Clustering of Cities having similar environment

1. Data Understanding and Cleaning
2. Data Preparation
3. Modelling

In [1]:
```
#import all the necessary libraries

import pandas as pd
import numpy as np
import pandas as pd

# For Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# To Scale our data
from sklearn.preprocessing import scale

# To perform KMeans clustering
from sklearn.cluster import KMeans

# To perform Hierarchical clustering
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree
```

Data Understanding

In [2]:
```
# read the dataset
# The data set has been taken from data.gov.in
dat = pd.read_csv('datafile.csv')
dat.head()
```

Out[2]:

	Station_Name	Month	Mean_temperature_in_degree_C_Maximum	Mean_temperature_in_degree_C_Minimum	Mean_Rainfall_in_mm
0	Abu	January	19.3	8.0	5.3
1	Abu	February	21.0	10.0	4.4
2	Abu	March	25.3	14.5	6.5
3	Abu	April	29.4	18.7	2.6
4	Abu	May	31.5	21.0	16.4

In [3]:
```
dat.shape
```

Out[3]:
```
(1331, 5)
```

In [4]:
```
#basic data checks
dat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1331 entries, 0 to 1330
Data columns (total 5 columns):
Station_Name                          1331 non-null object
Month                                 1331 non-null object
Mean_temperature_in_degree_C_Maximum  1331 non-null float64
Mean_temperature_in_degree_C_Minimum  1331 non-null float64
Mean_Rainfall_in_mm                   1331 non-null float64
dtypes: float64(3), object(2)
memory usage: 52.1+ KB
```

In [5]:
```
#basic data cleaning checks
dat.isna().sum()
```

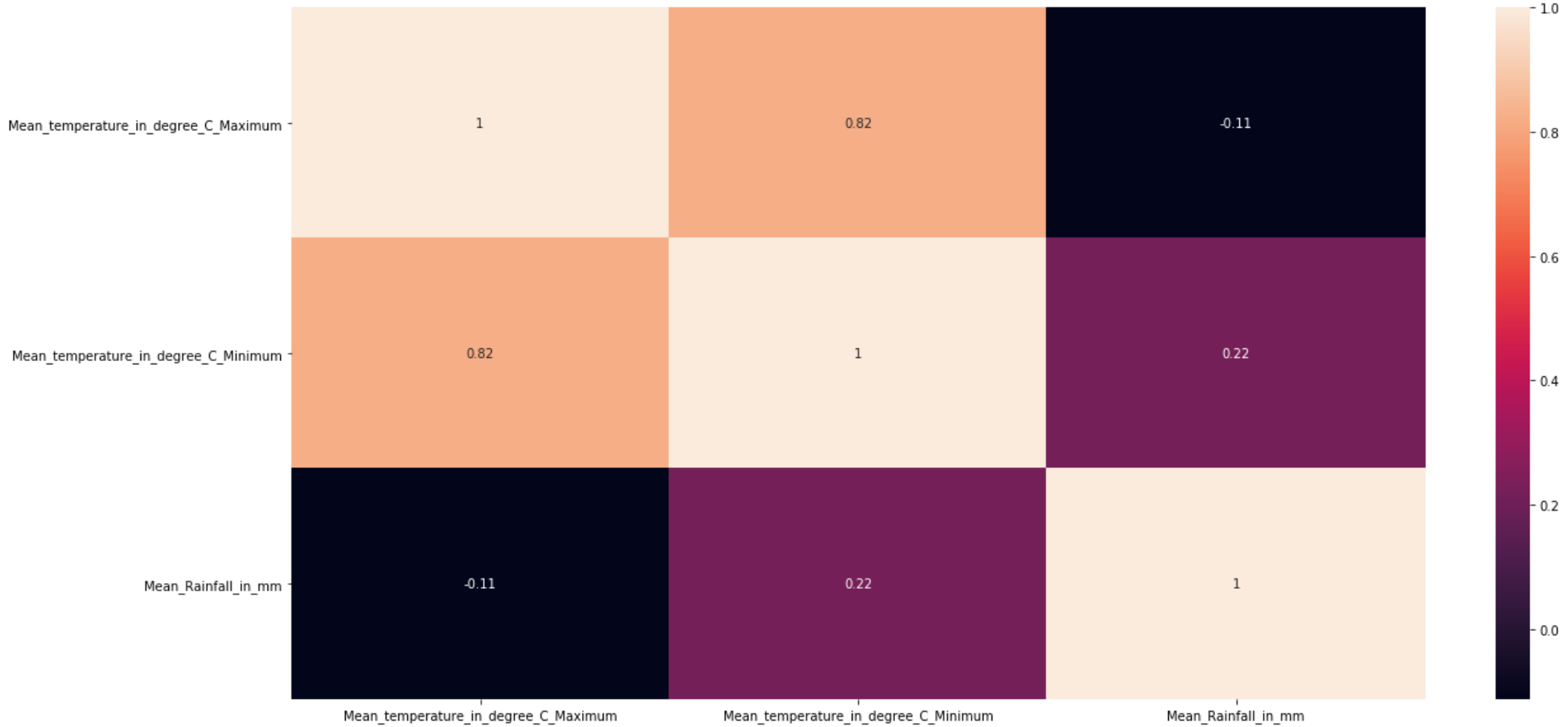
Out[5]:
```
Station_Name                          0
Month                                 0
Mean_temperature_in_degree_C_Maximum  0
Mean_temperature_in_degree_C_Minimum  0
Mean_Rainfall_in_mm                   0
dtype: int64
```

In [6]:
```
# Number of unique cities

len(dat['Station_Name'].unique().tolist())
```

Out[6]:
```
111
```

In [7]:
```
# data seems largely clean.
#ploting the correlation matrix and check if the data is indeed highly correlated
plt.figure(figsize = (20,10))
sns.heatmap(dat.corr(),annot = True)
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b11b007128>
```



In [8]:
```
#The Mean _temperature_in_degree_C _Maximum  is highly correlated with Mean_temperature_in_degree_C _Minimum
#  droping one of the feature "Mean_temperature_in_degree_C _Minimum"
datm=dat.drop(['Mean_temperature_in_degree_C_Minimum'],axis=1)
datm.head()
```

Out[8]:

	Station_Name	Month	Mean_temperature_in_degree_C_Maximum	Mean_Rainfall_in_mm
0	Abu	January	19.3	5.3
1	Abu	February	21.0	4.4
2	Abu	March	25.3	6.5
3	Abu	April	29.4	2.6
4	Abu	May	31.5	16.4

Dummy Variables

The variable `month` has 12 levels. We need to convert these levels into integer as well. For this, we will use something called `dummy variables` .

In [9]:
```
# Get the dummy variables for the feature 'fueltype' and store it in a new variable - 'status'

status = pd.get_dummies(datm['Month'])

# Check what the dataset 'status' looks like
status.head()
```

Out[9]:

	April	August	December	February	January	July	June	March	May	November	October	September
0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1	0	0	0

Now, you don't need 12 columns. You can drop any one column, as month can be identified with just the last 11 columns

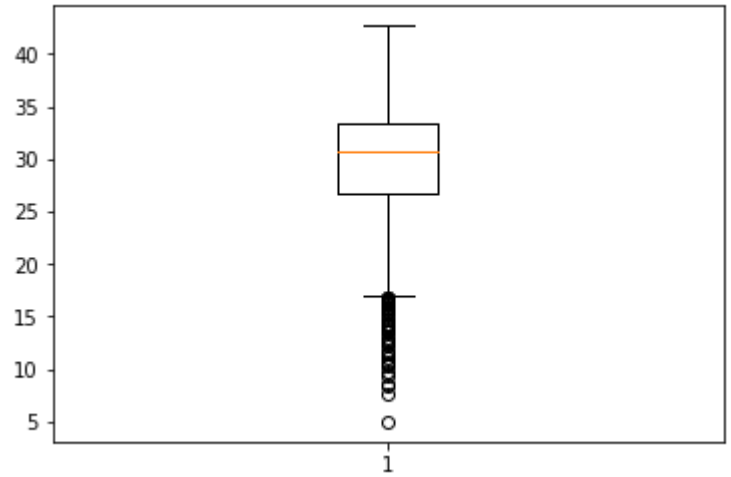
In [10]:
```
# droping the first column from status df using 'drop_first = True'
status = pd.get_dummies(datm['Month'], drop_first = True)

# Add the results to the original housing dataframe
datm = pd.concat([datm, status], axis = 1)

#  head of our dataframe.
datm.head()
```

Out[10]:

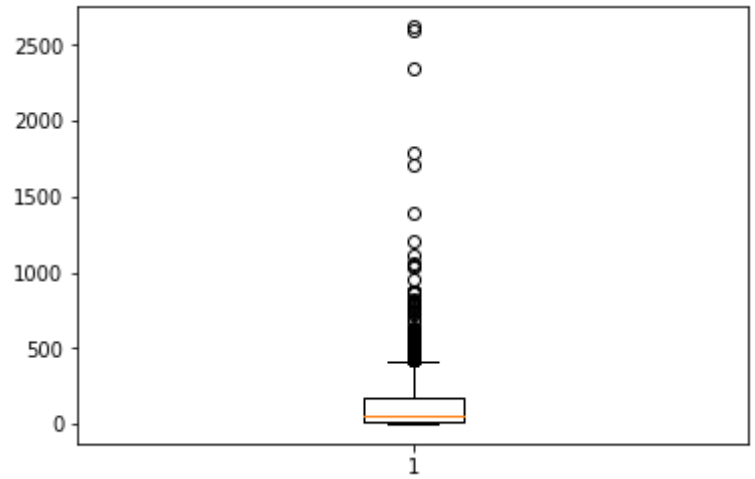
	Station_Name	Month	Mean_temperature_in_degree_C_Maximum	Mean_Rainfall_in_mm	August	December	February	January	July	June	March	May
0	Abu	January	19.3	5.3	0	0	0	1	0	0	0	0
1	Abu	February	21.0	4.4	0	0	1	0	0	0	0	0
2	Abu	March	25.3	6.5	0	0	0	0	0	0	1	0
3	Abu	April	29.4	2.6	0	0	0	0	0	0	0	0
4	Abu	May	31.5	16.4	0	0	0	0	0	0	0	1

Outlier treatment

In [11]:
```
plt.boxplot(datm.Mean_temperature_in_degree_C_Maximum)
Q1 = datm.Mean_temperature_in_degree_C_Maximum.quantile(0.25)
Q3 = datm.Mean_temperature_in_degree_C_Maximum.quantile(0.75)
IQR = Q3 - Q1
datm = datm[(datm.Mean_temperature_in_degree_C_Maximum >= Q1 - 1.5*IQR) & (datm.Mean_temperature_in_degree_C_Maximum <= Q3 + 1.
5*IQR)]
```



In [12]:
```
plt.boxplot(datm.Mean_Rainfall_in_mm)
Q1 = datm.Mean_Rainfall_in_mm.quantile(0.25)
Q3 = datm.Mean_Rainfall_in_mm.quantile(0.75)
IQR = Q3 - Q1
datm = datm[(datm.Mean_Rainfall_in_mm >= Q1 - 1.5*IQR) & (datm.Mean_Rainfall_in_mm <= Q3 + 1.5*IQR)]
```



Clustering

As we checked previously the dataset looks of similar magnitude. Hence no further standardisation is necessary. Let's proceed to calculating the Hopkins statistic to ensure that the data is good for clustering.

Hopkins Statistics:

The Hopkins statistic, is a statistic which gives a value which indicates the cluster tendency, in other words: how well the data can be clustered.

- If the value is between {0.01, ...,0.3}, the data is regularly spaced.
- If the value is around 0.5, it is random.
- If the value is between {0.7, ..., 0.99}, it has a high tendency to cluster.

In [13]:
```
#Calculating the Hopkins statistic
from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
import numpy as np
from math import isnan

def hopkins(X):
    d = X.shape[1]
    #d = len(vars) # columns
    n = len(X) # rows
    m = int(0.1 * n)
    nbrs = NearestNeighbors(n_neighbors=1).fit(X.values)

    rand_X = sample(range(0, n, 1), m)

    ujd = []
    wjd = []
    for j in range(0, m):
        u_dist, _ = nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),d).reshape(1, -1), 2, return_distance=True)
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1, -1), 2, return_distance=True)
        wjd.append(w_dist[0][1])

    H = sum(ujd) / (sum(ujd) + sum(wjd))
    if isnan(H):
        print(ujd, wjd)
        H = 0

    return H
```

In [14]:
```
#Let's check the Hopkins measure
hopkins(datm.drop(['Station_Name','Month'],axis=1))
```

Out[14]:
```
0.7493929571771981
```

In [15]:
```
#0.73 is a good Hopkins score. Hence the data is suitable for clustering. Preliminary check is now done.
# standardisation
dat3 = datm
from sklearn.preprocessing import StandardScaler
standard_scaler = StandardScaler()
dat3_1 = standard_scaler.fit_transform(dat3.drop(['Station_Name','Month'],axis=1))
```

```
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:645: DataConversionWarning: Data with input dtype uint8,
float64 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\hp\Anaconda3\lib\site-packages\sklearn\base.py:464: DataConversionWarning: Data with input dtype uint8, float64 were a
ll converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

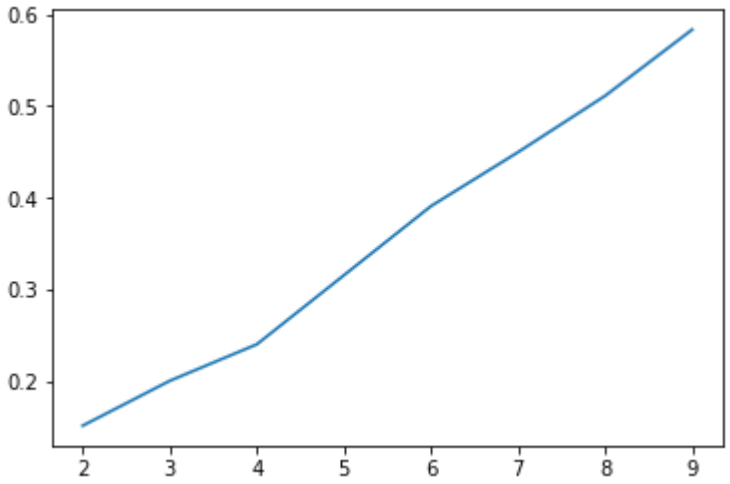
In [16]:
```
dat3.columns
```

Out[16]:
```
Index(['Station_Name', 'Month', 'Mean_temperature_in_degree_C_Maximum',
       'Mean_Rainfall_in_mm', 'August', 'December', 'February', 'January',
       'July', 'June', 'March', 'May', 'November', 'October', 'September'],
      dtype='object')
```

K-means Clustering

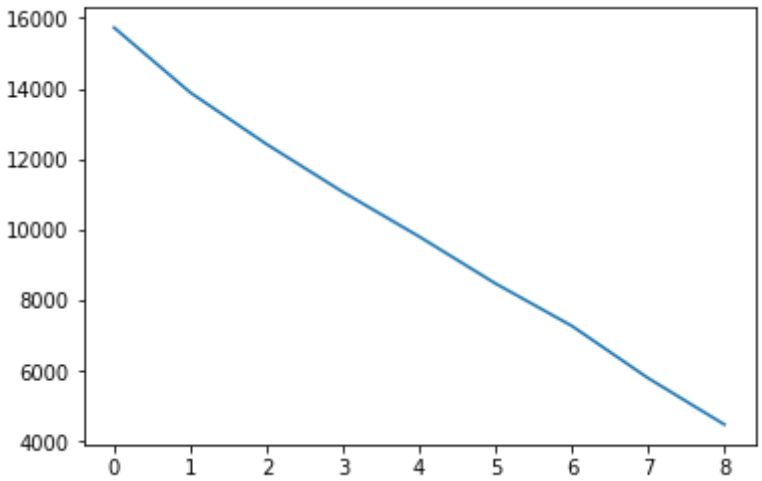
In [17]:
```
#Let's check the silhouette score first to identify the ideal number of clusters
from sklearn.metrics import silhouette_score
sse_ = []
for k in range(2, 10):
    kmeans = KMeans(n_clusters=k).fit(dat3_1)
    sse_.append([k, silhouette_score(dat3_1, kmeans.labels_)])
```

In [18]:
```
plt.plot(pd.DataFrame(sse_)[0], pd.DataFrame(sse_)[1]);
```



In [19]:
```
#The sihouette score reaches a peak at around 6 clusters indicating that it might be the ideal number of clusters.
#Let's use the elbow curve method to identify the ideal number of clusters.
ssd = []
for num_clusters in list(range(1,10)):
    model_clus = KMeans(n_clusters = num_clusters, max_iter=50)
    model_clus.fit(dat3_1)
    ssd.append(model_clus.inertia_)

plt.plot(ssd)
```

Out[19]:
```
[<matplotlib.lines.Line2D at 0x1b11b2d0278>]
```



In [20]:
```
#A distinct elbow is formed at around 3-7 clusters. Let's finally create the clusters and see for ourselves which ones fare bet
ter
#K-means with k=6 clusters
model_clus5 = KMeans(n_clusters = 6, max_iter=50)
model_clus5.fit(dat3_1)
```

Out[20]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=50,
    n_clusters=6, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [21]:
```
dat4=dat3
dat4.index = pd.RangeIndex(len(dat4.index))
dat_km = pd.concat([dat4, pd.Series(model_clus5.labels_)], axis=1)
dat_km.columns = ['Station_Name','Month', 'Mean_temperature_in_degree_C_Maximum',
       'Mean_Rainfall_in_mm', 'August', 'December', 'February', 'January',
       'July', 'June', 'March', 'May', 'November', 'October', 'September','ClusterID']
dat_km.head()
```

Out[21]:

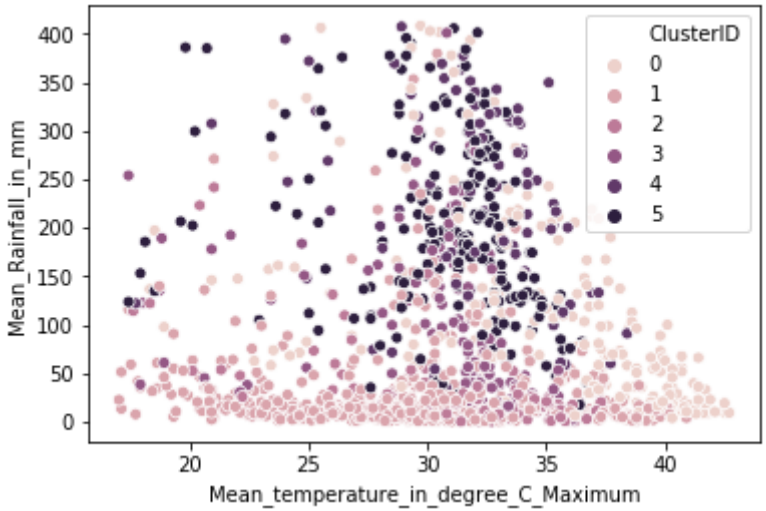
	Station_Name	Month	Mean_temperature_in_degree_C_Maximum	Mean_Rainfall_in_mm	August	December	February	January	July	June	March	May
0	Abu	January	19.3	5.3	0	0	0	1	0	0	0	0
1	Abu	February	21.0	4.4	0	0	1	0	0	0	0	0
2	Abu	March	25.3	6.5	0	0	0	0	0	0	1	0
3	Abu	April	29.4	2.6	0	0	0	0	0	0	0	0
4	Abu	May	31.5	16.4	0	0	0	0	0	0	0	1

In [22]:
```
dat_km['ClusterID'].value_counts()
```

Out[22]:
```
1    512
0    203
5    196
3    110
2    105
4     84
Name: ClusterID, dtype: int64
```

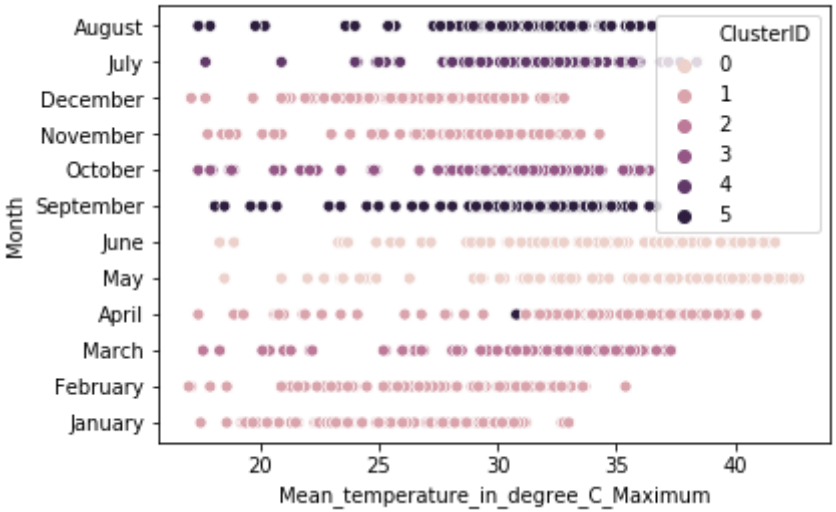
In [23]:
```
#Each cluster has a good number of city associated with it(at least 5% of the dataset under consideration)
#Let's do some further visualizations.
#We'll be visualising the clusters on the original principal components
sns.scatterplot(x='Mean_temperature_in_degree_C_Maximum',y='Mean_Rainfall_in_mm',hue='ClusterID',legend='full',data=dat_km)
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b11b2fec18>
```



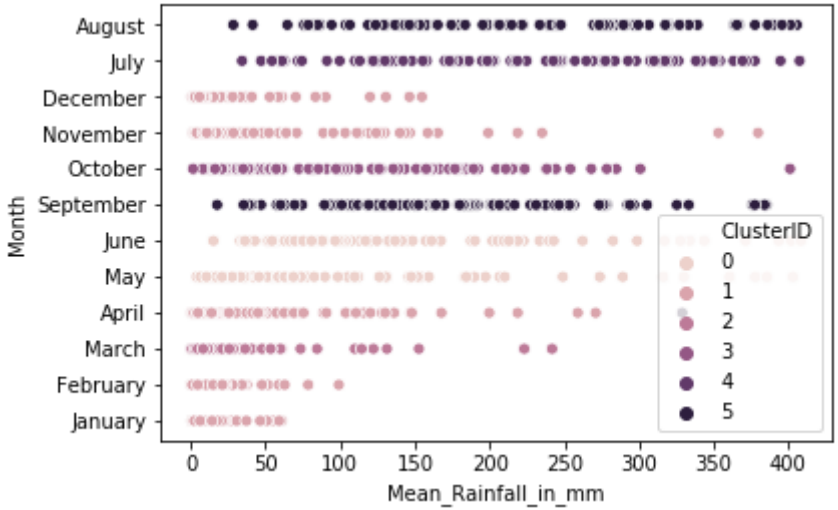
In [24]:
```
sns.scatterplot(x='Mean_temperature_in_degree_C_Maximum',y='Month',hue='ClusterID',legend='full',data=dat_km)
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b11b309ba8>
```



In [25]:
```
sns.scatterplot(x='Mean_Rainfall_in_mm',y='Month',hue='ClusterID',legend='full',data=dat_km)
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b11b142828>
```



In [26]:
```
#let's take a look at those city clusters and try to make sense if the clustering process worked well.
# city in cluster 0
cluster0=dat_km[dat_km['ClusterID']==0]
# As there are repeated values in the feature station_Name
cluster0[['Station_Name','Month']]
```

Out[26]:

	Station_Name	Month
4	Abu	May
5	Abu	June
14	Agartala (A)	May
15	Agartala (A)	June
26	Agra	May
27	Agra	June
38	Ahmedabad	May
39	Ahmedabad	June
50	Aijal/Aizwal	May
51	Aijal/Aizwal	June
62	Ajmer	May
63	Ajmer	June
74	Akola (A)	May
75	Akola (A)	June
86	Allahabad	May
87	Allahabad	June
98	Ambikapur	May
99	Ambikapur	June
109	Amini Divi	May
110	Amini Divi	June
121	Amritsar (Rajasansi)	May
122	Amritsar (Rajasansi)	June
133	Anantpur	May
134	Anantpur	June
145	Androth	May
146	Androth	June
156	Aurangabad	May
157	Aurangabad	June

	Station_Name	Month
168	Balasore	May
169	Balasore	June
...	...	...
1036	Ranchi (A)	May
1037	Ranchi (A)	June
1048	Sambalpur	May
1049	Sambalpur	June
1057	Shillong	May
1064	Shimla	May
1065	Shimla	June
1081	Solapur	May
1082	Solapur	June
1093	Sri Niketan	May
1094	Sri Niketan	June
1102	Srinagar	May
1103	Srinagar	June
1112	Surat	May
1113	Surat	June
1123	Thiruvananthapuram	May
1124	Thiruvananthapuram	June
1135	Tirupathy	May
1136	Tirupathy	June
1147	Tura	May
1156	Udaipur (Dabok)	May
1157	Udaipur (Dabok)	June
1168	Uthagamandalam	May
1169	Uthagamandalam	June
1179	Varanasi (Babatpur)	May
1180	Varanasi (Babatpur)	June

	Station_Name	Month
1191	Vijayawada	May
1192	Vijayawada	June
1203	Vishakhapatnam	May
1204	Vishakhapatnam	June

203 rows × 2 columns

In [27]:
```
# city in cluster 1
cluster1=dat_km[dat_km['ClusterID']==1]
cluster1[['Station_Name','Month']]
```

Out[27]:

	Station_Name	Month
0	Abu	January
1	Abu	February
3	Abu	April
8	Abu	November
9	Abu	December
10	Agartala (A)	January
11	Agartala (A)	February
13	Agartala (A)	April
20	Agartala (A)	November
21	Agartala (A)	December
22	Agra	January
23	Agra	February
25	Agra	April
32	Agra	November
33	Agra	December
34	Ahmedabad	January
35	Ahmedabad	February
37	Ahmedabad	April
44	Ahmedabad	November
45	Ahmedabad	December
46	Aijal/Aizwal	January
47	Aijal/Aizwal	February
49	Aijal/Aizwal	April
56	Aijal/Aizwal	November
57	Aijal/Aizwal	December
58	Ajmer	January
59	Ajmer	February
61	Ajmer	April

	Station_Name	Month
68	Ajmer	November
69	Ajmer	December
...	...	...
1142	Tirupathy	December
1143	Tura	January
1144	Tura	February
1146	Tura	April
1150	Tura	November
1151	Tura	December
1152	Udaipur (Dabok)	January
1153	Udaipur (Dabok)	February
1155	Udaipur (Dabok)	April
1162	Udaipur (Dabok)	November
1163	Udaipur (Dabok)	December
1164	Uthagamandalam	January
1165	Uthagamandalam	February
1167	Uthagamandalam	April
1173	Uthagamandalam	November
1174	Uthagamandalam	December
1175	Varanasi (Babatpur)	January
1176	Varanasi (Babatpur)	February
1178	Varanasi (Babatpur)	April
1185	Varanasi (Babatpur)	November
1186	Varanasi (Babatpur)	December
1187	Vijayawada	January
1188	Vijayawada	February
1190	Vijayawada	April
1197	Vijayawada	November
1198	Vijayawada	December

	Station_Name	Month
1199	Vishakhapatnam	January
1200	Vishakhapatnam	February
1202	Vishakhapatnam	April
1209	Vishakhapatnam	November

512 rows × 2 columns

In [28]:
```
# city in cluster 2
cluster2=dat_km[dat_km['ClusterID']==2]
cluster2[['Station_Name','Month']]
```

Out[28]:

	Station_Name	Month
2	Abu	March
12	Agartala (A)	March
24	Agra	March
36	Ahmedabad	March
48	Aijal/Aizwal	March
60	Ajmer	March
72	Akola (A)	March
84	Allahabad	March
96	Ambikapur	March
107	Amini Divi	March
119	Amritsar (Rajasansi)	March
131	Anantpur	March
143	Androth	March
154	Aurangabad	March
166	Balasore	March
178	Bangalore	March
190	Bareilly	March
202	Baroda (A)	March
214	Belgaum Samra	March
226	Bhagalpur	March
238	Bhatinda	March
250	Bhopal (Bairagarh)	March
262	Bhubaneshwar (A)	March
274	Bhuj (Rudramata)	March
286	Bikaner	March
298	Cannanore	March
307	Chandigarh	March
319	Chennai (Minambakkam)	March

	Station_Name	Month
330	Cherrapunji	March
335	Coimbatore (Pilamedu)	March
...	...	...
862	Mysore	March
874	Nagpur (Sonegaon)	March
891	Nasik	March
903	New Delhi (Palam)	March
915	New Delhi (SFD)	March
927	Palakkad (Palghat)	March
938	Panjim	March
947	Parbhani	March
959	Pasighat	March
967	Patna (A)	March
979	Pondicherry (A)	March
990	Port Blair	March
998	Pune	March
1010	Raipur	March
1022	Rajkot (A)	March
1034	Ranchi (A)	March
1046	Sambalpur	March
1055	Shillong	March
1071	Silchar	March
1079	Solapur	March
1091	Sri Niketan	March
1110	Surat	March
1121	Thiruvananthapuram	March
1133	Tirupathy	March
1145	Tura	March
1154	Udaipur (Dabok)	March

	Station_Name	Month
1166	Uthagamandalam	March
1177	Varanasi (Babatpur)	March
1189	Vijayawada	March
1201	Vishakhapatnam	March

105 rows × 2 columns

In [29]:
```
# city in cluster 3
cluster3=dat_km[dat_km['ClusterID']==3]
cluster3[['Station_Name','Month']]
```

Out[29]:

	Station_Name	Month
7	Abu	October
19	Agartala (A)	October
31	Agra	October
43	Ahmedabad	October
55	Aijal/Aizwal	October
67	Ajmer	October
79	Akola (A)	October
91	Allahabad	October
102	Ambikapur	October
114	Amini Divi	October
126	Amritsar (Rajasansi)	October
138	Anantpur	October
149	Androth	October
161	Aurangabad	October
173	Balasore	October
185	Bangalore	October
197	Bareilly	October
209	Baroda (A)	October
221	Belgaum Samra	October
233	Bhagalpur	October
245	Bhatinda	October
257	Bhopal (Bairagarh)	October
269	Bhubaneshwar (A)	October
281	Bhuj (Rudramata)	October
293	Bikaner	October
302	Cannanore	October
314	Chandigarh	October
326	Chennai (Minambakkam)	October

	Station_Name	Month
342	Coimbatore (Pilamedu)	October
350	Cooch Behar	October
...	...	...
898	Nasik	October
910	New Delhi (Palam)	October
922	New Delhi (SFD)	October
933	Palakkad (Palghat)	October
942	Panjim	October
954	Parbhani	October
962	Pasighat	October
974	Patna (A)	October
986	Pondicherry (A)	October
993	Port Blair	October
1005	Pune	October
1017	Raipur	October
1029	Rajkot (A)	October
1041	Ranchi (A)	October
1051	Sambalpur	October
1061	Shillong	October
1068	Shimla	October
1074	Silchar	October
1086	Solapur	October
1098	Sri Niketan	October
1107	Srinagar	October
1116	Surat	October
1128	Thiruvananthapuram	October
1140	Tirupathy	October
1149	Tura	October
1161	Udaipur (Dabok)	October

	Station_Name	Month
1172	Uthagamandalam	October
1184	Varanasi (Babatpur)	October
1196	Vijayawada	October
1208	Vishakhapatnam	October

110 rows × 2 columns

In [30]:
```
# city in cluster 4
cluster4=dat_km[dat_km['ClusterID']==4]
cluster4[['Station_Name','Month']]
```

Out[30]:

	Station_Name	Month
16	Agartala (A)	July
28	Agra	July
40	Ahmedabad	July
52	Aijal/Aizwal	July
64	Ajmer	July
76	Akola (A)	July
88	Allahabad	July
111	Amini Divi	July
123	Amritsar (Rajasansi)	July
135	Anantpur	July
158	Aurangabad	July
170	Balasore	July
182	Bangalore	July
194	Bareilly	July
206	Baroda (A)	July
218	Belgaum Samra	July
230	Bhagalpur	July
242	Bhatinda	July
254	Bhopal (Bairagarh)	July
266	Bhubaneshwar (A)	July
278	Bhuj (Rudramata)	July
290	Bikaner	July
311	Chandigarh	July
323	Chennai (Minambakkam)	July
339	Coimbatore (Pilamedu)	July
388	Gadag	July
404	Gaya	July
416	Gopalpur	July

	Station_Name	Month
428	Gorakhpur	July
440	Gulbarga	July
...	...	...
753	Lucknow (Amausi)	July
765	Ludhiana	July
777	Madurai (A)	July
797	Malda	July
806	Manali	July
826	Masulipatnam	July
838	Minicoy	July
847	Mukteswar (Kumaun)	July
866	Mysore	July
878	Nagpur (Sonegaon)	July
895	Nasik	July
907	New Delhi (Palam)	July
919	New Delhi (SFD)	July
951	Parbhani	July
971	Patna (A)	July
983	Pondicherry (A)	July
1002	Pune	July
1014	Raipur	July
1026	Rajkot (A)	July
1038	Ranchi (A)	July
1058	Shillong	July
1083	Solapur	July
1095	Sri Niketan	July
1104	Srinagar	July
1125	Thiruvananthapuram	July
1137	Tirupathy	July

	Station_Name	Month
1158	Udaipur (Dabok)	July
1181	Varanasi (Babatpur)	July
1193	Vijayawada	July
1205	Vishakhapatnam	July

84 rows × 2 columns

In [31]:
```
# city in cluster 5
cluster5=dat_km[dat_km['ClusterID']==5]
cluster5[['Station_Name','Month']]
```

Out[31]:

	Station_Name	Month
6	Abu	September
17	Agartala (A)	August
18	Agartala (A)	September
29	Agra	August
30	Agra	September
41	Ahmedabad	August
42	Ahmedabad	September
53	Aijal/Aizwal	August
54	Aijal/Aizwal	September
65	Ajmer	August
66	Ajmer	September
77	Akola (A)	August
78	Akola (A)	September
89	Allahabad	August
90	Allahabad	September
100	Ambikapur	August
101	Ambikapur	September
112	Amini Divi	August
113	Amini Divi	September
124	Amritsar (Rajasansi)	August
125	Amritsar (Rajasansi)	September
136	Anantpur	August
137	Anantpur	September
147	Androth	August
148	Androth	September
159	Aurangabad	August
160	Aurangabad	September
171	Balasore	August

	Station_Name	Month
172	Balasore	September
183	Bangalore	August
...	...	...
1050	Sambalpur	September
1059	Shillong	August
1060	Shillong	September
1066	Shimla	August
1067	Shimla	September
1072	Silchar	April
1073	Silchar	September
1084	Solapur	August
1085	Solapur	September
1096	Sri Niketan	August
1097	Sri Niketan	September
1105	Srinagar	August
1106	Srinagar	September
1114	Surat	August
1115	Surat	September
1126	Thiruvananthapuram	August
1127	Thiruvananthapuram	September
1138	Tirupathy	August
1139	Tirupathy	September
1148	Tura	September
1159	Udaipur (Dabok)	August
1160	Udaipur (Dabok)	September
1170	Uthagamandalam	August
1171	Uthagamandalam	September
1182	Varanasi (Babatpur)	August
1183	Varanasi (Babatpur)	September

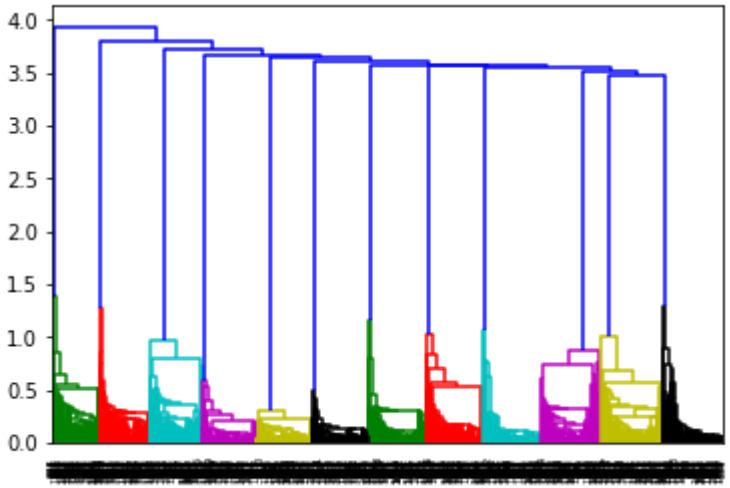
	Station_Name	Month
1194	Vijayawada	August
1195	Vijayawada	September
1206	Vishakhapatnam	August
1207	Vishakhapatnam	September

196 rows × 2 columns

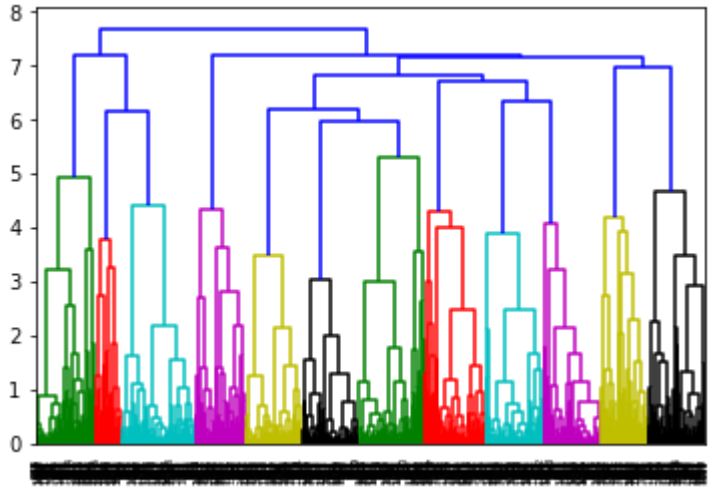
Hierarchical Clustering

Let's check if hierarchical clustering does a good job.

In [32]:
```
#Let's try hierarchical clustering to see if it works well
#First we'll try the single linkage procedure.
mergings = linkage(dat3_1, method = "single", metric='euclidean')
dendrogram(mergings)
plt.show()
```



In [33]:
```
#We do get good results here.
#Let's try complete linkage method
mergings = linkage(dat3_1, method = "complete", metric='euclidean')
dendrogram(mergings)
plt.show()
```



In [34]:
```
#Okay now we are seeing some good clusters here. Let's see if they make sense if we eliminate the barriers
clusterCut = pd.Series(cut_tree(mergings, n_clusters = 6).reshape(-1,))
dat3_hc = pd.concat([dat3, clusterCut], axis=1)
dat3_hc.columns = ['Station_Name','Month', 'Mean_temperature_in_degree_C_Maximum',
       'Mean_Rainfall_in_mm', 'August', 'December', 'February', 'January',
       'July', 'June', 'March', 'May', 'November', 'October', 'September','ClusterID']
```

In [35]:
```
dat3_hc.head()
```

Out[35]:

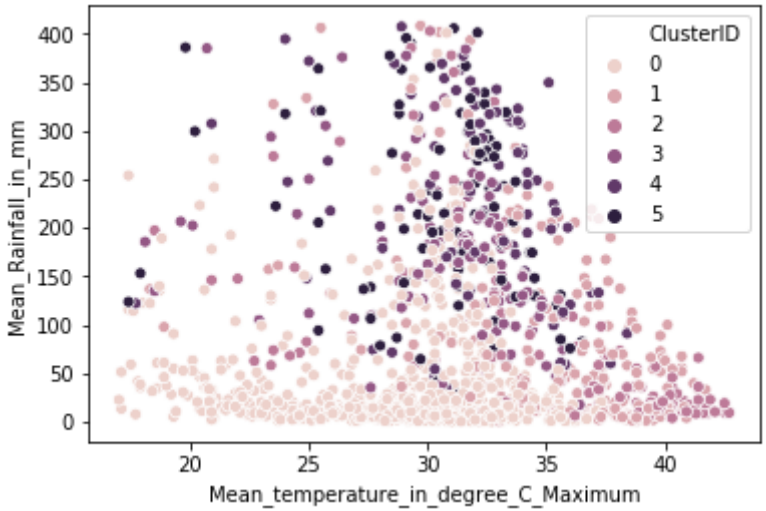
	Station_Name	Month	Mean_temperature_in_degree_C_Maximum	Mean_Rainfall_in_mm	August	December	February	January	July	June	March	May
0	Abu	January	19.3	5.3	0	0	0	1	0	0	0	0
1	Abu	February	21.0	4.4	0	0	1	0	0	0	0	0
2	Abu	March	25.3	6.5	0	0	0	0	0	0	1	0
3	Abu	April	29.4	2.6	0	0	0	0	0	0	0	0
4	Abu	May	31.5	16.4	0	0	0	0	0	0	0	1

In [36]:
```
dat3_hc['ClusterID'].value_counts()
```

Out[36]:
```
0    641
1    182
2    108
3    104
5     91
4     84
Name: ClusterID, dtype: int64
```

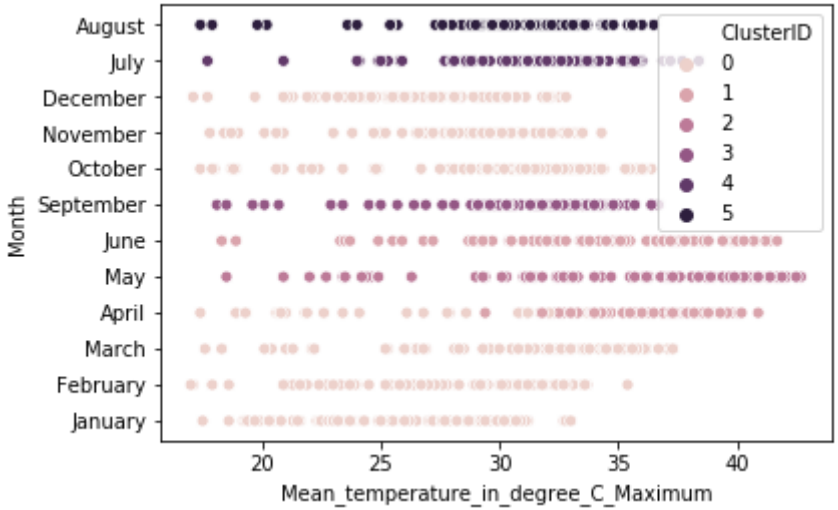
In [37]:
```
#Each cluster has a good number of city associated with it(at least 5% of the dataset under consideration)
#Let's do some further visualizations.
#We'll be visualising the clusters on the original principal components
sns.scatterplot(x='Mean_temperature_in_degree_C_Maximum',y='Mean_Rainfall_in_mm',hue='ClusterID',legend='full',data=dat3_hc)
```

Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b11c83d0f0>
```



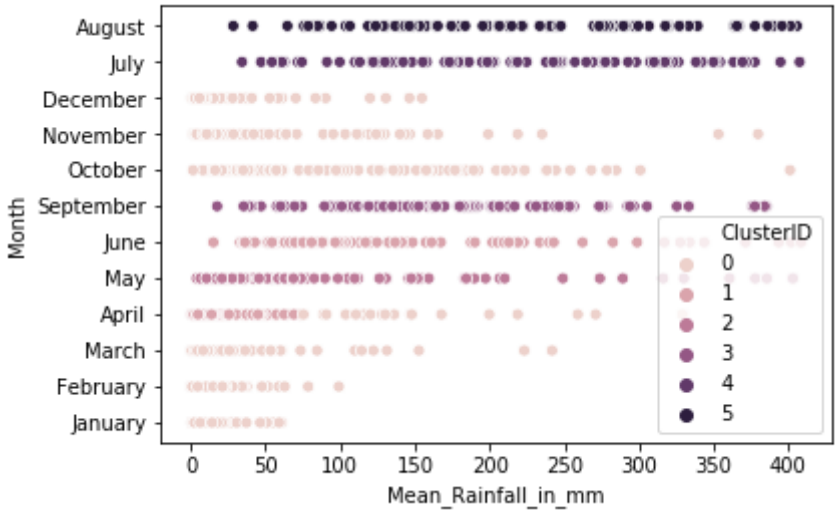
In [38]:
```
sns.scatterplot(x='Mean_temperature_in_degree_C_Maximum',y='Month',hue='ClusterID',legend='full',data=dat3_hc)
```

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x1b11c368898>



In [39]:
```
sns.scatterplot(x='Mean_Rainfall_in_mm',y='Month',hue='ClusterID',legend='full',data=dat3_hc)
```

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x1b11b435a58>



In [40]:
```
#let's take a look at those city clusters and try to make sense if the clustering process worked well.
# city in cluster 0
cluster0=dat3_hc[dat3_hc['ClusterID']==0]
# As there are repeated values in the feature station_Name
cluster0[['Station_Name','Month']]
```

Out[40]:

	Station_Name	Month
0	Abu	January
1	Abu	February
2	Abu	March
7	Abu	October
8	Abu	November
9	Abu	December
10	Agartala (A)	January
11	Agartala (A)	February
12	Agartala (A)	March
13	Agartala (A)	April
19	Agartala (A)	October
20	Agartala (A)	November
21	Agartala (A)	December
22	Agra	January
23	Agra	February
24	Agra	March
31	Agra	October
32	Agra	November
33	Agra	December
34	Ahmedabad	January
35	Ahmedabad	February
36	Ahmedabad	March
43	Ahmedabad	October
44	Ahmedabad	November
45	Ahmedabad	December
46	Aijal/Aizwal	January
47	Aijal/Aizwal	February
48	Aijal/Aizwal	March

	Station_Name	Month
49	Aijal/Aizwal	April
55	Aijal/Aizwal	October
...	...	...
1152	Udaipur (Dabok)	January
1153	Udaipur (Dabok)	February
1154	Udaipur (Dabok)	March
1161	Udaipur (Dabok)	October
1162	Udaipur (Dabok)	November
1163	Udaipur (Dabok)	December
1164	Uthagamandalam	January
1165	Uthagamandalam	February
1166	Uthagamandalam	March
1167	Uthagamandalam	April
1172	Uthagamandalam	October
1173	Uthagamandalam	November
1174	Uthagamandalam	December
1175	Varanasi (Babatpur)	January
1176	Varanasi (Babatpur)	February
1177	Varanasi (Babatpur)	March
1184	Varanasi (Babatpur)	October
1185	Varanasi (Babatpur)	November
1186	Varanasi (Babatpur)	December
1187	Vijayawada	January
1188	Vijayawada	February
1189	Vijayawada	March
1196	Vijayawada	October
1197	Vijayawada	November
1198	Vijayawada	December
1199	Vishakhapatnam	January

	Station_Name	Month
1200	Vishakhapatnam	February
1201	Vishakhapatnam	March
1208	Vishakhapatnam	October
1209	Vishakhapatnam	November

641 rows × 2 columns

In [41]:
```
# city in cluster 1
cluster1=dat3_hc[dat3_hc['ClusterID']==1]
# As there are repeated values in the feature station_Name
cluster1[['Station_Name','Month']]
```

Out[41]:

	Station_Name	Month
3	Abu	April
5	Abu	June
15	Agartala (A)	June
25	Agra	April
27	Agra	June
37	Ahmedabad	April
39	Ahmedabad	June
51	Aijal/Aizwal	June
61	Ajmer	April
63	Ajmer	June
73	Akola (A)	April
75	Akola (A)	June
85	Allahabad	April
87	Allahabad	June
97	Ambikapur	April
99	Ambikapur	June
108	Amini Divi	April
110	Amini Divi	June
120	Amritsar (Rajasansi)	April
122	Amritsar (Rajasansi)	June
132	Anantpur	April
134	Anantpur	June
144	Androth	April
146	Androth	June
155	Aurangabad	April
157	Aurangabad	June
167	Balasore	April
169	Balasore	June

	Station_Name	Month
179	Bangalore	April
181	Bangalore	June
...	...	...
999	Pune	April
1001	Pune	June
1011	Raipur	April
1013	Raipur	June
1023	Rajkot (A)	April
1025	Rajkot (A)	June
1035	Ranchi (A)	April
1037	Ranchi (A)	June
1047	Sambalpur	April
1049	Sambalpur	June
1065	Shimla	June
1080	Solapur	April
1082	Solapur	June
1092	Sri Niketan	April
1094	Sri Niketan	June
1103	Srinagar	June
1111	Surat	April
1113	Surat	June
1124	Thiruvananthapuram	June
1134	Tirupathy	April
1136	Tirupathy	June
1155	Udaipur (Dabok)	April
1157	Udaipur (Dabok)	June
1169	Uthagamandalam	June
1178	Varanasi (Babatpur)	April
1180	Varanasi (Babatpur)	June

	Station_Name	Month
1190	Vijayawada	April
1192	Vijayawada	June
1202	Vishakhapatnam	April
1204	Vishakhapatnam	June

182 rows × 2 columns

In [42]:
```
# city in cluster 2
cluster2=dat3_hc[dat3_hc['ClusterID']==2]
# As there are repeated values in the feature station_Name
cluster2[['Station_Name','Month']]
```

Out[42]:

	Station_Name	Month
4	Abu	May
14	Agartala (A)	May
26	Agra	May
38	Ahmedabad	May
50	Aijal/Aizwal	May
62	Ajmer	May
74	Akola (A)	May
86	Allahabad	May
98	Ambikapur	May
109	Amini Divi	May
121	Amritsar (Rajasansi)	May
133	Anantpur	May
145	Androth	May
156	Aurangabad	May
168	Balasore	May
180	Bangalore	May
192	Bareilly	May
204	Baroda (A)	May
216	Belgaum Samra	May
228	Bhagalpur	May
240	Bhatinda	May
252	Bhopal (Bairagarh)	May
264	Bhubaneshwar (A)	May
276	Bhuj (Rudramata)	May
288	Bikaner	May
300	Cannanore	May
309	Chandigarh	May
321	Chennai (Minambakkam)	May

	Station_Name	Month
337	Coimbatore (Pilamedu)	May
349	Cooch Behar	May
...	...	...
885	Nainital	May
893	Nasik	May
905	New Delhi (Palam)	May
917	New Delhi (SFD)	May
929	Palakkad (Palghat)	May
940	Panjim	May
949	Parbhani	May
961	Pasighat	May
969	Patna (A)	May
981	Pondicherry (A)	May
992	Port Blair	May
1000	Pune	May
1012	Raipur	May
1024	Rajkot (A)	May
1036	Ranchi (A)	May
1048	Sambalpur	May
1057	Shillong	May
1064	Shimla	May
1081	Solapur	May
1093	Sri Niketan	May
1102	Srinagar	May
1112	Surat	May
1123	Thiruvananthapuram	May
1135	Tirupathy	May
1147	Tura	May
1156	Udaipur (Dabok)	May

	Station_Name	Month
1168	Uthagamandalam	May
1179	Varanasi (Babatpur)	May
1191	Vijayawada	May
1203	Vishakhapatnam	May

108 rows × 2 columns

In [43]:
```
# city in cluster 3
cluster3=dat3_hc[dat3_hc['ClusterID']==3]
# As there are repeated values in the feature station_Name
cluster3[['Station_Name','Month']]
```

Out[43]:

	Station_Name	Month
6	Abu	September
18	Agartala (A)	September
30	Agra	September
42	Ahmedabad	September
54	Aijal/Aizwal	September
66	Ajmer	September
78	Akola (A)	September
90	Allahabad	September
101	Ambikapur	September
113	Amini Divi	September
125	Amritsar (Rajasansi)	September
137	Anantpur	September
148	Androth	September
160	Aurangabad	September
172	Balasore	September
184	Bangalore	September
196	Bareilly	September
208	Baroda (A)	September
220	Belgaum Samra	September
232	Bhagalpur	September
244	Bhatinda	September
256	Bhopal (Bairagarh)	September
268	Bhubaneshwar (A)	September
280	Bhuj (Rudramata)	September
292	Bikaner	September
301	Cannanore	September
313	Chandigarh	September
325	Chennai (Minambakkam)	September

	Station_Name	Month
341	Coimbatore (Pilamedu)	September
362	Dehra Dun	September
...	...	...
880	Nagpur (Sonegaon)	September
887	Nainital	September
897	Nasik	September
909	New Delhi (Palam)	September
921	New Delhi (SFD)	September
932	Palakkad (Palghat)	September
941	Panjim	September
953	Parbhani	September
973	Patna (A)	September
985	Pondicherry (A)	September
1004	Pune	September
1016	Raipur	September
1028	Rajkot (A)	September
1040	Ranchi (A)	September
1050	Sambalpur	September
1060	Shillong	September
1067	Shimla	September
1073	Silchar	September
1085	Solapur	September
1097	Sri Niketan	September
1106	Srinagar	September
1115	Surat	September
1127	Thiruvananthapuram	September
1139	Tirupathy	September
1148	Tura	September
1160	Udaipur (Dabok)	September

	Station_Name	Month
1171	Uthagamandalam	September
1183	Varanasi (Babatpur)	September
1195	Vijayawada	September
1207	Vishakhapatnam	September

104 rows × 2 columns

In [44]:
```
# city in cluster 4
cluster4=dat3_hc[dat3_hc['ClusterID']==4]
# As there are repeated values in the feature station_Name
cluster4[['Station_Name','Month']]
```

Out[44]:

	Station_Name	Month
16	Agartala (A)	July
28	Agra	July
40	Ahmedabad	July
52	Aijal/Aizwal	July
64	Ajmer	July
76	Akola (A)	July
88	Allahabad	July
111	Amini Divi	July
123	Amritsar (Rajasansi)	July
135	Anantpur	July
158	Aurangabad	July
170	Balasore	July
182	Bangalore	July
194	Bareilly	July
206	Baroda (A)	July
218	Belgaum Samra	July
230	Bhagalpur	July
242	Bhatinda	July
254	Bhopal (Bairagarh)	July
266	Bhubaneshwar (A)	July
278	Bhuj (Rudramata)	July
290	Bikaner	July
311	Chandigarh	July
323	Chennai (Minambakkam)	July
339	Coimbatore (Pilamedu)	July
388	Gadag	July
404	Gaya	July
416	Gopalpur	July

	Station_Name	Month
428	Gorakhpur	July
440	Gulbarga	July
...	...	...
753	Lucknow (Amausi)	July
765	Ludhiana	July
777	Madurai (A)	July
797	Malda	July
806	Manali	July
826	Masulipatnam	July
838	Minicoy	July
847	Mukteswar (Kumaun)	July
866	Mysore	July
878	Nagpur (Sonegaon)	July
895	Nasik	July
907	New Delhi (Palam)	July
919	New Delhi (SFD)	July
951	Parbhani	July
971	Patna (A)	July
983	Pondicherry (A)	July
1002	Pune	July
1014	Raipur	July
1026	Rajkot (A)	July
1038	Ranchi (A)	July
1058	Shillong	July
1083	Solapur	July
1095	Sri Niketan	July
1104	Srinagar	July
1125	Thiruvananthapuram	July
1137	Tirupathy	July

	Station_Name	Month
1158	Udaipur (Dabok)	July
1181	Varanasi (Babatpur)	July
1193	Vijayawada	July
1205	Vishakhapatnam	July

84 rows × 2 columns

In [45]:
```
# city in cluster 5
cluster5=dat3_hc[dat3_hc['ClusterID']==5]
# As there are repeated values in the feature station_Name
cluster5[['Station_Name','Month']]
```

Out[45]:

	Station_Name	Month
17	Agartala (A)	August
29	Agra	August
41	Ahmedabad	August
53	Aijal/Aizwal	August
65	Ajmer	August
77	Akola (A)	August
89	Allahabad	August
100	Ambikapur	August
112	Amini Divi	August
124	Amritsar (Rajasansi)	August
136	Anantpur	August
147	Androth	August
159	Aurangabad	August
171	Balasore	August
183	Bangalore	August
195	Bareilly	August
207	Baroda (A)	August
219	Belgaum Samra	August
231	Bhagalpur	August
243	Bhatinda	August
255	Bhopal (Bairagarh)	August
267	Bhubaneshwar (A)	August
279	Bhuj (Rudramata)	August
291	Bikaner	August
312	Chandigarh	August
324	Chennai (Minambakkam)	August
340	Coimbatore (Pilamedu)	August
389	Gadag	August

	Station_Name	Month
405	Gaya	August
417	Gopalpur	August
...	...	...
807	Manali	August
827	Masulipatnam	August
839	Minicoy	August
848	Mukteswar (Kumaun)	August
867	Mysore	August
879	Nagpur (Sonegaon)	August
896	Nasik	August
908	New Delhi (Palam)	August
920	New Delhi (SFD)	August
931	Palakkad (Palghat)	August
952	Parbhani	August
972	Patna (A)	August
984	Pondicherry (A)	August
1003	Pune	August
1015	Raipur	August
1027	Rajkot (A)	August
1039	Ranchi (A)	August
1059	Shillong	August
1066	Shimla	August
1084	Solapur	August
1096	Sri Niketan	August
1105	Srinagar	August
1114	Surat	August
1126	Thiruvananthapuram	August
1138	Tirupathy	August
1159	Udaipur (Dabok)	August

	Station_Name	Month
1170	Uthagamandalam	August
1182	Varanasi (Babatpur)	August
1194	Vijayawada	August
1206	Vishakhapatnam	August

91 rows × 2 columns

Result

From the data set of 1331 row and 5 feature 6 clusters where generated with corresponding values it’s clear on analysis that city varies in temperature and rainfall on monthly basis. Thus we can find month as a common factor in each cluster.