ABSTRACT

This paper is a description of the methods used in gaining useful information from large databases of baseball statistics. Baseball lends itself to accumulating vast quantities of information, leading to much useful data that can be ascertained and collected. This paper will specifically cover creating a model that attempts to predict the performance of teams in the playoffs. Given a set of statistics for a complete year of play, the model should be able to make a rough prediction about the winners of playoff series’, and who will ultimately win the World Series.

1. INTRODUCTION

Baseball of the American variety is distinctive among sports of the world in that every play of every professional game has been recorded for over a century. Further, every pitch thrown in a professional game has been detailed since roughly 1965, resulting in a huge amount of data that has yet to be fully explored [1].

So far, only one study has been done in this area that I have yet to find, and that study focused nearly entirely on pitching, specifically analyzing a pitcher’s performance during a game. In that study, the question was raised as to when a baseball manager might make the decision to retire a pitcher and replace him with a relief pitcher. Considering that a warmup period of roughly 10 minutes is required to allow a pitcher enough time to effectively throw the ball, and also warming up too much might cause him to be fatigued and ineffective, there is a point where the decision to allow a substitute can be made most optimally.

A very important question (perhaps the most important question) can be raised in regards to baseball statistics: is it possible to accurately predict the winners of the playoffs given trends evident in the regular season? This question is not easily answered, and as of yet no model exists that accomplishes this task. My approach to this problem was to focus almost entirely on offensive statistics, largely ignoring pitching and defense. The rationale behind this approach is two-fold. First, if we consider the statistics of the entire league instead of individual teams, the defensive performance of some teams should tend to average out with the offensive performance of other teams. For instance, a team that has excellent pitching might face a team with top-notch batters. This is a wash, and the score should reflect both of these facts. Secondly, scoring in general is closely tied to defensive performance of the opposing team, which means overall offensive performance takes pitching and fielding into account at some level. The most accurate way of accomplishing the task of an accurate predictive model would be, of course, to consider all factors separately and then create some overlying method of producing results, but that is far beyond the scope of what can legitimately be done without a lot more research. What follows in this paper is instead a “best guess” model that tries to be accurate but also reasonable. Although the focus was almost entirely on offensive production, Fielding Average was used in the final calculations as a means of smoothing out the results.

2. DATA COLLECTION AND PREPROCESSING

Thankfully, as detailed as baseball is, there are several avenues for gaining useful information about the games, although the format of the information is not very structured. The Retrosheet Organization, a purely volunteer organization, has collected and posted complete game statistics for every Major League game played from 1965 until 2004. The statistics themselves are merely ASCII test files with vast amounts of data; not very easy to comprehend nor to extract meaningful information. These files, denoted EVN, are structured in the following format. First, there is a date field which signifies the date the game was played upon. Further fields indicate day of week, if the game is a single game or a double header, home and away team league and name, scores of each team, park information, attendance, the visiting team’s offensive
statistics, the home team's offensive statistics, pitching stats, names of the umpires, winning a losing pitchers names, starting lineups, and various other minutiae relating to the game. Each piece of information is separated from the previous entry with a comma. Some examples of the raw data follow:

"20000329", "0", "Wed", "CHN", "NL", 1, "NYN", "NL", 1

This is the beginning of the entry. We see that the game occurred on March 29th, 2000, which was a Wednesday, and the Chicago Cubs, a National League team, played the New York Mets, another National League team. It was a single game (the “0”) and it was the 1st game for each team.

Obviously, there is a huge amount of data stored within the text files, as the explanation for the meanings of just the very beginning of an entry are quite meaningful. Overall there are 161 fields for each entry, although some of the fields are NULL, as no meaningful value can be stored for those fields except in the most strange of circumstances, such as the home team batting first or if a game was protested. These occurrences might happen, however, so the fields are required for sake of completeness.

Given the very rigid structure of the ASCII files, they can be stored in a database with a bit of manipulation. I chose to use Microsoft Access simply because of the ease in which to transition comma-delimited text files into a new database. First, a new database was created within Access; I called it simply Baseball. Then, the option to “Get External Data” was selected, using the option “Import”. The text file containing the statistics was chosen from the available files. For this experiment, I chose years 2000 to 2004, as these five years were recent enough to be meaningful and also a short enough time span to be reasonably analyzed. Less than five years would not, in my opinion, be an accurate judge of the model, and longer than five years would become too unwieldy for such a large dataset.

The “Import Text Wizard” recommended that the data was “delimited” (it was), so this option was selected, with the comma being the delimiter. Access then asked what names to assign to the fields. Retrosheet has a webpage that extensively describes the purpose of each field; I used this information to name each field something meaningful. Generally, the name of each field was something very easy to comprehend, such as Date or NumGame. Even after naming nearly all the fields, the sheer magnitude of the information was such that a written record of the naming scheme was printed out to aid in the correlation of data. Further, the default Access name (Field27 for instance) was kept in some circumstances, namely for fields that were not useful to my analysis and would mostly likely never be referenced.

Access then created a new field called ID, which was intended to be utilized for the primary key, i.e. to ease in the search for information. The data was now dumped into a new table and available to use as part of the database.

The fields, as indicated earlier, were named using a basic scheme that lends itself to ease of searching. Here is a list of the most relevant fields used, the V being replaced by an H in the instance of a Home team instead of a Visiting team:

[V/H]atbat – Visitor’s number of at-bats
[V/H]hits – Total hits
[V/H]doubles – Total number of doubles
[V/H]triples – Number of triples
[V/H]homeruns – Number of homers
[V/H]rbi – Number of RBIs
[V/H]sachits – Sacrifice hits
[V/H] sacflies – Sacrifice flies
[V/H]hitbypitch – Batters hit by pitches
[V/H]walks – Number of walks
[V/H]intentionalwalks – Batters intentionally walked
[V/H] strikeouts – Number of strikeouts
[V/H]stolenbases – Bases stolen
[V/H] caughtstealing – Runners picked off
[V/H]grounddouble – Infield ground rule doubles
[V/H] catcherinterference – Bases granted by interference from the catcher
[V/H] leftonbase – Runners left on
[V/H]pitchersused – Total number of pitchers
[V/H]indearnedruns – Individual earned runs
[V/H]teamearnedruns – Team earned runs
[V/H]wildpitches – Wild pitches
[V/H] balks – Number of balks
[V/H]putouts – Putouts
[V/H]assists – Number of assists
[V/H] errors – Total errors
After the data has been put into the database properly, it is very easy to manipulate it as soon as a particular piece of information is requested.

The SQL facilities utilized in Access, however, are not quite as detailed as is necessary for this study. MSSQL server offers greater capabilities, but is not as easily configured. To alleviate this problem, a simple PHP front-end was developed. PHP is a web-based scripting language that allows querying databases very easily. All the algorithms presented here were translated into PHP, with the resulting query displayed through a web browser. This allowed for rapid querying without much need for complicated programming. See the attached sheets for a complete listing of the scripts.

3. ISSUES ENCOUNTERED

The EVN files are huge. Uncompressed, the data just from the year 2000 is 2.5 megabytes in size, with a total number of 2429 total games indexed. This may not sound like a large number of records, but if we consider that each record stores the results of one game played by two teams, both offensive and defensive statistics, the amount of information is very impressive. For instance, running a simple query on the year 2000 database shows that the Florida Marlins played a total of 161 games. To find out even the simplest offensive statistic for the Marlins requires a lot of computational time. Further, developing complex algorithms for discovering this information is not an easy task. As such, I mostly focused on the “basics”, i.e. those methods already existing in baseball and decided to attempt to put together a reasonable strategy of queries later.

The total number of records is calculated as follows:

12142 total games listed for 2000 – 2005
161 fields of information
1954862 unique pieces of information for the whole database

The database itself turned out to be 24.3 MB, with the scripts that manipulated the data having 546 lines of code. There were several sets of intermediary data, but the final results were stored in a simple spreadsheet that was 33 KB in size and consisted of 150 individual fields; 30 teams with 5 Aggregate Values for each team, one for each year.

This should be a good indication that the dataset was reasonably large, and generating the required statistics was computationally expensive.

4. ALGORITHMS

As stated earlier, the only other study of a similar nature was performed on pitching statistics. The author of that study had a fairly complex algorithm for speculating on the performance of a pitcher over time. At first I thought of duplicating this work, but decided to attempt to create a predictive model for overall performance instead of merely individual performance. As such, I focused primarily on some of the commonly used baseball statistics as a starting point. These include:

**Base-on-balls Percentage** = (total walks) / (plate appearances)

**Batting Average** = (total hits) / (official at-bats)

**Home Run Ratio** = (at-bats) / (home runs)

**On-base Percentage** = (hits + walks + hits by pitch) / (at-bats + walks + hits by pitch + sacrifice flies)

**Slugging Average** = (total bases) / (at-bats)

**Strikeout Ratio** = (at-bats) / (strikeouts)

**On Base Slugging** = (on base percentage) + (slugging average)

**Winning Percentage** = (wins) / (wins + losses)

**Adjusted Production** = (on base percentage / League OBP) + (slugging average / League SA) – 1
Fielding Average – \( \frac{\text{put outs} + \text{assists}}{\text{put outs} + \text{assists} + \text{errors}} \)

As stated earlier, Fielding Average was included as a means to smooth the results. If we focus entirely on offensive statistics, it is possible to miss key elements of pitching and defense that may not be accurately represented. The inclusion of FA allows us to take into account the very likely case that good pitching and defense might nullify good offense, hence we are not penalizing a team’s offensive capabilities for facing a team that has excellent fielders.

The most relevant statistic is Adjusted Production, or APRO. APRO is a composite score that combines the most relevant offensive statistics and normalizes these statistics across the league. This normalization is intended to allow players from different eras to be compared to each other. This statistic is extremely useful for this experiment, namely for the reason that one composite score is much easier to compare than several scores. Also, since it essentially combines all over the previously mentioned offensive statistics, nothing is “left out” with regards to offense.

In all cases, team performance was judged, not individual performance. This is because while a team may have a great hitter, the overall performance of the team can only be altered slightly unless there are many great hitters. Further, since this model is an attempt to predict final team standings instead of individual standings, such as number of homers or batting averages, there as no real need to do those calculations.

Using the above formulas, I devised my own formula that should reflect an overall team performance metric and allow the comparison of teams at the end of the year and make an accurate guess as to their eventual playoff capabilities.

Calculated Aggregate Value – Adjusted Production * Winning Percentage + Fielding Average * Winning Percentage

This formula takes the APRO value and the Fielding Average and multiplies them by the team Winning Percentage then adds their values to create an aggregate score. The rationale behind this method is as follows. Given that APRO is an overall offensive statistic, and some teams exist that can have excellent offensive production and lousy pitching and defense, the multiplication by Winning Percentage penalizes those teams that have good batting but never win and rewards teams that may not be the best offensively but win games through defense. This is another attempt to account for the lack of pitching statistics included in my analysis, but accounting for the actual ability to win games most likely is a better way to predict future success than any other derived statistics, hence the inclusion it. Smoothing out Fielding Average in much the same way accounts for teams with great defense who cannot produce offensively.

The end result is a single number that should be useful in comparing two or more teams and determining which would win a game or a series of games.

5. RESULTS

First, queries were run on all 30 teams, generating overall team statistics. For each team, the APRO, Fielding Average and Winning Percentage was calculated. The other statistics referred to earlier were also generated, but were not used in the final analysis so they will be ignored for this report. APRO does include most of these statistics, however, so it was necessary to calculate them during the preliminary runs. After calculating these statistics, the CAV was then calculated. All of this information was presented in a web browser. The attached code sheets have full explanations of the method used in the queries, although a snippet is presented below for demonstrative purposes.

```sql
$atbatV = "SELECT Vatbat FROM $YEAR WHERE Visitor = '$TEAM';
$atbatH = "SELECT Hatbat FROM $YEAR WHERE Home = '$TEAM';
$QatbatV = odbc_do($connectionstring, $atbatV);
$QatbatH = odbc_do($connectionstring, $atbatH);

$Vtotalatbats = 0;
$Htotalatbats = 0;
while(odbc_fetch_row($QatbatV))
{
    $Vtotalatbats += odbc_result($QatbatV, "Vatbat");
}
while(odbc_fetch_row($QatbatH))
{
```
In the above code, a query is constructed for $QatbatV, which indicates the query will be used for the number of at-bats performed when $TEAM (each team in the league) was up to bat and was the visiting team. The $atbatH would be for when that same team was at home. The query is then run and stored in the $QatbatV or $QatbatH value. $Vtotalatbats will be the final number of at-bats as visitor, and this value is incremented using the run query. At the end of the sample code presented here, we have two calculated values, total at-bats for a team, both home and away. We can add these two values to get the total number of at-bats attempted during the year.

See Figure 1 below for reference of the output as presented by the web browser after running the queries.

The results were stored according to league, i.e. National or America, and then graphs were created that displayed the CAV by year for each team. This allowed for ease of visually determining the dominant teams for any particular year and also showed overall performance of teams over the five year period.

The team identifiers and their names are listed below.

<table>
<thead>
<tr>
<th>Team</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANA</td>
<td>Anaheim Angels</td>
</tr>
<tr>
<td>ARI</td>
<td>Arizona Diamondbacks</td>
</tr>
<tr>
<td>ATL</td>
<td>Atlanta Braves</td>
</tr>
<tr>
<td>BAL</td>
<td>Baltimore Orioles</td>
</tr>
<tr>
<td>BOS</td>
<td>Boston Red Sox</td>
</tr>
<tr>
<td>CHA</td>
<td>Chicago Cubs</td>
</tr>
<tr>
<td>CHN</td>
<td>Chicago White Sox</td>
</tr>
<tr>
<td>CIN</td>
<td>Cincinnati Reds</td>
</tr>
<tr>
<td>CLE</td>
<td>Cleveland Indians</td>
</tr>
<tr>
<td>COL</td>
<td>Colorado Rockies</td>
</tr>
<tr>
<td>DET</td>
<td>Detroit Tigers</td>
</tr>
<tr>
<td>FLO</td>
<td>Florida Marlins</td>
</tr>
<tr>
<td>HOU</td>
<td>Houston Astros</td>
</tr>
<tr>
<td>KCA</td>
<td>Kansas City Royals</td>
</tr>
<tr>
<td>LAN</td>
<td>Los Angeles Dodgers</td>
</tr>
<tr>
<td>MIL</td>
<td>Milwaukee Brewers</td>
</tr>
<tr>
<td>MIN</td>
<td>Minnesota Twins</td>
</tr>
<tr>
<td>MON</td>
<td>Montreal Expos</td>
</tr>
<tr>
<td>NNY</td>
<td>New York Mets</td>
</tr>
<tr>
<td>NCA</td>
<td>New York Yankees</td>
</tr>
<tr>
<td>OAK</td>
<td>Oakland Athletics</td>
</tr>
<tr>
<td>PHI</td>
<td>Philadelphia Phillies</td>
</tr>
</tbody>
</table>
The final results are represented in figures 3 and 4 on the attached sheet.

6. ANALYSIS

The first thing that is evident from the results is that some teams are clearly better than others. This can be supported by a quick look at their overall records. Since winning percentage is factored into the CAV result, the graph is a fair representation of two things: standings and ability to win games. To use a quick example, we can see from Figure 3 that the Seattle Mariners were very strong in 2001. This can be extrapolated out to say that the Mariners would most likely beat Baltimore, as their CAV is close to doubled. The records for 2001 show that Seattle beat Baltimore 8 out of 9 times during the season, giving weight to the CAV as a valid comparative value.

Figure 5 is a list of the playoff teams and their records for 2000 to 2004.

For each year, the four playoff teams from each league as well as the World Series results are shown. In 2000, for instance, the Yankees played the A’s and won the series in 3 games, then played the Mariners in the ALCS and won 4 games to 2. The Yankees then went on to beat the Mets 4 games to 1 in the World Series.

Figure 6 is a chart featuring the teams from each league by year along with their CAV values. Using the above concrete data, i.e. actual results of the season, and the Calculated Aggregate Value that was generated by the formulas given above, we arrive at the graph presented in Figure 6.

Comparing the results from Figure 5 and 6, we can see a trend. In 2000, the Yankees won the World Series with a CAV rating of 1.1264, beating, 4th best in the American League behind Oakland, Cleveland and the White Sox. Cleveland did not make the playoffs, Seattle did instead, but this was probably because of a weak division compared to a strong division (only 1 team from each division makes the playoffs, with 1 additional Wild Card team pulled from the strongest opponent in the whole of the league). In other words, Seattle was most likely in an easy division, winning it and thereby knocking out Cleveland from the playoffs. In the National League for 2000, the Giants, Cardinals, Rockies and Braves were the four strongest teams per CAV, with the eventual World Series representative Mets placing 5th overall. Again, the Rockies most likely were displaced by the Mets due to divisional strengths. In 2000, it appears that the CAV value does not do a very good job at predicting overall end standings, although if we compare the World Series results, Yankees vs. Mets, we see that the Yankees should win, having a CAV of 1.1264 compared to 1.1086 for the Mets. This is, in fact, what occurred. Looking at other playoff results, the Yankees should beat the Mariners (this happened), the Cards should overtake the Braves. Some of the other results are not predicted by comparing CAV in 2000, but again this may be the result of some divisional strengths.

Going to the 2001 season, we see that Seattle is the strongest team by far, yet they lost to the Yankees in the ALCS. This could have been simply a fluke, or a weak division, allowing the Mariners to pump up their stats. The World Series was played by the 3rd ranked team per CAV in each division, with the Diamondbacks beating the Yankees in 7 games, having a slightly smaller CAV value.

In 2002, the Angels, 3rd in American League CAV faced the Giants, rated 3rd in CAV in the National League. The Angels had a substantial CAV lead over the Giants, and were ultimately victorious. 2002’s results much more closely mirrored the results that the CAV would predict. 2003’s actual results were contradictory with the CAV values calculated, although both World Series representatives were in the top 3 of their respective leagues.

2004’s playoff results were nearly right on using CAV values, with the only anomaly being the win by the Astros over the Braves. Again, knowing the division Braves play in, this was most likely the result of a fairly weak schedule. This could probably be shown using a game-by-game breakdown instead of merely a single predictive result.

7. CONCLUSIONS
The CAV value calculated by the algorithm I presented works reasonably well from a realistic standpoint. In absolute terms, the results of each series of games was predicted nearly completely correctly in only one year, 2004, although the results were not bad in the other 4 years. The problems that came up mostly had to do with how fine the calculated values were taken. In nearly every case, my own knowledge of baseball allowed me to justify the conclusions that the CAV would make one reach, even in cases where its prediction was wrong. I believe this could be corrected by refining the algorithm the following ways.

First, instead of merely looking at a season in its entirety, look at each team’s CAV compared to divisional opponents, then league opponents. Then, re-weight the CAV based on how strong or weak a particular division is. The performance of a team in a weak division should be decreased, and the performance of a team in a strong division increased. This will still result in some teams that do not make the playoffs having a higher CAV than teams that do make the playoffs, but this is unavoidable given the nature of the playoff wildcard system. This newly weighted CAV could then be re-weighed for the World Series by comparing each league in the same way. Essentially, each team would require three CAV values: divisional value, league value, and overall value, which would be the value reflected if all 30 teams were considered as a whole.

Secondly, include pitching. I stated it more than once in this paper, but the inclusion of pitching statistics would most likely make the CAV much more useful. Pitching was not included mainly due to the fact that pitching statistics are incredibly unwieldy and far more complex to calculate than offensive stats. Further, individual pitchers can influence a game in a far greater capacity than a single batter, which means each game must be weighted using a pitcher’s overall performance. This level of complexity could be added to the current system I have created thus far with a lot of work, but without much complication.

Overall, I think the CAV value would be useful for predicting final standings if enough data was provided. For instance, if 80 games had been played, or half a season, I think one could make a reasonable guess as to the final standings based on the CAV value.

8. FUTURE RESEARCH

There is a huge amount of future research that can be done in this area. The refinement of the CAV would allow for a more realistic modeling system of baseball game outcomes, and could be a useful stat for comparing strengths of teams by year. Also, similar algorithms might be developed for other sports, including basketball and football.

9. REFERENCES


<table>
<thead>
<tr>
<th></th>
<th>American</th>
<th>National</th>
<th>World Series</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>NYA 3 - OAK 2</td>
<td>STL 3 - ATL 0</td>
<td>NYA 4 - NYN 1</td>
</tr>
<tr>
<td></td>
<td>SEA 3 - CHA 0</td>
<td>NYN 3 - SFN 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NYA 4 - SEA 2</td>
<td>NYN 4 - STL 1</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>NYA - OAK 2</td>
<td>ARI 3 - STL 2</td>
<td>ARI 4 - NYA 3</td>
</tr>
<tr>
<td></td>
<td>SEA 3 - CLE 2</td>
<td>ATL 3 - HOU 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NYA 4 - SEA 1</td>
<td>ARI 4 - ATL 1</td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>MIN 3 - OAK 2</td>
<td>SLN 3 - ARI 0</td>
<td>ANA 4 - SFN 3</td>
</tr>
<tr>
<td></td>
<td>ANA 3 - NYA 1</td>
<td>SFN 3 - ATL 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ANA 4 - MIN 1</td>
<td>SFN 4 - SLN 1</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>BOS 3 - OAK 2</td>
<td>CHN 3 - ATL 2</td>
<td>FLO 4 - NYA 2</td>
</tr>
<tr>
<td></td>
<td>NYA 3 - MIN 1</td>
<td>FLO 3 - SFN 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NYA 4 - BOS 3</td>
<td>FLO 4 - CHN 3</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>BOS 3 - ANA 0</td>
<td>HOU 3 - ATL 2</td>
<td>BOS 4 - SLN 0</td>
</tr>
<tr>
<td></td>
<td>NYA 3 - MIN 1</td>
<td>SLN 3 - LAN 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BOS 4 - NYA 3</td>
<td>SLN 4 - HOU 3</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5**

<table>
<thead>
<tr>
<th></th>
<th>American</th>
<th>National</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>CHA</td>
<td>SFN</td>
</tr>
<tr>
<td></td>
<td>1.279</td>
<td>1.2463</td>
</tr>
<tr>
<td></td>
<td>CLE</td>
<td>SLN</td>
</tr>
<tr>
<td></td>
<td>1.2101</td>
<td>1.1971</td>
</tr>
<tr>
<td></td>
<td>OAK</td>
<td>COL</td>
</tr>
<tr>
<td></td>
<td>1.1278</td>
<td>1.1913</td>
</tr>
<tr>
<td></td>
<td>NYA</td>
<td>ATL</td>
</tr>
<tr>
<td></td>
<td>1.1264</td>
<td>1.1613</td>
</tr>
<tr>
<td></td>
<td>TOR</td>
<td>NYN</td>
</tr>
<tr>
<td></td>
<td>1.1038</td>
<td>1.1086</td>
</tr>
<tr>
<td></td>
<td>ANA</td>
<td>CIN</td>
</tr>
<tr>
<td></td>
<td>1.0818</td>
<td>1.0773</td>
</tr>
<tr>
<td></td>
<td>SEA</td>
<td>ARI</td>
</tr>
<tr>
<td></td>
<td>1.0496</td>
<td>1.0535</td>
</tr>
<tr>
<td></td>
<td>BOS</td>
<td>LAN</td>
</tr>
<tr>
<td></td>
<td>1.0148</td>
<td>0.9996</td>
</tr>
<tr>
<td></td>
<td>KCA</td>
<td>HOU</td>
</tr>
<tr>
<td></td>
<td>1.0092</td>
<td>0.9637</td>
</tr>
<tr>
<td></td>
<td>DET</td>
<td>FLO</td>
</tr>
<tr>
<td></td>
<td>0.9743</td>
<td>0.9286</td>
</tr>
<tr>
<td></td>
<td>TEX</td>
<td>SDN</td>
</tr>
<tr>
<td></td>
<td>0.9374</td>
<td>0.8501</td>
</tr>
<tr>
<td></td>
<td>BAL</td>
<td>PIT</td>
</tr>
<tr>
<td></td>
<td>0.8946</td>
<td>0.8395</td>
</tr>
<tr>
<td></td>
<td>MIN</td>
<td>MON</td>
</tr>
<tr>
<td></td>
<td>0.8403</td>
<td>0.8174</td>
</tr>
<tr>
<td></td>
<td>TBA</td>
<td>MIL</td>
</tr>
<tr>
<td></td>
<td>0.7938</td>
<td>0.7935</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.7487</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.745</td>
</tr>
</tbody>
</table>

**2001**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SEA</td>
<td>HOU</td>
<td>1.218</td>
</tr>
<tr>
<td></td>
<td>OAK</td>
<td>SLN</td>
<td>1.2019</td>
</tr>
<tr>
<td></td>
<td>NYA</td>
<td>ARI</td>
<td>1.1721</td>
</tr>
<tr>
<td></td>
<td>CLE</td>
<td>SFN</td>
<td>1.0988</td>
</tr>
<tr>
<td></td>
<td>MIN</td>
<td>COL</td>
<td>1.0952</td>
</tr>
<tr>
<td></td>
<td>CHA</td>
<td>CHN</td>
<td>1.0709</td>
</tr>
<tr>
<td></td>
<td>BOS</td>
<td>ATL</td>
<td>1.0524</td>
</tr>
<tr>
<td>Year</td>
<td>Team</td>
<td>Opponent</td>
<td>Number</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>2002</td>
<td>NYA</td>
<td>ARI</td>
<td>1.259</td>
</tr>
<tr>
<td></td>
<td>OAK</td>
<td>ATL</td>
<td>1.2335</td>
</tr>
<tr>
<td></td>
<td>ANA</td>
<td>SLN</td>
<td>1.1614</td>
</tr>
<tr>
<td></td>
<td>MIN</td>
<td>SFN</td>
<td>1.1524</td>
</tr>
<tr>
<td></td>
<td>BOS</td>
<td>LAN</td>
<td>1.1029</td>
</tr>
<tr>
<td></td>
<td>SEA</td>
<td>HOU</td>
<td>1.0818</td>
</tr>
<tr>
<td></td>
<td>CHA</td>
<td>MON</td>
<td>1.0445</td>
</tr>
<tr>
<td></td>
<td>TOR</td>
<td>COL</td>
<td>1.0202</td>
</tr>
<tr>
<td></td>
<td>TEX</td>
<td>FLO</td>
<td>0.9629</td>
</tr>
<tr>
<td></td>
<td>CLE</td>
<td>PHI</td>
<td>0.9515</td>
</tr>
<tr>
<td></td>
<td>BAL</td>
<td>CIN</td>
<td>0.9512</td>
</tr>
<tr>
<td></td>
<td>KCA</td>
<td>NYN</td>
<td>0.8873</td>
</tr>
<tr>
<td></td>
<td>TBA</td>
<td>PIT</td>
<td>0.8133</td>
</tr>
<tr>
<td></td>
<td>DET</td>
<td>CHN</td>
<td>0.7779</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDN</td>
<td>0.7594</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MIL</td>
<td>0.6509</td>
</tr>
<tr>
<td>2003</td>
<td>BOS</td>
<td>ATL</td>
<td>1.346</td>
</tr>
<tr>
<td></td>
<td>NYA</td>
<td>SFN</td>
<td>1.2513</td>
</tr>
<tr>
<td></td>
<td>MIN</td>
<td>FLO</td>
<td>1.1148</td>
</tr>
<tr>
<td></td>
<td>OAK</td>
<td>SLN</td>
<td>1.1148</td>
</tr>
<tr>
<td></td>
<td>TOR</td>
<td>HOU</td>
<td>1.0949</td>
</tr>
<tr>
<td></td>
<td>SEA</td>
<td>CHN</td>
<td>1.0656</td>
</tr>
<tr>
<td></td>
<td>CHA</td>
<td>ARI</td>
<td>1.0569</td>
</tr>
<tr>
<td></td>
<td>KCA</td>
<td>PHI</td>
<td>1.0472</td>
</tr>
<tr>
<td></td>
<td>TEX</td>
<td>MON</td>
<td>1.0381</td>
</tr>
<tr>
<td></td>
<td>ANA</td>
<td>COL</td>
<td>1.004</td>
</tr>
<tr>
<td></td>
<td>BAL</td>
<td>PIT</td>
<td>0.9502</td>
</tr>
<tr>
<td></td>
<td>CLE</td>
<td>LAN</td>
<td>0.9458</td>
</tr>
<tr>
<td></td>
<td>TBA</td>
<td>MIL</td>
<td>0.838</td>
</tr>
<tr>
<td></td>
<td>DET</td>
<td>NYN</td>
<td>0.8069</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDN</td>
<td>0.7367</td>
</tr>
<tr>
<td>2004</td>
<td>BOS</td>
<td>SLN</td>
<td>1.3487</td>
</tr>
<tr>
<td></td>
<td>NYA</td>
<td>ATL</td>
<td>1.1832</td>
</tr>
<tr>
<td></td>
<td>TEX</td>
<td>CHN</td>
<td>1.1743</td>
</tr>
<tr>
<td></td>
<td>ANA</td>
<td>SFN</td>
<td>1.1579</td>
</tr>
<tr>
<td></td>
<td>MIN</td>
<td>HOU</td>
<td>1.1539</td>
</tr>
<tr>
<td></td>
<td>OAK</td>
<td>LAN</td>
<td>1.1134</td>
</tr>
<tr>
<td>Team</td>
<td>Value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHA</td>
<td>1.1043</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLE</td>
<td>0.9813</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAL</td>
<td>0.9779</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DET</td>
<td>0.8951</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBA</td>
<td>0.8249</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOR</td>
<td>0.8166</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEA</td>
<td>0.7347</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KCA</td>
<td>0.6577</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHI</td>
<td>1.0731</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDN</td>
<td>1.0088</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLO</td>
<td>0.9807</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COL</td>
<td>0.9314</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIT</td>
<td>0.8776</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIN</td>
<td>0.8694</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NYN</td>
<td>0.8224</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIL</td>
<td>0.7711</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MON</td>
<td>0.7477</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARI</td>
<td>0.608</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 6*
<form method=post>
<select Name=YEAR size=1>
<option value="2000">2000
<option value="2001">2001
<option value="2002">2002
<option value="2003">2003
<option value="2004">2004
</select>
<select Name=EASYSTATS size=1>
<option value="YES">Easy On
<option value="NO">Easy Off
</select>
<br />
<input type=submit value="Generate Statistics">
</form>

<?php
include 'league.php';
if (isset($_POST['YEAR'])){
$YEAR = $_POST['YEAR'];
$EASYSTATS = $_POST['EASYSTATS'];
$LStats = array();
$LStats = LeagueStats($YEAR);
$TeamBAL = $LStats[0];
$TeamBOBL = $LStats[1];
$TeamHRRL = $LStats[2];
$TeamOBPL = $LStats[3];
$TeamSAL = $LStats[4];
$TeamSORL = $LStats[5];
$TeamOBSL = $LStats[6];
if ($EASYSTATS == 'NO'){
print("<table><th colspan=12><b>Year: $YEAR</b></th><tr><td><b>Team Name</b></td><td><b>Adjusted Production</b></td><td><b>Batting Average</b></td><td><b>Base On Balls</b></td><td><b>Home Run Ratio</b></td><td><b>On Base Percentage</b></td><td><b>Slugging Average</b></td><td><b>Strikeout Ratio</b></td><td><b>On Base Slugging</b></td><td><b>Wins</b></td><td><b>Losses</b></td><td><b>Winning %</b></td><td><b>Fielding</b></td></tr>"
<table>
<thead>
<tr>
<th>Team Name</th>
<th>Adjusted Production</th>
<th>Batting Average</th>
<th>Base On Balls</th>
<th>Home Run Ratio</th>
<th>On Base Percentage</th>
<th>Slugging Average</th>
<th>Strikeout Ratio</th>
<th>On Base Slugging</th>
<th>Wins</th>
<th>Losses</th>
<th>Winning %</th>
<th>Fielding</th>
</tr>
</thead>
<tbody>
<tr>
<td>League</td>
<td> </td>
<td> </td>
<td>0.4f</td>
<td>0.4f</td>
<td>0.4f</td>
<td>0.4f</td>
<td> </td>
<td> </td>
<td> </td>
<td> </td>
<td> </td>
<td> </td>
</tr>
</tbody>
</table>
}
if ($EASYSTATS == 'YES'){
print("<table><th colspan=5><b>Year: $YEAR</b></th><tr><td><b>Team Name</b></td><td><b>Adjusted Production</b></td><td><b>Winning %</b></td><td><b>Fielding</b></td><td><b>Aggregate Value</b></td></tr>"
<table>
<thead>
<tr>
<th>Team Name</th>
<th>Adjusted Production</th>
<th>Winning %</th>
<th>Fielding</th>
<th>Aggregate Value</th>
</tr>
</thead>
<tbody>
<tr>
<td> </td>
<td> </td>
<td> </td>
<td> </td>
<td> </td>
</tr>
</tbody>
</table>
}
$connectionstring = odbc_connect("baseball", ", ", "");
foreach($TeamNames as $TEAM){
$winsV = "SELECT Vscore, Hscore FROM $YEAR WHERE Visitor = '$TEAM';
$winsH = "SELECT Vscore, Hscore FROM $YEAR WHERE Home = '$TEAM';
$atbatV = "SELECT Vatbat FROM $YEAR WHERE Visitor = '$TEAM';
$atbatH = "SELECT Hatbat FROM $YEAR WHERE Home = '$TEAM';
$hitsV = "SELECT Vhits FROM $YEAR WHERE Visitor = '$TEAM';
$hitsH = "SELECT Hhits FROM $YEAR WHERE Home = '$TEAM';
$walksV = "SELECT Vwalks FROM $YEAR WHERE Visitor = '$TEAM';
$walksH = "SELECT Hwalks FROM $YEAR WHERE Home = '$TEAM';
$spherunsV = "SELECT Vhomeruns FROM $YEAR WHERE Visitor = '$TEAM';

$homerunsH = "SELECT Hhomeruns FROM $YEAR WHERE Home = '$TEAM'";
$hitbypitchH = "SELECT Hhitbypitch FROM $YEAR WHERE Home = '$TEAM'";
$sacfliesV = "SELECT Vsacflies FROM $YEAR WHERE Visitor = '$TEAM'";
$sacfliesH = "SELECT Hasacflies FROM $YEAR WHERE Home = '$TEAM'";
$doublesV = "SELECT Vdoubles FROM $YEAR WHERE Visitor = '$TEAM'";
$doublesH = "SELECT Hdoubles FROM $YEAR WHERE Home = '$TEAM'";
$triplesV = "SELECT Vtriples FROM $YEAR WHERE Visitor = '$TEAM'";
$triplesH = "SELECT Htriples FROM $YEAR WHERE Home = '$TEAM'";
$sacfliesV = "SELECT Vsacflies FROM $YEAR WHERE Visitor = '$TEAM'";
$sacfliesH = "SELECT Hsacflies FROM $YEAR WHERE Home = '$TEAM'";
$triplesV = "SELECT Vtriples FROM $YEAR WHERE Visitor = '$TEAM'";
$triplesH = "SELECT Htriples FROM $YEAR WHERE Home = '$TEAM'";

$QwinsV = odbc_do($connectionstring, $winsV);
$QwinsH = odbc_do($connectionstring, $winsH);
$QatbatV = odbc_do($connectionstring, $atbatV);
$QatbatH = odbc_do($connectionstring, $atbatH);
$QhitsV = odbc_do($connectionstring, $hitsV);
$QhitsH = odbc_do($connectionstring, $hitsH);
$QwalksV = odbc_do($connectionstring, $walksV);
$QwalksH = odbc_do($connectionstring, $walksH);
$QhomerunsV = odbc_do($connectionstring, $homerunsV);
$QhomerunsH = odbc_do($connectionstring, $homerunsH);
$QhitbypitchV = odbc_do($connectionstring, $hitbypitchV);
$QhitbypitchH = odbc_do($connectionstring, $hitbypitchH);
$QsacfliesV = odbc_do($connectionstring, $sacfliesV);
$QsacfliesH = odbc_do($connectionstring, $sacfliesH);
$QdoublesV = odbc_do($connectionstring, $doublesV);
$QdoublesH = odbc_do($connectionstring, $doublesH);
$QtriplesV = odbc_do($connectionstring, $triplesV);
$QtriplesH = odbc_do($connectionstring, $triplesH);
$QstrikeoutsV = odbc_do($connectionstring, $strikeoutsV);
$QstrikeoutsH = odbc_do($connectionstring, $strikeoutsH);

$Vtotalatbats = 0;
$Htotalatbats = 0;
while(odbc_fetch_row($QatbatV))
{
    $Vtotalatbats += odbc_result($QatbatV, "Vatbat");
}
while(odbc_fetch_row($QatbatH))
{
    $Htotalatbats += odbc_result($QatbatH, "Hatbat");
}

$Vtotalhits = 0;
$Htotalhits = 0;
while(odbc_fetch_row($QhitsV))
{
    $Vtotalhits += odbc_result($QhitsV, "Vhits");
}
while(odbc_fetch_row($QhitsH))
{
    $Htotalhits += odbc_result($QhitsH, "Hhits");
}

$Vtotalwalks = 0;
$Htotalwalks = 0;
while(odbc_fetch_row($QwalksV))
{
    $Vtotalwalks += odbc_result($QwalksV, "Vwalks");
}
while(odbc_fetch_row($QwalksH))
{
    $Htotalwalks += odbc_result($QwalksH, "Hwalks");
}

$Vtotalhomeruns = 0;
$Htotalhomeruns = 0;
while(odbc_fetch_row($QhomerunsV))
{
    $Vtotalhomeruns += odbc_result($QhomerunsV, "Vhomeruns");
}
while(odbc_fetch_row($QhomerunsH))
{
    $Htotalhomeruns += odbc_result($QhomerunsH, "Hhomeruns");
}

$Vtotalhitbypitch = 0;
$Htotalhitbypitch = 0;
while(odbc_fetch_row($QhitbypitchV))
{
    $Vtotalhitbypitch += odbc_result($QhitbypitchV, "Vhitbypitch");
}
while(odbc_fetch_row($QhitbypitchH))
{
    $Htotalhitbypitch += odbc_result($QhitbypitchH, "Hhitbypitch");
}

$Vtotalsacflies = 0;
$Htotalsacflies = 0;
while(odbc_fetch_row($QsacfliesV))
{
    $Vtotalsacflies += odbc_result($QsacfliesV, "Vsacflies");
}
while(odbc_fetch_row($QsacfliesH))
{
    $Htotalsacflies += odbc_result($QsacfliesH, "Hsacflies");
}

$Vtotaldoubles = 0;
$Htotaldoubles = 0;
while(odbc_fetch_row($QdoublesV))
{
    $Vtotaldoubles += odbc_result($QdoublesV, "Vdoubles");
}
while(odbc_fetch_row($QdoublesH))
{
    $Htotaldoubles += odbc_result($QdoublesH, "Hdoubles");
}

$Vtotaltriples = 0;
$Htotaltriples = 0;
while(odbc_fetch_row($QtriplesV))
{
    $Vtotaltriples += odbc_result($QtriplesV, "Vtriples");
}
while(odbc_fetch_row($QtriplesH))
{
    $Htotaltriples += odbc_result($QtriplesH, "Htriples");
}

$Vtotalsingles = $Vtotalhits - $Vtotaldoubles - $Vtotaltriples - $Vtotalhomeruns;
$Htotalsingles = $Htotalhits - $Htotaldoubles - $Htotaltriples - $Htotalhomeruns;

$Vtotalstrikeouts = 0;
$Htotalstrikeouts = 0;
while(odbc_fetch_row($QstrikeoutsV))
{
    $Vtotalstrikeouts += odbc_result($QstrikeoutsV, "Vstrikeouts");
}
while(odbc_fetch_row($QstrikeoutsH))
{
    $Htotalstrikeouts += odbc_result($QstrikeoutsH, "Hstrikeouts");
}

$Teamwins = 0;
$Teamlosses = 0;
while(odbc_fetch_row($QwinsV))
{
    if(odbc_result($QwinsV, "Vscore") > odbc_result($QwinsV, "Hscore"))
        $Teamwins++;
    else
        $Teamlosses++;
}
while(odbc_fetch_row($QwinsH))
{
    if(odbc_result($QwinsH, "Vscore") < odbc_result($QwinsH, "Hscore"))
        $Teamwins++;
    else
        $Teamlosses++;
}

$Vtotalputouts = 0;
$Htotalputouts = 0;
while(odbc_fetch_row($QputoutsV))
{
    $Vtotalputouts += odbc_result($QputoutsV, "Vputouts");
}
while(odbc_fetch_row($QputoutsH))
{
    $Htotalputouts += odbc_result($QputoutsH, "Hputouts");
}

$Vtotalassists = 0;
$Htotalassists = 0;
while(odbc_fetch_row($QassistsV))
{
    $Vtotalassists += odbc_result($QassistsV, "Vassists");
}
while(odbc_fetch_row($QassistsH))
{
    $Htotalassists += odbc_result($QassistsH, "Hassists");
}

$Vtotalerrors = 0;
$Htotalerrors = 0;
while(odbc_fetch_row($QerrorsV))
{
    $Vtotalerrors += odbc_result($QerrorsV, "Verrors");
}
while(odbc_fetch_row($QerrorsH))
{
    $Htotalerrors += odbc_result($QerrorsH, "Herrors");
}

/* Start Batting Average */
$TeamBAH = $Htotalhits / $Htotalatbats;
$TeamBAV = $Vtotalhits / $Vtotalatbats;
$TeamBA = (($Vtotalhits+$Htotalhits)/($Vtotalatbats+$Htotalatbats));
/* End Batting Average */

/* Start Base On Balls */
$TeamBOBH = $Htotalwalks / $Htotalatbats;
$TeamBOBV = \frac{V_{totalwalks}}{V_{totalatbats}};
$TeamBOB = \frac{V_{totalwalks} + H_{totalwalks}}{V_{totalatbats} + H_{totalatbats}};
/* End Base On Balls */
/* Start Home Run Ratio */
$TeamHRRH = \frac{H_{totalhomeruns}}{H_{totalatbats}};
$TeamHRRV = \frac{V_{totalhomeruns}}{V_{totalatbats}};
$TeamHRR = \frac{V_{totalhomeruns} + H_{totalhomeruns}}{V_{totalatbats} + H_{totalatbats}};
/* End Home Run Ratio */
/* Start On Base Percentage */
$TeamOBPH = \frac{H_{totalhits} + H_{totalhitbypitch}}{H_{totalatbats} + H_{totalwalks} + H_{totalhitbypitch} + H_{totalassflies}};
$TeamOBPV = \frac{V_{totalhits} + V_{totalhitbypitch}}{V_{totalatbats} + V_{totalwalks} + V_{totalhitbypitch} + V_{totalassflies}};
$TeamOBP = \frac{H_{totalhits} + H_{totalhitbypitch} + V_{totalhits} + V_{totalhitbypitch}}{H_{totalatbats} + H_{totalwalks} + H_{totalhitbypitch} + V_{totalatbats} + V_{totalwalks} + H_{totalassflies}};
/* End On Base Percentage */
/* Start Slugging Average */
$TeamSAH = \frac{H_{totalsingles} + (2 \times H_{totaldoubles}) + (3 \times H_{totaltriples}) + (4 \times H_{totalhomeruns})}{H_{totalatbats}};
$TeamSAV = \frac{H_{totalsingles} + (2 \times H_{totaldoubles}) + (3 \times H_{totaltriples}) + (4 \times H_{totalhomeruns})}{V_{totalatbats}};
$TeamSA = \frac{H_{totalsingles} + (2 \times H_{totaldoubles}) + (3 \times H_{totaltriples}) + (4 \times H_{totalhomeruns}) + H_{totalsingles} + (2 \times H_{totaldoubles}) + (3 \times H_{totaltriples}) + (4 \times H_{totalhomeruns})}{H_{totalatbats} + V_{totalatbats}};
/* End Slugging Average */
/* Start Strikeout Ratio */
$TeamSORH = \frac{H_{totalstrikeouts}}{H_{totalatbats}};
$TeamSORV = \frac{V_{totalstrikeouts}}{V_{totalatbats}};
$TeamSOR = \frac{H_{totalstrikeouts} + V_{totalstrikeouts}}{H_{totalatbats} + V_{totalatbats}};
/* End Strikeout Ratio */
/* Start On Base Slugging */
$TeamOBSH = $TeamOBPH + $TeamSAH;
$TeamOBPV = $TeamOBPV + $TeamSAV;
$TeamOBS = $TeamOBP + $TeamSA;
/* End On Base Slugging */
/* Start Adjusted Production */
$TeamAPRO = \frac{\frac{V_{totalatbats}}{V_{totalwalks}}}{\frac{H_{totalhomeruns}}{H_{totalatbats}}};
/* End Adjusted Production */
/* Start Win/Loss */
$TeamTotalGames = $Teamwins + $Teamlosses;
$TeamWP = \frac{\text{wins}}{\text{total games}};
/* End Win/Loss */
/* Start Fielding Average */
$TeamFA = \frac{H_{totalputouts} + V_{totalputouts} + H_{totalassists} + V_{totalassists}}{H_{totalputouts} + V_{totalputouts} + H_{totalassists} + V_{totalassists} + \text{errors}};
/* End Fielding Average */
/* Start Calculated Aggregate Value */
$TeamCAV = \frac{\text{APRO} \times \text{WP}}{\text{FA}};
/* End Calculated Aggregate Value */
<?php

function LeagueStats($YEAR)
{
  $Stats = array();
  $connectionstring = odbc_connect("baseball", ",", ",");
  $atbatL = "SELECT Vatbat FROM $YEAR";
  $hitsL = "SELECT Vhits FROM $YEAR";
  $walksL = "SELECT Vwalks FROM $YEAR";
  $homerunsL = "SELECT Vhomeruns FROM $YEAR";
  $hitbypitchL = "SELECT Vhitbypitch FROM $YEAR";
  $sacfliesL = "SELECT Vsacflies FROM $YEAR";
  $doublesL = "SELECT Vdoubles FROM $YEAR";
  $triplesL = "SELECT Vtriples FROM $YEAR";
  $strikeoutsL = "SELECT Vstrikeouts FROM $YEAR";
  $QatbatL = odbc_do($connectionstring, $atbatL);
  $QhitsL = odbc_do($connectionstring, $hitsL);
  $QwalksL = odbc_do($connectionstring, $walksL);
  $QhomerunsL = odbc_do($connectionstring, $homerunsL);
  $QhitbypitchL = odbc_do($connectionstring, $hitbypitchL);
  $QsacfliesL = odbc_do($connectionstring, $sacfliesL);
  $QdoublesL = odbc_do($connectionstring, $doublesL);
  $QtriplesL = odbc_do($connectionstring, $triplesL);
  $QstrikeoutsL = odbc_do($connectionstring, $strikeoutsL);
  $Ltotalatbats = 0;
  while(odbc_fetch_row($QatbatL))
  {
    $Ltotalatbats += odbc_result($QatbatL, "Vatbat");
  }
  $Ltotalhits = 0;
  while(odbc_fetch_row($QhitsL))
  {
    $Ltotalhits += odbc_result($QhitsL, "Vhits");
  }
  $Ltotalwalks = 0;
  while(odbc_fetch_row($QwalksL))
  {
    $Ltotalwalks += odbc_result($QwalksL, "Vwalks");
  }
  $Ltotalhomeruns = 0;
  while(odbc_fetch_row($QhomerunsL))
  {
    $Ltotalhomeruns += odbc_result($QhomerunsL, "Vhomeruns");
  }
}
$Ltotalhitbypitch = 0;
while(odbc_fetch_row($QhitbypitchL))
{
    $Ltotalhitbypitch += odbc_result($QhitbypitchL, "Vhitbypitch");
}

$Ltotalsacflies = 0;
while(odbc_fetch_row($QsacfliesL))
{
    $Ltotalsacflies += odbc_result($QsacfliesL, "Vsacflies");
}

$Ltotaldoubles = 0;
while(odbc_fetch_row($QdoublesL))
{
    $Ltotaldoubles += odbc_result($QdoublesL, "Vdoubles");
}

$Ltotaltriples = 0;
while(odbc_fetch_row($QtriplesL))
{
    $Ltotaltriples += odbc_result($QtriplesL, "Vtriples");
}

$Ltotalsingles = $Ltotalhits - $Ltotaldoubles - $Ltotaltriples - $Ltotalhomeruns;

$Ltotalstrikeouts = 0;
while(odbc_fetch_row($QstrikeoutsL))
{
    $Ltotalstrikeouts += odbc_result($QstrikeoutsL, "Vstrikeouts");
}

/* Start Batting Average */
$TeamBAL = $Ltotalhits / $Ltotalatbats;
$Stats[0] = $TeamBAL;
/* End Batting Average */

/* Start Base On Balls */
$TeamBOBL = $Ltotalwalks / $Ltotalatbats;
$Stats[1] = $TeamBOBL;
/* End Base On Balls */

/* Start Home Run Ratio */
$TeamHRRL = $Ltotalhomeruns / $Ltotalatbats;
$Stats[2] = $TeamHRRL;
/* End Home Run Ratio */

/* Start On Base Percentage */
$TeamOBPL = ($Ltotalhits + $Ltotalhitbypitch) / ($Ltotalatbats + $Ltotalwalks + $Ltotalhitbypitch + $Ltotalsacflies);
$Stats[3] = $TeamOBPL;
/* End On Base Percentage */

/* Start Slugging Average */
$TeamSAL = ($Ltotalsingles + (2 * $Ltotaldoubles) + (3 * $Ltotaltriples) + (4 * $Ltotalhomeruns)) / $Ltotalatbats;
$Stats[4] = $TeamSAL;
/* End Slugging Average */

/* Start Strikeout Ratio */
$TeamSORL = $Ltotalstrikeouts / $Ltotalatbats;
$Stats[5] = $TeamSORL;
/* End Strikeout Ratio */
/* Start On Base Slugging */
$TeamOBSL = $TeamOBPL + $TeamSAL;
$Stats[6] = $TeamOBSL;
/* End On Base Slugging */
return($Stats);
}