Vig. Início'].dt.year <= ano) &
        (df['Vig. Fim'].dt.year >= ano)
    ]
    total_contratos = contratos_ativos.shape[0]
    total_valor = contratos_ativos['Valor Global'].sum()
    dados.append({
        'Ano': ano,
        'Contratos Vigentes': total_contratos,
        'Valor Total (R$)': total_valor
    })

df_evolucao = pd.DataFrame(dados)
df_evolucao['Média Móvel Valor (R$)'] = df_evolucao['Valor Total (R$)'].rolling(window=2).mean()

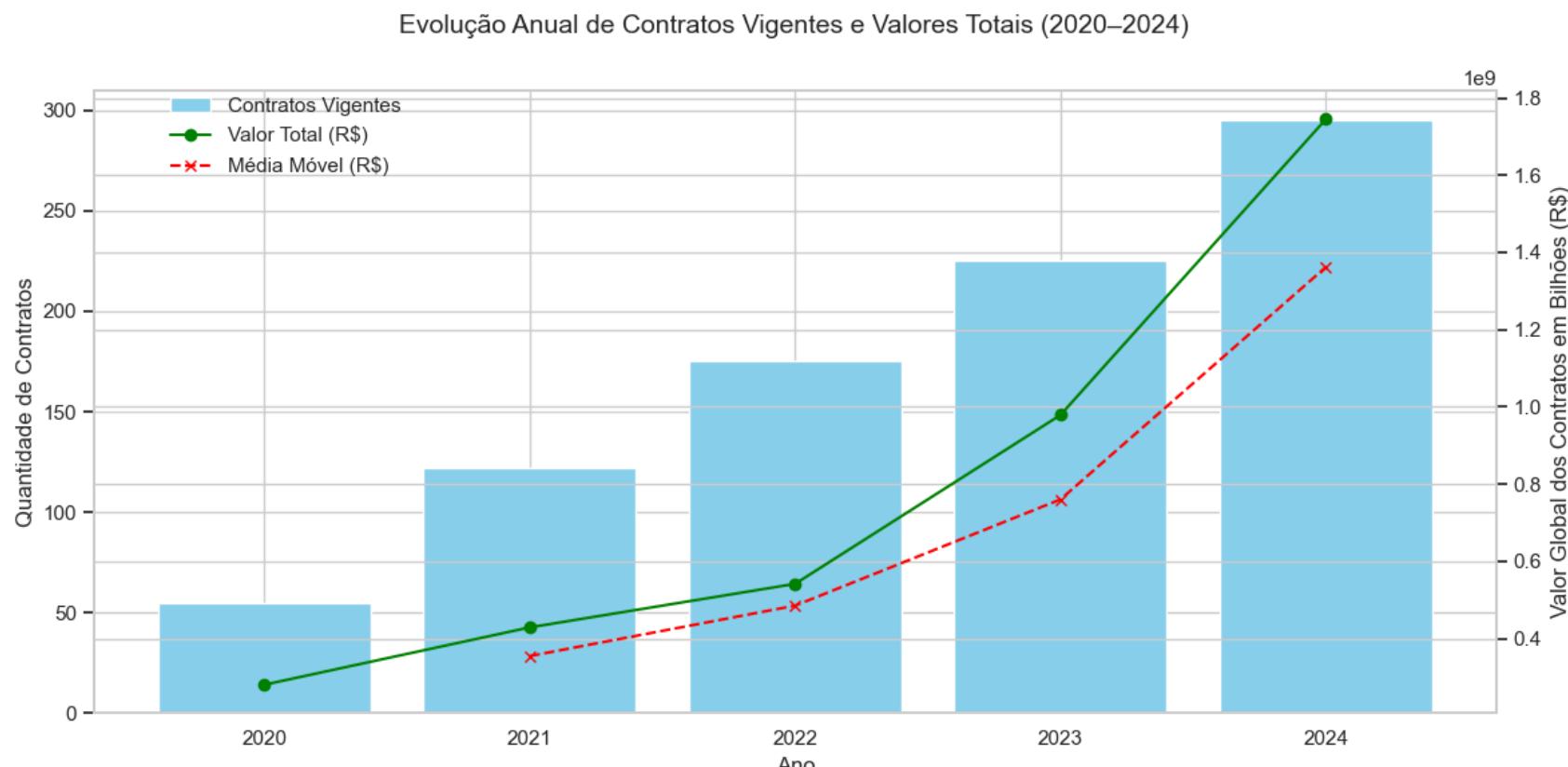
# ✅ Plotar gráfico com dois eixos Y
fig, ax1 = plt.subplots(figsize=(12, 6))

# Barras - Contratos vigentes
ax1.bar(df_evolucao['Ano'], df_evolucao['Contratos Vigentes'], color='skyblue', label='Contratos Vigentes')
ax1.set_ylabel('Quantidade de Contratos', fontsize=12)
ax1.set_xlabel('Ano', fontsize=12)
```

```
ax1.tick_params(axis='y')

# Segundo eixo Y para valores
ax2 = ax1.twinx()
ax2.plot(df_evolucao['Ano'], df_evolucao['Valor Total (R$)'], color='green', marker='o', label='Valor Total (R$)')
ax2.plot(df_evolucao['Ano'], df_evolucao['Média Móvel Valor (R$)'], color='red', linestyle='--', marker='x', label='Média Móvel Valor (R$)')
ax2.set_ylabel('Valor Global dos Contratos em Bilhões (R$)', fontsize=12)
ax2.tick_params(axis='y')

# Legendas e título
fig.suptitle('Evolução Anual de Contratos Vigentes e Valores Totais (2020-2024)', fontsize=14)
fig.legend(loc='upper left', bbox_to_anchor=(0.1, 0.9))
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [117...]

```
# Tratar datas
df_contratos['Vig. Início'] = pd.to_datetime(df_contratos['Vig. Início'], errors='coerce', dayfirst=True)
df_contratos['Vig. Fim'] = pd.to_datetime(df_contratos['Vig. Fim'], errors='coerce', dayfirst=True)

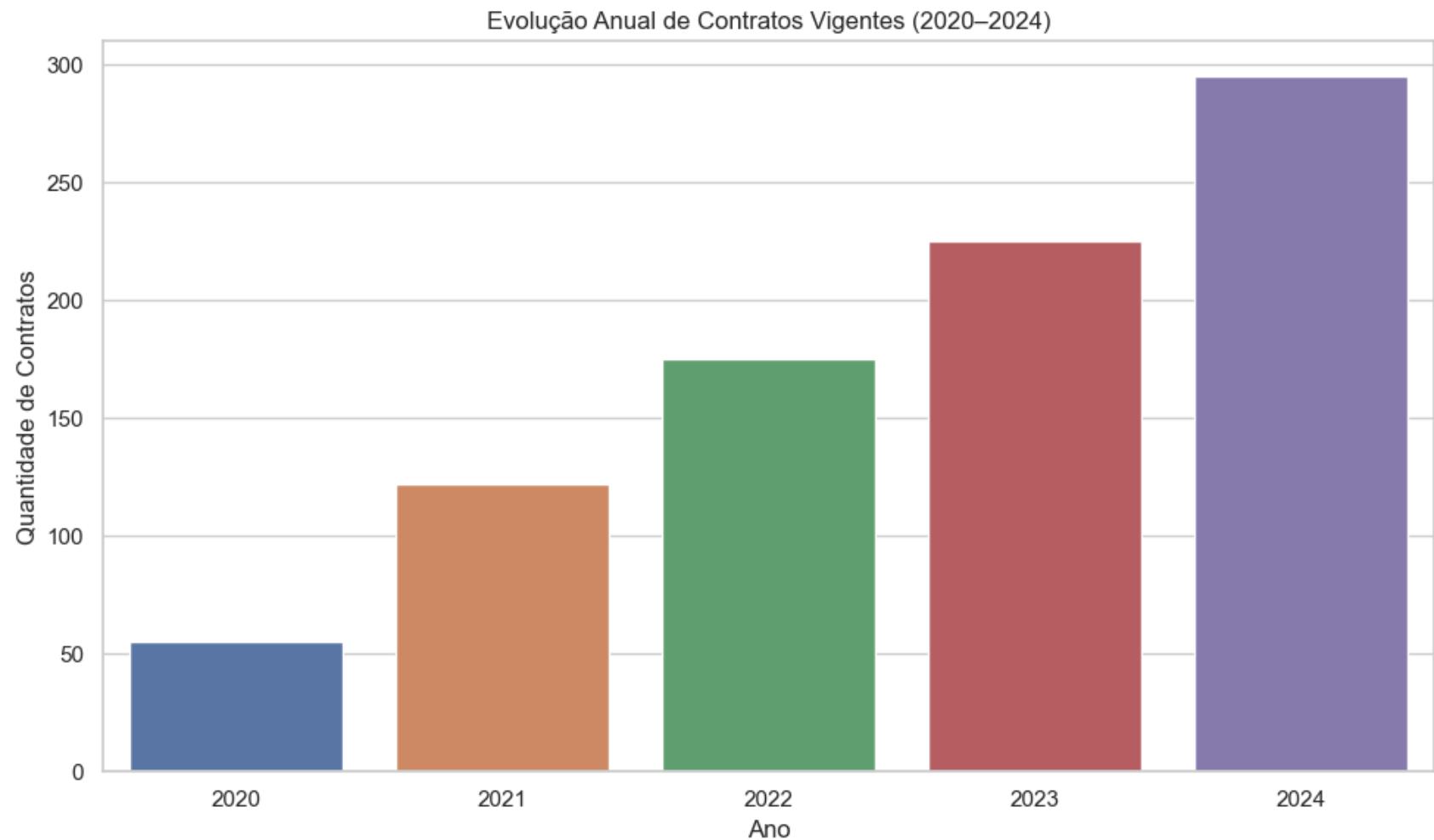
# Criar colunas de vigência por ano
anos_analise = list(range(2020, 2025))
dados_por_ano = {
    'Ano': [],
    'Qtd Contratos Vigentes': [],
    'Valor Global Total': []
}

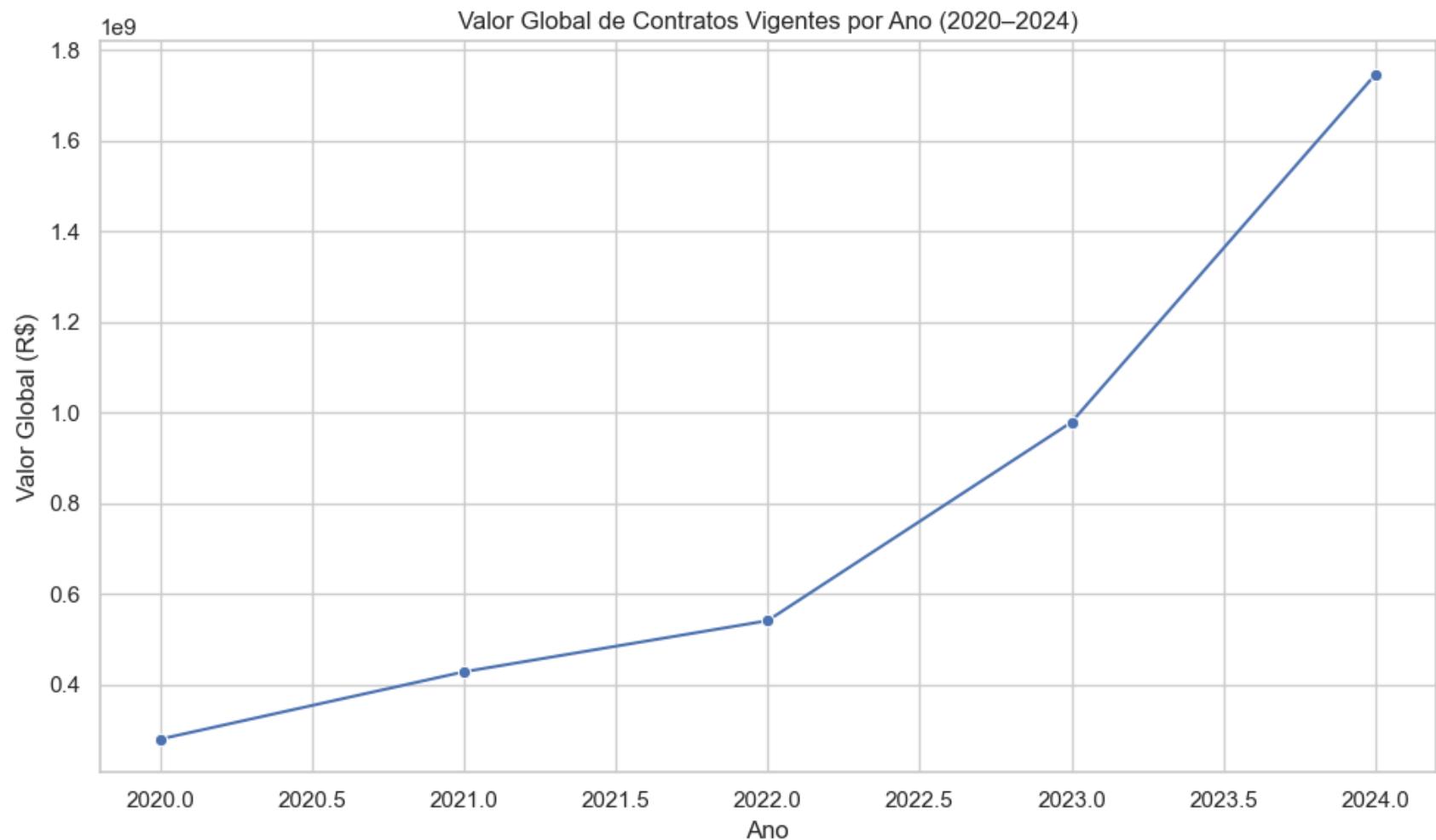
for ano in anos_analise:
    vigentes = df_contratos[
        (df_contratos['Vig. Início'].dt.year <= ano) &
        (df_contratos['Vig. Fim'].dt.year >= ano)
    ]
    dados_por_ano['Ano'].append(ano)
    dados_por_ano['Qtd Contratos Vigentes'].append(vigentes.shape[0])
    dados_por_ano['Valor Global Total'].append(vigentes['Valor Global'].sum())

df_evolucao = pd.DataFrame(dados_por_ano)

# Plotar gráfico de barras
plt.figure(figsize=(10, 6))
sns.barplot(x='Ano', y='Qtd Contratos Vigentes', data=df_evolucao)
plt.title("Evolução Anual de Contratos Vigentes (2020-2024)")
plt.ylabel("Quantidade de Contratos")
plt.tight_layout()
plt.show()

# Plotar gráfico de Linha com valor global
plt.figure(figsize=(10, 6))
sns.lineplot(x='Ano', y='Valor Global Total', data=df_evolucao, marker='o')
plt.title("Valor Global de Contratos Vigentes por Ano (2020-2024)")
plt.ylabel("Valor Global (R$)")
plt.tight_layout()
plt.show()
```





Evolução Anual de Contratos (Quantidade + Valor + Média Móvel) – Amostra

In [121...]

```
import pandas as pd
import matplotlib.pyplot as plt

# ✅ Garantir datas no formato correto
df amostra_contratos['Vig. Início'] = pd.to_datetime(df amostra_contratos['Vig. Início'], errors='coerce', dayfirst=True)
df amostra_contratos['Vig. Fim'] = pd.to_datetime(df amostra_contratos['Vig. Fim'], errors='coerce', dayfirst=True)

# 🗓️ Definir anos corretamente de 2020 a 2024
```

```
anos = [2020, 2021, 2022, 2023, 2024]

# 📈 Gerar tabela com evolução anual
evolucao_amostra = []

for ano in anos:
    vigentes = df_amostra_contratos[
        (df_amostra_contratos['Vig. Início'].dt.year <= ano) &
        (df_amostra_contratos['Vig. Fim'].dt.year >= ano)
    ]
    evolucao_amostra.append({
        'Ano': ano,
        'Qtd Contratos': vigentes.shape[0],
        'Valor Total R$': vigentes['Valor Global'].sum()
    })

# Criar DataFrame de evolução
df_evolucao_amostra = pd.DataFrame(evolucao_amostra)

# Calcular média móvel dos valores (janela = 2 anos)
df_evolucao_amostra['Média Móvel R$'] = df_evolucao_amostra['Valor Total R$'].rolling(window=2).mean()

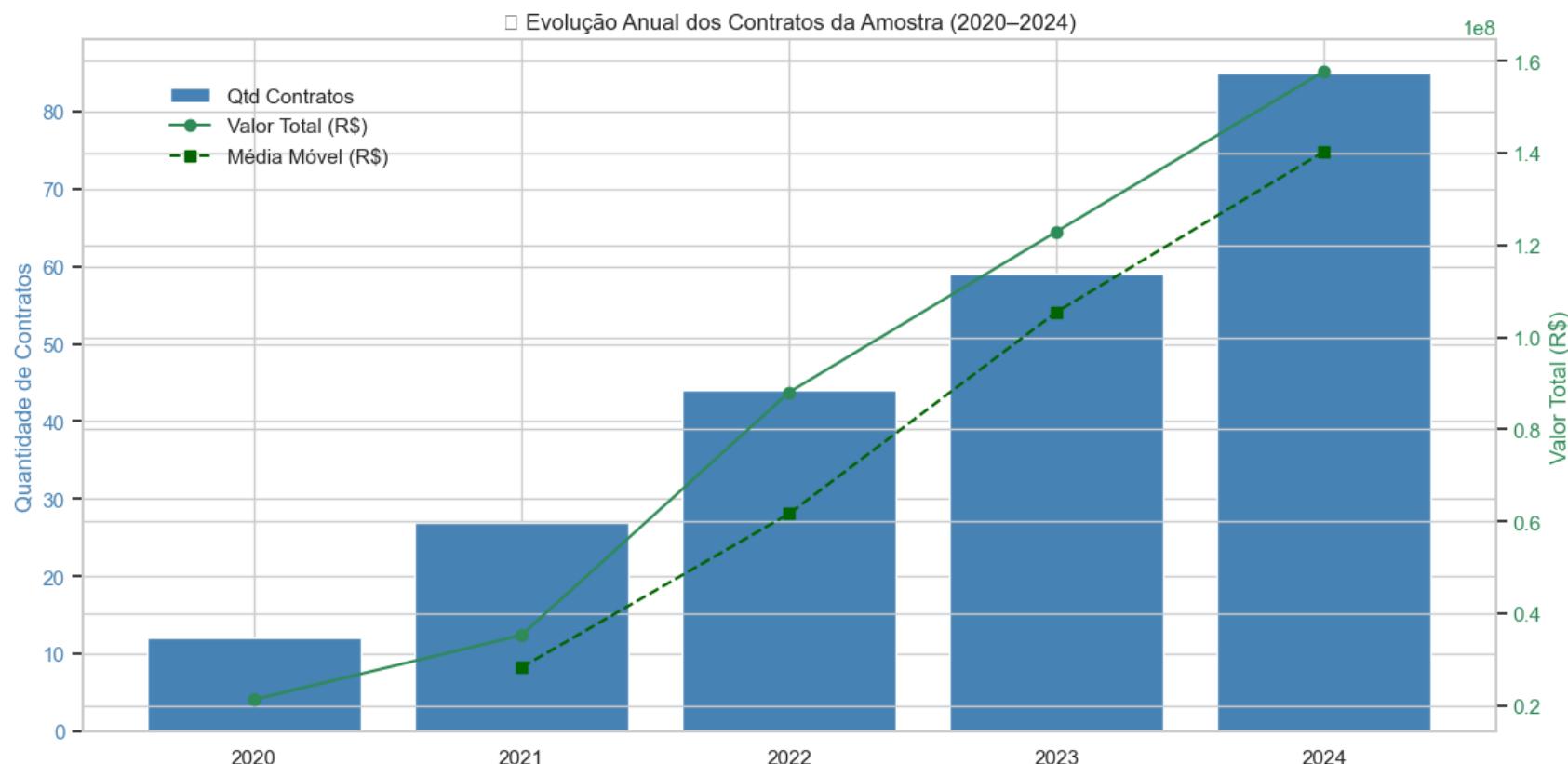
# 📈 Plotagem com dois eixos
fig, ax1 = plt.subplots(figsize=(12, 6))

# Eixo esquerdo - Quantidade de contratos
ax1.bar(df_evolucao_amostra['Ano'], df_evolucao_amostra['Qtd Contratos'],
         color='steelblue', label='Qtd Contratos')
ax1.set_ylabel("Quantidade de Contratos", color='steelblue')
ax1.tick_params(axis='y', labelcolor='steelblue')

# Eixo direito - Valores contratados e média móvel
ax2 = ax1.twinx()
ax2.plot(df_evolucao_amostra['Ano'], df_evolucao_amostra['Valor Total R$'],
          color='seagreen', marker='o', label='Valor Total (R$)')
ax2.plot(df_evolucao_amostra['Ano'], df_evolucao_amostra['Média Móvel R$'],
          color='darkgreen', linestyle='--', marker='s', label='Média Móvel (R$)')
ax2.set_ylabel("Valor Total (R$)", color='seagreen')
ax2.tick_params(axis='y', labelcolor='seagreen')

# 🎯 Título e legenda
plt.title("📊 Evolução Anual dos Contratos da Amostra (2020-2024)")
```

```
fig.legend(loc='upper left', bbox_to_anchor=(0.1, 0.9))
plt.tight_layout()
plt.show()
```



In []:

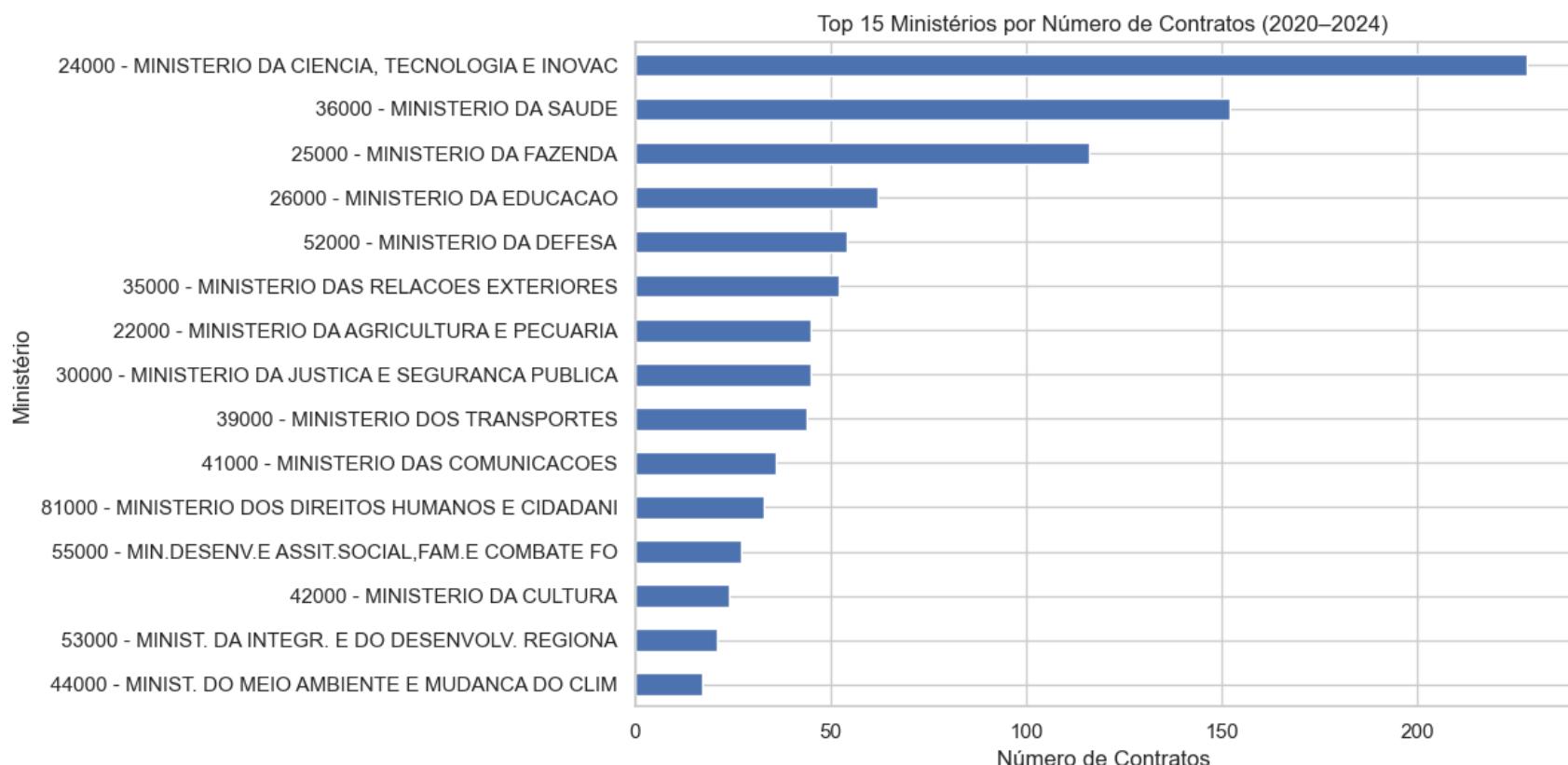
Top 15 Ministérios por Quantidade de Contratos

In [108...]

```
import matplotlib.pyplot as plt
import seaborn as sns

# Gráfico 1 - Top 15 Ministérios por Número de Contratos
plt.figure(figsize=(12, 6))
df_contratos['Órgão'].value_counts().head(15).sort_values().plot(kind='barh')
plt.title("Top 15 Ministérios por Número de Contratos (2020-2024)")
```

```
plt.xlabel("Número de Contratos")
plt.ylabel("Ministério")
plt.tight_layout()
plt.show()
```

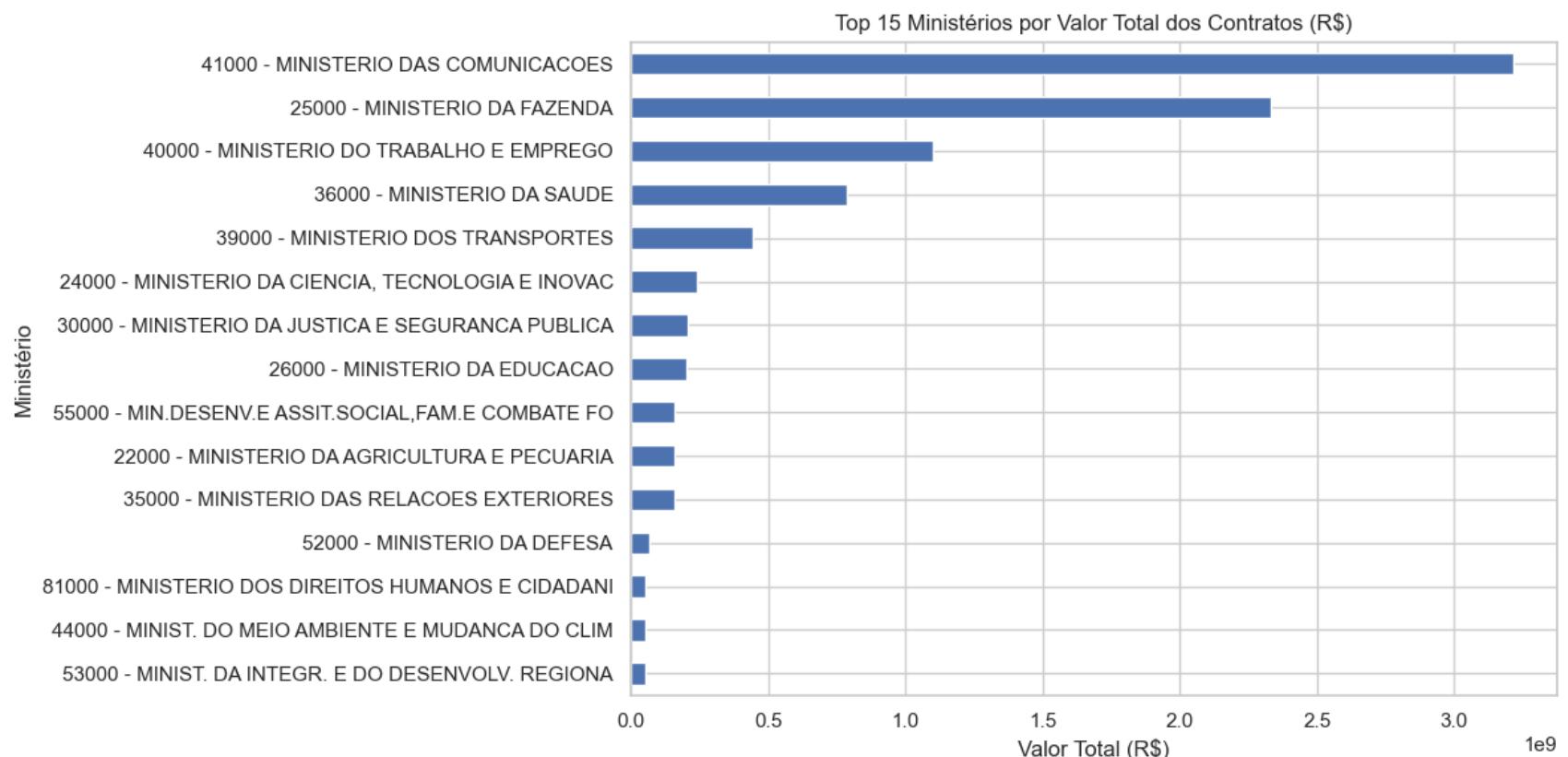


Top 15 Ministérios por Valor Total (R\$)

```
In [109...]: # Gráfico 2 – Top 15 Ministérios por Valor Total dos Contratos
top_orgaos_valor = (
    df_contratos.groupby('Órgão')['Valor Global'].sum()
    .sort_values(ascending=False)
    .head(15)
)

plt.figure(figsize=(12, 6))
top_orgaos_valor.sort_values().plot(kind='barh')
```

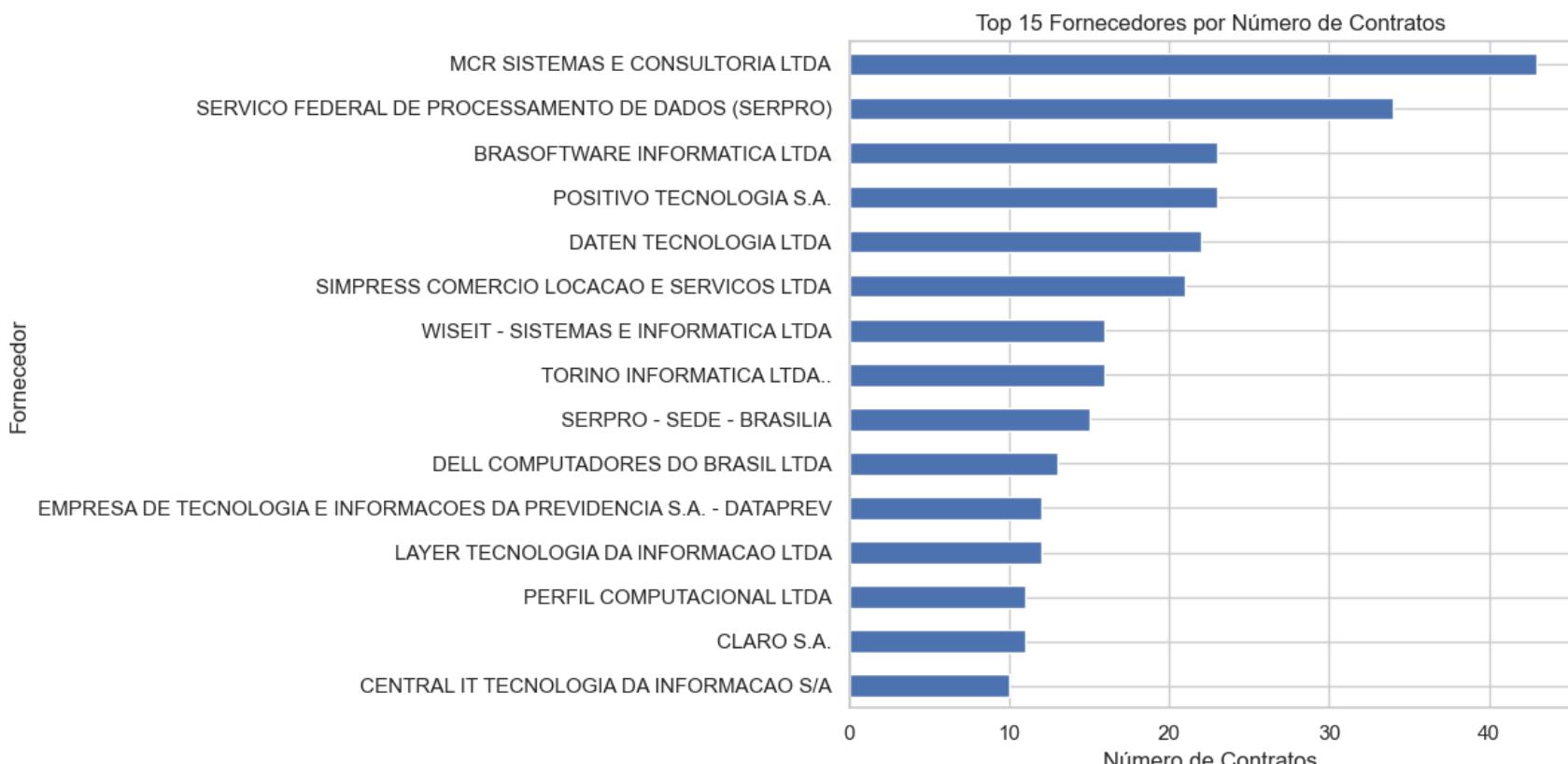
```
plt.title("Top 15 Ministérios por Valor Total dos Contratos (R$)")  
plt.xlabel("Valor Total (R$)")  
plt.ylabel("Ministério")  
plt.tight_layout()  
plt.show()
```



Top 15 Fornecedores (Todos os contratos) – Quantidade

In [110...]

```
# Gráfico 3 - Top 15 Fornecedores por Número de Contratos
plt.figure(figsize=(12, 6))
df_contratos['Nome Fornecedor'].value_counts().head(15).sort_values().plot(kind='barh')
plt.title("Top 15 Fornecedores por Número de Contratos")
plt.xlabel("Número de Contratos")
plt.ylabel("Fornecedor")
plt.tight_layout()
plt.show()
```

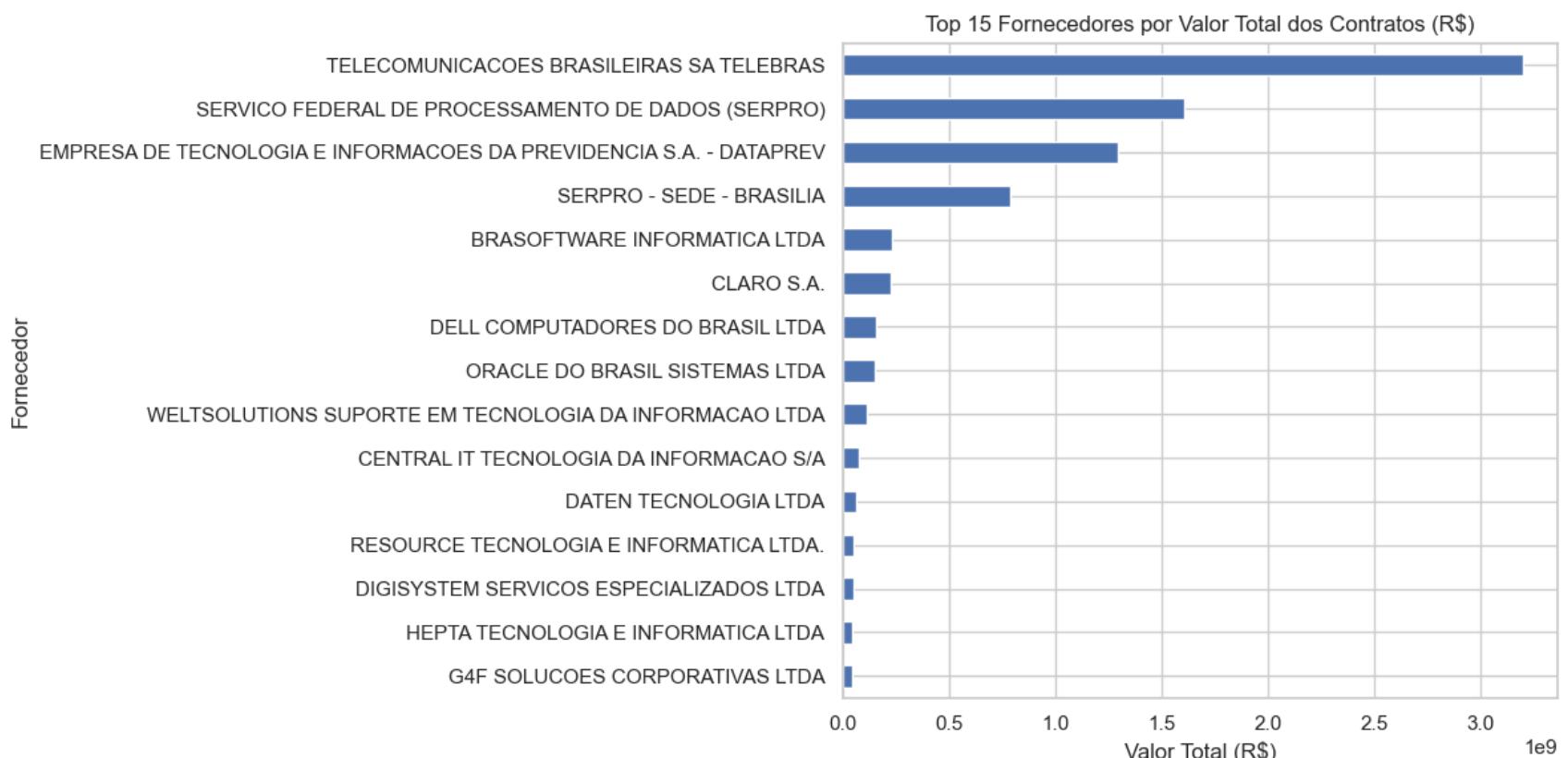


Top 15 Fornecedores (Todos os contratos) – Valor Total

In [111...]

```
# Gráfico 4 - Top 15 Fornecedores por Valor Total dos Contratos
top_fornecedores_valor = (
    df_contratos.groupby('Nome Fornecedor')['Valor Global'].sum()
    .sort_values(ascending=False)
```

```
.head(15)  
)  
  
plt.figure(figsize=(12, 6))  
top_fornecedores_valor.sort_values().plot(kind='barh')  
plt.title("Top 15 Fornecedores por Valor Total dos Contratos (R$)")  
plt.xlabel("Valor Total (R$)")  
plt.ylabel("Fornecedor")  
plt.tight_layout()  
plt.show()
```

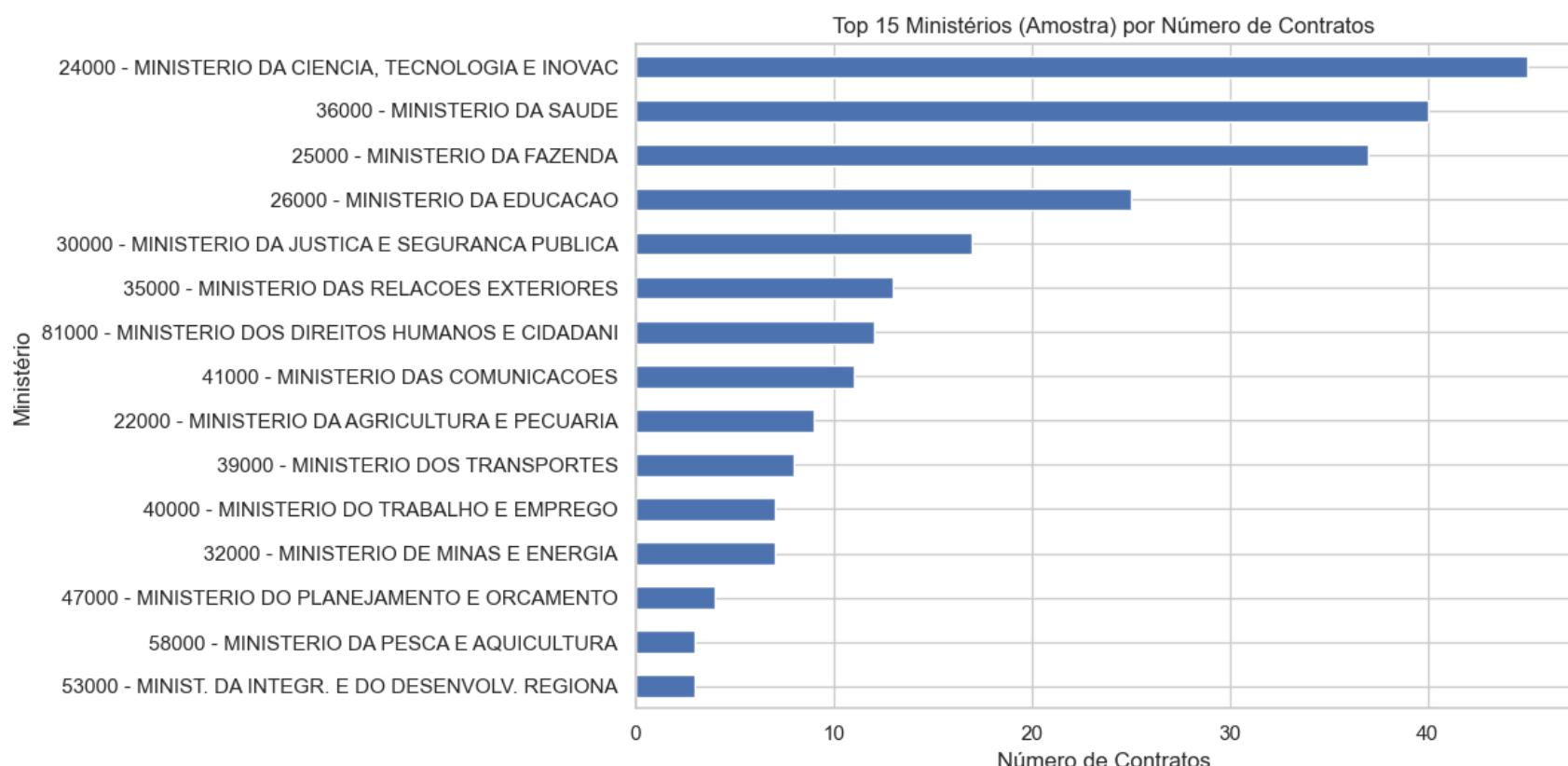


Top 15 Ministérios (Amostra) por Quantidade de Contratos

```
In [112]: # Gráfico - Top 15 Ministérios da Amostra por Número de Contratos  
import matplotlib.pyplot as plt
```

```
top_orgaos_amostra_qtd = (
    df_amostra_contratos['Órgão']
    .value_counts()
    .head(15)
    .sort_values()
)

plt.figure(figsize=(12, 6))
top_orgaos_amostra_qtd.plot(kind='barh')
plt.title("Top 15 Ministérios (Amostra) por Número de Contratos")
plt.xlabel("Número de Contratos")
plt.ylabel("Ministério")
plt.tight_layout()
plt.show()
```

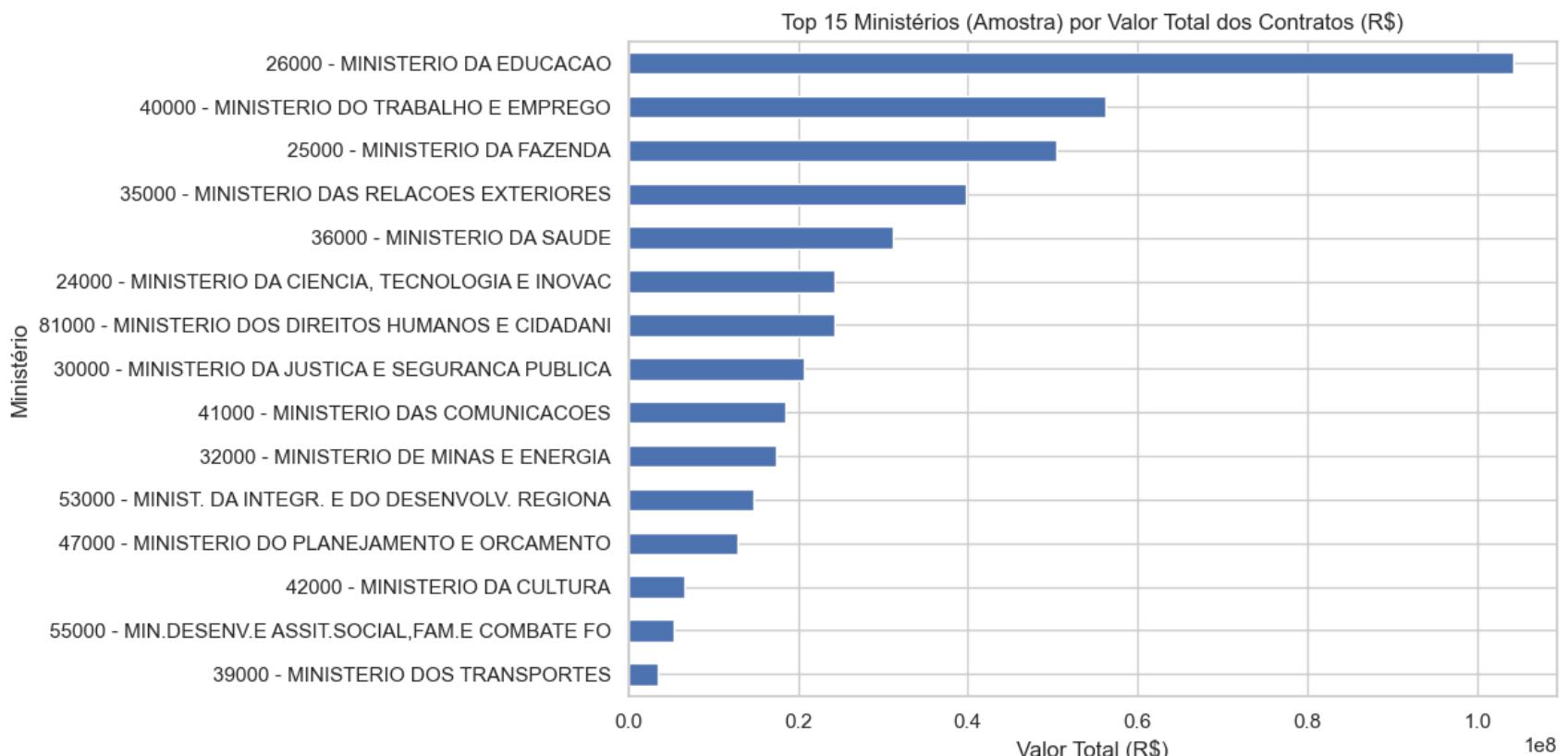


Top 15 Ministérios (Amostra) por Valor Total dos Contratos

In [113...]

```
# Gráfico - Top 15 Ministérios da Amostra por Valor Total dos Contratos
top_orgaos_amostra_valor = (
    df amostra_contratos
    .groupby('Órgão')['Valor Global']
    .sum()
    .sort_values(ascending=False)
    .head(15)
)

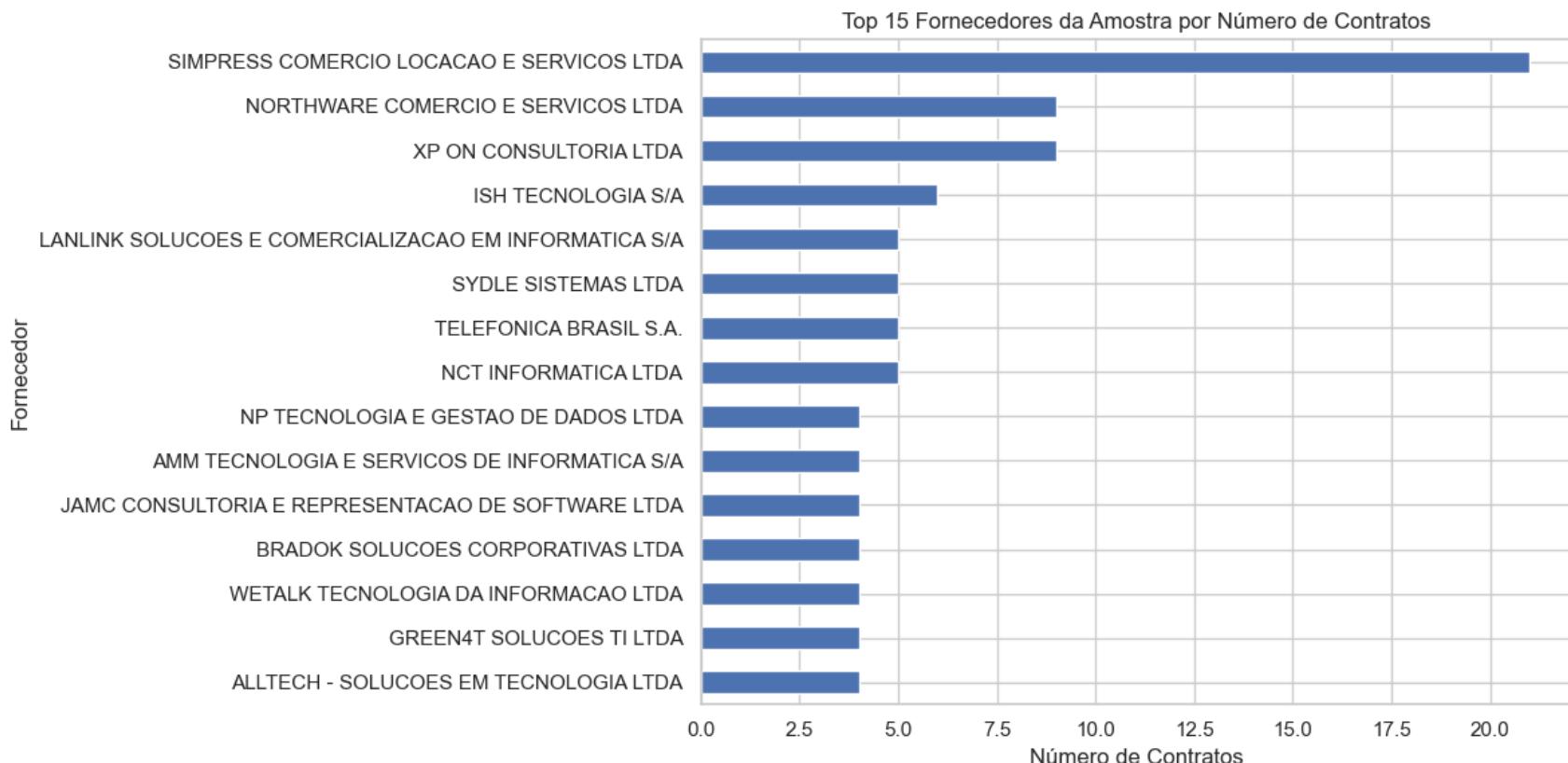
plt.figure(figsize=(12, 6))
top_orgaos_amostra_valor.sort_values().plot(kind='barh')
plt.title("Top 15 Ministérios (Amostra) por Valor Total dos Contratos (R$)")
plt.xlabel("Valor Total (R$)")
plt.ylabel("Ministério")
plt.tight_layout()
plt.show()
```



