190031920

A Nikhil Reddy

DS Skill 5

In [1]:
```
import math
import pandas as pd
import numpy as np
from matplotlib import pyplot
from scipy import stats
```

In [2]:
```
matches = pd.read_csv('matches.csv')
matches.head()
```

Out[2]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	

In [3]:
```
matches.describe()
```

Out[3]:

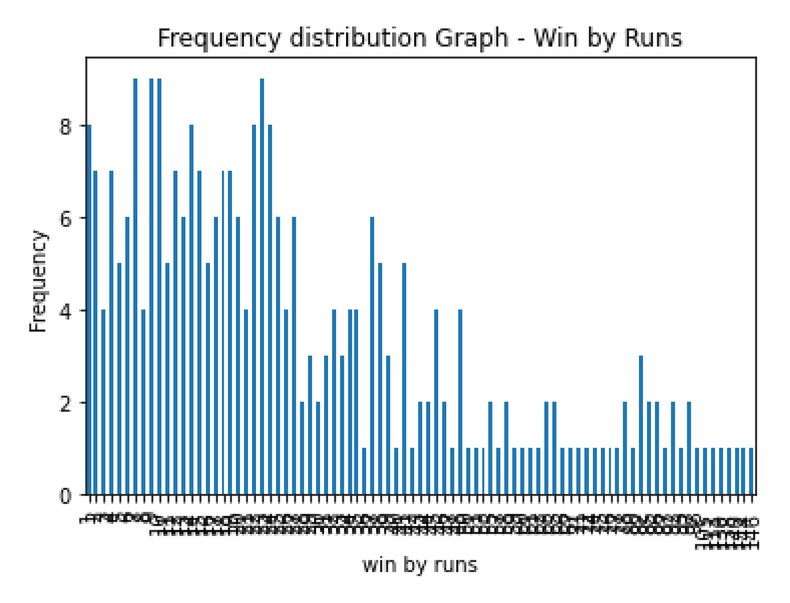
	id	season	dl_applied	win_by_runs	win_by_wickets	umpire3
count	636.000000	636.000000	636.000000	636.000000	636.000000	0.0
mean	318.500000	2012.490566	0.025157	13.682390	3.372642	NaN
std	183.741666	2.773026	0.156726	23.908877	3.420338	NaN
min	1.000000	2008.000000	0.000000	0.000000	0.000000	NaN
25%	159.750000	2010.000000	0.000000	0.000000	0.000000	NaN
50%	318.500000	2012.000000	0.000000	0.000000	4.000000	NaN
75%	477.250000	2015.000000	0.000000	20.000000	7.000000	NaN
max	636.000000	2017.000000	1.000000	146.000000	10.000000	NaN

In [4]:
```
#Returns win_by_runs data where win_by_runs is greater than zero
win_by_runs_data = matches[matches.win_by_runs > 0].win_by_runs

#Return a Series containing counts of unique rows in the DataFrame.
win_by_runs_freq = win_by_runs_data.value_counts(sort=False)

print(win_by_runs_freq)
```

```
1      8
2      7
3      4
4      7
5      5
      ..
130    1
138    1
140    1
144    1
146    1
Name: win_by_runs, Length: 85, dtype: int64
```

In [5]:
```
plt = win_by_runs_freq.plot.bar()
plt.set_title("Frequency distribution Graph - Win by Runs")
plt.set_xlabel("win by runs")
plt.set_ylabel("Frequency")
```

Out[5]: Text(0, 0.5, 'Frequency')

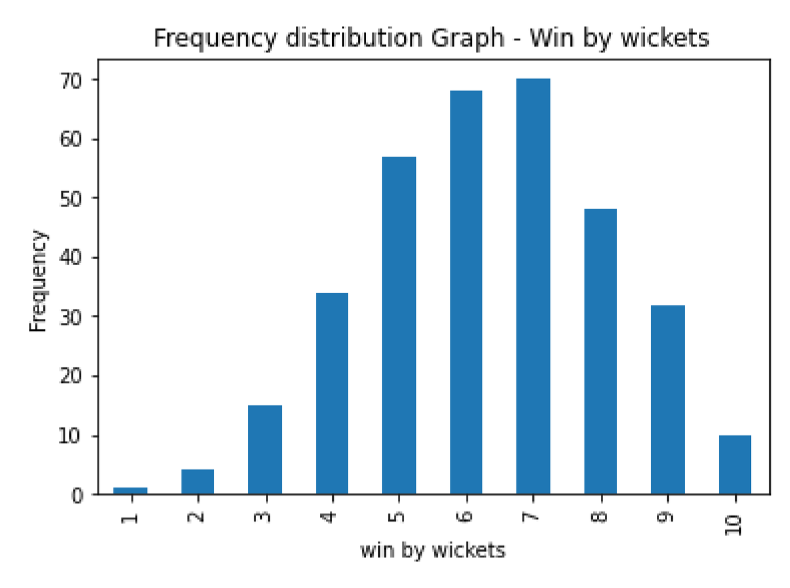


In [6]:
```
#Returns win_by_wickets_data data where win_by_wickets_data is greater than zero
win_by_wickets_data = matches[matches.win_by_wickets > 0].win_by_wickets

#Return a Series containing counts of unique rows in the DataFrame.
win_by_wickets_freq = win_by_wickets_data.value_counts(sort=False)

print(win_by_wickets_freq)
```

```
1      1
2      4
3     15
4     34
5     57
6     68
7     70
8     48
9     32
10    10
Name: win_by_wickets, dtype: int64
```

In [7]:
```
plt = win_by_wickets_freq.plot.bar()
plt.set_title("Frequency distribution Graph - Win by wickets")
plt.set_xlabel("win by wickets")
plt.set_ylabel("Frequency")
```

Out[7]: Text(0, 0.5, 'Frequency')

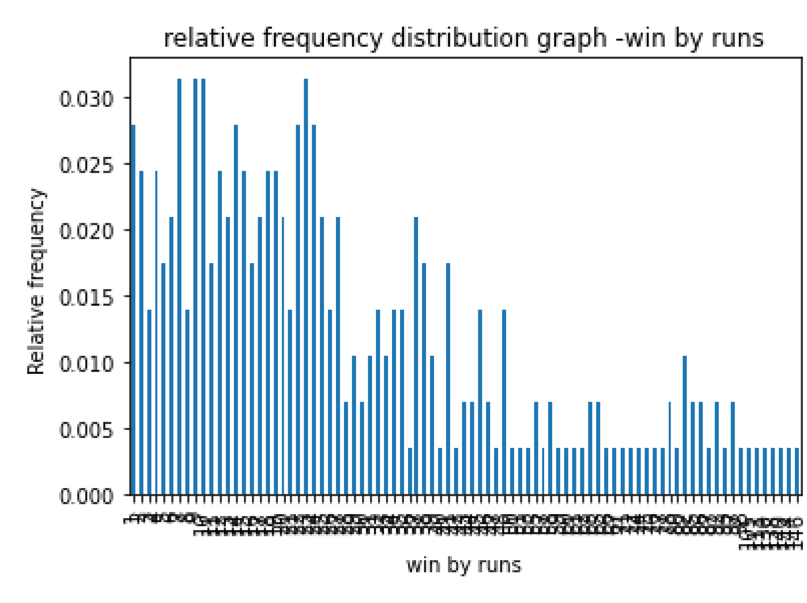


In [8]:
```
#normalize tag returns proportions rather than frequencies.
win_by_runs_rel_freq = win_by_runs_data.value_counts(sort = False , normalize=True)

print(win_by_runs_rel_freq)
```

```
1      0.027875
2      0.024390
3      0.013937
4      0.024390
5      0.017422
         ...
130    0.003484
138    0.003484
140    0.003484
144    0.003484
146    0.003484
Name: win_by_runs, Length: 85, dtype: float64
```

In [9]:
```
plt = win_by_runs_rel_freq.plot.bar()
plt.set_title("relative frequency distribution graph -win by runs")
plt.set_xlabel("win by runs")
plt.set_ylabel("Relative frequency ")
```

Out[9]: Text(0, 0.5, 'Relative frequency ')

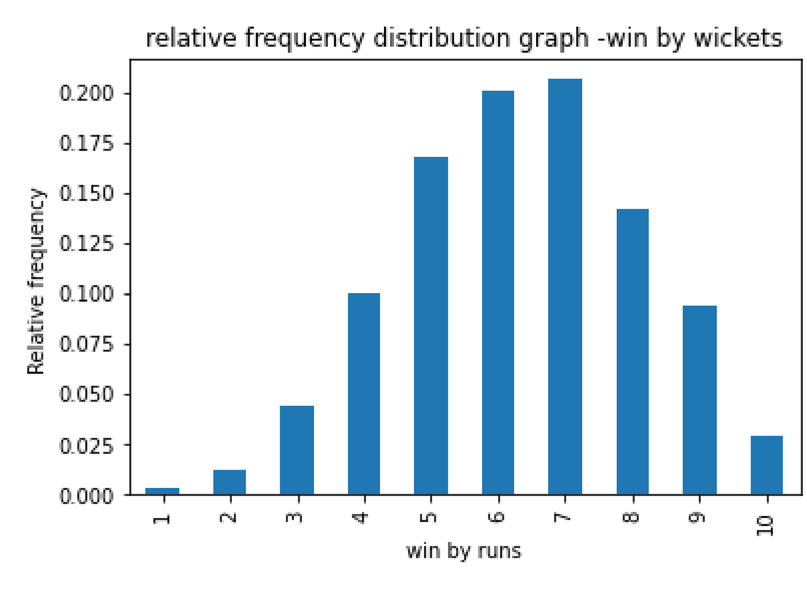


In [10]:
```
#normalize tag returns proportions rather than frequencies.
win_by_wickets_rel_freq = win_by_wickets_data.value_counts(sort = False , normalize=True)

print(win_by_wickets_rel_freq)
```

```
1     0.002950
2     0.011799
3     0.044248
4     0.100295
5     0.168142
6     0.200590
7     0.206490
8     0.141593
9     0.094395
10    0.029499
Name: win_by_wickets, dtype: float64
```

In [11]:
```
plt = win_by_wickets_rel_freq.plot.bar()
plt.set_title("relative frequency distribution graph -win by wickets")
plt.set_xlabel("win by runs")
plt.set_ylabel("Relative frequency ")
```

Out[11]: Text(0, 0.5, 'Relative frequency ')

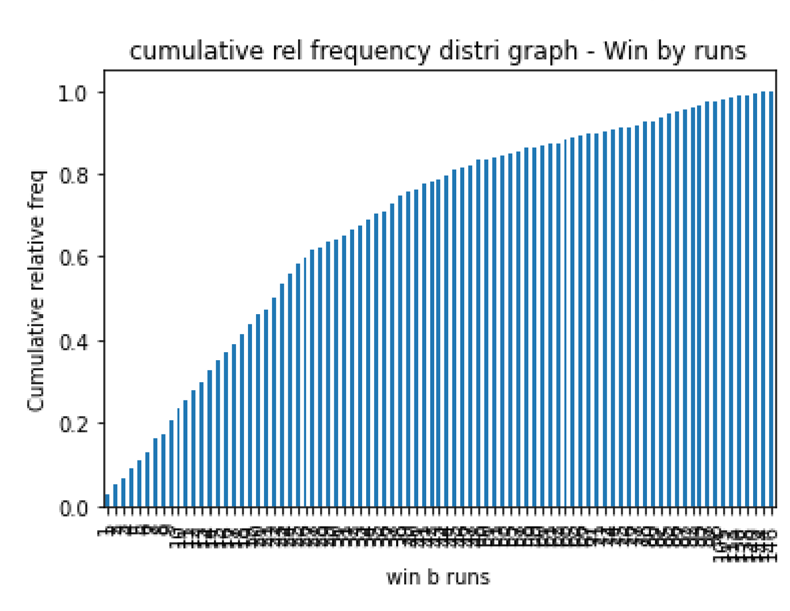


In [12]:
```
# cumsum = cumulative sum
win_by_runs_cumulative_freq = win_by_runs_data.value_counts(sort=False , normalize =True).cumsum()
print(win_by_runs_cumulative_freq)
```

```
1      0.027875
2      0.052265
3      0.066202
4      0.090592
5      0.108014
         ...
130    0.986063
138    0.989547
140    0.993031
144    0.996516
146    1.000000
Name: win_by_runs, Length: 85, dtype: float64
```

In [13]:
```
plt = win_by_runs_cumulative_freq.plot.bar()
plt.set_title("cumulative rel frequency distri graph - Win by runs")
plt.set_xlabel("win b runs")
plt.set_ylabel("Cumulative relative freq")
```

Out[13]: Text(0, 0.5, 'Cumulative relative freq')

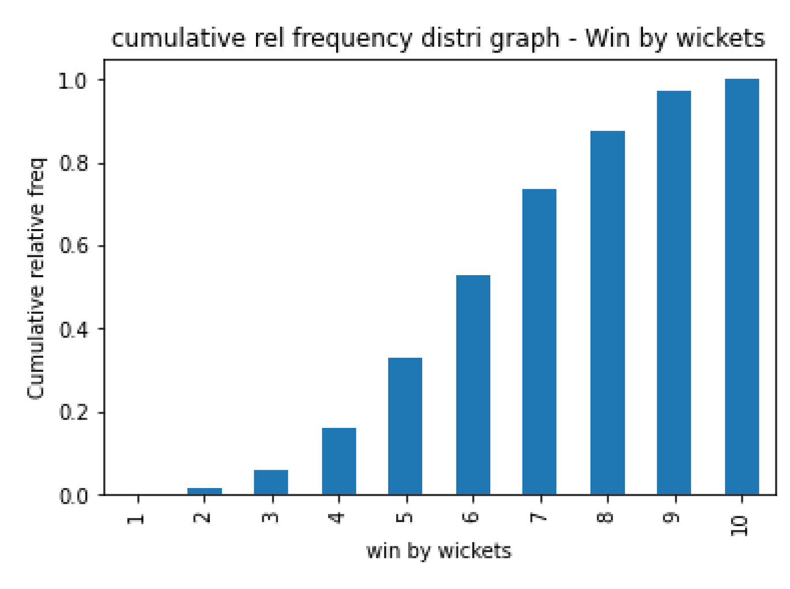


In [14]:
```
win_by_wickets_cumulative_freq = win_by_wickets_data.value_counts(sort=False , normalize =True).cum
sum()
print(win_by_wickets_cumulative_freq)
```

```
1     0.002950
2     0.014749
3     0.058997
4     0.159292
5     0.327434
6     0.528024
7     0.734513
8     0.876106
9     0.970501
10    1.000000
Name: win_by_wickets, dtype: float64
```

In [15]:
```
plt = win_by_wickets_cumulative_freq.plot.bar()
plt.set_title("cumulative rel frequency distri graph - Win by wickets")
plt.set_xlabel("win by wickets")
plt.set_ylabel("Cumulative relative freq")
```

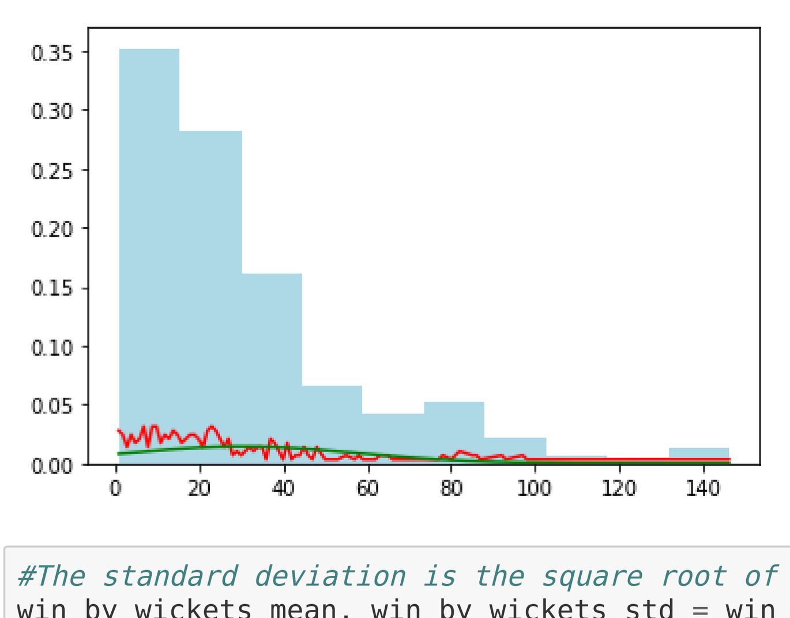
Out[15]: Text(0, 0.5, 'Cumulative relative freq')



Bell curve or Gaussian distribution

In [16]:
```
#The standard deviation is the square root of the average of the squared deviations from the mean
win_by_runs_mean,win_by_runs_std = win_by_runs_data.mean(),win_by_runs_data.std()

# Plot histogram (normalized) - LIGHT-BLUE
win_by_runs_data.hist(color='lightblue',weights=np.zeros_like(win_by_runs_data) + 1.0/win_by_runs_d
ata.count())

# Plot line graph - RED
win_by_runs_data.value_counts(sort=False, normalize=True).plot.line(color='red')

# Normal distribution for random points between 1 to 10 with mean, std.
random_data=np.arange(1,146,0.01)
pyplot.plot(random_data,stats.norm.pdf(random_data,win_by_runs_mean,win_by_runs_std),color='green')
```

Out[16]: [<matplotlib.lines.Line2D at 0x7f002d97f2d0>]



In [17]:
```
#The standard deviation is the square root of the average of the squared deviations from the mean
win_by_wickets_mean, win_by_wickets_std = win_by_wickets_data.mean(), win_by_wickets_data.std()

# Plot histogram (normalized) - LIGHT-BLUE
win_by_wickets_data.hist(color='lightblue', weights = np.zeros_like(win_by_wickets_data) + 1.0 / wi
n_by_wickets_data.count())

# Plot line graph - RED
win_by_wickets_data.value_counts(sort=False, normalize=True).plot.line(color='red')

# Normal distribution for random points between 1 to 10 with mean, std.
random_data = np.arange(1, 10, 0.001)
pyplot.plot(random_data, stats.norm.pdf(random_data, win_by_wickets_mean, win_by_wickets_std), colo
r='green')
```

Out[17]: [<matplotlib.lines.Line2D at 0x7f002d8a70d0>]