Sprint: Engenharia de Dados MVP

Aluno: Felipe Ribeiro da Silva

Objetivo

Documentação MVP

Realizar um estudo com base em dados públicos sobre veículos subtraídos, disponibilizados no Portal da SSP (Secretaria de Segurança Pública do Governo de São Paulo), com o objetivo de analisar estatisticamente as regiões, horários e os tipos e modelos de veículos mais roubados ou furtados no estado de São Paulo. A partir desses dados, busco contribuir para o aprimoramento da segurança pública, reforçando o patrulhamento nas áreas mais afetadas e orientando os cidadãos sobre medidas preventivas. Contudo, pretendo identificar os veículos com menor índice de roubo ou furto, auxiliando na escolha de um modelo mais seguro conforme a região de residência.

Perguntas à serem respondidas:

1) Quais são as regiões do Estado de São Paulo com maior incidência de roubos e furtos de veículos?

2) Em quais horários e dias da semana ocorrem mais roubos e furtos de veículos?

3) Quais são as três situações mais comuns nos eventos de roubo e furto de veículos?

4) Como os índices de roubo e furto de veículos variam ao longo do ano?

5) Quais são as marcas e modelos de veículos mais roubados ou furtados?

7) Qual a diferença entre a taxa de recuperação de veículos roubados e furtados?

8) Quais veículos apresentam menor risco de roubo ou furto em cada região?

9) Em quais regiões o risco de ter um veículo subtraído é menor?

10) O policiamento nas áreas mais afetadas tem impacto na redução dos crimes?

11) Há bairros ou cidades onde o reforço da segurança pública impede a ocorrência de roubos e furtos?

Instalação e Import de Bibliotecas

```
%pip install openpyxl
```

```
Python interpreter will be restarted.
Collecting openpyxl
  Downloading openpyxl-3.1.5-py2.py3-none-any.whl (250 kB)
Collecting et-xmlfile
  Downloading et_xmlfile-2.0.0-py3-none-any.whl (18 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-2.0.0 openpyxl-3.1.5
Python interpreter will be restarted.
```

```
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, min, max, lit
from pyspark.sql.types import NumericType, DateType, TimestampType
```

Coleta de dados

Verficação de dados no diretorio

```
display(dbutils.fs.ls("dbfs:/FileStore/shared_uploads/felipe.profissional16@gmail.com/"))
```

path	name	size	modificationTime
dbfs:/FileStore/shared_uploads/felipe.profissional16@gmail.com/VeiculosSubtraidos_2023.xlsx	VeiculosSubtraidos_2023.xlsx	82047243	1744162164000
dbfs:/FileStore/shared_uploads/felipe.profissional16@gmail.com/VeiculosSubtraidos_2024.xlsx	VeiculosSubtraidos_2024.xlsx	75459311	1744162176000
dbfs:/FileStore/shared_uploads/felipe.profissional16@gmail.com/VeiculosSubtraidos_2025.xlsx	VeiculosSubtraidos_2025.xlsx	5949384	1744162168000

Verificando diretório e arquivos

```
# Define o caminho do diretório
path = "dbfs:/mnt/dados_brutos/veiculos_subtraidos/"
```

```
# Garante que o diretório exista
dbutils.fs.mkdirs(path)

# Lista os arquivos dentro do diretório
files = dbutils.fs.ls(path)

# Verifica se o diretório contém arquivos
if files:
    print("⚠️ Arquivos já existem no diretório! Não é necessário baixar novamente.")
    print("📂 Arquivos encontrados:")
    for file in files:
        print(f"- {file.name}")
else:
    print("✅ Diretório vazio. Pode prosseguir com o download.")
```

```
⚠️ Arquivos já existem no diretório! Não é necessário baixar novamente.
📂 Arquivos encontrados:
- VeiculosSubtraidos_2023.csv
- VeiculosSubtraidos_2023.xlsx
- VeiculosSubtraidos_2024.csv
- VeiculosSubtraidos_2024.xlsx
- VeiculosSubtraidos_2025.csv
- VeiculosSubtraidos_2025.xlsx
```

Atenção !!! Caso não tenha arquivos, pode executar a celula abaixo ou importar os arquivos execel, aleterando **Caminho no DBFS**, na célula abaixo

```
#import requests

## URLs dos arquivos
urls = [
    "https://www.ssp.sp.gov.br/assets/estatistica/transparencia/baseDados/veiculosSub/VeiculosSubtraidos_2023.xlsx",
    "https://www.ssp.sp.gov.br/assets/estatistica/transparencia/baseDados/veiculosSub/VeiculosSubtraidos_2024.xlsx",
    "https://www.ssp.sp.gov.br/assets/estatistica/transparencia/baseDados/veiculosSub/VeiculosSubtraidos_2025.xlsx"
]

# Caminho no DBFS
dbfs_path = "dbfs:/mnt/dados_brutos/veiculos_subtraidos/"

# Criar diretório no DBFS se não existir
dbutils.fs.mkdirs(dbfs_path)

# Baixar e mover arquivos para DBFS
for url in urls:
    nome_arquivo = url.split("/")[-1]  # Nome do arquivo
    local_path = f"/tmp/{nome_arquivo}"  # Caminho temporário

    # Baixar o arquivo
    response = requests.get(url)
    with open(local_path, "wb") as file:
        file.write(response.content)  # Escrever o conteúdo baixado no arquivo local

    # Mover para o DBFS
    dbutils.fs.cp(f"file:{local_path}", f"{dbfs_path}{nome_arquivo}")

print("✅ Arquivos baixados e movidos para o DBFS.")
```

Criando os bancos de dados em Cloud

Recomendo criar um cluster para executar

```
%sql
-- Execura o DROP, se necessario, apagar as bases de dados e todos os objetos dentro delas caso esteja usando um cluster.

--DROP SCHEMA IF EXISTS bronze CASCADE;
-- DROP SCHEMA IF EXISTS silver CASCADE;
-- DROP SCHEMA IF EXISTS gold CASCADE;

-- Criar as bases de dados
CREATE DATABASE IF NOT EXISTS bronze;
CREATE DATABASE IF NOT EXISTS silver;
CREATE DATABASE IF NOT EXISTS gold;
```

Carga de dados

Carregando as bases e salvando na camada bronze

```
# Criar sessão Spark
spark = SparkSession.builder.appName("VeiculosSubtraidos").getOrCreate()

# Caminho no DBFS
dbfs_path = "dbfs:/mnt/dados_brutos/veiculos_subtraidos/"
files = [
    "VeiculosSubtraidos_2023.xlsx",
    "VeiculosSubtraidos_2024.xlsx",
    "VeiculosSubtraidos_2025.xlsx"
]

for file in files:
    try:
        print(f"📂 Processando {file}...")

        # Caminhos
        dbfs_file_path = f"{dbfs_path}{file}"
        local_xlsx_path = f"/tmp/{file}"
        local_csv_path = local_xlsx_path.replace(".xlsx", ".csv")
        dbfs_csv_path = f"{dbfs_path}{file.replace('.xlsx', '.csv')}"

        # Copiar do DBFS para local
        dbutils.fs.cp(dbfs_file_path, f"file:{local_xlsx_path}")

        # Ler com pandas e converter para string
        df = pd.read_excel(local_xlsx_path, engine="openpyxl")
        df = df.astype(str)

        # Salvar CSV com codificação UTF-8
        df.to_csv(local_csv_path, index=False, encoding='utf-8')

        # Copiar CSV para DBFS
        dbutils.fs.cp(f"file:{local_csv_path}", dbfs_csv_path)

        # Ler CSV com Spark (sem inferSchema)
        df_spark = spark.read.csv(dbfs_csv_path, header=True)

        # ✅ Força todas as colunas para string via selectExpr
        df_spark = df_spark.selectExpr([f"CAST(`{c}` AS STRING) AS `{c}`" for c in df_spark.columns])

        # Nome da tabela Bronze
        tabela_bronze = f"bronze.veiculos_subtraidos_{file.split('_')[-1].replace('.xlsx', '')}"

        # Salvar na camada Bronze
        df_spark.write.mode("overwrite").saveAsTable(tabela_bronze)

        print(f"✅ {file} carregado com sucesso na {tabela_bronze} 🚀")

    except Exception as e:
        print(f"❌ Erro ao processar {file}: {e}")
```

```
📂 Processando VeiculosSubtraidos_2023.xlsx...
✅ VeiculosSubtraidos_2023.xlsx carregado com sucesso na bronze.veiculos_subtraidos_2023 🚀
📂 Processando VeiculosSubtraidos_2024.xlsx...
✅ VeiculosSubtraidos_2024.xlsx carregado com sucesso na bronze.veiculos_subtraidos_2024 🚀
📂 Processando VeiculosSubtraidos_2025.xlsx...
✅ VeiculosSubtraidos_2025.xlsx carregado com sucesso na bronze.veiculos_subtraidos_2025 🚀
```

Verificando bases salvas

```
%sql
-- Exibir 10 linhas da tabela - (mudar o ano para consultar as tabelas)
SELECT *
FROM bronze.veiculos_subtraidos_2025
LIMIT 10;
```

ID_DELEGACIA	NOME_DEPARTAMENTO	NOME_SECCIONAL	NOME_DELEGACIA	NOME_MUNICIPIO	ANO_BO	NUM_BO	VERSAO	CIDADE
10108	DECAP	DEL.SEC.1º CENTRO	08º D.P. BRAS	S.PAULO	2025	AB6311	2	S.PAULO
10108	DECAP	DEL.SEC.1º CENTRO	08º D.P. BRAS	S.PAULO	2025	AB6311	2	S.PAULO
10108	DECAP	DEL.SEC.1º CENTRO	08º D.P. BRAS	S.PAULO	2025	AB6311	2	S.PAULO
10108	DECAP	DEL.SEC.1º CENTRO	08º D.P. BRAS	S.PAULO	2025	AB6311	2	S.PAULO

Modelagem Flat

Unindo as bases e savando na camada silver para tratamento

```
# Ler tabelas Bronze
df_bronze_2023 = spark.table("bronze.veiculos_subtraidos_2023")
df_bronze_2024 = spark.table("bronze.veiculos_subtraidos_2024")
df_bronze_2025 = spark.table("bronze.veiculos_subtraidos_2025")

# União com alinhamento automático de colunas
df_unificado = df_bronze_2023.unionByName(df_bronze_2024, allowMissingColumns=True) \
                             .unionByName(df_bronze_2025, allowMissingColumns=True)

# Salvar na camada Silver
df_unificado.write.mode("overwrite").saveAsTable("silver.veiculos_subtraidos")

print("✅ Dados combinados com sucesso na camada Silver! 🎉")
```

```
✅ Dados combinados com sucesso na camada Silver! 🎉
```

Verificando a base salva na camada Silver

```
%sql
-- Exibir 10 linhas da tabela
SELECT *
FROM silver.veiculos_subtraidos
LIMIT 10;
```

ID_DELEGACIA	NOME_DEPARTAMENTO	NOME_SECCIONAL	NOME_DELEGACIA	NOME_MUNICIPIO	ANO_BO	NUM_BO	VERSAO	NOME_DE
20210	DECAP	DEL.SEC.5º LESTE	10º D.P. PENHA DE FRANCA	S.PAULO	2022	JS5624	3	DECAP
20210	DECAP	DEL.SEC.5º LESTE	10º D.P. PENHA DE FRANCA	S.PAULO	2022	JS5624	3	DECAP
20210	DECAP	DEL.SEC.5º LESTE	10º D.P. PENHA DE FRANCA	S.PAULO	2022	JS5624	3	DECAP

Filtrando as colunas selecionadas para ajuste e limpeza, que serão utilizadas para o desenvolvimento das análises, e deixando uma tabela de cópia

```
%sql
-- Criar a tabela filtrada com as colunas desejadas
CREATE OR REPLACE TABLE silver.veiculos_subtraidos_tratamento AS
SELECT
    ID_DELEGACIA,
    NOME_DELEGACIA,
    NOME_MUNICIPIO,
    ANO_BO,
    NUM_BO,
```

```
    DATA_OCORRENCIA_BO,
    HORA_OCORRENCIA,
    DATAHORA_REGISTRO_BO,
    DATA_COMUNICACAO_BO,
    DATAHORA_IMPRESSAO_BO,
    DESCR_PERIODO,
    FLAG_ATO_INFRACIONAL,
    DESCR_TIPOLOCAL,
    CIDADE,
    BAIRRO,
    CEP,
    LOGRADOURO,
    LATITUDE,
    LONGITUDE,
    DESCR_OCORRENCIA_VEICULO,
    DESCR_TIPO_VEICULO,
    DESCR_MARCA_VEICULO,
    ANO_FABRICACAO,
    ANO_MODELO,
    PLACA_VEICULO,
    DESC_COR_VEICULO,
    MES,
    ANO,
    DESCR_CONDUTA
FROM silver.veiculos_subtraidos;
```

num_affected_rows	num_inserted_rows

Tratamento de dados

Tirando os espaços antes e após, um dado

```
%sql
UPDATE silver.veiculos_subtraidos_tratamento
SET
    ID_DELEGACIA = TRIM(ID_DELEGACIA),
    NOME_DELEGACIA = TRIM(NOME_DELEGACIA),
    NOME_MUNICIPIO = TRIM(NOME_MUNICIPIO),
    ANO_BO = TRIM(ANO_BO),
    NUM_BO = TRIM(NUM_BO),
    DATA_OCORRENCIA_BO = TRIM(DATA_OCORRENCIA_BO),
    HORA_OCORRENCIA = TRIM(HORA_OCORRENCIA),
    DATAHORA_REGISTRO_BO = TRIM(DATAHORA_REGISTRO_BO),
    DATA_COMUNICACAO_BO = TRIM(DATA_COMUNICACAO_BO),
    DATAHORA_IMPRESSAO_BO = TRIM(DATAHORA_IMPRESSAO_BO),
    DESCR_PERIODO = TRIM(DESCR_PERIODO),
    FLAG_ATO_INFRACIONAL = TRIM(FLAG_ATO_INFRACIONAL),
    DESCR_TIPOLOCAL = TRIM(DESCR_TIPOLOCAL),
    CIDADE = TRIM(CIDADE),
    BAIRRO = TRIM(BAIRRO),
    CEP = TRIM(CEP),
    LOGRADOURO = TRIM(LOGRADOURO),
    LATITUDE = TRIM(LATITUDE),
    LONGITUDE = TRIM(LONGITUDE),
    DESCR_OCORRENCIA_VEICULO = TRIM(DESCR_OCORRENCIA_VEICULO),
    DESCR_TIPO_VEICULO = TRIM(DESCR_TIPO_VEICULO),
    DESCR_MARCA_VEICULO = TRIM(DESCR_MARCA_VEICULO),
    ANO_FABRICACAO = TRIM(ANO_FABRICACAO),
    ANO_MODELO = TRIM(ANO_MODELO),
    PLACA_VEICULO = TRIM(PLACA_VEICULO),
    DESC_COR_VEICULO = TRIM(DESC_COR_VEICULO),
    MES = TRIM(MES),
    ANO = TRIM(ANO),
    DESCR_CONDUTA = TRIM(DESCR_CONDUTA);

-- Visualizando atualização
SELECT *
FROM silver.veiculos_subtraidos_tratamento
LIMIT 10;
```

ID_DELEGACIA	NOME_DELEGACIA	NOME_MUNICIPIO	ANO_BO	NUM_BO	DATA_OCORRENCIA_BO	HORA_OCORRENCIA	DATAHORA_REGI
20210	10º D.P. PENHA DE FRANCA	S.PAULO	2022	JS5624	2022-11-24	17:40:00	2023-01-17
20210	10º D.P. PENHA DE FRANCA	S.PAULO	2022	JS5624	2022-11-24	17:40:00	2023-01-17
20210	10º D.P. PENHA DE FRANCA	S.PAULO	2022	JS5624	2022-11-24	17:40:00	2023-01-17
20210	10º D.P. PENHA DE FRANCA	S.PAULO	2022	JS5624	2022-11-24	17:40:00	2023-01-17
20210	10º D.P. PENHA DE FRANCA	S.PAULO	2022	JS5624	2022-11-24	17:40:00	2023-01-17
20210	10º D.P. PENHA DE	S.PAULO	2022	JS5624	2022-11-24	17:40:00	2023-01-17

Removendo duplicadas

```
%sql
-- Remover duplicatas por boletim e criar a tabela com os dados sem duplicatas
CREATE OR REPLACE TABLE silver.veiculos_subtraidos_tratamento AS
SELECT
    ID_DELEGACIA,
    NOME_DELEGACIA,
    NOME_MUNICIPIO,
    ANO_BO,
    NUM_BO,
    DATA_OCORRENCIA_BO,
    HORA_OCORRENCIA,
    DATAHORA_REGISTRO_BO,
    DATA_COMUNICACAO_BO,
    DATAHORA_IMPRESSAO_BO,
    DESCR_PERIODO,
    FLAG_ATO_INFRACIONAL,
    DESCR_TIPOLOCAL,
    CIDADE,
    BAIRRO,
    CEP,
    LOGRADOURO,
    LATITUDE,
    LONGITUDE,
    DESCR_OCORRENCIA_VEICULO,
    DESCR_TIPO_VEICULO,
    DESCR_MARCA_VEICULO,
    ANO_FABRICACAO,
    ANO_MODELO,
    PLACA_VEICULO,
    DESC_COR_VEICULO,
    MES,
    ANO,
    DESCR_CONDUTA
FROM (
    SELECT *,
           ROW_NUMBER() OVER (PARTITION BY NOME_DELEGACIA, ANO_BO, NUM_BO ORDER BY DATA_OCORRENCIA_BO DESC) as row_num
    FROM silver.veiculos_subtraidos_tratamento
) tmp
WHERE row_num = 1;

-- Visualização da tabela
SELECT * FROM silver.veiculos_subtraidos_tratamento LIMIT 10;
```

