In [1]:
```
## PREDICT WEATHER CONDITIONS USING MACHINE LEARNING -- use case

## Importing Libraries

%matplotlib inline
from copy import deepcopy ## not just point to it copy it over using deepcopy
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = (16,9)
plt.style.use('ggplot')
```

In [5]:
```
## importing the dataset

data = pd.read_csv('https://raw.githubusercontent.com/jupyter/docker-demo-images/master/datasets/cluster/xclara.csv')
print(data.shape)
data.head()
```

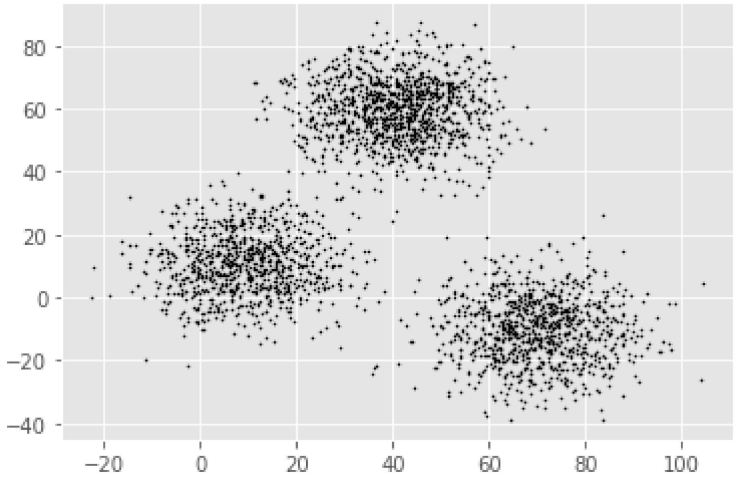
```
(3000, 3)
```

Out[5]:

	Unnamed: 0	V1	V2
0	1	2.072345	-3.241693
1	2	17.936710	15.784810
2	3	1.083576	7.319176
3	4	11.120670	14.406780
4	5	23.711550	2.557729

In [16]:
```
## Getting the values and plotting it

f2 = data['V1'].values
f3 = data['V2'].values
x = np.array(list(zip(f2,f3)))
plt.scatter(f2, f3, c='black',s=1)
```

Out[16]: <matplotlib.collections.PathCollection at 0x2a0aed7dc48>

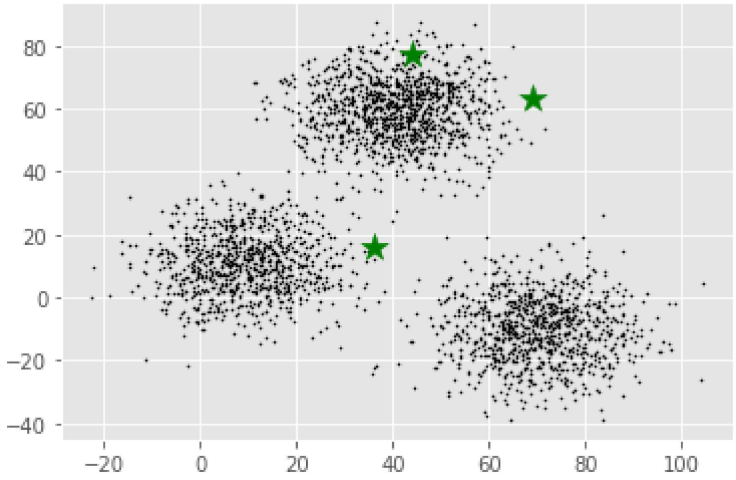


In [12]:
```
## Euclidean Distance Calculator
def dist(a, b, ax=1):
    return np.linalg.norm(a - b, axis=ax)

## Number of clusters
k = 3

## x coordinates of random centroids
C_x = np.random.randint(0, np.max(x)-20, size=k)

## y coordinates of random centroids
C_y = np.random.randint(0, np.max(x)-20, size=k)
C = np.array(list(zip(C_x, C_y)), dtype=np.float32)
print(C)
```

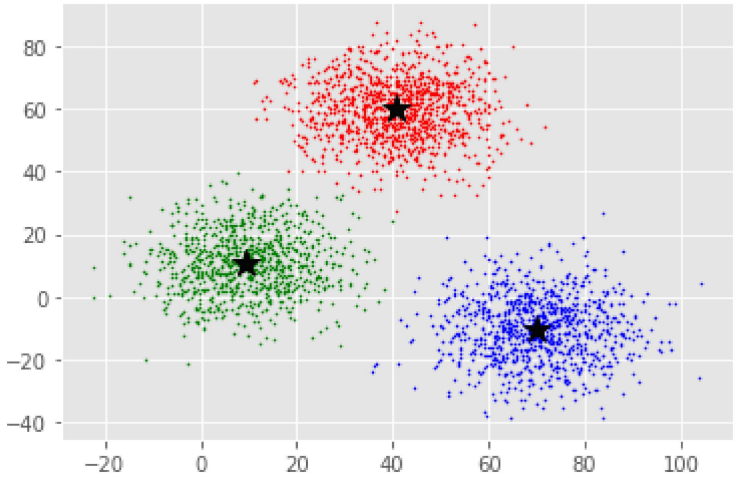
```
[[69. 63.]
 [44. 77.]
 [36. 16.]]
```

In [15]:
```
## Plotting along with the centroids

plt.scatter(f2, f3, c='#050505', s=1)
plt.scatter(C_x, C_y, marker='*', s=200, c='g')
```

Out[15]: <matplotlib.collections.PathCollection at 0x2a0aed1ad88>



In [26]:
```
## To store the value of centroids when it updates
C_old = np.zeros(C.shape)

# Cluster Lables(0,1,2)
clusters = np.zeros(len(x))

#Error func. - Distance between new centroids and old centroids
error = dist(C, C_old, None)

# Loop will run till the error becomes zero
while error != 0:

    ## Assigning each value to its closest cluster
    for i in range(len(x)):
        distances = dist(x[i], C)
        cluster = np.argmin(distances)
        clusters[i] = cluster

    # Storing the old centroid values
    C_old = deepcopy(C)

    # Finding the new centroids by taking the average value
    for i in range(k):
        points = [x[j] for j in range(len(x)) if clusters[j] == i]
        C[i] = np.mean(points, axis=0)
        error = dist(C, C_old, None)

    colors = ['r', 'g', 'b', 'y', 'c', 'm']
    fig, ax = plt.subplots()

    for i in range(k):
        points = np.array([x[j] for j in range(len(x)) if clusters[j] == i])
        ax.scatter(points[:, 0], points[:, 1], s=1, c=colors[i])
        ax.scatter(C[:, 0], C[:, 1], marker='*', s=200, c='#050505')

## blue region with the highest temperature and lowest pressure will have high
rainfall

# red and green regions will have lesser rainfall
```



In [ ]: