

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Irlen Vinicius Teixeira Menezes

ANÁLISE EXPLORATÓRIA DOS ACIDENTES RODOVIARIOS BRASILEIRO

Belo Horizonte
2021

Irlen Vinicius Teixeira Menezes

ANÁLISE EXPLORATÓRIA DOS ACIDENTES RODOVIARIOS BRASILEIRO

Trabalho de Conclusão de Curso apresentado
ao Curso de Especialização em Ciência de
Dados e Big Data como requisito parcial à
obtenção do título de especialista.

Belo Horizonte
2021

LISTAS DE TABELAS

Tabela 1- Fluxo de Ferramentas Utilizadas no Projeto.....8

Tabela 2 - Dados do Dataset.....9

Sumário

1. Introdução.....	5
1.1. Contextualização.....	5
1.2. O problema proposto.....	6
1.3. Perguntas a Responder.....	7
1.4. Fluxo do Projeto.....	7
2. Coleta de Dados.....	8
3. Processamento/Tratamento de Dados.....	11
3.1 Estrutura de Pastas do Projeto.....	11
3.2 Verificação e Ajustes de Codificação de Caracteres.....	11
3.2 Importando os Dados CSV para o Jupyter Lab.....	12
3.3 Analisando os Dados importados para o Jupyter Lab.....	14
3.4 Limpeza e tratamento dos Dados.....	15
4. Explorando os dados no Power BI.....	31
4.1. Criando a interface.....	33
4.2. Primeira tela do Dashboard.....	40
4.3. Segunda tela do Dashaboard.....	42
4.4 Terceira tela do Dashboard.....	44
4.5. Quarta tela do Dashboard.....	45
4.5. Quinta tela do Dashboard.....	46
4.6. Sexta tela do Dashboard.....	46
5. Resultados.....	47
6. Links.....	52

1. Introdução

1.1. Contextualização

Segundo os dados atualizados em 2019 da Organização Pan-Americana da Saúde, é estimado que 1,3 milhão de pessoas no mundo morrem vítimas dos acidentes de trânsito (AT) a cada ano e que 90% dessas mortes ocorrem em países de rendas baixas a média. Pesquisas apontam que os acidentes de trânsito custam à maioria dos países 3% de seu produto interno bruto (PIB);

Em 2016, segundo dados do Sistema de Informações sobre Mortalidade (SIM) do Ministério da Saúde, disponibilizados no site do Departamento de Informática do Sistema Único de Saúde (Datasus), houve no país um total de 38.265 mortes provocadas por acidentes de transporte terrestre. Entre esses óbitos, 12.036 (31,5%) eram motociclistas, 8.899 (23,2%) ocupantes de automóveis e 6.158 (16,1%) pedestres.

Tendo em vista os acidentes que ocorrem, conseqüentemente várias pessoas acabam morrendo ou ficam feridas. Isso acaba reacendendo discussões que destacam a segurança das rodovias, dos carros e até onde o fator humano é capaz de influenciar no desenrolar de um acidente.

Esse trabalho visa realizar uma análise de dados em busca de informações mais superficiais sobre o assunto, mas também não deixando de responder o questionamento acima.

Para responder a essa e outras perguntas irei dividir o meu trabalho em duas partes. A parte da coleta e do processamento de dados será feita no próprio Jupyter Notebook. Já o Power BI ficará com a parte da exploração dos dados e visualização final.

1.2. O problema proposto

A Polícia Rodoviária Federal – PRF atende cerca de 70 mil quilômetros de rodovias federais e está distribuída em todo o território nacional, combatendo a criminalidade, prestando auxílio ao cidadão, fiscalizando, autuando e atendendo acidentes. O registro de acidentes é realizado através do sistema BR-Brasil, que coleta informações referentes aos envolvidos (identificação, estado físico, se era passageiro, condutor etc.), ao local, aos veículos, à dinâmica do acidente e etc.

Apesar da bela atuação da PRF os dados são disponibilizados de forma bruta, com isso, acaba inviabilizando o entendimento de pessoas comuns que não tem conhecimento em análise de dados para entender de forma simples e visual dados como quantas mortes tem por mês na rodovia de sua cidade ou quais tipos de acidentes, quais trechos são mais perigosos, qual horário tem maior incidência de acidentes, qual previsão do tempo é mais favorável para acidentes entre outras análises possíveis dentro desse conjunto de dados.

Um índice de dados abertos das cidades brasileiras mostrou que os dados fornecidos à população estão incompletos ou não são transparentes. O gráfico abaixo foi elaborado pela Diretoria de Análise de Políticas Públicas da Fundação Getúlio Vargas, em parceria com a Open Knowledge Brasil.

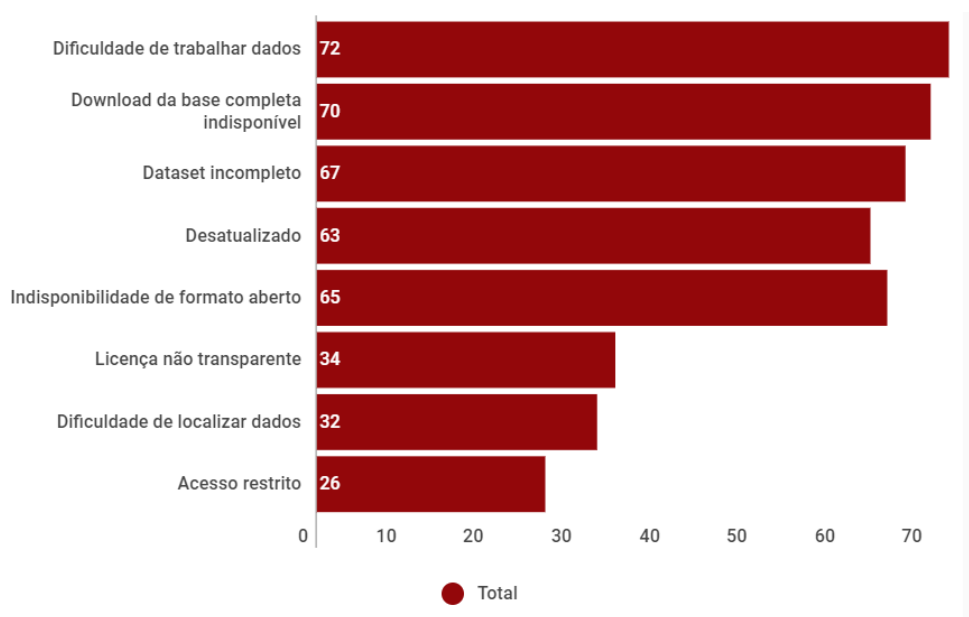


FIGURA 1 - Índice de dados abertos para cidades. Fonte: FGV DAPP

1.3. Perguntas a Responder

Separei algumas perguntas que considero importante para este trabalho:

- Qual a quantidade de ocorrências e fatalidade ao longo do último triênio?
- Quais as principais causas de acidentes e as causas mais letais?
- Quais os principais tipos de acidentes e os acidentes mais letais?
- Quais dias ocorrem os acidentes e quais dias são mais letais?
- Quais as BRs tem mais acidentes e são mais letais?
- Quais horários ocorrem os acidentes e em quais horários mais letais?
- Qual região, estado e município tem mais acidentes e quais são as mais letais?
- Quais são os principais influenciadores dos acidentes sem vítimas?
- Quais são os principais influenciadores dos acidentes com vítimas fatais?
- Quais são os principais influenciadores dos acidentes com vítimas feridas?
- Quais as condições meteorológicas influenciam em mais acidentes?
- Baseado em ocorrências onde o fator humano foi o causador, quais foram os erros mais cometidos?
- Qual a previsão de acidentes para os próximos meses?

1.4. Fluxo do Projeto

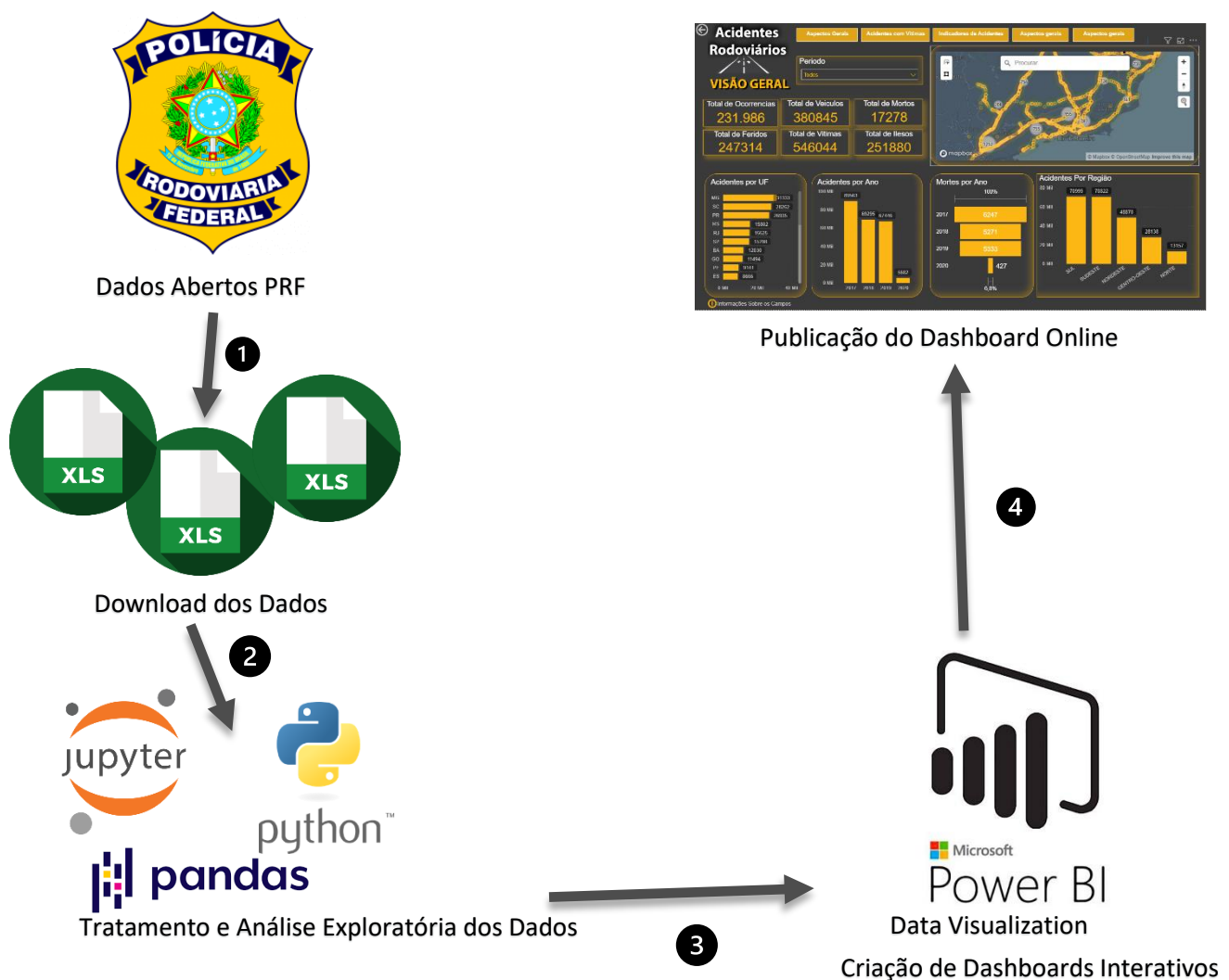


Figura 2- Fluxo do Projeto

ID	NOME	Descrição	Ferramenta
1	Dados Abertos PRF	Dados Abertos são dados publicados em um formato legível por máquina e sem restrição de licenças, patentes ou mecanismos de controle, de modo a estarem livremente disponíveis para serem utilizados e redistribuídos à vontade.	Google Chrome - Versão 83.0.4103.106 (Versão oficial) 64 bits
2	Download dos Dados	Download manual dos arquivos em formato Zip	Google Chrome - Versão 83.0.4103.106 (Versão oficial) 64 bits
3	Tratamento e Análise Exploratória dos Dados	Processo construído em identificar, tratar e limpar os dados faltantes ou errados contido no conjunto de dados, para que possa entrar em fase de análise exploratória onde será possível descobrir diversos insights	Python versão 3.7.4 JupyterLab versão 1.1.4 Pandas versão 0.25.1 Sublime Text3 versão 3.2.2
4	Criação do Dashboard	Levar os Dados tratados para o Power BI e criar relatórios interativos e visuais através de Painéis com Dashboards de todos os dados	Power BI Versão: 2.88.1385.0 64-bit (December 2020)

Tabela 1- Fluxo de Ferramentas Utilizadas no Projeto

2. Coleta de Dados

Foram obtidos 4 conjuntos de dados dos Acidentes Rodoviários por todas as causas e tipos de acidentes referente aos anos de 2017, 2018, 2019 e 2020. Os dados apresentados nesse trabalho foram baixados na data 01/08/2020, data em que esse trabalho foi redigido, sendo assim o dataset de 2020 tem os dados parciais referente a data de criação desse trabalho.

Com origem na base de dados da PRF – Polícia Rodoviária Federal, os dados aqui são referentes a todos os acidentes registrados pela PRF:

- [Link para Download](#)
- Formato do Arquivo: Arquivo de Valores Separados por Vírgulas do Microsoft Excel (.CSV)
- Quantidade de Arquivos: 4 (quatro)

NOME DA VARIÁVEL	DESCRIÇÃO
<i>id</i>	Variável com valores numéricos, representando o identificador do acidente.
<i>data_inversa</i>	Data da ocorrência no formato dd/mm/aaaa.
<i>dia_semana</i>	Dia da semana da ocorrência. Ex.: Segunda, Terça, etc.
<i>horario</i>	Horário da ocorrência no formato hh:mm:ss.
<i>uf</i>	Unidade da Federação. Ex.: MG, PE, DF, etc.
<i>br</i>	Variável com valores numéricos, representando o identificador da BR do acidente.
<i>km</i>	Identificação do quilômetro onde ocorreu o acidente, com valor mínimo de 0,1 km e com a casa decimal separada por ponto.
<i>municipio</i>	Nome do município de ocorrência do acidente
<i>causa_acidente</i>	Causa presumível do acidente, baseada nos vestígios, indícios e provas colhidas no local do acidente.
<i>ordem_tipo_acidente</i>	Valor numérico que identifica a sequência dos eventos sucessivos que ocorreram no acidente.
<i>tipo_acidente</i>	Identificação do tipo de acidente. Ex.: Colisão frontal, Saída de pista, etc.
<i>classificação_acidente</i>	Classificação quanto à gravidade do acidente: Sem Vítimas, Com Vítimas Feridas, Com Vítimas Fatais e Ignorado.
<i>fase_dia</i>	Fase do dia no momento do acidente. Ex. Amanhecer, Pleno dia, etc.
<i>sentido_via</i>	Sentido da via considerando o ponto de colisão: Crescente e decrescente.
<i>condição_meteorologica</i>	Condição meteorológica no momento do acidente: Céu claro, chuva, vento, etc.

<i>tipo_pista</i>	Tipo da pista considerando a quantidade de faixas: Dupla, simples ou múltipla.
<i>tracado_via</i>	Descrição do traçado da via.
<i>uso_solo</i>	Descrição sobre as características do local do acidente: Urbano=Sim; Rural=Não.
<i>veiculos</i>	Todos os veículos envolvidos em ocorrências.
<i>ilesos</i>	Valor binário que identifica se o envolvido foi classificado como ileso.
<i>feridos_leves</i>	Valor binário que identifica se o envolvido foi classificado como ferido leve.
<i>feridos_graves</i>	Valor binário que identifica se o envolvido foi classificado como ferido grave.
<i>Ignorados</i>	Total de pessoas envolvidos em ocorrências e que não soube o estado físico.u
<i>mortos</i>	Valor binário que identifica se o envolvido foi classificado como morto.
<i>latitude</i>	Latitude do local do acidente em formato geodésico decimal.
<i>longitude</i>	Longitude do local do acidente em formato geodésico decimal.

Tabela 2- Dados do Dataset

3. Processamento/Tratamento de Dados

Nessa etapa realizei o processamento e tratamento prévio dos dados. Os dados da PRF que utilizei, estão agrupados por pessoa e com todas as causas e tipos de acidentes a partir de 2017.

3.1 Estrutura de Pastas do Projeto

Para criar um padrão no projeto eu criei uma pasta principal chamada TCC 2020 e criei duas subpastas, uma chamada download onde irá armazenar o arquivo original baixado direto da PRF e a outra chamada dataset que irá armazenar o arquivo tratado e modificado que será utilizado na criação do Dashboard no Power BI.



Figura 3 – Estrutura de Pasta usada no projeto

3.2 Verificação e Ajustes de Codificação de Caracteres

Nesta etapa será verificado qual é o tipo de codificação de caracteres do arquivo baixado no site da PRF, como estou trabalhando com o Python 3, por padrão o encoding é UTF-8. Para evitar erros e retrabalho, primeiro vou saber qual a versão do python estou rodando no Jupyter Lab, para descobrir basta executar o comando:

!python --version

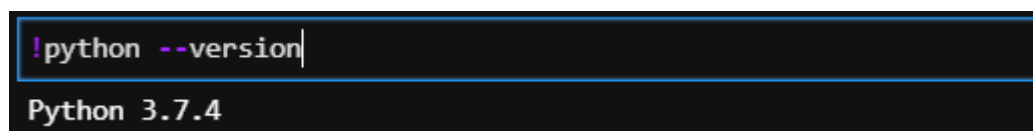


Figura 4 – Descobrindo a Versão do Python

Existem diversas formas de converter um arquivo para UTF-8, nesse projeto eu vou utilizar a forma que considero mais rápida, simples e eficiente. Abra os Arquivos CSV com o Programa Sublime Text 3, clique em File > Save with Encoding > UTF-8

e pronto o arquivo já estará no encoding padrão para ser usado no Python e Power BI.

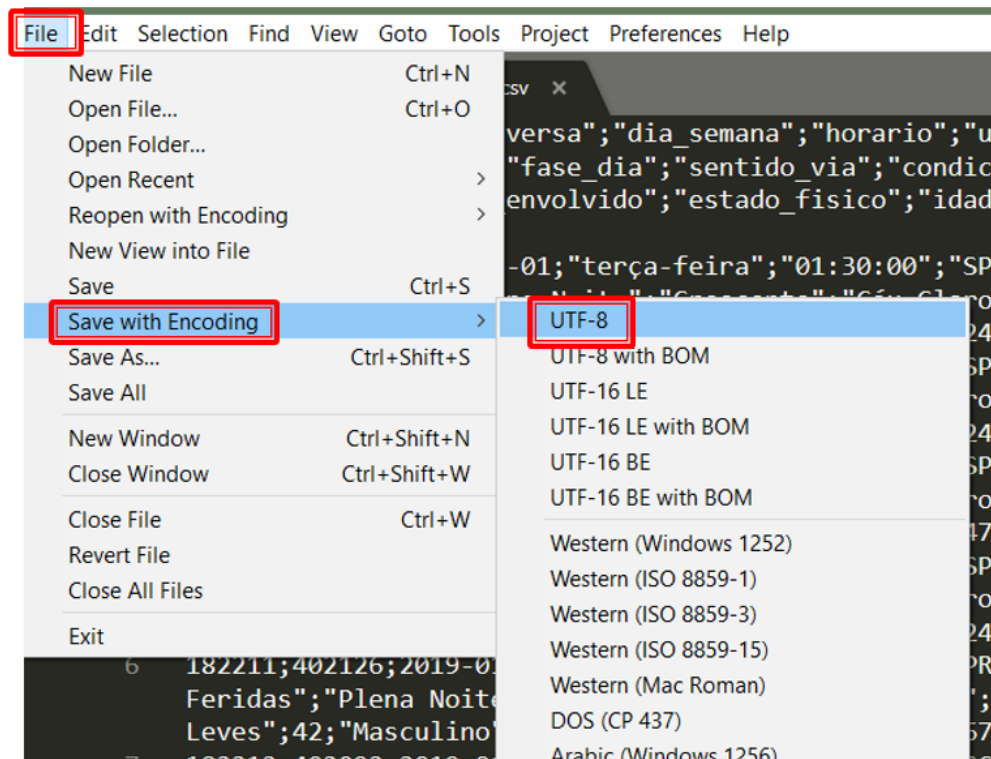
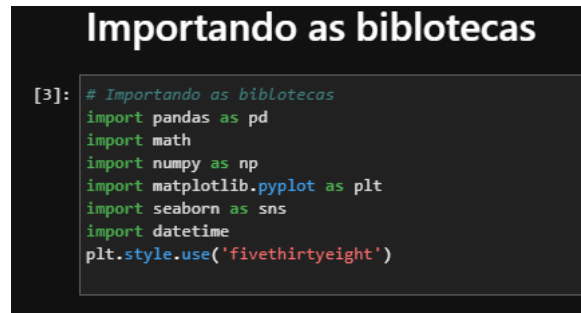


Figura 5 – Salvando com encoding para UTF-8

Esse processo de Encoding poderia ser feito de diversas maneiras como pelo próprio Jupyter Notebook usando a linguagem Python, com Power BI ou até mesmo abrindo no Excel e mudando o Encoding, porem todas essas formas são mais complexas do que o método utilizado acima.

3.2 Importando os Dados CSV para o Jupyter Lab

Após converter os 4 arquivos CSV baixados no site da PRF para a codificação de caracteres UTF-8, agora iremos importar as bibliotecas principais para abrir e trabalhar esses arquivos no Jupyter Lab.

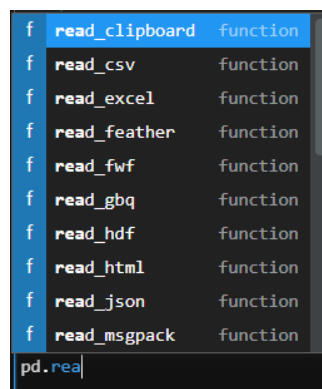


```
[3]: # Importando as bibliotecas
import pandas as pd
import math
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
plt.style.use('fivethirtyeight')
```

Figura 6 – Importando as bibliotecas

Neste projeto eu faço o uso das bibliotecas: Pandas, Math, Numpy, Matplotlib, Searborn e Datatime e wordcloud.

Após importar as bibliotecas no Jupyter Lab, em uma nova linha, eu utilizarei a função Read_CSV que é disponibilizada na biblioteca Pandas. Na figura acima eu crio um apelido chamado pd para função pandas, sempre que for necessário usar qualquer recurso do pandas, basta digitar o apelido e depois digitar o caractere “ponto”, se apertar a tecla TAB o jupyter habilita o IntelliSense, que é uma ajuda de preenchimento de código que oferece diversas funcionalidades: Informações do Parâmetro, Informações Rápidas e Completar Palavra.



f	read_clipboard	function
f	read_csv	function
f	read_excel	function
f	read_feather	function
f	read_fwf	function
f	read_gbq	function
f	read_hdf	function
f	read_html	function
f	read_json	function
f	read_msgpack	function

pd.read

Figura 7 – usando o IntelliSense no Jupyter Lab

Nesta etapa eu crio quatro variáveis identificando com nomenclatura df de dataframe e o respectivo ano de cada um dos arquivos CSV. Cada uma dessas variáveis vão receber os dados do arquivo CSV, na figura abaixo eu importo os quatro arquivos CSV utilizando a função read_csv do tipo (parsing) para conversão dos arquivos em Dataframes, eu utilizo o método SEP que serve para indicar qual o separador desse CSV no caso desses arquivos é o caractere ponto e virgula “;”.

```
''' Importando os três arquivos CSV utilizando a função read_csv do tipo
(parsing) para conversão dos arquivos em Dataframes '''

df2020 = pd.read_csv('acidentes2020_todas_causas_tipos.csv', sep=';')
df2019 = pd.read_csv('acidentes2019_todas_causas_tipos.csv', sep=';')
df2018 = pd.read_csv('acidentes2018_todas_causas_tipos.csv', sep=';')|
df2017 = pd.read_csv('acidentes2017_todas_causas_tipos.csv', sep=';')
```

Figura 8 – Criando as variáveis para receber os arquivos em CSV

3.3 Analisando os Dados importados para o Jupyter Lab

Concatenando todos os arquivos em um só

```
#Usando a Função para Concatenar os DF
DadosAcidentes = pd.concat([df2017, df2018, df2019, df2020], sort=False, ignore_index=True)
```

Figura 9 – Concatenando os dataframes em um só

Após ter criado uma variável para cada um dos quatros arquivos CSV, agora o objetivo é juntar todos os dataframes em apenas um dataframe. Para realizar esse objetivo eu usei a função concat do pandas, para mesclar

- **df2017** - Esta variável ela recebe os dados de 2017;
- **df2018** - Esta variável recebe os dados de 2017;
- **df2019** - Esta variável ela recebe os dados de 2019;
- **df2020** - Esta variável ela recebe os dados de 2019;
- **DadosAcidentes** – Esta variável recebe a concatenação de todos os dados por anos em um único dataframe;

Existem diversas maneiras de juntar os arquivos em CSV em um só arquivo, neste caso eu resolvi fazer diretamente no Jupyter Lab usando o Python, porém o power BI oferece um recurso que une automaticamente todos os arquivos em um só, com apenas alguns cliques.

Agora iremos verificar o shape de cada dataset criado acima, a função shape() usada na figura abaixo, ela retornar o número de linhas e colunas de uma matriz.

```
print('Todas as Linhas X Colunas de 2017', df2017.shape)
print('Todos as Linhas X Colunas de 2018', df2018.shape)
print('Todos as Linhas X Colunas de 2019', df2019.shape)
print('Todos as Linhas X Colunas de 2020', df2020.shape)

print('SOMA DE TODOS OS DF Linhas X Colunas 2017+2018+2019+2020 = ', DadosAcidentes.shape)
```

Figura 10 – Verificação de quantas Linhas x Colunas de cada Dataframe

O resultado do código programado acima, usando a função shape, vai retornar os valores da figura abaixo. Podemos observar que todos os dataframes possuem 30 colunas e cada um possui o número de linhas diferentes, podemos verificar abaixo que a união de todos os datasets resultam na casa dos 266.013 registros.

```
Todas as Linhas X Colunas de 2017 (89563, 30)
Todos as Linhas X Colunas de 2018 (69295, 30)
Todos as Linhas X Colunas de 2019 (67446, 30)
Todos as Linhas X Colunas de 2020 (39709, 30)
SOMA DE TODOS OS DF Linhas X Colunas 2017+2018+2019+2020 = (266013, 30)
```

Figura 11 – Quantidade de Linha x Colunas de cada Dataframe

3.4 Limpeza e tratamento dos Dados

Agora vamos entrar em uma das etapas mais importantes e trabalhosas quando o assunto é limpeza e tratamento dos dados.

Nesta etapa fica claro que é importante entender os dados que estamos trabalhando, identificar os dados que estão faltando e descobrir se existem possíveis valores que podem alterar o resultado, encontrar padrões e soluções para os dados apresentados. Sendo assim já sabemos quantas linhas e colunas temos no dataframe, agora precisamos entender quais colunas são essas, quais tipos de dado cada coluna armazena esses dados.

Para iniciar iremos usar a função .head(10) ela exibirá os primeiros 10 registros desse dataframe, assim já podemos ter uma noção inicial de como está nosso dataframe, de cara já foi possível descobrir padrões estranho na figura abaixo, temos colunas como ID, pesid, br, km e até idade com o tipo de dado flutuante (Float64)

```
DadosAcidentes.head(10)
```

	id	data_inversa	día_semana	horario	uf	br	km	município	causa_acidente	tipo_acidente	...
0	8.0	2017-01-01	domingo	00:00:00	PR	376.0	112	PARANAVAI	Fenômenos da Natureza	Queda de ocupante de veículo	...
1	9.0	2017-01-01	domingo	00:01:00	SC	101.0	234	PALHOCA	Falta de Atenção à Condução	Colisão com objeto estático	...
2	11.0	2017-01-01	domingo	00:00:00	PR	153.0	56,9	SANTO ANTONIO DA PLATINA	Animais na Pista	Capotamento	...
3	12.0	2017-01-01	domingo	00:00:00	GO	153.0	435	ANAPOLIS	Avarias e/ou desgaste excessivo no pneu	Tombamento	...
4	13.0	2017-01-01	domingo	00:00:00	SC	280.0	77,3	CORUPA	Ingestão de Álcool	Saída de leito carroçável	...
5	14.0	2017-01-01	domingo	00:40:00	GO	60.0	188	GUAPO	Falta de Atenção à Condução	Colisão traseira	...
6	15.0	2017-01-01	domingo	00:01:00	PB	104.0	3,4	NOVA FLORESTA	Ingestão de Álcool	Tombamento	...
7	16.0	2017-01-01	domingo	00:30:00	TO	153.0	141,7	ARAGUAINA	Ingestão de Álcool	Colisão traseira	...
8	17.0	2017-01-01	domingo	01:45:00	RS	116.0	34,9	VACARIA	Defeito Mecânico no Veículo	Colisão traseira	...
9	18.0	2017-01-01	domingo	01:40:00	RS	290.0	722	URUGUAIANA	Animais na Pista	Atropelamento de Animal	...

Figura 12 – imprimindo as primeiras linhas do Dataframe com .head(10)

Eu particularmente prefiro usar a função `sample`, pois ela retorna uma amostra selecionada de forma aleatória. Eu solicitei 5 itens do `DADOSACIDENTES` usando a forma transposta dos dados, a diferença para o `.head` é que na figura acima eu não consigo visualizar todas as colunas e linhas da amostra, já dessa forma consigo visualizar todos os dados conforme na figura baixo.


```
# A função sample retorna uma amostra selecionada de forma aleatória usando a forma transposta.
DadosAcidentes.sample(5).T
```

	253514	126452	32999	236636	257375
id	291287	146096	38033	272109	295617
data_inversa	2020-06-21	2018-07-12	2017-05-17	2020-02-28	2020-07-15
dia_semana	domingo	quinta-feira	quarta-feira	sexta-feira	quarta-feira
horario	19:45:00	07:30:00	15:05:00	02:00:00	20:30:00
uf	PR	MS	SC	SC	PB
br	153	158	101	101	230
km	48	57,5	144,9	146	334,5
municipio	SANTO ANTONIO DA PLATINA	PARANAIBA	ITAPEMA	ITAPEMA	PATOS
causa_acidente	Falta de Atenção à Condução	Avarias e/ou desgaste excessivo no pneu	Não guardar distância de segurança	Objeto estático sobre o leito carroçável	Falta de Atenção do Pedestre
tipo_acidente	Colisão traseira	Colisão frontal	Colisão traseira	Colisão com objeto estático	Atropelamento de Pedestre
classificacao_acidente	Sem Vítimas	Com Vítimas Feridas	Com Vítimas Feridas	Com Vítimas Feridas	Com Vítimas Feridas
fase_dia	Plena Noite	Pleno dia	Pleno dia	Plena Noite	Plena Noite
sentido_via	Crescente	Crescente	Crescente	Decrescente	Crescente
condicao_metereologica	Céu Claro	Céu Claro	Céu Claro	Céu Claro	Céu Claro
tipo_pista	Simple	Simple	Múltipla	Múltipla	Simple
tracado_via	Não Informado	Curva	Reta	Curva	Reta
uso_solo	Não	Não	Não	Sim	Não
peessoas	3	2	2	1	3
mortos	0	0	0	0	0
feridos_leves	0	1	1	1	0
feridos_graves	0	0	0	0	2
ilesos	1	1	1	0	1
ignorados	2	0	0	0	0
feridos	0	1	1	1	2
veiculos	2	2	2	1	2
latitude	-23,34623012	-19,427238	-27,08038827	-27,08666525	-7,018
longitude	-50,0664882	-51,374901	-48,60073325	-48,60955587	-37,2589
regional	SR-PR	SR-MS	SR-SC	SR-SC	SR-PB

Figura 13 – visualizando os dados com a função sample com a forma transposta

Para iniciar a análise com mais detalhes, iremos usar a função .info() ela mostra todos os tipos de dados de cada coluna, conseguindo identificar se o dataframe estão com os tipos de dados recomendados para suas finalidades.

Visualização dos Tipos de cada Coluna

```
DadosAcidentes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266013 entries, 0 to 266012
Data columns (total 30 columns):
id                266013 non-null float64
data_inversa      266013 non-null object
dia_semana        266013 non-null object
horario           266013 non-null object
uf                266013 non-null object
br                265560 non-null float64
km                265560 non-null object
municipio         266013 non-null object
causa_acidente    266013 non-null object
tipo_acidente     266013 non-null object
classificacao_acidente 266013 non-null object
fase_dia          266013 non-null object
sentido_via       266013 non-null object
condicao_metereologica 266013 non-null object
tipo_pista        266013 non-null object
tracado_via       266013 non-null object
uso_solo          266013 non-null object
pessoas           266013 non-null int64
mortos            266013 non-null int64
feridos_leves     266013 non-null int64
feridos_graves    266013 non-null int64
ilesos            266013 non-null int64
ignorados         266013 non-null int64
feridos           266013 non-null int64
veiculos          266013 non-null int64
latitude          266013 non-null object
longitude         266013 non-null object
regional          266013 non-null object
delegacia         266013 non-null object
uop               254923 non-null object
dtypes: float64(2), int64(8), object(20)
memory usage: 60.9+ MB
```

Figura 14 – tipos de dados do dataframe

Após identificar os tipos de dados cada coluna armazena, eu já posso traçar a estratégia de quais colunas precisam de alteração nos tipos de dados e assim consequentemente otimizar os dados e diminuir o uso da memória.

Antes de alterar os tipos de cada coluna eu irei criar uma coluna com um contador com auto incremento e remover colunas que não fazem sentidos para nossa análise.

```
DadosAcidentes['contador'] = range(1, 1+len(DadosAcidentes))
```

Figura 15 – Criação da Coluna Contador com Auto incremento

```
DadosAcidentes.pop("delegacia")  
DadosAcidentes.pop("uop")  
DadosAcidentes.pop("regional")
```

Figura 16 – Remoção das colunas inúteis através da função pop

Conforme descrito anteriormente no dicionário dos dados utilizados nessa análise, essas três colunas não fornecem informações para responder as perguntas que foram questionadas no início desse documento. Sendo assim, resolvi remover do dataframe, mantendo apenas as colunas que fazem sentido para nossa análise exploratória dos dados.

Já tendo ciência dos tipos de dados que cada coluna possui, agora iremos fazer a alteração e a identificação de cada tipo de dado. Nesta etapa dados que se enquadram em categorias, como por exemplo a condição meteorológica do acidente, eu considero como um tipo de dado categórico ela consiste em informar se no dia estava com Céu Claro, Chuva, Vento ou outra condição meteorológica. Cada ocorrência vai informar o estado meteorológico cadastrado. Sendo assim, esse tipo de dado segue um padrão nominal e faz mais sentido em ser category do que Object que são dados textuais e são tratados como string.

```

DadosAcidentes.id = DadosAcidentes.id.astype('int32')
DadosAcidentes.data_inversa = DadosAcidentes.data_inversa.astype('object')
DadosAcidentes.dia_semana = DadosAcidentes.dia_semana.astype('object')
DadosAcidentes.horario = DadosAcidentes.horario.astype('object')
DadosAcidentes.uf = DadosAcidentes.uf.astype('category')
DadosAcidentes.br = DadosAcidentes.br.astype('object')
DadosAcidentes.km = DadosAcidentes.km.astype('object')
DadosAcidentes.municipio = DadosAcidentes.municipio.astype('category')
DadosAcidentes.causa_acidente = DadosAcidentes.causa_acidente.astype('category')
DadosAcidentes.tipo_acidente = DadosAcidentes.tipo_acidente.astype('category')
DadosAcidentes.classificacao_acidente = DadosAcidentes.classificacao_acidente.astype('category')
DadosAcidentes.fase_dia = DadosAcidentes.fase_dia.astype('category')
DadosAcidentes.sentido_via = DadosAcidentes.sentido_via.astype('category')
DadosAcidentes.condicao_metereologica = DadosAcidentes.condicao_metereologica.astype('category')
DadosAcidentes.tipo_pista = DadosAcidentes.tipo_pista.astype('category')
DadosAcidentes.tracado_via = DadosAcidentes.tracado_via.astype('category')
DadosAcidentes.uso_solo = DadosAcidentes.uso_solo.astype('category')
DadosAcidentes.pessoas = DadosAcidentes.pessoas.astype('uint8')
DadosAcidentes.mortos = DadosAcidentes.mortos.astype('uint8')
DadosAcidentes.feridos_leves = DadosAcidentes.feridos_leves.astype('uint8')
DadosAcidentes.feridos_graves = DadosAcidentes.feridos_graves.astype('uint8')
DadosAcidentes.ilesos = DadosAcidentes.ilesos.astype('uint8')
DadosAcidentes.ignorados = DadosAcidentes.ignorados.astype('uint8')
DadosAcidentes.feridos = DadosAcidentes.feridos.astype('uint8')
DadosAcidentes.veiculos = DadosAcidentes.veiculos.astype('uint8')
DadosAcidentes.latitude = DadosAcidentes.latitude.astype('object')
DadosAcidentes.longitude = DadosAcidentes.longitude.astype('object')
DadosAcidentes.contador = DadosAcidentes.contador.astype('int32')

```

Figura 17 – Ajustando os Dados para os Formatos Corretos e Otimização dos tipos de dados para economizar memória

Após realizar a padronização dos tipos de dados, vamos verificar se houve redução no consumo de memória usada nesses dados modificados e também verificar se as categorias foram modificadas com sucesso. Para isso vou chamar novamente a função `.info()` para mostrar todos os tipos de dados de cada coluna do dataframe.

Comparativo Antes da normalização dos tipos de dados 60.9+ MB

memory usage: 60.9+ MB

Figura 18 – Antes de Normalizar

Comparativo após a normalização dos tipos de dados 21.4+ MB

memory usage: 21.4+ MB

Figura 19 – Antes de Normalizar

Foi possível notar uma economia de 39.5 MB de memória apenas com essa normalização dos tipos dos dados.

```
DadosAcidentes.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266013 entries, 0 to 266012
Data columns (total 28 columns):
id                266013 non-null int32
data_inversa      266013 non-null object
dia_semana        266013 non-null object
horario           266013 non-null object
uf                266013 non-null category
br                265560 non-null object
km                265560 non-null object
municipio         266013 non-null category
causa_acidente    266013 non-null category
tipo_acidente     266013 non-null category
classificacao_acidente 266013 non-null category
fase_dia          266013 non-null category
sentido_via       266013 non-null category
condicao_metereologica 266013 non-null category
tipo_pista        266013 non-null category
tracado_via       266013 non-null category
uso_solo          266013 non-null category
pessoas           266013 non-null uint8
mortos            266013 non-null uint8
feridos_leves     266013 non-null uint8
feridos_graves    266013 non-null uint8
ilesos            266013 non-null uint8
ignorados         266013 non-null uint8
feridos           266013 non-null uint8
veiculos          266013 non-null uint8
latitude          266013 non-null object
longitude         266013 non-null object
contador          266013 non-null int32
dtypes: category(11), int32(2), object(7), uint8(8)
memory usage: 21.4+ MB
```

Figura 20 – Verificando os novos tipos de dados

Agora que temos os tipos de dados otimizados e padronizados para nosso projeto, eu gostaria de criar uma nova coluna para esse dataframe. Acredito que criar uma coluna para analisar qual região do país o acidente aconteceu pode gerar insight interessantes quando for fazer análises por ano, semestre, trimestre e até por mês. Para realizar essa tarefa eu vou precisar usar o campo UF e associar essa UF a sua respectiva região e para facilitar esse processo vou criar as seguintes listas abaixo e associar as suas respectivas UF.

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

Eu usei o wikipedia para facilitar e não errar na hora de vincular as unidades federativas com suas respectivas regiões, usei a imagem abaixo como referência.



Figura 21 – Mapa do brasil com suas respectivas regiões (fonte: [wikipedia](https://pt.wikipedia.org/wiki/Regi%C3%B5es_do_Brasil))

Após criar as listas das regiões e vincular com suas respectivas unidades federativas, eu crio uma variável chamada regiões que vai receber uma lista vazia e em seguida eu crio um laço de repetição com o comando for, onde ele vai analisar cada registro do campo UF e verificar em qual lista de regiões ele se enquadra, uma vez se enquadrando na sua respectiva região, eu adiciono na coluna regiões o nome da região que ele representa através da função append.

Segue abaixo a estrutura de código necessária para criar essa coluna Regiões no Dataframe.

```

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

for x in DadosAcidentes.uf:
    if x in norte:
        regioes.append('NORTE')

    if x in nordeste:
        regioes.append('NORDESTE')

    if x in centro_oeste:
        regioes.append('CENTRO-OESTE')

    if x in sudeste:
        regioes.append('SUDESTE')

    if x in sul:
        regioes.append('SUL')

    if x == '****':
        regioes.append('****')

print(regioes[0:10])
print(len(regioes))
DadosAcidentes['regioes'] = regioes
DadosAcidentes.regioes = DadosAcidentes.regioes.astype('category')

```

Figura 22 –Criação da Coluna região no Dataframe

No código acima eu solicito que ele imprima um slice dos 10 primeiros registros da coluna regiões para ter certeza que meu código estava realmente funcionando, além disso eu crio dentro do meu Dataframe a coluna regiões e informo que o tipo de dado dessa coluna será category. Imprimo quantidade total de registros na coluna regiões, assim tenho certeza que todos os campos foram preenchidos corretamente.

```

['SUL', 'SUL', 'SUL', 'CENTRO-OESTE', 'SUL', 'CENTRO-OESTE', 'NORDESTE', 'NORTE', 'SUL', 'SUL']
266013

```

Figura 23 – Resultado dos Prints do comando executado na figura 19

Para verificar todas as colunas que foram criadas até o momento eu utilizo o comando

Columns e ele me retorna todas as colunas do meu dataframe. É possível verificar que as colunas delegacia, regional e UOP foram removidas e as colunas contador e regiões foram adicionadas no final do meu dataframe DadosAcidentes.

```
: DadosAcidentes.columns

: Index(['id', 'data_inversa', 'dia_semana', 'horario', 'uf', 'br', 'km',
        'municipio', 'causa_acidente', 'tipo_acidente',
        'classificacao_acidente', 'fase_dia', 'sentido_via',
        'condicao_meteorologica', 'tipo_pista', 'tracado_via', 'uso_solo',
        'pessoas', 'mortos', 'feridos_leves', 'feridos_graves', 'ilesos',
        'ignorados', 'feridos', 'veiculos', 'latitude', 'longitude', 'regional',
        'contador', 'regioes'],
        dtype='object')
```

Figura 24 - Todas as Colunas do Dataframe Dados Acidentes

O próximo passo é realizar a padronização dos valores armazenados nas colunas com tipos categóricos. Esse processo é importante pois estamos utilizando quatro datasets de anos diferentes e eles podem ter padrões de dados diferentes. Com isso vamos usar o comando .unique para identificar os valores únicos dentro de uma coluna, assim podemos eliminar problemas como erros de digitação, uso de maiúsculo e minúsculo gerando categorias duplicadas ou ambíguas.

```
sorted(DadosAcidentes.causa_acidente.unique())

['Agressão Externa',
 'Animais na Pista',
 'Avarias e/ou desgaste excessivo no pneu',
 'Carga excessiva e/ou mal acondicionada',
 'Condutor Dormindo',
 'Defeito Mecânico no Veículo',
 'Defeito na Via',
 'Deficiência ou não Acionamento do Sistema de Iluminação/Sinalização do Veículo',
 'Desobediência às normas de trânsito pelo condutor',
 'Desobediência às normas de trânsito pelo pedestre',
 'Falta de Atenção do Pedestre',
 'Falta de Atenção à Condução',
 'Fenômenos da Natureza',
 'Ingestão de Substâncias Psicoativas',
 'Ingestão de Alcool',
 'Ingestão de álcool e/ou substâncias psicoativas pelo pedestre',
 'Mal Súbito',
 'Não guardar distância de segurança',
 'Objeto estático sobre o leito carroçável',
 'Pista Escorregadia',
 'Restrição de Visibilidade',
 'Sinalização da via insuficiente ou inadequada',
 'Ultrapassagem Indevida',
 'Velocidade Incompatível']
```

Figura 25 – Verificando valores únicos dentro da coluna Causa_Acidente


```
sorted(DadosAcidentes.tracado_via.unique())
```

```
['Curva',
 'Desvio Temporário',
 'Interseção de vias',
 'Não Informado',
 'Ponte',
 'Reta',
 'Retorno Regulamentado',
 'Rotatória',
 'Túnel',
 'Viaduto']
```

Figura 26 – Verificando valores únicos dentro da coluna tracado_via

```
sorted(DadosAcidentes.tipo_acidente.unique())
```

```
['Atropelamento de Animal',
 'Atropelamento de Pedestre',
 'Capotamento',
 'Colisão com objeto em movimento',
 'Colisão com objeto estático',
 'Colisão frontal',
 'Colisão lateral',
 'Colisão transversal',
 'Colisão traseira',
 'Danos eventuais',
 'Derramamento de carga',
 'Engavetamento',
 'Incêndio',
 'Queda de ocupante de veículo',
 'Saída de leito carroçável',
 'Tombamento']
```

Figura 27 – Verificando valores únicos dentro da coluna tipo_acidente

Para não tornar esse documento extenso, eu executei o comando `.unique` em cada coluna que possuem tipo de dados categóricos do dataframe `DadosAcidentes`. Eu basicamente solicito que o python execute apenas os valores únicos dentro da coluna de tipos categóricos, para exemplificar eu mostro a saída desse comando em algumas colunas, podemos notar na figura 25 até a figura 27 todos os valores são únicos e não existe ambiguidades ou duplicidades nos valores dos dados.

Outro fator que considero importante para checar a qualidade dos dados é analisar se existe outliers nos números de mortos e acidentados, já que esses são os dados de extrema relevância para este projeto.

Os Outliers são dados que se diferenciam drasticamente de todos os outros, são pontos fora da curva normal, que podem ser dados realmente atípicos e verdadeiros, ou até mesmos dados que foram inseridos de forma errada. Imagine na hora de armazenar a idade no dataset ao invés de digitarem 20 anos, por descuido acaba digitando um ou dois zero a mais (200 ou 2000 anos), isso pode gerar um problema muito grande caso alguém faça a média das idades por exemplo.

Para identificar se existem valores exorbitantes, causados por erros de digitação, vou usar um boxplot, que é uma ferramenta gráfica onde possibilita visualizar a distribuição e valores discrepantes (outliers) dos dados. Para isso eu uso o comando boxplot e referencio a coluna mortos.

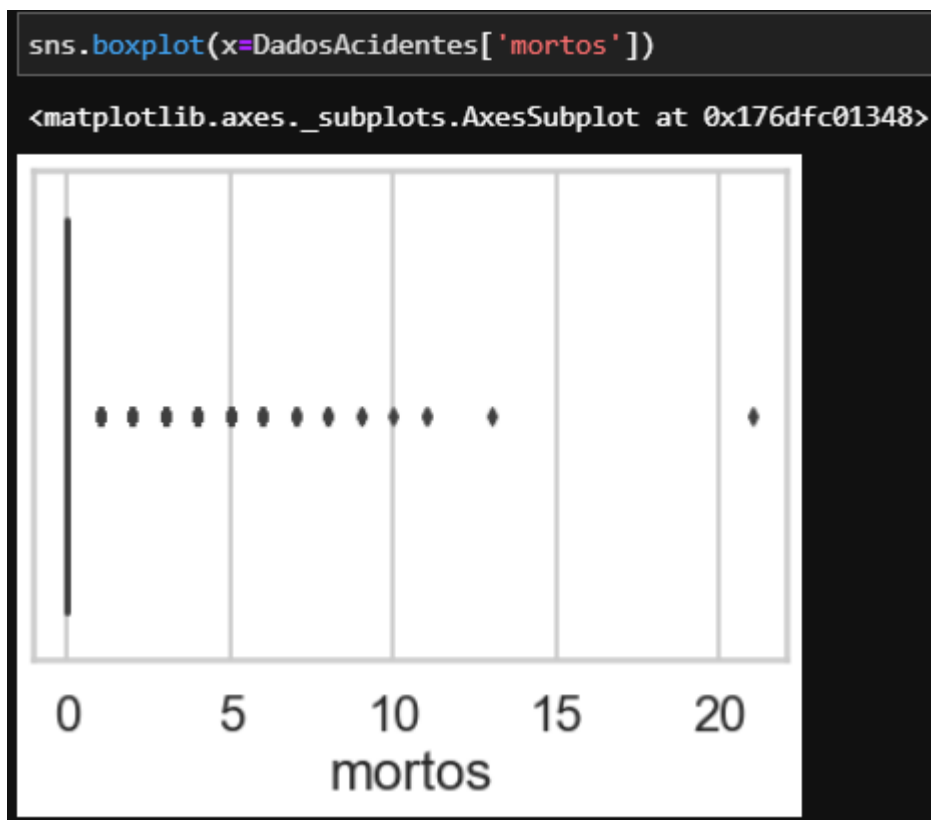
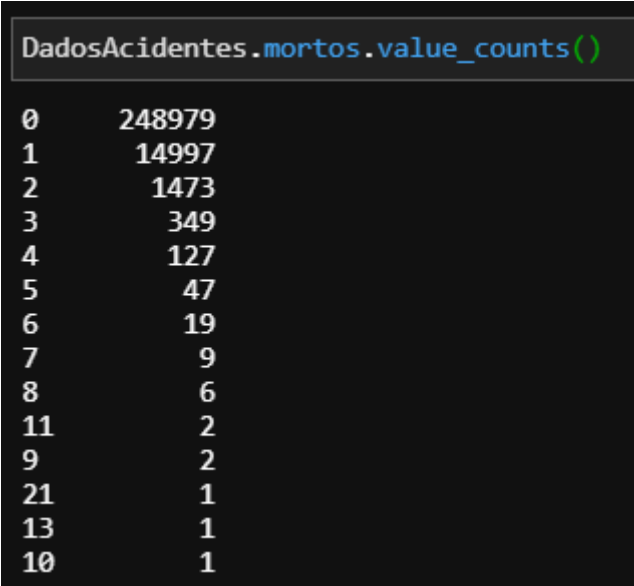


Figura 28 – Identificando Outliers na coluna Mortos

Como podemos verificar na Figura 28 é possível identificar que a grande maioria dos registros o padrão é 0 mortes, porém alguns acidentes realmente podem ter números de mortes maiores, como um carro com 4 passageiros, uma van com 10 passageiros ou até um ônibus que possui uma capacidade maior.

Para ter uma noção de qual a quantidade de mortos é mais frequente, vamos usar o comando `value_counts` na coluna `mortos`, ela vai me listar a contagem de cada quantidade de mortes registradas no Dataframe.



```
DadosAcidentes.mortos.value_counts()
```

0	248979
1	14997
2	1473
3	349
4	127
5	47
6	19
7	9
8	6
11	2
9	2
21	1
13	1
10	1

Figura 29 – Quantidade de Mortos

Sabemos que temos 266.013 registros no DataFrame `DadosAcidentes`, sendo 248.979 são registros com 0 mortos, 14.997 com 1 mortos, 1.473 com 2 mortos e assim sucessivamente conforme a Figura 26.

Resolvi investigar esse acidente que consta 21 mortos, para tentar descobrir se existe alguma informação sobre ele na internet em jornais ou portais de notícias.

Para extrair essa informação eu vou usar o comando `.query` e vou pedir para filtra a coluna mortos e me mostrar somente se for igual (`==`) a 21, essa execução retorna a figura abaixo:

DadosAcidentes.query('mortos == 21')

id	data_inversa	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente
48048	2017-06-22	quinta-feira	06:10:00	ES	101	343,4	GUARAPARI	Velocidade Incompatível	Tombamento
feridos_leves	feridos_graves	ilesos	ignorados	feridos	veiculos	latitude	longitude	contador	regioes
13	9	0	0	22	4	-20,6636897	-40,5939666	41870	SUDESTE

Figura 30 – Registro do Dataset de 21 mortos

Com as informações do comando acima, vi que ocorreu um acidente no dia 22-06-2017 em Guarapari em que o tipo de acidente foi tombamento e foi causado por velocidade incompatível, além dos 21 mortos, tiveram 22 feridos. Realizei uma pesquisa na internet para verificar se esse acidente foi noticiado na mídia.

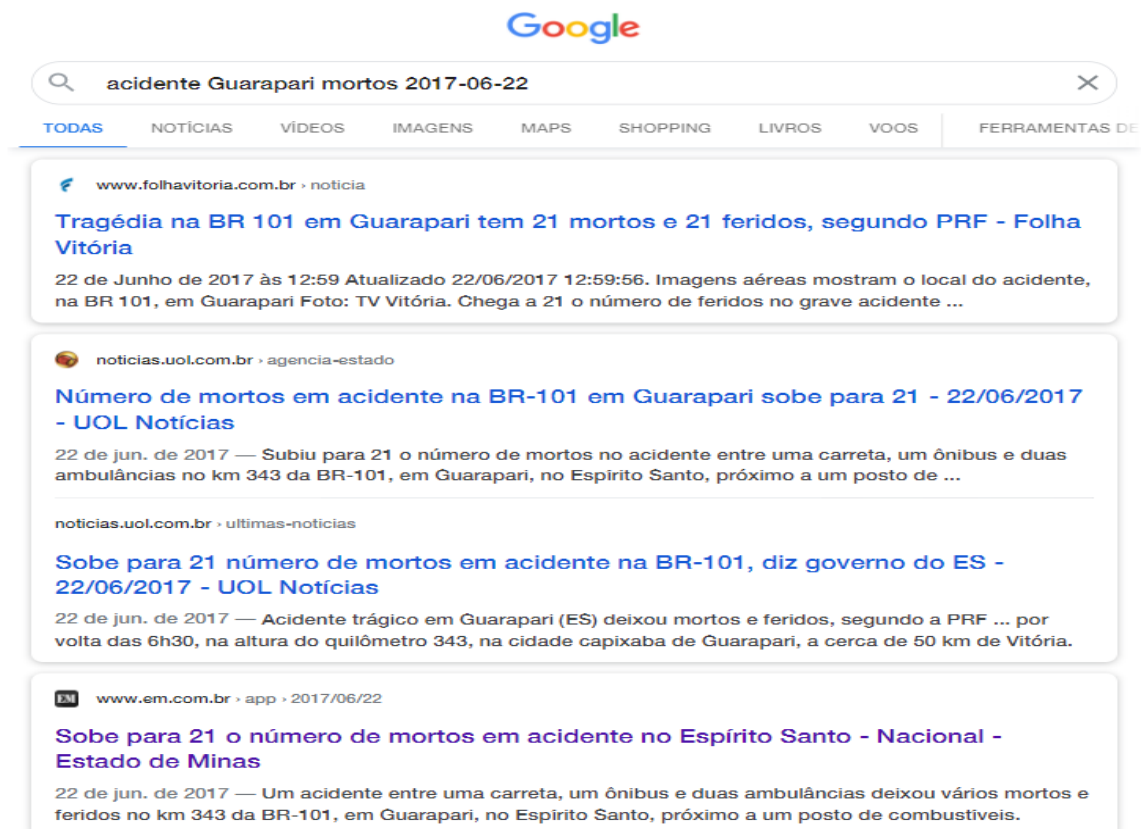


Figura 31 – Pesquisa no Google pelo acidente com mortos em 22-06-2017

Após verificar que os outlier das colunas importantes são reais, não vamos realizar nenhuma exclusão até o momento. Agora que já tenho ciência dos tipos de dados que cada coluna possui, padronizando os valores dos dados categóricos, eu preciso identificar se existem dados faltantes no meu dataset. Para realizar essa descoberta eu irei utilizar a função `isnull().sum()` ela me retorna a soma de todos os valores ausentes por coluna em meu dataset, confira a imagem abaixo.

```
DadosAcidentes.isnull().sum()
id                0
data_inversa      0
dia_semana        0
horario           0
uf                0
br                453
km                453
municipio         0
causa_acidente    0
tipo_acidente     0
classificacao_acidente 0
fase_dia          0
sentido_via       0
condicao_meteorologica 0
tipo_pista        0
tracado_via       0
uso_solo          0
pessoas           0
mortos            0
feridos_leves     0
feridos_graves    0
ilesos            0
ignorados         0
feridos           0
veiculos          0
latitude          0
longitude         0
contador          0
regioes           0
dtype: int64
```

Figura 32 – Dados Faltantes

Conforme a figura acima, é possível identificar que temos duas colunas com dados faltantes. São dados que informam qual BR e KM houve o acidente e como não temos essa informação, podemos tratar de diversas formas, seja excluindo esses dados faltantes ou simplesmente criando um valor para eles. Nesse caso eu vou preencher todos os valores faltantes como Não Informado.

```
DadosAcidentes.update(DadosAcidentes['br'].fillna('Não Informado'))
DadosAcidentes.update(DadosAcidentes['km'].fillna('Não Informado'))
```

Figura 33 – Transformando dados faltantes em um valor desejado

Agora vamos fazer novamente o comando `IsNull().sum()` para verificar se os dados que eram nulos agora foram preenchidos com o texto 'não informado'.

```
DadosAcidentes.isnull().sum()
id 0
data_inversa 0
dia_semana 0
horario 0
uf 0
br 0
km 0
municipio 0
causa_acidente 0
tipo_acidente 0
classificacao_acidente 0
fase_dia 0
sentido_via 0
condicao_meteorologica 0
tipo_pista 0
tracado_via 0
uso_solo 0
pessoas 0
mortos 0
feridos_leves 0
feridos_graves 0
ilesos 0
ignorados 0
feridos 0
veiculos 0
latitude 0
longitude 0
contador 0
regioes 0
dtype: int64
```

Figura 34 – Execução do `IsNull().sum()`

Finalizando a primeira etapa de tratamento prévio dos dados e preparação para exportar para o power BI, para gerar esse novo arquivo utilizei a função `.to_csv` onde ele converte esse arquivo para csv, além de ser possível deixar explícito o tipo de separador que o arquivo vai conter e o encoding final do arquivo.

```
# Salvando todas as alterações desse dataset em um arquivo csv  
df17181920.to_csv('tcc-2020.csv', sep=';', encoding='utf-8-sig')
```

Figura 35 – Exportando para CSV

4. Explorando os dados no Power BI

Agora chegamos na segunda etapa que, após fazer uma breve limpeza e tratamento nos dados, é hora de levar para o power BI e explorar os recursos poderosos que essa ferramenta oferece para análises e visualizações dos dados.

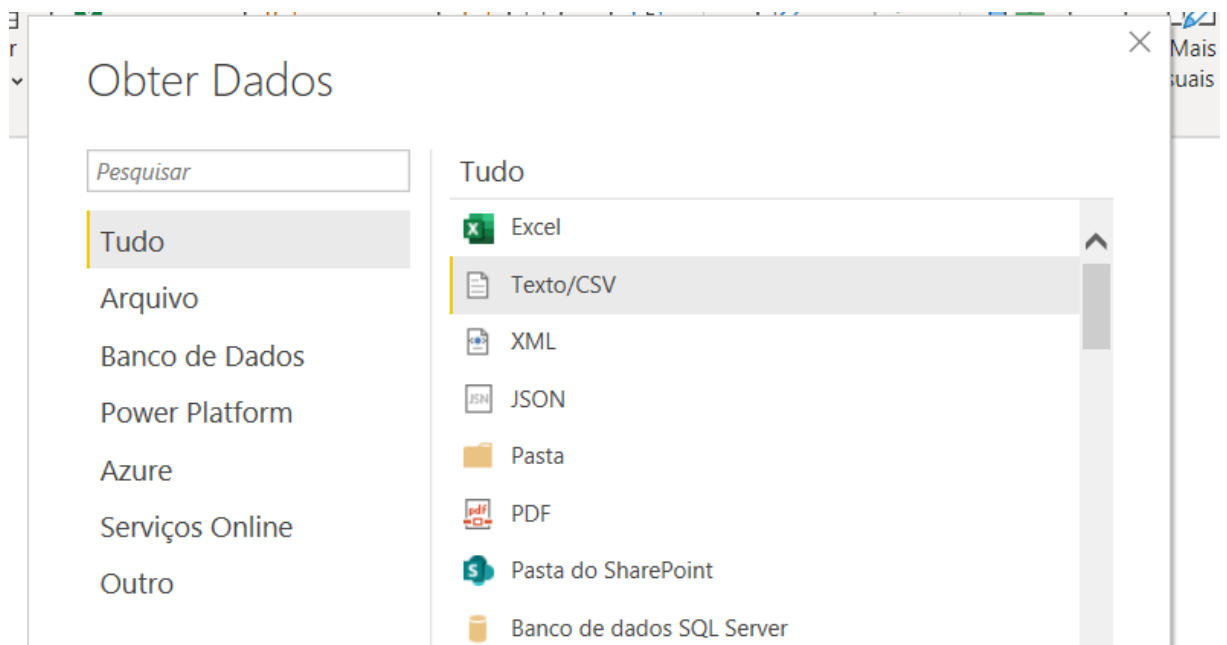


Figura 36 – Obtendo Dados com Power BI

Ao abrir o Power BI ele já abre com uma opção para obter os dados e ao clicar nesse botão ele abre uma imagem para escolher o tipo de arquivo que queremos carregar. Nesse caso vamos usar a opção Texto/CSV, conforme a figura 36.

Origem do Arquivo: 65001: Unicode (UTF-8) Delimitador: Ponto e vírgula Detecção de Tipo de Dados: Com base nas primeiras 200 linhas

	id	pesid	data_inversa	dia_semana	horario	uf	br	km	município	causa_principal	causa
0	8	1	2017-01-01	domingo	00:00:00	PR	376	112	PARANAVAI	Sim	Fenômenos da N
1	9	955	2017-01-01	domingo	00:01:00	SC	101	234	PALHOCA	Sim	Falta de Atenção
2	11	2	2017-01-01	domingo	00:00:00	PR	153	569	SANTO ANTONIO DA PLATINA	Sim	Animais na Pista
3	11	3	2017-01-01	domingo	00:00:00	PR	153	569	SANTO ANTONIO DA PLATINA	Sim	Animais na Pista
4	12	1499	2017-01-01	domingo	00:00:00	GO	153	435	ANAPOLIS	Sim	Avarias e/ou des
5	13	1892	2017-01-01	domingo	00:00:00	SC	280	773	CORUPA	Sim	Ingestão de Álco
6	14	1558	2017-01-01	domingo	00:40:00	GO	60	188	GUAPO	Sim	Falta de Atenção
7	14	1558	2017-01-01	domingo	00:40:00	GO	60	188	GUAPO	Não	Não guardar dist
8	14	1563	2017-01-01	domingo	00:40:00	GO	60	188	GUAPO	Sim	Falta de Atenção
9	14	1563	2017-01-01	domingo	00:40:00	GO	60	188	GUAPO	Não	Não guardar dist
10	15	555	2017-01-01	domingo	00:01:00	PB	104	34	NOVA FLORESTA	Sim	Ingestão de Álco
11	16	1913	2017-01-01	domingo	00:30:00	TO	153	1417	ARAGUAINA	Sim	Ingestão de Álco
12	16	1913	2017-01-01	domingo	00:30:00	TO	153	1417	ARAGUAINA	Não	Não guardar dist
13	16	1913	2017-01-01	domingo	00:30:00	TO	153	1417	ARAGUAINA	Sim	Ingestão de Álco
14	16	1913	2017-01-01	domingo	00:30:00	TO	153	1417	ARAGUAINA	Não	Não guardar dist
15	16	1909	2017-01-01	domingo	00:30:00	TO	153	1417	ARAGUAINA	Sim	Ingestão de Álco
16	16	1909	2017-01-01	domingo	00:30:00	TO	153	1417	ARAGUAINA	Não	Não guardar dist
17	16	988	2017-01-01	domingo	00:30:00	TO	153	1417	ARAGUAINA	Sim	Ingestão de Álco
18	16	988	2017-01-01	domingo	00:30:00	TO	153	1417	ARAGUAINA	Não	Não guardar dist
19	16	990	2017-01-01	domingo	00:30:00	TO	153	1417	ARAGUAINA	Sim	Ingestão de Álco

Carregar Transformar Dados Cancelar

Figura 37 – Tela de Carregamento dos Dados

Ao clicar na opção Texto/CSV o Power BI vai pedir para selecionar um arquivo para ser carregado, vamos usar arquivo que acabamos de gerar na etapa anterior que foi o TCC-2020-v3.csv, o próprio Power BI ele já identifica automaticamente o encoding do arquivo, o tipo de separador/ delimitador e exibe uma amostra breve dos dados para que você possa analisar. Estando tudo certo, basta clicar em carregar.

4.1. Criando a interface

Após carregar os dados já tratados para dentro do Power BI eu criei um template que consistem em uma arte gráfica para deixar o Dashboard com visual mais profissional. Usei para a base as cores preto e amarelo para representar visual, além disso uso uma imagem da PRF em transparência no fundo amarelo e a imagem em PNG de um carro da PRF.



Figura 38 – Imagem criada para o Background do Dashboard

O processo para inserir o Background no Power BI é bem simples. No menu formato > Papel de Parede e inseri a imagem acima no meu dashboard.

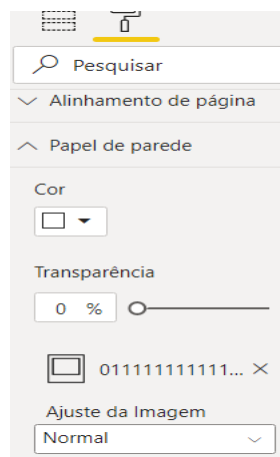


Figura 39 – Inseri a imagem e coloquei o Ajuste da Imagem para Normal

Após Adicionar meu template, eu vou adicionar na primeira página do meu Dashboard um painel com informações gerais

- Filtro por Período (Ano, Mês, Dia)
- Filtro por Estado
- Filtro por Regiões
- Total de Ocorrências
- Total de Vítimas
- Total de Feridos
- Total de Veículos
- Total de Mortos
- Mortos por 100 Acidentes
- Acidentes por Tipo de Pista
- Feridos vs lesos por Ano
- Acidentes por Região
- Mortos por Ano
- Acidentes por UF
- Mapa com todos os acidentes por Latitude e Longitude



Figura 40 – Total de Ocorrências

Total de Ocorrência - Eu uso um cartão com a soma a contagem distinta do campo ID do meu dataset, cada ID representa uma ocorrência.



Figura 41 – Total de Vítimas

Total de Vítimas – Eu uso um cartão e uso a soma do total de vítimas em cada ocorrência.



Figura 42 – Total de Feridos

Total de Feridos – Eu uso um cartão com a soma do total de feridos em cada ocorrência.



Figura 43 – Total de Veículos

Total de Veículos - Nesse campo eu uso a soma do total de veículos em cada ocorrência.



Figura 44 – Total de Mortos

Total de Mortos - Nesse campo eu uso a soma do total de mortos em cada ocorrência.

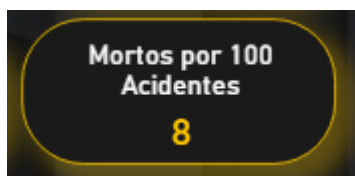


Figura 45 – Mortos por 100 Acidentes

Mortos por 100 Acidentes - Aqui eu crio uma média rápida no Power BI usando o seguinte código:

$$\text{Mortos por 100 acidentes} = \text{CALCULATE}((\text{SUM}('tcc-2020-v4'[Mortos]) / \text{COUNTA}('tcc-2020-v4'[Id])) * 100))$$

Eu uso a função calculate do power BI e faço a soma de todos os mortos do dataset e divido pela soma de cada ID que representa uma ocorrência e multiplico por 100.



Figura 46 – Acidentes por Região

Acidentes por Região - Eu uso o gráfico de barra empilhadas usando a coluna regiões como eixo e a contagem dos Ids como valor



Figura 47 – Feridos vs Ilesos por Ano

Feridos vs Ilesos por Ano – Eu uso um gráfico de faixa de opções com Eixo com os Anos e os valores com as colunas Feridos e as Colunas Ileso

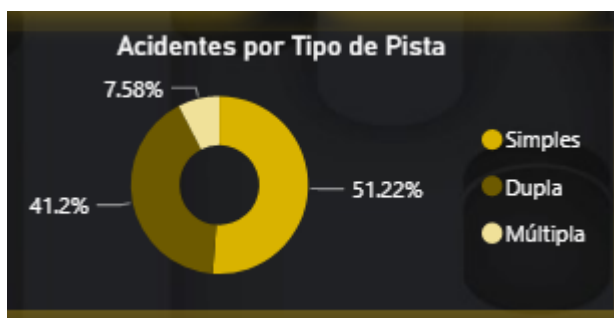


Figura 48 – Acidentes por Tipo de Pista

Acidente por Tipo de Pista – Aqui eu uso um gráfico de rosca com a legenda em tipo de pista e o valor a contagem de Id (ocorrências).

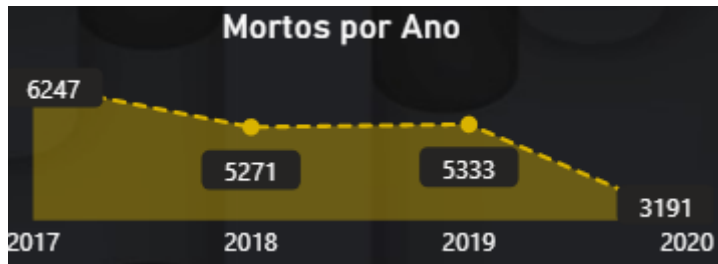


Figura 49 – Mortos por Ano

Mortos por Ano – Aqui eu uso um gráfico de áreas com eixo em anos e o valor com a coluna mortos



Figura 50 – Acidentes por UF

Acidentes por UF – Aqui eu uso o eixo UF e o valor utilizo a contagem da coluna ID (ocorrências)



Período

Todos ▾

Estado

Todos ▾

Regiões

Todos ▾

Figura 51 – Filtro

Filtros – Aqui eu tenho três campos de filtro, por período eu posso filtrar por (Ano, Trimestre, Mês e Dia), por Estado eu posso filtrar por UF e por Regiões onde posso filtrar as ocorrências por cada região especificada.

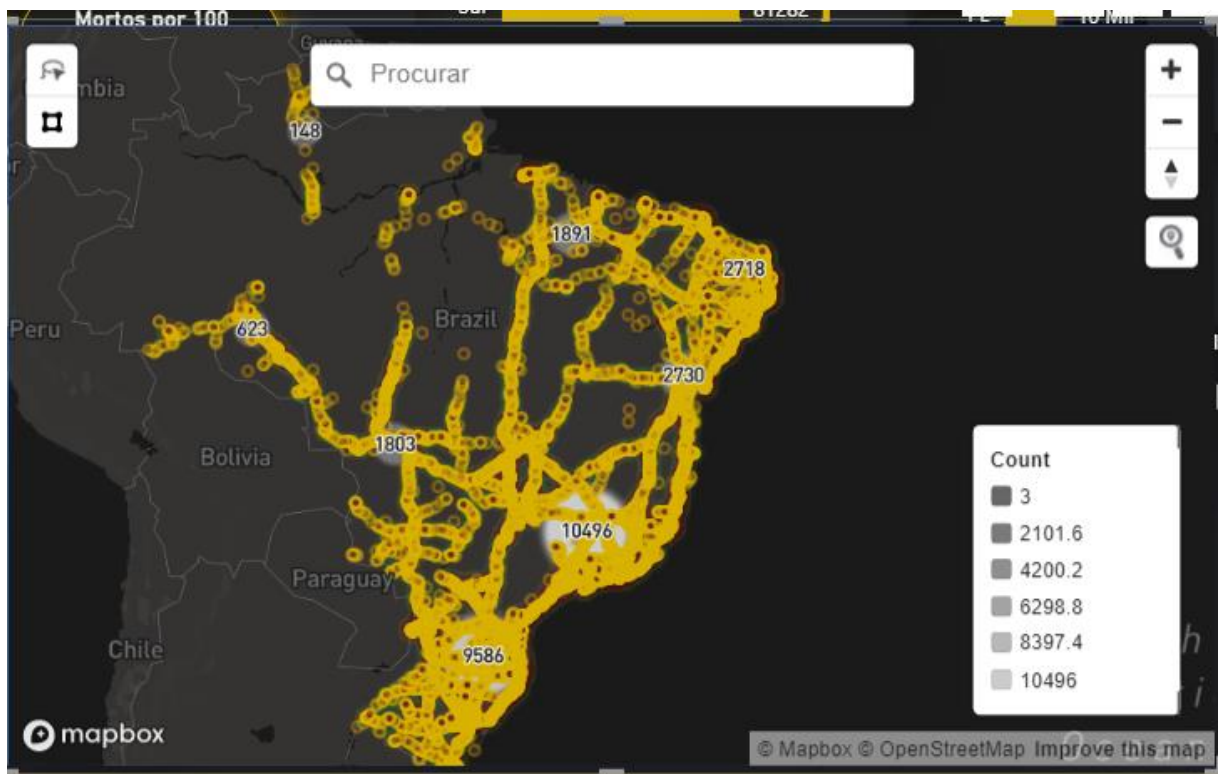


Figura 52 – Mapa de Ocorrência de Acidentes Rodoviários Brasileiro

Mapa de Ocorrências - Este é um mapa onde pode visualizar exatamente quais as BRs que mais tem acidentes, além disso posso pesquisar por região, estado, cidade e até município, posso inclusive selecionar com mouse um raio do local dos acidentes e explorar as informações detalhadas desse acidente.

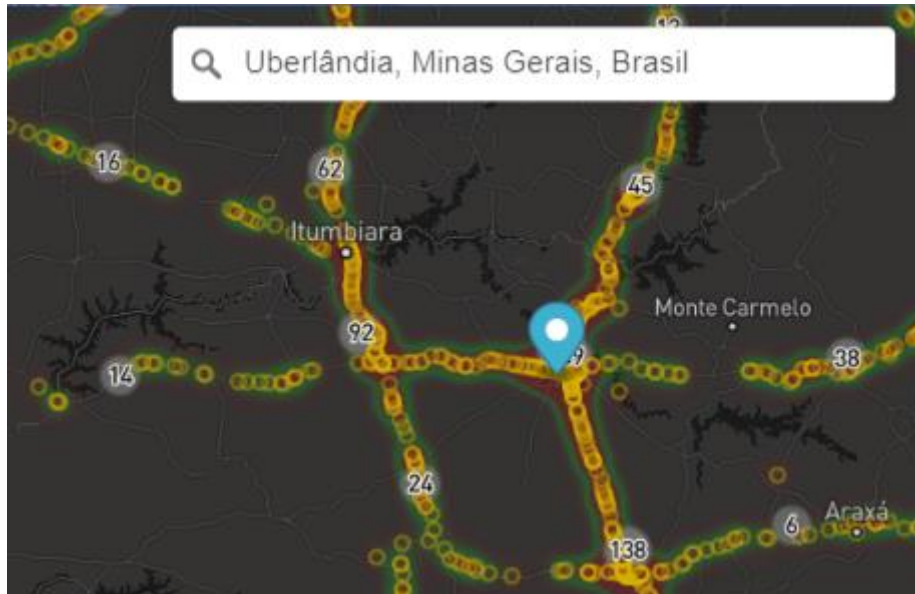


Figura 53 – Mapa de Ocorrência após aplicar o zoom dentro de uma localização.

Após pesquisar o município Uberlândia eu posso dar zoom no mapa e verificar os clusters de acidentes acumulados em cada ponto da BR, assim posso identificar pontos que mais acontecem acidentes.



Figura 54 – Mapa de Ocorrência de Acidentes Rodoviários Brasileiro após selecionar o raio de marcação

Para gerar esses gráficos direto do Power BI eu uso www.mapbox.com através de uma API e integro os dados do meu dataset, aqui uso ID, Latitude e Longitude e Data Inversa como Cluster.

4.2. Primeira tela do dashboard

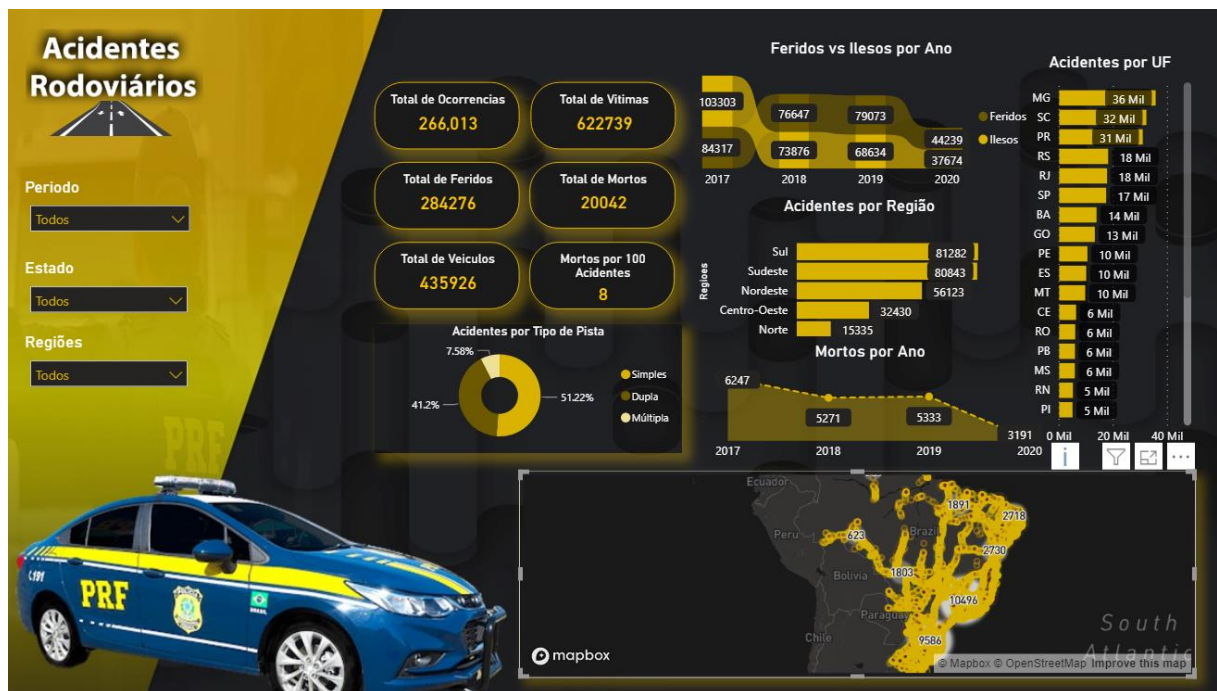


Figura 55 – Tela com Overview do Dashboard

Visualizando o mapa, é possível notar que algumas localizações estão totalmente fora do mapa do Brasil, alguma no meio do oceano ou próximo o continente africano. Porém isso é devido a um possível erro de latitude e longitude que pode ter ocorrido na conversão do arquivo para csv, durante a gravação do arquivo. Por questão de estética eu vou remover todos os dados que estão apresentando esse erro.

Para identificação desse erro eu criei uma tabela com a coluna id e usei o recurso do mapa, onde posso selecionar um raio específico no mapa. Selecionei todos os campos que estavam fora do mapa do Brasil e retornei a ID deles na tabela.

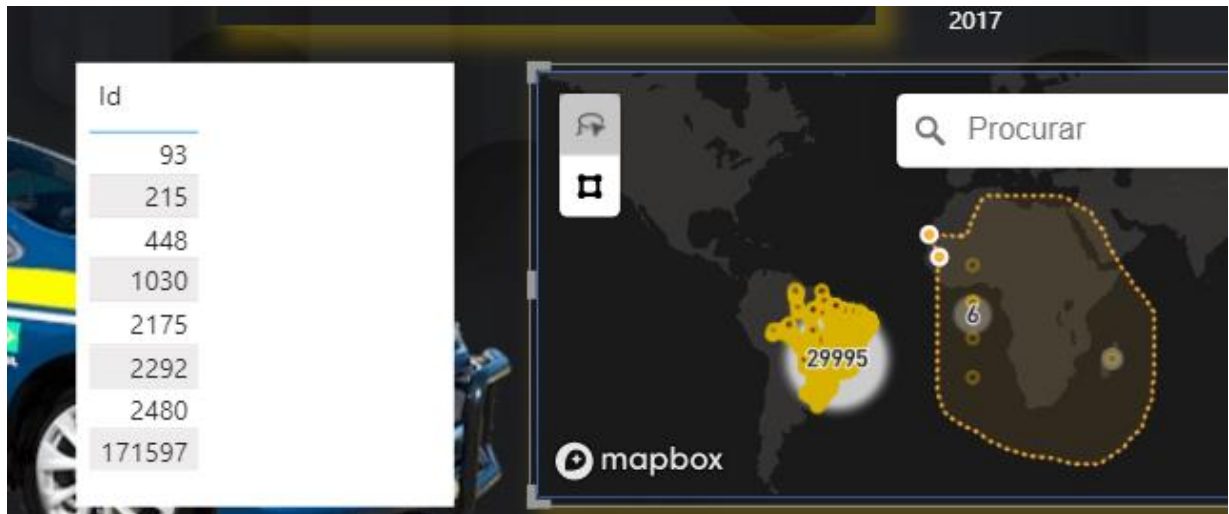


Figura 56 – Selecionando os Outliers no mapa

Agora que eu tenho as informações de quais dados eu tenho que remover, cliço no ícone transformar dados e filtro as ID que desejo remover. Vale lembrar que aqui estou apagando todos os dados referente essa ID, tomei essa decisão já que são apenas 8 registros em um universo de mais de 250 mil registros.

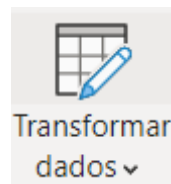


Figura 57 – Selecionando a opção transformar dados

= Table.SelectRows(#"Tipo Alterado1", each [Id] = 93)						
Id	Data_Inversa	Dia_Semana	Horario	Uf	Br	
93	1/1/2017	Domingo	8:00:00 AM	MT	163.0	

Figura 58 – Apagando os outliers via power BI

4.3. Segunda tela do dashboard

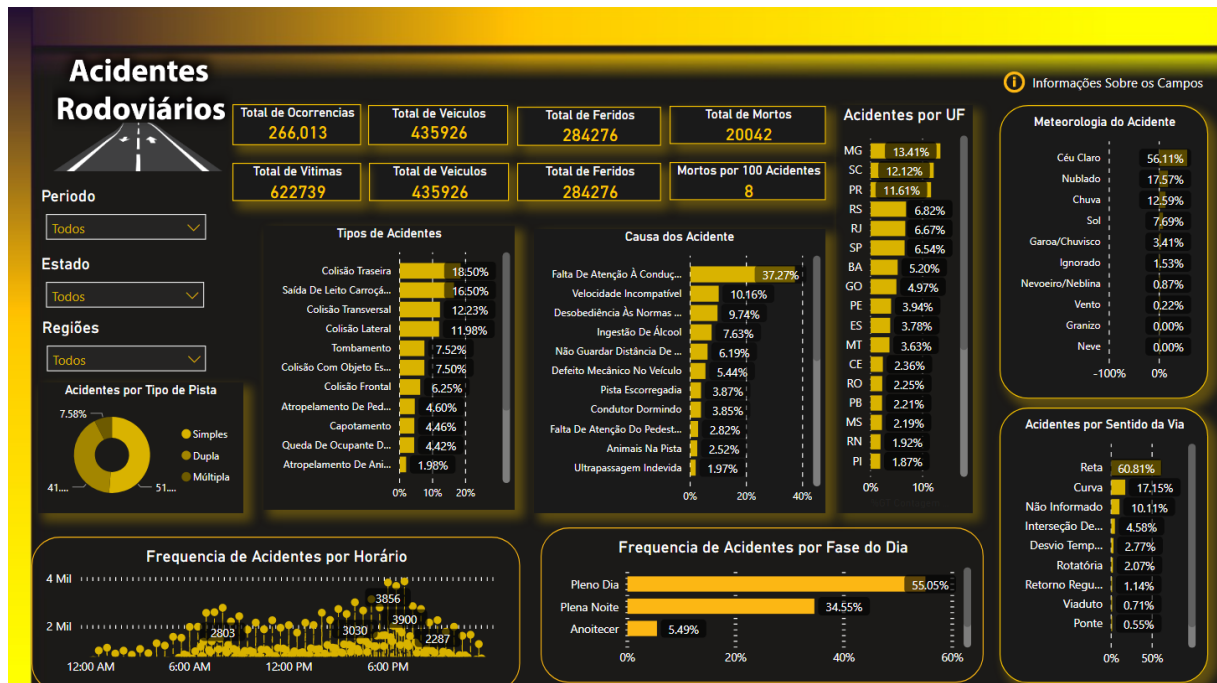


Figura 59 – Segunda tela do Dashboard com Overview Detalhado dos Acidentes

Nessa tela eu mostro com detalhes os principais dados dos acidentes, é possível saber o Tipo de Acidentes mais frequentes, causa dos Acidentes mais frequentes, acidentes por sentido da via, meteorologia do acidente, acidente por tipo de pista, frequência horaria do acidente e acidente por UF.

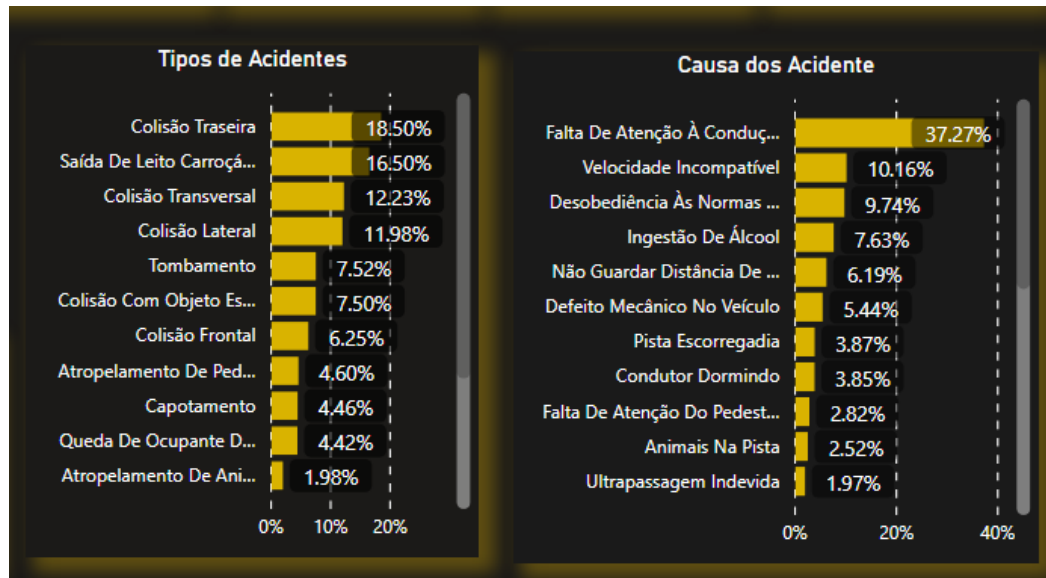


Figura 60 – Dados Tipo de Acidentes e Causa de Acidentes

Tipos de Acidentes e Causa dos Acidentes sendo exibidos com gráfico de barras empilhados e contabilizando a contagem do percentual total dos tipos de acidentes e das causas de acidentes.

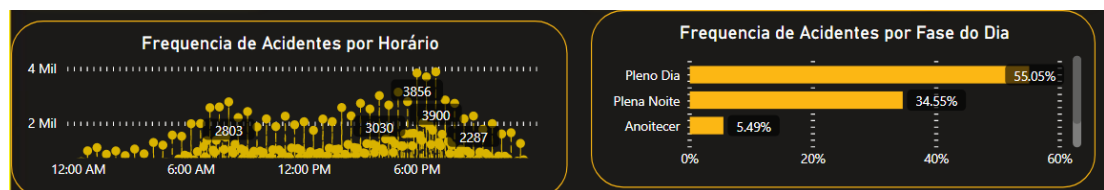


Figura 61 – Dados da frequência de acidentes por horário e por fase do dia

Frequência de Acidentes por Horário e Frequência de Acidentes por Fase do Dia - neste caso eu uso o gráfico de área empilhado e o gráfico para exibir os horários e o gráfico empilhado em barra para exibir os acidentes por fase do dia.

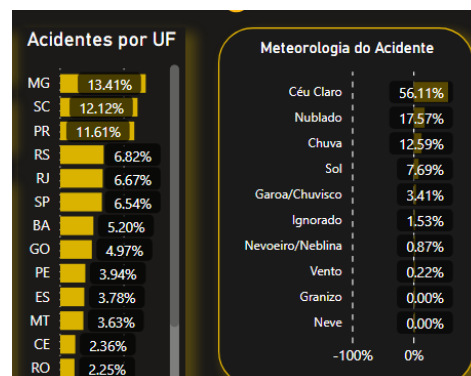


Figura 62 – Dados acidentes por UF e por Meteorologia

Acidentes por UF e Meteorologia do Acidente – Usei o gráfico de barra empilhado para Acidente por UF e o Gráfico de Barra Clusterizado para meteorologia do Acidente.

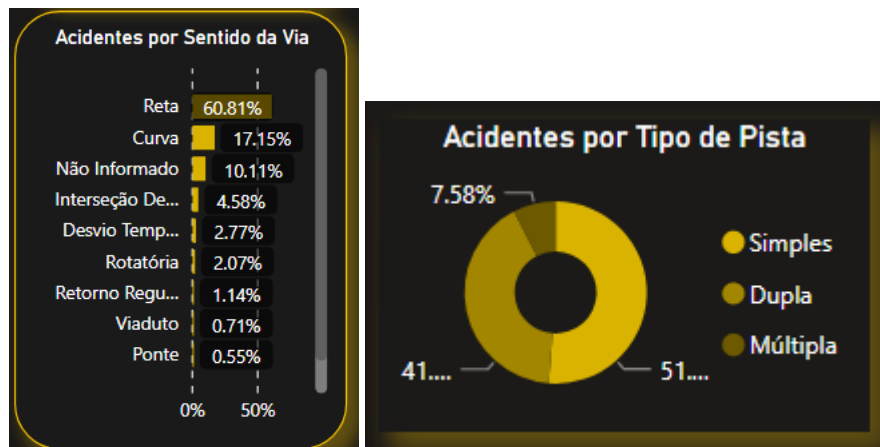


Figura 63 – Dados de acidentes por sentido da via e por tipo de pista

Finalizando os dois últimos gráficos dessa segunda tela do Dashboard que são os gráficos de Acidentes por Sentido da Via e Acidentes por Tipo de Pista. Sendo um gráfico de barra empilhadas e um gráfico de pizza.

4.4 Terceira tela do Dashboard

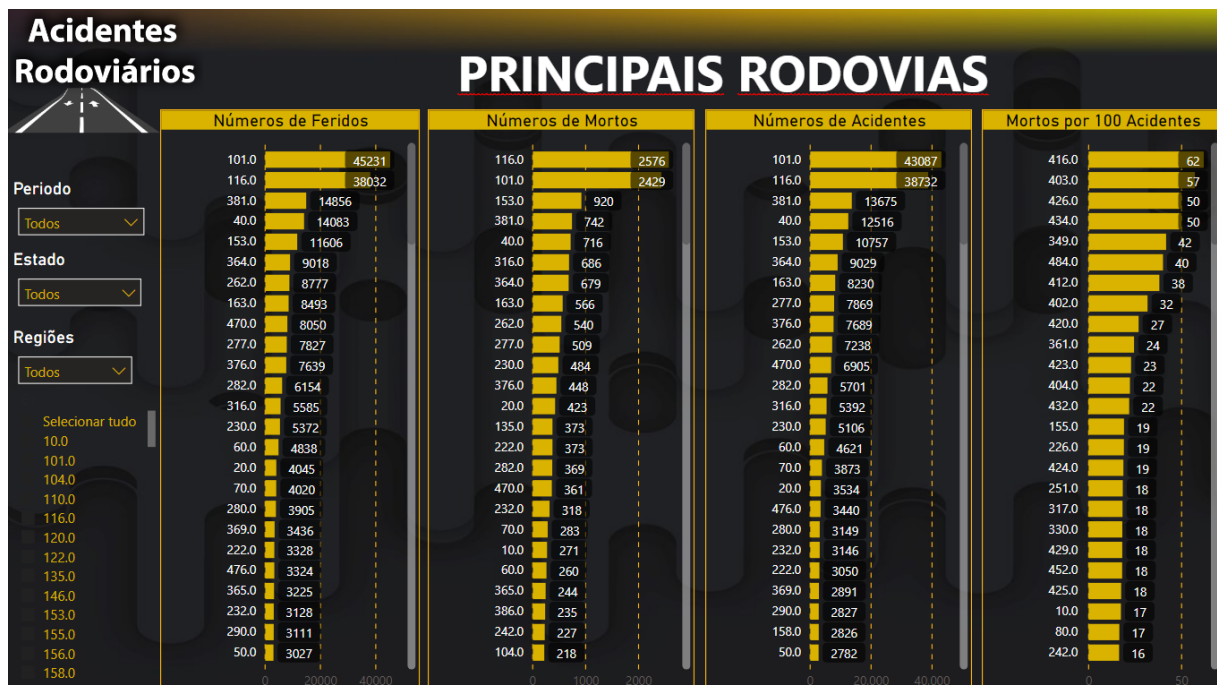


Figura 64 – Terceira tela do Dashboard com dados sobre as Rodovias

Este dashboard consiste em mostrar para ao usuário o nível de perigo que cada rodovia possui. São 4 gráficos de barra empilhados mostrando o número de

feridos por BR, número de mortos por BR , número de Acidentes por BR e número de Mortos por 100 Acidentes por BR e um filtro para selecionar por período, estado, região ou por BR que pode ser usado para comparar Diversas BRs selecionadas no filtro.

4.5. Quarta tela do Dashboard



Figura 65 – Quarta tela do Dashboard com dados sobre as Acidentes x Mortes por Local

Este dashboard consiste em mostrar para o usuário a correlação de Acidente x Mortes por Local. Aqui é possível ter a estatística dos dados por Estado, Região e Município. Eu criei um rank para os Estados, Regiões e Municípios.

4.5. Quinta tela do Dashboard



Figura 66 – Quinta tela do Dashboard com dados sobre Acidentes por dia e horário

Este dashboard consiste em mostrar para o usuário a correlação por dia e Horário Assim o usuário consegue saber quais dias e horários os tipos de acidentes são mais frequentes.

4.6. Sexta tela do Dashboard

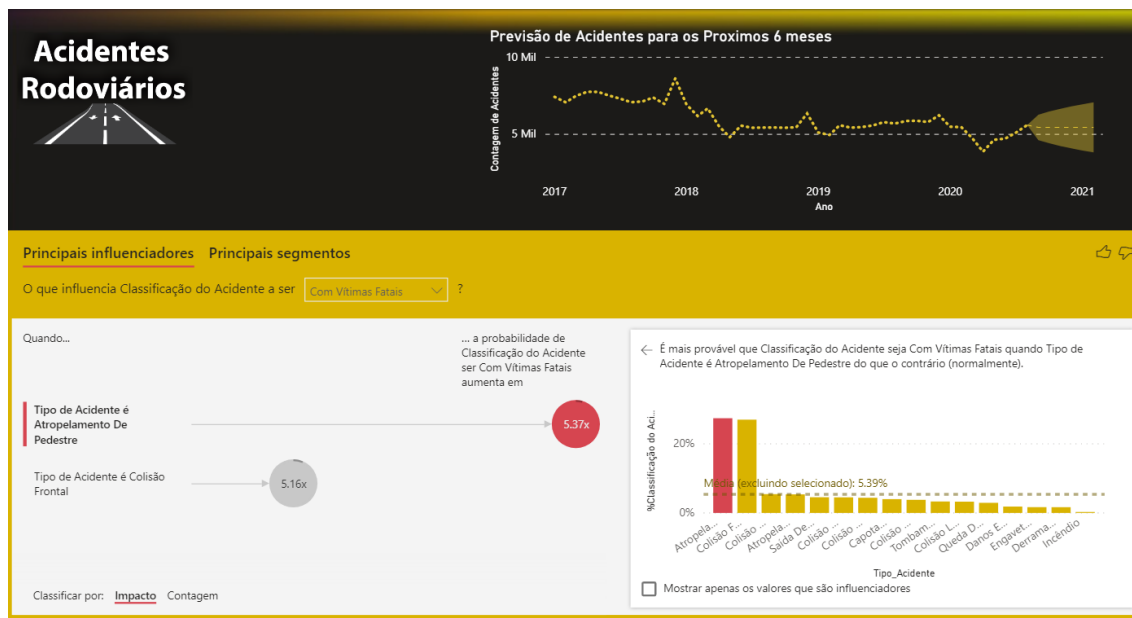


Figura 67 – Sexta tela do Dashboard com dados de Previsão e Indicadores de Acidentes

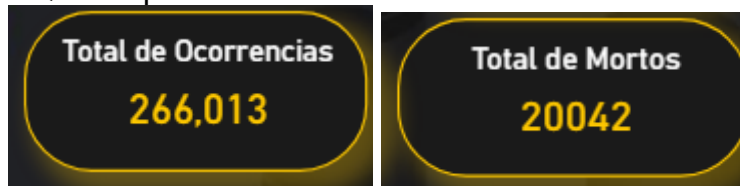
Este dashboard são indicadores e previsões usando os dados históricos desse dataset. Aqui é possível ter a previsão de acidentes dos próximos 6 meses,

além de ter uma painel com os principais influenciadores que fazem um acidente ser com vítimas fatais, sem vítimas ou com feridos leves.

5. Resultados

Após concluir toda elaboração do projeto, foi possível concluir um dashboard com dados apresentado visualmente para responder as perguntas feitas no início desse projeto.

- Qual a quantidade de ocorrências e fatalidade ao longo dos últimos anos?

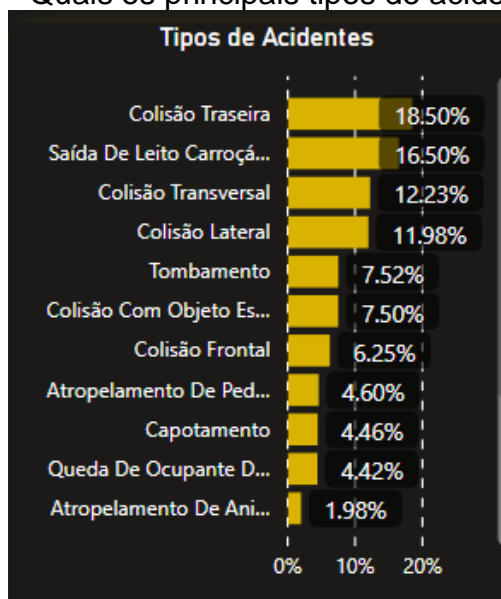


Dados de 2017 até 2020

- Quais as principais causas de acidentes?



- Quais os principais tipos de acidentes?



- Quais dias ocorrem os acidentes e quais dias são mais letais?

Dia_Semana	Mortos	% Mortos	Feridos	% Feridos	Mortos_por_100_acidentes_
Domingo	4268	21.30%	50844	17.89%	9.73
Sábado	3736	18.64%	48659	17.12%	8.47
Sexta-Feira	2842	14.18%	43100	15.16%	6.92
Segunda-Feira	2527	12.61%	38330	13.48%	7.00
Quinta-Feira	2374	11.85%	35379	12.45%	6.80
Quarta-Feira	2137	10.66%	34122	12.00%	6.41
Terça-Feira	2158	10.77%	33842	11.90%	6.61
Total	20042	100.00%	284276	100.00%	7.53

- Quais as BRs tem mais acidentes e são mais letais?



- Quais horários ocorrem os acidentes e em quais horários mais letais?

Horario	Feridos	% Feridos	Mortos	% Mortos	Mortos_por_100_acidentes_
7:00:00 PM	4317	1.52%	434	2.17%	11.13
6:30:00 PM	4231	1.49%	339	1.69%	9.13
6:00:00 PM	4251	1.50%	303	1.51%	7.86
8:00:00 PM	2798	0.98%	296	1.48%	10.77
7:30:00 PM	3029	1.07%	271	1.35%	9.45
9:00:00 PM	2335	0.82%	261	1.30%	11.41
10:00:00 PM	2061	0.72%	217	1.08%	10.80

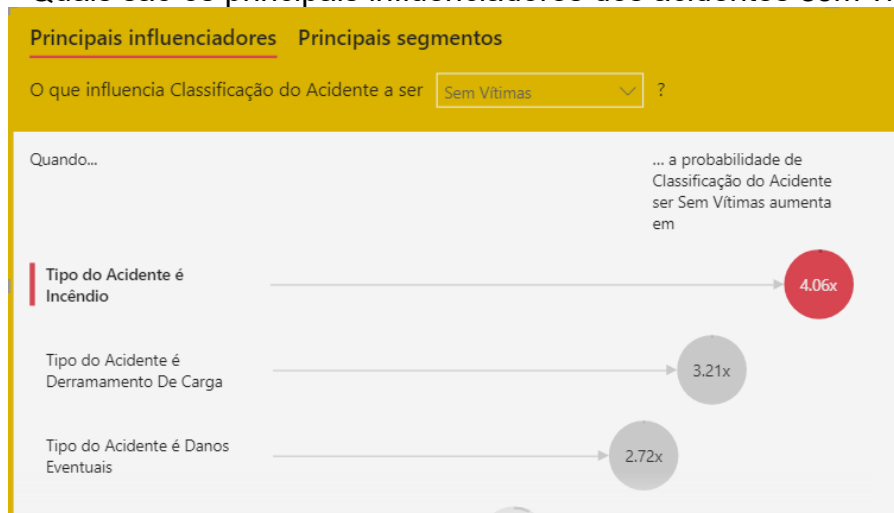
- Qual região, estado e município tem mais acidentes e quais são as mais letais?

Ranking dos Estados					
Uf	Mortos	% Mortos	Acidentes	% Acidentes	Mortos por 100 Acidentes
MG	2675	13.35%	35662	13.41%	7.50
SC	1408	7.03%	32248	12.12%	4.37
PR	1959	9.77%	30896	11.61%	6.34
RS	1160	5.79%	18138	6.82%	6.40
RJ	1135	5.66%	17740	6.67%	6.40
SP	816	4.07%	17394	6.54%	4.69
BA	1817	9.07%	13845	5.20%	13.12
GO	1065	5.31%	13209	4.97%	8.06
PE	1142	5.70%	10481	3.94%	10.90
ES	564	2.81%	10047	3.78%	5.61
MT	789	3.94%	9667	3.63%	8.16
CE	617	3.08%	6289	2.36%	9.81
RO	344	1.72%	5982	2.25%	5.75
PB	458	2.29%	5881	2.21%	7.79
Total	20042	100.00%	266013	100.00%	7.53

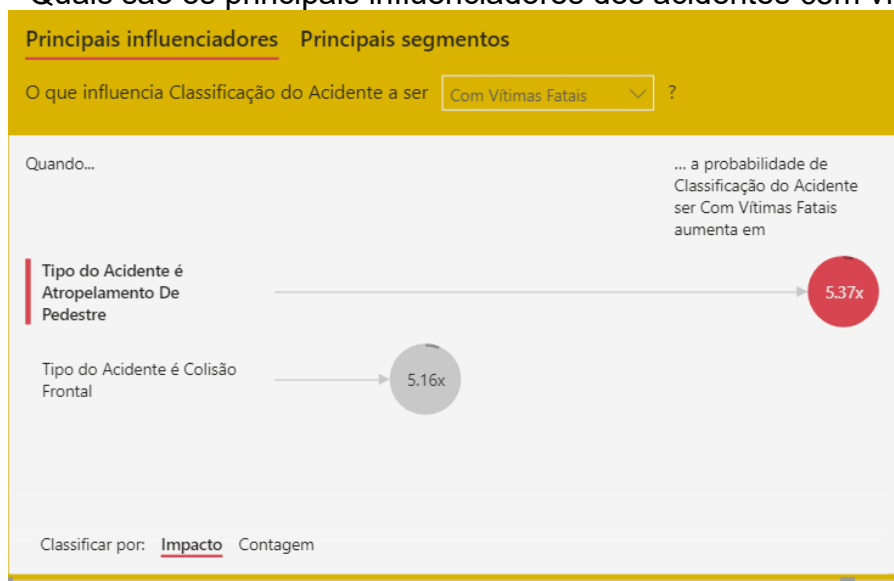
Ranking das Regiões					
Regiões	Mortos	% Mortos	Acidentes	% Acidentes	Mortos por 100 Acidentes
Sul	4527	22.59%	81282	30.56%	5.57
Sudeste	5190	25.90%	80843	30.39%	6.42
Nordeste	6365	31.76%	56123	21.10%	11.34
Centro-Oeste	2498	12.46%	32430	12.19%	7.70
Norte	1462	7.29%	15335	5.76%	9.53
Total	20042	100.00%	266013	100.00%	7.53

Ranking dos Municípios					
Município	Mortos	% Mortos	Acidentes	% Acidentes	Mortos por 100 Acidentes
Curitiba	145	0.72%	3981	1.50%	3.64
Brasília	170	0.85%	3723	1.40%	4.57
Sao Jose	45	0.22%	3035	1.14%	1.48
Guarulhos	96	0.48%	2724	1.02%	3.52
Palhoca	39	0.19%	2444	0.92%	1.60
Total	20042	100.00%	266013	100.00%	7.53

- Quais são os principais influenciadores dos acidentes sem vítimas?



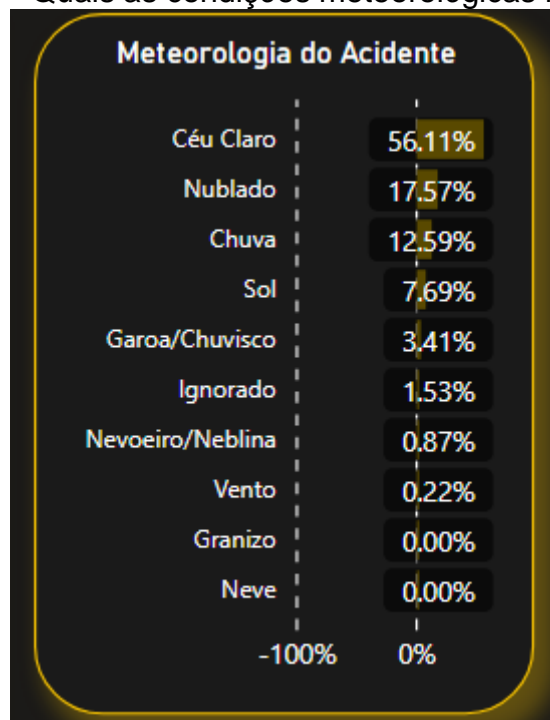
- Quais são os principais influenciadores dos acidentes com vítimas fatais?



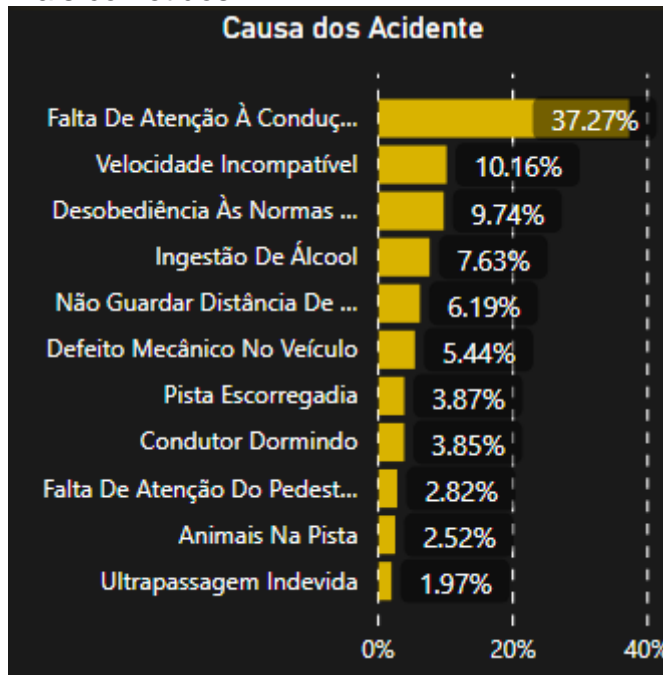
- Quais são os principais influenciadores dos acidentes com vítimas feridas?



- Quais as condições meteorológicas influenciam em mais acidentes?



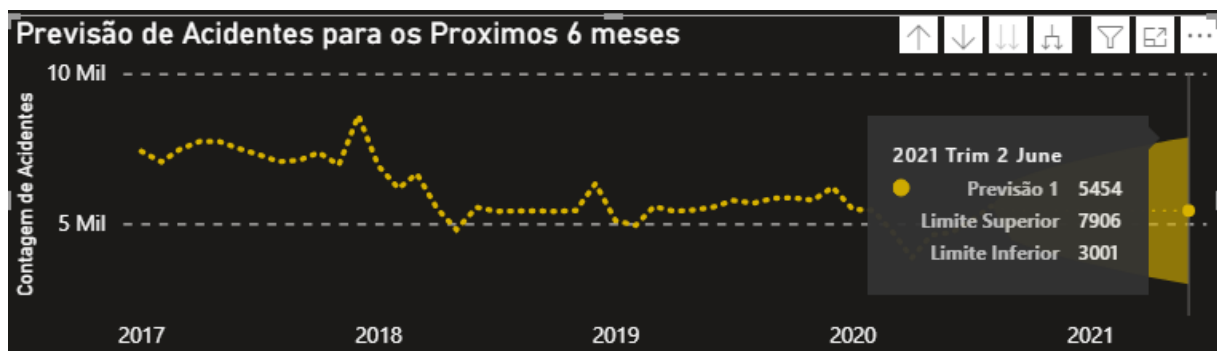
- Baseado em ocorrências onde o fator humano foi o causador, quais foram os erros mais cometidos?



- Qual a impacto da COVID-19 nos acidentes?



- Qual a previsão de acidentes para os próximos meses?



6. Links

Repositório do Projeto

- https://github.com/irlenmenezes/TCC_ciencia_de_dados

Vídeo da Previa da Apresentação

- <https://www.youtube.com/watch?v=5PIG8WDBj4U>