KMC

Weather Aust

Import Libraries

In [1]:
```
import pandas as pd

from geopy.geocoders import Nominatim
from progressbar import ProgressBar
import time

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt
import plotly.graph_objects as go

import warnings
warnings.filterwarnings('ignore')
```

Import the dataset

In [2]:
```
df=pd.read_csv('weatherAUS_84.csv')
df.head()
```

Out[2]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	1007.7	1007.1	8.0	NaN
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	1010.6	1007.8	NaN	NaN
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	1007.6	1008.7	NaN	2.0
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	1017.6	1012.8	NaN	NaN
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	1010.8	1006.0	7.0	8.0

5 rows × 23 columns

Data Analysis & Pre-processing

In [3]:
```
# Drop records where target RainTomorrow=NaN
df = df[pd.isnull(df['RainTomorrow'])==False]
```

In [4]:
```
# For other columns with missing values, fill them in with column mean
df = df.fillna(df.mean())
```

In [5]:
```
# Add spaces between multiple words in location names
df['Location2']=df['Location'].str.replace( r"([A-Z])", r" \1").str.strip()
```

In [6]:
```
# Update Location for Pearce RAAF so it can be found by geolocator
df['Location2']=df['Location2'].apply(lambda x: 'Pearce, Bullsbrook' if x=='Pearce R A A F' else x)
```

In [7]:
```
# Create a flag for RainToday and RainTomorrow, note RainTomorrowFlag can be used as target variable fro classification
df['RainTodayFlag']=df['RainToday'].apply(lambda x: 1 if x=='Yes' else 0)
df['RainTomorrowFlag']=df['RainTomorrow'].apply(lambda x: 1 if x=='Yes' else 0)
```

In [8]:
```
df.head()
```

Out[8]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday
0	2008-12-01	Albury	13.4	22.9	0.6	5.469824	7.624853	W	44.0	W	...	1007.1	8.000000	4.503167	16.9	21.8	No
1	2008-12-02	Albury	7.4	25.1	0.0	5.469824	7.624853	WNW	44.0	NNW	...	1007.8	4.437189	4.503167	17.2	24.3	No
2	2008-12-03	Albury	12.9	25.7	0.0	5.469824	7.624853	WSW	46.0	W	...	1008.7	4.437189	2.000000	21.0	23.2	No
3	2008-12-04	Albury	9.2	28.0	0.0	5.469824	7.624853	NE	24.0	SE	...	1012.8	4.437189	4.503167	18.1	26.5	No
4	2008-12-05	Albury	17.5	32.3	1.0	5.469824	7.624853	W	41.0	ENE	...	1006.0	7.000000	8.000000	17.8	29.7	No

5 rows × 26 columns

In [9]:
```
# Create a list of unique locations (cities)

loc_list=list(df.Location2.unique())
geolocator = Nominatim(user_agent="add-your-agent-name")
country ="Australia"
loc_res=[]
```

In [10]:
```
pbar=ProgressBar() # This will help us to show the progress of our iteration
for city in pbar(loc_list):
    loc = geolocator.geocode(city+','+ country)
    res = [city, loc.latitude, loc.longitude]
    loc_res = loc_res + [res]
    time.sleep(1) # sleep for 1 second before submitting the next query

# Add locations to a dataframe
df_loc=pd.DataFrame(loc_res, columns=['Loc', 'Latitude', 'Longitude'])

# Show data
df_loc
```

```
100% |########################################################################|
```

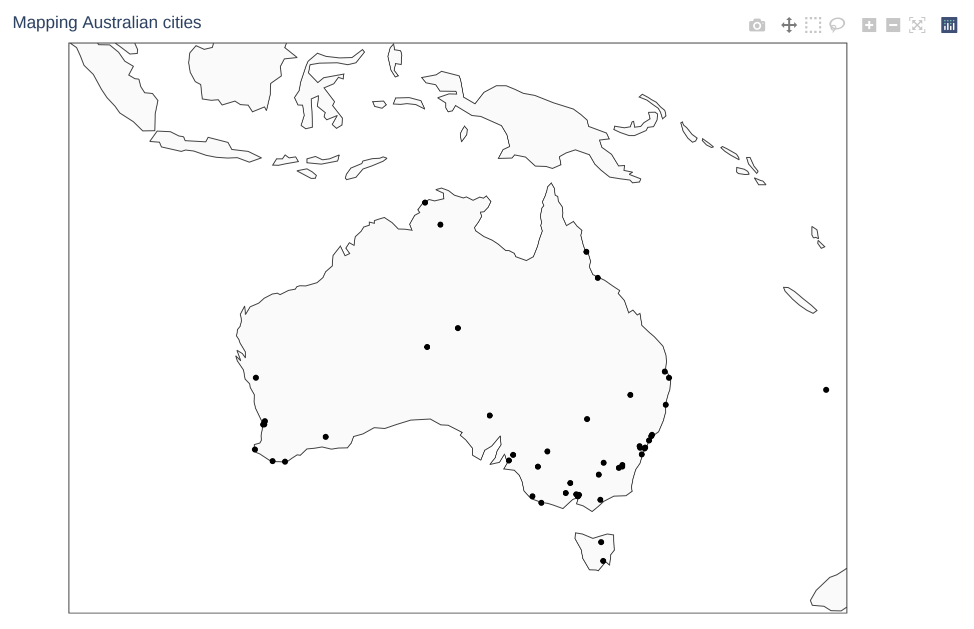
Out[10]:

	Loc	Latitude	Longitude
0	Albury	-36.080477	146.916280
1	Badgerys Creek	-33.881667	150.744163
2	Cobar	-31.498333	145.834444
3	Coffs Harbour	-30.296241	153.113529
4	Moree	-29.461720	149.840715
5	Newcastle	-32.919295	151.779535
6	Norah Head	-33.281667	151.567778
7	Norfolk Island	-29.028958	167.958729
8	Penrith	-33.751195	150.694171
9	Richmond	-37.820395	145.002515
10	Sydney	-33.869844	151.208285
11	Sydney Airport	-33.935309	151.165582
12	Wagga Wagga	-35.115000	147.367778
13	Williamtown	-32.815000	151.842778
14	Wollongong	-34.427808	150.893054
15	Canberra	-35.297591	149.101268
16	Tuggeranong	-35.420977	149.092134
17	Mount Ginini	-35.529744	148.772540
18	Ballarat	-37.562303	143.860565
19	Bendigo	-36.758877	144.282593
20	Sale	-38.105036	147.064790
21	Melbourne Airport	-37.666951	144.833493
22	Melbourne	-37.814218	144.963161
23	Mildura	-34.184726	142.162497
24	Nhil	-35.432528	141.283319
25	Portland	-38.345623	141.604230
26	Watsonia	-37.711002	145.083635
27	Dartmoor	-27.996162	115.189218
28	Brisbane	-27.468968	153.023499
29	Cairns	-16.920666	145.772185
30	Gold Coast	-28.002373	153.414599
31	Townsville	-19.256939	146.823954
32	Adelaide	-34.928181	138.599931
33	Mount Gambier	-37.824670	140.782007
34	Nuriootpa	-34.469335	138.993901
35	Woomera	-31.199914	136.825353
36	Albany	-35.024782	117.883608
37	Witchcliffe	-34.026335	115.100477
38	Pearce, Bullsbrook	-31.673960	116.017544
39	Perth Airport	-31.940610	115.967608
40	Perth	-31.955896	115.860580
41	Salmon Gums	-32.981535	121.643942
42	Walpole	-34.977680	116.731006
43	Hobart	-42.882509	147.328123
44	Launceston	-41.434081	147.137350
45	Alice Springs	-23.698388	133.881289
46	Darwin	-12.460440	130.841047
47	Katherine	-14.464616	132.263599
48	Uluru	-25.345554	131.036961

Geo-graph visualization

In [11]:
```
fig = go.Figure(data=go.Scattergeo(
    lat=df_loc['Latitude'],
    lon=df_loc['Longitude'],
    hovertext=df_loc['Loc'],
    mode = 'markers',
    marker_color = 'black',
    ))
```

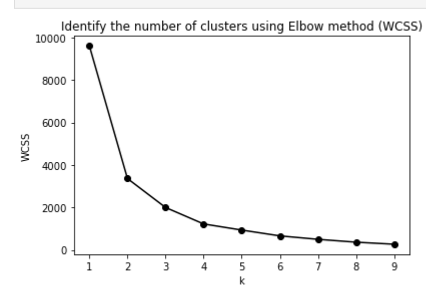
In [12]:
```
fig.update_layout(
    title = 'Mapping Australian cities',
    width=1000,
    height=600,
    margin={"r":10,"t":30,"l":0,"b":0},
    geo = dict(
        scope='world',
        projection_type='miller',
        landcolor = "rgb(250, 250, 250)",
        center=dict(lat=-23.69839, lon=133.8813), # focus point
        projection_scale=5 # zoom in on
        ),
    )
fig.show()
```



Use elbow method to find optimal number of clusters

In [13]:
```
# Identify the number of clusters using Elbow method (WCSS)

WCSS = []

K=range(1,10)
for k in K:
    kmod = KMeans(n_clusters=k)
    kmod.fit(df_loc[['Latitude', 'Longitude']])
    WCSS.append(kmod.inertia_)
```

In [14]:
```
# Plot elbow graph
plt.plot(K, WCSS, 'bo-', color='black')
plt.xlabel('k')
plt.ylabel('WCSS')
plt.title('Identify the number of clusters using Elbow method (WCSS)')
plt.show()
```



Model Building with K=3 & 4

In [15]:
```
# Select data for clustering model
X = df_loc[['Latitude', 'Longitude']]

# Set the model and its parameters - 3 clusters
model3 = KMeans(n_clusters=3,
                init='k-means++', # Smart initialization of centroids, alternative option 'ra
                n_init=10, # Number of time the k-means algorithm will be run with different
                max_iter=100, # maximum number of iterations to run default=300
                )

# Set the model and its parameters - 4 clusters
model4 = KMeans(n_clusters=4,
                init='k-means++', # Smart initialization of centroids, alternative option 'ra
                n_init=10, # Number of time the k-means algorithm will be run with different
                max_iter=100, # maximum number of iterations to run default=300
                )

# Fit the model (3 and 4 clusters)
clust3 = model3.fit(X)
clust4 = model4.fit(X)

# Print model summary
print('*************** 3 Cluster Model ***************')
print('Cluster centers: ', clust3.cluster_centers_)
print('Inertia (WCSS): ', clust3.inertia_)
print('No. of iterations: ', clust3.n_iter_)
print()

print('*************** 4 Cluster Model ***************')
print('Cluster centers: ', clust4.cluster_centers_)
print('Inertia (WCSS): ', clust4.inertia_)
print('No. of iterations: ', clust4.n_iter_)
```

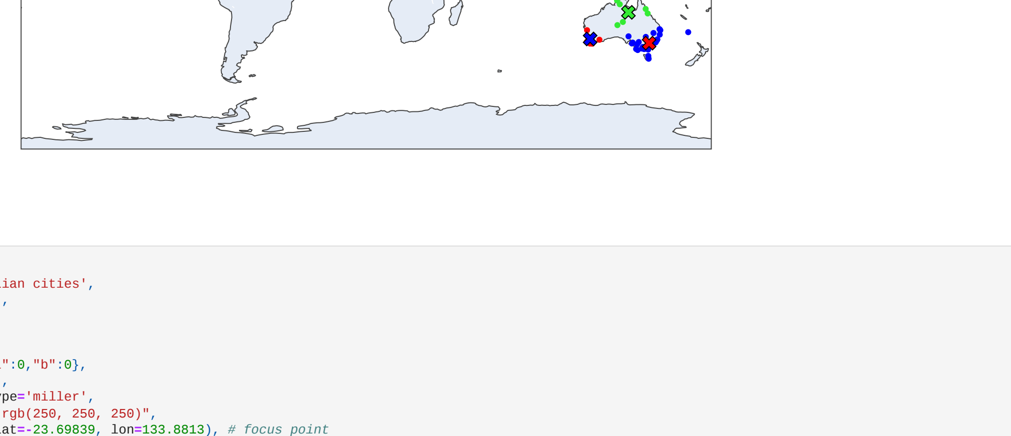
```
*************** 3 Cluster Model ***************
Cluster centers:  [[-34.74921474 147.54680922]
 [-32.57211995 116.79924778]
 [-18.69110057 136.76983921]]
Inertia (WCSS):  1994.7309795181147
No. of iterations:  3

*************** 4 Cluster Model ***************
Cluster centers:  [[-32.57211995 116.79924778]
 [-32.4617723  152.13827238]
 [-18.69110057 136.76983921]
 [-36.67548207 143.68031393]]
Inertia (WCSS):  1219.1626081918919
No. of iterations:  4
```

Cluster visualization

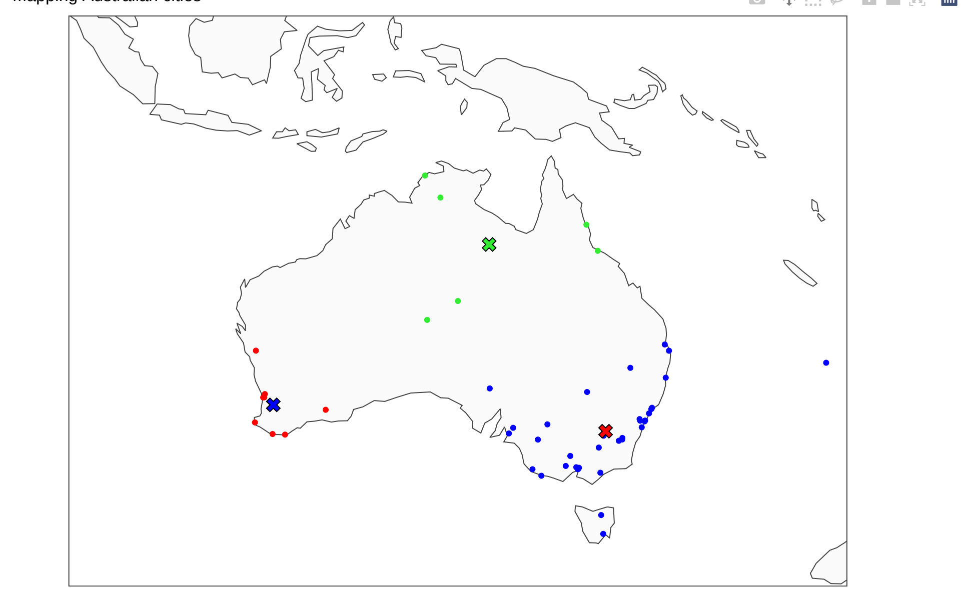
In [16]:
```
# Attach cluster labels to the original dataset
df_loc['Clust3']=clust3.labels_
df_loc['Clust4']=clust4.labels_
```

In [17]:
```
# Plot cluster on the map
fig = go.Figure(data=go.Scattergeo(
    lat=df_loc['Latitude'],
    lon=df_loc['Longitude'],
    hovertext=df_loc[['Loc', 'Clust3']],
    mode = 'markers',
    marker=dict(colorscale=['blue', 'red', '#34eb34']),
    marker_color = df_loc['Clust3'],
))
```

In [18]:
```
# Add traces
fig.add_trace(go.Scattergeo(lat=model3.cluster_centers_[:,0], lon=model3.cluster_centers_[:,1],
                            mode='markers', marker_symbol='x', marker_size=12,
                            marker_color=['red', 'blue', '#34eb34'],
                            marker_line_color='black',
                            marker_line_width=1
                            ))
```



In [19]:
```
fig.update_layout(
    title = 'Mapping Australian cities',
    title_font_color='black',
    showlegend=False,
    width=1000,
    height=600,
    margin={"r":10,"t":30,"l":0,"b":0},
    geo = dict(scope='world',
               projection_type='miller',
               landcolor = "rgb(250, 250, 250)",
               center=dict(lat=-23.69839, lon=133.8813), # focus point
               projection_scale=5 # zoom in on
            ),
        )
fig.show()
```



In [ ]: