Desafio 14

```r
cat("Arquivo compilado em:", format(Sys.time(), "%d/%m/%Y às %H:%M:%S"), "\n")
```

```
Arquivo compilado em: 30/10/2025 às 11:51:46
```

```r
library(reticulate)
py_install("pyarrow")
```

```
Using virtual environment "C:/Users/Felipe/Documents/.virtualenvs/r-reticulate" ...

+ "C:/Users/Felipe/Documents/.virtualenvs/r-reticulate/Scripts/python.exe" -m pip install --u
```

```r
import polars as pl
```

```r
diamonds = pl.read_csv("diamonds.csv.gz")
print(diamonds)
```

```
shape: (53_940, 10)

 carat   cut        color   clarity   …   price   x      y      z
 ---     ---        ---     ---           ---     ---    ---    ---
 f64     str        str     str           i64     f64    f64    f64

 0.23    Ideal      E       SI2       …   326     3.95   3.98   2.43
 0.21    Premium    E       SI1       …   326     3.89   3.84   2.31
 0.23    Good       E       VS1       …   327     4.05   4.07   2.31
 0.29    Premium    I       VS2       …   334     4.2    4.23   2.63
 0.31    Good       J       SI2       …   335     4.34   4.35   2.75
 …       …          …       …         …   …       …      …      …
 0.72    Ideal      D       SI1       …   2757    5.75   5.76   3.5
```

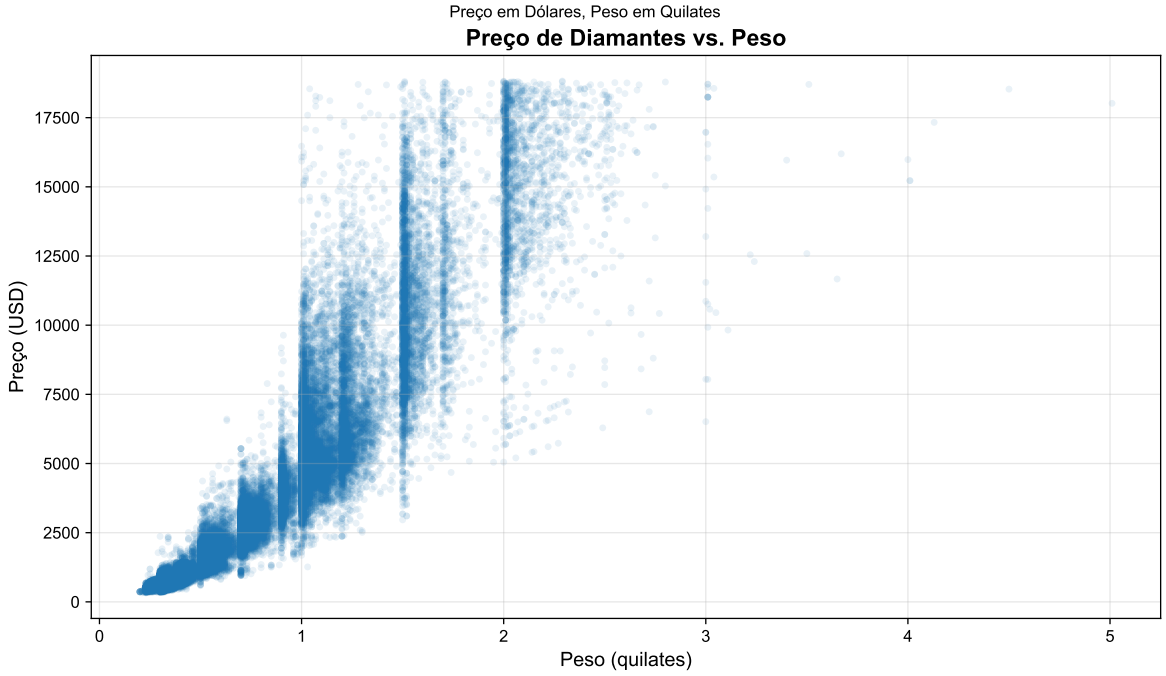
```
  0.72    Good        D       SI1       …   2757    5.69   5.75   3.61
  0.7     Very Good   D       SI1       …   2757    5.66   5.68   3.56
  0.86    Premium     H       SI2       …   2757    6.15   6.12   3.74
  0.75    Ideal       D       SI2       …   2757    5.83   5.87   3.64
```

```
diamonds = diamonds.with_columns(
  pl.col("cut").cast(pl.Enum(["Fair", "Good", "Very Good", "Premium", "Ideal"])),
  pl.col("color").cast(pl.Enum(["D", "E", "F", "G", "H", "I", "J"])),
  pl.col("clarity").cast(pl.Enum(["IF", "VVS1", "VVS2", "VS1", "VS2", "SI1", "SI2", "I1"]))
)
print(diamonds)
```

```
shape: (53_940, 10)

  carat   cut         color   clarity   …   price   x      y      z
  ---     ---         ---     ---           ---     ---    ---    ---
  f64     enum        enum    enum          i64     f64    f64    f64

  0.23    Ideal       E       SI2       …   326     3.95   3.98   2.43
  0.21    Premium     E       SI1       …   326     3.89   3.84   2.31
  0.23    Good        E       VS1       …   327     4.05   4.07   2.31
  0.29    Premium     I       VS2       …   334     4.2    4.23   2.63
  0.31    Good        J       SI2       …   335     4.34   4.35   2.75
  …       …           …       …         …   …       …      …      …
  0.72    Ideal       D       SI1       …   2757    5.75   5.76   3.5
  0.72    Good        D       SI1       …   2757    5.69   5.75   3.61
  0.7     Very Good   D       SI1       …   2757    5.66   5.68   3.56
  0.86    Premium     H       SI2       …   2757    6.15   6.12   3.74
  0.75    Ideal       D       SI2       …   2757    5.83   5.87   3.64
```

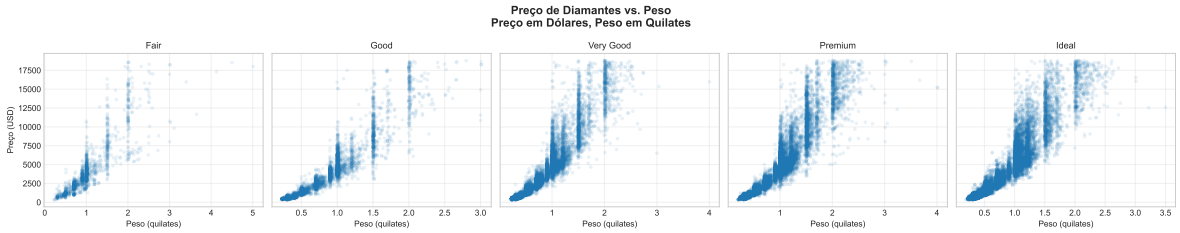
```
import matplotlib.pyplot as plt

# Criar o gráfico
plt.figure(figsize=(10, 6))
plt.scatter(diamonds['carat'], diamonds['price'], alpha=0.1, s=10)

# Configurar labels e título
plt.xlabel('Peso (quilates)', fontsize=12)
plt.ylabel('Preço (USD)', fontsize=12)
plt.title('Preço de Diamantes vs. Peso', fontsize=14, fontweight='bold')
```

```
plt.suptitle('Preço em Dólares, Peso em Quilates', y=0.96, fontsize=10)

# Estilo similar ao theme_bw()
plt.grid(True, alpha=0.3)
plt.style.use('seaborn-v0_8-whitegrid')

plt.tight_layout()
plt.show()
```




```
diamonds = diamonds.with_columns(
    pl.col("cut").cast(pl.Enum(["Fair", "Good", "Very Good", "Premium", "Ideal"])),
    pl.col("color").cast(pl.Enum(["D", "E", "F", "G", "H", "I", "J"])),
    pl.col("clarity").cast(pl.Enum(["IF", "VVS1", "VVS2", "VS1", "VS2", "SI1", "SI2", "I1"]))
)

# Converter para pandas
diamonds_pd = diamonds.to_pandas()

# Criar subplots para cada categoria de "cut"
cuts = ["Fair", "Good", "Very Good", "Premium", "Ideal"]
n_cuts = len(cuts)
```

```
fig, axes = plt.subplots(1, n_cuts, figsize=(20, 4), sharey=True)
fig.suptitle('Preço de Diamantes vs. Peso\nPreço em Dólares, Peso em Quilates',
             fontsize=14, fontweight='bold')

for i, cut in enumerate(cuts):
    data_subset = diamonds_pd[diamonds_pd['cut'] == cut]
    axes[i].scatter(data_subset['carat'], data_subset['price'], alpha=0.1, s=10)
    axes[i].set_title(cut, fontsize=11)
    axes[i].set_xlabel('Peso (quilates)', fontsize=10)
    axes[i].grid(True, alpha=0.3)

axes[0].set_ylabel('Preço (USD)', fontsize=10)
plt.tight_layout()
plt.show()
```




```
diamonds = diamonds.with_columns(
    pl.col("cut").cast(pl.Enum(["Fair", "Good", "Very Good", "Premium", "Ideal"])),
    pl.col("color").cast(pl.Enum(["D", "E", "F", "G", "H", "I", "J"])),
    pl.col("clarity").cast(pl.Enum(["IF", "VVS1", "VVS2", "VS1", "VS2", "SI1", "SI2", "I1"]))
)

# Criar coluna "clareza" numérica
diamonds = diamonds.with_columns(
    (8 - pl.col("clarity").to_physical()).alias("clareza")
)

# Converter para pandas
diamonds_pd = diamonds.to_pandas()

# Definir ordem das categorias
cuts = ["Fair", "Good", "Very Good", "Premium", "Ideal"]
colors = ["D", "E", "F", "G", "H", "I", "J"]

# Criar grid de subplots com MAIS ESPAÇO
```

```
fig, axes = plt.subplots(len(cuts), len(colors),
                         figsize=(24, 18),  # Aumentado
                         sharex=True, sharey=True)

# Ajustar espaçamento entre subplots
plt.subplots_adjust(hspace=0.3, wspace=0.3)

fig.suptitle('Preço de Diamantes vs. Peso\nPreço em Dólares, Peso em Quilates',
             fontsize=18, fontweight='bold', y=0.998)

# Plotar cada combinação
for i, cut in enumerate(cuts):
    for j, color in enumerate(colors):
        ax = axes[i, j]
        data_subset = diamonds_pd[(diamonds_pd['cut'] == cut) &
                                  (diamonds_pd['color'] == color)]

        if len(data_subset) > 0:
            scatter = ax.scatter(data_subset['carat'],
                                 data_subset['price'],
                                 c=data_subset['clareza'],
                                 cmap='coolwarm',
                                 alpha=0.15,  # Aumentado para ver melhor
                                 s=15,  # Pontos maiores
                                 vmin=1, vmax=8)

        # Títulos nas bordas - MAIORES
        if i == 0:
            ax.set_title(f'Color: {color}', fontsize=13, fontweight='bold', pad=10)
        if j == 0:
            ax.set_ylabel(f'{cut}\n\nPreço (USD)', fontsize=11, fontweight='bold')
        if i == len(cuts) - 1:
            ax.set_xlabel('Peso (quilates)', fontsize=11)

        ax.grid(True, alpha=0.3, linewidth=0.5)
        ax.tick_params(labelsize=9)

# Adicionar colorbar maior e mais visível
cbar = fig.colorbar(scatter, ax=axes, orientation='vertical',
                    fraction=0.015, pad=0.02, aspect=30)
cbar.set_label('Clareza', fontsize=14, fontweight='bold')
cbar.set_ticks(range(1, 9))
```

```
cbar.set_ticklabels(['1', '2', '3', '4', '5', '6', '7', '8'])
cbar.ax.tick_params(labelsize=11)

plt.show()
```