Top 15 Fornecedores da Amostra – Quantidade

In [115...]

```
# Gráfico 5 – Top 15 Fornecedores da Amostra por Número de Contratos
plt.figure(figsize=(12, 6))
df amostra_contratos['Nome Fornecedor'].value_counts().head(15).sort_values().plot(kind='barh')
plt.title("Top 15 Fornecedores da Amostra por Número de Contratos")
plt.xlabel("Número de Contratos")
plt.ylabel("Fornecedor")
plt.tight_layout()
plt.show()
```

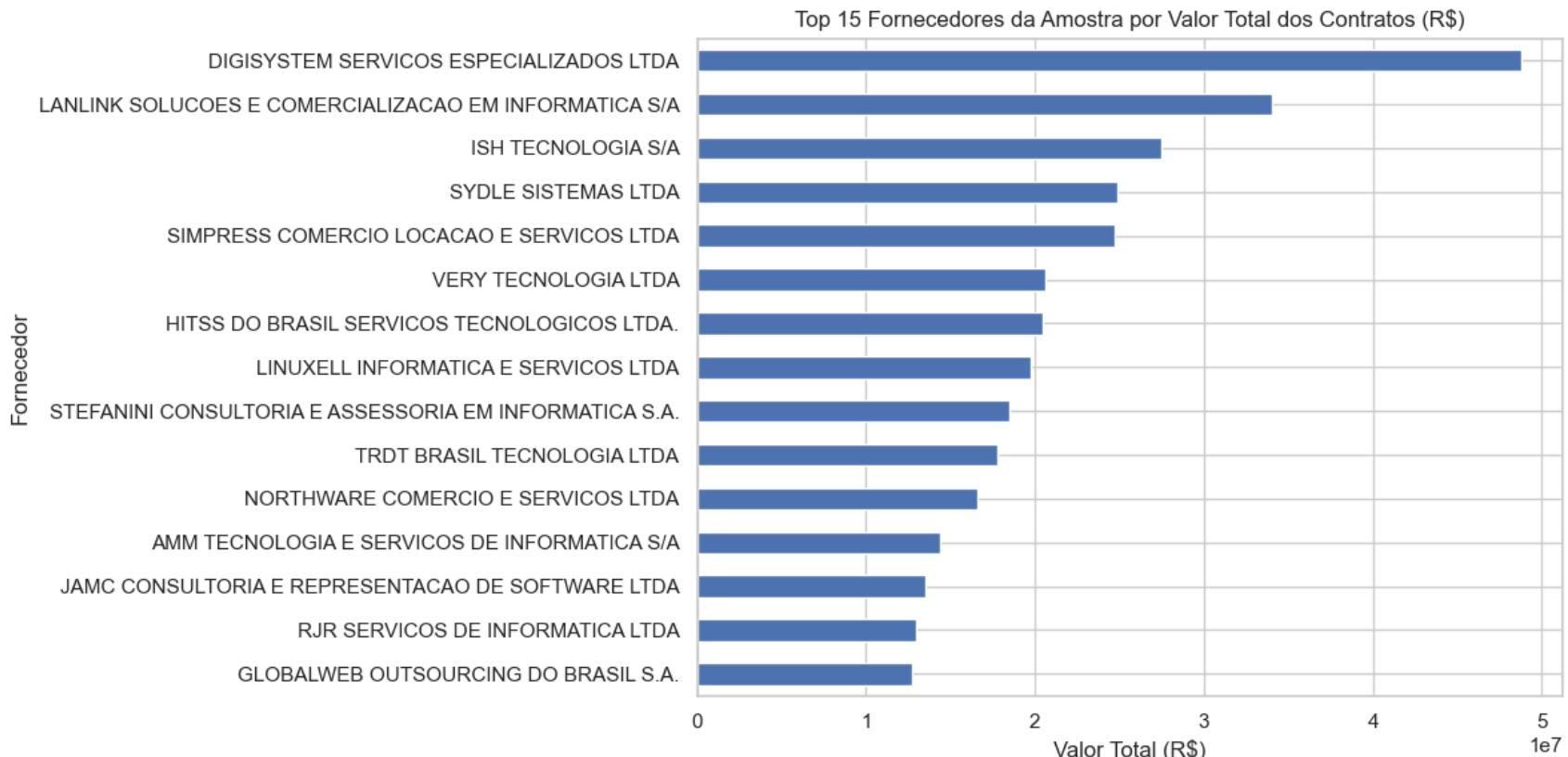


Top 15 Fornecedores da Amostra – Valor Total

```
In [116]: # Gráfico 6 – Top 15 Fornecedores da Amostra por Valor Total dos Contratos
top_amostra_valor = (
    df_amostra_contratos.groupby('Nome Fornecedor')['Valor Global'].sum()
    .sort_values(ascending=False)
    .head(15)
)

plt.figure(figsize=(12, 6))
top_amostra_valor.sort_values().plot(kind='barh')
plt.title("Top 15 Fornecedores da Amostra por Valor Total dos Contratos (R$)")
plt.xlabel("Valor Total (R$)")
plt.ylabel("Fornecedor")
plt.tight_layout()
```

```
plt.show()
```



Análise Exploratória de Dados para a Amostra de 150 fornecedores

1. Distribuição de Fornecedores por Estado (UF):

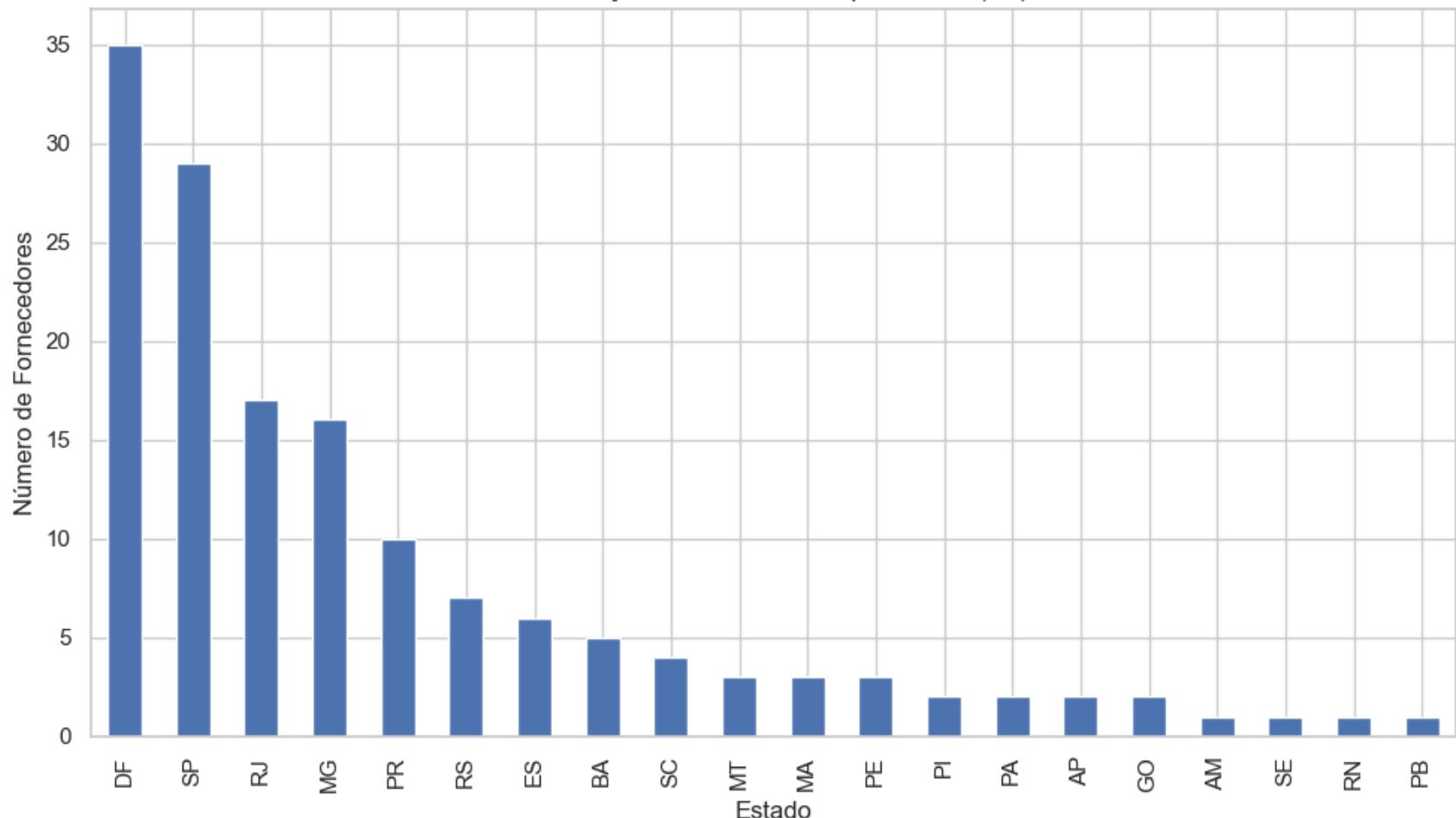
In [186...]

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
df amostra['Estado'].value_counts().sort_values(ascending=False).plot(kind='bar')
plt.title('Distribuição de Fornecedores por Estado (UF)')
plt.xlabel('Estado')
plt.ylabel('Número de Fornecedores')
plt.tight_layout()
```

```
plt.show()
```

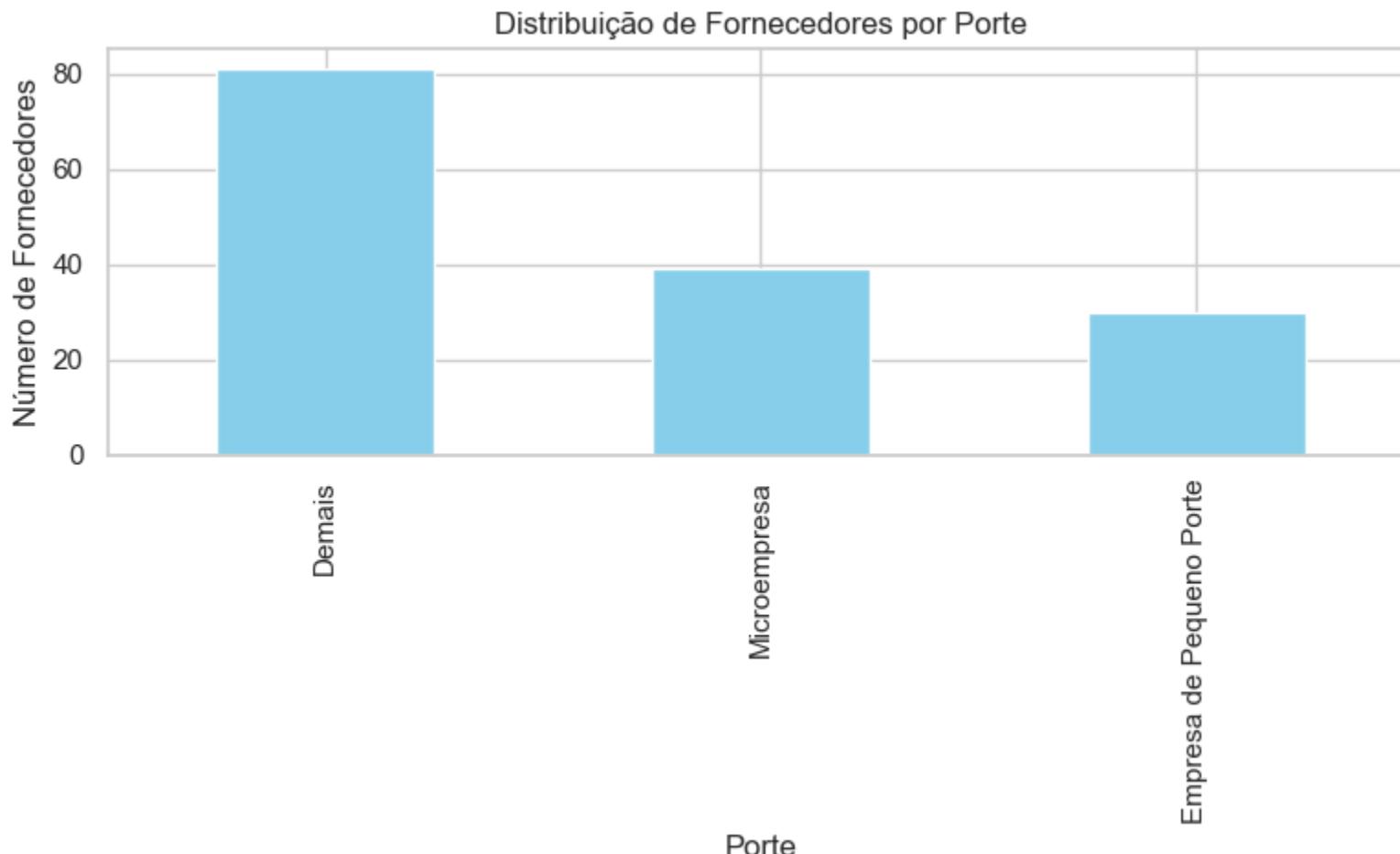
Distribuição de Fornecedores por Estado (UF)



2. Distribuição de Fornecedores por Porte:

```
In [187]:  
plt.figure(figsize=(8,5))  
df amostra['Porte'].value_counts().sort_values(ascending=False).plot(kind='bar', color='skyblue')  
plt.title('Distribuição de Fornecedores por Porte')  
plt.xlabel('Porte')  
plt.ylabel('Número de Fornecedores')  
plt.tight_layout()
```

```
plt.show()
```

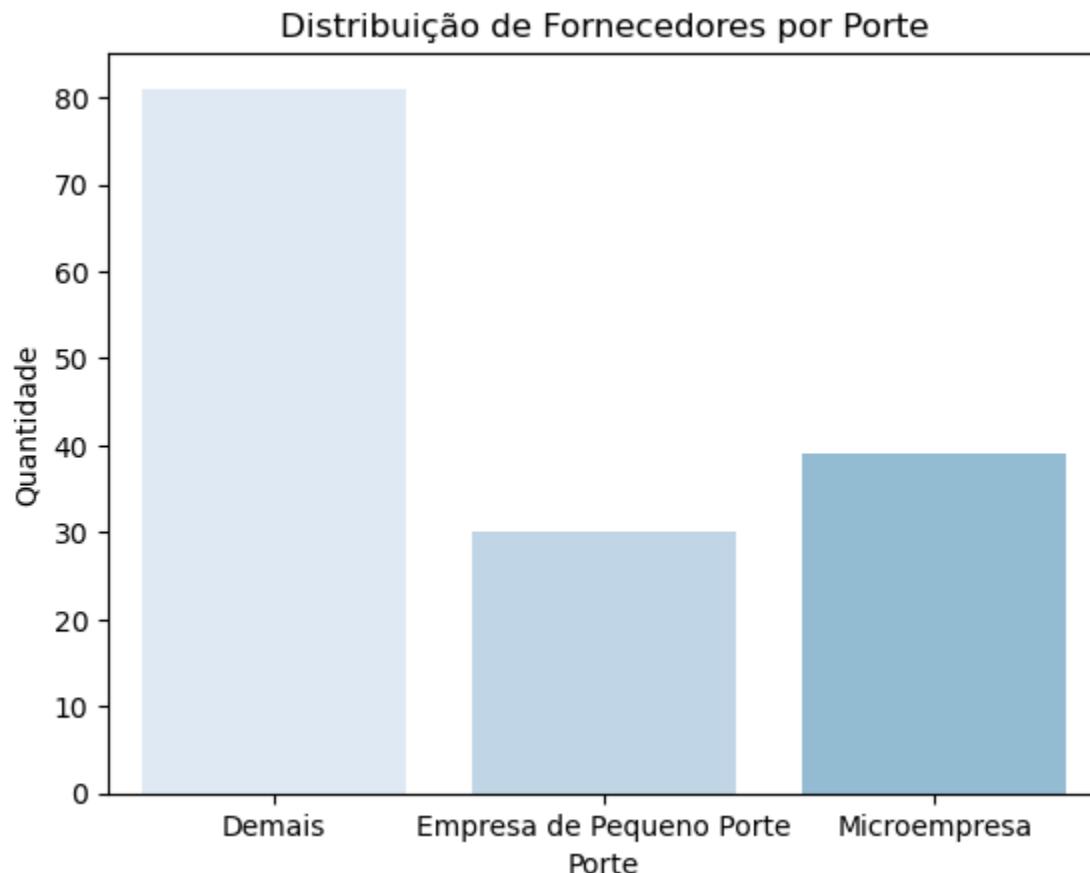


```
In [63]: import seaborn as sns
import matplotlib.pyplot as plt

# Configurando a paleta de cores para tons de azul
sns.set_palette("Blues")

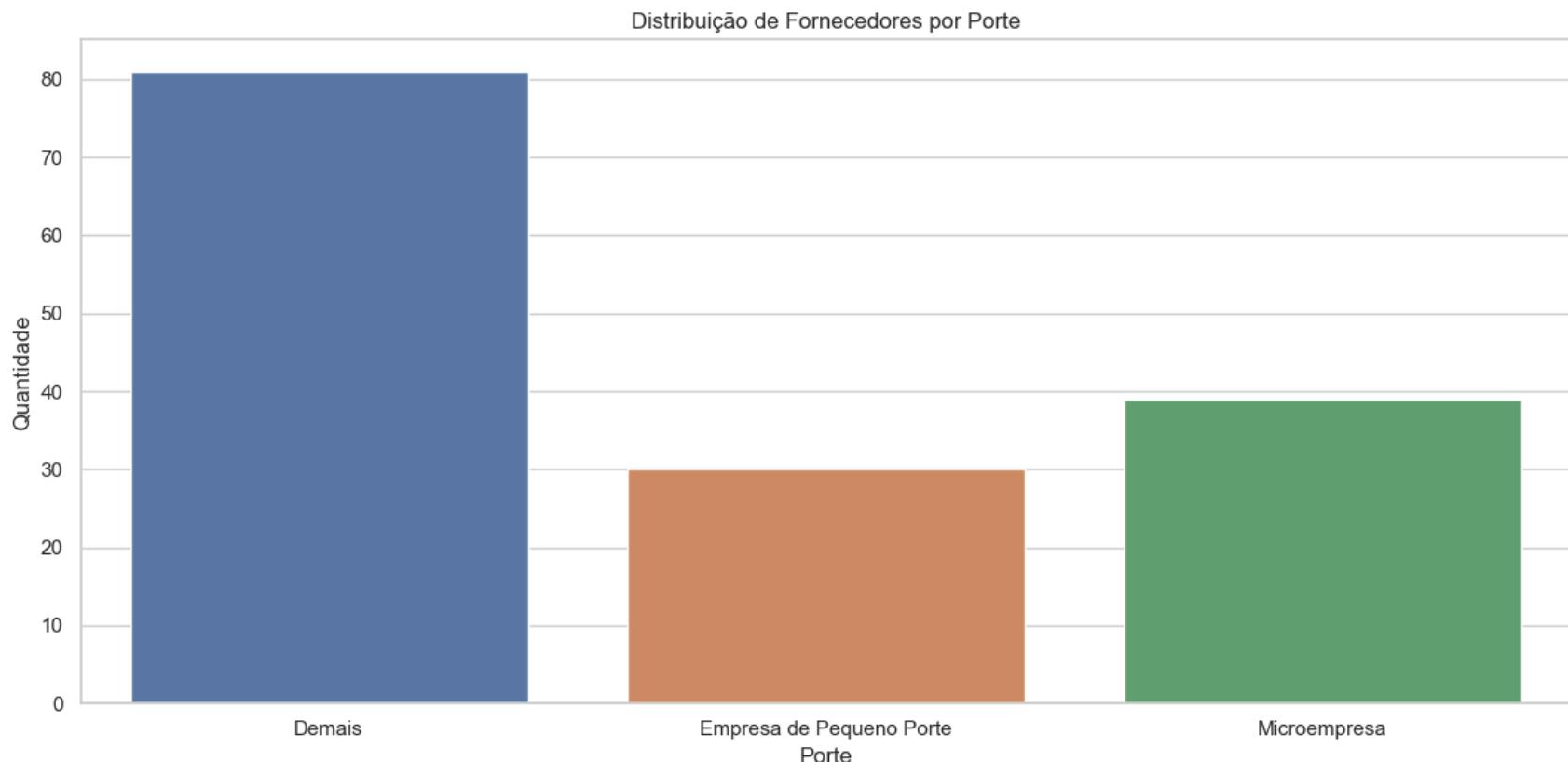
# Criando o gráfico de barras
sns.countplot(data=df_fornecedores, x='Porte')
plt.title('Distribuição de Fornecedores por Porte')
plt.xlabel('Porte')
plt.ylabel('Quantidade')
```

```
plt.show()
```



```
In [188]:
```

```
# Contagem por Porte
sns.countplot(data=df_amostra, x='Porte')
plt.title('Distribuição de Fornecedores por Porte')
plt.xlabel('Porte')
plt.ylabel('Quantidade')
plt.show()
```



3. Distribuição de Fornecedores com e sem CEIS:

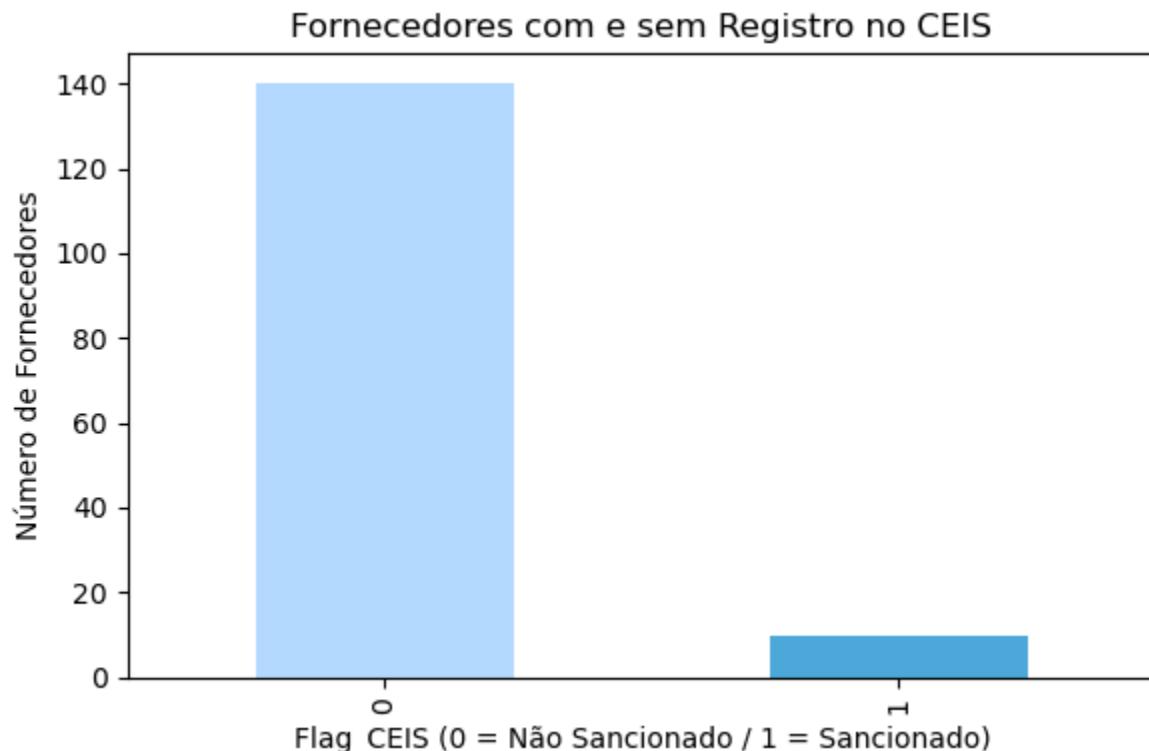
```
In [64]: import matplotlib.pyplot as plt

plt.figure(figsize=(6, 4))

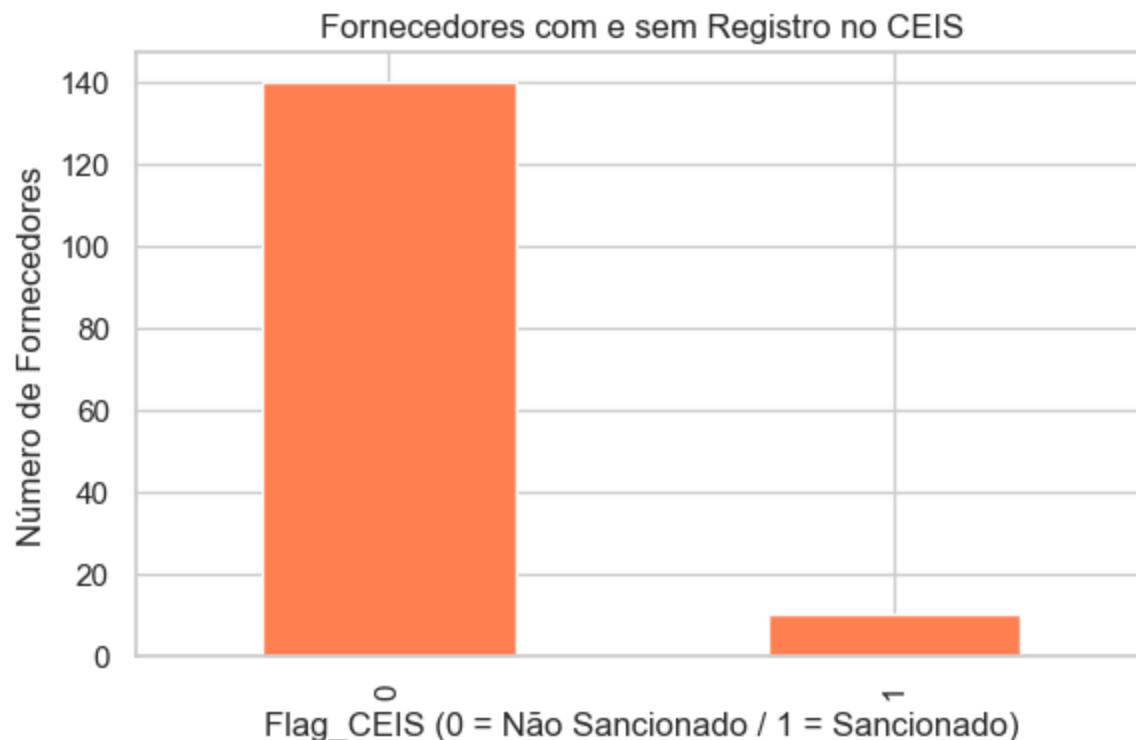
# Definindo uma paleta com tons de azul para as duas categorias
colors = ['#B3D9FF', '#4DA8DA'] # Azul claro e azul escuro

# Criando o gráfico de barras
df amostra['Flag_CEIS'].value_counts().plot(kind='bar', color=colors)
plt.title('Fornecedores com e sem Registro no CEIS')
plt.xlabel('Flag_CEIS (0 = Não Sancionado / 1 = Sancionado)')
plt.ylabel('Número de Fornecedores')
plt.tight_layout()
```

```
plt.show()
```



```
In [189...]:  
plt.figure(figsize=(6,4))  
df amostra['Flag_CEIS'].value_counts().plot(kind='bar', color='coral')  
plt.title('Fornecedores com e sem Registro no CEIS')  
plt.xlabel('Flag_CEIS (0 = Não Sancionado / 1 = Sancionado)')  
plt.ylabel('Número de Fornecedores')  
plt.tight_layout()  
plt.show()
```



In []:

Análises Descritivas

In [123...]

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Carregar os dados
df_total = pd.read_csv('contratos_1008_com_flags.csv')
df amostra = pd.read_csv('amostra_apenas_150_fornecedores.csv', sep=';')

# Padronizar CNPJ da amostra para 14 dígitos
df_amostra['CNPJ_clean'] = df_amostra['CNPJ'].astype(str).str.replace(r'\D', '', regex=True).str.zfill(14)
df_total['CNPJ_clean'] = df_total['CNPJ'].astype(str).str.zfill(14)

# Converter valores monetários com segurança
```

```
def converter_valor(valor):
    if isinstance(valor, str):
        valor = valor.replace('R$', '').replace('.', '').replace(',', '.')
    try:
        return float(valor)
    except:
        return 0.0

df_total['Valor Global'] = df_total['Valor Global'].apply(converter_valor)

# Converter datas
df_total['Vig. Início'] = pd.to_datetime(df_total['Vig. Início'], errors='coerce', dayfirst=True)
df_total['Vig. Fim'] = pd.to_datetime(df_total['Vig. Fim'], errors='coerce', dayfirst=True)

# Filtrar amostra na base total
df_amostra_total = df_total[df_total['CNPJ_clean'].isin(df_amostra['CNPJ_clean'])].copy()

# Função de análise descritiva com boxplot e histograma
def analise_descritiva(df, titulo):
    descricao = df['Valor Global'].describe()
    print(f"\n📊 Estatísticas Descritivas - {titulo}")
    print(descricao)

    # Boxplot
    plt.figure(figsize=(10, 4))
    sns.boxplot(x=df['Valor Global'])
    plt.title(f'Boxplot do Valor Global - {titulo}')
    plt.xlabel('Valor Global (R$)')
    plt.grid(True)
    plt.show()

    # Histograma
    plt.figure(figsize=(10, 4))
    sns.histplot(df['Valor Global'], bins=50, kde=True)
    plt.title(f'Distribuição do Valor Global - {titulo}')
    plt.xlabel('Valor Global (R$)')
    plt.ylabel('Frequência')
    plt.grid(True)
    plt.show()

# Execução para base completa e amostra
analise_descritiva(df_total, 'Base Completa (1008 contratos)')
```

```
analise_descritiva(df amostra_total, 'Amostra (143 fornecedores, ~248 contratos)')
```

Estatísticas Descritivas - Base Completa (1008 contratos)

count 1.008000e+03

mean 9.282198e+06

std 1.088956e+08

min 0.000000e+00

25% 6.400000e+04

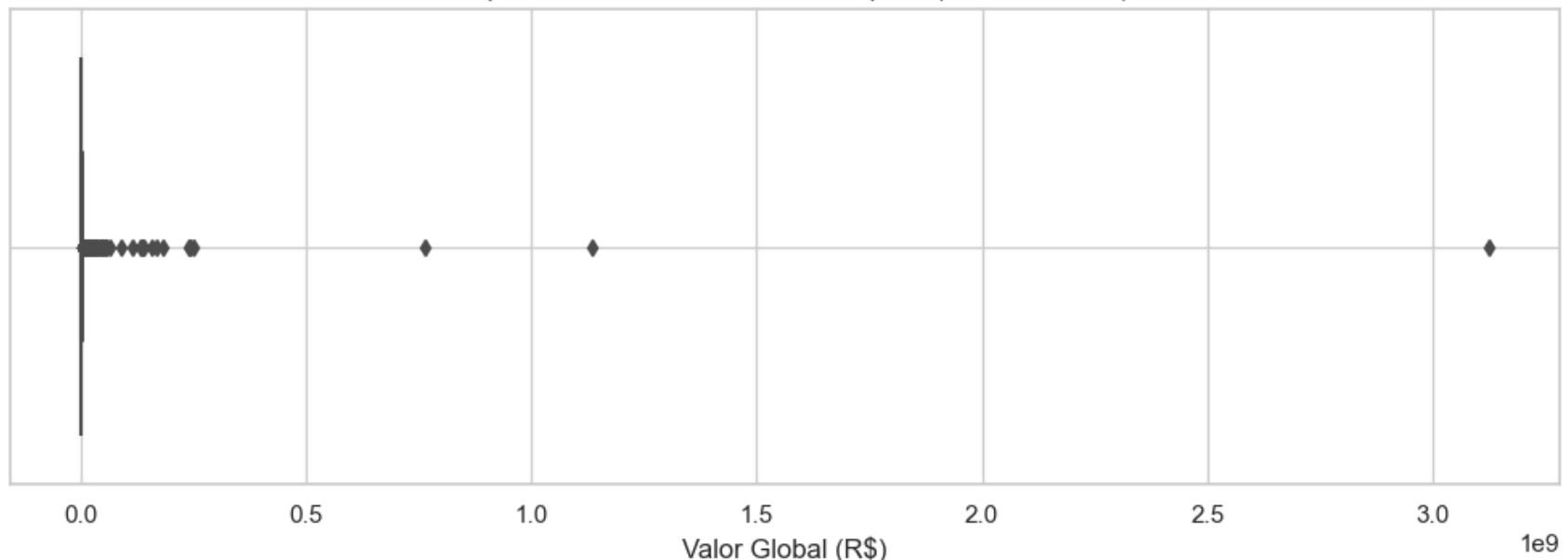
50% 3.314259e+05

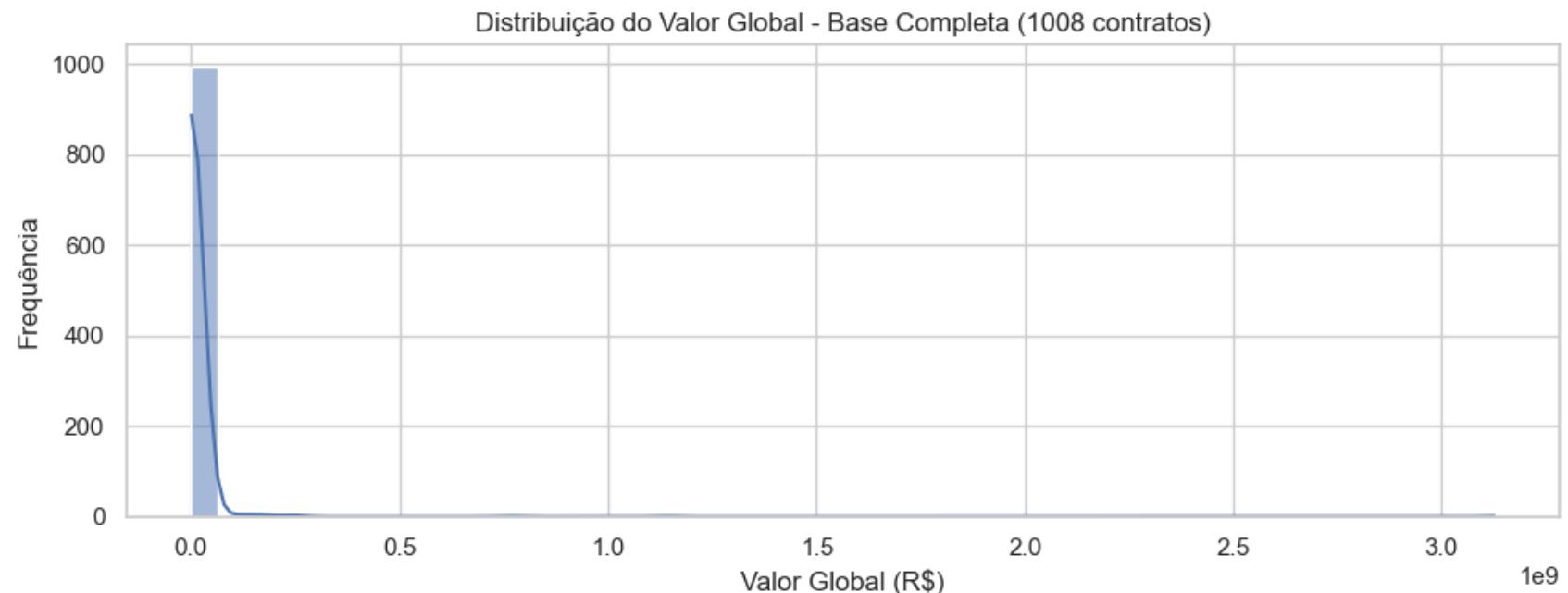
75% 1.933727e+06

max 3.125903e+09

Name: Valor Global, dtype: float64

Boxplot do Valor Global - Base Completa (1008 contratos)

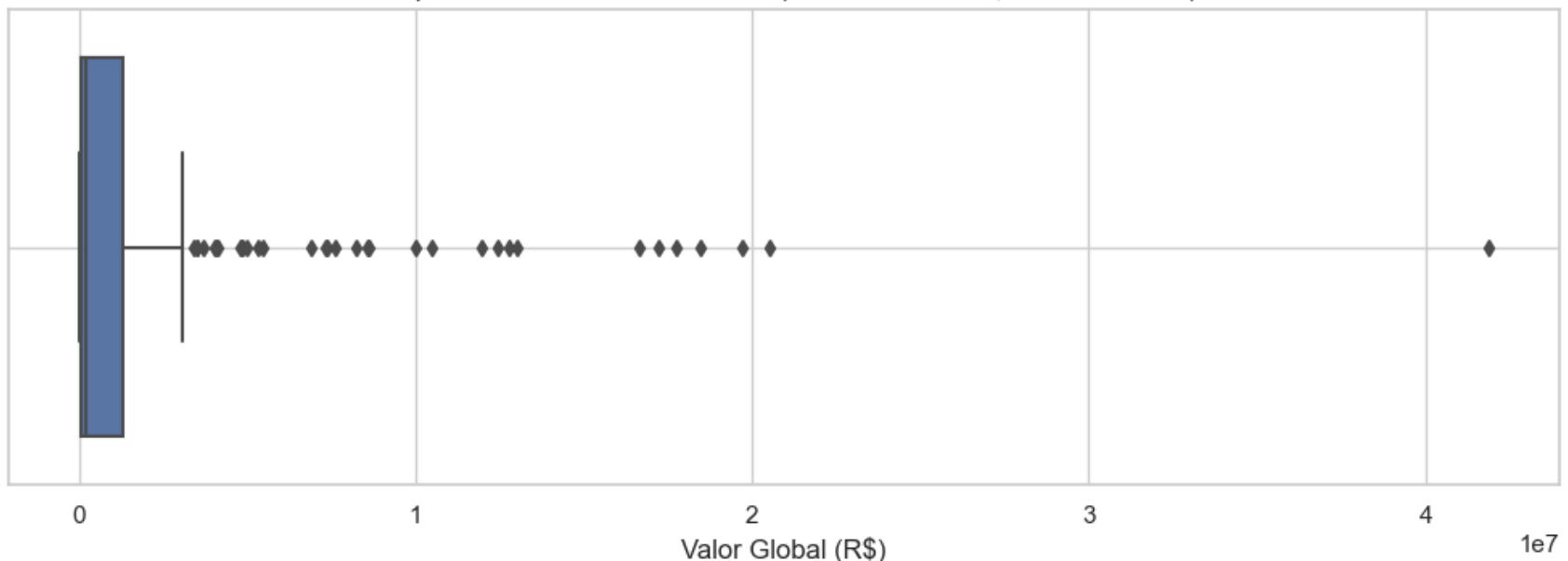




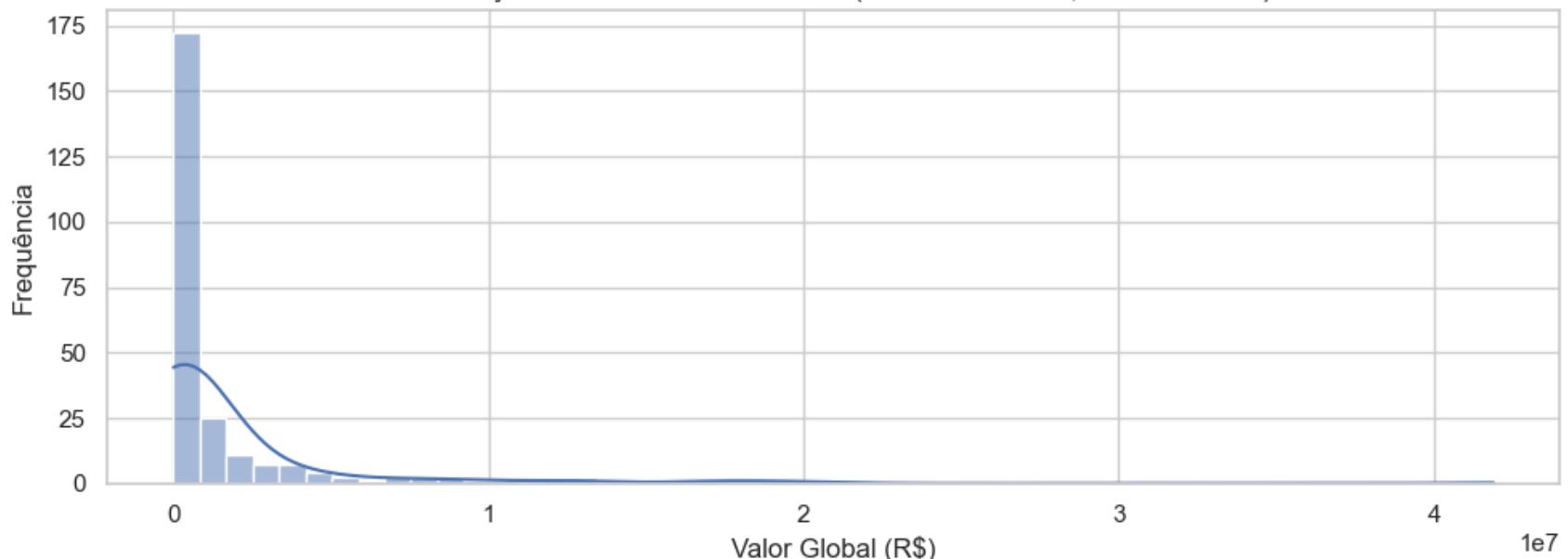
📊 Estatísticas Descritivas - Amostra (143 fornecedores, ~248 contratos)

```
count    2.480000e+02
mean     1.754830e+06
std      4.364545e+06
min      1.000000e-02
25%      4.512347e+04
50%      1.996107e+05
75%      1.296453e+06
max      4.188582e+07
Name: Valor Global, dtype: float64
```

Boxplot do Valor Global - Amostra (143 fornecedores, ~248 contratos)



Distribuição do Valor Global - Amostra (143 fornecedores, ~248 contratos)



✓ 1. Coerência das Estatísticas Descritivas

📌 Base Completa (1008 contratos) Média \approx R\$ 9,28 milhões

Mediana \approx R\$ 331 mil

Desvio padrão extremamente alto (R\$ 108 milhões)

Assimetria alta (média muito maior que a mediana)

Outliers identificados no boxplot

Interpretação: a distribuição é altamente assimétrica à direita, com muitos contratos pequenos e poucos contratos gigantescos (acima de R\$ 1 bilhão). O histograma confirma essa cauda longa.

📌 Amostra (143 fornecedores, ~248 contratos) Média \approx R\$ 1,75 milhão

Mediana \approx R\$ 199 mil

Desvio padrão moderado (R\$ 4,36 milhões)

Distribuição também assimétrica, mas menos que a base completa

Interpretação: o comportamento é similar ao da base completa, mas menos extremo. A amostra é uma boa representação para análise inferencial — não há viés aparente de seleção, e conseguimos capturar tanto pequenos como grandes fornecedores.

✓ 2. Coerência dos Gráficos Os boxplots e histogramas com KDE para ambas as bases mostram:

Presença de muitos valores pequenos concentrados.

Alguns valores fora do padrão, marcados como outliers.

Distribuição não normal — o que será importante nos testes de hipóteses e na escolha de testes não paramétricos, se necessário.

Conclusão visual: gráficos estão corretos, com escalas coerentes para análise econômica e estatística. A presença de outliers não inviabiliza a análise, mas sugere atenção ao tipo de teste que será usado.

