# Análise de dados - Enem 2021

## Apresentação

O Exame Nacional do Ensino Médio (Enem) foi instituído em 1998, com o objetivo de avaliar o desempenho escolar dos estudantes ao término da educação básica. Em 2009, o exame aperfeiçoou sua metodologia e passou a ser utilizado como mecanismo de acesso à educação superior. Desde 2020, o participante pode escolher entre fazer o exame impresso ou o Enem Digital, com provas aplicadas em computadores, em locais de prova definidos pelo Inep.

As notas do Enem podem ser usadas para acesso ao Sistema de Seleção Unificada (Sisu) e ao Programa Universidade para Todos (ProUni). Elas também são aceitas em mais de 50 instituições de educação superior portuguesas. Além disso, os participantes do Enem podem pleitear financiamento estudantil em programas do governo, como o Fundo de Financiamento Estudantil (Fies). Os resultados do Enem possibilitam, ainda, o desenvolvimento de estudos e indicadores educacionais.

Qualquer pessoa que já concluiu o ensino médio ou está concluindo a etapa pode fazer o Enem para acesso à educação superior. Os participantes que ainda não concluíram o ensino médio podem participar como “treineiros” e seus resultados no exame servem somente para autoavaliação de conhecimentos.

A aplicação do Enem ocorre em dois dias. A Política de Acessibilidade e Inclusão do Inep garante atendimento especializado e tratamento pelo nome social, além de diversos recursos de acessibilidade. Há também uma aplicação para pessoas privadas de liberdade.

Os participantes fazem provas de quatro áreas de conhecimento: linguagens, códigos e suas tecnologias; ciências humanas e suas tecnologias; ciências da natureza e suas tecnologias; e matemática e suas tecnologias, que ao todo somam 180 questões objetivas. Os participantes também são avaliados por meio de uma redação, que exige o desenvolvimento de um texto dissertativo-argumentativo a partir de uma situação-problema.

## Importando as principais bibliotecas

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Além de importar as bibliotecas, vamos ajustar o Pandas para que seja capaz de exibir todas as features do dataframe
pd.set_option("display.max_rows", 100)
```

## Coletando os dados

Os microdados de todas as provas do ENEM de 1998 até 2021 podem ser obtidos através deste link.

A partir deste link, foram baixados os dados relacionados ao ENEM 2021. Ao descompactar o arquivo baixado, o arquivo .csv com os dados usados para esta análise encontra-se dentro da pasta DADOS. Na pasta DICIONÁRIOS encontra-se um arquivo nomeado *Dicionário_Microdados_Enem_2021.xlsx* com a descrição no nome de cada uma das colunas do conjunto de dados.

```
# Os dados encontram-se no Google Drive
path = "/content/drive/MyDrive/DataScience_Datasets/MICRODADOS_ENEM_2021.csv"
df = pd.read_csv(path,sep=';',encoding='iso-8859-1')
df.head()
```

| | NU_INSCRICAO | NU_ANO | TP_FAIXA_ETARIA | TP_SEXO | TP_ESTADO_CIVIL | TP_COR_R |
|---|---|---|---|---|---|---|
| **0** | 210053865474 | 2021 | 5 | F | 1 | |
| **1** | 210052384164 | 2021 | 12 | M | 1 | |
| **2** | 210052589243 | 2021 | 13 | F | 3 | |
| **3** | 210052128335 | 2021 | 3 | M | 1 | |
| **4** | 210051353021 | 2021 | 2 | F | 1 | |

5 rows × 76 columns

## Análise exploratória dos dados

```
df.columns
```

```
Index(['NU_INSCRICAO', 'NU_ANO', 'TP_FAIXA_ETARIA', 'TP_SEXO',
       'TP_ESTADO_CIVIL', 'TP_COR_RACA', 'TP_NACIONALIDADE', 'TP_ST_CONCLUSAO',
       'TP_ANO_CONCLUIU', 'TP_ESCOLA', 'TP_ENSINO', 'IN_TREINEIRO',
       'CO_MUNICIPIO_ESC', 'NO_MUNICIPIO_ESC', 'CO_UF_ESC', 'SG_UF_ESC',
       'TP_DEPENDENCIA_ADM_ESC', 'TP_LOCALIZACAO_ESC', 'TP_SIT_FUNC_ESC',
       'CO_MUNICIPIO_PROVA', 'NO_MUNICIPIO_PROVA', 'CO_UF_PROVA',
       'SG_UF_PROVA', 'TP_PRESENCA_CN', 'TP_PRESENCA_CH', 'TP_PRESENCA_LC',
       'TP_PRESENCA_MT', 'CO_PROVA_CN', 'CO_PROVA_CH', 'CO_PROVA_LC',
       'CO_PROVA_MT', 'NU_NOTA_CN', 'NU_NOTA_CH', 'NU_NOTA_LC', 'NU_NOTA_MT',
       'TX_RESPOSTAS_CN', 'TX_RESPOSTAS_CH', 'TX_RESPOSTAS_LC',
       'TX_RESPOSTAS_MT', 'TP_LINGUA', 'TX_GABARITO_CN', 'TX_GABARITO_CH',
       'TX_GABARITO_LC', 'TX_GABARITO_MT', 'TP_STATUS_REDACAO',
       'NU_NOTA_COMP1', 'NU_NOTA_COMP2', 'NU_NOTA_COMP3', 'NU_NOTA_COMP4',
       'NU_NOTA_COMP5', 'NU_NOTA_REDACAO', 'Q001', 'Q002', 'Q003', 'Q004',
       'Q005', 'Q006', 'Q007', 'Q008', 'Q009', 'Q010', 'Q011', 'Q012', 'Q013',
       'Q014', 'Q015', 'Q016', 'Q017', 'Q018', 'Q019', 'Q020', 'Q021', 'Q022',
       'Q023', 'Q024', 'Q025'],
      dtype='object')
```

