**Aim:**

To gain insights and make sense of the data, identifying patterns, trends, and relationships within the dataset.

**Data Collection Methods:**

Data collection methods in data analysis refer to the ways in which data is gathered and acquired for subsequent analysis.

**Exploratory data analysis and initial insights. EDA**

Exploratory Data Analysis (EDA) is a crucial step in any data analysis project. It helps you understand the dataset, discover patterns, spot anomalies, and formulate hypotheses. Here’s how you can approach EDA and derive initial insights:

**Data Cleaning:**

**Understand Data frame :**

**Data frame consist 3 column of Health care details with multiple data categories.**

**Need to analysis and make it proper for inferences.**

**Step 1:**

Find Unique values of category column and split it into an multiple worksheets.

**Snippet:**

with pd.ExcelWriter("NewDf.xlsx") as writer:

gender.to\_excel(writer, sheet\_name="Gender", index=False)

**Step2:**

Read one by one worksheets and map the values for column and drop the extra column

**Snippet:**

df1 = pd.read\_excel("NewDf.xlsx", sheet\_name="Hospital")

df1.rename(columns={'value': 'Hospital'}, inplace=True)

df1 = df1.drop('variable', axis=1)

**Step3:**

Merge all the worksheets as one single sheet .

**Snippet:**

combined\_df = pd.concat([df15,df14, df13,df12,df11,df10,df9,df8,df7,df6,df5,df4,df3,df2,df1], ignore\_index=True)

print(combined\_df)

**Step4:**

Extract the file to new frame

**Snippet:**

ready\_to\_clean = combined\_df.to\_csv('ready\_to\_clean.csv', index=False)

**Step5:**

Read a new file and merge all sheets to one frame using Join function

**Snippet:**

**Inner join function:**

inner\_join = pd.merge(df14, df13, on ='Name', how ='inner')

inner\_join1 = pd.merge(df12, df11, on ='Name', how ='inner')

**Merge each join variables as one sheet :**

m1 = pd.merge(left=inner\_join, right=inner\_join1, how='left'

**Step6:**

Remove Duplicates in Frame

**Snippet :**

duplicates = m5.drop\_duplicates()

**Step7:**

Extract the cleaned file for analysis

**Snippet:**

newDf1 = df\_drop.to\_csv("healthcare\_dataset1.csv", index=False)

**Step8:**

Create a new folder to store all the working files .

Read all csv and excel files in this folder using glob library

**Snippet:**

import glob

# getting csv files from the folder

path = "C:/Users/Nilla/DFproject"

# read all the files with extension .csv

csv = glob.glob(path + "\\*.csv")

**Step9:**

To find the length of excel and csv file

**Snippet:**

import os

print(len([file for file in os.listdir('C:/Users/Nilla/DFproject') if os.path.isfile(os.path.join('C:/Users/Nilla/DFproject',file))]))

**Step10:**

Now Data frame is ready to analysis.

**Libraries used:**

libraries (also known as packages or modules) are collections of functions and methods that extend the capabilities of the core Python language. These libraries cover a wide range of functionalities, from numerical computations to web development and data analysis.

1. **NumPy (Numerical Python)**

Provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

1. **Pandas**

Offers data structures and data analysis tools for handling structured data, primarily in the form of DataFrame objects.

1. **Matplotlib**

A 2D plotting library which produces publication-quality figures in various formats and interactive environments.

1. **Seaborn**

Built on top of Matplotlib, Seaborn provides a high-level interface for drawing attractive and informative statistical graphics.

1. **Plotly.express**

Plotly Express provides a simple syntax for creating a wide range of plots such as scatter plots, line plots, bar charts, box plots, histograms, and more..

1. **Plotly.graph\_objects**

Allows detailed customization of plot elements such as traces, layouts, annotations, axes, and more.

1. **Warnings**

You can temporarily ignore warnings using the warnings.filterwarnings() function. This is useful when you want to suppress warnings in specific parts of your code.

Snippet:

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

import plotly.graph\_objects as goimport warnings

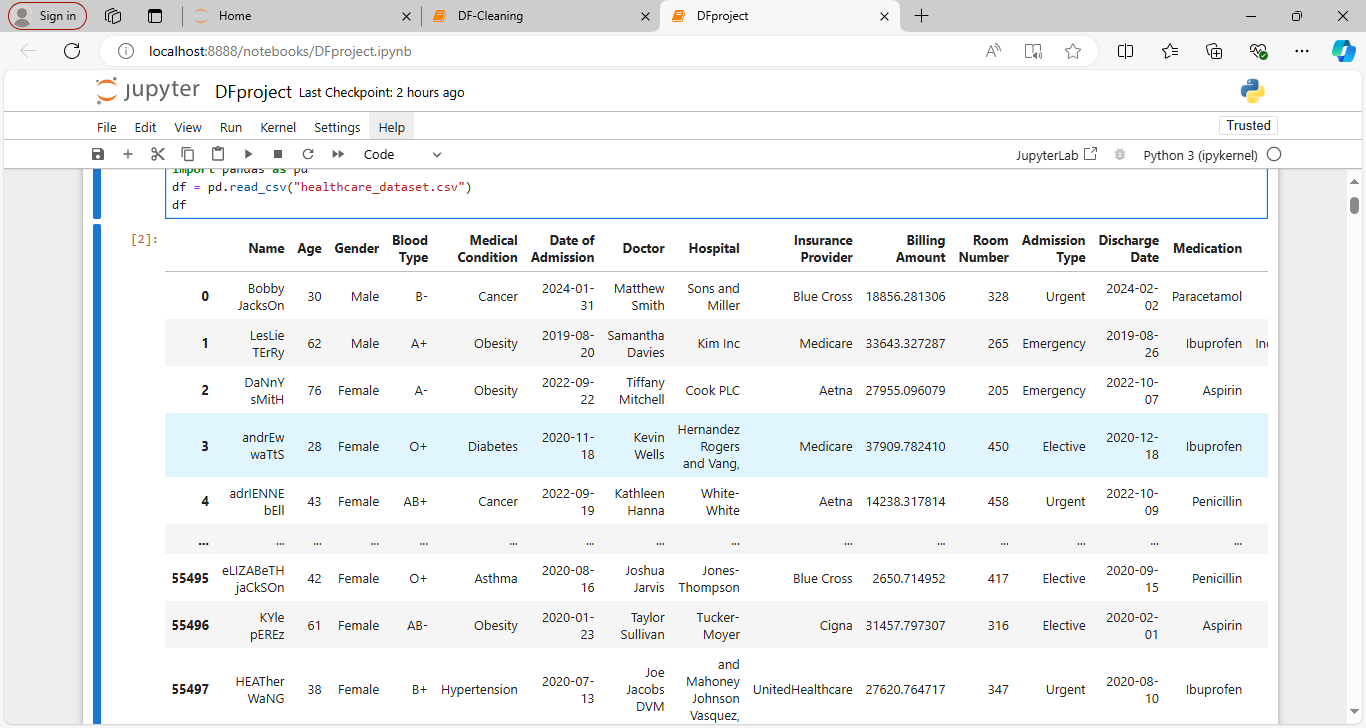
warnings.filterwarnings('ignore')

import pandas as pd

Read the file :

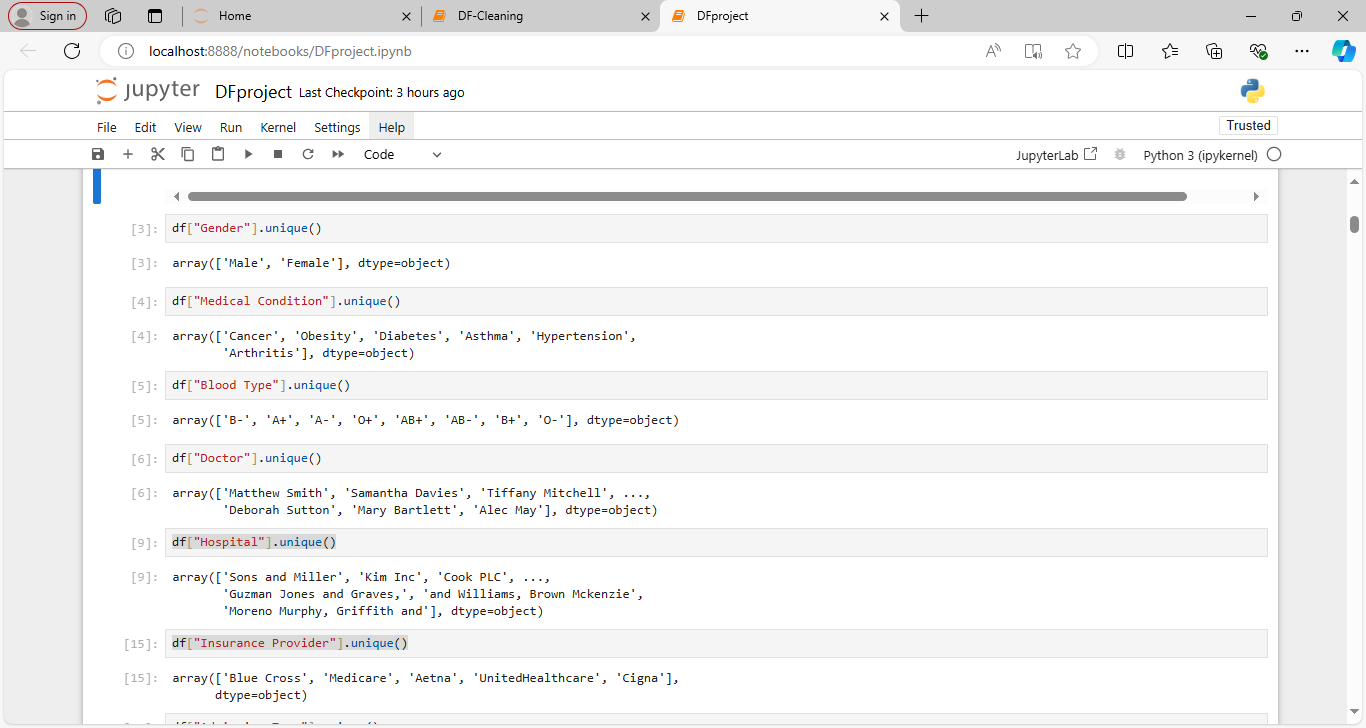
df = pd.read\_csv("healthcare\_dataset.csv")

