Business Problem Understanding

The COVID-19 pandemic has revealed significant disparities in health outcomes based on patient demographics, preexisting conditions, and vaccination coverage. Public health authorities aim to reduce hospitalization rates, manage reinfection risk, and improve recovery outcomes by identifying key risk factors and evaluating the effectiveness of vaccination programs.

This project seeks to analyze detailed patient-level COVID-19 data to uncover patterns in disease severity, recovery duration, reinfection trends, and long COVID occurrences. The analysis will also help assess how vaccination type, dosage, and patient profiles influence recovery and hospitalization outcomes.

Insights from this analysis will support healthcare decision-makers in optimizing vaccination strategies, allocating medical resources efficiently, and designing targeted interventions for high-risk populations.

Data Understanding

In [13]:
```
import pandas as pd
from scipy.stats import skew
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.simplefilter("ignore")
```

In [14]:
```
df = pd.read_csv("covid_related_disease_data.csv")
df
```

Out[14]:

	Patient_ID	Age	Gender	Region	Preexisting_Condition	Date_of_Infection
0	1	69	Male	Hovedstaden	Obesity	2022-06-21
1	2	38	Male	Sjælland	Asthma	2024-02-02
2	3	41	Female	Syddanmark	Hypertension	2023-05-28
3	4	81	Female	Hovedstaden	Asthma	2023-08-13
4	5	50	Female	Syddanmark	Cardiovascular	2023-03-10
...	...	...	...	...	...	...
2995	2996	43	Male	Nordjylland	Hypertension	2022-10-19
2996	2997	36	Female	Syddanmark	Obesity	2022-12-16
2997	2998	75	Female	Sjælland	Cardiovascular	2023-09-30
2998	2999	45	Female	Hovedstaden	Asthma	2023-06-06
2999	3000	83	Female	Midtjylland	Obesity	2023-09-07

3000 rows × 26 columns

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

Out[15]: (3000, 26)

In [16]:
```
df.size
```

Out[16]: 78000

In [17]:
```
df.head(5)
```

Out[17]:

	Patient_ID	Age	Gender	Region	Preexisting_Condition	Date_of_Infection	COV
0	1	69	Male	Hovedstaden	Obesity	2022-06-21	
1	2	38	Male	Sjælland	Asthma	2024-02-02	
2	3	41	Female	Syddanmark	Hypertension	2023-05-28	
3	4	81	Female	Hovedstaden	Asthma	2023-08-13	
4	5	50	Female	Syddanmark	Cardiovascular	2023-03-10	

5 rows × 26 columns

In [18]:
```
df.tail(3)
```

Out[18]:

	Patient_ID	Age	Gender	Region	Preexisting_Condition	Date_of_Infection
2997	2998	75	Female	Sjælland	Cardiovascular	2023-09-30
2998	2999	45	Female	Hovedstaden	Asthma	2023-06-06
2999	3000	83	Female	Midtjylland	Obesity	2023-09-07

3 rows × 26 columns

In [19]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 26 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Patient_ID              3000 non-null   int64  
 1   Age                     3000 non-null   int64  
 2   Gender                  3000 non-null   object 
 3   Region                  3000 non-null   object 
 4   Preexisting_Condition   2531 non-null   object 
 5   Date_of_Infection       3000 non-null   object 
 6   COVID_Strain            3000 non-null   object 
 7   Symptoms                3000 non-null   object 
 8   Severity                3000 non-null   object 
 9   Hospitalized            3000 non-null   object 
 10  Hospital_Admission_Date 876 non-null    object 
 11  Hospital_Discharge_Date 876 non-null    object 
 12  ICU_Admission           3000 non-null   object 
 13  Ventilator_Support      3000 non-null   object 
 14  Recovered               3000 non-null   object 
 15  Date_of_Recovery        1508 non-null   object 
 16  Reinfection             3000 non-null   object 
 17  Date_of_Reinfection     285 non-null    object 
 18  Vaccination_Status      3000 non-null   object 
 19  Vaccine_Type            1191 non-null   object 
 20  Doses_Received          3000 non-null   int64  
 21  Date_of_Last_Dose       1472 non-null   object 
 22  Long_COVID_Symptoms     220 non-null    object 
 23  Occupation              3000 non-null   object 
 24  Smoking_Status          3000 non-null   object 
 25  BMI                     3000 non-null   float64
dtypes: float64(1), int64(3), object(22)
memory usage: 609.5+ KB
```

In [20]:

```
df.columns.tolist()
```

Out[20]:
```
['Patient_ID',
 'Age',
 'Gender',
 'Region',
 'Preexisting_Condition',
 'Date_of_Infection',
 'COVID_Strain',
 'Symptoms',
 'Severity',
 'Hospitalized',
 'Hospital_Admission_Date',
 'Hospital_Discharge_Date',
 'ICU_Admission',
 'Ventilator_Support',
 'Recovered',
 'Date_of_Recovery',
 'Reinfection',
 'Date_of_Reinfection',
 'Vaccination_Status',
 'Vaccine_Type',
 'Doses_Received',
 'Date_of_Last_Dose',
 'Long_COVID_Symptoms',
 'Occupation',
 'Smoking_Status',
 'BMI']
```

In [21]:
```
print(df)
```

```
      Patient_ID  Age  Gender       Region Preexisting_Condition  \
0              1   69    Male  Hovedstaden               Obesity   
1              2   38    Male     Sjælland                Asthma   
2              3   41  Female   Syddanmark          Hypertension   
3              4   81  Female  Hovedstaden                Asthma   
4              5   50  Female   Syddanmark        Cardiovascular   
...          ...  ...     ...          ...                   ...   
2995        2996   43    Male  Nordjylland          Hypertension   
2996        2997   36  Female   Syddanmark               Obesity   
2997        2998   75  Female     Sjælland        Cardiovascular   
2998        2999   45  Female  Hovedstaden                Asthma   
2999        3000   83  Female  Midtjylland               Obesity   

     Date_of_Infection COVID_Strain  Symptoms  Severity Hospitalized  ...  \
0           2022-06-21        Delta      Mild  Moderate          Yes  ...   
1           2024-02-02      XBB.1.5      Mild  Moderate           No  ...   
2           2023-05-28         Beta      Mild      High          Yes  ...   
3           2023-08-13        Delta    Severe      High           No  ...   
4           2023-03-10        Delta      Mild      High           No  ...   
...                ...          ...       ...       ...          ...  ...   
2995        2022-10-19      XBB.1.5    Severe  Critical           No  ...   
2996        2022-12-16      Omicron  Moderate       Low           No  ...   
2997        2023-09-30         Beta    Severe  Moderate           No  ...   
2998        2023-06-06        Delta    Severe  Moderate           No  ...   
2999        2023-09-07      XBB.1.5  Moderate       Low           No  ...   

     Reinfection Date_of_Reinfection Vaccination_Status Vaccine_Type  \
0             No                 NaN                Yes          NaN   
1             No                 NaN                 No          NaN   
2             No                 NaN                Yes      Janssen   
3            Yes          2024-08-24                Yes  AstraZeneca   
4             No                 NaN                Yes          NaN   
...          ...                 ...                ...          ...   
2995          No                 NaN                Yes          NaN   
2996          No                 NaN                Yes       Pfizer   
2997          No                 NaN                Yes      Moderna   
2998          No                 NaN                Yes  AstraZeneca   
2999          No                 NaN                 No          NaN   

     Doses_Received Date_of_Last_Dose Long_COVID_Symptoms     Occupation  \
0                 1        2022-09-22                 NaN     Healthcare   
1                 0               NaN                 NaN     Healthcare   
2                 3        2024-05-14                 NaN     Unemployed   
3                 1        2024-10-31                 NaN  Office Worker   
4                 2        2023-07-05                 NaN        Student   
...             ...               ...                 ...            ...   
2995              1        2024-09-20                 NaN         Driver   
2996              2        2023-10-05                 NaN     Healthcare   
2997              3        2023-05-13                 NaN        Teacher   
2998              1        2024-05-13                 NaN        Student   
2999              0               NaN                 NaN        Teacher   

     Smoking_Status   BMI  
0             Never  27.7  
1             Never  21.9  
2             Never  22.7  
3             Never  27.7  
4             Never  11.9  
...             ...   ...  
2995          Never  22.0  
```

```
2996            Never   27.8
2997           Former   20.9
2998            Never   19.3
2999           Former   33.0

[3000 rows x 26 columns]
```

In [22]:
```
df.dtypes
```

Out[22]:
```
Patient_ID                   int64
Age                          int64
Gender                      object
Region                      object
Preexisting_Condition       object
Date_of_Infection           object
COVID_Strain                object
Symptoms                    object
Severity                    object
Hospitalized                object
Hospital_Admission_Date     object
Hospital_Discharge_Date     object
ICU_Admission               object
Ventilator_Support          object
Recovered                   object
Date_of_Recovery            object
Reinfection                 object
Date_of_Reinfection         object
Vaccination_Status          object
Vaccine_Type                object
Doses_Received               int64
Date_of_Last_Dose           object
Long_COVID_Symptoms         object
Occupation                  object
Smoking_Status              object
BMI                        float64
dtype: object
```

Data Cleaning / Data Preprocessing

In [25]:
```
df.columns
```

Out[25]:
```
Index(['Patient_ID', 'Age', 'Gender', 'Region', 'Preexisting_Condition',
       'Date_of_Infection', 'COVID_Strain', 'Symptoms', 'Severity',
       'Hospitalized', 'Hospital_Admission_Date', 'Hospital_Discharge_Date',
       'ICU_Admission', 'Ventilator_Support', 'Recovered', 'Date_of_Recovery',
       'Reinfection', 'Date_of_Reinfection', 'Vaccination_Status',
       'Vaccine_Type', 'Doses_Received', 'Date_of_Last_Dose',
       'Long_COVID_Symptoms', 'Occupation', 'Smoking_Status', 'BMI'],
      dtype='object')
```

In [26]:
```
# to check the duplicated record
df.duplicated().sum()
```

Out[26]:
```
0
```

In [27]:
```
df.shape[0]
```

Out[27]:
```
3000
```

In [28]:
```
# to check the missing values
df.isnull().sum()
```

Out[28]:
```
Patient_ID                    0
Age                           0
Gender                        0
Region                        0
Preexisting_Condition       469
Date_of_Infection             0
COVID_Strain                  0
Symptoms                      0
Severity                      0
Hospitalized                  0
Hospital_Admission_Date    2124
Hospital_Discharge_Date    2124
ICU_Admission                 0
Ventilator_Support            0
Recovered                     0
Date_of_Recovery           1492
Reinfection                   0
Date_of_Reinfection        2715
Vaccination_Status            0
Vaccine_Type               1809
Doses_Received                0
Date_of_Last_Dose          1528
Long_COVID_Symptoms        2780
Occupation                    0
Smoking_Status                0
BMI                           0
dtype: int64
```

In [29]:
```
(df.isnull().sum()/len(df)) * 100
```

Out[29]:
```
Patient_ID                  0.000000
Age                         0.000000
Gender                      0.000000
Region                      0.000000
Preexisting_Condition      15.633333
Date_of_Infection           0.000000
COVID_Strain                0.000000
Symptoms                    0.000000
Severity                    0.000000
Hospitalized                0.000000
Hospital_Admission_Date    70.800000
Hospital_Discharge_Date    70.800000
ICU_Admission               0.000000
Ventilator_Support          0.000000
Recovered                   0.000000
Date_of_Recovery           49.733333
Reinfection                 0.000000
Date_of_Reinfection        90.500000
Vaccination_Status          0.000000
Vaccine_Type               60.300000
Doses_Received              0.000000
Date_of_Last_Dose          50.933333
Long_COVID_Symptoms        92.666667
Occupation                  0.000000
Smoking_Status              0.000000
BMI                         0.000000
dtype: float64
```

In [30]:
```
# Check if there are any missing values in the entire DataFrame
print(df.isnull().values.any())
```

```
True
```

In [31]:
```
# filling missing values insted of nan -> None , means the patient had no preexi
df['Preexisting_Condition'].fillna('None', inplace=True)
```

In [34]:
```
# Hospital_Admission_Date and Hospital_Discharge_Date HAVING MORETHAN 30% OF NUL
df.drop(['Hospital_Admission_Date', 'Hospital_Discharge_Date'], axis=1, inplace=
```

In [42]:
```
# Replace NaN values in 'Recovery_Days' with the string 'None'
df['Recovery_Days'] = df['Recovery_Days'].fillna(0)
```

In [44]:
```
# Step 1: Convert to numeric (non-numeric values become NaN)
df['Recovery_Days'] = pd.to_numeric(df['Recovery_Days'], errors='coerce')
# Step 2: Replace NaN with 0
df['Recovery_Days'] = df['Recovery_Days'].fillna(0)
# Step 3: Convert the column to integer type
#['Recovery_Days'] = df['Recovery_Days'].astype(int)
```

In [ ]:
```
# IN Date_of_Recovery having more than 30%  of Null Values so i am drop the colu
df.drop(['Date_of_Recovery'] , axis=1, inplace=True)
```

In [ ]:
```
# drop the Date_of_Reinfection columnin the datasets having more than 30% of nul
df.drop(['Date_of_Reinfection'], axis=1, inplace=True)
```

In [ ]:
```
# This line removes the Date_of_Last_Dose column from the DataFrame df
df.drop(['Date_of_Last_Dose'], axis=1, inplace=True)
```

In [ ]:
```
# This line replaces all missing values in the Long_COVID_Symptoms column with t
df['Long_COVID_Symptoms'] = df['Long_COVID_Symptoms'].fillna("None")
```

In [ ]:
```
df['Vaccine_Type'] = df['Vaccine_Type'].fillna("None")
```

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

In [ ]:
```
# Calculate Q1, Q3, and IQR for the BMI column
Q1 = df['BMI'].quantile(0.25)
Q3 = df['BMI'].quantile(0.75)
IQR = Q3 - Q1

# Define lower and upper bounds for detecting outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

In [ ]:
```
# Remove outliers from the DataFrame based on BMI
df_no_outliers = df[(df['BMI'] >= lower_bound) & (df['BMI'] <= upper_bound)]
```

In [ ]:
```
# Step 1: Calculate Q1, Q3 and IQR
Q1 = df['Recovery_Days'].quantile(0.25)
Q3 = df['Recovery_Days'].quantile(0.75)
IQR = Q3 - Q1

# Step 2: Define outlier bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Step 3: Calculate median (excluding outliers)
median_value = df[(df['Recovery_Days'] >= lower_bound) & (df['Recovery_Days'] <=

# Step 4: Replace outliers with the median
df['Recovery_Days'] = df['Recovery_Days'].apply(lambda x: median_value if x < lo

# Final output
print(df)
```

In [ ]:
```
# Display the number of rows before and after removal
original_count = len(df)
cleaned_count = len(df_no_outliers)

original_count, cleaned_count
```

Feature Engineering

In [35]:
```
print(df['Recovered'].unique())
#print(df['Date_of_Recovery'].isna().sum())
```

```
['Yes' 'No']
```

In [36]:
```
# Clean Recovered column
df['Recovered'] = df['Recovered'].fillna('Unknown')
df['Recovered_cleaned'] = df['Recovered'].str.strip().str.lower()

# Create Death column: if not recovered, then assume death
df['Death'] = df['Recovered_cleaned'].apply(lambda x: 'No' if x == 'yes' else 'Y
```

```
# Print how many people are marked as Death = Yes or No
print(df['Death'].value_counts())
```

```
Death
No     1508
Yes    1492
Name: count, dtype: int64
```

In [37]:
```
df["Death"].unique()
```

Out[37]:
```
array(['No', 'Yes'], dtype=object)
```

In [38]:
```
# Show the new Death column with related columns for verification
df[['Patient_ID', 'Recovered', 'Date_of_Recovery', 'Death']].head()
```

Out[38]:

	Patient_ID	Recovered	Date_of_Recovery	Death
0	1	Yes	2023-04-19	No
1	2	No	NaN	Yes
2	3	No	NaN	Yes
3	4	Yes	2025-02-09	No
4	5	No	NaN	Yes

Checking the skewness for Numerical columns

In [182…
```
# Select numeric columns
numeric_cols = ['Age', 'Doses_Received', 'BMI']

# Calculate skewness for each column
skewness = df[numeric_cols].apply(skew)

skewness
```

Out[182…
```
Age              -0.015157
Doses_Received    0.682816
BMI              -0.029625
dtype: float64
```

Age: Skewness ≈ -0.015 — distribution is nearly symmetric.(Almost symmetrical)

Doses_Received: Skewness ≈ +0.683 — moderately right-skewed; some individuals received higher doses.(Moderately positively skewed (right-tail))

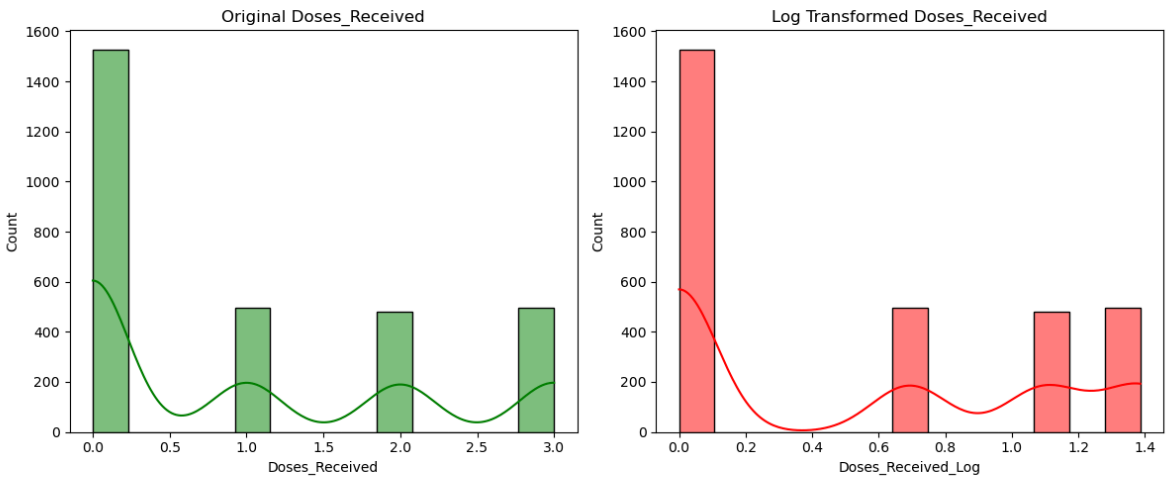
BMI: Skewness ≈ -0.030 — distribution is almost symmetric with slight left tilt.(Almost symmetrical)

log Transformation

In [183…
```
# Apply log transformation to Doses_Received
df['Doses_Received_Log'] = np.log1p(df['Doses_Received'])  # Handles 0 values sa
```

In [184…

```
# Plot before and after transformation
plt.figure(figsize=(12, 5))

# Original
plt.subplot(1, 2, 1)
sns.histplot(df['Doses_Received'], kde=True, color='green')
plt.title("Original Doses_Received")

# Transformed
plt.subplot(1, 2, 2)
sns.histplot(df['Doses_Received_Log'], kde=True, color='red')
plt.title("Log Transformed Doses_Received")
plt.tight_layout()
plt.show()
```



The original Doses_Received column was moderately right-skewed with a long tail of high values. After applying log transformation, the distribution became more symmetric and compact, making it suitable for modeling.

In [185…

```
# Age Grouping
df['Age_Group'] = pd.cut(df['Age'], bins=[0, 18, 35, 50, 65, 100], labels=['Chil
```

In [186…

```
# BMI Category
df['BMI_Category'] = pd.cut(df['BMI'], bins=[0, 18.5, 24.9, 29.9, 100], labels=[
```

In [187…

```
# Total Risk Score (mock feature combining Age and BMI)
df['Risk_Score'] = df['Age'] * df['BMI']
```

In [41]:

```
# Recovery Time
if 'Date_of_Recovery' in df.columns and 'Date_of_Infection' in df.columns:
    df['Date_of_Recovery'] = pd.to_datetime(df['Date_of_Recovery'], errors='coer
    df['Date_of_Infection'] = pd.to_datetime(df['Date_of_Infection'], errors='cc
    df['Recovery_Days'] = (df['Date_of_Recovery'] - df['Date_of_Infection']).dt.
```

In [189…

```
# Preview new columns
print(df[['Age_Group', 'BMI_Category', 'Risk_Score', 'Recovery_Days']].head())
```

```
  Age_Group BMI_Category  Risk_Score  Recovery_Days
0   Elderly   Overweight      1911.3          302.0
1     Adult       Normal       832.2            NaN
2     Adult       Normal       930.7            NaN
3   Elderly   Overweight      2243.7          546.0
4     Adult  Underweight       595.0            NaN
```

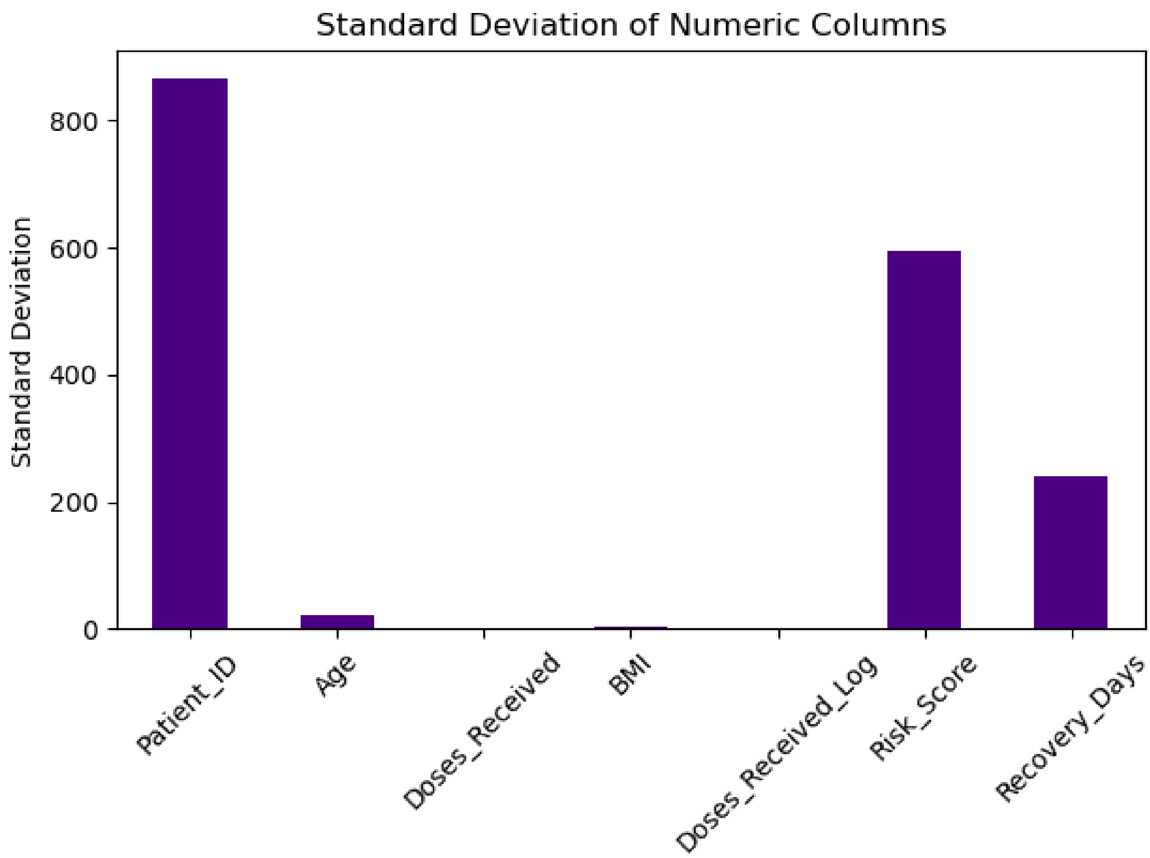
In [190…

```python
# Show only numeric columns
numeric_cols = df.select_dtypes(include='number')

# Calculate standard deviation
std_devs = numeric_cols.std()

# Display
print("📊 Standard Deviation of Numeric Columns:\n")
print(std_devs)
```

```
📊 Standard Deviation of Numeric Columns:

Patient_ID           866.169729
Age                   20.872919
Doses_Received         1.154025
BMI                    4.898435
Doses_Received_Log     0.566189
Risk_Score           593.294407
Recovery_Days        239.422352
dtype: float64
```

In [191…

```python
std_devs.plot(kind='bar', color='indigo')
plt.title('Standard Deviation of Numeric Columns')
plt.ylabel('Standard Deviation')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Visualise the standard deviation using plots

EDA (Exploratory Data Analysis)

Seprate the each n every column as per given data

In [192…

```
continuous_cols = ["Age","Recovery_Days", "BMI", "Risk_Score"]

discrete_cols = ["Doses_Received"]

categorical_cols = ["Gender","Age_Group", "BMI_Category","Long_COVID_Symptoms","
```

Count the unique values and values counts for each columns

In [193…

```
for i in continuous_cols:
    print(i,':',df[i].unique())
    print('=================================================================
    print(i,':',df[i].nunique())
    print('=================================================================
    print(i,':',df[i].value_counts())
```

```
Age : [69 38 41 81 50 66 76 77 79 72 20 56 35 70 64 53 23 71 80 61 89 32 82 18
 25 22 45 40 54 52 44 68 59 87 49 75 73 19 85 34 55 28 33 36 84 88 74 26
 37 42 65 39 86 63 43 24 27 21 67 29 60 30 51 46 47 62 83 31 57 48 58 78]
==================================================================================
==========
Age : 72
==================================================================================
==========
Age : Age
85    55
30    52
75    52
71    51
79    51
      ..
65    34
58    34
78    31
24    30
67    28
Name: count, Length: 72, dtype: int64
Recovery_Days : [3.020e+02       nan 5.460e+02 6.200e+02 2.390e+02 6.190e+02 6.10
0e+01
 9.100e+01 3.350e+02 4.090e+02 1.040e+02 4.200e+01 2.020e+02 2.060e+02
 4.570e+02 5.810e+02 5.090e+02 4.560e+02 3.900e+02 9.750e+02 1.640e+02
 4.000e+01 7.540e+02 3.650e+02 8.510e+02 3.150e+02 2.080e+02 1.440e+02
 3.600e+02 5.000e+00 2.280e+02 2.610e+02 3.810e+02 7.000e+00 1.800e+02
 3.630e+02 4.260e+02 4.930e+02 4.290e+02 6.550e+02 2.090e+02 2.230e+02
 2.640e+02 6.830e+02 9.900e+01 1.010e+02 3.740e+02 1.540e+02 2.050e+02
 1.610e+02 1.930e+02 4.770e+02 3.270e+02 2.750e+02 5.250e+02 1.330e+02
 4.370e+02 4.200e+02 3.440e+02 8.810e+02 1.980e+02 3.300e+02 5.340e+02
 5.650e+02 1.000e+02 8.630e+02 6.710e+02 3.250e+02 4.720e+02 8.800e+01
 1.150e+02 7.280e+02 8.430e+02 2.140e+02 2.000e+00 1.730e+02 7.800e+01
 1.900e+02 2.620e+02 5.060e+02 6.610e+02 1.000e+00 5.530e+02 2.600e+01
 2.220e+02 5.830e+02 7.980e+02 2.580e+02 4.680e+02 3.060e+02 3.140e+02
 2.250e+02 1.520e+02 1.550e+02 6.170e+02 2.700e+02 5.800e+01 5.550e+02
 4.960e+02 3.320e+02 5.110e+02 1.680e+02 4.220e+02 2.550e+02 5.200e+01
 8.460e+02 2.540e+02 4.100e+01 4.920e+02 3.800e+02 1.000e+01 8.100e+01
 1.380e+02 2.480e+02 1.200e+01 2.210e+02 2.010e+02 3.100e+02 7.830e+02
 3.560e+02 6.500e+01 7.100e+01 2.780e+02 4.800e+01 2.900e+02 2.430e+02
 8.080e+02 2.100e+01 5.790e+02 7.640e+02 2.310e+02 8.760e+02 6.960e+02
 7.600e+02 6.730e+02 2.110e+02 1.590e+02 3.820e+02 2.720e+02 4.410e+02
 2.300e+02 3.770e+02 1.140e+02 7.560e+02 6.460e+02 6.880e+02 5.670e+02
 3.700e+01 8.500e+01 2.200e+02 3.310e+02 7.000e+01 4.270e+02 3.370e+02
 1.770e+02 3.360e+02 1.620e+02 7.400e+02 3.500e+02 3.080e+02 6.530e+02
 2.040e+02 3.130e+02 8.200e+01 4.490e+02 4.970e+02 7.420e+02 3.070e+02
 1.400e+01 3.040e+02 6.260e+02 6.240e+02 5.600e+01 1.230e+02 9.600e+01
 3.900e+01 4.690e+02 3.680e+02 4.760e+02 6.750e+02 2.590e+02 7.550e+02
 9.500e+01 1.190e+02 7.570e+02 4.520e+02 2.880e+02 5.660e+02 3.490e+02
 3.750e+02 2.930e+02 3.450e+02 1.960e+02 1.530e+02 4.080e+02 1.750e+02
 7.790e+02 2.950e+02 7.300e+01 5.600e+02 8.600e+02 1.670e+02 5.470e+02
 4.670e+02 5.450e+02 6.630e+02 9.060e+02 9.110e+02 6.000e+00 5.260e+02
 3.380e+02 7.400e+01 1.490e+02 7.470e+02 5.890e+02 1.710e+02 7.900e+01
 5.040e+02 4.750e+02 3.600e+01 5.570e+02 7.500e+02 3.390e+02 1.320e+02
 1.470e+02 3.880e+02 7.670e+02 9.800e+01 1.210e+02 5.400e+01 3.410e+02
 5.350e+02 3.920e+02 4.000e+00 4.430e+02 4.340e+02 1.510e+02 8.300e+01
 8.100e+02 9.800e+02 9.130e+02 2.670e+02 3.620e+02 1.310e+02 6.450e+02
 1.360e+02 4.000e+02 6.640e+02 3.670e+02 3.730e+02 3.720e+02 4.890e+02
 5.500e+01 2.850e+02 4.860e+02 1.870e+02 4.350e+02 8.230e+02 1.077e+03
 3.470e+02 1.120e+02 1.890e+02 1.180e+02 6.780e+02 8.870e+02 4.710e+02
```

```
4.870e+02 3.180e+02 5.900e+02 3.700e+02 5.780e+02 1.760e+02 5.240e+02
5.230e+02 4.730e+02 6.220e+02 6.940e+02 3.890e+02 4.280e+02 4.780e+02
4.480e+02 8.380e+02 2.650e+02 6.030e+02 4.440e+02 1.500e+01 3.430e+02
2.400e+01 4.190e+02 1.250e+02 7.380e+02 2.830e+02 3.570e+02 2.380e+02
1.560e+02 6.500e+02 8.160e+02 7.000e+02 1.660e+02 6.380e+02 8.670e+02
3.160e+02 1.350e+02 0.000e+00 3.460e+02 2.180e+02 3.090e+02 1.130e+02
6.180e+02 3.000e+02 9.510e+02 7.200e+01 1.280e+02 8.300e+02 4.580e+02
5.940e+02 9.000e+01 1.060e+02 3.200e+02 2.820e+02 5.160e+02 4.830e+02
6.200e+01 5.760e+02 1.920e+02 2.240e+02 4.130e+02 1.420e+02 8.390e+02
6.270e+02 3.790e+02 9.180e+02 3.520e+02 5.380e+02 4.500e+02 5.170e+02
3.000e+00 5.440e+02 2.360e+02 5.200e+02 4.550e+02 1.370e+02 5.070e+02
1.240e+02 5.080e+02 1.600e+02 6.050e+02 3.530e+02 8.260e+02 1.990e+02
6.400e+01 5.100e+01 9.330e+02 1.460e+02 8.150e+02 5.390e+02 3.580e+02
5.990e+02 1.800e+01 5.730e+02 1.570e+02 4.640e+02 3.170e+02 3.660e+02
4.950e+02 3.330e+02 2.630e+02 3.780e+02 7.300e+02 6.860e+02 8.540e+02
1.400e+02 2.690e+02 1.580e+02 4.250e+02 8.660e+02 4.790e+02 5.140e+02
1.030e+02 4.990e+02 4.360e+02 4.320e+02 3.220e+02 5.860e+02 3.300e+01
1.050e+02 2.530e+02 2.410e+02 5.630e+02 6.690e+02 6.110e+02 5.270e+02
5.820e+02 1.700e+02 1.700e+01 3.610e+02 1.430e+02 6.900e+01 4.380e+02
1.100e+01 9.400e+01 1.830e+02 3.840e+02 6.480e+02 4.600e+02 5.310e+02
2.660e+02 9.480e+02 3.010e+02 2.560e+02 2.330e+02 3.690e+02 3.830e+02
2.130e+02 8.070e+02 9.740e+02 5.120e+02 4.150e+02 6.950e+02 4.310e+02
5.610e+02 5.800e+02 5.000e+02 4.450e+02 2.290e+02 3.190e+02 6.600e+01
5.700e+01 1.600e+01 9.640e+02 1.500e+02 8.190e+02 2.490e+02 2.900e+01
2.910e+02 2.770e+02 5.330e+02 4.110e+02 9.360e+02 2.070e+02 4.040e+02
2.730e+02 6.130e+02 7.070e+02 2.470e+02 4.850e+02 7.140e+02 2.270e+02
2.120e+02 1.690e+02 6.970e+02 3.230e+02 4.900e+02 2.760e+02 9.400e+02
3.260e+02 7.890e+02 8.000e+00 5.850e+02 2.920e+02 2.570e+02 3.850e+02
4.030e+02 4.540e+02 5.920e+02 4.210e+02 1.044e+03 9.030e+02 4.300e+02
6.000e+01 1.042e+03 7.960e+02 1.900e+01 2.970e+02 6.700e+01 5.490e+02
5.410e+02 4.800e+02 3.200e+01 4.600e+01 5.190e+02 3.400e+02 8.480e+02
3.050e+02 6.740e+02 8.270e+02 9.790e+02 1.008e+03 7.770e+02 7.800e+02
2.890e+02 4.660e+02 3.030e+02 5.370e+02 1.340e+02 2.500e+01 9.020e+02
3.590e+02 2.940e+02 4.180e+02 4.050e+02 2.200e+01 5.010e+02 2.710e+02
6.020e+02 1.850e+02 5.950e+02 6.310e+02 8.850e+02 1.790e+02 7.840e+02
1.300e+02 4.300e+01 2.440e+02 4.160e+02 8.470e+02 2.100e+02 4.910e+02
2.860e+02 4.980e+02 4.400e+01 8.550e+02 7.210e+02 5.750e+02 4.630e+02
6.010e+02 2.320e+02 2.520e+02 7.020e+02 3.100e+01 2.160e+02 3.980e+02
9.140e+02 3.510e+02 1.840e+02 2.800e+01 6.470e+02 6.850e+02 3.340e+02
7.430e+02 4.820e+02 7.060e+02 8.730e+02 9.580e+02 2.450e+02 7.700e+02
5.360e+02 1.026e+03 2.370e+02 1.260e+02 5.180e+02 3.940e+02 3.500e+01
6.300e+01 7.760e+02 8.700e+01 5.210e+02 8.900e+01 8.350e+02 7.090e+02
5.910e+02 5.430e+02 6.840e+02 9.690e+02 2.680e+02 7.500e+01 1.160e+02
3.110e+02 5.640e+02 3.420e+02 9.700e+02 5.300e+02 3.960e+02 3.930e+02
7.190e+02 1.051e+03 3.480e+02 4.620e+02 1.950e+02 8.110e+02 5.020e+02
4.700e+02 6.920e+02 8.210e+02 2.500e+02 8.000e+01 2.300e+01 4.530e+02
9.300e+01 4.170e+02 2.700e+01 5.030e+02 3.760e+02 6.900e+02 1.910e+02
5.870e+02 4.330e+02 9.290e+02 6.100e+02 7.200e+02 8.600e+01 7.650e+02
1.810e+02 2.990e+02 7.010e+02 6.250e+02 7.970e+02 9.650e+02 1.860e+02
2.980e+02 7.690e+02 1.080e+02 1.300e+01 5.720e+02 2.340e+02 8.710e+02
6.620e+02 3.910e+02 7.590e+02 5.130e+02 6.670e+02 2.800e+02 3.710e+02
4.390e+02 2.960e+02 5.690e+02 1.020e+02 3.870e+02 7.270e+02 6.390e+02
6.890e+02 2.000e+02 4.470e+02 2.030e+02 7.170e+02 9.710e+02 6.060e+02
6.590e+02 6.650e+02 4.460e+02 4.070e+02 7.160e+02 7.290e+02 1.270e+02
1.170e+02 6.080e+02 9.200e+01 9.920e+02 3.550e+02 9.000e+00 6.980e+02
4.900e+01 1.480e+02 1.063e+03 2.790e+02 9.700e+01 6.820e+02 8.490e+02
1.410e+02 7.700e+01 3.800e+01 1.220e+02 2.510e+02 6.400e+02 8.220e+02
4.650e+02 3.000e+01 3.290e+02 9.590e+02 1.200e+02 1.058e+03 6.700e+02
4.100e+02 4.500e+01 3.400e+01 8.680e+02 6.800e+01 8.030e+02 5.300e+01
7.480e+02 1.940e+02 9.100e+02 2.840e+02 7.410e+02 8.330e+02 8.900e+02
```

```
 2.260e+02 1.450e+02 5.590e+02 8.860e+02 7.080e+02 4.020e+02 5.480e+02
 6.370e+02 1.005e+03 1.970e+02 2.460e+02 9.600e+02 1.034e+03 2.000e+01
 5.840e+02 6.930e+02 4.140e+02 1.100e+02 8.050e+02 6.520e+02 6.870e+02]
==================================================================================
==========
Recovery_Days : 706
==================================================================================
==========
Recovery_Days : Recovery_Days
437.0    7
323.0    7
457.0    6
177.0    6
486.0    6
        ..
431.0    1
695.0    1
415.0    1
974.0    1
687.0    1
Name: count, Length: 706, dtype: int64
BMI : [27.7 21.9 22.7 11.9 29.8 22.3 24.4 26.1 21.2 27.1 29.2 22.  29.7 24.9
 19.7 18.7 30.5 18.4 26.  28.7 25.9 17.6 21.4 19.3 25.7 24.8 23.6 30.2
 28.  20.5 17.1 28.2 26.8 21.8 32.5 26.7 24.2 20.2 29.5 25.1 33.1 29.1
 23.4 22.9 18.  11.7 20.9 21.7 24.3 27.3 24.7 25.  23.9 21.6 32.  22.5
 29.6 33.5 29.4 20.4 32.7 23.2 27.8 25.3 27.5 31.5 20.7 28.1 31.1 18.9
 23.8 38.7 25.5 38.9 15.3 20.3 18.8 19.6 31.9 28.3 17.8 19.1 23.  25.4
 15.6 24.5 22.6 21.1 33.3 17.3 23.3 35.6 34.4 14.5 28.8 25.6 35.7 15.2
 30.3 32.8 20.1 32.3 26.9 22.8 12.7 24.6 26.6 21.  26.2 34.1 27.  30.8
 34.3 32.2 15.4 17.9 12.2 30.7 25.2 21.5 35.9 14.4 22.4 32.4 30.9 34.6
 32.6 21.3 18.2 31.7 28.4 31.4 29.  30.  18.6 18.5 14.3 27.6 19.2 34.9
 29.3 27.4 20.6 17.2 14.6 20.8 19.9 27.9 24.  32.9 26.5 13.6 26.3 32.1
 16.8 26.4 16.6 23.7 27.2 33.  28.9 13.4 19.4 11.5 25.8 22.2 30.6 23.5
 28.6 17.5 18.3 16.2 29.9 33.9 30.1 14.9 19.8 19.  22.1 23.1 35.1 31.6
 37.6 16.7 30.4 31.3 31.8 31.  24.1 35.  10.8 35.4 40.7 19.5 34.2 33.2
 31.2 28.5 34.7 13.3 13.5 12.4 36.4 12.  35.8 33.6 16.1 34.  16.4 36.1
 20.  18.1 33.4 11.4 10.4 11.2 37.7 16.9 35.2 35.5 15.7 16.5 15.9 16.
 15.5 17.  15.  16.3 36.2 14.2 11.8 42.5 17.7 15.8 37.1 36.5 39.6 34.8
 15.1 35.3 33.7 33.8 13.8 34.5 37.8 12.3 17.4 44.6 14.  14.1 10.2 13.
 14.7 38.2 12.6 38.4 37.  12.9 13.1 38.8 12.5 11.1 36.9 38.  14.8]
==================================================================================
==========
BMI : 265
==================================================================================
==========
BMI : BMI
25.6    38
26.7    33
25.7    33
22.9    33
24.9    32
        ..
11.2     1
36.2     1
11.8     1
42.5     1
14.8     1
Name: count, Length: 265, dtype: int64
Risk_Score : [1911.3  832.2  930.7 ... 1567.5  868.5 2739. ]
==================================================================================
```

```
==========
Risk_Score : 2436
================================================================================
==========
Risk_Score : Risk_Score
1632.0    6
1320.0    5
858.0     5
756.0     4
1134.0    4
         ..
1131.6    1
1407.0    1
2705.6    1
888.0     1
2739.0    1
Name: count, Length: 2436, dtype: int64
```

In [194…

```
for i in discrete_cols:
    print(i,':',df[i].unique())
    print('======================================================================
    print(i,':',df[i].nunique())
    print('======================================================================
    print(i,':',df[i].value_counts())
```

```
Doses_Received : [1 0 3 2]
================================================================================
==========
Doses_Received : 4
================================================================================
==========
Doses_Received : Doses_Received
0    1528
3     497
1     496
2     479
Name: count, dtype: int64
```

In [195…

```
for i in categorical_cols:
    print(i,':',df[i].unique())
    print('======================================================================
    print(i,':',df[i].nunique())
    print('======================================================================
    print(i,':',df[i].value_counts())
```

```
Gender : ['Male' 'Female']
================================================================================
==========
Gender : 2
================================================================================
==========
Gender : Gender
Female    1527
Male      1473
Name: count, dtype: int64
Age_Group : ['Elderly', 'Adult', 'Youth', 'Senior', 'Child']
Categories (5, object): ['Child' < 'Youth' < 'Adult' < 'Senior' < 'Elderly']
================================================================================
==========
Age_Group : 5
================================================================================
==========
Age_Group : Age_Group
Elderly    1048
Youth       685
Adult       636
Senior      588
Child        43
Name: count, dtype: int64
BMI_Category : ['Overweight', 'Normal', 'Underweight', 'Obese']
Categories (4, object): ['Underweight' < 'Normal' < 'Overweight' < 'Obese']
================================================================================
==========
BMI_Category : 4
================================================================================
==========
BMI_Category : BMI_Category
Normal         1188
Overweight     1058
Obese           488
Underweight     266
Name: count, dtype: int64
Long_COVID_Symptoms : ['None' 'Fatigue' 'Chest Pain' 'Shortness of Breath' 'Brain
Fog']
================================================================================
==========
Long_COVID_Symptoms : 5
================================================================================
==========
Long_COVID_Symptoms : Long_COVID_Symptoms
None                   2780
Fatigue                  62
Brain Fog                59
Chest Pain               52
Shortness of Breath      47
Name: count, dtype: int64
Death : ['No' 'Yes']
================================================================================
==========
Death : 2
================================================================================
==========
Death : Death
No     1508
Yes    1492
```

```
Name: count, dtype: int64
Vaccination_Status : ['Yes' 'No']
================================================================================
==========
Vaccination_Status : 2
================================================================================
==========
Vaccination_Status : Vaccination_Status
No     1528
Yes    1472
Name: count, dtype: int64
```

In [196…

```
# for Numerical Variables
df[continuous_cols].describe()
```

Out[196…

	Age	Recovery_Days	BMI	Risk_Score
count	3000.000000	1508.000000	3000.000000	3000.000000
mean	53.944000	362.282493	25.096500	1351.793367
std	20.872919	239.422352	4.898435	593.294407
min	18.000000	0.000000	10.200000	205.200000
25%	36.000000	172.500000	21.800000	859.650000
50%	54.000000	338.000000	25.100000	1298.000000
75%	72.000000	512.000000	28.500000	1773.375000
max	89.000000	1077.000000	44.600000	3570.000000

In [197…

```
# for discrete Variables
df[discrete_cols].describe()
```

Out[197…

	Doses_Received
count	3000.000000
mean	0.981667
std	1.154025
min	0.000000
25%	0.000000
50%	0.000000
75%	2.000000
max	3.000000

In [198…

```
# for categorical Variables
df[categorical_cols].describe()
```

Out[198…

	Gender	Age_Group	BMI_Category	Long_COVID_Symptoms	Death	Vaccination
count	3000	3000	3000	3000	3000	
unique	2	5	4	5	2	
top	Female	Elderly	Normal	None	No	
freq	1527	1048	1188	2780	1508	

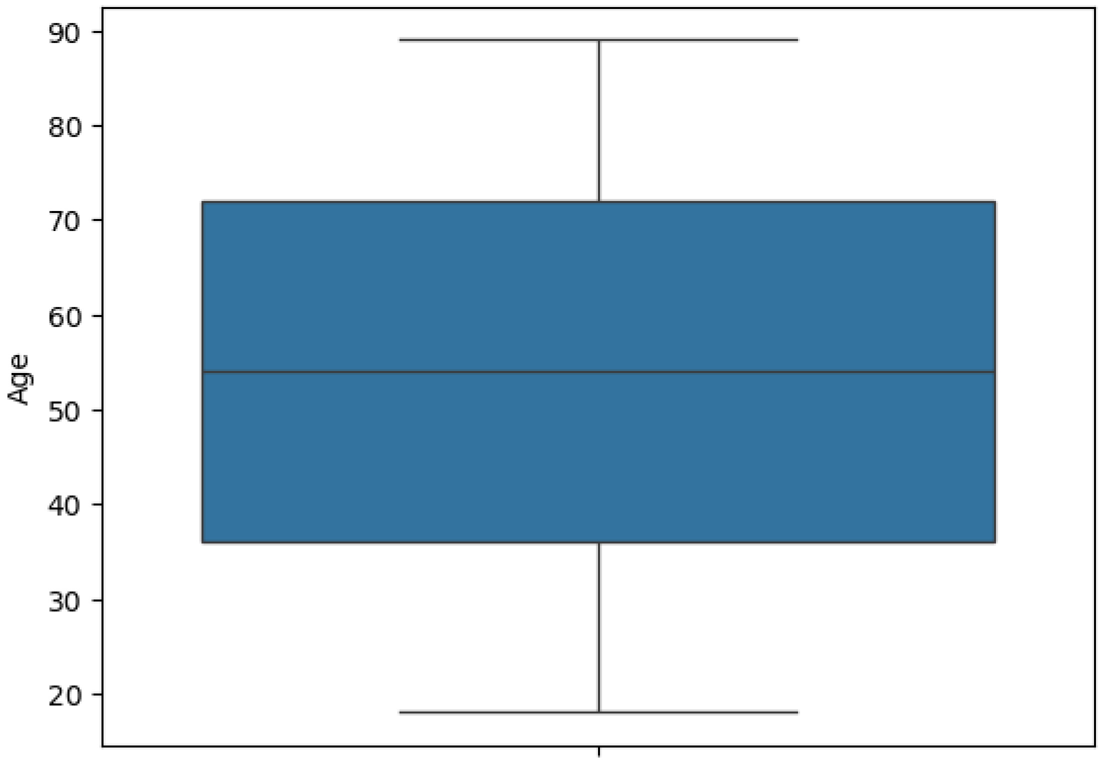
Visualize The Plots

Univariate Plots

Checking Outliers For Continuous Column

In [199…

```
sns.boxplot(df["Age"])
plt.show()
```



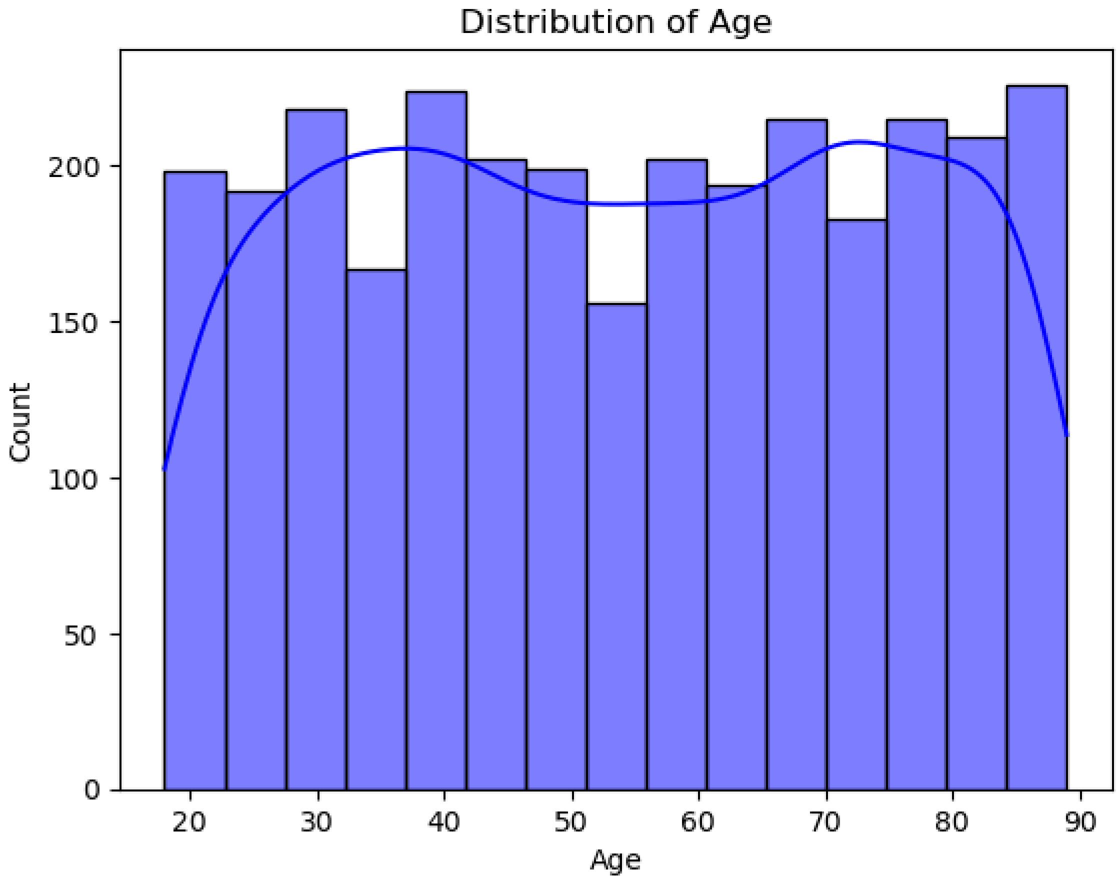
The boxplot shows that most patients are aged between ~37 and ~73, with a median age of ~55, ranging from ~18 to ~90, and no visible outliers.

In [200…

```
sns.histplot(df['Age'],  kde=True,  color='blue')
plt.title('Distribution of Age')
```

Out[200…

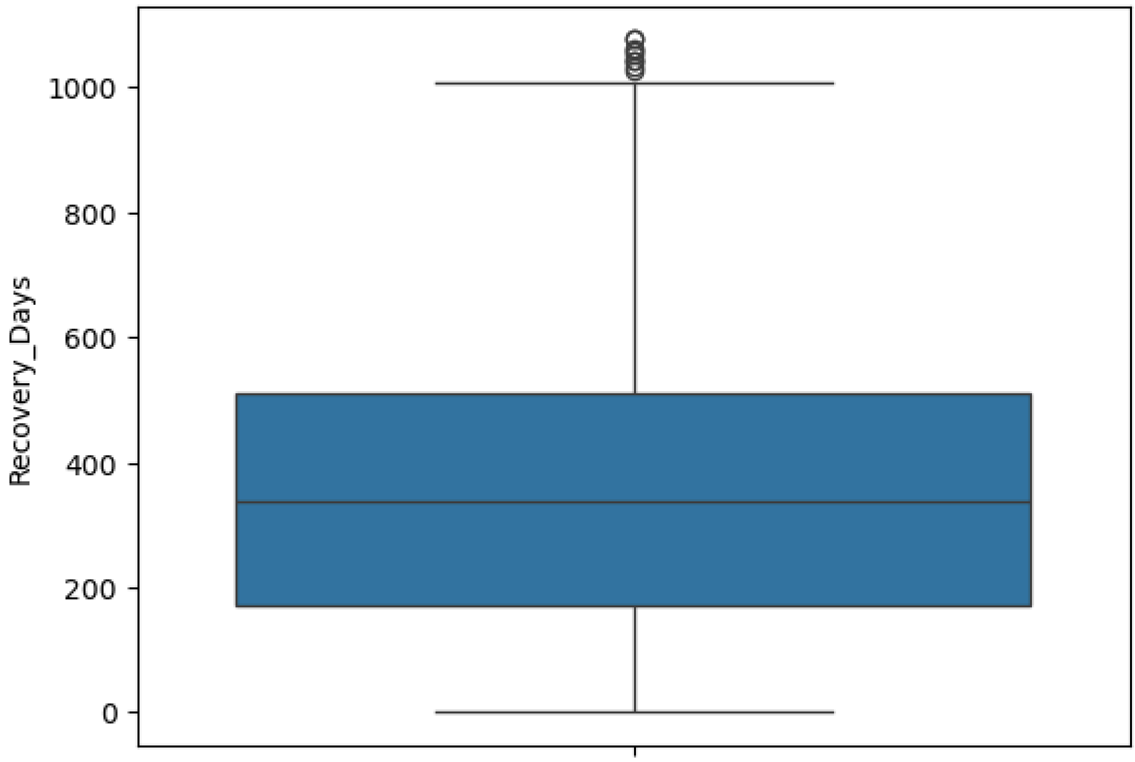
```
Text(0.5, 1.0, 'Distribution of Age')
```



The age distribution is fairly uniform with slight peaks around ages 30–40 and 80–90, indicating a balanced spread across age groups.

In [201…

```
sns.boxplot(df["Recovery_Days"])
plt.show()
```



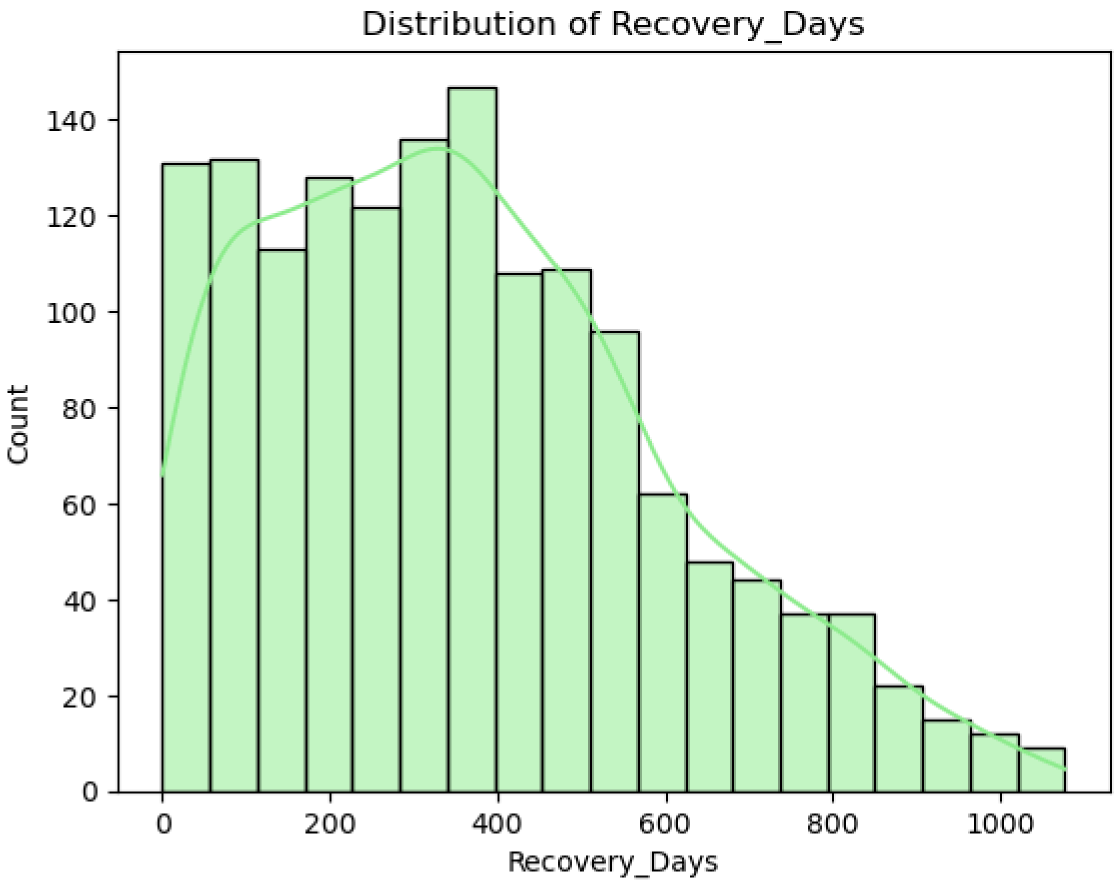
The boxplot shows that most patients recovered within 0–350 days, but there are many extreme outliers above 850 days, indicating unusually long recovery times for some cases.

In [202…

```
sns.histplot(df['Recovery_Days'],  kde=True,  color='lightgreen')
plt.title('Distribution of Recovery_Days')
```

Out[202…

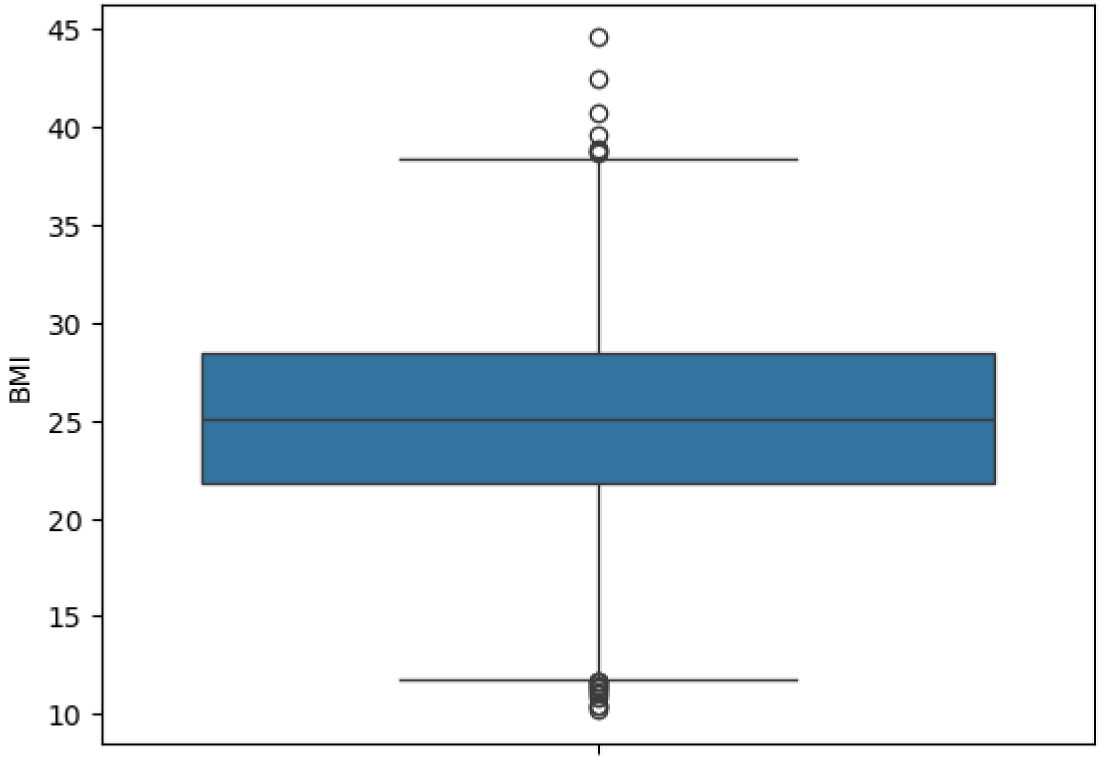
```
Text(0.5, 1.0, 'Distribution of Recovery_Days')
```



The distribution of Recovery_Days is highly right-skewed, indicating that most patients recovered quickly, but a few cases had extremely long recovery durations (potential outliers).

In [203…

```
sns.boxplot(df["BMI"])
plt.show()
```



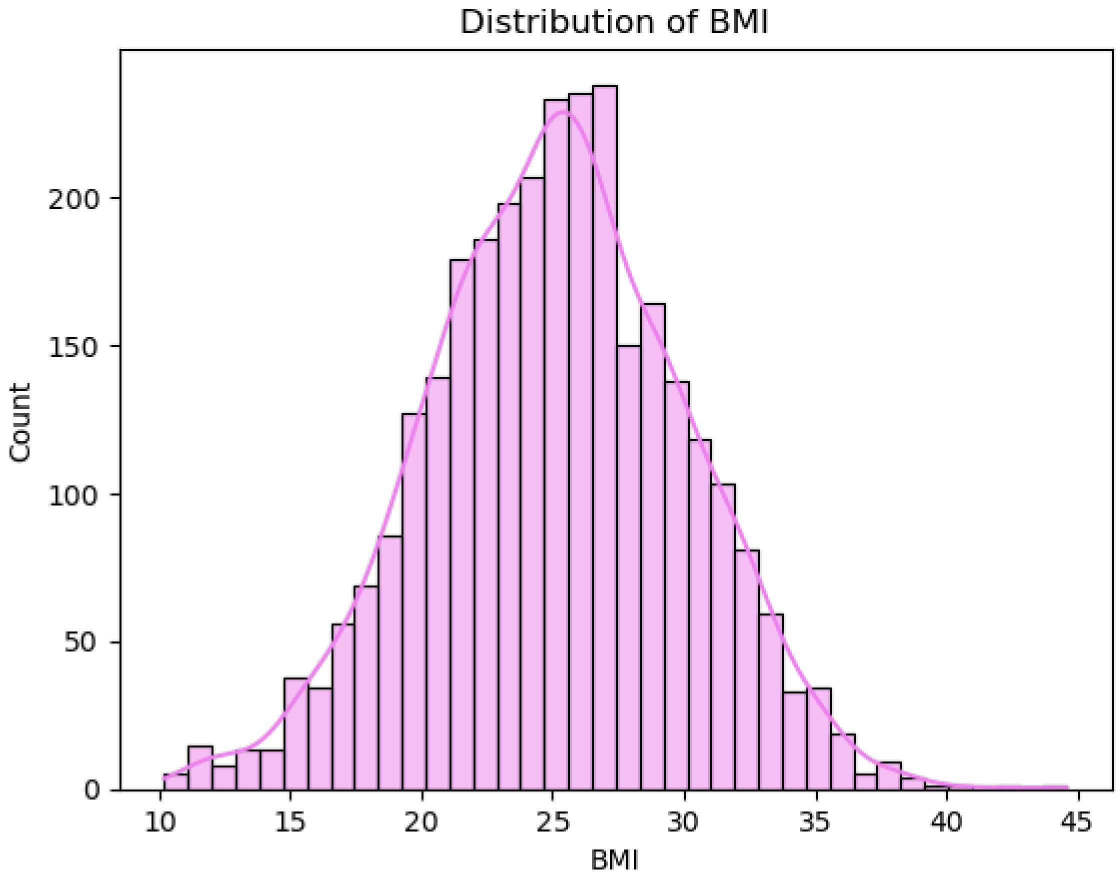
The BMI boxplot shows that most values lie between 22 and 30, but there are several outliers below 15 and above 38, indicating a few underweight and obese individuals.

In [204…

```
sns.histplot(df['BMI'],  kde=True,  color='violet')
plt.title('Distribution of BMI')
```

Out[204…

```
Text(0.5, 1.0, 'Distribution of BMI')
```



The distribution of BMI is slightly right-skewed, indicating that while most individuals have a healthy to slightly overweight BMI, a few outliers have significantly higher values. Most patients have BMI values between 20 and 30, indicating many are in the normal to overweight range.

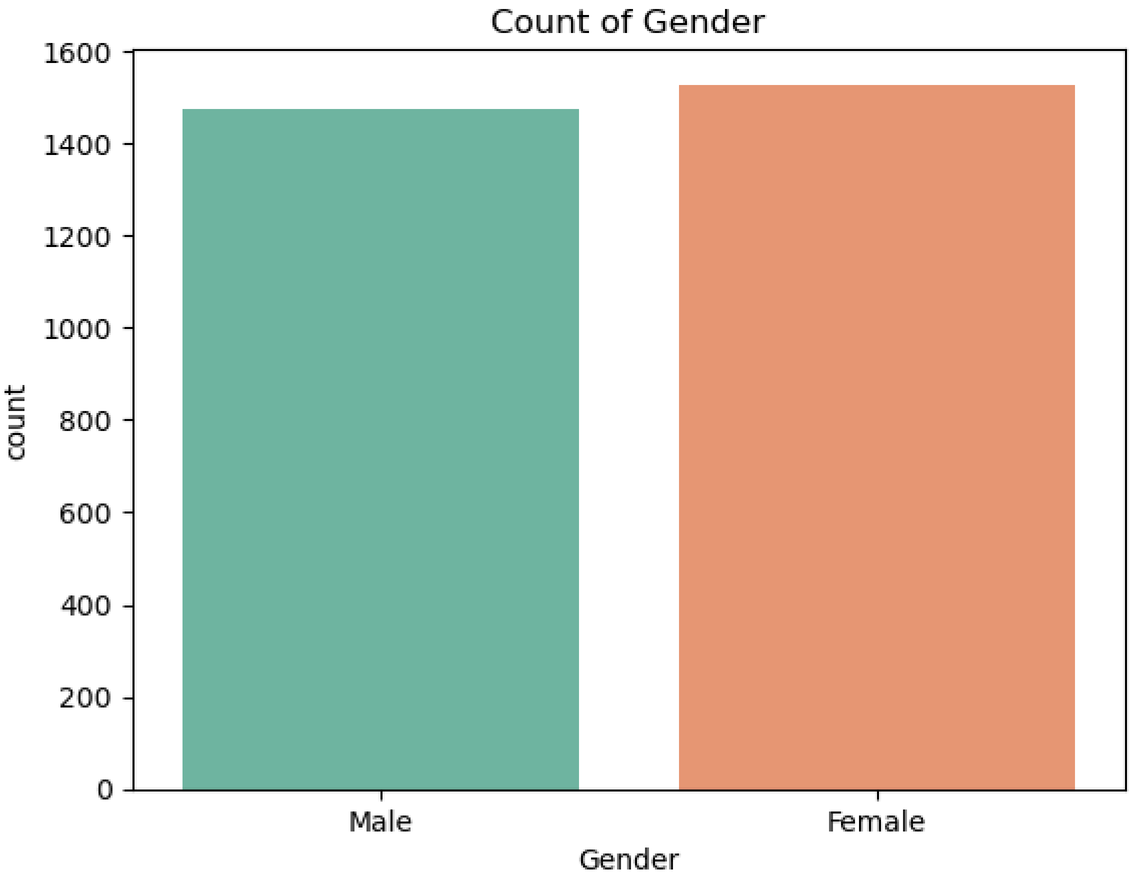
CountPlot

In [205…

```
#  Gender - Countplot
sns.countplot(data=df, x='Gender', palette='Set2')
plt.title('Count of Gender')
```

Out[205…

```
Text(0.5, 1.0, 'Count of Gender')
```

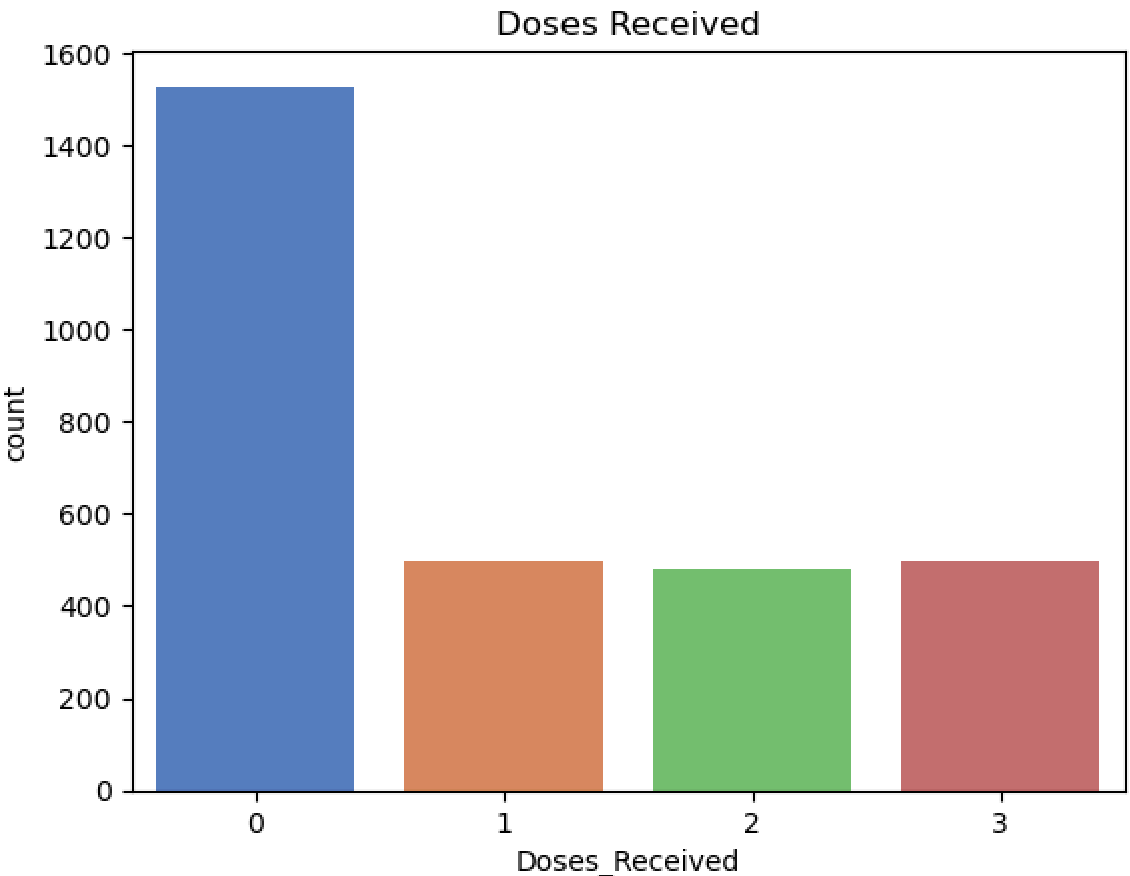


In [206…

```
#  Doses Received - Countplot
sns.countplot(data=df, x='Doses_Received',  palette='muted')
plt.title('Doses Received')
```

Out[206…

```
Text(0.5, 1.0, 'Doses Received')
```

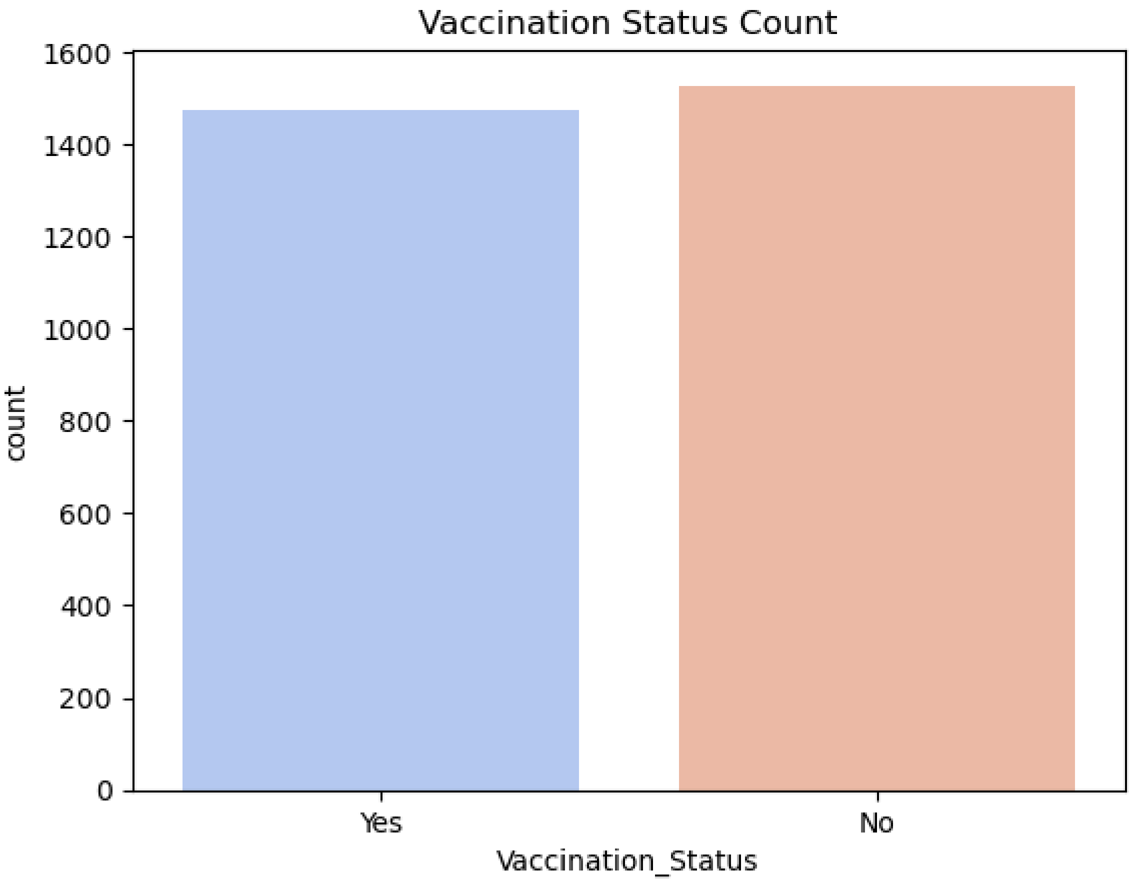


In [207…

```
#  Vaccination Status - Countplot
sns.countplot(data=df, x='Vaccination_Status', palette='coolwarm')
plt.title('Vaccination Status Count')
```

Out[207…

```
Text(0.5, 1.0, 'Vaccination Status Count')
```

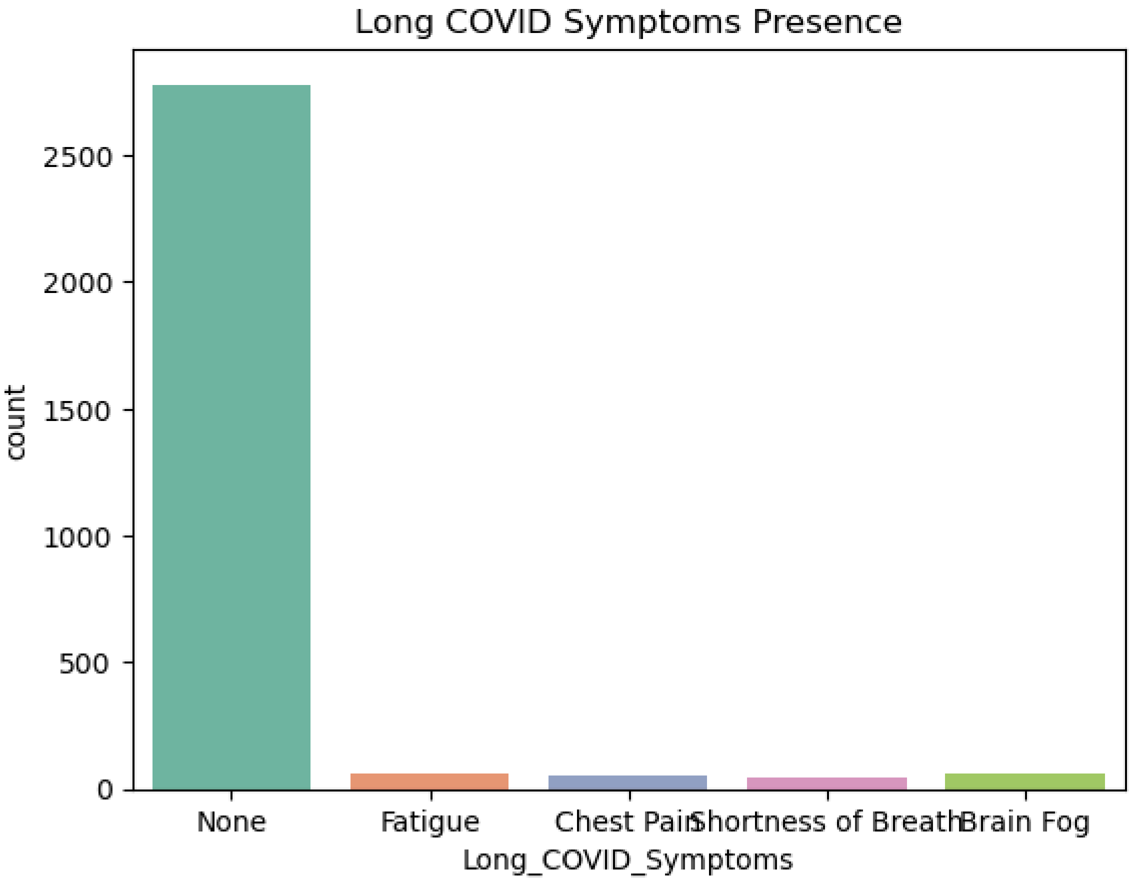


In [208…

```
#  Long COVID Symptoms - Countplot
sns.countplot(data=df, x='Long_COVID_Symptoms',  palette='Set2')
plt.title('Long COVID Symptoms Presence')
```

Out[208…

```
Text(0.5, 1.0, 'Long COVID Symptoms Presence')
```



Bivariate Plots

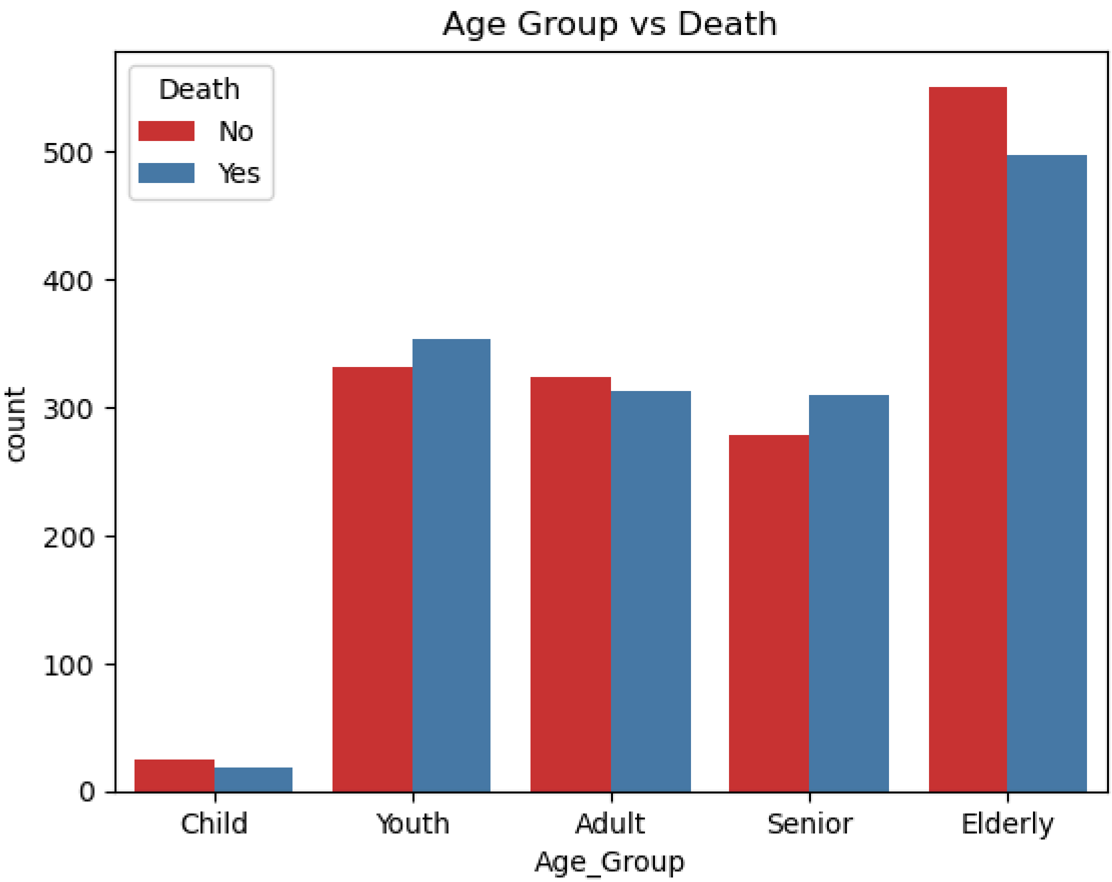
Age Group vs Death

In [209…

```
#  Age Group vs Death
sns.countplot(data=df, x='Age_Group', hue='Death',  palette='Set1')
plt.title('Age Group vs Death')
```

Out[209…

```
Text(0.5, 1.0, 'Age Group vs Death')
```



Vaccination Status vs Death

In [4]:
```
#  Vaccination Status vs Death
sns.countplot(data=df, x='Vaccination_Status', hue='Death',  palette='Set2')
plt.title('Vaccination Status vs Death')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[4], line 2
      1 #  Vaccination Status vs Death
----> 2 sns.countplot(data=df, x='Vaccination_Status', hue='Death',  palette='Set
2')
      3 plt.title('Vaccination Status vs Death')

File ~\anaconda3\Lib\site-packages\seaborn\categorical.py:2631, in countplot(dat
a, x, y, hue, order, hue_order, orient, color, palette, saturation, fill, hue_nor
m, stat, width, dodge, gap, log_scale, native_scale, formatter, legend, ax, **kwa
rgs)
   2628 elif x is not None and y is not None:
   2629     raise TypeError("Cannot pass values for both `x` and `y`.")
-> 2631 p = _CategoricalAggPlotter(
   2632     data=data,
   2633     variables=dict(x=x, y=y, hue=hue),
   2634     order=order,
   2635     orient=orient,
   2636     color=color,
   2637     legend=legend,
   2638 )
   2640 if ax is None:
   2641     ax = plt.gca()

File ~\anaconda3\Lib\site-packages\seaborn\categorical.py:67, in _CategoricalPlot
ter.__init__(self, data, variables, order, orient, require_numeric, color, legen
d)
     56 def __init__(
     57     self,
     58     data=None,
   (...)
     64     legend="auto",
     65 ):
---> 67     super().__init__(data=data, variables=variables)
     69     # This method takes care of some bookkeeping that is necessary becaus
e the
     70     # original categorical plots (prior to the 2021 refactor) had some ru
les that
     71     # don't fit exactly into VectorPlotter logic. It may be wise to have
a second
   (...)
     76     # default VectorPlotter rules. If we do decide to make orient part of
the
     77     # _base variable assignment, we'll want to figure out how to express
that.
     78     if self.input_format == "wide" and orient in ["h", "y"]:

File ~\anaconda3\Lib\site-packages\seaborn\_base.py:634, in VectorPlotter.__init_
_(self, data, variables)
    629 # var_ordered is relevant only for categorical axis variables, and may
    630 # be better handled by an internal axis information object that tracks
    631 # such information and is set up by the scale_* methods. The analogous
    632 # information for numeric axes would be information about log scales.
    633 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 634 self.assign_variables(data, variables)
    636 # TODO Lots of tests assume that these are called to initialize the
    637 # mappings to default values on class initialization. I'd prefer to
    638 # move away from that and only have a mapping when explicitly called.
    639 for var in ["hue", "size", "style"]:
```

```
File ~\anaconda3\Lib\site-packages\seaborn\_base.py:679, in VectorPlotter.assign_
variables(self, data, variables)
    674 else:
    675     # When dealing with long-form input, use the newer PlotData
    676     # object (internal but introduced for the objects interface)
    677     # to centralize / standardize data consumption logic.
    678     self.input_format = "long"
--> 679     plot_data = PlotData(data, variables)
    680     frame = plot_data.frame
    681     names = plot_data.names

File ~\anaconda3\Lib\site-packages\seaborn\_core\data.py:58, in PlotData.__init__
(self, data, variables)
     51 def __init__(
     52     self,
     53     data: DataSource,
     54     variables: dict[str, VariableSpec],
     55 ):
     57     data = handle_data_source(data)
---> 58     frame, names, ids = self._assign_variables(data, variables)
     60     self.frame = frame
     61     self.names = names

File ~\anaconda3\Lib\site-packages\seaborn\_core\data.py:232, in PlotData._assign
_variables(self, data, variables)
    230     else:
    231         err += "An entry with this name does not appear in `data`."
--> 232     raise ValueError(err)
    234 else:
    235 
    236     # Otherwise, assume the value somehow represents data
    237 
    238     # Ignore empty data structures
    239     if isinstance(val, Sized) and len(val) == 0:

ValueError: Could not interpret value `Death` for `hue`. An entry with this name
does not appear in `data`.
```

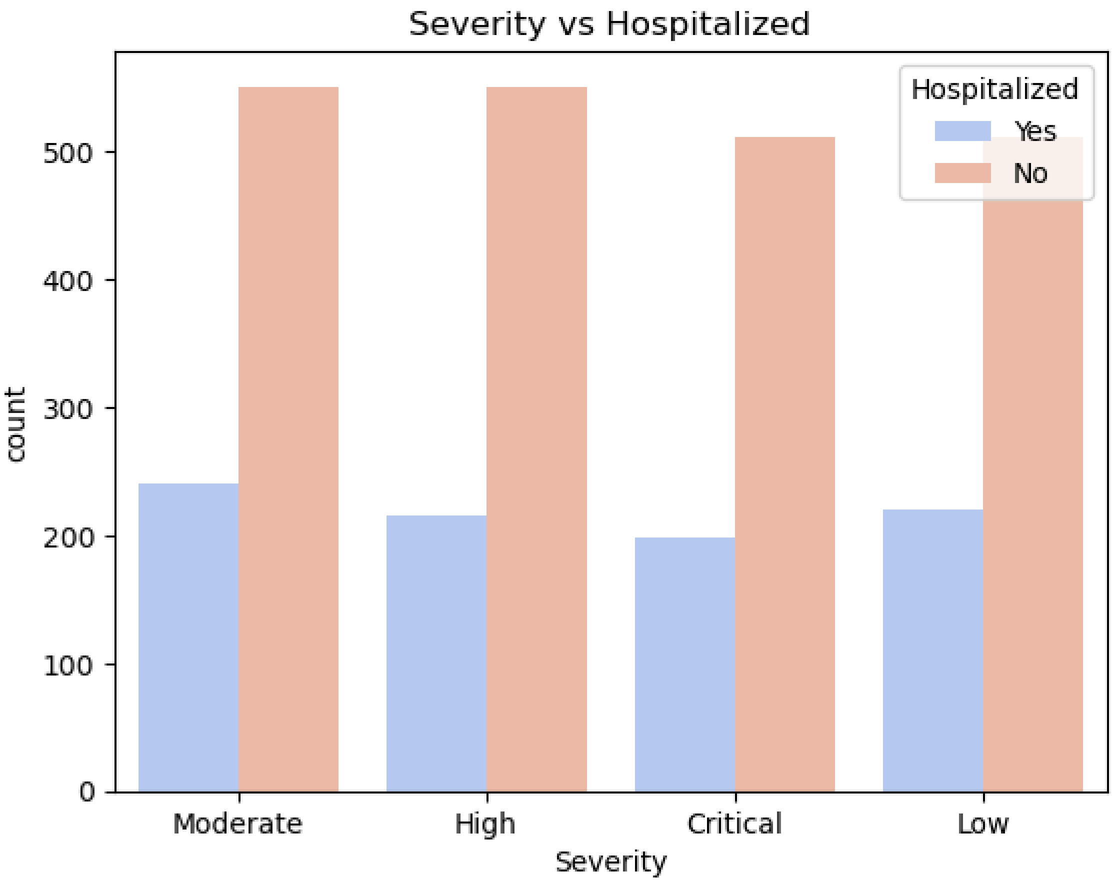
Severity vs Hopitalized

In [211…

```python
# Severity vs Hospitalized
sns.countplot(data=df, x='Severity', hue='Hospitalized',  palette='coolwarm')
plt.title('Severity vs Hospitalized')
```

Out[211…

```
Text(0.5, 1.0, 'Severity vs Hospitalized')
```



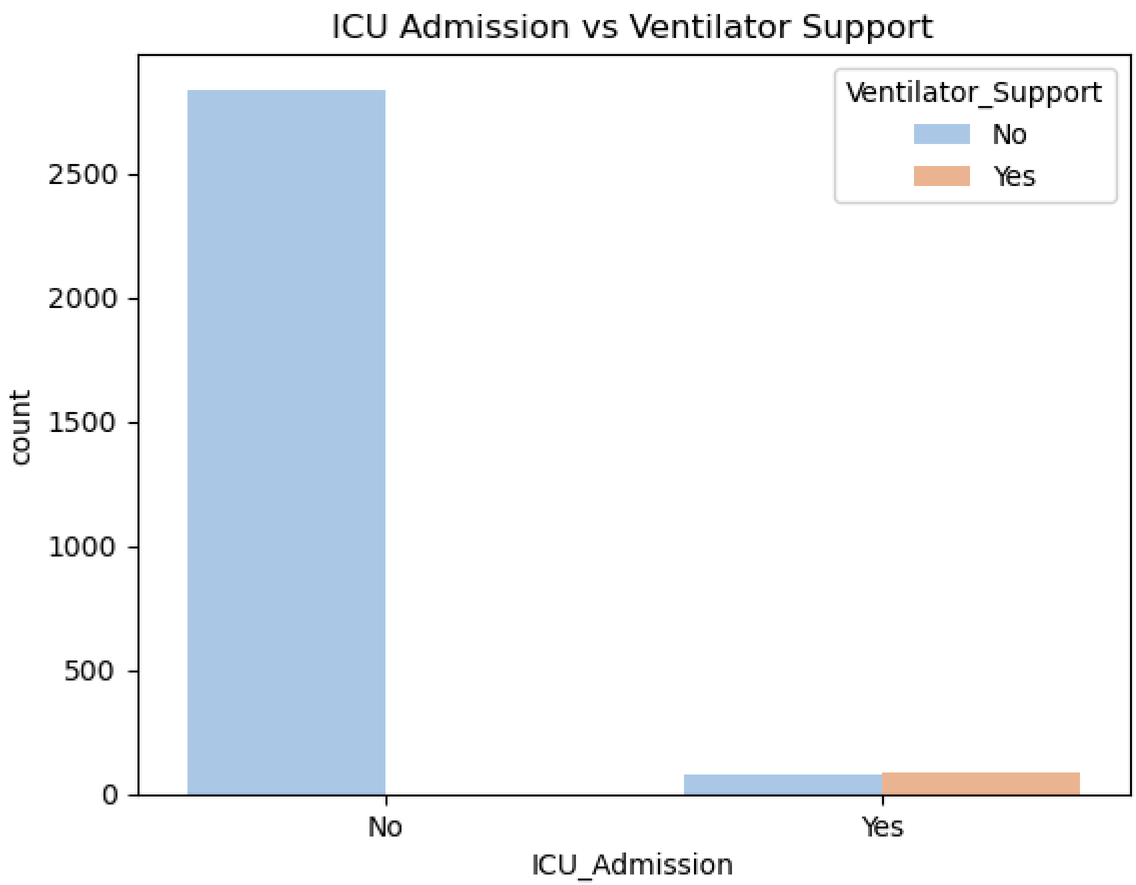
ICU Admission vs Ventilator Support

In [212…

```
#  ICU Admission vs Ventilator Support
sns.countplot(data=df, x='ICU_Admission', hue='Ventilator_Support', palette='pas
plt.title('ICU Admission vs Ventilator Support')
```

Out[212…

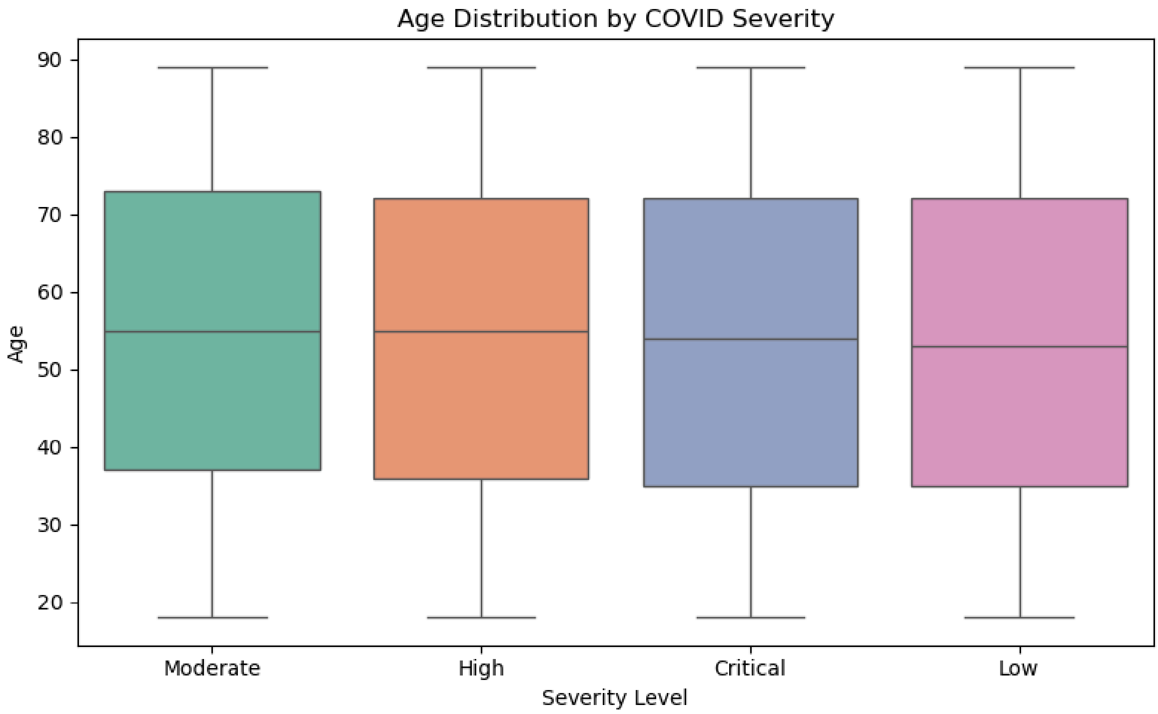
```
Text(0.5, 1.0, 'ICU Admission vs Ventilator Support')
```



Age Distribution by Covid Severity

In [213…

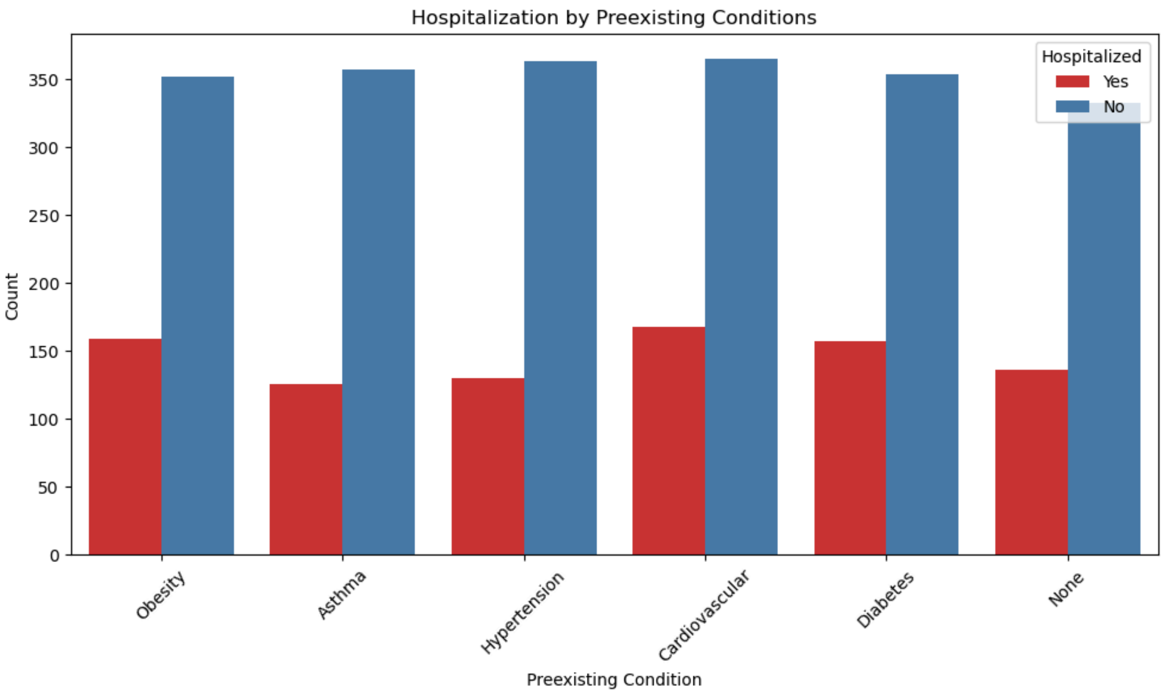
```
# Does Age affect COVID Severity?

plt.figure(figsize=(8, 5))
sns.boxplot(x='Severity', y='Age', data=df, palette='Set2')
plt.title('Age Distribution by COVID Severity')
plt.xlabel('Severity Level')
plt.ylabel('Age')
plt.tight_layout()
plt.show()
```



Hospitalization by Preexisting Conditions

In [214…

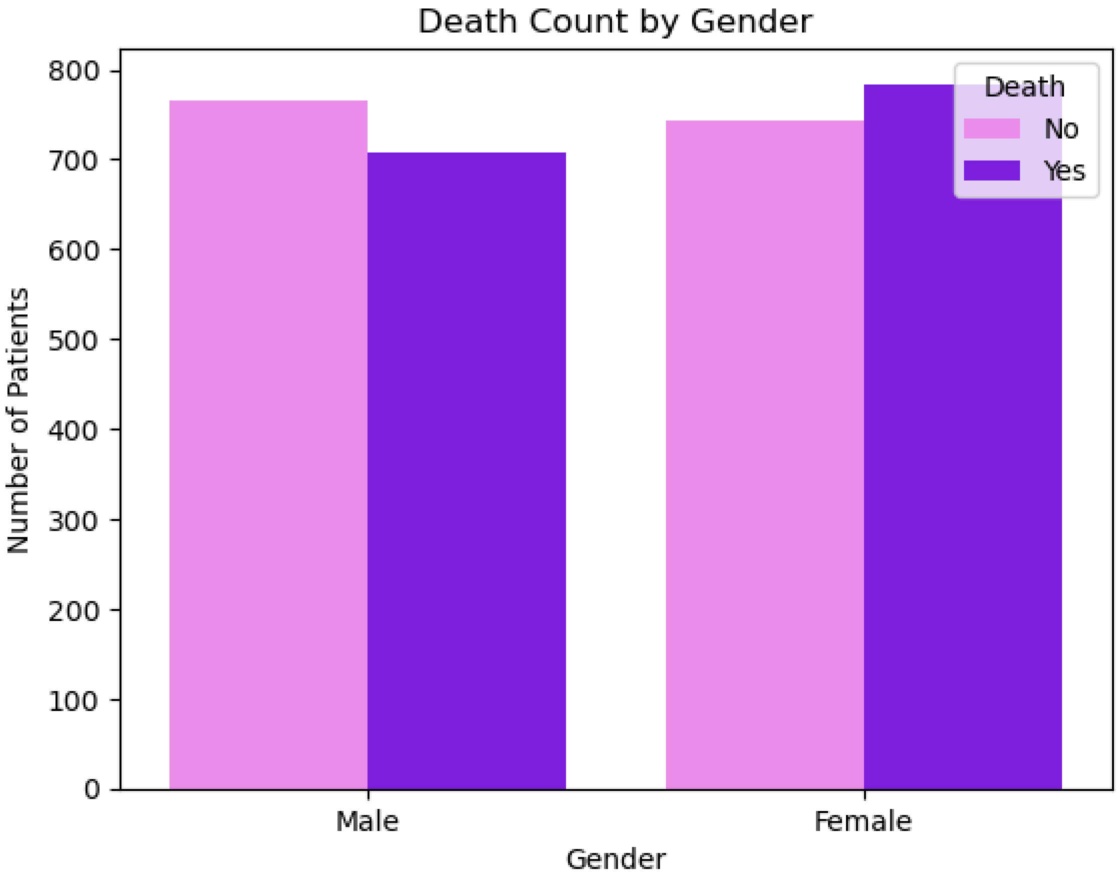
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Preexisting_Condition', hue='Hospitalized', palette='S
plt.xticks(rotation=45)
plt.title('Hospitalization by Preexisting Conditions')
plt.xlabel('Preexisting Condition')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



Gender vs Death Outcome

In [215…

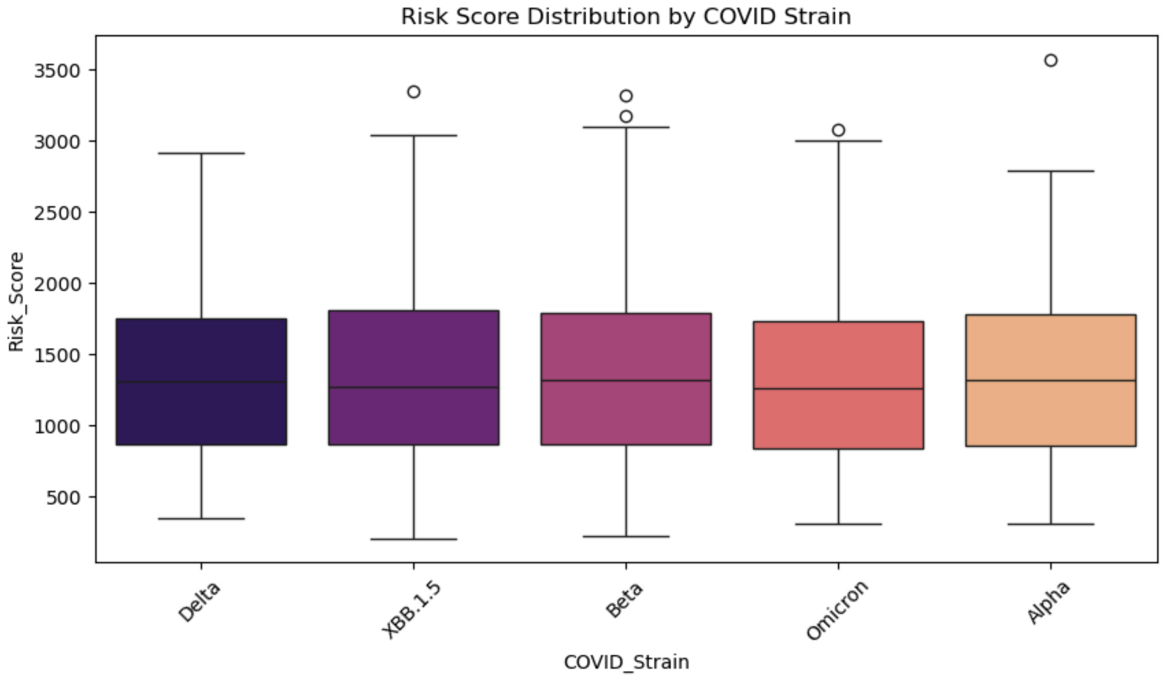
```
sns.countplot(data=df, x='Gender', hue='Death', palette=["#ff80ff", "#8000ff"])
plt.title('Death Count by Gender')
plt.ylabel('Number of Patients')
plt.xlabel('Gender')
plt.show()
```



COVID Strain vs Risk Score

In [216…

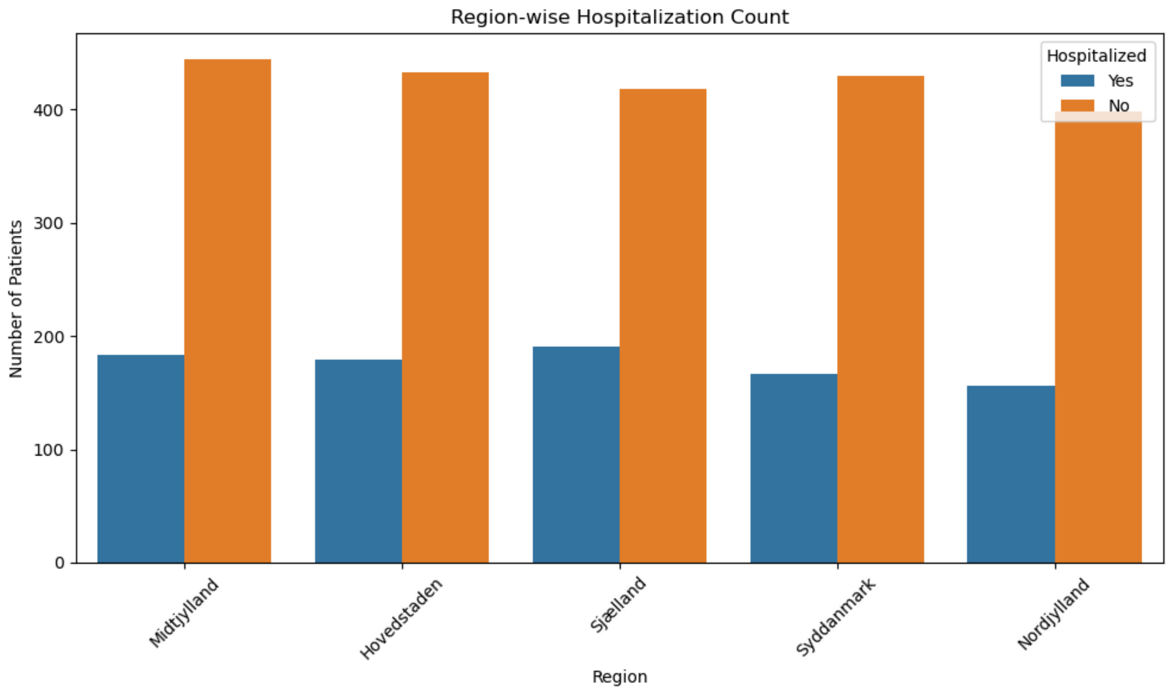
```
plt.figure(figsize=(10, 5))
sns.boxplot(data=df, x='COVID_Strain', y='Risk_Score', palette='magma')
plt.title('Risk Score Distribution by COVID Strain')
plt.xticks(rotation=45)
plt.show()
```



Region vs Hospitalized

In [217…

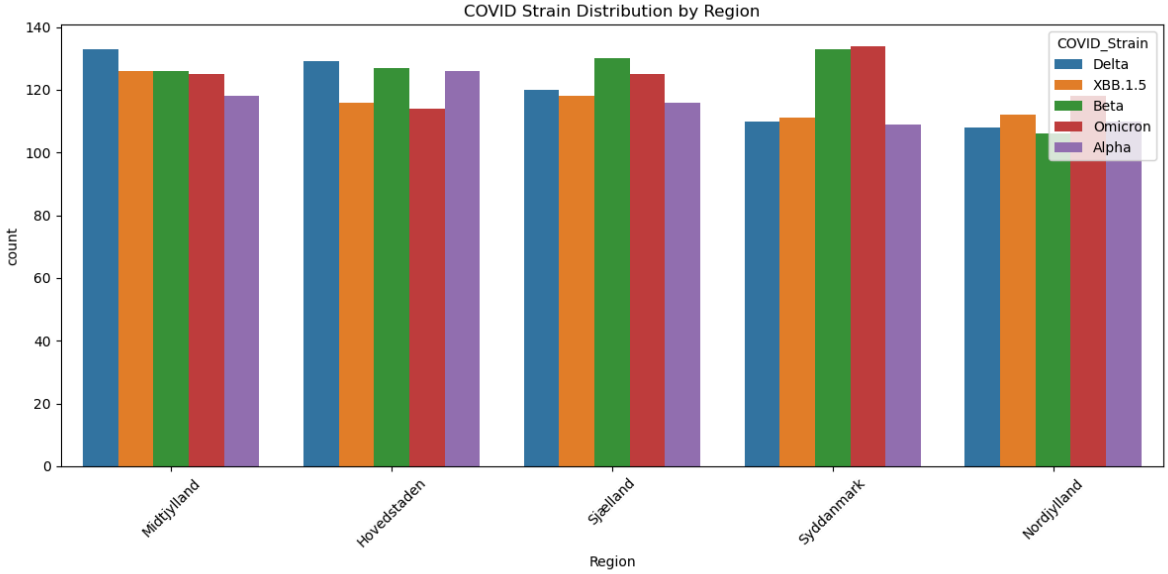
```
plt.figure(figsize=(10,6))
sns.countplot(data=df, x="Region", hue="Hospitalized", order=df["Region"].value_

plt.title("Region-wise Hospitalization Count")
plt.xlabel("Region")
plt.ylabel("Number of Patients")
plt.xticks(rotation=45)
plt.legend(title="Hospitalized")
plt.tight_layout()
plt.show()
```



Region + COVID Strain vs Hospitalized

In [218…

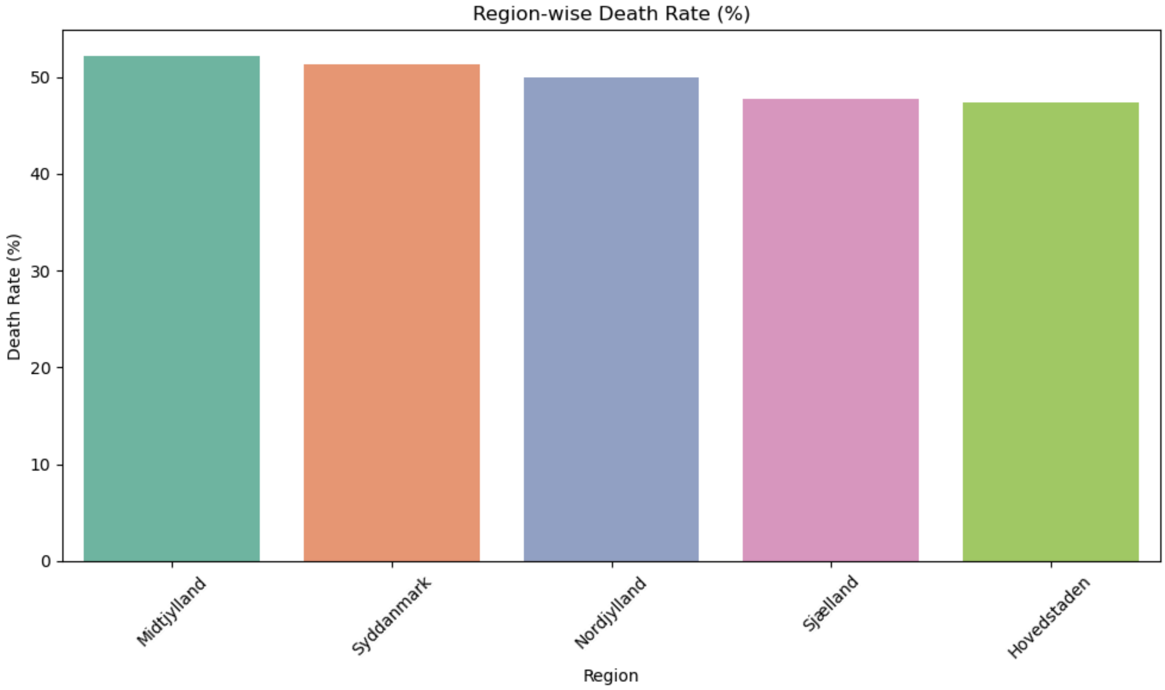
```
plt.figure(figsize=(12,6))
sns.countplot(data=df, x="Region", hue="COVID_Strain", order=df["Region"].value_
plt.title("COVID Strain Distribution by Region")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Region-wise Death Rate(%)

In [219…

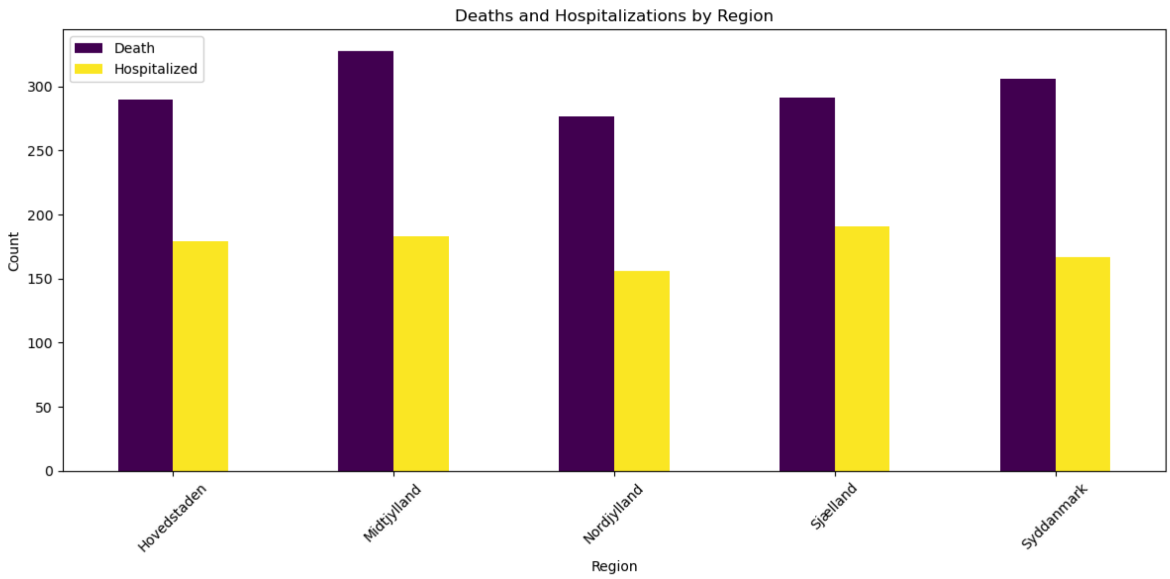
```
region_death_rate = (
    df.groupby("Region")["Death"]
    .value_counts(normalize=True)
    .unstack()
    .fillna(0) * 100
).reset_index()

plt.figure(figsize=(10,6))
sns.barplot(data=region_death_rate, x="Region", y="Yes", order=region_death_rate
plt.title("Region-wise Death Rate (%)")
plt.ylabel("Death Rate (%)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Which region has the highest number of deaths and hospitalizations?

In [220…

```
region_death_hosp = df.groupby('Region')[['Death', 'Hospitalized']].apply(lambda
region_death_hosp.plot(kind='bar', figsize=(12, 6), colormap='viridis')
plt.title('Deaths and Hospitalizations by Region')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

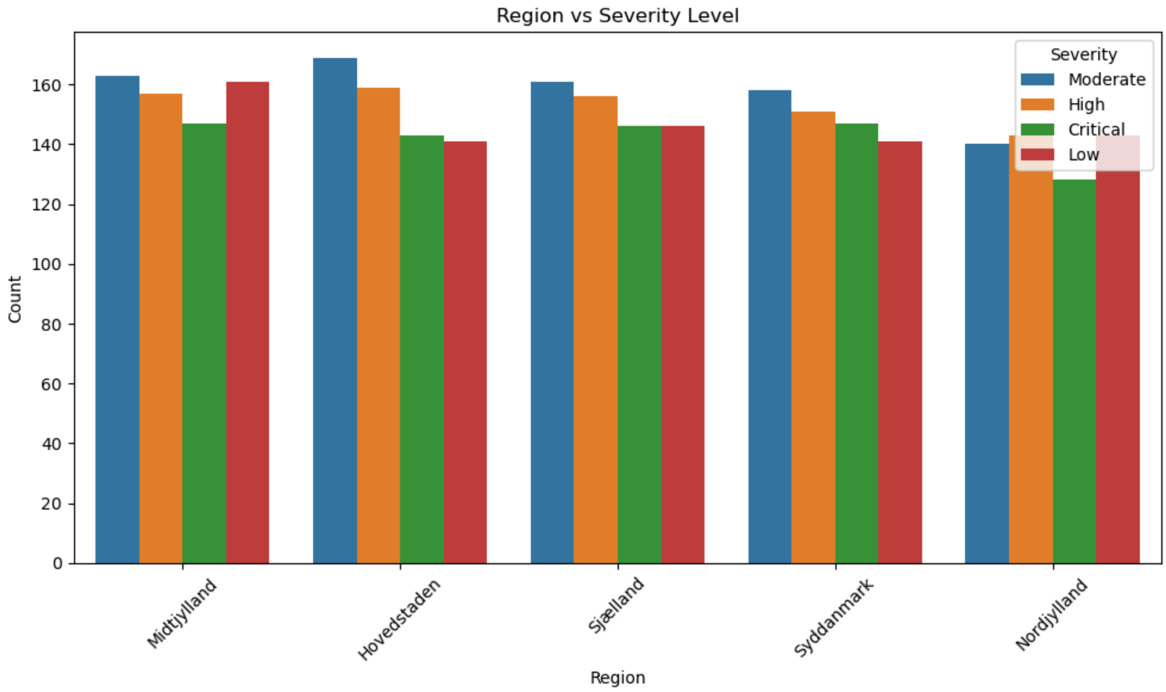


Region vs Severity

In [221…

```
plt.figure(figsize=(10,6))
sns.countplot(data=df, x="Region", hue="Severity", order=df["Region"].value_coun

plt.title("Region vs Severity Level")
plt.xlabel("Region")
plt.ylabel("Count")
plt.xticks(rotation=45)
```

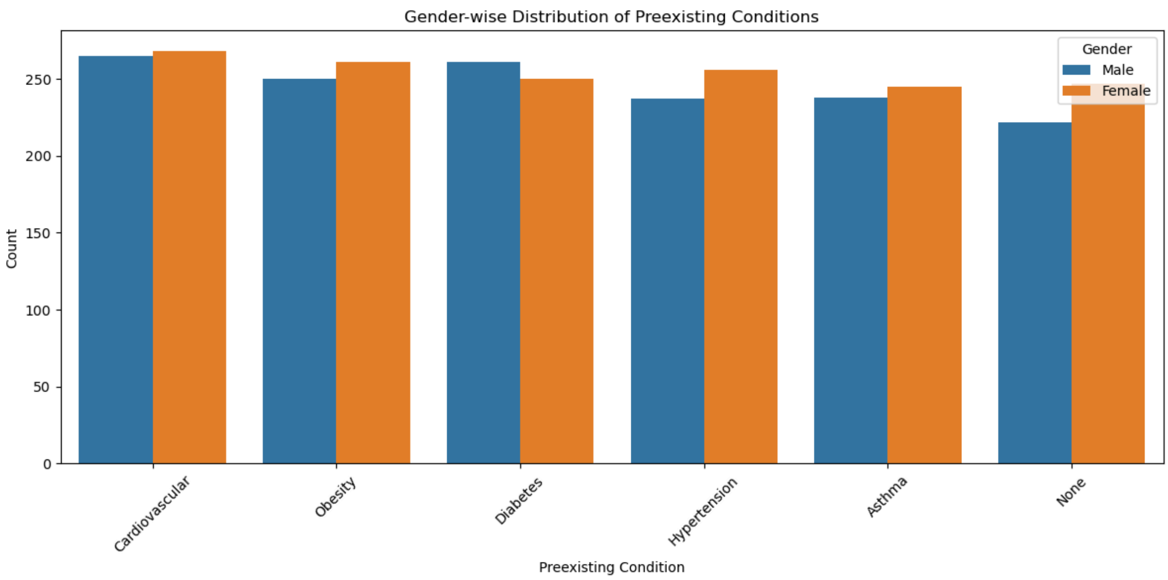
```
plt.legend(title="Severity")
plt.tight_layout()
plt.show()
```



Gender vs Preexisting_Condition

In [222…

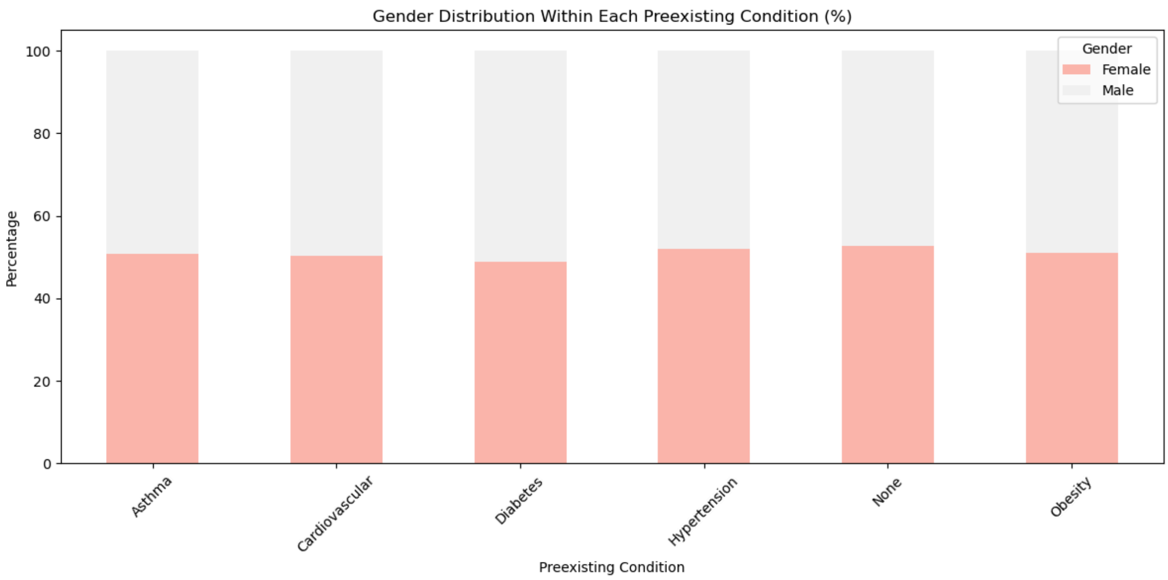
```
plt.figure(figsize=(12,6))
sns.countplot(data=df, x="Preexisting_Condition", hue="Gender", order=df["Preexi

plt.title("Gender-wise Distribution of Preexisting Conditions")
plt.xlabel("Preexisting Condition")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.legend(title="Gender")
plt.tight_layout()
plt.show()
```



Gender vs Preexisting_Condition(%)

In [223…

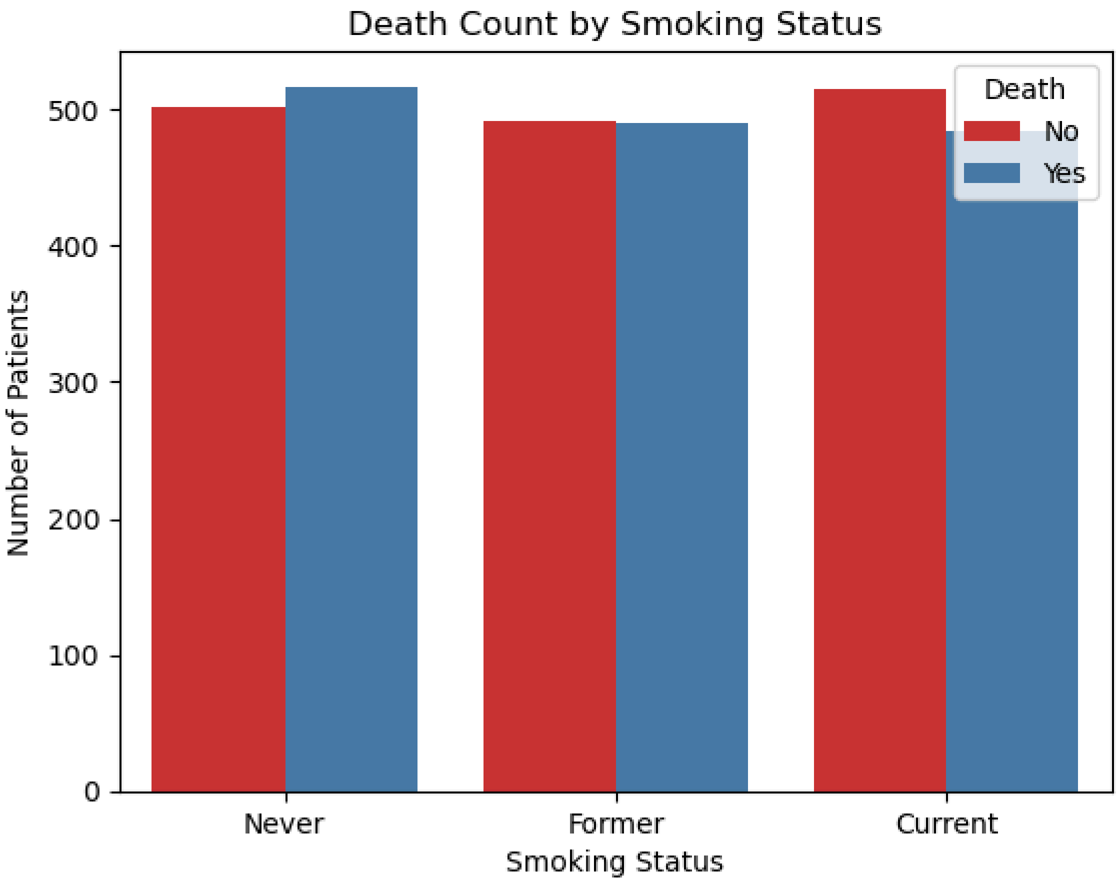
```
# Create a normalized crosstab
gender_condition_pct = (
    pd.crosstab(df["Preexisting_Condition"], df["Gender"], normalize="index") *
).round(1)

# Plot
gender_condition_pct.plot(kind="bar", stacked=True, figsize=(12,6), colormap="Pa
plt.title("Gender Distribution Within Each Preexisting Condition (%)")
plt.xlabel("Preexisting Condition")
plt.ylabel("Percentage")
plt.xticks(rotation=45)
plt.legend(title="Gender")
plt.tight_layout()
plt.show()
```



Count Plot: Smoking_Status vs Death

In [224…

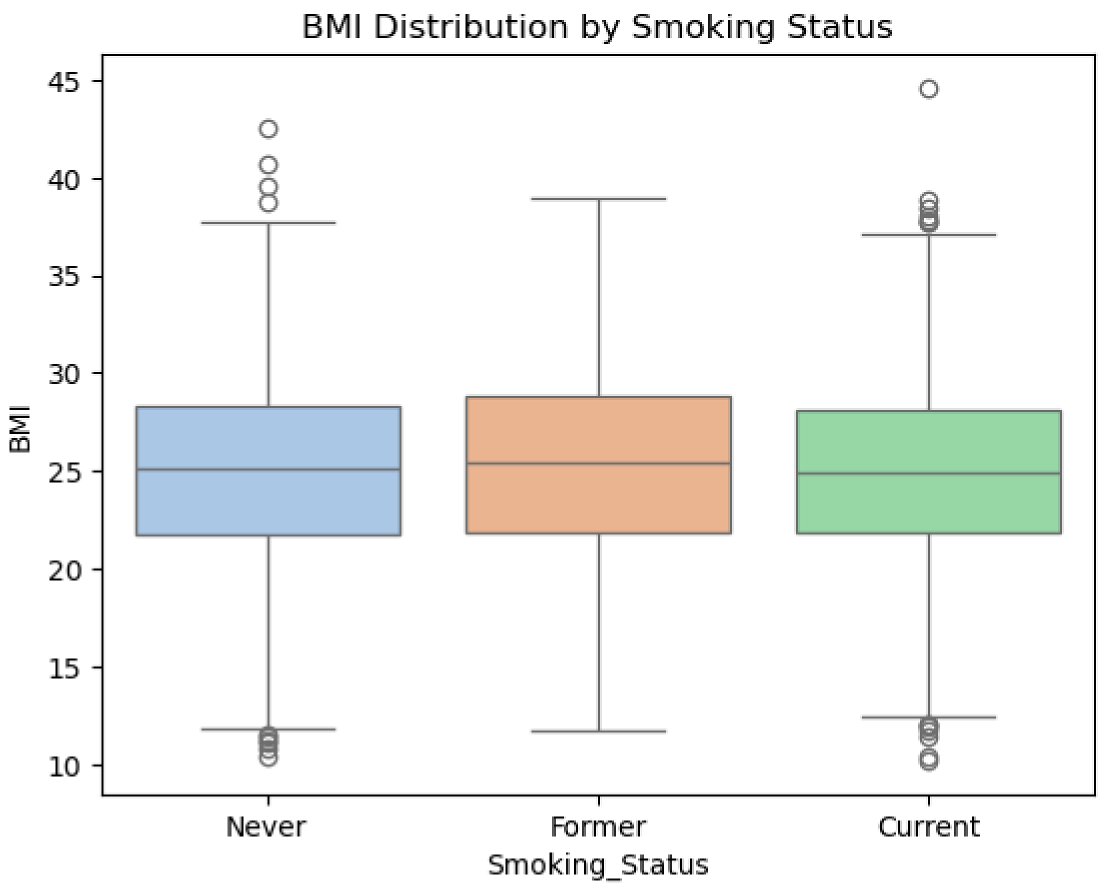
```
sns.countplot(data=df, x='Smoking_Status', hue='Death', palette='Set1')
plt.title('Death Count by Smoking Status')
plt.xlabel('Smoking Status')
plt.ylabel('Number of Patients')
plt.show()
```



Box Plot: Smoking_Status vs Death

In [225…

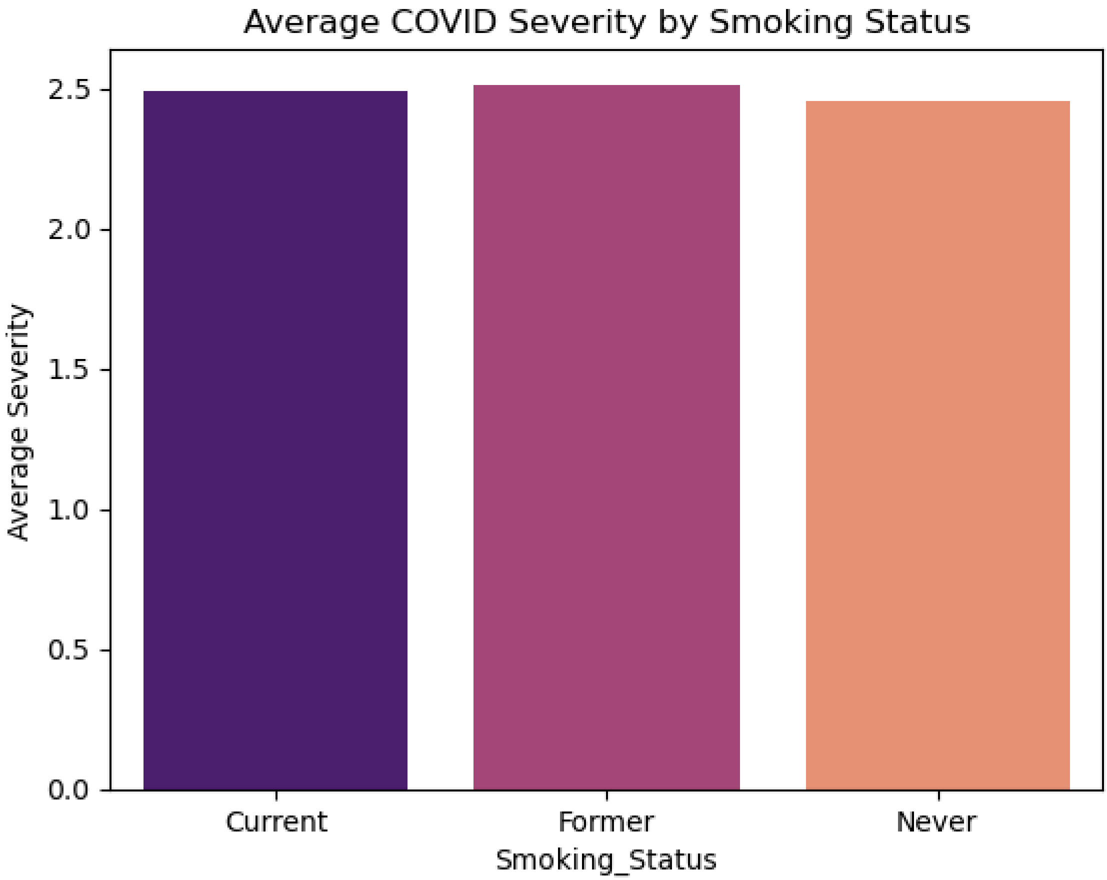
```
sns.boxplot(data=df, x='Smoking_Status', y='BMI', palette='pastel')
plt.title('BMI Distribution by Smoking Status')
plt.show()
```



Box Plot: Average Severity_Num by Smoking_Status

In [226…

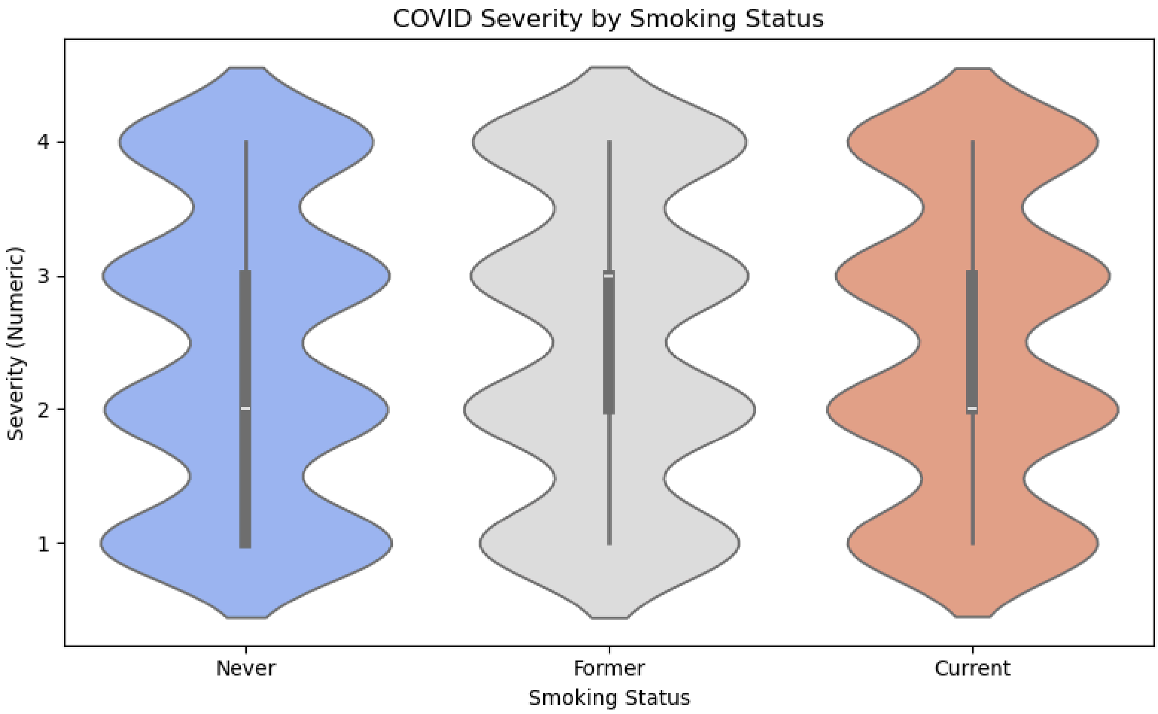
```
severity_map = {'Low': 1, 'Moderate': 2, 'High': 3, 'Critical': 4}
df['Severity_Num'] = df['Severity'].map(severity_map)

avg_severity = df.groupby('Smoking_Status')['Severity_Num'].mean().reset_index()

sns.barplot(data=avg_severity, x='Smoking_Status', y='Severity_Num', palette='ma
plt.title('Average COVID Severity by Smoking Status')
plt.ylabel('Average Severity')
plt.show()
```



Violin Plot: Severity_Num vs Smoking_Status

In [227…

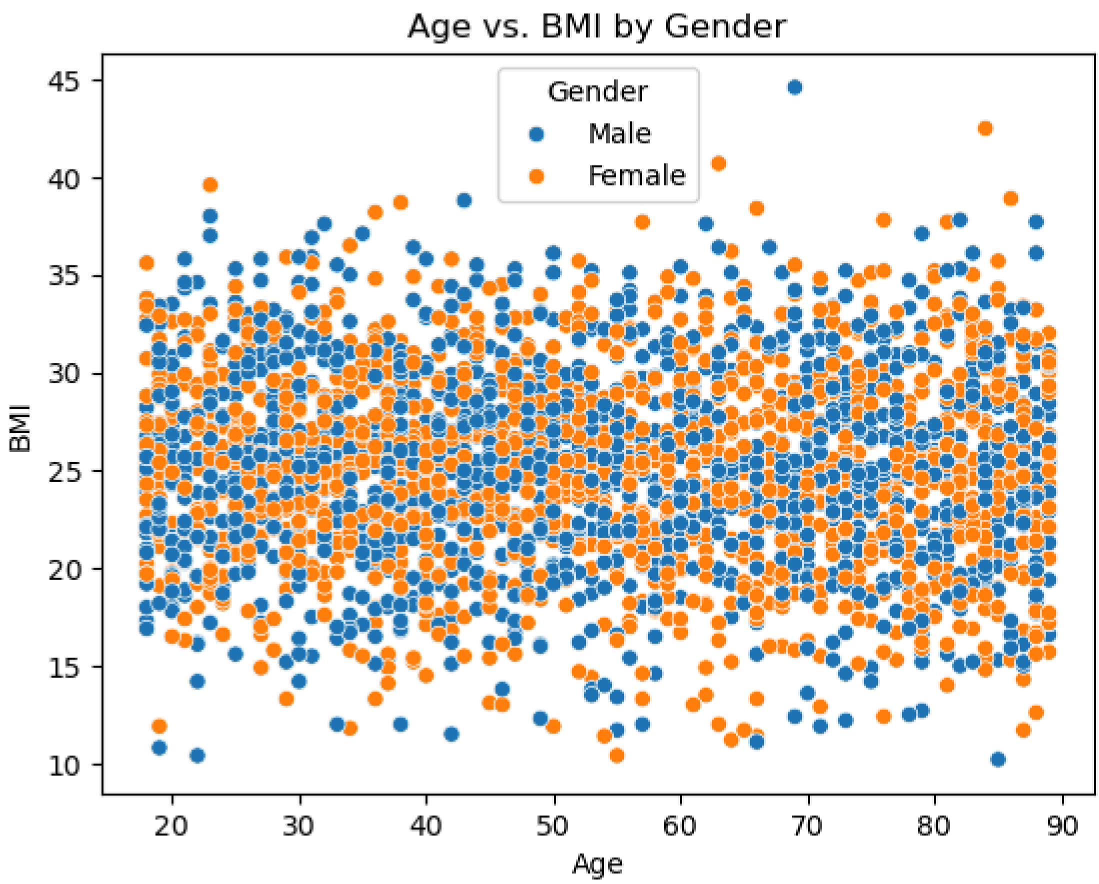
```
plt.figure(figsize=(8, 5))
sns.violinplot(data=df, x='Smoking_Status', y='Severity_Num', palette='coolwarm'
plt.title('COVID Severity by Smoking Status')
plt.xlabel('Smoking Status')
plt.ylabel('Severity (Numeric)')
plt.tight_layout()
plt.show()
```



Age vs. BMI

In [228…

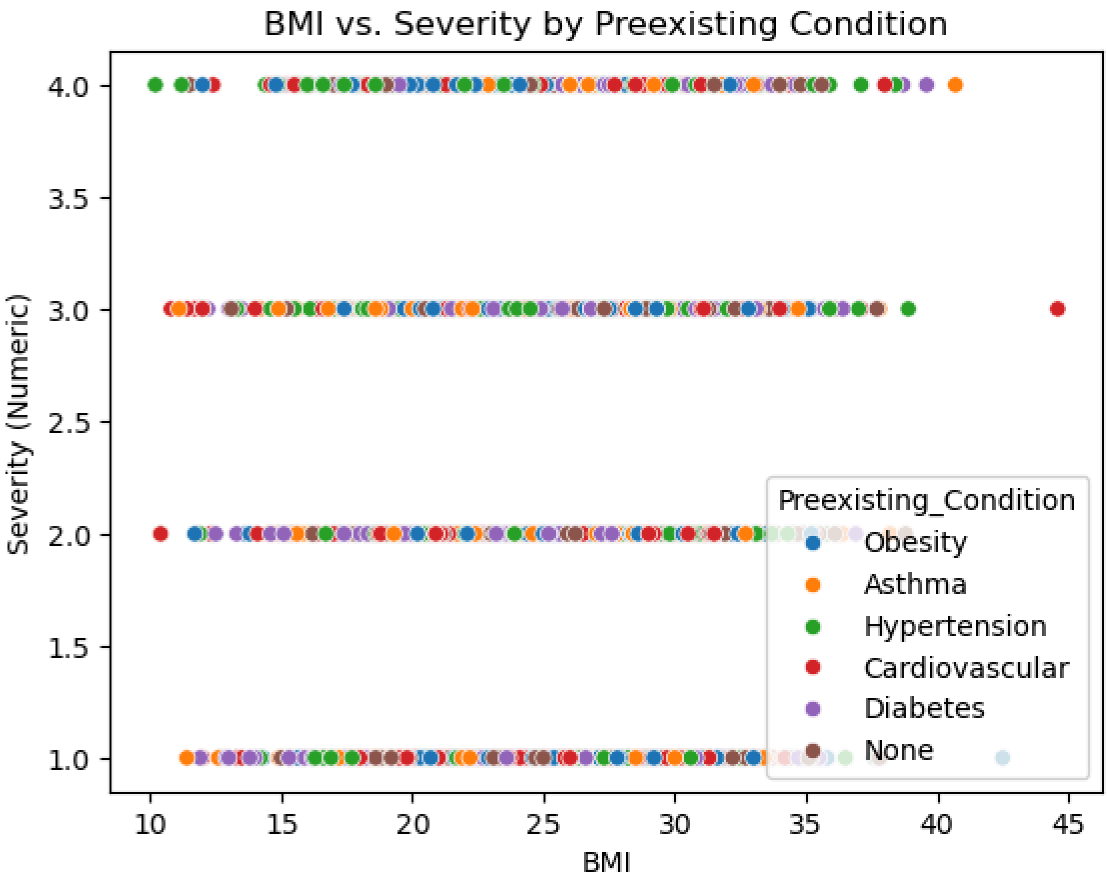
```
sns.scatterplot(data=df, x='Age', y='BMI', hue='Gender')
plt.title('Age vs. BMI by Gender')
plt.show()
```



In [229…

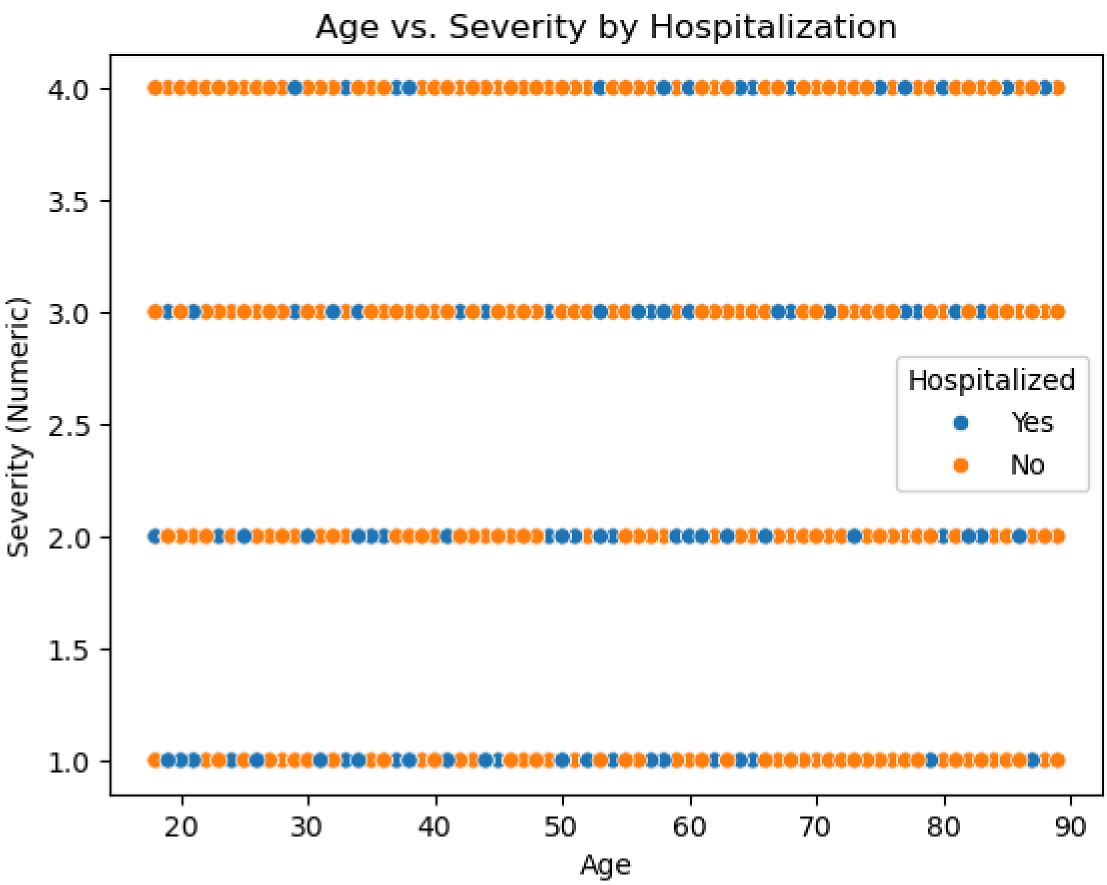
```
sns.scatterplot(data=df, x='BMI', y='Severity_Num', hue='Preexisting_Condition')
plt.title('BMI vs. Severity by Preexisting Condition')
```

```
plt.ylabel('Severity (Numeric)')
plt.show()
```



In [230…

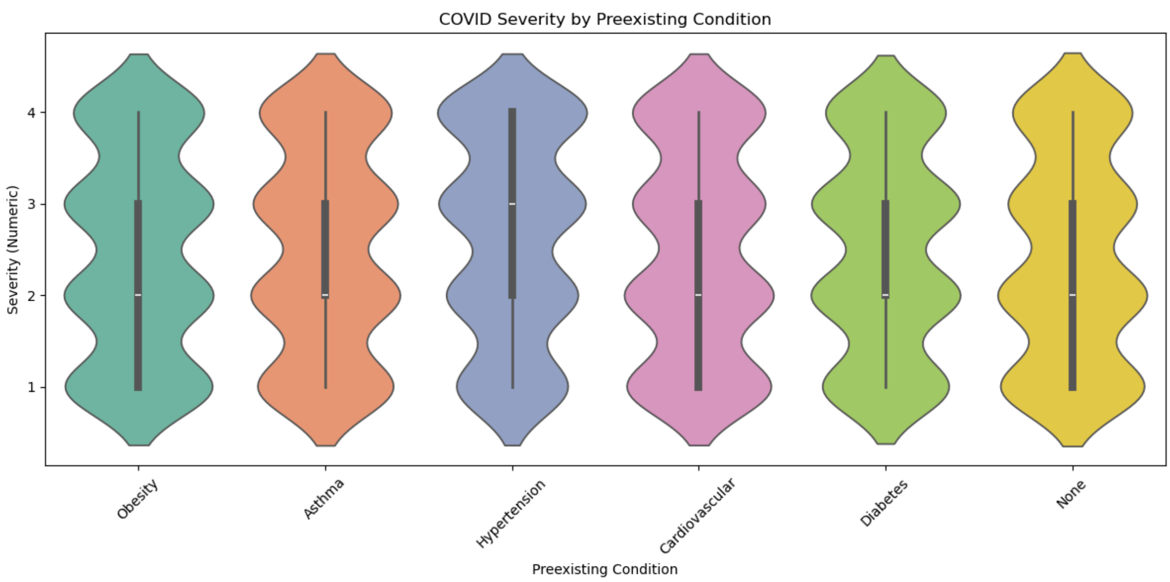
```
sns.scatterplot(data=df, x='Age', y='Severity_Num', hue='Hospitalized')
plt.title('Age vs. Severity by Hospitalization')
plt.ylabel('Severity (Numeric)')
plt.show()
```



Violin Plot: Severity_Num vs Preexisting_Condition

In [231…

```
plt.figure(figsize=(12, 6))
sns.violinplot(data=df, x='Preexisting_Condition', y='Severity_Num', palette='Se
plt.title('COVID Severity by Preexisting Condition')
plt.xlabel('Preexisting Condition')
plt.ylabel('Severity (Numeric)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



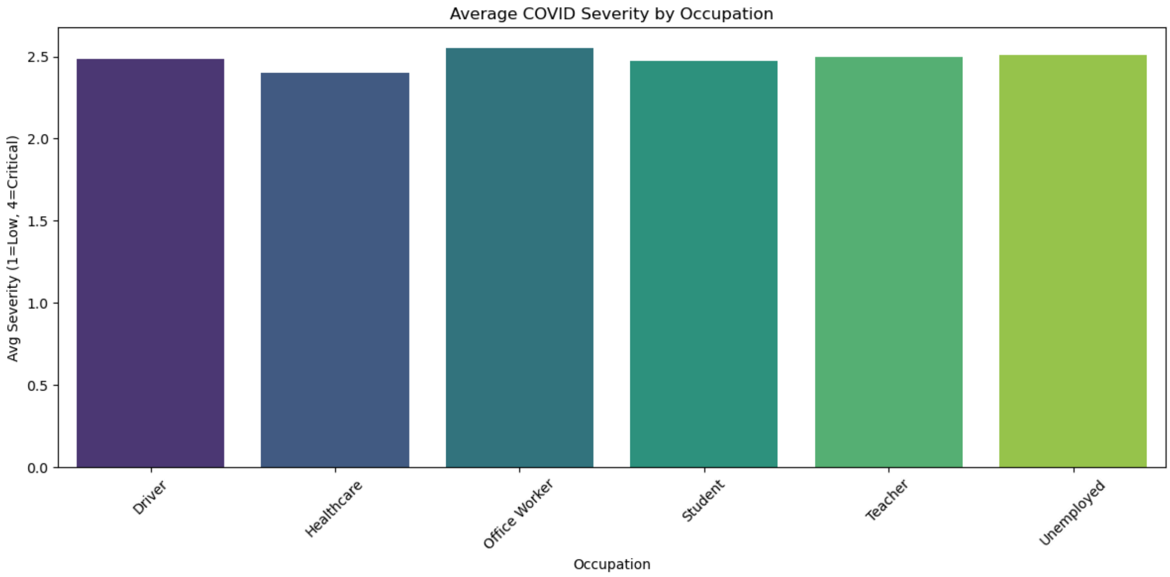
The violin plot shows the distribution of COVID severity scores across different preexisting conditions. Most conditions, including obesity, asthma, hypertension, cardiovascular issues, and diabetes, have a similar median severity level around 2.5–3. However, hypertension and diabetes appear to have a slightly higher range of severity, indicating a tendency toward more severe outcomes. Individuals with no preexisting conditions generally show a narrower severity spread, suggesting milder cases on average. Overall, the plot suggests that while preexisting conditions may impact severity, the variation within each group is still quite broad.

Plots on the basis of Occupation

Bar Plot: Average Severity by Occupation

In [232…

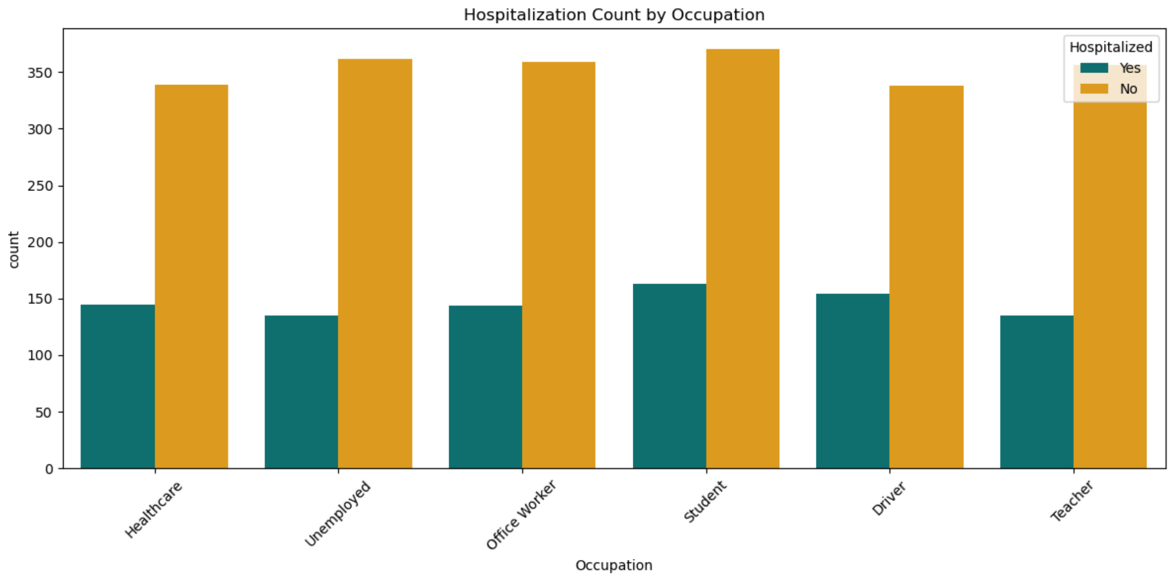
```
avg_severity = df.groupby('Occupation')['Severity_Num'].mean().reset_index()

plt.figure(figsize=(12,6))
sns.barplot(data=avg_severity, x='Occupation', y='Severity_Num', palette='viridi
plt.title('Average COVID Severity by Occupation')
plt.xticks(rotation=45)
plt.ylabel('Avg Severity (1=Low, 4=Critical)')
plt.tight_layout()
plt.show()
```



Count Plot: Hospitalization by Occupation

In [233…

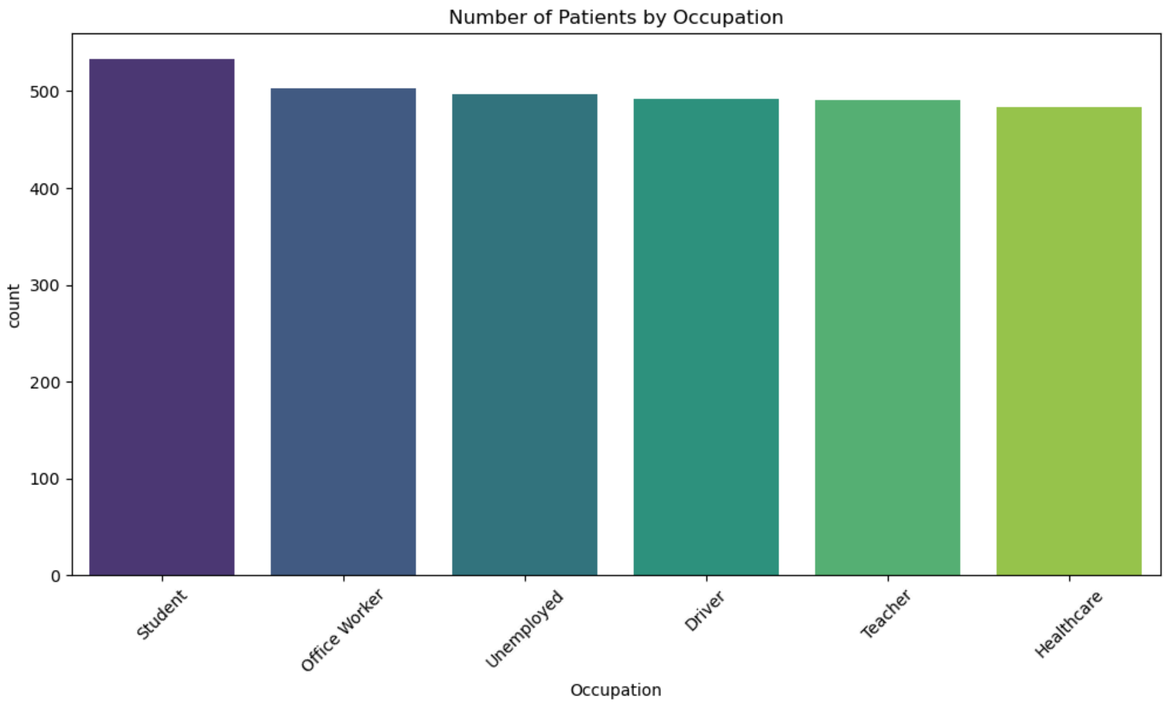
```
plt.figure(figsize=(12,6))
sns.countplot(data=df, x='Occupation', hue='Hospitalized', palette=["#008080", "
plt.title('Hospitalization Count by Occupation')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Count Plot: Number of Patients by Occupation

In [234…

```
plt.figure(figsize=(12,6))
sns.countplot(data=df, x='Occupation', order=df['Occupation'].value_counts().ind
plt.title('Number of Patients by Occupation')
plt.xticks(rotation=45)
plt.show()
```



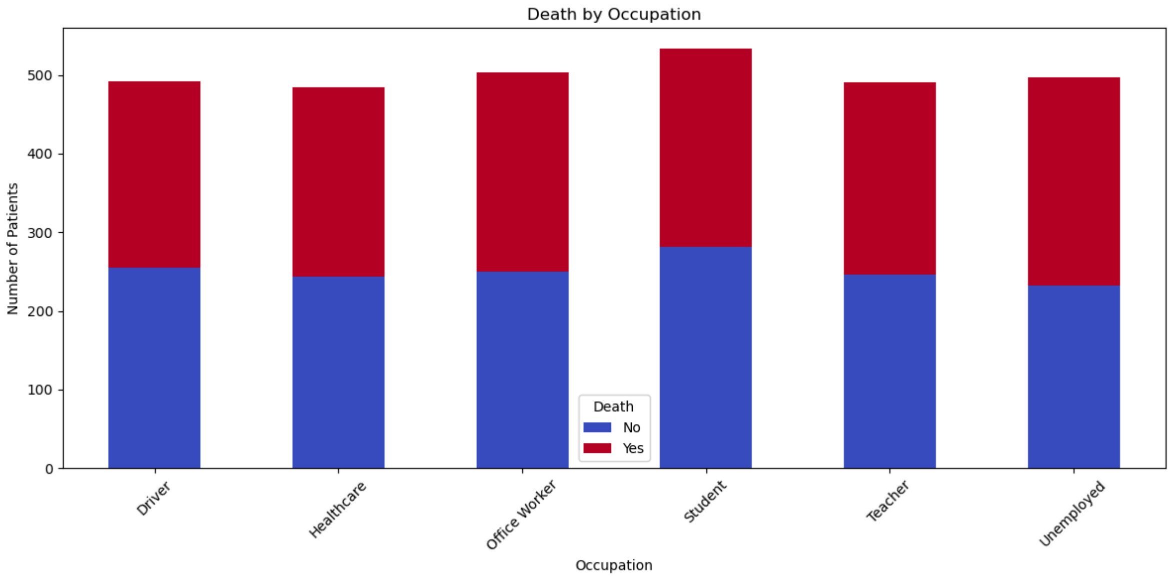
Most patients are from Healthcare, Office Worker, and Student groups.

Very few patients belong to Retired or Teacher categories, indicating different exposure or age distribution.

Stacked Bar Plot: Death Count by Occupation

In [235…

```
death_by_occ = df.groupby(['Occupation', 'Death']).size().unstack().fillna(0)

death_by_occ.plot(kind='bar', stacked=True, figsize=(12,6), colormap='coolwarm')
plt.title('Death by Occupation')
plt.ylabel('Number of Patients')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



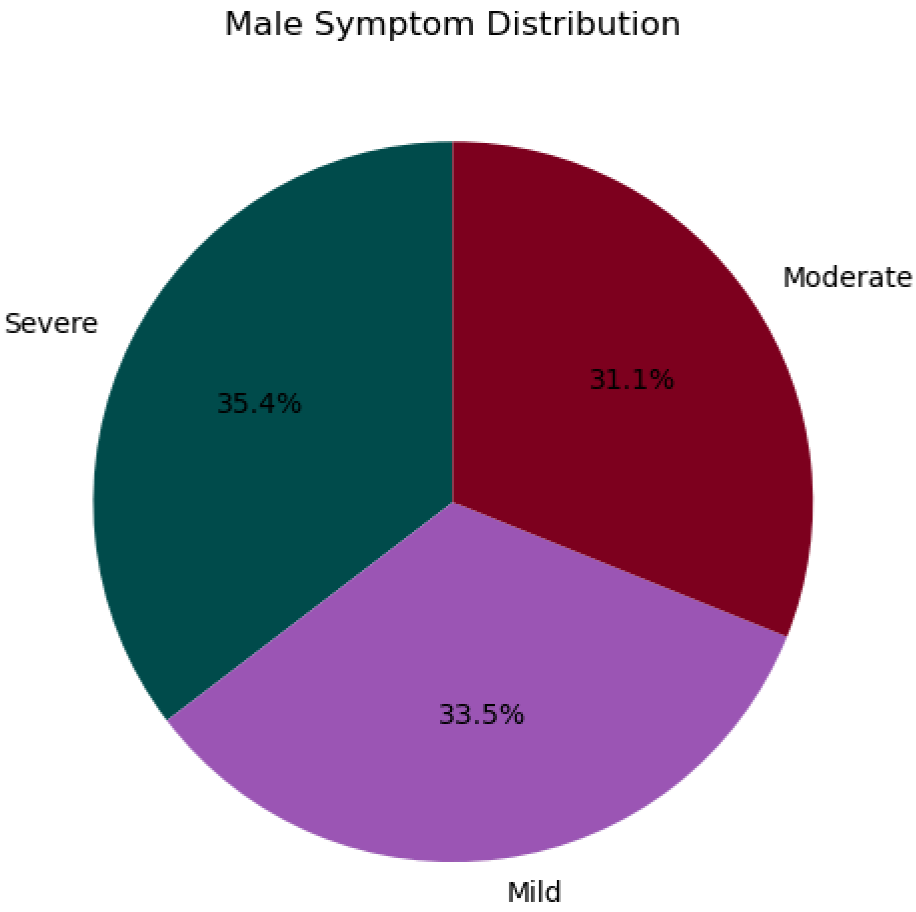
Multivariate Plots

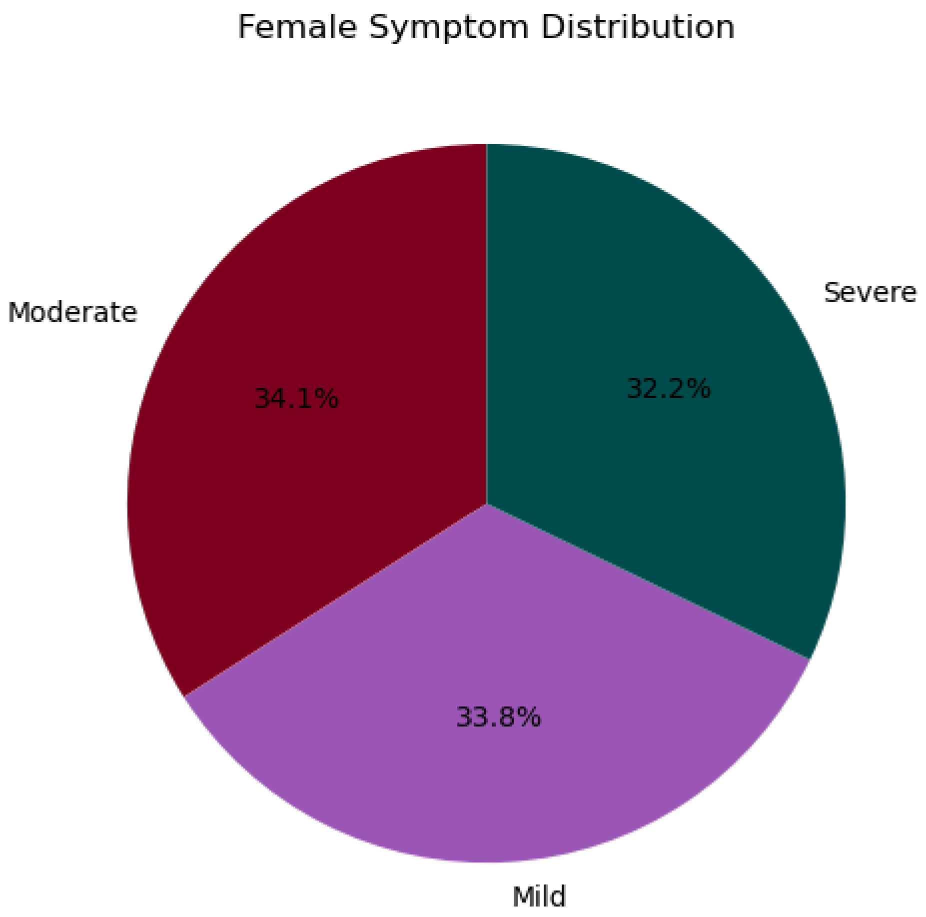
Pie Charts: Gender vs Symptoms

In [236…

```
# Define new unique colors
symptom_colors = {
    "Mild": "#9b59b6",
    "Moderate": "#800020",
    "Severe": "#004d4d"
}

# Male Pie Chart
male_symptoms = df[df["Gender"] == "Male"]["Symptoms"].value_counts()
male_colors = [symptom_colors[s] for s in male_symptoms.index]

plt.figure(figsize=(5,5))
plt.pie(male_symptoms, labels=male_symptoms.index, autopct='%1.1f%%',
        colors=male_colors, startangle=90)
plt.title("Male Symptom Distribution")
plt.tight_layout()
plt.show()

# Female Pie Chart
female_symptoms = df[df["Gender"] == "Female"]["Symptoms"].value_counts()
female_colors = [symptom_colors[s] for s in female_symptoms.index]

plt.figure(figsize=(5,5))
plt.pie(female_symptoms, labels=female_symptoms.index, autopct='%1.1f%%',
        colors=female_colors, startangle=90)
plt.title("Female Symptom Distribution")
```

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

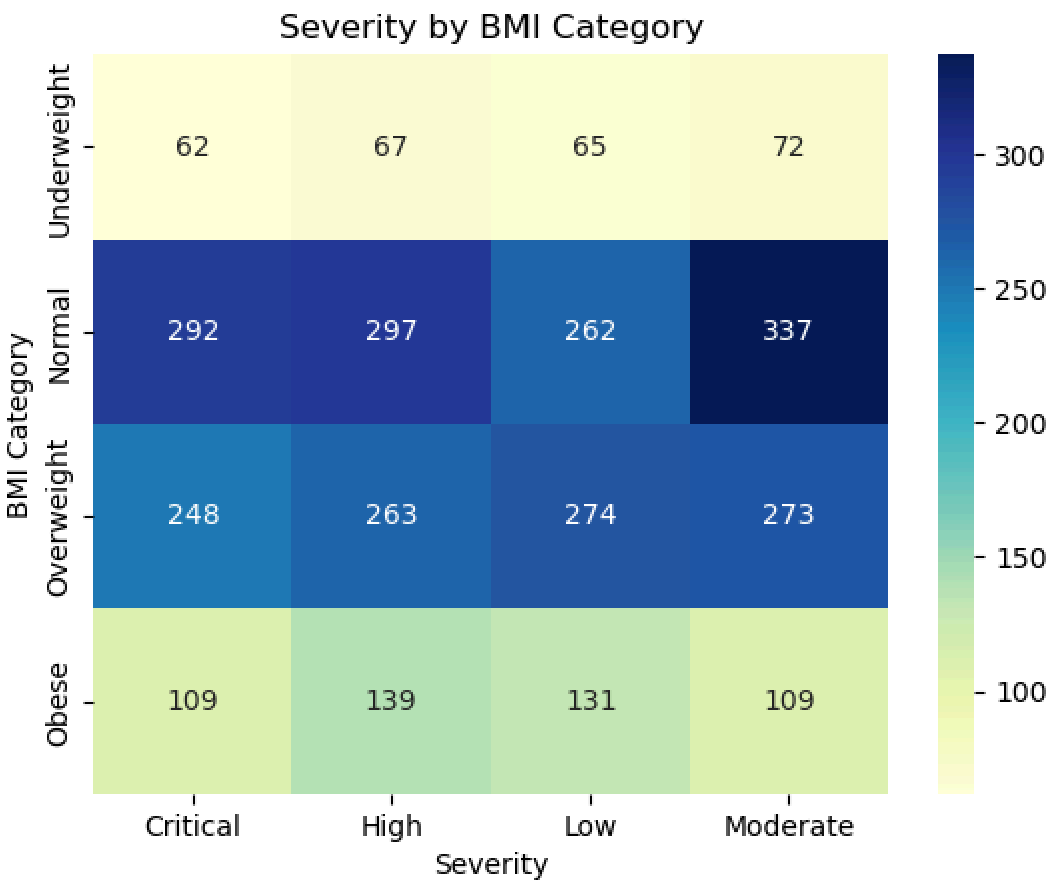




BMI Category vs Severity

In [237…

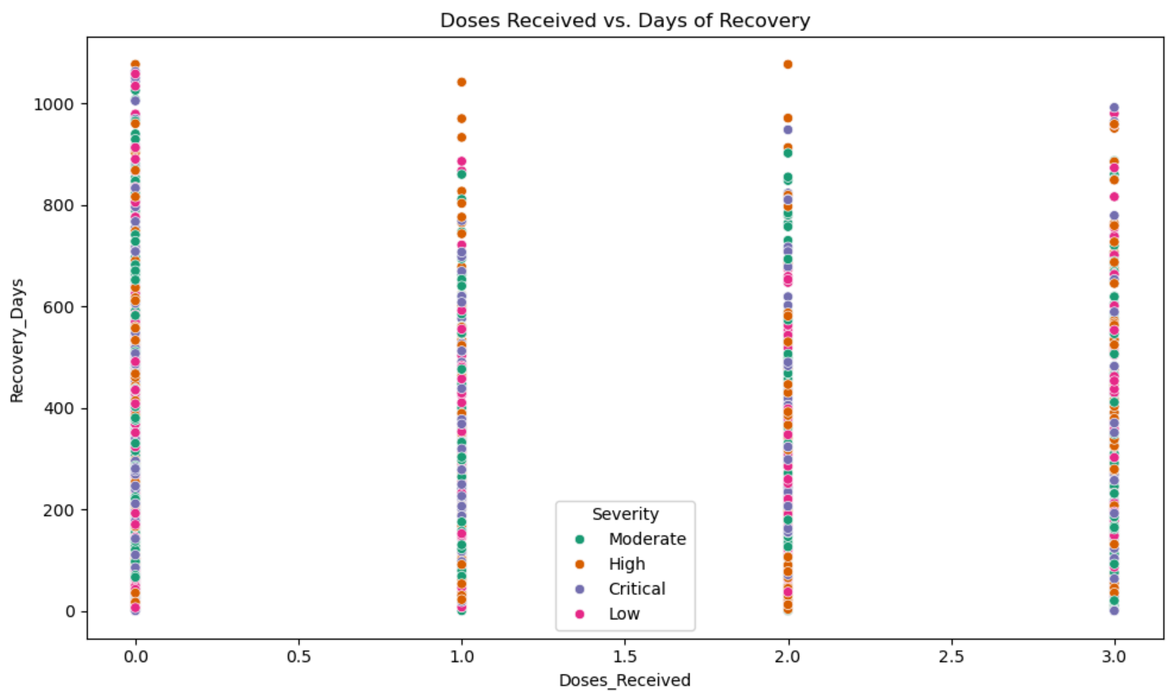
```
crosstab = pd.crosstab(df['BMI_Category'], df['Severity'])
sns.heatmap(crosstab, annot=True, cmap='YlGnBu', fmt='d')
plt.title('Severity by BMI Category')
plt.xlabel('Severity')
plt.ylabel('BMI Category')
plt.show()
```



Is there any relationship between vaccine doses and recovery time?

In [238…

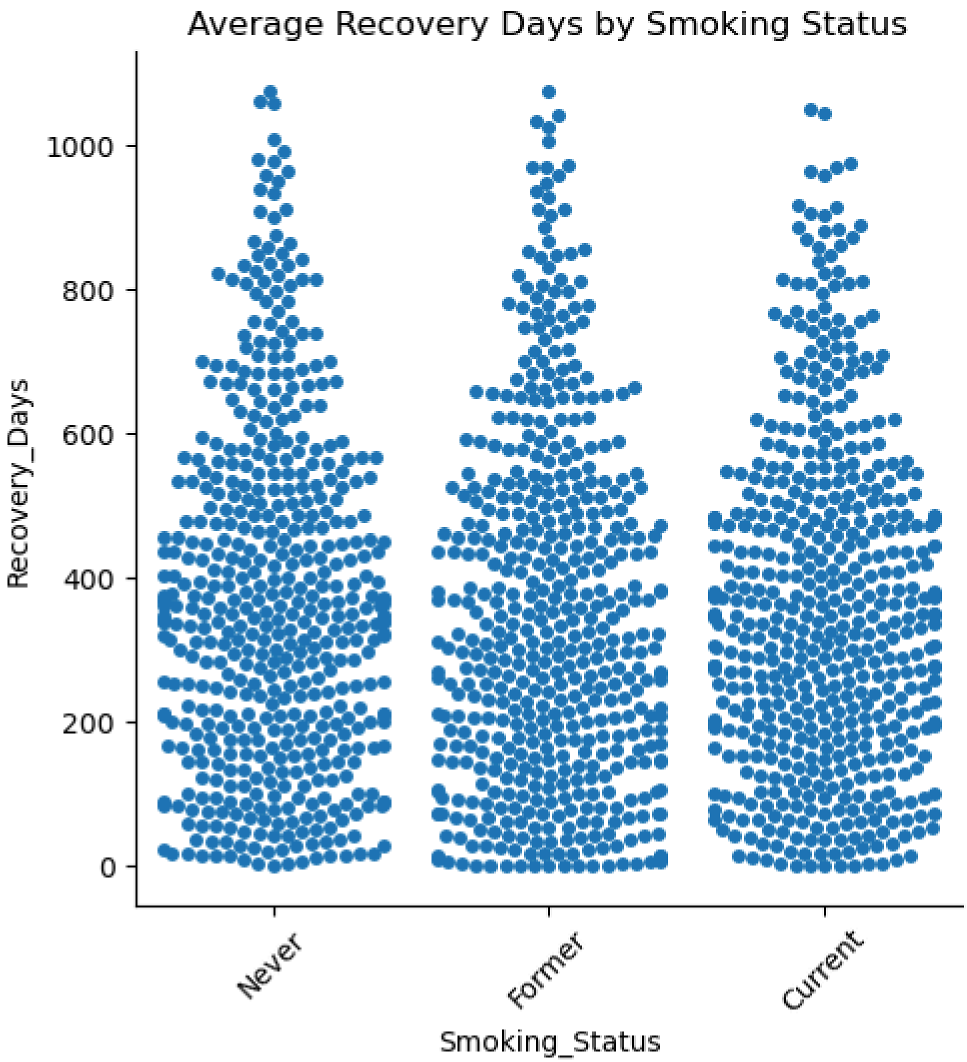
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Doses_Received', y='Recovery_Days', hue='Severity',
plt.title('Doses Received vs. Days of Recovery')
plt.tight_layout()
plt.show()
```



In [239…

```
sns.catplot(x='Smoking_Status', y='Recovery_Days', kind='swarm', data=df)
plt.title("Average Recovery Days by Smoking Status")
```

```
plt.xticks(rotation=45)
plt.show()
```

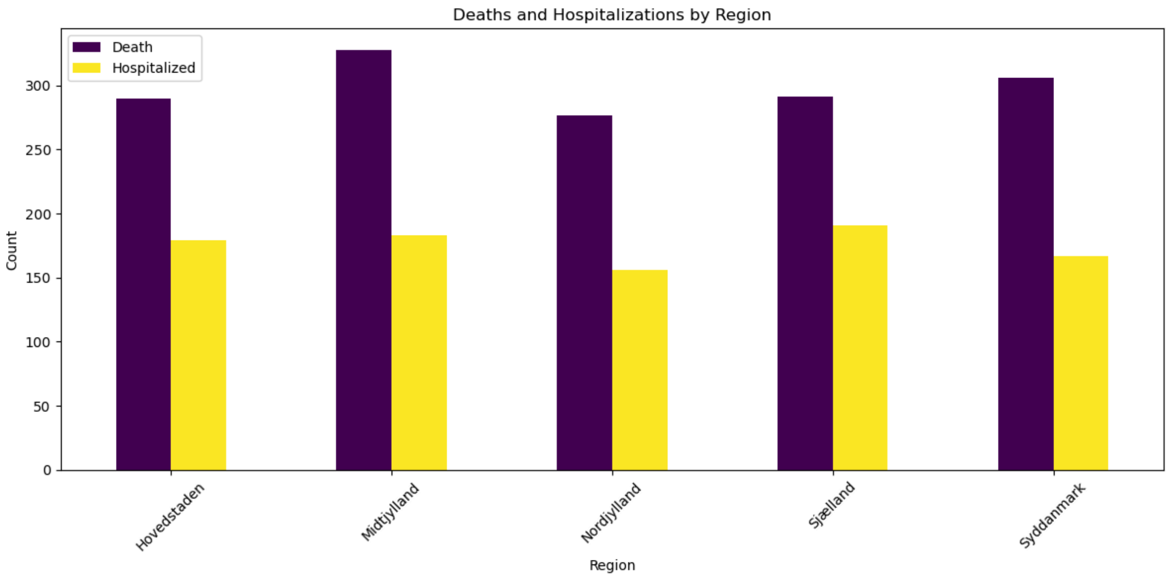


Analysis Questions

1. Which region has the highest number of deaths and hospitalizations?

In [241…

```
region_death_hosp = df.groupby('Region')[['Death', 'Hospitalized']].apply(lambda
region_death_hosp.plot(kind='bar', figsize=(12, 6), colormap='viridis')
plt.title('Deaths and Hospitalizations by Region')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



2. Cretae Binary Tables

In [242…

```
# Create binary column: 1 if Doses_Received is a number, 0 if 'None' or invalid
df['Doses_Received_Binary'] = pd.to_numeric(df['Doses_Received'], errors='coerce

# Create binary column: 1 if Recovery_Days is a number, 0 if 'None' or invalid
df['Recovery_Days_Binary'] = pd.to_numeric(df['Recovery_Days'], errors='coerce')
```

In [243…

```
print(df[['Doses_Received', 'Doses_Received_Binary']].head())
```

```
   Doses_Received  Doses_Received_Binary
0               1                      1
1               0                      1
2               3                      1
3               1                      1
4               2                      1
```

In [247…

```
print(df[['Recovery_Days', 'Recovery_Days_Binary']].head())
```

```
   Recovery_Days  Recovery_Days_Binary
0            302                     1
1              0                     0
2              0                     0
3            546                     1
4              0                     0
```

3. Is there a trend between vaccine doses and recovery days?

In [248…

```
correlation = df[['Doses_Received_Binary', 'Recovery_Days_Binary']].corr().loc['
print(f"Correlation between Doses Received and Days to Recovery: {correlation}")
```

```
Correlation between Doses Received and Days to Recovery: nan
```

4. Which region has the highest hospitalization and death count?

In [249…

```
# Count of 'Yes' in Hospitalized and Death columns per region
region_stats = df.groupby('Region')[['Hospitalized', 'Death']].apply(lambda x: (
print(region_stats.sort_values(by='Death', ascending=False))
```

```
             Hospitalized  Death
Region
Midtjylland           183    328
Syddanmark            167    306
Sjælland              191    291
Hovedstaden           179    290
Nordjylland           156    277
```

5. What occupations are most associated with high severity and death?

In [250…

```
# Mean severity and death count per occupation
severity_by_occ = df.groupby('Occupation')['Severity_Num'].mean().sort_values(as
death_by_occ = df[df['Death'] == 'Yes'].groupby('Occupation').size().sort_values
print("Severity by Occupation:\n", severity_by_occ)
print("\nDeath Count by Occupation:\n", death_by_occ)
```

```
Severity by Occupation:
 Occupation
Office Worker    2.548708
Unemployed       2.507042
Teacher          2.496945
Driver           2.483740
Student          2.472795
Healthcare       2.400826
Name: Severity_Num, dtype: float64

Death Count by Occupation:
 Occupation
Unemployed       265
Office Worker    253
Student          251
Teacher          245
Healthcare       241
Driver           237
dtype: int64
```

6. Are elderly people more at risk of death and hospitalization?

In [251…

```
# Create age group bins
df['Age_Group'] = pd.cut(df['Age'], bins=[0, 18, 40, 60, 80, 100],
                         labels=['0-18', '19-40', '41-60', '61-80', '81+'])

risk_by_age = df.groupby('Age_Group')[['Hospitalized', 'Death']].apply(lambda x:
print(risk_by_age)
```

```
           Hospitalized  Death
Age_Group
0-18                 17     19
19-40               255    460
41-60               246    408
61-80               250    429
81+                 108    176
```

7. Which type of patients took the longest time to recover?

In [252…

```
long_recovery = df.groupby('Preexisting_Condition')['Recovery_Days'].mean().roun
print(long_recovery)
```

```
Preexisting_Condition
Diabetes          195
Asthma            190
Cardiovascular    185
Obesity           182
Hypertension      173
None              168
Name: Recovery_Days, dtype: int32
```

8. Which patients with preexisting conditions experienced the highest severity?

In [253…

```
severity_condition = df.groupby('Preexisting_Condition')['Severity_Num'].mean().
print(severity_condition)
```

```
Preexisting_Condition
Hypertension      3
Asthma            2
Cardiovascular    2
Diabetes          2
None              2
Obesity           2
Name: Severity_Num, dtype: int32
```

9. Which region had the highest number of hospitalizations (possibly indicating lower healthcare access)?

In [254…

```
hosp_by_region = df.groupby('Region')['Hospitalized'].apply(lambda x: (x == 'Yes
print(hosp_by_region)
```

```
Region
Sjælland       191
Midtjylland    183
Hovedstaden    179
Syddanmark     167
Nordjylland    156
Name: Hospitalized, dtype: int64
```

10. Which age group had the longest recovery time?

In [255…

```
df['Age_Group'] = pd.cut(df['Age'], bins=[0,18,40,60,80,100], labels=['0-18','19
recovery_by_age = df.groupby('Age_Group')['Recovery_Days'].mean().round().astype
print(recovery_by_age)
```

```
Age_Group
0-18     240
81+      205
19-40    187
61-80    176
41-60    169
Name: Recovery_Days, dtype: int32
```

11. Did patients with fewer vaccine doses have a longer recovery time?

In [256…

```
# Group by doses and calculate mean recovery time
recovery_by_dose = df.groupby('Doses_Received')['Recovery_Days'].mean().round().
print("Average Recovery Time by Number of Doses:\n")
print(recovery_by_dose)
```

```
Average Recovery Time by Number of Doses:

Doses_Received
0    190
1    172
2    161
3    188
Name: Recovery_Days, dtype: int32
```

In [257…

```
correlation = df[['Doses_Received_Binary', 'Recovery_Days_Binary']].corr().loc['
print("Correlation between Doses Received and  Recovery Days :", correlation)
```

```
Correlation between Doses Received and  Recovery Days : nan
```

12. Which gender had a higher number of deaths or severe cases?

In [258…

```
gender_severity = df.groupby('Gender')['Severity_Num'].mean()
death_gender = df.groupby('Gender')['Death'].apply(lambda x: (x == 'Yes').sum())
print("Severity:\n", gender_severity)
print("\nDeath Count:\n", death_gender)
```

```
Severity:
 Gender
Female    2.491814
Male      2.478615
Name: Severity_Num, dtype: float64

Death Count:
 Gender
Female    784
Male      708
Name: Death, dtype: int64
```

13. Which occupation group had a higher severity or death rate?

In [259…

```
occ_severity = df.groupby('Occupation')['Severity_Num'].mean().sort_values(ascen
occ_death = df[df['Death'] == 'Yes'].groupby('Occupation').size().sort_values(as
print("Severity by Occupation:\n", occ_severity)
print("\nDeath Count by Occupation:\n", occ_death)
```

```
Severity by Occupation:
 Occupation
Office Worker    2.548708
Unemployed       2.507042
Teacher          2.496945
Driver           2.483740
Student          2.472795
Healthcare       2.400826
Name: Severity_Num, dtype: float64

Death Count by Occupation:
 Occupation
Unemployed       265
Office Worker    253
Student          251
Teacher          245
Healthcare       241
Driver           237
dtype: int64
```

14. Does COVID severity tend to be higher among smokers?

In [260…

```
severity_by_smoke = df.groupby('Smoking_Status')['Severity_Num'].mean().sort_val
print(severity_by_smoke)
```

```
Smoking_Status
Former     2.513761
Current    2.492000
Never      2.451423
Name: Severity_Num, dtype: float64
```

15. Which smoking group had the longest recovery time?

In [261…

```
#  check average recovery time per smoking group
recovery_by_smoke = df.groupby('Smoking_Status')['Recovery_Days'].mean().round(0
print("Average Recovery Days by Smoking Status (rounded):\n", recovery_by_smoke)
```

```
Average Recovery Days by Smoking Status (rounded):
 Smoking_Status
Current    186
Never      184
Former     176
Name: Recovery_Days, dtype: int32
```

16. Which BMI group had the longest recovery time?

In [262…

```
# Use proper recovery days column
recovery_by_bmi = df.groupby('BMI_Category')['Recovery_Days'].mean().round(0).as
print("Average Recovery Days by BMI Category:\n", recovery_by_bmi)
```

```
Average Recovery Days by BMI Category:
 BMI_Category
Underweight    193
Normal         183
Overweight     181
Obese          177
Name: Recovery_Days, dtype: int32
```

17. Does the combination of smoking and obesity increase the risk of death?

In [263…

```
combo_death = df.groupby(['Smoking_Status', 'BMI_Category'])['Death'].apply(lamb
print(combo_death.sort_values(ascending=False))
```

```
Smoking_Status  BMI_Category
Never           Normal          202
Current         Normal          196
Never           Overweight      186
Former          Normal          177
                Overweight      175
Current         Overweight      164
Former          Obese            92
Current         Obese            84
Never           Obese            76
                Underweight      53
Former          Underweight      46
Current         Underweight      41
Name: Death, dtype: int64
```

18. Which combination of smoking and preexisting condition is associated with the highest severity?

In [264…
```
combo_severity = df.groupby(['Smoking_Status', 'Preexisting_Condition'])['Severi
print(combo_severity.head(10))
```

```
Smoking_Status  Preexisting_Condition
Current         Asthma                  3
Former          Cardiovascular          3
Never           Hypertension            3
Former          Obesity                 3
                Diabetes                3
                Hypertension            3
Current         Hypertension            3
                Diabetes                3
Former          Asthma                  2
Current         Obesity                 2
Name: Severity_Num, dtype: int32
```

19.Is there any relationship between BMI and vaccine doses (e.g., do individuals with lower BMI tend to receive fewer doses)?

In [265…
```
bmi_dose_corr = df[['BMI', 'Doses_Received']].corr().loc['BMI', 'Doses_Received'
print("Correlation between BMI and Vaccine Doses:", bmi_dose_corr)
```

```
Correlation between BMI and Vaccine Doses: -0.015218010745538513
```

20.Is the death rate higher among smokers despite receiving vaccine doses?

In [266…
```
smoke_death = df.groupby('Smoking_Status')['Death'].apply(lambda x: (x == 'Yes')
print(smoke_death)
```

```
Smoking_Status
Current    485
Former     490
Never      517
Name: Death, dtype: int64
```

21.Even though obese individuals received more vaccine doses, how was their recovery time?

In [270…
```
obese_data = df[df['BMI_Category'].isin(['Obese', 'Severely Obese'])]
obese_recovery = obese_data.groupby('Doses_Received')['Recovery_Days'].mean().rc
print(obese_recovery)
```

```
Doses_Received
0    181.0
1    161.0
2    172.0
3    182.0
Name: Recovery_Days, dtype: float64
```

Vaccine Doses Based Data Analysis

1. Does taking more vaccine doses reduce the severity of COVID?

Analysis: Check average severity score (Severity_Num) grouped by Doses_Received

In [275…
```
df.groupby('Doses_Received')['Severity_Num'].mean().round()
```

Out[275…
```
Doses_Received
0    2.0
1    3.0
2    2.0
3    3.0
Name: Severity_Num, dtype: float64
```

2. Do more vaccine doses reduce the recovery time?

Analysis: Compare Recovery_Days across vaccine dose groups.

In [277…
```
df.groupby('Doses_Received')['Recovery_Days'].mean().round()
```

Out[277…
```
Doses_Received
0    190.0
1    172.0
2    161.0
3    188.0
Name: Recovery_Days, dtype: float64
```

3. What is the death rate for each vaccine dose group?

Analysis: Check percentage of 'Yes' in Death column for each Doses_Received.

In [284…
```
death_by_dose = df.groupby('Doses_Received')['Death'].value_counts(normalize=Tru
print(death_by_dose)
```

```
Doses_Received
0    48.625654
1    50.403226
2    52.192067
3    50.100604
Name: Yes, dtype: float64
```

4. How many patients who received 3 doses were hospitalized?

Analysis: Count Hospitalized status among people with 3 doses.

In [285…
```
df[(df['Doses_Received'] == 3) & (df['Hospitalized'] == 'Yes')].shape[0]
```

Out[285…
```
143
```

5. Which age group received the most vaccine doses (>=2)?

Analysis: Group by Age_Group and count how many received ≥2 doses.

In [286…
```
df[df['Doses_Received'] >= 2].groupby('Age_Group').size().sort_values(ascending=
```

Out[286…
```
Age_Group
61-80    292
19-40    285
41-60    258
81+      124
0-18      17
dtype: int64
```

6. Is there a relationship between vaccine dose count and preexisting conditions?

Analysis: Cross-tabulate Preexisting_Condition with Doses_Received.

In [287…
```
pd.crosstab(df['Preexisting_Condition'], df['Doses_Received'])
```

Out[287…

Doses_Received	0	1	2	3
Preexisting_Condition				
Asthma	244	70	90	79
Cardiovascular	276	80	80	97
Diabetes	259	82	70	100
Hypertension	252	86	83	72
None	236	84	80	69
Obesity	261	94	76	80

7. Among vaccinated people (≥1 dose), which preexisting condition had the most deaths?

Analysis: Focus only on vaccinated people and see deaths by condition.

In [288…
```
df_vac = df[df['Doses_Received'] >= 1]
df_vac[df_vac['Death'] == 'Yes']['Preexisting_Condition'].value_counts()
```

Out[288…
```
Preexisting_Condition
Obesity           130
Diabetes          127
Cardiovascular    126
None              126
Hypertension      125
Asthma            115
Name: count, dtype: int64
```

8. Is there any effect of vaccine doses on symptoms?

Analysis: Average count of symptoms per dose group.

In [289…
```
df['Symptom_Count'] = df['Symptoms'].fillna('').apply(lambda x: len(x.split(','))
df.groupby('Doses_Received')['Symptom_Count'].mean()
```

Out[289…

```
Doses_Received
0    1.0
1    1.0
2    1.0
3    1.0
Name: Symptom_Count, dtype: float64
```

9. Among patients with 0 vaccine doses, what’s the most common preexisting condition?

Analysis: Find most common health issue in unvaccinated group.

In [290…

```
df[df['Doses_Received'] == 0]['Preexisting_Condition'].value_counts()
```

Out[290…

```
Preexisting_Condition
Cardiovascular    276
Obesity           261
Diabetes          259
Hypertension      252
Asthma            244
None              236
Name: count, dtype: int64
```

10. Are vaccinated people less likely to be hospitalized?

Analysis: Compare hospitalization rate among 0, 1, 2, 3 dose groups.

In [292…

```
df.groupby('Doses_Received')['Hospitalized'].value_counts(normalize=True).unstac
```

Out[292…

```
Doses_Received
0    28.795812
1    29.435484
2    30.688935
3    28.772636
Name: Yes, dtype: float64
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: