# Dataset Information

This dataset comprises comprehensive health information for 2,105 patients diagnosed with Parkinson's Disease, each uniquely identified with IDs ranging from 3058 to 5162. The dataset includes demographic details, lifestyle factors, medical history, clinical measurements, cognitive and functional assessments, symptoms, and a diagnosis indicator. This dataset is valuable for researchers and data scientists aiming to explore factors associated with Parkinson's Disease, develop predictive models, and conduct statistical analyses.

## Patient Information

- PatientID: A unique identifier assigned to each patient (3058 to 516)s indicate greater impairment.

## Demographic Details

- Age: The age of the patients ranges from 50 to 90 years.
- Gender: Gender of the patients, where 0 represents Male and 1 represents Females

### Ethnicity: The ethnicity of the patients, coded as follow-:

- 0: Caucasian
- 1: African American
- 2: Asian
- 3: Other

### EducationLevel: The education level of the patients, coded as follows:

- 0: None
- 1: High School
- 2 : Bachelor's
- 3 : Higher

## Symptoms

- Tremor: Presence of tremor, where 0 indicates No and 1 indicates Yes.
- Rigidity: Presence of muscle rigidity, where 0 indicates No and 1 indicates Yes.
- Bradykinesia: Presence of bradykinesia (slowness of movement), where 0 indicates No and 1 indicates Yes.

- PosturalInstability: Presence of postural instability, where 0 indicates No and 1 indicates Yes.
- SpeechProblems: Presence of speech problems, where 0 indicates No and 1 indicates Yes.
- SleepDisorders: Presence of sleep disorders, where 0 indicates No and 1 indicates Yes.
- Constipation: Presence of constipation, where 0 indicates No and 1 indicates Yes.

## Lifestyle factors

- BMI: Body Mass Index of the patients, ranging from 15 to 40.
- Smoking: Smoking status, where 0 indicates No and 1 indicates Yes.
- AlcoholConsumption: Weekly alcohol consumption in units, ranging from 0 to 20.
- PhysicalActivity: Weekly physical activity in hours, ranging from 0 to 10.
- DietQuality: Diet quality score, ranging from 0 to 10.
- SleepQuality: Sleep quality score, ranging from 4 to 10.

## Medical History

- FamilyHistoryParkinsons: Family history of Parkinson's Disease, where 0 indicates No and 1 indicates Yes.
- TraumaticBrainInjury: History of traumatic brain injury, where 0 indicates No and 1 indicates Yes.
- Hypertension: Presence of hypertension, where 0 indicates No and 1 indicates Yes.
- Diabetes: Presence of diabetes, where 0 indicates No and 1 indicates Yes.
- Depression: Presence of depression, where 0 indicates No and 1 indicates Yes.
- Stroke: History of stroke, where 0 indicates No and 1 indicates Yes.

## Clinical Measurements

- SystolicBP: Systolic blood pressure, ranging from 90 to 180 mmHg.
- DiastolicBP: Diastolic blood pressure, ranging from 60 to 120 mmHg.
- CholesterolTotal: Total cholesterol levels, ranging from 150 to 300 mg/dL.
- CholesterolLDL: Low-density lipoprotein cholesterol levels, ranging from 50 to 200 mg/dL.
- CholesterolHDL: High-density lipoprotein cholesterol levels, ranging from 20 to 100 mg/dL.
- CholesterolTriglycerides: Triglycerides levels, ranging from 50 to 400 mg/dL.

## Cognitive and Functional Assessments

- UPDRS: Unified Parkinson's Disease Rating Scale score, ranging from 0 to 199. Higher scores indicate greater severity of the disease.
- MoCA: Montreal Cognitive Assessment score, ranging from 0 to 30. Lower scores indicate cognitive impairment.
- FunctionalAssessment: Functional assessment score, ranging from 0 to 10. Lower scores indicate greater impairment.

In [359…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


from scipy.stats import pointbiserialr
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confus
from sklearn.model_selection import cross_val_score
```

In [3]:
```
pd.set_option('display.max_rows', None)  # To display all rows
pd.set_option('display.max_columns', None)  # To display all columns
```

In [4]:
```
from IPython.display import HTML
HTML('<style>div.output_scroll {height: 400px; overflow-y: scroll;}</styl
```

Out[4]:

In [5]:
```
data=pd.read_csv('parkinsons_disease_data.csv')
```

In [6]:
```
data.head()
```

Out[6]:

| | PatientID | Age | Gender | Ethnicity | EducationLevel | BMI | Smoking | Alcohol |
|---|---|---|---|---|---|---|---|---|
| **0** | 3058 | 85 | 0 | 3 | 1 | 19.619878 | 0 | |
| **1** | 3059 | 75 | 0 | 0 | 2 | 16.247339 | 1 | |
| **2** | 3060 | 70 | 1 | 0 | 0 | 15.368239 | 0 | |
| **3** | 3061 | 52 | 0 | 0 | 0 | 15.454557 | 0 | |
| **4** | 3062 | 87 | 0 | 0 | 1 | 18.616042 | 0 | |

In [7]:
```
data.shape
```

Out[7]:
```
(2105, 35)
```

In [8]:
```
data.describe().T
```

Out[8]:

| | count | mean | std | min | 25 |
|---|---|---|---|---|---|
| **PatientID** | 2105.0 | 4110.000000 | 607.805479 | 3058.000000 | 3584.0000 |
| **Age** | 2105.0 | 69.601900 | 11.594511 | 50.000000 | 60.0000 |
| **Gender** | 2105.0 | 0.492637 | 0.500065 | 0.000000 | 0.0000 |
| **Ethnicity** | 2105.0 | 0.692637 | 1.003827 | 0.000000 | 0.0000 |
| **EducationLevel** | 2105.0 | 1.337292 | 0.895840 | 0.000000 | 1.0000 |
| **BMI** | 2105.0 | 27.209493 | 7.208099 | 15.008333 | 20.7821 |
| **Smoking** | 2105.0 | 0.296437 | 0.456795 | 0.000000 | 0.0000 |
| **AlcoholConsumption** | 2105.0 | 10.040413 | 5.687014 | 0.002228 | 5.1502 |
| **PhysicalActivity** | 2105.0 | 5.016674 | 2.890919 | 0.004157 | 2.4557 |
| **DietQuality** | 2105.0 | 4.912901 | 2.872115 | 0.000011 | 2.4785 |
| **SleepQuality** | 2105.0 | 6.996639 | 1.753065 | 4.000497 | 5.4888 |
| **FamilyHistoryParkinsons** | 2105.0 | 0.145843 | 0.353033 | 0.000000 | 0.0000 |
| **TraumaticBrainInjury** | 2105.0 | 0.106413 | 0.308439 | 0.000000 | 0.0000 |
| **Hypertension** | 2105.0 | 0.145843 | 0.353033 | 0.000000 | 0.0000 |
| **Diabetes** | 2105.0 | 0.148219 | 0.355401 | 0.000000 | 0.0000 |
| **Depression** | 2105.0 | 0.205226 | 0.403962 | 0.000000 | 0.0000 |
| **Stroke** | 2105.0 | 0.048931 | 0.215775 | 0.000000 | 0.0000 |
| **SystolicBP** | 2105.0 | 133.719715 | 26.502355 | 90.000000 | 110.0000 |
| **DiastolicBP** | 2105.0 | 90.249881 | 17.061488 | 60.000000 | 75.0000 |
| **CholesterolTotal** | 2105.0 | 226.860840 | 43.589406 | 150.062698 | 189.3851 |
| **CholesterolLDL** | 2105.0 | 126.147858 | 43.407036 | 50.022828 | 88.8419 |
| **CholesterolHDL** | 2105.0 | 59.670352 | 23.370920 | 20.027981 | 39.5386 |
| **CholesterolTriglycerides** | 2105.0 | 222.940500 | 101.895822 | 50.113604 | 132.5201 |
| **UPDRS** | 2105.0 | 101.415318 | 56.591448 | 0.028441 | 53.0481 |
| **MoCA** | 2105.0 | 15.094314 | 8.643014 | 0.021191 | 7.5171 |
| **FunctionalAssessment** | 2105.0 | 4.989694 | 2.933877 | 0.001505 | 2.4158 |
| **Tremor** | 2105.0 | 0.431829 | 0.495449 | 0.000000 | 0.0000 |
| **Rigidity** | 2105.0 | 0.252732 | 0.434682 | 0.000000 | 0.0000 |
| **Bradykinesia** | 2105.0 | 0.207601 | 0.405686 | 0.000000 | 0.0000 |
| **PosturalInstability** | 2105.0 | 0.138717 | 0.345733 | 0.000000 | 0.0000 |
| **SpeechProblems** | 2105.0 | 0.295012 | 0.456156 | 0.000000 | 0.0000 |

| | | | | | |
|---|---|---|---|---|---|
| **SleepDisorders** | 2105.0 | 0.245131 | 0.430267 | 0.000000 | 0.0000 |
| **Constipation** | 2105.0 | 0.296912 | 0.457006 | 0.000000 | 0.0000 |
| **Diagnosis** | 2105.0 | 0.619477 | 0.485631 | 0.000000 | 0.0000 |

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2105 entries, 0 to 2104
Data columns (total 35 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   PatientID                  2105 non-null   int64  
 1   Age                        2105 non-null   int64  
 2   Gender                     2105 non-null   int64  
 3   Ethnicity                  2105 non-null   int64  
 4   EducationLevel             2105 non-null   int64  
 5   BMI                        2105 non-null   float64
 6   Smoking                    2105 non-null   int64  
 7   AlcoholConsumption         2105 non-null   float64
 8   PhysicalActivity           2105 non-null   float64
 9   DietQuality                2105 non-null   float64
 10  SleepQuality               2105 non-null   float64
 11  FamilyHistoryParkinsons    2105 non-null   int64  
 12  TraumaticBrainInjury       2105 non-null   int64  
 13  Hypertension               2105 non-null   int64  
 14  Diabetes                   2105 non-null   int64  
 15  Depression                 2105 non-null   int64  
 16  Stroke                     2105 non-null   int64  
 17  SystolicBP                 2105 non-null   int64  
 18  DiastolicBP                2105 non-null   int64  
 19  CholesterolTotal           2105 non-null   float64
 20  CholesterolLDL             2105 non-null   float64
 21  CholesterolHDL             2105 non-null   float64
 22  CholesterolTriglycerides   2105 non-null   float64
 23  UPDRS                      2105 non-null   float64
 24  MoCA                       2105 non-null   float64
 25  FunctionalAssessment       2105 non-null   float64
 26  Tremor                     2105 non-null   int64  
 27  Rigidity                   2105 non-null   int64  
 28  Bradykinesia               2105 non-null   int64  
 29  PosturalInstability        2105 non-null   int64  
 30  SpeechProblems             2105 non-null   int64  
 31  SleepDisorders             2105 non-null   int64  
 32  Constipation               2105 non-null   int64  
 33  Diagnosis                  2105 non-null   int64  
 34  DoctorInCharge             2105 non-null   object 
dtypes: float64(12), int64(22), object(1)
memory usage: 575.7+ KB
```

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

Out[10]: (2105, 35)

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
PatientID                  0
Age                        0
Gender                     0
Ethnicity                  0
EducationLevel             0
BMI                        0
Smoking                    0
AlcoholConsumption         0
PhysicalActivity           0
DietQuality                0
SleepQuality               0
FamilyHistoryParkinsons    0
TraumaticBrainInjury       0
Hypertension               0
Diabetes                   0
Depression                 0
Stroke                     0
SystolicBP                 0
DiastolicBP                0
CholesterolTotal           0
CholesterolLDL             0
CholesterolHDL             0
CholesterolTriglycerides   0
UPDRS                      0
MoCA                       0
FunctionalAssessment       0
Tremor                     0
Rigidity                   0
Bradykinesia               0
PosturalInstability        0
SpeechProblems             0
SleepDisorders             0
Constipation               0
Diagnosis                  0
DoctorInCharge             0
dtype: int64
```

In [12]:
```
numerical_col=[
    'Age', 'BMI', 'SystolicBP', 'DiastolicBP', 'CholesterolTotal',
    'CholesterolLDL', 'CholesterolHDL', 'CholesterolTriglycerides', 'UPDR
    'MoCA', 'FunctionalAssessment','AlcoholConsumption','PhysicalActivity
]
```

In [ ]:
```

```

In [13]:
```
numerical_col
```

Out[13]:
```
['Age',
 'BMI',
 'SystolicBP',
 'DiastolicBP',
 'CholesterolTotal',
 'CholesterolLDL',
 'CholesterolHDL',
 'CholesterolTriglycerides',
 'UPDRS',
 'MoCA',
 'FunctionalAssessment',
 'AlcoholConsumption',
 'PhysicalActivity',
 'DietQuality',
 'SleepQuality']
```

In [14]:
```
data[numerical_col].describe().T
```

Out[14]:

| | count | mean | std | min | 25% |
|---|---|---|---|---|---|
| **Age** | 2105.0 | 69.601900 | 11.594511 | 50.000000 | 60.000000 |
| **BMI** | 2105.0 | 27.209493 | 7.208099 | 15.008333 | 20.782176 |
| **SystolicBP** | 2105.0 | 133.719715 | 26.502355 | 90.000000 | 110.000000 |
| **DiastolicBP** | 2105.0 | 90.249881 | 17.061488 | 60.000000 | 75.000000 |
| **CholesterolTotal** | 2105.0 | 226.860840 | 43.589406 | 150.062698 | 189.385178 |
| **CholesterolLDL** | 2105.0 | 126.147858 | 43.407036 | 50.022828 | 88.841960 |
| **CholesterolHDL** | 2105.0 | 59.670352 | 23.370920 | 20.027981 | 39.538643 |
| **CholesterolTriglycerides** | 2105.0 | 222.940500 | 101.895822 | 50.113604 | 132.520174 |
| **UPDRS** | 2105.0 | 101.415318 | 56.591448 | 0.028441 | 53.048148 |
| **MoCA** | 2105.0 | 15.094314 | 8.643014 | 0.021191 | 7.517160 |
| **FunctionalAssessment** | 2105.0 | 4.989694 | 2.933877 | 0.001505 | 2.415890 |
| **AlcoholConsumption** | 2105.0 | 10.040413 | 5.687014 | 0.002228 | 5.150278 |
| **PhysicalActivity** | 2105.0 | 5.016674 | 2.890919 | 0.004157 | 2.455703 |
| **DietQuality** | 2105.0 | 4.912901 | 2.872115 | 0.000011 | 2.478503 |
| **SleepQuality** | 2105.0 | 6.996639 | 1.753065 | 4.000497 | 5.488864 |

In [15]:
```
data = data.drop(columns=['DoctorInCharge'])
```

In [16]:
```
categorical_col = [
    'Gender', 'Ethnicity', 'EducationLevel', 'Smoking', 'FamilyHistoryPar
    'TraumaticBrainInjury', 'Hypertension', 'Diabetes', 'Depression', 'St
    'Tremor', 'Rigidity', 'Bradykinesia', 'PosturalInstability', 'SpeechP
```

