

MVP - Sprint - Engenharia de Dados

1. Objetivos

1.1. Descrição

Busca-se através da construção de um pipeline de dados realizar análises de serviços comerciais e emergenciais que não geram valor a Companhia, ou seja, os serviços impedidos no atendimento ao Cliente. Além do impacto financeiro do ponto de vista da corporação, quando ocorre os impedimentos dos serviços, gera-se a insatisfação de nossos Clientes.

Neste trabalho pretende-se, analisar os dados do ano de 20223, entender os principais motivos dos impedimentos de serviços comerciais e emergenciais.

- Identificar as regiões com maior índices de impedimentos de serviços comerciais e emergenciais;
- Verificar se existe alguma sazonalidade, quanto aos impedimentos de serviços comerciais e emergenciais;
- Elencar os principais motivos de impedimentos de serviços comerciais e emergenciais;
- Comparar as equipes executoras, analisando se a proporção é igualitária de impedimentos ou existe descrepância;
- Criar um ranking "top five" entre os motivos de reprovações;
- Criar um ranking "top ten" das equipes com os maiores índices de impedimentos.

2. Definição da Fonte de Dados

2.1. Fonte de Dados

Ao definir os objetivos, optou-se por buscar uma solução a um problema real de negócio dentro da Companhia que trabalha, no setor de engenharia, a qual gerencia os desempenhos e logísticas das equipes executoras dos serviços comerciais e emergenciais. Diante disso, foi necessário buscar as informações necessárias, neste caso os dados, no sistema da Companhia. Para a disponibilidade deste, foi necessário a descaracterização das informações na extração utilizando a linguagem SQL e transformando estes dados em arquivos CSV. Buscando proteger as informações prevista em legislação, tal como a LGPD e dados confidenciais da Companhia, se fez necessário esta ação e forma de obtenção dos dados.

3. Modelagem dos Dados

3.1. Extração dos Dados

As informações da corporação são armazenadas em servidores físicos próprios em suas dependências. Para este caso específico, os dados estão armazenados em um dos servidores da Companhia, em um banco gerenciado através do SGBD do PostgreSQL.

A conexão neste banco de dados foi realizada através da conexão abaixo, pelo jupyter notebooklab.

```
engine = create_engine('postgresql+psycopg2://xxx:xxx@Pxxxxxx/*')
```

```
conn = engine.connect()
```

Para a geração dos dataset foram executadas as query apresentadas na descrição de cada tabela através do jupyter notebooklab, conectado ao banco de dados PostgreSQL, onde obtevesse os dataset em CSV, conforme as imagens apresentadas nas figuras abaixo.

Obs.: foram descaracterizadas as informações possíveis de identificar Clientes e/ou dados confidenciais.

3.1.1. tb_comerciais

```
query = """ select cast(SPLIT_PART(atributo1,'.',2) as atributo1)fatormultiplicador as
ordem_servico, cast(SPLIT_PART(atributo2,'.',3) as atributo2)fatormultiplicador as seq_servico,
atributo3 as cod_servico, atributo4 as cod_veiculo, atributo5 as cod_equipe,
atributo6fatormultiplicador as colab_1, atributo7fatormultiplicador as colab_2, atributo8 as
cod_susp, atributo9 as cod_impedido, atributo10 as dt_saida, atributo11 as dt_inicio,
atributo12 as dt_fim, atributo13 as dt_termino, atributo14 as cod_local,
substring(atributo15,1,5) as sub_regiao, (atributo16-atributo17) as Km FROM
scheme.tabela_comercial where atributo6 not in (0,99999) and atributo3 <> 0 and atributo1
<> 'Protocolo' and atributo12
between '2023-01-01' and current_date;"""
```

	tb_comercial	#	pd.read_sql(tb_comercial,conn)													
	tb_comercial	#	shape													
	tb_comercial	#	(2130331, 16)													
	tb_comercial	#	head(10)													
6:	ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2	cod_impedido	dt_saida	dt_inicio	dt_fim	dt_termino	cod_local	sub_regiao	Km	
0	9.100753e+16	35.0	195	E2136	CB625	980187	980307	42	0	2023-01-01 05:22:00	2023-01-01 05:57:00	2023-01-01 06:03:00	2023-01-01 06:52:46	7624	SUL 0	36
1	9.100746e+16	35.0	9925	E3P70	MA945	972165	972417	20	8001	2023-01-01 07:59:00	2023-01-01 08:09:00	2023-01-01 08:29:00	2023-01-01 08:30:00	4402	MGN 0	5
2	9.100747e+16	35.0	9947	E3H22	LO510	907242	985230	0	0	2023-01-01 07:31:00	2023-01-01 08:01:00	2023-01-01 08:30:00	2023-01-01 08:30:00	4156	LNN 4	12
3	9.105254e+16	35.0	3404	V30138	PG777	137079	777112	0	0	2023-01-01 09:59:00	2023-01-01 09:59:00	2023-01-01 10:04:00	2023-01-01 10:04:00	5428	PG071	1
4	9.100752e+16	35.0	9947	E3H23	LO511	909414	986055	0	0	2023-01-01 10:01:00	2023-01-01 10:07:00	2023-01-01 10:25:00	2023-01-01 10:25:00	4150	LNA 3	4
5	9.100706e+16	70.0	331	E3L27	CB151	973695	973698	51	0	2023-01-01 09:41:00	2023-01-01 10:29:00	2023-01-01 10:31:00	2023-01-01 10:31:00	1510	CTC 1	22
6	9.100747e+16	35.0	9947	E3H23	LO511	909414	986055	2	0	2023-01-01 10:25:00	2023-01-01 10:28:00	2023-01-01 10:32:00	2023-01-01 10:32:00	4150	LNA 3	1
7	9.100749e+16	35.0	9947	E3H23	LO511	909414	986055	0	0	2023-01-01 10:32:00	2023-01-01 10:39:00	2023-01-01 10:48:00	2023-01-01 10:48:00	4150	LNA 3	4
8	9.100706e+16	70.0	331	E3L27	CB151	973695	973698	51	0	2023-01-01 10:31:00	2023-01-01 10:43:00	2023-01-01 10:52:00	2023-01-01 10:52:00	1510	CTC 1	2
9	9.100706e+16	70.0	331	E3L27	CB151	973695	973698	51	0	2023-01-01 10:52:00	2023-01-01 10:58:00	2023-01-01 11:18:00	2023-01-01 11:18:00	1510	CTC 1	4

3.1.2. tb_emergenciais

```
query = """ select
(atributo1)fatormultiplicador as ordem_servico, atributo2 as cod_servico, atributo3 as
cod_veiculo, atributo4 as cod_equipe, atributo5fatormultiplicador as colab_1,
atributo6*fatormultiplicador as colab_2, atributo7 as cod_impedido, atributo8 as cod_susp,
atributo9 as dt_saida, atributo10 as dt_inicio, atributo11 as dt_fim, atributo12 as dt_termino,
atributo13 as cod_local, atributo14(atributo14,1,5) as sub_regiao, atributo15 as Km from
```

schema.tabela_emergencial where atributo5 not in (0,999999) and atributo11 between '2023-01-01' and current_date'''

```
[9]: tb_emergencial pd.read_sql(tb_emergencial, conn)
[10]: tb_emergencial.shape
[10]: (546558, 15)
[11]: tb_emergencial.head(10)
[11]: 
  ordem_servico cod_servico cod_veiculo cod_equipe colab_1 colab_2 cod_impedido cod_susp      dt_saida      dt_inicio      dt_fim      dt_termino cod_local sub_regiao   km
  0  24910905000  6002        E3499  CB054  974637  974640        22      52  2022-12-31 22:51:00  2023-01-01 00:00:00  2023-01-01 00:02:00  4264    M1B 0  40
  1  24911005500  6001        E3K06  CB175  984888  985038        71      52  2022-12-31 23:29:00  2022-12-31 23:51:00  2023-01-01 00:03:00  2023-01-01 00:05:00  1510    CTC 1  14
  2  19155807000  6001        V30090  LOT13  151302  2402082       22      52  2023-01-01 00:01:00  2023-01-01 00:03:00  2023-01-01 00:04:00  2023-01-01 00:04:00  2332    CLP 0  1
  3  19155595500  6002        V30090  LOT13  151302  2402082       22      52  2023-01-01 00:04:00  2023-01-01 00:05:00  2023-01-01 00:07:00  2023-01-01 00:07:00  2332    CLP 0  2
  4  25427898000  6002        E3F42  CA843  893694  964815       38      89  2023-01-01 00:01:00  2023-01-01 00:05:00  2023-01-01 00:10:00  2023-01-01 00:10:00  2806    JET 0  1
  5  24908211000  6001        E2136  CB625  980187  983037       02      50  2022-12-31 22:10:00  2022-12-31 22:44:00  2023-01-01 00:11:00  2023-01-01 00:11:00  7624    SUL 0  15
  6  24910951500  6001        E3N83  CB260  902052  983601       03      51  2022-12-31 23:06:00  2022-12-31 23:29:00  2023-01-01 00:11:00  2023-01-01 00:17:00  5434    PTP 0  15
  7  19155622500  6001        V30090  LOT13  151302  2402082       22      52  2023-01-01 00:07:00  2023-01-01 00:14:00  2023-01-01 00:15:00  2023-01-01 00:31:00  2332    CLP 0  5
  8  19155721500  6001        E3809  LO529  869694  966621       74      52  2022-12-31 23:55:00  2023-01-01 00:14:00  2023-01-01 00:19:00  2023-01-01 00:19:00  4156    LNA 4  10
  9  16275577500  6001        T30002  PG801  140697  75204       85      52  2022-12-31 23:55:00  2023-01-01 00:16:00  2023-01-01 00:19:00  2023-01-01 00:19:00  5428    PG070  19

[12]: tb_emergencial.to_csv('D:/tb_emergencial.csv', index = False)
```

3.1.3. tb_regioes

```
query = """ select atributo1 as codigo, CASE WHEN(atributo2 LIKE '%%XXXXXX%%') THEN
REPLACE(atributo2, 'XXXXXX', 'DPTO01') WHEN(atributo2 LIKE '%%XXXXXX%%') THEN
REPLACE(atributo2, 'XXXXXX', 'DPTO02') WHEN(atributo2 LIKE '%%XXXXXX%%') THEN
REPLACE(atributo2, 'XXXXXX', 'DPTO03') WHEN(atributo2 LIKE '%%XXXXXX%%') THEN
REPLACE(atributo2, 'XXXXXX', 'DPTO04') WHEN(atributo2 LIKE '%%XXXXXX%%') THEN
REPLACE(atributo2, 'XXXXXX', 'DPTO05') END AS dpto, CASE WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG001') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG002') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG003') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG004') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG005') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG006') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG007') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG008') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG009') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG010') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG011') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG012') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG013') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG014') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG015') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG016') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG017') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG018') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG019') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG020') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG021') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG022') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG023') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG024') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG025') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX', 'AG026') WHEN(atributo3 LIKE
```

```
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG027') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG028') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG029') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG030') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG031') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG032') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG033') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG034') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG035') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG036') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG037') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG038') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG039') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG040') WHEN(atributo3 LIKE
'%%XXXXXX%%') THEN REPLACE(atributo3, 'XXXXX','AG041') END AS gerencia
from scheme.tabela_regioes; ""
```

```
[15]: tb_regioes = pd.read_sql(tb_regioes, conn)
[16]: tb_regioes.shape
[16]: (1151, 3)
[17]: tb_regioes.head(10)
[17]:   codigo      dpto  gerencia
[0]  2350  DPTO01  AG001
[1]  3388  DPTO01  AG001
[2]  1006  DPTO01  AG001
[3]  7360  DPTO01  AG001
[4]  4894  DPTO01  AG001
[5]  7078  DPTO01  AG001
[6]  2272  DPTO01  AG001
[7]  2308  DPTO01  AG001
[8]  2416  DPTO01  AG001
[9]  1402  DPTO01  AG001
[19]: tb_regioes.to_csv('d:/tb_regioes.csv', index = False)
```

3.1.4. tb_grupo_servicos

```
query = """ SELECT atributo1, case WHEN atributo1 in
(101,102,103,111,112,113,150,181,182,183,186,191,192,193,194,195,196,197,198,199) then
'Grupo_100' WHEN atributo1 in (311,312,313,315,316,319,323,330,331,394,396,397,398,399)
then 'Grupo_300' WHEN atributo1 in
(401,402,403,404,405,406,407,409,412,417,419,420,430,440,441,442,443,449,499) then
'Grupo_400' WHEN atributo1 in (501,503,510,511,513,517,518,519,520,530,540,550,560) then
'Grupo_500' WHEN atributo1 in (2001,2002,2003,2004,2005,2006,2008,2907,2909) then
'Grupo_2000' WHEN atributo1 in
(3230,3231,3232,3233,3234,3235,3236,3237,3238,3239,3240,3245,3294,3403,3404,3405,3495)
then 'Grupo_3000' WHEN atributo1 in (6001,6002) then 'Grupo_6000' WHEN atributo1 in
(9003,9004,9901,9902,9903,9904,9905,9906,9907,9908,9909,9910,9911,9912,9913,9914,9915,991
9921,9922,9923,9924,9925,9926,9927,9930,9933,9934,9938,9939,9940,9941,9942,9943,9944,9945
9953,9954,9955,9956,9957,9958,9959,9960,9961,9962,9963,9964,9965,9966,9967,9968,9969,9970
9978,9979,9980,9982,9984,9985,9986,9987,9988,9989,9991,9992,9993,9994,9995,9996,9997,9998
then 'Grupo_9000' end as situacao FROM scheme.tabela_grupo_servicos; """
```

```
[22]: tb_grupo_servicos = pd.read_sql(tb_grupo_servicos, conn)
[23]: tb_grupo_servicos.shape
[23]: (184, 2)
[24]: tb_grupo_servicos.head(10)
[24]:

| cod | grupo         |
|-----|---------------|
| 0   | 101 Grupo_100 |
| 1   | 102 Grupo_100 |
| 2   | 103 Grupo_100 |
| 3   | 111 Grupo_100 |
| 4   | 112 Grupo_100 |
| 5   | 113 Grupo_100 |
| 6   | 150 Grupo_100 |
| 7   | 181 Grupo_100 |
| 8   | 182 Grupo_100 |
| 9   | 183 Grupo_100 |


[25]: tb_grupo_servicos.to_csv('d:/tb_grupo_servicos.csv',index = False)
```

3.1.5. tb_corte_impedido

```
query = """ SELECT atributo1, case WHEN atributo1 in
(1,2,3,4,15,20,35,39,40,41,42,48,49,50,51,52,53,54,55,56,57,58,59,60,61,70,72,99,10000) then
'Corte_Executado' WHEN atributo1 in
(6,7,8,9,10,12,13,14,16,17,18,19,21,22,23,24,25,26,27,71) then 'Corte_Impedido' end as
situacao FROM scheme.tabela_corte_impedido order by atributo1 asc; """
```

```
[27]: tb_corte_impedido = pd.read_sql(tb_corte_impedido, conn)
[28]: tb_corte_impedido.shape
[28]: (46, 2)
[29]: tb_corte_impedido.head(10)
[29]:

| cod | situacao        |
|-----|-----------------|
| 0   | Corte_Executado |
| 1   | Corte_Executado |
| 2   | Corte_Executado |
| 3   | Corte_Executado |
| 4   | Corte_Impedido  |
| 5   | Corte_Impedido  |
| 6   | Corte_Impedido  |
| 7   | Corte_Impedido  |
| 8   | Corte_Impedido  |
| 9   | Corte_Impedido  |


[30]: tb_corte_impedido.to_csv('d:/tb_corte_impedido.csv',index = False)
```

3.1.6. tb_comercial_impedido

```
query = """ SELECT DISTINCT atributo1 as cod, case WHEN atributo1 in (0) then
'Com_Executado' WHEN atributo1 in
(100,111,112,113,114,150,200,211,212,213,214,215,216,217,218,219,220,300,311,312,313,314,315
500,511,512,513,514,515,516,517,518,519,520,521,600,611,612,613,614,615,616,617,618,619,620,
812,813,814,815,816,817,818,819,900,911,912,913,914,915,916,917,1000,1011,1012,1013,1014,10
1113,1114,1115,1116,1117,1200,1211,1212,1213,1214,1215,1216,1217,1218,1219,1220,1300,1311
8009,8010,8011,8016,8018,8030,8100,8150,8165,8176,8187,8299,8301,8302,8310,8315,8341,8351
then 'Com_Impedido' end as situacao FROM scheme.tabela_comercial_impedido order by
atributo1 asc; """
```

```
[32]: tb_comercial_impedido = pd.read_sql(tb_comercial_impedido, conn)
[33]: tb_comercial_impedido.shape
[33]: (144, 2)
[34]: tb_comercial_impedido.head(10)
[34]:
  cod      situacao
  0      0  Com_Executado
  1     100  Com_Impedido
  2     111  Com_Impedido
  3     112  Com_Impedido
  4     113  Com_Impedido
  5     114  Com_Impedido
  6     200  Com_Impedido
  7     211  Com_Impedido
  8     212  Com_Impedido
  9     213  Com_Impedido
[35]: tb_comercial_impedido.to_csv('d:/tb_comercial_impedido.csv',index = False)
```

3.1.7. tb_componente_imped

```
query = """ SELECT atributo1, case WHEN atributo1 in
('A1','A3','A4','A5','A7','A8','A9','B1','B2','B3','B4','B6','B8','B9','C1','C2','C3','D1','D2','D3','E1','E2','E3',
'E5','E6','E7','E8','E9','F2','G1','H1','H2','H3','H4','H5','H6','H7','H8','H9','I1','I2','I6','I7','I8','I9','L1','L2',
'L3','L4','L5','L6','L7','P1','P3','P4','T1','T4','T5','T7','T8','T9','01','02','03','04','06','07','08','09','11','12',
'14','16','17','18','19','20','21','22','23','24','25','26','27','28','29','30','31','32','33','34','35','36','38','40',
'41','42','44','45','46','47','48','49','50','51','52','53','54','56','60','61','62','63','64','65','66','67','74','79',
'82','83','84','85','88','89','90','92','93') then 'Emer_' end as situacao FROM
scheme.tabela_componente_impedido order by atributo1 asc; """
```

```
[37]: tb_emer_componente_imped = pd.read_sql(tb_emer_componente_imped, conn)
[38]: tb_emer_componente_imped.shape
[38]: (129, 2)
[39]: tb_emer_componente_imped.head(10)
[39]:
  cod      situacao
  0      01  Emer_
  1      02  Emer_
  2      03  Emer_
  3      04  Emer_
  4      06  Emer_
  5      07  Emer_
  6      08  Emer_
  7      09  Emer_
  8      11  Emer_
  9      12  Emer_
[40]: tb_emer_componente_imped.to_csv('d:/tb_emer_componente_imped.csv',index = False)
```

3.1.8. tb_causa_imped

```
query = """ SELECT atributo1, case WHEN atributo1 in
('01','02','03','04','05','06','07','08','09','10','13','15','18','19','20','21','22','23','24','26','28','33','34','38',
'39','40','41','42','43','44','45','47','48','50','51','52','54','60','68','69','74','76','77','78','82','83','87','88',
'93','95','98','E1','E2','E3','E4','E5','H1','H2','H3','K1','K2','K3','K4','K5','K6','K7','K8','K9','P1') then
'Executada' WHEN atributo1 in ('71','75','85') then 'Improcedente' end as situacao FROM
scheme.tabela_causa_imped order by atributo1 asc; """
```

```
[42]: tb_emer_causa_imped = pd.read_sql(tb_emer_causa_imped, conn)
[43]: tb_emer_causa_imped.shape
[43]: (75, 2)
[44]: tb_emer_causa_imped.head(10)
[44]:   cod  situacao
  0  01  Executada
  1  02  Executada
  2  03  Executada
  3  04  Executada
  4  05  Executada
  5  06  Executada
  6  07  Executada
  7  08  Executada
  8  09  Executada
  9  10  Executada
[45]: tb_emer_causa_imped.to_csv('d:/tb_emer_causa_imped.csv', index = False)
```

3.1.9. Catálogo de Dados e seus domínios

Realizado a criação de uma conta gratuita na plataforma databricks azure da microsoft, para a realização do projeto de MVP da disciplina de Engenharia de Dados da Universidade PUC RIO. Após a criação e configuração do sistema, realizado a conexão via cluster, para a realização dos UPload dos dataset CSV, para a tratativa dos arquivicos através do ETL e posteriormente realizar a conexão com o Power BI para a criação dos dashboard, para responder aos objetivos de acordo com as regras de negócios da Comapnhia.

O carregamento dos datasets, foram realizados com sucesso, na plataforma databricks através da função Engenharia de Dados/Ingestão de dados no diretório default do catalágo, conforme abaixo:

The screenshot shows the Databricks Catalog Explorer interface. On the left, there is a tree view of databases: 'hive_metastore' and 'default'. The 'default' database is expanded, showing tables: 'tb_comercial', 'tb_comercial_impedido', 'tb_corte_impedido', 'tb_emer_causa_imped', 'tb_emer_componente_imped', 'tb_emergencial', 'tb_grupo_servicos', and 'tb_regioes'. On the right, there is a detailed view of the 'default' database. It shows the database is a 'Default Hive database' with '8 tabelas'. Below this, a table lists the names of the 8 tables: tb_comercial, tb_comercial_impedido, tb_corte_impedido, tb_emer_causa_imped, tb_emer_componente_imped, tb_emergencial, tb_grupo_servicos, and tb_regioes.

Para realizar a descrição dos datasets, não foi possível realizar através da plataforma do databricks, devido a não ter acesso para tal na versão gratuita, conforme é apresentado na tela a seguir.

Diante disso, segue a baixo a descrição das tabelas criadas e carregadas na plataforma.

3.1.9.1. tb_comercial

Para esta análise dividiu-se a tabela entre os atributos numéricos e categórios, e tratado através da função **DESCRIBE()** no python com o pandas, conforme segue.

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

	ordem_servico	seq_servico	cod_servico	colab_1	colab_2	cod_susp	cod_impedido	cod_local	km
count	2.130331e+06	2.130331e+06	2.130331e+06	2.130331e+06	2.130331e+06	2.130331e+06	2.130331e+06	2.130331e+06	2.130331e+06
mean	9.105330e+16	3.396830e+03	1.914085e+03	8.446194e+05	4.617815e+05	1.962773e+01	3.044101e+02	4.320051e+03	1.009410e+02
std	1.027630e+13	6.026078e+03	3.271612e+03	4.334180e+05	5.526626e+05	2.147447e+01	1.498038e+03	1.919684e+03	5.884857e+03
min	9.079070e+16	3.500000e+01	1.010000e+02	5.522700e+04	0.000000e+00	0.000000e+00	0.000000e+00	1.002000e+03	0.000000e+00
25%	9.105381e+16	3.500000e+01	4.020000e+02	8.988900e+05	0.000000e+00	2.000000e+00	0.000000e+00	2.488000e+03	1.000000e+00
50%	9.105524e+16	3.500000e+01	4.420000e+02	9.827220e+05	1.388430e+05	3.000000e+00	0.000000e+00	4.360000e+03	1.000000e+00
75%	9.105669e+16	4.795000e+03	5.400000e+02	9.926520e+05	9.814860e+05	4.200000e+01	0.000000e+00	5.692000e+03	5.000000e+00
max	9.105831e+16	7.875000e+04	9.999000e+03	2.412054e+06	2.412045e+06	9.900000e+01	8.906000e+03	7.882000e+03	3.610022e+06

```
1 # apresenta um resumo de estatística das variáveis categóricas
2 df_comercial[cats_comer].describe()
```

	cod_veiculo	cod_equipe	dt_saida	dt_inicio	dt_fim	dt_termino	sub_regiao
count	2130331	2118474	2127063	2129875	2130331	2130331	2130327
unique	1783	2589	193202	192479	192650	334847	1156
top	E3J21	CB476	2023-06-06 10:39:00	2023-02-28 10:08:00	2023-04-04 11:30:00	2023-03-27 10:16:00	SUL 0
freq	11356	11337	51	51	51	50	121263

- **ordem_servico** - do tipo int - representa a ordem de serviço gerada para o atendimento ao cliente;
- **seq_servico** - do tipo int - representa a sequencia gerada para acompanhamento do atendimento ao cliente;

- cod_serviço - do tipo int - representa o código para identificar o grupo de serviços a ser realizado;
- colab_1 - do tipo int - representa o código de identificação do colaborador executor do serviço;
- colab_2 - do tipo int - representa o código de identificação do colaborador executor do serviço;
- cod_susp - do tipo int - representa o código da conclusão das ordem de suspensão do fornecimento;
- cod_impedimento - do tipo int - representa o código de impedimento ou execução da ordem de serviço;
- cod_local - do tipo int - representa a localidade da região do serviço realizado;
- km - do tipo int - representa a quantidade de quilometros informado que a equipe rodou para o atendimento das ordens de serviços;
- cod_veiculo - do tipo string - representa a identificação do veículo executor das ordens de serviços;
- cod_equipe - do tipo string - representa a identificação da equipe executora das ordens de serviços;
- dt_saida - do tipo datetime - representa o dia e horário que a equipe executora deu saída para o atendimento da ordem de serviço;
- dt_inicio - do tipo datetime - representa o dia e horário que a equipe executora iniciou o atendimento da ordem de serviço;
- dt_fim - do tipo datetime - representa o dia e horário que a equipe executora encerrou o atendimento da ordem de serviço;
- dt_termino - do tipo datetime - representa o dia e horário que a equipe executora deu retorno do atendimento da ordem de serviço;
- sub_regiao - do tipo string - identifica a área local do serviço, como bairro ou vila.

3.1.9.2. tb_emergencial

Para esta análise dividiu-se a tabela entre os atributos numéricos e categórios, e tratado através da função **DESCRIBE()** no python com o pandas, conforme segue.

```
1 # apresenta um resumo de estatística das variáveis numéricas
2 df_emergencial[nums_emerg].describe()
```

	cod_servico	colab_1	colab_2	cod_impedido	cod_local	km
count	546558.000000	5.465580e+05	5.465580e+05	546558.000000	546558.000000	546558.000000
mean	6001.441124	8.358469e+05	8.525477e+05	38.369189	4288.343806	19.509011
std	0.496522	4.478982e+05	4.580734e+05	31.653513	1910.906199	21.659662
min	6001.000000	6.020400e+04	0.000000e+00	1.000000	1006.000000	1.000000
25%	6001.000000	8.645520e+05	8.945940e+05	4.000000	2512.000000	5.000000
50%	6001.000000	9.730230e+05	9.796080e+05	39.000000	4246.000000	12.000000
75%	6002.000000	9.867450e+05	9.915300e+05	71.000000	5794.000000	27.000000
max	6002.000000	2.411964e+06	2.412246e+06	98.000000	7882.000000	485.000000

O comando demorou 0,24 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 20:50:07 em Leandro Portela da Luz's Cluster

```

1  # apresenta um resumo de estatística das variáveis categóricas
2  df_emergencial[cats_emerg].describe()

```

	cod_veiculo	cod_equipe	cod_susp	dt_saida	dt_inicio	dt_fim	dt_termino	sub_regiao
count	546558	546558	546558	546558	546558	546558	546558	546517
unique	1260	1757	78	235452	235127	234584	237069	1093
top	E3P69	CB144	52	2023-07-13 16:14:00	2023-07-13 16:34:00	2023-09-05 15:47:00	2023-09-05 16:30:00	SUL 0
freq	1674	1674	161737	20	22	20	19	20938

0 comando demorou 1,10 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 20:50:09 em Leandro Portela da Luz's Cluster

- ordem_servico - do tipo int - representa a ordem de serviço gerada para o atendimento ao cliente;
- cod_serviço - do tipo int - representa o código para identificar o grupo de serviços a ser realizado;
- colab_1 - do tipo int - representa o código de identificação do colaborador executor do serviço;
- colab_2 - do tipo int - representa o código de identificação do colaborador executor do serviço;
- cod_impedimento - do tipo int - representa o código do tipo da execução da ordem de serviço;
- cod_local - do tipo int - representa a localidade da região do serviço realizado;
- km - do tipo int - representa a quantidade de quilometros informado que a equipe rodou para o atendimento das ordens de serviços;
- cod_veiculo - do tipo string - representa a identificação do veículo executor das ordens de serviços;
- cod_equipe - do tipo string - representa a identificação da equipe executora das ordens de serviços;
- cod_susp - do tipo string - representa o código da conclusão das ordem de suspensão do fornecimento;
- dt_saida - do tipo datetime - representa o dia e horário que a equipe executora deu saída para o atendimento da ordem de serviço;
- dt_inicio - do tipo datetime - representa o dia e horário que a equipe executora iniciou o atendimento da ordem de serviço;
- dt_fim - do tipo datetime - representa o dia e horário que a equipe executora encerrou o atendimento da ordem de serviço;
- dt_termino - do tipo datetime - representa o dia e horário que a equipe executora deu retorno do atendimento da ordem de serviço;
- sub_regiao - do tipo string - identifica a área local do serviço, como bairro ou vila.

3.1.9.3. tb_comercial_impedido, tb_corte_impedido, tb_emer_componente_imped, tb_emer_causa_imped, tb_regioes e tb_grupo_serviços

Estes datasets são tabelas com os códigos dos tipos de serviços, localidades, execução ou de impedimentos das ordens de serviços. Além disso, apresentam a descrição dos tipos de serviços, os departamentos, regiões dos serviços, os grupos de serviços e formas de impedimentos.

- cod - do tipo int - descreve o código da localidade, serviço, execução ou impedimento dos serviços;

- situação - descrição do tipo de serviço, do local, departamento, região, forma de execução ou impedimento.

Exemplo:

The screenshot shows a Databricks dataset named 'default.tb_comercial_impedido'. The dataset has two columns: 'cod' (type int) and 'situacao' (type string). The interface includes tabs for 'Colunas', 'Dados de exemplo', 'Detalhes', 'Permissões', and 'Histórico'. A search bar at the top says 'Filtrar colunas...'. A 'Criar' button is in the top right corner.

Coluna	Tipo	Comentário
cod	int	
situacao	string	

Conexão na Plataforma Databricks e conexão com Power BI

Realizado a criação de uma conta gratuita na plataforma databricks azura em nuvem da microsoft, para a realização do projeto de MVP da disciplina de Engenharia de Dados da Universidade PUC RIO. Após a criação e configuração do sistema, realizado a conexão via Cluster, para a realização dos Upload dos dataset CSV, para a tratativa dos ETL e posteriormente realizar a conexão com o Power Bi para a criação dos dashboard, para responder aos objetivos de acordo com as regras de negócios da Comapnhia.

4. Carga dos dados na plataforma em nuvem

A partir desta etapa, realizou-se a carga dos datasets no ambiente python, para as análises e tratativas necessários dos dados para realização de cálculos estatísticos e responder as questões de negócios da corporação.

Primeiramente, carregou-se as bibliotecas necessária para as análise e tratativa dos datasets.

```
Cmd 1
MPV - Sprint: Engenharia de Dados

1 import pandas as pd
2 import numpy as np
3 from pyspark.sql import SparkSession

0 comando demorou 0,79 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:11:59 em Leandro Portela da Luz's Cluster
```

Realizando a carga do dataset tb_comercial no ambiente em nuvem do databricks.

1. Dataset tb_comerciais

```

1  # Carregando o datasets com dados servicos comerciais
2  query = """
3  SELECT *
4  FROM `hive_metastore`.`default`.`tb_comercial`
5  """
6  df_spark = spark.sql(query)
7  df_comercial = df_spark.toPandas()

```

▼ (1) trabalhos Spark

► Job 8 Ver (Etapas: 1/1)

```

▼ df_spark: pyspark.sql.dataframe.DataFrame
  ordem_servico: float
  seq_servico: float
  cod_servico: integer
  cod_veiculo: string
  cod_equipe: string
  colab_1: integer
  colab_2: integer
  cod_susp: integer
  cod_impedido: integer
  dt_saida: string
  dt_inicio: string
  dt_fim: string
  dt_termino: string
  cod_local: integer
  sub_regiao: string
  km: integer

```

0 comando demorou 6,70 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:15:56 em Leandro Portela da Luz's Cluster

Realizando as primeiras análises do dataset de forma visual primeiramente, referente as primeiras linhas conforme segue.

```

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

```

	ordem_servico	seq_servico	cod_servico	cod_veiculo	cod_equipe	colab_1	colab_2	cod_susp	cod_impedido	dt_saida	dt_inicio	dt_fim	dt_termino	cod_local	sub_regiao	km
0	9.100752e+16	35.0	195	E2136	CB625	980187	980307	42	0	2023-01-01 05:22:00	2023-01-01 05:57:00	2023-01-01 06:03:00	2023-01-01 06:52:46	7624	SUL 0	36
1	9.100745e+16	35.0	9925	E3P70	MA945	972165	972417	20	8001	2023-01-01 07:59:00	2023-01-01 08:09:00	2023-01-01 08:29:00	2023-01-01 08:29:00	4402	MGN 0	5
2	9.100747e+16	35.0	9947	E3H22	LO510	907242	985230	0	0	2023-01-01 07:31:00	2023-01-01 08:01:00	2023-01-01 08:30:00	2023-01-01 08:30:00	4156	LNN 4	12

0 comando demorou 0,09 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:17:12 em Leandro Portela da Luz's Cluster

Identificou-se a necessidade de tratar os valores ausentes dos atributos cod_equipe, dt_saida, dt_inicio e sub_regiao, conforme apresenta a tela abaixo.

```

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

```

```

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          456
dt_fim              0
dt_termino          0
cod_local            0
sub_regiao           4
km                  0
dtype: int64

```

0 comando demorou 2,03 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:19:30 em Leandro Portela da Luz's Cluster

Num primeiro momento, optou-se por utilizar um valor 'coringa' para eliminar os valores ausentes do datasets, veja a seguir.

```

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

0 comando demorou 2,09 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:11:59 em Leandro Portela da Luz's Cluster
Cmd 9

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

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_imedido 0
dt_saida 0
dt_inicio 0
dt_fim 0
dt_termino 0
cod_local 0
sub_regiao 0
km 0
dtype: int64

0 comando demorou 2,49 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:11:59 em Leandro Portela da Luz's Cluster

```

Na sequencia iniciou-se as tratativas individuais por atributo. Para a coluna "dt_inicio", conhecendo a regra de negócio e valores padrões estabelecidos na corporação, adotou-se para incrementar os valores ausentes no atributo "dt_inicio", a utilização do valor da coluna "dt_fim", respectivamente da mesma linha do registro ausente, com 30 minutos antecedentes, pois este é um tempo médio padrão para a execução da maioria das ordens de serviços estabelecido pela corporação.

```

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

0 comando demorou 0,18 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:27:54 em Leandro Portela da Luz's Cluster

```

Veja que após a tratativa os valores coringa não se apresentam mais na coluna dt_inicio, foram sanados os valores ausentes de acordo com as regras de negócios definidos anteriormente.

```

Cmd 19

1 # Verificando se os valores coringas foram tratados adequadamente
2 df_comercial.loc[df_comercial['dt_inicio'] == 100000]

ordem_servico seq_servico cod_servico cod_veiculo cod_equipe colabor_1 colabor_2 cod_susp cod_imedido dt_saida dt_inicio dt_fim dt_termino cod_local sub_regiao km

0 comando demorou 0,06 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:29:37 em Leandro Portela da Luz's Cluster

```

Para a coluna "dt_saida", utilizou-se a mesma regra anterior, porém em vez de utilizar 30 minutos, para este caso foi utilizado 15 minutos, pois este é um tempo médio padrão observado na maioria das ordens de serviços.

```

Cmd 13

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

0 comando demorou 0,19 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:36:04 em Leandro Portela da Luz's Cluster

```

Veja que após a tratativa os valores coringa não se apresentam mais na coluna dt_saida, foram sanados os valores ausentes de acordo com as regras de negócios definidos anteriormente.

```
Cmd 18
1 # Verificando se os valores coringas foram tratados adequadamente
2 df_comercial.loc[df_comercial['dt_saida'] == 100000]

ordem_servico seq_servico cod_servico cod_veiculo cod_equipe colab_1 colab_2 cod_susp cod_impedido dt_saida dt_inicio dt_fim dt_termino cod_local sub_regiao km

0 comando demorou 0,09 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:11:59 em Leandro Portela da Luz's Cluster
```

Quanto ao atributo "cod_equipe", conhecendo a regra de negócio, temos o atributo "cod_veiculo" que também identifica a equipe executora das ordens de serviços. Para tratar este caso, neste dataset, foi utilizado a técnica de substituir o valor coringa pela sigla "NOT_INF", já informado a falha ao DBA da corporação, o que vai facilitar identificar as falhas e prever as ações futuras e encontrar a solução do problema.

```
Cmd 19
1 # Tratativa dos valores coringas no atributo 'cod_equipe' conforme segue
2 df_comercial.loc[df_comercial['cod_equipe'] == 100000, 'cod_equipe'] = 'NOT_INF'

0 comando demorou 0,26 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:11:59 em Leandro Portela da Luz's Cluster
```

Em relação ao atributo "sub_regiao" verificou-se que são 4 casos apenas com valores ausentes. Diante disso, para um universo de mais de 2 milhões de registros, é insignificante aos análises e cálculos, ou seja, optou-se por eliminar esses registros do datasets.

```
Cmd 20
1 # Devido a ser somente 4 registro para um universo de 2130331 milhões de registro esta supressão não impactara no resultado final do projeto
2 df_comercial=df_comercial.drop(df_comercial[df_comercial['sub_regiao'] == 100000].index)

0 comando demorou 0,59 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:11:59 em Leandro Portela da Luz's Cluster
```

Outro problema encontrado nos análises, foi a identificação que o código "0" nos atributos "cod_susp" e "cod_impedido" do dataset "tb_comercial" , era comum para a conclusão de serviços executados pelas equipes. Durante a análise, verificou-se que pelo atributo "cod_servico" era possível separar a utilização do mesmo código e substitui-lo por outro diferente no atributo "cod_susp" para dar proseguimento as análise, conforme segue.

```
Cmd 22
1 # Substituir os valores "0" na coluna 'cod_susp' pelo valor "10000" para diferenciar do valor '0' na coluna "cod_impedido" com mesmo número e
2 # conflito na comparação de conclusão
3 #dos serviços comerciais executados e cortes executados.
4 df_comercial.loc[(df_comercial['cod_servico'] == 401) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
5 df_comercial.loc[(df_comercial['cod_servico'] == 402) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
6 df_comercial.loc[(df_comercial['cod_servico'] == 404) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
7 df_comercial.loc[(df_comercial['cod_servico'] == 405) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
8 df_comercial.loc[(df_comercial['cod_servico'] == 406) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
9 df_comercial.loc[(df_comercial['cod_servico'] == 407) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
10 df_comercial.loc[(df_comercial['cod_servico'] == 412) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
11 df_comercial.loc[(df_comercial['cod_servico'] == 417) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
12 df_comercial.loc[(df_comercial['cod_servico'] == 419) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
13 df_comercial.loc[(df_comercial['cod_servico'] == 420) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
14 df_comercial.loc[(df_comercial['cod_servico'] == 430) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
15 df_comercial.loc[(df_comercial['cod_servico'] == 440) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
16 df_comercial.loc[(df_comercial['cod_servico'] == 441) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
17 df_comercial.loc[(df_comercial['cod_servico'] == 442) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
18 df_comercial.loc[(df_comercial['cod_servico'] == 443) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
19 df_comercial.loc[(df_comercial['cod_servico'] == 449) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000
20 df_comercial.loc[(df_comercial['cod_servico'] == 499) & (df_comercial['cod_susp'] == 0), 'cod_susp'] = 10000

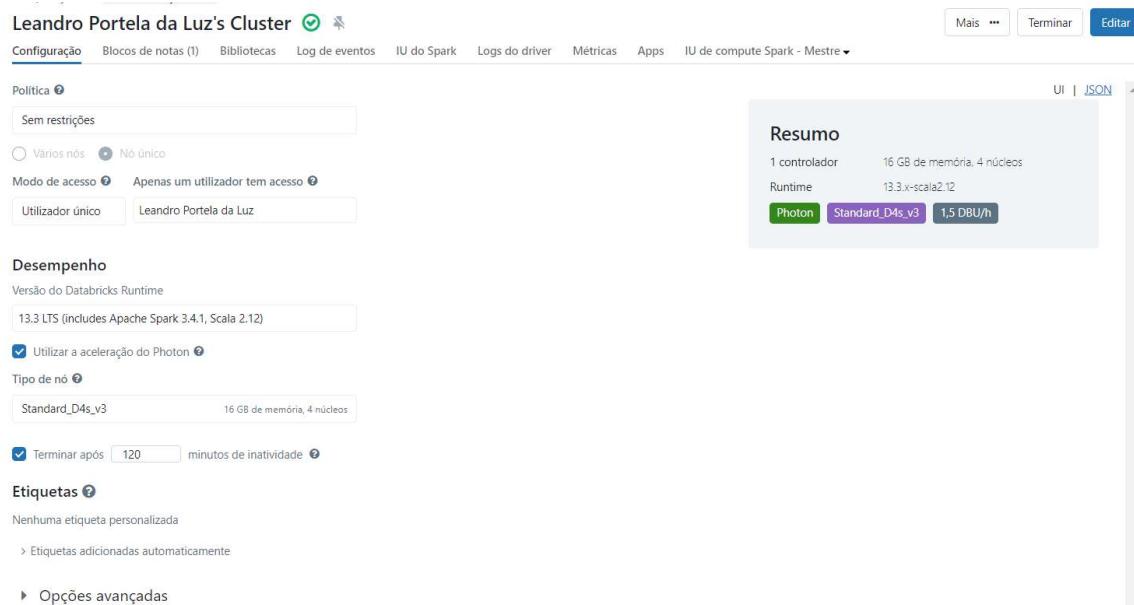
0 comando demorou 0,19 segundos - executado por leoportelaluz@hotmail.com a 01/10/2023, 21:11:59 em Leandro Portela da Luz's Cluster
```

Na tratativa foi substituído o código "0" pelo valor "10000" e alterado também na tabela dimensão "tb_corte_impedido" no atributo "cod" para que não ocorresse percar de informações, somente tratativas dos registros, ou seja, dos dados.

Cabe ressaltar, que para as outros datasets foram realizadas as tratativas necessárias, utilizando as mesmas técnicas descritas acima e foi possível zerar as falhas encontradas no datasets.

Após a realização da preparação dos dados, foi realizada a carga no Power BI, conforme segue, para os cálculos estatísticos e apresentação dos resultados em dashboard. A conexão da nuvem, utilizando a plataforma databricks azure da microsoft com o Power BI, foi realizada conforme os passos a seguir.

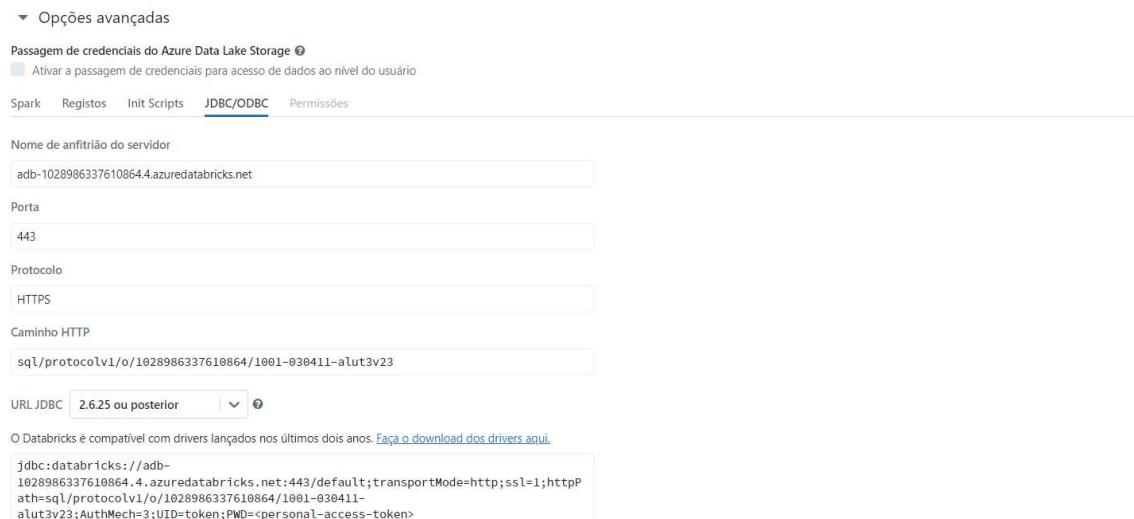
Coneção com o Cluster criado na plataforma databricks



The screenshot shows the Databricks Cluster configuration interface for a cluster named 'Leandro Portela da Luz's Cluster'. The 'Resumo' (Summary) section indicates 1 controlador, 16 GB de memória, 4 núcleos, Runtime 13.3.x-scala2.12, and two nodes: Photon and Standard_D4s_v3, both with 1.5 DBU/h. The 'Configuração' tab is selected, showing the following settings:

- Política**: Nô único (Unique node).
- Modo de acesso**: Apenas um utilizador tem acesso (Only one user has access).
- Utilizador único**: Leandro Portela da Luz.
- Desempenho**: Versão do Databricks Runtime 13.3 LTS (includes Apache Spark 3.4.1, Scala 2.12). Utilizar a aceleração do Photon (Use Photon acceleration) is checked.
- Tipo de nó**: Standard_D4s_v3 (16 GB de memória, 4 núcleos).
- Terminar após**: 120 minutos de inatividade.
- Etiquetas**: Nenhuma etiqueta personalizada.
- Opções avançadas**: Opção para Passagem de credenciais do Azure Data Lake Storage.

Após o carregamento dos datasets e as tratativas de ETL aos mesmo, chegou a hora de realizar a conexão ao Power BI, para a criação dos dashboard. Para isso acessar ao cluster, em opção avançadas, na aba dos conectores JDBC/ODBC, verificar o servidor e caminha da página HTTP.

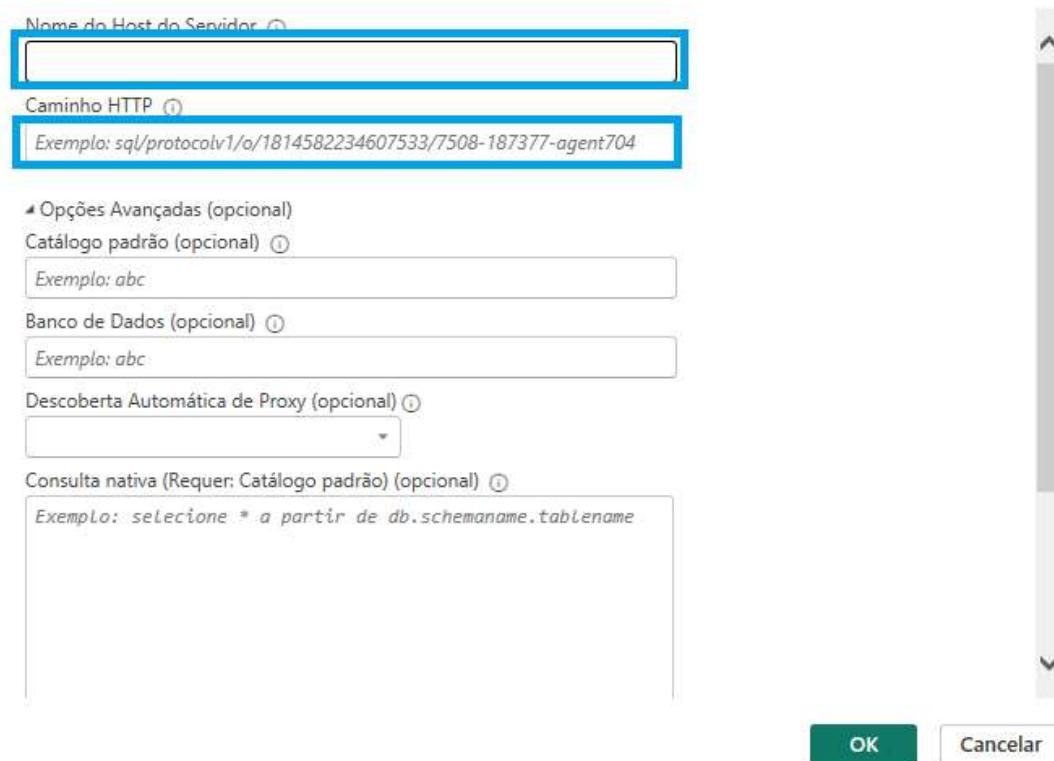


The screenshot shows the JDBC/ODBC configuration interface for a Databricks cluster. The 'JDBC/ODBC' tab is selected. The configuration includes:

- Nome de anfitrião do servidor**: adb-1028986337610864.azuredatabricks.net
- Porta**: 443
- Protocolo**: HTTPS
- Caminho HTTP**: sql/protocolv1/o/1028986337610864/1001-030411-alut3v23
- URL JDBC**: 2.6.25 ou posterior
- Nota**: O Databricks é compatível com drivers lançados nos últimos dois anos. [Faça o download dos drivers aqui.](#)
- Código de conexão** (copiado): `jdbc:databricks://adb-1028986337610864.azuredatabricks.net:443/default;transportMode=http;ssl=1;httpPath=/protocolv1/o/1028986337610864/1001-030411-alut3v23;AuthMech=3;UID=token;PWD=<personal-access-token>`

Para realizar a conexão ao Power BI, acessar ao software no desktop e realizar a conexão com a plataforma databricks através do acesso a obter dados no Power BI, na opção azure databricks.

Azure Databricks



Na sequencia informar os dados do cluster da plataforma databricks azure, conforme coletado no acesso a plataforma como já mencionado acima.

Na sequencia é necessário logar com sua conta na nuvem para realizar a conexão, e após isso conectar aos datasets e realizar a preparação e criação dos dashboard para posterior publicar na nuvem para acesso de todos.

Navegador

5. Análise dos resultados

5.1. Qualidade dos dados

Em relação aos dados, os datasets foram analisados individualmente, todos os atributos foram verificados e tratados conforme as técnicas aprendidas no curso e outras áreas. Os problemas encontrados, já foram descritos anteriormente, bem como as soluções adotadas.

A seguir segue a tela do Power Query no Power BI onde é possível verificar que os dados não apresentam nenhum problema após a realização do ETL e técnicas aplicadas na nuvem no databricks.

	dt_descrição	cod_serviço	cod_veículo	cod_agente	colab_1	colab_2	Conclusão	dt_saida	dt_início	dt_fim	dt_feriado	cod_local	sub_região
• Válidos	100%	• Válidos	100%	• Válidos	100%	• Válidos	100%	• Válidos	100%	• Válidos	100%	• Válidos	100%
• Vazio	0%	• Vazio	0%	• Vazio	0%	• Vazio	0%	• Vazio	0%	• Vazio	0%	• Vazio	0%
1	9300752323824640.35	129 E2136	CB625	980187	800307 0	0	0	01/01/2023 00:22:00	01/01/2023 05:57:00	01/01/2023 06:53:00	01/01/2023 06:52:48	7624 SUL 0	
2	9300752323824640.35	9921 E3970	MA649	972185	992187 0001	0	0	01/01/2023 07:39:00	01/01/2023 08:00:00	01/01/2023 08:29:00	01/01/2023 08:29:00	4402 MDR 0	
3	9300752323824640.35	9947 E3972	MA649	980187	800307 0	0	0	01/01/2023 08:00:00	01/01/2023 08:59:00	01/01/2023 09:00:00	01/01/2023 09:00:00	4210 UNA 0	
4	9300752323824640.35	3041 E20118	LO577	232079	77712	0	0	01/01/2023 09:58:00	01/01/2023 10:00:00	01/01/2023 10:00:00	01/01/2023 10:00:00	5428 MDT 0	
5	9300751513890408.35	9947 E3923	LO511	909414	800305 0	0	0	01/01/2023 10:00:00	01/01/2023 10:00:00	01/01/2023 10:00:00	01/01/2023 10:00:00	4150 UNA 3	
6	930075084675958672.70	332 E3127	CB531	973895	973896 0	0	0	01/01/2023 09:41:00	01/01/2023 10:00:00	01/01/2023 10:00:00	01/01/2023 10:00:00	2310 CTC 1	
7	930075084675958672.70	9947 E3923	LO511	909414	800305 0	0	0	01/01/2023 09:53:00	01/01/2023 10:00:00	01/01/2023 10:00:00	01/01/2023 10:00:00	4150 UNA 1	
8	930075084675958672.70	9947 E3923	LO511	909414	800305 0	0	0	01/01/2023 09:53:00	01/01/2023 10:00:00	01/01/2023 10:00:00	01/01/2023 10:00:00	4150 UNA 3	
9	930075084675958672.70	331 E3127	CB513	973895	973896 0	0	0	01/01/2023 09:41:00	01/01/2023 10:00:00	01/01/2023 10:00:00	01/01/2023 10:00:00	2310 CTC 1	
10	930075084675958672.70	332 E3127	CB513	973895	973896 0	0	0	01/01/2023 09:41:00	01/01/2023 10:00:00	01/01/2023 10:00:00	01/01/2023 10:00:00	2310 CTC 1	
11	930075084675958672.70	330 E3121	GA411	810298	810305 0	0	0	01/01/2023 11:31:00	01/01/2023 11:46:00	01/01/2023 12:00:00	01/01/2023 12:00:00	3278 FOZ 4	
12	930075084675958672.70	9947 E3923	LO511	909414	800305 0	0	0	01/01/2023 11:46:00	01/01/2023 12:00:00	01/01/2023 12:00:00	01/01/2023 12:00:00	4150 UNA 0	
13	930075084675958672.70	330 E3126	MA697	980187	980187 0001	0	0	01/01/2023 12:00:00	01/01/2023 12:20:00	01/01/2023 12:20:00	01/01/2023 12:20:00	6892 SOR 0	
14	930075084675958672.70	9947 E3923	LO511	909414	800305 0	0	0	01/01/2023 12:00:00	01/01/2023 12:20:00	01/01/2023 12:20:00	01/01/2023 12:20:00	6892 SOR 0	
15	930075084675958672.70	9947 E3923	LO511	909414	800305 0	0	0	01/01/2023 12:20:00	01/01/2023 12:20:00	01/01/2023 12:20:00	01/01/2023 12:20:00	4150 UNA 3	
16	930075084675958672.70	9947 E3938	MA697	985941	985944 0	0	0	01/01/2023 12:20:00	01/01/2023 12:40:00	01/01/2023 12:40:00	01/01/2023 12:40:00	6892 SOR 3	
17	930075084675958672.70	3404 V30006	CB529	124623	143865 0	0	0	01/01/2023 12:20:00	01/01/2023 12:40:00	01/01/2023 12:40:00	01/01/2023 12:40:00	5088 AGA 0	
18	930075084675958672.70	330 E3126	MA697	985941	985944 0	0	0	01/01/2023 12:20:00	01/01/2023 12:40:00	01/01/2023 12:40:00	01/01/2023 12:40:00	6892 SOR 3	
19	930075084675958672.70	9947 E3923	LO511	909414	800305 0	0	0	01/01/2023 12:20:00	01/01/2023 12:40:00	01/01/2023 12:40:00	01/01/2023 12:40:00	3278 FOZ 1	
20	930075084675958672.70	9921 E3709	MA649	980187	980187 0001	0	0	01/01/2023 12:41:00	01/01/2023 13:31:00	01/01/2023 13:49:00	01/01/2023 13:49:00	4402 MDR 0	
21	9300649350861728.70	331 E3109	MA649	985949	985949 8018	0	0	01/01/2023 14:15:00	01/01/2023 14:31:00	01/01/2023 14:33:00	01/01/2023 14:33:00	4402 MDR 0	
22	9300649350861728.70	332 E3109	MA649	985949	985949 8001	0	0	01/01/2023 14:35:00	01/01/2023 14:47:00	01/01/2023 14:49:00	01/01/2023 14:49:00	4402 MDR 0	
23	9300736730050001864.35	5902 E336	MA697	985941	985944 0	0	0	01/01/2023 14:39:00	01/01/2023 14:42:00	01/01/2023 14:42:00	01/01/2023 14:42:00	6892 SOR 3	
24	9300752323824640.35	9947 E3923	LO511	909414	800305 0	0	0	01/01/2023 14:37:00	01/01/2023 14:44:00	01/01/2023 14:44:00	01/01/2023 14:44:00	4150 UNA 3	

5.2. Solução do problema

Neste trabalho buscou-se a solução de um problema real, dos principais motivos de impedimentos de serviços comerciais e emergenciais na corporação.

Diante disso, foram elencados os seguintes objetivos conforme segue:

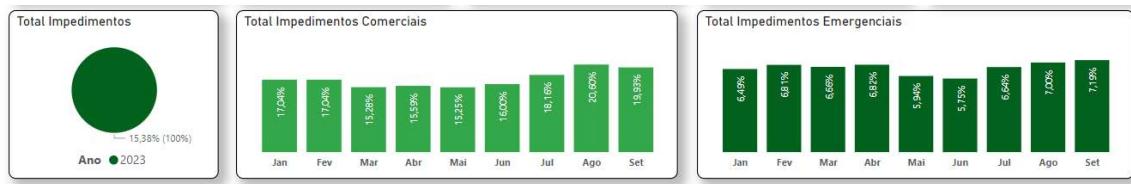
- Identificar as regiões com maior índices de impedimentos de serviços comerciais e emergenciais;

Na análise dos dados, aplicando técnicas básicas de estatísticas, foi possível mapear por departamento e regiões os índices de impedimentos dos serviços comerciais e emergenciais. Verificou-se que em relação aos departamentos de serviços existe uma pequena diferença, quanto as proporções nos impedimentos. Já quanto as regiões, percebe-se que fica mais evidente este gap, com valores mais espaçados entre as áreas. Cabe ressaltar, o período considerado para análise foi do ano corrente de 2023. Na imagem abaixo foi carregado somente as regiões TOP FIVE em impedimento no período analisado.



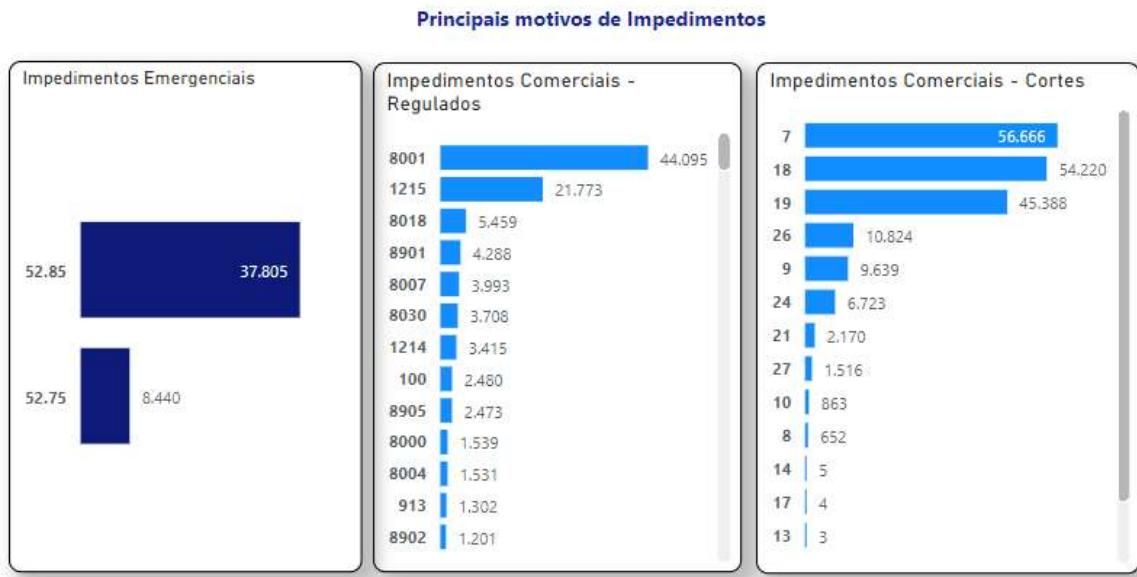
- Verificar se existe alguma sazonalidade, quanto aos impedimentos de serviços comerciais e emergenciais;

Verificou-se que a sazonalidade, quanto ao impedimento dos serviços comerciais e emergenciais, durante os meses do ano, segue uma tendência continua até o momento.



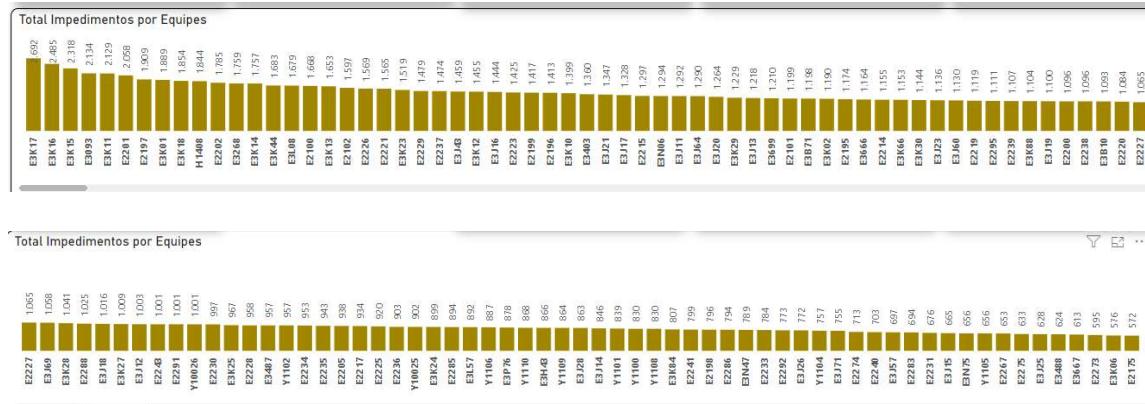
- Elencar os principais motivos de impedimentos de serviços comerciais e emergenciais;

Percebe-se que existe alguns motivos bem destacados, quanto aos impedimentos de serviços comerciais e emergenciais. Diante disso, será possível aprofundar os estudos para identificar as causas raízes e propor ações de melhorias nos processos.



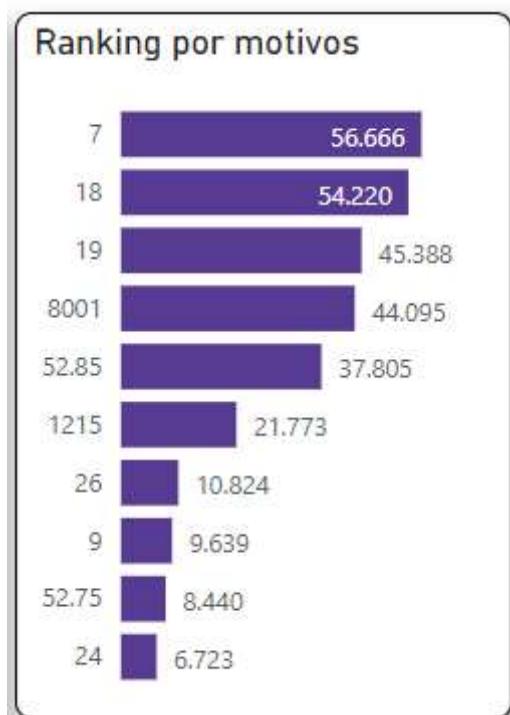
- Comparar as equipes executoras, analisando se a proporção é igualitária de impedimentos ou existe descrença;

Ao analisar as equipes pelas quantidades, percebe-se que existe uma proporção com certa descrença entre as mesmas, conforme o gráfico abaixo. A partir de agora, será aprofundado os análises, verificando as regiões de pertencimento de cada equipe, áreas urbanas e rurais, para identificar os problemas e corrigi-los.



- Criar um ranking "top five" entre os motivos de reprovações;

Realizado a construção de dashboard com o ranking com os TOP TEN, um pouco acima dos objetivos, dos principais motivos de impedimento de serviços comerciais e emergenciais.



- Criar um ranking "top ten" das equipes com os maiores índices de impedimentos.

Realizado a construção de dashboard com o ranking com os TOP TEN, em relação as equipes executoras para identificar as causas do grande volume de impedimento de serviços comerciais e emergenciais, por elas.

Quanto as dificuldades na realização deste trabalho, se deu por optar por um caso real, diante da liberação de acessos e os dados. Após liberação foi necessário a tratativas e descaracterização dos mesmo para eveitar problemas com a legislação e as regras de negócio da corporação. Acredito que a maior dificuldade esta sendo na execução do projeto, haja visto que precisa de técnicas apuradas na área de ciências de dados, e ainda estou começando nesta área. Porém aos poucos vamos evoluindo e desenvolvendo os conhecimentos. Além disso, conciliar os estudos com o trabalho, afeta também no desempenho e conclusão dos trabalhos, entretanto foi possível a conciliação.

5.2. Conclusão

Este projeto buscou analisar os dados dos serviços comerciais e emergenciais executados no ano de 2023, para entender os seus motivos, verificar os índices em cada área, as atuações das equipes e identificar oportunidades de melhorias.

Os próximos passos, será aprofundar os motivos com maior índices, equipes com elevadas taxas de impedimentos e entender as causas raízes, Num segundo momento, será aprofundado o estudo buscando os valores médios de cada serviços para entender os desperdícios e oportunidades de ganhos para a corporação, quanto a diminuição dos índices de impedimentos nos serviços comerciais e emergenciais.

In []:

In []: