파이썬
데이터 분석

이종욱

INDEX

주제

제주도 오름

제주도에 있는 오름의 분포지역

어느 지역에 있는 오름이 더 높은가 ?

EPL

맨맨아첼 ?

15-16시즌 레스터 시티의 우승이
기적이라고 불리는 이유

가설 - 제주도

- 제주도 여행 중 제주시 보다 서귀포시가 커브길이 많고 도로의 높낮이가 달라져서 운전하기 힘들었다.
- 서귀포에 오름이 많이 분포해 있어서 이렇게 느끼는 건가 ?
- 만약 아니라면 서귀포에 높은 오름들이 있어서 운전하기 힘들게 느껴지는 것인가 ?

구현 - 제주도

	오름명	행정시	소재지	비고	표고	면적	형태	데이터기준일자
0	가마오름	제주시	제주특별자치도 제주시 한경면 청수리 1202	51	140.5	154486	말굽형(북동향)	2022-02-14
1	가메오름	제주시	제주특별자치도 제주시 애월읍 봉성리 산124	17	372.2	28371	복합형	2022-02-14
2	가메옥	제주시	제주특별자치도 제주시 구좌읍 송당리 1712	28	368.0	22764	복합형	2022-02-14
3	가메창	제주시	제주특별자치도 제주시 한경면 저지리 1496	6	145.8	17037	원형	2022-02-14
4	가문이오름	서귀포시	제주특별자치도 서귀포시 표선면 가시리 산158-2	106	496.2	116176	말굽형(남서향)	2022-02-14

- data.head()로 출력한 결과이다.
- 분포도를 보기 위해 행정시와 높이를 알기 위한 표고를 이용한다.

구현 - 제주도

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

data = pd.read_csv('/content/jeju.csv', encoding="cp949")

jeju = data[data["행정시"] == "제주시"]
seo = data[data["행정시"] == "서귀포시"]

print("제주시",len(jeju), "/",len(data))
print("서귀포" ,len(seo), "/",len(data))
```

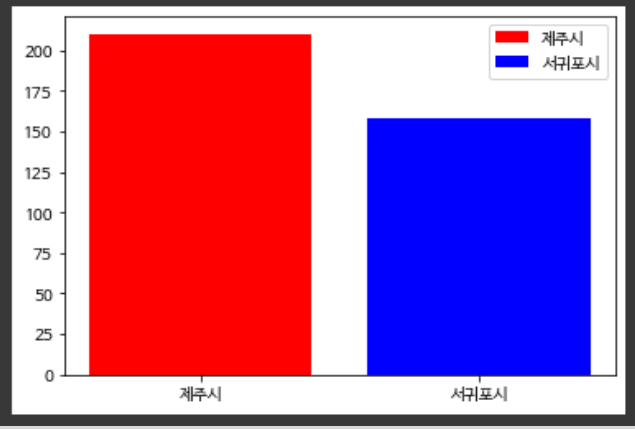
```
제주시 210 / 368
서귀포 158 / 368
```

- “행정시”셀에서 “제주시”는 jeju, “서귀포시”는 seo에 대입한다.
- Print로 확인 후 시각화 작업을 한다.

구현 - 제주도

```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

data = pd.read_csv('/content/jeju.csv', encoding="cp949")

jeju = data[data["행정시"] == "제주시"]
seo = data[data["행정시"] == "서귀포시"]

print("제주시",len(jeju), "/",len(data))
print("서귀포" ,len(seo), "/",len(data))
plt.rc('font', family='NanumBarunGothic')
plt.bar(jeju['행정시'],len(jeju),label="제주시",color="red")
plt.bar(seo['행정시'],len(seo),label="서귀포시",color="blue")
plt.legend()
plt.show()
```

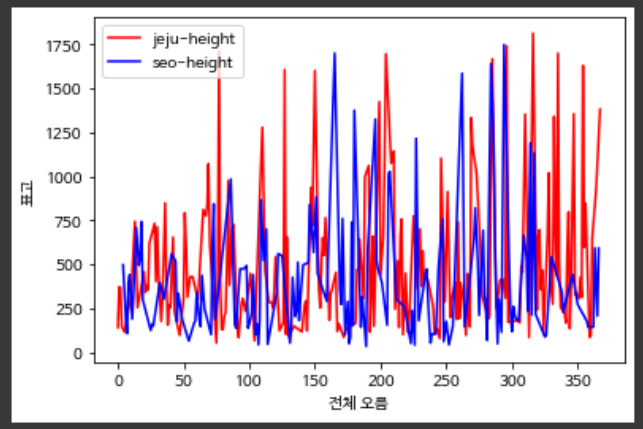
```
제주시 210 / 368
서귀포 158 / 368
```



- 첫 번째 가설인 서귀포에 오름이 많이 분포해 있는가?는 증명이 되지 못하였다...
- 그렇다면 두번째 가설인 서귀포에 높은 오름들이 있어서 힘든 것인가?를 증명해보자 !

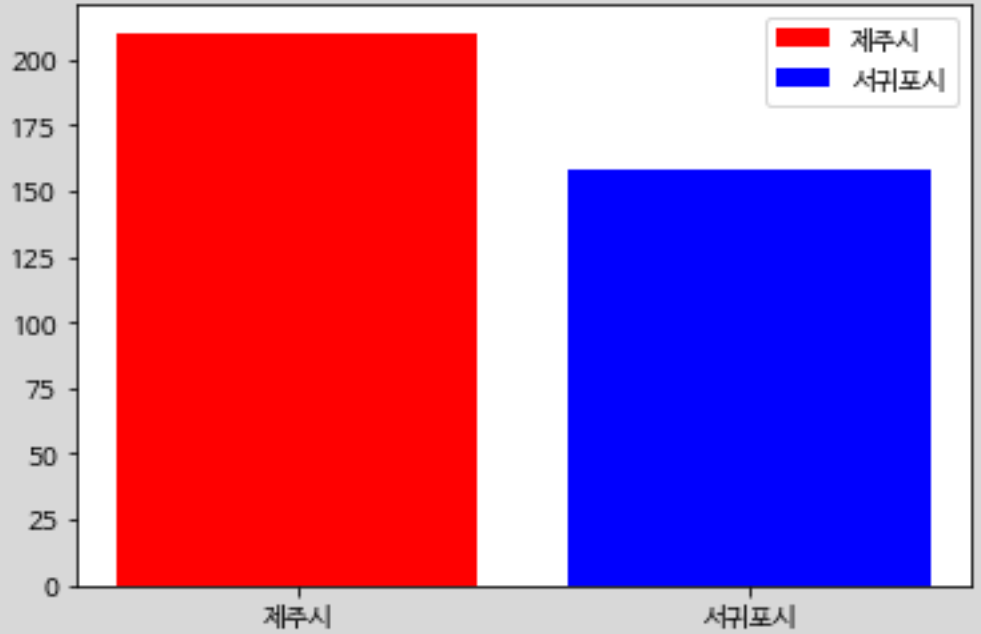
구현 - 제주도

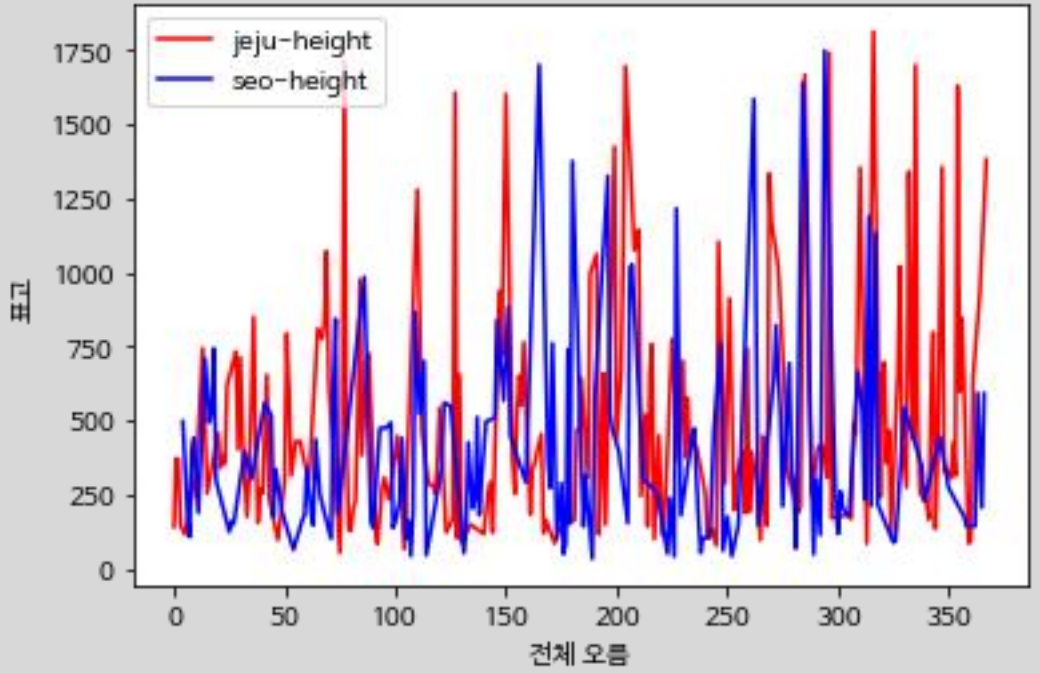
```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