```
    'SleepDisorders', 'Constipation'
]
```

In [17]:
```
categorical_col
```

Out[17]:
```
['Gender',
 'Ethnicity',
 'EducationLevel',
 'Smoking',
 'FamilyHistoryParkinsons',
 'TraumaticBrainInjury',
 'Hypertension',
 'Diabetes',
 'Depression',
 'Stroke',
 'Tremor',
 'Rigidity',
 'Bradykinesia',
 'PosturalInstability',
 'SpeechProblems',
 'SleepDisorders',
 'Constipation']
```

In [18]:
```
len(numerical_col)
```

Out[18]:
```
15
```

In [19]:
```
data[categorical_col].describe().T
```

Out[19]:

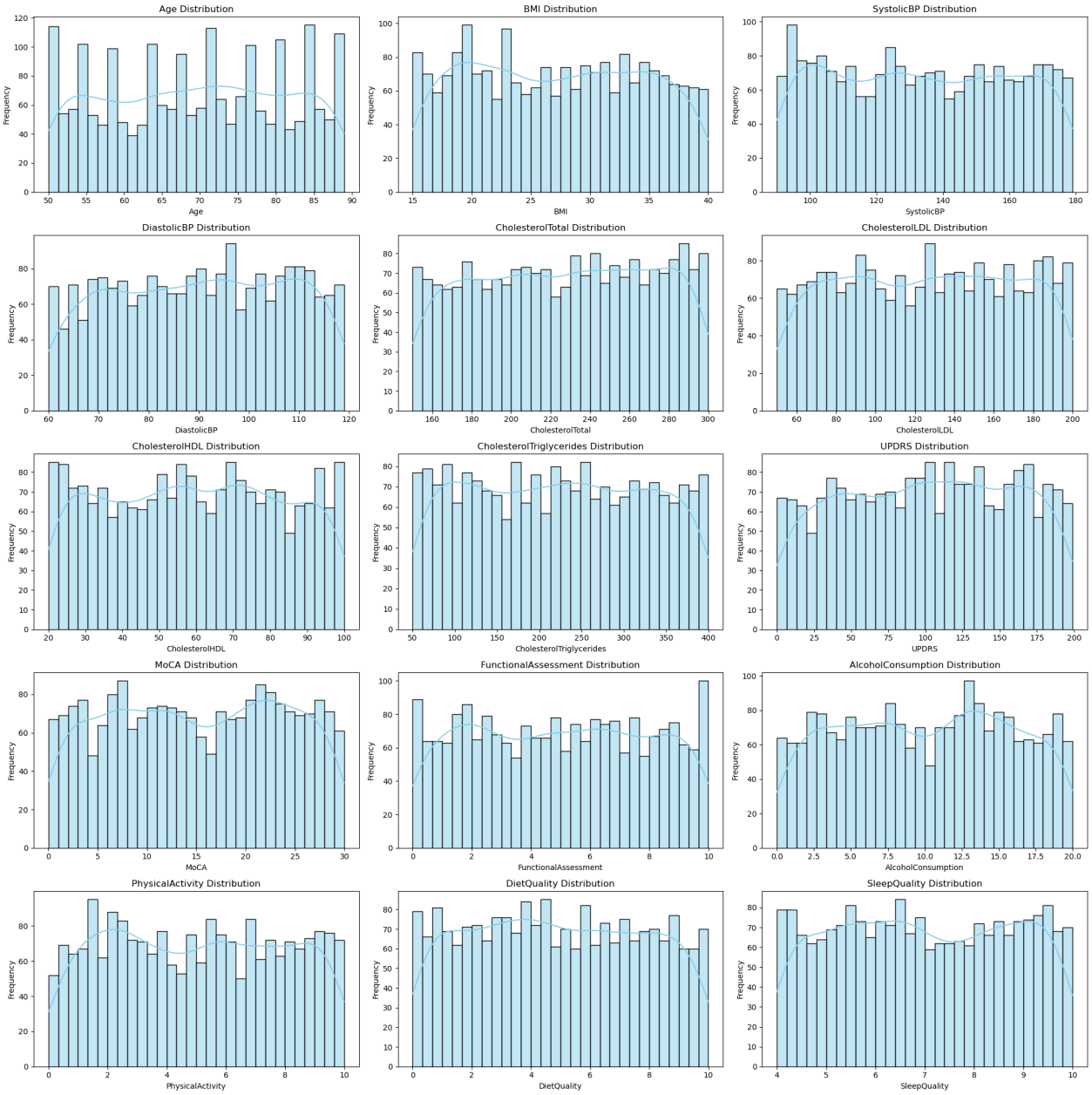
| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Gender** | 2105.0 | 0.492637 | 0.500065 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| **Ethnicity** | 2105.0 | 0.692637 | 1.003827 | 0.0 | 0.0 | 0.0 | 1.0 | 3.0 |
| **EducationLevel** | 2105.0 | 1.337292 | 0.895840 | 0.0 | 1.0 | 1.0 | 2.0 | 3.0 |
| **Smoking** | 2105.0 | 0.296437 | 0.456795 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| **FamilyHistoryParkinsons** | 2105.0 | 0.145843 | 0.353033 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **TraumaticBrainInjury** | 2105.0 | 0.106413 | 0.308439 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **Hypertension** | 2105.0 | 0.145843 | 0.353033 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **Diabetes** | 2105.0 | 0.148219 | 0.355401 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **Depression** | 2105.0 | 0.205226 | 0.403962 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **Stroke** | 2105.0 | 0.048931 | 0.215775 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **Tremor** | 2105.0 | 0.431829 | 0.495449 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| **Rigidity** | 2105.0 | 0.252732 | 0.434682 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| **Bradykinesia** | 2105.0 | 0.207601 | 0.405686 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **PosturalInstability** | 2105.0 | 0.138717 | 0.345733 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **SpeechProblems** | 2105.0 | 0.295012 | 0.456156 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| **SleepDisorders** | 2105.0 | 0.245131 | 0.430267 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **Constipation** | 2105.0 | 0.296912 | 0.457006 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |

In [20]:
```
for col in categorical_col:
    unique_values = data[col].unique()
    print(f'Unique values in {col}: {unique_values}')
```

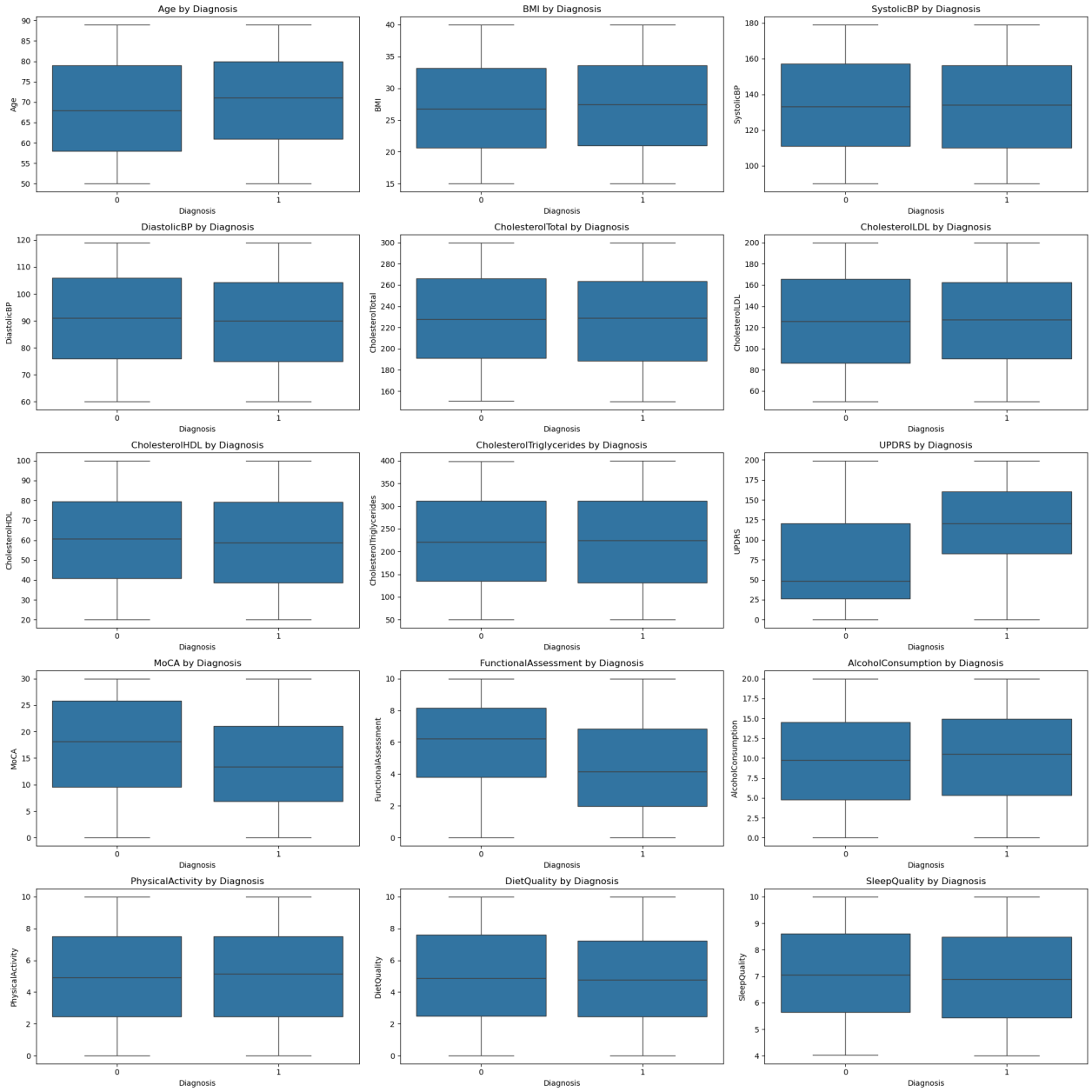
```
Unique values in Gender: [0 1]
Unique values in Ethnicity: [3 0 2 1]
Unique values in EducationLevel: [1 2 0 3]
Unique values in Smoking: [0 1]
Unique values in FamilyHistoryParkinsons: [0 1]
Unique values in TraumaticBrainInjury: [0 1]
Unique values in Hypertension: [0 1]
Unique values in Diabetes: [0 1]
Unique values in Depression: [0 1]
Unique values in Stroke: [0 1]
Unique values in Tremor: [1 0]
Unique values in Rigidity: [0 1]
Unique values in Bradykinesia: [0 1]
Unique values in PosturalInstability: [0 1]
Unique values in SpeechProblems: [0 1]
Unique values in SleepDisorders: [0 1]
Unique values in Constipation: [0 1]
```

- There is no requirement of label encoding as everything already is encoded in binary.

In [22]:
```
# Set the figure size for better visibility
plt.figure(figsize=(20, 20))

# Loop through each numerical column and create subplots
for i, col in enumerate(numerical_col):
    plt.subplot(5, 3, i + 1)  # Using 4 rows and 3 columns, fitting 12 pl
    sns.histplot(data[col], bins=30, kde=True, color='skyblue')  # Adding
    plt.title(f'{col} Distribution')
    plt.xlabel(col)
    plt.ylabel('Frequency')

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plots
plt.show()
```

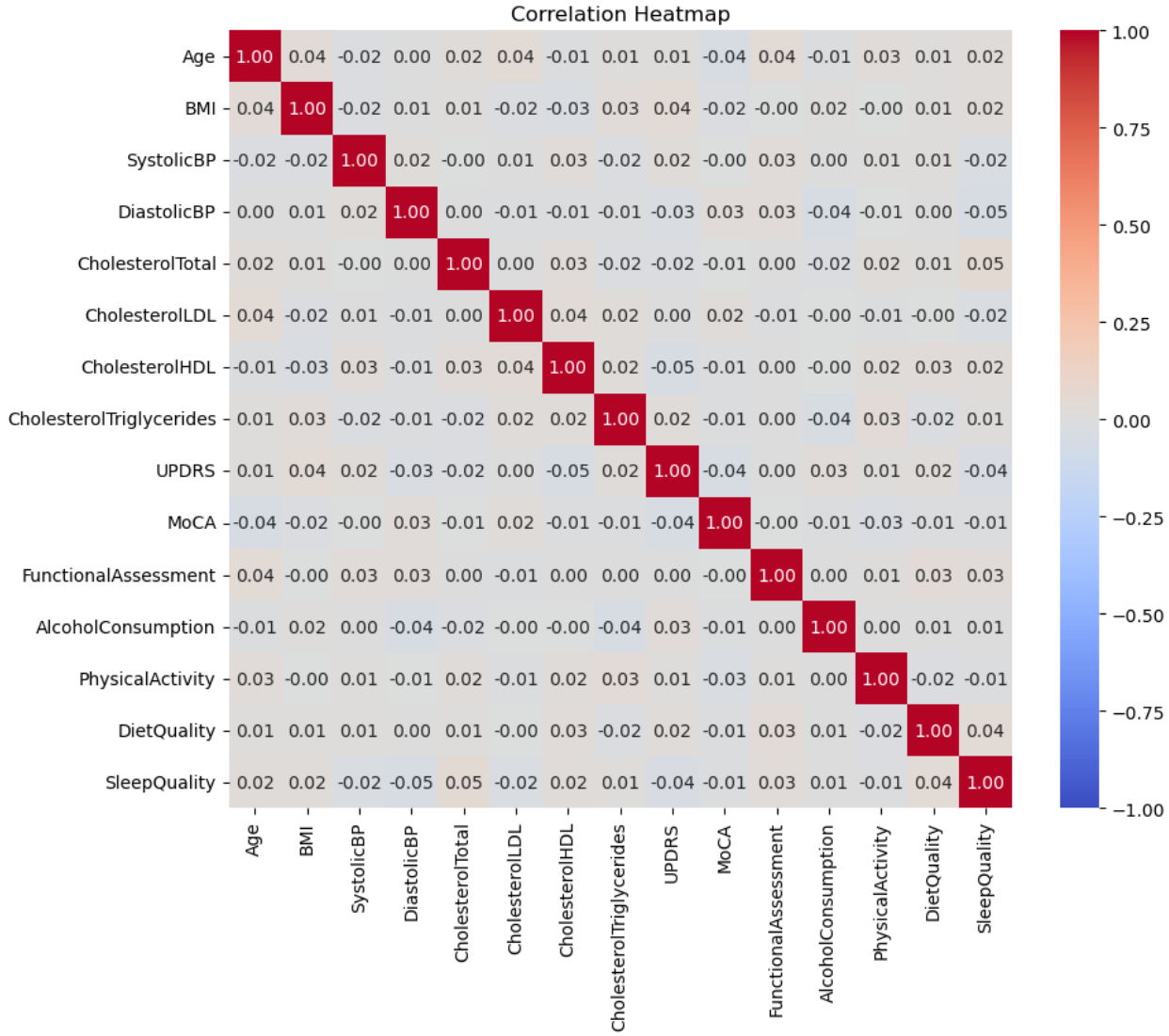



In [23]:
```
# Box plots for numerical features grouped by Diagnosis
plt.figure(figsize=(20, 20))
for i, col in enumerate(numerical_col):
    plt.subplot(5, 3, i + 1)
    sns.boxplot(x='Diagnosis', y=col, data=data)
    plt.title(f'{col} by Diagnosis')
    plt.xlabel('Diagnosis')
    plt.ylabel(col)
plt.tight_layout()
plt.show()
```




- No outlier Present in the dataset

In [25]:
```
correlation_matrix = data[numerical_col].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", v
plt.title("Correlation Heatmap")
plt.show()
```




In [26]:
```
# Correlation between the numerical variables and diagnosis
correlation_results={}

#calculating point biserial correlation for each feature
for feature in numerical_col:
    correlation,p_value=pointbiserialr(data['Diagnosis'],data[feature])
    correlation_results[feature]=correlation
```

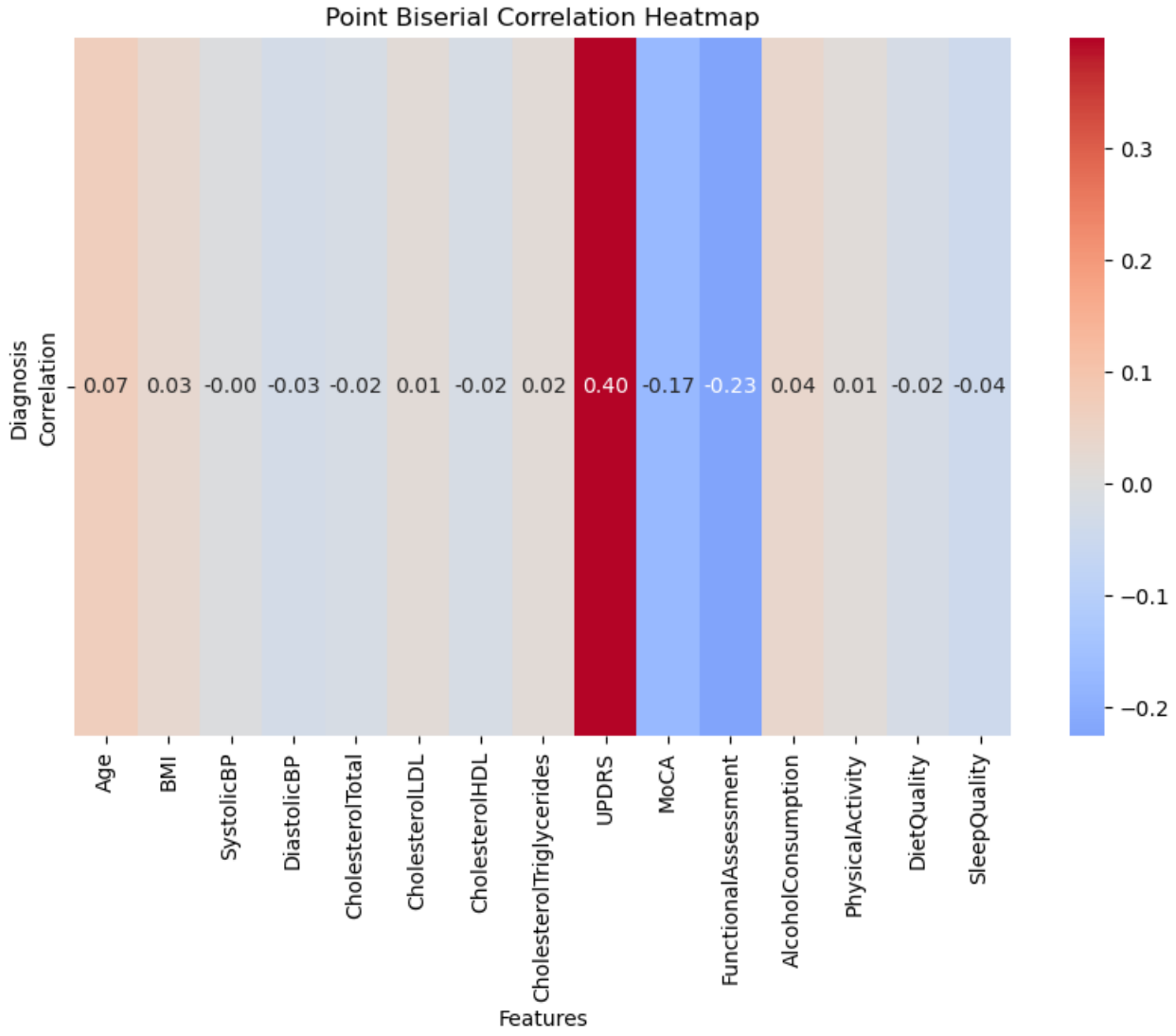
In [27]:
```
#convert the result into dataframe
corr_df=pd.DataFrame(list(correlation_results.items()),columns=['feature'

diagnosis_correlation = pd.DataFrame({'Feature': ['Diagnosis'], 'Correlat
corr_df = pd.concat([corr_df, diagnosis_correlation], ignore_index=True)
```

In [28]:
```
#correlatation matrix creation
corr_matrix=pd.DataFrame(correlation_results.values(),index=numerical_col
corr_df['Correlation'].values[:-1]
corr_matrix=corr_matrix.transpose()
corr_matrix
```

Out[28]:

| | Age | BMI | SystolicBP | DiastolicBP | CholesterolTotal | Choleste |
|---|---|---|---|---|---|---|
| **Correlation** | 0.065344 | 0.030114 | -0.004413 | -0.029074 | -0.019001 | 0. |

In [29]:
```
#heatmap
plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix,annot=True,cmap="coolwarm",fmt=".2f",center=0)
plt.title("Point Biserial Correlation Heatmap")
plt.xlabel("Features")
plt.ylabel("Diagnosis")
plt.show()
```




# General Trends

## 1. Distribution of Diagnosis across age group

In [32]:
```
#Presence of the disease among different age groups
new_df=data[['Age','Diagnosis']].copy()
```

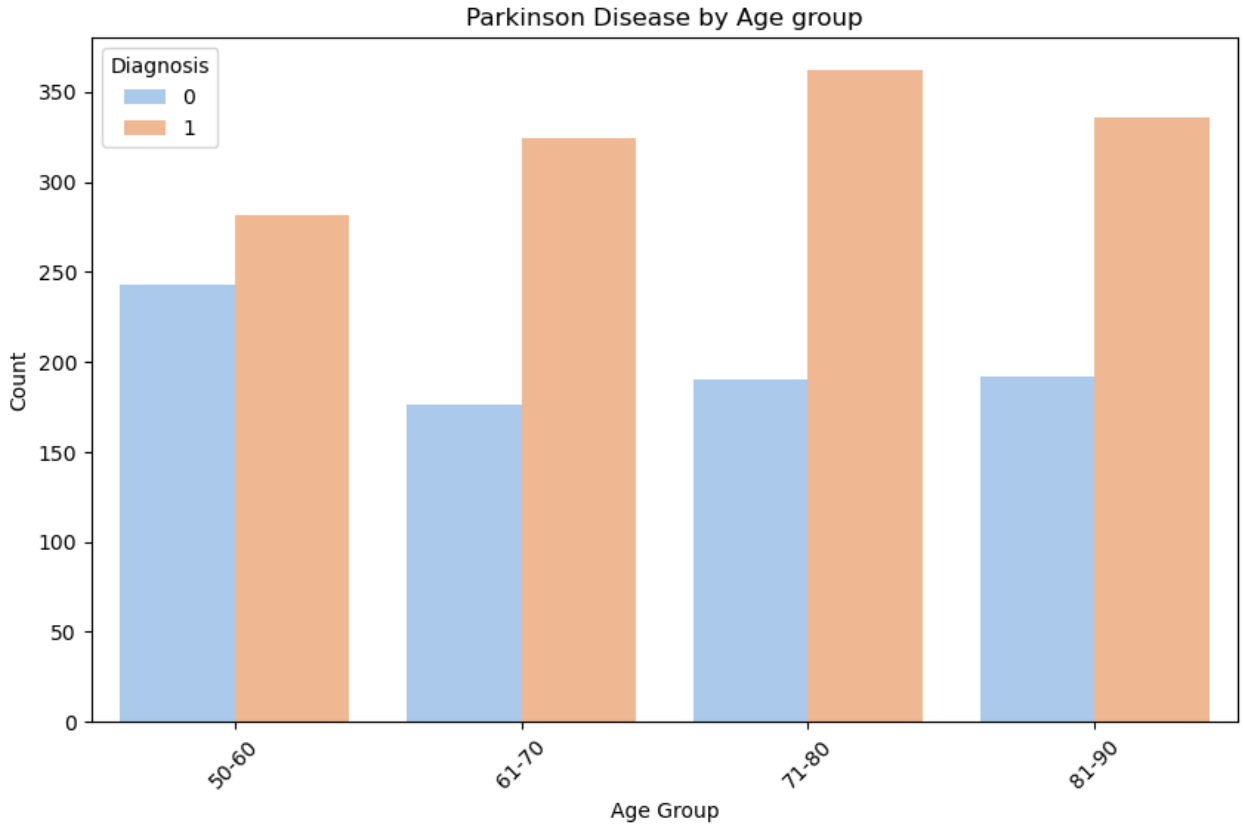
```
#create bins for creating
bins=[50,60,70,80,90]
labels=['50-60','61-70','71-80','81-90']
new_df['AgeGroup']=pd.cut(new_df['Age'],bins=bins,labels=labels,right=Fal
new_df.head()
```

Out[32]:

| | Age | Diagnosis | AgeGroup |
|---|---|---|---|
| **0** | 85 | 0 | 81-90 |
| **1** | 75 | 1 | 71-80 |
| **2** | 70 | 1 | 71-80 |
| **3** | 52 | 1 | 50-60 |
| **4** | 87 | 0 | 81-90 |

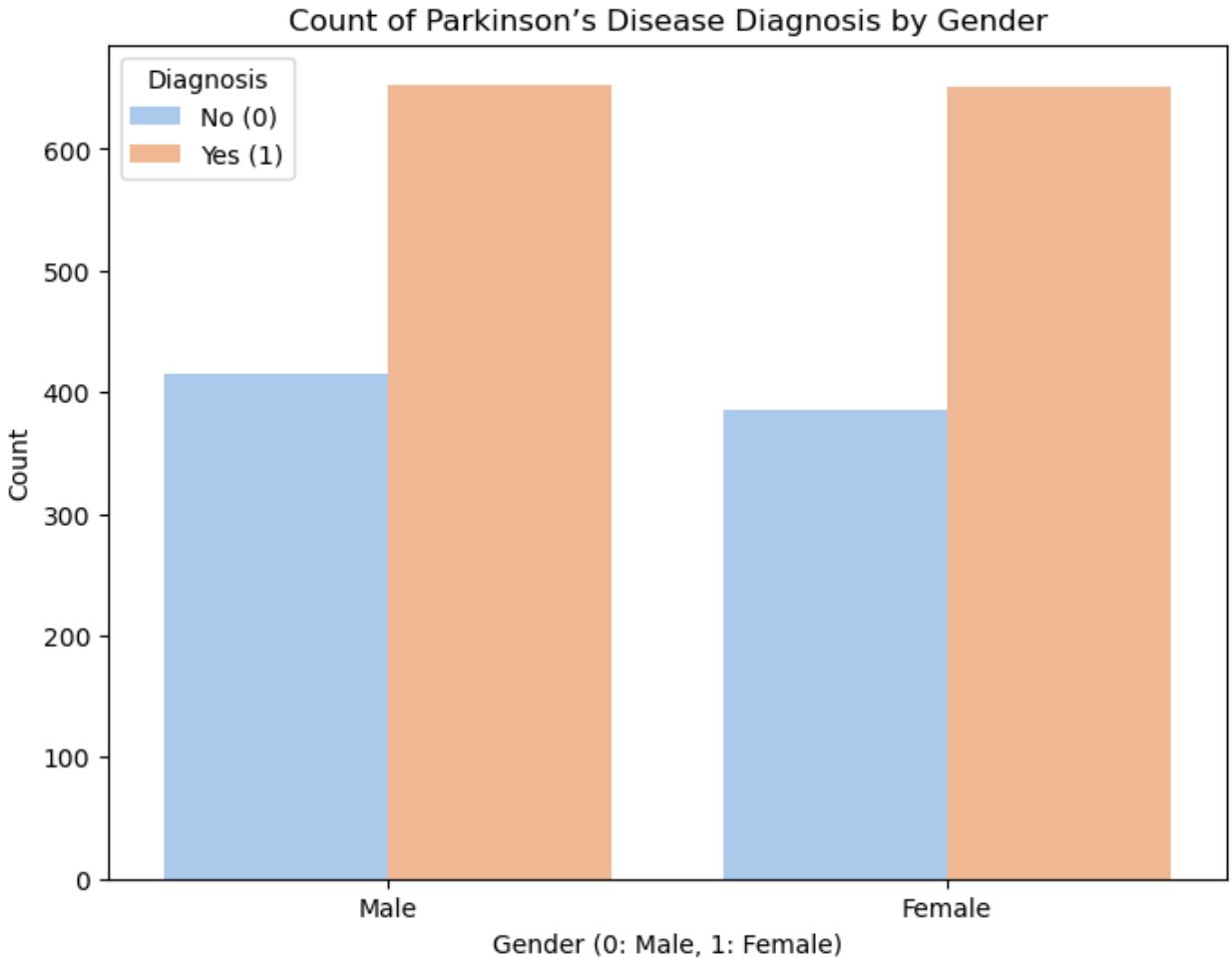
In [41]:
```
plt.figure(figsize=(10,6))
sns.countplot(x='AgeGroup',hue='Diagnosis',data=new_df,palette='pastel')
plt.title('Parkinson Disease by Age group')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.xticks(rotation=45)
```

Out[41]:
```
([0, 1, 2, 3],
 [Text(0, 0, '50-60'),
  Text(1, 0, '61-70'),
  Text(2, 0, '71-80'),
  Text(3, 0, '81-90')])
```



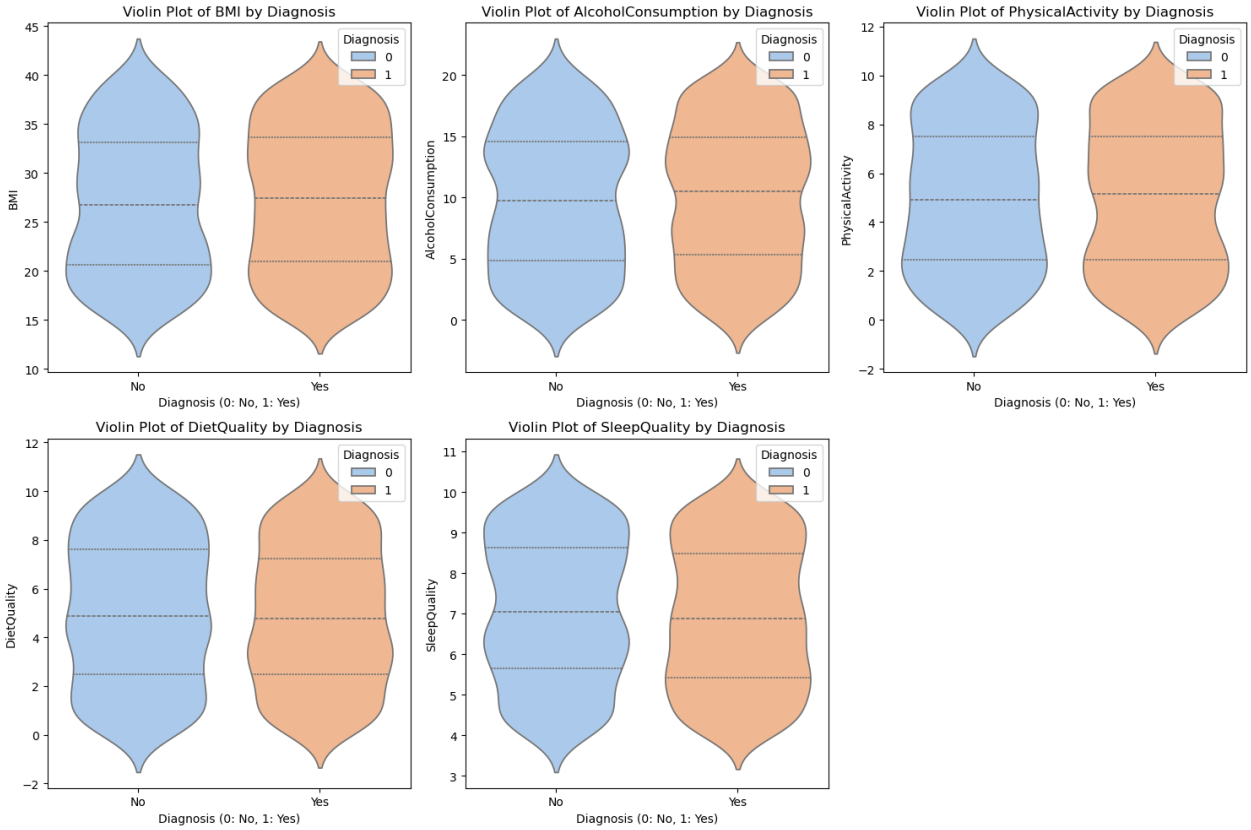

## 2. Distribution of Diagnosis by Gender

In [43]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='Gender', hue='Diagnosis', data=data, palette='pastel')
plt.title('Count of Parkinson's Disease Diagnosis by Gender')
plt.xlabel('Gender (0: Male, 1: Female)')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=['Male', 'Female'])
plt.legend(title='Diagnosis', labels=['No (0)', 'Yes (1)'])
plt.show()
```

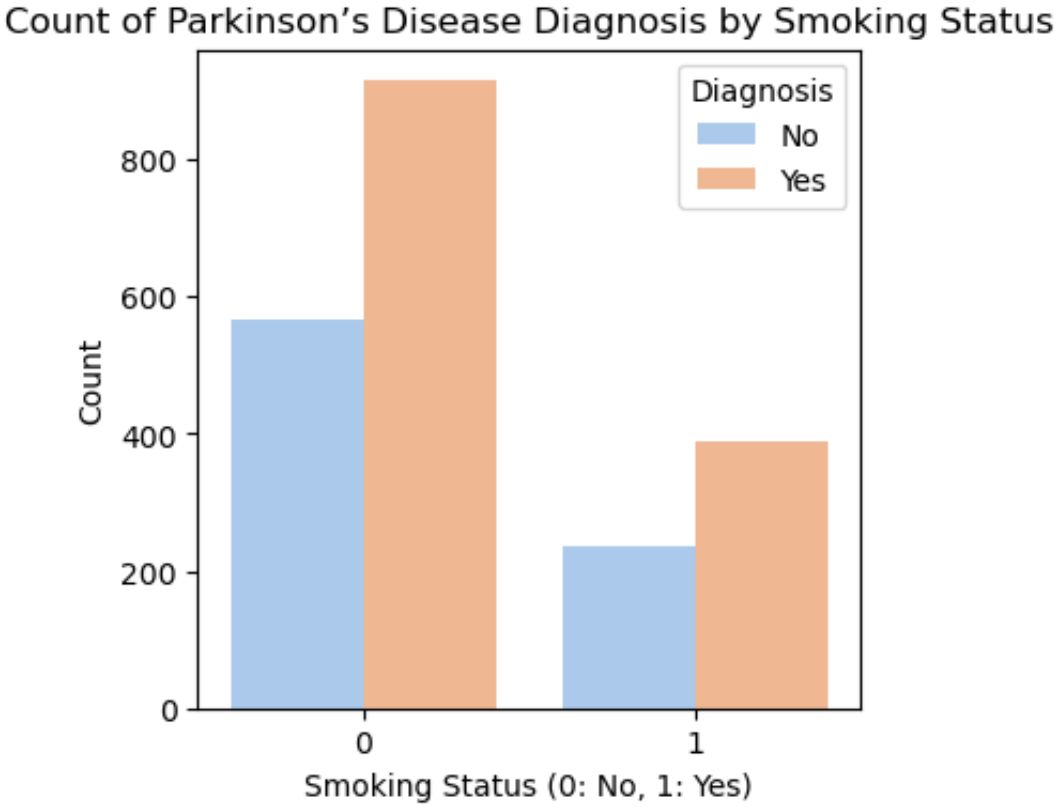



## 3. Lifestyle factor influence on Diagnosis

In [46]:
```
lifestyle_factors = ['BMI', 'AlcoholConsumption', 'PhysicalActivity', 'Di


plt.figure(figsize=(15, 10))

for i, factor in enumerate(lifestyle_factors):
    plt.subplot(2, 3, i + 1)
    sns.violinplot(x='Diagnosis', y=factor, data=data,hue='Diagnosis', pa
    plt.title(f'Violin Plot of {factor} by Diagnosis')
    plt.xlabel('Diagnosis (0: No, 1: Yes)')
    plt.ylabel(factor)
    plt.xticks(ticks=[0, 1], labels=['No', 'Yes'])

plt.tight_layout()
plt.show()
```

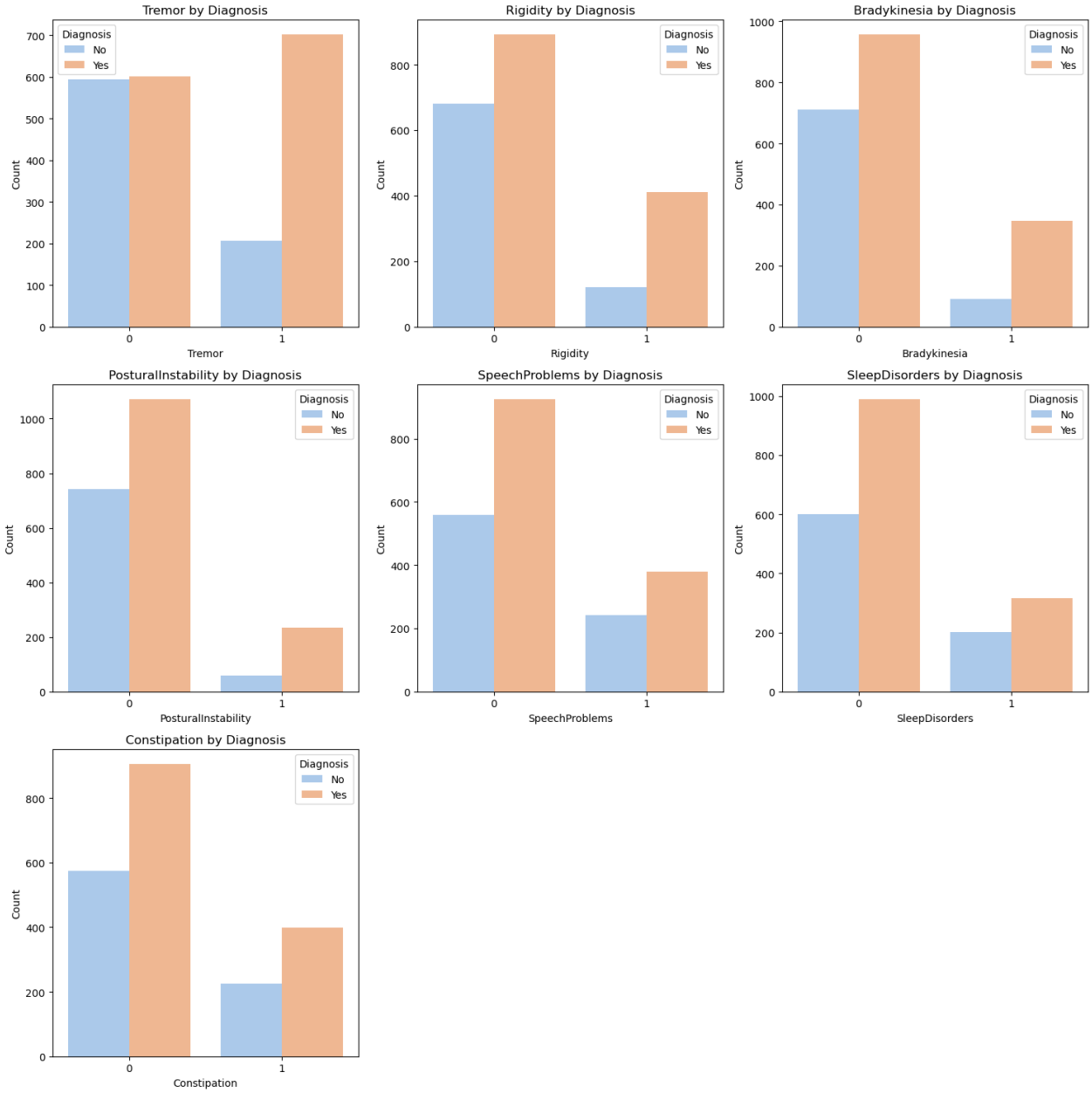



In [47]:
```
smoking_counts = data.groupby(['Smoking', 'Diagnosis']).size()

plt.figure(figsize=(4, 4))
sns.countplot(x='Smoking',hue='Diagnosis',data=data,palette='pastel')
plt.title("Count of Parkinson's Disease Diagnosis by Smoking Status")
plt.xlabel("Smoking Status (0: No, 1: Yes)")
plt.ylabel("Count")
plt.xticks(rotation=0)
plt.legend(title='Diagnosis', labels=['No', 'Yes'])
plt.show()
```

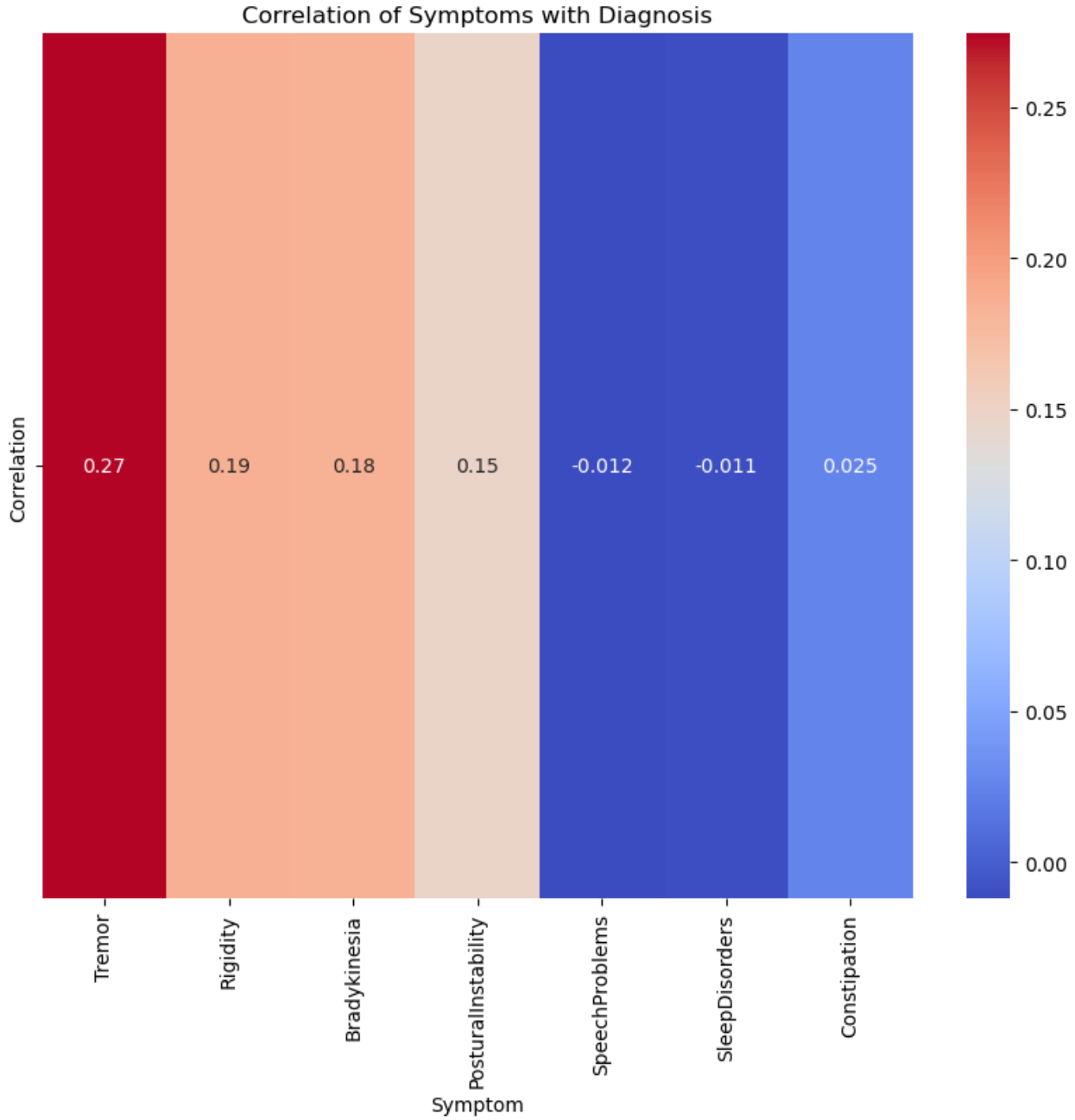



In [50]:
```
#categorical columns divided into different categories
symptoms=['Tremor','Rigidity','Bradykinesia','PosturalInstability','Speec
medical_history=['FamilyHistoryParkinsons','TraumaticBrainInjury','Hypert
```

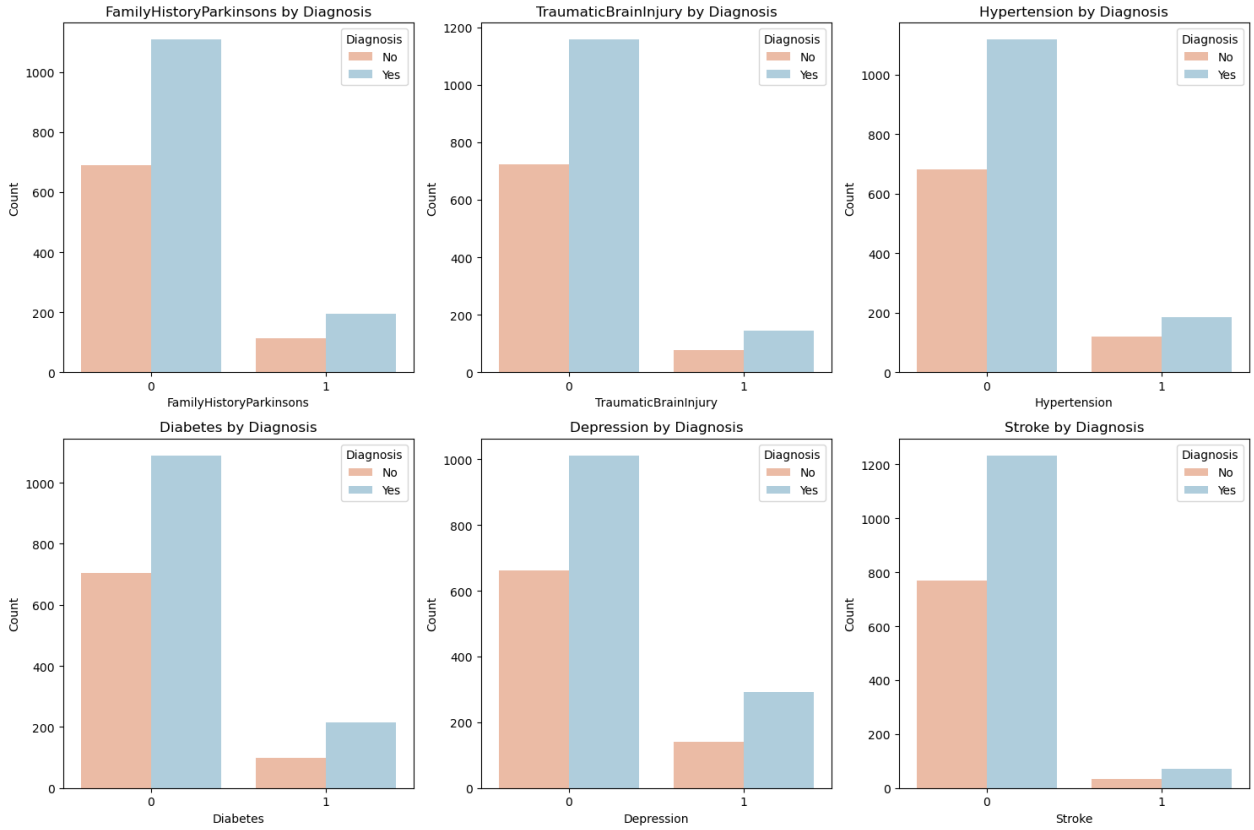
In [52]:
```
#Symptoms by Diagnosis
plt.figure(figsize=(15,15))
for i ,factor in enumerate(symptoms):
    plt.subplot(3,3,i+1)
    sns.countplot(x=factor,hue='Diagnosis',data=data,palette='pastel')
    plt.title(f'{factor} by Diagnosis')
    plt.xlabel(factor)
    plt.ylabel('Count')
    plt.legend(title='Diagnosis',labels=['No','Yes'])

plt.tight_layout()
plt.show()
```




In [67]:
```
#Heatmap for symtpom and diagnosis
correlation_values = []
for factor in symptoms:
    correlation, _ = pointbiserialr(data['Diagnosis'], data[factor])
    correlation_values.append(correlation)

corr_df = pd.DataFrame({
    'Symptom': symptoms,
    'Correlation': correlation_values
})

plt.figure(figsize=(10, 8))
sns.heatmap(corr_df.set_index('Symptom').T, annot=True, cmap="coolwarm")
plt.title("Correlation of Symptoms with Diagnosis")
plt.show()
```

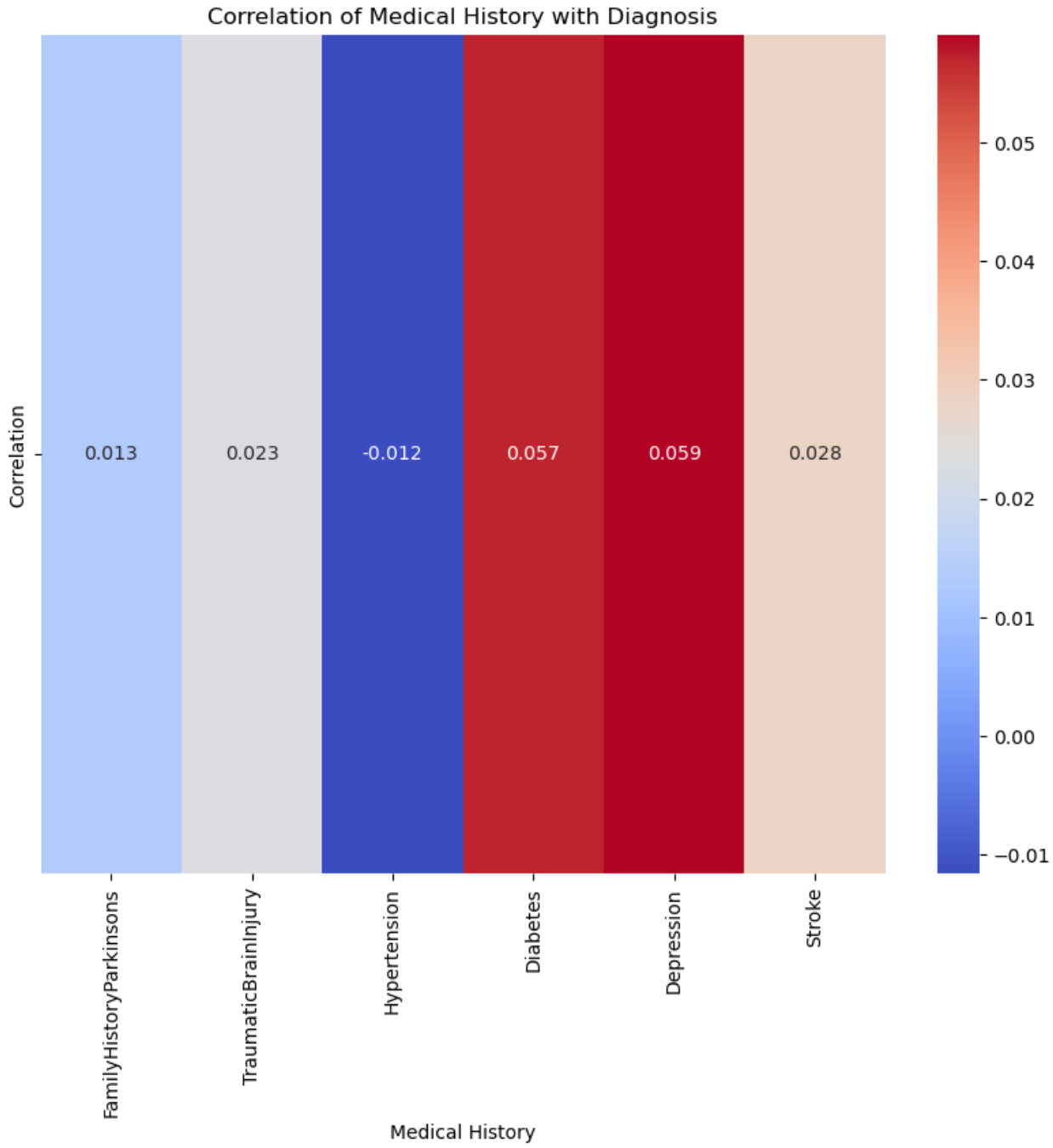



In [59]:
```
#Medical History by Diagnosis
plt.figure(figsize=(15,10))
for i ,factor in enumerate(medical_history):
    plt.subplot(2,3,i+1)
    sns.countplot(x=factor,hue='Diagnosis',data=data,palette='RdBu')
    plt.title(f'{factor} by Diagnosis')
    plt.xlabel(factor)
    plt.ylabel('Count')
    plt.legend(title='Diagnosis',labels=['No','Yes'])

plt.tight_layout()
plt.show()
```




In [304…

```
#Heatmap for Medical History and diagnosis
correlation_values = []
for factor in medical_history:
    correlation,_ = pointbiserialr(data['Diagnosis'], data[factor])
    correlation_values.append(correlation)

corr_df = pd.DataFrame({
    'Medical History': medical_history,
    'Correlation': correlation_values
})

plt.figure(figsize=(10, 8))
sns.heatmap(corr_df.set_index('Medical History').T, annot=True, cmap="coo
plt.title("Correlation of Medical History with Diagnosis")
plt.show()
```




In [306…
```
scaler = preprocessing.StandardScaler().fit(X_train)
scaler
```

Out[306…
```
StandardScaler
StandardScaler()
```

In [312…
```
print('Mean:',scaler.mean_)
print('Scale:',scaler.scale_)
```

```
Mean: [4.10611580e+03 6.95332542e+01 4.88123515e-01 6.75771971e-01
 1.33135392e+00 2.72486652e+01 2.85035629e-01 1.00913736e+01
 4.99015241e+00 4.94074728e+00 7.02208275e+00 1.52019002e-01
 1.08076010e-01 1.39548694e-01 1.51425178e-01 2.08432304e-01
 4.80997625e-02 1.33834917e+02 9.02915677e+01 2.27434782e+02
 1.26094384e+02 5.98906320e+01 2.22018878e+02 1.02428837e+02
 1.51082759e+01 5.04819147e+00 4.25771971e-01 2.50000000e-01
 2.07244656e-01 1.33016627e-01 2.93942993e-01 2.47030879e-01
 2.97505938e-01]
Scale: [6.02457280e+02 1.15923212e+01 4.99858929e-01 9.90144875e-01
 9.09537911e-01 7.19135492e+00 4.51431412e-01 5.67560830e+00
 2.87539206e+00 2.86396057e+00 1.75448131e+00 3.59039309e-01
 3.10476385e-01 3.46518190e-01 3.58462820e-01 4.06187492e-01
 2.13977044e-01 2.63441064e+01 1.70501886e+01 4.34389804e+01
 4.31778985e+01 2.34228808e+01 1.01200093e+02 5.62153803e+01
 8.60612841e+00 2.93846144e+00 4.94459503e-01 4.33012702e-01
 4.05332343e-01 3.39592703e-01 4.55566142e-01 4.31284852e-01
 4.57160973e-01]
```

In [314…
```
X_scaled = scaler.transform(X_train)
X_scaled
```

Out[314…
```
array([[-0.28900936,  0.12652736,  1.02404189, ..., -0.64522572,
        -0.57277894, -0.65076845],
       [-0.66081996, -0.90864064,  1.02404189, ...,  1.54984522,
        -0.57277894, -0.65076845],
       [ 0.90941586, -1.33996064,  1.02404189, ..., -0.64522572,
        -0.57277894, -0.65076845],
       ...,
       [ 0.13591703, -0.21852864, -0.97652255, ...,  1.54984522,
        -0.57277894, -0.65076845],
       [ 0.4081355 ,  0.29905536, -0.97652255, ...,  1.54984522,
        -0.57277894, -0.65076845],
       [-0.31224753, -0.73611264, -0.97652255, ...,  1.54984522,
        -0.57277894,  1.53664487]])
```

In [320…
```
print('Mean\n',X_scaled.mean(axis=0))

print('Standard Deviation\n',X_scaled.std(axis=0))
```

```
Mean
 [-5.61176864e-16 -4.24047179e-16 -3.37549993e-17 -1.68774997e-17
 -1.18142498e-16  1.94091246e-16  6.32906237e-18 -9.70456231e-17
 -2.89027182e-16 -4.11389054e-16  3.79743742e-17  1.26581247e-17
 -3.69195305e-17 -2.10968746e-18 -2.95356244e-17 -5.06324990e-17
 -2.32065620e-17 -5.23202490e-16 -3.73414680e-16 -4.14553585e-16
 -6.75099986e-17 -1.29745779e-16 -1.09703748e-16  3.29111243e-16
  1.56116872e-16  6.96196861e-17  9.91553105e-17 -2.95356244e-17
  4.43034366e-17  2.53162495e-17 -4.21937492e-18  8.43874983e-17
  1.13923123e-16]
Standard Deviation
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1.]
```

# Model Building

In [329…
```
X = data.drop('Diagnosis', axis=1)
y = data['Diagnosis']
```

In [331…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
```

In [396…
```
from sklearn.linear_model import LogisticRegression

lr_model=LogisticRegression(max_iter=5000)
lr_model.fit(X_train,y_train)
y_pred_lr=lr_model.predict(X_test)
accuracy_lr=accuracy_score(y_test,y_pred_lr)

print("Linear Regression Accuracy:",accuracy_lr)

cv_scores=cross_val_score(lr_model,X,y,cv=5,scoring='accuracy')
mean_cv_accuracy_lr=cv_scores.mean()
std_cv_accuracy_=cv_scores.std()

print(f"Cross Validation Accuracy:{mean_cv_accuracy_svm:4f}(+/- {std_cv_a
print(classification_report(y_test,y_pred_lr))
```

```
Linear Regression Accuracy: 0.7957244655581948
Cross Validation Accuracy:0.801425(+/- 0.0347)

              precision    recall  f1-score   support

           0       0.72      0.69      0.71       150
           1       0.83      0.85      0.84       271

    accuracy                           0.80       421
   macro avg       0.78      0.77      0.78       421
weighted avg       0.79      0.80      0.79       421
```

In [378…
```
from sklearn.svm import SVC

svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
accuracy_svm=accuracy_score(y_test, y_pred_svm)

print("SVM Accuracy:", accuracy_svm)

cv_scores = cross_val_score(svm_model, X, y, cv=5, scoring='accuracy')
mean_cv_accuracy_svm = cv_scores.mean()
std_cv_accuracy = cv_scores.std()

print(f"Cross-Validation Accuracy: {mean_cv_accuracy_svm:.4f} (+/- {std_c
print(classification_report(y_test, y_pred_svm))
```

```
SVM Accuracy: 0.7862232779097387
Cross-Validation Accuracy: 0.8014 (+/- 0.0347)

              precision    recall  f1-score   support

           0       0.69      0.74      0.71       150
           1       0.85      0.81      0.83       271

    accuracy                           0.79       421
   macro avg       0.77      0.78      0.77       421
weighted avg       0.79      0.79      0.79       421
```

In [369…

```
from sklearn.tree import DecisionTreeClassifier

dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
accuracy_dt=accuracy_score(y_test, y_pred_dt)

print("Decision Tree Accuracy:", accuracy_dt)

cv_scores = cross_val_score(dt_model, X, y, cv=5, scoring='accuracy')
mean_cv_accuracy_dt = cv_scores.mean()
std_cv_accuracy = cv_scores.std()

print(f"Cross-Validation Accuracy: {mean_cv_accuracy_dt:.4f} (+/- {std_cv
print(classification_report(y_test, y_pred_dt))
```

```
Decision Tree Accuracy: 0.8978622327790974
Cross-Validation Accuracy: 0.8979 (+/- 0.0313)

              precision    recall  f1-score   support

           0       0.85      0.87      0.86       150
           1       0.93      0.91      0.92       271

    accuracy                           0.90       421
   macro avg       0.89      0.89      0.89       421
weighted avg       0.90      0.90      0.90       421
```

In [373…

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
accuracy_rf=accuracy_score(y_test, y_pred_rf)


print("Random Forest Accuracy:", accuracy_rf)

cv_scores = cross_val_score(rf_model, X, y, cv=5, scoring='accuracy')
mean_cv_accuracy_rf = cv_scores.mean()
```

```
std_cv_accuracy = cv_scores.std()

print(f"Cross-Validation Accuracy: {mean_cv_accuracy_rf:.4f} (+/- {std_cv
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.9097387173396675
Cross-Validation Accuracy: 0.9031 (+/- 0.0473)

              precision    recall  f1-score   support

           0       0.85      0.91      0.88       150
           1       0.95      0.91      0.93       271

    accuracy                           0.91       421
   macro avg       0.90      0.91      0.90       421
weighted avg       0.91      0.91      0.91       421
```

In [434…

```
accuracy_scores=[accuracy_lr*100,accuracy_svm*100,accuracy_dt*100,accurac
cc_scores=[mean_cv_accuracy_lr*100,mean_cv_accuracy_svm*100,mean_cv_accur

models=["Linear Regression","SVM","Decision Tree","Random Forest"]

fig=go.Figure(data=[go.Bar(
    name='Accuracy',
    x=models,
    y=accuracy_scores,
    marker_color='lightcoral'
),
    go.Bar(
        name='Cross validation score',
        x=models,
        y=cc_scores,
        marker_color='lightblue')
        ])
fig.show()
```

## Random Forest can be chosen as the best model based on Accuracy and Cross validation scores

In [ ]: