Detalhamento do MVP

Nome: Talita Lemes Nogueira

Sumário

1. Objetivo

Gerar análises através da comparação da idade das instalações e a quantidade em horas das manutenções programadas ou intercorrências forçadas que geraram restrição em máquinas que pertençam a sua estrutura.

As dúvidas a serem respondidas orbitam em entender os fatores podem ser determinantes para que uma máquina ou instalação tenha mais movimentações restritivas, ou seja, limitações de operação ou interrupções de operação do que outras. Por exemplo, uma instalação mais antiga, com mais de 20 anos em operação terá mais horas de manutenção programada do que as instalações mais novas? Dessas instalações que consomem mais horas com manutenção são de que tipo? Qual combustível? Todos esses pontos serão complementados com mais dúvidas respondidas no decorrer do trabalho.

Conceitualizando: Uma instalação pode ter uma ou mais máquinas em sua estrutura. As instalações são de um determinado tipo, possuem data de entrada em operação, data de desativação e as máquinas associadas.

As máquinas consomem um determinado combustível, possuem data início de operação, data de início de apuração e data de desativação.

Cada máquina possui movimentações que indicam o comportamento dela dentro de uma janela de tempo. Ou seja, ela precisou ser desligada para uma manutenção programada? Ela sofreu algum impacto surpresa que gerou uma manutenção forçada? Essas movimentações são encontradas na tabela mov.

Cada movimentação está associada a uma máquina que pertence a uma instalação, a movimentação tem data e hora da ocorrência e uma classificação que pode ser determinada pelas colunas EO, CO, OR. Essas classificações indicam qual a movimentação que está iniciando a partir daquela data ocorrência e se a limitação foi gerada por uma causa forçada ou programada.

Vamos analisar as movimentações e suas durações para entender qual o impacto da data de entrada em operação das instalações nas movimentações.

2. Plataforma

Databricks Community Edition.

3. Busca pelos dados

Estou utilizando uma base que indica as movimentações das máquinas das instalações de geração de energia em uma janela do tempo. Os dados que identificam as instalações e as máquinas foram alteradas para permitir que não sejam identificadas.

O recorte da base foi menor, devido ao seu tamanho original, por isso não teremos uma visão fidedigna da situação. Porém, entendo que o tratamento e a análise serão os mesmos quando utilizarmos a base total.

4. Coleta

Os dados foram tratados e convertidos em .csv para serem consultados pelo notebook do databricks.

Dos três arquivos que serão trabalhados, dois estão no github e o maior está no databricks devido ao limite de upload do github.

Os arquivos do github são:

- instalacaocompv2.csv
- maquinacompv2.csv

o arquivo que está no databricks:

- movimentacaov2.csv

5. Modelagem

a. Catálogo de Dados

Tabela: tb_inst

Descrição: Armazena informações sobre instalações.

Campos:

- inst_id (PK):	INT	Identificador único da instalação.
- tpins:	STRING	Tipo de instalação.
- din_entrada:	DATE	Data de entrada em operação da instalação
- din_desativacao:	DATE	Data de desativação da instalação

Tabela: tb_maq

Descrição: Armazena informações sobre máquinas associadas a uma instalação.

Campos:

- maq_id (PK):	INT	Identificador único da máquina.
- inst_id (FK):	INT	Identificador único da instalação.
- tpcomb:	STRING	Tipo de combustível

- dtini_oper:	DATE	Data de entrada em operação da máquina
- din_des:	DATE	Data de desativação da máquina
- din_iniapur:	DATE	Data início da apuração da máquina

Tabela: tb_movacum

Descrição: Armazena movimentações de máquinas pertencentes a uma instalação.

Campos:

- mov_id (PK):	INT	Identificador único da movimentação.
- inst_id (FK1):	INT	Referência para a instalação associada à movimentação.
- maq_id (FK2):	INT	Referência para a máquina associada à movimentação.
- data_ocorrencia:	DATE	Data da ocorrência da movimentação.
- eo:	STRING	Estado de movimentação.
- co:	STRING	Código de movimentação.
- or:	STRING	Outro dado de movimentação

Tabela: tb_movcoacum

Descrição: Armazena durações acumuladas decorrentes de cada movimentação.

Campos:

- movdur_id (PK):	INT	Identificador único da duração acumulada.
- inst_id (FK1):	INT	Referência para a instalação associada à duração.
- maq_id (FK2): I	NT	Referência para a máquina associada à duração.
- mov_id (FK3):	INT	Referência para a movimentação associada à duração.
- data_anterior:	TIMESTAMP	Data anterior relevante para o cálculo da duração.
- duracao:	DOUBLE	Duração da movimentação.
- soma_duracao_prog:	DOUBLE	Soma da duração programada.
- soma_duracao_nula:	DOUBLE	Soma da duração de desligamento total.
- soma_duracao_nor:	DOUBLE	Soma da duração normal.

- soma_duracao_inst: DOUBLE Soma da duração por instalação.

Tabela: tb_instacum

Descrição: Armazena a duração acumulada das movimentações por instalação.

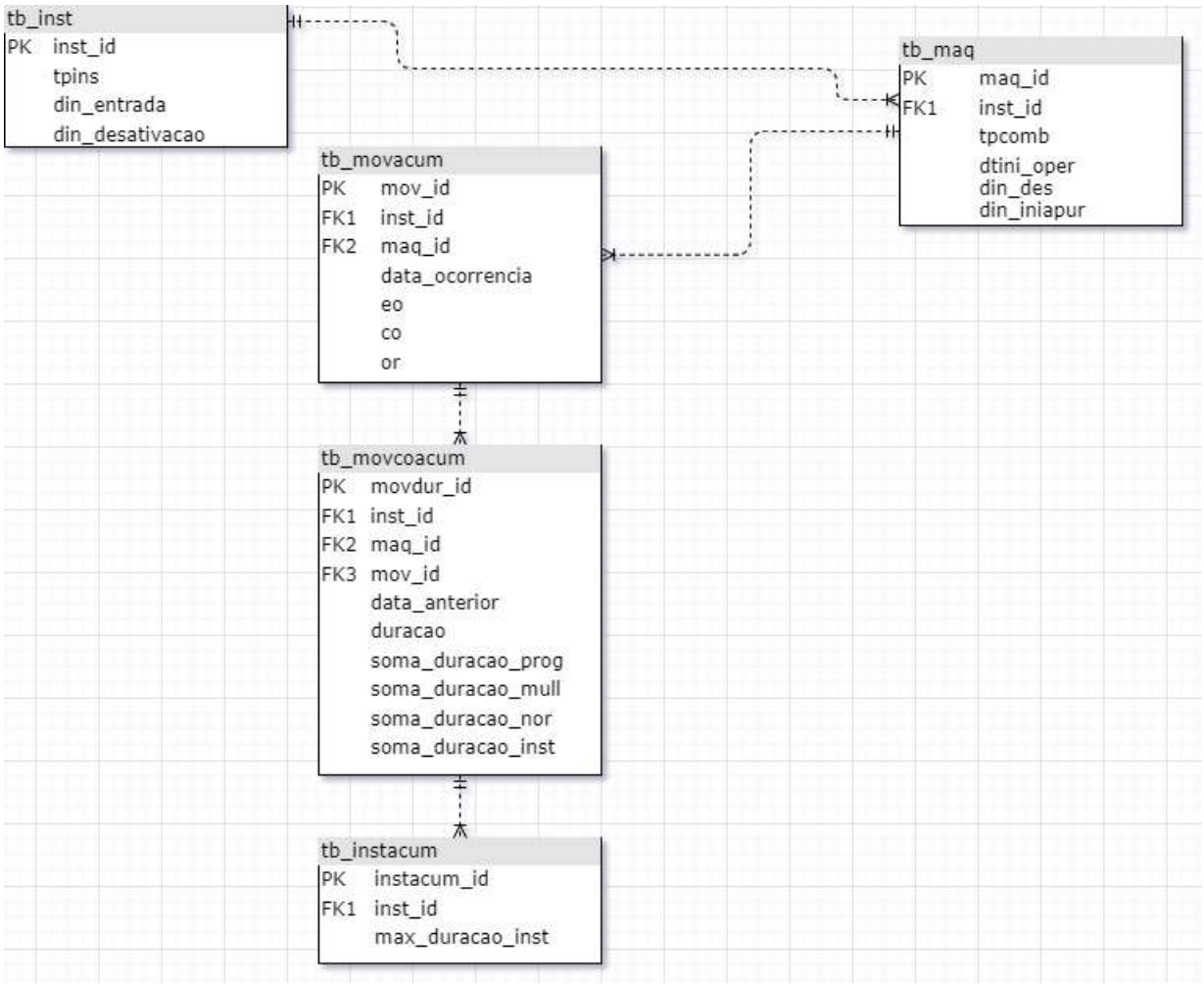
Campos:

- instacum_id (PK): INT Identificador único da duração acumulada da instalação.
- inst_id (FK): INT Referência para a instalação associada à duração.
- max_duracao_inst: DOUBLE Soma da duração instalação

Relacionamentos:

- inst - maq: Relacionamento 1:N (uma instalação pode ter várias máquinas).
- inst - tb_movacum: Relacionamento 1:N (uma instalação pode ter várias movimentações).
- maq - tb_movacum: Relacionamento 1:N (uma máquina pode ter várias movimentações).
- tb_movacum - tb_movcoacum: Relacionamento 1:1 (uma movimentação tem uma duração acumulada associada).
- inst - tb_instacum: Relacionamento 1:1 (uma instalação tem uma duração acumulada).

b. Modelo de dados



6. Carga

Início do código do notebook MPV_Manutencao

- Importação do Pandas e algumas bibliotecas

```
import pandas as pd
from pyspark.sql.functions import to_date, to_timestamp
from pyspark.sql import SparkSession
```

```
spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")
```

- Importação dos datasets

Dois arquivos estão no repositório do github e um no databricks devido ao tamanho do arquivo.

```
base = "https://raw.githubusercontent.com/talitanog/mvpii/main/instalacaocompv2.csv"
df = spark.createDataFrame(pd.read_csv(base, sep=";"))
```

df: pyspark.sql.dataframe.DataFrame = [tpins: string, inst: string ... mais 2 campos]

```
base = "https://raw.githubusercontent.com/talitanog/mvpii/main/maquinacompv2.csv"
df2 = spark.createDataFrame(pd.read_csv(base, sep=";"))
```

df2: pyspark.sql.dataframe.DataFrame = [instal: string, maquina: string ... mais 4 campos]

```
df3 = spark.read \
        .format("CSV") \
        .option("sep",";") \
        .option("header", "true") \
        .option("inferSchema", "true") \
        .load("/FileStore/tables/mvpii/movimentacaov2.csv")
```

(2) jobs Spark

df3: pyspark.sql.dataframe.DataFrame = [inst: string, maquina: string ... mais 4 campos]

- Criação de tabelas temporárias para testes iniciais.

```
df.createOrReplaceTempView("inst")
df2.createOrReplaceTempView("maquinas")
df3.createOrReplaceTempView("movimentacao")
```

- Resultados dos selects nas tabelas.
  A. Inst

```
%sql
select * from inst
```

_sqldf: pyspark.sql.dataframe.DataFrame = [tpins: string, inst: string ... mais 2 campos]

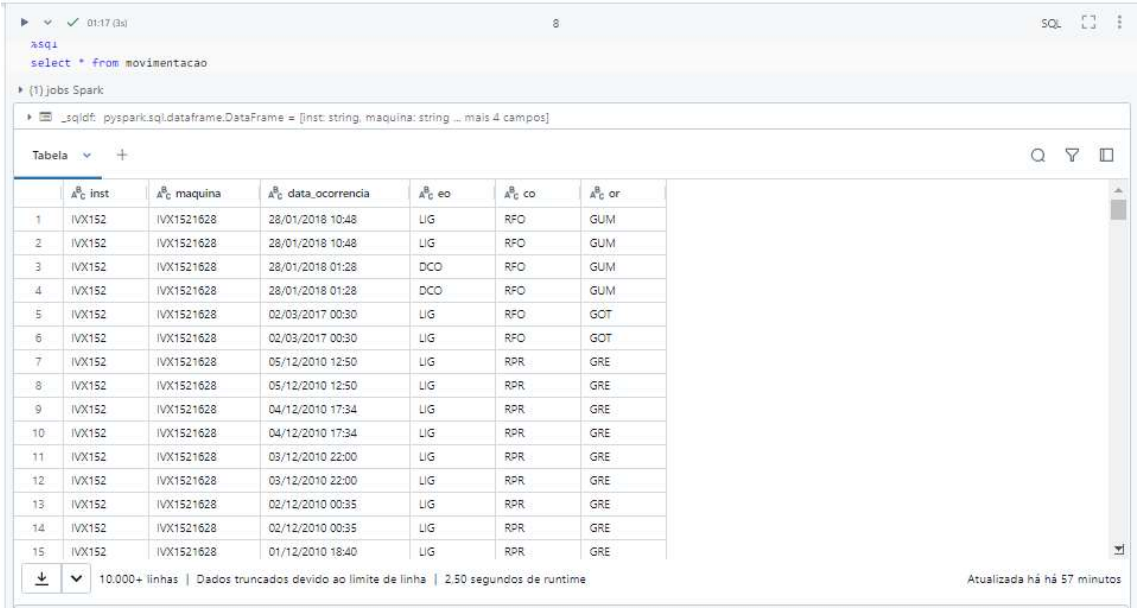
Tabela

	tpins	inst	din_entrada	din_desativacao
1	TER	IVX36	21/08/2002	01/02/2024
2	HID	IVX37	01/01/1967	null
3	TER	IVX38	01/01/2008	09/08/2022
4	HID	IVX39	01/04/1956	null
5	HID	IVX40	24/08/2004	null
6	HID	IVX41	09/05/1999	null
7	HID	IVX42	15/05/1999	null
8	TER	IVX43	17/11/2009	null
9	TER	IVX44	01/05/2002	null
10	TER	IVX45	20/04/2002	26/06/2003
11	TER	IVX46	17/04/2002	26/06/2003
12	TER	IVX47	16/07/2002	null
13	NUC	IVX48	25/02/2006	null
14	HID	IVX49	01/04/1969	null
15	HID	IVX50	23/01/2003	null

339 linhas | 1,22 segundo de runtime

Atualizada há há 56 minutos

  B. movimentacao



```
%sql
select * from movimentacao
```

(1) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [inst: string, maquina: string ... mais 4 campos]

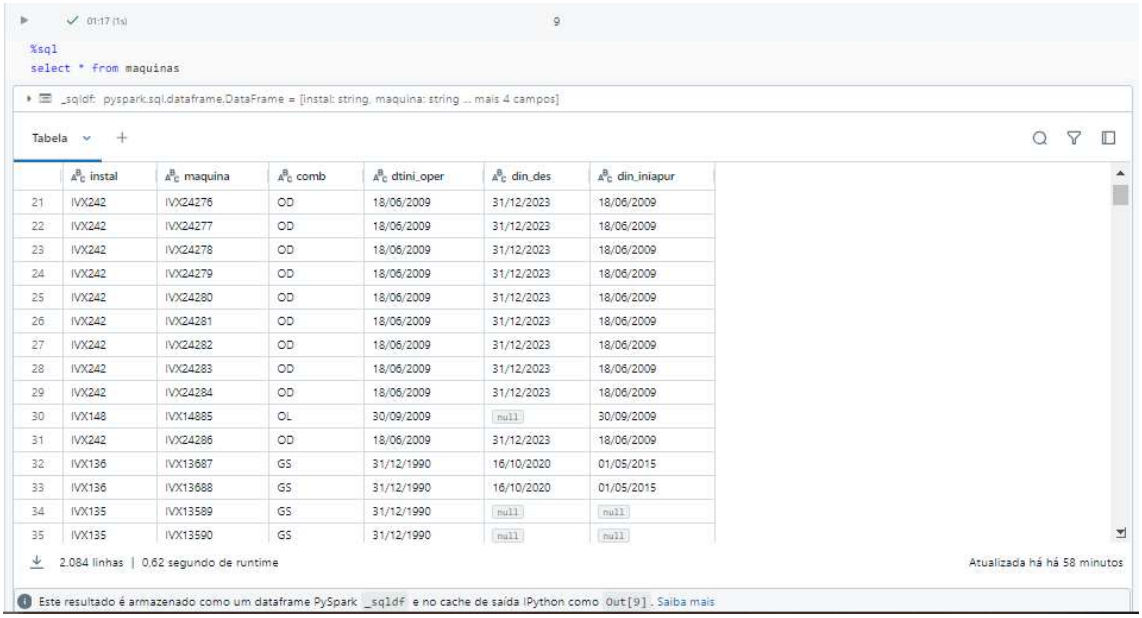
Tabela

	inst	maquina	data_ocorrencia	eo	co	or
1	IVX152	IVX1521628	28/01/2018 10:48	LIG	RFO	GUM
2	IVX152	IVX1521628	28/01/2018 10:48	LIG	RFO	GUM
3	IVX152	IVX1521628	28/01/2018 01:28	DCO	RFO	GUM
4	IVX152	IVX1521628	28/01/2018 01:28	DCO	RFO	GUM
5	IVX152	IVX1521628	02/03/2017 00:30	LIG	RFO	GOT
6	IVX152	IVX1521628	02/03/2017 00:30	LIG	RFO	GOT
7	IVX152	IVX1521628	05/12/2010 12:50	LIG	RPR	GRE
8	IVX152	IVX1521628	05/12/2010 12:50	LIG	RPR	GRE
9	IVX152	IVX1521628	04/12/2010 17:34	LIG	RPR	GRE
10	IVX152	IVX1521628	04/12/2010 17:34	LIG	RPR	GRE
11	IVX152	IVX1521628	03/12/2010 22:00	LIG	RPR	GRE
12	IVX152	IVX1521628	03/12/2010 22:00	LIG	RPR	GRE
13	IVX152	IVX1521628	02/12/2010 00:35	LIG	RPR	GRE
14	IVX152	IVX1521628	02/12/2010 00:35	LIG	RPR	GRE
15	IVX152	IVX1521628	01/12/2010 18:40	LIG	RPR	GRE

10.000+ linhas | Dados truncados devido ao limite de linha | 2.50 segundos de runtime

Atualizada há há 57 minutos

C. maquinas



```
%sql
select * from maquinas
```

_sqldf: pyspark.sql.dataframe.DataFrame = [instal: string, maquina: string ... mais 4 campos]

Tabela

	instal	maquina	comb	dtini_oper	din_des	din_iniapur
21	IVX242	IVX24276	OD	18/06/2009	31/12/2023	18/06/2009
22	IVX242	IVX24277	OD	18/06/2009	31/12/2023	18/06/2009
23	IVX242	IVX24278	OD	18/06/2009	31/12/2023	18/06/2009
24	IVX242	IVX24279	OD	18/06/2009	31/12/2023	18/06/2009
25	IVX242	IVX24280	OD	18/06/2009	31/12/2023	18/06/2009
26	IVX242	IVX24281	OD	18/06/2009	31/12/2023	18/06/2009
27	IVX242	IVX24282	OD	18/06/2009	31/12/2023	18/06/2009
28	IVX242	IVX24283	OD	18/06/2009	31/12/2023	18/06/2009
29	IVX242	IVX24284	OD	18/06/2009	31/12/2023	18/06/2009
30	IVX148	IVX14885	OL	30/09/2009	null	30/09/2009
31	IVX242	IVX24286	OD	18/06/2009	31/12/2023	18/06/2009
32	IVX136	IVX13687	GS	31/12/1990	16/10/2020	01/05/2015
33	IVX136	IVX13688	GS	31/12/1990	16/10/2020	01/05/2015
34	IVX135	IVX13589	GS	31/12/1990	null	null
35	IVX135	IVX13590	GS	31/12/1990	null	null

2.084 linhas | 0.62 segundo de runtime

Atualizada há há 58 minutos

Este resultado é armazenado como um dataframe PySpark `_sqldf` e no cache de saída IPython como `Out[9]`. Saiba mais

D. Contagem dos tipos de classificadores de manutenção

01:12 (12s) 12 SQL

```
%sql

SELECT
    SUM(CASE WHEN CO = 'NOR' OR CO = 'NOT' OR CO = 'TST' THEN 1 ELSE 0 END) AS cond_nor,
    SUM(CASE WHEN CO = 'RFO' THEN 1 ELSE 0 END) AS cond_for,
    SUM(CASE WHEN CO = 'RPR' THEN 1 ELSE 0 END) AS cond_prog,
    SUM(CASE WHEN CO IS NULL THEN 1 ELSE 0 END) AS cond_desl
FROM movimentacao;
```

(2) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [cond_nor: long, cond_for: long ... mais 2 campos]

Tabela

	cond_nor	cond_for	cond_prog	cond_desl
1	549	1010902	37124	0

1 linha | 12.06 segundos de runtime — Atualizada há há 1 hora

Este resultado é armazenado como um dataframe PySpark `_sqldf` e no cache de saída IPython como `Out[7]`. Saiba mais

01:17 (1s) 10

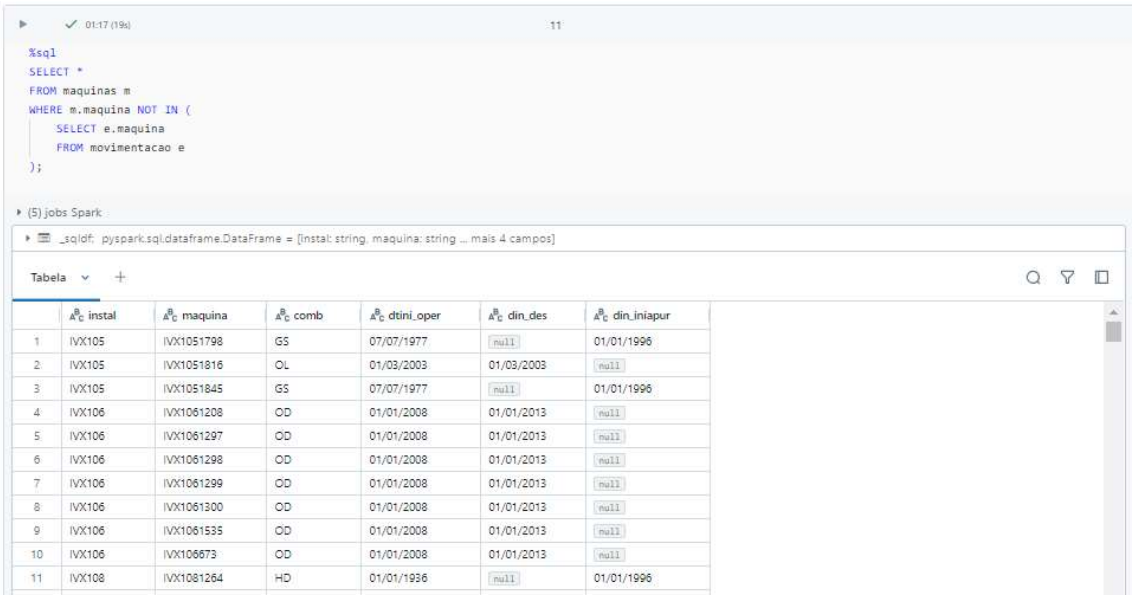
```
display(df3)
```

(1) jobs Spark

Tabela

	inst	maquina	data_ocorrencia	eo	co	or
1	IVX152	IVX1521628	28/01/2018 10:48	LIG	RFO	GUM
2	IVX152	IVX1521628	28/01/2018 10:48	LIG	RFO	GUM
3	IVX152	IVX1521628	28/01/2018 01:28	DCO	RFO	GUM
4	IVX152	IVX1521628	28/01/2018 01:28	DCO	RFO	GUM
5	IVX152	IVX1521628	02/03/2017 00:30	LIG	RFO	GOT
6	IVX152	IVX1521628	02/03/2017 00:30	LIG	RFO	GOT
7	IVX152	IVX1521628	05/12/2010 12:50	LIG	RPR	GRE
8	IVX152	IVX1521628	05/12/2010 12:50	LIG	RPR	GRE
9	IVX152	IVX1521628	04/12/2010 17:34	LIG	RPR	GRE
10	IVX152	IVX1521628	04/12/2010 17:34	LIG	RPR	GRE
11	IVX152	IVX1521628	03/12/2010 22:00	LIG	RPR	GRE
12	IVX152	IVX1521628	03/12/2010 22:00	LIG	RPR	GRE
13	IVX152	IVX1521628	02/12/2010 00:35	LIG	RPR	GRE
14	IVX152	IVX1521628	02/12/2010 00:35	LIG	RPR	GRE
15	IVX152	IVX1521628	01/12/2010 18:40	LIG	RPR	GRE

10.000+ linhas | Dados truncados devido ao limite de linha | 0.88 segundo de runtime — Atualizada há há 59 minutos



```
%sql
SELECT *
FROM maquinas m
WHERE m.maquina NOT IN (
    SELECT e.maquina
    FROM movimentacao e
);
```

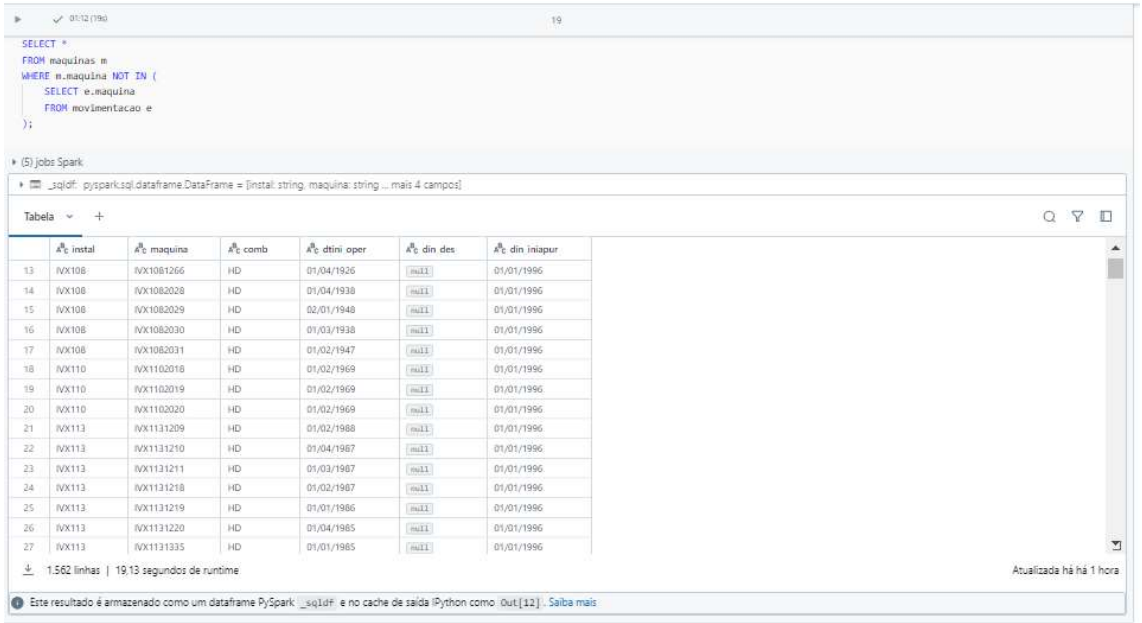
(5) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [instal: string, maquina: string ... mais 4 campos]

Tabela

	instal	maquina	comb	dtini_oper	din_des	din_iniapur
1	IVX105	IVX1051798	GS	07/07/1977	null	01/01/1996
2	IVX105	IVX1051816	OL	01/03/2003	01/03/2003	null
3	IVX105	IVX1051845	GS	07/07/1977	null	01/01/1996
4	IVX106	IVX1061208	OD	01/01/2008	01/01/2013	null
5	IVX106	IVX1061297	OD	01/01/2008	01/01/2013	null
6	IVX106	IVX1061298	OD	01/01/2008	01/01/2013	null
7	IVX106	IVX1061299	OD	01/01/2008	01/01/2013	null
8	IVX106	IVX1061300	OD	01/01/2008	01/01/2013	null
9	IVX106	IVX1061535	OD	01/01/2008	01/01/2013	null
10	IVX106	IVX106673	OD	01/01/2008	01/01/2013	null
11	IVX108	IVX1081264	HD	01/01/1936	null	01/01/1996

- Início do tratamento de dados
  A. Consultar as máquinas que não possuem movimentação



```
SELECT *
FROM maquinas m
WHERE m.maquina NOT IN (
    SELECT e.maquina
    FROM movimentacao e
);
```

(5) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [instal: string, maquina: string ... mais 4 campos]

Tabela

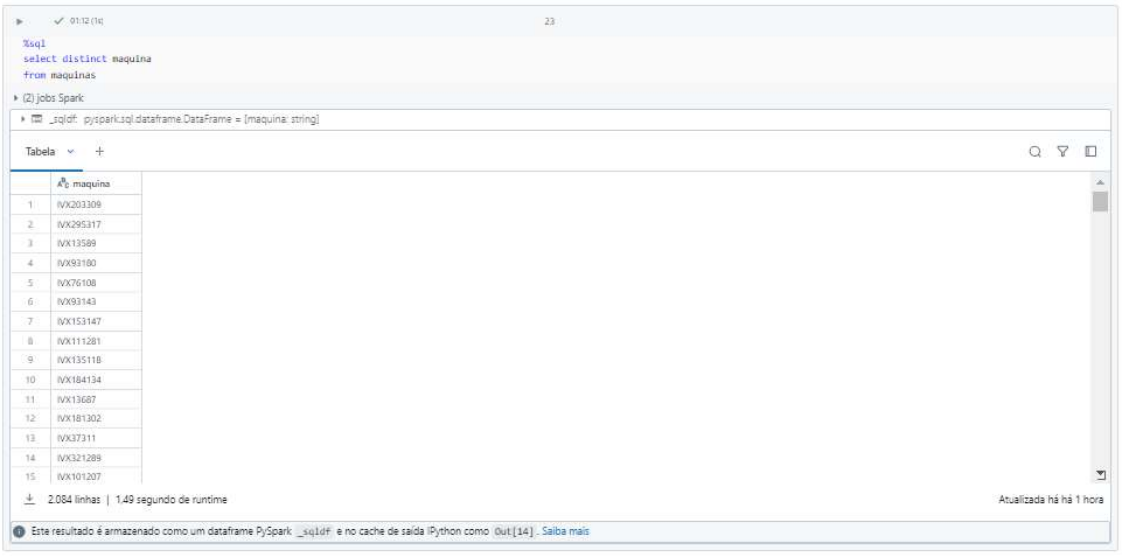
	instal	maquina	comb	dtini_oper	din_des	din_iniapur
13	IVX108	IVX1081266	HD	01/04/1926	null	01/01/1996
14	IVX108	IVX1082028	HD	01/04/1938	null	01/01/1996
15	IVX108	IVX1082029	HD	02/01/1948	null	01/01/1996
16	IVX108	IVX1082030	HD	01/03/1938	null	01/01/1996
17	IVX108	IVX1082031	HD	01/02/1947	null	01/01/1996
18	IVX110	IVX1102018	HD	01/02/1969	null	01/01/1996
19	IVX110	IVX1102019	HD	01/02/1969	null	01/01/1996
20	IVX110	IVX1102020	HD	01/02/1969	null	01/01/1996
21	IVX113	IVX1131209	HD	01/02/1988	null	01/01/1996
22	IVX113	IVX1131210	HD	01/04/1967	null	01/01/1996
23	IVX113	IVX1131211	HD	01/03/1967	null	01/01/1996
24	IVX113	IVX1131218	HD	01/02/1907	null	01/01/1996
25	IVX113	IVX1131219	HD	01/01/1986	null	01/01/1996
26	IVX113	IVX1131220	HD	01/04/1985	null	01/01/1996
27	IVX113	IVX1131335	HD	01/01/1985	null	01/01/1996

1.562 linhas | 19,13 segundos de runtime

Atualizada há há 1 hora

Este resultado é armazenado como um dataframe PySpark `_sqldf` e no cache de saída IPython como `Out[12]`. Saiba mais

B. Total de máquinas da tabela máquinas



```
%sql
select distinct maquina
from maquinas
```

(2) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [maquina: string]

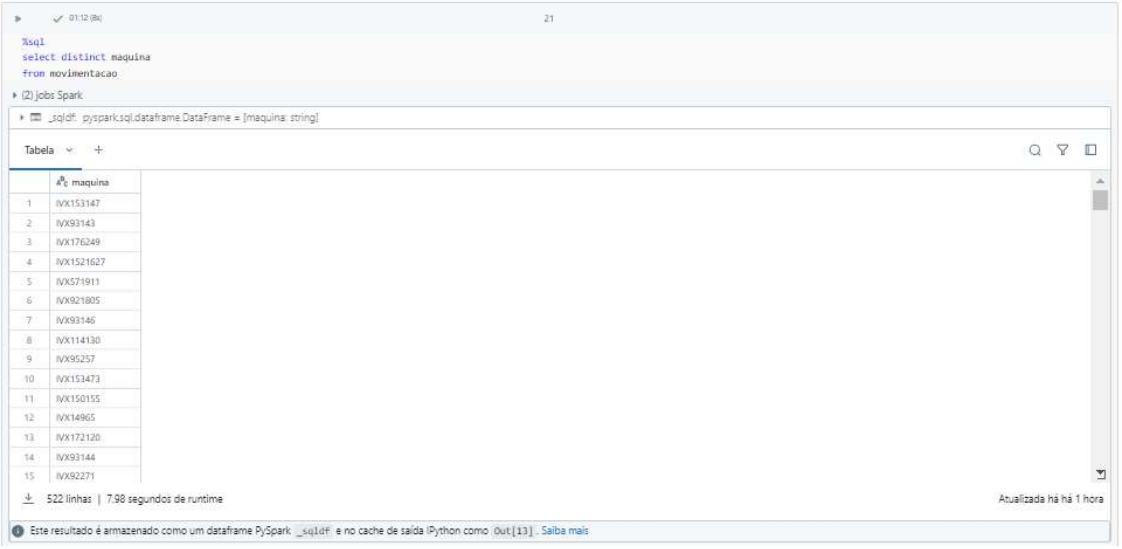
Tabela

	maquina
1	IVX203309
2	IVX295317
3	IVX13589
4	IVX93180
5	IVX76108
6	IVX93143
7	IVX153147
8	IVX111281
9	IVX135118
10	IVX184134
11	IVX13687
12	IVX181302
13	IVX37311
14	IVX321289
15	IVX101207

2.084 linhas | 1.49 segundo de runtime — Atualizada há há 1 hora

Este resultado é armazenado como um dataframe PySpark `_sqldf` e no cache de saída IPython como `Out[14]`. Saiba mais

C. Lista de máquinas que deverão ser mantidas no trabalho



```
%sql
select distinct maquina
from movimentacao
```

(2) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [maquina: string]

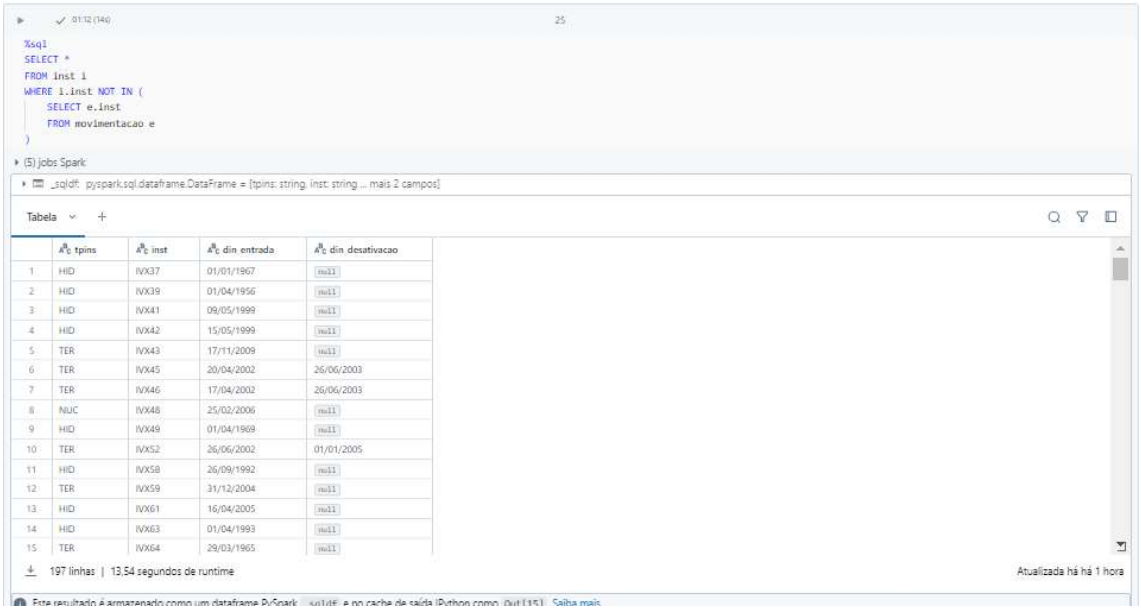
Tabela

	maquina
1	IVX153147
2	IVX93143
3	IVX176249
4	IVX1521627
5	IVX571911
6	IVX921805
7	IVX93146
8	IVX114130
9	IVX95257
10	IVX153473
11	IVX150155
12	IVX14965
13	IVX172120
14	IVX93144
15	IVX92271

522 linhas | 7.98 segundos de runtime — Atualizada há há 1 hora

Este resultado é armazenado como um dataframe PySpark `_sqldf` e no cache de saída IPython como `Out[13]`. Saiba mais

D. Confirmação das instalações que são encontradas em movimentação



```
%sql
SELECT *
FROM inst i
WHERE i.inst NOT IN (
    SELECT e.inst
    FROM movimentacao e
)
```

(5) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [tpins: string, inst: string ... mais 2 campos]

Tabela

	tpins	inst	din_entrada	din_desativacao
1	HID	IVX37	01/01/1967	null
2	HID	IVX39	01/04/1956	null
3	HID	IVX41	09/05/1999	null
4	HID	IVX42	15/05/1999	null
5	TER	IVX43	17/11/2009	null
6	TER	IVX45	20/04/2002	26/06/2003
7	TER	IVX46	17/04/2002	26/06/2003
8	NUC	IVX48	25/02/2006	null
9	HID	IVX49	01/04/1969	null
10	TER	IVX52	26/06/2002	01/01/2005
11	HID	IVX58	26/09/1992	null
12	TER	IVX59	31/12/2004	null
13	HID	IVX61	16/04/2005	null
14	HID	IVX63	01/04/1993	null
15	TER	IVX64	29/03/1965	null

197 linhas | 13.54 segundos de runtime

Atualizada há 1 hora

Este resultado é armazenado como um dataframe PySpark _sqldf e no cache de saída IPython como Out[15]. Saiba mais

E. Contagem do número de instalações

```
%sql
SELECT count(1)
FROM inst
```

(2) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [count(1): long]

Tabela

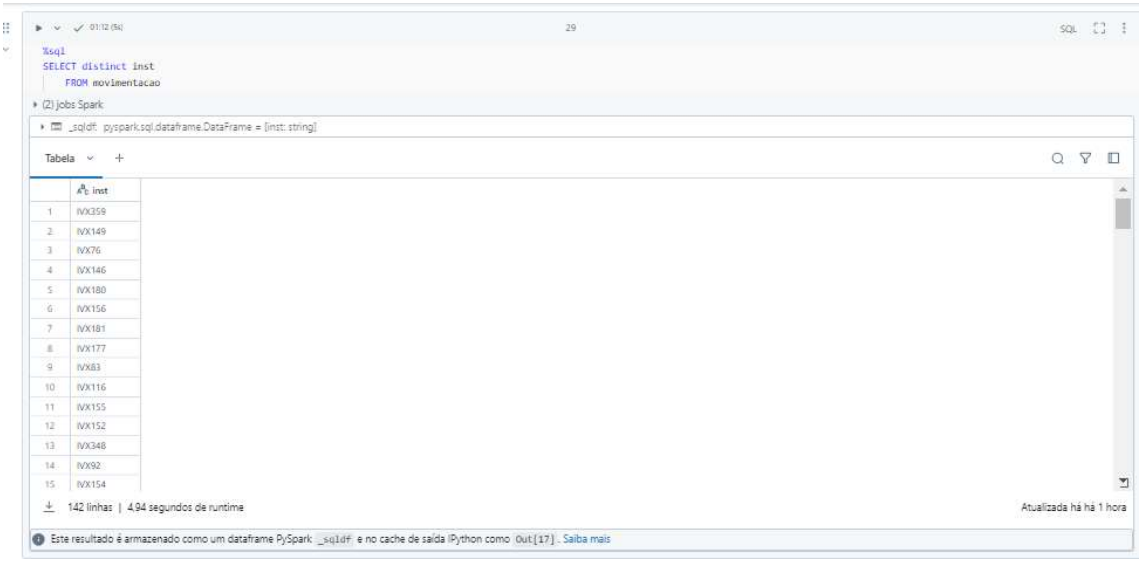
	count(1)
1	339

1 linha | 0.86 segundo de runtime

Atualizada há 1 hora

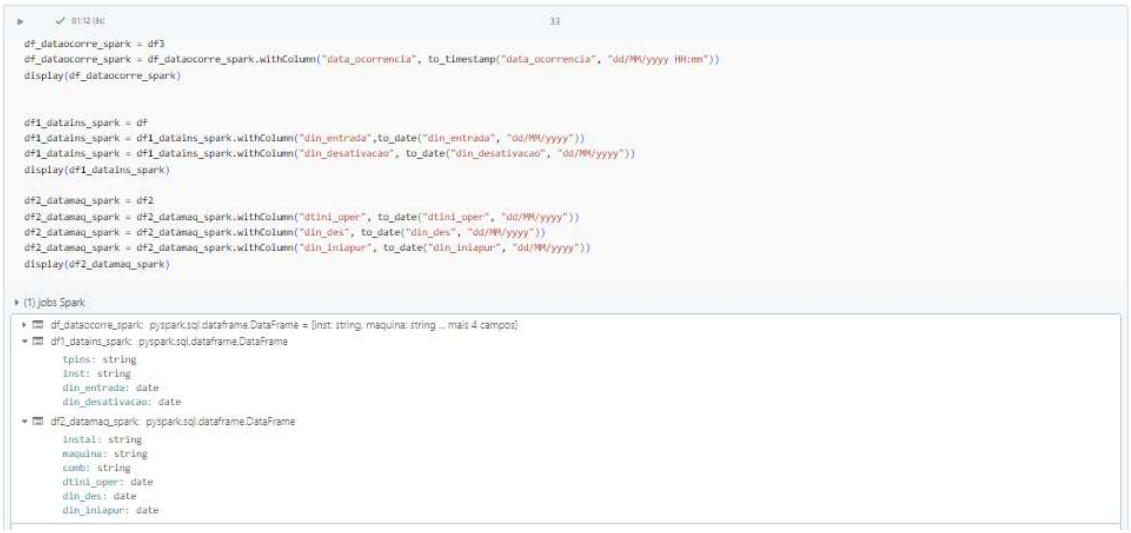
Este resultado é armazenado como um dataframe PySpark _sqldf e no cache de saída IPython como Out[16]. Saiba mais

F. Contagem do instalações encontradas na tabela movimentação

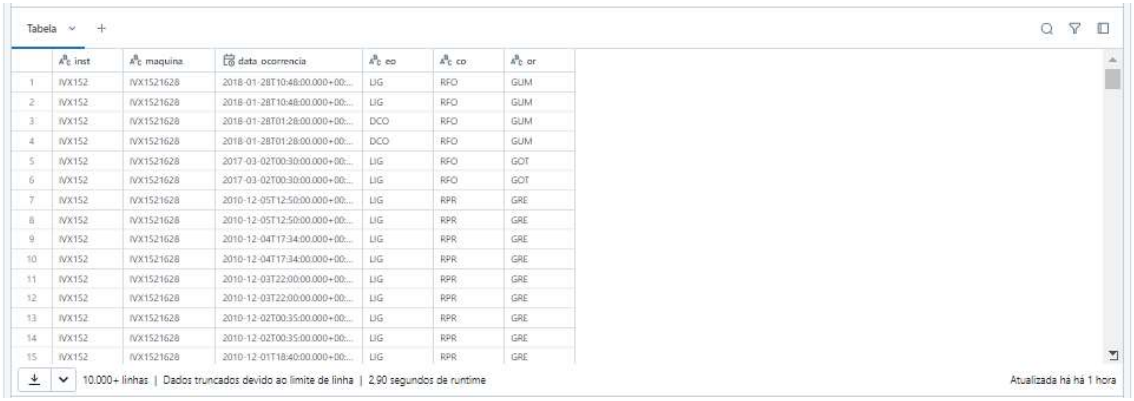


G. Ajustando o tipo de atributos das tabelas

H. Transformando as colunas de string em datas e timestamp



I. Resultados

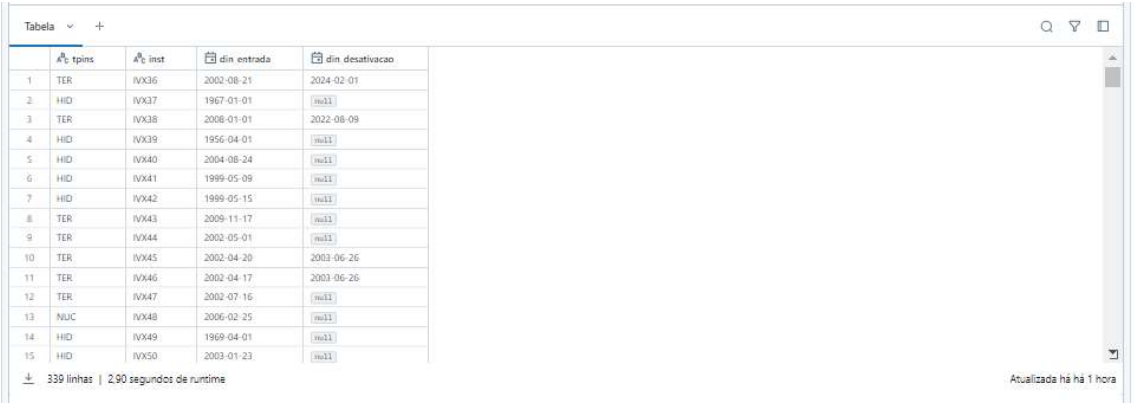


Tabela

	inst	maquina	data_ocorrencia	eo	co	or
1	IVX152	IVX1521628	2018-01-28T10:48:00.000+00:...	LIG	RFO	GUM
2	IVX152	IVX1521628	2018-01-28T10:48:00.000+00:...	LIG	RFO	GUM
3	IVX152	IVX1521628	2018-01-28T01:28:00.000+00:...	DCO	RFO	GUM
4	IVX152	IVX1521628	2018-01-28T01:28:00.000+00:...	DCO	RFO	GUM
5	IVX152	IVX1521628	2017-03-02T00:30:00.000+00:...	LIG	RFO	GOT
6	IVX152	IVX1521628	2017-03-02T00:30:00.000+00:...	LIG	RFO	GOT
7	IVX152	IVX1521628	2010-12-05T12:50:00.000+00:...	LIG	RPR	GRE
8	IVX152	IVX1521628	2010-12-05T12:50:00.000+00:...	LIG	RPR	GRE
9	IVX152	IVX1521628	2010-12-04T17:34:00.000+00:...	LIG	RPR	GRE
10	IVX152	IVX1521628	2010-12-04T17:34:00.000+00:...	LIG	RPR	GRE
11	IVX152	IVX1521628	2010-12-03T22:00:00.000+00:...	LIG	RPR	GRE
12	IVX152	IVX1521628	2010-12-03T22:00:00.000+00:...	LIG	RPR	GRE
13	IVX152	IVX1521628	2010-12-02T00:35:00.000+00:...	LIG	RPR	GRE
14	IVX152	IVX1521628	2010-12-02T00:35:00.000+00:...	LIG	RPR	GRE
15	IVX152	IVX1521628	2010-12-01T18:40:00.000+00:...	LIG	RPR	GRE

10.000+ linhas | Dados truncados devido ao limite de linha | 2,90 segundos de runtime

Atualizada há há 1 hora

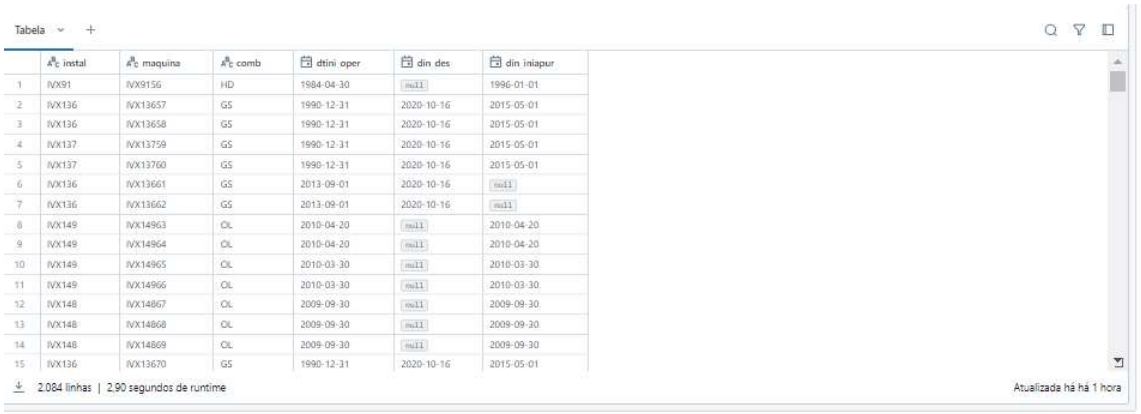


Tabela

	tpins	inst	din_entrada	din_desativacao
1	TER	IVX36	2002-08-21	2024-02-01
2	HID	IVX37	1967-01-01	null
3	TER	IVX38	2008-01-01	2022-08-09
4	HID	IVX39	1956-04-01	null
5	HID	IVX40	2004-08-24	null
6	HID	IVX41	1999-05-09	null
7	HID	IVX42	1999-05-15	null
8	TER	IVX43	2009-11-17	null
9	TER	IVX44	2002-05-01	null
10	TER	IVX45	2002-04-20	2003-06-26
11	TER	IVX46	2002-04-17	2003-06-26
12	TER	IVX47	2002-07-16	null
13	NUC	IVX48	2006-02-25	null
14	HID	IVX49	1969-04-01	null
15	HID	IVX50	2003-01-23	null

339 linhas | 2,90 segundos de runtime

Atualizada há há 1 hora



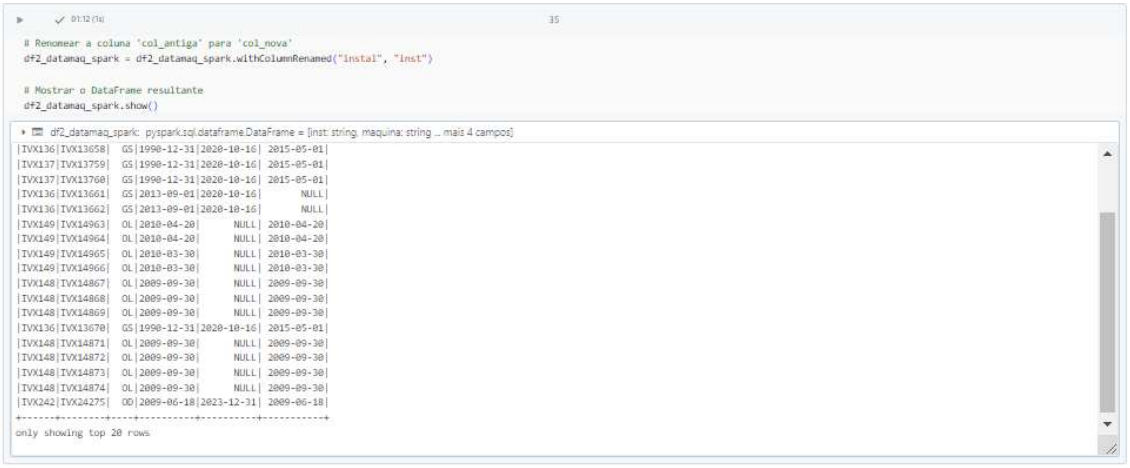
Tabela

	instal	maquina	comb	dtini_oper	din_des	din_iniapur
1	IVX91	IVX9156	HD	1984-04-30	null	1996-01-01
2	IVX136	IVX13657	GS	1990-12-31	2020-10-16	2015-05-01
3	IVX136	IVX13658	GS	1990-12-31	2020-10-16	2015-05-01
4	IVX137	IVX13759	GS	1990-12-31	2020-10-16	2015-05-01
5	IVX137	IVX13760	GS	1990-12-31	2020-10-16	2015-05-01
6	IVX136	IVX13661	GS	2013-09-01	2020-10-16	null
7	IVX136	IVX13662	GS	2013-09-01	2020-10-16	null
8	IVX149	IVX14963	OL	2010-04-20	null	2010-04-20
9	IVX149	IVX14964	OL	2010-04-20	null	2010-04-20
10	IVX149	IVX14965	OL	2010-03-30	null	2010-03-30
11	IVX149	IVX14966	OL	2010-03-30	null	2010-03-30
12	IVX148	IVX14867	OL	2009-09-30	null	2009-09-30
13	IVX148	IVX14868	OL	2009-09-30	null	2009-09-30
14	IVX148	IVX14869	OL	2009-09-30	null	2009-09-30
15	IVX136	IVX13670	GS	1990-12-31	2020-10-16	2015-05-01

2.084 linhas | 2,90 segundos de runtime

Atualizada há há 1 hora

J. Ajuste nome de coluna para manter um padrão



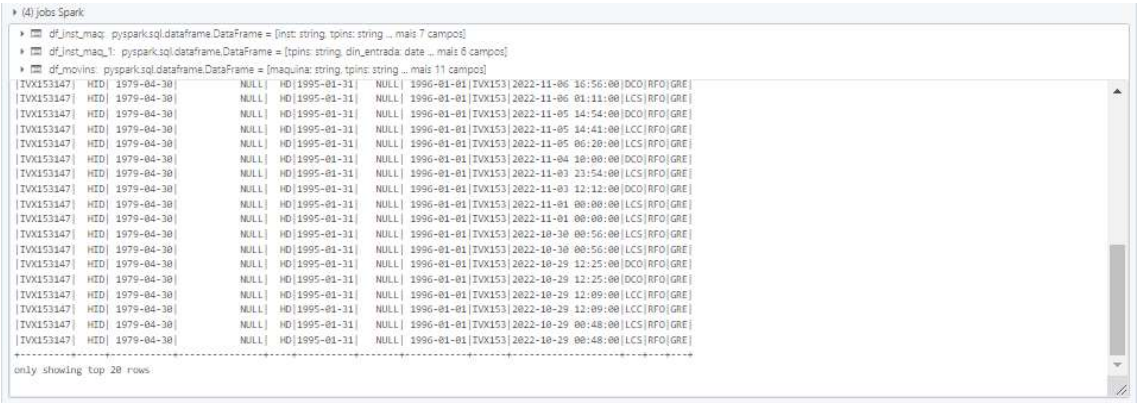
```
# Renomear a coluna 'col_antiga' para 'col_nova'
df2_datamaq_spark = df2_datamaq_spark.withColumnRenamed("instal", "inst")

# Mostrar o DataFrame resultante
df2_datamaq_spark.show()
```

```
df2_datamaq_spark:  pyspark.sql.dataframe.DataFrame = [inst: string, maquina: string ... mais 4 campos]
|IVX136|IVX13658|  GS|1990-12-31|2020-10-16| 2015-05-01|
|IVX137|IVX13759|  GS|1990-12-31|2020-10-16| 2015-05-01|
|IVX137|IVX13760|  GS|1990-12-31|2020-10-16| 2015-05-01|
|IVX136|IVX13661|  GS|2013-09-01|2020-10-16|       NULL|
|IVX136|IVX13662|  GS|2013-09-01|2020-10-16|       NULL|
|IVX149|IVX14963|  OL|2010-04-20|      NULL| 2010-04-20|
|IVX149|IVX14964|  OL|2010-04-20|      NULL| 2010-04-20|
|IVX149|IVX14965|  OL|2010-03-30|      NULL| 2010-03-30|
|IVX149|IVX14966|  OL|2010-03-30|      NULL| 2010-03-30|
|IVX148|IVX14867|  OL|2009-09-30|      NULL| 2009-09-30|
|IVX148|IVX14868|  OL|2009-09-30|      NULL| 2009-09-30|
|IVX148|IVX14869|  OL|2009-09-30|      NULL| 2009-09-30|
|IVX136|IVX13670|  GS|1990-12-31|2020-10-16| 2015-05-01|
|IVX148|IVX14871|  OL|2009-09-30|      NULL| 2009-09-30|
|IVX148|IVX14872|  OL|2009-09-30|      NULL| 2009-09-30|
|IVX148|IVX14873|  OL|2009-09-30|      NULL| 2009-09-30|
|IVX148|IVX14874|  OL|2009-09-30|      NULL| 2009-09-30|
|IVX242|IVX24275|  OD|2009-06-18|2023-12-31| 2009-06-18|
+------+--------+----+----------+----------+-----------+
only showing top 20 rows
```

K. Realizando join para criar uma única visão

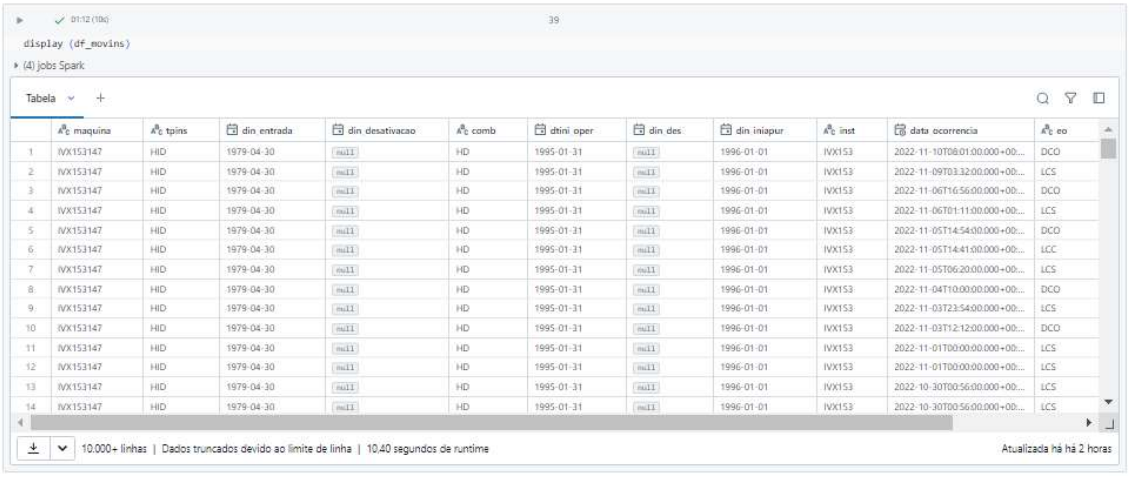
```
from pyspark.sql import SparkSession

# Criar uma sessão Spark
spark = SparkSession.builder.appName("JoinTabelas").getOrCreate()

# Verificar os esquemas
df2_datamaq_spark.printSchema()
df1_datains_spark.printSchema()
df_dataocorre_spark.printSchema()

# Realizar as junções
df_inst_maq = df1_datains_spark.join(df2_datamaq_spark,"inst", "inner")
df_inst_maq = df_inst_maq.drop(df2_datamaq_spark.inst)
df_inst_maq_1 = df_inst_maq.drop(df_inst_maq.inst)
df_movins = df_inst_maq_1.join(df_dataocorre_spark, "maquina", "inner")

# Mostrar o DataFrame final
df_movins.show()
```

```
(4) jobs Spark
```

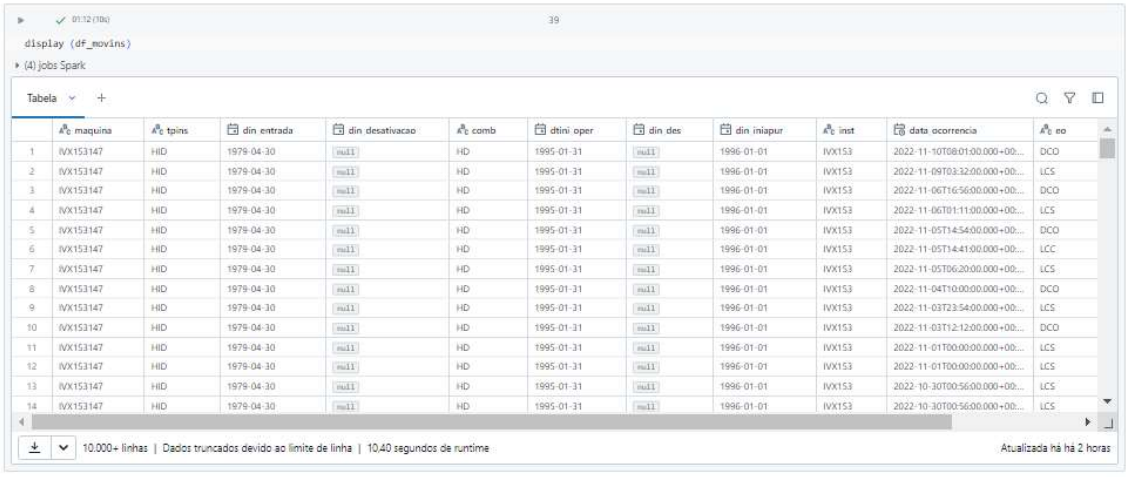


```
(4) jobs Spark
df_inst_maq:  pyspark.sql.dataframe.DataFrame = [inst: string, tpins: string ... mais 7 campos]
df_inst_maq_1:  pyspark.sql.dataframe.DataFrame = [tpins: string, dtin_entrada: date ... mais 6 campos]
df_movins:  pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 11 campos]
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-11-06 16:56:00|DCO|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-11-06 01:11:00|LCS|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-11-05 14:54:00|DCO|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-11-05 14:41:00|LCC|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-11-05 06:20:00|LCS|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-11-04 10:00:00|DCO|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-11-03 23:54:00|LCS|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-11-03 12:12:00|DCO|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-11-01 00:00:00|LCS|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-11-01 00:00:00|LCS|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-10-30 00:56:00|LCS|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-10-30 00:56:00|LCS|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-10-29 12:25:00|DCO|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-10-29 12:25:00|DCO|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-10-29 12:09:00|LCC|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-10-29 12:09:00|LCC|RFO|GRE|
|IVX153147|  HID| 1979-04-30|           NULL|  HD|1995-01-31|    NULL| 1996-01-01|IVX153|2022-10-29 00:48:00|LCS|RFO|GRE|
+---------+-----+------------+---------------+----+----------+--------+-----------+------+-------------------+---+---+---+
only showing top 20 rows
```

L. Visualizando resultado do dataframe

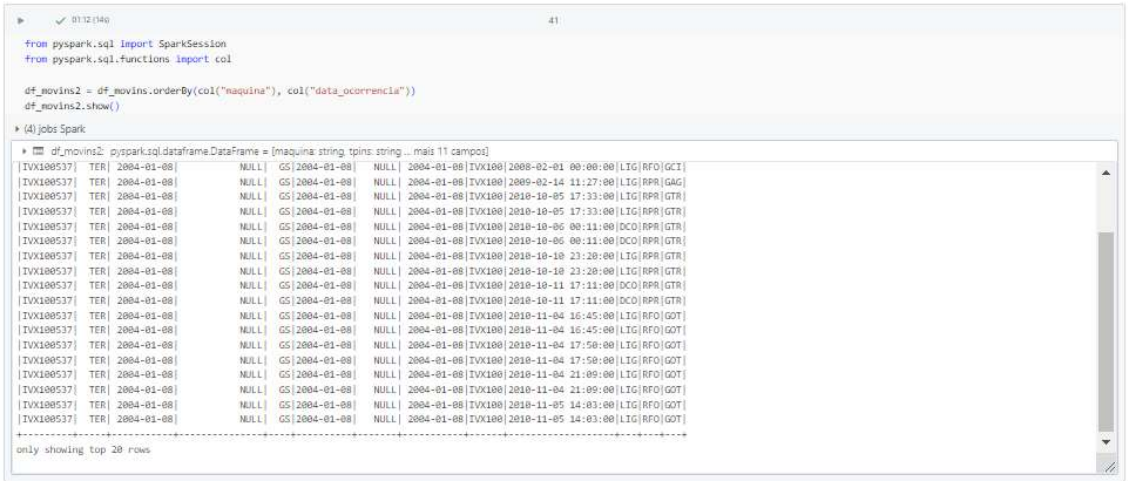


M. Visualizando resultado do dataframe



N. Criando ordenação por máquina e data de ocorrência

O. Criando ordenação por máquina e data de ocorrência



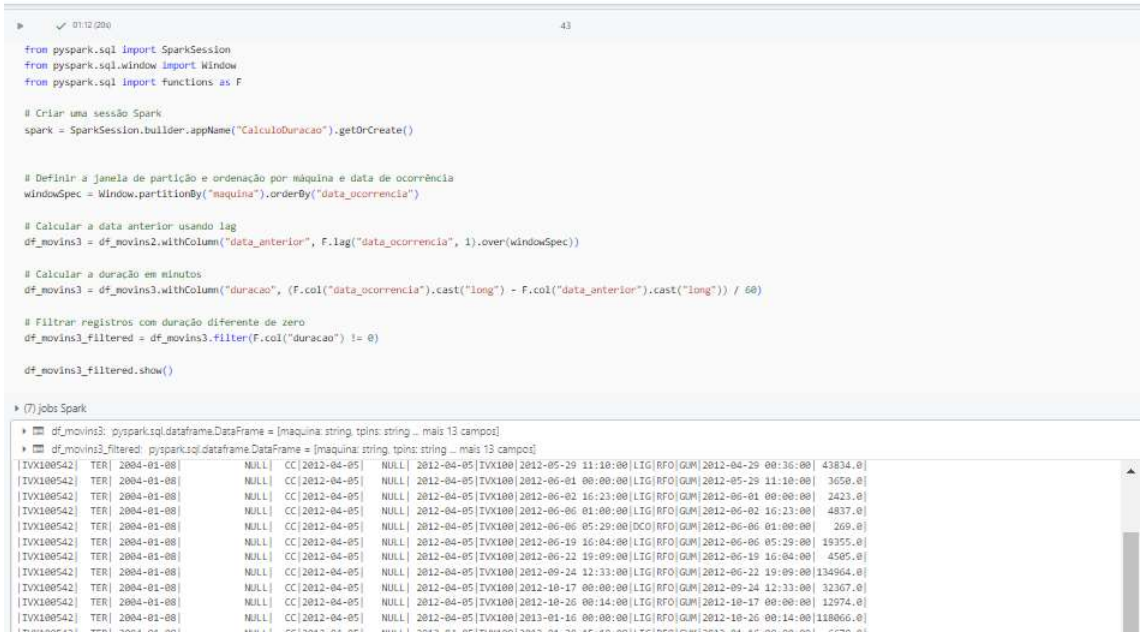
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

df_movins2 = df_movins.orderBy(col("maquina"), col("data_ocorrencia"))
df_movins2.show()
```

(4) jobs Spark

df_movins2: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 11 campos]

```
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2008-02-01 00:00:00|LIG|RFO|GCI|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2009-02-14 11:27:00|LIG|RPR|GAG|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-10-05 17:33:00|LIG|RPR|GTR|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-10-05 17:33:00|LIG|RPR|GTR|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-10-06 00:11:00|DCO|RPR|GTR|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-10-06 00:11:00|DCO|RPR|GTR|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-10-10 23:20:00|LIG|RPR|GTR|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-10-10 23:20:00|LIG|RPR|GTR|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-10-11 17:11:00|DCO|RPR|GTR|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-10-11 17:11:00|DCO|RPR|GTR|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-11-04 16:45:00|LIG|RFO|GOT|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-11-04 16:45:00|LIG|RFO|GOT|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-11-04 17:50:00|LIG|RFO|GOT|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-11-04 17:50:00|LIG|RFO|GOT|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-11-04 21:09:00|LIG|RFO|GOT|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-11-04 21:09:00|LIG|RFO|GOT|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-11-05 14:03:00|LIG|RFO|GOT|
|IVX100537|  TER| 2004-01-08|           NULL|  GS|2004-01-08|   NULL| 2004-01-08|IVX100|2010-11-05 14:03:00|LIG|RFO|GOT|
+---------+-----+-----------+---------------+----+----------+-------+-----------+------+-------------------+---+---+---+
only showing top 20 rows
```

P. Cálculo de Durações de todas as movimentações



```
from pyspark.sql import SparkSession
from pyspark.sql.window import Window
from pyspark.sql import functions as F

# Criar uma sessão Spark
spark = SparkSession.builder.appName("CalculoDuracao").getOrCreate()


# Definir a janela de partição e ordenação por máquina e data de ocorrência
windowSpec = Window.partitionBy("maquina").orderBy("data_ocorrencia")

# Calcular a data anterior usando lag
df_movins3 = df_movins2.withColumn("data_anterior", F.lag("data_ocorrencia", 1).over(windowSpec))

# Calcular a duração em minutos
df_movins3 = df_movins3.withColumn("duracao", (F.col("data_ocorrencia").cast("long") - F.col("data_anterior").cast("long")) / 60)

# Filtrar registros com duração diferente de zero
df_movins3_filtered = df_movins3.filter(F.col("duracao") != 0)

df_movins3_filtered.show()
```

(7) jobs Spark

df_movins3: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 13 campos]

df_movins3_filtered: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 13 campos]

```
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2012-05-29 11:10:00|LIG|RFO|GUM|2012-04-29 00:36:00| 43834.0|
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2012-06-01 00:00:00|LIG|RFO|GUM|2012-05-29 11:10:00|  3650.0|
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2012-06-02 16:23:00|LIG|RFO|GUM|2012-06-01 00:00:00|  2423.0|
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2012-06-06 01:00:00|LIG|RFO|GUM|2012-06-02 16:23:00|  4837.0|
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2012-06-06 05:29:00|DCO|RFO|GUM|2012-06-06 01:00:00|   269.0|
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2012-06-19 16:04:00|LIG|RFO|GUM|2012-06-06 05:29:00| 19355.0|
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2012-06-22 19:09:00|LIG|RFO|GUM|2012-06-19 16:04:00|  4505.0|
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2012-09-24 12:33:00|LIG|RFO|GUM|2012-06-22 19:09:00|134964.0|
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2012-10-17 00:00:00|LIG|RFO|GUM|2012-09-24 12:33:00| 32367.0|
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2012-10-26 00:14:00|LIG|RFO|GUM|2012-10-17 00:00:00| 12974.0|
|IVX100542|  TER| 2004-01-08|           NULL|  CC|2012-04-05|   NULL| 2012-04-05|IVX100|2013-01-16 00:00:00|LIG|RFO|GUM|2012-10-26 00:14:00|118066.0|
```

Q. Contagem do número de registros do dataframe total e com filtro "RPR"

```
01:12 (13s)                                                    45

# Supondo que seu DataFrame já exista como df_movins2
num_registros = df_movins3.count()

print(f"O DataFrame possui {num_registros} registros.")


# Supondo que seu DataFrame já exista como df_movins2
num_registros = df_movins3.filter(df_movins3["co"] == "RPR").count()

print(f"O DataFrame possui {num_registros} registros onde CO = RPR.")

(9) jobs Spark

O DataFrame possui 1048575 registros.
O DataFrame possui 37124 registros onde CO = RPR.
```

R. Contagem do número de registros do dataframe total e com filtro "RPR"

```
01:12 (18s)                                                    46

display(df_movins3)

(7) jobs Spark
```

Tabela

	maquina	tpins	din_entrada	din_desativacao	comb	dtini_oper	din_des	din_iniapur	inst	data_ocorrencia
13	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-06-06T01:00:00.000+00:...
14	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-06-06T01:00:00.000+00:...
15	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-06-06T05:29:00.000+00:...
16	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-06-06T05:29:00.000+00:...
17	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-06-19T16:04:00.000+00:...
18	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-06-19T16:04:00.000+00:...
19	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-06-22T19:09:00.000+00:...
20	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-06-22T19:09:00.000+00:...
21	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-09-24T12:33:00.000+00:...
22	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-09-24T12:33:00.000+00:...
23	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-10-17T00:00:00.000+00:...
24	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-10-17T00:00:00.000+00:...
25	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-10-26T00:14:00.000+00:...
26	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2012-10-26T00:14:00.000+00:...

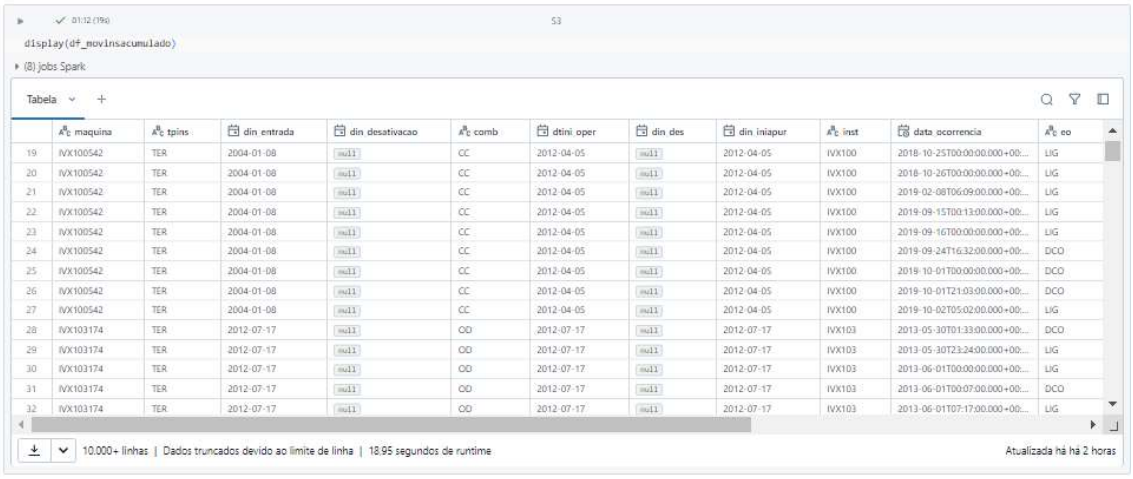
10.000+ linhas | Dados truncados devido ao limite de linha | 17,76 segundos de runtime — Atualizada há há 2 horas

S. Dataframe com a duração acumulada de todos tipos de CO

```
01:12 (18s)                                                    48

from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.window import Window

# Criar uma sessão Spark
spark = SparkSession.builder.appName("AcumularDuracaoPorMaquina").getOrCreate()

# Definir janela de partição por máquina e ordenação por data_ocorrencia
windowSpec = Window.partitionBy("maquina").orderBy("data_ocorrencia").rowsBetween(Window.unboundedPreceding, 0)

df_movins4_acum = df_movins3.withColumn("soma_duracao_for",
                                        F.coalesce(F.when(F.col("co") == "RFO", F.sum("duracao").over(windowSpec)), F.lit(0)))
df_movins4_acum = df_movins4_acum.withColumn("soma_duracao_prog",
                                        F.coalesce(F.when(F.col("co") == "RPR", F.sum("duracao").over(windowSpec)), F.lit(0)))
df_movins4_acum = df_movins4_acum.withColumn("soma_duracao_null",
                                        F.coalesce(F.when(F.col("co").isNull(), F.sum("duracao").over(windowSpec)), F.lit(0)))
df_movins4_acum = df_movins4_acum.withColumn("soma_duracao_nor",
                                        F.coalesce(F.when(F.col("co").isin("NOT", "TST", "NOR"), F.sum("duracao").over(windowSpec)), F.lit(0)))


# Filtrar registros com duração diferente de zero
df_movins4_acum = df_movins4_acum.filter(F.col("duracao") != 0)

# Mostrar o DataFrame resultante
df_movins4_acum.show(truncate=False)

(7) jobs Spark
```

T. Contagem de registro depois dos tratamentos

U. Criação de dataframe apenas com o filtro RPR somando todas as durações por máquina.



```
display(df_movinsacumulado)
```

(8) jobs Spark

Tabela

	maquina	tpins	din_entrada	din_desativacao	comb	dtini_oper	din_des	din_iniapur	inst	data_ocorrencia	eo
19	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2018-10-25T00:00:00.000+00:...	LIG
20	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2018-10-26T00:00:00.000+00:...	LIG
21	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2019-02-08T06:09:00.000+00:...	LIG
22	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2019-09-15T00:13:00.000+00:...	LIG
23	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2019-09-16T00:00:00.000+00:...	LIG
24	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2019-09-24T16:32:00.000+00:...	DCO
25	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2019-10-01T00:00:00.000+00:...	DCO
26	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2019-10-01T21:03:00.000+00:...	DCO
27	IVX100542	TER	2004-01-08	null	CC	2012-04-05	null	2012-04-05	IVX100	2019-10-02T05:02:00.000+00:...	LIG
28	IVX103174	TER	2012-07-17	null	OD	2012-07-17	null	2012-07-17	IVX103	2013-05-30T01:33:00.000+00:...	DCO
29	IVX103174	TER	2012-07-17	null	OD	2012-07-17	null	2012-07-17	IVX103	2013-05-30T23:24:00.000+00:...	LIG
30	IVX103174	TER	2012-07-17	null	OD	2012-07-17	null	2012-07-17	IVX103	2013-06-01T00:00:00.000+00:...	LIG
31	IVX103174	TER	2012-07-17	null	OD	2012-07-17	null	2012-07-17	IVX103	2013-06-01T00:07:00.000+00:...	DCO
32	IVX103174	TER	2012-07-17	null	OD	2012-07-17	null	2012-07-17	IVX103	2013-06-01T07:17:00.000+00:...	LIG

10.000+ linhas | Dados truncados devido ao limite de linha | 18.95 segundos de runtime

Atualizada há há 2 horas

V. Contagem dos registros que ficaram no novo dataframe

```
num_registros = df_movinsacumulado.count()

print(f"O DataFrame possui {num_registros} registros.")
```

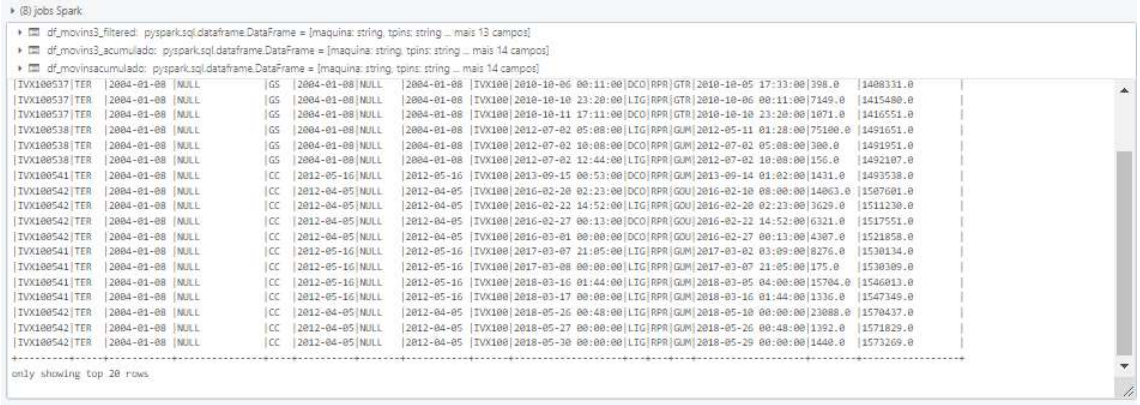
(8) jobs Spark

```
O DataFrame possui 20414 registros.
```

W. Vou realizar o tratamento para encontrar a duração acumulada por instalação

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.window import Window

# Criar uma sessão Spark
spark = SparkSession.builder.appName("AcumularDuracaoPorInstalacao").getOrCreate()

# Supondo que df_movins3_filtered já exista e contenha a coluna 'duracao' calculada
df_movins3_filtered = df_movins3.filter(F.col("co") == "RPR")  # Substitua por seu DataFrame filtrado com 'co' == 'RPR'

# Definir janela de partição por instalação e ordenação por data_ocorrencia
windowSpec = Window.partitionBy("inst").orderBy("data_ocorrencia").rowsBetween(Window.unboundedPreceding, 0)

# Adicionar coluna de soma acumulada de duracao por instalação
df_movins3_acumulado = df_movins3_filtered.withColumn("soma_duracao_inst", F.sum("duracao").over(windowSpec))

# Filtrar registros com duração diferente de zero
df_movinsacumulado = df_movins3_acumulado.filter(F.col("duracao") != 0)

# Mostrar o DataFrame resultante
df_movinsacumulado.show(truncate=False)
```

(8) jobs Spark



(8) jobs Spark

df_movins3_filtered: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 13 campos]

df_movins3_acumulado: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 14 campos]

df_movinsacumulado: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 14 campos]

```
|IVX100537|TER  |2004-01-08 |NULL          |GS  |2004-01-08|NULL    |2004-01-08 |IVX100|2010-10-06 00:11:00|DCO|RPR|GTR|2010-10-05 17:13:00|398.0   |1408331.0       |
|IVX100537|TER  |2004-01-08 |NULL          |GS  |2004-01-08|NULL    |2004-01-08 |IVX100|2010-10-10 23:20:00|LIG|RPR|GTR|2010-10-06 00:11:00|7149.0  |1415480.0       |
|IVX100537|TER  |2004-01-08 |NULL          |GS  |2004-01-08|NULL    |2004-01-08 |IVX100|2010-10-11 17:11:00|DCO|RPR|GTR|2010-10-10 23:20:00|1071.0  |1416551.0       |
|IVX100538|TER  |2004-01-08 |NULL          |GS  |2004-01-08|NULL    |2004-01-08 |IVX100|2012-07-02 05:08:00|LIG|RPR|GUM|2012-05-11 01:28:00|75100.0 |1491651.0       |
|IVX100538|TER  |2004-01-08 |NULL          |GS  |2004-01-08|NULL    |2004-01-08 |IVX100|2012-07-02 10:08:00|DCO|RPR|GUM|2012-07-02 05:08:00|300.0   |1491951.0       |
|IVX100538|TER  |2004-01-08 |NULL          |GS  |2004-01-08|NULL    |2004-01-08 |IVX100|2012-07-02 12:44:00|LIG|RPR|GUM|2012-07-02 10:08:00|156.0   |1492107.0       |
|IVX100541|TER  |2004-01-08 |NULL          |CC  |2012-05-16|NULL    |2012-05-16 |IVX100|2013-09-15 00:53:00|DCO|RPR|GUM|2013-09-14 01:02:00|1431.0  |1493538.0       |
|IVX100542|TER  |2004-01-08 |NULL          |CC  |2012-04-05|NULL    |2012-04-05 |IVX100|2016-02-20 02:23:00|DCO|RPR|GOU|2016-02-10 08:00:00|14063.0 |1507601.0       |
|IVX100542|TER  |2004-01-08 |NULL          |CC  |2012-04-05|NULL    |2012-04-05 |IVX100|2016-02-22 14:52:00|LIG|RPR|GOU|2016-02-20 02:23:00|3629.0  |1511230.0       |
|IVX100542|TER  |2004-01-08 |NULL          |CC  |2012-04-05|NULL    |2012-04-05 |IVX100|2016-02-27 00:13:00|DCO|RPR|GOU|2016-02-22 14:52:00|6321.0  |1517551.0       |
|IVX100542|TER  |2004-01-08 |NULL          |CC  |2012-04-05|NULL    |2012-04-05 |IVX100|2016-03-01 00:00:00|DCO|RPR|GOU|2016-02-27 00:13:00|4307.0  |1521858.0       |
|IVX100541|TER  |2004-01-08 |NULL          |CC  |2012-05-16|NULL    |2012-05-16 |IVX100|2017-03-07 21:05:00|LIG|RPR|GUM|2017-03-02 03:09:00|8276.0  |1530134.0       |
|IVX100541|TER  |2004-01-08 |NULL          |CC  |2012-05-16|NULL    |2012-05-16 |IVX100|2017-03-08 00:00:00|LIG|RPR|GUM|2017-03-07 21:05:00|175.0   |1530309.0       |
|IVX100541|TER  |2004-01-08 |NULL          |CC  |2012-05-16|NULL    |2012-05-16 |IVX100|2018-03-16 01:44:00|LIG|RPR|GUM|2018-03-05 04:00:00|15704.0 |1546013.0       |
|IVX100541|TER  |2004-01-08 |NULL          |CC  |2012-05-16|NULL    |2012-05-16 |IVX100|2018-03-17 00:00:00|LIG|RPR|GUM|2018-03-16 01:44:00|1336.0  |1547349.0       |
|IVX100542|TER  |2004-01-08 |NULL          |CC  |2012-04-05|NULL    |2012-04-05 |IVX100|2018-05-26 00:48:00|LIG|RPR|GUM|2018-05-10 00:00:00|23088.0 |1570437.0       |
|IVX100542|TER  |2004-01-08 |NULL          |CC  |2012-04-05|NULL    |2012-04-05 |IVX100|2018-05-27 00:00:00|LIG|RPR|GUM|2018-05-26 00:48:00|1392.0  |1571829.0       |
|IVX100542|TER  |2004-01-08 |NULL          |CC  |2012-04-05|NULL    |2012-04-05 |IVX100|2018-05-30 00:00:00|LIG|RPR|GUM|2018-05-29 00:00:00|1440.0  |1573269.0       |
+---------+-----+-----------+--------------+----+----------+--------+-----------+------+-------------------+---+---+---+-------------------+--------+----------------+
only showing top 20 rows
```

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.window import Window

# Criar uma sessão Spark
spark = SparkSession.builder.appName("MaxDuracaoPorInstalacao").getOrCreate()

# Supondo que df_movins3_acumulado já exista
# Filtrar registros com duração diferente de zero
df_filtrado = df_movins3_acumulado.filter(F.col("duracao") != 0)

# Definir janela de partição por instalação e ordenação por data_ocorrencia
windowSpec = Window.partitionBy("inst").orderBy("data_ocorrencia").rowsBetween(Window.unboundedPreceding, 0)

# Adicionar coluna de soma acumulada de duracao por instalação
df_movins3_acumulado = df_filtrado.withColumn("soma_duracao_inst", F.sum("duracao").over(windowSpec))

# Agrupar por instalação (inst) e calcular a duração máxima por instalação
df_max_duracao_inst = df_movins3_acumulado.groupBy("inst", "tpins", "din_entrada").agg(F.max("soma_duracao_inst").alias("max_duracao_inst"))

# Mostrar o DataFrame resultante
df_max_duracao_inst.show(truncate=False)
```

```
(8) jobs Spark
  df_filtrado: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 14 campos]
  df_movins3_acumulado: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 14 campos]
  df_max_duracao_inst: pyspark.sql.dataframe.DataFrame = [inst: string, tpins: string ... mais 2 campos]
|IVX104|TER  |2013-02-05 |106958.0        |
|IVX107|HID  |1997-11-02 |4027389.0       |
|IVX109|TER  |2002-07-08 |2951208.0       |
|IVX111|HID  |1999-01-01 |1.0534397E7     |
|IVX112|TER  |1978-03-01 |2150945.0       |
|IVX114|TER  |1977-07-07 |993000.0        |
|IVX116|HID  |1969-04-30 |6345110.0       |
|IVX117|HID  |1956-04-01 |4784171.0       |
|IVX118|HID  |2010-05-25 |104285.0        |
|IVX130|TER  |1990-12-31 |302967.0        |
|IVX146|HID  |1977-02-28 |3019.0          |
|IVX148|TER  |2009-09-30 |1156179.0       |
|IVX149|TER  |2010-03-30 |118048.0        |
|IVX150|HID  |2004-12-16 |1406761.0       |
|IVX151|TER  |2010-05-07 |2950333.0       |
|IVX152|HID  |2003-02-05 |3522985.0       |
|IVX153|HID  |1979-04-30 |476750.0        |
|IVX154|HID  |1955-01-31 |5966720.0       |
+------+-----+-----------+----------------+
only showing top 20 rows
```

7. Análise de Dados

a. Qualidade de dados

<u>Existem problemas no conjunto de dados? Caso haja, como esses problemas podem ser resolvidos para que não afetem as respostas das perguntas que quer solucionar?</u>

Sim, a base original é muito grande, tive que fazer um recorte na massa para trabalhar.

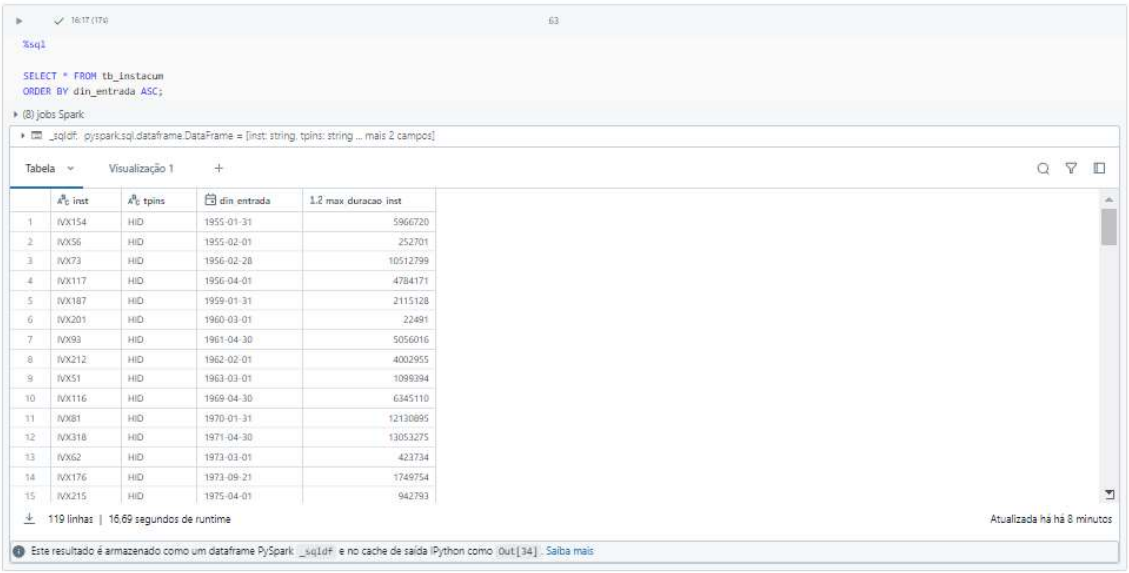
O ideal é que esse recorte seja trabalhado para apresentar proporcionalmente o mesmo número de movimentações para todas as instalações ou então que a janela de tempo tenha a mesma duração para todas as instalações. Por exemplo, não é possível comparar a duração das movimentações de uma determinada característica de uma instalação que tem mais de 20 anos de operação com uma instalação com movimentações parecidas, mas que tem apenas 5 anos de operação.

Além disso, acredito que houve muita perda de registros por utilizar arquivos .csv na carga dos dados. E isso gera uma dúvida sobre os números de durações que estão sendo calculados, uma solução seria realizar a cópia da base de dados original para outra tratando os dados sensíveis e permitindo assim a utilização dessa base no trabalho. Garantindo o mesmo número de registros que a base original.

b. Solução do problema

Informações iniciais para ajudar na complementação da análise

Qual a distribuição de instalações por tipo?



```
%sql

SELECT * FROM tb_instacum
ORDER BY din_entrada ASC;
```

(8) jobs Spark

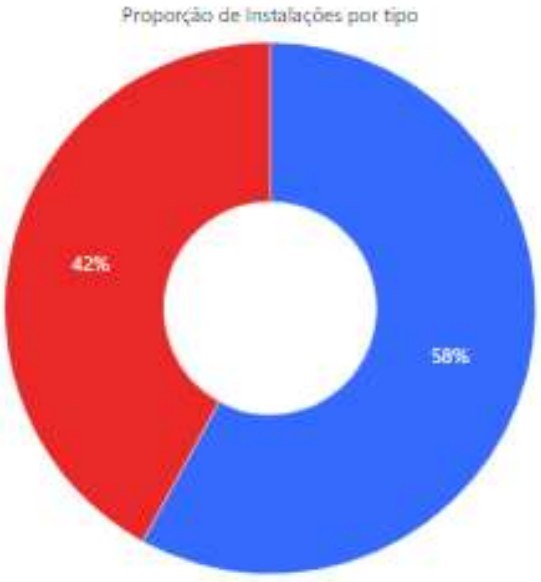
_sqldf: pyspark.sql.dataframe.DataFrame = [inst: string, tpins: string ... mais 2 campos]

Tabela | Visualização 1 | +

	inst	tpins	din_entrada	1.2 max duracao inst
1	IVX154	HID	1955-01-31	5966720
2	IVX56	HID	1955-02-01	252701
3	IVX73	HID	1956-02-28	10512799
4	IVX117	HID	1956-04-01	4784171
5	IVX187	HID	1959-01-31	2115128
6	IVX201	HID	1960-03-01	22491
7	IVX93	HID	1961-04-30	5056016
8	IVX212	HID	1962-02-01	4002955
9	IVX51	HID	1963-03-01	1099394
10	IVX116	HID	1969-04-30	6345110
11	IVX81	HID	1970-01-31	12130895
12	IVX318	HID	1971-04-30	13053275
13	IVX62	HID	1973-03-01	423734
14	IVX176	HID	1973-09-21	1749754
15	IVX215	HID	1975-04-01	942793

119 linhas | 16.69 segundos de runtime — Atualizada há há 8 minutos

Este resultado é armazenado como um dataframe PySpark _sqldf e no cache de saída IPython como Out[34]. Saiba mais



Ponto de Atenção 1:

Ao todo 58% das nossas instalações são do tipo HD e as demais do tipo TER

Qual a distribuição de horas acumuladas das instalações considerando a sua data de entrada?

```
%sql

SELECT * FROM tb_instacum
ORDER BY din_entrada ASC
```

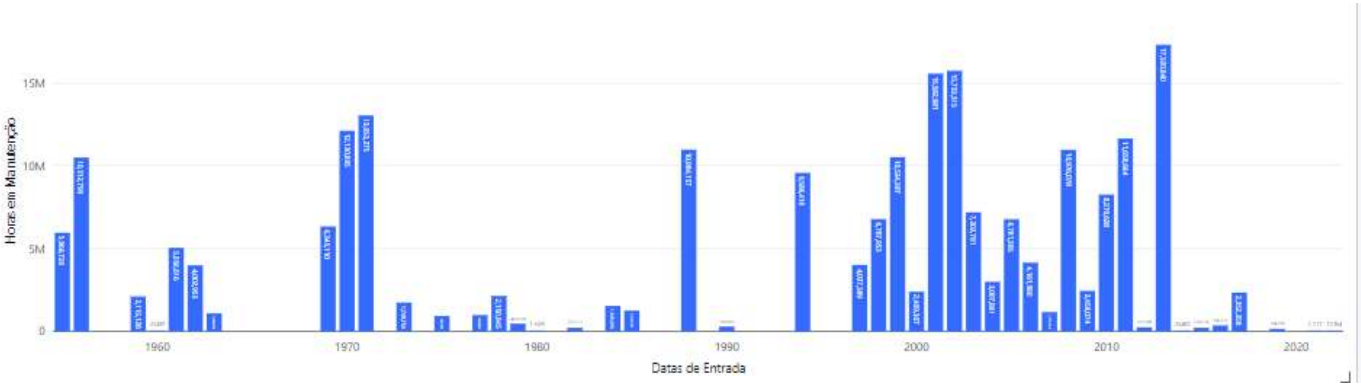
(8) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [inst: string, tpins: string ... mais 2 campos]

Tabela | Visualização 1 | +

	inst	tpins	din_entrada	1.2 max duracao_inst
1	IVX154	HID	1955-01-31	5966720
2	IVX56	HID	1955-02-01	252701
3	IVX73	HID	1956-02-28	10512799
4	IVX117	HID	1956-04-01	4784171
5	IVX187	HID	1958-01-31	2115128
6	IVX201	HID	1960-03-01	22491
7	IVX93	HID	1961-04-30	5056016
8	IVX212	HID	1962-02-01	4002955
9	IVX51	HID	1963-03-01	1089394
10	IVX116	HID	1969-04-30	6345110
11	IVX81	HID	1970-01-31	12130895
12	IVX318	HID	1971-04-30	13053275
13	IVX62	HID	1973-03-01	423734
14	IVX176	HID	1973-09-21	1749754
15	IVX215	HID	1975-04-01	942793

119 linhas | 18.50 segundos de runtime — Atualizada há há 36 minutos



Ponto de Atenção 2:

Nesse select podemos interpretar que a data de operação mais antiga não necessariamente determina que uma instalação terá mais horas em operação do que uma outra instalação que entrou mais recentemente.

Temos instalações de 2010 que tem mais horas de apuração que as instalações mais antigas.

Esse dado precisa ser confirmado com a utilização da base toda.

Qual a instalação teve mais horas dedicadas à manutenção programada

```
%sql

SELECT * FROM tb_instacum
ORDER BY max_duracao_inst DESC
```

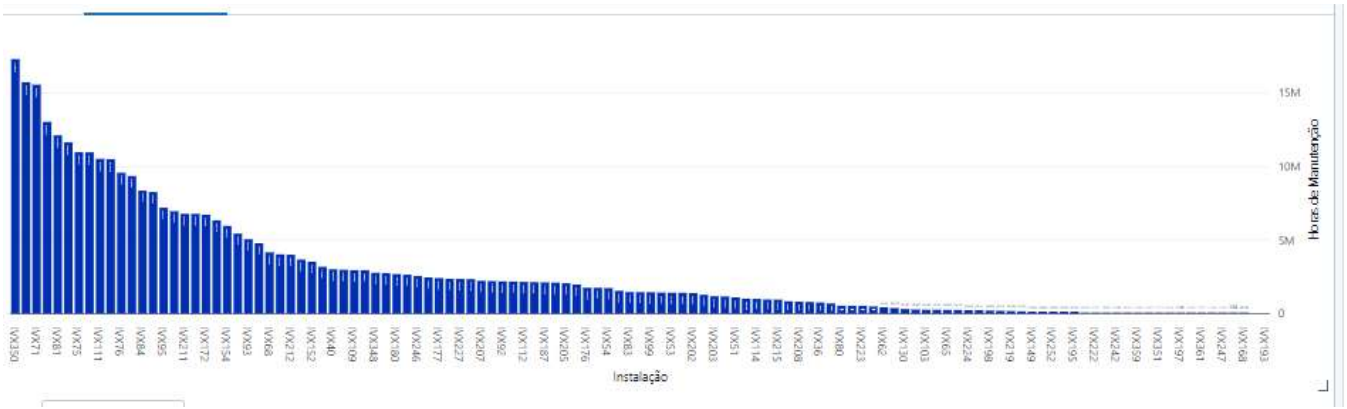
(8) Jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [inst: string, tpins: string ... mais 2 campos]

Tabela | Visualização 1 | +

	inst	tpins	din_entrada	max_duracao_inst
1	IVX350	TER	2013-01-24	17320840
2	IVX67	HID	2002-05-22	15753315
3	IVX71	HID	2001-04-01	15582981
4	IVX318	HID	1971-04-30	13053275
5	IVX81	HID	1970-01-31	12130895
6	IVX315	HID	2011-04-29	11658664
7	IVX75	HID	1988-02-28	10984157
8	IVX234	TER	2008-01-01	10976078
9	IVX111	HID	1999-01-01	10534397
10	IVX73	HID	1956-02-28	10512799
11	IVX76	HID	1994-04-30	9586410
12	IVX218	TER	1999-04-01	9362805
13	IVX84	TER	2008-01-01	8363773
14	IVX347	TER	2010-12-30	8270628
15	IVX95	TER	2003-12-27	7202791

119 linhas | 16,66 segundos de runtime — Atualizada há há 16 minutos



Ponto de Atenção 3:

É possível verificar que as instalações com mais horas acumuladas em operação não são as mais antigas. Das cinco primeiras temos uma distribuição de três que entraram após 2000 e duas são da década de 70.

Então é mais um ponto que indica que a idade da instalação não é determinante para o número de horas de operação ou manutenções.

Qual o tipo de instalação que consome mais horas programadas de manutenção?

```
%sql

SELECT tpins, SUM(max_duracao_inst) AS soma_max_duracao_tpins
FROM tb_instacum
GROUP BY tpins
```

(9) jobs Spark

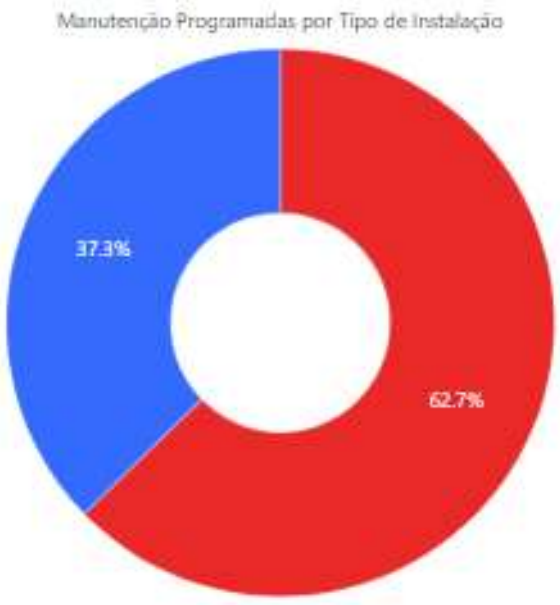
_sqldf: pyspark.sql.dataframe.DataFrame = [tpins: string, soma_max_duracao_tpins: double]

Tabela | Visualização 1 +

	tpins	soma_max_duracao_tpins
1	TER	127782635
2	HID	214910728

2 linhas | 18.01 segundos de runtime — Atualizada há há 19 minutos

Este resultado é armazenado como um dataframe PySpark `_sqldf` e no cache de saída IPython como `Out[37]`. Saiba mais



Ponto de Atenção 4:

As instalações que mais têm horas de manutenção programada são as instalações do tipo HD. Isso também pode ser explicado por serem as de maior número e as mais antigas. Importante salientar que nas análises anteriores identificamos que a instalação que tem maior duração de horas acumuladas é uma do tipo TER e sua entrada foi em 2013.

Quais são as características das 20 primeiras instalações com mais duração de manutenção programada acumulada?

```
%sql
SELECT inst, tpins, din_entrada, max_duracao_inst
FROM tb_instacum
ORDER BY max_duracao_inst DESC
LIMIT 20
```

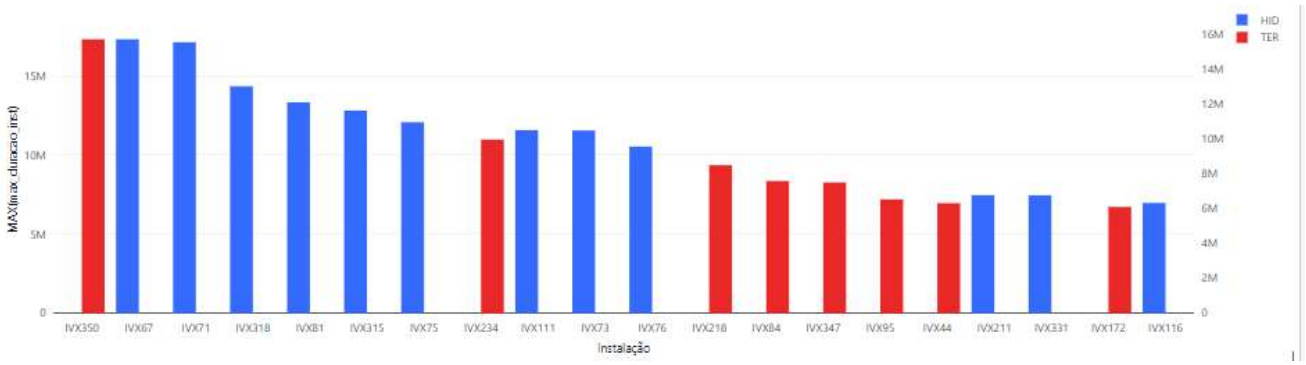
(8) jobs Spark

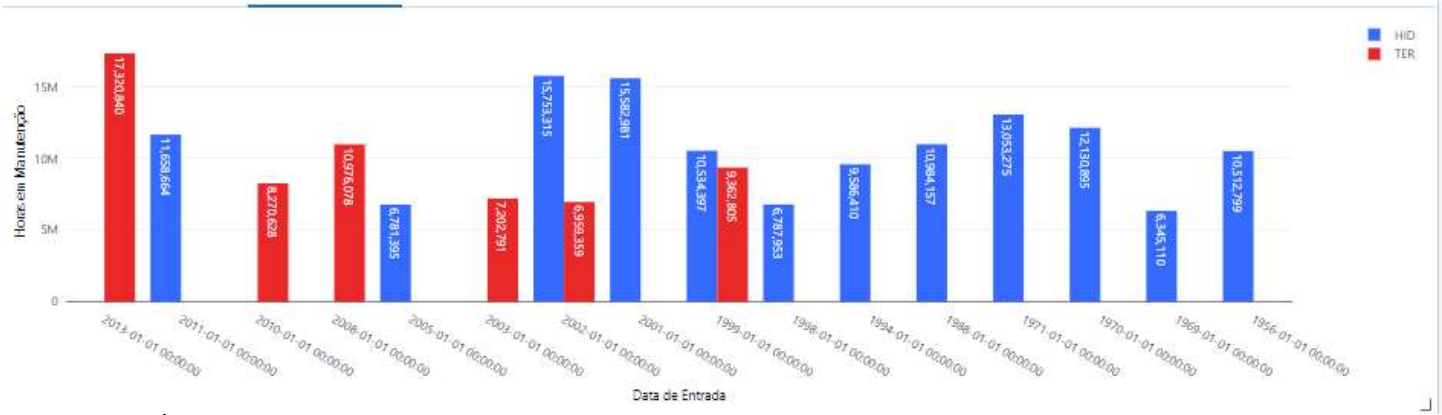
_sqldf: pyspark.sql.dataframe.DataFrame = [inst: string, tpins: string ... mais 2 campos]

Tabela | Visualização 1 | Visualização 2 | +

	inst	tpins	din_entrada	max_duracao_inst
1	IVX350	TER	2013-01-24	17320840
2	IVX67	HID	2002-05-22	15753315
3	IVX71	HID	2001-04-01	15582981
4	IVX318	HID	1971-04-30	13053275
5	IVX81	HID	1970-01-31	12130895
6	IVX315	HID	2011-04-29	11658664
7	IVX75	HID	1988-02-28	10984157
8	IVX234	TER	2008-01-01	10976078
9	IVX111	HID	1999-01-01	10534397
10	IVX73	HID	1956-02-28	10512799
11	IVX76	HID	1984-04-30	9586410
12	IVX218	TER	1999-04-01	9362805
13	IVX84	TER	2008-01-01	8363773
14	IVX347	TER	2010-12-30	8270628
15	IVX95	TER	2003-12-27	7202791

20 linhas | 17,70 segundos de runtime — Atualizada há 40 minutos





É bem distribuída as diferentes instalações que tem maior duração de manutenção programada. Entre eles 08 são do tipo TER, sendo desse tipo a instalação com maior número de horas acumulado em operação. E também é a mais recente da lista.

As demais instalações do tipo HD, 12 no total também são as que possuem maior número de horas acumulado, sendo algumas vezes 2 vezes mais número de horas se comparado o segundo com o terceiro lugar da lista acima.

Agora queremos saber como é a proporção dessas horas desde que iniciou o registro das movimentações.

Quanto cada máquina participa na soma do tempo total em operação da instalação?

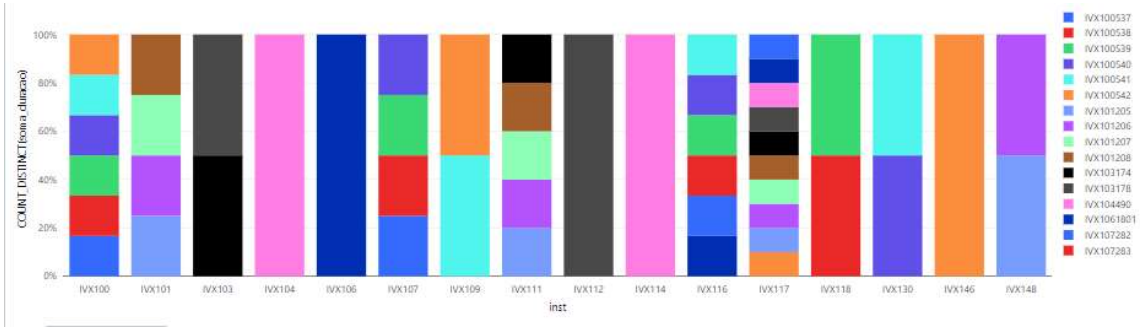
```
%sql

SELECT inst,
       tpins,
       comb,
       din_entrada,
       dtini_oper,
       din_des,
       din_iniapur,
       maquina,
       data_ocorrencia,
       duracao,
       SUM(duracao) OVER (PARTITION BY maquina ORDER BY din_entrada) as soma_duracao
FROM tb_movcoacum
ORDER BY din_entrada ASC;
```

(7) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [inst: string, tpins: string ... mais 9 campos]

Tabela | Visualização 1

	inst	tpins	comb	din entrada	dtini oper	din des	din iniapur	maquina	data ocorrencia	1.2 duracao	1.2 soma duracao
13	IVX154	HID	HD	1955-01-31	1955-01-31	null	1996-01-01	IVX154275	2012-02-15T10:54:00.000+00:...	2494	65639
14	IVX154	HID	HD	1955-01-31	1955-01-31	null	1996-01-01	IVX154276	2012-09-05T14:44:00.000+00:...	239	72399
15	IVX154	HID	HD	1955-01-31	1955-03-31	null	1996-01-01	IVX154274	2010-05-08T14:36:00.000+00:...	1305	60675
16	IVX154	HID	HD	1955-01-31	1955-01-31	null	1996-01-01	IVX154276	2012-09-05T15:24:00.000+00:...	40	72399
17	IVX154	HID	HD	1955-01-31	1955-01-31	null	1996-01-01	IVX154275	2020-11-17T23:24:00.000+00:...	4605870	65639
18	IVX154	HID	HD	1955-01-31	1955-01-31	null	1996-01-01	IVX154276	2012-10-03T09:28:00.000+00:...	39964	72399
19	IVX154	HID	HD	1955-01-31	1955-03-31	null	1996-01-01	IVX154274	2012-11-26T17:24:00.000+00:...	1343688	60675
20	IVX154	HID	HD	1955-01-31	1955-01-31	null	1996-01-01	IVX154276	2020-11-17T21:03:00.000+00:...	4273175	72399
21	IVX154	HID	HD	1955-01-31	1955-01-31	null	1996-01-01	IVX154275	2020-11-18T23:08:00.000+00:...	1424	65639
22	IVX154	HID	HD	1955-01-31	1955-01-31	null	1996-01-01	IVX154276	2020-11-18T23:08:00.000+00:...	1565	72399
23	IVX154	HID	HD	1955-01-31	1955-03-31	null	1996-01-01	IVX154274	2012-11-27T16:14:00.000+00:...	1370	60675



Importante verificar que mesmo para as instalações que possuem maior número de horas acumuladas de operação, a quantidade de máquina varia bastante. Por exemplo, temos 06 máquinas na primeira instalação, 4 na segunda e 2 na terceira. Em alguns casos temos apenas uma. Então isso também não interfere na análise inicial.

Qual a distribuição de horas dos registros de movimentações acumulados?

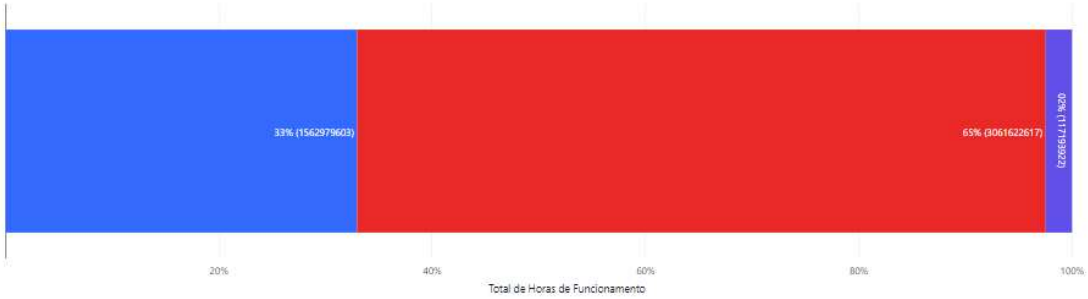
```
%sql

SELECT
    MAX(soma_duracao_prog) AS max_soma_duracao_prog,
    MAX(soma_duracao_for) AS max_soma_duracao_for,
    MAX(soma_duracao_null) AS max_soma_duracao_null,
    MAX(soma_duracao_nor) AS max_soma_duracao_nor,
    SUM(soma_duracao_prog + soma_duracao_for + soma_duracao_null + soma_duracao_nor) AS duracao_total
FROM tb_movcoacum
GROUP BY maquina
```

(9) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [max_soma_duracao_prog: double, max_soma_duracao_for: double ... mais 3 campos]

Tabela | Visualização 1

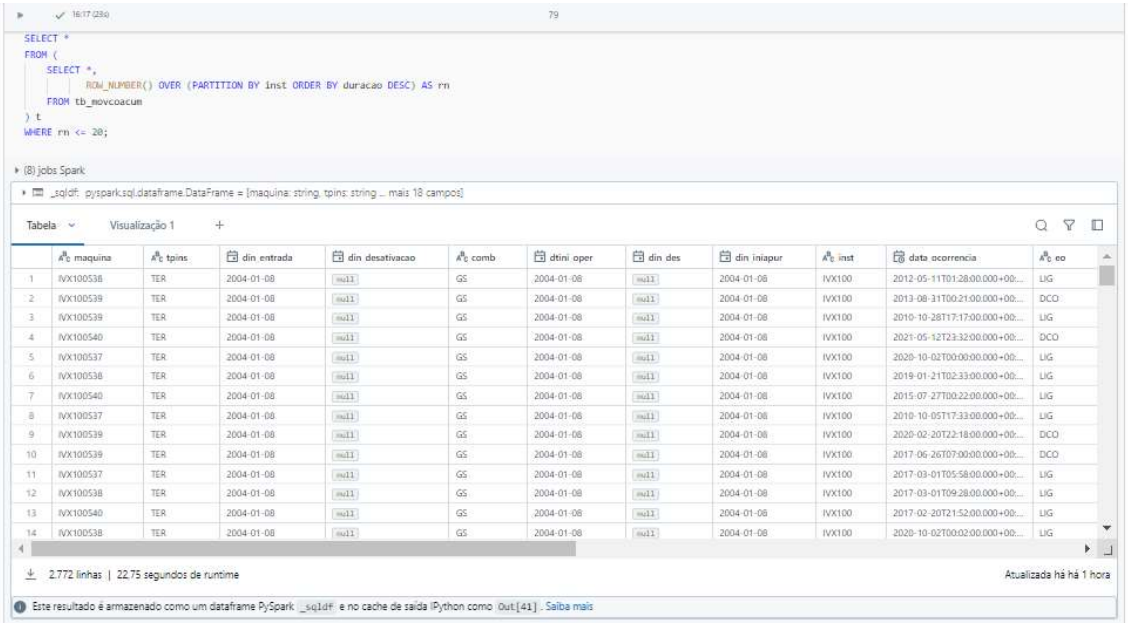
	1.2 max soma duracao prog	1.2 max soma duracao for	1.2 max soma duracao null	1.2 max soma duracao nor	1.2 duracao total
1	3927024	5548162	0	772033	3973473302
2	0	6762857	0	0	6968102905
3	4927314	5385951	0	0	1459191321
4	0	2453760	0	0	622845572
5	7792225	9246625	0	0	193532542
6	6524220	7978620	0	0	400656802
7	6775239	8229639	0	0	1948241B1
8	993000	0	0	0	3964080
9	0	5315774	0	0	8153416939
10	0	6597846	0	0	7101748875
11	0	9238233	0	0	10921356063
12	2974064	3608747	0	0	1121817828
13	1002	721161	0	0	129798182
14	4318920	5179680	0	0	3556525142
15	1335383	7618098	0	0	359262407



Ponto de Atenção 5:

A resposta acima indica que temos mais horas sendo gastas em manutenções ou restrições que fogem do controle dos proprietários das instalações do que pelas programações. Além disso, claramente temos um problema de desequilíbrio nos dados de movimentação. Os dados podem ter sido perdidos na transformação do arquivo csv. Não é comum uma máquina ficar mais tempo com restrição do que em condições normais de funcionamento. E no gráfico acima é possível verificar que a grande proporção de movimentações indica restrições programadas e urgentes (forçadas) e quase não aparece número substanciais de condições de operação normal.

Qual a distribuição de horas dos registros de movimentações por máquina?



```
SELECT *
FROM (
    SELECT *,
           ROW_NUMBER() OVER (PARTITION BY inst ORDER BY duracao DESC) AS rn
    FROM tb_movcoacum
) t
WHERE rn <= 20;
```

(8) jobs Spark

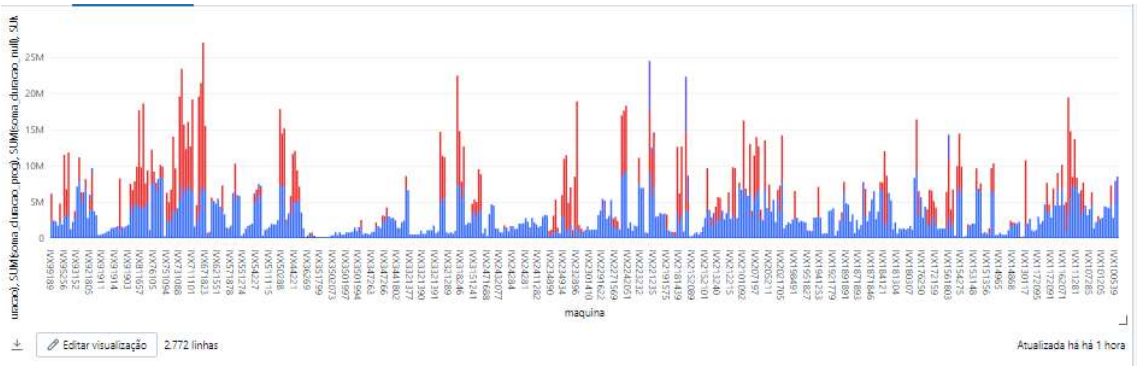
_sqldf: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 18 campos]

Tabela | Visualização 1 | +

	maquina	tpins	din_entrada	din_desativacao	comb	dtini_oper	din_des	din_iniapur	inst	data_ocorrencia	eo
1	IVX100538	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2012-05-11T01:28:00.000+00:...	LIG
2	IVX100539	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2013-08-31T00:21:00.000+00:...	DCO
3	IVX100539	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2010-10-28T17:17:00.000+00:...	LIG
4	IVX100540	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2021-05-12T23:32:00.000+00:...	DCO
5	IVX100537	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2020-10-02T00:00:00.000+00:...	LIG
6	IVX100538	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2019-01-21T02:33:00.000+00:...	LIG
7	IVX100540	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2015-07-27T00:22:00.000+00:...	LIG
8	IVX100537	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2010-10-05T17:33:00.000+00:...	LIG
9	IVX100539	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2020-02-20T22:18:00.000+00:...	DCO
10	IVX100539	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2017-06-26T07:00:00.000+00:...	DCO
11	IVX100537	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2017-03-01T05:58:00.000+00:...	LIG
12	IVX100538	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2017-03-01T09:28:00.000+00:...	LIG
13	IVX100540	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2017-02-20T21:52:00.000+00:...	LIG
14	IVX100538	TER	2004-01-08	null	GS	2004-01-08	null	2004-01-08	IVX100	2020-10-02T00:02:00.000+00:...	LIG

2.772 linhas | 22.75 segundos de runtime — Atualizada há há 1 hora

Este resultado é armazenado como um dataframe PySpark _sqldf e no cache de saída IPython como Out[41]. Saiba mais



Ponto de Atenção 6:

Mesma visão do item acima, a diferença é que agora a visão é por máquina. E a mesma observação é necessária. Temos poucas horas indicando operação normal ou desligamento total das máquinas com esses dados.

Qual a proporção de movimentações para as máquinas mais antigas?

```
SELECT *
FROM (
    SELECT *, ROW_NUMBER() OVER (PARTITION BY maquina ORDER BY din_iniapur DESC) AS rn
    FROM tb_movcoacum
) subquery
WHERE rn = 1
ORDER BY din_iniapur DESC
LIMIT 20;
```

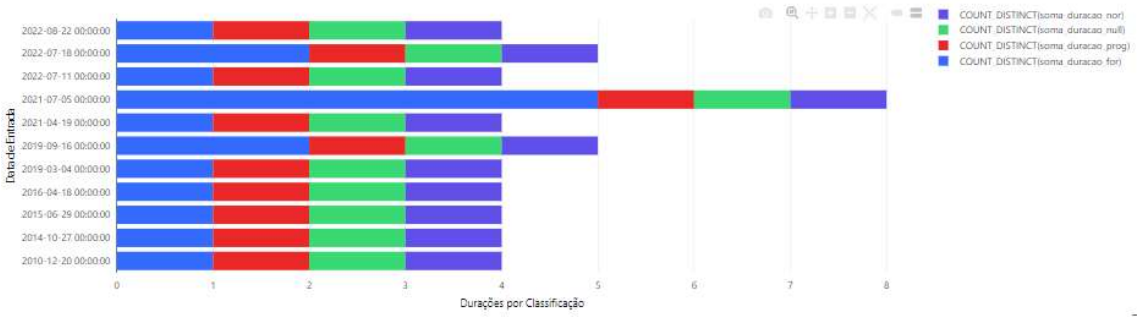
(7) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 18 campos]

Tabela | Visualização 1 | +

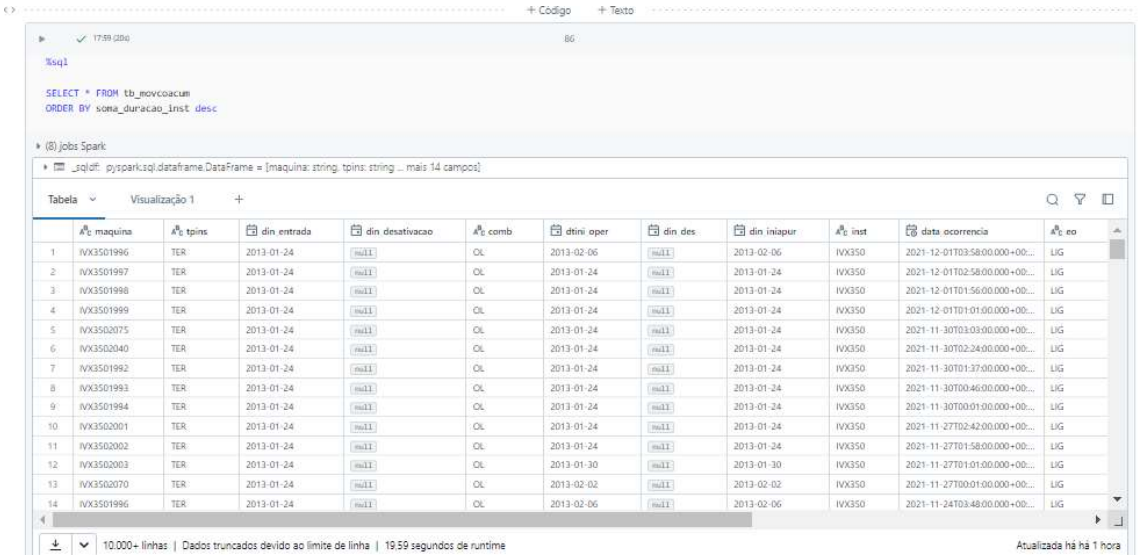
	maquina	tpins	din_entrada	din_desativacao	comb	dtini_oper	din_des	din_iniapur	inst	data_ocorrencia	eo
1	IVX361732	TER	2022-08-24	null	GS	2022-08-24	null	2022-08-24	IVX361	2022-09-04T20:00:00.000+00:...	LIG
2	IVX359730	TER	2022-07-23	null	GS	2022-07-30	null	2022-07-28	IVX359	2022-08-01T00:00:00.000+00:...	DCO
3	IVX359731	TER	2022-07-23	null	GS	2022-07-23	null	2022-07-23	IVX359	2022-07-29T13:35:00.000+00:...	LIG
4	IVX181729	TER	2010-12-23	null	GS	2022-07-20	null	2022-07-20	IVX181	2022-08-09T10:41:00.000+00:...	LIG
5	IVX358099	TER	2022-07-14	null	GS	2022-07-14	null	2022-07-14	IVX358	2022-07-22T12:07:00.000+00:...	DCO
6	IVX351799	TER	2021-07-10	null	GS	2021-08-30	null	2021-08-30	IVX351	2021-08-31T00:00:00.000+00:...	LIG
7	IVX351801	TER	2021-07-10	null	GS	2021-07-17	null	2021-07-17	IVX351	2021-07-20T00:00:00.000+00:...	LIG
8	IVX351798	TER	2021-07-10	null	GS	2021-07-17	null	2021-07-17	IVX351	2021-07-19T07:03:00.000+00:...	LIG
9	IVX351800	TER	2021-07-10	null	GS	2021-07-13	null	2021-07-13	IVX351	2021-07-13T14:22:00.000+00:...	DCO
10	IVX351797	TER	2021-07-10	null	GS	2021-07-10	null	2021-07-10	IVX351	2021-07-12T16:04:00.000+00:...	LIG
11	IVX324343	TER	2021-04-23	null	MS	2021-04-23	null	2021-04-23	IVX324	2021-05-18T12:58:00.000+00:...	LIG
12	IVX1551806	TER	2014-10-31	null	MS	2014-10-31	null	2021-01-01	IVX155	2021-01-02T00:00:00.000+00:...	LIG
13	IVX2171394	TER	2015-06-30	null	MS	2015-06-30	null	2020-01-01	IVX217	2020-09-26T00:00:00.000+00:...	LIG
14	IVX2191572	HID	2019-03-09	null	HD	2019-12-21	null	2019-12-21	IVX219	2020-02-10T00:00:00.000+00:...	LIG

20 linhas | 19,73 segundos de runtime — Atualizada há há 11 minutos



Para as máquinas mais antigas existe um certo equilíbrio de distribuição de horas para cada classificação. A única que se diferencia um pouco é a máquina que entrou em operação em 07/2021 e tem a sua maior fatia como manutenções forçadas. É necessário cruzar esses dados com alguma intercorrência inclusive regional. Para entender os motivos dela fugir do padrão apresentado pelas outras máquinas.

Qual o combustível das máquinas com maior tempo em operação?



```
%sql

SELECT * FROM tb_movcoacum
ORDER BY soma_duracao_inst desc
```

(8) jobs Spark

_sqldf: pyspark.sql.dataframe.DataFrame = [maquina: string, tpins: string ... mais 14 campos]

Tabela | Visualização 1

	maquina	tpins	din entrada	din desativacao	comb	dtini oper	din des	din iniapur	inst	data ocorrencia	eo
1	IVX3501996	TER	2013-01-24	null	OL	2013-02-06	null	2013-02-06	IVX350	2021-12-01T03:58:00.000+00:...	LIG
2	IVX3501997	TER	2013-01-24	null	OL	2013-01-24	null	2013-01-24	IVX350	2021-12-01T02:58:00.000+00:...	LIG
3	IVX3501998	TER	2013-01-24	null	OL	2013-01-24	null	2013-01-24	IVX350	2021-12-01T01:56:00.000+00:...	LIG
4	IVX3501999	TER	2013-01-24	null	OL	2013-01-24	null	2013-01-24	IVX350	2021-12-01T01:01:00.000+00:...	LIG
5	IVX3502075	TER	2013-01-24	null	OL	2013-01-24	null	2013-01-24	IVX350	2021-11-30T03:03:00.000+00:...	LIG
6	IVX3502040	TER	2013-01-24	null	OL	2013-01-24	null	2013-01-24	IVX350	2021-11-30T02:24:00.000+00:...	LIG
7	IVX3501992	TER	2013-01-24	null	OL	2013-01-24	null	2013-01-24	IVX350	2021-11-30T01:37:00.000+00:...	LIG
8	IVX3501993	TER	2013-01-24	null	OL	2013-01-24	null	2013-01-24	IVX350	2021-11-30T00:46:00.000+00:...	LIG
9	IVX3501994	TER	2013-01-24	null	OL	2013-01-24	null	2013-01-24	IVX350	2021-11-30T00:01:00.000+00:...	LIG
10	IVX3502001	TER	2013-01-24	null	OL	2013-01-24	null	2013-01-24	IVX350	2021-11-27T02:42:00.000+00:...	LIG
11	IVX3502002	TER	2013-01-24	null	OL	2013-01-24	null	2013-01-24	IVX350	2021-11-27T01:58:00.000+00:...	LIG
12	IVX3502003	TER	2013-01-24	null	OL	2013-01-30	null	2013-01-30	IVX350	2021-11-27T01:01:00.000+00:...	LIG
13	IVX3502070	TER	2013-01-24	null	OL	2013-02-02	null	2013-02-02	IVX350	2021-11-27T00:01:00.000+00:...	LIG
14	IVX3501996	TER	2013-01-24	null	OL	2013-02-06	null	2013-02-06	IVX350	2021-11-24T03:48:00.000+00:...	LIG

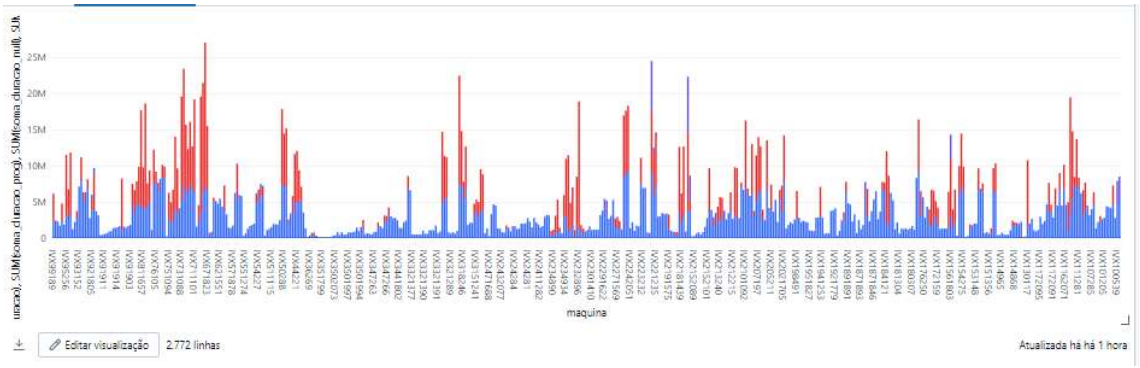
10.000+ linhas | Dados truncados devido ao limite de linha | 19.59 segundos de runtime — Atualizada há há 1 hora

As máquinas de combustível OL são as que possuem mais tempo em operação.

8. Conclusão

Conforme análises anteriores foi possível identificar que as instalações mais antigas não são as que utilizam mais tempo para manutenções programada. Também foi possível identificar que aquelas que tem maior concentração de horas de manutenções programadas por instalação são as do tipo HD. Porém ao verificar por máquina aquela que mais tem manutenções em sua operação são as instalações de combustível OL.

O que mais chama atenção na análise é que a maio proporção de manutenção para as máquinas deveria ser a programada e não a intempestiva ou forçada. Pois se o dono da instalação faz manutenções programadas periodicamente seguindo as boas práticas o numero de horas de intercorrências emergenciais deveria ser menor do que tão maior quanto é apresentado nas análises. Outras informações devem ser utilizadas para complementar esse estudo: como região, clima e demais impactos.



Conseguimos concluir que não é possível determinar apenas através da antiguidade de uma instalação que ela terá mais manutenções programadas que outras. É necessário bater outras informações como tipo da instalação, combustível da máquina e quantidade de máquinas. Será necessário realizar as análises acima olhando para a base completa para confirmar os pontos.

9. Autoavaliação

Acredito que o trabalho executado foi o possível diante das limitações que encontrei. Eu não posso tratar todos os dados que preciso por serem dados sensíveis, além disso, a base necessária para fazer essas análises deveria ser a original que é imensamente maior.

Estamos falando de uma linha do tempo em que cada instalação desenha a sua, então quanto mais informações históricas eu conseguir utilizar será melhor. E fui prejudicada pela perda das informações com relação a utilização do csv. Não tentei outra forma porque fiquei com receio de não conseguir entregar o trabalho.

Mesmo assim, fiquei surpresa em perceber alguns resultados que não esperava e que estão na minha conclusão. O ideal é cruzar as informações desse trabalho com outras análises, conforme mencionei acima, a fim de tentar descobrir se o número de horas de manutenções forçadas se deve às poucas horas de manutenções programadas.

Pontos de melhoria futura:

- Base completa
- Melhor tratativa em dados futuros
- Melhor modelagem
- Utilizar mais informações além das fontes consultadas.