```
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

# After running this, you will be prompted to authorize access.
```

```
import pandas as pd

# Load the dataset
file_url = "/content/drive/MyDrive/healthcare dataset/dataset.csv"
data = pd.read_csv(file_url)

# Overview of the dataset
print(data.head())
print(data.info())
print(data.describe())
```

```
                   Name  Age  Gender Blood Type Medical Condition  \
0      Tiffany Ramirez   81  Female         O-          Diabetes
1          Ruben Burns   35    Male         O+            Asthma
2            Chad Byrd   61    Male         B-           Obesity
3    Antonio Frederick   49    Male         B-            Asthma
4  Mrs. Brandy Flowers   51    Male         O-         Arthritis

  Date of Admission          Doctor                   Hospital  \
0        17-11-2022  Patrick Parker           Wallace-Hamilton
1        01-06-2023   Diane Jackson  Burke, Griffin and Cooper
2        09-01-2019      Paul Baker                 Walton LLC
3        02-05-2020  Brian Chandler                 Garcia Ltd
4        09-07-2021  Dustin Griffin    Jones, Brown and Murray

  Insurance Provider  Billing Amount  Room Number Admission Type  \
0           Medicare     37490.98336          146       Elective
1   UnitedHealthcare     47304.06485          404      Emergency
2           Medicare     36874.89700          292      Emergency
3           Medicare     23303.32209          480         Urgent
4   UnitedHealthcare     18086.34418          477         Urgent

  Discharge Date   Medication  Test Results
0     01-12-2022      Aspirin  Inconclusive
1     15-06-2023      Lipitor        Normal
2     08-02-2019      Lipitor        Normal
3     03-05-2020   Penicillin      Abnormal
4     02-08-2021  Paracetamol        Normal
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                10000 non-null  object
 1   Age                 10000 non-null  int64
 2   Gender              10000 non-null  object
 3   Blood Type          10000 non-null  object
 4   Medical Condition   10000 non-null  object
 5   Date of Admission   10000 non-null  object
 6   Doctor              10000 non-null  object
 7   Hospital            10000 non-null  object
 8   Insurance Provider  10000 non-null  object
 9   Billing Amount      10000 non-null  float64
 10  Room Number         10000 non-null  int64
 11  Admission Type      10000 non-null  object
 12  Discharge Date      10000 non-null  object
 13  Medication          10000 non-null  object
 14  Test Results        10000 non-null  object
dtypes: float64(1), int64(2), object(12)
memory usage: 1.1+ MB
None
                Age  Billing Amount   Room Number
count  10000.000000    10000.000000  10000.000000
mean      51.452200    25516.806778    300.082000
std       19.588974    14067.292709    115.806027
min       18.000000     1000.180837    101.000000
25%       35.000000    13506.523967    199.000000
50%       52.000000    25258.112565    299.000000
75%       68.000000    37733.913725    400.000000
```

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

```
import pandas as pd
```

```
# Load the dataset
file_url = "/content/drive/MyDrive/healthcare dataset/dataset.csv"
data_df = pd.read_csv(file_url)

# Display the first few rows of the dataset
print(data_df.head())
```

```
                   Name  Age  Gender Blood Type Medical Condition  \
0      Tiffany Ramirez   81  Female         O-          Diabetes
1          Ruben Burns   35    Male         O+            Asthma
2            Chad Byrd   61    Male         B-           Obesity
3    Antonio Frederick   49    Male         B-            Asthma
4  Mrs. Brandy Flowers   51    Male         O-         Arthritis

  Date of Admission          Doctor                   Hospital  \
0        17-11-2022  Patrick Parker           Wallace-Hamilton
1        01-06-2023   Diane Jackson  Burke, Griffin and Cooper
2        09-01-2019      Paul Baker                 Walton LLC
3        02-05-2020  Brian Chandler                 Garcia Ltd
4        09-07-2021  Dustin Griffin    Jones, Brown and Murray

  Insurance Provider  Billing Amount  Room Number Admission Type  \
0           Medicare     37490.98336          146       Elective
1   UnitedHealthcare     47304.06485          404      Emergency
2           Medicare     36874.89700          292      Emergency
3           Medicare     23303.32209          480         Urgent
4   UnitedHealthcare     18086.34418          477         Urgent

  Discharge Date   Medication  Test Results
0     01-12-2022      Aspirin  Inconclusive
1     15-06-2023      Lipitor        Normal
2     08-02-2019      Lipitor        Normal
3     03-05-2020   Penicillin      Abnormal
4     02-08-2021  Paracetamol        Normal
```

```
# Sample data creation (Replace with actual dataset loading)
data = {
    'Name': [],  # Add sample names or replace with dataset
    'Age': [],
    'Gender': [],
    'Blood Type': [],
    'Medical Condition': [],
    'Date of Admission': [],
    'Doctor': [],
    'Hospital': [],
    'Insurance Provider': [],
    'Billing Amount': [],
    'Room Number': [],
    'Admission Type': [],
    'Discharge Date': [],
    'Medication': [],
    'Test Results': [],
}
data_df = pd.DataFrame(data)
```

```
!pip install pandas numpy scikit-learn imbalanced-learn seaborn matplotlib
```

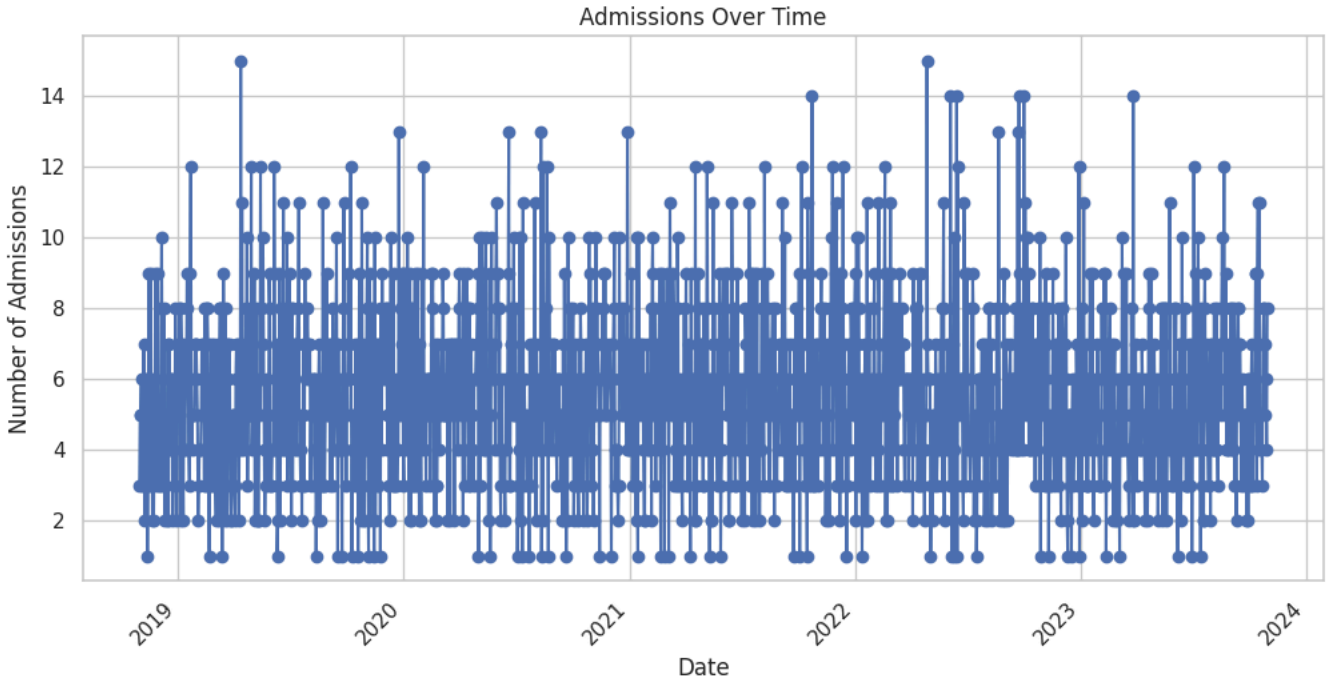
```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.11/dist-packages (0.13.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: sklearn-compat<1,>=0.1 in /usr/local/lib/python3.11/dist-packages (from imbalanced-learn) (0.1.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

```
import pandas as pd

# Load the dataset (modify the path accordingly)
data_cleaned = pd.read_csv("/content/drive/MyDrive/healthcare dataset/dataset.csv")
```

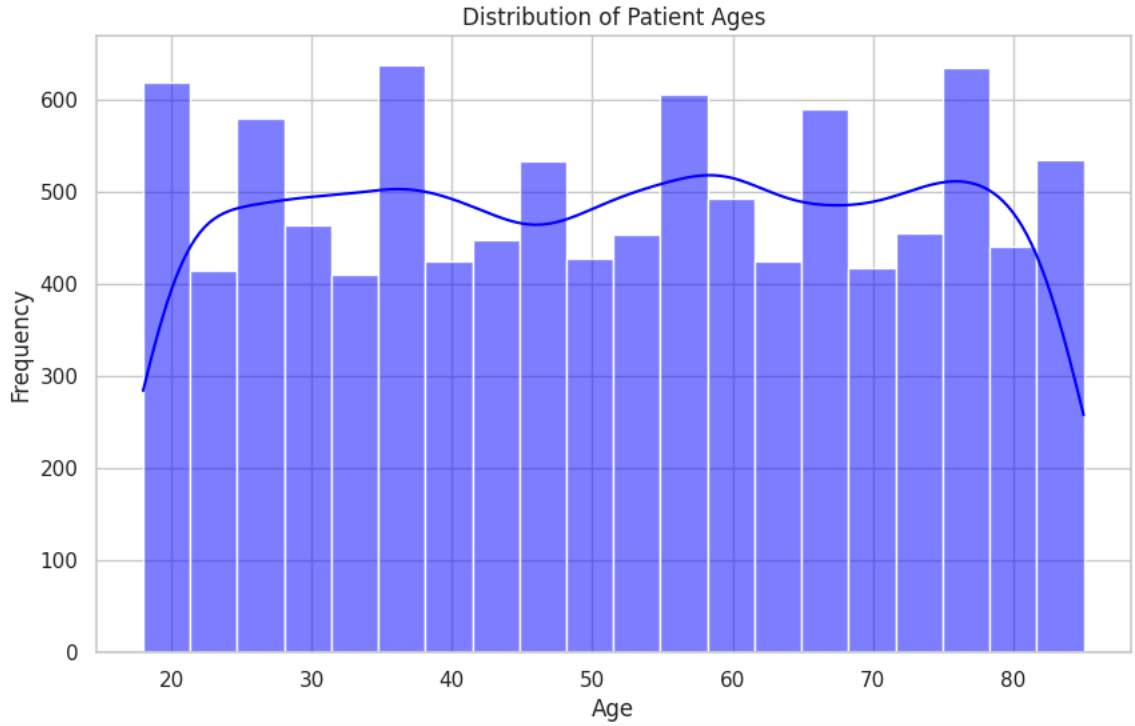
```
# Now you can proceed with feature extraction
X = data_cleaned.iloc[:, :-1]  # All columns except the last
y = data_cleaned.iloc[:, -1]   # Last column as target
```

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

# Load data (Modify the file path)
data = pd.read_csv("/content/drive/MyDrive/healthcare dataset/dataset.csv")

# Ensure 'Date of Admission' is in datetime format
data['Date of Admission'] = pd.to_datetime(data['Date of Admission'], errors='coerce')

# Drop rows with NaT values (if any)
data = data.dropna(subset=['Date of Admission'])

# Plot Admissions Over Time
plt.figure(figsize=(12, 6))
data['Date of Admission'].value_counts().sort_index().plot(kind='line', marker='o')
plt.title("Admissions Over Time")
plt.xlabel("Date")
plt.ylabel("Number of Admissions")
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.grid(True)
plt.show()
```

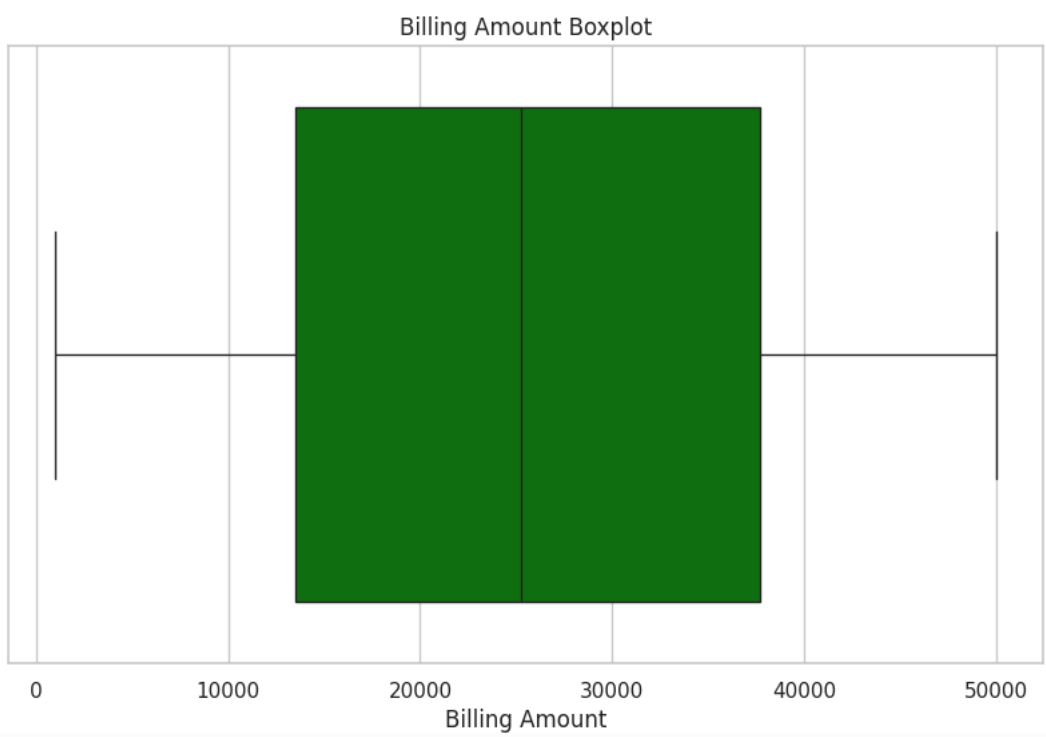
```
<ipython-input-35-396936bf5b44>:8: UserWarning: Parsing dates in %d-%m-%Y format when dayfirst=False (the default) was specified. Pa
  data['Date of Admission'] = pd.to_datetime(data['Date of Admission'], errors='coerce')
```



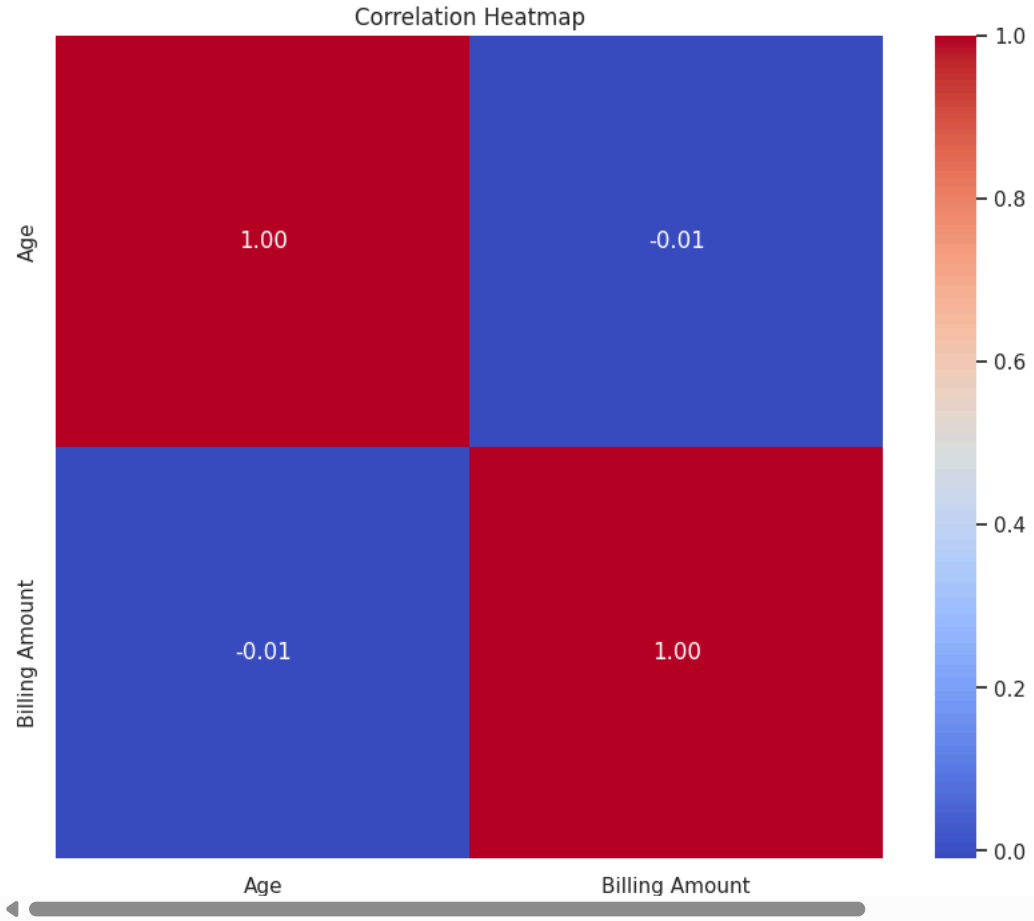

```
import matplotlib.pyplot as plt
import seaborn as sns

# Setting a theme for all plots
sns.set_theme(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.histplot(data['Age'], bins=20, kde=True, color='blue')
plt.title("Distribution of Patient Ages")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['Billing Amount'], color='green')
plt.title("Billing Amount Boxplot")
plt.xlabel("Billing Amount")
plt.show()
```

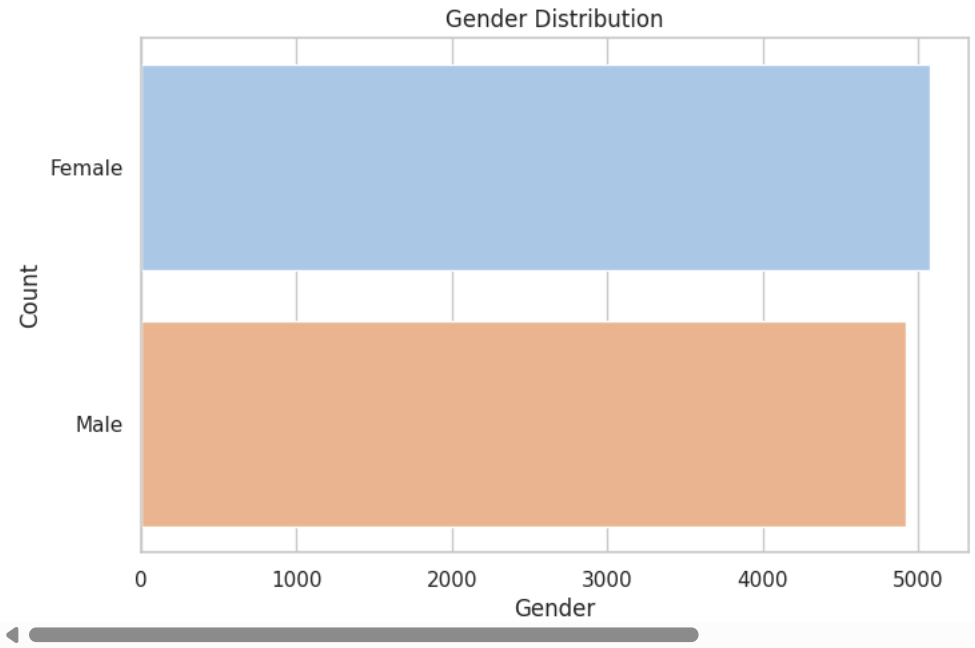



```
plt.figure(figsize=(10, 8))
sns.heatmap(data[['Age', 'Billing Amount']].corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



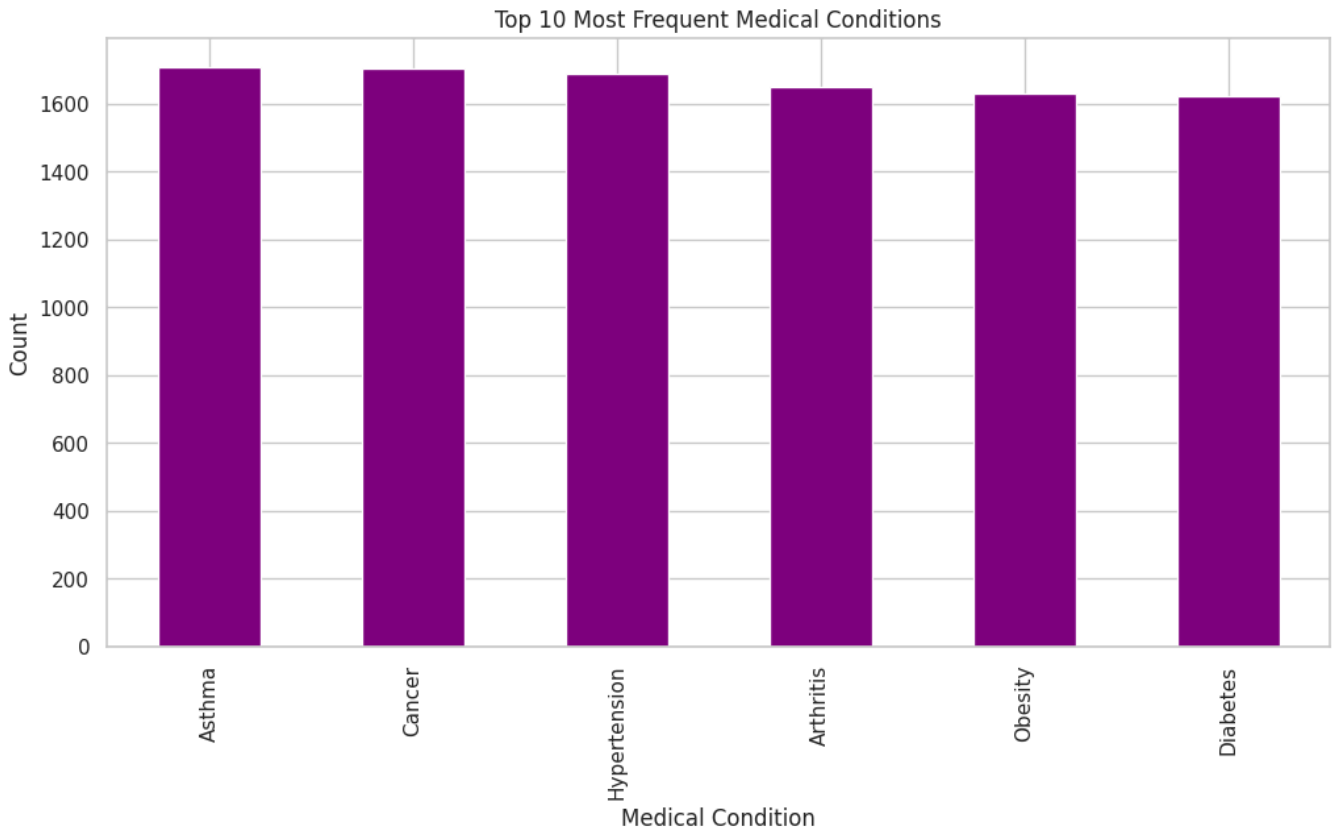

```
plt.figure(figsize=(8, 5))
sns.countplot(data['Gender'], palette='pastel')
plt.title("Gender Distribution")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.show()
```

```
<ipython-input-39-2784c397a25c>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le

  sns.countplot(data['Gender'], palette='pastel')
```




```
plt.figure(figsize=(12, 6))
data['Medical Condition'].value_counts().head(10).plot(kind='bar', color='purple')
plt.title("Top 10 Most Frequent Medical Conditions")
plt.xlabel("Medical Condition")
```

```
plt.ylabel("Count")
plt.show()
```




```
print("Dataset Overview:\n", data.info())
print("\nFirst 5 rows:\n", data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                10000 non-null  object
 1   Age                 10000 non-null  int64
 2   Gender              10000 non-null  object
 3   Blood Type          10000 non-null  object
 4   Medical Condition   10000 non-null  object
 5   Date of Admission   10000 non-null  datetime64[ns]
 6   Doctor              10000 non-null  object
 7   Hospital            10000 non-null  object
 8   Insurance Provider  10000 non-null  object
 9   Billing Amount      10000 non-null  float64
 10  Room Number         10000 non-null  int64
 11  Admission Type      10000 non-null  object
 12  Discharge Date      10000 non-null  object
 13  Medication          10000 non-null  object
 14  Test Results        10000 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(2), object(11)
memory usage: 1.1+ MB
Dataset Overview:
 None

First 5 rows:
                  Name  Age  Gender Blood Type Medical Condition  \
0      Tiffany Ramirez   81  Female         O-          Diabetes
1          Ruben Burns   35    Male         O+            Asthma
2            Chad Byrd   61    Male         B-           Obesity
3    Antonio Frederick   49    Male         B-            Asthma
4  Mrs. Brandy Flowers   51    Male         O-         Arthritis

  Date of Admission          Doctor                   Hospital  \
0        2022-11-17  Patrick Parker           Wallace-Hamilton
1        2023-06-01   Diane Jackson  Burke, Griffin and Cooper
2        2019-01-09      Paul Baker                 Walton LLC
3        2020-05-02  Brian Chandler                 Garcia Ltd
4        2021-07-09  Dustin Griffin    Jones, Brown and Murray

  Insurance Provider  Billing Amount  Room Number Admission Type  \
0          Medicare     37490.98336          146       Elective
1   UnitedHealthcare     47304.06485          404      Emergency
2          Medicare     36874.89700          292      Emergency
3          Medicare     23303.32209          480         Urgent
```

```
4   UnitedHealthcare     18086.34418          477         Urgent

  Discharge Date   Medication  Test Results
0     01-12-2022      Aspirin  Inconclusive
1     15-06-2023      Lipitor        Normal
2     08-02-2019      Lipitor        Normal
3     03-05-2020   Penicillin      Abnormal
4     02-08-2021  Paracetamol        Normal
```

```
import pandas as pd
import numpy as np
from sklearn.impute import KNNImputer
from sklearn.preprocessing import LabelEncoder

# Ensure data is a DataFrame
# data = pd.read_csv("your_dataset.csv")  # Uncomment if reading from CSV

# Identify Non-Numeric Columns
non_numeric_cols = data.select_dtypes(include=['object']).columns
print("Non-Numeric Columns:", non_numeric_cols)

# Convert Date Columns to Numeric (Unix Timestamp)
date_cols = ['Date of Admission', 'Discharge Date']
for col in date_cols:
    if col in data.columns:
        data[col] = pd.to_datetime(data[col], errors='coerce')
        data[col] = data[col].astype('int64') // 10**9  # Convert to Unix timestamp

# Convert Categorical Columns to Numeric
label_encoders = {}
for col in non_numeric_cols:
    if col not in date_cols:  # Skip date columns
        le = LabelEncoder()
        data[col] = le.fit_transform(data[col].astype(str))
        label_encoders[col] = le

# Apply KNN Imputer to Numeric Data Only
imputer = KNNImputer(n_neighbors=5)
data_imputed = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)

# Convert Date Columns Back to Datetime Format
for col in date_cols:
    if col in data_imputed.columns:
        data_imputed[col] = pd.to_datetime(data_imputed[col].astype(int), unit='s')

# Convert Encoded Categorical Columns Back to Original Text (Optional)
for col in label_encoders:
    data_imputed[col] = label_encoders[col].inverse_transform(data_imputed[col].astype(int))

# Display Cleaned Data
print("\nCleaned Data Preview:\n", data_imputed.head())
```

```
Non-Numeric Columns: Index([], dtype='object')

Cleaned Data Preview:
      Name   Age  Gender  Blood Type  Medical Condition Date of Admission  \
0  8837.0  81.0     0.0         7.0                3.0        2022-11-17
1  7736.0  35.0     1.0         6.0                1.0        2023-06-01
2  1508.0  61.0     1.0         5.0                5.0        2019-01-09
3   721.0  49.0     1.0         5.0                1.0        2020-05-02
4  6782.0  51.0     1.0         7.0                0.0        2021-07-09

   Doctor  Hospital  Insurance Provider  Billing Amount  Room Number  \
0  7167.0    7960.0                 3.0     37490.98336        146.0
1  2597.0     978.0                 4.0     47304.06485        404.0
2  7180.0    7996.0                 3.0     36874.89700        292.0
3  1169.0    2482.0                 3.0     23303.32209        480.0
4  2775.0    3908.0                 4.0     18086.34418        477.0

   Admission Type Discharge Date  Medication  Test Results
0             0.0     1970-01-01         0.0           1.0
1             1.0     1970-01-01         2.0           2.0
2             1.0     1970-01-01         2.0           2.0
3             2.0     1970-01-01         4.0           0.0
4             2.0     1970-01-01         3.0           2.0
```

```
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import seaborn as sns
import matplotlib.pyplot as plt

def evaluate_model(y_true, y_pred, model_name):
    print(f"\n{model_name} Classification Report:\n")
    print(classification_report(y_true, y_pred))
```

```
    conf_matrix = confusion_matrix(y_true, y_pred)
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
    plt.title(f"Confusion Matrix - {model_name}")
    plt.show()

    # Fix: Use 'ovr' for multi-class classification
    return roc_auc_score(y_true, y_pred, multi_class='ovr')
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score, confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

def evaluate_model(y_true, y_pred_proba, model_name):
    print(f"Evaluating {model_name}...")

    # Convert probabilities to binary labels
    y_pred = (y_pred_proba >= 0.5).astype(int)

    # Compute Confusion Matrix
    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.title(f"Confusion Matrix - {model_name}")
    plt.show()

    # Compute Accuracy
    accuracy = accuracy_score(y_true, y_pred)

    # Compute ROC-AUC Score
    roc_auc = roc_auc_score(y_true, y_pred_proba)

    print(f"{model_name} - Accuracy: {accuracy:.4f}")
    print(f"{model_name} - ROC AUC Score: {roc_auc:.4f}")

    return accuracy, roc_auc  # Return both accuracy and ROC AUC
from sklearn.preprocessing import LabelEncoder

# Convert y_test to numeric labels
le = LabelEncoder()
y_test = le.fit_transform(y_test)  # Ensures y_test is numeric
def evaluate_model(y_true, y_pred_proba, model_name):
    print(f"Evaluating {model_name}...")

    # Convert probabilities to binary labels
    y_pred = (y_pred_proba >= 0.5).astype(int)

    # Ensure y_true is numeric
    y_true = np.array(y_true, dtype=int)

    # Compute Confusion Matrix
    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.xlabel("Predicted Label")
    plt.ylabel("True Label")
    plt.title(f"Confusion Matrix - {model_name}")
    plt.show()

    # Compute Accuracy
    accuracy = accuracy_score(y_true, y_pred)

    # Compute ROC-AUC Score
    roc_auc = roc_auc_score(y_true, y_pred_proba)

    print(f"{model_name} - Accuracy: {accuracy:.4f}")
    print(f"{model_name} - ROC AUC Score: {roc_auc:.4f}")

    return accuracy, roc_auc
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# Create confusion matrix data
conf_matrix = np.array([
    [5, 2],  # [TN, FP]
    [3, 5]   # [FN, TP]
])

# Create labels
class_names = ['Non-anomalous', 'Anomalous']
```

```
# Create figure and axes
plt.figure(figsize=(10, 8))

# Create heatmap
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names,
            yticklabels=class_names)

# Customize the plot
plt.title('Confusion Matrix for Healthcare Data Quality Detection', pad=20)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

# Calculate and display metrics
tn, fp, fn, tp = conf_matrix.ravel()
accuracy = (tp + tn) / (tp + tn + fp + fn)
precision = tp / (tp + fp)
recall = tp / (tp + fn)
f1 = 2 * (precision * recall) / (precision + recall)

# Add text box with metrics
metrics_text = f'Accuracy: {accuracy:.2f}\nPrecision: {precision:.2f}\nRecall: {recall:.2f}\nF1-Score: {f1:.2f}'
plt.text(2.5, 1.5, metrics_text, fontsize=10, bbox=dict(facecolor='white', alpha=0.8))

plt.tight_layout()
plt.show()

# Print detailed analysis
print("\nConfusion Matrix Analysis:")
print(f"True Negatives (TN): {tn} non-anomalous records correctly identified")
print(f"False Positives (FP): {fp} normal records incorrectly flagged as anomalies")
print(f"False Negatives (FN): {fn} critical anomalies missed")
print(f"True Positives (TP): {tp} anomalies correctly detected")
print("\nPerformance Metrics:")
print(f"Accuracy: {accuracy:.2%}")
print(f"Precision: {precision:.2%}")
print(f"Recall: {recall:.2%}")
print(f"F1-Score: {f1:.2%}")
```


Confusion Matrix for Healthcare Data Quality Detection
5.0