Case Scenario

In this healthcare case study, we have a dataset containing the medical records of 20 patients admitted to a hospital for various treatments. The patients have been diagnosed with conditions such as Hypertension, Type 2 Diabetes, Pneumonia, and Heart Failure, and have received different treatments, including blood pressure control, insulin management, and respiratory support. The dataset includes details such as admission and discharge dates, age, gender, diagnoses, treatments, medications, vital signs, discharge status, and payment information.

The goal is to analyze and visualize several key aspects of patient health, treatment outcomes, hospital resource utilization, and financial details using Python. The dataset allows for an exploration of trends, correlations, and insights related to patient care.

Questions to be Analyzed

1. **What is the average length of stay for patients by diagnosis?**
    - We can analyze the average duration of patient stays in the hospital for different diagnoses (Hypertension, Type 2 Diabetes, Pneumonia, and Heart Failure). This analysis will help in understanding the hospital's capacity for handling various medical conditions.

    Python Analysis Steps:

    - Group the dataset by the `Diagnosis_Description`.
    - Calculate the average `Length_of_Stay` for each diagnosis.

    Visualization:

    - Bar chart to compare the average length of stay by diagnosis.
2. **What is the distribution of total costs for treatments by payment method?**
    - This analysis will provide insight into how payment methods (Insurance vs. Cash) impact the total cost of treatment.

    Python Analysis Steps:

    - Group the dataset by the `Payment_Method`.
    - Calculate the total cost of treatment for each payment method.

Visualization:

- Box plot or histogram to visualize the distribution of total costs for each payment method.

3. **What is the relationship between age and the total cost of treatment?**
   - Investigating whether older patients incur higher treatment costs. This could be due to longer hospital stays or more complex medical needs.

   Python Analysis Steps:

   - Plot a scatter plot between `Age` and `Total_Cost` to explore the relationship.

   Visualization:

   - Scatter plot with age on the x-axis and total cost on the y-axis.

4. **How do vital signs (temperature, heart rate, BP, and oxygen levels) vary across different diagnoses?**
   - We can analyze the average vital signs (temperature, heart rate, BP, oxygen levels) for each diagnosis group to understand if there are any noticeable trends or differences.

   Python Analysis Steps:

   - Group the dataset by `Diagnosis_Description`.
   - Calculate the average values of `Vital_Signs_Temperature`, `Vital_Signs_Heart_Rate`, `Vital_Signs_BP`, and `Vital_Signs_Oxygen`.

   Visualization:

   - Radar chart or grouped bar chart to compare the average vital signs across different diagnoses.

5. **What percentage of patients are discharged under "Improved," "Stable," and "Discharged" status?**
   - Analyzing the discharge status of patients will provide insights into patient recovery trends and the success of treatments.

   Python Analysis Steps:

   - Count the occurrences of each discharge status (`Discharge_Status`).
   - Calculate the percentage of patients for each status.

   Visualization:

   - Pie chart to show the percentage distribution of discharge statuses.

Python Analysis and Visualization Code Example

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv('hospital_data.csv')

# Check data types and inspect the 'Length_of_Stay' column
print(data.dtypes)
print(data['Length_of_Stay'].unique())  # Check unique values in
Length_of_Stay

# Convert 'Length_of_Stay' to numeric, invalid values will become NaN
data['Length_of_Stay'] = pd.to_numeric(data['Length_of_Stay'],
errors='coerce')

# Check if there are any NaN values after conversion
print(data['Length_of_Stay'].isna().sum())

# 1. Average length of stay by diagnosis
avg_stay_by_diagnosis =
data.groupby('Diagnosis_Description')['Length_of_Stay'].mean()
avg_stay_by_diagnosis.plot(kind='bar', color='skyblue')
plt.title('Average Length of Stay by Diagnosis')
plt.xlabel('Diagnosis')
plt.ylabel('Average Length of Stay (days)')
plt.show()

# 2. Distribution of total costs by payment method
sns.boxplot(x='Payment_Method', y='Total_Cost', data=data)
plt.title('Total Cost Distribution by Payment Method')
plt.xlabel('Payment Method')
plt.ylabel('Total Cost ($)')
plt.show()

# 3. Relationship between age and total cost
```

```python
plt.scatter(data['Age'], data['Total_Cost'], alpha=0.6)
plt.title('Age vs Total Cost of Treatment')
plt.xlabel('Age')
plt.ylabel('Total Cost ($)')
plt.show()

# 4. Average vital signs across diagnoses
# Ensure that vital signs are numeric
data[['Vital_Signs_Temperature', 'Vital_Signs_Heart_Rate',
'Vital_Signs_BP', 'Vital_Signs_Oxygen']] = \
    data[['Vital_Signs_Temperature', 'Vital_Signs_Heart_Rate',
'Vital_Signs_BP', 'Vital_Signs_Oxygen']].apply(pd.to_numeric,
errors='coerce')

vital_signs_avg =
data.groupby('Diagnosis_Description')[['Vital_Signs_Temperature',
'Vital_Signs_Heart_Rate', 'Vital_Signs_BP', 'Vital_Signs_Oxygen']].mean()
vital_signs_avg.plot(kind='bar', figsize=(10,6))
plt.title('Average Vital Signs by Diagnosis')
plt.xlabel('Diagnosis')
plt.ylabel('Average Vital Signs')
plt.xticks(rotation=45)
plt.show()

# 5. Percentage of patients discharged by status
discharge_status_percentage =
data['Discharge_Status'].value_counts(normalize=True) * 100
discharge_status_percentage.plot(kind='pie', autopct='%1.1f%%',
colors=['lightblue', 'orange', 'green'])
plt.title('Patient Discharge Status Percentage')
plt.ylabel('')
plt.show()
```

Importing Libraries:

python
Copy code
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

- **`import pandas as pd`**: Imports the pandas library, which is used for data manipulation and analysis. It's typically aliased as `pd` to shorten the code.
- **`import matplotlib.pyplot as plt`**: Imports the `matplotlib` library, specifically the `pyplot` module, for creating visualizations like plots and graphs. It's typically aliased as `plt` to make the code more concise.
- **`import seaborn as sns`**: Imports the seaborn library, which is built on top of `matplotlib` and provides a high-level interface for creating visually appealing and informative statistical plots.

Loading the Dataset:

python
Copy code
```
data = pd.read_csv('hospital_data.csv')
```

- **`data = pd.read_csv('hospital_data.csv')`**: Loads the dataset from a CSV file named `hospital_data.csv` into a pandas DataFrame called `data`. This allows for easy manipulation and analysis of the data.

Checking Data Types and Unique Values:

python
Copy code
```
print(data.dtypes)
print(data['Length_of_Stay'].unique())  # Check unique values in
Length_of_Stay
```

- **`print(data.dtypes)`**: Prints the data types of each column in the `data` DataFrame. This helps identify if any column needs to be converted to a different type (e.g., from string to numeric).
- **`print(data['Length_of_Stay'].unique())`**: Prints the unique values in the `Length_of_Stay` column. This helps check for any non-numeric values or unexpected data.

Converting `Length_of_Stay` to Numeric:

python
Copy code
```
data['Length_of_Stay'] = pd.to_numeric(data['Length_of_Stay'],
errors='coerce')
```

- **`data['Length_of_Stay'] = pd.to_numeric(data['Length_of_Stay'], errors='coerce')`**: Converts the `Length_of_Stay` column to numeric values. If any value cannot be converted (e.g., due to non-numeric characters), it will be replaced with `NaN` (Not a Number). This is done using the `errors='coerce'` argument.

Checking for Missing Values (`NaN`):

python
Copy code
```
print(data['Length_of_Stay'].isna().sum())
```

- **`data['Length_of_Stay'].isna().sum()`**: Checks how many `NaN` values are present in the `Length_of_Stay` column. This is important to ensure that invalid values were handled properly during the conversion to numeric.

Plotting the Average Length of Stay by Diagnosis:

python
Copy code
```
avg_stay_by_diagnosis =
data.groupby('Diagnosis_Description')['Length_of_Stay'].mean()
avg_stay_by_diagnosis.plot(kind='bar', color='skyblue')
plt.title('Average Length of Stay by Diagnosis')
plt.xlabel('Diagnosis')
plt.ylabel('Average Length of Stay (days)')
plt.show()
```

- **`data.groupby('Diagnosis_Description')['Length_of_Stay'].mean()`**: Groups the data by the `Diagnosis_Description` column and calculates the mean of `Length_of_Stay` for each diagnosis.
- **`avg_stay_by_diagnosis.plot(kind='bar', color='skyblue')`**: Plots the average length of stay for each diagnosis as a bar chart with a sky-blue color.

- **`plt.title('Average Length of Stay by Diagnosis')`**: Sets the title of the plot.
- **`plt.xlabel('Diagnosis')`**: Sets the label for the x-axis (representing diagnoses).
- **`plt.ylabel('Average Length of Stay (days)')`**: Sets the label for the y-axis (representing the length of stay in days).
- **`plt.show()`**: Displays the plot.

Plotting the Distribution of Total Costs by Payment Method:

python
Copy code
```python
sns.boxplot(x='Payment_Method', y='Total_Cost', data=data)
plt.title('Total Cost Distribution by Payment Method')
plt.xlabel('Payment Method')
plt.ylabel('Total Cost ($)')
plt.show()
```

- **`sns.boxplot(x='Payment_Method', y='Total_Cost', data=data)`**: Creates a box plot that shows the distribution of `Total_Cost` grouped by `Payment_Method`. This plot helps identify the spread and any outliers in the total cost for each payment method.
- **`plt.title('Total Cost Distribution by Payment Method')`**: Sets the title of the plot.
- **`plt.xlabel('Payment Method')`**: Sets the label for the x-axis (representing the payment method).
- **`plt.ylabel('Total Cost ($)')`**: Sets the label for the y-axis (representing the total cost).
- **`plt.show()`**: Displays the plot.

Plotting the Relationship Between Age and Total Cost:

python
Copy code
```python
plt.scatter(data['Age'], data['Total_Cost'], alpha=0.6)
plt.title('Age vs Total Cost of Treatment')
plt.xlabel('Age')
plt.ylabel('Total Cost ($)')
plt.show()
```

- **`plt.scatter(data['Age'], data['Total_Cost'], alpha=0.6)`**: Creates a scatter plot to show the relationship between `Age` and `Total_Cost`. The `alpha=0.6` parameter adds transparency to the points, making overlapping points more visible.
- **`plt.title('Age vs Total Cost of Treatment')`**: Sets the title of the plot.
- **`plt.xlabel('Age')`**: Sets the label for the x-axis (representing the age of the patients).
- **`plt.ylabel('Total Cost ($)')`**: Sets the label for the y-axis (representing the total cost of treatment).
- **`plt.show()`**: Displays the plot.

Plotting the Average Vital Signs by Diagnosis:

python
Copy code
```
data[['Vital_Signs_Temperature', 'Vital_Signs_Heart_Rate',
'Vital_Signs_BP', 'Vital_Signs_Oxygen']] = \
    data[['Vital_Signs_Temperature', 'Vital_Signs_Heart_Rate',
'Vital_Signs_BP', 'Vital_Signs_Oxygen']].apply(pd.to_numeric,
errors='coerce')
```

- **`data[['Vital_Signs_Temperature', 'Vital_Signs_Heart_Rate', 'Vital_Signs_BP', 'Vital_Signs_Oxygen']]`**: Selects the columns related to vital signs.
- **`.apply(pd.to_numeric, errors='coerce')`**: Converts the selected columns to numeric values, handling any non-numeric values by converting them to `NaN`.

python
Copy code
```
vital_signs_avg =
data.groupby('Diagnosis_Description')[['Vital_Signs_Temperature',
'Vital_Signs_Heart_Rate', 'Vital_Signs_BP',
'Vital_Signs_Oxygen']].mean()
vital_signs_avg.plot(kind='bar', figsize=(10,6))
plt.title('Average Vital Signs by Diagnosis')
plt.xlabel('Diagnosis')
plt.ylabel('Average Vital Signs')
plt.xticks(rotation=45)
plt.show()
```

- **`data.groupby('Diagnosis_Description')[...]`**: Groups the data by diagnosis description and calculates the mean for the selected vital sign columns.
- **`.plot(kind='bar', figsize=(10,6))`**: Creates a bar chart of the average vital signs for each diagnosis, with a specified figure size.
- **`plt.title('Average Vital Signs by Diagnosis')`**: Sets the title of the plot.
- **`plt.xlabel('Diagnosis')`**: Sets the label for the x-axis (representing diagnoses).
- **`plt.ylabel('Average Vital Signs')`**: Sets the label for the y-axis (representing average vital signs).
- **`plt.xticks(rotation=45)`**: Rotates the x-axis labels by 45 degrees for better readability.
- **`plt.show()`**: Displays the plot.

Plotting the Percentage of Patients Discharged by Status:

python
Copy code
```python
discharge_status_percentage =
data['Discharge_Status'].value_counts(normalize=True) * 100
discharge_status_percentage.plot(kind='pie', autopct='%1.1f%%',
colors=['lightblue', 'orange', 'green'])
plt.title('Patient Discharge Status Percentage')
plt.ylabel('')
plt.show()
```

- **`data['Discharge_Status'].value_counts(normalize=True) * 100`**: Calculates the percentage of patients with each discharge status. `value_counts(normalize=True)` gives the relative frequency, and multiplying by 100 converts it to a percentage.
- **`.plot(kind='pie', autopct='%1.1f%%', colors=['lightblue', 'orange', 'green'])`**: Plots a pie chart representing the discharge status percentages with specified colors and percentage formatting.
- **`plt.title('Patient Discharge Status Percentage')`**: Sets the title of the plot.
- **`plt.ylabel('')`**: Removes the y-axis label (since it's unnecessary for a pie chart).
- **`plt.show()`**: Displays the pie chart.

This code loads the data, processes it, and visualizes various aspects such as average length of stay, total costs, vital signs, and discharge status percentages.