```
#Vamos checar algumas informações extras a respeito do dataset
df.info()
```

```
 20  NO_MUNICIPIO_PROVA      object
 21  CO_UF_PROVA             int64
 22  SG_UF_PROVA             object
 23  TP_PRESENCA_CN          int64
 24  TP_PRESENCA_CH          int64
 25  TP_PRESENCA_LC          int64
 26  TP_PRESENCA_MT          int64
 27  CO_PROVA_CN             float64
 28  CO_PROVA_CH             float64
 29  CO_PROVA_LC             float64
 30  CO_PROVA_MT             float64
 31  NU_NOTA_CN              float64
 32  NU_NOTA_CH              float64
 33  NU_NOTA_LC              float64
 34  NU_NOTA_MT              float64
 35  TX_RESPOSTAS_CN         object
 36  TX_RESPOSTAS_CH         object
 37  TX_RESPOSTAS_LC         object
 38  TX_RESPOSTAS_MT         object
 39  TP_LINGUA               int64
 40  TX_GABARITO_CN          object
 41  TX_GABARITO_CH          object
 42  TX_GABARITO_LC          object
 43  TX_GABARITO_MT          object
 44  TP_STATUS_REDACAO       float64
 45  NU_NOTA_COMP1           float64
 46  NU_NOTA_COMP2           float64
 47  NU_NOTA_COMP3           float64
 48  NU_NOTA_COMP4           float64
 49  NU_NOTA_COMP5           float64
 50  NU_NOTA_REDACAO         float64
 51  Q001                    object
 52  Q002                    object
 53  Q003                    object
 54  Q004                    object
 55  Q005                    float64
 56  Q006                    object
 57  Q007                    object
 58  Q008                    object
 59  Q009                    object
 60  Q010                    object
 61  Q011                    object
 62  Q012                    object
 63  Q013                    object
 64  Q014                    object
 65  Q015                    object
 66  Q016                    object
 67  Q017                    object
 68  Q018                    object
 69  Q019                    object
 70  Q020                    object
 71  Q021                    object
 72  Q022                    object
 73  Q023                    object
 74  Q024                    object
 75  Q025                    object
dtypes: float64(22), int64(17), object(37)
memory usage: 1.9+ GB
```

De acordo com o dicionário disponível junto com os dados, os dados apresentados acima, estão divididos da seguinte forma:

- DADOS DO PARTICIPANTE: Colunas `NU_INSCRICAO` até `IN_TREINEIRO`
- DADOS DA ESCOLA: Colunas `CO_MUNICIPIO_ESC` até `TP_SIT_FUNC_ESC`
- DADOS DO LOCAL DE APLICAÇÃO DA PROVA: Colunas `CO_MUNICIPIO_PROVA` até `SG_UF_PROVA`

- DADOS DA PROVA OBJETIVA: Colunas `TP_PRESENCA_CN` até `TX_GABARITO_MT`
- DADOS DA REDAÇÃO: Colunas `TP_STATUS_REDACAO` até `NU_NOTA_REDACAO`
- DADOS DO QUESTIONÁRIO SOCIOECONÔMICO: Colunas `Q001` até `Q025`.

Neste documento vamos analisar o desempenho dos participantes **presentes** em todas as provas objetivas e de redação. Além dos dados do participante, usaremos alguns dados da escola do participante, além dos dados da prova objetiva e de redação. As colunas que correspondem às características desejadas encontram-se na linha de código a seguir.

```
dados_selecionados = ['TP_FAIXA_ETARIA', 'TP_SEXO', 'TP_ESTADO_CIVIL', 'TP_COR_RACA',
                      'TP_NACIONALIDADE', 'TP_ST_CONCLUSAO', 'TP_ANO_CONCLUIU','TP_ESCOLA', 'TP_ENSINO',
                      'IN_TREINEIRO', 'NO_MUNICIPIO_ESC', 'CO_UF_ESC', 'SG_UF_ESC',
                      'TP_DEPENDENCIA_ADM_ESC', 'TP_LOCALIZACAO_ESC', 'TP_SIT_FUNC_ESC',
                      'TP_PRESENCA_CN', 'TP_PRESENCA_CH', 'TP_PRESENCA_LC',
                      'TP_PRESENCA_MT', 'NU_NOTA_CN', 'NU_NOTA_CH', 'NU_NOTA_LC', 'NU_NOTA_MT',
                      'TP_STATUS_REDACAO', 'NU_NOTA_COMP1', 'NU_NOTA_COMP2', 'NU_NOTA_COMP3',
                      'NU_NOTA_COMP4','NU_NOTA_COMP5', 'NU_NOTA_REDACAO']


#Dataframe composto apenas pelas colunas escolhidas acima
df2 = df[dados_selecionados]


df2.head()
```

| | TP_FAIXA_ETARIA | TP_SEXO | TP_ESTADO_CIVIL | TP_COR_RACA | TP_NACIONALIDADE |
|---|---|---|---|---|---|
| **0** | 5 | F | 1 | 1 | 1 |
| **1** | 12 | M | 1 | 1 | 1 |
| **2** | 13 | F | 3 | 1 | 1 |
| **3** | 3 | M | 1 | 3 | 1 |
| **4** | 2 | F | 1 | 3 | 1 |

5 rows × 31 columns

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3389832 entries, 0 to 3389831
Data columns (total 31 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   TP_FAIXA_ETARIA        int64  
 1   TP_SEXO                object 
 2   TP_ESTADO_CIVIL        int64  
 3   TP_COR_RACA            int64  
 4   TP_NACIONALIDADE       int64  
 5   TP_ST_CONCLUSAO        int64  
 6   TP_ANO_CONCLUIU        int64  
 7   TP_ESCOLA              int64  
 8   TP_ENSINO              float64
 9   IN_TREINEIRO           int64  
 10  NO_MUNICIPIO_ESC       object 
 11  CO_UF_ESC              float64
 12  SG_UF_ESC              object 
 13  TP_DEPENDENCIA_ADM_ESC  float64
 14  TP_LOCALIZACAO_ESC     float64
 15  TP_SIT_FUNC_ESC        float64
 16  TP_PRESENCA_CN         int64  
 17  TP_PRESENCA_CH         int64  
 18  TP_PRESENCA_LC         int64  
 19  TP_PRESENCA_MT         int64  
 20  NU_NOTA_CN             float64
 21  NU_NOTA_CH             float64
 22  NU_NOTA_LC             float64
 23  NU_NOTA_MT             float64
 24  TP_STATUS_REDACAO      float64
 25  NU_NOTA_COMP1          float64
 26  NU_NOTA_COMP2          float64
 27  NU_NOTA_COMP3          float64
 28  NU_NOTA_COMP4          float64
 29  NU_NOTA_COMP5          float64
 30  NU_NOTA_REDACAO        float64
dtypes: float64(16), int64(12), object(3)
memory usage: 801.7+ MB
```

Vamos ver como estão distribuídos os alunos incritos de acordo com a faixa etária. Antes, vamos usar o rótulo do dicionário para converter os números em dados.

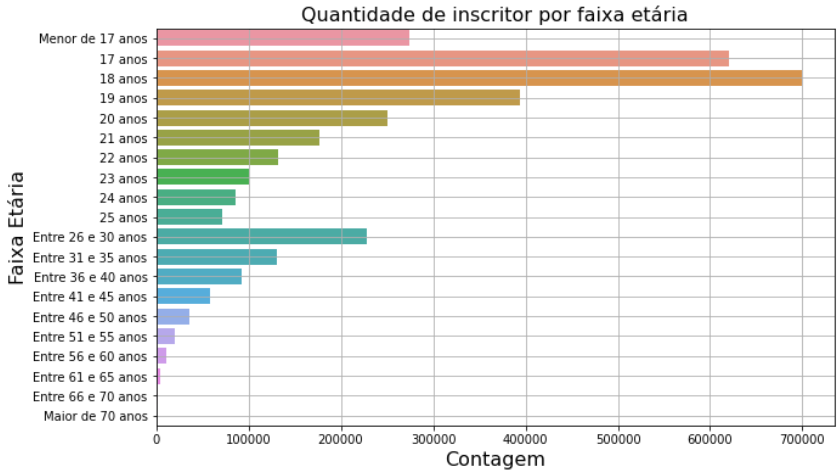
```
dict_faixa_etaria = {1:'Menor de 17 anos',
                     2:'17 anos',
                     3:'18 anos',
                     4:'19 anos',
                     5:'20 anos',
                     6:'21 anos',
                     7:'22 anos',
                     8:'23 anos',
                     9:'24 anos',
                     10:'25 anos',
                     11:'Entre 26 e 30 anos',
                     12:'Entre 31 e 35 anos',
                     13:'Entre 36 e 40 anos',
                     14:'Entre 41 e 45 anos',
                     15:'Entre 46 e 50 anos',
                     16:'Entre 51 e 55 anos',
                     17:'Entre 56 e 60 anos',
                     18:'Entre 61 e 65 anos',
                     19:'Entre 66 e 70 anos',
                     20:'Maior de 70 anos'}
df2['TP_FAIXA_ETARIA'] = df2['TP_FAIXA_ETARIA'].map(dict_faixa_etaria)
df2['TP_FAIXA_ETARIA'].head()
```

```
<ipython-input-10-5702e40372c6>:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df2['TP_FAIXA_ETARIA'] = df2['TP_FAIXA_ETARIA'].map(dict_faixa_etaria)
0              20 anos
1    Entre 31 e 35 anos
2    Entre 36 e 40 anos
3              18 anos
4              17 anos
Name: TP_FAIXA_ETARIA, dtype: object
```

```
df2['TP_FAIXA_ETARIA'].value_counts(normalize= True,sort=True)
```

```
18 anos              0.206313
17 anos              0.183131
19 anos              0.116433
Menor de 17 anos     0.080924
20 anos              0.073954
Entre 26 e 30 anos   0.067134
21 anos              0.052240
22 anos              0.038955
Entre 31 e 35 anos   0.038570
23 anos              0.029671
Entre 36 e 40 anos   0.027457
24 anos              0.025294
25 anos              0.021004
Entre 41 e 45 anos   0.017332
Entre 46 e 50 anos   0.010563
Entre 51 e 55 anos   0.006066
Entre 56 e 60 anos   0.003187
Entre 61 e 65 anos   0.001210
Entre 66 e 70 anos   0.000396
Maior de 70 anos     0.000165
Name: TP_FAIXA_ETARIA, dtype: float64
```

```
plt.figure(figsize=(10,6))
sns.countplot(y= df2['TP_FAIXA_ETARIA'],order=['Menor de 17 anos','17 anos','18 anos','19 anos','20 anos','21 anos',
            '22 anos','23 anos','24 anos','25 anos','Entre 26 e 30 anos',
            'Entre 31 e 35 anos','Entre 36 e 40 anos','Entre 41 e 45 anos',
            'Entre 46 e 50 anos','Entre 51 e 55 anos','Entre 56 e 60 anos',
            'Entre 61 e 65 anos','Entre 66 e 70 anos','Maior de 70 anos'])
plt.title('Quantidade de inscritor por faixa etária', fontsize = 16)
plt.xlabel('Contagem',fontsize=16)
plt.ylabel('Faixa Etária', fontsize = 16)
plt.grid(True)
plt.show()
#plt.legend(['Menor de 17 anos','17 anos','18 anos','19 anos','20 anos','21 anos',
#            '22 anos','23 anos','24 anos','25 anos','Entre 26 e 30 anos',
#            'Entre 31 e 35 anos','Entre 36 e 40 anos','Entre 41 e 45 anos',
#            'Entre 46 e 50 anos','Entre 51 e 55 anos','Entre 56 e 60 anos',
#            'Entre 61 e 65 anos','Entre 66 e 70 anos','Maior de 70 anos'])
```




Como era de se esperar, a grande maioria dos candidadatos do ENEM 2021 encontram-se na faixa etária dos que estão finalizando ou que finalizaram a pouco tempo o ensino médio. Isso justifica-se pelo fato de que, desde 2009 o exame passou a ser a porta de entrada para universidades federais por todo o país, além dos programas de bolsas e financiamentos estudantis que fazem uso das provas do ENEM (ProUni, Fies).

```
#Treineiros
df2['IN_TREINEIRO'].value_counts(normalize=True)
```

```
0    0.871029
1    0.128971
Name: IN_TREINEIRO, dtype: float64
```

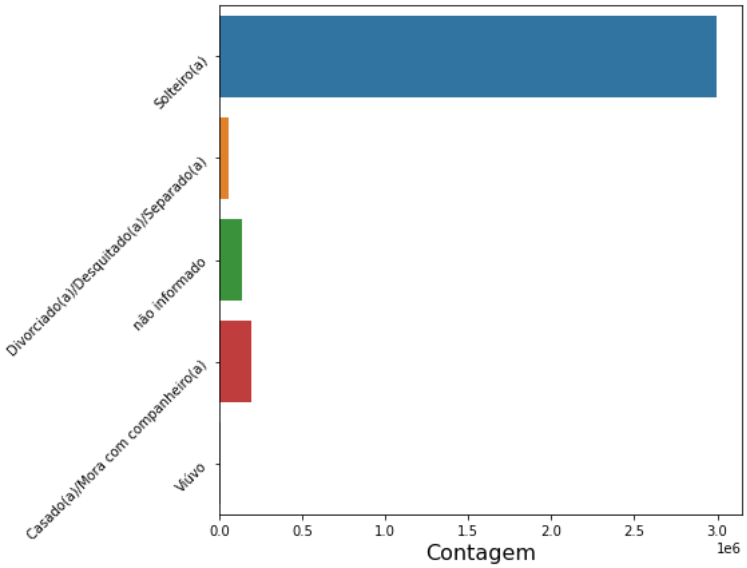
Do total de inscritos, 12,9% fizeram a prova como treineiros, ou seja, fizeram a prova apenas com o intuito de treinar seus conhecimentos.

```
dict_estcivil = {
    0:'não informado',
    1: 'Solteiro(a)',
    2: 'Casado(a)/Mora com companheiro(a)',
    3: 'Divorciado(a)/Desquitado(a)/Separado(a)',
    4: 'Viúvo'
}

dict_cor = {
    0:'Não declarado',
    1: 'Branca',
    2: 'Preta',
    3: 'Parda',
    4: 'Amarela',
    5: 'Indígena',
    6: 'Não dispõe da informação'
}

df2['TP_ESTADO_CIVIL'] = df2['TP_ESTADO_CIVIL'].map(dict_estcivil)
df2['TP_COR_RACA'] = df2['TP_COR_RACA'].map(dict_cor)
```

```
<ipython-input-14-401b4d64b187>:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df2['TP_ESTADO_CIVIL'] = df2['TP_ESTADO_CIVIL'].map(dict_estcivil)
<ipython-input-14-401b4d64b187>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df2['TP_COR_RACA'] = df2['TP_COR_RACA'].map(dict_cor)
```

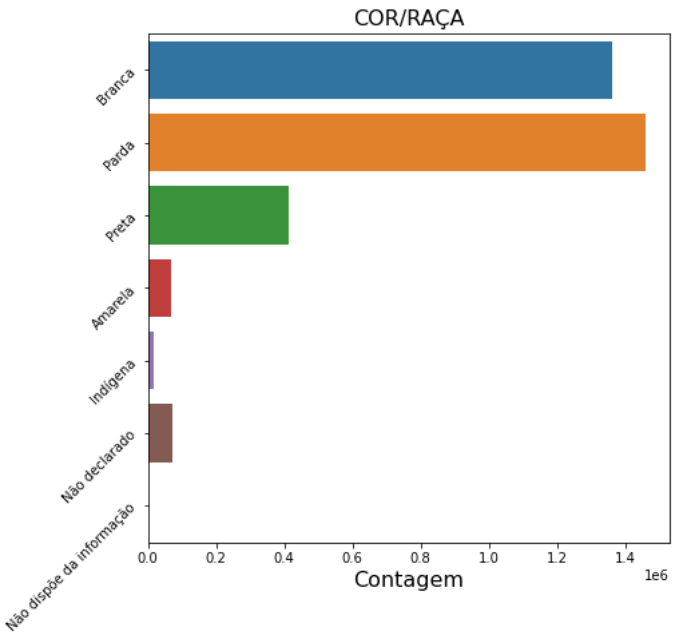
```
#Distribuição de candidatos por estado civil
plt.figure(figsize=(7,7))
sns.countplot(y = df2['TP_ESTADO_CIVIL'])
plt.title('ESTADO CIVIL',fontsize = 16)
plt.xlabel('Contagem',fontsize = 16)
plt.ylabel('',fontsize = 16)
plt.yticks(rotation=45)
plt.show()
```

```
round(df2['TP_ESTADO_CIVIL'].value_counts(normalize=True),3)
```




```
Solteiro(a)                             0.884
Casado(a)/Mora com companheiro(a)       0.058
não informado                           0.041
Divorciado(a)/Desquitado(a)/Separado(a) 0.016
Viúvo                                   0.001
Name: TP_ESTADO_CIVIL, dtype: float64
```

```
#Distribuição de candidatos por cor/raça
plt.figure(figsize=(7,7))
sns.countplot(y = df2['TP_COR_RACA'])
plt.title('COR/RAÇA',fontsize = 16)
plt.xlabel('Contagem',fontsize = 16)
plt.ylabel('',fontsize = 16)
plt.yticks(rotation=45)
plt.show()

round(df2['TP_COR_RACA'].value_counts(normalize=True),3)
```



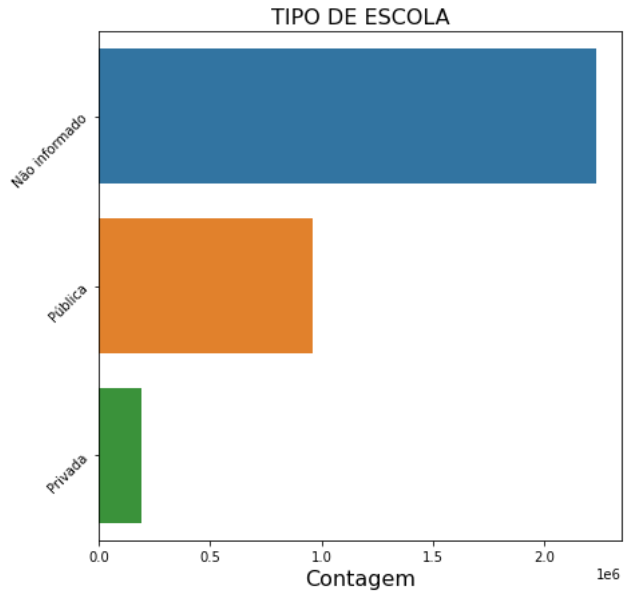

```
Parda                       0.430
Branca                      0.402
Preta                       0.121
Não declarado               0.021
Amarela                     0.020
Indígena                    0.006
Não dispõe da informação    0.000
Name: TP_COR_RACA, dtype: float64
```

```
dict_tipo_escola = {
    1: 'Não informado',
    2: 'Pública',
    3: 'Privada'
}
```

```
df2['TP_ESCOLA'] = df2['TP_ESCOLA'].map(dict_tipo_escola)
```

```
<ipython-input-17-7487afc35aaf>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df2['TP_ESCOLA'] = df2['TP_ESCOLA'].map(dict_tipo_escola)
```

```
#Distribuição de candidatos por tipo de escola (publico/privado)
plt.figure(figsize=(7,7))
sns.countplot(y = df2['TP_ESCOLA'])
plt.title('TIPO DE ESCOLA',fontsize = 16)
plt.xlabel('Contagem',fontsize = 16)
plt.ylabel('',fontsize = 16)
plt.yticks(rotation=45)
plt.show()

round(df2['TP_ESCOLA'].value_counts(normalize=True),3)
```




```
Não informado    0.660
Pública          0.283
Privada          0.057
Name: TP_ESCOLA, dtype: float64
```

Infelizmente, 66% dos inscritos não informaram o tipo de escola que frequentaram. Do total, 28,3% declararam que frequentaram escola pública e apenas 5,7% declararam ter estudado em escola particular.

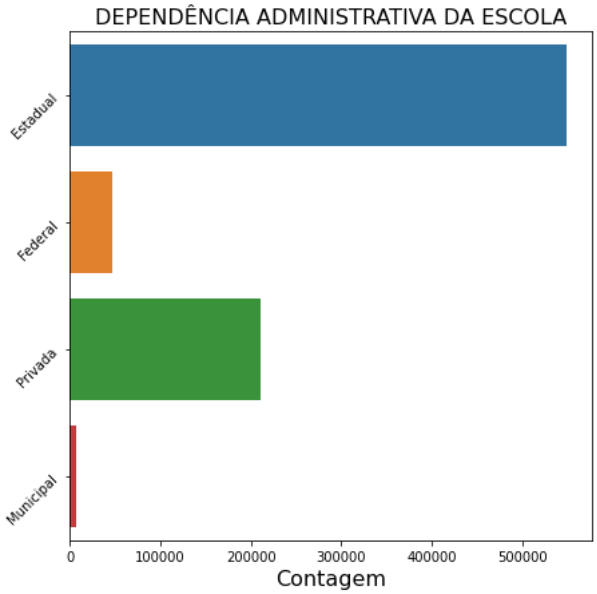
É possível que esta informação tenha sido absorvida na coluna que diz respeito à esfera administrativa da escola do candidato. Vejamos...

```
dict_esfera_administrativa = {
1: 'Federal',
2: 'Estadual',
3: 'Municipal',
4: 'Privada'
}

df2['TP_DEPENDENCIA_ADM_ESC'] = df2['TP_DEPENDENCIA_ADM_ESC'].map(dict_esfera_administrativa)
```

```
<ipython-input-19-51f66b5ab018>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df2['TP_DEPENDENCIA_ADM_ESC'] = df2['TP_DEPENDENCIA_ADM_ESC'].map(dict_esfera_administrativa)
```

```
#Distribuição de candidatos por esfera administrativa da escola (Federal,Estadual,Municipal e Privada)
plt.figure(figsize=(7,7))
sns.countplot(y = df2['TP_DEPENDENCIA_ADM_ESC'])
plt.title('DEPENDÊNCIA ADMINISTRATIVA DA ESCOLA',fontsize = 16)
plt.xlabel('Contagem',fontsize = 16)
plt.ylabel('',fontsize = 16)
plt.yticks(rotation=45)
plt.show()
```

```
round(df2['TP_DEPENDENCIA_ADM_ESC'].value_counts(normalize=True),3)
```




```
Estadual     0.675
Privada      0.259
Federal      0.057
Municipal    0.009
Name: TP_DEPENDENCIA_ADM_ESC, dtype: float64
```

Agora sim temos uma noção melhor sobre o tipo de escola que os candidatos frequentaram. Dos 5,7% que declararam terem estudado em escolas privadas no item anterior, temos que este número agora é de 25,9%, deixando 74,1% para alunos de escolas públicas (federais, estaduais e municipais).

## Análise dos candidatos presentes nos dois dias

Para selecionar os candidatos que estiveram presente nos dois dias e, consequentemente, fizeram as quatro provas(CH,LC,CN,MT) e a redação, vamos selecionar candidatos cujo status nas respectivas colunas seja igual a 1. Dado que, de acordo com o dicionário, 0 significa **ausente** e 2 significa **eliminado**.

```
df2['TP_PRESENCA_CH'].value_counts()
```

```
1    2378379
0    1007397
2       4056
Name: TP_PRESENCA_CH, dtype: int64
```

```
df_p = df2[(df2['TP_PRESENCA_CN'] == 1) & (df2['TP_PRESENCA_CH'] == 1) & (df2['TP_PRESENCA_LC'] == 1) & (df2['TP_PRESENCA_MT'
df_p.head()
```

| | TP_FAIXA_ETARIA | TP_SEXO | TP_ESTADO_CIVIL | TP_COR_RACA | TP_NACIONALIDADE |
|---|---|---|---|---|---|
| **1** | Entre 31 e 35 anos | M | Solteiro(a) | Branca | 1 |
| **3** | 18 anos | M | Solteiro(a) | Parda | 1 |
| **4** | 17 anos | F | Solteiro(a) | Parda | 1 |
| **8** | 23 anos | F | Solteiro(a) | Parda | 1 |
| **9** | 19 anos | F | Solteiro(a) | Parda | 1 |

5 rows × 31 columns

```
#Verificando o formato do dataframe resultante
df_p.shape
```

```
(2238107, 31)
```

Para aliviar a memória do sistema, vamos deletar os dataframes usados anteriormente e ficar exclusivamente com o dataframe dos candidatos presentes a partir de agora.

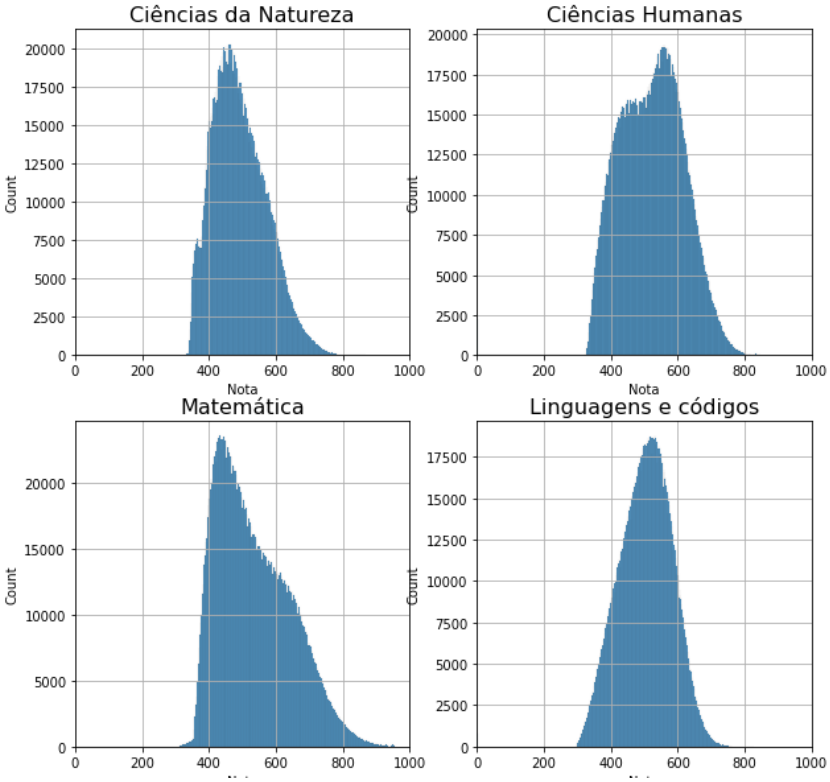
```
del df
del df2
```

```
df_p.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2238107 entries, 1 to 3389830
Data columns (total 31 columns):
 #   Column                  Dtype  
---  ------                  -----  
 0   TP_FAIXA_ETARIA         object 
 1   TP_SEXO                 object 
 2   TP_ESTADO_CIVIL         object 
 3   TP_COR_RACA             object 
 4   TP_NACIONALIDADE        int64  
 5   TP_ST_CONCLUSAO         int64  
 6   TP_ANO_CONCLUIU         int64  
 7   TP_ESCOLA               object 
 8   TP_ENSINO               float64
 9   IN_TREINEIRO            int64  
 10  NO_MUNICIPIO_ESC        object 
 11  CO_UF_ESC               float64
 12  SG_UF_ESC               object 
 13  TP_DEPENDENCIA_ADM_ESC  object 
 14  TP_LOCALIZACAO_ESC      float64
 15  TP_SIT_FUNC_ESC         float64
 16  TP_PRESENCA_CN          int64  
 17  TP_PRESENCA_CH          int64  
 18  TP_PRESENCA_LC          int64  
 19  TP_PRESENCA_MT          int64  
 20  NU_NOTA_CN              float64
 21  NU_NOTA_CH              float64
 22  NU_NOTA_LC              float64
 23  NU_NOTA_MT              float64
 24  TP_STATUS_REDACAO       float64
 25  NU_NOTA_COMP1           float64
 26  NU_NOTA_COMP2           float64
 27  NU_NOTA_COMP3           float64
 28  NU_NOTA_COMP4           float64
 29  NU_NOTA_COMP5           float64
 30  NU_NOTA_REDACAO         float64
dtypes: float64(15), int64(8), object(8)
memory usage: 546.4+ MB
```

Agora vamos fazer histogramas com as distribuições das notas de Liguagem e códigos, Matemática, Ciências humanas e Ciências da Natureza.

```
plt.figure(figsize=(10,10))
plt.title('NOTAS DAS QUATRO PROVAS')
plt.subplot(2,2,1)
sns.histplot(x = df_p['NU_NOTA_CN'])
plt.xlabel('Nota')
plt.xlim([0,1000])
plt.grid(True)
plt.title('Ciências da Natureza',fontsize=16)

plt.subplot(2,2,2)
sns.histplot(x = df_p['NU_NOTA_CH'])
plt.xlabel('Nota')
plt.xlim([0,1000])
plt.title('Ciências Humanas',fontsize=16)
plt.grid(True)

plt.subplot(2,2,3)
sns.histplot(x = df_p['NU_NOTA_MT'])
plt.xlabel('Nota')
plt.xlim([0,1000])
plt.grid(True)
plt.title('Matemática',fontsize=16)

plt.subplot(2,2,4)
sns.histplot(x = df_p['NU_NOTA_LC'])
plt.xlabel('Nota')
plt.xlim([0,1000])
plt.title('Linguagens e códigos',fontsize=16)
plt.grid(True)

plt.show()
```




```
df_p.groupby(['TP_DEPENDENCIA_ADM_ESC'])['NU_NOTA_CN','NU_NOTA_MT','NU_NOTA_CH','NU_NOTA_LC'].mean()

# O mesmo resultado acima pode ser obtido com uma pivot table
#table = pd.pivot_table(df_p,values=['NU_NOTA_CN','NU_NOTA_MT','NU_NOTA_CH','NU_NOTA_LC'],index=['TP_DEPENDENCIA_ADM_ESC'],ag
#table
```

```
<ipython-input-27-1b1c9f048298>:1: FutureWarning: Indexing with multiple ke
  df_p.groupby(['TP_DEPENDENCIA_ADM_ESC'])['NU_NOTA_CN','NU_NOTA_MT','NU_NC
```

| TP_DEPENDENCIA_ADM_ESC | NU_NOTA_CN | NU_NOTA_MT | NU_NOTA_CH | NU_NOTA_LC |
|---|---|---|---|---|
| **Estadual** | 464.019468 | 502.559888 | 492.744045 | 480.778366 |
| **Federal** | 531.647300 | 599.432620 | 567.340799 | 543.505746 |
| **Municipal** | 475.947207 | 521.442767 | 506.620749 | 492.462595 |
| **Privada** | 538.450762 | 607.110279 | 567.108973 | 545.721261 |

Na tabela acima vemos que a rede privada possui as melhores notas em praticamente todas as provas, com exceção da prova de ciências humanas que é praticamente igual à do nível federal.

Por fim, vamos avaliar as notas das redações geral e por rede de ensino.

```
plt.figure(figsize=(7,7))
plt.title('NOTA DA REDAÇÃO',fontsize=16)
sns.histplot(x = df_p['NU_NOTA_REDACAO'])
plt.xlabel('Nota',fontsize = 16)
plt.ylabel('Contagem',fontsize = 16)
plt.grid(True)


plt.show()
```


```
#Tabela com o resumo das notas da redação por rede de ensino
print(df_p.groupby(['TP_DEPENDENCIA_ADM_ESC'])['NU_NOTA_REDACAO'].mean())
print('\n')
print((df_p['NU_NOTA_REDACAO'] == 1000).sum(),' redações atingiram a nota 1000 (mil)')
```

```
TP_DEPENDENCIA_ADM_ESC
Estadual     563.202694
Federal      700.532898
Municipal    578.788623
Privada      730.505273
Name: NU_NOTA_REDACAO, dtype: float64


21  redações atingiram a nota 1000 (mil)
```

```
df_redacao1000 = df_p[df_p['NU_NOTA_REDACAO'] == 1000]
df_redacao1000
```

| | | | | |
|---|---|---|---|---|
| **77754** | 17 anos | M | Solteiro(a) | Branca |
| **129842** | Menor de 17 anos | F | Solteiro(a) | Branca |

```
df_p['TP_DEPENDENCIA_ADM_ESC'].isnull().sum()
```

```
1635656
```

| | | | | |
|---|---|---|---|---|
| **248838** | 20 anos | F | Solteiro(a) | Branca |

## Conclusões

Dos resultados mostrados nesta análise exploratória podemos ver as distribuições de notas dos quatro tipos de provas e da redação de cada aluno. Vimos que as escolas da rede privada lideram o ranking de escolas com maiores notas, seguido pela rede federal. Este resultado deve servir de alerta aos governos estaduais pois, aparentemente, a rede estadual é a pior entre as quatro redes existentes.

| | | | | |
|---|---|---|---|---|
| **604348** | 19 anos | M | Solteiro(a) | Branca |
| **725913** | 18 anos | F | Solteiro(a) | Branca |
| **899896** | 22 anos | F | Solteiro(a) | Branca |
| **905609** | 22 anos | F | Solteiro(a) | Branca |
| **998606** | 20 anos | F | Solteiro(a) | Branca |
| **1022568** | 20 anos | F | Solteiro(a) | Branca |
| **1193131** | 21 anos | F | Solteiro(a) | Branca |
| **1397534** | 18 anos | F | Solteiro(a) | Branca |
| **1402308** | 19 anos | F | Solteiro(a) | Branca |

✓ 0s conclusão: 21:49