Client Report - Project 3: Finding Relationships in Baseball

See code

Course DS 250

AUTHOR
Bracken Sant

Elevator pitch

In this project I utilized the sqlite3, pandas, and altair libraries to read and interpret some relationships in baseball. Some of the relationships that will be looked into are: salary, successfulness of players from BYU-Idaho, batting averages and at-bats, and home runs.

▼ Read and format project data

```
data = sql.connect('lahmansbaseballdb.sqlite')
```

Highlight the grand questions

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.

There are not that many baseball players that came from BYU-Idaho and only two that have a recorded salary in the data.

▼ table example

```
tables = pd.read_sql_query('SELECT playerID, schoolID FROM collegeplaying WHERE schoolID ==
         "idbyuid"', data)
playerList = tables['playerID'].to_list()
GQ1 = pd.read_sql_query('SELECT playerID, salary, yearID, teamID FROM salaries WHERE playerID
         IN '+str(tuple(playerList)), data)
GQ1['schoolID'] = "idbyuid"
GQ1 = GQ1.sort_values(by='salary', ascending=False)
salaries = []
for salary in GQ1['salary']:
    salaries.append("$"+"{:,}".format(salary)+"0")
GQ1['salary'] = salaries
GQ1 = GQ1.reset_index()
print(tabulate(GQ1[['playerID', 'schoolID', 'salary', 'yearID', 'teamID']], headers='keys',
         tablefmt='fancy_grid'))
```

```
tablefmt= 'fancy_grid' ))
```

	playerID	schoolID	salary	yearID	teamID
0	lindsma01	idbyuid	$4,000,000.00	2014	CHA
1	lindsma01	idbyuid	$3,600,000.00	2012	BAL
2	lindsma01	idbyuid	$2,800,000.00	2011	COL
3	lindsma01	idbyuid	$2,300,000.00	2013	CHA
4	lindsma01	idbyuid	$1,625,000.00	2010	HOU
5	stephga01	idbyuid	$1,025,000.00	2001	SLN
6	stephga01	idbyuid	$900,000.00	2002	SLN
7	stephga01	idbyuid	$800,000.00	2003	SLN
8	stephga01	idbyuid	$550,000.00	2000	SLN
9	lindsma01	idbyuid	$410,000.00	2009	FLO
10	lindsma01	idbyuid	$395,000.00	2008	FLO
11	lindsma01	idbyuid	$380,000.00	2007	FLO
12	stephga01	idbyuid	$215,000.00	1999	SLN
13	stephga01	idbyuid	$185,000.00	1998	PHI
14	stephga01	idbyuid	$150,000.00	1997	PHI

The highest salary was from lindsma01 with 4 million in a year and the lowest was from stephga01 with 150,000 in a year. The player lindsma01 started out being paid 380,000 a year and progressed in 7 years to 4 million in a year he was also traded to different teams frequently which explains the difference in salary from year to year. The player stephga01 had a fairly consistent raise in salary and also mainly played for the same team SLN.

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 1 at bat that year. Sort the table from highest batting average to lowest, and then by playerid

alphabetically. Show the top 5 results in your report.

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

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

As the requirements for at-bats increased the batting average decreased, this is due to more chances to miss the ball and, therefore, get a smaller amount of hits compared to at-bats.

▼ table example

```
# Part a of Grand Question 2
bat_av_1 = pd.read_sql_query('SELECT playerID, yearID, H, AB FROM batting GROUP BY playerID
         HAVING AB > 0', data)
bat_av_1['bat_av'] = bat_av_1['H'] / bat_av_1['AB']
bat_av_1 = bat_av_1.sort_values(by='bat_av', ascending=False)
bat_av_1 = bat_av_1.reset_index()
print(tabulate(bat_av_1[['playerID', 'yearID', 'bat_av', 'H', 'AB']].head(), headers='keys',
         tablefmt='fancy_grid'))
```

	playerID	yearID	bat_av	H	AB
0	tupmama01	2008	1	1	1
1	cotteho01	1922	1	1	1
2	holloji01	1929	1	1	1
3	bruneju01	2000	1	1	1
4	egeco01	2016	1	1	1

As you can see here the 5 highest were all 100% because if they didn't hit the ball it would then be 0%.

▼ table example

```
# Part b of Grand Question 2
bat_av_10 = pd.read_sql_query('SELECT playerID, yearID, H, AB FROM batting GROUP BY playerID
         HAVING AB > 10', data)
bat_av_10['bat_av'] = bat_av_10['H'] / bat_av_10['AB']
bat_av_10 = bat_av_10.sort_values(by='bat_av', ascending=False)
bat_av_10 = bat_av_10.reset_index()
print(tabulate(bat_av_10[['playerID', 'yearID', 'bat_av', 'H', 'AB']].head(), headers='keys',
         tablefmt='fancy_grid'))
```

	playerID	yearID	bat_av	H	AB
0	nymanny01	1974	0.642857	9	14
1	silvech01	1948	0.571429	8	14
2	puccige01	1930	0.5625	9	16
3	ariasjo01	2006	0.545455	6	11
4	skafffr01	1935	0.545455	6	11

As you can see from the table, the highest batting averages decreased compared to the batters with the requirement of 1 at-bat.

▼ table example

```
# Part c of Grand Question 3
bat_av_100 = pd.read_sql_query('SELECT playerID, yearID, H, AB FROM batting GROUP BY playerID
        HAVING AB > 100', data)
bat_av_100['bat_av'] = bat_av_100['H'] / bat_av_100['AB']
bat_av_100 = bat_av_100.sort_values(by='bat_av', ascending=False)
bat_av_100 = bat_av_100.reset_index()
print(tabulate(bat_av_100[['playerID', 'yearID', 'bat_av', 'H', 'AB']].head(), headers='keys',
        tablefmt='fancy_grid'))
```

	playerID	yearID	bat_av	H	AB
0	meyerle01	1871	0.492308	64	130
1	mcveyca01	1871	0.431373	66	153
2	barnero01	1871	0.401274	63	157
3	kingst01	1871	0.395833	57	144
4	brownpe01	1882	0.378472	109	288

As you can see from the table, the highest batting averages decreased compared to the batters with the requirement of 10 at-bat.

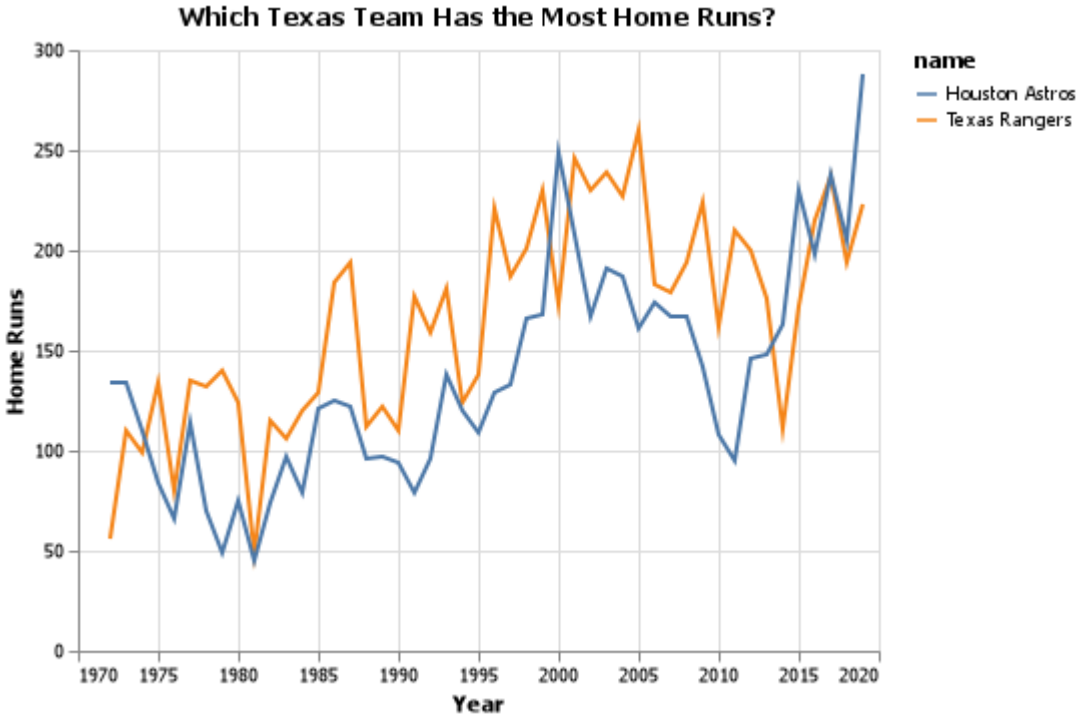
GRAND QUESTION 3

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, then make a graph in Altair to visualize the comparison. What do you learn?

For my teams I chose the Texas Rangers and the Houston Astros and chose home runs as the metric of comparison. In the graph you cannot only see the differences in home runs between the Texas Rangers and Houston Astros but there is also an evident increase in home runs throughout the years most likely due to increase in training and technology. The Texas Rangers generally have more home runs than the Houston Astros but there were a couple spikes in the data from the Houston Astros that caused them to overtake the Texas Rangers in the early 1970's, around 2000, and in more recent years from 2015 to about 2019.

▼ graph example

```
teams = pd.read_sql_query("SELECT name, HR, yearID FROM teams WHERE (name == 'Houston Astros'
        or name == 'Texas Rangers') and yearID > 1971", data)
teams["Year"] = teams["yearID"]
teams["Home Runs"] = teams["HR"]
chart = al.Chart(teams).encode(x = al.X("Year", axis = al.Axis(format = "d")), y = "Home Runs",
        color = "name").mark_line().properties(title = "Which Texas Team Has the Most Home
        Runs?")
```



Home Runs

As you can see in the figure there is a positve trend and the Texas Rangers are above the Houston Astros in home runs in most years besides the early 1970's, around 2000, and between the years 2015 and 2019.