FICHA TRABALHO 3

MODALIDADE:	Aprendizagem +	Não aplicável	
CURSO:	Técnico/a Especialista em Gestão de Informação e Ciência dos Dados		
UFCD:	Visualização de dados em Python	CÓDIGO UFCD:	10809
FORMADOR/A:	Bruno Silva	DATA:	

OBJETIVOS

- Análise de dados com gráficos
- Storytelling - enquadramento de um problema através de visualização de dados

Para esta ficha de trabalho, vamos pegar num dataset existente no portal Kaggle e vamos construir agrupamentos de informação para fazer o um enquadramento de um problema através de visualização de dados.

O ficheiro "Coffee_sales.csv" é um ficheiro que tem as informações sobre registos de transações de cafeteria, incluindo detalhes sobre vendas, tipo de pagamento, tempo de compra e preferências do cliente. Com atributos que abrangem a hora do dia, dias úteis, meses, tipos de café e receita, esse conjunto de dados fornecem uma base sólida para analisar o comportamento do cliente, padrões de vendas e tendências de desempenho de negócios.

O ficheiro "Coffee_sales.csv" é um ficheiro constituído pelos seguintes campos:

1. hour_of_day → Hour da compra (0–23)
2. cash_type → Modo de pagamento (cash / card)
3. money → Valor da transação
4. coffee_name → Tipo de café que foi comprado (ex: Latte, Americano, Hot Chocolate)
5. Time_of_Day → Período do dia da compra (Manhã (Morning), Tarde (Afternoon) ou Noite (Night))
6. Weekday → Dia da Semana (ex: Mon, Tue, …)
7. Month_name → Mês da compra (ex: Jan, Feb, Mar)
8. Weekdaysort → Representação numérica para dias da semana (1 = Seg., 7 = Domingo)
9. Monthsort → Representação numérica para meses (1 = Jan., 12 = Dez.)
10. Date → Data da transação (YYYY-MM-DD → Ano-Mês-Dia)
11. Time → Hora da transação (HH:MM:SS)

Coffee_sales

hour_of_day	cash_type	money	coffee_name	Time_of_Day	Weekday	Month_name	Weekdaysort	Monthsort	Date	Time
10	card	38.7	Latte	Morning	Fri	Mar	5	3	2024-03-01	10:15:50.520000
12	card	38.7	Hot Chocolate	Afternoon	Fri	Mar	5	3	2024-03-01	12:19:22.539000
12	cash	38.7	Hot Chocolate	Afternoon	Fri	Mar	5	3	2024-03-01	12:20:18.089000

Com base nesta informação, podemos fazer agrupamentos de informação para realizar a construção dos seguintes dados visuais:

- 📊 Vendas por Tipo de Café (por exemplo, bebidas mais vendidas)
- ⏰ Vendas por Hora do Dia (pico horário comercial)
- 🌅 Vendas por Hora do Dia (Tendências da Manhã vs Tarde vs Noite)
- 📆 Vendas por Weekday & Month
- 💳 Qual o tipo do método de pagamento (dinheiro vs uso do cartão)
- 📈 Tendências de Receita Ao Longo do Tempo (análise diária/mensal de crescimento)

1 – Faça a importação das bibliotecas Matplotlib e Pandas:

```
import pandas as pd
import matplotlib.pyplot as mat
```

2 – Crie a variável **data** e com a função read_csv() deve **carregar** a informação do ficheiro "**Coffee_sales.csv"** com o **delimitador** , (vírgula);

```
data = pd.read_csv("Coffee_sales.csv", sep=",")
```

3 – Quando pretendemos saber quais são os valores únicos e sem repetições de uma coluna (para resumir a informação), devemos utilizar **duas funcionalidades da biblioteca Pandas**.

Para este exemplo, temos a coluna "coffee_name" que tem o registo dos **vários tipos de cafés**. Como tal, vamos utilizar **2 funções para resumir os dados da coluna e devolver dados únicos** (sem repetições), com as funções:

- **nunique()** → Saber a quantidade de dados de forma única (sem repetições)
- **unique()** → Saber os dados de forma única (sem repetições)

```
#Saber a lista de cafés

#Saber a quantidade de dados de forma única (sem repetições)
quantidade_cafes = data["coffee_name"].nunique()

#Saber os dados de forma única (sem repetições)
lista_cafes = data["coffee_name"].unique()
```

No final mostre a seguinte mensagem:

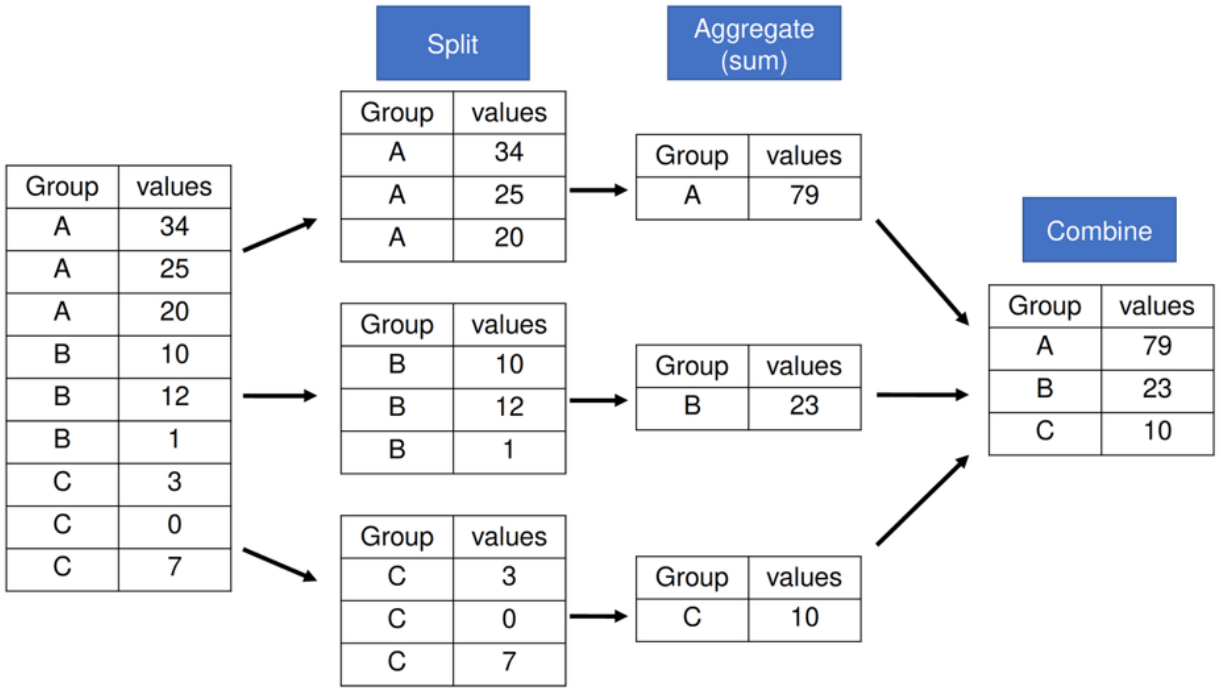
```
print(f"Existem {quantidade_cafes} tipos de café: {lista_cafes}")
```

Se tudo correr bem, será exibida a seguinte informação:

```
Existem 8 tipos de café: ['Latte' 'Hot Chocolate' 'Americano' 'Americano with Milk' 'Cocoa
 'Cortado' 'Espresso' 'Cappuccino']
```

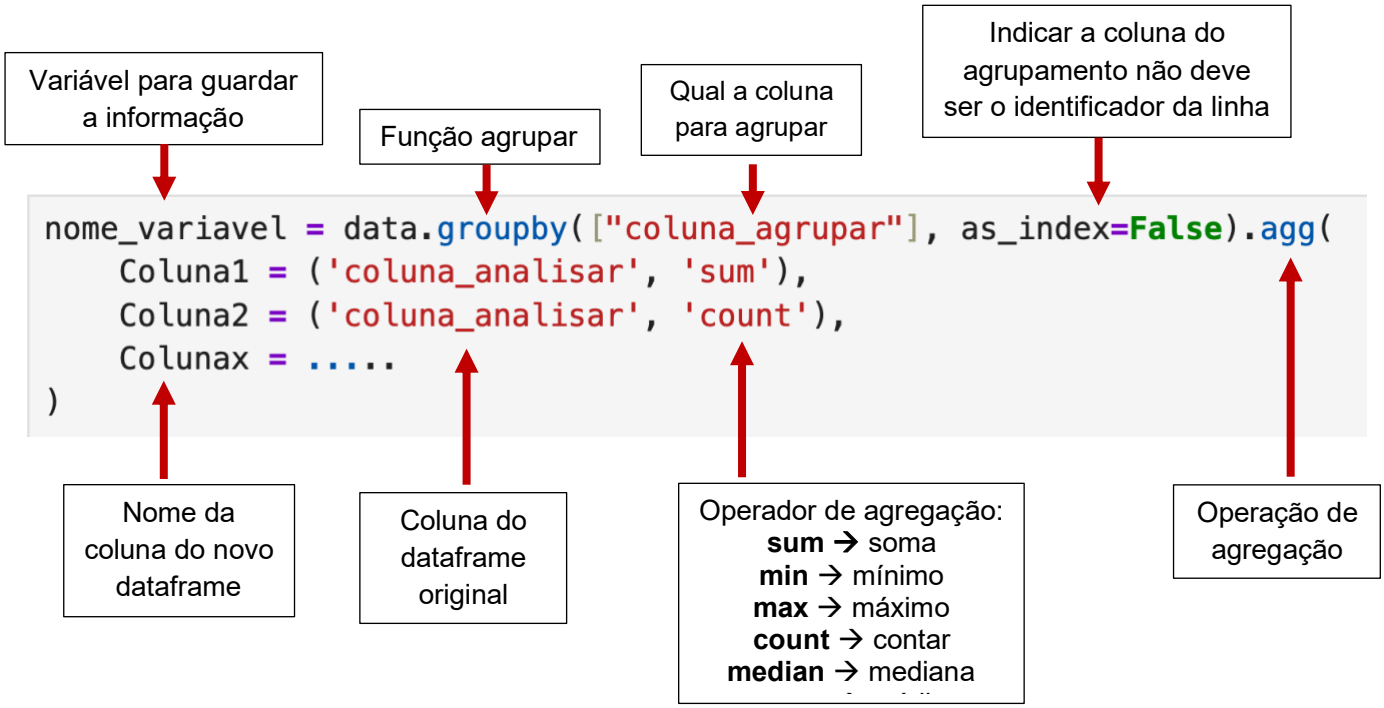
4 – Para ajudar a **construir gráficos visuais com agrupamentos de dados**, podemos usar a função da biblioteca pandas **groupby()** para indicar qual é o campo que pretendemos agrupar a informação. Esta funcionalidade permite agrupar todos os dados com a mesma informação e a partir daí podemos indicar que tipos de métodos que devem ser aplicados, como por exemplo, soma, contagem, média, mediana, entre outros;

Exemplificação visual do processo do agrupamento:



4.1 – O próximo passo será fazer o agrupamento da informação do tipo de café "coffee_name" para representar mais do que um valor de agrupamento de dados, ou seja, as **vendas monetárias** e **unidades vendidas** do produto. Neste caso, precisamos de utilizar a função do agrupamento com uma nova funcionalidade chamada agg() (agregação de informação).

Como tal, vamos utilizar a seguinte expressão:



Exemplo: Agrupar dados da coluna "coffee_name" e calcular as vendas (lucros monetários) e unidades vendidas de cada café:

```
#Agrupar coluna coffee_name e calcular as vendas (sum) e unidades vendidas (count)
vendas_quantidades = data.groupby(["coffee_name"], as_index=False).agg(
    Vendas = ('money', 'sum'),
    Unidades_Vendidas = ('money', 'count'),
)
```

Neste exemplo, a variável "vendas_quantidades" vão agrupar os dados da coluna "coffee_name". No entanto, **reparem na diferença ao utilizar o parâmetro as_index** do qual vai indicar qual é o campo do índice, mais especificamente:

- **as_index = True** (vai indicar que o identificador do agrupamento é a coluna que pediu para agrupar, mas assim será mais difícil representar e extrair informação);
- **as_index = False** (vai criar um identificador para cada agrupamento de informação que fez e separar a informação logicamente);

as_index = True

coffee_name	Vendas	Unidades_Vendidas
Espresso	2690.28	129
Cortado	7384.86	287
Cocoa	8521.16	239
Hot Chocolate	9933.46	276
Americano	14650.26	564
Cappuccino	17439.14	486
Americano with Milk	24751.12	809
Latte	26875.30	757

as_index = False

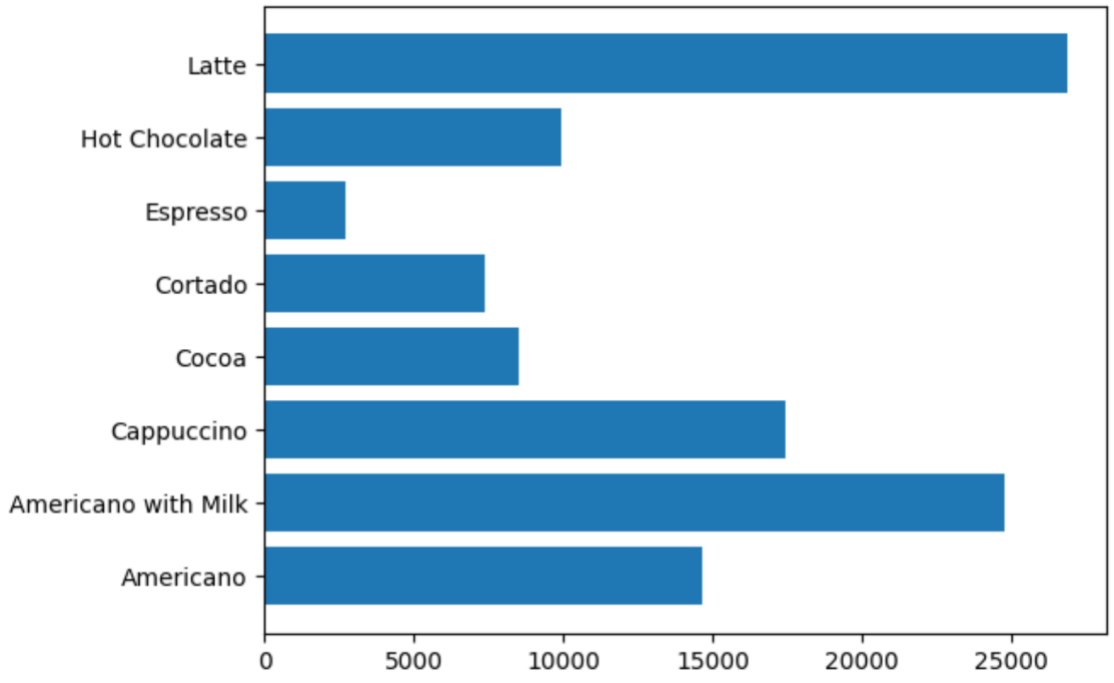
	coffee_name	Vendas	Unidades_Vendidas
5	Espresso	2690.28	129
4	Cortado	7384.86	287
3	Cocoa	8521.16	239
6	Hot Chocolate	9933.46	276
0	Americano	14650.26	564
2	Cappuccino	17439.14	486
1	Americano with Milk	24751.12	809
7	Latte	26875.30	757

Resumindo, será melhor utilizar a segunda forma, pois podemos aceder diretamente aos novos campos que gerou na nova dataframe.

Depois disso, vamos mostrar os dados do novo dataframe com os dados agrupados:

```
#mostrar um gráfico de barras horizontais (barh) com os dados do novo dataframe
mat.barh(vendas_quantidades["coffee_name"], vendas_quantidades["Vendas"])
```

Resultado:

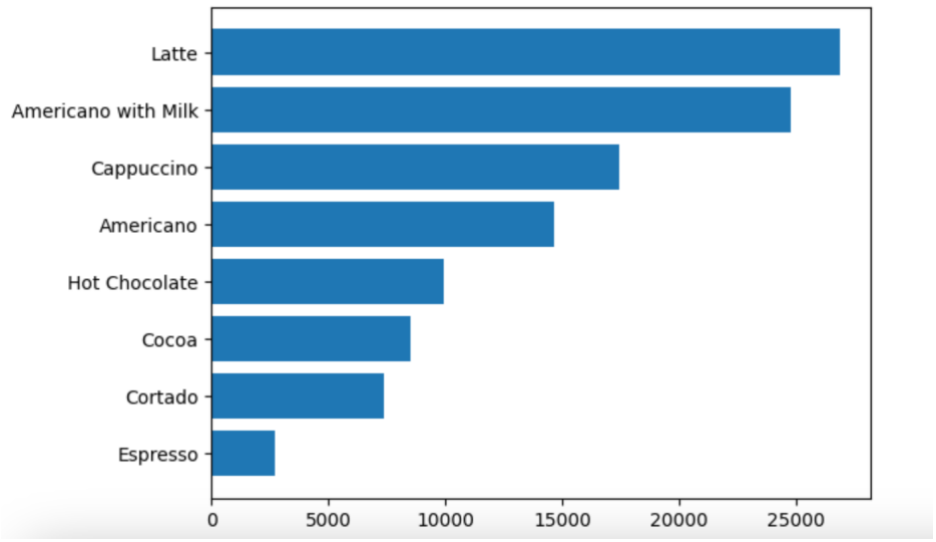


Caso deseje **ordenar os dados** por ordem crescente, basta acrescentar no final dos parênteses da agregação a função sort_values e dentro da função indicar dois parâmetros:

- Parâmetro **by**: indicar qual a coluna a considerar na ordenação;
- Parâmetro **ascending**: indicar duas possibilidades:
  - **True** → ordenar de forma **crescente**;
  - **False** → ordenar de forma **decrescente**;

```
#Agrupar coluna coffee_name e calcular as vendas (sum) e unidades vendidas (count)
vendas_quantidades = data.groupby(["coffee_name"], as_index=False).agg(
    Vendas = ('money', 'sum'),
    Unidades_Vendidas = ('money', 'count'),
).sort_values(by='Vendas',ascending=True)
```

Resultado:



Exercícios:

1 – Neste exercício vamos analisar quais as percentagens de tipos de pagamentos utilizados na compra dos vários tipos de cafés.

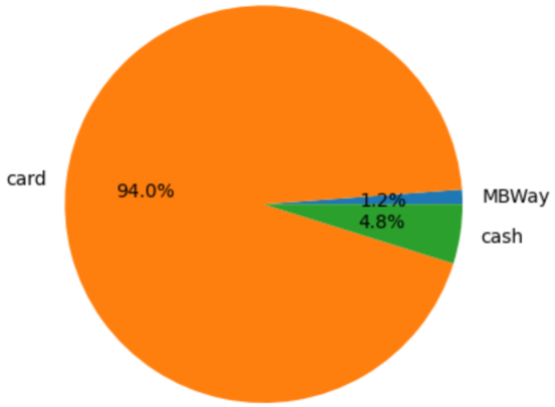
Crie uma variável com o nome "tipos_pagamento" e faça:

- **o agrupamento de dados** pela coluna "cash_type";
- **o índice** deve estar como falso;
- **fazer a agregação com a função agg()** dos dados em que deve criar as seguintes colunas:
    - **coluna** "Tipos" e fazer a **contagem** de dados da coluna "cash_type";

No final, deve criar um **gráfico do tipo circular (pie())** com os seguintes **parâmetros**:

- **1º Parâmetro:** colocar o campo **tipos_pagamento["Tipos"]** para representar as fatias do gráfico circular;
- **2º Parâmetro:** colocar o campo **tipos_pagamento["cash_type"]** para associar os nomes a cada fatia do parâmetro anterior;
- **3º Parâmetro:** converter o valor numérico em percentagem com a instrução **autopct="%1.1f%%"**

Resultado esperado:



2 – Neste exercício vamos analisar as vendas dos cafés por partes do dia, utilizando a coluna "Time_of_Day", do qual refere para cada registo se a venda foi de Manhã (Morning), Tarde (Afternoon) ou Noite (Night).

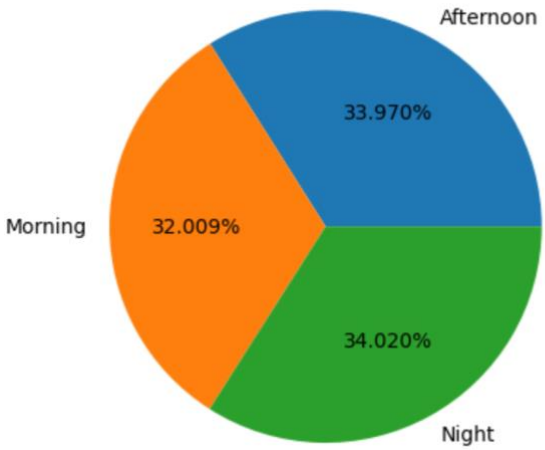
Crie uma variável com o nome "vendas_partes_dia" e faça:

- **o agrupamento de dados** pela coluna "Time_of_Day";
- **o índice** deve estar como falso;
- **fazer a agregação com a função agg()** dos dados em que deve criar as seguintes colunas:
  - **coluna** "Vendas" e fazer a **soma** de dados da coluna "money";
  - **coluna** "Unidades_Vendidas" e fazer a **contagem** de dados da coluna "money";

No final, deve criar um **gráfico do tipo circular (pie())** com os seguintes **parâmetros**:

- **1º Parâmetro:** colocar o campo **vendas_partes_dia["Vendas"]** para representar as fatias do gráfico circular;
- **2º Parâmetro:** colocar o campo **vendas_partes_dia["Time_of_Day"]** para associar os nomes a cada fatia do parâmetro anterior;
- **3º Parâmetro:** converter o valor numérico em percentagem com a instrução **autopct="%1.1f%%"**

Resultado esperado:



3 – Neste exercício vamos analisar as vendas dos cafés pelas várias horas do dia, utilizando a coluna "hour_of_day", do qual refere para cada registo a hora em que ocorreu a venda do café.

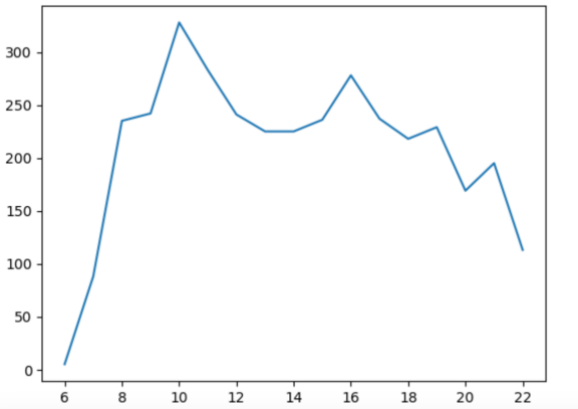
Crie uma variável com o nome "vendas_por_hora" e faça:

- **o agrupamento de dados** pela coluna "hour_of_day";
- **o índice** deve estar como falso;
- **fazer a agregação com a função agg()** dos dados em que deve criar as seguintes colunas:
  - **coluna** "Unidades_Vendidas" e fazer a **contagem** de dados da coluna "money";

No final, deve criar um **gráfico de linhas (plot())** com os seguintes **parâmetros**:

- **1º Parâmetro:** colocar o campo **vendas_por_hora["hour_of_day"]** para representar os valores das horas no eixo das abcissas;
- **2º Parâmetro:** colocar o campo **vendas_por_hora["Unidades_Vendidas"]** para representar os valores das unidades vendidas no eixo das ordenadas;

Resultado esperado:

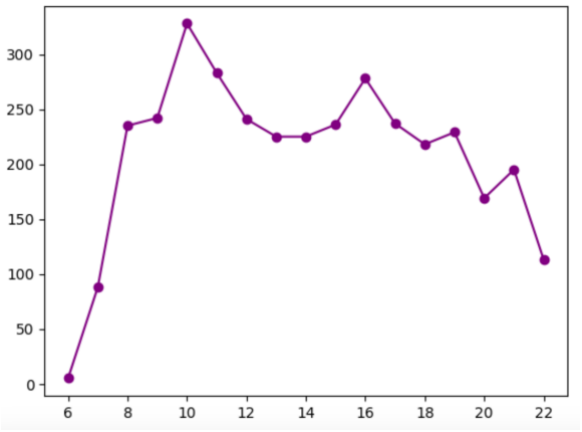


Opcionalmente, pode personalizar o gráfico como por exemplo:

Opção 1 - Mudar cor da linha e colocar o ponto de interseção, adicionando **2 parâmetros adicionais na instrução plot()**:

- **3º Parâmetro:** colocar o campo **marker='o'** para representar um círculo no ponto de interseção;
- **4º Parâmetro:** colocar o campo **color** para representar a cor da linha;

```
#gráfico de linhas
mat.plot(vendas_por_hora["hour_of_day"], vendas_por_hora["Unidades_Vendidas"], marker='o', color='purple')
```



Opção 2 – Colocar os valores de cada ponto de interseção no gráfico com o método genérico para o gráfico de linhas.

#estrutura para representar dados nos pontos de interseção

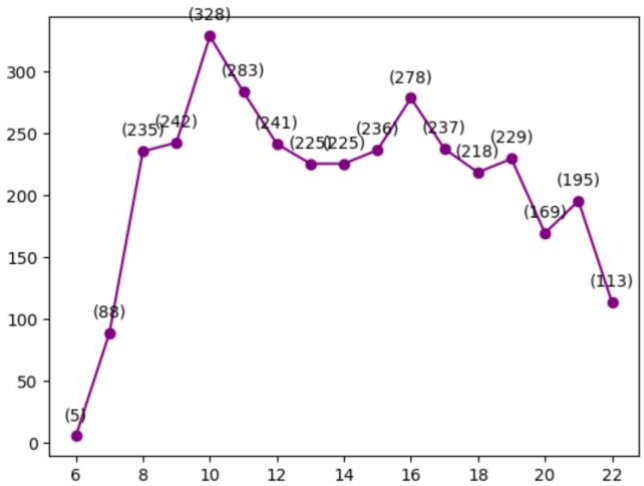
for i, (x, y) in enumerate(zip(vendas_por_hora["hour_of_day"], vendas_por_hora["Unidades_Vendidas"])):

mat.annotate(f"({y})", (x, y), textcoords="offset points", xytext=(0, 10), ha="center")

```
vendas_por_hora = data.groupby(["hour_of_day"], as_index=False).agg(
    Unidades_Vendidas=('money', 'count')
)

for i, (x, y) in enumerate(zip(vendas_por_hora["hour_of_day"], vendas_por_hora["Unidades_Vendidas"])):
    mat.annotate(f"({y})", (x, y), textcoords="offset points", xytext=(0, 10), ha="center")

#gráfico de linhas
mat.plot(vendas_por_hora["hour_of_day"], vendas_por_hora["Unidades_Vendidas"], marker='o', color='purple')
```



4 – Neste exercício vamos analisar as vendas dos cafés por dias da semana, utilizando a coluna "Weekday", do qual refere para cada registo qual o dia da semana que foi vendido os vários cafés.

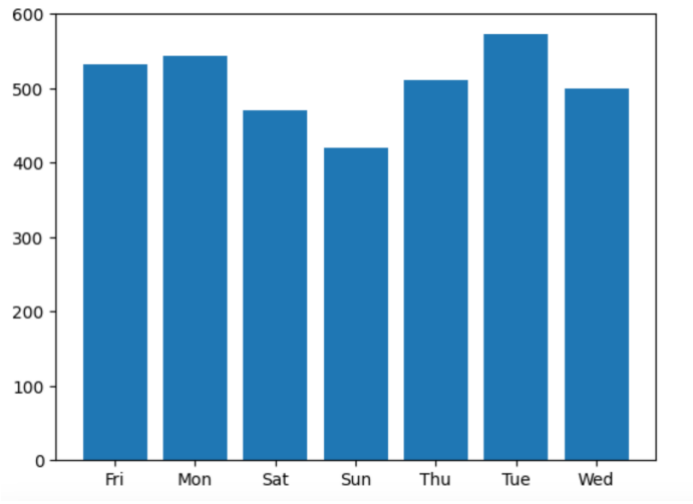
Crie uma variável com o nome "vendas_dia" e faça:

- **o agrupamento de dados** pela coluna "Weekday";
- **o índice** deve estar como falso;
- **fazer a agregação com a função agg()** dos dados em que deve criar as seguintes colunas:
  - **coluna** "Contagem" e fazer a **contagem** de dados da coluna "Weekday";

No final, deve criar um **gráfico de barras (bar())** com os seguintes **parâmetros**:

- **1º Parâmetro:** colocar o campo **vendas_dia["Weekday"]** para representar os valores das dos dias da semana no eixo das abcissas;
- **2º Parâmetro:** colocar o campo **vendas_dia["Contagem"]** para representar os valores das unidades vendidas no eixo das ordenadas;

Resultado esperado:



Opcionalmente, pode personalizar o gráfico como por exemplo:

Opção 1 – Criar uma rotação dos dados do eixo das abcissas para mostrar dados dos nomes sem haver sobreposição. Como tal, utilize a função **xticks**(rotation=x), do qual x representa o ângulo da rotação:

```
mat.xticks(rotation=45)
```

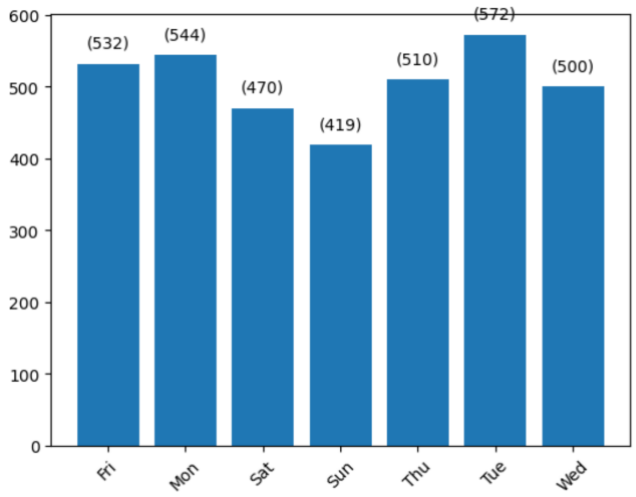
Opção 2 – Colocar os valores de cada ponto de interseção no gráfico com o método genérico para o gráfico de linhas.

for i, (x, y) in enumerate(zip(vendas_dia["Weekday"], vendas_dia["Contagem"])):

mat.annotate(f"({y})", (x, y), textcoords="offset points", xytext=(0, 10), ha="center")

```
for i, (x, y) in enumerate(zip(vendas_dia["Weekday"], vendas_dia["Contagem"])):
    mat.annotate(f"({y})", (x, y), textcoords="offset points", xytext=(0, 10), ha="center")
```

Resultado:



5 – Neste exercício vamos analisar as vendas dos cafés por mês, utilizando a coluna “Monthsort”.

5.1 - Crie uma variável com o nome “vendas_mes” e faça:

- **o agrupamento de dados** pela coluna “Monthsort”;
- **o índice** deve estar como falso;
- **fazer a agregação com a função agg()** dos dados em que deve criar as seguintes colunas:
  - **coluna** “Vendas” e fazer a **soma** de dados da coluna “money”;

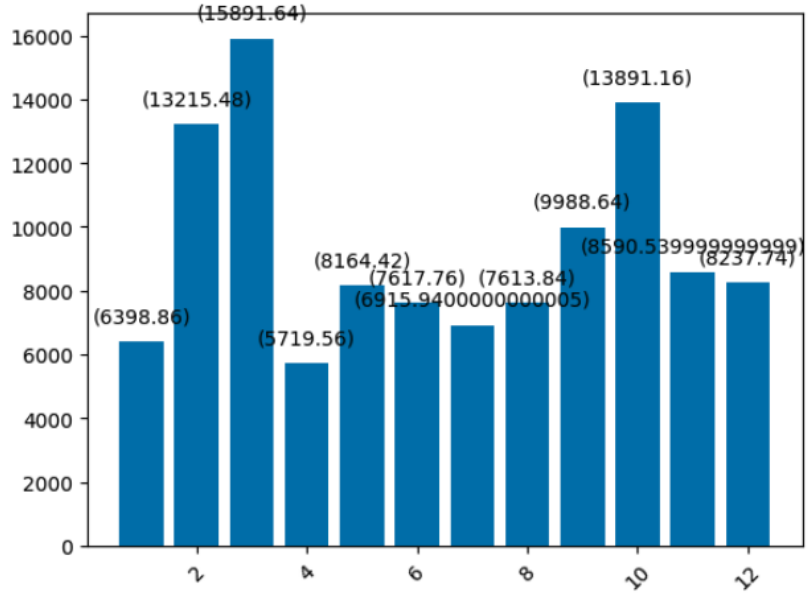
No final, deve criar um **gráfico de barras (bar())** com os seguintes **parâmetros**:

- **1º Parâmetro:** colocar o campo **vendas_mes["Monthsort"]** para representar os valores das dos dias da semana no eixo das abcissas;
- **2º Parâmetro:** colocar o campo **vendas_mes["Vendas"]** para representar os valores das unidades vendidas no eixo das ordenadas;

No final, antes de mostrar o gráfico, deve colocar os valores de cada ponto de interseção no gráfico com o método genérico para o gráfico de linhas.

```
for i, (x, y) in enumerate(zip(vendas_dia["Weekday"], vendas_dia["Contagem"])):
    mat.annotate(f"({y})", (x, y), textcoords="offset points", xytext=(0, 10), ha="center")
```

Resultado esperado:



5.2 – No entanto, podemos filtrar a informação combinar o agrupamento com mais de uma coluna. Até agora, apenas utilizamos a função groupby() com 1 coluna. No entanto podem colocar **mais do que um valor para fazer agrupamento por hierarquia**, ou seja, neste caso queremos primeiro agrupar todos os dados pelo nome do café e só depois pelo mês.

Como agora usamos mais de um agrupamento, temos de utilizar as chavetas retas [ ] e indicar os campos a analisar (pela ordem lógica e separados por virgula):

```
#Agrupar dados por 2 colunas (Nome café e depois mês)
#e agregar a contagem das vendas por mês

vendas_mes = data.groupby(["coffee_name","Monthsort"], as_index=False).agg(
    Vendas = ('Monthsort', 'count')
)
```

Quando exibir o agrupamento de dados, temos a seguinte informação:

	coffee_name	Monthsort	Vendas
0	Americano	1	25
1	Americano	2	117
2	Americano	3	134
3	Americano	4	33
4	Americano	5	40
...	...	...	...
91	Latte	8	58
92	Latte	9	94
93	Latte	10	120
94	Latte	11	68
95	Latte	12	47

Neste caso, agrupou os dados em primeiro pelos tipos de café e depois por mês. Se quiser filtrar apenas o café “Americano”, temos de utilizar a função **query()** e pedir apenas para aproveitar essa informação:

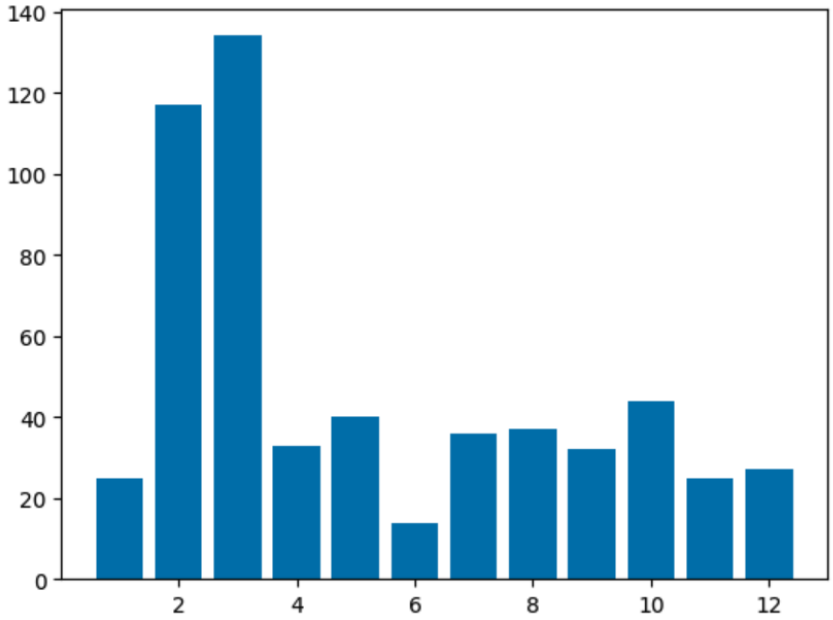
```
#filtrar a informação para mostrar apenas café do tipo "Americano"
vendas_mes = vendas_mes.query(' coffee_name == "Americano" ')
```

Resultado:

	coffee_name	Monthsort	Vendas
0	Americano	1	25
1	Americano	2	117
2	Americano	3	134
3	Americano	4	33
4	Americano	5	40
5	Americano	6	14
6	Americano	7	36
7	Americano	8	37
8	Americano	9	32
9	Americano	10	44
10	Americano	11	25
11	Americano	12	27

Feito isto, podemos construir o gráfico com as novas informações:

```
mat.bar(vendas_mes["Monthsort"], vendas_mes["Vendas"])
```



6 – Neste exercício vamos analisar as vendas num determinado período de tempo.

6.1 – Para este exercício faça a importação da biblioteca datetime;

```
import datetime
```

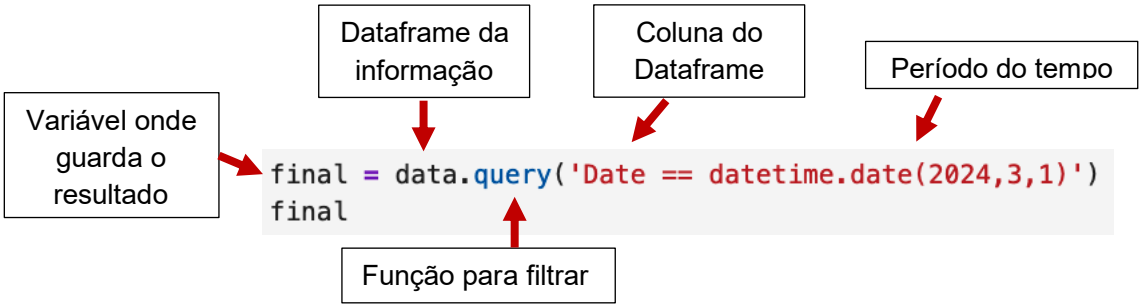
6.2 – As colunas dos ficheiros, normalmente vem em formato de texto. Como tal, vamos garantir que a coluna “Date” fica em formato do tipo Datetime. Faça a conversão do campo com a seguinte instrução:

```
data['Date'] = pd.to_datetime(data['Date'])
```

6.3 – Com a ajuda da função **query()** vamos isolar a informação por períodos de data:

6.3.1 – Mostre todos os dados das vendas que ocorreram no dia 1-3-2024. Quando estiver a comparar o valor deve indicar qual o campo da data e colocar à frente a seguinte instrução:

datetime.date(Ano, Mês, Dia)



Resultado:

	hour_of_day	cash_type	money	coffee_name	Time_of_Day	Weekday	Month_name	Weekdaysort	Monthsort	Date	Time
0	10	card	38.7	Latte	Morning	Fri	Mar	5	3	2024-03-01	10:15:50.520000
1	12	card	38.7	Hot Chocolate	Afternoon	Fri	Mar	5	3	2024-03-01	12:19:22.539000
2	12	cash	38.7	Hot Chocolate	Afternoon	Fri	Mar	5	3	2024-03-01	12:20:18.089000
3	13	card	28.9	Americano	Afternoon	Fri	Mar	5	3	2024-03-01	13:46:33.006000
4	13	card	38.7	Latte	Afternoon	Fri	Mar	5	3	2024-03-01	13:48:14.626000
5	15	card	33.8	Americano with Milk	Afternoon	Fri	Mar	5	3	2024-03-01	15:39:47.726000
6	16	card	38.7	Hot Chocolate	Afternoon	Fri	Mar	5	3	2024-03-01	16:19:02.756000
7	18	card	33.8	Americano with Milk	Night	Fri	Mar	5	3	2024-03-01	18:39:03.580000
8	19	card	38.7	Cocoa	Night	Fri	Mar	5	3	2024-03-01	19:22:01.762000
9	19	MBWay	33.8	Americano with Milk	Night	Fri	Mar	5	3	2024-03-01	19:23:15.887000
10	19	card	33.8	Americano with Milk	Night	Fri	Mar	5	3	2024-03-01	19:29:17.391000

6.3.2 – Se desejar num intervalo entre duas datas, podemos utiliza os operadores relacionais e lógico:

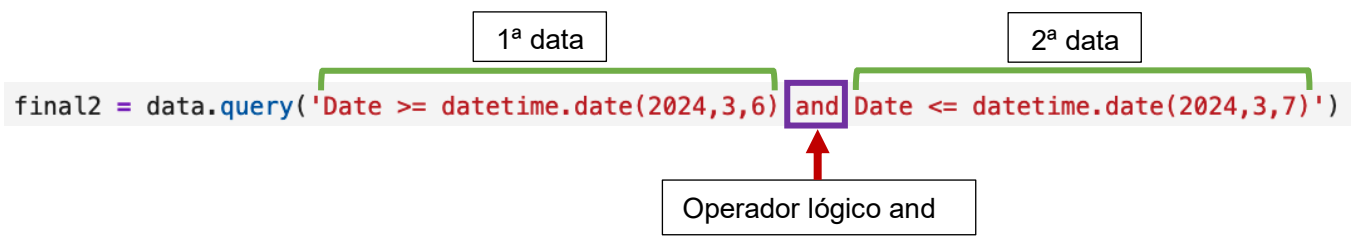
Operadores relacionais

- inferior (<)
- inferior ou igual (<=)
- superior (>)
- superior ou igual (>=)

Operadores relacionais

- and (e)

Neste caso, vamos filtrar as **datas maior ou igual** 6/3/2014 **e menor ou igual** 7/3/2025:



	hour_of_day	cash_type	money	coffee_name	Time_of_Day	Weekday	Month_name	Weekdaysort	Monthsort	Date	Time
39	13	card	28.9	Americano	Afternoon	Wed	Mar	3	3	2024-03-06	13:24:07.667000
40	13	card	28.9	Cortado	Afternoon	Wed	Mar	3	3	2024-03-06	13:25:14.351000
41	14	card	38.7	Cappuccino	Afternoon	Wed	Mar	3	3	2024-03-06	14:52:01.761000
42	14	card	38.7	Cappuccino	Afternoon	Wed	Mar	3	3	2024-03-06	14:53:18.344000
43	10	card	38.7	Hot Chocolate	Morning	Thu	Mar	4	3	2024-03-07	10:18:40.543000
44	11	card	38.7	Latte	Morning	Thu	Mar	4	3	2024-03-07	11:03:58.976000
45	15	card	28.9	Americano	Afternoon	Thu	Mar	4	3	2024-03-07	15:40:22.606000
46	15	card	33.8	Americano with Milk	Afternoon	Thu	Mar	4	3	2024-03-07	15:41:28.784000