data = pd.read_csv('/content/jeju.csv', encoding="cp949")

jeju = data[data["행정시"] == "제주시"]
seo = data[data["행정시"] == "서귀포시"]

# print("제주시",len(jeju), "/",len(data))
# print("서귀포" ,len(seo), "/",len(data))

plt.plot(jeju['표고'], label='jeju-height',color='r')
plt.plot(seo['표고'], label='seo-height',color='b')
plt.rc('font', family='NanumBarunGothic')
plt.xlabel("전체 오름")
plt.ylabel("표고")
plt.legend()
plt.show()
```



- 빨간색은 제주시, 파란색은 서귀포 시인데 표를 보니 서귀포도 높은 오름들이 있지만 제주시가 더 많았다.
- 서귀포시 여행하는 날에 유난히 피곤했던 것 같다.

결과 - 제주도





- 오름들은 제주시가 더 많았고, 높은 오름들도 제주시가 많았다.
- 운전하기 힘들게 느껴졌던건 그날 피곤해서 인 것 같다.

가설 - EPL

- 흔히 EPL 팬들은 위의 엠블럼 팀들이 우승을 거머쥔다고 해서 맨맨아첼이라고 부른다.
- 왜 맨맨아첼이라고 불리는지, 우승 팀들의 우승 횟수를 출력해서 증명해보자!

가설 - EPL

- 여우군단이라고 별명을 가지고 있는 레스터시티는 2016시즌에 잉글랜드 프리미어 리그를 우승했는데 모든이가 이를 기적이라고 한다.
- 레스터 시티의 시즌별 기록들로 왜 기적이라 불렸는지 증명해보자!

구현 - EPL

```
EPL_data = pd.read_csv('/content/results.csv', encoding="cp949")

# 시즌 list
season=list(EPL_data['Season'].unique())#시즌 별로 가져옴

#enumerate() 함수는 인자로 넘어온 목록을 기준으로 인덱스와 원소를 차례대로 접근하기
for count, i in enumerate(season, 1994):
  rank = {}
  #loc는 소괄호()가 아닌 대괄호[]로 감쌉니다.
  #.loc[행 , 열] => season의 i번째 index의 모든 열을 가져온다.
  s = EPL_data.loc[EPL_data['Season']==i,:]
  for j in s.index:

    if s['FTR'][j] == "H":#s["FTR"]에서 가져온 항목 중에 H가 있다면
      if s["HomeTeam"][j] in rank:#딕셔너리 안에 있으면 추가
      #3을 더해서 대입해라
        rank[s["HomeTeam"][j]] = rank[s['HomeTeam'][j]] + 3
      else:
        rank[s["HomeTeam"][j]] = 3

    elif s['FTR'][j] == "D":#s["FTR"]에서 가져온 항목 중에 D가 있다면
      if s["HomeTeam"][j] in rank:#딕셔너리 안에 있으면 추가
        #1을 더해서 대입해라
        rank[s["HomeTeam"][j]] = rank[s['HomeTeam'][j]] + 1
      else:
        rank[s["HomeTeam"][j]] = 1
        #어웨이 팀 동점
      if s["AwayTeam"][j] in rank:#딕셔너리 안에 있으면 추가
        rank[s["AwayTeam"][j]] = rank[s['AwayTeam'][j]] + 1
      else:
        rank[s["AwayTeam"][j]] = 1


    else:#s["FTR"]에서 가져온 항목 중에 D가 있다면
      if s["AwayTeam"][j] in rank:#딕셔너리 안에 있으면 추가
        rank[s["AwayTeam"][j]] = rank[s['AwayTeam'][j]] + 3
      else:
        rank[s["AwayTeam"][j]] = 3

  df = pd.DataFrame(list(rank.items()), columns=["TEAM", "Point"])
```

```
df = pd.DataFrame(list(rank.items()), columns=["TEAM", "Point"])
# sort_values 메서드는 값을 기준으로 레이블을 정렬하는 메서드입니다.
# 즉, 데이터 df를 Point 컬럼의 값을 기준으로 정렬하는 메소드
# 기본 정렬 방식은 오름차순(ascending)입니다.
# 내림차순으로 정렬하고 싶다면 ascending 옵션을 False로 설정하면 됩니다.
df = df.sort_values(by='Point' ,ascending=False)
df= df.reset_index(drop=True)#drop = true면 기존 인덱스 제거 / false면 유지
df.index = df.index + 1
df["season"] = count

#변수명을 자동화하는 globals 함수
globals()["s_"+ str(count)] = df
```

- 가져온 데이터에서 홈팀, 어웨이팀이 이기면 승점 3점, 비기면 1점을 추가해서 rank 딕셔너리에 추가
- 데이터 프레임 sort정렬로 point 기준으로 내림차순
- globals()함수로 변수명 지정

구현 - EPL

	TEAM	Point	season
1	Man United	92	1994
1	Blackburn	89	1995
1	Man United	82	1996
1	Man United	75	1997
1	Arsenal	78	1998
1	Man United	79	1999
1	Man United	91	2000
1	Man United	80	2001
1	Arsenal	87	2002
1	Man United	83	2003
1	Arsenal	90	2004
1	Chelsea	95	2005
1	Chelsea	91	2006
1	Man United	89	2007
1	Man United	87	2008
1	Man United	90	2009
1	Chelsea	86	2010
1	Man United	80	2011
1	Man United	89	2012
1	Man United	89	2013
1	Man City	86	2014
1	Chelsea	87	2015
1	Leicester	81	2016
1	Chelsea	93	2017
1	Man City	100	2018
1	Man City	98	2019
1	Liverpool	99	2020

- 역대 우승팀 들의 승점과 시즌을 출력
- 맨맨아첼을 증명하기 위해 우승 횟수 카운트를 해보자!

구현 - EPL

```
best_season['season'].groupby(best_season["TEAM"]).count()
```

```
TEAM
Arsenal        3
Blackburn      1
Chelsea        5
Leicester      1
Liverpool      1
Man City       3
Man United    13
Name: season, dtype: int64
```

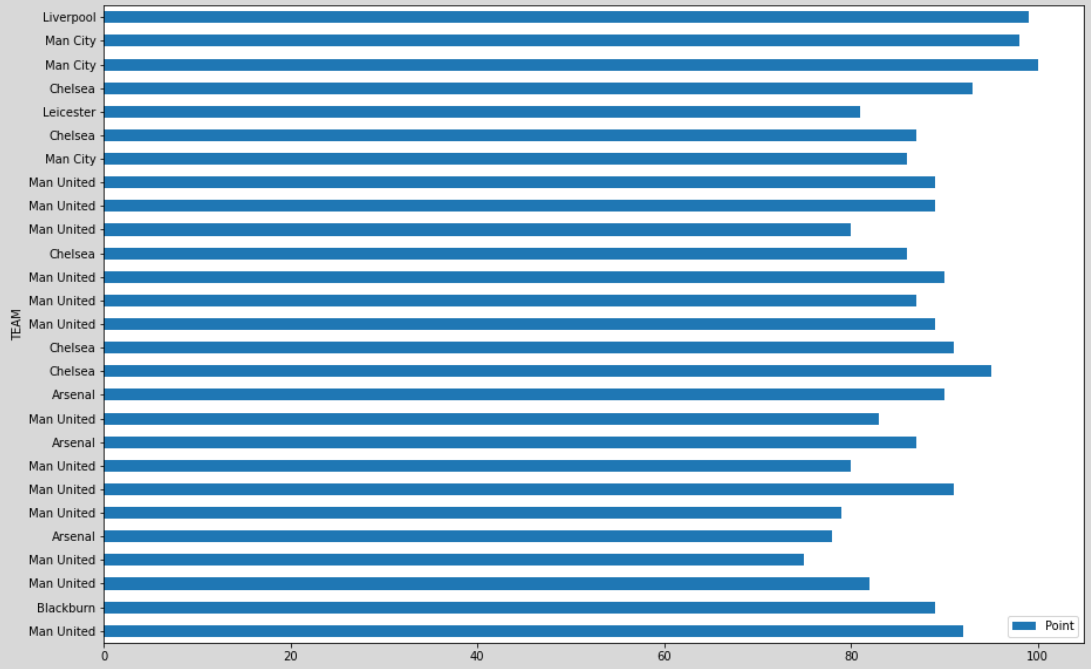
13회

3회

3회

5회

구현 - EPL

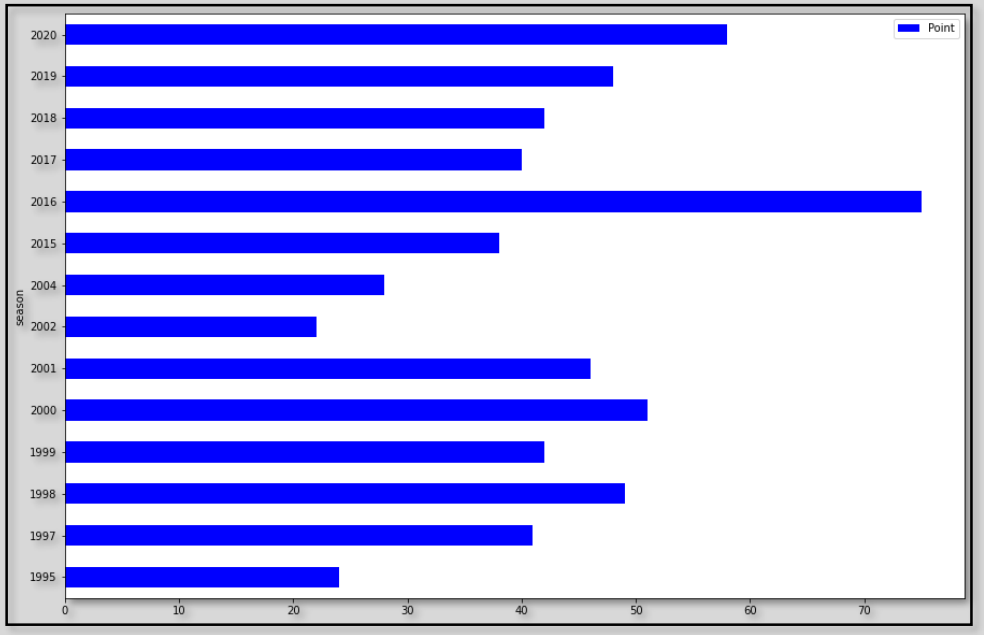



```
best_season.describe()
```

	Point	season
count	27.000000	27.000000
mean	87.629630	2007.000000
std	6.445918	7.937254
min	75.000000	1994.000000
25%	82.500000	2000.500000
50%	89.000000	2007.000000
75%	91.000000	2013.500000
max	100.000000	2020.000000

- 시즌별 승점을 plot.barh로 출력
- pandas describe() 로 통계 출력
- 평균 87 ~ 89점의 팀들이 우승, 제일 적은 점수는 75점이다.

구현 - EPL



	TEAM	Point	season
1	Leicester	24	1995
1	Leicester	41	1997
1	Leicester	49	1998
1	Leicester	42	1999
1	Leicester	51	2000
1	Leicester	46	2001
1	Leicester	22	2002
1	Leicester	28	2004
1	Leicester	38	2015
1	Leicester	75	2016
1	Leicester	40	2017
1	Leicester	42	2018
1	Leicester	48	2019
1	Leicester	58	2020

```
best_season.describe()
```

	Point	season
count	27.000000	27.000000
mean	87.629630	2007.000000
std	6.445918	7.937254
min	75.000000	1994.000000
25%	82.500000	2000.500000
50%	89.000000	2007.000000
75%	91.000000	2013.500000
max	100.000000	2020.000000

- 레스터 시티의 역대 승점 기록들을 출력
- 우승을 했던 2016시즌 외의 기록들은 처참함
- 중간중간 비어있는 년도는 2부리그로 강등되어 자료가 없음.
- 오른쪽 자료는 역대 우승팀 평균인데 평소 우승 최소 값인 75점도 못미쳤는데 2016년엔 최소값을 넘은 81점이다.
- 승점 81점인데 각 시즌별로 4~6점씩 다른 팀의 승점으로 들어감…

구현 - EPL

```
for s, t in zip(season, best_attack):#season과 best_attack리스트의 값을 같은 인덱스 짝 별로 묶어 튜플에 담아 반환
  goal = {}
  shot= {}
  shot_target = {}
  s_data = EPL_data.loc[(EPL_data["Season"]==s),:]
  #조건 1 | 조건 2 / HomteTeam 이거나 AwayTeam의 열을 모두 가져와라
  s_data = s_data.loc[(s_data["HomeTeam"]==t)|(s_data["AwayTeam"]==t),:]

  for j in s_data.index:
    if s_data["HomeTeam"][j] == t:#홈 팀일때
      #홈에서의 골
      if s_data["HomeTeam"][j] in goal:
        goal[s_data["HomeTeam"][j]] = goal[s_data["HomeTeam"][j]] + s_data["FTHG"][j]
      else:
        goal[s_data["HomeTeam"][j]] = s_data["FTHG"][j]
      #슈팅
      if s_data["HomeTeam"][j] in shot:#홈에서의 슈팅
        shot[s_data["HomeTeam"][j]] = shot[s_data["HomeTeam"][j]] + s_data["HS"][j]
      else:
        shot[s_data["HomeTeam"][j]] = s_data["HS"][j]
      #유효슈팅
      if s_data["HomeTeam"][j] in shot_target:#홈에서의 유효슈팅
        shot_target[s_data["HomeTeam"][j]] = shot_target[s_data["HomeTeam"][j]] + s_data["HST"][j]
      else:
        shot_target[s_data["HomeTeam"][j]] = s_data["HST"][j]

    elif s_data["AwayTeam"][j] == t:#어웨이 팀일때
      #어웨이에서의 골
      if s_data["AwayTeam"][j] in goal:
        goal[s_data["AwayTeam"][j]] = goal[s_data["AwayTeam"][j]] + s_data["FTAG"][j]
      else:
        goal[s_data["AwayTeam"][j]] = s_data["FTAG"][j]
      #어웨이에서의 슈팅
      if s_data["AwayTeam"][j] in shot:
        shot[s_data["AwayTeam"][j]] = shot[s_data["AwayTeam"][j]] + s_data["AS"][j]
      else:
        shot[s_data["AwayTeam"][j]] = s_data["AS"][j]
        #어웨이 유효슈팅
      if s_data["AwayTeam"][j] in shot_target:
        shot_target[s_data["AwayTeam"][j]] = shot_target[s_data["AwayTeam"][j]] + s_data["AST"][j]
      else:
        shot_target[s_data["AwayTeam"][j]] = s_data["AST"][j]
```

- 앞서 승점을 구한 코드들과 굉장히 유사하다.
- goal, shot, shot_target 딕셔너리에 각각 조건문에 맞는 값들을 추가한다.

구현 - EPL

0	Man United	80	1994	Man United	NaN	NaN
1	Blackburn	80	1995	Blackburn	NaN	NaN
2	Man United	73	1996	Man United	NaN	NaN
3	Man United	76	1997	Man United	NaN	NaN
4	Arsenal	68	1998	Arsenal	NaN	NaN
5	Man United	80	1999	Man United	NaN	NaN
6	Man United	97	2000	Man United	NaN	NaN
7	Man United	79	2001	Man United	556.0	264.0
8	Arsenal	79	2002	Arsenal	518.0	261.0
9	Man United	74	2003	Man United	531.0	308.0
10	Arsenal	73	2004	Arsenal	471.0	273.0
11	Chelsea	72	2005	Chelsea	533.0	304.0
12	Chelsea	72	2006	Chelsea	514.0	277.0
13	Man United	83	2007	Man United	627.0	342.0
14	Man United	80	2008	Man United	620.0	353.0
15	Man United	68	2009	Man United	621.0	376.0
16	Chelsea	103	2010	Chelsea	697.0	388.0
17	Man United	78	2011	Man United	543.0	310.0
18	Man United	89	2012	Man United	592.0	369.0
19	Man United	86	2013	Man United	512.0	299.0
20	Man City	102	2014	Man City	673.0	238.0
21	Chelsea	73	2015	Chelsea	563.0	210.0
22	Leicester	68	2016	Leicester	523.0	181.0
23	Chelsea	85	2017	Chelsea	580.0	204.0
24	Man City	106	2018	Man City	664.0	261.0
25	Man City	95	2019	Man City	683.0	260.0

	Goal	season	Shot	Shot_target
count	27.000000	27.000000	20.000000	20.00000
mean	81.629630	2007.000000	580.550000	285.45000
std	10.870384	7.937254	65.102733	58.97321
min	68.000000	1994.000000	471.000000	181.00000
25%	73.000000	2000.500000	529.000000	254.50000
50%	80.000000	2007.000000	571.500000	275.00000
75%	85.500000	2013.500000	622.500000	318.00000
max	106.000000	2020.000000	697.000000	388.00000

- 골, 슛, 유효슈팅 출력 결과
- 2001년부터 슛, 유효슈팅 측정해서 NaN으로 나옴
- .describe()으로 출력 결과
- 평균 80골 570~580슛 275~284유효슈팅을 기록

구현 - EPL

	TEAM_x	Goal	season	Shot	TEAM_y	Shot_target
0	Leicester	45	1995	NaN	Leicester	NaN
1	Leicester	46	1997	NaN	Leicester	NaN
2	Leicester	51	1998	NaN	Leicester	NaN
3	Leicester	40	1999	NaN	Leicester	NaN
4	Leicester	55	2000	NaN	Leicester	NaN
5	Leicester	39	2001	326.0	Leicester	159.0
6	Leicester	30	2002	304.0	Leicester	158.0
7	Leicester	48	2004	366.0	Leicester	214.0
8	Man United	80	2008	620.0	Man United	353.0
9	Man United	68	2009	621.0	Man United	376.0
10	Chelsea	103	2010	697.0	Chelsea	388.0
11	Man United	78	2011	543.0	Man United	310.0
12	Man United	89	2012	592.0	Man United	369.0
13	Man United	86	2013	512.0	Man United	299.0
14	Man City	102	2014	673.0	Man City	238.0
15	Chelsea	73	2015	563.0	Chelsea	210.0
16	Leicester	68	2016	523.0	Leicester	181.0
17	Chelsea	85	2017	580.0	Chelsea	204.0
18	Man City	106	2018	664.0	Man City	261.0
19	Man City	95	2019	683.0	Man City	260.0
20	Liverpool	85	2020	590.0	Liverpool	231.0

- 레스터 시티의 골, 슛, 유효슈팅 출력을 했다.
- 08년 이후부턴 강등당해서 데이터가 없는데 우승 팀들의 값들이 들어감…
- 통계치를 출력하면 우승팀들의 값들도 들어가있어서 출력을 못했다.
- 우승 시즌을 보면 골, 슛 그리고 유효슈팅이 다른 우승팀들보다는 적은 것을 볼 수 있다.

결과 - EPL

	TEAM	Point	season
1	Leicester	81	2016
2	Arsenal	71	2016
3	Tottenham	70	2016
4	Man United	66	2016
5	Man City	66	2016
6	Southampton	63	2016
7	West Ham	62	2016
8	Liverpool	60	2016
9	Stoke	51	2016
10	Chelsea	50	2016
11	Swansea	47	2016
12	Everton	47	2016
13	Watford	45	2016
14	West Brom	43	2016
15	Bournemouth	42	2016
16	Crystal Palace	42	2016
17	Sunderland	39	2016
18	Newcastle	37	2016
19	Norwich	34	2016
20	Aston Villa	17	2016

- 2016시즌 기록을 출력을 했다.
- 앞서 보았던 경기력 데이터를 본 결과 맨맨아첼뿐만 아니라 다른 모든 팀들의 부진이 있었고, 그 속에서 레스터 시티는 승점을 차곡차곡 쌓아 올려 우승을 할 수 있었다.

감사합니다