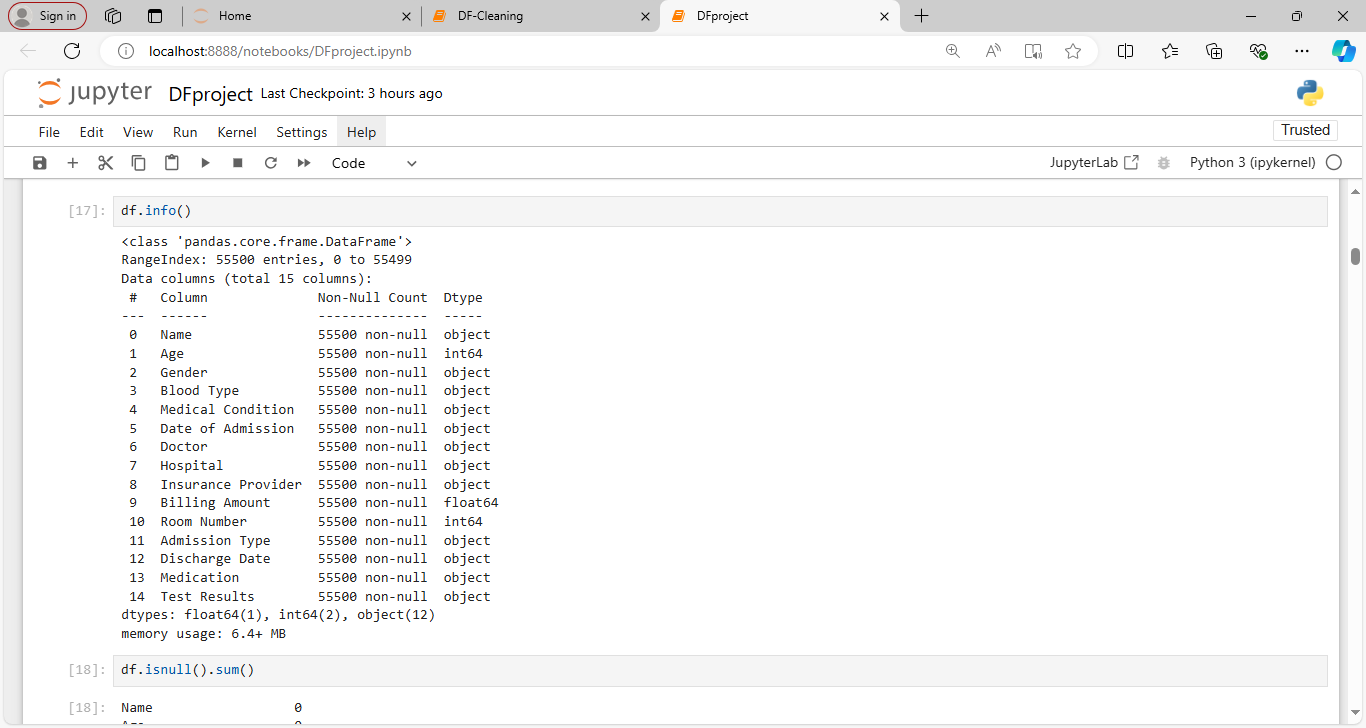
df

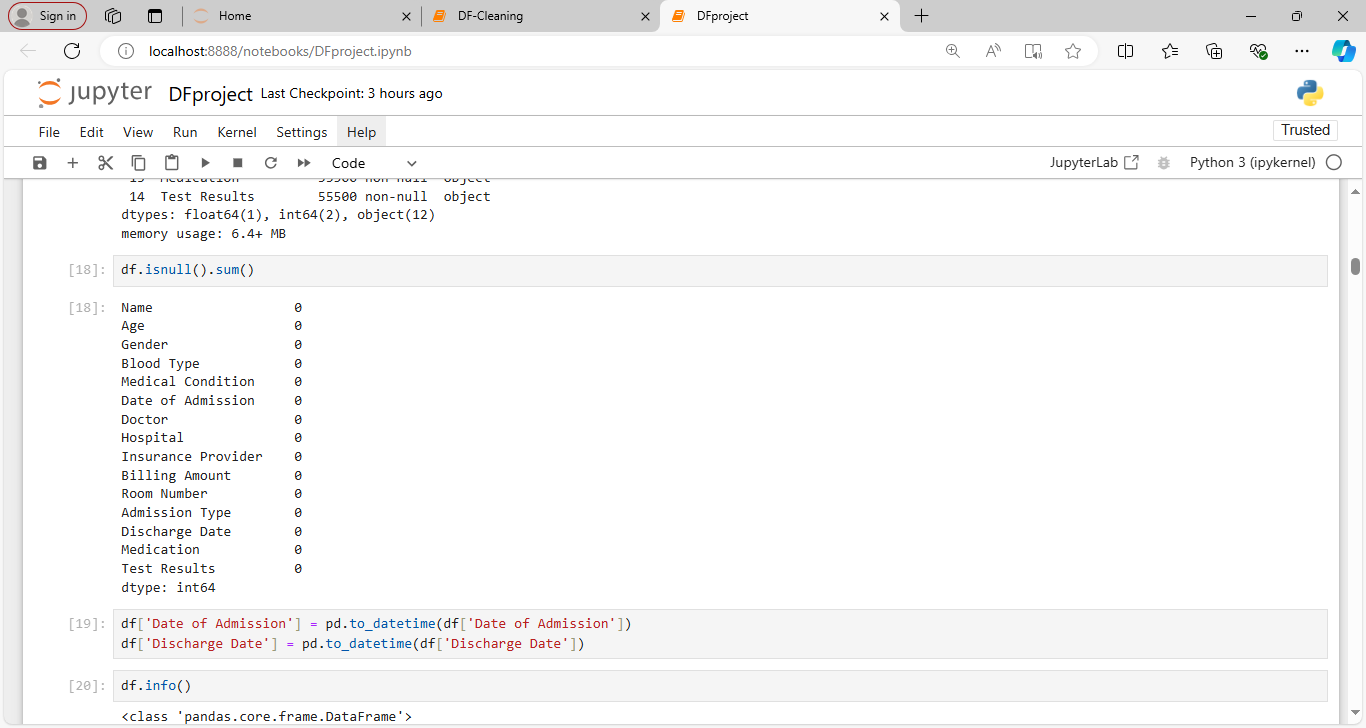


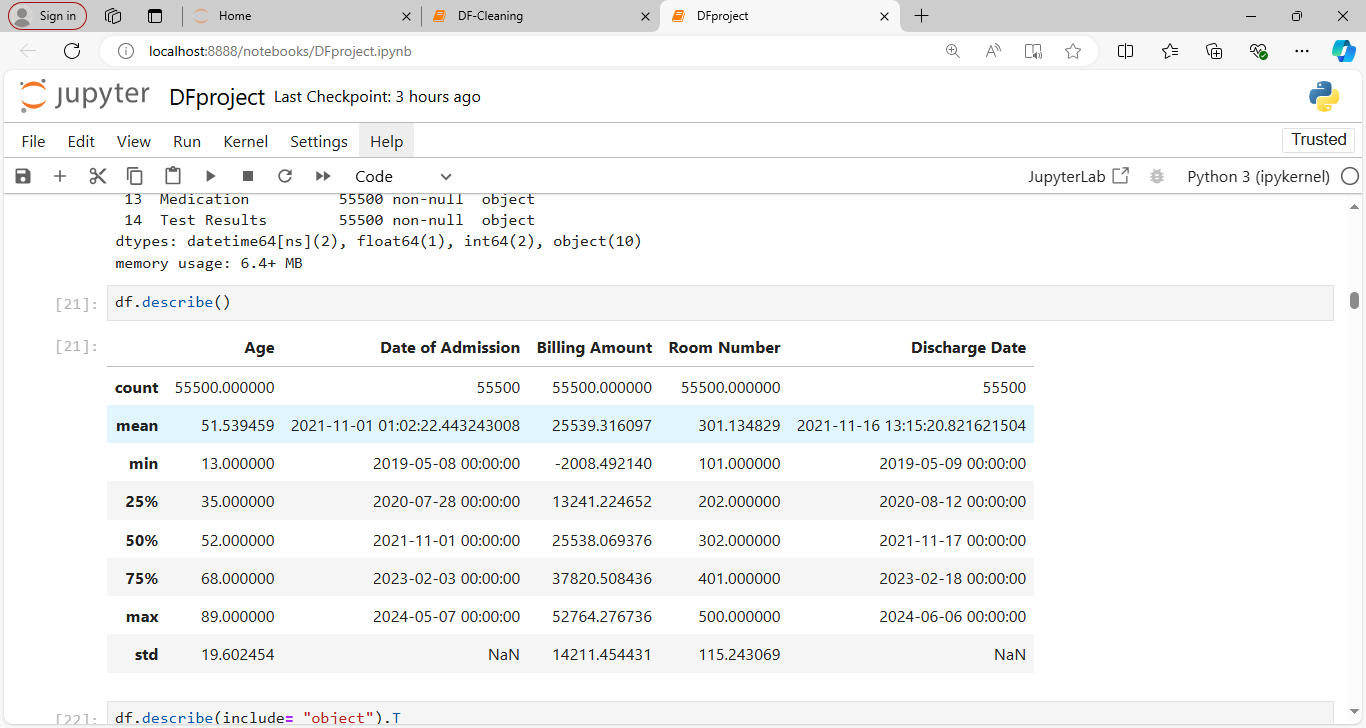
55500 rows × 15 columns

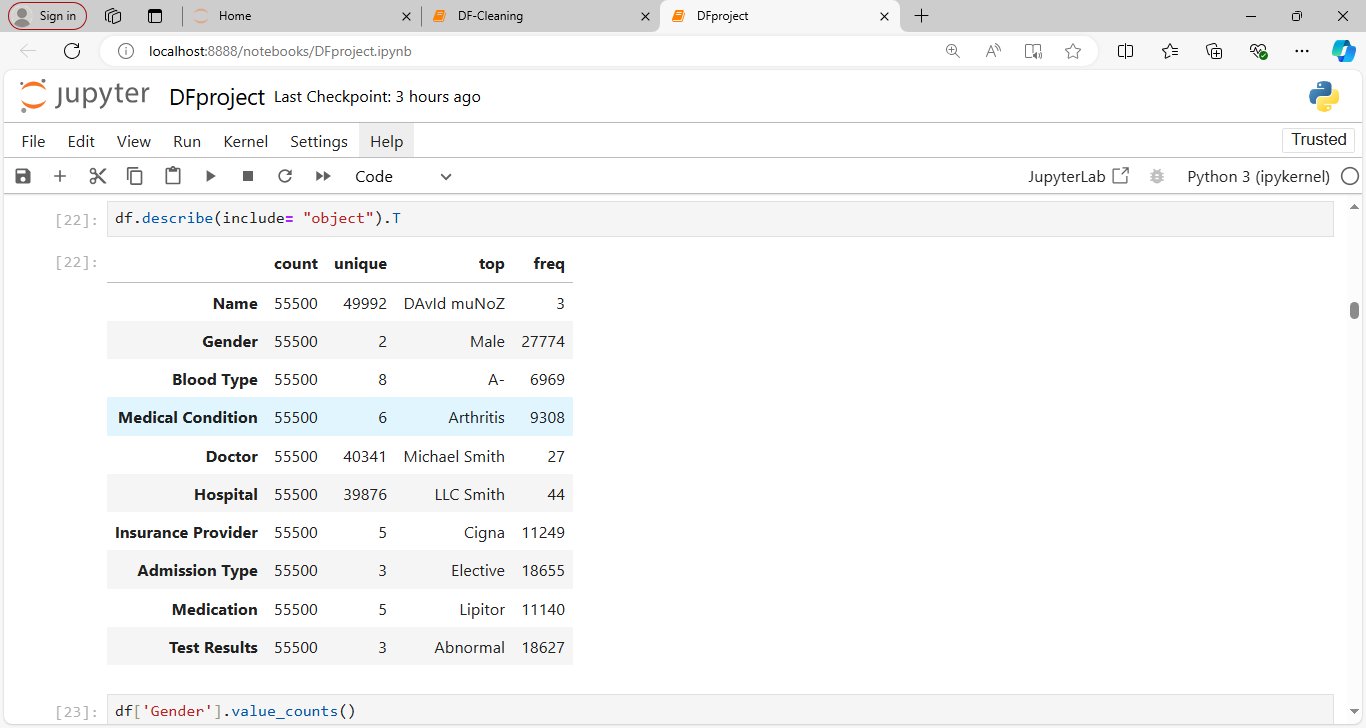
Find Unique values of each category

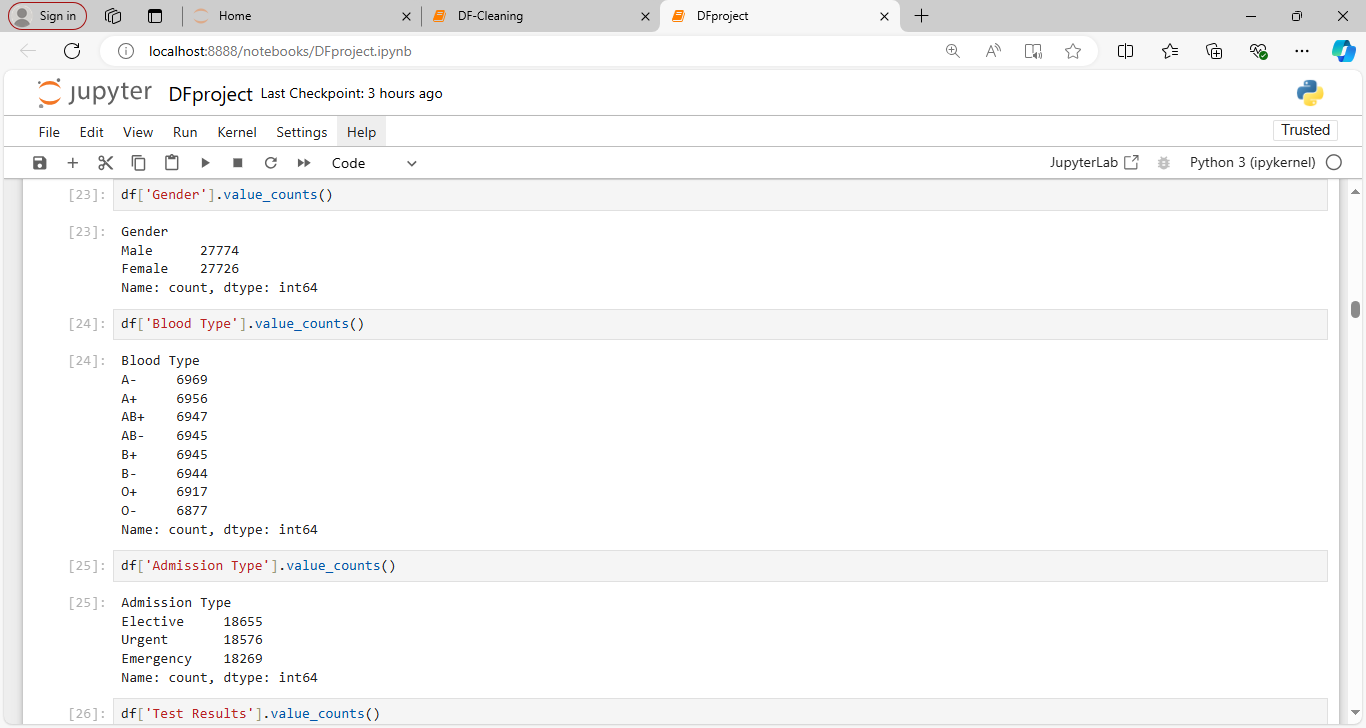












**Historgram plot for age distribution :**

**plt.figure(figsize=(10, 6))**

**plt.hist(df['Age'], bins=20, color='skyblue', edgecolor='black')**

**plt.title('Age Distribution')**

**plt.xlabel('Age')**

**plt.ylabel('Frequency')**

**plt.grid(True)**

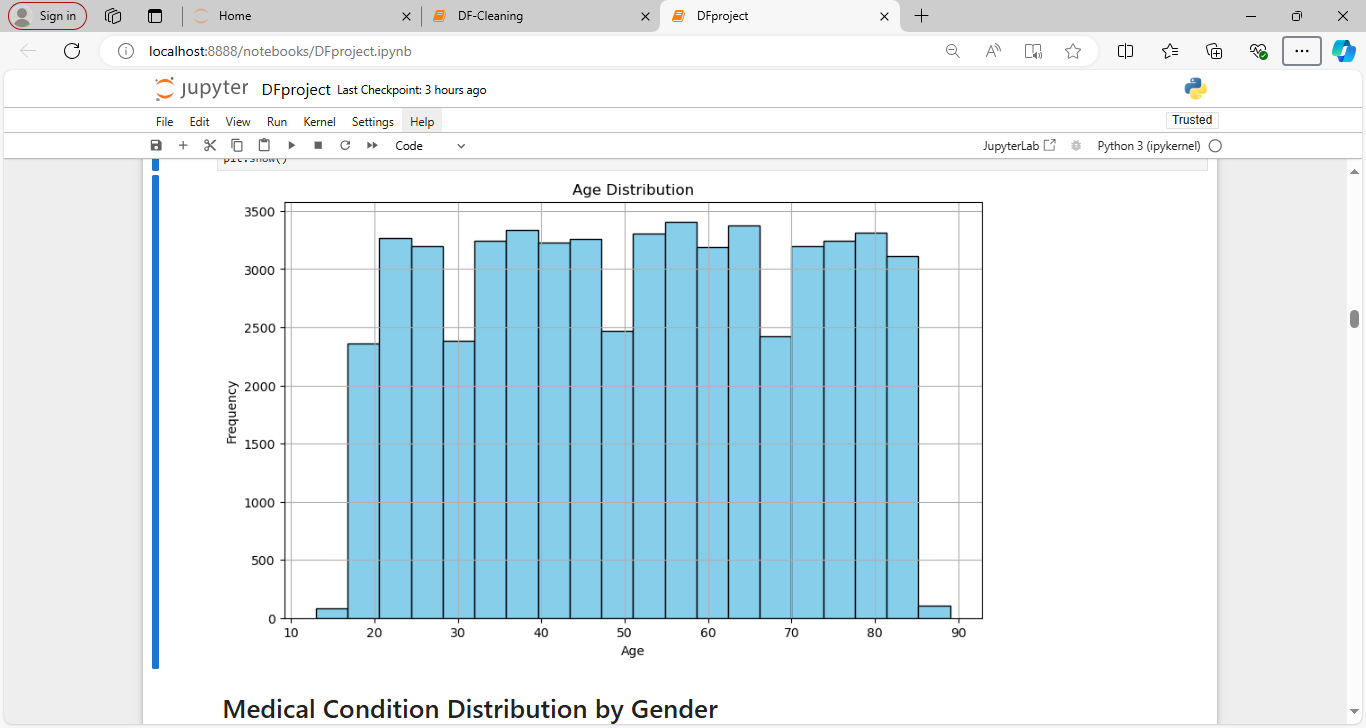
**plt.show()**

**Inference:**

**The maximum age of patents are lies in between 55 to 59**

**Maximum patients age between 20 to 80**

**Minimum patients age lies in 15 to 20 & 85 to 89**



**Medical Condition Distribution by Gender**

**import seaborn as sns**

**# Plot count plot for Gender vs. Medical