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

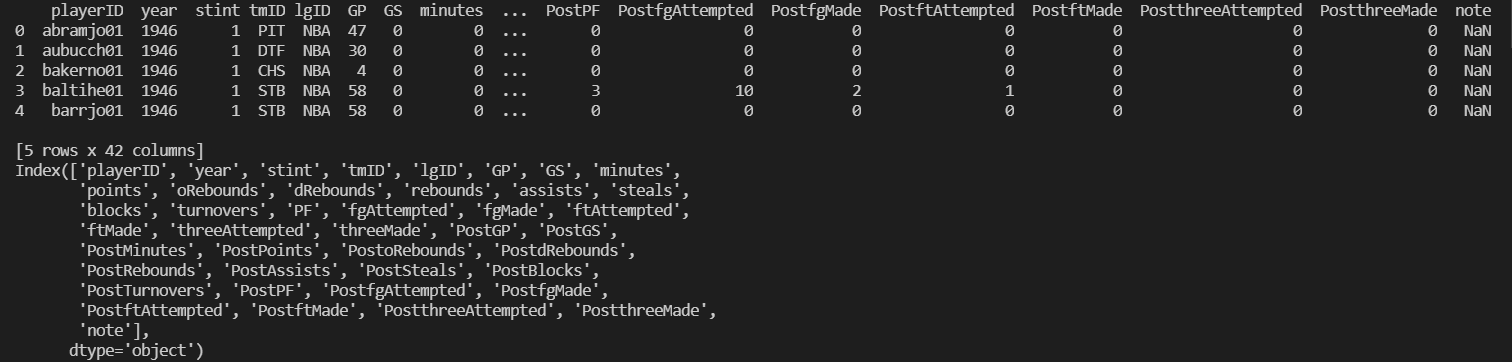
import matplotlib.pyplot as plt #If we need any low level methods

bestb\_players = pd.read\_csv("basketball\_players.csv")

print(bestb\_players.head())

print(bestb\_players.columns)

**OUTPUT**

****

**REQUIREMENT 01.**

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

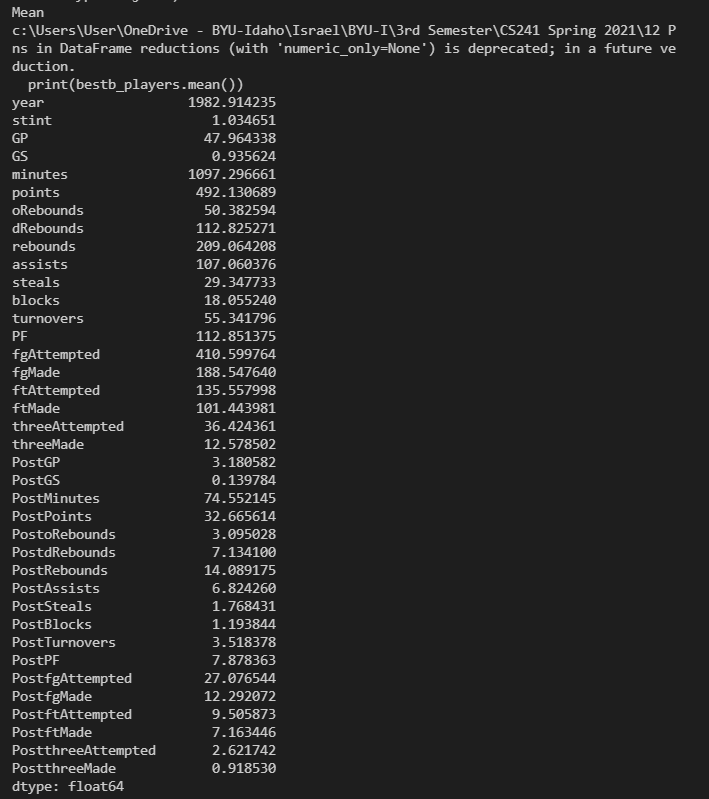
print("Mean")

print(bestb\_players.mean())

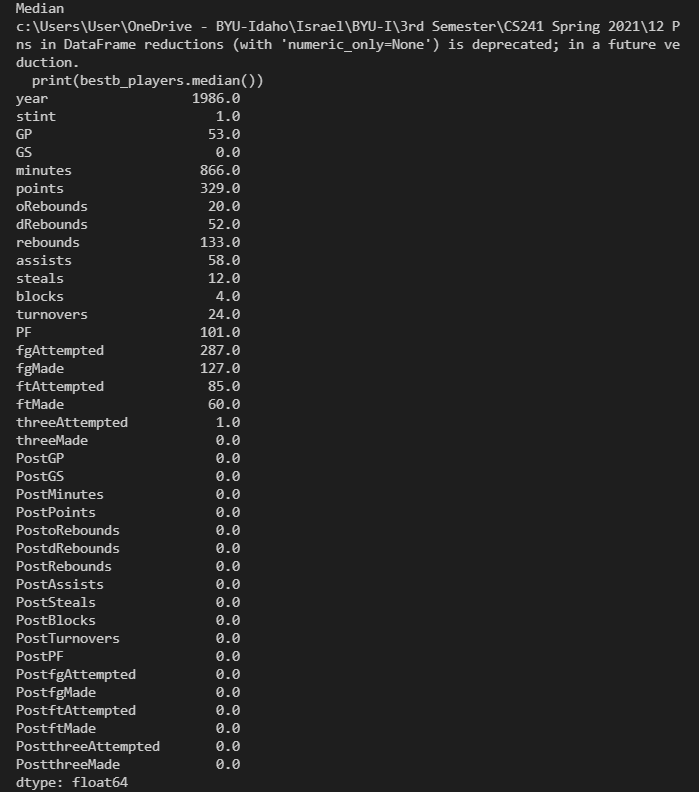
print("Median")

print(bestb\_players.median())

**MEAN**



**MEDIAN**

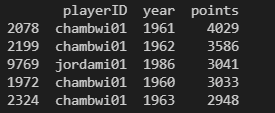


**REQUIREMENT 02.**

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 = False).head(5))

**OUTPUT**



**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**

**Chart, box and whisker chart

Description automatically generated**

**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())

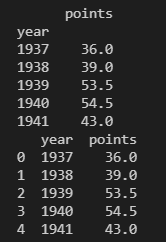
year\_nbagroup = year\_nbagroup.reset\_index()

print(year\_nbagroup.head())

sns.scatterplot(data = year\_nbagroup, x = "year", y = "points")

plt.show()

**OUTPUT**

 **Chart, scatter chart

Description automatically generated**

**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**

**Text

Description automatically generated**

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**

**Chart, box and whisker chart

Description automatically generated**

**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) & (players.threeSuccessPercent > .6)]

print(over\_150[["playerID", "fgSuccessPercent", "ftSuccessPercent", "threeSuccessPercent", "assists", "rebounds"]])

**OUTPUT**

**Text

Description automatically generated**

**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()

**Chart, scatter chart

Description automatically generated**

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()

**Chart, scatter chart

Description automatically generated**

**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")

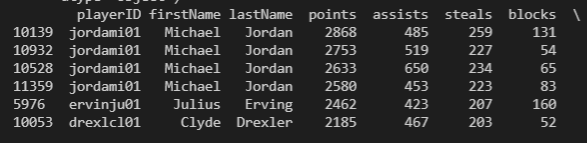
master = pd.read\_csv("basketball\_master.csv")

**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)

****

It creates the throw success percentages.

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

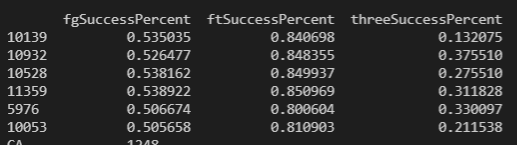
players = players[(players.fgAttempted > 0) & (players.fgSuccessPercent <= 1)]

players["ftSuccessPercent"] = players["ftMade"] / players["ftAttempted"]

players = players[(players.ftAttempted > 0) & (players.ftSuccessPercent <= 1)]

players["threeSuccessPercent"] = players["threeMade"] / players["threeAttempted"]

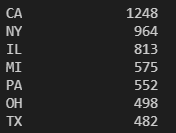
players = players[(players.threeAttempted > 0) & (players.threeSuccessPercent <= 1)]

****

**REQUIREMENT 02.**

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

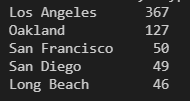
print(players["birthState"].value\_counts())

****

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

CAcity = players[players.birthState == "CA"]

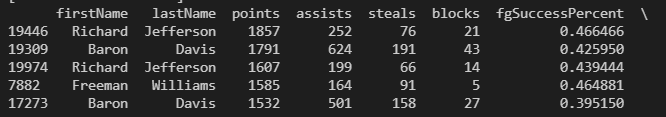
print(CAcity["birthCity"].value\_counts())

****

It prints the best players of Los Angeles.

LAcity = players[players.birthCity == "Los Angeles"]

print(LAcity[["firstName", "lastName", "points", "assists", "steals", "blocks", "fgSuccessPercent", "ftSuccessPercent", "threeSuccessPercent"]].sort\_values("points", ascending=False))

****

**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()

**Chart, scatter chart

Description automatically generated**