Diabetes Prediction Dataset

EDA PART-1

Import Requierd Packages

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read the Dataset

In [3]:
```
file_path='C:\\Users\\kurre\\OneDrive\\Documents\\Naresh IT\\datafiles\\dia
pd.read_csv(file_path)
```

Out[3]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	
...	...	...	...	...	...	...	...	
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99997	Male	66.0	0	0	former	27.83	5.7	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

100000 rows × 9 columns

In [3]:
```
file_path='C:\\Users\\kurre\\OneDrive\\Documents\\Naresh IT\\datafiles\\dia
diabetes_df=pd.read_csv(file_path)
```

In [4]:
```
diabetes_df
```

Out[4]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	
...	...	...	...	...	...	...	...	
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99997	Male	66.0	0	0	former	27.83	5.7	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

100000 rows × 9 columns

Shape:

In [5]:
```
diabetes_df.shape
```

Out[5]:
```
(100000, 9)
```

Size:

In [6]:
```
diabetes_df.size
```

Out[6]:
```
900000
```

Columns:

In [7]:
```
diabetes_df.columns
```

Out[7]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'smoking_histor
y',
       'bmi', 'HbA1c_level', 'blood_glucose_level', 'diabetes'],
      dtype='object')
```

Dtype:

In [8]:
```
diabetes_df.dtypes

# object means categorical column
# int64 and float64 means numerical column
```

Out[8]:
```
gender                  object
age                    float64
hypertension             int64
heart_disease            int64
smoking_history         object
bmi                    float64
HbA1c_level            float64
blood_glucose_level      int64
diabetes                 int64
dtype: object
```

Head:

In [9]:
```
diabetes_df.head()

# top 5 rows
# starting index with zero
```

Out[9]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glu
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	

Tail:

In [10]:
```
diabetes_df.tail()

# last 5 rows
```

Out[10]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99997	Male	66.0	0	0	former	27.83	5.7	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

Take:

In [11]:
```
# take random list which you want

list1=[100,200,300]
diabetes_df.take(list1)
```

Out[11]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_
100	Male	38.0	0	0	never	27.32	6.0	
200	Female	20.0	0	0	never	28.25	5.8	
300	Female	66.0	0	0	never	27.32	4.0	

In [12]:
```
#list1=[100,200,300]
#diabetes_df.take(list1,axis=1)

# axis=1 column
# 100,200,300
# it gives error because there is only 9 columns
```

In [13]:
```
diabetes_df.take([2,3,8],axis=1)

# in python index start with zero
# 2 means===== 3rd column
# 3 means===== 4th column
# 8 means====== 9th column
```

Out[13]:

	hypertension	heart_disease	diabetes
0	0	1	0
1	0	0	0
2	0	0	0
3	0	0	0
4	1	1	0
...	...	...	...
99995	0	0	0
99996	0	0	0
99997	0	0	0
99998	0	0	0
99999	0	0	0

100000 rows × 3 columns

In [14]:
```
diabetes_df.take([2,3,8],axis=0)    # now rows will come
```

Out[14]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glu
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
8	Male	42.0	0	0	never	33.64	4.8	

iloc:

In [15]:
```
# iloc will take numbers only
# this is patternt to find specific rows and columns


#diabetes_df.iloc[start:end,start:end]
```

In [16]:
```
diabetes_df.iloc[100:120] # if you not provide columns value that means all
```

Out[16]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_
100	Male	38.0	0	0	never	27.32	6.0	
101	Female	74.0	0	0	former	27.32	6.6	
102	Male	27.0	0	0	No Info	27.32	6.2	
103	Female	55.0	0	0	ever	18.60	6.0	
104	Female	80.0	1	0	never	27.32	6.8	
105	Female	57.0	0	0	never	24.18	5.8	
106	Female	80.0	0	0	No Info	27.32	4.8	
107	Female	80.0	0	0	never	37.26	4.0	
108	Male	40.0	0	0	No Info	27.32	4.8	
109	Female	9.0	0	0	No Info	19.39	6.0	
110	Female	50.0	0	0	No Info	31.21	6.2	
111	Female	62.0	0	0	never	21.12	5.0	
112	Male	37.0	0	0	No Info	27.14	3.5	
113	Female	47.0	0	0	never	20.60	4.0	
114	Female	55.0	1	0	never	34.20	5.7	
115	Male	28.0	0	0	never	27.63	6.2	
116	Female	66.0	0	0	not current	20.30	6.1	
117	Female	43.0	0	0	current	18.67	6.2	
118	Female	12.0	0	0	No Info	20.90	3.5	
119	Male	68.0	1	1	current	27.32	5.0	

In [17]:
```
diabetes_df.iloc[60:66,3:7]
```

Out[17]:

	heart_disease	smoking_history	bmi	HbA1c_level
60	0	current	27.86	6.6
61	0	not current	26.10	5.8
62	0	current	27.32	6.5
63	0	former	27.32	6.0
64	0	not current	30.22	5.7
65	1	ever	23.11	6.5

In [18]:
```
diabetes_df.iloc[[100,200,300],[3,5,8]]

# iloc take only number this drawback fix with loc keywords
```

Out[18]:

	heart_disease	bmi	diabetes
100	0	27.32	0
200	0	28.25	0
300	0	27.32	0

loc:

In [19]:
```
diabetes_df.loc[[100,200,300],'bmi']
```

Out[19]:
```
100    27.32
200    28.25
300    27.32
Name: bmi, dtype: float64
```

In [20]:
```
diabetes_df.loc[[100,200,300],['bmi']]
```

Out[20]:

	bmi
100	27.32
200	28.25
300	27.32

In [21]:
```
rows=[101,200,301]
cols=['smoking_history','bmi']
diabetes_df.loc[rows,cols]
```

Out[21]:

	smoking_history	bmi
101	former	27.32
200	never	28.25
301	No Info	27.32

len:

In [22]:
```
len(diabetes_df)
```

Out[22]:
```
100000
```

isnull:

- Any missing values are there in the data
- Is null you are asking qustion to computer: True or False
- If any missing value is there it is True
- If data present/no missing value it is False

In [23]:
```
diabetes_df.isnull()
```

Out[23]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
99995	False	False	False	False	False	False	False	
99996	False	False	False	False	False	False	False	
99997	False	False	False	False	False	False	False	
99998	False	False	False	False	False	False	False	
99999	False	False	False	False	False	False	False	

100000 rows × 9 columns

In [24]:
```
diabetes_df.isnull().sum()  # no missing value in this data set because 0
```

Out[24]:
```
gender                 0
age                    0
hypertension           0
heart_disease          0
smoking_history        0
bmi                    0
HbA1c_level            0
blood_glucose_level    0
diabetes               0
dtype: int64
```

info:

In [25]:
```
diabetes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   gender               100000 non-null  object 
 1   age                  100000 non-null  float64
 2   hypertension         100000 non-null  int64  
 3   heart_disease        100000 non-null  int64  
 4   smoking_history      100000 non-null  object 
 5   bmi                  100000 non-null  float64
 6   HbA1c_level          100000 non-null  float64
 7   blood_glucose_level  100000 non-null  int64  
 8   diabetes             100000 non-null  int64  
dtypes: float64(3), int64(4), object(2)
memory usage: 6.9+ MB
```

Missing values analysis

In [26]:
```
# No missing value
```

EDA PART-2

Categorical data analysis

How to read a column

In [27]:
```
file_path="C:\\Users\\kurre\\OneDrive\\Documents\\Naresh IT\\datafiles\\dia
diabetes_df=pd.read_csv(file_path)
```

In [28]:
```
diabetes_df
```

Out[28]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	
...	...	...	...	...	...	...	...	
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99997	Male	66.0	0	0	former	27.83	5.7	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

100000 rows × 9 columns

unique

In [29]:
```
# first read the column
# the apply unique

# dont apply unique operation for dataframe : [[]]
# apply only for series :[]
```

In [30]:
```
diabetes_df['smoking_history'].unique()
```

Out[30]:
```
array(['never', 'No Info', 'current', 'former', 'ever', 'not current'],
      dtype=object)
```

In [31]:
```
len(diabetes_df['smoking_history'].unique())
```

Out[31]:
```
6
```

In [32]:
```
len(diabetes_df['smoking_history'])
```

Out[32]:
```
100000
```

In [33]:
```
diabetes_df['smoking_history'].value_counts()
```

Out[33]:
```
smoking_history
No Info        35816
never          35095
former          9352
current         9286
not current     6447
ever            4004
Name: count, dtype: int64
```

nunique

In [34]:
```
diabetes_df['smoking_history'].nunique()

# number of unique lables
```

Out[34]:
```
6
```

In [35]:
```
diabetes_df[['gender','smoking_history']]
```

Out[35]:

	gender	smoking_history
0	Female	never
1	Female	No Info
2	Male	never
3	Female	current
4	Male	current
...	...	...
99995	Female	No Info
99996	Female	No Info
99997	Male	former
99998	Female	never
99999	Female	current

100000 rows × 2 columns

In [36]:
```
# we read smoking history column
# we understood there 6 unique lables are there
# these 6 unique lables repaeting and total 100000 observations
# how many are 'current' are there
# how many are 'never' are there


# we read gender column
# we understood there 6 unique lables are there
# these 6 unique lables repaeting and total 100000 observations
# how many are 'Female' are there
# how many are 'Male' are there
```

In [37]:
```
diabetes_df['gender']=='Female'
```

Out[37]:
```
0         True
1         True
2        False
3         True
4        False
         ...  
99995     True
99996     True
99997    False
99998     True
99999     True
Name: gender, Length: 100000, dtype: bool
```

In [38]:
```
len(diabetes_df['gender']=='Female')
```

Out[38]:
```
100000
```

In [39]:
```
diabetes_df[diabetes_df['gender']=='Female']
```

Out[39]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
3	Female	36.0	0	0	current	23.45	5.0	
5	Female	20.0	0	0	never	27.32	6.6	
6	Female	44.0	0	0	never	19.31	6.5	
...	...	...	...	...	...	...	...	
99994	Female	36.0	0	0	No Info	24.60	4.8	
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

58552 rows × 9 columns

In [40]:
```
len(diabetes_df[diabetes_df['gender']=='Female'])
```

Out[40]:
```
58552
```

In [41]:
```
diabetes_df['smoking_history']
```

Out[41]:
```
0          never
1        No Info
2          never
3        current
4        current
          ...
99995    No Info
99996    No Info
99997     former
99998      never
99999    current
Name: smoking_history, Length: 100000, dtype: object
```

In [42]:
```
diabetes_df['smoking_history']=='never'
```

Out[42]:
```
0         True
1        False
2         True
3        False
4        False
         ...
99995    False
99996    False
99997    False
99998     True
99999    False
Name: smoking_history, Length: 100000, dtype: bool
```

In [43]:
```
diabetes_df[diabetes_df['smoking_history']=='never']
```

Out[43]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	Female	80.0	0	1	never	25.19	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
5	Female	20.0	0	0	never	27.32	6.6	
6	Female	44.0	0	0	never	19.31	6.5	
8	Male	42.0	0	0	never	33.64	4.8	
...	...	...	...	...	...	...	...	
99986	Female	63.0	0	0	never	29.01	4.8	
99987	Female	23.0	0	0	never	17.87	5.8	
99992	Female	26.0	0	0	never	34.34	6.5	
99993	Female	40.0	0	0	never	40.69	3.5	
99998	Female	24.0	0	0	never	35.42	4.0	

35095 rows × 9 columns

In [44]:
```
len(diabetes_df[diabetes_df['smoking_history']=='never'])
```

Out[44]:
```
35095
```

Value-Count

In [45]:
```
diabetes_df['gender'].value_counts()
```

Out[45]:
```
gender
Female    58552
Male      41430
Other        18
Name: count, dtype: int64
```

In [46]:
```
diabetes_df['smoking_history'].value_counts()
```

Out[46]:
```
smoking_history
No Info        35816
never          35095
former          9352
current         9286
not current     6447
ever            4004
Name: count, dtype: int64
```

$Method - 1$

Type *Markdown* and LaTeX: α^2

In [49]:
```
#How to create a dataframe using value counts
#       or   using series

diabetes_df['smoking_history'].value_counts()


# in order to create a dataframe
# we need two list
# or one dictionary

# from value counts create two lists
# values
# keys
```

Out[49]:
```
smoking_history
No Info        35816
never          35095
former          9352
current         9286
not current     6447
ever            4004
Name: count, dtype: int64
```

In [50]:
```
smoking=diabetes_df['smoking_history'].value_counts().keys()
count=diabetes_df['smoking_history'].value_counts().values
smoking_df=pd.DataFrame(zip(smoking,count),columns=['smoking_history','coun


smoking_df
```

Out[50]:

	smoking_history	count
0	No Info	35816
1	never	35095
2	former	9352
3	current	9286
4	not current	6447
5	ever	4004

$Method-2$

In [51]:
```
dict1=dict(diabetes_df['smoking_history'].value_counts())
print(dict1)

pd.DataFrame(dict1,index=['count'])
```

```
{'No Info': 35816, 'never': 35095, 'former': 9352, 'current': 9286, 'not c
urrent': 6447, 'ever': 4004}
```

Out[51]:

	No Info	never	former	current	not current	ever
count	35816	35095	9352	9286	6447	4004

$Method-3$

In [52]:
```
dict1=dict(diabetes_df['smoking_history'].value_counts())
keys=dict1.keys()
values=dict1.values()

pd.DataFrame(zip(keys,values),columns=['smoking_history','count'])
```

Out[52]:

	smoking_history	count
0	No Info	35816
1	never	35095
2	former	9352
3	current	9286
4	not current	6447
5	ever	4004

Frequency Table

In [53]:
```
# this is a frequancy table

smoking_df

# one column= categorical : Continents
#     column= numerical : Count
```

Out[53]:

	smoking_history	count
0	No Info	35816
1	never	35095
2	former	9352
3	current	9286
4	not current	6447
5	ever	4004

Bar Chart

- x-axis: categorical column
- y-axis: numerical column
- where you are taking the data: smoking_df

In [54]:
```
smoking_df
# we are creating from scratch
```

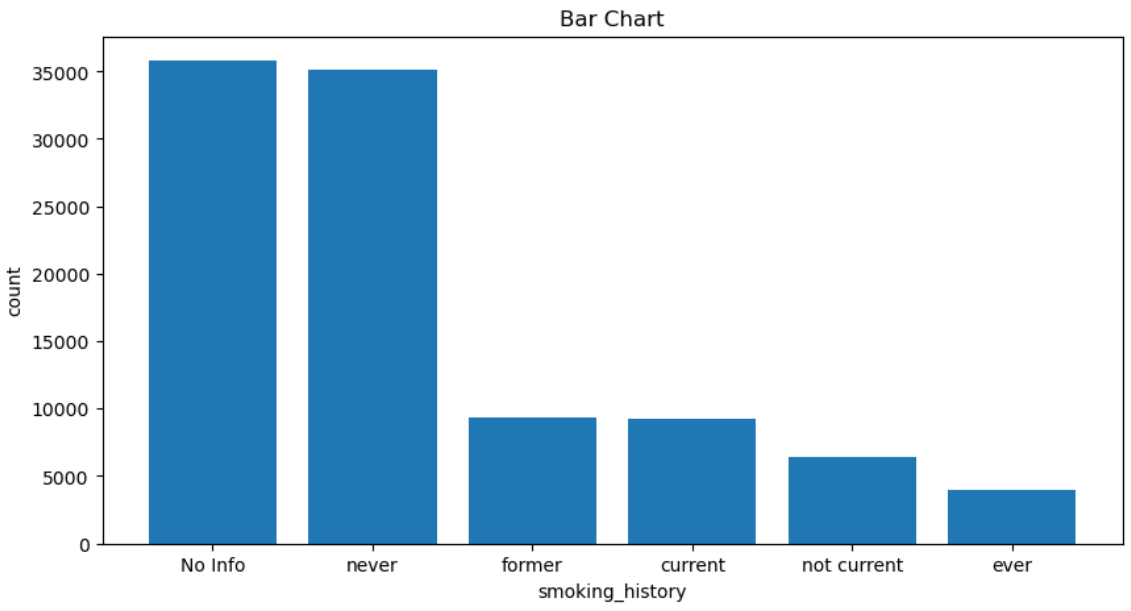
Out[54]:

	smoking_history	count
0	No Info	35816
1	never	35095
2	former	9352
3	current	9286
4	not current	6447
5	ever	4004

In [55]:
```
# one column= categorical : smoking_history  == x-axis
#     collumn= numerical : Count        == y-axis
```

In [56]:
```
# always take values count dataframe == for proper order
# dont take scratch level dataframe
```

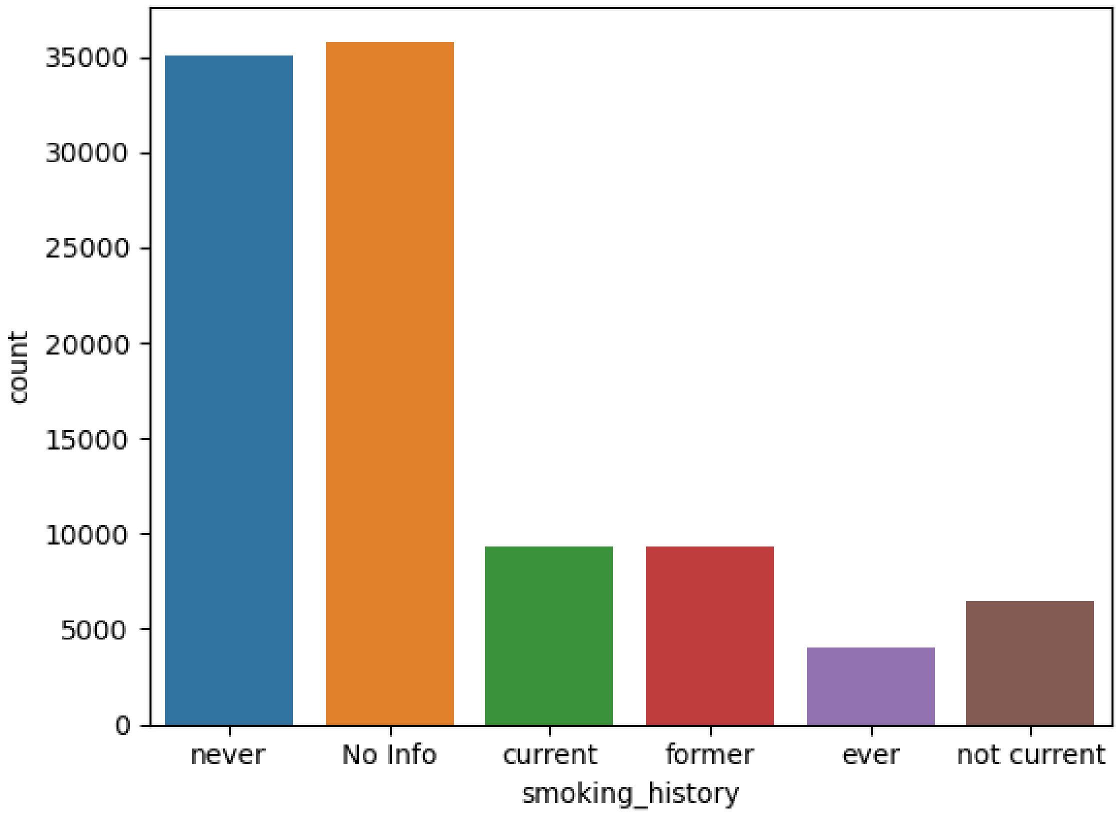
In [57]:
```python

# for clear visulization of both axis
plt.figure(figsize=(10,5))
# 10= horizontal x
# 5= vertical  y

plt.bar('smoking_history','count',data=smoking_df)
plt.title('Bar Chart')
plt.xlabel('smoking_history')
plt.ylabel('count')
plt.show()
```



Bar plot using seaborn

In [58]:
```
import seaborn as sns

sns.countplot(data=diabetes_df,x='smoking_history')
```

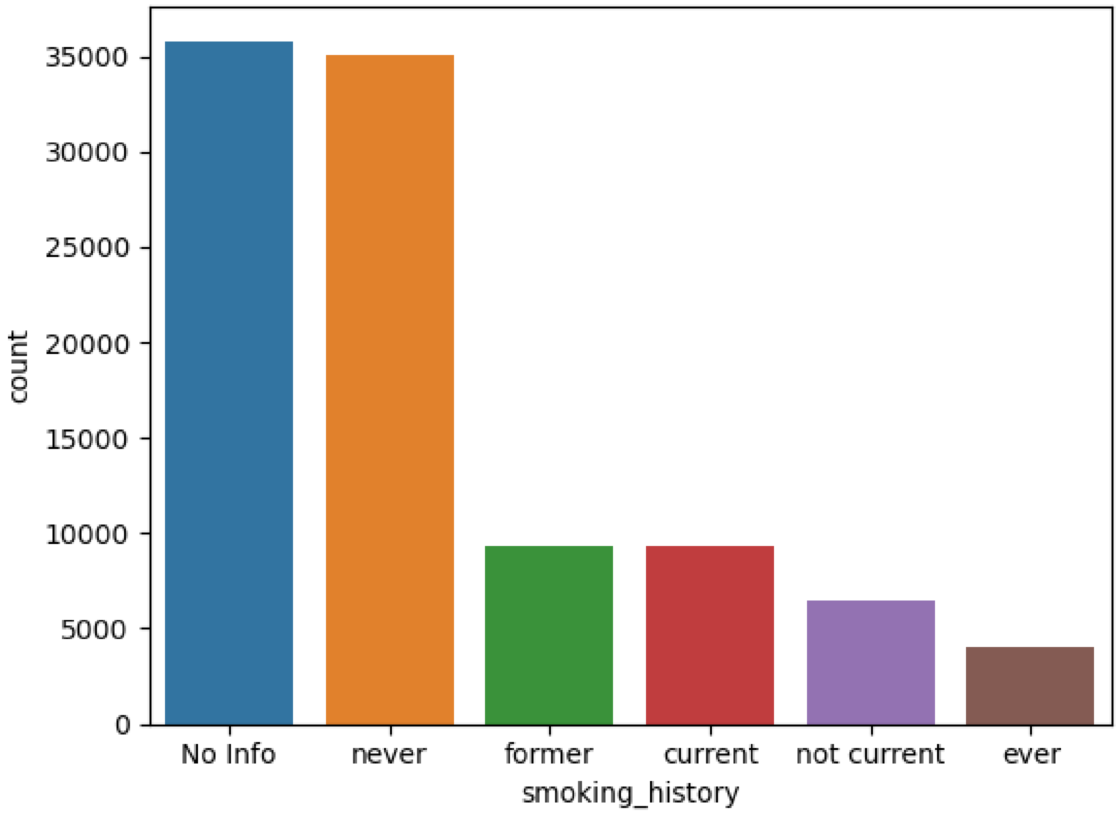
Out[58]:
```
<Axes: xlabel='smoking_history', ylabel='count'>
```



$Method - 1$

In [59]:
```
import seaborn as sns
labels=['No Info','never','former','current','not current','ever']
sns.countplot(data=diabetes_df,x='smoking_history',order=labels)

