Trabalho de python 475560

September 2, 2021

Nome: Azael Frota Viana Gomes Matrícula: 475560

1 Isto está formatado como código

Link para este COLAB:
https://colab.research.google.com/drive/1S49x2_XDgcqAoRgCF153dxAR0SUPMDww?usp=sharing

[64]:
```
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('pdf', 'svg')
```

Importando as bibliotecas Os imports por convenção são colocados na primeira célula

[ ]:
```
import pandas as pd
```

Importando o data set nba

[ ]:
```
dados_nba = pd.read_csv('https://raw.githubusercontent.com/fivethirtyeight/data/
↪master/nba-elo/nbaallelo.csv')
dados_nba
```

[ ]:
```
        gameorder       game_id lg_id  ...  game_result  forecast notes
0               1  194611010TRH   NBA  ...            L  0.640065   NaN
1               1  194611010TRH   NBA  ...            W  0.359935   NaN
2               2  194611020CHS   NBA  ...            W  0.631101   NaN
3               2  194611020CHS   NBA  ...            L  0.368899   NaN
4               3  194611020DTF   NBA  ...            L  0.640065   NaN
...           ...           ...   ...  ...          ...       ...   ...
126309      63155  201506110CLE   NBA  ...            L  0.546572   NaN
126310      63156  201506140GSW   NBA  ...            W  0.765565   NaN
126311      63156  201506140GSW   NBA  ...            L  0.234435   NaN
126312      63157  201506170CLE   NBA  ...            L  0.481450   NaN
126313      63157  201506170CLE   NBA  ...            W  0.518550   NaN

[126314 rows x 23 columns]
```

Comando head

[ ]:
```
type(dados_nba)
```

[ ]:
```
pandas.core.frame.DataFrame
```

Qual o tamanho de dados há em dados_nba e colunas respectivamente

[ ]:
```
len(dados_nba)
```

[ ]:
```
126314
```

[ ]:
```
dados_nba.shape
```

[ ]:
```
(126314, 23)
```

Para descobrir realmete se os dados batem com a descrição usamós o comando head() para visualizar algumas linha de dados

[ ]:
```
dados_nba.head()
```

[ ]:
```
   gameorder       game_id lg_id  ...  game_result  forecast notes
0          1  194611010TRH   NBA  ...            L  0.640065   NaN
1          1  194611010TRH   NBA  ...            W  0.359935   NaN
2          2  194611020CHS   NBA  ...            W  0.631101   NaN
3          2  194611020CHS   NBA  ...            L  0.368899   NaN
4          3  194611020DTF   NBA  ...            L  0.640065   NaN

[5 rows x 23 columns]
```

A não ser que sua tela seja muito grande esse comando exibirá apenas os 23 primeiros elementos podemós usar o comando a seguir com o segundo argumento como 'none' para exibir todas as colunas

[ ]:
```
pd.set_option("display.max.columns", None)
dados_nba.head()
```

[ ]:
```
   gameorder       game_id lg_id  _iscopy  year_id  date_game  seasongame  \
0          1  194611010TRH   NBA        0     1947  11/1/1946           1
1          1  194611010TRH   NBA        1     1947  11/1/1946           1
2          2  194611020CHS   NBA        0     1947  11/2/1946           1
3          2  194611020CHS   NBA        1     1947  11/2/1946           2
4          3  194611020DTF   NBA        0     1947  11/2/1946           1

   is_playoffs team_id  fran_id  pts      elo_i      elo_n  win_equiv opp_id  \
0            0     TRH  Huskies   66  1300.0000  1293.2767  40.294830    NYK
1            0     NYK   Knicks   68  1300.0000  1306.7233  41.705170    TRH
2            0     CHS    Stags   63  1300.0000  1309.6521  42.012257    NYK
3            0     NYK   Knicks   47  1306.7233  1297.0712  40.692783    CHS
4            0     DTF  Falcons   33  1300.0000  1279.6189  38.864048    WSC

   opp_fran  opp_pts  opp_elo_i  opp_elo_n game_location game_result  \
0   Knicks       68  1300.0000  1306.7233             H           L
1  Huskies       66  1300.0000  1293.2767             A           W
2   Knicks       47  1306.7233  1297.0712             H           W
3    Stags       63  1300.0000  1309.6521             A           L
4  Capitols      50  1300.0000  1320.3811             H           L

   forecast notes
0  0.640065   NaN
```

```
1  0.359935   NaN
2  0.631101   NaN
3  0.368899   NaN
4  0.640065   NaN
```

Mas como estamós um super computador ele normalmente tem um grande poder de processamento.

Para exibir as últimas linha ao contrario do cabeçalho usasse o comando tail()

[ ]:
```
dados_nba.tail()
```

[ ]:
```
        gameorder       game_id lg_id  _iscopy  year_id  date_game  \
126309      63155  201506110CLE   NBA        0     2015  6/11/2015
126310      63156  201506140GSW   NBA        0     2015  6/14/2015
126311      63156  201506140GSW   NBA        1     2015  6/14/2015
126312      63157  201506170CLE   NBA        0     2015  6/16/2015
126313      63157  201506170CLE   NBA        1     2015  6/16/2015

        seasongame  is_playoffs team_id    fran_id  pts      elo_i      elo_n  \
126309         100            1     CLE  Cavaliers   82  1723.4149  1704.3949
126310         102            1     GSW   Warriors  104  1809.9791  1813.6349
126311         101            1     CLE  Cavaliers   91  1704.3949  1700.7391
126312         102            1     CLE  Cavaliers   97  1700.7391  1692.0859
126313         103            1     GSW   Warriors  105  1813.6349  1822.2881

        win_equiv opp_id   opp_fran  opp_pts  opp_elo_i  opp_elo_n  \
126309  60.309792    GSW   Warriors      103  1790.9591  1809.9791
126310  68.013329    CLE  Cavaliers       91  1704.3949  1700.7391
126311  60.010067    GSW   Warriors      104  1809.9791  1813.6349
126312  59.290245    GSW   Warriors      105  1813.6349  1822.2881
126313  68.519516    CLE  Cavaliers       97  1700.7391  1692.0859

       game_location game_result  forecast notes
126309             H           L  0.546572   NaN
126310             H           W  0.765565   NaN
126311             A           L  0.234435   NaN
126312             H           L  0.481450   NaN
126313             A           W  0.518550   NaN
```

Com a biblioteca pandas você pode identifica os diferentes tipos de dados neles contidos, as estrutura de dados do pandas são muito mais rápidos do que a tecnologia de lista do python

[ ]:
```
dados_nba.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126314 entries, 0 to 126313
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   gameorder      126314 non-null  int64
```

```
 1   game_id         126314 non-null  object
 2   lg_id           126314 non-null  object
 3   _iscopy         126314 non-null  int64
 4   year_id         126314 non-null  int64
 5   date_game       126314 non-null  object
 6   seasongame      126314 non-null  int64
 7   is_playoffs     126314 non-null  int64
 8   team_id         126314 non-null  object
 9   fran_id         126314 non-null  object
 10  pts             126314 non-null  int64
 11  elo_i           126314 non-null  float64
 12  elo_n           126314 non-null  float64
 13  win_equiv       126314 non-null  float64
 14  opp_id          126314 non-null  object
 15  opp_fran        126314 non-null  object
 16  opp_pts         126314 non-null  int64
 17  opp_elo_i       126314 non-null  float64
 18  opp_elo_n       126314 non-null  float64
 19  game_location   126314 non-null  object
 20  game_result     126314 non-null  object
 21  forecast        126314 non-null  float64
 22  notes           5424 non-null    object
dtypes: float64(6), int64(7), object(10)
memory usage: 22.2+ MB
```

Você pode ver uma lista de todas as colunas em seu conjunto de dados e o tipo de dados que cada coluna contém. Aqui, você pode ver os tipos de dados int64, float64 e objeto. O Pandas usa a biblioteca NumPy para trabalhar com esses tipos.

O próximo comando visa da uma descrição dos dados como média, desvio padrão, minimo, máximo e etc.

[ ]:
```
dados_nba.describe()
```

[ ]:
```
           gameorder        _iscopy        year_id     seasongame  \
count  126314.000000  126314.000000  126314.000000  126314.000000
mean    31579.000000       0.500000    1988.200374      43.533733
std     18231.927643       0.500002      17.582309      25.375178
min         1.000000       0.000000    1947.000000       1.000000
25%     15790.000000       0.000000    1975.000000      22.000000
50%     31579.000000       0.500000    1990.000000      43.000000
75%     47368.000000       1.000000    2003.000000      65.000000
max     63157.000000       1.000000    2015.000000     108.000000

         is_playoffs            pts          elo_i          elo_n  \
count  126314.000000  126314.000000  126314.000000  126314.000000
mean        0.063857     102.729982    1495.236055    1495.236055
std         0.244499      14.814845     112.139945     112.461687
min         0.000000       0.000000    1091.644500    1085.774400
25%         0.000000      93.000000    1417.237975    1416.994900
```

```
50%          0.000000     103.000000    1500.945550    1500.954400
75%          0.000000     112.000000    1576.060000    1576.291625
max          1.000000     186.000000    1853.104500    1853.104500

           win_equiv        opp_pts      opp_elo_i      opp_elo_n  \
count  126314.000000  126314.000000  126314.000000  126314.000000
mean       41.707889     102.729982    1495.236055    1495.236055
std        10.627332      14.814845     112.139945     112.461687
min        10.152501       0.000000    1091.644500    1085.774400
25%        34.103035      93.000000    1417.237975    1416.994900
50%        42.113357     103.000000    1500.945550    1500.954400
75%        49.635328     112.000000    1576.060000    1576.291625
max        71.112038     186.000000    1853.104500    1853.104500

            forecast
count  126314.000000
mean        0.500000
std         0.215252
min         0.020447
25%         0.327989
50%         0.500000
75%         0.672011
max         0.979553
```

O comando .describe apenas analiza as colunas númericas por padrão, mas podemós incluir outro tipos de dados com o comando include. Observe que .describe () não tentará calcular uma média ou um desvio padrão para as colunas que são objetos python, Já que não são dados numéricos e sim texto. No entanto, ele ainda exibirá algumas estatísticas descritivas

[ ]:
```
import numpy as np
dados_nba.describe(include=object)
```

[ ]:
```
             game_id   lg_id  date_game team_id fran_id  opp_id opp_fran  \
count         126314  126314     126314  126314  126314  126314   126314
unique         63157       2      12426     104      53     104       53
top     200712080DEN     NBA  4/13/2011     BOS  Lakers     BOS   Lakers
freq               2  118016         30    5997    6024    5997     6024

       game_location game_result           notes
count         126314      126314            5424
unique             3           2             231
top                A           W  at New York NY
freq           63138       63157             440
```

Com o seguinte comando você pode analisar o número que deterinado elementos se repeteM

[ ]:
```
dados_nba["team_id"].value_counts()
```

[ ]:
```
BOS    5997
NYK    5769
```

```
LAL    5078
DET    4985
PHI    4533
       ...
TRH      60
INJ      60
PIT      60
DTF      60
SDS      11
Name: team_id, Length: 104, dtype: int64
```

[ ]:
```
dados_nba["fran_id"].value_counts()
```

[ ]:
```
Lakers          6024
Celtics         5997
Knicks          5769
Warriors        5657
Pistons         5650
Sixers          5644
Hawks           5572
Kings           5475
Wizards         4582
Spurs           4309
Bulls           4307
Pacers          4227
Thunder         4178
Rockets         4154
Nuggets         4120
Nets            4106
Suns            4080
Bucks           4034
Trailblazers    3870
Cavaliers       3810
Clippers        3733
Jazz            3555
Mavericks       3013
Heat            2371
Pelicans        2254
Magic           2207
Timberwolves    2131
Grizzlies       1657
Raptors         1634
Hornets          894
Colonels         846
Squires          799
Spirits          777
Stars            756
Sounds           697
```

```
Baltimore        467
Floridians       440
Condors          430
Capitols         291
Olympians        282
Sails            274
Stags            260
Bombers          249
Steamrollers     168
Packers           72
Redskins          65
Rebels            63
Denver            62
Waterloo          62
Ironmen           60
Falcons           60
Huskies           60
Jets              60
Name: fran_id, dtype: int64
```

Podemós combinar comandos agora para saber onde o lakers jogou fora do Los angeles lakers e descobrimós que corresponde a "MNL" Minneapolis Lakers.

[ ]:
```
dados_nba.loc[dados_nba["fran_id"] == "Lakers", "team_id"].value_counts()
```

[ ]:
```
LAL    5078
MNL     946
Name: team_id, dtype: int64
```

Podemós até ver qual o jogo mais recente e mais antigo jogado em MNL

[ ]:
```
dados_nba["date_played"] = pd.to_datetime(dados_nba["date_game"])
dados_nba.loc[dados_nba["team_id"] == "MNL", "date_played"].agg(("min", "max"))
```

[ ]:
```
min   1948-11-04
max   1960-03-26
Name: date_played, dtype: datetime64[ns]
```

2 Entendendo a serie de Objetos

A estrutura de dados mais básica do Python é a lista, que também é um bom ponto de partida para conhecer os objetos pandas.Series. Crie um novo objeto Series com base em uma lista:

Um exemplo do tipo de dados Series é apresentado em seguida, adaptando parte do código para usar a estrutura dados de serie.

[ ]:
```
revenues = pd.Series([5555, 7000, 1980])
revenues
type(revenues.values)
```

[ ]:
```
numpy.ndarray
```

[ ]:
```
city_revenues = pd.Series(
    [4200, 8000, 6500],
    index=["Amsterdam", "Toronto", "Tokyo"]
    )
city_revenues
```

[ ]:
```
Amsterdam    4200
Toronto      8000
Tokyo        6500
dtype: int64
```

[ ]:
```
city_employee_count = pd.Series({"Amsterdam": 5, "Tokyo": 8})
city_employee_count
```

[ ]:
```
Amsterdam    5
Tokyo        8
dtype: int64
```

3 Entendendo os objetos DataFrame

Embora uma série seja uma estrutura de dados muito poderosa, ela tem suas limitações. Por exemplo, você só pode armazenar um atributo por chave. Como você viu com o conjunto de dados nba, que possui 23 colunas, a biblioteca Pandas Python tem mais a oferecer com seu DataFrame. Esta estrutura de dados é uma sequência de objetos Series que compartilham o mesmo índice.

Se você acompanhou os exemplos da Série, já deve ter dois objetos da Série com cidades como chaves:

```
city_revenues
city_employee_count
```

Você pode combinar esses objetos em um DataFrame, fornecendo um dicionário no construtor. As chaves de dicionário se tornarão os nomes das colunas, e os valores devem conter os objetos Series:

[ ]:
```
city_data = pd.DataFrame({
    "revenue": city_revenues,
    "employee_count": city_employee_count
})
city_data
```

[ ]:
```
           revenue  employee_count
Amsterdam     4200             5.0
Tokyo         6500             8.0
Toronto       8000             NaN
```

[ ]:
```
city_data.index
```

[ ]:
```
Index(['Amsterdam', 'Tokyo', 'Toronto'], dtype='object')
```

[ ]:
```
city_data.values
```

[ ]:
```
array([[4.2e+03, 5.0e+00],
       [6.5e+03, 8.0e+00],
       [8.0e+03,     nan]])
```

[ ]:
```
city_data.axes
city_data.axes[0]
city_data.axes[1]
city_data.keys()
```

[ ]:
```
Index(['revenue', 'employee_count'], dtype='object')
```

[ ]:

Agora, vamós adiciona o o dataset da nba que ele selecionará linhas com base nos valores das colunas do conjunto de dados para consultar seus dados. Por exemplo, vamos criar um novo DataFrame que contenha apenas jogos disputados após 2010:

[ ]:
```
current_decade = dados_nba[dados_nba["year_id"] > 2010]
current_decade.shape
```

[ ]:
```
(12658, 24)
```

Agora vamos seleciona as linhas que o apresentam valor não nulo:

[ ]:
```
games_with_notes = dados_nba[dados_nba["notes"].notnull()]
games_with_notes.shape
```

[ ]:
```
(5424, 24)
```

Você também pode usar .notna () para realizar o mesmo objetivo. Você pode até acessar valores do objeto tipo de dados como uma str e executar métodos de string neles:

[ ]:
```
ers = dados_nba[dados_nba["fran_id"].str.endswith("ers")]
ers.shape
```

[ ]:
```
(27797, 24)
```

usando o comamdo .str.endswith() para filtrar todos os jogos que os time de casa o nome terminas com "ers".

Fazaaer um pesquisa quais os jogos de Baltimore os dois times pontuaram mais de 100 pontos e depois vamos excluir as duplicatas:

[ ]:
```
dados_nba[
          (dados_nba["_iscopy"] == 0) &
          (dados_nba["pts"] > 0) &
          (dados_nba["opp_pts"] > 100) &
          (dados_nba["team_id"] == "BLB")
]
```

[ ]:
```
      gameorder       game_id lg_id  _iscopy  year_id   date_game  seasongame  \
1495        748  194901130BLB   NBA        0     1949   1/13/1949          33
1683        842  194902170BLB   NBA        0     1949   2/17/1949          50
1726        864  194902260BLB   NBA        0     1949   2/26/1949          53
4058       2030  195112210BLB   NBA        0     1952  12/21/1951          24
4296       2149  195202080BLB   NBA        0     1952    2/8/1952          46
4665       2333  195211270BLB   NBA        0     1953  11/27/1952          13
```

```
4715       2358  195212060BLB   NBA        0     1953   12/6/1952          18
4780       2391  195212200BLB   NBA        0     1953  12/20/1952          22
4890       2446  195301100BLB   NBA        0     1953   1/10/1953          32
4909       2455  195301140BLB   NBA        0     1953   1/14/1953          34
5026       2514  195302070BLB   NBA        0     1953    2/7/1953          46
5059       2530  195302130BLB   NBA        0     1953   2/13/1953          51
5064       2533  195302140BLB   NBA        0     1953   2/14/1953          52
5208       2605  195303110BLB   NBA        0     1953   3/11/1953          66
5382       2692  195311210BLB   NBA        0     1954  11/21/1953          10
5455       2728  195312060BLB   NBA        0     1954   12/6/1953          17
5684       2843  195401280BLB   NBA        0     1954   1/28/1954          45
5825       2913  195402220BLB   NBA        0     1954   2/22/1954          60
5839       2920  195402250BLB   NBA        0     1954   2/25/1954          62
5873       2937  195403030BLB   NBA        0     1954    3/3/1954          64

      is_playoffs team_id    fran_id  pts      elo_i      elo_n  win_equiv  \
1495            0     BLB  Baltimore   85  1513.9349  1496.6301  46.617260
1683            0     BLB  Baltimore  100  1452.0424  1441.4016  40.859585
1726            0     BLB  Baltimore  114  1421.9351  1419.4280  38.557545
4058            0     BLB  Baltimore   89  1420.4839  1407.0557  30.434895
4296            0     BLB  Baltimore   83  1321.0188  1310.5110  21.712118
4665            0     BLB  Baltimore   92  1363.8076  1356.6135  25.794765
4715            0     BLB  Baltimore   91  1347.6221  1336.2109  23.985638
4780            0     BLB  Baltimore   99  1323.0144  1320.2793  22.626024
4890            0     BLB  Baltimore  126  1328.6687  1356.6470  25.797792
4909            0     BLB  Baltimore  104  1349.8323  1346.3594  24.876236
5026            0     BLB  Baltimore   98  1331.5853  1327.8528  23.266380
5059            0     BLB  Baltimore   96  1311.2952  1308.0129  21.612337
5064            0     BLB  Baltimore   88  1308.0129  1298.6824  20.861076
5208            0     BLB  Baltimore  107  1284.5239  1282.2396  19.579676
5382            0     BLB  Baltimore   83  1341.8109  1323.2826  22.471680
5455            0     BLB  Baltimore   95  1319.6473  1314.3059  21.730190
5684            0     BLB  Baltimore   82  1333.0491  1324.7513  22.594488
5825            0     BLB  Baltimore  110  1303.7493  1301.9685  20.736986
5839            0     BLB  Baltimore   86  1295.8394  1289.8191  19.788719
5873            0     BLB  Baltimore  100  1285.8671  1279.9624  19.041433

     opp_id      opp_fran  opp_pts  opp_elo_i  opp_elo_n game_location  \
1495    ROC         Kings      102  1566.2607  1583.5656             H
1683    PRO  Steamrollers      102  1259.5144  1270.1552             H
1726    MNL        Lakers      115  1637.9852  1640.4923             H
4058    BOS       Celtics      106  1543.5579  1556.9861             H
4296    NYK        Knicks      103  1529.6166  1540.1244             H
4665    NYK        Knicks      104  1580.5809  1587.7750             H
4715    FTW       Pistons      103  1452.9038  1464.3149             H
4780    NYK        Knicks      102  1589.5110  1592.2461             H
4890    BOS       Celtics      105  1591.1434  1563.1652             H
```

```
4909    MNL        Lakers      112  1665.4396  1668.9125            H
5026    BOS       Celtics      101  1531.2122  1534.9447            H
5059    NYK        Knicks      106  1663.1764  1666.4586            H
5064    FTW       Pistons      102  1488.5692  1497.8998            H
5208    NYK        Knicks      113  1649.1516  1651.4359            H
5382    PHW      Warriors      103  1407.0597  1425.5880            H
5455    BOS       Celtics      102  1530.5361  1535.8776            H
5684    MNL        Lakers      104  1606.0645  1614.3622            H
5825    BOS       Celtics      111  1591.4943  1593.2749            H
5839    ROC         Kings      102  1589.2533  1595.2736            H
5873    MLH         Hawks      103  1370.7719  1376.6765            H

     game_result  forecast               notes date_played
1495           L  0.568181                 NaN  1949-01-13
1683           L  0.843424                 NaN  1949-02-17
1726           L  0.338936                 NaN  1949-02-26
4058           L  0.466843   at Providence RI  1951-12-21
4296           L  0.348614  at Philadelphia PA  1952-02-08
4665           L  0.338004  at Philadelphia PA  1952-11-27
4715           L  0.492400                 NaN  1952-12-06
4780           L  0.277192                 NaN  1952-12-20
4890           W  0.281855                 NaN  1953-01-10
4909           L  0.224238                 NaN  1953-01-14
5026           L  0.360430                 NaN  1953-02-07
5059           L  0.190010       at Boston MA  1953-02-13
5064           L  0.386104                 NaN  1953-02-14
5208           L  0.178973       at Boston MA  1953-03-11
5382           L  0.549845      at Raleigh NC  1953-11-21
5455           L  0.345625                 NaN  1953-12-06
5684           L  0.269737  at Grand Forks ND  1954-01-28
5825           L  0.253365    at Worcester MA  1954-02-22
5839           L  0.247241  at Philadelphia PA  1954-02-25
5873           L  0.521710                 NaN  1954-03-03
```

Aqui, você usa dados_nba ["_ iscopy"] == 0 para incluir apenas as entradas que não são cópias.

4 Agrupando e agregando seus dados

A biblioteca Pandas Python oferece funções de agrupamento e agregação para ajudar a realizar essa tarefa.

[ ]:
```
city_revenues.sum()
```

[ ]:
```
18700
```

[ ]:
```
city_revenues.max()
```

[ ]:
```
8000
```

Lembre-se de que uma coluna de um DataFrame é, na verdade, um objeto Series. Por esse

motivo, podemós usar essas mesmas funções nas colunas do dados_nba:

[ ]:
```
points = dados_nba["pts"]
type(points)
points.sum()
```

[ ]:
```
12976235
```

[ ]:
```
dados_nba.groupby("fran_id", sort=False)["pts"].sum()
```

[ ]:
```
fran_id
Huskies           3995
Knicks          582497
Stags            20398
Falcons           3797
Capitols         22387
Celtics         626484
Steamrollers     12372
Ironmen           3674
Bombers          17793
Rebels            4474
Warriors        591224
Baltimore        37219
Jets              4482
Pistons         572758
Lakers          637444
Kings           569245
Hawks           567261
Denver            4818
Olympians        22864
Redskins          5372
Waterloo          4921
Packers           6193
Sixers          585891
Wizards         474809
Bulls           437269
Thunder         437735
Squires          91127
Stars            84940
Rockets         432504
Colonels         94435
Pacers          438288
Nuggets         445780
Spurs           453822
Spirits          85874
Sounds           75582
Floridians       49568
Nets            417809
Condors          49642
```

```
Bucks           418326
Suns            437486
Clippers        380523
Cavaliers       380416
Trailblazers    402695
Sails            30080
Jazz            363155
Mavericks       309239
Pelicans        220794
Heat            229103
Timberwolves    207693
Magic           219436
Grizzlies       157683
Raptors         158370
Hornets          84489
Name: pts, dtype: int64
```

[ ]:
```
dados_nba[
    (dados_nba["fran_id"] == "Spurs") &
    (dados_nba["year_id"] > 2010)
    ].groupby(["year_id", "game_result"])["game_id"].count()
```

[ ]:
```
year_id  game_result
2011     L              25
         W              63
2012     L              20
         W              60
2013     L              30
         W              73
2014     L              27
         W              78
2015     L              31
         W              58
Name: game_id, dtype: int64
```

5 Manipulando Colunas

Você precisará saber como manipular as colunas do seu conjunto de dados em diferentes fases do processo de análise de dados. Você pode adicionar e descartar colunas como parte da fase inicial de limpeza de dados ou posteriormente com base nos insights de sua análise.

[ ]:
```
df = dados_nba.copy()
df.shape
```

[ ]:
```
(126314, 24)
```

Você pode definir novas colunas baseado nas já existentes

[ ]:
```
df["difference"] = df.pts - df.opp_pts
df.shape
```

[ ]:
```
(126314, 25)
```

Criando uma nova coluna para armenezar a difereça de pontos entre colna "pts" e a coluna "opp_pts"

[ ]:
```
df["difference"].max()
```

[ ]:
```
68
```

Podemós os seguintes comandos para renomear as colunas

[ ]:
```
renamed_df = df.rename(
    columns={"gameorder": "ordemdejogo", "game_id": "id_jogo", "game_result":␣
 ↪"resultado", "game_location": "localizacao"}
)
renamed_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126314 entries, 0 to 126313
Data columns (total 25 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   ordemdejogo  126314 non-null  int64
 1   id_jogo      126314 non-null  object
 2   lg_id        126314 non-null  object
 3   _iscopy      126314 non-null  int64
 4   year_id      126314 non-null  int64
 5   date_game    126314 non-null  object
 6   seasongame   126314 non-null  int64
 7   is_playoffs  126314 non-null  int64
 8   team_id      126314 non-null  object
 9   fran_id      126314 non-null  object
 10  pts          126314 non-null  int64
 11  elo_i        126314 non-null  float64
 12  elo_n        126314 non-null  float64
 13  win_equiv    126314 non-null  float64
 14  opp_id       126314 non-null  object
 15  opp_fran     126314 non-null  object
 16  opp_pts      126314 non-null  int64
 17  opp_elo_i    126314 non-null  float64
 18  opp_elo_n    126314 non-null  float64
 19  localizacao  126314 non-null  object
 20  resultado    126314 non-null  object
 21  forecast     126314 non-null  float64
 22  notes        5424 non-null    object
 23  date_played  126314 non-null  datetime64[ns]
 24  difference   126314 non-null  int64
dtypes: datetime64[ns](1), float64(6), int64(8), object(10)
memory usage: 24.1+ MB
```

Você pode excluir as colunas com o seguinte comando

[ ]:
```
df.shape
```

[ ]:
```
(126314, 25)
```

[ ]:
```
df.shape
elo_columns = ["elo_i", "elo_n", "opp_elo_i", "opp_elo_n"]
df.drop(elo_columns, inplace=True, axis=1)
df.shape
```

[ ]:
```
(126314, 21)
```

6 Especificando o tipo de dados

Com os seguintes comandos você pode trabalhar os tipos de dados e até alterar o tipo de dados:

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126314 entries, 0 to 126313
Data columns (total 21 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   gameorder      126314 non-null  int64
 1   game_id        126314 non-null  object
 2   lg_id          126314 non-null  object
 3   _iscopy        126314 non-null  int64
 4   year_id        126314 non-null  int64
 5   date_game      126314 non-null  object
 6   seasongame     126314 non-null  int64
 7   is_playoffs    126314 non-null  int64
 8   team_id        126314 non-null  object
 9   fran_id        126314 non-null  object
 10  pts            126314 non-null  int64
 11  win_equiv      126314 non-null  float64
 12  opp_id         126314 non-null  object
 13  opp_fran       126314 non-null  object
 14  opp_pts        126314 non-null  int64
 15  game_location  126314 non-null  object
 16  game_result    126314 non-null  object
 17  forecast       126314 non-null  float64
 18  notes          5424 non-null    object
 19  date_played    126314 non-null  datetime64[ns]
 20  difference     126314 non-null  int64
dtypes: datetime64[ns](1), float64(2), int64(8), object(10)
memory usage: 20.2+ MB
```

Convertendo a coluna data_game para datetime

[ ]:
```
df["date_game"] = pd.to_datetime(df["date_game"])
```

[ ]:
```
df["game_location"].nunique()
```

[ ]:
```
3
```

A coluna game_location só pode ter 3 tipos diferentes de valor

[ ]:
```
df["game_location"].value_counts()
```

[ ]:
```
A    63138
H    63138
N       38
Name: game_location, dtype: int64
```

[ ]:
```
df["game_location"] = pd.Categorical(df["game_location"])
df["game_location"].dtype
```

[ ]:
```
CategoricalDtype(categories=['A', 'H', 'N'], ordered=False)
```

Categorizando os dados com o tipo de dado "categorical" tem vantagens sobre o texto não estruturado, como na alocação de memória.

7 Combinando múltiplos datasets

Anteriormente, você combinou dois objetos Series em um DataFrame com base em seus índices. Agora, você vai dar um passo adiante e usar .concat () para combinar city_data com outro DataFrame. Digamos que você tenha conseguido coletar alguns dados sobre mais duas cidades: Anteriormente, você combinou dois objetos Series em um DataFrame com base em seus índices. Agora, você vai dar um passo adiante e usar .concat () para combinar city_data com outro DataFrame. Digamos que você tenha conseguido coletar alguns dados sobre mais duas cidades:

[ ]:
```
further_city_data = pd.DataFrame(
    {"revenue": [7000, 3400], "employee_count":[2, 2]},
    index=["New York", "Barcelona"]
)
```

O segundo data frame contém a cidade de barcelona e nova york

[ ]:
```
all_city_data = pd.concat([city_data, further_city_data], sort=False)
all_city_data
```

[ ]:

	revenue	employee_count
Amsterdam	4200	5.0
Tokyo	6500	8.0
Toronto	8000	NaN
New York	7000	2.0
Barcelona	3400	2.0

Por padrão, concat () combina ao longo do eixo = 0. Em outras palavras, ele anexa linhas. Você também pode usá-lo para anexar colunas, fornecendo o parâmetro eixo = 1:

[ ]:
```
city_countries = pd.DataFrame({
    "country": ["Holland", "Japan", "Holland", "Canada", "Spain"],
    "capital": [1, 1, 0, 0, 0]},
    index=["Amsterdam", "Tokyo", "Rotterdam", "Toronto", "Barcelona"]
```