Acidentes Terrestres

MEGGA Consultoria

Projeto Final do Curso de Engenharia de Dados

SoulCode Academy

A Equipe

Alyne Cristina

Eveline Marques

Gustavo Santoro

Matheus Reis

Escopo do Projeto

- Uso mínimo de dois Datasets em formatos diferentes
- Armazenamento na GCP
- Procedimentos de ETL e análise realizadas através de Pandas, PySpark e SparkSQL
- Análises realizadas através do BigQuery na GCP
- Armazenamento em MySQL para Datasets brutos e MongoDB aos tratados
- Criação de um Dashboard no DataStudio e um Workflow simples

Realizar todo processo de ETL e análise dos dados utilizando stacks diversas, banco de dados e computação em nuvem.

Objetivos

Analisar dados sobre acidentes de trânsito nas rodovias federais brasileiras, encontrar as causas mais comuns associadas a esses acidentes, o perfil das vítimas e rodovias.

Dados

Encontrar uma fonte de dados relevante para o tema; Normalização dos dados

Cloud

Migração e disponibilização dos dados brutos e tratados na nuvem

Análise

A partir dos dados coletados e transformados retirar informações relevantes

Banco de Dados

Utilização de bancos de dados relacionais e não-relacionais

Ferramentas

Tecnologias

MySQL

mongoDB

python

Google Cloud

Canva

Trello

Análises

pandas

PySpark

Power BI

colab

Data Studio

jupyter

Sobre os dados

Dados obtidos através dos Registros de Ocorrência da Polícia Rodoviária Federal no período de 2017 a 2021

Pessoas

Dados voltados a cada indivíduo envolvido em uma ocorrências:

- Sexo
- Gravidade da ocorrências
- Estado físico
- Idade

Ocorrências

Dados voltados diretamente aos pormenores das ocorrência:

- Tipo da pista
- Quantidade de pessoas envolvidas
- Condições metereológicas
- Veículos envolvidos

Workflow

Web Scraping

Ocorrências PRF .zip

Pessoas PRF .zip

Datalake
Cloud Storage

Pipeline
Cloud Dataflow

Dados Brutos

Arquivos zip
Cloud Storage

pandas

- Verifica nulos e dados inconsistentes
- Remoção de coluna
- Altera tipo dos dados
- Plotagens

PySpark

- Spark Session
- StructType
- Verificação de colunas
- Novas colunas

BigQuery

Consultas com SQL
Separação em três planilhas:

- Veículos
- Pessoas
- Ocorrências

Dados Tratados

Arquivos em CSV
Cloud Storage

Dados Tratados

mongoDB

PowerBI
Insigths

DataStudio
Insights

Workflow

Datalake
Cloud Storage

Pipeline
Cloud Dataflow

Web Scraping

Ocorrências PRF .zip

Pessoas PRF .zip

Dados Brutos

Arquivos zip
Cloud Storage

pandas

- Verifica nulos e dados inconsistentes
- Remoção de coluna
- Altera tipo dos dados
- Plotagens

PySpark

- Spark Session
- StructType
- Verificação de colunas
- Novas colunas

BigQuery

Consultas com SQL
Separação em três planilhas:

- Veículos
- Pessoas
- Ocorrências

Dados Tratados

Arquivos em CSV
Cloud Storage

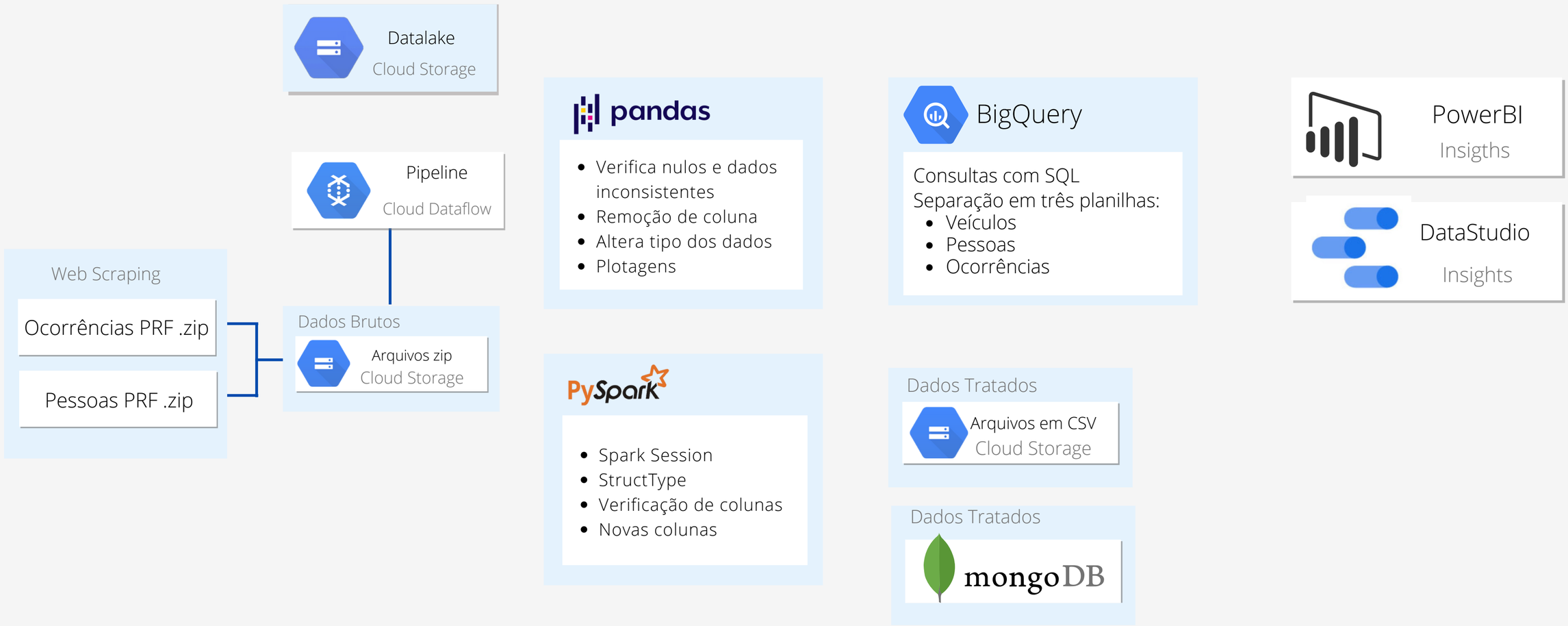
Dados Tratados

mongoDB

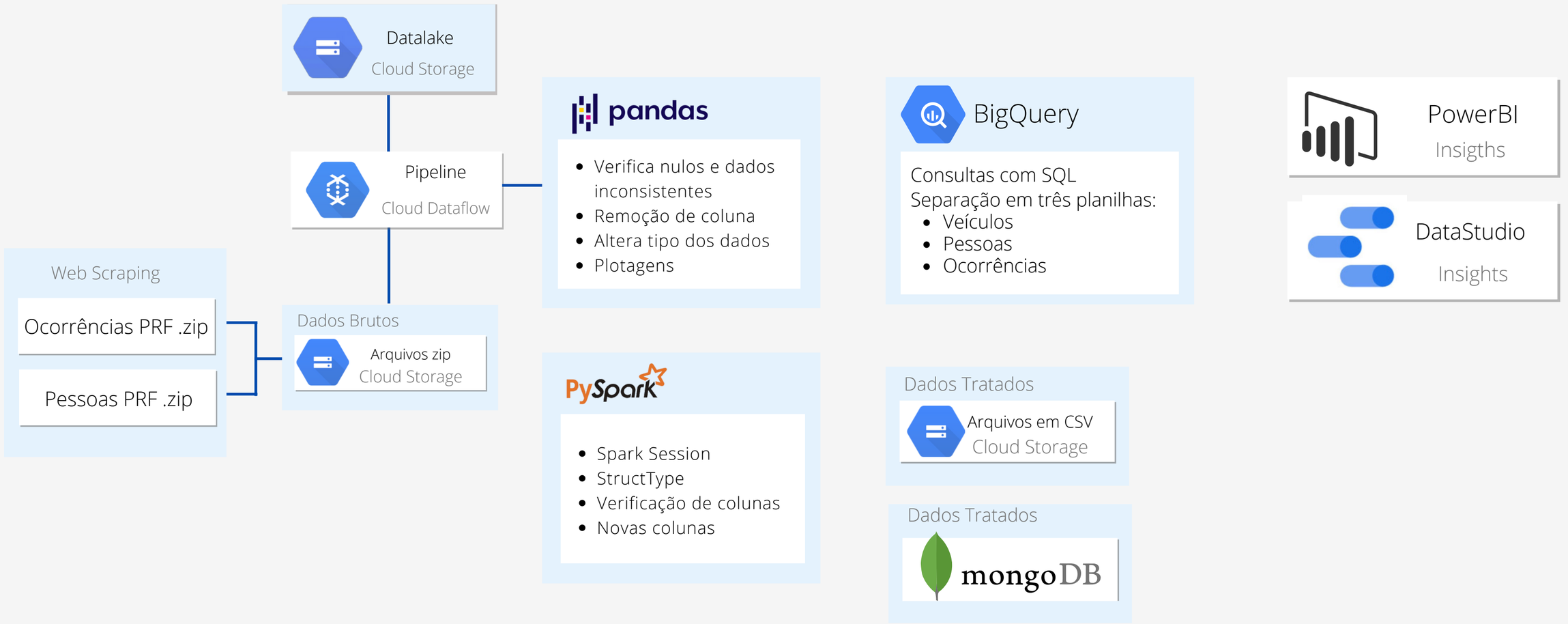
PowerBI
Insigths

DataStudio
Insights

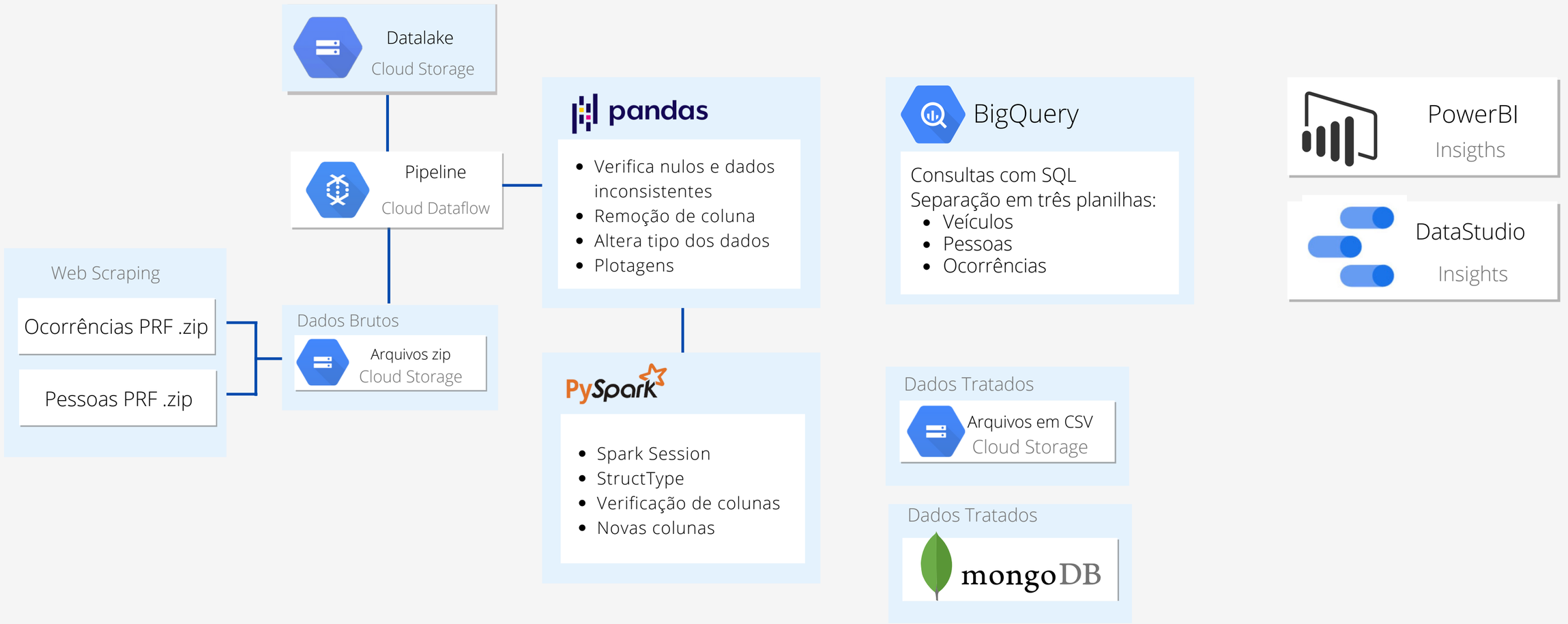
Workflow


Datalake
Cloud Storage
Pipeline
Cloud Dataflow
Web Scraping
Ocorrências PRF .zip
Pessoas PRF .zip
Dados Brutos
Arquivos zip
Cloud Storage
pandas
Verifica nulos e dados inconsistentes
Remoção de coluna
Altera tipo dos dados
Plotagens
PySpark
Spark Session
StructType
Verificação de colunas
Novas colunas
BigQuery
Consultas com SQL
Separação em três planilhas:
Veículos
Pessoas
Ocorrências
Dados Tratados
Arquivos em CSV
Cloud Storage
Dados Tratados
mongoDB
PowerBI
Insigths
DataStudio
Insights

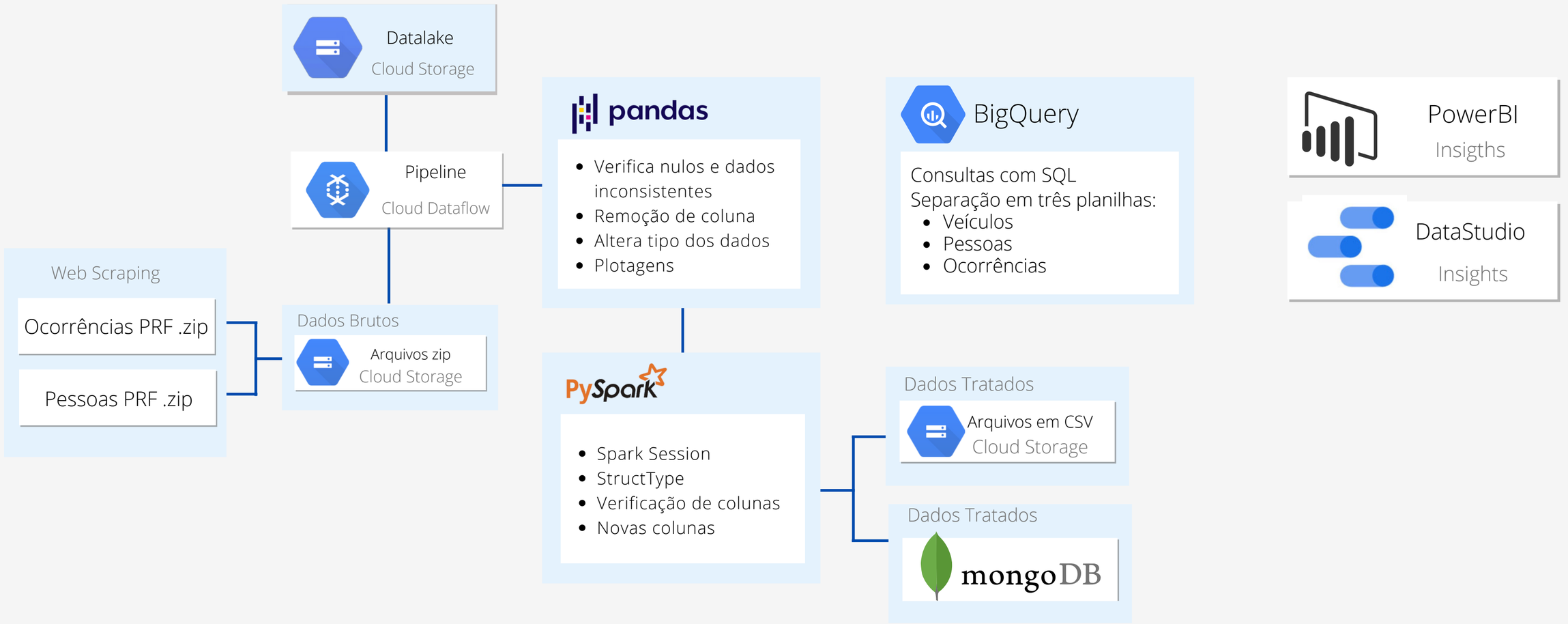
Workflow


Datalake
Cloud Storage
Pipeline
Cloud Dataflow
Web Scraping
Ocorrências PRF .zip
Pessoas PRF .zip
Dados Brutos
Arquivos zip
Cloud Storage
pandas
Verifica nulos e dados inconsistentes
Remoção de coluna
Altera tipo dos dados
Plotagens
PySpark
Spark Session
StructType
Verificação de colunas
Novas colunas
BigQuery
Consultas com SQL
Separação em três planilhas:
Veículos
Pessoas
Ocorrências
Dados Tratados
Arquivos em CSV
Cloud Storage
Dados Tratados
mongoDB
PowerBI
Insigths
DataStudio
Insights

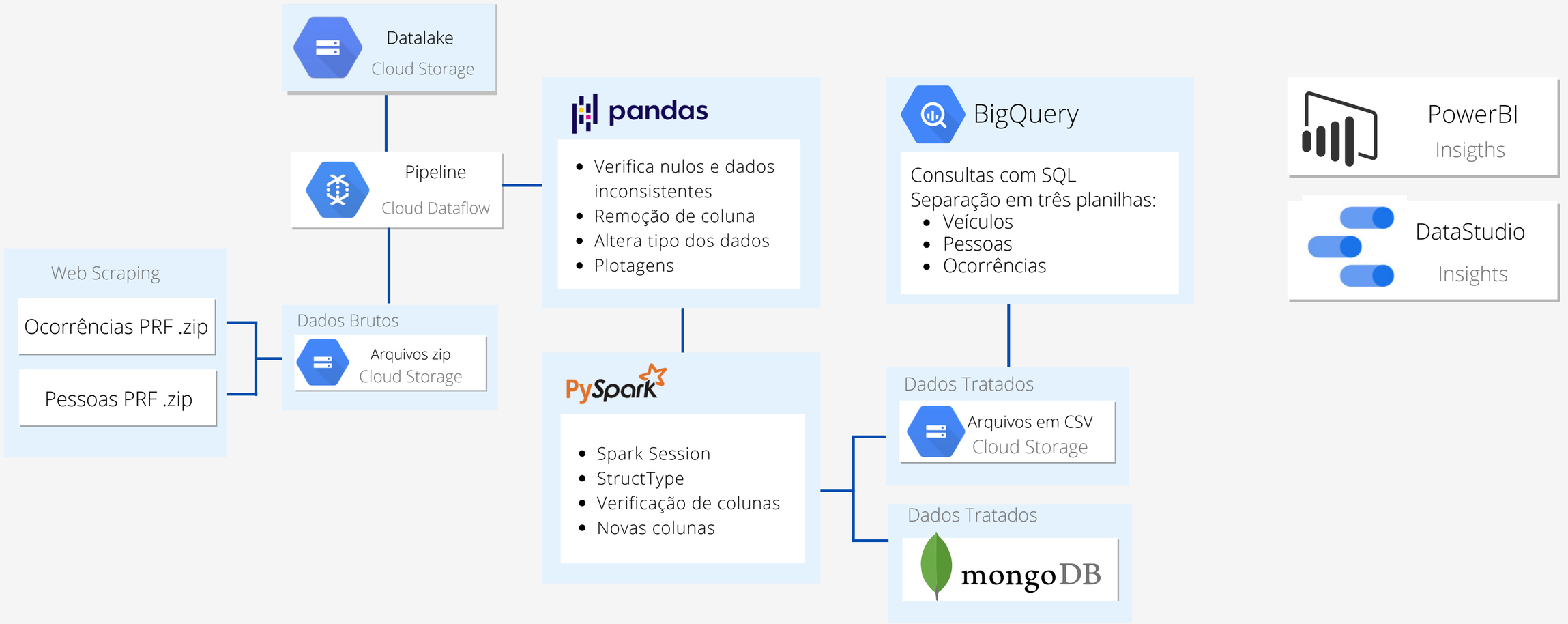
Workflow


Datalake
Cloud Storage
Pipeline
Cloud Dataflow
Web Scraping
Ocorrências PRF .zip
Pessoas PRF .zip
Dados Brutos
Arquivos zip
Cloud Storage
pandas
Verifica nulos e dados inconsistentes
Remoção de coluna
Altera tipo dos dados
Plotagens
PySpark
Spark Session
StructType
Verificação de colunas
Novas colunas
BigQuery
Consultas com SQL
Separação em três planilhas:
Veículos
Pessoas
Ocorrências
Dados Tratados
Arquivos em CSV
Cloud Storage
Dados Tratados
mongoDB
PowerBI
Insigths
DataStudio
Insights

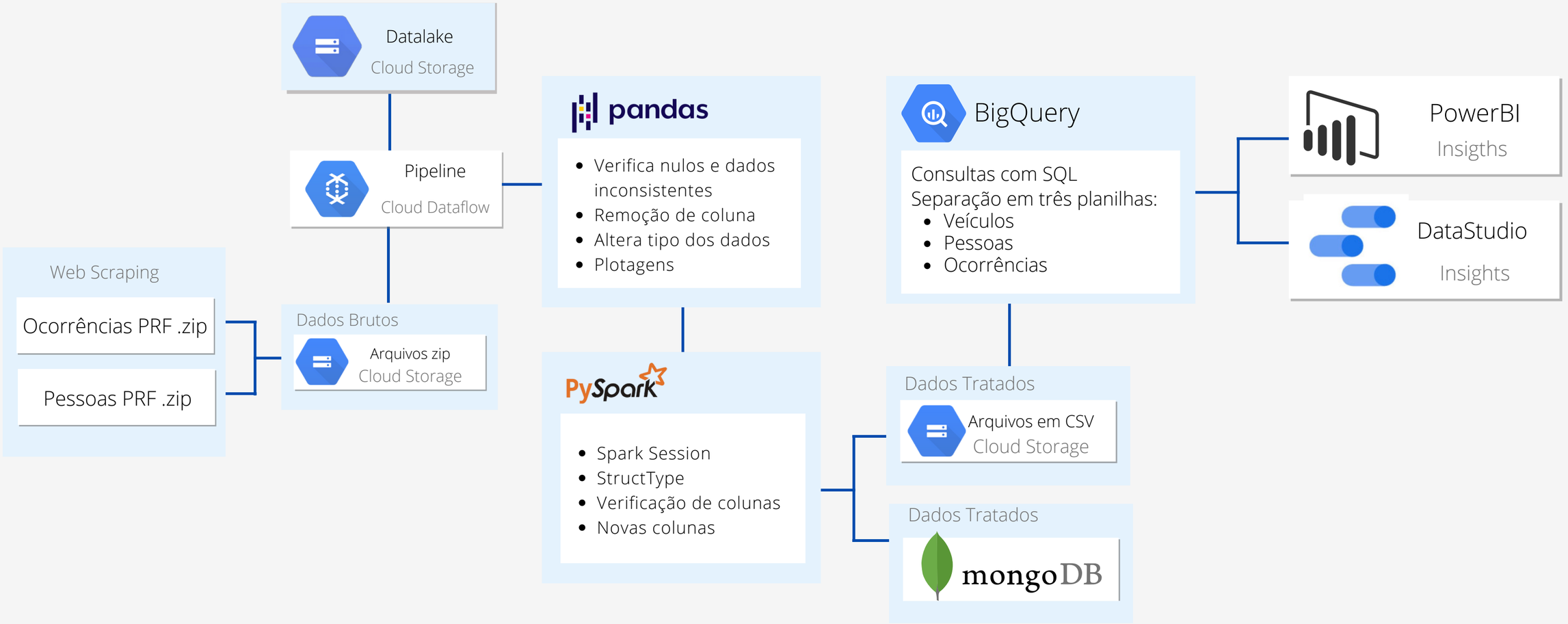
Workflow


Datalake
Cloud Storage
Pipeline
Cloud Dataflow
Web Scraping
Ocorrências PRF .zip
Pessoas PRF .zip
Dados Brutos
Arquivos zip
Cloud Storage
pandas
Verifica nulos e dados inconsistentes
Remoção de coluna
Altera tipo dos dados
Plotagens
PySpark
Spark Session
StructType
Verificação de colunas
Novas colunas
BigQuery
Consultas com SQL
Separação em três planilhas:
Veículos
Pessoas
Ocorrências
Dados Tratados
Arquivos em CSV
Cloud Storage
Dados Tratados
mongoDB
PowerBI
Insigths
DataStudio
Insights

Workflow


Datalake
Cloud Storage
Pipeline
Cloud Dataflow
Web Scraping
Ocorrências PRF .zip
Pessoas PRF .zip
Dados Brutos
Arquivos zip
Cloud Storage
pandas
Verifica nulos e dados inconsistentes
Remoção de coluna
Altera tipo dos dados
Plotagens
PySpark
Spark Session
StructType
Verificação de colunas
Novas colunas
BigQuery
Consultas com SQL
Separação em três planilhas:
Veículos
Pessoas
Ocorrências
Dados Tratados
Arquivos em CSV
Cloud Storage
Dados Tratados
mongoDB
PowerBI
Insigths
DataStudio
Insights

Workflow


Datalake
Cloud Storage
Pipeline
Cloud Dataflow
Web Scraping
Ocorrências PRF .zip
Pessoas PRF .zip
Dados Brutos
Arquivos zip
Cloud Storage
pandas
Verifica nulos e dados inconsistentes
Remoção de coluna
Altera tipo dos dados
Plotagens
PySpark
Spark Session
StructType
Verificação de colunas
Novas colunas
BigQuery
Consultas com SQL
Separação em três planilhas:
Veículos
Pessoas
Ocorrências
Dados Tratados
Arquivos em CSV
Cloud Storage
Dados Tratados
mongoDB
PowerBI
Insigths
DataStudio
Insights

Extração de Dados Brutos e Upload GCS

scrap/scrap_upload.ipynb
scrap/GCS.py

```
# Abre o link, retira HTML e filtra somente a tag HTML desejada
# as tags de interesse são <a href class='external-link' para o ano e link do arquivo
# e h2 para o tipo de arquivo Ocorrências ou Pessoas
try:
    html=urlopen("https://www.gov.br/prf/pt-br/acesso-a-informacao/dados-abertos/dados-abertos-acidentes")
    bs=BeautifulSoup(html.read(), 'html.parser')
    links=bs.findAll('a', {'class':'external-link'})
    listagrupos=bs.findAll('h2')
except (URLError, HTTPError) as e:
    print('Não foi possivel ler o site', e)

```

```
def upload(bucket,ano,planilha, caminho):
    '''Função que faz o download dos arquivos .zip e upload para a bucket'''
    # bucket = "nome-da-nova-bucket"
    # ano = a partir de que ano será o download
    # planilha = qual planilha: Pessoas ou Ocorrências
    # caminho = define em quais pastas será salvo na GCS

    for i,j in arquivos.items():
        if(i==planilha):
            for k, v in j.items():
                if(int(k)>=ano and int(k)<2022):
                    if(int(k)==2021):
                        v=v+'/download'
                    arquivoslocal= wget.download(v)
                    upload_objeto(bucket,arquivolocal,caminho+filename)
                    os.remove(filename)
```

```
#Define a pasta dadosbrutos para envio dos arquivos
# define a bucket para envio
#cria objeto é função do módulo GCS.py
caminho="dadosbrutos/pessoas/"
bucket='projetofinalsc'
cria_bucket(bucket)
upload(bucket,2017,'Agrupados por pessoa', caminho)
```

Pipeline

pipeline/pipelineCSV.ipynb
pipeline/pipelineJSON.ipynb

```
def run():
    #Lê arquivo zipado, descomprime e separa por linhas
    def unzip(readable_file):
        import zipfile
        from zipfile import ZipFile
        arq_zip=zipfile.ZipFile(readable_file.open('r'))
        nome=arq_zip.namelist()
        arquivo=arq_zip.read(nome[0])
        arquivo.decode(encoding='iso-8859-1').split('\n')
        bytearray(b)
        yield b[363:] # tira o header
    #informações para rodar pipeline no DataFlow
    argv = [
      '--project={0}'.format(project),
      '--region=us-central1',
      '--staging_location=gs://{0}/staging/'.format(bucket),
      '--temp_location=gs://{0}/staging/'.format(bucket),
      '--runner=DataflowRunner',
      '--template_location:gs://{0}/model/'.format(bucket)]

    p = beam.Pipeline(argv=argv)
    (p
      | 'Procura arquivo' >> beam.io.fileio.MatchFiles('gs://projetofinalsc/dadosbrutos/ocorrencias/datatran20*.zip')
      | 'Encontra os targets' >> beam.io.fileio.ReadMatches()
      | 'Unzipa' >> beam.FlatMap(unzip)
      | 'Escreve arquivo' >> beam.io.WriteToText('gs://projetofinalsc/dadosbrutos/ocorrencias/ocorrencias', file_name_suffix='.csv'))

    p.run()
```

MySQL

pipeline/mysql.ipynb

```
import os
import mysql.connector
from google.cloud import storage
from mysql.connector import Error
from google.cloud import storage
os.environ["GOOGLE_APPLICATION_CREDENTIALS"]="sca-at-b5179177a346.json"
```

```
try:
    mydb=msql.connect(host='35.192.50.164', user='user',
                      password="senha")
    if mydb.is_connected():
        cursor=mydb.cursor()
        cursor.execute("CREATE DATABASE acidentes")
        print("Database foi criado")
except Error as e:
    print("Sem conexão com MySQL", e)
```

```
storage_client = storage.Client()
bucket=storage_client.get_bucket('acidentes_terrestres')
arquivo=bucket.blob('dados_tratados/acidente_ocorrencia.csv')
blob=arquivo.download_as_string()
blob = blob.decode('8859-1')
blob = StringIO(blob)
names=csv.reader(blob)
```

```
query="""CREATE TABLE acidentes.ocorrencias(id INT NOT NULL,data VARCHAR(45) NULL,dia_semana VARCHAR(45) NULL,uf VARCHAR(45) NULL,
  rodovia VARCHAR(45) NULL, km VARCHAR(45) NULL,municipio VARCHAR(45) NULL,causa_acidente VARCHAR(300) NULL,tipo_acidente VARCHAR(45) NULL,
  classificacao_acidente VARCHAR(45) NULL,fase_dia VARCHAR(45) NULL,sentido_via VARCHAR(45) NULL,condicao_metereologica VARCHAR(45) NULL,
  tipo_pista VARCHAR(45) NULL,tracado_via VARCHAR(45) NULL,uso_solo VARCHAR(45) NULL,pessoas VARCHAR(45) NULL,
  mortos VARCHAR(45) NULL,feridos_leves VARCHAR(45) NULL,feridos_graves VARCHAR(45) NULL,ilesos VARCHAR(45) NULL,
  ignorados VARCHAR(45) NULL,feridos VARCHAR(45) NULL,veículos VARCHAR(45) NULL,latitude VARCHAR(45) NULL,
  longitude VARCHAR(45) NULL, PRIMARY KEY (id))
DEFAULT CHARACTER SET = latin1;"""
cursor.execute(query)
```

Tratamentos

pandas/ocorrencias.ipynb
pandas/pessoas.ipynb

```
#Corrigindo a coluna Dia Semana
df['dia_semana'].replace(['sexta'],'sexta-feira',inplace=True)
df['dia_semana'].replace(['segunda'],'segunda-feira',inplace=True)
df['dia_semana'].replace(['quinta'],'quinta-feira',inplace=True)
df['dia_semana'].replace(['quarta'],'quarta-feira',inplace=True)
df['dia_semana'].replace(['terça'],'terça-feira',inplace=True)
```

Transforma latitude e longitude para float

```
df['data_inversa']=pd.to_datetime(df['data_inversa'])
df['ano']=pd.DatetimeIndex(df['data_inversa']).year
df.loc[df['ano']<2021, 'latitude'] = df['latitude'].str.replace(',','.')
df.loc[df['ano']<2021, 'longitude'] = df['longitude'].str.replace(',','.')
df=df.drop('ano', axis=1)
df['latitude']=df['latitude'].astype(float)
df['longitude']=df['longitude'].astype(float)
```

```
filtro1=df['latitude']==0
filtro2=df['longitude']==0
df[filtro1&filtro2]
```

Transformação de 'km' para float

```
df['km']=df['km'].str.replace(',','.')
df['km']=df['km'].astype(float)
```

```
#Corrigindo coluna Sexo
df['sexo'].replace(['Ignorado'],'Não Informado',inplace=True)
```

PySpark

pyspark/ocorrencias.ipynb
pyspark/pessoas.ipynb

Cria coluna que diz relaciona a causa do acidente ao consumo de álcool

```
df=df.withColumn("consumo_alcool", F.col('causa_acidente').rlike("Álcool"))
df=df.withColumn('consumo_alcool', when(df.consumo_alcool=='true', 'sim').otherwise('não'))
```

4.3 Novas coluna

```
#Criando uma nova coluna de faixa etária
dfs=dfs.withColumn('faixa_idade', F.when((col('idade')>100), lit('Não Informado'))
                                  .when((col('idade')>85), lit('85+'))
                                  .when((col('idade')>=80), lit('80-84'))
                                  .when((col('idade')>=75), lit('75-79'))
                                  .when((col('idade')>=70), lit('70-74'))
                                  .when((col('idade')>=65), lit('65-69'))
                                  .when((col('idade')>=60), lit('60-64'))
                                  .when((col('idade')>=55), lit('55-59'))
                                  .when((col('idade')>=50), lit('50-54'))
                                  .when((col('idade')>=45), lit('45-49'))
                                  .when((col('idade')>=40), lit('40-44'))
                                  .when((col('idade')>=35), lit('35-39'))
                                  .when((col('idade')>=30), lit('30-34'))
                                  .when((col('idade')>=25), lit('25-29'))
                                  .when((col('idade')>=20), lit('20-24'))
                                  .when((col('idade')>=16), lit('16-19'))
                                  .when((col('idade')>=11), lit('11-15'))
                                  .when((col('idade')<=10), lit('00-10')))
dfs.select('faixa_idade', 'idade').show(100)
```

Análises Pandas

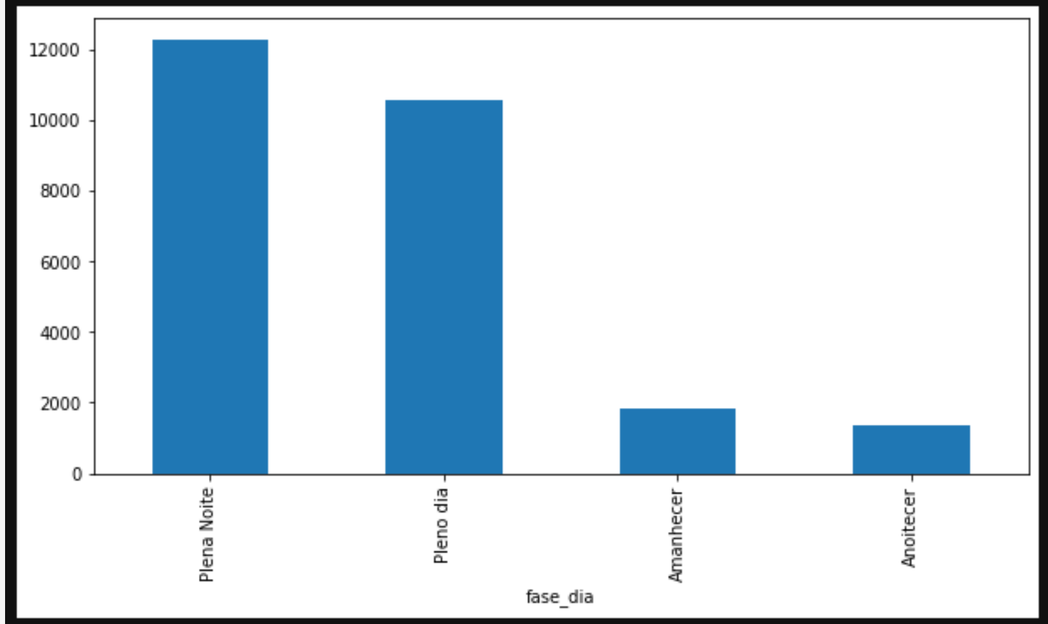
pyspark/ocorrencias.ipynb
pyspark/pessoas.ipynb

```
filtro6=uniao['estado_fisico']==  'Óbito'
```

```
obitos=uniao.loc[filtro6]
```

```
obitos.groupby(['fase_dia']).size().sort_values(ascending=False).plot.bar(figsize=(10,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc998f5e90>
```



Embora se tenha mais acidentes durante o dia, a noite eles tendem a ser mais letais, devido a fatores como baixa iluminação, condição do motorista, falta de sinalização

SparkSQL

6. SparkSQL

```
#Criando a tabela temporaria que será utilizada pelo SQL
dfs.createOrReplaceTempView('pessoas')
```

```
#QUANTIDADE DE ACIDENTES AGRUPADO POR ESTADO FISICO, SEXO
spark.sql("SELECT estado_fisico AS Nivel_Lesao, sexo, count(DISTINCT id) AS Numero_Acidentes FROM pessoas  GROUP BY estado_fisico, sexo ORDER BY estado_fisico ASC").show()
```

```
#QUANTIDADE DE ACIDENTES AGRUPADO POR SEXO
spark.sql("SELECT estado_fisico AS Nivel_Lesao, sexo, count(DISTINCT id) AS Numero_Acidentes FROM pessoas GROUP BY estado_fisico, sexo ORDER BY estado_fisico ASC").show()
```

```
#QUANTIDADE DE ACIDENTES AGRUPADO POR TIPO DE AUTOMOVEL E ESTADO FISICO
spark.sql("SELECT estado_fisico AS Nivel_Lesao, sexo, count(DISTINCT id) AS Numero_Acidentes FROM pessoas  GROUP BY estado_fisico, sexo ORDER BY estado_fisico ASC").show()
```

```
df.createOrReplaceTempView('ocorrencias')
```

```
spark.sql('SELECT causa_grupos, COUNT(id) FROM  ocorrencias GROUP BY causa_grupos ORDER BY COUNT(id)' ).show()
```

```
spark.sql('SELECT dia_semana, count(id) from ocorrencias GROUP BY dia_semana ORDER BY count(id) desc' ).show()
```

MongoDB

pandas/mongo.ipynb

5. Mongo

5.1 Instalando e configurando o MongoDB no Colab

```
pip install pymongo[srv]
```

```
import pymongo
from pymongo import MongoClient
```

```
client = pymongo.MongoClient("mongodb+srv://soulcode:<senha>@cluster0.9jrfe.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")
```

5.2 Enviando DataFrame para o Mongo DB

```
#Setando o DB e a coleção
db = client['']
collection=db['']
```

```
#Modificando algumas estrutras para que seja aceita pelo mongo
uniao2=uniao.toPandas()
uniao2['Data']=uniao2['Data'].astype(str)
```

```
#convertendo DataFrame em dicionário

uniao2.reset_index(inplace=True)

data_dict = uniao2.to_dict("records")
```

```
#Inserindo informações finalmente
collection.insert_many(data_dict)
```

BigQuery

Query

EDIT QUERY

```sql
--Nº DE ÓBITOS POR GENERO E DIA DA SEMANA--
SELECT estado_fisico AS Nivel_Lesao, sexo, dia_semana, count(id_pessoa) AS Numero_Acidentes
FROM `sca-at.projeto_final.acidentes_pessoas`
WHERE estado_fisico = 'Óbito'
GROUP BY estado_fisico, sexo, dia_semana
ORDER BY COUNT (id_pessoa) DESC
```

Query

EDIT QUERY

```sql
--Nº DE ACIDENTES POR ESTADO AGRUPADO PELA FASE DO DIA--
SELECT fase_dia, uf AS Estado, count(id_pessoa) AS Acidentes
FROM `sca-at.projeto_final.acidentes_pessoas`
GROUP BY fase_dia, uf
ORDER BY count(id_pessoa) DESC
```

Query

EDIT QUERY

```sql
--Nº DE ACIDENTES POR TIPO ENVOLVIDO E NÍVEL DE LESAO--
SELECT tipo_envolvido as Envolvido, estado_fisico as Nivel_Lesao, sexo AS Genero, count(id_pessoa) AS Acidentes
FROM `sca-at.projeto_final.acidentes_pessoas`
GROUP BY tipo_envolvido, estado_fisico, sexo
ORDER BY count(id_pessoa) DESC
```

BigQuery

Info_veiculos

QUERY SHARE COPY

SCHEMA DETAILS PREVIEW

Filter Enter property name or value

Field name	Type	Mode	Policy tags
id_veiculo	INTEGER	NULLABLE	
id	INTEGER	NULLABLE	
tipo_veiculo	STRING	NULLABLE	
marca_modelo	STRING	NULLABLE	
ano_fabricacao_veiculo	INTEGER	NULLABLE	

Info_ocorrencias

QUERY SHARE COPY

SCHEMA DETAILS PREVIEW

Filter Enter property name or value

Field name	Type	Mode	Policy tags
id	INTEGER	NULLABLE	
data	DATE	NULLABLE	
dia_semana	STRING	NULLABLE	
uf	STRING	NULLABLE	
municipio	STRING	NULLABLE	
rodovia	INTEGER	NULLABLE	
km	FLOAT	NULLABLE	
latitude	FLOAT	NULLABLE	
longitude	FLOAT	NULLABLE	

Info_pessoas

QUERY SHARE COPY

SCHEMA DETAILS PREVIEW

Filter Enter property name or value

Field name	Type	Mode	Policy tags
id_pessoa	INTEGER	NULLABLE	
id	INTEGER	NULLABLE	
tipo_envolvido	STRING	NULLABLE	
estado_fisico	STRING	NULLABLE	
sexo	STRING	NULLABLE	
faixa_idade	STRING	NULLABLE	

Análises

- As BR's com maior número de acidentes são as 101 e 116, ambas na costa leste do Brasil (Litoral)
- Visível aumento de acidentes em períodos de férias e datas festivas.
- Visível queda de acidentes logo no início da pandemia
- A maior parcela das causas de acidentes são por falha humana
- Quanto mais perto do fim de semana, maior a quantidade de acidentes

Google Data Studio

Power BI

DataStudio

PowerBI

Custo do Projeto

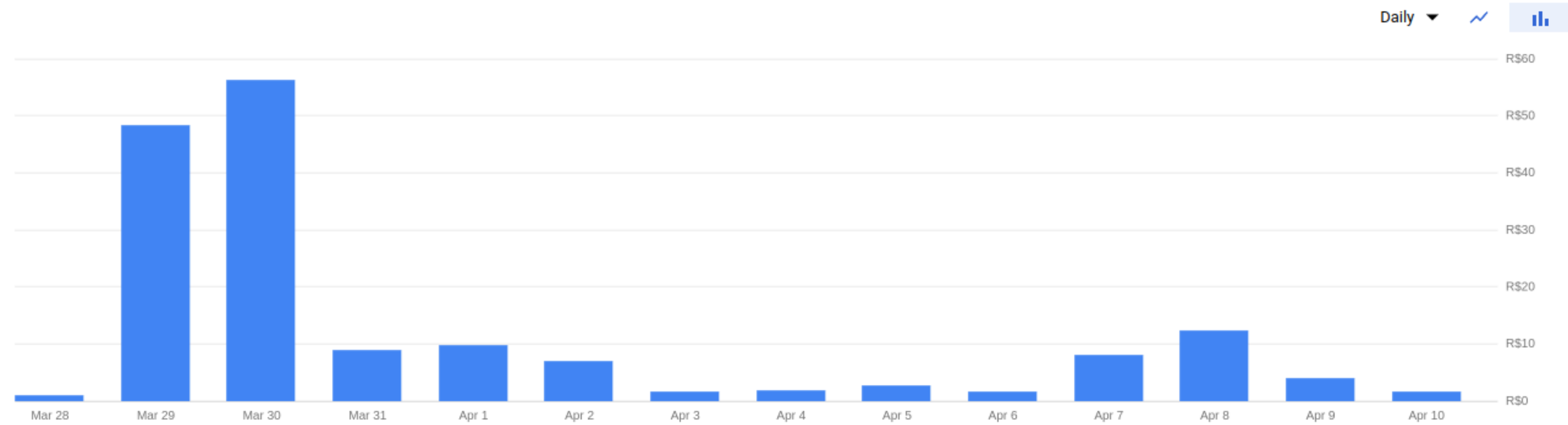
28 March – 10 April 2022 (total cost)

R$164.54

includes R$0.00 in credits

↑ –

R$164.54 over 14–27 March 2022



Project	Project ID	Project number	Cost	Discounts	Promotions and others	↓ Subtotal
SCA - AT	sca-at	735975141052	R$164.54	–	–	R$164.54

Resumo

Através das ferramentas em nuvem, uma codificação eficiente e objetividade nos procedimentos, podemos realizar uma análise e refinamento dos dados que foram cedidos. Desse ponto em diante as informações estão mais claras, o que facilita a tomada de decisões.

Os custos derivados desse processo são ínfimos em comparação ao tempo que seria necessário para manipulação e tratamento dos mesmos Datasets em uma metodologia convencional.

Ao usarmos métodos de automatização de processos como as pipelines aumentamos ainda mais eficiência e poupamos tempo de atualizações mecânicas dos Dataframes analisados.

Contatos

Gustavo Santoro

 gusantoro

Matheus Reis

 reismatheus1999

Alyne Cristina

 alynecjes

Eveline Marques

 evelinemg