Análise de Dados INEP 2023

Como fatores além estudo impactam a Prova?

Isac Vieira

12/24

Definição de Objetivo:

Por intuição podemos definir que fatores de tempo/qualidade de estudo não são os únicos que impactam os resultados de uma prova. Nessa análise, nosso objetivo é analisar os dados do ENEM de 2023 e obter conclusões não apenas sobre a existência da desigualdade mas o quanto ela impacta.

Definição de Principais Perguntas:

- Qual faixa etária teve o melhor desepenho?
- As notas do Enem variam com a renda familiar?
- O nível de escolaridade dos pais influencia as notas dos estudantes?
- As notas diferem significativamente entre homens e mulheres?
- Qual é a distribuição de notas por etnia/cor/raça e há diferenças marcantes?
- O tipo de dependência administrativa da escola (pública/privada) afeta as notas?
- Qual é o perfil de pessoas que tinham como objetivo passar em Medicina/Cursos muito concorridos (750+)?
- Qual é o perfil de pessoas que tiveram um mau desempenho na prova (-500)?

Próximas Perguntas (Próximas Análises):

- Pessoas com acesso a internet tiveram um desempenho melhor?
- Pessoas com acesso a computador tiveram um desempenho melhor?
- Escolas Urbanas têm vantagem de pontos em relação a escolas rurais? Qual o tamanho dessa vantagem?
- Qual região tem melhor desempenho na Prova? Qual estado se sai melhor?

In [1]:
```
#pip install nbconvert
```

In [2]:
```
import pandas as pd
import polars as pl
import matplotlib.pyplot as plt
```

```
import matplotlib.ticker as ticker
import seaborn as sns
import numpy as np
sns.set()
```

In [3]:
```
dfa = pd.read_csv(r'./microdados_enem_2023/DADOS/MICRODADOS_ENEM_2023.csv', encodin
dfa.head()
```

Out[3]:

	NU_INSCRICAO	NU_ANO	TP_FAIXA_ETARIA	TP_SEXO	TP_ESTADO_CIVIL	TP_COR_RACA
0	210059085136	2023	14	M	2	1
1	210059527735	2023	12	M	2	1
2	210061103945	2023	6	F	1	1
3	210060214087	2023	2	F	1	3
4	210059980948	2023	3	F	1	3

5 rows × 76 columns

In [4]:
```
dfa.shape
```

Out[4]:
```
(3933955, 76)
```

In [5]:
```
colex = [
    "NU_INSCRICAO", "NU_ANO", "TP_ESTADO_CIVIL", "TP_NACIONALIDADE", "TP_ST_CONCLUS
    "TP_ENSINO", "IN_TREINEIRO", "CO_MUNICIPIO_ESC", "TX_RESPOSTAS_CN", "TX_RESPOST
    "TX_GABARITO_CN", "TX_GABARITO_CH", "TX_GABARITO_LC", "TX_GABARITO_MT",
    "CO_PROVA_CN",      "CO_PROVA_CH",  "CO_PROVA_LC",  "CO_PROVA_MT",
    "NU_NOTA_COMP1",    "NU_NOTA_COMP2",        "NU_NOTA_COMP3",        "NU_NOTA_CO
    "CO_MUNICIPIO_PROVA", "NO_MUNICIPIO_PROVA", "CO_UF_PROVA",  "SG_UF_PROVA", "Q00
    'Q007',     'Q008', 'Q009', 'Q010', 'Q011', 'Q012', 'Q013', 'Q014', 'Q015', 'Q0
    'CO_UF_ESC', 'NO_MUNICIPIO_ESC', 'TP_SIT_FUNC_ESC', 'TP_PRESENCA_CN',       'TP
    'TP_LINGUA',        'TP_STATUS_REDACAO'
]

df = dfa.drop(columns = colex)
```

In [6]:
```
df.head()
```

Out[6]:

	TP_FAIXA_ETARIA	TP_SEXO	TP_COR_RACA	TP_ESCOLA	SG_UF_ESC	TP_DEPENDENCIA_A
0	14	M	1	1	NaN	
1	12	M	1	1	NaN	
2	6	F	1	1	NaN	
3	2	F	3	2	CE	
4	3	F	3	2	CE	

In [7]:
```
#Dimensão do dataset
print("Dimensão:", df.shape)
```

```
Dimensão: (3933955, 14)
```

Tratamento de Dados

In [8]:
```
#Checando os valores nulos com is_null().sum()
df.isnull().sum()
```

Out[8]:
```
TP_FAIXA_ETARIA                0
TP_SEXO                        0
TP_COR_RACA                    0
TP_ESCOLA                      0
SG_UF_ESC                2975449
TP_DEPENDENCIA_ADM_ESC   2975449
TP_LOCALIZACAO_ESC       2975449
NU_NOTA_CN               1241528
NU_NOTA_CH               1111312
NU_NOTA_LC               1111312
NU_NOTA_MT               1241528
NU_NOTA_REDACAO          1111312
Q001                           0
Q006                           0
dtype: int64
```

Abaixo poderíamos fazer o tratamento dos dados trocado simplesmente pela mediana dos dados, porém após alguns testes, isso trouxe um enviesamento muito grande nos dados, logo é intencional deixar os valores NaN do jeito que estão pelo bem da análise. Posteriormente pode ser interessante usar algum algoritmo de ML para fazer um tratamento adequando evitando esse risco.

In [9]:
```
'''
df.replace(["nan", "NaN"], np.nan, inplace=True)
df['NU_NOTA_CN'] = df['NU_NOTA_CN'].fillna(df['NU_NOTA_CN'].median())
df['NU_NOTA_CH'] = df['NU_NOTA_CH'].fillna(df['NU_NOTA_CH'].median())
df['NU_NOTA_LC'] = df['NU_NOTA_LC'].fillna(df['NU_NOTA_LC'].median())
df['NU_NOTA_MT'] = df['NU_NOTA_MT'].fillna(df['NU_NOTA_MT'].median())
df['NU_NOTA_REDACAO'] = df['NU_NOTA_REDACAO'].fillna(df['NU_NOTA_REDACAO'].median()
df.head()
'''
```

Out[9]:
```
'\ndf.replace(["nan", "NaN"], np.nan, inplace=True)\ndf[\'NU_NOTA_CN\'] = df[\'NU_
NOTA_CN\'].fillna(df[\'NU_NOTA_CN\'].median())\ndf[\'NU_NOTA_CH\'] = df[\'NU_NOTA_
CH\'].fillna(df[\'NU_NOTA_CH\'].median())\ndf[\'NU_NOTA_LC\'] = df[\'NU_NOTA_LC
\'].fillna(df[\'NU_NOTA_LC\'].median())\ndf[\'NU_NOTA_MT\'] = df[\'NU_NOTA_MT\'].f
illna(df[\'NU_NOTA_MT\'].median())\ndf[\'NU_NOTA_REDACAO\'] = df[\'NU_NOTA_REDACAO
\'].fillna(df[\'NU_NOTA_REDACAO\'].median())\ndf.head()\n'
```

In [10]:
```
df['media_notas'] = df[['NU_NOTA_CN', 'NU_NOTA_CH', 'NU_NOTA_LC', 'NU_NOTA_MT', 'NU
```

In [11]:
```
#Checando os valores nulos com is_null().sum()
```

```
df.isnull().sum()
```

Out[11]:
```
TP_FAIXA_ETARIA                0
TP_SEXO                        0
TP_COR_RACA                    0
TP_ESCOLA                      0
SG_UF_ESC                2975449
TP_DEPENDENCIA_ADM_ESC   2975449
TP_LOCALIZACAO_ESC       2975449
NU_NOTA_CN               1241528
NU_NOTA_CH               1111312
NU_NOTA_LC               1111312
NU_NOTA_MT               1241528
NU_NOTA_REDACAO          1111312
Q001                           0
Q006                           0
media_notas              1097149
dtype: int64
```

O mesmo aqui, é uma base muito grande para simplesmente trocarmos pela moda da categoria mais utilizada.

In [12]:
```
'''
df.replace(["nan", "NaN"], np.nan, inplace=True)
df['SG_UF_ESC'] = df['SG_UF_ESC'].fillna(df['SG_UF_ESC'].mode()[0])
df['TP_DEPENDENCIA_ADM_ESC'] = df['TP_DEPENDENCIA_ADM_ESC'].fillna(df['TP_DEPENDENC
df['TP_LOCALIZACAO_ESC'] = df['TP_LOCALIZACAO_ESC'].fillna(df['TP_LOCALIZACAO_ESC']
df.head()
'''
```

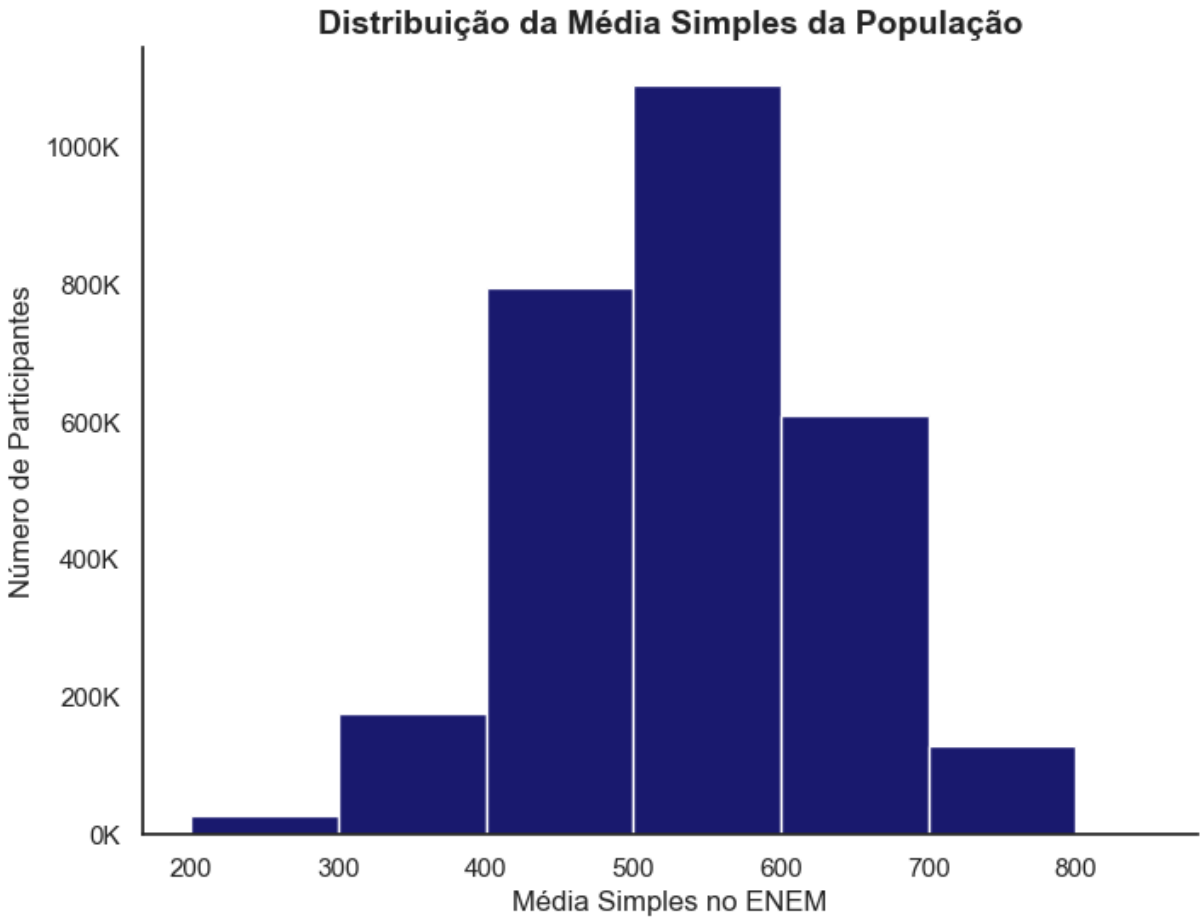
Out[12]:
```
'\ndf.replace(["nan", "NaN"], np.nan, inplace=True)\ndf[\'SG_UF_ESC\'] = df[\'SG_U
F_ESC\'].fillna(df[\'SG_UF_ESC\'].mode()[0])\ndf[\'TP_DEPENDENCIA_ADM_ESC\'] = df
[\'TP_DEPENDENCIA_ADM_ESC\'].fillna(df[\'TP_DEPENDENCIA_ADM_ESC\'].mode()[0])\ndf
[\'TP_LOCALIZACAO_ESC\'] = df[\'TP_LOCALIZACAO_ESC\'].fillna(df[\'TP_LOCALIZACAO_E
SC\'].mode()[0])\ndf.head()\n'
```

In [13]:
```
#Checando os valores nulos com is_null().sum()
df.isnull().sum()
```

Out[13]:
```
TP_FAIXA_ETARIA                0
TP_SEXO                        0
TP_COR_RACA                    0
TP_ESCOLA                      0
SG_UF_ESC                2975449
TP_DEPENDENCIA_ADM_ESC   2975449
TP_LOCALIZACAO_ESC       2975449
NU_NOTA_CN               1241528
NU_NOTA_CH               1111312
NU_NOTA_LC               1111312
NU_NOTA_MT               1241528
NU_NOTA_REDACAO          1111312
Q001                           0
Q006                           0
media_notas              1097149
dtype: int64
```

Distribuição de Nota por Participantes

In [14]:
```
sns.set_style('white')
bins = [200, 300, 400, 500, 600, 700, 800, 850]
plt.figure(figsize = (8,6))
plt.hist(df['media_notas'],
         bins = bins,
        color ='midnightblue')
plt.title('Distribuição da Média Simples da População', fontsize = 14, weight = 'bo
plt.xlabel("Média Simples no ENEM")
plt.ylabel('Número de Participantes')
# Ajustando o eixo Y para exibir números em notação de milhar
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show()
```



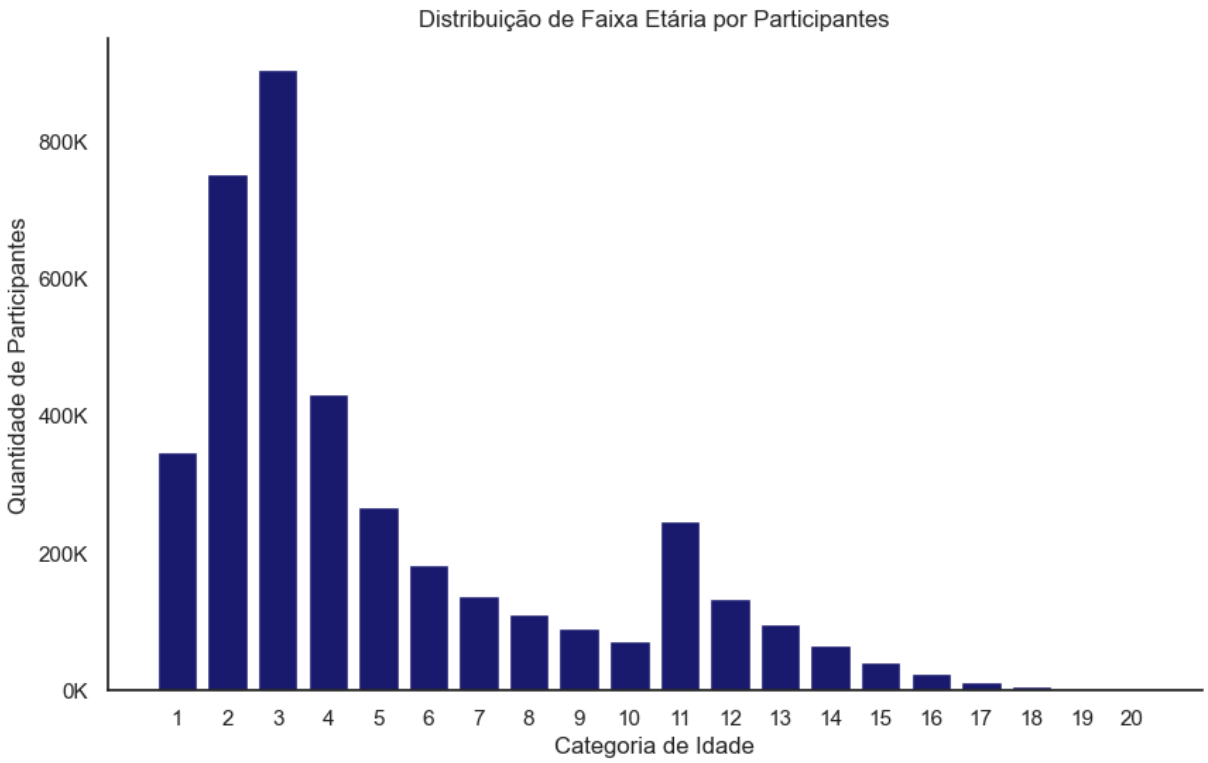
Distribuição de Faixa Etária

In [15]:
```
df['TP_FAIXA_ETARIA'].astype('str')
```

Out[15]:
```
0          14
1          12
2           6
3           2
4           3
           ..
3933950    12
3933951     1
3933952     3
3933953     2
3933954     3
Name: TP_FAIXA_ETARIA, Length: 3933955, dtype: object
```

In [16]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df['TP_FAIXA_ETARIA'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Distribuição de Faixa Etária por Participantes')
plt.xlabel('Categoria de Idade')
plt.ylabel('Quantidade de Participantes')
plt.xticks(unique, rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```



Categorias de Faixa Etária

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

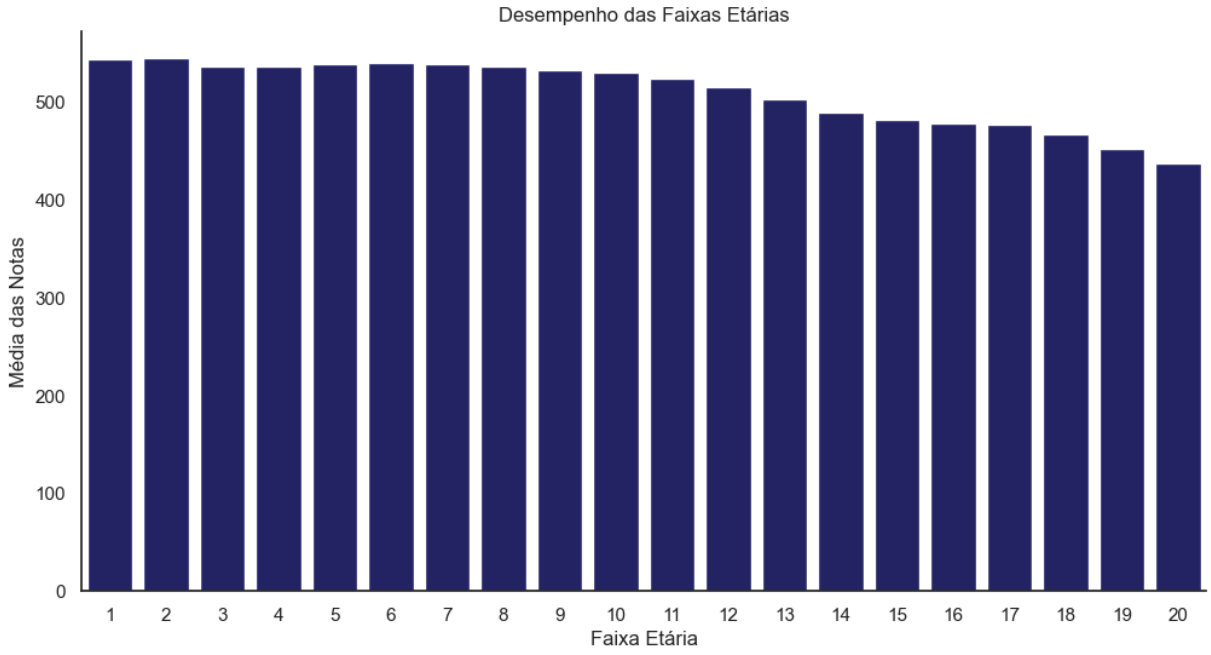
19: Entre 66 e 70 anos

20: Maior de 70 anos

In [17]:
```
# Agrupar as faixas etárias e calcular a média das notas
faixa_etaria= df.groupby('TP_FAIXA_ETARIA')['media_notas'].mean().reset_index()

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de barras com Matplotlib
plt.figure(figsize=(12, 6))
sns.barplot(x='TP_FAIXA_ETARIA', y='media_notas', data=faixa_etaria, color='midnigh

# Definindo título e rótulos
plt.title('Desempenho das Faixas Etárias')
plt.xlabel('Faixa Etária')
plt.ylabel('Média das Notas')
```

```
# Ajustando as categorias no eixo X
sns.despine(top = True, right = True)
plt.xticks(rotation=0)  # Girar os rótulos do eixo X para visualização

# Exibindo o gráfico
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



Categorias de Faixa Etária

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

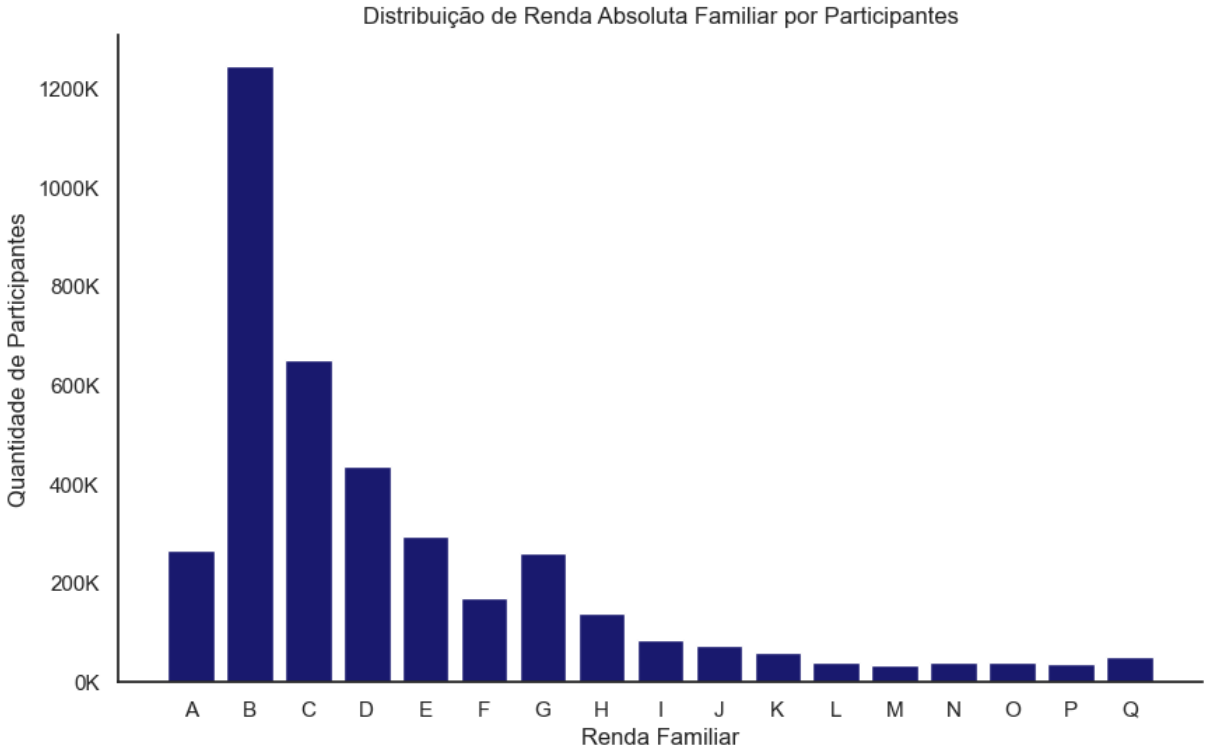
20: Maior de 70 anos

Percebe-se a medida que a idade aumenta existe uma tendência que a nota média decaia.

Distribuição de Renda por Participantes

In [46]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df['Q006'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Distribuição de Renda Absoluta Familiar por Participantes')
plt.xlabel('Renda Familiar')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```



Faixas de Renda Familiar

A: Nenhuma Renda

B: Até R$ 1.320,00

C: De R$1.320,01$**até**R **1.980,00**

D: De R$1.980,01$**até**R **2.640,00**

E: De R$2.640,01$**até**R **3.300,00**

F: De R$3.300,01$**até**R **3.960,00**

G: De R$3.960,01$**até**R **5.280,00**

H: De R$5.280,01$**até**R **6.600,00**

I: De R$6.600,01$**até**R **7.920,00**

J: De R$7.920,01$**até**R **9.240,00**

K: De R$9.240,01$**até**R **10.560,00**

L: De R$10.560,01$**até**R **11.880,00**

M: De R$11.880,01$**até**R **13.200,00**

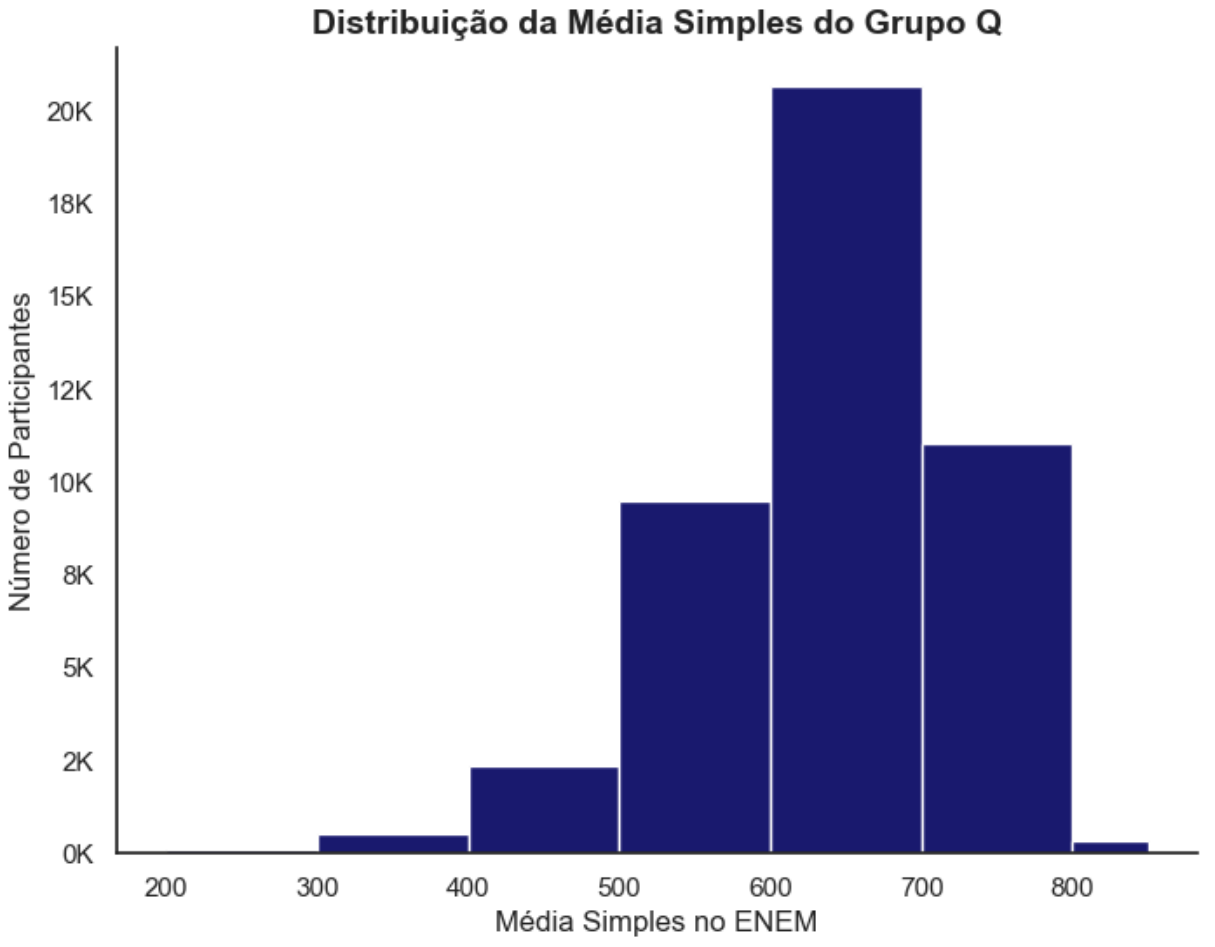
N: De R$13.200,01$**até**R **15.840,00**

O: De R$ 15.840,01 até R$ 19.800,00

P: De R$ 19.800,01 até R$ 26.400,00

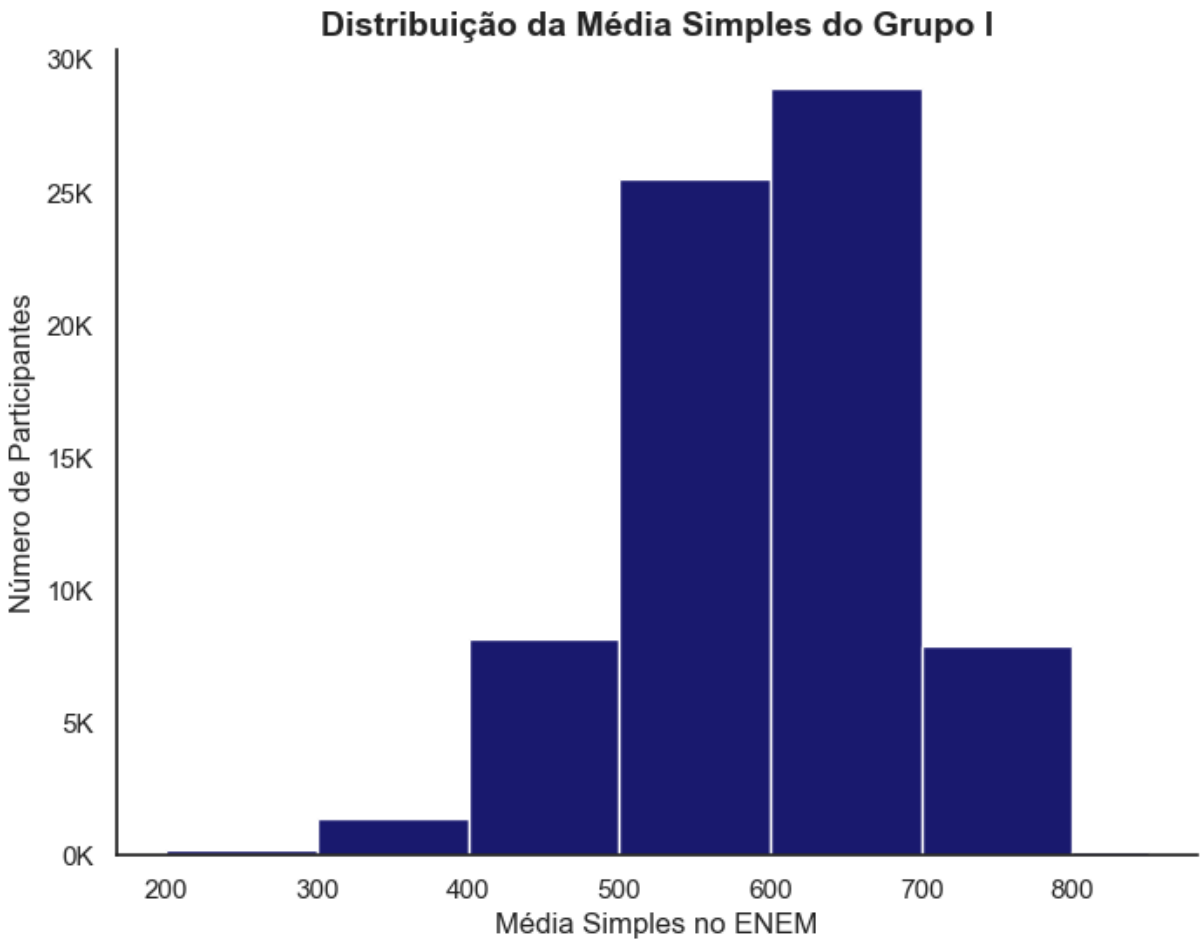
Q: Acima de R$ 26.400,00

In [19]:
```
df_q = df[df['Q006'] == 'Q']

sns.set_style('white')
bins = [200, 300, 400, 500, 600, 700, 800, 850]
plt.figure(figsize = (8,6))
plt.hist(df_q['media_notas'],
         bins = bins,
        color ='midnightblue')
plt.title('Distribuição da Média Simples do Grupo Q', fontsize = 14, weight = 'bold
plt.xlabel("Média Simples no ENEM")
plt.ylabel('Número de Participantes')
# Ajustando o eixo Y para exibir números em notação de milhar
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show()
```

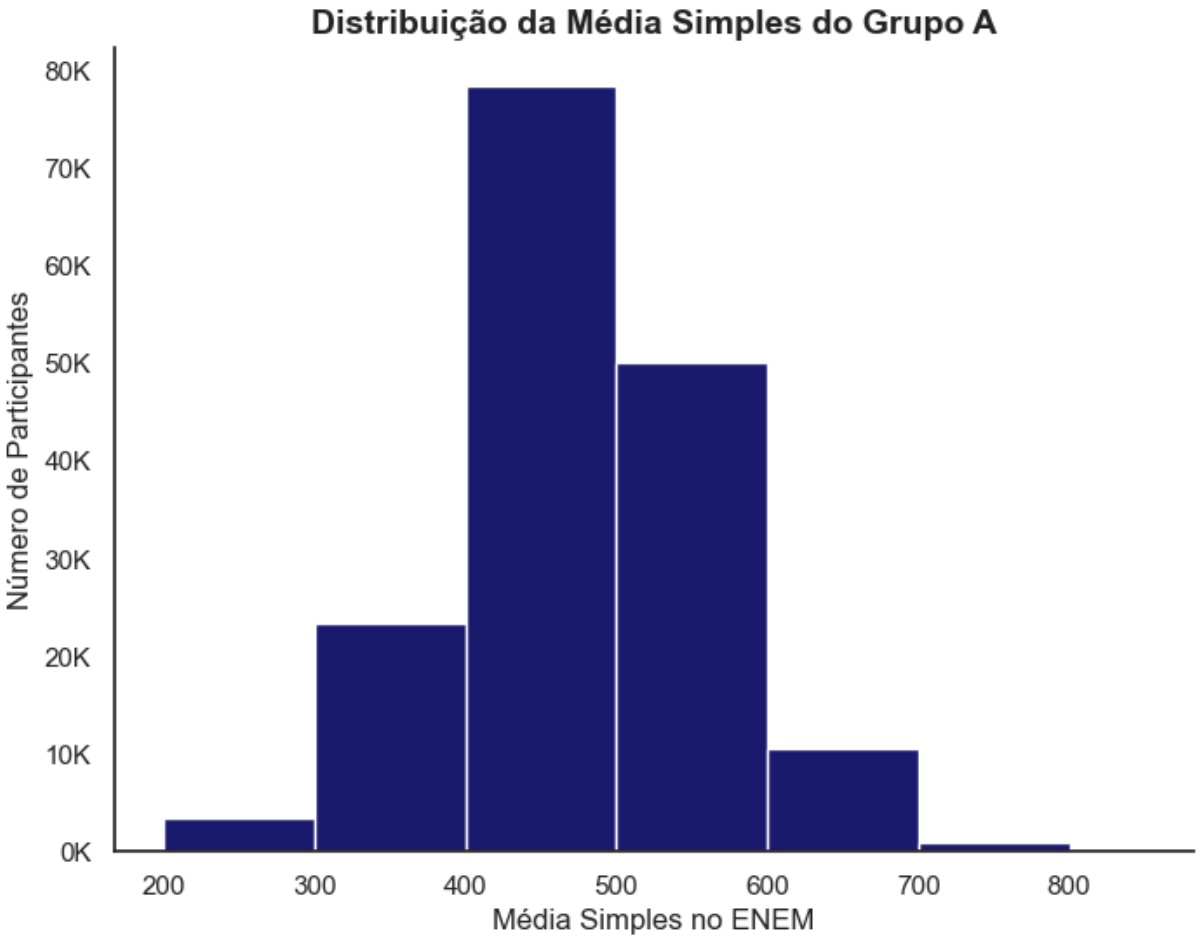


In [20]:
```
df_q = df[df['Q006'] == 'I']

sns.set_style('white')
bins = [200, 300, 400, 500, 600, 700, 800, 850]
```

```python
plt.figure(figsize = (8,6))
plt.hist(df_q['media_notas'],
         bins = bins,
        color ='midnightblue')
plt.title('Distribuição da Média Simples do Grupo I', fontsize = 14, weight = 'bold
plt.xlabel("Média Simples no ENEM")
plt.ylabel('Número de Participantes')
# Ajustando o eixo Y para exibir números em notação de milhar
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show()
```

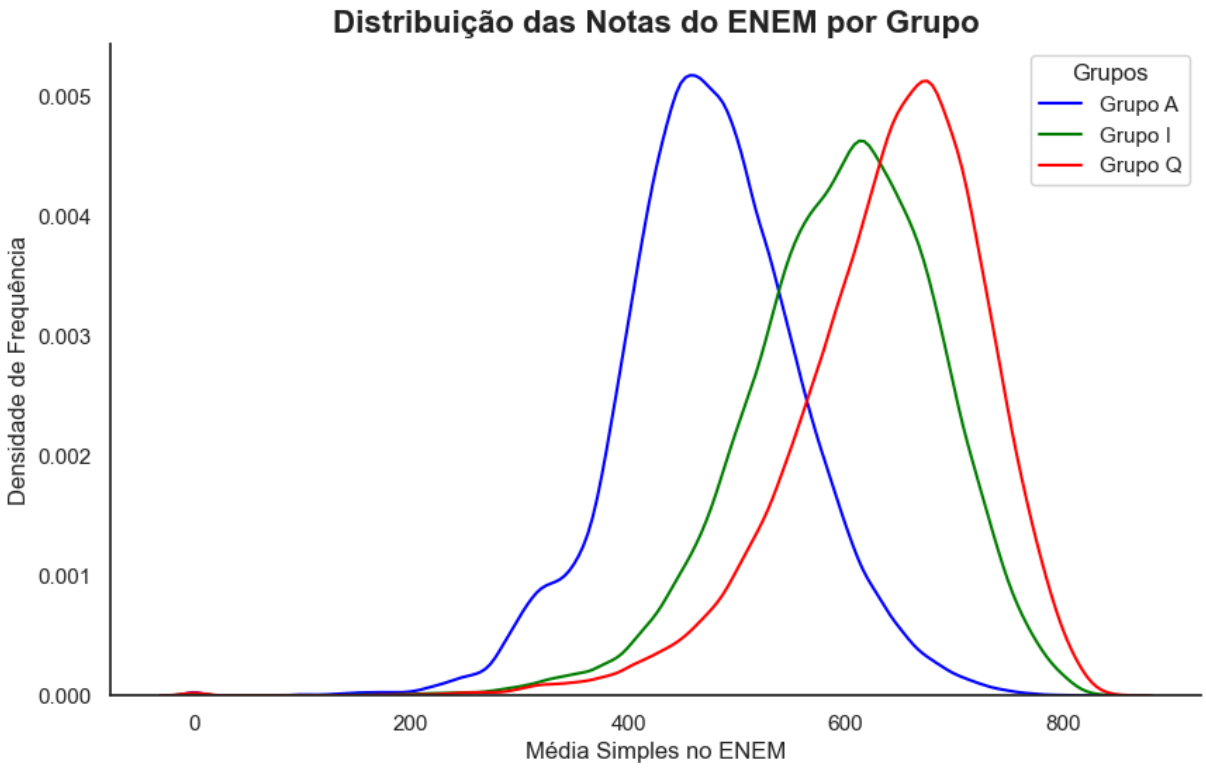


In [21]:
```
df_q = df[df['Q006'] == 'A']

sns.set_style('white')
bins = [200, 300, 400, 500, 600, 700, 800, 850]
plt.figure(figsize = (8,6))
plt.hist(df_q['media_notas'],
         bins = bins,
        color ='midnightblue')
plt.title('Distribuição da Média Simples do Grupo A', fontsize = 14, weight = 'bold
plt.xlabel("Média Simples no ENEM")
plt.ylabel('Número de Participantes')
# Ajustando o eixo Y para exibir números em notação de milhar
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show()
```



In [22]:
```
# Filtrando os dados para os três grupos A, I e Q
df_a = df[df['Q006'] == 'A']
df_i = df[df['Q006'] == 'I']
df_q = df[df['Q006'] == 'Q']

# Definindo o estilo do gráfico
sns.set_style('white')

# Criando a figura para o gráfico
plt.figure(figsize=(10, 6))
```

```
# Plotando as distribuições de notas para cada grupo
sns.kdeplot(df_a['media_notas'].to_numpy(), label='Grupo A', color='blue', fill=Fal
sns.kdeplot(df_i['media_notas'].to_numpy(), label='Grupo I', color='green', fill=Fa
sns.kdeplot(df_q['media_notas'].to_numpy(), label='Grupo Q', color='red', fill=Fals

# Adicionando título e rótulos aos eixos
plt.title('Distribuição das Notas do ENEM por Grupo', fontsize=16, weight='bold')
plt.xlabel('Média Simples no ENEM', fontsize=12)
plt.ylabel('Densidade de Frequência', fontsize=12)

# Adicionando legenda
plt.legend(title='Grupos', loc='upper right')
sns.despine(top = True, right = True)

# Exibindo o gráfico
plt.show()
```



Percepção da Diferença nas Notas com a Situação Econômica Familiar

Percebe-se que existe uma diferença significativa da nota com a situação econômica familiar, com o grupo A (Nenhuma Renda) apresentando notas mais centralizadas à esquerda, indicando um desempenho mais baixo.

O grupo Q (Acima de R$ 26.400) tem sua média simples à direita, sugerindo um desempenho superior.

Já o grupo I (De R$6.600,01 até R$ 7.920,00) manteve uma distribuição de notas no centro, mostrando uma performance média comparada aos outros grupos. A partir de sua média,

pode-se observar como o nível de renda impacta diretamente no desempenho.

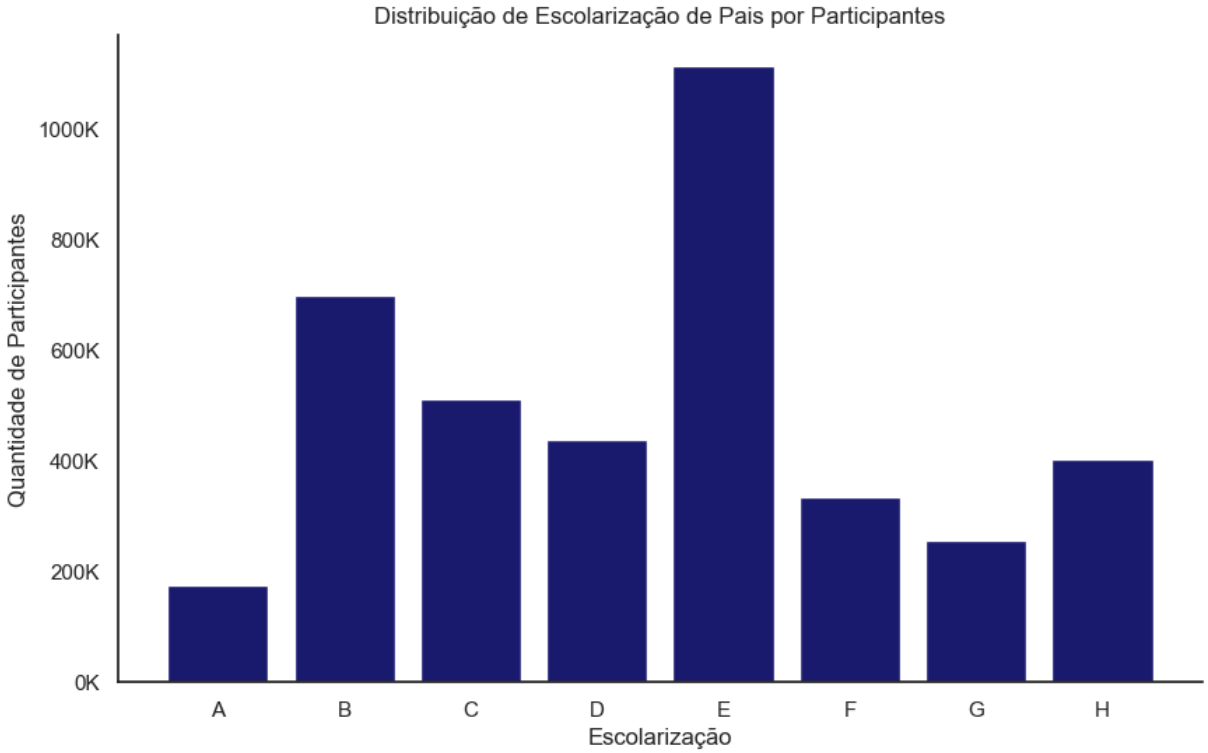
O nível de escolaridade do pai influencia na nota dos filhos?

Categorias de Escolarização

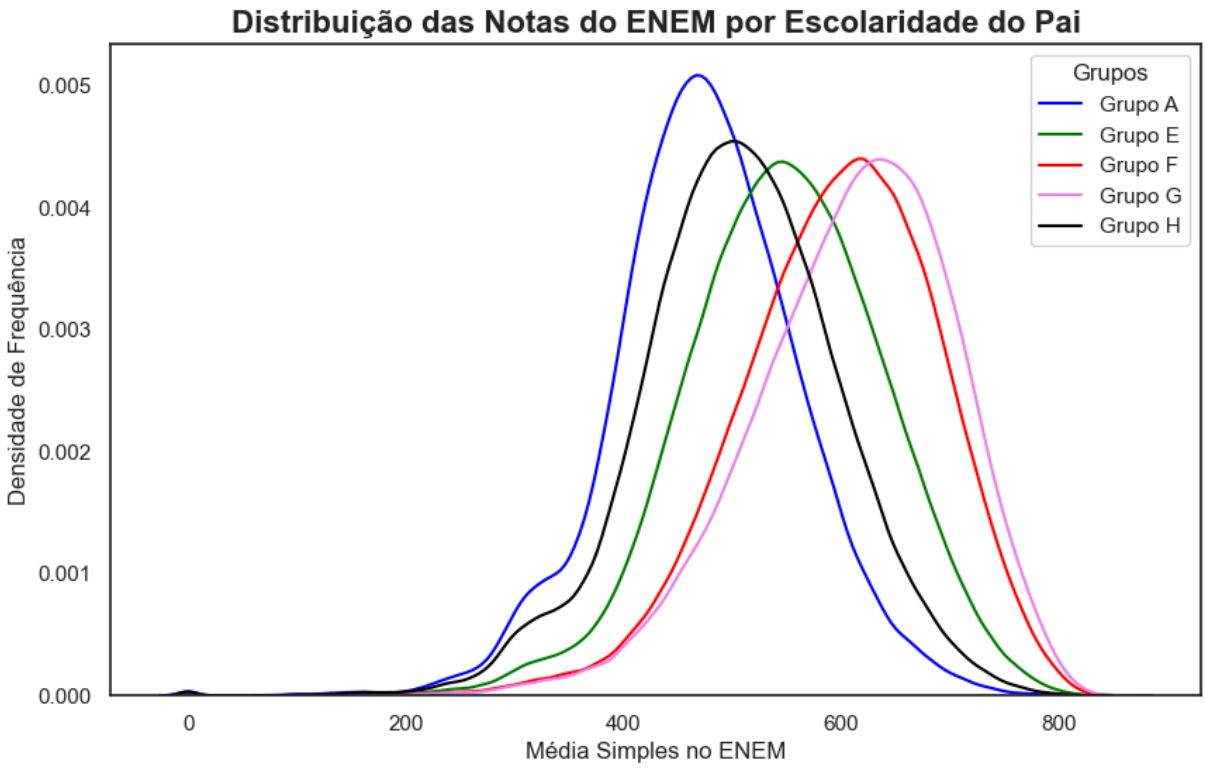
- A: Nunca estudou.
- B: Não completou a 4ª série/5º ano do Ensino Fundamental.
- C: Completou a 4ª série/5º ano, mas não completou a 8ª série/9º ano do Ensino Fundamental.
- D: Completou a 8ª série/9º ano do Ensino Fundamental, mas não completou o Ensino Médio.
- E: Completou o Ensino Médio, mas não completou a Faculdade.
- F: Completou a Faculdade, mas não completou a Pós-graduação.
- G: Completou a Pós-graduação.
- H: Não sei.

In [23]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df['Q001'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Distribuição de Escolarização de Pais por Participantes')
plt.xlabel('Escolarização')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```



In [24]:
```
# Filtrando os dados para os três grupos A, I e Q
df_a = df[df['Q001'] == 'A']
df_e = df[df['Q001'] == 'E']
df_f = df[df['Q001'] == 'F']
df_g = df[df['Q001'] == 'G']
df_h = df[df['Q001'] == 'H']

# Definindo o estilo do gráfico
sns.set_style('white')

# Criando a figura para o gráfico
plt.figure(figsize=(10, 6))

# Plotando as distribuições de notas para cada grupo
sns.kdeplot(df_a['media_notas'].to_numpy(), label='Grupo A', color='blue', fill=Fal
sns.kdeplot(df_e['media_notas'].to_numpy(), label='Grupo E', color='green', fill=Fa
sns.kdeplot(df_f['media_notas'].to_numpy(), label='Grupo F', color='red', fill=Fals
sns.kdeplot(df_g['media_notas'].to_numpy(), label='Grupo G', color='violet', fill=F
sns.kdeplot(df_h['media_notas'].to_numpy(), label='Grupo H', color='black', fill=Fa

# Adicionando título e rótulos aos eixos
plt.title('Distribuição das Notas do ENEM por Escolaridade do Pai', fontsize=16, we
plt.xlabel('Média Simples no ENEM', fontsize=12)
plt.ylabel('Densidade de Frequência', fontsize=12)

# Adicionando legenda
plt.legend(title='Grupos', loc='upper right')

# Exibindo o gráfico
plt.show()
```



Temos:

- A: Nunca estudou.
- E: Completou o Ensino Médio, mas não completou a Faculdade.
- F: Completou a Faculdade, mas não completou a Pós-graduação.
- G: Completou a Pós-graduação.
- H: Não sei.

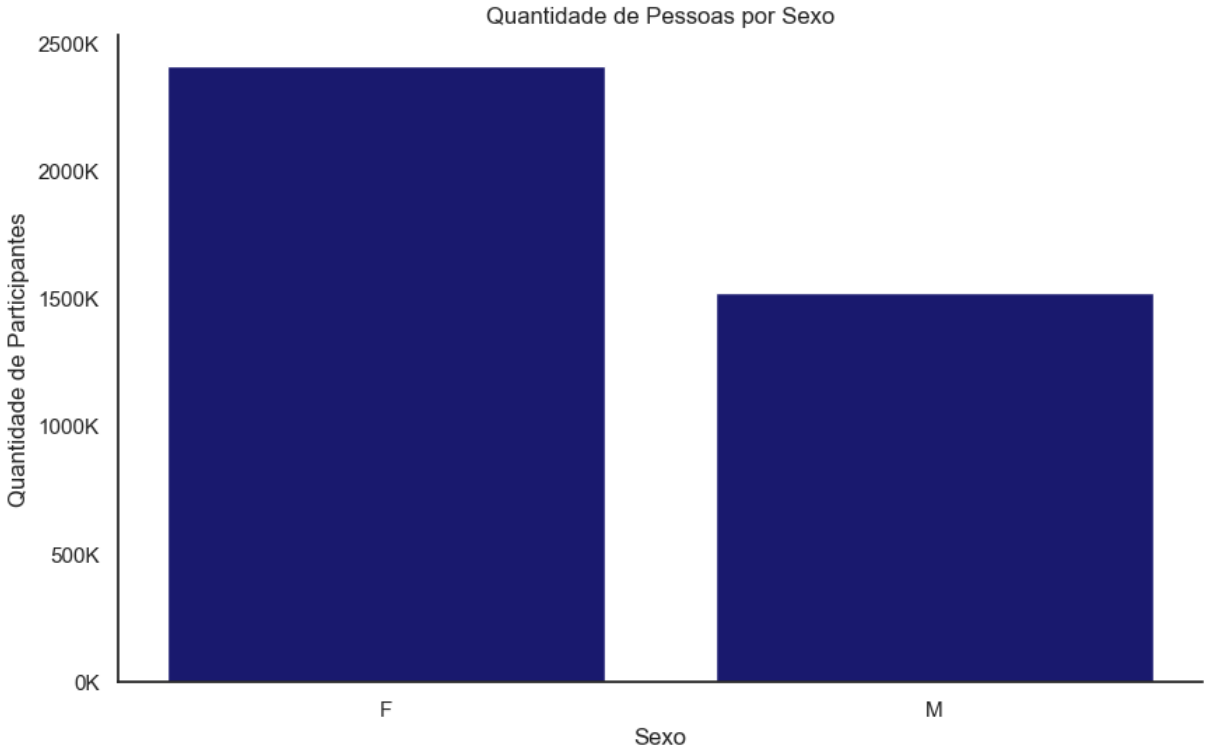
É interessante notar que quanto maior a escolarização da figura paterna existe maior tendência de aumento na nota do ENEM. Isso só não se mostra verdade quando não saber a escolaridade tem tendência melhor do que, saber a escolaridade do pai porém ela ser do grupo A.

Existe diferença entre notas quando falamos de homens/mulheres?

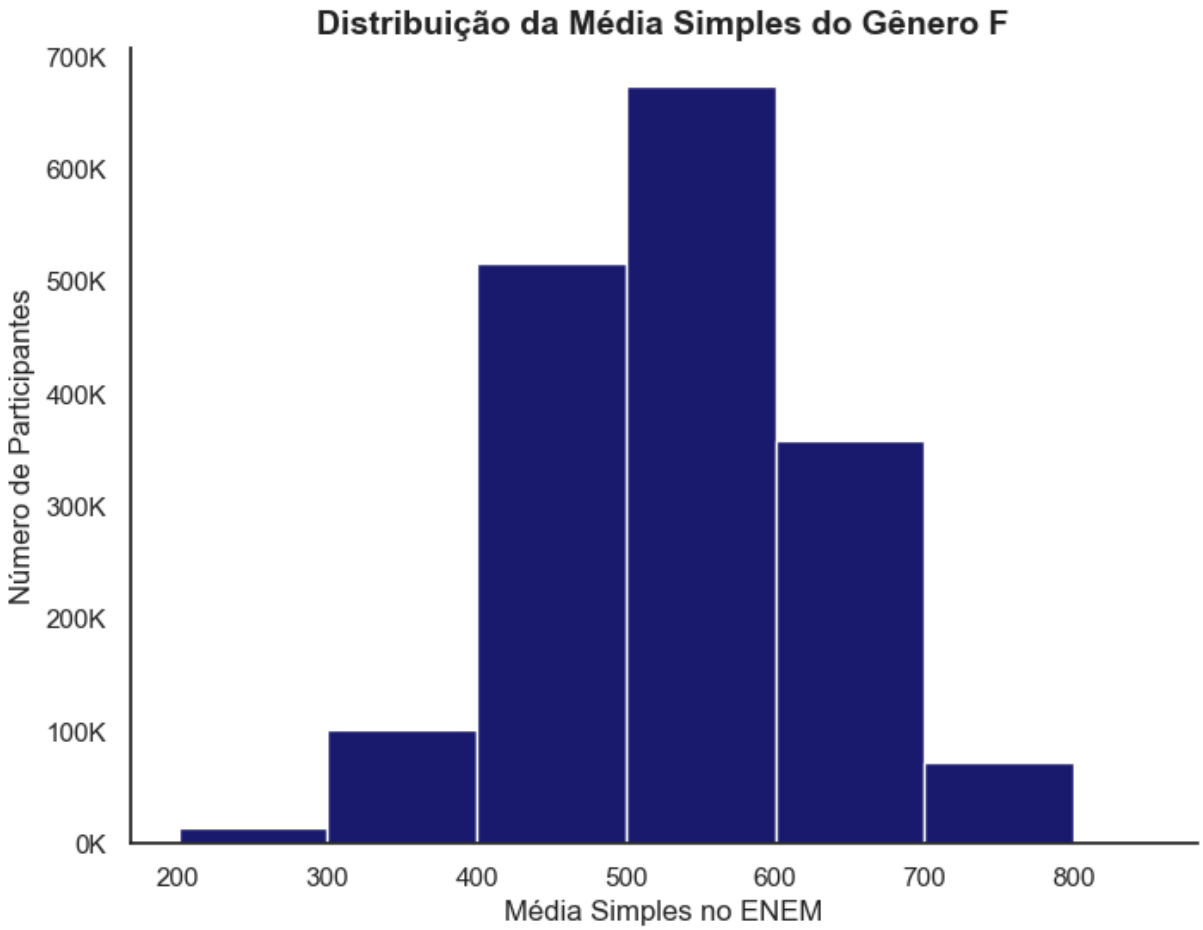
In [25]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df['TP_SEXO'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Quantidade de Pessoas por Sexo')
plt.xlabel('Sexo')
```

```
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

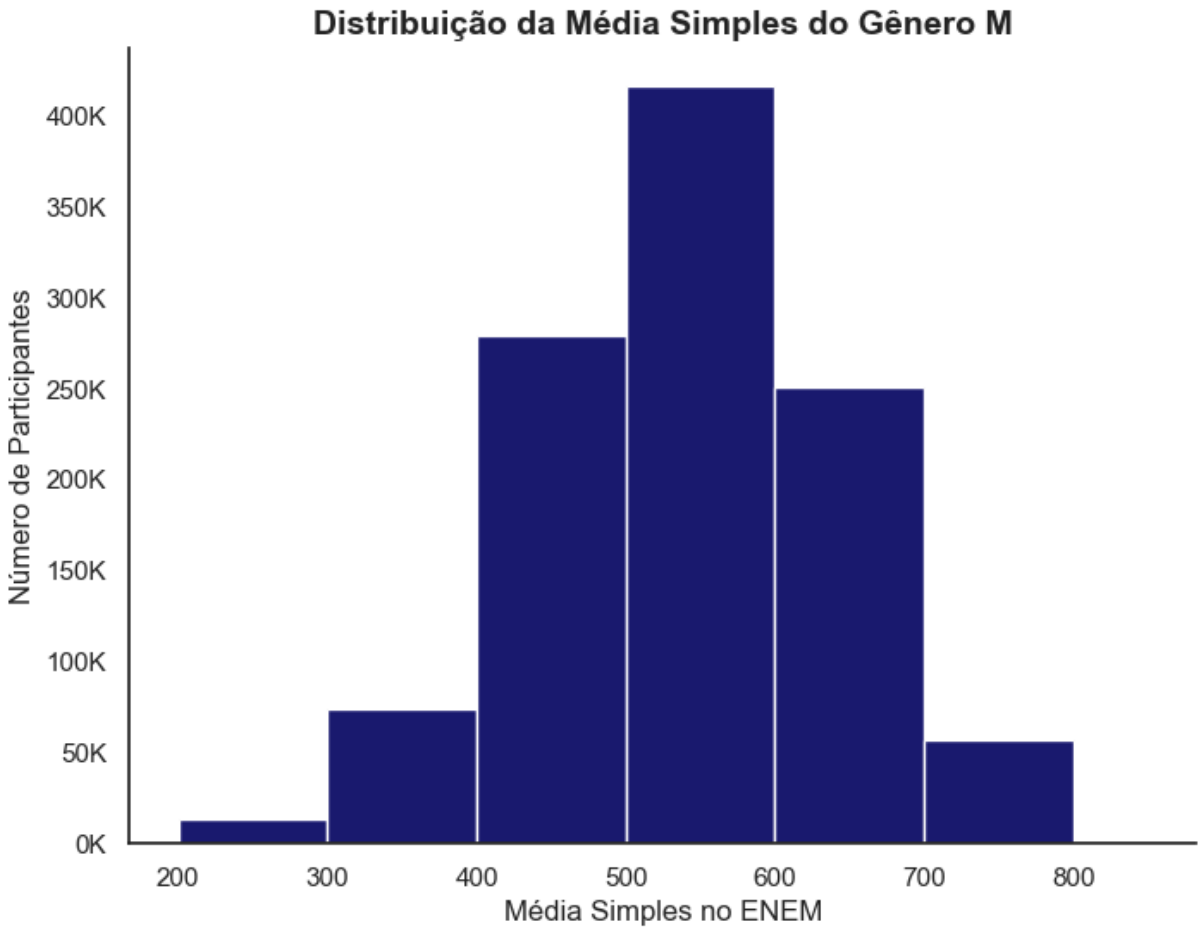
```
<Figure size 640x480 with 0 Axes>
```



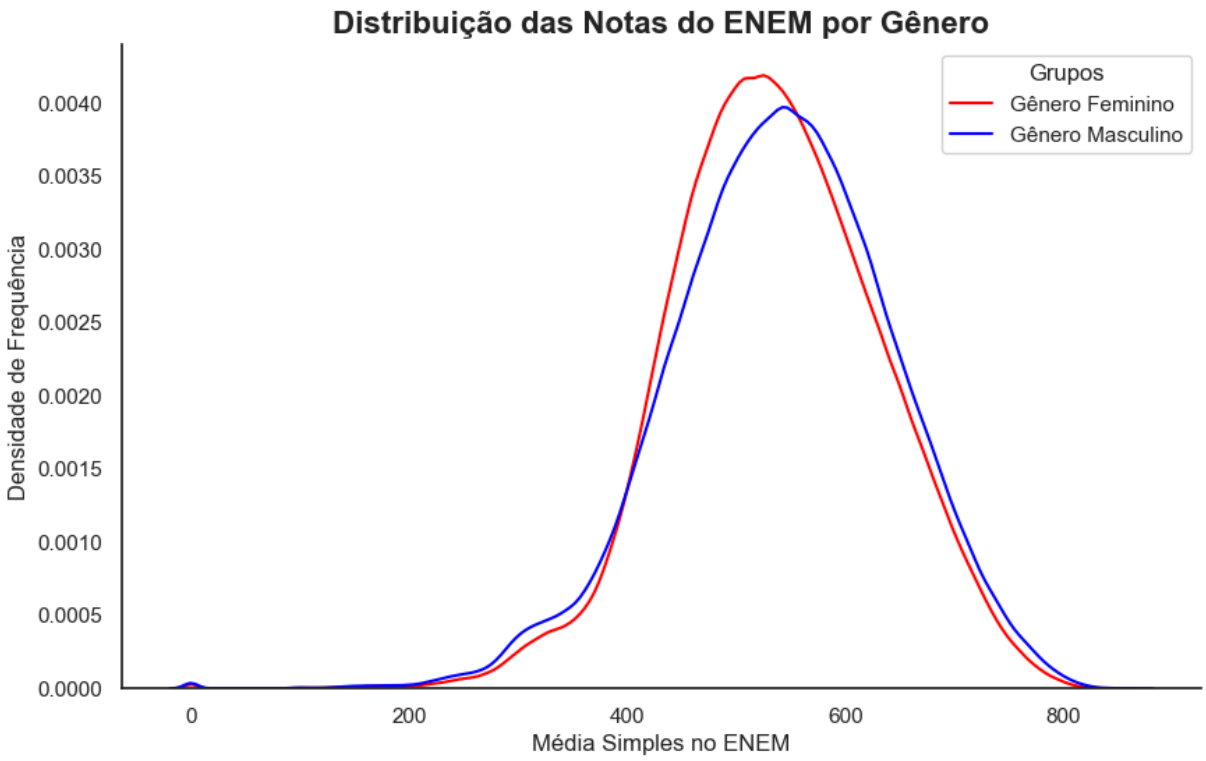
In [26]:
```
df_q = df[df['TP_SEXO'] == 'F']

sns.set_style('white')
bins = [200, 300, 400, 500, 600, 700, 800, 850]
plt.figure(figsize = (8,6))
plt.hist(df_q['media_notas'],
         bins = bins,
        color ='midnightblue')
plt.title('Distribuição da Média Simples do Gênero F', fontsize = 14, weight = 'bol
plt.xlabel("Média Simples no ENEM")
plt.ylabel('Número de Participantes')
# Ajustando o eixo Y para exibir números em notação de milhar
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show()
```



In [27]:
```
df_q = df[df['TP_SEXO'] == 'M']

sns.set_style('white')
bins = [200, 300, 400, 500, 600, 700, 800, 850]
plt.figure(figsize = (8,6))
plt.hist(df_q['media_notas'],
         bins = bins,
        color ='midnightblue')
plt.title('Distribuição da Média Simples do Gênero M', fontsize = 14, weight = 'bol
plt.xlabel("Média Simples no ENEM")
plt.ylabel('Número de Participantes')
# Ajustando o eixo Y para exibir números em notação de milhar
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show()
```



In [28]:
```
# Filtrando os dados para os três grupos A, I e Q
df_f = df[df['TP_SEXO'] == 'F']
df_m = df[df['TP_SEXO'] == 'M']

# Definindo o estilo do gráfico
sns.set_style('white')

# Criando a figura para o gráfico
plt.figure(figsize=(10, 6))

# Plotando as distribuições de notas para cada grupo
sns.kdeplot(df_f['media_notas'].to_numpy(), label='Gênero Feminino', color='red', f
sns.kdeplot(df_m['media_notas'].to_numpy(), label='Gênero Masculino', color='blue',

# Adicionando título e rótulos aos eixos
plt.title('Distribuição das Notas do ENEM por Gênero', fontsize=16, weight='bold')
plt.xlabel('Média Simples no ENEM', fontsize=12)
plt.ylabel('Densidade de Frequência', fontsize=12)

# Adicionando legenda
plt.legend(title='Grupos', loc='upper right')
sns.despine(top = True, right = True)

# Exibindo o gráfico
plt.show()
```



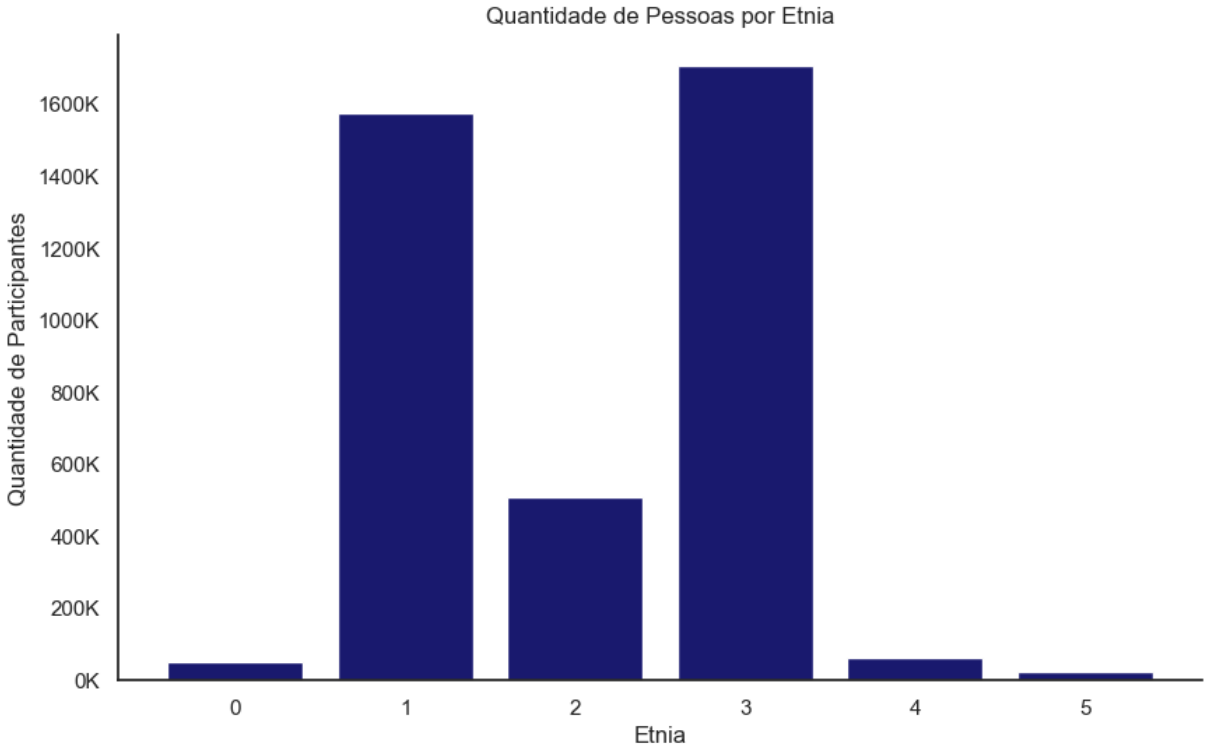
Não existe uma diferença significativa ao comparamos os dados de homens e mulheres nos dados.

Existe diferença de notas entre etnias?

(Estamos falando de racismo estrutural, não existe apoio para racismo científico aqui ok?)

In [29]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df['TP_COR_RACA'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Quantidade de Pessoas por Etnia')
plt.xlabel('Etnia')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```

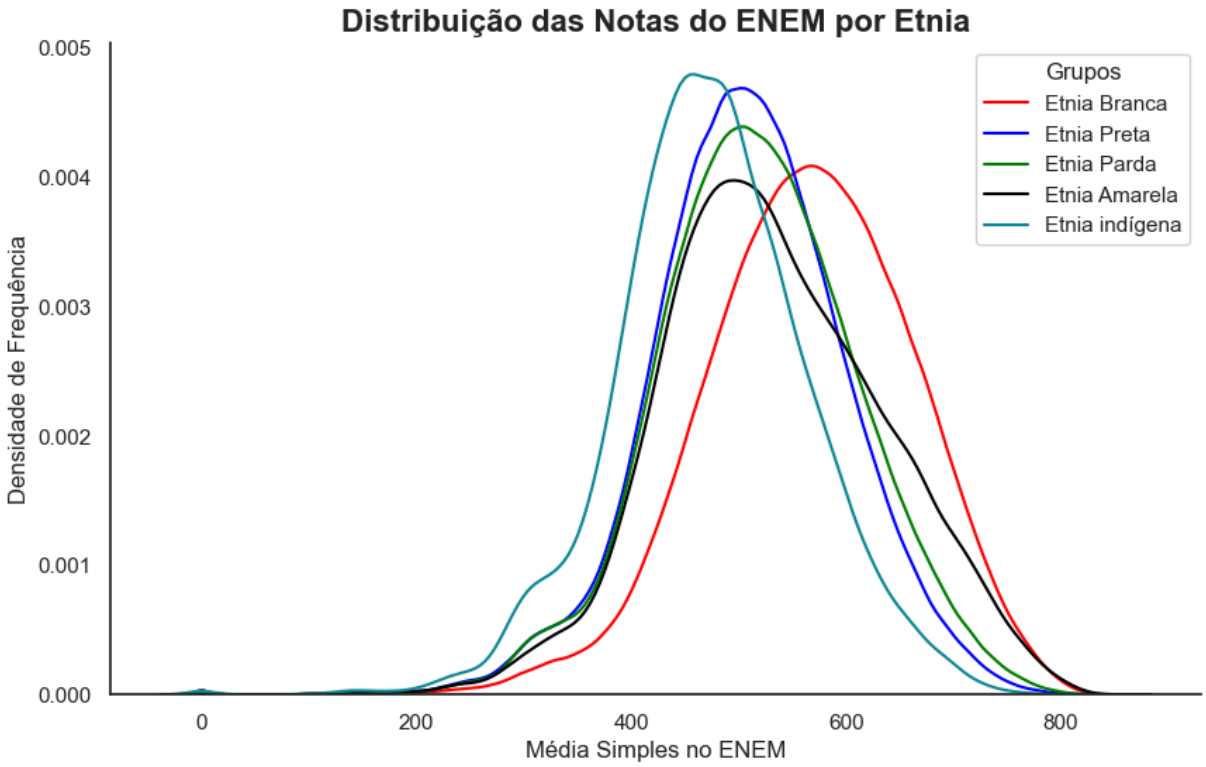


Etnias:

- 0: Não declarado
- 1: Branca
- 2: Preta
- 3: Parda
- 4: Amarela
- 5: Indígena
- 6: Não dispõe da informação

In [30]:
```
df_1 = df[df['TP_COR_RACA'] == 1]
df_2 = df[df['TP_COR_RACA'] == 2]
df_3 = df[df['TP_COR_RACA'] == 3]
df_4= df[df['TP_COR_RACA'] == 4]
df_5 = df[df['TP_COR_RACA'] == 5]

# Definindo o estilo do gráfico
sns.set_style('white')

# Criando a figura para o gráfico
plt.figure(figsize=(10, 6))

# Plotando as distribuições de notas para cada grupo
sns.kdeplot(df_1['media_notas'].to_numpy(), label='Etnia Branca', color='red', fill
sns.kdeplot(df_2['media_notas'].to_numpy(), label='Etnia Preta', color='blue', fill
sns.kdeplot(df_3['media_notas'].to_numpy(), label='Etnia Parda', color='green', fil
sns.kdeplot(df_4['media_notas'].to_numpy(), label='Etnia Amarela', color='black', f
sns.kdeplot(df_5['media_notas'].to_numpy(), label='Etnia indígena', color='#108A99'

# Adicionando título e rótulos aos eixos
```

```
plt.title('Distribuição das Notas do ENEM por Etnia', fontsize=16, weight='bold')
plt.xlabel('Média Simples no ENEM', fontsize=12)
plt.ylabel('Densidade de Frequência', fontsize=12)

# Adicionando legenda
plt.legend(title='Grupos', loc='upper right')
sns.despine(top = True, right = True)

# Exibindo o gráfico
plt.show()
```



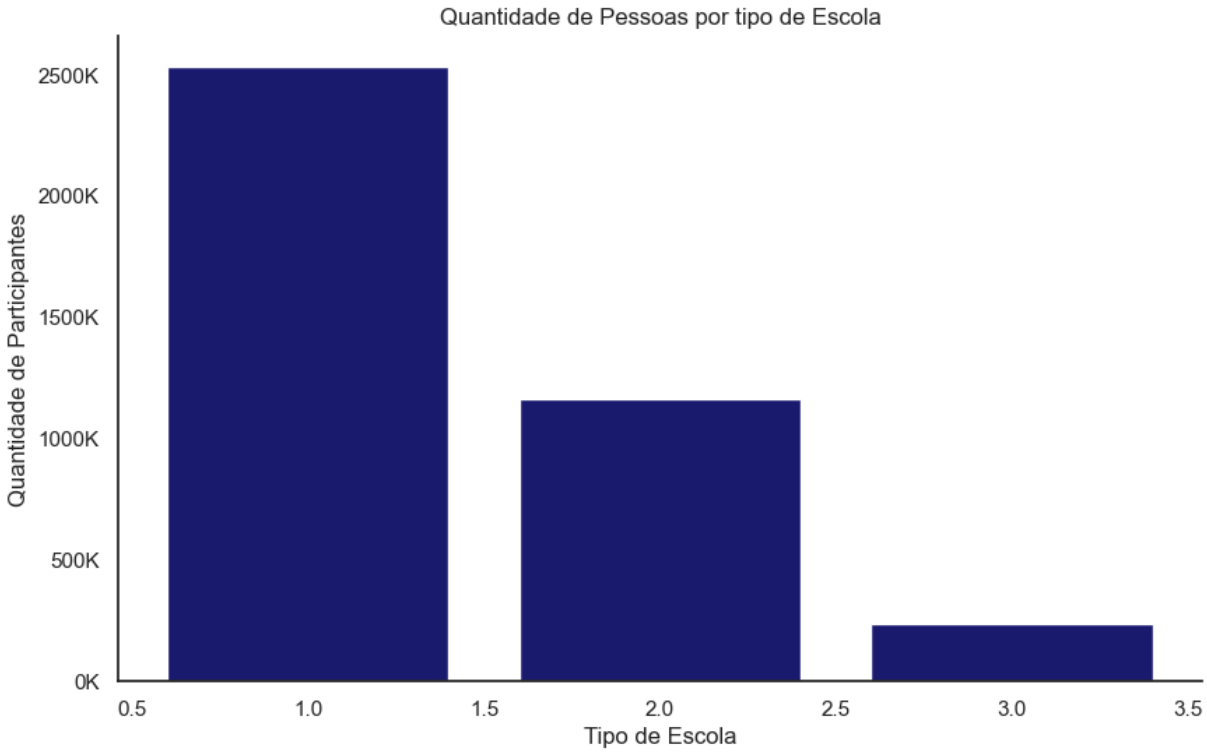
Fica claro que pessoas com etnia branca têm uma tendência maior a irem melhor no ENEM, e isso acaba por refletir a realidade dos estudantes brasileiros, enquanto o grupo com pior desempenho foi o de etnia indígena, indicando a falta de estrutura para essa etnia. Novamente, tratam-se de dados, essa análise não tem como objetivo reforçar nenhuma ideia relacionada a racismo, apenas mostra a realidade desigual enfrentada.

Existe correlação de notas por escola pública/privada?

In [31]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df['TP_ESCOLA'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
```

```
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Quantidade de Pessoas por tipo de Escola')
plt.xlabel('Tipo de Escola')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```

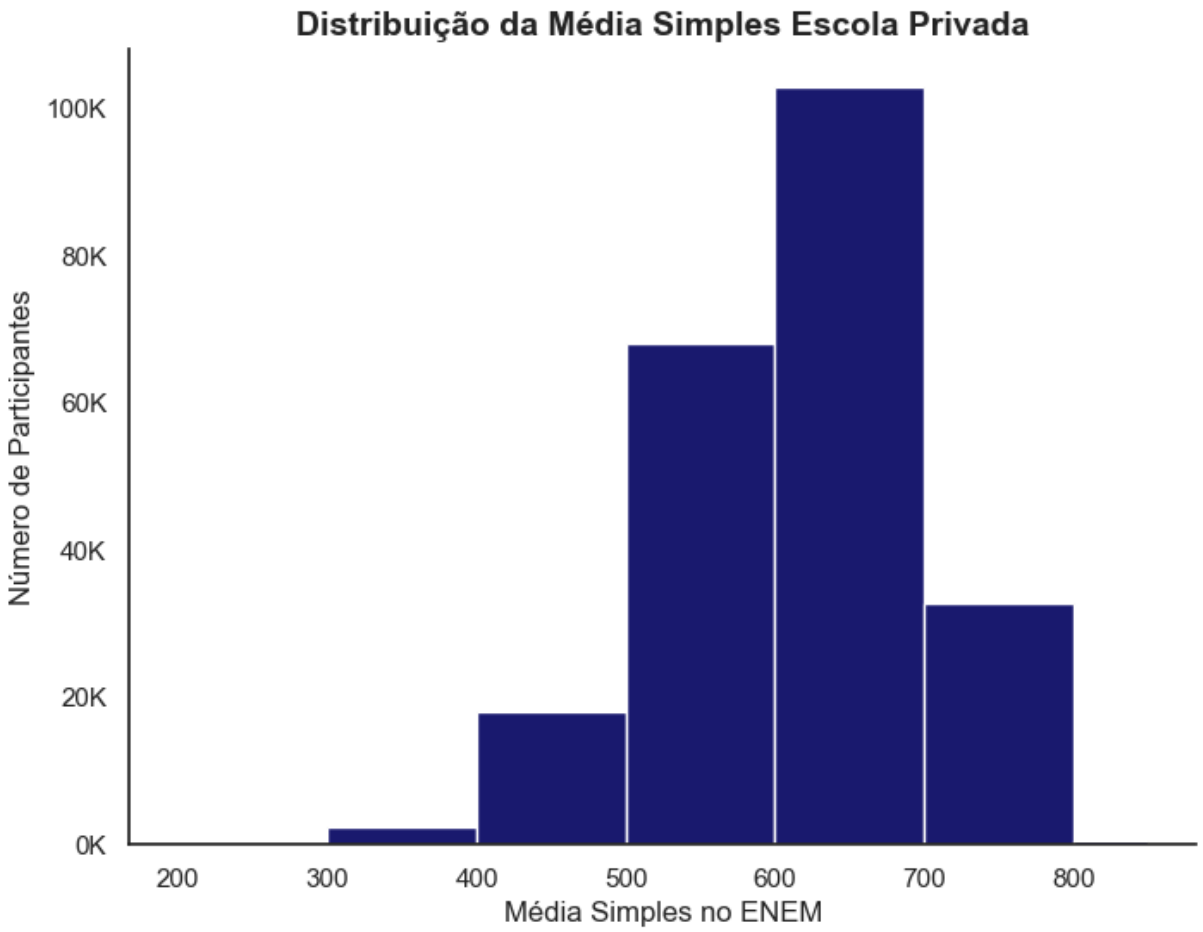


Categorias:

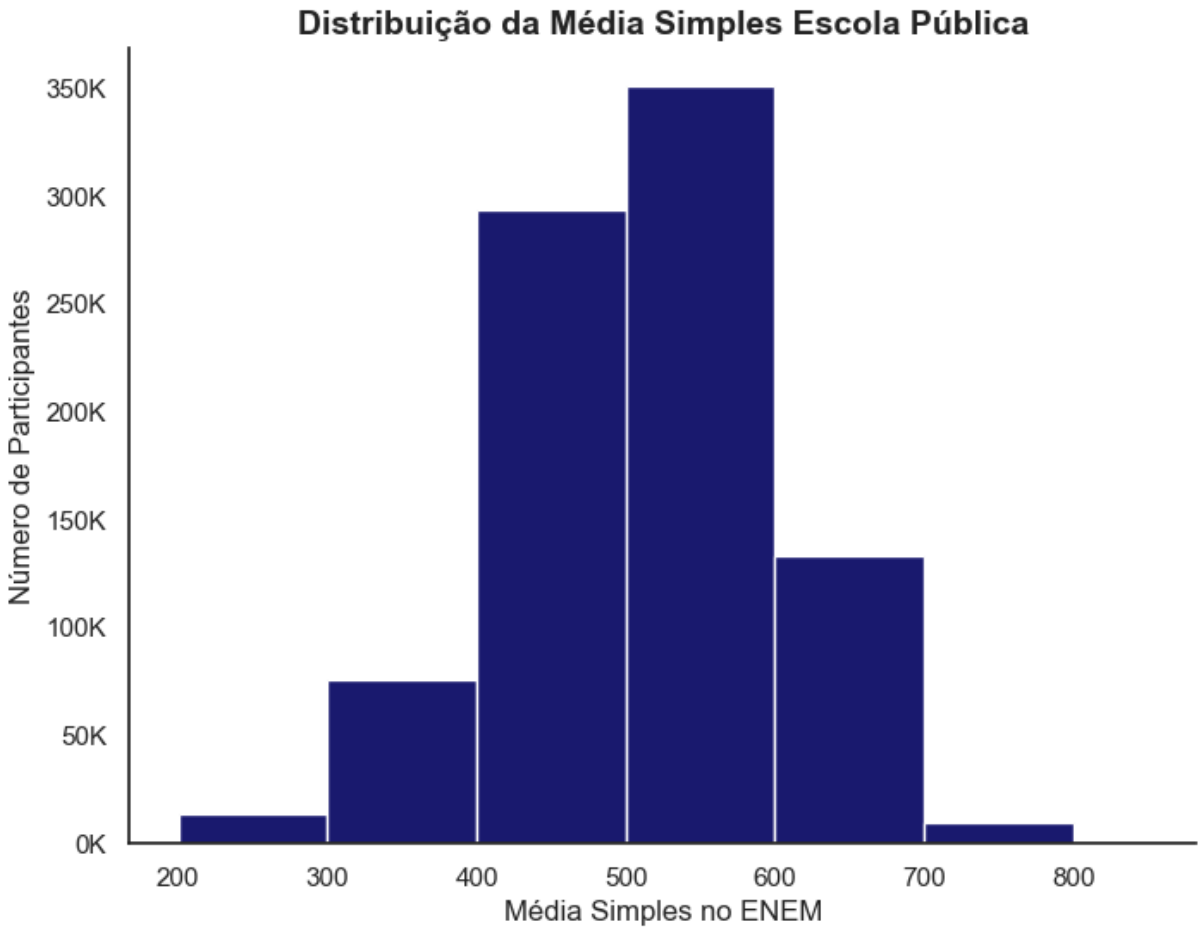
- 1: Não Respondeu
- 2: Pública
- 3: Privada

In [32]:
```
df_q = df[df['TP_ESCOLA'] == 3]

sns.set_style('white')
bins = [200, 300, 400, 500, 600, 700, 800, 850]
plt.figure(figsize = (8,6))
plt.hist(df_q['media_notas'],
         bins = bins,
        color ='midnightblue')
plt.title('Distribuição da Média Simples Escola Privada', fontsize = 14, weight = '
plt.xlabel("Média Simples no ENEM")
plt.ylabel('Número de Participantes')
# Ajustando o eixo Y para exibir números em notação de milhar
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
```

```
sns.despine(top = True, right = True)
plt.show()
```



In [33]:
```
df_q = df[df['TP_ESCOLA'] == 2]

sns.set_style('white')
bins = [200, 300, 400, 500, 600, 700, 800, 850]
plt.figure(figsize = (8,6))
plt.hist(df_q['media_notas'],
         bins = bins,
        color ='midnightblue')
plt.title('Distribuição da Média Simples Escola Pública', fontsize = 14, weight = '
plt.xlabel("Média Simples no ENEM")
plt.ylabel('Número de Participantes')
# Ajustando o eixo Y para exibir números em notação de milhar
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show()
```



Existe uma diferença bem clara na distribuição de notas dos estudantes. Pessoas que fizeram seu ensino médio em escolas privadas tem uma tendência a terem melhores notas em comparação à aqueles que fizeram em escola pública.

Qual o perfil das pessoas que tiraram acima de 750 no ENEM?

In [34]:
```
df_750 = df[df['media_notas'] >= 750]
df_750
```

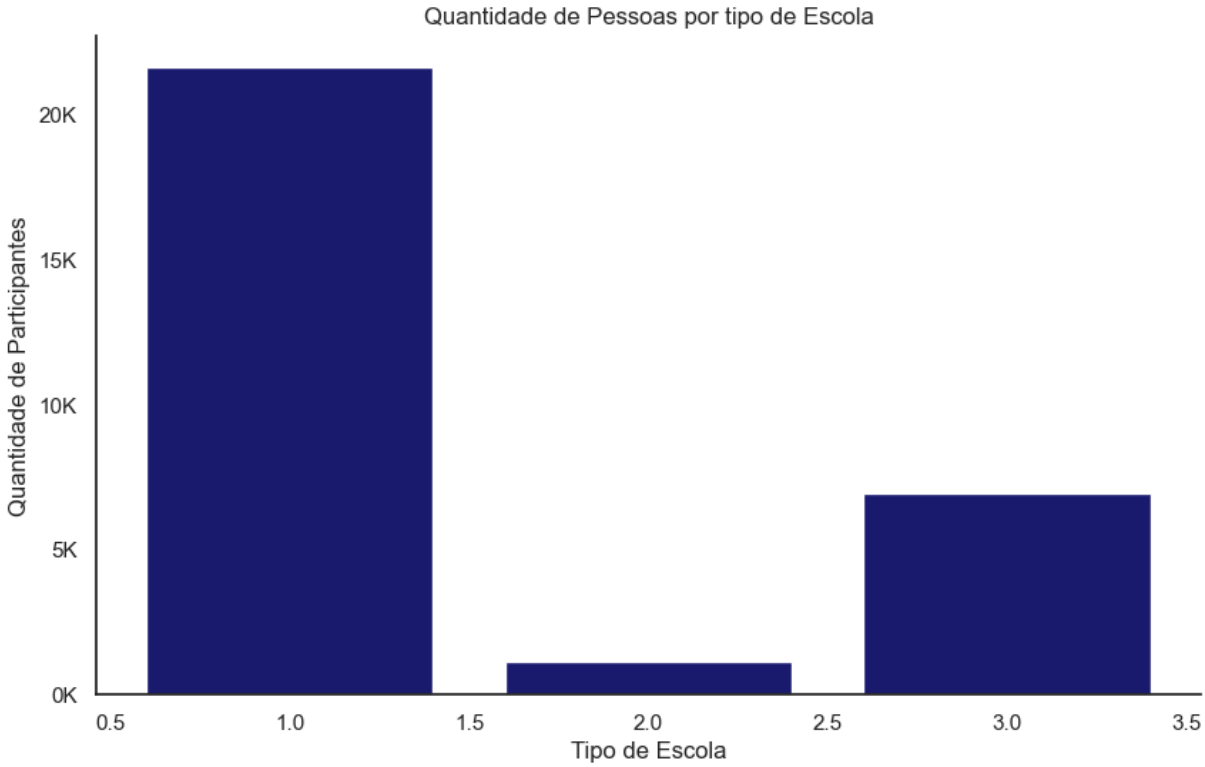
Out[34]:

	TP_FAIXA_ETARIA	TP_SEXO	TP_COR_RACA	TP_ESCOLA	SG_UF_ESC	TP_DEPENDE
137	9	F	3	1	NaN	
646	3	F	1	1	NaN	
701	7	F	1	1	NaN	
707	4	M	1	1	NaN	
712	4	M	3	1	NaN	
...	...	...	...	...	...	
3930111	4	M	2	1	NaN	
3930797	4	M	1	1	NaN	
3930919	7	F	1	1	NaN	
3931336	6	F	1	1	NaN	
3933836	6	F	1	1	NaN	

29724 rows × 15 columns

In [35]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df_750['TP_ESCOLA'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Quantidade de Pessoas por tipo de Escola')
plt.xlabel('Tipo de Escola')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```

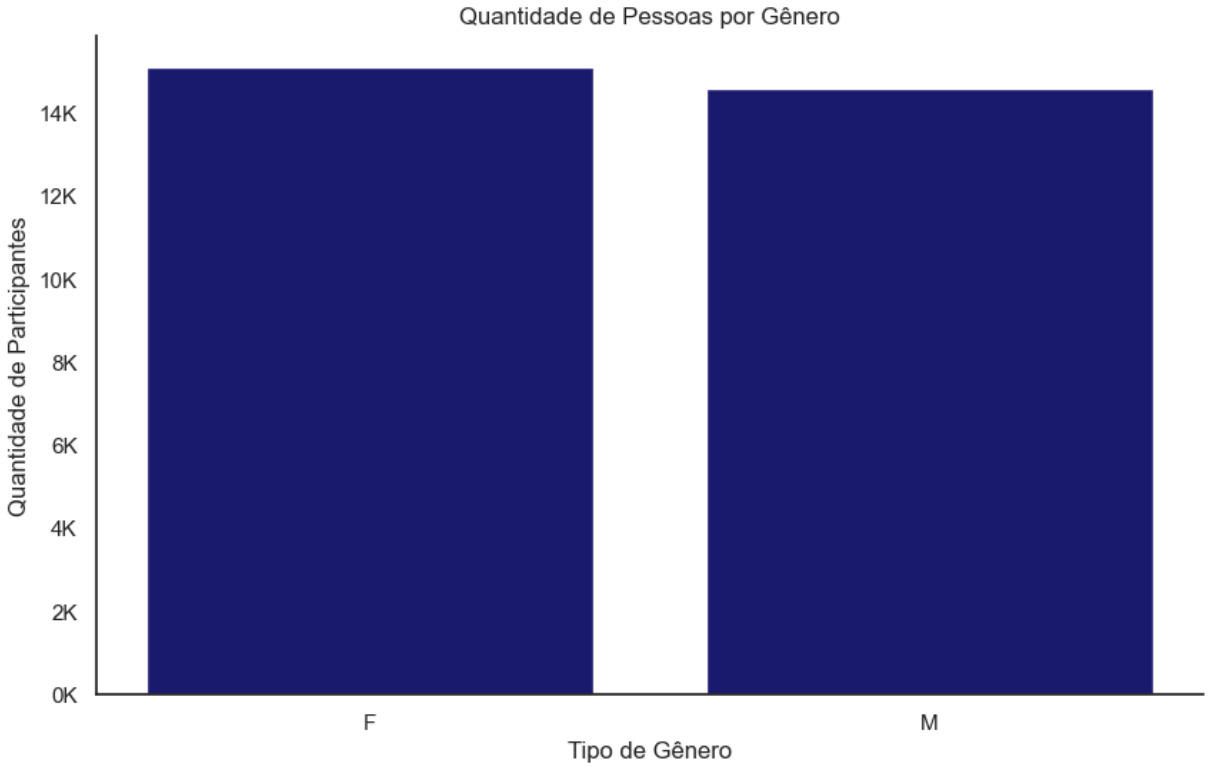


Categorias:

- 1: Não Respondeu
- 2: Pública
- 3: Privada

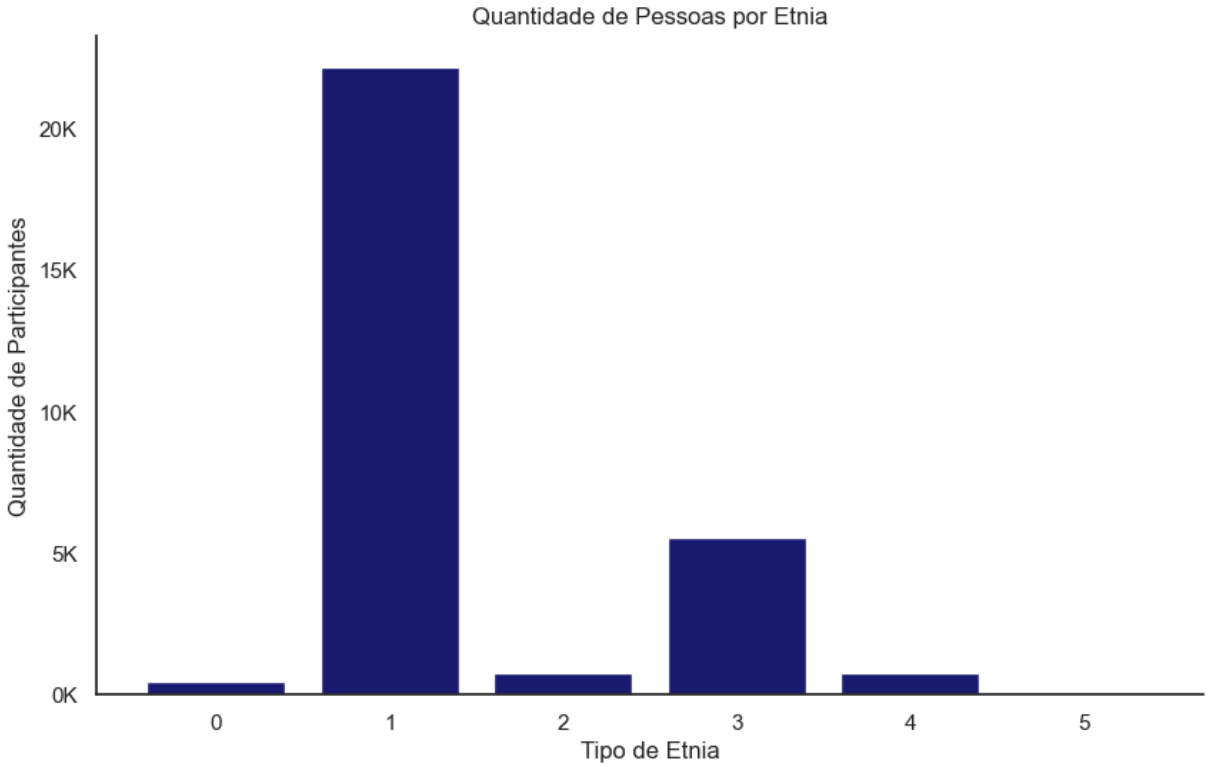
In [36]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df_750['TP_SEXO'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Quantidade de Pessoas por Gênero')
plt.xlabel('Tipo de Gênero')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```



In [37]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df_750['TP_COR_RACA'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Quantidade de Pessoas por Etnia')
plt.xlabel('Tipo de Etnia')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```

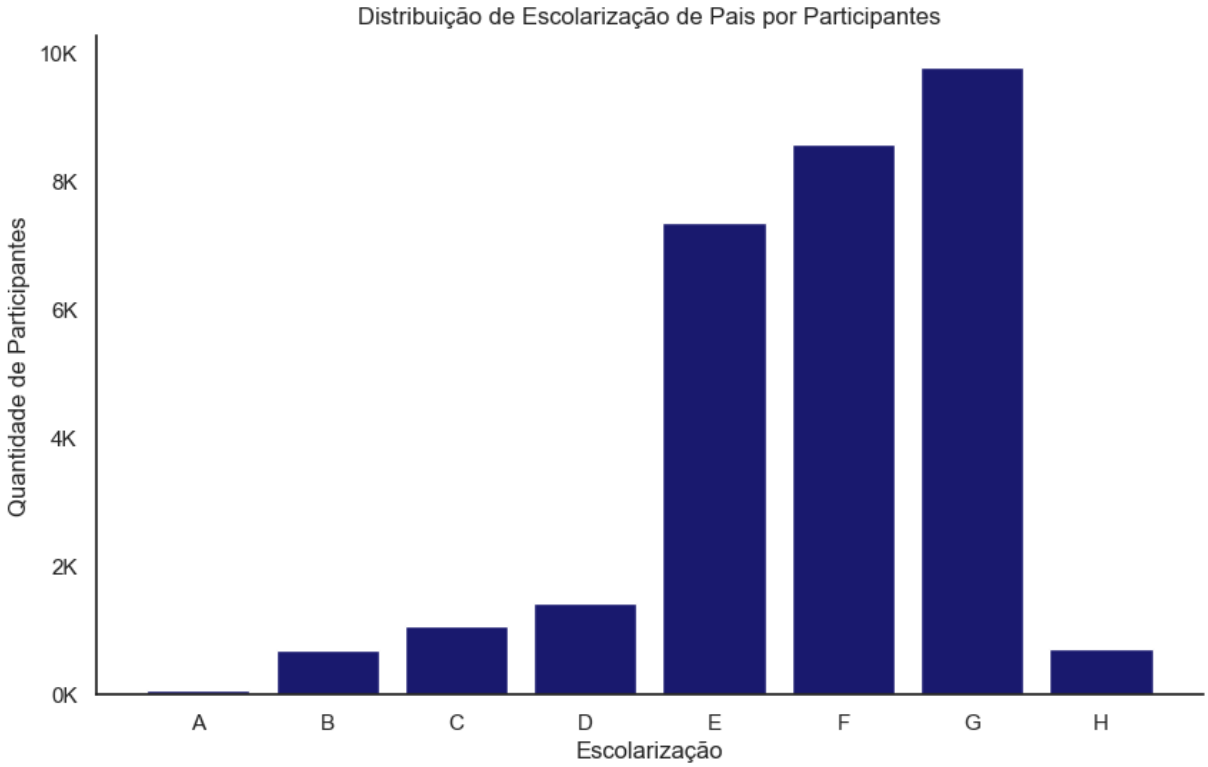


Etnias:

- 0: Não declarado
- 1: Branca
- 2: Preta
- 3: Parda
- 4: Amarela
- 5: Indígena
- 6: Não dispõe da informação

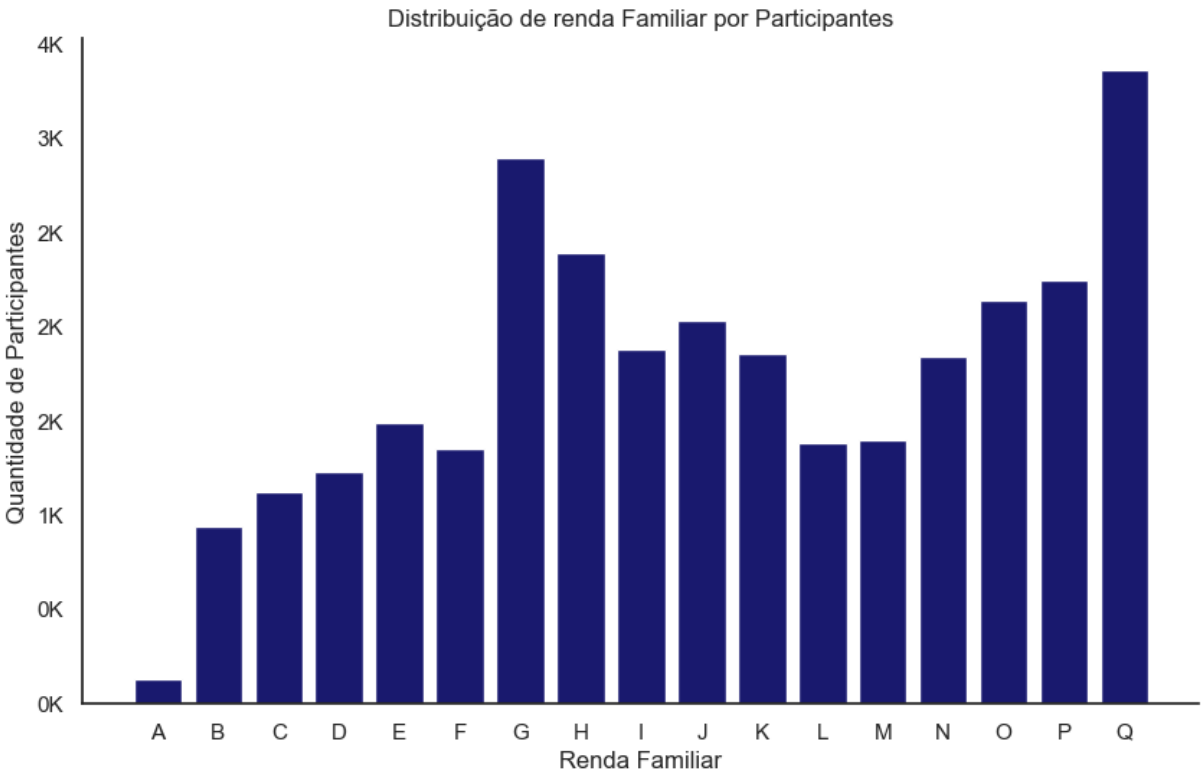
In [38]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df_750['Q001'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Distribuição de Escolarização de Pais por Participantes')
plt.xlabel('Escolarização')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```



In [39]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df_750['Q006'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Distribuição de renda Familiar por Participantes')
plt.xlabel('Renda Familiar')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```



Faixas de Renda Familiar

A: Nenhuma Renda

B: Até R$ 1.320,00

C: De R$1.320,01 até R$ 1.980,00

D: De R$1.980,01 até R$ 2.640,00

E: De R$2.640,01 até R$ 3.300,00

F: De R$3.300,01 até R$ 3.960,00

G: De R$3.960,01 até R$ 5.280,00

H: De R$5.280,01 até R$ 6.600,00

I: De R$6.600,01 até R$ 7.920,00

J: De R$7.920,01 até R$ 9.240,00

K: De R$9.240,01 até R$ 10.560,00

L: De R$10.560,01 até R$ 11.880,00

M: De R$11.880,01 até R$ 13.200,00

N: De R$13.200,01 até R$ 15.840,00

O: De R$15.840,01 até R$ 19.800,00

P: De R$19.800,01 até R$ 26.400,00

Q: Acima de R$ 26.400,00

Qual o perfil das pessoas que tiraram abaixo de 500 no ENEM?

In [40]:
```
df_500 = df[df['media_notas'] <= 500]
df_500
```

Out[40]:

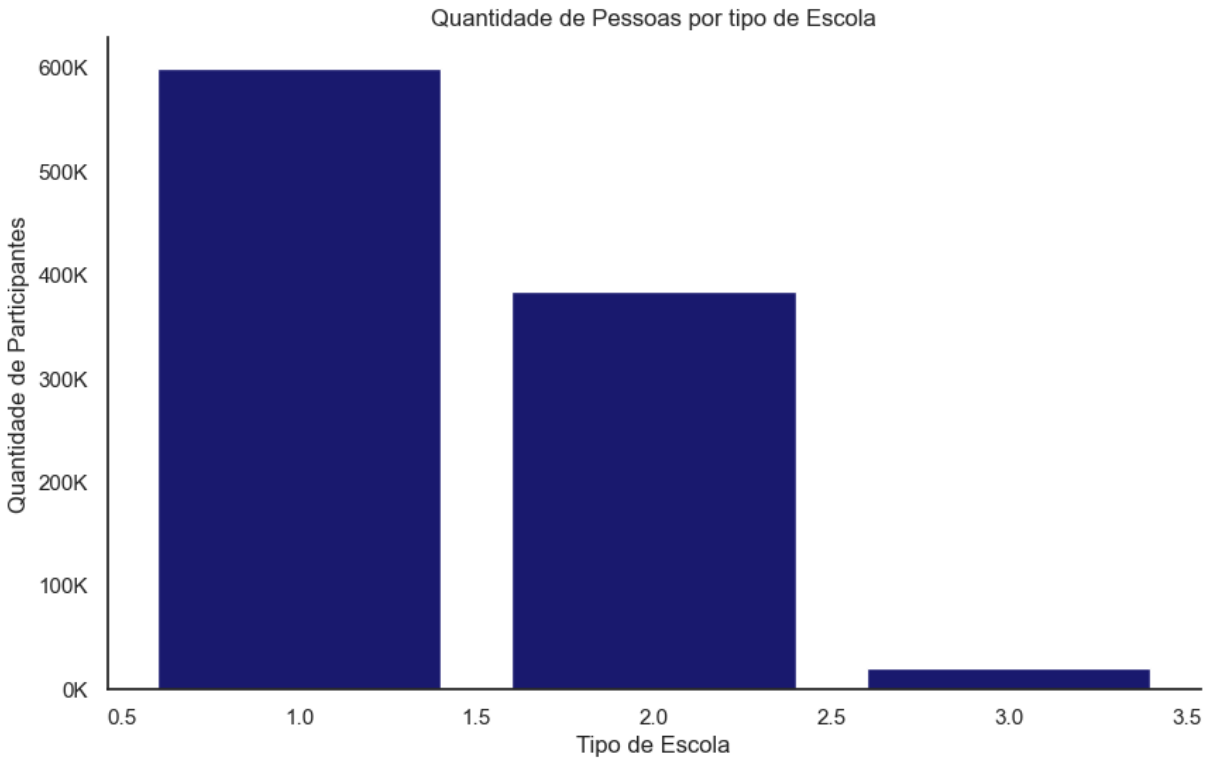
	TP_FAIXA_ETARIA	TP_SEXO	TP_COR_RACA	TP_ESCOLA	SG_UF_ESC	TP_DEPENDE
4	3	F	3	2	CE	
15	7	F	3	1	NaN	
17	11	F	1	1	NaN	
25	8	F	3	1	NaN	
26	4	F	3	1	NaN	
...	...	...	...	...	...	
3933920	2	F	1	2	NaN	
3933921	3	M	1	2	RS	
3933930	19	M	1	1	NaN	
3933936	3	M	1	2	RS	
3933939	15	M	1	1	NaN	

1004483 rows × 15 columns

In [41]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df_500['TP_ESCOLA'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Quantidade de Pessoas por tipo de Escola')
plt.xlabel('Tipo de Escola')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
```

```
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```

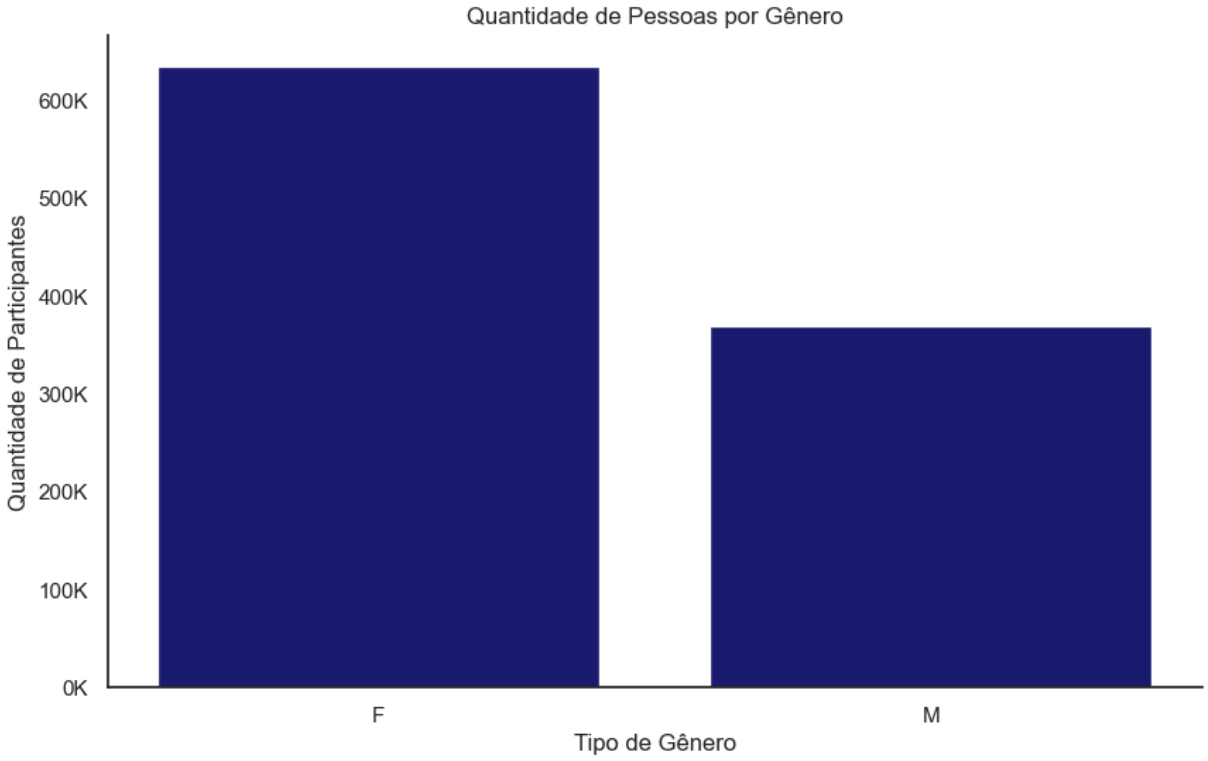


Categorias:

- 1: Não Respondeu
- 2: Pública
- 3: Privada

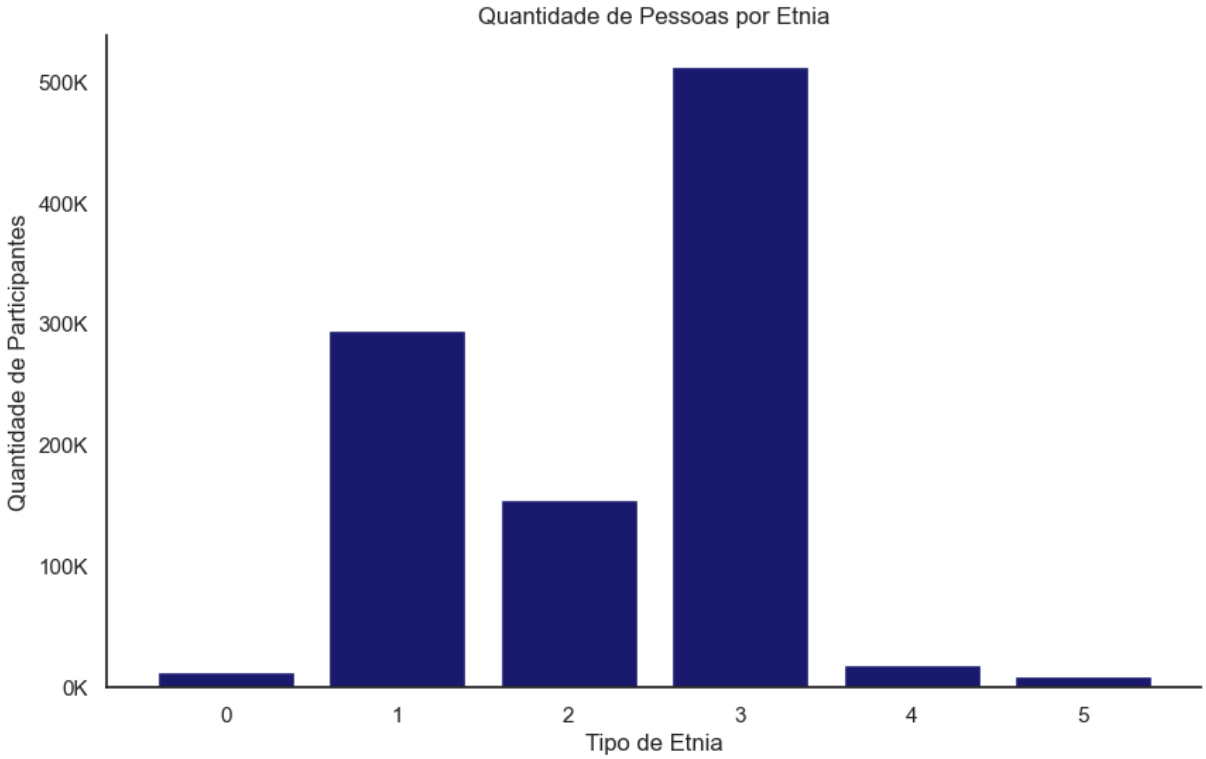
In [42]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df_500['TP_SEXO'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Quantidade de Pessoas por Gênero')
plt.xlabel('Tipo de Gênero')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```



In [43]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df_500['TP_COR_RACA'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Quantidade de Pessoas por Etnia')
plt.xlabel('Tipo de Etnia')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```

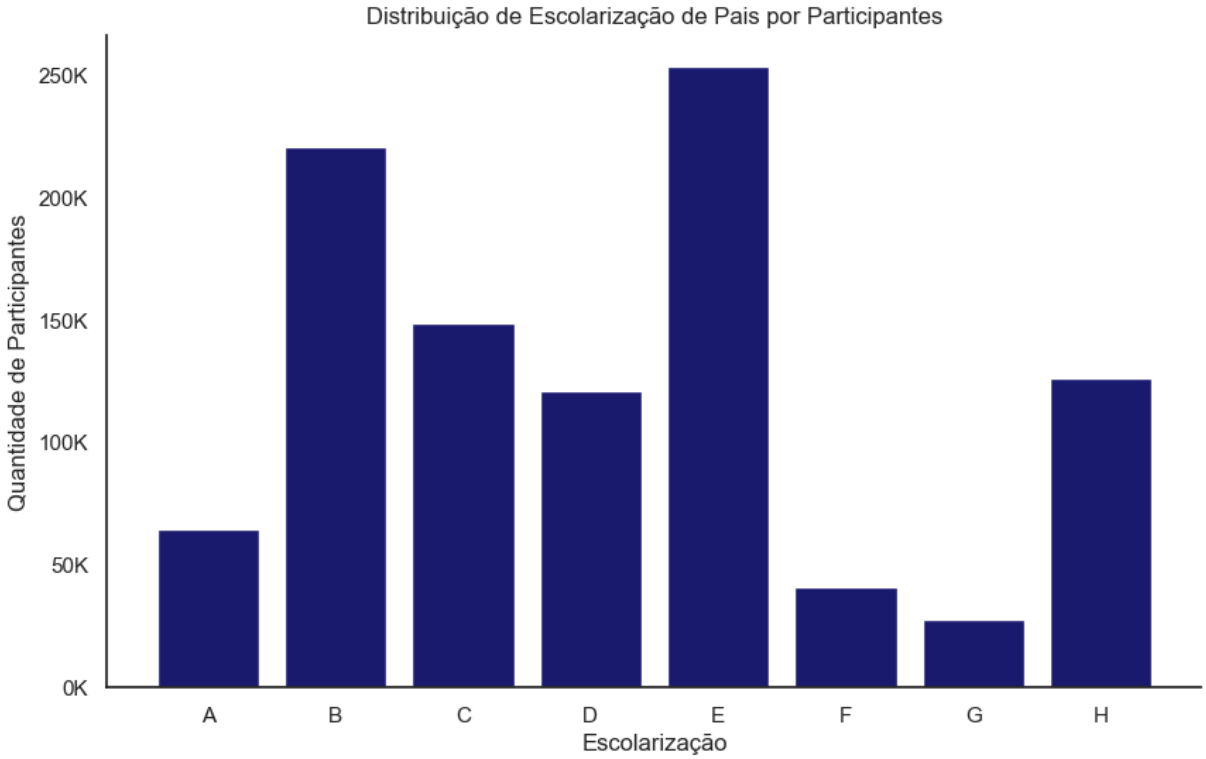


Etnias:

- 0: Não declarado
- 1: Branca
- 2: Preta
- 3: Parda
- 4: Amarela
- 5: Indígena
- 6: Não dispõe da informação

In [44]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df_500['Q001'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Distribuição de Escolarização de Pais por Participantes')
plt.xlabel('Escolarização')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```



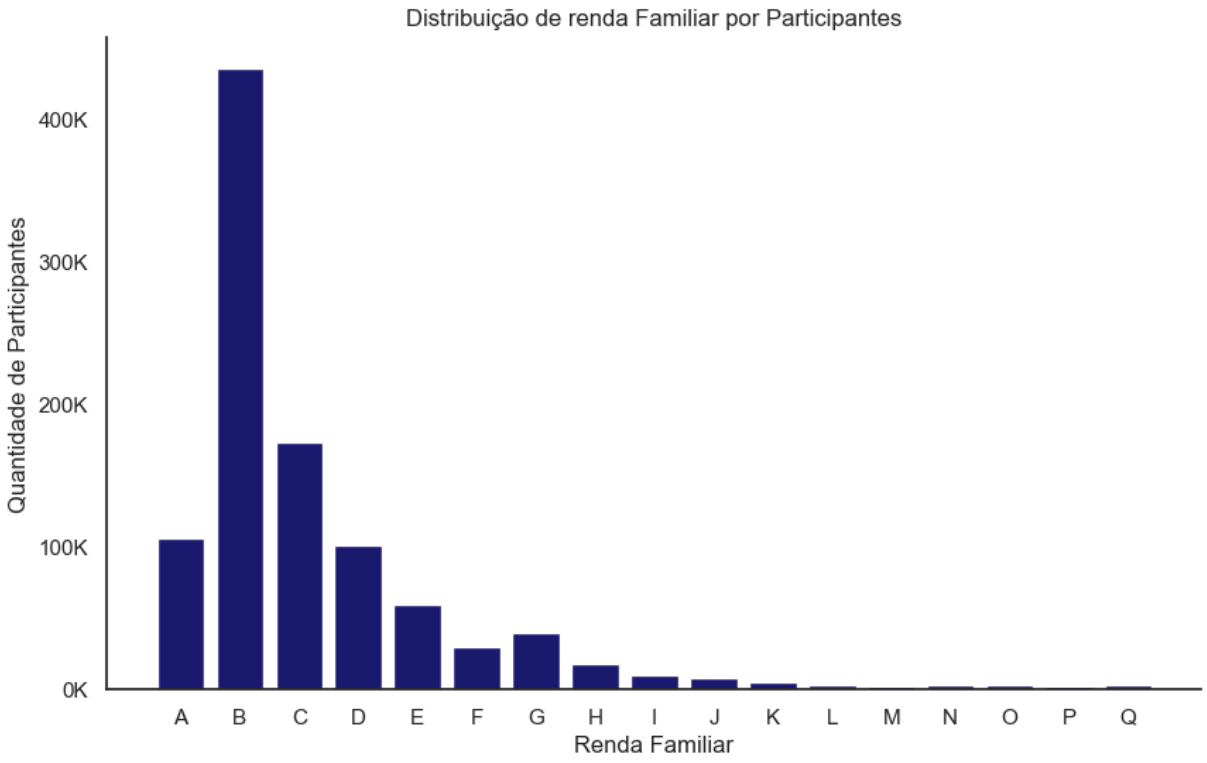
Categorias de Escolarização

- A: Nunca estudou.
- B: Não completou a 4ª série/5º ano do Ensino Fundamental.
- C: Completou a 4ª série/5º ano, mas não completou a 8ª série/9º ano do Ensino Fundamental.
- D: Completou a 8ª série/9º ano do Ensino Fundamental, mas não completou o Ensino Médio.
- E: Completou o Ensino Médio, mas não completou a Faculdade.
- F: Completou a Faculdade, mas não completou a Pós-graduação.
- G: Completou a Pós-graduação.
- H: Não sei.

In [45]:
```
# Contando as ocorrências das categorias usando numpy
unique, counts = np.unique(df_500['Q006'], return_counts=True)

# Definindo o estilo do gráfico
sns.set_style('white')
sns.despine()

# Criando o gráfico de colunas com matplotlib
plt.figure(figsize=(10, 6))
plt.bar(unique, counts, color='midnightblue')
plt.title('Distribuição de renda Familiar por Participantes')
plt.xlabel('Renda Familiar')
plt.ylabel('Quantidade de Participantes')
plt.xticks(rotation=0)
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '%1.0fK' %
```

```
sns.despine(top = True, right = True)
plt.show();
```

```
<Figure size 640x480 with 0 Axes>
```



Faixas de Renda Familiar

A: Nenhuma Renda

B: Até R$ 1.320,00

C: De R$1.320,01 até R$ 1.980,00

D: De R$1.980,01 até R$ 2.640,00

E: De R$2.640,01 até R$ 3.300,00

F: De R$3.300,01 até R$ 3.960,00

G: De R$3.960,01 até R$ 5.280,00

H: De R$5.280,01 até R$ 6.600,00

I: De R$6.600,01 até R$ 7.920,00

J: De R$7.920,01 até R$ 9.240,00

K: De R$9.240,01 até R$ 10.560,00

L: De R$10.560,01 até R$ 11.880,00

M: De R$11.880,01 até R$ 13.200,00

N: De R$13.200,01 até R$ 15.840,00

O: De R$15.840,01 até R$ 19.800,00

P: De R$19.800,01 até R$ 26.400,00

Q: Acima de R$ 26.400,00

Considerações Finais

Impacto da Renda Familiar

Famílias com melhores rendas tendem a apresentar filhos com melhores notas no ENEM.

Escolaridade dos Pais

O nível de escolaridade do pai influencia significativamente a nota do filho no ENEM.

- Diferenças são mais pronunciadas entre pais com ensino médio ou inferior.
- Pouca diferença é observada entre pais com graduação e pós-graduação.

Impacto do Gênero

Não existe uma diferença significativa de desempenho entre candidatos do gênero masculino e feminino.

Impacto da Etnia

A análise revelou que:

- Candidatos brancos apresentaram as melhores notas, em média.
- Candidatos indígenas tiveram as menores notas.
- Candidatos de outras etnias ficaram em níveis intermediários.

Tipo de Escola

Estudantes de escolas privadas obtiveram resultados significativamente melhores do que os de escolas públicas.

Perfil: Notas Acima de 800

Pessoas que alcançaram notas acima de 800 apresentaram as seguintes características:

- Frequentaram escolas privadas.
- São, majoritariamente, homens.
- Pertencem, em sua maioria, às etnias branca ou parda.
- Tiveram pais com escolaridade no ensino médio, graduação ou pós-graduação.
- Possuem uma maior distribuição de renda familiar.

Perfil: Notas Abaixo de 500

Pessoas que alcançaram notas abaixo de 500 apresentaram as seguintes características:

- Frequentaram escolas públicas.
- São, majoritariamente, mulheres.
- Pertencem, em sua maioria, às etnias parda ou branca.
- Tiveram pais com escolaridade concentrada no ensino médio ou inferior.
- Possuem uma menor distribuição de renda familiar.

In [47]:
```
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
x = data[['SAT', 'Rand 1,2,3']]
y = data['GPA']
```