CS 241

13 Prove Milestone - Data Analysis

PART I

Jose Israel Carmona Morales

To initialize the libraries and load the data:

```
import pandas as pd #The library that manipulates our data
import seaborn as sns #Used to plotting and graphing
import matplotlib.pyplot as plt #If we need any low level methods

bestb_players = pd.read_csv("basketball_players.csv")
print(bestb_players.head())
print(bestb_players.columns)
```

OUTPUT

```
    playerID  year  stint tmID lgID  GP  GS  minutes  ...  PostPF  PostfgAttempted  PostfgMade  PostftAttempted  PostftMade  PostthreeAttempted  PostthreeMade  note
0  abramjo01  1946      1  PIT  NBA  47   0        0  ...       0                0           0                0           0                   0              0   NaN
1  aubucch01  1946      1  DTF  NBA  30   0        0  ...       0                0           0                0           0                   0              0   NaN
2  bakerno01  1946      1  CHS  NBA   4   0        0  ...       0                0           0                0           0                   0              0   NaN
3  baltihe01  1946      1  STB  NBA  58   0        0  ...       3               10           2                1           0                   0              0   NaN
4   barrjo01  1946      1  STB  NBA  58   0        0  ...       0                0           0                0           0                   0              0   NaN

[5 rows x 42 columns]
Index(['playerID', 'year', 'stint', 'tmID', 'lgID', 'GP', 'GS', 'minutes',
       'points', 'oRebounds', 'dRebounds', 'rebounds', 'assists', 'steals',
       'blocks', 'turnovers', 'PF', 'fgAttempted', 'fgMade', 'ftAttempted',
       'ftMade', 'threeAttempted', 'threeMade', 'PostGP', 'PostGS',
       'PostMinutes', 'PostPoints', 'PostoRebounds', 'PostdRebounds',
       'PostRebounds', 'PostAssists', 'PostSteals', 'PostBlocks',
       'PostTurnovers', 'PostPF', 'PostfgAttempted', 'PostfgMade',
       'PostftAttempted', 'PostftMade', 'PostthreeAttempted', 'PostthreeMade',
       'note'],
      dtype='object')
```

REQUIREMENT 01.

It finds the mean and the median of numbers of points scored.

```
print("Mean")
print(bestb_players.mean())
print("Median")
print(bestb_players.median())
```

MEAN

```
Mean
c:\Users\User\OneDrive - BYU-Idaho\Israel\BYU-I\3rd Semester\CS241 Spring 2021\12 P
ns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future ve
duction.
  print(bestb_players.mean())
year                1982.914235
stint                  1.034651
GP                    47.964338
GS                     0.935624
minutes             1097.296661
points               492.130689
oRebounds             50.382594
dRebounds            112.825271
rebounds             209.064208
assists              107.060376
steals                29.347733
blocks                18.055240
turnovers             55.341796
PF                   112.851375
fgAttempted          410.599764
fgMade               188.547640
ftAttempted          135.557998
ftMade               101.443981
threeAttempted        36.424361
threeMade             12.578502
PostGP                 3.180582
PostGS                 0.139784
PostMinutes           74.552145
PostPoints            32.665614
PostoRebounds          3.095028
PostdRebounds          7.134100
PostRebounds          14.089175
PostAssists            6.824260
PostSteals             1.768431
PostBlocks             1.193844
PostTurnovers          3.518378
PostPF                 7.878363
PostfgAttempted       27.076544
PostfgMade            12.292072
PostftAttempted        9.505873
PostftMade             7.163446
PostthreeAttempted     2.621742
PostthreeMade          0.918530
dtype: float64
```

MEDIAN

```
Median
c:\Users\User\OneDrive - BYU-Idaho\Israel\BYU-I\3rd Semester\CS241 Spring 2021\12 P
ns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future ve
duction.
  print(bestb_players.median())
year                1986.0
stint                  1.0
GP                    53.0
GS                     0.0
minutes              866.0
points               329.0
oRebounds             20.0
dRebounds             52.0
rebounds             133.0
assists               58.0
steals                12.0
blocks                 4.0
turnovers             24.0
PF                   101.0
fgAttempted          287.0
fgMade               127.0
ftAttempted           85.0
ftMade                60.0
threeAttempted         1.0
threeMade              0.0
PostGP                 0.0
PostGS                 0.0
PostMinutes            0.0
PostPoints             0.0
PostoRebounds          0.0
PostdRebounds          0.0
PostRebounds           0.0
PostAssists            0.0
PostSteals             0.0
PostBlocks             0.0
PostTurnovers          0.0
PostPF                 0.0
PostfgAttempted        0.0
PostfgMade             0.0
PostftAttempted        0.0
PostftMade             0.0
PostthreeAttempted     0.0
PostthreeMade          0.0
dtype: float64
```

REQUIREMENT 02.

It finds the highest number of points per season, sorted the data by points, year and shows the highest 5.

```
print(bestb_players[["playerID", "year", "points"]].sort_values("points", ascending = Fals
e).head(5))
```

OUTPUT

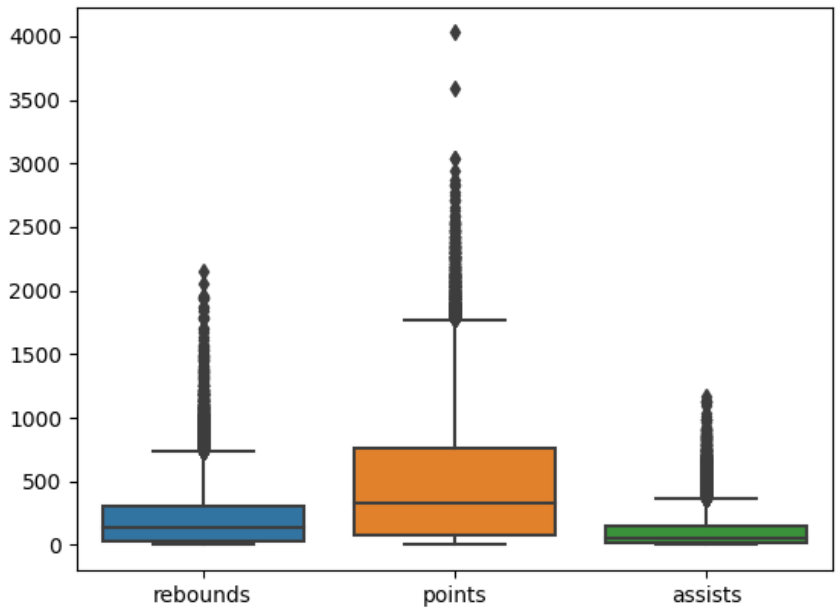
	playerID	year	points
2078	chambwi01	1961	4029
2199	chambwi01	1962	3586
9769	jordami01	1986	3041
1972	chambwi01	1960	3033
2324	chambwi01	1963	2948

REQUIREMENT 03.

Produces a boxplot that shows the distribution of total points, total assists, and total rebounds.

```
sns.boxplot(data = bestb_players[["rebounds", "points", "assists"]])
plt.show()
```

OUTPUT



REQUIREMENT 04.

Produces a boxplot that shows the distribution of total points, total assists, and total rebounds.

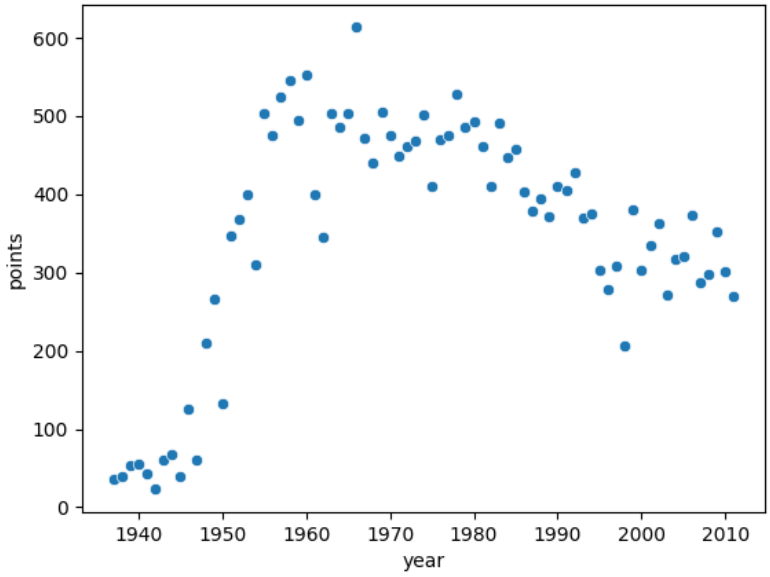
```
year_nbagroup = bestb_players[["points", "year"]].groupby("year").median()
print(year_nbagroup.head())

year_nbagroup = year_nbagroup.reset_index()
print(year_nbagroup.head())

sns.scatterplot(data = year_nbagroup, x = "year", y = "points")
plt.show()
```

OUTPUT

```
        points
year
1937    36.0
1938    39.0
1939    53.5
1940    54.5
1941    43.0
   year  points
0  1937    36.0
1  1938    39.0
2  1939    53.5
3  1940    54.5
4  1941    43.0
```



PART II

To import libraries

```
import pandas
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

It displays more columns

```
pandas.set_option("display.max_columns", 10)
```

REQUIREMENT 01.

Reads the csv file

```
players = pd.read_csv("basketball_players.csv")
print(players.columns)
```

OUTPUT

```
Index(['playerID', 'year', 'stint', 'tmID', 'lgID', 'GP', 'GS', 'minutes',
       'points', 'oRebounds', 'dRebounds', 'rebounds', 'assists', 'steals',
       'blocks', 'turnovers', 'PF', 'fgAttempted', 'fgMade', 'ftAttempted',
       'ftMade', 'threeAttempted', 'threeMade', 'PostGP', 'PostGS',
       'PostMinutes', 'PostPoints', 'PostoRebounds', 'PostdRebounds',
       'PostRebounds', 'PostAssists', 'PostSteals', 'PostBlocks',
       'PostTurnovers', 'PostPF', 'PostfgAttempted', 'PostfgMade',
       'PostftAttempted', 'PostftMade', 'PostthreeAttempted', 'PostthreeMade',
       'note'],
      dtype='object')
```

It creates the field goal success percent

```
players["fgSuccessPercent"] = players["fgMade"] / players["fgAttempted"]
```

It finds the players that have tried more than 0 shots

```
players = players[(players.fgAttempted > 0) & (players.fgSuccessPercent <= 1)]
```

Finds the free throw success percent

```
players["ftSuccessPercent"] = players["ftMade"] / players["ftAttempted"]
players = players[(players.ftAttempted > 0) & (players.ftSuccessPercent <= 1)]
```

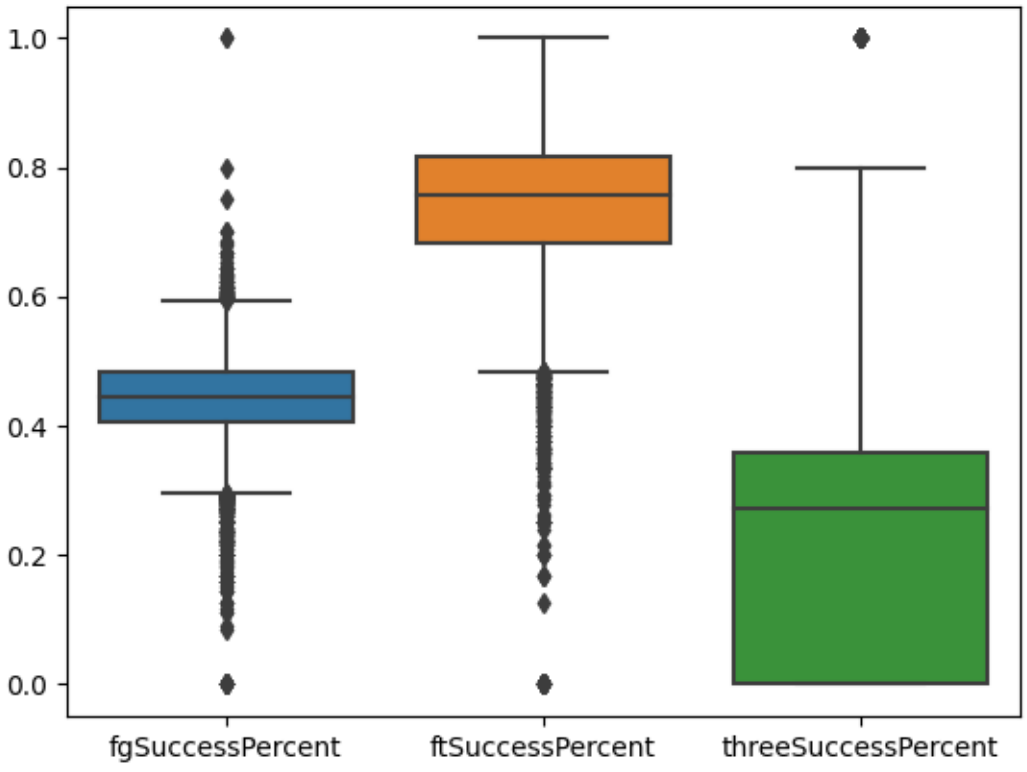
Finds the three-success percent.

```
players["threeSuccessPercent"] = players["threeMade"] / players["threeAttempted"]
players = players[(players.threeAttempted > 0) & (players.threeSuccessPercent <= 1)]
```

It shows the distribution of the percentages.

```
sns.boxplot(data = players[["fgSuccessPercent", "ftSuccessPercent", "threeSuccessPercent"]
])
plt.show()
```

OUTPUT



REQUIREMENT 02.

It finds players that have done more than 150 points per season

```
over_150 = players[(players.points > 150)]
over_150 = players[(players.ftSuccessPercent > .6) & (players.fgSuccessPercent > .6) & (pl
ayers.threeSuccessPercent > .6)]
print(over_150[["playerID", "fgSuccessPercent", "ftSuccessPercent", "threeSuccessPercent",
 "assists", "rebounds"]])
```

OUTPUT

```
        playerID  fgSuccessPercent  ftSuccessPercent  threeSuccessPercent  \
7993   gilmoar01          0.652330          0.768116             1.000000
13644  pritcke01          0.666667          0.666667             1.000000
18440  johnsam01          0.700000          1.000000             0.666667
21167  stojape01          0.700000          1.000000             0.666667

       assists  rebounds
7993       136       835
13644        7         2
18440        3         4
21167        1         3
```

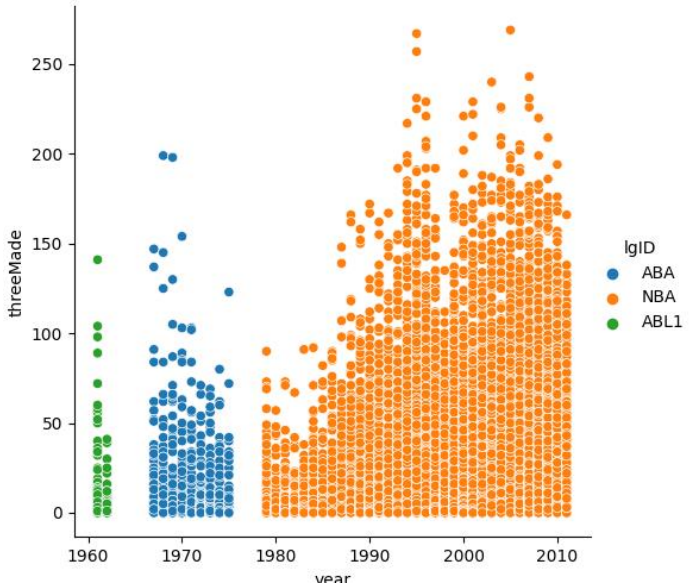
REQUIREMENT 03.

It finds the mean and median per year by leagues.

```
year_group = players.groupby('year')
three_stats = year_group['threeMade'].agg([np.mean, np.median])
three_stats = three_stats.reset_index()
three_stats = pd.melt(three_stats, id_vars = ["year"], var_name = "stat")
print(three_stats)
```

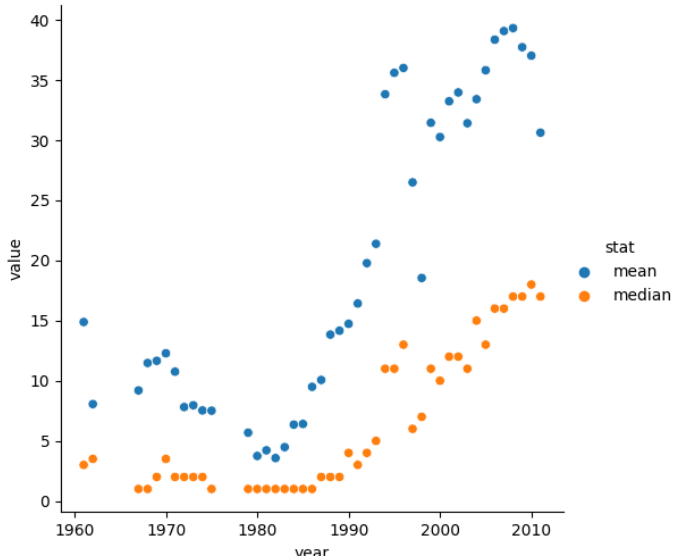
It shows the main distribution between the different leagues.

```
sns.relplot(data = players, x = "year", y = "threeMade", hue = "lgID")
plt.show()
```



It shows the trend and the distribution between the mean and median.

```
sns.relplot(data = three_stats, x = "year", y = "value", hue = "stat")
plt.show()
```



PART III

To import libraries

```
import pandas as pd
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
```

To show more columns and read the files

```
pandas.set_option("display.max_columns", 10)
players = pd.read_csv("basketball_players.csv")
master = pd.read_csv("basketball_master.csv")
```

REQUIREMENT 01.

It merges the two data sets and prints the players with the highest number of points and displays the GOAT: Michael Jordan.

```
players = pd.merge(players, master, how="left", left_on="playerID", right_on="bioID")
print(players.columns)
```

	playerID	firstName	lastName	points	assists	steals	blocks	\
10139	jordami01	Michael	Jordan	2868	485	259	131	
10932	jordami01	Michael	Jordan	2753	519	227	54	
10528	jordami01	Michael	Jordan	2633	650	234	65	
11359	jordami01	Michael	Jordan	2580	453	223	83	
5976	ervinju01	Julius	Erving	2462	423	207	160	
10053	drexlcl01	Clyde	Drexler	2185	467	203	52	

It creates the throw success percentages.

```
players["fgSuccessPercent"] = players["fgMade"] / players["fgAttempted"]
players = players[(players.fgAttempted > 0) & (players.fgSuccessPercent <= 1)]
players["ftSuccessPercent"] = players["ftMade"] / players["ftAttempted"]
players = players[(players.ftAttempted > 0) & (players.ftSuccessPercent <= 1)]
players["threeSuccessPercent"] = players["threeMade"] / players["threeAttempted"]
players = players[(players.threeAttempted > 0) & (players.threeSuccessPercent <= 1)]
```

	fgSuccessPercent	ftSuccessPercent	threeSuccessPercent
10139	0.535035	0.840698	0.132075
10932	0.526477	0.848355	0.375510
10528	0.538162	0.849937	0.275510
11359	0.538922	0.850969	0.311828
5976	0.506674	0.800604	0.330097
10053	0.505658	0.810903	0.211538

REQUIREMENT 02.

It finds how many players are in each state, and California has the highest number of them.

```
print(players["birthState"].value_counts())
```

```
CA    1248
NY     964
IL     813
MI     575
PA     552
OH     498
TX     482
```

Look through California to see which city has the highest number of players.

```
CAcity = players[players.birthState == "CA"]
print(CAcity["birthCity"].value_counts())
```

```
Los Angeles      367
Oakland          127
San Francisco     50
San Diego         49
Long Beach        46
```

It prints the best players of Los Angeles.

```
LAcity = players[players.birthCity == "Los Angeles"]
print(LAcity[["firstName", "lastName", "points", "assists", "steals", "blocks", "fgSuccess
Percent", "ftSuccessPercent", "threeSuccessPercent"]].sort_values("points", ascending=Fals
e))
```

```
       firstName   lastName  points  assists  steals  blocks  fgSuccessPercent  \
19446    Richard  Jefferson    1857      252      76      21          0.466466
19309      Baron      Davis    1791      624     191      43          0.425950
19974    Richard  Jefferson    1607      199      66      14          0.439444
7882     Freeman   Williams    1585      164      91       5          0.464881
17273      Baron      Davis    1532      501     158      27          0.395150
```

REQUIREMENT 03.

Shows the correlation between points with the time.

```
points_time = players[players.minutes >= 0]
sns.relplot(data = points_time, x = "minutes", y = "points", hue = "GP")
plt.show()
```