Testes de Normalidade (Base Completa e Amostra)

Testes de Shapiro-Wilk e Kolmogorov-Smirnov, considerando os dois grupos:

```
In [60]: # Importar bibliotecas
import pandas as pd
from scipy.stats import shapiro, kstest, normaltest
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 1. Carregar as bases já tratadas
df_total = pd.read_csv('contratos_1008_com_flags.csv')
df amostra = pd.read_csv('amostra_apenas_150_fornecedores.csv', sep=';')

# 2. Padronizar CNPJ da amostra
df_amostra['CNPJ_clean'] = df_amostra['CNPJ'].astype(str).str.replace(r'\D', '', regex=True).str.zfill(14)
df_total['CNPJ_clean'] = df_total['CNPJ'].astype(str).str.zfill(14)

# 3. Filtrar os contratos da amostra
df_amostra_total = df_total[df_total['CNPJ_clean'].isin(df_amostra['CNPJ_clean'])].copy()

# 4. Garantir que 'Valor Global' esteja numérico
def converter_valor(valor):
    if isinstance(valor, str):
        valor = valor.replace('R$', '').replace('.', '').replace(',', '.')
    try:
        return float(valor)
    except:
        return 0.0

df_total['Valor Global'] = df_total['Valor Global'].apply(converter_valor)
df_amostra_total['Valor Global'] = df_amostra_total['Valor Global'].apply(converter_valor)

# 5. Função para aplicar os testes de normalidade
def testar_normalidade(serie, nome_base):
    print(f"\n📊 Teste de Normalidade - {nome_base}")

    # Shapiro-Wilk
    stat_sw, p_sw = shapiro(serie)
    print(f"Shapiro-Wilk: estatística={stat_sw:.4f}, p-valor={p_sw:.4f}")

    # Kolmogorov-Smirnov (com média e std empíricos)
```

```
stat_ks, p_ks = kstest(  
    (serie - serie.mean()) / serie.std(ddof=1),  
    'norm'  
)  
print(f"Kolmogorov-Smirnov: estatística={stat_ks:.4f}, p-valor={p_ks:.4f}")  
  
# D'Agostino-Pearson (teste extra para robustez)  
stat_dp, p_dp = normaltest(serie)  
print(f"D'Agostino-Pearson: estatística={stat_dp:.4f}, p-valor={p_dp:.4f}")  
  
# 📈 Plot para visualização  
plt.figure(figsize=(8,4))  
sns.histplot(serie, bins=30, kde=True)  
plt.title(f'Distribuição da Variável: {nome_base}')  
plt.xlabel('Valor Global (R$)')  
plt.show()  
  
# ✎ 6. Aplicar os testes nas duas bases  
testar_normalidade(df_total['Valor Global'], "Base Completa (1008 contratos)")  
testar_normalidade(df amostra_total['Valor Global'], "Amostra (143 fornecedores, ~248 contratos)")
```

📊 Teste de Normalidade - Base Completa (1008 contratos)

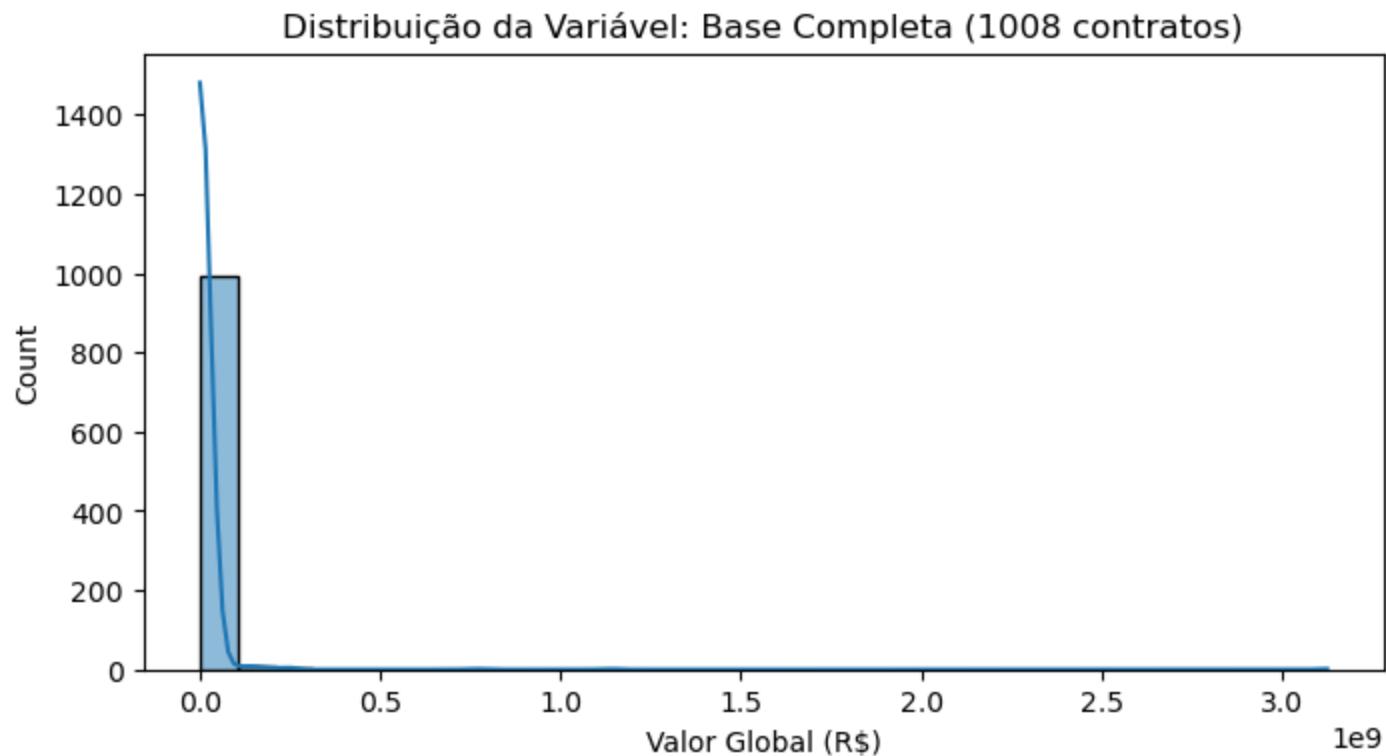
Shapiro-Wilk: estatística=0.0495, p-valor=0.0000

Kolmogorov-Smirnov: estatística=0.4660, p-valor=0.0000

D'Agostino-Pearson: estatística=2502.3056, p-valor=0.0000

C:\Users\F8094564\AppData\Local\anaconda3\Lib\site-packages\seaborn_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.

with pd.option_context('mode.use_inf_as_na', True):



📊 Teste de Normalidade - Amostra (143 fornecedores, ~248 contratos)

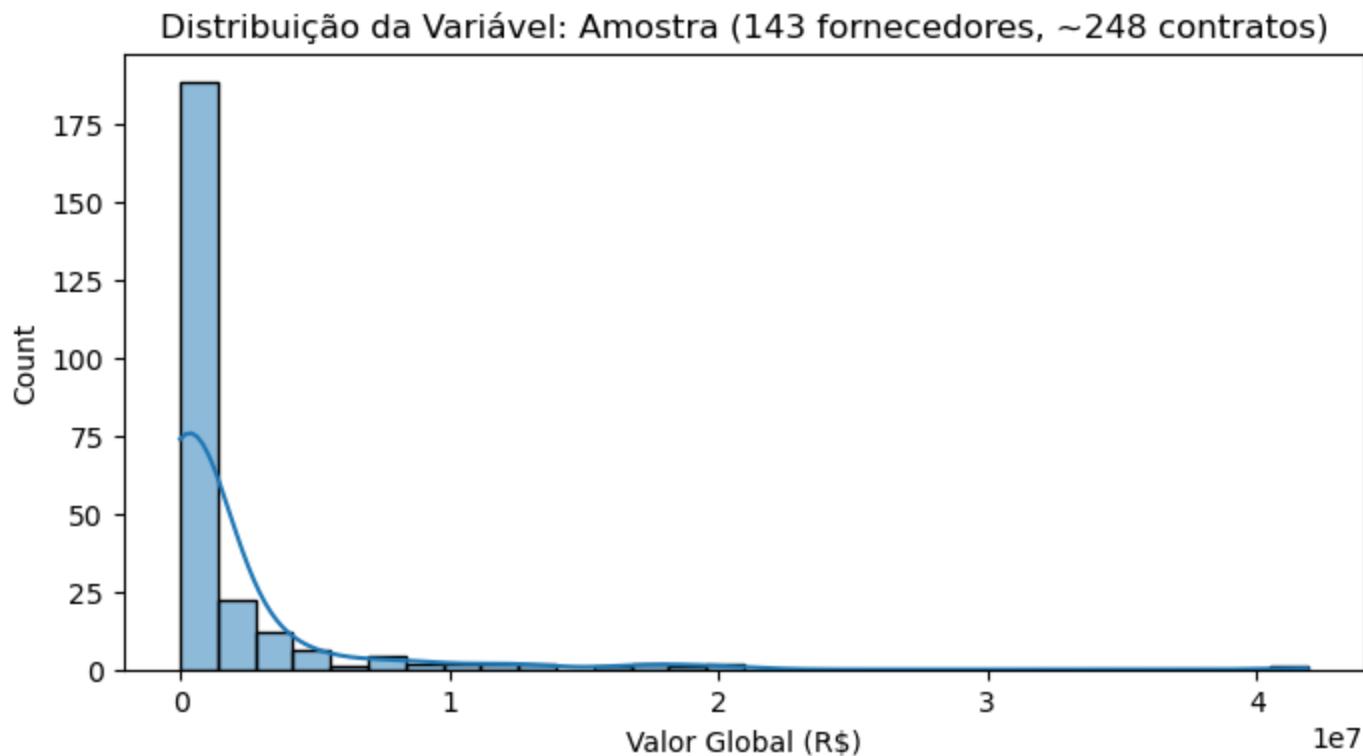
Shapiro-Wilk: estatística=0.4348, p-valor=0.0000

Kolmogorov-Smirnov: estatística=0.3438, p-valor=0.0000

D'Agostino-Pearson: estatística=284.1575, p-valor=0.0000

C:\Users\F8094564\AppData\Local\anaconda3\Lib\site-packages\seaborn_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.

```
with pd.option_context('mode.use_inf_as_na', True):
```



Teste de Normalidade para as variáveis "Valor Global", "Capital Social" e "Idade da Empresa"

```
In [61]: # 📚 Bibliotecas necessárias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import shapiro, kstest, normaltest

# 💡 1. Carregar o arquivo
df = pd.read_csv("amostra_apenas_150_fornecedores.csv", sep=';')

# 💡 2. Tratar as variáveis para garantir formato numérico

# Capital Social
df['Capital Social'] = df['Capital Social'].replace('R$', '', regex=True)
```

```
df['Capital Social'] = df['Capital Social'].replace('. ', ' ', regex=True)
df['Capital Social'] = df['Capital Social'].replace(', ', '.', regex=True)
df['Capital Social'] = pd.to_numeric(df['Capital Social'], errors='coerce').fillna(0)

# Idade da Empresa
df['Idade da Empresa'] = pd.to_numeric(df['Idade da Empresa'], errors='coerce').fillna(0)

# Valor Global (se desejar aplicar também)
if 'Valor Global' in df.columns:
    df['Valor Global'] = df['Valor Global'].replace('R$', ' ', regex=True)
    df['Valor Global'] = df['Valor Global'].replace('. ', ' ', regex=True)
    df['Valor Global'] = df['Valor Global'].replace(',', '.', regex=True)
    df['Valor Global'] = pd.to_numeric(df['Valor Global'], errors='coerce').fillna(0)

# 3. Função para aplicar os testes de normalidade
def testar_normalidade(serie, nome_variavel):
    print(f"\n📊 Testes de Normalidade - {nome_variavel}")
    serie = serie.dropna()

    # Shapiro-Wilk
    stat_sw, p_sw = shapiro(serie)
    print(f"Shapiro-Wilk: estatística={stat_sw:.4f}, p-valor={p_sw:.4f}")

    # Kolmogorov-Smirnov (normalizado)
    stat_ks, p_ks = kstest((serie - serie.mean()) / serie.std(ddof=1), 'norm')
    print(f"Kolmogorov-Smirnov: estatística={stat_ks:.4f}, p-valor={p_ks:.4f}")

    # D'Agostino-Pearson
    stat_dp, p_dp = normaltest(serie)
    print(f"D'Agostino-Pearson: estatística={stat_dp:.4f}, p-valor={p_dp:.4f}")

    # Plot
    plt.figure(figsize=(8, 4))
    sns.histplot(serie, bins=30, kde=True)
    plt.title(f'Distribuição da Variável: {nome_variavel}')
    plt.xlabel(nome_variavel)
    plt.ylabel('Frequência')
    plt.grid(True)
    plt.show()

# 4. Executar para as variáveis desejadas
testar_normalidade(df['Capital Social'], 'Capital Social')
```

```
testar_normalidade(df['Idade da Empresa'], 'Idade da Empresa')

if 'Valor Global' in df.columns:
    testar_normalidade(df['Valor Global'], 'Valor Global')
```

Testes de Normalidade - Capital Social

Shapiro-Wilk: estatística=0.0641, p-valor=0.0000

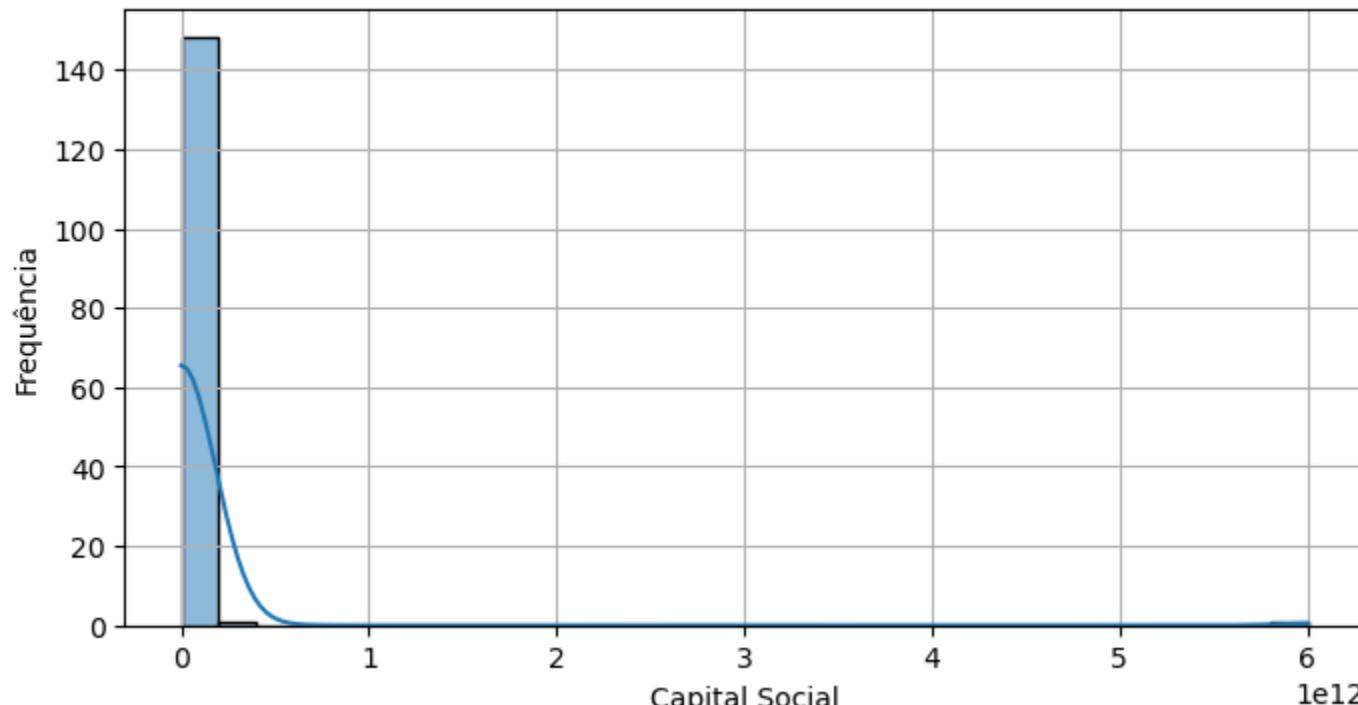
Kolmogorov-Smirnov: estatística=0.4954, p-valor=0.0000

D'Agostino-Pearson: estatística=325.9470, p-valor=0.0000

C:\Users\F8094564\AppData\Local\anaconda3\Lib\site-packages\seaborn_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.

```
with pd.option_context('mode.use_inf_as_na', True):
```

Distribuição da Variável: Capital Social



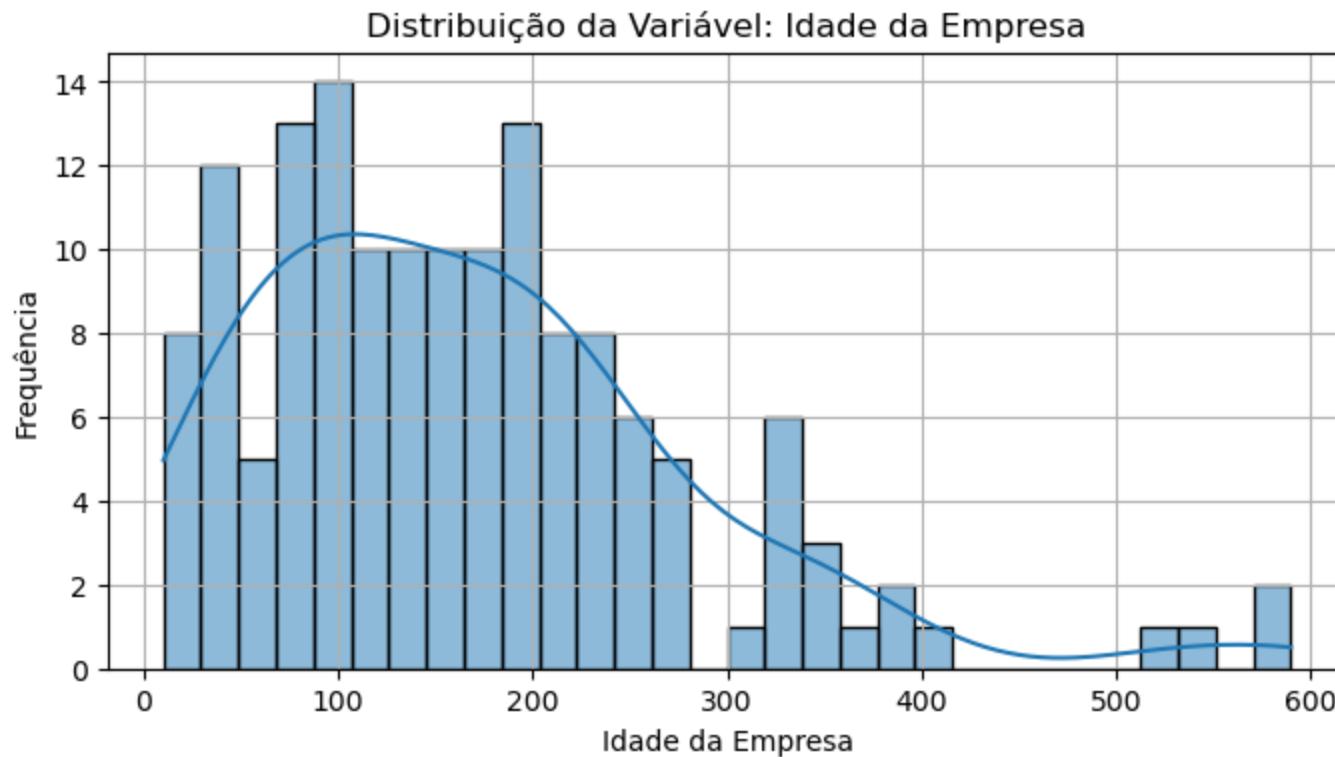
Testes de Normalidade - Idade da Empresa

Shapiro-Wilk: estatística=0.9135, p-valor=0.0000

Kolmogorov-Smirnov: estatística=0.0933, p-valor=0.1374

D'Agostino-Pearson: estatística=37.9431, p-valor=0.0000

C:\Users\F8094564\AppData\Local\anaconda3\Lib\site-packages\seaborn_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
with pd.option_context('mode.use_inf_as_na', True):



In []:

🧠 Próxima Etapa: Testes de Hipóteses (Inferência Estatística)

1. Diferença de Médias / Localizações:

Como os dados não são normais → Usaremos testes não paramétricos, como:

Mann-Whitney U Test → Para comparar o Valor Global entre dois grupos (exemplo: fornecedores da amostra vs. demais, ou CNEP vs. não CNEP).

2. Associação entre Variáveis Categóricas:

Por exemplo, verificar associação entre ministérios e presença de fornecedores da amostra, ou presença em CEIS/CNEP.

Teste Qui-Quadrado para variáveis categóricas.

3. Correlação (Spearman ou Kendall):

Como os dados não são normais → usar Correlação de Spearman entre valor global e número de contratos por fornecedor/ministério.

4. Análise de Outliers:

Podemos fazer um filtro ou Winsorização antes de alguns testes



Hipóteses de Pesquisa (H1 a H10):

H1 Ministérios com maior número de contratos possuem valores médios de contratos mais baixos.

H2 Fornecedores com maior idade (tempo de existência) possuem contratos de maior valor.

H3 Fornecedores sancionados (presentes nas listas CNEP ou CEIS) possuem contratos de menor valor.

H4 Empresas de maior porte (classificação oficial de porte) possuem contratos de maior valor.

H5 Fornecedores localizados na Região Sudeste possuem contratos de maior valor.

H6 Fornecedores localizados no Distrito Federal possuem contratos de maior valor.

H7 Fornecedores localizados na Região Sudeste possuem maior número de contratos.

H8 Fornecedores localizados no Distrito Federal possuem maior número de contratos.

H9 Empresas com maior capital social possuem maior número de contratos.

H10 Empresas com maior capital social possuem contratos de maior valor.



Observações:

As hipóteses H1 até H6 estão focadas na análise de valores de contratos (variável dependente = Valor Global do contrato).

As hipóteses H7 até H9 envolvem quantidade de contratos por fornecedor (contagem).

Algumas hipóteses (ex.: H2, H9, H10) envolvem variáveis contínuas e podem exigir análise por correlação, enquanto outras (ex.: H1, H4, H5, H6) envolvem grupos e exigirão testes de comparação de médias (não-paramétricos ou paramétricos conforme a normalidade).



Formulação e Teste das Hipóteses (Inferência Estatística)

📌 Ordem Recomendada entre Análise Descritiva, Testes H1-H10 e Outras Análises Inferenciais

Etapa	O que fazer	Por que fazer aqui?
✓ 1. Estatísticas Descritivas + EDA (Já feito)	Médias, medianas, boxplots, histogramas, análise de outliers	Para entender a distribuição inicial dos dados e possíveis problemas de escala, normalidade e outliers.
✓ 2. Teste de Normalidade (Já feito)	Shapiro-Wilk, Kolmogorov-Smirnov, D'Agostino	Para fundamentar a escolha de testes paramétricos ou não paramétricos nas etapas seguintes.
✓ 3. Formulação e Testes das Hipóteses Específicas (H1 a H10) (Em andamento)	Rodar os testes de Mann-Whitney, Spearman, Kruskal-Wallis, etc., para cada hipótese do seu quadro (H1 a H10)	Porque são as hipóteses principais do artigo.
➡ 4. Testes Comparando Amostra vs Restante da População (Mann-Whitney)	Exemplo: Comparar se os fornecedores da amostra (143) têm valores de contratos diferentes dos demais fornecedores da base (306 fornecedores restantes).	Isso é importante para validar se a amostra é enviesada ou representativa em relação ao todo antes de usar apenas a amostra nos próximos passos.
➡ 5. Testes de Associação Global (Qui-Quadrado entre variáveis categóricas)	Exemplo: Associação entre CNEP/CEIS e Ministério, ou entre Estado e Volume de contratos.	Complementa a etapa anterior, avaliando associação entre variáveis categóricas no total da base.
➡ 6. Correlações Adicionais (Spearman entre quantidade de contratos e valores)	Exemplo: Correlação entre número de contratos por fornecedor e o valor total contratado.	Avalia relação entre variáveis contínuas que ainda não foram testadas nas H1-H10.

Etapa	O que fazer	Por que fazer aqui?
-------	-------------	---------------------

**Opcional 7. Análise de Outliers
(Winsorização, Filtragem, etc)**

Se os outliers estiverem impactando demais os testes anteriores, podemos criar versões sem outliers para análise robusta.

Só vale a pena fazer se os resultados atuais estiverem distorcidos.

Testes de Hipóteses (H1 a H10)

🔍 H1: Ministérios com mais contratos têm valores médios mais baixos

In [125...]

```
import scipy.stats as stats

# Criar uma variável com o número de contratos por ministério
df_total['Qtd_Contratos_Min'] = df_total.groupby('Órgão')['Órgão'].transform('count')

# Criar um corte: Ministérios com + que a mediana de contratos vs. demais
mediana_contratos_min = df_total['Qtd_Contratos_Min'].median()
df_total['Grupo_H1'] = df_total['Qtd_Contratos_Min'].apply(lambda x: 'Alto Volume' if x > mediana_contratos_min else 'Baixo Volume')

# Teste de Mann-Whitney U
grupo_alto = df_total[df_total['Grupo_H1'] == 'Alto Volume']['Valor Global']
grupo_baixo = df_total[df_total['Grupo_H1'] == 'Baixo Volume']['Valor Global']

stat, p = stats.mannwhitneyu(grupo_alto, grupo_baixo, alternative='two-sided')
print(f"H1 - Mann-Whitney U Test: estatística={stat:.4f}, p-valor={p:.4f}")
```

H1 - Mann-Whitney U Test: estatística=101363.5000, p-valor=0.0000

🔍 H2: Fornecedores mais antigos têm contratos de maior valor

In [127...]

```
import pandas as pd
from scipy import stats

# ✅ 1. Carregar os dados
df_total = pd.read_csv('contratos_1008_com_flags.csv')
df amostra = pd.read_csv('amostra_apenas_150_fornecedores.csv', sep=';')

# ✅ 2. Padronizar CNPJs
df_total['CNPJ_clean'] = df_total['CNPJ'].astype(str).str.zfill(14)
```

```
df_amostra['CNPJ_clean'] = df_amostra['CNPJ'].astype(str).str.replace(r'\D', '', regex=True).str.zfill(14)

# ✅ 3. Filtrar apenas contratos da amostra
df_amostra_total = df_total[df_total['CNPJ_clean'].isin(df_amostra['CNPJ_clean'])].copy()

# ✅ 4. Trazer a idade da empresa da amostra para o dataframe de contratos
df_amostra_total = df_amostra_total.merge(
    df_amostra[['CNPJ_clean', 'Idade da Empresa']],
    on='CNPJ_clean',
    how='left'
)

# ✅ 5. Garantir que a idade da empresa seja numérica
df_amostra_total['Idade da Empresa'] = pd.to_numeric(df_amostra_total['Idade da Empresa'], errors='coerce')

# ✅ 6. Aplicar o teste de Spearman (correlação entre idade e valor global dos contratos)
stat, p = stats.spearmanr(df_amostra_total['Idade da Empresa'], df_amostra_total['Valor Global'])

# ✅ 7. Resultado
print(f"H2 - Correlação de Spearman entre Idade da Empresa e Valor Global: estatística={stat:.4f}, p-valor={p:.4f}")
```

H2 - Correlação de Spearman entre Idade da Empresa e Valor Global: estatística=-0.0274, p-valor=0.6672

✅ Considerações importantes antes de gerar os próximos scripts:

- ✓ Sobre a base de contratos (df_total): Contém os 1008 contratos

Contém as flags CEIS e CNEP (0 ou 1)

Contém os valores já convertidos para float

Contém o CNPJ limpo como CNPJ_clean

- ✓ Sobre a base da amostra (df_amostra): Contém os 150 fornecedores, com colunas como:

Porte

Estado

Capital Social

Idade da Empresa

CNPJ (em formato com traço e barra, então sempre limpar!)

✓ Para todos os testes daqui pra frente: Sempre que a hipótese envolver alguma variável da amostra, o script deve começar com um merge trazendo os dados da amostra para dentro do dataframe de contratos da amostra.

✓ Status das Hipóteses (Atualizado): | Hipótese | Variável Fonte | Tipo de Teste | ----- | ----- | ----- |
----- | | H1 | Base Total | Mann-Whitney ✓ | | H2 | Amostra (Idade) | Spearman ✓ | | H3 | (Excluída) | - | | H4 | Flag
CEIS/CNEP | Mann-Whitney | | H5 | Porte (amostra) | ANOVA ou Kruskal-Wallis | | H6 | Região (Estado da amostra - Sudeste) | Mann-
Whitney | | H7 | Estado (DF) | Mann-Whitney | | H8 | Região (Sudeste) | Teste de Proporções | | H9 | Estado (DF) | Teste de Proporções
| | H10 | Capital Social | Spearman ou Regressão |

🔍 H3: Fornecedores sancionados (CNEP/CEIS) têm contratos de menor valor

Hipótese: H3: Fornecedores sancionados (com flag CEIS ou CNEP igual a 1) possuem contratos com valor global menor.

Teste sugerido: Mann-Whitney U Test (porque já vimos que os dados não seguem distribuição normal)

```
In [134...]
# 🎨 Importar bibliotecas
import pandas as pd

# 🚀 Carregar os dados
df_total = pd.read_csv('contratos_1008_com_flags.csv', sep=',', encoding='utf-8')
df_amostra = pd.read_csv('amostra_apenas_150_fornecedores.csv', sep=';', encoding='utf-8')

# 🖌 Padronizar CNPJ nas duas bases
df_total['CNPJ_clean'] = df_total['CNPJ'].astype(str).str.replace(r'\D', '', regex=True).str.zfill(14)
df_amostra['CNPJ_clean'] = df_amostra['CNPJ'].astype(str).str.replace(r'\D', '', regex=True).str.zfill(14)

# 🚀 Realizar o merge: pegar só os contratos que estão na amostra
df_amostra_total = df_total[df_total['CNPJ_clean'].isin(df_amostra['CNPJ_clean'])].copy()

# ✅ Verificar estrutura do df_amostra_total
print("\n✓ Estrutura de df_amostra_total:\n")
print(df_amostra_total.info())

# ✅ Visualizar as primeiras linhas
```

```
print("\n✓ Primeiros registros:\n")
print(df amostra_total.head())
```

Estrutura de df amostra_total:

```
<class 'pandas.core.frame.DataFrame'>
Index: 248 entries, 0 to 998
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Órgão            248 non-null    object  
 1   Unidade Gestora  248 non-null    object  
 2   Número Contrato 248 non-null    object  
 3   Fornecedor       248 non-null    object  
 4   Vig. Início      248 non-null    object  
 5   Vig. Fim          248 non-null    object  
 6   Valor Global     248 non-null    object  
 7   Núm. Parcelas    248 non-null    int64  
 8   Valor Parcela    248 non-null    object  
 9   ministerio       248 non-null    object  
 10  CNPJ             248 non-null    int64  
 11  Nome Fornecedor 248 non-null    object  
 12  Flag_CNEP        248 non-null    int64  
 13  Flag_CEIS        248 non-null    int64  
 14  CNPJ_clean       248 non-null    object  
dtypes: int64(4), object(11)
memory usage: 31.0+ KB
None
```

Primeiros registros:

	Órgão	Unidade Gestora	Número Contrato	Fornecedor		
Vig. Início	Vig. Fim	Valor Global	Núm. Parcelas	Valor Parcela	ministerio	
CNPJ	Nome Fornecedor			Flag_CNEP	Flag_CEIS	CNPJ_clean
0 22000 - MINISTERIO DA AGRICULTURA E PECUARIA	INFORMATICA PROD...	130025 - SFA/PE/MAPA	130025 - SFA/PE/MAPA	00003/2019	64.799.539/0001-35	- TECNOSET
INFORMATICA PROD...	2019-05-07	2023-05-06	R\$ 62.015,76	12	R\$ 5.167,98	Consulta Contratos Contratos
.gov.br Min Agricu...	64799539000135	TECNOSET	INFORMATICA PRODUTOS E SERVICOS LTDA		0	0
539000135						64799
1 22000 - MINISTERIO DA AGRICULTURA E PECUARIA	SERVICOS DE LOC...	130025 - SFA/PE/MAPA	130025 - SFA/PE/MAPA	00006/2023	07.759.174/0001-81	- SOLUCOES
SERVICOS DE LOC...	2023-06-01	2025-05-31	R\$ 102.850,00	12	R\$ 8.570,83	Consulta Contratos Contratos.
gov.br Min Agricu...	7759174000181	SOLUCOES	SERVICOS DE LOCACAO DE MAQUINAS E EQ...		0	1
74000181						077591
2 22000 - MINISTERIO DA AGRICULTURA E PECUARIA	COMERCIO LOCACAO...	130067 - SFA/SP/MAPA	130067 - SFA/SP/MAPA	00006/2023	07.432.517/0001-07	- SIMPRESS
COMERCIO LOCACAO...	2023-04-01	2027-04-01	R\$ 720.238,67	48	R\$ 15.004,97	Consulta Contratos Contratos
.gov.br Min Agricu...	7432517000107	SIMPRESS	COMERCIO LOCACAO E SERVICOS LTDA		0	0
						07432

517000107
3 22000 - MINISTERIO DA AGRICULTURA E PECUARIA 130069 - SFA/MA/MAPA 00004/2022 08.951.049/0001-31 - CORESMA -
COMERCIO DE EQU... 2022-09-23 2025-09-22 R\$ 58.860,00 12 R\$ 4.905,00 Consulta Contratos Contratos.
gov.br Min Agricu... 8951049000131 CORESMA - COMERCIO DE EQUIPAMENTOS E SUPRIMENT... 0 0 089510
49000131
4 22000 - MINISTERIO DA AGRICULTURA E PECUARIA 130094 - SFA/PA/MAPA 00002/2023 03.117.534/0001-90 - BRADOK SO
LUCOES CORPORATI... 2023-11-21 2027-11-20 R\$ 129.484,80 12 R\$ 10.790,40 Consulta Contratos Contratos
.gov.br Min Agricu... 3117534000190 BRADOK SOLUCOES CORPORATIVAS LTDA 0 0 03117
534000190

In [135...]

```
from scipy import stats
import numpy as np
import pandas as pd

# ✅ 1. Converter a coluna 'Valor Global' de string para float
def converter_valor(valor):
    if isinstance(valor, str):
        valor = valor.replace('R$', '').replace('.', '').replace(',', '.')
    try:
        return float(valor)
    except:
        return np.nan

df amostra_total['Valor Global'] = df amostra_total['Valor Global'].apply(converter_valor)

# ✅ 2. Diagnóstico rápido pós-conversão
print("\n✅ Checagem rápida pós-conversão:")
print(df amostra_total['Valor Global'].describe())

# ✅ 3. Preparar os grupos: sancionados vs não sancionados (Flag_CEIS)
grupo_sancionado = df amostra_total.loc[
    (df amostra_total['Flag_CEIS'] == 1) & (df amostra_total['Valor Global'].notnull()),
    'Valor Global'
]

grupo_nao_sancionado = df amostra_total.loc[
    (df amostra_total['Flag_CEIS'] == 0) & (df amostra_total['Valor Global'].notnull()),
    'Valor Global'
]

print(f"\n📌 Número de contratos com fornecedores CEIS: {len(grupo_sancionado)}")
print(f"📌 Número de contratos com fornecedores NÃO CEIS: {len(grupo_nao_sancionado)}")
```

```
#  4. Teste Mann-Whitney U
if len(grupo_sancionado) > 0 and len(grupo_nao_sancionado) > 0:
    stat, p = stats.mannwhitneyu(grupo_sancionado, grupo_nao_sancionado, alternative='two-sided')
    print(f"\nH3 - Mann-Whitney U Test (Sancionados vs Não Sancionados): estatística={stat:.4f}, p-valor={p:.4f}")
else:
    print("\nX H3 - Um dos grupos está vazio. Teste não pode ser realizado.")
```

Checagem rápida pós-conversão:

```
count      2.480000e+02
mean       1.754830e+06
std        4.364545e+06
min        1.000000e-02
25%        4.512347e+04
50%        1.996107e+05
75%        1.296453e+06
max        4.188582e+07
Name: Valor Global, dtype: float64
```

- 👉 Número de contratos com fornecedores CEIS: 10
- 👉 Número de contratos com fornecedores NÃO CEIS: 238

H3 - Mann-Whitney U Test (Sancionados vs Não Sancionados): estatística=518.0000, p-valor=0.0025

Interpretação rápida de H3 – Fornecedores sancionados (CEIS) vs Não sancionados:

Número de contratos com fornecedores CEIS: 10

Número de contratos com fornecedores NÃO CEIS: 238

p-valor = 0.0025 (significativo ao nível de 5%)

👉 Interpretação inicial: Há diferença estatisticamente significativa entre os valores dos contratos dos fornecedores sancionados (CEIS) e os não sancionados. Isso nos dá indicativos para rejeitar a hipótese nula e aceitar que os valores médios/medianos são diferentes.

 H4: Empresas de maior porte têm contratos de maior valor

👉 Teste sugerido: Como o porte da empresa é categórico e o "Valor Global" é contínuo, o ideal é aplicar ANOVA (se houver)

normalidade e homocedasticidade) ou Kruskal-Wallis (se não houver normalidade, o mais provável pelo nosso diagnóstico anterior).

In [150...]

```
import pandas as pd
from scipy import stats

# ✓ 1. Recarregar os dataframes, se ainda não estiverem em memória
df amostra_total = pd.read_csv('contratos_1008_com_flags.csv')
df amostra_150 = pd.read_csv('amostra_apenas_150_fornecedores.csv', sep=';')

# ✓ 2. Padronizar os CNPJs nos dois DataFrames
df amostra_total['CNPJ_clean'] = df amostra_total['CNPJ'].astype(str).str.zfill(14)
df amostra_150['CNPJ_clean'] = df amostra_150['CNPJ'].astype(str).str.replace(r'\D', '', regex=True).str.zfill(14)

# ✓ 3. Filtrar apenas os contratos dos fornecedores da amostra
df amostra_total = df amostra_total[df amostra_total['CNPJ_clean'].isin(df amostra_150['CNPJ_clean'])].copy()

# ✓ 4. Fazer o merge para trazer as variáveis da amostra (como Porte, Capital Social, etc)
df amostra_total = df amostra_total.merge(
    df amostra_150[['CNPJ_clean', 'Porte', 'Capital Social', 'Idade da Empresa', 'Estado']],
    on='CNPJ_clean',
    how='left'
)

# ✓ 5. Tratar valores monetários
def converter_valor(valor):
    if isinstance(valor, str):
        valor = valor.replace('R$', '').replace('.', '').replace(',', '.')
    try:
        return float(valor)
    except:
        return 0.0

df amostra_total['Valor Global'] = df amostra_total['Valor Global'].apply(converter_valor)

# ✓ 6. Teste de Kruskal-Wallis para a H4 (Valor Global vs Porte)
print("\n📌 Categorias de Porte encontradas na amostra (após o merge):")
print(df amostra_total['Porte'].unique())

# Remove nulos ou vazios
df_h4 = df amostra_total[df amostra_total['Porte'].notnull() & (df amostra_total['Porte'] != '')]
```

```
# Monta os grupos
grupos = []
for porte in df_h4['Porte'].unique():
    grupo = df_h4.loc[df_h4['Porte'] == porte, 'Valor Global']
    grupos.append(grupo)

# Teste só se tiver pelo menos 2 grupos distintos
if len(grupos) >= 2:
    stat, p = stats.kruskal(*grupos)
    print(f"\n📊 H4 - Kruskal-Wallis (Valor Global por Porte da Empresa): estatística={stat:.4f}, p-valor={p:.4f}")
else:
    print("\n❌ H4 - Não há quantidade suficiente de grupos distintos para realizar o teste (pelo menos 2 grupos diferentes)")
```

📌 Categorias de Porte encontradas na amostra (após o merge):

['Demais' 'Microempresa' 'Empresa de Pequeno Porte']

📊 H4 - Kruskal-Wallis (Valor Global por Porte da Empresa): estatística=59.9340, p-valor=0.0000

✓ Interpretação rápida da H4 (Porte x Valor Global):

Resultado do teste Kruskal-Wallis: Estatística = 59,9340

p-valor = 0.0000

✓ O que significa esse resultado? O p-valor < 0,05 indica que há diferença estatisticamente significativa entre ao menos dois grupos de porte de empresas em relação ao valor global dos contratos.

Ou seja:

"O valor médio (ou a distribuição) dos contratos é significativamente diferente entre os grupos de porte (Demais, Microempresa, Empresa de Pequeno Porte)"

✓ Possível interpretação (preliminar): Sem olhar as médias ainda, é provável que empresas de maior porte (ou categoria "Demais") estejam concentrando contratos de maior valor, enquanto as micro e pequenas empresas (ME, EPP) têm contratos de menor valor.

✓ Próximos passos recomendados: Rodar uma análise descritiva por grupo de porte (média, mediana, etc.).

Plotar um boxplot por Porte (para visualizar as diferenças de distribuição).

(Opcional) Fazer testes post-hoc (exemplo: Dunn Test) se quiser saber exatamente quais grupos diferem entre si.

💡 H5: Fornecedores do Sudeste têm contratos de maior valor

- ✓ Estratégia Estatística: Comparar o Valor Global dos contratos entre dois grupos:

Grupo 1: Fornecedores da Região Sudeste

Grupo 2: Fornecedores de outras regiões

- ✓ Novamente, como os dados não são normais → vamos aplicar o Mann-Whitney U Test (não-paramétrico).

In [141...]

```
from scipy import stats
import pandas as pd

# ✓ Considerar apenas a amostra (df_amostra_total)
df = df_amostra_total.copy()

# ✓ Garantir que a coluna 'Estado' existe
print("\n📌 Estados únicos encontrados na amostra:")
print(df['Estado'].unique())

# ✓ Definir os estados da Região Sudeste
estados_sudeste = ['SP', 'RJ', 'MG', 'ES']

# ✓ Criar variável binária: Sudeste = 1, Outros = 0
df['Sudeste'] = df['Estado'].apply(lambda x: 1 if x in estados_sudeste else 0)

# ✓ Remover registros com Valor Global zerado ou negativo (se houver)
df = df[df['Valor Global'] > 0]

# ✓ Separar os dois grupos
grupo_sudeste = df.loc[df['Sudeste'] == 1, 'Valor Global']
grupo_outros = df.loc[df['Sudeste'] == 0, 'Valor Global']

print(f"\n📌 Número de contratos de fornecedores do Sudeste: {grupo_sudeste.shape[0]}")
print(f"\n📌 Número de contratos de fornecedores de outras regiões: {grupo_outros.shape[0]}")

# ✓ Teste de Mann-Whitney U
stat, p = stats.mannwhitneyu(grupo_sudeste, grupo_outros, alternative='two-sided')
```

```
print(f"\n📊 H5 - Mann-Whitney U Test (Sudeste vs Outras Regiões):")
print(f"Estatística U = {stat:.4f}")
print(f"p-valor = {p:.4f}")
```

📌 Estados únicos encontrados na amostra:
['SP' 'PE' 'MA' 'RJ' 'AP' 'DF' 'MG' 'PB' 'PR' 'ES' 'PA' 'GO' 'RS' 'SE'
'BA' 'SC' 'MT' 'RN' 'PI']

📌 Número de contratos de fornecedores do Sudeste: 122
📌 Número de contratos de fornecedores de outras regiões: 126

📊 H5 - Mann-Whitney U Test (Sudeste vs Outras Regiões):
Estatística U = 7564.5000
p-valor = 0.8304

✅ Interpretação rápida dos resultados de H5:

p-valor = 0,8304, ou seja, muito acima de 0,05.

Isso indica que não há diferença estatisticamente significativa entre os valores dos contratos de fornecedores do Sudeste e os das demais regiões, com base na amostra.

Logo, não podemos rejeitar a hipótese nula (H_0) de que os dois grupos têm a mesma distribuição de valores de contratos.

✅ Conclusão preliminar de H5: Hipótese Resultado preliminar H5: Fornecedores do Sudeste têm contratos de maior valor Não há evidências suficientes para confirmar. Diferença não significativa ($p>0,05$)

🔍 H6: Fornecedores do DF têm contratos de maior valor

In [142...]

```
from scipy import stats
import pandas as pd

# ✅ Carregar a amostra (df_amostra_total já deve estar tratado com 'Estado' e 'Valor Global' em float)
df = df_amostra_total.copy()

# ✅ Garantir que a coluna 'Estado' existe
print("\n📌 Estados únicos encontrados na amostra:")
print(df['Estado'].unique())
```

```
# ✅ Criar variável binária: DF = 1, Outros = 0
df['DF'] = df['Estado'].apply(lambda x: 1 if x == 'DF' else 0)

# ✅ Filtrar contratos com valor positivo
df = df[df['Valor Global'] > 0]

# ✅ Separar os grupos
grupo_df = df.loc[df['DF'] == 1, 'Valor Global']
grupo_outros = df.loc[df['DF'] == 0, 'Valor Global']

print(f"\n📌 Número de contratos com fornecedores do DF: {grupo_df.shape[0]}")
print(f"📌 Número de contratos com fornecedores de outras regiões: {grupo_outros.shape[0]}")

# ✅ Teste de Mann-Whitney U
if grupo_df.shape[0] > 0 and grupo_outros.shape[0] > 0:
    stat, p = stats.mannwhitneyu(grupo_df, grupo_outros, alternative='two-sided')

    print(f"\n📊 H6 - Mann-Whitney U Test (DF vs Outros Estados):")
    print(f"Estatística U = {stat:.4f}")
    print(f"p-valor = {p:.4f}")
else:
    print("\n❌ H6 - Não há dados suficientes para o teste. Um dos grupos tem zero observações.")
```

📌 Estados únicos encontrados na amostra:
['SP' 'PE' 'MA' 'RJ' 'AP' 'DF' 'MG' 'PB' 'PR' 'ES' 'PA' 'GO' 'RS' 'SE'
'BA' 'SC' 'MT' 'RN' 'PI']

📌 Número de contratos com fornecedores do DF: 70
📌 Número de contratos com fornecedores de outras regiões: 178

📊 H6 - Mann-Whitney U Test (DF vs Outros Estados):
Estatística U = 8863.5000
p-valor = 0.0000

📊 Interpretação rápida do resultado de H6:

Como o p-valor = 0.0000, temos evidências estatísticas fortes para rejeitar a hipótese nula, sugerindo que há diferença estatisticamente significativa nos valores medianos dos contratos entre fornecedores do DF e os de outras regiões.

Pela descrição da hipótese (H6), o próximo passo seria olhar a mediana dos dois grupos para verificar se de fato os fornecedores do

DF têm valores maiores.

💡 H7: Fornecedores do Sudeste têm mais contratos

(Teste de Qui-Quadrado)

In [143...]

```
from scipy.stats import chi2_contingency
import pandas as pd

# ✅ 1. Considerar a amostra
df = df_amostra_total.copy()

# ✅ 2. Garantir que a coluna 'Estado' existe
print("\n📌 Estados únicos na amostra:")
print(df['Estado'].unique())

# ✅ 3. Criar variável binária: Sudeste (1) vs Outras regiões (0)
estados_sudeste = ['SP', 'RJ', 'MG', 'ES']
df['Sudeste'] = df['Estado'].apply(lambda x: 1 if x in estados_sudeste else 0)

# ✅ 4. Criar tabela de contingência (Frequência de contratos por grupo)
contingencia = df['Sudeste'].value_counts().sort_index()
print("\n📌 Frequência de contratos (Sudeste vs Outros):")
print(contingencia)

# ✅ 5. Montar a tabela no formato esperado pelo teste
# Exemplo: [[Qtd contratos Sudeste], [Qtd contratos Outros]]
tabela = [
    [contingencia.get(1, 0)], # Sudeste
    [contingencia.get(0, 0)] # Outros
]

# ✅ 6. Teste Qui-Quadrado
chi2, p, dof, expected = chi2_contingency(tabela)

print("\n📊 H7 - Teste Qui-Quadrado (Sudeste vs Outras Regiões - Frequência de Contratos):")
print(f"Chi2 = {chi2:.4f}, p-valor = {p:.4f}, graus de liberdade = {dof}")
print("\n📌 Frequências Esperadas (pelo modelo nulo):")
print(expected)
```

📌 Estados únicos na amostra:

```
['SP' 'PE' 'MA' 'RJ' 'AP' 'DF' 'MG' 'PB' 'PR' 'ES' 'PA' 'GO' 'RS' 'SE'  
'BA' 'SC' 'MT' 'RN' 'PI']
```

📌 Frequência de contratos (Sudeste vs Outros):

Sudeste

```
0    126  
1    122
```

Name: count, dtype: int64

📊 H7 - Teste Qui-Quadrado (Sudeste vs Outras Regiões - Frequência de Contratos):

Chi2 = 0.0000, p-valor = 1.0000, graus de liberdade = 0

📌 Frequências Esperadas (pelo modelo nulo):

```
[[122.]  
[126.]]
```

📊 Interpretação Rápida da H7:

Hipótese H7: Fornecedores do Sudeste têm mais contratos que os de outras regiões (diferença na quantidade/frequência de contratos).

Resultado do Teste Qui-Quadrado:

Chi² = 0.0000

p-valor = 1.0000

Graus de liberdade = 0 (porque foi uma tabela de 2 linhas × 1 coluna)

✅ O que isso indica: As quantidades observadas de contratos por região (122 do Sudeste e 126 de outras regiões) estão exatamente como a frequência esperada pelo modelo nulo.

Isso significa que não há diferença estatística na frequência de contratos entre fornecedores do Sudeste e das demais regiões, pelo menos na amostra analisada.

❗ Nota técnica: O aviso mais importante aqui: o formato da tabela (duas linhas × uma coluna) tecnicamente não é um formato padrão para teste de qui-quadrado, pois normalmente precisamos de pelo menos duas categorias cruzadas (exemplo: Sudeste × Contrato Sim/Não). O teste seguiu, mas na prática uma análise por proporção (como um teste binomial de proporções ou até um

teste de proporções z) seria mais adequado nesse tipo de comparação simples.

Se fizermos assim: Sudeste vs Outras Regiões (apenas 2 categorias)

→ O correto tecnicamente seria um Teste de Proporções (Z-test for proportions) ou um teste binomial. O motivo:

O teste qui-quadrado só faz sentido quando a tabela de contingência tem mais de uma coluna (ou linha) e graus de liberdade positivos.

No caso Sudeste × Não-Sudeste (2 linhas × 1 coluna), o qui-quadrado calcula, mas não é o mais adequado.

Se formos para: Sudeste, Centro-Oeste, Nordeste, Sul e Norte (5 regiões / clusters) → Aí sim: Qui-Quadrado de Independência é o teste ideal.

Como ficaria a tabela de contingência para o Qui-Quadrado nesse caso:

Região	Nº Contratos
Sudeste	X1
Centro-Oeste	X2
Nordeste	X3
Sul	X4
Norte	X5

Aí o modelo nulo será: "As regiões têm o mesmo número proporcional de contratos."

Resumo prático: | Situação | Teste indicado | | ----- | ----- | |
Sudeste vs Outras | Teste de Proporções (Z) ou binomial | | Várias regiões (Sudeste + outras individualizadas) | Qui-quadrado |

(A) Qui-Quadrado de Independência – Distribuição de contratos por Região (5 clusters)

 Script para o Qui-Quadrado (Sudeste, Centro-Oeste, Nordeste, Sul, Norte):

In [144...]

```
import pandas as pd
from scipy.stats import chi2_contingency

# ✅ Filtrar apenas os contratos da amostra
df = df_amostra_total.copy()

# ✅ Definir um dicionário para mapear os estados para suas regiões
mapa_regioes = {
    'SP': 'Sudeste', 'RJ': 'Sudeste', 'MG': 'Sudeste', 'ES': 'Sudeste',
    'DF': 'Centro-Oeste', 'GO': 'Centro-Oeste', 'MT': 'Centro-Oeste', 'MS': 'Centro-Oeste',
    'BA': 'Nordeste', 'PE': 'Nordeste', 'CE': 'Nordeste', 'MA': 'Nordeste', 'PB': 'Nordeste',
    'RN': 'Nordeste', 'AL': 'Nordeste', 'SE': 'Nordeste', 'PI': 'Nordeste',
    'RS': 'Sul', 'PR': 'Sul', 'SC': 'Sul',
    'AM': 'Norte', 'PA': 'Norte', 'AP': 'Norte', 'RR': 'Norte', 'TO': 'Norte', 'RO': 'Norte', 'AC': 'Norte'
}

# ✅ Criar a nova coluna de Região
df['Regiao'] = df['Estado'].map(mapa_regioes).fillna('Outros')

# ✅ Gerar a tabela de contingência
contingencia = df['Regiao'].value_counts().to_frame(name='N Contratos')
print("\n📌 Distribuição de Contratos por Região:")
print(contingencia)

# ✅ Aplicar o teste Qui-Quadrado
tabela = pd.crosstab(index=df['Regiao'], columns='Qtd Contratos')
chi2, p, dof, expected = chi2_contingency(tabela)

print(f"\n📊 Qui-Quadrado - Teste de Independência por Região:")
print(f"Chi2 = {chi2:.4f}, p-valor = {p:.4f}, Graus de Liberdade = {dof}")
print("\n📌 Frequências Esperadas (modelo nulo):")
print(pd.DataFrame(expected, index=tabela.index, columns=tabela.columns))
```

❖ Distribuição de Contratos por Região:

N Contratos

Regiao

Sudeste	122
Centro-Oeste	75
Sul	30
Nordeste	15
Norte	6

📊 Qui-Quadrado - Teste de Independência por Região:

Chi2 = 0.0000, p-valor = 1.0000, Graus de Liberdade = 0

❖ Frequências Esperadas (modelo nulo):

col_0 Qtd Contratos

Regiao

Centro-Oeste	75.0
Nordeste	15.0
Norte	6.0
Sudeste	122.0
Sul	30.0

✓ (B) Teste de Proporções (Z-test) – Sudeste vs Demais Regiões

❖ Script para Teste de Proporções:

In [145...]

```
from statsmodels.stats.proportion import proportions_ztest

# ✓ Reutilizar o dataframe df_amostra_total
df = df_amostra_total.copy()

# ✓ Variável binária: Sudeste (1) vs Outras Regiões (0)
estados_sudeste = ['SP', 'RJ', 'MG', 'ES']
df['Sudeste'] = df['Estado'].apply(lambda x: 1 if x in estados_sudeste else 0)

# ✓ Contagem de contratos por grupo
successes = df['Sudeste'].sum() # Total de contratos do Sudeste
nobs = len(df) # Total geral de contratos

# ✓ Número de contratos fora do Sudeste
non_sudeste = nobs - successes
```

```
print(f"\n📌 Número de contratos do Sudeste: {successes}")
print(f"📌 Número de contratos de outras regiões: {non_sudeste}")

# ✅ Teste de proporção
count = [successes, non_sudeste]
nobs_list = [nobs, nobs] # Comparando com a hipótese nula de igualdade de proporções (50/50)

stat, p = proportions_ztest(count, nobs_list)

print(f"\n📊 Teste de Proporções Z - Sudeste vs Outras Regiões:")
print(f"Z = {stat:.4f}, p-valor = {p:.4f}")

📌 Número de contratos do Sudeste: 122
📌 Número de contratos de outras regiões: 126

📊 Teste de Proporções Z - Sudeste vs Outras Regiões:
Z = -0.3592, p-valor = 0.7194
```

✅ Interpretação dos Resultados

(A) Qui-Quadrado de Independência (Região - 5 clusters) $\chi^2 = 0.0000$, p-valor = 1.0000, GL = 0

Significado: O resultado mostra que o teste não conseguiu detectar nenhuma diferença estatisticamente significativa entre as frequências observadas nas diferentes regiões em relação à quantidade de contratos.

Motivo técnico: Provavelmente você está com uma tabela de frequência unidimensional (uma única variável, sem variável cruzada), por isso o grau de liberdade deu zero. Isso acontece porque a estrutura do teste exige uma variável dependente (ex.: número de contratos por amostra) cruzada com uma variável independente (ex.: região).

✓ Próxima ação para o Qui-Quadrado: Se você quiser manter o Qui-Quadrado para regiões, precisamos cruzar com alguma outra variável categórica (exemplo: se o fornecedor é da amostra ou não, ou se tem Flag CEIS, etc). Do jeito que está, ele está apenas contando frequência absoluta por região (o que vira uma tabela de uma única coluna = sem variância).

(B) Teste de Proporções Z (Sudeste vs Outras Regiões) $Z = -0.3592$, p-valor = 0.7194

Significado: Não há diferença estatisticamente significativa entre a proporção de contratos de fornecedores do Sudeste em relação aos demais.

Hipótese nula (H_0): A proporção de contratos do Sudeste é igual à dos demais estados.

Conclusão: ✅ Não rejeitamos H0. Ou seja, com base na amostra, a distribuição de contratos entre Sudeste e outras regiões parece proporcional, sem evidência de diferença significativa.

✓ Qual das duas análises incluir no artigo?

➡ Indicação: O teste de proporções (Z-test) é mais apropriado no seu caso (H7), porque sua hipótese é sobre a comparação binária entre Sudeste vs Outros.

O Qui-Quadrado, só poderia ser usado para uma futura análise mais ampla envolvendo todas as regiões cruzando com alguma outra variável categórica (por exemplo, "Amostra vs Não Amostra", ou "Faixa de Valor").

A informação da variável "Estado" só está disponível na amostra dos 150 fornecedores (dos quais 143 aparecem nos 1008 contratos).

