

# Nota Técnica: Processo de Transformação de Dados de Leitos do CNES

## Comparativo: Arquivo Original vs. Arquivo Tratado

Cieges - Brasil Estadual

2026-01-21

### Table of contents

1	Resumo Executivo	1
2	Descrição dos Arquivos	1
2.1	Arquivo Original (arq1_original.csv)	1
2.2	Arquivo Tratado (arq2_tratado.csv)	2
3	Transformações Realizadas	2
3.1	Etapa 1: Filtro de Competência	2
3.2	Etapa 2: Remoção de Colunas	2
3.3	Etapa 3: Enriquecimento de Dados (JOINS)	3
3.3.1	Mapeamento tp_leito → DS_TP_LEITO	3
3.3.2	Mapeamento co_leito → DS_CO_LEITO	3
3.4	Etapa 4: Limpeza de Dados	3
3.4.1	Remoção do Código de Leito 66	3
3.4.2	Remoção de 5 Estabelecimentos (CNES)	3
4	Validação de Totais	4
5	Estatísticas do Arquivo Tratado	4
5.1	Distribuição por Tipo de Leito	4
5.2	Top 10 Especialidades (por quantidade de leitos)	4
5.3	Distribuição SUS vs Não-SUS	4
5.4	Abrangência Geográfica	5
6	Fluxo de Processamento (PINTI)	5
7	Considerações para Tipologia Derivada	5

## 1 Resumo Executivo

Este documento descreve o processo de ETL (Extract, Transform, Load) aplicado aos dados de leitos hospitalares do CNES (Cadastro Nacional de Estabelecimentos de Saúde), detalhando as transformações realizadas entre o arquivo original (arq1\_original.csv) e o arquivo tratado (arq2\_tratado.csv).

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

# Carregar dados
df1 = pd.read_csv('arq1_original.csv', sep=';', encoding='latin1', low_memory=False)
df2 = pd.read_csv('arq2_tratado.csv', sep=';', encoding='latin1', low_memory=False)

# Resumo comparativo
resumo = pd.DataFrame({
    'Métrica': ['Registros', 'Colunas', 'Tamanho (MB)', 'Valores Nulos'],
    'Original': [f'{len(df1):,}', len(df1.columns), '~30', 'Muitos'],
    'Tratado': [f'{len(df2):,}', len(df2.columns), '~3', 'Zero'],
    'Variação': ['-83,9%', '-63,3%', '-90%', '-100%']
})
resumo
```

	Métrica	Original	Tratado	Variação
0	Registros	309,610	49,804	-83,9%
1	Colunas	30	11	-63,3%
2	Tamanho (MB)	~30	~3	-90%
3	Valores Nulos	Muitos	Zero	-100%

## 2 Descrição dos Arquivos

### 2.1 Arquivo Original (arq1\_original.csv)

Características:

- **Registros:** 309.610
- **Colunas:** 30
- **Período:** Janeiro a Junho de 2025 (202501 a 202506)

```
# Estrutura do arquivo original
print(f"Colunas do arquivo original ({len(df1.columns)}):")
for i, col in enumerate(df1.columns, 1):
    nulos = df1[col].isnull().sum()
    pct = (nulos / len(df1)) * 100
    print(f" {i:2}. {col:<15} - Nulos: {pct:.1f}%")
```

Colunas do arquivo original (30):

1.	i»¿D_R	- Nulos: 100.0%
2.	cnes	- Nulos: 0.0%
3.	codufmun	- Nulos: 0.0%
4.	regsaude	- Nulos: 29.8%

5. micr_reg	- Nulos: 100.0%
6. distrsan	- Nulos: 91.8%
7. distradm	- Nulos: 100.0%
8. tpgestao	- Nulos: 0.0%
9. pf_pj	- Nulos: 0.0%
10. cpf_cnpj	- Nulos: 0.0%
11. niv_dep	- Nulos: 0.0%
12. cnpj_man	- Nulos: 0.0%
13. esfera_a	- Nulos: 100.0%
14. atividade	- Nulos: 0.0%
15. retencao	- Nulos: 100.0%
16. natureza	- Nulos: 100.0%
17. clientel	- Nulos: 0.3%
18. tp_unid	- Nulos: 0.0%
19. turno_at	- Nulos: 0.0%
20. niv_hier	- Nulos: 100.0%
21. terceiro	- Nulos: 100.0%
22. tp_leito	- Nulos: 0.0%
23. codleito	- Nulos: 0.0%
24. qt_exist	- Nulos: 0.0%
25. qt_contr	- Nulos: 0.0%
26. qt_sus	- Nulos: 0.0%
27. qt_nsus	- Nulos: 0.0%
28. competen	- Nulos: 0.0%
29. nat_jur	- Nulos: 0.0%
30. File Paths	- Nulos: 0.0%

2.2 Arquivo Tratado (arq2\_tratado.csv)

Características:

- **Registros:** 49.804
- **Colunas:** 11
- **Período:** Apenas Junho de 2025 (202506)
- **Qualidade:** Zero valores nulos

