Data Wrangling for Pitcher and Hitter Datasets

For this project I decided to create two separate dataframes, one consisting of pitchers and another for hitters/position players. This was due to the fact that pitchers and hitters each have their own set of recorded metrics. For example, a pitcher's measure of success is heavily weighted towards statistics such as 'W' Wins per season, 'K' Strikeouts per season, and 'ERA' Earned Run Average. Hitters are mostly focused on 'H' Hits per season, 'HR' Home Runs per season, and 'BAVG' Batting Average.

One of the first steps I took towards cleaning my dataset was searching for any missing values. During my search I found a significant amount of null values in both of my datasets. The pitcher dataset had 16% of salaries missing and the hitter dataset had 13%. This was significant because all of the missing values were located in the target variable 'Salary'. After some research i decided not to drop the missing values but to instead fill them. I filled each missing value with the minimum salary for that respective year. Due to the fact that the MLB has been increasing their minimum salary throughout the years, I figured I couldn't just fill the nulls with one single value.

In order to fill in the null values I created a dictionary with years as the key and minimum salary for that year as a value. I then created a new column called 'salary_filled', by using the map function to iterate through each row and fill any missing values with the salary dictionary. The code sample can be seen below.

```
#checking for percentage of missing data
df_pitcher.isnull().sum()/len(df_pitcher)

#replacing salary null values with league minimum for respective years
salary_dict = {2019:555000, 2018:545000, 2017:535000, 2016:507500,
2015:507500, 2014:480000,
               2013:480000, 2012:480000, 2011:414000, 2010:400000,
2009:400000, 2008:390000,
               2007:380000, 2006:327000, 2005:316000, 2004:300000,
2003:300000, 2002:300000,
               2001:200000, 2000:200000, 1999:109000, 1998:109000,
1997:109000, 1996:109000,
               1995:109000, 1994:100000, 1993:100000, 1992:100000,
1991:100000, 1990:100000}

df_pitcher['salary_filled'] =
df_pitcher['salary'].fillna(df_pitcher['year'].map(salary_dict))
```

After dealing with all the missing values and rearranging columns, I started to do some feature engineering. I needed to figure out how many years each player had been in the MLB. Both datasets were unorganized and only included players by groups for each season. Implementing this feature in each data set will help with organization and filtering going forward.

In order to create this new feature I sorted the entire dataset by columns 'playerName' and 'year'. Next, I used the Pandas groupby function to split my dataset into groups for each player, and chained it with a cumulative count method. The new feature was called 'total_years_mlb', and returned the number of years in the MLB for each player . I also added another feature which returned the minimum salary for each year in 'total_years_column'. This was built by using the apply method with a lambda function throughout each row in the 'year' column. The addition of these new features will help me gain a more valuable insight into my data. The code can be seen below.

```
#add column with number of year in MLB
df_pitcher_final = df_pitcher.sort_values(by=['playerName', 'year'])
df_pitcher_final['year'] = df_pitcher_final['year'].astype(str)
df_pitcher_final['total_years_mlb'] =
df_pitcher_final.groupby('playerName').cumcount()+1

#add earned minimum salary for player year in MLB
df_pitcher_final['minimum_year'] = df_pitcher_final['year'].apply(lambda x:
salary_dict[int(x)])
```

Another issue that had to be accounted for was monetary inflation. The salaries paid toward athletes in the 1990's was very different than that paid today. In order to account for inflation I installed a python library called CPI. CPI is a library that adjusts U. S. dollars by the current consumer price index. A combination of the apply method and a lambda function was used by passing the salary and year as arguments. The code can be seen below.

```
#adjust salary for inflation
df_pitcher['adj_salary'] = df_pitcher.apply(lambda x: cpi.inflate(x.salary,
x.year), axis=1)
df_pitcher['adj_salary_filled'] = df_pitcher.apply(lambda x:
cpi.inflate(x.salary_filled, x.year), axis=1)
df_pitcher['adj_salary'] = df_pitcher['adj_salary'].round(1)
df_pitcher['adj_salary_filled'] = df_pitcher['adj_salary_filled'].round(1)
```

After looking through both datasets I did find a couple of data points that would be considered ‘outliers’. I ultimately decided to leave the outliers within my data for a couple of reasons. The first reason is because the salaries paid towards athletes are always going to increase year by year. For example, throughout the last 20 years the minimum salary for MLB players has increased from $200,000 to $555,000. So, what might be considered an outlier today might not be in a couple of years. The final step I took during my data wrangling process was subsetting the data and using players only after the 2010 season. MLB is an expanding and ever changing sport and a lot has happened over the last 10 years. Some minor and major changes, such as including new rules.