In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [8]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [9]:
```
df=pd.read_csv("oral_cancer_prediction_dataset.csv")
df
```

Out[9]:

	ID	Country	Gender	Age	Tobacco_Use	Alcohol_Use	Socioeconomic_St
0	1	Ethiopia	Male	34	1	1	
1	2	Turkey	Female	84	1	1	
2	3	Turkey	Female	62	1	1	Mi
3	4	Tanzania	Male	48	1	1	Mi
4	5	France	Male	26	1	1	Mi
...	...	...	...	...	...	...	
160287	160288	United Kingdom	Female	53	0	1	Mi
160288	160289	Brazil	Female	81	0	0	
160289	160290	Nigeria	Male	59	0	1	
160290	160291	Philippines	Female	43	0	0	
160291	160292	Nigeria	Female	20	0	0	

160292 rows × 11 columns

In [10]:
```
df.head()
```

Out[10]:

	ID	Country	Gender	Age	Tobacco_Use	Alcohol_Use	Socioeconomic_Status	Diagnc
0	1	Ethiopia	Male	34	1	1	High	
1	2	Turkey	Female	84	1	1	High	
2	3	Turkey	Female	62	1	1	Middle	
3	4	Tanzania	Male	48	1	1	Middle	
4	5	France	Male	26	1	1	Middle	

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 160292 entries, 0 to 160291
Data columns (total 11 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   ID                    160292 non-null  int64
 1   Country               160292 non-null  object
 2   Gender                160292 non-null  object
 3   Age                   160292 non-null  int64
 4   Tobacco_Use           160292 non-null  int64
 5   Alcohol_Use           160292 non-null  int64
 6   Socioeconomic_Status  160292 non-null  object
 7   Diagnosis_Stage       160292 non-null  object
 8   Treatment_Type        160292 non-null  object
 9   Survival_Rate         160292 non-null  float64
 10  HPV_Related           160292 non-null  int64
dtypes: float64(1), int64(5), object(5)
memory usage: 13.5+ MB
```

In [12]:
```
df.describe()
```

Out[12]:

	ID	Age	Tobacco_Use	Alcohol_Use	Survival_Rate	HI
count	160292.000000	160292.000000	160292.000000	160292.000000	160292.000000	160.
mean	80146.500000	46.564102	0.601677	0.499638	0.599990	
std	46272.459012	20.594431	0.489554	0.500001	0.172882	
min	1.000000	20.000000	0.000000	0.000000	0.300002	
25%	40073.750000	29.000000	0.000000	0.000000	0.450680	
50%	80146.500000	39.000000	1.000000	0.000000	0.599586	
75%	120219.250000	64.000000	1.000000	1.000000	0.749291	
max	160292.000000	89.000000	1.000000	1.000000	0.899992	

In [13]:
```
df.dtypes
```

Out[13]:
```
ID                      int64
Country                object
Gender                 object
Age                     int64
Tobacco_Use             int64
Alcohol_Use             int64
Socioeconomic_Status   object
Diagnosis_Stage        object
Treatment_Type         object
Survival_Rate         float64
HPV_Related             int64
dtype: object
```

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

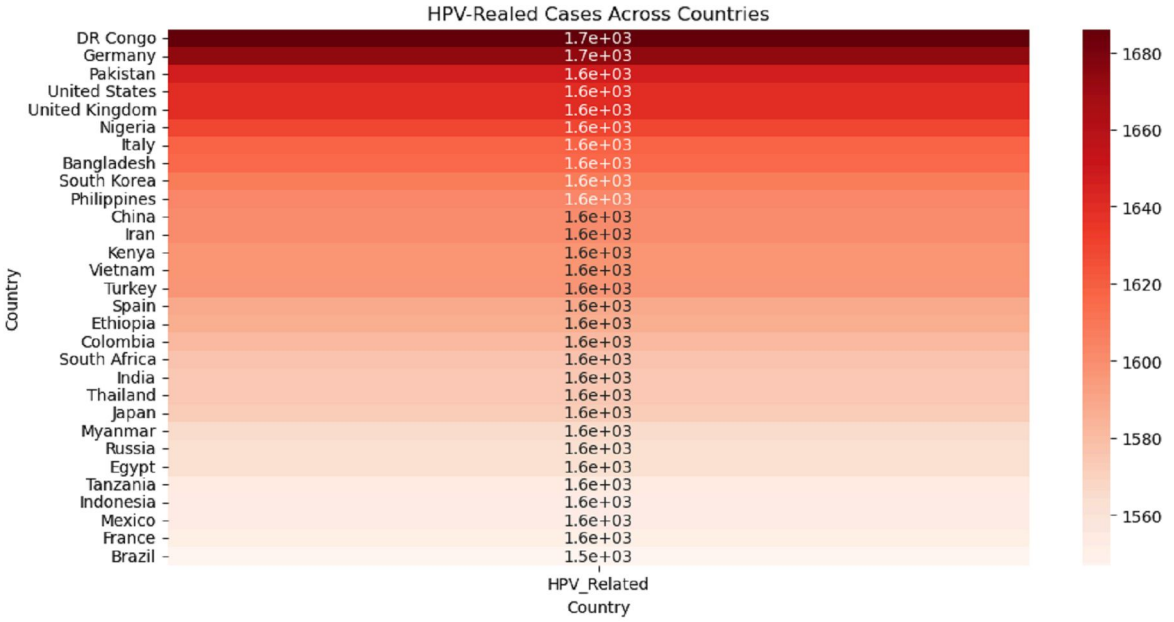
Out[14]:
```
(160292, 11)
```

In [15]:
```
df.columns
```

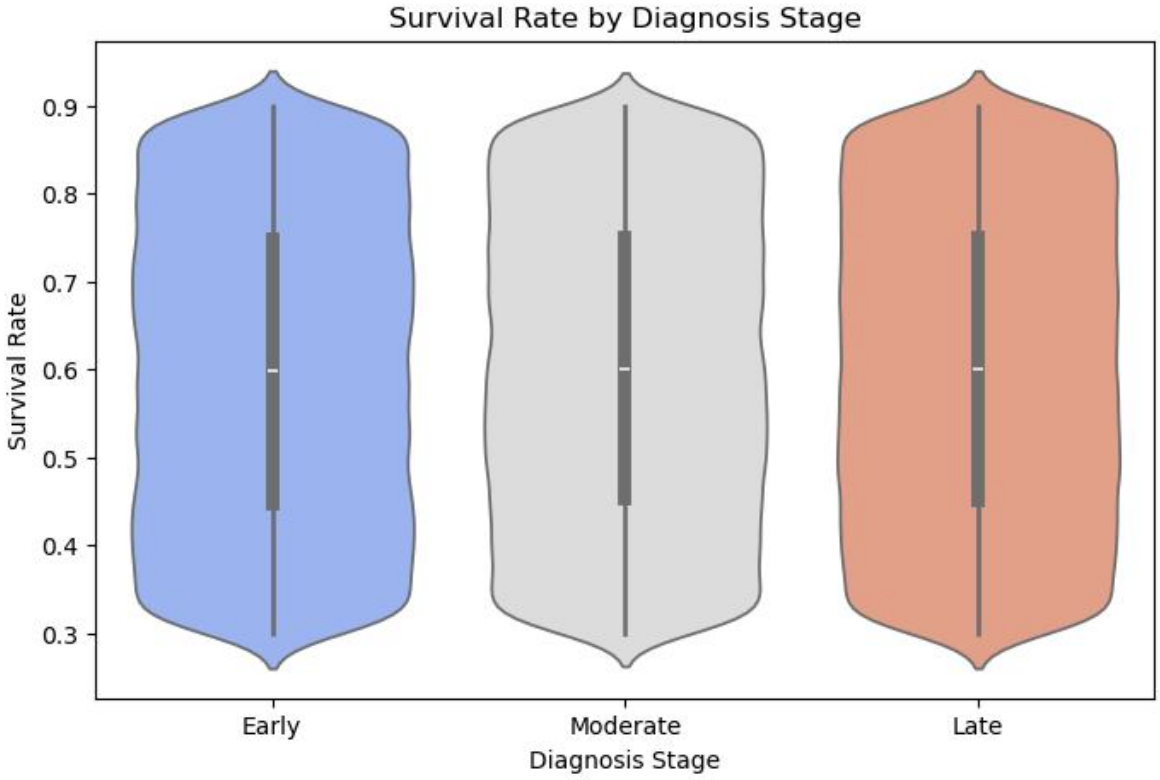
Out[15]:
```
Index(['ID', 'Country', 'Gender', 'Age', 'Tobacco_Use', 'Alcohol_Use',
       'Socioeconomic_Status', 'Diagnosis_Stage', 'Treatment_Type',
       'Survival_Rate', 'HPV_Related'],
      dtype='object')
```

EDA

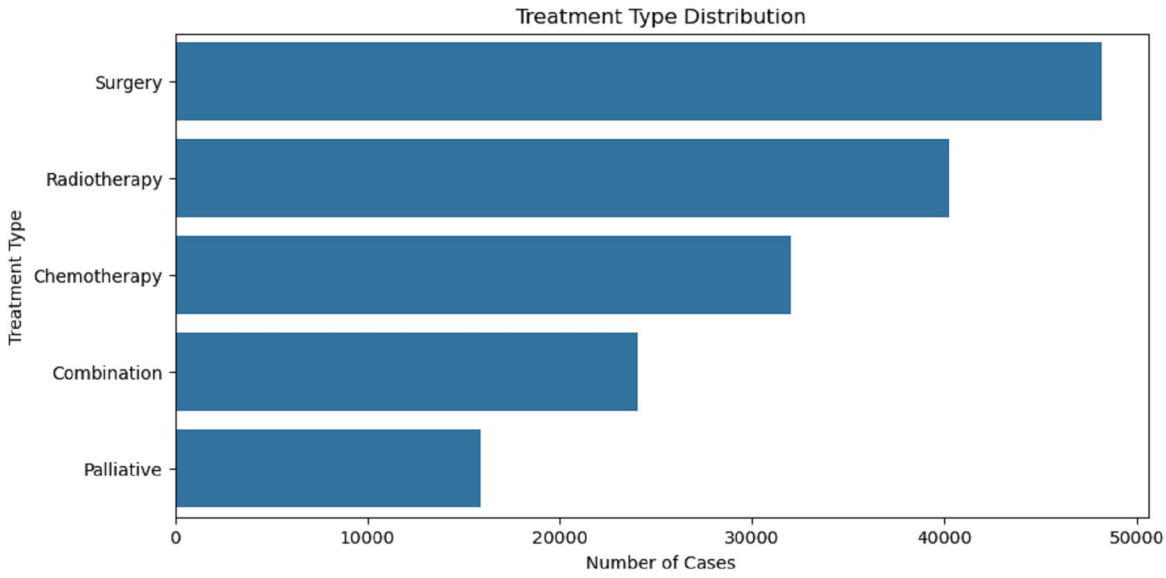
In [17]:
```
plt.figure(figsize=(12,6))
hpv_counts =df.groupby("Country")["HPV_Related"].sum().reset_index()
hpv_counts=hpv_counts.sort_values(by="HPV_Related",ascending=False)
sns.heatmap(hpv_counts.set_index("Country"),annot=True,cmap="Reds")
plt.title("HPV-Realed Cases Across Countries")
plt.xlabel("Country")
plt.show()
```



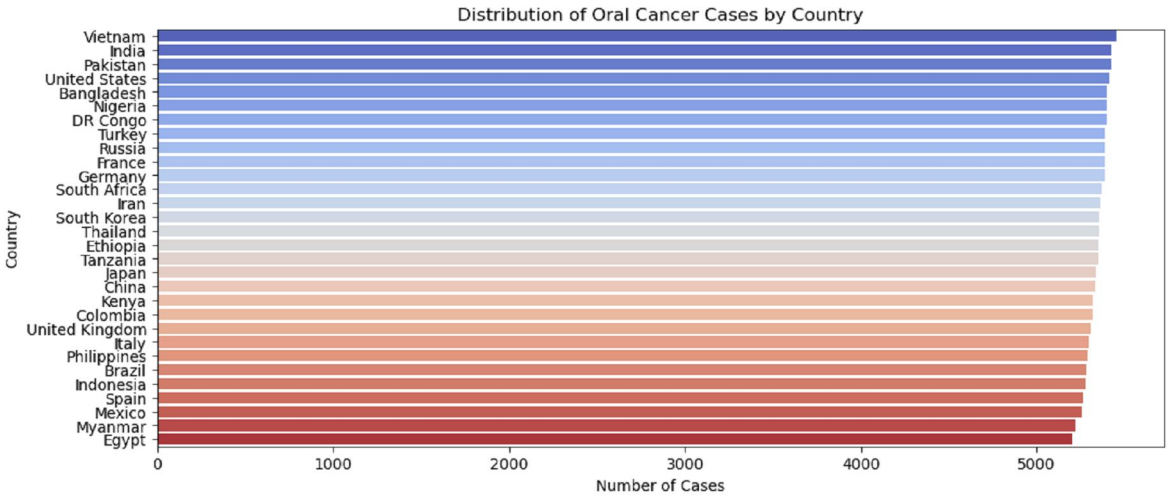
In [18]:
```
plt.figure(figsize=(8,5))
sns.violinplot(x="Diagnosis_Stage", y="Survival_Rate", data=df, palette="coolwar
plt.xlabel("Diagnosis Stage")
plt.ylabel("Survival Rate")
plt.title("Survival Rate by Diagnosis Stage")
plt.show()
```



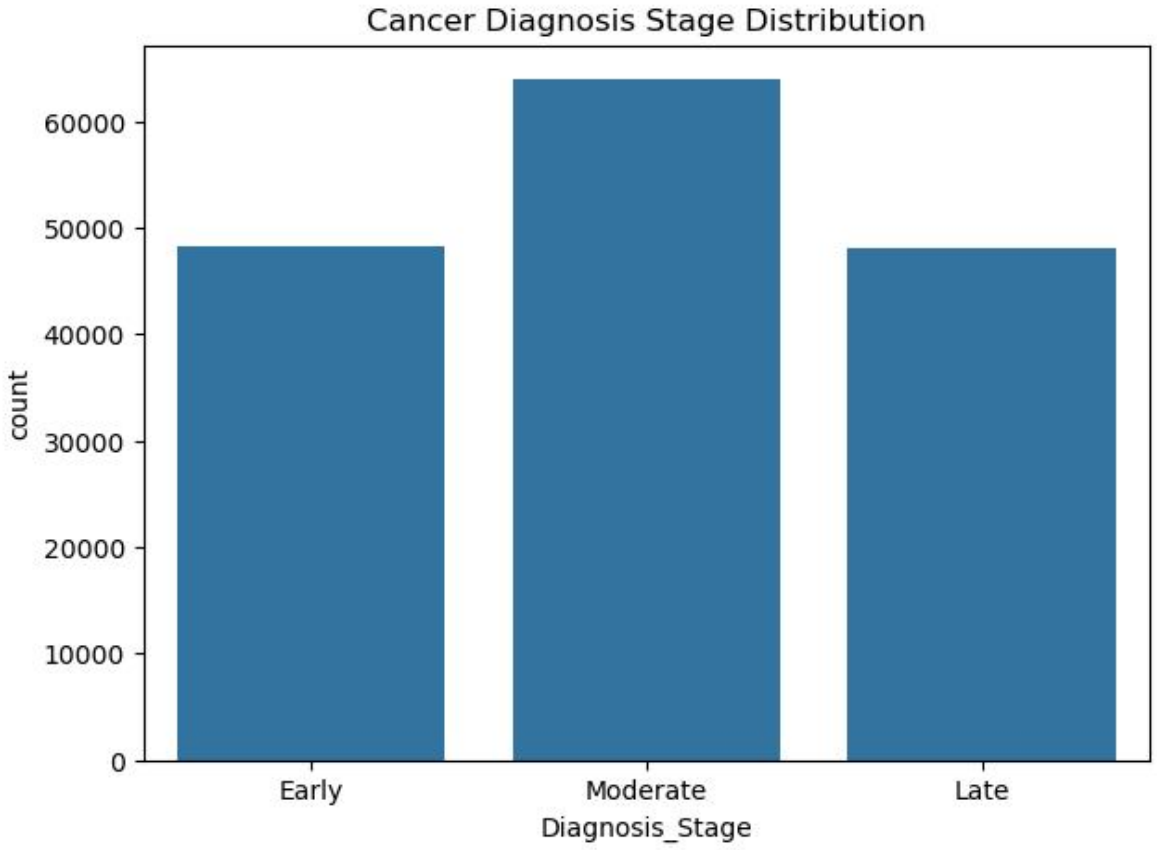
In [19]:
```
plt.figure(figsize=(10,5))
sns.countplot(y=df['Treatment_Type'], order=df['Treatment_Type'].value_counts().
plt.xlabel("Number of Cases")
plt.ylabel("Treatment Type")
plt.title("Treatment Type Distribution")
plt.show()
```



In [20]:
```
plt.figure(figsize=(12,5))
sns.countplot(y=df['Country'],order=df['Country'].value_counts().index,palette='
plt.xlabel("Number of Cases")
plt.ylabel("Country")
plt.title("Distribution of Oral Cancer Cases by Country")
plt.show()
```

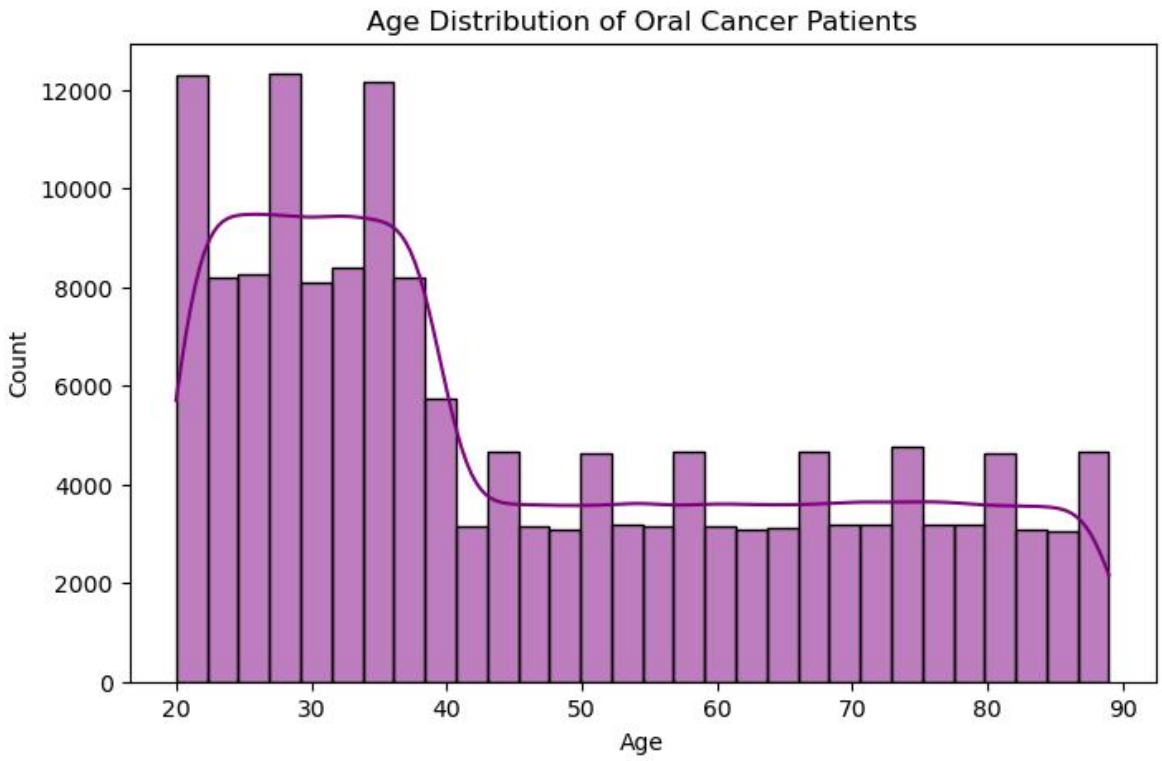


In [21]:
```
plt.figure(figsize=(7,5))
sns.countplot(x='Diagnosis_Stage', data=df, order=["Early", "Moderate", "Late"])
plt.title("Cancer Diagnosis Stage Distribution")
plt.show()
```

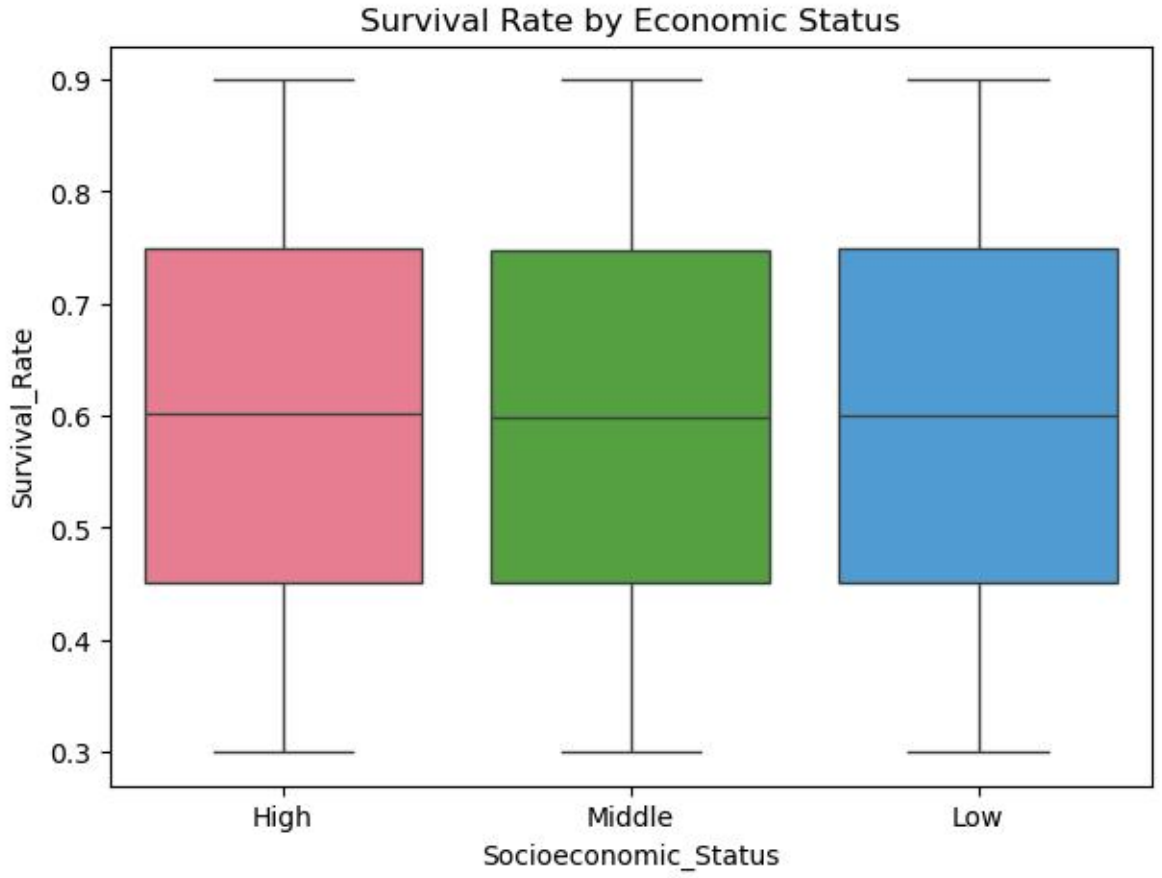


In [22]:
```
plt.figure(figsize=(8,5))
sns.histplot(df['Age'],bins=30,kde=True,color="purple")
plt.title("Age Distribution of Oral Cancer Patients")
plt.xlabel("Age")
```

Out[22]:
```
Text(0.5, 0, 'Age')
```

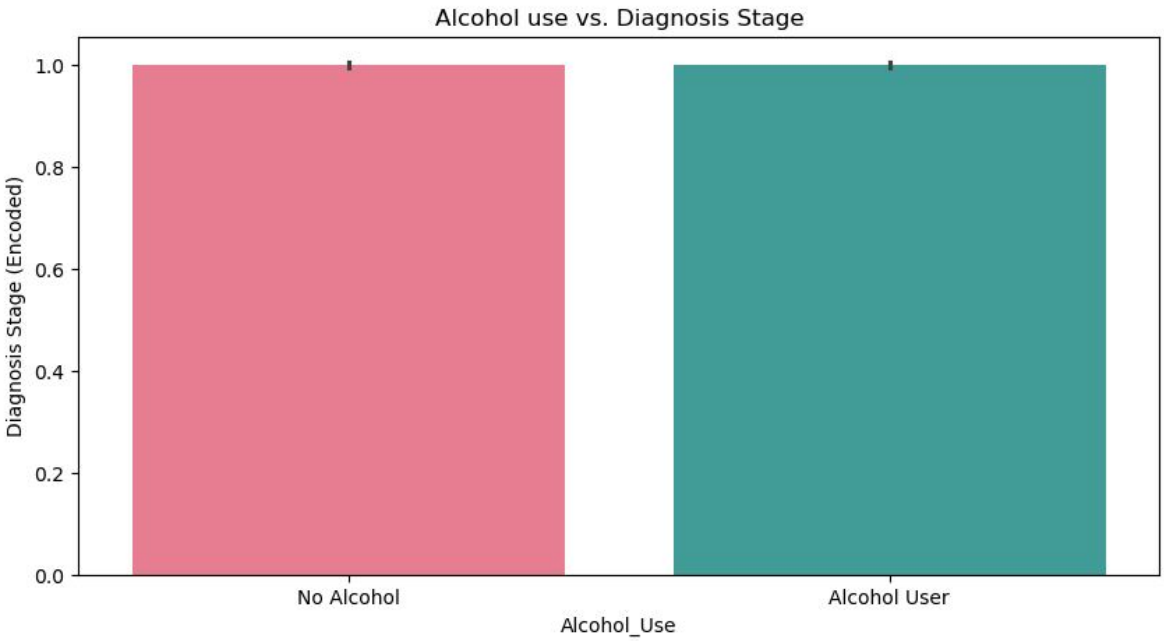


In [23]:
```
plt.figure(figsize=(7,5))
sns.boxplot(x="Socioeconomic_Status",y="Survival_Rate",data=df,palette="husl")
plt.title("Survival Rate by Economic Status")
plt.show()
```



In [24]:
```
plt.figure(figsize=(10,5))
sns.barplot(x="Alcohol_Use",y=df["Diagnosis_Stage"].factorize()[0],data=df,palet
plt.xticks([0,1],["No Alcohol","Alcohol User"])
plt.ylabel("Diagnosis Stage (Encoded)")
```

```
plt.title("Alcohol use vs. Diagnosis Stage")
plt.show()
```



Machine Learning

In [26]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [27]:
```
#Drop the id column
df=df.drop(columns=["ID"],errors='ignore')
```

In [28]:
```
cat=df.select_dtypes(include='object').columns
cat
```

Out[28]:
```
Index(['Country', 'Gender', 'Socioeconomic_Status', 'Diagnosis_Stage',
       'Treatment_Type'],
      dtype='object')
```

In [29]:
```
#Encode Categorical variables
label_encoder={}
categorical_columns=['Country', 'Gender', 'Socioeconomic_Status', 'Diagnosis_Sta
       'Treatment_Type']
for col in categorical_columns:
    le=LabelEncoder()
    df[col]=le.fit_transform(df[col])
    label_encoder[col]=le
```

In [30]:
```
x=df.drop(columns=["Survival_Rate"])
y=df["Survival_Rate"]
```

In [31]:
```
#Reduce Data Size for efficient training
df_sampled=df.sample(n=5000,random_state=42)
```

```
x_sampled=df_sampled.drop(columns=["Survival_Rate"])
y_sampled=df_sampled["Survival_Rate"]
```

In [32]:
```
#Split the dataset
x_train,x_test,y_train,y_test=train_test_split(x_sampled,y_sampled,test_size=0.2
```

In [33]:
```
#Scale the numerical features
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

In [34]:
```
#Define Models
models={
    "Linear Regression":LinearRegression(),
    "Random Forest":RandomForestRegressor(n_estimators=100,random_state=42),
    "Gradient Boosting":GradientBoostingRegressor(n_estimators=100,random_state=
    "Super Vector Regresssor":SVR(kernel='rbf')
}
```

In [35]:
```
#Train and Evaluate models
results={}
for name,model in models.items():
    model.fit(x_train,y_train)
    y_pred =model.predict(x_test)
    mae=mean_absolute_error(y_test,y_pred)
    mse=mean_squared_error(y_test,y_pred)
    r2=r2_score(y_test,y_pred)
    results[name]={"MAE":mae,"MSE":mse,"R2 Score":r2}
```

In [36]:
```
#Display the Results
result_df=pd.DataFrame(results).T
print(result_df)
```

```
                             MAE       MSE  R2 Score
Linear Regression       0.148577  0.029565 -0.004955
Random Forest           0.152366  0.032125 -0.091976
Gradient Boosting       0.149386  0.030161 -0.025209
Super Vector Regresssor 0.155642  0.033600 -0.142127
```

In [ ]: