Market Segmentation
Segmentation for Bio-tech Start-up

Team Members:

M.Phani Tarun

Aayush Jariwala

Pratik Ramchandani

Jatin Gupta

Problem Statement

Our team has to work under a Bio-Tech Startup, who is going to launch its Home Checkup Service with Online Booking offering the following initial services.

1. Full Body Checkup with a Bio-Tech Device based on Blood Samples
2. Online Health Techs offering :
    i. Diabetes checkup device,
    ii. Blood Pressure checkup device,
    iii. Vitamins deficiency checkup device

Data Collection

- Most of the Data is primarily collected from Kaggle
- http://data.gov.in/
- https://data.worlbank.org

Segmentation Criteria

The term segmentation Criteria relates to the nature or the type of information used for market segmentation, unlike the segmentation variable which means the variable in empirical data in common sense segmentation for splitting the sample into market segments. In Segmentation we usually find the identifiable characteristics of individuals in the data sample and segmenting them into a same cluster and analyze the common interest needs to maximize the organizations profits. Segmentation Criteria is an important factor in market segmentation as well. The four main types in segmentation criteria are Geographic Segmentation, socio-demographic segmentation, psychographic segmentation and behavioral segmentation.

Geographic Segmentation

In Geographic Segmentation the key criteria to form market segments is the geographic location or the residence of the customer. There are some specific advantages of doing geographic segmentation, they are, we can segment down all the customers in that particular area, do promotions which are meaningful in that area and even run adds in news-papers, television, etc. in that local area. The only key disadvantage is that it not always the case that all the people residing in the same location will have same opinions and preferences in the products.

Socio-Demographic Segmentation

Socio-Demographic Segmentation criteria includes parameters like age, gender, education, income, etc. For ex, while buying cosmetics criteria associated is gender, while buying branded and luxury items criteria associated is income, while planning on vacation destination criteria associated is age (i.e., if people go in couple the vacation destination will be different if people going with children, then the

vacation destination is different). The socio-demographic segmentation at times with better data can give us the better market segments and gives us the clear clarity on the who the customer is, this is achievable provided better data that provides sufficient insights about who the customer is and the market segments. But in many cases, socio-demographic segmentation would not be the best fit for product preferences.

Psychographic Segmentation

For making market segments using the Psychographic segmentation the criteria is the Psychological criteria for grouping people. Parameters like interest, beliefs, aspirations, preferences, benefits, etc. can be used to define psychological criteria. Psychographic segmentation is more complex by nature compared to Geographic Segmentation and Socio-Demographic segmentation because, we cannot find a single fixed parameter for insights for better segmentation, there are a lot of factors effecting the psychographic criteria and the factors are different in each person. Therefore, we must use a lot of segmentation variables. And the main advantage that psychographic segmentation has is that clustering a common set of customers based on psychographic criteria for maximizing profits. For ex., people who want to go on a vacation and has a preference for attending historic pilgrims can be clustered and can be taken together which can reduce cost for company and maximize the profit as well.

Behavioral Segmentation

In Behavioral segmentation we can directly find similarities in behaviors of customers. There can be many useful implementations possible for doing market segments. Behavioural segmentation criteria depend on the way visitors interact with the website. Some data depends on their immediate online behaviour and giving positive feedback while other data depends on their past offline behaviour or negative feedback.

Pre-Processing Data before performing Segmentation

1) Categorical Variables

Two pre-processing procedures are often used for categorical variables. One is merging levels of categorical variables before further analysis, the other one is converting categorical variables to numeric ones, if it makes sense to do so. Merging levels of categorical variables is useful if the original categories are too differentiated (too many).

2) Numerical Variables

In distance-based methods of segment extraction, the range of values of a segmentation variable determines its relative influence. If one of the segmentation variables is binary (with values 0 or 1 indicating whether or not a customer views on the product of fast food), and a second variable indicates the expenditure in dollars per person per day (with values ranging from zero to $1000), a one-dollar difference in spend per person per day is weighted equally as the difference in liking to dine out or not.

3) Univariate Variables

We take one feature and based on that we will try to classify what the output is going to be. In McDonald’s dataset, we took age as feature and classified based how much they are liked. From our data all the persons who gave positive feedback ‘4’ and above their age is around ‘20’ and the data are fit (overlapped) one guy from age.

4) Bivariate Variables

Bivariate analysis is slightly more analytical than Univariate analysis. When the data set contains two variables and researchers aim to undertake comparisons between the two data set then Bivariate analysis is the right type of analysis technique.

5) Multivariate Variables

Multivariate analysis is a more complex form of statistical analysis technique and used when there are more than two variables in the data set. Here we can apply PCA to reduce the dimensions.

Extracting Market Segments

k-Means and k-Centroid Clustering

For the k-means algorithm based on the squared Euclidean distance, the centroid consists of the column-wise mean values across all members of the market segment. The data set contains observations (consumers) in rows, and variables (behavioural information or answers to survey questions) in columns. The column-wise mean, therefore, is the average response pattern across all segmentation variables for all members of the segment. The algorithm represents a heuristic for solving the optimisation problem of dividing consumers into a given number of segments such that consumers are similar to their fellow segment members, but dissimilar to members of other segments. This algorithm is iterative; it improves the partition in each step, and is bound to converge, but not necessarily to the global optimum.

“Improved” k-Means

Many attempts have been made to refine and improve the k-means clustering algorithm. The simplest improvement is to initialise k-means

using “smart” starting values, rather than randomly drawing k consumers from the data set and using them as starting points. Using randomly drawn consumers is suboptimal because it may result in some of those randomly drawn consumers being located very close to one another, and thus not being representative of the data space. Using starting points that are not representative of the data space increases the likelihood of the k-means algorithm getting stuck in what is referred to as a local optimum. A local optimum is a good solution, but not the best possible solution. The best starting points are those that best represent the data. Good representatives are close to their segment members; the total distance of all segment members to their representatives is small. Bad representatives are far away from their segment members; the total distance of all segment members to their representatives is high.

READING DATA

In [1]:
```
import pandas as pd

path_of_Diabetes_Data_csv = r'C:\Users\TARUN\Desktop\Internship\Task-2\Datasets\diabetes\diabetes.csv'
data = pd.read_csv(path_of_Diabetes_Data_csv)
```

In [2]:
```
data.head()
```

Out[2]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia
0	40	Male	No	Yes	No	Yes	No	No	No	Yes	No	Yes	No	Yes	Yes
1	58	Male	No	No	No	Yes	No	No	Yes	No	No	No	Yes	No	Yes
2	41	Male	Yes	No	No	Yes	Yes	No	No	Yes	No	Yes	No	Yes	Yes
3	45	Male	No	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No
4	60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes

ANALYSING DATASET

In [3]:
```
data.shape
```

Out[3]:
```
(520, 17)
```

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
Age                 0
Gender              0
Polyuria            0
Polydipsia          0
sudden weight loss  0
weakness            0
Polyphagia          0
Genital thrush      0
visual blurring     0
Itching             0
Irritability        0
delayed healing     0
partial paresis     0
muscle stiffness    0
Alopecia            0
Obesity             0
class               0
dtype: int64
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Age                 520 non-null    int64 
 1   Gender              520 non-null    object
 2   Polyuria            520 non-null    object
 3   Polydipsia          520 non-null    object
 4   sudden weight loss  520 non-null    object
 5   weakness            520 non-null    object
 6   Polyphagia          520 non-null    object
```

```
 7   Genital thrush      520 non-null    object
 8   visual blurring     520 non-null    object
 9   Itching             520 non-null    object
 10  Irritability        520 non-null    object
 11  delayed healing     520 non-null    object
 12  partial paresis     520 non-null    object
 13  muscle stiffness    520 non-null    object
 14  Alopecia            520 non-null    object
 15  Obesity             520 non-null    object
 16  class               520 non-null    object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
```

In [6]:
```
data.describe()
```

Out[6]:

	Age
count	520.000000
mean	48.028846
std	12.151466
min	16.000000
25%	39.000000
50%	47.500000
75%	57.000000
max	90.000000

DROPPING AND SELECTING FEATURES

In [7]:
```
subset = data[["Age", "Gender", "class"]]
subset.drop(subset[subset['class'] == 'Negative'].index, inplace=True)
subset.drop('class', axis=1, inplace=True)

subset.head()
```

```
<ipython-input-7-3abb4cd54897>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  subset.drop(subset[subset['class'] == 'Negative'].index, inplace=True)
<ipython-input-7-3abb4cd54897>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  subset.drop('class', axis=1, inplace=True)
```

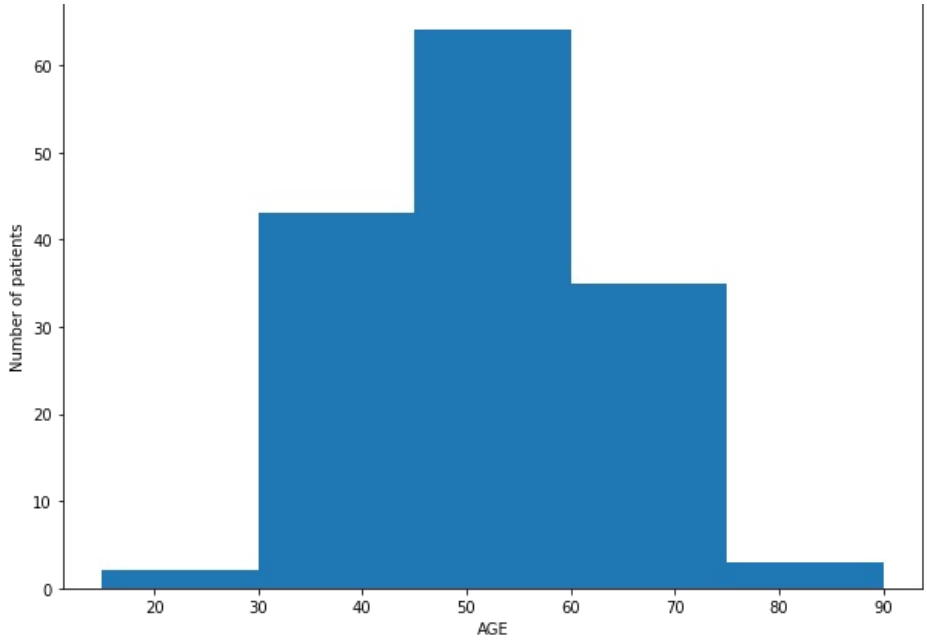
Out[7]:

	Age	Gender
0	40	Male
1	58	Male
2	41	Male
3	45	Male
4	60	Male

VISUALISATION

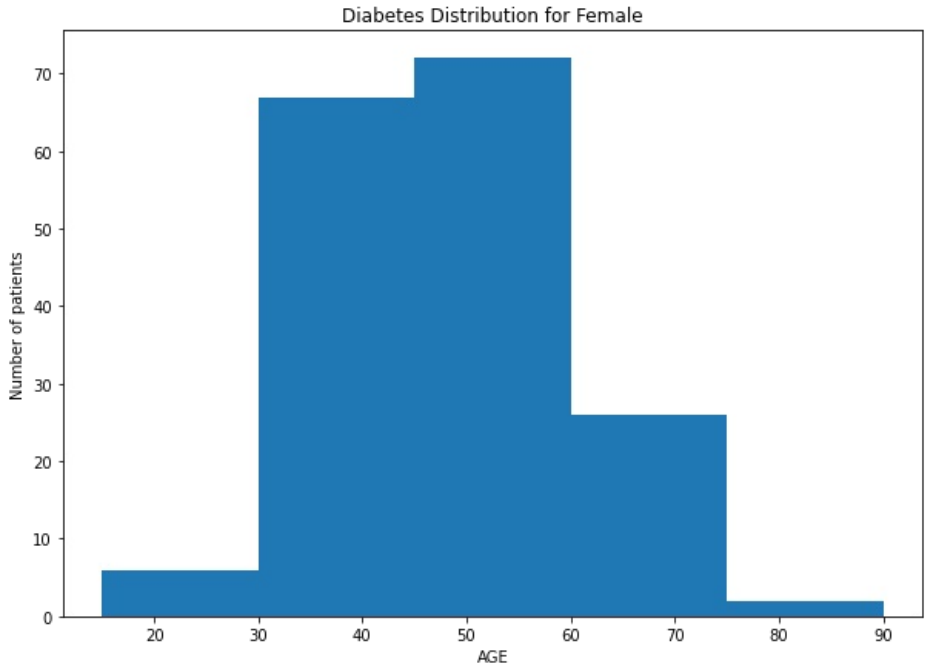
In [8]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize =(10, 7))
ax.hist( subset.loc[subset['Gender'] == 'Male'].Age , bins = [15, 30, 45, 60, 75, 90])
ax.set_title('Diabetes Distribution for Male')
ax.set_xlabel("AGE")
ax.set_ylabel("Number of patients")
```

Out[8]:
```
Text(0, 0.5, 'Number of patients')
```



In [9]:
```
fig, ax = plt.subplots(figsize =(10, 7))
ax.hist( subset.loc[subset['Gender'] == 'Female'].Age , bins = [15, 30, 45, 60, 75, 90])
ax.set_title('Diabetes Distribution for Female')
ax.set_xlabel("AGE")
ax.set_ylabel("Number of patients")
```

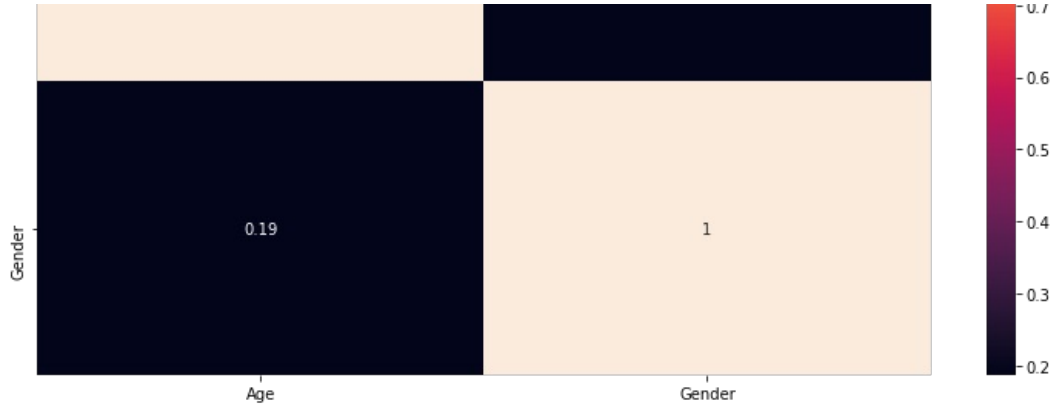
Out[9]:
```
Text(0, 0.5, 'Number of patients')
```



In [10]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df1=subset.apply(le.fit_transform)
```

In [11]:
```
import seaborn as sb
plt.figure(figsize=(13,7))
sb.heatmap(df1.corr(),annot=True)
```

Out[11]:
```
<AxesSubplot:>
```



II

READING DATASET

In [12]:
```
import pandas as pd

path_of_BP_Data_csv = r'C:\Users\TARUN\Desktop\Internship\Task-2\Datasets\BP\BP_data.csv'
data = pd.read_csv(path_of_BP_Data_csv)
```

In [13]:
```
data.head()
```

Out[13]:

	Patient_Number	Blood_Pressure_Abnormality	Level_of_Hemoglobin	Genetic_Pedigree_Coefficient	Age	BMI	Sex	Pregnancy	Smoking	Phys
0	1	1	11.28	0.90	34	23	1	1.0	0	
1	2	0	9.75	0.23	54	33	1	NaN	0	
2	3	1	10.79	0.91	70	49	0	NaN	0	
3	4	0	11.00	0.43	71	50	0	NaN	0	
4	5	1	14.17	0.83	52	19	0	NaN	0	

ANALYSING DATASET

In [14]:
```
data.shape
```

Out[14]:
```
(2000, 15)
```

In [15]:
```
data.isnull().sum()
```

Out[15]:
```
Patient_Number                    0
Blood_Pressure_Abnormality        0
Level_of_Hemoglobin               0
Genetic_Pedigree_Coefficient     92
Age                               0
BMI                               0
Sex                               0
Pregnancy                      1558
Smoking                           0
Physical_activity                 0
salt_content_in_the_diet          0
alcohol_consumption_per_day     242
Level_of_Stress                   0
Chronic_kidney_disease            0
Adrenal_and_thyroid_disorders     0
dtype: int64
```

In [16]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype
```

```
---  ------                          --------------  -----
 0   Patient_Number                  2000 non-null   int64
 1   Blood_Pressure_Abnormality      2000 non-null   int64
 2   Level_of_Hemoglobin             2000 non-null   float64
 3   Genetic_Pedigree_Coefficient    1908 non-null   float64
 4   Age                             2000 non-null   int64
 5   BMI                             2000 non-null   int64
 6   Sex                             2000 non-null   int64
 7   Pregnancy                       442 non-null    float64
 8   Smoking                         2000 non-null   int64
 9   Physical_activity               2000 non-null   int64
 10  salt_content_in_the_diet        2000 non-null   int64
 11  alcohol_consumption_per_day     1758 non-null   float64
 12  Level_of_Stress                 2000 non-null   int64
 13  Chronic_kidney_disease          2000 non-null   int64
 14  Adrenal_and_thyroid_disorders   2000 non-null   int64
dtypes: float64(4), int64(11)
memory usage: 234.5 KB
```

In [17]:
```
data.describe()
```

Out[17]:

	Patient_Number	Blood_Pressure_Abnormality	Level_of_Hemoglobin	Genetic_Pedigree_Coefficient	Age	BMI	Sex
count	2000.000000	2000.000000	2000.000000	1908.000000	2000.000000	2000.000000	2000.000000
mean	1000.500000	0.493500	11.710035	0.494817	46.558500	30.081500	0.496000
std	577.494589	0.500083	2.186701	0.291736	17.107832	11.761208	0.500109
min	1.000000	0.000000	8.100000	0.000000	18.000000	10.000000	0.000000
25%	500.750000	0.000000	10.147500	0.240000	32.000000	20.000000	0.000000
50%	1000.500000	0.000000	11.330000	0.490000	46.000000	30.000000	0.000000
75%	1500.250000	1.000000	12.945000	0.740000	62.000000	40.000000	1.000000
max	2000.000000	1.000000	17.560000	1.000000	75.000000	50.000000	1.000000

DROPPING AND SELECTING FEATURES

In [18]:
```
# removing pateints(rows) without diabets
data.drop(data[data['Blood_Pressure_Abnormality'] == 0].index, inplace = True)

# selecting columns which are useful for segmentation
subset = data[["Age", "BMI", "alcohol_consumption_per_day"]]
subset = subset.dropna()
```

In [19]:
```
data.isnull().sum()
```

Out[19]:
```
Patient_Number                    0
Blood_Pressure_Abnormality        0
Level_of_Hemoglobin               0
Genetic_Pedigree_Coefficient     30
Age                               0
BMI                               0
Sex                               0
Pregnancy                       754
Smoking                           0
Physical_activity                 0
salt_content_in_the_diet          0
alcohol_consumption_per_day     146
Level_of_Stress                   0
Chronic_kidney_disease            0
Adrenal_and_thyroid_disorders     0
dtype: int64
```

In [20]:
```
subset.head()
```

Out[20]:

	Age	BMI	alcohol_consumption_per_day
2	70	49	67.0
4	52	19	397.0
6	43	41	206.0
9	40	44	95.0
10	70	28	46.0

In [21]:
```
subset.describe()
```

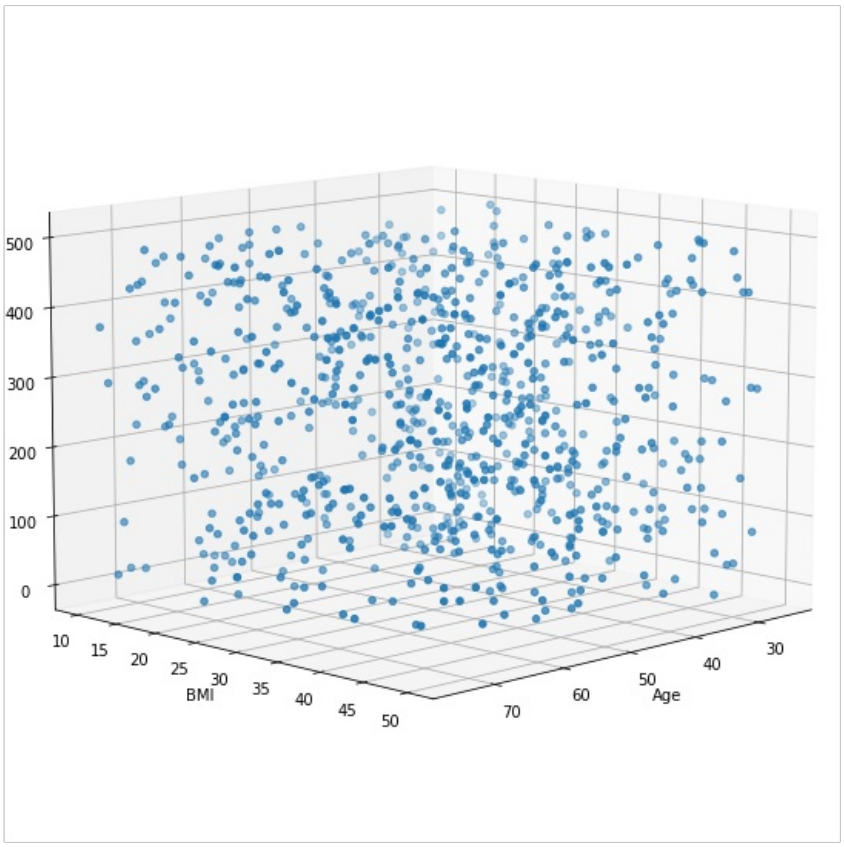
Out[21]:

	Age	BMI	alcohol_consumption_per_day
count	841.000000	841.000000	841.000000
mean	49.665874	30.552913	254.123662
std	14.968185	11.689937	144.294990
min	25.000000	10.000000	0.000000
25%	37.000000	21.000000	131.000000
50%	49.000000	30.000000	252.000000
75%	63.000000	41.000000	382.000000
max	75.000000	50.000000	499.000000

VISUALISATION

In [22]:
```
import matplotlib.pyplot as plt
```

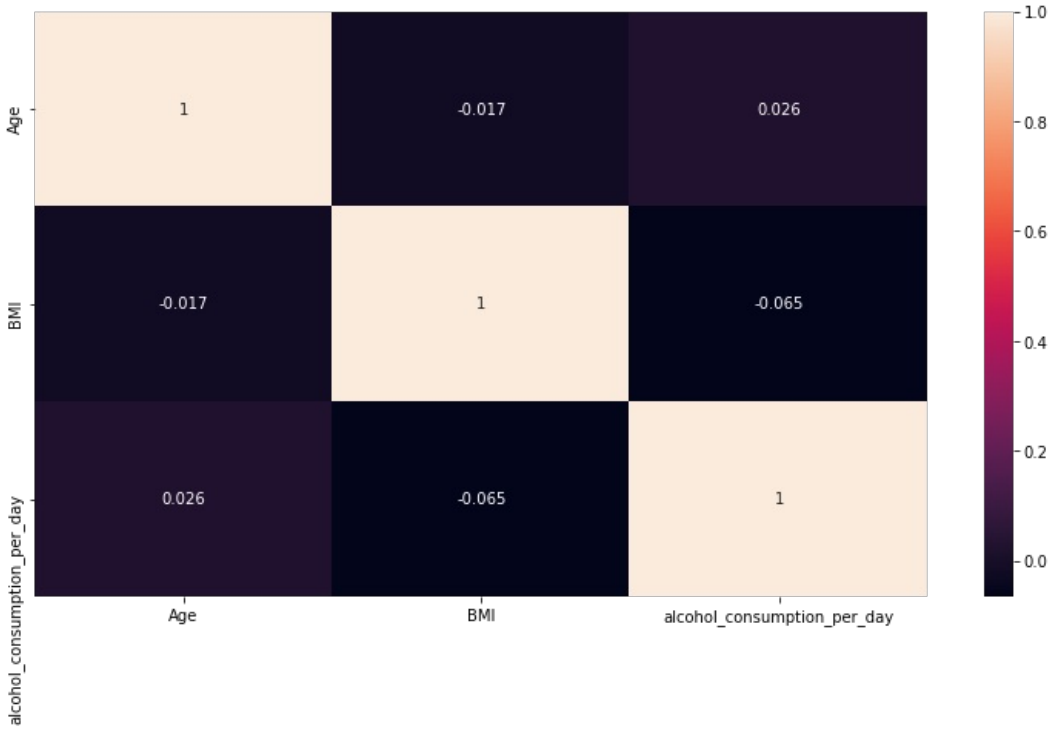
In [23]:
```
graph = plt.figure(figsize=(10,10)).gca(projection='3d')
col1, col2, col3 = 'Age', 'BMI', 'alcohol_consumption_per_day'

graph.scatter(subset[col1], subset[col2], subset[col3])
graph.set_xlabel(col1)
graph.set_ylabel(col2)
graph.set_zlabel(col3)
graph.view_init(10, 45)
plt.show()
```



In [24]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df2=subset.apply(le.fit_transform)
```

In [25]:
```
import seaborn as sb
plt.figure(figsize=(13,7))
sb.heatmap(df2.corr(),annot=True)
```

Out[25]: <AxesSubplot:>



K-MEANS CLUSTERING

In [26]:
```
from sklearn.cluster import KMeans
from matplotlib import pyplot as plt
import numpy as np

X=np.array(df2)
```

In [27]:
```
kmeans=KMeans(n_clusters=3)
kmeans.fit(X)
y_kmeans=kmeans.predict(X)
print(y_kmeans)
```

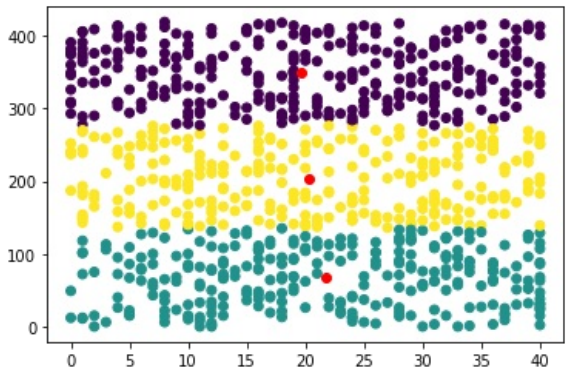
```
[1 0 2 1 1 0 1 0 0 1 0 1 0 2 2 2 0 2 0 2 2 2 1 2 1 1 2 0 0 2 0 1 0 1 0 2 0
 2 0 0 2 1 0 2 1 0 2 0 1 0 2 1 2 2 2 0 2 0 2 1 0 2 1 1 2 1 1 2 0 1 1 0 0 0
 1 1 2 0 0 1 1 0 2 1 0 0 0 0 0 1 2 0 2 0 0 1 1 0 0 0 0 1 0 0 2 2 2 1 1 2 1
 0 1 2 2 2 0 2 0 2 2 2 1 2 2 2 1 2 1 0 0 0 2 2 1 2 2 0 2 2 0 0 2 0 2 0 0 0
 1 1 2 0 2 2 1 1 1 1 1 1 0 1 0 0 0 2 0 2 1 0 0 1 1 1 1 2 1 2 1 2 1 1 1 2 2
 1 1 2 1 1 2 0 1 2 0 0 1 0 2 1 2 2 0 1 2 0 1 0 0 1 1 2 2 2 2 0 1 0 0 0 0 1
 0 0 2 2 0 0 2 1 0 2 0 0 0 2 1 0 0 0 1 1 2 2 1 2 0 0 0 0 1 1 0 1 0 2 0 2 1 0
 2 0 2 1 2 1 1 2 0 0 1 2 1 2 1 0 1 0 0 0 2 2 1 1 1 0 1 2 0 0 2 0 1 2 2 0 0 0
 0 2 1 0 0 1 0 1 1 2 0 2 1 1 0 0 2 0 0 2 0 2 0 0 2 0 2 0 0 2 0 1 2 2 0 0 0
 2 1 0 2 0 1 2 1 1 2 0 1 1 2 0 0 2 0 0 1 2 1 1 1 2 0 1 0 2 2 2 0 0 0 1 0 1
 2 0 1 2 1 0 2 2 2 0 1 1 1 1 1 2 1 2 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 2 2 0 1
 1 0 0 1 0 1 2 0 2 2 2 0 2 1 2 0 2 0 2 2 1 2 1 1 2 2 1 2 2 1 0 1 1 0 1 0 1
 2 0 0 2 2 0 1 0 2 2 1 1 1 1 1 0 1 2 0 2 2 0 0 1 0 2 2 1 0 0 0 0 2 0 2 0 1
 2 2 2 0 0 1 2 0 0 2 0 2 0 2 1 0 1 2 2 0 1 1 2 1 2 0 0 0 0 2 0 1 2 1 0 0 0
 2 1 1 1 1 0 1 2 0 2 2 0 2 2 0 2 2 2 1 0 2 0 0 1 2 2 0 0 1 0 1 1 0 1 0 2 2
 0 1 0 1 2 1 1 2 1 0 1 1 0 0 0 2 2 1 1 2 2 2 2 2 2 1 1 2 0 1 1 1 2 0 0 0 1 1
 0 2 1 2 2 1 2 1 2 0 0 1 0 0 2 1 1 2 1 1 0 2 0 0 2 2 1 2 2 1 0 1 1 1 2 2 1
 0 0 0 2 0 1 1 0 1 2 1 1 2 1 2 1 1 0 1 0 0 2 1 0 2 1 1 1 0 1 2 0 1 2 0 0 2
 0 2 2 1 0 1 2 0 1 1 2 0 1 1 0 2 1 2 0 2 2 2 1 2 2 0 0 2 1 0 1 1 0 2 2 2 1
 2 1 1 0 2 1 0 1 0 0 2 1 0 0 2 2 0 0 2 0 0 1 0 0 0 1 0 2 1 2 2 0 0 0 0 0 2
 0 0 0 2 1 2 2 2 0 2 1 0 2 0 2 2 0 0 2 2 1 2 0 0 0 1 2 1 2 0 2 1 1 1 0 2 1
 1 0 0 1 2 0 0 2 0 1 1 0 2 1 1 1 2 2 2 1 2 2 1 0 0 2 1 1 2 2 0 0 2 2 1 1 1
 0 1 0 0 2 2 1 2 0 1 1 2 1 0 0 1 1 2 0 0 2 1 1 1 1 1 2]
```

In [28]:
```
print(kmeans.cluster_centers_)
```

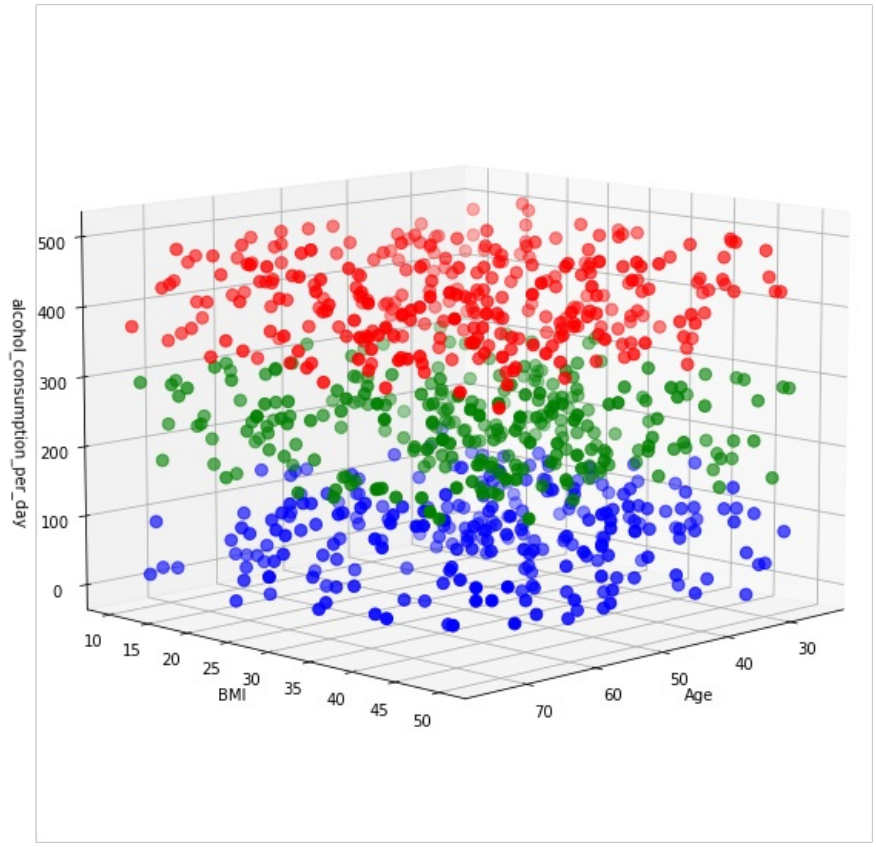
```
[[ 24.82312925  19.62585034 349.71428571]
 [ 24.11355311  21.76923077  67.65934066]
 [ 25.04744526  20.33576642 203.28832117]]
```

In [30]:
```
plt.scatter(X[:,1],X[:,2],c=y_kmeans)
centers=kmeans.cluster_centers_
plt.scatter(centers[:,1],centers[:,2],c='red')
```

Out[30]: <matplotlib.collections.PathCollection at 0x266c56398e0>



In [31]:
```
from mpl_toolkits.mplot3d import Axes3D

km = KMeans(n_clusters=3)
clusters = km.fit_predict(subset)
subset["label"] = clusters

fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(subset[col1][subset.label == 0], subset[col2][subset.label == 0], subset[col3][subset.label == 0], c='
ax.scatter(subset[col1][subset.label == 1], subset[col2][subset.label == 1], subset[col3][subset.label == 1], c='
ax.scatter(subset[col1][subset.label == 2], subset[col2][subset.label == 2], subset[col3][subset.label == 2], c='

ax.view_init(10, 45)
plt.xlabel(col1)
plt.ylabel(col2)
ax.set_zlabel(col3)
plt.show()
```



III

READING DATA

In [32]:
```
import pandas as pd

path_of_BP_Data_csv = r"C:\Users\TARUN\Desktop\Internship\Task-2\Datasets\Vitamin\vitamins_data.csv"
data = pd.read_csv(path_of_BP_Data_csv)
```

In [33]:
```
data.head()
```

Out[33]:

	State	Population(0-6)years	VitA_deficit%	VitD_deficit%
0	India	163819614	17.6	13.8
1	Delhi	2016849	17.8	32.5
2	Haryana	3335537	26.1	27.6
3	Himachal Pradesh	793137	5.9	4.6
4	Jammu & Kashmir	1485803	8.7	22.9

ANALYSING DATASET

In [34]:
```
data.shape
```

Out[34]:
```
(31, 4)
```

In [35]:
```
data.isnull().sum()
```

Out[35]:
```
State                  0
Population(0-6)years   0
VitA_deficit%          2
VitD_deficit%          0
dtype: int64
```

In [36]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31 entries, 0 to 30
Data columns (total 4 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   State                 31 non-null     object
 1   Population(0-6)years  31 non-null     int64
 2   VitA_deficit%         29 non-null     float64
 3   VitD_deficit%         31 non-null     float64
dtypes: float64(2), int64(1), object(1)
memory usage: 1.1+ KB
```

In [37]:
```
data.describe().apply(lambda s: s.apply('{0:.5f}'.format))
```

Out[37]:

	Population(0-6)years	VitA_deficit%	VitD_deficit%
count	31.00000	29.00000	31.00000
mean	10622915.22581	17.43103	15.79355
std	29193291.30761	9.73378	13.26670
min	78195.00000	2.40000	1.10000
25%	630558.00000	9.60000	6.10000
50%	3554916.00000	17.10000	12.80000
75%	8852130.50000	21.40000	22.80000
max	163819614.00000	43.20000	52.10000

DROPPING ROWS HAVING NULL VALUES

In [38]:
```
VitA = data[["State", "VitA_deficit%"]]
VitA = VitA.dropna()
VitA.drop(VitA[VitA.State == 'India'].index, inplace=True)
VitA.head()
```

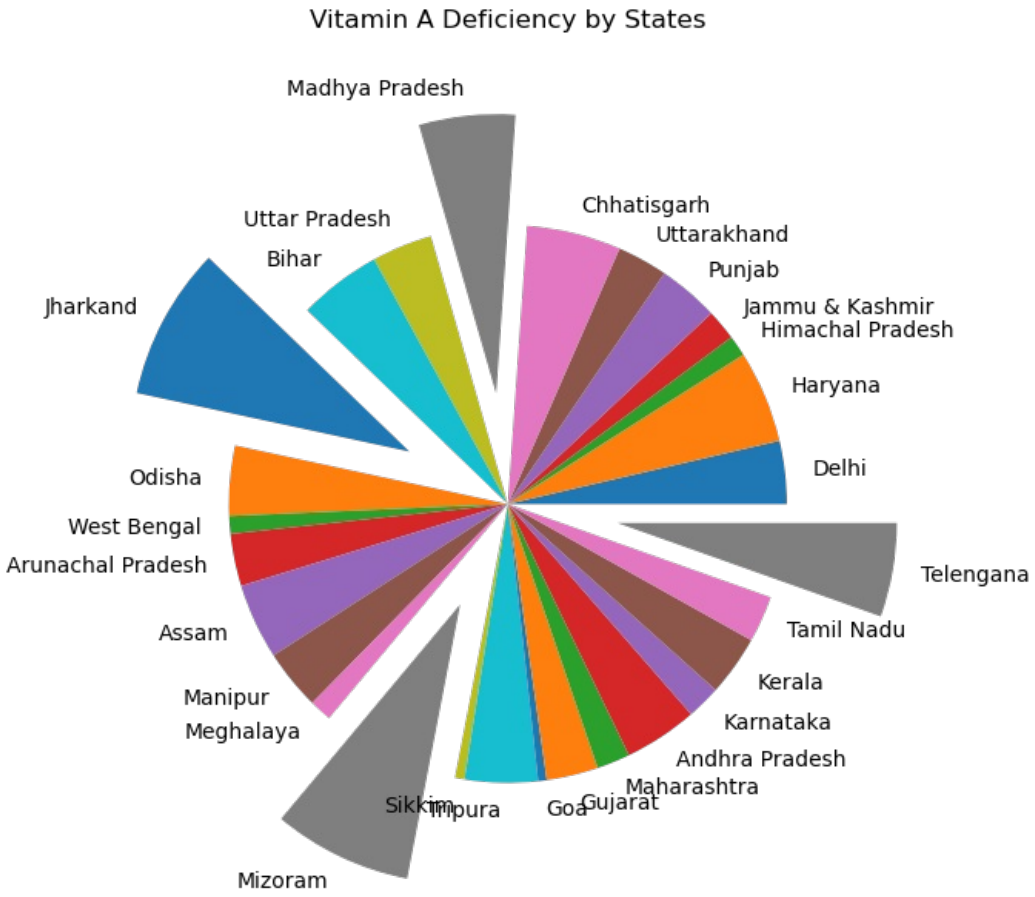
Out[38]:

	State	VitA_deficit%
1	Delhi	17.8
2	Haryana	26.1

3	Himachal Pradesh	5.9
4	Jammu & Kashmir	8.7
5	Punjab	17.2

VISUALISATION

In [39]:
```
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

figure(figsize=(6, 6), dpi=100)
myexplode = [0, 0, 0, 0, 0, 0, 0, 0.4, 0, 0, 0.4, 0, 0, 0, 0, 0, 0, 0.4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.4]
plt.pie(VitA['VitA_deficit%'], labels = VitA.State, explode = myexplode)
plt.title('Vitamin A Deficiency by States', pad=60)
plt.show()
```

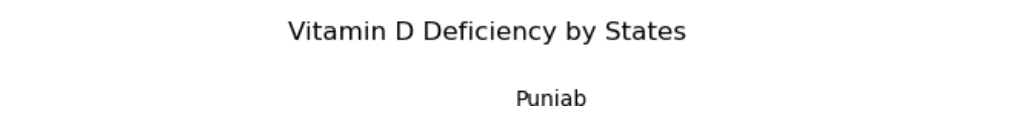


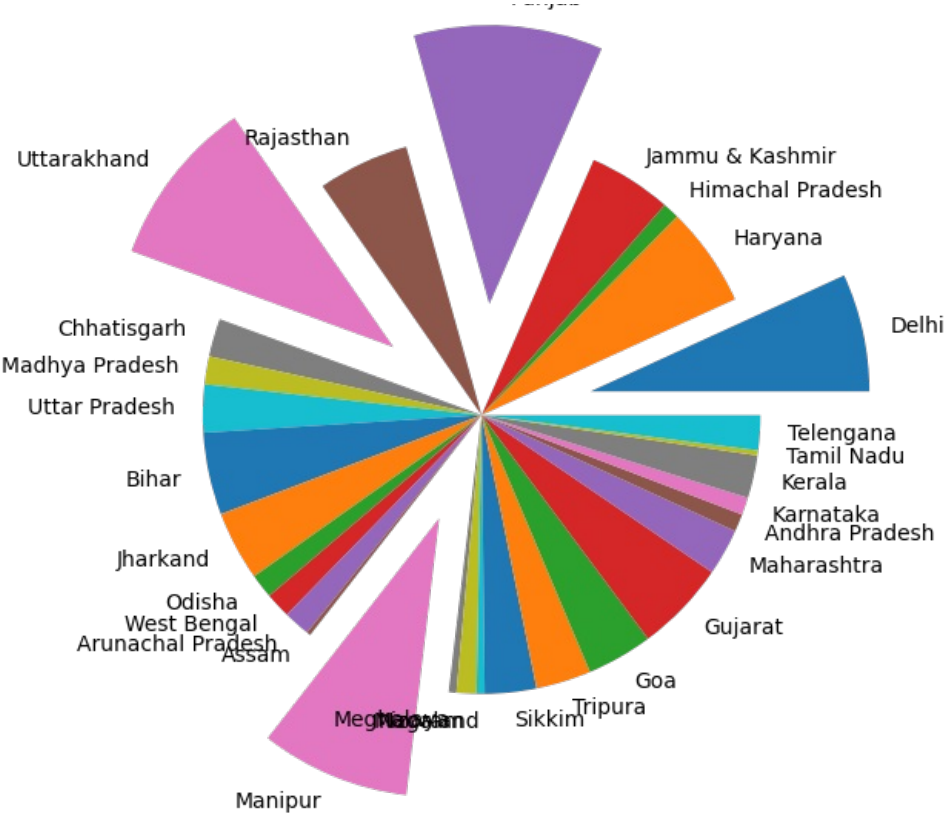
In [40]:
```
VitD = data[["State", "VitD_deficit%"]]
VitD = VitD.dropna()
VitD.drop(VitD[VitD.State == 'India'].index, inplace=True)
VitD.head()
```

Out[40]:

	State	VitD_deficit%
1	Delhi	32.5
2	Haryana	27.6
3	Himachal Pradesh	4.6
4	Jammu & Kashmir	22.9
5	Punjab	52.1

In [41]:
```
figure(figsize=(6, 6), dpi=100)
myexplode = [0.4, 0, 0, 0, 0.4, 0, 0.4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
plt.pie(VitD['VitD_deficit%'], labels = VitD.State, explode = myexplode)
plt.title('Vitamin D Deficiency by States', pad=60)
plt.show()
```





IV

READING DATASET

In [42]:
```
import pandas as pd
heart_attack_data = pd.read_csv(r"C:\Users\TARUN\Desktop\Internship\Task-2\Datasets\Full body checkup\heart_attac
heart_attack_data.head()
```

Out[42]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

ANALYSING DATASET

In [43]:
```
heart_attack_data.shape
```

Out[43]:
```
(918, 12)
```

In [44]:
```
heart_attack_data.isnull().sum()
```

Out[44]:
```
Age              0
Sex              0
ChestPainType    0
RestingBP        0
Cholesterol      0
FastingBS        0
RestingECG       0
MaxHR            0
ExerciseAngina   0
Oldpeak          0
ST_Slope         0
HeartDisease     0
dtype: int64
```

In [45]:
```
heart_attack_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

In [46]:
```
heart_attack_data.describe()
```

Out[46]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

DROPPING AND SELECTING FEATURES

In [47]:
```
# removing pateints(rows) without diabets
heart_attack_data.drop(heart_attack_data[heart_attack_data['HeartDisease'] == 0].index, inplace = True)

#selecting columns for subset
subset = heart_attack_data[["Age","Cholesterol","MaxHR"]]
subset = subset.dropna()
subset.head()
```

Out[47]:

	Age	Cholesterol	MaxHR
1	49	180	156
3	48	214	108
8	37	207	130
11	58	164	99
13	49	234	140

In [48]:
```
subset.describe()
```

Out[48]:

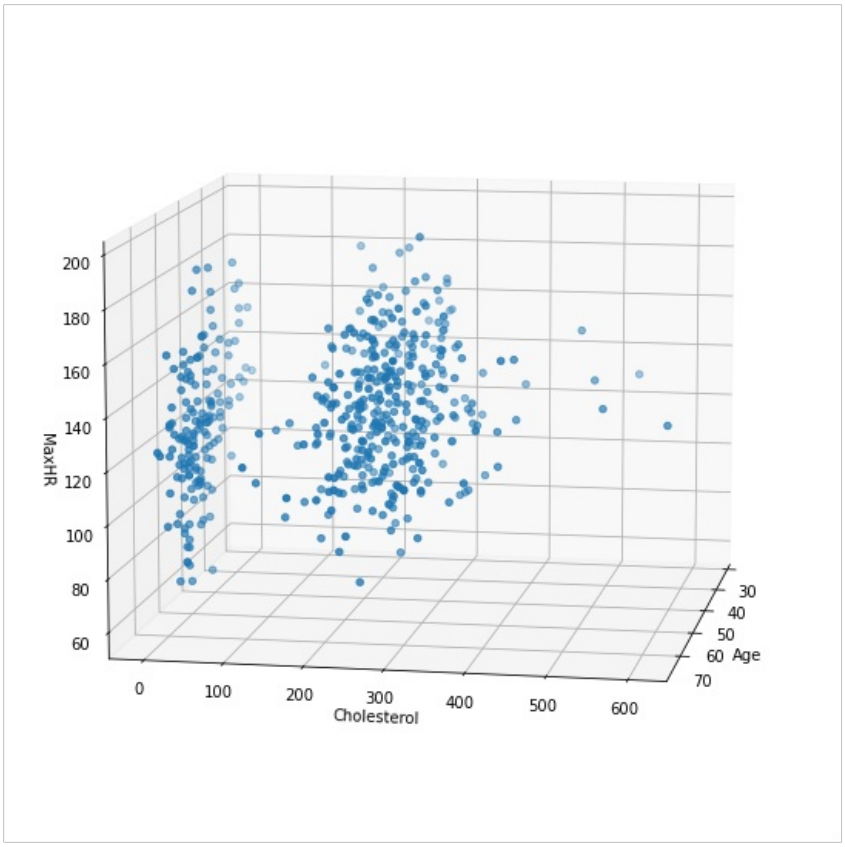
	Age	Cholesterol	MaxHR
count	508.000000	508.000000	508.000000
mean	55.899606	175.940945	127.655512
std	8.727056	126.391398	23.386923
min	31.000000	0.000000	60.000000
25%	51.000000	0.000000	112.000000
50%	57.000000	217.000000	126.000000
75%	62.000000	267.000000	144.250000

max	77.000000	603.000000	195.000000

VISUALISATION

In [49]:
```
import matplotlib.pyplot as plt
```

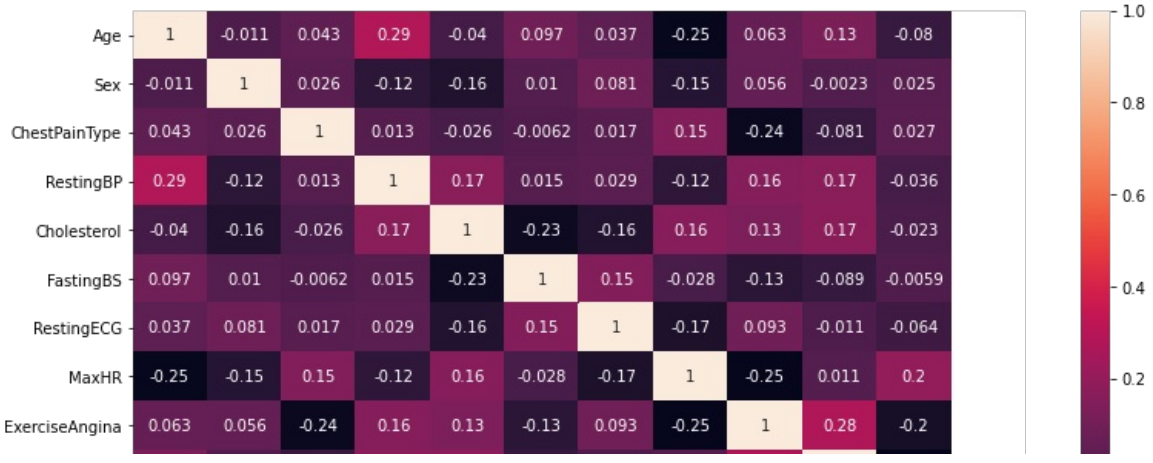
In [50]:
```
graph = plt.figure(figsize=(10,10)).gca(projection='3d')
col1, col2, col3 = 'Age', 'Cholesterol', 'MaxHR'

graph.scatter(subset[col1], subset[col2], subset[col3])
graph.set_xlabel(col1)
graph.set_ylabel(col2)
graph.set_zlabel(col3)
graph.view_init(10, 10)
plt.show()
```

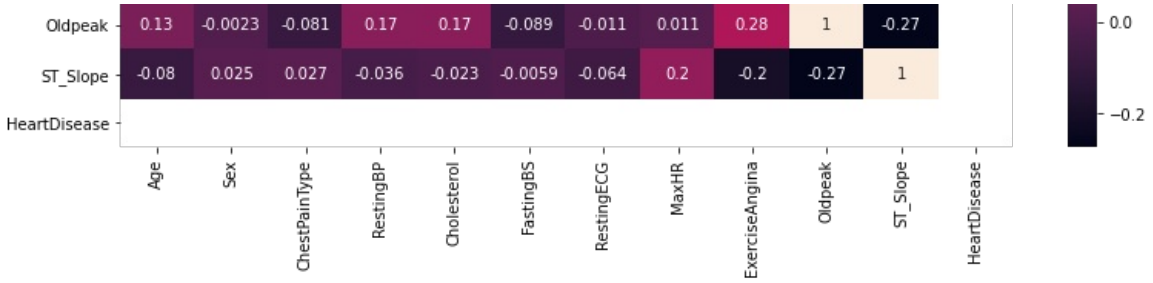


In [51]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df4=heart_attack_data.apply(le.fit_transform)
```

In [52]:
```
import seaborn as sb
plt.figure(figsize=(13,7))
sb.heatmap(df4.corr(),annot=True)
```

Out[52]: <AxesSubplot:>





K-MEANS CLUSTERING

In [54]:
```
from sklearn.cluster import KMeans
from matplotlib import pyplot as plt
import numpy as np

X=np.array(df4)
```

In [55]:
```
kmeans=KMeans(n_clusters=3)
kmeans.fit(X)
y_kmeans=kmeans.predict(X)
print(y_kmeans)
```

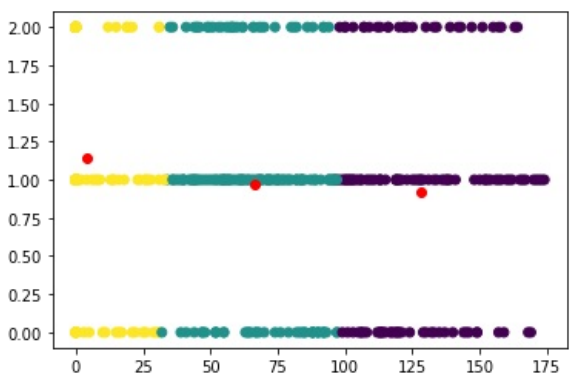
```
[2 1 1 2 1 1 1 0 0 0 1 2 0 0 2 0 1 1 0 1 0 0 1 0 2 0 0 1 1 1 2 1 0 0 1 0 0
 0 2 1 1 0 0 2 1 0 1 0 1 0 2 1 0 0 0 0 0 0 1 0 1 1 0 1 0 0 1 1 0 1 1 0 1 1
 1 1 0 1 0 1 0 0 2 0 0 0 2 1 0 0 1 0 1 0 0 1 1 1 2 1 0 1 1 2 1 1 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
 1 1 1 2 0 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 0 1 2 2 2 0 2 1 2 2 2
 2 2 2 0 2 2 0 2 2 2 0 2 2 1 1 1 1 1 2 0 1 0 1 0 0 1 0 1 1 1 1 2 1 0 1 2 1
 1 2 0 0 0 2 0 2 1 1 0 2 1 1 2 2 1 1 1 2 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0
 1 1 2 1 2 1 0 2 0 2 0 2 1 1 1 0 1 1 1 0 1 1 2 1 1 1 0 0 0 1 0 2 0 0 1 1 0
 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 0 1 1 2 1 1 1 0 2 1 0 0 0 0 1 1 0 1 1 0 1 0
 0 1 1 0 1 1 0 1 0 1 2 0 0 1 0 0 2 0 0 0 1 0 1 1 1 1 1 0 0 1 2 1 1 1 1 1 1
 0 1 0 0 0 1 1 0 2 2 0 2 0 0 0 1 0 0 0 0 1 1 1 0 1 0 0 1 1 2 2 0 0 1 1 2 0
 2 0 0 0 1 0 1 0 1 1 1 1 0 1 1 0 1 1 2 1 1 2 1 1 1 2 1]
```

In [56]:
```
print(kmeans.cluster_centers_)
```

```
[[ 23.93283582   0.82835821   0.42537313  32.53731343 128.40298507
    0.23880597   0.91791045  55.68656716   0.70895522  22.2238806
    1.03731343   0.        ]
 [ 25.67977528   0.91573034   0.46629213  29.16292135  66.43258427
    0.24157303   0.97191011  54.16292135   0.64606742  21.61797753
    1.0505618    0.        ]
 [ 24.85204082   0.93877551   0.44897959  26.18367347   4.2244898
    0.48469388   1.1377551   46.28571429   0.54081633  18.17346939
    1.07653061   0.        ]]
```

In [57]:
```
plt.scatter(X[:,4],X[:,6],c=y_kmeans)
centers=kmeans.cluster_centers_
plt.scatter(centers[:,4],centers[:,6],c='red')
```

Out[57]:
```
<matplotlib.collections.PathCollection at 0x266c71c1460>
```



In [58]:
```
from mpl_toolkits.mplot3d import Axes3D

km = KMeans(n_clusters=3)
clusters = km.fit_predict(subset)
subset["label"] = clusters

fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(subset[col1][subset.label == 0], subset[col2][subset.label == 0], subset[col3][subset.label == 0], c='
ax.scatter(subset[col1][subset.label == 1], subset[col2][subset.label == 1], subset[col3][subset.label == 1], c='
ax.scatter(subset[col1][subset.label == 2], subset[col2][subset.label == 2], subset[col3][subset.label == 2], c='

ax.view_init(10, 10)
plt.xlabel(col1)
plt.ylabel(col2)
ax.set_zlabel(col3)
plt.show()
```