Trabalho de Python

Aluno: Israel da Silva Pereira

Matricula: 497145

Link para este COLAB: https://colab.research.google.com/drive/1FARnYxvaJMZc6hcXTaoJoV0dZ6Du5_n9?usp=sharing

---

PARTE 1

Importando a biblioteca pandas como 'pd':

```
import pandas as pd
```

Download do arquivo nba_all_elo.csv:

```
import requests

download_url = "https://raw.githubusercontent.com/fivethirtyeight/data/master/nba-elo/nbaa
target_csv_path = "nba_all_elo.csv"

response = requests.get(download_url)
response.raise_for_status()    # Check that the request was successful
with open(target_csv_path, "wb") as f:
    f.write(response.content)
print("Download ready.")
```

```
Download ready.
```

Lendo e salvando o arquivo como DataFrame:

```
nba = pd.read_csv("nba_all_elo.csv")
type(nba)
```

```
pandas.core.frame.DataFrame
```

Vendo a quantidades de linhas do arquivo:

```
len(nba)
```

```
126314
```

Vendo a quantidade de linhas e colunas dos arquivos:

```
nba.shape
```

```
(126314, 23)
```

Exibindo as 5 primeiras linhas:

```
nba.head()
```

	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is_playc
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	

Configurando os valores para 2 casas decimais e exibindo as 5 ultimas linhas:

```
pd.set_option("display.precision", 2)
nba.tail()
```

	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is
126309	63155	201506110CLE	NBA	0	2015	6/11/2015	100	
126310	63156	201506140GSW	NBA	0	2015	6/14/2015	102	
126311	63156	201506140GSW	NBA	1	2015	6/14/2015	101	
126312	63157	201506170CLE	NBA	0	2015	6/16/2015	102	
126313	63157	201506170CLE	NBA	1	2015	6/16/2015	103	

Exibindo a quantidade de linhas não nulas que cada coluna possui, podemos observar que 'notes' é a unica coluna que possui valores nulos:

```
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
```

Exibir de todas as colunas numericas, os valores:

Quantidade total de valores, média, desvio padrão, valor mínimo, 25% da coluna, 50% da coluna (média), 75% da coluna e valor máximo.

```
nba.describe()
```

	gameorder	_iscopy	year_id	seasongame	is_playoffs	pts	elo_i
count	126314.00	126314.0	126314.00	126314.00	126314.00	126314.00	126314.00
mean	31579.00	0.5	1988.20	43.53	0.06	102.73	1495.24
std	18231.93	0.5	17.58	25.38	0.24	14.81	112.14
min	1.00	0.0	1947.00	1.00	0.00	0.00	1091.64
25%	15790.00	0.0	1975.00	22.00	0.00	93.00	1417.24
50%	31579.00	0.5	1990.00	43.00	0.00	103.00	1500.95
75%	47368.00	1.0	2003.00	65.00	0.00	112.00	1576.06
max	63157.00	1.0	2015.00	108.00	1.00	186.00	1853.10

Gerando estastísticas descritivas para colunas não numéricas:

Quantidade total de valores, quantidade de valores diferentes, o valor que mais aparece e a

```
import numpy as np
nba.describe(include=object)
```

	game_id	lg_id	date_game	team_id	fran_id	opp_id	opp_fran	game_l
count	126314	126314	126314	126314	126314	126314	126314	
unique	63157	2	12426	104	53	104	53	
top	198204100GSW	NBA	4/17/2013	BOS	Lakers	BOS	Lakers	

A frequencia com que cada valor aparece na coluna especificada:

```
nba["team_id"].value_counts()
```

```
BOS    5997
NYK    5769
LAL    5078
DET    4985
PHI    4533
       ...
PIT      60
TRH      60
DTF      60
INJ      60
SDS      11
Name: team_id, Length: 104, dtype: int64
```

Selecionando as linhas da tabela em que a coluna "fran_id" = "Lakers", e fazendo a frequencia dos valores da coluna "team_id" apenas nas linhas selecionadas:

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

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

Criando uma nova coluna "date_played" igual a coluna "date_game" transformada em formato de data real. Em seguida, selecionando as linhas da tabela em que a coluna "team_id" = "MNL" e depois exibindo a menor data, ou seja, a mais antiga, da coluna "date_played" entre as linhas selecionadas:

```
nba["date_played"] = pd.to_datetime(nba["date_game"])
nba.loc[nba["team_id"] == "MNL", "date_played"].min()
```

```
Timestamp('1948-11-04 00:00:00')
```

Mesmo processo da função anterior, porém dessa vez vai ser apresentado a maior data, ou seja, a mais recente:

```
nba.loc[nba['team_id'] == 'MNL', 'date_played'].max()
```

```
Timestamp('1960-03-26 00:00:00')
```

Ainda seguindo o mesmo processo, exibindo a menor e maior data:

```
nba.loc[nba["team_id"] == "MNL", "date_played"].agg(("min", "max"))
```

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

Salvando uma nova tabela "current_decade" com as linhas da tabela "nba" em que o valor da coluna "year_id" maior que 2010:

```
current_decade = nba[nba["year_id"] > 2010]
current_decade.shape
```

```
(12658, 24)
```

Semelhante ao processo anterior, tendo como referencia dessa vez a coluna "notes" em que o valor é diferente de nulo:

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

```
(5424, 24)
```

Fazendo o processo anterior em colunas não numericas criando uma tabela "ers" com todas as linhas da tabela "nba" em que na coluna "fran_id" os nomes terminam em 'ers':

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

```
(27797, 24)
```

Exibindo a parte da tabela que atende as condições ("_iscopy" igual a 0 e "pts" maior que 100 e "opp_pts" maior que 100 e "team_id" igual a 'BLB'):

```
nba[(nba["_iscopy"] == 0) & (nba["pts"] > 100) & (nba["opp_pts"] > 100) & (nba["team_id"]
```

	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is_pl
1726	864	194902260BLB	NBA	0	1949	2/26/1949	53	
4890	2446	195301100BLB	NBA	0	1953	1/10/1953	32	
4909	2455	195301140BLB	NBA	0	1953	1/14/1953	34	
5208	2605	195303110BLB	NBA	0	1953	3/11/1953	66	
5825	2913	195402220BLB	NBA	0	1954	2/22/1954	60	

Salvando a coluna "pts" como uma serie ("points"):

```
points = nba["pts"]
type(points)
```

```
pandas.core.series.Series
```

Soma da serie, nesse caso seria a soma total de pontos:

```
points.sum()
```

```
12976235
```

Soma total dos pontos por equipe:

```
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
```

Exibindo a quantidade de vitórias e derrotas, em todos os anos depois de 2010, da equipe "Spurs":

```
nba[(nba["fran_id"] == "Spurs") & (nba["year_id"] > 2010)].groupby(["year_id", "game_resul
```

```
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
```

Criando uma copia "df" da tabela "nba":

```
df = nba.copy()
df.shape
```

```
df.shape
```

```
(126314, 24)
```

Adicionando uma nova coluna "difference" em "df" que vai receber o resultado da diferença entre as colunas "pts" e "opp_pts":

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

```
(126314, 25)
```

O maximo valor da coluna "difference", ou seja, a maior diferença de pontos que houve em uma partida:

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

```
68
```

Renomeando colunas:

```
renamed_df = df.rename(columns={"game_result": "result", "game_location": "location"})
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   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  location     126314 non-null  object
 20  result       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
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126314 entries, 0 to 126313
Data columns (total 25 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
 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
```

Excluindo as colunas "elo_i", "elo_n", "opp_elo_i", "opp_elo_n":

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

```
(126314, 21)
```

```
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
```

Transforma a coluna "date_game" em formato de data real:

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

Verifica a quantidade de valores diferente na coluna:

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

```
3
```

Verificando a frequencia que cada valor da coluna aparece:

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

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

Transformando a coluna "game_location" em categoria, levando em conta seus três valores diferentes:

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

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

Criando um nova tabela que resulta de excluir as linhas que possuem algum valor nulo:

```
rows_without_missing_data = nba.dropna()
rows_without_missing_data.shape
```

```
(5424, 24)
```

Nova tabela que resulta de excluir as colunas que possuem algum valor nulo.Nesse caso, como só a coluna "notes" possui valores nulos, somente a mesma foi excluida:

```
data_without_missing_columns = nba.dropna(axis=1)
data_without_missing_columns.shape
```

```
(126314, 23)
```

Preenchendo todas as linhas nulas da coluna "notes":

```
data_with_default_notes = nba.copy()
data_with_default_notes["notes"].fillna( value="no notes at all", inplace=True)
data_with_default_notes["notes"].describe()
```

```
count              126314
unique                232
top       no notes at all
freq               120890
Name: notes, dtype: object
```

Retornando todas as linhas da tabela em que a coluna "pts" é igual a 0:

```
nba[nba["pts"] == 0]
```

	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is_p
26684	13343	197210260VIR	ABA	1	1973	10/26/1972	7	

Verificando a existencia de algum valor inconcistente:

```
nba[(nba["pts"] > nba["opp_pts"]) & (nba["game_result"] != 'W')].empty
```

```
True
```

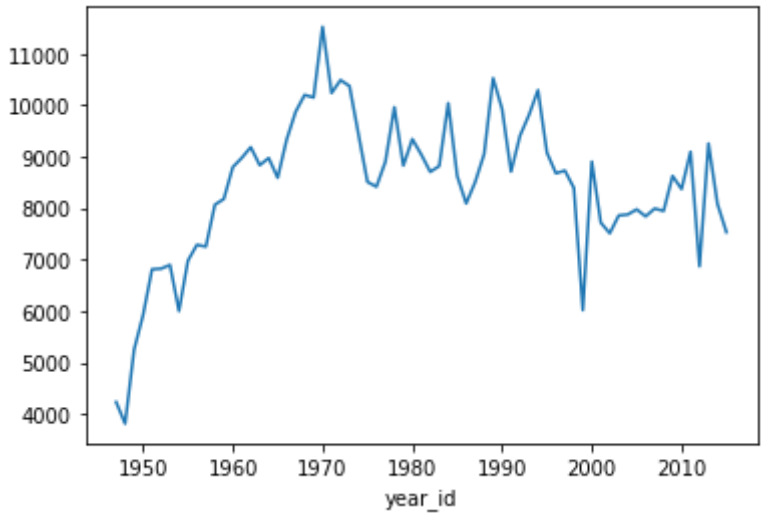
```
nba[(nba["pts"] < nba["opp_pts"]) & (nba["game_result"] != 'L')].empty
```

```
True
```

Gerando um gráfico da soma dos pontos(soma da coluna "pts") em cada ano(coluna "yeard_id"), tendo como referencia as linhas em que a coluna "fran_id" é igual a "Knicks":

```
nba[nba["fran_id"] == "Knicks"].groupby("year_id")["pts"].sum().plot()
```

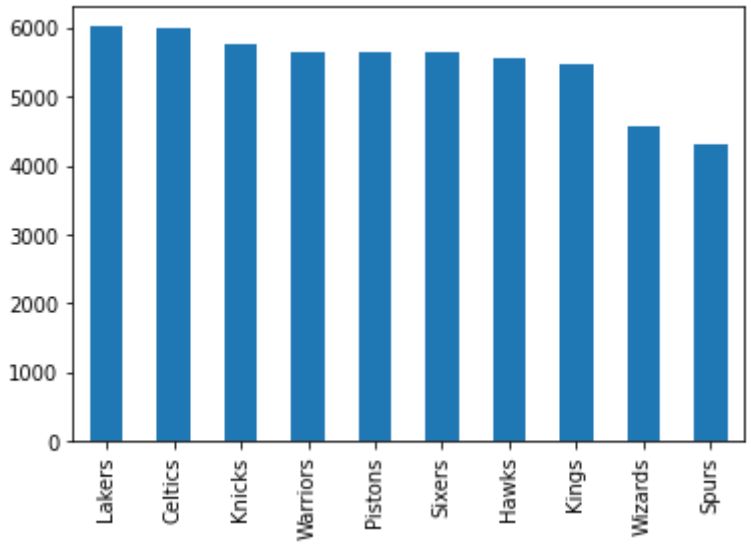
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe36c69c150>
```



Gerando um gráfico de barra com a 10 maiores frequencia da coluna "fran_id". Nesse caso, significa as 10 equipes com mais jogos disputados:

```
nba["fran_id"].value_counts().head(10).plot(kind="bar")
```

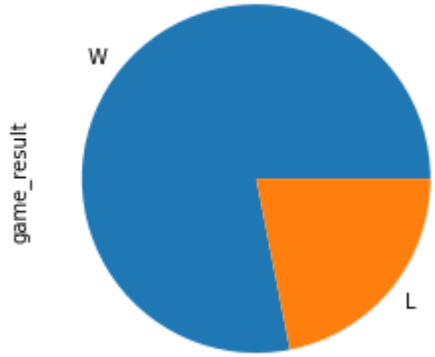
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe36c028f50>
```



Gerando um gráfico de pizza com a frequencia dos valores da coluna "game_result", seguindo as condições ("fran_id" igual a "Heat" e "year_id" igual a '2013'):

```
nba[(nba["fran_id"] == "Heat") & (nba["year_id"] == 2013)]["game_result"].value_counts().p
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe36bfb1c90>
```



PARTE 2

Dataset escolhida: "Cabinet Turnover"

Lendo e salvando o arquivo como DataFrame em uma variavel:

```
import pandas as pd
gabinete = pd.read_csv("cabinet-turnover.csv")
type(gabinete)
```

```
pandas.core.frame.DataFrame
```

Quantidades de linhas e colunas do arquivo:

```
len(gabinete) #linhas
```

```
312
```

```
gabinete.shape #linhas e colunas
```

```
(312, 7)
```

7 primeiras linhas do arquivo:

```
gabinete.head(7)
```

	president	position	appointee	start	end	length	days
0	Carter	OMB Director	Bert Lance	1/21/77	9/23/77	245	247.0
1	Carter	Secretary of Transportation	Brock Adams	1/23/77	7/20/79	908	912.0
2	Carter	Secretary of Health, Education & Welfare	Joseph Califano Jr.	1/25/77	8/3/79	920	926.0

Exibindo as últimas 5 linhas:

```
gabinete.tail()
```

	president	position	appointee	start	end	length	days
307	Trump	Secretary of Homeland Security	Kirstjen Nielsen	12/6/17	Still in office	NaN	NaN
308	Trump	Secretary of Health & Human Services	Alex Azar	1/29/18	Still in office	NaN	NaN
309	Trump	Secretary of State	Mike Pompeo	4/26/18	Still in office	NaN	NaN
310	Trump	CIA Director	Gina Haspel	5/21/18	Still in	NaN	NaN

Exibindo a quantidades de linhas não nulas que cada coluna possui:

```
gabinete.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312 entries, 0 to 311
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   president  312 non-null    object 
 1   position   312 non-null    object 
 2   appointee  312 non-null    object 
 3   start      312 non-null    object 
 4   end        312 non-null    object 
 5   length     294 non-null    object 
 6   days       288 non-null    float64
dtypes: float64(1), object(6)
memory usage: 17.2+ KB
```

Exibir de todas as colunas numericas, os valores:

Quantidade total de valores, média, desvio padrão, valor mínimo, 25% da coluna, 50% da coluna (média), 75% da coluna e valor máximo.

```
gabinete.describe()
```

	days
count	288.00
mean	1861.51
std	832.30
min	190.00
25%	1442.00
50%	1487.00

```
gabinete.describe(include=object)
```

	president	position	appointee	start	end	length
count	312	312	312	312	312	294
unique	7	28	270	238	175	248
top	Bush 43	Chief of Staff	James Baker	1/22/93	1/20/17	2922
freq	58	21	4	10	21	4

A quantidade de nomeações que cada presidente fez:

```
gabinete["president"].value_counts()
```

```
Bush 43    58
Obama      54
Reagan     53
Clinton    50
Bush 41    34
Carter     33
Trump      30
Name: president, dtype: int64
```

Todos os nomeados durante o mandato do presidente Obama, e as respectiva frequencia de nomeação:

```
gabinete.loc[gabinete["president"]=="Obama", "appointee" ].value_counts()
```

```
Jack Lew                3
Shaun Donovan           2
Leon Panetta            2
Sylvia Burwell          2
Ray LaHood              1
Rahm Emanuel            1
Janet Napolitano        1
Bob Gates               1
David Petraeus          1
Samantha Power          1
John King Jr.           1
```

```
Sally Jewell            1
Kathleen Sebelius       1
Gina McCarthy           1
John Bryson             1
Steven Chu              1
Bill Daley              1
Tim Geithner            1
James Clapper           1
Jeh Johnson             1
John Brennan            1
Anthony Foxx            1
Arne Duncan             1
Chuck Hagel             1
Hillary Clinton         1
Joe Biden               1
Ron Kirk                1
Ken Salazar             1
Ash Carter              1
Tom Perez               1
Denis McDonough         1
Maria Contreras-Sweet   1
Lisa Jackson            1
Eric Shinseki           1
Bob McDonald            1
Penny Pritzker          1
Gary Locke              1
John Kerry              1
Loretta Lynch           1
Dennis Blair            1
Michael Froman          1
Eric Holder             1
Hilda Solis             1
Tom Vilsack             1
Susan Rice              1
Karen Mills             1
Peter Orszag            1
Julián Castro           1
Ernest Moniz            1
Name: appointee, dtype: int64
```

Criando novas colunas que receberão as conversões:

~coluna "start" do tipo object para datatime;

~coluna "end" do tipo object para datatime;

~coluna "length" do tipo object para o tipo float;

```
#No processo será desconsiderado as linhas das colunas que possuem caracteres diferente do
#serão preenchidos na nova coluna com valores nulos
gabinete["inicio"] = pd.to_datetime(gabinete["start"],errors = 'coerce')
gabinete["fim"] = pd.to_datetime(gabinete["end"],errors = 'coerce')
gabinete["tmp_mand"] = pd.to_numeric(gabinete["length"],errors = 'coerce')
gabinete.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312 entries, 0 to 311
Data columns (total 10 columns):
```

```
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   president  312 non-null    object
 1   position   312 non-null    object
 2   appointee  312 non-null    object
 3   start      312 non-null    object
 4   end        312 non-null    object
 5   length     294 non-null    object
 6   days       288 non-null    float64
 7   inicio     306 non-null    datetime64[ns]
 8   fim        288 non-null    datetime64[ns]
 9   tmp_mand   282 non-null    float64
dtypes: datetime64[ns](2), float64(2), object(6)
memory usage: 24.5+ KB
```

As datas de inicio da mais antiga e da mais recente nomeação de "Leon Panetta":

```
gabinete.loc[gabinete["appointee"] == "Leon Panetta", "inicio"].agg(("min", "max"))
```

```
min   1993-01-21
max   2011-07-01
Name: inicio, dtype: datetime64[ns]
```

A quantidade de mandato em que a duração foi maior que 2000 dias:

```
mand_2000 = gabinete[gabinete["tmp_mand"] > 2000]
len(mand_2000)
```

```
22
```

Filtrando a parte da tabela em que a coluna "length" é diferente de nulo, ou seja, as linhas da tabela em que a quantidade de dias no mandato já está calculado:

```
mand_with_tmp = gabinete[gabinete["length"].notnull()]
mand_with_tmp
```

	president	position	appointee	start	end	length	days	inicio	fi
0	Carter	OMB Director	Bert Lance	1/21/77	9/23/77	245	247.0	1977-01-21	1977 09-2
1	Carter	Secretary of Transportation	Brock Adams	1/23/77	7/20/79	908	912.0	1977-01-23	1979 07-2

As 5 primeiras linhas da tabela em que a coluna "position" termina em "tor":

		Education &	Califano Jr.					01-25	08-0

```
tor = gabinete[gabinete["position"].str.endswith("tor")]
tor.head()
```

	president	position	appointee	start	end	length	days	inicio	fim
0	Carter	OMB Director	Bert Lance	1/21/77	9/23/77	245	247.0	1977-01-21	1977-09-23
18	Carter	EPA Administrator	Douglas Costle	3/7/77	1/20/81	1415	1462.0	1977-03-07	1981-01-20
19	Carter	OMB Director	Jim McIntyre	9/24/77	1/20/81	1214	1462.0	1977-09-24	1981-01-20
20	Carter	SBA	A. Vernon	4/1/77	1/20/81	1390	1462.0	1977-	1981-

Todos os mandatos em que "James Baker" exerceu por mais de 1000 dias:

```
gabinete[(gabinete["tmp_mand"] > 1000) & (gabinete["appointee"] == "James Baker")]
```

	president	position	appointee	start	end	length	days	inicio	fim	tm
43	Reagan	Chief of Staff	James Baker	1/20/81	2/3/85	1475	1476.0	1981-01-20	1985-02-03	
62	Reagan	Secretary of the Treasury	James Baker	2/3/85	8/17/88	1291	2767.0	1985-02-03	1988-08-17	

A soma total de dias de todos os mandatos dos nomeados:

```
gabinete.groupby("appointee", sort=False)["tmp_mand"].sum()
```

```
appointee
Bert Lance                245.0
Brock Adams               908.0
Joseph Califano Jr.       920.0
Patricia Harris          1458.0
W. Michael Blumenthal     923.0
                          ...
Robert Lighthizer           0.0
Kirstjen Nielsen            0.0
Alex Azar                   0.0
Gina Haspel                 0.0
```

```
Robert Wilkie               0.0
Name: tmp_mand, Length: 270, dtype: float64
```

Criando copia da tabela:

```
df = gabinete.copy()
df.shape
```

```
(312, 10)
```

Criando uma nova coluna que recebe a diferença entre as colunas "days" e "tmp_mand":

```
df["difference"] = df["days"] - df["tmp_mand"]
df
```

	president	position	appointee	start	end	length	days	inicio	fim
0	Carter	OMB Director	Bert Lance	1/21/77	9/23/77	245	247.0	1977-01-21	1977-09-23
1	Carter	Secretary of Transportation	Brock Adams	1/23/77	7/20/79	908	912.0	1977-01-23	1979-07-20
2	Carter	Secretary of Health, Education & Welfare	Joseph Califano Jr.	1/25/77	8/3/79	920	926.0	1977-01-25	1979-08-03
3	Carter	Secretary of Housing & Urban Development	Patricia Harris	1/23/77	8/3/79	922	926.0	1977-01-23	1979-08-03
4	Carter	Secretary of the Treasury	W. Michael Blumenthal	1/23/77	8/4/79	923	927.0	1977-01-23	1979-08-04
...	...	...	...	...	...	...	...	...	...
307	Trump	Secretary of Homeland	Kirstjen Nielsen	12/6/17	Still in office	NaN	NaN	2017-12-06	NaT

Encontrando a menor diferença entre as colunas "days" e "tmp_mand", ou seja, encontrar o menor valor da coluna "difference":

```
df["difference"].min()
```

```
1.0
```

Renomeando a coluna "difference":

```
renamed_df = df.rename(columns={"difference": "diferença"})
renamed_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312 entries, 0 to 311
Data columns (total 11 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   president  312 non-null    object
 1   position   312 non-null    object
 2   appointee  312 non-null    object
 3   start      312 non-null    object
 4   end        312 non-null    object
 5   length     294 non-null    object
 6   days       288 non-null    float64
 7   inicio     306 non-null    datetime64[ns]
 8   fim        288 non-null    datetime64[ns]
 9   tmp_mand   282 non-null    float64
 10  diferença  282 non-null    float64
dtypes: datetime64[ns](2), float64(3), object(6)
memory usage: 26.9+ KB
```

Excluindo as colunas "inicio", "fim", "difference" e "tmp_mand" em "df":

```
add_columns = ["inicio", "fim", "difference", "tmp_mand"]
df.drop(add_columns, inplace=True, axis=1)
df.shape
```

```
(312, 7)
```

A quantidade de cargos diferentes na coluna "position":

```
df["position"].nunique()
```

```
28
```

Verificando a frequencia dos 28 valores:

```
df["position"].value_counts()
```

```
Chief of Staff                                                  21
OMB Director                                                    19
Secretary of Commerce                                           16
UN Ambassador                                                   16
Secretary of the Treasury                                       15
Secretary of Energy                                             14
SBA Administrator                                               14
Secretary of Defense                                            14
Attorney General                                                14
Secretary of Transportation                                     14
Secretary of State                                              14
U.S. Trade Representative                                       13
Secretary of Agriculture                                        12
Secretary of Labor                                              12
```

```
Secretary of Housing & Urban Development                                    12
EPA Administrator                                                           12
Secretary of Education                                                      12
Secretary of Health & Human Services                                        11
Secretary of the Interior                                                   11
Secretary of Veterans Affairs                                               10
Director of Central Intelligence                                             9
Vice President                                                               7
Secretary of Homeland Security                                               6
CIA Director                                                                 6
Director of National Intelligence                                            5
Secretary of Health, Education & Welfare                                     1
Director of Central Intelligence/CIA Director                                1
Secretary of Health, Education & Welfare/Secretary of Health & Human Services    1
Name: position, dtype: int64
```

Transformando a coluna "position" em categoria:

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

```
CategoricalDtype(categories=['Attorney General', 'CIA Director', 'Chief of Staff',
                  'Director of Central Intelligence',
                  'Director of Central Intelligence/CIA Director',
                  'Director of National Intelligence', 'EPA Administrator',
                  'OMB Director', 'SBA Administrator',
                  'Secretary of Agriculture', 'Secretary of Commerce',
                  'Secretary of Defense', 'Secretary of Education',
                  'Secretary of Energy',
                  'Secretary of Health & Human Services',
                  'Secretary of Health, Education & Welfare',
                  'Secretary of Health, Education & Welfare/Secretary of Health & Hum
                  'Secretary of Homeland Security',
                  'Secretary of Housing & Urban Development',
                  'Secretary of Labor', 'Secretary of State',
                  'Secretary of Transportation',
                  'Secretary of Veterans Affairs', 'Secretary of the Interior',
                  'Secretary of the Treasury', 'U.S. Trade Representative',
                  'UN Ambassador', 'Vice President'],
                 ordered=False)
```

Excluindo linhas com valores nulos:

```
no_null = df.dropna()
no_null.shape
```

```
(288, 7)
```

Excluindo as colunas com valores nulos:

```
no_null = df.dropna(axis=1)
```

```
no_null.shape
```

```
(312, 5)
```

Preenchendo as linhas de valores nulos da coluna "length" com "-----":

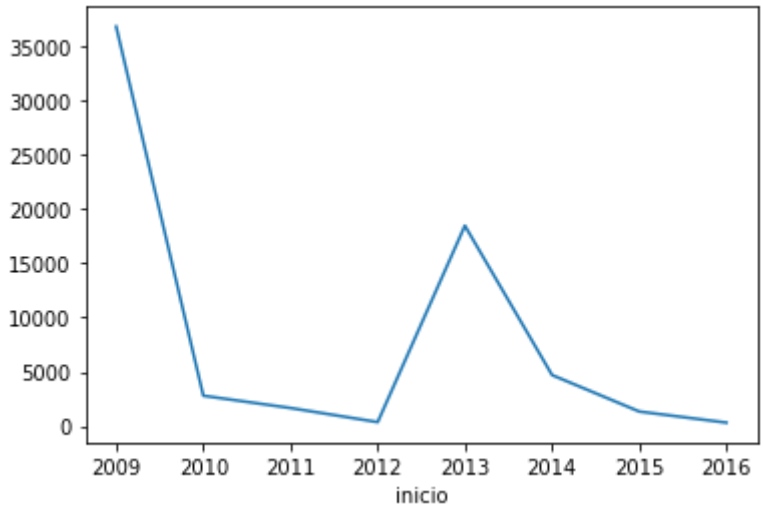
```
df["length"].fillna( value="-----", inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312 entries, 0 to 311
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype   
---  ------     --------------  -----   
 0   president  312 non-null    object  
 1   position   312 non-null    category
 2   appointee  312 non-null    object  
 3   start      312 non-null    object  
 4   end        312 non-null    object  
 5   length     312 non-null    object  
 6   days       288 non-null    float64 
dtypes: category(1), float64(1), object(5)
memory usage: 16.5+ KB
```

Gráfico mostra a quantidade de dias que os nomeados, em determinado ano, por Obama, exerceram o mandato:

```
gabinete[gabinete["president"] == "Obama"].groupby(gabinete["inicio"].dt.year)["tmp_mand"]
#Podemos observar que os nomeados em 2009, foram os que mais permaneceram no cargo, com de
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe36d31f390>
```



O gráfico de barra com os 5 cargos que mais ocorreram nomeações:

```
gabinete["position"].value_counts().head(5).plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe36d2a9f50>
```

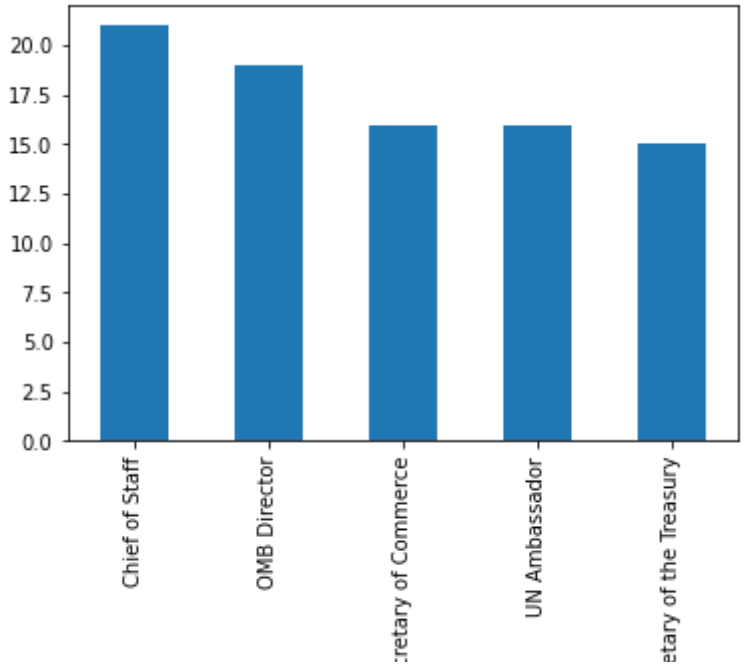


Gráfico Pizza com a proporção com que cada presidente nomeou diretores do OMB:

```
gabinete[(gabinete["position"] == "OMB Director")]["president"].value_counts().plot(kind="
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe36d21cc10>
```