롤 API를 사용한 시각화

김시현

오건오

이태현

목차

CHAPTER 01

주제선정

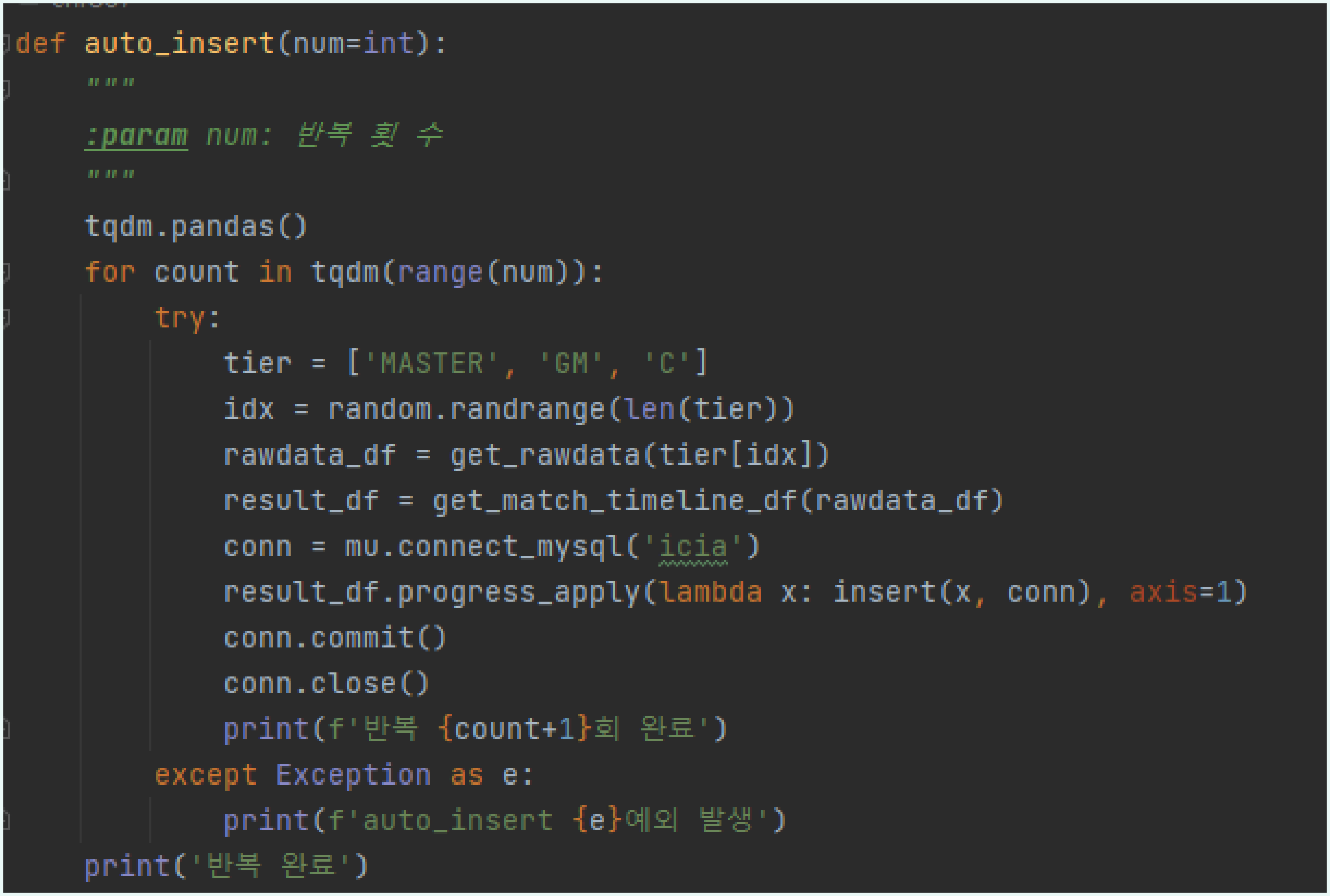
1.와드 설치, 파괴 횟수와 승패의 관계

2.퍼블, 포블, 첫용과 승률의 관계

3.많이 사용한 프로필 아이콘과 승률

CHAPTER 2

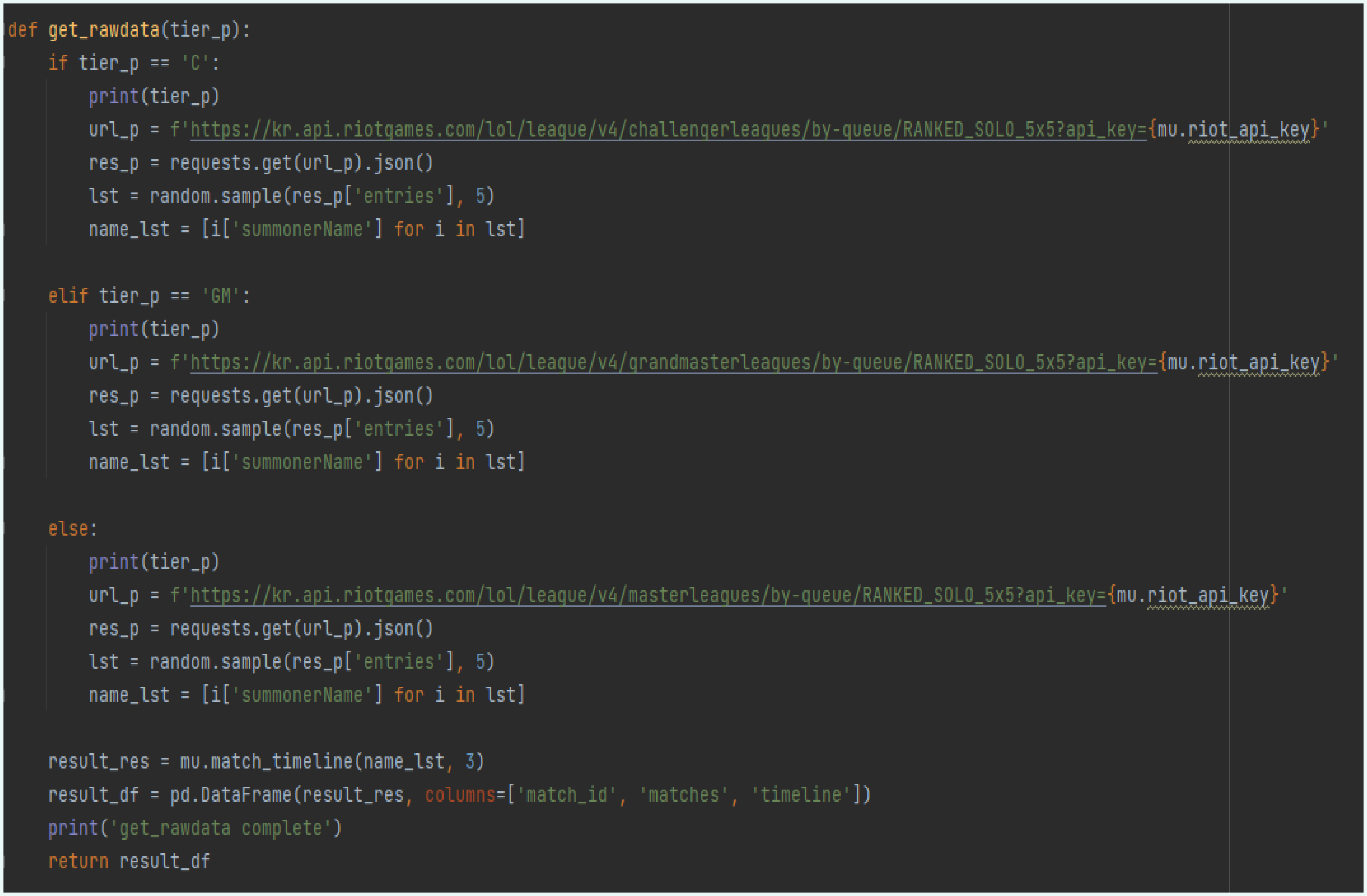
공통 모듈



```
def auto_insert(num=int):
    """
    :param num: 반복 횟 수
    """
    tqdm.pandas()
    for count in tqdm(range(num)):
        try:
            tier = ['MASTER', 'GM', 'C']
            idx = random.randrange(len(tier))
            rawdata_df = get_rawdata(tier[idx])
            result_df = get_match_timeline_df(rawdata_df)
            conn = mu.connect_mysql('icia')
            result_df.progress_apply(lambda x: insert(x, conn), axis=1)
            conn.commit()
            conn.close()
            print(f'반복 {count+1}회 완료')
        except Exception as e:
            print(f'auto_insert {e}에외 발생')
    print('반복 완료')
```

CHAPTER 2

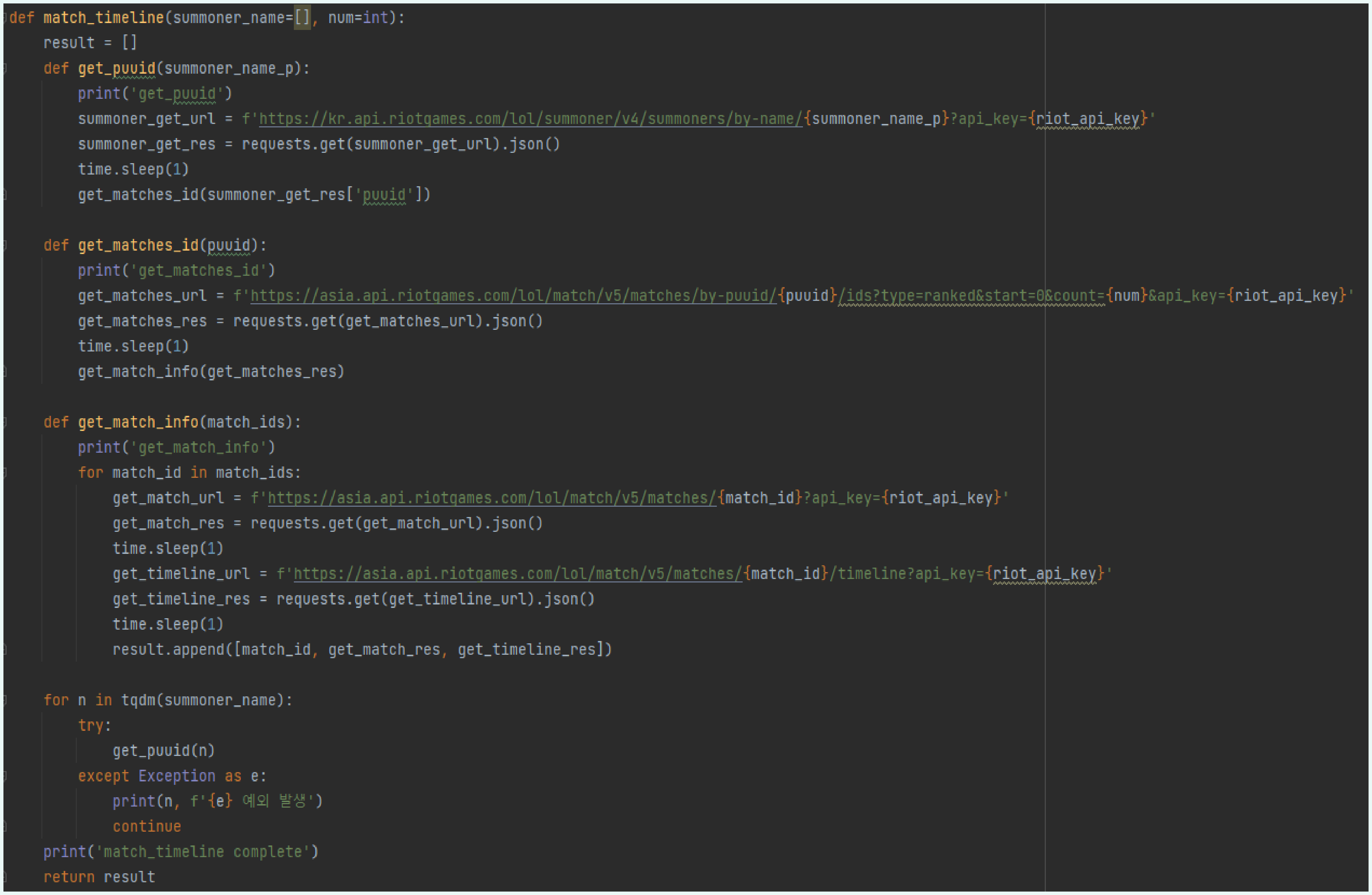
공통 모듈



```
def get_rawdata(tier_p):
    if tier_p == 'C':
        print(tier_p)
        url_p = f'https://kr.api.riotgames.com/lol/league/v4/challengerleagues/by-queue/RANKED_SOLO_5x5?api_key={mu.riot_api_key}'
        res_p = requests.get(url_p).json()
        lst = random.sample(res_p['entries'], 5)
        name_lst = [i['summonerName'] for i in lst]

    elif tier_p == 'GM':
        print(tier_p)
        url_p = f'https://kr.api.riotgames.com/lol/league/v4/grandmasterleagues/by-queue/RANKED_SOLO_5x5?api_key={mu.riot_api_key}'
        res_p = requests.get(url_p).json()
        lst = random.sample(res_p['entries'], 5)
        name_lst = [i['summonerName'] for i in lst]

    else:
        print(tier_p)
        url_p = f'https://kr.api.riotgames.com/lol/league/v4/masterleagues/by-queue/RANKED_SOLO_5x5?api_key={mu.riot_api_key}'
        res_p = requests.get(url_p).json()
        lst = random.sample(res_p['entries'], 5)
        name_lst = [i['summonerName'] for i in lst]

    result_res = mu.match_timeline(name_lst, 3)
    result_df = pd.DataFrame(result_res, columns=['match_id', 'matches', 'timeline'])
    print('get_rawdata complete')
    return result_df
```

CHAPTER 2

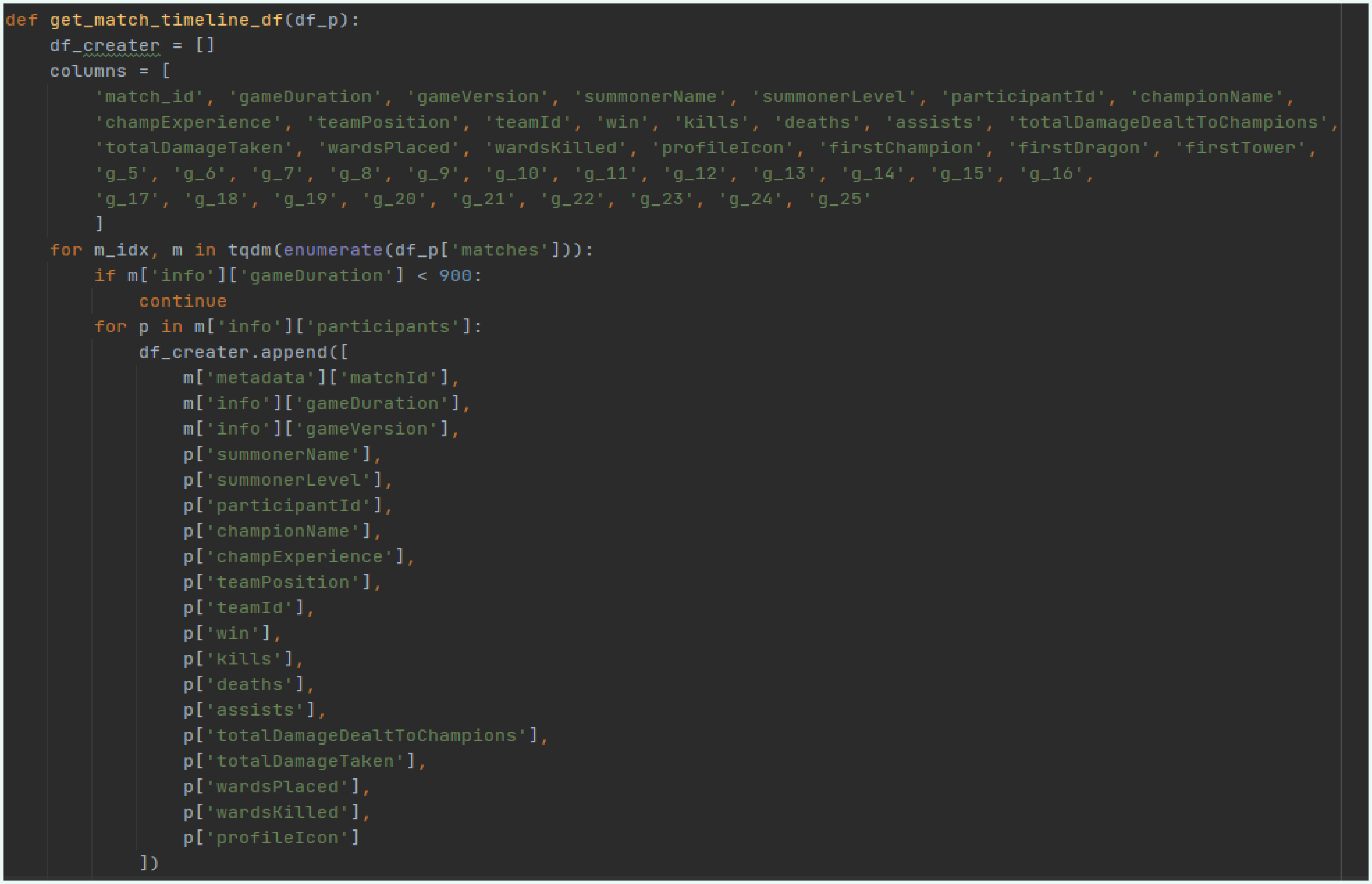
공통 모듈



```python
def match_timeline(summoner_name=[], num=int):
    result = []
    def get_puuid(summoner_name_p):
        print('get_puuid')
        summoner_get_url = f'https://kr.api.riotgames.com/lol/summoner/v4/summoners/by-name/{summoner_name_p}?api_key={riot_api_key}'
        summoner_get_res = requests.get(summoner_get_url).json()
        time.sleep(1)
        get_matches_id(summoner_get_res['puuid'])

    def get_matches_id(puuid):
        print('get_matches_id')
        get_matches_url = f'https://asia.api.riotgames.com/lol/match/v5/matches/by-puuid/{puuid}/ids?type=ranked&start=0&count={num}&api_key={riot_api_key}'
        get_matches_res = requests.get(get_matches_url).json()
        time.sleep(1)
        get_match_info(get_matches_res)

    def get_match_info(match_ids):
        print('get_match_info')
        for match_id in match_ids:
            get_match_url = f'https://asia.api.riotgames.com/lol/match/v5/matches/{match_id}?api_key={riot_api_key}'
            get_match_res = requests.get(get_match_url).json()
            time.sleep(1)
            get_timeline_url = f'https://asia.api.riotgames.com/lol/match/v5/matches/{match_id}/timeline?api_key={riot_api_key}'
            get_timeline_res = requests.get(get_timeline_url).json()
            time.sleep(1)
            result.append([match_id, get_match_res, get_timeline_res])

    for n in tqdm(summoner_name):
        try:
            get_puuid(n)
        except Exception as e:
            print(n, f'{e} 예외 발생')
            continue
    print('match_timeline complete')
    return result
```

CHAPTER 2

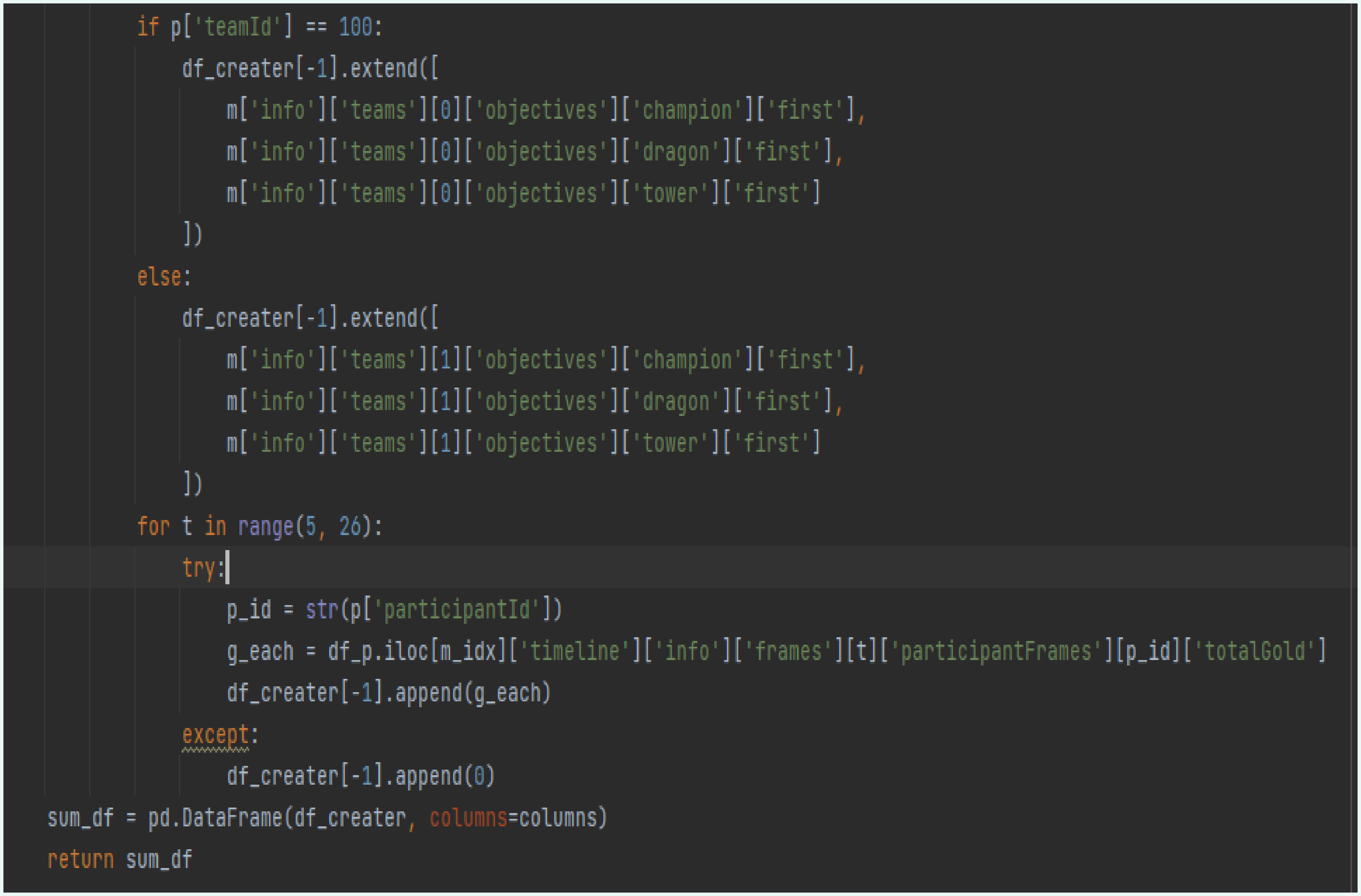
공통 모듈



```python
def get_match_timeline_df(df_p):
    df_creater = []
    columns = [
        'match_id', 'gameDuration', 'gameVersion', 'summonerName', 'summonerLevel', 'participantId', 'championName',
        'champExperience', 'teamPosition', 'teamId', 'win', 'kills', 'deaths', 'assists', 'totalDamageDealtToChampions',
        'totalDamageTaken', 'wardsPlaced', 'wardsKilled', 'profileIcon', 'firstChampion', 'firstDragon', 'firstTower',
        'g_5', 'g_6', 'g_7', 'g_8', 'g_9', 'g_10', 'g_11', 'g_12', 'g_13', 'g_14', 'g_15', 'g_16',
        'g_17', 'g_18', 'g_19', 'g_20', 'g_21', 'g_22', 'g_23', 'g_24', 'g_25'
        ]
    for m_idx, m in tqdm(enumerate(df_p['matches'])):
        if m['info']['gameDuration'] < 900:
            continue
        for p in m['info']['participants']:
            df_creater.append([
                m['metadata']['matchId'],
                m['info']['gameDuration'],
                m['info']['gameVersion'],
                p['summonerName'],
                p['summonerLevel'],
                p['participantId'],
                p['championName'],
                p['champExperience'],
                p['teamPosition'],
                p['teamId'],
                p['win'],
                p['kills'],
                p['deaths'],
                p['assists'],
                p['totalDamageDealtToChampions'],
                p['totalDamageTaken'],
                p['wardsPlaced'],
                p['wardsKilled'],
                p['profileIcon']
            ])
```

CHAPTER 2

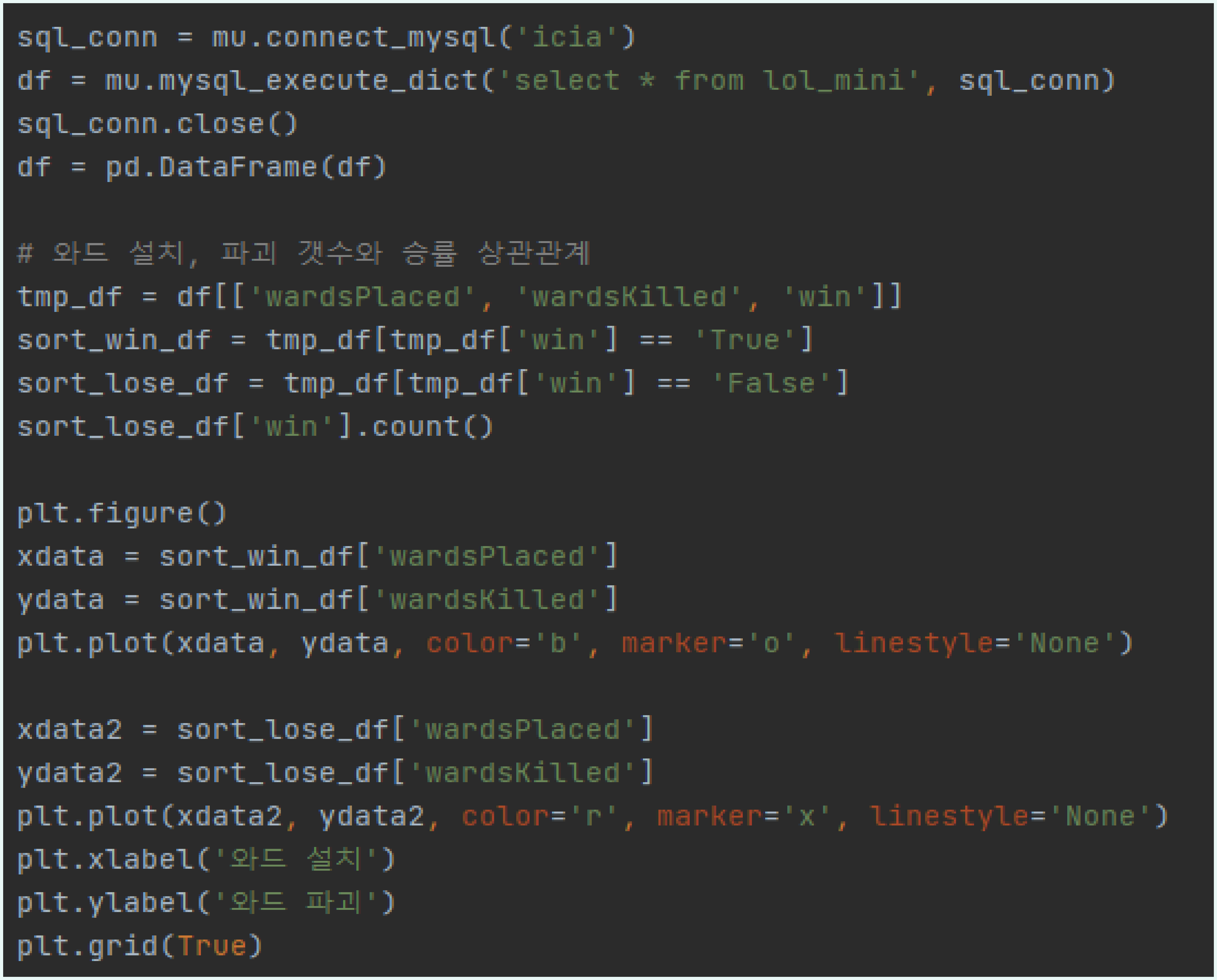
공통 모듈



```
            if p['teamId'] == 100:
                df_creater[-1].extend([
                    m['info']['teams'][0]['objectives']['champion']['first'],
                    m['info']['teams'][0]['objectives']['dragon']['first'],
                    m['info']['teams'][0]['objectives']['tower']['first']
                ])
            else:
                df_creater[-1].extend([
                    m['info']['teams'][1]['objectives']['champion']['first'],
                    m['info']['teams'][1]['objectives']['dragon']['first'],
                    m['info']['teams'][1]['objectives']['tower']['first']
                ])
            for t in range(5, 26):
                try:
                    p_id = str(p['participantId'])
                    g_each = df_p.iloc[m_idx]['timeline']['info']['frames'][t]['participantFrames'][p_id]['totalGold']
                    df_creater[-1].append(g_each)
                except:
                    df_creater[-1].append(0)
    sum_df = pd.DataFrame(df_creater, columns=columns)
    return sum_df
```

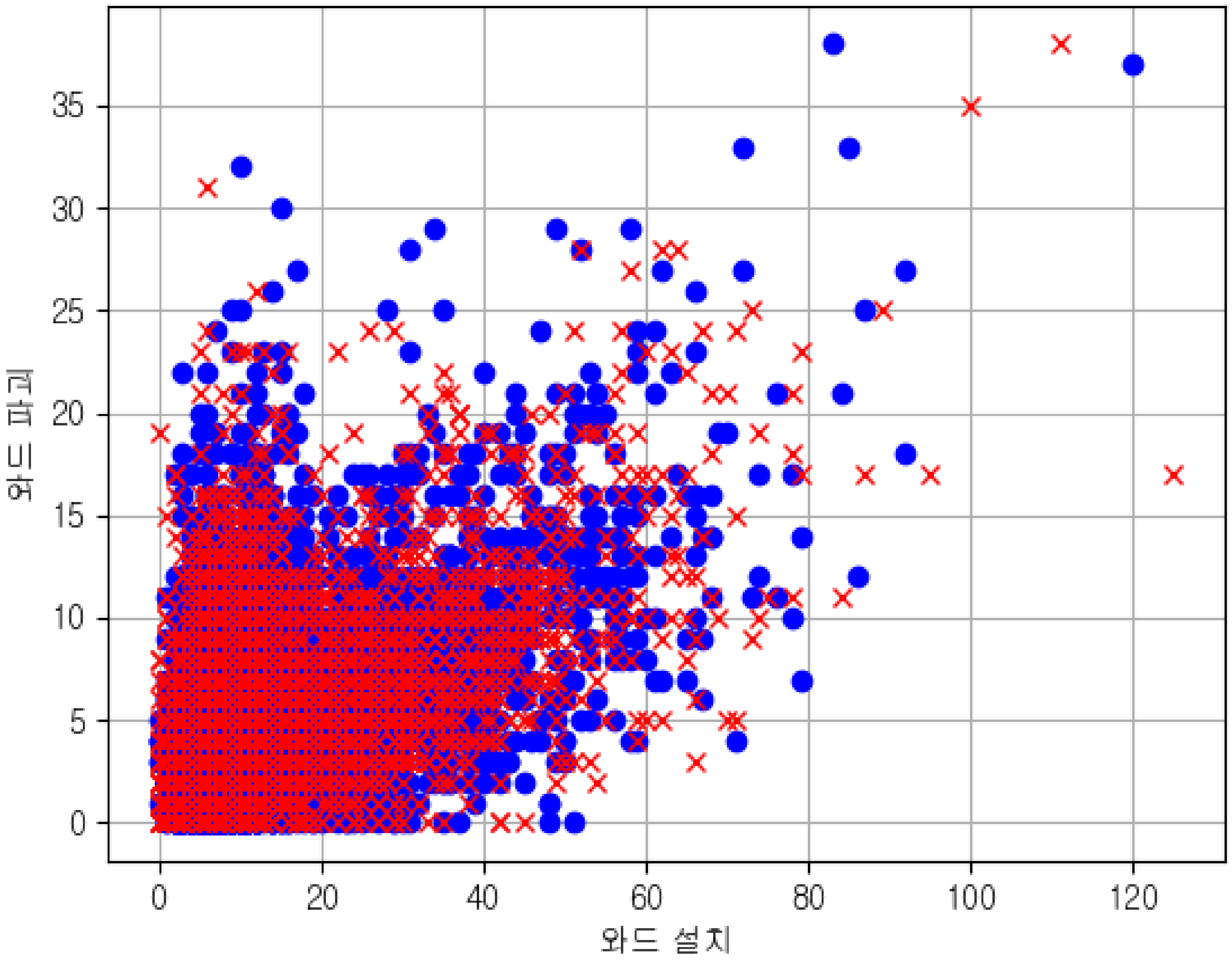
CHAPTER 03

와드 설치, 파괴 횟수와 승패의 관계



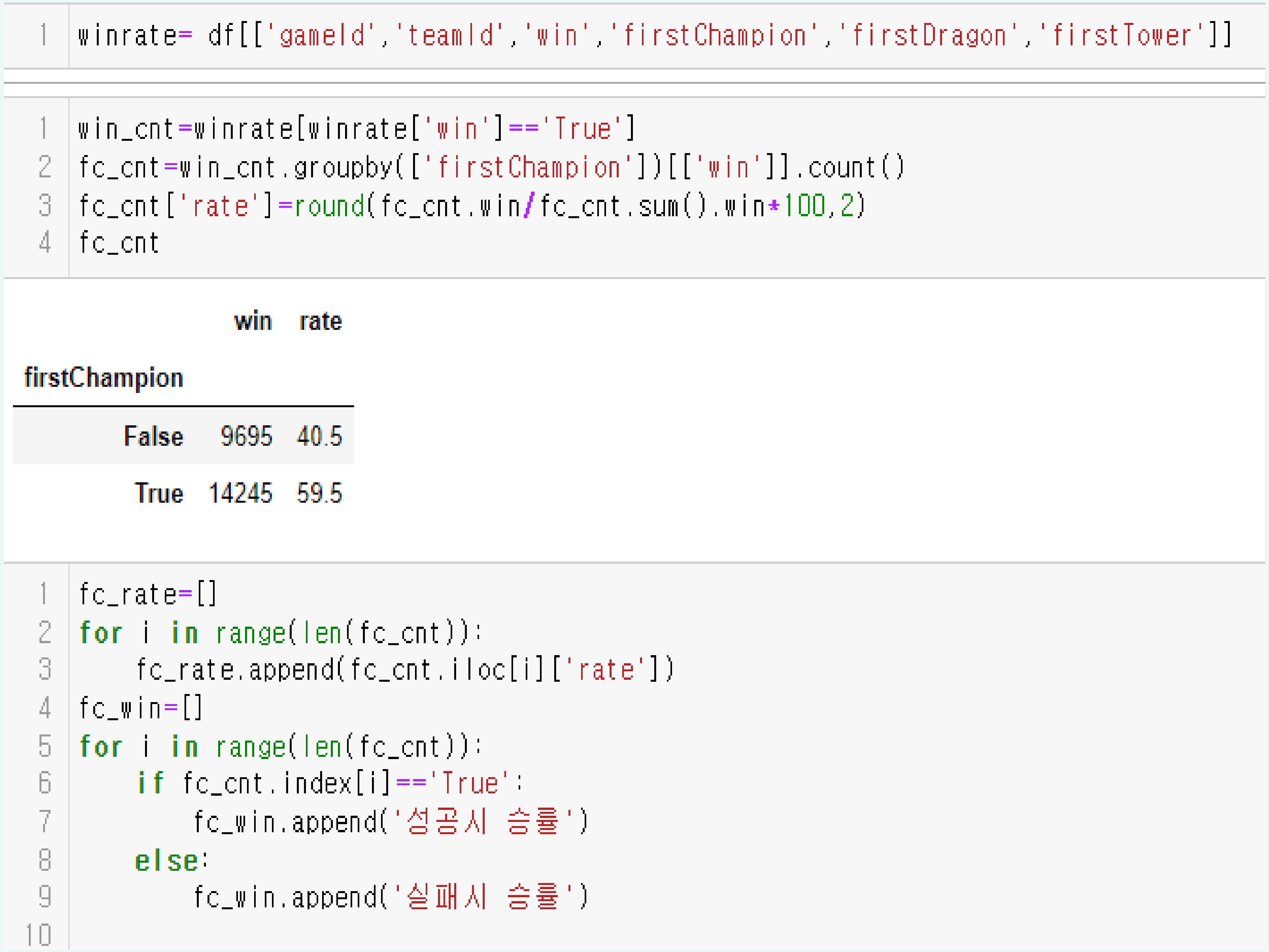
```
sql_conn = mu.connect_mysql('icia')
df = mu.mysql_execute_dict('select * from lol_mini', sql_conn)
sql_conn.close()
df = pd.DataFrame(df)

# 와드 설치, 파괴 갯수와 승률 상관관계
tmp_df = df[['wardsPlaced', 'wardsKilled', 'win']]
sort_win_df = tmp_df[tmp_df['win'] == 'True']
sort_lose_df = tmp_df[tmp_df['win'] == 'False']
sort_lose_df['win'].count()

plt.figure()
xdata = sort_win_df['wardsPlaced']
ydata = sort_win_df['wardsKilled']
plt.plot(xdata, ydata, color='b', marker='o', linestyle='None')

xdata2 = sort_lose_df['wardsPlaced']
ydata2 = sort_lose_df['wardsKilled']
plt.plot(xdata2, ydata2, color='r', marker='x', linestyle='None')
plt.xlabel('와드 설치')
plt.ylabel('와드 파괴')
plt.grid(True)
```

와드 설치, 파괴 횟수와 승패의 관계



CHAPTER 03

퍼블, 포블, 첫용과 승률의 관계



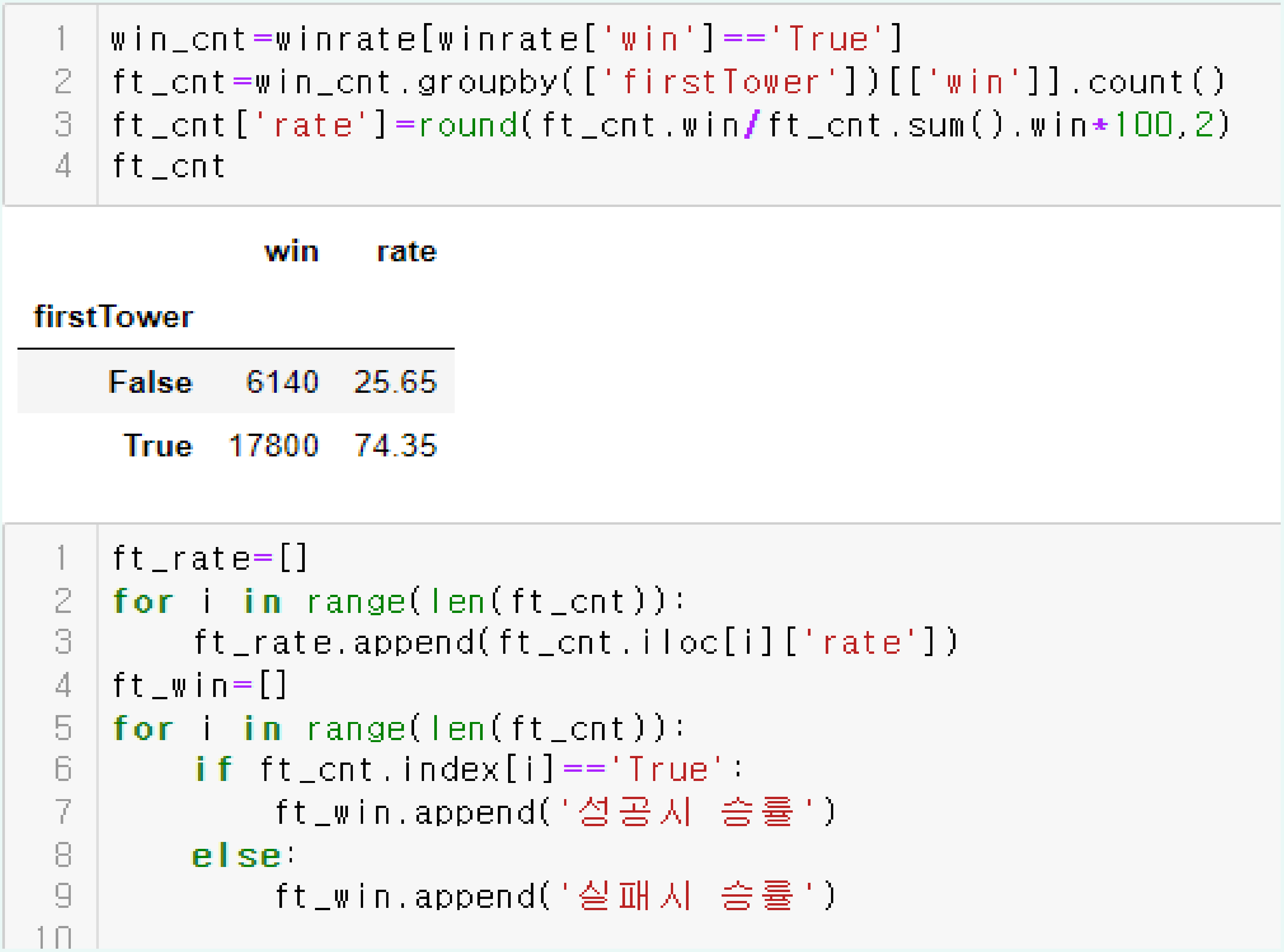
```python
winrate= df[['gameId','teamId','win','firstChampion','firstDragon','firstTower']]
```

```python
win_cnt=winrate[winrate['win']=='True']
fc_cnt=win_cnt.groupby(['firstChampion'])[['win']].count()
fc_cnt['rate']=round(fc_cnt.win/fc_cnt.sum().win*100,2)
fc_cnt
```

```
                win   rate
firstChampion
        False  9695   40.5
         True 14245   59.5
```

```python
fc_rate=[]
for i in range(len(fc_cnt)):
    fc_rate.append(fc_cnt.iloc[i]['rate'])
fc_win=[]
for i in range(len(fc_cnt)):
    if fc_cnt.index[i]=='True':
        fc_win.append('성공시 승률')
    else:
        fc_win.append('실패시 승률')

```

퍼블, 포블, 첫용과 승률의 관계

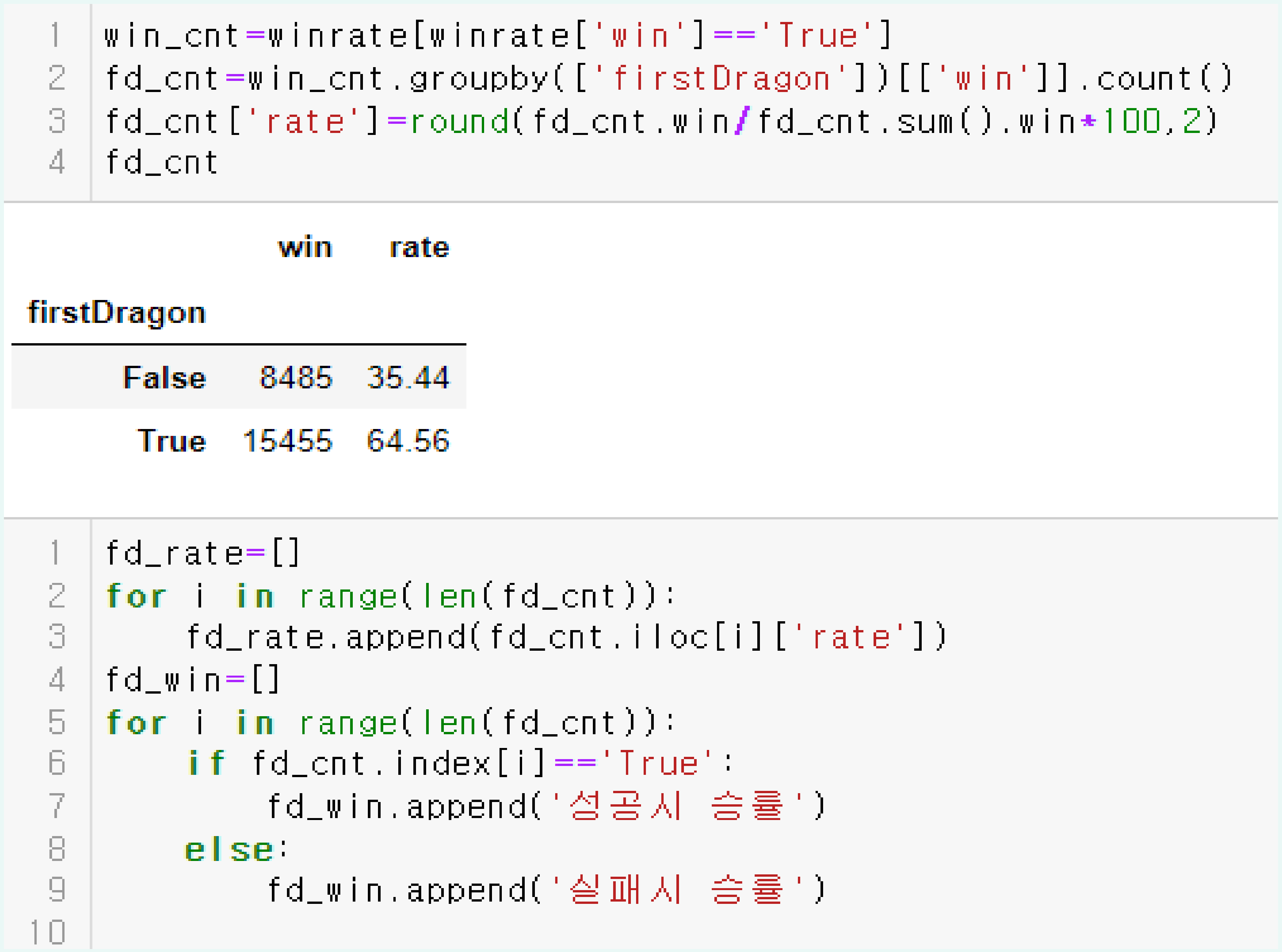


```
win_cnt=winrate[winrate['win']=='True']
ft_cnt=win_cnt.groupby(['firstTower'])[['win']].count()
ft_cnt['rate']=round(ft_cnt.win/ft_cnt.sum().win*100,2)
ft_cnt
```

```
             win   rate
firstTower
     False  6140  25.65
      True 17800  74.35
```

```
ft_rate=[]
for i in range(len(ft_cnt)):
    ft_rate.append(ft_cnt.iloc[i]['rate'])
ft_win=[]
for i in range(len(ft_cnt)):
    if ft_cnt.index[i]=='True':
        ft_win.append('성공시 승률')
    else:
        ft_win.append('실패시 승률')
```

퍼블, 포블, 첫용과 승률의 관계



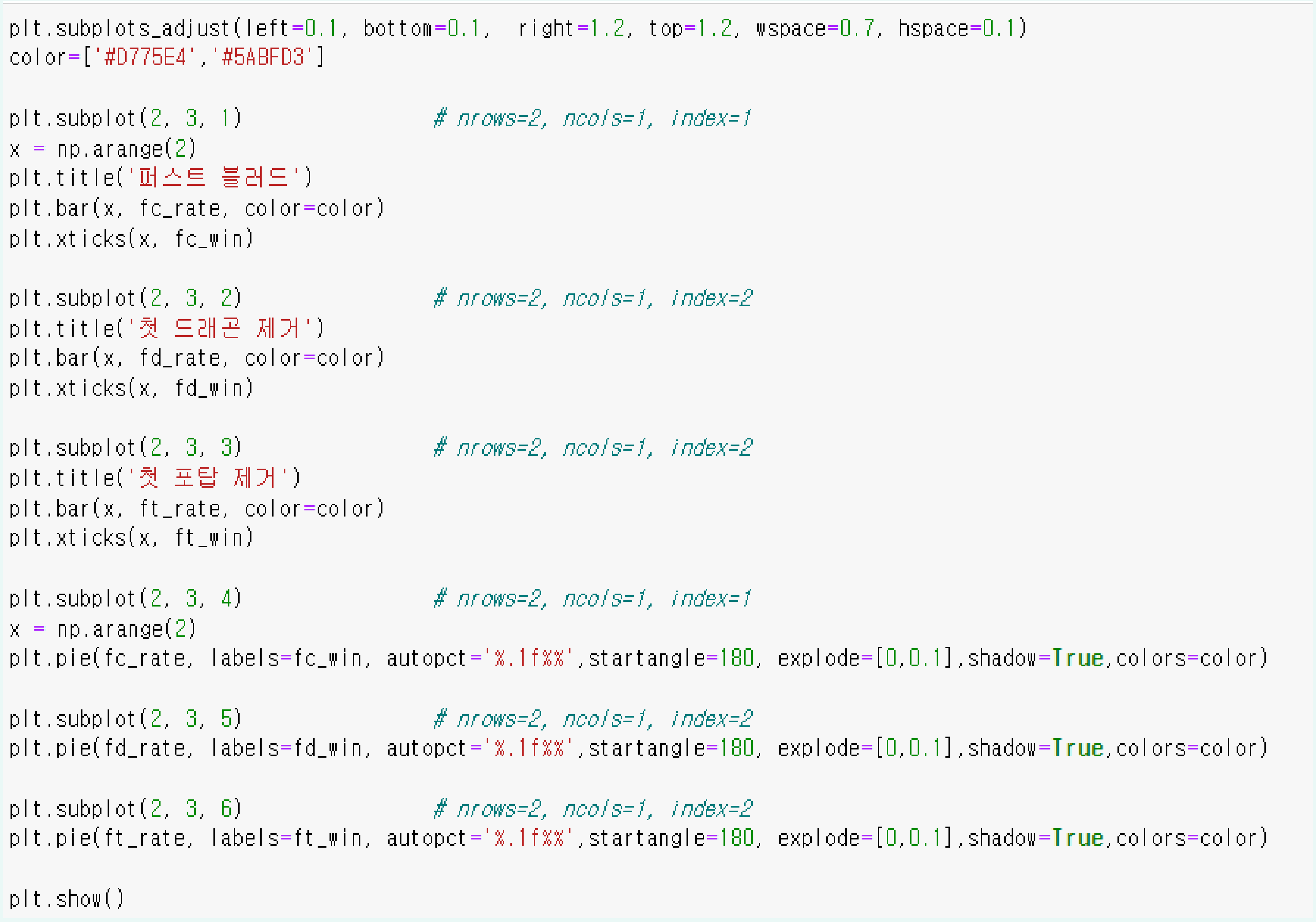
```python
win_cnt=winrate[winrate['win']=='True']
fd_cnt=win_cnt.groupby(['firstDragon'])[['win']].count()
fd_cnt['rate']=round(fd_cnt.win/fd_cnt.sum().win*100,2)
fd_cnt
```

```
               win    rate
firstDragon
      False   8485   35.44
       True  15455   64.56
```

```python
fd_rate=[]
for i in range(len(fd_cnt)):
    fd_rate.append(fd_cnt.iloc[i]['rate'])
fd_win=[]
for i in range(len(fd_cnt)):
    if fd_cnt.index[i]=='True':
        fd_win.append('성공시 승률')
    else:
        fd_win.append('실패시 승률')
```

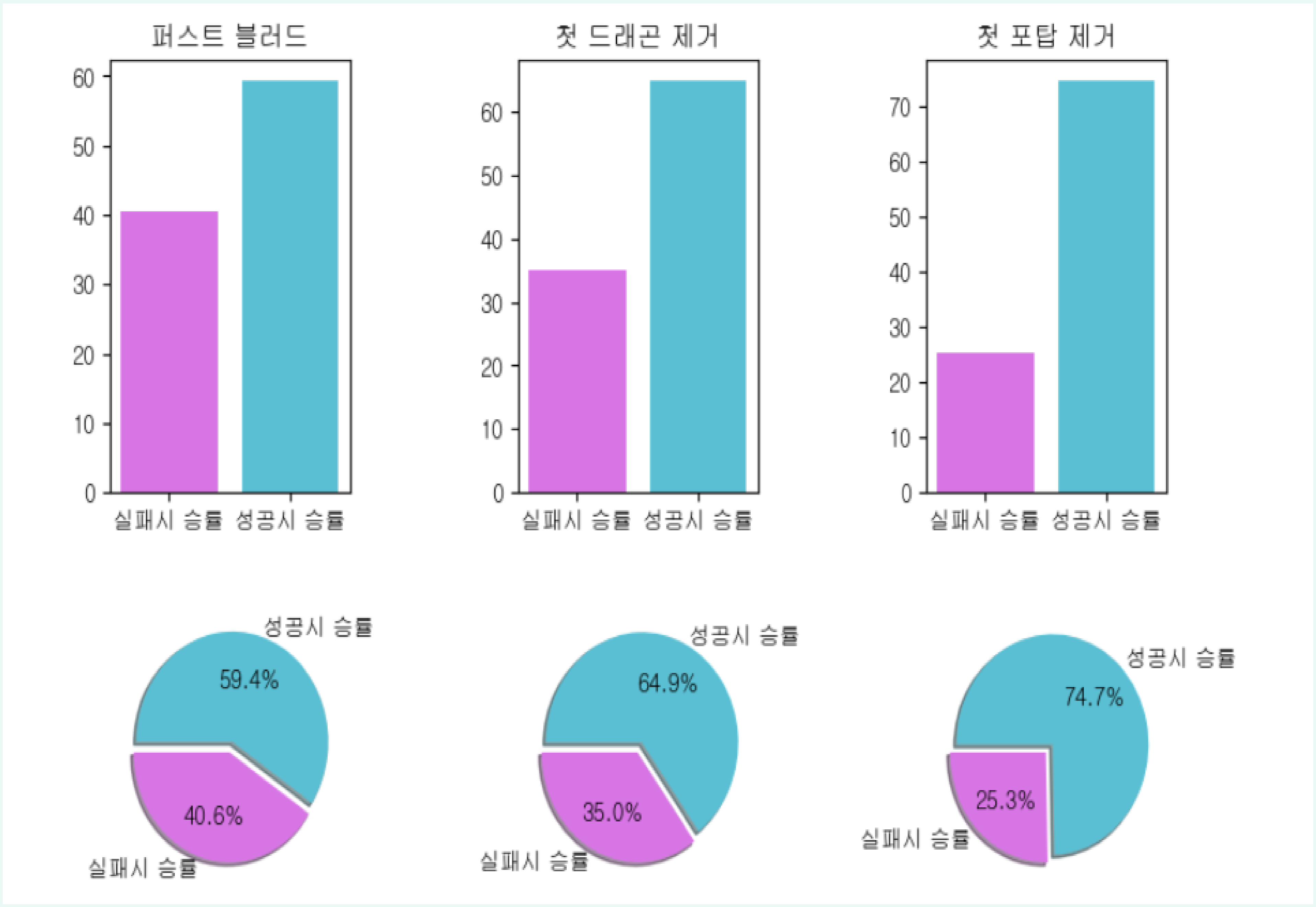
CHAPTER 03

퍼블, 포블, 첫용과 승률의 관계



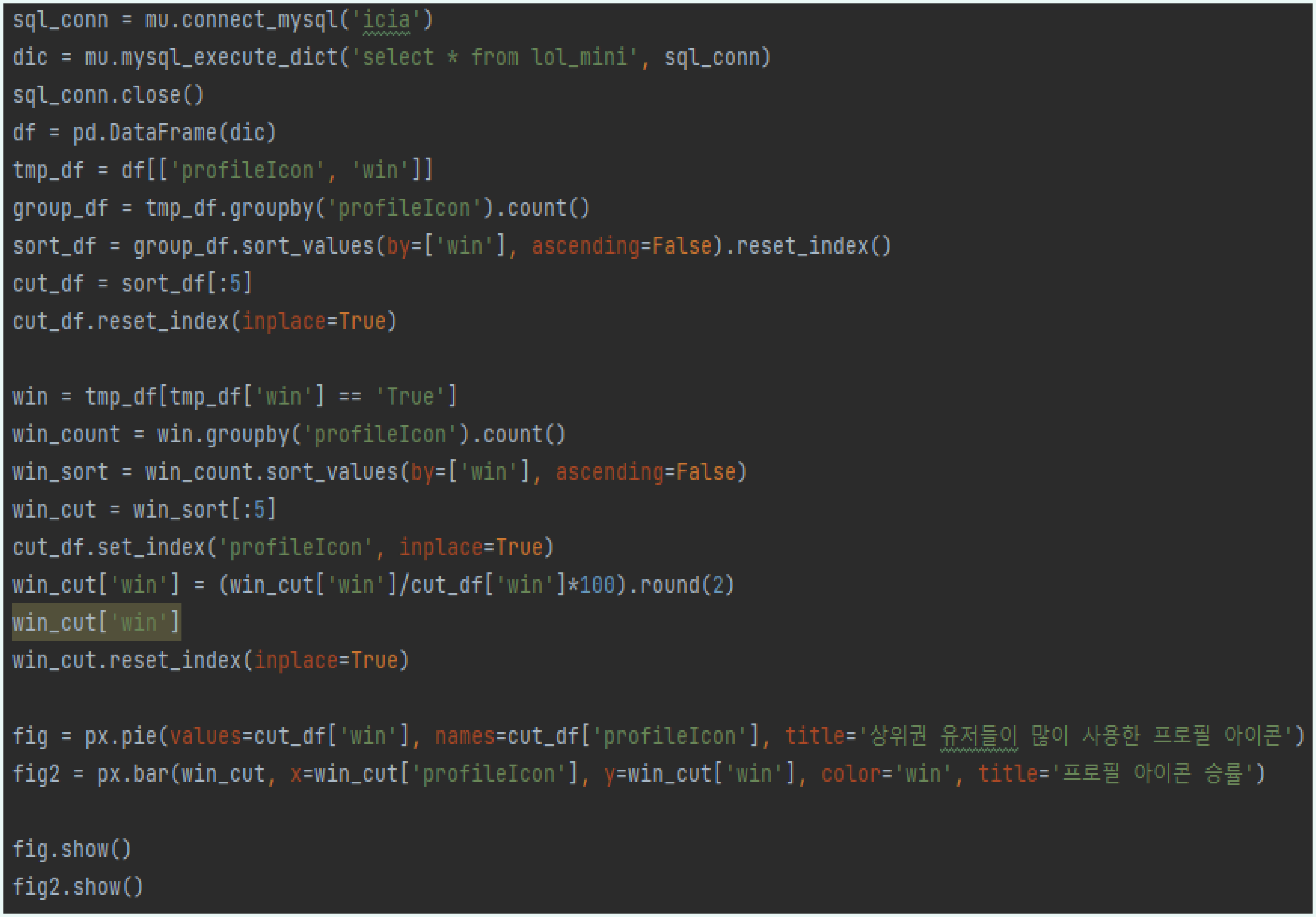
```
plt.subplots_adjust(left=0.1, bottom=0.1,  right=1.2, top=1.2, wspace=0.7, hspace=0.1)
color=['#D775E4','#5ABFD3']

plt.subplot(2, 3, 1)              # nrows=2, ncols=1, index=1
x = np.arange(2)
plt.title('퍼스트 블러드')
plt.bar(x, fc_rate, color=color)
plt.xticks(x, fc_win)

plt.subplot(2, 3, 2)              # nrows=2, ncols=1, index=2
plt.title('첫 드래곤 제거')
plt.bar(x, fd_rate, color=color)
plt.xticks(x, fd_win)

plt.subplot(2, 3, 3)              # nrows=2, ncols=1, index=2
plt.title('첫 포탑 제거')
plt.bar(x, ft_rate, color=color)
plt.xticks(x, ft_win)

plt.subplot(2, 3, 4)              # nrows=2, ncols=1, index=1
x = np.arange(2)
plt.pie(fc_rate, labels=fc_win, autopct='%.1f%%',startangle=180, explode=[0,0.1],shadow=True,colors=color)

plt.subplot(2, 3, 5)              # nrows=2, ncols=1, index=2
plt.pie(fd_rate, labels=fd_win, autopct='%.1f%%',startangle=180, explode=[0,0.1],shadow=True,colors=color)

plt.subplot(2, 3, 6)              # nrows=2, ncols=1, index=2
plt.pie(ft_rate, labels=ft_win, autopct='%.1f%%',startangle=180, explode=[0,0.1],shadow=True,colors=color)

plt.show()
```

퍼블, 포블, 첫용과 승률의 관계



CHAPTER 03

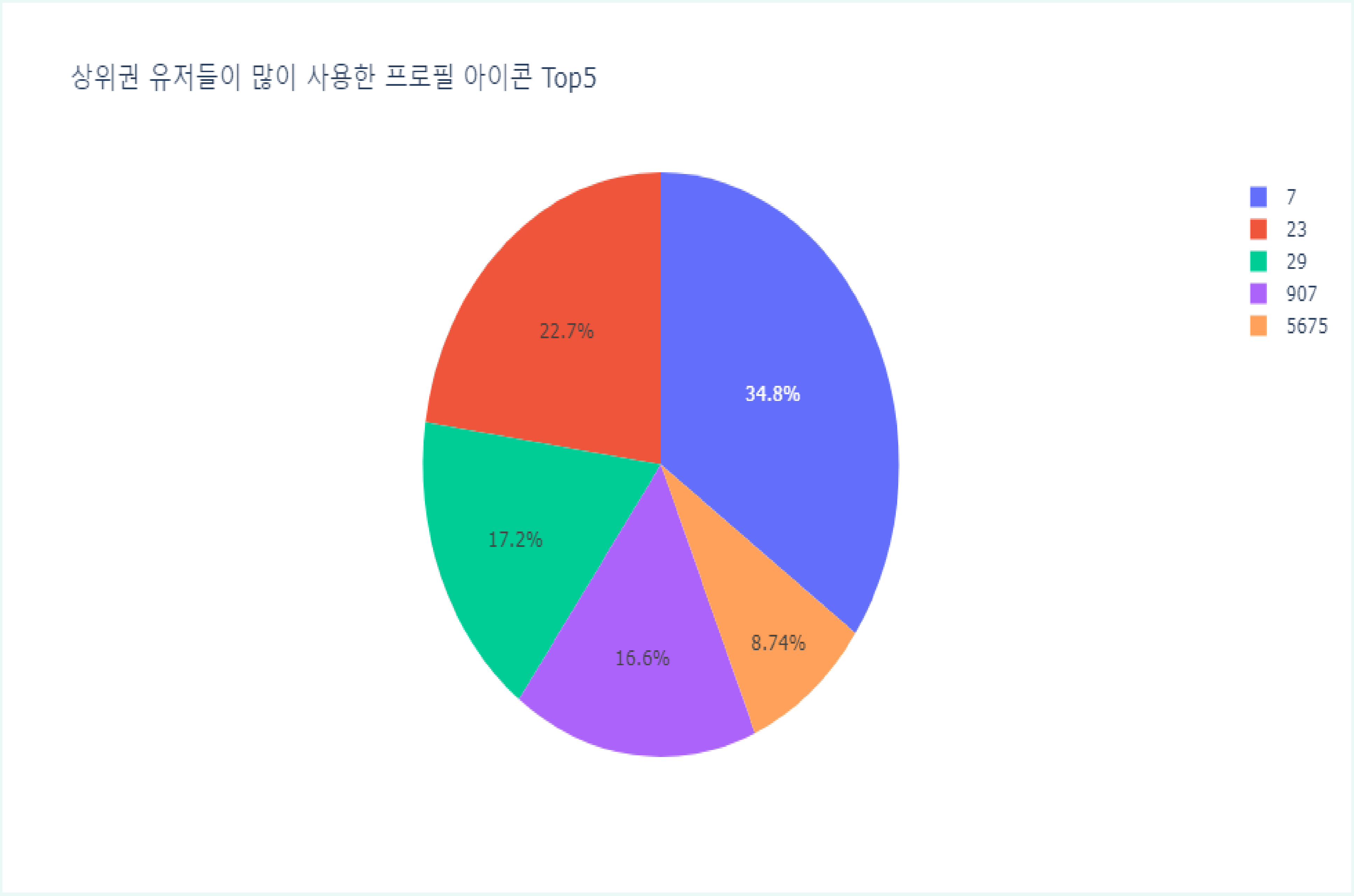
많이 사용한 프로필 아이콘과 승률



```
sql_conn = mu.connect_mysql('icia')
dic = mu.mysql_execute_dict('select * from lol_mini', sql_conn)
sql_conn.close()
df = pd.DataFrame(dic)
tmp_df = df[['profileIcon', 'win']]
group_df = tmp_df.groupby('profileIcon').count()
sort_df = group_df.sort_values(by=['win'], ascending=False).reset_index()
cut_df = sort_df[:5]
cut_df.reset_index(inplace=True)

win = tmp_df[tmp_df['win'] == 'True']
win_count = win.groupby('profileIcon').count()
win_sort = win_count.sort_values(by=['win'], ascending=False)
win_cut = win_sort[:5]
cut_df.set_index('profileIcon', inplace=True)
win_cut['win'] = (win_cut['win']/cut_df['win']*100).round(2)
win_cut['win']
win_cut.reset_index(inplace=True)

fig = px.pie(values=cut_df['win'], names=cut_df['profileIcon'], title='상위권 유저들이 많이 사용한 프로필 아이콘')
fig2 = px.bar(win_cut, x=win_cut['profileIcon'], y=win_cut['win'], color='win', title='프로필 아이콘 승률')

fig.show()
fig2.show()
```

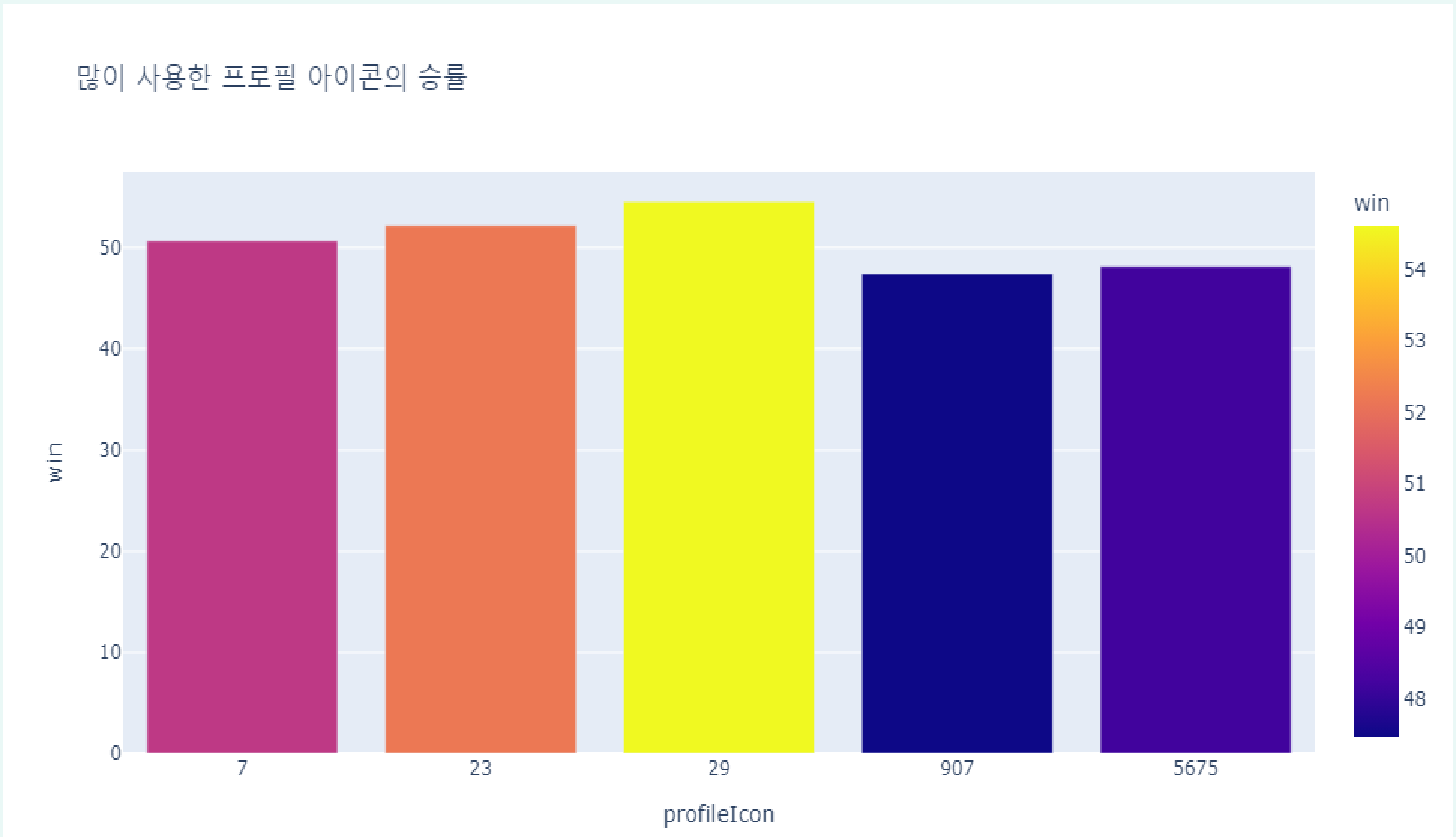
CHAPTER 03

많이 사용한 프로필 아이콘과 승률



CHAPTER 03

많이 사용한 프로필 아이콘과 승률


많이 사용한 프로필 아이콘의 승률
win
profileIcon
win