K mediod Clustering when K=4

In [1]:

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
from math import*

from scipy.spatial.distance import pdist,squareform


```

In [2]:

```
def manhattan_distance(person1, person2):
    distance = 0
    distance += abs(person1 - person2)
    return distance
```

In [3]:

```
Assigned_cluster = pd.DataFrame(np.zeros((196, 1), dtype=int))

```

In [4]:

```
dataset = pd.read_csv('Mall_Customers.csv')#---------reading dataset
customer = dataset.to_numpy()
```

calculating Disimilarity MAtrix below

In [5]:
```
squareform(pdist(customer[:,2:], metric='euclidean'))#---------calculating dissimilarity matrix
```

Out[5]:
```
array([[  0.        ,  42.04759208,  33.03028913, ..., 117.1110584 ,
        124.47489707, 130.15759678],
       [ 42.04759208,   0.        ,  75.01333215, ..., 111.7631424 ,
        137.74614332, 122.34786471],
       [ 33.03028913,  75.01333215,   0.        , ..., 129.87686476,
        122.18428704, 143.77065069],
       ...,
       [117.1110584 , 111.7631424 , 129.87686476, ...,   0.        ,
         57.07013229,  14.35270009],
       [124.47489707, 137.74614332, 122.18428704, ...,  57.07013229,
          0.        ,  65.03076195],
       [130.15759678, 122.34786471, 143.77065069, ...,  14.35270009,
         65.03076195,   0.        ]])
```

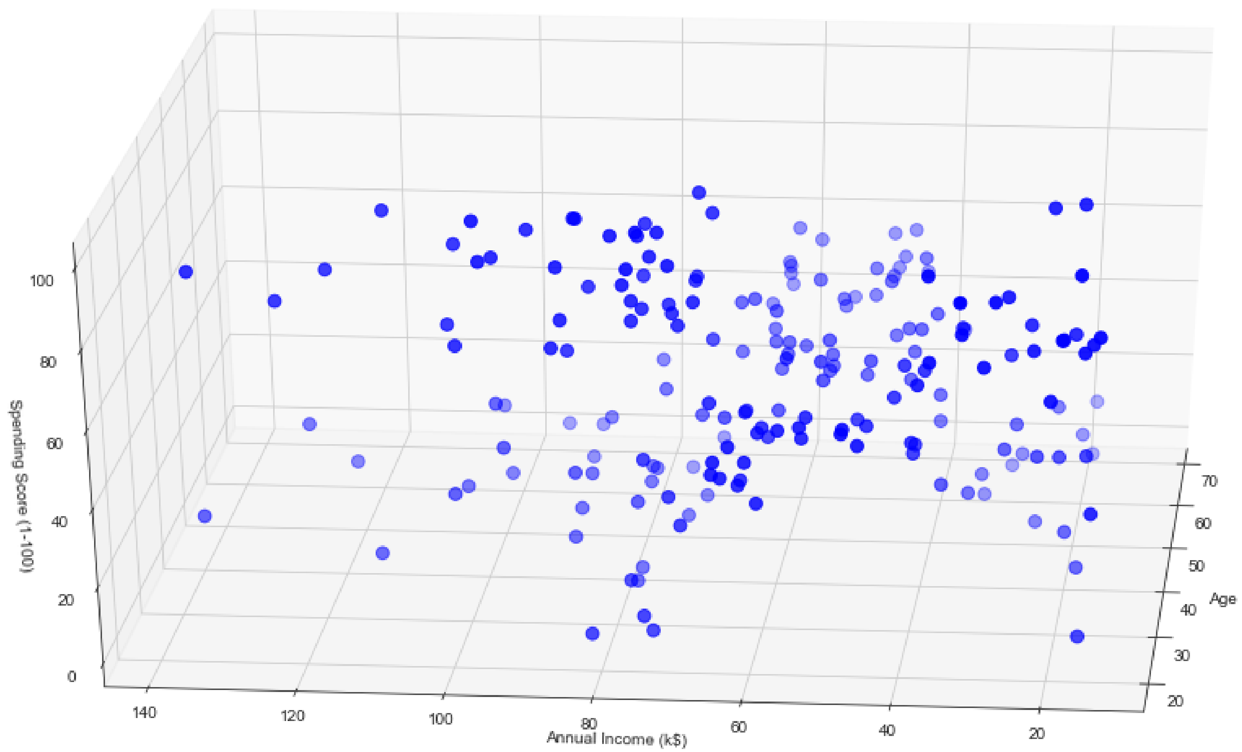
In [ ]:
```


```

In [6]:
```

k = 4 #---------------NUMBER OF CLUSTERS

```

Plotting graph before clustering

In [7]:

```

sns.set_style("white")
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(dataset.Age, dataset["Annual Income (k$)"], dataset["Spending Score (1-100)"], c='blue', s=60)
ax.view_init(30, 185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```



In [ ]:

```python

```

In [8]:

```python
mediod = customer[np.random.choice(customer.shape[0], k, replace=False), :]#---------randomly choosing mediods

```

In [11]:

```python
###**********************                                    #----------delete the mediods value from dataset
customer_new = np.delete(customer, (mediod[0,0] - 1, mediod[1,0] - 1, mediod[2,0] - 1, mediod[3,0] - 1), axis=0)#---
#dataset1 = np.delete(dataset, (mediod[0,0] - 1, mediod[1,0] - 1, mediod[2,0] - 1, mediod[3,0] - 1), axis=0)#-------
dataset_1=dataset.drop(dataset.index[[mediod[0,0] - 1, mediod[1,0] - 1, mediod[2,0] - 1, mediod[3,0] - 1]])

```

In [12]:

```
print(mediod[2, 2])
print(customer.shape)
print(customer_new.shape)
distance_age = 0
distance_income = 0
distance_score = 0
total_distance_cost = np.zeros((len(customer_new),k))
print(total_distance_cost.shape)
print(customer_new.shape)
print(dataset_1.shape)
non_mediod = 0

```

```
50
(200, 5)
(196, 5)
(196, 4)
(196, 5)
(196, 5)
```

Calculating distance between mediod and points and assigning clusters to it

In [13]:

```
###***********************----------------needs to be modified as the value of k changes
#--------------------------comparing when value of k=1
old_cost = 0
total_cost_cluster = 0
difference_cost=0
for l in range(len(customer_new)):

    distance_age_k1 = []
    distance_income_k1 = []
    distance_score_k1 = []
    total_distance_k1 = []
    q=0
    w=0
    e=0
    total=0
    i=0

    for j in range(len(customer_new)):
        q = manhattan_distance(mediod[i,2],customer_new[j,2])
        distance_age_k1.append(q)
        w = manhattan_distance(mediod[i,3],customer_new[j,3])
        distance_income_k1.append(w)
        e = manhattan_distance(mediod[i,4],customer_new[j,4])
        distance_score_k1.append(e)
        total = q+w+e
        total_distance_k1.append(total)

    #--------------------------comparing when value of k=2

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    q=0
    w=0
    e=0
    total=0
    i=1
    for j in range(len(customer_new)):
        q = manhattan_distance(mediod[i,2],customer_new[j,2])
        distance_age_k2.append(q)
```

```
        w = manhattan_distance(mediod[i,3],customer_new[j,3])
        distance_income_k2.append(w)
        e = manhattan_distance(mediod[i,4],customer_new[j,4])
        distance_score_k2.append(e)
        total = q+w+e
        total_distance_k2.append(total)

    #--------------------------comparing when value of k=3
    distance_age_k3 = []
    distance_income_k3 = []
    distance_score_k3 = []
    total_distance_k3 = []
    q=0
    w=0
    e=0
    total=0
    i=2


    for j in range(len(customer_new)):
        q = manhattan_distance(mediod[i,2],customer_new[j,2])
        distance_age_k3.append(q)
        w = manhattan_distance(mediod[i,3],customer_new[j,3])
        distance_income_k3.append(w)
        e = manhattan_distance(mediod[i,4],customer_new[j,4])
        distance_score_k3.append(e)
        total = q+w+e
        total_distance_k3.append(total)


    #--------------------------comparing when value of k=4
    distance_age_k4 = []
    distance_income_k4 = []
    distance_score_k4 = []
    total_distance_k4 = []
    q=0
    w=0
    e=0
    total=0
    i=3
    for j in range(len(customer_new)):
        q = manhattan_distance(mediod[i,2],customer_new[j,2])
```

```
            distance_age_k4.append(q)
            w = manhattan_distance(mediod[i,3],customer_new[j,3])
            distance_income_k4.append(w)
            e = manhattan_distance(mediod[i,4],customer_new[j,4])
            distance_score_k4.append(e)
            total = q+w+e
            total_distance_k4.append(total)





    cost_1 = pd.DataFrame({'Cost_1':total_distance_k1})
    cost_2 = pd.DataFrame({'Cost_2':total_distance_k2})
    cost_3 = pd.DataFrame({'Cost_3':total_distance_k3})
    cost_4 = pd.DataFrame({'Cost_4':total_distance_k4})


    dataset_1 = dataset_1.assign(cost_1=cost_1.values,cost_2=cost_2.values,cost_3=cost_3.values,cost_4=cost_4.value


    customer_new = dataset_1.to_numpy()
    #print(customer_new[0:10, :])#---------------combined dataset which displays cost of all k, the last four colour





    #**************************************
    #----now we will create 4 clusters as the value of k =4 on the basis of total distance or cost


    cluster_1 = []
    cluster_2 = []
    cluster_3 = []
    cluster_4 = []
    q=0
    w=0
    e=0
    r=0
    i=0

    for i in range(len(customer_new)):
```

```
        if(customer_new[i][5] <= customer_new[i][6] and customer_new[i][5] <= customer_new[i][7] and customer_new[i
            #print("five")
            q = customer_new[i]
            cluster_1.append(q)
            dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_1"


        if(customer_new[i][6] <= customer_new[i][5] and customer_new[i][6] <= customer_new[i][7] and customer_new[i
            #print("six")
            w = customer_new[i]
            cluster_2.append(w)
            dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_2"

        if(customer_new[i][7] <= customer_new[i][6] and customer_new[i][7] <= customer_new[i][5] and customer_new[i
           # print("seven")
            e = customer_new[i]
            cluster_3.append(e)
            dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_3"

        if(customer_new[i][8] <= customer_new[i][6] and customer_new[i][8] <= customer_new[i][7] and customer_new[i
           # print("eight")
            r = customer_new[i]
            cluster_4.append(r)
            dataset_1.iloc[i, dataset_1.columns.get_loc('Assigned_cluster')] = "Cluster_4"

    print("lenth is", + len(cluster_4))

    if(len(cluster_4) == 0):
        cluster_4 = cluster_4_old
    if(len(cluster_3) == 0):
        cluster_3 = cluster_3_old
    if(len(cluster_2) == 0):
        cluster_2 = cluster_2_old
    if(len(cluster_1) == 0):
        cluster_1 = cluster_2_old
    total_cost_cluster = 0
    cluster_1 = np.asarray(cluster_1)
    cluster_2 = np.asarray(cluster_2)
    cluster_3 = np.asarray(cluster_3)
    cluster_4 = np.asarray(cluster_4)
    sum_cluster_1 = cluster_1[:,5].sum(axis=0)#----------adding all the particalura coloumn which is cost of partic
```

```
    sum_cluster_2 = cluster_2[:,6].sum(axis=0)
    sum_cluster_3 = cluster_3[:,7].sum(axis=0)#-------adding all the minimum cost of particualr clusters
    sum_cluster_4 = cluster_4[:,8].sum(axis=0)
    total_cost_cluster = sum_cluster_1 + sum_cluster_2 + sum_cluster_3 +sum_cluster_4
    print(total_cost_cluster, l+1)
    #print(mediod)

    difference_cost= total_cost_cluster - old_cost
    if (difference_cost<=0):#------------bad cost
        non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), :]#---------for cluste
        non_mediod = np.delete(non_mediod, np.s_[4:9], axis=1)
        mediod[3]= non_mediod
        old_cost = total_cost_cluster
        #print("difference is", + difference_cost)
    cluster_4_old = cluster_4
    old_cost = total_cost_cluster

    difference_cost= total_cost_cluster - old_cost
    if (difference_cost<=0):#------------bad cost
        non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), :]#---------for cluste
        non_mediod = np.delete(non_mediod, np.s_[4:9], axis=1)
        mediod[2]= non_mediod
        old_cost = total_cost_cluster
        #print("difference is", + difference_cost)
    cluster_3_old = cluster_3
    old_cost = total_cost_cluster


    if (difference_cost<=0):#------------bad cost
        non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), :]#---------for cluste
        non_mediod = np.delete(non_mediod, np.s_[4:9], axis=1)
        mediod[1]= non_mediod
        old_cost = total_cost_cluster
        #print("difference is", + difference_cost)
    cluster_2_old = cluster_2
    old_cost = total_cost_cluster

    if (difference_cost<=0):#------------bad cost
        non_mediod = customer_new[np.random.choice(customer_new.shape[0], 1, replace=False), :]#---------for cluste
        non_mediod = np.delete(non_mediod, np.s_[4:9], axis=1)
        mediod[0]= non_mediod
        old_cost = total_cost_cluster
```

```
        #print("difference is", + difference_cost)
    cluster_1_old = cluster_1
    old_cost = total_cost_cluster

```

```
lenth is 41
7988 1
lenth is 79
13166 2
lenth is 53
13366 3
lenth is 83
13374 4
lenth is 60
12880 5
lenth is 66
14507 6
lenth is 50
14139 7
lenth is 49
14261 8
lenth is 30
15029 9
lenth is 21
```

New coloumn added below for assigned cluster

In [14]:
```
dataset_1
```

Out[14]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	cost_1	cost_2	cost_3	cost_4	Assigned_cluster
0	1	Male	19	15	39	132	70	97	110	Cluster_2
1	2	Male	21	15	81	172	114	137	150	Cluster_2
2	3	Female	20	16	6	97	37	62	75	Cluster_2
3	4	Female	23	16	77	165	111	130	143	Cluster_2
4	5	Female	31	17	40	123	81	84	97	Cluster_2
...	...	...	...	...	...	...	...	...	...	...
195	196	Female	35	120	79	107	169	142	129	Cluster_1
196	197	Female	45	126	28	72	134	107	94	Cluster_1
197	198	Male	32	126	74	105	167	140	127	Cluster_1
198	199	Male	32	137	18	60	122	95	82	Cluster_1
199	200	Male	30	137	83	123	185	162	149	Cluster_1

196 rows × 10 columns

Calculating silhoutte width for each clusters

In [26]:

```
distance_age_k2 = []
distance_income_k2 = []
distance_score_k2 = []
total_distance_k2 = []
distance_sil = []
average_sill = []
total=0
q=0
w=0
e=0
a=0

total=0

for i in range(len(cluster_1)):
    for j in range(len(cluster_1)):#------------calcualting distance
        q = manhattan_distance(cluster_1[i,2],cluster_1[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_1[i,3],cluster_1[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_1[i,4],cluster_1[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total/3)
    a_sil = sum(distance_sil)/len(cluster_1)
    #print(a_sil)
    distance_sil.clear()


    cluster_min=0

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
```

```

    total=0
    n=0#--------------for cluster 1
    for j in range(len(mediod)):#------------calcualting nearest cluster
        q = manhattan_distance(mediod[n,2],mediod[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(mediod[n,3],mediod[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(mediod[n,4],mediod[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total)
    #print(distance_sil)
    index = [i for i in range(0,4)]
    #print(index)
    index=np.asarray(index)
    distance_sil= np.asarray(distance_sil)
    distance_sil.transpose
    #print(distance_sil)
    distance_sil = np.column_stack((index,distance_sil))
    #print(distance_sil[:,1])
    minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
    #print(minval)
    for b in range(len(distance_sil)):
        if (minval == distance_sil[b,1]):
            cluster_min = b
            #print(cluster_min)




    #---------now find the value of b
    if (cluster_min == 0):
            selected_cluster= cluster_1
    if (cluster_min == 1):
            selected_cluster= cluster_2
    if (cluster_min == 2):
            selected_cluster= cluster_3
    if (cluster_min == 3):
            selected_cluster= cluster_4
    if (cluster_min == 4):
            selected_cluster= cluster_5
```

```

    distance_age_k2 = []
    distance_b= []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
    s=0

    total=0

    for j in range(len(selected_cluster)):#------------calcualting distance


        q = manhattan_distance(cluster_1[i,2],selected_cluster[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_1[i,3],selected_cluster[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_1[i,4],selected_cluster[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_b.append(total)
   # print(min(distance_b))
    b = min(distance_b)
    maxi = max(b, a_sil)
    #print("a is", + maxi)
    d = b - a_sil
    sill = ((b - a_sil) / maxi)#------------silhoutte widht formulae
    print("Silhoutte width for each point in cluster 1", +  sill)

    average_sill.append(sill)
s_c1 = sum(average_sill)/len(average_sill)
print("Average silhoutee width for cluster 1", + s_c1)


```

```
Silhoutte width for each point in cluster 1 0.5499999999999999
Silhoutte width for each point in cluster 1 0.48893105629348516
Silhoutte width for each point in cluster 1 0.5996339510409516
Silhoutte width for each point in cluster 1 0.6308243727598568
Silhoutte width for each point in cluster 1 0.6693975081071856
Silhoutte width for each point in cluster 1 0.6170250896057348
Silhoutte width for each point in cluster 1 0.6731182795698925
Silhoutte width for each point in cluster 1 0.5685483870967742
Silhoutte width for each point in cluster 1 0.5504851822711776
Silhoutte width for each point in cluster 1 0.6062048298959987
Silhoutte width for each point in cluster 1 0.7070707070707071
Silhoutte width for each point in cluster 1 0.6251221896383187
Silhoutte width for each point in cluster 1 0.7289359653346174
Silhoutte width for each point in cluster 1 0.6990876507005539
Silhoutte width for each point in cluster 1 0.7337216248506571
Silhoutte width for each point in cluster 1 0.5916601101494886
Silhoutte width for each point in cluster 1 0.7374770521898768
Silhoutte width for each point in cluster 1 0.6575268817204302
Silhoutte width for each point in cluster 1 0.6851851851851852
Silhoutte width for each point in cluster 1 0.720269619643717
Silhoutte width for each point in cluster 1 0.7502986857825568
Silhoutte width for each point in cluster 1 0.7214706902532085
Silhoutte width for each point in cluster 1 0.7089093701996928
Silhoutte width for each point in cluster 1 0.747800586510264
Silhoutte width for each point in cluster 1 0.7643966547192352
Silhoutte width for each point in cluster 1 0.7310577644411103
Silhoutte width for each point in cluster 1 0.7545239968528717
Silhoutte width for each point in cluster 1 0.7237083661159193
Silhoutte width for each point in cluster 1 0.7475678443420378
Silhoutte width for each point in cluster 1 0.7443841158309379
Silhoutte width for each point in cluster 1 0.7557313856766078
Average silhoutee width for cluster 1 0.6770991968983566
```

In [27]:

```
distance_age_k2 = []
distance_income_k2 = []
distance_score_k2 = []
total_distance_k2 = []
distance_sil = []
average_sill = []
total=0
q=0
w=0
e=0
a=0

total=0

for i in range(len(cluster_2)):
    for j in range(len(cluster_2)):#------------calcualting distance
        q = manhattan_distance(cluster_2[i,2],cluster_2[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_2[i,3],cluster_2[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_2[i,4],cluster_2[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total/3)
    a_sil = sum(distance_sil)/len(cluster_2)
    #print(a_sil)
    distance_sil.clear()



    cluster_min=0

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
```

```
    total=0
    n=1#--------------for cluster 2
    for j in range(len(mediod)):#------------calcualting nearest cluster
        q = manhattan_distance(mediod[n,2],mediod[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(mediod[n,3],mediod[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(mediod[n,4],mediod[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total)
    #print(distance_sil)
    index = [i for i in range(0,4)]
    #print(index)
    index=np.asarray(index)
    distance_sil= np.asarray(distance_sil)
    distance_sil.transpose
    #print(distance_sil)
    distance_sil = np.column_stack((index,distance_sil))
    #print(distance_sil[:,1])
    minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
    #print(minval)
    for b in range(len(distance_sil)):
        if (minval == distance_sil[b,1]):
            cluster_min = b
            #print(cluster_min)




    #---------now find the value of b
    if (cluster_min == 0):
            selected_cluster= cluster_1
    if (cluster_min == 1):
            selected_cluster= cluster_2
    if (cluster_min == 2):
            selected_cluster= cluster_3
    if (cluster_min == 3):
            selected_cluster= cluster_4
    if (cluster_min == 4):
            selected_cluster= cluster_5
```

```

    distance_age_k2 = []
    distance_b= []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
    s=0

    total=0

    for j in range(len(selected_cluster)):#------------calcualting distance


        q = manhattan_distance(cluster_2[i,2],selected_cluster[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_2[i,3],selected_cluster[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_2[i,4],selected_cluster[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_b.append(total)
   # print(min(distance_b))
    b = min(distance_b)
    maxi = max(b, a_sil)
    #print("a is", + maxi)
    d = b - a_sil
    sill = ((b - a_sil) / maxi)#------------silhoutte widht formulae
    print("Silhoutte width for each point in cluster 2", +  sill)

    average_sill.append(sill)
s_c2 = sum(average_sill)/len(average_sill)
print("Average silhoutee width for cluster 1", + s_c2)

```

```
Silhoutte width for each point in cluster 2 0.7154667837284167
```

```
Silhoutte width for each point in cluster 2 0.6806546975268585
Silhoutte width for each point in cluster 2 0.5570423743352204
Silhoutte width for each point in cluster 2 0.6713208152784463
Silhoutte width for each point in cluster 2 0.7021908003840447
Silhoutte width for each point in cluster 2 0.6816710100292189
Silhoutte width for each point in cluster 2 0.6340604668962877
Silhoutte width for each point in cluster 2 0.6354793198188163
Silhoutte width for each point in cluster 2 0.5421354764638348
Silhoutte width for each point in cluster 2 0.6746268656716418
Silhoutte width for each point in cluster 2 0.5625049356392638
Silhoutte width for each point in cluster 2 0.5895114590979531
Silhoutte width for each point in cluster 2 0.5617448471926084
Silhoutte width for each point in cluster 2 0.6631753513136075
Silhoutte width for each point in cluster 2 0.6682740145426712
Silhoutte width for each point in cluster 2 0.6722127069570182
Silhoutte width for each point in cluster 2 0.6903445734291507
Silhoutte width for each point in cluster 2 0.714680565426834
Silhoutte width for each point in cluster 2 0.667762880512691
Silhoutte width for each point in cluster 2 0.5710732054015636
Silhoutte width for each point in cluster 2 0.6808116281338408
Silhoutte width for each point in cluster 2 0.6846126510305615
Silhoutte width for each point in cluster 2 0.5294117647058824
Silhoutte width for each point in cluster 2 0.651842826682912
Silhoutte width for each point in cluster 2 0.5539947322212468
Silhoutte width for each point in cluster 2 0.6140724946695094
Silhoutte width for each point in cluster 2 0.6381633499170812
Silhoutte width for each point in cluster 2 0.6905269570514774
Silhoutte width for each point in cluster 2 0.6543811554472535
Silhoutte width for each point in cluster 2 0.5909667349526875
Silhoutte width for each point in cluster 2 0.44928419128845554
Silhoutte width for each point in cluster 2 0.669983416252073
Silhoutte width for each point in cluster 2 0.4650153487879751
Silhoutte width for each point in cluster 2 0.5744007236544549
Silhoutte width for each point in cluster 2 0.5212824765063571
Silhoutte width for each point in cluster 2 0.6169154228855723
Silhoutte width for each point in cluster 2 0.5558872305140964
Silhoutte width for each point in cluster 2 0.5924038344861061
Silhoutte width for each point in cluster 2 0.6170235777633571
Silhoutte width for each point in cluster 2 0.5963659961064246
Silhoutte width for each point in cluster 2 0.5685572139303484
Silhoutte width for each point in cluster 2 0.49366802351876976
Silhoutte width for each point in cluster 2 0.5442524221000262
```

```
Silhoutte width for each point in cluster 2 0.5865268253327955
Silhoutte width for each point in cluster 2 0.5814262023217248
Silhoutte width for each point in cluster 2 0.590049751243781
Silhoutte width for each point in cluster 2 0.5804726368159203
Silhoutte width for each point in cluster 2 0.5743504698728579
Silhoutte width for each point in cluster 2 0.5937418172296413
Silhoutte width for each point in cluster 2 0.574212271973466
Silhoutte width for each point in cluster 2 0.531370923161968
Silhoutte width for each point in cluster 2 0.5526652452025587
Silhoutte width for each point in cluster 2 0.4835820895522388
Silhoutte width for each point in cluster 2 0.5793373557743201
Silhoutte width for each point in cluster 2 0.3547079417726185
Silhoutte width for each point in cluster 2 0.42603648424543944
Silhoutte width for each point in cluster 2 0.4104477611940298
Silhoutte width for each point in cluster 2 0.48428312980551824
Silhoutte width for each point in cluster 2 0.3670110983543819
Silhoutte width for each point in cluster 2 0.2799464217374663
Silhoutte width for each point in cluster 2 0.5571200600769736
Silhoutte width for each point in cluster 2 0.40398009950248764
Silhoutte width for each point in cluster 2 0.38773872572620754
Silhoutte width for each point in cluster 2 0.37744610281923735
Silhoutte width for each point in cluster 2 0.08054963278843849
Silhoutte width for each point in cluster 2 0.007462686567164134
Silhoutte width for each point in cluster 2 0.1630735212824765
Average silhoutee width for cluster 1 0.551302933979199
```

In [28]:
```
distance_age_k2 = []
distance_income_k2 = []
distance_score_k2 = []
total_distance_k2 = []
distance_sil = []
average_sill = []
total=0
q=0
w=0
e=0
a=0

total=0

for i in range(len(cluster_3)):
    for j in range(len(cluster_3)):#------------calcualting distance
        q = manhattan_distance(cluster_3[i,2],cluster_3[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_3[i,3],cluster_3[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_3[i,4],cluster_3[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total/3)
    a_sil = sum(distance_sil)/len(cluster_3)
    #print(a_sil)
    distance_sil.clear()


    cluster_min=0

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0

```

```
    total=0
    n=2#--------------for cluster 2
    for j in range(len(mediod)):#------------calcualting nearest cluster
        q = manhattan_distance(mediod[n,2],mediod[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(mediod[n,3],mediod[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(mediod[n,4],mediod[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total)
    #print(distance_sil)
    index = [i for i in range(0,4)]
    #print(index)
    index=np.asarray(index)
    distance_sil= np.asarray(distance_sil)
    distance_sil.transpose
    #print(distance_sil)
    distance_sil = np.column_stack((index,distance_sil))
    #print(distance_sil[:,1])
    minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
    #print(minval)
    for b in range(len(distance_sil)):
        if (minval == distance_sil[b,1]):
            cluster_min = b
            #print(cluster_min)




    #---------now find the value of b
    if (cluster_min == 0):
            selected_cluster= cluster_1
    if (cluster_min == 1):
            selected_cluster= cluster_2
    if (cluster_min == 2):
            selected_cluster= cluster_3
    if (cluster_min == 3):
            selected_cluster= cluster_4
    if (cluster_min == 4):
            selected_cluster= cluster_5

```

```
    distance_age_k2 = []
    distance_b= []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
    s=0

    total=0

    for j in range(len(selected_cluster)):#------------calcualting distance


        q = manhattan_distance(cluster_3[i,2],selected_cluster[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_3[i,3],selected_cluster[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_3[i,4],selected_cluster[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_b.append(total)
   # print(min(distance_b))
    b = min(distance_b)
    maxi = max(b, a_sil)
    #print("a is", + maxi)
    d = b - a_sil
    sill = ((b - a_sil) / maxi)#------------silhoutte widht formulae
    print("Silhoutte width for each point in cluster 3", +  sill)

    average_sill.append(sill)
s_c3 = sum(average_sill)/len(average_sill)
print("Average silhoutee width for cluster 1", + s_c3)
```

```
Silhoutte width for each point in cluster 3 0.7216296296296297
Silhoutte width for each point in cluster 3 0.6790990990990993
Silhoutte width for each point in cluster 3 0.7112820512820514
Silhoutte width for each point in cluster 3 0.7006249999999998
Silhoutte width for each point in cluster 3 0.7
Silhoutte width for each point in cluster 3 0.6585416666666666
Silhoutte width for each point in cluster 3 0.726938775510204
Silhoutte width for each point in cluster 3 0.5637681159420288
Silhoutte width for each point in cluster 3 0.6827083333333334
Silhoutte width for each point in cluster 3 0.6306666666666667
Silhoutte width for each point in cluster 3 0.3079166666666667
Silhoutte width for each point in cluster 3 0.6621333333333334
Silhoutte width for each point in cluster 3 0.6315942028985507
Silhoutte width for each point in cluster 3 0.4395833333333333
Silhoutte width for each point in cluster 3 0.6683950617283949
Silhoutte width for each point in cluster 3 0.5309999999999998
Silhoutte width for each point in cluster 3 0.6152688172043013
Silhoutte width for each point in cluster 3 0.5477192982456139
Silhoutte width for each point in cluster 3 0.5687719298245614
Silhoutte width for each point in cluster 3 0.6358024691358026
Silhoutte width for each point in cluster 3 0.2554166666666664
Silhoutte width for each point in cluster 3 0.42070175438596497
Silhoutte width for each point in cluster 3 0.38410256410256405
Silhoutte width for each point in cluster 3 0.6935135135135135
Silhoutte width for each point in cluster 3 0.5220833333333333
Silhoutte width for each point in cluster 3 -0.07216494845360835
Silhoutte width for each point in cluster 3 0.324888888888889
Silhoutte width for each point in cluster 3 0.22428571428571434
Silhoutte width for each point in cluster 3 0.48311111111111116
Silhoutte width for each point in cluster 3 0.22615384615384612
Silhoutte width for each point in cluster 3 0.1326666666666668
Silhoutte width for each point in cluster 3 0.41499999999999987
Silhoutte width for each point in cluster 3 0.35333333333333333
Silhoutte width for each point in cluster 3 0.3461111111111111
Silhoutte width for each point in cluster 3 0.6493333333333333
Silhoutte width for each point in cluster 3 -0.3686109440769693
Silhoutte width for each point in cluster 3 0.5114814814814815
Silhoutte width for each point in cluster 3 0.43242424242424227
Silhoutte width for each point in cluster 3 0.5976923076923076
Silhoutte width for each point in cluster 3 0.13266666666666663
Silhoutte width for each point in cluster 3 0.4428571428571429
Silhoutte width for each point in cluster 3 0.5647222222222222
```

```
Silhoutte width for each point in cluster 3 0.6135802469135802
Silhoutte width for each point in cluster 3 0.08809523809523816
Silhoutte width for each point in cluster 3 -0.4942683749157114
Silhoutte width for each point in cluster 3 0.3946031746031749
Silhoutte width for each point in cluster 3 0.3098245614035088
Silhoutte width for each point in cluster 3 0.19958333333333331
Silhoutte width for each point in cluster 3 0.20714285714285716
Silhoutte width for each point in cluster 3 0.26333333333333353
Average silhoutee width for cluster 1 0.4327421765621817
```

In [29]:

```
distance_age_k2 = []
distance_income_k2 = []
distance_score_k2 = []
total_distance_k2 = []
distance_sil = []
average_sill = []
total=0
q=0
w=0
e=0
a=0

total=0

for i in range(len(cluster_4)):
    for j in range(len(cluster_4)):#------------calcualting distance
        q = manhattan_distance(cluster_4[i,2],cluster_4[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_4[i,3],cluster_4[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_4[i,4],cluster_4[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total/3)
    a_sil = sum(distance_sil)/len(cluster_4)
    #print(a_sil)
    distance_sil.clear()





    cluster_min=0

    distance_age_k2 = []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
```

```
    e=0
    a=0

    total=0
    n=3#--------------for cluster 4
    for j in range(len(mediod)):#------------calcualting nearest cluster
        q = manhattan_distance(mediod[n,2],mediod[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(mediod[n,3],mediod[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(mediod[n,4],mediod[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_sil.append(total)
    #print(distance_sil)
    index = [i for i in range(0,4)]
    #print(index)
    index=np.asarray(index)
    distance_sil= np.asarray(distance_sil)
    distance_sil.transpose
    #print(distance_sil)
    distance_sil = np.column_stack((index,distance_sil))
    #print(distance_sil[:,1])
    minval = np.min(distance_sil[:,1][np.nonzero(distance_sil[:,1])])
    #print(minval)
    for b in range(len(distance_sil)):
        if (minval == distance_sil[b,1]):
            cluster_min = b
            #print(cluster_min)




    #---------now find the value of b
    if (cluster_min == 0):
            selected_cluster= cluster_1
    if (cluster_min == 1):
            selected_cluster= cluster_2
    if (cluster_min == 2):
            selected_cluster= cluster_3
    if (cluster_min == 3):
            selected_cluster= cluster_4
```

```
    if (cluster_min == 4):
            selected_cluster= cluster_5

    distance_age_k2 = []
    distance_b= []
    distance_income_k2 = []
    distance_score_k2 = []
    total_distance_k2 = []
    distance_sil = []
    q=0
    w=0
    e=0
    a=0
    s=0

    total=0

    for j in range(len(selected_cluster)):#------------calcualting distance


        q = manhattan_distance(cluster_4[i,2],selected_cluster[j,2])
        #distance_age_k2.append(q)
        w = manhattan_distance(cluster_4[i,3],selected_cluster[j,3])
        #distance_income_k2.append(w)
        e = manhattan_distance(cluster_4[i,4],selected_cluster[j,4])
        #distance_score.append(e)
        total = q+w+e
        distance_b.append(total)
   # print(min(distance_b))
    b = min(distance_b)
    maxi = max(b, a_sil)
    #print("a is", + maxi)
    d = b - a_sil
    sill = ((b - a_sil) / maxi)#------------silhoutte widht formulae
    print("Silhoutte width for each point in cluster 4", +  sill)

    average_sill.append(sill)
s_c4 = sum(average_sill)/len(average_sill)
print("Average silhoutee width for cluster 1", + s_c4)

```

```
Silhoutte width for each point in cluster 4 -0.4558772235786535
Silhoutte width for each point in cluster 4 -0.5785411038209186
Silhoutte width for each point in cluster 4 -0.6812423375561913
Silhoutte width for each point in cluster 4 -0.08657465495608561
Silhoutte width for each point in cluster 4 0.5876532887402452
Silhoutte width for each point in cluster 4 0.12019230769230785
Silhoutte width for each point in cluster 4 0.5002003205128204
Silhoutte width for each point in cluster 4 0.18184885290148434
Silhoutte width for each point in cluster 4 0.6134992458521871
Silhoutte width for each point in cluster 4 0.44673382173382176
Silhoutte width for each point in cluster 4 0.5214342948717948
Silhoutte width for each point in cluster 4 0.5409382284382283
Silhoutte width for each point in cluster 4 0.5182291666666667
Silhoutte width for each point in cluster 4 0.14878542510121456
Silhoutte width for each point in cluster 4 0.4649725274725275
Silhoutte width for each point in cluster 4 0.5467032967032968
Silhoutte width for each point in cluster 4 0.6204309874522641
Silhoutte width for each point in cluster 4 0.6057692307692306
Silhoutte width for each point in cluster 4 0.5057692307692307
Silhoutte width for each point in cluster 4 0.5405518394648829
Silhoutte width for each point in cluster 4 0.4984526967285587
Silhoutte width for each point in cluster 4 0.5620845204178536
Silhoutte width for each point in cluster 4 0.645586785009862
Silhoutte width for each point in cluster 4 0.25993589743589746
Silhoutte width for each point in cluster 4 0.6392525913802508
Silhoutte width for each point in cluster 4 0.5904403567447046
Silhoutte width for each point in cluster 4 0.6556931768796175
Silhoutte width for each point in cluster 4 0.26660839160839184
Silhoutte width for each point in cluster 4 0.5796703296703296
Silhoutte width for each point in cluster 4 0.5902496626180836
Silhoutte width for each point in cluster 4 0.6493589743589743
Silhoutte width for each point in cluster 4 0.6088286713286714
Silhoutte width for each point in cluster 4 0.6677761341222881
Silhoutte width for each point in cluster 4 0.5972222222222221
Silhoutte width for each point in cluster 4 0.6254807692307692
Silhoutte width for each point in cluster 4 0.5843531468531469
Silhoutte width for each point in cluster 4 0.6377060439560438
Silhoutte width for each point in cluster 4 0.6283577533577533
Silhoutte width for each point in cluster 4 0.6567796610169492
Silhoutte width for each point in cluster 4 0.5958073458073458
Silhoutte width for each point in cluster 4 0.3552166224580018
```

```
Silhoutte width for each point in cluster 4 0.6483707264957265
Silhoutte width for each point in cluster 4 0.6044429708222812
Silhoutte width for each point in cluster 4 0.6756844850065189
Silhoutte width for each point in cluster 4 0.5447191697191698
Silhoutte width for each point in cluster 4 0.6309731934731935
Silhoutte width for each point in cluster 4 0.6187232905982907
Silhoutte width for each point in cluster 4 0.6763822115384615
Silhoutte width for each point in cluster 4 0.5754807692307693
Silhoutte width for each point in cluster 4 0.5209276018099546
Silhoutte width for each point in cluster 4 0.6532838506522719
Silhoutte width for each point in cluster 4 0.6814204314204315
Average silhoutee width for cluster 1 0.46897590767698333
```

Average silhoutte width of dataset

In [33]:

```
Average_dataset = (s_c1+s_c2+s_c3+s_c4)/4
print("Average silhoutte width of dataset", + Average_dataset)
```

```
Average silhoutte width of dataset 0.5325300537791802
```

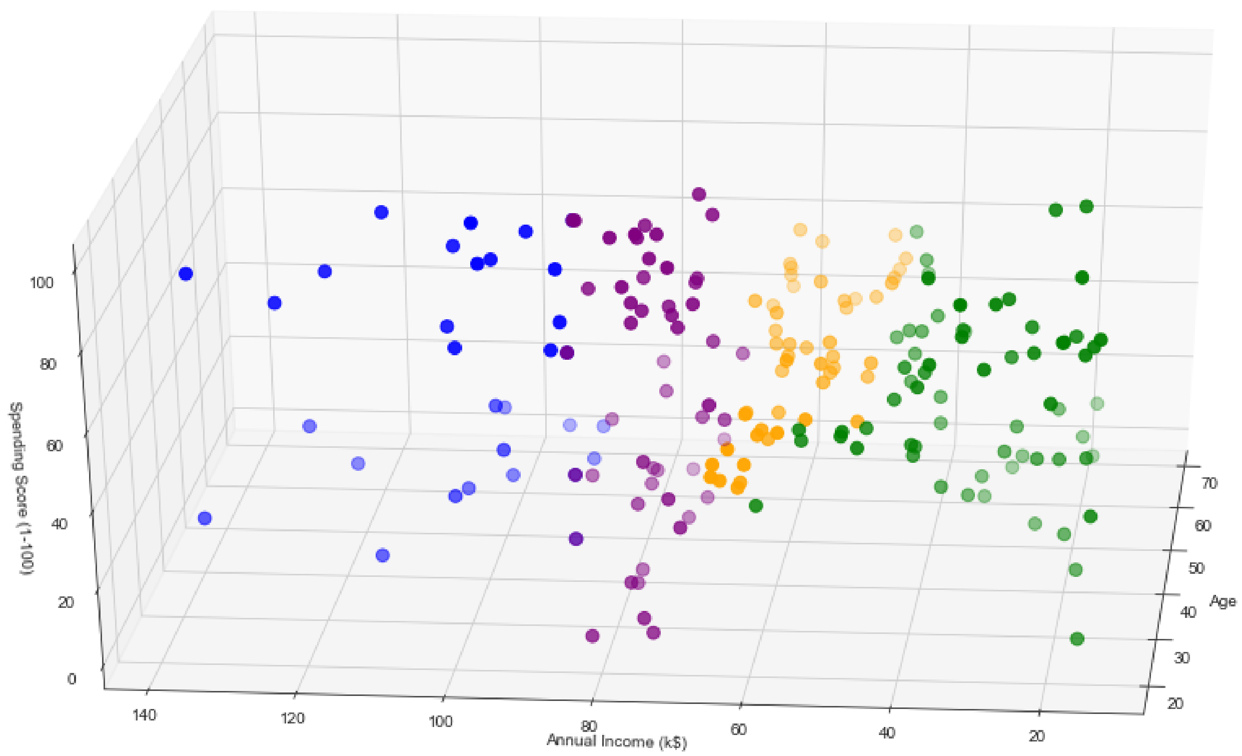
In [ ]:

```
```

Plotting assigned clusters below

In [31]:

```

cluster_1 = np.asarray(cluster_1)
cluster_2 = np.asarray(cluster_2)
cluster_3 = np.asarray(cluster_3)
cluster_4 = np.asarray(cluster_4)
sns.set_style("white")
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
clusterX1 = cluster_1[:,2].tolist()
clusterY1 = cluster_1[:,3].tolist()
clusterZ1 = cluster_1[:,4].tolist()
clusterX2 = cluster_2[:,2].tolist()
clusterY2 = cluster_2[:,3].tolist()
clusterZ2 = cluster_2[:,4].tolist()
clusterX3 = cluster_3[:,2].tolist()
clusterY3 = cluster_3[:,3].tolist()
clusterZ3 = cluster_3[:,4].tolist()
clusterX4 = cluster_4[:,2].tolist()
clusterY4 = cluster_4[:,3].tolist()
clusterZ4 = cluster_4[:,4].tolist()
ax.scatter(clusterX1, clusterY1, clusterZ1, c='blue', s=60)
ax.scatter(clusterX2, clusterY2, clusterZ2, c='green', s=60)
ax.scatter(clusterX3, clusterY3, clusterZ3, c='orange', s=60)
ax.scatter(clusterX4, clusterY4, clusterZ4, c='purple', s=60)
ax.view_init(30, 185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```



Saving to csv file

In [35]:
```
Lakehead Study material\Big data\Assignment 2\K mediod\Work directory\k_is_4\clusters_k.csv', index = None, header=True)
```

In [ ]:
```
1
```

In [ ]:
```
1
```