TCC PUC MG - Processamento e Tratamento de Dados

October 18, 2021

1 PUC - MG : Pós Graduação em Ciência de Dados e Big Data

ESTUDO E PREDIÇÃO DO COMPORTAMENTO DE INFRAÇÕES E ACIDENTES DE TRANSITO EM RODOVIAS FEDERAIS DO RIO DE JANEIRO

Aluna: Camila da Mata Rabelo Coleta das bases de infrações e acidentes nos sites:

#Infrações: https://www.gov.br/prf/pt-br/acesso-a-informacao/dados-abertos/dados-abertos-infracoes

#Acidentes: https://www.gov.br/prf/pt-br/acesso-a-informacao/dados-abertos/dados-abertos-acidentes

[1]:
```
#Carregar bibliotecas gerais
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime
from datetime import datetime
```

[14]:
```
#Os arquivos foram descompactados manualmente
#Renomeados todos para jan, fev, mar....
#Renomeadas as pastas para ficarem padronizadas anoXXXX
```

[2]:
```
#Lendo o arquivo de infrações dos meses por ano
#Ano 2017
jan2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\jan.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
fev2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\fev.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
mar2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\mar.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
abr2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\abr.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
```

```
mai2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\mai.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
jun2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\jun.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
jul2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\jul.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
ago2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\ago.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
set2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\set.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
out2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\out.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
nov2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\nov.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
dez2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2017\dez.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
```

[3]:
```
# Depois de verificar o shape de cada arquivo, vamos concatenar
ano2017 = pd.concat([jan2017, fev2017, mar2017, abr2017, mai2017, jun2017,␣
 ↪jul2017, ago2017, set2017, out2017, nov2017, dez2017], join = "inner")
ano2017.head(1)
```

[3]:
```
  dat_infracao tip_abordagem ind_assinou_auto ind_veiculo_estrangeiro  \
0   2017-01-01             S              NaN                        N

  ind_sentido_trafego uf_placa uf_infracao num_br_infracao num_km_infracao  \
0                 NaN       SP          SP             153              56

           nom_municipio  … enquadramento data_inicio_vigencia  \
0  SAO JOSE DO RIO PRETO  …        218 II           2016-11-01

  data_fim_vigencia med_realizada med_considerada exc_verificado especie  \
0               NaN         96.00           89.00          29.00     NaN

  nome_veiculo_marca nom_modelo_veiculo      hora
0                NaN           PARATI S  00:00:54

[1 rows x 22 columns]
```

[4]:
```
#Lendo o arquivo de infrações dos meses por ano
#Ano 2018
jan2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\jan.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
fev2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\fev.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
mar2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\mar.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
abr2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\abr.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
mai2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\mai.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
jun2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\jun.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
jul2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\jul.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
ago2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\ago.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
set2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\set.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
out2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\out.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
nov2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\nov.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
dez2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2018\dez.csv", sep=',', decimal = '.', encoding = "cp1252",␣
 ↪dtype = 'object')
```

[5]:
```
# Depois de avaliar o shape de cada um, vamos concatenar
ano2018 = pd.concat([jan2018, fev2018, mar2018, abr2018, mai2018, jun2018,␣
 ↪jul2018, ago2018, set2018, out2018, nov2018, dez2018], join = "inner")
ano2018.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7356159 entries, 0 to 727724
Data columns (total 22 columns):
```

```
 #   Column                    Dtype
---  ------                    -----
 0   dat_infracao              object
 1   tip_abordagem             object
 2   ind_assinou_auto          object
 3   ind_veiculo_estrangeiro   object
 4   ind_sentido_trafego       object
 5   uf_placa                  object
 6   uf_infracao               object
 7   num_br_infracao           object
 8   num_km_infracao           object
 9   nom_municipio             object
 10  cod_infracao              object
 11  descricao_abreviada       object
 12  enquadramento             object
 13  data_inicio_vigencia      object
 14  data_fim_vigencia         object
 15  med_realizada             object
 16  med_considerada           object
 17  exc_verificado            object
 18  especie                   object
 19  nome_veiculo_marca        object
 20  nom_modelo_veiculo        object
 21  hora                      object
dtypes: object(22)
memory usage: 1.3+ GB
```

[6]:
```
#Lendo o arquivo de infrações dos meses por ano
#Ano 2019
jan2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\jan.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
fev2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\fev.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
mar2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\mar.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
abr2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\abr.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
mai2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\mai.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
jun2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\jun.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
```

```
jul2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\jul.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
ago2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\ago.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
set2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\set.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
out2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\out.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
nov2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\nov.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
dez2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2019\dez.csv", sep = ';', encoding = "utf8", dtype =␣
 ↪'object')
# explicar q o encoding cp1252 n funcionou
```

[7]:
```
# Depois de verificar os shapes dos arquivos, vamos concatenar
ano2019 = pd.concat([jan2019, fev2019, mar2019, abr2019, mai2019, jun2019,␣
 ↪jul2019, ago2019, set2019, out2019, nov2019, dez2019], join = "inner")
ano2019.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5742110 entries, 0 to 492723
Data columns (total 22 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Número do Auto                  object
 1   Data da Infração (DD/MM/AAAA)   object
 2   Indicador de Abordagem          object
 3   Assinatura do Auto              object
 4   Indicador Veiculo Estrangeiro   object
 5   Sentido Trafego                 object
 6   UF Placa                        object
 7   UF Infração                     object
 8   BR Infração                     object
 9   Km Infração                     object
 10  Município                       object
 11  Código da Infração              object
 12  Descrição Abreviada Infração    object
 13  Enquadramento da Infração       object
 14  Início Vigência da Infração     object
 15  Fim Vigência Infração           object
 16  Medição Infração                object
```

```
 17  Descrição Especie Veículo      object
 18  Descrição Marca Veículo        object
 19  Hora Infração                  object
 20  Medição Considerada            object
 21  Excesso Verificado             object
dtypes: object(22)
memory usage: 1007.6+ MB
```

[8]:
```
#Lendo o arquivo de infrações dos meses por ano
#Ano 2020
jan2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\jan.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
fev2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\fev.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
mar2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\mar.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
abr2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\abr.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
mai2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\mai.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
jun2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\jun.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
jul2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\jul.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
ago2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\ago.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
set2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\set.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
out2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\out.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
nov2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\nov.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
dez2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Infracoes\ano2020\dez.csv", sep = ';', encoding = 'cp1252', dtype =␣
 ↪'object')
```

[9]:
```
# a base de Janeiro de 2020 tem uma coluna a mais do que as outras, por isso␣
 ↪precisaremos deletá-la antes de concatenar
# todos os meses de 2020
del jan2020['UF Placa']
#Agora sim, vamos unir todos os meses
ano2020 = pd.concat([jan2020, fev2020, mar2020, abr2020, mai2020, jun2020,␣
 ↪jul2020, ago2020, set2020, out2020, nov2020, dez2020], join = "inner")
ano2020.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5182537 entries, 0 to 500301
Data columns (total 22 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Número do Auto                  object
 1   Data da Infração (DD/MM/AAAA)   object
 2   Indicador de Abordagem          object
 3   Assinatura do Auto              object
 4   Indicador Veiculo Estrangeiro   object
 5   Sentido Trafego                 object
 6   UF Infração                     object
 7   BR Infração                     object
 8   Km Infração                     object
 9   Município                       object
 10  Código da Infração              object
 11  Descrição Abreviada Infração    object
 12  Enquadramento da Infração       object
 13  Início Vigência da Infração     object
 14  Fim Vigência Infração           object
 15  Medição Infração                object
 16  Descrição Especie Veículo       object
 17  Descrição Marca Veículo         object
 18  Hora Infração                   object
 19  Medição Considerada             object
 20  Excesso Verificado              object
 21  Qtd Infrações                   object
dtypes: object(22)
memory usage: 909.4+ MB
```

2 _____

3 Processamento e Tratamento das bases

4 _____

4.1 Higienização de ano2017 - Uniformização das colunas

[10]:
```
#Verificando as colunas no dataframe para  verificar quais serão deletadas
ano2017.columns.tolist()
```

[10]:
```
['dat_infracao',
 'tip_abordagem',
 'ind_assinou_auto',
 'ind_veiculo_estrangeiro',
 'ind_sentido_trafego',
 'uf_placa',
 'uf_infracao',
 'num_br_infracao',
 'num_km_infracao',
 'nom_municipio',
 'cod_infracao',
 'descricao_abreviada',
 'enquadramento',
 'data_inicio_vigencia',
 'data_fim_vigencia',
 'med_realizada',
 'med_considerada',
 'exc_verificado',
 'especie',
 'nome_veiculo_marca',
 'nom_modelo_veiculo',
 'hora']
```

[11]:
```
#Ano2017 - Eliminando colunas que não serão usadas
ano2017a = ano2017.drop(columns=['tip_abordagem',␣
 ↪'ind_assinou_auto','ind_veiculo_estrangeiro', 'ind_sentido_trafego',␣
 ↪'uf_placa','enquadramento', 'data_inicio_vigencia', 'data_fim_vigencia',␣
 ↪'med_realizada', 'med_considerada','especie', 'nome_veiculo_marca',␣
 ↪'nom_modelo_veiculo', 'exc_verificado',  'num_km_infracao',␣
 ↪'nom_municipio'],axis=1)
ano2017a.head()
```

[11]:

	dat_infracao	uf_infracao	num_br_infracao	cod_infracao	\
0	2017-01-01	SP	153	74630	

```
1   2017-01-01          ES              101         65992
2   2017-01-01          RJ              116         74550
3   2017-01-01          RJ              101         74550
4   2017-01-01          SP              116         74550
```

```
                                 descricao_abreviada      hora
0  Transitar em velocidade superior à máxima perm…  00:00:54
1  Conduzir o veículo registrado que não esteja d…  00:01:00
2  Transitar em velocidade superior à máxima perm…  00:02:08
3  Transitar em velocidade superior à máxima perm…  00:03:43
4  Transitar em velocidade superior à máxima perm…  00:03:50
```

[12]:
```
#Uma vez que já vimos que a coluna de hora está como 00:00:00 vamos nomeá-la de␣
↪uma forma provisória, pois vamos criar outra
ano2017a.columns = ['Data', 'UF',  'BR', 'Codigo', 'Descricao','Hora_infr_prov']
ano2017a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6023833 entries, 0 to 580894
Data columns (total 6 columns):
 #   Column          Dtype 
---  ------          ----- 
 0   Data            object
 1   UF              object
 2   BR              object
 3   Codigo          object
 4   Descricao       object
 5   Hora_infr_prov  object
dtypes: object(6)
memory usage: 321.7+ MB
```

[13]:
```
# Aqui criaremos outra coluna de hora, porém em formato inteiro
ano2017a['Hora'] = ano2017a['Hora_infr_prov'].str.split(':').str[0]
ano2017a.head(1)
#Agora deletaremos a coluna antiga
del ano2017a['Hora_infr_prov']
ano2017a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6023833 entries, 0 to 580894
Data columns (total 6 columns):
 #   Column     Dtype 
---  ------     ----- 
 0   Data       object
 1   UF         object
 2   BR         object
 3   Codigo     object
 4   Descricao  object
```

```
 5   Hora       object
dtypes: object(6)
memory usage: 321.7+ MB
```

[14]:
```
#Verificando colunas com valores NaN
ano2017a.isnull().sum()
```

[14]:
```
Data          0
UF            0
BR            0
Codigo        0
Descricao     0
Hora         14
dtype: int64
```

[15]:
```
#Deletaremos agora as linhas NaN presentes
ano2017b = ano2017a.dropna(subset = ['Hora'])
ano2017b.isnull().sum()
```

[15]:
```
Data         0
UF           0
BR           0
Codigo       0
Descricao    0
Hora         0
dtype: int64
```

[16]:
```
#Verificando o tipo de dados das colunas
ano2017b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6023819 entries, 0 to 580894
Data columns (total 6 columns):
 #   Column     Dtype
---  ------     -----
 0   Data       object
 1   UF         object
 2   BR         object
 3   Codigo     object
 4   Descricao  object
 5   Hora       object
dtypes: object(6)
memory usage: 321.7+ MB
```

[17]:
```
# Formatando tipo das colunas
# OBS. Vamos manter data ainda como object para fazermos a identificação de MÊS
ano2017c = ano2017b.astype({"BR": int, "Hora": int, "Codigo": int})
ano2017c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6023819 entries, 0 to 580894
Data columns (total 6 columns):
 #   Column     Dtype
---  ------     -----
 0   Data       object
 1   UF         object
 2   BR         int32
 3   Codigo     int32
 4   Descricao  object
 5   Hora       int32
dtypes: int32(3), object(3)
memory usage: 252.8+ MB
```

4.2 Higienização ano2018 - Uniformização das colunas

[18]:
```
#Verificando as colunas no dataframe para  verificar quais serão deletadas
ano2018.columns.tolist()
```

[18]:
```
['dat_infracao',
 'tip_abordagem',
 'ind_assinou_auto',
 'ind_veiculo_estrangeiro',
 'ind_sentido_trafego',
 'uf_placa',
 'uf_infracao',
 'num_br_infracao',
 'num_km_infracao',
 'nom_municipio',
 'cod_infracao',
 'descricao_abreviada',
 'enquadramento',
 'data_inicio_vigencia',
 'data_fim_vigencia',
 'med_realizada',
 'med_considerada',
 'exc_verificado',
 'especie',
 'nome_veiculo_marca',
 'nom_modelo_veiculo',
 'hora']
```

[19]:
```
#Ano2018 - Eliminando colunas que não serão usadas
```

```
ano2018a = ano2018.drop(columns=['tip_abordagem',␣
 ↪'ind_assinou_auto','ind_veiculo_estrangeiro', 'ind_sentido_trafego',␣
 ↪'uf_placa','enquadramento', 'data_inicio_vigencia', 'data_fim_vigencia',␣
 ↪'med_realizada', 'med_considerada','especie', 'nome_veiculo_marca',␣
 ↪'nom_modelo_veiculo', 'exc_verificado',  'num_km_infracao',␣
 ↪'nom_municipio'],axis=1)
ano2018a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7356159 entries, 0 to 727724
Data columns (total 6 columns):
 #   Column               Dtype
---  ------               -----
 0   dat_infracao         object
 1   uf_infracao          object
 2   num_br_infracao      object
 3   cod_infracao         object
 4   descricao_abreviada  object
 5   hora                 object
dtypes: object(6)
memory usage: 392.9+ MB
```

[20]:
```
#Uma vez que já vimos que a coluna de hora está como 00:00:00 vamos nomeá-la de␣
 ↪uma forma provisória, pois vamos criar outra
ano2018a.columns = ['Data', 'UF',  'BR', 'Codigo', 'Descricao','Hora_infr_prov']
ano2018a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7356159 entries, 0 to 727724
Data columns (total 6 columns):
 #   Column          Dtype
---  ------          -----
 0   Data            object
 1   UF              object
 2   BR              object
 3   Codigo          object
 4   Descricao       object
 5   Hora_infr_prov  object
dtypes: object(6)
memory usage: 392.9+ MB
```

[21]:
```
# Aqui criaremos outra coluna de hora, porém em formato inteiro
ano2018a['Hora'] = ano2018a['Hora_infr_prov'].str.split(':').str[0]
# Agora deletaremos a coluna antiga
del ano2018a['Hora_infr_prov']
ano2018a.head(1)
```

[21]:

	Data	UF	BR	Codigo	Descricao	Hora
0	2018-01-01	GO	153	56142	Parar na pista de rolamento das rodovias	00

[22]:
```
ano2018a.isnull().sum()
```

[22]:
```
Data          0
UF            0
BR            0
Codigo        0
Descricao     0
Hora         28
dtype: int64
```

[23]:
```
#Deletar linhas nulas da coluna de Hora
ano2018b = ano2018a.dropna(subset = ['Hora'])
ano2018b.isnull().sum()
#Não converti aqui o astype(int) porque isso torna o DF emm SERIE e aí não dá␣
 ↪pra agrupar
```

[23]:
```
Data         0
UF           0
BR           0
Codigo       0
Descricao    0
Hora         0
dtype: int64
```

[24]:
```
#Formatando tipo das colunas
# Vamos manter data ainda como object pra fazer a identificação de MÊS
ano2018c = ano2018b.astype({"BR": int, "Hora": int, "Codigo": int})
ano2018c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7356131 entries, 0 to 727724
Data columns (total 6 columns):
 #   Column     Dtype
---  ------     -----
 0   Data       object
 1   UF         object
 2   BR         int32
 3   Codigo     int32
 4   Descricao  object
 5   Hora       int32
dtypes: int32(3), object(3)
memory usage: 308.7+ MB
```

4.3 Higienizando ano2019 - Uniformização das colunas

[25]:
```
# Verificando colunas
ano2019.columns.tolist()
```

[25]:
```
['Número do Auto',
 'Data da Infração (DD/MM/AAAA)',
 'Indicador de Abordagem',
 'Assinatura do Auto',
 'Indicador Veiculo Estrangeiro',
 'Sentido Trafego',
 'UF Placa',
 'UF Infração',
 'BR Infração',
 'Km Infração',
 'Município',
 'Código da Infração',
 'Descrição Abreviada Infração',
 'Enquadramento da Infração',
 'Início Vigência da Infração',
 'Fim Vigência Infração',
 'Medição Infração',
 'Descrição Especie Veículo',
 'Descrição Marca Veículo',
 'Hora Infração',
 'Medição Considerada',
 'Excesso Verificado']
```

[26]:
```
#Ano2019 - Eliminando colunas que não serão usadas
ano2019a = ano2019.drop(columns=['Número do Auto', 'Indicador de Abordagem',␣
 ↪'Assinatura do Auto', 'Indicador Veiculo Estrangeiro', 'Sentido Trafego',␣
 ↪'UF Placa','Enquadramento da Infração', 'Início Vigência da Infração', 'Fim␣
 ↪Vigência Infração', 'Medição Infração', 'Descrição Especie Veículo',␣
 ↪'Descrição Marca Veículo','Medição Considerada', 'Excesso Verificado',  'Km␣
 ↪Infração', 'Município'],axis=1)
ano2019a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5742110 entries, 0 to 492723
Data columns (total 6 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Data da Infração (DD/MM/AAAA)   object
 1   UF Infração                     object
 2   BR Infração                     object
 3   Código da Infração              object
 4   Descrição Abreviada Infração    object
 5   Hora Infração                   object
```

```
dtypes: object(6)
memory usage: 306.7+ MB
```

[27]:
```
#Renomear para o padrão usado em 2017 e 2018
ano2019a.columns = ['Data', 'UF',  'BR', 'Codigo', 'Descricao', 'Hora']
ano2019a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5742110 entries, 0 to 492723
Data columns (total 6 columns):
 #   Column     Dtype 
---  ------     ----- 
 0   Data       object
 1   UF         object
 2   BR         object
 3   Codigo     object
 4   Descricao  object
 5   Hora       object
dtypes: object(6)
memory usage: 306.7+ MB
```

[28]:
```
# Verificando se há linhas nulas
ano2019a.isnull().sum()
```

[28]:
```
Data         0
UF           0
BR           0
Codigo       0
Descricao    0
Hora         0
dtype: int64
```

[29]:
```
# Não há linhas nulas
# Mas o processo será realizado da mesma forma para manter a lógica da␣
 ↪nomenclatura
ano2019b = ano2019a.dropna(subset = ['Hora'])
ano2019b.isnull().sum()

#Não converti aqui o astype(int) pq isso torna o DF emm SERIE e aí não dá pra␣
 ↪agrupar
```

[29]:
```
Data         0
UF           0
BR           0
Codigo       0
Descricao    0
Hora         0
dtype: int64
```

[30]:
```
#Formatando tipo das colunas
# Vamos manter data ainda como object pra fazer a identificação de MÊS
ano2019c = ano2019b.astype({"BR": int, "Hora": int, "Codigo": int})
ano2019c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5742110 entries, 0 to 492723
Data columns (total 6 columns):
 #   Column     Dtype
---  ------     -----
 0   Data       object
 1   UF         object
 2   BR         int32
 3   Codigo     int32
 4   Descricao  object
 5   Hora       int32
dtypes: int32(3), object(3)
memory usage: 240.9+ MB
```

4.4 Higienização de ano2020 - Uniformização das colunas

[31]:
```
# Verificando as colunas
ano2020.columns.tolist()
```

[31]:
```
['Número do Auto',
 'Data da Infração (DD/MM/AAAA)',
 'Indicador de Abordagem',
 'Assinatura do Auto',
 'Indicador Veiculo Estrangeiro',
 'Sentido Trafego',
 'UF Infração',
 'BR Infração',
 'Km Infração',
 'Município',
 'Código da Infração',
 'Descrição Abreviada Infração',
 'Enquadramento da Infração',
 'Início Vigência da Infração',
 'Fim Vigência Infração',
 'Medição Infração',
 'Descrição Especie Veículo',
 'Descrição Marca Veículo',
 'Hora Infração',
 'Medição Considerada',
 'Excesso Verificado',
 'Qtd Infrações']
```

[32]:
```
#Ano2020 - Eliminando colunas que não serão usadas
ano2020a = ano2020.drop(columns=['Número do Auto', 'Indicador de Abordagem',␣
 ↪'Assinatura do Auto', 'Indicador Veiculo Estrangeiro', 'Sentido Trafego',␣
 ↪'Enquadramento da Infração', 'Início Vigência da Infração', 'Fim Vigência␣
 ↪Infração', 'Medição Infração', 'Descrição Especie Veículo', 'Descrição Marca␣
 ↪Veículo','Medição Considerada','Qtd Infrações',  'Excesso Verificado',  'Km␣
 ↪Infração', 'Município'],axis=1)
ano2020a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5182537 entries, 0 to 500301
Data columns (total 6 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Data da Infração (DD/MM/AAAA)  object
 1   UF Infração                     object
 2   BR Infração                     object
 3   Código da Infração              object
 4   Descrição Abreviada Infração   object
 5   Hora Infração                   object
dtypes: object(6)
memory usage: 276.8+ MB
```

[33]:
```
# Renomear para o padrão usado
ano2020a.columns = ['Data', 'UF',  'BR', 'Codigo', 'Descricao', 'Hora']
ano2020a.head(1)
```

[33]:

	Data	UF	BR	Codigo	Descricao	Hora
0	2020-01-16	RS	290	58350	Desobedecer às ordens emanadas da autorid comp…	0

[34]:
```
# Verificando se há linhas nulas
ano2020a.isnull().sum()
```

[34]:
```
Data        0
UF          0
BR          0
Codigo      0
Descricao   0
Hora        0
dtype: int64
```

[35]:
```
# Não há linhas nulas, mas procederemos com o comando mesmo assim, para␣
 ↪mantermos o padrão de nomes de dataframes
ano2020b = ano2020a.dropna(subset = ['Hora'])
ano2020b.info()
```

```
#Não converti aqui o astype(int) pq isso torna o DF emm SERIE e aí não dá pra␣
↪agrupar
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5182537 entries, 0 to 500301
Data columns (total 6 columns):
 #   Column     Dtype
---  ------     -----
 0   Data       object
 1   UF         object
 2   BR         object
 3   Codigo     object
 4   Descricao  object
 5   Hora       object
dtypes: object(6)
memory usage: 276.8+ MB
```

[36]:
```
#Formatando tipo das colunas
# Vamos manter Data ainda como object pra fazer a identificação de MÊS
ano2020c = ano2020b.astype({"BR": int, "Hora": int, "Codigo": int})
ano2020c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5182537 entries, 0 to 500301
Data columns (total 6 columns):
 #   Column     Dtype
---  ------     -----
 0   Data       object
 1   UF         object
 2   BR         int32
 3   Codigo     int32
 4   Descricao  object
 5   Hora       int32
dtypes: int32(3), object(3)
memory usage: 217.5+ MB
```

5

6 INFRAÇÕES POR UF

7

7.1 Ranking e gráfico de Infrações por UF - 2017

[37]:
```
# Agrupando infrações do ano por UF
ano2017_ufs = ano2017c.groupby(['UF']).size().sort_values(ascending=False)
ano2017_ufs = ano2017_ufs.rename("Total de Infrações por ano")
ano2017_ufs.rename_axis("Total de infrações por ano")
```

```
ano2017_ufs
```

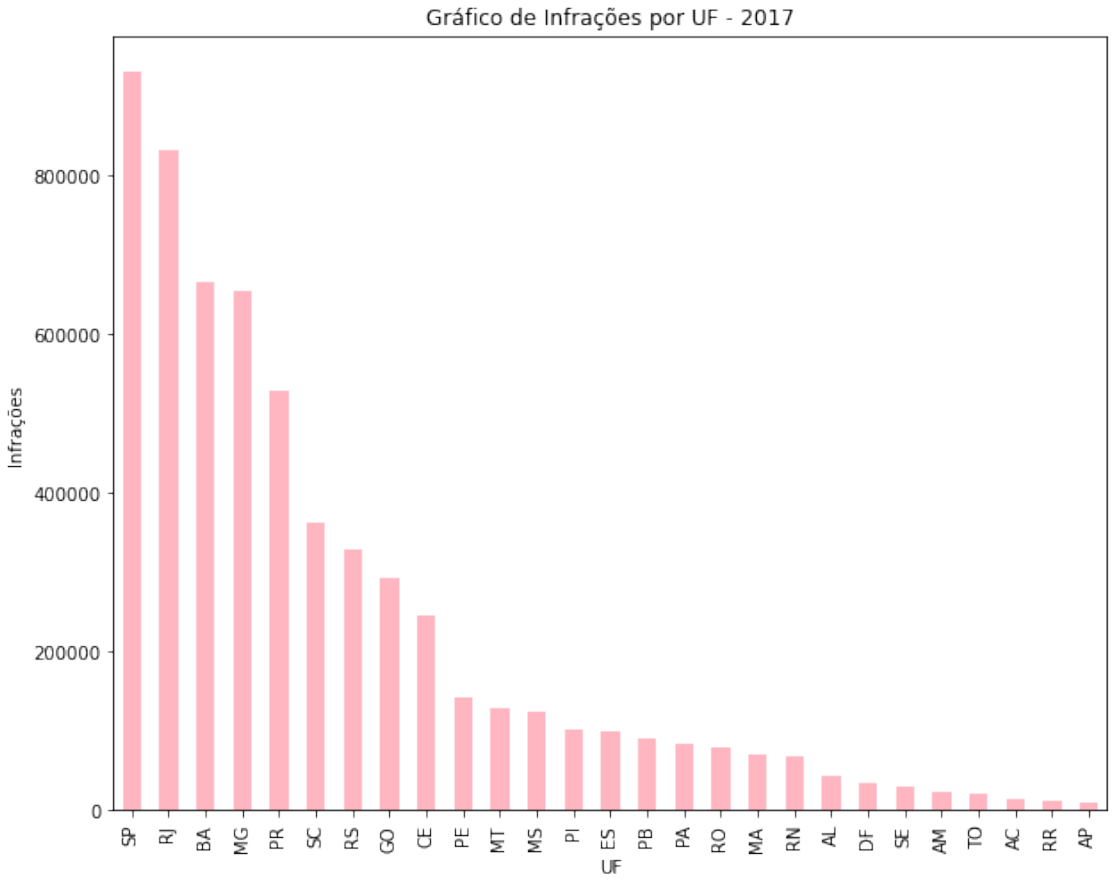
[37]:
```
UF
SP    930622
RJ    832436
BA    667166
MG    653881
PR    528561
SC    362163
RS    328716
GO    291935
CE    245597
PE    143129
MT    130043
MS    123643
PI    101387
ES    100495
PB     91151
PA     83947
RO     80156
MA     70645
RN     67464
AL     43836
DF     35066
SE     30898
AM     22659
TO     22088
AC     13968
RR     12843
AP      9324
Name: Total de Infrações por ano, dtype: int64
```

[38]:
```
type(ano2017_ufs)
```

[38]:
```
pandas.core.series.Series
```

[39]:
```
# Gráfico de Infrações por UF - 2017
ano2017_ufs.plot.bar(figsize=(10,8), color = 'lightpink', title = "Gráfico de␣
↪Infrações por UF - 2017", ylabel = 'Infrações')
```

[39]:
```
<AxesSubplot:title={'center':'Gráfico de Infrações por UF - 2017'}, xlabel='UF',
ylabel='Infrações'>
```



7.2 Ranking e gráfico de Infrações por UF - 2018

[40]:
```
# Agrupando infrações do ano por UF
ano2018_ufs = ano2018c.groupby(['UF']).size().sort_values(ascending=False)
ano2018_ufs = ano2018_ufs.rename("Total de Infrações por ano")
ano2018_ufs
```

[40]:
```
UF
MG    1206324
RJ    1127572
SP     799175
BA     714759
PR     525525
SC     428671
GO     346295
DF     331487
RS     289684
MS     264469
CE     261135
```

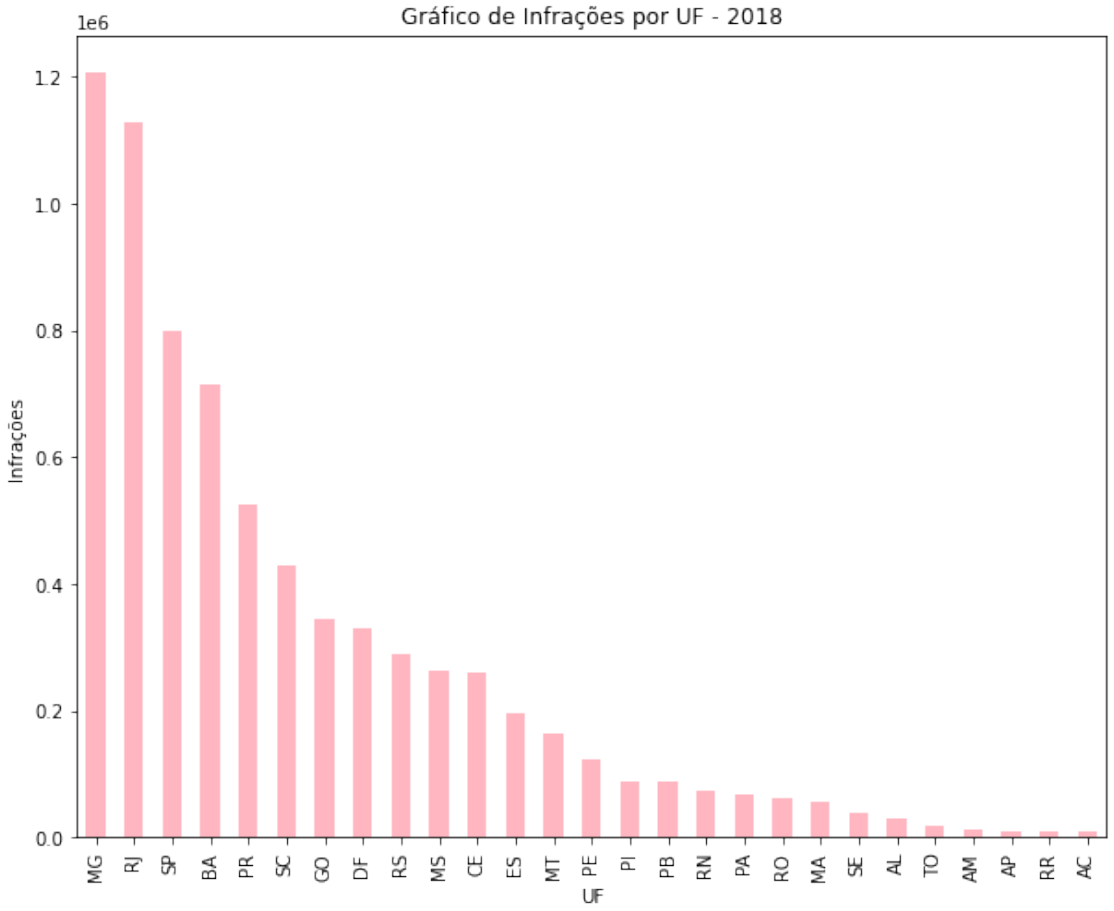
```
ES     196197
MT     163691
PE     124469
PI      88519
PB      87599
RN      74317
PA      67740
RO      63857
MA      56837
SE      40125
AL      31556
TO      19749
AM      14123
AP      11140
RR      10711
AC      10405
Name: Total de Infrações por ano, dtype: int64
```

[41]:
```
type(ano2018_ufs)
```

[41]:
```
pandas.core.series.Series
```

[42]:
```
# Gráfico de Infrações por UF - 2018
ano2018_ufs.plot.bar(figsize=(10,8), color = 'lightpink', title = "Gráfico de␣
↪Infrações por UF - 2018", ylabel = 'Infrações')
```

[42]:
```
<AxesSubplot:title={'center':'Gráfico de Infrações por UF - 2018'}, xlabel='UF',
ylabel='Infrações'>
```



7.3 Ranking e gráfico de Infrações por UF - 2019

[43]:
```
# Agrupando infrações do ano por UF
ano2019_ufs = ano2019c.groupby(['UF']).size().sort_values(ascending=False)
ano2019_ufs = ano2019_ufs.rename("Total de Infrações por ano")
ano2019_ufs
```

[43]:
```
UF
RJ    765807
SP    631019
MG    611677
BA    594990
PR    364895
SC    357031
MS    354123
GO    304334
ES    233648
MT    210832
```

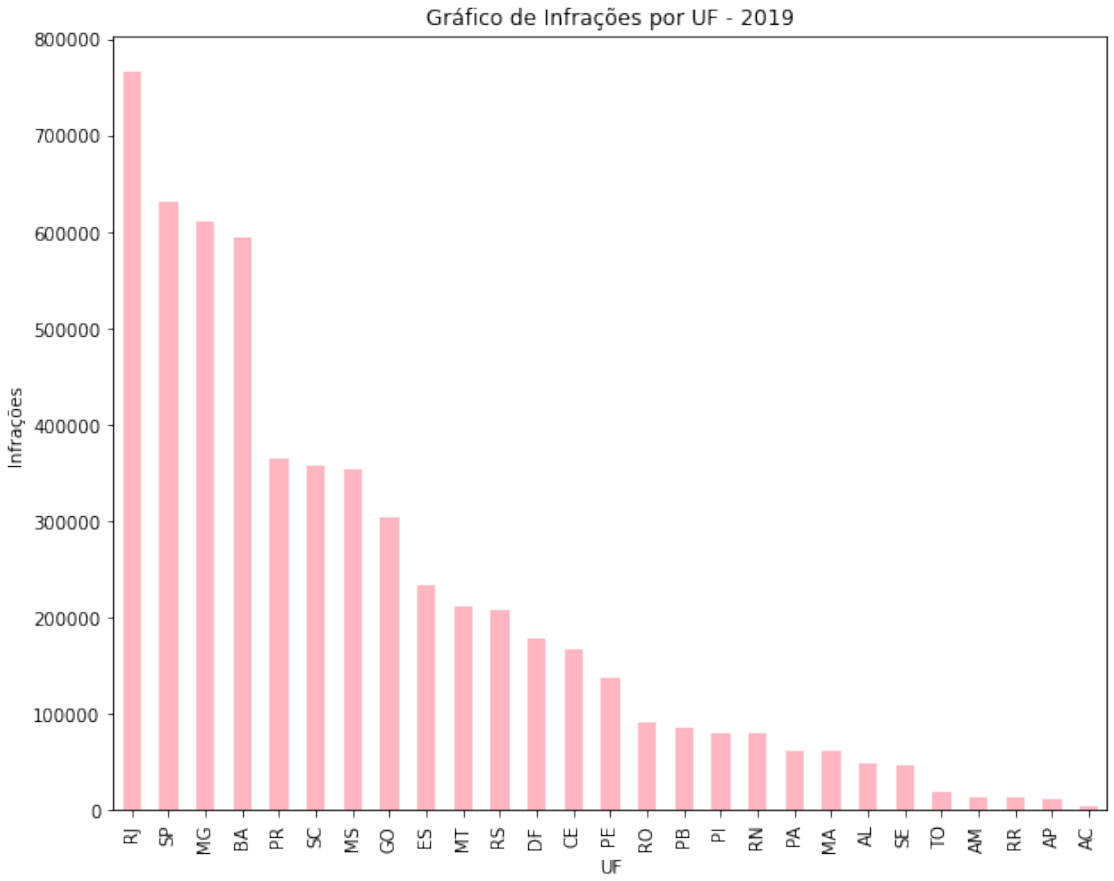
```
RS    208267
DF    177956
CE    167252
PE    138120
RO     91157
PB     85457
PI     80995
RN     79933
PA     62606
MA     61561
AL     48694
SE     47628
TO     19468
AM     14615
RR     13410
AP     11232
AC      5403
Name: Total de Infrações por ano, dtype: int64
```

[44]:
```
type(ano2019_ufs)
```

[44]:
```
pandas.core.series.Series
```

[45]:
```
# Gráfico de Infrações por UF - 2019
ano2019_ufs.plot.bar(figsize=(10,8), color = 'lightpink', title = 'Gráfico de␣
↪Infrações por UF - 2019', ylabel = 'Infrações')
```

[45]:
```
<AxesSubplot:title={'center':'Gráfico de Infrações por UF - 2019'}, xlabel='UF',
ylabel='Infrações'>
```



7.4 Ranking e gráfico de Infrações por UF - 2020

[46]:
```
# Agrupando infrações do ano por UF
ano2020_ufs = ano2020c.groupby(['UF']).size().sort_values(ascending=False)
ano2020_ufs = ano2020_ufs.rename("Total de Infrações por ano")
ano2020_ufs
```

[46]:
```
UF
RJ    919184
MG    665769
SP    475498
BA    403442
MS    399371
PR    338031
SC    318324
GO    221705
MT    172090
ES    162210
RS    147056
```

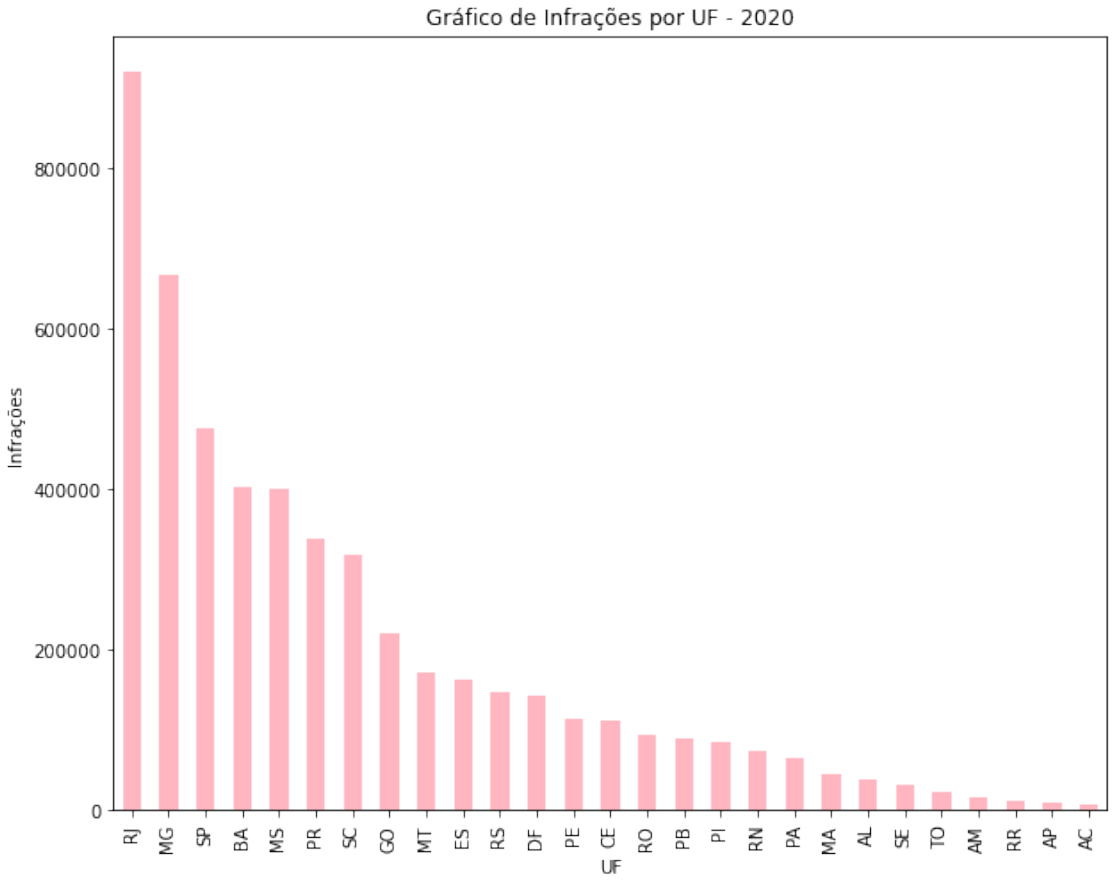
```
DF    141893
PE    114052
CE    111986
RO     94962
PB     89023
PI     84293
RN     74617
PA     64343
MA     44799
AL     38972
SE     31730
TO     22114
AM     17333
RR     12117
AP      9877
AC      7746
Name: Total de Infrações por ano, dtype: int64
```

[47]:
```
type(ano2020_ufs)
```

[47]:
```
pandas.core.series.Series
```

[48]:
```
# Gráfico de Infrações por UF - 2020
ano2020_ufs.plot.bar(figsize=(10,8), color = 'lightpink', title = 'Gráfico de␣
↪Infrações por UF - 2020', ylabel = 'Infrações')
```

[48]:
```
<AxesSubplot:title={'center':'Gráfico de Infrações por UF - 2020'}, xlabel='UF',
ylabel='Infrações'>
```



8

9 DATASET ACIDENTES

10

[49]:
```
#Lendo o arquivo de infrações dos meses por ano

acid2017 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Acidentes\acidentes2017_todas_causas_tipos.csv", sep=';', decimal = '.',␣
 ↪encoding = 'cp1252', dtype = 'object')
acid2018 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Acidentes\acidentes2018_todas_causas_tipos.csv", sep=';', decimal = '.',␣
 ↪encoding = 'cp1252', dtype = 'object')
acid2019 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
 ↪TCC\Acidentes\acidentes2019_todas_causas_tipos.csv", sep=';', decimal = '.',␣
 ↪encoding = 'cp1252', dtype = 'object')
```

```
acid2020 = pd.read_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
↪TCC\Acidentes\acidentes2020_todas_causas_tipos.csv", sep=';', decimal = '.',␣
↪encoding = 'utf8', dtype = 'object')
```

[50]:
```
# Usando cp1252 na planilha de 2020 deu erro de
#"UnicodeDecodeError: 'charmap' codec can't decode byte 0x81 in position 4165:␣
↪character maps to <undefined>"
# por isso decidi usar outro encoding.
```

11 HIGIENIZAÇÃO ACIDENTES

12 Ano 2017

[51]:
```
# Avaliando a base
acid2017.columns.tolist()
```

[51]:
```
['id',
 'pesid',
 'data_inversa',
 'dia_semana',
 'horario',
 'uf',
 'br',
 'km',
 'municipio',
 'causa_principal',
 'causa_acidente',
 'ordem_tipo_acidente',
 'tipo_acidente',
 'classificacao_acidente',
 'fase_dia',
 'sentido_via',
 'condicao_metereologica',
 'tipo_pista',
 'tracado_via',
 'uso_solo',
 'id_veiculo',
 'tipo_veiculo',
 'marca',
 'ano_fabricacao_veiculo',
 'tipo_envolvido',
 'estado_fisico',
 'idade',
 'sexo',
 'ilesos',
 'feridos_leves',
```

```
 'feridos_graves',
 'mortos',
 'latitude',
 'longitude',
 'regional',
 'delegacia',
 'uop']
```

[52]:
```
#Acid2017 - Eliminando colunas que não serão usadas
acid2017a = acid2017.drop(columns=['id', 'pesid', 'causa_principal',␣
 ↪'ordem_tipo_acidente','fase_dia', 'sentido_via',␣
 ↪'condicao_metereologica','tipo_pista','tracado_via', 'uso_solo',␣
 ↪'id_veiculo', 'tipo_veiculo', 'marca', 'ano_fabricacao_veiculo',␣
 ↪'tipo_envolvido', 'estado_fisico', 'idade', 'sexo','regional', 'delegacia',␣
 ↪'uop','latitude','longitude', 'municipio', 'km'],axis=1)
acid2017a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349067 entries, 0 to 349066
Data columns (total 12 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   data_inversa            349067 non-null  object
 1   dia_semana              349067 non-null  object
 2   horario                 349067 non-null  object
 3   uf                      349067 non-null  object
 4   br                      348546 non-null  object
 5   causa_acidente          349067 non-null  object
 6   tipo_acidente           349067 non-null  object
 7   classificacao_acidente  349067 non-null  object
 8   ilesos                  349067 non-null  object
 9   feridos_leves           349067 non-null  object
 10  feridos_graves          349067 non-null  object
 11  mortos                  349067 non-null  object
dtypes: object(12)
memory usage: 32.0+ MB
```

[53]:
```
# Criando coluna de hora inteira
acid2017a['Hora'] = acid2017a['horario'].str.split(':').str[0]

#Deletando a coluna antiga
del acid2017a['horario']
acid2017a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349067 entries, 0 to 349066
Data columns (total 12 columns):
 #   Column                  Non-Null Count   Dtype
```

```
---  ------                    --------------   -----
 0   data_inversa              349067 non-null  object
 1   dia_semana                349067 non-null  object
 2   uf                        349067 non-null  object
 3   br                        348546 non-null  object
 4   causa_acidente            349067 non-null  object
 5   tipo_acidente             349067 non-null  object
 6   classificacao_acidente    349067 non-null  object
 7   ilesos                    349067 non-null  object
 8   feridos_leves             349067 non-null  object
 9   feridos_graves            349067 non-null  object
 10  mortos                    349067 non-null  object
 11  Hora                      349067 non-null  object
dtypes: object(12)
memory usage: 32.0+ MB
```

[54]:
```
# Renomear a base para o padrão usado
acid2017a.columns = ['Data', 'Dia_semana', 'UF',  'BR', 'Causa',␣
 ↪'Tipo_acidente','Classificacao', 'Ilesos', 'Feridos_leves',␣
 ↪'Feridos_graves', 'Obitos', 'Hora']
acid2017a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349067 entries, 0 to 349066
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Data            349067 non-null  object
 1   Dia_semana      349067 non-null  object
 2   UF              349067 non-null  object
 3   BR              348546 non-null  object
 4   Causa           349067 non-null  object
 5   Tipo_acidente   349067 non-null  object
 6   Classificacao   349067 non-null  object
 7   Ilesos          349067 non-null  object
 8   Feridos_leves   349067 non-null  object
 9   Feridos_graves  349067 non-null  object
 10  Obitos          349067 non-null  object
 11  Hora            349067 non-null  object
dtypes: object(12)
memory usage: 32.0+ MB
```

[55]:
```
# Verificando se há linhas nulas
acid2017a.isnull().sum()
```

[55]:
```
Data              0
Dia_semana        0
UF                0
```

```
BR                521
Causa               0
Tipo_acidente       0
Classificacao       0
Ilesos              0
Feridos_leves       0
Feridos_graves      0
Obitos              0
Hora                0
dtype: int64
```

[56]:
```
#Deletaremos agora as linhas nulas
acid2017b = acid2017a.dropna(subset = ['BR'])
acid2017b.isnull().sum()
```

[56]:
```
Data              0
Dia_semana        0
UF                0
BR                0
Causa             0
Tipo_acidente     0
Classificacao     0
Ilesos            0
Feridos_leves     0
Feridos_graves    0
Obitos            0
Hora              0
dtype: int64
```

[57]:
```
# Formatando tipo das colunas
# Vamos manter data ainda como object pra fazer a identificação de MÊS

acid2017c = acid2017b.astype({"BR": int, "Hora": int, 'Ilesos': int,␣
 ↪'Feridos_leves' : int, 'Feridos_graves': int, 'Obitos' : int})
acid2017c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 348546 entries, 0 to 349066
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Data            348546 non-null  object
 1   Dia_semana      348546 non-null  object
 2   UF              348546 non-null  object
 3   BR              348546 non-null  int32
 4   Causa           348546 non-null  object
 5   Tipo_acidente   348546 non-null  object
 6   Classificacao   348546 non-null  object
```

```
 7   Ilesos          348546 non-null  int32
 8   Feridos_leves   348546 non-null  int32
 9   Feridos_graves  348546 non-null  int32
 10  Obitos          348546 non-null  int32
 11  Hora            348546 non-null  int32
dtypes: int32(6), object(6)
memory usage: 26.6+ MB
```

13 ANO 2018

[58]:
```
# Verificando a base
acid2018.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 324809 entries, 0 to 324808
Data columns (total 37 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   id                      324809 non-null  object
 1   pesid                   299277 non-null  object
 2   data_inversa            324809 non-null  object
 3   dia_semana              324809 non-null  object
 4   horario                 324809 non-null  object
 5   uf                      324809 non-null  object
 6   br                      324257 non-null  object
 7   km                      324257 non-null  object
 8   municipio               324809 non-null  object
 9   causa_principal         324809 non-null  object
 10  causa_acidente          324809 non-null  object
 11  ordem_tipo_acidente     324754 non-null  object
 12  tipo_acidente           324754 non-null  object
 13  classificacao_acidente  324809 non-null  object
 14  fase_dia                324809 non-null  object
 15  sentido_via             324809 non-null  object
 16  condicao_metereologica  324809 non-null  object
 17  tipo_pista              324809 non-null  object
 18  tracado_via             324809 non-null  object
 19  uso_solo                324809 non-null  object
 20  id_veiculo              324809 non-null  object
 21  tipo_veiculo            324809 non-null  object
 22  marca                   311669 non-null  object
 23  ano_fabricacao_veiculo  309128 non-null  object
 24  tipo_envolvido          324809 non-null  object
 25  estado_fisico           324809 non-null  object
 26  idade                   263305 non-null  object
 27  sexo                    324809 non-null  object
 28  ilesos                  324809 non-null  object
```

```
 29  feridos_leves           324809 non-null  object
 30  feridos_graves          324809 non-null  object
 31  mortos                  324809 non-null  object
 32  latitude                324809 non-null  object
 33  longitude               324809 non-null  object
 34  regional                324809 non-null  object
 35  delegacia               324809 non-null  object
 36  uop                     308467 non-null  object
dtypes: object(37)
memory usage: 91.7+ MB
```

[59]:
```
#Acid2018 - Eliminando colunas que não serão usadas
acid2018a = acid2018.drop(columns=['id', 'pesid', 'causa_principal',␣
 ↪'ordem_tipo_acidente','fase_dia', 'sentido_via',␣
 ↪'condicao_metereologica','tipo_pista','tracado_via', 'uso_solo',␣
 ↪'id_veiculo', 'tipo_veiculo', 'marca', 'ano_fabricacao_veiculo',␣
 ↪'tipo_envolvido', 'estado_fisico', 'idade', 'sexo','regional', 'delegacia',␣
 ↪'uop', 'km',  'latitude', 'longitude', 'municipio'] ,axis=1)
acid2018a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 324809 entries, 0 to 324808
Data columns (total 12 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   data_inversa            324809 non-null  object
 1   dia_semana              324809 non-null  object
 2   horario                 324809 non-null  object
 3   uf                      324809 non-null  object
 4   br                      324257 non-null  object
 5   causa_acidente          324809 non-null  object
 6   tipo_acidente           324754 non-null  object
 7   classificacao_acidente  324809 non-null  object
 8   ilesos                  324809 non-null  object
 9   feridos_leves           324809 non-null  object
 10  feridos_graves          324809 non-null  object
 11  mortos                  324809 non-null  object
dtypes: object(12)
memory usage: 29.7+ MB
```

[60]:
```
# Criando coluna de hora inteira
acid2018a['Hora'] = acid2018a['horario'].str.split(':').str[0]

#Deletando horario
del acid2018a['horario']
acid2018a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 324809 entries, 0 to 324808
```

```
Data columns (total 12 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   data_inversa            324809 non-null  object
 1   dia_semana              324809 non-null  object
 2   uf                      324809 non-null  object
 3   br                      324257 non-null  object
 4   causa_acidente          324809 non-null  object
 5   tipo_acidente           324754 non-null  object
 6   classificacao_acidente  324809 non-null  object
 7   ilesos                  324809 non-null  object
 8   feridos_leves           324809 non-null  object
 9   feridos_graves          324809 non-null  object
 10  mortos                  324809 non-null  object
 11  Hora                    324809 non-null  object
dtypes: object(12)
memory usage: 29.7+ MB
```

[61]:
```
# Renomeando colunas para o padrão usado
acid2018a.columns = ['Data', 'Dia_semana', 'UF',  'BR', 'Causa',␣
 ↪'Tipo_acidente','Classificacao', 'Ilesos', 'Feridos_leves',␣
 ↪'Feridos_graves', 'Obitos', 'Hora']
acid2018a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 324809 entries, 0 to 324808
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Data            324809 non-null  object
 1   Dia_semana      324809 non-null  object
 2   UF              324809 non-null  object
 3   BR              324257 non-null  object
 4   Causa           324809 non-null  object
 5   Tipo_acidente   324754 non-null  object
 6   Classificacao   324809 non-null  object
 7   Ilesos          324809 non-null  object
 8   Feridos_leves   324809 non-null  object
 9   Feridos_graves  324809 non-null  object
 10  Obitos          324809 non-null  object
 11  Hora            324809 non-null  object
dtypes: object(12)
memory usage: 29.7+ MB
```

[62]:
```
# Verificando se há linhas nulas
acid2018a.isnull().sum()
```

[62]:
```
Data               0
Dia_semana         0
UF                 0
BR               552
Causa              0
Tipo_acidente     55
Classificacao      0
Ilesos             0
Feridos_leves      0
Feridos_graves     0
Obitos             0
Hora               0
dtype: int64
```

[63]:
```
# Deletaremos agora as linhas nulas
acid2018b = acid2018a.dropna(subset = ['BR'])
acid2018b = acid2018b.dropna(subset = ['Tipo_acidente'])
acid2018b.isnull().sum()
```

[63]:
```
Data             0
Dia_semana       0
UF               0
BR               0
Causa            0
Tipo_acidente    0
Classificacao    0
Ilesos           0
Feridos_leves    0
Feridos_graves   0
Obitos           0
Hora             0
dtype: int64
```

[64]:
```
#Formatando tipo das colunas
# Vamos manter data ainda como object pra fazer a identificação de MÊS
acid2018c = acid2018b.astype({"BR": int, "Hora": int, 'Ilesos': int,␣
 ↪'Feridos_leves' : int, 'Feridos_graves': int, 'Obitos' : int})
acid2018c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 324202 entries, 0 to 324808
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Data            324202 non-null  object
 1   Dia_semana      324202 non-null  object
 2   UF              324202 non-null  object
 3   BR              324202 non-null  int32
```

```
 4   Causa           324202 non-null  object
 5   Tipo_acidente   324202 non-null  object
 6   Classificacao   324202 non-null  object
 7   Ilesos          324202 non-null  int32
 8   Feridos_leves   324202 non-null  int32
 9   Feridos_graves  324202 non-null  int32
 10  Obitos          324202 non-null  int32
 11  Hora            324202 non-null  int32
dtypes: int32(6), object(6)
memory usage: 24.7+ MB
```

14 Ano 2019

[65]:
```
# Verificando a base
acid2019.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 331666 entries, 0 to 331665
Data columns (total 37 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   id                      331666 non-null  object
 1   pesid                   307223 non-null  object
 2   data_inversa            331666 non-null  object
 3   dia_semana              331666 non-null  object
 4   horario                 331666 non-null  object
 5   uf                      331666 non-null  object
 6   br                      331291 non-null  object
 7   km                      331291 non-null  object
 8   municipio               331666 non-null  object
 9   causa_principal         331666 non-null  object
 10  causa_acidente          331666 non-null  object
 11  ordem_tipo_acidente     331626 non-null  object
 12  tipo_acidente           331626 non-null  object
 13  classificacao_acidente  331666 non-null  object
 14  fase_dia                331666 non-null  object
 15  sentido_via             331666 non-null  object
 16  condicao_metereologica  331666 non-null  object
 17  tipo_pista              331666 non-null  object
 18  tracado_via             331666 non-null  object
 19  uso_solo                331666 non-null  object
 20  id_veiculo              331666 non-null  object
 21  tipo_veiculo            331666 non-null  object
 22  marca                   317602 non-null  object
 23  ano_fabricacao_veiculo  314393 non-null  object
 24  tipo_envolvido          331666 non-null  object
 25  estado_fisico           331666 non-null  object
```

```
 26  idade                     269798 non-null  object
 27  sexo                      331666 non-null  object
 28  ilesos                    331666 non-null  object
 29  feridos_leves             331666 non-null  object
 30  feridos_graves            331666 non-null  object
 31  mortos                    331666 non-null  object
 32  latitude                  331666 non-null  object
 33  longitude                 331666 non-null  object
 34  regional                  331666 non-null  object
 35  delegacia                 331666 non-null  object
 36  uop                       314468 non-null  object
dtypes: object(37)
memory usage: 93.6+ MB
```

[66]:
```
#Acid2019 - Eliminando colunas que não serão usadas
acid2019a = acid2019.drop(columns=['id', 'pesid', 'km', 'causa_principal',␣
 ↪'ordem_tipo_acidente','fase_dia', 'sentido_via',␣
 ↪'condicao_metereologica','tipo_pista','tracado_via', 'uso_solo',␣
 ↪'id_veiculo', 'tipo_veiculo', 'marca', 'ano_fabricacao_veiculo',␣
 ↪'tipo_envolvido', 'estado_fisico', 'idade', 'sexo','regional', 'delegacia',␣
 ↪'uop',  'latitude','longitude', 'municipio'],axis=1)
acid2019a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 331666 entries, 0 to 331665
Data columns (total 12 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   data_inversa            331666 non-null  object
 1   dia_semana              331666 non-null  object
 2   horario                 331666 non-null  object
 3   uf                      331666 non-null  object
 4   br                      331291 non-null  object
 5   causa_acidente          331666 non-null  object
 6   tipo_acidente           331626 non-null  object
 7   classificacao_acidente  331666 non-null  object
 8   ilesos                  331666 non-null  object
 9   feridos_leves           331666 non-null  object
 10  feridos_graves          331666 non-null  object
 11  mortos                  331666 non-null  object
dtypes: object(12)
memory usage: 30.4+ MB
```

[67]:
```
# Criando coluna de hora inteira
acid2019a['Hora'] = acid2019a['horario'].str.split(':').str[0]

#Deletando a coluna antiga de horário
del acid2019a['horario']
```

```
acid2019a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 331666 entries, 0 to 331665
Data columns (total 12 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   data_inversa            331666 non-null  object
 1   dia_semana              331666 non-null  object
 2   uf                      331666 non-null  object
 3   br                      331291 non-null  object
 4   causa_acidente          331666 non-null  object
 5   tipo_acidente           331626 non-null  object
 6   classificacao_acidente  331666 non-null  object
 7   ilesos                  331666 non-null  object
 8   feridos_leves           331666 non-null  object
 9   feridos_graves          331666 non-null  object
 10  mortos                  331666 non-null  object
 11  Hora                    331666 non-null  object
dtypes: object(12)
memory usage: 30.4+ MB
```

[68]:
```
# Renomeando para o padrão usado
acid2019a.columns = ['Data', 'Dia_semana', 'UF',  'BR', 'Causa',␣
 ↪'Tipo_acidente','Classificacao', 'Ilesos', 'Feridos_leves',␣
 ↪'Feridos_graves', 'Obitos', 'Hora']
acid2019a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 331666 entries, 0 to 331665
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Data            331666 non-null  object
 1   Dia_semana      331666 non-null  object
 2   UF              331666 non-null  object
 3   BR              331291 non-null  object
 4   Causa           331666 non-null  object
 5   Tipo_acidente   331626 non-null  object
 6   Classificacao   331666 non-null  object
 7   Ilesos          331666 non-null  object
 8   Feridos_leves   331666 non-null  object
 9   Feridos_graves  331666 non-null  object
 10  Obitos          331666 non-null  object
 11  Hora            331666 non-null  object
dtypes: object(12)
memory usage: 30.4+ MB
```

[69]:
```
# Verificando se há linhas nulas
acid2019a.isnull().sum()
```

[69]:
```
Data               0
Dia_semana         0
UF                 0
BR               375
Causa              0
Tipo_acidente     40
Classificacao      0
Ilesos             0
Feridos_leves      0
Feridos_graves     0
Obitos             0
Hora               0
dtype: int64
```

[70]:
```
#Deletaremos as linhas nulas
acid2019b = acid2019a.dropna(subset = ['BR'])
acid2019b = acid2019b.dropna(subset = ['Tipo_acidente'])
acid2019b.isnull().sum()
```

[70]:
```
Data             0
Dia_semana       0
UF               0
BR               0
Causa            0
Tipo_acidente    0
Classificacao    0
Ilesos           0
Feridos_leves    0
Feridos_graves   0
Obitos           0
Hora             0
dtype: int64
```

[71]:
```
#Formatando tipo das colunas
# Vamos manter data ainda como object pra fazer a identificação de MÊS
acid2019c = acid2019b.astype({"BR": int, "Hora": int, 'Ilesos': int,␣
 ↪'Feridos_leves' : int, 'Feridos_graves': int, 'Obitos' : int})
acid2019c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 331251 entries, 0 to 331665
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Data            331251 non-null  object
```

```
 1   Dia_semana      331251 non-null  object
 2   UF              331251 non-null  object
 3   BR              331251 non-null  int32
 4   Causa           331251 non-null  object
 5   Tipo_acidente   331251 non-null  object
 6   Classificacao   331251 non-null  object
 7   Ilesos          331251 non-null  int32
 8   Feridos_leves   331251 non-null  int32
 9   Feridos_graves  331251 non-null  int32
 10  Obitos          331251 non-null  int32
 11  Hora            331251 non-null  int32
dtypes: int32(6), object(6)
memory usage: 25.3+ MB
```

14.1 Ano 2020

[72]:
```
# Verificando a base
acid2020.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 394482 entries, 0 to 394481
Data columns (total 37 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   id                      394482 non-null  object
 1   pesid                   361405 non-null  object
 2   data_inversa            394482 non-null  object
 3   dia_semana              394482 non-null  object
 4   horario                 394482 non-null  object
 5   uf                      394482 non-null  object
 6   br                      393468 non-null  object
 7   km                      393468 non-null  object
 8   municipio               394482 non-null  object
 9   causa_principal         394482 non-null  object
 10  causa_acidente          394482 non-null  object
 11  ordem_tipo_acidente     394481 non-null  object
 12  tipo_acidente           394481 non-null  object
 13  classificacao_acidente  394481 non-null  object
 14  fase_dia                394482 non-null  object
 15  sentido_via             394482 non-null  object
 16  condicao_metereologica  394482 non-null  object
 17  tipo_pista              394482 non-null  object
 18  tracado_via             394482 non-null  object
 19  uso_solo                394482 non-null  object
 20  id_veiculo              394481 non-null  object
 21  tipo_veiculo            394482 non-null  object
 22  marca                   375429 non-null  object
 23  ano_fabricacao_veiculo  370896 non-null  object
```

```
 24  tipo_envolvido          394482 non-null  object
 25  estado_fisico           394482 non-null  object
 26  idade                   314893 non-null  object
 27  sexo                    394482 non-null  object
 28  ilesos                  394482 non-null  object
 29  feridos_leves           394482 non-null  object
 30  feridos_graves          394482 non-null  object
 31  mortos                  394482 non-null  object
 32  latitude                394482 non-null  object
 33  longitude               394482 non-null  object
 34  regional                394482 non-null  object
 35  delegacia               394482 non-null  object
 36  uop                     391833 non-null  object
dtypes: object(37)
memory usage: 111.4+ MB
```

[73]:
```
#Acid2020 - Eliminando as colunas que não serão usadas
acid2020a = acid2020.drop(columns=['id', 'pesid', 'km', 'causa_principal',␣
 ↪'ordem_tipo_acidente','fase_dia', 'sentido_via',␣
 ↪'condicao_metereologica','tipo_pista','tracado_via', 'uso_solo',␣
 ↪'id_veiculo', 'tipo_veiculo', 'marca', 'ano_fabricacao_veiculo',␣
 ↪'tipo_envolvido', 'estado_fisico', 'idade', 'sexo','regional', 'delegacia',␣
 ↪'uop',  'latitude','longitude', 'municipio'], axis = 1)
acid2020a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 394482 entries, 0 to 394481
Data columns (total 12 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   data_inversa            394482 non-null  object
 1   dia_semana              394482 non-null  object
 2   horario                 394482 non-null  object
 3   uf                      394482 non-null  object
 4   br                      393468 non-null  object
 5   causa_acidente          394482 non-null  object
 6   tipo_acidente           394481 non-null  object
 7   classificacao_acidente  394481 non-null  object
 8   ilesos                  394482 non-null  object
 9   feridos_leves           394482 non-null  object
 10  feridos_graves          394482 non-null  object
 11  mortos                  394482 non-null  object
dtypes: object(12)
memory usage: 36.1+ MB
```

[74]:
```
# Criando coluna de hora inteira
acid2020a['Hora'] = acid2020a['horario'].str.split(':').str[0]
```

```
#Deletando coluna antiga de horário
del acid2020a['horario']
acid2020a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 394482 entries, 0 to 394481
Data columns (total 12 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   data_inversa            394482 non-null  object
 1   dia_semana              394482 non-null  object
 2   uf                      394482 non-null  object
 3   br                      393468 non-null  object
 4   causa_acidente          394482 non-null  object
 5   tipo_acidente           394481 non-null  object
 6   classificacao_acidente  394481 non-null  object
 7   ilesos                  394482 non-null  object
 8   feridos_leves           394482 non-null  object
 9   feridos_graves          394482 non-null  object
 10  mortos                  394482 non-null  object
 11  Hora                    394482 non-null  object
dtypes: object(12)
memory usage: 36.1+ MB
```

[75]:
```
# Renomeando para o padrão usado
acid2020a.columns = ['Data', 'Dia_semana', 'UF',  'BR', 'Causa',␣
 ↪'Tipo_acidente','Classificacao', 'Ilesos', 'Feridos_leves',␣
 ↪'Feridos_graves', 'Obitos', 'Hora']
acid2020a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 394482 entries, 0 to 394481
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Data            394482 non-null  object
 1   Dia_semana      394482 non-null  object
 2   UF              394482 non-null  object
 3   BR              393468 non-null  object
 4   Causa           394482 non-null  object
 5   Tipo_acidente   394481 non-null  object
 6   Classificacao   394481 non-null  object
 7   Ilesos          394482 non-null  object
 8   Feridos_leves   394482 non-null  object
 9   Feridos_graves  394482 non-null  object
 10  Obitos          394482 non-null  object
 11  Hora            394482 non-null  object
dtypes: object(12)
```

```
memory usage: 36.1+ MB
```

[76]:
```
# Verificando se há linhas nulas
acid2020a.isnull().sum()
```

[76]:
```
Data                0
Dia_semana          0
UF                  0
BR               1014
Causa               0
Tipo_acidente       1
Classificacao       1
Ilesos              0
Feridos_leves       0
Feridos_graves      0
Obitos              0
Hora                0
dtype: int64
```

[77]:
```
#Deletaremos agora as linhas nulas
acid2020b = acid2020a.dropna(subset = ['BR'])
acid2020b = acid2020b.dropna(subset = ['Tipo_acidente'])
acid2020b.isnull().sum()
```

[77]:
```
Data             0
Dia_semana       0
UF               0
BR               0
Causa            0
Tipo_acidente    0
Classificacao    0
Ilesos           0
Feridos_leves    0
Feridos_graves   0
Obitos           0
Hora             0
dtype: int64
```

[78]:
```
#Formatando tipo das colunas
# Vamos manter data ainda como object pra fazer a identificação de MÊS
acid2020c = acid2020b.astype({"BR": int, "Hora": int, 'Ilesos': int,␣
 ↪'Feridos_leves' : int, 'Feridos_graves': int, 'Obitos' : int})
acid2020c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 393467 entries, 0 to 394481
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype
```

```
---  ------           --------------   -----
 0   Data             393467 non-null  object
 1   Dia_semana       393467 non-null  object
 2   UF               393467 non-null  object
 3   BR               393467 non-null  int32
 4   Causa            393467 non-null  object
 5   Tipo_acidente    393467 non-null  object
 6   Classificacao    393467 non-null  object
 7   Ilesos           393467 non-null  int32
 8   Feridos_leves    393467 non-null  int32
 9   Feridos_graves   393467 non-null  int32
 10  Obitos           393467 non-null  int32
 11  Hora             393467 non-null  int32
dtypes: int32(6), object(6)
memory usage: 30.0+ MB
```

15 Análise do estado com mais acidentes

16 Ranking e gráfico de acidentes por UF - 2017

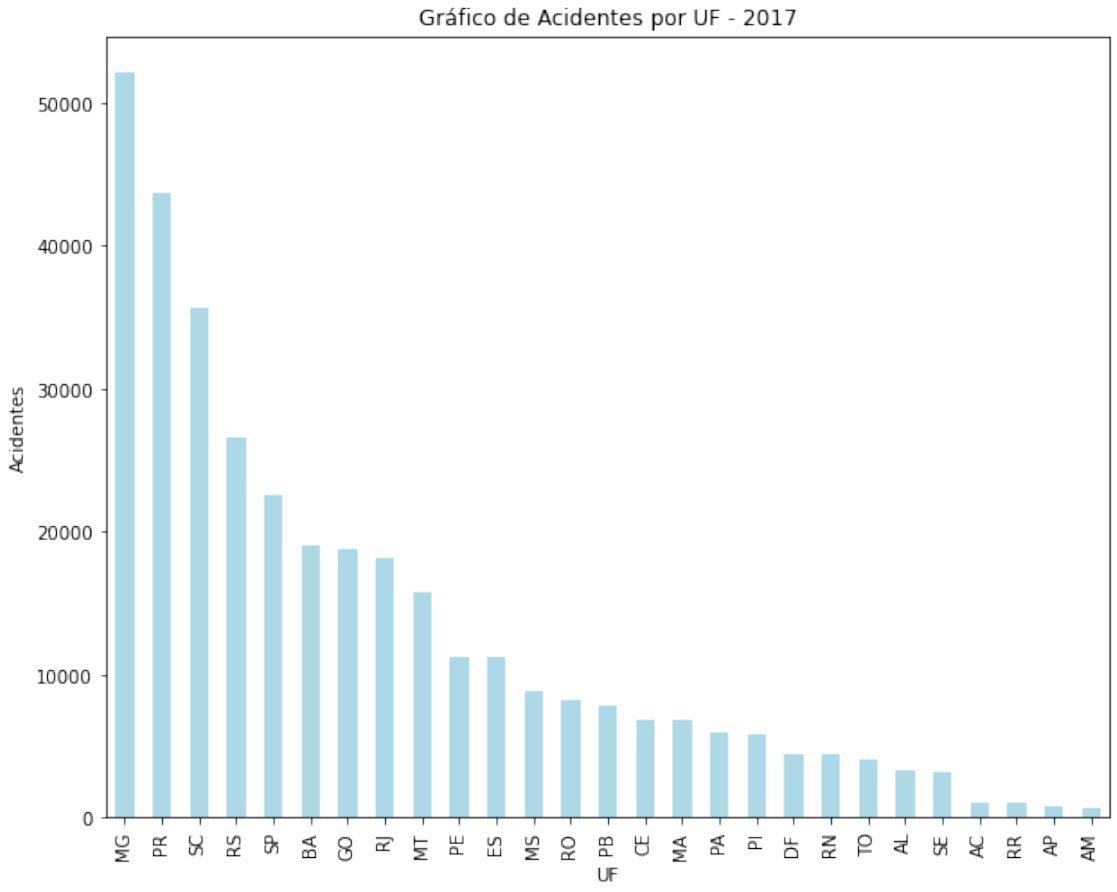
[79]:
```
# Agrupando acidentes por UF - 2017
acid2017top = acid2017c.groupby("UF").size().sort_values(ascending=False)
acid2017top = acid2017top.rename("Total de acidentes por ano")
acid2017top
```

[79]:
```
UF
MG    52091
PR    43634
SC    35665
RS    26650
SP    22513
BA    19093
GO    18851
RJ    18107
MT    15812
PE    11242
ES    11236
MS     8886
RO     8217
PB     7894
CE     6894
MA     6851
PA     5925
PI     5850
DF     4466
RN     4443
```

```
TO     4034
AL     3344
SE     3182
AC     1086
RR     1068
AP      787
AM      725
Name: Total de acidentes por ano, dtype: int64
```

[80]:
```
# Gráfico de Acidentes por UF - 2017
acid2017top.plot.bar(figsize=(10,8), color = 'lightblue', title = "Gráfico de␣
↪Acidentes por UF - 2017", ylabel = 'Acidentes')
```

[80]:
```
<AxesSubplot:title={'center':'Gráfico de Acidentes por UF - 2017'}, xlabel='UF',
ylabel='Acidentes'>
```



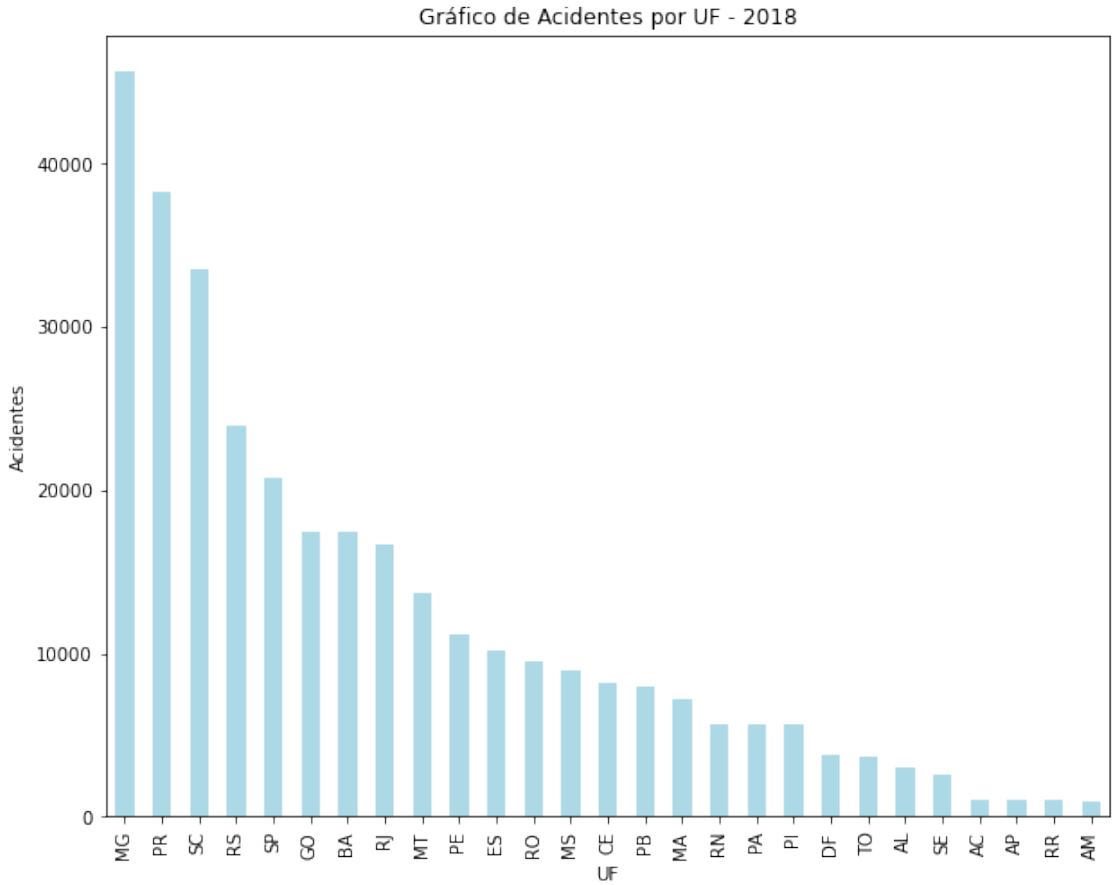
17 Ranking e gráfico de acidentes por UF - 2018

[81]:
```
# Agrupando acidentes por UF
acid2018top = acid2018c.groupby("UF").size().sort_values(ascending=False)
acid2018top = acid2018top.rename("Total de acidentes por ano")
acid2018top
```

[81]:
```
UF
MG    45568
PR    38170
SC    33458
RS    23949
SP    20695
GO    17433
BA    17413
RJ    16634
MT    13668
PE    11124
ES    10166
RO     9491
MS     8917
CE     8225
PB     7992
MA     7250
RN     5653
PA     5626
PI     5616
DF     3781
TO     3722
AL     3005
SE     2605
AC     1067
AP     1052
RR     1042
AM      880
Name: Total de acidentes por ano, dtype: int64
```

[82]:
```
# Gráfico de Acidentes por UF - 2018
acid2018top.plot.bar(figsize=(10,8), color = 'lightblue', title = "Gráfico de␣
↪Acidentes por UF - 2018", ylabel = 'Acidentes')
```

[82]:
```
<AxesSubplot:title={'center':'Gráfico de Acidentes por UF - 2018'}, xlabel='UF',
ylabel='Acidentes'>
```



18 Ranking e gráfico de acidentes por UF - 2019

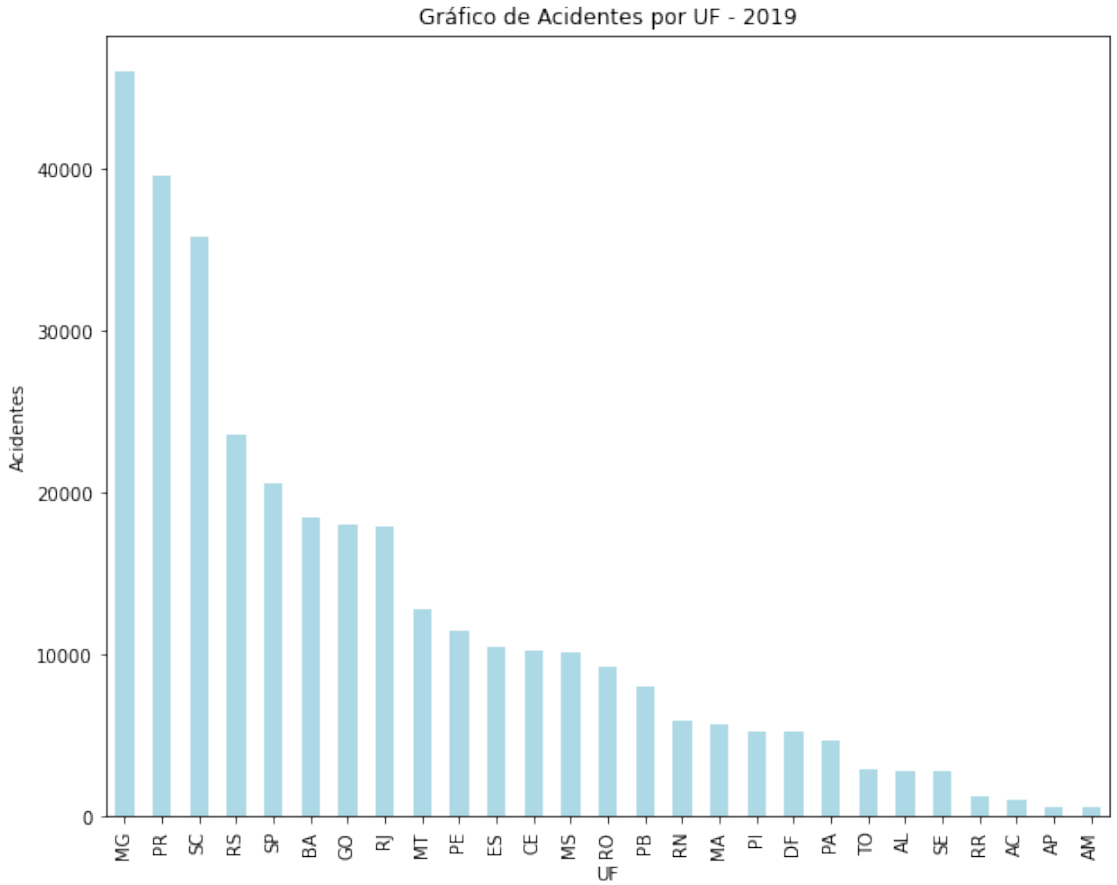
[83]:
```
# Agrupando acidentes por UF
acid2019top = acid2019c.groupby("UF").size().sort_values(ascending=False)
acid2019top = acid2019top.rename("Total de acidentes por ano")
acid2019top
```

[83]:
```
UF
MG    45947
PR    39487
SC    35812
RS    23581
SP    20567
BA    18466
GO    17979
RJ    17943
MT    12780
PE    11495
```

```
ES    10434
CE    10260
MS    10109
RO     9205
PB     8086
RN     5882
MA     5741
PI     5311
DF     5270
PA     4713
TO     2976
AL     2834
SE     2813
RR     1261
AC     1037
AP      654
AM      608
Name: Total de acidentes por ano, dtype: int64
```

[84]:
```
# Gráfico de Acidentes por UF - 2019
acid2019top.plot.bar(figsize=(10,8), color = 'lightblue', title = "Gráfico de␣
↪Acidentes por UF - 2019", ylabel = 'Acidentes')
```

[84]:
```
<AxesSubplot:title={'center':'Gráfico de Acidentes por UF - 2019'}, xlabel='UF',
ylabel='Acidentes'>
```



19 Ranking e gráfico de acidentes por UF - 2020

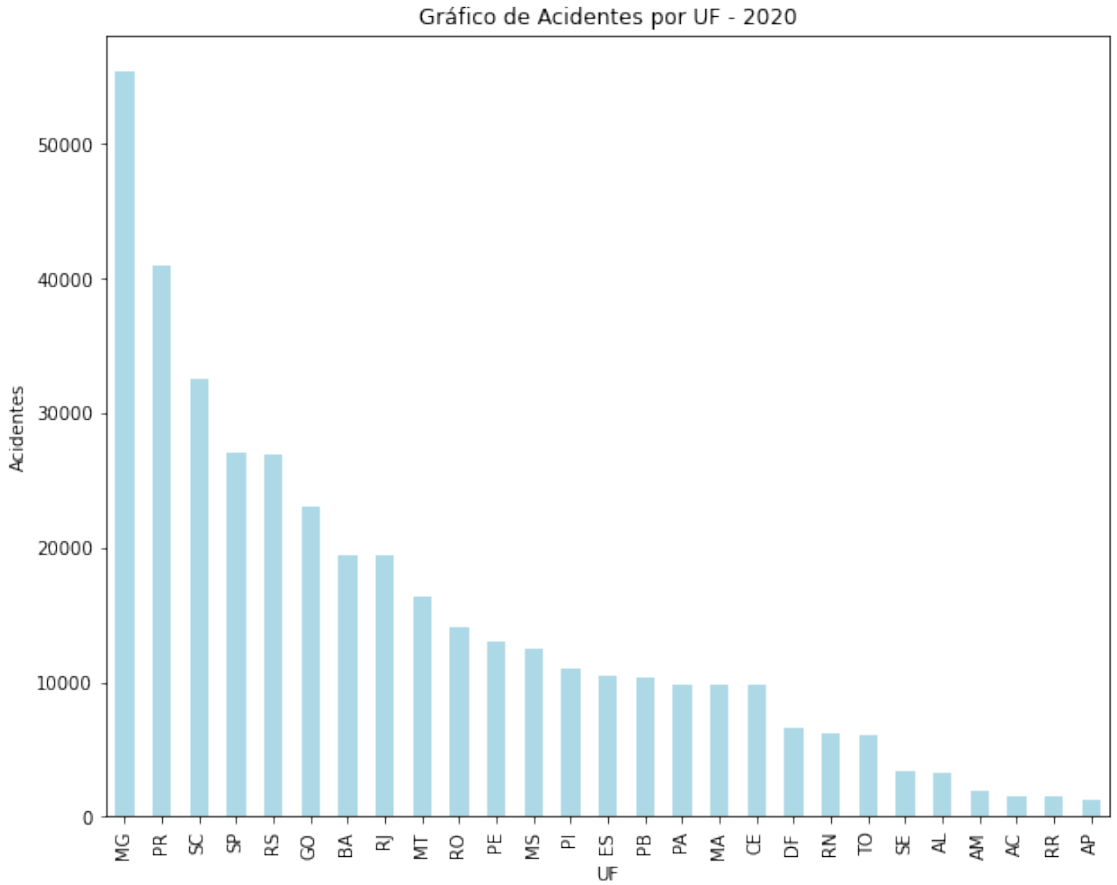
[85]:
```
# Agrupando acidentes por UF - 2020
acid2020top = acid2020c.groupby("UF").size().sort_values(ascending=False)
acid2020top = acid2020top.rename("Total de acidentes por ano")
acid2020top
```

[85]:
```
UF
MG    55309
PR    40925
SC    32492
SP    27048
RS    26843
GO    23021
BA    19476
RJ    19384
MT    16403
RO    14124
```

```
PE    13028
MS    12474
PI    10982
ES    10426
PB    10284
PA     9872
MA     9797
CE     9772
DF     6646
RN     6180
TO     6071
SE     3397
AL     3197
AM     1962
AC     1582
RR     1579
AP     1193
Name: Total de acidentes por ano, dtype: int64
```

[86]:
```
# Gráfico de Acidentes por UF - 2020
acid2020top.plot.bar(figsize=(10,8), color = 'lightblue', title = "Gráfico de␣
↪Acidentes por UF - 2020", ylabel = 'Acidentes')
```

[86]:
```
<AxesSubplot:title={'center':'Gráfico de Acidentes por UF - 2020'}, xlabel='UF',
ylabel='Acidentes'>
```



20

21 Mergir o top infrações com top acidentes UF

22

22.1 Ano 2017

[87]:
```
# Mergindo o ranking de infrações por UF com o ranking de acidentes por UF - 2017
print ("UFs com maior ocorrências de infrações e acidentes - 2017")
toptotal2017 = pd.merge(ano2017_ufs, acid2017top, how ='left',  on = ['UF'])
toptotal2017['Infrações + Acidentes'] = (toptotal2017['Total de Infrações por ano'] + toptotal2017['Total de acidentes por ano'])
toptotal2017.sort_values(ascending = False, by = 'Infrações + Acidentes')[:5]
```

```
UFs com maior ocorrências de infrações e acidentes - 2017
```

[87]:
```
    Total de Infrações por ano  Total de acidentes por ano  \
UF
SP                      930622                       22513
RJ                      832436                       18107
MG                      653881                       52091
BA                      667166                       19093
PR                      528561                       43634

    Infrações + Acidentes
UF
SP                  953135
RJ                  850543
MG                  705972
BA                  686259
PR                  572195
```

22.2 Ano 2018

[88]:
```
# Mergindo o ranking de infrações por UF com o ranking de acidentes por UF -␣
↪2018
print ("UFs com maior ocorrências de infrações e acidentes - 2018")
toptotal2018 = pd.merge(ano2018_ufs, acid2018top, how ='left',  on = ['UF'])
toptotal2018['Infrações + Acidentes'] = (toptotal2018['Total de Infrações por␣
↪ano'] + toptotal2018['Total de acidentes por ano'])
toptotal2018.sort_values(ascending = False, by = 'Infrações + Acidentes')[:5]
```

```
UFs com maior ocorrências de infrações e acidentes - 2018
```

[88]:
```
    Total de Infrações por ano  Total de acidentes por ano  \
UF
MG                     1206324                       45568
RJ                     1127572                       16634
SP                      799175                       20695
BA                      714759                       17413
PR                      525525                       38170

    Infrações + Acidentes
UF
MG                1251892
RJ                1144206
SP                 819870
BA                 732172
PR                 563695
```

22.3 Ano 2019

[89]:
```
# Mergindo o ranking de infrações por UF com o ranking de acidentes por UF -␣
↪2019
print ("UFs com maior ocorrências de infrações e acidentes - 2019")
toptotal2019 = pd.merge(ano2019_ufs, acid2019top, how ='left',  on = ['UF'])
toptotal2019['Infrações + Acidentes'] = (toptotal2019['Total de Infrações por␣
↪ano'] + toptotal2019['Total de acidentes por ano'])
toptotal2019.sort_values(ascending = False, by = 'Infrações + Acidentes')[:5]
```

```
UFs com maior ocorrências de infrações e acidentes - 2019
```

[89]:

UF	Total de Infrações por ano	Total de acidentes por ano	Infrações + Acidentes
RJ	765807	17943	783750
MG	611677	45947	657624
SP	631019	20567	651586
BA	594990	18466	613456
PR	364895	39487	404382

22.4 Ano 2020

[90]:
```
# Mergindo o ranking de infrações por UF com o ranking de acidentes por UF -␣
↪2020
print ("UFs com maior ocorrências de infrações e acidentes - 2020")
toptotal2020 = pd.merge(ano2020_ufs, acid2020top, how ='left',  on = ['UF'])
toptotal2020['Infrações + Acidentes'] = (toptotal2020['Total de Infrações por␣
↪ano'] + toptotal2020['Total de acidentes por ano'])
toptotal2020.sort_values(ascending = False, by = 'Infrações + Acidentes')[:5]
```

```
UFs com maior ocorrências de infrações e acidentes - 2020
```

[90]:

UF	Total de Infrações por ano	Total de acidentes por ano
RJ	919184	19384
MG	665769	55309
SP	475498	27048
BA	403442	19476
MS	399371	12474

```
    Infrações + Acidentes
UF
RJ                 938568
MG                 721078
SP                 502546
BA                 422918
MS                 411845
```

22.4.1 Foi decidido então que o estado foco do projeto seria o Rio de Janeiro

23 Filtrando: estado do Rio de Janeiro

23.1 Ano 2017

[91]:
```
# INFRAÇÕES RIO DE JANEIRO 2017
estado = ['RJ']
inf2017rj = ano2017c[ano2017c['UF'].isin(estado)]
inf2017rj.info(0)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 832436 entries, 2 to 580890
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Data       832436 non-null  object
 1   UF         832436 non-null  object
 2   BR         832436 non-null  int32
 3   Codigo     832436 non-null  int32
 4   Descricao  832436 non-null  object
 5   Hora       832436 non-null  int32
dtypes: int32(3), object(3)
memory usage: 34.9+ MB
```

[92]:
```
# ACIDENTES RIO DE JANEIRO 2017
estado = ['RJ']
acid17rj = acid2017c[acid2017c['UF'].isin(estado)]
acid17rj.head(2)
acid17rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18107 entries, 145 to 349055
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Data            18107 non-null  object
 1   Dia_semana      18107 non-null  object
 2   UF              18107 non-null  object
 3   BR              18107 non-null  int32
 4   Causa           18107 non-null  object
```

```
 5   Tipo_acidente   18107 non-null  object
 6   Classificacao   18107 non-null  object
 7   Ilesos          18107 non-null  int32
 8   Feridos_leves   18107 non-null  int32
 9   Feridos_graves  18107 non-null  int32
 10  Obitos          18107 non-null  int32
 11  Hora            18107 non-null  int32
dtypes: int32(6), object(6)
memory usage: 1.4+ MB
```

23.2 Ano 2018

[93]:
```
# INFRAÇÕES RIO DE JANEIRO 2018
estado = ['RJ']
inf2018rj = ano2018c[ano2018c['UF'].isin(estado)]
inf2018rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1127572 entries, 18 to 727724
Data columns (total 6 columns):
 #   Column     Non-Null Count    Dtype
---  ------     --------------    -----
 0   Data       1127572 non-null  object
 1   UF         1127572 non-null  object
 2   BR         1127572 non-null  int32
 3   Codigo     1127572 non-null  int32
 4   Descricao  1127572 non-null  object
 5   Hora       1127572 non-null  int32
dtypes: int32(3), object(3)
memory usage: 47.3+ MB
```

[94]:
```
# ACIDENTES RIO DE JANEIRO 2018
estado = ['RJ']
acid18rj = acid2018c[acid2018c['UF'].isin(estado)]
acid18rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16634 entries, 0 to 324785
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Data            16634 non-null  object
 1   Dia_semana      16634 non-null  object
 2   UF              16634 non-null  object
 3   BR              16634 non-null  int32
 4   Causa           16634 non-null  object
 5   Tipo_acidente   16634 non-null  object
 6   Classificacao   16634 non-null  object
 7   Ilesos          16634 non-null  int32
```

```
 8   Feridos_leves   16634 non-null  int32
 9   Feridos_graves  16634 non-null  int32
 10  Obitos          16634 non-null  int32
 11  Hora            16634 non-null  int32
dtypes: int32(6), object(6)
memory usage: 1.3+ MB
```

23.3 Ano 2019

[95]:
```
# INFRAÇÕES RIO DE JANEIRO 2019
estado = ['RJ']
inf2019rj = ano2019c[ano2019c['UF'].isin(estado)]
inf2019rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 765807 entries, 1 to 492719
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Data       765807 non-null  object
 1   UF         765807 non-null  object
 2   BR         765807 non-null  int32
 3   Codigo     765807 non-null  int32
 4   Descricao  765807 non-null  object
 5   Hora       765807 non-null  int32
dtypes: int32(3), object(3)
memory usage: 32.1+ MB
```

[96]:
```
# ACIDENTES RIO DE JANEIRO 2019
estado = ['RJ']
acid19rj = acid2019c[acid2019c['UF'].isin(estado)]
acid19rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17943 entries, 45 to 331584
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Data            17943 non-null  object
 1   Dia_semana      17943 non-null  object
 2   UF              17943 non-null  object
 3   BR              17943 non-null  int32
 4   Causa           17943 non-null  object
 5   Tipo_acidente   17943 non-null  object
 6   Classificacao   17943 non-null  object
 7   Ilesos          17943 non-null  int32
 8   Feridos_leves   17943 non-null  int32
 9   Feridos_graves  17943 non-null  int32
 10  Obitos          17943 non-null  int32
```

```
 11  Hora            17943 non-null  int32
dtypes: int32(6), object(6)
memory usage: 1.4+ MB
```

23.4 Ano 2020

[97]:
```
# INFRAÇÕES RIO DE JANEIRO 2020
estado = ['RJ']
inf2020rj = ano2020c[ano2020c['UF'].isin(estado)]
inf2020rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 919184 entries, 4 to 499945
Data columns (total 6 columns):
 #   Column     Non-Null Count   Dtype 
---  ------     --------------   ----- 
 0   Data       919184 non-null  object
 1   UF         919184 non-null  object
 2   BR         919184 non-null  int32 
 3   Codigo     919184 non-null  int32 
 4   Descricao  919184 non-null  object
 5   Hora       919184 non-null  int32 
dtypes: int32(3), object(3)
memory usage: 38.6+ MB
```

[98]:
```
# ACIDENTES RIO DE JANEIRO 2020
estado = ['RJ']
acid20rj = acid2020c[acid2020c['UF'].isin(estado)]
acid20rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19384 entries, 10 to 394054
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Data            19384 non-null  object
 1   Dia_semana      19384 non-null  object
 2   UF              19384 non-null  object
 3   BR              19384 non-null  int32 
 4   Causa           19384 non-null  object
 5   Tipo_acidente   19384 non-null  object
 6   Classificacao   19384 non-null  object
 7   Ilesos          19384 non-null  int32 
 8   Feridos_leves   19384 non-null  int32 
 9   Feridos_graves  19384 non-null  int32 
 10  Obitos          19384 non-null  int32 
 11  Hora            19384 non-null  int32 
dtypes: int32(6), object(6)
memory usage: 1.5+ MB
```

24

25 DATASET CLASSIFICAÇÃO DAS INFRAÇÕES

26

[99]:
```
#Lendo o arquivo de tipos de infrações
tiposdeinfracoes = pd.read_excel(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.
↪TCC\Tipos\tabela_infracoes.xlsx", squeeze=True, dtype = 'object')
```

[100]:
```
# Verificando a base
tiposdeinfracoes.columns.tolist()
```

[100]:
```
['CÓDIGO DA INFRAÇÃO',
 'INFRAÇÃO',
 'RESPONSÁVEL',
 'VALOR DA MULTA',
 'ARTIGOS DO CTB',
 'Tipo']
```

[101]:
```
# Tiposdeinfracao - Eliminando colunas que não serão usadas
tipos_infra = tiposdeinfracoes.drop(columns = ['INFRAÇÃO', 'ARTIGOS DO CTB',␣
↪'VALOR DA MULTA'], axis = 1)

# Renomeando para o padrão
tipos_infra.rename(columns = {'CÓDIGO DA INFRAÇÃO':'Codigo','RESPONSÁVEL' :␣
↪'Responsavel'}, inplace = True)

# Alterando o tipo de dados da coluna
tipos_infra = tipos_infra.astype({"Codigo": int})
tipos_infra.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 412 entries, 0 to 411
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Codigo       412 non-null    int32
 1   Responsavel  412 non-null    object
 2   Tipo         412 non-null    object
dtypes: int32(1), object(2)
memory usage: 8.2+ KB
```

26.0.1 Mergindo com inf2017rj

[102]:
```
# Mergindo a base tiposdeinfracao com base de infrações RJ - 2017
infra17rj = pd.merge(inf2017rj, tipos_infra, on ='Codigo', how ='left')

# Verificando se há linhas nulas
infra17rj.isnull().sum()
```

[102]:
```
Data              0
UF                0
BR                0
Codigo            0
Descricao         0
Hora              0
Responsavel    12385
Tipo           12385
dtype: int64
```

[103]:
```
# Deletaremos as linhas nulas
infra17rj = infra17rj.dropna(subset = ['Tipo'])
infra17rj.isnull().sum()
```

[103]:
```
Data           0
UF             0
BR             0
Codigo         0
Descricao      0
Hora           0
Responsavel    0
Tipo           0
dtype: int64
```

[104]:
```
# Verificando a nova base
infra17rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 820363 entries, 0 to 832747
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Data         820363 non-null  object
 1   UF           820363 non-null  object
 2   BR           820363 non-null  int32
 3   Codigo       820363 non-null  int32
 4   Descricao    820363 non-null  object
 5   Hora         820363 non-null  int32
 6   Responsavel  820363 non-null  object
 7   Tipo         820363 non-null  object
```

```
dtypes: int32(3), object(5)
memory usage: 46.9+ MB
```

26.0.2 Mergindo com inf2018rj

[105]:
```
# Mergindo a base tiposdeinfracao com base de infrações RJ - 2018
infra18rj = pd.merge(inf2018rj, tipos_infra, on ='Codigo', how ='left')

# Verificando se há linhas nulas
infra18rj.isnull().sum()
```

[105]:
```
Data              0
UF                0
BR                0
Codigo            0
Descricao         0
Hora              0
Responsavel    9693
Tipo           9693
dtype: int64
```

[106]:
```
# Deletaremos as linhas nulas
infra18rj = infra18rj.dropna(subset = ['Tipo'])
infra18rj.isnull().sum()
```

[106]:
```
Data           0
UF             0
BR             0
Codigo         0
Descricao      0
Hora           0
Responsavel    0
Tipo           0
dtype: int64
```

[107]:
```
# Verificando a nova base
infra18rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1118090 entries, 0 to 1127782
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype
---  ------       --------------    -----
 0   Data         1118090 non-null  object
 1   UF           1118090 non-null  object
 2   BR           1118090 non-null  int32
 3   Codigo       1118090 non-null  int32
 4   Descricao    1118090 non-null  object
```

```
 5   Hora          1118090 non-null  int32
 6   Responsavel   1118090 non-null  object
 7   Tipo          1118090 non-null  object
dtypes: int32(3), object(5)
memory usage: 64.0+ MB
```

26.0.3 Mergindo com inf2019rj

[108]:
```
# Mergindo a base tiposdeinfracao com base de infrações RJ - 2019
infra19rj = pd.merge(inf2019rj, tipos_infra, on ='Codigo', how ='left')

#Verificando se há linhas nulas
infra19rj.isnull().sum()
```

[108]:
```
Data              0
UF                0
BR                0
Codigo            0
Descricao         0
Hora              0
Responsavel    15730
Tipo           15730
dtype: int64
```

[109]:
```
# Deletaremos as linhas nulas
infra19rj = infra19rj.dropna(subset = ['Tipo'])
infra19rj.isnull().sum()
```

[109]:
```
Data           0
UF             0
BR             0
Codigo         0
Descricao      0
Hora           0
Responsavel    0
Tipo           0
dtype: int64
```

[110]:
```
# Verificando a nova base
infra19rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 750338 entries, 0 to 766067
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Data         750338 non-null  object
 1   UF           750338 non-null  object
```

```
 2   BR           750338 non-null  int32
 3   Codigo       750338 non-null  int32
 4   Descricao    750338 non-null  object
 5   Hora         750338 non-null  int32
 6   Responsavel  750338 non-null  object
 7   Tipo         750338 non-null  object
dtypes: int32(3), object(5)
memory usage: 42.9+ MB
```

26.0.4 Mergindo com inf2020rj

[111]:
```
# Mergindo a base tiposdeinfracao com base de infrações RJ - 2020
infra20rj = pd.merge(inf2020rj, tipos_infra, on ='Codigo', how ='left')

# Verificando se há linhas nulas
infra20rj.isnull().sum()
```

[111]:
```
Data              0
UF                0
BR                0
Codigo            0
Descricao         0
Hora              0
Responsavel    17773
Tipo           17773
dtype: int64
```

[112]:
```
# Deletaremos as linhas nulas
infra20rj = infra20rj.dropna(subset = ['Tipo'])
infra20rj.isnull().sum()
```

[112]:
```
Data           0
UF             0
BR             0
Codigo         0
Descricao      0
Hora           0
Responsavel    0
Tipo           0
dtype: int64
```

[113]:
```
# Verificando a nova base
infra20rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 901659 entries, 0 to 919431
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
```

```
---  ------       --------------   -----
 0   Data         901659 non-null  object
 1   UF           901659 non-null  object
 2   BR           901659 non-null  int32
 3   Codigo       901659 non-null  int32
 4   Descricao    901659 non-null  object
 5   Hora         901659 non-null  int32
 6   Responsavel  901659 non-null  object
 7   Tipo         901659 non-null  object
dtypes: int32(3), object(5)
memory usage: 51.6+ MB
```

26.1 Adicionaremos coluna de PERÍODO DO DIA

[114]:
```
# Primeiramente criaremos cópias dos dataframes de infrações usados

import copy
infra17rj_copy = pd.DataFrame(columns = infra17rj.columns, data = copy.
 ↪deepcopy(infra17rj.values))
print ("Infrações RJ 2017 - completa")
infra18rj_copy = pd.DataFrame(columns = infra18rj.columns, data = copy.
 ↪deepcopy(infra18rj.values))
print ("Infrações RJ 2018 - completa")
infra19rj_copy = pd.DataFrame(columns = infra19rj.columns, data = copy.
 ↪deepcopy(infra19rj.values))
print ("Infrações RJ 2019 - completa")
infra20rj_copy = pd.DataFrame(columns = infra20rj.columns, data = copy.
 ↪deepcopy(infra20rj.values))
print ("Infrações RJ 2020 - completa")
```

```
Infrações RJ 2017 - completa
Infrações RJ 2018 - completa
Infrações RJ 2019 - completa
Infrações RJ 2020 - completa
```

[115]:
```
# Agora criaremos cópias dos dataframes de acidentes

acid17rj_copy = pd.DataFrame(columns = acid17rj.columns, data = copy.
 ↪deepcopy(acid17rj.values))
print ("Acidentes RJ 2017 - completa")
acid18rj_copy = pd.DataFrame(columns = acid18rj.columns, data = copy.
 ↪deepcopy(acid18rj.values))
print ("Acidentes RJ 2018 - completa")
acid19rj_copy = pd.DataFrame(columns = acid19rj.columns, data = copy.
 ↪deepcopy(acid19rj.values))
print ("Acidentes RJ 2019 - completa")
```

```
acid20rj_copy = pd.DataFrame(columns = acid20rj.columns, data = copy.
 ↪deepcopy(acid20rj.values))
print ("Acidentes RJ 2020 - completa")
```

```
Acidentes RJ 2017 - completa
Acidentes RJ 2018 - completa
Acidentes RJ 2019 - completa
Acidentes RJ 2020 - completa
```

[116]:
```
# Verificaremos as cópias de infrações
print ("Infrações RJ 2017")
infra17rj_copy.info()
print ("Infrações RJ 2018")
infra18rj_copy.info()
print ("Infrações RJ 2019")
infra19rj_copy.info()
print ("Infrações RJ 2020")
infra20rj_copy.info()
print ("Cópias realizadas")
```

```
Infrações RJ 2017
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 820363 entries, 0 to 820362
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype 
---  ------       --------------   ----- 
 0   Data         820363 non-null  object
 1   UF           820363 non-null  object
 2   BR           820363 non-null  object
 3   Codigo       820363 non-null  object
 4   Descricao    820363 non-null  object
 5   Hora         820363 non-null  object
 6   Responsavel  820363 non-null  object
 7   Tipo         820363 non-null  object
dtypes: object(8)
memory usage: 50.1+ MB
Infrações RJ 2018
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1118090 entries, 0 to 1118089
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype 
---  ------       --------------    ----- 
 0   Data         1118090 non-null  object
 1   UF           1118090 non-null  object
 2   BR           1118090 non-null  object
 3   Codigo       1118090 non-null  object
 4   Descricao    1118090 non-null  object
 5   Hora         1118090 non-null  object
```

```
 6   Responsavel  1118090 non-null  object
 7   Tipo         1118090 non-null  object
dtypes: object(8)
memory usage: 68.2+ MB
Infrações RJ 2019
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 750338 entries, 0 to 750337
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Data         750338 non-null  object
 1   UF           750338 non-null  object
 2   BR           750338 non-null  object
 3   Codigo       750338 non-null  object
 4   Descricao    750338 non-null  object
 5   Hora         750338 non-null  object
 6   Responsavel  750338 non-null  object
 7   Tipo         750338 non-null  object
dtypes: object(8)
memory usage: 45.8+ MB
Infrações RJ 2020
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 901659 entries, 0 to 901658
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Data         901659 non-null  object
 1   UF           901659 non-null  object
 2   BR           901659 non-null  object
 3   Codigo       901659 non-null  object
 4   Descricao    901659 non-null  object
 5   Hora         901659 non-null  object
 6   Responsavel  901659 non-null  object
 7   Tipo         901659 non-null  object
dtypes: object(8)
memory usage: 55.0+ MB
Cópias realizadas
```

[117]:
```
# E agora verificaremos as cópias de acidentes
print ("Acidentes RJ 2017")
acid17rj_copy.info()
print ("Acidentes RJ 2018")
acid18rj_copy.info()
print ("Acidentes RJ 2019")
acid19rj_copy.info()
print ("Acidentes RJ 2020")
acid20rj_copy.info()
```

```
Acidentes RJ 2017
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18107 entries, 0 to 18106
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Data            18107 non-null  object
 1   Dia_semana      18107 non-null  object
 2   UF              18107 non-null  object
 3   BR              18107 non-null  object
 4   Causa           18107 non-null  object
 5   Tipo_acidente   18107 non-null  object
 6   Classificacao   18107 non-null  object
 7   Ilesos          18107 non-null  object
 8   Feridos_leves   18107 non-null  object
 9   Feridos_graves  18107 non-null  object
 10  Obitos          18107 non-null  object
 11  Hora            18107 non-null  object
dtypes: object(12)
memory usage: 1.7+ MB
Acidentes RJ 2018
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16634 entries, 0 to 16633
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Data            16634 non-null  object
 1   Dia_semana      16634 non-null  object
 2   UF              16634 non-null  object
 3   BR              16634 non-null  object
 4   Causa           16634 non-null  object
 5   Tipo_acidente   16634 non-null  object
 6   Classificacao   16634 non-null  object
 7   Ilesos          16634 non-null  object
 8   Feridos_leves   16634 non-null  object
 9   Feridos_graves  16634 non-null  object
 10  Obitos          16634 non-null  object
 11  Hora            16634 non-null  object
dtypes: object(12)
memory usage: 1.5+ MB
Acidentes RJ 2019
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17943 entries, 0 to 17942
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Data            17943 non-null  object
 1   Dia_semana      17943 non-null  object
```

```
 2   UF              17943 non-null  object
 3   BR              17943 non-null  object
 4   Causa           17943 non-null  object
 5   Tipo_acidente   17943 non-null  object
 6   Classificacao   17943 non-null  object
 7   Ilesos          17943 non-null  object
 8   Feridos_leves   17943 non-null  object
 9   Feridos_graves  17943 non-null  object
 10  Obitos          17943 non-null  object
 11  Hora            17943 non-null  object
dtypes: object(12)
memory usage: 1.6+ MB
Acidentes RJ 2020
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19384 entries, 0 to 19383
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Data            19384 non-null  object
 1   Dia_semana      19384 non-null  object
 2   UF              19384 non-null  object
 3   BR              19384 non-null  object
 4   Causa           19384 non-null  object
 5   Tipo_acidente   19384 non-null  object
 6   Classificacao   19384 non-null  object
 7   Ilesos          19384 non-null  object
 8   Feridos_leves   19384 non-null  object
 9   Feridos_graves  19384 non-null  object
 10  Obitos          19384 non-null  object
 11  Hora            19384 non-null  object
dtypes: object(12)
memory usage: 1.8+ MB
```

[118]:
```python
# Precisamos definir a fórmula que irá dividir o dia em períodos

def f(Hora):
    if Hora>=0 and Hora<=5 :
        return "Madrugada"
    elif Hora>= 6 and Hora<= 11:
        return "Manhã"
    elif Hora>= 12 and Hora<= 17:
        return "Tarde"
    else:
        return "Noite"
```

[119]:
```python
# Aplicando a fórmula - Inserindo coluna Período para bases de Infrações RJ
infra17rj_copy["Periodo"] = infra17rj_copy["Hora"].apply(lambda Hora: f(Hora))
```

```
print ("2017 ok")
infra18rj_copy["Periodo"] = infra18rj_copy["Hora"].apply(lambda Hora: f(Hora))
print ("2018 ok")
infra19rj_copy["Periodo"] = infra19rj_copy["Hora"].apply(lambda Hora: f(Hora))
print ("2019 ok")
infra20rj_copy["Periodo"] = infra20rj_copy["Hora"].apply(lambda Hora: f(Hora))
print ("2020 ok")
```

```
2017 ok
2018 ok
2019 ok
2020 ok
```

[120]:
```
## Aplicando a fórmula - Inserindo coluna Período para bases de Acidentes RJ
acid17rj_copy["Periodo"] = acid17rj_copy["Hora"].apply(lambda Hora: f(Hora))
print ("2017 ok")
acid18rj_copy["Periodo"] = acid18rj_copy["Hora"].apply(lambda Hora: f(Hora))
print ("2018 ok")
acid19rj_copy["Periodo"] = acid19rj_copy["Hora"].apply(lambda Hora: f(Hora))
print ("2019 ok")
acid20rj_copy["Periodo"] = acid20rj_copy["Hora"].apply(lambda Hora: f(Hora))
print ("2020 ok")
```

```
2017 ok
2018 ok
2019 ok
2020 ok
```

27 O que temos agora são as seguintes bases:

27.0.1 Infrações RJ por ano

infra17rj_copy

infra18rj_copy

infra19rj_copy

infra20rj_copy

27.0.2 Acidentes RJ por ano

acid17rj_copy

acid18rj_copy

acid19rj_copy

acid20rj_copy

[121]:
```
#Aqui se verificou as linhas nulas e não havia nenhuma em nenhum DF
# ex: acid20rj_copy.isnull().sum()
```

28 Adicionando coluna Index Mês

[123]:
```
# Inserindo a coluna Index_mes em todas as bases

infra17rj_copy['Index_mes'] = infra17rj_copy['Data'].str.split('-').str[1]
print ("infra 2017 ok")
infra18rj_copy['Index_mes'] = infra18rj_copy['Data'].str.split('-').str[1]
print ("infra 2018 ok")
infra19rj_copy['Index_mes'] = infra19rj_copy['Data'].str.split('-').str[1]
print ("infra 2019 ok")
infra20rj_copy['Index_mes'] = infra20rj_copy['Data'].str.split('-').str[1]
print ("infra 2020 ok")

acid17rj_copy['Index_mes'] = acid17rj_copy['Data'].str.split('-').str[1]
print ("acid 2017 ok")
acid18rj_copy['Index_mes'] = acid18rj_copy['Data'].str.split('-').str[1]
print ("acid 2018 ok")
acid19rj_copy['Index_mes'] = acid19rj_copy['Data'].str.split('-').str[1]
print ("acid 2019 ok")
acid20rj_copy['Index_mes'] = acid20rj_copy['Data'].str.split('-').str[1]
print ("acid 2020 ok")
```

```
infra 2017 ok
infra 2018 ok
infra 2019 ok
infra 2020 ok
acid 2017 ok
acid 2018 ok
acid 2019 ok
acid 2020 ok
```

29 Criando a coluna ANO e ANO-MÊS

[126]:
```
#Criando coluna ano
infra17rj_copy['ano'] = infra17rj_copy['Data'].str.split('-').str[0]
infra18rj_copy['ano'] = infra18rj_copy['Data'].str.split('-').str[0]
infra19rj_copy['ano'] = infra19rj_copy['Data'].str.split('-').str[0]
infra20rj_copy['ano'] = infra20rj_copy['Data'].str.split('-').str[0]

acid17rj_copy['ano'] = acid17rj_copy['Data'].str.split('-').str[0]
acid18rj_copy['ano'] = acid18rj_copy['Data'].str.split('-').str[0]
acid19rj_copy['ano'] = acid19rj_copy['Data'].str.split('-').str[0]
acid20rj_copy['ano'] = acid20rj_copy['Data'].str.split('-').str[0]
```

[128]:
```
#Juntando ano com mês
infra17rj_copy['mes_ano'] = infra17rj_copy[['ano', 'Index_mes']].agg('-'.join,␣
 ↪axis=1)
```

```
infra18rj_copy['mes_ano'] = infra18rj_copy[['ano', 'Index_mes']].agg('-'.join,␣
 ↪axis=1)
infra19rj_copy['mes_ano'] = infra19rj_copy[['ano', 'Index_mes']].agg('-'.join,␣
 ↪axis=1)
infra20rj_copy['mes_ano'] = infra20rj_copy[['ano', 'Index_mes']].agg('-'.join,␣
 ↪axis=1)

acid17rj_copy['mes_ano'] = acid17rj_copy[['ano', 'Index_mes']].agg('-'.join,␣
 ↪axis=1)
acid18rj_copy['mes_ano'] = acid18rj_copy[['ano', 'Index_mes']].agg('-'.join,␣
 ↪axis=1)
acid19rj_copy['mes_ano'] = acid19rj_copy[['ano', 'Index_mes']].agg('-'.join,␣
 ↪axis=1)
acid20rj_copy['mes_ano'] = acid20rj_copy[['ano', 'Index_mes']].agg('-'.join,␣
 ↪axis=1)
```

30 Criando a coluna de Mês por nome de mês

[130]:
```
# Converter tipo da coluna Index_mes para inteiro

infra17rj_copy['Index_mes'] = pd.to_numeric(infra17rj_copy['Index_mes'],␣
 ↪errors="coerce").astype('int64')
print ("infra 2017 ok")
infra18rj_copy['Index_mes'] = pd.to_numeric(infra18rj_copy['Index_mes'],␣
 ↪errors="coerce").astype('int64')
print ("infra 2018 ok")
infra19rj_copy['Index_mes'] = pd.to_numeric(infra19rj_copy['Index_mes'],␣
 ↪errors="coerce").astype('int64')
print ("infra 2019 ok")
infra20rj_copy['Index_mes'] = pd.to_numeric(infra20rj_copy['Index_mes'],␣
 ↪errors="coerce").astype('int64')
print ("infra 2020 ok")

acid17rj_copy['Index_mes'] =  pd.to_numeric(acid17rj_copy['Index_mes'],␣
 ↪errors="coerce").astype('int64')
print("acid 2017 ok")
acid18rj_copy['Index_mes'] =  pd.to_numeric(acid18rj_copy['Index_mes'],␣
 ↪errors="coerce").astype('int64')
print ("acid 2018 ok")
acid19rj_copy['Index_mes'] =  pd.to_numeric(acid19rj_copy['Index_mes'],␣
 ↪errors="coerce").astype('int64')
print ("acid 2019 ok")
acid20rj_copy['Index_mes'] =  pd.to_numeric(acid20rj_copy['Index_mes'],␣
 ↪errors="coerce").astype('int64')
print ("acid 2020 ok")
```

```
infra 2017 ok
infra 2018 ok
infra 2019 ok
infra 2020 ok
acid 2017 ok
acid 2018 ok
acid 2019 ok
acid 2020 ok
```

[132]:
```
# Precisamos definir agora a fórmula que nos dará o nome do mês baseado no seu␣
↪índice, Index_mes, criado anteriormente

def g(Index_mes):
    if Index_mes == 1 :
        return "Janeiro"
    elif Index_mes == 2:
        return "Fevereiro"
    elif Index_mes == 3:
        return "Março"
    elif Index_mes == 4:
        return "Abril"
    elif Index_mes == 5:
        return "Maio"
    elif Index_mes == 6:
        return "Junho"
    elif Index_mes == 7:
        return "Julho"
    elif Index_mes == 8:
        return "Agosto"
    elif Index_mes == 9:
        return "Setembro"
    elif Index_mes == 10:
        return "Outubro"
    elif Index_mes == 11:
        return "Novembro"
    else:
        return "Dezembro"
```

[133]:
```
# Inserindo coluna Mes para Infrações RJ

infra17rj_copy["Mes"] = infra17rj_copy["Index_mes"].apply(lambda Index_mes:␣
↪g(Index_mes))
print ("2017 ok")
infra18rj_copy["Mes"] = infra18rj_copy["Index_mes"].apply(lambda Index_mes:␣
↪g(Index_mes))
print ("2018 ok")
```

```
infra19rj_copy["Mes"] = infra19rj_copy["Index_mes"].apply(lambda Index_mes:␣
 ↪g(Index_mes))
print ("2019 ok")
infra20rj_copy["Mes"] = infra20rj_copy["Index_mes"].apply(lambda Index_mes:␣
 ↪g(Index_mes))
print ("2020 ok")
```

```
2017 ok
2018 ok
2019 ok
2020 ok
```

[134]:
```
# Inserindo coluna  Mes para Acidentes RJ

acid17rj_copy["Mes"] = acid17rj_copy["Index_mes"].apply(lambda Index_mes:␣
 ↪g(Index_mes))
print ("2017 ok")
acid18rj_copy["Mes"] = acid18rj_copy["Index_mes"].apply(lambda Index_mes:␣
 ↪g(Index_mes))
print ("2018 ok")
acid19rj_copy["Mes"] = acid19rj_copy["Index_mes"].apply(lambda Index_mes:␣
 ↪g(Index_mes))
print ("2019 ok")
acid20rj_copy["Mes"] = acid20rj_copy["Index_mes"].apply(lambda Index_mes:␣
 ↪g(Index_mes))
print ("2020 ok")
```

```
2017 ok
2018 ok
2019 ok
2020 ok
```

[136]:
```
# Deletando colunas Index Mês para todas as bases

del infra17rj_copy['Index_mes']
del infra18rj_copy['Index_mes']
del infra19rj_copy['Index_mes']
del infra20rj_copy['Index_mes']

del acid17rj_copy['Index_mes']
del acid18rj_copy['Index_mes']
del acid19rj_copy['Index_mes']
del acid20rj_copy['Index_mes']
```

30.1 Convertando Data para datetime

[138]:
```
#Convertendo coluna 'Data' de Infrações pra tipo datetime

infra17rj_copy['Data'] = infra17rj_copy.Data.astype('datetime64')
print ("Infra 2017 ok")
infra18rj_copy['Data'] = infra18rj_copy.Data.astype('datetime64')
print ("Infra 2018 ok")
infra19rj_copy['Data'] = infra19rj_copy.Data.astype('datetime64')
print ("Infra 2019 ok")
infra20rj_copy['Data'] = infra20rj_copy.Data.astype('datetime64')
print ("Infra 2020 ok")
```

```
Infra 2017 ok
Infra 2018 ok
Infra 2019 ok
Infra 2020 ok
```

[139]:
```
#Convertendo Data de Acidentes pra datetime

acid17rj_copy['Data'] = acid17rj_copy.Data.astype('datetime64')
print ("Acid 2017 ok")
acid18rj_copy['Data'] = acid18rj_copy.Data.astype('datetime64')
print ("Acid 2018 ok")
acid19rj_copy['Data'] = acid19rj_copy.Data.astype('datetime64')
print ("Acid 2019 ok")
acid20rj_copy['Data'] = acid20rj_copy.Data.astype('datetime64')
print ("Acid 2020 ok")
```

```
Acid 2017 ok
Acid 2018 ok
Acid 2019 ok
Acid 2020 ok
```

30.2 Convertendo Ilesos, Feridos e Óbitos para inteiro

[141]:
```
# Converter Ilesos para inteiro

acid17rj_copy['Ilesos'] =  pd.to_numeric(acid17rj_copy['Ilesos'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2017 ok")
acid18rj_copy['Ilesos'] =  pd.to_numeric(acid18rj_copy['Ilesos'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2018 ok")
acid19rj_copy['Ilesos'] =  pd.to_numeric(acid19rj_copy['Ilesos'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2019 ok")
```

```
acid20rj_copy['Ilesos'] =  pd.to_numeric(acid20rj_copy['Ilesos'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2020 ok")
```

```
Conversão Acid 2017 ok
Conversão Acid 2018 ok
Conversão Acid 2019 ok
Conversão Acid 2020 ok
```

[142]:
```
# Converter Feridos_leves para inteiro

acid17rj_copy['Feridos_leves'] =  pd.to_numeric(acid17rj_copy['Feridos_leves'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2017 ok")
acid18rj_copy['Feridos_leves'] =  pd.to_numeric(acid18rj_copy['Feridos_leves'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2018 ok")
acid19rj_copy['Feridos_leves'] =  pd.to_numeric(acid19rj_copy['Feridos_leves'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2019 ok")
acid20rj_copy['Feridos_leves'] =  pd.to_numeric(acid20rj_copy['Feridos_leves'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2020 ok")
```

```
Conversão Acid 2017 ok
Conversão Acid 2018 ok
Conversão Acid 2019 ok
Conversão Acid 2020 ok
```

[143]:
```
# Converter Feridos_graves para inteiro

acid17rj_copy['Feridos_graves'] =  pd.
 ↪to_numeric(acid17rj_copy['Feridos_graves'], errors="coerce").astype('int64')
print ("Conversão Acid 2017 ok")
acid18rj_copy['Feridos_graves'] =  pd.
 ↪to_numeric(acid18rj_copy['Feridos_graves'], errors="coerce").astype('int64')
print ("Conversão Acid 2018 ok")
acid19rj_copy['Feridos_graves'] =  pd.
 ↪to_numeric(acid19rj_copy['Feridos_graves'], errors="coerce").astype('int64')
print ("Conversão Acid 2019 ok")
acid20rj_copy['Feridos_graves'] =  pd.
 ↪to_numeric(acid20rj_copy['Feridos_graves'], errors="coerce").astype('int64')
print ("Conversão Acid 2020 ok")
```

```
Conversão Acid 2017 ok
Conversão Acid 2018 ok
Conversão Acid 2019 ok
Conversão Acid 2020 ok
```

[144]:
```
# Converter Obitos para inteiro

acid17rj_copy['Obitos'] =  pd.to_numeric(acid17rj_copy['Obitos'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2017 ok")
acid18rj_copy['Obitos'] =  pd.to_numeric(acid18rj_copy['Obitos'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2018 ok")
acid19rj_copy['Obitos'] =  pd.to_numeric(acid19rj_copy['Obitos'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2019 ok")
acid20rj_copy['Obitos'] =  pd.to_numeric(acid20rj_copy['Obitos'],␣
 ↪errors="coerce").astype('int64')
print ("Conversão Acid 2020 ok")
```

```
Conversão Acid 2017 ok
Conversão Acid 2018 ok
Conversão Acid 2019 ok
Conversão Acid 2020 ok
```

[176]:
```
# Verificou-se todas as bases para garantir que estivessem com o mesmo type nas␣
 ↪colunas
# Ex: infra20rj_copy2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 901659 entries, 0 to 901658
Data columns (total 12 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Data         901659 non-null  datetime64[ns]
 1   UF           901659 non-null  object
 2   BR           901659 non-null  object
 3   Codigo       901659 non-null  object
 4   Descricao    901659 non-null  object
 5   Hora         901659 non-null  object
 6   Responsavel  901659 non-null  object
 7   Tipo         901659 non-null  object
 8   Periodo      901659 non-null  object
 9   Mes          901659 non-null  object
 10  ano          901659 non-null  object
 11  mes_ano      901659 non-null  object
dtypes: datetime64[ns](1), object(11)
memory usage: 82.5+ MB
```

31 Agora vamos unir as planilhas de infrações e criar apenas uma geral para o estado de RJ

[152]:
```
#Verificou-se o shape de todos para garantir que estivessem com o mesmo shape
infra20rj_copy.shape
```

[152]:
```
(901659, 12)
```

[150]:
```
# Agora que checamos tudo, podemos concatenar as bases
# Aqui concatenamos INFRACOES
infra_rj = pd.concat([infra17rj_copy, infra18rj_copy, infra19rj_copy,␣
 ↪infra20rj_copy], join = 'inner')
print("Concatenação Infrações RJ concluída")
```

```
Concatenação Infrações RJ concluída
```

[147]:
```
# Concatenando ACIDENTES
acid_rj = pd.concat([acid17rj_copy, acid18rj_copy, acid19rj_copy,␣
 ↪acid20rj_copy], join = 'inner')
print("Concatenação Acidentes RJ concluída")
```

```
Concatenação Acidentes RJ concluída
```

[153]:
```
#Verificaremos o shape, para garantir que todas as linhas estão presentes
infra_rj.shape
```

[153]:
```
(3590450, 12)
```

[154]:
```
#Verificaremos o shape, para garantir que todas as linhas estão presentes
acid_rj.shape
```

[154]:
```
(72068, 16)
```

[155]:
```
acid_rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 72068 entries, 0 to 19383
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Data            72068 non-null  datetime64[ns]
 1   Dia_semana      72068 non-null  object
 2   UF              72068 non-null  object
 3   BR              72068 non-null  object
 4   Causa           72068 non-null  object
 5   Tipo_acidente   72068 non-null  object
 6   Classificacao   72068 non-null  object
 7   Ilesos          72068 non-null  int64
 8   Feridos_leves   72068 non-null  int64
```

```
 9   Feridos_graves  72068 non-null  int64
 10  Obitos          72068 non-null  int64
 11  Hora            72068 non-null  object
 12  Periodo         72068 non-null  object
 13  ano             72068 non-null  object
 14  mes_ano         72068 non-null  object
 15  Mes             72068 non-null  object
dtypes: datetime64[ns](1), int64(4), object(11)
memory usage: 9.3+ MB
```

[156]:
```
infra_rj.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3590450 entries, 0 to 901658
Data columns (total 12 columns):
 #   Column       Dtype
---  ------       -----
 0   Data         datetime64[ns]
 1   UF           object
 2   BR           object
 3   Codigo       object
 4   Descricao    object
 5   Hora         object
 6   Responsavel  object
 7   Tipo         object
 8   Periodo      object
 9   ano          object
 10  mes_ano      object
 11  Mes          object
dtypes: datetime64[ns](1), object(11)
memory usage: 356.1+ MB
```

32 Agora exportaremos essas bases para um arquivo

[148]:
```
# Exportação Acidentes RJ
acid_rj.to_excel(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.TCC\Final\ACID_RJ_FIN.
 ↪xlsx", index = False, header=True)
print("Exportação concluída")
```

```
Exportação concluída
```

[151]:
```
# Exportação Infrações RJ
infra_rj.to_csv(r"E:\CAMILA DRIVE\02.BIG_DATA_PUC\13.TCC\Final\INFRA_RJ_FIN.
 ↪csv", sep = ',', index = False, header=True)
print("Exportação concluída")
```

```
Exportação concluída
```