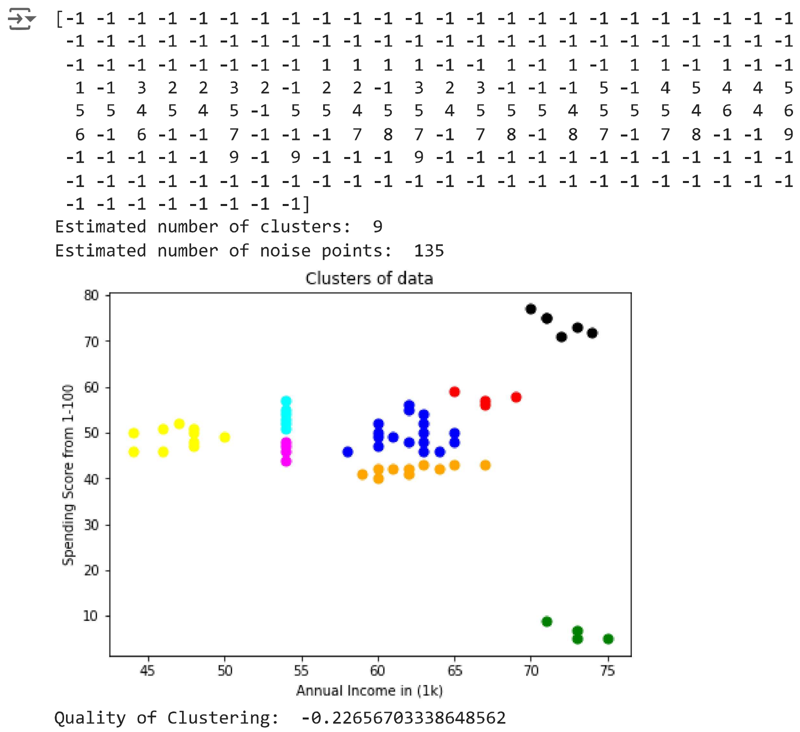
DBSCAN

CODE

```
import numpy
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
from sklearn.cluster import DBSCAN

def dbscan(D, eps, MinPts):
    labels = [0]*len(D)
    C = 0
    for P in range(0, len(D)):
        if not (labels[P] == 0):
           continue
        NeighborPts = region_query(D, P, eps)
        else:
           C += 1
           grow_cluster(D, labels, P, NeighborPts, C, eps, MinPts)
    return labels

def grow_cluster(D, labels, P, NeighborPts, C, eps, MinPts):
    labels[P] = C
    i = 0
    while i < len(NeighborPts):
        Pn = NeighborPts[i]
        if labels[Pn] == -1:
           labels[Pn] = C
        elif labels[Pn] == 0:
            labels[Pn] = C
            PnNeighborPts = region_query(D, Pn, eps)
            if len(PnNeighborPts) >= MinPts:
                NeighborPts = NeighborPts + PnNeighborPts
        i += 1

def region_query(D, P, eps):
    neighbors = []
    for Pn in range(0, len(D)):
        if numpy.linalg.norm(D[P] - D[Pn]) < eps:
           neighbors.append(Pn)
    return neighbors

data = pd.read_csv("/content/drive/MyDrive/JISNIT/Courses/ML/LectureNotes/data/Mall_Customers.csv")
#f1 = data['Age'].values
f2 = data['Annual Income (k$)'].values
f3 = data['Spending Score (1-100)'].values
X = numpy.array(list(zip(f2, f3)))
cluster_labels = numpy.array(dbscan(X, 3, 4))
#db = DBSCAN(eps = 3, min_samples = 4).fit(X)
#cluster_labels = db.labels_
print(cluster_labels)

n_clusters_ = len(set(cluster_labels)) - (1 if -1 in cluster_labels else 0)
n_noise_ = list(cluster_labels).count(-1)

print("Estimated number of clusters: " , n_clusters_)
print("Estimated number of noise points: " , n_noise_)

plt.figure(figsize=(7,5))
plt.scatter(X[cluster_labels == 0, 0], X[cluster_labels == 0, 1], s = 50, c = 'pink')
plt.scatter(X[cluster_labels == 1, 0], X[cluster_labels == 1, 1], s = 50, c = 'yellow')
plt.scatter(X[cluster_labels == 2, 0], X[cluster_labels == 2, 1], s = 50, c = 'cyan')
plt.scatter(X[cluster_labels == 3, 0], X[cluster_labels == 3, 1], s = 50, c = 'magenta')
plt.scatter(X[cluster_labels == 4, 0], X[cluster_labels == 4, 1], s = 50, c = 'orange')
plt.scatter(X[cluster_labels == 5, 0], X[cluster_labels == 5, 1], s = 50, c = 'blue')
plt.scatter(X[cluster_labels == 6, 0], X[cluster_labels == 6, 1], s = 50, c = 'red')
plt.scatter(X[cluster_labels == 7, 0], X[cluster_labels == 7, 1], s = 50, c = 'black')
plt.scatter(X[cluster_labels == 8, 0], X[cluster_labels == 8, 1], s = 50, c = 'green')
plt.xlabel('Annual Income in (1k)')
plt.ylabel('Spending Score from 1-100')
plt.title('Clusters of data')
plt.show()
if(n_clusters_ > 1):
  sil = silhouette_score(X, cluster_labels,
                         metric='euclidean',
                         sample_size = len(data))
  print("Quality of Clustering: ", sil)
```

OUTPUT



```
[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1 -1 -1  1 -1  1 -1  1  1 -1  1 -1 -1
  1 -1  3  2  2  3  2 -1  2  2 -1  3  2  3 -1 -1 -1  5 -1  4  5  4  4  5
  5  5  4  5  4  5 -1  5  5  4  5  5  4  5  5  5  4  5  5  5  4  6  4  6
  6 -1  6 -1 -1  7 -1 -1 -1  7  8  7 -1  7  8 -1  8  7 -1  7  8 -1 -1  9
 -1 -1 -1 -1 -1  9 -1  9 -1 -1 -1  9 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1]
Estimated number of clusters:  9
Estimated number of noise points:  135
```

```
Quality of Clustering:  -0.22656703338648562
```