Importing liabraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing dataset

In [5]:
```
df=pd.read_csv("diabetes_prediction_dataset.csv")
```

In [6]:
```
df
```

Out[6]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	di
0	Female	80.0	0	1	never	25.19	6.6	140	
1	Female	54.0	0	0	No Info	27.32	6.6	80	
2	Male	28.0	0	0	never	27.32	5.7	158	
3	Female	36.0	0	0	current	23.45	5.0	155	
4	Male	76.0	1	1	current	20.14	4.8	155	
...	...	...	...	...	...	...	...	...	...
99995	Female	80.0	0	0	No Info	27.32	6.2	90	
99996	Female	2.0	0	0	No Info	17.37	6.5	100	
99997	Male	66.0	0	0	former	27.83	5.7	155	
99998	Female	24.0	0	0	never	35.42	4.0	100	
99999	Female	57.0	0	0	current	22.43	6.6	90	

100000 rows × 9 columns

In [7]:
```
df.head()
```

Out[7]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabete
0	Female	80.0	0	1	never	25.19	6.6	140	
1	Female	54.0	0	0	No Info	27.32	6.6	80	
2	Male	28.0	0	0	never	27.32	5.7	158	
3	Female	36.0	0	0	current	23.45	5.0	155	
4	Male	76.0	1	1	current	20.14	4.8	155	

In [8]:
```
df.tail()
```

Out[8]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	dia
99995	Female	80.0	0	0	No Info	27.32	6.2	90	
99996	Female	2.0	0	0	No Info	17.37	6.5	100	
99997	Male	66.0	0	0	former	27.83	5.7	155	
99998	Female	24.0	0	0	never	35.42	4.0	100	
99999	Female	57.0	0	0	current	22.43	6.6	90	

In [10]:
```
df.shape
```

Out[10]:
```
(100000, 9)
```

In [11]:
```
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
```

checking whether dataset contains any null values or not

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
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
```

checking whether dataset contains any duplicate values or not

In [13]:
```
df.duplicated()
```

Out[13]:
```
0        False
1        False
2        False
3        False
4        False
         ...
99995     True
99996    False
99997    False
99998    False
99999    False
Length: 100000, dtype: bool
```

In [30]:
```
df.drop_duplicates(inplace=True)
```

In [31]:
```
df.duplicated()
```

Out[31]:
```
0        False
1        False
2        False
3        False
4        False
         ...
99993    False
99994    False
99997    False
99998    False
99999    False
Length: 82200, dtype: bool
```

- We are able to remove all the dupicated values from dataset

Converting float values into int values

In [20]:
```
df["age"]=df["age"].astype('int')
```

In [21]:
```
df["bmi"]=df["bmi"].astype('int')
```

In [22]:
```
df["HbA1c_level"]=df["HbA1c_level"].astype('int')
```

In [23]:
```
df.head()
```

Out[23]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80	0	1	never	25	6	140	0
1	Female	54	0	0	No Info	27	6	80	0
2	Male	28	0	0	never	27	5	158	0
3	Female	36	0	0	current	23	5	155	0
4	Male	76	1	1	current	20	4	155	0

separation of numeric & categorical Columns

In [32]:
```
numeric_columns=df.columns[df.dtypes!="object"]
categorical_columns=df.columns[df.dtypes=="object"]
```

In [33]:
```
numeric_columns
```

Out[33]:
```
Index(['age', 'hypertension', 'heart_disease', 'bmi', 'HbA1c_level',
       'blood_glucose_level', 'diabetes'],
      dtype='object')
```

In [34]:
```
categorical_columns
```

Out[34]:
```
Index(['gender', 'smoking_history'], dtype='object')
```

separation of numeric and categorical data

In [35]:
```
numeric_data=df.select_dtypes(include=[np.number])
categorical_data=df.select_dtypes(exclude=[np.number])
```

In [36]:
```
numeric_data
```

Out[36]:

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
0	80	0	1	25	6	140	0
1	54	0	0	27	6	80	0
2	28	0	0	27	5	158	0
3	36	0	0	23	5	155	0
4	76	1	1	20	4	155	0
...	...	...	...	...	...	...	...
99993	40	0	0	40	3	155	0
99994	36	0	0	24	4	145	0
99997	66	0	0	27	5	155	0
99998	24	0	0	35	4	100	0
99999	57	0	0	22	6	90	0

82200 rows × 7 columns

In [37]:
```
categorical_data
```

Out[37]:

	gender	smoking_history
0	Female	never
1	Female	No Info
2	Male	never
3	Female	current
4	Male	current
...	...	...
99993	Female	never
99994	Female	No Info
99997	Male	former
99998	Female	never
99999	Female	current

82200 rows × 2 columns

In [39]:
```
numeric_data.shape
```

Out[39]:
```
(82200, 7)
```

In [40]:
```
categorical_data.shape
```

Out[40]:
```
(82200, 2)
```

statistical Analysis

In [41]:
```
# describe() method returns description of the data in the DataFrame (i.e. count, mean,
df.describe()
```

Out[41]:

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabete
count	82200.000000	82200.000000	82200.000000	82200.000000	82200.000000	82200.000000	82200.00000
mean	43.185523	0.089075	0.046898	27.307591	5.108285	139.072689	0.10193
std	21.933089	0.284854	0.211421	6.949149	1.136180	41.775890	0.30256
min	0.000000	0.000000	0.000000	10.000000	3.000000	80.000000	0.00000
25%	26.000000	0.000000	0.000000	23.000000	4.000000	100.000000	0.00000
50%	45.000000	0.000000	0.000000	27.000000	5.000000	140.000000	0.00000
75%	60.000000	0.000000	0.000000	30.000000	6.000000	159.000000	0.00000
max	80.000000	1.000000	1.000000	95.000000	9.000000	300.000000	1.00000

In [42]:
```
numeric_data
```

Out[42]:

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
0	80	0	1	25	6	140	0
1	54	0	0	27	6	80	0
2	28	0	0	27	5	158	0
3	36	0	0	23	5	155	0
4	76	1	1	20	4	155	0
...	...	...	...	...	...	...	...
99993	40	0	0	40	3	155	0
99994	36	0	0	24	4	145	0
99997	66	0	0	27	5	155	0
99998	24	0	0	35	4	100	0
99999	57	0	0	22	6	90	0

82200 rows × 7 columns

mean, median ,mode

In [44]:
```
np.mean(numeric_data)
```

```
C:\Users\SHREE\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3430: FutureWarning: In a future version, DataFrame.mean(axis=None) will return a scalar mean over the entire DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or just 'frame.mean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

Out[44]:
```
age                   43.185523
hypertension           0.089075
heart_disease          0.046898
bmi                   27.307591
HbA1c_level            5.108285
blood_glucose_level  139.072689
diabetes               0.101934
dtype: float64
```

In [45]:
```
np.median(numeric_data)
```

Out[45]:
```
5.0
```

In [46]:
```
import statistics

statistics.mode(numeric_data)
```

Out[46]:
```
'age'
```

dispersion

In [70]:
```
q1=np.percentile(numeric_data,[25])
```

In [72]:
```
q3=np.percentile(numeric_data,[75])
```

In [68]:
```
np.percentile(numeric_data,[25,50,75,100])
```

Out[68]:
```
array([  0.,   5.,  35., 300.])
```

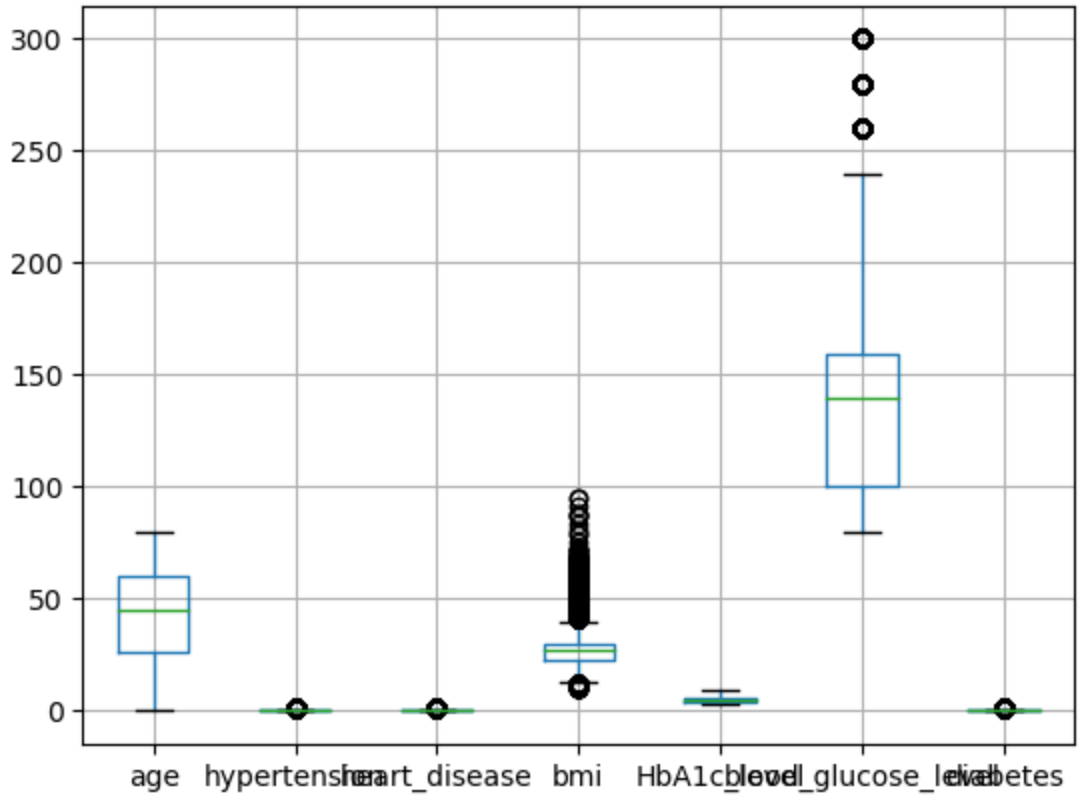
In [73]:
```
iqr=q3-q1
```

In [76]:
```
upper_limit = percentile75 + 1.5 * iqr
lower_limit = percentile25 - 1.5 * iqr
```

BOXPLOT visualization to check whether any outliers present or not

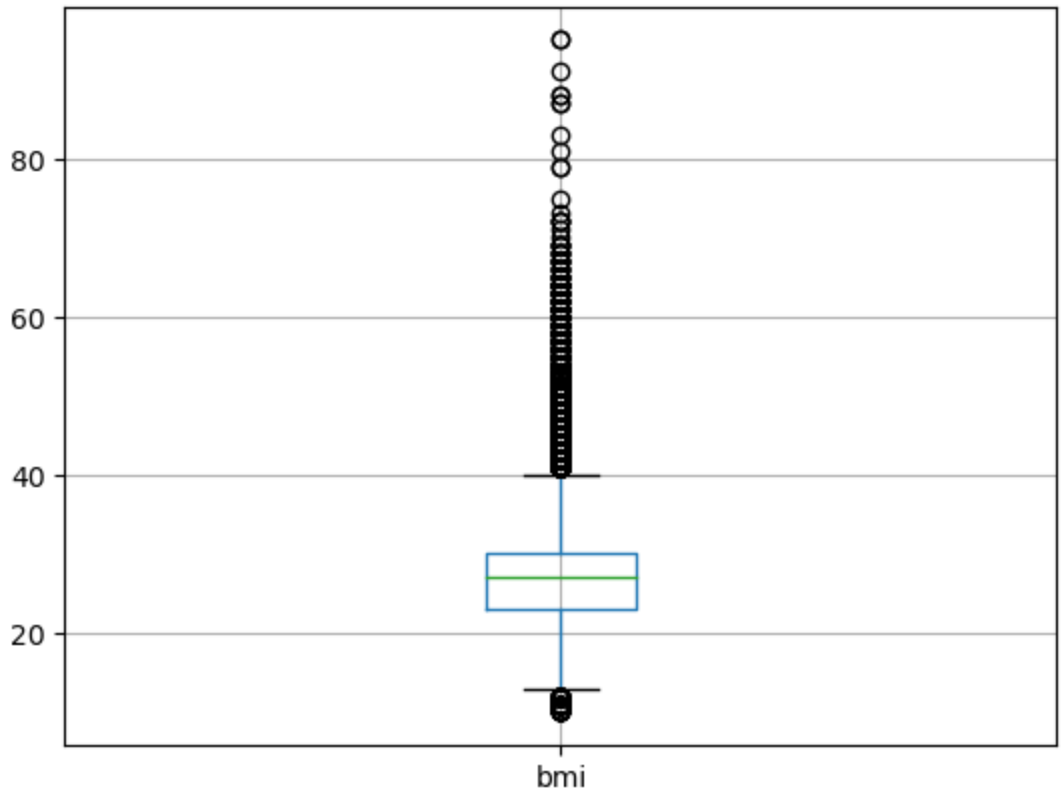
In [88]:
```
import matplotlib.pyplot as plt

numeric_data.boxplot()
```

Out[88]:
```
<Axes: >
```

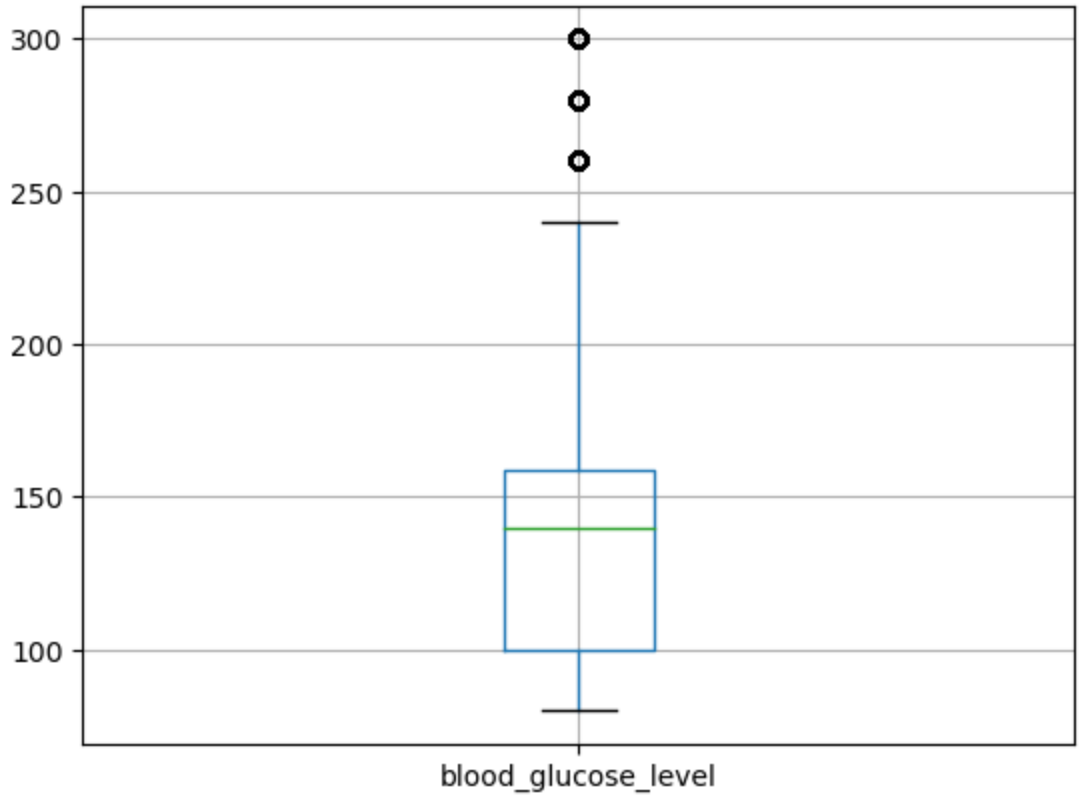


-here we can see that column bmi & column blood_glucose_level contains outliers

In [90]:
```
numeric_data.boxplot(column="bmi")
plt.show()
```



In [98]:
```
numeric_data.boxplot(column="blood_glucose_level")
plt.show()
```



In [91]:
```
def remove_outlier(col):
    sorted(col)
    q1,q3 = col.quantile([0.25,0.75])
    IQR= q3-q1
    lwr_bound=q1-(1.5*IQR)
    upr_bound=q3+(1.5*IQR)
    return lwr_bound,upr_bound
```

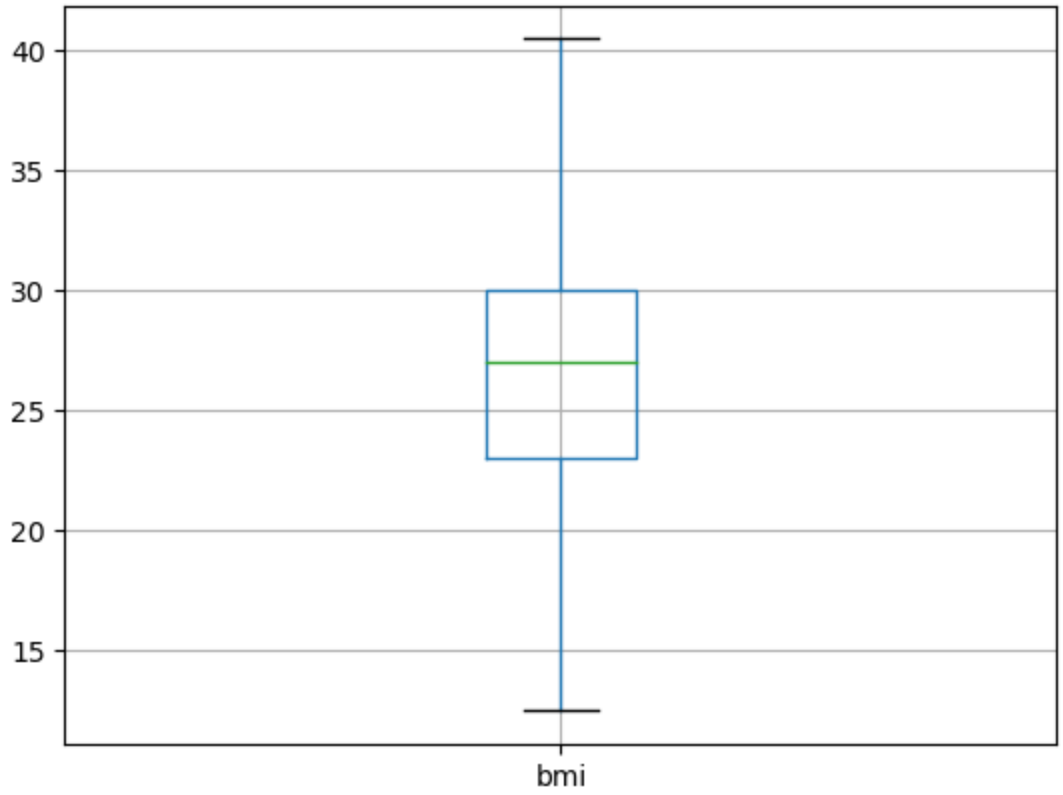
In [92]:
```
low,high=remove_outlier(numeric_data["bmi"])
```

In [94]:
```
numeric_data["bmi"]=np.where(numeric_data["bmi"]>high,high,numeric_data["bmi"])
```

In [95]:
```
numeric_data["bmi"]=np.where(numeric_data["bmi"]<low,low,numeric_data["bmi"])
```

In [96]:
```
numeric_data.boxplot(column="bmi")
```

Out[96]:
```
<Axes: >
```



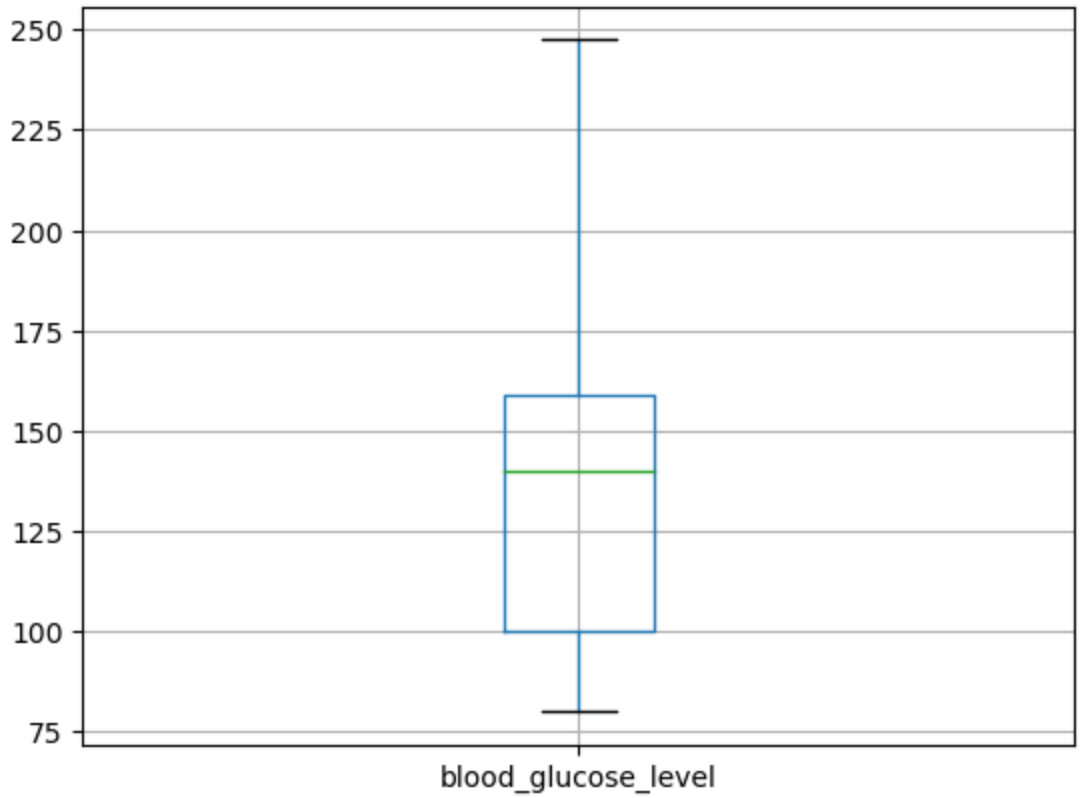
In [99]:
```
low,high=remove_outlier(numeric_data["blood_glucose_level"])
```

In [101…
```
numeric_data["blood_glucose_level"]=np.where(numeric_data["blood_glucose_level"]>high,hi
```

In [102…
```
numeric_data["blood_glucose_level"]=np.where(numeric_data["blood_glucose_level"]<low,low
```

In [103…
```
numeric_data.boxplot(column="blood_glucose_level")
```

Out[103]: <Axes: >



covariance and corelation

In [104…
```
df.cov()
```

```
C:\Users\SHREE\AppData\Local\Temp\ipykernel_6924\1545644723.py:1: FutureWarning: The def
ault value of numeric_only in DataFrame.cov is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only to sile
nce this warning.
  df.cov()
```

Out[104]:

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	dia
age	481.060387	1.636232	1.145218	45.827738	3.184086	111.023790	1.7
hypertension	1.636232	0.081142	0.006918	0.273651	0.028871	1.001469	0.0
heart_disease	1.145218	0.006918	0.044699	0.075381	0.018109	0.628509	0.0
bmi	45.827738	0.273651	0.075381	48.290671	0.769500	26.863954	0.4
HbA1c_level	3.184086	0.028871	0.018109	0.769500	1.290906	8.864122	0.1
blood_glucose_level	111.023790	1.001469	0.628509	26.863954	8.864122	1745.224962	5.5
diabetes	1.795746	0.016151	0.010560	0.435549	0.145545	5.598535	0.0

In [105…
```
df.corr()
```

```
C:\Users\SHREE\AppData\Local\Temp\ipykernel_6924\1134722465.py:1: FutureWarning: The def
ault value of numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only to sile
nce this warning.
  df.corr()
```

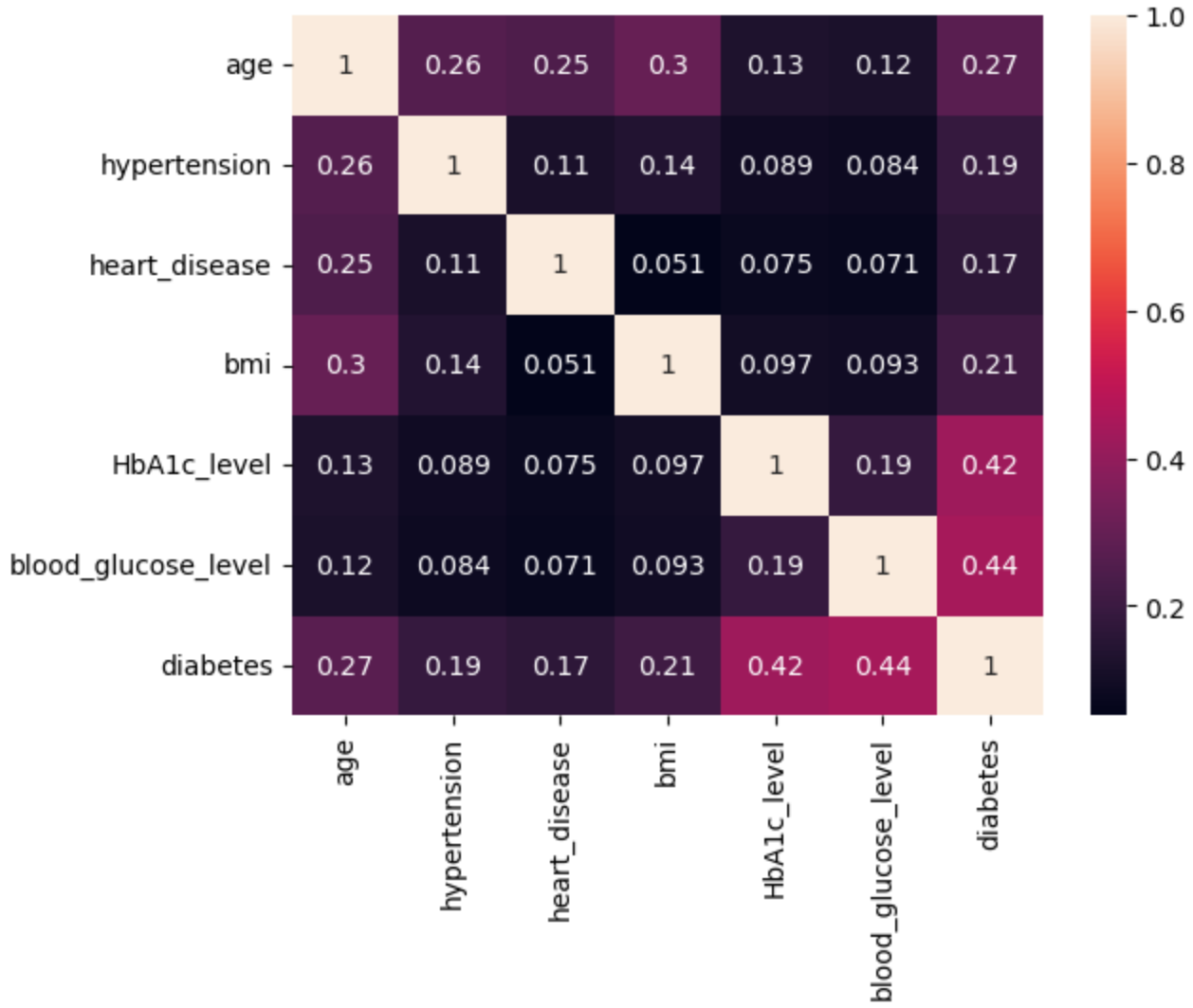
Out[105]:

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabet
age	1.000000	0.261892	0.246968	0.300675	0.127773	0.121169	0.2706
hypertension	0.261892	1.000000	0.114863	0.138243	0.089204	0.084157	0.1874
heart_disease	0.246968	0.114863	1.000000	0.051308	0.075388	0.071160	0.1650
bmi	0.300675	0.138243	0.051308	1.000000	0.097461	0.092536	0.2071
HbA1c_level	0.127773	0.089204	0.075388	0.097461	1.000000	0.186751	0.4233
blood_glucose_level	0.121169	0.084157	0.071160	0.092536	0.186751	1.000000	0.4429
diabetes	0.270600	0.187402	0.165086	0.207152	0.423384	0.442927	1.0000

In [108…
```
### heatmap
sns.heatmap(df.corr(),annot=True)
```

```
C:\Users\SHREE\AppData\Local\Temp\ipykernel_6924\3374984919.py:2: FutureWarning: The def
ault value of numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only to sile
nce this warning.
  sns.heatmap(df.corr(),annot=True)
```

Out[108]:
```
<Axes: >
```



Graph

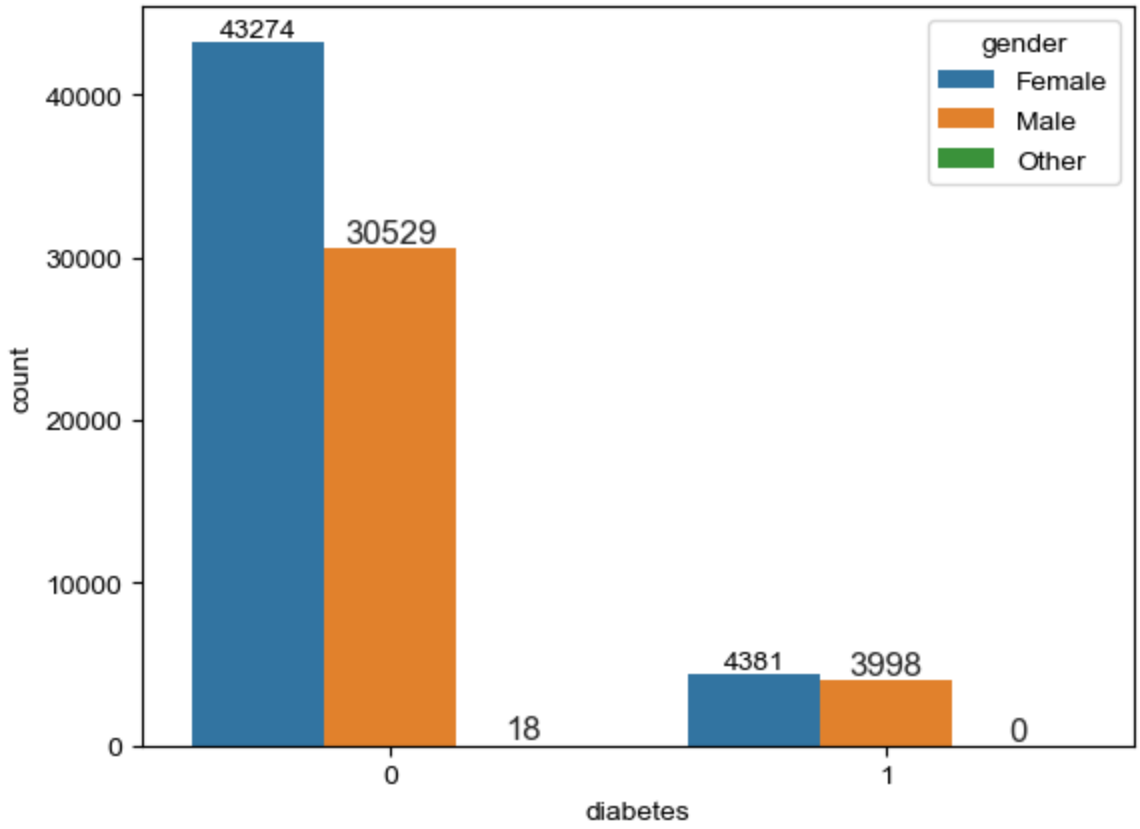
In [109…
```
df.head()
```

Out[109]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80	0	1	never	25	6	140	0
1	Female	54	0	0	No Info	27	6	80	0
2	Male	28	0	0	never	27	5	158	0
3	Female	36	0	0	current	23	5	155	0
4	Male	76	1	1	current	20	4	155	0

gender

In [111…
```
ax = sns.countplot(data = df, x = 'diabetes', hue = 'gender')

for bars in ax.containers:
    ax.bar_label(bars)
    sns.set(rc={'figure.figsize':(25,5)})
```

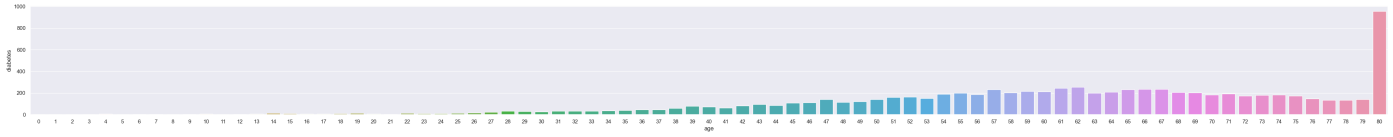


-All over the data 4381 females and 3998 males have diabetes

age

In [128…

```
diabetes_prediction= df.groupby(['age'], as_index=False)['diabetes'].sum().sort_values(b

sns.barplot(x = 'age',y= 'diabetes' ,data =diabetes_prediction)
sns.set(rc={'figure.figsize':(50,4)})
```

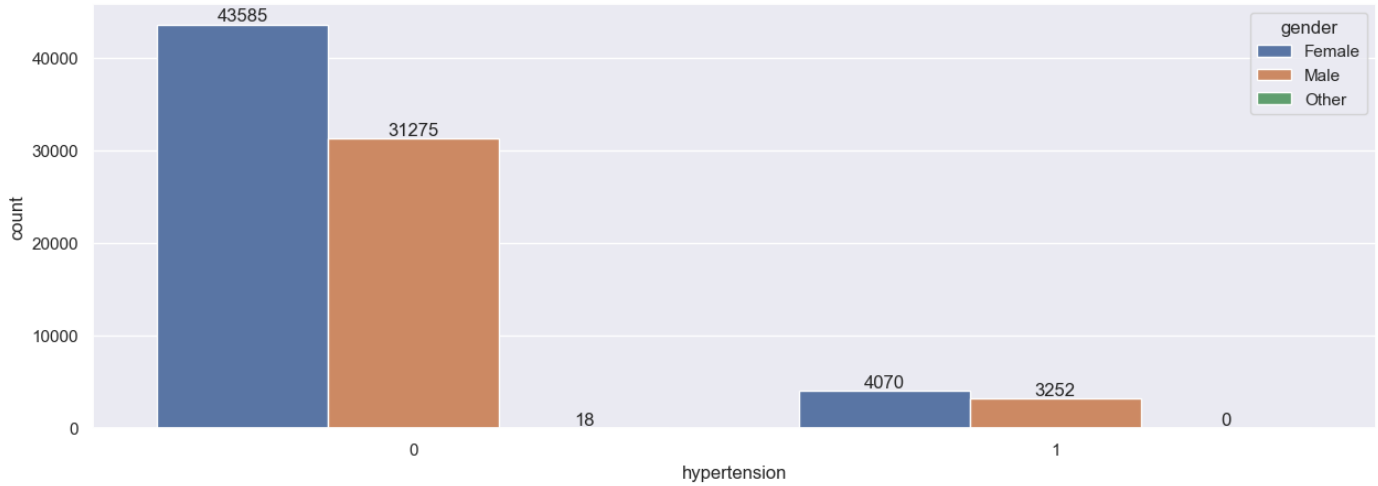


-This grapth shows that most of the people have diabeties at the age 80

hypertension

In [114…

```
ax = sns.countplot(data = df, x = 'hypertension', hue = 'gender')

for bars in ax.containers:
    ax.bar_label(bars)
    sns.set(rc={'figure.figsize':(15,10)})
```

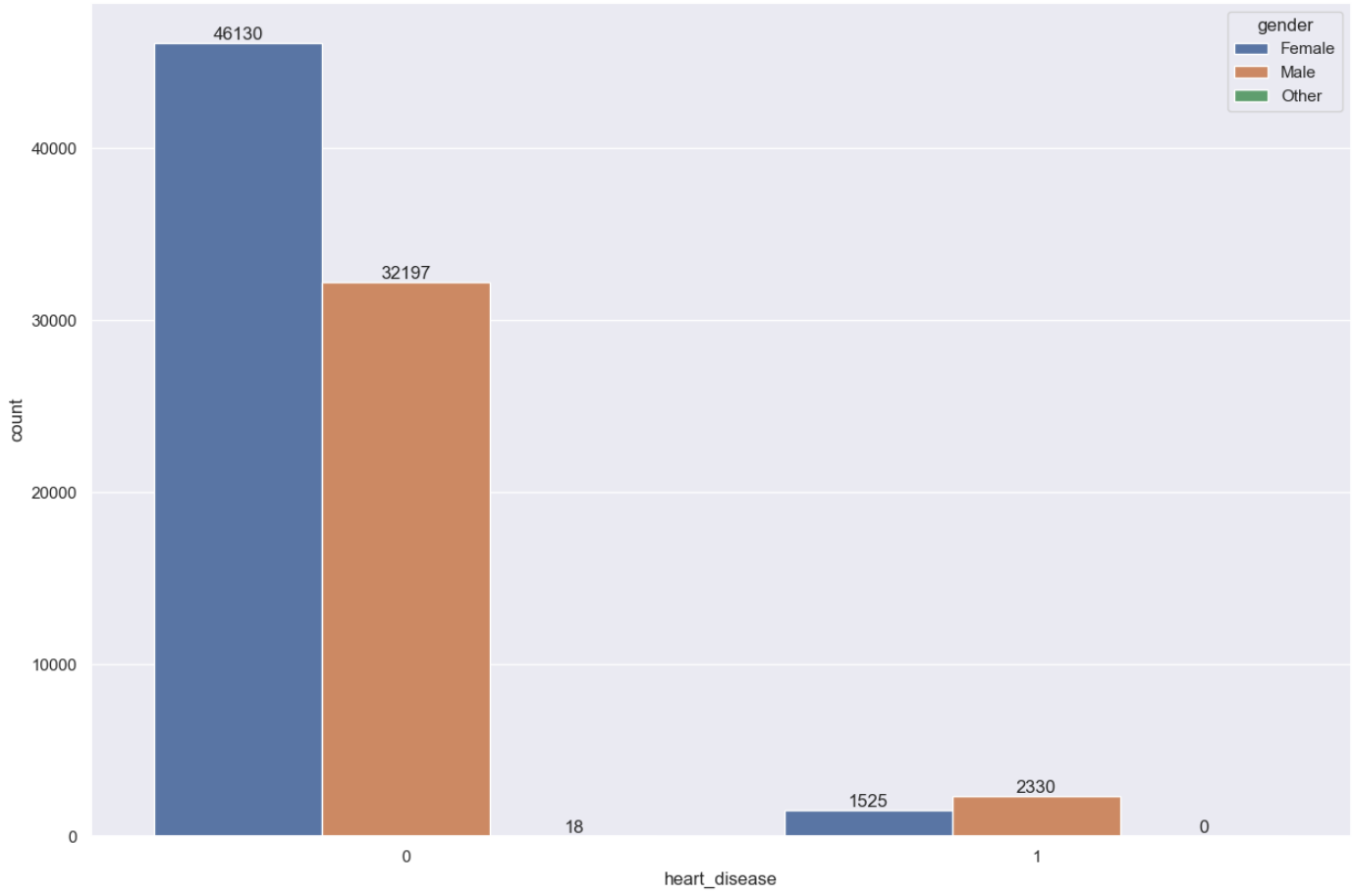


-4070 womens have hypertension whereas 3252 mens have hypertension

heart_disease

In [115…

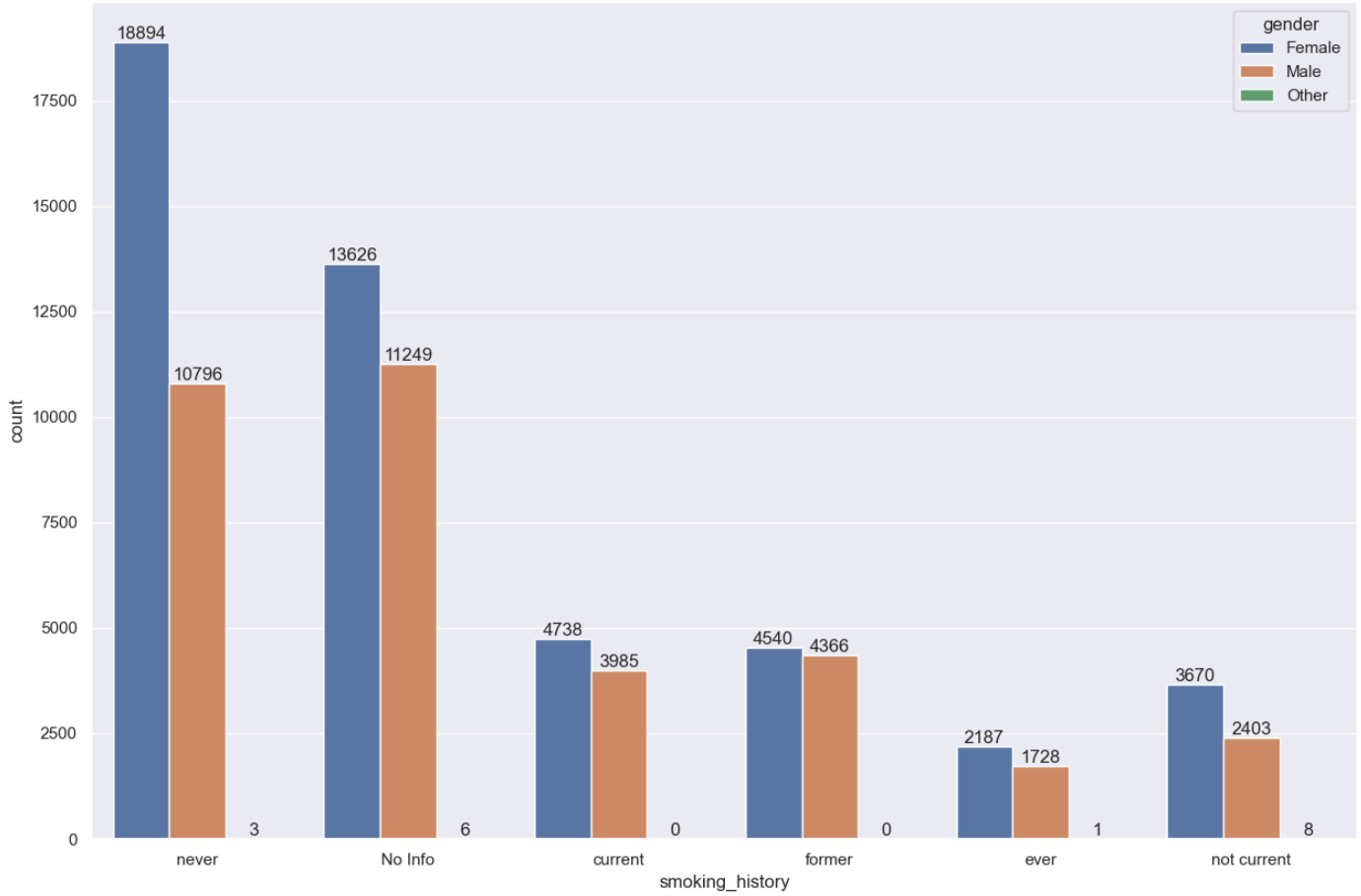
```
ax = sns.countplot(data = df, x = 'heart_disease', hue = 'gender')

for bars in ax.containers:
    ax.bar_label(bars)
    sns.set(rc={'figure.figsize':(15,10)})
```



-Here we can clearly see that ratio of the heart disease is greater in males rather than the females

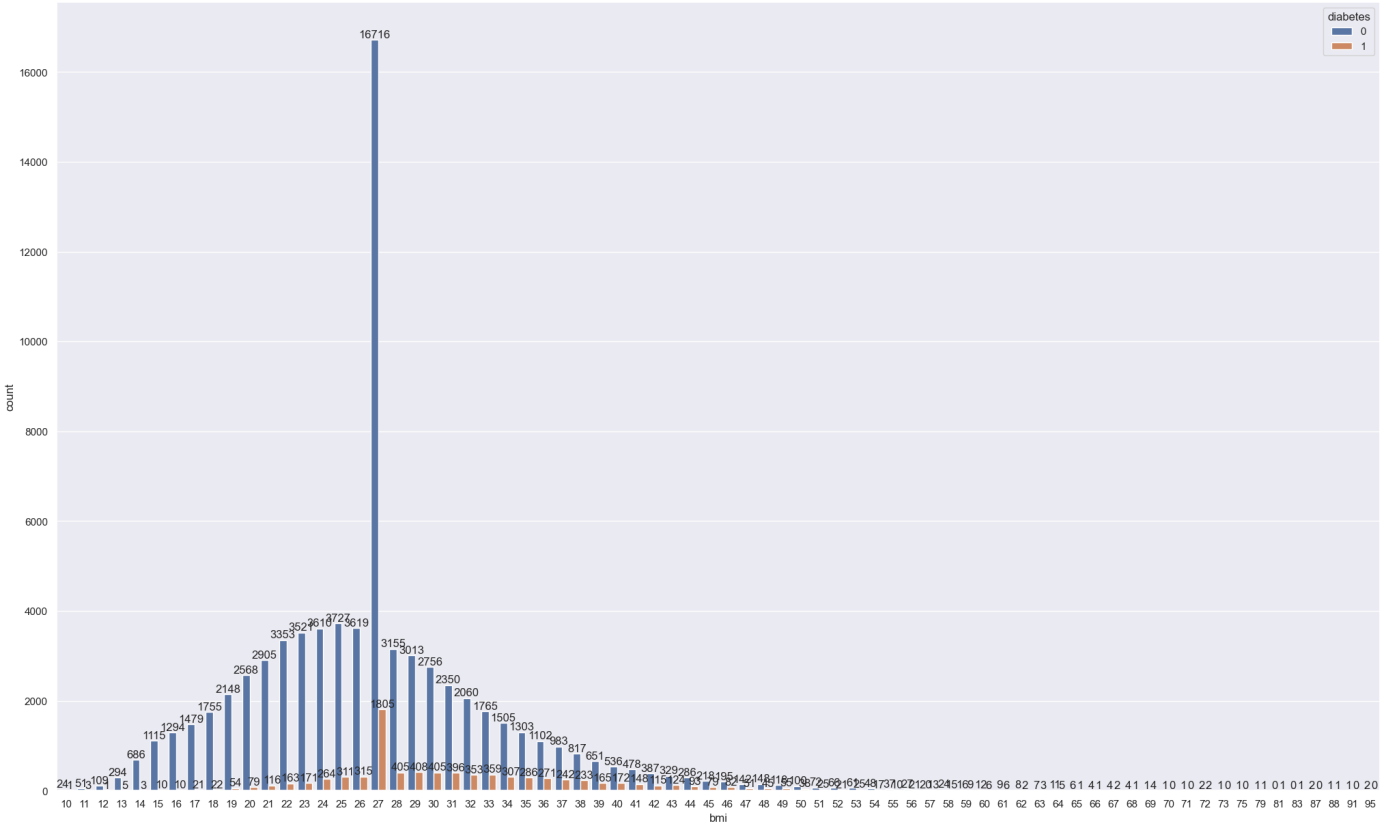
smoking history

In [116…
```
ax = sns.countplot(data = df, x = 'smoking_history', hue = 'gender')

for bars in ax.containers:
    ax.bar_label(bars)
    sns.set(rc={'figure.figsize':(15,10)})
```



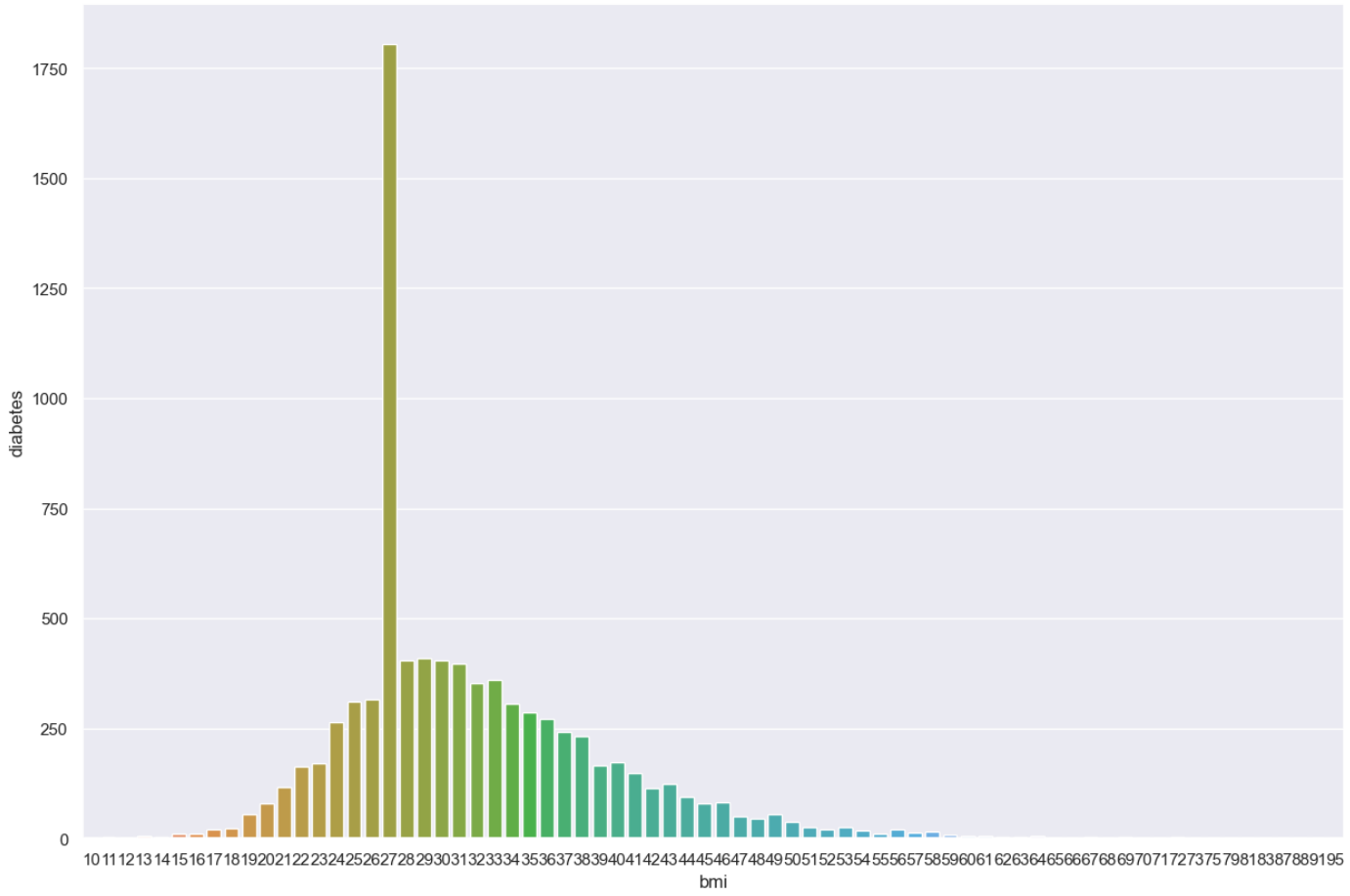
bmi

In [133…
```
ax = sns.countplot(data = df, x = 'bmi', hue = 'diabetes')

for bars in ax.containers:
    ax.bar_label(bars)
    sns.set(rc={'figure.figsize':(25,15)})
```



In [136…

```
diabetes_prediction= df.groupby(['bmi'], as_index=False)['diabetes'].sum().sort_values(b

sns.barplot(x = 'bmi',y= 'diabetes' ,data =diabetes_prediction)
sns.set(rc={'figure.figsize':(50,4)})
```



-bmi 27 shows the high rate of diabetes

conclusion= The females of age 80 which have hypertension, smoking habits and 27 bmi are predict to be diabetic.

In [ ]: