t20

June 4, 2021

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
df = pd.read_csv('data/t20.csv')
```

[2]:
```
df.head()
```

[2]:
```
   mid        date          venue bat_team  bowl_team         batsman bowler  \
0    1  2005-06-13  The Rose Bowl  England  Australia  ME Trescothick  B Lee
1    1  2005-06-13  The Rose Bowl  England  Australia  ME Trescothick  B Lee
2    1  2005-06-13  The Rose Bowl  England  Australia        GO Jones  B Lee
3    1  2005-06-13  The Rose Bowl  England  Australia        GO Jones  B Lee
4    1  2005-06-13  The Rose Bowl  England  Australia        GO Jones  B Lee

   runs  wickets  overs  runs_last_5  wickets_last_5  striker  non-striker  \
0     0        0    0.1            0               0        0            0
1     1        0    0.2            1               0        1            0
2     1        0    0.3            1               0        1            0
3     1        0    0.4            1               0        1            0
4     1        0    0.5            1               0        1            0

   total
0    179
1    179
2    179
3    179
4    179
```

[3]:
```
df=df.drop(['date'],axis=1)
```

[4]:
```
df.head()
```

[4]:
```
   mid          venue bat_team  bowl_team         batsman bowler  runs  \
0    1  The Rose Bowl  England  Australia  ME Trescothick  B Lee     0
1    1  The Rose Bowl  England  Australia  ME Trescothick  B Lee     1
```

```
2    1  The Rose Bowl  England  Australia        GO Jones  B Lee     1
3    1  The Rose Bowl  England  Australia        GO Jones  B Lee     1
4    1  The Rose Bowl  England  Australia        GO Jones  B Lee     1

   wickets  overs  runs_last_5  wickets_last_5  striker  non-striker  total
0        0    0.1            0               0        0            0    179
1        0    0.2            1               0        1            0    179
2        0    0.3            1               0        1            0    179
3        0    0.4            1               0        1            0    179
4        0    0.5            1               0        1            0    179
```

1 Data Visuaisation

[5]:
```
df.describe()
```

[5]:
```
                mid           runs        wickets          overs  \
count  180777.000000  180777.000000  180777.000000  180777.000000
mean      736.844726      74.553195       2.564541       9.767688
std       425.421789      48.530296       2.101655       5.768688
min         1.000000       0.000000       0.000000       0.000000
25%       368.000000      34.000000       1.000000       4.600000
50%       737.000000      70.000000       2.000000       9.600000
75%      1105.000000     110.000000       4.000000      14.600000
max      1474.000000     263.000000      10.000000      19.600000

         runs_last_5  wickets_last_5        striker    non-striker  \
count  180777.000000   180777.000000  180777.000000  180777.000000
mean       32.898798        1.193847      24.154953       8.163511
std        14.769098        1.077084      19.559632      10.043627
min         0.000000        0.000000       0.000000       0.000000
25%        24.000000        0.000000       9.000000       1.000000
50%        34.000000        1.000000      20.000000       4.000000
75%        43.000000        2.000000      34.000000      12.000000
max       113.000000        8.000000     175.000000     109.000000

               total
count  180777.000000
mean      158.308225
std        30.457209
min        39.000000
25%       139.000000
50%       159.000000
75%       179.000000
max       263.000000
```

[6]:
```
df.dtypes
```

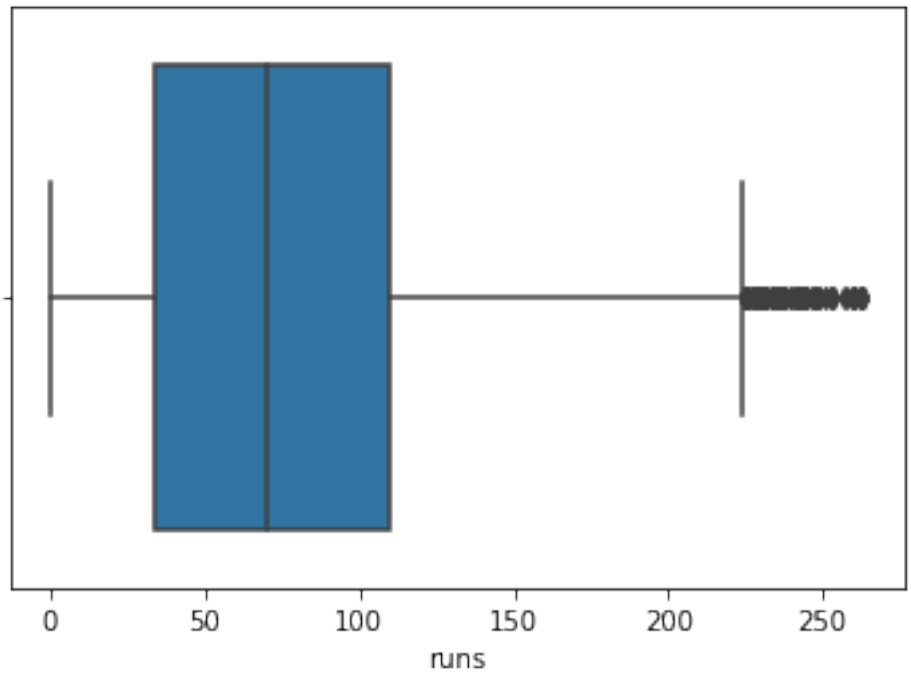
[6]:
```
mid                int64
venue             object
bat_team          object
bowl_team         object
batsman           object
bowler            object
runs               int64
wickets            int64
overs            float64
runs_last_5        int64
wickets_last_5     int64
striker            int64
non-striker        int64
total              int64
dtype: object
```

[7]:
```
df.shape
```

[7]:
```
(180777, 14)
```

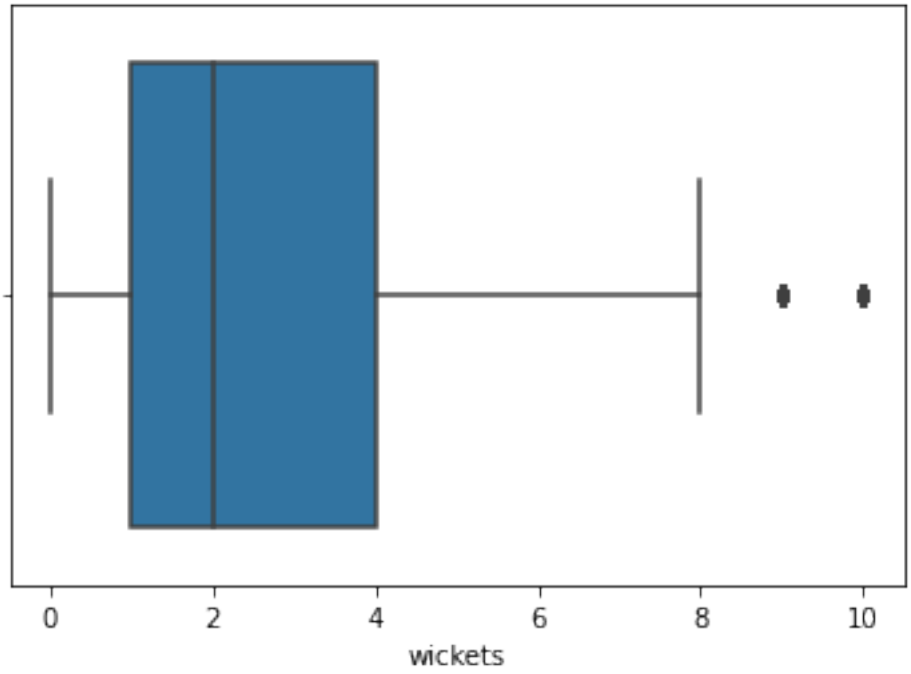
[8]:
```
sns.boxplot(x=df['runs'])
```

[8]:
```
<AxesSubplot:xlabel='runs'>
```



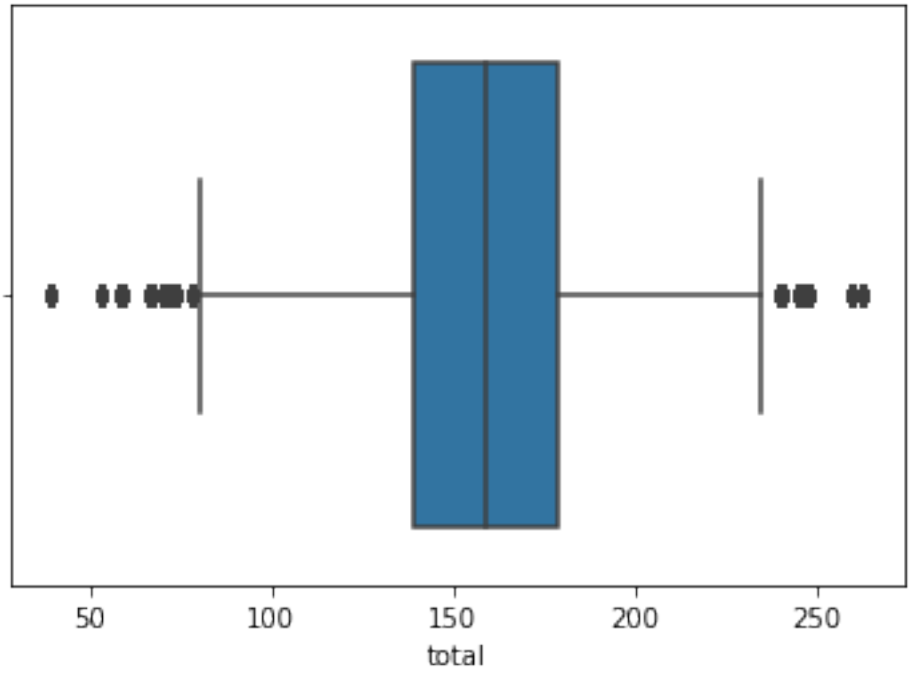
[9]:
```
sns.boxplot(x=df['wickets'])
```

[9]: <AxesSubplot:xlabel='wickets'>



[10]:
```
sns.boxplot(x=df['total'])
```

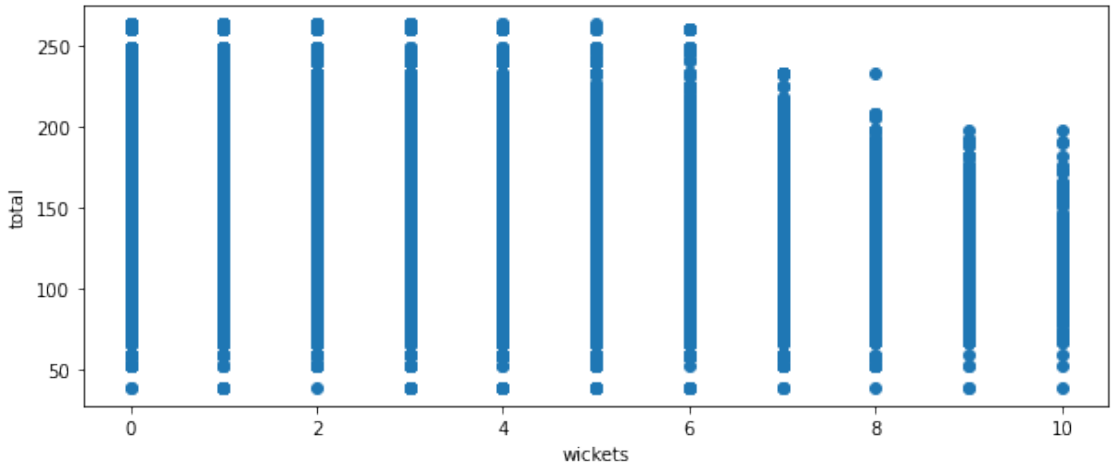
[10]: <AxesSubplot:xlabel='total'>

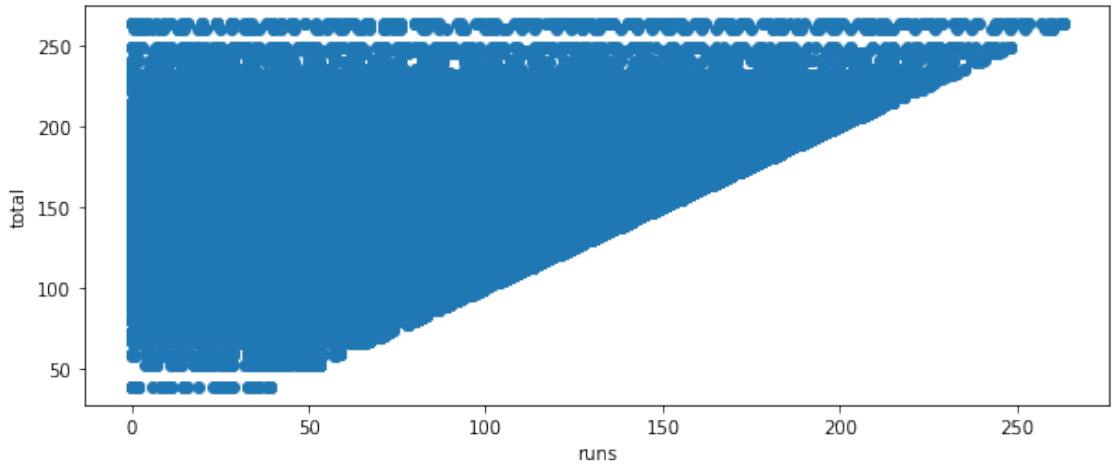


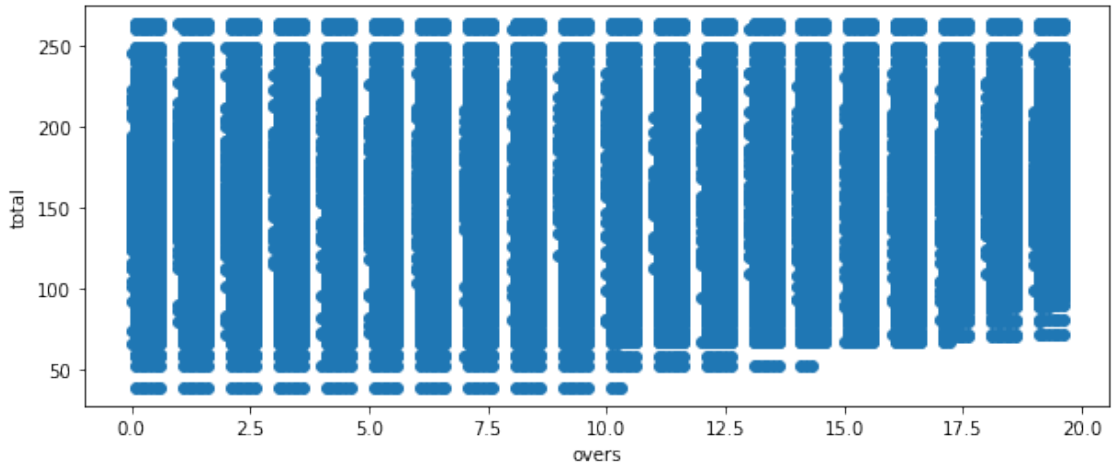
[11]:
```
fig, ax = plt.subplots(figsize=(10,4))
ax.scatter(df['wickets'] , df['total'])
ax.set_xlabel('wickets')
ax.set_ylabel('total')
plt.show()

fig, ax = plt.subplots(figsize=(10,4))
ax.scatter(df['runs'] , df['total'])
ax.set_xlabel('runs')
ax.set_ylabel('total')
plt.show()

fig, ax = plt.subplots(figsize=(10,4))
ax.scatter(df['overs'] , df['total'])
ax.set_xlabel('overs')
ax.set_ylabel('total')
plt.show()
```

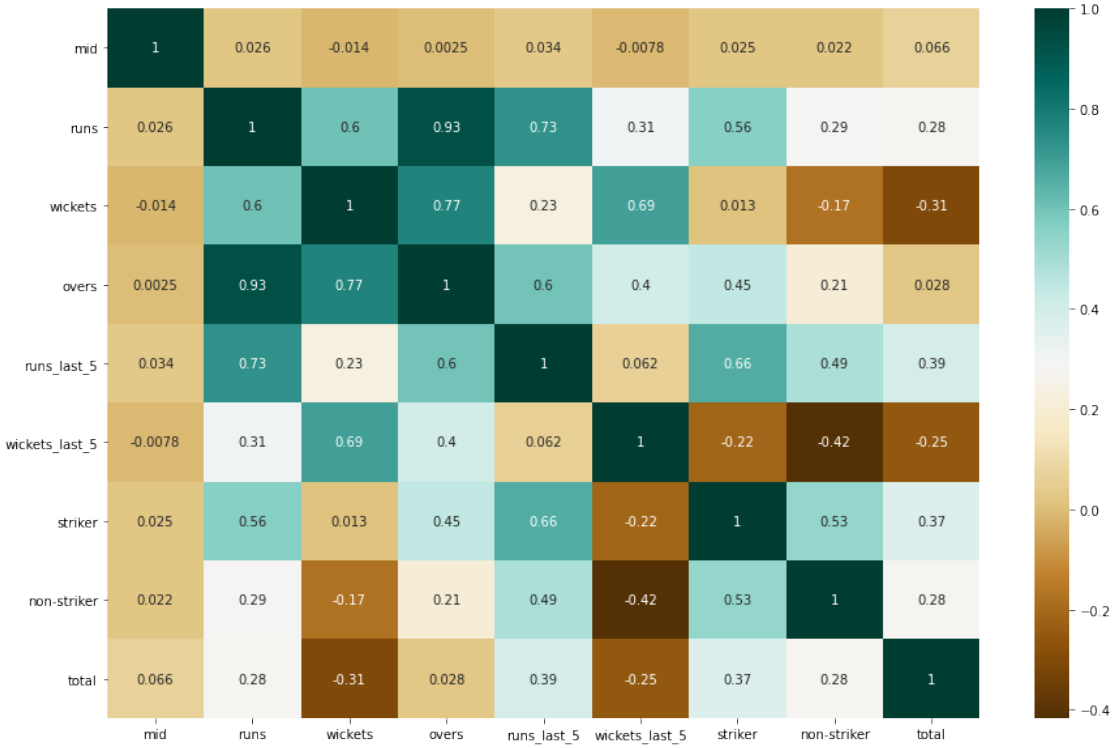




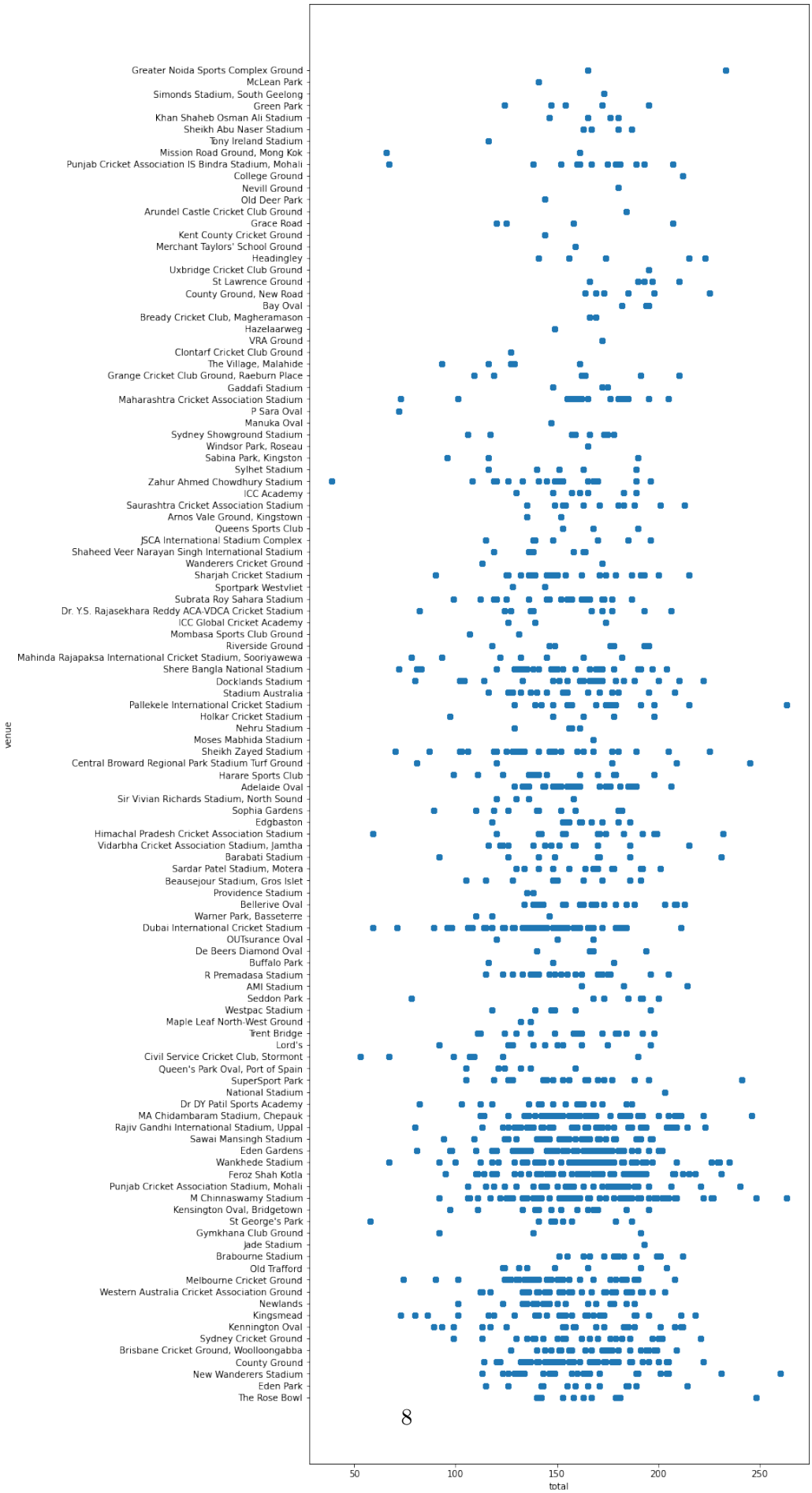


[12]:
```
plt.figure(figsize=(15,10))
c= df.corr()
sns.heatmap(c,cmap='BrBG',annot=True)
```

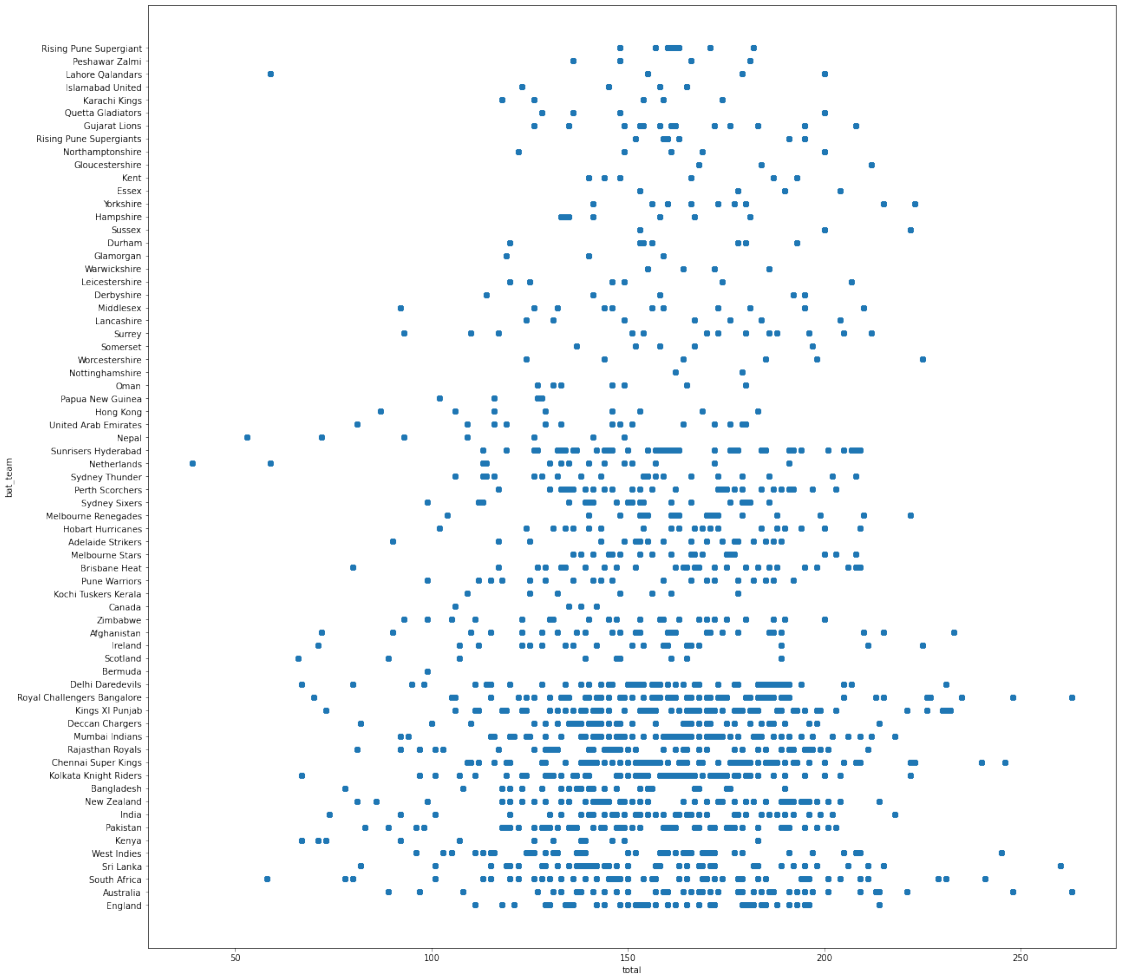
[12]: <AxesSubplot:>



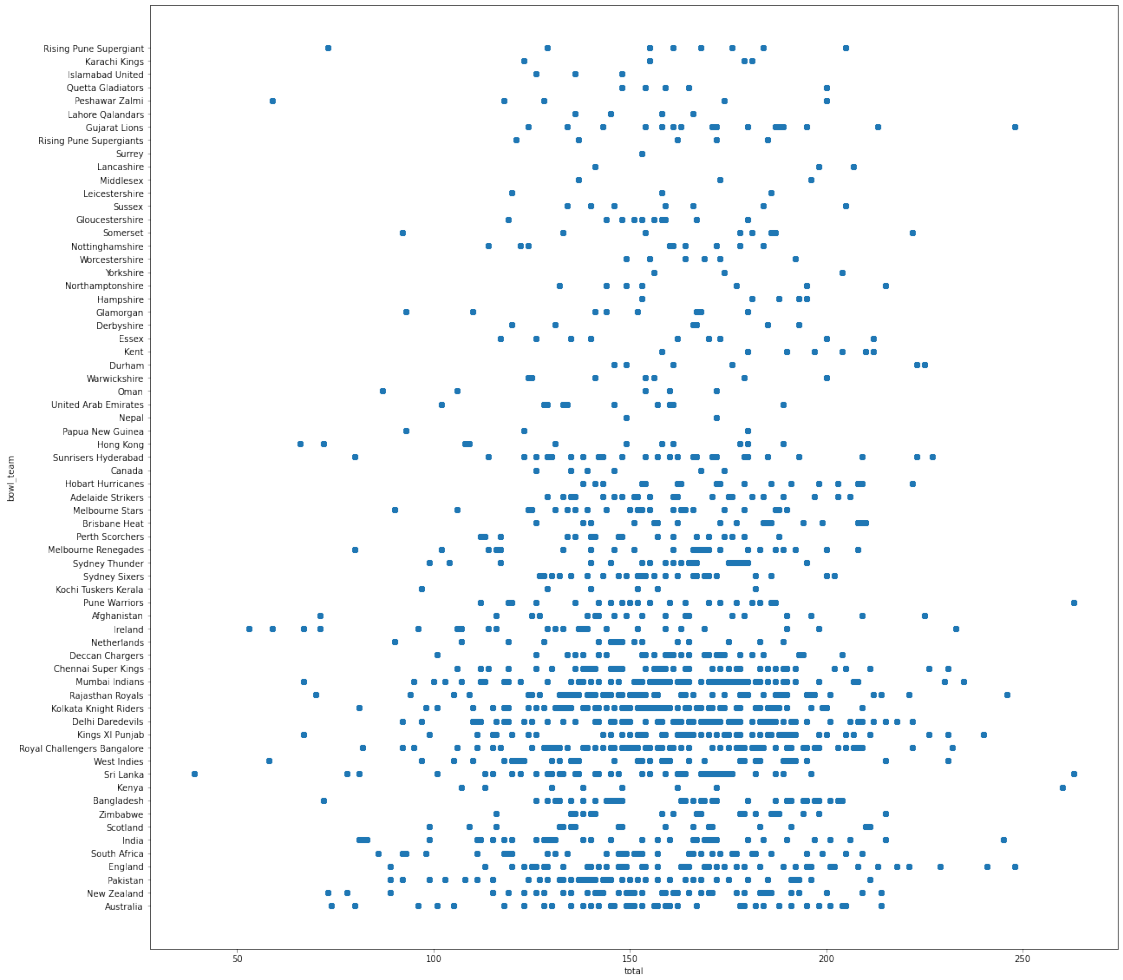
[13]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10,30))
ax.scatter(df['total'] , df['venue'])
ax.set_xlabel('total')
ax.set_ylabel('venue')
plt.show()
```


Greater Noida Sports Complex Ground
McLean Park
Simonds Stadium, South Geelong
Green Park
Khan Shaheb Osman Ali Stadium
Sheikh Abu Naser Stadium
Tony Ireland Stadium
Mission Road Ground, Mong Kok
Punjab Cricket Association IS Bindra Stadium, Mohali
College Ground
Nevill Ground
Old Deer Park
Arundel Castle Cricket Club Ground
Grace Road
Kent County Cricket Ground
Merchant Taylors' School Ground
Headingley
Uxbridge Cricket Club Ground
St Lawrence Ground
County Ground, New Road
Bay Oval
Bready Cricket Club, Magheramason
Hazelaarweg
VRA Ground
Clontarf Cricket Club Ground
The Village, Malahide
Grange Cricket Club Ground, Raeburn Place
Gaddafi Stadium
Maharashtra Cricket Association Stadium
P Sara Oval
Manuka Oval
Sydney Showground Stadium
Windsor Park, Roseau
Sabina Park, Kingston
Sylhet Stadium
Zahur Ahmed Chowdhury Stadium
ICC Academy
Saurashtra Cricket Association Stadium
Arnos Vale Ground, Kingstown
Queens Sports Club
JSCA International Stadium Complex
Shaheed Veer Narayan Singh International Stadium
Wanderers Cricket Ground
Sharjah Cricket Stadium
Sportpark Westvliet
Subrata Roy Sahara Stadium
Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium
ICC Global Cricket Academy
Mombasa Sports Club Ground
Riverside Ground
Mahinda Rajapaksa International Cricket Stadium, Sooriyawewa
Shere Bangla National Stadium
Docklands Stadium
Stadium Australia
Pallekele International Cricket Stadium
Holkar Cricket Stadium
Nehru Stadium
Moses Mabhida Stadium
Sheikh Zayed Stadium
Central Broward Regional Park Stadium Turf Ground
Harare Sports Club
Adelaide Oval
Sir Vivian Richards Stadium, North Sound
Sophia Gardens
Edgbaston
Himachal Pradesh Cricket Association Stadium
Vidarbha Cricket Association Stadium, Jamtha
Barabati Stadium
Sardar Patel Stadium, Motera
Beausejour Stadium, Gros Islet
Providence Stadium
Bellerive Oval
Warner Park, Basseterre
Dubai International Cricket Stadium
OUTsurance Oval
De Beers Diamond Oval
Buffalo Park
R Premadasa Stadium
AMI Stadium
Seddon Park
Westpac Stadium
Maple Leaf North-West Ground
Trent Bridge
Lord's
Civil Service Cricket Club, Stormont
Queen's Park Oval, Port of Spain
SuperSport Park
National Stadium
Dr DY Patil Sports Academy
MA Chidambaram Stadium, Chepauk
Rajiv Gandhi International Stadium, Uppal
Sawai Mansingh Stadium
Eden Gardens
Wankhede Stadium
Feroz Shah Kotla
Punjab Cricket Association Stadium, Mohali
M Chinnaswamy Stadium
Kensington Oval, Bridgetown
St George's Park
Gymkhana Club Ground
Jade Stadium
Brabourne Stadium
Old Trafford
Melbourne Cricket Ground
Western Australia Cricket Association Ground
Newlands
Kingsmead
Kennington Oval
Sydney Cricket Ground
Brisbane Cricket Ground, Woolloongabba
County Ground
New Wanderers Stadium
Eden Park
The Rose Bowl
venue
50
100
150
200
250
total

[14]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(20,20))
ax.scatter(df['total'] , df['bat_team'])
ax.set_xlabel('total')
ax.set_ylabel('bat_team')
plt.show()
```

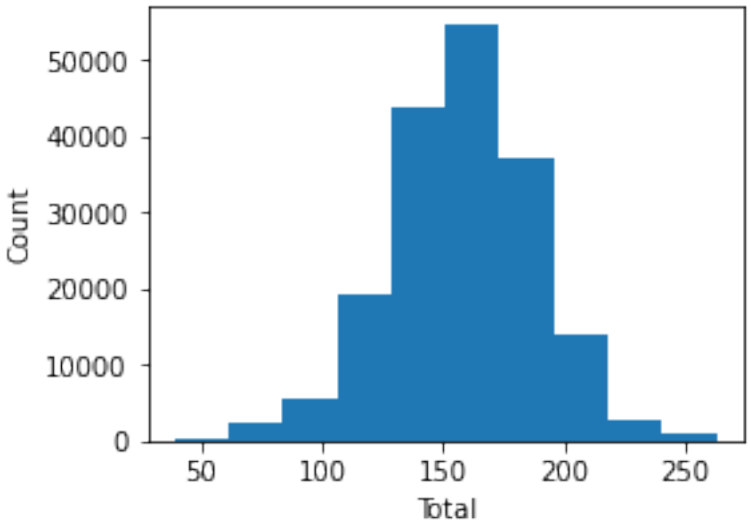


[15]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(20,20))
ax.scatter(df['total'] , df['bowl_team'])
ax.set_xlabel('total')
ax.set_ylabel('bowl_team')
plt.show()
```



[16]:
```
plt.figure(figsize=(4,3))
plt.hist(df.total)
plt.xlabel('Total')
plt.ylabel('Count')
plt.tight_layout
```

[16]:
```
<function matplotlib.pyplot.tight_layout(*, pad=1.08, h_pad=None, w_pad=None,
rect=None)>
```



2 We will now convert the textual data to numeric data so that those columns can be used for prediciton

[17]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['bat_team']=le.fit_transform(df['bat_team'])
df['venue']=le.fit_transform(df['venue'])
df['bowl_team']=le.fit_transform(df['bowl_team'])
df['batsman']=le.fit_transform(df['batsman'])
df['bowler']=le.fit_transform(df['bowler'])
```

[18]:
```
df.head(15)
```

[18]:

	mid	venue	bat_team	bowl_team	batsman	bowler	runs	wickets	overs	\
0	1	100	12	2	690	112	0	0	0.1	
1	1	100	12	2	690	112	1	0	0.2	
2	1	100	12	2	391	112	1	0	0.3	
3	1	100	12	2	391	112	1	0	0.4	
4	1	100	12	2	391	112	1	0	0.5	
5	1	100	12	2	391	112	2	0	0.5	
6	1	100	12	2	391	112	4	0	0.6	
7	1	100	12	2	690	283	4	0	1.1	
8	1	100	12	2	690	283	4	0	1.2	
9	1	100	12	2	690	283	5	0	1.2	
10	1	100	12	2	690	283	5	0	1.3	
11	1	100	12	2	690	283	5	0	1.4	
12	1	100	12	2	690	283	5	0	1.5	

```
13    1    100        12          2      690     283     6        0    1.6
14    1    100        12          2      690     112    10        0    2.1
```

	runs_last_5	wickets_last_5	striker	non-striker	total
0	0	0	0	0	179
1	1	0	1	0	179
2	1	0	1	0	179
3	1	0	1	0	179
4	1	0	1	0	179
5	2	0	1	0	179
6	4	0	2	1	179
7	4	0	2	1	179
8	4	0	2	1	179
9	5	0	2	1	179
10	5	0	2	1	179
11	5	0	2	1	179
12	5	0	2	1	179
13	6	0	2	2	179
14	10	0	6	2	179

3 Dividing Dependent and Independent Variables

[19]:
```
x = df.iloc[:,[1,2,3,4,5,6,7,8,11,12]].values
y = df.iloc[:, 13].values
```

4 Train Test Split

[20]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,␣
 ↪random_state = 42)
print(x_test[0], y_test[0])
```

```
[ 55.    5.   32.  808.  256.  156.    4.   16.6  23.    1. ] 188
```

5 Scaling the Dependent and Independent Variables

[21]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

6 Linar Regression

[22]:
```
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
lin.fit(x_train,y_train)
```

[22]:
```
LinearRegression()
```

7 Testing the Accuracy

[23]:
```
# Testing the dataset on trained model
from sklearn.metrics import r2_score,accuracy_score
y_pred = lin.predict(x_test)
score = lin.score(x_test,y_test)
print("R square value:" , score)
```

```
R square value: 0.5197147885180441
```

[24]:
```
def custom_accuracy(y_test,y_pred,thresold):
    right = 0

    l = len(y_pred)
    for i in range(0,l):
        if(abs(y_pred[i]-y_test[i]) <= thresold):
            right += 1
    return ((right/l)*100)
print("Custom accuracy:" , custom_accuracy(y_test,y_pred,20))
```

```
Custom accuracy: 72.05443079986725
```

8 Test Case Using Present Data

[25]:
```
for i in range(4):
    pred = lin.predict(sc.transform(np.array([x_test[i]])))
    print("Actual Score: ", y_test[i])
    print("Predicted Score: ", pred)
    print("Margin of Error: ", abs(y_test[i] - pred[0]))
    print("Margin of Error percent", ((abs(y_test[i] - pred[0]))*100)/y_test[i])
    print("")
```

```
Actual Score:  188
Predicted Score:  [152.19092726]
Margin of Error:  35.80907274258314
Margin of Error percent 19.047379118395288

Actual Score:  138
```

```
Predicted Score:  [176.81981598]
Margin of Error:  38.81981598343373
Margin of Error percent 28.130301437270816

Actual Score:  142
Predicted Score:  [166.69743839]
Margin of Error:  24.697438389606816
Margin of Error percent 17.392562246201983

Actual Score:  140
Predicted Score:  [146.47521555]
Margin of Error:  6.475215547206631
Margin of Error percent 4.62515396229045
```

9 Test Case using Random Data

[26]:
```
lin1 = lin.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , lin1)
```

```
Prediction score: [156.21197834]
```

[27]:
```
lin2 = lin.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , lin2)
```

```
Prediction score: [203.03467897]
```

[28]:
```
lin3 = lin.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , lin3)
```

```
Prediction score: [182.40036193]
```

10 Decision Tree Regressor

[29]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf = clf.fit(x_train,y_train)
y_pred1 = clf.predict(x_test)
```

11 Testing the accuracy

[30]:
```
y_pred1 = clf.predict(x_test)
score1 = clf.score(x_test,y_test)
print("R square value:" , score1)
```

```
R square value: 0.9811926098019692
```

[31]:
```
def custom_accuracy1(y_test,y_pred1,thresold):
    right = 0
    l = len(y_pred1)
    for i in range(0,l):
        if(abs(y_pred1[i]-y_test[i]) <= thresold):
            right += 1
    return ((right/l)*100)

print("Custom accuracy:" , custom_accuracy1(y_test,y_pred1,20),'%')
```

```
Custom accuracy: 98.99509532765424 %
```

12 Test Case using Present Data

[32]:
```
for i in range(4):
    predclf = clf.predict(sc.transform(np.array([x_test[i]])))
    print("Actual Score: ", y_test[i])
    print("Predicted Score: ", predclf)
    print("Margin of Error: ", abs(y_test[i] - predclf[0]))
    print("Margin of Error percent", ((abs(y_test[i] - predclf[0]))*100)/
 ↪y_test[i])
    print("")
```

```
Actual Score:  188
Predicted Score:  [185]
Margin of Error:  3
Margin of Error percent 1.5957446808510638

Actual Score:  138
Predicted Score:  [185]
Margin of Error:  47
Margin of Error percent 34.05797101449275

Actual Score:  142
Predicted Score:  [185]
Margin of Error:  43
Margin of Error percent 30.281690140845072

Actual Score:  140
Predicted Score:  [185]
Margin of Error:  45
Margin of Error percent 32.142857142857146
```

13 Test Case using Random Data

[33]:
```
clf1 = clf.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , clf1)
```

```
Prediction score: [179]
```

[34]:
```
clf2 = clf.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , clf2)
```

```
Prediction score: [171]
```

[35]:
```
clf3 = clf.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , clf3)
```

```
Prediction score: [130]
```

14 Random Forest Regressor

[36]:
```
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor(n_estimators=100,max_features=None)
reg.fit(x_train,y_train)
```

[36]:
```
RandomForestRegressor(max_features=None)
```

15 Testing the accuracy

[37]:
```
# Testing the dataset on trained model
y_pred2 = reg.predict(x_test)
score2 = reg.score(x_test,y_test)
print("R square value:" , score2)
```

```
R square value: 0.9242651012571862
```

[38]:
```
def custom_accuracy1(y_test,y_pred2,thresold):
    right = 0
    l = len(y_pred2)
    for i in range(0,l):
        if(abs(y_pred2[i]-y_test[i]) <= thresold):
            right += 1
    return ((right/l)*100)

print("Custom accuracy:" , custom_accuracy1(y_test,y_pred2,20),'%')
```

```
Custom accuracy: 96.45794151270421 %
```

16 Test Case using Present Data

[39]:
```
for i in range(4):
    predreg = reg.predict(sc.transform(np.array([x_test[i]])))
    print("Actual Score: ", y_test[i])
    print("Predicted Score: ", predreg)
    print("Margin of Error: ", abs(y_test[i] - predreg[0]))
    print("Margin of Error percent", ((abs(y_test[i] - predreg[0]))*100)/
 ↪y_test[i])
    print("")
```

```
Actual Score:  188
Predicted Score:  [143.96]
Margin of Error:  44.03999999999999
Margin of Error percent 23.42553191489361

Actual Score:  138
Predicted Score:  [181.84]
Margin of Error:  43.84
Margin of Error percent 31.768115942028984

Actual Score:  142
Predicted Score:  [180.77]
Margin of Error:  38.77000000000001
Margin of Error percent 27.302816901408455

Actual Score:  140
Predicted Score:  [156.67]
Margin of Error:  16.669999999999987
Margin of Error percent 11.907142857142848
```

17 Test Case

[40]:
```
reg1 = reg.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , reg1)
```

```
Prediction score: [129.87]
```

[41]:
```
reg2 = reg.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , reg2)
```

```
Prediction score: [201.4]
```

[42]:
```
reg3 = reg.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , reg3)
```

```
Prediction score: [138.03]
```