In [1]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('CVD_cleaned.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabetes	Arthritis	Sex	Age_Category	Height_(cm)	Weight_(kg)	BMI	Smoking_History	Alcohol_Consumption	Fruit_C
0	Poor	Within the past 2 years	No	No	No	No	No	No	Yes	Female	70-74	150.0	32.66	14.54	Yes	0.0	
1	Very Good	Within the past year	No	Yes	No	No	No	Yes	No	Female	70-74	165.0	77.11	28.29	No	0.0	
2	Very Good	Within the past year	Yes	No	No	No	No	Yes	No	Female	60-64	163.0	88.45	33.47	No	4.0	
3	Poor	Within the past year	Yes	Yes	No	No	No	Yes	No	Male	75-79	180.0	93.44	28.73	No	0.0	
4	Good	Within the past year	No	No	No	No	No	No	No	Male	80+	191.0	88.45	24.37	Yes	0.0	

In [4]:
```
df.shape
```

Out[4]:
```
(308854, 19)
```

In [5]:
```
df.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
Height_(cm)	308854.0	170.615249	10.658026	91.00	163.00	170.00	178.00	241.00
Weight_(kg)	308854.0	83.588655	21.343210	24.95	68.04	81.65	95.25	293.02
BMI	308854.0	28.626211	6.522323	12.02	24.21	27.44	31.85	99.33
Alcohol_Consumption	308854.0	5.096366	8.199763	0.00	0.00	1.00	6.00	30.00
Fruit_Consumption	308854.0	29.835200	24.875735	0.00	12.00	30.00	30.00	120.00
Green_Vegetables_Consumption	308854.0	15.110441	14.926238	0.00	4.00	12.00	20.00	128.00
FriedPotato_Consumption	308854.0	6.296616	8.582954	0.00	2.00	4.00	8.00	128.00

In [6]:
```
df.describe(exclude='number').T
```

Out[6]:

	count	unique	top	freq
General_Health	308854	5	Very Good	110395
Checkup	308854	5	Within the past year	239371
Exercise	308854	2	Yes	239381
Heart_Disease	308854	2	No	283883
Skin_Cancer	308854	2	No	278860
Other_Cancer	308854	2	No	278976
Depression	308854	2	No	246953
Diabetes	308854	4	No	259141
Arthritis	308854	2	No	207783
Sex	308854	2	Female	160196
Age_Category	308854	13	65-69	33434
Smoking_History	308854	2	No	183590

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308854 entries, 0 to 308853
Data columns (total 19 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   General_Health                308854 non-null  object 
 1   Checkup                       308854 non-null  object 
 2   Exercise                      308854 non-null  object 
 3   Heart_Disease                 308854 non-null  object 
 4   Skin_Cancer                   308854 non-null  object 
 5   Other_Cancer                  308854 non-null  object 
 6   Depression                    308854 non-null  object 
 7   Diabetes                      308854 non-null  object 
 8   Arthritis                     308854 non-null  object 
 9   Sex                           308854 non-null  object 
 10  Age_Category                  308854 non-null  object 
 11  Height_(cm)                   308854 non-null  float64
 12  Weight_(kg)                   308854 non-null  float64
 13  BMI                           308854 non-null  float64
 14  Smoking_History               308854 non-null  object 
 15  Alcohol_Consumption           308854 non-null  float64
 16  Fruit_Consumption             308854 non-null  float64
 17  Green_Vegetables_Consumption  308854 non-null  float64
 18  FriedPotato_Consumption       308854 non-null  float64
dtypes: float64(7), object(12)
memory usage: 44.8+ MB
```

In [8]:
```
numeric = df.columns[df.dtypes != 'object']
cat = df.columns[df.dtypes == 'object']
print(numeric)
print(cat)
```

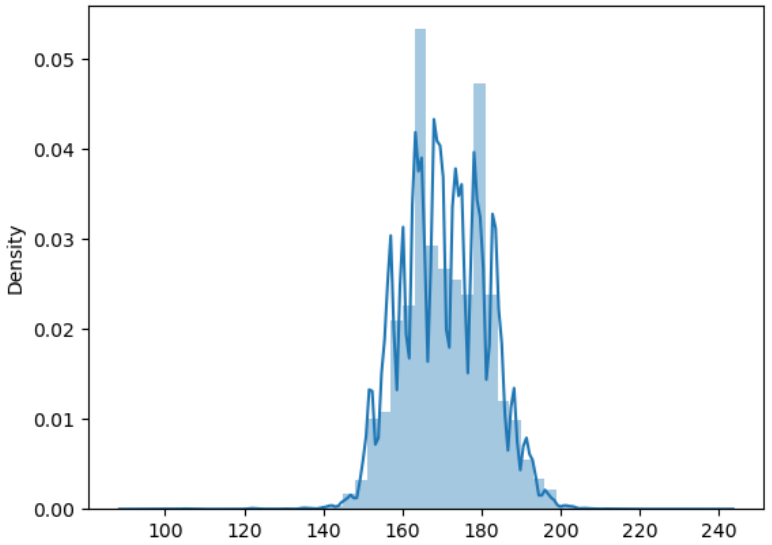
```
Index(['Height_(cm)', 'Weight_(kg)', 'BMI', 'Alcohol_Consumption',
       'Fruit_Consumption', 'Green_Vegetables_Consumption',
       'FriedPotato_Consumption'],
      dtype='object')
Index(['General_Health', 'Checkup', 'Exercise', 'Heart_Disease', 'Skin_Cancer',
       'Other_Cancer', 'Depression', 'Diabetes', 'Arthritis', 'Sex',
       'Age_Category', 'Smoking_History'],
      dtype='object')
```

Analysing Dataset

Univariate Analysis

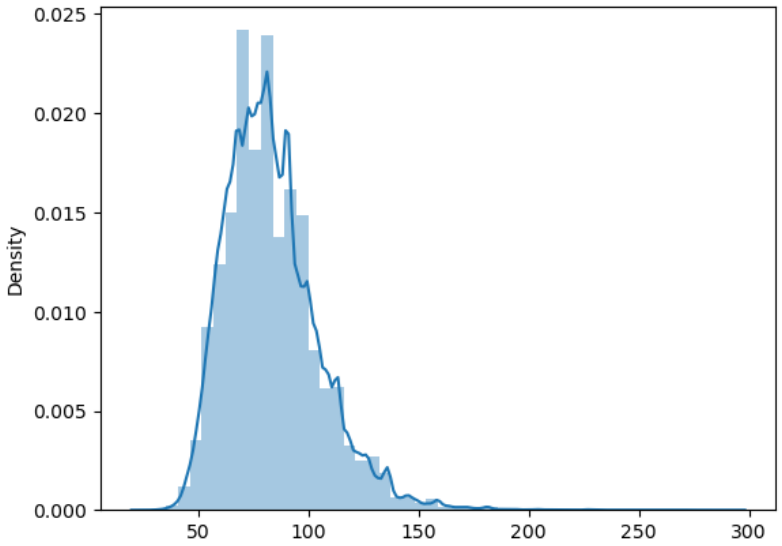
In [9]:
```
sns.distplot(x = df['Height_(cm)'])
```

Out[9]:
```
<Axes: ylabel='Density'>
```



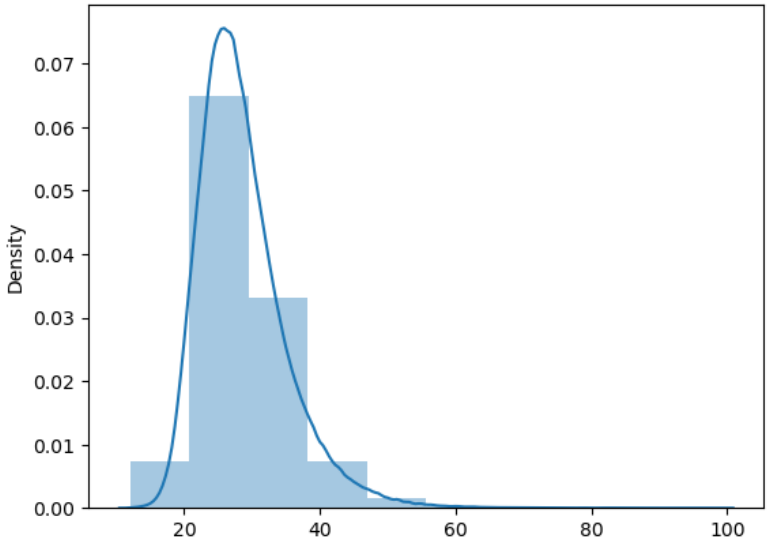
In [10]:
```
sns.distplot(x = df['Weight_(kg)'])
```

Out[10]:
```
<Axes: ylabel='Density'>
```



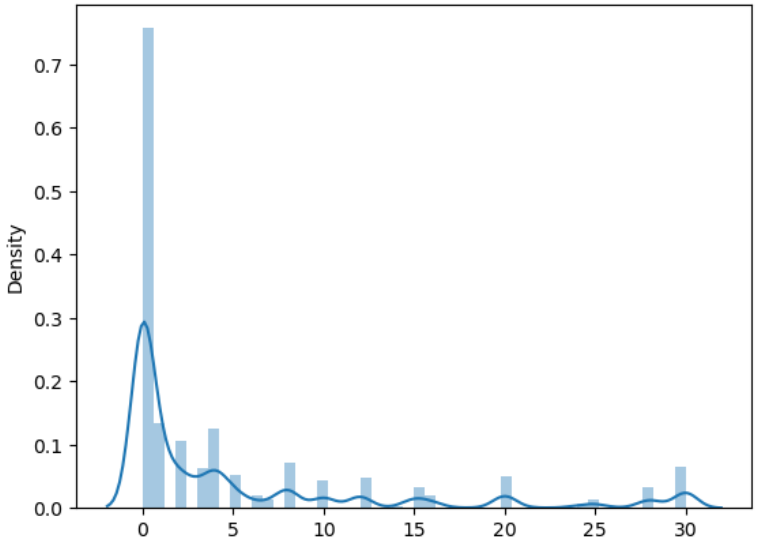
In [11]:
```
sns.distplot(x = df['BMI'], bins=10)
```

Out[11]:
```
<Axes: ylabel='Density'>
```



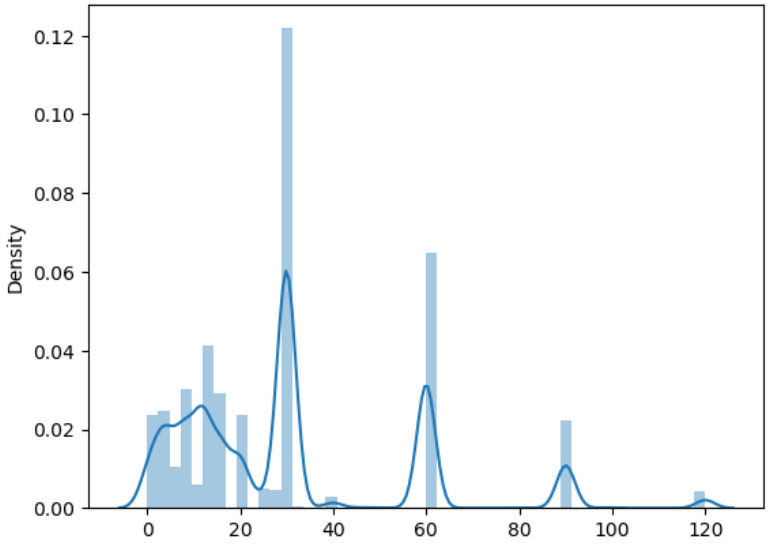
In [12]:
```
sns.distplot(x = df['Alcohol_Consumption'])
```

Out[12]:
```
<Axes: ylabel='Density'>
```



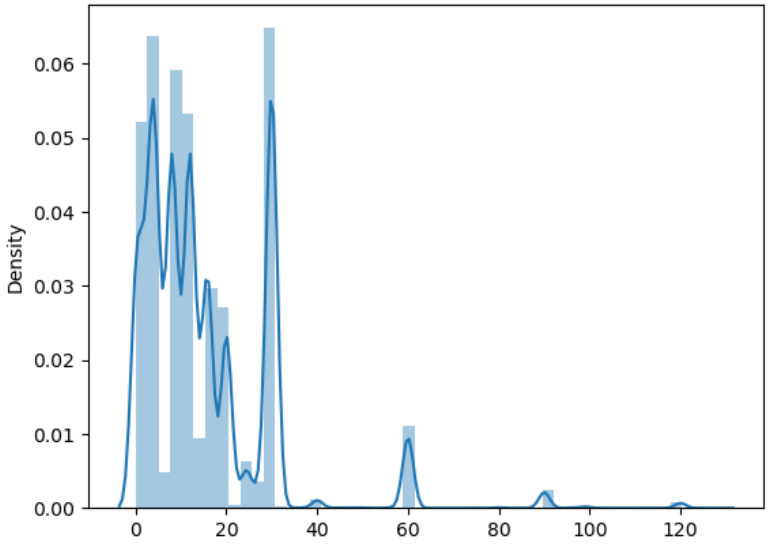
In [13]:
```
sns.distplot(x = df['Fruit_Consumption'])
```

Out[13]:
```
<Axes: ylabel='Density'>
```



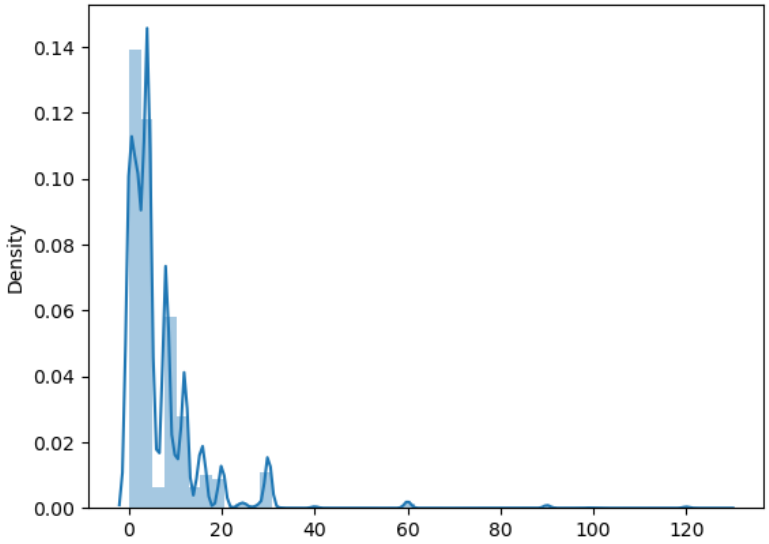
In [14]:
```
sns.distplot(x = df['Green_Vegetables_Consumption'])
```

Out[14]:
```
<Axes: ylabel='Density'>
```



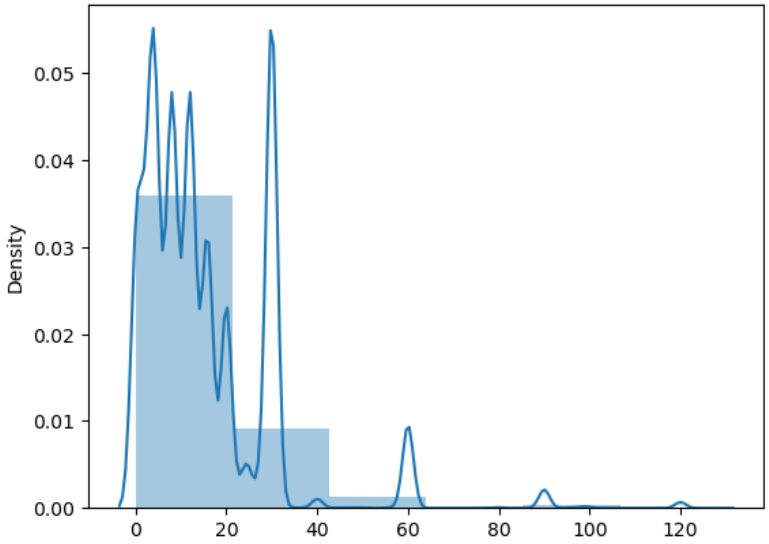
In [15]:
```
sns.distplot(x = df['FriedPotato_Consumption'])
```

Out[15]:
```
<Axes: ylabel='Density'>
```



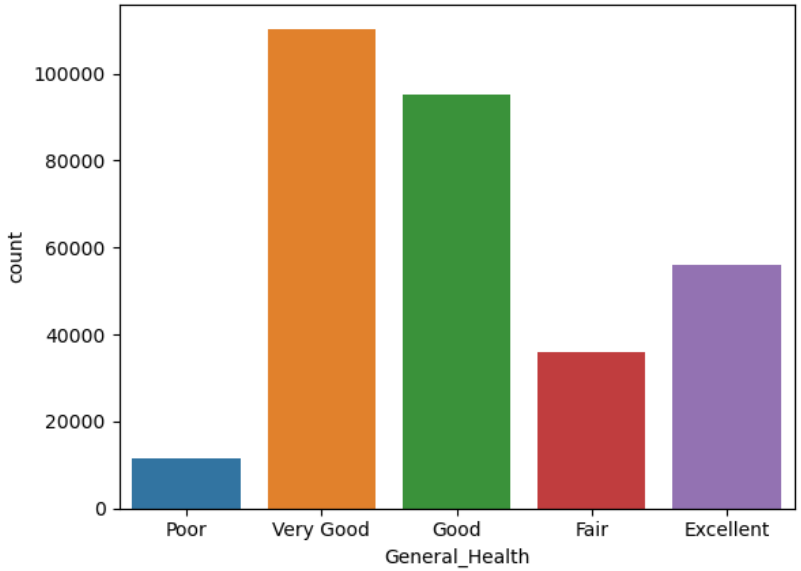
In [16]:
```
sns.distplot(x = df['Green_Vegetables_Consumption'], bins=6)
```

Out[16]:
```
<Axes: ylabel='Density'>
```



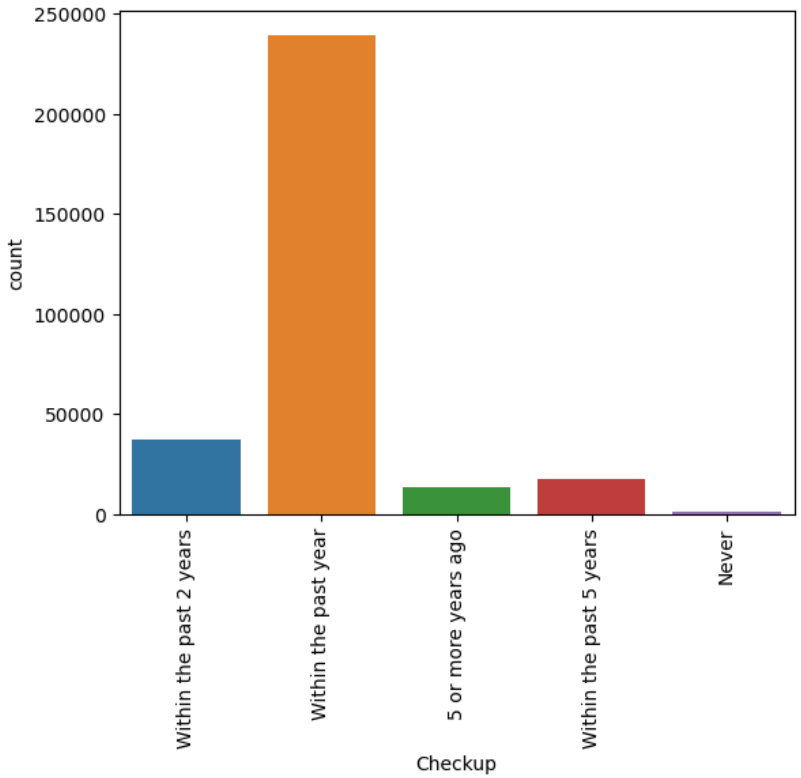
In [17]:
```
sns.countplot(x=df['General_Health'])
```

Out[17]:
```
<Axes: xlabel='General_Health', ylabel='count'>
```



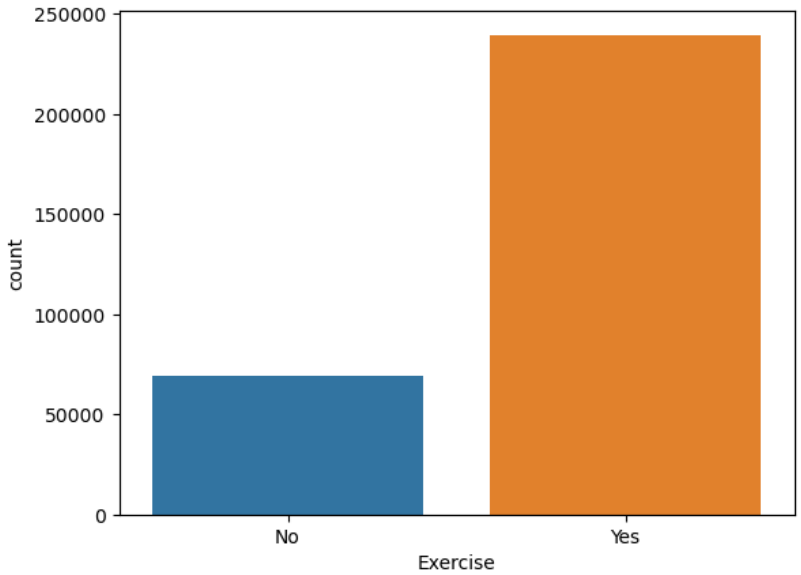
In [18]:
```
sns.countplot(x=df['Checkup'])
plt.xticks(rotation=90)
```