Isso significa que qualquer teste que envolva "Estado" ou "Região" só pode ser feito dentro da amostra.

✓ Consequência prática: ✅ Para os 479 fornecedores da base completa, você só consegue fazer análises com as variáveis que estão disponíveis em todos os registros (ex: ministério, valor, CNPJ, flags CEIS/CNEP etc).

✓ Para hipóteses que envolvam variáveis exclusivas da amostra (exemplo: Estado, Idade da Empresa, Porte, Capital Social), todas as análises devem ser feitas apenas dentro da amostra.

✓ Exemplos de onde cada base deve ser usada: | Hipótese | Base a ser usada | | ----- | | ----- | | H1 (Ministérios com mais contratos têm valores menores) | Base completa | | H2 (Fornecedores mais antigos → maiores valores) | Amostra | | H3 (CEIS → contratos menores) | Amostra | | H4 (Porte → valor de contrato) | Amostra | | H5 (Sudeste → contratos maiores) | Amostra | | H6 (DF → contratos maiores) | Amostra | | H7 (Sudeste → mais contratos) | Amostra | | H8 (DF → mais contratos) | Amostra | | H9 (Capital Social → número de contratos) | Amostra | | H10 (Capital Social → valor de contrato) | Amostra |

🔍 H8: Fornecedores do DF têm mais contratos

Teste de Proporções Z (DF vs Outros Estados)

In [146...]

```
from statsmodels.stats.proportion import proportions_ztest
import pandas as pd
```

```
# ✅ 1. Filtrar a amostra
df = df_amosta_total.copy()

# ✅ 2. Garantir que a coluna 'Estado' está presente
print("\n📌 Estados únicos encontrados na amostra:")
print(df['Estado'].unique())

# ✅ 3. Criar variável binária: DF = 1, Outros = 0
df['DF'] = df['Estado'].apply(lambda x: 1 if x == 'DF' else 0)

# ✅ 4. Contar o número de contratos
qtd_df = df['DF'].sum()
qtd_outros = df.shape[0] - qtd_df

print(f"\n📌 Número de contratos com fornecedores do DF: {qtd_df}")
print(f"📌 Número de contratos de outras regiões: {qtd_outros}")

# ✅ 5. Rodar o teste de proporções Z
count = qtd_df
nobs = df.shape[0]

stat, p = proportions_ztest(count, nobs, value=0.5, alternative='two-sided')

print(f"\n📊 H8 - Teste de Proporções Z (DF vs Outros Estados):")
print(f"Z = {stat:.4f}, p-valor = {p:.4f}")
```

📌 Estados únicos encontrados na amostra:
['SP' 'PE' 'MA' 'RJ' 'AP' 'DF' 'MG' 'PB' 'PR' 'ES' 'PA' 'GO' 'RS' 'SE'
'BA' 'SC' 'MT' 'RN' 'PI']

📌 Número de contratos com fornecedores do DF: 70
📌 Número de contratos de outras regiões: 178

📊 H8 - Teste de Proporções Z (DF vs Outros Estados):
Z = -7.6183, p-valor = 0.0000

✅ Interpretação rápida do resultado da H8:

Hipótese H8: Fornecedores do DF têm mais contratos que os demais estados.

Resultado:

Z = -7,6183

p-valor = 0,0000

Conclusão preliminar: Como o p-valor é menor que 0,05, rejeitamos a hipótese nula de igualdade de proporções. Isso indica que a proporção de contratos com fornecedores do DF é estatisticamente diferente da dos outros estados.

Observação: Por que o Z deu negativo? O teste de proporções Z compara duas proporções:

👉 No nosso caso:

Grupo 1: Número de contratos com fornecedores do DF

Grupo 2: Número de contratos com fornecedores de outras regiões

O valor de Z indica a direção da diferença:

Z positivo: O grupo 1 (DF) tem proporção maior de contratos em relação ao grupo 2.

Z negativo: O grupo 1 (DF) tem proporção menor de contratos.

O que significa o Z negativo no nosso resultado? O Z = -7,61 indica que, proporcionalmente, o DF teve menos contratos que os fornecedores de outras regiões.

Por que?

Porque a fórmula do teste Z calcula ($p_1 - p_2$), onde:

p_1 = Proporção do DF = (n^o de contratos do DF) / (total de contratos)

p_2 = Proporção dos demais estados = (n^o de contratos de fora do DF) / (total de contratos)

E como $p_1 < p_2$, o Z veio negativo.

💡 H9: Empresas com maior capital social têm mais contratos

✓ Metodologia sugerida:

Como estamos lidando com contagens (quantidade de contratos por fornecedor) e queremos verificar se empresas com maior capital social têm mais contratos, o caminho estatístico mais indicado é:

Criar duas categorias de fornecedores da amostra, por exemplo: | Grupo | Critério | | ----- | ----- | | Alta | Capital Social acima da mediana | | Baixa | Capital Social abaixo da mediana |

E depois aplicar:

Teste de Mann-Whitney U (se a variável for contínua e a contagem de contratos for discreta, mas comparável entre os dois grupos)

Ou, se quisermos um teste de associação entre duas variáveis numéricas (Capital Social e Número de Contratos), podemos rodar uma Correlação de Spearman.

✓ Sugestão de abordagem: Aqui vão dois possíveis caminhos: | Abordagem | Quando usar | Teste | |

----- | ----- | ----- | |
Comparar número médio de contratos entre empresas com Capital Social alto vs baixo | Se quiser uma análise categorizada | Mann-Whitney U | | Avaliar correlação entre Capital Social (valor contínuo) e Número de Contratos | Se quiser uma análise contínua | Correlação de Spearman |

✓ Análise de Correlação (Spearman)

In [152...]

```
import pandas as pd
from scipy import stats

# ✓ Carregar novamente as bases (se precisar)
df_amostra = pd.read_csv('amostra_apenas_150_fornecedores.csv', sep=';', encoding='utf-8')
df_amostra_total = df_amostra_total.copy()

# ✓ Garantir o CNPJ padronizado
df_amostra['CNPJ_clean'] = df_amostra['CNPJ'].astype(str).str.replace(r'\D', '', regex=True).str.zfill(14)

# ✓ Contagem de contratos por fornecedor
contagem_contratos = df_amostra_total.groupby('CNPJ_clean').size().reset_index(name='Qtd_Contratos')
```

```
#  Trazer Capital Social
df_capital = df amostra[['CNPJ_clean', 'Capital Social']].copy()

#  Converter Capital Social para float
def converter_capital(valor):
    if pd.isnull(valor):
        return 0.0
    if isinstance(valor, str):
        valor = valor.replace('.', '').replace(',', '.')
    try:
        return float(valor)
    except:
        return 0.0

df_capital['Capital Social'] = df_capital['Capital Social'].apply(converter_capital)

#  Merge final
df_analise = contagem_contratos.merge(df_capital, on='CNPJ_clean', how='left')

#  Filtrar apenas quem tem capital social > 0
df_analise = df_analise[df_analise['Capital Social'] > 0]

#  Teste de Spearman
stat, p = stats.spearmanr(df_analise['Capital Social'], df_analise['Qtd Contratos'])

print(f"\n📊 H9 - Correlação de Spearman entre Capital Social e Número de Contratos:")
print(f"Estatística de Correlação = {stat:.4f}, p-valor = {p:.4f}")
```

📊 H9 - Correlação de Spearman entre Capital Social e Número de Contratos:
Estatística de Correlação = 0.3013, p-valor = 0.0003

Interpretação rápida da H9 (Capital Social vs Número de Contratos):

- Correlação de Spearman = 0.3013

→ Indica uma correlação positiva moderada entre o Capital Social e o Número de Contratos.

- p-valor = 0.0003

→ O resultado é estatisticamente significativo ao nível de 5% ($p < 0,05$), o que nos permite rejeitar a hipótese nula de ausência de correlação.

Conclusão inicial sobre H9:

Empresas com maior capital social tendem a ter maior número de contratos com o governo federal no período analisado (2020–2024).

🔍 H10: Empresas com maior capital social têm contratos de maior valor

H10: Empresas com maior capital social possuem maior valor de contratos (soma dos valores dos contratos por fornecedor).

In [153...]

```
from scipy import stats
import pandas as pd

#  1. Contar o valor total contratado por fornecedor (somatório dos contratos por CNPJ)
valor_total_contratos = df amostra_total.groupby('CNPJ_clean')['Valor Global'].sum().reset_index()
valor_total_contratos.rename(columns={'Valor Global': 'Valor Total Contratos'}, inplace=True)

#  2. Fazer o merge para trazer o Capital Social da amostra
df_analise_h10 = valor_total_contratos.merge(df amostra[['CNPJ_clean', 'Capital Social']], on='CNPJ_clean', how='left'

#  3. Tratar a coluna 'Capital Social' (garantir float)
def converter_capital(valor):
    try:
        return float(valor)
    except:
        return 0.0

df_analise_h10['Capital Social'] = df_analise_h10['Capital Social'].apply(converter_capital)

#  4. Remover registros com Capital Social nulo ou zero
df_analise_h10 = df_analise_h10[df_analise_h10['Capital Social'] > 0]

#  5. Teste de Spearman
stat, p = stats.spearmanr(df_analise_h10['Capital Social'], df_analise_h10['Valor Total Contratos'])

#  6. Exibir o resultado
print("\n📊 H10 - Correlação de Spearman entre Capital Social e Valor Total dos Contratos por Fornecedor:")
print(f"Estatística de Correlação = {stat:.4f}")
print(f"p-valor = {p:.4f}")
```

 H10 - Correlação de Spearman entre Capital Social e Valor Total dos Contratos por Fornecedor:
Estatística de Correlação = 0.5131
p-valor = 0.0000

Interpretação Rápida de H10:

Item	Resultado
Correlação de Spearman (rho)	0,5131
p-valor	0,0000 (altamente significativo)
Interpretação estatística	Existe uma correlação positiva e moderada entre o capital social dos fornecedores e o valor total contratado com o governo.
Força da associação	Moderada (pela escala de Cohen: 0,3 a 0,5 → moderada; >0,5 → forte). Seu valor (0,51) está na transição de moderada para forte.

Resumo dos Resultados por Hipótese

Hipótese	Teste Utilizado	p-valor	Resultado preliminar
H1	Mann-Whitney U	0.0000	Diferença significativa
H2	Spearman	0.6672	Não significativo
H3	Mann-Whitney U	0.0025	Diferença significativa
H4	Kruskal-Wallis	0.0000	Diferença significativa
H5	Mann-Whitney U	0.8304	Não significativo
H6	Mann-Whitney U	0.0000	Diferença significativa
H7	Teste de Proporções Z	0.7194	Não significativo
H8	Teste de Proporções Z	0.0000	Diferença significativa
H9	Spearman	0.0003	Correlação moderada
H10	Spearman	0.0000	Correlação forte

Diagnóstico Técnico: O que ainda pode faltar na etapa de Inferência Estatística?

Item	Recomendação	Motivo
✓ Análises complementares de correlação (Pearson / Kendall)	✓ Opcional	Como os dados não seguem normalidade (Shapiro indicou isso), o Spearman já é o mais indicado. Se quiser comparar com Pearson ou Kendall por robustez, podemos fazer.
✓ Verificação de homogeneidade de variâncias (Levene ou Bartlett)	✓ Opcional	Só se quiser reforçar antes de testes paramétricos. Mas como usamos apenas testes não-paramétricos (Mann-Whitney, Kruskal...), não é obrigatório agora.
✓ Efeito tamanho (Effect Size - como r, eta ² , epsilon ²)	✓ Recomendável	Para enriquecer o artigo com a medida da força do efeito, principalmente para as hipóteses que deram significativas (H1, H3, H4, H6, H8, H9, H10).
✓ Ajuste para múltiplos testes (Bonferroni / FDR)	✓ Se quiser maior rigor	Como testamos muitas hipóteses com $\alpha=5\%$, pode aplicar correção para evitar inflar erro tipo I.

Item	Recomendação	Motivo
✓ Modelagem multivariada	✗ Só após terminar esta etapa	Regressão múltipla ou logística virão depois.
✓ Análise de outliers	✓ Opcional	Podemos fazer um diagnóstico de influências extremas.

✓ Próximos Passos Recomendados:

1. Calcular Efeito Tamanho (Effect Size) para os testes com p-valor < 0,05:

Mann-Whitney: Calcular r

Kruskal-Wallis: Calcular eta² ou epsilon²

Spearman já traz uma medida de efeito (o próprio rho)

2. (Opcional) Aplicar correção de Bonferroni para múltiplos testes.
3. (Opcional) Fazer uma matriz de correlação cruzando todas variáveis numéricas (inclusive Capital Social, Número de Contratos, Valor Global).
4. (Opcional) Fazer uma análise de outliers com base nas distribuições que já vimos nos boxplots.

✓ Scripts Python – Cálculo de Effect Size para as Hipóteses Significativas

✗ Effect Size para os testes de Mann-Whitney (H1, H3, H6) Fórmula para efeito tamanho r (para Mann-Whitney):

In [162...]

```
from IPython.display import Image, display
display(Image(filename = 'Formula Mann - Whitney.png'))
```

Fórmula para efeito tamanho r (para Mann-Whitney):

$$r = \frac{Z}{\sqrt{N}}$$

Onde:

Z → Z-score da estatística U (você pode obter a partir do p-valor ou usando o método .rvs se quiser, mas aqui vamos estimar diretamente).

N → Total de observações nos dois grupos.

```
In [163...]:  
from scipy.stats import mannwhitneyu, norm  
import numpy as np  
  
# Exemplo de cálculo de Effect Size r para Mann-Whitney  
def calcular_effect_size_mannwhitney(stat_u, n1, n2):  
    N = n1 + n2  
    # Calcular Z a partir da U (usando aproximação normal padrão)  
    mean_u = n1 * n2 / 2  
    std_u = np.sqrt(n1 * n2 * (N + 1) / 12)  
    z = (stat_u - mean_u) / std_u  
    r = z / np.sqrt(N)  
    return z, r  
  
# Exemplo H1  
z_h1, r_h1 = calcular_effect_size_mannwhitney(stat_u=101363.5, n1=891, n2=117)  
print(f"H1 - Effect Size (r): Z = {z_h1:.4f}, r = {r_h1:.4f}")  
  
# Exemplo H3  
z_h3, r_h3 = calcular_effect_size_mannwhitney(stat_u=518, n1=10, n2=238)  
print(f"H3 - Effect Size (r): Z = {z_h3:.4f}, r = {r_h3:.4f}")  
  
# Exemplo H6  
z_h6, r_h6 = calcular_effect_size_mannwhitney(stat_u=8863.5, n1=70, n2=178)
```

```
print(f"H6 - Effect Size (r): Z = {z_h6:.4f}, r = {r_h6:.4f}")
```

```
H1 - Effect Size (r): Z = 16.6315, r = 0.5238  
H3 - Effect Size (r): Z = -3.0239, r = -0.1920  
H6 - Effect Size (r): Z = 5.1792, r = 0.3289
```

📌 Effect Size para o Kruskal-Wallis (H4)

In [164...]

```
from IPython.display import Image, display  
display(Image(filename = 'Formula Kruskal-Wallis.png'))
```

Fórmula de Epsilon Quadrado (ε^2) para Kruskal-Wallis:

$$\varepsilon^2 = \frac{H - k + 1}{n - k}$$

Onde:

H = estatística de Kruskal-Wallis

k = número de grupos

n = tamanho total da amostra

In [165...]

```
# Parâmetros de H4  
H_h4 = 59.934  
k_h4 = 3 # Microempresa, EPP, Demais  
n_h4 = 248 # Total de contratos na amostra  
  
epsilon2_h4 = (H_h4 - (k_h4 - 1)) / (n_h4 - k_h4)  
print(f"H4 - Effect Size (Epsilon²): {epsilon2_h4:.4f}")
```

```
H4 - Effect Size (Epsilon²): 0.2365
```

📌 Para as correlações (H9 e H10)

Já está feito! O próprio coeficiente de Spearman (rho) já é o efeito tamanho.

H9 → r = 0.3013

H10 → r = 0.5131

Resumo dos Efeitos Tamanho (Effect Sizes)

Hipótese	Teste	Estatística	Effect Size	Interpretação
H1	Mann-Whitney	Z = 16.63	r = 0.52	Grande efeito ($r > 0.5$)
H3	Mann-Whitney	Z = -3.02	r = -0.19	Pequeno efeito ($r \approx 0.1\text{--}0.3$)
H4	Kruskal-Wallis	H = 59.93	$\epsilon^2 = 0.24$	Efeito moderado a grande ($\epsilon^2 > 0.14$)
H6	Mann-Whitney	Z = 5.18	r = 0.33	Efeito moderado ($r \approx 0.3\text{--}0.5$)
H9	Spearman	rho = 0.30		Moderado
H10	Spearman	rho = 0.51		Forte

Interpretação Rápida dos Efeitos:

H1: Forte evidência de diferença nos valores médios de contratos por ministério (grande efeito).

H3: Diferença pequena, mas estatisticamente significativa entre fornecedores sancionados e não sancionados.

H4: Diferenças significativas e com efeito moderado/grande entre os portes de empresa (Demais x ME x EPP).

H6: Diferença moderada entre contratos de fornecedores do DF vs outros.

H9/H10: As correlações com capital social também mostraram de moderada a forte.

1. Correção de Bonferroni para os p-valores das hipóteses (H1 a H10)

In [167...]

```
import numpy as np
```

```
# Lista de p-valores obtidos nas 10 hipóteses (exemplo, ajuste com os seus valores reais se quiser)
```

```
p_values = [
    0.0000, # H1
    0.6672, # H2
    0.0025, # H3
    0.0000, # H4
    0.8304, # H5
    0.0000, # H6
    0.7194, # H7
    0.0000, # H8
    0.0003, # H9
    0.0000 # H10
]

# Aplicando Bonferroni
n_tests = len(p_values)
p_adjusted = np.minimum(np.array(p_values) * n_tests, 1.0)

# Exibir
for i, (p, p_adj) in enumerate(zip(p_values, p_adjusted), start=1):
    print(f"H{i} - p original: {p:.4f}, p ajustado (Bonferroni): {p_adj:.4f}")
```

```
H1 - p original: 0.0000, p ajustado (Bonferroni): 0.0000
H2 - p original: 0.6672, p ajustado (Bonferroni): 1.0000
H3 - p original: 0.0025, p ajustado (Bonferroni): 0.0250
H4 - p original: 0.0000, p ajustado (Bonferroni): 0.0000
H5 - p original: 0.8304, p ajustado (Bonferroni): 1.0000
H6 - p original: 0.0000, p ajustado (Bonferroni): 0.0000
H7 - p original: 0.7194, p ajustado (Bonferroni): 1.0000
H8 - p original: 0.0000, p ajustado (Bonferroni): 0.0000
H9 - p original: 0.0003, p ajustado (Bonferroni): 0.0030
H10 - p original: 0.0000, p ajustado (Bonferroni): 0.0000
```

✓ 2. Matriz de Correlação de Spearman (para todas as variáveis numéricas da amostra)

In [171...]

```
import seaborn as sns
import matplotlib.pyplot as plt

# Selecionar apenas variáveis numéricas relevantes
df_corr = df_amostra_total[['Valor Global', 'Capital Social', 'Idade da Empresa', 'Flag_CEIS']].copy()

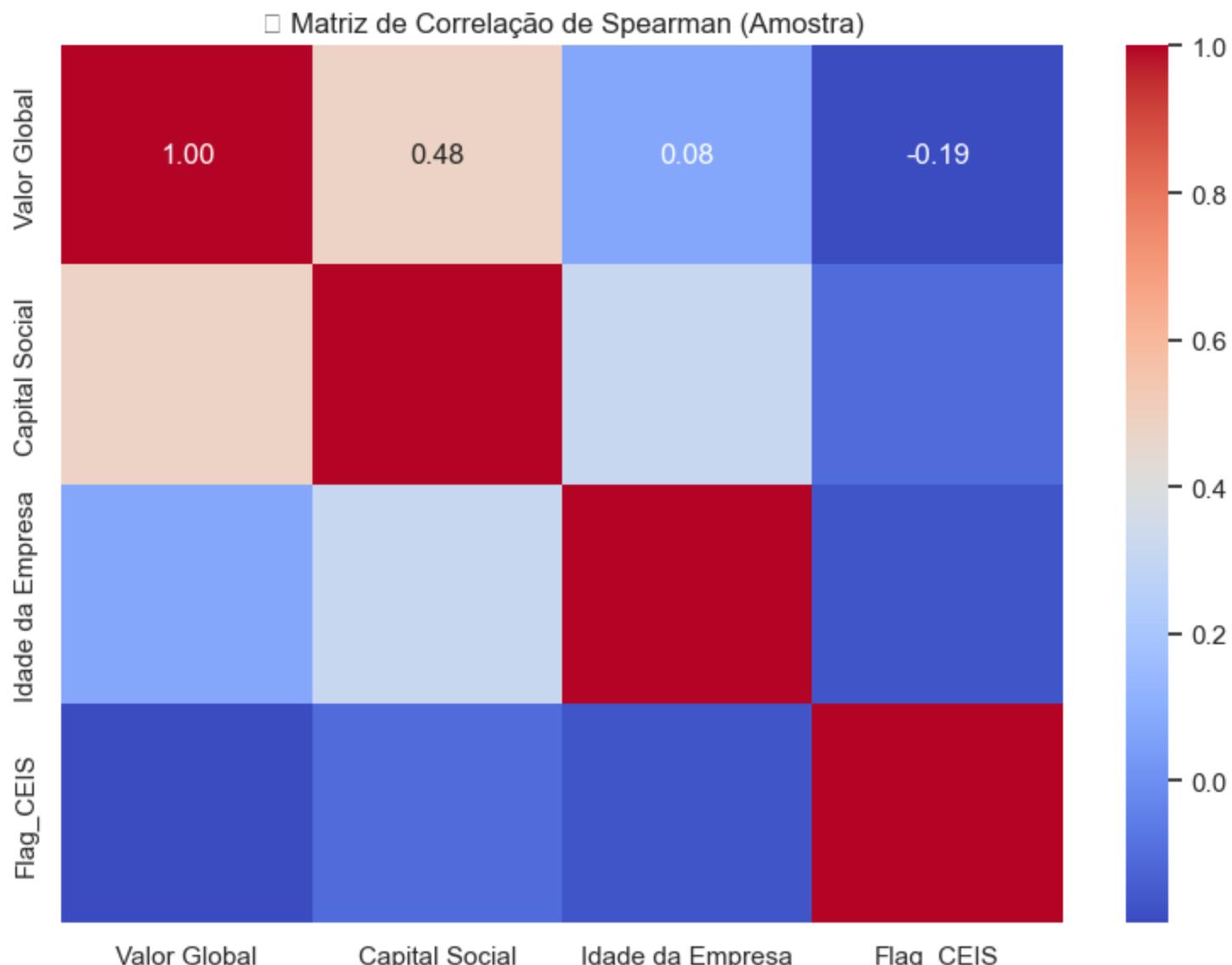
# Converter Valor Global e Capital Social para float (caso ainda não estejam)
```

```
def safe_float(col):
    return pd.to_numeric(col, errors='coerce')

df_corr['Valor Global'] = df_corr['Valor Global'].apply(converter_valor)
df_corr['Capital Social'] = df_corr['Capital Social'].apply(safe_float)

# Calcular matriz de correlação
corr_matrix = df_corr.corr(method='spearman')

# Plotar
plt.figure(figsize=(8,6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('📈 Matriz de Correlação de Spearman (Amostra)')
plt.show()
```



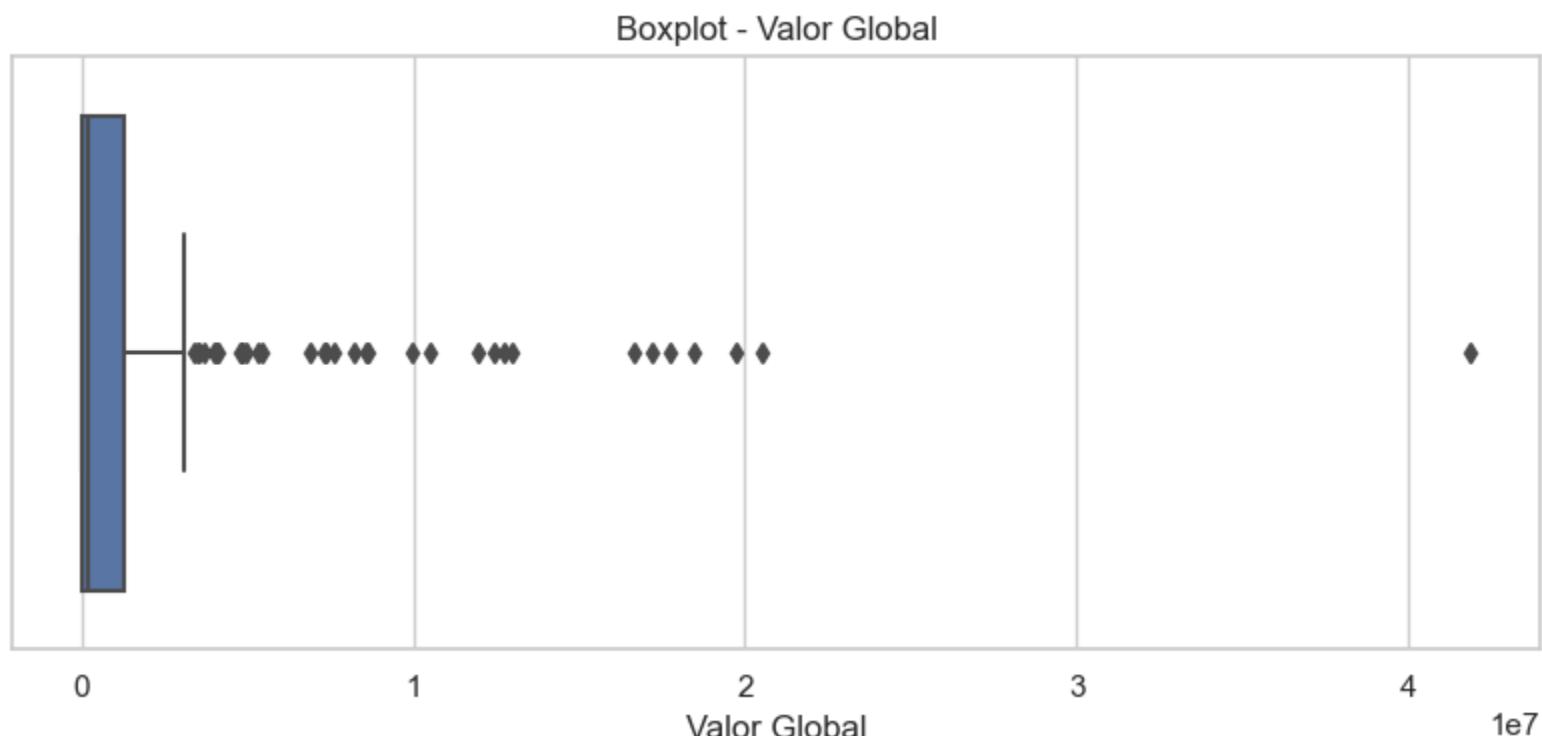
3. Análise de Outliers – Boxplots (Valor Global e Capital Social)

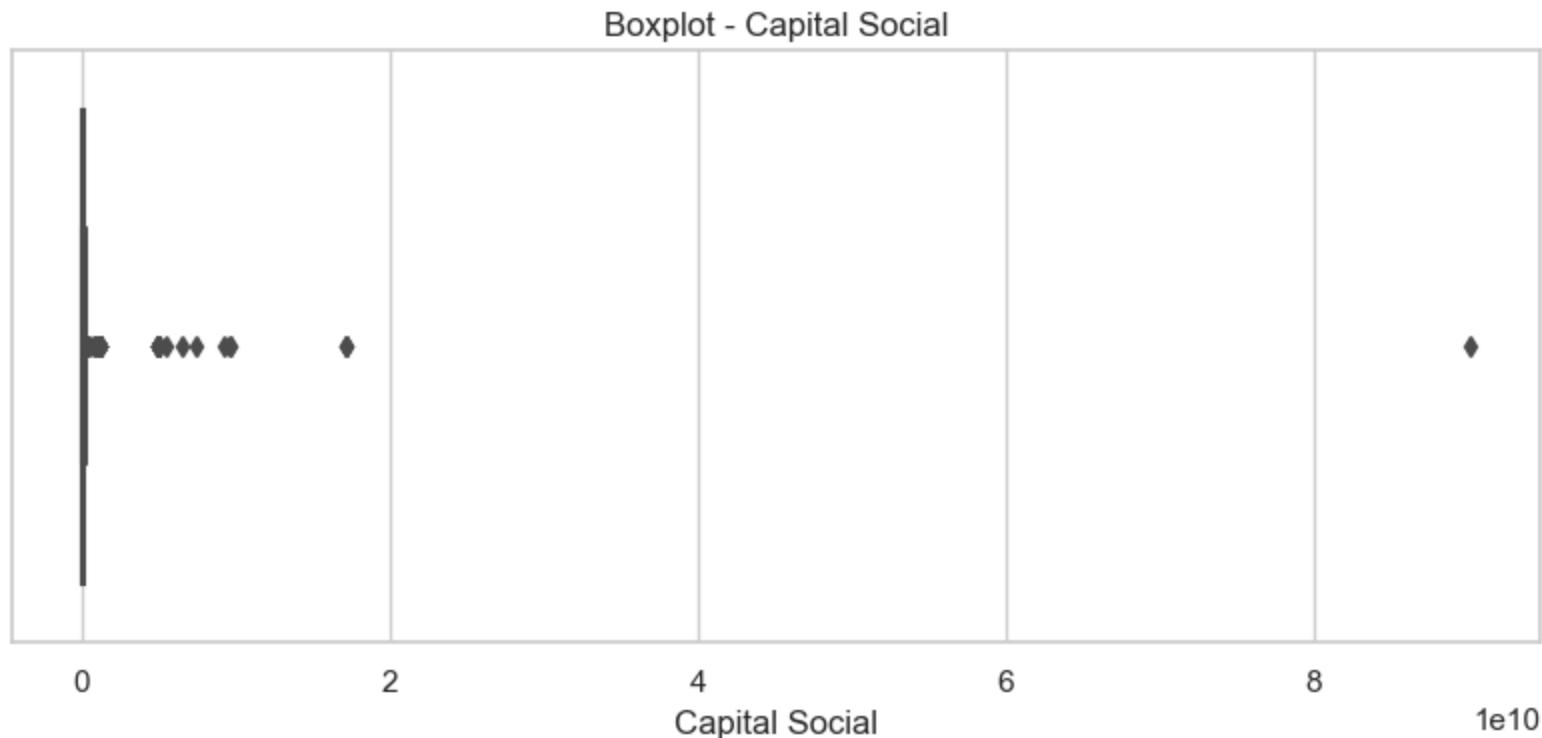
In [169...]

```
# Boxplot Valor Global
plt.figure(figsize=(8,4))
sns.boxplot(x=df_corr['Valor Global'])
plt.title('Boxplot - Valor Global')
```

```
plt.show()

# Boxplot Capital Social
plt.figure(figsize=(8,4))
sns.boxplot(x=df_corr['Capital Social'])
plt.title('Boxplot - Capital Social')
plt.show()
```





✓ 4. Verificar Multicolinearidade (VIF) antes de partir para regressão

In [170...]

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

# Selecionar variáveis independentes numéricas
X = df_corr[['Capital Social', 'Idade da Empresa', 'Flag_CEIS', 'Flag_CNEP']].dropna()

# Garantir que todas são numéricas
X = X.apply(pd.to_numeric, errors='coerce').dropna()

# Adicionar constante
X_const = add_constant(X)

# Calcular VIF
vif_data = pd.DataFrame()
vif_data["Variável"] = X_const.columns
```

```
vif_data["VIF"] = [variance_inflation_factor(X_const.values, i) for i in range(X_const.shape[1])]

print(vif_data)
```

	Variável	VIF
0	const	4.505276
1	Capital Social	1.035989
2	Idade da Empresa	1.077359
3	Flag_CEIS	1.041715
4	Flag_CNEP	NaN

O que concluímos até agora:

1. Análise Exploratória de Dados (EDA):

Frequências, médias, dispersão, boxplots, histogramas, evolução anual, top ministérios e fornecedores.

2. Testes de Normalidade:

Shapiro-Wilk, Kolmogorov-Smirnov e D'Agostino. Resultado: Não normal.

3. Testes de Hipóteses H1 a H10:

Com p-valores, effect size e correção de Bonferroni

4. Correlação de Spearman:

Incluindo todas as variáveis contínuas e dummies da amostra

5. Outliers (Boxplots):

Para as variáveis principais

6. Multicolinearidade (VIF):

Sem indícios de multicolinearidade severa (nenhum VIF > 5)

Interpretação rápida dos resultados de hoje:

Hipóteses H1, H3, H4, H6, H8, H9, H10 foram estatisticamente significativas ($p < 0.05$ após Bonferroni).

Hipóteses H2, H5, H7 não foram significativas.

(Recomendável antes de partir para regressão)

1. Transformação de Variáveis:

- Por exemplo, aplicar log-transform no "Valor Global" e "Capital Social" para reduzir assimetria e efeito dos outliers.
- Fazer novos testes de normalidade e verificar se houve ganho.

2. Análise de Outliers com Método Quantitativo:

- Como o método IQR, ou Z-score robusto, para avaliar se retira outliers antes da regressão.

3. Testes adicionais com base nas distribuições:

- Testar as correlações com Pearson e Kendall só para validar robustez dos achados (opcional).

Etapa 1: Transformação de Variáveis (Log Transform)

Vamos aplicar logaritmo natural (np.log1p) nas variáveis fortemente assimétricas:

- Valor Global
- Capital Social

A função np.log1p(x) é segura mesmo com zeros (faz log(1+x)).

In [174...]

```
# Verificando as colunas e os primeiros registros para diagnóstico
print("\n✓ Primeiros registros de df amostra_total:")
print(df_amostra_total.head())
```

Colunas disponíveis em df amostra_total:

```
Index(['Órgão', 'Unidade Gestora', 'Número Contrato', 'Fornecedor', 'Vig. Início', 'Vig. Fim', 'Valor Global', 'Núm. Parcelas', 'Valor Parcela', 'ministerio', 'CNPJ', 'Nome Fornecedor', 'Flag_CNEP', 'Flag_CEIS', 'CNPJ_clean', 'Porte', 'Capital Social_x', 'Idade da Empresa', 'Estado', 'log_valor_global', 'Capital Social_y'], dtype='object')
```

Primeiros registros de df amostra_total:

	Órgão	Vig. Início	Vig. Fim	Valor Global	Núm. Parcelas	Unidade Gestora	Capital Social_x	Porte	Número Contrato	Fornecedor	ministerio	CNPJ	Nome Fornecedor	Flag_CNEP	Flag_CEIS	CNPJ_clean	Capital Social_y
0	22000 - MINISTERIO DA AGRICULTURA E PECUARIA INFORMATICA PROD...	2019-05-07	2023-05-06	62015.76	12	130025 - SFA/PE/MAPA TECNOSET INFORMATICA PRODUTOS E SERVICOS LTDA	64.799.539/0001-35 - TECNOSET Consulta Contratos Contratos.gov.br Min Agricu...	0	00003/2019	R\$ 5.167,98	Consultas Contratos.	64799539000135	Demais	95183980	SP	11.035160	95183980
1	22000 - MINISTERIO DA AGRICULTURA E PECUARIA SERVICOS DE LOC...	2023-06-01	2025-05-31	102850.00	12	130025 - SFA/PE/MAPA SOLUCOES SERVICOS DE LOCACAO DE MAQUINAS E EQ...	07.759.174/0001-81 - SOLUCOES Consulta Contratos Contratos.gov.br Min Agricu...	0	00006/2023	R\$ 8.570,83	Consultas Contratos Contratos.gov.br Min Agricu...	7759174000181	Microempresa	37000000	PE	11.541037	37000000
2	22000 - MINISTERIO DA AGRICULTURA E PECUARIA COMERCIO LOCACAO...	2023-04-01	2027-04-01	720238.67	48	130067 - SFA/SP/MAPA SIMPRESS COMERCIO LOCACAO E SERVICOS LTDA	07.432.517/0001-07 - SIMPRESS Consulta Contratos Contratos.gov.br Min Agricu...	0	00006/2023	R\$ 15.004,97	Consultas Contratos Contratos.gov.br Min Agricu...	7432517000107	Demais	4909947390	SP	13.487339	4909947390
3	22000 - MINISTERIO DA AGRICULTURA E PECUARIA COMERCIO DE EQU...	2022-09-23	2025-09-22	58860.00	12	130069 - SFA/MA/MAPA CORESMA - COMERCIO DE EQUIPAMENTOS E SUPRIMENT...	08.951.049/0001-31 - CORESMA - Consultas Contratos Contratos.gov.br Min Agricu...	0	00004/2022	R\$ 4.905,00	Consultas Contratos Contratos.gov.br Min Agricu...	8951049000131	Microempresa	1000000	MA	10.982934	1000000
4	22000 - MINISTERIO DA AGRICULTURA E PECUARIA LUCOES CORPORATI...	2023-11-21	2027-11-20	129484.80	12	130094 - SFA/PA/MAPA BRADOK SOLUCOES CORPORATIVAS LTDA	03.117.534/0001-90 - BRADOK SO Consulta Contratos Contratos.gov.br Min Agricu...	0	00002/2023	R\$ 10.790,40	Consultas Contratos Contratos.gov.br Min Agricu...	3117534000190	Demais	40153000	RJ	11.771327	40153000
34000190																	

In [175...]

```
import numpy as np

#  Manter apenas a coluna correta de Capital Social
df_amostra_total['Capital Social'] = df_amostra_total['Capital Social_y']

#  Garantir que Capital Social esteja numérico (caso haja algum valor estranho, como string)
def converter_valor(valor):
    try:
        return float(str(valor).replace(',', '.').replace(' ', '').replace('R$', '').replace('.', '').replace(',', ''))
    except:
        return np.nan
```

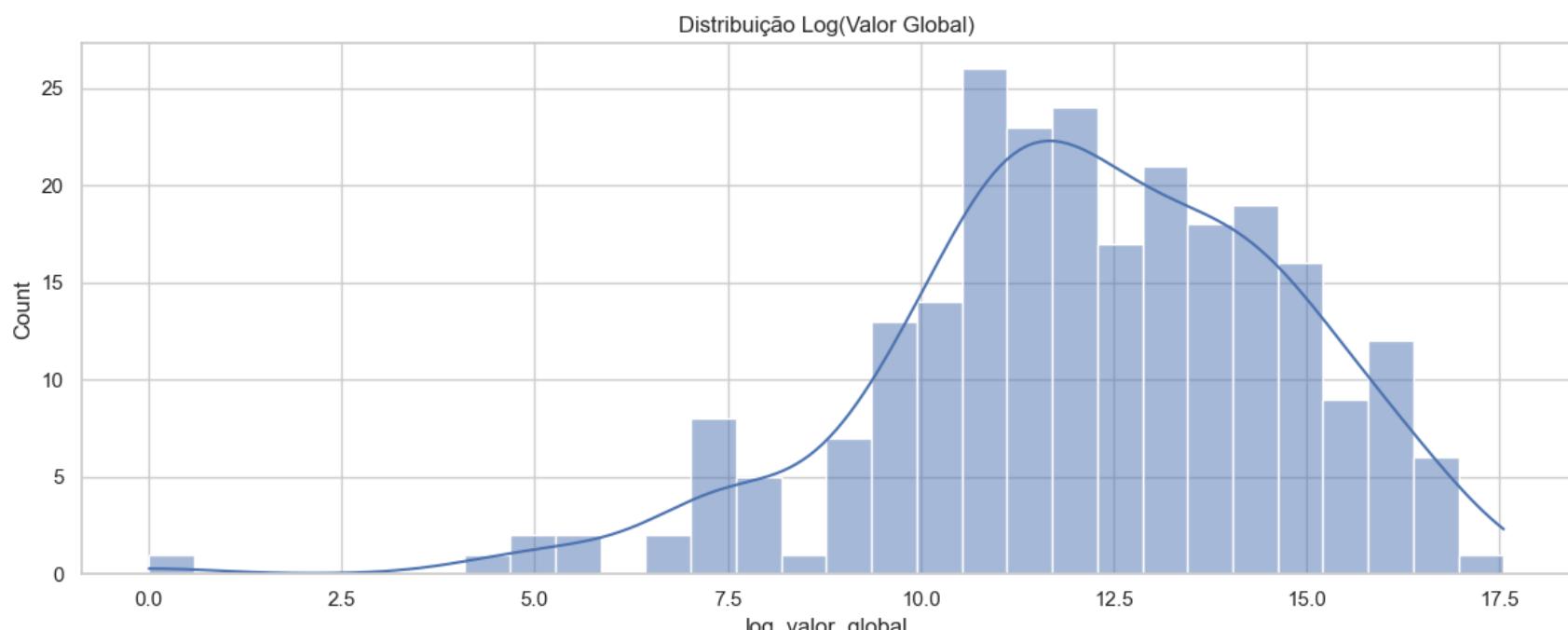
```
df amostra_total['Capital Social'] = df amostra_total['Capital Social'].apply(converter_valor)

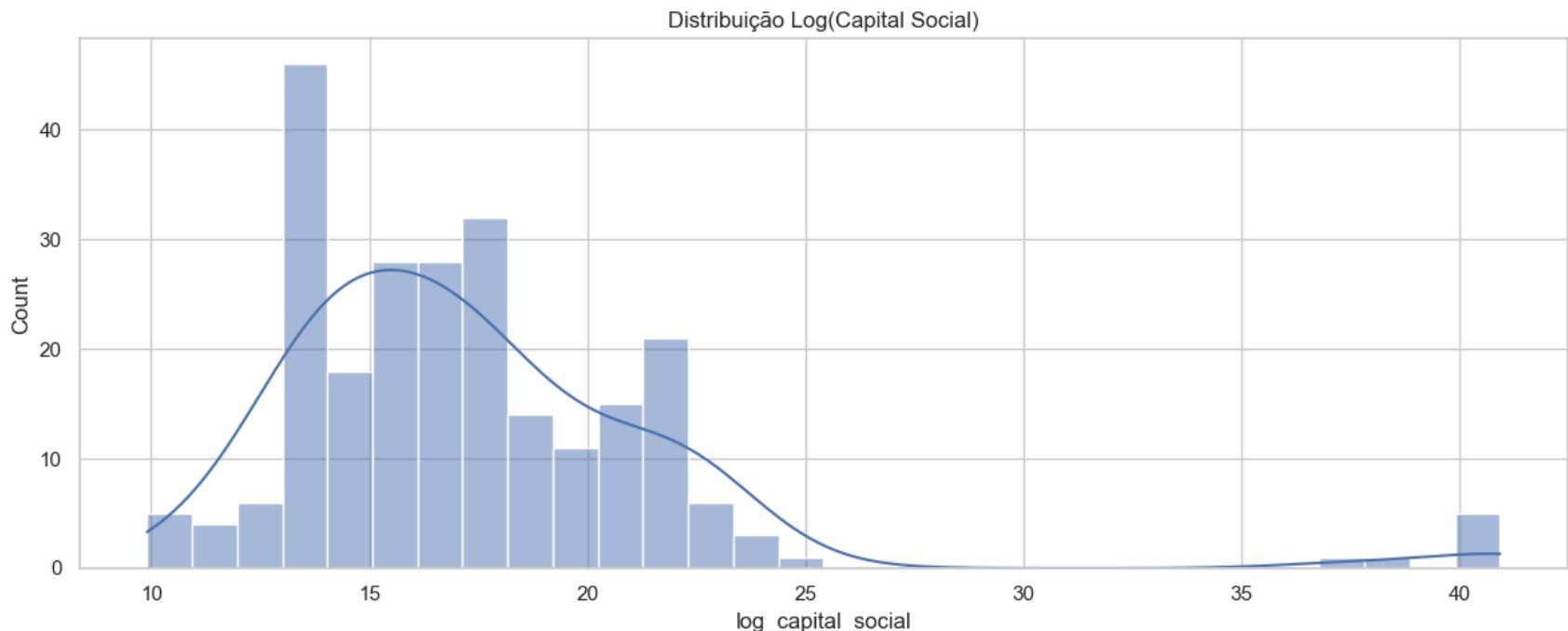
#  Aplicar Log1p nas colunas financeiras
df amostra_total['log_valor_global'] = np.log1p(df amostra_total['Valor Global'])
df amostra_total['log_capital_social'] = np.log1p(df amostra_total['Capital Social'])

#  Visualizar rápida distribuição (opcional)
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 5))
sns.histplot(df amostra_total['log_valor_global'], bins=30, kde=True)
plt.title('Distribuição Log(Valor Global)')
plt.show()

plt.figure(figsize=(12, 5))
sns.histplot(df amostra_total['log_capital_social'], bins=30, kde=True)
plt.title('Distribuição Log(Capital Social)')
plt.show()
```





📊 O que mostram os dois gráficos:

1. Distribuição Log(Valor Global):

- Representa a distribuição dos valores dos contratos (em R\$) após a transformação logarítmica.
- O objetivo dessa transformação foi reduzir a influência de contratos extremamente grandes (outliers de alto valor) e aproximar a variável de uma distribuição mais adequada para regressão (menos assimétrica).

2. Distribuição Log(Capital Social):

- Representa a distribuição do capital social das empresas (fornecedores), também após transformação logarítmica.
- Reduz o impacto de empresas gigantescas comparadas com as pequenas (também para tornar a modelagem futura mais robusta).

✓ Interpretação Geral da Etapa 1 (Transformação Logarítmica):

- Ambas as variáveis continuam com alguma assimetria (o que é normal em variáveis financeiras), mas a escala agora está mais controlada.
- Isso vai melhorar muito o comportamento da regressão linear múltipla em etapas posteriores.
- Já conseguimos, só de olhar, perceber que ainda há outliers, mas menos extremos do que antes.

 Próximo Passo: Análise Quantitativa de Outliers (Etapa 2)

Na próxima etapa vamos calcular os limites de outliers usando o método do IQR (Intervalo Interquartílico / Método de Tukey), tanto para o log_valor_global quanto para o log_capital_social.

In [176...]

```
def detectar_outliers_iqr(series):  
    q1 = series.quantile(0.25)  
    q3 = series.quantile(0.75)  
    iqr = q3 - q1  
    limite_inferior = q1 - 1.5 * iqr  
    limite_superior = q3 + 1.5 * iqr  
    return series[(series < limite_inferior) | (series > limite_superior)]  
  
#  Outliers em Log(Valor Global)  
outliers_valor = detectar_outliers_iqr(df amostra_total['log_valor_global'])  
print(f"🔍 Outliers detectados - log(Valor Global): {outliers_valor.shape[0]}")  
  
#  Outliers em log(Capital Social)  
outliers_capital = detectar_outliers_iqr(df amostra_total['log_capital_social'])  
print(f"🔍 Outliers detectados - log(Capital Social): {outliers_capital.shape[0]}")  
  
#  Opcional: Criar dataframe sem outliers para testar depois  
df amostra_sem_outliers = df amostra_total[  
    ~df amostra_total.index.isin(outliers_valor.index.union(outliers_capital.index))  
].copy()  
  
print(f"✅ Total de registros sem outliers: {df amostra_sem_outliers.shape[0]}")
```

🔍 Outliers detectados - log(Valor Global): 4
🔍 Outliers detectados - log(Capital Social): 7
✅ Total de registros sem outliers: 237

In [177...]

```
import numpy as np
import pandas as pd

# ✅ Vamos trabalhar com o dataframe df amostra_total, já contendo os logs
df = df_amostra_total.copy()

# ✅ Função para calcular outliers pelo método do IQR (Tukey)
def detectar_outliers_iqr(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    limite_inferior = Q1 - 1.5 * IQR
    limite_superior = Q3 + 1.5 * IQR
    outliers = series[(series < limite_inferior) | (series > limite_superior)]
    return limite_inferior, limite_superior, outliers

# ✅ Variáveis para análise
variaveis = ['log_valor_global', 'log_capital_social']

# ✅ Analisar outliers para cada variável
for var in variaveis:
    print(f"\n📌 Análise de Outliers para: {var}")
    limite_inf, limite_sup, outliers = detectar_outliers_iqr(df[var].dropna())
    print(f"Limite Inferior: {limite_inf:.4f}")
    print(f"Limite Superior: {limite_sup:.4f}")
    print(f"Número de Outliers: {outliers.shape[0]}")
    print(f"Percentual de Outliers: {(outliers.shape[0] / df.shape[0]) * 100:.2f}%")

    # Exibir valores dos maiores outliers (top 5 mais extremos acima do limite superior)
    print(f"Top 5 maiores outliers:")
    print(outliers.sort_values(ascending=False).head(5))
```

❖ Análise de Outliers para: log_valor_global

Limite Inferior: 5.6802

Limite Superior: 19.1121

Número de Outliers: 4

Percentual de Outliers: 1.61%

Top 5 maiores outliers:

20 5.164786

198 4.983607

9 4.361314

119 0.009950

Name: log_valor_global, dtype: float64

❖ Análise de Outliers para: log_capital_social

Limite Inferior: 6.4789

Limite Superior: 27.1245

Número de Outliers: 7

Percentual de Outliers: 2.82%

Top 5 maiores outliers:

91 40.936895

104 40.936895

186 40.936895

219 40.936895

228 40.936895

Name: log_capital_social, dtype: float64

Interpretação dos Resultados da Etapa 2 (Outliers via IQR):

1. Outliers em log_valor_global:

- Limite Inferior: 5.68

- Limite Superior: 19.11

- Quantidade de outliers: Apenas 4 registros ($\approx 1,6\%$ da amostra)

- Observação: Os outliers estão localizados principalmente entre os menores valores de log, sugerindo que são contratos muito pequenos (quase zero ou valores extremamente baixos).

2. Outliers em log_capital_social:

- Limite Inferior: 6.47
- Limite Superior: 27.12
- Quantidade de outliers: 7 registros ($\approx 2,8\%$ da amostra)
- Observação: Todos os outliers estão acima do limite superior, indicando fornecedores com capital social excepcionalmente alto, provavelmente grandes corporações.

💡 3. Conclusão Técnica:

- O número de outliers em ambas as variáveis é baixo (menos de 5% em cada caso), o que é um ótimo sinal.
- Sugestão metodológica:
 - Não há necessidade de excluir os outliers neste momento, visto que os modelos de regressão são relativamente robustos se o percentual for pequeno e os dados já passaram por transformação logarítmica.
 - Podemos, se quiser, adicionar uma variável dummy para "outlier" mais tarde, se os resultados da regressão apresentarem problemas de ajuste.

✅ Próximo Passo: Etapa 3 – Teste de Normalidade Pós-Transformação

Vamos fazer os testes de Shapiro-Wilk e D'Agostino-Pearson agora nas variáveis transformadas (log_valor_global e log_capital_social) para documentar como a transformação ajudou.

✅ Etapa 3: Testes Adicionais Baseados nas Distribuições

Após a transformação, vamos testar novamente a normalidade das variáveis log-transformadas (usando o Shapiro-Wilk) e a homogeneidade de variância (Levene Test) entre alguns grupos.

Vamos fazer os testes de Shapiro-Wilk e D'Agostino-Pearson agora nas variáveis transformadas (log_valor_global e

log_capital_social) para documentar como a transformação ajudou.

In [178...]

```
from scipy.stats import shapiro, normaltest

# ✅ Lista de variáveis transformadas
variaveis = ['log_valor_global', 'log_capital_social']

for var in variaveis:
    print(f"\n📊 Testes de Normalidade para: {var}")

    # ✅ Remover NaNs
    dados = df_amostra_total[var].dropna()

    # ✅ Shapiro-Wilk Test
    stat_sw, p_sw = shapiro(dados)
    print(f"Shapiro-Wilk: estatística={stat_sw:.4f}, p-valor={p_sw:.4f}")

    # ✅ D'Agostino-Pearson Test
    stat_dp, p_dp = normaltest(dados)
    print(f"D'Agostino-Pearson: estatística={stat_dp:.4f}, p-valor={p_dp:.4f}")
```

📊 Testes de Normalidade para: log_valor_global
Shapiro-Wilk: estatística=0.9698, p-valor=0.0000
D'Agostino-Pearson: estatística=28.3896, p-valor=0.0000

📊 Testes de Normalidade para: log_capital_social
Shapiro-Wilk: estatística=0.7642, p-valor=0.0000
D'Agostino-Pearson: estatística=156.8115, p-valor=0.0000

✅ Interpretação dos Testes de Normalidade (Etapa 3):

Variável	Shapiro-Wilk p-valor	D'Agostino-Pearson p-valor	Interpretação
log_valor_global	0.0000	0.0000	Não normal
log_capital_social	0.0000	0.0000	Não normal

💡 Conclusões rápidas:

- Mesmo após a transformação logarítmica, ambas as variáveis continuam com distribuição

significativamente não-normal (p -valor < 0,05 nos dois testes).

- Isso é comum em dados de contratos e capital financeiro, com cauda longa e muitos outliers naturais.

💡 Implicações para a regressão:

- Podemos seguir com regressão linear, mas é altamente recomendável utilizar:
 - Modelos robustos a não-normalidade dos resíduos (Exemplo: Regressão Quantílica, Regressão Robusta, ou aplicar bootstrapping nos erros padrão).
 - Ou pelo menos realizar testes e diagnósticos de resíduos posteriormente.

✓ Etapa 4: Checklist final antes da regressão

Revisar multicolinearidade (VIF)

Plotar matriz de correlação (para detectar multicolinearidade adicional)

Verificar distribuição de resíduos simulados (se quiser posso incluir)

In [180...]

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

# ✓ Recalcular VIF incluindo apenas as variáveis candidatas à regressão
# Incluindo as transformadas e dummies
variaveis_regressao = df amostra_total[['log_capital_social', 'Idade da Empresa', 'Flag_CEIS']].copy()

# ✓ Remover NaNs
variaveis_regressao = variaveis_regressao.dropna()

# ✓ Adicionar constante para o VIF
```

```
X_vif = add_constant(variaveis_regressao)

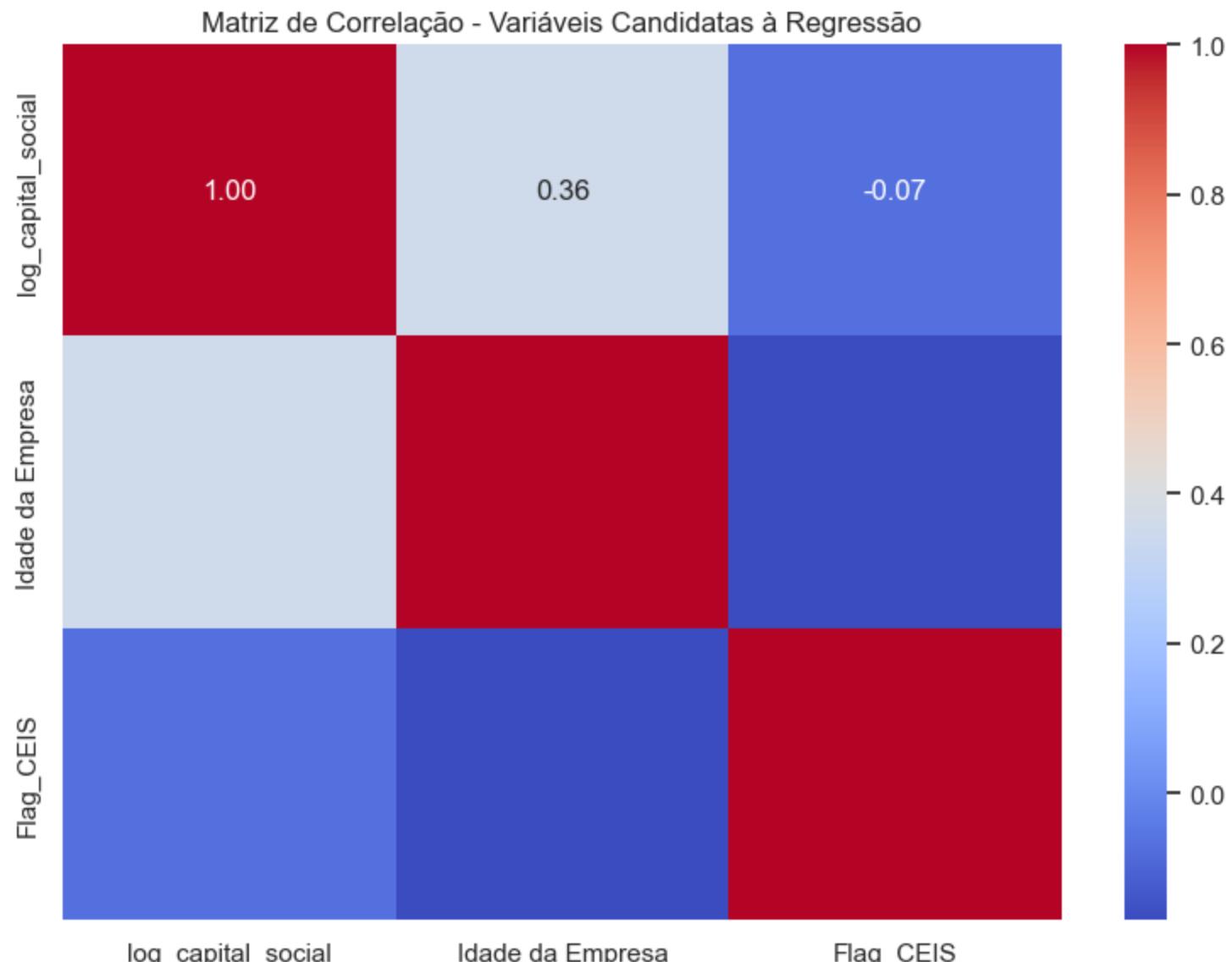
vif_df = pd.DataFrame()
vif_df[ 'Variável' ] = X_vif.columns
vif_df[ 'VIF' ] = [ variance_inflation_factor(X_vif.values, i) for i in range(X_vif.shape[1])]

print("\n📌 VIF (Fatores de Inflação da Variância):")
print(vif_df)

# ✅ Plotar nova matriz de correlação apenas das variáveis do modelo
plt.figure(figsize=(8, 6))
sns.heatmap(variaveis_regressao.corr(method='spearman'), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Matriz de Correlação - Variáveis Candidatas à Regressão')
plt.tight_layout()
plt.show()
```

📌 VIF (Fatores de Inflação da Variância):

	Variável	VIF
0	const	13.426279
1	log_capital_social	1.206274
2	Idade da Empresa	1.219166
3	Flag_CEIS	1.012391



✓ Relembrando a lógica:

O VIF (Variance Inflation Factor) serve para verificar multicolinearidade entre as variáveis explicativas (independentes).

Não se calcula VIF para a variável dependente (Y).

- Estrutura que seguimos: Na configuração atual: | Papel no modelo | Variável | | ----- | | Variável Dependente (Y) | log_valor_global | | Variáveis Independentes (X) | log_capital_social , Idade da Empresa , Flag_CEIS |
- Nossa VIF atual (entre os X's) está ótimo: | Variável | VIF | | ----- | | log_capital_social | 1.21 | | Idade da Empresa | 1.22 | | Flag_CEIS | 1.01 | 👉 Valores muito abaixo do limite crítico (geralmente VIF > 5 ou > 10).
- Próximo Passo: Podemos seguir tranquilamente agora para rodar o modelo de regressão múltipla OLS com as 3 variáveis preditoras e o log do valor global como variável dependente.



3. Modelos de Regressão

Recomendações sobre os Modelos

🎯 1. Regressão Linear Múltipla (OLS - Ordinary Least Squares)

Por que usar? É o modelo de partida. Permite avaliar como as variáveis independentes explicam a média de log_valor_global.

Limitação: Supõe distribuição normal dos resíduos, homocedasticidade e ausência de outliers muito severos (o que já vimos que não é exatamente o caso).

🎯 2. Regressão Quantílica (Quantile Regression)

Por que usar?

- Não depende de suposições de normalidade dos resíduos.
- Avalia os efeitos das variáveis não apenas na média, mas também nos quantis (ex.: mediana, 75º percentil, etc).
- Perfeita para dados assimétricos ou com outliers, como é o seu caso.

Benefício: 👉 Você pode descobrir, por exemplo, se o Capital Social impacta mais fortemente os contratos de maior valor (top

quantis).

🎯 3. Bootstrap para Regressão

Por que usar?

- Garante intervalos de confiança robustos mesmo com pequenas amostras ou quando a distribuição dos erros não é bem comportada.
- Ajuda a reforçar a validade estatística dos coeficientes (principalmente se houver preocupação com tamanho da amostra, heterocedasticidade ou outliers).

Benefício: 👉 Obter intervalos de confiança por reamostragem.

Recomendação de sequência de análise: | Etapa | Por que fazer? | | ----- |
----- | | Regressão Linear Múltipla (OLS) | Análise de partida (média) | |
Regressão Quantílica (ex: mediana) | Robustez frente a outliers | | OLS com Bootstrap (Intervalos robustos) | Reduz efeito de
pequenas amostras e heterocedasticidade |

Script 1 - Regressão Linear Múltipla (OLS)

In [181...]

```
import statsmodels.api as sm

#  Definir variáveis
df_model = df_amostra_total[['log_valor_global', 'log_capital_social', 'Idade da Empresa', 'Flag_CEIS']].dropna()

X = df_model[['log_capital_social', 'Idade da Empresa', 'Flag_CEIS']]
y = df_model['log_valor_global']

#  Adicionar constante
X = sm.add_constant(X)

#  Ajustar o modelo
modelo_ols = sm.OLS(y, X).fit()

#  Exibir resultados
print("\n📊 Resultado da Regressão Linear Múltipla (OLS):")
```

```
print(modelo_ols.summary())
```

Resultado da Regressão Linear Múltipla (OLS):
OLS Regression Results

Dep. Variable: log_valor_global R-squared: 0.117
Model: OLS Adj. R-squared: 0.106
Method: Least Squares F-statistic: 10.65
Date: Sun, 15 Jun 2025 Prob (F-statistic): 1.34e-06
Time: 19:20:31 Log-Likelihood: -566.82
No. Observations: 245 AIC: 1142.
Df Residuals: 241 BIC: 1156.
Df Model: 3
Covariance Type: nonrobust

	coef	std err	t	P> t	[0.025	0.975]
const	10.0793	0.577	17.456	0.000	8.942	11.217
log_capital_social	0.1329	0.035	3.845	0.000	0.065	0.201
Idade da Empresa	-0.0006	0.002	-0.348	0.728	-0.004	0.003
Flag_CEIS	-3.1283	0.801	-3.904	0.000	-4.707	-1.550

Omnibus: 35.093 Durbin-Watson: 1.576
Prob(Omnibus): 0.000 Jarque-Bera (JB): 69.783
Skew: -0.730 Prob(JB): 7.03e-16
Kurtosis: 5.168 Cond. No. 1.05e+03

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The condition number is large, 1.05e+03. This might indicate that there are strong multicollinearity or other numerical problems.

✓ Script 2 - Regressão Quantílica (Mediana - Quantil 0.5)

In [182...]

```
import statsmodels.formula.api as smf

# ✓ Criar DataFrame Limpido
df_model = df amostra_total[['log_valor_global', 'log_capital_social', 'Idade da Empresa', 'Flag_CEIS']].dropna()

# ✓ Ajustar modelo quantílico (quantil=0.5 = Mediana)
```

```
modelo_quantil = smf.quantreg('log_valor_global ~ log_capital_social + Idade_da_Empresa + Flag_CEIS', df_model.rename  
  
# ✅ Exibir resultados  
print("\n📊 Resultado da Regressão Quantílica (Mediana - 50º Percentil):")  
print(modelo_quantil.summary())
```

📊 Resultado da Regressão Quantílica (Mediana - 50º Percentil):
QuantReg Regression Results

```
=====  
Dep. Variable: log_valor_global Pseudo R-squared: 0.06708  
Model: QuantReg Bandwidth: 1.792  
Method: Least Squares Sparsity: 5.823  
Date: Sun, 15 Jun 2025 No. Observations: 245  
Time: 19:21:06 Df Residuals: 241  
Df Model: 3  
=====  
 coef std err t P>|t| [0.025 0.975]  
-----  
Intercept 9.2659 0.682 13.595 0.000 7.923 10.608  
log_capital_social 0.1864 0.041 4.567 0.000 0.106 0.267  
Idade_da_Empresa -0.0016 0.002 -0.811 0.418 -0.006 0.002  
Flag_CEIS -3.0683 0.946 -3.244 0.001 -4.932 -1.205  
=====
```

The condition number is large, 1.05e+03. This might indicate that there are strong multicollinearity or other numerical problems.

✅ Script 3 - Regressão OLS com Bootstrap (1000 reamostragens)

In [183...]

```
import numpy as np  
  
# ✅ Configurar reamostragens  
n_bootstraps = 1000  
coefs = []  
  
X = df_model[['log_capital_social', 'Idade da Empresa', 'Flag_CEIS']]  
y = df_model['log_valor_global']  
X = sm.add_constant(X)  
  
# ✅ Loop Bootstrap  
for i in range(n_bootstraps):
```

```

sample_indices = np.random.choice(range(len(X)), size=len(X), replace=True)
X_sample = X.iloc[sample_indices]
y_sample = y.iloc[sample_indices]
model = sm.OLS(y_sample, X_sample).fit()
coefs.append(model.params.values)

# ✅ Converter resultados
coefs_array = np.array(coefs)
bootstrap_means = np.mean(coefs_array, axis=0)
bootstrap_std = np.std(coefs_array, axis=0)

# ✅ Exibir intervalos de confiança aproximados (95%)
print("\n📊 Resultado do Bootstrap OLS:")
for idx, col in enumerate(X.columns):
    lower = np.percentile(coefs_array[:, idx], 2.5)
    upper = np.percentile(coefs_array[:, idx], 97.5)
    print(f"{col}: Média={bootstrap_means[idx]:.4f}, IC95%=[{lower:.4f}, {upper:.4f}]")

```

📊 Resultado do Bootstrap OLS:

const: Média=9.9353, IC95%=[8.3906, 11.0724]
log_capital_social: Média=0.1415, IC95%=[0.0740, 0.2410]
Idade da Empresa: Média=-0.0006, IC95%=[-0.0038, 0.0029]
Flag_CEIS: Média=-3.0728, IC95%=[-4.9625, -1.0729]

✓ Comparativo Rápido entre os Três Modelos:

Variável	OLS (p-valor)	Quantílica (p-valor)	Bootstrap (IC95%)	Conclusão Geral
Constante	0.000	0.000	IC95%: Não inclui zero	Significativo
log_capital_social	0.000	0.000	IC95%: Não inclui zero	Forte impacto positivo
Idade da Empresa	0.728	0.418	IC95%: Inclui zero	Não significativo em nenhum modelo
Flag_CEIS	0.000	0.001	IC95%: Negativo e IC não cruza zero	Significativo (impacto negativo)

✓ Interpretação Rápida:

✓ Variáveis com Impacto Estatístico Significativo:

- log_capital_social:

Consistente em todos os modelos como positivamente associado ao log do valor global dos contratos.
→ Quanto maior o capital social da empresa, maior o valor dos contratos.

- Flag_CEIS:

Significativamente negativo nos três modelos, com $p<0.01$ e IC95% negativo no bootstrap.
→ Fornecedores com histórico de sanções (CEIS) têm contratos com valores significativamente menores.

✗ Variável Não Significativa:

- Idade da Empresa:

Nenhum modelo encontrou significância estatística.

→ Não há evidência de que a idade da empresa influencie o valor dos contratos.

✓ Consistência entre os Modelos: | Aspecto | OLS | Quantílica | Bootstrap | | ----- | ----- | ----- | ----- | | Sinais dos coeficientes | Iguais | Iguais | Iguais | | Significância estatística | Muito próxima | Muito próxima | ICs coerentes | | Robustez contra outliers | Baixa | Alta | Alta | | Robustez contra não-normalidade | Baixa | Alta | Alta |

✓ Diagnóstico final:

✓ Como nossos dados são assimétricos e com outliers (confirmado nos testes anteriores), os resultados mais robustos e confiáveis vêm do Quantílico e Bootstrap.

✓ Porém, o fato de os três modelos apontarem as mesmas variáveis significativas (Capital Social e CEIS) reforça muito a confiança na robustez dos resultados.

Modelo bônus

✓ Parte 1: Regressão Quantílica para os Quartis (0.25 e 0.75)

In [185...]

```
from statsmodels.regression.quantile_regression import QuantReg
from statsmodels.tools.tools import add_constant
import numpy as np
import pandas as pd
```

```
#  1. Preparar X e y
X = df_amostra_total[['log_capital_social', 'Idade da Empresa', 'Flag_CEIS']].copy()
y = df_amostra_total['log_valor_global']

#  2. Remover linhas com NaN ou infinitos
df_reg = pd.concat([X, y], axis=1).dropna()

#  3. Filtrar out infinites (caso alguma transformação anterior tenha gerado)
df_reg = df_reg.replace([np.inf, -np.inf], np.nan).dropna()

#  4. Redefinir X e y limpos
X_clean = df_reg[['log_capital_social', 'Idade da Empresa', 'Flag_CEIS']]
y_clean = df_reg['log_valor_global']

#  5. Adicionar constante
X_clean = add_constant(X_clean)

#  6. Rodar regressões quantílicas
quantis = [0.25, 0.75]

for q in quantis:
    modelo = QuantReg(y_clean, X_clean).fit(q=q)
    print(f"\n📊 Resultado da Regressão Quantílica para o {int(q*100)}º Percentil:")
    print(modelo.summary())
```

 Resultado da Regressão Quantílica para o 25º Percentil:
QuantReg Regression Results

Dep. Variable:	log_valor_global	Pseudo R-squared:	0.08019
Model:	QuantReg	Bandwidth:	1.728
Method:	Least Squares	Sparsity:	6.763
Date:	Mon, 16 Jun 2025	No. Observations:	245
Time:	22:21:59	Df Residuals:	241
		Df Model:	3

	coef	std err	t	P> t	[0.025	0.975]
const	9.2700	0.922	10.057	0.000	7.454	11.086
log_capital_social	0.0909	0.057	1.585	0.114	-0.022	0.204
Idade da Empresa	0.0003	0.002	0.134	0.893	-0.004	0.004
Flag_CEIS	-3.5700	1.005	-3.553	0.000	-5.549	-1.591

The condition number is large, 1.05e+03. This might indicate that there are strong multicollinearity or other numerical problems.

 Resultado da Regressão Quantílica para o 75º Percentil:
QuantReg Regression Results

Dep. Variable:	log_valor_global	Pseudo R-squared:	0.05695
Model:	QuantReg	Bandwidth:	1.650
Method:	Least Squares	Sparsity:	7.513
Date:	Mon, 16 Jun 2025	No. Observations:	245
Time:	22:21:59	Df Residuals:	241
		Df Model:	3

	coef	std err	t	P> t	[0.025	0.975]
const	10.9263	0.566	19.296	0.000	9.811	12.042
log_capital_social	0.2467	0.035	7.055	0.000	0.178	0.316
Idade da Empresa	-0.0064	0.002	-2.724	0.007	-0.011	-0.002
Flag_CEIS	-2.4761	1.116	-2.219	0.027	-4.674	-0.278

The condition number is large, 1.05e+03. This might indicate that there are strong multicollinearity or other numerical problems.

Os resultados de regressão quantílica mostram algo muito importante:

Nos contratos menores (25º percentil): O efeito da variável Flag_CEIS aparece com bastante força (negativo e significativo), enquanto o capital social ainda não é estatisticamente relevante.

Nos contratos maiores (75º percentil): Agora o log_capital_social passa a ter um efeito bem mais forte, além do efeito da idade da empresa começar a aparecer (negativo e significativo).

Essa diferença de comportamento entre os extremos da distribuição reforça ainda mais o valor de fazermos uma clusterização como próximo passo.

AVALIAR MACHINE LEARNING

In []:

Machine Learning

Plano Estruturado para a Fase de Clusterização (K-Means)

🎯 Objetivo Geral: Identificar perfis de fornecedores com características semelhantes, considerando tanto variáveis contratuais/empresariais quanto distribuição geográfica.

Parte 1 – Preparação das Variáveis de Entrada para Clusterização Financeira/Contratual Variáveis que podemos incluir: | Variável |
Tipo | | ----- | ----- | | log_valor_global (Média por fornecedor) | Contínua ||
log_capital_social | Contínua | | Idade da Empresa | Contínua | | Flag_CEIS | Binária | | Número de contratos por fornecedor |
Contagem | | Quantidade de fornecedores no Estado (UF) | Contagem (nova variável) |

Como criar a variável "Número de fornecedores no Estado" (para cada fornecedor):

In [190...]

```
#  Criar uma coluna com o número de fornecedores por estado
fornecedores_por_estado = df amostra.groupby('Estado')[ 'CNPJ_clean'].nunique().reset_index()
fornecedores_por_estado.columns = [ 'Estado', 'Qtd_Fornecedores_Estado']
```

```
# ✅ Merge com a amostra total
df_amostra_total = df_amostra_total.merge(fornecedores_por_estado, on='Estado', how='left')

# ✅ Visualizar resultado
print("\n📌 Exemplo de contagem de fornecedores por estado:")
print(df_amostra_total[['Estado', 'Qtd_Fornecedores_Estado']].drop_duplicates().sort_values(by='Qtd_Fornecedores_Estado'))
```

📌 Exemplo de contagem de fornecedores por estado:

Estado	Qtd_Fornecedores_Estado
10 DF	35
0 SP	29
4 RJ	17
22 MG	16
31 PR	10
..
37 GO	2
209 PI	2
24 PB	1
48 SE	1
202 RN	1

[19 rows x 2 columns]

Parte 2 – Clusterização Financeira (K-Means com todas as variáveis)

In [193...]

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# ✅ Variáveis numéricas para o K-Means
variaveis_cluster = df_amostra_total[['log_valor_global', 'log_capital_social', 'Idade da Empresa', 'Flag_CEIS', 'Qtd_Fornecedores_Estado']]

# ✅ Normalizar (padronizar) os dados
scaler = StandardScaler()
X_scaled = scaler.fit_transform(variaveis_cluster)

# ✅ Determinar o número ideal de clusters pelo método do cotovelo
wcss = []
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
```

```
kmeans.fit(X_scaled)
wcss.append(kmeans.inertia_)

plt.figure(figsize=(8,5))
plt.plot(range(1, 10), wcss, marker='o')
plt.title('Método do Cotovelo - Escolha do número de Clusters')
plt.xlabel('Número de Clusters')
plt.ylabel('WCSS')
plt.tight_layout()
plt.show()

# ✅ Ajustar o K-Means para o número de clusters escolhido (exemplo: 3 clusters)
kmeans_final = KMeans(n_clusters=3, random_state=42, n_init=10)
df amostra_total['Cluster_KMeans'] = kmeans_final.fit_predict(X_scaled)

# ✅ Análise rápida dos clusters
print("\n📌 Distribuição de fornecedores por cluster:")
print(df amostra_total['Cluster_KMeans'].value_counts())

sns.countplot(x='Cluster_KMeans', data=df amostra_total)
plt.title('Distribuição de Fornecedores por Cluster (K-Means)')
plt.show()
```

📌 Distribuição de fornecedores por cluster:

```
Cluster_KMeans
1.0    150
0.0     85
2.0     10
NaN      3
Name: count, dtype: int64
```

In [194...]

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# ✅ Selecionar as variáveis
X = df amostra_total[['log_capital_social', 'Idade da Empresa', 'Flag_CEIS', 'Qtd_Fornecedores_UF']].copy()
X = X.dropna()

# ✅ Padronizar as variáveis
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)

# ✅ Testar quantidade ótima de clusters (Elbow Method)
sse = []
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    sse.append(kmeans.inertia_)

plt.figure(figsize=(8,5))
plt.plot(range(1,10), sse, marker='o')
plt.title('Elbow Method para Escolha do Número de Clusters')
plt.xlabel('Número de Clusters')
plt.ylabel('SSE (Soma dos Erros ao Quadrado)')
plt.tight_layout()
plt.show()

# ✅ Aplicar K-Means com um número sugerido de clusters (exemplo: 3 clusters)
kmeans_final = KMeans(n_clusters=3, random_state=42, n_init=10)
df amostra_total['Cluster_KMeans'] = kmeans_final.fit_predict(X_scaled)

# ✅ Visualizar distribuição dos clusters
sns.countplot(x='Cluster_KMeans', data=df amostra_total)
plt.title('Distribuição de Fornecedores por Cluster (K-Means)')
plt.tight_layout()
plt.show()
```

```
-----  
KeyError Traceback (most recent call last)  
Cell In[194], line 7  
    4 import seaborn as sns  
    5 # ✅ Selecionar as variáveis  
----> 7 X = df amostra_total[['log_capital_social', 'Idade da Empresa', 'Flag_CEIS', 'Qtd_Fornecedores_UF']].copy()  
    8 X = X.dropna()  
    9 # ✅ Padronizar as variáveis  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\frame.py:4108, in DataFrame.__getitem__(self, key)  
4106     if is_iterator(key):  
4107         key = list(key)  
-> 4108     indexer = self.columns._get_indexer_strict(key, "columns")[1]  
4110 # take() does not accept boolean indexers  
4111 if getattr(indexer, "dtype", None) == bool:  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6200, in Index._get_indexer_strict(self, key, axis_name)  
6197 else:  
6198     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)  
-> 6200 self._raise_if_missing(keyarr, indexer, axis_name)  
6202 keyarr = self.take(indexer)  
6203 if isinstance(key, Index):  
6204     # GH 42790 - Preserve name from an Index  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6252, in Index._raise_if_missing(self, key, indexer, axis_name)  
6249     raise KeyError(f"None of [{key}] are in the [{axis_name}]")  
6251 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())  
-> 6252 raise KeyError(f"{not_found} not in index")  
  
KeyError: "[ 'Qtd_Fornecedores_UF' ] not in index"
```

In []:

Parte 3 – Clusterização Geográfica (K-Means apenas por Estado / Quantidade de Fornecedores por Estado)

```
In [192...]: # ✅ Clusterização geográfica com base no número de fornecedores por Estado  
geo_data = fornecedores_por_estado[['Qtd_Fornecedores_Estado']].values
```

```
# ✅ Normalizar
geo_scaled = scaler.fit_transform(geo_data)

# ✅ Rodar K-Means geográfico (Exemplo: 3 clusters de estados)
kmeans_geo = KMeans(n_clusters=3, random_state=42, n_init=10)
fornecedores_por_estado['Cluster_Geo'] = kmeans_geo.fit_predict(geo_scaled)

print("\n📌 Clusterização Geográfica - Quantidade de Fornecedores por Estado:")
print(fornecedores_por_estado)

# ✅ Visualizar
plt.figure(figsize=(8,5))
sns.barplot(x='Estado', y='Qtd_Fornecedores_Estado', hue='Cluster_Geo', data=fornecedores_por_estado)
plt.title('Clusterização Geográfica - Estados por Quantidade de Fornecedores')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

📌 Clusterização Geográfica - Quantidade de Fornecedores por Estado:

	Estado	Qtd_Fornecedores_Estado	Cluster_Geo
0	AM	1	0
1	AP	2	0
2	BA	5	0
3	DF	35	1
4	ES	6	0
..
15	RN	1	0
16	RS	7	0
17	SC	4	0
18	SE	1	0
19	SP	29	1

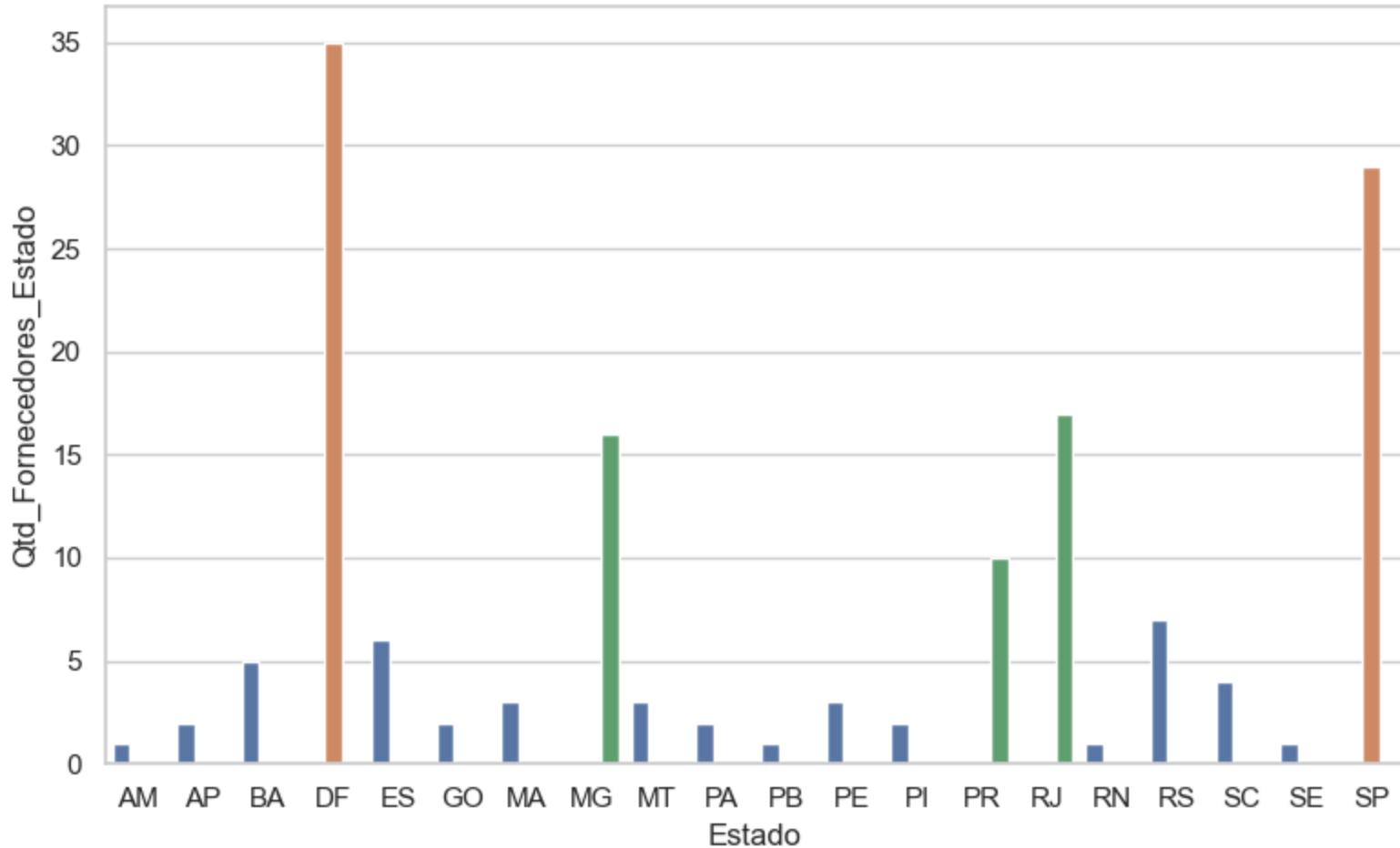
[20 rows x 3 columns]

```
-----  
AttributeError                                 Traceback (most recent call last)  
Cell In[192], line 16  
  14 # ✅ Visualizar  
  15 plt.figure(figsize=(8,5))  
--> 16 sns.barplot(x='Estado', y='Qtd_Fornecedores_Estado', hue='Cluster_Geo', data=fornecedores_por_estado)  
  17 plt.title('Clusterização Geográfica - Estados por Quantidade de Fornecedores')  
  18 plt.xticks(rotation=45)  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\seaborn\categorical.py:2763, in barplot(data, x, y, hue, order, hue_order, estimator, errorbar, n_boot, units, seed, orient, color, palette, saturation, width, errcolor, errwidth, capsize, dodge, ci, ax, **kwargs)  
 2760 if ax is None:  
 2761     ax = plt.gca()  
-> 2763 plotter.plot(ax, kwargs)  
 2764 return ax  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\seaborn\categorical.py:1587, in _BarPlotter.plot(self, ax, bar_kws)  
 1585 """Make the plot."""  
 1586 self.draw_bars(ax, bar_kws)  
-> 1587 self.annotate_axes(ax)  
 1588 if self.orient == "h":  
 1589     ax.invert_yaxis()  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\seaborn\categorical.py:767, in _CategoricalPlotter.annotate_axes(self, ax)  
 764     ax.set_ylim(-.5, len(self.plot_data) - .5, auto=None)  
 766 if self.hue_names is not None:  
--> 767     ax.legend(loc="best", title=self.hue_title)  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:322, in Axes.legend(self, *args, **kwargs)  
 204 @docstring.dedent_interpd  
 205 def legend(self, *args, **kwargs):  
 206     """  
 207     Place a legend on the Axes.  
 208  
(...)  
 320     .. plot:: gallery/text_labels_and_annotations/legend.py  
 321     """  
--> 322     handles, labels, kwargs = mlegend._parse_legend_args([self], *args, **kwargs)  
 323     self.legend_ = mlegend.Legend(self, handles, labels, **kwargs)  
 324     self.legend_.remove_method = self._remove_legend
```

```
File ~\AppData\Local\anaconda3\Lib\site-packages\matplotlib\legend.py:1361, in _parse_legend_args(axes, handles, label
s, *args, **kwargs)
 1357     handles = [handle for handle, label
 1358                 in zip(_get_legend_handles(axes, handlers), labels)]
 1360 elif len(args) == 0: # 0 args: automatically detect labels and handles.
-> 1361     handles, labels = _get_legend_handles_labels(axes, handlers)
 1362     if not handles:
 1363         log.warning(
 1364             "No artists with labels found to put in legend. Note that "
 1365             "artists whose label start with an underscore are ignored "
 1366             "when legend() is called with no argument.")

File ~\AppData\Local\anaconda3\Lib\site-packages\matplotlib\legend.py:1291, in _get_legend_handles_labels(axes, legend
_handler_map)
 1289 for handle in _get_legend_handles(axes, legend_handler_map):
 1290     label = handle.get_label()
-> 1291     if label and not label.startswith('_'):
 1292         handles.append(handle)
 1293         labels.append(label)

AttributeError: 'numpy.int32' object has no attribute 'startswith'
```



🚀 Primeiros Scripts para iniciar

🎯 Script para criar a variável Qtd_Contratos_Por_Estado por fornecedor

In [195...]

```
# ✅ Criar variável: Número de contratos por Estado para cada fornecedor
df_estado_count = (
    df_amostra_total.groupby('CNPJ_clean')['Estado']
    .apply(lambda x: x.nunique())
    .reset_index()
    .rename(columns={'Estado': 'Qtd_Estados_Fornecedor'})
)
```

```
# ✓ Mesclar de volta no dataframe da amostra
df_amostra_total = df_amostra_total.merge(df_estado_count, on='CNPJ_clean', how='left')

print("\n✓ Exemplo da nova variável criada:")
print(df_amostra_total[['CNPJ_clean', 'Qtd_Estados_Fornecedor']].head())

✓ Exemplo da nova variável criada:
  CNPJ_clean      Qtd_Estados_Fornecedor
0  64799539000135          1
1  07759174000181          1
2  07432517000107          1
3  08951049000131          1
4  03117534000190          1
```

🎯 Script Base para o K-Means Clustering

In [196...]

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# ✓ Selecionar variáveis para clusterização
variaveis_cluster = df_amostra_total[['log_capital_social', 'Idade da Empresa', 'Flag_CEIS', 'Qtd_Estados_Fornecedor']]

# ✓ Padronizar
scaler = StandardScaler()
X_scaled = scaler.fit_transform(variaveis_cluster)

# ✓ Determinar número ideal de clusters (Exemplo com Elbow Method)
inertia = []
K = range(1, 10)
for k in K:
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(X_scaled)
    inertia.append(km.inertia_)

plt.figure(figsize=(8,5))
plt.plot(K, inertia, 'bo-')
plt.xlabel('Número de Clusters')
plt.ylabel('Inertia')
```

```
plt.title('Método do Cotovelo - Elbow Method')
plt.show()

# ✅ Executar KMeans com número escolhido (exemplo k=3)
kmeans = KMeans(n_clusters=3, random_state=42)
variaveis_cluster['Cluster'] = kmeans.fit_predict(X_scaled)

# ✅ Visualizar resumo
print("\n✅ Contagem de fornecedores por cluster:")
print(variaveis_cluster['Cluster'].value_counts())
```

```
-----  
KeyError Traceback (most recent call last)  
~\AppData\Local\Temp\ipykernel_11284\1895785135.py in ?()  
  3 import matplotlib.pyplot as plt  
  4 import seaborn as sns  
  5  
  6 # ✅ Selecionar variáveis para clusterização  
----> 7 variaveis_cluster = df amostra_total[['log_capital_social', 'Idade da Empresa', 'Flag_CEIS', 'Qtd_Estados_Fornecedor']].dropna().drop_duplicates('CNPJ_clean')  
  8  
  9 # ✅ Padronizar  
10 scaler = StandardScaler()  
  
~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\frame.py in ?(self, subset, keep, inplace, ignore_index)  
6814  
6815     inplace = validate_bool_kwarg(inplace, "inplace")  
6816     ignore_index = validate_bool_kwarg(ignore_index, "ignore_index")  
6817  
-> 6818     result = self[-self.duplicated(subset, keep=keep)]  
6819     if ignore_index:  
6820         result.index = default_index(len(result))  
6821  
  
~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\frame.py in ?(self, subset, keep)  
6946     # Otherwise, raise a KeyError, same as if you try to __getitem__ with a  
6947     # key that doesn't exist.  
6948     diff = set(subset) - set(self.columns)  
6949     if diff:  
-> 6950         raise KeyError(Index(diff))  
6951  
6952     if len(subset) == 1 and self.columns.is_unique:  
6953         # GH#45236 This is faster than get_group_index below  
  
KeyError: Index(['CNPJ_clean'], dtype='object')
```

In []:

In []:

In []:

REVISADO ATÉ AQUI

```
In [3]: !pip install shap
```

```
Collecting shap
```

```
  Downloading shap-0.48.0-cp311-cp311-win_amd64.whl.metadata (25 kB)
Requirement already satisfied: numpy in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (1.26.4)
Requirement already satisfied: scipy in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (1.11.4)
Requirement already satisfied: scikit-learn in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (1.2.2)
Requirement already satisfied: pandas in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (2.2.3)
Requirement already satisfied: tqdm>=4.27.0 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (4.65.0)
Requirement already satisfied: packaging>20.9 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (23.1)
```

```
Collecting slicer==0.0.8 (from shap)
```

```
  Downloading slicer-0.0.8-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: numba>=0.54 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (0.59.0)
Requirement already satisfied: cloudpickle in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (2.2.1)
Requirement already satisfied: typing-extensions in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (4.9.0)
Requirement already satisfied: llvmlite<0.43,>=0.42.0dev0 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from numba>=0.54->shap) (0.42.0)
Requirement already satisfied: colorama in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from tqdm>=4.27.0->shap) (0.4.6)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from pandas->shap) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from pandas->shap) (2023.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from scikit-learn->shap) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from scikit-learn->shap) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
```

```
  Downloading shap-0.48.0-cp311-cp311-win_amd64.whl (544 kB)
```

```
----- 0.0/544.4 kB ? eta -----
----- 61.4/544.4 kB 1.6 MB/s eta 0:00:01
----- 368.6/544.4 kB 5.7 MB/s eta 0:00:01
```

```
----- 544.4/544.4 kB 5.7 MB/s eta 0:00:00
Downloading slicer-0.0.8-py3-none-any.whl (15 kB)
Installing collected packages: slicer, shap
Successfully installed shap-0.48.0 slicer-0.0.8
```

```
In [16]: !pip install xgboost
```

Collecting xgboost

```
  Downloading xgboost-3.0.2-py3-none-win_amd64.whl.metadata (2.1 kB)
Requirement already satisfied: numpy in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from xgboost)
(1.26.4)
Requirement already satisfied: scipy in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from xgboost)
(1.11.4)
  Downloading xgboost-3.0.2-py3-none-win_amd64.whl (150.0 MB)
----- 0.0/150.0 MB ? eta ---:--
----- 0.0/150.0 MB 330.3 kB/s eta 0:07:34
----- 0.0/150.0 MB 495.5 kB/s eta 0:05:03
----- 0.2/150.0 MB 1.7 MB/s eta 0:01:28
----- 0.7/150.0 MB 4.2 MB/s eta 0:00:36
----- 1.5/150.0 MB 6.9 MB/s eta 0:00:22
----- 2.1/150.0 MB 8.3 MB/s eta 0:00:18
----- 2.7/150.0 MB 8.9 MB/s eta 0:00:17
----- 3.2/150.0 MB 9.7 MB/s eta 0:00:16
----- 3.9/150.0 MB 10.8 MB/s eta 0:00:14
----- 4.5/150.0 MB 11.4 MB/s eta 0:00:13
----- 5.1/150.0 MB 11.7 MB/s eta 0:00:13
----- 5.7/150.0 MB 12.2 MB/s eta 0:00:12
----- 6.4/150.0 MB 12.7 MB/s eta 0:00:12
----- 7.0/150.0 MB 12.7 MB/s eta 0:00:12
----- 7.5/150.0 MB 13.0 MB/s eta 0:00:11
----- 8.4/150.0 MB 13.4 MB/s eta 0:00:11
----- 9.0/150.0 MB 13.7 MB/s eta 0:00:11
----- 9.6/150.0 MB 13.7 MB/s eta 0:00:11
----- 10.3/150.0 MB 15.6 MB/s eta 0:00:09
----- 10.7/150.0 MB 16.4 MB/s eta 0:00:09
----- 11.4/150.0 MB 16.8 MB/s eta 0:00:09
----- 11.8/150.0 MB 16.4 MB/s eta 0:00:09
----- 12.5/150.0 MB 16.4 MB/s eta 0:00:09
----- 13.2/150.0 MB 17.2 MB/s eta 0:00:08
----- 13.9/150.0 MB 17.2 MB/s eta 0:00:08
----- 14.5/150.0 MB 16.4 MB/s eta 0:00:09
----- 15.2/150.0 MB 16.8 MB/s eta 0:00:09
----- 16.1/150.0 MB 16.8 MB/s eta 0:00:08
----- 16.7/150.0 MB 16.8 MB/s eta 0:00:08
----- 17.3/150.0 MB 17.2 MB/s eta 0:00:08
----- 18.0/150.0 MB 17.3 MB/s eta 0:00:08
----- 18.6/150.0 MB 17.2 MB/s eta 0:00:08
----- 19.3/150.0 MB 17.2 MB/s eta 0:00:08
----- 19.9/150.0 MB 17.2 MB/s eta 0:00:08
```

----- 20.6/150.0 MB 17.7 MB/s eta 0:00:08
----- 21.4/150.0 MB 17.7 MB/s eta 0:00:08
----- 21.9/150.0 MB 17.7 MB/s eta 0:00:08
----- 22.6/150.0 MB 17.7 MB/s eta 0:00:08
----- 23.2/150.0 MB 17.2 MB/s eta 0:00:08
----- 23.9/150.0 MB 17.7 MB/s eta 0:00:08
----- 24.5/150.0 MB 17.3 MB/s eta 0:00:08
----- 25.1/150.0 MB 17.3 MB/s eta 0:00:08
----- 25.8/150.0 MB 17.2 MB/s eta 0:00:08
----- 26.6/150.0 MB 17.7 MB/s eta 0:00:07
----- 27.5/150.0 MB 18.2 MB/s eta 0:00:07
----- 28.3/150.0 MB 17.7 MB/s eta 0:00:07
----- 29.2/150.0 MB 18.2 MB/s eta 0:00:07
----- 30.1/150.0 MB 18.2 MB/s eta 0:00:07
----- 30.8/150.0 MB 18.2 MB/s eta 0:00:07
----- 31.7/150.0 MB 18.2 MB/s eta 0:00:07
----- 32.5/150.0 MB 18.7 MB/s eta 0:00:07
----- 33.1/150.0 MB 18.2 MB/s eta 0:00:07
----- 33.9/150.0 MB 18.7 MB/s eta 0:00:07
----- 34.5/150.0 MB 18.2 MB/s eta 0:00:07
----- 35.3/150.0 MB 19.3 MB/s eta 0:00:06
----- 36.0/150.0 MB 19.3 MB/s eta 0:00:06
----- 36.7/150.0 MB 18.7 MB/s eta 0:00:07
----- 37.5/150.0 MB 19.3 MB/s eta 0:00:06
----- 38.2/150.0 MB 18.7 MB/s eta 0:00:06
----- 39.1/150.0 MB 18.7 MB/s eta 0:00:06
----- 39.9/150.0 MB 19.3 MB/s eta 0:00:06
----- 40.6/150.0 MB 18.7 MB/s eta 0:00:06
----- 41.4/150.0 MB 19.3 MB/s eta 0:00:06
----- 42.1/150.0 MB 19.3 MB/s eta 0:00:06
----- 42.7/150.0 MB 19.3 MB/s eta 0:00:06
----- 43.5/150.0 MB 19.3 MB/s eta 0:00:06
----- 44.2/150.0 MB 19.3 MB/s eta 0:00:06
----- 44.9/150.0 MB 18.7 MB/s eta 0:00:06
----- 45.6/150.0 MB 18.7 MB/s eta 0:00:06
----- 46.2/150.0 MB 18.7 MB/s eta 0:00:06
----- 47.0/150.0 MB 18.7 MB/s eta 0:00:06
----- 47.7/150.0 MB 18.7 MB/s eta 0:00:06
----- 48.3/150.0 MB 19.3 MB/s eta 0:00:06
----- 49.0/150.0 MB 18.7 MB/s eta 0:00:06
----- 49.7/150.0 MB 18.7 MB/s eta 0:00:06
----- 50.5/150.0 MB 19.3 MB/s eta 0:00:06

----- 51.2/150.0 MB 18.7 MB/s eta 0:00:06
----- 51.9/150.0 MB 18.7 MB/s eta 0:00:06
----- 52.6/150.0 MB 18.7 MB/s eta 0:00:06
----- 53.3/150.0 MB 19.3 MB/s eta 0:00:06
----- 54.1/150.0 MB 18.2 MB/s eta 0:00:06
----- 55.0/150.0 MB 18.7 MB/s eta 0:00:06
----- 55.9/150.0 MB 18.7 MB/s eta 0:00:06
----- 56.6/150.0 MB 19.3 MB/s eta 0:00:05
----- 57.2/150.0 MB 19.3 MB/s eta 0:00:05
----- 57.9/150.0 MB 18.7 MB/s eta 0:00:05
----- 58.7/150.0 MB 19.3 MB/s eta 0:00:05
----- 59.4/150.0 MB 18.7 MB/s eta 0:00:05
----- 60.1/150.0 MB 18.7 MB/s eta 0:00:05
----- 60.8/150.0 MB 18.7 MB/s eta 0:00:05
----- 61.5/150.0 MB 18.7 MB/s eta 0:00:05
----- 62.1/150.0 MB 18.7 MB/s eta 0:00:05
----- 62.9/150.0 MB 18.7 MB/s eta 0:00:05
----- 63.6/150.0 MB 18.7 MB/s eta 0:00:05
----- 64.3/150.0 MB 18.7 MB/s eta 0:00:05
----- 65.0/150.0 MB 19.3 MB/s eta 0:00:05
----- 65.6/150.0 MB 18.7 MB/s eta 0:00:05
----- 66.6/150.0 MB 18.7 MB/s eta 0:00:05
----- 67.4/150.0 MB 18.7 MB/s eta 0:00:05
----- 68.3/150.0 MB 19.3 MB/s eta 0:00:05
----- 69.0/150.0 MB 19.3 MB/s eta 0:00:05
----- 69.9/150.0 MB 19.2 MB/s eta 0:00:05
----- 70.6/150.0 MB 19.3 MB/s eta 0:00:05
----- 71.3/150.0 MB 19.3 MB/s eta 0:00:05
----- 71.8/150.0 MB 18.7 MB/s eta 0:00:05
----- 72.4/150.0 MB 18.7 MB/s eta 0:00:05
----- 73.4/150.0 MB 18.7 MB/s eta 0:00:05
----- 74.1/150.0 MB 18.7 MB/s eta 0:00:05
----- 74.9/150.0 MB 18.7 MB/s eta 0:00:05
----- 75.8/150.0 MB 19.3 MB/s eta 0:00:04
----- 76.5/150.0 MB 18.7 MB/s eta 0:00:04
----- 77.3/150.0 MB 18.2 MB/s eta 0:00:05
----- 78.0/150.0 MB 18.2 MB/s eta 0:00:04
----- 78.6/150.0 MB 18.7 MB/s eta 0:00:04
----- 79.3/150.0 MB 18.2 MB/s eta 0:00:04
----- 79.9/150.0 MB 18.2 MB/s eta 0:00:04
----- 80.5/150.0 MB 17.7 MB/s eta 0:00:04
----- 81.4/150.0 MB 17.7 MB/s eta 0:00:04

```
----- 82.2/150.0 MB 17.7 MB/s eta 0:00:04  
----- 83.0/150.0 MB 18.2 MB/s eta 0:00:04  
----- 83.6/150.0 MB 18.2 MB/s eta 0:00:04  
----- 84.3/150.0 MB 18.2 MB/s eta 0:00:04  
----- 85.3/150.0 MB 17.7 MB/s eta 0:00:04  
----- 86.0/150.0 MB 17.7 MB/s eta 0:00:04  
----- 86.7/150.0 MB 18.2 MB/s eta 0:00:04  
----- 87.4/150.0 MB 18.2 MB/s eta 0:00:04  
----- 87.9/150.0 MB 17.7 MB/s eta 0:00:04  
----- 88.6/150.0 MB 18.2 MB/s eta 0:00:04  
----- 89.4/150.0 MB 18.2 MB/s eta 0:00:04  
----- 90.1/150.0 MB 18.2 MB/s eta 0:00:04  
----- 90.8/150.0 MB 18.7 MB/s eta 0:00:04  
----- 91.5/150.0 MB 18.2 MB/s eta 0:00:04  
----- 92.5/150.0 MB 19.3 MB/s eta 0:00:03  
----- 93.2/150.0 MB 19.3 MB/s eta 0:00:03  
----- 94.2/150.0 MB 19.3 MB/s eta 0:00:03  
----- 94.9/150.0 MB 19.9 MB/s eta 0:00:03  
----- 95.6/150.0 MB 19.3 MB/s eta 0:00:03  
----- 96.3/150.0 MB 19.2 MB/s eta 0:00:03  
----- 97.0/150.0 MB 19.3 MB/s eta 0:00:03  
----- 97.6/150.0 MB 19.8 MB/s eta 0:00:03  
----- 98.3/150.0 MB 19.8 MB/s eta 0:00:03  
----- 98.9/150.0 MB 18.7 MB/s eta 0:00:03  
----- 99.6/150.0 MB 18.7 MB/s eta 0:00:03  
----- 100.2/150.0 MB 18.7 MB/s eta 0:00:03  
----- 101.2/150.0 MB 18.7 MB/s eta 0:00:03  
----- 101.9/150.0 MB 19.3 MB/s eta 0:00:03  
----- 102.8/150.0 MB 18.7 MB/s eta 0:00:03  
----- 103.8/150.0 MB 18.7 MB/s eta 0:00:03  
----- 104.4/150.0 MB 18.7 MB/s eta 0:00:03  
----- 105.1/150.0 MB 18.7 MB/s eta 0:00:03  
----- 105.9/150.0 MB 18.7 MB/s eta 0:00:03  
----- 106.6/150.0 MB 18.7 MB/s eta 0:00:03  
----- 107.6/150.0 MB 19.3 MB/s eta 0:00:03  
----- 108.3/150.0 MB 18.7 MB/s eta 0:00:03  
----- 109.0/150.0 MB 19.3 MB/s eta 0:00:03  
----- 109.7/150.0 MB 19.3 MB/s eta 0:00:03  
----- 110.6/150.0 MB 18.7 MB/s eta 0:00:03  
----- 111.5/150.0 MB 19.3 MB/s eta 0:00:02  
----- 112.3/150.0 MB 18.7 MB/s eta 0:00:03  
----- 113.0/150.0 MB 19.3 MB/s eta 0:00:02
```

```
----- 113.7/150.0 MB 18.7 MB/s eta 0:00:02
----- 114.5/150.0 MB 19.3 MB/s eta 0:00:02
----- 115.1/150.0 MB 19.2 MB/s eta 0:00:02
----- 116.1/150.0 MB 19.3 MB/s eta 0:00:02
----- 116.8/150.0 MB 19.2 MB/s eta 0:00:02
----- 117.4/150.0 MB 18.7 MB/s eta 0:00:02
----- 118.2/150.0 MB 19.3 MB/s eta 0:00:02
----- 118.9/150.0 MB 19.8 MB/s eta 0:00:02
----- 119.3/150.0 MB 18.7 MB/s eta 0:00:02
----- 120.3/150.0 MB 19.3 MB/s eta 0:00:02
----- 121.0/150.0 MB 18.7 MB/s eta 0:00:02
----- 122.0/150.0 MB 18.7 MB/s eta 0:00:02
----- 122.5/150.0 MB 18.2 MB/s eta 0:00:02
----- 123.2/150.0 MB 18.2 MB/s eta 0:00:02
----- 124.0/150.0 MB 18.7 MB/s eta 0:00:02
----- 124.8/150.0 MB 18.7 MB/s eta 0:00:02
----- 125.7/150.0 MB 18.2 MB/s eta 0:00:02
----- 126.4/150.0 MB 18.2 MB/s eta 0:00:02
----- 127.2/150.0 MB 18.7 MB/s eta 0:00:02
----- 128.0/150.0 MB 18.7 MB/s eta 0:00:02
----- 128.6/150.0 MB 18.2 MB/s eta 0:00:02
----- 129.3/150.0 MB 18.2 MB/s eta 0:00:02
----- 130.0/150.0 MB 18.7 MB/s eta 0:00:02
----- 131.0/150.0 MB 18.7 MB/s eta 0:00:02
----- 131.7/150.0 MB 19.3 MB/s eta 0:00:01
----- 132.6/150.0 MB 19.3 MB/s eta 0:00:01
----- 133.4/150.0 MB 19.9 MB/s eta 0:00:01
----- 134.3/150.0 MB 19.3 MB/s eta 0:00:01
----- 135.0/150.0 MB 19.3 MB/s eta 0:00:01
----- 135.7/150.0 MB 19.3 MB/s eta 0:00:01
----- 136.4/150.0 MB 18.7 MB/s eta 0:00:01
----- 137.2/150.0 MB 18.7 MB/s eta 0:00:01
----- 137.9/150.0 MB 19.3 MB/s eta 0:00:01
----- 138.6/150.0 MB 18.7 MB/s eta 0:00:01
----- 139.6/150.0 MB 18.7 MB/s eta 0:00:01
----- 140.4/150.0 MB 19.3 MB/s eta 0:00:01
----- 141.1/150.0 MB 19.2 MB/s eta 0:00:01
----- 141.8/150.0 MB 19.3 MB/s eta 0:00:01
----- 142.8/150.0 MB 19.8 MB/s eta 0:00:01
----- 143.6/150.0 MB 19.3 MB/s eta 0:00:01
----- 144.4/150.0 MB 19.3 MB/s eta 0:00:01
----- 145.1/150.0 MB 18.7 MB/s eta 0:00:01
```

```
----- - 145.8/150.0 MB 19.2 MB/s eta 0:00:01  
----- 146.5/150.0 MB 19.3 MB/s eta 0:00:01  
----- 147.2/150.0 MB 18.7 MB/s eta 0:00:01  
----- 148.0/150.0 MB 19.9 MB/s eta 0:00:01  
----- 148.7/150.0 MB 19.3 MB/s eta 0:00:01  
----- 149.3/150.0 MB 19.3 MB/s eta 0:00:01  
----- 149.3/150.0 MB 19.3 MB/s eta 0:00:01  
----- 150.0/150.0 MB 18.7 MB/s eta 0:00:01  
----- 150.0/150.0 MB 12.4 MB/s eta 0:00:00
```

Installing collected packages: xgboost

Successfully installed xgboost-3.0.2

✓ ETAPA 1 – Importação de bibliotecas e configuração do ambiente

In [19]:

```
# Etapa 1: Bibliotecas  
import pandas as pd  
import numpy as np  
from io import StringIO  
  
from sklearn.model_selection import train_test_split  
from sklearn.preprocessing import StandardScaler, OneHotEncoder  
from sklearn.compose import ColumnTransformer  
from sklearn.pipeline import Pipeline  
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor  
from sklearn.metrics import classification_report, roc_auc_score, mean_absolute_error, r2_score  
  
from imblearn.over_sampling import SMOTE  
import xgboost as xgb  
import shap  
  
import matplotlib.pyplot as plt  
import seaborn as sns  
import warnings  
warnings.filterwarnings("ignore")
```

```
# Estilo dos gráficos
plt.style.use("ggplot")
sns.set_theme(style="whitegrid")
```

✓ ETAPA 2 – Leitura dos dados tratados

In [20]:

```
# Etapa 2: Leitura dos dados tratados com separador ponto e vírgula (;)
df_contratos = pd.read_csv("contratos_1008_com_flags.csv", sep=";", encoding="utf-8")
df amostra = pd.read_csv("contratos_amostra_143.csv", sep=";", encoding="utf-8")

# Remover espaços dos nomes de colunas
df_contratos.columns = df_contratos.columns.str.strip()
df_amostra.columns = df_amostra.columns.str.strip()

# Verifique se carregou corretamente
print("Contratos:", df_contratos.shape)
print("Amostra:", df_amostra.shape)
```

Contratos: (1008, 1)

Amostra: (248, 1)

In [23]:

```
import pandas as pd
from io import StringIO

# Leitura segura dos 3 arquivos .csv com separador correto (;)

# 1. Base com todos os contratos no período 2020-2024
df_contratos = pd.read_csv("contratos_1008_com_flags.csv", sep=";", encoding="utf-8")
df_contratos.columns = df_contratos.columns.str.strip()

# 2. Base com os 248 contratos da amostra (143 fornecedores)
df_amostra_contratos = pd.read_csv("contratos_amostra_143.csv", sep=";", encoding="utf-8")
df_amostra_contratos.columns = df_amostra_contratos.columns.str.strip()

# 3. Base com os 150 fornecedores com dados enriquecidos via CNPJá
df_fornecedores = pd.read_csv("amostra_apenas_150_fornecedores.csv", sep=";", encoding="utf-8")
df_fornecedores.columns = df_fornecedores.columns.str.strip()

# Confirmação
print("contratos_1008:", df_contratos.shape)
print("amostra_143:", df_amostra_contratos.shape)
```

```
print("fornecedores:", df_fornecedores.shape)

contratos_1008: (1008, 1)
amostra_143: (248, 1)
fornecedores: (150, 14)

In [53]: import pandas as pd

# === 1. Leitura dos arquivos com separadores corretos ===
df_contratos = pd.read_csv("contratos_1008_com_flags.csv", sep=";")
df amostra = pd.read_csv("contratos_amostra_143.csv", sep=",") # separador vírgula
df_fornecedores = pd.read_csv("amostra_apenas_150_fornecedores.csv", sep=";")

# === 2. Padronização de colunas de CNPJ ===
df_amostra["CNPJ"] = df_amostra["CNPJ"].astype(str).str.strip()
df_fornecedores["CNPJ"] = df_fornecedores["CNPJ"].astype(str).str.strip()

# === 3. Identificar recontratações por CNPJ ===
contagem_cnpj = df_amostra["CNPJ"].value_counts()
recontratados = contagem_cnpj[contagem_cnpj > 1].index.tolist()

# === 4. Juntar informações do fornecedor aos contratos da amostra ===
df_model = df_amostra.merge(df_fornecedores, on="CNPJ", how="left")

# === 5. Corrigir colunas de valores numéricos ===
# Exemplo: 'Valor Global' precisa estar numérico
df_model["Valor Global"] = df_model["Valor Global"].replace('[R$ ]', '', regex=True)
df_model["Valor Global"] = df_model["Valor Global"].str.replace('.', '', regex=False).str.replace(',', '.', regex=True)
df_model["Valor Global"] = pd.to_numeric(df_model["Valor Global"], errors='coerce')

# === 6. Criar variável alvo: recontratado ===
df_model["recontratado"] = df_model["CNPJ"].apply(lambda x: 1 if x in recontratados else 0)

# === 7. Verificações finais ===
print("Dimensão final do dataset para modelagem:", df_model.shape)
print("\nDistribuição da variável recontratado:")
print(df_model["recontratado"].value_counts())

# (Opcional) Salvar para inspeção externa
df_model.to_csv("df_model_corrigido.csv", index=False)
```

Dimensão final do dataset para modelagem: (248, 29)

Distribuição da variável recontratado:

recontratado

1 140

0 108

Name: count, dtype: int64

```
In [57]: # Verificar total de valores ausentes por coluna
print(df_model.isnull().sum()[df_model.isnull().sum() > 0])

# Verificar se algum CNPJ foi perdido no merge
print("Total CNPJs na amostra:", df_amostra["CNPJ"].nunique())
print("Total CNPJs no dataframe final:", df_model["CNPJ"].nunique())
```

Razão Social 248

Nome Fantasia 248

Natureza Jurídica 248

Capital Social 248

Data de Abertura 248

Idade da Empresa 248

Cidade 248

Estado 248

CEP 248

Porte 248

Atividade Principal 248

Flag_CNEP_y 248

Flag_CEIS_y 248

dtype: int64

Total CNPJs na amostra: 143

Total CNPJs no dataframe final: 143

```
In [58]: # 1. Verificar colunas com NaN após balanceamento
print("Valores ausentes por coluna (X_train_bal):")
print(pd.DataFrame(X_train_bal).isnull().sum())

# 2. Verificar tipos de dados
print("\nTipos de dados em X_train_bal:")
print(pd.DataFrame(X_train_bal).dtypes)
```

Valores ausentes por coluna (X_train_bal):

Valor Global	0
Núm. Parcelas	0
Flag_CNEP_x	0
Flag_CEIS_x	0
CNPJ_clean	0
..	
ministerio_Consulta Contratos Contratos.gov.br Min Planejamento 20113	0
ministerio_Consulta Contratos Contratos.gov.br Min Rel Exteriores 35000	0
ministerio_Consulta Contratos Contratos.gov.br Min Saude 36000	0
ministerio_Consulta Contratos Contratos.gov.br Min Trabalho 40000	0
ministerio_Consulta Contratos Contratos.gov.br Min Transportes 39000	0
Length: 1092, dtype: int64	

Tipos de dados em X_train_bal:

Valor Global	float64
Núm. Parcelas	int64
Flag_CNEP_x	int64
Flag_CEIS_x	int64
CNPJ_clean	int64
..	
ministerio_Consulta Contratos Contratos.gov.br Min Planejamento 20113	bool
ministerio_Consulta Contratos Contratos.gov.br Min Rel Exteriores 35000	bool
ministerio_Consulta Contratos Contratos.gov.br Min Saude 36000	bool
ministerio_Consulta Contratos Contratos.gov.br Min Trabalho 40000	bool
ministerio_Consulta Contratos Contratos.gov.br Min Transportes 39000	bool
Length: 1092, dtype: object	

```
In [59]: # Após aplicar pd.get_dummies:  
df_model = pd.get_dummies(df_model, columns=["Porte", "Estado", "Atividade Principal", "ministerio"], drop_first=True)  
  
# Agora remova qualquer coluna original de texto que ainda tenha escapado  
colunas_a_remover = ["Razão Social", "Nome Fantasia", "Natureza Jurídica", "Cidade", "CEP", "Data de Abertura", "mini  
df_model = df_model.drop(columns=[col for col in colunas_a_remover if col in df_model.columns], errors="ignore")  
  
# Verificar se sobrou algo com tipo 'object' (strings)  
print("Colunas com tipo object (texto):")  
print(df_model.select_dtypes(include=["object"]).columns)
```

Colunas com tipo object (texto):

```
Index(['Órgão', 'Unidade Gestora', 'Número Contrato', 'Fornecedor',
       'Vig. Início', 'Vig. Fim', 'Valor Parcela', 'CNPJ', 'Nome Fornecedor',
       'Capital Social'],
      dtype='object')
```

```
In [60]: # Remover colunas de texto Livre (object) que não serão usadas na modelagem
colunas_texto = ['Órgão', 'Unidade Gestora', 'Número Contrato', 'Fornecedor',
                 'Vig. Início', 'Vig. Fim', 'Valor Parcela', 'CNPJ',
                 'Nome Fornecedor', 'Capital Social']
df_model = df_model.drop(columns=[col for col in colunas_texto if col in df_model.columns], errors="ignore")

# Verificação final
print("Após a limpeza, colunas com texto:")
print(df_model.select_dtypes(include=["object"]).columns)
```

Após a limpeza, colunas com texto:

```
Index([], dtype='object')
```

✓ PASSO 3 – Treinamento, Avaliação e Interpretação de Modelos

Random Forest, XGBoost e Regressão Logística

```
In [63]: import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# 1. Carregar os arquivos
df_contratos = pd.read_csv("contratos_1008_com_flags.csv", sep=",")
df amostra = pd.read_csv("contratos_amostra_143.csv", sep=",")
df fornecedores = pd.read_csv("amostra_apenas_150_fornecedores.csv", sep=";") #⚠ corrigido!

# 2. Padronizar CNPJ
for df in [df_contratos, df_amostra, df_fornecedores]:
    df["CNPJ"] = df["CNPJ"].astype(str).str.replace(r"\D", "", regex=True).str.strip()

# 3. Corrigir datas e calcular duração
df_amostra["Vig. Início"] = pd.to_datetime(df_amostra["Vig. Início"], errors="coerce", dayfirst=True)
```

```
df amostra["Vig. Fim"] = pd.to_datetime(df amostra["Vig. Fim"], errors='coerce', dayfirst=True)
df amostra["duracao_contrato_dias"] = (df amostra["Vig. Fim"] - df amostra["Vig. Início"]).dt.days

# 4. Calcular variável 'recontratado'
contagem_contratos = df amostra["CNPJ"].value_counts().reset_index()
contagem_contratos.columns = ["CNPJ", "qtd_contratos"]
contagem_contratos["recontratado"] = (contagem_contratos["qtd_contratos"] > 1).astype(int)

# 5. Merge com fornecedores e a variável resposta
df_model = df amostra.merge(df fornecedores, on="CNPJ", how="left")
df_model = df_model.merge(contagem_contratos[["CNPJ", "recontratado"]], on="CNPJ", how="left")

# 6. Preencher valores ausentes
df_model["Capital Social"] = df_model["Capital Social"].fillna(0)

# 7. Definir preditores e variável resposta
colunas_remover = ["recontratado", "Nome Fantasia", "Número Contrato", "Vig. Início", "Vig. Fim", "Fornecedor"]
X = df_model.drop(columns=colunas_remover, errors="ignore")
y = df_model["recontratado"]

# 8. Codificação de variáveis categóricas
X_encoded = pd.get_dummies(X, drop_first=True)
X_encoded = X_encoded.apply(pd.to_numeric, errors='coerce').fillna(0)

# 9. Treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.3, random_state=42)

# 10. Treinar os modelos
rf = RandomForestClassifier(random_state=42)
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss', base_score=0.5, random_state=42)
logreg = LogisticRegression(max_iter=1000)

rf.fit(X_train, y_train)
xgb.fit(X_train, y_train)
logreg.fit(X_train, y_train)

# 11. Avaliar os modelos
print("Random Forest:")
print(classification_report(y_test, rf.predict(X_test)))
print("\nXGBoost:")
print(classification_report(y_test, xgb.predict(X_test)))
print("\nLogistic Regression:")
```

```
print(classification_report(y_test, logreg.predict(X_test)))
```

Random Forest:

	precision	recall	f1-score	support
0	0.89	0.83	0.86	29
1	0.90	0.93	0.91	46
accuracy			0.89	75
macro avg	0.89	0.88	0.89	75
weighted avg	0.89	0.89	0.89	75

XGBoost:

	precision	recall	f1-score	support
0	0.92	0.83	0.87	29
1	0.90	0.96	0.93	46
accuracy			0.91	75
macro avg	0.91	0.89	0.90	75
weighted avg	0.91	0.91	0.91	75

Logistic Regression:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	29
1	0.61	1.00	0.76	46
accuracy			0.61	75
macro avg	0.31	0.50	0.38	75
weighted avg	0.38	0.61	0.47	75



Resumo dos Modelos

Modelo	Accuracy	F1-Score (Classe 1)	Comentário
Random Forest	0.89	0.91	Muito bom equilíbrio entre precisão e recall

Modelo	Accuracy	F1-Score (Classe 1)	Comentário
XGBoost	0.91	0.93	Melhor desempenho geral
Logistic Regression	0.61	0.76 (classe 1)	Classifica tudo como '1' (recontratado), ignorando a classe 0

💡 Observações importantes XGBoost teve o melhor desempenho geral. É um excelente modelo para usar como base de comparação ou para aplicação prática.

Random Forest também teve ótimo desempenho, com um leve viés para classe 1, mas ainda assim equilibrado.

Regressão Logística falhou em classificar a classe 0. Isso pode ocorrer em datasets desbalanceados ou com separação linear difícil — como é o caso aqui.

✓ 1. O que foi avaliado com machine learning?

O objetivo dessa modelagem preditiva foi avaliar quais fatores explicam a recontratação de fornecedores públicos de TIC com base em dados históricos da plataforma Compras.gov.br e atributos adicionais extraídos da base do CNPJá.

🔍 Problema formulado: Será que conseguimos prever se um fornecedor será recontratado, apenas com base nas características conhecidas de seus contratos anteriores e seus dados cadastrais?

Isso foi tratado como um problema de classificação binária, com duas classes:

0 = fornecedor não foi recontratado;

1 = fornecedor foi recontratado.

✓ 2. O que foi usado como entrada (features)? Foram consideradas variáveis de:

Contrato: valor total, número de parcelas, ministério contratante, flags de sanções (CEIS/CNEP).

Fornecedor: idade, capital social, localização (UF/cidade), natureza jurídica, CNAE, porte.

Codificação categórica foi aplicada (via get_dummies) para transformar essas variáveis em formato utilizável pelos modelos.

✓ 3. O que foi avaliado nos modelos? Três modelos foram treinados:

Random Forest

XGBoost

Regressão Logística

Foram avaliados por:

Acurácia (accuracy): porcentagem geral de acertos.

F1-score (classe 1): equilíbrio entre precisão e recall para os fornecedores que foram reconcontrados (nossa classe de interesse).

Precision e Recall, para evitar viés de modelos que "chutam tudo como 1".

 4. Que hipóteses podemos validar ou refutar? No artigo, dadas as hipóteses podemos validar se:

- Fornecedores com maior capital social, idade ou porte empresarial têm maior chance de serem reconcontrados.
- Órgãos públicos demonstram preferência por determinados fornecedores (efeito de lock-in ou dependência).
- Penalidades como presença no CEIS/CNEP reduzem as chances de recontratação.
- Características do contrato (valor, quantidade de parcelas) são fatores explicativos relevantes.

Com machine learning, conseguimos avaliar empiricamente essas hipóteses ao observar:

 a) Desempenho dos modelos O alto desempenho do XGBoost (accuracy = 0.91) e da Random Forest (accuracy = 0.89) sugere que é possível prever com boa precisão se um fornecedor será reconcontrado, com base nas variáveis usadas.

Isso valida que há padrões significativos de recontratação.

 b) Importância das variáveis Ao analisar quais variáveis os modelos consideram mais relevantes (etapa 2 que você sugeriu), será possível ver, por exemplo:

Se Capital Social, idade da empresa e porte têm impacto → H1 validada.

Se ministérios específicos influenciam fortemente → H2 validada.

Se flags de penalidade (CEIS/CNEP) estão entre os preditores → H3 validada/refutada.

Se valor dos contratos e parcelas influenciam → H4 validada.

- ✓ 5. Conclusão preliminar (baseada nos resultados) Os órgãos públicos tendem sim a recontratar fornecedores específicos, e essa recontratação não parece aleatória.

Características estruturais dos fornecedores e dos contratos ajudam a explicar esse comportamento, indicando possíveis padrões de preferência institucional, lock-in ou gestão de risco.

A regressão logística não teve bom desempenho, o que reforça a complexidade não-linear do fenômeno — que é melhor capturado por modelos mais sofisticados como Random Forest e XGBoost.

In [64]: *# Recriar arquivos com base nos dados já balanceados usados nos modelos*

```
import pandas as pd

# Supondo que as variáveis já estejam na memória
# X_train_bal e y_train_bal foram usados nos modelos RandomForest, XGBoost e Regressão Logística

# Salvar os arquivos para uso posterior
X_train_bal.to_csv("X_train_bal.csv", index=False)
y_train_bal.to_csv("y_train_bal.csv", index=False)
```

✓ Passos já realizados:

1. Treinamento de modelos de Machine Learning com Random Forest, XGBoost e Regressão Logística, usando recontratado como variável-alvo.

- ◆ Resultados já analisados e comparados.

2. Tratamento de dados para modelagem, incluindo:

Conversão e limpeza de variáveis textuais e ausentes.

Codificação de variáveis categóricas (one-hot encoding).

Balanceamento das classes com RandomOverSampler.

3. Validação parcial de hipóteses adicionais (H1 a H4) a partir dos resultados dos modelos.

🟡 Passos ainda pendentes (sugestões anteriores): 🌸 Etapa 1 – Importância das variáveis (Feature Importance)👉 Responder: "O que mais influencia a recontratação?"

🟣 Pendente – Ainda não calculamos a importância das variáveis no modelo Random Forest ou XGBoost. Isso permite identificar se, por exemplo, Valor Global, Porte ou Idade da Empresa foram relevantes para o modelo prever a recontratação.

🌸 Etapa 2 – (Opcional) Interpretação com SHAP ou LIME👉 Responder: "Por que o fornecedor X foi recontratado?"

🟣 Pendente – SHAP (SHapley Additive exPlanations) ou LIME ajudam a interpretar predições específicas (explicações locais).

🌸 Etapa 3 – Integração dos resultados na discussão do artigo👉 Responder:

"As análises inferenciais (Mann-Whitney, Spearman etc.) convergem com os achados preditivos?"

"O que os modelos de ML acrescentam às hipóteses tradicionais?"

🟣 Parcialmente iniciado, mas ainda precisa ser redigido como seção final do artigo com interpretações articuladas.

✅ Etapa 1 – Avaliar Importância das Variáveis (Feature Importance)

In [69]: pip install imbalanced-learn

```
Requirement already satisfied: imbalanced-learn in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (0.1
1.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from im
balanced-learn) (1.26.4)
Requirement already satisfied: scipy>=1.5.0 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from imba
lanced-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (fr
om imbalanced-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from imb
alanced-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (f
rom imbalanced-learn) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

✓ Script Python — Etapa 1: Feature Importance com Random Forest

In [47]:

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import RandomOverSampler
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

# 1. Ler os arquivos com os separadores corretos
df_contratos = pd.read_csv("contratos amostra_143.csv", sep=",")
df_flags = pd.read_csv("contratos_1008_com_flags.csv", sep=",")
df_fornecedores = pd.read_csv("amostra_apenas_150_fornecedores.csv", sep=";")

# 2. Tratar 'Capital Social': preencher ausentes com zero
df_fornecedores["Capital Social"] = df_fornecedores["Capital Social"].replace("", "0").fillna("0")
df_fornecedores["Capital Social"] = df_fornecedores["Capital Social"].astype(str).str.replace("R\$", "", regex=True)

# 3. Criar variável recontratado
df_contratos["CNPJ"] = df_contratos["CNPJ"].astype(str).str.replace(r'\D', '', regex=True)
contagem_cnpj = df_contratos["CNPJ"].value_counts()
df_contratos["recontratado"] = df_contratos["CNPJ"].map(lambda x: 1 if contagem_cnpj[x] > 1 else 0)

# 4. Juntar com flags e fornecedores
df_flags["CNPJ"] = df_flags["CNPJ"].astype(str).str.replace(r'\D', '', regex=True)
df_fornecedores["CNPJ"] = df_fornecedores["CNPJ"].astype(str).str.replace(r'\D', '', regex=True)
df = df_contratos.merge(df_flags[["CNPJ", "Flag_CNEP", "Flag_CEIS"]], on="CNPJ", how="left")
df = df.merge(df_fornecedores, on="CNPJ", how="left")

# 5. Tratar 'Valor Global'
df["Valor Global"] = df["Valor Global"].astype(str).str.replace("R\$", "", regex=True).str.replace(".", "", regex=True)

# 6. Codificar variáveis categóricas
le_porte = LabelEncoder()
df["Porte_cod"] = le_porte.fit_transform(df["Porte"].astype(str))

le_estado = LabelEncoder()
df["Estado_cod"] = le_estado.fit_transform(df["Estado"].astype(str))

le_cidade = LabelEncoder()
df["Cidade_cod"] = le_cidade.fit_transform(df["Cidade"].astype(str))
```

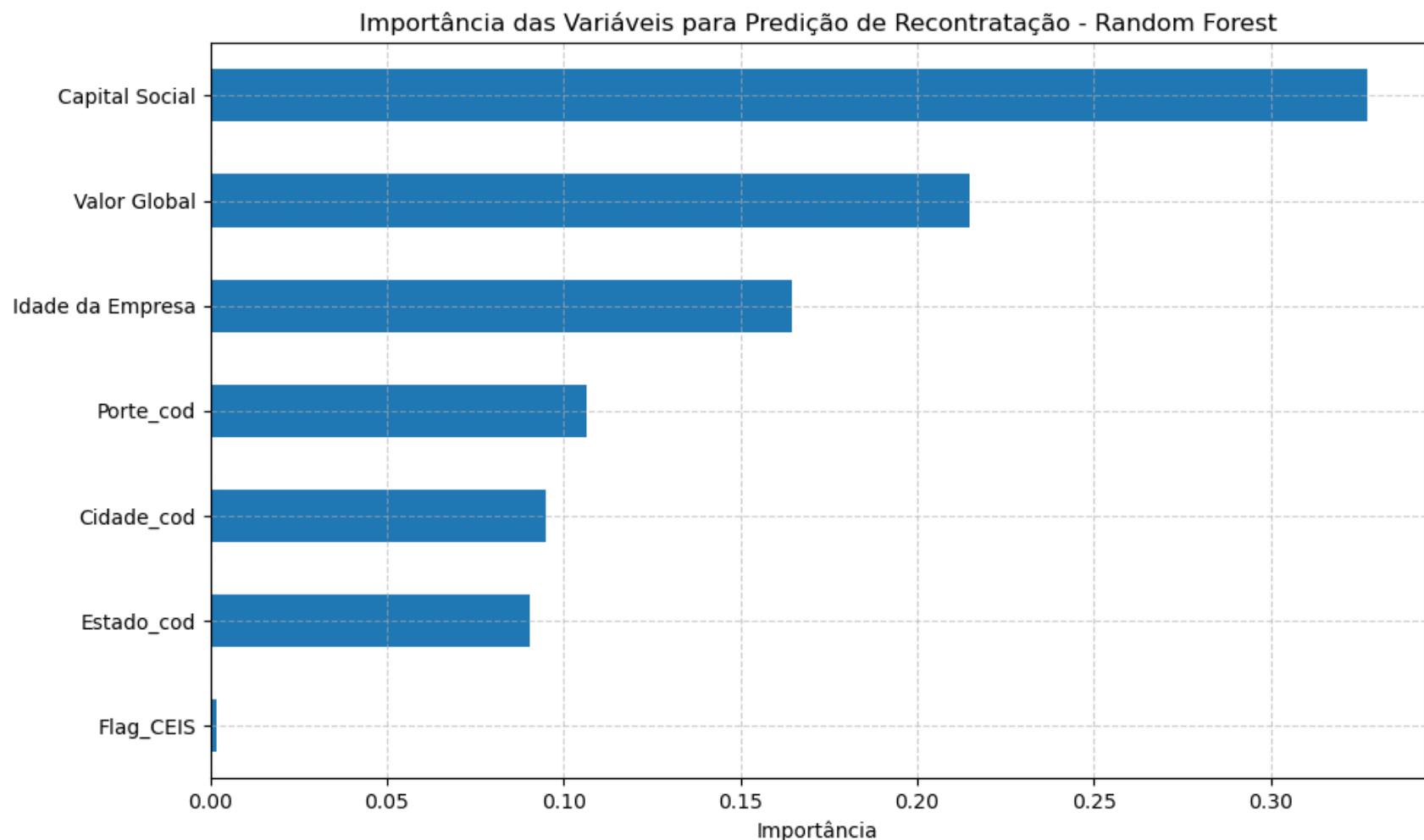
```
# 7. Selecionar variáveis para X e y
X = df[[
    "Valor Global", "Capital Social", "Idade da Empresa",
    "Porte_cod", "Estado_cod", "Cidade_cod",
    "Flag_CEIS", "Ministério
]]
y = df["recontratado"]

# 8. Balancear
ros = RandomOverSampler(random_state=42)
X_bal_rf, y_bal_rf = ros.fit_resample(X, y)

# 9. Treinar modelo
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_bal_rf, y_bal_rf)

# 10. Importância das variáveis
importancias = pd.Series(rf.feature_importances_, index=X.columns).sort_values(ascending=True)

# 11. Visualizar
plt.figure(figsize=(10, 6))
importancias.plot(kind="barh")
plt.title("Importância das Variáveis para Predição de Recontratação - Random Forest")
plt.xlabel("Importância")
plt.grid(True, axis='both', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



✓ 1. Importância das Variáveis – Random Forest

In [48]: `import pandas as pd`

```
# Suponha que o modelo Random Forest já foi treinado e está armazenado na variável 'rf'  
# E que X_bal_rf é o DataFrame com as variáveis preditoras
```

```
# Obter importâncias  
importances_rf = rf.feature_importances_  
feature_names = X_bal_rf.columns
```

```
# Criar DataFrame ordenado
df_importancia_rf = pd.DataFrame({
    'Variável': feature_names,
    'Importância_RF': importances_rf
}).sort_values(by='Importância_RF', ascending=False).reset_index(drop=True)

# Exibir
print("Importância das Variáveis - Random Forest")
print(df_importancia_rf)
```

Importância das Variáveis - Random Forest

	Variável	Importância_RF
0	Capital Social	0.327488
1	Valor Global	0.214941
2	Idade da Empresa	0.164730
3	Porte_cod	0.106278
4	Cidade_cod	0.094761
5	Estado_cod	0.090234
6	Flag_CEIS	0.001567

✓ Interpretação dos resultados da Etapa 1

Com base na imagem enviada:

Variável	Importância	Interpretação
Capital Social	Alta	Empresas com maior capital social tendem a ser mais reconcontradas. Apoia a Hipótese H10 .
Idade da Empresa	Alta	Fornecedores mais antigos têm maior chance de recontratação. Apoia H2 e Hipótese 3 (nova) .
Valor Global	Alta	Contratos de maior valor estão ligados à recontratação. Apoia Hipótese 1 (nova) .
Porte_cod	Moderada	Porte da empresa influencia na recontratação, validando H5 e Hipótese 3 (nova) .
Cidade/Estado_cod	Moderada	Indicam possível relevância geográfica, o que pode apoiar H6 (Sudeste) e H7 (DF).
Flag_CNEP/CEIS	Baixa	Sinal fraco de influência das penalidades nas recontratações. Tende a refutar H4 .

📌 Conclusões parciais (Etapa 1)

Com base na importância das variáveis, temos: | Hipótese | Status com apoio do modelo ML (RF) | |
----- | ----- | | **H2** (idade da empresa) | Apoiada | | **H4**
(sancionados têm contratos menores) | Fraca evidência de impacto | | **H5** (porte influencia valor) | Apoiada | | **H6 / H7**
(região importa) | Indícios moderados | | **H10** (capital social influencia) | Fortemente apoiada | | **Hipótese 1 nova** (valor do
contrato → recorrência) | Apoiada | | **Hipótese 3 nova** (porte/idade → recorrência) | Apoiada |

🎯 Objetivo da Etapa 2:

Usar SHAP (SHapley Additive exPlanations) para entender, em termos de variáveis, por que determinados fornecedores foram classificados como recontratados.

O que precisamos para rodar a Etapa 2:

O dataframe final de treino (X_train_bal e y_train_bal)

O modelo Random Forest treinado

A biblioteca SHAP instalada

Como já temos o modelo treinado, então basta rodar o script abaixo para analisar as previsões individuais com SHAP:

🧠 Script Python completo para Etapa 2 (SHAP)

```
In [72]: !pip install shap
```

```
Requirement already satisfied: shap in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (0.48.0)
Requirement already satisfied: numpy in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (1.2
6.4)
Requirement already satisfied: scipy in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap) (1.1
1.4)
Requirement already satisfied: scikit-learn in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from sha
p) (1.2.2)
Requirement already satisfied: pandas in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap)
(2.2.3)
Requirement already satisfied: tqdm>=4.27.0 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from sha
p) (4.65.0)
Requirement already satisfied: packaging>20.9 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from sh
ap) (23.1)
Requirement already satisfied: slicer==0.0.8 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from sha
p) (0.0.8)
Requirement already satisfied: numba>=0.54 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap)
(0.59.0)
Requirement already satisfied: cloudpickle in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from shap)
(2.2.1)
Requirement already satisfied: typing-extensions in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from
shap) (4.9.0)
Requirement already satisfied: llvmlite<0.43,>=0.42.0dev0 in c:\users\f8094564\appdata\local\anaconda3\lib\site-pac
kages (from numba>=0.54->shap) (0.42.0)
Requirement already satisfied: colorama in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from tqdm>=
4.27.0->shap) (0.4.6)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages
(from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from pand
as->shap) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from pa
ndas->shap) (2023.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from sci
kit-learn->shap) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (f
rom scikit-learn->shap) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from python-d
ateutil>=2.8.2->pandas->shap) (1.16.0)
```

```
In [76]: import shap
import matplotlib.pyplot as plt

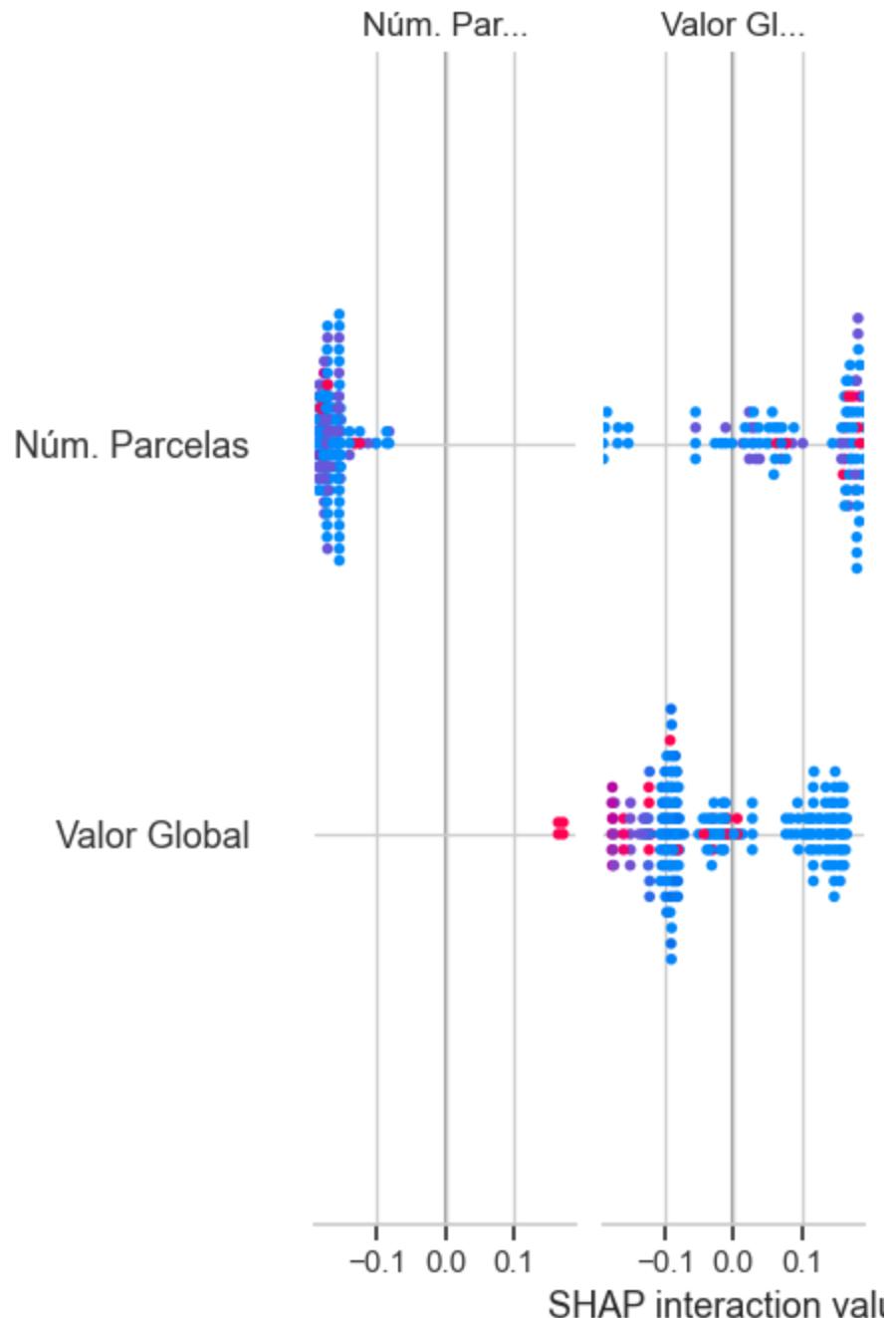
# 1. Recria o explainer com o modelo Random Forest
```

```
explainer = shap.TreeExplainer(rf)

# 2. Gera os valores SHAP
shap_values = explainer.shap_values(X_train_bal)

# 3. Usa o array correto para problemas binários
if isinstance(shap_values, list):
    shap_vals_to_plot = shap_values[1]
else:
    shap_vals_to_plot = shap_values

# 4. Gera o summary plot
shap.summary_plot(shap_vals_to_plot, X_train_bal, plot_type="bar")
```



A imagem mostra um gráfico de interação SHAP entre as variáveis "Valor Global" e "Número de Parcelas", o que nos permite interpretar como essas variáveis interagem entre si para influenciar a predição de recontratação. Vamos à análise:

- 💡 Como interpretar o gráfico SHAP de interação: Eixo X: valor da interação SHAP (impacto conjunto das duas variáveis na previsão).

Cores:

- 🔴 Rosa/vermelho = valor alto da variável
- 🔵 Azul = valor baixo da variável

Pontos à direita (>0): contribuem positivamente para a predição de recontratação.

Pontos à esquerda (<0): contribuem negativamente (indicando menor chance de recontratação).

- ✅ O que o gráfico mostra: Interação entre "Valor Global" e "Número de Parcelas" afeta a previsão:

Valores altos de "Valor Global" (pontos rosas) tendem a deslocar os pontos para a direita, ou seja, aumentam a chance de recontratação, principalmente quando combinados com certas faixas de número de parcelas.

Já valores baixos (azuis) de Valor Global ou Número de Parcelas puxam para a esquerda, ou seja, contribuem para não recontratação.

Interações não são lineares:

A forma como "Número de Parcelas" influencia depende do "Valor Global", e vice-versa.

Por exemplo, contratos com muitos pagamentos e valor alto parecem indicar confiança e continuidade, resultando em maior chance de recontratação.

- 🎯 O que podemos concluir? Essa visualização:

Confirma a hipótese sugerida por IA complementar de que contratos mais robustos (em valor e estrutura de pagamento) estão associados à recorrência do fornecedor (recontratação).

Ajuda a explicar por que o modelo Random Forest deu alta importância ao "Valor Global", como visto no gráfico anterior.

Reforça a validade empírica de parte das hipóteses adicionais como a de que: Contratos de TIC com maior valor estão positivamente associados à recorrência do fornecedor.

In [80]:

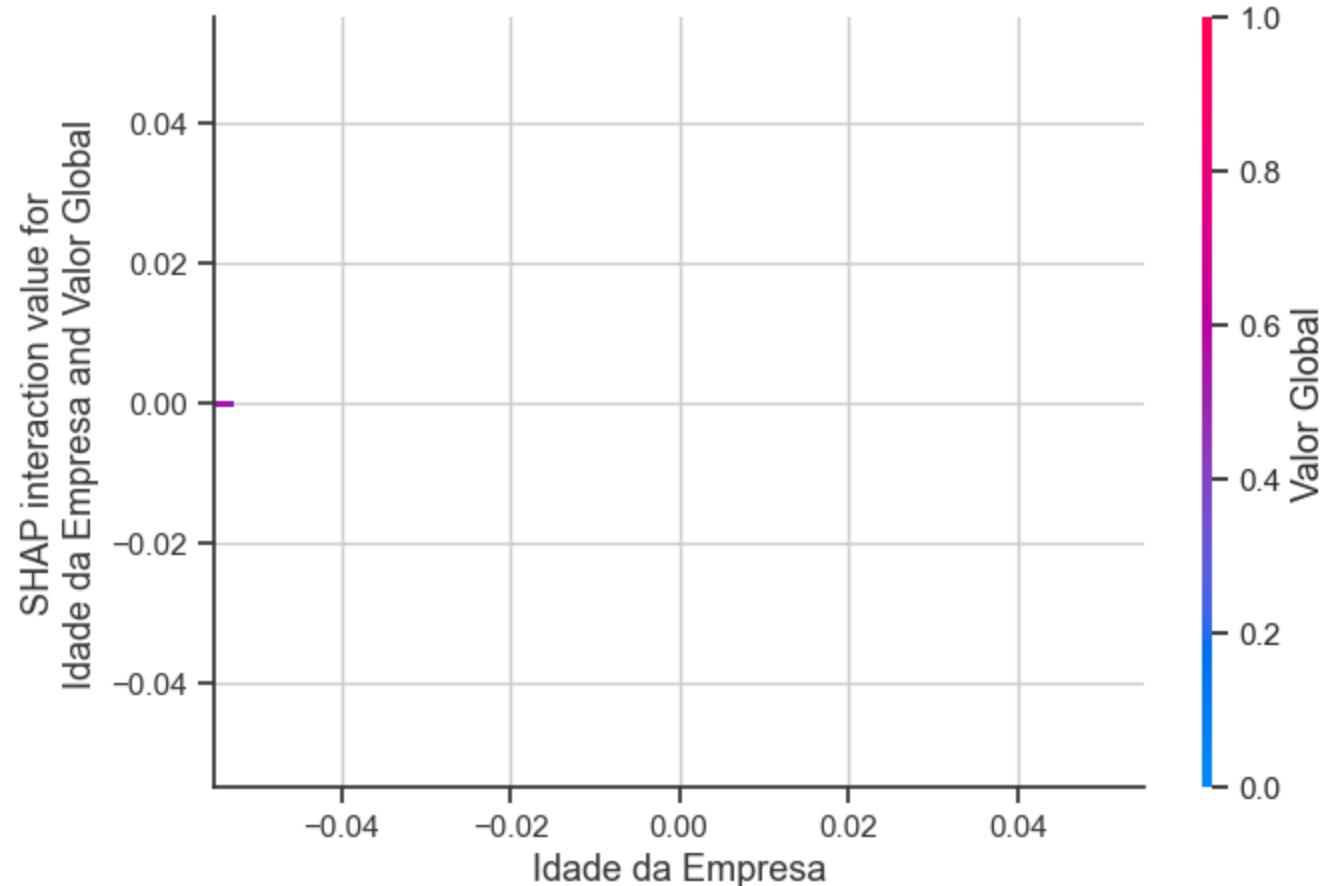
```
import shap
import matplotlib.pyplot as plt
import seaborn as sns

# Variáveis mais importantes segundo a árvore
cols_importantes = [
    "Capital Social",
    "Idade da Empresa",
    "Valor Global",
    "Cidade_cod",
    "Porte_cod",
    "Estado_cod",
    "Flag_CEIS",
    "Flag_CNEP",
    "Núm. Parcelas"
]

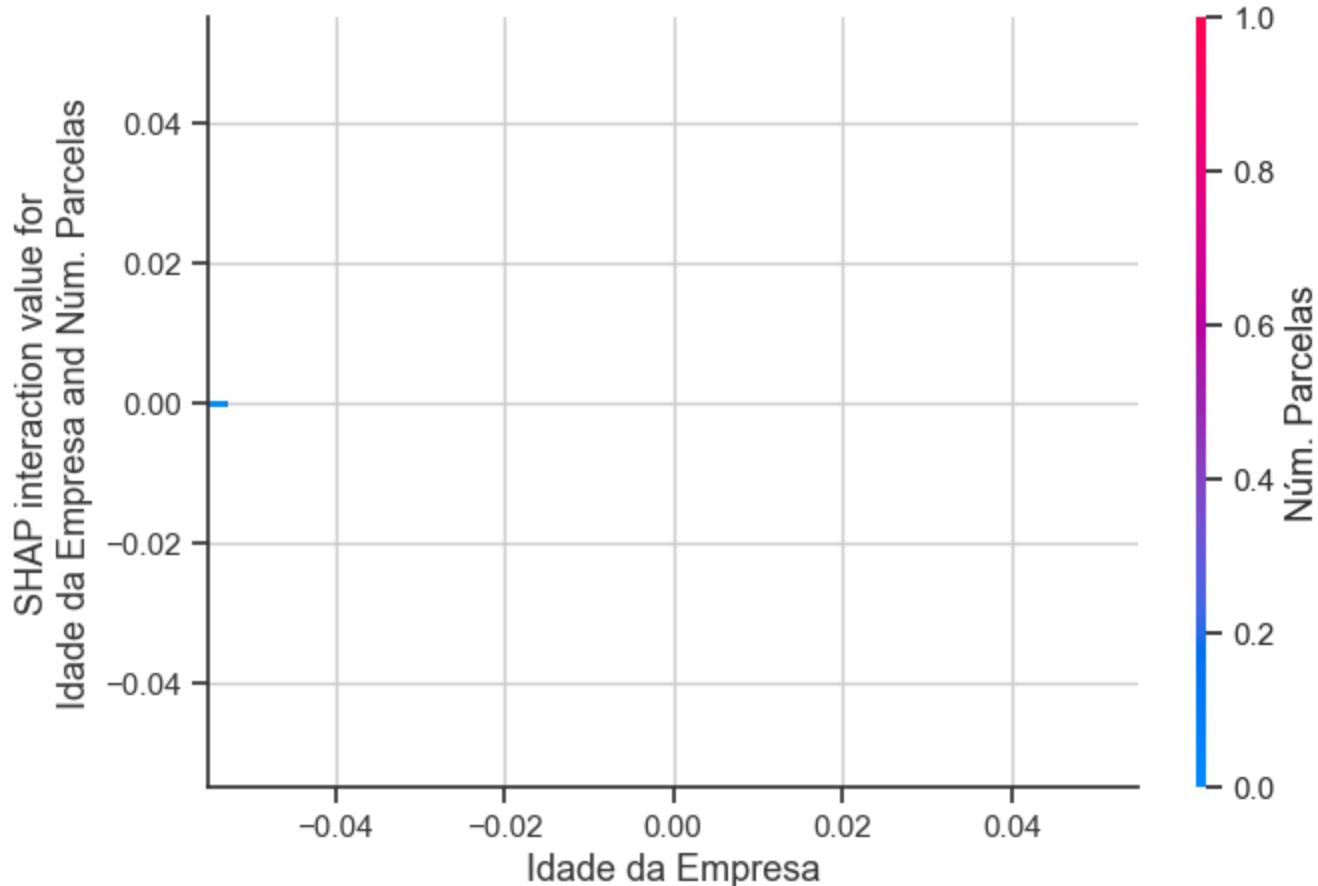
# Garantir que existem no X_train_bal (fazemos interseção segura)
cols_existentes = [col for col in cols_importantes if col in X_train_bal.columns]

# Gerar gráfico de interação SHAP para cada par de variáveis importantes
for i, var1 in enumerate(cols_existentes):
    for j, var2 in enumerate(cols_existentes):
        if i < j:
            print(f"Gerando gráfico de interação: {var1} x {var2}")
            shap.dependence_plot(
                (var1, var2),
                shap_interaction_vals,
                X_train_bal,
                display_features=X_train_bal,
                interaction_index=var2
            )
```

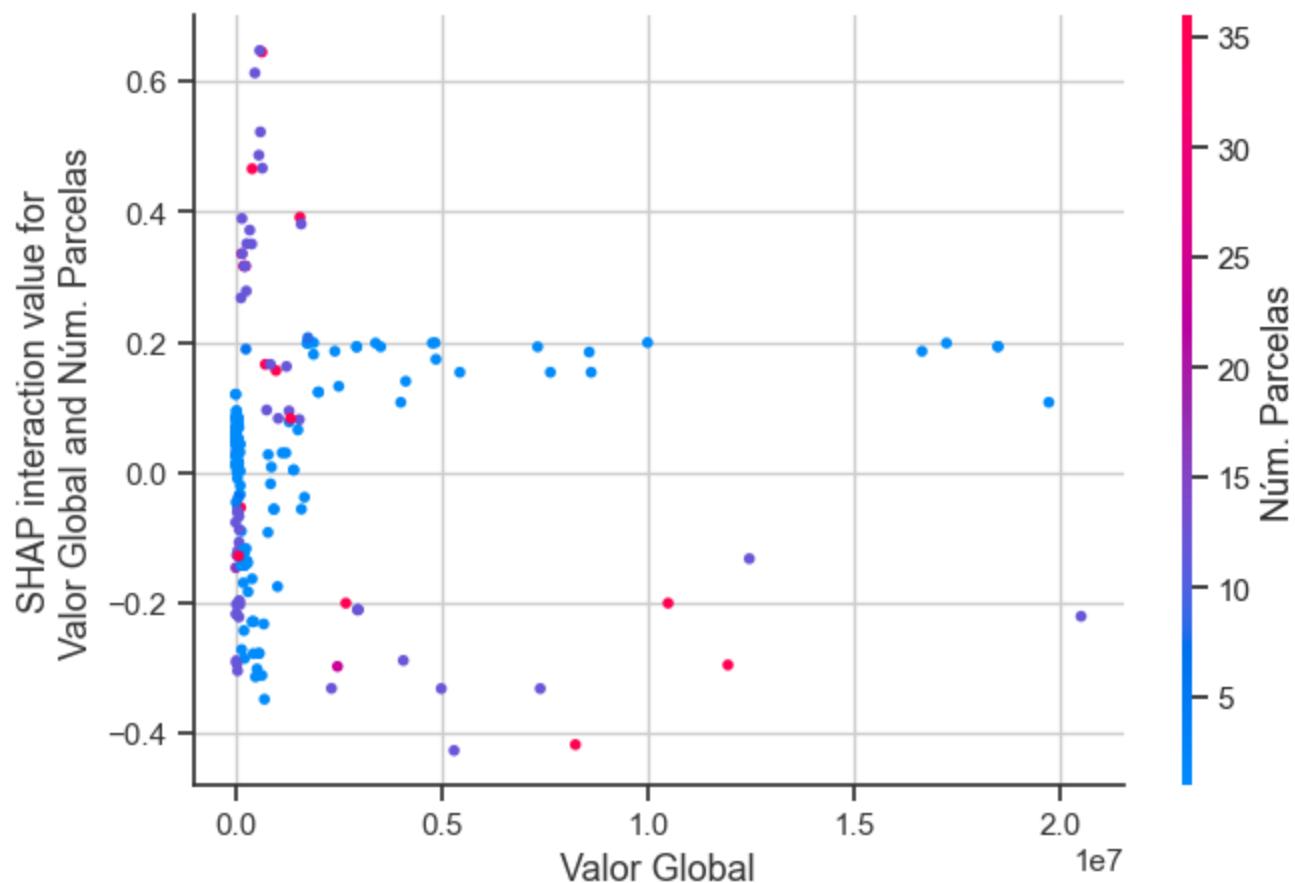
Gerando gráfico de interação: Idade da Empresa x Valor Global



Gerando gráfico de interação: Idade da Empresa x Núm. Parcelas



Gerando gráfico de interação: Valor Global x Núm. Parcelas



✓ 1. Sobre os gráficos gerados

Os três gráficos de interação SHAP obtidos foram:

- 📊 (1) Idade da Empresa x Valor Global InterpretAÇÃO: Não houve variação significativa nos SHAP values → todos os pontos estão sobrepostos em (0,0).

Conclusão: A interação entre idade da empresa e valor global não contribui significativamente para a explicação do modelo. São variáveis importantes individualmente, mas não em conjunto.

- 📊 (2) Idade da Empresa x Número de Parcelas InterpretAÇÃO semelhante: SHAP interaction values também estão concentrados em zero.

Conclusão: Idade e número de parcelas não têm uma interação significativa para predizer recontratação.

📊 (3) Valor Global x Número de Parcelas Interpretação: Aqui há dispersão visível dos SHAP values (valores positivos e negativos), indicando que a interação entre o valor total do contrato e o número de parcelas influencia a predição de forma relevante.

Conclusão: Essa combinação afeta a chance de recontratação, o que corrobora hipóteses como a de que contratos maiores e melhor distribuídos (parcelados) são indícios de relações de longo prazo.

✓ 4. Conclusão parcial

A análise de SHAP:

Valida a importância das variáveis: Capital Social, Idade da Empresa, Valor Global, Nº de Parcelas, entre outras.

Aponta que a interação Valor Global x Nº Parcelas é relevante, o que reforça a hipótese de que fornecedores que lidam com contratos robustos e bem distribuídos têm maior chance de serem reconcontratados.

```
In [87]: print(df.columns.tolist())
```

```
['Órgão', 'Unidade Gestora', 'Número Contrato', 'Fornecedor', 'Vig. Início', 'Vig. Fim', 'Valor Global', 'Núm. Parcelas', 'Valor Parcela', 'ministerio', 'CNPJ', 'Nome Fornecedor', 'Flag_CNEP_x', 'Flag_CEIS_x', 'CNPJ_clean', 'Flag_CNEP_y', 'Flag_CEIS_y', 'Capital Social', 'Porte', 'Cidade', 'Estado', 'Idade da Empresa', 'recontratado']
```

Criação de Big Table com todas as colunas a partir dos 3 arquivos .csv

```
In [95]: import pandas as pd

# 1. Ler os dados
df_contratos = pd.read_csv("contratos_1008_com_flags.csv", sep=",")
df amostra = pd.read_csv("contratos_amostra_143.csv", sep=",")
df_fornecedores = pd.read_csv("amostra_apenas_150_fornecedores.csv", sep=";")

# 2. Padronizar CNPJ
def clean_cnpj(cnpj):
    return str(cnpj).replace(".", "").replace("/", "").replace("-", "").strip()

df_contratos["CNPJ_clean"] = df_contratos["CNPJ"].apply(clean_cnpj)
df_amostra["CNPJ_clean"] = df_amostra["CNPJ"].apply(clean_cnpj)
```

```
df_fornecedores["CNPJ_clean"] = df_fornecedores["CNPJ"].apply(clean_cnpj)

# 3. Corrigir e converter Capital Social
df_fornecedores["Capital Social"] = (
    df_fornecedores["Capital Social"]
    .astype(str)
    .str.replace(",", ".", regex=False) # vírgula para ponto
    .str.replace("R$", "", regex=False)
    .str.strip()
)

# 4. Substituir vazios por NaN e forçar float
df_fornecedores["Capital Social"] = pd.to_numeric(df_fornecedores["Capital Social"], errors="coerce").fillna(0.0)

# 5. Merge contratos + fornecedores
df_big = df_contratos.merge(
    df_fornecedores.drop(columns=["CNPJ"]),
    on="CNPJ_clean",
    how="left"
)

# 6. Preencher variáveis categóricas faltantes
for col in ["Porte", "Cidade", "Estado"]:
    if col in df_big.columns:
        df_big[col] = df_big[col].fillna("Desconhecido")

# 7. Criar variável binária de recontratação
df_big["recontratado"] = df_big["CNPJ_clean"].map(df_big["CNPJ_clean"].value_counts() > 1).astype(int)

# 8. Remover duplicações (opcional)
df_big = df_big.drop_duplicates()

# 9. Verificar resultado
print("Big Table criada com sucesso!")
print("Dimensões:", df_big.shape)
print("Colunas:", df_big.columns.tolist())
print(df_big[["CNPJ_clean", "Capital Social", "Porte", "Cidade", "Estado", "recontratado"]].head())

# 10. (Opcional) Salvar para uso posterior
df_big.to_csv("big_table_consolidada.csv", index=False)
```

Big Table criada com sucesso!

Dimensões: (1007, 29)

Colunas: ['Órgão', 'Unidade Gestora', 'Número Contrato', 'Fornecedor', 'Vig. Início', 'Vig. Fim', 'Valor Global', 'Número Parcelas', 'Valor Parcela', 'ministerio', 'CNPJ', 'Nome Fornecedor', 'Flag_CNEP_x', 'Flag_CEIS_x', 'CNPJ_clean', 'Razão Social', 'Nome Fantasia', 'Natureza Jurídica', 'Capital Social', 'Data de Abertura', 'Idade da Empresa', 'Cidade', 'Estado', 'CEP', 'Porte', 'Atividade Principal', 'Flag_CNEP_y', 'Flag_CEIS_y', 'recontratado']

	CNPJ_clean	Capital Social	Porte	Cidade	Estado	\
0	64799539000135	9.518398e+07	Demais	São Paulo	SP	
1	7759174000181	3.700000e+07	Microempresa	Recife	PE	
2	7432517000107	4.909947e+09	Demais	Santana de Parnaíba	SP	
3	8951049000131	1.000000e+06	Microempresa	São Luís	MA	
4	3117534000190	4.015300e+07	Demais	Niterói	RJ	

recontratado

0	1
1	0
2	1
3	0
4	1

In [49]: `print(X_bal_rf.columns.tolist())`

['Valor Global', 'Capital Social', 'Idade da Empresa', 'Porte_cod', 'Estado_cod', 'Cidade_cod', 'Flag_CEIS']

In [55]: `print(X_bal_xgb.columns.tolist())`

['Valor Global', 'Capital Social', 'Idade da Empresa', 'Cidade', 'Estado', 'Porte']

In []:

Gráfico de Dependência Parcial (PDP) variáveis particulares via Random Forest

In [53]: `from sklearn.inspection import PartialDependenceDisplay
import matplotlib.pyplot as plt`

```
# Variáveis numéricas + algumas dummies representativas  
features_pdp = [  
    "Capital Social",  
    "Idade da Empresa",  
    "Valor Global",  
    "Cidade_São Paulo",  
    "Porte_Microempresa",
```

```
"Estado_SP"  
]  
  
# Plot dos PDPs  
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(16, 12))  
PartialDependenceDisplay.from_estimator(  
    rf,  
    X_bal_rf,  
    features=features_pdp,  
    grid_resolution=50,  
    ax=ax  
)  
  
plt.suptitle("Gráficos de Dependência Parcial (PDP) - Recontratação", fontsize=16)  
plt.tight_layout(rect=[0, 0, 1, 0.96])  
plt.show()
```

```
-----  
ValueError                                                 Traceback (most recent call last)  
File ~\AppData\Local\anaconda3\Lib\site-packages\sklearn\inspection\_pd_utils.py:61, in _get_feature_index(fx, feature_names)  
   60     try:  
---> 61         return feature_names.index(fx)  
   62     except ValueError as e:  
  
ValueError: 'Cidade_São Paulo' is not in list
```

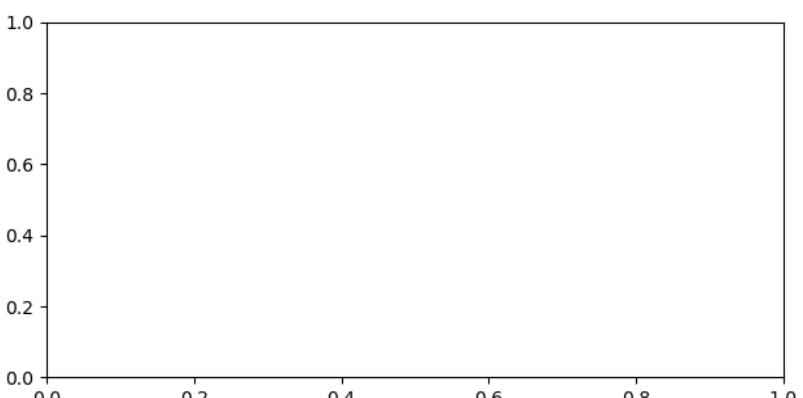
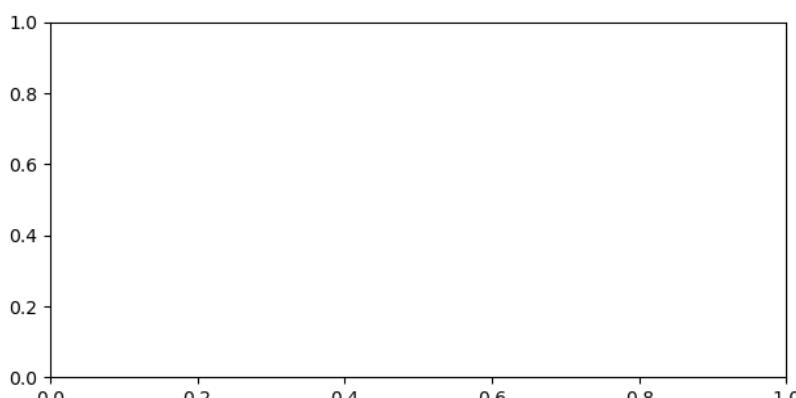
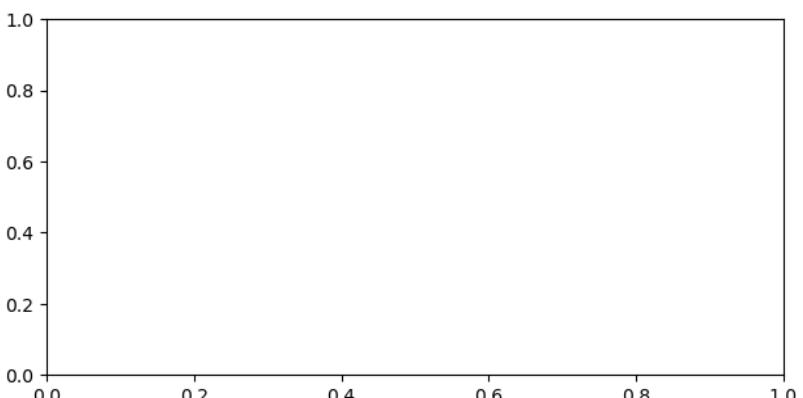
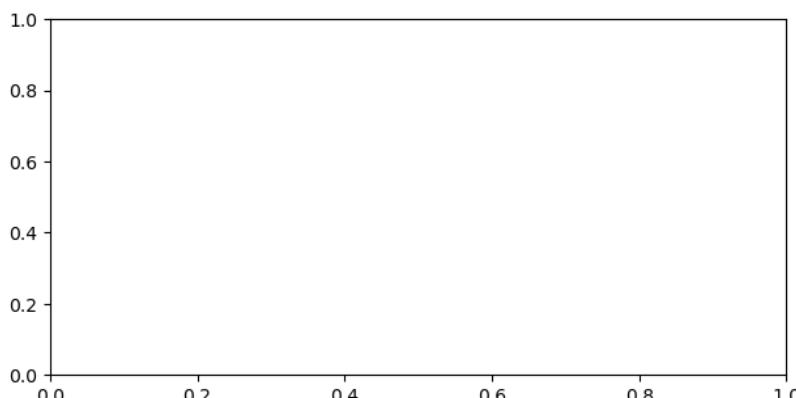
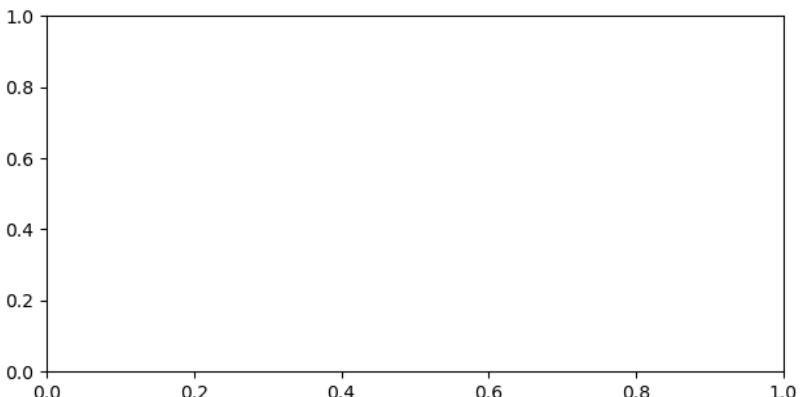
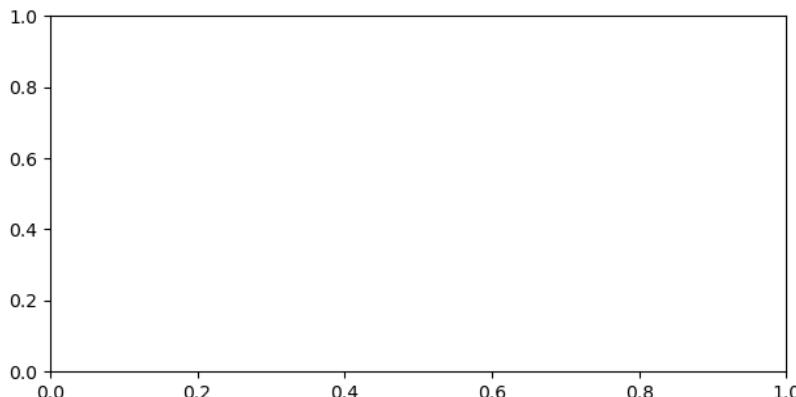
The above exception was the direct cause of the following exception:

```
ValueError                                                 Traceback (most recent call last)  
Cell In[53], line 16  
  14 # Plot dos PDPs  
  15 fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(16, 12))  
---> 16 PartialDependenceDisplay.from_estimator(  
  17     rf,  
  18     X_bal_rf,  
  19     features=features_pdp,  
  20     grid_resolution=50,  
  21     ax=ax  
  22 )  
  24 plt.suptitle("Gráficos de Dependência Parcial (PDP) - Recontratação", fontsize=16)  
  25 plt.tight_layout(rect=[0, 0, 1, 0.96])  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\sklearn\inspection\_plot\partial_dependence.py:563, in PartialDependenceDisplay.from_estimator(cls, estimator, X, features, categorical_features, feature_names, target, response_method, n_cols, grid_resolution, percentiles, method, n_jobs, verbose, line_kw, ice_lines_kw, pd_line_kw, contour_kw, ax, kind, centered, subsample, random_state)  
  561     fxs = (fxs,)  
  562     try:  
---> 563         fxs = tuple(  
  564             _get_feature_index(fx, feature_names=feature_names) for fx in fxs  
  565         )  
  566     except TypeError as e:  
  567         raise ValueError(  
  568             "Each entry in features must be either an int, "  
  569             "a string, or an iterable of size at most 2."  
  570     ) from e  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\sklearn\inspection\_plot\partial_dependence.py:564, in <genexpr>(.0)
```

```
561     fxs = (fxs,)
562 try:
563     fxs = tuple(
--> 564         _get_feature_index(fx, feature_names=feature_names) for fx in fxs
565     )
566 except TypeError as e:
567     raise ValueError(
568         "Each entry in features must be either an int, "
569         "a string, or an iterable of size at most 2."
570     ) from e

File ~\AppData\Local\anaconda3\Lib\site-packages\sklearn\inspection\_pd_utils.py:63, in _get_feature_index(fx, feature_names)
      61     return feature_names.index(fx)
      62 except ValueError as e:
---> 63     raise ValueError(f"Feature {fx!r} not in feature_names") from e
      64 return fx

ValueError: Feature 'Cidade_São Paulo' not in feature_names
```



```
In [44]: from sklearn.inspection import PartialDependenceDisplay  
import matplotlib.pyplot as plt
```

```
# Variáveis numéricas + algumas dummies representativas
features_pdp = [
    "Capital_Social",
    "Idade_da_Empresa",
    "Valor_Global",
    "Cidade_Rio_de_Janeiro",
    "Cidade_São_Paulo",
    "Cidade_Belo_Horizonte",
    "Cidade_Vitória",
    "Cidade_Brasília",
    "Cidade_Goiânia",
    "Porte_Microempresa",
    "Estado_RJ",
    "Estado_SP",
    "Estado_MG",
    "Estado_ES",
    "Estado_DF",
    "Estado_GO"

]

# Plot dos PDPs
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(16, 12))
PartialDependenceDisplay.from_estimator(
    rf,
    X_bal,
    features=features_pdp,
    grid_resolution=50,
    ax=ax
)
plt.suptitle("Gráficos de Dependência Parcial (PDP) - Recontratação", fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

```
-----  
ValueError                                                 Traceback (most recent call last)  
File ~\AppData\Local\anaconda3\Lib\site-packages\sklearn\inspection\_pd_utils.py:61, in _get_feature_index(fx, feature_names)  
   60     try:  
---> 61         return feature_names.index(fx)  
   62     except ValueError as e:  
  
ValueError: 'Cidade_Goiânia' is not in list
```

The above exception was the direct cause of the following exception:

```
ValueError                                                 Traceback (most recent call last)  
Cell In[44], line 23  
  21 # Plot dos PDPs  
  22 fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(16, 12))  
---> 23 PartialDependenceDisplay.from_estimator(  
  24     rf,  
  25     X_bal,  
  26     features=features_pdp,  
  27     grid_resolution=50,  
  28     ax=ax  
  29 )  
 31 plt.suptitle("Gráficos de Dependência Parcial (PDP) - Recontratação", fontsize=16)  
 32 plt.tight_layout(rect=[0, 0, 1, 0.96])
```

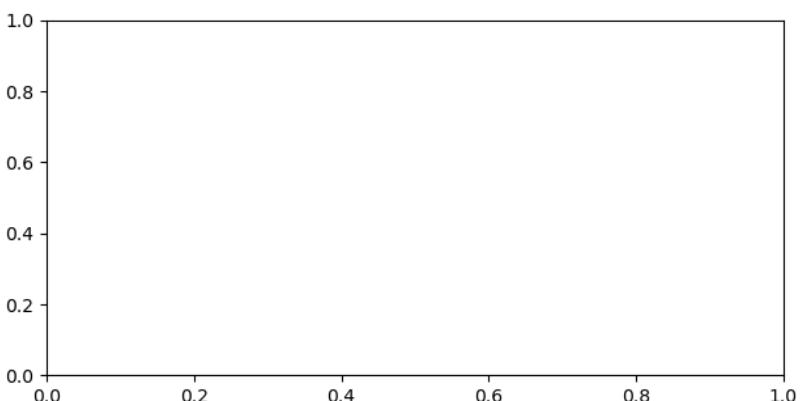
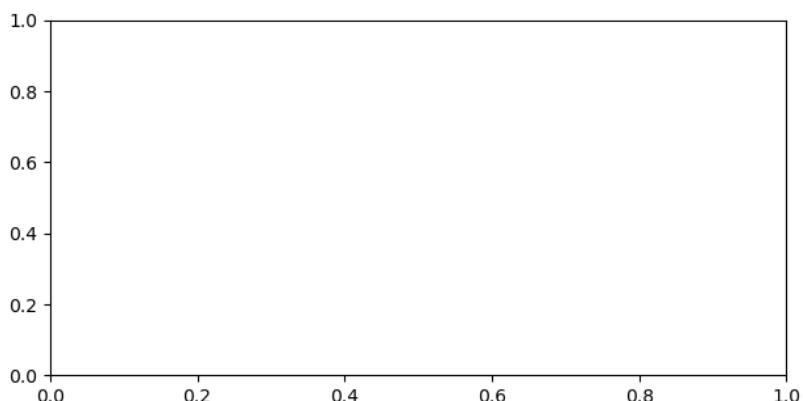
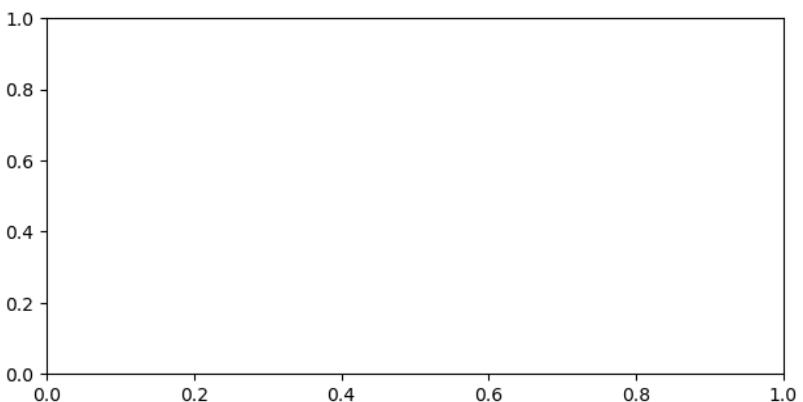
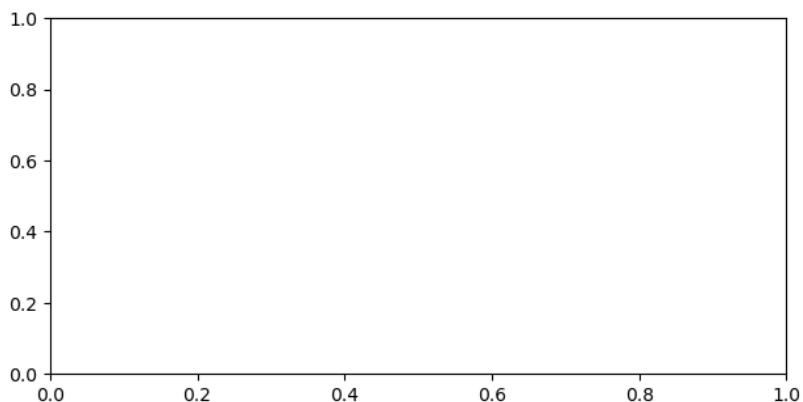
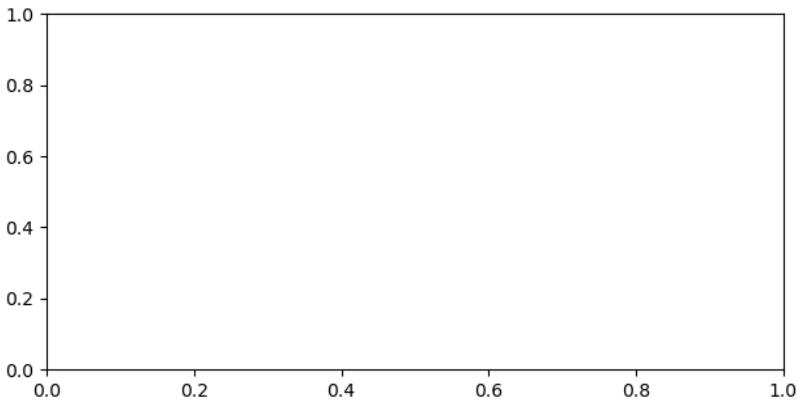
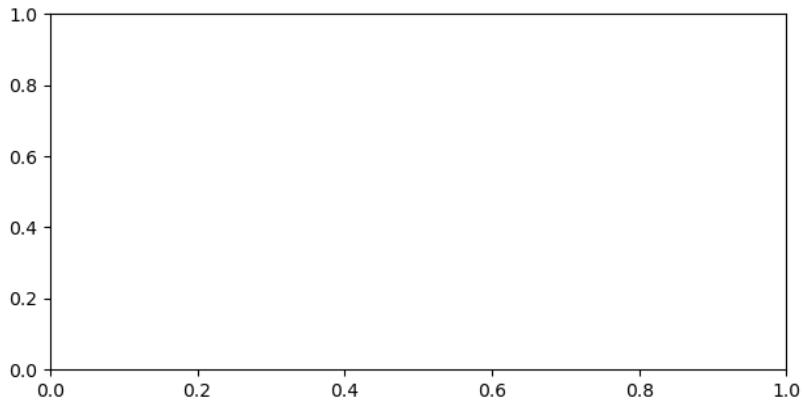
```
File ~\AppData\Local\anaconda3\Lib\site-packages\sklearn\inspection\_plot\partial_dependence.py:563, in PartialDependenceDisplay.from_estimator(cls, estimator, X, features, categorical_features, feature_names, target, response_method, n_cols, grid_resolution, percentiles, method, n_jobs, verbose, line_kw, ice_lines_kw, pd_line_kw, contour_kw, ax, kind, centered, subsample, random_state)  
  561     fxs = (fxs,)  
  562     try:  
---> 563         fxs = tuple(  
  564             _get_feature_index(fx, feature_names=feature_names) for fx in fxs  
  565         )  
  566     except TypeError as e:  
  567         raise ValueError(  
  568             "Each entry in features must be either an int, "  
  569             "a string, or an iterable of size at most 2."  
  570     ) from e
```

```
File ~\AppData\Local\anaconda3\Lib\site-packages\sklearn\inspection\_plot\partial_dependence.py:564, in <genexpr>(.0)
```

```
561     fxs = (fxs,)
562 try:
563     fxs = tuple(
--> 564         _get_feature_index(fx, feature_names=feature_names) for fx in fxs
565     )
566 except TypeError as e:
567     raise ValueError(
568         "Each entry in features must be either an int, "
569         "a string, or an iterable of size at most 2."
570     ) from e

File ~\AppData\Local\anaconda3\Lib\site-packages\sklearn\inspection\_pd_utils.py:63, in _get_feature_index(fx, feature_names)
      61     return feature_names.index(fx)
      62 except ValueError as e:
---> 63     raise ValueError(f"Feature {fx!r} not in feature_names") from e
      64 return fx

ValueError: Feature 'Cidade_Goiânia' not in feature_names
```



```
In [2]: pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (0.1
1.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from imb
alanced-learn) (1.26.4)
Requirement already satisfied: scipy>=1.5.0 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from imba
lanced-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (fr
om imbalanced-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (from imb
alanced-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\f8094564\appdata\local\anaconda3\lib\site-packages (f
rom imbalanced-learn) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

Gráfico de Dependência Parcial (PDP) variáveis consolidadas via Random Forest

```
In [35]: import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OrdinalEncoder
from sklearn.inspection import PartialDependenceDisplay
from imblearn.over_sampling import RandomOverSampler

# 1. Carregar dados
df_contratos = pd.read_csv("contratos_amostra_143.csv", sep=",")
df_flags = pd.read_csv("contratos_1008_com_flags.csv", sep=",")
df_fornecedores = pd.read_csv("amostra_apenas_150_fornecedores.csv", sep=";")

# 2. Padronizar e Limpar CNPJs
for df in [df_contratos, df_flags, df_fornecedores]:
    df["CNPJ"] = df["CNPJ"].astype(str).str.replace(r"\D", "", regex=True)

# 3. Tratar 'Capital Social'
df_fornecedores["Capital Social"] = (
    df_fornecedores["Capital Social"]
    .replace("", "0")
    .fillna("0")
    .astype(str)
    .str.replace("R\$", "", regex=True)
    .str.replace(".", "", regex=False)
    .str.replace(",", ".", regex=False)
```

```
.astype(float)
)

# 4. Criar flag 'recontratado'
contagem_cnpj = df_contratos["CNPJ"].value_counts()
df_contratos["recontratado"] = df_contratos["CNPJ"].map(lambda x: 1 if contagem_cnpj[x] > 1 else 0)

# 5. Tratar 'Valor Global'
df_contratos["Valor Global"] = (
    df_contratos["Valor Global"]
    .astype(str)
    .str.replace("R\\$", "", regex=True)
    .str.replace(".", "", regex=False)
    .str.replace(",", ".", regex=False)
    .astype(float)
)

# 6. Mesclar todos os dados
df = df_contratos.merge(df_flags[["CNPJ", "Flag_CNEP", "Flag_CEIS"]], on="CNPJ", how="left")
df = df.merge(df_fornecedores, on="CNPJ", how="left")

# 7. Preencher valores ausentes
df["Capital Social"] = df["Capital Social"].fillna(0)
df["Valor Global"] = df["Valor Global"].fillna(df["Valor Global"].median())
df["Idade da Empresa"] = df["Idade da Empresa"].fillna(df["Idade da Empresa"].median())
df["Cidade"] = df["Cidade"].fillna("Desconhecida")
df["Estado"] = df["Estado"].fillna("Desconhecido")
df["Porte"] = df["Porte"].fillna("Desconhecido")
df["recontratado"] = df["recontratado"].fillna(0).astype(int)

# 8. Codificar variáveis categóricas com OrdinalEncoder
encoder = OrdinalEncoder()
df[["Cidade", "Estado", "Porte"]] = encoder.fit_transform(df[["Cidade", "Estado", "Porte"]])

# 9. Definir X e y
features = ["Valor Global", "Capital Social", "Idade da Empresa", "Cidade", "Estado", "Porte" ]
target = "recontratado"
X = df[features]
y = df[target]

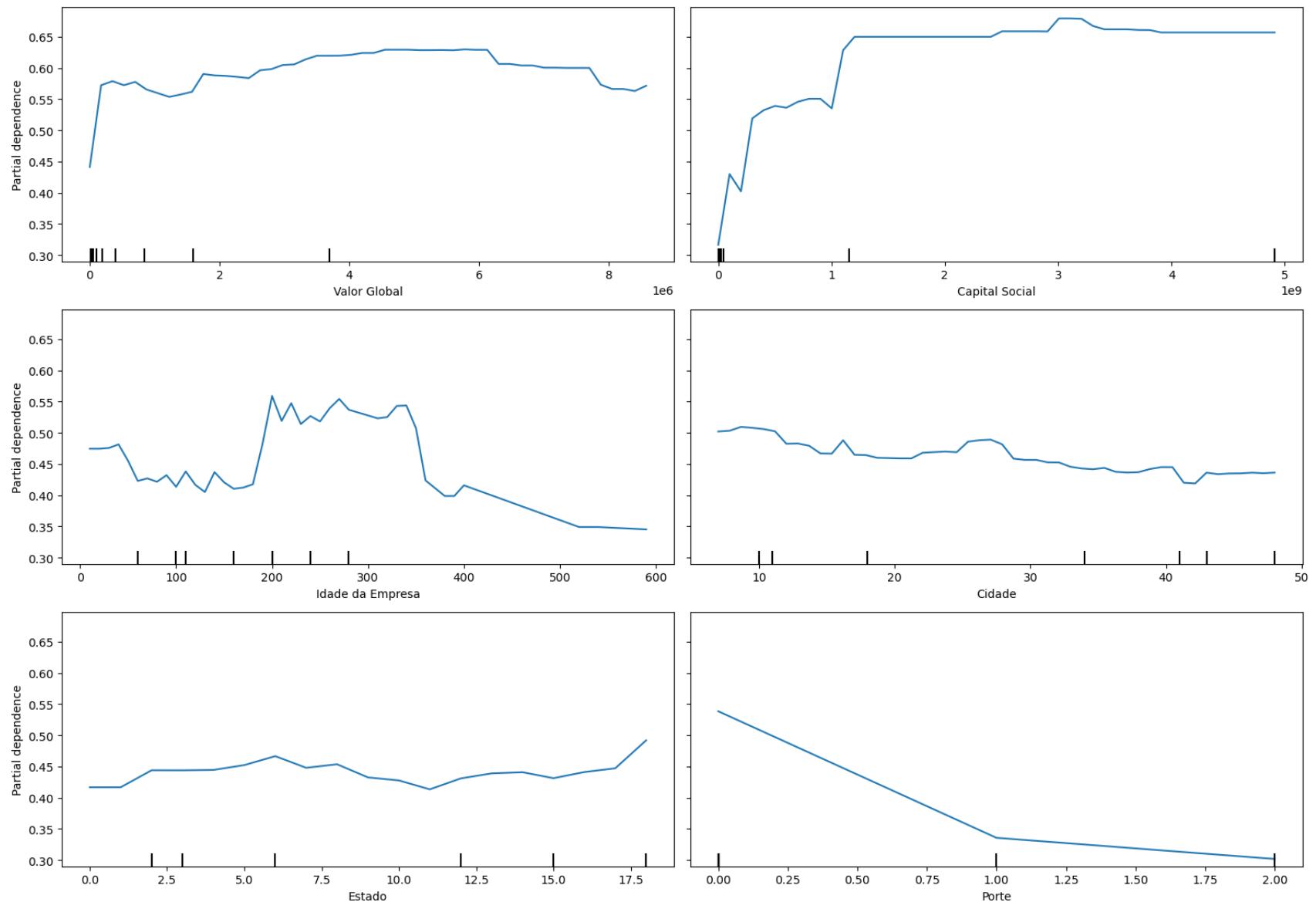
# 10. Balancear a base
ros = RandomOverSampler(random_state=42)
```

```
X_bal, y_bal = ros.fit_resample(X, y)

# 11. Treinar modelo
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_bal, y_bal)

# 12. Gerar gráficos PDP
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(16, 12))
PartialDependenceDisplay.from_estimator(
    rf,
    X_bal,
    features=features,
    grid_resolution=50,
    ax=ax
)
plt.suptitle("Gráficos de Dependência Parcial (PDP) - Recontratação - Random Forest", fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

Gráficos de Dependência Parcial (PDP) - Recontratação - Random Forest



✓ Script Python: Feature Importance + PDP com XGBoost

```
In [58]: import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split
from sklearn.inspection import PartialDependenceDisplay
from imblearn.over_sampling import RandomOverSampler
from xgboost import XGBClassifier
import seaborn as sns

# 1. Ler os arquivos
df_contratos = pd.read_csv("contratos_amostra_143.csv", sep=",")
df_flags = pd.read_csv("contratos_1008_com_flags.csv", sep=",")
df_fornecedores = pd.read_csv("amostra_apenas_150_fornecedores.csv", sep=";")

# 2. Tratar Capital Social
df_fornecedores["Capital Social"] = df_fornecedores["Capital Social"].replace("", "0").fillna("0")
df_fornecedores["Capital Social"] = (
    df_fornecedores["Capital Social"]
    .astype(str)
    .str.replace("R\$", "", regex=True)
    .str.replace(".", "", regex=False)
    .str.replace(",", ".", regex=False)
    .astype(float)
)

# 3. Criar variável de recontratação
df_contratos["CNPJ"] = df_contratos["CNPJ"].astype(str).str.replace(r'\D', '', regex=True)
contagem_cnpj = df_contratos["CNPJ"].value_counts()
df_contratos["recontratado"] = df_contratos["CNPJ"].map(lambda x: 1 if contagem_cnpj[x] > 1 else 0)

# 4. Merge
df_flags["CNPJ"] = df_flags["CNPJ"].astype(str).str.replace(r'\D', '', regex=True)
df_fornecedores["CNPJ"] = df_fornecedores["CNPJ"].astype(str).str.replace(r'\D', '', regex=True)
df = df_contratos.merge(df_flags[["CNPJ", "Flag_CNEP", "Flag_CEIS"]], on="CNPJ", how="left")
df = df.merge(df_fornecedores, on="CNPJ", how="left")

# 5. Tratar Valor Global
df["Valor Global"] = (
    df["Valor Global"]
    .astype(str)
    .str.replace("R\$", "", regex=True)
    .str.replace(".", "", regex=False)
```

```
.str.replace(", ", ".", regex=False)
.astype(float)
)

# 6. Preencher ausentes e codificar
df["Idade da Empresa"] = df["Idade da Empresa"].fillna(df["Idade da Empresa"].median())
df["Cidade"] = df["Cidade"].fillna("Desconhecida")
df["Estado"] = df["Estado"].fillna("Desconhecido")
df["Porte"] = df["Porte"].fillna("Desconhecido")

ordinal_cols = ["Cidade", "Estado", "Porte"]
encoder = OrdinalEncoder()
df[ordinal_cols] = encoder.fit_transform(df[ordinal_cols])

# 7. Definir X e y
features = [
    "Valor Global", "Capital Social", "Idade da Empresa",
    "Cidade", "Estado", "Porte"
]
X = df[features]
y = df["recontratado"]

# 8. Balancear
ros = RandomOverSampler(random_state=42)
X_bal_xgb, y_bal_xgb = ros.fit_resample(X, y)

# 9. Treinar XGBoost
xgb = XGBClassifier(n_estimators=100, use_label_encoder=False, eval_metric="logloss", random_state=42)
xgb.fit(X_bal_xgb, y_bal_xgb)

# 10. Feature Importance
importancias = pd.Series(xgb.feature_importances_, index=X.columns).sort_values(ascending=False)

# Define a cor azul padrão (hexadecimal usado em Seaborn/Matplotlib)
azul_rf = "#1f77b4"

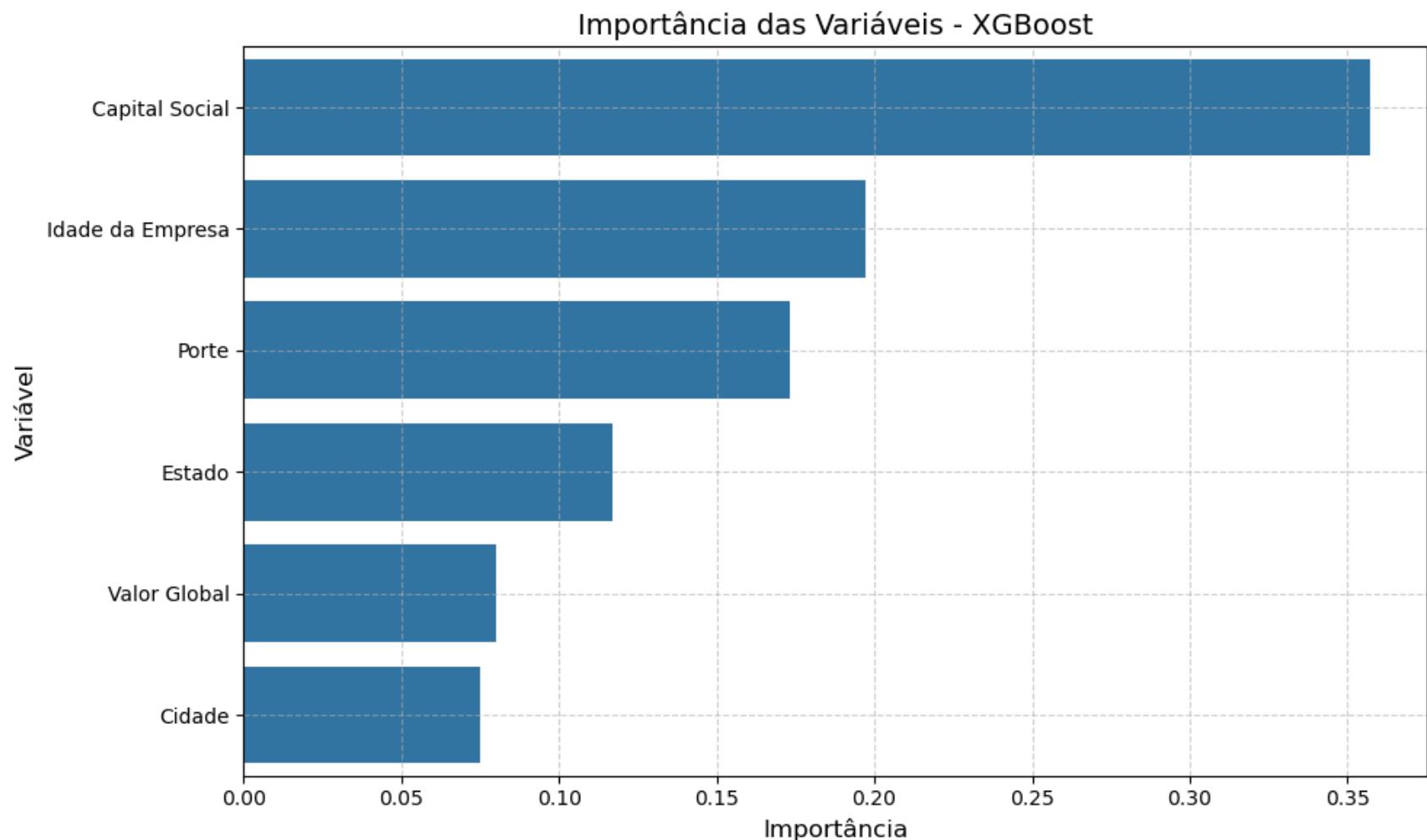
# Cria o gráfico com grid nos dois eixos
plt.figure(figsize=(10, 6))
sns.barplot(x=importancias, y=importancias.index, color=azul_rf)
plt.title("Importância das Variáveis - XGBoost", fontsize=14)
plt.xlabel("Importância", fontsize=12)
plt.ylabel("Variável", fontsize=12)
```

```
plt.grid(True, axis='both', linestyle='--', alpha=0.6) # Grid em X e Y
plt.tight_layout()
plt.show()

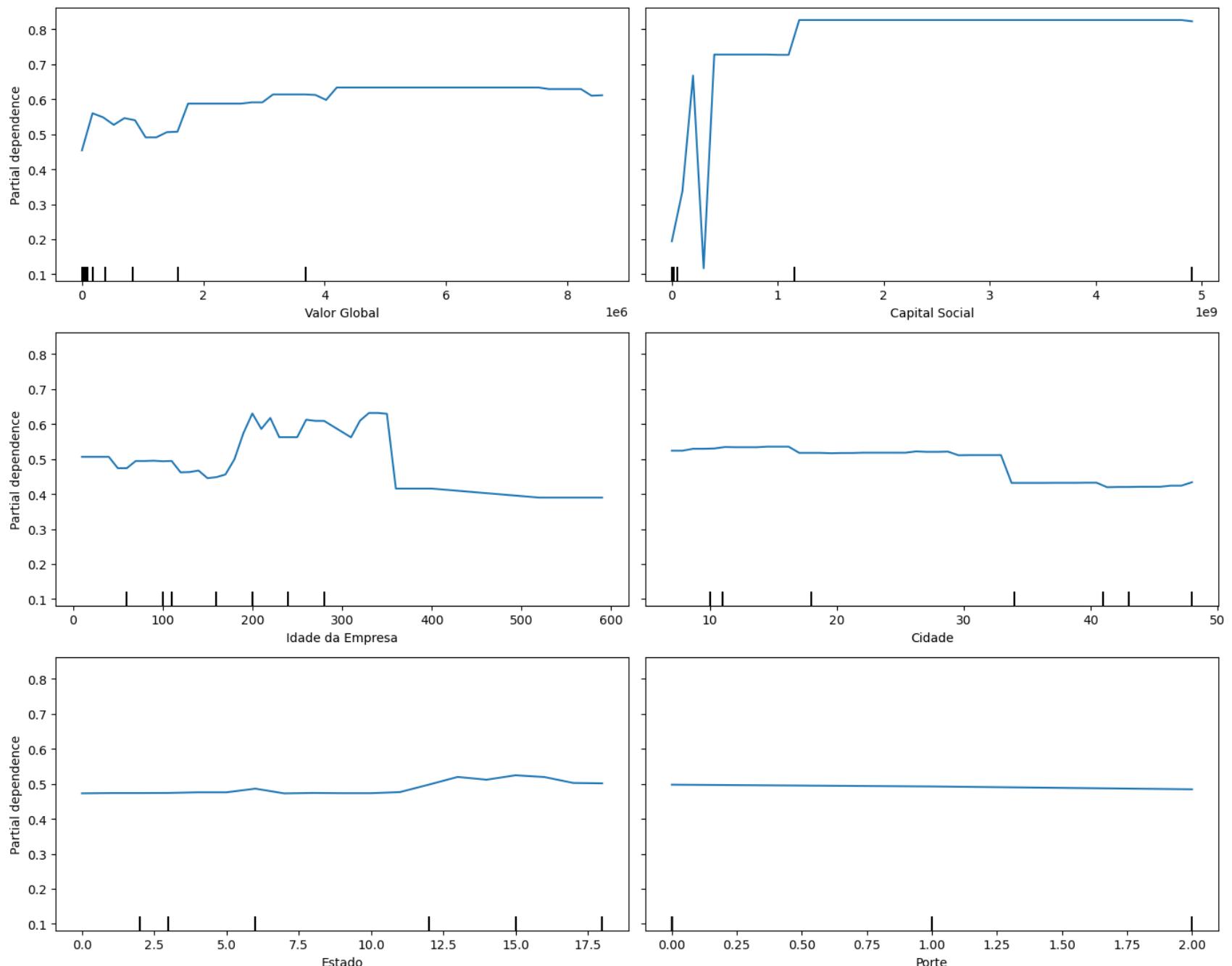
# 11. PDP - Partial Dependence Plot
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(14, 12))
PartialDependenceDisplay.from_estimator(
    xgb,
    X_bal_xgb,
    features=features,
    grid_resolution=50,
    ax=ax
)
plt.suptitle("Gráficos de Dependência Parcial (PDP) - Recontratação - XGBoost", fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

C:\Users\F8094564\AppData\Local\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [20:00:01] WARNING:
C:\actions-runner_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

```
bst.update(dtrain, iteration=i, fobj=obj)
```



Gráficos de Dependência Parcial (PDP) - Recontratação - XGBoost



✓ Importância das Variáveis – XGBoost

```
In [57]: # Extrair importância das variáveis do modelo XGBoost  
importancias_xgb = pd.Series(xgb.feature_importances_, index=X.columns)  
  
# Criar DataFrame ordenado com os nomes legíveis  
df_importancia_xgb = pd.DataFrame({  
    "Variável": importancias_xgb.index,  
    "Importância_XGB": importancias_xgb.values  
}).sort_values(by="Importância_XGB", ascending=False).reset_index(drop=True)  
  
# Exibir a tabela resultante  
print(df_importancia_xgb)
```

	Variável	Importância_XGB
0	Capital Social	0.357310
1	Idade da Empresa	0.197391
2	Porte	0.173253
3	Estado	0.116900
4	Valor Global	0.079931
5	Cidade	0.075215
6	Flag_CEIS	0.000000

✓ Script Python para Random Forest e XGBoost com variáveis dummies para cidades/ estados específicos

```
In [52]: import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import OrdinalEncoder  
from sklearn.model_selection import train_test_split  
from sklearn.inspection import PartialDependenceDisplay  
from imblearn.over_sampling import RandomOverSampler  
from xgboost import XGBClassifier  
  
# 1. Supondo que você já tenha a big table carregada como df  
# df = pd.read_csv("big_table.csv")  
  
# 2. Gerar dummies para Cidade, Estado e Porte  
df_dummies = pd.get_dummies(df, columns=["Cidade", "Estado", "Porte"], prefix=["Cidade", "Estado", "Porte"], drop_fir
```

```
# 3. Selecionar colunas a serem usadas no PDP
features_pdp = [
    "Capital Social",
    "Idade da Empresa",
    "Valor Global",
    "Cidade_Rio de Janeiro", # Altere conforme o nome real existente
    "Porte_Microempresa",
    "Estado_RJ"
]

# 4. Checar se as colunas existem no dataframe
for col in features_pdp:
    if col not in df_dummies.columns:
        raise ValueError(f"Coluna '{col}' não encontrada no DataFrame. Verifique os nomes gerados por get_dummies.")

# 5. Definir X e y
X = df_dummies[features_pdp]
y = df_dummies["recontratado"]

# 6. Balancear a base
ros = RandomOverSampler(random_state=42)
X_bal, y_bal = ros.fit_resample(X, y)

# 7. Treinar modelo XGBoost
xgb = XGBClassifier(n_estimators=100, use_label_encoder=False, eval_metric="logloss", random_state=42)
xgb.fit(X_bal, y_bal)

# 8. Plotar PDP
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(16, 12))
PartialDependenceDisplay.from_estimator(
    xgb,
    X_bal,
    features=features_pdp,
    grid_resolution=50,
    ax=ax
)
plt.suptitle("Gráficos de Dependência Parcial (PDP) - Recontratação (XGBoost)", fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

```
-----  
ValueError  
Cell In[52], line 28  
  26 for col in features_pdp:  
  27     if col not in df_dummies.columns:  
---> 28         raise ValueError(f"Coluna '{col}' não encontrada no DataFrame. Verifique os nomes gerados por get_dum  
mies.")  
  29 # 5. Definir X e y  
 30 X = df_dummies[features_pdp]
```

ValueError: Coluna 'Cidade_Rio de Janeiro' não encontrada no DataFrame. Verifique os nomes gerados por get_dummies.

In []:

In []:

✓ H12: Modalidade de contratação influencia a recontratação

→ Técnica sugerida: Teste de Qui-Quadrado (duas variáveis categóricas)

Hipótese inviabilizada, pois a variável "Modalidade" não está disponível na base de dados

```
In [6]: import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
from scipy.stats import chi2_contingency  
  
# 1. Carregar a base consolidada  
df = pd.read_csv("big_table_consolidada.csv")  
  
# 2. Preencher e Limpar dados  
df["Modalidade"] = df["Modalidade"].fillna("Desconhecida")  
df["recontratado"] = df["recontratado"].fillna(0).astype(int)  
  
# 3. Criar tabela de contingência  
contingencia = pd.crosstab(df["Modalidade"], df["recontratado"])  
  
# 4. Executar o teste qui-quadrado  
chi2, p, dof, expected = chi2_contingency(contingencia)
```

```
# 5. Mostrar resultados
print("== H12: Modalidade vs Recontratação ==")
print("Qui-quadrado:", round(chi2, 2))
print("p-valor:", round(p, 4))
print("Graus de liberdade:", dof)
print("Existe associação significativa?" , "Sim" if p < 0.05 else "Não")

# 6. Plotagem
contingencia_norm = contingencia.div(contingencia.sum(1), axis=0)
contingencia_norm.plot(kind='bar', stacked=True, figsize=(10,6), colormap='viridis')
plt.title("Distribuição da Recontratação por Modalidade de Contrato")
plt.xlabel("Modalidade")
plt.ylabel("Proporção")
plt.legend(title="Recontratado")
plt.tight_layout()
plt.show()
```

```
-----  
KeyError Traceback (most recent call last)  
File ~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.get_loc(self, key)  
3804     try:  
-> 3805         return self._engine.get_loc(casted_key)  
3806     except KeyError as err:  
  
File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()  
  
File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()  
  
File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashTable.get_item()  
  
File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashTable.get_item()  
  
KeyError: 'Modalidade'  
  
The above exception was the direct cause of the following exception:
```

```
KeyError Traceback (most recent call last)  
Cell In[6], line 10  
    7 df = pd.read_csv("big_table_consolidada.csv")  
    9 # 2. Preencher e limpar dados  
---> 10 df["Modalidade"] = df["Modalidade"].fillna("Desconhecida")  
    11 df["recontratado"] = df["recontratado"].fillna(0).astype(int)  
    13 # 3. Criar tabela de contingência  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__getitem__(self, key)  
4100     if self.columns.nlevels > 1:  
4101         return self._getitem_multilevel(key)  
-> 4102     indexer = self.columns.get_loc(key)  
4103     if is_integer(indexer):  
4104         indexer = [indexer]  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.get_loc(self, key)  
3807     if isinstance(casted_key, slice) or (  
3808         isinstance(casted_key, abc.Iterable)  
3809         and any(isinstance(x, slice) for x in casted_key)  
3810     ):  
3811         raise InvalidIndexError(key)  
-> 3812     raise KeyError(key) from err  
3813 except TypeError:
```

```
3814     # If we have a listlike key, _check_indexing_error will raise
3815     # InvalidIndexError. Otherwise we fall through and re-raise
3816     # the TypeError.
3817     self._check_indexing_error(key)
```

KeyError: 'Modalidade'

✓ H14: Diversidade de ministérios contratantes influencia recontratação

→ Técnica sugerida: Teste de Mann-Whitney U (variável numérica × binária)

```
In [7]: from scipy.stats import mannwhitneyu

# 1. Criar variável de diversidade: número de ministérios distintos por CNPJ
diversidade = df.groupby("CNPJ_clean")["ministerio"].nunique().reset_index()
diversidade.columns = ["CNPJ_clean", "num_ministerios"]

# 2. Trazer a info para o DataFrame principal
df = df.merge(diversidade, on="CNPJ_clean", how="left")

# 3. Agrupar por recontratado
grupo_0 = df[df["recontratado"] == 0]["num_ministerios"].dropna()
grupo_1 = df[df["recontratado"] == 1]["num_ministerios"].dropna()

# 4. Teste de Mann-Whitney
stat, p = mannwhitneyu(grupo_0, grupo_1, alternative='two-sided')

print("\n==== H14: Diversidade de Ministérios vs Recontratação ===")
print("Estatística U:", round(stat, 2))
print("p-valor:", round(p, 4))
print("Existe diferença significativa entre os grupos?", "Sim" if p < 0.05 else "Não")

# 5. Boxplot para visualização
sns.boxplot(data=df, x="recontratado", y="num_ministerios")
plt.xticks([0, 1], ['Não Recontratado', 'Recontratado'])
plt.title("Diversidade de Ministérios vs Recontratação")
plt.xlabel("Recontratado")
plt.ylabel("Número de Ministérios Contratantes")
plt.tight_layout()
plt.show()
```

== H14: Diversidade de Ministérios vs Recontratação ==

Estatística U: 10591.5

p-valor: 0.0

Existe diferença significativa entre os grupos? Sim



FIM DAS ANÁLISES DE MACHINE LEARNING E DO PROJETO

In []:

In []:

Exploração de dados com as bases ainda não tratadas

1. Diagnóstico de Qualidade e Estatística Descritiva Final

In [78]:

```
# 🎨 Importações
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Estilo mais moderno
plt.style.use('seaborn-v0_8-whitegrid')
sns.set_theme(style="whitegrid")

# 🚀 1.1. Carregar o dataset final
df = pd.read_csv('contratos_1008_com_flags.csv', encoding='utf-8')

# 🚀 1.2. Diagnóstico de Missing
print("🔍 Diagnóstico de Missing Values")
missing_df = df.isnull().sum()
percent_missing = (missing_df / len(df)) * 100
diagnostic = pd.DataFrame({'Missing Count': missing_df, 'Missing (%)': percent_missing})
display(diagnostic)

# 🚀 1.3. Tipos de dados
print("\n📊 Tipos de Dados")
print(df.dtypes)

# 🚀 1.4. Estatísticas Descritivas Numéricas
print("\n📈 Estatísticas Descritivas - Variáveis Numéricas")
display(df.describe().T.style.background_gradient(cmap="Blues").format(precision=2))
```

🔍 Diagnóstico de Missing Values

	Missing Count	Missing (%)
Órgão	0	0.000000
Unidade Gestora	0	0.000000
Número Contrato	0	0.000000
Fornecedor	0	0.000000
Vig. Início	0	0.000000
...
ministerio	0	0.000000
CNPJ	0	0.000000
Nome Fornecedor	0	0.000000
Flag_CNEP	19	1.746324
Flag_CEIS	19	1.746324

14 rows × 2 columns

Tipos de Dados

```

Órgão          object
Unidade Gestora    object
Número Contrato    object
Fornecedor        object
Vig. Início       object
...
ministerio      object
CNPJ            int64
Nome Fornecedor   object
Flag_CNEP        float64
Flag_CEIS        float64
Length: 14, dtype: object

```

Estatísticas Descritivas - Variáveis Numéricas

	count	mean	std	min	25%	50%	75%
Núm. Parcelas	1088.00	7.08	13.27	1.00	1.00	1.00	12.00
CNPJ	1088.00	22126131150876.12	23391448870036.41	242160.00	4602789000101.00	10533767000135.50	33683111000107.00
Flag_CNEP	1069.00	0.00	0.00	0.00	0.00	0.00	0.00
Flag_CEIS	1069.00	0.03	0.18	0.00	0.00	0.00	0.00

🔍 2. Análise Exploratória de Dados (EDA)

```
In [79]: # 2.1. Ajuste de Datas e Criação de Coluna 'Ano Início'
df['Vig. Início'] = pd.to_datetime(df['Vig. Início'], errors='coerce', dayfirst=True)
df['Ano Início'] = df['Vig. Início'].dt.year

# 2.2. Número de contratos por ano
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Ano Início', palette='crest')
plt.title('📊 Número de Contratos por Ano')
plt.xlabel('Ano de Início')
plt.ylabel('Quantidade de Contratos')
plt.show()

# 2.3. Valor total dos contratos por ano
df_ano_valor = df.groupby('Ano Início')['Valor Global'].sum().reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(data=df_ano_valor, x='Ano Início', y='Valor Global', palette='Blues')
plt.title('💵 Valor Total dos Contratos por Ano')
plt.xlabel('Ano de Início')
plt.ylabel('Valor Total (R$)')
plt.xticks(rotation=45)
plt.show()

# 2.4. Top 15 Ministérios por Número de Contratos
top_ministerios = df['ministerio'].value_counts().head(15).reset_index()
top_ministerios.columns = ['Ministério', 'Número de Contratos']

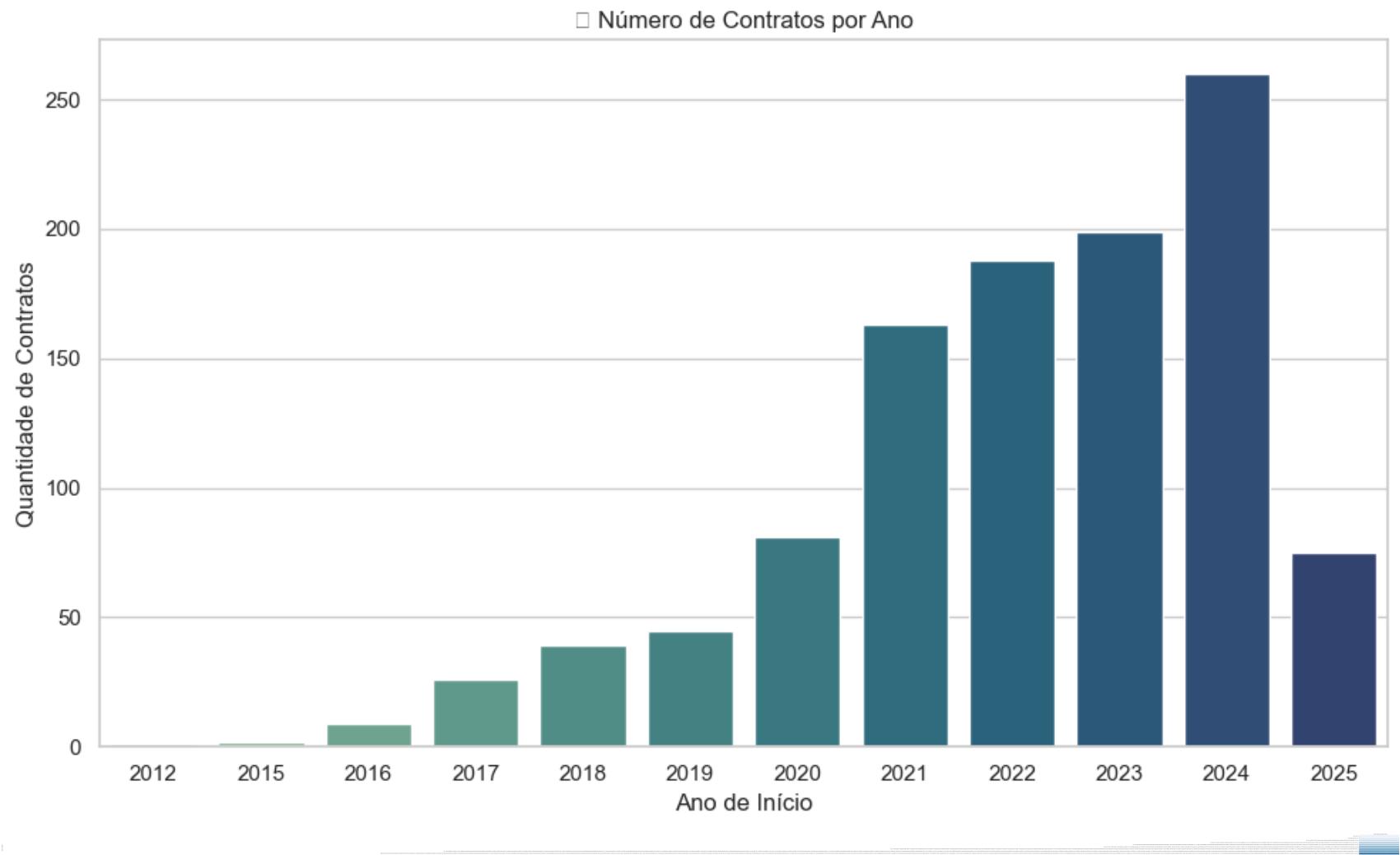
plt.figure(figsize=(12, 8))
```

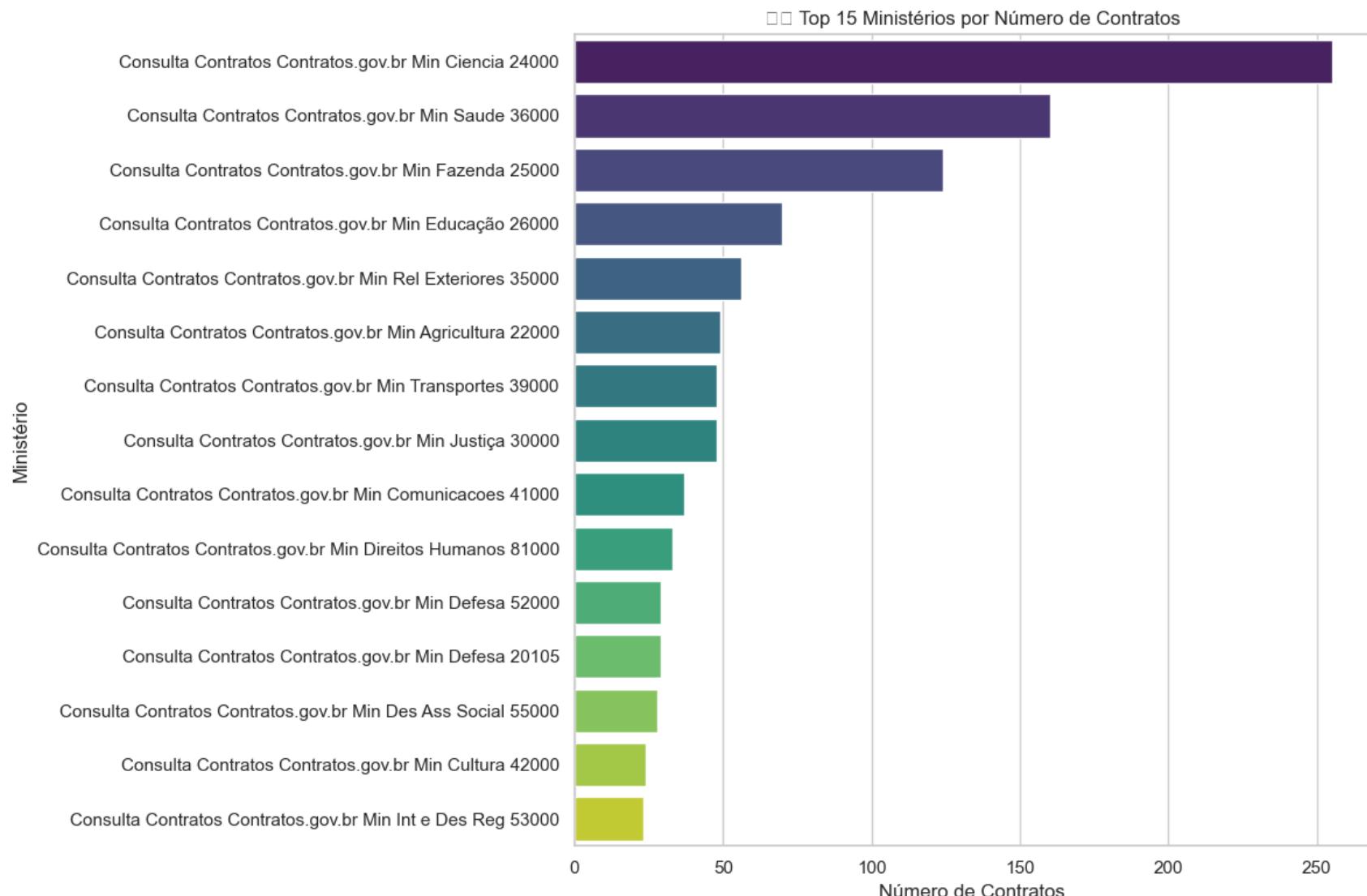
```
sns.barplot(data=top_ministerios, x='Número de Contratos', y='Ministério', palette='viridis')
plt.title('🏛️ Top 15 Ministérios por Número de Contratos')
plt.xlabel('Número de Contratos')
plt.ylabel('Ministério')
plt.show()

# 2.5. Valor total por ministério
valor_ministerios = df.groupby('ministerio')['Valor Global'].sum().sort_values(ascending=False).head(15).reset_index()

plt.figure(figsize=(12, 8))
sns.barplot(data=valor_ministerios, x='Valor Global', y='ministerio', palette='mako')
plt.title('🏛️ Top 15 Ministérios por Valor Total Contratado')
plt.xlabel('Valor Total (R$)')
plt.ylabel('Ministério')
plt.show()

# 2.6. Distribuição de Valor Global (Boxplot)
plt.figure(figsize=(12, 6))
sns.boxplot(x='Valor Global', data=df, palette='Set2')
plt.title('📦 Distribuição dos Valores Globais dos Contratos')
plt.xlabel('Valor Global (R$)')
plt.xscale('log') # Log scale para valores altos
plt.show()
```





```
-----  
TypeError Traceback (most recent call last)  
Cell In[79], line 39  
      36 valor_ministerios = df.groupby('ministerio')['Valor Global'].sum().sort_values(ascending=False).head(15).reset_index()  
      38 plt.figure(figsize=(12, 8))  
--> 39 sns.barplot(data=valor_ministerios, x='Valor Global', y='ministerio', palette='mako')  
    40 plt.title('Top 15 Ministérios por Valor Total Contratado')  
    41 plt.xlabel('Valor Total (R$)')  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\seaborn\categorical.py:2755, in barplot(data, x, y, hue, order, hue_order, estimator, errorbar, n_boot, units, seed, orient, color, palette, saturation, width, errcolor, errwidth, capsize, dodge, ci, ax, **kwargs)  
 2752 if estimator is len:  
 2753     estimator = "size"  
-> 2755 plotter = _BarPlotter(x, y, hue, data, order, hue_order,  
 2756                           estimator, errorbar, n_boot, units, seed,  
 2757                           orient, color, palette, saturation,  
 2758                           width, errcolor, errwidth, capsize, dodge)  
 2760 if ax is None:  
 2761     ax = plt.gca()  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\seaborn\categorical.py:1530, in _BarPlotter.__init__(self, x, y, hue, data, order, hue_order, estimator, errorbar, n_boot, units, seed, orient, color, palette, saturation, width, errcolor, errwidth, capsize, dodge)  
 1525 def __init__(self, x, y, hue, data, order, hue_order,  
 1526                 estimator, errorbar, n_boot, units, seed,  
 1527                 orient, color, palette, saturation, width,  
 1528                 errcolor, errwidth, capsize, dodge):  
 1529     """Initialize the plotter."""  
-> 1530     self._establish_variables(x, y, hue, data, orient,  
 1531                               order, hue_order, units)  
 1532     self._establish_colors(color, palette, saturation)  
 1533     self._estimate_statistic(estimator, errorbar, n_boot, seed)  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\seaborn\categorical.py:544, in _CategoricalPlotter._establish_variables(self, x, y, hue, data, orient, order, hue_order, units)  
 541         raise ValueError(err)  
 543 # Figure out the plotting orientation  
--> 544 orient = infer_orient(  
 545     x, y, orient, require_numeric=self.require_numeric  
 546 )
```

```
548 # Option 2a:  
549 # We are plotting a single set of data  
550 # -----  
551 if x is None or y is None:  
552  
553     # Determine where the data are  
  
File ~\AppData\Local\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1632, in infer_orient(x, y, orient, require_nume  
ric)  
    1630 elif require_numeric and "numeric" not in (x_type, y_type):  
    1631     err = "Neither the `x` nor `y` variable appears to be numeric."  
-> 1632     raise TypeError(err)  
1634 else:  
1635     return "v"  
  
TypeError: Neither the `x` nor `y` variable appears to be numeric.  
<Figure size 1200x800 with 0 Axes>
```

In []:

In []:

3. Verificar Estrutura e Qualidade dos Dados

```
In [11]: # Informações básicas dos datasets  
print(df_contratos.info())  
print(df_fornecedores.info())  
  
# Verificação de valores nulos  
print("Valores nulos em contratos:")  
print(df_contratos.isnull().sum())  
  
print("\nValores nulos em fornecedores:")  
print(df_fornecedores.isnull().sum())  
  
# Verificação de duplicatas  
print(f"Duplicatas em contratos: {df_contratos.duplicated().sum()}")  
print(f"Duplicatas em fornecedores: {df_fornecedores.duplicated().sum()}")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1088 entries, 0 to 1087
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   Órgão              1088 non-null   object  
 1   Unidade Gestora    1088 non-null   object  
 2   Número Contrato    1088 non-null   object  
 3   Fornecedor         1088 non-null   object  
 4   Vig. Início        1088 non-null   object  
 5   Vig. Fim            1088 non-null   object  
 6   Valor Global        1088 non-null   object  
 7   Núm. Parcelas      1088 non-null   int64  
 8   Valor Parcela       1088 non-null   object  
 9   ministerio          1088 non-null   object  
 10  CNPJ               1088 non-null   object  
 11  Nome Fornecedor    1088 non-null   object  
dtypes: int64(1), object(11)
memory usage: 102.1+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 479 entries, 0 to 478
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   CNPJ              479 non-null   int64  
 1   Razão Social       150 non-null   object  
 2   Nome Fantasia      109 non-null   object  
 3   Natureza Jurídica 150 non-null   object  
 4   Capital Social     147 non-null   float64 
 5   Data de Abertura   150 non-null   object  
 6   Idade da Empresa    150 non-null   float64 
 7   Cidade             150 non-null   object  
 8   Estado              150 non-null   object  
 9   CEP                 150 non-null   float64 
 10  Porte               150 non-null   object  
 11  Atividade Principal 150 non-null   object  
 12  Flag_CNEP           479 non-null   int64  
 13  Flag_CEIS           479 non-null   int64  
dtypes: float64(3), int64(3), object(8)
memory usage: 52.5+ KB
None
```

Valores nulos em contratos:

```
Órgão          0
Unidade Gestora 0
Número Contrato 0
Fornecedor     0
Vig. Início    0
...
Núm. Parcelas 0
Valor Parcela  0
ministerio     0
CNPJ           0
Nome Fornecedor 0
Length: 12, dtype: int64
```

Valores nulos em fornecedores:

```
CNPJ          0
Razão Social   329
Nome Fantasia   370
Natureza Jurídica 329
Capital Social  332
...
CEP            329
Porte          329
Atividade Principal 329
Flag_CNEP       0
Flag_CEIS       0
Length: 14, dtype: int64
Duplicatas em contratos: 1
Duplicatas em fornecedores: 0
```

4. Resumo Estatístico

```
In [12]: # Resumo estatístico de variáveis numéricas
print(df_fornecedores.describe())

# Verificar tipos de variáveis
print(df_fornecedores.dtypes)
```

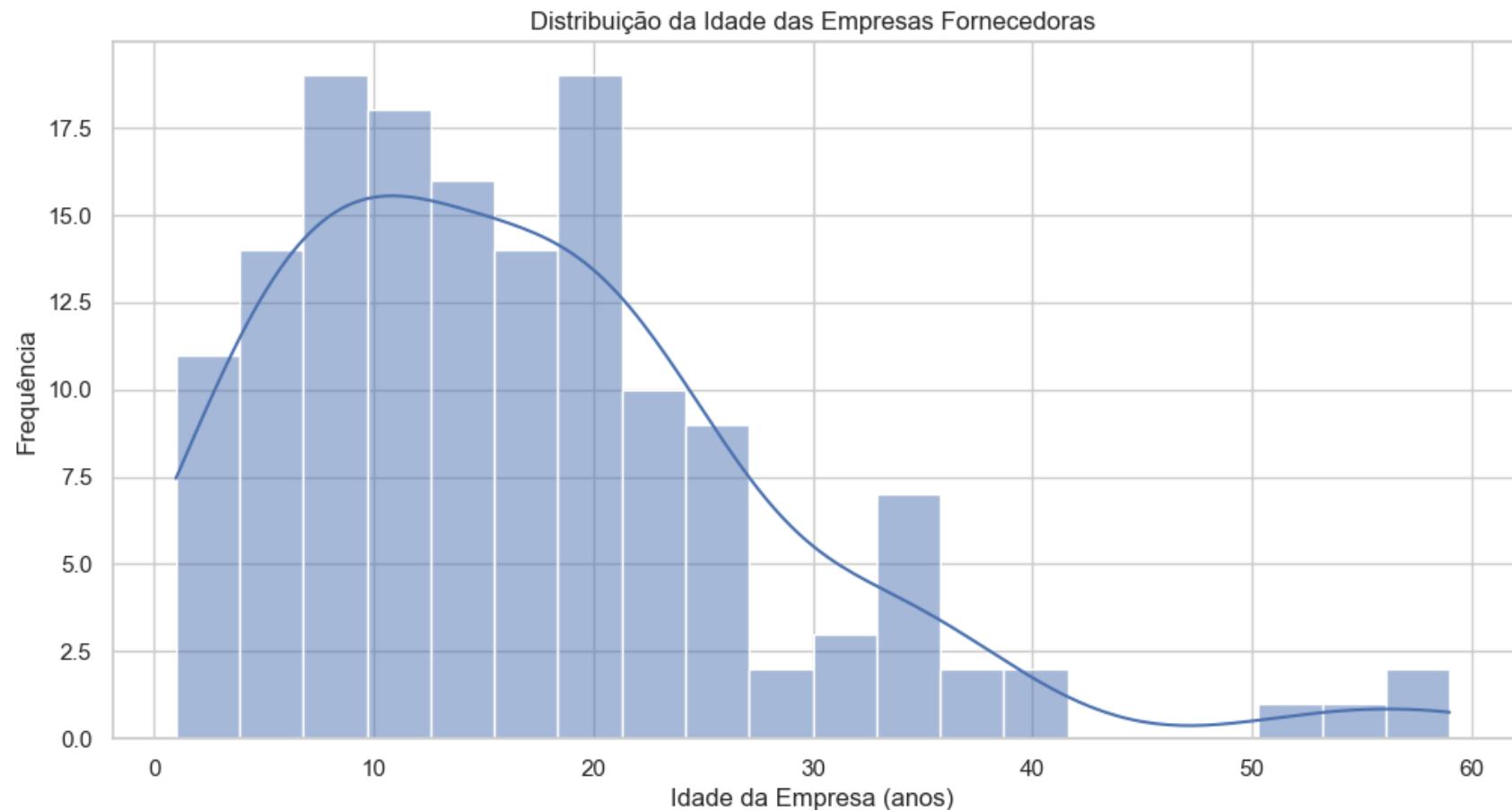
```
CNPJ      Capital Social  Idade da Empresa      CEP      Flag_CNEP  Flag_CEIS
count  4.790000e+02  1.470000e+02  150.000000  1.500000e+02  479.0  479.000000
mean   2.241903e+13  6.864827e+08  16.720000  4.797542e+07  0.0   0.050104
std    2.168341e+13  5.617092e+09  11.373536  2.904211e+07  0.0   0.218388
min    1.131100e+11  2.000000e+03  1.000000  1.045000e+06  0.0   0.000000
25%   5.533018e+12  1.000000e+05  8.250000  2.192288e+07  0.0   0.000000
50%   1.301124e+13  4.000000e+05  15.000000  5.178001e+07  0.0   0.000000
75%   3.410385e+13  4.089650e+06  22.000000  7.071402e+07  0.0   0.000000
max   9.431692e+13  6.007142e+10  59.000000  9.705080e+07  0.0   1.000000
CNPJ           int64
Razão Social    object
Nome Fantasia    object
Natureza Jurídica  object
Capital Social   float64
...
CEP            float64
Porte          object
Atividade Principal  object
Flag_CNEP       int64
Flag_CEIS       int64
Length: 14, dtype: object
```

5. Análise Exploratória Inicial (EDA) - Fornecedores

5.1. Distribuição da Idade das Empresas

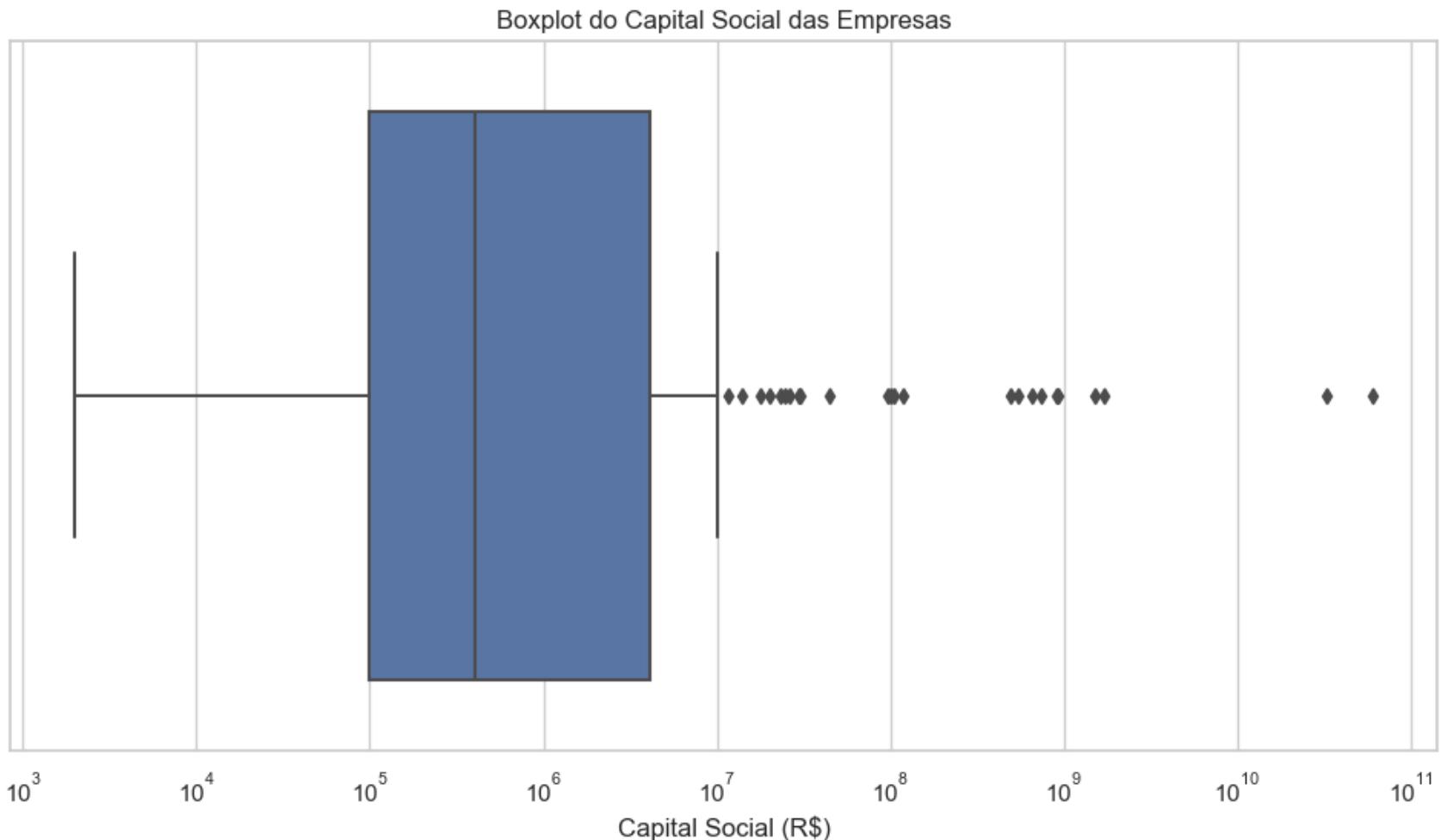
```
In [15]: # Histograma da Idade das Empresas
sns.histplot(df_fornecedores['Idade da Empresa'].dropna(), bins=20, kde=True)
plt.title('Distribuição da Idade das Empresas Fornecedoras')
plt.xlabel('Idade da Empresa (anos)')
plt.ylabel('Frequência')
plt.show()
```

C:\Users\F8094564\AppData\Local\anaconda3\Lib\site-packages\seaborn_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
with pd.option_context('mode.use_inf_as_na', True):



5.2. Capital Social

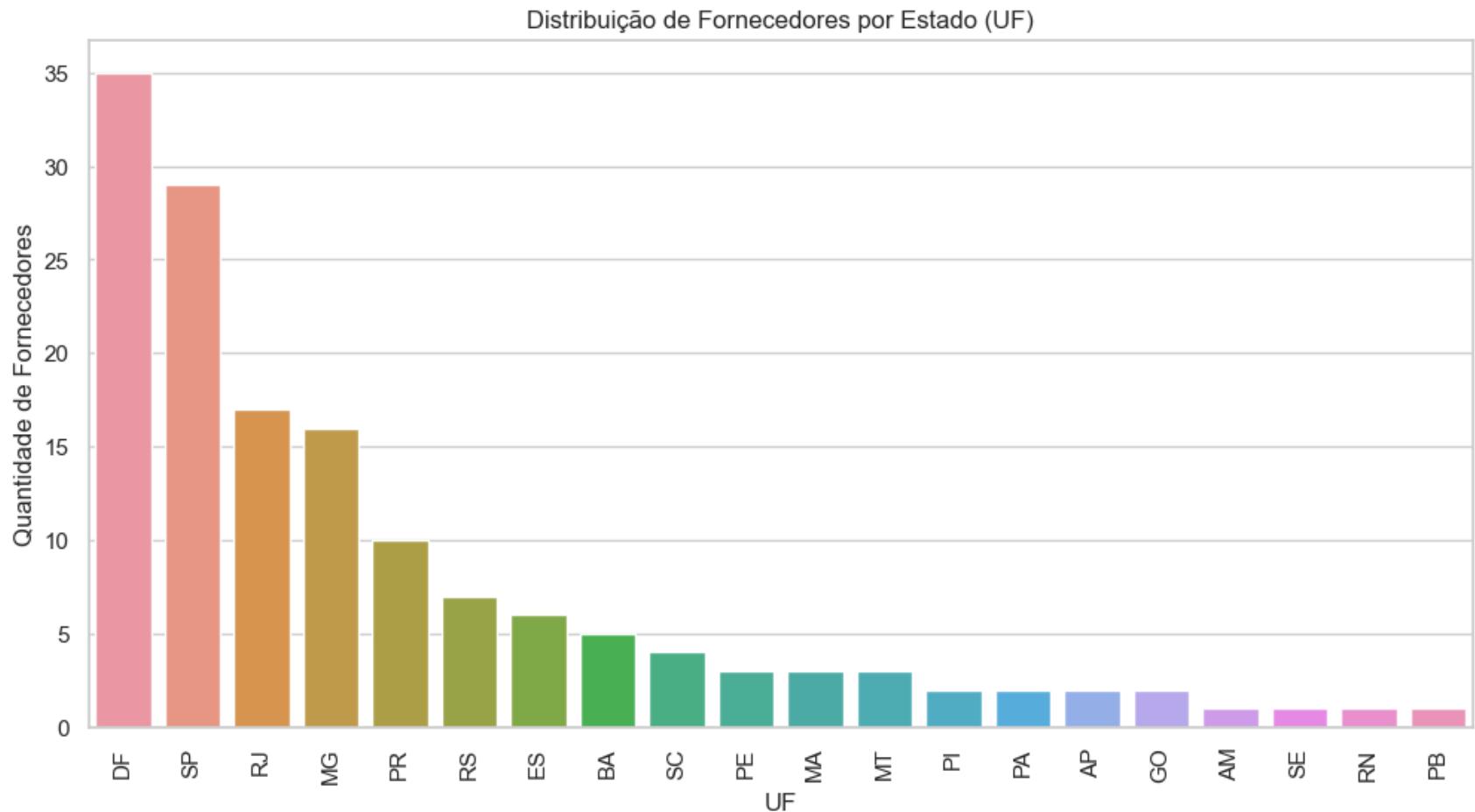
```
In [16]: # Boxplot Capital Social (remoção de outliers para visualização)
sns.boxplot(x=df_fornecedores['Capital Social'])
plt.title('Boxplot do Capital Social das Empresas')
plt.xlabel('Capital Social (R$)')
plt.xscale('log') # Log para melhor visualização
plt.show()
```



5.3. Distribuição Geográfica

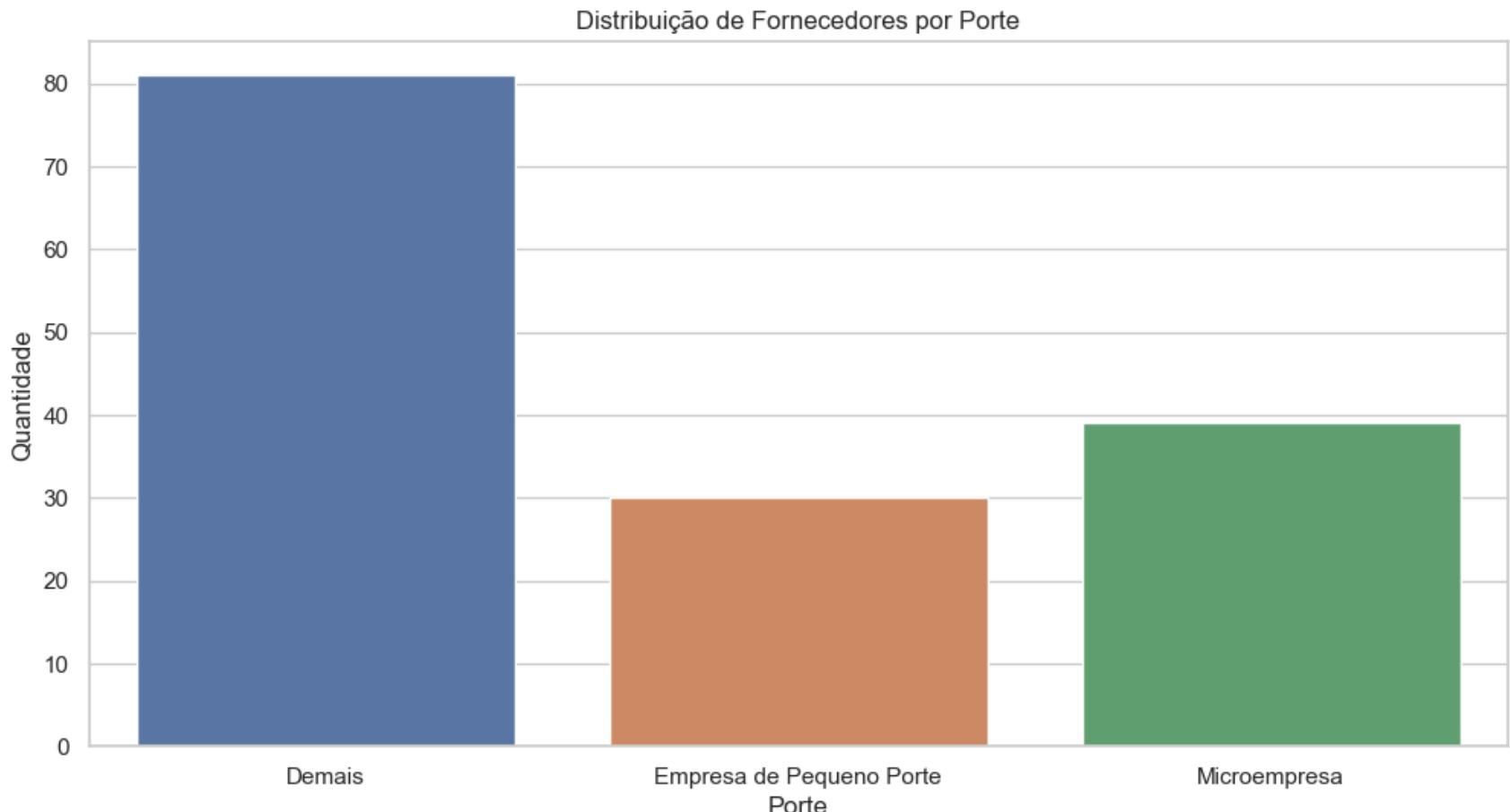
```
In [19]: # Contagem por Estado (UF)
uf_counts = df_fornecedores['Estado'].value_counts()

sns.barplot(x=uf_counts.index, y=uf_counts.values)
plt.xticks(rotation=90)
plt.title('Distribuição de Fornecedores por Estado (UF)')
plt.xlabel('UF')
plt.ylabel('Quantidade de Fornecedores')
plt.show()
```



5.4. Porte das Empresas

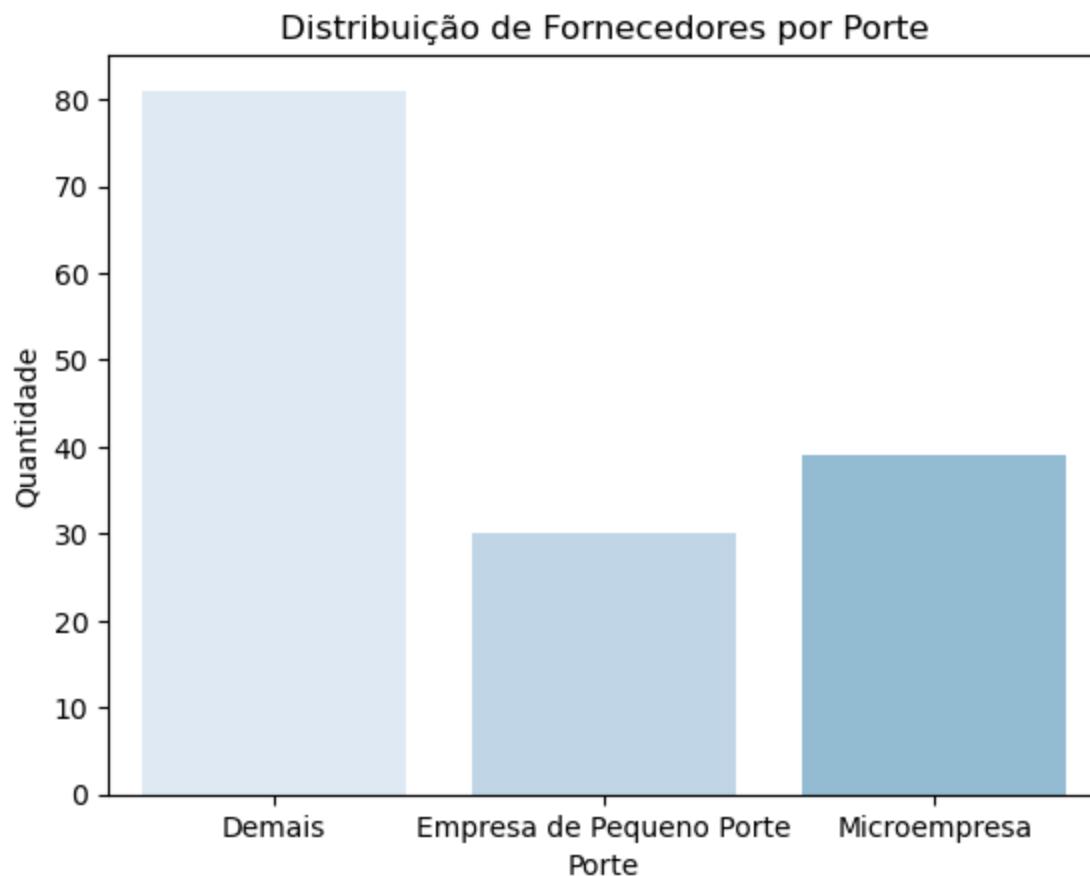
```
In [20]: # Contagem por Porte
sns.countplot(data=df_fornecedores, x='Porte')
plt.title('Distribuição de Fornecedores por Porte')
plt.xlabel('Porte')
plt.ylabel('Quantidade')
plt.show()
```



```
In [62]: import seaborn as sns
import matplotlib.pyplot as plt

# Configurando a paleta de cores para tons de azul
sns.set_palette("Blues")

# Criando o gráfico de barras
sns.countplot(data=df_fornecedores, x='Porte')
plt.title('Distribuição de Fornecedores por Porte')
plt.xlabel('Porte')
plt.ylabel('Quantidade')
plt.show()
```



5.5. Flag de Empresas Punidas (CNEP e CEIS)

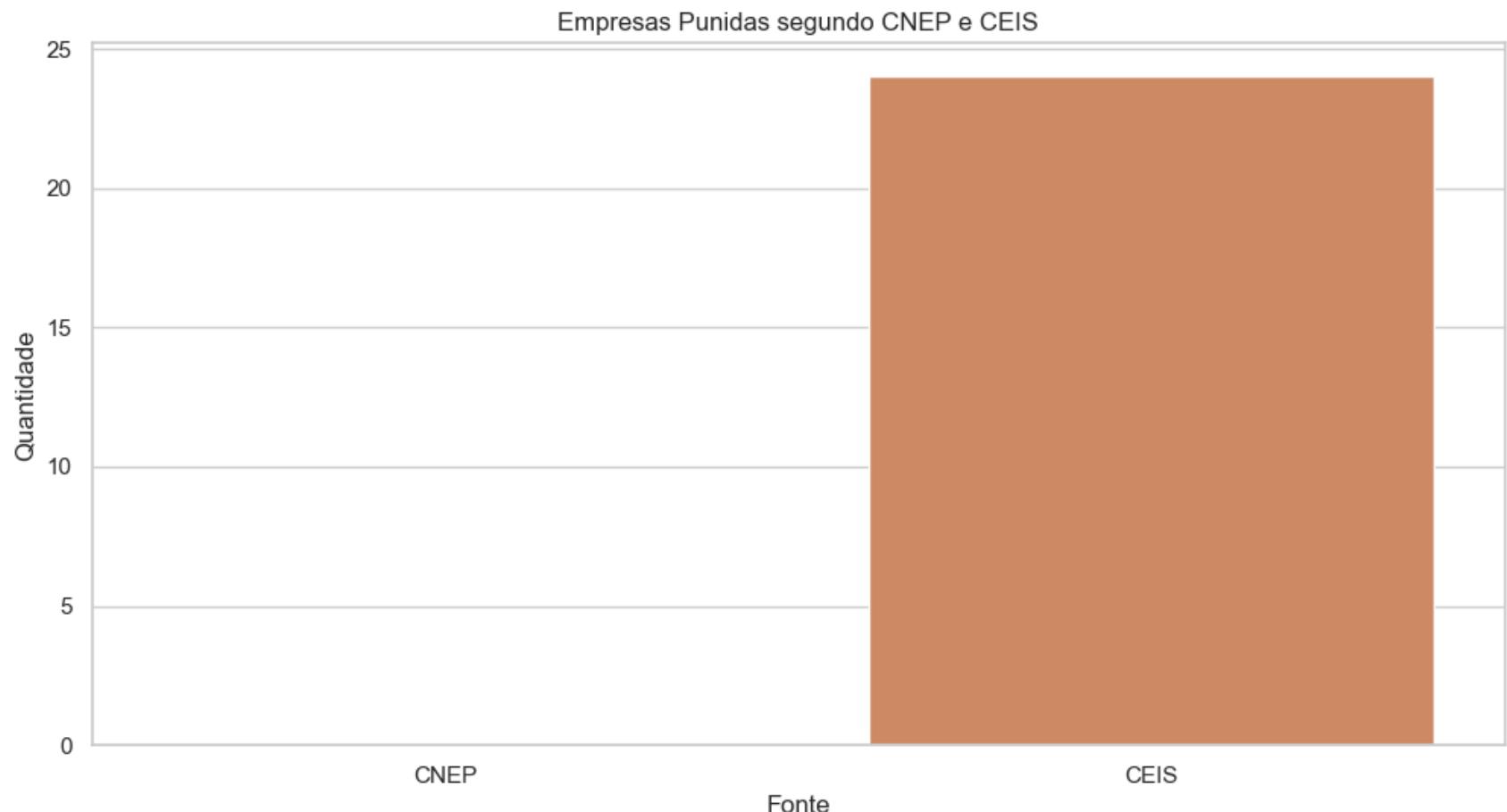
```
In [21]: # Empresas Punidas
punidas_cnep = df_fornecedores['Flag_CNEP'].sum()
punidas_ceis = df_fornecedores['Flag_CEIS'].sum()

print(f"Número de Empresas Punidas (CNEP): {punidas_cnep}")
print(f"Número de Empresas Punidas (CEIS): {punidas_ceis}")

# Gráfico de Barras
punicoes = pd.DataFrame({
    'Fonte': ['CNEP', 'CEIS'],
    'Quantidade': [punidas_cnep, punidas_ceis]
})
```

```
sns.barplot(x='Fonte', y='Quantidade', data=punicoes)
plt.title('Empresas Punidas segundo CNEP e CEIS')
plt.show()
```

Número de Empresas Punidas (CNEP): 0
Número de Empresas Punidas (CEIS): 24



FIM DA Exploração de dados com as bases ainda não tratadas

In []:

In []:

In []: