databricks MVP-Sprint: Eng. Dados

(https://databricks.com)

MPV - Sprint: Engenharia de Dados

```
import pandas as pd
import numpy as np
from pyspark.sql import SparkSession
```

1. Dataset tb_comerciais

```
# Carregando o datasets com dados servicos comerciais
query = """
SELECT *
FROM `hive_metastore`.`default`.`tb_comercial`
"""
df_spark = spark.sql(query)
df_comercial = df_spark.toPandas()
```

```
# Verifica o tamanho do dataset
df_comercial.shape
```

```
Out[3]: (2130331, 16)
```

```
# carrega as primeiras linhas do dataset (quantidade de linhas
especificada entre parenteses)
df_comercial.head(3)
```

	ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2
0	9.100752e+16	35.0	195	E2136	CB625	980187	980307
1	9.100745e+16	35.0	9925	E3P70	MA945	972165	972417

```
# verificar os tipos de dados por colunas e valores nulos
df_comercial.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2130331 entries, 0 to 2130330
Data columns (total 16 columns):
 #   Column         Dtype
---  ------         -----
 1   seq_servico    float32
 0   ordem_servico  float32
 2   cod_servico    int32
```

```
1   seq_servico    float32
0   ordem_servico  float32
2   cod_servico    int32
3   cod_veiculo    object
4   cod_equipe     object
5   colab_1        int32
6   colab_2        int32
7   cod_susp       int32
8   cod_impedido   int32
9   dt_saida       object
10  dt_inicio      object
11  dt_fim         object
12  dt_termino     object
13  cod_local      int32
14  sub_regiao     object
```

```
# verificar os tipos de dados por colunas
df_comercial.dtypes
```

```
Out[6]: ordem_servico    float32
seq_servico      float32
cod_servico        int32
cod_veiculo       object
cod_equipe        object
colab_1            int32
colab_2            int32
cod_susp           int32
cod_impedido       int32
dt_saida          object
dt_inicio         object
dt_fim            object
dt_termino        object
cod_local          int32
sub_regiao        object
km                 int32
dtype: object
```

```
# verificar os valores nulos em cada coluna do dataset
df_comercial[df_comercial.columns].isnull().sum()
```

```
Out[7]: ordem_servico        0
seq_servico          0
cod_servico          0
cod_veiculo          0
cod_equipe       11857
colab_1              0
colab_2              0
cod_susp             0
cod_impedido         0
dt_saida          3268
dt_inicio          450
dt_fim               0
dt_termino           0
```

```
dt_fim           450
dt_inicio          0
dt_termino         0
cod_local          0
sub_regiao         4
km                 0
dtype: int64

# substituição dos valores NAN por um 'coringa' denominado '100000'
para facilitar as tratativas futuras
df_comercial=df_comercial.fillna(100000)


# verificar que não existem mais NaN nas colunas do dataset
df_comercial[df_comercial.columns].isnull().sum()

Out[9]: ordem_servico    0
seq_servico      0
cod_servico      0
cod_veiculo      0
cod_equipe       0
colab_1          0
colab_2          0
cod_susp         0
cod_impedido     0
dt_saida         0
dt_inicio        0
dt_fim           0
dt_termino       0
cod_local        0
sub_regiao       0
km               0
dtype: int64


# Conversa]ão das colunas 'dt_fim' em data e horas para as trataivas de
valores
df_comercial['dt_fim'] = pd.to_datetime(df_comercial['dt_fim'],
errors='coerce')


# Tratativas do valor coringa incrementando a data de acordo com as
regras de negocios, neste caso 30 minutos para execução de um serviçco
de acordo com os tempos padrões estabelecidos pela Companhio.
df_comercial.loc[df_comercial['dt_inicio'] == 100000, 'dt_inicio'] =
df_comercial['dt_fim'] - pd.to_timedelta(30, unit='D')


# Conversão das colunas 'dt_inicio' em data e horas para as trataivas
de valores
df_comercial['dt_inicio'] = pd.to_datetime(df_comercial['dt_inicio'],
errors='coerce')
# Tratativas do valor coringa incrementando a data de acordo com as
```

```
errors='coerce')
# Tratativas do valor coringa incrementando a data de acordo com as
regras de negocios, neste caso 15 minutos na m[edia nos deslocamentos
considerando a área urbana.
df_comercial.loc[df_comercial['dt_saida'] == 100000, 'dt_saida'] =
df_comercial['dt_inicio'] - pd.to_timedelta(15, unit='D')


# Conversa]ão das colunas 'dt_saida' em data e horas para as trataivas
de valores
df_comercial['dt_saida'] = pd.to_datetime(df_comercial['dt_saida'],
errors='coerce')


# Conversa]ão das colunas 'dt_termino' em data e horas para as
trataivas de valores
df_comercial['dt_termino'] = pd.to_datetime(df_comercial['dt_termino'],
errors='coerce')


# Tratativa dos valores coringas no atributo 'cod_equipe' conforme
segue
df_comercial.loc[df_comercial['cod_equipe'] == 100000, 'cod_equipe'] =
'NOT_INF'


# Devido a ser somente 4 registro para um universo de 2130331 milhões
de registro esta supresão não impactara no resultado final do projeto
df_comercial=df_comercial.drop(df_comercial[df_comercial['sub_regiao']
== 100000].index)


# Verificando se os valores coringas foram tratados adquadamente
df_comercial.loc[df_comercial['dt_saida'] == 100000]
```

	ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2

```
# Verificando se os valores coringas foram tratados adquadamente
df_comercial.loc[df_comercial['dt_inicio'] == 100000]
```

	ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2

```
# Verificando se os valores coringas foram tratados adquadamente
df_comercial.loc[df_comercial['cod_equipe'] == 100000]
```

	ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2

```
# Verificando se os valores coringas foram tratados adquadamente
```

ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2

```
# Verificando se os valores coringas foram tratados adquadamente
df_comercial.loc[df_comercial['sub_regiao'] == 100000]
```

ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2

```
# Substituir os valores "0" na coluna 'cod_susp' pelo valor "10000"
para diferenciar do valor '0' na coluna "cod_impedido" com mesmo número
e conflita na comparacao de conclusão
#dos serviços comerciais executados e cortes executados.
df_comercial.loc[(df_comercial['cod_servico'] == 401) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 402) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 404) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 405) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 406) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 407) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 409) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 412) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 417) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 419) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 420) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 430) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 440) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 441) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 442) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 443) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 449) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
df_comercial.loc[(df_comercial['cod_servico'] == 499) &
(df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
```

```
# consulta se ficou valores remanecente dos serviços de cortes com o
```

```
# consulta se ficou valores remanecente dos serviços de cortes com o
"cod_susp" igual a '0'
df_comercial.loc[(df_comercial['cod_servico'] == 401) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 402) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 403) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 405) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 406) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 407) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 409) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 412) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 417) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 419) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 420) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 430) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 440) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 441) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 442) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 443) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 449) &
(df_comercial['cod_susp'] == 0)]
df_comercial.loc[(df_comercial['cod_servico'] == 499) &
(df_comercial['cod_susp'] == 0)]
```

	ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2

```
# consulta se a alteração dos valores '0' da coluna 'cod_susp' foram
```

```
# consulta se a alteração dos valores '0' da coluna 'cod_susp' foram
alterados para '10000'
df_comercial.loc[(df_comercial['cod_servico'] == 401) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 402) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 403) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 405) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 406) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 407) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 409) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 412) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 417) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 419) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 420) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 430) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 440) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 441) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 442) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 443) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 449) &
(df_comercial['cod_susp'] == 10000)]
df_comercial.loc[(df_comercial['cod_servico'] == 499) &
(df_comercial['cod_susp'] == 10000)]
```

	ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	c

	ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	c
191	9.100749e+16	3850.0	499	V50003	MA107	126345	
1684	9.100749e+16	10150.0	499	M50002	MA916	145149	
1892	9.100749e+16	9695.0	499	E3T88	E3T88	957981	9
2061	9.100749e+16	6020.0	499	E3499	CB054	974637	9
3208	9.100749e+16	280.0	499	V30006	CB529	124623	
...	...	...	...	...	...	...	
2098433	9.105803e+16	8085.0	499	E3J97	CB290	979269	9
2101267	9.105803e+16	7385.0	499	E3L38	CB611	1011288	1(
2120003	9.105811e+16	26495.0	499	E3N03	CB016	1012014	1(

```
# separando as variáveis numericas
nums_comer = ['ordem_servico', 'seq_servico', 'cod_servico', 'colab_1',
'colab_2', 'cod_susp', 'cod_impedido',
       'cod_local','km']
```

```
# apresenta um resumo de estatistica das variáveis numéricas
df_comercial[nums_comer].describe()
```

	ordem_servico	seq_servico	cod_servico	colab_1	colab_2	cod_
count	2.130327e+06	2.130327e+06	2.130327e+06	2.130327e+06	2.130327e+06	2.130327
mean	9.105330e+16	3.396836e+03	1.914084e+03	8.446204e+05	4.617817e+05	2.98891(
std	1.027607e+13	6.026083e+03	3.271610e+03	4.334176e+05	5.526629e+05	3.20260(
min	9.079070e+16	3.500000e+01	1.010000e+02	5.522700e+04	0.000000e+00	0.00000(
25%	9.105381e+16	3.500000e+01	4.020000e+02	8.988900e+05	0.000000e+00	2.00000(
50%	9.105524e+16	3.500000e+01	4.420000e+02	9.827220e+05	1.388430e+05	3.00000(
75%	9.105669e+16	4.795000e+03	5.400000e+02	9.926520e+05	9.814860e+05	4.20000(
max	9.105831e+16	7.875000e+04	9.999000e+03	2.412054e+06	2.412045e+06	1.00000(

```
# salvar o DataFrame como um arquivo CSV no diretório criado
```

```
# salvar o DataFrame como um arquivo CSV no diretório criado
#directory_path = '/dbfs/FileStore/tables'
#dbutils.fs.mkdirs(directory_path)
#df_comercial.to_csv("/dbfs/FileStore/tables/comercial")
```

2. Dataset tb_emergenciais

```
# Carregando o datasets com dados servicos emergenciais
query = """
SELECT *
FROM `hive_metastore`.`default`.`tb_emergencial`
"""
df_spark = spark.sql(query)
df_emergencial = df_spark.toPandas()
```

```
# Verifica o tamanho do dataset
df_emergencial.shape
```

```
Out[29]: (546558, 15)
```

```
# carrega as primeiras linhas do dataset (quantidade de linhas
especificada entre parenteses)
df_emergencial.head(3)
```

	ordem_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2	cod_impedid
0	25729389000	6002	T30031	CA850	143148	155268	8
1	16434832500	6002	E2189	E2189	980082	981486	

```
# verificar os tipos de dados por colunas e valores nulos
df_emergencial.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 546558 entries, 0 to 546557
Data columns (total 15 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   ordem_servico  546558 non-null  object
 1   cod_servico    546558 non-null  int32
 2   cod_veiculo    546558 non-null  object
 3   cod_equipe     546558 non-null  object
 4   colab_1        546558 non-null  int32
 5   colab_2        546558 non-null  int32
 6   cod_impedido   546558 non-null  int32
 8   dt_saida       546558 non-null  object
 7   cod_usuario    546558 non-null  object
 9   dt_interno     546558 non-null  object
```

```
 6   cod_impedido   546558 non-null  int32
 7   dt_saida       546558 non-null  object
 8   cod_susp       546558 non-null  object
 9   dt_interno     546558 non-null  object
 10  dt_fim         546558 non-null  object
 11  dt_termino     546558 non-null  object
 12  cod_local      546558 non-null  int32
 13  sub_regiao     546517 non-null  object
 14  km             546558 non-null  int32
dtypes: int32(6), object(9)
```

```
# verificar os tipos de dados por colunas
df_emergencial.dtypes
```

```
Out[32]: ordem_servico    object
cod_servico       int32
cod_veiculo      object
cod_equipe       object
colab_1           int32
colab_2           int32
cod_impedido      int32
cod_susp         object
dt_saida         object
dt_inicio        object
dt_fim           object
dt_termino       object
cod_local         int32
sub_regiao       object
km                int32
dtype: object
```

```
# verificar os valores nulos em cada coluna do dataset
df_emergencial[df_emergencial.columns].isnull().sum()
```

```
Out[33]: ordem_servico     0
cod_servico       0
cod_veiculo       0
cod_equipe        0
colab_1           0
colab_2           0
cod_impedido      0
cod_susp          0
dt_saida          0
dt_inicio         0
dt_fim            0
dt_termino        0
cod_local         0
sub_regiao       41
km                0
dtype: int64
```

```
# substituição dos valores NAN por um 'coringa' denominado '100000'
```

```
# substituição dos valores NAN por um 'coringa' denominado '100000'
para facilitar as tratativas futuras
df_emergencial=df_emergencial.fillna(100000)


# verificar que não existem mais NaN nas colunas do dataset
df_emergencial[df_emergencial.columns].isnull().sum()
```

```
Out[35]: ordem_servico    0
cod_servico      0
cod_veiculo      0
cod_equipe       0
colab_1          0
colab_2          0
cod_impedido     0
cod_susp         0
dt_saida         0
dt_inicio        0
dt_fim           0
dt_termino       0
cod_local        0
sub_regiao       0
km               0
dtype: int64
```

```
# Devido a ser somente 41 registro para um universo de 546558 mil,
sendo 0,007% dos registros, esta supresão não impactará no resultado
final do projeto
df_emergencial=df_emergencial.drop(df_emergencial[df_emergencial['sub_r
egiao'] == 100000].index)


# Verificando se os valores coringas foram tratados adquadamente
df_emergencial.loc[df_emergencial['sub_regiao'] == 100000]
```

ordem_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2	cod_impedido

```
# Conversa]ão das colunas 'dt_fim' em data e horas para as trataivas de
valores
df_emergencial['dt_fim'] = pd.to_datetime(df_emergencial['dt_fim'],
errors='coerce')


# Conversa]ão das colunas 'dt_fim' em data e horas para as trataivas de
valores
df_emergencial['dt_inicio'] =
pd.to_datetime(df_emergencial['dt_inicio'], errors='coerce')
# Conversa]ão das colunas 'dt_fim' em data e horas para as trataivas de
```

```
pd.to_datetime(df_emergencial['dt_inicio'], errors='coerce')
# Conversa]ão das colunas 'dt_fim' em data e horas para as trataivas de
valores
df_emergencial['dt_saida'] = pd.to_datetime(df_emergencial['dt_saida'],
errors='coerce')


# Conversão das colunas 'dt_fim' em data e horas para as trataivas de
valores
df_emergencial['dt_termino'] =
pd.to_datetime(df_emergencial['dt_termino'], errors='coerce')


# separando as variáveis numericas
nums_emerg = ['ordem_servico', 'cod_servico', 'colab_1','colab_2',
'cod_impedido','cod_local','km']


# apresenta um resumo de estatistica das variáveis numericas
df_emergencial[nums_emerg].describe()
```

	cod_servico	colab_1	colab_2	cod_impedido	cod_local	
count	546517.000000	5.465170e+05	5.465170e+05	546517.000000	546517.000000	546517
mean	6001.441139	8.358465e+05	8.525444e+05	38.369319	4288.326848	19
std	0.496524	4.479010e+05	4.580723e+05	31.653557	1910.888147	21
min	6001.000000	6.020400e+04	0.000000e+00	1.000000	1006.000000	1
25%	6001.000000	8.645520e+05	8.945940e+05	4.000000	2512.000000	5
50%	6001.000000	9.730230e+05	9.796080e+05	39.000000	4246.000000	12
75%	6002.000000	9.867450e+05	9.915300e+05	71.000000	5794.000000	27
max	6002.000000	2.411964e+06	2.412246e+06	98.000000	7882.000000	485

3. Dataset tb_imped_cortes

```
# Carregando o datasets com códigos de impedimentos de cortes
query = """
SELECT *
FROM `hive_metastore`.`default`.`tb_corte_impedido`
"""
df_spark = spark.sql(query)
df_corte_impedido = df_spark.toPandas()


# verificando o dataframes o carregamentos dos dados
df_corte_impedido.shape

Out[45]: (46, 2)
# verificando o dataframes o carregamentos dos dados
```

```
Out[45]: (46, 2)
# verificando o dataframes o carregamentos dos dados
df_corte_impedido.head()
```

	cod	situacao
0	0	Corte_Executado
1	1	Corte_Executado
2	2	Corte_Executado
3	3	Corte_Executado
4	4	Corte_Executado

```
# anãlise dos tipos dos dados em cada atributo do dataframe
df_corte_impedido.dtypes
```

```
Out[47]: cod          int32
situacao    object
dtype: object
```

```
# verificar os tipos de dados por colunas e valores nulos
df_corte_impedido.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46 entries, 0 to 45
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   cod       46 non-null     int32
 1   situacao  46 non-null     object
dtypes: int32(1), object(1)
memory usage: 680.0+ bytes
```

```
# substituido o código "0" por "10000" devido ao ´codigo '0' apresentar
duplicidade com o mesmo valor na coluna 'cod_impedimentos' e divergir
nos resultados das conclusões dos serviços comerciais
df_corte_impedido["cod"] = df_corte_impedido["cod"].replace({0:10000})
```

```
# verificando o dataframes o carregamentos dos dados
df_corte_impedido.head()
```

	cod	situacao
0	10000	Corte_Executado

	cod	situacao
0	10000	Corte_Executado
1	1	Corte_Executado
2	2	Corte_Executado
3	3	Corte_Executado
4	4	Corte_Executado

4. Dataset tb_imped_comerciais

```
# Carregando o datasets com códigos de impedimentos de cortes
query = """
SELECT *
FROM `hive_metastore`.`default`.`tb_comercial_impedido`
"""
df_spark = spark.sql(query)
df_comercial_impedido = df_spark.toPandas()
```

```
df_comercial_impedido.shape
```

```
Out[52]: (143, 2)
```

```
# verificando o dataframes o carregamentos dos dados
df_comercial_impedido.head(3)
```

	cod	situacao
0	100	Com_Impedido
1	111	Com_Impedido
2	112	Com_Impedido

```
df_comercial_impedido.loc[0]=[0,'Com_Executado']
```

```
# verificando o dataframes o carregamentos dos dados
df_comercial_impedido.head(3)
```

	cod	situacao
0	0	Com_Executado
1	111	Com_Impedido
2	112	Com_Impedido

```
# anãlise dos tipos dos dados em cada atributo do dataframe
```

```
# anãlise dos tipos dos dados em cada atributo do dataframe
df_comercial_impedido.dtypes
```

```
Out[56]: cod          int32
situacao    object
dtype: object
```

```
# verificar os tipos de dados por colunas e valores nulos
df_comercial_impedido.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 143 entries, 0 to 142
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   cod       143 non-null    int32
 1   situacao  143 non-null    object
dtypes: int32(1), object(1)
memory usage: 1.8+ KB
```

```
# concatenar os datasets, imped_cortes e imped_comerciais
df_ImpedComerciais = pd.concat([df_corte_impedido,
df_comercial_impedido], axis=0)

# reseta o index if needed
df_ImpedComerciais.reset_index(drop=True, inplace=True)
```

```
# verificando o dataframes o carregamentos dos dados
df_ImpedComerciais.head(3)
```

	cod	situacao
0	10000	Corte_Executado
1	1	Corte_Executado
2	2	Corte_Executado

```
# verificando o dataframes o carregamentos dos dados
df_ImpedComerciais.tail(3)
```

	cod	situacao
186	8903	Com_Impedido
187	8904	Com_Impedido
188	8905	Com_Impedido

5. Dataset tb_emer_causa_imped

5. Dataset tb_emer_causa_imped

```
# Carregando o datasets com códigos de impedimentos de cortes
query = """
SELECT *
FROM `hive_metastore`.`default`.`tb_emer_causa_imped`
"""
df_spark = spark.sql(query)
df_emer_causa_imped = df_spark.toPandas()
```

```
# anãlise dos tipos dos dados em cada atributo do dataframe
df_emer_causa_imped.shape
```

```
Out[62]: (75, 2)
```

```
# verificando o dataframes o carregamentos dos dados
df_emer_causa_imped.head(3)
```

	cod	situacao
0	01	Executada
1	02	Executada
2	03	Executada

```
# anãlise dos tipos dos dados em cada atributo do dataframe
df_emer_causa_imped.dtypes
```

```
Out[64]: cod         object
situacao    object
dtype: object
```

```
# verificar os tipos de dados por colunas e valores nulos
df_emer_causa_imped.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75 entries, 0 to 74
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   cod       75 non-null     object
 1   situacao  75 non-null     object
dtypes: object(2)
memory usage: 1.3+ KB
```

6. Dataset tb_emer_componente_imped

6. Dataset tb_emer_componente_imped

```
# Carregando o datasets com códigos de impedimentos de cortes
query = """
SELECT *
FROM `hive_metastore`.`default`.`tb_emer_componente_imped`
"""
df_spark = spark.sql(query)
df_emer_componente_imped = df_spark.toPandas()
```

```
# verificando o dataframes o carregamentos dos dados
df_emer_componente_imped.shape
```

```
Out[67]: (129, 2)
```

```
# verificando o dataframes o carregamentos dos dados
df_emer_componente_imped.head(3)
```

	cod	situacao
0	01	Emer_
1	02	Emer_
2	03	Emer_

```
# anãlise dos tipos dos dados em cada atributo do dataframe
df_emer_componente_imped.dtypes
```

```
Out[69]: cod         object
situacao    object
dtype: object
```

```
# verificar os tipos de dados por colunas e valores nulos
df_emer_componente_imped.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129 entries, 0 to 128
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   cod       129 non-null    object
 1   situacao  129 non-null    object
dtypes: object(2)
memory usage: 2.1+ KB
```

7. Dataset tb_emer_componente_imped

7. Dataset tb_emer_componente_imped

```
# Carregando o datasets com códigos de impedimentos de cortes
query = """
SELECT *
FROM `hive_metastore`.`default`.`tb_grupo_servicos`
"""
df_spark = spark.sql(query)
df_grupo_servicos = df_spark.toPandas()


# verificando o dataframes o carregamentos dos dados
df_grupo_servicos.shape
```

```
Out[72]: (184, 2)
```

```
# verificando o dataframes o carregamentos dos dados
df_grupo_servicos.head()
```

	codigo	Grupo_100
0	101	Grupo_100
1	102	Grupo_100
2	103	Grupo_100
3	111	Grupo_100
4	112	Grupo_100

```
# anãlise dos tipos dos dados em cada atributo do dataframe
df_grupo_servicos.dtypes
```

```
Out[74]: codigo        int32
Grupo_100    object
dtype: object
```

```
# verificar os tipos de dados por colunas e valores nulos
df_grupo_servicos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 184 entries, 0 to 183
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   codigo     184 non-null    int32
 1   Grupo_100  184 non-null    object
dtypes: int32(1), object(1)
memory usage: 2.3+ KB
```

8. Dataset tb_regioes

```
memory usage: 2.3+ KB
```

8. Dataset tb_regioes

```
# Carregando o datasets com códigos de impedimentos de cortes
query = """
SELECT *
FROM `hive_metastore`.`default`.`tb_regioes`
"""
df_spark = spark.sql(query)
df_regioes = df_spark.toPandas()
```

```
# verificando o dataframes o carregamentos dos dados
df_regioes.shape
```

```
Out[77]: (1151, 3)
```

```
# verificando o dataframes o carregamentos dos dados
df_regioes.head()
```

	codigo	dpto	gerencia
0	2350	DPTO01	AG001
1	3388	DPTO01	AG001
2	1006	DPTO01	AG001
3	7360	DPTO01	AG001
4	4894	DPTO01	AG001

```
# anãlise dos tipos dos dados em cada atributo do dataframe
df_regioes.dtypes
```

```
Out[79]: codigo       int32
dpto        object
gerencia    object
dtype: object
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1151 entries, 0 to 1150
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   codigo    1151 non-null   int32
 1   dpto      1151 non-null   object
 2   gerencia  1151 non-null   object
dtypes: int32(1), object(2)
memory usage: 22.6+ KB
```