In [1]:
```
# Select folder path based on user input
gender = 'M' #input('Enter gender (W for women, M for men): ')

# Assign the appropriate folder path based on the input
MAIN_DIR = './'
USE_DIR = MAIN_DIR + 'womens/' if gender.upper() == 'W' else MAIN_DIR + 'mer
PRE = 'W' if gender.upper() == 'W' else 'M'
NAME = 'womens' if gender.upper() == 'W' else 'mens'
```

In [2]:
```
# Import Libraries
import pandas as pd
from datetime import datetime, timedelta
import re
import random
import numpy as np
```

Data Section 3 - GAMES

In [3]:
```
# Create Seasons dataframe
seasons = pd.read_csv(USE_DIR + PRE + 'Seasons.csv')

# Convert DayZero column to datetime format
seasons['DayZero'] = pd.to_datetime(seasons['DayZero'])

# Show sample output
print(len(seasons.Season.unique()), 'seasons')
print('range:', seasons.Season.unique())
seasons.head(2)
```

```
39 seasons
range: [1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 19
98
 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023]
```

Out[3]:

	Season	DayZero	RegionW	RegionX	RegionY	RegionZ
0	1985	1984-10-29	East	West	Midwest	Southeast
1	1986	1985-10-28	East	Midwest	Southeast	West

In [4]:
```
# Create Seeds dataframe
seeds = pd.read_csv(USE_DIR + PRE + 'NCAATourneySeeds.csv')

# Create a regex to pull out number
regex = r'\d+'

# Apply the regular expression to the 'col1' column to extract the numeric v
seeds['Seeds'] = seeds['Seed'].apply(lambda x: re.search(regex, x).group())

# Convert the numeric values to integers
seeds['Seeds'] = seeds['Seeds'].astype(int)

# Show sample output
```

```
print('seeds:', seeds.shape)
print(seeds.head(3))
```

```
seeds: (2490, 4)
   Season Seed  TeamID  Seeds
0    1985  W01    1207      1
1    1985  W02    1210      2
2    1985  W03    1228      3
```

In [5]:
```
# Create Games dataframe
games = pd.read_csv(USE_DIR + PRE + 'RegularSeasonDetailedResults.csv')
tourney = pd.read_csv(USE_DIR + PRE + 'NCAATourneyDetailedResults.csv')
```

In [6]:
```
# Show sample output
print('games:', games.shape, 'tourney:', tourney.shape)
print(len(games.Season.unique()), 'seasons')
print('range:', games.Season.unique())
```

```
games: (107634, 34) tourney: (1248, 34)
21 seasons
range: [2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 20
16
 2017 2018 2019 2020 2021 2022 2023]
```

In [7]:
```
# Merge the 'seasons' and 'games' dataframes
merged_dfg = pd.merge(seasons, games, on='Season')
merged_dft = pd.merge(seasons, tourney, on='Season')

# Add 'DayNum' to 'DayZero' to create 'DayDate' column
merged_dfg['DayDate'] = pd.to_datetime(merged_dfg['DayZero']) + pd.to_timede
merged_dft['DayDate'] = pd.to_datetime(merged_dft['DayZero']) + pd.to_timede

# Create new 'DayDate' column in 'games' dataframe
games['DayDate'] = merged_dfg['DayDate']
tourney['DayDate'] = merged_dft['DayDate']
```

In [8]:
```
# Set order sequence
order = [0, 34, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1
         20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33]

# Reorder and drop unwanted columns from games dataframe
games = games.iloc[:,order].drop(columns=['DayNum'])
tourney = tourney.iloc[:,order].drop(columns=['DayNum'])
```

In [9]:
```
# Add new 'Outcome' column to identify which team won (to be used to order c
games['Outcome'] = np.where(games['WTeamID'] < games['LTeamID'], 0, 1)
tourney['Outcome'] = np.where(tourney['WTeamID'] < tourney['LTeamID'], 0, 1)

# Add new 'Team0' and 'Team1' columns to put teams in the proper order for p
games['Team0'] = np.where(games['WTeamID'] < games['LTeamID'], games['WTeamI
games['Team1'] = np.where(games['WTeamID'] < games['LTeamID'], games['LTeamI
tourney['Team0'] = np.where(tourney['WTeamID'] < tourney['LTeamID'], tourney
tourney['Team1'] = np.where(tourney['WTeamID'] < tourney['LTeamID'], tourney

# Add new 'Site' column to assign a numeric value to the site where game was
games['Site'] = np.where(games['WLoc'] == 'A', -1, np.where(games['WLoc'] ==
```

```
tourney['Site'] = np.where(tourney['WLoc'] == 'A', -1, np.where(tourney['WLc
tourney['Site'] = np.where(tourney['WLoc'] == 'A', -1, np.where(tourney['WLc
```

In [10]:
```
# Set order sequence
order = [0, 35, 36, 1, 37, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1
         20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34]

# Reorder columns from games dataframe
games = games.iloc[:,order]
tourney = tourney.iloc[:,order]
```

In [11]:
```
# Create a series of new columns (13 total) for the differenitals between ea
adjust_games = games.copy()
adjust_tourney = tourney.copy()

# MOV, Margin of Victory (Score Differential)
adjust_games.loc[:, 'MOV'] = np.where(games['Outcome'] == 0, games['WScore']
adjust_tourney.loc[:, 'MOV'] = np.where(tourney['Outcome'] == 0, tourney['WS

# FG2M, Field Goals (2-pts) Made Differential
adjust_games.loc[:, 'FG2M'] = np.where(games['Outcome'] == 0, games['WFGM']
adjust_tourney.loc[:, 'FG2M'] = np.where(tourney['Outcome'] == 0, tourney['W

# FG2A, Field Goals (2-pts) Attempted Differential
adjust_games.loc[:, 'FG2A'] = np.where(games['Outcome'] == 0, games['WFGA']
adjust_tourney.loc[:, 'FG2A'] = np.where(tourney['Outcome'] == 0, tourney['W

# FG3M, Field Goals (3-pts) Made Differential
adjust_games.loc[:, 'FG3M'] = np.where(games['Outcome'] == 0, games['WFGM3']
adjust_tourney.loc[:, 'FG3M'] = np.where(tourney['Outcome'] == 0, tourney['W

# FG3A, Field Goals (2-pts) Attempted Differential
adjust_games.loc[:, 'FG3A'] = np.where(games['Outcome'] == 0, games['WFGA3']
adjust_tourney.loc[:, 'FG3A'] = np.where(tourney['Outcome'] == 0, tourney['W

# FT1M, Free Throws (1-pt) Made Differential
adjust_games.loc[:, 'FT1M'] = np.where(games['Outcome'] == 0, games['WFTM']
adjust_tourney.loc[:, 'FT1M'] = np.where(tourney['Outcome'] == 0, tourney['W

# FT1A, Free Throws (1-pt) Attempted Differential
adjust_games.loc[:, 'FT1A'] = np.where(games['Outcome'] == 0, games['WFTA']
adjust_tourney.loc[:, 'FT1A'] = np.where(tourney['Outcome'] == 0, tourney['W

# ORB, Offensive Rebounds Differential
adjust_games.loc[:, 'ORB'] = np.where(games['Outcome'] == 0, games['WOR'] -
adjust_tourney.loc[:, 'ORB'] = np.where(tourney['Outcome'] == 0, tourney['WO

# DRB, Defensive Rebounds Differential
adjust_games.loc[:, 'DRB'] = np.where(games['Outcome'] == 0, games['WDR'] -
adjust_tourney.loc[:, 'DRB'] = np.where(tourney['Outcome'] == 0, tourney['WD

# AST, Assists Differential
adjust_games.loc[:, 'AST'] = np.where(games['Outcome'] == 0, games['WAst'] -
adjust_tourney.loc[:, 'AST'] = np.where(tourney['Outcome'] == 0, tourney['WA
```

```
# TOVR, Turnovers Differential
adjust_games.loc[:, 'TOVR'] = np.where(games['Outcome'] == 0, games['WTO'] -
adjust_tourney.loc[:, 'TOVR'] = np.where(tourney['Outcome'] == 0, tourney['W

# STL, Steals Differential
adjust_games.loc[:, 'STL'] = np.where(games['Outcome'] == 0, games['WStl'] -
adjust_tourney.loc[:, 'STL'] = np.where(tourney['Outcome'] == 0, tourney['WS

# BLK, Block Shots Differential
adjust_games.loc[:, 'BLK'] = np.where(games['Outcome'] == 0, games['WBlk'] -
adjust_tourney.loc[:, 'BLK'] = np.where(tourney['Outcome'] == 0, tourney['WB

# PFL, Personal Fouls (Team Totals) Differential
adjust_games.loc[:, 'PFL'] = np.where(games['Outcome'] == 0, games['WPF'] -
adjust_tourney.loc[:, 'PFL'] = np.where(tourney['Outcome'] == 0, tourney['WP
```

In [12]:
```
# Set order sequence
order = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1
         27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44,

drop_columns = ['WTeamID', 'WScore', 'LTeamID', 'LScore', 'WLoc', 'NumOT', '
                'WFTM', 'WFTA', 'WOR', 'WDR', 'WAst', 'WTO', 'WStl', 'WBlk',
                'LFGA3', 'LFTM', 'LFTA', 'LOR', 'LDR', 'LAst', 'LTO', 'LStl'

# Reorder columns from games dataframe
games = adjust_games.iloc[:,order].drop(columns=drop_columns).copy()
tourney = adjust_tourney.iloc[:,order].drop(columns=drop_columns).copy()
```

In [13]:
```
# Show sample output
print('games:', adjust_games.shape, 'tourney:', adjust_tourney.shape)
print(adjust_games.columns)
adjust_games.head()
```

```
games: (107634, 52) tourney: (1248, 52)
Index(['Season', 'Team0', 'Team1', 'DayDate', 'Site', 'WTeamID', 'WScore',
       'LTeamID', 'LScore', 'WLoc', 'NumOT', 'WFGM', 'WFGA', 'WFGM3', 'WFGA
3',
       'WFTM', 'WFTA', 'WOR', 'WDR', 'WAst', 'WTO', 'WStl', 'WBlk', 'WPF',
       'LFGM', 'LFGA', 'LFGM3', 'LFGA3', 'LFTM', 'LFTA', 'LOR', 'LDR', 'LAs
t',
       'LTO', 'LStl', 'LBlk', 'LPF', 'Outcome', 'MOV', 'FG2M', 'FG2A', 'FG3
M',
       'FG3A', 'FT1M', 'FT1A', 'ORB', 'DRB', 'AST', 'TOVR', 'STL', 'BLK',
       'PFL'],
      dtype='object')
```

Out[13]:

	Season	Team0	Team1	DayDate	Site	WTeamID	WScore	LTeamID	LScore	WLoc	...
0	2003	1104	1328	2002-11-14	0	1104	68	1328	62	N	...
1	2003	1272	1393	2002-11-14	0	1272	70	1393	63	N	...
2	2003	1266	1437	2002-11-15	0	1266	73	1437	61	N	...
3	2003	1296	1457	2002-11-15	0	1296	56	1457	50	N	...
4	2003	1208	1400	2002-11-15	0	1400	77	1208	71	N	...

In [14]:
```
# Merge the 'Tourney' and 'Seeds' dataframes on 'Season' and 'TeamID' columr
seeding1 = pd.merge(tourney, seeds, how='left', left_on=['Season', 'Team0'],
seeding1.drop(columns='TeamID', inplace=True)

seeding2 = pd.merge(seeding1, seeds, how='left', left_on=['Season', 'Team1']
seeding2.drop(columns='TeamID', inplace=True)

# Seed, Tournament Seeding Consideration (Differential)
games['Seed'] = 0 # holder ('fake' seed differential)
seeding2.loc[:, 'Seed'] = np.where(seeding2['Outcome'] == 0, seeding2['Seeds

# Set order sequence
order_games = [0, 1, 2, 3, 20, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
order_tourney = [0, 1, 2, 3, 24, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1

# Reorder columns from games dataframe
games = games.iloc[:,order_games].copy()
tourney = seeding2.iloc[:,order_tourney].drop(columns=['Seed_x', 'Seeds_x',
```

In [15]:
```
# Show sample output
print('games:', games.shape)
print(games.columns)
games.head()
```

```
games: (107634, 21)
Index(['Season', 'Team0', 'Team1', 'DayDate', 'Seed', 'Site', 'MOV', 'FG2
M',
       'FG2A', 'FG3M', 'FG3A', 'FT1M', 'FT1A', 'ORB', 'DRB', 'AST', 'TOVR',
       'STL', 'BLK', 'PFL', 'Outcome'],
      dtype='object')
```

Out[15]:

	Season	Team0	Team1	DayDate	Seed	Site	MOV	FG2M	FG2A	FG3M	...	FT1M	FT1
0	2003	1104	1328	2002-11-14	0	0	6	5	5	1	...	-5	-
1	2003	1272	1393	2002-11-14	0	0	7	2	-5	2	...	1	-
2	2003	1266	1437	2002-11-15	0	0	12	2	-15	5	...	3	
3	2003	1296	1457	2002-11-15	0	0	6	0	-11	-3	...	9	1
4	2003	1208	1400	2002-11-15	0	0	-6	-6	1	0	...	6	1

In [16]:
```
print('seed differentials:', games.Seed.unique())
```

```
seed differentials: [0]
```

In [17]:
```
# Show sample output
print('tourney:', tourney.shape)
print(games.columns)
games.head()
```

```
tourney: (1248, 21)
Index(['Season', 'Team0', 'Team1', 'DayDate', 'Seed', 'Site', 'MOV', 'FG2
M',
       'FG2A', 'FG3M', 'FG3A', 'FT1M', 'FT1A', 'ORB', 'DRB', 'AST', 'TOVR',
       'STL', 'BLK', 'PFL', 'Outcome'],
      dtype='object')
```

Out[17]:

	Season	Team0	Team1	DayDate	Seed	Site	MOV	FG2M	FG2A	FG3M	...	FT1M	FT1
0	2003	1104	1328	2002-11-14	0	0	6	5	5	1	...	-5	-
1	2003	1272	1393	2002-11-14	0	0	7	2	-5	2	...	1	-
2	2003	1266	1437	2002-11-15	0	0	12	2	-15	5	...	3	
3	2003	1296	1457	2002-11-15	0	0	6	0	-11	-3	...	9	1
4	2003	1208	1400	2002-11-15	0	0	-6	-6	1	0	...	6	1

5 rows × 21 columns

In [18]:
```
print('seed differentials:', tourney.Seed.unique())
```

```
seed differentials: [  0 -15   3   5  -1  -7 -11   1  -9 -13  -5   9   7  -
3  -8   8  -4   2
  -6   4  11  -2 -12  10   6  13 -10  15  12]
```

In [19]:

```
# Save dataframes as csv files
games_data = games.to_csv('games-' + NAME + '.csv',index=False)
tourney_data = tourney.to_csv('tourney-' + NAME + '.csv',index=False)
```