Sumário

[**1. Contextualização (Cenário do Projeto) 3**](#_gjdgxs)

[**2. Arquitetura do Projeto 3**](#_30j0zll)

[**3. Planejamento 3**](#_1fob9te)

[**4. Coleta de Dados 4**](#_2et92p0)

[**5. Preparação e Análise dos Dados 4**](#_1t3h5sf)

[**6. Modelagem e Treinamento 6**](https://docs.google.com/document/d/1RH1D8Quq_ANG1tWOU3eGfKUG9pgq6PZQ/edit#heading=h.3rdcrjn)

[**7. Otimização do Modelo 7**](#_4d34og8)

[**8. Integração 7**](#_17dp8vu)

[**9. Referências 8**](https://docs.google.com/document/d/1RH1D8Quq_ANG1tWOU3eGfKUG9pgq6PZQ/edit#heading=h.26in1rg)

**1. Contextualização (Cenário do Projeto)**

Após o incidente de segurança que expôs diversos dados sigilosos de clientes que o banco de Tóquio sofreu, foram designados vários inquéritos. O banco foi julgado e condenado a pagar uma multa milionária, onde o juiz compreendeu que não aconteceu um ato de má fé, mas que demonstrou negligência e portanto, condenou o banco a ressarcir os clientes por danos morais e também reverteu parte da multa para fins sociais. Para isso, o banco solicitou a diversos analistas e diretores o auxílio na tomada de decisões.

A nossa equipe é uma das responsáveis por realizar um estudo de Data Science, que foi definido após diversas reuniões com os diretores do banco para contribuir na tomada de decisões para investimento em uma atuação com fins sociais.

**2. Arquitetura do Projeto**

Com base no cenário apresentado pelo banco de Tóquio, a arquitetura apresentada pela disciplina que será seguida no projeto está representada na figura a seguir, a qual demonstra as principais atividades que serão realizadas durante todo o desenvolvimento do projeto.

Diagram

Description automatically generated

Fonte: Template para material de apoio da disciplina Oficina Maker.

**3. Planejamento**

A primeira etapa do projeto é o planejamento, onde o grupo busca entender o contexto apresentado pelo problema a ser solucionado durante o projeto e verificar quais serão os principais caminhos a serem seguidos nas próximas semanas. Além de coletar todos os requisitos necessários para a definição do escopo e fazer a seleção do dataset e a construção do cronograma detalhado.

* **Cronograma Detalhado:**

Graphical user interface, application

Description automatically generated

* **Descrição do dataset:**
  + *fonte/origem;*

O dataset selecionado foi o [SiGesGuarda](https://www.curitiba.pr.gov.br/dadosabertos/busca/?termo=sigesguarda), disponibilizado pela Prefeitura Municipal de Curitiba no Estado do Paraná – Brasil.

* + *apresentação do cenário (dataset);*

O dataset é composto por dados das ocorrências que são recebidas e atendidas pela Guarda Municipal de Curitiba. O órgão responsável é a Defesa Social e Trânsito. O principal responsável é Sergio Roberto da Silva Cruz. O dataset possui uma frequência de atualização mensal, sendo um espectro temporal de 2009 até o momento da extração.

* + *tipo do dataset (semi ou não estruturado);*

O dataset é semi estruturado, pois possui uma estrutura minimamente definida em arquivos csv ou xls, mas é necessário fazer um trabalho de tratamento, transformação e limpeza para que os dados estejam prontos para serem explorados e analisados.

* + *descrição do dataset (número de instâncias, variáveis, tipos de dado etc.);*

O csv selecionado para análise que é referente ao mês de outubro de 2022 possui ao todo 35 colunas x 419072 linhas, com 8657547 dados. O dataset todo é composto por dados do tipo object, mas ao interpretar podemos ver alguns dados que podem ser transformados para tipos como datetime e boolean, por exemplo.

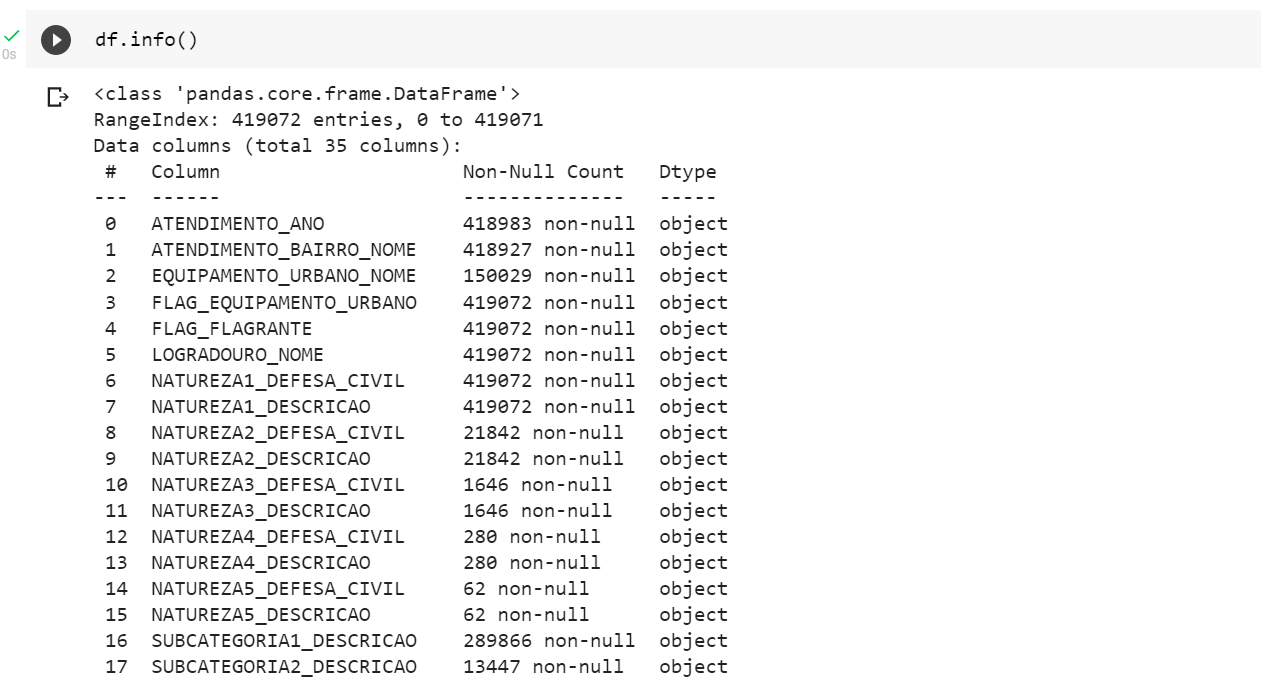
Alguns tipos de variáveis também podem ser percebidas: quantitativa contínua, como data, qualitativa nominal, como região e a qualitativa ordinal, ordinal como tipo de ocorrência.

| RangeIndex: 419072 entries, 0 to 419071 Data columns (total 35 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----   0 ATENDIMENTO\_ANO 418983 non-null object  1 ATENDIMENTO\_BAIRRO\_NOME 418927 non-null object  2 EQUIPAMENTO\_URBANO\_NOME 150029 non-null object  3 FLAG\_EQUIPAMENTO\_URBANO 419072 non-null object  4 FLAG\_FLAGRANTE 419072 non-null object  5 LOGRADOURO\_NOME 419072 non-null object  6 NATUREZA1\_DEFESA\_CIVIL 419072 non-null object  7 NATUREZA1\_DESCRICAO 419072 non-null object  8 NATUREZA2\_DEFESA\_CIVIL 21842 non-null object  9 NATUREZA2\_DESCRICAO 21842 non-null object  10 NATUREZA3\_DEFESA\_CIVIL 1646 non-null object  11 NATUREZA3\_DESCRICAO 1646 non-null object  12 NATUREZA4\_DEFESA\_CIVIL 280 non-null object  13 NATUREZA4\_DESCRICAO 280 non-null object  14 NATUREZA5\_DEFESA\_CIVIL 62 non-null object  15 NATUREZA5\_DESCRICAO 62 non-null object  16 SUBCATEGORIA1\_DESCRICAO 289866 non-null object  17 SUBCATEGORIA2\_DESCRICAO 13447 non-null object  18 SUBCATEGORIA3\_DESCRICAO 939 non-null object  19 SUBCATEGORIA4\_DESCRICAO 135 non-null object  20 SUBCATEGORIA5\_DESCRICAO 31 non-null object  21 OCORRENCIA\_ANO 419072 non-null object  22 OCORRENCIA\_CODIGO 419072 non-null object  23 OCORRENCIA\_DATA 419072 non-null object  24 OCORRENCIA\_DIA\_SEMANA 419072 non-null object  25 OCORRENCIA\_HORA 419072 non-null object  26 OCORRENCIA\_MES 419072 non-null object  27 OPERACAO\_DESCRICAO 306849 non-null object  28 ORIGEM\_CHAMADO\_DESCRICAO 419072 non-null object  29 REGIONAL\_FATO\_NOME 419061 non-null object  30 SECRETARIA\_NOME 419072 non-null object  31 SECRETARIA\_SIGLA 419072 non-null object  32 SERVICO\_NOME 419072 non-null object  33 SITUACAO\_EQUIPE\_DESCRICAO 293395 non-null object  34 NUMERO\_PROTOCOLO\_156 12110 non-null object dtypes: object(35) |
| --- |

* + *manipulação prévia do dataset;*

| **import** numpy **as** np **import** pandas **as** pd **import** matplotlib.pyplot **as** plt **import** seaborn **as** sns **import** numpy **as** np **from** scipy **import** stats **as** sp  df = pd.read\_csv("/content/2022-10-01\_sigesguarda\_-\_Base\_de\_Dados.csv", sep=';', encoding='latin-1')  df.head()  df.describe()  df.info() |
| --- |







* + *descrição das atividades realizadas no dataset;*

A primeira manipulação no dataset foi a exclusão da primeira linha vazia que havia e também o acréscimo da coluna QUANTIDADE\_OCORRENCIA populada com o valor 1 para cada linha, já que cada linha representa o valor de 1 ocorrência, que irá auxiliar nas somas e agrupamentos realizadas no Python.



Após essa etapa carregou-se o csv no Google Colab para iniciar a etapa de trabalho diretamente na linguagem de programação Python. Para ler o csv importou-se a biblioteca de pandas que permite a manipulação dos dados e utilizou-se a função read\_csv() que lê o arquivo csv.

A primeira função utilizada para observar os dados foi a head() que apresenta por default os 5 primeiros registros de cada coluna do dataframe e assim pode-se verificar se os dados foram carregados da forma correta e separados em colunas como em um arquivo excel.

A segunda função é a info() que mostra a quantidade de registros no arquivo, bem como a quantidade de colunas e dados não nulos em cada uma das colunas e também o tipo de dado presente em cada.

A terceira função para ter uma visão prévia dos dados é a describe() que apresenta algumas estatísticas de cada uma das colunas presentes no dataset.

**4. Coleta de Dados**

* **Descrição dos experimentos iniciais:**
  + *seleção das variáveis relevantes para o projeto;*

O dataset selecionado possui 35 colunas no total e destas colunas foram selecionadas 12 variáveis, sendo elas:

ATENDIMENTO\_ANO,

OCORRENCIA\_ANO,

ATENDIMENTO\_BAIRRO\_NOME,

FLAG\_FLAGRANTE,

NATUREZA1\_DESCRICAO,

OCORRENCIA\_DATA,

OCORRENCIA\_DIA\_SEMANA,

REGIONAL\_FATO\_NOME,

QUANTIDADE\_OCORRENCIA ,

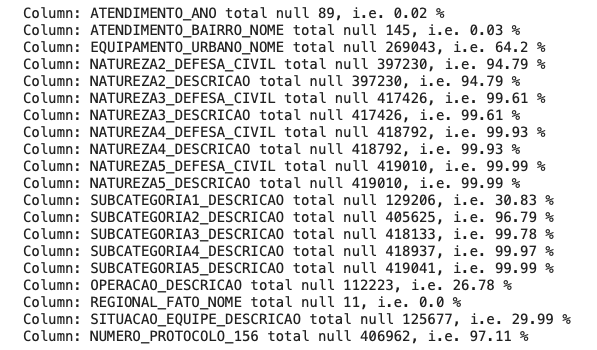
OCORRENCIA\_HORA,

OCORRENCIA\_MES e

OCORRENCIA\_DIA.

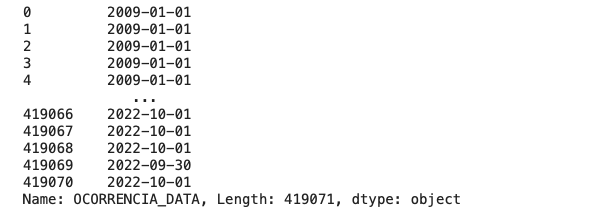
* + *descrição dos experimentos iniciais sobre o dataset do projeto;*
* **Contagem de dados nulos**

| Contagem de dados nulos **def** **contagemNulos**(tabela):  **for** col **in** tabela.columns:  **if** tabela[col].isnull().sum():  total\_null=tabela[col].isnull().sum()   print('Column: {} total null {}, i.e. {} %'.format(col,total\_null,round(total\_null\*100/len(df),2)))  dfContagemNulos = contagemNulos(df) |
| --- |



* **Convertendo colunas de horas**

| df['OCORRENCIA\_DATA'] = pd.to\_datetime(df.OCORRENCIA\_DATA, format='%Y-%m-%d')  df['OCORRENCIA\_DATA'] = df['OCORRENCIA\_DATA'].dt.strftime('%Y-%m-%d')  df['OCORRENCIA\_DATA'] |
| --- |



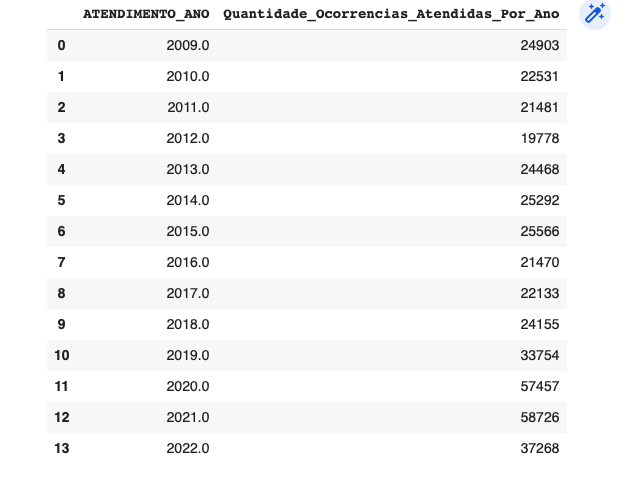
* **Agrupando as colunas e criando o novo dataset**

| df2 = df.groupby(['ATENDIMENTO\_ANO','OCORRENCIA\_ANO', "ATENDIMENTO\_BAIRRO\_NOME", "FLAG\_FLAGRANTE", "NATUREZA1\_DESCRICAO", "OCORRENCIA\_DATA", "OCORRENCIA\_DIA\_SEMANA", "REGIONAL\_FATO\_NOME" ])['QUANTIDADE\_OCORRENCIA'].sum().reset\_index()  df2 |
| --- |



* **Quantidade de ocorrências atendidas por ano (ATENDIMENTO\_ANO)**

| -- Quantidade de ocorrencias atendidas por ano (ATENDIMENTO\_ANO)  ocorrenciasAtendidasPorAno = df.groupby(['ATENDIMENTO\_ANO'])['QUANTIDADE\_OCORRENCIA'].sum().reset\_index()  ocorrenciasAtendidasPorAno = ocorrenciasAtendidasPorAno.rename({'QUANTIDADE\_OCORRENCIA': 'Quantidade\_Ocorrencias\_Atendidas\_Por\_Ano'}, axis=1)  ocorrenciasAtendidasPorAno |
| --- |

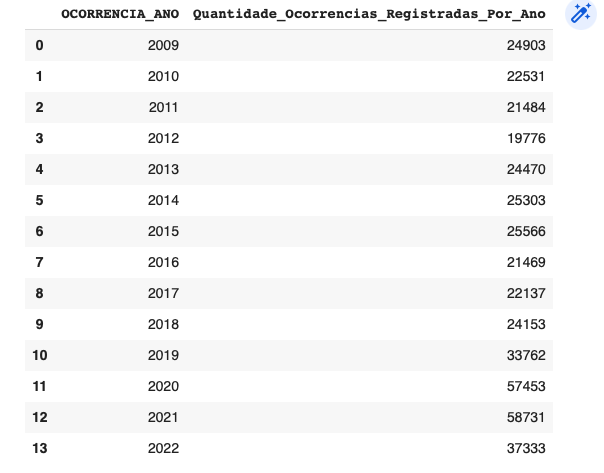


| ocorrenciasAtendidasPorAno.describe() |
| --- |

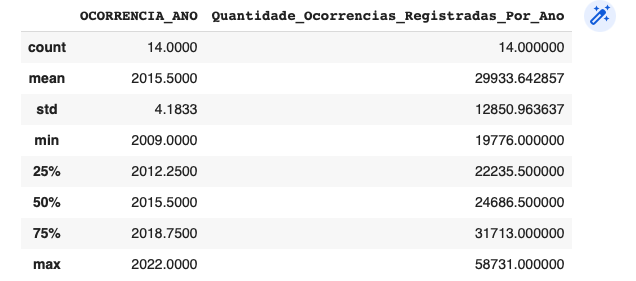


* **Quantidade de ocorrências recebidas por ano (OCORRENCIA\_ANO)**

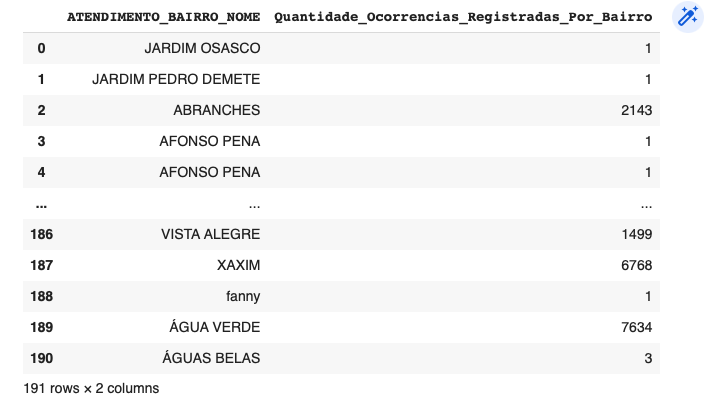
| -- Quantidade de ocorrências recebidas por ano (OCORRENCIA\_ANO)  ocorrenciasRegistradasPorAno = df.groupby(['OCORRENCIA\_ANO'])['QUANTIDADE\_OCORRENCIA'].sum().reset\_index()  ocorrenciasRegistradasPorAno = ocorrenciasRegistradasPorAno.rename({'QUANTIDADE\_OCORRENCIA': 'Quantidade\_Ocorrencias\_Registradas\_Por\_Ano'}, axis=1)  ocorrenciasRegistradasPorAno |
| --- |



| ocorrenciasRegistradasPorAno.describe() |
| --- |



| -- Quantidade de ocorrencias por bairro (ATENDIMENTO\_BAIRRO\_NOME)  ocorrenciasPorBairro = df.groupby(['ATENDIMENTO\_BAIRRO\_NOME'])['QUANTIDADE\_OCORRENCIA'].sum().reset\_index()  ocorrenciasPorBairro = ocorrenciasPorBairro.rename({'QUANTIDADE\_OCORRENCIA': 'Quantidade\_Ocorrencias\_Registradas\_Por\_Bairro'}, axis=1)  ocorrenciasPorBairro |
| --- |

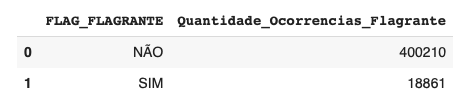


| ocorrenciasPorBairro.describe() |
| --- |



* **Quantidade de ocorrências com flagrante (FLAG\_FLAGRANTE)**

| -- Quantidade de ocorrencias com flagrante (FLAG\_FLAGRANTE)  ocorrenciasFlagrante = df.groupby(['FLAG\_FLAGRANTE'])['QUANTIDADE\_OCORRENCIA'].sum().reset\_index()  ocorrenciasFlagrante = ocorrenciasFlagrante.rename({'QUANTIDADE\_OCORRENCIA': 'Quantidade\_Ocorrencias\_Flagrante'}, axis=1)  ocorrenciasFlagrante |
| --- |

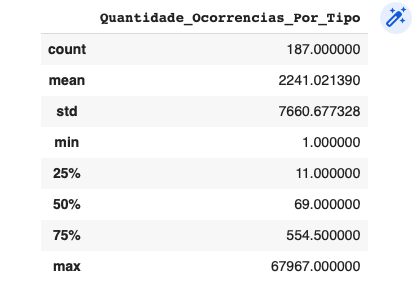


* **Quantidade de ocorrências por tipo (NATUREZA1\_DESCRICAO)**

| -- Quantidade de ocorrencias por tipo (NATUREZA1\_DESCRICAO)  ocorrenciasPorTipo = df.groupby(['NATUREZA1\_DESCRICAO'])['QUANTIDADE\_OCORRENCIA'].sum().reset\_index()  ocorrenciasPorTipo = ocorrenciasPorTipo.rename({'QUANTIDADE\_OCORRENCIA': 'Quantidade\_Ocorrencias\_Por\_Tipo'}, axis=1)  ocorrenciasPorTipo |
| --- |

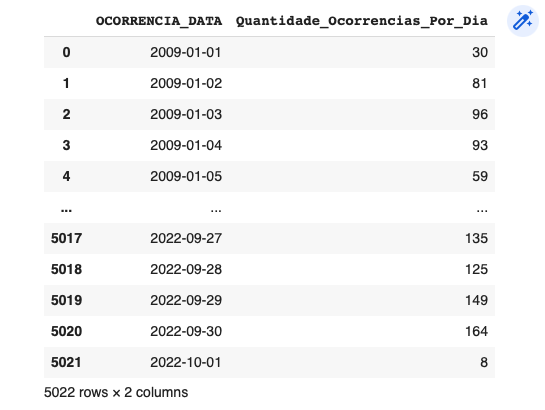


| ocorrenciasPorTipo.describe() |
| --- |



* **Quantidade de ocorrências por dia (OCORRENCIA\_DATA)**

| -- Quantidade de ocorrencias por dia (OCORRENCIA\_DATA)  ocorrenciasPorDia = df.groupby(['OCORRENCIA\_DATA'])['QUANTIDADE\_OCORRENCIA'].sum().reset\_index()  ocorrenciasPorDia = ocorrenciasPorDia.rename({'QUANTIDADE\_OCORRENCIA': 'Quantidade\_Ocorrencias\_Por\_Dia'}, axis=1)  ocorrenciasPorDia |
| --- |

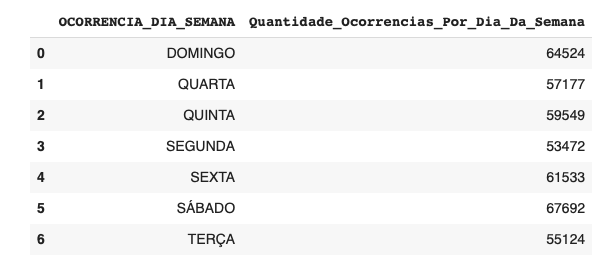


| ocorrenciasPorDia.describe() |
| --- |

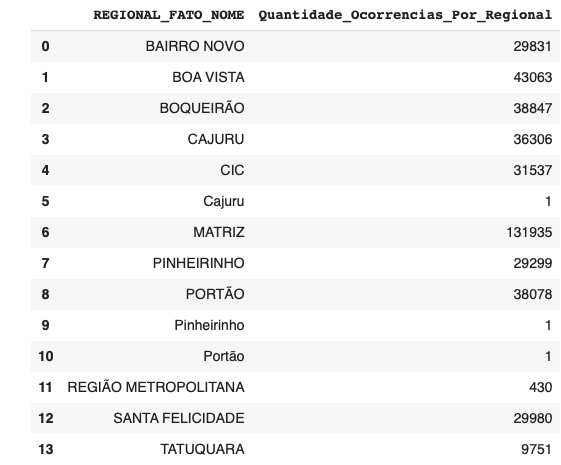


* **Quantidade de ocorrências por dia da semana (OCORRENCIA\_DIA\_SEMANA)**

| -- Quantidade de ocorrencias por dia da semana (OCORRENCIA\_DIA\_SEMANA)  ocorrenciasPorDiaDaSemana = df.groupby(['OCORRENCIA\_DIA\_SEMANA'])['QUANTIDADE\_OCORRENCIA'].sum().reset\_index()  ocorrenciasPorDiaDaSemana = ocorrenciasPorDiaDaSemana.rename({'QUANTIDADE\_OCORRENCIA': 'Quantidade\_Ocorrencias\_Por\_Dia\_Da\_Semana'}, axis=1)  ocorrenciasPorDiaDaSemana |
| --- |



| ocorrenciasPorRegional = df.groupby(['REGIONAL\_FATO\_NOME'])['QUANTIDADE\_OCORRENCIA'].sum().reset\_index()  ocorrenciasPorRegional = ocorrenciasPorRegional.rename({'QUANTIDADE\_OCORRENCIA': 'Quantidade\_Ocorrencias\_Por\_Regional'}, axis=1)  ocorrenciasPorRegional |
| --- |



Para realizar a segunda etapa da análise exploratória que são algumas principais perguntas que o grupo construiu e também a etapa de modelagem, construiu-se um novo notebook e foi carregado novamente um dataframe do dataset e realizado algumas transformações como apresenta-se a seguir.

Nesta etapa utiliza-se novamente as funções info() e describe() para verificar as informações e métricas do dataset e também a função median() que retorna a mediana de um conjunto de dados e a var() para a variância, bem como a função value\_counts() que faz a contagem dos valores presentes.

* **Importação das bibliotecas**

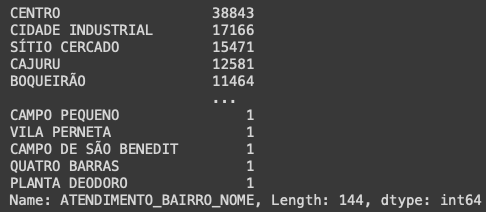
| **import** numpy **as** np **import** pandas **as** pd # importando o pandas para manipularmos o dataset **import** seaborn **as** sns # importando o Seaborn para visualizar o comportamento dos dados **import** matplotlib.pyplot **as** plt # importando o Matplotlib para o elbow method **from** pandas\_profiling **import** ProfileReport # importando o pandas-profiling para fazer o profile do dataset **from** scipy **import** stats **as** sp **from** sklearn.model\_selection **import** train\_test\_split # utilizado para o split entre treinamento e teste **from** sklearn.neighbors **import** KNeighborsRegressor # KNN para regressão **from** sklearn.linear\_model **import** LinearRegression # Regressão linear **from** sklearn.svm **import** SVR # SVM para regressão **from** sklearn.decomposition **import** PCA # PCA como aprendizagem não-supervisionada **from** sklearn.preprocessing **import** RobustScaler # utilizado para que todas as entradas estejam na mesma escala numérica **from** sklearn.preprocessing **import** StandardScaler **from** pandas.core.frame **import** DataFrame **from** matplotlib **import** pyplot **as** plt %matplotlib inline   df = pd.read\_csv('/content/teste.csv', sep=';', encoding='ISO-8859-1')  df |
| --- |



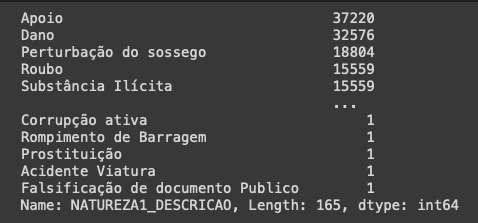
| df.info() |
| --- |



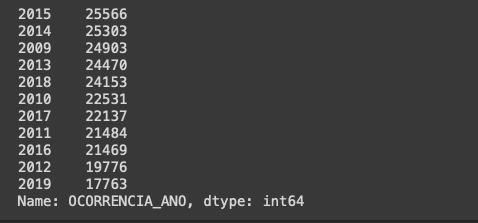
| # \*\*Informação do dataset\*\*  df['ATENDIMENTO\_BAIRRO\_NOME'].value\_counts() |
| --- |



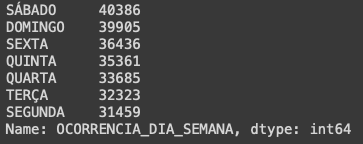
| df['NATUREZA1\_DESCRICAO'].value\_counts() |
| --- |



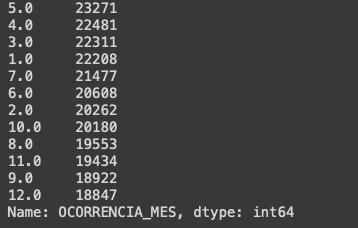
| df['OCORRENCIA\_ANO'].value\_counts() |
| --- |



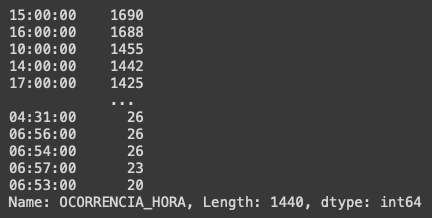
| df['OCORRENCIA\_DIA\_SEMANA'].value\_counts() |
| --- |



| df['OCORRENCIA\_MES'].value\_counts() |
| --- |



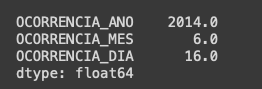
| df['OCORRENCIA\_HORA'].value\_counts() |
| --- |



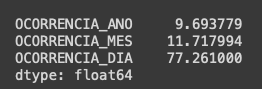
| df.describe() |
| --- |



| #.median() Função Pandas retorna a mediana dos valores para o eixo solicitado. df.median() |
| --- |



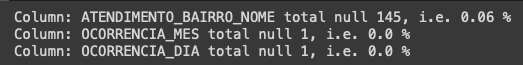
| #.var() calcula a variância no Pandas através da função df.var() |
| --- |



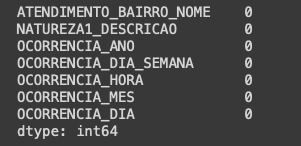
A próxima parte é realizado novamente o tratamento de dados, onde realiza-se a limpeza de dados nulos, converte datas, colunas para inteiro e também algumas transformações e criações de label. Também utiliza-se a função std() para retornar o desvio padrão das colunas numéricas.

* **Limpando dados nulos**

| # \*\*Tratamento de dados\*\*  ## \*Limpando dados nulos\* **for** col **in** df.columns:  **if** df[col].isnull().sum():  total\_null=df[col].isnull().sum()   print('Column: {} total null {}, i.e. {} %'.format(col,total\_null,round(total\_null\*100/len(df),2))) |
| --- |

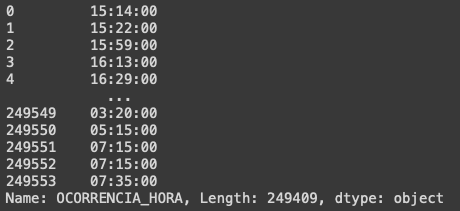


| #Limpando dados nulos df.dropna(inplace = **True**) df.isnull().sum() |
| --- |

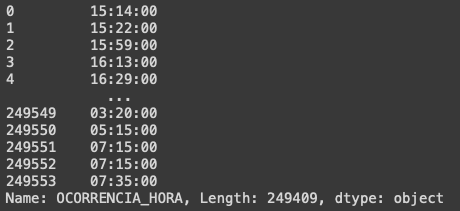


* **Convertendo para int e datetime**

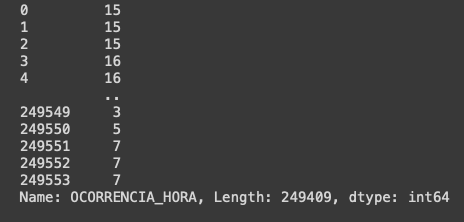
| ## \*convertendo para int\*  ### \*OCORRENCIA\_HORA\*  #converteu para datetime df['OCORRENCIA\_HORA'] = pd.to\_datetime(df['OCORRENCIA\_HORA'])  df['OCORRENCIA\_HORA'].dt.time |
| --- |



| df['OCORRENCIA\_HORA'] = df['OCORRENCIA\_HORA'].dt.strftime('%H')  df['OCORRENCIA\_HORA'] = df['OCORRENCIA\_HORA'].astype(str).astype(int)  df['OCORRENCIA\_HORA'] |
| --- |

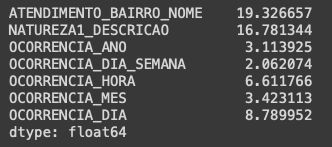


| df['OCORRENCIA\_HORA'] = df['OCORRENCIA\_HORA'].dt.strftime('%H')  df['OCORRENCIA\_HORA'] = df['OCORRENCIA\_HORA'].astype(str).astype(int)  df['OCORRENCIA\_HORA'] |
| --- |



* **Criação de labels e cálculo desvio padrão**

| ### ATENDIMENTO\_BAIRRO\_NOME  df['ATENDIMENTO\_BAIRRO\_NOME'] = df['ATENDIMENTO\_BAIRRO\_NOME'].replace({'CIDADE INDUSTRIAL':1, 'FAZENDINHA':2, 'UBERABA':3, 'SÍTIO CERCADO':4, 'TATUQUARA':5, 'SANTA CÂNDIDA':6, 'BOQUEIRÃO':7, 'CENTRO':8, 'BOA VISTA':9, 'TABOÃO':10, 'XAXIM':11, 'PILARZINHO':12, 'REBOUÇAS':13, 'ÁGUA VERDE':14, 'BATEL':15, 'NOVO MUNDO':16, 'ALTO BOQUEIRÃO':17, 'CAPÃO RASO':18, 'JARDIM BOTÂNICO':19, 'PORTÃO':20, 'ORLEANS':21, 'SANTA FELICIDADE':23, 'CASCATINHA':24, 'CAPÃO DA IMBUIA':25, 'BARREIRINHA':26, 'SEMINÁRIO':27, 'CAMPO COMPRIDO':28, 'PRADO VELHO':29, 'PINHEIRINHO':30, 'BUTIATUVINHA':31, 'CAMPINA DO SIQUEIRA':32, 'CAJURU':33, 'SÃO FRANCISCO':34, 'CENTRO CÍVICO':35, 'SÃO BRAZ':36, 'UMBARÁ':37, 'CAXIMBA':38, 'JARDIM SOCIAL':39, 'BACACHERI':40, 'CAMPO DE SANTANA':41, 'SANTO INÁCIO':42, 'JARDIM DAS AMÉRICAS':43, 'LINDÓIA':44, 'GANCHINHO':45, 'PAROLIN':46, 'ABRANCHES':47, 'SÃO JOÃO':48, 'ATUBA':49, 'TARUMÃ':50, 'ALTO DA RUA XV':51, 'MOSSUNGUÊ':52, 'TINGUI':53, 'BIGORRILHO':54, 'BAIRRO ALTO':55, 'HAUER':56, 'VILA IZABEL':57, 'CABRAL':58, 'BOM RETIRO':59, 'GUAÍRA':60, 'CACHOEIRA':61, 'AUGUSTA':62, 'CRISTO REI':63, 'AHÚ':64, 'ALTO DA GLÓRIA':65, 'GUABIROTUBA':66, 'MERCÊS':67, 'SANTA QUITÉRIA':68, 'SÃO MIGUEL':69, 'SÃO LOURENÇO':70, 'FANNY':71, 'JUVEVÊ':72, 'VISTA ALEGRE':73, 'HUGO LANGE':74, 'RIVIERA':75, 'LAMENHA PEQUENA':76, 'INDICAÇÕES CANCELADA':77, 'BAIRRO NAO INFORMADO':78, 'BAIRRO FICTÍCIO':79, 'fanny':80, 'TINGÜI':81, 'CIDADE JARDIM':82, 'VENEZA':83, 'PLANTA MEIRELES':84, 'TANGUA':85, 'MONTE REY':86, 'JD EUROPA':87, 'BORDA DO CAMPO':88, 'JARDIM BOA VISTA':89, 'SÃO JOSE':90, 'JARDIM COLONIAL':91, 'MENINO DEUS':92, 'SÃO JUDAS TADEU':93, 'VILA MARIA ANTONIETA':94, 'MARIA ANTONIETA':95, 'SANTO ANTONIO':96, 'COLOMBO':97, 'CANGUIRI':98, 'NÃO ENCONTRADO':99, 'FERRARIA':100, 'SÃO CRISTOVÃO':101, 'JD SUISSA':102, 'VILA FORMOSA':103, 'FORMOSO':104, 'SÃO PEDRO':105, 'SAO JOSE DOS PINHAIS':106, 'CAMPO PEQUENO':107, 'PINHAIS':108, 'VILA PERNETA ':109, 'SEM DADOS':110, 'CAMPO DE SÃO BENEDIT':111, 'QUATRO BARRAS':112, 'LOT. MARINONI':113, 'SÃO JORGE':114, 'BAIRRO NÃO LOCALIZAD':115, 'BRAGA':116, 'JARDIM LOANDA':117, 'NÃO INFORMADO ':118, 'SANTA TEREZINHA':119, 'SANTA TERESINHA':120, 'JARDIM WEISSOPOLIS':121, 'SITIO DAS PALMEIRAS':122, 'CAMPO PEQUENO ':123, 'NI':124, 'THOMAS COELHO':125, 'NF':126, 'SÃO THOMAS':127, 'JARDIM INDUSTRIAL':128, 'ROÇA NEGRA':129, 'SÃO THOMAZ':130, 'GRALHA AZUL':131, 'MARACANÃ':132, 'VILA BANCÁRIA':133, 'JARDIM BOM PASTOR':134, 'SAO GERONIMO':135, 'RIO VERDE':136, 'JD IPE':137, 'IGUAÇÚ 1':138, 'AGUAS BELAS':139, 'ÁGUAS BELAS':140, 'IGUAÇU 01':141, 'ESTADOS':142, 'CIC':143, 'JR TAISA':144, 'PLANTA DEODORO':145, 'MAUA':146, 'COLONIA FARIA':147, 'NAÇÕES':148, 'JARDIM SANTA MÔNICA':149, 'LOTEAMENTO SÃO GERÔN':150, 'TAMANDARE ':151, 'CAMPO LARGO':152, 'BOQUEIRÃO ':153, 'JARDIM BELA VISTA':154, 'ESTANCIA PINHAIS ':155, 'COLONIA SAO VENANCIO':156, 'FRANCISCO GORSKI':157, 'OSASCO':158, 'BARIGUI':159, 'GUATUPE ':160, 'PARQUE DAS NASCENTES':161, 'CENTRO ':162, 'JD. ORESTES THÁ':163, 'PARQUE DAS FONTES':164, 'PINEVILLE':165, 'BORDA DO CAMPO ':166, ' JARDIM OSASCO':167, 'JARDIM PRIMAVERA':168, 'JD DONA BELIZARIA':169, 'PIRAQUARA':170, 'JARDIM RAFAELA':171, 'BARRO PRETO':172, 'BELAS AGUAS':173, 'EUCALIPTOS':174, 'VILA GRAZIELA':175, 'CIDADE INDUSTRIAL DE':178, 'AFONSO PENA':179, 'PALMEIRINHA':180, 'IPE 2':181, 'SANTA MONICA':182, 'GUATUPE':183, 'AFONSO PENA ':184, 'SAO SEBASTIAO':185, 'MAUÁ':186, 'SÃO GERONIMO':187, 'OURO FINO':188, 'SANTO ANTÔNIO':189, 'CAMPINHA GRANDE DO S':190, ' JARDIM PEDRO DEMETE':191, 'ROÇA GRANDE':192, 'TINDIQUERA':193, 'SÃO BENEDITO':194})   df['ATENDIMENTO\_BAIRRO\_NOME'] = df['ATENDIMENTO\_BAIRRO\_NOME'].astype(str).astype(int)  ### NATUREZA1\_DESCRICAO  df['NATUREZA1\_DESCRICAO'] = df['NATUREZA1\_DESCRICAO'].replace({'Apoio':1, 'Alarmes':2, 'Invasão':3, 'Vistoria':4, 'Roubo':5, 'Perturbação do sossego':6, 'Trânsito':7, 'Risco de acidente/à vida (Defesa Civil)':8, 'Violação de Medida Protetiva Lei Maria da Penha':9, 'Dano':10, 'Lesão Corporal':11, 'Fundada Suspeita (Abordagem)':12, 'Substância Ilícita':13, 'Orientação':14, 'Alagamento':15, 'Animais':16, 'Furto':17, 'Desinteligência':18, 'Patrulha Maria da Penha':19, 'Atitude Suspeita':20, 'Atos obscenos/libidinosos':21, 'Vias de fato':22, 'Queima a céu aberto':23, 'Ameaça':24, 'Averiguação':25, 'Encaminhamento':26, 'Estupro':27, 'Saturação':28, 'Agressão física/verbal':29, 'AIFU':30, 'Escolta':31, 'Incêndio':32, 'Risco de acidente / à vida':33, 'Desacato':34, 'Paciente/usuário alterado':35, 'Veículo':36, 'Pesca em local proibido':37, 'Ronda':38, 'Destelhamento':39, 'Construção Irregular':40, 'Crime ambiental':41, 'Risco de desabamento / desmoronamento':42, 'Tentativa':43, 'Fornecimento de Lona':44, 'Suicídio':45, 'Obstrução de via':46, 'Substância Lícita':47, 'Depósito irregular':48, 'Corte irregular de árvore':49, 'Achado':50, 'Queda de árvore':51, 'Disparo de arma':52, 'Órgãos acionados':53, 'Averiguação (Defesa Civil)':54, 'Antecedentes Criminais - Verificação':55, 'Injúria':56, 'Desaparecimento':57, 'Manifestação':58, 'Seqüestro e cárcere privado':59, 'Arrastão':60, 'Deslizamenton de Terra':61, 'ZELADORIA URBANA':62, 'Desabamento':63, 'Devolução de coisa achada':64, 'Conduta inconveniente':65, 'Uso indevido do cartão transporte':66, 'Maus tratos à pessoas':67, 'Extravio de Equipamento':68, 'Porte Ilegal':69, 'Rixa':70, 'Erosão':71, 'Importunação\xa0sexual':72, 'Situação de risco':73, 'Queda de fios de energia':74, 'Estelionato':75, 'Desobediência':76, 'Racismo':77, 'Homicídio':78, 'Queda de galho':79, 'Homofobia':80, 'Descumprimento lei 15799/2021 COVID-19':81, 'Fuga de aluno/interno':82, 'Menores abordando transeuntes':83, 'Abandono de incapaz':84, 'Risco de queda de árvore':85, 'Retirada de invasão':86, 'Banho em local impróprio':87, 'Abuso de incapazes':88, 'Contrabando ou descaminho':89, 'Criança perdida/desaparecida':90, 'Extravio, sonegação ou inutilização de livro ou doc.':91, 'Resistência':92, 'Aliciamento de menor':93, 'Apropriação indébita':94, 'Proteção ao patrimônio':95, 'Infiltração':96, 'Roubo, furto, extravio, recuperação, apreensão de armas de fogo.':97, 'Receptação':98, 'Ataque de insetos':99, 'Fiscalizações e Orientações':100, 'Vazamento ou derramamento de Produto Perigoso ou Infectante':101, 'Falsidade ideológica (Falsa Identidade)':102, 'Câmera Off-Line':103, 'Poluição visual/ambiental':104, 'Óbito':105, 'Avaria em Equipamento/Patrimônio (não intencional)':106, 'Fuga de paciente':107, 'Moeda Falsa':108, 'Embriaguez':109, 'Queda de poste':110, 'Material abandonado':111, 'Calote':112, 'Quedas de objetos ou partes de construções':113, 'Acidente Viatura':114, 'Risco de queda de poste':105, 'Constrangimento ilegal':106, 'Comércio ambulante':107, 'Usar de uniforme, ou distintivo de função pública que não exerce':108, 'Envenenamento':109, 'Denúncia de bomba':110, 'Mendigar, por ociosidade ou cupidez':111, 'Extorsão':112, 'Atentado violento ao pudor':113, 'Verificação':114, 'Pragas Animais':115, 'Inundação/Enchente':116, 'Importunação ofensiva ao pudor':117, 'Jogo de Azar':118, 'Porte de artefato explosivo':119, 'Maus tratos a animais':120, 'Calúnia':121, 'Sedução':122, 'Violência arbitrária':123, 'Afogamento':124, 'Explosão':125, 'Câmeras de videomonitoramento':126, 'Bueiro aberto/sem tampa':127, 'Menor gazeando aula':128, 'Fornecimento de bebida alcoólica à menores':129, 'Vadiagem':130, 'Discriminação':131, 'Escrito ou objeto obsceno (panfletos pornográficos)':132, 'Favorecimento da prostituição':133, 'Peculato':134, 'Impedimento ou perturbação de cerimônia funerária':135, 'Risco de queda de fios de energia':136, 'Ataque cão feroz':137, 'Abandono de função':138, 'Uso indevido do telefone público':139, 'Aterro irregular':140, 'Risco de explosão':141, 'Obstrução da Atividade Policial':142, 'Bueiro entupido':143, 'Corrupção de menores':144, 'Queda de aeronave':145, 'Incendio/Explosão em edificação':146, 'Vilipêndio a cadáver':147, 'Risco de queda de galho':148, 'Prostituição':149, 'Violação de sepultura/túmulo':150, 'Fingir-se funcionário público':151, 'Trote Telefonico':152, 'Apologia de crime ou criminoso':153, 'Falsificação de documento Publico':154, 'Denuncia Improcedente':155, 'Quadrilha ou bando':156, 'Desabamento de Telhado/Cobertura':157, 'Exploração de menores':158, 'Queda de Muro':159, 'Abalo Sísmico':160, 'Omissão de socorro':161, 'Rompimento de Barragem':162, 'Liberação de pessoa presa/apreendida por recusa no recebimento pela DP':163, 'Venda proibida de produtos específicos à menores':164, 'Concussão':165, 'Charlatanismo':166, 'Difamação':167, 'RECUSAR SE IDENTIFICAR AO POLICIAL':168, 'Perseguição (stalking)':169, 'Enxurrada':170, 'Rufianismo':171, 'Incitação ao crime':172, 'Averiguação (COSEDI)':173, 'Queda de Revestimento de Fachadas':174, 'Corrupção ativa':175, 'Óbito (Defesa Civil)':176, 'Prevaricação':177})   df['NATUREZA1\_DESCRICAO'] = df['NATUREZA1\_DESCRICAO'].astype(str).astype(int)  ### OCORRENCIA\_DIA\_SEMANA  df['OCORRENCIA\_DIA\_SEMANA'] = df['OCORRENCIA\_DIA\_SEMANA'].replace({'DOMINGO':1, 'SEGUNDA':2, 'TERÇA':3, 'QUARTA':4, 'QUINTA':5, 'SEXTA':6, 'SÁBADO':7})   df['OCORRENCIA\_DIA\_SEMANA'] = df['OCORRENCIA\_DIA\_SEMANA'].astype(str).astype(int)  #.std() calcula o desvio padrão das colunas ou linhas numéricas df.std() |
| --- |



**5. Preparação e Análise dos Dados**

* **Análise exploratória do dataset do projeto:**
  + *seleção das ferramentas, técnicas e métricas de estatística descritiva; instalação, configuração e ajustes das ferramentas de análise estatística; resultado da estimativa estatística utilizando as medidas de posição e dispersão; descrição do protocolo para remoção de outliers; apresentação do processo de análise exploratória, medidas e métricas estatísticas utilizadas.*

Nessa próxima etapa realiza-se a análise descritiva e exploratória dos dados, para isso foram criadas e executadas algumas funções que calculam a média, mediana, variância, desvio padrão e quantis.

* **Média**

| # Análise descritiva de dados  tabelas = [ocorrenciasAtendidasPorAno, ocorrenciasRegistradasPorAno, ocorrenciasPorBairro, ocorrenciasFlagrante, ocorrenciasPorTipo, ocorrenciasPorDia, ocorrenciasPorDiaDaSemana, ocorrenciasPorRegional]  ### \*\*- Medidas de posição\*\*  #Média  **def** **calculaMedia**(tabela):   media = tabela.mean()   media = pd.DataFrame({'metricas':media.index, 'media':media.values})    **return** media  medias = [] **for** i **in** tabelas:   medias.append(calculaMedia(i))   dfMedia = pd.concat(medias) dfMedia |
| --- |



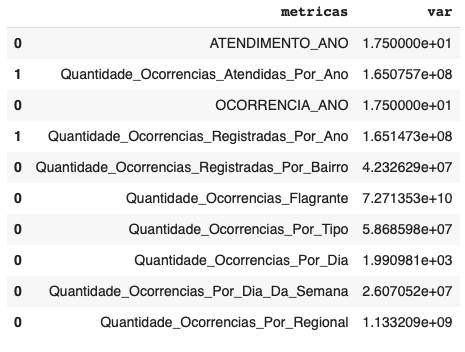
* **Mediana**

| #Mediana **def** **calculaMediana**(tabela):  mediana = tabela.median()  mediana = pd.DataFrame({'metricas':mediana.index, 'mediana':mediana.values})    **return** mediana medianas = [] **for** i **in** tabelas:   medianas.append(calculaMediana(i))   dfMediana = pd.concat(medianas) dfMediana |
| --- |



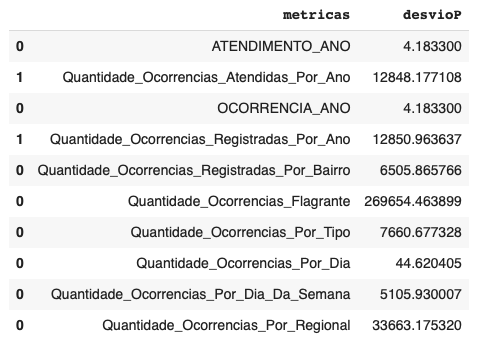
* **Variância**

| #Variância (valores ao quadrado) #Distância dos termos com relação a média #Quanto menor, melhor e mais próximo a média **def** **calculaVariancia**(tabela):  var = tabela.var()  var = pd.DataFrame({'metricas':var.index, 'var':var.values})    **return** var  variancias = [] **for** i **in** tabelas:   variancias.append(calculaVariancia(i))   dfVariancia = pd.concat(variancias) dfVariancia |
| --- |



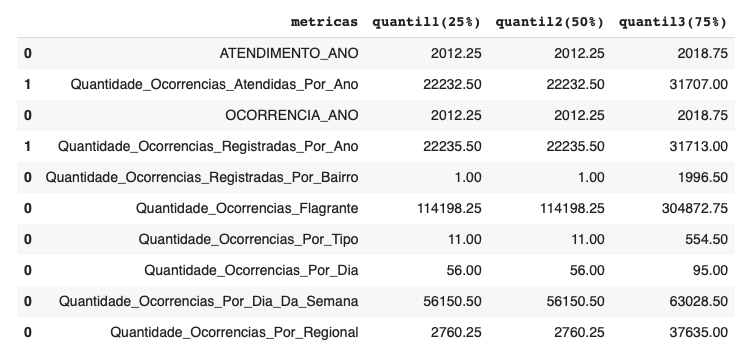
* **Desvio Padrão**

| #Desvio Padrão (√ Variância - mesma unidade media) #Distância dos termos com relação à média #Quanto menor, melhor e mais próximo a média  **def** **calculateDesvioPadrao**(tabela):   desvioP = tabela.std()   desvioP = pd.DataFrame({'metricas':desvioP.index, 'desvioP':desvioP.values})   **return** desvioP   desvios = [] **for** i **in** tabelas:   desvios.append(calculateDesvioPadrao(i))   dfDesvio = pd.concat(desvios) dfDesvio |
| --- |



* **Quantis**

| #Quantis **def** **calculaQuantil**(tabela):   q1 = tabela.quantile(0.25)  q1Df = pd.DataFrame({'metricas':q1.index, 'q1':q1.values})   q2 = tabela.quantile(0.50)  q2Df = pd.DataFrame({'metricas':q2.index, 'q2':q1.values})   q3 = tabela.quantile(0.75)  q3Df = pd.DataFrame({'metricas':q3.index, 'q3':q3.values})   quantile = pd.concat([q1Df['metricas'], q1Df['q1'], q2Df['q2'], q3Df['q3']], axis=1, ignore\_index=**True**)  quantile.columns = ['metricas', 'quantil1(25%)', 'quantil2(50%)', 'quantil3(75%)']    **return** quantile  quantis = [] **for** i **in** tabelas:   quantis.append(calculaQuantil(i))   dfQuantis = pd.concat(quantis) dfQuantis |
| --- |



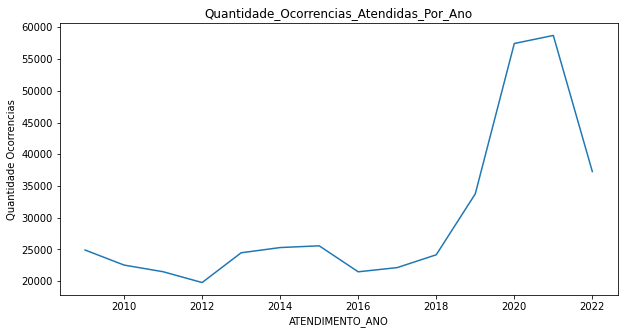
A etapa seguinte cria-se e executa-se funções para gráficos, para poder visualizar todas as variáveis que foram selecionadas e observadas anteriormente. Os gráficos são: gráfico de distribuição, histograma com bins, gráfico de pizza, gráfico de linha e boxplot.

| **def** **criaHistogramaComBins**(title, table, bins, color, collumn = None):   plt.figure(figsize=(10,5))   plt.title(title, fontsize =13)   **if** collumn **is** **None**:    plt.hist((table), bins = np.array(bins), alpha = 0.8, color = color)   **else**:   plt.hist((table[collumn]), bins = np.array(bins), alpha = 0.8, color = color)    plt.show  **def** **plotScatter**(collumnX, collumnY, table, title, color):    sns.set\_style('white')   plt.figure(figsize= (10, 10))   plt.title(title, fontsize = 13)   sns.scatterplot(x=collumnX, y=collumnY, data= table, color = color)   plt.show()  **def** **criaGraficoPizza**(table, y, label, title):  mycolors = ['plum', 'bisque']  plt.figure(figsize=(15,10), dpi=80)  plt.pie(table[y], labels = table[label], autopct = '%1.1f%%', colors = mycolors, frame = **False**)  #plt.legend()  plt.title(title)  plt.rcParams['axes.facecolor'] = 'white'  plt.show()  **def** **criaGraficoLinha**(table, x, y, title, xLabel, yLabel):  plt.figure(figsize=(10,5))  plt.plot(table[x], table[y])  plt.title(title)  plt.xlabel(xLabel)  plt.ylabel(yLabel)    plt.show()  **def** **criaGraficoDistribuicao**(table, coluna):  plt.figure(figsize=(10,5))  sns.distplot(table[coluna])  **def** **criaGraficoBoxplot**(table, x):  plt.figure(figsize=(10,5))  sns.boxplot(data=table, x=x) |
| --- |

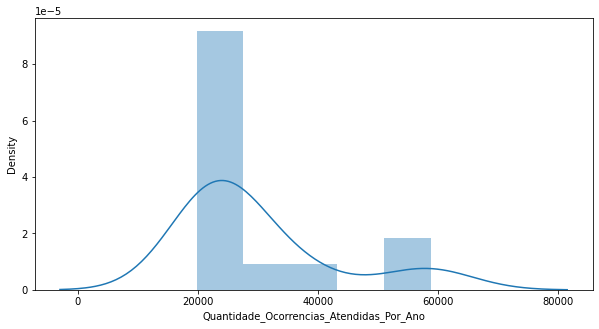
**Visualizações dos dados**

* **Ocorrências Atendidas Por Ano**

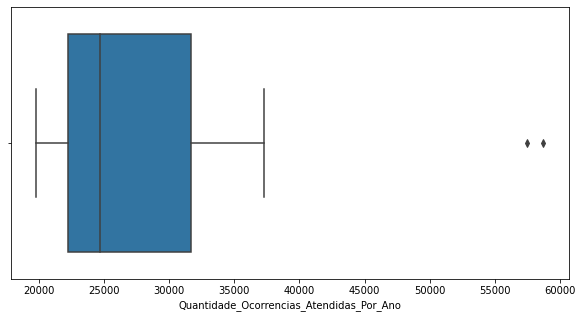
| ## Visualizações dos dados  ####\*\*Ocorrencias Atendidas Por Ano\*\*  criaGraficoLinha(ocorrenciasAtendidasPorAno, 'ATENDIMENTO\_ANO', 'Quantidade\_Ocorrencias\_Atendidas\_Por\_Ano', 'Quantidade\_Ocorrencias\_Atendidas\_Por\_Ano', 'ATENDIMENTO\_ANO', 'Quantidade Ocorrencias') |
| --- |



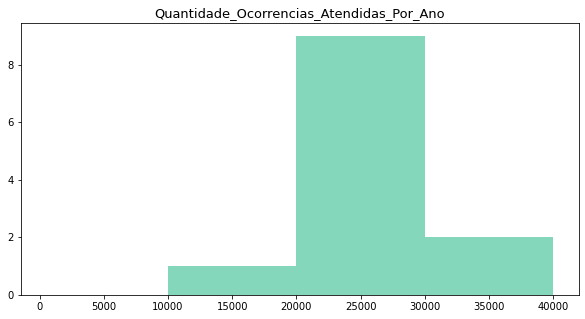
| criaGraficoDistribuicao(ocorrenciasAtendidasPorAno, 'Quantidade\_Ocorrencias\_Atendidas\_Por\_Ano') |
| --- |



| criaGraficoBoxplot(ocorrenciasAtendidasPorAno, 'Quantidade\_Ocorrencias\_Atendidas\_Por\_Ano') |
| --- |



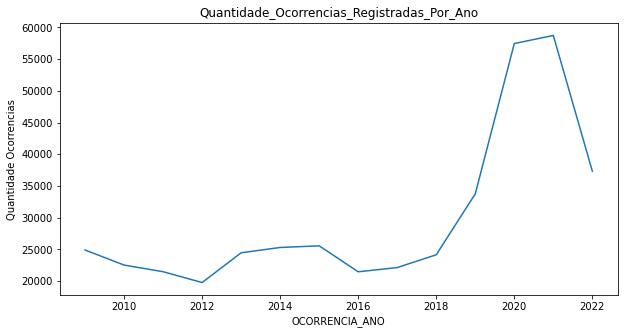
| bins = [500, 10000, 20000, 30000, 40000] criaHistogramaComBins('Quantidade\_Ocorrencias\_Atendidas\_Por\_Ano', ocorrenciasAtendidasPorAno, bins, 'mediumaquamarine', 'Quantidade\_Ocorrencias\_Atendidas\_Por\_Ano') |
| --- |



| bins = [50000, 60000, 70000, 80000, 100000] criaHistogramaComBins('Quantidade\_Ocorrencias\_Atendidas\_Por\_Ano', ocorrenciasAtendidasPorAno, bins, 'mediumaquamarine', 'Quantidade\_Ocorrencias\_Atendidas\_Por\_Ano') |
| --- |

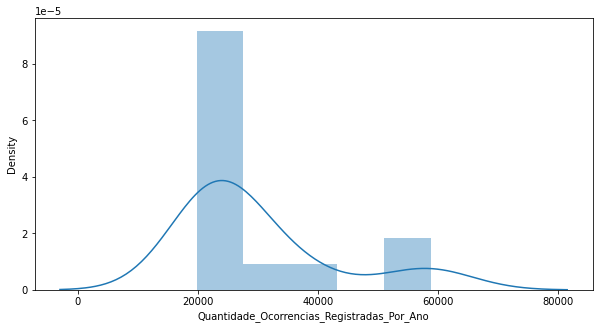


| criaGraficoLinha(ocorrenciasRegistradasPorAno, 'OCORRENCIA\_ANO', 'Quantidade\_Ocorrencias\_Registradas\_Por\_Ano', 'Quantidade\_Ocorrencias\_Registradas\_Por\_Ano', 'OCORRENCIA\_ANO', 'Quantidade Ocorrencias') |
| --- |

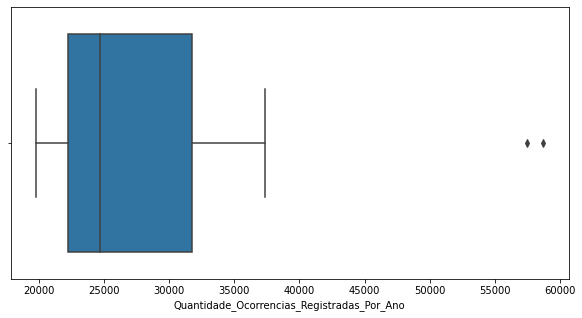


* **Ocorrências Registradas Por Ano**

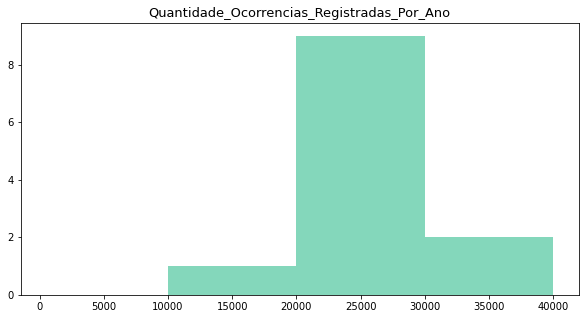
| criaGraficoDistribuicao(ocorrenciasRegistradasPorAno, 'Quantidade\_Ocorrencias\_Registradas\_Por\_Ano') |
| --- |



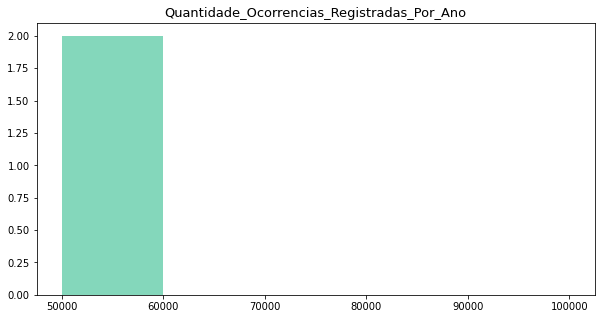
| criaGraficoBoxplot(ocorrenciasRegistradasPorAno, 'Quantidade\_Ocorrencias\_Registradas\_Por\_Ano') |
| --- |



| bins = [500, 10000, 20000, 30000, 40000]  criaHistogramaComBins('Quantidade\_Ocorrencias\_Registradas\_Por\_Ano', ocorrenciasRegistradasPorAno, bins, 'mediumaquamarine', 'Quantidade\_Ocorrencias\_Registradas\_Por\_Ano') |
| --- |

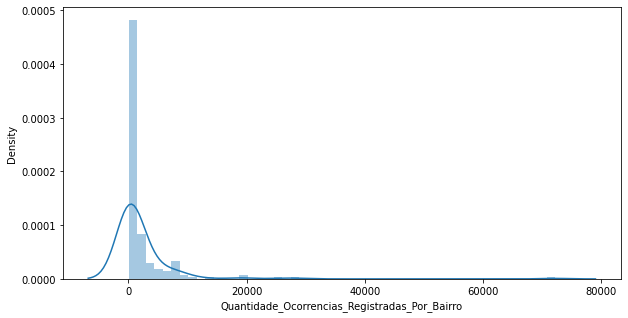


| bins = [50000, 60000, 70000, 80000, 100000] criaHistogramaComBins('Quantidade\_Ocorrencias\_Registradas\_Por\_Ano', ocorrenciasRegistradasPorAno, bins, 'mediumaquamarine', 'Quantidade\_Ocorrencias\_Registradas\_Por\_Ano') |
| --- |



* **Ocorrências Por Bairro**

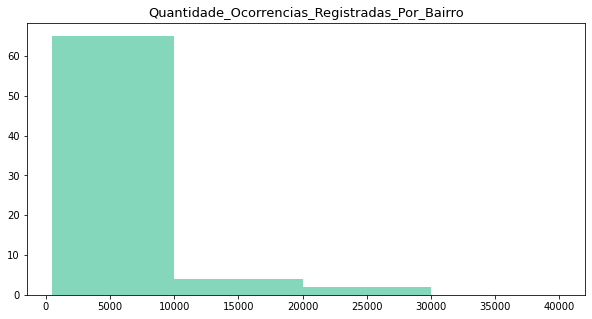
| ####\*\*Ocorrencias Por Bairro\*\*  criaGraficoDistribuicao(ocorrenciasPorBairro, 'Quantidade\_Ocorrencias\_Registradas\_Por\_Bairro') |
| --- |



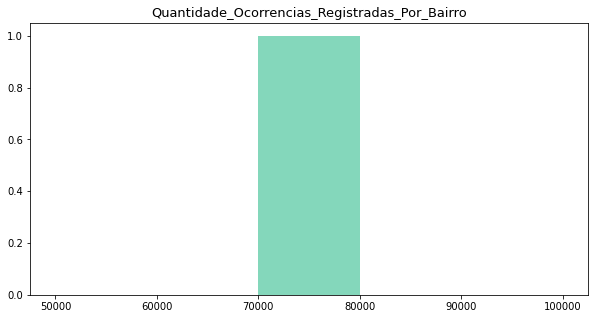
| criaGraficoBoxplot(ocorrenciasPorBairro, 'Quantidade\_Ocorrencias\_Registradas\_Por\_Bairro') |
| --- |



| bins = [500, 10000, 20000, 30000, 40000] criaHistogramaComBins('Quantidade\_Ocorrencias\_Registradas\_Por\_Bairro', ocorrenciasPorBairro, bins, 'mediumaquamarine', 'Quantidade\_Ocorrencias\_Registradas\_Por\_Bairro') |
| --- |

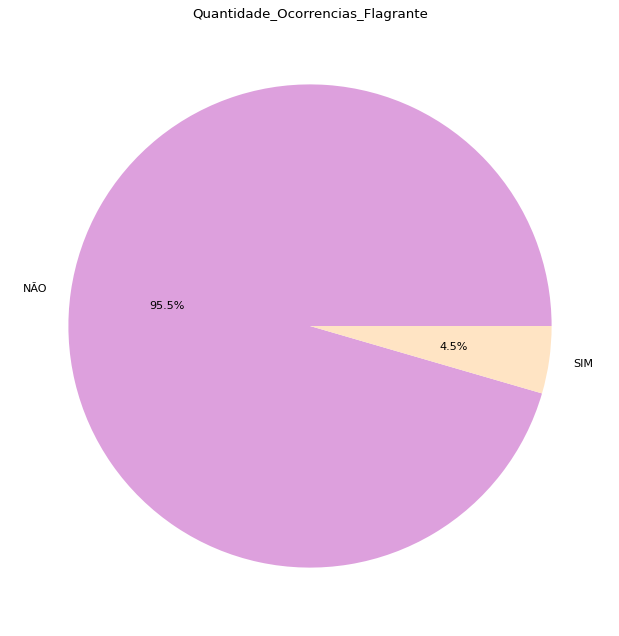


| bins = [50000, 60000, 70000, 80000, 100000]  criaHistogramaComBins('Quantidade\_Ocorrencias\_Registradas\_Por\_Bairro', ocorrenciasPorBairro, bins, 'mediumaquamarine', 'Quantidade\_Ocorrencias\_Registradas\_Por\_Bairro') |
| --- |



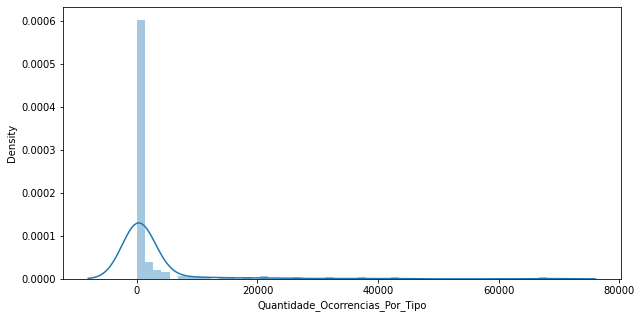
* **Ocorrências Flagrante**

| ####\*\*Ocorrencias Flagrante\*\*  quantidadeOcorrenciasFlagranteGraficoPizza = criaGraficoPizza(ocorrenciasFlagrante, 'Quantidade\_Ocorrencias\_Flagrante', 'FLAG\_FLAGRANTE', 'Quantidade\_Ocorrencias\_Flagrante') |
| --- |



* **Ocorrências Atendidas Por Tipo**

| ####\*\*Ocorrencias Por Tipo\*\*  criaGraficoDistribuicao(ocorrenciasPorTipo, 'Quantidade\_Ocorrencias\_Por\_Tipo') |
| --- |

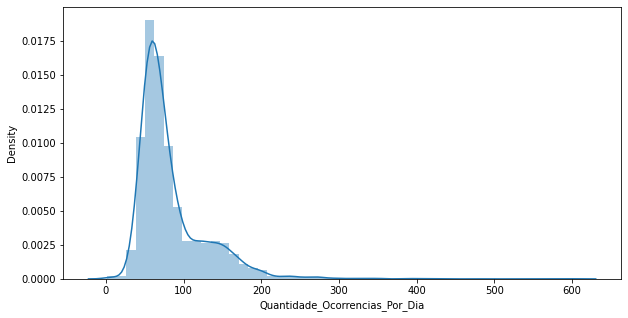


| criaGraficoBoxplot(ocorrenciasPorTipo, 'Quantidade\_Ocorrencias\_Por\_Tipo') |
| --- |

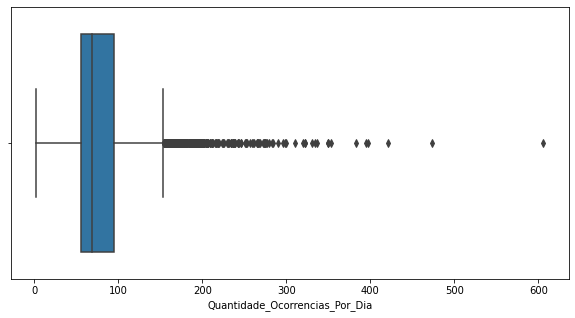


* **Ocorrências Atendidas Por Dia**

| ####\*\*Ocorrencias Por Dia\*\*  criaGraficoDistribuicao(ocorrenciasPorDia, 'Quantidade\_Ocorrencias\_Por\_Dia') |
| --- |

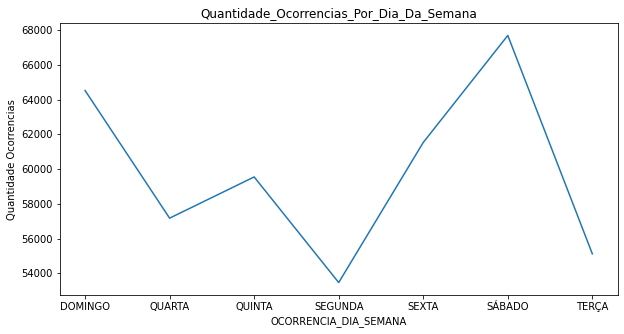


| criaGraficoBoxplot(ocorrenciasPorDia, 'Quantidade\_Ocorrencias\_Por\_Dia') |
| --- |

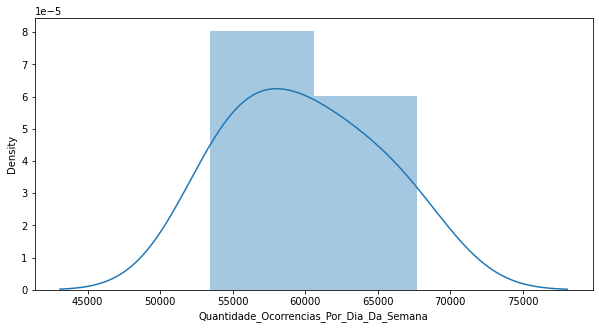


* **Ocorrências Atendidas Por Dia Da Semana**

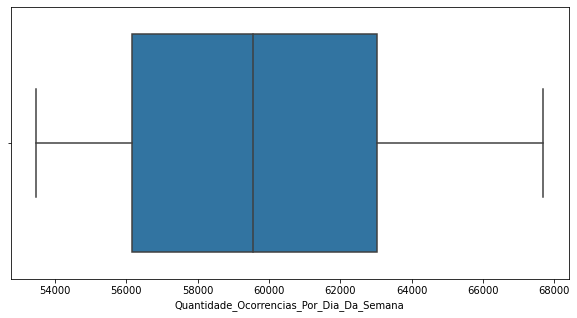
| ####\*\*Ocorrencias Por Dia Da Semana\*\*  criaGraficoLinha(ocorrenciasPorDiaDaSemana, 'OCORRENCIA\_DIA\_SEMANA', 'Quantidade\_Ocorrencias\_Por\_Dia\_Da\_Semana', 'Quantidade\_Ocorrencias\_Por\_Dia\_Da\_Semana', 'OCORRENCIA\_DIA\_SEMANA', 'Quantidade Ocorrencias') |
| --- |



| criaGraficoDistribuicao(ocorrenciasPorDiaDaSemana, 'Quantidade\_Ocorrencias\_Por\_Dia\_Da\_Semana') |
| --- |

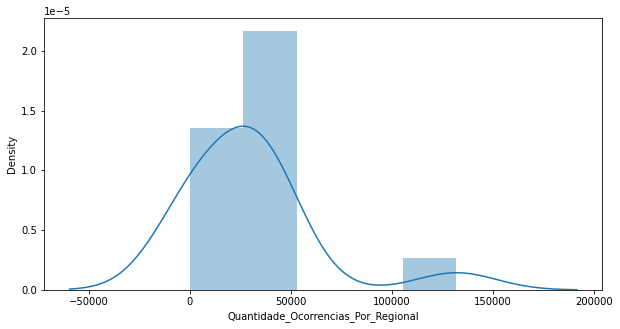


| criaGraficoBoxplot(ocorrenciasPorDiaDaSemana, 'Quantidade\_Ocorrencias\_Por\_Dia\_Da\_Semana') |
| --- |

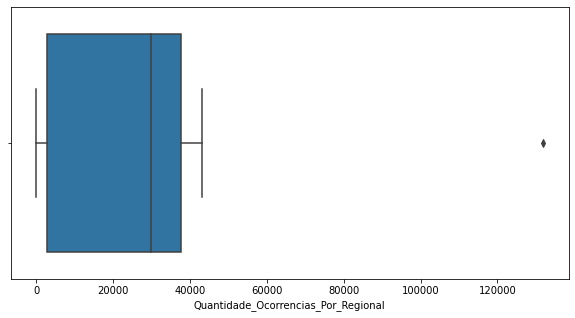


* **Ocorrências Atendidas Por Regional**

| ####\*\*Ocorrências Por Regional\*\*  criaGraficoDistribuicao(ocorrenciasPorRegional, 'Quantidade\_Ocorrencias\_Por\_Regional') |
| --- |



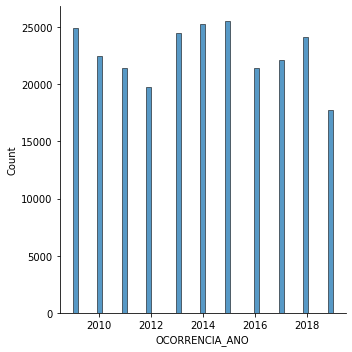
| criaGraficoBoxplot(ocorrenciasPorRegional, 'Quantidade\_Ocorrencias\_Por\_Regional') |
| --- |



Com base nas análises observadas anteriormente, foram criadas as perguntas a seguir para serem respondidas.

* **Qual ano possui mais atendimento?**

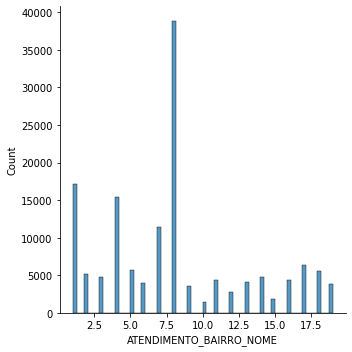
| plt.figure(figsize=(20,20)) sns.displot(data=df, x='OCORRENCIA\_ANO') |
| --- |



| Resposta: 2021, 2020 e 2022 |
| --- |

* **Qual bairro possui mais atendimento?**

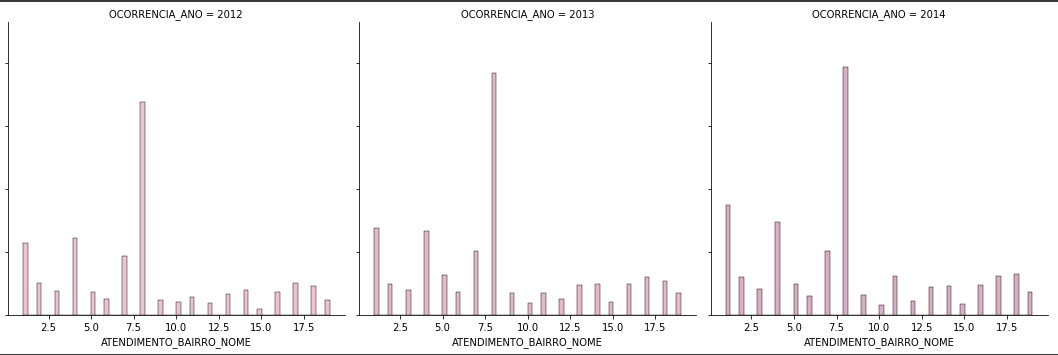
| plt.figure(figsize=(10,10)) sns.displot(data=df[df['ATENDIMENTO\_BAIRRO\_NOME']<20], x='ATENDIMENTO\_BAIRRO\_NOME') |
| --- |

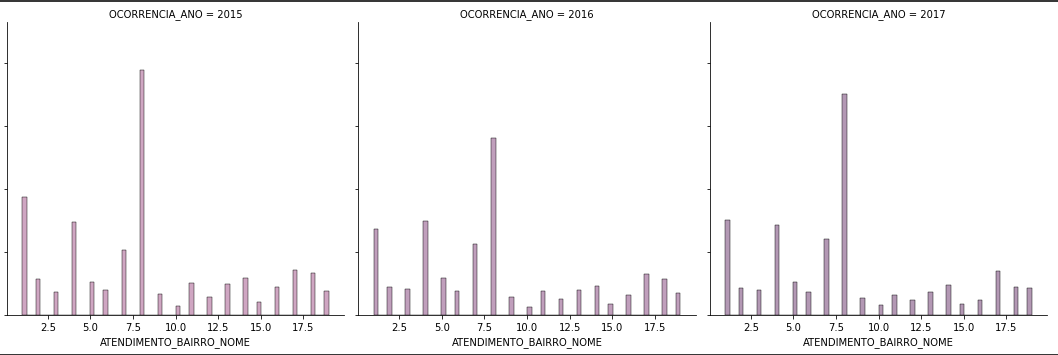


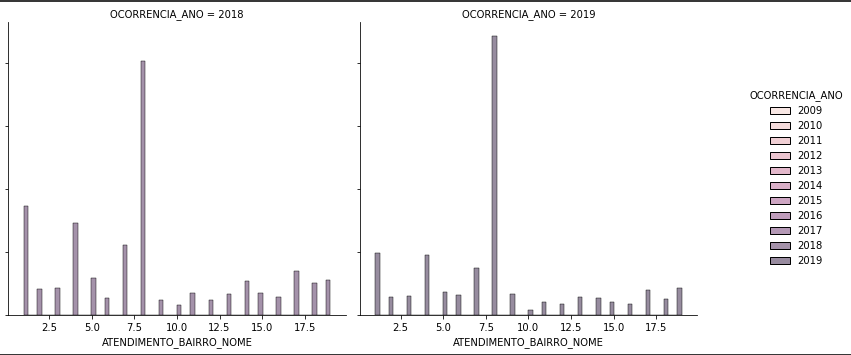
| Resposta: 8: Centro, 1: CIC e 4: SÍTIO CERCADO |
| --- |

* **Qual ano por bairro possui mais atendimento?**

| plt.figure(figsize=(100,80)) sns.displot(data=df[df['ATENDIMENTO\_BAIRRO\_NOME']<20], x='ATENDIMENTO\_BAIRRO\_NOME', hue='OCORRENCIA\_ANO', col='OCORRENCIA\_ANO', color='red') |
| --- |



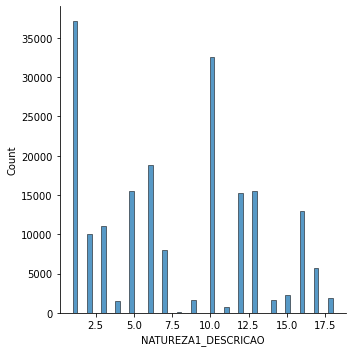




| Resposta:  2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018 e 2019; 1ºlugar: 8, Centro.  2ºlugar: 1, CIC. 3ºlugar: 4: SÍTIO CERCADO.  2020: 1ºlugar: 8, Centro.  2ºlugar: 4: SÍTIO CERCADO. 3ºlugar: 1, CIC.  2021: 2022 1ºlugar: 8, Centro.  2ºlugar: 1, CIC. 3ºlugar: 4: SÍTIO CERCADO. |
| --- |

* **Qual o tipo de natureza da ocorrência?**

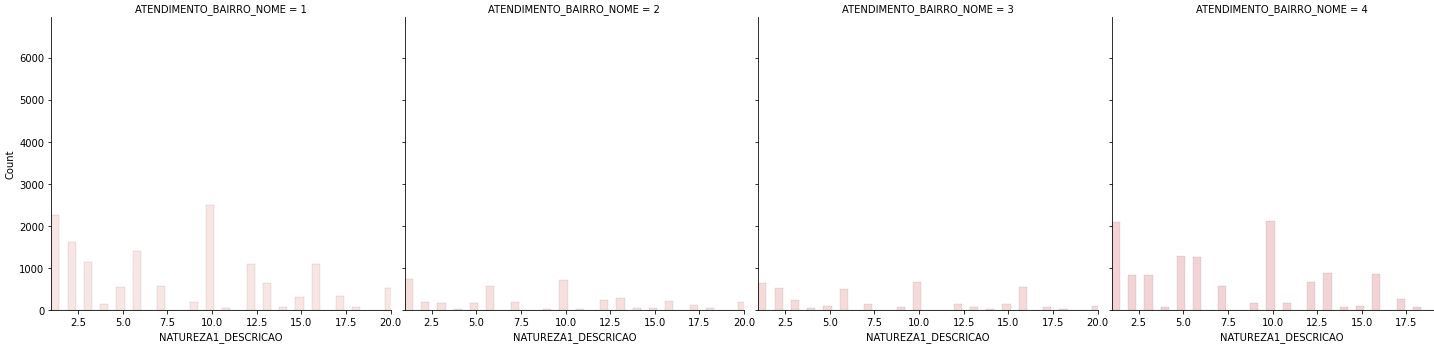
| plt.figure(figsize=(20,20)) sns.displot(data=df[df['NATUREZA1\_DESCRICAO']<20], x='NATUREZA1\_DESCRICAO') |
| --- |

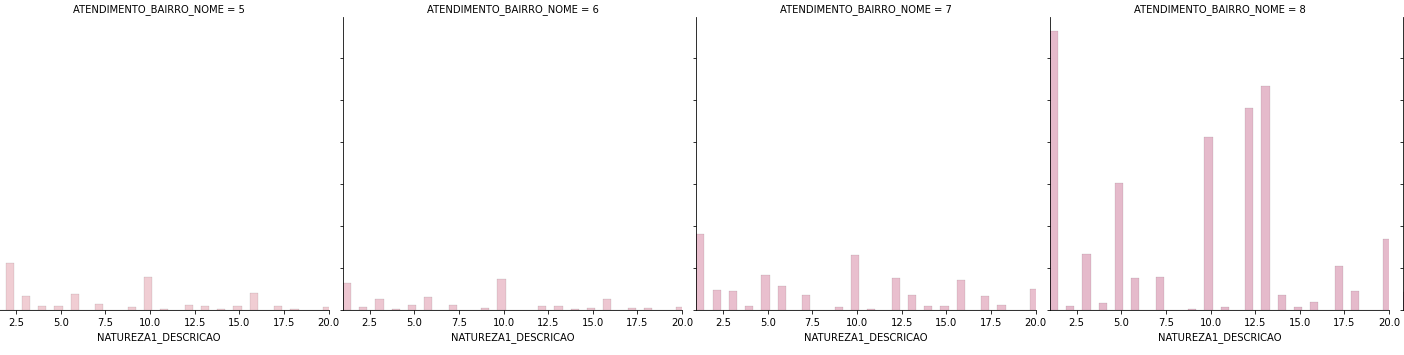


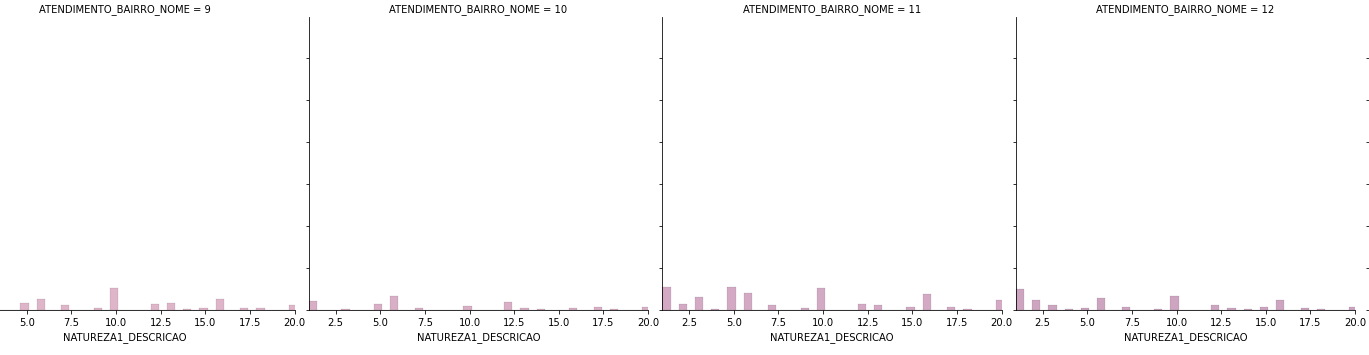
| Resposta: 1: Apoio, 12: Fundada Suspeita (Abordagem) e 10: Dano |
| --- |

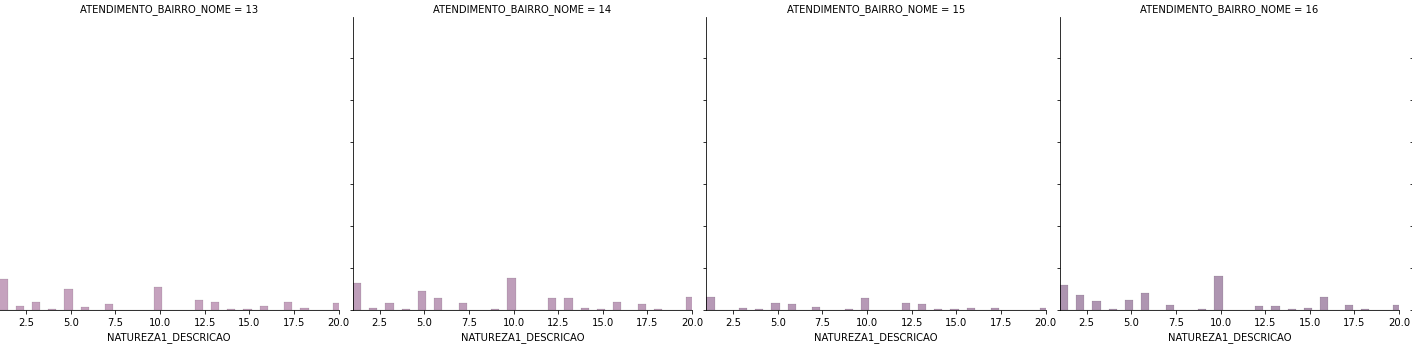
* **Qual o tipo de natureza da ocorrência em cada bairro?**

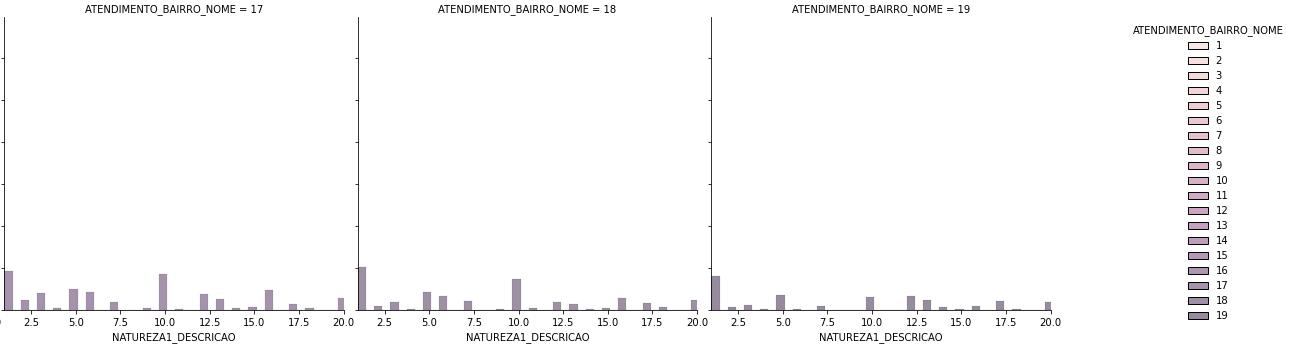
| plt.figure(figsize=(20,20)) sns.displot(data=df[df['ATENDIMENTO\_BAIRRO\_NOME']<20], x='NATUREZA1\_DESCRICAO', hue='ATENDIMENTO\_BAIRRO\_NOME', col='ATENDIMENTO\_BAIRRO\_NOME', color=['black']) plt.xlim(1,20) |
| --- |







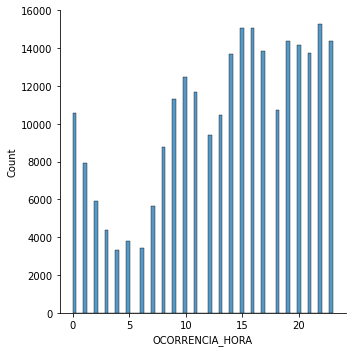




| Resposta: foram utilizados só os 20 tipos de ATENDIMENTO\_BAIRRO\_NOME e ATENDIMENTO BAIRRO NOME, que se tem mais registro. |
| --- |

* **Qual horário da ocorrência mais frequente?**

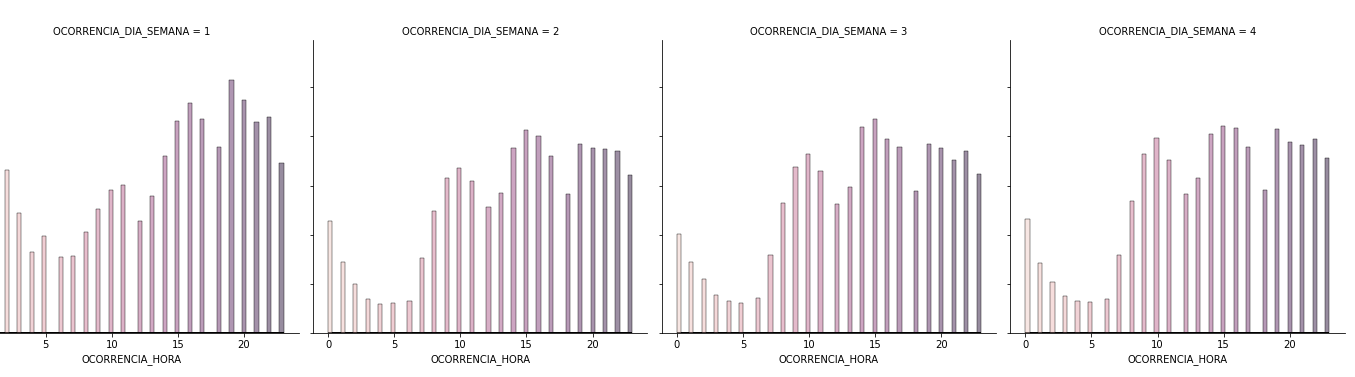
| plt.figure(figsize=(20,20)) sns.displot(data=df, x='OCORRENCIA\_HORA') |
| --- |

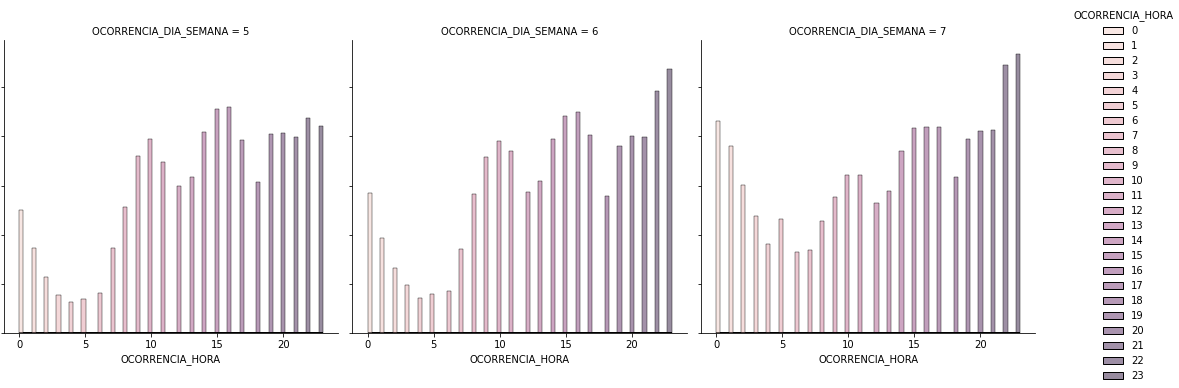


| Resposta: 15, 14 e 16 |
| --- |

* **Qual horário da semana e bairro mais frequentes?**

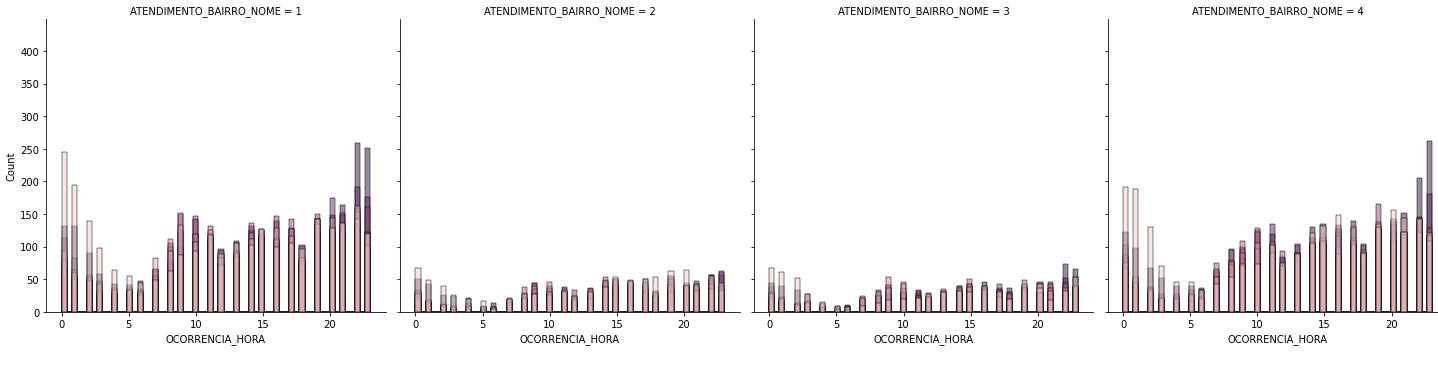
| plt.figure(figsize=(20,20)) sns.displot(data=df, x='OCORRENCIA\_HORA', hue='OCORRENCIA\_HORA', col='OCORRENCIA\_DIA\_SEMANA') |
| --- |

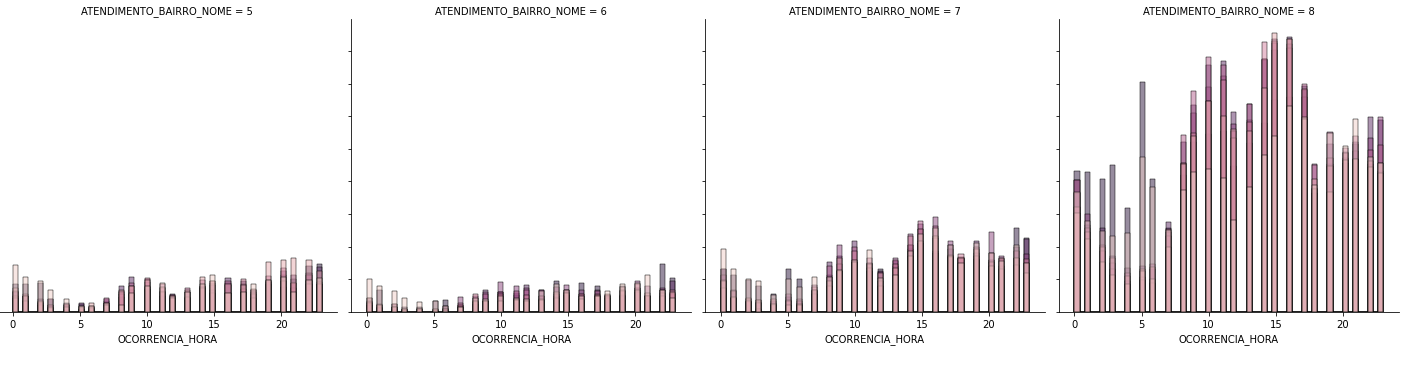


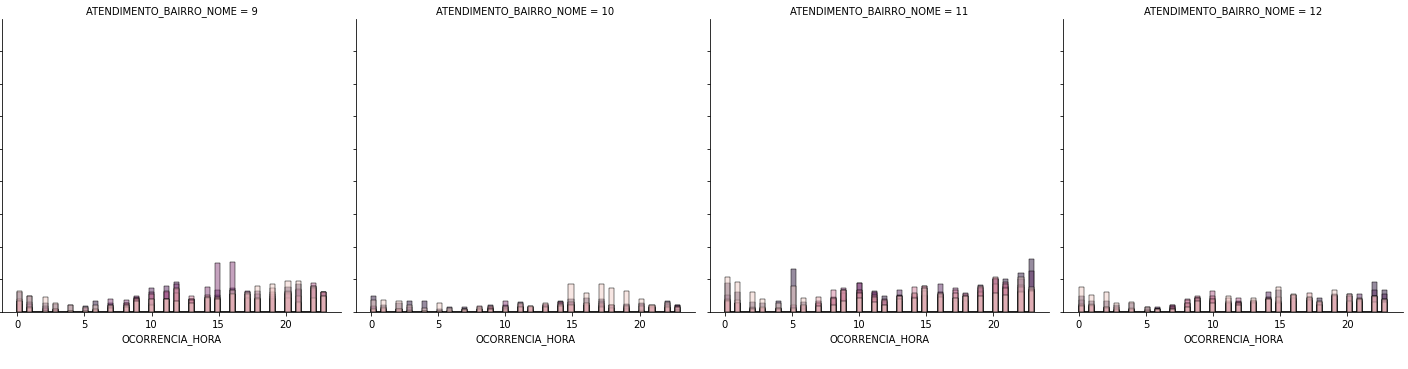


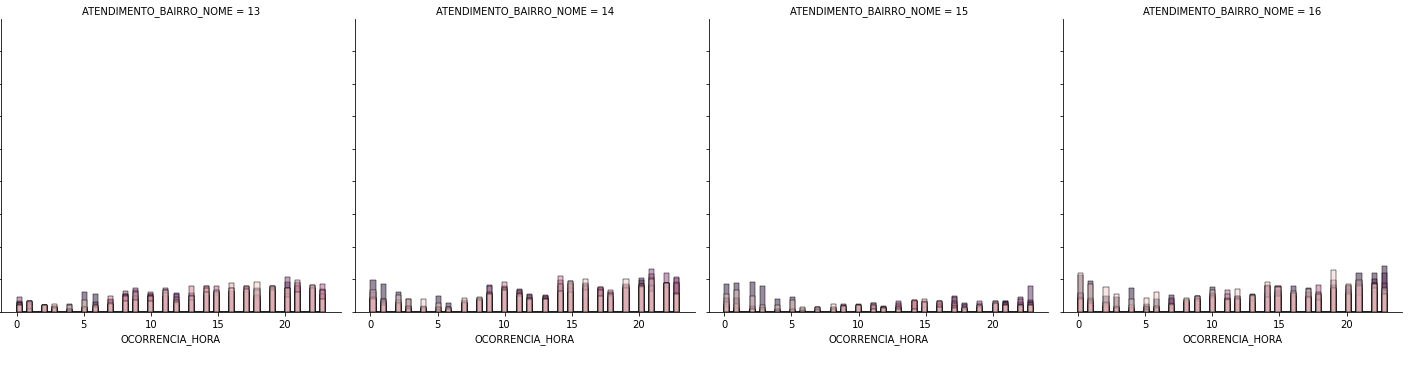
| Resposta: final de semana a mais ocorrência a tarde, 13h até umas 20h |
| --- |

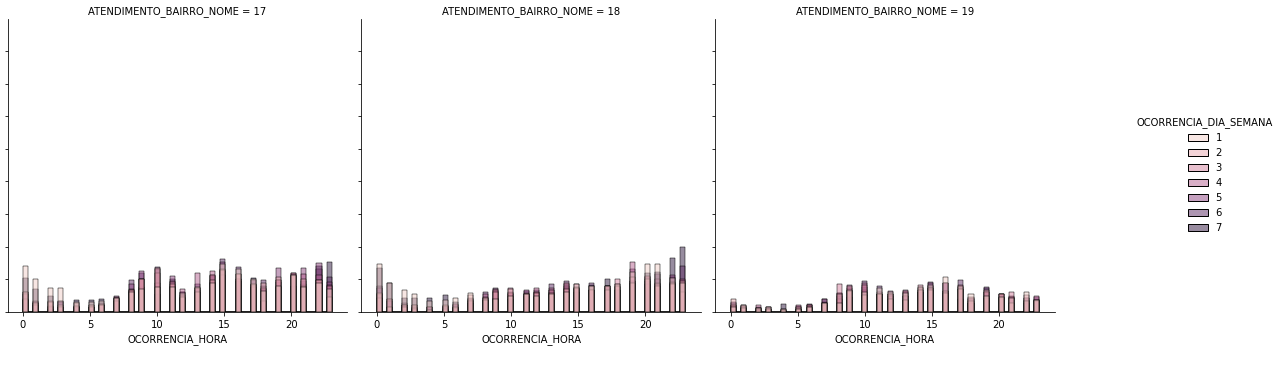
| plt.figure(figsize=(20,20)) sns.displot(data=df[df['ATENDIMENTO\_BAIRRO\_NOME']<20], x='OCORRENCIA\_HORA', hue='OCORRENCIA\_DIA\_SEMANA', col='ATENDIMENTO\_BAIRRO\_NOME', color=['black']) |
| --- |











| Resposta: Bairro 8 que é o Centro tem mais registro de manhã e à tarde, 1 CIC e 4 SÍTIO CERCADO, se mantém igual com descanso só na madrugada |
| --- |

* **Descrição dos experimentos com os frameworks de big data:**
  + *descrição das ferramentas utilizadas;*

A principal ferramenta utilizada para análise foi o PySpark, que é a API do Spark para Python. O Apache Spark é a estrutura de código aberto que proporciona o processamento em paralelo, ao mesmo tempo que demonstra a possibilidade do suporte ao processamento na memória e auxilia no desempenho dos aplicativos que utilizam e analisam contextos de Big Data.

* + *apresentação da motivação do uso do framework;*

A principal motivação para a utilização do framework de Big Data Apache Spark foi a relação com a velocidade de processamento e também da possibilidade processamento distribuído dos dados.

Durante o trabalho com o dataset um dos principais desafios foi o processamento dos dados. O dataset utilizado possui um total de , o que torna o trabalho de processamento bastante custoso.

Sendo assim, optou-se pelo uso do Apache Spark, pois é uma estrutura rápida que possibilita o acesso a uma API que pode ser utilizada diretamente na linguagem Python, que é o PySpark.

* + *apresentação das técnicas utilizadas; descrição do protocolo experimental; comparação entre as diferentes técnicas de big data (mínimo três técnicas diferentes); análise dos resultados obtidos.*

A primeira etapa para iniciar as análises e transformações é configurar o ambiente do Spark, para isso é necessário seguir todo o processo de instalação, importações e configurações a seguir.

Após todas as configurações necessárias, é preciso também criar uma sessão para inicializar todo o processo. Depois disso, basta criar o dataframe em Spark, passando o dataframe já utilizado anteriormente, utilizando spark.createDataFrame(). E então realiza-se o uso do método printSchema() para mostrar os tipos de dados e o show() para visualizar o dataframe carregado.

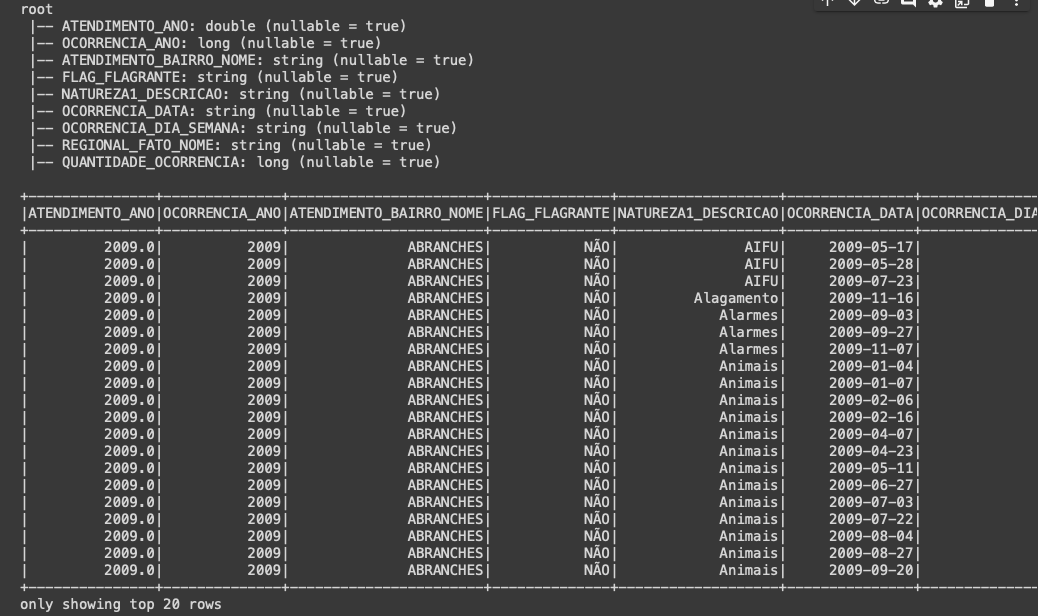
Sendo assim, é realizada também a utilização das funções já utilizadas anteriormente fora do Spark, como a describe() e head() para obter mais informações do dataset.

A primeira análise que é feita agora utilizando funções do Spark é a Contagem de colunas duplicadas e depois buscar colunas numéricas e categóricas e os valores nulos.

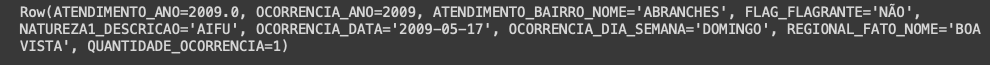
**Análises com Spark**

* **Configurando o Spark e Experimentos Iniciais com o Dataframe**

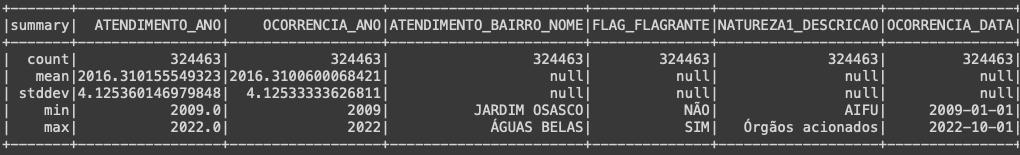
| ## Análises com Spark  Configurando o Spark  # install java !apt-get install openjdk-8-jdk-headless -qq > /dev/null  # install spark (change the version number if needed) !wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz  # unzip the spark file to the current folder !tar xf spark-3.0.0-bin-hadoop3.2.tgz  # set your spark folder to your system path environment.  **import** os os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64" os.environ["SPARK\_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"   # install findspark using pip !pip install -q findspark  **import** findspark findspark.init()  **import** pyspark **from** pyspark.sql **import** SparkSession  spark = SparkSession.builder\  .master("local")\  .appName("Colab")\  .config('spark.ui.port', '4050')\  .getOrCreate()  sparkDF=spark.createDataFrame(df2)  sparkDF.printSchema() sparkDF.show() |
| --- |



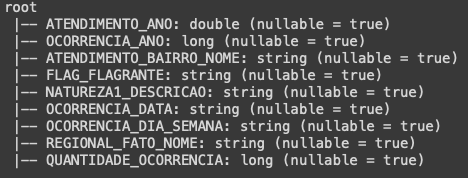
| sparkDF.head() |
| --- |



| sparkDF.describe().show() |
| --- |

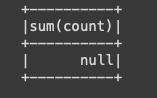


| sparkDF.printSchema() |
| --- |



* **Contagem de linhas duplicadas**

| Contagem de linhas duplicadas  **import** pyspark.sql.functions **as** funcs sparkDF.groupby(sparkDF.columns).count().where(funcs.col('count') > 1).select(funcs.sum('count')).show() |
| --- |



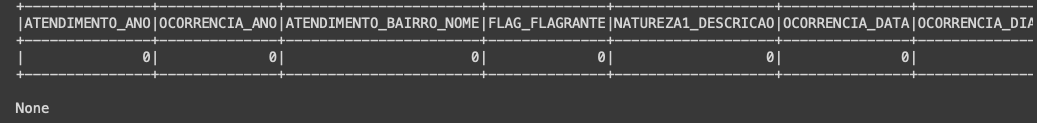
* **Buscando colunas numéricas e categóricas**

| Achar colunas numéricas e categóricas  colunas\_numericas = list() colunas\_categoricas = list() **for** col\_ **in** sparkDF.columns:  **if** sparkDF.select(col\_).dtypes[0][1] != "string":  colunas\_numericas.append(col\_)  **else**:  colunas\_categoricas.append(col\_)   print("Colunas Numericas",colunas\_numericas) print("Colunas Categoricas",colunas\_categoricas) |
| --- |



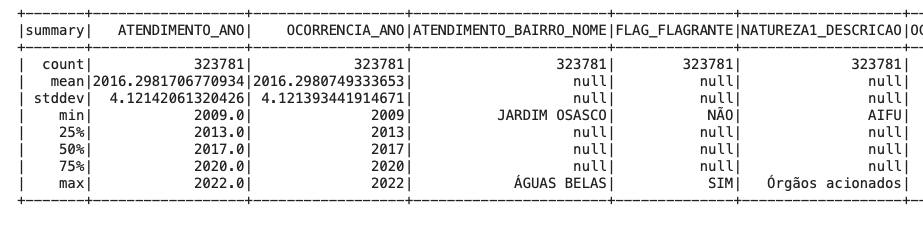
* **Contagem de valores nulos**

| Contagem de valores nulos  **from** pyspark.sql.functions **import** \* print(sparkDF.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) **for** c **in** sparkDF.columns]).show()) |
| --- |



* **Métricas**

| **sparkDF.summary().show()** |
| --- |

****

**6. Modelagem e Treinamento**

xxxxx

**7. Integração**

xxxxxx

**8. Referências**

AHMED, Asif. **PySpark in Google Colab.** Medium: 2019. Disponível em: <<https://towardsdatascience.com/pyspark-in-google-colab-6821c2faf41c#:~:text=To%20run%20spark%20in%20Colab,Jupyter%20Notebook%20of%20the%20Colab.&text=Our%20Colab%20is%20ready%20to%20run%20PySpark>>. Acesso em: 20/10/2022.

BARAD, Vishal. **Data Preprocessing Using Pyspark (Part:1).** Medium: 2021. Disponível em: <<https://medium.com/vedity/python-data-preprocessing-using-pyspark-cc3f709c3c23>>. Acesso em: 25/10/2022.

CETAX. **Tutorial Pyspark e MLlib.** Brasil: 2022. Disponível em: <<https://cetax.com.br/tutorial-pyspark-e-mllib/>>. Acesso em: 28/10/2022.

Data Science. **Tutorial PySpark.** Sem data. Disponível em:

<<https://datascience.eu/pt/programacao/tutorial-pyspark/>>. Acesso em: 20/10/2022.

LOPES, Alexandre. **Funcionamento do PySpark.** Medium: 2020. Disponível em <<https://medium.com/data-hackers/entendo-funcionamento-do-pyspark-2b5ab4321ab9>>. Acesso em: 25/10/2022.

Microsoft. **O que é o Apache Spark?** Learn: 2022. Disponível em <<https://learn.microsoft.com/pt-br/dotnet/spark/what-is-spark>>. Acesso em: 15/10/2022.

Prefeitura Municipal de Curitiba. **Dados Abertos.** Sem data. Disponível em: <<https://www.curitiba.pr.gov.br/dadosabertos/busca/?termo=sigesguarda>>. Acesso em: 05/10/2022.

RAONIAR, RAHUL. **Introduction to PySpark.** Medium: 2021. Disponível em: <<https://medium.com/the-researchers-guide/introduction-to-pyspark-a61f7217398e>>. Acesso em: 15/10/2022.

ROSEN, Josh. **PySpark Internals.** Wiki: 2016. Disponível em: <<https://cwiki.apache.org/confluence/display/SPARK/PySpark+Internals#:~:text=PySpark%20is%20built%20on%20top,JVM%20and%20create%20a%20JavaSparkContext.&text=RDD%20transformations%20in%20Python%20are,on%20PythonRDD%20objects%20in%20Java>>. Acesso em: 15/10/2022.

SHAFIQUE, Ayesha. **Exploratory Data Analysis using Pyspark Dataframe in Python.** Medium: 2019. Disponível em: <<https://medium.com/@aieeshashafique/exploratory-data-analysis-using-pyspark-dataframe-in-python-bd55c02a2852>>. Acesso em: 18/10/2022.

Spark By Examples. **How to Convert Pandas to PySpark DataFrame.** 2022. Disponível em: <<https://sparkbyexamples.com/pyspark/convert-pandas-to-pyspark-dataframe/>>. Acesso em: 10/10/2022.

Spark By Examples. **PySpark Union and UnionAll Explained.** 2022.Disponível em: <<https://sparkbyexamples.com/pyspark/pyspark-union-and-unionall/>>. Acesso em: 10/10/2022.

SÔTO, Rubens. **Encapsulando Transformações em DataFrames com Pyspark.** Medium: 2020. Disponível em: <[https://medium.com/data-hackers/encapsulando-transformações-em-dataframes-com-pyspark-9543e0cc69f1](https://medium.com/data-hackers/encapsulando-transforma%C3%A7%C3%B5es-em-dataframes-com-pyspark-9543e0cc69f1)>. Acesso em: 23/10/2022.