ID_DELEGACIA	NOME_DELEGACIA	NOME_MUNICIPIO	ANO_BO	NUM_BO	DATA_OCORRENCIA_BO	HORA_OCORRENCIA	DATAHORA_REGI
70344	01º D.P. AMERICANA	AMERICANA	2023	AD4657	NaT	20:30:07	2023-06-05
70344	01º D.P. AMERICANA	AMERICANA	2023	AK1154	2023-01-09	19:29:00	2023-01-09
70344	01º D.P. AMERICANA	AMERICANA	2023	AP4174	2023-01-14	09:15:00	2023-01-14
70344	01º D.P. AMERICANA	GUARULHOS	2023	BK3360	2023-01-30	06:00:00	2023-02-03
70344	01º D.P.	AMERICANA	2023	BZ1873	2023-02-12	nan	2023-02-12

Identicando valores nulos

```
%sql
SELECT
    SUM(CASE WHEN ID_DELEGACIA IN ('nat', 'nan', 'null') OR ID_DELEGACIA IS NULL OR TRIM(ID_DELEGACIA) = '' THEN 1 ELSE 0 END) AS total_
    SUM(CASE WHEN NOME_DELEGACIA IN ('nat', 'nan', 'null') OR NOME_DELEGACIA IS NULL OR TRIM(NOME_DELEGACIA) = '' THEN 1 ELSE 0 END) AS
    SUM(CASE WHEN NOME_MUNICIPIO IN ('nat', 'nan', 'null') OR NOME_MUNICIPIO IS NULL OR TRIM(NOME_MUNICIPIO) = '' THEN 1 ELSE 0 END) AS
    SUM(CASE WHEN ANO_BO IN ('nat', 'nan', 'null') OR ANO_BO IS NULL OR TRIM(ANO_BO) = '' THEN 1 ELSE 0 END) AS total_nulos_ANO_BO,
    SUM(CASE WHEN NUM_BO IN ('nat', 'nan', 'null') OR NUM_BO IS NULL OR TRIM(NUM_BO) = '' THEN 1 ELSE 0 END) AS total_nulos_NUM_BO,
    SUM(CASE WHEN DATA_OCORRENCIA_BO IN ('nat', 'nan', 'null') OR DATA_OCORRENCIA_BO IS NULL OR TRIM(DATA_OCORRENCIA_BO) = '' THEN 1 ELS
    SUM(CASE WHEN HORA_OCORRENCIA IN ('nat', 'nan', 'null') OR HORA_OCORRENCIA IS NULL OR TRIM(HORA_OCORRENCIA) = '' THEN 1 ELSE 0 END)
    SUM(CASE WHEN DATAHORA_REGISTRO_BO IN ('nat', 'nan', 'null') OR DATAHORA_REGISTRO_BO IS NULL OR TRIM(DATAHORA_REGISTRO_BO) = '' THEN
    SUM(CASE WHEN DATA_COMUNICACAO_BO IN ('nat', 'nan', 'null') OR DATA_COMUNICACAO_BO IS NULL OR TRIM(DATA_COMUNICACAO_BO) = '' THEN 1
    SUM(CASE WHEN DATAHORA_IMPRESSAO_BO IN ('nat', 'nan', 'null') OR DATAHORA_IMPRESSAO_BO IS NULL OR TRIM(DATAHORA_IMPRESSAO_BO) = '' T
    SUM(CASE WHEN DESCR_PERIODO IN ('nat', 'nan', 'null') OR DESCR_PERIODO IS NULL OR TRIM(DESCR_PERIODO) = '' THEN 1 ELSE 0 END) AS tot
    SUM(CASE WHEN FLAG_ATO_INFRACIONAL IN ('nat', 'nan', 'null') OR FLAG_ATO_INFRACIONAL IS NULL OR TRIM(FLAG_ATO_INFRACIONAL) = '' THEN
    SUM(CASE WHEN DESCR_TIPOLOCAL IN ('nat', 'nan', 'null') OR DESCR_TIPOLOCAL IS NULL OR TRIM(DESCR_TIPOLOCAL) = '' THEN 1 ELSE 0 END)
    SUM(CASE WHEN CIDADE IN ('nat', 'nan', 'null') OR CIDADE IS NULL OR TRIM(CIDADE) = '' THEN 1 ELSE 0 END) AS total_nulos_CIDADE,
    SUM(CASE WHEN BAIRRO IN ('nat', 'nan', 'null') OR BAIRRO IS NULL OR TRIM(BAIRRO) = '' THEN 1 ELSE 0 END) AS total_nulos_BAIRRO,
    SUM(CASE WHEN CEP IN ('nat', 'nan', 'null') OR CEP IS NULL OR TRIM(CEP) = '' THEN 1 ELSE 0 END) AS total_nulos_CEP,
    SUM(CASE WHEN LOGRADOURO IN ('nat', 'nan', 'null') OR LOGRADOURO IS NULL OR TRIM(LOGRADOURO) = '' THEN 1 ELSE 0 END) AS total_nulos_
    SUM(CASE WHEN LATITUDE IN ('nat', 'nan', 'null') OR LATITUDE IS NULL OR TRIM(LATITUDE) = '' THEN 1 ELSE 0 END) AS total_nulos_LATITU
    SUM(CASE WHEN LONGITUDE IN ('nat', 'nan', 'null') OR LONGITUDE IS NULL OR TRIM(LONGITUDE) = '' THEN 1 ELSE 0 END) AS total_nulos_LON
    SUM(CASE WHEN DESCR_OCORRENCIA_VEICULO IN ('nat', 'nan', 'null') OR DESCR_OCORRENCIA_VEICULO IS NULL OR TRIM(DESCR_OCORRENCIA_VEICUL
    SUM(CASE WHEN DESCR_TIPO_VEICULO IN ('nat', 'nan', 'null') OR DESCR_TIPO_VEICULO IS NULL OR TRIM(DESCR_TIPO_VEICULO) = '' THEN 1 ELS
    SUM(CASE WHEN DESCR_MARCA_VEICULO IN ('nat', 'nan', 'null') OR DESCR_MARCA_VEICULO IS NULL OR TRIM(DESCR_MARCA_VEICULO) = '' THEN 1
    SUM(CASE WHEN ANO_FABRICACAO IN ('nat', 'nan', 'null') OR ANO_FABRICACAO IS NULL OR TRIM(ANO_FABRICACAO) = '' THEN 1 ELSE 0 END) AS
    SUM(CASE WHEN ANO_MODELO IN ('nat', 'nan', 'null') OR ANO_MODELO IS NULL OR TRIM(ANO_MODELO) = '' THEN 1 ELSE 0 END) AS total_nulos_
    SUM(CASE WHEN PLACA_VEICULO IN ('nat', 'nan', 'null') OR PLACA_VEICULO IS NULL OR TRIM(PLACA_VEICULO) = '' THEN 1 ELSE 0 END) AS tot
    SUM(CASE WHEN DESC_COR_VEICULO IN ('nat', 'nan', 'null') OR DESC_COR_VEICULO IS NULL OR TRIM(DESC_COR_VEICULO) = '' THEN 1 ELSE 0 EN
    SUM(CASE WHEN MES IN ('nat', 'nan', 'null') OR MES IS NULL OR TRIM(MES) = '' THEN 1 ELSE 0 END) AS total_nulos_MES,
    SUM(CASE WHEN ANO IN ('nat', 'nan', 'null') OR ANO IS NULL OR TRIM(ANO) = '' THEN 1 ELSE 0 END) AS total_nulos_ANO,
    SUM(CASE WHEN DESCR_CONDUTA IN ('nat', 'nan', 'null') OR DESCR_CONDUTA IS NULL OR TRIM(DESCR_CONDUTA) = '' THEN 1 ELSE 0 END) AS tot
FROM silver.veiculos_subtraidos_tratamento;
```

total_nulos_ID_DELEGACIA	total_nulos_NOME_DELEGACIA	total_nulos_NOME_MUNICIPIO	total_nulos_ANO_BO	total_nulos_NUM_BO	total_nul
0	0	0	0	0	0

Substituindo os valores para "Sem informação"

```
%sql
-- Atualizar os dados na tabela existente 'silver.veiculos_subtraidos_tratamento'
UPDATE silver.veiculos_subtraidos_tratamento
SET
    ID_DELEGACIA = CASE
        WHEN ID_DELEGACIA IN ('nat', 'nan', 'null') OR ID_DELEGACIA IS NULL OR TRIM(ID_DELEGACIA) = '' THEN 'Sem informação'
        ELSE ID_DELEGACIA
    END,

    NOME_DELEGACIA = CASE
        WHEN NOME_DELEGACIA IN ('nat', 'nan', 'null') OR NOME_DELEGACIA IS NULL OR TRIM(NOME_DELEGACIA) = '' THEN 'Sem informação'
        ELSE NOME_DELEGACIA
    END,

    NOME_MUNICIPIO = CASE
        WHEN NOME_MUNICIPIO IN ('nat', 'nan', 'null') OR NOME_MUNICIPIO IS NULL OR TRIM(NOME_MUNICIPIO) = '' THEN 'Sem informação'
        ELSE NOME_MUNICIPIO
    END,

    ANO_BO = CASE
        WHEN ANO_BO IN ('nat', 'nan', 'null') OR ANO_BO IS NULL OR TRIM(ANO_BO) = '' THEN 'Sem informação'
        ELSE ANO_BO
    END,

    NUM_BO = CASE
        WHEN NUM_BO IN ('nat', 'nan', 'null') OR NUM_BO IS NULL OR TRIM(NUM_BO) = '' THEN 'Sem informação'
        ELSE NUM_BO
    END,

    DATA_OCORRENCIA_BO = CASE
        WHEN DATA_OCORRENCIA_BO IN ('nat', 'nan', 'null') OR DATA_OCORRENCIA_BO IS NULL OR TRIM(DATA_OCORRENCIA_BO) = '' THEN 'Sem infor
        ELSE DATA_OCORRENCIA_BO
    END,

    HORA_OCORRENCIA = CASE
        WHEN HORA_OCORRENCIA IN ('nat', 'nan', 'null') OR HORA_OCORRENCIA IS NULL OR TRIM(HORA_OCORRENCIA) = '' THEN 'Sem informação'
```

```
    ELSE HORA_OCORRENCIA
END,

DATAHORA_REGISTRO_BO = CASE
    WHEN DATAHORA_REGISTRO_BO IN ('nat', 'nan', 'null') OR DATAHORA_REGISTRO_BO IS NULL OR TRIM(DATAHORA_REGISTRO_BO) = '' THEN 'Ser
    ELSE DATAHORA_REGISTRO_BO
END,

DATA_COMUNICACAO_BO = CASE
    WHEN DATA_COMUNICACAO_BO IN ('nat', 'nan', 'null') OR DATA_COMUNICACAO_BO IS NULL OR TRIM(DATA_COMUNICACAO_BO) = '' THEN 'Sem in
    ELSE DATA_COMUNICACAO_BO
END,

DATAHORA_IMPRESSAO_BO = CASE
    WHEN DATAHORA_IMPRESSAO_BO IN ('nat', 'nan', 'null') OR DATAHORA_IMPRESSAO_BO IS NULL OR TRIM(DATAHORA_IMPRESSAO_BO) = '' THEN
    ELSE DATAHORA_IMPRESSAO_BO
END,

DESCR_PERIODO = CASE
    WHEN DESCR_PERIODO IN ('nat', 'nan', 'null') OR DESCR_PERIODO IS NULL OR TRIM(DESCR_PERIODO) = '' THEN 'Sem informação'
    ELSE DESCR_PERIODO
END,

FLAG_ATO_INFRACIONAL = CASE
    WHEN FLAG_ATO_INFRACIONAL IN ('nat', 'nan', 'null') OR FLAG_ATO_INFRACIONAL IS NULL OR TRIM(FLAG_ATO_INFRACIONAL) = '' THEN 'Ser
    ELSE FLAG_ATO_INFRACIONAL
END,

DESCR_TIPOLOCAL = CASE
    WHEN DESCR_TIPOLOCAL IN ('nat', 'nan', 'null') OR DESCR_TIPOLOCAL IS NULL OR TRIM(DESCR_TIPOLOCAL) = '' THEN 'Sem informação'
    ELSE DESCR_TIPOLOCAL
END,

CIDADE = CASE
    WHEN CIDADE IN ('nat', 'nan', 'null') OR CIDADE IS NULL OR TRIM(CIDADE) = '' THEN 'Sem informação'
    ELSE CIDADE
END,

BAIRRO = CASE
    WHEN BAIRRO IN ('nat', 'nan', 'null') OR BAIRRO IS NULL OR TRIM(BAIRRO) = '' OR BAIRRO = '-' THEN 'Sem informação'
    ELSE BAIRRO
END,

CEP = CASE
    WHEN CEP IN ('nat', 'nan', 'null') OR CEP IS NULL OR TRIM(CEP) = '' THEN 'Sem informação'
    ELSE REGEXP_REPLACE(CEP, '\\.0$', '')  -- Ajuste para CEP, removendo ".0"
END,

LOGRADOURO = CASE
    WHEN LOGRADOURO IN ('nat', 'nan', 'null') OR LOGRADOURO IS NULL OR TRIM(LOGRADOURO) = '' THEN 'Sem informação'
    ELSE LOGRADOURO
END,

LATITUDE = CASE
    WHEN LATITUDE IN ('nat', 'nan', 'null') OR LATITUDE IS NULL OR TRIM(LATITUDE) = '' THEN 'Sem informação'
    ELSE LATITUDE
END,

LONGITUDE = CASE
    WHEN LONGITUDE IN ('nat', 'nan', 'null') OR LONGITUDE IS NULL OR TRIM(LONGITUDE) = '' THEN 'Sem informação'
    ELSE LONGITUDE
END,

DESCR_OCORRENCIA_VEICULO = CASE
    WHEN DESCR_OCORRENCIA_VEICULO IN ('nat', 'nan', 'null') OR DESCR_OCORRENCIA_VEICULO IS NULL OR TRIM(DESCR_OCORRENCIA_VEICULO) =
    ELSE DESCR_OCORRENCIA_VEICULO
END,

DESCR_TIPO_VEICULO = CASE
    WHEN DESCR_TIPO_VEICULO IN ('nat', 'nan', 'null') OR DESCR_TIPO_VEICULO IS NULL OR TRIM(DESCR_TIPO_VEICULO) = '' THEN 'Sem infor
    ELSE DESCR_TIPO_VEICULO
END,

DESCR_MARCA_VEICULO = CASE
    WHEN DESCR_MARCA_VEICULO IN ('nat', 'nan', 'null') OR DESCR_MARCA_VEICULO IS NULL OR TRIM(DESCR_MARCA_VEICULO) = '' THEN 'Sem in
    ELSE DESCR_MARCA_VEICULO
END,

ANO_FABRICACAO = CASE
    WHEN ANO_FABRICACAO IN ('nat', 'nan', 'null') OR ANO_FABRICACAO IS NULL OR TRIM(ANO_FABRICACAO) = '' THEN 'Sem informação'
    ELSE ANO_FABRICACAO
END,
```

```
    ANO_MODELO = CASE
        WHEN ANO_MODELO IN ('nat', 'nan', 'null') OR ANO_MODELO IS NULL OR TRIM(ANO_MODELO) = '' THEN 'Sem informação'
        ELSE ANO_MODELO
    END,

    PLACA_VEICULO = CASE
        WHEN PLACA_VEICULO IN ('nat', 'nan', 'null') OR PLACA_VEICULO IS NULL OR TRIM(PLACA_VEICULO) = '' THEN 'Sem informação'
        ELSE PLACA_VEICULO
    END,

    DESC_COR_VEICULO = CASE
        WHEN DESC_COR_VEICULO IN ('nat', 'nan', 'null') OR DESC_COR_VEICULO IS NULL OR TRIM(DESC_COR_VEICULO) = '' THEN 'Sem informação
        ELSE DESC_COR_VEICULO
    END,

    MES = CASE
        WHEN MES IN ('nat', 'nan', 'null') OR MES IS NULL OR TRIM(MES) = '' THEN 'Sem informação'
        ELSE MES
    END,

    ANO = CASE
        WHEN ANO IN ('nat', 'nan', 'null') OR ANO IS NULL OR TRIM(ANO) = '' THEN 'Sem informação'
        ELSE ANO
    END,

    DESCR_CONDUTA = CASE
        WHEN DESCR_CONDUTA IN ('nat', 'nan', 'null') OR DESCR_CONDUTA IS NULL OR TRIM(DESCR_CONDUTA) = '' THEN 'Sem informação'
        ELSE DESCR_CONDUTA
    END
WHERE
    ID_DELEGACIA IS NOT NULL;  -- Ajustar conforme necessário para filtrar as linhas de interesse

-- Confirmar que a tabela foi atualizada com os ajustes
SELECT * FROM silver.veiculos_subtraidos_tratamento LIMIT 10;
```

ID_DELEGACIA	NOME_DELEGACIA	NOME_MUNICIPIO	ANO_BO	NUM_BO	DATA_OCORRENCIA_BO	HORA_OCORRENCIA	DATAHORA_REGI
70344	01º D.P. AMERICANA	AMERICANA	2023	AD4657	NaT	20:30:07	2023-06-05
70344	01º D.P. AMERICANA	AMERICANA	2023	AK1154	2023-01-09	19:29:00	2023-01-09
70344	01º D.P. AMERICANA	AMERICANA	2023	AP4174	2023-01-14	09:15:00	2023-01-14
70344	01º D.P. AMERICANA	GUARULHOS	2023	BK3360	2023-01-30	06:00:00	2023-02-03
70344	01º D.P. AMERICANA	AMERICANA	2023	BZ1873	2023-02-12	Sem informação	2023-02-12

Limpando e ajustando dados

Buscando valores unicos de todas as colunas da tabela, para identificar possiveis sujeiras, e identificar onde realizar os ajustes e padronização

```
%sql
SELECT DISTINCT
    ID_DELEGACIA,
    NOME_DELEGACIA,
    NOME_MUNICIPIO,
    ANO_BO,
    NUM_BO,
    DATA_OCORRENCIA_BO,
    HORA_OCORRENCIA,
    DATAHORA_REGISTRO_BO,
    DATA_COMUNICACAO_BO,
    DATAHORA_IMPRESSAO_BO,
    DESCR_PERIODO,
    FLAG_ATO_INFRACIONAL,
    DESCR_TIPOLOCAL,
    CIDADE,
    BAIRRO,
    CEP,
    LOGRADOURO,
    LATITUDE,
```

```
    LONGITUDE,
    DESCR_OCORRENCIA_VEICULO,
    DESCR_TIPO_VEICULO,
    DESCR_MARCA_VEICULO,
    ANO_FABRICACAO,
    ANO_MODELO,
    PLACA_VEICULO,
    DESC_COR_VEICULO,
    MES,
    ANO,
    DESCR_CONDUTA
FROM silver.veiculos_subtraidos_tratamento;
```

ID_DELEGACIA	NOME_DELEGACIA	NOME_MUNICIPIO	ANO_BO	NUM_BO	DATA_OCORRENCIA_BO	HORA_OCORRENCIA	DATAHORA_
70116	01º D.P. BRAGANÇA PAULISTA	BRAGANCA PAULISTA	2023	IF5516	2023-06-22	14:51:40	2023-06-22
30427	01º D.P. CARAPICUIBA	CARAPICUIBA	2025	BI2906	2025-01-24	Sem informação	2025-01-27
30426	01º D.P. EMBU DAS ARTES	EMBU DAS ARTES	2025	AE8071	2025-01-03	Sem informação	2025-01-06
30109	01º D.P. S.B.C-DR OMAR CASSIM	S.BERNARDO DO CAMPO	2024	CH1356	2024-02-17	Sem informação	2024-02-17
30109	01º D.P. S.B.C-DR	S.BERNARDO DO	2024	HO0508	2024-06-03	Sem informação	2024-06-04

Qualidade dos dados

Identificando pontos para ajustes

- **Coluna CEP**: Identifiquei a presença de valores com ".0" nos dados do CEP. Vamos ajustar.
- **Coluna HORA OCORRENCIA**: Identifiquei a presença de ".0000" na coluna de hora da ocorrência. Vamos ajustar.
- **Coluna BAIRRO**: Identifiquei valores com "-" na coluna Bairro. Vamos ajustar.
- **Coluna PLACA**: Foram encontrados valores como "8377", "2201", "0", "2918624", "xxxxx". Vamos padronizar.

Próximas ações:

- Desmembrar os dados.
- Ajustar os tipos das colunas para garantir consistência.

CEP

```
%sql
UPDATE silver.veiculos_subtraidos_tratamento
SET
    -- Aplicando a transformação na coluna CEP
    CEP = REGEXP_REPLACE(CEP, '\\.0$', '')  -- Remover ".0" do CEP
WHERE
    CEP IS NOT NULL;  -- Apenas atualizar onde o CEP não é nulo

-- Confirmar o ajuste

SELECT CEP
FROM silver.veiculos_subtraidos_tratamento
LIMIT 10;
```

CEP
13474160
13470662
13465360
7178400
13471325
Sem informação
13477234
13465380
13467490

HORA OCORENCIA

```
%sql
UPDATE silver.veiculos_subtraidos_tratamento
SET
    -- Remover ".0000" ou valores similares da coluna HORA_OCORRENCIA
    HORA_OCORRENCIA = REGEXP_REPLACE(CAST(HORA_OCORRENCIA AS STRING), '\\.\\d{4}$', '')
WHERE
    HORA_OCORRENCIA IS NOT NULL;  -- Apenas atualizar onde HORA_OCORRENCIA não é nulo

-- Confirmar que a tabela foi atualizada com o ajuste
SELECT HORA_OCORRENCIA
FROM silver.veiculos_subtraidos_tratamento LIMIT 10;
```

HORA_OCORRENCIA
20:30:07
19:29:00
09:15:00
06:00:00
Sem informação
16:42:00
02:00:00
20:30:00
19:15:25

BAIRRO

```
%sql
UPDATE silver.veiculos_subtraidos_tratamento
SET BAIRRO = CASE
                WHEN BAIRRO = '-' THEN 'Sem informação'
                ELSE BAIRRO
            END;
SELECT BAIRRO
FROM silver.veiculos_subtraidos_tratamento LIMIT 10;
```

BAIRRO
JARDIM BRASIL
MARIO COVAS
VILA REDHER
VILA CARMELA II
PARQUE DAS NACOES
SOBRADO VELHO
JARDIM SANTA SOFIA
VILA REHDER
PARQUE NOVO MUNDO

Placa

```
%sql
UPDATE silver.veiculos_subtraidos_tratamento
SET
    PLACA_VEICULO = CASE
        WHEN PLACA_VEICULO IN ('8377', '2201', '0', '2918624', 'xxxxx')
             OR TRIM(PLACA_VEICULO) = '' OR PLACA_VEICULO IS NULL THEN 'Sem informação'
        ELSE PLACA_VEICULO
    END
WHERE
    PLACA_VEICULO IS NOT NULL;  -- Apenas atualizar onde PLACA_VEICULO não é nulo

-- Confirmar que a tabela foi atualizada com o ajuste
SELECT PLACA_VEICULO
FROM silver.veiculos_subtraidos_tratamento LIMIT 10;
```

PLACA_VEICULO
GKG0880
FBI8E29
DQZ4935
ECN7B21
QQM2A96
EED5183
BWD5J77
EWZ8C16
FTM9C62

Desmembrando dados

```
%sql
UPDATE silver.veiculos_subtraidos_tratamento
SET FLAG_ATO_INFRACIONAL = CASE
    WHEN FLAG_ATO_INFRACIONAL = 'N' THEN 'Não'
    WHEN FLAG_ATO_INFRACIONAL = 'S' THEN 'Sim'
    ELSE FLAG_ATO_INFRACIONAL
END;

SELECT FLAG_ATO_INFRACIONAL
FROM silver.veiculos_subtraidos_tratamento
LIMIT 10;
```

FLAG_ATO_INFRACIONAL
Não
Não
Não
Não
Não
Não
Não
Não
Não

```
%sql
-- Criar uma nova tabela com os ajustes na lógica (sem alterar tipos de dados)
CREATE OR REPLACE TABLE silver.veiculos_subtraidos_tratamento AS
SELECT
    ID_DELEGACIA,
    NOME_DELEGACIA,
    NOME_MUNICIPIO,
    ANO_BO,
    NUM_BO,
    DATA_OCORRENCIA_BO,
    HORA_OCORRENCIA,
    DATAHORA_REGISTRO_BO,
    DATA_COMUNICACAO_BO,
```

```
    DATAHORA_IMPRESSAO_BO,
    DESCR_PERIODO,
    FLAG_ATO_INFRACIONAL,
    DESCR_TIPOLOCAL,
    CIDADE,
    BAIRRO,
    CEP,
    LOGRADOURO,
    LATITUDE,
    LONGITUDE,
    DESCR_OCORRENCIA_VEICULO,
    DESCR_TIPO_VEICULO,

    -- Ajuste na coluna DESCR_MARCA_VEICULO para dividir a marca e adicionar nova coluna Marca
    CASE
        WHEN DESCR_MARCA_VEICULO LIKE '%/%' THEN SPLIT(DESCR_MARCA_VEICULO, '/')[0]
        ELSE 'Sem marca'
    END AS Marca,

    -- Parte após a barra "/" vai para a coluna DESCR_MARCA_VEICULO
    CASE
        WHEN DESCR_MARCA_VEICULO LIKE '%/%' THEN SPLIT(DESCR_MARCA_VEICULO, '/')[1]
        ELSE NULL
    END AS DESCR_MARCA_VEICULO,

    ANO_FABRICACAO,
    ANO_MODELO,
    PLACA_VEICULO,
    DESC_COR_VEICULO,
    MES,
    ANO,
    DESCR_CONDUTA
FROM silver.veiculos_subtraidos_tratamento;

-- Confirmar a tabela criada com os ajustes
SELECT * FROM silver.veiculos_subtraidos_tratamento;
```

ID_DELEGACIA	NOME_DELEGACIA	NOME_MUNICIPIO	ANO_BO	NUM_BO	DATA_OCORRENCIA_BO	HORA_OCORRENCIA	DATAHOF
70344	01º D.P. AMERICANA	AMERICANA	2023	AD4657	NaT	20:30:07	2023-06-0
70344	01º D.P. AMERICANA	AMERICANA	2023	AK1154	2023-01-09	19:29:00	2023-01-0
70344	01º D.P. AMERICANA	AMERICANA	2023	AP4174	2023-01-14	09:15:00	2023-01-1
70344	01º D.P. AMERICANA	GUARULHOS	2023	BK3360	2023-01-30	06:00:00	2023-02-0
70344	01º D.P. AMERICANA	AMERICANA	2023	BZ1873	2023-02-12	Sem informação	2023-02-1
70344	01º D.P. AMERICANA	AMERICANA	2023	DJ4049	2023-03-12	16:42:00	2023-06-0

Aplicando tipos corretos nas colunas

```
%sql
-- Ajustando os tipos de dados da tabela veiculos_subtraidos_tratamento
CREATE OR REPLACE TABLE silver.veiculos_subtraidos_tratamento AS
SELECT
    CAST(ID_DELEGACIA AS STRING) AS ID_DELEGACIA,
    CAST(NOME_DELEGACIA AS STRING) AS NOME_DELEGACIA,
    CAST(NOME_MUNICIPIO AS STRING) AS NOME_MUNICIPIO,
    CAST(ANO_BO AS INT) AS ANO_BO,
    CAST(NUM_BO AS STRING) AS NUM_BO,
    CAST(DATA_OCORRENCIA_BO AS DATE) AS DATA_OCORRENCIA_BO,
    CAST(HORA_OCORRENCIA AS STRING) AS HORA_OCORRENCIA,
    CAST(DATAHORA_REGISTRO_BO AS DATE) AS DATAHORA_REGISTRO_BO,
    CAST(DATA_COMUNICACAO_BO AS DATE) AS DATA_COMUNICACAO_BO,
    CAST(DATAHORA_IMPRESSAO_BO AS DATE) AS DATAHORA_IMPRESSAO_BO,
    CAST(DESCR_PERIODO AS STRING) AS DESCR_PERIODO,
    CAST(FLAG_ATO_INFRACIONAL AS STRING) AS FLAG_ATO_INFRACIONAL,
    CAST(DESCR_TIPOLOCAL AS STRING) AS DESCR_TIPOLOCAL,
    CAST(CIDADE AS STRING) AS CIDADE,
    CAST(BAIRRO AS STRING) AS BAIRRO,
    CAST(CEP AS STRING) AS CEP,
    CAST(LOGRADOURO AS STRING) AS LOGRADOURO,
    CAST(LATITUDE AS STRING) AS LATITUDE,
    CAST(LONGITUDE AS STRING) AS LONGITUDE,
    CAST(DESCR_OCORRENCIA_VEICULO AS STRING) AS DESCR_OCORRENCIA_VEICULO,
```

```
    CAST(DESCR_TIPO_VEICULO AS STRING) AS DESCR_TIPO_VEICULO,
    CAST(Marca AS STRING) AS Marca,
    CAST(DESCR_MARCA_VEICULO AS STRING) AS DESCR_MARCA_VEICULO,
    CAST(ANO_FABRICACAO AS INT) AS ANO_FABRICACAO,
    CAST(ANO_MODELO AS INT) AS ANO_MODELO,
    CAST(PLACA_VEICULO AS STRING) AS PLACA_VEICULO,
    CAST(DESC_COR_VEICULO AS STRING) AS DESC_COR_VEICULO,
    CAST(MES AS INT) AS MES,
    CAST(ANO AS INT) AS ANO,
    CAST(DESCR_CONDUTA AS STRING) AS DESCR_CONDUTA
FROM silver.veiculos_subtraidos_tratamento;

-- Verificando os dados
SELECT * FROM silver.veiculos_subtraidos_tratamento LIMIT 10;
```

ID_DELEGACIA	NOME_DELEGACIA	NOME_MUNICIPIO	ANO_BO	NUM_BO	DATA_OCORRENCIA_BO	HORA_OCORRENCIA	DATAHORA_REGI
70344	01º D.P. AMERICANA	AMERICANA	2023	AD4657	null	20:30:07	2023-06-05
70344	01º D.P. AMERICANA	AMERICANA	2023	AK1154	2023-01-09	19:29:00	2023-01-09
70344	01º D.P. AMERICANA	AMERICANA	2023	AP4174	2023-01-14	09:15:00	2023-01-14
70344	01º D.P. AMERICANA	GUARULHOS	2023	BK3360	2023-01-30	06:00:00	2023-02-03
70344	01º D.P. AMERICANA	AMERICANA	2023	BZ1873	2023-02-12	Sem informação	2023-02-12

```
%sql
CREATE OR REPLACE TABLE silver.veiculos_subtraidos_tratamento AS
SELECT
    initcap(ID_DELEGACIA) AS Id_Delegacia,
    initcap(NOME_DELEGACIA) AS Nome_Delegacia,
    initcap(NOME_MUNICIPIO) AS Nome_Municipio,
    ANO_BO AS Ano_Bo,
    NUM_BO AS Numero_Bo,
    DATA_OCORRENCIA_BO AS Data_Ocorrencia_Bo,
    HORA_OCORRENCIA AS Hora_Ocorrencia,
    DATAHORA_REGISTRO_BO AS Datahora_Registro_Bo,
    DATA_COMUNICACAO_BO AS Data_Comunicacao_Bo,
    DATAHORA_IMPRESSAO_BO AS Datahora_Impressao_Bo,
    initcap(DESCR_PERIODO) AS Descricao_Periodo,
    initcap(FLAG_ATO_INFRACIONAL) AS Flagrante_Ato_Infracional,
    initcap(DESCR_TIPOLOCAL) AS Descricao_Tipo_Local,
    initcap(CIDADE) AS Cidade,
    initcap(BAIRRO) AS Bairro,
    CEP AS Cep,
    initcap(LOGRADOURO) AS Logradouro,
    LATITUDE AS Latitude,
    LONGITUDE AS Longitude,
    initcap(DESCR_OCORRENCIA_VEICULO) AS Descricao_Ocorrencia_Veiculo,
    initcap(DESCR_TIPO_VEICULO) AS Descricao_Tipo_Veiculo,
    initcap(Marca) AS Marca_Veiculo,
    initcap(DESCR_MARCA_VEICULO) AS Descricao_Marca_Veiculo,
    ANO_FABRICACAO AS Ano_Fabricacao,
    ANO_MODELO AS Ano_Modelo,
    PLACA_VEICULO AS Placa_Veiculo,
    initcap(DESC_COR_VEICULO) AS Descricao_Cor_Veiculo,
    MES AS Mes,
    ANO AS Ano,
    initcap(DESCR_CONDUTA) AS Descricao_Conduta
FROM silver.veiculos_subtraidos_tratamento;

-- Verificando os dados
SELECT * FROM silver.veiculos_subtraidos_tratamento LIMIT 10;
```

```
%sql
-- Exemplo: criando uma nova versão ajustada da tabela, se necessário
CREATE OR REPLACE TABLE silver.veiculos_subtraidos_pronta AS
SELECT
    *,
    initcap(CASE Mes
        WHEN '1' THEN 'janeiro'
        WHEN '2' THEN 'fevereiro'
```

```
        WHEN '3' THEN 'março'
        WHEN '4' THEN 'abril'
        WHEN '5' THEN 'maio'
        WHEN '6' THEN 'junho'
        WHEN '7' THEN 'julho'
        WHEN '8' THEN 'agosto'
        WHEN '9' THEN 'setembro'
        WHEN '10' THEN 'outubro'
        WHEN '11' THEN 'novembro'
        WHEN '12' THEN 'dezembro'
        ELSE 'Desconhecido'
    END) AS Mes_Nome
FROM silver.veiculos_subtraidos_tratamento;


-- Verificando os dados
SELECT * FROM silver.veiculos_subtraidos_pronta LIMIT 10;
```

Id_Delegacia	Nome_Delegacia	Nome_Municipio	Ano_Bo	Numero_Bo	Data_Ocorrencia_Bo	Hora_Ocorrencia	Datahora_Registro_Bo	Data_Co
70344	01º D.p. Americana	Americana	2023	AD4657	null	20:30:07	2023-06-05	2023-01-
70344	01º D.p. Americana	Americana	2023	AK1154	2023-01-09	19:29:00	2023-01-09	2023-01-
70344	01º D.p. Americana	Americana	2023	AP4174	2023-01-14	09:15:00	2023-01-14	2023-01-
70344	01º D.p. Americana	Guarulhos	2023	BK3360	2023-01-30	06:00:00	2023-02-03	2023-01-

Base pronta para a camada Gold

A base está pronta para ser promovida à **camada Gold**, onde será utilizada para **análises e geração de insights**.

```
%sql
DROP TABLE IF EXISTS gold.base_veiculos_subtraidos;
```

```
# Carregar a tabela da base silver
df_silver = spark.table("silver.veiculos_subtraidos_pronta")

# Salvar a tabela na base gold
df_silver.write.format("delta").mode("overwrite").saveAsTable("gold.Base_veiculos_subtraidos")
```

```
%sql
SELECT * FROM gold.base_veiculos_subtraidos
```

Id_Delegacia	Nome_Delegacia	Nome_Municipio	Ano_Bo	Numero_Bo	Data_Ocorrencia_Bo	Hora_Ocorrencia	Datahora_Registro_Bo	Data_
70344	01º D.p. Americana	Americana	2023	AD4657	null	20:30:07	2023-06-05	2023-(
70344	01º D.p. Americana	Americana	2023	AK1154	2023-01-09	19:29:00	2023-01-09	2023-(
70344	01º D.p. Americana	Americana	2023	AP4174	2023-01-14	09:15:00	2023-01-14	2023-(
70344	01º D.p. Americana	Guarulhos	2023	BK3360	2023-01-30	06:00:00	2023-02-03	2023-(
70344	01º D.p. Americana	Americana	2023	BZ1873	2023-02-12	Sem informação	2023-02-12	2023-(
70344	01º D.p. Americana	Americana	2023	DJ4049	2023-03-12	16:42:00	2023-06-05	2023-(

Catálogo de dados

Criando um catálogo de dados, com base na tabela Base_veiculos_subtraidos, na camada Gold.

```
# Carrega a tabela do schema gold
df = spark.table("gold.base_veiculos_subtraidos")
schema = df.schema

# Descrições personalizadas
descricao_customizada = {
    "Id_Delegacia": "Informação sobre id delegacia que tratrou a ocorrencia",
    "Nome_Delegacia": "Informação sobre nome delegacia",
    "Nome_Municipio": "Informação sobre nome municipio da delegacia",
    "Ano_Bo": "Informação sobre ano bo",
    "Numero_Bo": "Informação sobre numero bo",
    "Data_Ocorrencia_Bo": "Informação sobre data ocorrencia bo",
    "Hora_Ocorrencia": "Informação sobre hora ocorrencia",
    "Datahora_Registro_Bo": "Informação sobre datahora registro bo",
    "Data_Comunicacao_Bo": "Informação sobre data comunicacao bo",
    "Datahora_Impressao_Bo": "Informação sobre datahora impressao bo",
    "Descricao_Periodo": "Informação sobre descricao periodo",
    "Flagrante_Ato_Infracional": "Informação sobre flagrante ato infracional",
    "Descricao_Tipo_Local": "Informação sobre descricao tipo local",
    "Cidade": "Informação sobre cidade",
    "Bairro": "Informação sobre bairro onde aconteceu o evento",
    "Cep": "Informação sobre cep",
    "Logradouro": "Informação sobre logradouro",
    "Latitude": "Informação sobre latitude",
    "Longitude": "Informação sobre longitude",
    "Descricao_Ocorrencia_Veiculo": "Informação sobre descricao ocorrencia veiculo",
    "Descricao_Tipo_Veiculo": "Informação sobre descricao tipo veiculo",
    "Marca_Veiculo": "Informação sobre marca veiculo",
    "Descricao_Marca_Veiculo": "Informação sobre descricao marca veiculo",
    "Ano_Fabricacao": "Informação sobre ano fabricacao",
    "Ano_Modelo": "Informação sobre ano modelo",
    "Placa_Veiculo": "Informação sobre placa veiculo",
    "Descricao_Cor_Veiculo": "Informação sobre descricao cor veiculo",
    "Mes": "Informação sobre mes",
    "Ano": "Informação sobre ano",
    "Descricao_Conduta": "Informação sobre descricao conduta",
    "Mes_Nome": "Informação sobre mes nome"
}

# Coleta valores min/max para colunas numéricas ou de data
min_cols = []
max_cols = []

for field in schema:
    if isinstance(field.dataType, (NumericType, DateType, TimestampType)):
        min_cols.append(min(col(field.name)).alias(f"{field.name}_min"))
        max_cols.append(max(col(field.name)).alias(f"{field.name}_max"))

minmax_values = df.select(min_cols + max_cols).collect()[0].asDict()

# Monta catálogo
catalogo = []
for field in schema:
    nome = field.name
    tipo = field.dataType.simpleString()
    minimo = minmax_values.get(f"{nome}_min", None)
    maximo = minmax_values.get(f"{nome}_max", None)
    variacao = f"{minimo} - {maximo}" if minimo is not None and maximo is not None else "-"
    descricao = descricao_customizada.get(nome, f"Informação sobre {nome.replace('_', ' ').lower()}")

    catalogo.append((nome, tipo, variacao, descricao))

# Cria um DataFrame Spark com o catálogo
catalogo_spark_df = spark.createDataFrame(
    catalogo,
    ["coluna", "tipo", "variacao", "descricao"]
)

# Salva como tabela no banco gold
catalogo_spark_df.write.mode("overwrite").saveAsTable("gold.catalogo_base_veiculos_subtraidos")
```

coluna	tipo	variacao	descricao
Id_Delegacia	string	-	Informação sobre id delegacia que tratrou a ocorrencia
Nome_Delegacia	string	-	Informação sobre nome delegacia
Nome_Municipio	string	-	Informação sobre nome municipio da delegacia
Ano_Bo	int	2021 - 2025	Informação sobre ano bo
Numero_Bo	string	-	Informação sobre numero bo
Data_Ocorrencia_Bo	date	2003-07-01 - 2025-03-07	Informação sobre data ocorrencia bo
Hora_Ocorrencia	string	-	Informação sobre hora ocorrencia
Datahora_Registro_Bo	date	2023-01-01 - 2025-02-28	Informação sobre datahora registro bo
Data_Comunicacao_Bo	date	2012-09-03 - 2025-03-07	Informação sobre data comunicacao bo

Visualizando catálogo

```
%sql
SELECT * FROM gold.catalogo_base_veiculos_subtraidos
```

coluna	tipo	variacao	descricao
Hora_Ocorrencia	string	-	Informação sobre hora ocorrencia
Datahora_Registro_Bo	date	2023-01-01 - 2025-02-28	Informação sobre datahora registro bo
Data_Comunicacao_Bo	date	2012-09-03 - 2025-03-07	Informação sobre data comunicacao bo
Datahora_Impressao_Bo	date	2023-01-01 - 2025-03-10	Informação sobre datahora impressao bo
Descricao_Periodo	string	-	Informação sobre descricao periodo
Flagrante_Ato_Infracional	string	-	Informação sobre flagrante ato infracional
Longitude	string	-	Informação sobre longitude
Descricao_Ocorrencia_Veiculo	string	-	Informação sobre descricao ocorrencia veiculo
Descricao_Tipo_Veiculo	string	-	Informação sobre descricao tipo veiculo

Análises

1) Quais são as Cidades do Estado de São Paulo, com maior incidência de roubos e furtos de veículos?

```
%sql
SELECT
  Nome_Municipio,
  COUNT(*) AS total_ocorrencias
FROM gold.base_veiculos_subtraidos
GROUP BY Nome_Municipio
ORDER BY total_ocorrencias DESC
LIMIT 10;
```

Nome_Municipio	total_ocorrencias
S.paulo	136720
Campinas	13774
S.andre	12792
Guarulhos	11878
S.bernardo Do Campo	8822
Osasco	8774
Maua	5163
Diadema	5141
Sorocaba	5099
Ribeirao Preto	4726

Databricks visualization. Run in Databricks to view.

O estado de São Paulo apresenta, disparado, a maior incidência de ocorrências

2) Em quais horários e dias da semana ocorrem mais roubos e furtos de veículos?

```
%sql
SELECT
  CASE dayofweek(Data_Ocorrencia_Bo)
    WHEN 1 THEN 'Domingo'
    WHEN 2 THEN 'Segunda-feira'
    WHEN 3 THEN 'Terça-feira'
    WHEN 4 THEN 'Quarta-feira'
    WHEN 5 THEN 'Quinta-feira'
    WHEN 6 THEN 'Sexta-feira'
    WHEN 7 THEN 'Sábado'
  END AS Dia_Semana,
  COUNT(*) AS Total_Ocorrencias,
  dayofweek(Data_Ocorrencia_Bo) AS Dia_Ordem
FROM gold.base_veiculos_subtraidos
WHERE Data_Ocorrencia_Bo IS NOT NULL
GROUP BY
  CASE dayofweek(Data_Ocorrencia_Bo)
    WHEN 1 THEN 'Domingo'
    WHEN 2 THEN 'Segunda-feira'
    WHEN 3 THEN 'Terça-feira'
    WHEN 4 THEN 'Quarta-feira'
    WHEN 5 THEN 'Quinta-feira'
    WHEN 6 THEN 'Sexta-feira'
    WHEN 7 THEN 'Sábado'
  END,
  dayofweek(Data_Ocorrencia_Bo)
ORDER BY Dia_Ordem;
```

Dia_Semana	Total_Ocorrencias	Dia_Ordem
Domingo	34770	1
Segunda-feira	42287	2
Terça-feira	54429	3
Quarta-feira	56571	4
Quinta-feira	55911	5
Sexta-feira	51850	6
Sábado	38813	7

Databricks visualization. Run in Databricks to view.

A quarta-feira é o dia da semana com maior número de roubos de veículos.

3) Quais são as três situações mais comuns nos eventos de roubo e furto de veículos?

```
%sql
SELECT
  Flagrante_Ato_Infracional,
  Descricao_Tipo_Local,
  Cidade,
  Hora_Ocorrencia,
  FORMAT_NUMBER(COUNT(*), 0) AS Total_Ocorrencias
FROM gold.base_veiculos_subtraidos
WHERE Flagrante_Ato_Infracional IS NOT NULL
  AND Descricao_Tipo_Local IS NOT NULL
  AND Cidade IS NOT NULL
  AND Hora_Ocorrencia IS NOT NULL
GROUP BY
  Flagrante_Ato_Infracional,
  Descricao_Tipo_Local,
  Cidade,
  Hora_Ocorrencia
ORDER BY CAST(REPLACE(FORMAT_NUMBER(COUNT(*), 0), ',', '') AS INT) DESC
LIMIT 3;
```

Flagrante_Ato_Infracional	Descricao_Tipo_Local	Cidade	Hora_Ocorrencia	Total_Ocorrencias
Não	Via Pública	S.paulo	Sem informação	8,179
Não	Via Pública	S.paulo	20:00:00	2,383
Não	Via Pública	S.paulo	21:00:00	2,108

As situações mais comuns ocorrem à noite, em via pública, na cidade de São Paulo.

4) Como os índices de roubo e furto de veículos variam ao longo do ano?

```
%sql
SELECT
  Ano,
  initcap(Mes_Nome) AS Mes,
  COUNT(*) AS Total_Ocorrencias
FROM gold.base_veiculos_subtraidos
WHERE
  Ano IS NOT NULL
  AND Mes_Nome IS NOT NULL
  AND lower(trim(Mes_Nome)) != '2 - desconhecido'
  AND lower(trim(Ano)) NOT IN ('2 - desconhecido', 'desconhecido')
GROUP BY Ano, Mes_Nome
ORDER BY
  Ano,
  CASE
    WHEN lower(trim(Mes_Nome)) = 'janeiro' THEN 1
    WHEN lower(trim(Mes_Nome)) = 'fevereiro' THEN 2
    WHEN lower(trim(Mes_Nome)) = 'março' THEN 3
    WHEN lower(trim(Mes_Nome)) = 'abril' THEN 4
    WHEN lower(trim(Mes_Nome)) = 'maio' THEN 5
    WHEN lower(trim(Mes_Nome)) = 'junho' THEN 6
    WHEN lower(trim(Mes_Nome)) = 'julho' THEN 7
    WHEN lower(trim(Mes_Nome)) = 'agosto' THEN 8
    WHEN lower(trim(Mes_Nome)) = 'setembro' THEN 9
    WHEN lower(trim(Mes_Nome)) = 'outubro' THEN 10
    WHEN lower(trim(Mes_Nome)) = 'novembro' THEN 11
    WHEN lower(trim(Mes_Nome)) = 'dezembro' THEN 12
  END;
```

Ano	Mes	Total_Ocorrencias
2	Desconhecido	1
2023	Janeiro	14089
2023	Fevereiro	12912
2023	Março	15426
2023	Abril	13267
2023	Maio	14334
2023	Junho	13191
2023	Julho	13413
2023	Agosto	13599
2023	Setembro	12883

Databricks visualization. Run in Databricks to view.

Os roubos variam consideravelmente, mas em 2023 houve uma curva acentuada nos quatro primeiros meses do ano. Já 2025 se inicia com números próximos aos de 2024.

5) Quais são as marcas e modelos de veículos mais roubados ou furtados?

```
%sql
SELECT
  Marca_Veiculo AS Marca,
  Descricao_Marca_Veiculo AS Modelo,
  format_number(COUNT(*), 0) AS Total_Ocorrencias
FROM gold.base_veiculos_subtraidos
WHERE
  Marca_Veiculo IS NOT NULL
  AND Descricao_Marca_Veiculo IS NOT NULL
  AND upper(trim(Descricao_Marca_Veiculo)) NOT IN ('DESCONHECIDO', 'NAO INFORMADO', 'IGNORADO')
GROUP BY
  Marca_Veiculo,
  Descricao_Marca_Veiculo
ORDER BY
  COUNT(*) DESC
LIMIT 20
```

Marca	Modelo	Total_Ocorrencias
Honda	Cg 160 Fan	15,803
Honda	Cg 160 Titan	8,015
Honda	Cg 160 Start	5,391
Yamaha	Fz25 Fazer	4,572
Gm	Corsa Wind	4,180
Hyundai	Hb20 1.0m Comfor	2,844
Fiat	Mobi Like	2,829
Yamaha	Xtz250 Lander	2,803
Fiat	Argo Drive 1.0	2,637
Honda	Pcx 150	2,468

Databricks visualization. Run in Databricks to view.

A marca de veículo mais furtada é a Honda, com a CG disparada na frente em número de ocorrências.

6) Qual a diferença entre a taxa de recuperação de veículos roubados e furtados?

Dados que compõem a base, não o suficiente para responder

7) Quais veículos apresentam menor risco de roubo ou furto em cada região?

```
%sql
WITH ocorrencias_por_veiculo_regiao AS (
  SELECT
    Nome_Municipio AS Regiao,
    Marca_Veiculo AS Marca,
    Descricao_Marca_Veiculo AS Modelo,
    COUNT(*) AS Total_Ocorrencias
  FROM gold.base_veiculos_subtraidos
  WHERE
    Marca_Veiculo IS NOT NULL
    AND Descricao_Marca_Veiculo IS NOT NULL
    AND Nome_Municipio IS NOT NULL
    AND upper(trim(Descricao_Marca_Veiculo)) NOT IN ('DESCONHECIDO', 'NAO INFORMADO', 'IGNORADO')
  GROUP BY Nome_Municipio, Marca_Veiculo, Descricao_Marca_Veiculo
),
ordenado AS (
  SELECT *,
         ROW_NUMBER() OVER (PARTITION BY Regiao ORDER BY Total_Ocorrencias ASC) AS rn
  FROM ocorrencias_por_veiculo_regiao
)
SELECT
  Regiao,
  Marca,
  Modelo,
  Total_Ocorrencias
FROM ordenado
WHERE rn <= 10
ORDER BY Regiao, Total_Ocorrencias;
```

Regiao	Marca	Modelo	Total_Ocorrencias
Adamantina	Vw	Voyage	1
Adamantina	Honda	Cg 160 Titan	1
Adamantina	Yamaha	Factor Ybr125 Ed	1
Adamantina	Gm	Kadett Sl Efi	1
Adamantina	Honda	Cg 125 Fan Ks	1
Adamantina	I	Shineray Mvk Xy110 2	1
Adamantina	Fiat	Tempra	1
Adamantina	Honda	Biz 110i	1
Adamantina	Honda	C100 Biz Es	1

```
%sql
SELECT
  CAST(Latitude AS DOUBLE) AS Latitude,
  CAST(Longitude AS DOUBLE) AS Longitude,
  upper(trim(Descricao_Ocorrencia_Veiculo)) AS Tipo_Ocorrencia
FROM gold.base_veiculos_subtraidos
```

```
WHERE
  Latitude IS NOT NULL
  AND Longitude IS NOT NULL
  AND Descricao_Ocorrencia_Veiculo IS NOT NULL
  AND upper(trim(Descricao_Ocorrencia_Veiculo)) IN ('ROUBADO', 'FURTADO')
LIMIT 1000
```

Latitude	Longitude	Tipo_Ocorrencia
-23.6271625	-46.670291	FURTADO
-22.7404707929999	-47.351045569	FURTADO
-22.7643792	-47.3536704	FURTADO
-22.7499199379999	-47.337274613	FURTADO
-22.726849225	-47.3096491219999	FURTADO
-22.7393086819999	-47.322627269	FURTADO
-22.750629744	-47.3187448379999	FURTADO
-22.7668305039999	-47.340772823	FURTADO
-22.7349988	-47.3308262	FURTADO
-22.7506861	-47.2605257	FURTADO

Databricks visualization. Run in Databricks to view.

Na região de Adamantina, o Voyage e a CG Titan seguem entre os mais visados. Já em Adolfo, a Parati lidera, embora com um número menor de ocorrências. Em Aguaí, a marca Mercedes também aparece entre os veículos com menos registros de furto

8) Em quais regiões o risco de ter um veículo subtraído é menor?

```
%sql
SELECT
  Nome_Municipio AS Cidade,
  Bairro AS Bairro,
  format_number(COUNT(*), 0) AS Total_Ocorrencias
FROM gold.base_veiculos_subtraidos
WHERE
  upper(trim(Descricao_Ocorrencia_Veiculo)) IN ('ROUBADO', 'FURTADO')
  AND Nome_Municipio IS NOT NULL
  AND Bairro IS NOT NULL
GROUP BY Nome_Municipio, Bairro
ORDER BY COUNT(*) ASC
LIMIT 10
```

Cidade	Bairro	Total_Ocorrencias
Itapevi	Jardim Vitápolis	1
Poa	Vila Sao Joao	1
Conchas	Rural	1
Porto Feliz	Jd. Vante	1