```
In [210]: %matplotlib inline

          import csv
          import os
          import requests
          import warnings
          import matplotlib.pyplot as plt
          import numpy as np
          import pandas as pd
          import seaborn as sns
          from bs4 import BeautifulSoup
          from datetime import datetime
          from requests import get
          from urllib.request import urlopen

          sns.set(color_codes=True)
```

```
In [211]: os.getcwd()

Out[211]: 'C:\\Users\\Akuma2099\\MachineLearning\\QTW_Project_2'
```

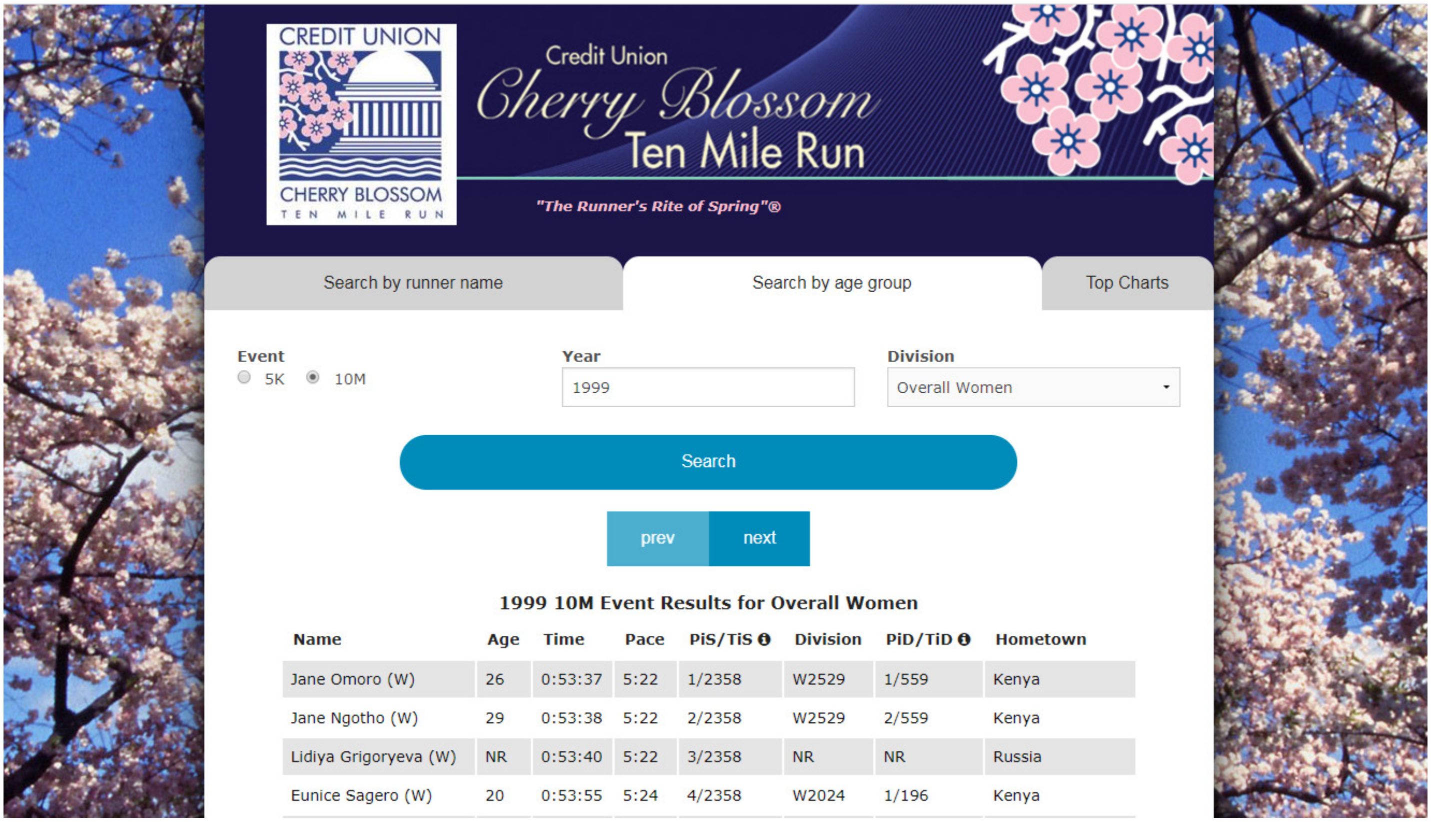
Introduction

The Credit Union Cherry Blossom Ten Mile Run is an annual run held in Washington, D.C. that brings together credit unions from across the country with a shared vision of fund raising and helping children. The proceeds from donations, registration fees, and merchandise sales from the event support Children's Hospitals that belong to the non-profit Children's Miracle Network.

Since the innuagural run in 1973, hundreds of thousands of runners from around the world have participated in the event. The registration of each runners name, age, hometown, and race times provide an enormous trove of information regarding potential trends in the age and performance of runners over the years. The organizations website http://www.cherryblossom.org/ (http://www.cherryblossom.org/) provides records of this data going back to 1999. This analysis will use web scraping to collect data from the website, inspect and clean the data, convert the data into a clean data frame, conduct exploratory data analysis and determine if ...

Data

Data for this analysis was taken from the Credit Union Cherry Blossom Ten Mile Run sites searcheable results web page using the URL http://www.cballtimeresults.org/performances?division=Overall+Women&page=1&section=10M&sex=W&utf8=%E2%9C%93&year=1999 (http://www.cballtimeresults.org/performances?division=Overall+Women&page=1&section=10M&sex=W&utf8=%E2%9C%93&year=1999) as our initial reference for our web scraping function.



Event ○ 5K ● 10M

Year: 1999

Division: Overall Women

1999 10M Event Results for Overall Women

Name	Age	Time	Pace	PiS/TiS	Division	PiD/TiD	Hometown
Jane Omoro (W)	26	0:53:37	5:22	1/2358	W2529	1/559	Kenya
Jane Ngotho (W)	29	0:53:38	5:22	2/2358	W2529	2/559	Kenya
Lidiya Grigoryeva (W)	NR	0:53:40	5:22	3/2358	NR	NR	Russia
Eunice Sagero (W)	20	0:53:55	5:24	4/2358	W2024	1/196	Kenya

Each page yielded only 20 runners and the number of female runners per year varied from 2,166 to 11,042, however from page to page the URL remained consistant making it relatively simple to iteratively cycle through the **page=** and **year=** variables of the URL. After an examination of the page source it was determined that the values: *"Year","Name", "Age", "Time","Pace","PiS/TiS", "Division", "PiD/TiD", and "Hometown"* could be obtained for each runner. A CSV file was the created with the appropriate headers to store the scraped data.

In [109]:

```
#create outputfile
outputFile = "CherryBlossomWomens10M.csv"

#create headers
file = open("CherryBlossomWomens10M.csv", "w",encoding='utf8' )
writer = csv.DictWriter(
    file, fieldnames=["Year","Name", "Age", "Time","Pace","PiS/TiS", "Division", "PiD/TiD", "Hometown", 'State','Blank1','Blank
2',])
writer.writeheader()
file.close()
```

In [108]:

```
#split the marathon url into two parts to format page number and year
websiteURLpart1 = "http://www.cballtimeresults.org/performances?division=Overall+Women&page="
websiteURLpart2 = "&section=10M&sex=W&utf8=%E2%9C%93&year="
```

Through an examination of the page source it was determined that 'tr class='print-link-color' indicated entries for a new runner and two functions where created. The first function would update the URL **page=** and **year=** and the second function would parse the URL calling the first function to update the URL when no more 'tr class='print-link-color' entries where found on the page and update the year when the end page was reached.

In [110]:

```
#create url function to format the url that is then passed to requests to get the html page
def createURL(url1, url2, pageNum, year):
    return(url1 + str(pageNum) + url2 + str(year))

#create parsing function that uses createURL to iterate through pages
def getRunners(url1, url2):
    year = 1999
    #iterate through all years from 1999 to 2018
    while(year < 2019):
        pageNum = 1
        morePages = True
        #iterate through all pages in each year
        while(morePages):
            websiteURL = createURL(url1, url2, pageNum, year)
            with open(outputFile, "a", encoding='utf8') as f:
                #requests return the html of the page in a raw object
                page = requests.get(websiteURL)
                #Beautiful Soup parses the requests object into a better formatted html object
                soup = BeautifulSoup(page.content, "html.parser")
                newLine = ""
                contents = soup.find_all("tr", class_="print-link-color")
                #check if the table is empty, if so, go to the next year
                if not contents:
                    morePages = False
                    year += 1
                #if the table has content
                else:
                    #iterate through all of the table rows that have class=print-link-color
                    for tr in soup.find_all("tr", class_="print-link-color"):
                        #add the previously parsed line to the file
                        if(newLine != ""):
                            f.write(newLine.rstrip(","))
                        newLine = ""
                        f.write("\n")
                        firsta = True
                        for a in tr.select("td a"):
                            if(firsta):
                                f.write(a.getText().split()[0])
                                firsta = False
                            else:
                                f.write(a.getText())
                            f.write(",")
                    #once all of the table rows for this page are parsed, go to the next page
                    pageNum += 1
                    f.close()
    return(True)
```

In [111]:

```
#run getRunners function to scrape website
getRunners(websiteURLpart1, websiteURLpart2)
```

Out[111]:

```
True
```

Data Preperation

In [112]:
```
df = pd.read_csv('CherryBlossomWomens10M.csv', header=0)
```

In [113]:
```
df.shape
```

Out[113]:
```
(138265, 12)
```

In [213]:
```
df.head(5)
```

Out[213]:

	Year	Name	Age	Time	Pace	PiS/TiS	Division	PiD/TiD	Hometown	State	Blank1	Blank2
0	1999	Jane Omoro (W)	26	0:53:37	5:22	1/2358	W2529	1/559	Kenya	NaN	NaN	NaN
1	1999	Jane Ngotho (W)	29	0:53:38	5:22	2/2358	W2529	2/559	Kenya	NaN	NaN	NaN
2	1999	Lidiya Grigoryeva (W)	NR	0:53:40	5:22	3/2358	NR	NR	Russia	NaN	NaN	NaN
3	1999	Eunice Sagero (W)	20	0:53:55	5:24	4/2358	W2024	1/196	Kenya	NaN	NaN	NaN
4	1999	Alla Zhilyayeva (W)	29	0:54:08	5:25	5/2358	W2529	3/559	Russia	NaN	NaN	NaN

In [214]:
```
df1 = df.drop(['PiS/TiS', 'Division', 'PiD/TiD', 'Hometown', 'State','Blank1','Blank2'], axis=1)
```

In [215]:
```
df1.shape
```

Out[215]:
```
(138265, 5)
```

In [216]:
```
df1['Age'].value_counts()['NR']
```

Out[216]:
```
20
```

In [217]:
```
df1 = df1[df1.Age != "NR"]
```

In [218]:
```
df1.shape
```

Out[218]:
```
(138245, 5)
```

In [219]:
```
df1.head(5)
```

Out[219]:

	Year	Name	Age	Time	Pace
0	1999	Jane Omoro (W)	26	0:53:37	5:22
1	1999	Jane Ngotho (W)	29	0:53:38	5:22
3	1999	Eunice Sagero (W)	20	0:53:55	5:24
4	1999	Alla Zhilyayeva (W)	29	0:54:08	5:25
5	1999	Teresa Wanjiku (W)	24	0:54:10	5:25

In [220]:
```
print(df1.isnull().sum())
```

```
Year    0
Name    0
Age     0
Time    0
Pace    0
dtype: int64
```

In [221]:
```
df1.dtypes
```

Out[221]:
```
Year     int64
Name    object
Age     object
Time    object
Pace    object
dtype: object
```

In [222]:
```
df1['Age'] = df1['Age'].astype(str).astype(int)
df1['Pace'] = pd.to_datetime(df1['Pace'], format='%M:%S').dt.time
#df1['Time'] = pd.to_datetime(df1['Time'], format='%H:%M:%S').dt.time
```

In [223]:
```
df1.head(5)
```

Out[223]:

	Year	Name	Age	Time	Pace
0	1999	Jane Omoro (W)	26	0:53:37	00:05:22
1	1999	Jane Ngotho (W)	29	0:53:38	00:05:22
3	1999	Eunice Sagero (W)	20	0:53:55	00:05:24
4	1999	Alla Zhilyayeva (W)	29	0:54:08	00:05:25
5	1999	Teresa Wanjiku (W)	24	0:54:10	00:05:25

In [224]:

```
df1.to_csv ('CherryBlossomWomens10MTidy.csv', index = False, header=True)
```

Exploratory Data Analysis

In [238]:

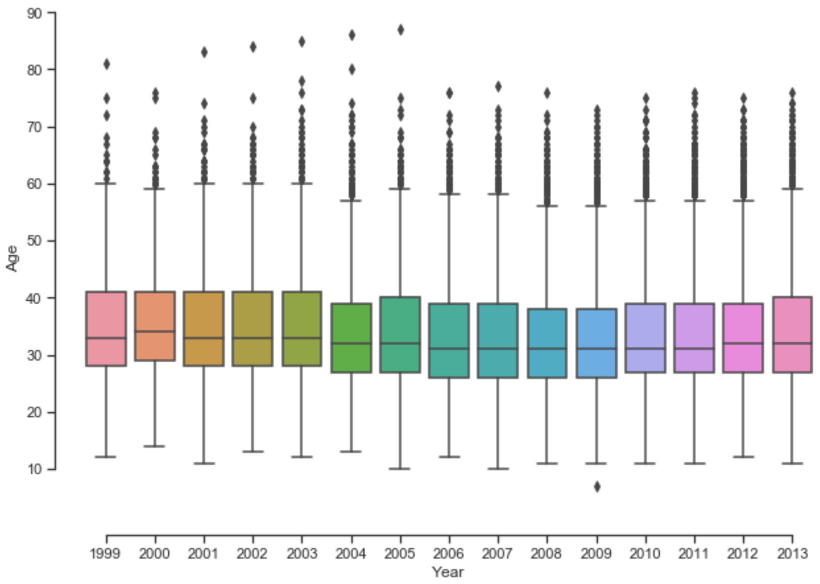
```
df1 = df1[df1.Year < 2014]
df1[["Age"]].describe()
```

Out[238]:

	Age
count	86169.000000
mean	33.919820
std	9.228929
min	7.000000
25%	27.000000
50%	32.000000
75%	39.000000
max	87.000000

In [277]:

```
plt.figure(figsize=(10,7))
sns.set(style="ticks", palette="pastel")
sns.boxplot(x="Year", y="Age", data=df1)
sns.despine(offset=20, trim=True)
```

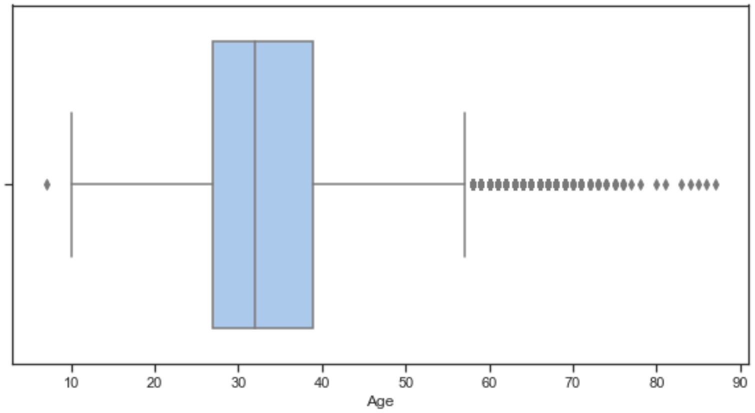


In [279]:

```
plt.figure(figsize=(10,5))
sns.boxplot(x=df1["Age"])
```

Out[279]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2362f3c8>
```

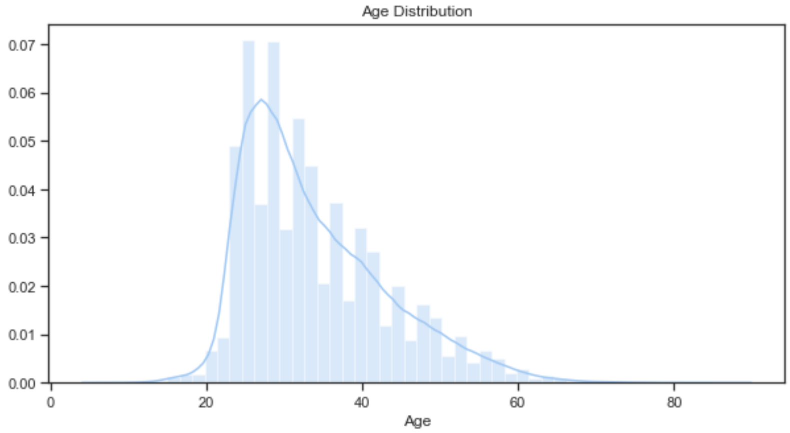


In [282]:

```
plt.figure(figsize=(10,5))
sns.distplot(df1['Age'])
plt.title("Age Distribution")
```

Out[282]:

```
Text(0.5,1,'Age Distribution')
```

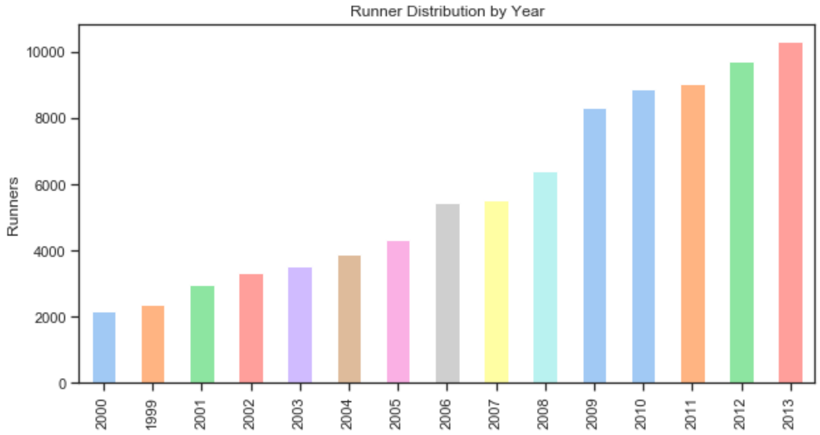


In [281]:

```
df1.Year.value_counts().nsmallest(15).plot(kind='bar', figsize=(10,5))
plt.title("Runner Distribution by Year")
plt.ylabel("Runners")
```

Out[281]:

```
Text(0,0.5,'Runners')
```

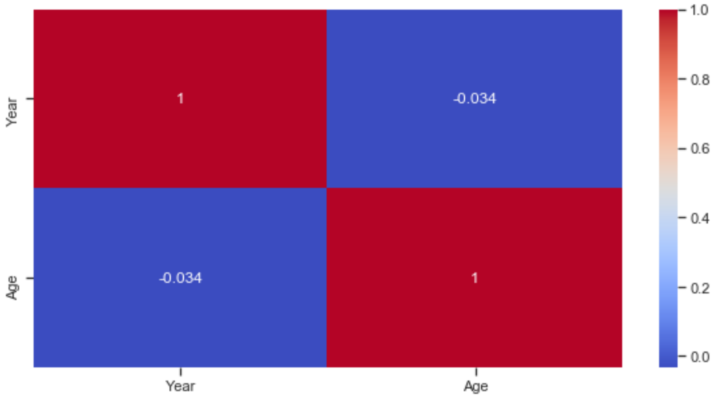


In [287]:

```
plt.figure(figsize=(10,5))
c= df1.corr()
sns.heatmap(c,cmap='coolwarm',annot=True)
c
```

Out[287]:

	Year	Age
Year	1.000000	-0.033776
Age	-0.033776	1.000000



In [230]:

```
df1[["Pace","Time"]].describe()
```

Out[230]:

	Pace	Time
count	138245	138245
unique	680	9183
top	00:09:46	1:37:03
freq	696	85

In [ ]:

```

```