CURSO DE ESTATÍSTICA - PARTE 1

1 CONHECENDO OS DADOS

1.1 Dataset do projeto

Pesquisa Nacional por Amostra de Domicílios - 2015

A **Pesquisa Nacional por Amostra de Domicílios - PNAD** investiga anualmente, de forma permanente, características gerais da população, de educação, trabalho, rendimento e habitação e outras, com periodicidade variável, de acordo com as necessidades de informação para o país, como as características sobre migração, fecundidade, nupcialidade, saúde, segurança alimentar, entre outros temas. O levantamento dessas estatísticas constitui, ao longo dos 49 anos de realização da pesquisa, um importante instrumento para formulação, validação e avaliação de políticas orientadas para o desenvolvimento socioeconômico e a melhoria das condições de vida no Brasil.

Fonte dos Dados

https://ww2.ibge.gov.br/home/estatistica/populacao/trabalhoerendimento/pnad2015/microdados.shtm

Variáveis utilizadas

> ##### Renda

Rendimento mensal do trabalho principal para pessoas de 10 anos ou mais de idade.

> ##### Idade

Idade do morador na data de referência em anos.

> ##### Altura (elaboração própria)

Altura do morador em metros.

> ##### UF

Código	Descrição
11	Rondônia
12	Acre
13	Amazonas

Código	Descrição
14	Roraima
15	Pará
16	Amapá
17	Tocantins
21	Maranhão
22	Piauí
23	Ceará
24	Rio Grande do Norte
25	Paraíba
26	Pernambuco
27	Alagoas
28	Sergipe
29	Bahia
31	Minas Gerais
32	Espírito Santo
33	Rio de Janeiro
35	São Paulo
41	Paraná
42	Santa Catarina
43	Rio Grande do Sul
50	Mato Grosso do Sul
51	Mato Grosso
52	Goiás
53	Distrito Federal

Sexo

Código	Descrição
0	Masculino
1	Feminino

Anos de Estudo

Código	Descrição
1	Sem instrução e menos de 1 ano
2	1 ano
3	2 anos
4	3 anos
5	4 anos
6	5 anos

Código	Descrição
7	6 anos
8	7 anos
9	8 anos
10	9 anos
11	10 anos
12	11 anos
13	12 anos
14	13 anos
15	14 anos
16	15 anos ou mais
17	Não determinados
	Não aplicável

> ## Cor

Código	Descrição
0	Indígena
2	Branca
4	Preta
6	Amarela
8	Parda
9	Sem declaração

Observação

> Os seguintes tratamentos foram realizados nos dados originais:
>
> 1. Foram eliminados os registros onde a **Renda** era inválida (999 999 999 999);
> 2. Foram eliminados os registros onde a **Renda** era missing;
> 3. Foram considerados somente os registros das **Pessoas de Referência** de cada domicílio (responsável pelo domicílio).

Importando pandas e lendo o dataset do projeto

https://pandas.pydata.org/

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
dados = pd.read_csv('dados.csv')
```

In [3]:
```
dados.head(10)
```


	Sexo	Idade	Cor	Anos de Estudo	Renda	Altura

	UF	Sexo	Idade	Cor	Anos de Estudo	Renda	Altura
0	11	0	23	8	12	800	1.603808
1	11	1	23	2	12	1150	1.739790
2	11	1	35	8	15	880	1.760444
3	11	0	46	2	6	3500	1.783158
4	11	1	47	8	9	150	1.690631
5	11	1	34	8	12	790	1.637906
6	11	0	57	8	12	3150	1.570078
7	11	1	60	8	12	1700	1.608495
8	11	1	50	4	14	1800	1.780329
9	11	0	26	8	12	1150	1.793203

1.2 Tipos de dados

Variáveis qualitativas ordinais

► Variáveis que podem ser ordenadas ou hierarquizardas

In [4]:
```
sorted(dados['Anos de Estudo'].unique())
```

Out[4]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
```

Variáveis qualitativas nominais

► Variáveis que não podem ser ordenadas ou hierarquizardas

In [5]:
```
sorted(dados['Sexo'].unique())
```

Out[5]:
```
[0, 1]
```

In [6]:
```
sorted(dados['Cor'].unique())
```

Out[6]:
```
[0, 2, 4, 6, 8]
```

In [7]:
```
sorted(dados['UF'].unique())
```

Out[7]:
```
[11,
 12,
 13,
 14,
 15,
 16,
 17,
 21,
 22,
 23,
 24,
 25,
 26,
 27,
```

```
31,
32,
33,
35,
41,
42,
43,
50,
51,
52,
53]
```

Variáveis quantitativas discretas

► Variáveis que representam uma contagem onde os valores possíveis formam um conjunto finito ou enumerável.

In [8]:
```
print(f"De {dados.Idade.min()} até {dados.Idade.max()} anos")
```

```
De 13 até 99 anos
```

Observação

> A variável idade pode ser classificada de três formas distintas:
>
> 1. **QUANTITATIVA DISCRETA** - quando representa anos completos (números inteiros);
> 2. **QUANTITATIVA CONTÍNUA** - quando representa a idade exata, sendo representado por frações de anos; e
> 3. **QUALITATIVA ORDINAL** - quando representa faixas de idade.

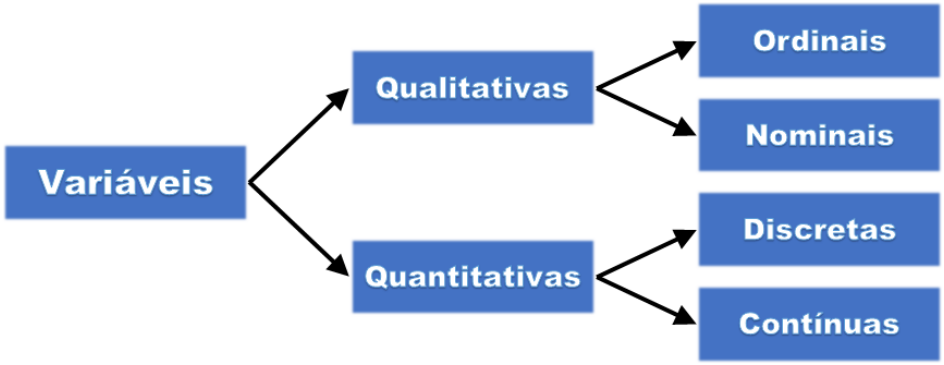
Variáveis quantitativas contínuas

► Variáveis que representam uma contagem ou mensuração que assumem valores em uma escala contínua (números reais).

In [9]:
```
print(f"De {dados.Altura.min()} até {dados.Altura.max()} metros")
```

```
De 1.339244614 até 2.028496765 metros
```

Classificação de uma variável



2 DISTRIBUIÇÃO DE FREQUÊNCIAS

O primeiro passo em um trabalho de análise é o conhecimento do comportamento das variáveis envolvidas no estudo. Utilizando técnicas estatísticas como as análises das **DISTRIBUIÇÕES DE FREQUÊNCIAS** e **RAMAS** podemos avaliar melhor a forma como os fenômenos em estudo se distribuem.

2.1 Distribuição de frequências para variáveis qualitativas

Método 1

https://pandas.pydata.org/pandas-docs/version/0.22/generated/pandas.Series.value_counts.html

In [10]:
```
frequencia = dados['Sexo'].value_counts()
percentual = dados['Sexo'].value_counts(normalize = True) * 100
dist_freq_qualitativas = pd.DataFrame({
    'Frequência': frequencia,
    'Porcentagem (%)': percentual
    })
dist_freq_qualitativas.rename(index = {0: 'Masculino', 1: 'Feminino'}, inplace = True)
dist_freq_qualitativas.rename_axis('Sexo', axis = 'columns', inplace = True)
dist_freq_qualitativas
```

Out[10]:

Sexo	Frequência	Porcentagem (%)
Masculino	53250	69.299844
Feminino	23590	30.700156

In [11]:
```
frequencia = dados['Cor'].value_counts()
percentual = dados['Cor'].value_counts(normalize = True) * 100
dist_freq_qualitativas = pd.DataFrame({
    'Frequência': frequencia,
    'Porcentagem (%)': percentual
    })
dist_freq_qualitativas.rename(index = {
    0: 'Indígena',
    2: 'Branca',
    4: 'Preta',
    6: 'Amarela',
    8: 'Parda',
    9: 'Sem declaração'
    }, inplace = True)
dist_freq_qualitativas.rename_axis('Cor', axis = 'columns', inplace = True)
dist_freq_qualitativas
```

Out[11]:

Cor	Frequência	Porcentagem (%)
Parda	35925	46.752993
Branca	31815	41.404217
Preta	8391	10.920094
Indígena	357	0.464602
Amarela	352	0.458095

In [12]:
```
frequencia = dados['UF'].value_counts()
percentual = dados['UF'].value_counts(normalize = True) * 100
dist_freq_qualitativas = pd.DataFrame({
    'Frequência': frequencia,
    'Porcentagem (%)': percentual
    })
dist_freq_qualitativas.rename(index = {
    11: 'Rondônia',
    12: 'Acre',
    13: 'Amazonas',
    14: 'Roraima',
    15: 'Pará',
    16: 'Amapá',
```

```
    17: 'Tocantins',
    21: 'Maranhão',
    22: 'Piauí',
    23: 'Ceará',
    24: 'Rio Grande do Norte',
    25: 'Paraíba',
    26: 'Pernambuco',
    27: 'Alagoas',
    28: 'Sergipe',
    29: 'Bahia',
    31: 'Minas Gerais',
    32: 'Espírito Santo',
    33: 'Rio de Janeiro',
    35: 'São Paulo',
    41: 'Paraná',
    42: 'Santa Catarina',
    43: 'Rio Grande do Sul',
    50: 'Mato Grosso do Sul',
    51: 'Mato Grosso',
    52: 'Goiás',
    53: 'Distrito Federal'
    }, inplace = True)
dist_freq_qualitativas.rename_axis('UF', axis = 'columns', inplace = True)
dist_freq_qualitativas
```

Out[12]:

UF	Frequência	Porcentagem (%)
São Paulo	8544	11.119209
Minas Gerais	7686	10.002603
Rio Grande do Sul	6322	8.227486
Bahia	5717	7.440135
Rio de Janeiro	5556	7.230609
Pará	4449	5.789953
Paraná	4356	5.668922
Pernambuco	3820	4.971369
Goiás	3478	4.526288
Ceará	3359	4.371421
Santa Catarina	2275	2.960698
Amazonas	2206	2.870901
Distrito Federal	2054	2.673087
Mato Grosso	1800	2.342530
Maranhão	1787	2.325612
Rondônia	1537	2.000260
Espírito Santo	1511	1.966424
Mato Grosso do Sul	1440	1.874024
Tocantins	1306	1.699636
Sergipe	1287	1.674909
Paraíba	1274	1.657991
Piauí	1211	1.576002
Rio Grande do Norte	973	1.266268
Acre	937	1.219417

UF	Frequência	Porcentagem (%)
Alagoas	903	1.175169
Roraima	540	0.702759
Amapá	512	0.666320

In [13]:

```
frequencia = dados['Anos de Estudo'].value_counts()
percentual = dados['Anos de Estudo'].value_counts(normalize = True) * 100
dist_freq_qualitativas = pd.DataFrame({
    'Frequência': frequencia,
    'Porcentagem (%)': percentual
    })
dist_freq_qualitativas.rename(index = {
    1:  'Sem instrução e menos de 1 ano',
    2:  '1 ano',
    3:  '2 anos',
    4:  '3 anos',
    5:  '4 anos',
    6:  '5 anos',
    7:  '6 anos',
    8:  '7 anos',
    9:  '8 anos',
    10: '9 anos',
    11: '10 anos',
    12: '11 anos',
    13: '12 anos',
    14: '13 anos',
    15: '14 anos',
    16: '15 anos ou mais',
    17: 'Não determinados'
    }, inplace = True)
dist_freq_qualitativas.rename_axis('Anos de Estudo', axis = 'columns', inplace = True)
dist_freq_qualitativas
```

Out[13]:

Anos de Estudo	Frequência	Porcentagem (%)
11 anos	20848	27.131702
15 anos ou mais	10795	14.048673
8 anos	7980	10.385216
4 anos	6729	8.757158
Sem instrução e menos de 1 ano	5849	7.611921
5 anos	4499	5.855023
3 anos	2891	3.762363
7 anos	2689	3.499479
6 anos	2445	3.181936
10 anos	2118	2.756377
2 anos	2101	2.734253
9 anos	1840	2.394586
12 anos	1836	2.389381
1 ano	1388	1.806351
14 anos	1388	1.806351
13 anos	1253	1.630661
Não determinados	191	0.248568

Método 2

https://pandas.pydata.org/pandas-docs/version/0.22/generated/pandas.crosstab.html

In [14]:
```
sexo = {
    0: 'Masculino',
    1: 'Feminino'
    }

cor = {
    0: 'Indígena',
    2: 'Branca',
    4: 'Preta',
    6: 'Amarela',
    8: 'Parda',
    9: 'Sem declaração'
    }
```

In [15]:
```
frequencia = pd.crosstab(dados.Sexo,
                         dados.Cor)
frequencia.rename(index = sexo, inplace = True)
frequencia.rename(columns = cor, inplace = True)
frequencia
```

Out[15]:

Cor	Indígena	Branca	Preta	Amarela	Parda
Sexo					
Masculino	256	22194	5502	235	25063
Feminino	101	9621	2889	117	10862

In [16]:
```
percentual = pd.crosstab(dados.Sexo,
                         dados.Cor,
                         normalize = True) * 100
percentual.rename(index = sexo, inplace = True)
percentual.rename(columns = cor, inplace = True)
percentual
```

Out[16]:

Cor	Indígena	Branca	Preta	Amarela	Parda
Sexo					
Masculino	0.333160	28.883394	7.160333	0.305830	32.617126
Feminino	0.131442	12.520822	3.759761	0.152264	14.135867

In [17]:
```
renda_media = pd.crosstab(dados.Sexo,
                         dados.Cor,
                         aggfunc = 'mean',
                         values = dados.Renda)
renda_media.rename(index = sexo, inplace = True)
renda_media.rename(columns = cor, inplace = True)
renda_media
```

Out[17]:

Cor	Indígena	Branca	Preta	Amarela	Parda
Sexo					
Masculino	1081.710938	2925.744435	1603.861687	4758.251064	1659.577425
Feminino	2464.386139	2109.866750	1134.596400	3027.341880	1176.758516

2.2 Distribuição de frequências para variáveis quantitativas (classes personalizadas)

Passo 1 - Especificar os limites de cada classe

Utilizar a seguinte classificação:

A ► Acima de 20 SM

B ► De 10 a 20 SM

C ► De 4 a 10 SM

D ► De 2 a 4 SM

E ► Até 2 SM

onde **SM** é o valor do salário mínimo na época. Em nosso caso **R$ 788,00** (2015):

A ► Acima de 15.760

B ► De 7.880 a 15.760

C ► De 3.152 a 7.880

D ► De 1.576 a 3.152

E ► Até 1.576

In [18]:
```
dados.Renda.min()
```

Out[18]:
```
0
```

In [19]:
```
dados.Renda.max()
```

Out[19]:
```
200000
```

In [20]:
```
sm = 788 # Salário mínimo no ano o qual os dados pertencem (2015)
classes = [0, (2 * sm), (4 * sm), (10 * sm), (20 * sm), 200000]
labels = ['E', 'D', 'C', 'B', 'A']
```

Passo 2 - Criar a tabela de frequências

https://pandas.pydata.org/pandas-docs/version/0.22/generated/pandas.cut.html

In [21]:
```
frequencia = pd.value_counts(pd.cut(x = dados.Renda,
                                    bins = classes,
                                    labels = labels,
                                    include_lowest = True)
                                    )
percentual = pd.value_counts(pd.cut(x = dados.Renda,
                                    bins = classes,
                                    labels = labels,
                                    include_lowest = True),
                                    normalize = True
                                    ) * 100
dist_freq_qualitativas = pd.DataFrame({
                                        'Frequência': frequencia,
```

```
                                                'Porcentagem (%)': percentual
                                                })
dist_freq_qualitativas.sort_index(ascending = False)
```

Out[21]:

	Frequência	Porcentagem (%)
A	608	0.791255
B	2178	2.834461
C	7599	9.889381
D	16700	21.733472
E	49755	64.751432

2.3 Distribuição de frequências para variáveis quantitativas (classes de amplitude fixa)

Importando bibliotecas

http://www.numpy.org/

In [22]:
```
import numpy as np
```

Passo 1 - Difinindo o número de classes

Regra de Sturges

$$k = 1 + {}^{103}\log_{10} n$$

In [23]:
```
n = dados.shape[0]
n
```

Out[23]: 76840

In [24]:
```
k = 1 + (10 / 3) * np.log10(n)
```

In [25]:
```
k
```

Out[25]: 17.285291187298853

In [26]:
```
k = int(k.round(0))
k
```

Out[26]: 17

Passo 2 - Criar a tabela de frequências

In [27]:
```
frequencia = pd.value_counts(
    pd.cut(
        x = dados.Renda,
        bins = k,
```

```
        include_lowest = True
    ),
    sort = False
)
percentual = pd.value_counts(
    pd.cut(
        x = dados.Renda,
        bins = k,
        include_lowest = True
    ),
    sort = False,
    normalize = True
) * 100
dist_freq_amplitude_fixa = pd.DataFrame({
                                        'Frequência': frequencia,
                                        'Porcentagem (%)': percentual
                                        })
dist_freq_amplitude_fixa.rename_axis('Faixas de Renda', axis = 'columns', inplace = True)
dist_freq_amplitude_fixa
```

Out[27]:

Faixas de Renda	Frequência	Porcentagem (%)
(-200.001, 11764.706]	75594	98.378449
(11764.706, 23529.412]	1022	1.330036
(23529.412, 35294.118]	169	0.219938
(35294.118, 47058.824]	19	0.024727
(47058.824, 58823.529]	16	0.020822
(58823.529, 70588.235]	5	0.006507
(70588.235, 82352.941]	4	0.005206
(82352.941, 94117.647]	1	0.001301
(94117.647, 105882.353]	6	0.007808
(105882.353, 117647.059]	0	0.000000
(117647.059, 129411.765]	1	0.001301
(129411.765, 141176.471]	0	0.000000
(141176.471, 152941.176]	0	0.000000
(152941.176, 164705.882]	0	0.000000
(164705.882, 176470.588]	0	0.000000
(176470.588, 188235.294]	0	0.000000
(188235.294, 200000.0]	3	0.003904

2.4 Histograma

O **HISTOGRAMA** é a representação gráfica de uma distribuição de frequências. É uma gráfico formado por um conjunto de retângulos colocados lado a lado, onde a área de cada retângulo é proporcional à frequência da classe que ele representa.

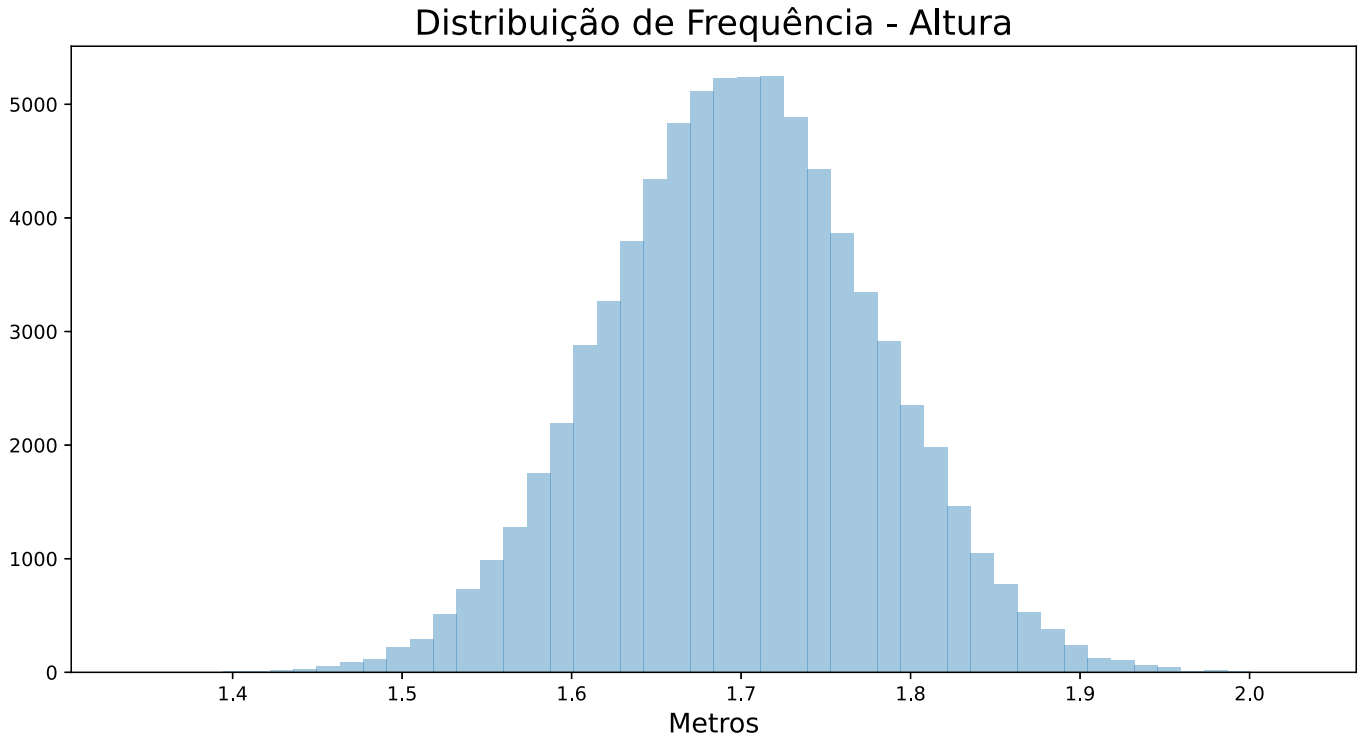
Importando a biblioteca

https://seaborn.pydata.org/


```
import seaborn as sns
```

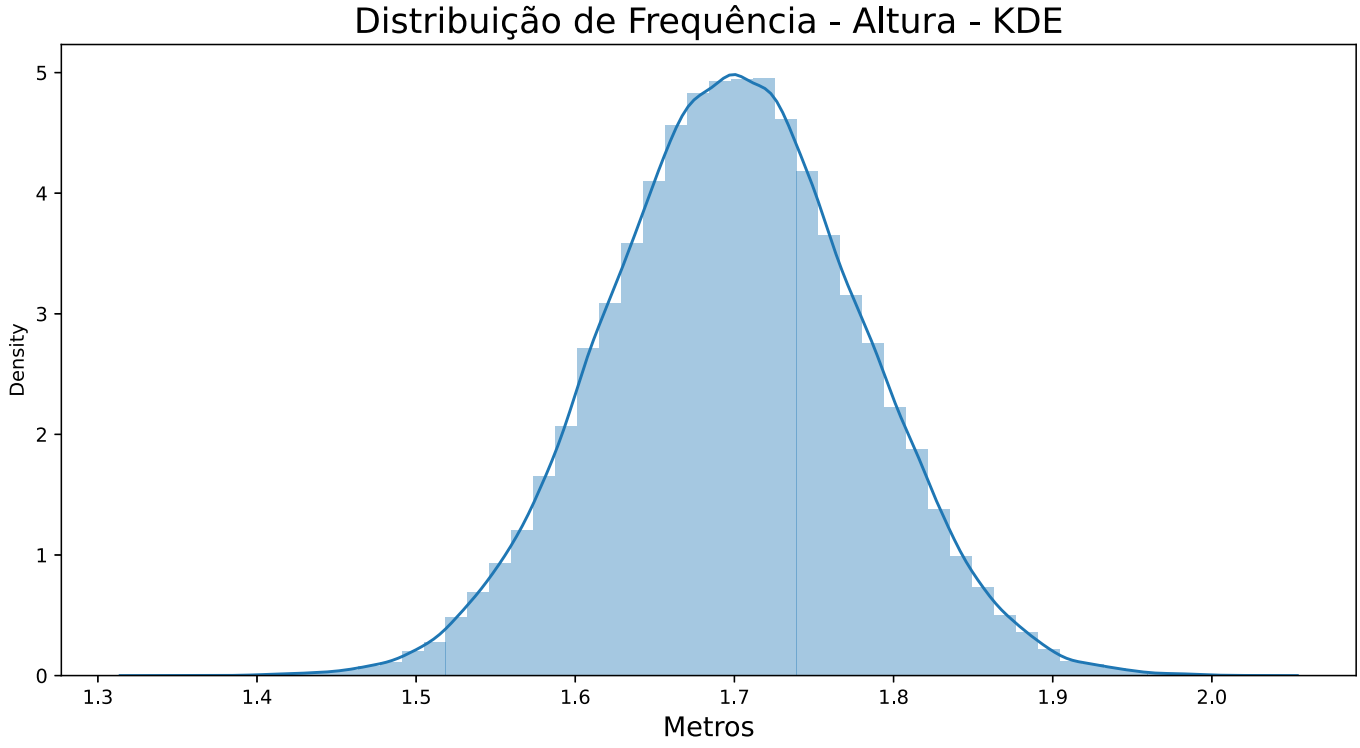
In [29]:
```
ax = sns.distplot(dados.Altura, kde = False)
ax.figure.set_size_inches(12, 6)
ax.set_title("Distribuição de Frequência - Altura", fontsize = 18)
ax.set_xlabel("Metros", fontsize = 14)
ax
```

Out[29]:
```
<AxesSubplot:title={'center':'Distribuição de Frequência - Altura'}, xlabel='Metros'>
```



In [30]:
```
ax = sns.distplot(dados.Altura)
ax.figure.set_size_inches(12, 6)
ax.set_title("Distribuição de Frequência - Altura - KDE", fontsize = 18)
ax.set_xlabel("Metros", fontsize = 14)
ax
```

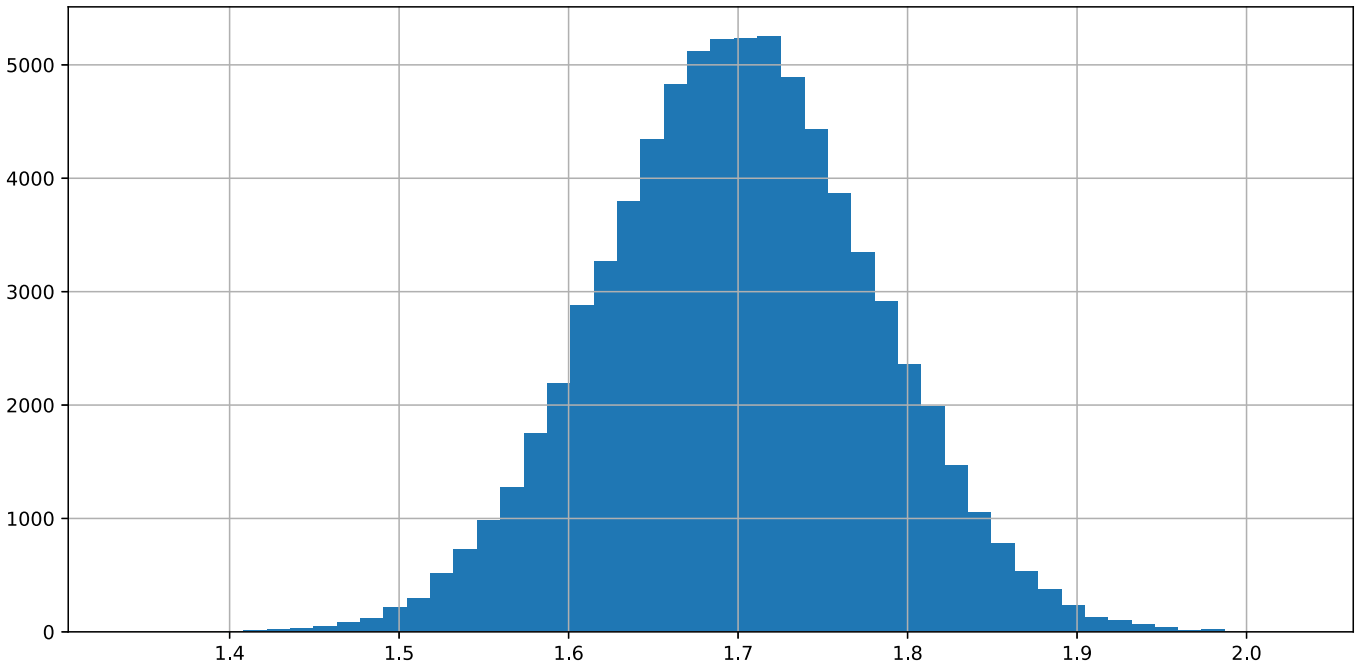
Out[30]:
```
<AxesSubplot:title={'center':'Distribuição de Frequência - Altura - KDE'}, xlabel='Metros', yla
bel='Density'>
```



Processing math: 100%

```
dados.Altura.hist(bins = 50, figsize = (12, 6))
```

Out[31]: <AxesSubplot:>



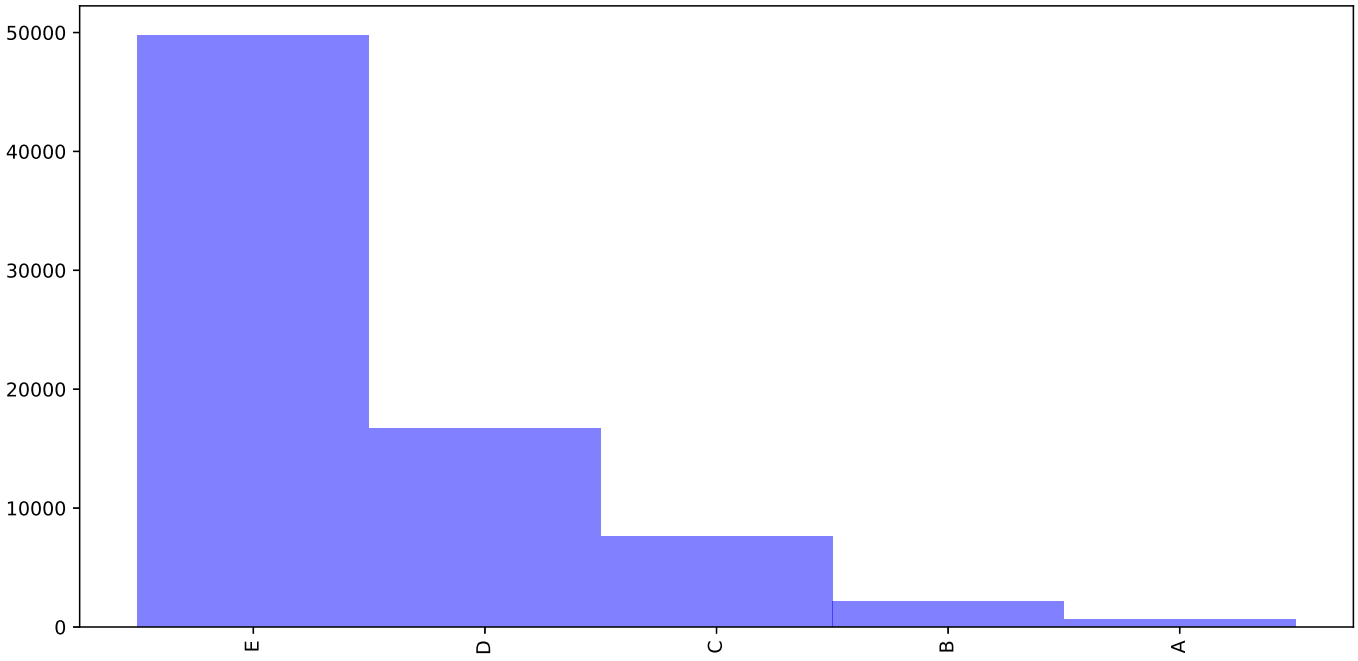
In [32]:
```
dist_freq_qualitativas
```

Out[32]:

	Frequência	Porcentagem (%)
E	49755	64.751432
D	16700	21.733472
C	7599	9.889381
B	2178	2.834461
A	608	0.791255

In [33]:
```
dist_freq_qualitativas['Frequência'].plot.bar(width = 1, color = 'blue', alpha = 0.5, figsize =
```

Out[33]: <AxesSubplot:>



Processing math: 100%

EDIDAS DE TENDÊNCIA CENTRAL

DataFrame de exemplo

In [34]:
```
df = pd.DataFrame(data = {'Fulano': [8, 10, 4, 8, 6, 10, 8],
                          'Beltrano': [10, 2, 0.5, 1, 3, 9.5, 10],
                          'Sicrano': [7.5, 8, 7, 8, 8, 8.5, 7]},
                  index = ['Matemática',
                           'Português',
                           'Inglês',
                           'Geografia',
                           'História',
                           'Física',
                           'Química'])
df.rename_axis('Matérias', axis = 'columns', inplace = True)
df
```

Out[34]:

Matérias	Fulano	Beltrano	Sicrano
Matemática	8	10.0	7.5
Português	10	2.0	8.0
Inglês	4	0.5	7.0
Geografia	8	1.0	8.0
História	6	3.0	8.0
Física	10	9.5	8.5
Química	8	10.0	7.0

3.1 Média aritmética

É representada por μ quando se refere à população e por $\bar{X}$ quando se refere à amostra

$$\mu = \frac{1}{n}\sum_{i=1}^{n} X_i$$

onde

n = número de observações (registros)

X_i = valor da i-ésima observação (registro)

In [35]:
```
(8 + 10 + 4 + 8 + 6 + 10 + 8) / 7
```

Out[35]: 7.714285714285714

In [36]:
```
df.Fulano.mean()
```

Out[36]: 7.714285714285714

In [37]:
```
dados.Renda.mean()
```

Out[37]: 2000.3831988547631

In [38]:
```
dados.groupby(['Sexo']).Renda.mean()
```

Out[38]:
```
Sexo
0    2192.441596
1    1566.847393
Name: Renda, dtype: float64
```

3.2 Mediana

Para obtermos a mediana de uma conjunto de dados devemos proceder da seguinte maneira:

1. Ordenar o conjunto de dados;
2. Identificar o número de observações (registros) do conjunto de dados (n);
3. Identicar o elemento mediano:

> Quando n for ímpar, a posição do elemento mediano será obtida da seguinte forma:

$$\text{Elemento}_{Md} = n + 12$$

> Quando n for par, a posição do elemento mediano será obtida da seguinte forma:

$$\text{Elemento}_{Md} = n2$$

1. Obter a mediana:

> Quando n for ímpar:

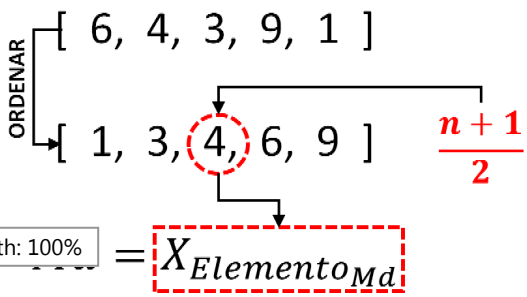
$$Md = X_{\text{Elemento}_{Md}}$$

> Quando n for par:

$$Md = X_{\text{Elemento}_{Md}} + X_{\text{Elemento}_{Md}+1}2$$

Exemplo 1 - n ímpar

n ímpar



In [39]:
```
notas_fulano = df.Fulano
notas_fulano
```

Out[39]:
```
Matemática     8
Português     10
Inglês         4
Geografia      8
História       6
Física        10
Química        8
Name: Fulano, dtype: int64
```

In [40]:
```
notas_fulano = notas_fulano.sort_values()
notas_fulano
```

Out[40]:
```
Inglês         4
História       6
Matemática     8
Geografia      8
Química        8
Português     10
Física        10
Name: Fulano, dtype: int64
```

In [41]:
```
notas_fulano = notas_fulano.reset_index()
notas_fulano
```

Out[41]:

	index	Fulano
0	Inglês	4
1	História	6
2	Matemática	8
3	Geografia	8
4	Química	8
5	Português	10
6	Física	10

In [42]:
```
n = notas_fulano.shape[0]
n
```

Out[42]:
```
7
```

In [43]:
```
elemento_md = int((n + 1) / 2)
elemento_md
```

Out[43]:
```
4
```

In [44]:
```
notas_fulano.loc[elemento_md - 1]
```

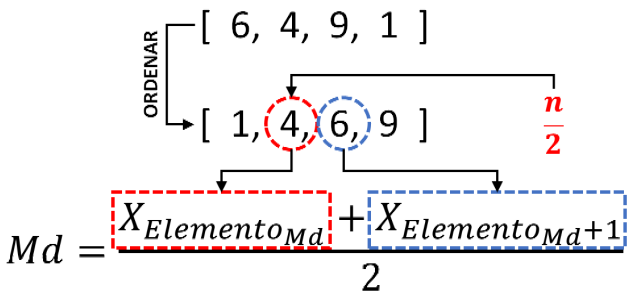
Out[44]:
```
index     Geografia
Fulano            8
Name: 3, dtype: object
```

In [45]:
```
notas_fulano.median()
```


Out[45]:
```
8.0
dtype: float64
```

Exemplo 2 - n par

n par



In [46]:
```
notas_beltrano = df.Beltrano.sample(6, random_state = 101)
notas_beltrano
```

Out[46]:
```
Matemática    10.0
Inglês         0.5
Física         9.5
História       3.0
Química       10.0
Português      2.0
Name: Beltrano, dtype: float64
```

In [47]:
```
notas_beltrano = notas_beltrano.sort_values()
notas_beltrano
```

Out[47]:
```
Inglês         0.5
Português      2.0
História       3.0
Física         9.5
Matemática    10.0
Química       10.0
Name: Beltrano, dtype: float64
```

In [48]:
```
notas_beltrano = notas_beltrano.reset_index()
notas_beltrano
```

Out[48]:

	index	Beltrano
0	Inglês	0.5
1	Português	2.0
2	História	3.0
3	Física	9.5
4	Matemática	10.0
5	Química	10.0

In [49]:
```
n = notas_fulano.shape[0]
n
```

Out[49]: 7

In [50]:
```
elemento_md = int(n / 2)
elemento_md
```

Out[50]: 3

In [51]:
```
(notas_beltrano.loc[elemento_md - 1].Beltrano + notas_beltrano.loc[elemento_md].Beltrano) / 2
```

Out[51]:
```
6.25
```

In [52]:
```
notas_beltrano.median()
```

Out[52]:
```
Beltrano    6.25
dtype: float64
```

Obtendo a mediana em nosso dataset

In [53]:
```
dados.Renda.median()
```

Out[53]:
```
1200.0
```

In [54]:
```
dados.Renda.quantile()
```

Out[54]:
```
1200.0
```

3.3 Moda

Pode-se definir a moda como sendo o valor mais frequente de um conjunto de dados. A moda é bastante utilizada para dados qualitativos.

In [55]:
```
df
```

Out[55]:

Matérias	Fulano	Beltrano	Sicrano
Matemática	8	10.0	7.5
Português	10	2.0	8.0
Inglês	4	0.5	7.0
Geografia	8	1.0	8.0
História	6	3.0	8.0
Física	10	9.5	8.5
Química	8	10.0	7.0

In [56]:
```
df.mode()
```

Out[56]:

Matérias	Fulano	Beltrano	Sicrano
0	8	10.0	8.0

In [57]:
```
exemplo = pd.Series([1, 2, 2, 3, 4, 4, 5, 6, 6])
exemplo
```

Out[57]:
```
0    1
1    2
2    2
3    3
4    4
5    4
```

```
7    6
8    6
dtype: int64
```

In [58]:
```
exemplo.mode()
```

Out[58]:
```
0    2
1    4
2    6
dtype: int64
```

Obtendo a moda em nosso dataset

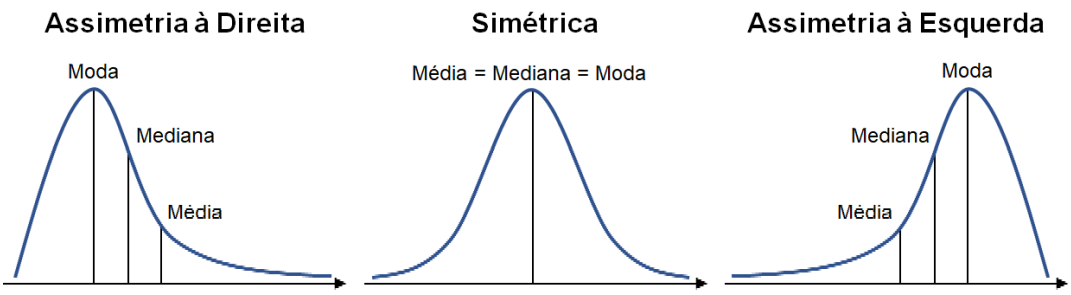
In [59]:
```
dados.Renda.mode()
```

Out[59]:
```
0    788
dtype: int64
```

In [60]:
```
dados.Altura.mode()
```

Out[60]:
```
0    1.568128
1    1.671225
2    1.681659
3    1.692977
4    1.708163
5    1.708370
6    1.753842
7    1.779073
8    1.796462
dtype: float64
```

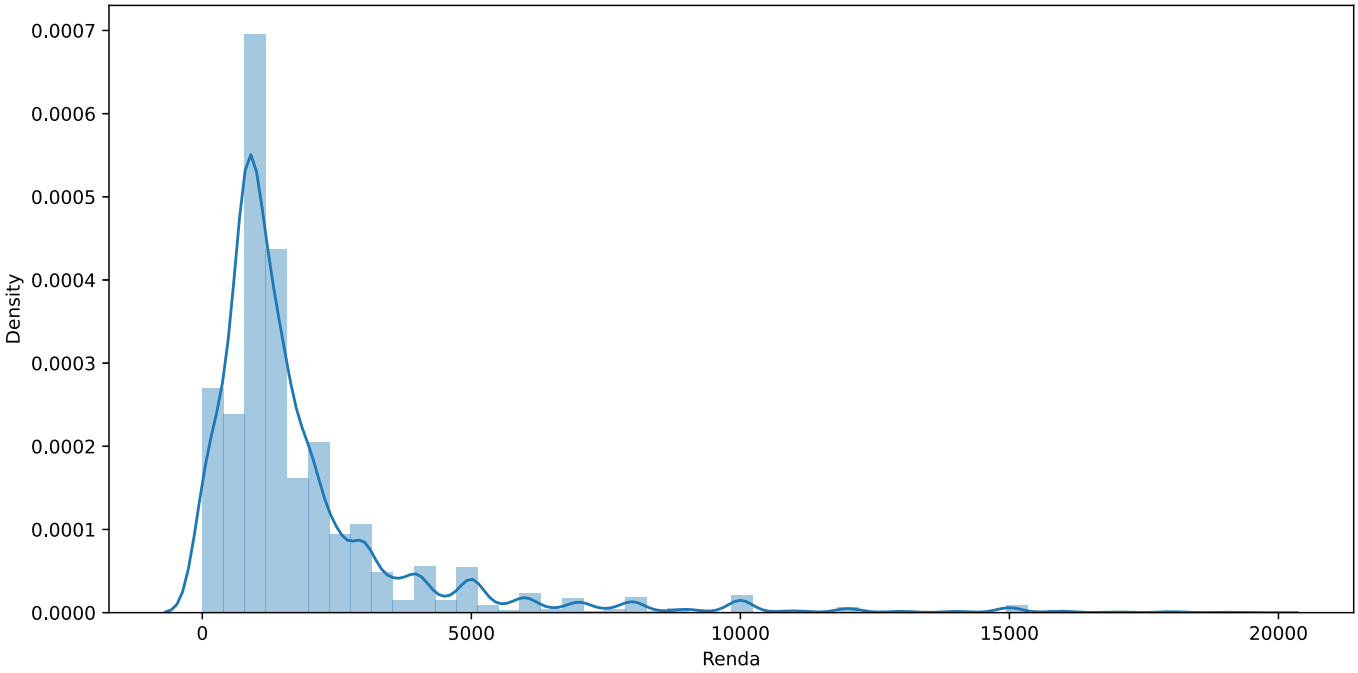
3.4 Relação entre média, mediana e moda



Avaliando a variável RENDA

In [61]:
```
ax = sns.distplot(dados.query('Renda < 20000').Renda)
ax.figure.set_size_inches(12, 6)
ax
```

Out[61]:
```
<AxesSubplot:xlabel='Renda', ylabel='Density'>
```



In [62]:
```
moda = int(dados.Renda.mode([0])[0])
moda
```

Out[62]:
```
788
```

In [63]:
```
mediana = dados.Renda.median()
mediana
```

Out[63]:
```
1200.0
```

In [64]:
```
media = dados.Renda.mean()
media
```

Out[64]:
```
2000.3831988547631
```

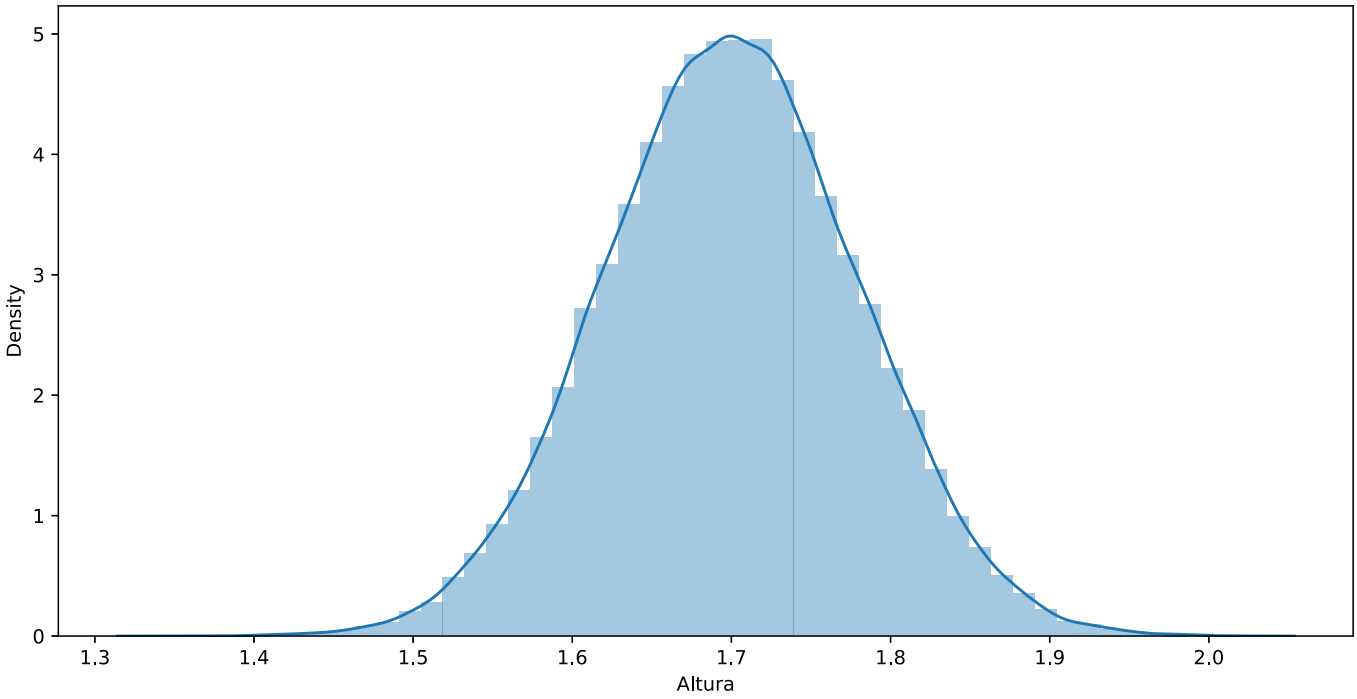
In [65]:
```
moda < mediana < media
```

Out[65]:
```
True
```

Avaliando a variável ALTURA

In [66]:
```
ax = sns.distplot(dados.Altura)
ax.figure.set_size_inches(12, 6)
ax
```

Out[66]:
```
<AxesSubplot:xlabel='Altura', ylabel='Density'>
```



In [67]:
```
moda = dados.Altura.mode()
moda
```

Out[67]:
```
0    1.568128
1    1.671225
2    1.681659
3    1.692977
4    1.708163
5    1.708370
6    1.753842
7    1.779073
8    1.796462
dtype: float64
```

In [68]:
```
mediana = dados.Altura.median()
mediana
```

Out[68]:
```
1.6993247325
```

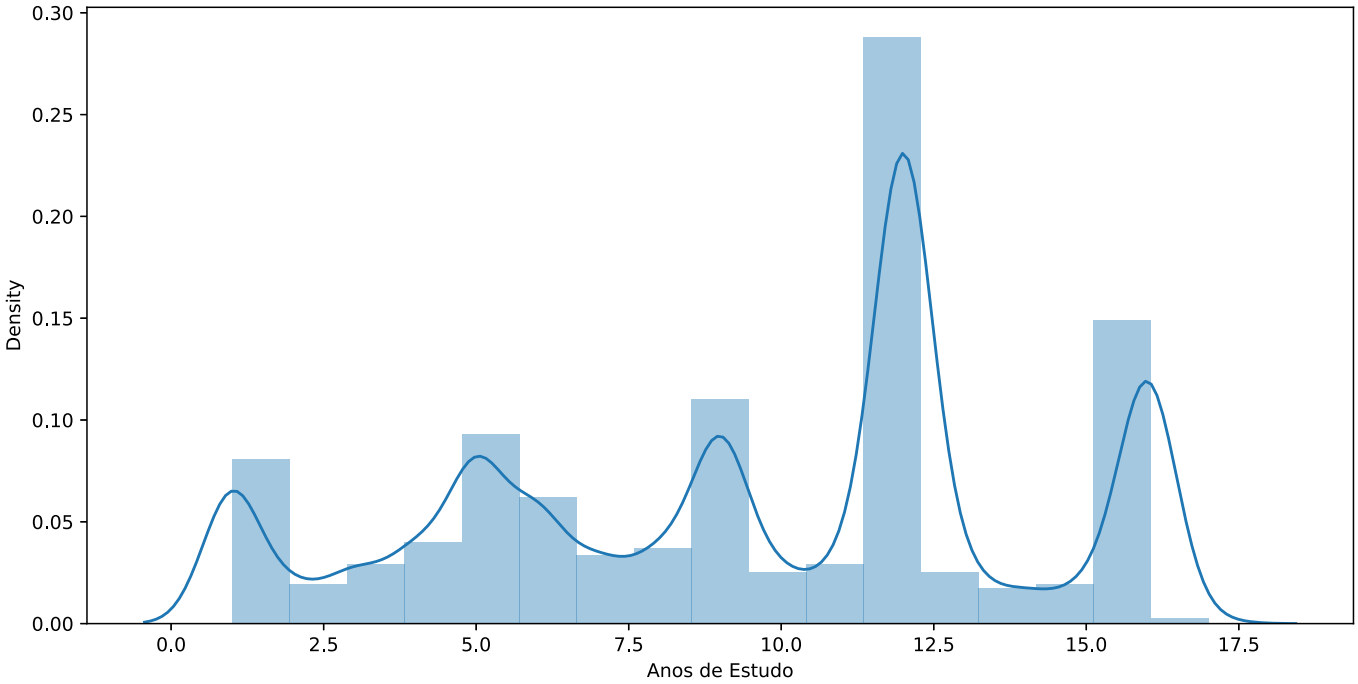
In [69]:
```
media = dados.Altura.mean()
media
```

Out[69]:
```
1.6995124540575741
```

Avaliando a variável ANOS DE ESTUDO

In [70]:
```
ax = sns.distplot(dados['Anos de Estudo'], bins = 17)
ax.figure.set_size_inches(12, 6)
ax
```

Out[70]:
```
<AxesSubplot:xlabel='Anos de Estudo', ylabel='Density'>
```



In [71]:
```
moda = dados['Anos de Estudo'].mode()[0]
moda
```

Out[71]: 12

In [72]:
```
mediana = dados['Anos de Estudo'].median()
mediana
```

Out[72]: 11.0

In [73]:
```
media = dados['Anos de Estudo'].mean()
media
```

Out[73]: 9.469664237376367

In [74]:
```
moda > mediana > media
```

Out[74]: True

4 MEDIDAS SEPARATRIZES

4.1 Quartis, decis e percentis

Há uma série de medidas de posição semelhantes na sua concepção à mediana, embora não sejam medidas de tendência central. Como se sabe, a mediana divide a distribuição em duas partes iguais quanto ao número de elementos de cada parte. Já os quartis permitem dividir a distribuição em quatro partes iguais quanto ao número de elementos de cada uma; os decis em dez partes e os centis em cem partes iguais.

In [75]:
```
dados.Renda.quantile([0.25, 0.5, 0.75])
```

Out[75]:
```
0.25     788.0
1200.0
```

```
0.75    2000.0
Name: Renda, dtype: float64
```

In [76]:
```
dados.Renda.quantile([i / 10 for i in range(1, 10)])
```

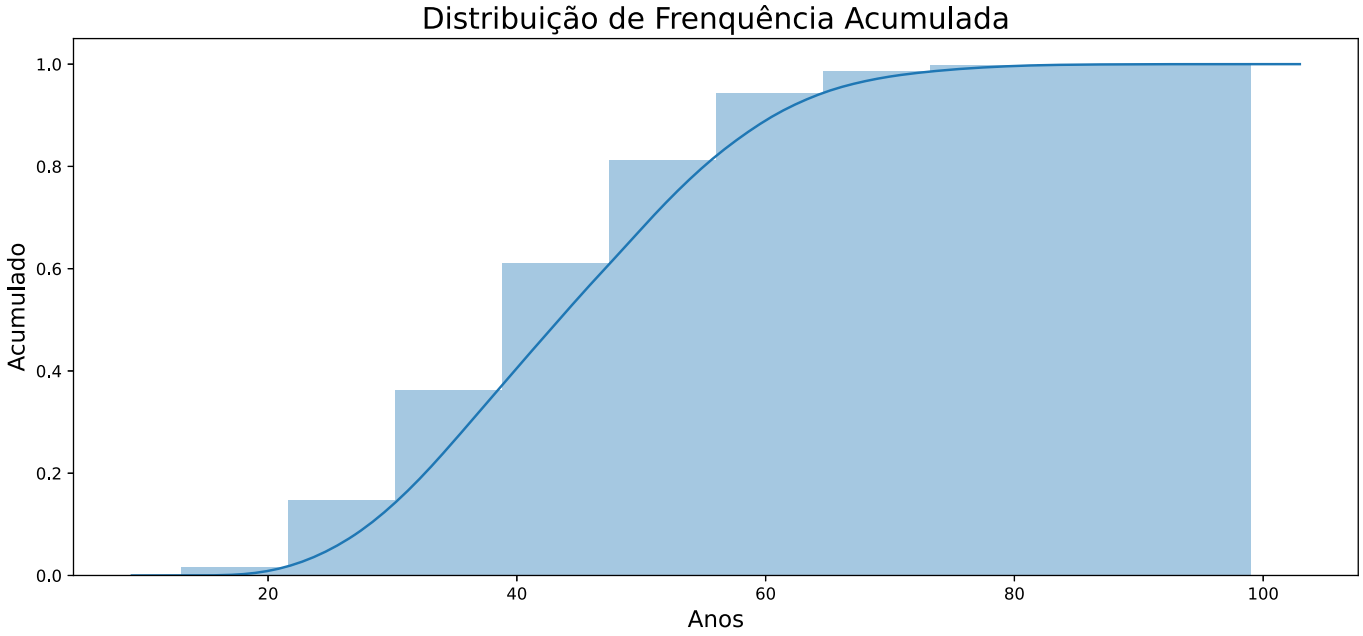
Out[76]:
```
0.1     350.0
0.2     788.0
0.3     800.0
0.4    1000.0
0.5    1200.0
0.6    1500.0
0.7    1900.0
0.8    2500.0
0.9    4000.0
Name: Renda, dtype: float64
```

In [77]:
```
dados.Renda.quantile([i / 100 for i in range(1, 100)])
```

Out[77]:
```
0.01        0.0
0.02        0.0
0.03        0.0
0.04       50.0
0.05      100.0
          ...
0.95     6000.0
0.96     7000.0
0.97     8000.0
0.98    10000.0
0.99    15000.0
Name: Renda, Length: 99, dtype: float64
```

In [78]:
```
ax = sns.distplot(dados.Idade,
                  hist_kws = {'cumulative': True},
                  kde_kws = {'cumulative': True},
                  bins = 10)
ax.figure.set_size_inches(14, 6)
ax.set_title("Distribuição de Frenquência Acumulada", fontsize = 18)
ax.set_ylabel("Acumulado", fontsize = 14)
ax.set_xlabel("Anos", fontsize = 14)
ax
```

Out[78]:
```
<AxesSubplot:title={'center':'Distribuição de Frenquência Acumulada'}, xlabel='Anos', ylabel='A
cumulado'>
```



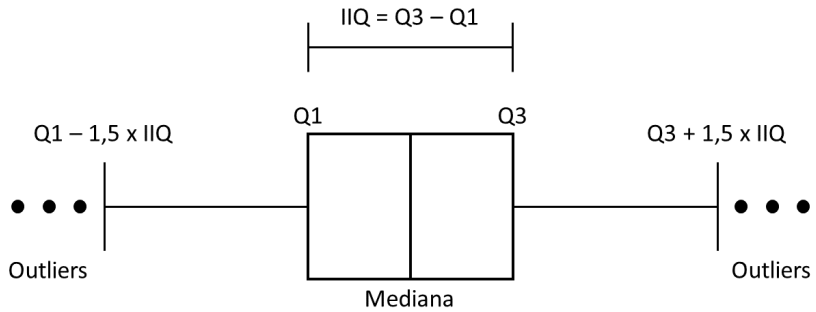
In [79]:
```
dados.Idade.quantile([i / 10 for i in range(1, 10)])
```


Out[79]:
```
0.1    28.0
```

```
0.2    33.0
0.3    36.0
0.4    40.0
0.5    43.0
0.6    47.0
0.7    51.0
0.8    55.0
0.9    61.0
Name: Idade, dtype: float64
```

4.2 Box-plot

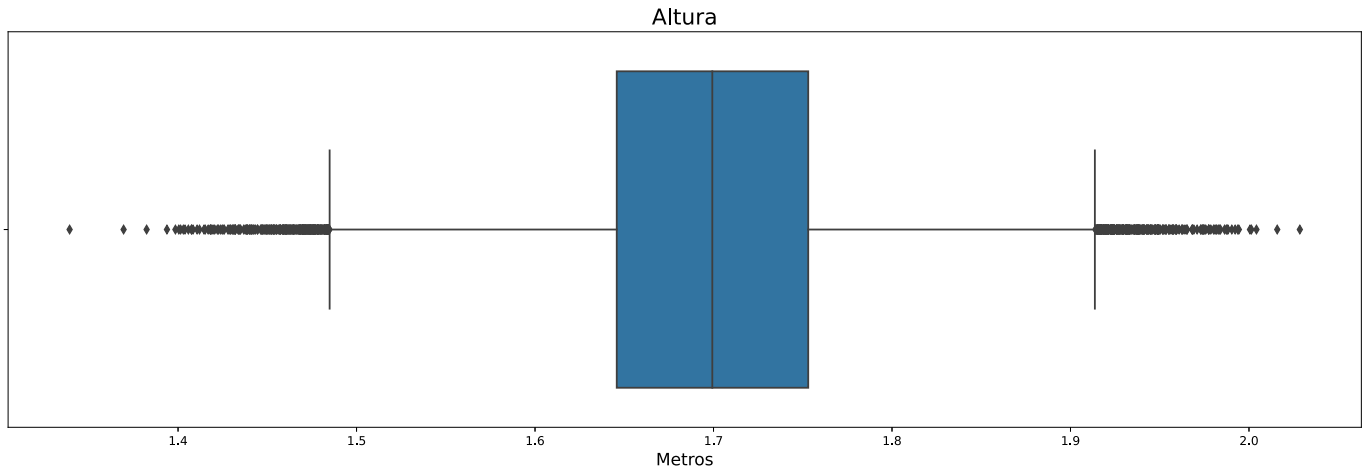
O box plot dá uma idéia da posição, dispersão, assimetria, caudas e dados discrepantes (outliers). A posição central é dada pela mediana e a dispersão por IIQ. As posições relativas de Q1, Mediana e Q3 dão uma noção da simetria da distribuição. Os comprimentos das cauda são dados pelas linhas que vão do retângulo aos valores remotos e pelos valores atípicos.



Box-plot

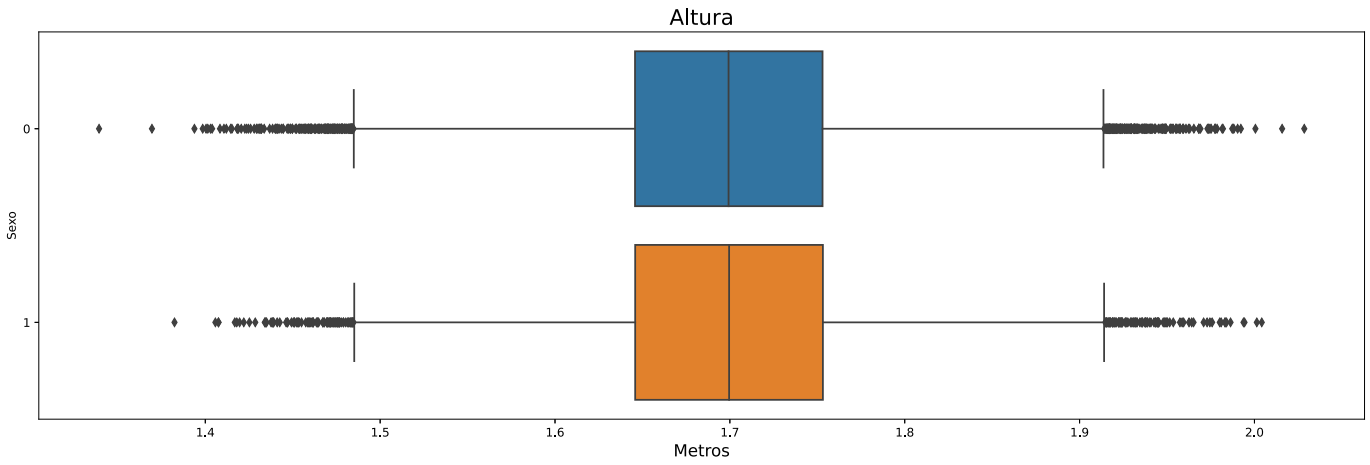
In [80]:
```
ax = sns.boxplot(x = 'Altura', data = dados, orient = 'h')
ax.figure.set_size_inches(20, 6)
ax.set_title("Altura", fontsize = 18)
ax.set_xlabel("Metros", fontsize = 14)
ax
```

Out[80]:
```
<AxesSubplot:title={'center':'Altura'}, xlabel='Metros'>
```



In [81]:
```
ax = sns.boxplot(x = 'Altura', y = 'Sexo', data = dados, orient = 'h')
ax.figure.set_size_inches(20, 6)
ax.set_title("Altura", fontsize = 18)
ax.set_xlabel("Metros", fontsize = 14)
ax
```

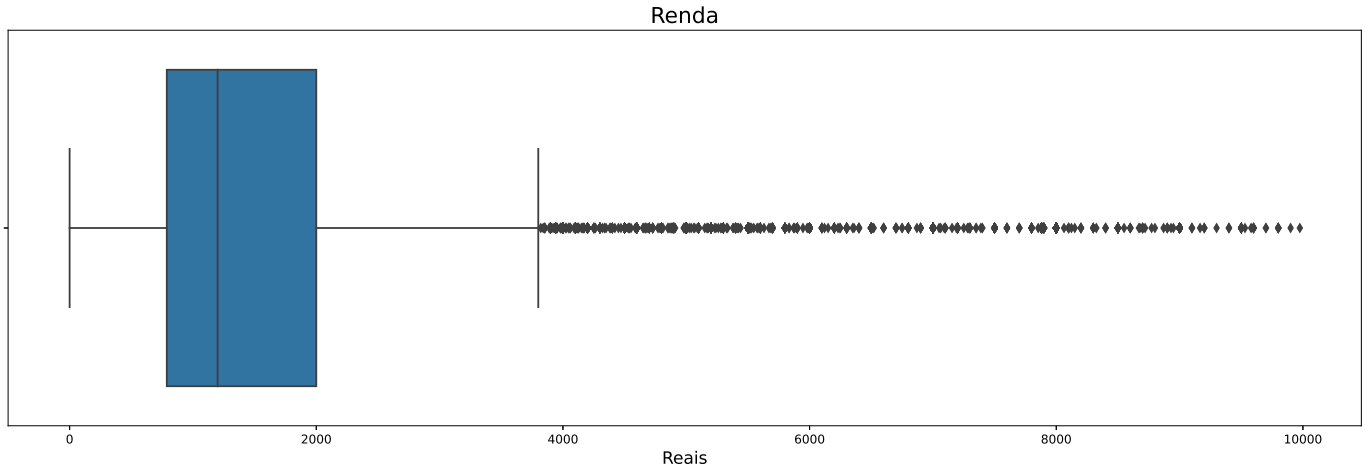

```
ubplot:title={'center':'Altura'}, xlabel='Metros', ylabel='Sexo'>
```



In [82]:

```
ax = sns.boxplot(x = 'Renda', data = dados.query('Renda < 10000'), orient = 'h')
ax.figure.set_size_inches(20, 6)
ax.set_title("Renda", fontsize = 18)
ax.set_xlabel("Reais", fontsize = 14)
ax
```

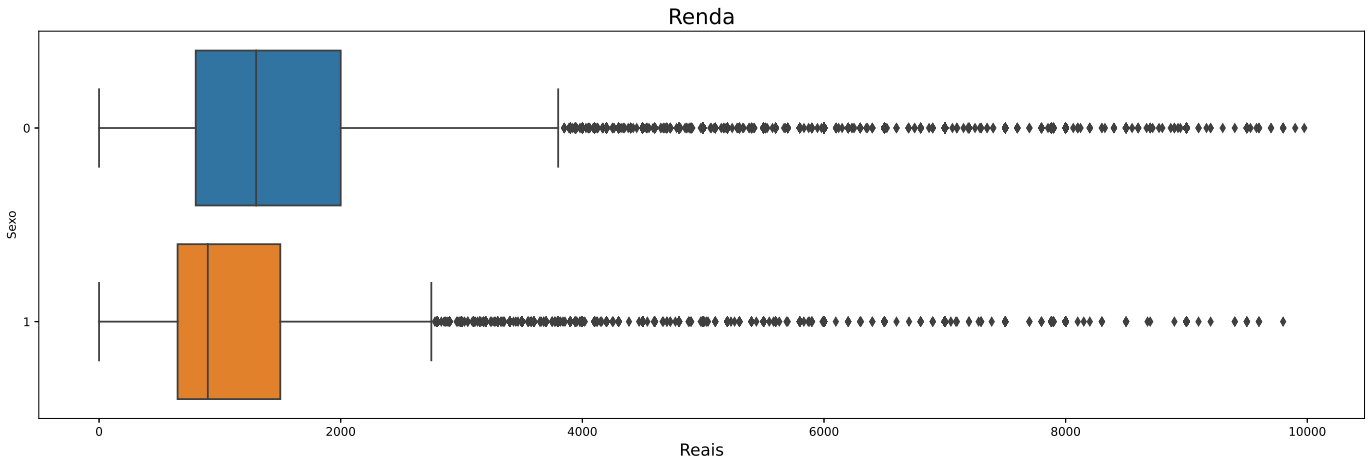
Out[82]: <AxesSubplot:title={'center':'Renda'}, xlabel='Reais'>



In [83]:

```
ax = sns.boxplot(x = 'Renda', y = 'Sexo', data = dados.query('Renda < 10000'), orient = 'h')
ax.figure.set_size_inches(20, 6)
ax.set_title("Renda", fontsize = 18)
ax.set_xlabel("Reais", fontsize = 14)
ax
```

Out[83]: <AxesSubplot:title={'center':'Renda'}, xlabel='Reais', ylabel='Sexo'>

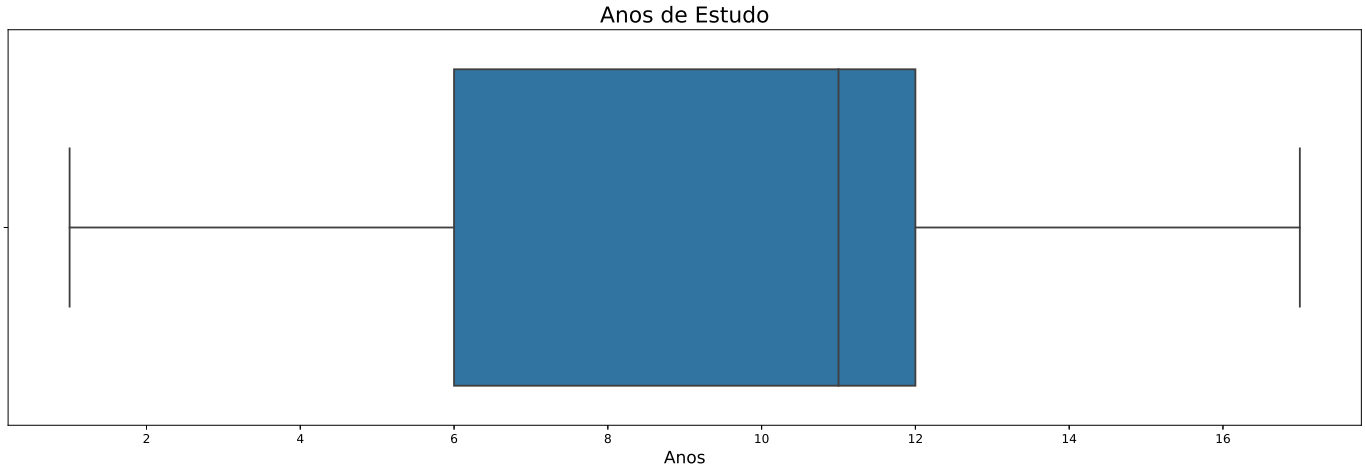


In [84]:

```
ax = sns.boxplot(x = 'Anos de Estudo', data = dados, orient = 'h')
ax.figure.set_size_inches(20, 6)
ax._title("Anos de Estudo", fontsize = 18)
```

```
ax.set_xlabel("Anos", fontsize = 14)
ax
```

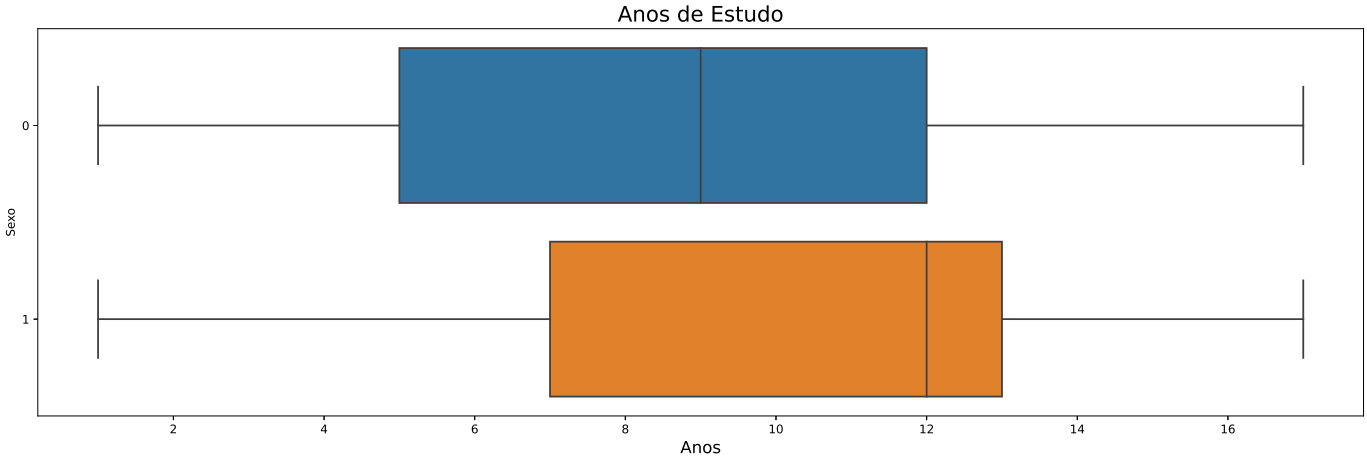
Out[84]: <AxesSubplot:title={'center':'Anos de Estudo'}, xlabel='Anos'>

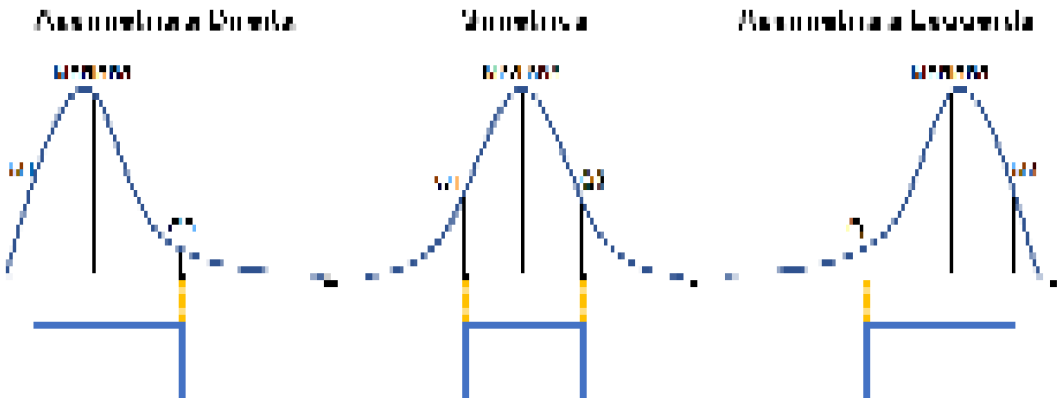


In [85]:

```
ax = sns.boxplot(x = 'Anos de Estudo', y = 'Sexo', data = dados, orient = 'h')
ax.figure.set_size_inches(20, 6)
ax.set_title("Anos de Estudo", fontsize = 18)
ax.set_xlabel("Anos", fontsize = 14)
ax
```

Out[85]: <AxesSubplot:title={'center':'Anos de Estudo'}, xlabel='Anos', ylabel='Sexo'>





5 MEDIDAS DE DISPERSÃO

Embora as medidas de posição forneçam uma sumarização bastante importante dos dados, elas podem não ser suficientes para caracterizar conjuntos distintos, especialmente quando as observações de determinada distribuição apresentarem dados muito dispersos.

5.1 Desvio médio absoluto

$$DM = {}^{1n}n\sum i=1|X_i - {}^{-}X|$$

In [86]:
```
notas_fulano = df['Fulano']
notas_fulano
```

Out[86]:
```
Matemática     8
Português     10
Inglês         4
Geografia      8
História       6
Física        10
Química        8
Name: Fulano, dtype: int64
```

In [87]:
```
notas_fulano = df[['Fulano']]
notas_fulano
```

Out[87]:

Matérias	Fulano
Matemática	8
Português	10
Inglês	4
Geografia	8
História	6
Física	10
Química	8

In [88]:
```
nota_media_fulano = notas_fulano.mean()[0]
nota_media_fulano
```

Out[88]:
```
7.714285714285714
```

In [89]:
```
notas_fulano['Desvio'] = notas_fulano['Fulano'] - nota_media_fulano
notas_fulano
```

Out[89]:

Matérias	Fulano	Desvio
Matemática	8	0.285714
Português	10	2.285714
Inglês	4	-3.714286
Geografia	8	0.285714
História	6	-1.714286
Física	10	2.285714
Química	8	0.285714

```
.fulano['Desvio'].sum()
```

Out[90]: -8.881784197001252e-16

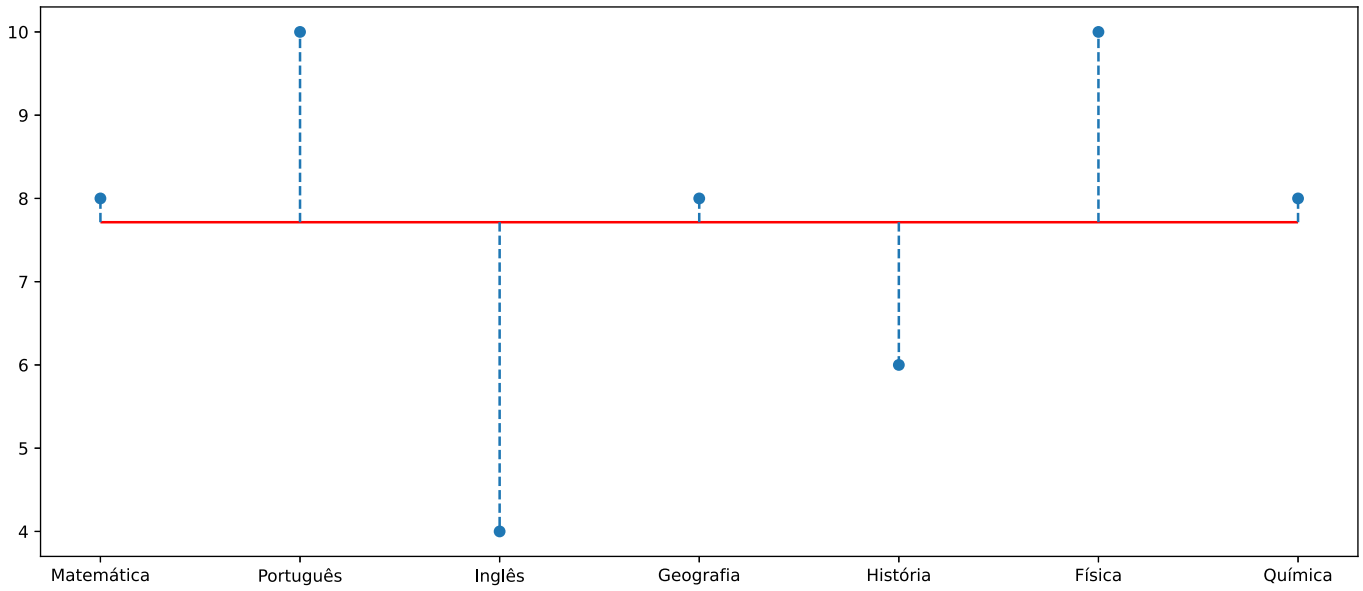
In [91]:
```
notas_fulano['|Desvio|'] = notas_fulano['Desvio'].abs()
notas_fulano
```

Out[91]:

Matérias	Fulano	Desvio	|Desvio|
Matemática	8	0.285714	0.285714
Português	10	2.285714	2.285714
Inglês	4	-3.714286	3.714286
Geografia	8	0.285714	0.285714
História	6	-1.714286	1.714286
Física	10	2.285714	2.285714
Química	8	0.285714	0.285714

In [92]:
```
ax = notas_fulano['Fulano'].plot(style = 'o')
ax.figure.set_size_inches(14, 6)
ax.hlines(y = nota_media_fulano, xmin = 0, xmax = notas_fulano.shape[0] - 1, colors = 'red')
for i in range(notas_fulano.shape[0]):
    ax.vlines(x = i, ymin = nota_media_fulano, ymax = notas_fulano['Fulano'][i], linestyle='das
ax
```

Out[92]: <AxesSubplot:>



In [93]:
```
notas_fulano['|Desvio|'].mean()
```

Out[93]: 1.5510204081632648

In [94]:
```
desvio_medio_absoluto = notas_fulano['Fulano'].mad()
desvio_medio_absoluto
```

Out[94]: 1.5510204081632648

5.2 Variância

---

ncia

A variância é construída a partir das diferenças entre cada observação e a média dos dados, ou seja, o desvio em torno da média. No cálculo da variância, os desvios em torno da média são elevados ao quadrado.

Variância populacional

$$\sigma^2 = 1n n\sum i=1(X_i - \mu)^2$$

Variância amostral

$$s^2 = 1n - 1 n\sum i=1(X_i - \bar{X})^2$$

In [95]:
```
notas_fulano['(Desvio)^2'] = notas_fulano['Desvio'].pow(2)
notas_fulano
```

Out[95]:

Matérias	Fulano	Desvio	|Desvio|	(Desvio)^2
Matemática	8	0.285714	0.285714	0.081633
Português	10	2.285714	2.285714	5.224490
Inglês	4	-3.714286	3.714286	13.795918
Geografia	8	0.285714	0.285714	0.081633
História	6	-1.714286	1.714286	2.938776
Física	10	2.285714	2.285714	5.224490
Química	8	0.285714	0.285714	0.081633

In [96]:
```
notas_fulano['(Desvio)^2'].sum() / (len(notas_fulano) - 1)
```

Out[96]:
```
4.57142857142857
```

In [97]:
```
variancia = notas_fulano['Fulano'].var()
variancia
```

Out[97]:
```
4.57142857142857
```

5.3 Desvio padrão

Uma das restrições da variância é o fato de fornecer medidas em quadrados das unidades originais - a variância de medidas de comprimento, por exemplo, é em unidades de área. Logo, o fato de as unidades serem diferentes dificulta a comparação da dispersão com as variáveis que a definem. Um modo de eliminar essa dificuldade é considerar sua raiz quadrada.

Desvio padrão populacional

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \implies \sigma = \sqrt{\sigma^2}$$

Desvio padrão amostral

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \implies s = \sqrt{s^2}$$

In [98]:
```
np.sqrt(variancia)
```

Out[98]:
```
2.1380899352993947
```

In [99]:
```
desvio_padrao = notas_fulano['Fulano'].std()
desvio_padrao
```

Out[99]:
```
2.1380899352993947
```

In [100…
```
df.std()
```

Out[100…
```
Matérias
Fulano      2.138090
Beltrano    4.460141
Sicrano     0.566947
dtype: float64
```