Nzengou E. D.

v21eugnz@du.se

```
pd.to_datetime('today')
```

```
Timestamp('2021-04-13 14:17:13.825035')
```

AMI23B – Business Intelligence Lab1

Using Pandas to Explore a Dataset

Task1: Setting up the environment

Using colab, with all the dependencies already installed.

Task2: Using the Pandas Python Library

2.1) Create a download script download_nba_all_elo.py to download the data

```
import requests

#get the data from a download link
download_url = "https://raw.githubusercontent.com/fivethirtyeight/data/master/nba-elo/nbaalle

target_csv_path = "nba_all_elo.csv"

response = requests.get(download_url)
response.raise_for_status() #check that the request was successful

#save the file nba_all_elo.csv in your current working directory.
with open(target_csv_path, "wb") as f:
  f.write(response.content)
  print("download ready")
```

```
download ready
```

2.2) Now create a new script lab2_NBA in which you will use the Pandas Python library to take a look at your data

```
#importing Pandas in Python with the pd alias
import pandas as pd

#read in the dataset and store it as a DataFrame object in the variable nba

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

#check nba's type, it should be a DataFrame
type(nba)
```

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

2.3) Let’s see how much data is actually in nba (report these findings)

```
#len() determines the number of rows (observations) in a dataset
len(nba)
```

```
126314
```

```
#.shape determines dimensionality
#the result is a tuple containing number of rows and columns
nba.shape
```

```
(126314, 23)
```

```
#take a look at the first five rows to see the actual data
nba.head()
```

	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is_playoff
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	

```
#configure Pandas to display all 23 columns
pd.set_option("display.max.columns", None)
```

```
#show only two decimal places
pd.set_option("display.precision", 2)

#display last five rows
nba.tail()
```

	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is_p
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	

Question.1:

```
Display the first 3 rows of your dataset.
Remember that the default of nba.head() shows the first 5 rows.
```

```
nba.head(3)
```

	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is_playoff
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	

Task3: Get to Know Your Data.

3.1) Discover the different data types your dataset contains.

```
#display all columns and their data types with .info()
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
```

3.2) Showing basic statistics

Get an idea of the values each column contains.

```
#get an idea of the values each column contains
# by showing basic descrip
nba.describe()
```

```
# provide other data types by using the include parameter
import numpy as np

nba.describe(include=np.object)
```

	game_id	lg_id	date_game	team_id	fran_id	opp_id	opp_fran	game_locat
count	126314	126314	126314	126314	126314	126314	126314	126
unique	63157	2	12426	104	53	104	53	
top	198204160ATL	NBA	4/13/2011	BOS	Lakers	BOS	Lakers	

Question.2:

```
Take a look at the team_id and fran_id (franchise) columns, what observations can you make at this p
```

At first sight, it appears that team and franchise do not match: the dataset contains almost twice as many unique team IDs (104) as unique franchise IDs (53). This is surprising, we would have expected to have same number for both sides. Additionally, the most common ID is BOS for team, Lakers for franchise. This requires deeper exploration.

3.3) Exploring the dataset

Exploratory data analysis helps answer questions about your dataset, for example, exploring how often specific values occur in a column. Let's take a look at the two columns team_id and fran_id.

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

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

```
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
Huskies           60
Falcons           60
Ironmen           60
Jets              60
Name: fran_id, dtype: int64
```

It seems that a team named "Lakers" played 6024 games, but only 5078 of those were played by the Los Angeles Lakers. To find out who the other “Lakers” team is execute the following line of code

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

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

pandas.DataFrame.loc is used to access a group of rows and columns by label(s) or a boolean array. The output shows that the Minneapolis Lakers ("MNL") played the other 946 games. Let’s find out when they played those games:

```
#Find out when they played those games
nba.loc[nba["team_id"] == "MNL", "date_game"].min()
#aggregate the two functions
```

```
'1/1/1949'
```

```
nba.loc[nba["team_id"] == "MNL", "date_game"].max()
```

```
'4/9/1959'
```

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

```
min    1/1/1949
max    4/9/1959
Name: date_game, dtype: object
```

It looks like the Minneapolis Lakers played between the years of 1949 and 1959. That explains why you might not recognize this team!

Question.3 (report your answer):

```
Find out how many wins and losses the Minneapolis Lakers had, also find how many points they scored
```

```
nba.loc[nba["team_id"] == "MNL", "game_result"].value_counts()
```

```
W    524
L    422
Name: game_result, dtype: int64
```

Minneapolis Lakers won 524 times, lost 422 times (between 1949 and 1959).

```
mnl_scores = nba.loc[nba["team_id"] == "MNL", "pts"].value_counts()
```

```
mnl_scores.tail(3)
```

```
125    1
127    1
18     1
Name: pts, dtype: int64
```

```
mnl_scores.shape
```

```
(81,)
```

```
# total pts over all MNL match
nba.loc[nba["team_id"] == "MNL", "pts"].sum()
```

```
88229
```

Minneapolis Lakers have scored 626484 points during all matches contained in this dataset.

```
# save in csv format
mnl_scores.to_csv("mnl_scores.csv")
```

Question.4:

```
ow you understand why the Boston Celtics team "BOS" played the most games in the dataset, find out h
```

```
# total pts over all BOS match
nba.loc[nba["team_id"] == "BOS", "pts"].sum()
```

```
626484
```

Boston Celtics have scored 626484 points during all matches contained in this dataset.

Question.5:

```
After having explored your dataset, explain your observations from Question.2 in a structured way
```

Task4: Data access methods (loc and iloc):

Check Pandas official docs for these two functions. With data access methods like .loc and .iloc, you can select just the right subset of your DataFrame to help you answer questions about your dataset. .loc uses the label and .iloc the positional index

Question.6 (report your answer):

```
6.1) Use a data access method to display the 4th row from the bottom of the nba dataset.
```

```
nba.iloc[-4]
```

```
gameorder               63156
game_id          201506140GSW
lg_id                     NBA
_iscopy                     0
year_id                  2015
date_game           6/14/2015
seasongame                102
is_playoffs                 1
team_id                   GSW
fran_id              Warriors
pts                       104
elo_i                 1.8e+03
elo_n                 1.8e+03
win_equiv                  68
opp_id                    CLE
opp_fran            Cavaliers
opp_pts                    91
opp_elo_i             1.7e+03
opp_elo_n             1.7e+03
game_location               H
game_result                 W
forecast                 0.77
notes                     NaN
Name: 126310, dtype: object
```

```
6.2) Use a data access method to display the 2nd row from the top of the nba dataset.
```

```
nba.iloc[2]
```

```
gameorder                   2
game_id          194611020CHS
lg_id                     NBA
_iscopy                     0
year_id                  1947
date_game           11/2/1946
seasongame                  1
is_playoffs                 0
team_id                   CHS
fran_id                 Stags
pts                        63
elo_i                 1.3e+03
elo_n                 1.3e+03
win_equiv                  42
opp_id                    NYK
opp_fran               Knicks
opp_pts                    47
opp_elo_i             1.3e+03
opp_elo_n             1.3e+03
game_location               H
game_result                 W
forecast                 0.63
notes                     NaN
Name: 2, dtype: object
```

```
6.3) Access all games between the labels 5555 and 5559, you only want to see the names of teams and
```

```
nba.loc[5555:5559,["team_id", "pts"]]
```

	team_id	pts
5555	FTW	83
5556	BOS	95
5557	NYK	74
5558	ROC	81
5559	SYR	86

Task5: Querying the Dataset

You have seen how to access subsets of a huge dataset based on its indices, now you will select rows based on the values in your dataset's columns to query your data.

```
#create a new DataFrame that contains only games played after 2010
current_decade = nba[nba["year_id"] > 2010]
current_decade.shape
```

```
(12658, 23)
```

Question.7:

```
Create a new DataFrame which consists of the games played between 2000 and 2009.
```

```
df0 = nba.loc[(nba["year_id"] > 2000) & (nba["year_id"] < 2009)]

df0.head()
```

	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is_pla
87750	43876	200010310ATL	NBA	0	2001	10/31/2000	1	
87751	43876	200010310ATL	NBA	1	2001	10/31/2000	1	
87752	43877	200010310CHI	NBA	0	2001	10/31/2000	1	
87753	43877	200010310CHI	NBA	1	2001	10/31/2000	1	
87754	43878	200010310DAL	NBA	0	2001	10/31/2000	1	

```
# save the dataframe into a csv file
df0.to_csv("df0.csv")
```

Selecting rows where a specific field is not null.

```
#selecting rows where a specific field is not null .notnull() or .notna()
games_with_notes = nba[nba["notes"].notnull()]
games_with_notes.shape
```

```
(5424, 23)
```

```
#filter your dataset and find all games where the home team’s name ends with "ers".
ers = nba[nba["fran_id"].str.endswith("ers")]
ers.shape

#search for Baltimore games where both teams scored over 100 points.
#In order to see each game only once, you’ll need to exclude duplicates
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_playo
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	

Question.8:

```
 Filter your dataset and find all the playoffs games where the number of points scored by both home 
than 100, in the year 2011 and make sure you don't include duplicates (don't forget the parentheses)
```

```
ba[(nba["_iscopy"] == 0) & (nba["pts"] > 100) & (nba["opp_pts"] > 100) & (nba["year_id"] > 20

d(3)
```

	gameorder	game_id	lg_id	_iscopy	year_id	date_game	seasongame	is_playo
3659	56830	201010260LAL	NBA	0	2011	10/26/2010	1	
3668	56835	201010270GSW	NBA	0	2011	10/27/2010	1	
3673	56837	201010270MEM	NBA	0	2011	10/27/2010	1	

Task6: Grouping and Aggregating Your Data

You may also want to learn other features of your dataset, like the sum, mean, or average value of a group of elements. Luckily, the Pandas Python library offers grouping and aggregation functions to help you accomplish this task.

```
#Grouping - group all games for fran_id and sum their points and override the default of sort
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
```

```
#group by multiple columns
nba[(nba["fran_id"] == "Spurs") & (nba["year_id"] > 2010)].groupby(["year_id", "game_result"]
```

```
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
```

Question.9:

```
Take a look at the New York Knicks 2011-12 season (year_id: 2012). How many wins and losses did they
```

```
nba[(nba["fran_id"] == "Knicks") & (nba["year_id"] > 2010)  & (nba["year_id"] < 2012)].groupb
```

```
year_id  game_result
2011     L              44
         W              42
Name: game_id, dtype: int64
```

The NY Knicks won 42 games and lost 44 during the 2011-2012 season.

Task7: Manipulating Columns

You can add and drop columns as part of the initial data cleaning phase, or later based on the insights of your analysis.

```
#create a copy of your original DataFrame to work with
df = nba.copy()
df.shape

#define new columns based on the existing ones
df["difference"] = df.pts - df.opp_pts
df.shape

#use an aggregation function .max() to find the largest value of your new column
```

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

#rename the columns of your dataset
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 24 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  difference   126314 non-null  int64
dtypes: float64(6), int64(8), object(10)
memory usage: 23.1+ MB
```

```
renamed_df.head(3)
```

ts	elo_i	elo_n	win_equiv	opp_id	opp_fran	opp_pts	opp_elo_i	opp_elo_n	location
66	1300.0	1293.28	40.29	NYK	Knicks	68	1300.00	1306.72	H
68	1300.0	1306.72	41.71	TRH	Huskies	66	1300.00	1293.28	A
63	1300.0	1309.65	42.01	NYK	Knicks	47	1306.72	1297.07	H

```
# save the renamed dataframe into a csv file
renamed_df.to_csv("renamed_df.csv")
```

Note that there's a new object, renamed_df. Like several other data manipulation methods, .rename() returns a new DataFrame by default.

```
#Delete unwanted columns - wont be analyzing Elo ratings here so go ahead and delete them
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, 20)
```

Understanding the df.drop function:

```
DataFrame.drop(self, labels=None, axis=0, index=None, columns=None, level=None, inplace=False, error
```

Is translated into:
Index or column labels to drop, Whether to drop labels from the index (0 or 'index') or columns (1 or 'columns'), 'inplace=True' to make permanent changes to the dataframe.

Task8: Specifying Data Types

When you create a new DataFrame, Pandas assigns a data type to each column based on its values. Sometimes is not too accurate. Choose the correct data type for your columns upfront to improve performance.
Take another look at the columns of the nba dataset:

```
#take a look at the DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126314 entries, 0 to 126313
Data columns (total 20 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  difference     126314 non-null  int64
dtypes: float64(2), int64(8), object(10)
memory usage: 19.3+ MB
```

Ten of your columns have the data type object and some of these are good candidates for data type conversion.

```
# use .to_datetime() to specify all game dates as datetime objects.
df["date_game"] = pd.to_datetime(df["date_game"])
```

Similary, game_location can have only three different values. In a relational database, you would use the type enum for this column. Pandas provides the categorical data type for that same purpose.

```
#game_location column can have only three different values.
#you can see this by executing this code
df["game_location"].nunique()
df["game_location"].value_counts()

#change the data type to categorical and check it
df["game_location"] = pd.Categorical(df["game_location"])
df["game_location"].dtype
```

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

After changing to categorical data, execute df.info. You will notice a drop in memory usage, hence improving performance.

```
df.tail(3)
```

```
ıngame  is_playoffs  team_id  fran_id  pts  win_equiv  opp_id  opp_fran  opp_pts  game_lc
```

Question.10:

```
Find another column in the nba dataset that has a generic data type and convert it to a more specifi
```

```
# convert game_result column values (W, L) into categorical type
df["game_result"].nunique()
df["game_result"].value_counts()

#change the data type to categorical and check it
df["game_result"] = pd.Categorical(df["game_result"])
df["game_result"].dtype
```

```
CategoricalDtype(categories=['L', 'W'], ordered=False)
```

Task9: Cleaning the Data

9.1) Missing Values

.info() shows how many non-null values a column contains. That is very important information for you to have about your data. Null values often indicate a problem in the data-gathering process. When you inspect the dataset with nba.info() you will see that the dataset is quite neat except for the notes column which contains null values for most of its rows. This output shows that the notes column has only 5424 non-null values.

That means that over 120,000 rows of your dataset have null values in this column.
Here are a few ways to deal with null values:

```
#1st way- usually best approach is to ignore them, remove all rows with missing values
rows_without_missing_data = nba.dropna()
rows_without_missing_data.shape
```

```
(5424, 23)
```

```
#2nd way - Drop columns if they are not relevant to your analysis
# notes column is removed, rows remain unchanged
data_without_missing_columns = nba.dropna(axis=1)
data_without_missing_columns.shape
```

```
(126314, 22)
```

```
#3rd way - replace the missing values with a meaningful default value for your use case
# basically create dummy value to fill the void that would leave NAs
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
```

Regarding the 1st way , that kind of data clean-up doesn't make sense for your nba dataset, because it's not a problem for a game to lack notes. But if your dataset contains a million valid records and a hundred where relevant data is missing, then dropping the incomplete records can be a reasonable solution.

9.2) Invalid Values

Use .describe to understand more about your dataset. This can help you identify invalid values that may throw off your analysis.

```
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	12(
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	

Looking at the output you will see that the year_id varies between 1947 and 2015. That sounds plausible. But how can the minimum points of a game be 0.

Take a look at those games to find out if it makes sense or not.

```
#selecting the games where pts are 0
nba[nba["pts"] == 0]
```

elo_i	elo_n	win_equiv	opp_id	opp_fran	opp_pts	opp_elo_i	opp_elo_n	game_locat
1460.34	1457.45	40.41	VIR	Squires	2	1484.19	1487.08	

It seems the game was forfeited. Depending on your analysis, you may want to remove it from the dataset.

9.3) Inconsistent Values

Always check for inconsistent values. The values of the fields pts, opp_pts and game_result should be consistent with each other.

```
#check using the .empty() attribute
# here we check that teams which pts are higher than the opponent
# have indeed a winning status 'W'; empty if otherwise
# And vice-versa
nba[(nba["pts"] > nba["opp_pts"]) & (nba["game_result"] != 'W')].empty
nba[(nba["pts"] < nba["opp_pts"]) & (nba["game_result"] != 'L')].empty
```

```
True
```

Fortunately, both of these queries return an empty DataFrame. But be prepared for surprises, always check consistency.

Task10: Data Visualisation

Sometimes, the numbers speak for themselves, but often a chart helps a lot with communicating your insights. Data visualizations make big and small data easier for the human brain to understand, and visualization also makes it easier to detect patterns, trends, and outliers in groups of data.
Data visualisation is one of the things that works much better in a Jupyter notebook than in a terminal. If you need help getting started, then check out **Jupyter Notebook: An Introduction**. Both

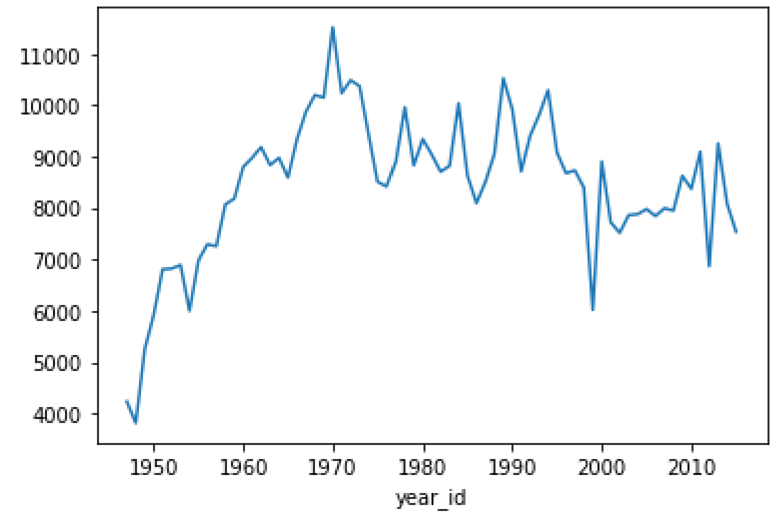
Series and DataFrame objects have a .plot() method, which is a wrapper around matplotlib.pyplot.plot().

Visualize how many points the Knicks scored throughout the seasons.

```
#Include this line to show plots directly in the notebook
%matplotlib inline
```

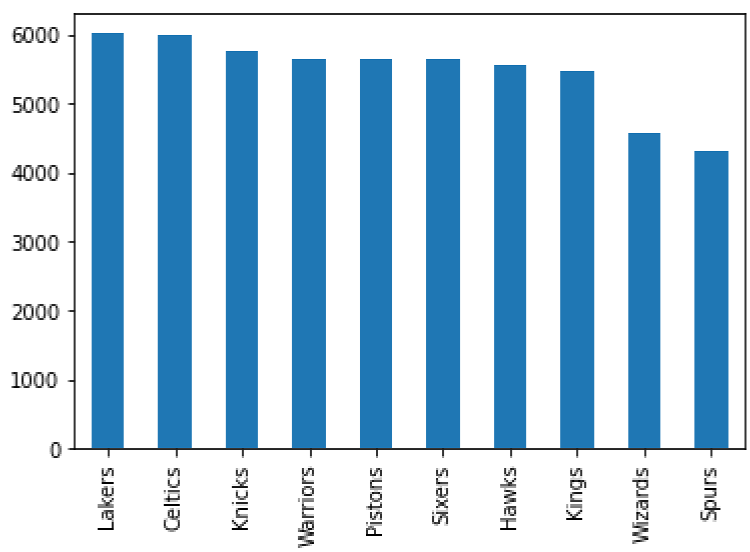
```
#Visualize how many points the Knicks scored throughout the seasons
nba[nba["fran_id"] == "Knicks"].groupby("year_id")["pts"].sum().plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f490a9a5090>
```




```
#create a bar plot to show the top 10 franchises with the most games played
nba["fran_id"].value_counts().head(10).plot(kind="bar")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f490a939610>
```



Question.11 (report your answer):

```
11.1) Explain what the above line plot, showing how many points the Knicks scored throughout the sea
```

This line plot shows several peaks and valleys, with the notable sections:

- a growth trend from 1950 (most likely when data starting being recorded) to 1970, date of the first major peak (over 11000 points);
- then a descent followed by a series of lower peak until around 1995, when there is a major fall (towards 6000 points before 2000);
- Re-increase after year 2000 followed by a valley that ends with another fall around the year 2010.

```
11.2) Describe what the above bar plot reveals to you about the franchises with the most games playe
```

The bar chart shows that the Lakers are slightly ahead of the Celtics but just by a very smal margine, on one hand. Furthermore, if we consider a benchmark of 5000 points, six other teams are above it. Then score drops for the last two teams (Wizards, Spurs) in the top 10.

```
include only the Heat’s games from 2013. Then, create a plot in the same way as you’ve seen above).
```

```
# Pie plot
# where we first define a criterion to include only the Heat’s games from 2013;
# Then, we counts the game results and finally plot it all with a kind 'pie'
nba[(nba["fran_id"] == "Heat") & (nba["year_id"] == 2013)]["game_result"].value_counts().plot
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f490a8aa910>
```

Double-click (or enter) to edit

References:

- Pandas Docs: https://pandas.pydata.org/pandas-docs/stable/index.html
- Download Python: https://www.python.org/downloads/ How to install pip: https://www.liquidweb.com/kb/install-pip-windows/
- Jupyter Notebook: https://realpython.com/jupyter-notebook-introduction/
- Install Pandas Python: https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html
- Install matplotlib: https://matplotlib.org/users/installing.html#installing-from-source
- Visualisation with Pandas: https://pandas.pydata.org/pandasdocs/stable/user_guide/visualization.html
- Tutorial inspiration (Real Python): https://realpython.com/pandas-python-explore-dataset/
- Data Source: https://fivethirtyeight.com/
- The raw data: https://raw.githubusercontent.com/fivethirtyeight/data/master/nba-elo/nbaallelo.csv
- NBA Data Analysis Using Python & Machine Learning: https://randerson112358.medium.com/nba-data-analysis-exploration-9293f311e0e8

✓ 0s completed at 4:20 PM