Análise de dados: Investidores do Tesouro Direto

Neste projeto serão analisados os dados de investidores (pesssoas físicas) do tesouro direto que aderiram ao programa a partir de seu lançamento em janeiro de 2002, com base nos dados (.csv) acessíveis no portal de Dados Abertos disponibilizados pelo Tesouro Nacional Transparente.

O Tesouro Direto é um programa do Tesouro Nacional desenvolvido em parceria com a B3 (antiga BM&F Bovespa) para venda de títulos públicos federais para pessoas físicas pela internet, com o objetivo de captar recursos e financiar as dívidas públicas, permite fazer aplicações com valores muito baixos e oferece liquidez diária.

Demanda da análise

- Analisar a evolução de aderências ao programa e elaborar métricas
- Identificar os perfis e elaborar métricas dos investidores

Indagações a serem respondidas pela análise exploratória dos dados

1. Qual o total de novos investidores por ano (2002 a 2021)?
2. Qual o total de investidores por estado (01/2002 a 07/2022)?
3. Qual o total de investidores em um ranking de 10 cidades (01/2002 a 07/2022)?
4. Qual a quantidade de investidores que operaram nos últimos 12 meses (07/2021 a 07/2022)?
5. Qual a distribuição de investidores por faixa etária e gênero (01/2002 a 07/2022)?
6. Qual a distribuição de investidores por faixa etária, gênero e estado civil (01/2002 e 07/2022)?
7. Qual a distribuição de investidores em um ranking de 10 perfis profissionais por faixa etária e gênero (01/2002 a 07/2022)?

Importação de pacotes

In [1]:
```
# Importação de pacotes e definição de parâmetros globais

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import gc

from tabulate import tabulate
from statistics import mode

warnings.filterwarnings('ignore')
sns.set_style('darkgrid')
```

Carregamento dos dados

In [2]:
```
# Efetuando limpeza de memória antes do carregamento de dados
gc.collect()

# Caminho do arquivo csv
csv = 'dados/InvestidoresTesouroDireto.csv'

# Carregando em uma lista os nomes das colunas
cols = list(pd.read_csv(csv, encoding='ISO-8859-1', sep=';', nrows=1))

# Criando um dataframe a partir do arquivo csv no diretório dados
# Devido ao tamanho a leitura dos dados será dividida em blocos

chunks = []
```

```
for i in pd.read_csv(csv, encoding='ISO-8859-1', sep=';', usecols=[i for i in cols if i != 'Pais do Investid
    chunks.append(i)

df = pd.concat(chunks)

# Informações do dataset como nome das colunas, contagem de linhas, tipo de dados e memória utilizada

df.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34163675 entries, 0 to 34163674
Data columns (total 10 columns):
 #   Column                Dtype
---  ------                -----
 0   Codigo do Investidor  int64
 1   Data de Adesao        object
 2   Estado Civil          object
 3   Genero                object
 4   Profissao             object
 5   Idade                 int64
 6   UF do Investidor      object
 7   Cidade do Investidor  object
 8   Situacao da Conta     object
 9   Operou 12 Meses       object
dtypes: int64(2), object(8)
memory usage: 17.2 GB
```

Detalhes do dataframe

In [3]:

```
# Amostra de dados do dataframe

df.sample(5)
```

Out[3]:

	Codigo do Investidor	Data de Adesao	Estado Civil	Genero	Profissao	Idade	UF do Investidor	Cidade do Investidor	Situacao da Conta	Operou 12 Meses
31844833	6098840	20/04/2021	Solteiro(a)	F	OUTROS	24	SP	SAO PAULO	A	N
13900871	22820790	17/09/2021	Desquitado(a)	M	ESCULTOR, PINTOR E ASSEMELHADOS	35	MG	RESENDE COSTA	A	N
5943052	18746488	21/12/2021	Solteiro(a)	M	OUTROS TRABALHADORES DE NÍVEL SUPERIOR LIGADOS...	28	RJ	RIO DE JANEIRO	A	N
29605468	19883536	09/06/2021	Desquitado(a)	F	DECORADOR	29	PR	CAMBIRA	A	N
4945833	30229413	14/02/2022	Solteiro(a)	M	OUTROS	19	RO	VILHENA	A	N

In [4]:

```
# 5 primeiros registros do dataframe

df.head(5)
```

Out[4]:

	Codigo do Investidor	Data de Adesao	Estado Civil	Genero	Profissao	Idade	UF do Investidor	Cidade do Investidor	Situacao da Conta	Operou 12 Meses
0	1680523	06/07/2021	Desquitado(a)	M	MÉDICO	61	ES	COLATINA	A	N
1	1680525	28/04/2021	Solteiro(a)	F	SECRETARIO, ESTENÓGRAFO, DATILÓGRAFO, RECEPCIO...	48	SP	SAO BERNARDO DO CAMPO	A	S
2	1680527	22/03/2017	Não se aplica	M	OUTROS	32	DF	BRASILIA	A	N
3	1680528	28/07/2017	Desquitado(a)	M	SERVIDO PÚBLICO ESTADUAL	52	PE	RECIFE	A	N
4	1680529	05/05/2022	Solteiro(a)	M	OUTROS	30	AM	MANAUS	A	N

In [5]:
```
# 5 últimos registros do dataframe

df.tail(5)
```

Out[5]:

	Codigo do Investidor	Data de Adesao	Estado Civil	Genero	Profissao	Idade	UF do Investidor	Cidade do Investidor	Situacao da Conta	Operou 12 Meses
34163670	31198462	21/03/2022	Solteiro(a)	F	BIÓLOGO E BIOMÉDICO	25	RS	VIAMAO	A	N
34163671	31199515	21/03/2022	Solteiro(a)	F	TRABALHADOR AUTÔNOMO	22	RJ	QUEIMADOS	A	N
34163672	31199859	21/03/2022	Solteiro(a)	M	ADMINISTRADOR	30	SC	JOINVILLE	A	N
34163673	31199860	21/03/2022	Solteiro(a)	M	ADMINISTRADOR	60	SC	BRUSQUE	A	N
34163674	31200082	21/03/2022	Solteiro(a)	F	OUTROS	44	PR	CURITIBA	A	N

In [6]:
```
# Quantidade de linhas e colunas

df.shape
```

Out[6]:
```
(34163675, 10)
```

In [7]:
```
# Renomeando as colunas

print(df.columns.to_list())

df.rename(columns={
    'Codigo do Investidor': 'codigo',
    'Data de Adesao': 'data_adesao',
    'Estado Civil': 'estado_civil',
    'Genero': 'genero',
    'Profissao': 'profissao',
    'Idade': 'idade',
    'UF do Investidor': 'uf',
    'Cidade do Investidor': 'cidade',
    'Situacao da Conta': 'situacao',
    'Operou 12 Meses': 'operacao'
}, inplace=True)

print('\n', df.columns.to_list())
```

```
['Codigo do Investidor', 'Data de Adesao', 'Estado Civil', 'Genero', 'Profissao', 'Idade', 'UF do Investido
r', 'Cidade do Investidor', 'Situacao da Conta', 'Operou 12 Meses']

 ['codigo', 'data_adesao', 'estado_civil', 'genero', 'profissao', 'idade', 'uf', 'cidade', 'situacao', 'oper
acao']
```

Tratando os tipos de valores

In [8]:
```
# Identificando os tipos de dados das colunas

df.dtypes
```

Out[8]:
```
codigo          int64
data_adesao    object
estado_civil   object
genero         object
profissao      object
idade           int64
uf             object
cidade         object
situacao       object
operacao       object
dtype: object
```

In [9]:
```
# Amostra de dados das colunas que terão seus tipos alterados
```

```
df[['codigo', 'data_adesao', 'idade', 'situacao', 'operacao']].sample(5)
```

Out[9]:

	codigo	data_adesao	idade	situacao	operacao
17699404	30982940	14/03/2022	55	A	N
8755985	18363573	08/07/2021	27	A	N
28894338	17409215	08/03/2021	33	A	N
20679727	10887192	08/04/2020	26	A	N
3661076	28810956	03/01/2022	29	A	N

In [10]:

```
# Efetuando as conversões necessárias dos tipos de dados e exibindo novamente as informações das colunas

df['codigo'] = df['codigo'].astype('int32', errors='ignore')
df['data_adesao'] = pd.to_datetime(df['data_adesao'], format='%d/%m/%Y')
df['idade'] = df['idade'].astype('int8', errors='ignore')
df['situacao'] = df['situacao'].astype('category', errors='ignore')
df['operacao'] = df['operacao'].astype('category', errors='ignore')

df.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34163675 entries, 0 to 34163674
Data columns (total 10 columns):
 #   Column        Dtype
---  ------        -----
 0   codigo        int32
 1   data_adesao   datetime64[ns]
 2   estado_civil  object
 3   genero        object
 4   profissao     object
 5   idade         int8
 6   uf            object
 7   cidade        object
 8   situacao      category
 9   operacao      category
dtypes: category(2), datetime64[ns](1), int32(1), int8(1), object(5)
memory usage: 11.3 GB
```

Tratando valores nulos

In [11]:

```
# Identificando a quantidade de valores nulos

df.isnull().sum()
```

Out[11]:

```
codigo           0
data_adesao      0
estado_civil     0
genero           0
profissao        0
idade            0
uf               0
cidade          14
situacao         0
operacao         0
dtype: int64
```

In [12]:

```
# Identificando a quantidade campos preenchidos com espaços

# Função para identifcar campos preenchidos com espaços
# param1: dataframe
# param2: coluna

def verifica_espacos(param1, param2):
    x = param1[param2].isin([' ', '  ', '   ', '    ', '     ']).value_counts()
    try:
        print(param2, ': ', x[1])
    except:
        print(param2, ': ', 0)
```

```
verifica_espacos(df, 'codigo')
verifica_espacos(df, 'data_adesao')
verifica_espacos(df, 'estado_civil')
verifica_espacos(df, 'genero')
verifica_espacos(df, 'profissao')
verifica_espacos(df, 'idade')
verifica_espacos(df, 'uf')
verifica_espacos(df, 'cidade')
verifica_espacos(df, 'situacao')
verifica_espacos(df, 'operacao')
```

```
codigo :  0
data_adesao :  0
estado_civil :  0
genero :  0
profissao :  0
idade :  0
uf :  11
cidade :  0
situacao :  0
operacao :  0
```

In [13]:

```
# Usando o valor mais frequente para preencher os campos da coluna [uf]

x = mode(df['uf'])

print('uf : ', x)

df.loc[df['uf'].isin([' ', '  ', '   ', '    ', '     ']), 'uf'] = x

verifica_espacos(df, 'uf')
```

```
uf :  SP
uf :  0
```

In [14]:

```
# Usando o valor mais frequente para preencher os campos da coluna [cidade] de acordo com a coluna [estado]

y = df.groupby('uf')['cidade'].value_counts()
x = y.loc[x].index[0]

print('cidade : ', x)

df['cidade'].fillna(x, inplace=True)

print('cidade : ', df['cidade'].isnull().sum())
```

```
cidade :  SAO PAULO
cidade :  0
```

In [15]:

```
# Identificando valores inconsistentes na coluna [idade]
# Registros com valor '0' representam erro no cadastro

print(df.loc[df['idade'] == 0, 'idade'].value_counts(), '\n')

# Usando o valor mais frequente para preencher os campos da coluna [idade]

df.loc[df['idade'] == 0, 'idade'] = mode(df['idade'])

x = df['idade'].unique()
print(np.sort(x))
```

```
0    696
Name: idade, dtype: int64

[  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
 109 110 111 112 113 114 115 116 117 118 119 120 121 122]
```

Removendo registros que não pertencem a pessoas físicas

In [16]:
```
# Excluindo colunas que possuem dados que funcionam como indicadores de investidores que não são pessoas fís

print(df.shape)

# Função para identifcar e excluir linhas com valores inválidos
# param1: dataframe
# param2: coluna
# param3: valor inválido


def exclui_linhas(param1, param2, param3):
    x = param1[param1[param2] == param3]
    param1 = param1.drop(x.index)
    return param1


df = exclui_linhas(df, 'estado_civil', 'Não se aplica')
df = exclui_linhas(df, 'profissao', 'Não se aplica')
df = exclui_linhas(df, 'genero', 'N')
df = exclui_linhas(df, 'data_adesao', '01/01/1900')

df.shape
```

```
(34163675, 10)
```

Out[16]:
```
(32838395, 10)
```

Amostra dos dados após tratamento das informações

In [17]:
```
df.sample(10)
```

Out[17]:

	codigo	data_adesao	estado_civil	genero	profissao	idade	uf	cidade	situacao	operacao
19108128	13761071	2021-05-27	Solteiro(a)	M	OUTROS	37	RJ	RIO DE JANEIRO	A	N
2211228	11235925	2021-05-06	Divorciado(a)	M	SERVIDOR PÚBLICO FEDERAL	52	RS	PASSO FUNDO	A	N
8641697	14691603	2021-10-15	Solteiro(a)	F	OUTROS TRABALHADORES DE NÍVEL SUPERIOR LIGADOS...	28	MG	BELO HORIZONTE	A	N
21864013	18561395	2021-04-27	Solteiro(a)	M	OUTROS	48	PR	CLEVELANDIA	A	N
2492186	21207662	2021-08-23	Solteiro(a)	F	OUTROS	24	PB	JOAO PESSOA	A	N
11630123	11059369	2022-04-04	Desquitado(a)	M	VENDEDOR DE COMÉRCIO VAREJISTA E ATACADISTA	23	SP	SAO PAULO	A	N
19402500	3064926	2018-04-23	Solteiro(a)	F	OUTROS	28	SP	SAO PAULO	A	N
8774086	164849	2008-12-11	Solteiro(a)	M	MÉDICO	41	RJ	RESENDE	D	N
19394968	3195959	2018-05-02	Solteiro(a)	M	EMPRESÁRIO	47	BA	SIMOES FILHO	A	N
18264450	1892138	2017-03-21	Solteiro(a)	M	ENGENHEIRO	27	PA	BELEM	A	N

Criando um dataframe sem contas duplicadas

In [18]:
```
# Um investidor pode ter mais de uma conta em mais de uma instituição financeira habilitada
# a efetuar operações no tesouro direto, logo será criado um dataframe onde só contara
# a primeira conta criada pelo investidor

df_investidores = df.copy()

df_investidores.sort_values(by=['codigo', 'data_adesao'])
```

```
df_investidores.drop_duplicates(subset=['codigo'], inplace=True)

df_investidores.shape
```

Out[18]: (19110853, 10)

Criando um dataframe substituindo a idade por faixa etária

In [19]:
```
# Para melhor resultado em algumas análises será criado outro dataframe onde
# as idades serão substituídas por faixas etárias

df_investidores_faixa = df_investidores.copy()

faixa = [1, 18, 25, 35, 45, 55, 65, df_investidores_faixa['idade'].max()]
rotulos = ['Menos de 18 anos',
           '18 a 24 anos',
           '25 a 34 anos',
           '35 a 44 anos',
           '45 a 54 anos',
           '55 a 64 anos',
           'Mais de 65 anos']

df_investidores_faixa['idade'] = pd.cut(
    df_investidores_faixa['idade'], bins=faixa, labels=rotulos)
df_investidores_faixa = pd.DataFrame(df_investidores_faixa)

df_investidores_faixa.shape
```

Out[19]: (19110853, 10)

Redefinindo os index dos Dataframes

In [20]:
```
df.reset_index(inplace=True, drop=True)

df.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32838395 entries, 0 to 32838394
Data columns (total 10 columns):
 #   Column        Dtype
---  ------        -----
 0   codigo        int32
 1   data_adesao   datetime64[ns]
 2   estado_civil  object
 3   genero        object
 4   profissao     object
 5   idade         int8
 6   uf            object
 7   cidade        object
 8   situacao      category
 9   operacao      category
dtypes: category(2), datetime64[ns](1), int32(1), int8(1), object(5)
memory usage: 10.9 GB
```

In [21]:
```
df_investidores.reset_index(inplace=True, drop=True)

df_investidores.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19110853 entries, 0 to 19110852
Data columns (total 10 columns):
 #   Column        Dtype
---  ------        -----
 0   codigo        int32
 1   data_adesao   datetime64[ns]
 2   estado_civil  object
 3   genero        object
 4   profissao     object
 5   idade         int8
 6   uf            object
 7   cidade        object
 8   situacao      category
```

```
 9   operacao      category
dtypes: category(2), datetime64[ns](1), int32(1), int8(1), object(5)
memory usage: 6.4 GB
```

In [22]:

```
df_investidores_faixa.reset_index(inplace=True, drop=True)

df_investidores_faixa.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19110853 entries, 0 to 19110852
Data columns (total 10 columns):
 #   Column        Dtype
---  ------        -----
 0   codigo        int32
 1   data_adesao   datetime64[ns]
 2   estado_civil  object
 3   genero        object
 4   profissao     object
 5   idade         category
 6   uf            object
 7   cidade        object
 8   situacao      category
 9   operacao      category
dtypes: category(3), datetime64[ns](1), int32(1), object(5)
memory usage: 6.4 GB
```

1. Qual o total de novos investidores por ano (2002 a 2021)?

Qual o total de novos investidores que aderiram ao programa por ano, no período de 2002 a 2021, independentemente se o cadastro em algum momento foi desativado e considerando somente a primeira adesão ao tesouro direto.

In [23]:

```
# Função para criar a coluna [%] no dataframe
# param1: dataframe
# param2: coluna que será usada o cálculo

def coluna_percentual(param1, param2):
    param1['%'] = param1[param2] / param1[param2].sum() * 100
    return param1['%']


df_temp = df_investidores.copy()
df_temp['ano'] = df_temp['data_adesao'].dt.year
df_temp = df_temp.drop(df_temp.loc[df_temp['ano'] == 2022].index)
df_investidores_ano = df_temp['ano'].value_counts()
df_investidores_ano = pd.DataFrame(df_investidores_ano)
df_investidores_ano.sort_index(inplace=True)
df_investidores_ano.index.names = ['ano']
df_investidores_ano.rename(columns={'ano': 'qtde'}, inplace=True)
coluna_percentual(df_investidores_ano, 'qtde')

df_investidores_ano
```

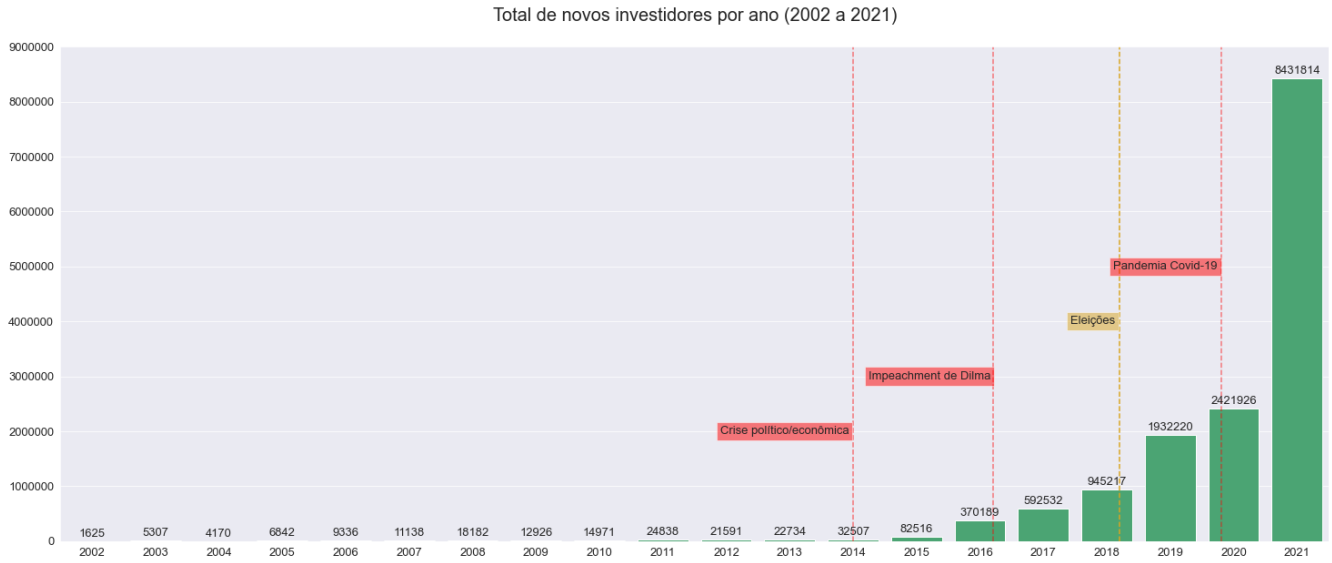
Out[23]:

ano	qtde	%
2002	1625	0.010860
2003	5307	0.035468
2004	4170	0.027870
2005	6842	0.045727
2006	9336	0.062396
2007	11138	0.074439
2008	18182	0.121516
2009	12926	0.086389
2010	14971	0.100056
2011	24838	0.166001

ano	qtde	%
2012	21591	0.144300
2013	22734	0.151939
2014	32507	0.217255
2015	82516	0.551482
2016	370189	2.474099
2017	592532	3.960092
2018	945217	6.317206
2019	1932220	12.913681
2020	2421926	16.186552
2021	8431814	56.352671

In [24]:

```
plt.figure(figsize=(25, 10))
plot = sns.barplot(x=df_investidores_ano.index,
                   y=df_investidores_ano['qtde'],
                   orient='v',
                   palette=['mediumseagreen'])
for i in plot.patches:
    plot.annotate(format(i.get_height(), '3.0f'),
                  (i.get_x() + i.get_width() / 2, i.get_height()),
                  ha='center',
                  va='baseline',
                  fontsize=13,
                  xytext=(0, 5),
                  textcoords='offset points')
plt.axvline(x=12, linestyle='--', color='red', alpha=0.5)
plt.axvline(x=14.2, linestyle='--', color='red', alpha=0.5)
plt.axvline(x=16.2, linestyle='--', color='goldenrod')
plt.axvline(x=17.8, linestyle='--', color='red', alpha=0.5)
plt.text(9.9, 2000000, 'Crise político/econômica', va='center',
         bbox=dict(facecolor='red', alpha=0.5), fontsize=13)
plt.text(12.24, 3000000, 'Impeachment de Dilma', va='center',
         bbox=dict(facecolor='red', alpha=0.5), fontsize=13)
plt.text(15.42, 4000000, 'Eleições', va='center', bbox=dict(
    facecolor='goldenrod', alpha=0.5), fontsize=13)
plt.text(16.1, 5000000, 'Pandemia Covid-19', va='center',
         bbox=dict(facecolor='red', alpha=0.5), fontsize=13)
plt.ylim(0, 9000000)
plt.xticks(size=13)
plt.yticks(size=13)
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('')
plt.ylabel('')
plt.title('\nTotal de novos investidores por ano (2002 a 2021)\n', fontsize=20)
plt.show(plot)
```



Desde sua criação em **2002** o número de novos inscritos só ultrapassou a marca de 100.000 em **2016** em um **período de 14 anos** com 370.189 adesões, anteriormente em **2014** iniciou-se uma série de **acontecimentos políticos/econômicos que culmiram em uma crise** no país, já em **2017** a marca de meio milhão foi superada quando houve um salto para 592.532 de inscritos, em **2019, ano posterior as eleições**, o record de um milhão foi quebrado com um total 1.932.220 cadastros, no ano de **2020** ocorreu a pandemia de Covid-19, com medidas restritivas, lockdown e intensificação do home office encerrando o ano com um total 2.421.926 adesões, no ano seguinte, **2021**, o número de inscritos era de **8.431.814**, ou seja, neste ano houve um aumento de **348%** de novos investidores.

2. Qual o total de investidores por estado (01/2002 a 07/2022)?

Desde que foi criado o programa em janeiro de 2002, qual o total de investidores que efetuaram a adesão ao tesouro direto até julho de 2022.

In [25]:
```
df_temp = df_investidores.copy()
df_investidores_uf = df_temp['uf'].value_counts()
df_investidores_uf = pd.DataFrame(df_investidores_uf)
df_investidores_uf.sort_index(inplace=True)
df_investidores_uf.index.name = 'uf'
estados = ['Acre', 'Alagoas', 'Amazonas', 'Amapá', 'Bahia', 'Ceará', 'Distrito Federal', 'Espírito Santo',
           'Goiás', 'Maranhão', 'Minas Gerais', 'Mato Grosso do Sul', 'Mato Grosso', 'Pará', 'Paraíba',
           'Pernambuco', 'Piauí', 'Paraná', 'Rio de Janeiro', 'Rio Grande do Norte', 'Rondônia', 'Roraima',
           'Rio Grande do Sul', 'Santa Catarina', 'Sergipe', 'São Paulo', 'Tocantins']
df_investidores_uf.insert(0, 'estado', estados)
df_investidores_uf.rename(columns={'uf': 'qtde'}, inplace=True)
coluna_percentual(df_investidores_uf, 'qtde')

df_investidores_uf
```

Out[25]:

uf	estado	qtde	%
AC	Acre	45501	0.238090
AL	Alagoas	174578	0.913502
AM	Amazonas	266487	1.394428
AP	Amapá	53407	0.279459
BA	Bahia	895087	4.683658
CE	Ceará	571215	2.988956
DF	Distrito Federal	490326	2.565694
ES	Espírito Santo	379205	1.984239
GO	Goiás	642741	3.363225

uf	estado	qtde	%
MA	Maranhão	292672	1.531444
MG	Minas Gerais	1876814	9.820671
MS	Mato Grosso do Sul	229031	1.198434
MT	Mato Grosso	300086	1.570239
PA	Pará	440188	2.303340
PB	Paraíba	235554	1.232567
PE	Pernambuco	601598	3.147939
PI	Piauí	157228	0.822716
PR	Paraná	1110137	5.808935
RJ	Rio de Janeiro	1938444	10.143158
RN	Rio Grande do Norte	233309	1.220819
RO	Rondônia	131899	0.690179
RR	Roraima	42865	0.224297
RS	Rio Grande do Sul	945369	4.946765
SC	Santa Catarina	811049	4.243918
SE	Sergipe	149257	0.781006
SP	São Paulo	5997831	31.384423
TO	Tocantins	98975	0.517899

In [26]:

```
# Função para criar dataframes de estados divididos por regiões
# param1: dataframe
# param2: lista de estados

def uf_por_regiao(param1, param2):
    df_temp = param1[param1.index.isin(param2)]
    df_temp['%'] = df_temp['qtde'] / df_temp['qtde'].sum() * 100
    return df_temp


sul = ['PR', 'SC', 'RS']
sudeste = ['ES', 'MG', 'RJ', 'SP']
centro_oeste = ['DF', 'GO', 'MS', 'MT']
nordeste = ['AL', 'BA', 'CE', 'MA', 'PB', 'PE', 'PI', 'RN', 'SE']
norte = ['AC', 'AM', 'AP', 'PA', 'RO', 'RR', 'TO']

df_investidores_sul = uf_por_regiao(df_investidores_uf, sul)
df_investidores_sudeste = uf_por_regiao(df_investidores_uf, sudeste)
df_investidores_centro_oeste = uf_por_regiao(df_investidores_uf, centro_oeste)
df_investidores_nordeste = uf_por_regiao(df_investidores_uf, nordeste)
df_investidores_norte = uf_por_regiao(df_investidores_uf, norte)

print('\n REGIÃO NORTE')
print(tabulate(df_investidores_norte, headers='keys', tablefmt='fancy_grid'))
print('\n REGIÃO NORDESTE')
print(tabulate(df_investidores_nordeste, headers='keys', tablefmt='fancy_grid'))
print('\n REGIÃO CENTRO-OESTE')
print(tabulate(df_investidores_centro_oeste, headers='keys', tablefmt='fancy_grid'))
print('\n REGIÃO SUDESTE')
print(tabulate(df_investidores_sudeste, headers='keys', tablefmt='fancy_grid'))
print('\n REGIÃO SUL')
print(tabulate(df_investidores_sul, headers='keys', tablefmt='fancy_grid'))
```

```
 REGIÃO NORTE
```

uf	estado	qtde	%
AC	Acre	45501	4.2157

AM	Amazonas	266487	24.6902
AP	Amapá	53407	4.9482
PA	Pará	440188	40.7838
RO	Rondônia	131899	12.2205
RR	Roraima	42865	3.97147
TO	Tocantins	98975	9.17011

REGIÃO NORDESTE

uf	estado	qtde	%
AL	Alagoas	174578	5.27347
BA	Bahia	895087	27.0378
CE	Ceará	571215	17.2547
MA	Maranhão	292672	8.84072
PB	Paraíba	235554	7.11536
PE	Pernambuco	601598	18.1724
PI	Piauí	157228	4.74938
RN	Rio Grande do Norte	233309	7.04755
SE	Sergipe	149257	4.5086

REGIÃO CENTRO-OESTE

uf	estado	qtde	%
DF	Distrito Federal	490326	29.4989
GO	Goiás	642741	38.6685
MS	Mato Grosso do Sul	229031	13.7789
MT	Mato Grosso	300086	18.0537

REGIÃO SUDESTE

uf	estado	qtde	%
ES	Espírito Santo	379205	3.72051
MG	Minas Gerais	1876814	18.414
RJ	Rio de Janeiro	1938444	19.0187
SP	São Paulo	5997831	58.8467

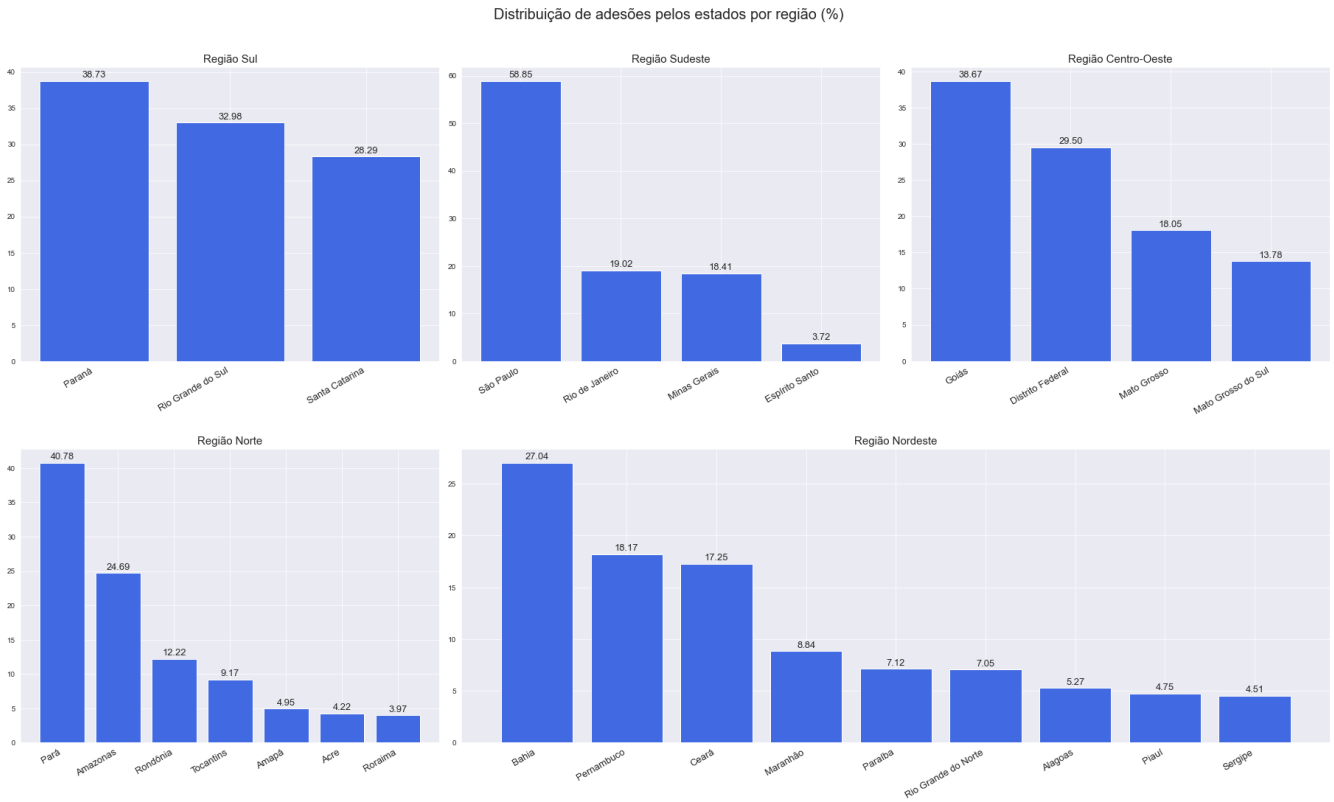
REGIÃO SUL

uf	estado	qtde	%
PR	Paraná	1110137	38.7272
RS	Rio Grande do Sul	945369	32.9793
SC	Santa Catarina	811049	28.2935

In [27]:

```
# Função para gerar gráficos de barras de estados
# param1: dataframe
# param2: titulo
# param3: axis
# param4: spec
```

```
def graf_estado(param1, param2, param3, param4):
    param3 = fig.add_subplot(param4)
    param3.bar(param1['estado'], param1['%'], color='royalblue')
    param3.set_title(param2, fontsize=15)
    for i in param3.patches:
        param3.annotate(format(i.get_height(), '.2f'),
                        (i.get_x() + i.get_width()/2, i.get_height()),
                        ha='center', va='baseline', fontsize=13,
                        xytext=(0, 5), textcoords='offset points')
    plt.setp(param3.get_xticklabels(), rotation=30, ha='right', fontsize=13)


df_investidores_sul.sort_values(by='%', ascending=False, inplace=True)
df_investidores_sudeste.sort_values(by='%', ascending=False, inplace=True)
df_investidores_centro_oeste.sort_values(by='%', ascending=False, inplace=True)
df_investidores_nordeste.sort_values(by='%', ascending=False, inplace=True)
df_investidores_norte.sort_values(by='%', ascending=False, inplace=True)

fig = plt.figure(figsize=(25, 15), constrained_layout=True)
spec = fig.add_gridspec(2, 3)

ax00 = ax01 = ax02 = ax10 = ax11 = None

graf_estado(df_investidores_sul, '\nRegião Sul', ax00, spec[0, 0])
graf_estado(df_investidores_sudeste, 'Região Sudeste', ax01, spec[0, 1])
graf_estado(df_investidores_centro_oeste, 'Região Centro-Oeste', ax02, spec[0, 2])
graf_estado(df_investidores_norte, 'Região Norte', ax10, spec[1, 0])
graf_estado(df_investidores_nordeste, '\nRegião Nordeste', ax11, spec[1, 1:])

fig.suptitle(
    'Distribuição de adesões pelos estados por região (%)\n', fontsize=20)
plt.show()
```



Considerando o perído de **janeiro de 2002 até julho de 2022**, **São Paulo** é o estado com maior número de adesões com 5.997.831 representando aproximadamente **31,39%** do total de **19.110.853** cadastros, seguido de **Rio de Janeiro** com 1.938.444 e **Minas Gerais** com 1.876.814, respectivamente **10,15%** e **9,9%** aproximados, somandos, os três estados da região sudeste, respondem por mais de **51% do total de inscritos** no programa. Na distribuição de adesões por região o cenário é de que na região sul o estado do Paraná responde por 38,73%, na região sudeste São Paulo responde por 58,85%, na região centro-oeste Goiás possui 38,67% dos cadastros, na região norte o Pará concentra 40,78% e a na região nordeste a Bahia detém 27,04%, São Paulo que apresenta a maior diferença na sua região concentrando quase 60% das adesões.

3. Qual o total de investidores em um ranking de 10 cidades (01/2002 a 07/2022)?

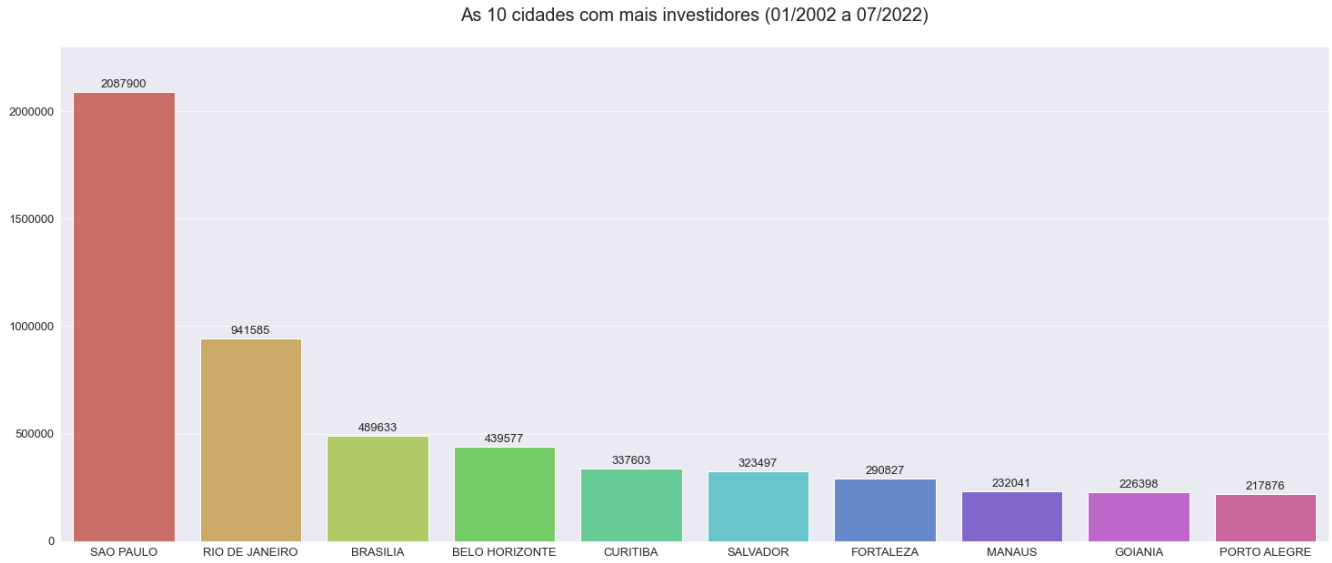
Considerando o período de janeiro de 2002 à julho de 2022, quais são as dez cidades onde estão concentradas a maioria dos investidores do tesouro direto.

In [28]:
```
df_temp = df_investidores.copy()
df_investidores_cidades = df_temp['cidade'].value_counts().nlargest(10)
df_investidores_cidades = pd.DataFrame(df_investidores_cidades)
df_investidores_cidades.rename(columns={'cidade': 'qtde'}, inplace=True)
df_investidores_cidades.index.name = 'cidade'
coluna_percentual(df_investidores_cidades, 'qtde')

df_investidores_cidades
```

Out[28]:

cidade	qtde	%
SAO PAULO	2087900	37.371103
RIO DE JANEIRO	941585	16.853331
BRASILIA	489633	8.763890
BELO HORIZONTE	439577	7.867943
CURITIBA	337603	6.042721
SALVADOR	323497	5.790239
FORTALEZA	290827	5.205482
MANAUS	232041	4.153278
GOIANIA	226398	4.052274
PORTO ALEGRE	217876	3.899740

In [29]:
```
plt.figure(figsize=(25, 10))
plot = sns.barplot(x=df_investidores_cidades.index,
                   y=df_investidores_cidades['qtde'],
                   orient='v',
                   palette='hls')
for i in plot.patches:
    plot.annotate(format(i.get_height(), '3.0f'),
                  (i.get_x()+i.get_width()/2, i.get_height()),
                  ha='center',
                  va='baseline',
                  fontsize=13,
                  xytext=(0, 5),
                  textcoords='offset points')
plt.ylim(0, 2300000)
plt.xticks(size=13)
plt.yticks(size=13)
plt.ticklabel_format(style='plain', axis='y')
plt.xlabel('')
plt.ylabel('')
plt.title('\nAs 10 cidades com mais investidores (01/2002 a 07/2022)\n', fontsize=20)
plt.show(plot)
```



A cidade de **São Paulo** é a que mais possui inscritos, com **2.087.900** concentrando mais que o dobro da segunda cidade, **Rio de Janeiro**, com **941.585**, em terceiro está a capital, **Brasília**, com **489.633** e assim seguido pelas cidades de Belo Horizonte, 439.577, Curitiba, 337.603, Salvador, 323.497, Fortaleza, 290.827, Manaus, 232.041, Goiânia, 226.398 e Porto Alegre com 217.876. Todas as regiões do país possuem ao menos um representante no ranking, destaque para região sudeste com três, São Paulo, Rio de Janeiro e Belo Horizonte e a região norte com apenas Manaus a representando.

4. Qual a quantidade de investidores que operaram nos últimos 12 meses (07/2021 a 07/2022)?

Qual a quantidade e percentual de investidores com contas ativas e desativadas que fizeram alguma operação nos últimos 12 meses, período que corresponde de julho de 2021 a julho de 2022.

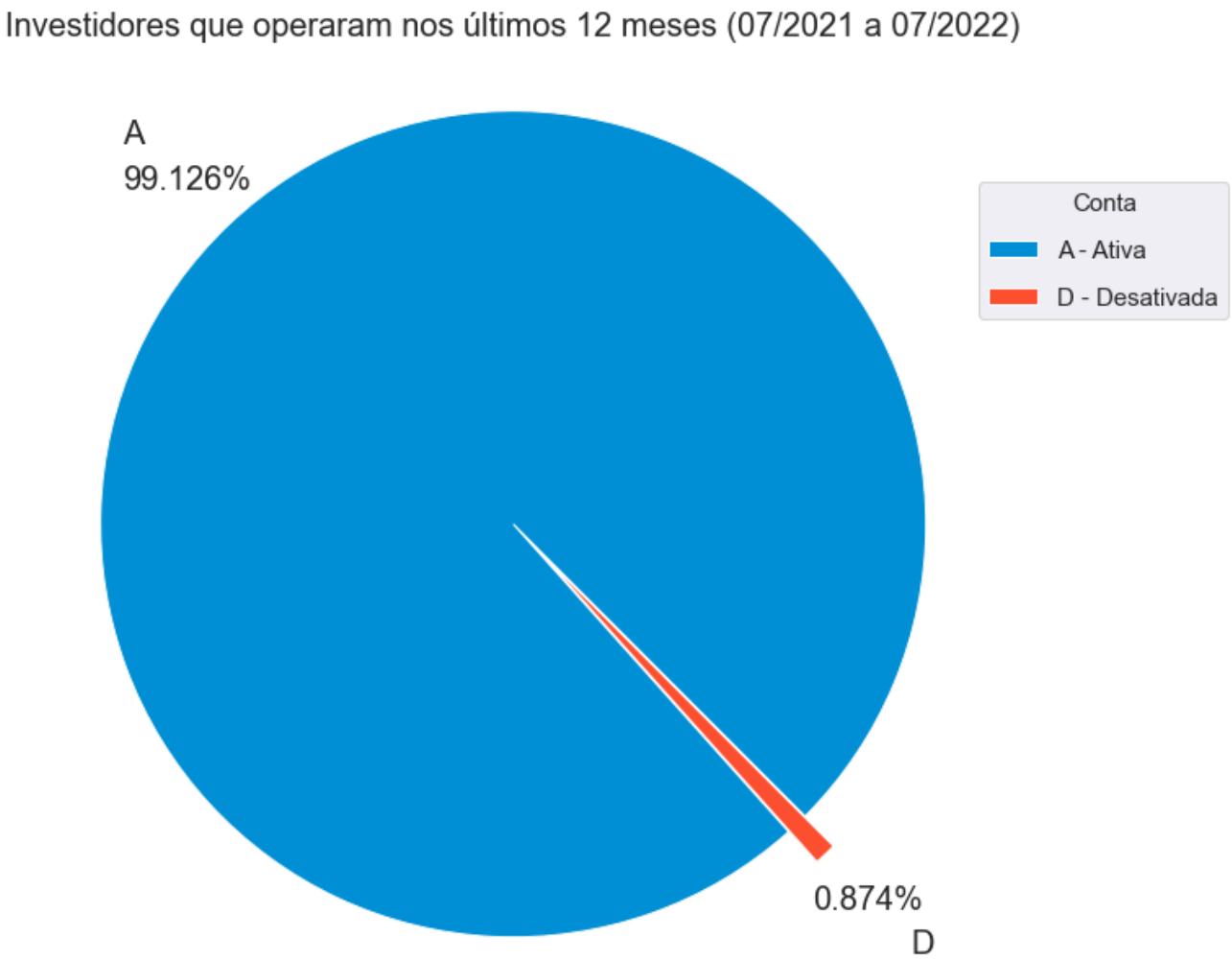
In [30]:
```
df_temp = df.copy()
df_temp = df_temp[df_temp['operacao'] == 'S']
df_temp = df_temp.sort_values(by=['codigo'])
df_temp.drop_duplicates(subset=['codigo'], inplace=True)
df_temp = df_temp.groupby(['situacao'])['operacao'].value_counts()
df_situacao_operacao = pd.DataFrame(df_temp)
coluna_percentual(df_situacao_operacao, 'operacao')
df_situacao_operacao.index.names = ['situacao', 'operacao_mes']

print('''
    Os últimos 12 meses correspondem ao período de julho de 2021 a julho de 2022

    A = Investidores com conta ativa no tesouro direto
    D = Investidores com conta desativada no tesouro direto

    S = Operou nos últimos 12 meses (07/2021 a 07/2022)
    N = Não operou nos últimos 12 meses (07/2021 a 07/2022)
    ''')

df_situacao_operacao
```

```
    Os últimos 12 meses correspondem ao período de julho de 2021 a julho de 2022

    A = Investidores com conta ativa no tesouro direto
    D = Investidores com conta desativada no tesouro direto

    S = Operou nos últimos 12 meses (07/2021 a 07/2022)
    N = Não operou nos últimos 12 meses (07/2021 a 07/2022)
```

Out[30]:

		operacao	%
situacao	**operacao_mes**		
A	**S**	1704061	99.125874
D	**S**	15027	0.874126

In [31]:
```
df_graf = df_situacao_operacao.reset_index()

plt.figure(figsize=(10, 10))
plt.pie(df_graf['%'],
        labels=df_graf['situacao'],
        labeldistance=1.3,
        pctdistance=1.15,
        explode=(0, 0.1),
        colors=['#008fd5', '#fc4f30'],
        autopct='%0.3f%%',
        startangle=315,
        textprops=dict(fontsize=20))
plt.title('\nInvestidores que operaram nos últimos 12 meses (07/2021 a 07/2022)\n', fontsize=20)
plt.legend(loc='best',  bbox_to_anchor=(0.7, 0, 0.6, 0.9), labelspacing=1,
           labels=['A - Ativa', 'D - Desativada'], fontsize=15, title='Conta', title_fontsize=15)
plt.axis('equal')
plt.show()
```



Com **1.704.061** o que representa **99,126%** das contas ativas de investidores que realizaram algum tipo de operação no programa, o restante **15.027**, cerca de **0,874%** de investidores também efetuaram alguma operação no programa, porém, neste mesmo período suas contas foram desativadas.

5. Qual a distribuição de investidores por faixa etária e gênero (01/2002 a 07/2022)?

Qual a distribuição de investidores (quantidade e percentual) entre janeiro de 2002 à julho de 2022 segmentados pela faixa etária e por gênero (masculino ou feminino).

In [32]:
```
df_faixa_genero = df_investidores_faixa.copy()

df_faixa_genero = df_faixa_genero.groupby(['idade'])['genero'].value_counts()
df_faixa_genero = pd.DataFrame(df_faixa_genero)
```

```
df_faixa_genero.rename(columns={'genero': 'qtde'}, inplace=True)
coluna_percentual(df_faixa_genero, 'qtde')

df_faixa_genero
```

Out[32]:

idade	genero	qtde	%
Menos de 18 anos	**M**	252681	1.322304
	F	67951	0.355594
18 a 24 anos	**M**	3396491	17.774166
	F	877548	4.592294
25 a 34 anos	**M**	4966369	25.989490
	F	1720120	9.001554
35 a 44 anos	**M**	3190386	16.695599
	F	1237612	6.476543
45 a 54 anos	**M**	1303771	6.822760
	F	583427	3.053130
55 a 64 anos	**M**	617687	3.232416
	F	356404	1.865097
Mais de 65 anos	**M**	318689	1.667730
	F	220008	1.151323

In [33]:

```
df_investidores_f = df_faixa_genero.loc[df_faixa_genero.index.get_level_values(
    'genero') == 'F']
coluna_percentual(df_investidores_f, 'qtde')
df_investidores_f.index = df_investidores_f.index.droplevel(1)

df_investidores_m = df_faixa_genero.loc[df_faixa_genero.index.get_level_values(
    'genero') == 'M']
coluna_percentual(df_investidores_m, 'qtde')
df_investidores_m.index = df_investidores_m.index.droplevel(1)

print('\n GÊNERO FEMININO')
print(tabulate(df_investidores_f, headers='keys',
               tablefmt='fancy_outline', floatfmt=('', '.0f', '.5f')))
print('\n GÊNERO MASCULINO')
print(tabulate(df_investidores_m, headers='keys',
               tablefmt='fancy_outline', floatfmt=('', '.0f', '.5f')))
```

GÊNERO FEMININO

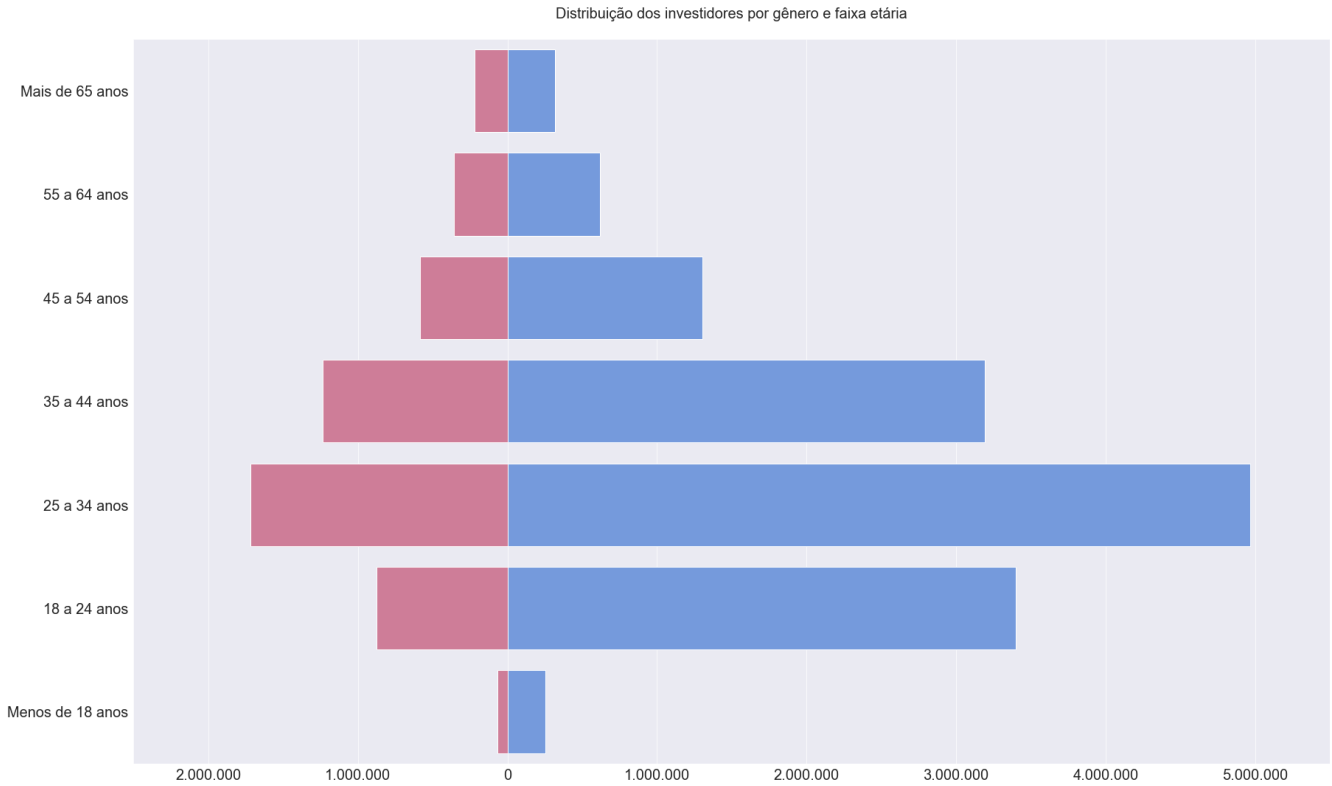
idade	qtde	%
Menos de 18 anos	67951	1.34209
18 a 24 anos	877548	17.33233
25 a 34 anos	1720120	33.97385
35 a 44 anos	1237612	24.44390
45 a 54 anos	583427	11.52319
55 a 64 anos	356404	7.03929
Mais de 65 anos	220008	4.34535

GÊNERO MASCULINO

idade	qtde	%
Menos de 18 anos	252681	1.79894
18 a 24 anos	3396491	24.18107
25 a 34 anos	4966369	35.35770
35 a 44 anos	3190386	22.71372
45 a 54 anos	1303771	9.28210
55 a 64 anos	617687	4.39758
Mais de 65 anos	318689	2.26888

In [34]:

```
df_investidores_f['qtde'] = df_investidores_f['qtde'].mul(-1)

rotulos.reverse()
rotulos_x = ['3.000.000', '2.000.000', '1.000.000', '0',
             '1.000.000', '2.000.000', '3.000.000', '4.000.000', '5.000.000']

plt.figure(figsize=(25, 15), constrained_layout=True)
plot = sns.barplot(x='qtde', y=df_investidores_f.index,
                   data=df_investidores_f, order=rotulos, palette=['palevioletred'])
plot = sns.barplot(x='qtde', y=df_investidores_m.index,
                   data=df_investidores_m, order=rotulos, palette=['cornflowerblue'])
plt.xlim(-2500000, 5500000)
plt.xticks(size=20)
plt.yticks(size=20)
plt.ticklabel_format(style='plain', axis='x')
plt.xlabel('')
plt.ylabel('')
plt.title('\nDistribuição dos investidores por gênero e faixa etária\n', fontsize=20)
plot.set_xticklabels(rotulos_x)
plt.show()
```



6. Qual a distribuição de investidores por faixa etária, gênero e estado civil (01/2002 e 07/2022)?

Qual a distribuição de investidores (quantidade e percentual) entre janeiro de 2002 à julho de 2022 segmentados pelo estado civil e para cada condição divididos por faixa etária e por gênero (masculino ou feminino).

In [35]:

```
df_faixa_genero_civil = df_investidores_faixa.copy()

k = df_faixa_genero_civil['estado_civil'].unique()
k.sort()

df_faixa_genero_civil = df_faixa_genero_civil.groupby(
    ['estado_civil', 'idade'])['genero'].value_counts()
df_faixa_genero_civil = pd.DataFrame(df_faixa_genero_civil)
df_faixa_genero_civil.rename(columns={'genero': 'qtde'}, inplace=True)

for i in k:
    x = df_faixa_genero_civil.loc[i]
    x = x.unstack(0)
```

```
    print('\n\n\033[1m' + i.upper() + '\033[0m')
    display(x)
```

CASADO(A) COM BRASILEIRO(A) NATO(A)

	qtde						
idade	Menos de 18 anos	18 a 24 anos	25 a 34 anos	35 a 44 anos	45 a 54 anos	55 a 64 anos	Mais de 65 anos
genero							
F	41	6147	89390	135351	77175	68297	72784
M	122	8342	150276	271736	153681	106432	84112

CASADO(A) COM BRASILEIRO(A) NATURALIZADO(A)

	qtde						
idade	Menos de 18 anos	18 a 24 anos	25 a 34 anos	35 a 44 anos	45 a 54 anos	55 a 64 anos	Mais de 65 anos
genero							
F	148	8113	35127	32860	13979	6104	1748
M	243	11677	66216	74643	31166	13391	5784

CASADO(A) COM ESTRANGEIRO(A)

	qtde				
idade	25 a 34 anos	35 a 44 anos	45 a 54 anos	55 a 64 anos	Mais de 65 anos
genero					
F	2.0	11.0	25.0	18.0	31.0
M	NaN	27.0	36.0	24.0	48.0

DESQUITADO(A)

	qtde						
idade	Menos de 18 anos	18 a 24 anos	25 a 34 anos	35 a 44 anos	45 a 54 anos	55 a 64 anos	Mais de 65 anos
genero							
F	553	42247	301503	359123	173944	102242	48059
M	2754	179595	1100052	1225005	556558	275099	142916

DIVORCIADO(A)

	qtde						
idade	Menos de 18 anos	18 a 24 anos	25 a 34 anos	35 a 44 anos	45 a 54 anos	55 a 64 anos	Mais de 65 anos
genero							
F	128	6513	66060	122787	96820	74135	35841
M	253	13191	141811	235234	157308	88308	35235

SEPARADO JUDIC.

	qtde						
idade	Menos de 18 anos	18 a 24 anos	25 a 34 anos	35 a 44 anos	45 a 54 anos	55 a 64 anos	Mais de 65 anos
genero							
F	4	684	7880	8317	3765	2973	1772
M	4	764	7166	10306	5269	3310	1976

SOLTEIRO(A)

	qtde

idade	Menos de 18 anos	18 a 24 anos	25 a 34 anos	35 a 44 anos	45 a 54 anos	55 a 64 anos	Mais de 65 anos
genero							
F	67011	810746	1199126	551836	200998	87814	39434
M	249002	3175360	3451435	1301450	359822	110996	37366

UNIÃO ESTÁVEL

qtde

idade	Menos de 18 anos	18 a 24 anos	25 a 34 anos	35 a 44 anos	45 a 54 anos	55 a 64 anos	Mais de 65 anos
genero							
F	3	1633	14160	15783	6535	2921	789
M	1	2123	24621	33720	14602	6167	2748

VIÚVO(A)

qtde

idade	Menos de 18 anos	18 a 24 anos	25 a 34 anos	35 a 44 anos	45 a 54 anos	55 a 64 anos	Mais de 65 anos
genero							
F	63	1465	6872	11544	10186	11900	19550
M	302	5439	24792	38265	25329	13960	8504

In [36]:

```
# Função para gerar gráficos de barras de faixa etária, estado civil e gênero
# param1: dataframe
# param2: estado civil
# param3: axis

def graf_faixa_estado_genero(param1, param2, param3):
    sns.barplot(data=param1.loc[param1['estado_civil'] == param2],
                x='qtde', y='idade', hue='genero', hue_order=['M', 'F'],
                ci=False, orient='horizontal', dodge=True,
                ax=param3, palette=['cornflowerblue', 'palevioletred'])
    param3.tick_params(labelsize=15)
    param3.set_ylabel('')
    param3.set_xlabel('')
    param3.legend()
    param3.invert_yaxis()
    param3.set_title(param2, loc='right', fontsize=15)
    param3.ticklabel_format(style='plain', axis='x')


fig, ((ax1, ax2), (ax3, ax4), (ax5, ax6), (ax7, ax8), (ax9, ax10)
      ) = plt.subplots(ncols=2, nrows=5, figsize=(25, 20))


df_temp = df_faixa_genero_civil.reset_index()

graf_faixa_estado_genero(df_temp, 'Casado(a) com brasileiro(a) nato(a)', ax1)
graf_faixa_estado_genero(df_temp, 'Casado(a) com brasileiro(a) naturalizado(a)', ax2)
graf_faixa_estado_genero(df_temp, 'Casado(a) com estrangeiro(a)', ax3)
graf_faixa_estado_genero(df_temp, 'Desquitado(a)', ax4)
graf_faixa_estado_genero(df_temp, 'Divorciado(a)', ax5)
graf_faixa_estado_genero(df_temp, 'Separado judic.', ax6)
graf_faixa_estado_genero(df_temp, 'Solteiro(a)', ax7)
graf_faixa_estado_genero(df_temp, 'União estável', ax8)
graf_faixa_estado_genero(df_temp, 'Viúvo(a)', ax9)

fig.suptitle(
    'Distribuição de investidores por faixa etária, gênero e por estado civil\n\n', fontsize=20)
plt.subplots_adjust(top=1, bottom=0.95)
plt.tight_layout()
plt.show()
```

7. Qual a distribuição de investidores em um ranking de 10 perfis profissionais por gênero (01/2002 a 07/2022)?

Considerando o período de janeiro de 2002 à julho de 2022, segmentados por gênero (masculino e feminino), quais são os dez perfis profissionais que mais concentram investidores do tesouro direto.

In [37]:

```
df_temp = df_investidores.copy()

df_temp = df_temp.groupby(['profissao'])['genero'].value_counts()
df_investidores_profissao_f = df_temp.loc[df_temp.index.get_level_values('genero') == 'F'].sort_values().nla
df_investidores_profissao_f = pd.DataFrame(df_investidores_profissao_f)
df_investidores_profissao_f.rename(columns={'genero': 'qtde'}, inplace=True)
df_investidores_profissao_f.index.name = 'profissao'
coluna_percentual(df_investidores_profissao_f, 'qtde')
df_investidores_profissao_f.index = df_investidores_profissao_f.index.droplevel(1)

df_investidores_profissao_m = df_temp.loc[df_temp.index.get_level_values('genero') == 'M'].sort_values().nla
df_investidores_profissao_m = pd.DataFrame(df_investidores_profissao_m)
df_investidores_profissao_m.rename(columns={'genero': 'qtde'}, inplace=True)
df_investidores_profissao_m.index.name = 'profissao'
coluna_percentual(df_investidores_profissao_m, 'qtde')
df_investidores_profissao_m.index = df_investidores_profissao_m.index.droplevel(1)

print('\n GÊNERO FEMININO')
print(tabulate(df_investidores_profissao_f, headers='keys',
               tablefmt='fancy_outline', floatfmt=('', '.0f', '.5f')))

print('\n GÊNERO MASCULINO')
print(tabulate(df_investidores_profissao_m, headers='keys',
               tablefmt='fancy_outline', floatfmt=('', '.0f', '.5f')))
```

```
 GÊNERO FEMININO
```

profissao	qtde	%
OUTROS	1910865	58.20718
ADMINISTRADOR	239297	7.28927
TRABALHADOR AUTÔNOMO	206637	6.29440
ESTUDANTE	196132	5.97441
AUXILIAR DE ESCRITÓRIO E ASSEMELHADOS	142454	4.33932
PROFESSOR DE PRIMEIRO E SEGUNDO GRAUS	128393	3.91100
VENDEDOR DE COMÉRCIO VAREJISTA E ATACADISTA	126281	3.84667
APOSENTADO (EXCETO FUNCIONÁRIO PÚBLICO)	126217	3.84472
ADVOGADO	104301	3.17713
EMPRESÁRIO	102291	3.11590

GÊNERO MASCULINO

profissao	qtde	%
OUTROS	3279572	39.84338
VENDEDOR PRACISTA, REPRESENTANTE COMERCIAL, CAIXEIRO VIAJANTE	831583	10.10287
AUXILIAR DE ESCRITÓRIO E ASSEMELHADOS	817838	9.93588
ADMINISTRADOR	814718	9.89797
ESTUDANTE	641200	7.78991
VENDEDOR DE COMÉRCIO VAREJISTA E ATACADISTA	510976	6.20782
ENGENHEIRO	412128	5.00692
TRABALHADOR AUTÔNOMO	331209	4.02384
PROPRIETARIO DE MICROEMPRESAS	300734	3.65360
ANALISTA DE SISTEMAS	291202	3.53780

In [38]:

```
fig, (ax1, ax2) = plt.subplots(ncols=2, nrows=1, figsize=(25, 10))

plot = sns.barplot(x='qtde', y=df_investidores_profissao_f.index,
                ci=False, dodge=True, ax=ax1, orient='h',
                data=df_investidores_profissao_f, palette=['palevioletred'])

plot = sns.barplot(x='qtde', y=df_investidores_profissao_m.index,
                ci=False, dodge=True, ax=ax2, orient='h',
                data=df_investidores_profissao_m, palette=['cornflowerblue'])

ax1.tick_params(labelsize=13)
ax1.set_ylabel('')
ax1.set_xlabel('')
ax1.set_title('F', loc='right', fontsize=10)
ax1.ticklabel_format(style='plain', axis='x')

ax2.tick_params(labelsize=13)
ax2.set_ylabel('')
ax2.set_xlabel('')
ax2.set_title('M', loc='right', fontsize=10)
ax2.ticklabel_format(style='plain', axis='x')

fig.suptitle(
    '\nOs 10 perfis profissionais dos investidores divididos por gênero\n\n', fontsize=20)
plt.subplots_adjust(top=1, bottom=0.95)
plt.tight_layout()
plt.show()
```

Os 10 perfis profissionais dos investidores divididos por gênero