In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
from sklearn.datasets import make_blobs
```

In [3]:
```
#import the dataset (data file needs to be in the same folder as noteb
#missing data pounts removed
data = pd.read_csv("clustering1.csv")
print("Imput Data and Shape")
print(data.shape)
data.head()
```

```
Imput Data and Shape
(115, 37)
```

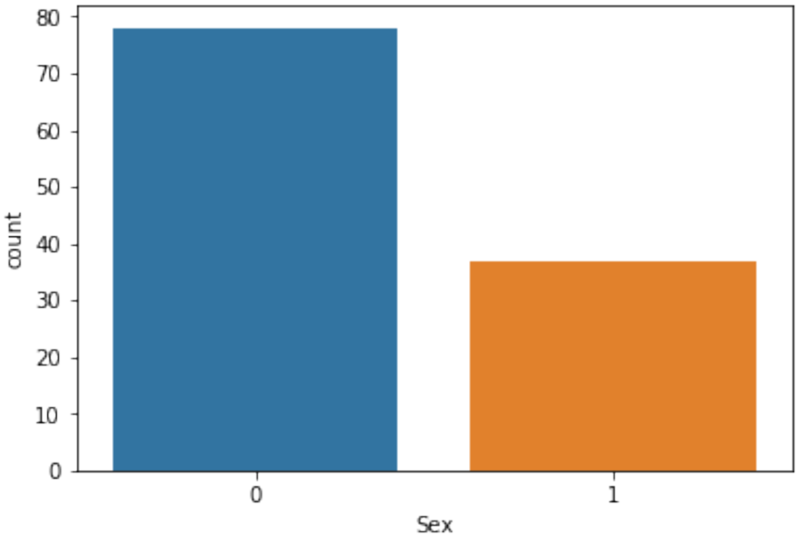
Out[3]:

	Unnamed: 0	Sex	BMI	Stroop_Accuracy	GDT_Risky_Dice_Percentage	WSCT_Accuracy
0	S0001	0	16.184275	0.991667	0.222222	0.8167
1	S0004	0	25.390219	0.991667	0.000000	0.8833
2	S0005	0	19.146722	1.000000	0.055556	0.8167
3	S0006	1	20.761246	0.966667	0.166667	0.8833
4	S0009	0	23.533043	0.991667	0.444444	0.8500

5 rows × 37 columns

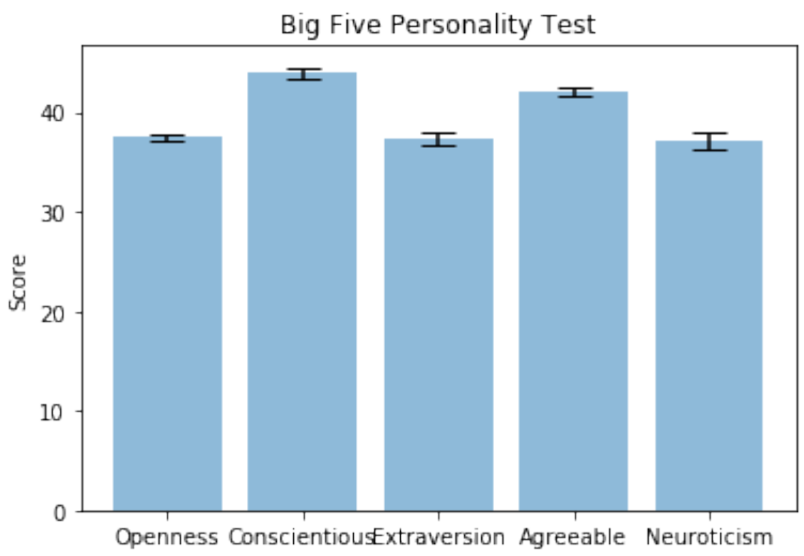
In [102]:
```
print(data['Sex'].unique())
print(data.groupby('Sex').size())

import seaborn as sns
sns.countplot(data['Sex'],label="Count")
plt.savefig('Sex_Difference.png')
plt.show()
```

```
[0 1]
Sex
0    78
1    37
dtype: int64
```

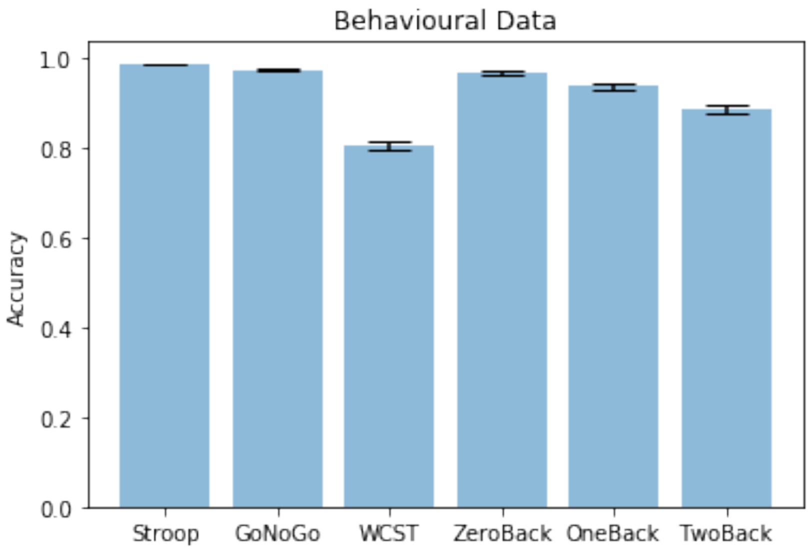


In [99]:

```
m_Openness = np.mean(data['Openness'])
sem_Openness = np.std(data['Openness'])/np.sqrt(115)
m_Conscientious = np.mean(data['Conscientious'])
sem_Conscientious = np.std(data['Conscientious'])/np.sqrt(115)
m_Extraversion = np.mean(data['Extraversion'])
sem_Extraversion = np.std(data['Extraversion'])/np.sqrt(115)
m_Agreeable = np.mean(data['Agreeable'])
sem_Agreeable = np.std(data['Agreeable'])/np.sqrt(115)
m_Neuroticism = np.mean(data['Neuroticism'])
sem_Neuroticism = np.std(data['Neuroticism'])/np.sqrt(115)


means = [m_Openness, m_Conscientious, m_Extraversion, m_Agreeable, m_N
errorbars = [sem_Openness, sem_Conscientious, sem_Extraversion, sem_Ag


Group=['Openness', 'Conscientious', 'Extraversion', 'Agreeable', 'Neur
x_pos = np.arange(len(Group))

fig, ax= plt.subplots()

ax.bar(x_pos, means, yerr=errorbars, align='center', alpha=0.5, ecolor
ax.set_ylabel('Score')
ax.set_xticks(x_pos)
ax.set_xticklabels(Group)
ax.set_title('Big Five Personality Test')
plt.savefig('big5.png')
```



In [100]:
```
m_Stroop = np.mean(data['Stroop_Accuracy'])
sem_Stroop = np.std(data['Stroop_Accuracy'])/np.sqrt(115)
m_GoNoGo = np.mean(data['GoNoGo_Accuracy'])
sem_GoNoGo = np.std(data['GoNoGo_Accuracy'])/np.sqrt(115)
m_WCST = np.mean(data['WSCT_Accuracy'])
sem_WCST = np.std(data['WSCT_Accuracy'])/np.sqrt(115)
m_ZeroBack = np.mean(data['NBack_ZeroBack_Accuracy'])
sem_ZeroBack = np.std(data['NBack_ZeroBack_Accuracy'])/np.sqrt(115)
m_OneBack = np.mean(data['NBack_OneBack_Accuracy'])
sem_OneBack = np.std(data['NBack_OneBack_Accuracy'])/np.sqrt(115)
m_TwoBack = np.mean(data['NBack_TwoBack_Accuracy'])
sem_TwoBack = np.std(data['NBack_TwoBack_Accuracy'])/np.sqrt(115)

means = [m_Stroop, m_GoNoGo, m_WCST, m_ZeroBack, m_OneBack, m_TwoBack]
errorbars = [sem_Stroop, sem_GoNoGo, sem_WCST, sem_ZeroBack, sem_OneBac


Group=['Stroop', 'GoNoGo', 'WCST', 'ZeroBack', 'OneBack', 'TwoBack']
x_pos = np.arange(len(Group))

fig, ax= plt.subplots()

ax.bar(x_pos, means, yerr=errorbars, align='center', alpha=0.5, ecolor=
ax.set_ylabel('Accuracy')
ax.set_xticks(x_pos)
ax.set_xticklabels(Group)
ax.set_title('Behavioural Data')
plt.savefig('accuracy.png')
```



In [51]:
```
data['BMI'].describe()
```

Out[51]:
```
count    115.000000
mean      21.503261
std        3.449098
min       16.184275
25%       19.472274
50%       20.829995
75%       23.068114
max       36.203614
Name: BMI, dtype: float64
```

In [54]:
```
data['Stroop_Accuracy'].describe()
```

Out[54]:
```
count    115.000000
mean       0.985217
std        0.018406
min        0.908333
25%        0.983333
50%        0.991667
75%        1.000000
max        1.000000
Name: Stroop_Accuracy, dtype: float64
```

In [58]:
```
data['GoNoGo_Accuracy'].describe()
```

Out[58]:
```
count    114.000000
mean       0.971754
std        0.028634
min        0.860000
25%        0.960000
50%        0.980000
75%        1.000000
max        1.000000
Name: GoNoGo_Accuracy, dtype: float64
```

In [62]:
```
data['WSCT_Accuracy'].describe()
```

Out[62]:
```
count    115.000000
mean       0.806230
std        0.104633
min        0.216700
25%        0.783300
50%        0.833300
75%        0.858350
max        0.916700
Name: WSCT_Accuracy, dtype: float64
```

In [64]:
```
data['NBack_ZeroBack_Accuracy'].describe()
```

Out[64]:
```
count    115.000000
mean       0.967633
std        0.047107
min        0.638889
25%        0.972222
50%        0.972222
75%        1.000000
max        1.000000
Name: NBack_ZeroBack_Accuracy, dtype: float64
```

In [65]:
```
data['NBack_OneBack_Accuracy'].describe()
```

Out[65]:
```
count    115.000000
mean       0.936473
std        0.088895
min        0.250000
25%        0.916667
50%        0.944444
75%        0.972222
max        1.000000
Name: NBack_OneBack_Accuracy, dtype: float64
```

In [66]:
```
data['NBack_TwoBack_Accuracy'].describe()
```

Out[66]:
```
count    115.000000
mean       0.886473
std        0.111267
min        0.333333
25%        0.847222
50%        0.916667
75%        0.944444
max        1.000000
Name: NBack_TwoBack_Accuracy, dtype: float64
```

In [57]:
```
data['GDT_Risky_Dice_Percentage'].describe()
```

Out[57]:
```
count    115.000000
mean       0.182126
std        0.240652
min        0.000000
25%        0.000000
50%        0.111111
75%        0.222222
max        1.000000
Name: GDT_Risky_Dice_Percentage, dtype: float64
```

In [71]:
```
data['IGT_B'].describe()
```

Out[71]:
```
count    115.000000
mean       0.380174
std        0.182848
min        0.010000
25%        0.262500
50%        0.355000
75%        0.467500
max        0.900000
Name: IGT_B, dtype: float64
```

In [72]:
```
data['Neuroticism'].describe()
```

Out[72]:
```
count    115.000000
mean      37.182609
std        9.187694
min       13.000000
25%       31.000000
50%       37.000000
75%       45.000000
max       59.000000
Name: Neuroticism, dtype: float64
```

In [73]:
```
data['Extraversion'].describe()
```

Out[73]:
```
count    115.000000
mean      37.304348
std        6.852038
min       18.000000
25%       33.000000
50%       37.000000
75%       41.000000
max       55.000000
Name: Extraversion, dtype: float64
```

In [74]:
```
data['Openness'].describe()
```

Out[74]:
```
count    115.000000
mean      37.469565
std        4.323177
min       26.000000
25%       35.000000
50%       38.000000
75%       41.000000
max       46.000000
Name: Openness, dtype: float64
```

In [75]:
```
data['Agreeable'].describe()
```

Out[75]:
```
count    115.000000
mean      42.147826
std        4.507778
min       30.000000
25%       39.500000
50%       43.000000
75%       45.000000
max       52.000000
Name: Agreeable, dtype: float64
```

In [76]:
```
data['Conscientious'].describe()
```

Out[76]:
```
count    115.000000
mean      43.895652
std        6.343077
min       27.000000
25%       40.000000
50%       45.000000
75%       48.000000
max       59.000000
Name: Conscientious, dtype: float64
```

In [77]:
```
data['BART'].describe()
```

Out[77]:
```
count    115.000000
mean      27.986297
std       13.406352
min        4.500000
25%       18.214286
50%       26.000000
75%       36.550000
max       60.750000
Name: BART, dtype: float64
```

In [79]:
```
data['IGT Score'].describe()
```

Out[79]:
```
count    115.000000
mean      -0.678261
std       77.193258
min     -188.000000
25%      -54.000000
50%       -4.000000
75%       53.000000
max      194.000000
Name: IGT Score, dtype: float64
```

In [80]:
```
data['DDT_AUC'].describe()
```

Out[80]:
```
count    115.000000
mean       0.615548
std        0.274932
min        0.192761
25%        0.384512
50%        0.568013
75%        0.898990
max        1.000000
Name: DDT_AUC, dtype: float64
```

In [81]:
```
data['PDT_AUC'].describe()
```

Out[81]:
```
count    115.000000
mean       0.459350
std        0.140894
min        0.180135
25%        0.343939
50%        0.461616
75%        0.544613
max        0.837037
Name: PDT_AUC, dtype: float64
```

In [82]:
```
data['BAS_D'].describe()
```

Out[82]:
```
count    115.000000
mean      11.652174
std        2.106979
min        8.000000
25%       11.000000
50%       12.000000
75%       13.000000
max       16.000000
Name: BAS_D, dtype: float64
```

In [83]:
```
data['BAS_FS'].describe()
```

Out[83]:
```
count    115.000000
mean      11.930435
std        1.931828
min        7.000000
25%       11.000000
50%       12.000000
75%       13.000000
max       16.000000
Name: BAS_FS, dtype: float64
```

In [84]:
```
data['BAS_RR'].describe()
```

Out[84]:
```
count    115.000000
mean      17.113043
std        1.927480
min       12.000000
25%       15.500000
50%       17.000000
75%       19.000000
max       20.000000
Name: BAS_RR, dtype: float64
```

In [85]:
```
data['BIS'].describe()
```

Out[85]:
```
count    115.000000
mean      21.208696
std        3.406639
min       14.000000
25%       19.000000
50%       21.000000
75%       24.000000
max       28.000000
Name: BIS, dtype: float64
```

In [86]:
```
data['PANAS_PA'].describe()
```

Out[86]:
```
count    115.000000
mean      27.573913
std        6.260515
min       14.000000
25%       23.000000
50%       26.000000
75%       32.000000
max       46.000000
Name: PANAS_PA, dtype: float64
```

In [87]:
```
data['PANAS_NA'].describe()
```

Out[87]:
```
count    115.000000
mean      23.643478
std        7.925300
min       10.000000
25%       19.000000
50%       22.000000
75%       27.500000
max       50.000000
Name: PANAS_NA, dtype: float64
```

In [88]:
```
data['DividedAttention'].describe()
```

Out[88]:
```
count    115.000000
mean       0.100894
std        0.088469
min        0.000000
25%        0.035714
50%        0.091429
75%        0.129286
max        0.631429
Name: DividedAttention, dtype: float64
```

In [89]:
```
data['SPAI_10'].describe()
```

Out[89]:
```
count    115.000000
mean      24.173913
std        4.935125
min       11.000000
25%       21.000000
50%       24.000000
75%       28.000000
max       38.000000
Name: SPAI_10, dtype: float64
```

In [90]:
```
data['CIAS'].describe()
```

Out[90]:
```
count    115.000000
mean      55.043478
std       11.505446
min       30.000000
25%       47.000000
50%       54.000000
75%       63.000000
max       89.000000
Name: CIAS, dtype: float64
```

In [91]:
```
data['PMGQ'].describe()
```

Out[91]:
```
count    115.000000
mean      11.947826
std        9.723225
min        0.000000
25%        0.000000
50%       14.000000
75%       20.000000
max       32.000000
Name: PMGQ, dtype: float64
```

In [92]:
```
data['IGD'].describe()
```

Out[92]:
```
count    115.000000
mean       1.000000
std        1.611363
min        0.000000
25%        0.000000
50%        0.000000
75%        2.000000
max        6.000000
Name: IGD, dtype: float64
```

In [103]:
```
data['Stress'].describe()
```

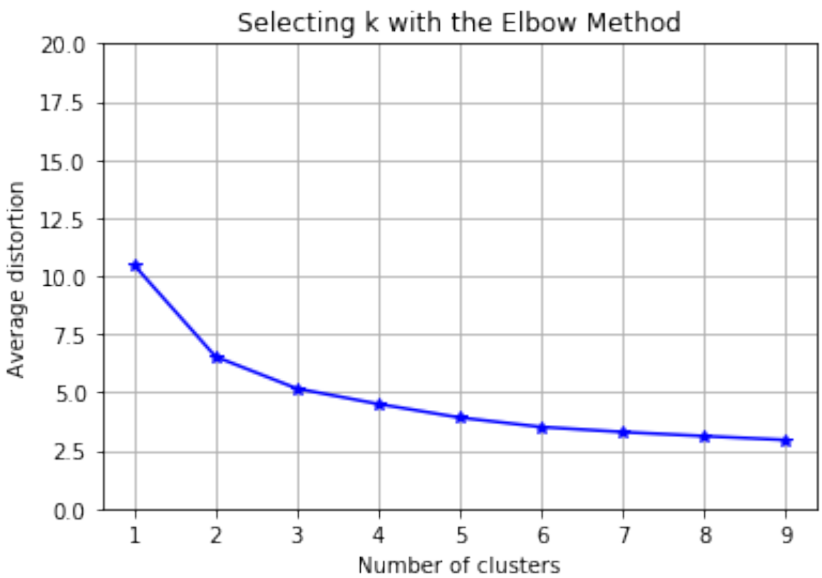
Out[103]:
```
count    115.000000
mean      12.600000
std        3.028519
min        7.000000
25%       10.000000
50%       13.000000
75%       14.000000
max       22.000000
Name: Stress, dtype: float64
```

In [19]:
```
#value input for each column
f0 = data['Sex'].values
f1 = data['BMI'].values
f2 = data['Stroop_Accuracy'].values
f3 = data['GDT_Risky_Dice_Percentage'].values
f4 = data['WSCT_Accuracy'].values
f5 = data['GoNoGo_Accuracy'].values
f6 = data['NBack_ZeroBack_Accuracy'].values
f7 = data['NBack_OneBack_Accuracy'].values
f8 = data['NBack_TwoBack_Accuracy'].values
f9 = data['IGT_A'].values
f10 = data['IGT_B'].values
f11 = data['IGT_C'].values
f12 = data['IGT_D'].values
f13 = data['IGT Score'].values
f14 = data['DDT_AUC'].values
f15 =data['PDT_AUC'].values
f16 = data['BART'].values
f17 = data['Stress'].values
f18 = data['Neuroticism'].values
f19 = data['Extraversion'].values
f20 = data['Openness'].values
f21 = data['Agreeable'].values
f22 = data['Conscientious'].values
f23 = data['BAS_D'].values #BAS Drive
f24 = data['BAS_FS'].values #BAS Fun Seeking
f25 = data['BAS_RR'].values # BAS Reward Response
f26 = data['BAS'].values
f27 = data['BIS'].values
f28 = data['PANAS_PA'].values
f29 = data['PANAS_NA'].values
f30 = data['DividedAttention'].values
f31 = data['SPAI_10'].values #smartphone addiction scores
f32 = data['CIAS'].values #chen's internet addiction scales
f33 = data['PMGQ'].values #problematic mobile gaming questionnaires
f34 = data['IGD'].values #internet gaming disorder
```

In [16]:
```
#Focused research topic 1: technology addiction
#Smartphone & Internet addiction
A = np.array(list(zip(f31,f32)))
```

In [17]:
```
#elbow plot
##### cluster data into k=1....10 clusters #####
K_range=range(1,10)
distortions = []

for i in K_range:
    kmModel = KMeans(n_clusters=i)
    kmModel.fit(A)
    distortions.append(sum(np.min(cdist(A, kmModel.cluster_centers_, '

fig1 = plt.figure()
ex = fig1.add_subplot(111)
ex.plot(K_range, distortions, 'b*-')
#mark the elboq
#ex.plot(K_range[2], distortions[2], marker='o', markersixe=12, marker

plt.grid(True)
#plt.xlim([0,20])
plt.ylim([0, 20])
plt.xlabel('Number of clusters')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
```

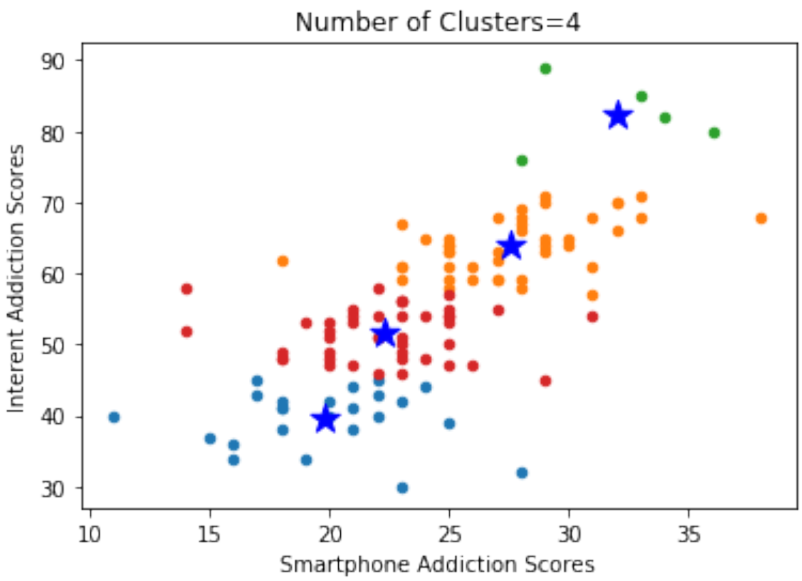
Out[17]: Text(0.5,1,'Selecting k with the Elbow Method')



In [41]:
```
##Nuber of clusters
k=4

##X coordinates of random centroids
#C_x = np.random.randint(0, np.max(X), size=k)
##Y coordinates of random centroids
#C_y = np.random.randint(0, np.max(X), size=k)
#C = np.array(list(zip(C_x, C_y)), dtype=np.float32)

##plotting along with the Centroids
#plt.scatter(f1, f2, c='#050505', s=20)
```

```
#plt.scatter(C_x, C_y, marker='*', s=200, c='g')
#plt.xlabel('IGD')
#plt.ylabel('PMG')
#plt.title('Internet Gaming & Mobile Gaming')

#cluster plot
kmeans = KMeans(n_clusters=k)
#Fitting the imput data
kmeans = kmeans.fit(A)
#Getting the cluster labels
labels = kmeans.predict(A)
#Centroid values
centroids = kmeans.cluster_centers_
colors = ['r', 'g', 'y', 'm', 'o', 'w']
fig2 = plt.figure()
kg = fig2.add_subplot(111)

for i in range(k):
    points = np.array([A[j] for j in range(len(A)) if labels[j] == i])
    kg.scatter(points[:,0], points[:,1], s=20, cmap='rainbow')


kg.scatter(centroids[:,0], centroids[:,1], marker='*', s=200, c='b')
print('Fianl Centroids')
print('centroids')
plt.xlabel('Smartphone Addiction Scores')
plt.ylabel('Interent Addiction Scores')
plt.title('Number of Clusters={}'.format(k))
plt.savefig('A_Clustering.png')
```

```
Fianl Centroids
centroids
```

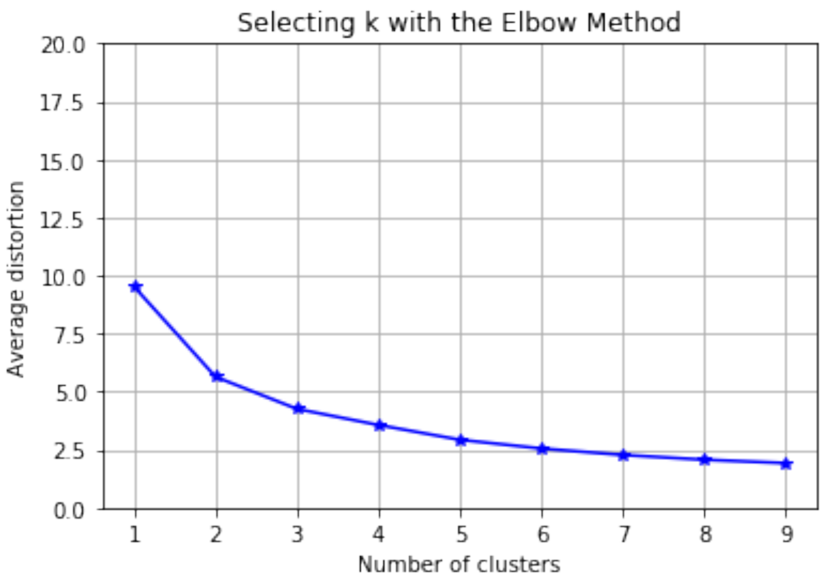


In [20]:

```
#Internet addiction & Internet Gaming addiction
B = np.array(list(zip(f32,f34)))
```

In [21]:
```
#elbow plot
##### cluster data into k=1....10 clusters #####
K_range=range(1,10)
distortions = []

for i in K_range:
    kmModel = KMeans(n_clusters=i)
    kmModel.fit(B)
    distortions.append(sum(np.min(cdist(B, kmModel.cluster_centers_, 'e

fig1 = plt.figure()
ex = fig1.add_subplot(111)
ex.plot(K_range, distortions, 'b*-')
#mark the elboq
#ex.plot(K_range[2], distortions[2], marker='o', markersixe=12, markere

plt.grid(True)
#plt.xlim([0,20])
plt.ylim([0, 20])
plt.xlabel('Number of clusters')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
```

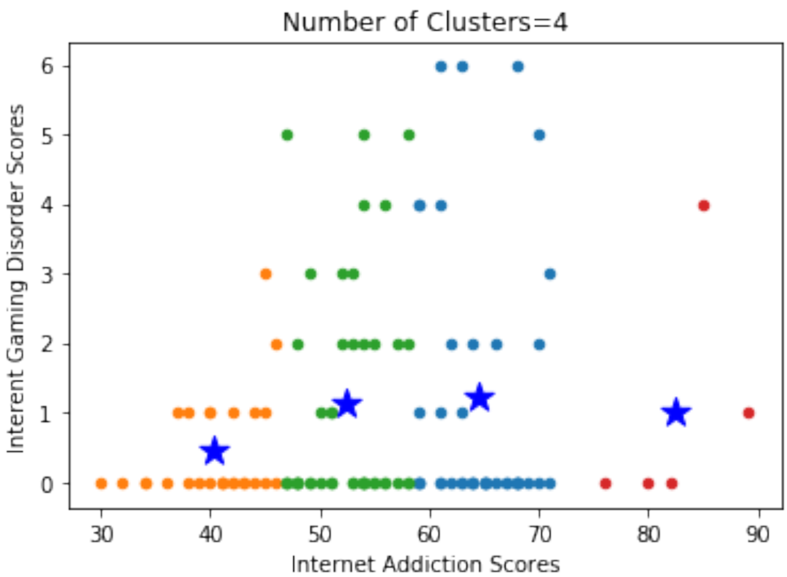
Out[21]: Text(0.5,1,'Selecting k with the Elbow Method')



In [43]:
```
##Nuber of clusters
k=4

##X coordinates of random centroids
#C_x = np.random.randint(0, np.max(X), size=k)
##Y coordinates of random centroids
#C_y = np.random.randint(0, np.max(X), size=k)
#C = np.array(list(zip(C_x, C_y)), dtype=np.float32)

##plotting along with the Centroids
#plt.scatter(f1, f2, c='#050505', s=20)
```

```
#plt.scatter(C_x, C_y, marker='*', s=200, c='g')
#plt.xlabel('IGD')
#plt.ylabel('PMG')
#plt.title('Internet Gaming & Mobile Gaming')

#cluster plot
kmeans = KMeans(n_clusters=k)
#Fitting the imput data
kmeans = kmeans.fit(B)
#Getting the cluster labels
labels = kmeans.predict(B)
#Centroid values
centroids = kmeans.cluster_centers_
colors = ['r', 'g', 'y', 'm', 'o', 'w']
fig2 = plt.figure()
kg = fig2.add_subplot(111)

for i in range(k):
    points = np.array([B[j] for j in range(len(B)) if labels[j] == i])
    kg.scatter(points[:,0], points[:,1], s=20, cmap='rainbow')


kg.scatter(centroids[:,0], centroids[:,1], marker='*', s=200, c='b')
print('Fianl Centroids')
print('centroids')
plt.xlabel('Internet Addiction Scores')
plt.ylabel('Interent Gaming Disorder Scores')
plt.title('Number of Clusters={}'.format(k))
plt.savefig('B_clustering.png')
```

```
Fianl Centroids
centroids
```

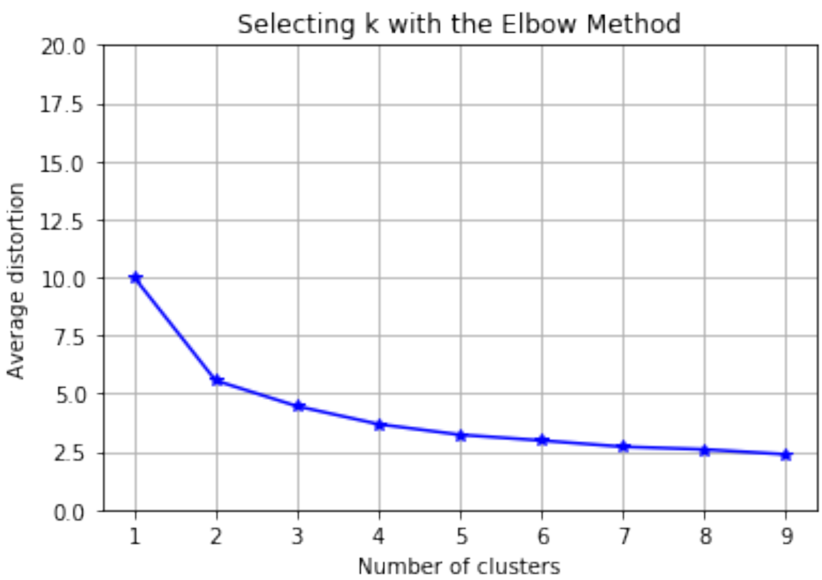


In [24]:

```
#Smartphone addiction & Problematic Smartphone Gaming
C = np.array(list(zip(f31,f33)))
```

In [26]:
```
#elbow plot
##### cluster data into k=1....10 clusters #####
K_range=range(1,10)
distortions = []

for i in K_range:
    kmModel = KMeans(n_clusters=i)
    kmModel.fit(C)
    distortions.append(sum(np.min(cdist(C, kmModel.cluster_centers_, 'eu

fig1 = plt.figure()
ex = fig1.add_subplot(111)
ex.plot(K_range, distortions, 'b*-')
#mark the elboq
#ex.plot(K_range[2], distortions[2], marker='o', markersixe=12, markerec

plt.grid(True)
#plt.xlim([0,20])
plt.ylim([0, 20])
plt.xlabel('Number of clusters')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
```

Out[26]: Text(0.5,1,'Selecting k with the Elbow Method')

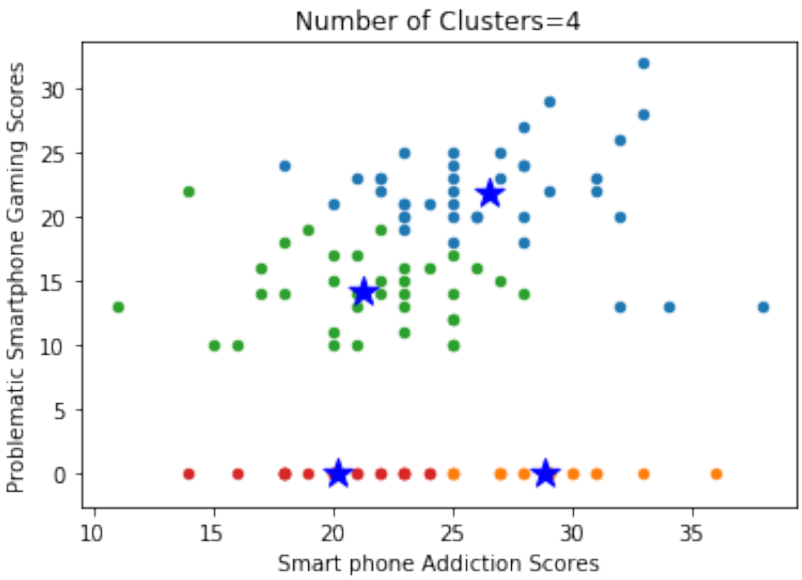


In [28]:
```
##Nuber of clusters
k=4

##X coordinates of random centroids
#C_x = np.random.randint(0, np.max(X), size=k)
##Y coordinates of random centroids
#C_y = np.random.randint(0, np.max(X), size=k)
#C = np.array(list(zip(C_x, C_y)), dtype=np.float32)

##plotting along with the Centroids
#plt.scatter(f1, f2, c='#050505', s=20)
```

```
#plt.scatter(C_x, C_y, marker='*', s=200, c='g')
#plt.xlabel('IGD')
#plt.ylabel('PMG')
#plt.title('Internet Gaming & Mobile Gaming')

#cluster plot
kmeans = KMeans(n_clusters=k)
#Fitting the imput data
kmeans = kmeans.fit(C)
#Getting the cluster labels
labels = kmeans.predict(C)
#Centroid values
centroids = kmeans.cluster_centers_
colors = ['r', 'g', 'y', 'm', 'o', 'w']
fig2 = plt.figure()
kg = fig2.add_subplot(111)

for i in range(k):
    points = np.array([C[j] for j in range(len(C)) if labels[j] == i])
    kg.scatter(points[:,0], points[:,1], s=20, cmap='rainbow')


kg.scatter(centroids[:,0], centroids[:,1], marker='*', s=200, c='b')
print('Fianl Centroids')
print('centroids')
plt.xlabel('Smartphone Addiction Scores')
plt.ylabel('Problematic Smartphone Gaming Scores')
plt.title('Number of Clusters={}'.format(k))
```

```
Fianl Centroids
centroids
```

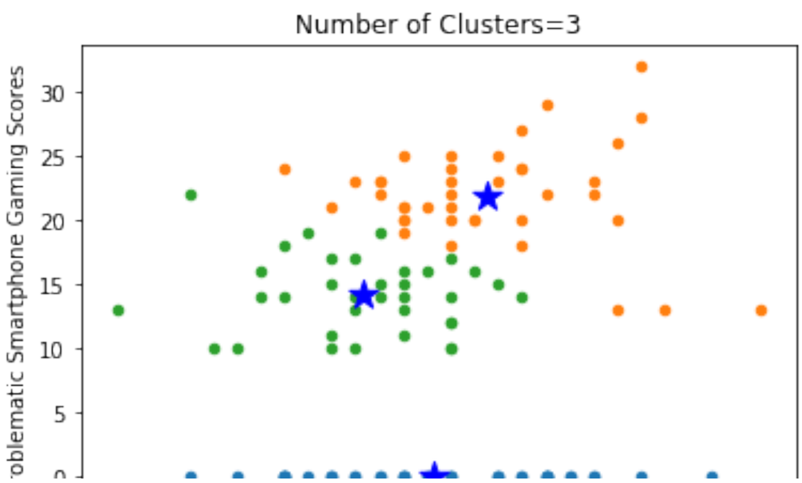
Out[28]: `Text(0.5,1,'Number of Clusters=4')`

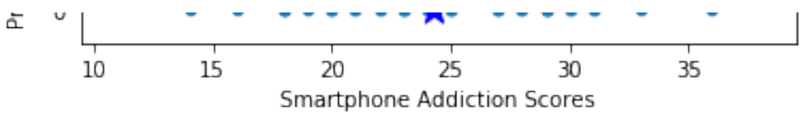


In [47]:
```
##Nuber of clusters
k=3
```

```
##X coordinates of random centroids
#C_x = np.random.randint(0, np.max(X), size=k)
##Y coordinates of random centroids
#C_y = np.random.randint(0, np.max(X), size=k)
#C = np.array(list(zip(C_x, C_y)), dtype=np.float32)

##plotting along with the Centroids
#plt.scatter(f1, f2, c='#050505', s=20)
#plt.scatter(C_x, C_y, marker='*', s=200, c='g')
#plt.xlabel('IGD')
#plt.ylabel('PMG')
#plt.title('Internet Gaming & Mobile Gaming')

#cluster plot
kmeans = KMeans(n_clusters=k)
#Fitting the imput data
kmeans = kmeans.fit(C)
#Getting the cluster labels
labels = kmeans.predict(C)
#Centroid values
centroids = kmeans.cluster_centers_
colors = ['r', 'g', 'y', 'm', 'o', 'w']
fig2 = plt.figure()
kg = fig2.add_subplot(111)

for i in range(k):
    points = np.array([C[j] for j in range(len(C)) if labels[j] == i])
    kg.scatter(points[:,0], points[:,1], s=20, cmap='rainbow')


kg.scatter(centroids[:,0], centroids[:,1], marker='*', s=200, c='b')
print('Fianl Centroids')
print('centroids')
plt.xlabel('Smartphone Addiction Scores')
plt.ylabel('Problematic Smartphone Gaming Scores')
plt.title('Number of Clusters={}'.format(k))
plt.savefig('C_clustering.png')
```

```
Fianl Centroids
centroids
```

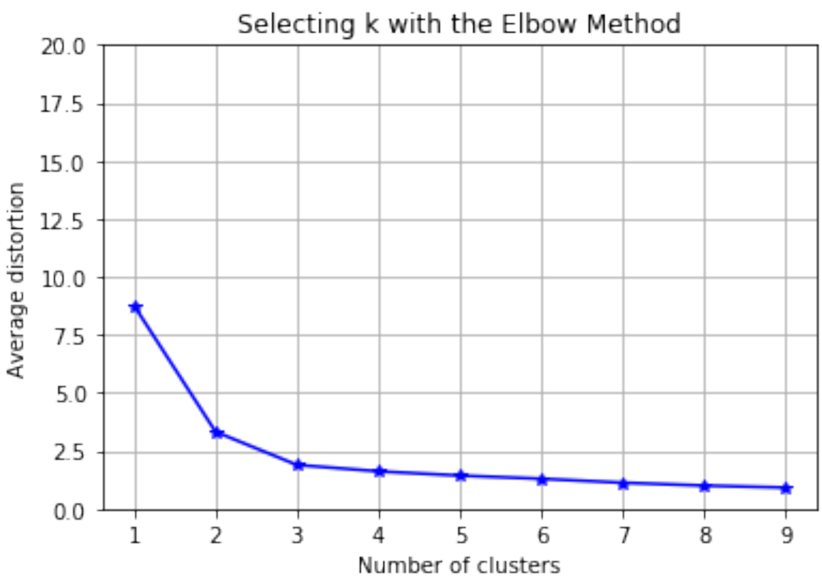




In [29]:
```
#Internet Gaming Disorder & Problematic Smartphone Gaming
D = np.array(list(zip(f33,f34)))
```

In [30]:
```
#elbow plot
##### cluster data into k=1....10 clusters #####
K_range=range(1,10)
distortions = []

for i in K_range:
    kmModel = KMeans(n_clusters=i)
    kmModel.fit(D)
    distortions.append(sum(np.min(cdist(D, kmModel.cluster_centers_, '

fig1 = plt.figure()
ex = fig1.add_subplot(111)
ex.plot(K_range, distortions, 'b*-')
#mark the elboq
#ex.plot(K_range[2], distortions[2], marker='o', markersixe=12, marker

plt.grid(True)
#plt.xlim([0,20])
plt.ylim([0, 20])
plt.xlabel('Number of clusters')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
```

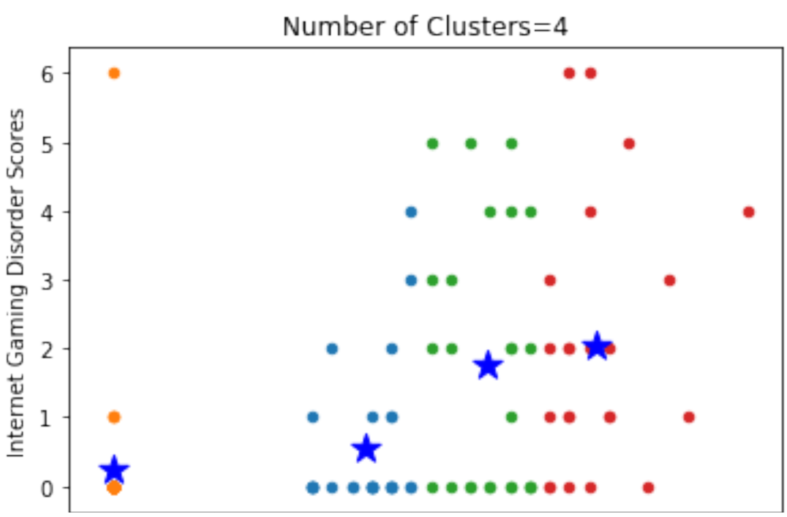
Out[30]: Text(0.5,1,'Selecting k with the Elbow Method')

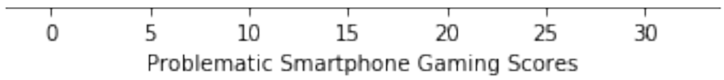


In [44]:
```
##Nuber of clusters
k=4

##X coordinates of random centroids
```

```
#C_x = np.random.randint(0, np.max(X), size=k)
##Y coordinates of random centroids
#C_y = np.random.randint(0, np.max(X), size=k)
#C = np.array(list(zip(C_x, C_y)), dtype=np.float32)

##plotting along with the Centroids
#plt.scatter(f1, f2, c='#050505', s=20)
#plt.scatter(C_x, C_y, marker='*', s=200, c='g')
#plt.xlabel('IGD')
#plt.ylabel('PMG')
#plt.title('Internet Gaming & Mobile Gaming')

#cluster plot
kmeans = KMeans(n_clusters=k)
#Fitting the imput data
kmeans = kmeans.fit(D)
#Getting the cluster labels
labels = kmeans.predict(D)
#Centroid values
centroids = kmeans.cluster_centers_
colors = ['r', 'g', 'y', 'm', 'o', 'w']
fig2 = plt.figure()
kg = fig2.add_subplot(111)

for i in range(k):
    points = np.array([D[j] for j in range(len(D)) if labels[j] == i])
    kg.scatter(points[:,0], points[:,1], s=20, cmap='rainbow')


kg.scatter(centroids[:,0], centroids[:,1], marker='*', s=200, c='b')
print('Fianl Centroids')
print('centroids')
plt.xlabel('Problematic Smartphone Gaming Scores')
plt.ylabel('Internet Gaming Disorder Scores')
plt.title('Number of Clusters={}'.format(k))
plt.savefig('D_clustering.png')
```

```
Fianl Centroids
centroids
```

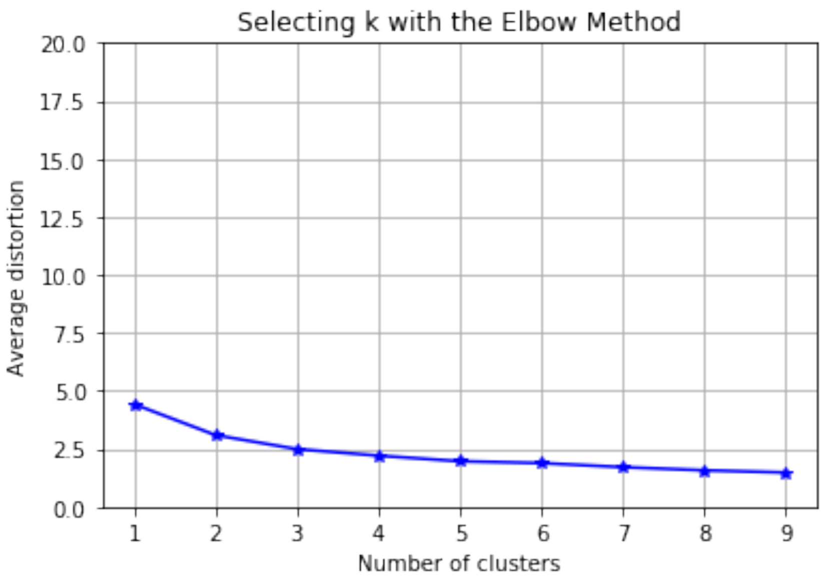




In [32]:
```
#BAS fun seeking & smartphone addiction
E = np.array(list(zip(f24,f31)))
```

In [33]:
```
#elbow plot
##### cluster data into k=1....10 clusters #####
K_range=range(1,10)
distortions = []

for i in K_range:
    kmModel = KMeans(n_clusters=i)
    kmModel.fit(E)
    distortions.append(sum(np.min(cdist(E, kmModel.cluster_centers_, '

fig1 = plt.figure()
ex = fig1.add_subplot(111)
ex.plot(K_range, distortions, 'b*-')
#mark the elboq
#ex.plot(K_range[2], distortions[2], marker='o', markersixe=12, marker

plt.grid(True)
#plt.xlim([0,20])
plt.ylim([0, 20])
plt.xlabel('Number of clusters')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
```

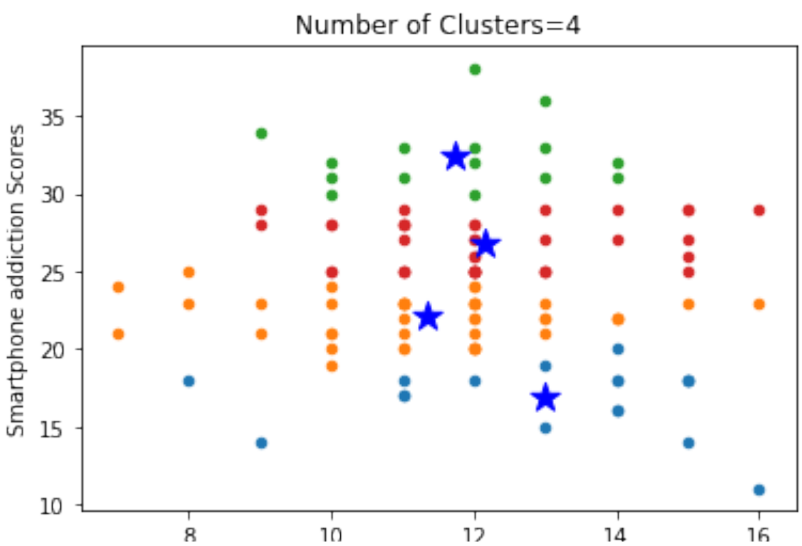
Out[33]: Text(0.5,1,'Selecting k with the Elbow Method')



In [45]:
```
##Nuber of clusters
k=4

##X coordinates of random centroids
#C_x = np.random.randint(0, np.max(X), size=k)
```

```
#C_x = np.random.randint(0, np.max(X), size=k)
##Y coordinates of random centroids
#C_y = np.random.randint(0, np.max(X), size=k)
#C = np.array(list(zip(C_x, C_y)), dtype=np.float32)

##plotting along with the Centroids
#plt.scatter(f1, f2, c='#050505', s=20)
#plt.scatter(C_x, C_y, marker='*', s=200, c='g')
#plt.xlabel('IGD')
#plt.ylabel('PMG')
#plt.title('Internet Gaming & Mobile Gaming')

#cluster plot
kmeans = KMeans(n_clusters=k)
#Fitting the imput data
kmeans = kmeans.fit(E)
#Getting the cluster labels
labels = kmeans.predict(E)
#Centroid values
centroids = kmeans.cluster_centers_
colors = ['r', 'g', 'y', 'm', 'o', 'w']
fig2 = plt.figure()
kg = fig2.add_subplot(111)

for i in range(k):
    points = np.array([E[j] for j in range(len(E)) if labels[j] == i])
    kg.scatter(points[:,0], points[:,1], s=20, cmap='rainbow')


kg.scatter(centroids[:,0], centroids[:,1], marker='*', s=200, c='b')
print('Fianl Centroids')
print('centroids')
plt.xlabel('BAS Fun Seeking')
plt.ylabel('Smartphone addiction Scores')
plt.title('Number of Clusters={}'.format(k))
plt.savefig('E_clustering.png')
```

```
Fianl Centroids
centroids
```



BAS Fun Seeking

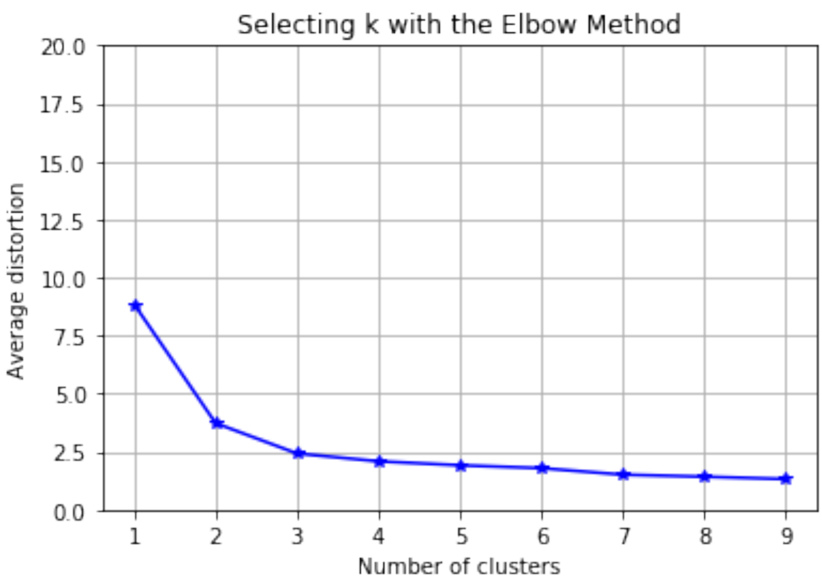
In [36]:

```
#BAS fun seeking & Problematic smartphone Gaming
F = np.array(list(zip(f24, f33)))
```

In [37]:

```
bow plot
## cluster data into k=1....10 clusters #####
ange=range(1,10)
tortions = []

 i in K_range:
 kmModel = KMeans(n_clusters=i)
 kmModel.fit(F)
 distortions.append(sum(np.min(cdist(F, kmModel.cluster_centers_, 'eucl

1 = plt.figure()
= fig1.add_subplot(111)
plot(K_range, distortions, 'b*-')
rk the elboq
.plot(K_range[2], distortions[2], marker='o', markersixe=12, markeredge

.grid(True)
t.xlim([0,20])
.ylim([0, 20])
.xlabel('Number of clusters')
.ylabel('Average distortion')
.title('Selecting k with the Elbow Method')
```

Out[37]: Text(0.5,1,'Selecting k with the Elbow Method')

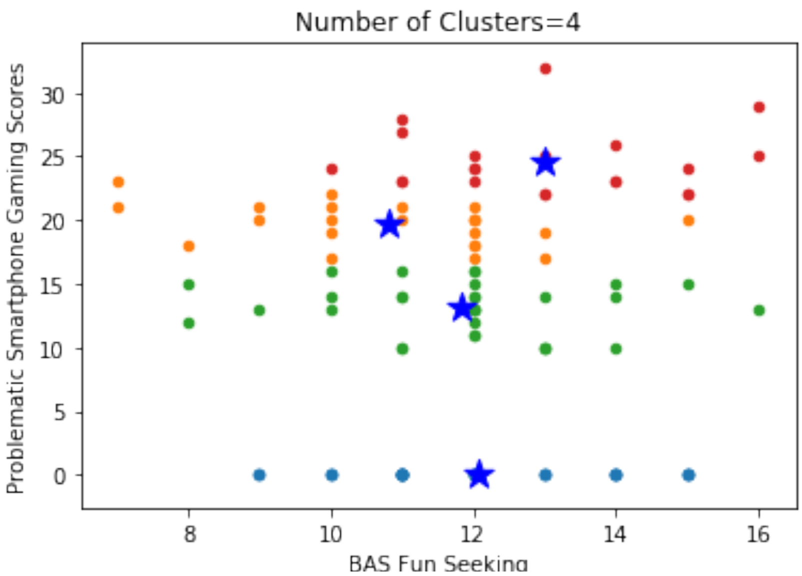


In [46]:

```
##Nuber of clusters
k=4

##X coordinates of random centroids
#C_x = np.random.randint(0, np.max(X), size=k)
```

```
##Y coordinates of random centroids
#C_y = np.random.randint(0, np.max(X), size=k)
#C = np.array(list(zip(C_x, C_y)), dtype=np.float32)

##plotting along with the Centroids
#plt.scatter(f1, f2, c='#050505', s=20)
#plt.scatter(C_x, C_y, marker='*', s=200, c='g')
#plt.xlabel('IGD')
#plt.ylabel('PMG')
#plt.title('Internet Gaming & Mobile Gaming')

#cluster plot
kmeans = KMeans(n_clusters=k)
#Fitting the imput data
kmeans = kmeans.fit(F)
#Getting the cluster labels
labels = kmeans.predict(F)
#Centroid values
centroids = kmeans.cluster_centers_
colors = ['r', 'g', 'y', 'm', 'o', 'w']
fig2 = plt.figure()
kg = fig2.add_subplot(111)

for i in range(k):
    points = np.array([F[j] for j in range(len(F)) if labels[j] == i])
    kg.scatter(points[:,0], points[:,1], s=20, cmap='rainbow')


kg.scatter(centroids[:,0], centroids[:,1], marker='*', s=200, c='b')
print('Fianl Centroids')
print('centroids')
plt.xlabel('BAS Fun Seeking')
plt.ylabel('Problematic Smartphone Gaming Scores')
plt.title('Number of Clusters={}'.format(k))
plt.savefig('F_clustering.png')
```

```
Fianl Centroids
centroids
```



```
In [ ]:
```