

# **Crimes de Morte**



**Comparativo entre as cidades  
de São Paulo e Nova York**

# A Empresa

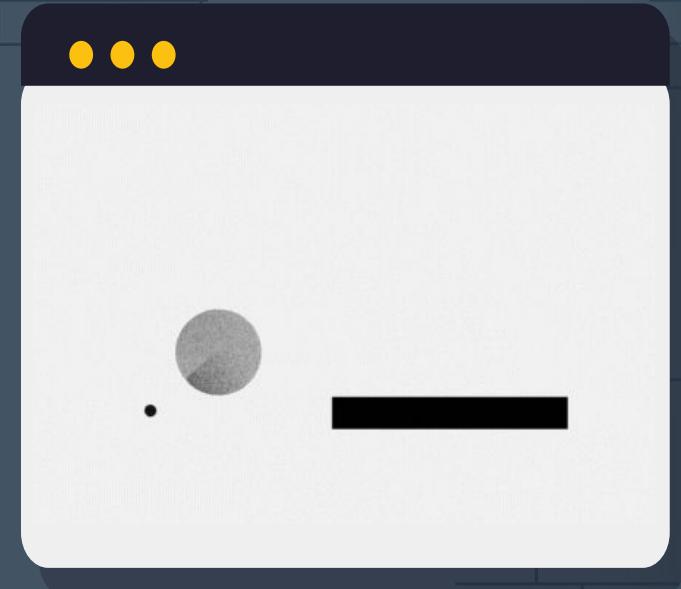
01





## A Empresa

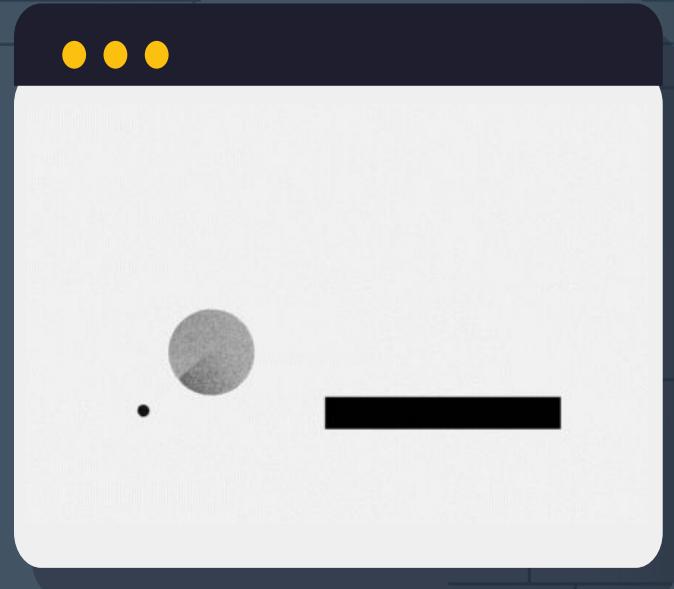
A Mindful Data é uma empresa de consultoria na área de engenharia de dados, fundada em 2023 por quatro alunos recém formados no bootcamp da SoulCode Academy.



## Nosso Propósito

A Mindful Data tem a proposta de atender à demanda de empresas nos processos de:

- Data Migration
- Processos de ETL
- Produção de insights e dashboards para apoiar decisões de negócios





# A Equipe

Suzana Gomes



Wilbert Silva



Giovana de Brito



Anderson Gabriel



Engenheira de Dados



Engenheiro de Dados



Engenheira de Dados



Engenheiro de Dados



# O Projeto



## Escopo

- Uso mínimo de dois Datasets em formatos diferentes
- Armazenamento na Google Cloud Platform (GCP)
- Procedimentos de ETL e análise realizadas através de Pandas, PySpark e Spark SQL
- Análises realizadas através do BigQuery na GCP
- Armazenamento em MySQL para Datasets brutos e MongoDB Atlas para os tratados
- Criação de um Dashboard no Looker Studio e um Workflow simples

## Áreas Escolhidas



### ● NOVA YORK

- Área total: 141.296,75 km<sup>2</sup>
- População: 20.201.249 habitantes
- Densidade: 143 habitantes/km<sup>2</sup>
- IDH: 0,944 (muito alto)
- PIB: US\$ 1.701 trilhão
- PIB per capita: US\$ 64.579

### ● SÃO PAULO

- Área total: 248.219,481 km<sup>2</sup>
- População: 46.649.132 habitantes
- Densidade: 187,94 habitantes/km<sup>2</sup>
- IDH: 0,826 (muito alto)
- PIB: R\$ 2.377 trilhões
- PIB per capita: R\$ 51.364,73



# Fonte dos Dados

Portal do Governo | Cidadão SP

Segurança Pública



# Fonte dos Dados

The image shows a computer monitor with a dark grey bezel. On the screen, the NYC OpenData logo is at the top left, followed by a navigation bar with links for Home, Data, About, Learn, Alerts, and Contact Us. The main content area has a light grey background. A large title "NYPD Complaint Data Historic" is centered at the top. Below it, a category "Public Safety" is shown in a grey box. The main text area contains a description of the dataset: "This dataset includes all valid felony, misdemeanor, and violation crimes reported to the New York City Police Department (NYPD) from 2006 to the end of last year (2019). For additional details, please see the attached data dictionary in the 'About' section." To the right of this text, there are two buttons: "View Data" (in a dark blue box) and "Visualize" (with a dropdown arrow). Further down, under the heading "About this Dataset", there is an "Updated" timestamp "June 9, 2022" and a "Dataset Information" link. On the far right, there are two more sections: "Updated June 9, 2022" and "Data Provided by Police Department (". The monitor sits on a white stand against a dark blue background with faint white grid lines.

**NYC OpenData**

Home Data About ▾ Learn ▾ Alerts Contact Us

NYPD Complaint Data Historic

Public Safety

This dataset includes all valid felony, misdemeanor, and violation crimes reported to the New York City Police Department (NYPD) from 2006 to the end of last year (2019). For additional details, please see the attached data dictionary in the 'About' section.

View Data Visualize ▾

Updated  
June 9, 2022

Data Provided by  
Police Department (

About this Dataset

Updated  
**June 9, 2022**

Dataset Information



## Questionamentos preliminares



01

**Analisar dados sobre crimes de mortes em grandes cidades**

02

**Traçar o perfil das vítimas desses crimes**

03

**Mapear quais são os locais mais violentos**

04

**Evolução da quantidade de crimes de acordo com o período**

# O Processo



Mindful  
Data

DATA SCIENCE

# WORKFLOW

**EXTRAÇÃO**

**TRATAMENTO**

**CARREGAMENTO**

**DATAVIZ**



**Extract**

**Transform**

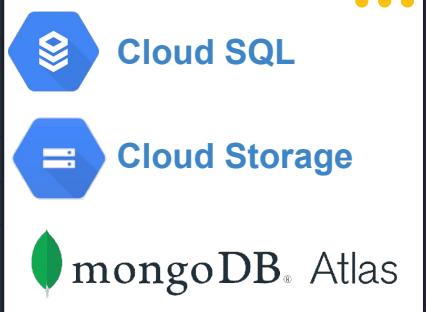


**Load**



# Ferramentas

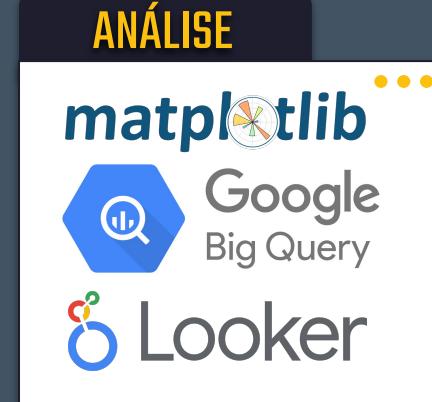
## ARMAZENAMENTO



## TRATAMENTO



## ANÁLISE



## AUXILIARES





# Extração

# Extração

## datasets-originais-agsw

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Subject to object ACLs	None
<input type="checkbox"/> Name	Size	Type	
<input type="checkbox"/> <a href="#">Homicidio_2017_2022.xlsx</a>	3.5 MB	app	
<input type="checkbox"/> <a href="#">LCSM_2016_2022.xlsx</a>	131.2 KB	app	
<input type="checkbox"/> <a href="#">Latrocínio_2018_2022.xlsx</a>	279.5 KB	app	
<input type="checkbox"/> <a href="#">MDIP_2013_2022.xlsx</a>	165.3 MB	app	

# Extração Pandas

- Criação do DF
  - pd.read\_excel
  - pd.concat

```
# Trazendo cada aba da planilha para o dataset
dfle1 = pd.read_excel('gs://datasets-originais-agsw/ssp/LCSM_2016_2022.xlsx', sheet_name='2017')
dfle2 = pd.read_excel('gs://datasets-originais-agsw/ssp/LCSM_2016_2022.xlsx', sheet_name='2018')
dfle3 = pd.read_excel('gs://datasets-originais-agsw/ssp/LCSM_2016_2022.xlsx', sheet_name='2019')
dfle4 = pd.read_excel('gs://datasets-originais-agsw/ssp/LCSM_2016_2022.xlsx', sheet_name='2020')
dfle5 = pd.read_excel('gs://datasets-originais-agsw/ssp/LCSM_2016_2022.xlsx', sheet_name='2021')
```

```
# Concatenando os Data Frames
dfsp_leao = pd.concat([dfle1, dfle2, dfle3, dfle4, dfle5])
```

## Load no GCS

### Carregamento para bucket

#### Função upload.blob

```
# Função para fazer upload de arquivo no bucket
def upload_blob(bucket, arquivo, destino):
    client = storage.Client()
    bucket = client.bucket(bucket)
    blob = bucket.blob(destino)

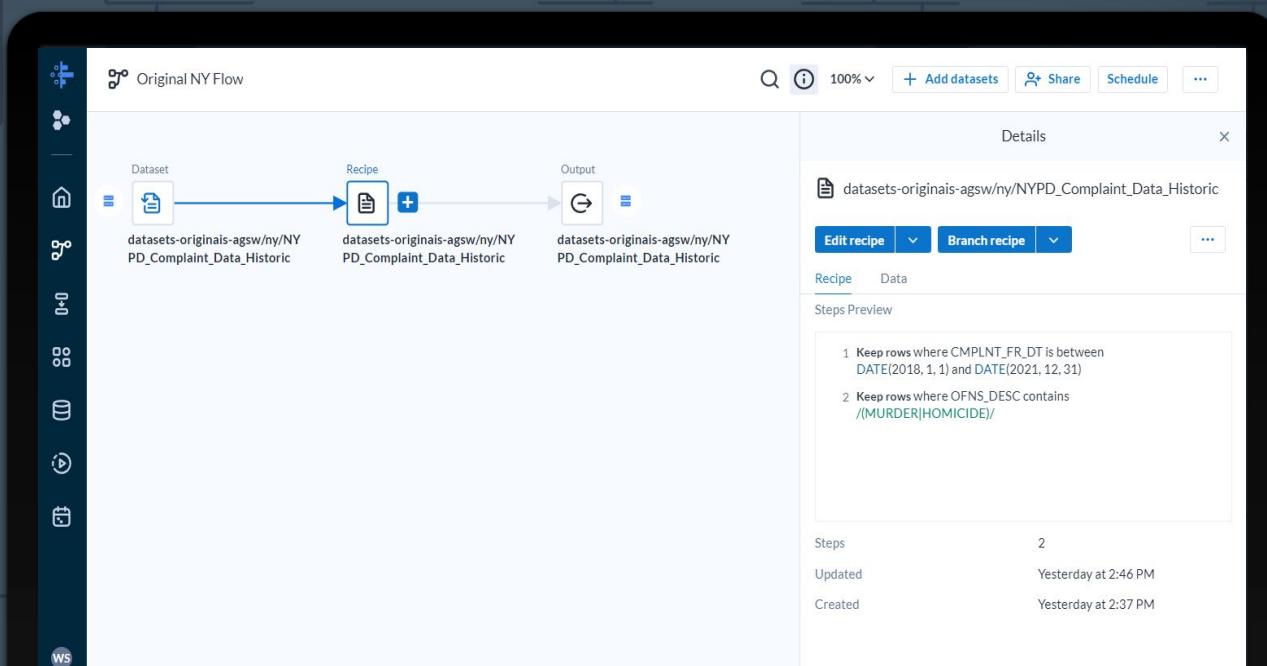
    blob.upload_from_filename(arquivo)

    print(
        f"Arquivo {arquivo} enviado para {destino}."
    )

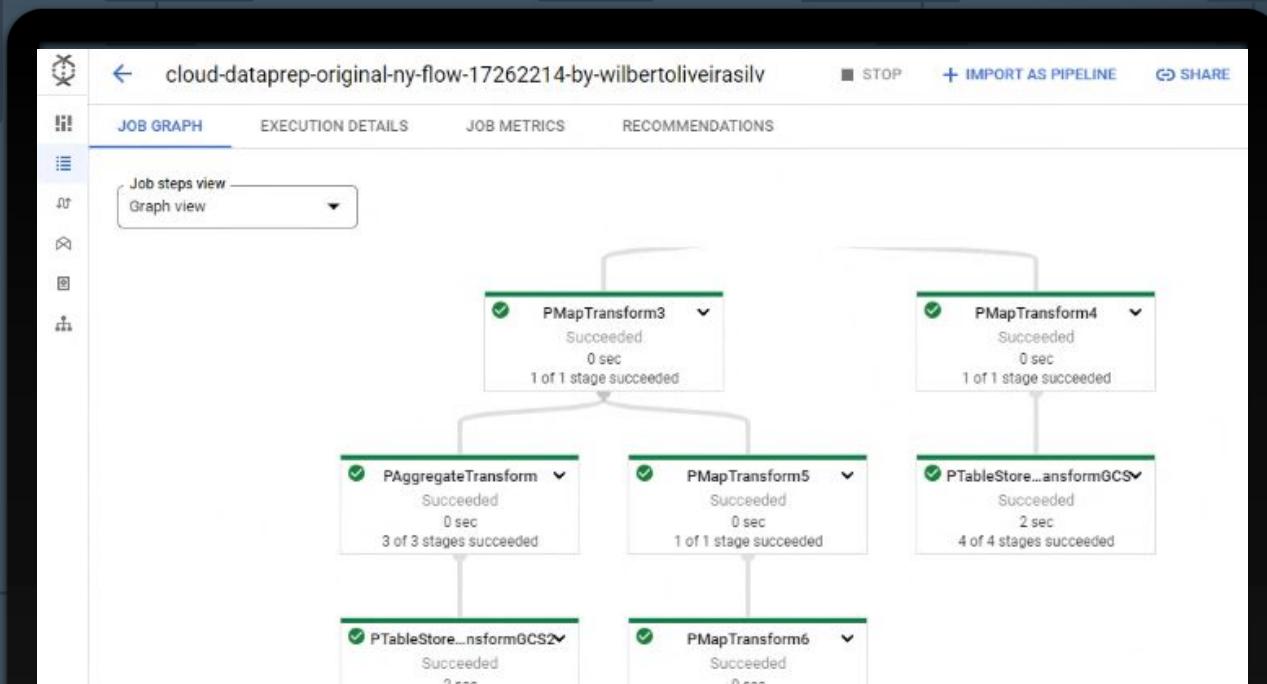
# Salvando variáveis necessárias
bucket = 'projeto-final-agsw'
arquivo = ['/content/homicidios_Sp_orig.xlsx','/content/lesao_Sp_orig.xlsx',
           '/content/policia_Sp_orig.xlsx','/content/latrocinio_Sp_orig.xlsx']
destino = ['originais/homicidios_Sp_orig.xlsx','originais/lesao_Sp_orig.xlsx',
           'originais/policia_Sp_orig.xlsx','originais/latrocinio_Sp_orig.xlsx']
y = 0

# Realizando upload do arquivo
for x in arquivo:
    upload_blob(bucket, x, destino[y])
    y +=1
```

# Extração Dataprep



# Load no GCS





# Load no GCS

## projeto-final-agsw

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Subject to object ACLs	None
Name	Size	Type	
<a href="#">homicidios_Sp_orig.xlsx</a>	2.1 MB	application/xlsx	
<a href="#">latrocinio_Sp_orig.xlsx</a>	167.7 KB	application/xlsx	
<a href="#">lesao_Sp_orig.xlsx</a>	88 KB	application/xlsx	
<a href="#">ny_orig.csv</a>	404.9 KB	text/csv	
<a href="#">policia_Sp_orig.xlsx</a>	697.9 KB	application/xlsx	

# Load no MySQL

- Carregamento para MySQL
  - Connector
  - connect
  - sqlalchemy
  - df.to\_sql

```
# Iniciando variável de conexão
connector = Connector()

# Gerando função de retorno da conexão
def getconn():
    conn = connector.connect(
        INSTANCE_CONNECTION_NAME,
        "pymysql",
        user=DB_USER,
        password=DB_PASS,
        db=DB_NAME
    )
    return conn

# Criando a conexão
pool = sqlalchemy.create_engine(
    "mysql+pymysql://",
    creator=getconn,
)

# Enviando para o MYSQL
# DF lesão
dfsp_homicidio.to_sql('dfsp_homicidio_orig', con=pool)
```

# Load no MySQL

SQL Overview EDIT IMPORT EXPORT RESTART STOP DELETE

CLOUD SHELL Terminal (projeto-final-373521) + -

```
mysql> use dados-originais;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_dados-originais |
+-----+
| dfny_orig           |
| dfsp_homicidio_orig |
| dfsp_latrocínio_orig |
| dfsp_lesao_orig      |
| dfsp_policia_orig    |
+-----+
5 rows in set (0.11 sec)

mysql> █
```

# Tratamento



# Pandas

Tratamento\_SP\_Pandas.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda As alterações não serão salvas

Índice

+ Código + Texto Copiar para o Drive

Tratamento dos Data Frames de São Paulo com Pandas

{x}

- 1 - Instalação e importação das bibliotecas
- 2 - Conexão com o DataLake
- 3 - Extração das bases de dados
- 4 - Padronização dos DF's
  - Dados de homicídios dolosos
  - Dados de latrocínio
  - Dados de Lesão corporal seguida de morte
  - Dados de morte decorrente de intervenção policial
- 5 - Concatenação dos arquivos e filtrando por cidade São Paulo



Tratamento dos Data Frames de São Paulo com Pandas

1 - Instalação e importação das bibliotecas

# Tratamentos Pandas

- Tratamentos
  - drop
  - rename

```
# Realizando drop das colunas
dfsp_hom.drop(labels=['HD',
                      'ID_DELEGACIA',
                      'MUNICIPIO_ELABORACAO',
                      'DP_ELABORACAO',
                      'SEC_ELABORACAO',
                      'DEP_ELABORACAO'], axis=1, inplace = True)
```

```
# Renomear as colunas Homicidio
dfsp_hom.rename(columns={

'DEPARTAMENTO_CIRCUNSCRICAO': 'departamento_circunscrecao',
'SECCIONAL_CIRCUNSCRICAO' : 'seccional_circunscrecao',
'MUNICIPIO_CIRCUNSCRICAO': 'cidade',
'DP_CIRCUNSCRICAO ': 'dp_circunscrecao','Nº DE VÍT HD': 'n_vitimas',
'MÊS ESTATISTICA': 'mes_ocorrencia','ANO ESTATISTICA': 'ano_ocorrencia',
'DATAHORA_REGISTRO_BO': 'datahora_bo',
'NUM_BO': 'num_bo','ANO_BO': 'ano_bo','DATA_FATO': 'dt_fato',
'HORA_FATO': 'hora_fato','DESC_TIPOLOCAL':'tipo_local_fato',
'LOGRADOURO': 'logradouro','NUMERO_LOGRADOURO': 'n_logradouro',
'LATITUDE': 'latitude','LONGITUDE': 'longitude',
'TIPO_PESSOA':'tipo_pessoa','SEXO_PESSOA': 'sexo_pessoa',
'IDADE_PESSOA': 'idade_pessoa','DATA_NASCIMENTO_PESSOA': 'dt_nasc_pessoa',
'COR_PELE': 'cor_pele','PROFISSAO':'profissao',
'NATUREZA_APURADA': 'tipo_crime'},inplace=True)
```

# Tratamentos Pandas

- Tratamentos
  - concat
  - reset\_index
  - drop de coluna e de linha

```
# Concat dos dados  
dfsp_geral = pd.concat([dfsp_hom, dfsp_lat, dfsp_les, dfsp_pol])
```

```
# Resetando o index  
dfsp_geral = dfsp_geral.reset_index()
```

```
# Dropando coluna com o index antigo  
dfsp_geral.drop(['index'], axis=1, inplace=True)
```

```
# Drop da linha inconsistente  
dfsp_geral.drop(labels=3009, axis=0, inplace=True)
```

# Tratamentos Pandas

## • Tratamentos

### ○ Filtrando cidade e range de data

```
# Filtrando por cidade = São Paulo  
ft = dfsp_geral['cidade'] == 'São Paulo'  
dfsp_geral = dfsp_geral.loc[ft]
```

```
# Drop das ocorrências de 2017  
ft2018 = dfsp_geral['ano_ocorrencia'] == 2018  
ft2019 = dfsp_geral['ano_ocorrencia'] == 2019  
ft2020 = dfsp_geral['ano_ocorrencia'] == 2020  
ft2021 = dfsp_geral['ano_ocorrencia'] == 2021  
  
dfsp_geral = dfsp_geral.loc[ft2018 | ft2019 | ft2020 | ft2021]
```

# Tratamentos Pandas

## Tratamentos

- strip
- drop\_duplicates
- replace

```
# Usando strip para retirar os espaços de antes e depois das string das colunas  
for x in colunas:  
    dfsp_geral[x] = dfsp_geral[x].str.strip()
```

```
# Deletar linhas duplicadas  
dfsp_geral.drop_duplicates(keep=False,inplace=True)
```

```
# Corrigir idade_pessoa de 118 para 52 (2018-1966)  
dfsp_geral.idade_pessoa.replace('118','52',inplace=True)
```

```
# Normalizando string 'Via Pública'  
dfsp_geral.tipo_local_fato.replace('VIA PÚBLICA','Via Pública',inplace=True)  
dfsp_geral.tipo_local_fato.replace('Via pública','Via Pública',inplace=True)
```

# Tratamentos Pandas

## Tratamentos

- slice
- to\_datetime

```
# Transformando colunas no formato string
dfsp_geral['dt_fato'] = dfsp_geral['dt_fato'].astype(str)
dfsp_geral['hora_fato'] = dfsp_geral['hora_fato'].astype(str)

# Juntando colunas
dfsp_geral['datahora_fato'] = \
    dfsp_geral['dt_fato'].str.slice(0,10) + ' ' + dfsp_geral['hora_fato']

# Tornando coluna datetime
dfsp_geral['datahora_fato']= pd.to_datetime(dfsp_geral['datahora_fato'])
```

# Tratamentos Pandas

## Tratamentos apply

```
# Criando função para juntar horarios
def define_periodo(linha):
    horario = linha['hora_fato']
    if '06:00:00' <= horario <= '11:59:00':
        return 'Manhã'
    elif '12:00:00' <= horario <= '17:59:00':
        return 'Tarde'
    elif '18:00:00' <= horario <= '23:58:00':
        return 'Noite'
    elif '00:00:00' <= horario <= '05:59:00':
        return 'Madrugada'
    else:
        return 'Não Informado'

# Aplicando função
dfsp_geral['periodo_ocorrencia'] = dfsp_geral.apply(define_periodo, axis=1)
```

# Tratamentos Pandas

- Tratamentos
  - astype()
  - Organização

```
# Juntando colunas
dfsp_geral['latitude_longitude'] = dfsp_geral['latitude'].astype(str) |
+ ',' + dfsp_geral['longitude'].astype(str)
```

```
# Organizando Data Frame
df = df[['tipo_crime','n_vitimas','datahora_fato','periodo_ocorrencia',
'tipo_pessoa', 'sexo_pessoa','dt_nasc_pessoa','idade_pessoa',
'faixa_etaria_pessoa', 'cor_pele', 'profissao',
'cidade','dp_circunscrecao','tipo_local_fato','logradouro',
'n_logradouro','latitude_longitude',
'corporacao_policial','situacao_policial']]
```



## Tratamentos Pandas

# Tratamentos split

```
# Criar novas colunas, a partir de uma única coluna
dfsp['dp_num_nome'] = dfsp['dp_circunscrecao'].str.split(';', expand=True)[0]
dfsp['bairro'] = dfsp['dp_circunscrecao'].str.split(';', expand=True)[1]
dfsp['distrito_zona'] = dfsp['dp_circunscrecao'].str.split(';', expand=True)[2]
```

# PySpark

Tratamento\_NY\_PySpark.ipynb

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda As alterações não serão salvas

+ Código + Texto Copiar para o Drive

Índice

- Tratamento do Data Frame de Nova York com PySpark
- {x} 1 - Instalação e importação das bibliotecas
- 2 - Conexões com o DataLake e SparkSession
  - Conectando com a Google Cloud Storage
  - SparkSession
- 3 - Extração das bases de dados
  - Criando Schema
  - Configurando Data Frame
- 6 - Tratamento do Data Frame
  - Renomeando colunas
  - Conferindo se há espaços extras



Tratamento do Data Frame de Nova York com PySpark

1 - Instalação e importação das bibliotecas

# Tratamentos PySpark

## Schema

### StructType

### Criação do DF

```
# Criando esquema de dados para determinar seus tipos
esquema = (
    StructType([
        StructField('CMPLNT_NUM', IntegerType()), StructField('CMPLNT_FR_DT', StringType()),
        StructField('CMPLNT_FR_TM', StringType()), StructField('CMPLNT_TO_DT', StringType()),
        StructField('CMPLNT_TO_TM', StringType()), StructField('ADDR_PCT_CD', StringType()),
        StructField('RPT_DT', StringType()), StructField('KY_CD', IntegerType()),
        StructField('OFNS_DESC', StringType()), StructField('PD_CD', FloatType()),
        StructField('PD_DESC', StringType()), StructField('CRM_ATPT_CPTD_CD', StringType()),
        StructField('LAW_CAT_CD', StringType()), StructField('BORO_NM', StringType()),
        StructField('LOC_OF_OCCUR_DESC', StringType()), StructField('PREM_TYP_DESC', StringType()),
        StructField('JURIS_DESC', StringType()), StructField('JURISDICTION_CODE', FloatType()),
        StructField('PARKS_NM', StringType()), StructField('HADEVELOPT', StringType()),
        StructField('HOUSING_PSA', StringType()), StructField('X_COORD_CD ', FloatType()),
        StructField('Y_COORD_CD', FloatType()), StructField('SUSP_AGE_GROUP', StringType()),
        StructField('SUSP_RACE', StringType()), StructField('SUSP_SEX', StringType()),
        StructField('TRANSIT_DISTRICT', FloatType()), StructField('Latitude', FloatType()),
        StructField('Longitude', FloatType()), StructField('Lat_Lon', StringType()),
        StructField('PATROL_BORO', StringType()), StructField('STATION_NAME', StringType()),
        StructField('VIC_AGE_GROUP', StringType()), StructField('VIC_RACE', StringType()),
        StructField('VIC_SEX', StringType())]))
```

# Tratamentos PySpark

## Renomeando colunas

### withColumnRenamed

```
# Renomeando colunas
df = (
    df.withColumnRenamed('CMPLNT_NUM', 'id_ocorrencia')
    .withColumnRenamed('CMPLNT_FR_DT', 'dt_inicio_ocorrencia')
    .withColumnRenamed('CMPLNT_FR_TM', 'hr_inicio_ocorrencia')
    .withColumnRenamed('CMPLNT_TO_DT', 'dt_final_ocorrencia')
    .withColumnRenamed('CMPLNT_TO_TM', 'hr_final_ocorrencia')
    .withColumnRenamed('ADDR_PCT_CD', 'cod_delegacia')
    .withColumnRenamed('RPT_DT', 'dt_comunicacao')
    .withColumnRenamed('KY_CD', 'cod_class_ofensa')
    .withColumnRenamed('OFNS_DESC', 'tipo_crime')
    .withColumnRenamed('PD_CD', 'cod_class_interna')
    .withColumnRenamed('PD_DESC', 'desc_class_interna')
    .withColumnRenamed('CRM_ATPT_CPTD_CD', 'tentado_consumado')
    .withColumnRenamed('LAW_CAT_CD', 'nivel_ofensa')
    .withColumnRenamed('BORO_NM', 'distrito')
    .withColumnRenamed('LOC_OF_OCCUR_DESC', 'loc_esp_ocorrencia')
    .withColumnRenamed('PREM_TYP_DESC', 'desc_local')
    .withColumnRenamed('JURIS_DESC', 'desc_foro')
    .withColumnRenamed('JURISDICTION_CODE', 'cod_foro')
    .withColumnRenamed('PARKS_NM', 'nome_parque'))
```



# Tratamentos PySpark

## Tratamentos

O **trim**

O **count**

```
# Usando o trim que tira apenas os espaços do início e fim da string
df = df.select([F.trim(F.col(c)).alias(c) for c in df.columns])
df.show(5, truncate=False)
```

```
# Função para mostrar a quantidade de dados nulos, nan e vazios de cada coluna
df.select(([F.count(F.when(F.col(c).contains('None') |
                           F.col(c).contains('NULL') |
                           (F.col(c) == '') |
                           F.col(c).isNull() |
                           F.isnan(c), c)).alias(c) for c in df.columns])
           ).show()
```

# Tratamentos PySpark

## Tratamentos

- `drop(*cols)`  
`withColumn`

```
# Listando colunas
cols = ('dt_final_ocorrencia', 'hr_final_ocorrencia', 'cod_class_interna',
        'desc_class_interna', 'distrito', 'desc_local', 'cod_foro',
        'nome_parque', 'nome_nycha', 'cod_nycha',
        'cod_dist_transito', 'distrito_patrulha', 'estacao_transporte')
```

```
# Dropando todas de vez  
df = df.drop(*cols)
```



# Tratamentos PySpark

## ● Tratamentos

- regex\_replace
- to\_timestamp

```
# Replace dos valores  
df = df.withColumn('data_ocorrencia',  
                    F.regexp_replace('data_ocorrencia', '/', '-'))
```

```
# Verificando alterações  
df.show(truncate=False)
```

```
# Converter data do formato string para o formato date - yyyy-MM-dd  
df = df.withColumn('data_ocorrencia',  
                    F.to_timestamp('data_ocorrencia','MM-dd-yyyy HH:mm:ss'))
```

```
# Verificando alterações  
df.show(5, truncate=False)
```

# Tratamentos PySpark

- Tratamentos
  - **withColumn**
  - **when/otherwise**

```
# Criando a coluna distrito a partir do cod da delegacia
df = df.withColumn('distrito', F.when(F.col('cod_delegacia') < 35, 'Manhattan')
                    .when(F.col('cod_delegacia') < 53, 'Bronx')
                    .when(F.col('cod_delegacia') < 95, 'Brooklyn')
                    .when(F.col('cod_delegacia') < 116, 'Queens')
                    .when(F.col('cod_delegacia') < 124, 'Staten Island')
                    .otherwise('Desconhecido')
)
```

```
# Criando coluna periodo_ocorrencia
df = df.withColumn('periodo_ocorrencia',
                    F.when(F.col("hr_inicio_ocorrencia"))
                      .between("06:00:00", "11:59:59"), 'Manhã')
                    .when(F.col("hr_inicio_ocorrencia"))
                      .between("12:00:00", "17:59:59"), 'Tarde')
                    .when(F.col("hr_inicio_ocorrencia"))
                      .between("18:00:00", "23:59:59"), 'Noite')
                    .when(F.col("hr_inicio_ocorrencia"))
                      .between("00:00:00", "05:59:59"), 'Madrugada')
                    .otherwise(F.col('hr_inicio_ocorrencia')))
```

# Tratamentos PySpark

- Tratamentos
  - Traduções
  - translate

```
# Traduzindo dados
df = (df.withColumn('cor_pele_suspeito',
    F.when(F.col('cor_pele_suspeito')=='WHITE','Branca')
    .when(F.col('cor_pele_suspeito')=='BLACK','Preta')
    .when(F.isnull(F.col('cor_pele_suspeito')),'Desconhecido')
    .when(F.col('cor_pele_suspeito')=='BLACK HISPANIC','Preta')
    .when(F.col('cor_pele_suspeito')=='WHITE HISPANIC','Branca')
    .when(F.col('cor_pele_suspeito')=='UNKNOWN','Desconhecido')
    .when(F.col('cor_pele_suspeito')=='ASIAN / PACIFIC ISLANDER','Amarela')
    .otherwise(F.col('cor_pele_suspeito'))))
)
df.show(truncate=False)
```

```
# Corrigindo inconsistências
df = df.withColumn('latitude_longitude',
    F.translate('latitude_longitude', '()', ''))
df.show(5, truncate=False)
```

# União dos DFs

A screenshot of a Jupyter Notebook interface. The title bar shows the file name "União dados SP e NY.ipynb". The menu bar includes Arquivo, Editar, Ver, Inserir, Ambiente de execução, Ferramentas, Ajuda, and a note "As alterações não serão salvas". The left sidebar displays a table of contents (Índice) with the following structure:

- Tratamento dos Data Frames de São Paulo e NY
  - 1 - Instalação e importação das bibliotecas
  - 2 - Conexão com o DataLake
  - 3 - Extração das bases de dados
  - 4 - Padronização dos DF's
    - Renomeando um coluna NY
    - Renomeando colunas SP
    - Criando coluna bairro e coluna distrito\_zona em SP
  - 5 - Concatenação dos arquivos
  - 6 - Upload de arquivo final
    - Envio ao MongoDB
    - Envio ao Google Cloud Storage

The main content area shows the first section expanded:

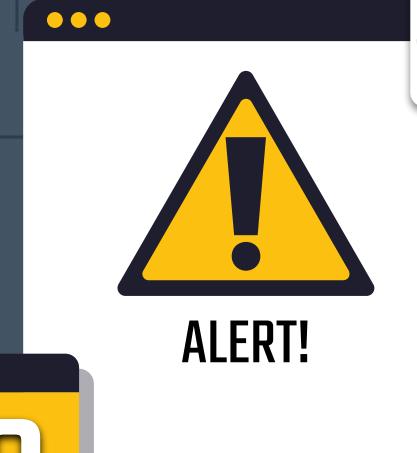
## Tratamento dos Data Frames de São Paulo e NY

Link para a documentação completa do projeto: [clique aqui](#)

- 1 - Instalação e importação das bibliotecas**
- 2 - Conexão com o DataLake**
- 3 - Extração das bases de dados**

# Carregamento (Load)

06



# Load no MongoDB

```
# Transformando Nulos em 0 (Mongo não aceita NaT)
df['dt_nasc_pessoa'].replace([np.nan, pd.NA],0, inplace=True)
df['datahora_fato'].replace([np.nan, pd.NA],0, inplace=True)
```

Overview

Real Time

Metrics

Collections

Search

Profiler

Performance Advisor

DATABASES: 1 COLLECTIONS: 3

+ Create Database

Search Namespaces

projeto\_final

ny\_sp\_tratado

ny\_tratado

sp\_tratado

projeto\_final

LOGICAL DATA SIZE: 7.22MB STORAGE SIZE: 1.63MB INDEX SIZE: 388KB TOTAL COLLECTIONS: 3

Collection Name	Documents	Logical Data Size	Avg Document Size
ny_sp_tratado	5720	4.18MB	767B
ny_tratado	1559	709.3KB	466B
sp_tratado	4161	2.34MB	591B

## Load no GCS via Apache Beam

### Carregamento via Apache Beam

- pipeline\_options
- Criação do modelo
- Rodando modelo Dataflow

```
pipe = (
    p1
    |'Abrir arquivo csv'>> beam.io.ReadFromText('gs://projeto-final-agsw/tratados/sp_tratado.csv', skip_header_lines=False)
    |'Separador de dados'>> beam.Map(lambda record: record.split(','))
    |'Load final'>> beam.io.WriteToText('gs://projeto-final-agsw/tratados/sp_tratado_beam', file_name_suffix='.csv')
)

p1.run()
```

```
# Criando as configurações da Pipeline para conexão com o GCP
pipeline_options = {
    'project': 'projeto-final-373521',
    'runner': 'DataflowRunner',
    'region': 'southamerica-east1',
    'staging_location': 'gs://pipeline-apachebeam/staging/',
    'temp_location': 'gs://pipeline-apachebeam/temp/',
    'template_location': 'gs://pipeline-apachebeam/models/modelo_batch_sp'
}

serviceAccount = '/content/projeto-final-373521-25961e56ca37.json'
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = serviceAccount

# Transformando o dicionário para o tipo de PipelineOption
pipeline_options = PipelineOptions.from_dictionary(pipeline_options)

# Criando a Pipeline com as opções estabelecidas
p1 = beam.Pipeline(options=pipeline_options)
```



## Load no BigQuery via Apache Beam

- Carregamento via Apache Beam
  - Criação de classe DataIngestion

```
# Classe com os passos para transformar um arquivo CSV
# em um formato aceito pelo BigQuery
class DataIngestion:
    # Método que pega uma linha de dados separados por ','
    # e transforma em dicionário para ser carregado no BigQuery
    def parse_method(self, string_input):
        # Separando a linha por ',' e removendo ''' , '\n' e '\r'
        values = re.split(",", re.sub('\r\n', '', re.sub("'''", '', string_input)))
        # Criando um dicionário com os dados
        row = dict(
            zip(('cidade', 'tipo_crime', 'distrito_zona', 'data_ocorrencia',
                 'periodo_ocorrencia', 'faixa_etaria_suspeito',
                 'cor_pele_suspeito', 'sexo_suspeito', 'faixa_etaria_vitima',
                 'cor_pele_vitima', 'sexo_vitima', 'latitude_longitude',
                 'desc_foro', 'n_vitimas', 'tipo_pessoa', 'dt_nasc_pessoa',
                 'idade_pessoa', 'profissao', 'tipo_local_fato',
                 'logradouro', 'n_logradouro', 'corporacao_policial',
                 'situacao_policial', 'dp_num_nome', 'bairro'),
                values))
        return row
```



## Load no BigQuery via Apache Beam

- Carregamento via Apache Beam
  - Definição de função run
  - Instanciamento pipeline\_options

```
# Função para criar o custom template
def run(argv=None):

    # Instanciando a classe DataIngestion
    data_ingestion = DataIngestion()

    # Argumentos da pipeline
    pipeline_options_dict = {
        'runner': 'DataflowRunner',
        'project': 'projeto-final-373521',
        'staging_location': 'gs://pipeline-apachebeam/staging',
        'temp_location': 'gs://pipeline-apachebeam/temp',
        'template_location': 'gs://pipeline-apachebeam/models/nyspGCStoBQ',
        'region': 'southamerica-east1',
        'input': 'gs://projeto-final-agsw/tratados/ny_sp_tratado.csv',
        'output': 'dataset_projeto_final_pipeline.ny_sp_tratado_base'
    }

    pipeline_options = PipelineOptions.from_dictionary(pipeline_options_dict)
```



## Load no BigQuery via Apache Beam

### Carregamento via Apache Beam

- beam
- run()

```
(p
    # Lendo o arquivo, pulando a primeira linha
    # que tem as informações das colunas
    | 'Lendo o CSV' >> beam.io.ReadFromText(pipeline_options_dict.get('input'),\
                                                skip_header_lines=1)
    # Usando o método parse_method da classe DataIngestion
    # para preparar os dados
    | 'Preparando os dados' >> beam.Map(lambda s: data_ingestion.parse_method(s))
    # Escrevendo no BigQuery
    | 'Escrevendo no BigQuery' >> beam.io.Write(
        beam.io.BigQuerySink(
            # Nome da tabela
            pipeline_options_dict.get('output'),
            # Esquema da tabela
            schema="cidade:STRING, tipo_crime:STRING, distrito_zona:STRING, \"\
                    \"data_ocorrencia:TIMESTAMP, periodo_ocorrencia:STRING,\"\
                    \"faixa_etaria_suspeito:STRING, cor_pele_suspeito:STRING,\"\
                    \"sexo_suspeito:STRING, faixa_etaria_vitima:STRING,\"\
                    \"cor_pele_vitima:STRING, sexo_vitima:STRING,\"\
                    \"latitude_longitude:STRING, desc_foro:STRING,\"\
                    \"n_vitimas:STRING, tipo_pessoa:STRING,\"\
                    \"dt_nasc_pessoa:STRING, idade_pessoa:STRING,\"\
                    \"profissao:STRING, tipo_local_fato:STRING, logradouro:STRING,\"\
                    \"n_logradouro:STRING, corporacao_policial:STRING, \"\
                    \"situacao_policial:STRING, dp_num_nome:STRING, bairro:STRING",
            # Cria a tabela se ela não existir
            create_disposition=beam.io.BigQueryDisposition.CREATE_IF_NEEDED,
            # Deleta os dados da tabela existente antes de escrever
            write_disposition=beam.io.BigQueryDisposition.WRITE_TRUNCATE))
    )
    p.run().wait_until_finish()
run()
```

# Jobs Dataflow

Google Cloud Projeto-Final dataflow Search

Jobs CREATE JOB FROM TEMPLATE ENABLE SORTING

Running Filter Filter jobs

Name	Type	Elapsed time	Status
nyspGCStoBQ	Batch	5 min 2 sec	Succeeded
sp_tratato_to_storage	Batch	5 min 14 sec	Succeeded
cloud-dataprep-original-ny-flow-17262214-by-wilbertoliveirasilv	Batch	18 min 55 sec	Succeeded



# Load no GCS

## projeto-final-agsw

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Subject to object ACLs	None
<input type="checkbox"/> Name		Size	Type
<input type="checkbox"/> <a href="#">ny_sp_tratado.csv</a>		1.3 MB	tex
<input type="checkbox"/> <a href="#">ny_tratado.csv</a>		286.2 KB	tex
<input type="checkbox"/> <a href="#">sp_tratado.csv</a>		927.7 KB	tex
<input type="checkbox"/> <a href="#">sp_tratado_beam-00000-of-00001.csv</a>		1.1 MB	tex



# Pré-análise

# Percentual da quantidade geral de vítimas por sexo

```
qtd = np.array([5023, 641, 56])
sexo = ['Masculino', 'Feminino', 'Desconhecido']

cores=['#40505f', '#849c6c', 'gold']
# O atributo explode indica que fatia do gráfico será destacada.
# No exemplo abaixo, será a primeira fatia.
A quantidade de valores é igual ao número de fatias do gráfico.
explode = (0.1, 0, 0,) # explode 1st slice

# Atribuindo um título ao gráfico
plt.title('Quantidade de vítimas por sexo')

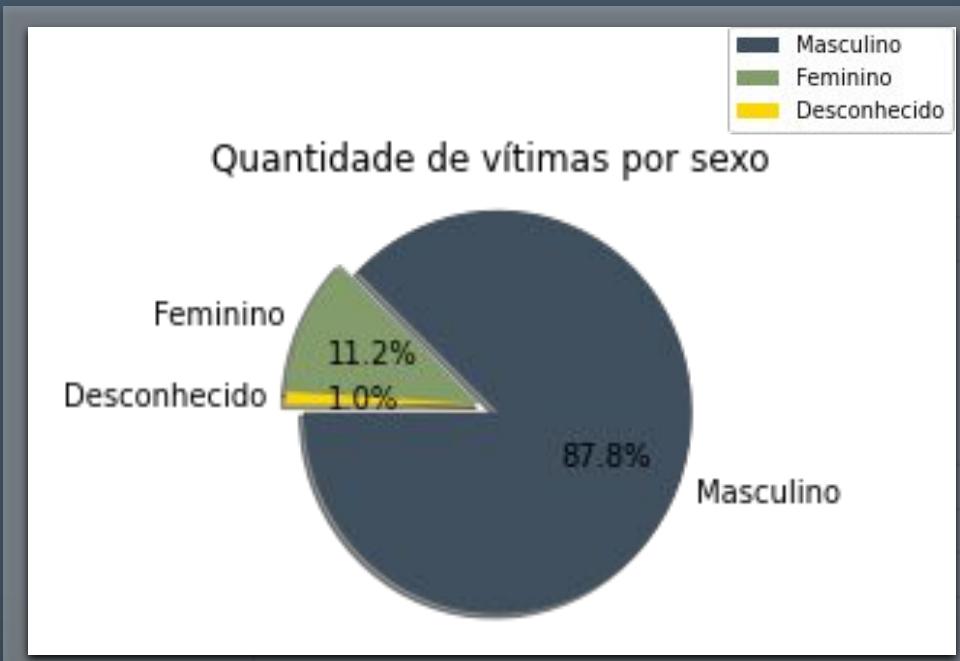
plt.pie(qtd, explode=explode, labels=sexo, colors=cores,
        autopct='%1.1f%%', shadow=True, startangle=180)

#Adiciona Legenda
plt.legend(sexo, bbox_to_anchor=(1.3, 1.3), loc='lower right')

#Centraliza o gráfico
plt.axis('equal')

#Ajusta o espaçamento para evitar o recorte do rótulo
plt.tight_layout()

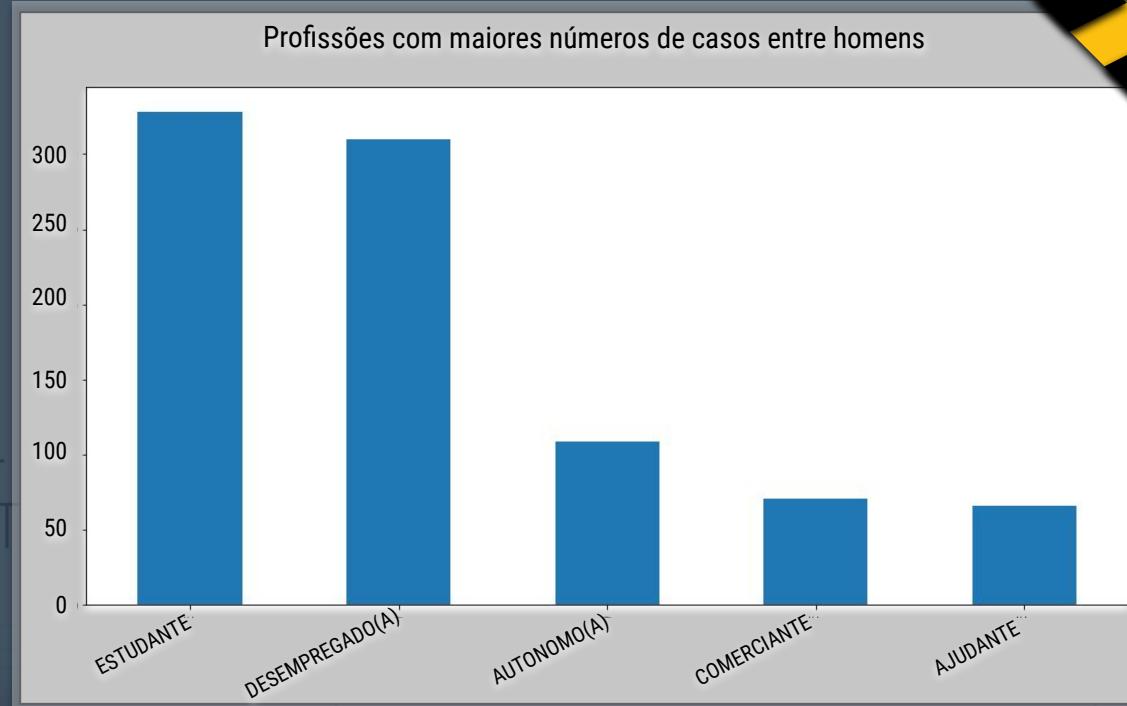
plt.show()
```





# Principais profissões declaradas das vítimas do sexo masculino em SP

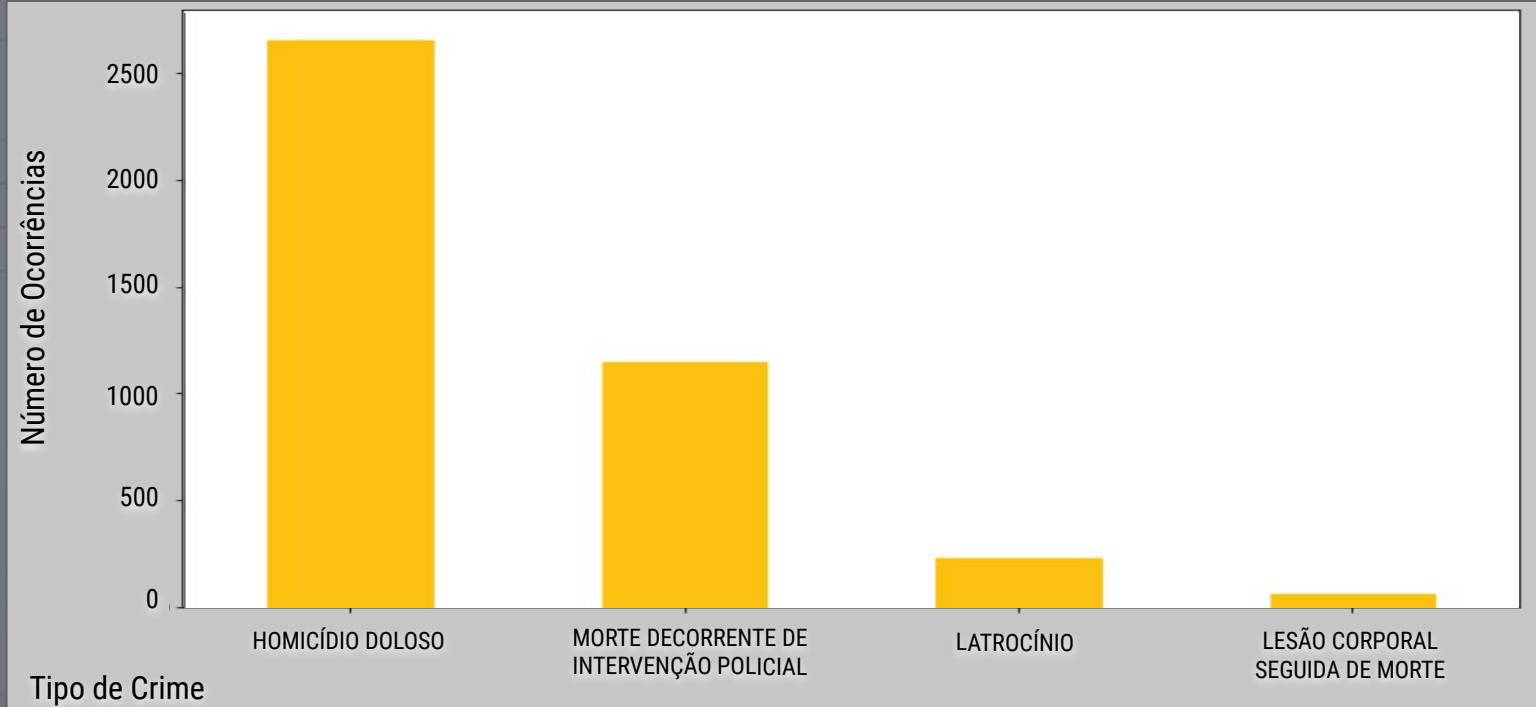
```
# Insight 3: Qual a profissão dos homens  
# com maior número de ocorrências?  
  
# Separando por sexo Masculino  
fthomem = df['sexo_pessoa'] == 'Masculino'  
ftnulo = df['profissao'] != 'NAO INFORMADA'  
case3 = df.loc[fthomem & ftnulo]  
  
# Criando novo Data Frame  
case3 = case3.groupby(case3['profissao'])  
    .size().sort_values(ascending=False).head(5)  
  
# Setando as variáveis  
fig, ax = plt.subplots(figsize = (16,8))  
case3.plot.bar()  
  
# Ajustando o gráfico  
plt.xlabel('Profissão \n')  
plt.ylabel('Número de ocorrências \n')  
plt.title('\n Profissões com maiores \n  
números de casos entre os Homens \n')  
  
#Plot  
plt.show()
```





## Ranking dos tipos de crimes de morte em SP

```
case1 = df.groupby(df['tipo_crime']).size().sort_values(ascending=False)
case1.plot.bar(figsize=(12,8), xlabel='Tipo de Crime',
               ylabel='Número de ocorrências',
               color='#fbc010')
```



• • •

# Quantidade de ocorrências por dia do mês em SP

```
# Isolando os dados
dias = list(sorted(df['datahora_fato'].dt.day.unique()))
valores_dia = df.groupby(df['datahora_fato'].dt.day).size()

# Criando novo Data Frame
case2 = pd.DataFrame({'dias': dias, 'total_oc' : valores_dia})
```

```
# Setando as variáveis
fig, ax = plt.subplots(figsize = (16,3))
case2.plot('dias', 'total_oc', ax = ax, color ='#fbc010')

# Ajustando o gráfico
ax.set_xticks(range(len(case2['dias'])))
ax.set_xticklabels(case2['dias'])
plt.xlabel('Dias do Mês \n')
plt.ylabel('Número de ocorrências \n')
plt.title('\n Número de ocorrências a cada dia do mês \n')

# Plot
plt.show()
```





# Perfil das vítimas de crime de morte em NY

```
# Faixa etária da vítima
spark.sql('''SELECT faixa_etaria_vitima, COUNT(*) AS crimes_morte
FROM ny_tratado
WHERE NOT faixa_etaria_vitima = "Desconhecido"
GROUP BY faixa_etaria_vitima
ORDER BY crimes_morte DESC''').show(truncate=False)
```

faixa_etaria_vitima	crimes_morte
25-44	803
18-24	299
45-64	245
<18	98
65+	86

```
# Cor de pele da vítima de crimes de morte por distrito
spark.sql('''SELECT distrito, cor_pele_vitima, COUNT(*) AS crimes_morte
FROM ny_tratado
WHERE NOT cor_pele_vitima = "Desconhecido"
GROUP BY distrito, cor_pele_vitima
ORDER BY distrito, crimes_morte DESC''').show(truncate=False)
```

distrito	cor_pele_vitima	crimes_morte
Bronx	Preta	371
Bronx	Branca	53
Bronx	Amarela	2
Brooklyn	Preta	443
Brooklyn	Branca	52
Brooklyn	Amarela	17
Desconhecido	Preta	3
Manhattan	Preta	193
Manhattan	Branca	55
Manhattan	Amarela	12
Queens	Preta	224
Queens	Branca	30
Queens	Amarela	24
Queens	Povos originários	1
Staten Island	Preta	45
Staten Island	Branca	14
Staten Island	Amarela	1



# Perfil dos suspeitos de crime de morte em NY

```
# Faixa etária do suspeito
spark.sql('''SELECT faixa_etaria_suspeito, COUNT(*) AS crimes_morte
FROM ny_tratado
WHERE NOT faixa_etaria_suspeito = "Desconhecido"
GROUP BY faixa_etaria_suspeito
ORDER BY crimes_morte DESC''').show(truncate=False)
```

faixa_etaria_suspeito	crimes_morte
25-44	542
18-24	243
45-64	151
<18	51
65+	9

```
# Cor de pele do suspeito de crimes de morte por distrito
spark.sql('''SELECT distrito, cor_pele_suspeito,COUNT(*) AS crimes_morte
FROM ny_tratado
WHERE NOT cor_pele_suspeito = "Desconhecido"
GROUP BY distrito, cor_pele_suspeito
ORDER BY distrito, crimes_morte DESC''').show(truncate=False)
```

distrito	cor_pele_suspeito	crimes_morte
Bronx	Preta	291
Bronx	Branca	37
Bronx	Amarela	1
Brooklyn	Preta	236
Brooklyn	Branca	33
Brooklyn	Amarela	5
Manhattan	Preta	146
Manhattan	Branca	34
Manhattan	Amarela	2
Queens	Preta	122
Queens	Branca	23
Queens	Amarela	23
Staten Island	Preta	27
Staten Island	Branca	10
Staten Island	Amarela	2



# Consultas BigQuery

# BigQuery

The screenshot shows the Google Cloud Platform BigQuery Explorer interface. At the top, there's a navigation bar with the Google Cloud logo, the project name "Projeto-Final", and a search bar labeled "Pesquisa Produtos, recursos, documentos (/)". Below the navigation bar is the "Explorer" section, which displays a search bar with placeholder text "Digite para pesquisar" and a message "Visualizando todos os recursos. [Mostrar apenas recursos com estrela.](#)". The main tree view shows the project structure:

- projeto-final-373521
  - Conexões externas
  - Consultas salvas (13)
  - Consultas do projeto
  - dataset\_projeto\_final
    - ny\_sp\_tratado
    - ny\_sp\_tratado\_base
    - ny\_tratado
    - sp\_tratado

On the right side of the interface, there are buttons for "EXECUTAR", "SALVAR", "COMPARTILHAR", and "PROG". A yellow button at the bottom center of the screen contains the text "CLIQUE AQUI".

CLIQUE AQUI

# BigQuery

## ● Crimes por dia de semana

Row	dia_semana	crimes_morte_ny	crimes_morte_sp
1	Domingo	290	747
2	Sábado	268	729
3	Segunda	212	544
4	Sexta	210	570
5	Quarta	200	497
6	Terça	200	544
7	Quinta	179	530

```
1 --Selecionando todos os crimes e agrupando por dia da semana,
2 --Ordenando por quantidade de crimes de morte
3 SELECT
4   (CASE EXTRACT(DAYOFWEEK
5     FROM
6       data_ocorrencia)
7     WHEN 1 THEN 'Domingo'
8     WHEN 2 THEN 'Segunda'
9     WHEN 3 THEN 'Terça'
10    WHEN 4 THEN 'Quarta'
11    WHEN 5 THEN 'Quinta'
12    WHEN 6 THEN 'Sexta'
13    WHEN 7 THEN 'Sábado'
14  END
15  ) AS dia_semana,
16  COUNT(CASE
17    WHEN cidade = 'Nova York' THEN 1
18    ELSE
19      NULL
20  END
21  ) AS crimes_morte_ny,
22  COUNT(CASE
23    WHEN cidade = 'São Paulo' THEN 1
24    ELSE
25      NULL
26  END
27  ) AS crimes_morte_sp
28 FROM
29   `dataset_projeto_final.ny_sp_tratado`
30 GROUP BY
31   dia_semana
32 ORDER BY
33   crimes_morte_ny DESC
```

# BigQuery

- Crimes por período do dia

Row	periodo_ocorrencia	crimes_morte_ny	crimes_morte_sp
1	Noite	577	1476
2	Madrugada	453	1016
3	Tarde	352	834
4	Manhã	177	811

```
1 --Selecionando todos os crimes e agrupando por período,
2 --ordenando por quantidade de crimes, não exibindo "Não Informado"
3 SELECT
4   periodo_ocorrencia,
5   COUNT(CASE
6     | WHEN cidade = 'Nova York' THEN 1
7     | ELSE
8     |   NULL
9     | END
0   ) AS crimes_morte_ny,
1   COUNT(CASE
2     | WHEN cidade = 'São Paulo' THEN 1
3     | ELSE
4     |   NULL
5     | END
6   ) AS crimes_morte_sp
7   FROM
8   `dataset_projeto_final.ny_sp_tratado`
9   WHERE
10  NOT periodo_ocorrencia = "Não Informado"
11  GROUP BY
12  periodo_ocorrencia
13  ORDER BY
14  crimes_morte_ny DESC
```

# BigQuery

## ● Crimes por mês NY

Row	mes	crimes_morte_ny
1	Agosto	179
2	Setembro	172
3	Julho	156
4	Junho	146
5	Outubro	142
6	Maio	129
7	Abril	127
8	Dezembro	118
9	Janeiro	103
10	Março	101
11	Novembro	100
12	Fevereiro	86

```
1 -- Selecionando todos os crimes e agrupando por mês,
2 -- ordenando por quantidade de crimes em NY
3 SELECT
4   (CASE EXTRACT(month
5     FROM
6       data_ocorrencia)
7      WHEN 1 THEN 'Janeiro'
8      WHEN 2 THEN 'Fevereiro'
9      WHEN 3 THEN 'Março'
0      WHEN 4 THEN 'Abril'
1      WHEN 5 THEN 'Maio'
2      WHEN 6 THEN 'Junho'
3      WHEN 7 THEN 'Julho'
4      WHEN 8 THEN 'Agosto'
5      WHEN 9 THEN 'Setembro'
6      WHEN 10 THEN 'Outubro'
7      WHEN 11 THEN 'Novembro'
8      WHEN 12 THEN 'Dezembro'
9    END
0   ) AS mes,
1   COUNT(*) AS crimes_morte_ny
2 FROM
3   `dataset_projeto_final.ny_sp_tratado`
4 WHERE
5   cidade = 'Nova York'
6 GROUP BY
7   mes
8 ORDER BY
9   crimes_morte_ny DESC
```

# BigQuery

## ● Crimes por mês SP

Row	mes	crimes_morte_sp
1	Janeiro	390
2	Dezembro	376
3	Março	375
4	Maio	372
5	Abril	366
6	Fevereiro	341
7	Setembro	335
8	Outubro	329
9	Junho	325
10	Julho	323
11	Agosto	316
12	Novembro	313

```
1 -- Selecionando todos os crimes e agrupando por mês,  
2 -- ordenando por quantidade de crimes em SP  
3 SELECT  
4   (CASE EXTRACT(month  
5     FROM  
6       data_ocorrencia)  
7       WHEN 1 THEN 'Janeiro'  
8       WHEN 2 THEN 'Fevereiro'  
9       WHEN 3 THEN 'Março'  
10      WHEN 4 THEN 'Abril'  
11      WHEN 5 THEN 'Maio'  
12      WHEN 6 THEN 'Junho'  
13      WHEN 7 THEN 'Julho'  
14      WHEN 8 THEN 'Agosto'  
15      WHEN 9 THEN 'Setembro'  
16      WHEN 10 THEN 'Outubro'  
17      WHEN 11 THEN 'Novembro'  
18      WHEN 12 THEN 'Dezembro'  
19    END  
20   ) AS mes,  
21   COUNT(*) AS crimes_morte_sp  
22 FROM  
23   `dataset_projeto_final.ny_sp_tratado`  
24 WHERE  
25   cidade = 'São Paulo'  
26 GROUP BY  
27   mes  
28 ORDER BY  
29   crimes_morte_sp DESC
```

## Coluna nova por estação do ano

```
1 --Criando uma coluna para indicar a estação do ano
2 CREATE TABLE IF NOT EXISTS
3   `dataset_projeto_final.ny_sp_tratado` AS
4 SELECT
5   *,
6   (CASE when cidade = 'Nova York' then
7     (CASE
8       WHEN EXTRACT(month FROM data_ocorrencia) IN (12, 1, 2) THEN 'Inverno'
9       WHEN EXTRACT(month FROM data_ocorrencia) IN (3, 4, 5) THEN 'Primavera'
10      WHEN EXTRACT(month FROM data_ocorrencia) IN (6, 7, 8) THEN 'Verão'
11      WHEN EXTRACT(month FROM data_ocorrencia) IN (9, 10, 11) THEN 'Outono'
12    END
13  )
14  when cidade = 'São Paulo' then
15    (CASE
16      WHEN EXTRACT(month FROM data_ocorrencia) IN (12, 1, 2) THEN 'Verão'
17      WHEN EXTRACT(month FROM data_ocorrencia) IN (3, 4, 5) THEN 'Outono'
18      WHEN EXTRACT(month FROM data_ocorrencia) IN (6, 7, 8) THEN 'Inverno'
19      WHEN EXTRACT(month FROM data_ocorrencia) IN (9, 10, 11) THEN 'Primavera'
20    END
21  )
22  END) AS estacao
23 FROM
24   `dataset_projeto_final.ny_sp_tratado_base`
```

# Looker Studio





Looker Studio

# Crimes de Morte



**Comparativo entre as cidades  
de São Paulo e Nova York**

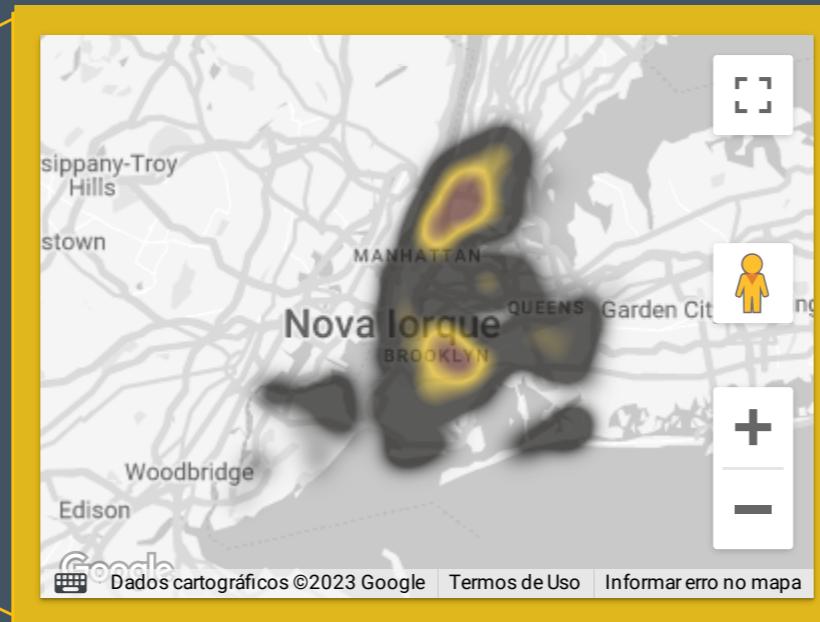
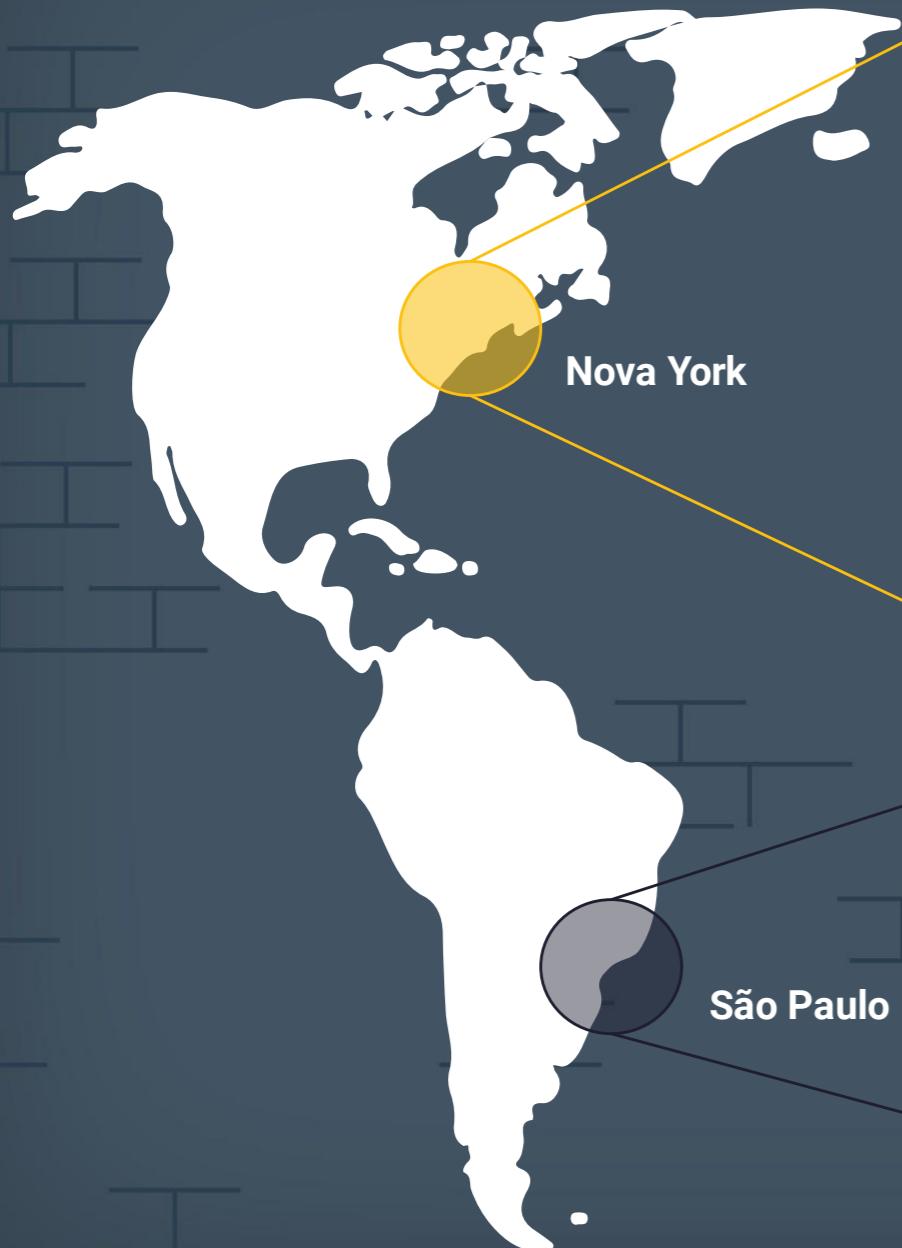
**CLIQUE AQUI**

# **Crimes de Morte**

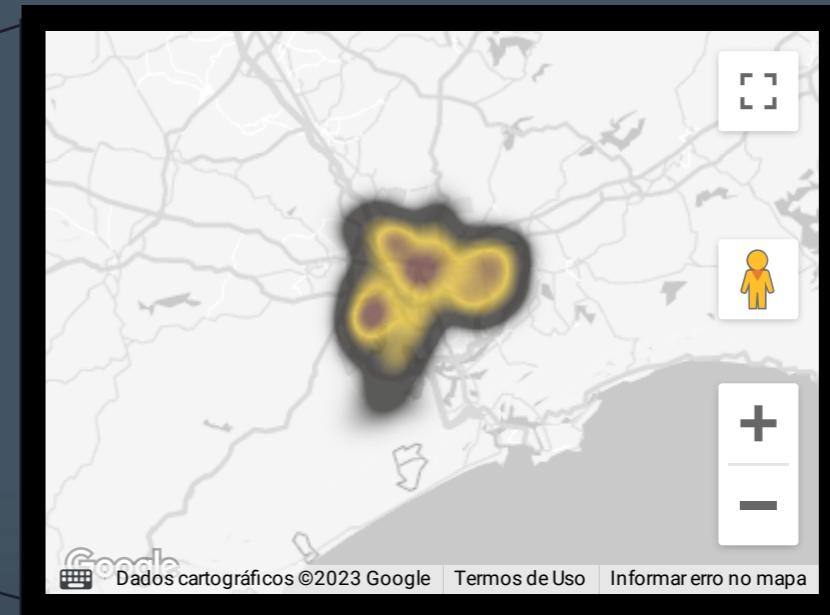


**Comparativo entre as cidades  
de São Paulo e Nova York**

# INCIDÊNCIA DE CRIMES DE MORTE



São Paulo



# OCORRÊNCIAS POR SAZONALIDADE

DO NOT CROSS

CRIME SCENE

DO NOT CROSS

CRIME SCENE

DO NOT CROSS



Cidades

Ano da Ocorrê...

Mês da Ocorrê...

Estação do Ano

Zona Distrital

Bairros SP

Sexo da Vítima

Cor da Pele da ...

Faixa Etária da ...

## LEGENDA

São Paulo

Nova York

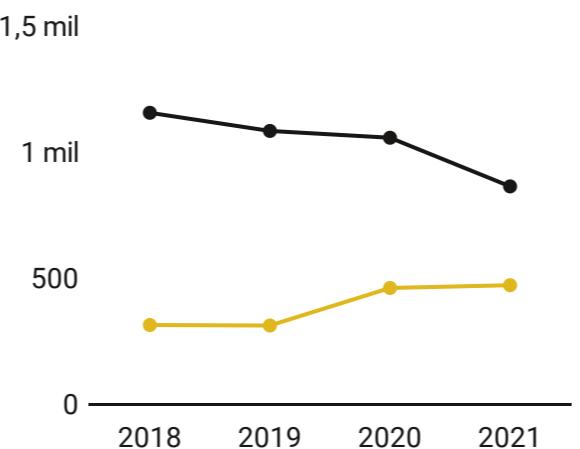
## Total Vítimas (NY)

**1.559**

## Total Vítimas (SP)

**4.161**

## Ano



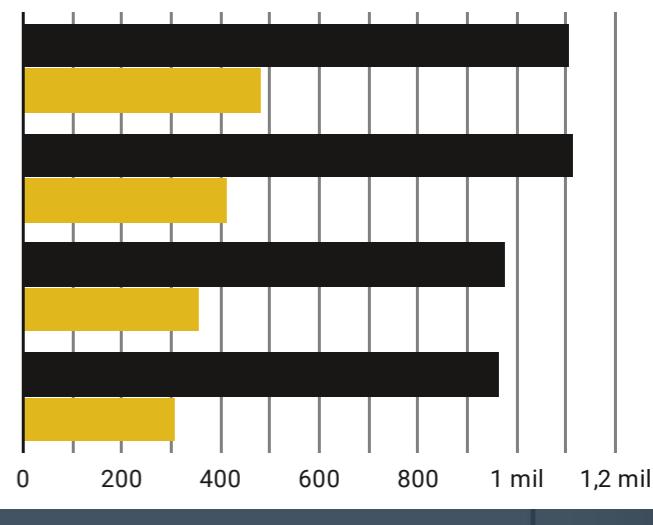
## Estações do Ano

Verão

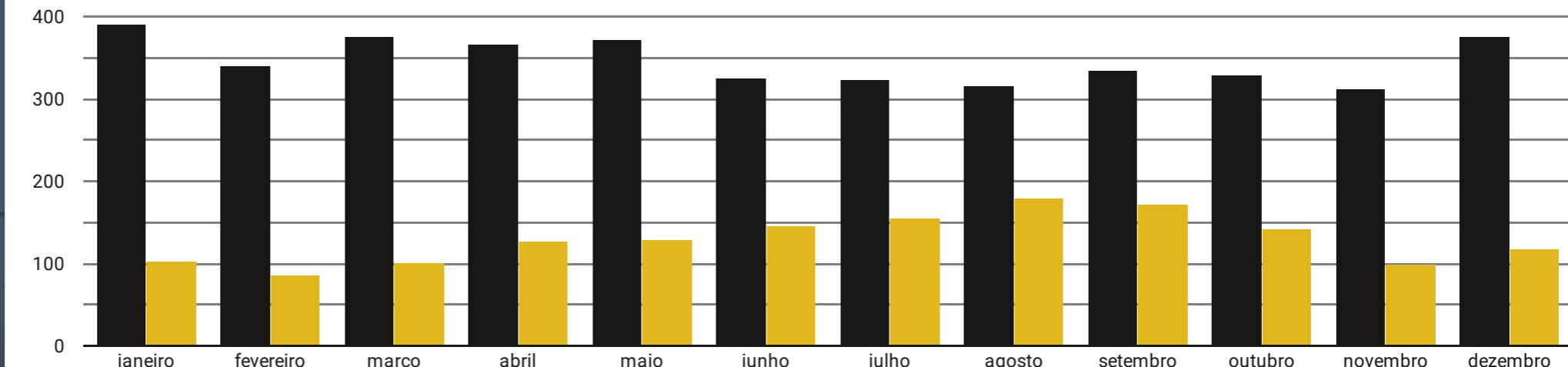
Outono

Primavera

Inverno



## Mês



# OCORRÊNCIAS POR PERÍODOS E LOCAIS

CRIME SCENE

DO NOT CROSS

CRIME SCENE

DO NOT CROSS

CRIME



Cidades

Ano da Ocorrê...

Mês da Ocorrê...

Estação do Ano

Zona Distrital

Bairros SP

Sexo da Vítima

Cor da Pele da ...

Faixa Etária da ...

## LEGENDA

São Paulo

Nova York

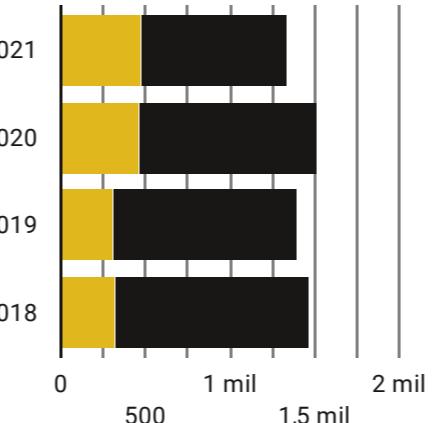
Total Vítimas (NY)

**1.559**

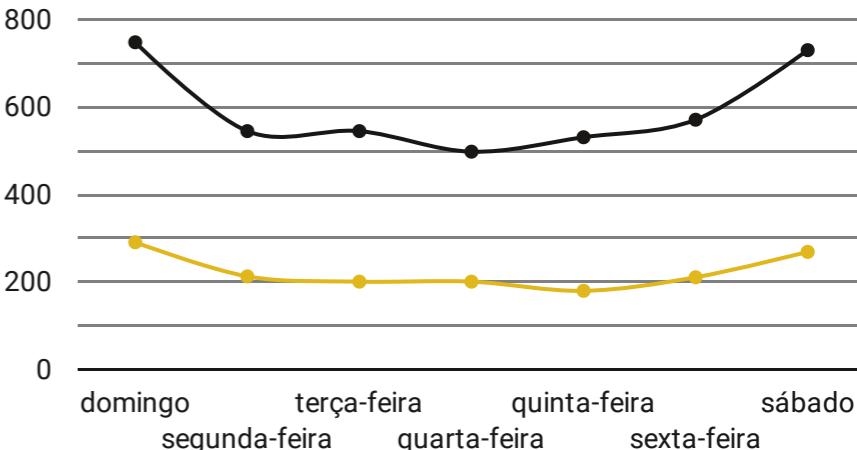
Total Vítimas (SP)

**4.161**

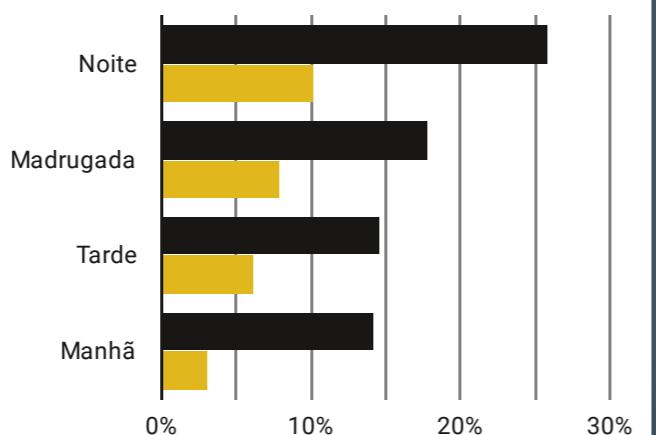
Número de Crimes



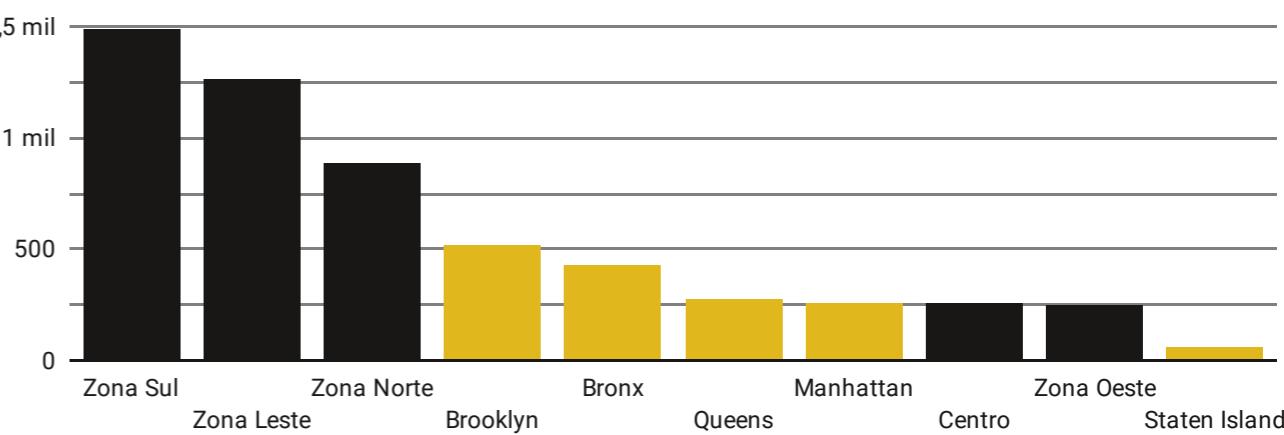
Dia da semana



Período



Zonas



# PERFIL DAS VÍTIMAS

CRIME SCENE

DO NOT CROSS

CRIME SCENE

DO NOT CROSS

CRIME SCENE

DO NOT CROSS



Cidades

Ano da Ocorrê...

Mês da Ocorrê...

Estação do Ano

Zona Distrital

Bairros SP

Sexo da Vítima

Cor da Pele da ...

Faixa Etária da ...

## LEGENDA

São Paulo

Nova York

### Total Vítimas (NY)

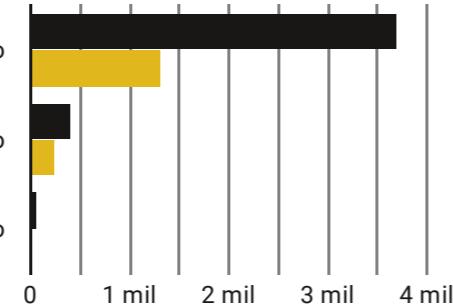
**1.559**

### Sexo

Masculino

Feminino

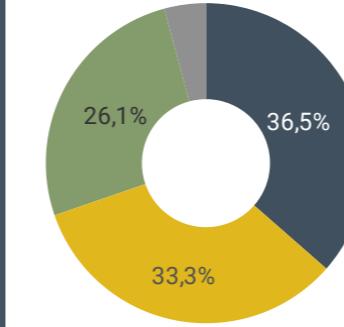
Desconhecido



### Total Vítimas (SP)

**4.161**

### Cor da Pele



- Parda
- Branca
- Preta
- outros

### Profissões das vítimas

ESTUDANTE

DESEMPREGADO(A)

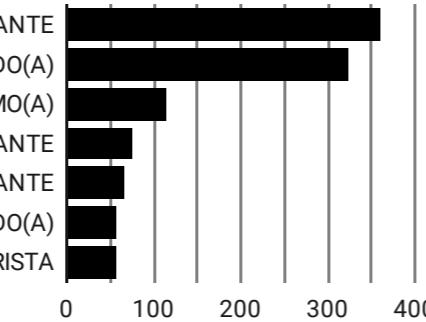
AUTONOMO(A)

COMERCIANTE

AJUDANTE

APOSENTADO(A)

MOTORISTA



### Faixa Etária

2 mil

1 mil

0

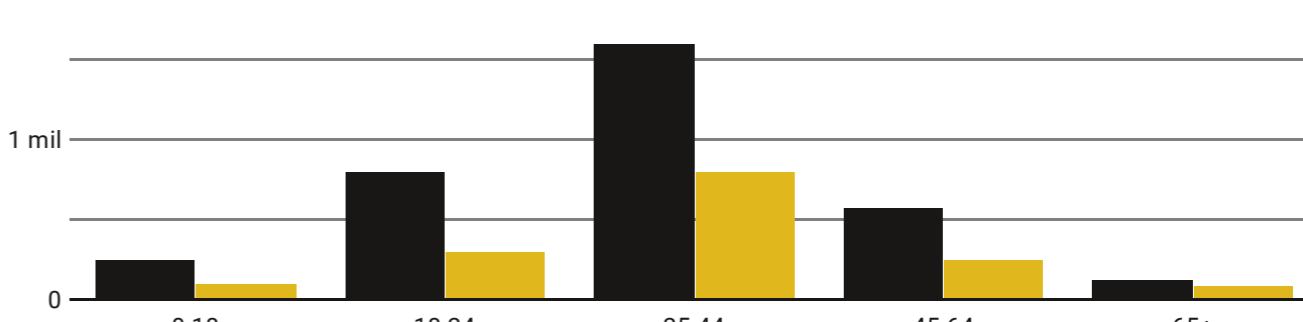
0-18

18-24

25-44

45-64

65+



# DETALHAMENTO NOVA YORK: SUSPEITOS

...

Ano da Ocorrência

Estação do Ano

Mês da Ocorrência

Tipo de Crime

Zona Distrital

Sexo Suspeito

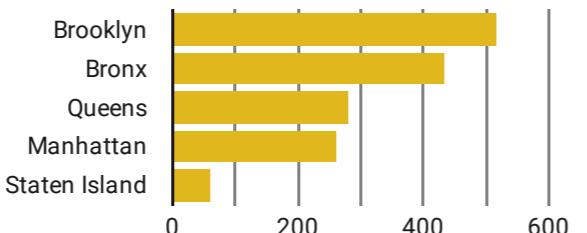
Cor Pele Suspeito

Faixa Etária Suspeito

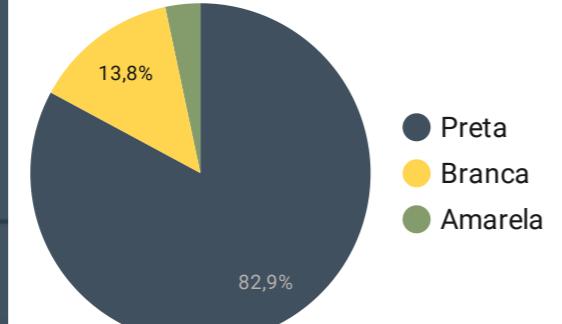
Total de vítimas (Acumulado)

**1.559**

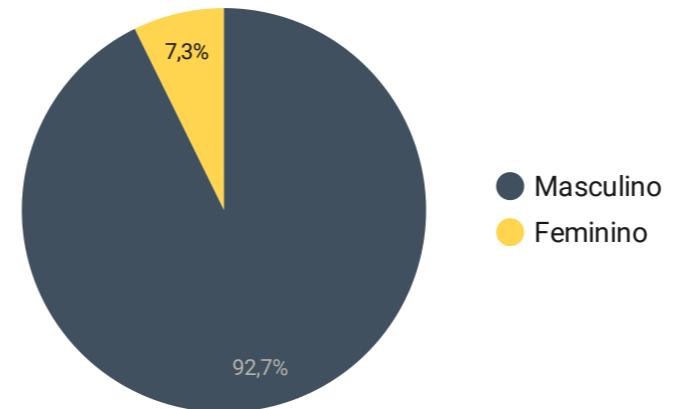
Crimes por região



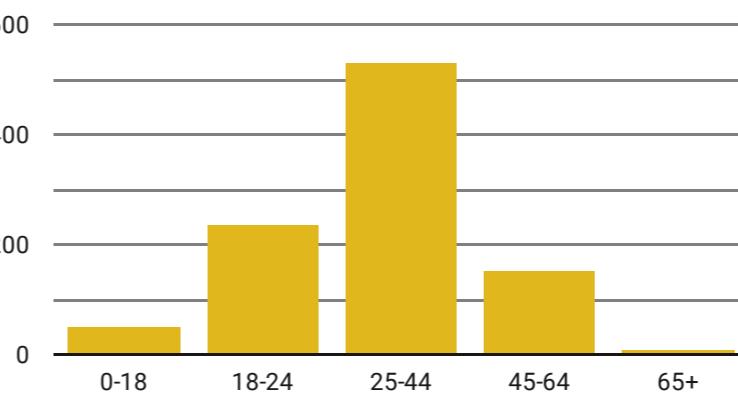
Cor da Pele dos suspeitos



Sexo dos suspeitos



Faixa etária dos suspeitos



# SP: MORTES POR INTERVENÇÃO DE POLICIAL

...

Ano Ocorrência

Estação do Ano

Mês da Ocorrê...

Tipo de Crime

DP

Bairro

Zona Distrital

Sexo da Vítima

Cor da Pele da ...

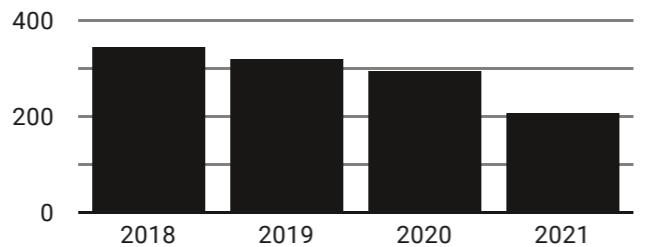
Faixa Etária da ...

Profissão da Vít...

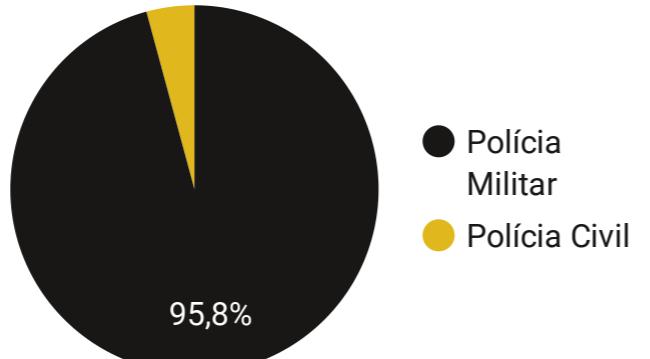
## Total de ocorrências

**1.165**

## Ano



## Corporação do policial



## DP

089 DP - Portal do Morumbi

073 DP - Jaçanã

075 DP - Jardim Arpoador

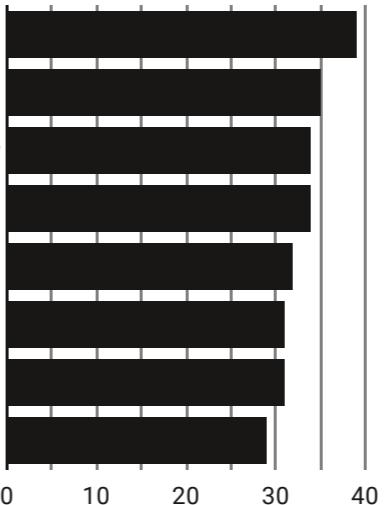
055 DP - Parque São Rafael

037 DP - Campo Limpo

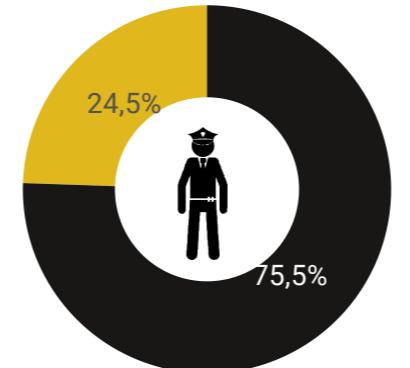
054 DP - Cidade Tiradentes

049 DP - São Mateus

044 DP - Guianaizes



## Situação do policial



# Custos do Projeto





## Custos do Projeto

Produto	Valor Estimado	Valor Utilizado
Compute Engine (Colab)	R\$ 112,93	R\$ 128,31
Cloud Storage	R\$ 0,18	R\$ 0,51
Networking Egress	R\$ 3,24	R\$ 1,15
BigQuery	R\$ 0,00	R\$ 0,00
Cloud SQL for MySQL	R\$ 110,69	R\$ 92,69
Dataflow	R\$ 1,12	R\$ 0,47
Dataprep by Trifacta	R\$ 0,53	R\$ 0,00
<b>Total</b>	<b>R\$ 228,69</b>	<b>R\$ 223,13</b>

# Considerações finais





## Considerações finais

- Busca da fonte de dados
- Filtro por localização e laço temporal
- Processo de ETL completo
- Insights com Pandas, PySpark e BigQuery
- Criação de Dashboard com Looker Studio
- Validação da efetividade dos dados



# Documentação



## Mindful Data Consultoria

Status	Finalizado
Autores	Anderson Calasans, Giovana de Brito, Suzana Gomes, Wilbert Silva
Likedin's	Empty
Data de entrega	January 17, 2023
Público em	Github, Linkedin, Notion
Link's públicos	links dos nossos github's
Type	Projeto Final - Soul Code
Tema	Crimes (Segurança Pública)
Área de estudo	Engenharia de dados, Python, SQL, MySQL, ETL, MongoDB, GCP



## Links e Referências

- **Dados Gerais**

- **Dados Demográficos de Nova York**

- <https://www.census.gov/>

- **Dados Demográficos de São Paulo**

- <https://www.ibge.gov.br/cidades-e-estados/sp.html>

- **Dados de Ocorrências Policiais de Nova York**

- <https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgeo-i56i>

- **Dados de Ocorrências Policiais de São Paulo**

- <http://www.ssp.sp.gov.br/transparenciassp/Consulta.aspx>

- 

- Calculadora GCP**

- <https://cloud.google.com/products/calculator#id=0cef5378-1437-41d8-abd3-eeeeccc5f3596>

- 

- Notion com todo o material**

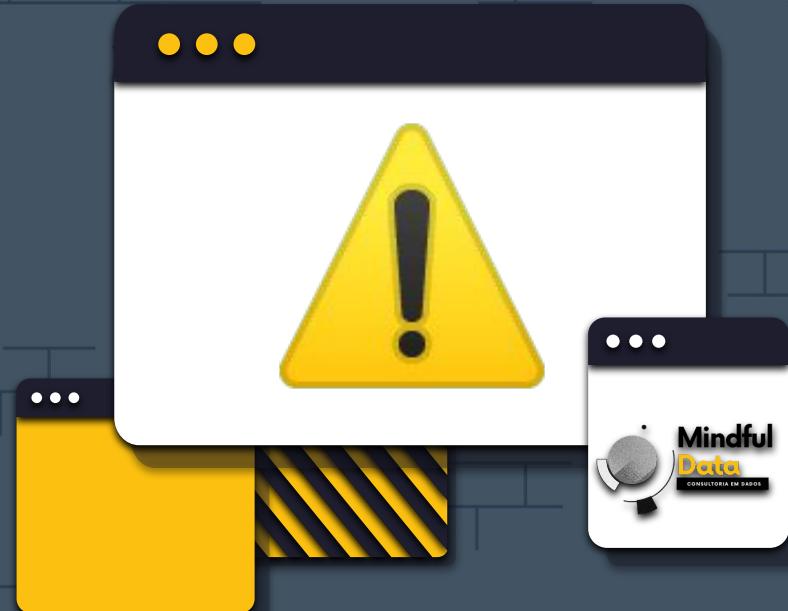
- <https://www.notion.so/Mindful-Data-Consultoria-d495964059c34acb9c8fe3d21dbb5cf6>



# Agradecimentos



# Dúvidas?



# Obrigado!



Anderson Gabriel Calasans



/in/gabriel-calasans/



AndersonGabrielCalasans



Suzana Gomes



/in/suzanag/



suzanagomes



Giovana de Brito



/in/giovanadebritosilva/



giobritos



Wilbert Silva



/in/wilbertsilva/



WilbertSilva