Baseball Data Analysis

Austin Wilson

Elevator pitch

In the project i will use SQL queries to pull data from a database to help to illustrate data in a meaningfull way. What i hope to show is my knowledge of how to interact with SQL queries within python and how then to visualize the data.

TECHNICAL DETAILS

GRAND QUESTION 1

Write an SQL query to create a new dataframe about baseball players who attended BYU-Idaho. The new table should contain five columns: playerID, schoolID, salary, and the yearID/teamID associated with each salary. Order the table by salary (highest to lowest) and print out the table in your report.

To answer this question i frist started by joining the tables of CollegePlaying and Salaries. Then i set the where condtion equal to 'idbyuid' so that it would return only the players that atteneded BYUI.

```
baseball_byui = pd.read_sql_query("""
SELECT CollegePlaying.playerid, CollegePlaying.schoolid,
Salaries.salary, CollegePlaying.yearid, Salaries.teamid
FROM Collegeplaying
    JOIN Salaries
    ON Collegeplaying.playerid = Salaries.playerid
WHERE schoolID = 'idbyuid'
ORDER BY salary DESC
""", con
)
```

Below is the results of the query. This data shows all of the different salaries of players that went to BYUI seperated by year and school. What we can clearly see from this is that lindsma01 clearly has is the highest paid player that attended BYUI.

	playerID	schoolID	salary	yearID	teamID
0	lindsma01	idbyuid	4000000	2001	CHA
1	lindsma01	idbyuid	4000000	2002	CHA
2	lindsma01	idbyuid	3600000	2001	BAL
3	lindsma01	idbyuid	3600000	2002	BAL
4	lindsma01	idbyuid	2800000	2001	COL
5	lindsma01	idbyuid	2800000	2002	COL

	playerID	schoolID	salary	yearID	teamID
6	lindsma01	idbyuid	2300000	2001	CHA
7	lindsma01	idbyuid	2300000	2002	CHA
8	lindsma01	idbyuid	1625000	2001	HOU
9	lindsma01	idbyuid	1625000	2002	HOU
10	stephga01	idbyuid	1025000	1991	SLN
11	stephga01	idbyuid	1025000	1992	SLN
12	stephga01	idbyuid	900000	1991	SLN
13	stephga01	idbyuid	900000	1992	SLN
14	stephga01	idbyuid	800000	1991	SLN
15	stephga01	idbyuid	800000	1992	SLN
16	stephga01	idbyuid	550000	1991	SLN
17	stephga01	idbyuid	550000	1992	SLN
18	lindsma01	idbyuid	410000	2001	FLO
19	lindsma01	idbyuid	410000	2002	FLO
20	lindsma01	idbyuid	395000	2001	FLO
21	lindsma01	idbyuid	395000	2002	FLO
22	lindsma01	idbyuid	380000	2001	FLO
23	lindsma01	idbyuid	380000	2002	FLO
24	stephga01	idbyuid	215000	1991	SLN
25	stephga01	idbyuid	215000	1992	SLN
26	stephga01	idbyuid	185000	1991	PHI
27	stephga01	idbyuid	185000	1992	PHI
28	stephga01	idbyuid	150000	1991	PHI
29	stephga01	idbyuid	150000	1992	PHI

GRAND QUESTION 2

This three-part question requires you to calculate batting average (number of hits divided by the number of at-bats)

(a) Write an SQL query that provides playerID, yearID, and batting average for players with at least one at bat. Sort the table from highest batting average to lowest, and show the top 5 results in your report.

To answer this i had to start by setting both h and ab as floats because it was returning integers numbers. I found this easiest to be done using ht ecast function. Then from there all i needed to do is filter out all players who did not have at least 1 at bat to get the desired results.

```
at_least_one_at_bat = pd.read_sql_query("""
SELECT playerid, yearid,
ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) AS 'batting
average'
FROM Batting
WHERE ab > 1
ORDER BY ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) DESC
LIMIT 5
""", con
)
```

playerID	yearID	batting average
oconnfr01	1893	1
brownpe01	1894	1
mcbripe01	1898	1
hopkimi01	1902	1
tonkido01	1907	1

This data is not very relevant because although these people may have a 1.0 batting average, it is only because they each have about 2 hits and 2 at bats. There is not enough data there to make this batting average indicitive of player skill.

(b) Use the same query as above, but only include players with more than 10 “at bats” that year. Print the top 5 results.

Using the code from the question above i adjsuted the where clause to instead filter out anyone without at least 10 at bats with the following code.

```
at_least_ten_at_bat = pd.read_sql_query("""
SELECT playerid, yearid,
ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) AS 'batting
average'
FROM Batting
WHERE ab > 10
ORDER BY ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) DESC
LIMIT 5
""", con
)
```

playerID	yearID	batting average

playerID	yearID	batting average
nymanny01	1974	0.643
carsoma01	2013	0.636
silvech01	1948	0.571
puccige01	1930	0.563
mccabsw01	1909	0.545

With the larger amount of at bat data to pull from, we see much more realistic data results. This is because we do not have data anomolies like in the previous question where they had a batting average of 1.0 only because they had 2 at bats and 2 hits. Here we can see from the data that of the players with at least 10 at bats, nymanny01 has the highest average.

(c) Now calculate the batting average for players over their entire careers (all years combined). Only include players with more than 100 at bats, and print the top 5 results.

GRAND QUESTION 3

Taking the previous code, i adjusted the code to instead group by the playerid. I then applied a having condition to only shows players with at least 100 total at bats.

```
group_at_least_100_at_bat = pd.read_sql_query("""
SELECT playerid, yearid,
ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) AS 'batting
average'
FROM Batting
GROUP BY playerid
HAVING ab > 100
ORDER BY ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) DESC
LIMIT 5
""", con
)
```

	playerID	yearID	batting average
0	meyerle01	1871	0.492
1	mcveyca01	1871	0.431
2	barnero01	1871	0.401
3	kingst01	1871	0.396
4	brownpe01	1882	0.378

With this much larger dataset to pull from we see that meyerle01 has the highest batting average over his whole career.

Pick any two baseball teams and compare them using a metric of your choice (average salary, home runs, number of wins, etc.). Write an SQL query to get the data you need. Use Python if additional data wrangling is needed, then make a graph in Altair to visualize the comparison. Provide the visualization and its description.

For my team comparison i choose to compare the Boston Red Socks and the New York Yankees. I did this by comparing two metrics seperatly. I started by first comparing the batting averages but as seen below, they were too close to make any conclusions about.

teamID	name	batting average
NYA	New York Yankees	0.2676
BOS	Boston Red Sox	0.268

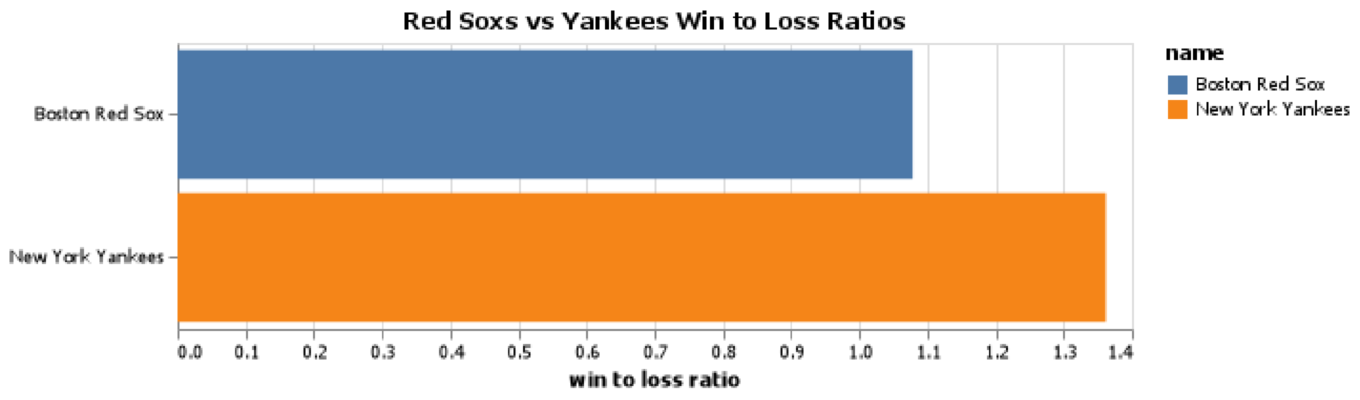
Because the numbers were too close for the batting average, i choose instead to compare the win to loss ratio for each team using the following code.

```
win_loss_team_compare = pd.read_sql_query("""
SELECT teamid, name,
ROUND((CAST (sum(w) AS FLOAT (30,4))) / (CAST (sum(l) AS FLOAT (30,4))),4) AS 'win
to loss ratio'
FROM Teams
GROUP BY name
HAVING name = 'Boston Red Sox' OR name = 'New York Yankees'
ORDER BY ROUND((CAST (sum(w) AS FLOAT (30,4))) / (CAST (sum(l) AS FLOAT
(30,4))),4)
""", con
)
```

Then i took that data and turned it into a dataframe.

	teamID	name	win to loss ratio
0	BOS	Boston Red Sox	1.0782
1	NYA	New York Yankees	1.362

I could then graph the both of them together using a bar chart to visualize the differences in win to loss ratios.



Above we can see the Boston Red Soxs with in blue compared to the New York Yankees in orange. From this data we see that although they both have similair batting averages, the New York Yankees has a significantly higher win to loss ratio.

APPENDIX A (PYTHON SCRIPT)

```
# %%
# imports libraries
import pandas as pd
import altair as alt
import numpy as np
import datadotworld as dw
import sqlite3

# %%
# imports data for baseball
sqlite_file = 'lahmansbaseballdb.sqlite'
con = sqlite3.connect(sqlite_file)

# %%
# 1 - queries database for baseball players who went to BYUI
# creates dataframe of data
baseball_byui = pd.read_sql_query("""
SELECT CollegePlaying.playerid, CollegePlaying.schoolid,
Salaries.salary, CollegePlaying.yearid, Salaries.teamid
FROM Collegeplaying
    JOIN Salaries
    ON Collegeplaying.playerid = Salaries.playerid
WHERE schoolID = 'idbyuid'
ORDER BY salary DESC
""", con
)
baseball_byui

# %%
# 1 - prints byui table to markdown
print(baseball_byui.to_markdown)

# %%
# 2-a lists players with at least one at bat, limit to 5
at_least_one_at_bat = pd.read_sql_query("""
SELECT playerid, yearid,
ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) AS 'batting
average'
FROM Batting
WHERE ab > 1
ORDER BY ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) DESC
LIMIT 5
""", con
)
at_least_one_at_bat
```

```
# %%
# 2-a prints at_least_one_at_bat to markdown
print(at_least_one_at_bat.to_markdown())

# %%
# 2-b lists players with at least ten at bat, limit to 5
at_least_ten_at_bat = pd.read_sql_query("""
SELECT playerid, yearid,
ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) AS 'batting
average'
FROM Batting
WHERE ab > 10
ORDER BY ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) DESC
LIMIT 5
""", con
)
at_least_ten_at_bat

# %%
# 2-b prints at_least_ten_at_bat to markdown
print(at_least_ten_at_bat.to_markdown())

# %%
# 2-c group by players and include only people who have at least 100 at bats
group_at_least_100_at_bat = pd.read_sql_query("""
SELECT playerid, yearid,
ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) AS 'batting
average'
FROM Batting
GROUP BY playerid
HAVING ab > 100
ORDER BY ROUND((CAST (h AS FLOAT (18,4))) / (CAST (ab AS FLOAT (18,4))),3) DESC
LIMIT 5
""", con
)
group_at_least_100_at_bat

# %%
# 2-c print group_at_least_100_at_bat to markdown
print(group_at_least_100_at_bat.to_markdown())

# %%
# 3 - creates dataframe comparing batting average for the two teams
batting_average_team_compare = pd.read_sql_query("""
SELECT teamid, name,
ROUND((CAST (sum(h) AS FLOAT (30,4))) / (CAST (sum(ab) AS FLOAT (30,4))),4) AS
'batting average'
FROM Teams
GROUP BY name
HAVING name = 'Boston Red Sox' OR name = 'New York Yankees'
ORDER BY ROUND((CAST (sum(h) AS FLOAT (30,4))) / (CAST (sum(ab) AS FLOAT
(30,4))),4)
""", con
)
```

```
batting_average_team_compare

# %%
# 3 - print batting_average_team_compare to markdown
print(batting_average_team_compare.to_markdown())

# %%
# 3 - creates dataframe comaring wins to loses
win_loss_team_compare = pd.read_sql_query("""
SELECT teamid, name,
ROUND((CAST (sum(w) AS FLOAT (30,4))) / (CAST (sum(l) AS FLOAT (30,4))),4) AS 'win
to loss ratio'
FROM Teams
GROUP BY name
HAVING name = 'Boston Red Sox' OR name = 'New York Yankees'
ORDER BY ROUND((CAST (sum(w) AS FLOAT (30,4))) / (CAST (sum(l) AS FLOAT
(30,4))),4)
""", con
)
win_loss_team_compare

# %%
# 3 - prints win_loss_team_compare to markdown
print(win_loss_team_compare.to_markdown())

# %%
# graphs win to loss ratio comparing new york yankees and boston red soxs
alt.Chart(win_loss_team_compare,
title = 'Red Soxs vs Yankees Win to Loss Ratios').encode(
    alt.X('win to loss ratio', title = 'win to loss ratio'),
    alt.Y('name', title = None),
    alt.Color('name'),
).mark_bar().properties(width=500, height=150).save('win_loss_team_compare.png')
```