Code used to generate data for England collapse post, published 20th February 2022

1. Import Selenium, openpyxl, and pandas, set up url and driver, and open chrome. Then create a loop for finding information on each test match in a given year and append this data to a pandas dataframe. Export dataframe as csv and repeat for each year since 2012. Import new dataset that has every test match played since 2012 after appending each year together in to one csv file (done in excel)

```
[ ]: #Import Selenium, set up url and driver, and open Chrome to the url
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
import openpyxl
import pandas as pd
url = 'https://stats.espncricinfo.com/ci/engine/records/team/match_results.html?class=1;id=2021;type=year'
s = Service('/Users/jacktait/Downloads//chromedriver')
driver = webdriver.Chrome(service=s)
driver.get(url)
```

```
[ ]: #Find elements matching class name 'data1', the class name for the main table of matches
matches = driver.find_elements(By.CLASS_NAME, 'data1')
```

```
[ ]: #Create a loop for finding the match information of all test matches in the list, append to a blank dataframe
dataframe = []

for match in matches:
    Test_Results = {
    'home_team' : match.find_elements(By.TAG_NAME, 'td')[0].text,
    'away_team' : match.find_elements(By.TAG_NAME, 'td')[1].text,
    'outcome' : match.find_elements(By.TAG_NAME, 'td')[2].text,
    'date' : match.find_elements(By.TAG_NAME, 'td')[5].text,
    'match_id' : match.find_elements(By.TAG_NAME, 'td')[6].text,
    'scorecard_link' : match.find_element(By.PARTIAL_LINK_TEXT, 'Test').get_attribute('href')}

    dataframe.append(Test_Results)

print(dataframe)
```

```
: #Turn dataframe into a Pandas dataframe, then export as csv
Test_Results_2021 = pd.DataFrame(dataframe)
Test_Results_2021.to_csv('/Users/jacktait/Google Drive/Sports Analytics/Test_Results_2021')
```

```
: #Open new chrome window using web driver
s = Service('/Users/jacktait/Downloads//chromedriver')
driver = webdriver.Chrome(service=s)
```

```
: #Import the csv file with all test matches played since 2012
df = pd.read_csv('/Users/jacktait/Google Drive/Sports Analytics/Test_Matches_2012-22.csv')
```

2. Define a function that runs a loop through every link in the scorecard_link column of the csv, finds the fall of wicket data in each scorecard, and returns these variables in a pandas dataframe. Each wicket (1-10) features a double try-except pattern to account for innings where not every player batted, in which case another table is generated in the scorecard which changes the x-path of the needed element. Repeat this for each innings 1-4 and append into a single csv file with one row for every innings of every test match

```
#Define a function that runs a loop through every link in the scorecard_link column of the csv, finds the fall of wicket data in each scorecard, and returns those variables in a pandas dataframe
#Double except loop in each variable is to account for innings where not every player batted. In that case, another table is generated in the scorecard which changes the X-path of the needed element
def get_FoW_info(row):
    global bat_team
    global one
    global two
    global three
    global four
    global five
    global six
    global seven
    global eight
    global nine
    global ten
    driver.get(row['scorecard_link'])
    try:
        bat_team = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/span/div/div/div[2]/h5').text.split("2")[0]
        print(bat_team)
    except NoSuchElementException:
        bat_team = "N/A"
        print ("N/A")
    try:
        one = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[1]').text.split(" ")[0]
        print(one)
    except NoSuchElementException:
        try:
            one = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[1]').text.split(" ")[0]
            print(one)
        except NoSuchElementException:
            one = "N/A"
            print ("N/A")
    try:
        two = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[2]').text.split(" ")[1]
        print(two)
    except NoSuchElementException:
        try:
            two = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[2]').text.split(" ")[1]
            print(two)
        except NoSuchElementException:
            two = "N/A"
            print ("N/A")
```

```
    try:
        three = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[3]').text.split(" ")[1]
        print(three)
    except NoSuchElementException:
        try:
            three = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[3]').text.split(" ")[1]
            print(three)
        except NoSuchElementException:
            three = "N/A"
            print ("N/A")
    try:
        four = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[4]').text.split(" ")[1]
        print(four)
    except NoSuchElementException:
        try:
            four = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[4]').text.split(" ")[1]
            print(four)
        except NoSuchElementException:
            four = "N/A"
            print ("N/A")
    try:
        five = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[5]').text.split(" ")[1]
        print(five)
    except NoSuchElementException:
        try:
            five = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[5]').text.split(" ")[1]
            print(five)
        except NoSuchElementException:
            five = "N/A"
            print ("N/A")
    try:
        six = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[6]').text.split(" ")[1]
        print(six)
    except NoSuchElementException:
        try:
            six = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[6]').text.split(" ")[1]
            print(six)
        except NoSuchElementException:
            six = "N/A"
            print ("N/A")
    try:
        seven = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[7]').text.split(" ")[1]
        print(seven)
```

(Loop continues on next page)

```
    try:
        seven = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[7]').text.split(" ")[1]
        print(seven)
    except NoSuchElementException:
        try:
            seven = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[7]').text.split(" ")[1]
            print(seven)
        except NoSuchElementException:
            seven = "N/A"
            print ("N/A")
    try:
        eight = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[8]').text.split(" ")[1]
        print(eight)
    except NoSuchElementException:
        try:
            eight = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[8]').text.split(" ")[1]
            print(eight)
        except NoSuchElementException:
            eight = "N/A"
            print ("N/A")
    try:
        nine = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[9]').text.split(" ")[1]
        print(nine)
    except NoSuchElementException:
        try:
            nine = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[9]').text.split(" ")[1]
            print(nine)
        except NoSuchElementException:
            nine = "N/A"
            print ("N/A")
    try:
        ten = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[10]').text.split(" ")[1]
        print(ten)
    except NoSuchElementException:
        try:
            ten = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[10]').text.split(" ")[1]
            print(ten)
        except NoSuchElementException:
            ten = "N/A"
            print ("N/A")

            print ("N/A")
    try:
        ten = driver.find_element(By.XPATH,'//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[2]/td/div/span[10]').text.split(" ")[1]
        print(ten)
    except NoSuchElementException:
        try:
            ten = driver.find_element(By.XPATH, '//*[@id="main-container"]/div[1]/div/div[2]/div[2]/div[1]/div[2]/div[2]/div[4]/div[1]/div/div/div/table[1]/tfoot/tr[3]/td/div/span[10]').text.split(" ")[1]
            print(ten)
        except NoSuchElementException:
            ten = "N/A"
            print ("N/A")


    return pd.Series({
    'bat_team' : bat_team,
    'one' : one,
    'two' : two,
    'three' : three,
    'four' : four,
    'five' : five,
    'six' : six,
    'seven' : seven,
    'eight' : eight,
    'nine' : nine,
    'ten' : ten
    })
```

```
#Run the function and append the pandas dataframe with the new variables to the existing csv file
new_df = df.apply(get_FoW_info, axis=1).join(df)
```

```
#Save the new dataframe as a csv
new_df.to_csv('/Users/jacktait/Google Drive/Sports Analytics/4th_Innings_2012-22.csv')
```

3. Create variables for number of runs lost for the loss of previous three wickets, then define a collapse variable that is equal to 1 if any of the created variables are equal to or less than 30. Do the same again for the number of runs lost for the loss of previous four wickets, then define a collapse4 variable that is equal to 1 if any of the created variables are equal to or less than 40. As explained in the full article, I used collapse4 variable for all subsequent analysis. Create new dataframes for each innings 1-4

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

```
#Import csv with every innings from every test match since 2012 with fall of wicket data
test_inns = pd.read_csv('/Users/jacktait/Google Drive/Sports Analytics/Test_Match_Innings_2012-22_no_NA.csv')
```

```
#Create variables for whether a collapse occured.
#First, calculate how many runs scored between the previous three wickets falling:
test_inns['3-1'] = test_inns['three']-test_inns['one']
test_inns['4-2'] = test_inns['four']-test_inns['two']
test_inns['5-3'] = test_inns['five']-test_inns['three']
test_inns['6-4'] = test_inns['six']-test_inns['four']
test_inns['7-5'] = test_inns['seven']-test_inns['five']

#Then create a dummy 'collapse' variable that's equal to 1 if three wickets fell for less than 30 runs (up to the 7th wicket)
test_inns['collapse'] = np.where(test_inns['3-1'] <= 30, 1, np.where(test_inns['4-2'] <= 30, 1, np.where(test_inns['5-3'] <= 30, 1, np.where(test_inns['6-4'] <= 30, 1, np.where(test_inns['7-5'] <=30, 1, 0))

#Repeat the above for the number of runs scored between the previous four wickets falling
test_inns['4-1'] = test_inns['four']-test_inns['one']
test_inns['5-2'] = test_inns['five']-test_inns['two']
test_inns['6-3'] = test_inns['six']-test_inns['three']
test_inns['7-4'] = test_inns['seven']-test_inns['four']

#Create another dummy 'collapse4' that's equal to 1 if four wickets fell for less than 40 runs (up to the 7th wicket)
test_inns['collapse4'] = np.where(test_inns['4-1'] <= 40, 1, np.where(test_inns['5-2'] <= 40, 1, np.where(test_inns['6-3'] <= 40, 1, np.where(test_inns['7-4'] <= 40, 1, 0))))

#Change match_id to a variable that is just a number rather than "Test_number", e.g. change "Test #1000" to "1000"
new = test_inns["match_id"].str.split("#", n = 1, expand = True)
test_inns['match_number'] = new[1]
test_inns.drop(columns='match_id', inplace=True)
```

```
#New dataframes for each innings
first_innings = test_inns.iloc[0:424, :,]
second_innings = test_inns.iloc[424:848, :,]
third_innings = test_inns.iloc[849:1272, :,]
fourth_innings = test_inns.iloc[1273:, :,]
```

4. Create dataframe for each country's innings, then calculate collapse percentage and big collapse percentage (big collapse percentage uses collapse4 variable). Create horizontal bar chart based on this data. Create new dataframe of matches played since February 2017 then group by batting team.

```
#Show number of total collapses per team since 2012
bat_country_all_innings = test_inns_two.groupby(['bat_team'])['collapse', 'collapse4'].sum().reset_index()
#Add a column for the number of innings per team
bat_country_all_innings['innings_count'] = [0, 12, 187, 106, 244, 177, 6, 152, 148, 154, 177, 157, 58]
#Calculate percentage of innings with a collapse and big collapse
#Big collapse uses the higher bar (collapse4 variable) and collapse uses regular collapse variable
bat_country_all_innings['collapse_pct'] = (bat_country_all_innings['collapse']/bat_country_all_innings['innings_count'])*100
bat_country_all_innings['big_collapse_pct'] = (bat_country_all_innings['collapse4']/bat_country_all_innings['innings_count'])*100
```

```
#Create a new dataframe sorted by big collapse percentage, and drop rows for Zimbabwe, Ireland, and Afghanistan
all_innings_sorted_by_big_collapse = bat_country_all_innings.sort_values('big_collapse_pct', ascending = False)
all_innings_sorted_by_big_collapse = all_innings_sorted_by_big_collapse.drop([0, 5, 11])
```

```
#Create horizontal bar graph showing big collapse percentage by team
#Set columns to graph, labels for axes, title, and colours for each team
fig, ax = plt.subplots(figsize=(13, 6))
bars = ax.barh(all_innings_sorted_by_big_collapse['bat_team'], all_innings_sorted_by_big_collapse['big_collapse_pct'], color = ['maroon', 'darkgreen', 'red', 'c
plt.xlabel('Percentage of Innings with a Collapse')
plt.ylabel('Country')
plt.title('Percentage of Test Match Innings with a Batting Collapse Among Top Nine Nations, 2012-22')
plt.show()
```

```
#Create new dataframe with test matches since 2017
test_innings_since_2017 = test_inns[test_inns['match_number'] > 2249]

#New dataframe with collapses per team since 2017
bat_team_2017 = test_innings_since_2017.groupby(['bat_team'])['collapse', 'collapse4',].sum().reset_index()

#Repeat steps to calculate collapse percentage , drop Afghanistan, Ireland, and Afghanistan, and sort by collapse percentage
#Repeat the horizontal bar chart for batting collapses since 2017
```

5. Calculate collapse percentages for first innings, first innings since 2012, then repeat the steps for innings 2-4. Create new dataframes for home and away test innings, repeat calculating the collapse percentages and generating graphs for each of these dataframes.

```
#Create dataset of just first innings, then group by team to show collapses per team in the first innings
first_innings = test_inns[test_inns['Innings'] == 1]
team_first_innings = first_innings.groupby(['bat_team'])['collapse', 'collapse4'].sum().reset_index()
#Repeat sorting by big collapse percentage and creating the graph
```

```
#Create dataframe with first innings since 2017, then aggregate by team
first_innings_since_2017 = first_innings[first_innings['match_number'] > 2249]
team_first_innings_since_2017 = first_innings_since_2017.groupby(['bat_team'])['collapse', 'collapse4'].sum().reset_index()
#Repeat sorting by big collapse percentage and creating the graph for this data
```

```
#Repeat the steps for first innings and first innings since 2017 for 2nd, 3rd, and 4th innings of the match.
#Subsetting the whole dataframe by innings (and by year for post-2017), then aggregating by team for totals
#Create collapse percentage variables, sort by collapse percentage, and generate the graph of the data
#All code for these steps are the same as the first innings code
```

```
#Subset the dataframe for innings where the home team is batting
home_test_innings = test_innings[test_innings['home_team'] == test_innings['bat_team']]
team_home_innings = home_test_innings.groupby(['bat_team'])['collapse', 'collapse4'].sum().reset_index()
#Add the number of innings for each team
team_home_innings['innings_count'] = [8, 105, 56, 132, 77, 2, 82, 30, 83, 84, 77, 32]
#Repeat above steps for creating collapse percentage variables, sort by collapse percentage and create graph
```

```
#Subset the dataframe for innings where the away team is batting
away_test_innings = test_innings[test_innings['away_team'] == test_innings['bat_team']]
team_away_innings = away_test_innings.groupby(['bat_team'])['collapse', 'collapse4'].sum().reset_index()
#Add the number of innings for each team
team_away_innings['innings_count'] = [4, 82, 50, 112, 100, 4, 70, 118, 71, 93, 80, 26]
#Repeat above steps for creating collapse percentage variables, sort by collapse percentage and create graph
```

6. Create dataframe for all innings played in England, group by batting team and repeat code to calculate collapse percentage and generate graph. Do the same for batting innings grouped by location, rather than batting team

```
#Subset the dataframe for innings that are played in England (where England is the home team, but any team is batting)
innings_in_england = test_innings[test_innings['home_team'] == 'England']
team_innings_in_england = innings_in_england.groupby(['bat_team'])['collapse', 'collapse4'].sum().reset_index()
#Drop Ireland and "N/A" from the data (N/A row is for innings that did not even start, e.g. 4th innings not needed)
team_innings_in_england = team_innings_in_england.drop([0, 4])
```

```
#Create new dataframe based on the location of the innings (grouping by home team rather than batting team)
innings_by_location = test_innings.groupby(['home_team'])['collapse', 'collapse4'].sum().reset_index()
#Drop Afghanistan, Ireland, and Zimbabwe rows
innings_by_location = innings_by_location.drop([0, 5, 11])
#Repeat above steps for creating collapse percentage and sorting by collapse percentage
#Create graph of collapse percentage based on host nation using same base code as above
```

7. Create dataframe consisting of innings played in England by all teams *except* England and sum the values. Append this new row for all other teams to a dataframe consisting of all teams separately. Then create new dataframe with just two rows, one for England and one for all other teams combined. Calculate collapse percentage and generate graph. Create new dataframe grouped by innings to find collapse rates for each innings.

```
#This section of code is very messy, so bare with me
#Create a new dataframe from the innings in England dataframe that contains summed data for all teams except England
all_other_teams = team_innings_in_england.iloc[2:8].sum()
#The initial iloc call did not include Australia's numbers, so added those in manually (adding 20, 13, and 29)
all_other_teams['collapse'] = all_other_teams['collapse'] + 20
all_other_teams['collapse4'] = all_other_teams['collapse4']+13
all_other_teams['innings_count'] = all_other_teams['innings_count']+29
#Add the data for all other teams to the existing dataframe as a new row
team_innings_in_england = team_innings_in_england.append(all_other_teams.transpose(), ignore_index=True)
#This row was initially called "IndiaNew ZealandPakistan..." rather than all other teams, so needed to rename
#Make bat_team the index rather than a column to make renaming the new row easier
team_innings_in_england.set_index('bat_team', inplace=True)
#Rename the index to rename the final row as All Other Teams
team_innings_in_england.index = ['Australia', 'England', 'India', 'New Zealand', 'Pakistan', 'South Africa', 'Sri Lanka', 'West Indies', 'All Other Teams']
#Create new dataframe containing just the rows for England and All Other Teams when batting in England
#Reset the index to return bat_team to a column
england_and_others_in_england = team_innings_in_england.loc[['England', 'All Other Teams']]
england_and_others_in_england.reset_index(inplace=True)
#Calculate collapse percentage
england_and_others_in_england['collapse_pct'] = (england_and_others_in_england['collapse']/england_and_others_in_england['innings_count'])*100
england_and_others_in_england['big_collapse_pct'] = (england_and_others_in_england['collapse4']/england_and_others_in_england['innings_count'])*100
#Create a graph of these two rows using the same code as previous graphs
```

```
#Aggregate the original dataframe by innings 1-4
collapse_by_innings = test_innings.groupby(['Innings'])['collapse', 'collapse4'].sum().reset_index()
#Add innings count for each innings and calculate collapse percentages
collapse_by_innings['innings_count'] = [424, 422, 415, 317]
collapse_by_innings['collapse_pct'] = (collapse_by_innings['collapse']/collapse_by_innings['innings_count'])*100
collapse_by_innings['big_collapse_pct'] = (collapse_by_innings['collapse4']/collapse_by_innings['innings_count'])*100
#Graph this data using existing graph code
```

9. Create new dataframe for substantial innings, has every innings except those in the 4th innings that saw less than 4 wickets fall. This was done to overcome the issue of 4th innings having very low collapse rates because so many 4th innings either didn't happen or were very short due to the game ending. Then repeat calculating collapse percentages and generating graphs. Finally, create new dataframe for batting innings in lost matches, pull out just 4th innings rows, then calculate collapse percentage and generate graph.

```
#Create new dataframe for substantial innings where the fourth innings saw at least four wickets fall
substantial_test_innings = test_innings[~test_innings['four'].isnull()]
#Aggregate substantial innings by team
substantial_innings_by_innings = substantial_test_innings.groupby(['Innings'])['collapse', 'collapse4'].sum().reset_index()
#Repeat code for calculating collapse percentage and graph from the substantial innings data
```

```
#Create new dataframe for innings that were played in matches that the batting team lost
innings_in_losses = test_innings[test_innings['outcome'] == test_innings['bowl_team']]
#Create dataframe for just 4th innings batting performance in lost test matches, then group by batting team
fourth_innings_in_losses = innings_in_losses[innings_in_losses['Innings'] == 4]
team_fourth_innings_in_losses = fourth_innings_in_losses.groupby(['bat_team'])['collapse4'].sum().reset_index()
#Add number of innings for each team column
team_fourth_innings_in_losses['innings_count'] = [59, 20, 14, 30, 13, 1, 13, 16, 18, 17, 21, 10]
#Drop innings in which no team batted, Ireland, and Zimbabwe from the dataframe
team_fourth_innings_in_losses = team_fourth_innings_in_losses.drop([0, 5, 11])
#Calculate collapse percentage per team
team_fourth_innings_in_losses['collapse_pct'] = (team_fourth_innings_in_losses['collapse4']/team_fourth_innings_in_losses['innings_count'])*100
team_fourth_innings_in_losses_sorted_by_collapse = team_fourth_innings_in_losses.sort_values('collapse_pct', ascending=False)
#Repeat code to create graph for 4th innings in losses
```