# long method
```

Out[59]:
```
<Axes: xlabel='smoking_history', ylabel='count'>
```



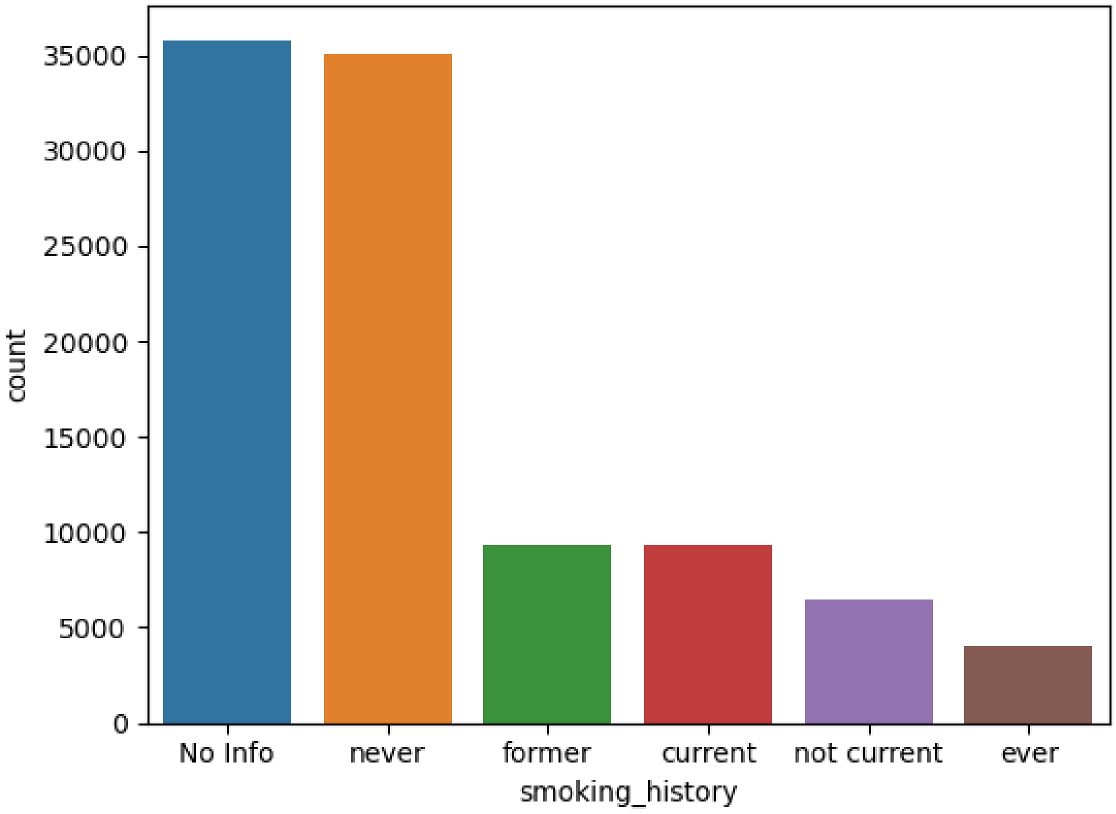
$Method - 2$

In [60]:
```
# make this in order

import seaborn as sns

labels=diabetes_df['smoking_history'].value_counts().keys()
labels

# short method
```

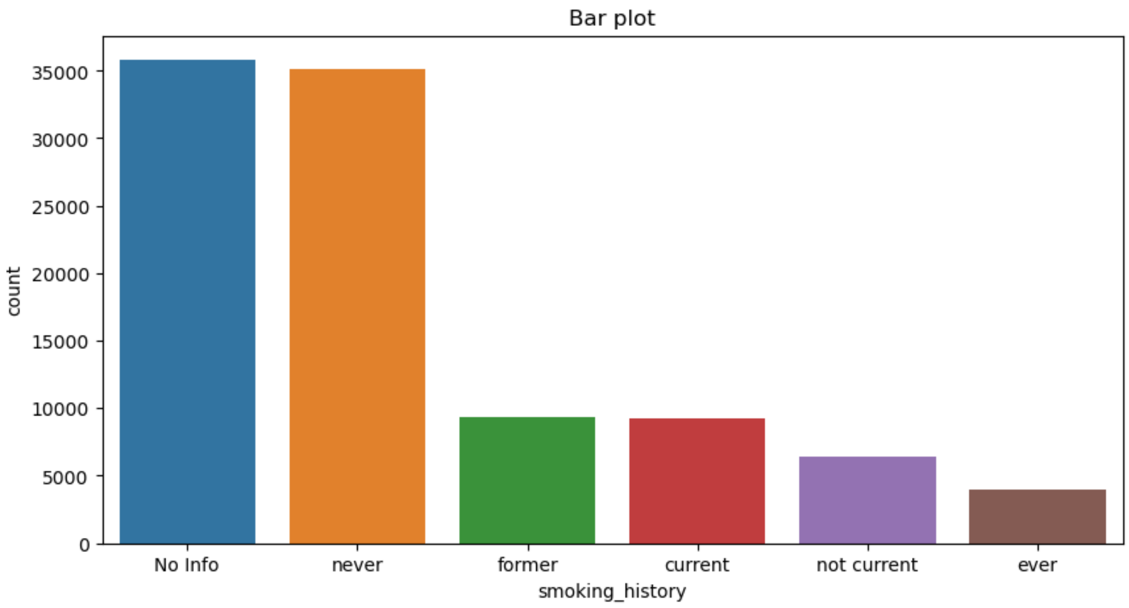
Out[60]:
```
Index(['No Info', 'never', 'former', 'current', 'not current', 'ever'], dt
ype='object', name='smoking_history')
```

In [61]:
```
labels=diabetes_df['smoking_history'].value_counts().keys()
sns.countplot(data=diabetes_df,x='smoking_history',order=labels)
```

Out[61]:
```
<Axes: xlabel='smoking_history', ylabel='count'>
```



In [62]:
```
labels=diabetes_df['smoking_history'].value_counts().keys()
plt.figure(figsize=(10,5))
sns.countplot(data=diabetes_df,x='smoking_history',order=labels)
plt.title("Bar plot")
plt.savefig('smoking_history_seaborn.jpg')
plt.show()
```



Pie Chart

In [63]:
```
diabetes_df['smoking_history'].value_counts(normalize=True)

# normalize= %
# because pie chat always make with the % data only
```

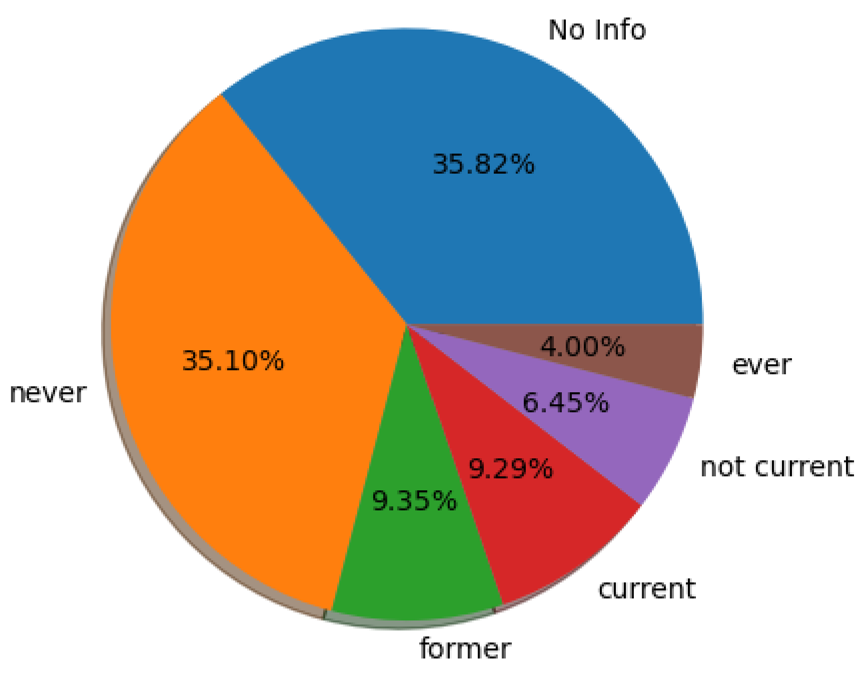
Out[63]:
```
smoking_history
No Info        0.35816
never          0.35095
former         0.09352
current        0.09286
not current    0.06447
ever           0.04004
Name: proportion, dtype: float64
```

In [64]:
```
# keys
# values

keys=diabetes_df['smoking_history'].value_counts(normalize=True).keys()
values=diabetes_df['smoking_history'].value_counts(normalize=True).values
pd.DataFrame(zip(keys,values),columns=['smoking history','Relative Frequenc
```

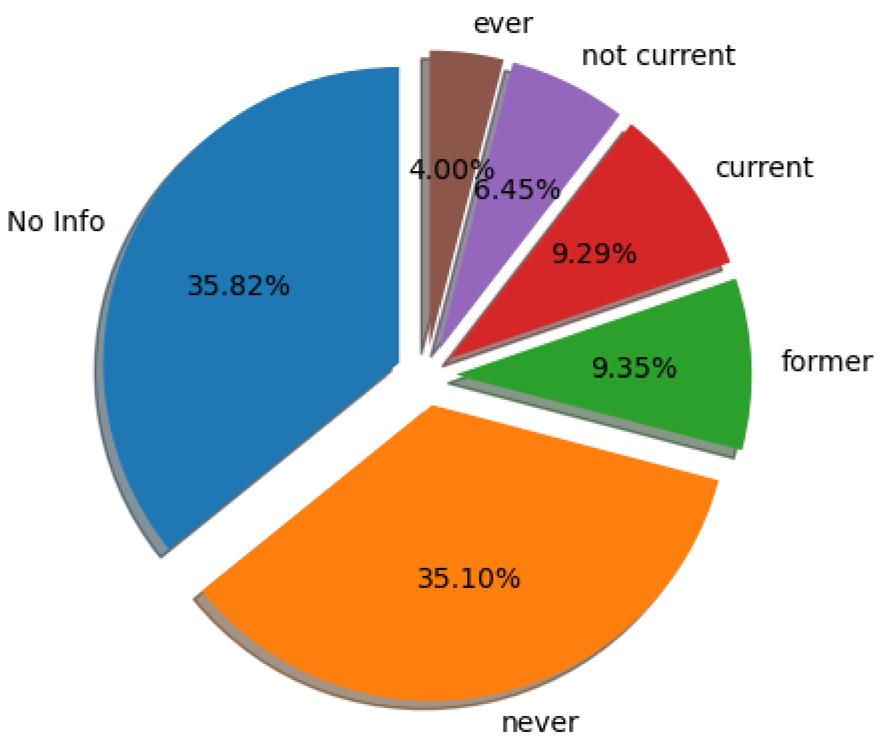
Out[64]:

	smoking history	Relative Frequency
0	No Info	0.35816
1	never	0.35095
2	former	0.09352
3	current	0.09286
4	not current	0.06447
5	ever	0.04004

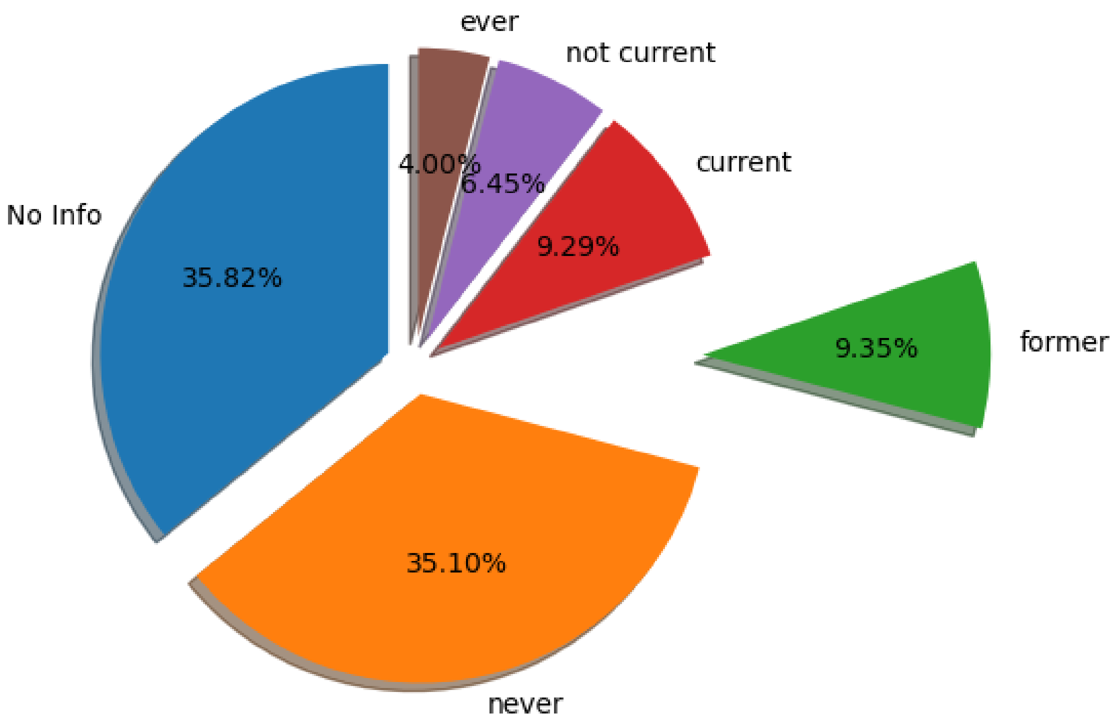
In [65]:
```
plt.pie(x=values,labels=keys,autopct='%0.2f%%',shadow=True)
plt.show()
```



In [66]:
```
# pice the diagrames

plt.pie(x=values,labels=keys,autopct='%0.2f%%',shadow=True,startangle=90,ex
plt.show()
```



In [67]:
```
plt.pie(x=values,labels=keys,autopct='%0.2f%%',shadow=True,startangle=90,ex
plt.show()
```



EDA PART-3

Numerical data analysis

How to read a column

In [68]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [69]:
```
file_path="C:\\Users\\kurre\\OneDrive\\Documents\\Naresh IT\\datafiles\\dia
diabetes_df=pd.read_csv(file_path)
```

In [70]:
```
diabetes_df
```

Out[70]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	
...	...	...	...	...	...	...	...	
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99997	Male	66.0	0	0	former	27.83	5.7	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

100000 rows × 9 columns

Statistical measurements

In [71]:
```
diabetes_df.columns
```

Out[71]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'smoking_histor
y',
       'bmi', 'HbA1c_level', 'blood_glucose_level', 'diabetes'],
      dtype='object')
```

In [72]:
```
diabetes_df['bmi']    # as a series
```

Out[72]:
```
0        25.19
1        27.32
2        27.32
3        23.45
4        20.14
         ...
99995    27.32
99996    17.37
99997    27.83
99998    35.42
99999    22.43
Name: bmi, Length: 100000, dtype: float64
```

In [73]:
```
diabetes_df['bmi'].values
```

Out[73]:
```
array([25.19, 27.32, 27.32, ..., 27.83, 35.42, 22.43])
```

- count
- min
- max
- mean
- median
- standerd deviation

$Method - 1$

Using dictionary to make dataframe

In [74]:
```
dict2={}
count1=round(diabetes_df['bmi'].count(),2)
min1=round(diabetes_df['bmi'].min(),2)
max1=round(diabetes_df['bmi'].max(),2)
mean1=round(diabetes_df['bmi'].mean(),2)
median1=round(diabetes_df['bmi'].median(),2)
std1=round(diabetes_df['bmi'].std(),2)

dict2['count']=count1
dict2['min']=min1
dict2['max']=max1
dict2['mean']=mean1
dict2['median']=median1
dict2['std']=std1

pd.DataFrame(dict2,index=['bmi'])
```

Out[74]:

	count	min	max	mean	median	std
bmi	100000	10.01	95.69	27.32	27.32	6.64

In [75]:
```
dict2={}
count1=round(diabetes_df['bmi'].count(),2)
min1=round(diabetes_df['bmi'].min(),2)
max1=round(diabetes_df['bmi'].max(),2)
mean1=round(diabetes_df['bmi'].mean(),2)
median1=round(diabetes_df['bmi'].median(),2)
std1=round(diabetes_df['bmi'].std(),2)
list2=[count1,min1,max1,mean1,median1,std1]

pd.DataFrame(list2,columns=['bmi'])
```

Out[75]:

	bmi
0	100000.00
1	10.01
2	95.69
3	27.32
4	27.32
5	6.64

In [76]:
```
dict2={}
count1=round(diabetes_df['bmi'].count(),2)
min1=round(diabetes_df['bmi'].min(),2)
max1=round(diabetes_df['bmi'].max(),2)
mean1=round(diabetes_df['bmi'].mean(),2)
median1=round(diabetes_df['bmi'].median(),2)
std1=round(diabetes_df['bmi'].std(),2)
list2=[count1,min1,max1,mean1,median1,std1]

dict2['bmi']=list2
dict2

pd.DataFrame(dict2)
```

Out[76]:

	bmi
0	100000.00
1	10.01
2	95.69
3	27.32
4	27.32
5	6.64

$Method-2$

Using-list

In [77]:
```
count1=round(diabetes_df['bmi'].count(),2)
min1=round(diabetes_df['bmi'].min(),2)
max1=round(diabetes_df['bmi'].max(),2)
mean1=round(diabetes_df['bmi'].mean(),2)
median1=round(diabetes_df['bmi'].median(),2)
std1=round(diabetes_df['bmi'].std(),2)

list3=[count1,min1,max1,mean1,median1,std1]
pd.DataFrame(list3,columns=['bmi'],index=['count','min','max','mean','media
```

Out[77]:

	bmi
count	100000.00
min	10.01
max	95.69
mean	27.32
median	27.32
std	6.64

In [78]:
```
# Step-1 Numerical column list using list comprihention method

dtypes=dict(diabetes_df.dtypes)
num1=[i for i in dtypes if dtypes[i]!='O']
print(num1)
```

```
['age', 'hypertension', 'heart_disease', 'bmi', 'HbA1c_level', 'blood_gluc
ose_level', 'diabetes']
```

In [79]:
```
# column with numerical data

dict3={}
for i in num1:
    count2=round(diabetes_df[i].count(),2)
    min2=round(diabetes_df[i].min(),2)
    max2=round(diabetes_df[i].max(),2)
    mean2=round(diabetes_df[i].mean(),2)
    median2=round(diabetes_df[i].median(),2)
    std2=round(diabetes_df[i].std(),2)

    list4=[count2,min2,max2,mean2,median2,std2]
    dict3[i]=list4

    df=pd.DataFrame(dict3,index=['count','min','max','mean','median','std']

dict3
```

Out[79]:
```
{'age': [100000, 0.08, 80.0, 41.89, 43.0, 22.52],
 'hypertension': [100000, 0, 1, 0.07, 0.0, 0.26],
 'heart_disease': [100000, 0, 1, 0.04, 0.0, 0.19],
 'bmi': [100000, 10.01, 95.69, 27.32, 27.32, 6.64],
 'HbA1c_level': [100000, 3.5, 9.0, 5.53, 5.8, 1.07],
 'blood_glucose_level': [100000, 80, 300, 138.06, 140.0, 40.71],
 'diabetes': [100000, 0, 1, 0.08, 0.0, 0.28]}
```

In [80]:
```
df
```

Out[80]:

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level
count	100000.00	100000.00	100000.00	100000.00	100000.00	100000.00
min	0.08	0.00	0.00	10.01	3.50	80.00
max	80.00	1.00	1.00	95.69	9.00	300.00
mean	41.89	0.07	0.04	27.32	5.53	138.06
median	43.00	0.00	0.00	27.32	5.80	140.00
std	22.52	0.26	0.19	6.64	1.07	40.71

In [81]:
```
# Reading a specific column
# we have a mean method

diabetes_df['age'].mean()
```

Out[81]:
```
41.885856
```

using numpy we draw measurements

In [82]:
```
# np.mean(<specific column data>)
```

In [83]:
```
np.mean(diabetes_df['bmi'])
```

Out[83]:
```
27.3207671
```

In [84]:
```
np.min(diabetes_df['bmi'])
```

Out[84]:
```
10.01
```

In [85]:
```
np.max(diabetes_df['bmi'])
```

Out[85]:
```
95.69
```

In [86]:
```
np.median(diabetes_df['bmi'])
```

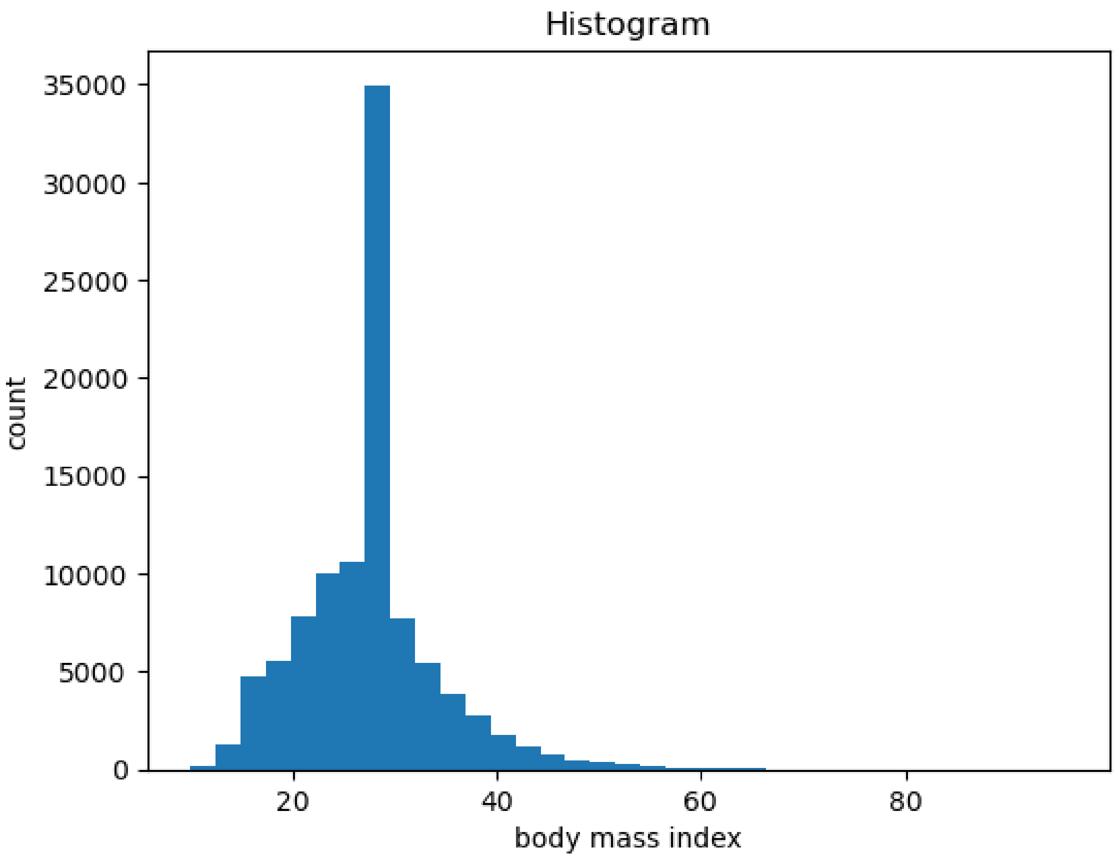
Out[86]:
```
27.32
```

In [87]:
```
np.std(diabetes_df['bmi'])
```

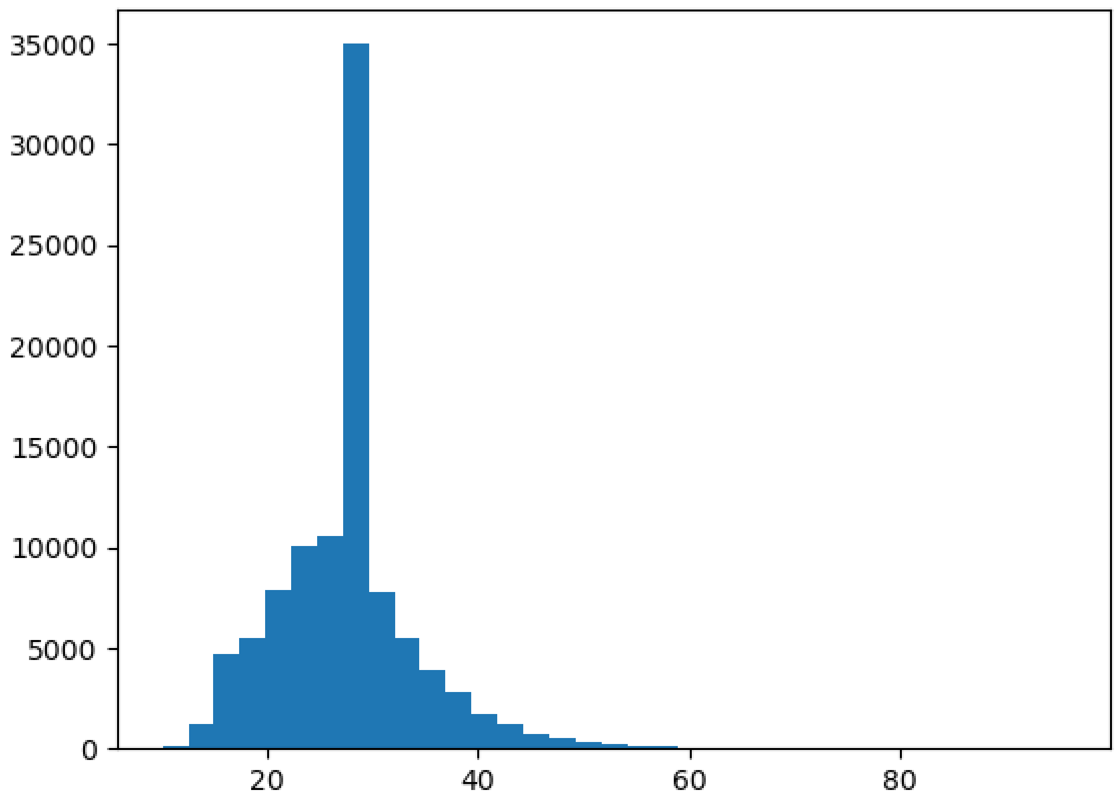
Out[87]:
```
6.636750232649537
```

Histogram

In [88]:
```
plt.hist(diabetes_df['bmi'],bins=35)
plt.title('Histogram')
plt.xlabel('body mass index')
plt.ylabel('count')
plt.show()



# by defualt it will give as 10 intervals
# if you want increase the intervals
# argument name bins
```



In [89]:
```
frequency,interval,n=plt.hist(diabetes_df['bmi'],bins=35)
```



In [90]:
```
len(frequency)
```

Out[90]:
```
35
```

In [91]:
```
len(interval)
```

Out[91]:
```
36
```

We checked the empiricle rule

When data follows a normal *distribution*

- u-1*sigma to u+1*sigma : 68%
- u-2*sigma to u+2*sigma : 95%
- u-3*sigma to u+3*sigma : 99.7%

In [92]:
```
mean1,std1  # from list method (dataframe)
```

Out[92]:
```
(27.32, 6.64)
```

In [93]:
```
###################### 68% ####################

val_minus_1=round(mean1-1*std1,2)
val_plus_1=round(mean1+1*std1,2)
```

```
In [94]: ###################### 95% ####################

         val_minus_2=round(mean1-2*std1,2)
         val_plus_2=round(mean1+2*std1,2)
```

```
In [95]: ###################### 99.7% ####################

         val_minus_3=round(mean1-3*std1,2)
         val_plus_3=round(mean1+3*std1,2)
```

```
In [96]: print(val_minus_1,val_plus_1,val_minus_2,val_plus_2,val_minus_3,val_plus_3)

         20.68 33.96 14.04 40.6 7.4 47.24
```

- 68 percentage of observations have values between [20.68,33.96]
- 95 percentage of observations have values between [14.04,40.6]
- 99.7 percentage of observations have values between [7.4,47.24]

```
In [97]: # 68%

         A=diabetes_df['bmi']>val_minus_1
         A1=diabetes_df['bmi']<val_plus_1
         len(diabetes_df[A&A1])
         len(diabetes_df[A&A1])/len(diabetes_df)

Out[97]: 0.73195
```

```
In [98]: # 95%

         B=diabetes_df['bmi']>val_minus_2
         B1=diabetes_df['bmi']<val_plus_2
         len(diabetes_df[B&B1])
         len(diabetes_df[B&B1])/len(diabetes_df)

Out[98]: 0.95248
```

```
In [99]: # 99.7%

         C=diabetes_df['bmi']>val_minus_3
         C1=diabetes_df['bmi']<val_plus_3
         len(diabetes_df[C&C1])
         len(diabetes_df[C&C1])/len(diabetes_df)

Out[99]: 0.98706
```

```
In [ ]:
```

EDA PART-4 Outlier analysis

In [24]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [25]:
```
file_path="C:\\Users\\kurre\\OneDrive\\Documents\\Naresh IT\\datafiles\\dia
diabetes_df=pd.read_csv(file_path)
```

In [7]:
```
diabetes_df
```

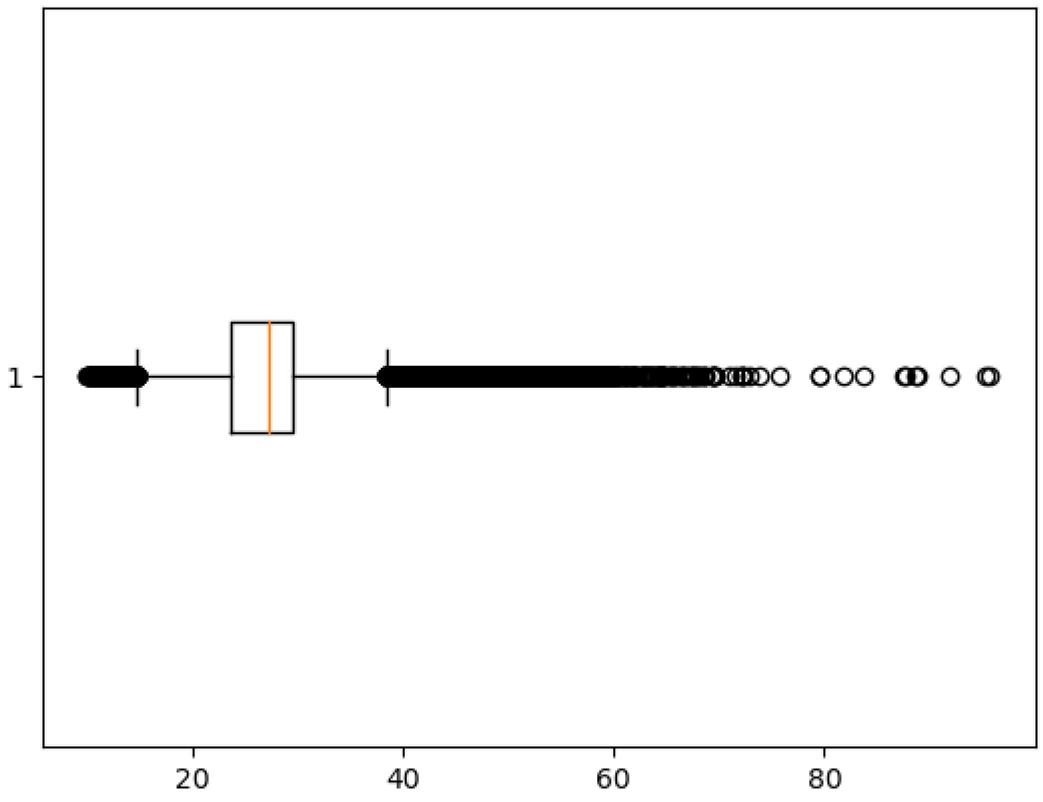
Out[7]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	
...	...	...	...	...	...	...	...	
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99997	Male	66.0	0	0	former	27.83	5.7	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

100000 rows × 9 columns

We draw box plot

In [10]:
```
plt.boxplot(diabetes_df['bmi'],vert=False)
plt.show()
```



How to find outliers

- Removal of outliers
- Impute the outliers with medain value
  - because medain is not impact by Outliers
- Cap the outliers with Q3, which are having more than Q3
- Cap the outliers with Q1, which are having less than Q1

$Steps-$

- Q3+1.5*IQR> and Q1-1.5*IQR
- Step-1: Calculate Q1 Q2 Q3
- Step-2: Calculate IQR=(Q3-Q1)
- Step-3: UB=Q3+1.5*IQR
- Step-4: LB=Q1-1.5*IQR
- Step-5: con1= col>UB
- Step-6: con2= col<LB
- Step-7: con1|con2
- Step-8: col[con1|con2]

```
In [23]: #Step-1: Calculate Q1 Q2 Q3

         q1=np.quantile(diabetes_df['bmi'],0.25)
         q2=np.quantile(diabetes_df['bmi'],0.50)
         q3=np.quantile(diabetes_df['bmi'],0.75)


         #Step-2:Calculate IQR=(Q3-Q1)
         IQR=q3-q1

         #Step-3: UB=Q3+1.5*IQR (UB=upper bound)
         ub=q3+1.5*IQR

         #Step-4: LB=Q1-1.5*IQR  (LB=lower bound)
         lb=q1-1.5*IQR

         #Step-5: con1= col>UB
         #Step-6: con2= col<LB


         con1=diabetes_df['bmi']>ub
         con2=diabetes_df['bmi']<lb

         #step-7 and step-8
         outliers=diabetes_df['bmi'][con1|con2]

         # series into array of values by applying a .values
         outlires_data=outliers.values
         outlires_data


         # we find lenght of outlires
         len(outlires_data)

Out[23]: 7086
```

```
In [30]: ub,lb

Out[30]: (38.504999999999995, 14.705)
```

```
In [18]: # get importent data from data set
         # now we try to get percentages

         len(outlires_data),len(diabetes_df),len(outlires_data)*100/len(diabetes_df)

Out[18]: (7086, 100000, 7.086)
```

How remove the outliers

$Case - 1$

- We have 7086 outliers in 'bmi' column.
- What means we need to remove 7086 rows from entire dataframe.

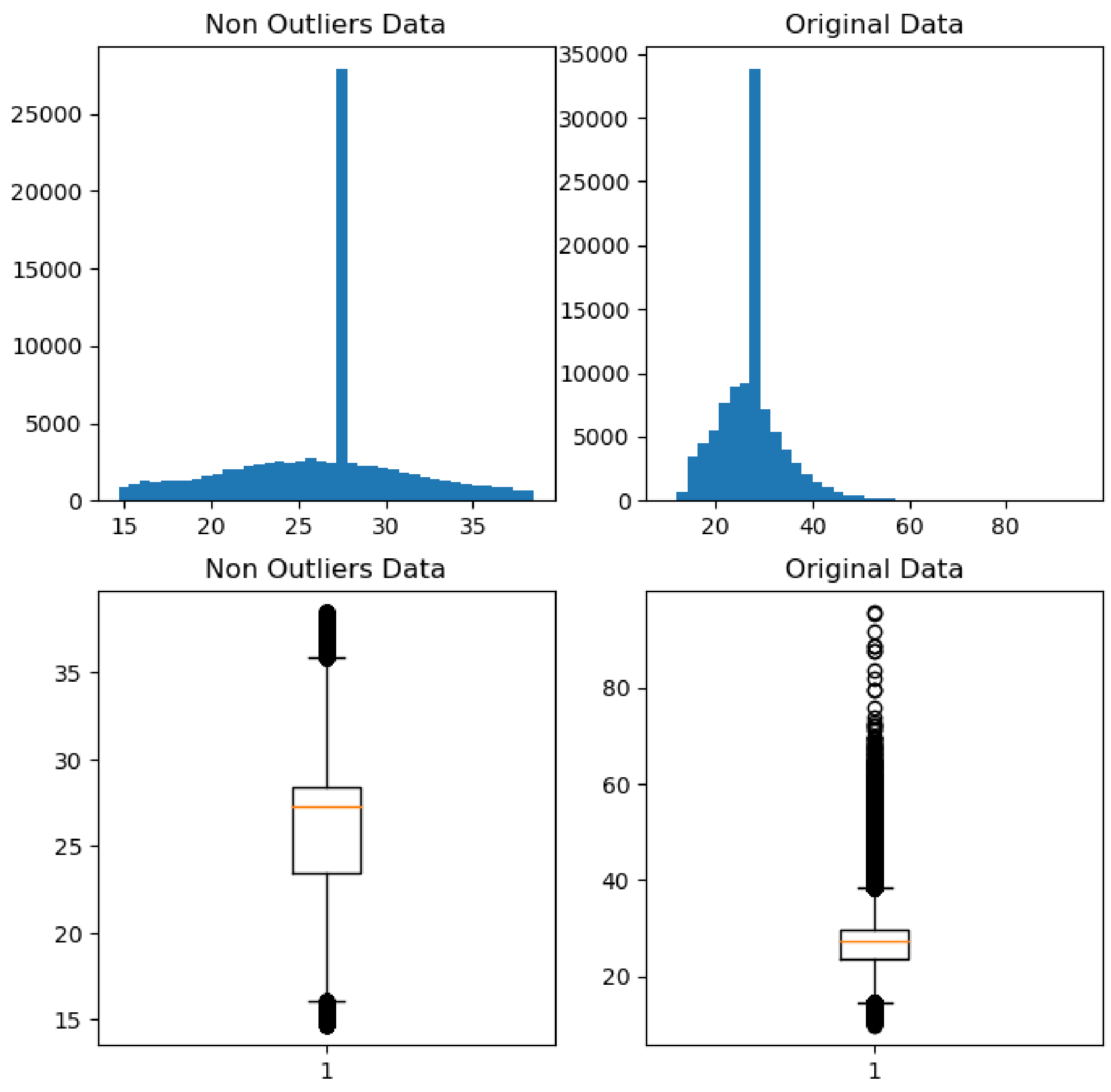
In [27]:
```
q1=np.quantile(diabetes_df['bmi'],0.25)
q2=np.quantile(diabetes_df['bmi'],0.50)
q3=np.quantile(diabetes_df['bmi'],0.75)
IQR=q3-q1
ub=q3+1.5*IQR
lb=q1-1.5*IQR
con1=diabetes_df['bmi']<ub   # here sign is change
con2=diabetes_df['bmi']>lb   # # here sign is change
non_outliers_df=diabetes_df[con1&con2]
non_outliers_df
```

Out[27]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	
...	...	...	...	...	...	...	...	
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99997	Male	66.0	0	0	former	27.83	5.7	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

92914 rows × 9 columns

In [28]:
```
############################# Histogram ######################

plt.figure(figsize=(8,8))
plt.subplot(2,2,1)
plt.title('Non Outliers Data')
plt.hist(non_outliers_df['bmi'],bins=40)

plt.subplot(2,2,2)
plt.title('Original Data')
plt.hist(diabetes_df['bmi'],bins=40)

########################### Box Plot ####################

plt.subplot(2,2,3)
plt.title('Non Outliers Data')
plt.boxplot(non_outliers_df['bmi'])

plt.subplot(2,2,4)
plt.title('Original Data')
plt.boxplot(diabetes_df['bmi'])
plt.show()
```



$Case - 2$

Impute with Median

- We got bmi has 7086 outliers
- we replace those 7086 with median value of 'bmi'
- and using np.where for final data

In [29]:
```
ub,lb
```
Out[29]:
```
(38.504999999999995, 14.705)
```

In [31]:
```
q1=np.quantile(diabetes_df['bmi'],0.25)
q2=np.quantile(diabetes_df['bmi'],0.50)
q3=np.quantile(diabetes_df['bmi'],0.75)
IQR=q3-q1
ub=q3+1.5*IQR
lb=q1-1.5*IQR
con1=diabetes_df['bmi']>ub
con2=diabetes_df['bmi']<lb
outliers=diabetes_df[con1|con2]
outliers
```
Out[31]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
11	Female	54.0	0	0	former	54.70	6.0	
24	Female	4.0	0	0	No Info	13.99	4.0	
39	Female	34.0	0	0	never	56.43	6.2	
59	Female	67.0	0	0	never	63.48	8.8	
93	Male	38.0	0	0	never	55.61	6.5	
...	...	...	...	...	...	...	...	
99933	Female	5.0	0	0	No Info	13.34	6.5	
99948	Female	56.0	1	0	former	39.62	4.5	
99953	Female	59.0	1	0	ever	60.52	3.5	
99960	Female	47.0	0	0	former	45.15	4.0	
99993	Female	40.0	0	0	never	40.69	3.5	

7086 rows × 9 columns

In [32]:
```
len(outliers)
```
Out[32]:
```
7086
```

In [34]:
```python
new_data=[]
for i in diabetes_df['bmi']:
    if i>ub or i<lb:
        new_data.append(diabetes_df['bmi'].median)
    else:
        new_data.append(i)
len(new_data)




# We are iterate through 'bmi' data
# if any datapoint >ub or <lb means it is a outliers so in that postition
#                   we are keeping medain value of the column

# otherwise we are keeping the same value
```

Out[34]:
```
100000
```

np.where

In [ ]:
```python
# steps to use np.where

# step-1: write the condition
# step-2: True value: Medain value
# Step-3: False value: same column values
# Step-4: implment np.where(<con1>,<True_vale>,<False_vale>)
# Step-5: Overwrite in the same column name
# Step-6: Draw the boxplot for 'bmi'
# Step-7: Draw the histogram  'bmi'
```

In [35]:
```python
con1=diabetes_df['bmi']>ub
con2=diabetes_df['bmi']<lb
con=con1|con2
wage_median=diabetes_df['bmi'].median()
diabetes_df['bmi']=np.where(con,wage_median,diabetes_df['bmi'])
```

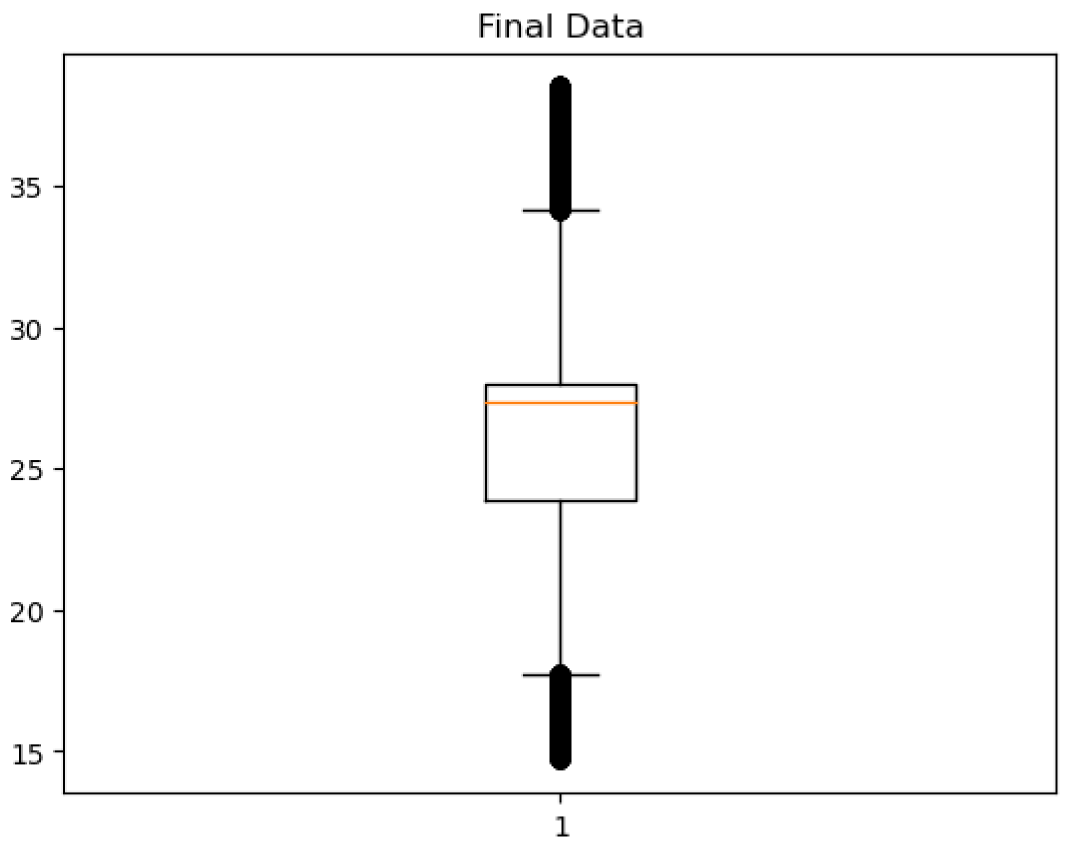
In [38]:
```python
diabetes_df['bmi']
```

Out[38]:
```
0        25.19
1        27.32
2        27.32
3        23.45
4        20.14
         ...
99995    27.32
99996    17.37
99997    27.83
99998    35.42
99999    22.43
Name: bmi, Length: 100000, dtype: float64
```

In [39]:
```python
len(diabetes_df['bmi'])
```

Out[39]:
```
100000
```

In [41]:
```
# Now draw box plot

plt.boxplot(diabetes_df['bmi'])
plt.title('Final Data')
plt.show()
```



EDA PART-5: Bi variate and multivariate analysis

In [42]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [43]:
```
file_path='C:\\Users\\kurre\\OneDrive\\Documents\\Naresh IT\\datafiles\\dia
diabetes_df=pd.read_csv(file_path)
```

In [44]:
```
diabetes_df
```

Out[44]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	
...	...	...	...	...	...	...	...	
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99997	Male	66.0	0	0	former	27.83	5.7	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

100000 rows × 9 columns

We draw two categorical columns analysis

In [46]:
```
# smoking_history value counts
diabetes_df['smoking_history'].value_counts()
```

Out[46]:
```
smoking_history
No Info        35816
never          35095
former          9352
current         9286
not current     6447
ever            4004
Name: count, dtype: int64
```

In [47]:
```
# gender value counts
diabetes_df['gender'].value_counts()
```

Out[47]:
```
gender
Female    58552
Male      41430
Other        18
Name: count, dtype: int64
```

In [ ]:
```
# 1. out of all male how many current smoking
# 2. out of all female how many current somking
```

In [48]:
```
con1=diabetes_df['smoking_history']=='current'
con2=diabetes_df['gender']=='Male'
con=con1&con2
len(diabetes_df[con])
```

Out[48]:
```
4228
```

In [50]:
```
diabetes_df['smoking_history'].unique()
diabetes_df['smoking_history'].value_counts().keys()
```

Out[50]:
```
Index(['No Info', 'never', 'former', 'current', 'not current', 'ever'], dt
ype='object', name='smoking_history')
```

In [51]:
```
# Generalised

lables=diabetes_df['smoking_history'].unique()
male_count=[]
female_count=[]
for i in lables:
    con1=diabetes_df['smoking_history']==i
    con2=diabetes_df['gender']=='Male'
    con3=diabetes_df['gender']=='Female'
    male_count.append(len(diabetes_df[con1&con2]))
    female_count.append(len(diabetes_df[con1&con3]))

pd.DataFrame(zip(lables,male_count,female_count),
             columns=['smoking_history','Male','Female'])
```

Out[51]:

	smoking_history	Male	Female
0	never	12223	22869
1	No Info	16110	19700
2	current	4228	5058
3	former	4578	4774
4	ever	1765	2238
5	not current	2526	3913

In [52]:
```
pd.DataFrame(zip(lables,male_count,female_count),
             columns=['smoking_history','Male','Female']).set_index('smokin
```

Out[52]:

	Male	Female
smoking_history		
never	12223	22869
No Info	16110	19700
current	4228	5058
former	4578	4774
ever	1765	2238
not current	2526	3913

pd.crosstab

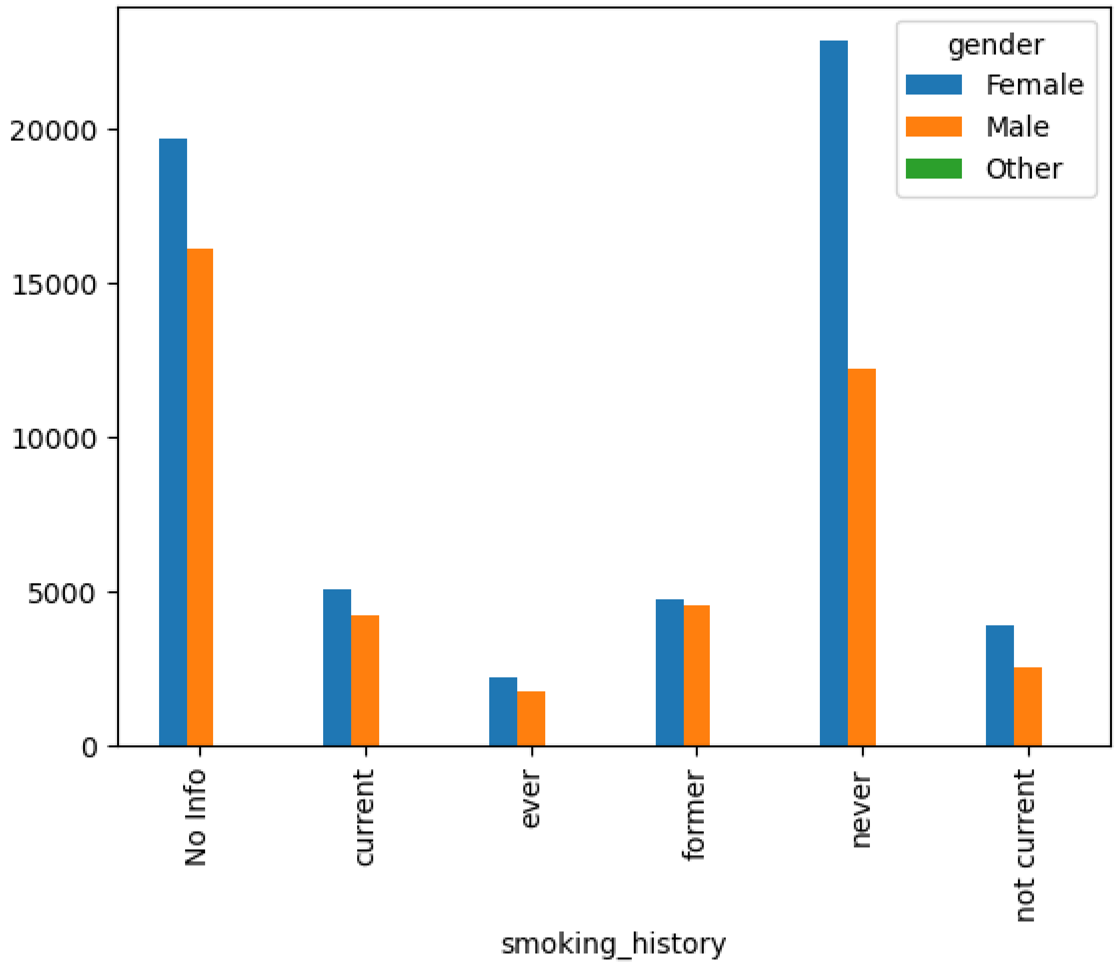
In [53]:
```
col1=diabetes_df['smoking_history']
col2=diabetes_df['gender']

result1=pd.crosstab(col1,col2)
result1
```

Out[53]:

gender	Female	Male	Other
smoking_history			
No Info	19700	16110	6
current	5058	4228	0
ever	2238	1765	1
former	4774	4578	0
never	22869	12223	3
not current	3913	2526	8

Draw the plots

In [54]:
```
result1.plot(kind='bar')
plt.show()
```



We repeated multiple columns

In [ ]:
```
# in this data set only 2 columns are
#         categorical so we can not use multiple columns
```

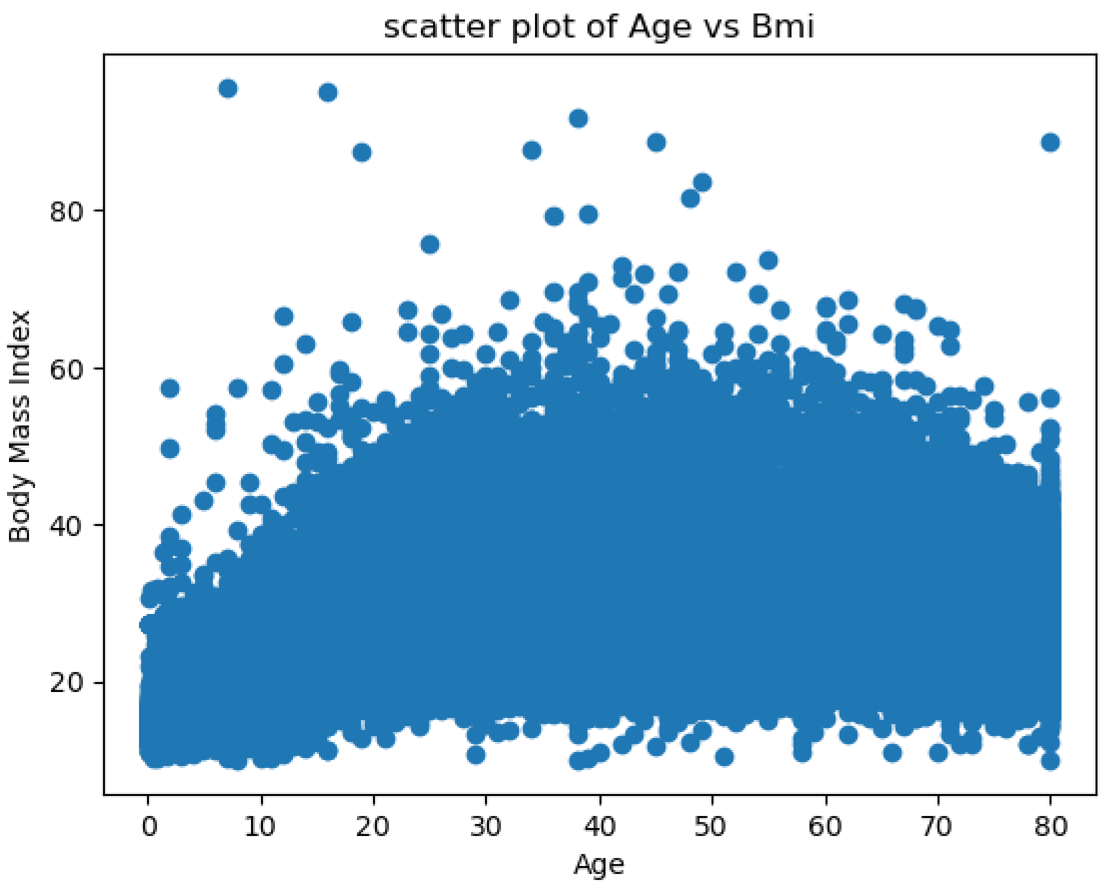
We draw two numerical columns analysis

In [56]:
```
dtypes=dict(diabetes_df.dtypes)
num10=[i for i in dtypes if dtypes[i]!='O']
num10
```

Out[56]:
```
['age',
 'hypertension',
 'heart_disease',
 'bmi',
 'HbA1c_level',
 'blood_glucose_level',
 'diabetes']
```

Scatter plot

In [61]:
```
col1=diabetes_df['age']
col2=diabetes_df['bmi']
plt.scatter(col1,col2)
plt.title('scatter plot of Age vs Bmi')
plt.xlabel('Age')
plt.ylabel('Body Mass Index')
plt.show()
```



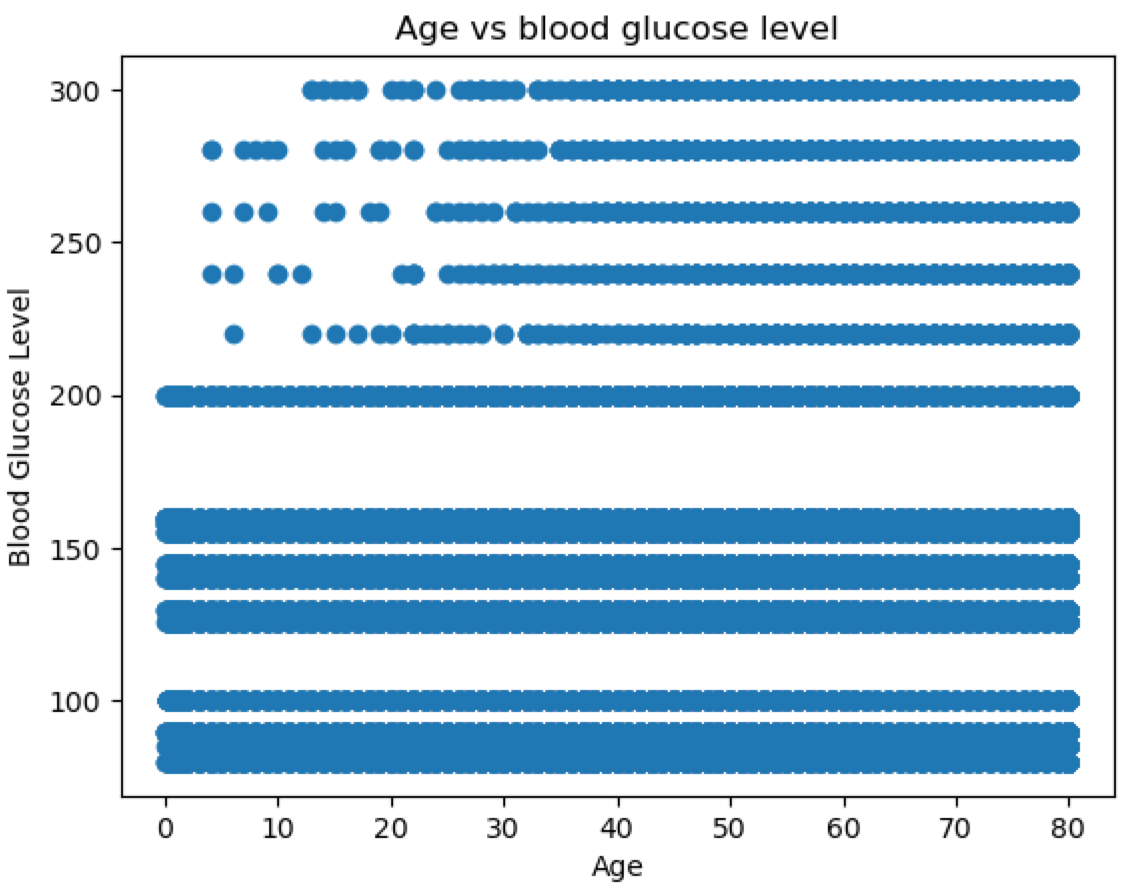
correlation data

In [64]:
```
diabetes_df.corr(numeric_only=True)
```

Out[64]:

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glu
age	1.000000	0.251171	0.233354	0.337396	0.101354	
hypertension	0.251171	1.000000	0.121262	0.147666	0.080939	
heart_disease	0.233354	0.121262	1.000000	0.061198	0.067589	
bmi	0.337396	0.147666	0.061198	1.000000	0.082997	
HbA1c_level	0.101354	0.080939	0.067589	0.082997	1.000000	
blood_glucose_level	0.110672	0.084429	0.070066	0.091261	0.166733	
diabetes	0.258008	0.197823	0.171727	0.214357	0.400660	

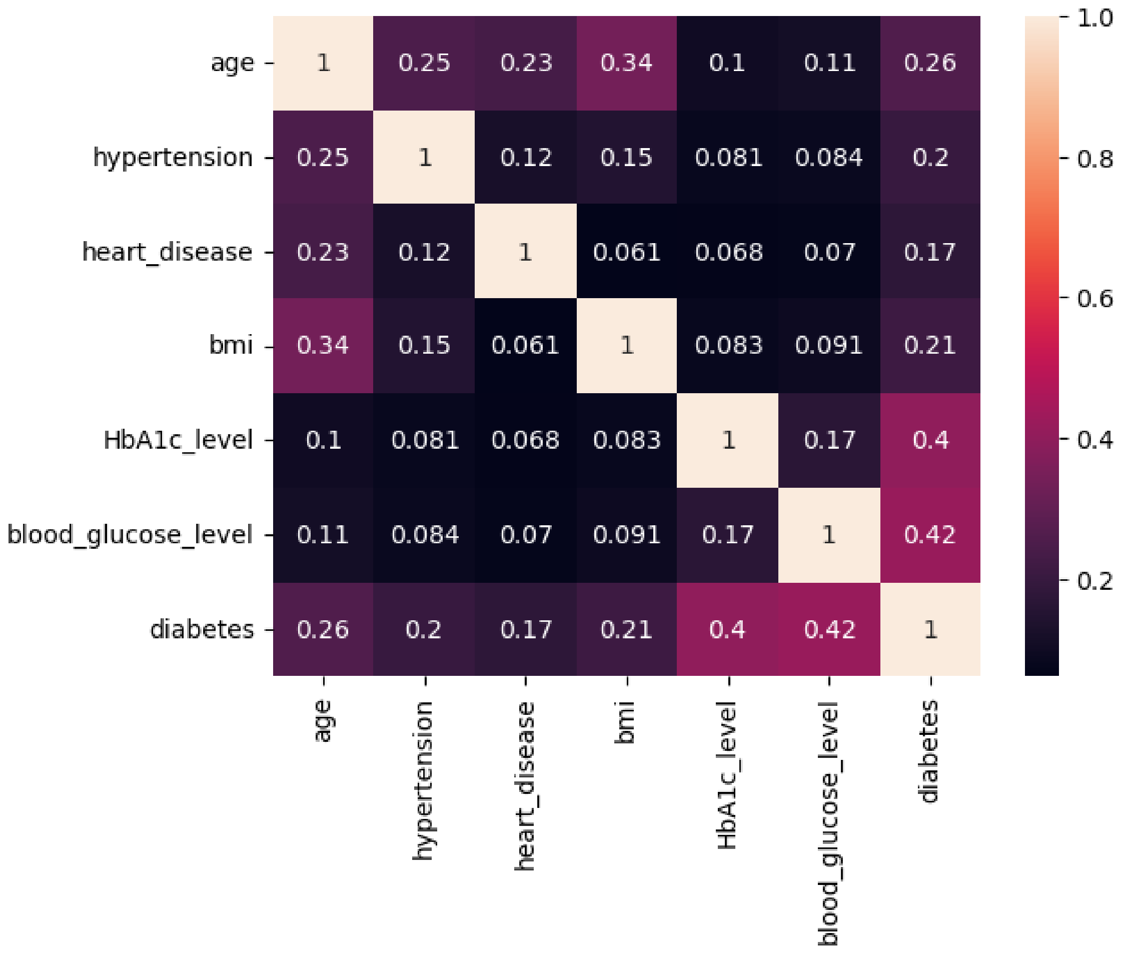
In [79]:
```
plt.scatter(diabetes_df['age'],diabetes_df['blood_glucose_level'])
plt.title('Age vs blood glucose level')
plt.xlabel('Age')
plt.ylabel('Blood Glucose Level')
plt.show()
```



Matrix,Heatmap

In [ ]:
```
# matrix
# showing values in a matrix
# showing values in a picture: Heatmap
```

In [81]:
```
corr_data=diabetes_df.corr(numeric_only=True)
sns.heatmap(corr_data,annot=True)
plt.show()
```



	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
age	1	0.25	0.23	0.34	0.1	0.11	0.26
hypertension	0.25	1	0.12	0.15	0.081	0.084	0.2
heart_disease	0.23	0.12	1	0.061	0.068	0.07	0.17
bmi	0.34	0.15	0.061	1	0.083	0.091	0.21
HbA1c_level	0.1	0.081	0.068	0.083	1	0.17	0.4
blood_glucose_level	0.11	0.084	0.07	0.091	0.17	1	0.42
diabetes	0.26	0.2	0.17	0.21	0.4	0.42	1

EDA PART-6: Categorical to Numerical

In [82]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [84]:
```
# Read the data

file_path='C:\\Users\\kurre\\OneDrive\\Documents\\Naresh IT\\datafiles\\dia
diabetes_df=pd.read_csv(file_path)
diabetes_df
```

Out[84]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	bloo
0	Female	80.0	0	1	never	25.19	6.6	
1	Female	54.0	0	0	No Info	27.32	6.6	
2	Male	28.0	0	0	never	27.32	5.7	
3	Female	36.0	0	0	current	23.45	5.0	
4	Male	76.0	1	1	current	20.14	4.8	
...	...	...	...	...	...	...	...	
99995	Female	80.0	0	0	No Info	27.32	6.2	
99996	Female	2.0	0	0	No Info	17.37	6.5	
99997	Male	66.0	0	0	former	27.83	5.7	
99998	Female	24.0	0	0	never	35.42	4.0	
99999	Female	57.0	0	0	current	22.43	6.6	

100000 rows × 9 columns

Lable encoder

- labelencoder is a method from sklearn
- Under sklearn we have sub modules
- One of the submodule: preprocessing
- Any sklearn packages we have only 3 steps
- step-1: read the packages
- step-2: save the packages
- step-3: apply fit transform

In [85]:
```
# step-1
from sklearn.preprocessing import LabelEncoder

# step-2
leb=LabelEncoder()

# step-3
diabetes_df['gender']=leb.fit_transform(diabetes_df['gender'])
diabetes_df['smoking_history']=leb.fit_transform(diabetes_df['smoking_histo
print(diabetes_df[['smoking_history','gender']].head(10))
```

```
   smoking_history  gender
0                4       0
1                0       0
2                4       1
3                1       0
4                1       1
5                4       0
6                4       0
7                0       0
8                4       1
9                4       0
```

In [86]:
```
print(diabetes_df['smoking_history'][:5])
leb.inverse_transform(diabetes_df['smoking_history'])
```

```
0    4
1    0
2    4
3    1
4    1
Name: smoking_history, dtype: int32
```

Out[86]:
```
array(['never', 'No Info', 'never', ..., 'former', 'never', 'current'],
      dtype=object)
```

EDA PART-7: Normalization and Standardization

MinMaxScalar

$$X_{new} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$

steps

- MinMaxScalar is a method from sklearn preprocessing
- Read the packages

- Save the package
- Apply fit transform

In [87]:
```
# import packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [88]:
```
# Read the data

file_path='C:\\Users\\kurre\\OneDrive\\Documents\\Naresh IT\\datafiles\\dia
diabetes_df=pd.read_csv(file_path)
diabetes_df
```

Out[88]:
```
       gender   age  hypertension  heart_disease  smoking_history    bmi  HbA1c_level  bloo
    0  Female  80.0             0              1            never  25.19          6.6
    1  Female  54.0             0              0          No Info  27.32          6.6
    2    Male  28.0             0              0            never  27.32          5.7
    3  Female  36.0             0              0          current  23.45          5.0
    4    Male  76.0             1              1          current  20.14          4.8
  ...     ...   ...           ...            ...              ...    ...          ...
99995  Female  80.0             0              0          No Info  27.32          6.2
99996  Female   2.0             0              0          No Info  17.37          6.5
99997    Male  66.0             0              0           former  27.83          5.7
99998  Female  24.0             0              0            never  35.42          4.0
99999  Female  57.0             0              0          current  22.43          6.6

100000 rows × 9 columns
```

In [ ]:
```
# step-1: calcaulate min value of smoking_history= min_val
# step-2: calculate max value of smoking_history = max_val
# step-3: Dr= max_val-min_val
# step-4: Nr= smoking_history-min_val
# step-5: Nr/Dr
```

In [93]:
```
file_path='C:\\Users\\kurre\\OneDrive\\Documents\\Naresh IT\\datafiles\\dia
diabetes_df=pd.read_csv(file_path)
diabetes_df



min_val=diabetes_df['bmi'].min()
max_val=diabetes_df['bmi'].max()
dr=max_val-min_val
nr=diabetes_df['bmi']-min_val
diabetes_df['bmi_norm']=nr/dr
```

In [94]:
```
diabetes_df[['bmi','bmi_norm']]
```

Out[94]:
```
         bmi   bmi_norm
    0  25.19   0.177171
    1  27.32   0.202031
    2  27.32   0.202031
    3  23.45   0.156863
    4  20.14   0.118231
  ...    ...        ...
99995  27.32   0.202031
99996  17.37   0.085901
99997  27.83   0.207983
99998  35.42   0.296569
99999  22.43   0.144958

100000 rows × 2 columns
```

In [98]:
```
#step-1

from sklearn.preprocessing import MinMaxScaler

#step-2

v2=MinMaxScaler()

# step-3

diabetes_df['bmi_norm']=v2.fit_transform(diabetes_df[['bmi']])
```

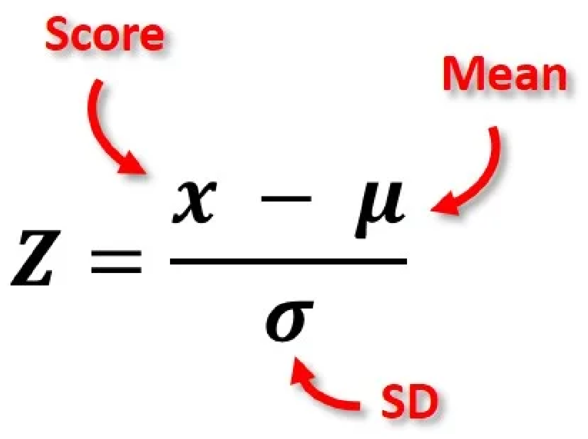
In [99]:
```
diabetes_df[['bmi_norm','bmi']]
```

Out[99]:

	bmi_norm	bmi
0	0.177171	25.19
1	0.202031	27.32
2	0.202031	27.32
3	0.156863	23.45
4	0.118231	20.14
...	...	...
99995	0.202031	27.32
99996	0.085901	17.37
99997	0.207983	27.83
99998	0.296569	35.42
99999	0.144958	22.43

100000 rows × 2 columns

Z-score



$$Z = \frac{x - \mu}{\sigma}$$

In [ ]:
```
# step-1: calculate mean
# step-2: calculate std
# step-3: Nr= x-mean
# step-4: Nr/Std
```

In [100]:
```
mean_val=diabetes_df['bmi'].mean()
std_val=diabetes_df['bmi'].std()
nr=diabetes_df['bmi']-mean_val
diabetes_df['bmi_zscore']=nr/std_val
```

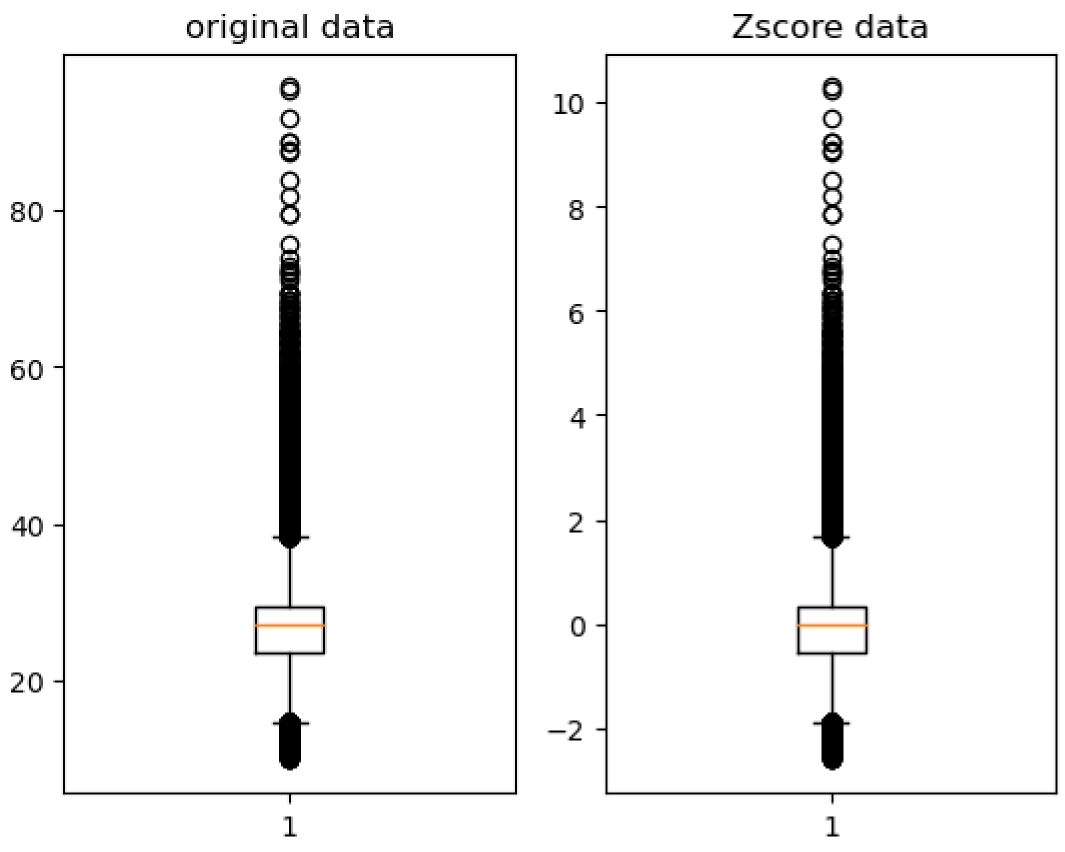
In [101]:
```
diabetes_df[['bmi','bmi_zscore']]
```

Out[101]:

	bmi	bmi_zscore
0	25.19	-0.321054
1	27.32	-0.000116
2	27.32	-0.000116
3	23.45	-0.583229
4	20.14	-1.081965
...	...	...
99995	27.32	-0.000116
99996	17.37	-1.499336
99997	27.83	0.076729
99998	35.42	1.220355
99999	22.43	-0.736918

100000 rows × 2 columns

In [104]:
```
plt.subplot(1,2,1)
plt.boxplot(diabetes_df['bmi'])
plt.title('original data')

plt.subplot(1,2,2)
plt.boxplot(diabetes_df['bmi_zscore'])
plt.title('Zscore data')
plt.show()
```



```
In [ ]:
```

```
In [ ]:
```