Master Thesis
The Best Position To Be In: A Study on the Impact of League Quality, Club Quality, and Playing Position on a Soccer Player's Post-Transfer Performance

Supervisor:
Thorvald Hærem

Program:
Master of Science in Leadership and Organizational Psychology

Supplemental Document:
Python Code for Data Scraping from Transfermarkt

Main Running File: Data Scraping.py

```
from bs4 import BeautifulSoup
#import urllib3.request
import requests
import pandas as pd
import re
import js2xml
import datetime
from datetime import date
from datetime import timedelta
from funcMVChart import getMVChart
from transfers import getTransferChart
from cleaningFunction import cleaningFunction
from sameClubBefore import sameClubBefore

##tell Transfermkt who I am
headers = {'User-Agent':
```

```
            'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36'}



##ask user what input page
#page = input("Enter transfer record page 1 URL, make sure ends in plus=1%2Fpage: ")
#Torwart, Abwehr, Mittelfeld, Sturm
 # !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!CHANGE HERE CHANGE HERE CHANGE HERE CHANGE HERE !!!!!!!!!!!!!!!!!!!!!!!!!! $%^&#*(*$&^%^#&*($*&#^%$&#*
page =
    'https://www.transfermarkt.com/transfers/transferrekorde/statistik?saison_id=2010&land_id=0&ausrichtung=Torwart&spielerposition_id=&alterskla
    sse=&leihe=&w_s=&plus=1'
print(page)
position = "Goalkeeper"

##ONE ABOVE REMEMBER ITS A LOOP RANGE
 # !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!CHANGE HERE CHANGE HERE CHANGE HERE CHANGE HERE !!!!!!!!!!!!!!!!!!!!!!!!!! $%^&#*(*$&^%^#&*($*&#^%$&#*
amountPages = 5
##fileName = input("Enter excel file path to save in")
##print(fileName)


playerNamesTitle = "Player Name"
formerClubsTitle = "Former Club"
cleanedFromLeagueTitle = "Old League"
newClubsTitle = "New Club"
cleanedToLeagueTitle = "New League"
beenOnBeforeListTitle = "Been on Club Before?"
numBeenOnBeforeTitle = "How many times before?"
mktValuesAtTradeTitle = "Market Values at Trade in millions of Euros"
cleanedmktValuesAtTradeTitle = "Market Values at Trade in millions of Euros clean"
transferFeesTitle = "Transfer Fee in millions of Euros"
cleanedtransferFeesTitle = "Transfer Fee in millions of Euros clean"
agesTitle = "Age at Transfer"
nationalityTitle = "Nationality (first listed)"
positionTitle = "Position"
goalkeeperTitle = "Goalkeeper"
midfielderTitle = "Midfielder"
defenderTitle = "Defender"
forwardTitle = "Forward"
footTitle = "Foot"
seasonTitle = "Season"
transferDateTitle = "Transfer Date"
daysSinceLastTransferTitle = "Days since last transfer"
t1MVsTitle = "T1 MV"
cleanedT1ListTitle = "Cleaned T1 MV"
```

```
T2DateTitle = "Date of T2"
T2T1dateGapsTitle = "Gap in days b/t T2 and T1"
t2MVsTitle = "T2 MV"
cleanedT2ListTitle = "Cleaned T2 MV"
indexOfT2PointsTitle = "Index of T2 on MV Chart"
t3MVDatesTitle = "T3 MV Date"
T3T2DateGapsTitle = "T3 Date Gap from T2"
T3T1DateGapsTitle = "T3 Date Gap from T1"
t3MVsTitle = "T3 MV"
cleanedT3ListTitle = "Cleaned T3 MV"
indexOfT3PointsTitle = "Index of T3 on MV Chart"
t4MVDatesTitle = "T4 MV Date"
t4MVsTitle = "T4 MV"
T4T1DateGapsTitle = "T4 Date Gap from T1"
T4T2DateGapsTitle = "T4 Date Gap from T2"
T4T3DateGapsTitle = "T4 Date Gap from T3"
cleanedT4ListTitle = "Cleaned T4 MV"
indexOfT4PointsTitle = "Index of T4 on MV Chart"
indexOfTransferTitle = "# on TrxChart"
numberOfTransfersBeforeTitle = "Num Transfers Before"
fullMVLinksTitle = "MV Chart Link"
majorLinksTitle = "Profile Link"
fullTransferLinksTitle = "Transfer Links"


playerNames = []
formerClubs = []
newClubs = []
mktValuesAtTrade = []
transferFees = []

cleanedClubs = []
cleanedFinancialValues = []
cleanedAges = []
cleanedFromLeague = []
cleanedToLeague = []

masterNationalityList = []
masterPositionList = []
masterFootList = []
 masterAgeList = []
 positions = []
 profileLinks = []
 fullPlayerLinks = []
 fullMVLinks = []
```

```
105.  fullTransferLinks = []
106.  majorLinks = []
107.  seasonList = []
108.
109.  allMVCharts = []
110.  allTransferCharts = []
111.  t1MVs = []
112.  t2MVs = []
113.  masterTransferDates = []
114.  daysSinceLastTransfer = []
115.  T2T1dateGaps = []
116.  T2Date = []
117.  indexOfTransfer = []
118.  numberOfTransfersBefore = []
119.  nextClosest = []
120.
121.  beenOnBeforeList = []
122.  numBeenOnBefore = []
123.
124.  indexOfT2Points = []
125.  t3MVDates = []
126.  t3MVs = []
127.  T3T2DateGaps = []
128.  T3T1DateGaps = []
129.  indexOfT3Points = []
130.
131.  t4MVDates = []
132.  t4MVs = []
133.  T4T1DateGaps = []
134.  T4T2DateGaps = []
135.  T4T3DateGaps = []
136.  indexOfT4Points = []
137.
138.  goalkeeper = []
139.  midfielder = []
140.  defender = []
141.  forward = []
142.
143.
144.
145.  for sheet in range(1,amountPages):
146.      page = page + "&page=" + str(sheet)
147.      print("Page: " + str(sheet))
148.      cleanedClubs.clear()
149.      cleanedFinancialValues.clear()
```

```
    cleanedAges.clear()

    if sheet == 4 and amountPages == 5:
        numberPlayers = 21
    else:
        numberPlayers = 25


##build beautiful soup object from input
    pageTree = requests.get(page, headers=headers)
    pageSoup = BeautifulSoup(pageTree.content, 'html.parser')


    ##------AGES  ------------------------------------------------------------------
    ##find the raw "numbers" aka ages and season text and store them
    ##extract only ages them into array
    ## ages are index 1, 5, 9 so counter at 1 then incremented by 4

    agesRaw = pageSoup.find_all("td", {"class": "zentriert"})

    for i in range(0,len(agesRaw)):
        cleanedAges.append(agesRaw[i].text)

    s = 1
    for i in range(0,numberPlayers):
        masterAgeList.append(cleanedAges[s])
        s = s + 4


    ##------GET PLAYER NAMES, FORMER CLUB, AND NEW CLUB NAME  ------------------------

    tables = pageSoup.find_all("table",{"class":"inline-table"})
##    print(tables)
##    print(len(tables))

    names = []
    images = []
    for each in tables:
        names.append(each.find("img").get('alt'))

##    print(names)
##    print(len(names))

    for j in range(0,len(names),3):
        playerNames.append(names[j])
```

```
195.
196.      for j in range(1,len(names),3):
197.          formerClubs.append(names[j])
198.
199.
200.      for j in range(2,len(names),3):
201.          newClubs.append(names[j])
202.
203.      ## ---- FIND OLD AND NEW LEAGUES ---------------
204.      rawFromLeague = []
205.      rawToLeague = []
206.      for j in range(1,len(tables),3):
207.          rawFromLeague.append(tables[j])
208.
209.      for each in rawFromLeague:
210.          rawFromLeagueImages = each.find_all("a")
211.          if(len(rawFromLeagueImages)<3):
212.              cleanedFromLeague.append("League Not Found")
213.          else:
214.              cleanedFromLeague.append(rawFromLeagueImages[2].get('title'))
215.
216.
217.      for i in range(2,len(tables),3):
218.          #print(tables[j])
219.          #print("")
220.          rawToLeague.append(tables[i])
221.
222.      for each in rawToLeague:
223.          rawToLeagueImages = each.find_all("a")
224.          if(len(rawToLeagueImages)<3):
225.              cleanedToLeague.append("League Not Found")
226.          else:
227.              cleanedToLeague.append(rawToLeagueImages[2].get('title'))
228.
229.  ##    print("")
230.  ##    print(cleanedToLeague)
231.  ##    print(len(cleanedToLeague))
232.
233.
234.      ## ----_FIND SEASON YEARS-----------------------------------------
235.      table = pageSoup.find_all("td",{"class":"zentriert"})
236.      #print(len(table))
237.
238.      for j in range(2,len(table),4):
239.          seasonList.append(table[j].text)
```

```
    #print(seasonList)
    #print(len(seasonList))


    ##--------MARKET VALUE AT TRADE AND TRANSFER FEE---------------------------------
    ##declare title, find the financial values of player, strip text, and store
    ##them in array without m or euro sign, then sort them into
    ##market value or transfer fee



    ##find all the financial values and clean them
    rawFinancialValues = pageSoup.find_all("td", {"class": "rechts"})
    for i in range(0,len(rawFinancialValues)):
        cleanedFinancialValues.append(rawFinancialValues[i].text)
        #.replace('€','').replace('m',''))

    ##Print checks for this
    ##print(cleanedFinancialValues)
    ##print(len(cleanedFinancialValues))


    ##sort the financial values based on even on
    ##into transfer or market value
    z = 1
    for each in cleanedFinancialValues:
        if z % 2 == 1:
            mktValuesAtTrade.append(each)
            z = z + 1
        else:
            transferFees.append(each)
            z = z + 1



    ##-----------GET ALL PLAYER LINKS AND STORE ------------------------------------

    profileLinksRaw = pageSoup.find_all("a",{"class": "spielprofil_tooltip"})
    #print(profileLinksRaw)
    #print(len(profileLinksRaw))
    profileLinks.clear()
    fullPlayerLinks.clear()

    for i in range(0,numberPlayers):
       profileLinks.append(profileLinksRaw[i].get('href',''))
```

```
285.
286.      #print(profileLinks)
287.      #print(len(profileLinks))
288.
289.
290.      for j in range(0,numberPlayers):
291.          fullPlayerLinks.append('https://www.transfermarkt.com' + profileLinks[j])
292.
293.
294.      count = 0
295.      for each in fullPlayerLinks:
296.          print("Player Page: " + str(count))
297.          playerPage = each
298.          playerPageTree = requests.get(playerPage, headers=headers)
299.          playerPageSoup = BeautifulSoup(playerPageTree.content,'html.parser')
300.          pageMissing = 0
301.          if "error message" in playerPageSoup.text and "Error | Transfermarkt" in playerPageSoup.text:
302.              print ("Page doesn't exist")
303.              pageMissing = 1
304.              masterNationalityList.append("Page does not exist")
305.              masterFootList.append("Page does not exist")
306.          if pageMissing == 0:
307.
308.
309.              ##----------------GET NATIONALITY-------------------------------------------------------------
310.              table = playerPageSoup.find_all("table", {"class": "auflistung"})
311.              if "Citizenship:" not in str(table):
312.                  masterNationalityList.append("No citizenship given")
313.              else:
314.                  nat1 = playerPageSoup.find(text="Citizenship:")
315.                  #if nat1 == "Citizenship:":
316.                  natParent = nat1.parent
317.                  #print('2')
318.                  #print(natParent)
319.
320.                  natTarget = natParent.parent
321.                  #print('3')
322.                  #print(natTarget)
323.
324.                  #print(natTarget.get_text(strip= True).replace("Citizenship:",''))
325.                  cleanedNationality = natTarget.get_text(strip= True).replace("Citizenship:",'')
326.                  #print(cleanedNationality)
327.
328.                  masterNationalityList.append(cleanedNationality)
329.
```

```
                #print('1')
                #print(nat1)


            ##----------------GET FOOT-----------------------------------------------
            table = playerPageSoup.find_all("table", {"class": "auflistung"})
            if "Foot:" not in str(table):
                masterFootList.append("No foot given")
            else:
                foot1 = playerPageSoup.find(text="Foot:")
                #print('1')
                #print(foot1)
                footParent = foot1.parent
                #print('2')
                #print(footParent)

                footTarget = footParent.parent
                #print('3')
                #print(footTarget)

                cleanedFoot = footTarget.get_text(strip= True).replace("Foot:",'')
                masterFootList.append(cleanedFoot)

        count = count + 1
    for each in fullPlayerLinks:
        majorLinks.append(each)


    ##NOW WE GET THE MV POINTS##
print("time to get mv points")
#print(len(majorLinks))
for j in range(0,len(majorLinks)):
    fullMVLinks.append(majorLinks[j].replace("profil","marktwertverlauf"))
#print(len(fullMVLinks))


counter12=1
for every in fullMVLinks:
    print("Market Value Chart: " + str(every))
    ##TAKE THIS OUT
    if every == "https://www.transfermarkt.com/marcello-gazzola/marktwertverlauf/spieler/57643":
        MVChart = "No Market Values"
    else:
        MVChart = getMVChart(every)
    if MVChart == "No Market Values":
```

```
            allMVCharts.append("No MV Chart")
        else:
            for each in MVChart:
                if "u00E9" in each[1]:
                    each[1] = each[1].replace("u00E9","é")
                if "u00E1" in each[1]:
                    each[1] = each[1].replace("u00E1","á")
                if "u00ED" in each[1]:
                    each[1] = each[1].replace("u00ED","í")
                if "u00F3" in each[1]:
                    each[1] = each[1].replace("u00F3","ó")
                if "u00FA" in each[1]:
                    each[1] = each[1].replace("u00FA","ú")
                if "u00FC" in each[1]:
                    each[1] = each[1].replace("u00FC","ü")
                if "u00F6" in each[1]:
                    each[1] = each[1].replace("u00F6","ö")
                if "u00E3" in each[1]:
                    each[1] = each[1].replace("u00E3","ã")
                if "u00E7" in each[1]:
                    each[1] = each[1].replace("u00E7","ü")
                if "u00E8" in each[1]:
                    each[1] = each[1].replace("u00E8","ü")
                if "u0219" in each[1]:
                    each[1] = each[1].replace("u0219","ș")
            allMVCharts.append(MVChart)
            #print(MVChart)
        counter12 = counter12 + 1

    #print(len(allMVCharts))
    #print(allMVCharts[0])

    print("Getting transfer charts")
    for j in range(0,len(majorLinks)):
        fullTransferLinks.append(majorLinks[j].replace("profil","transfers"))
    #print(len(fullTransferLinks))


    for every in fullTransferLinks:
        print("Transfer Chart: " + str(every))
        transferChart = getTransferChart(every)
        allTransferCharts.append(transferChart)

        #print(len(allTransferCharts))

```

```
#print(allMVCharts)
print("Comparing Now")
for j in range(0,len(allMVCharts)):
    print(j)
    #print(playerNames[j])
    transferAppended = False
    targetMVChart = allMVCharts[j]
    #print(targetMVChart)
    if targetMVChart == "No MV Chart":
        print("trigger1")
        t2MVs.append("No MV Chart")
        T2Date.append("No MV Chart")
        T2T1dateGaps.append("No MV Chart")
        masterTransferDates.append("No MV Chart")
        t1MVs.append("No MV Chart")
        indexOfTransfer.append("No MV Chart")
        numberOfTransfersBefore.append("No MV Chart")
        transferAppended = True
        #print("HIT NO MV")

        indexOfT2Points.append("No MV Chart")
        t3MVDates.append("No MV Chart")
        t3MVs.append("No MV Chart")
        T3T1DateGaps.append("No MV Chart")
        T3T2DateGaps.append("No MV Chart")
        indexOfT3Points.append("No MV Chart")

        t4MVDates.append("No MV Chart")
        t4MVs.append("No MV Chart")
        T4T1DateGaps.append("No MV Chart")
        T4T2DateGaps.append("No MV Chart")
        T4T3DateGaps.append("No MV Chart")
        indexOfT4Points.append("No MV Chart")


    else:
        compareOld = formerClubs[j]
        #print(compareOld)
        compareNew = newClubs[j]
        #print(compareNew)
        compareSeason = seasonList[j]
        #print(compareSeason)
        compareMV = mktValuesAtTrade[j]
        #print(compareMV)
        compareFee = transferFees[j]
```

```
465.
466.          #print(compareNew)
467.          #print(compareOld)
468.
469.          targetTransferChart = allTransferCharts[j]
470.          #print(targetTransferChart)
471.
472.          #find the specific transfer details from player's transfer chart
473.          for s in range(0,len(targetTransferChart)):
474.              if compareSeason == targetTransferChart[s][0] \
475.              and compareOld == targetTransferChart[s][2]\
476.              and compareNew == targetTransferChart[s][4]\
477.              and compareMV == targetTransferChart[s][6]\
478.              and compareFee == targetTransferChart[s][7]:
479.                  targetDate = targetTransferChart[s][1]
480.                  t1MVs.append(targetTransferChart[s][6])
481.                  masterTransferDates.append(targetDate)
482.                  indexOfTransfer.append(s)
483.                  numberOfTransfersBefore.append(len(targetTransferChart)-1-s)
484.                  #print(type(targetDate))
485.                  #print(targetDate)
486.                  transferAppended = True
487.                  #print(transferAppended)
488.
489.
490.          #weird case is where we couldn't find the transfer point
491.          weirdCase = False
492.
493.
494.          t2Found = False
495.          t3Found = False
496.          t4Found = False
497.
498.          #This if statement covers the one where it's not existent, or the transfer point just wasn't found
499.          if transferAppended == False:
500.              print("trigger2")
501.              #t2MVs.append("either before or after not on chart")
502.              masterTransferDates.append("could not find transfer point, check characters")
503.
504.          #print(masterTransferDates[j])
505.          #print("Length of MV Chart " + str(len(targetMVChart)))
506.
507.
508.          if masterTransferDates[j] == "could not find transfer point, check characters":
509.                  print("trigger 3")
```

```
                weirdCase = True
                t2MVs.append("transfer not found")
                T2Date.append("transfer not found")
                T2T1dateGaps.append("transfer not found")
                t1MVs.append("transfer not found")
                indexOfTransfer.append("transfer not found")
                #print("HIT NO MV")

                indexOfT2Points.append("transfer not found")
                t3MVDates.append("transfer not found")
                t3MVs.append("transfer not found")
                T3T1DateGaps.append("transfer not found")
                T3T2DateGaps.append("transfer not found")
                indexOfT3Points.append("transfer not found")
                numberOfTransfersBefore.append("transfer not found")

                t4MVDates.append("transfer not found")
                t4MVs.append("transfer not found")
                T4T1DateGaps.append("transfer not found")
                T4T2DateGaps.append("transfer not found")
                T4T3DateGaps.append("transfer not found")
                indexOfT4Points.append("transfer not found")
                t2Found = True
                t3Found = True
                t4Found = True

        else:
            t2Found = False
            t3Found = False
            t4Found = False
            weirdCase = False

            #FIND T2
            m = 0
            while m < len(targetMVChart) and t2Found == False:
                #if targetMVChart[m-1][1] == compareOld\
                if targetMVChart[m][1] == compareNew\
                and (targetMVChart[m][4] - masterTransferDates[j]).days < 120\
                and (targetMVChart[m][4] - masterTransferDates[j]).days > 0:
                    T2Date.append(targetMVChart[m][4])
                    t2MVs.append(targetMVChart[m][3])
                    T2T1dateGaps.append((targetMVChart[m][4] - masterTransferDates[j]).days)
                    indexOfT2Points.append(m)
                    #t2Appended = True
                    t2Found = True
```

```
                m = m + 1
            #print(t2Found)


            #DO NOT FIND T2
            if t2Found == False and weirdCase == False:
                T2Date.append("no point t2 matching criteria")
                t2MVs.append("no point t2 matching criteria")
                T2T1dateGaps.append("no point t2 matching criteria")
                indexOfT2Points.append("no point t2 matching criteria")


            #FIND T3, FOUND T2
            if t2Found == True and weirdCase == False:
                z = int(indexOfT2Points[j]) + 1
                while z < len(targetMVChart) and t3Found == False:
                    if targetMVChart[z][1] == compareNew\
                    and (targetMVChart[z][4] - masterTransferDates[j]).days >= 120\
                    and (targetMVChart[z][4] - masterTransferDates[j]).days < 300:
                        indexOfT3Points.append(z)
                        T3T1DateGaps.append((targetMVChart[z][4] - masterTransferDates[j]).days)
                        T3T2DateGaps.append((targetMVChart[z][4]-T2Date[j]).days)
                        t3MVs.append(targetMVChart[z][3])
                        t3MVDates.append(targetMVChart[z][4])
                        #t3Appended = True
                        t3Found = True
                    z = z + 1


            #FIND T3, DID NOT FIND T2
            if t2Found == False and weirdCase == False:
                r = 0
                while r < len(targetMVChart) and t3Found == False:
                    #if targetMVChart[r-1][1] == compareOld\
                    if targetMVChart[r][1] == compareNew\
                    and (targetMVChart[r][4] - masterTransferDates[j]).days >= 120\
                    and (targetMVChart[r][4] - masterTransferDates[j]).days < 300:
                        t3MVDates.append(targetMVChart[r][4])
                        t3MVs.append(targetMVChart[r][3])
                        T3T1DateGaps.append((targetMVChart[r][4] - masterTransferDates[j]).days)
                        T3T2DateGaps.append("No T2 Value")
                        indexOfT3Points.append(r)
                        t3Found = True
                        #t3AppendedRound2 = True
                        #print("HIT R FOR LOOP AKA t2 doesn't exist")
```

```
                    r = r + 1

            #DID NOT FIND T3
            if t3Found == False:
                t3MVDates.append('No T3 Found')
                t3MVs.append('No T3 Found')
                T3T1DateGaps.append('No T3 Found')
                T3T2DateGaps.append('No T3 Found')
                indexOfT3Points.append('No T3 Found')
                #print("HIT LAST RESORT, neither should exist")


            #FIND T4, FOUND T3, both did or did not find t2
            if t3Found == True and weirdCase == False:
                x = int(indexOfT3Points[j]) + 1
                while x < len(targetMVChart) and t4Found == False:
                    if targetMVChart[x][1] == compareNew\
                    and (targetMVChart[x][4] - masterTransferDates[j]).days >= 300\
                    and (targetMVChart[x][4] - masterTransferDates[j]).days < 480:
                        indexOfT4Points.append(x)
                        T4T1DateGaps.append((targetMVChart[x][4] - masterTransferDates[j]).days)
                        T4T3DateGaps.append((targetMVChart[x][4]-t3MVDates[j]).days)
                        t4MVs.append(targetMVChart[x][3])
                        t4MVDates.append(targetMVChart[x][4])
                        #t4Appended = True
                        t4Found = True
                        if t2Found == True:
                            T4T2DateGaps.append((targetMVChart[x][4]-T2Date[j]).days)
                        if t2Found == False:
                            T4T2DateGaps.append("No T2 Value")
                    x = x + 1

            #FIND T4, DID NOT FIND T3, FOUND T2
            if t3Found == False and t2Found == True and weirdCase == False:
                d = int(indexOfT2Points[j]) + 1
                while d < len(targetMVChart) and t4Found == False:
                    if targetMVChart[d][1] == compareNew\
                    and (targetMVChart[d][4] - masterTransferDates[j]).days >= 300\
                    and (targetMVChart[d][4] - masterTransferDates[j]).days < 480:
                        indexOfT4Points.append(d)
                        T4T1DateGaps.append((targetMVChart[d][4] - masterTransferDates[j]).days)
                        T4T2DateGaps.append((targetMVChart[d][4]-T2Date[j]).days)
                        T4T3DateGaps.append("No T3 Value")
                        t4MVs.append(targetMVChart[d][3])
                        t4MVDates.append(targetMVChart[d][4])
```

```
                        #t4Appended = True
                        t4Found = True
                    d = d + 1


            #FIND T4, DID NOT FIND T3, DID NOT FIND T2
            if t3Found == False and t2Found == False and weirdCase == False:
                l = 0
                while l < len(targetMVChart) and t4Found == False:
                    #if targetMVChart[l-1][1] == compareOld\
                    if targetMVChart[l][1] == compareNew\
                    and (targetMVChart[l][4] - masterTransferDates[j]).days >= 300\
                    and (targetMVChart[l][4] - masterTransferDates[j]).days < 480:
                        t4MVDates.append(targetMVChart[l][4])
                        t4MVs.append(targetMVChart[l][3])
                        T4T1DateGaps.append((targetMVChart[l][4] - masterTransferDates[j]).days)
                        T4T2DateGaps.append("No T2 Value")
                        T4T3DateGaps.append("To T3 Value")
                        indexOfT4Points.append(l)
                        t4Found = True
                        #print("HIT R FOR LOOP AKA t2 doesn't exist")
                    l = l + 1

            #DID NOT FIND T4
            if t4Found == False:
                t4MVDates.append('No T4 Found')
                t4MVs.append('No T4 Found')
                T4T1DateGaps.append('No T4 Found')
                T4T2DateGaps.append('No T4 Found')
                T4T3DateGaps.append('No T4 Found')
                indexOfT4Points.append('No T4 Found')
                #print("HIT LAST RESORT, neither should exist")

        #print("Done")
        #print("")

for j in range(0,len(allTransferCharts)):
    totals = sameClubBefore(allTransferCharts[j],indexOfTransfer[j],newClubs[j])
    beenOnBeforeList.append(totals[0])
    numBeenOnBefore.append(totals[1])
    daysSinceLastTransfer.append(totals[2])

#print(t2MVs)
```

```
#--------CLEANING THE FINANCIAL VALUES TO TAKE OFF EURO SIGN, M, AND CONVERT THOUSANDS TO MILLIONS---------
cleanedmktValuesAtTrade = cleaningFunction(mktValuesAtTrade)
cleanedtransferFees = cleaningFunction(transferFees)
cleanedT1List = cleaningFunction(t1MVs)
cleanedT2List = cleaningFunction(t2MVs)
cleanedT3List = cleaningFunction(t3MVs)
cleanedT4List = cleaningFunction(t4MVs)


#-------ADD NAME OF POSIITON ------------------------------------------------------------------------------
for k in range(0,len(playerNames)):
    if position == "Goalkeeper":
        goalkeeper.append(1)
        defender.append(0)
        midfielder.append(0)
        forward.append(0)
    elif position == "Defender":
        goalkeeper.append(0)
        defender.append(1)
        midfielder.append(0)
        forward.append(0)
    elif position == "Midfielder":
        goalkeeper.append(0)
        defender.append(0)
        midfielder.append(1)
        forward.append(0)
    elif position == "Forward":
        goalkeeper.append(0)
        defender.append(0)
        midfielder.append(0)
        forward.append(1)

    masterPositionList.append(str(position))

#-------PRINTING LIST LENGTHS TO CHECK THEY ARE ALL THE SAME ----------------------------------------------
print("Checking List Lengths")
print("List 1 Player Names: " + str(len(playerNames)))
print("List 2 Former Clubs: " + str(len(formerClubs)))
print("List 3 Former League: " + str(len(cleanedFromLeague)))
print("List 4 New Clubs: " + str(len(newClubs)))
print("List 5 New League: " + str(len(cleanedToLeague)))
print("List 6 Been On Before: " + str(len(beenOnBeforeList)))
print("List 7 # Times Been On Before: " + str(len(numBeenOnBefore)))
print("List 8 MVs At Trade: " + str(len(mktValuesAtTrade)))
print("List 9 Cleaned MVs At Trade: " + str(len(cleanedmktValuesAtTrade)))
```

```
print("List 10 Transfer Fees: " + str(len(transferFees)))
print("List 11 Cleaned Transfer Fees: " + str(len(cleanedtransferFees)))
print("List 12 Ages: " + str(len(masterAgeList)))

print("List 13 Nationalities: " + str(len(masterNationalityList)))

print("List 14 Position: " + str(len(masterPositionList)))
print("List 14.1 Goalkeeper Dummy: " + str(len(goalkeeper)))
print("List 14.2 Defender Dummy: " + str(len(defender)))
print("List 14.3 Midfielder Dummy: " + str(len(midfielder)))
print("List 14.4 Forward Dummy: " + str(len(forward)))

print("List 15 Foot: " + str(len(masterFootList)))

print("List 16 Season: " + str(len(seasonList)))
print("List 17 Transfer Date: " + str(len(masterTransferDates)))
print("List 17.1 Days Since Last Transfer: " + str(len(daysSinceLastTransfer)))

print("List 18 T1 MV: " + str(len(t1MVs)))
print("List 19 Cleaned T1 MV: " + str(len(cleanedT1List)))

print("List 20 T2 Date: " + str(len(T2Date)))
print("List 21 T1-T2 Date Gap: " + str(len(T2T1dateGaps)))
print("List 22 T2 MV: " + str(len(t2MVs)))
print("List 23 Cleaned T2 MV: " + str(len(cleanedT2List)))
print("List 24 T2 Point Index: " + str(len(indexOfT2Points)))

print("List 25 T3 Date: " + str(len(t3MVDates)))
print("List 26 T2-T3 Date Gap: " + str(len(T3T2DateGaps)))
print("List 27 T1-T3 Date Gap: " + str(len(T3T1DateGaps)))
print("List 28 T3 MV: " + str(len(t3MVs)))
print("List 29 Cleaned T3 MV: " + str(len(cleanedT3List)))
print("List 30 T3 Point Index: " + str(len(indexOfT3Points)))

print("List 31 T4 Date: " + str(len(t4MVDates)))
print("List 32 T4-T3 Date Gap: " + str(len(T4T3DateGaps)))
print("List 33 T4-T2 Date Gap: " + str(len(T4T2DateGaps)))
print("List 34 T4-T1 Date Gap: " + str(len(T4T1DateGaps)))
print("List 35 T4 MV: " + str(len(t4MVs)))
print("List 36 Cleaned T4 MV: " + str(len(cleanedT4List)))
print("List 37 T4 Point Index: " + str(len(indexOfT4Points)))

print("List 38 Index of Transfer: " + str(len(indexOfTransfer)))
print("List 39 # Transfers Before: " + str(len(numberOfTransfersBefore)))
print("List 40 Profile Links: " + str(len(majorLinks)))
```

```
print("List 41 MV Links: " + str(len(fullMVLinks)))
print("List 42 Transfer Links: " + str(len(fullTransferLinks)))


##------CREATE THE TABLE-----------------------------------------------------------------------------------

polishedTable = pd.DataFrame({playerNamesTitle:playerNames,\
                              formerClubsTitle:formerClubs,\
                              cleanedFromLeagueTitle:cleanedFromLeague,\
                              newClubsTitle:newClubs,\
                              cleanedToLeagueTitle:cleanedToLeague,\
                              beenOnBeforeListTitle:beenOnBeforeList,\
                              numBeenOnBeforeTitle:numBeenOnBefore,\
                              mktValuesAtTradeTitle:mktValuesAtTrade,\
                              cleanedmktValuesAtTradeTitle: cleanedmktValuesAtTrade,\
                              transferFeesTitle:transferFees,\
                              cleanedtransferFeesTitle:cleanedtransferFees,\
                              agesTitle:masterAgeList,\
                              nationalityTitle:masterNationalityList,\
                              positionTitle:masterPositionList,\
                              goalkeeperTitle:goalkeeper,\
                              defenderTitle:defender,\
                              midfielderTitle:midfielder,\
                              forwardTitle:forward,\
                              footTitle:masterFootList,\
                              seasonTitle:seasonList,\
                              transferDateTitle:masterTransferDates,\
                              daysSinceLastTransferTitle:daysSinceLastTransfer,\
                              t1MVsTitle:t1MVs,\
                              cleanedT1ListTitle:cleanedT1List,\
                              T2DateTitle:T2Date,\
                              T2T1dateGapsTitle:T2T1dateGaps,\
                              t2MVsTitle:t2MVs,\
                              cleanedT2ListTitle:cleanedT2List,\
                              indexOfT2PointsTitle:indexOfT2Points,\
                              t3MVDatesTitle:t3MVDates,\
                              T3T2DateGapsTitle:T3T2DateGaps,\
                              T3T1DateGapsTitle:T3T1DateGaps ,\
                              t3MVsTitle:t3MVs,\
                              cleanedT3ListTitle:cleanedT3List,\
                              indexOfT3PointsTitle:indexOfT3Points,\
                              t4MVDatesTitle:t4MVDates,\
                              T4T3DateGapsTitle:T4T3DateGaps,\
                              T4T2DateGapsTitle:T4T2DateGaps,\
```

```
                                T4T1DateGapsTitle:T4T1DateGaps,\
                                t4MVsTitle:t4MVs,\
                                cleanedT4ListTitle:cleanedT4List,\
                                indexOfT4PointsTitle:indexOfT4Points,\
                                indexOfTransferTitle:indexOfTransfer,\
                                numberOfTransfersBeforeTitle:numberOfTransfersBefore,\
                                majorLinksTitle:majorLinks,\
                                fullMVLinksTitle:fullMVLinks,\
                                fullTransferLinksTitle:fullTransferLinks
                                })

 headerNames = {playerNamesTitle,\
                 formerClubsTitle,\
                 cleanedFromLeagueTitle,\
                 newClubsTitle,\
                 cleanedToLeagueTitle,\
                 beenOnBeforeListTitle,\
                 numBeenOnBeforeTitle,\
                 mktValuesAtTradeTitle,\
                 cleanedmktValuesAtTradeTitle,\
                 transferFeesTitle,\
                 cleanedtransferFeesTitle,\
                 agesTitle,\
                 nationalityTitle,\
                 positionTitle,\
                 goalkeeperTitle,\
                 defenderTitle,\
                 midfielderTitle,\
                 forwardTitle,\
                 footTitle,\
                 seasonTitle,\
                 transferDateTitle,\
                 daysSinceLastTransferTitle,\
                 t1MVsTitle,\
                 cleanedT1ListTitle,\
                 T2DateTitle,\
                 T2T1dateGapsTitle,\
                 t2MVsTitle,\
                 cleanedT2ListTitle,\
                 indexOfT2PointsTitle,\
                 t3MVDatesTitle,\
                 T3T2DateGapsTitle,\
                 T3T1DateGapsTitle ,\
                 t3MVsTitle,\
                 cleanedT3ListTitle,\
```

```
                indexOfT3PointsTitle,
                t4MVDatesTitle,\
                T4T3DateGapsTitle,\
                T4T2DateGapsTitle,\
                T4T1DateGapsTitle,\
                t4MVsTitle,\
                cleanedT4ListTitle,\
                indexOfT4PointsTitle,\
                indexOfTransferTitle,\
                numberOfTransfersBeforeTitle,\
                majorLinksTitle,\
                fullMVLinksTitle,\
                fullTransferLinksTitle
                }


 polishedTable.to_excel("1011 10 March 2021.xlsx",header = headerNames, index=False)


 print('file created, done')
```

Function File 1: funcMVChart.py

Referred to in line 10 of Data Scraping.py: `from funcMVChart import getMVChart`

```
def getMVChart(playerLink):

    from bs4 import BeautifulSoup
    #import urllib3.request
    import requests
    import pandas as pd
    import re
    import js2xml
    import datetime
    from datetime import date
    from datetime import timedelta
```

```
    from unidecode import unidecode


    ##tell Transfermkt who I am
    headers = {'User-Agent':
               'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36'}


    ##ask user what input page
    page = str(playerLink)
    #print(page)
    #page = input("Enter transfer record page 1 URL, make sure ends in plus=1%2Fpage: ")
    #page = 'https://www.transfermarkt.com/neymar/marktwertverlauf/spieler/68290'
    #page = 'https://www.transfermarkt.com/philippe-coutinho/marktwertverlauf/spieler/80444'
    #page = 'https://www.transfermarkt.com/cristiano-ronaldo/marktwertverlauf/spieler/8198'
    #page = 'https://www.transfermarkt.com/lucas-hernandez/marktwertverlauf/spieler/281963'
    #page = 'https://www.transfermarkt.com/neymar/profil/spieler/68290'

    #print(page)
    ##fileName = input("Enter excel file path to save in")
    ##print(fileName)

    pageTree = requests.get(page, headers=headers)
    pageSoup = BeautifulSoup(pageTree.content, 'html.parser')


    rawValues = pageSoup.find_all('script',{"type":"text/javascript"})

    if len(rawValues) < 30:
        return "No Market Values"

    else:
        ##seems to be specifically the 29th script on each page feb 19 suddenly it's 30...
        specificScript = str(rawValues[30])

        splitValues = re.split(":{'",specificScript)
        #print(len(splitValues))
        symbolValues = []
        number = 0

        ##-----LOOP #1---- for looping through SCRIPT MESS to get the raw formatting markers that actually contain the data
        for each in splitValues:
            ##include the first point that has different formatting
            if splitValues[number][:37] == "enabled':true,'formatter':function(){":
                toSubtract = splitValues[number][0:779]
```

```
                firstValue = splitValues[number].replace(toSubtract,'')
                symbolValues.append(firstValue)

                number = number + 1

            ##exclude the last one that isn't a real point but still starts with symbol
            elif splitValues[number][:6] == "symbol" and splitValues[number][-9:] != "'credits'":
                symbolValues.append(splitValues[number])
                number = number + 1

            else:
                number = number+1


        ##define final array destination with hopefully the five useful ones
        niceTable = [[0] * 5]  * len(symbolValues)


        #print("NUMBER OF MARKERS " + str(len(symbolValues)))


        ##seem to be 7 parts to the symbolvalues, with delimiter ':, the URL, Y, veirein, age, mw, date, x, split each into parts
        row = 0
        col = 0

        ## ----TO GET THE FIRST VALUE IN THE LOOP BECAUSE OF DIF LENGTH----------------------------------------##

        #print(symbolValues[0])
        if len(symbolValues) > 0:
            initialPointSplit = re.split("':",str((symbolValues[0])))

            #number10 = 0
            #for every in initialPointSplit:
                #print (number10)
                #print(initialPointSplit[number10])
                #number10 = number10 + 1

            firstYValRaw = re.split(",",str(initialPointSplit[5]))
            firstCleanedYVal = str(firstYValRaw[0]).strip()

            firstClubRaw = str(initialPointSplit[6]).replace("age","")
            #print(firstClubRaw)
            firstClub = firstClubRaw.replace("\\","").replace("x20"," ").replace("\'","").replace(",","").replace("x2D","-")
              #print(firstClub)

```

```
            firstAge = ""
            for letter in initialPointSplit[7]:
                if letter.isdigit() == True:
                    firstAge = firstAge + letter

            buildFirstMV = ""
            toRemoveMV = ",'datum_mw"
            toRemove2MV = "\'"
            buildFirstMV = str(initialPointSplit[8]).replace("\\","").replace("u20AC","")
            buildFirstMV = buildFirstMV.replace(toRemoveMV,"")
            buildFirstMV = buildFirstMV.replace(toRemove2MV,"")

            firstDateAsArray = []
            firstDateAsArray = str(initialPointSplit[9]).split("\\x20")
            #print(dateAsArray)

            firstMonthNumber = ""
            #make month numerical
            firstMonth = str(firstDateAsArray[0])
            #print(firstMonth)
            if firstMonth == "'Jan":
                firstMonthNumber = "01"
            elif firstMonth == "'Feb":
                firstMonthNumber = "02"
            elif firstMonth == "'Mar":
                firstMonthNumber = "03"
            elif firstMonth == "'Apr":
                firstMonthNumber = "04"
            elif firstMonth == "'May":
                firstMonthNumber = "05"
            elif firstMonth == "'Jun":
                firstMonthNumber = "06"
            elif firstMonth == "'Jul":
                firstMonthNumber = "07"
            elif firstMonth == "'Aug":
                firstMonthNumber = "08"
            elif firstMonth == "'Sep":
                firstMonthNumber = "09"
            elif firstMonth == "'Oct":
                firstMonthNumber = "10"
            elif firstMonth == "'Nov":
                firstMonthNumber = "11"
            elif firstMonth == "'Dec":
                firstMonthNumber = "12"
            #print(monthNumber)
```

```
147.             #if len(firstMonthNumber) != 2:
148.                 #print("!!!!!!!!!!!!!!!!!!!!!! SOMETHING IS WRONG WITH THE MONTH!!BFUR(($)$(($()$(#()*#()#*#)(*#)()#")
149.
150.             firstDay = str(firstDateAsArray[1]).replace(",","")
151.             #make sure date is two digits
152.             if len(firstDay) == 1:
153.                 firstDay = "0" + firstDay
154.             #print(day)
155.
156.
157.             #if len(firstDay) != 2:
158.                 #print("!!!!!!!!!!!!!!!!!!!!!! SOMETHING IS WRONG WITH THE date !!BFUR(($)$(($()$(#()*#()#*#)(*#)()#")
159.
160.             firstYear = ""
161.             firstRawYear = str(firstDateAsArray[2])
162.             for letter4 in firstRawYear:
163.                 if letter4.isdigit() == True:
164.                     firstYear = firstYear + letter4
165.             #print(year)
166.             #if len(firstYear) != 4:
167.                 #print("!!!!!!!!!!!!!!!!!!!!!! SOMETHING IS WRONG WITH THE year!!BFUR(($)$(($()$(#()*#()#*#)(*#)()#")
168.
169. ##            print(firstYear)
170. ##            print(type(firstYear))
171. ##            print(firstMonthNumber)
172. ##            print(type(firstMonthNumber))
173. ##            print(firstDay)
174. ##            print(type(firstDay))
175.
176.
177.             #firstDateStitch = firstMonthNumber + "-" + firstDay + "-" + firstYear
178.             firstDateStitch = datetime.date(int(firstYear),int(firstMonthNumber),int(firstDay))
179.
180.             #print(dateStitch)
181.
182.             niceTable[0] = [firstCleanedYVal,firstClub, firstAge, buildFirstMV, firstDateStitch]
183.
184.             #print("FIRST VALUE IN  NICE TABLE" + str(niceTable))
185.
186.
187.
188.         ##-----LOOP #2---- for looping through to get the raw formatting markers and splitting into the 8/11 parts <-- SKIPS THE FIRST ONE
    THOUGH
189.         for j in range(1,len(symbolValues)):
190.             ##make pointsplits which is each of the 8-11 parts
```

```
            pointSplits = re.split("':",str((symbolValues[j])))
            #print(pointSplits)
            #print(str(j) + " --- LENGTH " + str(len(pointSplits)))


            ##-----LOOP #2.1---- for looping each of the 8-11 parts to format them nicely and store them in array called singleDataPoint
            for k in range(0,len(pointSplits)):

                #clean and save the numeric Y value aka market value
                if k == 2:
                    uncleanedYVals = re.split(",", str(pointSplits[k]))
                    cleanedYVal = str(uncleanedYVals[0]).strip()
                    #print(cleanedYVal)

                #clean and save the club name
                if k == 3:
                    step1club = str(pointSplits[k]).replace("age","")
                    #print(step1club)
                    step2club = step1club.replace("\\","").replace("x20"," ").replace("\'","").replace(",","").replace("x2D","-")
                    #print(step2club)

                #clean and save the age
                if k == 4:
                    buildAge = ""
                    for letter in pointSplits[k]:
                        if letter.isdigit() == True:
                            buildAge = buildAge + letter

                #clean and save the market value in millions
                if k == 5:
                    buildMV2 = ""
                    toRemove = ",'datum_mw"
                    toRemove2 = "\'"
                    #take off backslash and u20AC
                    buildMV = str(pointSplits[k]).replace("\\","").replace("u20AC","")
                    #take off datum_mw
                    buildMV = buildMV.replace(toRemove,"")
                    #take off starting and ending single quotes '
                    buildMV = buildMV.replace(toRemove2,"")
                    #print(buildMV)

                #clean and save the date of measurement in AMERICAN MM/DD/YYYY FORMAT
                if k == 6:
                    dateAsArray = []
                    dateAsArray = str(pointSplits[k]).split("\\x20")
```

```
                    #print(dateAsArray)

                    monthNumber = ""
                    #make month numerical
                    month = str(dateAsArray[0])
                    if month == "'Jan":
                        monthNumber = "01"
                    elif month == "'Feb":
                        monthNumber = "02"
                    elif month == "'Mar":
                        monthNumber = "03"
                    elif month == "'Apr":
                        monthNumber = "04"
                    elif month == "'May":
                        monthNumber = "05"
                    elif month == "'Jun":
                        monthNumber = "06"
                    elif month == "'Jul":
                        monthNumber = "07"
                    elif month == "'Aug":
                        monthNumber = "08"
                    elif month == "'Sep":
                        monthNumber = "09"
                    elif month == "'Oct":
                        monthNumber = "10"
                    elif month == "'Nov":
                        monthNumber = "11"
                    elif month == "'Dec":
                        monthNumber = "12"
                    #print(monthNumber)
                    #if len(monthNumber) != 2:
                        #print("!!!!!!!!!!!!!!!!!!!!!!! SOMETHING IS WRONG WITH THE MONTH!!BFUR(($)$(($()$(#()*#()#*#)(*#)()#")

                    day = str(dateAsArray[1]).replace(",","")
                    #make sure date is two digits
                    if len(day) == 1:
                        day = "0" + day
                    #print(day)

                    #if len(day) != 2:
                        #print("!!!!!!!!!!!!!!!!!!!!!!! SOMETHING IS WRONG WITH THE date !!BFUR(($)$(($()$(#()*#()#*#)(*#)()#")

                    year = ""
                    rawYear = str(dateAsArray[2])
                    for letter4 in rawYear:
```

```
                        if letter4.isdigit() == True:
                            year = year + letter4
                    #print(year)
                    #if len(year) != 4:
                        #print("!!!!!!!!!!!!!!!!!!!!!!!! SOMETHING IS WRONG WITH THE year!!BFUR(($)$(($()$(#()*#()#*#)(*#)()#")


                    dateStitch = datetime.date(int(year),int(monthNumber),int(day))
                    #print(dateStitch)

            niceTable[j] = [cleanedYVal, step2club, buildAge, buildMV, dateStitch]
        #print(niceTable)
        #print(len(niceTable))
        #finnum = 0
        #for every in niceTable:
            #print (finnum)
            #print(niceTable[finnum])
            #finnum = finnum + 1
        #print("----------NEXT POINT--------")

        #print("The following player MV chart has been saved")
        #print (niceTable)
        return niceTable

def printMVPretty(chart):
    ## ----- LOOP  #2 ----- FOR LOOPING THROUGH
    number1 = 0
    for every in chart:
        print ("Index #: " + str(number1))
        print(chart[number1])
        print(" ")
        number1 = number1 + 1
```

Function File 2: transfers.py

Referred to in line 11 of Data Scraping.py: `from transfers import getTransferChart`

```
1.   def getTransferChart(playerPage):
2.      from bs4 import BeautifulSoup
3.      #import urllib3.request
4.      import requests
5.      import pandas as pd
6.      import re
7.      import js2xml
8.      import datetime
9.      from datetime import date
10.     from datetime import timedelta
11.
12.     ##tell Transfermkt who I am
13.     headers = {'User-Agent':
14.                'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36'}
15.     page = playerPage
16.
17.     pageTree = requests.get(page, headers=headers)
18.     pageSoup = BeautifulSoup(pageTree.content, 'html.parser')
19.
20.     rawTransfers = pageSoup.find_all("tr",{"class":"zeile-transfer"})
21.     #print(rawTransfers[0])
22.     #print(len(rawTransfers))
23.     numberTrans = len(rawTransfers)
24.     #print(rawTransfers)
25.
26.     transferTable = [[0] * 6]  * numberTrans
27.
28.     ##for i in range(0,numberTrans):
29.     ##    print(transferTable[i])
30.     ##    print("")
31.
32.     ##names = []
33.     ##firstSeason = rawTransfers[0].find("td",{"class":"zentriert"})
34.     ##nextPoint = firstSeason.find_next_sibling()
35.     ##nextPoint1 = firstSeason.next_elements
36.     ##print(nextPoint1)
37.     ##print(rawTransfers[0].find("img").descendents)
38.     ##print(rawTransfers[0].find_all("img")[2].get('alt'))
39.     ##print(len(rawTransfers[0].find_all("img")))
40.     dateAsArray = []
41.
42.
43.     for j in range(0, numberTrans):
44.         club = rawTransfers[j].find("td",{"class":"zentriert"}).text
45.         rawDate = rawTransfers[j].find("td",{"class":"zentriert"}).find_next_sibling().text
```

```
46.         dateAsArray = re.split(" ",str(rawDate))
47.         #print(dateAsArray)
48.         monthNumber = ""
49.         month = str(dateAsArray[0])
50.         if month == "Jan":
51.             monthNumber = "01"
52.         elif month == "Feb":
53.             monthNumber = "02"
54.         elif month == "Mar":
55.             monthNumber = "03"
56.         elif month == "Apr":
57.             monthNumber = "04"
58.         elif month == "May":
59.             monthNumber = "05"
60.         elif month == "Jun":
61.             monthNumber = "06"
62.         elif month == "Jul":
63.             monthNumber = "07"
64.         elif month == "Aug":
65.             monthNumber = "08"
66.         elif month == "Sep":
67.             monthNumber = "09"
68.         elif month == "Oct":
69.             monthNumber = "10"
70.         elif month == "Nov":
71.             monthNumber = "11"
72.         elif month == "Dec":
73.             monthNumber = "12"
74.         day = str(dateAsArray[1]).replace(",","")
75.         #make sure date is two digits
76.         if len(day) == 1:
77.             day = "0" + day
78.         dateStitch = datetime.date(int(dateAsArray[2]),int(monthNumber),int(day))
79.         formerClub = rawTransfers[j].find("img").get('alt')
80.         newClub = rawTransfers[j].find_all("img")[2].get('alt')
81.         #print(newClub)
82.         marketValue = rawTransfers[j].find("td",{"class":"zelle-mw"}).text
83.         transferFee = rawTransfers[j].find("td",{"class":"zelle-abloese"}).text
84.
85.         if newClub == "Retired" or newClub == "Without Club" or newClub == "Career Break" or newClub == "Disqualification" or newClub ==
    None:
86.             newCountry = "N/A"
87.         else:
88.             newCountry = rawTransfers[j].find_all("img")[3].get('alt')
89.
```

```
        if formerClub == "Retired" or formerClub == "Without Club" or formerClub == "Career Break" or formerClub == "Disqualification" or formerClub == None:
            formerCountry = "N/A"
        else:
            formerCountry = rawTransfers[j].find_all("img")[1].get('alt')

        transferTable[j] = [club, dateStitch, formerClub, formerCountry, newClub, newCountry, marketValue, transferFee]

    return transferTable


```

Function File 3: cleaningFunction.py

Referred to in line 12 of Data Scraping.py: `from cleaningFunction import cleaningFunction`

```
 def cleaningFunction(array):
    cleanedArray = []
    for each in array:
        if 'Loan fee' in each \
           or 'loan' in each \
           or 'free transfer' in each\
           or '-' in each\
           or "no point" in each\
           or "No MV Chart" in each\
           or 'could not find' in each\
           or 'not on chart' in each\
           or "No T3 Found" in each\
           or "transfer not found" in each\
           or "no point t2 matching criteria"in each\
           or "No T4 Found" in each\
           or "x2D" in each:
            cleanedArray.append(each)
            continue

        if "Th." in each:
            #print(each)
            strippedNum = str(each).replace('€','').replace('Th.','')
            #print(strippedNum)
            finalNum = int(strippedNum)/1000
            cleanedArray.append(finalNum)

```

```
            #print(finalNum)
        if "m" in each:
            finalNum = str(each).replace('€','').replace('m','')
            cleanedArray.append(finalNum)


    #print(array)
    return(cleanedArray)

```

Function File 4: sameClubBefore.py

Referred to in line 13 of Data Scraping.py: `from sameClubBefore import sameClubBefore`

```
def sameClubBefore(transferChart,numTransfer,newClub):
    from bs4 import BeautifulSoup
    #import urllib3.request
    import requests
    import pandas as pd
    import re
    import js2xml
    from datetime import date
    from datetime import timedelta
    from transfers2 import getTransferChart2
    from funcMVChart import getMVChart


    #playerPage = 'https://www.transfermarkt.com/julio-cesar/profil/spieler/22412'
    #mvPage = 'https://www.transfermarkt.com/philippe-coutinho/marktwertverlauf/spieler/80444'

    #transferChart = getTransferChart2(str(playerPage))
    #mvChart = getMVChart(mvPage)
    #print(mvChart)
    #print(transferChart)

    if numTransfer == "No MV Chart" :
        return ["No MV Chart from before", "No MV Chart from Before", "No MV Chart from Before"]
    elif numTransfer ==  "could not find transfer point, check characters":
```

```
        return ["could not find transfer point", "could not find transfer point", "could not find transfer point"]
    elif numTransfer == "transfer not found":
        return ["transfer not found","transfers not found","transfers not found"]
    else:
        daysSinceLastTransfer = "Not assigned"
        listOfFormerClubs = []
        lookingFor = newClub
        beenOnBefore = 0
        indexOfTransfer = numTransfer
        howManyTimesOnClubBefore = 0

        counter = len(transferChart)
        for each in range(indexOfTransfer,len(transferChart)):
            #print(each[3])
            #print(type(each[3]))
            if transferChart[each][2] == lookingFor:
                beenOnBefore = 1
                howManyTimesOnClubBefore = howManyTimesOnClubBefore + 1

        if (indexOfTransfer+1) < len(transferChart):
            daysSinceLastTransfer = (transferChart[indexOfTransfer][1]-transferChart[indexOfTransfer+1][1]).days

        if (indexOfTransfer+1) == len(transferChart):
            daysSinceLastTransfer = "Last transfer on chart"


        return [beenOnBefore, howManyTimesOnClubBefore,daysSinceLastTransfer]
```