FICHA TRABALHO 4

MODALIDADE:	Aprendizagem +	Não aplicável	
CURSO:	Técnico/a Especialista em Gestão de Informação e Ciência dos Dados		
UFCD:	Limpeza e transformação de dados em Python	CÓDIGO UFCD:	10808
FORMADOR/A:	Bruno Silva	DATA:	

OBJETIVOS

- Comandos com a biblioteca Pandas para classificação, divisão e extração de dados

Parte 1 – Classificação, divisão e extração de dados

A biblioteca do Pandas oferece várias maneiras de classificar, dividir, extrair, filtrar e isolar dados nos DataFrames. Vejamos as formas mais comuns.

Para esta ficha de trabalho, vamos usar o ficheiro nba1.csv (que está nesta ficha de trabalho). Como tal, na plataforma do jupyter, deve criar a pasta "ficha_4" e dentro desta o ficheiro "ficha_4.ipynb". Quando estiver dentro do ficheiro da ficha 4 deve fazer a importação do módulo Pandas e do ficheiro CSV com a função read_csv().

Parte 1.1 – *Função sort_values()*

Para classificar um DataFrame por uma coluna específica com o método **sort_values**. Para além disso, temos ainda **2 parâmetros adicionais**:

- **by** → **qual a coluna** que pretende **ordenar** os valores;
- **asceding** → **true** para ordenação ascendente e **false** para descendente;

```
#Ordenar coluna TEAM e por ordem crescente
data_ordenada = data.sort_values(by="Team", ascending=True)
print(data_ordenada)
```

```
                Name                Team  Number Position  ...  Height Weight
327    Walter Tavares       Atlanta Hawks    22.0        C  ...     7-3  260.0
314     Kent Bazemore       Atlanta Hawks    24.0       SF  ...     6-5  201.0
315  Tim Hardaway Jr.       Atlanta Hawks    10.0       SG  ...     6-6  205.0
316      Kirk Hinrich       Atlanta Hawks    12.0       SG  ...     6-4  190.0
317        Al Horford       Atlanta Hawks    15.0        C  ...    6-10  245.0
..                ...                 ...     ...      ...  ...     ...    ...
376      Jarell Eddie  Washington Wizards     8.0       SG  ...     6-7  218.0
375      Jared Dudley  Washington Wizards     1.0       SF  ...     6-7  225.0
```

Parte 1.2 – *Filtragem de dados usando condições*

Para extrair dados com base em uma condição. Como tal, deve primeiro **invocar o dataframe** a trabalhar e depois **referir o índice da coluna a filtrar a informação** e **colocar a condição**:

```
#Para extrair dados com base numa condição
data_altura = data[data["Weight"] > 195]
print(data_altura)
```

	Name	Team	Number	Position	Age	Height	Weight
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0
2	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0
3	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0
5	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0
6	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0
..	...	...	...	...	...	...	...
456	Chris Johnson	Utah Jazz	23.0	SF	26.0	6-6	206.0
457	Trey Lyles	Utah Jazz	41.0	PF	20.0	6-10	234.0
458	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0
460	Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0

Parte 1.3 – *Isolar colunas (individualmente ou mais do que uma)*

Podemos apenas isolar uma única coluna usando os símbolos das chavetas retas [ ] e indicar o nome de uma coluna que quer exibir.

O resultado é um objeto pandas Series. Uma série pandas é uma matriz unidimensional que contém dados de qualquer tipo, incluindo inteiros, flutuantes, strings, booleanos, objetos python, etc.

Um DataFrame é composto de muitas séries que funcionam como colunas.

Exemplo: exibir apenas a coluna **Glucose**

```
#Exibir apenas uma coluna
coluna = data["Weight"]
print(coluna)
```

```
0      180.0
1      235.0
2      235.0
3      205.0
4      185.0
       ...
458    203.0
459    179.0
```

Exemplo: Para mostrar mais do que uma coluna, deve colocar dupla chaveta reta e separar os campos por com o símbolo, (vírgula)

```
#Exibir mais que uma coluna
coluna = data[["Weight", "Age"]]
print(coluna)
```

```
     Weight   Age
0     180.0  25.0
1     235.0  25.0
2     235.0  25.0
3     205.0  27.0
4     185.0  22.0
..      ...   ...
458   203.0  26.0
459   179.0  24.0
```

Parte 1.4 – *Função .loc[] e .iloc[]*

Para listar um subconjunto de linhas ou colunas de um Dataframe, podemos usar a função **iloc().**

Por exemplo, o comando **.iloc[0:4, 0:3]** vai listar apenas as **primeiras 4 linhas** e as **primeiras 3 colunas** do Dataframe que estiver a trabalhar.

```
#Função iloc (seleção de colunas e linhas)
data_loc = data.iloc[0:4, 0:3]
print(data_loc)
```

```
             Name            Team  Number
0  Avery Bradley  Boston Celtics     0.0
1     Jae Crowder  Boston Celtics    99.0
2     Jae Crowder  Boston Celtics    99.0
3    John Holland  Boston Celtics    30.0
```

Se quisermos listar ou filtrar os dados do DataFrame através dos rótulos das linhas e/ou colunas, podemos usar a função .loc[ ]

Esta função aceita um ou dois argumentos:

- **1º argumento:** refere-se as linhas, indicando o intervalo de colunas a exibir);
- **2º argumento:** refere-se as colunas (indicando os nomes que quer exibir);

```
#Função loc (seleção de colunas e linhas)
data_loc = data.loc[0:3, ["Age","Height","College"]]
print(data_loc)
```

```
   Age Height           College
0  25.0    6-2             Texas
1  25.0    6-6         Marquette
2  25.0    6-6         Marquette
3  27.0    6-5  Boston University
```

Parte 1.5 – Operações aritméticas (mínimo, máximo, média, mediana)

Podemos realizar operações aritméticas com valores numéricos. Como por exemplo, vamos calcular a idade mínima, máxima, média e mediana.

```
#Exemplo para calcular a média, minimo e máximo da coluna idade
print(f"Idade Minima: {data["Age"].min()}")
print(f"Idade Máxima: {data["Age"].max()}")
print(f"Idade Média: {data["Age"].mean()}")
print(f"Idade Mediana: {data["Age"].median()}")
```

```
Idade Minima: 21
Idade Máxima: 81
Idade Média: 33.240885416666664
Idade Mediana: 29.0
```

Parte 1.6 – *Função .value_counts(): Contar e agrupar valores da coluna*

Podemos realizar operações para contar os valores únicos de uma coluna com a função .**value_counts()**, como por exemplo a coluna Idade:

```
#Contar os valores únicos da coluna (Idade - Age)
contagem_idades = data["Age"].value_counts()
print(f"Idades Agrupadas: {contagem_idades}")
```

```
Idades Agrupadas: Age
22    72
21    63
25    48
24    46
23    38
28    35
26    33
27    32
29    29
31    24
```

Parte 1.7 – Isolar apenas uma linha (pelo índice)

Podemos apenas obter um registo, bastando indicar o valor do índice que deseja exibir. No exemplo mais abaixo, utilizamos o método **index** e igualar o valor a procurar 3 (**exemplo:** data.index == 3):

```
#Isolar apenas uma linha de informação (pelo indice do registo)
print(data[data.index == 3])
```

```
           Name           Team  Number Position   Age Height  Weight           College
3  John Holland  Boston Celtics    30.0       SG  27.0    6-5   205.0  Boston University
```

Parte 1.8 – *Função index.isin(): Isolar várias linhas (pelo índice)*

No entanto, também podemos indicar um intervalo de vários índices. Como tal, para além do método index, deve adicionar a função .isin(range(x,y)) onde **x** representa o início do índice a procurar e o **y** é o fim da sequência:

```
#Isolar informação (num intervalo de registos)
print(data[data.index.isin(range(2, 6))])
```

```
            Name           Team  Number Position   Age Heigh
2    Jae Crowder  Boston Celtics    99.0       SF  25.0    6-
3   John Holland  Boston Celtics    30.0       SG  27.0    6-
4    R.J. Hunter  Boston Celtics    28.0       SG  22.0    6-
5  Jonas Jerebko  Boston Celtics     8.0       PF  29.0   6-1
```

Parte 1.9 – *Função .groupby()*

Na classificação da informação, podemos fazer agrupamento da informação, para resumir dados repetidos. Como tal, podemos usar a função .groupby() para agrupar os dados pela coluna Team e a partir desse agrupamento da informação, pedir para criar uma variável estruturada com 3 campos:

- **Campo Salario_Total** → campo para guardar a soma do campo Salary (salários);
- **Campo Contagem_Jogadores** → campo para guardar a contagem do campo Name (nomes);

```python
#Total da soma de salarios por equipa

Total_Salarios = data.groupby("Team").agg(
    Salario_total=('Salary', 'sum'),
    Contagem_Jogadores=('Name', 'count')
)

print(Total_Salarios)
```

```
                Salario_total  Contagem_Jogadores
Team
Atlanta Hawks          72902950.0                  15
Boston Celtics         58541068.0                  15
Brooklyn Nets          52528475.0                  15
Charlotte Hornets      78340920.0                  15
Chicago Bulls          86783378.0                  15
```

Parte 1.10 – *Procura de informação (por valor)*

Imagine que não sabe o índice do registo. Temos de realizar uma pesquisa por alguma informação específica. Se pretendermos encontrar uma informação numa coluna, temos de usar a pesquisa por coluna.

Exemplo:

Queremos obter os dados na coluna **College** com o nome "Marquette". Como tal, vamos a data, indicar que quero selecionar a coluna **College** e **indicar** qual o texto a encontrar:

```python
#Procurar informação na coluna College por um nome especifico
print(data["College"] == "Marquette")
```

Resultado: Como pode existir mais do que uma encomenda com o mesmo cliente, é devolvida uma lista resumida se encontrou informação:

```
0      False
1       True
2       True
3      False
4      False
       ...
458    False
459    False
460    False
```

Mas se desejar obter a informação da linha em causa, deve colocar a pesquisa dentro da instrução data e depois o resto da informação da filtragem:

```python
#Procurar informação na coluna College por um nome especifico
print(data[data["College"] == "Marquette"])
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
156	Jimmy Butler	Chicago Bulls	21.0	SG	26.0	6-7	220.0	Marquette	16407500.0
225	Steve Novak	Milwaukee Bucks	6.0	SF	32.0	6-10	225.0	Marquette	295327.0
237	Wesley Matthews	Dallas Mavericks	23.0	SG	29.0	6-5	220.0	Marquette	16407500.0
354	Dwyane Wade	Miami Heat	3.0	SG	34.0	6-4	220.0	Marquette	20000000.0

Parte 1.11 – *Procura de informação (por valor)*

O exemplo anterior permite devolver os dados, mas, também existe o método **query** que consegue ser mais eficiente na escrita de condições de filtragem.

Como tal, basta indicar o objeto com a data e a função dentro da função query, indicamos qual a coluna a trabalhar e o valor dentro de peliculas (aspa simples).

Veja o seguinte exemplo:

```
#Procurar informação na coluna College com a função query()
print(data.query('College == "Marquette"'))
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
156	Jimmy Butler	Chicago Bulls	21.0	SG	26.0	6-7	220.0	Marquette	16407500.0
225	Steve Novak	Milwaukee Bucks	6.0	SF	32.0	6-10	225.0	Marquette	295327.0
237	Wesley Matthews	Dallas Mavericks	23.0	SG	29.0	6-5	220.0	Marquette	16407500.0
354	Dwyane Wade	Miami Heat	3.0	SG	34.0	6-4	220.0	Marquette	20000000.0

Mas, também pode colocar várias condições:

```
#Procurar informação na coluna College com a função query()
print(data.query('College == "Marquette" and Age < 27'))
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
156	Jimmy Butler	Chicago Bulls	21.0	SG	26.0	6-7	220.0	Marquette	16407500.0