Out[18]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Within the past 2 years'),
  Text(1, 0, 'Within the past year'),
  Text(2, 0, '5 or more years ago'),
  Text(3, 0, 'Within the past 5 years'),
  Text(4, 0, 'Never')])
```



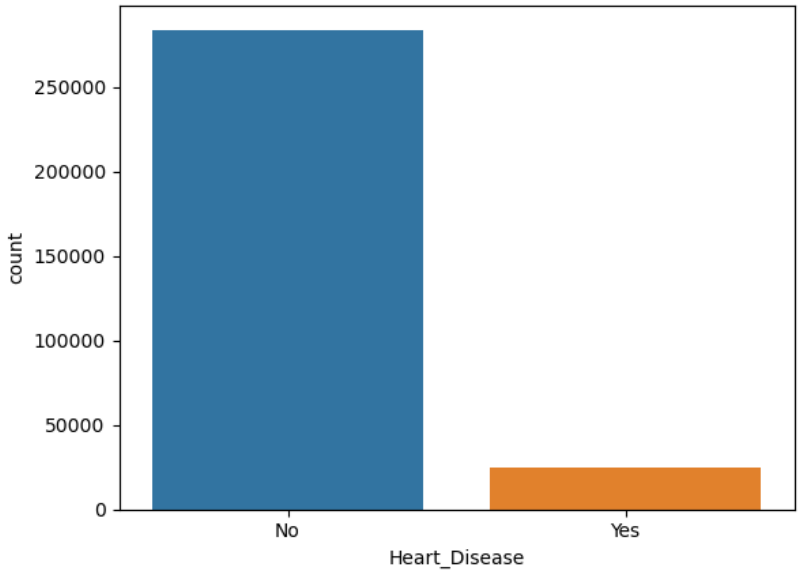
In [19]:
```
sns.countplot(x=df['Exercise'])
```

Out[19]:
```
<Axes: xlabel='Exercise', ylabel='count'>
```



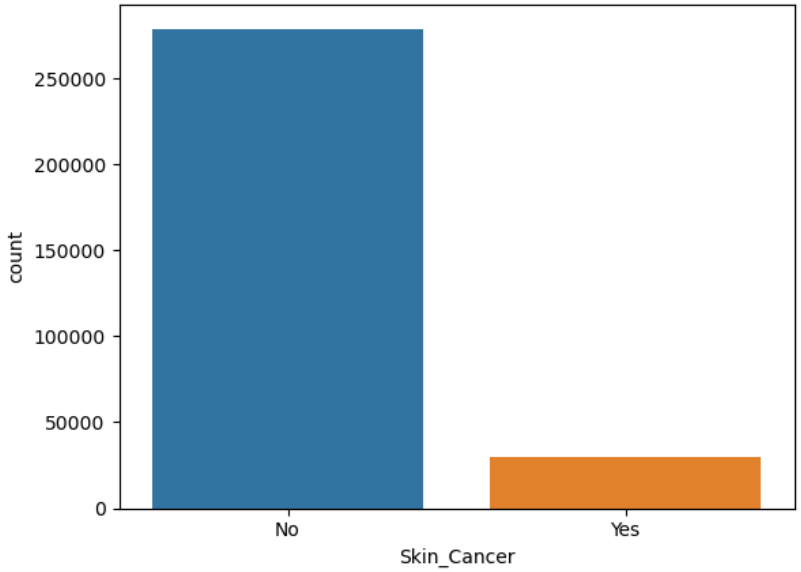
In [20]:
```
sns.countplot(x=df['Heart_Disease'])
```

Out[20]:
```
<Axes: xlabel='Heart_Disease', ylabel='count'>
```



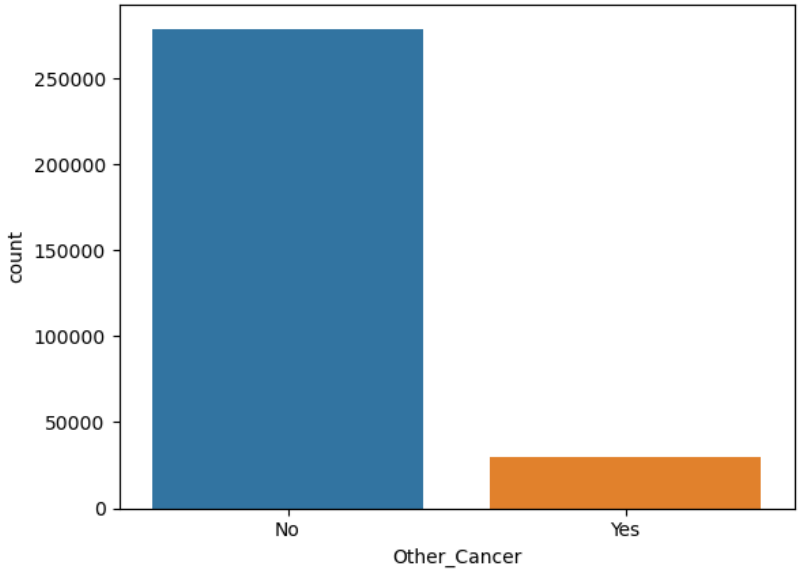
In [21]:
```
sns.countplot(x=df['Skin_Cancer'])
```

Out[21]:
```
<Axes: xlabel='Skin_Cancer', ylabel='count'>
```



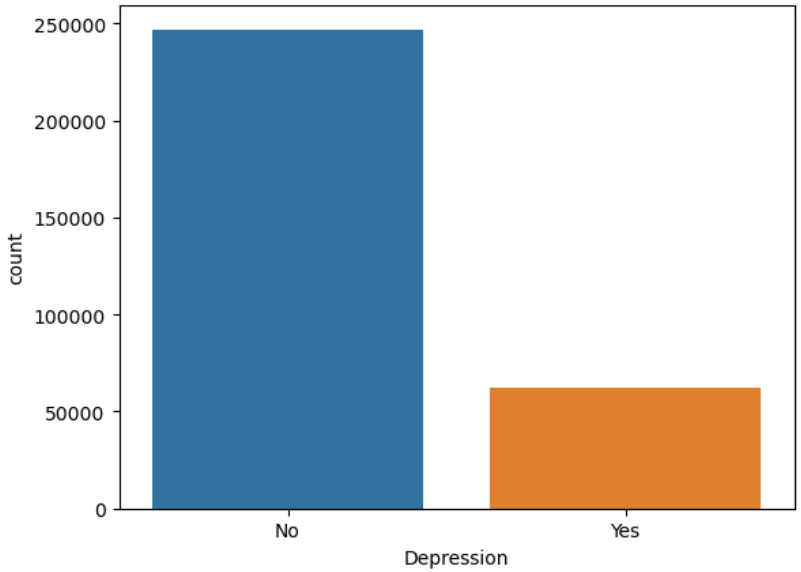
In [22]:
```
sns.countplot(x=df['Other_Cancer'])
```

Out[22]:
```
<Axes: xlabel='Other_Cancer', ylabel='count'>
```



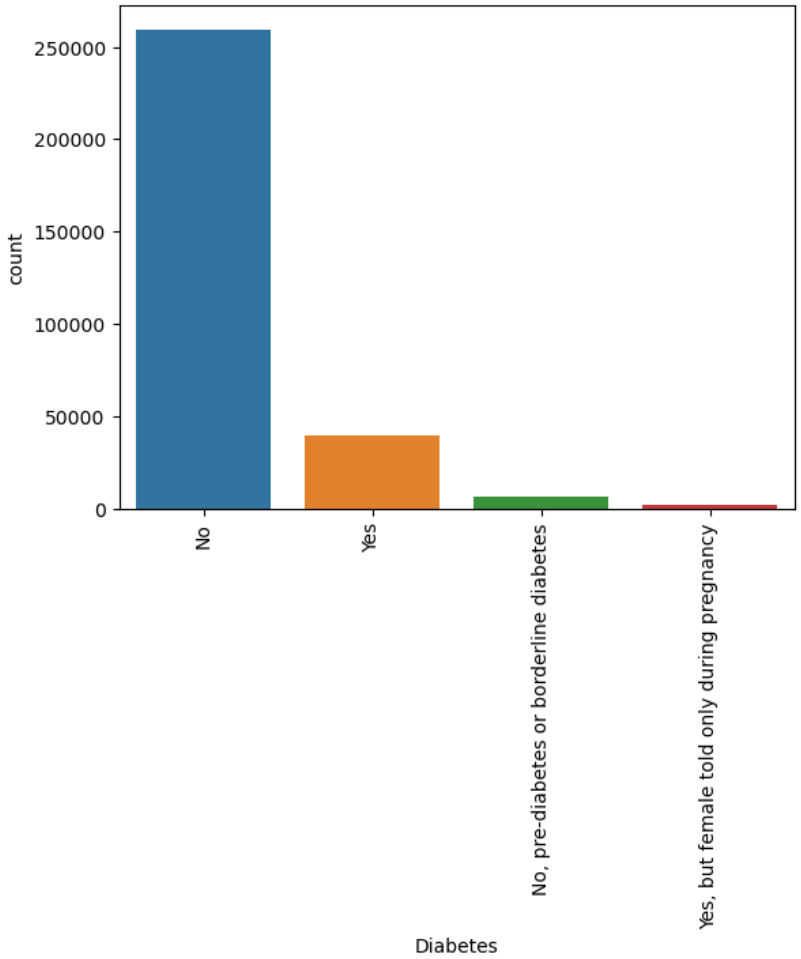
In [23]:
```
sns.countplot(x=df['Depression'])
```

Out[23]:
```
<Axes: xlabel='Depression', ylabel='count'>
```



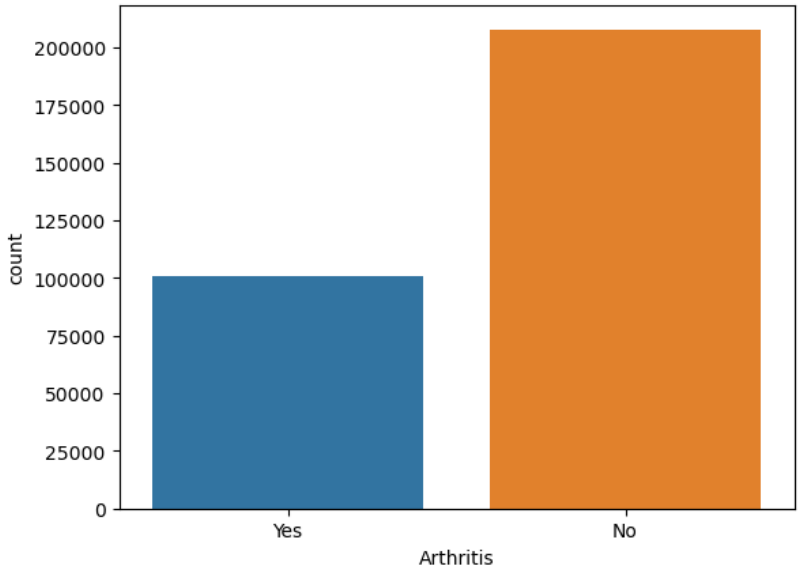
In [24]:
```
sns.countplot(x=df['Diabetes'])
plt.xticks(rotation= 90)
```

Out[24]:
```
(array([0, 1, 2, 3]),
 [Text(0, 0, 'No'),
  Text(1, 0, 'Yes'),
  Text(2, 0, 'No, pre-diabetes or borderline diabetes'),
  Text(3, 0, 'Yes, but female told only during pregnancy')])
```



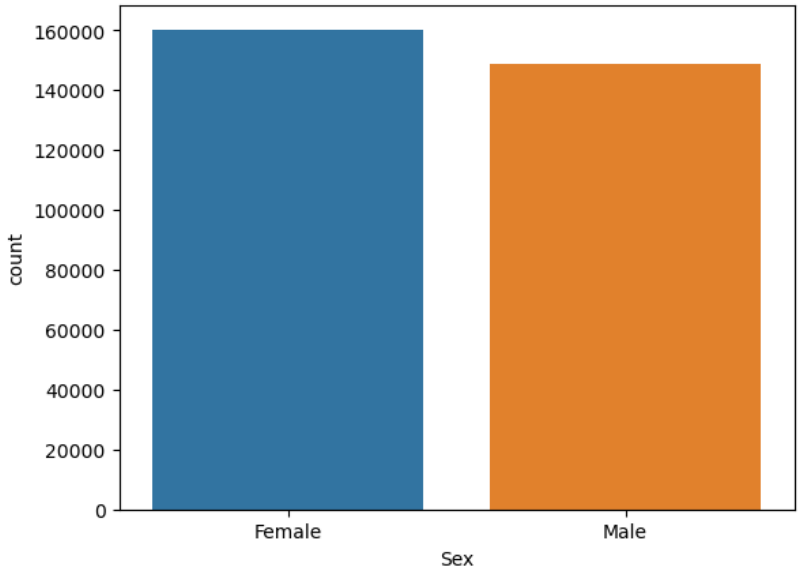
In [25]:
```
sns.countplot(x=df['Arthritis'])
```

Out[25]:
```
<Axes: xlabel='Arthritis', ylabel='count'>
```



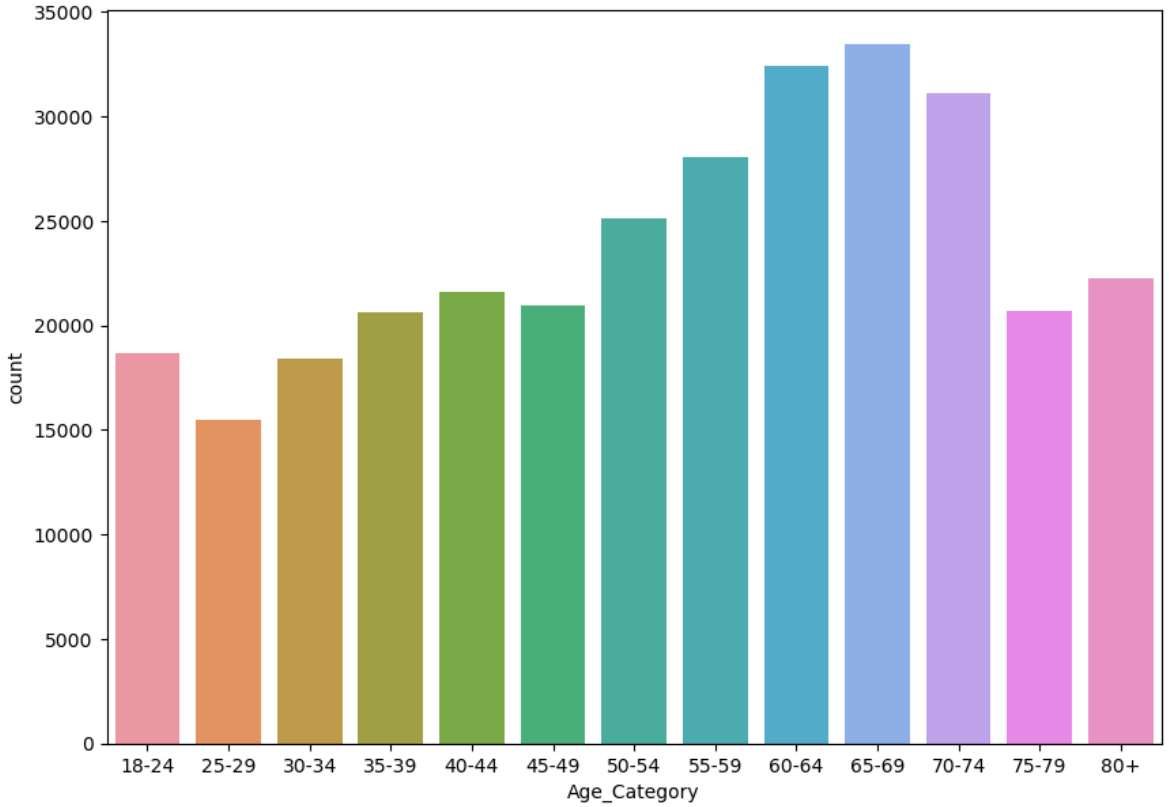
In [26]:
```
sns.countplot(x=df['Sex'])
```

Out[26]:
```
<Axes: xlabel='Sex', ylabel='count'>
```



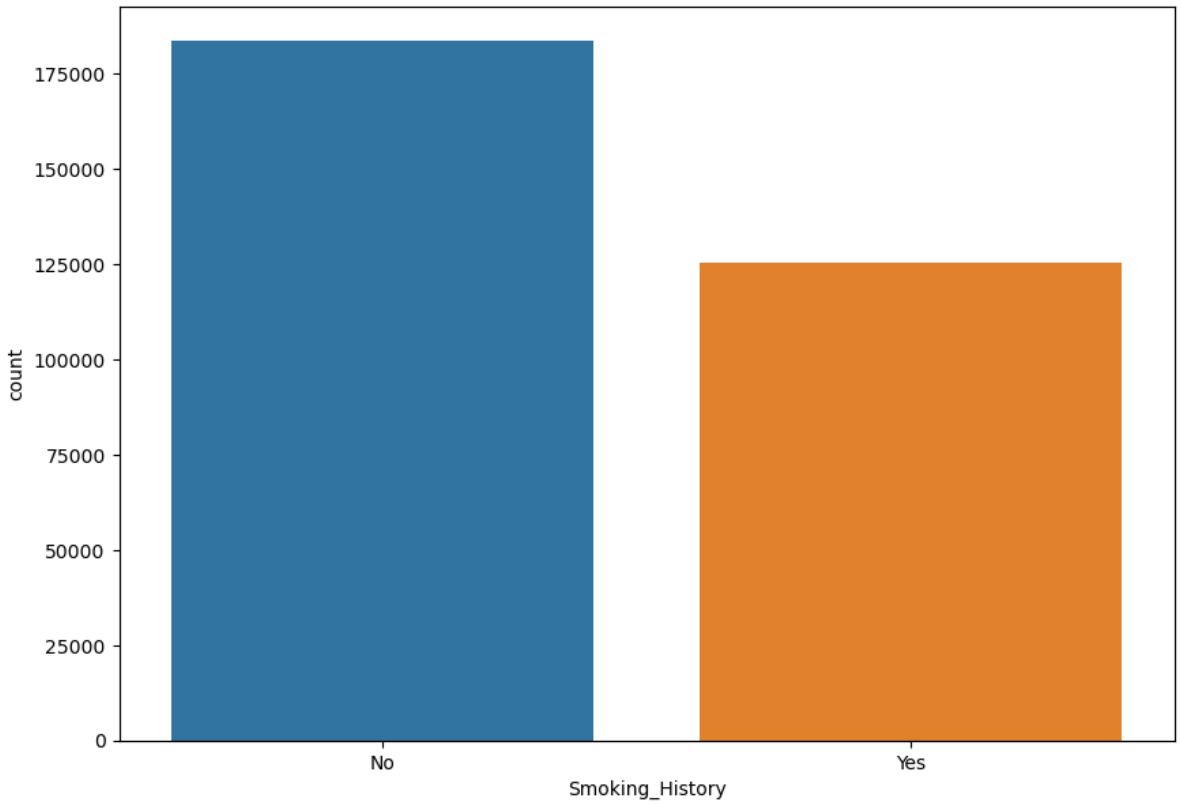
In [27]:
```
plt.figure(figsize = (10,7))
sns.countplot(x=df['Age_Category'].sort_values())
```

Out[27]:
```
<Axes: xlabel='Age_Category', ylabel='count'>
```



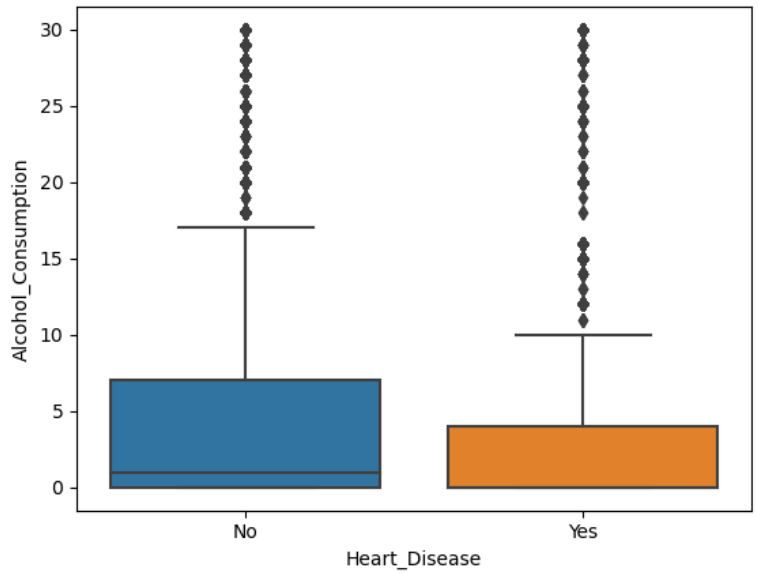
In [28]:
```
plt.figure(figsize = (10,7))
sns.countplot(x=df['Smoking_History'].sort_values())
```

Out[28]:
```
<Axes: xlabel='Smoking_History', ylabel='count'>
```



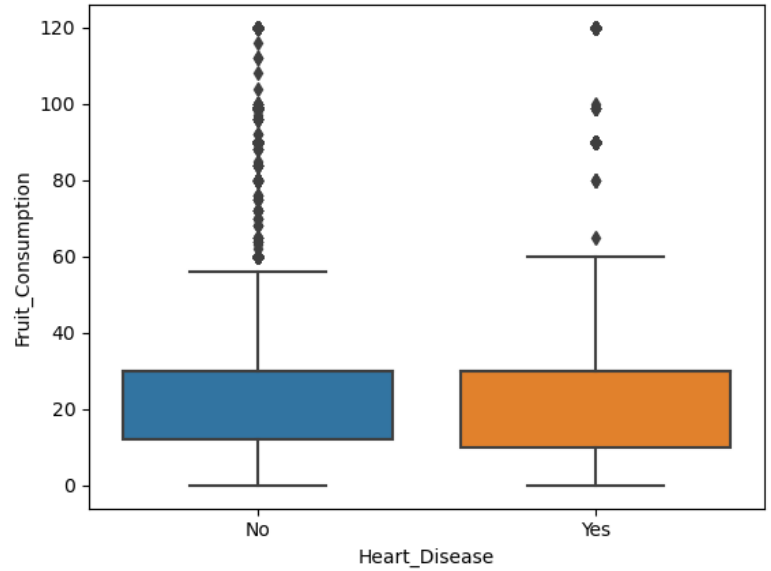
In [29]:
```
sns.boxplot(x = df['Heart_Disease'], y= df ['Alcohol_Consumption'])
```

Out[29]:
```
<Axes: xlabel='Heart_Disease', ylabel='Alcohol_Consumption'>
```



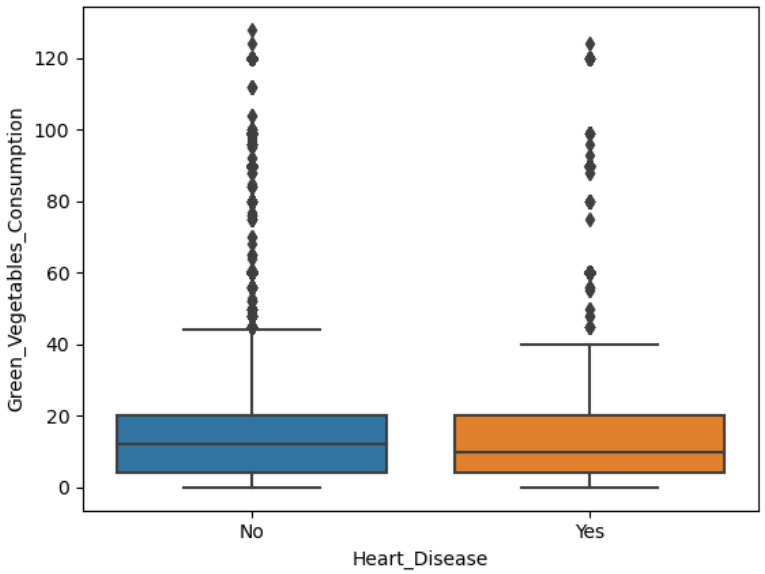
In [30]:
```
sns.boxplot(x = df['Heart_Disease'], y= df ['Fruit_Consumption'])
```

Out[30]:
```
<Axes: xlabel='Heart_Disease', ylabel='Fruit_Consumption'>
```



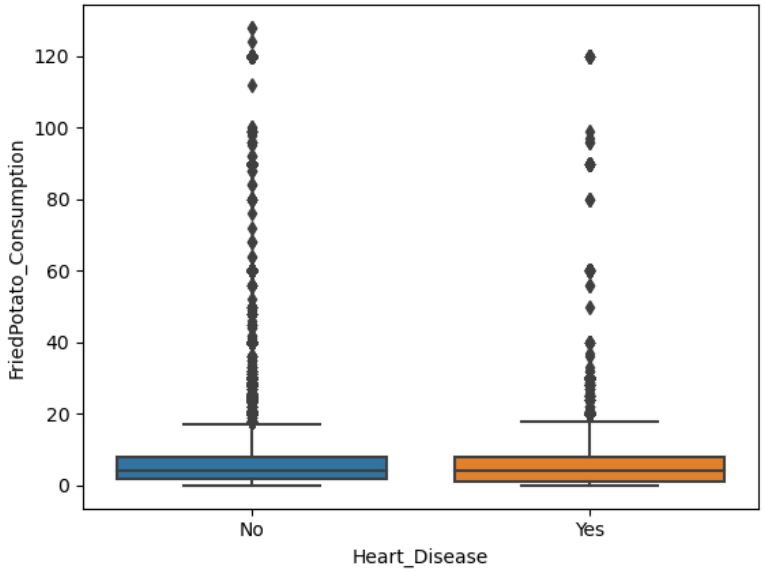
In [31]:
```
sns.boxplot(x = df['Heart_Disease'], y= df ['Green_Vegetables_Consumption'])
```

Out[31]:
```
<Axes: xlabel='Heart_Disease', ylabel='Green_Vegetables_Consumption'>
```



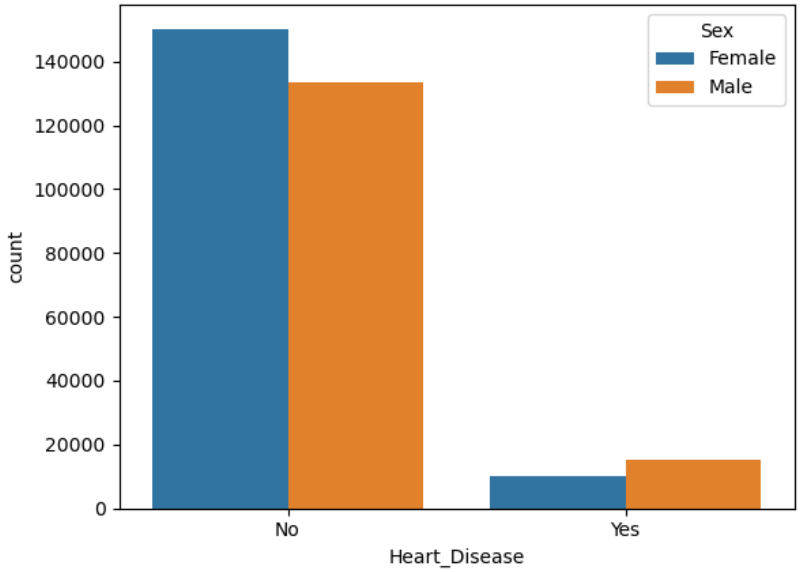
In [32]:
```
sns.boxplot(x = df['Heart_Disease'], y= df ['FriedPotato_Consumption'])
```

Out[32]:
```
<Axes: xlabel='Heart_Disease', ylabel='FriedPotato_Consumption'>
```



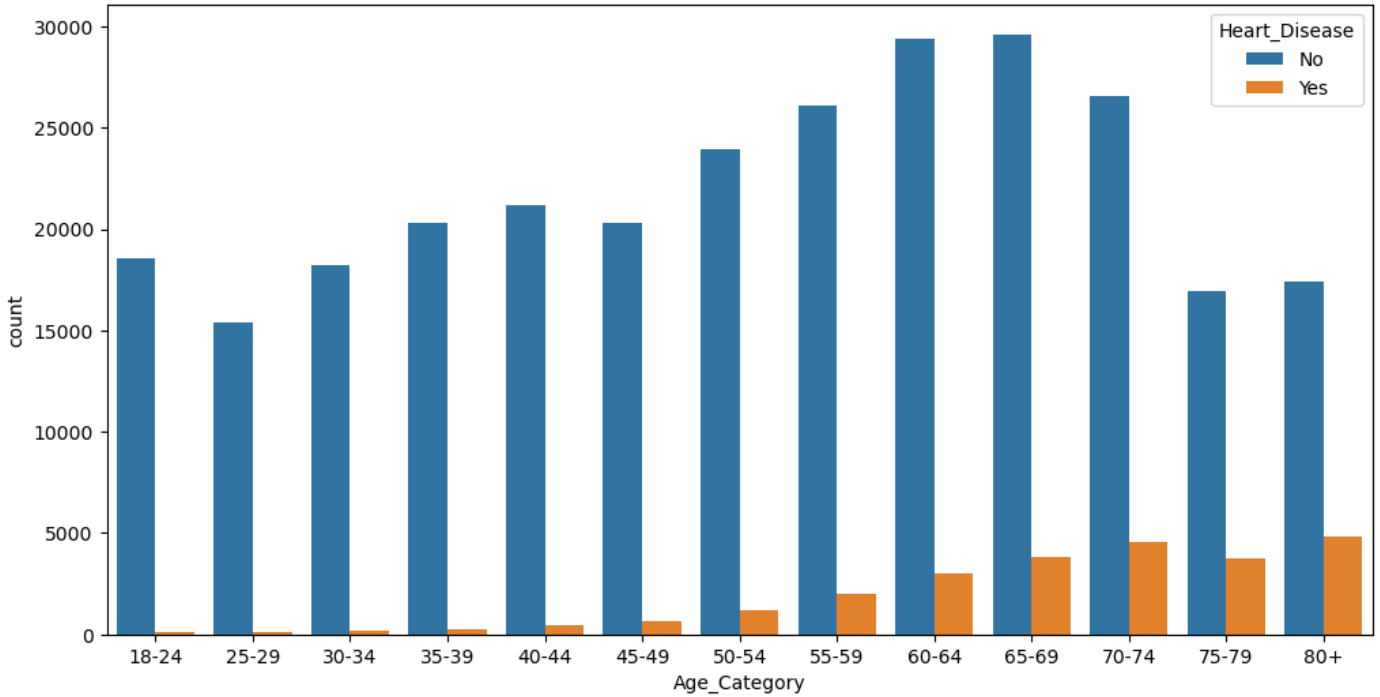
In [33]:
```
sns.countplot(x = df['Heart_Disease'], hue= df ['Sex'])
```

Out[33]:
```
<Axes: xlabel='Heart_Disease', ylabel='count'>
```



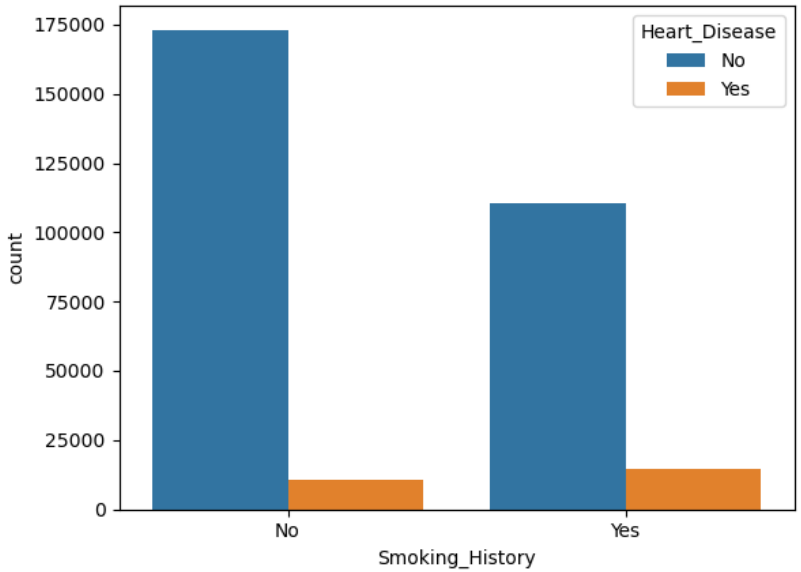
In [34]:
```
plt.figure(figsize = (12,6))
sns.countplot(x = df['Age_Category'].sort_values(), hue= df['Heart_Disease'])
```

Out[34]:
```
<Axes: xlabel='Age_Category', ylabel='count'>
```



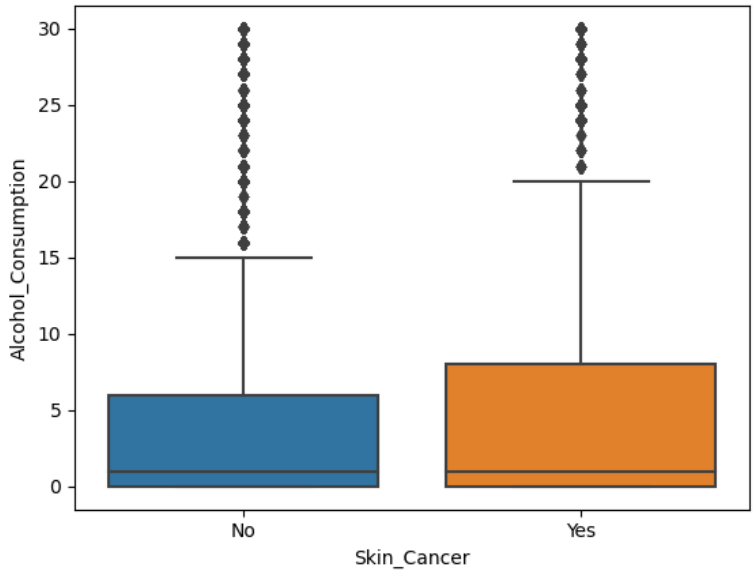
In [35]:
```
sns.countplot(x = df['Smoking_History'].sort_values(), hue= df['Heart_Disease'])
```

Out[35]:
```
<Axes: xlabel='Smoking_History', ylabel='count'>
```



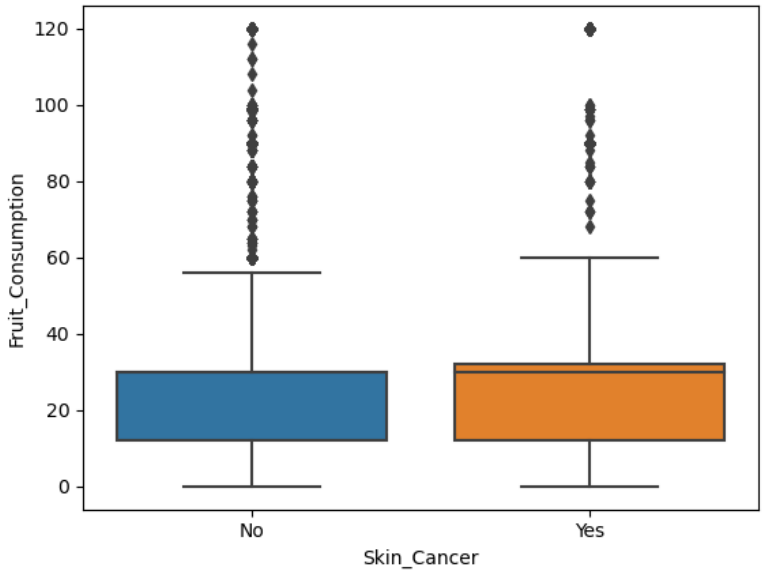
In [36]:
```
sns.boxplot(x = df['Skin_Cancer'], y= df['Alcohol_Consumption'])
```

Out[36]:
```
<Axes: xlabel='Skin_Cancer', ylabel='Alcohol_Consumption'>
```



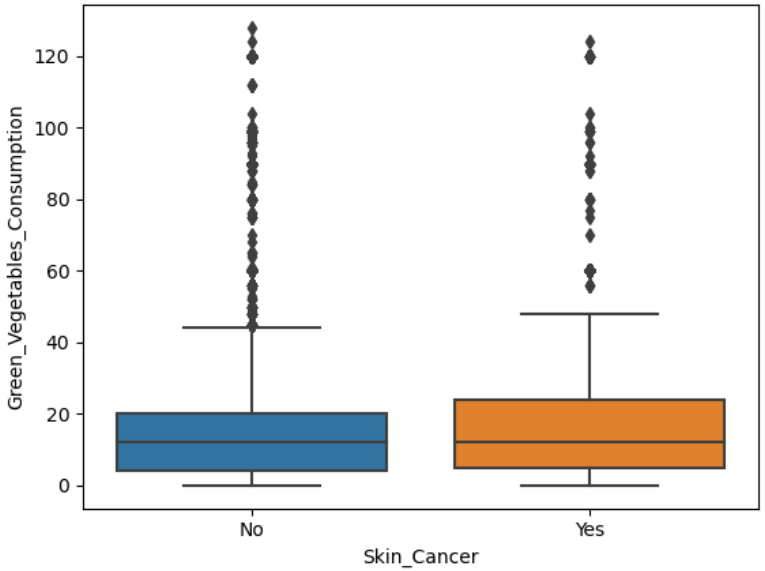
In [37]:
```
sns.boxplot(x = df['Skin_Cancer'], y= df['Fruit_Consumption'])
```

Out[37]: <Axes: xlabel='Skin_Cancer', ylabel='Fruit_Consumption'>



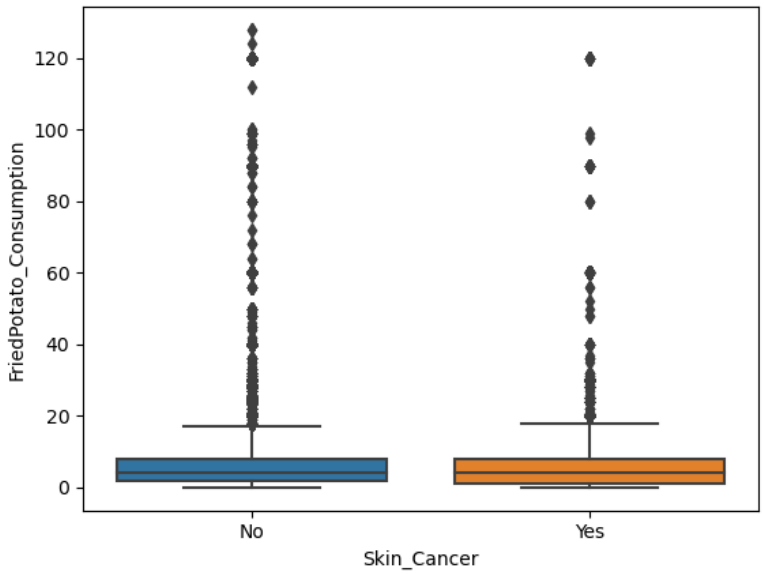
In [38]:
```
sns.boxplot(x = df['Skin_Cancer'], y= df['Green_Vegetables_Consumption'])
```

Out[38]: <Axes: xlabel='Skin_Cancer', ylabel='Green_Vegetables_Consumption'>



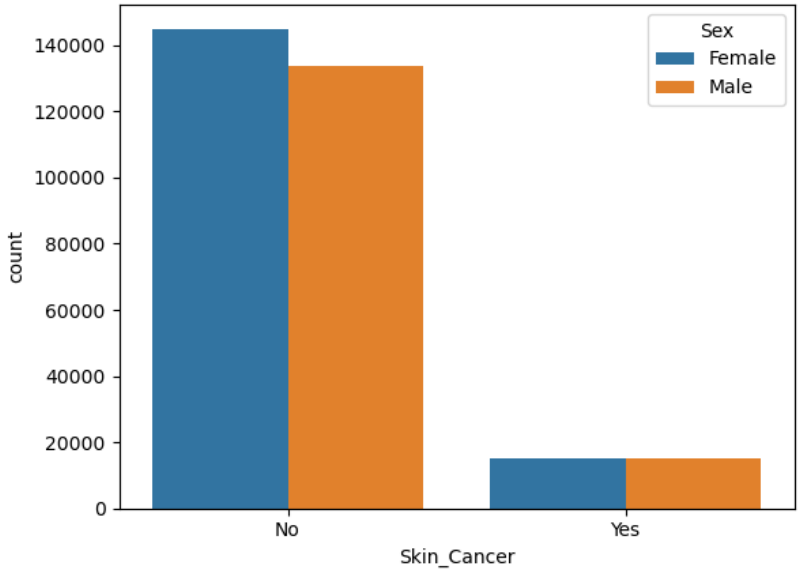
In [39]:
```
sns.boxplot(x = df['Skin_Cancer'], y= df['FriedPotato_Consumption'])
```

Out[39]: <Axes: xlabel='Skin_Cancer', ylabel='FriedPotato_Consumption'>



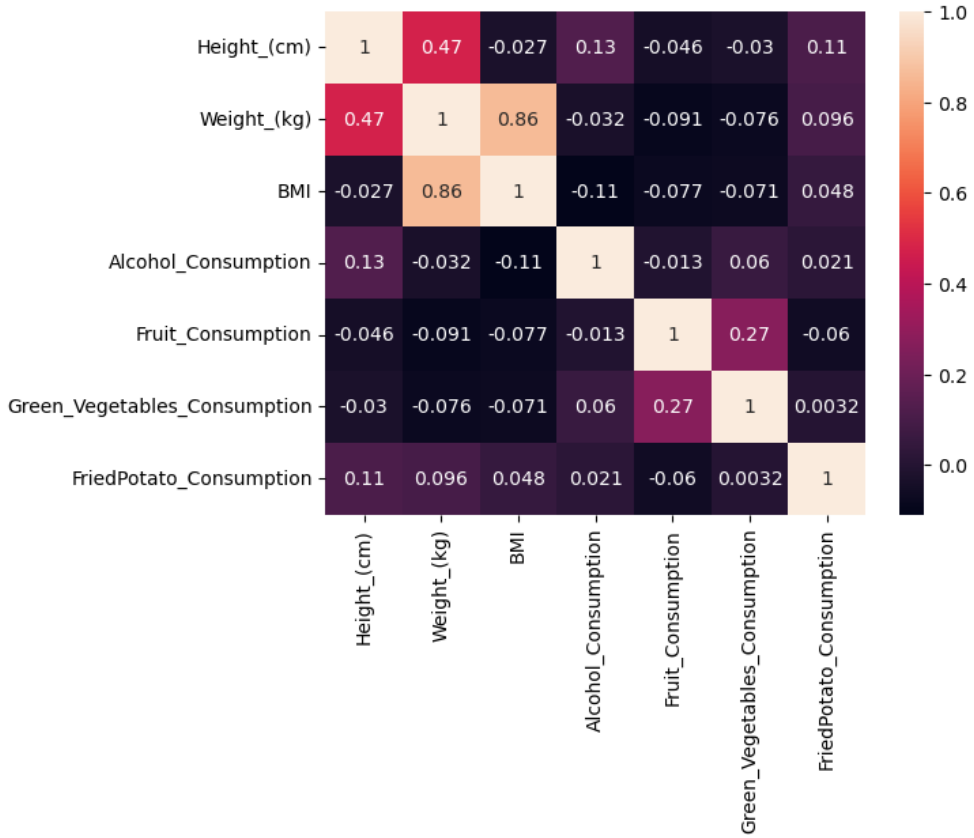
In [40]:
```
sns.countplot(x = df['Skin_Cancer'], hue= df['Sex'])
```

Out[40]:
```
<Axes: xlabel='Skin_Cancer', ylabel='count'>
```



In [41]:
```
sns.heatmap(df.corr(method='pearson'), annot= True)
```

Out[41]:
```
<Axes: >
```



In [42]:
```
data = df.copy()
```

Logistic Regression

In [43]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier , ExtraTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score
```

In [44]:
```
lab = LabelEncoder()
for i in cat:
    data[i] = lab.fit_transform(data[i])

data.head()
```

Out[44]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabetes	Arthritis	Sex	Age_Category	Height_(cm)	Weight_(kg)	BMI	Smoking_History	Alcohol_Consumption	Fruit_Cons
0	3	2	0	0	0	0	0	0	1	0	10	150.0	32.66	14.54	1	0.0	
1	4	4	0	1	0	0	0	2	0	0	10	165.0	77.11	28.29	0	0.0	
2	4	4	1	0	0	0	0	2	0	0	8	163.0	88.45	33.47	0	4.0	
3	3	4	1	1	0	0	0	2	0	1	11	180.0	93.44	28.73	0	0.0	
4	2	4	0	0	0	0	0	0	0	1	12	191.0	88.45	24.37	1	0.0	

In [45]:
```
x = data.drop('Heart_Disease', axis= 'columns')
y= data['Heart_Disease']
```

In [46]:
```
smo = SMOTE(random_state = 5)
```

In [47]:
```
x_bal , y_bal = smo.fit_resample(x,y)
```

In [48]:
```
X_train , x_test, Y_train, y_test = train_test_split(x_bal, y_bal, test_size = 0.3, random_state = 5)
```

In [49]:
```
log = LogisticRegression()
```

In [50]:
```
log.fit(X_train, Y_train)
```

Out[50]:
```
LogisticRegression
LogisticRegression()
```

In [51]:
```
pred = log.predict(x_test)
```

In [52]:
```
prob = log.predict_proba(x_test)
```

In [53]:
```
print(confusion_matrix(y_test, pred))
print('\n')
print(classification_report(y_test, pred))
print('\n')
print(roc_auc_score(y_test, prob[:,1]))
```

```
[[62040 23073]
 [20027 65190]]


              precision    recall  f1-score   support

           0       0.76      0.73      0.74     85113
           1       0.74      0.76      0.75     85217

    accuracy                           0.75    170330
   macro avg       0.75      0.75      0.75    170330
weighted avg       0.75      0.75      0.75    170330


0.8230475424891888
```

Decision Tree

In [54]:
```
dtree = DecisionTreeClassifier()
```

In [55]:
```
dtree.fit(X_train,Y_train)
```

Out[55]:
```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

In [56]:
```
tree_pred = dtree.predict(x_test)
tree_prob = dtree.predict_proba(x_test)
```

In [57]:
```
print(confusion_matrix(y_test, tree_pred))
print('\n')
print(classification_report(y_test, tree_pred))
print(roc_auc_score(y_test, tree_prob[:,1]))
```

```
[[73573 11540]
 [ 8744 76473]]


              precision    recall  f1-score   support

           0       0.89      0.86      0.88     85113
           1       0.87      0.90      0.88     85217

    accuracy                           0.88    170330
   macro avg       0.88      0.88      0.88    170330
weighted avg       0.88      0.88      0.88    170330

0.8809112107563303
```

In [ ]: