Clustering - K Means Clustering

Clustering

K Means Clustering

K ? Elbow method

```
    Silhouette method
```

Implement K means on a simulated dataset

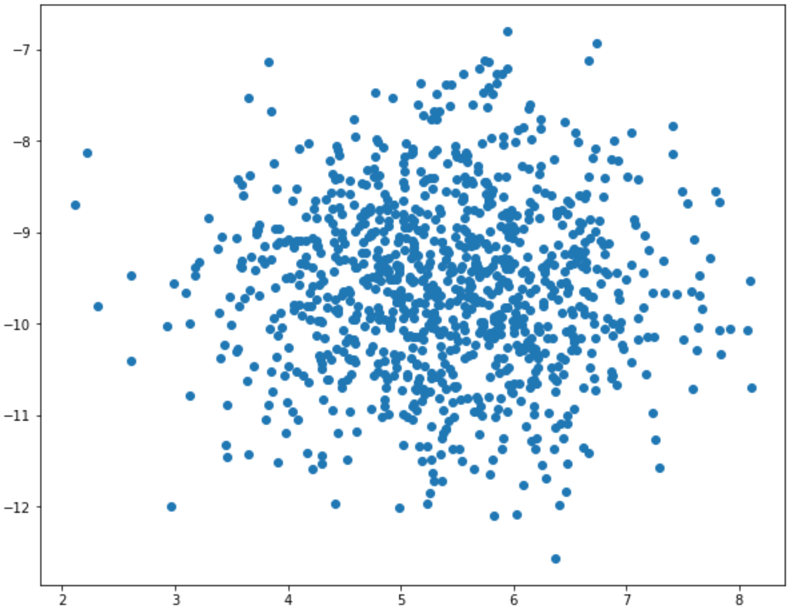
Also, implement k means on a standard dataset

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
```

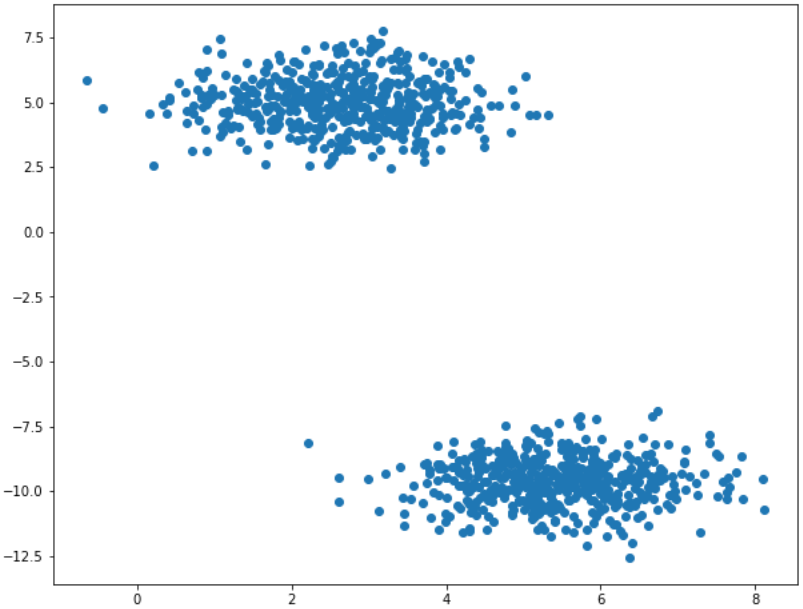
In [3]:

```
plt.figure(figsize=(10,8))

X,y=make_blobs(n_samples=1000, n_features=2, centers=1, random_state=10)
plt.scatter(X[:,0],X[:,1]);
```



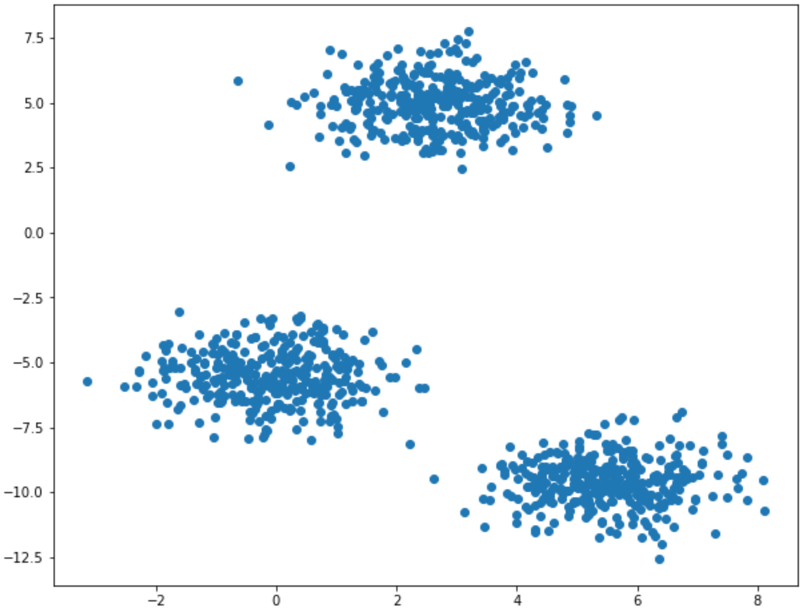
In [5]:

```
plt.figure(figsize=(10,8))

X,y=make_blobs(n_samples=1000, n_features=2,centers=1,random_state=10)
plt.scatter(X[:,0],X[:,1]);
```

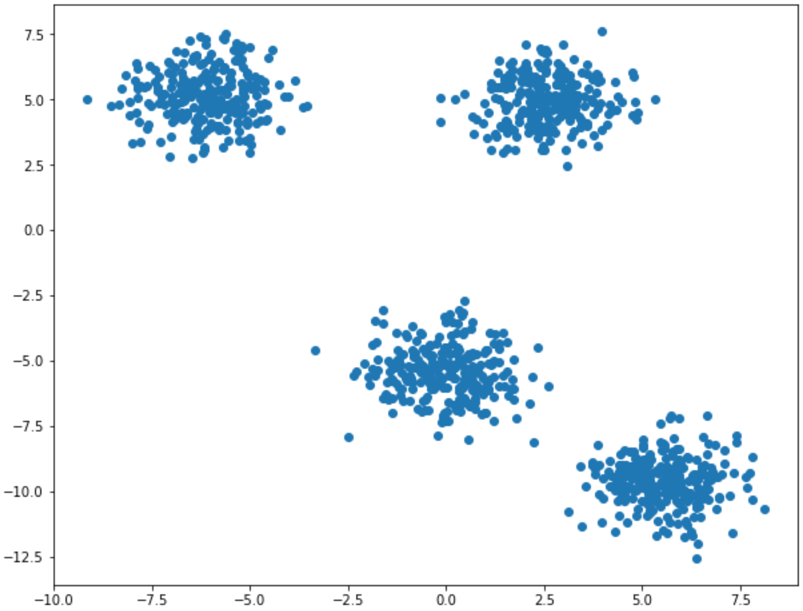


In [6]:

```
plt.figure(figsize=(10,8))

X,y=make_blobs(n_samples=1000, n_features=2,centers=3,random_state=10)
plt.scatter(X[:,0],X[:,1]);
```



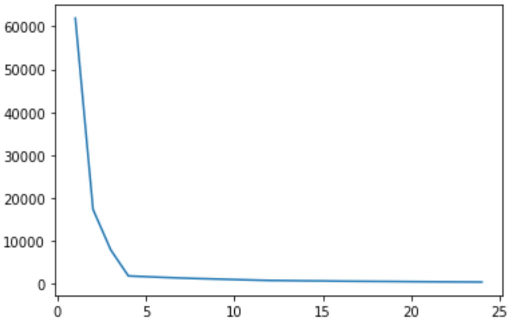
In [9]:
```
plt.figure(figsize=(10,8))

X,y=make_blobs(n_samples=1000, n_features=2,centers=4,random_state=10)
plt.scatter(X[:,0],X[:,1]);
```



Implementing K Means on this dataset

Elbow method for finding k

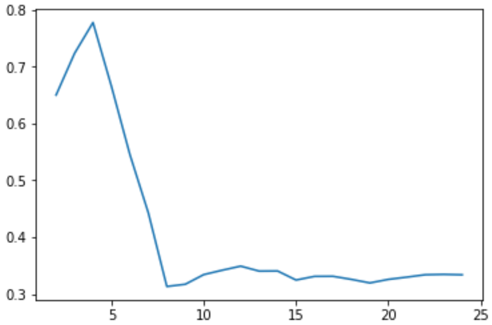
In [10]:
```
from sklearn.cluster import KMeans

SSD=[]
for k in range(1,25):
    kmeans=KMeans(n_clusters=k,random_state=10)
    kmeans.fit(X)
    SSD.append(kmeans.inertia_)
plt.plot(range(1,25),SSD);
```



From the graph, the best value of k= 4

In [11]:

```
### Silhouette method
from sklearn.metrics import silhouette_score
SS=[]
for k in range(2,25):
    kmeans=KMeans(n_clusters=k, random_state=10)
    kmeans.fit(X)
    SS.append(silhouette_score(X,kmeans.predict(X)))
plt.plot(range(2,25),SS);
```



In [12]:

```
The best value of =4, the highest peak
```

```
  Input In [12]
    The best value of =4, the highest peak
        ^
SyntaxError: invalid syntax
```

Building the best model

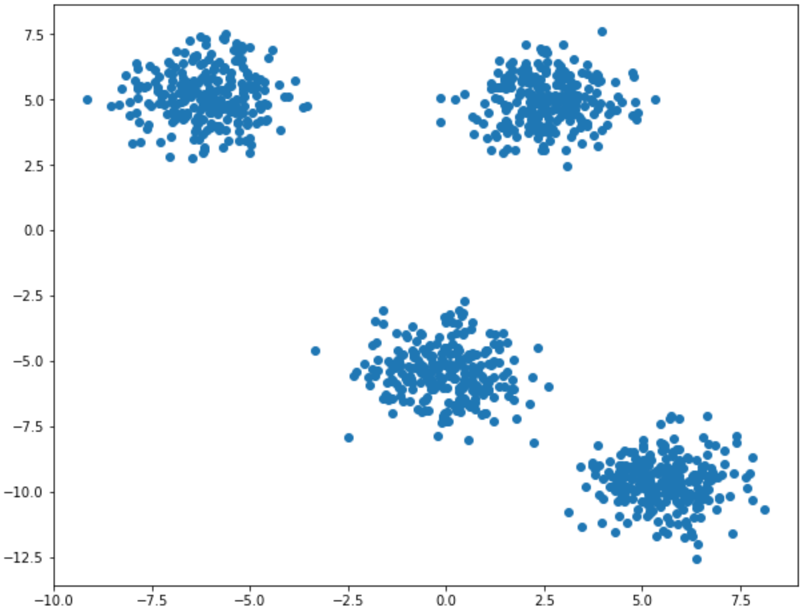
In [14]:
```
k_best=KMeans(n_clusters=4,random_state=10)
k_best.fit(X)
clust_pred=k_best.predict(X)
clust_pred
```

Out[14]:
```
array([1, 2, 2, 1, 3, 3, 0, 3, 1, 2, 3, 1, 2, 0, 2, 0, 2, 2, 1, 0, 0, 3,
       0, 3, 2, 2, 3, 2, 1, 1, 1, 1, 3, 0, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3,
       3, 3, 0, 2, 3, 3, 1, 3, 1, 3, 0, 1, 2, 2, 1, 3, 1, 3, 0, 2, 2, 2,
       1, 2, 1, 3, 1, 0, 3, 2, 0, 1, 0, 2, 2, 3, 1, 1, 1, 1, 2, 3, 3, 0,
       0, 2, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 0, 3, 2, 3, 0, 1,
       0, 3, 3, 3, 0, 2, 3, 3, 3, 1, 0, 3, 3, 0, 2, 2, 3, 1, 1, 2, 0, 3,
       2, 3, 0, 0, 2, 3, 2, 0, 2, 3, 2, 2, 0, 2, 3, 1, 0, 1, 3, 2, 3, 2,
       2, 3, 0, 3, 0, 3, 0, 2, 0, 3, 0, 3, 1, 2, 3, 2, 0, 2, 2, 1, 0, 0,
       3, 2, 2, 1, 3, 3, 3, 1, 1, 3, 2, 1, 2, 2, 3, 2, 2, 3, 2, 3, 2, 0,
       1, 1, 0, 1, 2, 3, 0, 2, 3, 1, 2, 2, 1, 1, 0, 3, 3, 3, 2, 2, 1, 3,
       3, 2, 1, 1, 3, 1, 3, 0, 2, 1, 1, 3, 0, 3, 3, 1, 2, 3, 1, 0, 2, 1,
       2, 0, 1, 0, 0, 2, 2, 0, 0, 3, 1, 0, 2, 3, 0, 1, 2, 3, 0, 2, 0, 0,
       1, 0, 2, 2, 1, 3, 1, 2, 1, 1, 2, 3, 1, 1, 1, 0, 0, 3, 0, 1, 0, 0,
       0, 3, 0, 0, 1, 0, 2, 2, 3, 3, 3, 0, 3, 2, 1, 1, 2, 2, 1, 3, 1, 1,
       1, 3, 2, 1, 3, 0, 1, 3, 1, 1, 3, 3, 1, 3, 2, 2, 0, 3, 1, 2, 1, 2,
       2, 0, 2, 0, 3, 1, 0, 1, 0, 1, 1, 2, 1, 0, 2, 0, 0, 1, 0, 3, 2, 1,
       0, 1, 1, 1, 2, 3, 2, 2, 1, 0, 3, 2, 2, 0, 3, 1, 3, 2, 2, 0, 0, 3,
       2, 1, 3, 3, 2, 0, 1, 2, 1, 1, 3, 2, 2, 0, 0, 1, 3, 1, 1, 2, 3, 0,
       3, 2, 3, 0, 2, 0, 0, 3, 3, 2, 2, 2, 1, 3, 3, 2, 3, 1, 1, 3, 2, 2,
       1, 1, 1, 1, 0, 2, 1, 2, 1, 2, 1, 0, 3, 3, 2, 3, 0, 3, 0, 1, 0, 3,
       3, 3, 0, 1, 3, 1, 1, 0, 0, 0, 1, 0, 3, 0, 0, 0, 3, 1, 0, 2, 0, 1,
       0, 1, 2, 3, 1, 0, 2, 2, 0, 2, 0, 3, 2, 1, 2, 2, 0, 1, 1, 3, 1, 1,
       3, 3, 2, 1, 2, 2, 0, 0, 1, 2, 3, 2, 0, 3, 3, 3, 0, 3, 0, 0, 1, 3,
       2, 3, 1, 1, 1, 1, 2, 0, 0, 0, 0, 1, 1, 2, 0, 2, 2, 0, 2, 0, 3, 3,
       0, 1, 0, 2, 2, 1, 2, 0, 3, 0, 3, 2, 3, 2, 2, 2, 3, 3, 3, 2, 3, 1,
       2, 0, 3, 3, 1, 2, 2, 0, 3, 0, 0, 2, 0, 0, 3, 1, 1, 2, 1, 3, 0, 0,
       0, 3, 3, 2, 1, 1, 2, 1, 2, 2, 2, 3, 3, 0, 2, 0, 1, 0, 2, 0, 1, 1,
       0, 0, 3, 0, 0, 0, 0, 2, 2, 3, 3, 2, 0, 2, 1, 3, 3, 1, 1, 2, 3, 2,
       0, 2, 3, 2, 0, 0, 3, 1, 2, 2, 1, 0, 3, 0, 3, 2, 1, 0, 0, 0, 2, 3,
       3, 3, 0, 0, 3, 1, 3, 0, 0, 0, 0, 0, 1, 1, 3, 2, 3, 2, 3, 3, 0, 3,
       3, 0, 3, 1, 0, 1, 0, 1, 2, 2, 3, 2, 0, 2, 2, 2, 2, 0, 1, 3, 0, 3,
       3, 1, 2, 3, 2, 3, 1, 2, 2, 3, 1, 0, 3, 2, 0, 0, 1, 1, 1, 1, 1, 1,
       3, 1, 2, 0, 2, 1, 0, 2, 2, 2, 2, 0, 3, 1, 2, 0, 0, 2, 0, 2, 0, 2,
       0, 0, 3, 3, 0, 1, 0, 2, 1, 3, 1, 3, 1, 1, 1, 2, 0, 1, 0, 0, 3, 0,
       1, 0, 3, 2, 2, 1, 0, 1, 3, 1, 3, 0, 1, 1, 3, 1, 0, 3, 3, 0, 3, 1,
       2, 3, 1, 2, 1, 1, 3, 0, 2, 3, 1, 3, 0, 0, 2, 3, 2, 0, 2, 3, 3, 3,
       3, 1, 2, 1, 2, 0, 2, 0, 2, 0, 3, 3, 3, 2, 1, 0, 2, 0, 1, 2, 3, 0,
       1, 3, 0, 1, 1, 1, 3, 2, 3, 1, 3, 0, 3, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 2, 2, 1, 3, 1, 0, 1, 3, 3, 2, 0, 2, 1, 3, 2, 1, 2, 2, 0, 1, 1,
       3, 1, 1, 3, 1, 2, 0, 1, 3, 2, 3, 2, 3, 1, 3, 3, 0, 0, 3, 2, 1, 2,
       1, 1, 0, 3, 2, 0, 0, 1, 0, 3, 1, 2, 2, 3, 0, 3, 0, 1, 1, 1, 0, 0,
       2, 2, 0, 0, 2, 3, 0, 1, 0, 3, 2, 3, 1, 1, 3, 0, 0, 0, 3, 1, 2, 3,
       0, 0, 1, 0, 0, 0, 1, 2, 2, 2, 1, 3, 1, 3, 3, 0, 2, 0, 1, 1, 1, 2,
       3, 3, 1, 1, 3, 0, 2, 0, 2, 2, 2, 1, 3, 3, 0, 2, 0, 0, 3, 2, 0, 0,
       2, 2, 0, 1, 0, 2, 3, 1, 1, 0, 2, 1, 1, 3, 3, 3, 1, 3, 0, 1, 0, 0,
       3, 3, 3, 0, 2, 3, 3, 2, 1, 0], dtype=int32)
```

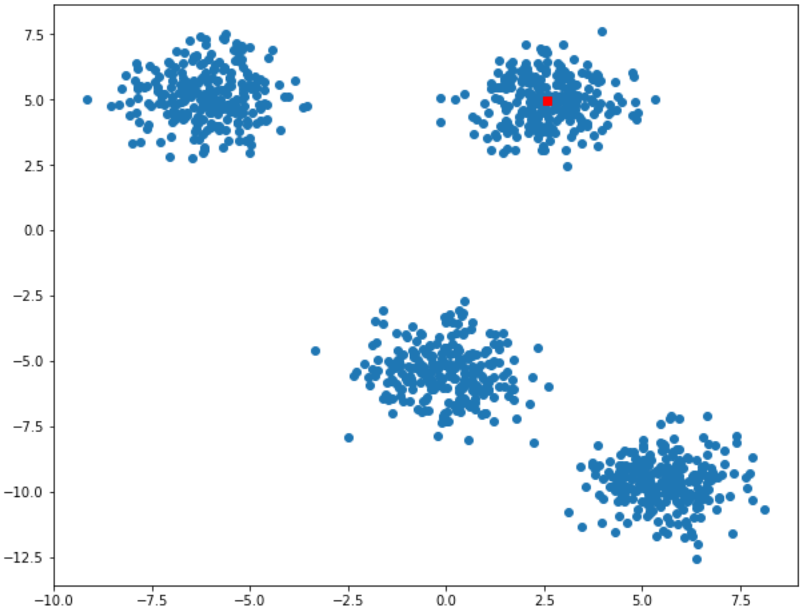
In [15]:
```
k_best.cluster_centers_
```

Out[15]:
```
array([[ 2.57427374,  4.9551547 ],
       [ 5.54690135, -9.62123904],
       [-6.10307996,  5.14422118],
       [-0.03749354, -5.43011018]])
```

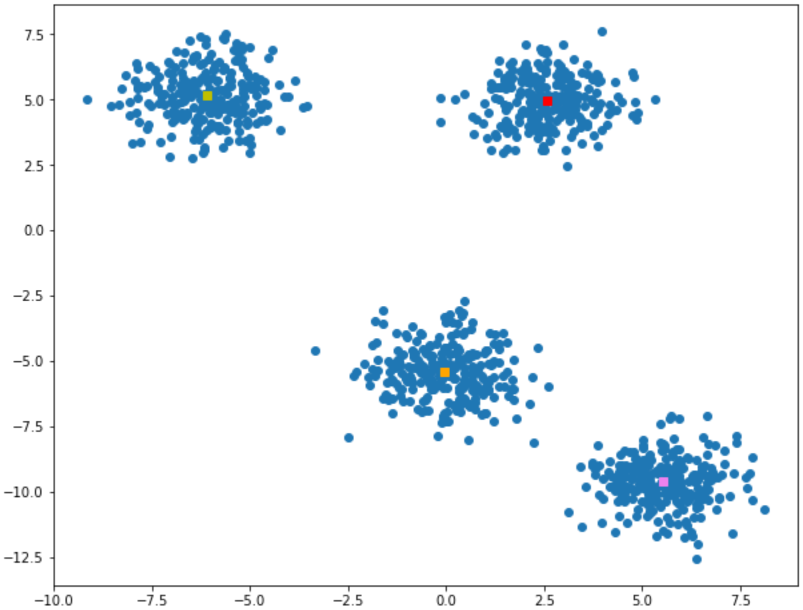
In [16]:
```
plt.figure(figsize=(10,8))
plt.scatter(X[:,0],X[:,1]);
```



In [17]:
```
plt.figure(figsize=(10,8))
plt.scatter(X[:,0],X[:,1])
plt.plot(2.57427374,  4.9551547, c='r',marker='s');
```



In [20]:
```
plt.figure(figsize=(10,8))
plt.scatter(X[:,0],X[:,1])
plt.plot(2.57427374,  4.9551547, c='r',marker='s')
plt.plot(5.54690135, -9.62123904,c='violet',marker='s')
plt.plot(-6.10307996,  5.14422118,c='y',marker='s')
plt.plot(-0.03749354, -5.43011018,c='orange',marker='s');
```



K Means for a standard dataset

In [21]:
```
from sklearn.datasets import load_iris
iris=load_iris()
iris
```

Out[21]:
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
        [5.7, 3.8, 1.7, 0.3],
```

In [22]:
```
# Converting to DF

X= pd.DataFrame(iris['data'],columns=['SL','SW','PL','PW'])
X
```

Out[22]:

	SL	SW	PL	PW
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

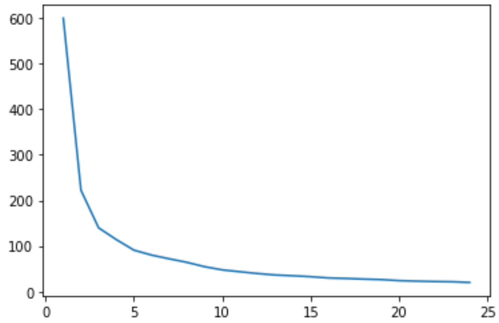
150 rows × 4 columns

In [23]:
```
# Standardisation

from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
X_scaled
```

Out[23]:
```
array([[-9.00681170e-01,  1.01900435e+00, -1.34022653e+00,
        -1.31544430e+00],
       [-1.14301691e+00, -1.31979479e-01, -1.34022653e+00,
        -1.31544430e+00],
       [-1.38535265e+00,  3.28414053e-01, -1.39706395e+00,
        -1.31544430e+00],
       [-1.50652052e+00,  9.82172869e-02, -1.28338910e+00,
        -1.31544430e+00],
       [-1.02184904e+00,  1.24920112e+00, -1.34022653e+00,
        -1.31544430e+00],
       [-5.37177559e-01,  1.93979142e+00, -1.16971425e+00,
        -1.05217993e+00],
       [-1.50652052e+00,  7.88807586e-01, -1.34022653e+00,
        -1.18381211e+00],
       [-1.02184904e+00,  7.88807586e-01, -1.28338910e+00,
        -1.31544430e+00],
       [-1.74885626e+00, -3.62176246e-01, -1.34022653e+00,
        -1.31544430e+00],
       [-1.14301691e+00,  9.82172869e-02, -1.28338910e+00,
        -1.44707648e+00],
```

Finding using elbow

In [28]:
```
SSD=[]
for k in range(1,25):
    kmeans=KMeans(n_clusters=k,random_state=10)
    kmeans.fit(X_scaled)
    SSD.append(kmeans.inertia_)
plt.plot(range(1,25),SSD);
# plt.xlim([0,7]);
```



```
Choose the best value of k to be 3.
```

In [29]:
```
k_final=KMeans(n_clusters=3,random_state=10)
k_final.fit(X_scaled)
clusters=k_final.predict(X_scaled)
clusters
```

Out[29]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0], dtype=int32)
```

In [30]:
```
# The cluster centroids

k_final.cluster_centers_
```

Out[30]:
```
array([[-0.05021989, -0.88337647,  0.34773781,  0.2815273 ],
       [ 1.13597027,  0.08842168,  0.99615451,  1.01752612],
       [-1.01457897,  0.85326268, -1.30498732, -1.25489349]])
```

In [ ]: