

Zahra Jalilpour

Email: [h19zahja@du.se](mailto:h19zahja@du.se)

Report of AMI23B – Business Intelligence Lab2

nba\_all\_elo.csv is a game information about basketball games in different years.

At first we create a script to download the data. After running the script, we will have nba\_all\_elo.csv in our current directory.

By using Pandas Python Library , we can analysis our dataset.

Here we can see the definition of the header of dataset:

| Header        | Definition   |
|---------------|--|
| gameorder     | Play order of game in NBA history  |
| game_id       | Unique ID for each game  |
| lg_id         | Which league the game was played in  |
| _iscopy       | Each row of data is tied to a single team for a single game, so _iscopy flags if this game_id has already occurred for the opposing team in the same matchup |
| year_id       | Season id, named based on year in which the season ended   |
| date_game     | Game date  |
| is_playoffs   | Flag for playoff games   |
| team_id       | Three letter code for team name, from Basketball Reference   |
| fran_id       | Franchise id. Multiple team_ids can fall under the same fran_id due to name changes or moves. Interactive is grouped by fran_id.                             |
| pts           | Points scored by team  |
| elo_i         | Team elo entering the game   |
| elo_n         | Team elo following the game  |
| win_equiv     | Equivalent number of wins in a 82-game season for a team of elo_n quality  |
| opp_id        | Team id of opponent  |
| opp_fran      | Franchise id of opponent   |
| opp_pts       | Points scored by opponent  |
| opp_elo_i     | Opponent elo entering the game   |
| opp_elo_n     | Opponent elo following the game  |
| game_location | Home (H), away (A), or neutral (N)   |
| game_result   | Win or loss for team in the team_id column   |
| forecast      | Elo-based chances of winning for the team in the team_id column, based on elo ratings and game location  |
| notes         | Additional information   |

The number of rows(Observations) is equal to : **126314**

By **nba.shape** we can find the number of rows and columns: The result is a tuple containing the number of rows and columns. **(126314, 23)**

By using **nba.head()**, we can see the first 5 rows of output.

as I use Jupyter notenook, I cannot see the output completely, but I can scroll it. It is also practical to see all the columns by:

**pd.set\_option("display.max.columns", None)**

By **nba.tail()**, we can see the last 5 rows. Here we can see all columns , because we define the command **pd.set\_option**.

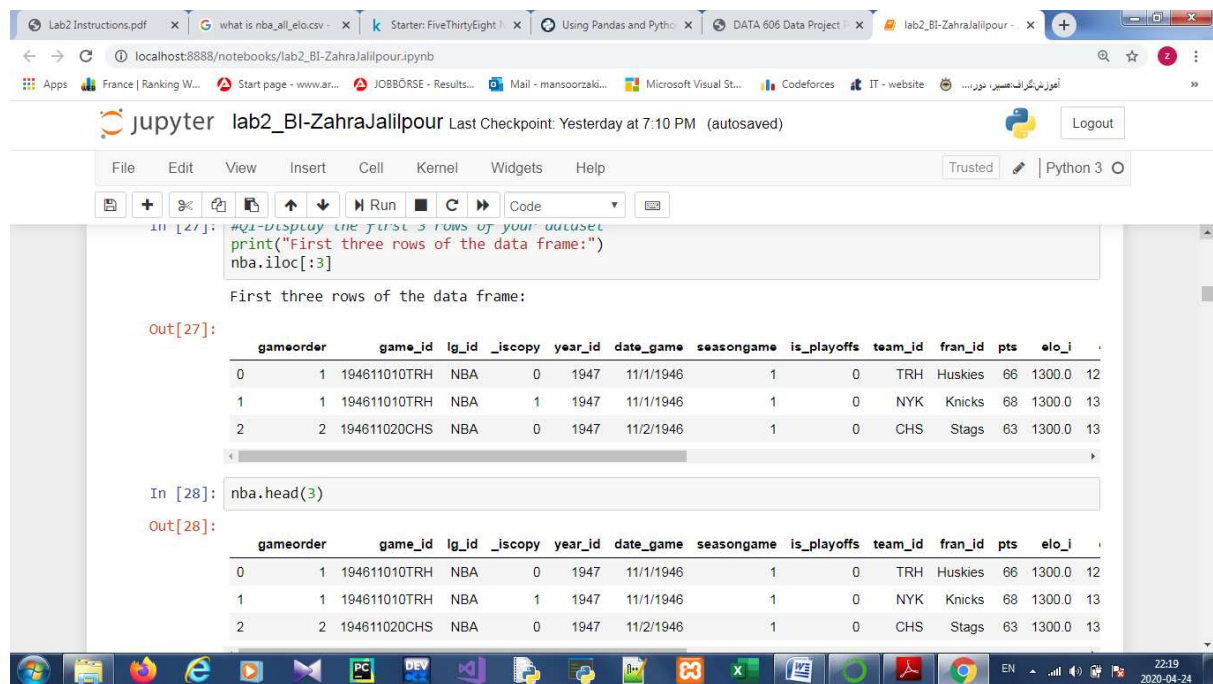
### Q1: Display the first 3 rows of your dataset.

- It can be done through different commands:

```
nba.iloc[:3]
```

or

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



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell uses `nba.iloc[:3]` to display the first three rows of the dataset. The second cell uses `nba.head(3)` to display the same data. Both outputs show a table with 13 columns: gameorder, game\_id, lg\_id, \_iscopy, year\_id, date\_game, seassongame, is\_playoffs, team\_id, fran\_id, pts, elo\_i, and another column. The data represents the first three games of the 1946-47 season for the TRH, NYK, and CHS teams.

```
In [27]: #Q1-Display the first 3 rows of your dataset
print("First three rows of the data frame:")
nba.iloc[:3]
```

First three rows of the data frame:

```
Out[27]:
```

|   | gameorder | game_id      | lg_id | _iscopy | year_id | date_game | seassongame | is_playoffs | team_id | fran_id | pts | elo_i  |
|---|-----------|--------------|-------|---------|---------|-----------|-------------|-------------|---------|---------|-----|--------|
| 0 | 1         | 194611010TRH | NBA   | 0       | 1947    | 11/1/1946 | 1           | 0           | TRH     | Huskies | 66  | 1300.0 |
| 1 | 1         | 194611010TRH | NBA   | 1       | 1947    | 11/1/1946 | 1           | 0           | NYK     | Knicks  | 68  | 1300.0 |
| 2 | 2         | 194611020CHS | NBA   | 0       | 1947    | 11/2/1946 | 1           | 0           | CHS     | Stags   | 63  | 1300.0 |

```
In [28]: nba.head(3)
```

```
Out[28]:
```

|   | gameorder | game_id      | lg_id | _iscopy | year_id | date_game | seassongame | is_playoffs | team_id | fran_id | pts | elo_i  |
|---|-----------|--------------|-------|---------|---------|-----------|-------------|-------------|---------|---------|-----|--------|
| 0 | 1         | 194611010TRH | NBA   | 0       | 1947    | 11/1/1946 | 1           | 0           | TRH     | Huskies | 66  | 1300.0 |
| 1 | 1         | 194611010TRH | NBA   | 1       | 1947    | 11/1/1946 | 1           | 0           | NYK     | Knicks  | 68  | 1300.0 |
| 2 | 2         | 194611020CHS | NBA   | 0       | 1947    | 11/2/1946 | 1           | 0           | CHS     | Stags   | 63  | 1300.0 |

In previous task we learn how to display first and last rows , size of dataset, how to import csv.file, now we learn how to examine our data. At first we see the different data type that exist in our dataset.

By `nba.info()`, we will see all columns by their data types.

```
date_game      126314 non-null object
seasongame     126314 non-null int64
is_playoffs    126314 non-null int64
team_id        126314 non-null object
fran_id        126314 non-null object
pts            126314 non-null int64
elo_i          126314 non-null float64
elo_n          126314 non-null float64
win_equiv      126314 non-null float64
opp_id         126314 non-null object
opp_fran       126314 non-null object
opp_pts        126314 non-null int64
opp_elo_i      126314 non-null float64
opp_elo_n      126314 non-null float64
game_location  126314 non-null object
game_result    126314 non-null object
forecast       126314 non-null float64
notes          5424 non-null object
dtypes: float64(6), int64(7), object(10)
memory usage: 22.2+ MB
```

we can see the data types int64, float64, and object.

Object means that all of the values in the column are strings.

Now we can see an overview of the values each column contains. By this command **nba.describe()**:

```
In [30]: #get an idea of the values each column contains
nba.describe()

Out[30]:
```

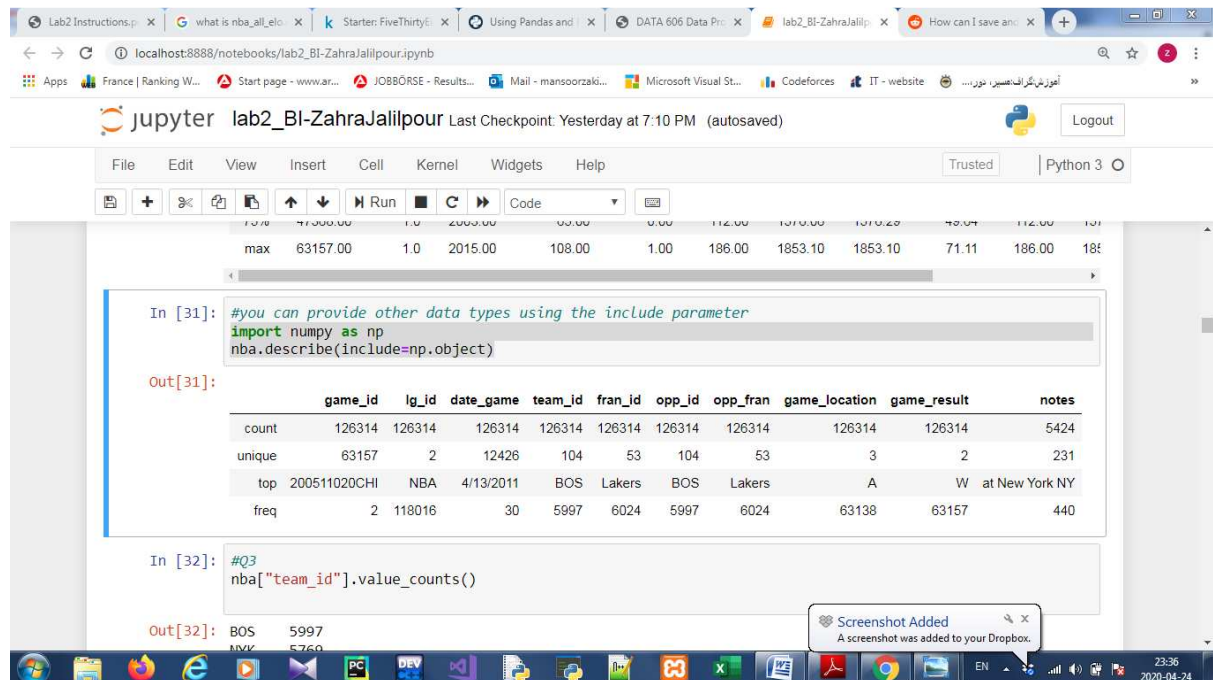
|       | gameorder | _iscopy  | year_id   | seasongame | is_playoffs | pts       | elo_i     | elo_n     | win_equiv | opp_pts   | opp_      |
|-------|-----------|----------|-----------|------------|-------------|-----------|-----------|-----------|-----------|-----------|-----------|
| count | 126314.00 | 126314.0 | 126314.00 | 126314.00  | 126314.00   | 126314.00 | 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   | 1495.24   | 41.71     | 102.73    | 1495.24   |
| std   | 18231.93  | 0.5      | 17.58     | 25.38      | 0.24        | 14.81     | 112.14    | 112.46    | 10.63     | 14.81     | 112.46    |
| min   | 1.00      | 0.0      | 1947.00   | 1.00       | 0.00        | 0.00      | 1091.64   | 1085.77   | 10.15     | 0.00      | 1085.77   |
| 25%   | 15790.00  | 0.0      | 1975.00   | 22.00      | 0.00        | 93.00     | 1417.24   | 1416.99   | 34.10     | 93.00     | 1416.99   |
| 50%   | 31579.00  | 0.5      | 1990.00   | 43.00      | 0.00        | 103.00    | 1500.95   | 1500.95   | 42.11     | 103.00    | 1500.95   |
| 75%   | 47368.00  | 1.0      | 2003.00   | 65.00      | 0.00        | 112.00    | 1576.06   | 1576.29   | 49.64     | 112.00    | 1576.29   |
| max   | 63157.00  | 1.0      | 2015.00   | 108.00     | 1.00        | 186.00    | 1853.10   | 1853.10   | 71.11     | 186.00    | 1853.10   |

```
In [31]: #you can provide other data types using the include parameter
import numpy as np
nba.describe(include=np.object)
```

.describe() only analyzes numeric columns but we can provide other data types if we use the include parameter:

```
import numpy as np
```

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



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [31]: #you can provide other data types using the include parameter
import numpy as np
nba.describe(include=np.object)
```

Out[31]:

|        | game_id      | lg_id  | date_game | team_id | fran_id | opp_id | opp_fran | game_location | game_result | notes          |
|--------|--------------|--------|-----------|---------|---------|--------|----------|---------------|-------------|----------------|
| count  | 126314       | 126314 | 126314    | 126314  | 126314  | 126314 | 126314   | 126314        | 126314      | 5424           |
| unique | 63157        | 2      | 12426     | 104     | 53      | 104    | 53       | 3             | 2           | 231            |
| top    | 200511020CHI | NBA    | 4/13/2011 | BOS     | Lakers  | BOS    | Lakers   | A             | W           | at New York NY |
| freq   | 2            | 118016 | 30        | 5997    | 6024    | 5997   | 6024     | 63138         | 63157       | 440            |

In [32]: #Q3
nba["team\_id"].value\_counts()

Out[32]: BOS 5997  
          NYK 5760

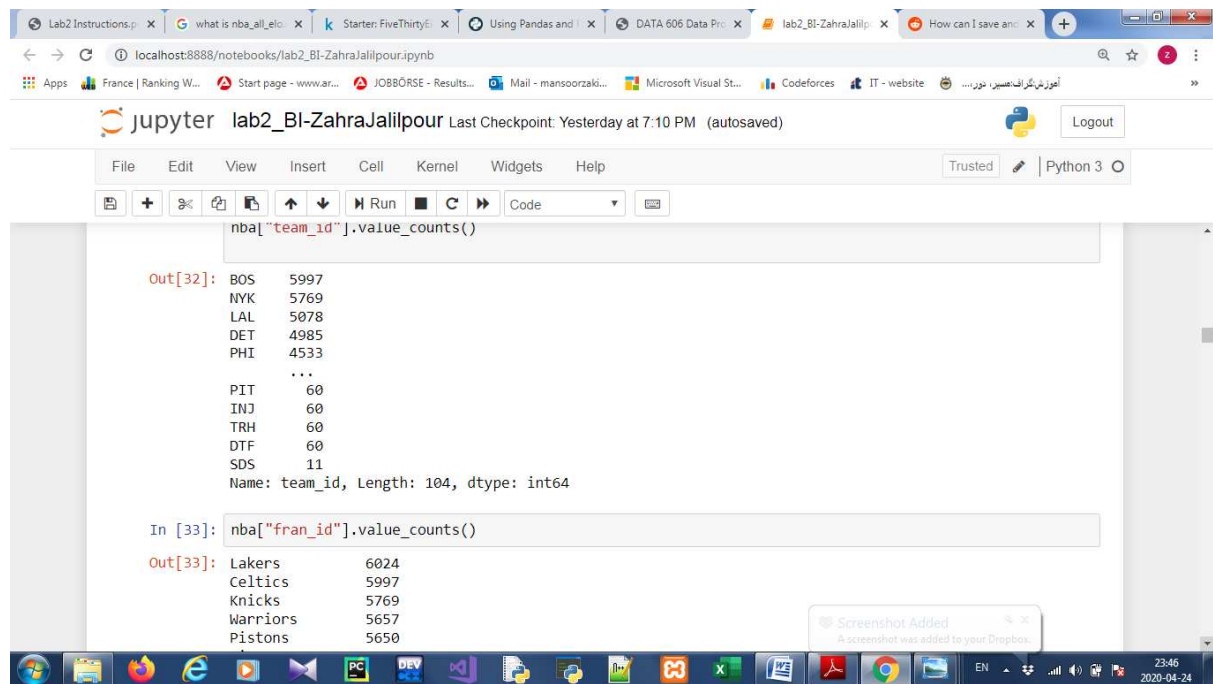
**Q2: Take a look at the team\_id and fran\_id (franchise) columns, what observations can you make at this point (i.e. do you see anything strange here)?**

By looking at team\_id and fran\_id, we see that our dataset contains 104 different team id, but 53 different fran id, and the most frequent team id is BOS, but the most frequent fran id is Lakers.

### 3.3-Exploring the dataset

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

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



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

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

The Minneapolis Lakers ("MNL") played 946 games.

**Q3: (report your answer): Find out how many wins and losses the Minneapolis Lakers had, also find how many points they scored during the matches contained in the dataset.**

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

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

The Minneapolis Lakers ("MNL") played 946 games and during these games, the number of win is 524 and the number of loss is 422.

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

```
88229
```

The number of points of "MNL" is equal to 88229.

```

Out[37]: min 1/1/1949
max 4/9/1959
Name: date_game, dtype: object

In [93]: nba.loc[nba["team_id"] == "MNL", "game_result"].value_counts()
Out[93]: W 524
L 422
Name: game_result, dtype: int64

In [94]: nba.loc[nba["team_id"] == "MNL", "pts"].sum()
Out[94]: 88229

In [43]: #Q4
nba.loc[nba["team_id"] == "BOS", "pts"].sum()
Out[43]: 626484

In [44]: #Q6
nba.iloc[-4]

```

**Q4: (report your answer): Now you understand why the Boston Celtics team "BOS" played the most games in the dataset, find out how many points the Boston Celtics have scored during all matches contained in this dataset.**

By this command we can find the total points that Boston have scored during all matches:

```
nba.loc[nba["team_id"] == "BOS", "pts"].sum()
```

The Boston Celtics scored a total of 626,484 points.

**Q5: (report your answer): After having explored your dataset, explain your observations from Question.2 in a structured way.**

It looks like the Minneapolis Lakers played between the years of 1949 and 1959 and "LAL" played between the years of 1961 and 2009 and Boston played between 1948 and 2012. That explains why we might not recognize Minneapolis Lakers team!

We've also found out why the Boston Celtics team "bos" played the most games in the dataset.

**Q6-**

**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[1]`

```

gameorder      1
game_id        194611010TRH
lg_id          NBA
_iscopy        1
year_id        1947
date_game      11/1/1946
seasongame     1
is_playoffs    0
team_id        NYK
fran_id        Knicks
pts            68
elo_i          1.3e+03
elo_n          1.3e+03
win_equiv      42
opp_id         TRH
opp_fran       Huskies
opp_pts        66
opp_elo_i      1.3e+03
opp_elo_n      1.3e+03
game_location  A
game_result    W
forecast       0.36
notes         NaN
Name: 1, dtype: object

```

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

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

The screenshot shows a JupyterLab interface with a code cell and its output. The code cell contains the following code:

```

In [101]: #Q6-3
nba.loc[5555:5559, ["team_id", "fran_id", "pts"]]

```

The output of the code cell is a DataFrame with the following data:

|      | team_id | fran_id | pts |
|------|---------|---------|-----|
| 5555 | FTW     | Pistons | 83  |
| 5556 | BOS     | Celtics | 95  |
| 5557 | NYK     | Knicks  | 74  |
| 5558 | ROC     | Kings   | 81  |
| 5559 | SYR     | Sixers  | 86  |

Below the first code cell, there are two more code cells. The second code cell contains the following code:

```

In [47]: #create a new DataFrame that contains only games played after 2010
current_decade = nba[nba["year_id"] > 2010]
current_decade.shape

```

The output of the second code cell is:

```

Out[47]: (12658, 23)

```

The third code cell contains the following code:

```

In [48]: #Q7
new_nba = nba[(nba["year_id"]>2000) & (nba["year_id"]<2009)]

```



Question.7 (report your answer): Create a new DataFrame which consists of the games played between 2000 and 2009.

```
new_nba = nba[(nba["year_id"]>=2000) & (nba["year_id"]<=2009)]
```

```
new_nba.shape
```

```
(25810, 23)
```

Question 8:

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

```
nba[(nba["_iscopy"] == 0) & (nba["pts"] > 100) & (nba["year_id"] == 2011) &
(nba["is_playoffs"]==1) & (nba["opp_pts"]>100) ]
```

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [49]: #Q8
nba[(nba["_iscopy"] == 0) & (nba["pts"] > 100) & (nba["year_id"] == 2011) & (nba["is_playoffs"]==1) &
(nba["opp_pts"]>100) ]
```

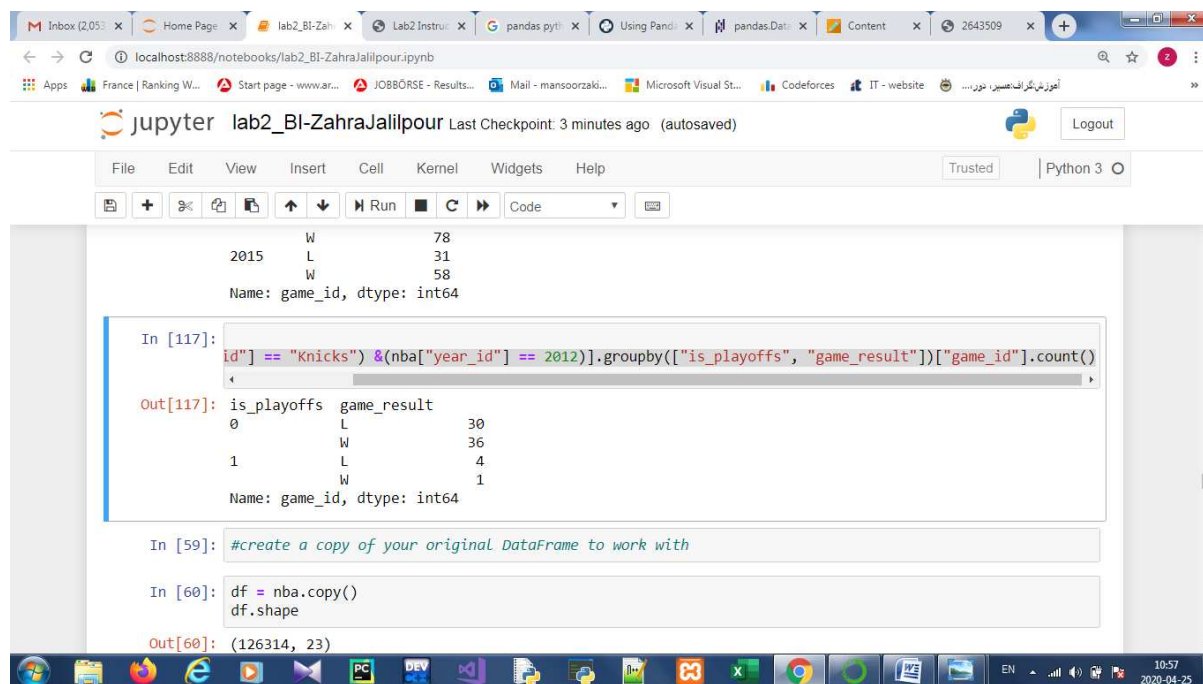
Out[49]:

|        | gameorder | game_id      | lg_id | _iscopy | year_id | date_game | seasonsgame | is_playoffs | team_id | fran_id   | pts |
|--------|-----------|--------------|-------|---------|---------|-----------|-------------|-------------|---------|-----------|-----|
| 116128 | 58065     | 201104170OKC | NBA   | 0       | 2011    | 4/17/2011 | 83          | 1           | OKC     | Thunder   | 107 |
| 116178 | 58090     | 201104250DEN | NBA   | 0       | 2011    | 4/25/2011 | 86          | 1           | DEN     | Nuggets   | 104 |
| 116193 | 58097     | 201104270SAS | NBA   | 0       | 2011    | 4/27/2011 | 87          | 1           | SAS     | Spurs     | 110 |
| 116205 | 58103     | 201105010OKC | NBA   | 0       | 2011    | 5/1/2011  | 88          | 1           | OKC     | Thunder   | 101 |
| 116213 | 58107     | 201105030OKC | NBA   | 0       | 2011    | 5/3/2011  | 89          | 1           | OKC     | Thunder   | 111 |
| 116233 | 58117     | 201105090MEM | NBA   | 0       | 2011    | 5/9/2011  | 92          | 1           | MEM     | Grizzlies | 123 |
| 116248 | 58125     | 201105170DAL | NBA   | 0       | 2011    | 5/17/2011 | 93          | 1           | DAL     | Mavericks | 121 |
| 116258 | 58130     | 201105230OKC | NBA   | 0       | 2011    | 5/23/2011 | 98          | 1           | OKC     | Thunder   | 105 |
| 116275 | 58138     | 201106090DAL | NBA   | 0       | 2011    | 6/9/2011  | 102         | 1           | DAL     | Mavericks | 112 |

Question.9 (report your answer): Take a look at the New York Knicks 2011-12 season (year\_id: 2012). How many wins and losses did they score during the regular season and the playoffs?

```
nba[(nba["fran_id"] == "Knicks") & (nba["year_id"] == 2012)].groupby(["is_playoffs",
"game_result"])["game_id"].count()
```

```
is_playoffs  game_result
0            L           30
             W           36
1            L           4
             W            1
Name: game_id, dtype: int64
```



Question.10 (report your answer): Find another column in the nba dataset that has a generic data type and convert it to a more specific one.

By using `df.info()` we see that ten of your columns have the data type object. Some of these objects can be converted to specific data type, like `game_date`, `game_location` and `game result`.

When we specify the categorical data type, we make validation easier and save a ton of memory.

```
df["game result"].nunique()
```

```
df["game result"].value counts()
```

```
df["game result"] = pd.Categorical(df["game result"])
```

```

In [72]: #Q10-
df["game_result"].nunique()

Out[72]: 2

In [73]: df["game_result"].value_counts()

Out[73]: W    63157
         L    63157
         Name: game_result, dtype: int64

In [74]: df["game_result"] = pd.Categorical(df["game_result"])

In [75]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126314 entries, 0 to 126313
Data columns (total 20 columns):
gameorder    126314 non-null int64
game_id      126314 non-null object
le_id        126314 non-null object

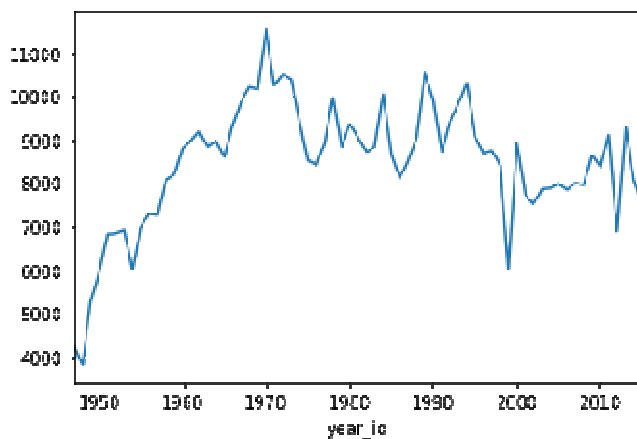
```

Question.10 (report your answer):

10.1) Explain what the above line plot, showing how many points the Knicks scored throughout the seasons, reveals to you (i.e. describe what you find out).

%matplotlib inline

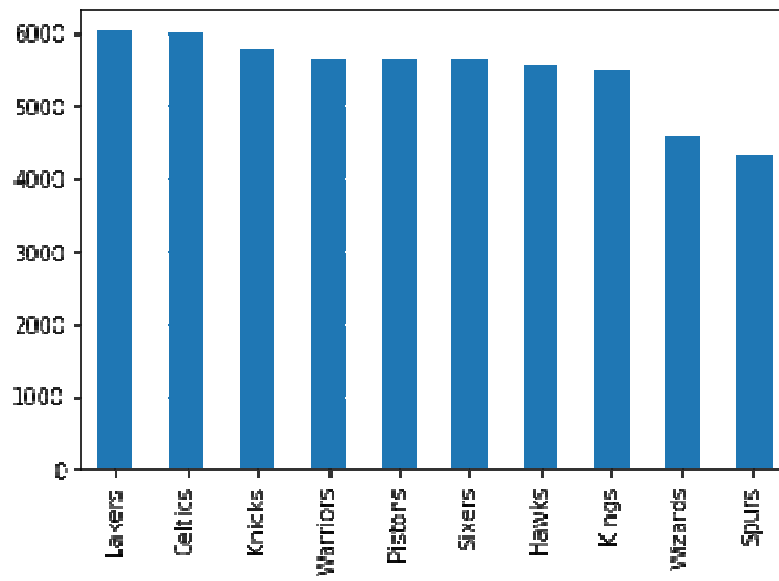
nba[nba["fran\_id"] == "Knicks"].groupby("year\_id")["pts"].sum().plot()



This shows a line plot with several peaks and two notable valleys around the years 2000 and 2010.

10.2) Describe what the above bar plot reveals to you about the franchises with the most games played.

nba["fran\_id"].value\_counts().head(10).plot(kind="bar")



The Lakers and Celtics have the most games, and there are six further teams with a game count above 5000.