Project_0506_15_Readme

Overview

Welcome to the MoonShot Pro Analytics toolkit designed for the Baseball Team of the University Of Maryland. Our goal here is to equip coaches with valuable insights derived from the historical results of the past 16 years, i.e. 2005 to 2020. By leveraging comprehensive data analysis, we aim to offer a nuanced understanding of the team's performance and dynamics.

Objective

Our analysis delves into several critical aspects of the team's performance addressing questions such as:

- ➔ Average winning margin per year
- ➔ Average losing margin per year
- ➔ Average number of hits to ratio per year
- ➔ Average number of errors per year
- ➔ Year wise-team that lost against by the highest margin
- ➔ Year wise team that won against by highest margin
- ➔ State Wise winning and losing
- ➔ What is the winning and losing margin across different teams for all the years?

Directory Structure

-Project_0506_15_Proposal.docx: business proposal doc. containing the objective and proposal
-SQL Files:Project_0506_15_Business_Queries.sql, Project_0506_15_Create_and_Insert.sql: a file containing create and insert queries and a file with business queries is used.
-Project_0506_15_Presentation.pptx: a PowerPoint presentation depicting the work
-Project_0506_15_Moonshot_Dashboard.twbx: a tableau file containing all the visualizations
-Project_0506_15_Readme.docx: a folder containing all the data inserted into the database

Steps to replicate

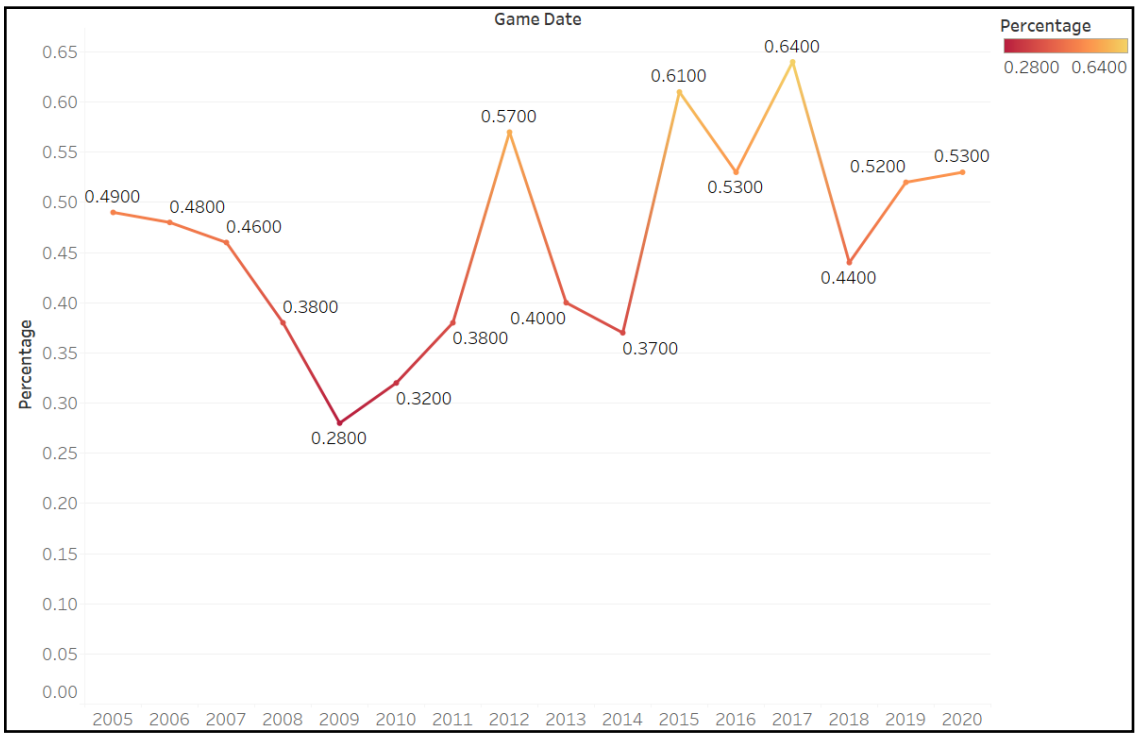
1. Use the Python file to merge different year's Excel files into a single file and generate primary keys for four databases i.e. gameId, scoreId, locationId, and teamId.
2. Use https://sqlizer.io/ to convert the master Excel file to SQL insert statements.
3. Follow the *create_insert.sql* file to insert the data into the database.
4. Use the *business_objectives.sql* file to run the analysis queries.
5. Use *Project_0506_15.twb* to visualize the analysis

Outputs

[B1] What was the year-wise win percentage for UMD baseball?

	Year	Win Percentage
1	2005	0.49
2	2006	0.48
3	2007	0.46
4	2008	0.38
5	2009	0.28
6	2010	0.32
7	2011	0.38
8	2012	0.57
9	2013	0.4
10	2014	0.37
11	2015	0.61
12	2016	0.53
13	2017	0.64
14	2018	0.44

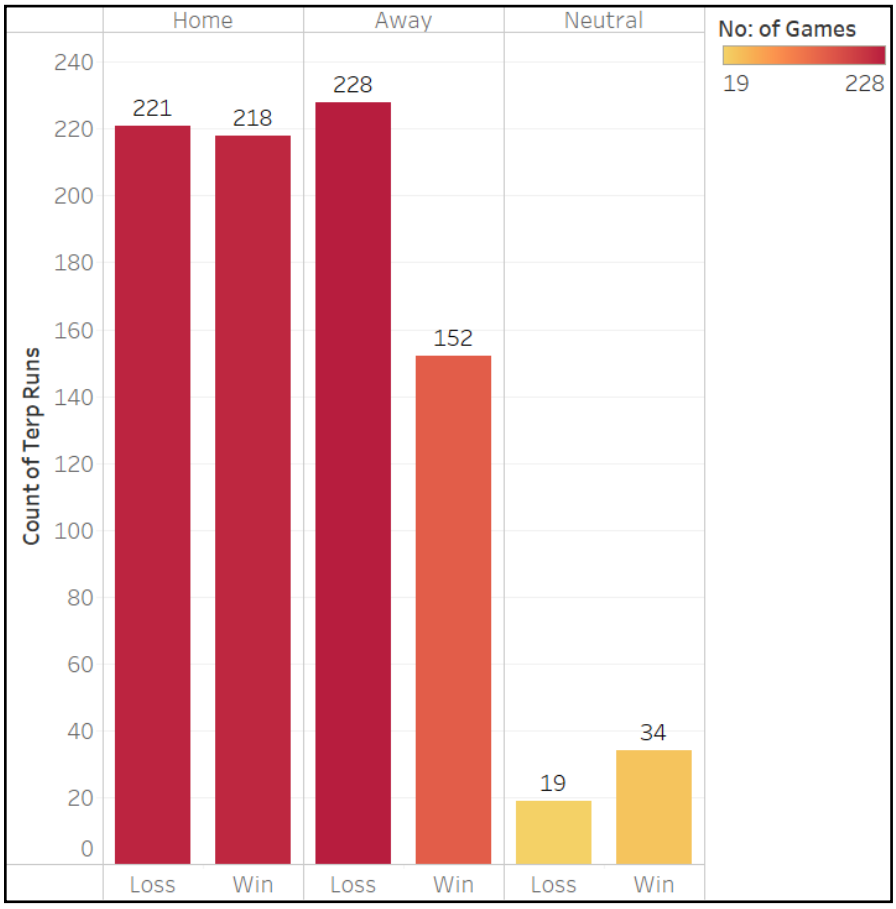
Using the years and the calculated field for percentage, we are evaluating runs of UMD Terps to measure the winning percentage over the years.



[B2] How many games won by MD were at home away and neutral?

	Game At	Result	NumGames
1	Away	loss	228
2	Home	loss	221
3	Neutral	loss	19
4	Away	win	152
5	Home	win	218
6	Neutral	win	34

Using the locations - home, away and neutral; we are calculating the total wins and losses of UMD Terps.

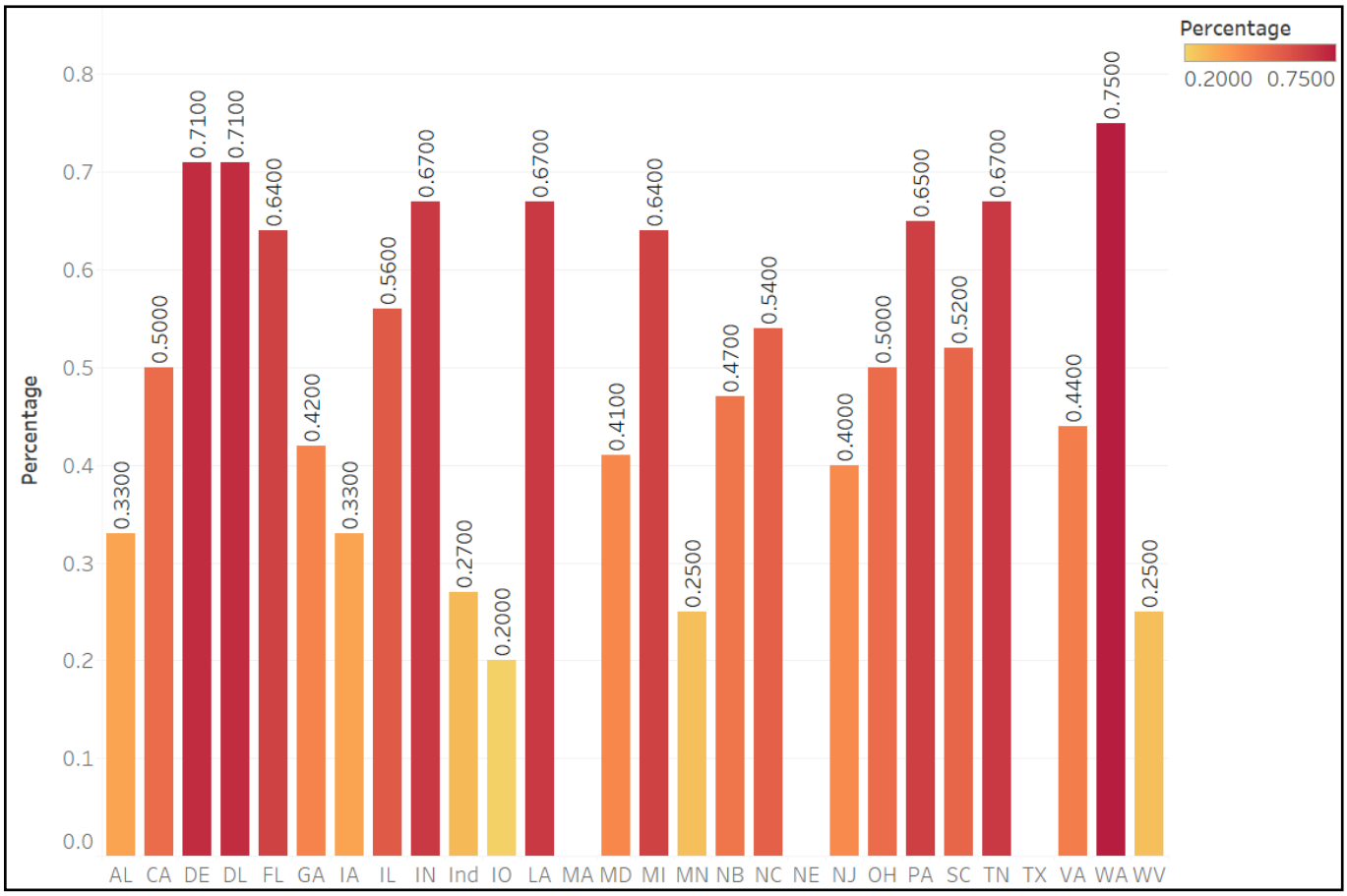


[B3] Which month did UMD win the most games?

Results | Messages

	GameMonth	NumWins
1	3	142
2	4	132
3	5	70
4	2	54
5	6	6

Using the Month of matches we are calculating the number of wins and losses by evaluating runs of teams.

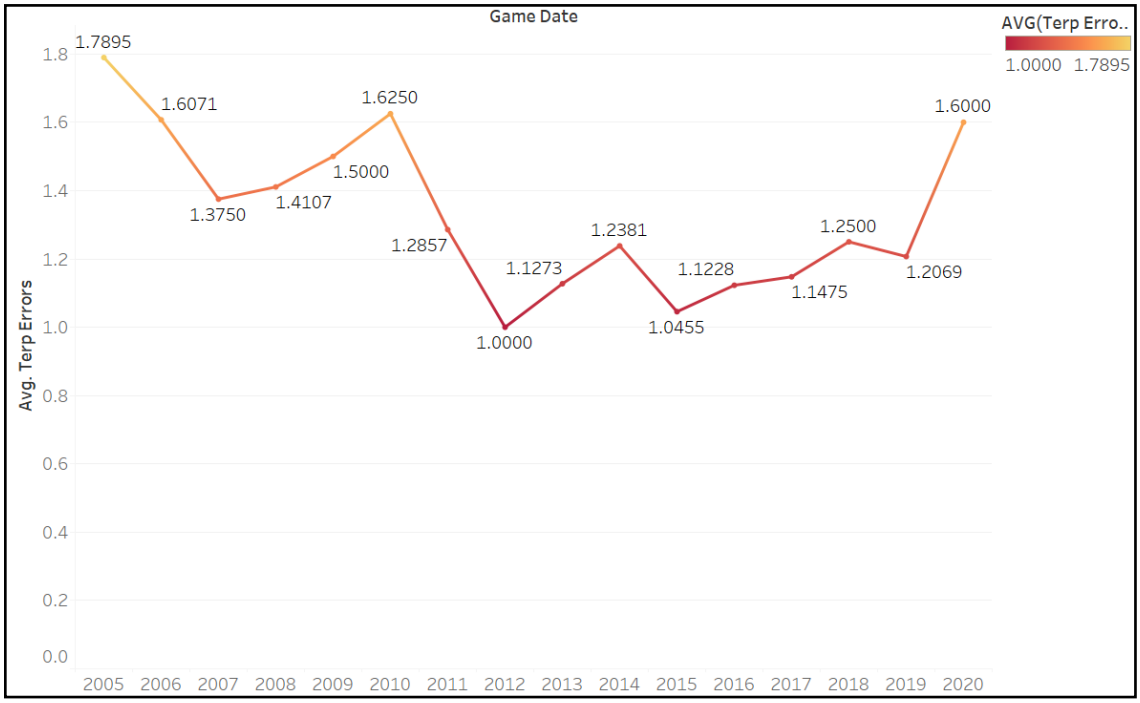


[B4] What was the state-wise winning percentage in the past sixteen years?

Results | Messages

	State	Win Percentage
1	AL	33.33
2	CA	50
3	DE	71.43
4	DL	71.43
5	FL	63.89
6	GA	41.67
7	IA	33.33
8	IL	55.56
9	IN	66.67
10	Ind	27.27
11	IO	20
12	LA	66.67
13	MA	NULL
14	MD	41.48

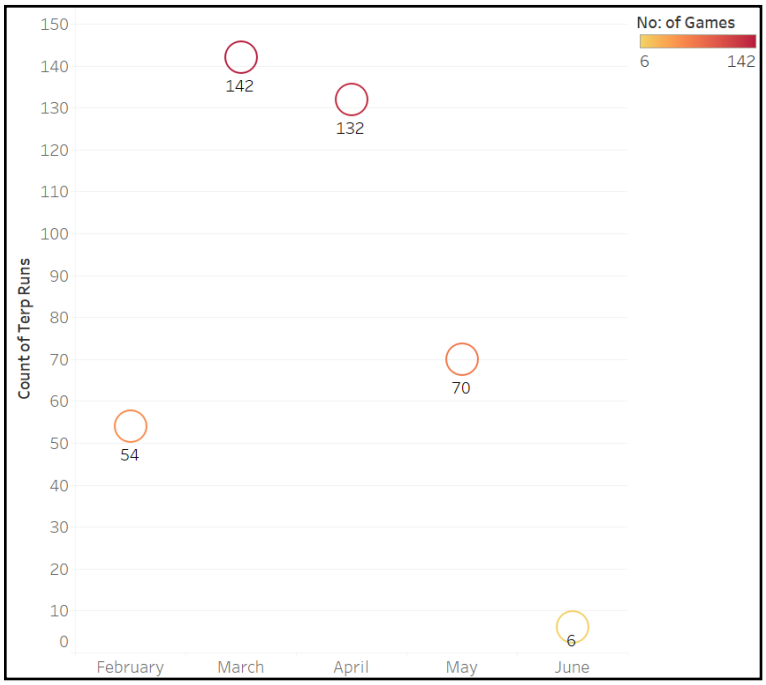
The percentage of state-wise winning is calculated by evaluating wins and losses and making them a percentage.



[B5] What was the average number of hits-to-run ratio per year?

	Year	Avg. Run to Hit Ratio
1	2005	0.55
2	2006	0.54
3	2007	0.56
4	2008	0.53
5	2009	0.53
6	2010	0.52
7	2011	0.49
8	2012	0.55
9	2013	0.47
10	2014	0.46
11	2015	0.63
12	2016	0.55
13	2017	0.67
14	2018	0.12

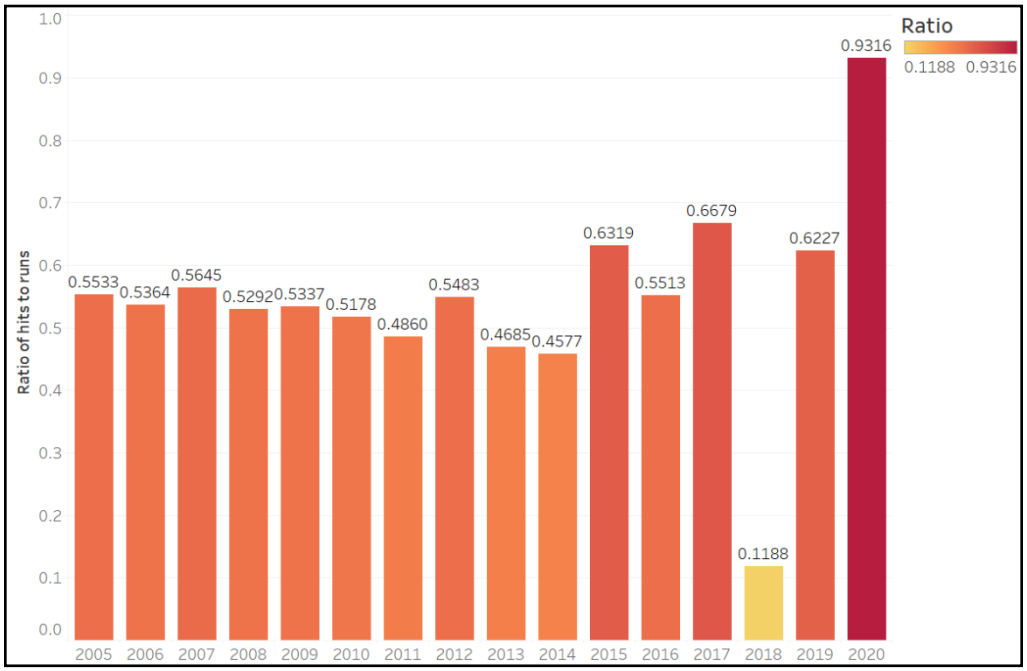
The hit-to-run ratio is the evaluation of the ratio of the runs and hits to measure the overall performance.



[B6] What was the average number of errors made per year?

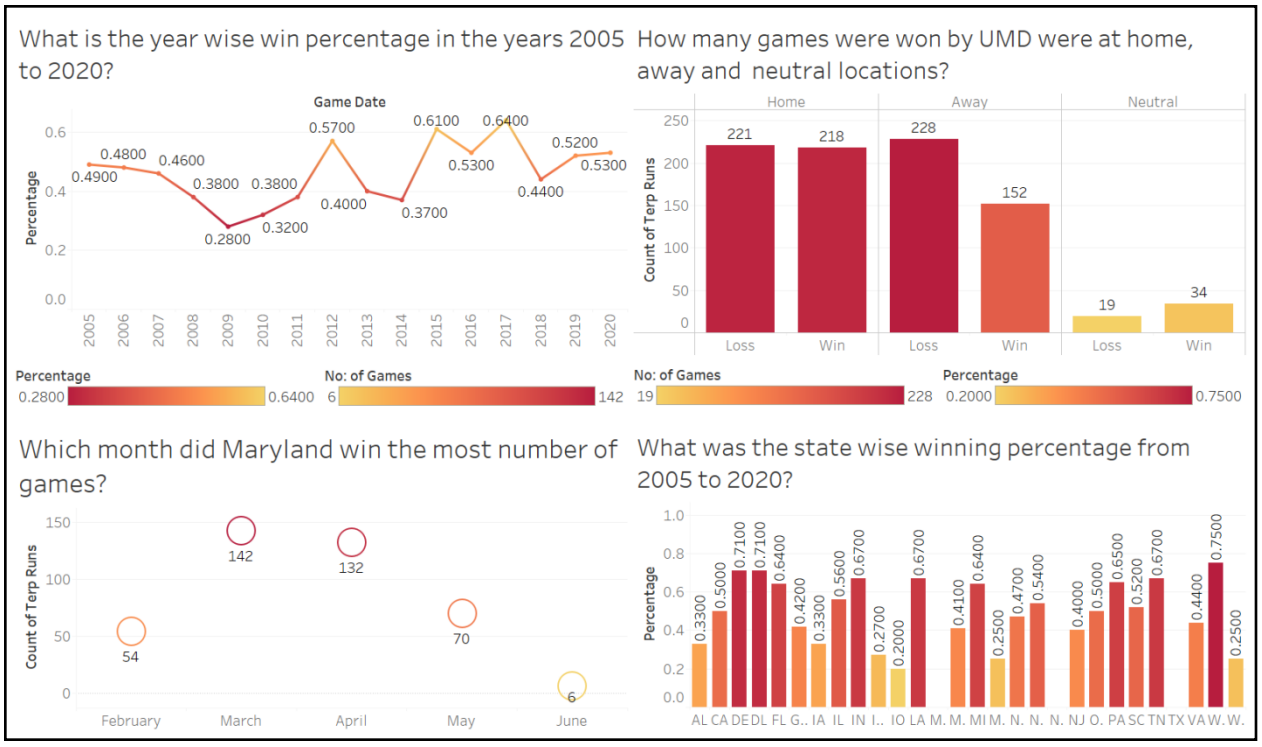
	Year	Avg. Errors
1	2005	1.79
2	2006	1.61
3	2007	1.38
4	2008	1.41
5	2009	1.5
6	2010	1.63
7	2011	1.29
8	2012	1
9	2013	1.13
10	2014	1.24
11	2015	1.05
12	2016	1.12
13	2017	1.15
14	2018	1.25

The mean of errors made every year is calculated in this query.

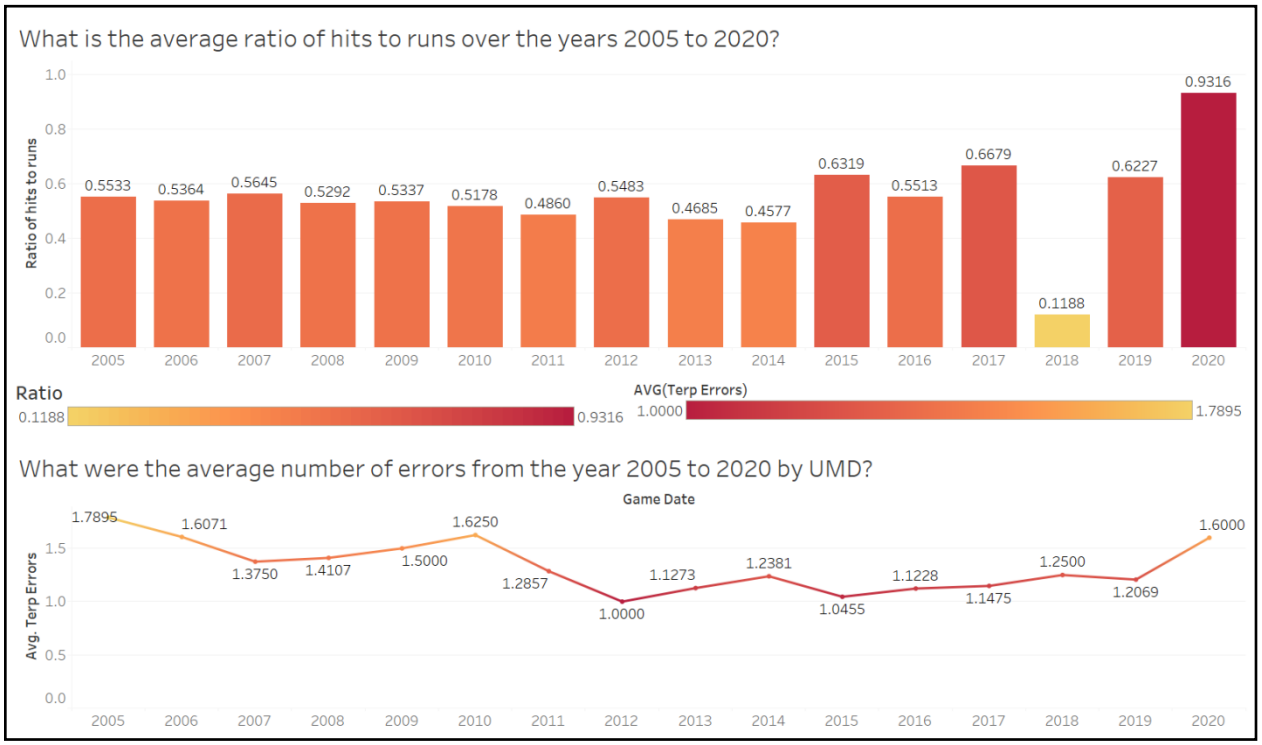


Dashboards:

The first four queries put together are a collection of winning evaluations of games played.



This dashboard shows the average performance evaluation of hits and ratio and error average.



Recommendation and Future Works

➔ *UMD's Baseball has been consistently performing in the past ten years!*
➔ The records of the game in the database have been inconsistent in terms of data being recorded, so we recommend having a better data collection.
➔ Having information consistency will help better analyze its effect on the team's performance.
➔ For this, we propose ***True Tournament Standing***

Comprehensive Team Performance Metric

$$\textbf{True Tournament Standing} = \sum_{n=1}^{x} \frac{runs}{hits} * \left(k_r e^{runs\ diff}\right) * k_t T * k_h H * k_s S * k_r R$$

$$\text{Temperature T} = \begin{cases} 1 & \text{when t} < 75 \\ 1 - 0.01 * (t - 75)/100 & \text{when t} > 75 \end{cases}$$

$$\text{Humidity H} = \begin{cases} 1 & \text{when H} < h_{ref} \\ 1 - 0.01 * (H - h_{ref})/100 & \text{when H} > x \end{cases}$$

$$\mu = \text{mean of home ground attendance}$$
$$\sigma = \text{standard deviation of home ground attendance}$$
$$\text{Surrounding S} = \begin{cases} 1 & \text{attendance} = \mu \\ 1.004^n & \text{attendance} = \mu + n^*\sigma \end{cases}$$

$$\textit{Rank R} = \frac{1}{1 + e^{(\text{rank1 - rank2})}}$$

Example TTS For Single Game

	Runs	Hits	Rank	Home
Team1	8	12	2	Yes
Team 2	10	15	12	No

Temperature: 85
Humidity: 70 and **href**: 60
Home Attendance: 1 std above mean

TTS team 1 = (8/12)*0.135*1.001*1.001*1.004*
= **0.091**
TTS team 2 = (10/15)*7.385*1.001*1.001*1
= **4.932**

References
Temperature effect : Brandon Lee D. Koch & Anna K. Pannorska
Crowd effect: Erin E. Smith & Jon D. Groetzinger

Resources & References

-Dataset: https://umterps.com/sports/baseball/schedule/2006
-Excel to SQL converter: https://sqlizer.io/

Python Code

This is the respective Python code used to segregate data into four different Excel files viz. GameDB, ScoreDB, TeamDB, and LocationDB, and creating a primary key for each.

```
import os
import sys
import random
import math
import numpy as np
import pandas as pd
```

```
dir_path = ['./Data/Raw_data/'+ x for x in os.listdir('./Data/Raw_data/')]
print('Found {} years of data'.format(len(dir_path)))

df = pd.DataFrame()
for i in dir_path:
    df = pd.concat([df,pd.read_excel(i)])
print('Merged {} of data in single dataframe of shape
{}'.format(len(dir_path),df.shape))
df.to_excel('./master_compilation.xlsx')

print('Initiating master file for all the data ...')

# Creating team database with following columns
# Team: teamId, tameName, teamHome
print('Creating team database...')
unique_opponent = df.Opponent.unique()
unique_opponent_id = random.sample(range(100,999),len(unique_opponent))
team_db = pd.DataFrame(list(zip(unique_opponent_id, unique_opponent)),
columns=['teamId','teamName'])
team_db.to_excel('./teamDB.xlsx')
print('Found {} unique team names and saved the results at
./TeamDB.xlsx'.format(len(unique_opponent_id)))
print(team_db.head())

# Creating location database with following columns
# Location: locationId, stadiumName
print('Creating location database...')
unique_location = df.Location.unique()
unique_location_id = random.sample(range(100,999), len(unique_location))

#loc_dict = [random.randint(100,999) for x in df.Location.unique()]
stadiumName = [' '.join(i.split()[:-1]).replace(',','') for i in
df.Location.unique()]
state = [i.split()[-1].replace('.','') if len(i.split()[-
1].replace('.',''))<4 else 0 for i in df.Location.unique()]
```

```
location_db = pd.DataFrame(list(zip(unique_location_id, stadiumName,
state)), columns=['locId','locStadium','locState'])

location_db[-1:] = [174,'Omaha','NE']

location_db.to_excel('./locationDB.xlsx')

print('Found {} unique stadium names and saved the results at
./locationDB.xlsx'.format(len(unique_location_id)))

print(location_db.head())


# Creating game database with following columns

# 'gameid', 'scoreId', 'locationId', 'teamId1','teamId2', 'gameDate',
'gameStartTime','gameDuration', 'gameAttendance', 'gameWeather'

print('Creating game database...')

gameid = random.sample(range(10000,99999), df.shape[0])
#[random.randint(10000,99999) for i in range(df.shape[0])]

score_id = random.sample(range(10000,99999), df.shape[0])

tmp_multiplier = []


tmp_multiplier = [1 if i !='Cancelled' or i != 0  else 0 for i in
(df.Result)]

score_id = list(np.multiply(score_id,tmp_multiplier))


#game_loc_id = [location_db[location_db.locStadium == '
'.join(x.split()[:-1]).replace(',','')].locId.to_list() for x in
df.Location.to_list()]

game_loc_id = [location_db[location_db.locStadium == ' '.join(x.split()[:-
1]).replace(',','')].locId.to_list() for x in df.Location.to_list()]

locationId = [x for l in game_loc_id for x in l]


teamId1 = [100 for i in range (df.shape[0])]

teamId2 = [team_db[team_db.teamName == x].teamId.to_list() for x in
df.Opponent.to_list()]

teamId2 = [x for l in teamId2 for x in l]


gameDate = df.Date

gameDay = df.Weekday

gameStartTime = df.Time

gameDuration = df.Duration
```

```
gameAttendance = df.Attendance

gameWeather = df.Weather

game_db = pd.DataFrame(list(zip(gameid, score_id, locationId,
teamId1,teamId2, gameDate, gameDay, gameStartTime,gameDuration,
gameAttendance, gameWeather)),

                       columns=['gameid', 'scoreId', 'locationId',
'teamId1','teamId2', 'gameDate', 'gameDay',
'gameStartTime','gameDuration',

                                'gameAttendance', 'gameWeather'])

game_db.to_excel('./gameDB.xlsx')

print('Created game database at ./gameDB.xlsx with shape
{}'.format(game_db.shape))

print(game_db.head())



# Creating Score database with following columns

# Score: scoreId, gameId, teamId1, runs1,hit1,error1, teamid2, runs2,
hits2, error2

print('Creating score database...')

gameId = [game_db[game_db['scoreId']==x].gameid.to_list() for x in
score_id if x !=0]

gameId = [x for l in gameId for x in l]

#scoreId = [x for x in score_id if score_id !=0]

scoreId = [game_db[game_db['scoreId']==x].scoreId.to_list() for x in
score_id if x !=0]

scoreId = [x for l in scoreId for x in l]


teamId1 = [100 for x in score_id if score_id !=0]

teamId2 = [game_db[game_db['scoreId']==x ].teamId2.to_list() for x in
score_id if x !=0]

teamId2 = [x for l in teamId2 for x in l]

runs1 = df[df.Runs_1.notnull()].Runs_1

runs2 = df[df.Runs_1.notnull()].Runs_2

hits1 = df[df.Runs_1.notnull()].Hits_1

hits2 = df[df.Runs_1.notnull()].Hits_2

error1 = df[df.Runs_1.notnull()].Errors_1

error2 = df[df.Runs_1.notnull()].Errors_2
```

```
score_db =
pd.DataFrame(list(zip(scoreId,gameId,teamId1,teamId2,runs1,runs2,hits1,hit
s2,error1,error2)),


columns=['scoreId','gameId','teamId1','teamId2','runs1','runs2','hits1','h
its2','error1','error2'])

print(score_db.shape)

score_db.to_excel('./scoreDB.xlsx')

print('Created score database at ./scoreDB.xlsx with shape
{}'.format(score_db.shape))

print(score_db.head())


print('Finished.')
```