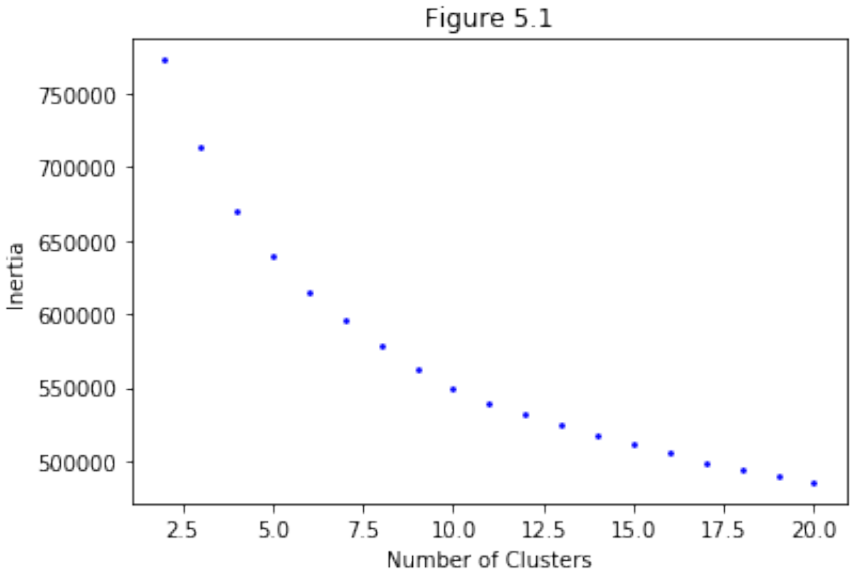
In [1]:
```
# Import required packages here (after they are installed)
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as mp
from pylab import show

# Load data. csv file should be in the same folder as the notebook for
this to work, otherwise
# give data path.
data = np.loadtxt("data.csv")
```

In [2]:
```
#shuffle the data and select training and test data
np.random.seed(100)
np.random.shuffle(data)


features = [[]]

for row in data:
    #import the data
        features.append(row[1:])
```

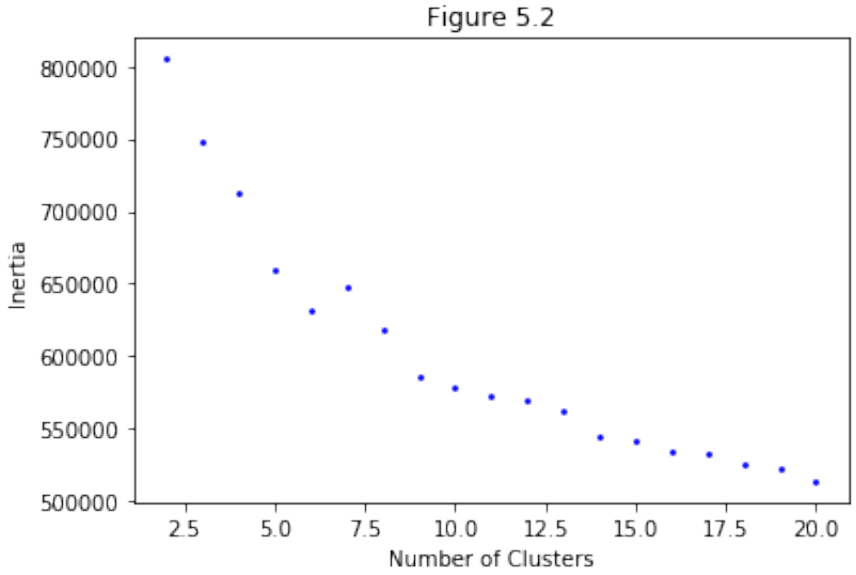
In [17]:
```
numClusters = range( 2, 21)
inertias = []

for numCluster in numClusters:
    model = KMeans( n_clusters = numCluster)
    model.fit( features)
    inertias.append( model.inertia_)
```

In [20]:
```
## Visualize Results
#plot the points
mp.scatter( numClusters, inertias,s=3,c="blue")

#setup the axes
mp.xlabel("Number of Clusters")
mp.ylabel("Inertia")

#label the figure
mp.title("Figure 5.1")

show()
```



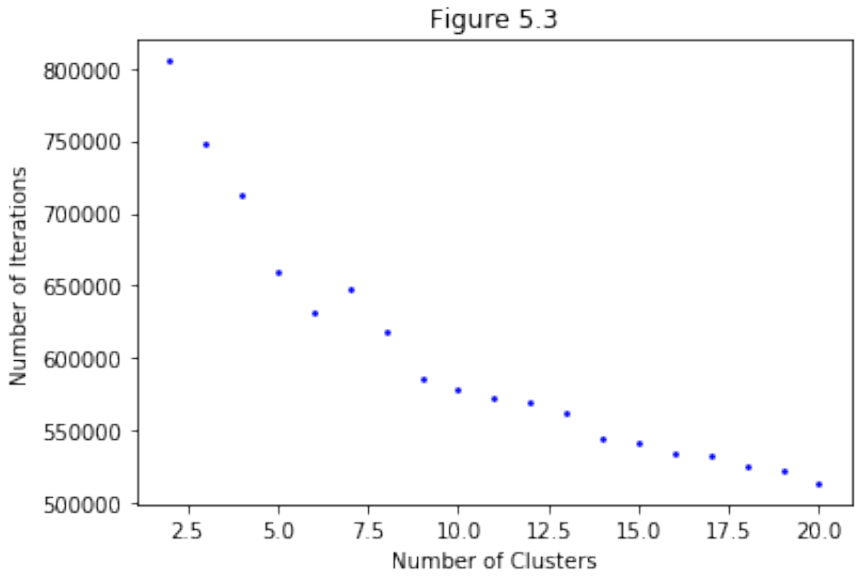
According to the inertia attribute values, the best values for n_clusters should be 20. This matches my expectations because I would assume that the more clusters there are, the smaller the distances would be from each point to the nearest cluster center on average.

In [21]:
```
numClusters = range( 2, 21)
inertias = []

for numCluster in numClusters:
    model = KMeans( n_clusters = numCluster, n_init = 1, max_iter = 1)
    model.fit( features)
    inertias.append( model.inertia_)


## Visualize Results
#plot the points
mp.scatter( numClusters, inertias,s=3,c="blue")

#setup the axes
mp.xlabel("Number of Clusters")
mp.ylabel("Inertia")

#label the figure
mp.title("Figure 5.2")

show()
```



In the second graph (using n_init = 1 and max_iter = 1), all the inertias seem to be slightly higher, so points are not as close to the cluster center on average. However, besides a couple outliers at n_clusters = 7 and 8, the overall trend seems to be the same for both graphs: as n_clusters increases, inertia decreases.

In [22]:
```
numClusters = range( 2, 21)
numIters = []

for numCluster in numClusters:
    model = KMeans( n_clusters = numCluster)
    model.fit( features)
    numIters.append( model.n_iter_)

## Visualize Results
#plot the points
mp.scatter( numClusters, inertias,s=3,c="blue")

#setup the axes
mp.xlabel("Number of Clusters")
mp.ylabel("Number of Iterations")

#label the figure
mp.title("Figure 5.3")

show()
```



I expected the number of iterations to increase as the number of cluseters increased because I thought if there were more clusters, more iterations would be needed to determine the best centroid seeds. However, the data does not match my expectations because, in reality, for the most part, number of iterations decreases as number of clusters increases. This is likely because the algorithm that implements this model is efficient amd clever and does not run as I would have guessed.