Condition**

**plt.figure(figsize=(12, 8))**

**sns.countplot(data=df, x='Medical Condition', hue='Gender', palette='Set2')**

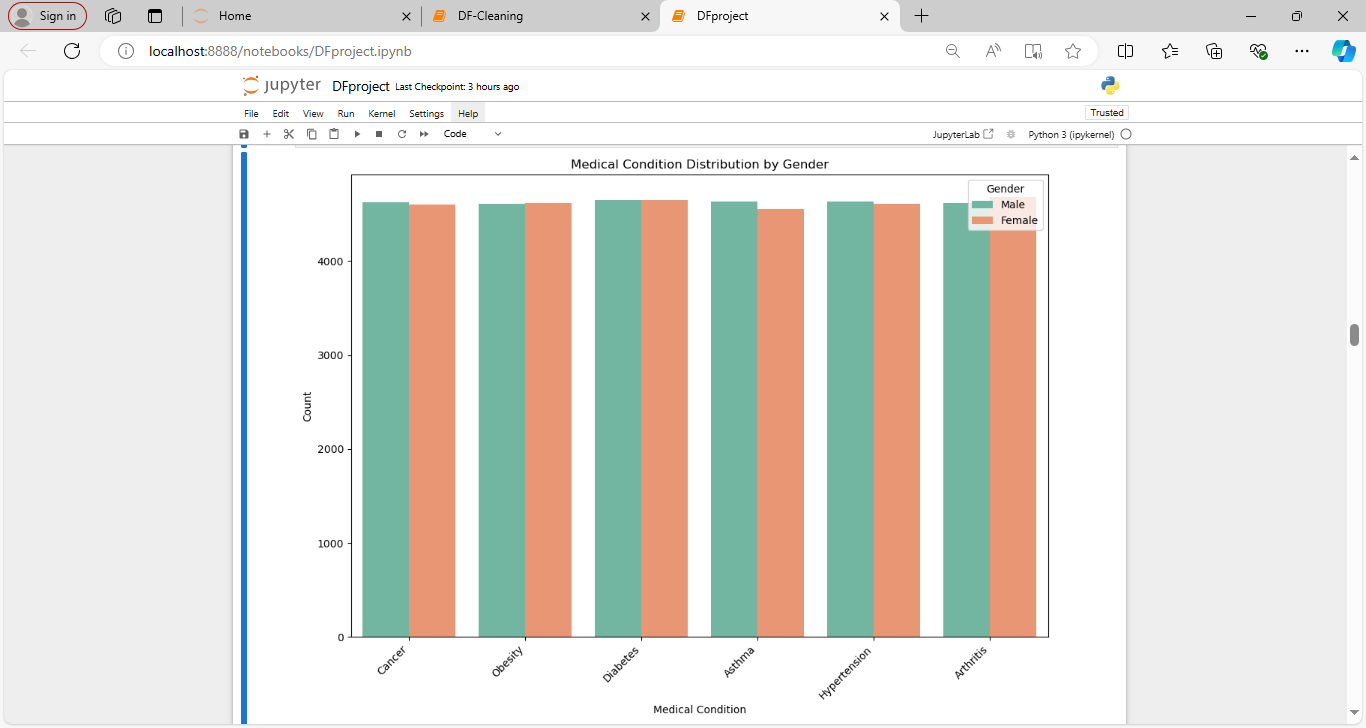
**plt.title('Medical Condition Distribution by Gender')**

**plt.xlabel('Medical Condition')**

**plt.ylabel('Count')**

**plt.xticks(rotation=45, ha='right')**

**plt.show()**



# Age Distribution by Medical Condition

plt.figure(figsize=(12, 8))

sns.barplot(data=df, x='Medical Condition', y='Age', palette='muted')

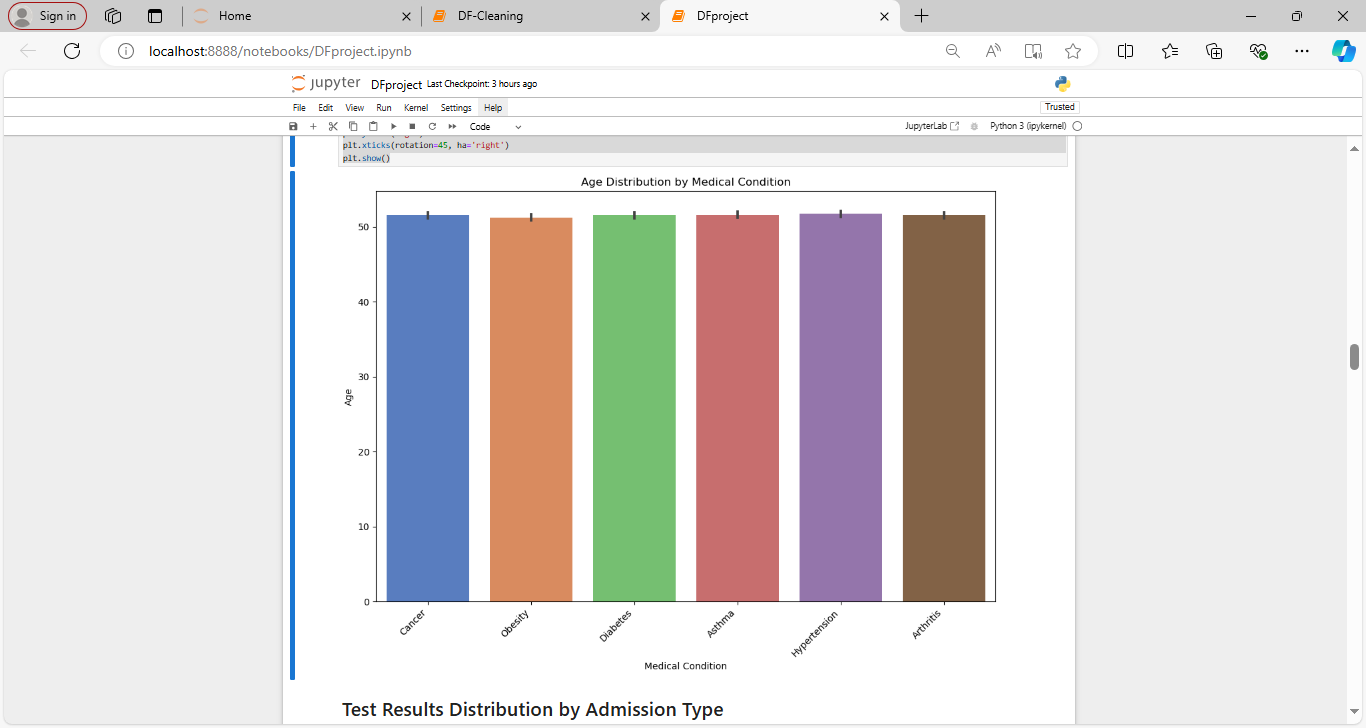
plt.title('Age Distribution by Medical Condition')

plt.xlabel('Medical Condition')

plt.ylabel('Age')

plt.xticks(rotation=45, ha='right')

plt.show()



# Test Results Distribution by Admission Type

# Plot count plot for Test Results by Admission Type

plt.figure(figsize=(12, 8))

sns.countplot(data=df, x='Admission Type', hue='Test Results', palette='Set2')

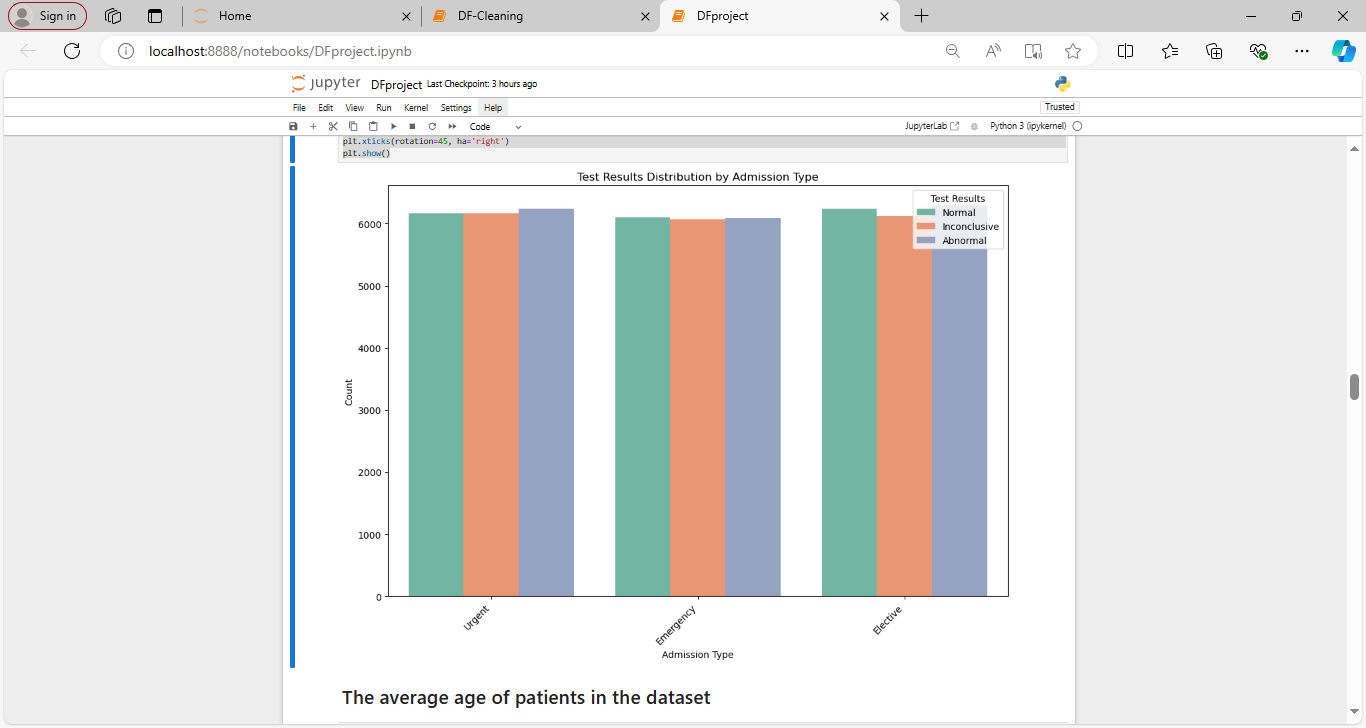
plt.title('Test Results Distribution by Admission Type')

plt.xlabel('Admission Type')

plt.ylabel('Count')

plt.xticks(rotation=45, ha='right')

plt.show()



# The average age of patients in the dataset

The average age of patients is 51.54 years.

**The most common medical condition among patients**

The most common medical condition is Arthritis.

# The average billing amount for each medical condition

Average billing amount by medical condition:

Arthritis: 25497.33

Asthma: 25635.25

Cancer: 25161.79

Diabetes: 25638.41

Hypertension: 25497.10

Obesity: 25805.97

**The most number of admissions of doctor**

Doctors with most admissions: Michael Smith

# Distribution of patients across different hospitals

Distribution of patients across hospitals: Hospital

LLC Smith 44

Ltd Smith 39

Johnson PLC 38

Smith Ltd 37

Smith PLC 36

..

and Montoya Flores, Boyer 1

Carter and Dunn King, 1

Hall, Brown Black and 1

Peterson Scott and Thomas, 1

Moreno Murphy, Griffith and 1

Name: count, Length: 39876, dtype: int64

**The oldest patient in the dataset, and their age**

The oldest patient in the dataset is DAVId NeWTOn with an age of 89 years.

# The Youngest patient in the dataset, and their age

The young patient in the dataset is jamES BasS phD with an age of 13 years.

# The most frequently prescribed medication

The most frequently prescribed medication is Lipitor.

# How many male and female patients are there?

There are 27774 Male patients and 27726 Female patients.

## **The top three most common medical conditions and patients admitted count**

Top Three Most Common Medical Conditions:

Medical Condition

Arthritis 9308

Diabetes 9304

Hypertension 9245

Name: count, dtype: int64

# Monthly admissions trend

try:

df['Date of Admission'] = pd.to\_datetime(df['Date of Admission'])

df['Discharge Date'] = pd.to\_datetime(df['Discharge Date'])

except ValueError:

df['Admission Year'] = df['Date of Admission'].dt.year

df['Admission Month'] = df['Date of Admission'].dt.month

df['Admission Day'] = df['Date of Admission'].dt.day

df['Admission Weekday'] = df['Date of Admission'].dt.weekday

else:

print("\n No exception occured \n ")

monthly\_admissions = df.groupby(['Admission Year', 'Admission Month']).size().reset\_index(name='Counts')

monthly\_admissions\_pivot = monthly\_admissions.pivot(index='Admission Month', columns='Admission Year', values='Counts')

finally:

plt.figure(figsize=(12, 6))

monthly\_admissions\_pivot.plot(kind='line', marker='o')

plt.xticks(ticks=range(1, 13), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])

plt.title('Monthly Admissions Trend')

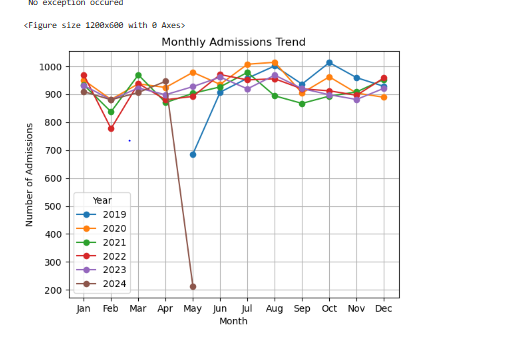
plt.xlabel('Month')

plt.ylabel('Number of Admissions')

plt.legend(title='Year')

plt.grid(True)

plt.show()



# Weekly admissions trend

weekly\_admissions = df.groupby(['Admission Year', 'Admission Weekday']).size().reset\_index(name='Counts')

weekly\_admissions\_pivot = weekly\_admissions.pivot(index='Admission Weekday', columns='Admission Year', values='Counts')

plt.figure(figsize=(12, 6))

weekly\_admissions\_pivot.plot(kind='line', marker='o')

plt.title('Weekly Admissions Trend')

plt.xlabel('Weekday')

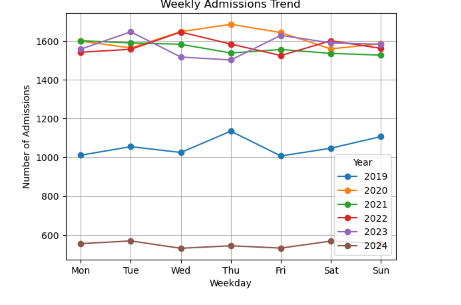
plt.ylabel('Number of Admissions')

plt.xticks(ticks=range(7), labels=['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'])

plt.legend(title='Year')

plt.grid(True)

plt.show()



# Univariate Analysis

# Box plot for one numerical column

plt.figure(figsize=(10, 6))

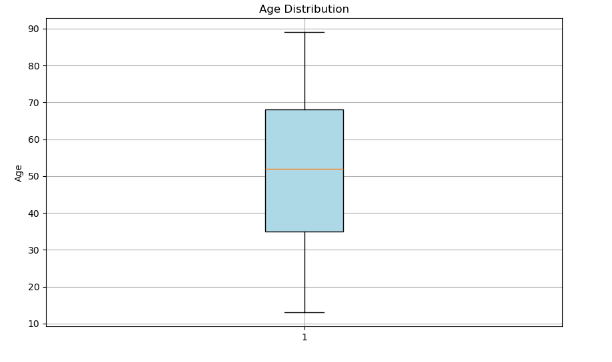
plt.boxplot(df['Age'], patch\_artist=True, boxprops=dict(facecolor='lightblue', color='black'))

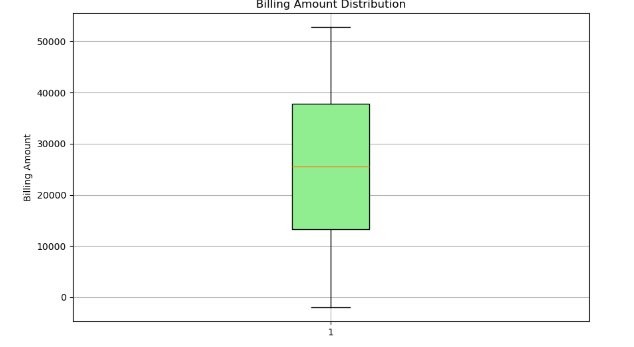
plt.title('Age Distribution')

plt.ylabel('Age')

plt.grid(True)

plt.show()





# Bivariate Analysis

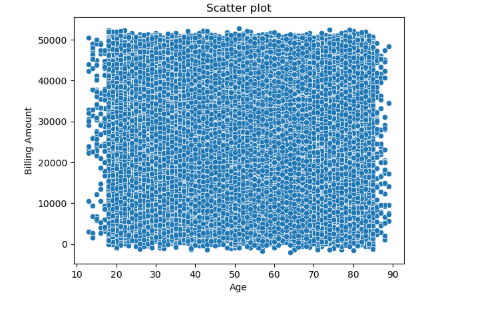
## Scatter plot for 2 numerical column

import seaborn as sns

sns.scatterplot( x='Age', y= 'Billing Amount', data = df)

plt.title("Scatter plot")

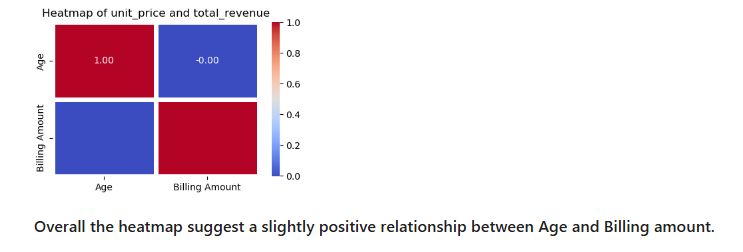
plt.show()



# Inference

## Scatterplot says the most data points are clustered together, there might be a few outliers with unusally high or low billings for theri age

### There seems to be a positive correlation between Age and billing amount [¶](http://localhost:8888/notebooks/DFproject.ipynb#There-seems-to-be-a-positive-correlation-between-Age-and-billing-amount-)

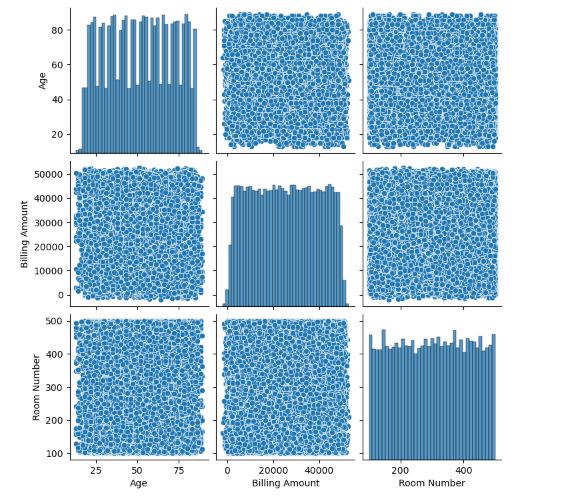


# Multivariate Analysis

### Explore interactions between multiple variables. Pair plots (scatter matrix), parallel coordinates plots, etc., to visualize complex relationships.[¶](http://localhost:8888/notebooks/DFproject.ipynb#Explore-interactions-between-multiple-variables.Pair-plots-(scatter-matrix),-parallel-coordinates-plots,-etc.,-to-visualize-complex-relationships.)

sns.pairplot(df)

plt.show()



# Hypothesis Testing

## One sample T-test

[54]:

# 

## Independent T-test

# 

# CHI-SQUARE TEST

# import numpy as np

# import scipy.stats as stats

# from scipy.stats import chi2\_contingency

# contingency\_table = pd.crosstab(df["Hospital"],df["Insurance Provider"])

# chi2, p, dof, expected = stats.chi2\_contingency(contingency\_table)

# print(f'Chi-square stattistic:{chi2}')

# print(f'P\_value :{p}')

# print(f'Degrees of freedom:{dof}')

# print('Expected frequencies:')

# print(expected)

# if dof < chi2:

# print(f"Reject null hypothesis at alpha ={alpha}")

# else:

# print(f"Faild to reject null hypothesis at alpha = {alpha}")

# 

# 

# 