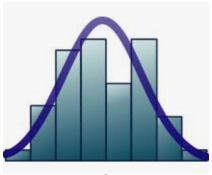
# Análise Descritiva



```
from google.colab import drive
drive.mount('/content/drive')
```

# Pandas Dataframe

- Mas é muito comum ler dados externos (como .csv) e gerar um **Dataframe** (tabela de dados)

```
import pandas as pd
auto = pd.read_csv("autoinsurance.csv")
```

| us_state | state | capital | pricelevel | Y2021 | Y2022 | Y2023 | pop | lat | lon | Region | Division |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MI | Michigan | Lansing | E | 5740 | 4386 | 2352 | 10,077,331 | 42.733635 | -84.555328 | Midwest | East North Central |
| RI | Rhode Island | Providence | E | 1375 | 1197 | 1200 | 1,097,379 | 41.830914 | -71.414963 | Northeast | New England |
| NV | Nevada | Carson City | E | 1033 | 1138 | 1164 | 3,104,614 | 39.163914 | -119.766120 | West | Mountain |

  - Cada coluna pode ser interpretada com um dicionário

```
auto['Y2023'] ou auto[['Y2023','Y2022','Y2021']]
```

  - Operações podem ser realizadas em vetores inteiros

```
(auto['Y2023'] + auto['Y2022'] + auto['Y2021'])/3
```

# Pandas Dataframe

- Mostrar as primeiras e últimas linhas

```
auto.head(3) e auto.tail(2)
```

  - Filtrar linhas

```
auto[auto['Region']=='Midwest']
```

```
auto[auto['Y2023']>999]
```

| us_state | state | capital | pricelevel | Y2021 | Y2022 | Y2023 | pop | lat | lon | Region | Division |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MI | Michigan | Lansing | E | 5740 | 4386 | 2352 | 10,077,331 | 42.733635 | -84.555328 | Midwest | East North Central |
| RI | Rhode Island | Providence | E | 1375 | 1197 | 1200 | 1,097,379 | 41.830914 | -71.414963 | Northeast | New England |
| NV | Nevada | Carson City | E | 1033 | 1138 | 1164 | 3,104,614 | 39.163914 | -119.766120 | West | Mountain |
| FL | Florida | Tallahassee | E | 2361 | 2072 | 1092 | 21,538,187 | 30.438118 | -84.281296 | South | South Atlantic |
| NJ | New Jersey | Trenton | E | 812 | 979 | 1032 | 9,288,994 | 40.220596 | -74.769913 | Northeast | Middle Atlantic |
| DE | Delaware | Dover | E | 1200 | 1183 | 1008 | 989,948 | 39.157307 | -75.519722 | South | South Atlantic |

# Tipos de Variáveis

**Qualitativas**

***Nominais***

-> notas_turma['Curso']

***Ordinais***

-> notas_turma['Conceito']

**Quantitativas**

***Contínuas***

-> notas_turma['Nota']

***Discretas***

-> números inteiros

| **Qualitativas** | |
|---|---|
| ***Nominais*** | categóricas |
| ***Ordinais*** | categóricas ordenadas |
| **Quantitativas** | |
| ***Discretas*** | números inteiros |
| ***Contínuas*** | números decimais |

# Consolidando Dados

- Usamos a função ***groupby()*** para consolidar os dados, passando como parâmetro o critério de agrupamento, no exemplo, **'Region'**. Na sequência aplicar a função desejada, no exemplo, **mean**(). Por fim, indicar qual coluna interessa ser contada/apresentada, no exemplo, **'Y2023'**.

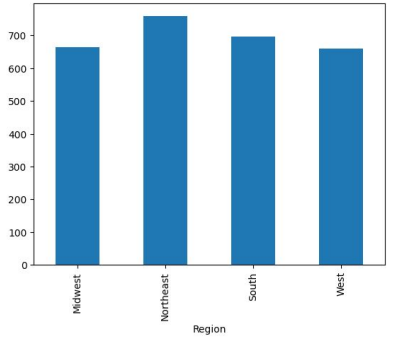
```
auto.groupby('Region')['Y2023'].mean()
```

| Region | Y2023 |
|---|---|
| Midwest | 663.000000 |
| Northeast | 758.666667 |
| South | 696.750000 |
| West | 659.076923 |

# Gráficos: Barras (Comparação)

- O Data frame já dispõe de uma função **plot**() passando como parâmetro o tipo (**kind='bar'**)

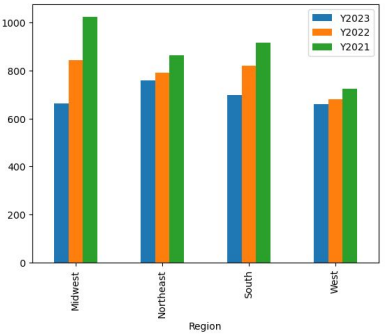
```
avg_premium_region.plot(kind='bar')
```




- Idem para múltiplas colunas

```
avg_premium_region = auto.groupby('Region')[['Y2023','Y2022','Y2021']].mean()
avg_premium_region.plot(kind='bar')
```

| Region | Y2023 | Y2022 | Y2021 |
|---|---|---|---|
| Midwest | 663.000000 | 844.250000 | 1022.666667 |
| Northeast | 758.666667 | 789.666667 | 862.000000 |
| South | 696.750000 | 820.125000 | 914.625000 |
| West | 659.076923 | 681.307692 | 724.153846 |

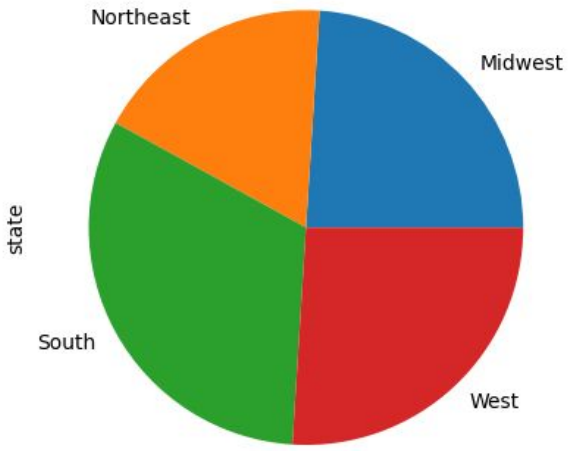


# Gráficos: Pizza (Proporção)

- Um gráfico de pizza é usado para visualizar a proporção de categorias dentro de um conjunto de dados. Quando você executa count_state_region.**plot(kind='pie')**, ele cria um gráfico de pizza com base nas contagens de estados em cada região.

```
count_state_region = auto.groupby('Region')['state'].count()
count_state_region.plot(kind='pie')
```

| Region | state |
|---|---|
| Midwest | 12 |
| Northeast | 9 |
| South | 16 |
| West | 13 |

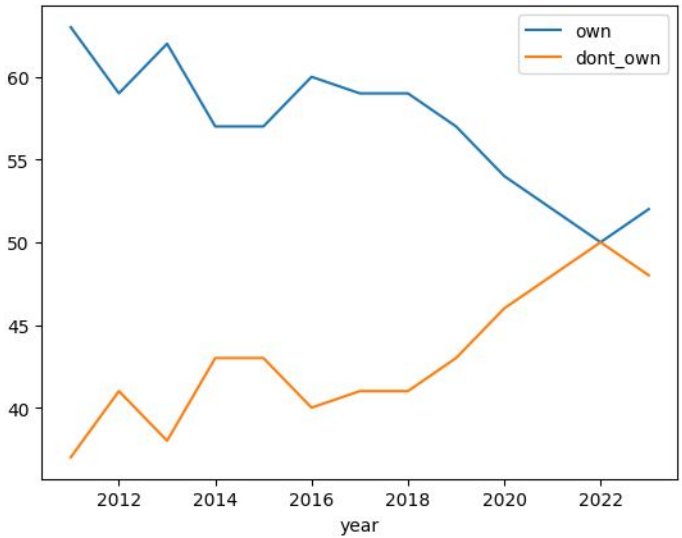


# Consolidando Dados

- A line plot in pandas, like life.plot(), shows trends by plotting data points over a continuous time axis (e.g., years). Each line represents a variable (e.g., percentage of life insurance ownership), with data points connected to reveal changes over time

```
life = pd.read_csv("life_insurance.csv")
life.index = life['year']
life = life.drop(columns=['year'])
life.plot()
```

| | Y2023 |
|---|---|
| **Region** | |
| **Midwest** | 663.000000 |
| **Northeast** | 758.666667 |
| **South** | 696.750000 |
| **West** | 659.076923 |



# Exercício: Pandas

- Apresente em um gráfico de barras a quantidade total (soma) por produto

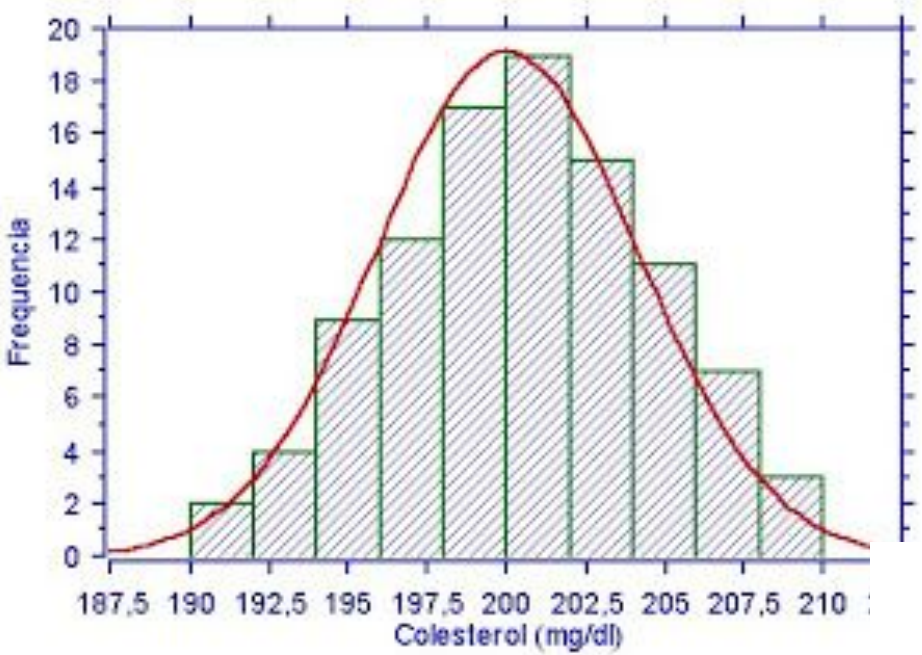
- Apresente em um gráfico de barras a quantidade total (soma) por departamento

| Produto *(character)* | Depto *(character)* | Quantidade *(integer)* |
|---|---|---|
| Papel A4 | ADM | 3 |
| Grampo | ADM | 2 |
| Lápis | VENDAS | 3 |
| Caneta Azul | RH | 10 |

https://raw.githubusercontent.com/lcbjrrr/quant/master/pedidos%20-%20qtd.csv

# Análise Estatística

# Estatística é sobre ...




# R: Medidas de Tendência Central e Dispersão

- Tendência Central: **Média** e **Mediana**

```
print(auto['Y2023'].mean())
print(auto['Y2023'].median())
```

- Dispersão: **Desvio Padrão** e **Variância**

```
print(auto['Y2023'].std())
print(auto['Y2023'].var())
```

# Variáveis Qualitativas e Quantitativas

**Qualitativas**

***Nominais***

-> notas_turma['Curso']

***Ordinais***

-> notas_turma['Conceito']

**Quantitativas**

***Contínuas***

-> notas_turma['Nota']

***Discretas***

-> números inteiros

| **Qualitativas** | |
|---|---|
| ***Nominais*** | categóricas |
| ***Ordinais*** | categóricas ordenadas |
| **Quantitativas** | |
| ***Discretas*** | números inteiros |
| ***Contínuas*** | números decimais |

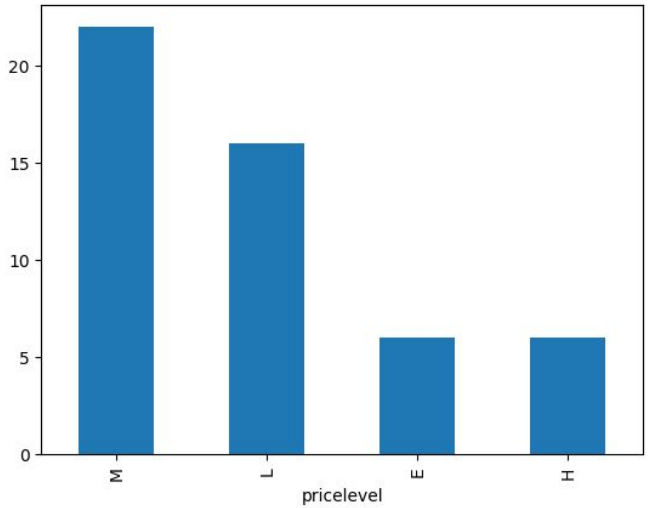
# Histograma: Variáveis Categóricas / Qualitativas

- Pode-se utilizar simplesmente o método **value_counts**()

```
cat_freq = auto['pricelevel'].value_counts()
cat_freq
```

- Daí só apresentar em gráfico de barras.

```
cat_freq.plot(kind='bar')
```



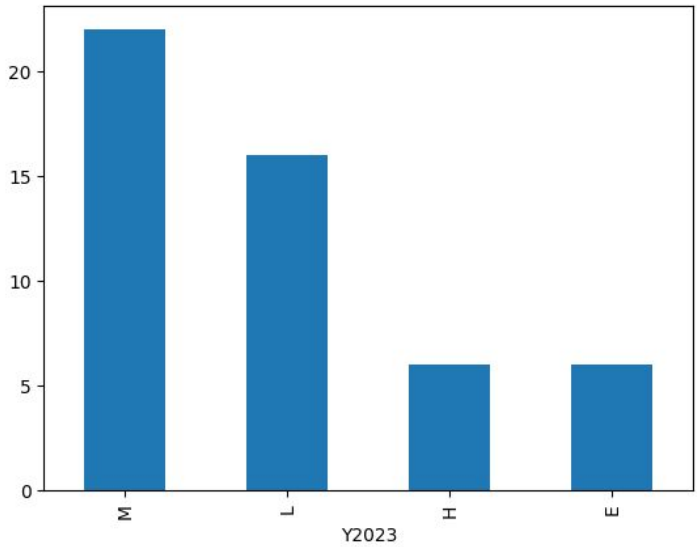
# Histograma: Variáveis Numéricas/Quantitativas

- Para calcular a frequência, use a funcao **value_counts**() combinada com a **cut**().

```
pd.value_counts(pd.cut(auto['Y2023'],bins=[0,500,800,1000,5000],labels=['L','M','H','E']))
```

- Dai basta usar **.plot.bar()**

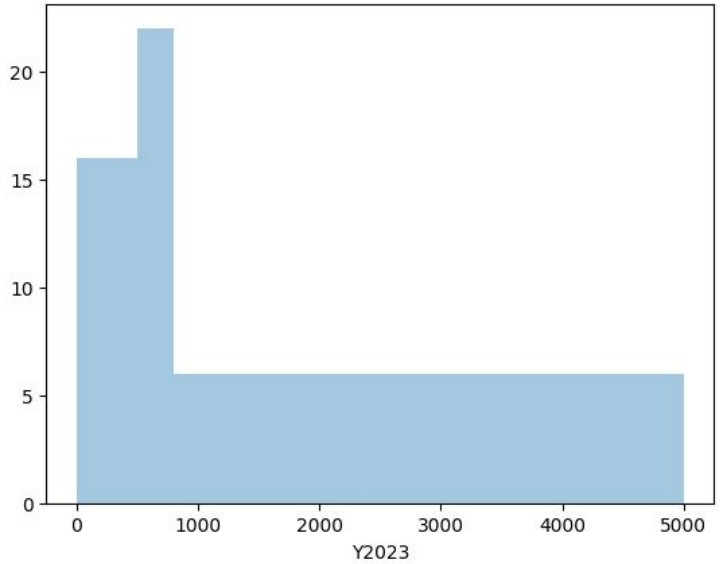
```
num_freq.plot(kind='bar')
```



# Histograma: Variáveis Numéricas/Quantitativas

Opções mais simples podem ser pelas bibliotecas como ***seaborn***

```
import seaborn as sns
sns.distplot(auto['Y2023'], bins=[0,500,800,1000,5000],kde=False)
```



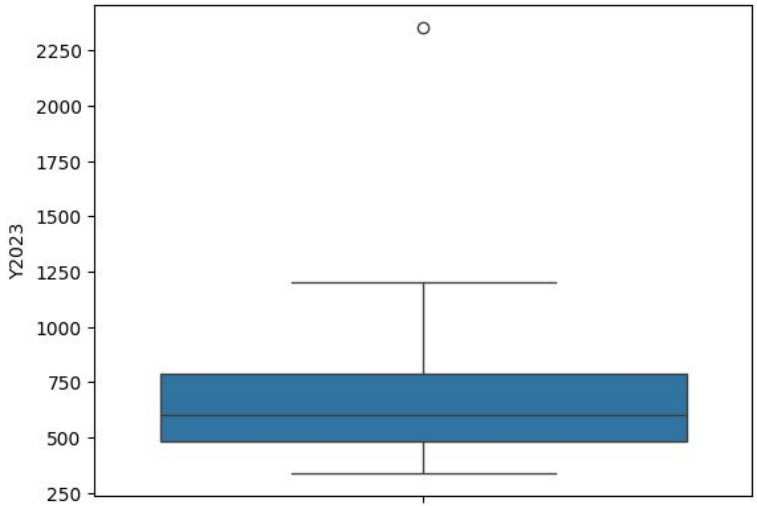
# Boxplot

- Basta usar a função **boxplot()** do **seaborn**.

```
auto['Y2023'].describe()
```

- E um resumo estatístico pode ser obtido com a função, **describe()**.

```
sns.boxplot(auto['Y2023'])
```



# Exercício: Estatística

- Escolha uma base com a seguinte base de dados
- Defina categorias (bins), muito baixo, baixo, médio, alto, muito alto 140,150,155,165,175,200
- Apresente o histograma
- Calcule os quartis
- Apresente o boxplot

| altura_masc *(integer)* |
|---|
| 187 |
| 157 |
| 172 |
| 181 |
| 179 |
| 157 |
| 172 |
| 183 |

Disponível no drive público
dados:https://raw.githubusercontent.com/lcbjrrr/quant/master/natacao%20-%20m.csv


MAD MEN
Intuición · Creatividad · Appeal
MATH MEN
Analytics · Resultados · Lógica
Google

# ATIVIDADE (Em grupo): EDA

- Escolha uma base de dados no https://www.kaggle.com/datasets, e se familiarize com sua base
- Faca análises estatísticas
    - Apresente o histograma
    - Calcule os quartis
    - Apresente o boxplot
- Não esqueça de junto com seus códigos realizar suas análises/conclusões (use o botão de +Texto).