MINI _ PROJECT

Riot API 활용 데이터 처리

LOL 데이터 처리 미니 프로젝트

문정환 이세인 이주찬

Contents

01

데이터 수집 과정

- 데이터 수집을 위한 함수
- 테이블 구성 및 저장 과정

데이터 수집 과정

데이터 수집을 위한 함수

```python
def get_rawdata(tier):
    division_list = ['I', 'II', 'III', 'IV']
    lst = []
    page = random.randrange(1, 99)
    print('get_SummonerName....from', page, 'page')
    for division in tqdm(division_list):
        url = f'https://kr.api.riotgames.com/lol/league/v4/entries/RANKED_SOLO_5x5/{tier}/{division}?page={page}&api_key={riot_api_key}'
        res = requests.get(url).json()
        lst += random.sample(res, 5)
    summonerName_lst = list(map(lambda x: x['summonerName'], lst))

    print('get puuid.....')
    puuid_lst = []
    for i in tqdm(summonerName_lst):
        try:
            puuid_lst.append(get_puuid(i))
        except:
            print(i)
            continue

    print('get_match_id........')
    match_id_lst = []
    for j in tqdm(puuid_lst):
        match_id_lst += get_match_id(j, 3)

    print('get matches & timeline ......')
    df_create = []
    for match_id in tqdm(match_id_lst):
        matches, timeline = get_matches_timelines(match_id)
        time.sleep(2)
        df_create.append([match_id, matches, timeline])

    df = pd.DataFrame(df_create, columns=['match_id', 'matches', 'timeline'])
    print('complete')

    return df
```

함수

def get_rawdata(tier)

Riot Api를 이용한 티어리스트 불러오기
(summonerName_lst 저장)

계정 고유의 아이디 puuid 추출
(puuid_lst 저장)

puuid를 활용! match_id (게임) 추출

match_id를 활용 하여 게임 정보 추출
(matches, timeline)

def get_match_timeline_df(df)

```python
def get_match_timeline_df(df):
    df_creater = []
    info = df.iloc[0].matches['info']['participants']
    columns = ['match_id', 'teamBaronKills', 'gameDuration', 'gameVersion', 'summonerName', 'summonerLevel',
               'participantId'
        , 'championName', 'teamId', 'teamPosition', 'win', 'kills', 'deaths', 'assists'
        , 'totalDamageDealtToChampions', 'totalDamageTaken'
        , 'firstBloodKill', 'cs5', 'cs10', 'cs15', 'cs20', 'cs25'
        , 'ad5', 'ad10', 'ad15', 'ad20', 'ad25'
        , 'ap5', 'ap10', 'ap15', 'ap20', 'ap25']

    print('소환사 스탯 생성중....')
    for i in tqdm(range(len(df))):
        try:
            if df.iloc[i].matches['info']['gameDuration'] > 900:
                # matches 관련된 데이터
                for j in range(len(df.iloc[0].matches['info']['participants'])):
                    lst = []
                    lst.append(df.iloc[i].match_id)
                    if df.iloc[i].matches['info']['participants'][0]['win'] == True:
                        lst.append(df.iloc[i].matches['info']['participants'][0]['challenges']['teamBaronKills'])
                    elif df.iloc[i].matches['info']['participants'][5]['win'] == True:
                        lst.append(df.iloc[i].matches['info']['participants'][5]['challenges']['teamBaronKills'])

                    lst.append(df.iloc[i].matches['info']['gameDuration'])
                    lst.append(df.iloc[i].matches['info']['gameVersion'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['summonerName'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['summonerLevel'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['participantId'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['championName'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['teamId'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['teamPosition'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['win'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['kills'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['deaths'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['assists'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['totalDamageDealtToChampions'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['totalDamageTaken'])
                    lst.append(df.iloc[i].matches['info']['participants'][j]['firstBloodKill'])

                    # 미니언,ad,ap
                    for k in range(5, 26, 5):
                        try:
                            lst.append(df.iloc[i].timeline['info']['frames'][k]['participantFrames'][str(j + 1)][
                                           'minionsKilled'] +
                                       df.iloc[i].timeline['info']['frames'][k]['participantFrames'][str(j + 1)][
                                           'jungleMinionsKilled'])
                        except:
                            lst.append(0)
                    for m in range(5, 26, 5):
                        try:
                            lst.append(df.iloc[i].timeline['info']['frames'][m]['participantFrames'][str(j + 1)][
                                           'championStats']['attackDamage'])
                        except:
                            lst.append(0)
                    for n in range(5, 26, 5):
                        try:
                            lst.append(df.iloc[i].timeline['info']['frames'][n]['participantFrames'][str(j + 1)][
                                           'championStats']['abilityPower'])
                        except:
                            lst.append(0)
                    df_creater.append(lst)
                    lol_df = pd.DataFrame(df_creater, columns=columns)
        except:
            print(i)
            continue
    print("완료!")
    return lol_df
```

데이터 수집 과정

Create table

```python
def get_rawdata(tier):
    division_list = ['I', 'II', 'III', 'IV']
    lst = []
    page = random.randrange(1, 99)
    print('get_SummonerName....from', page, 'page')
    for division in tqdm(division_list):
        url = f'https://kr.api.riotgames.com/lol/league/v4/entries/RANKED_SOLO_5x5/{tier}/{division}?page={page}&api_key={riot_api_key}'
        res = requests.get(url).json()
        lst += random.sample(res, 5)
    summonerName_lst = list(map(lambda x: x['summonerName'], lst))

    print('get puuid.....')
    puuid_lst = []
    for i in tqdm(summonerName_lst):
        try:
            puuid_lst.append(get_puuid(i))
        except:
            print(i)
            continue

    print('get_match_id........')
    match_id_lst = []
    for j in tqdm(puuid_lst):
        match_id_lst += get_match_id(j, 3)

    print('get matches & timeline ......')
    df_create = []
    for match_id in tqdm(match_id_lst):
        matches, timeline = get_matches_timelines(match_id)
        time.sleep(2)
        df_create.append([match_id, matches, timeline])

    df = pd.DataFrame(df_create, columns=['match_id', 'matches', 'timeline'])
    print('complete')

    return df
```

테이블 생성

Riot Api를 이용한 티어리스트 불러오기
(summonerName_lst 저장)

계정 고유의 아이디 puuid 추출
(puuid_lst 저장)

puuid를 활용! match_id (게임) 추출

match_id를 활용 하여 게임 정보 추출
(matches, timeline)

1. 테이블 생성

테이블 생성의 처리 과정

Create tabel

쿼리문

```
query = """
CREATE TABLE team_lol(match_id varchar(20), teamBaronKills int, gameDuration int,
gameVersion varchar(20), summonerName varchar(50),
summonerLevel int, participantId int, championName varchar(50), teamId int,
teamPosition varchar(20), win bool, kills int, deaths int, assists int,
totalDamageDealtToChampions int,
totalDamageTaken int,firstBloodKill bool, cs5 int, cs10 int, cs15 int,
cs20 int, cs25 int, ad5 int, ad10 int, ad15 int, ad20 int,
ad25 int,ap5 int, ap10 int, ap15 int, ap20 int, ap25 int,
primary key(match_id, participantId))
"""
#생성 쿼리문
```

생 성

```
conn = tu.connect_mysql('icia')
tu.mysql_execute(query,conn)
conn.close()
```

함 수

```
def connect_mysql(db):
    conn = pymysql.connect(host='localhost', user='root', password='1234', db=db, charset='utf8')
    return conn


new *
def mysql_execute(query, conn):
    cursor_mysql = conn.cursor()
    cursor_mysql.execute(query)
    result = cursor_mysql.fetchall()
    return result
```

2. Insert 함수

테이블 생성의 처리 과정

def
insert_mysql

함 수

```
def insert_mysql(x, conn):
    query = (
        f'insert into team_lol(match_id,teamBaronKills,gameDuration,gameVersion,summonerName,summonerLevel,'
        f'participantId,championName,teamId,teamPosition,'
        f'win,kills,deaths,assists,totalDamageDealtToChampions,totalDamageTaken,firstBloodKill,cs5,cs10,cs15,cs20,cs25,'
        f'ad5,ad10,ad15,ad20,ad25,ap5,ap10,ap15,ap20,ap25)'
        f'values({repr(x.match_id)},{x.teamBaronKills},{x.gameDuration},{repr(x.gameVersion)},{repr(x.summonerName)},'
        f'{x.summonerLevel},{x.participantId},{repr(x.championName)},{x.teamId},'
        f'{repr(x.teamPosition)},{x.win},{x.kills},{x.deaths},{x.assists},'
        f'{x.totalDamageDealtToChampions},{x.totalDamageTaken},{x.firstBloodKill},{x.cs5},{x.cs10},{x.cs15},{x.cs20},{x.cs25},{x.ad5},'
        f'{x.ad10},{x.ad15},{x.ad20},{x.ad25},{x.ap5},{x.ap10},{x.ap15},{x.ap20},{x.ap25})'
        f'ON DUPLICATE KEY UPDATE '
        f'match_id  = {repr(x.match_id)},teamBaronKills= {x.teamBaronKills},gameDuration = {x.gameDuration},gameVersion = {repr(x.gameVersion)},summonerName = {repr(x.summonerName)}'
        f',summonerLevel = {x.summonerLevel}, participantId = {x.participantId},championName = {repr(x.championName)},teamId = {x.teamId},teamPosition = {repr(x.teamPosition)}'
        f',win = {x.win}, kills= {x.kills},deaths = {x.deaths},assists = {x.assists},totalDamageDealtToChampions = {x.totalDamageDealtToChampions}'
        f',totalDamageTaken = {x.totalDamageTaken},firstBloodKill={x.firstBloodKill},cs5 ={x.cs5},cs10 ={x.cs10},cs15 ={x.cs15},cs20 ={x.cs20},cs25 ={x.cs25}'
        f',ad5 ={x.ad5},ad10 ={x.ad10},ad15 ={x.ad15},ad20 ={x.ad20},ad25 ={x.ad25},ap5 ={x.ap5},ap10 ={x.ap10},ap15 ={x.ap15},ap20 ={x.ap20},ap25={x.ap25}'
    )
    mysql_execute(query, conn)
    return
```

3. DB 저장

테이블 생성의 처리 과정

insert table

쿼리문

```
tier = 'GOLD'
for i in range(20):
    try:
        raw_data = tu.get_rawdata(tier)
        df = tu.get_match_timeline_df(raw_data)

        conn = tu.connect_mysql('icia')
        df.progress_apply(lambda x:tu.insert_mysql(x, conn),axis = 1)
        conn.commit()
        conn.close()
        print(f'{i+2}번째 반복문 실행 중')
    except Exception as e:
        print(f'{e}의 원인으로 insert 실패')
```

실 행

```
get puuid.....
100%|| 20/20 [00:05<00:00,  3.47it/s]
get_match_id........
100%|| 20/20 [00:05<00:00,  3.55it/s]
get matches & timeline ......
100%|| 60/60 [02:50<00:00,  2.85s/it]
complete
소환사 스텟 생성중....
100%|| 60/60 [00:04<00:00, 13.52it/s]
완료!
```

02

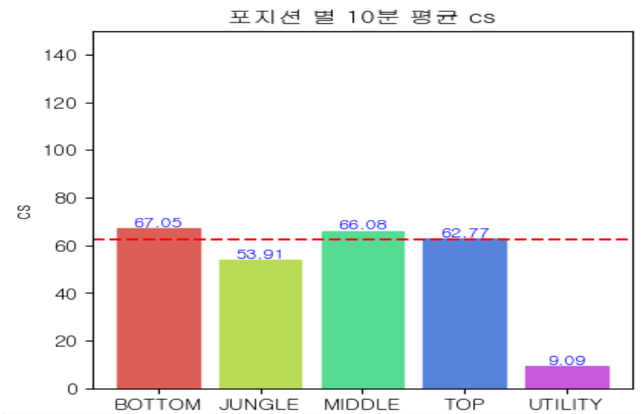
데이터 시각화

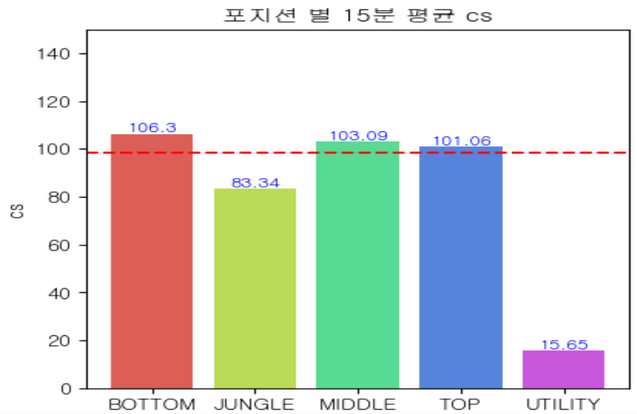
- 포지션 별 평균 CS
- AD, AP계수에 따른 CS
- 챔피언 별 AD, AP,별 CS 관계
- 바론 처치 횟수 와 승리 관계
- 선취점 취득과 승리 관계

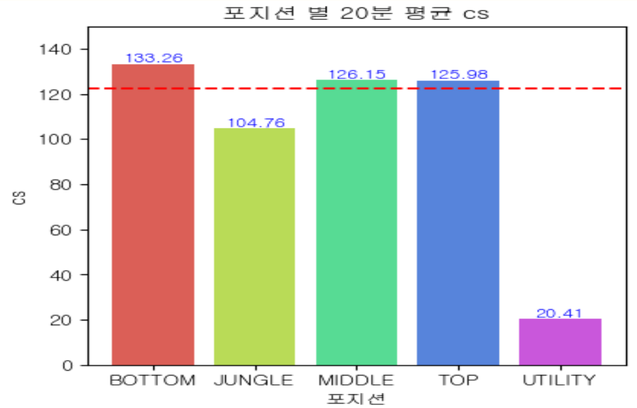
데이터 시각화

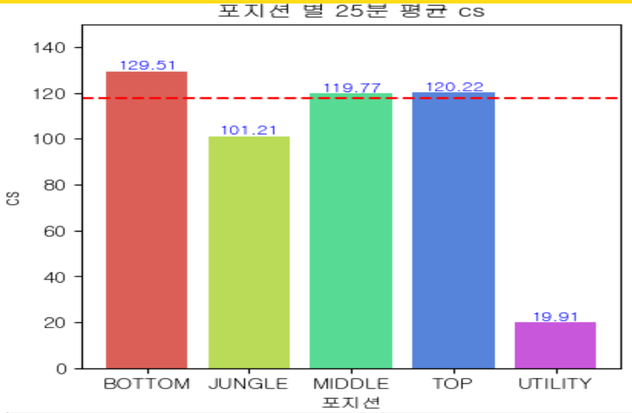
포지션 별 평균 cs

10분 → 15분 → 20분 → 25분










```
plt.figure(figsize=(10,10))
x = df_cs_tp25.teamPosition
y = df_cs_tp25['cs10']
colors = sns.color_palette('hls',len(x))

#그래프 그리기
plt.subplot(2,2,1)
plt.bar(x,y,color= colors,edgecolor = colors)
plt.xlabel('포지션')
plt.ylabel('cs')
plt.ylim([0, 150])
plt.title('포지션 별 10분 평균 cs')

#평균 선
mean_val = df_b_j_m_t['cs10'].mean()
plt.axhline(y=mean_val , color ='r',linestyle = 'dashed')

#그래프 값 달기
for i, v in enumerate(x):
    plt.text(v, y[i],str(y[i]),
             fontsize=9,
             color="blue",
             horizontalalignment='center',
             verticalalignment='bottom')
```

```
x = df_cs_tp25.teamPosition
y = df_cs_tp25['cs15']
colors = sns.color_palette('hls',len(x))
#그래프 그리기
plt.subplot(2,2,2)
plt.bar(x,y,color=colors)
plt.xlabel('포지션')
plt.ylabel('cs')
plt.ylim([0, 150])
plt.title('포지션 별 15분 평균 cs')

#평균 선
mean_val = df_b_j_m_t['cs15'].mean()
plt.axhline(y=mean_val , color ='r',linestyle = 'dashed')

#그래프 값 달기
for i, v in enumerate(x):
    plt.text(v, y[i],str(y[i]),
             fontsize=9,
             color="blue",
             horizontalalignment='center',
             verticalalignment='bottom')
```

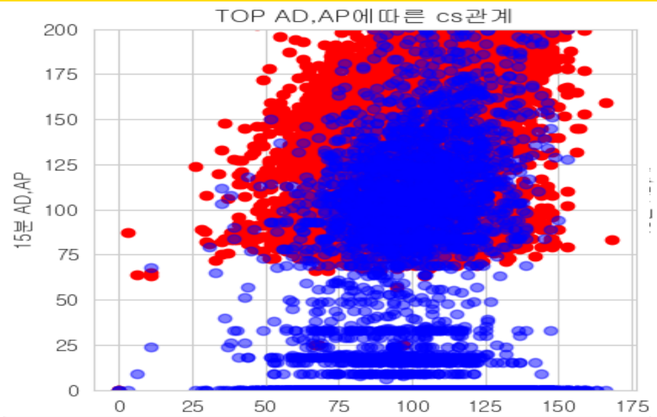
```
x = df_cs_tp25.teamPosition
y = df_cs_tp25['cs20']
colors = sns.color_palette('hls',len(x))

#그래프 그리기
plt.subplot(2,2,3)
plt.bar(x,y,color=colors)
plt.xlabel('포지션')
plt.ylabel('cs')
plt.ylim([0, 150])
plt.title('포지션 별 20분 평균 cs')
#평균 선
mean_val = df_b_j_m_t['cs20'].mean()
plt.axhline(y=mean_val , color ='r',linestyle = 'dashed')

#그래프 값 달기
for i, v in enumerate(x):
    plt.text(v, y[i],str(y[i]),
             fontsize=9,
             color="blue",
             horizontalalignment='center',
             verticalalignment='bottom')
```

```
x = df_cs_tp25.teamPosition
y = df_cs_tp25['cs25']
colors = sns.color_palette('hls',len(x))


#그래프 그리기
plt.subplot(2,2,4)
plt.bar(x,y,color=colors)
plt.xlabel('포지션')
plt.ylabel('cs')
plt.ylim([0, 150])
plt.title('포지션 별 25분 평균 cs')

#평균 선
mean_val = df_b_j_m_t['cs25'].mean()
plt.axhline(y=mean_val , color ='r',linestyle = 'dashed

#그래프 값 달기
for i, v in enumerate(x):
    plt.text(v, y[i],str(y[i]),
             fontsize=9,
             color="blue",
             horizontalalignment='center',
             verticalalignment='bottom')


plt.show()
```

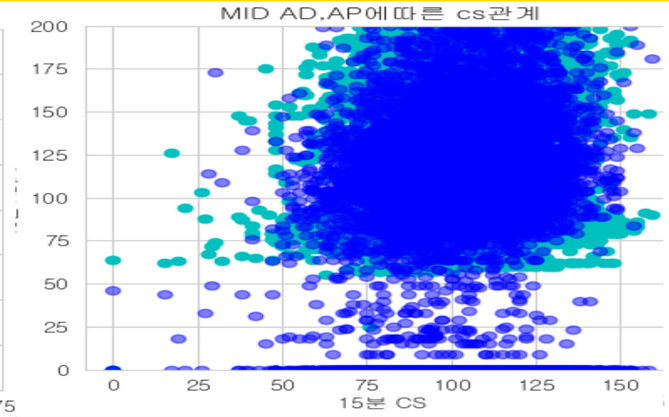
데이터 시각화

포지션 별 AD, AP계수에 따른 15분 cs 분포도

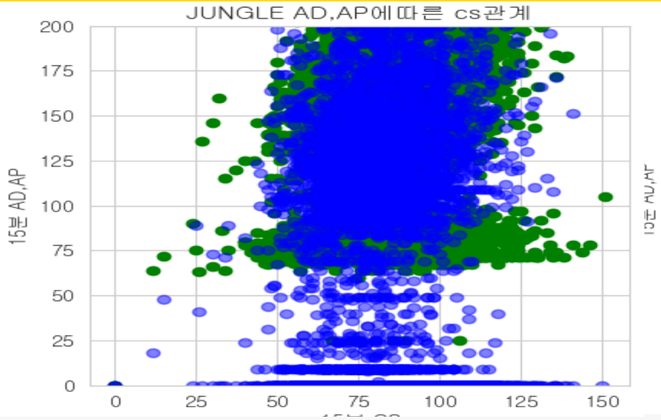
TOP → MID → JUG → BOT



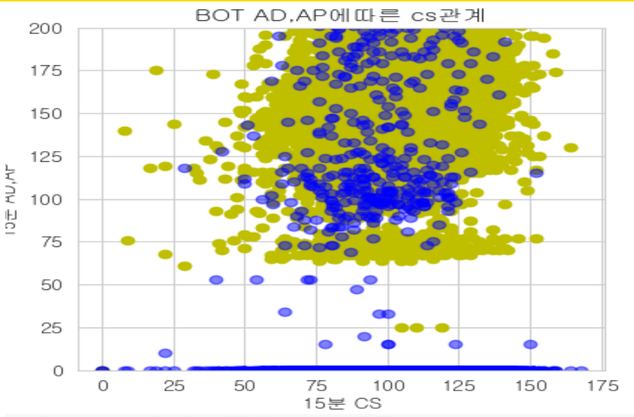

```
plt.figure(figsize=(10,10))
xdata = lol_df_top.cs15
ydata = lol_df_top.ad15
ydata2 = lol_df_top_ap.ap15

#산점도 그래프 그리기
plt.subplot(2,2,1)
plt.scatter(xdata,ydata,color = 'r')
plt.scatter(xdata,ydata2,color = 'b')
plt.xlabel('15분 CS')
plt.ylabel('15분 AD,AP')
plt.ylim([0,200])
plt.title('TOP AD,AP에따른 cs관계')
plt.grid(True)
```




```
xdata = lol_df_mid.cs15
ydata = lol_df_mid.ad15
ydata2 = lol_df_mid_ap.ap15

#산점도 그래프 그리기
plt.subplot(2,2,3)
plt.scatter(xdata,ydata,color = 'c')
plt.scatter(xdata,ydata2,color = 'b')
plt.xlabel('15분 CS')
plt.ylabel('15분 AD,AP')
plt.ylim([0,200])
plt.title('MID AD,AP에따른 cs관계')
plt.grid(True)
```




```
xdata = lol_df_jg.cs15
ydata = lol_df_jg.ad15
ydata2 = lol_df_jg_ap.ap15

#산점도 그래프 그리기
plt.subplot(2,2,2)
plt.scatter(xdata,ydata,color = 'g')
plt.scatter(xdata,ydata2,color = 'b')
plt.xlabel('15분 CS')
plt.ylabel('15분 AD,AP')
plt.ylim([0,200])
plt.title('JUNGLE AD,AP에따른 cs관계')
plt.grid(True)
```




```
xdata = lol_df_bot.cs15
ydata = lol_df_bot.ad15
ydata2 = lol_df_bot_ap.ap15

#산점도 그래프 그리기
plt.subplot(2,2,4)
plt.scatter(xdata,ydata,color = 'y')
plt.scatter(xdata,ydata2,color = 'b')
plt.xlabel('15분 CS')
plt.ylabel('15분 AD,AP')
plt.ylim([0,200])
plt.title('BOT AD,AP에따른 cs관계')
plt.grid(True)
```

데이터 시각화

전체 상관계수

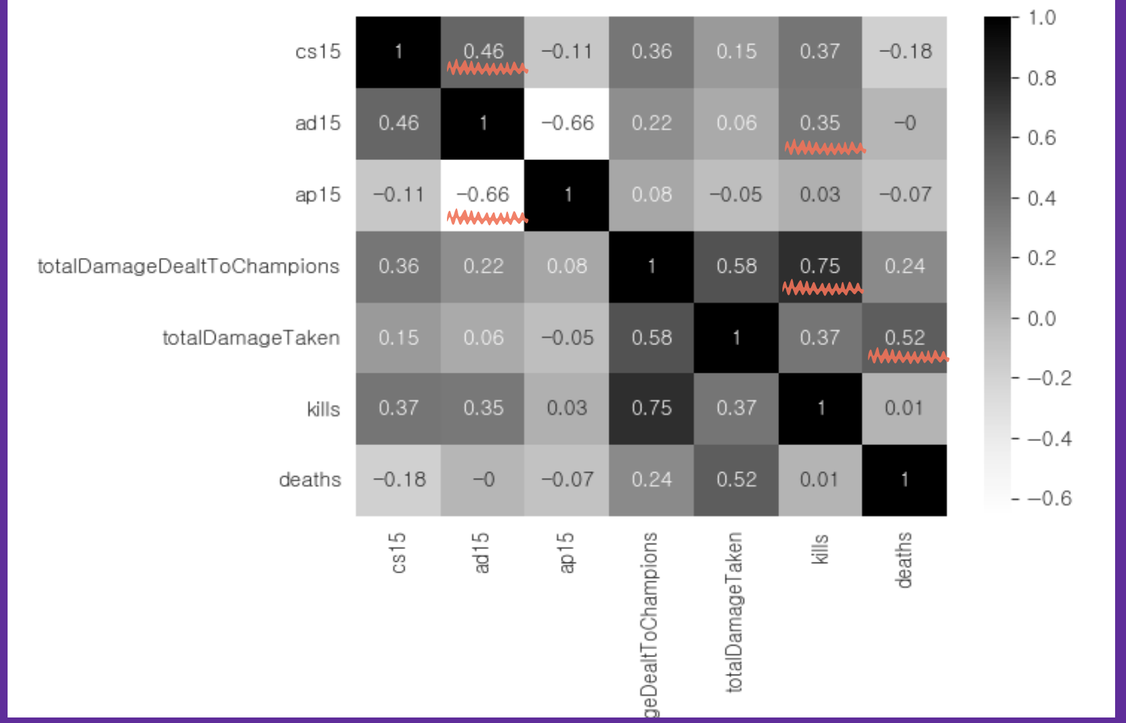
df_corr() ----------> 히트 맵

```
lol_df_lane.corr().round(2)
```

	cs15	ad15	ap15	totalDamageDealtToChampions	totalDamageTaken	kills	deaths
cs15	1.00	0.46	-0.11	0.36	0.15	0.37	-0.18
ad15	0.46	1.00	-0.66	0.22	0.06	0.35	-0.00
ap15	-0.11	-0.66	1.00	0.08	-0.05	0.03	-0.07
totalDamageDealtToChampions	0.36	0.22	0.08	1.00	0.58	0.75	0.24
totalDamageTaken	0.15	0.06	-0.05	0.58	1.00	0.37	0.52
kills	0.37	0.35	0.03	0.75	0.37	1.00	0.01
deaths	-0.18	-0.00	-0.07	0.24	0.52	0.01	1.00

```
sns.heatmap(lol_df_lane.corr().round(2),annot=True,cmap='Greys')
```

<AxesSubplot:>



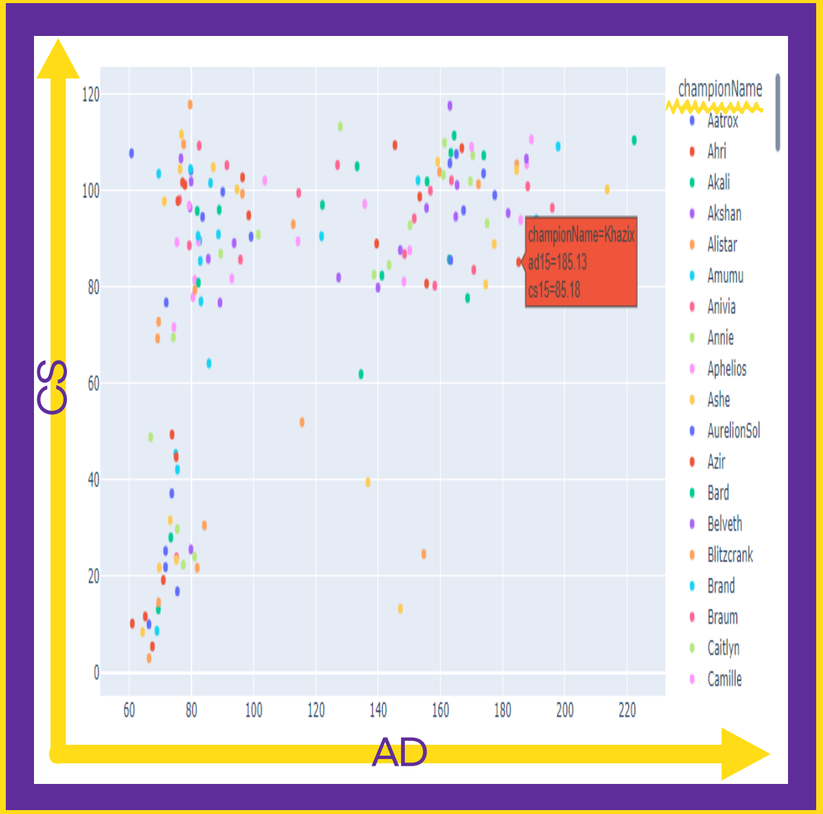
데이터 시각화

챔피언별 AD, AP 계수와 CS

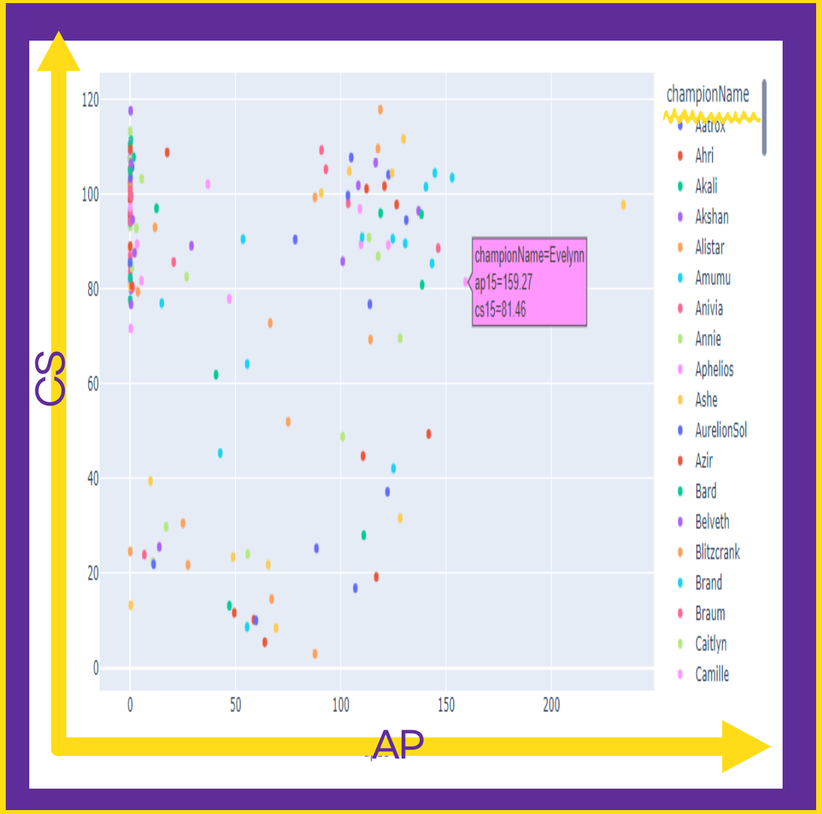
코드

```
#챔피언별 15분 CS, 15분 AD
fig = px.scatter(result_td,
                 x='ad15',
                 y='cs15',
                 color = 'championName'
)
plotly.offline.iplot(fig)

#챔피언별 15분 CS, 15분 AP
fig = px.scatter(result_td,
                 x='ap15',
                 y='cs15',
                 color = 'championName'
)
plotly.offline.iplot(fig)
```

AD



AP



데이터 시각화

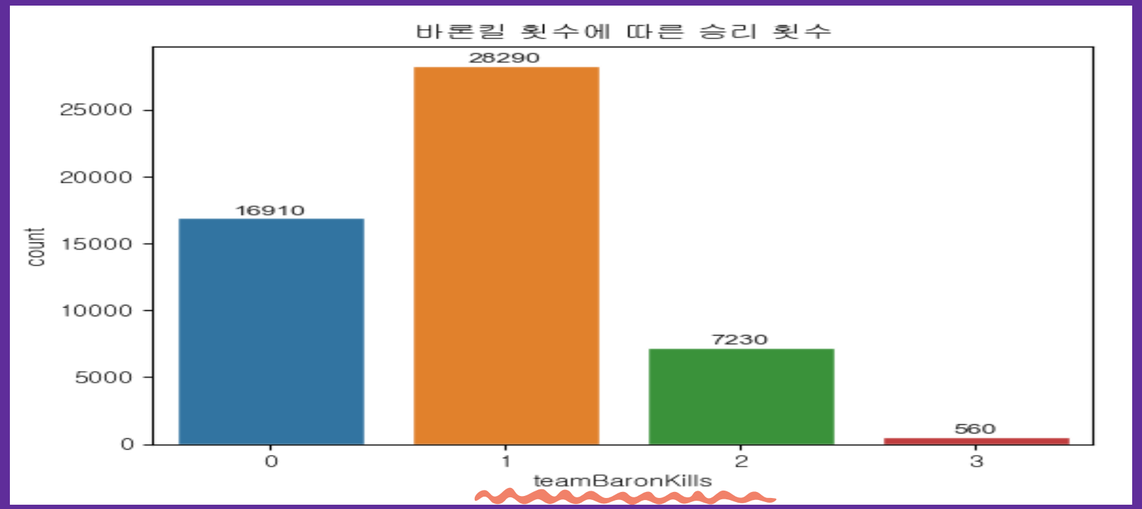
바론 처치 와 승리 관계

df_BaronKill

코드

```
plt.figure()
plt.title('바론킬 횟수에 따른 승리 횟수')
sns.set_style('whitegrid')
ax = sns.countplot(df_BaronKill['teamBaronKills'])
ax.bar_label(ax.containers[0])
```

시각화



데이터 시각화

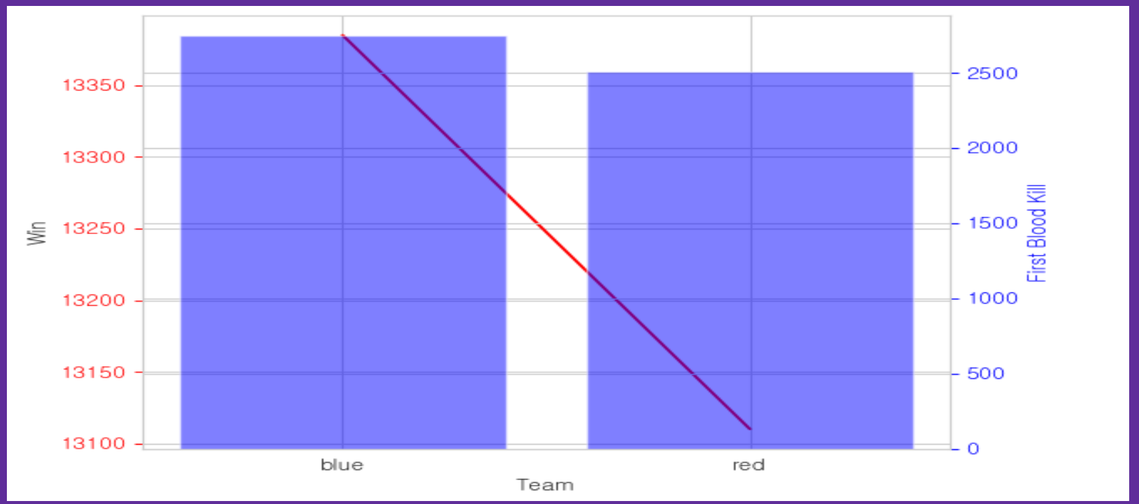
선취점 취득과 승리 관계

team_df

코드

```
fig, ax1 = plt.subplots()

TeamId = t_df['teamid']
Win = t_df['win']

ax1.plot(TeamId, Win, 'r')
ax1.set_xlabel('Team')

# Make the y-axis label, ticks and tick labels match the line color.
ax1.set_ylabel('Win')
ax1.tick_params('y', colors='r')

ax2 = ax1.twinx()
fb = t_df['firstBloodKill']

ax2.bar(TeamId, fb, color='b',alpha=0.5)
ax2.set_ylabel('First Blood Kill', color='b')
ax2.tick_params('y', colors='b')

fig.tight_layout()
plt.show()
```

시각화



데이터 시각화

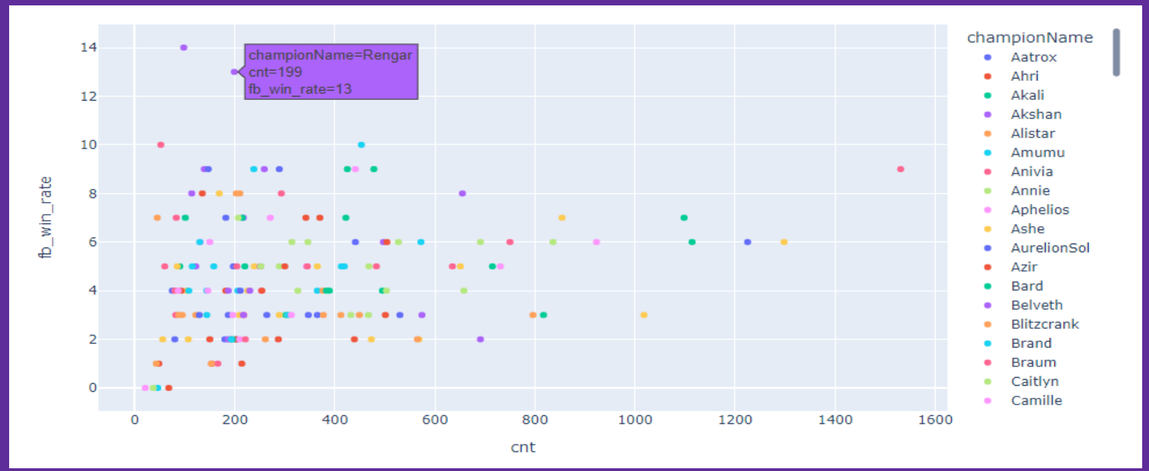
선취점 취득과 승리 관계

챔피언 별

코드

```
fig = px.scatter(result_df2, x = 'cnt',
                 y='fb_win_rate',
                 color = 'championName')
plotly.offline.iplot(fig)
```

시각화



QnA

질의응답