```
# Estrutura do arquivo tratado
print(f"Colunas do arquivo tratado ({len(df2.columns)}):")
for i, col in enumerate(df2.columns, 1):
    print(f"  {i:2}. {col}")
```

Colunas do arquivo tratado (11):

1. i»\_competen
2. codufmun
3. cnes
4. tp\_leito
5. DS\_TP\_LEITO
6. co\_leito
7. DS\_CO\_LEITO
8. qt\_exist
9. qt\_contr
10. qt\_sus
11. qt\_nsus

3 Transformações Realizadas

3.1 Etapa 1: Filtro de Competência

```
print("Competências no arquivo original:")
print(sorted(df1['competen'].unique()))

print(f"\nCompetência no arquivo tratado:")
print(sorted(df2.iloc[:,0].unique()))

print(f"\nRedução: {len(df1):,} → {len(df1[df1['competen']==202506]):,} registros")
```

Competências no arquivo original:  
[202501, 202502, 202503, 202504, 202505, 202506]

Competência no arquivo tratado:  
[202506]

Redução: 309,610 → 51,776 registros

3.2 Etapa 2: Remoção de Colunas

19 colunas removidas:

```
cols1_norm = [c.replace('i»_', '').lower() for c in df1.columns]
cols2_norm = [c.replace('i»_', '').lower() for c in df2.columns]

removidas = [c for c in df1.columns if c.replace('i»_', '').lower() not in cols2_norm]
print("Colunas removidas:")
for c in removidas:
    print(f"  - {c}")
```

Colunas removidas:

- i»\_D\_R
- regsaude
- micr\_reg
- distrsan
- distradm
- tpgestao
- pf\_pj

- cpf\_cnpj
- niv\_dep
- cnpj\_man
- esfera\_a
- atividade
- retencao
- natureza
- clientel
- tp\_unid
- turno\_at
- niv\_hier
- terceiro
- codleito
- nat\_jur
- File Paths

### 3.3 Etapa 3: Enriquecimento de Dados (JOINS)

#### 3.3.1 Mapeamento tp\_leito → DS\_TP\_LEITO

```
mapa_tp = df2[['tp_leito', 'DS_TP_LEITO']].drop_duplicates().sort_values('tp_leito')
mapa_tp
```

	tp_leito	DS_TP_LEITO
0	1	CIRURGICO
6	2	CLINICO
2	3	COMPLEMENTAR
12	4	OBSTERICO
1	5	PEDIATRICO
49	6	OUTRAS ESPECIALIDADES
22	7	HOSPITAL DIA

#### 3.3.2 Mapeamento co\_leito → DS\_CO\_LEITO

```
mapa_co = df2[['co_leito', 'DS_CO_LEITO']].drop_duplicates().sort_values('co_leito')
print(f"Total de códigos de leito mapeados: {len(mapa_co)}")
mapa_co.head(15)
```

Total de códigos de leito mapeados: 65

	co_leito	DS_CO_LEITO
361	1	BUCO MAXILO FACIAL
24	2	CARDIOLOGIA
5	3	CIRURGIA GERAL
364	4	ENDOCRINOLOGIA
365	5	GASTROENTEROLOGIA
70	6	GINECOLOGIA
22	7	CIRURGICO/DIAGNOSTICO/TERAPEUTICO
367	8	NEFROLOGIAUROLOGIA
46	9	NEUROCIRURGIA
12	10	OBSTETRICIA CIRURGICA
9	11	OFTALMOLOGIA
369	12	ONCOLOGIA
0	13	ORTOPEDIATRAUMATOLOGIA
371	14	OTORRINOLARINGOLOGIA
372	15	PLASTICA

### 3.4 Etapa 4: Limpeza de Dados

#### 3.4.1 Remoção do Código de Leito 66

```
df1_202506 = df1[df1['competen'] == 202506]
registros_66 = df1_202506[df1_202506['codleito'] == 66]

print(f"Registros com codleito = 66 removidos: {len(registros_66):,}")
print(f"Leitos removidos: {registros_66['qt_exist'].sum():,}")
print(f"Tipo de leito: 3 (COMPLEMENTAR)")
```

Registros com codleito = 66 removidos: 1,972  
Leitos removidos: 5,450  
Tipo de leito: 3 (COMPLEMENTAR)

#### 3.4.2 Remoção de 5 Estabelecimentos (CNES)

```
cnes_orig = set(df1_202506['cnes'].unique())
cnes_trat = set(df2['cnes'].unique())
cnes_removidos = list(cnes_orig - cnes_trat)

print("CNES removidos:")
for cnes in cnes_removidos:
    regs = df1_202506[df1_202506['cnes'] == cnes]
    print(f" CNES {cnes}: {len(regs)} registro(s), {regs['qt_exist'].sum()} leito(s)")
```

CNES removidos:  
CNES 2062054: 1 registro(s), 1 leito(s)  
CNES 7399626: 1 registro(s), 7 leito(s)  
CNES 7065299: 1 registro(s), 2 leito(s)  
CNES 198552: 1 registro(s), 3 leito(s)  
CNES 5883229: 1 registro(s), 2 leito(s)

## 4 Validação de Totais

```
validacao = pd.DataFrame({
    'Indicador': ['Registros', 'qt_exist', 'qt_sus', 'qt_nsus'],
    'Original (202506)': [
        f"{len(df1_202506):,}",
        f"{df1_202506['qt_exist'].sum():,}",
        f"{df1_202506['qt_sus'].sum():,}",
        f"{df1_202506['qt_nsus'].sum():,}"
    ],
    'Tratado': [
        f"{len(df2):,}",
        f"{df2['qt_exist'].sum():,}",
        f"{df2['qt_sus'].sum():,}",
        f"{df2['qt_nsus'].sum():,}"
    ],
    'Diferença': [
        f"{len(df1_202506) - len(df2):,}",
        f"{df1_202506['qt_exist'].sum() - df2['qt_exist'].sum():,}",
        f"{df1_202506['qt_sus'].sum() - df2['qt_sus'].sum():,}",
        f"{df1_202506['qt_nsus'].sum() - df2['qt_nsus'].sum():,}"
    ]
})
validacao
```

	Indicador	Original (202506)	Tratado	Diferença
0	Registros	51,776	49,804	1,972
1	qt_exist	540,583	535,133	5,450
2	qt_sus	359,171	355,179	3,992
3	qt_nsus	181,412	179,954	1,458

## 5 Estatísticas do Arquivo Tratado

### 5.1 Distribuição por Tipo de Leito

```
tipo_leito = df2.groupby('DS_TP_LEITO').agg({
    'cnes': 'count',
    'qt_exist': 'sum'
}).rename(columns={'cnes': 'Registros', 'qt_exist': 'Leitos'})
tipo_leito['% Leitos'] = (tipo_leito['Leitos'] / tipo_leito['Leitos'].sum() * 100).round(1)
tipo_leito = tipo_leito.sort_values('Leitos', ascending=False)
tipo_leito
```

DS_TP_LEITO	Registros	Leitos	% Leitos
CLINICO	12001	176667	33.0
CIRURGICO	15062	123582	23.1
COMPLEMENTAR	5910	77311	14.4
OBSTERICO	6670	50095	9.4
PEDIATRICO	5934	46609	8.7
OUTRAS ESPECIALIDADES	2402	46267	8.6
HOSPITAL DIA	1825	14602	2.7

### 5.2 Top 10 Especialidades (por quantidade de leitos)

```
especialidades = df2.groupby('DS_CO_LEITO')['qt_exist'].sum().sort_values(ascending=False).head(10)
especialidades_df = pd.DataFrame({
    'Especialidade': especialidades.index,
    'Leitos': especialidades.values
})
especialidades_df
```

	Especialidade	Leitos
0	CLINICA GERAL	138290
1	CIRURGIA GERAL	63475
2	PEDIATRIA CLINICA	40643
3	UTI ADULTO - TIPO II	33519
4	PSQUIATRIA	28734
5	OBSTETRICA CIRURGICA	25058
6	OBSTETRICA CLINICA	25037
7	ORTOPEDIATRAUMATOLOGIA	20118
8	CARDIOLOGIA	13119
9	ONCOLOGIA	12583

### 5.3 Distribuição SUS vs Não-SUS

```
sus = df2['qt_sus'].sum()
nsus = df2['qt_nsus'].sum()
total = sus + nsus

distribuicao_sus = pd.DataFrame({
    'Categoria': ['SUS', 'Não-SUS', 'Total'],
    'Leitos': [f"{sus:,}", f"{nsus:,}", f"{total:,}"],
    '%': [f"{sus/total*100:.1f}%", f"{nsus/total*100:.1f}%", "100%"]
})
distribuicao_sus
```

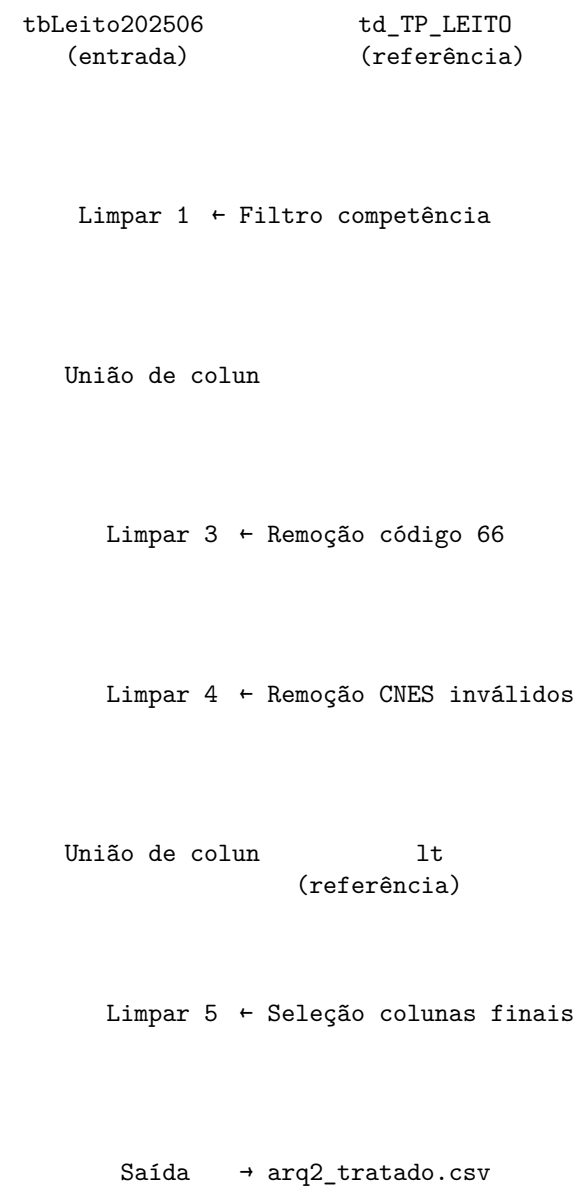
	Categoria	Leitos	%
0	SUS	355,179	66.4%
1	Não-SUS	179,954	33.6%
2	Total	535,133	100%

5.4 Abrangência Geográfica

```
print(f"Estabelecimentos (CNES): {df2['cnes'].nunique():,}")
print(f"Municípios: {df2['codufmun'].nunique():,}")
```

Estabelecimentos (CNES): 9,072  
Municípios: 3,597

6 Fluxo de Processamento (PINTI)



7 Considerações para Tipologia Derivada

Com base na análise realizada, os dados tratados estão prontos para a criação de uma tipologia derivada utilizando:

Variáveis disponíveis:

- 1. tp\_leito / DS\_TP\_LEITO - 7 categorias
- 2. co\_leito / DS\_CO\_LEITO - 65 categorias

Possíveis abordagens:

- Agrupamento hierárquico (Tipo → Especialidade)
- Clusterização por perfil de estabelecimento
- Tipologia por complexidade (UTI, Cirúrgico, Clínico)
- Tipologia por público-alvo (Adulto, Pediátrico, Obstétrico)

Elaborado por: Cascade AI  
Ferramenta ETL: Pinti  
Destino: Tableau (LT\_BR\_2501-2510.hyper)