

PROJETO FINAL



Curso: Engenharia de Dados
Turma: BC17

EQUIPE



Lucas David



Felipe Costa



Tiago Ferreira



Jefferson Costa



ÍNDICE



- 1. OBJETIVO**
- 2. ESCOPO DO PROJETO**
- 3. FERRAMENTAS UTILIZADAS**
- 4. METODOLOGIA**
- 5. WORKFLOW**
- 6. RESULTADO**

OBJETIVO



- **Demonstrar a principal causa de óbitos ocorridas no período de 2017 até 2020;**
- **Demonstrar a variação dos gastos do Governo Federal com saúde no mesmo período;**

ESCOPO DO PROJETO



- Converter, normalizar e tratar dados de dataset proposto pela SoulCode utilizando Pandas e PySpark, com a descrição de cada operação realizada;
- Armazenar dataset original em banco de dados SQL e utilizar triggers e procedures;
- Utilizar banco de dados NoSQL como um data Lake, importando os dataframes resultantes para um coleção do mongoDb Atlas;
- Salvar e operar os datasets em armazenamento cloud na plataforma GCP;
- Criação de Workflow com as etapas de realização do ETL e ferramentas utilizadas;
- Análises no Big Query utilizando a linguagem SQL;
- Armazenar os dados tratados em data Lake (GStorage) e/ou DW (BigQuery);
- Realizar operações utilizando o SparkSQL;
- Criação de dashboard no Datastudio trazendo insights importantes;
- Entregar todos os scripts.

FERRAMENTAS UTILIZADAS

Programação



Nuvem



Organização



FERRAMENTAS UTILIZADAS

E



T



L



METODOLOGIA

- Trabalho colaborativo, onde todos participam de maneira igualitária, porém com o foco nas qualidades individuais;
- Reunião inicial (kick-off) para definição dos passos a serem seguidos;
- Reuniões diárias (daily) da orientadora com a equipe para medição do progresso;
- Reuniões diárias da equipe (15 minutos) para alinhamento das expectativas e reorganização das tarefas;
- Apresentação da evolução do projeto entre os membros da equipe e discussão sobre alterações necessárias;

WORKFLOW

**CSV e Json
originais**



CSV e Json originais

DATASET PRINCIPAL - DESCRIÇÃO

- Sistema de Informação sobre Mortalidade (SIM);
- Desenvolvido pelo Ministério da Saúde em 1975;
- Produto da unificação de mais de quarenta modelos de Declaração de Óbito utilizados ao longo dos anos;
- Função de coletar dados sobre mortalidade no país.



CSV e Json originais

DATASET PRINCIPAL - CARACTERÍSTICAS

- 1.581.645 linhas
- 87 colunas
- Ano: 2020

```
spark_2020.show(20)
```

contador	ORIGEM	TIPOBITO	DTOBITO	HORAOBITO	NATURAL	CODMUNNATU	DTNASC	IDADE	SEXO	RACACOR	ESTCIV	ESC	ESC2010	SERIESCFAL	OCUP	CODMUNRES	LOCOCOR	CODESTAB	C
1	1	2	25012020	421	831	310160	10051951	468	1	1	4	2	1	3	999993	316930	3	null	
2	1	2	25012020	1910	831	316930	16031966	453	1	2	1	2	1	1	621005	316930	3	null	
3	1	2	14012020	1727	835	351360	10071937	482	2	1	2	2	1	1	999992	316930	1	2760657	
4	1	2	20012020	1930	831	313870	2111946	473	2	1	4	2	1	3	999992	316080	1	2760657	
5	1	2	26012020	2230	831	311090	12091959	460	1	1	1	5	5	null	231205	311390	1	2760657	
6	1	2	27012020	1626	831	316930	8051964	455	1	4	5	2	1	1	783225	316930	1	2760657	
7	1	2	12012020	1515	823	230580	17031973	446	2	4	2	3	1	4	612005	230423	3	null	
8	1	2	11012020	1300	835	350160	15061974	445	1	1	2	5	5	null	null	510520	4	null	
9	1	2	11012020	1630	831	316930	10051943	476	1	2	2	2	1	1	999993	316930	1	2760657	
10	1	2	15012020	1115	831	316930	13012020	202	1	4	null	null	null	null	null	316930	1	2760657	
11	1	2	12012020	942	831	311390	9091934	485	1	1	2	3	2	7	999993	316930	1	2760657	

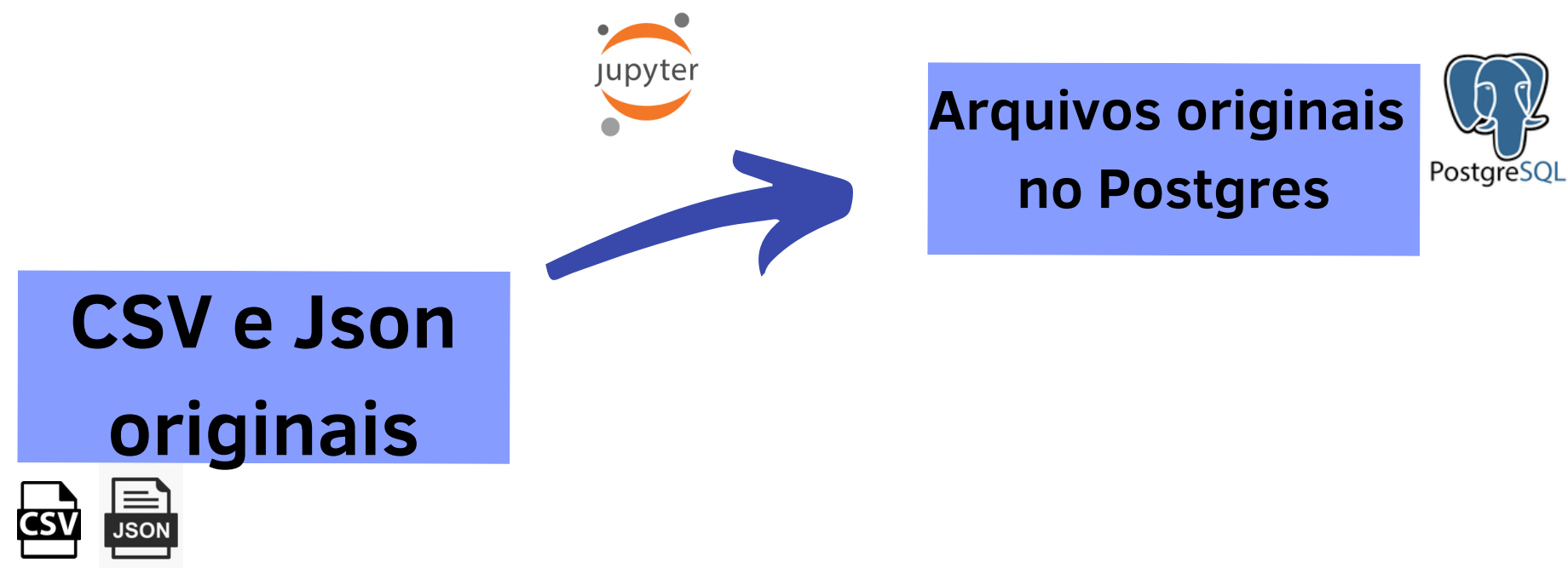
CSV e Json originais

ARQUIVOS ADICIONAIS

- **SIM 2017;**
- **SIM 2018;**
- **SIM 2019;**
- **Classificação Internacional de Doenças – CID 10, publicada pela Organização Mundial de Saúde (OMS). Visa padronizar a codificação de doenças e outros problemas relacionados à saúde;**
- **Despesas Saúde, prestação de contas apresentada pelo Ministério da Saúde contendo os gastos executados na área de saúde e suas ramificações.**



WORKFLOW



Arquivos originais no Postgres

1. Convertendo dataframes em formato .csv
2. Utilizando a biblioteca sqlalchemy para a criação de engine e envio dos arquivos para o banco de dados SQL

```
[4]: df_2017 = pd.read_csv('gs://projeto-final/arq-original/Mortalidade_Geral_2017.csv', sep = ';', low_memory=False)
```

```
[ ]: from sqlalchemy import create_engine  
engine = create_engine('postgresql://projfinal:postgres@35.192.214.202:5432/ProjetoFinal')  
df_2017.to_sql('sim2017', engine)
```

```
[4]: df_2018 = pd.read_csv('gs://projeto-final/arq-original/Mortalidade_Geral_2018.csv', sep = ';', low_memory=False)
```

```
[ ]: from sqlalchemy import create_engine  
engine = create_engine('postgresql://projfinal:postgres@35.192.214.202:5432/ProjetoFinal')  
df_2018.to_sql('sim2018', engine)
```

```
[4]: df_2019 = pd.read_csv('gs://projeto-final/arq-original/Mortalidade_Geral_2019.csv', sep = ';', low_memory=False)
```

```
[ ]: from sqlalchemy import create_engine  
engine = create_engine('postgresql://projfinal:postgres@35.192.214.202:5432/ProjetoFinal')  
df_2019.to_sql('sim2019', engine)
```



Arquivos originais no Postgres

1. Utilizando o comando "for" para a inserção do dataframe de 2020

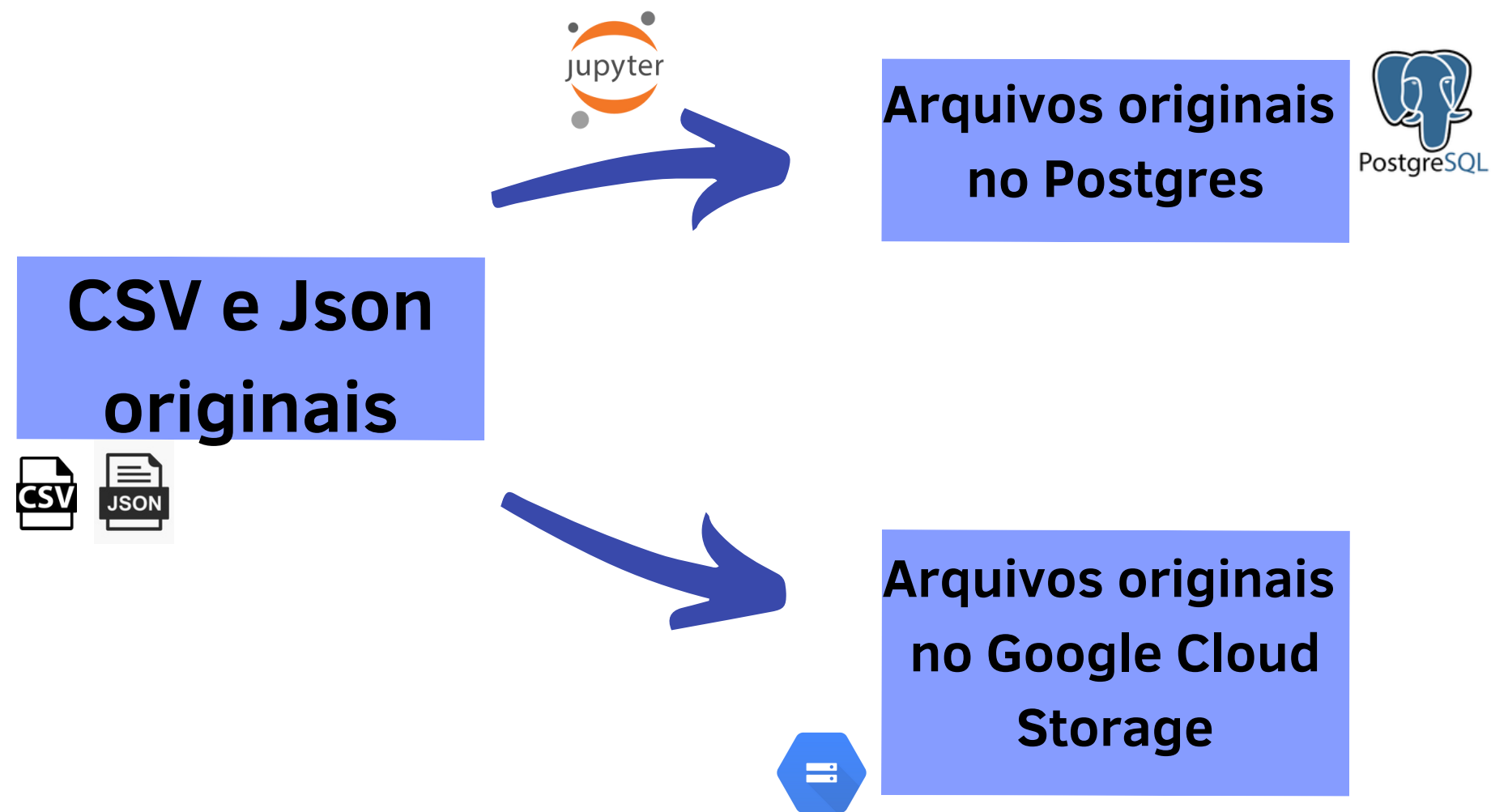
```
for index, row in df_2020.iterrows():
    banco.inserir(f"INSERT INTO sim2020 (contador , ORIGEM , TIPOBITO , DTOBITO , HORAOBITO , NATURA , CODMUNNATU , DTNASC , IDADE ,
SEXO , RACACOR, ESTCIV , ESC , ESC2010 , SERIESCFAL , OCUP , CODMUNRES , LOCOCOR , CODESTAB , CODMUNOCOR , IDADEMAE , \
ESMAE , ESMAE2010 , SERIESMAE , OCUPMAE , QTDFILVIVO , QTDFILMORT , GRAVIDEZ , SEMAGESTAC , GESTACAO , PARTO→ , \
OBITOPARTO , PESO , TPMORTEOCO , OBITOGRV , OBITOPUERP , ASSISTMED , EXAME , CIRURGIA , NECROPSIA , LINHAA , LINHAB , LINHAC , \
LINHAD , LINHAII , CAUSABAS , CB_PRE , COMUNSVOIM , DTATESTADO , CIRCOBITO , ACIDTRAB , FONTE , \
NUMEROLOTE , DTINVESTIG , DTCADASTRO , ATESTANTE , STCODIFICA , CODIFICADO , VERSAOSIST , VERSAOSCB , FONTEINV , \
DTRECEBIM , ATESTADO , DTRECORIGA , OPOR_DO , CAUSAMAT , ESMAEAGR1 , ESCFALAGR1 , STD OEPIDEM , STDONOVA , DIFDATA , \
NUDIASOBCO , DTCADINV , TPOBITOCOR , DTCONINV , FONTES , TPRESGINFO , TPNIVELINV , DTCADINF , MORTEPARTO , DTCONCASO , \
ALTCAUSA , CAUSABAS_O , TPPOS , TP_ALTERA , CB_ALT )\
VALUES ('{row['contador']}' , '{row['ORIGEM']}' , '{row['TIPOBITO']}' , '{row['DTOBITO']}' , '{row['HORAOBITO']}' , '{row['NATURA']}' , '{r
'{row['SEXO']}' , '{row['RACACOR']}' , '{row['ESTCIV']}' , '{row['ESC']}' , '{row['ESC2010']}' , '{row['SERIESCFAL']}' , '{row['OCUP']}' , '{
'{row['ESMAE']}' , '{row['ESMAE2010']}' , '{row['SERIESMAE']}' , '{row['OCUPMAE']}' , '{row['QTDFILVIVO']}' , '{row['QTDFILMORT']}' , '{
'{row['OBITOPARTO']}' , '{row['PESO']}' , '{row['TPMORTEOCO']}' , '{row['OBITOGRV']}' , '{row['OBITOPUERP']}' , '{row['ASSISTMED']}' , '{r
'{row['LINHAD']}' , '{row['LINHAII']}' , '{row['CAUSABAS']}' , '{row['CB_PRE']}' , '{row['COMUNSVOIM']}' , '{row['DTATESTADO']}' , '{row['C
'{row['NUMEROLOTE']}' , '{row['DTINVESTIG']}' , '{row['DTCADASTRO']}' , '{row['ATESTANTE']}' , '{row['STCODIFICA']}' , '{row['CODIFICADO'
'{row['DTRECEBIM']}' , '{row['ATESTADO']}' , '{row['DTRECORIGA']}' , '{row['OPOR_DO']}' , '{row['CAUSAMAT']}' , '{row['ESMAEAGR1']}' , '{r
'{row['TPOBITOCOR']}' , '{row['DTCONINV']}' , '{row['FONTES']}' , '{row['TPRESGINFO']}' , '{row['TPNIVELINV']}' , '{row['DTCADINF']}' , '{r
```

Arquivos originais no Postgres

1. Dados inseridos no banco de dados SQL

Data Output		Explain	Messages	Notifications								
	index bigint 	CONTADOR bigint 	ORIGEM bigint 	TIPOBITO bigint 	DTOBITO bigint 	HORAOBITO double precision 	NATURAL double precision 	CODMUNNATU double precision 	DTNASC double precision 	IDADE bigint 	SEXO bigint	
1	0	1	1	2	5092017	700	812	120039	3031997	420		
2	1	2	1	2	11022017	1330	812	120040	9022017	202		
3	2	3	1	2	11022017	500	812	120010	13071933	483		
4	3	4	1	2	11022017	830	812	120040	6022002	415		
5	4	5	1	2	11022017	320	812	120070	2061966	450		
6	5	6	1	2	11022017	1335	812	120040	28121946	470		
7	6	7	1	2	10022017	855	812	120010	4121955	461		
8	7	8	1	2	10022017	1115	812	120020	3101943	473		
9	8	9	1	2	10022017	1731	812	120040	6022017	204		

WORKFLOW



Arquivos originais no Cloud Storage

1. Arquivos originais salvos no bucket da Google Cloud

OBJETOS

CONFIGURAÇÃO

PERMISSÕES

PROTEÇÃO

CICLO DE VIDA

Intervalos > projeto-final > arq-original

FAZER UPLOAD DE ARQUIVOS

CARREGAR PASTA

CRIAR PASTA

GERENCIAR RETENÇÕES

FAZER O DOWNLOAD

EXCLUIR

Filtrar apenas pelo prefixo do nome

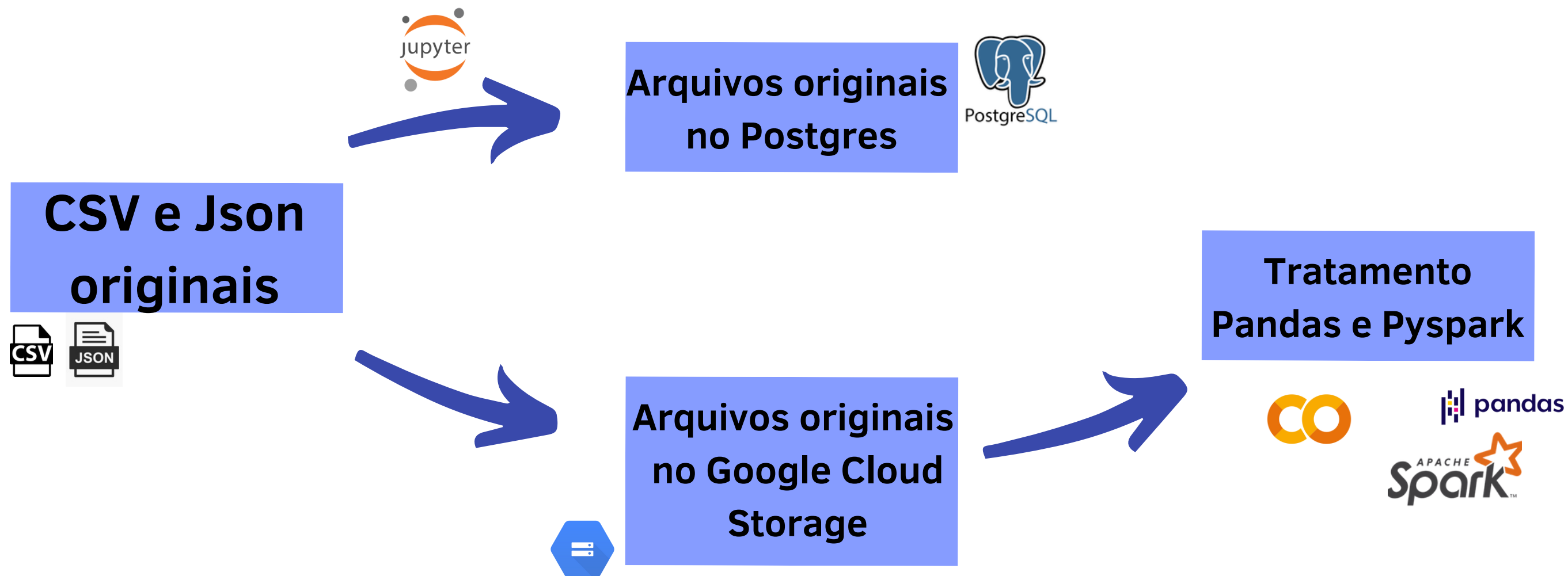
Filtro

Filtrar objetos e pastas

Mostrar da

<input type="checkbox"/>	Nome	Tamanho	Tipo	Criado ?	Classe de armazenamento
<input type="checkbox"/>	 Arquivos auxiliares/	—	Pasta	—	—
<input type="checkbox"/>	 Mortalidade_Geral_2017.csv	424,9 MB	text/csv	6 de jun....	Standard
<input type="checkbox"/>	 Mortalidade_Geral_2018.csv	427,8 MB	text/csv	6 de jun....	Standard
<input type="checkbox"/>	 Mortalidade_Geral_2019.csv	438,6 MB	text/csv	6 de jun....	Standard
<input type="checkbox"/>	 cid10.csv	939,4 KB	application/octet-stream	10 de ju...	Standard
<input type="checkbox"/>	 cid10.json	1,6 MB	application/json	6 de jun....	Standard
<input type="checkbox"/>	 dados.pdf	564,3 KB	application/pdf	2 de jun....	Standard
<input type="checkbox"/>	 despesas_saude.xlsx	91,2 KB	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	9 de jun....	Standard
<input type="checkbox"/>	 equipe4.csv	519,2 MB	text/csv	2 de jun....	Standard

WORKFLOW



Tratamento Pandas e Pyspark

TRATAMENTO PANDAS

- Avaliação dos campos e colunas que não serão aproveitadas;
- Substituição dos campos *NULL* por "NÃO INFORMADO" (String);
- Substituição dos campos *NULL* por "0"(Numérico);
- Criação de Schema e validação com Panderas;
- Plotagem de gráficos para geração de insights sobre a base de dados;

Tratamento Pandas e Pyspark

CONSTRUÇÃO DO SCHEMA

```
[ ] schema = pa.DataFrameSchema(  
    columns = {  
        "CAUSA_OBITO":pa.Column(pa.String),  
        "TIPO_OBITO":pa.Column(pa.Int32),  
        "DATA_OBITO":pa.Column(pa.String),  
        "DTNASC":pa.Column(pa.String),  
        "SEXO":pa.Column(pa.String),  
        "RACA_COR":pa.Column(pa.String),  
        "OCUPACAO":pa.Column(pa.Float64)
```

SUBSTITUINDO OS CAMPOS NULL

```
▶ df_pandas['DTNASC'].fillna(0, inplace=True)  
df_pandas['RACA_COR'].fillna('Não Informado', inplace=True)  
df_pandas['OCUPACAO'].fillna(0, inplace=True)
```

Tratamento Pandas e Pyspark

Agrupamentos do dataframe para melhor visualização dos números

```
df_pandas.groupby(['DESCRICAO_OBITO']).size().sort_values(ascending=False).head(10)
```

DESCRICAO_OBITO	
Infecção Por Coronavírus de Localização Não Especificada	209720
Infarto Agudo do Miocárdio Não Especificado	87008
Outras Causas Mal Definidas e as Não Especificadas de Mortalidade	63772
Pneumonia Não Especificada	37483
Hipertensão Essencial (primária)	37129
Acidente Vascular Cerebral, Não Especificado Como Hemorrágico ou Isquêmico	35382
Diabetes Mellitus Não Especificado - Sem Complicações	32067
Neoplasia Maligna Dos Brônquios ou Pulmões, Não Especificado	26368
Doença de Alzheimer Não Especificada	21798
Infecção do Trato Urinário de Localização Não Especificada	19963

dtype: int64

```
[ ] df_pandas.groupby(['RACA_COR']).size().sort_values(ascending=False)
```

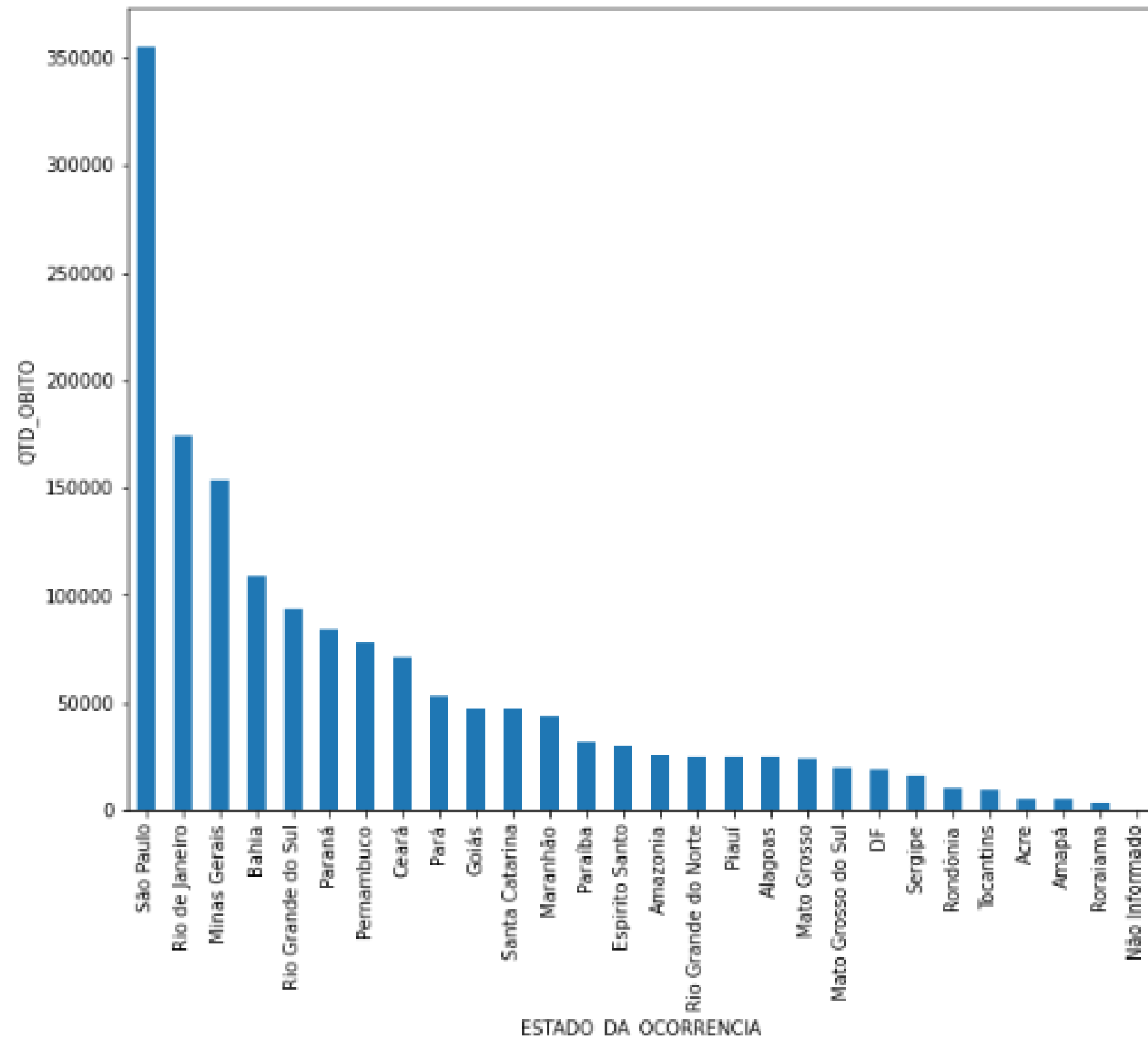
RACA_COR	
Branca	762577
Parda	602642
Preta	131638
Não Informado	70020
Amarela	9450
Indigena	5318

dtype: int64

Tratamento Pandas e Pyspark

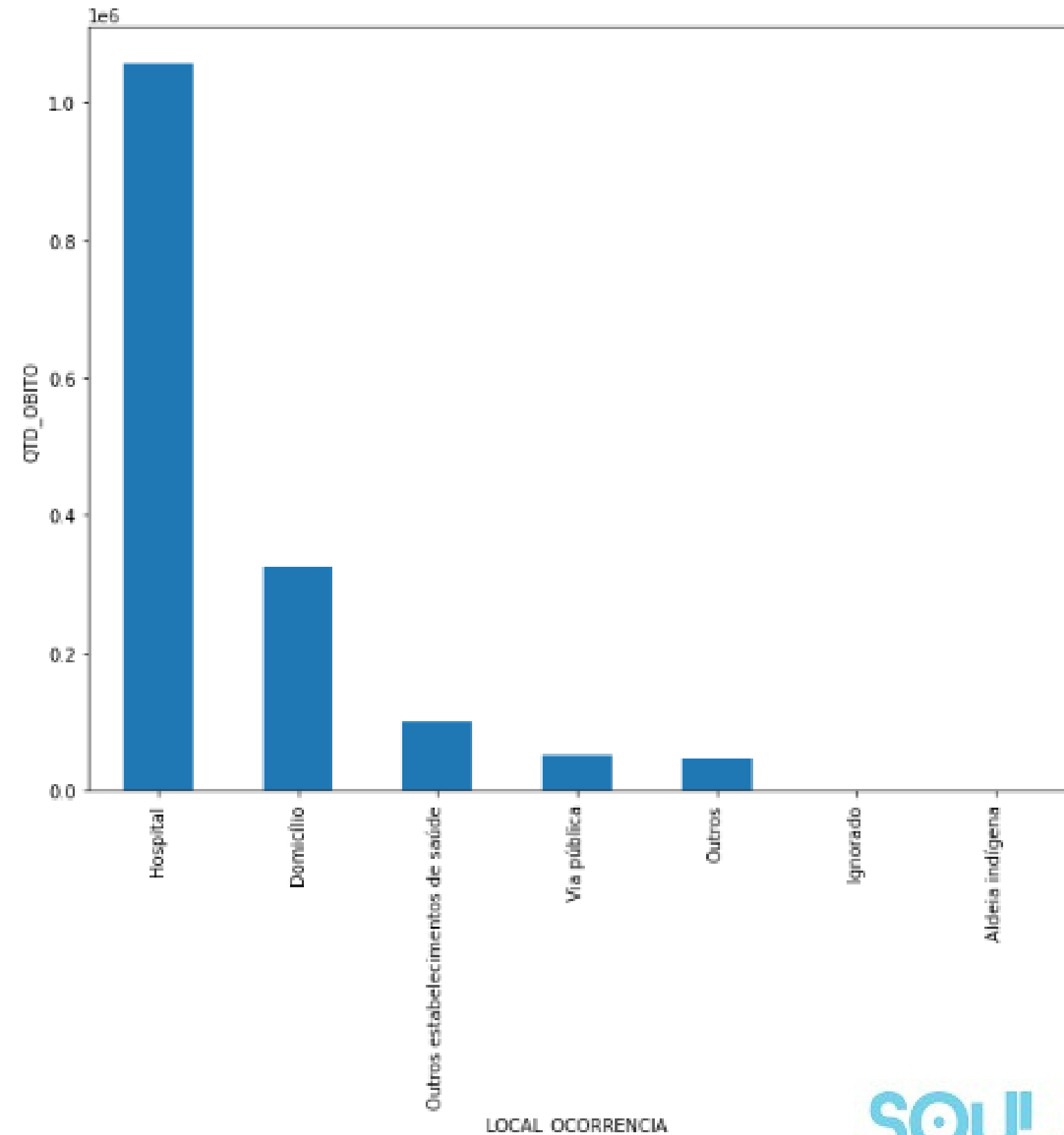
```
df_pandas.groupby(['ESTADO_DA_OCORRENCIA']).size().sort_values(ascending=False).plot
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f3d1b37d790>



```
df_pandas.groupby(['LOCAL_OCORRENCIA']).size().sort_values(ascending=False).plot
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f3d1a4cf910>



Tratamento Pandas e Pyspark

TRATAMENTO PYSPARK

- Contagem dos campos "*Null*" de cada coluna;
- Drop de colunas não utilizadas;
- Renomear colunas;
- Tratamento das colunas de datas;
- Calculo de idade do falecido;
- Criação de coluna com os estados em que ocorreram os óbitos;
- Criação de coluna com a causa dos óbitos utilizando "join";
- Inclusão de coluna com faixa etária dos óbitos.

Tratamento Pandas e Pyspark

RENOMEANDO INFORMAÇÕES DAS COLUNAS RACA_COR, LOCAL_OCORRENCIA E SEXO

```
df2020 = df2020.withColumn('RACA_COR', F.regexp_replace('RACA_COR', '1', 'Branca'))  
                .withColumn('RACA_COR', F.regexp_replace('RACA_COR', '2', 'Preta'))  
                .withColumn('RACA_COR', F.regexp_replace('RACA_COR', '3', 'Amarela'))  
                .withColumn('RACA_COR', F.regexp_replace('RACA_COR', '4', 'Parda'))  
                .withColumn('RACA_COR', F.regexp_replace('RACA_COR', '5', 'Indigena'))
```

```
df2020 = df2020.withColumn('LOCAL_OCORRENCIA', F.regexp_replace('LOCAL_OCORRENCIA', '1', 'Hospital'))  
                .withColumn('LOCAL_OCORRENCIA', F.regexp_replace('LOCAL_OCORRENCIA', '2', 'Outros estabelecimentos de saúde'))
```

Inclusão de coluna por:

- Jovens - Indivíduos de até 19 anos;
- Adultos - Indivíduos com idade entre 20 até 59 anos;
- Idosos - Indivíduos de 60 anos em diante.

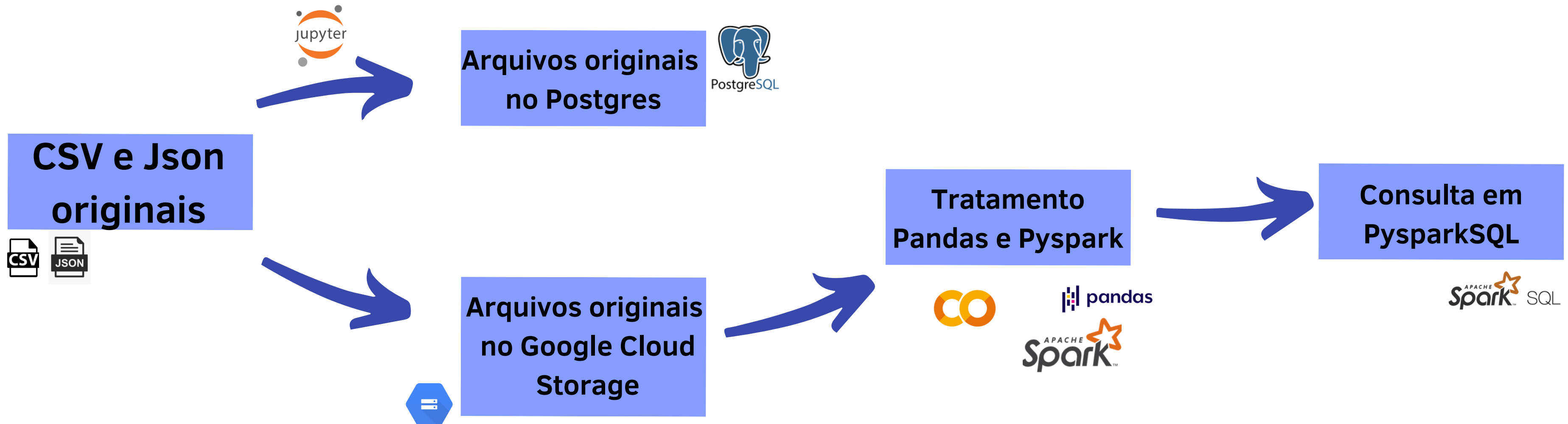
```
df2020 = (df2020.withColumn('IDADE', F.when((F.col('IDADE_OBITO') <= 19), F.lit('Jovem'))  
                .when((F.col('IDADE_OBITO') > 19) & (F.col('IDADE_OBITO') <= 59), F.lit('Adulto'))  
                .when((F.col('IDADE_OBITO') > 59), F.lit('Idoso'))  
                )
```

Tratamento Pandas e Pyspark

Substituição dos códigos das cidades em que ocorreram os óbitos pelo nome do estado.

```
2020.withColumn('ESTADO_DA_OCORRENCIA', F.when((F.col('CIDADE_OBITO') > 310000) & (F.col('CIDADE_OBITO') < 319999), F.lit('Minas Ge  
.when((F.col('CIDADE_OBITO') > 110000) & (F.col('CIDADE_OBITO') < 119999), F.lit('Rondônia  
.when((F.col('CIDADE_OBITO') > 120000) & (F.col('CIDADE_OBITO') < 129999), F.lit('Acre'))  
.when((F.col('CIDADE_OBITO') > 130000) & (F.col('CIDADE_OBITO') < 139999), F.lit('Amazonia  
.when((F.col('CIDADE_OBITO') > 140000) & (F.col('CIDADE_OBITO') < 149999), F.lit('Roraiama  
.when((F.col('CIDADE_OBITO') > 150000) & (F.col('CIDADE_OBITO') < 159999), F.lit('Pará'))  
.when((F.col('CIDADE_OBITO') > 160000) & (F.col('CIDADE_OBITO') < 169999), F.lit('Amapá'))  
.when((F.col('CIDADE_OBITO') > 170000) & (F.col('CIDADE_OBITO') < 179999), F.lit('Tocanti  
.when((F.col('CIDADE_OBITO') > 210000) & (F.col('CIDADE_OBITO') < 219999), F.lit('Maranhão
```


WORKFLOW



Consulta em PysparkSQL

Select realizado para filtrar os campos abaixo.

```
#CRIANDO CONSULTA PARA HOSPITAL
```

```
df2020.where(F.col('LOCAL_OCORRENCIA') == 'Hospital').show()
```

CAUSA_OBITO	TIPO_OBITO	DATA_OBITO	DTNASC	SEXO	RACA_COR	OCUPACAO	LOCAL_OCORRENCIA	COD_ESTABELECIMENTO	CIDADE_OBITO
K579	2	2020-01-14	1937-07-10	Fem	Branca	999992	Hospital	2760657	316930
J960	2	2020-01-20	1946-11-02	Fem	Branca	999992	Hospital	2760657	316930
A419	2	2020-01-26	1959-09-12	Masc	Branca	231205	Hospital	2760657	316930
J960	2	2020-01-27	1964-05-08	Masc	Parda	783225	Hospital	2760657	316930
R001	2	2020-01-11	1943-05-10	Masc	Preta	999993	Hospital	2760657	316930
P369	2	2020-01-15	2020-01-13	Masc	Parda	null	Hospital	2760657	316930
I219	2	2020-01-12	1934-09-09	Masc	Branca	999993	Hospital	2760657	316930
R99	2	2020-01-10	1961-04-24	Fem	Preta	999992	Hospital	2760657	316930
J189	2	2020-01-28	1950-08-28	Fem	Branca	999992	Hospital	2760657	316930
N390	2	2020-01-28	1949-08-10	Fem	Preta	999992	Hospital	2760657	316930
J189	2	2020-01-16	1946-10-26	Fem	Parda	999993	Hospital	6957501	210480
V224	2	2020-01-17	1948-08-01	Masc	Parda	999993	Hospital	6957501	210480

Consulta em PysparkSQL

Criação de consultas personalizadas sobre a faixa etária.

```
[ ] sim2020idade_jovem = spark.sql("select IDADE, count(*) QTD_OBITO from SIM2020 group by IDADE having (IDADE = 'Jovem')
```

```
[ ] sim2020idade_adulto = spark.sql("select IDADE, count(*) QTD_OBITO from SIM2020 group by IDADE having (IDADE = 'Adulto')
```

```
[ ] sim2020idade_idoso = spark.sql("select IDADE, count(*) QTD_OBITO from SIM2020 group by IDADE having (IDADE = 'Idoso')
```

```
▶ sim2020idade_adulto.show()
```

```
+-----+-----+
| IDADE|QTD_OBITO|
+-----+-----+
|Adulto|    392101|
+-----+-----+
```

WORKFLOW



DATALAKE

Utilização do MongoDB Atlas para importação dos dados tratados


ENVIO DOS DATAFRAMES TRATADOS PARA O BANCO DE DADOS



```
[ ] df_2017 = pd.read_csv('gs://projeto-final/tratados/df_2017.csv', delimiter=',')
```

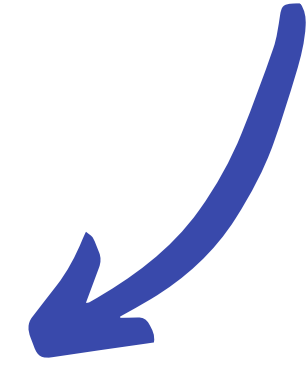
TRANSFORMANDO EM DICIONÁRIO

```
[ ] df_dicio = df_2017.to_dict('records')
```



DETERMINANDO A COLLECTION

```
[ ] collection = db['df_2017']
```



INSERINDO NO MONGO

```
[ ] collection.insert_many(df_dicio)
```

DATALAKE

Utilização do MongoDB Atlas para importação dos dados tratados

CONEXÃO COM O MONGO









```
[ ] client = MongoClient("mongodb+srv://[redacted]@cluster1.mquek.mongodb.net/?retryWri

[ ] #determinando o database
    db = client['Projeto_Final']
```

The screenshot displays the MongoDB Atlas web interface. On the left sidebar, there is a '+ Create Database' button and a 'Search Namespaces' search bar. Below the search bar, the 'Projeto_Final' namespace is expanded, showing a list of collections: 'df_2017' (highlighted with a green bar), 'df_2018', 'df_2019', and 'df_2020'. The main panel shows the details for the 'Projeto_Final.df_2017' collection. It includes statistics: 'STORAGE SIZE: 130.69MB', 'TOTAL DOCUMENTS: 1312663', and 'INDEXES TOTAL SIZE: 35.88MB'. Below these are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. The 'Find' tab is active, showing a query filter bar with the text '{ field: 'value' }' and an 'Apply' button. Below the filter bar, it says 'QUERY RESULTS: 1-20 OF MANY'. A sample document is shown with the following fields: '_id: ObjectId("62a247a2ef6021efeee9a3e4")', 'CAUSA_OBITO: "R58"', 'TIPO_OBITO: 2', and 'DATA_OBITO: "2017-09-05"'.

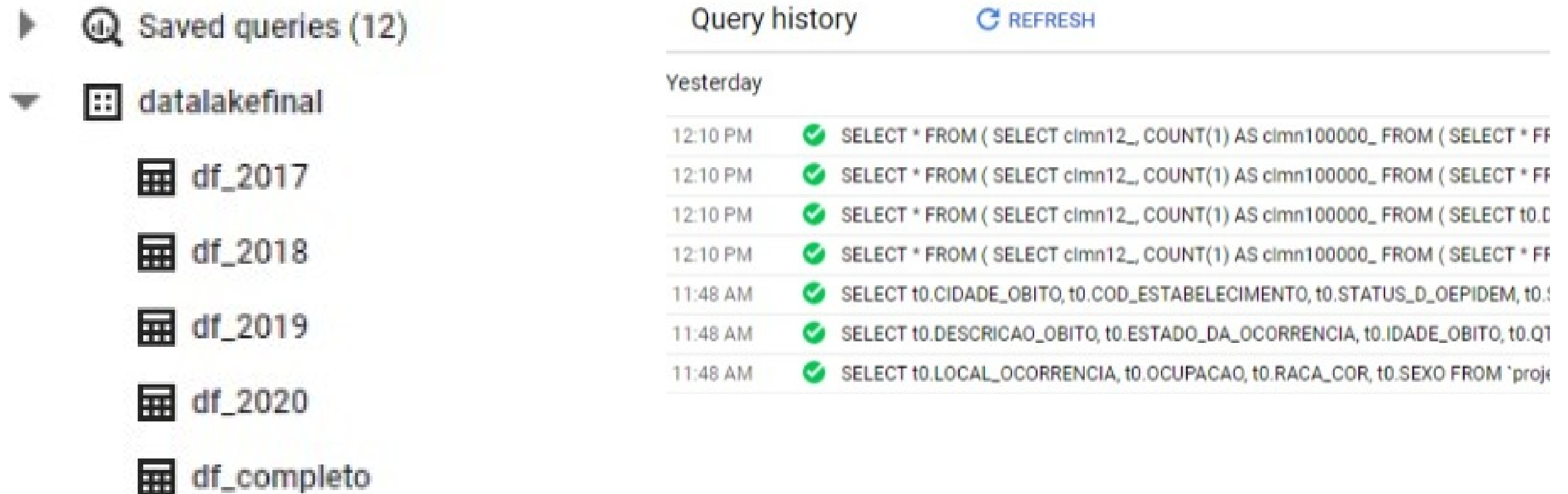
DATALAKE

Utilização do BigQuery e criação de consultas

 BigQuery	FEATURES & INFO	SHORTCUT	ENABLE EDITOR TABS
Query history	Query editor		
Saved queries	Saved queries		
Job history			
Transfers	 MORTES_ORDENADAS_POR_CAUSA_2017		
Scheduled queries	 MORTES_ORDENADAS_POR_CAUSA_2018		
Monitoring	 MORTES_ORDENADAS_POR_CAUSA_2019		
Capacity management	 MORTES_ORDENADAS_POR_CAUSA_2020		
BI Engine	 MORTES_ORDENADAS_POR_ESTADO_2017		
	 MORTES_ORDENADAS_POR_ESTADO_2018		
	 MORTES_ORDENADAS_POR_ESTADO_2019		

DATALAKE

Utilização do BigQuery como um DataLake e selects realizados



The screenshot displays the Google Cloud BigQuery interface. On the left, under 'Saved queries (12)', a dataset named 'datalakefinal' is expanded, showing several dataframes: 'df_2017', 'df_2018', 'df_2019', 'df_2020', and 'df_completo'. On the right, the 'Query history' tab is active, showing a list of queries executed yesterday. Each entry includes a timestamp, a status icon (a green checkmark), and the SQL query text. The queries are complex, involving multiple nested SELECT statements and JOINs.

Time	Status	Query
12:10 PM	✓	SELECT * FROM (SELECT clmn12_, COUNT(1) AS clmn100000_ FROM (SELECT * FROM ...
12:10 PM	✓	SELECT * FROM (SELECT clmn12_, COUNT(1) AS clmn100000_ FROM (SELECT * FROM ...
12:10 PM	✓	SELECT * FROM (SELECT clmn12_, COUNT(1) AS clmn100000_ FROM (SELECT t0.C...
12:10 PM	✓	SELECT * FROM (SELECT clmn12_, COUNT(1) AS clmn100000_ FROM (SELECT * FROM ...
11:48 AM	✓	SELECT t0.CIDADE_OBITO, t0.COD_ESTABELECIMENTO, t0.STATUS_D_OEPIDEM, t0.S...
11:48 AM	✓	SELECT t0.DESCRICAO_OBITO, t0.ESTADO_DA_OCORRENCIA, t0.IDADE_OBITO, t0.Q1...
11:48 AM	✓	SELECT t0.LOCAL_OCORRENCIA, t0.OCUPACAO, t0.RACA_COR, t0.SEXO FROM 'proj...

Importante ressaltar que todos os dataframes tratados e salvos no bucket possuíam mais de 100mb de tamanho, não sendo permitida a sua importação para o DataStudio. Utilizamos os dataframes salvos no BigQuery.

WORKFLOW



PIPELINE

Modelo de pipeline em ApacheBeam criado para rodar em DataFlow. A finalidade é separar as informações de Estado e Quantidade de mortes.

```
p1 = beam.Pipeline()
pipeline_options = {
    'project': 'projeto-soulcode-350816',
    'runner': 'DataflowRunner',
    'region': 'southamerica-east1',
    'staging_location': 'gs://projeto-final/staging',
    'temp_location': 'gs://projeto-final/staging',
    'template_location': 'gs://projeto-final/models/modelo_qtd_morte_estado'
}
pipeline_options = PipelineOptions.from_dictionary(pipeline_options)
p1 = beam.Pipeline(options=pipeline_options)
qtd_obito = (
    p1
    | 'Leitura do dataset' >> beam.io.ReadFromText('gs://projeto-final/tratados/df_2017.csv', skip_header_lines=1)
    | 'Separar por virgula' >> beam.Map(lambda record: record.split(','))
    | 'Filtrar por região' >> beam.Filter(lambda record: float(record[13]) == '1')
    | 'Agregar as colunas' >> beam.Map(lambda record: (record[15], float(record[13])))
    | 'Qtd de mortes' >> beam.combiners.Count.PerKey(sum)
    | 'Load para arquivo' >> beam.io.WriteToText("gs://projeto-final/test/modelo_qtd_morte_estado", file_name_suffix='.csv')
)
p1.run()
```

PIPELINE

Job steps view


Graph view

CLEAR SELECTION

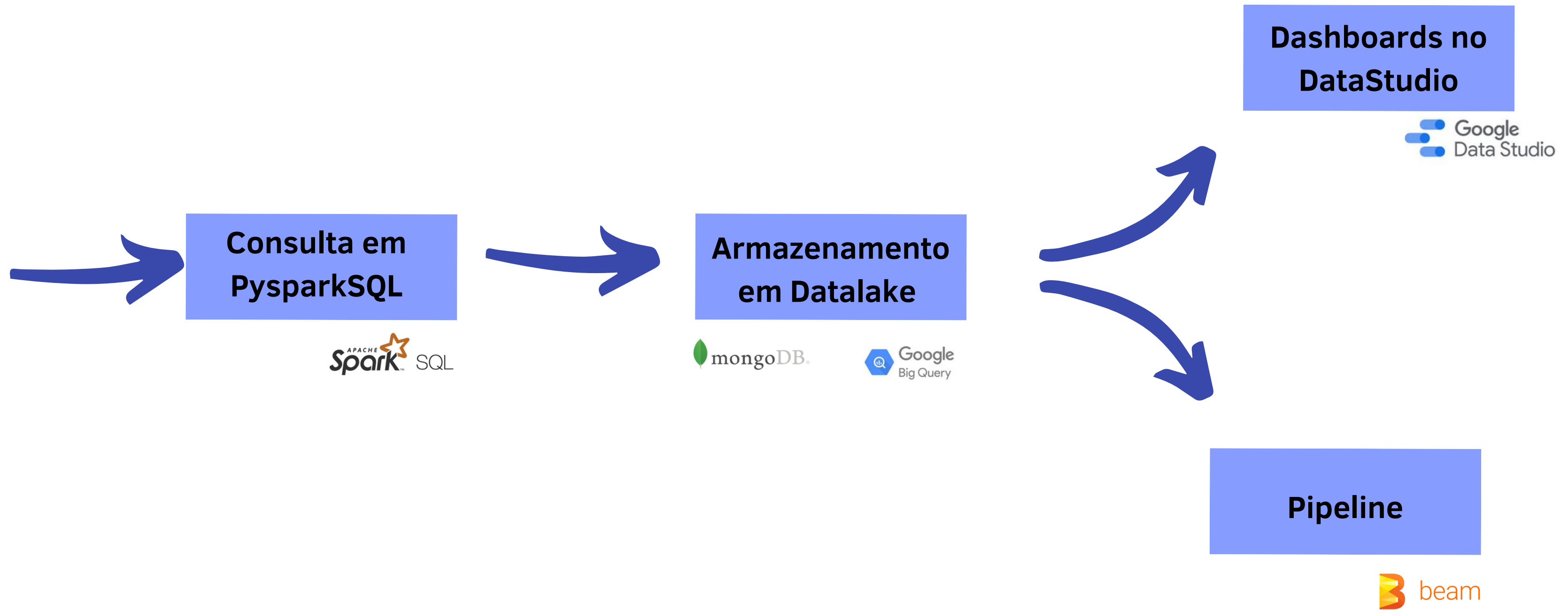
11,943 elements/s
1 sec
1 of 1 stage succeeded

✓ [39]: Qtd de mortes

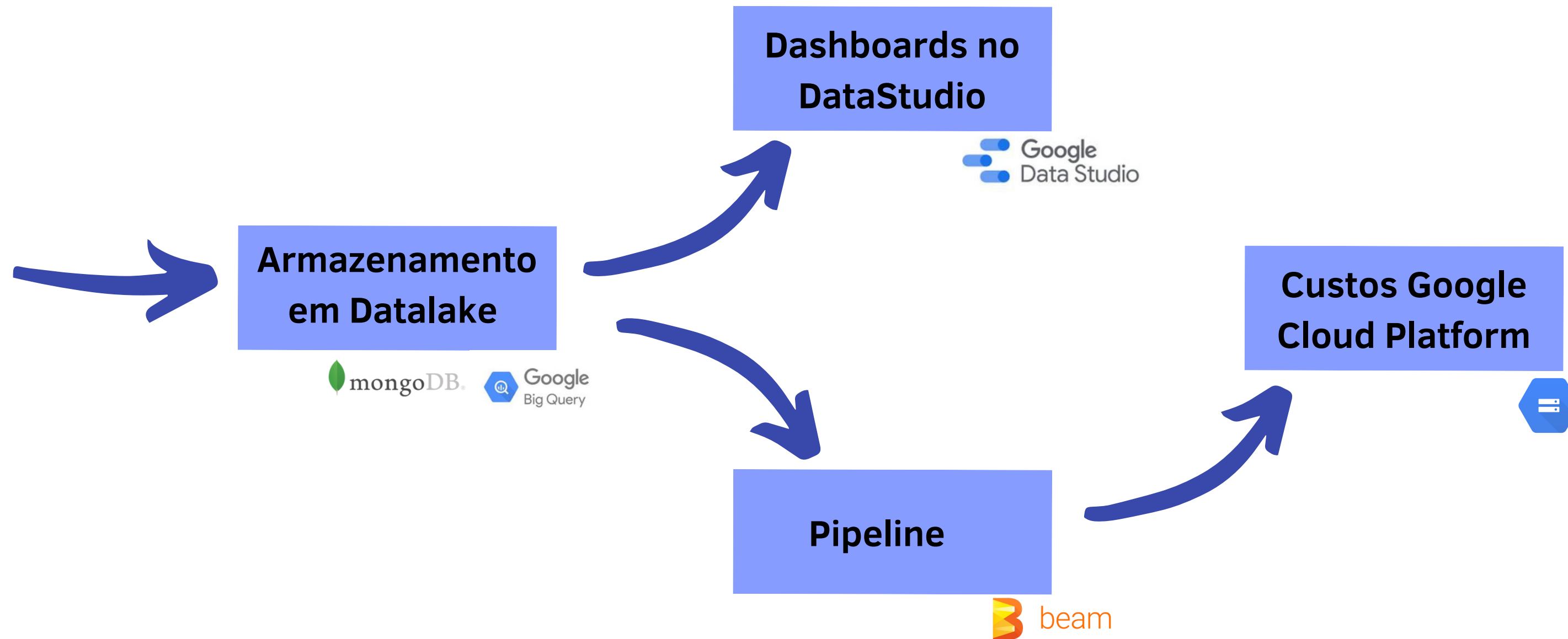
11,943 elements/s
3 sec
2 of 2 stages succeeded

Job name	projfinaljob
Job ID	2022-06-12_15_20_06-8190686384384358896
Job type	Batch
Job status	 Running
SDK version	Apache Beam Python 3.7 SDK 2.39.0
Job region ?	us-central1
Worker location ?	us-central1-b
Current workers ?	1
Latest worker status	Stopping worker pool.
Start time	June 12, 2022 at 3:00:06 PM GMT-3

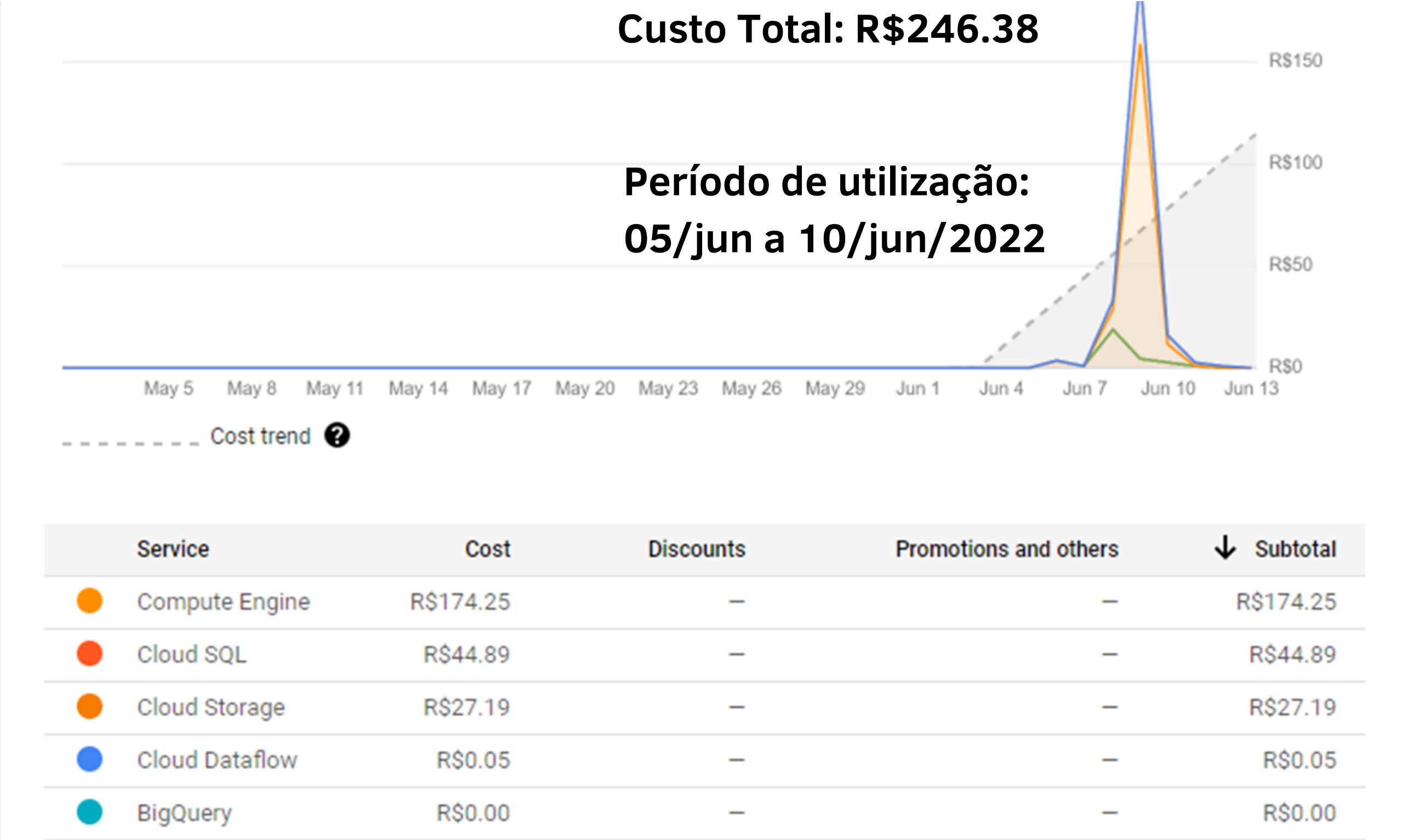
WORKFLOW



WORKFLOW



CUSTOS GOOGLE CLOUD PLATFORM



RESULTADO

Após a apresentação do escopo do projeto foram necessárias diversas ações, como pudemos verificar nesta apresentação.

De maneira geral, todos os pontos descritos no escopo foram atendidos, tendo como resultado o processo de ETL do arquivo apresentado, bem como a criação dos insights e análise das informações para a geração de informação necessária ao cliente SoulCode.

Além do resultado acima mencionado, também ocorreu uma transformação na vida profissional dos 4 integrantes dessa equipe, pois saímos desse bootcamp com uma nova profissão, sensação de dever cumprido e a cultura de aprendizado constante.

FONTES UTILIZADAS

Site do Ministério da Saúde (Governo Federal) - dados.gov.br

Site Hospital Albert Einstein - www.einstein.br/Pages/Home.aspx

Site IBGE para população por estado - www.ibge.gov.br/cidades-e-estados

GitHub para obtenção do CID10 - github.com/a21ns1g4ts/cid-api-php/tree/master/database/seeds

AGRADECIMENTOS

**Agradecemos à SoulCode pela oportunidade de aprendizado,
aos professores Adriano, Felipe, Igor e Bismarck pelos ensinamentos,
a orientadora do grupo Sayure pelos direcionamentos,
aos amigos da equipe pelo apoio mútuo e incentivo nas horas mais difíceis,
aos demais colegas do bootcamp pela convivência que levaremos para a vida
e finalmente às nossas famílias pelo suporte neste momento de transição!!!**

MUITO OBRIGADO!!!

CONTATOS

Felipe Rodrigues - www.linkedin.com/in/felipe-rodrigues-technician/

Jefferson Costa - www.linkedin.com/in/jefferson-costa-39951822b/

Lucas David - www.linkedin.com/in/lucas-monteiro-712754213/

Tiago Ferreira - www.linkedin.com/in/silva-tiagof/