In [1]:
```
#import standard 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]:
```
# Load the dataset
data=pd.read_csv('/kaggle/input/heart-attack-in-youth-vs-adult-in-americastate/
data.head()
```

Out[2]:

	ID	Age_Group	Gender	Ethnicity	Smoking_Status	Alcohol_Consumption	Diet_Quality	Chole
0	1	Youth	Female	Native American	Current Smoker	Moderate	Average	
1	2	Adult	Female	Native American	Non-smoker	Moderate	Average	
2	3	Adult	Male	Native American	Former Smoker	Moderate	Average	
3	4	Adult	Female	Black	Non-smoker	Moderate	Average	
4	5	Youth	Male	White	Non-smoker	Moderate	Average	

Basic Data Information

In [3]:
```
def basic_information(df):
    # Checking the data shape
    shape=df.shape
    print(f'The data set contains {shape[0]} values and {shape[1]} columns')
    # Checking the duplicate values
    duplicate=df.duplicated().sum()
    print(f'The duplicated values in the dataset {duplicate}')
    # Checking the null values
    null=df.isna().sum()
    print(null)
    # Checking the data information
    df.info()
# Apply the function
basic_information(data)
```

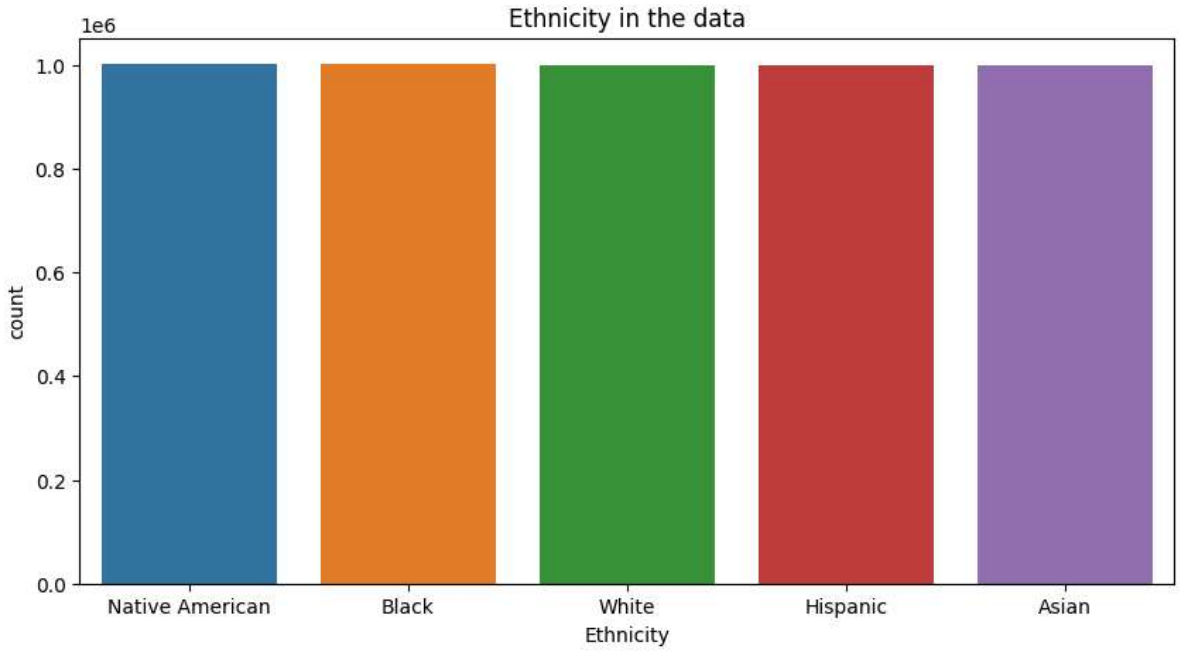
```
The data set contains 5000000 values and 20 columns
The duplicated values in the dataset 0
ID                          0
Age_Group                   0
Gender                      0
Ethnicity                   0
Smoking_Status              0
Alcohol_Consumption    1998915
Diet_Quality                0
Cholesterol_Level           0
Blood_Pressure              0
BMI                         0
Physical_Activity           0
Stress_Level                0
Family_History              0
Diabetes                    0
Air_Quality_Index           0
Income_Level                0
Sleep_Hours                 0
Heart_Rate                  0
Medication_Status           0
Heart_Attack                0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000000 entries, 0 to 4999999
Data columns (total 20 columns):
 #   Column               Dtype  
---  ------               -----  
 0   ID                   int64  
 1   Age_Group            object 
 2   Gender               object 
 3   Ethnicity            object 
 4   Smoking_Status       object 
 5   Alcohol_Consumption  object 
 6   Diet_Quality         object 
 7   Cholesterol_Level    int64  
 8   Blood_Pressure       int64  
 9   BMI                  float64
 10  Physical_Activity    int64  
 11  Stress_Level         object 
 12  Family_History       int64  
 13  Diabetes             int64  
 14  Air_Quality_Index    object 
 15  Income_Level         object 
 16  Sleep_Hours          float64
 17  Heart_Rate           int64  
 18  Medication_Status    int64  
 19  Heart_Attack         int64  
dtypes: float64(2), int64(9), object(9)
memory usage: 762.9+ MB
```

In [4]:
```
# information
data.describe().style.background_gradient(cmap='winter_r')
```

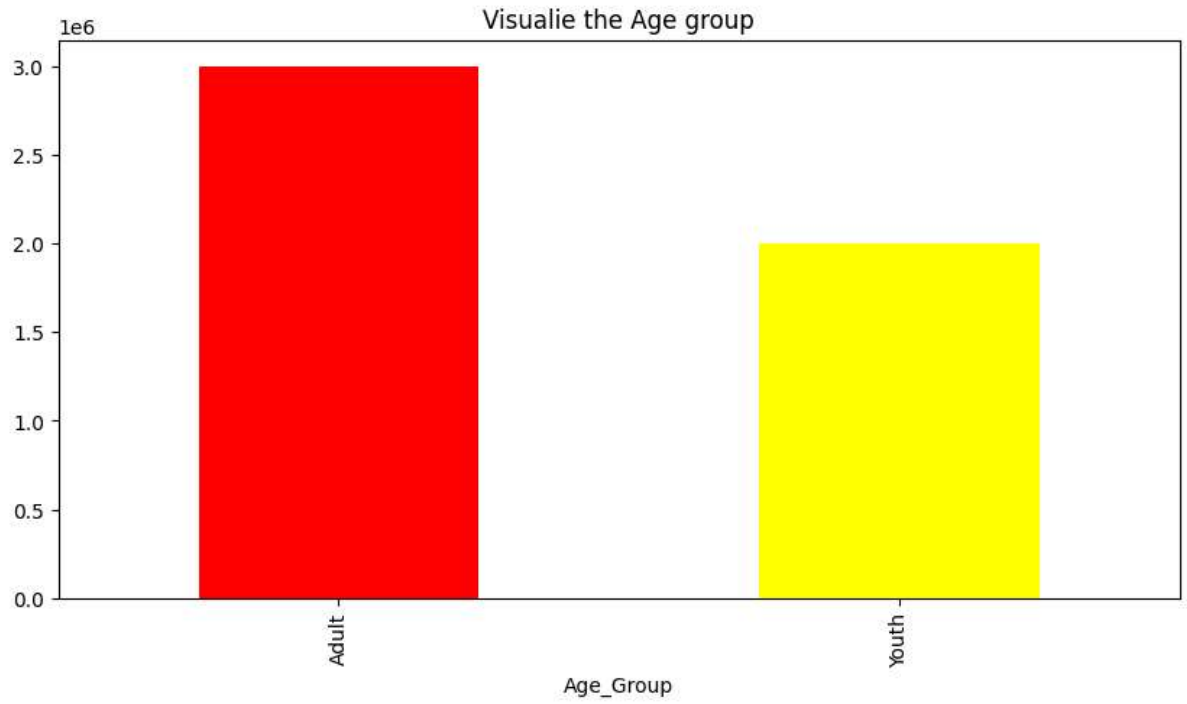
Out[4]:

	ID	Cholesterol_Level	Blood_Pressure	BMI	Physical_Activity	Fa
count	5000000.000000	5000000.000000	5000000.000000	5000000.000000	5000000.000000	500
mean	2500000.500000	209.467784	134.506569	28.999215	149.559525	
std	1443375.817312	51.953050	25.977452	6.349140	86.586017	
min	1.000000	120.000000	90.000000	18.000000	0.000000	
25%	1250000.750000	164.000000	112.000000	23.500000	75.000000	
50%	2500000.500000	209.000000	135.000000	29.000000	150.000000	
75%	3750000.250000	254.000000	157.000000	34.500000	225.000000	
max	5000000.000000	299.000000	179.000000	40.000000	299.000000	

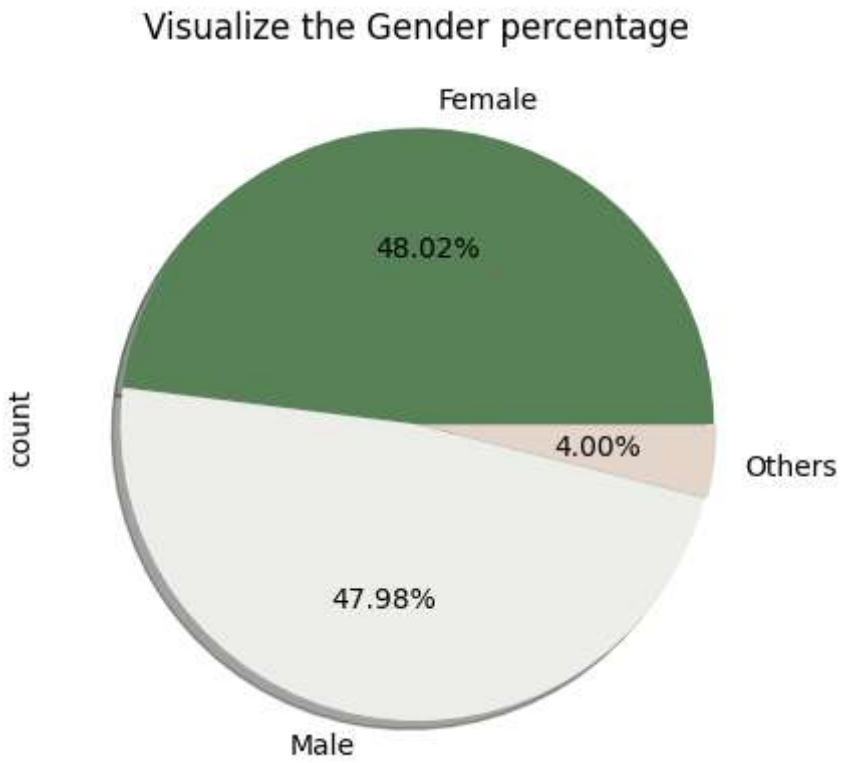
In [5]:
```
plt.figure(figsize=(10,5))
sns.countplot(data,x='Ethnicity')
plt.title('Ethnicity in the data')
plt.show()
```



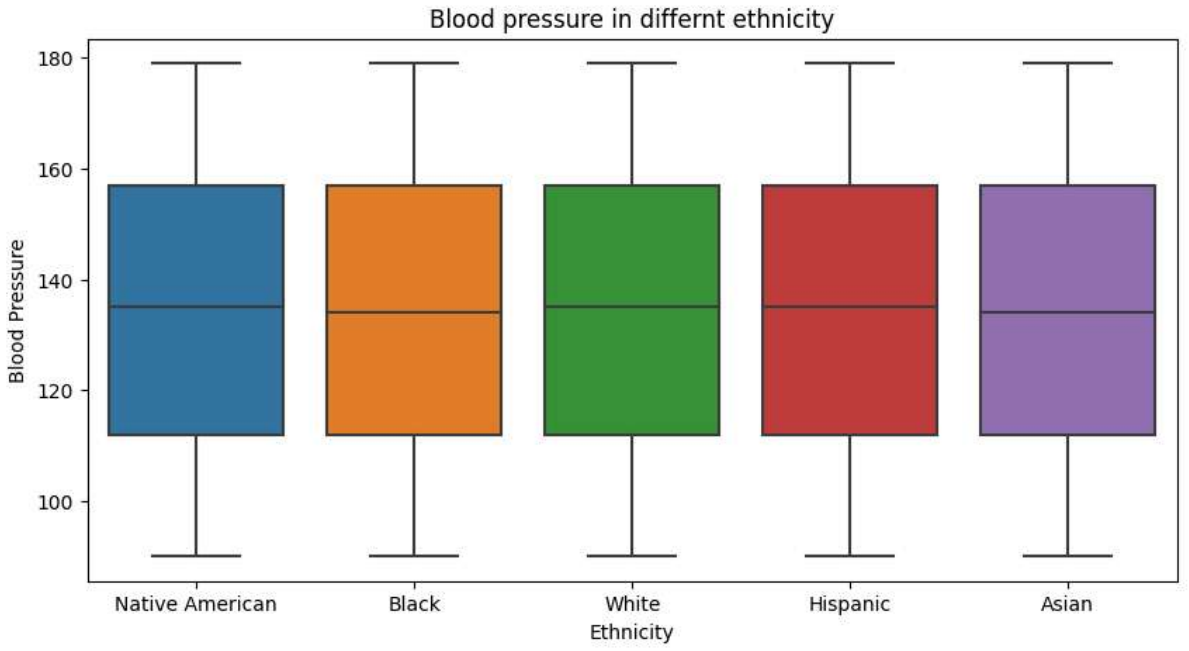
In [6]:
```
age=data['Age_Group'].value_counts()
plt.figure(figsize=(10,5))
age.plot(kind='bar',title='Visualie the Age group',color=['red','yellow'])
plt.show()
```



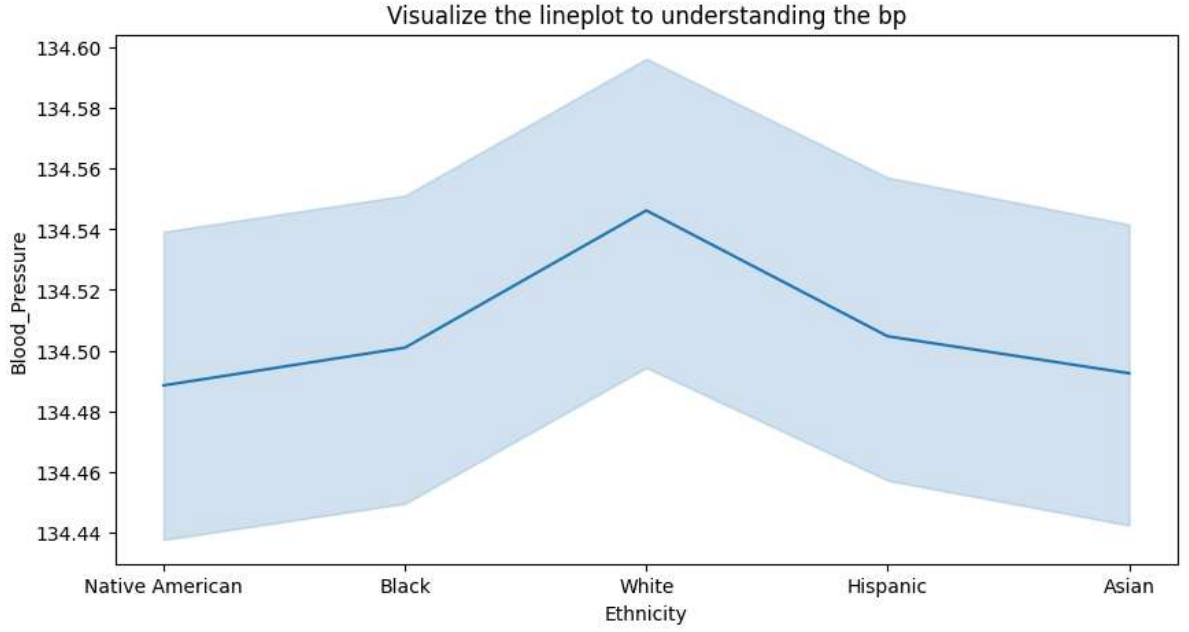
In [7]:
```
gender=data['Gender'].value_counts()
gender.plot(kind='pie',explode=[0,0,0.01],
    labels=['Female','Male','Others'],
    colors=['#588157','#ededed9','#e3d5ca'],
    autopct='%1.2f%%',
    shadow=True,
)
plt.title('Visualize the Gender percentage')
plt.show()
```



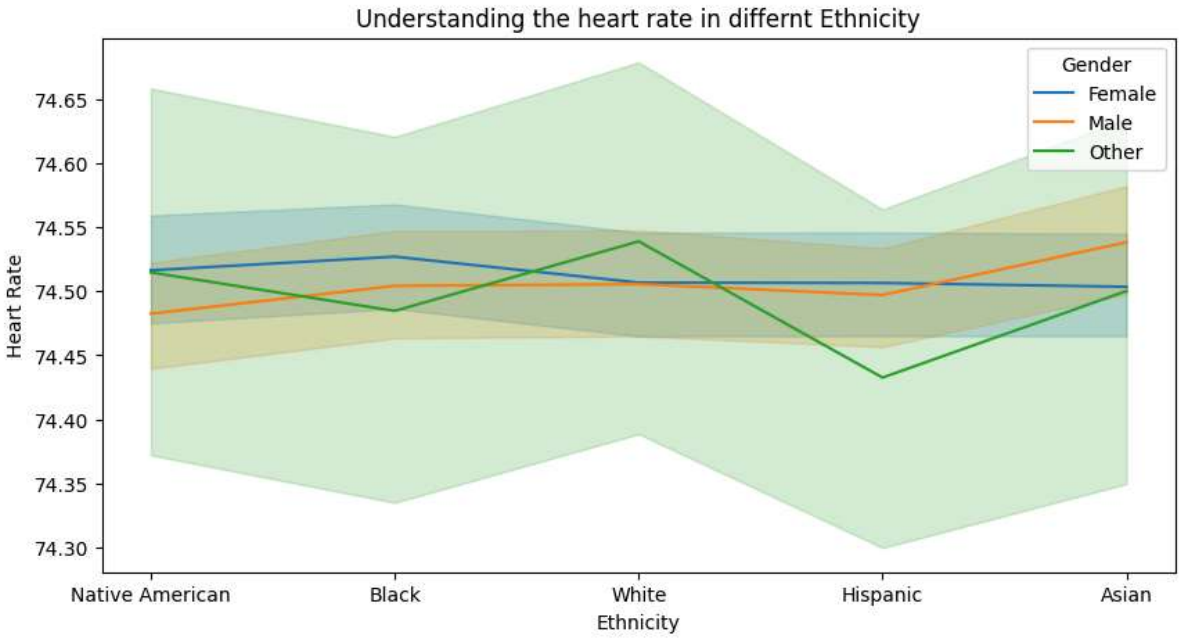
In [8]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=data,x='Ethnicity',y='Blood_Pressure')
plt.title('Blood pressure in differnt ethnicity')
plt.xlabel('Ethnicity')
plt.ylabel('Blood Pressure')
plt.show()
```



In [9]:
```
plt.figure(figsize=(10,5))
sns.lineplot(data,x='Ethnicity',y='Blood_Pressure')
plt.title('Visualize the lineplot to understanding the bp')
plt.xlabel('Ethnicity')
plt.ylabel('Blood_Pressure')
plt.show()
```



In [10]:
```
plt.figure(figsize=(10,5))
sns.lineplot(data=data,x='Ethnicity',y='Heart_Rate',hue='Gender')
plt.title('Understanding the heart rate in differnt Ethnicity')
plt.xlabel('Ethnicity')
plt.ylabel('Heart Rate')
plt.show()
```

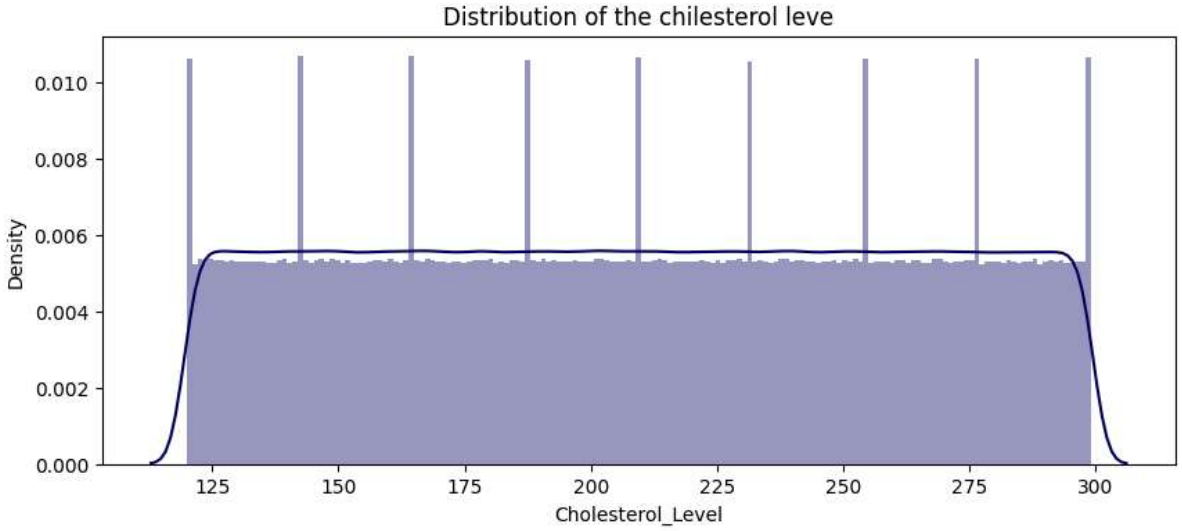


In [11]:
```
ethnicity=pd.DataFrame(data.groupby('Gender')['Ethnicity'].value_counts()).uns
ethnicity.style.background_gradient(cmap='cubehelix')
```

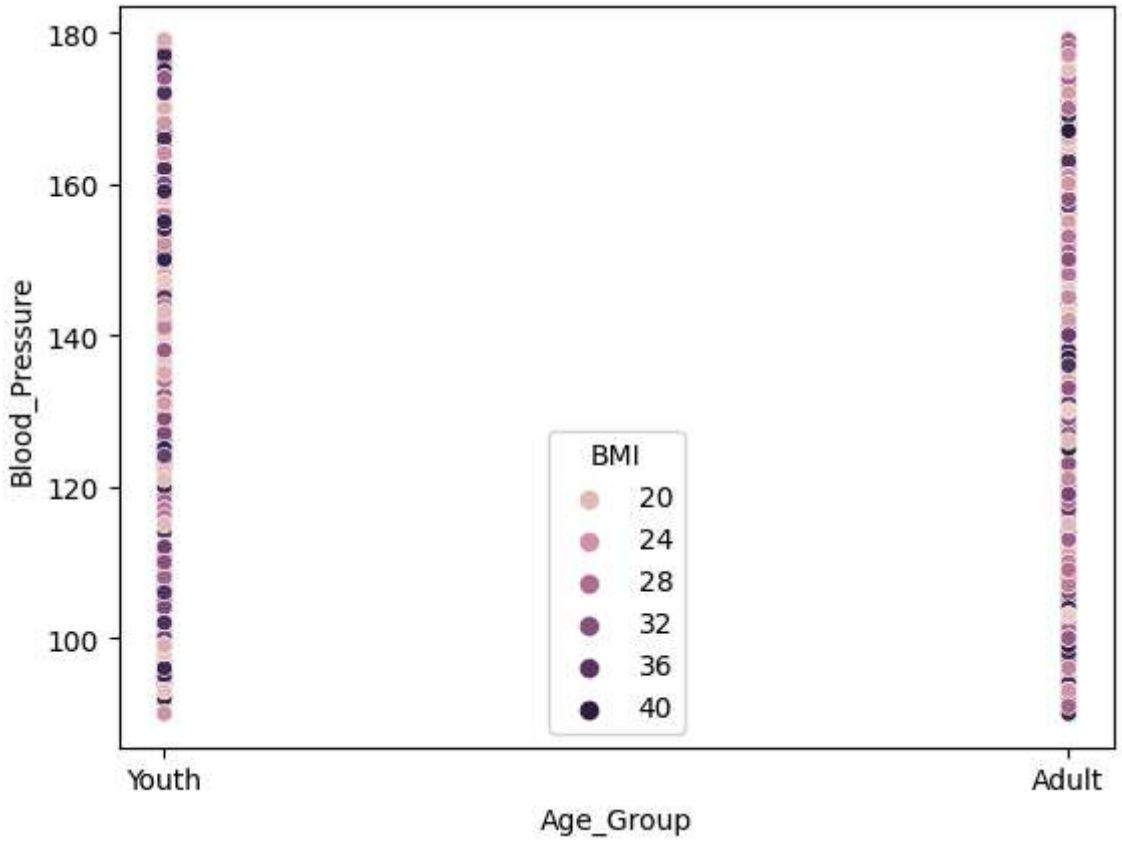
Out[11]:

	count				
Ethnicity	**Asian**	**Black**	**Hispanic**	**Native American**	**White**
Gender					
Female	480329	480689	479502	480154	480318
Male	478766	480760	479413	480527	479770
Other	40125	40079	39695	39733	40140

In [12]:
```
plt.figure(figsize=(10,4))
sns.distplot(data['Cholesterol_Level'],bins='auto',kde=True,color='#03045e')
plt.title('Distribution of the chilesterol leve')
plt.show()
```



In [13]:
```
sns.scatterplot(data=data,x='Age_Group',y='Blood_Pressure',hue='BMI')
plt.show()
```



In [14]:
```
alcohol_counsumption=pd.DataFrame(data.groupby('Age_Group')['Alcohol_Consumptio
alcohol_counsumption.style.background_gradient(cmap='gist_earth')
```

Out[14]:

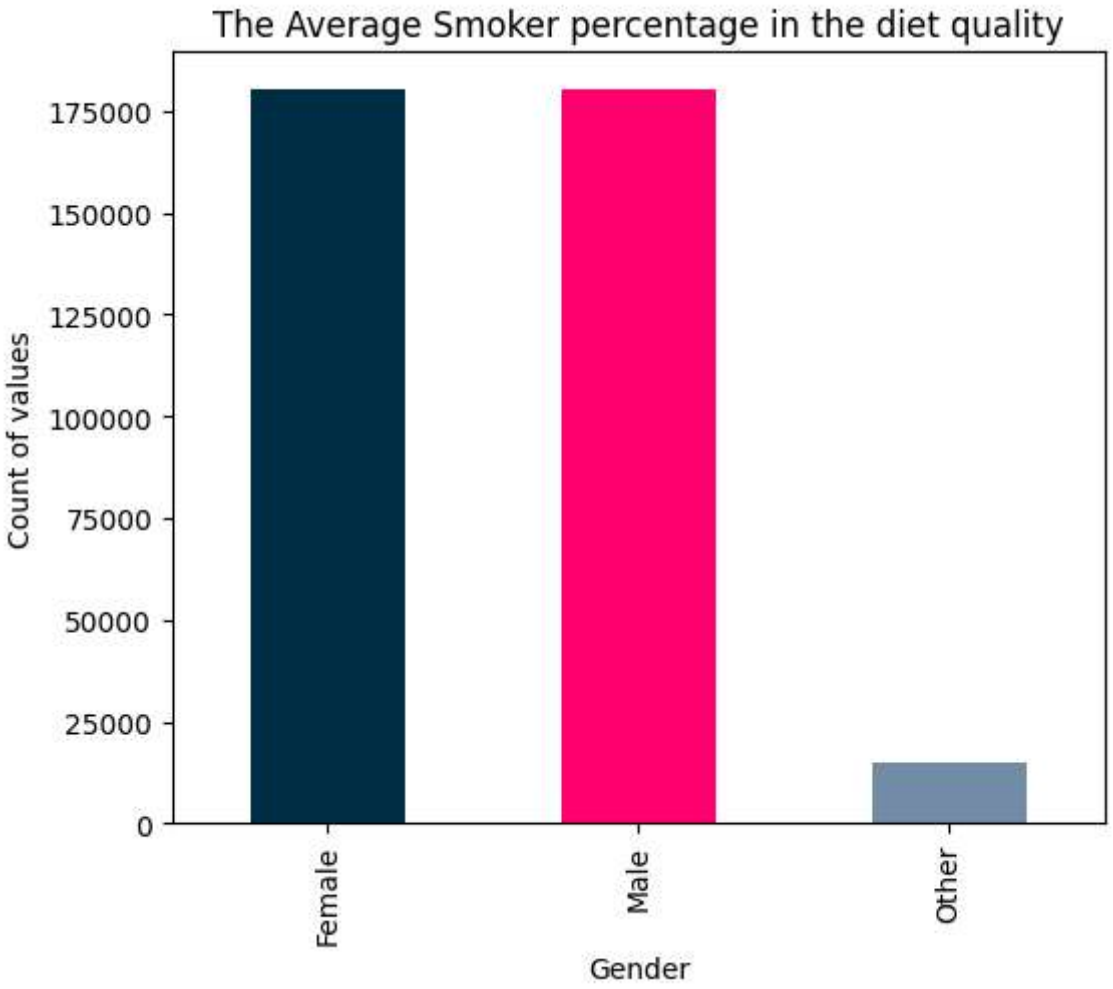
	count	
Alcohol_Consumption	**High**	**Moderate**
Age_Group		
Adult	299869	1501009
Youth	199656	1000551

In [15]:
```
average_cholestrol_bp=pd.DataFrame(data.groupby('Ethnicity')[['Cholesterol_Leve
average_cholestrol_bp
```

Out[15]:

	Cholesterol_Level	Blood_Pressure
Ethnicity		
Asian	209.520205	134.492491
Black	209.473224	134.500936
Hispanic	209.468619	134.504690
Native American	209.417422	134.488486
White	209.459508	134.546237

In [16]:
```
diet=data[data['Diet_Quality']=='Average']
diet=diet[data['Smoking_Status']=='Current Smoker']
diet['Gender'].value_counts()\
.plot(kind='bar',figsize=(6,5),color=['#023047','#ff006e','#778da9'])
plt.title('The Average Smoker percentage in the diet quality')
plt.xlabel('Gender')
plt.ylabel('Count of values')
plt.show()
```



In [17]:
```
average_sleep=pd.DataFrame(data.groupby('Age_Group')['Sleep_Hours'].mean())
average_sleep.style.background_gradient(cmap='afmhot_r')
```

Out[17]:

Age_Group	Sleep_Hours
Adult	6.999947
Youth	6.999736

In [18]:
```
min_max_cholesterol=pd.DataFrame(data.pivot_table(values='Cholesterol_Level',i
min_max_cholesterol.style.background_gradient(cmap='crest_r')
```

Out[18]:

	min			max			mean		
Gender	Female	Male	Other	Female	Male	Other	Female	Male	Other
Ethnicity									
Asian	120	120	120	299	299	299	209.540074	209.514086	209.355364
Black	120	120	120	299	299	299	209.444895	209.491291	209.596272
Hispanic	120	120	120	299	299	299	209.550897	209.385976	209.472856
Native American	120	120	120	299	299	299	209.364802	209.486530	209.217527
White	120	120	120	299	299	299	209.383092	209.516147	209.696936

In [19]:
```
average_physical_heart=pd.DataFrame(data.groupby('Ethnicity')[['Physical_Activ
average_physical_heart.style.background_gradient(cmap='nipy_spectral')
```

Out[19]:

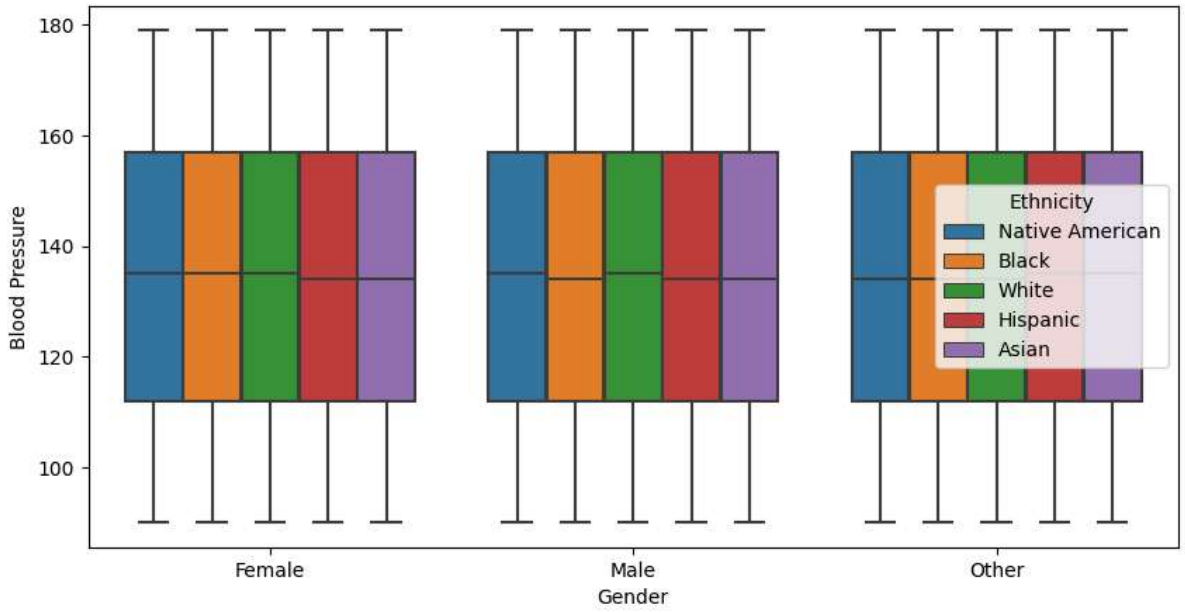
	Physical_Activity	Heart_Rate
Ethnicity		
Asian	149.779362	74.519952
Black	149.503763	74.514309
Hispanic	149.401939	74.498998
Native American	149.567694	74.499988
White	149.544906	74.507424

In [20]:
```
strees_level=pd.DataFrame(data.groupby('Ethnicity')['Stress_Level'].value_coun
strees_level.style.background_gradient(cmap='plasma')
```

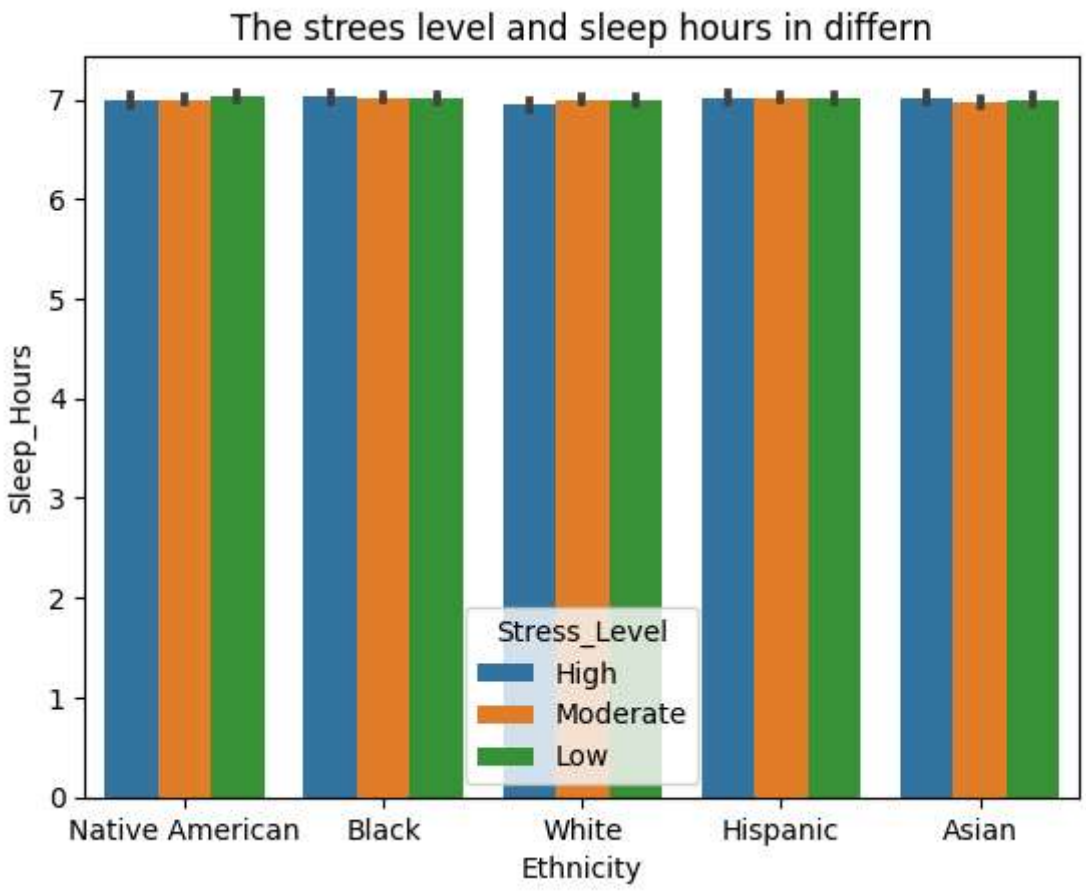
Out[20]:

Stress_Level	High	Low	Moderate
Ethnicity			
Asian	200088	299260	499872
Black	200211	300560	500757
Hispanic	199787	300057	498766
Native American	199798	299180	501436
White	199309	300319	500600

In [21]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=data,x='Gender',y='Blood_Pressure',hue='Ethnicity')
plt.xlabel('Gender')
plt.ylabel('Blood Pressure')
plt.show()
```



In [22]:
```
sns.barplot(data=data[:100000],x='Ethnicity',y='Sleep_Hours',hue='Stress_Level
plt.title('The strees level and sleep hours in differn')
plt.show()
```



In [23]:
```
data.head(1)
```

Out[23]:

	ID	Age_Group	Gender	Ethnicity	Smoking_Status	Alcohol_Consumption	Diet_Quality	Chole
0	1	Youth	Female	Native American	Current Smoker	Moderate	Average	

In [24]:
```
cate=data.select_dtypes(include='object')
cate.head()
```

Out[24]:

	Age_Group	Gender	Ethnicity	Smoking_Status	Alcohol_Consumption	Diet_Quality	Stress_Le
0	Youth	Female	Native American	Current Smoker	Moderate	Average	H
1	Adult	Female	Native American	Non-smoker	Moderate	Average	Moder
2	Adult	Male	Native American	Former Smoker	Moderate	Average	H
3	Adult	Female	Black	Non-smoker	Moderate	Average	H
4	Youth	Male	White	Non-smoker	Moderate	Average	H

In [25]:
```
from sklearn.preprocessing import LabelEncoder,MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report
```

In [26]:
```
label=LabelEncoder()
for columns in cate.columns:
    data[columns]=label.fit_transform(data[columns])
data.head()
```

Out[26]:

	ID	Age_Group	Gender	Ethnicity	Smoking_Status	Alcohol_Consumption	Diet_Quality	Chole
0	1	1	0	3	0	1	0	
1	2	0	0	3	2	1	0	
2	3	0	1	3	1	1	0	
3	4	0	0	1	2	1	0	
4	5	1	1	4	2	1	0	

In [27]:
```
X=data.drop(['Heart_Attack'],axis=1)
y=data['Heart_Attack']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state
```

In [28]:
```
logistic=LogisticRegression()
logistic.fit(X_train,y_train)
logistic_pred=logistic.predict(X_test)
logistic_pred
```

Out[28]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [29]:
```
print(f'The model accuracy score is {accuracy_score(logistic_pred,y_test)*100:
print(classification_report(logistic_pred,y_test))
```

```
The model accuracy score is 89.99
              precision    recall  f1-score   support

           0       1.00      0.90      0.95   1250000
           1       0.00      0.00      0.00         0

    accuracy                           0.90   1250000
   macro avg       0.50      0.45      0.47   1250000
weighted avg       1.00      0.90      0.95   1250000
```

In [30]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
tree=DecisionTreeClassifier()
tree.fit(X_train,y_train)
tree_pred=tree.predict(X_test)
print(f'The decisiontree model accuracy_score {accuracy_score(tree_pred,y_test
print(classification_report(tree_pred,y_test))
```

```
The decisiontree model accuracy_score 80.03
              precision    recall  f1-score   support

           0       0.88      0.90      0.89   1094362
           1       0.12      0.10      0.11    155638

    accuracy                           0.80   1250000
   macro avg       0.50      0.50      0.50   1250000
weighted avg       0.78      0.80      0.79   1250000
```

In [ ]:
```
random=RandomForestClassifier()
random.fit(X_train,y_train)
random_pred=random.predict(X_test)
print(f'The randomForest classificer score {accuracy_score(random_pred,y_test)
print(classification_report(random_pred,y_test))
```

In [ ]:
```

```