Pandas and Seaborn based homework DSE5002 Module 2 Lab 01 Peter Gyorda, revised March 29 2025 We will be working with the heart.csv data set https://www.kaggle.com/fedesoriano/heart-failure-prediction?select=heart.csv using tools in pandas and seaborn, and ideas from the two Jupyter notebooks we've seen this week

In [69]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import plotnine as p9
import matplotlib.pyplot as plt
import os
```

In [7]:

```
# make sure heart.csv is in your current working directory, or list the full path n

infile="heart.csv"

bp_df=pd.read_csv(infile)
bp_df.head()
```

Out[7]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exer
0	40	M	ATA	140	289	0	Normal	172	
1	49	F	NAP	160	180	0	Normal	156	
2	37	M	ATA	130	283	0	ST	98	
3	48	F	ASY	138	214	0	Normal	108	
4	54	M	NAP	150	195	0	Normal	122	

Find or create the following

a.) -Find the dimensions, memory used, and other basic information

b.) -Run the data summary

c.) Change the appropriate variables to type Categorical

d.) -Create a pivot table (using the Pandas groupby operation) showing mean Resting BP by Sex, Resting ECG and HeartDisease-What does this tell you? What else can you figure out using a Pivot table, show me two other helpful pivot tables based on different variables, different groupings or different aggregation functions (count, mean, max etc)

e.) -Show a histogram and the ECDF (empirical cumulative distribution function) for several continuous variables in the data set, in broad terms, what do the distributions look like, normal? exponential, poison-like?, uniform? Does this match your expectations?

```
https://seaborn.pydata.org/generated/seaborn.ecdfplot.html
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.ecdf.html
```

f.) -Show An SNS Pairplot, the most informative version you can find, set the hue based on Heart Disease, try using at least one other variable as the Hue. Discuss what you think you are seeing in this plot

Create all these results in this Notebook and turn it in

In [ ]:

```
g.) Create several useful or informative boxplots of continuous variables by catego
among the variables,   discuss what you think it means or implies

h.) Create violin plots of these same results
```

In [ ]:

```
1.) Find the mean, median and standard deviation of the Max heartrate variable in t

Turn this into a pivot table,   grouping by one or more predictors.
```

In [33]:

```
#a/b. Find the dimensions, memory used, and other basic information and run the dat
import pandas as pd

 # Direct approach without a function
print("\n--- Dataset Information (Direct approach) ---")
print(f"Shape (Rows, Columns): {bp_df.shape}")


# Get memory usage in bytes
memory_bytes = bp_df.memory_usage(deep=True).sum()

# Convert to more readable format
if memory_bytes < 1024:
    memory_str = f"{memory_bytes} bytes"
elif memory_bytes < 1024**2:
    memory_str = f"{memory_bytes/1024:.2f} KB"
elif memory_bytes < 1024**3:
    memory_str = f"{memory_bytes/(1024**2):.2f} MB"
else:
    memory_str = f"{memory_bytes/(1024**3):.2f} GB"

print(f"Memory Usage: {memory_str}")

def analyze_dataset(bp_df):
    """
    Comprehensive analysis of a pandas DataFrame

    Args:
        df (pd.DataFrame): The DataFrame to analyze
    """
    print("=" * 50)
    print("DATASET OVERVIEW")
    print("=" * 50)

def dataset_overview(df):
    """
    Basic overview of a pandas DataFrame

    Args:
        df (pd.DataFrame): The DataFrame to analyze
    """
    print("=== DATASET OVERVIEW ===")

    # Shape
    print(f"\nShape: {df.shape} (rows, columns)")

    # Data types
    print("\nData Types:")
    for col, dtype in df.dtypes.items():
        print(f"  {col}: {dtype}")

    # Missing values
    print("\nMissing Values:")
    missing = df.isnull().sum()
    for col, count in missing.items():
        if count > 0:
            print(f"  {col}: {count}")

    # Preview
    print("\nData Preview:")
    print(df.head(3))

# Call the function with your dataset
dataset_overview(bp_df)
```

```
--- Dataset Information (Direct approach) ---
Shape (Rows, Columns): (918, 12)
Memory Usage: 317.21 KB
=== DATASET OVERVIEW ===

Shape: (918, 12) (rows, columns)

Data Types:
  Age: int64
  Sex: object
  ChestPainType: object
  RestingBP: int64
  Cholesterol: int64
  FastingBS: int64
  RestingECG: object
  MaxHR: int64
  ExerciseAngina: object
  Oldpeak: float64
  ST_Slope: object
  HeartDisease: int64

Missing Values:

Data Preview:
   Age Sex ChestPainType  RestingBP  Cholesterol  FastingBS RestingECG  MaxHR  \
0   40   M           ATA        140          289          0     Normal    172
1   49   F           NAP        160          180          0     Normal    156
2   37   M           ATA        130          283          0         ST     98

  ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0              N      0.0       Up             0
1              N      1.0     Flat             1
2              N      0.0       Up             0
```

In [45]:

```python
#c.) Change the appropriate variables to type Categorical

import pandas as pd

def convert_categorical_columns(bp_df):
    categorical_cols = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST
    for col in categorical_cols:
        bp_df[col] = bp_df[col].astype('category')
    return bp_df

# Call the function to modify the DataFrame
bp_df = convert_categorical_columns(bp_df)

# Print the data types after the conversion
print("\nData Types:")
for col, dtype in bp_df.dtypes.items():
    print(f"  {col}: {dtype}")
```

```
Data Types:
  Age: int64
  Sex: category
  ChestPainType: category
  RestingBP: int64
  Cholesterol: int64
  FastingBS: int64
  RestingECG: category
  MaxHR: int64
  ExerciseAngina: category
  Oldpeak: float64
  ST_Slope: category
  HeartDisease: int64
```

In [61]:

In [62]:

```
#d.) -Create a pivot table (using the Pandas groupby operation) showing mean Restin

import pandas as pd
import io

# Create your DataFrame properly first
bp_df = pd.read_csv('heart.csv')  # Replace with your actual data source

def analyze_heart_data_groupby(bp_df):
    grouped = bp_df.groupby(['Sex', 'RestingECG', 'HeartDisease'])['RestingBP'].mea
    output = grouped.to_string(index=False)
    print(output)

analyze_heart_data_groupby(bp_df)
```

```
Sex RestingECG  HeartDisease  RestingBP
  F        LVH             0 128.696970
  F        LVH             1 148.928571
  F     Normal             0 129.123596
  F     Normal             1 139.310345
  F         ST             0 127.523810
  F         ST             1 139.285714
  M        LVH             0 131.836735
  M        LVH             1 135.467391
  M     Normal             0 129.921348
  M     Normal             1 130.675781
  M         ST             0 134.275000
  M         ST             1 137.727273
```

In [66]:

```
# Conclusions:
#Sex Differences:
#In general, males tend to have higher average resting blood pressure (RestingBP) t

#Resting ECG Impact:
#For females, those with ST abnormalities in their RestingECG tend to have higher a
#For males, those with ST abnormalities also have a higher average resting blood pr
#Males with LVH(Left ventricular hypertrophy) have the lowest resting blood pressur
#Heart Disease Correlation:

#For both males and females, the mean RestingBP tends to be higher in individuals w
```

In [65]:

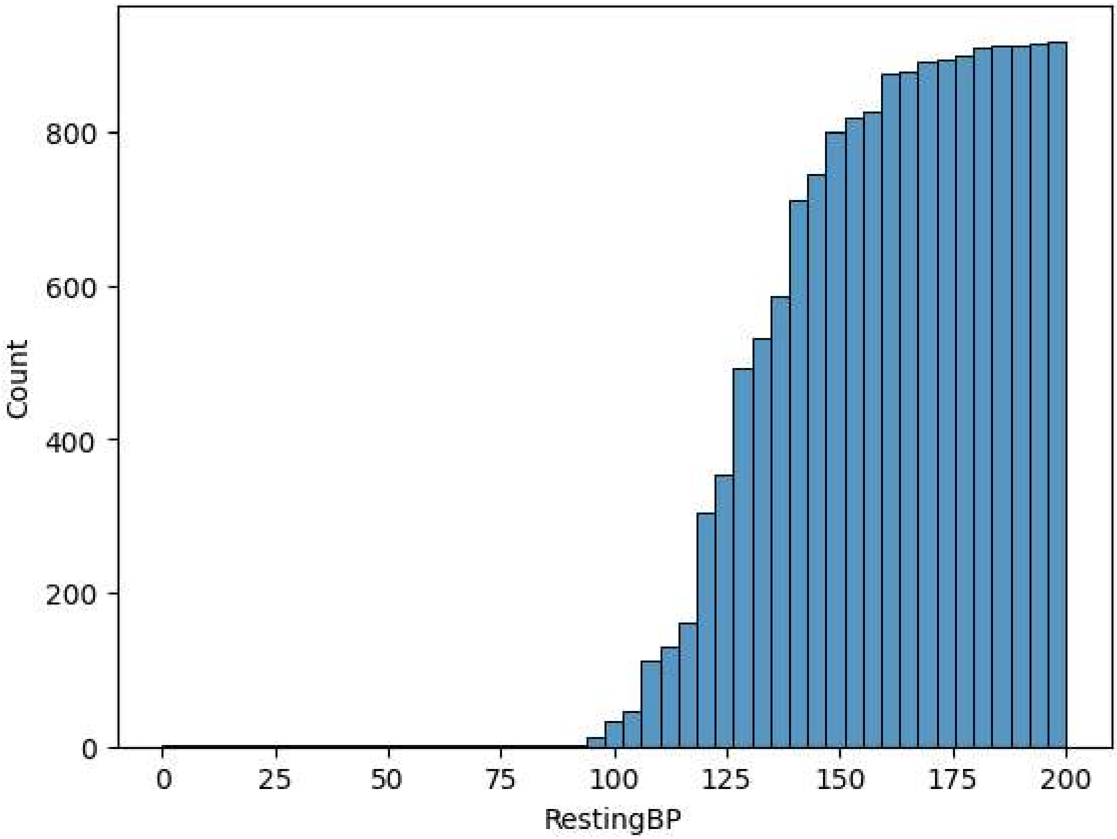
```
#e.) Show a histogram and the ECDF (empirical cumulative distribution function) for
#    continuous variables in the data set, in broad terms, what do the distribution
#    normal? exponential, poison-like?, uniform? Does this match your expectations?

#histogram
sns.histplot(bp_df,x="RestingBP",cumulative=True)
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/seaborn/_o
ldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be remove
d in a future version. Convert inf values to NaN before operating instead.
```

Out[65]:

```
<Axes: xlabel='RestingBP', ylabel='Count'>
```



In [ ]:

```
#ANSWER:  This distribution is exponential.  In short the number of people with Res
```

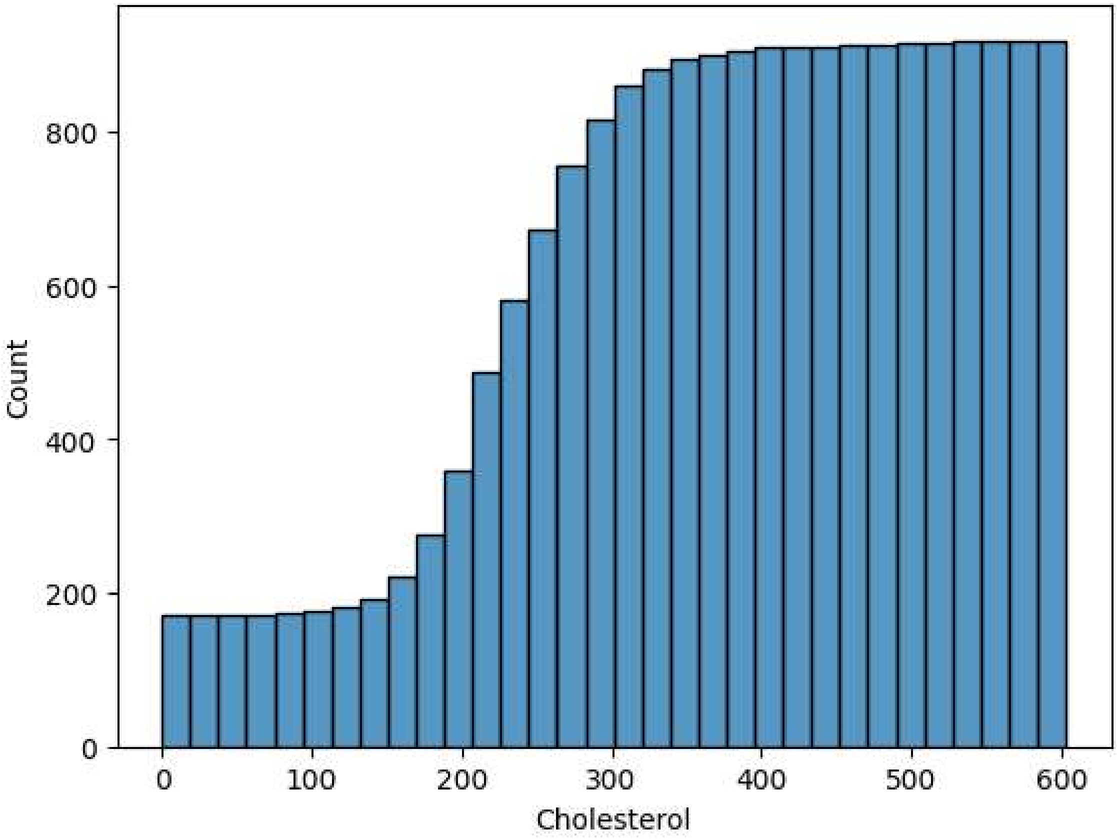
In [67]:

```
# Here is a histogram for cholesterol
sns.histplot(bp_df,x="Cholesterol",cumulative=True)
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/seaborn/_o
ldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be remove
d in a future version. Convert inf values to NaN before operating instead.
```

Out[67]:

```
<Axes: xlabel='Cholesterol', ylabel='Count'>
```



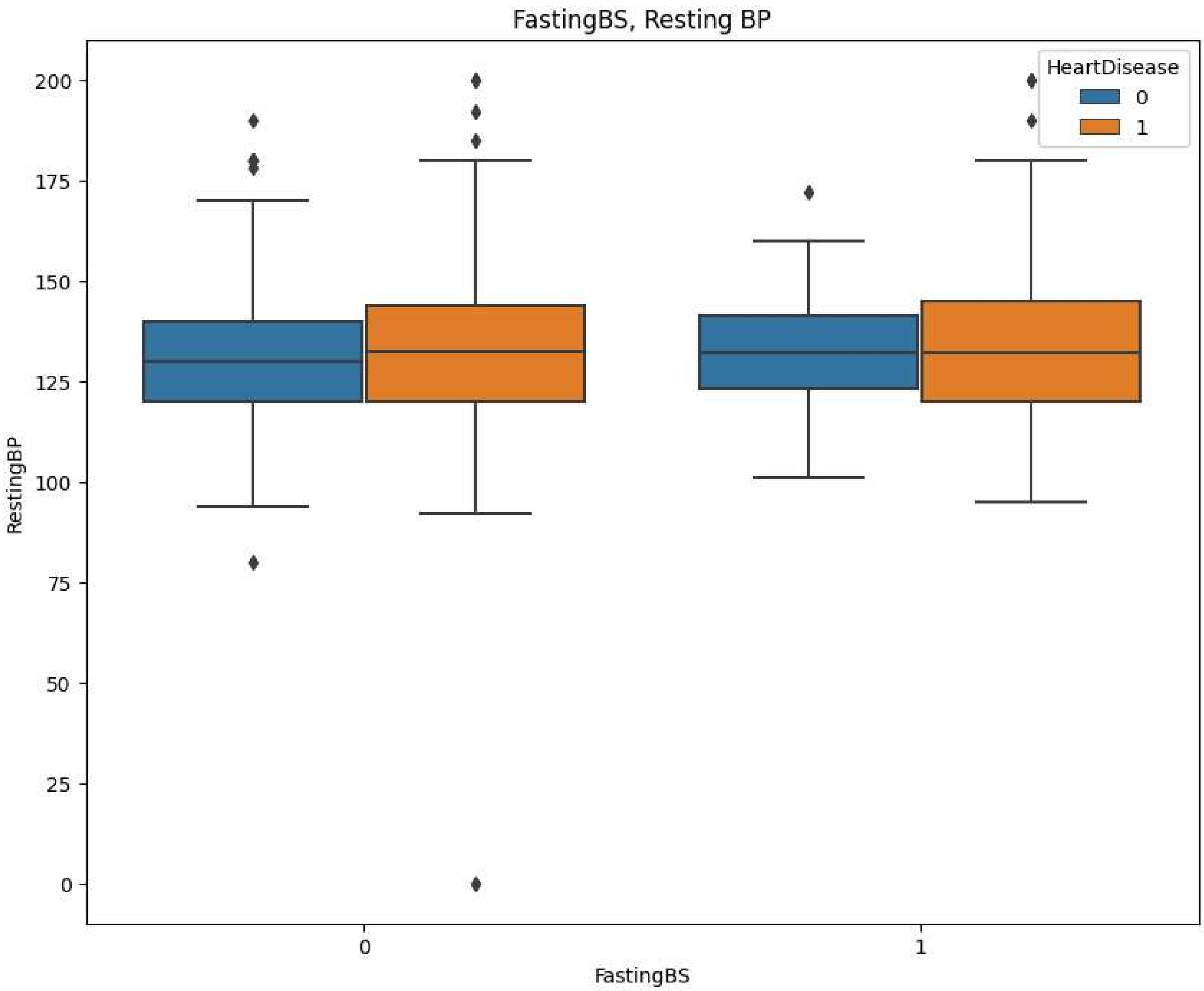
In [ ]:

```
ANSWER: Once you start getting up to 180-190 in cholesterol, then the number of peo
```

In [79]:

```
# f.) -Show An SNS Pairplot, the most informative version you can find, set the hue
# Heart Disease, try using at least one other variable as the Hue. Discuss what you
# are seeing in this plot

plt.figure(figsize=(10, 8))
sns.boxplot(data=bp_df, x='FastingBS', y='RestingBP', hue='HeartDisease', orient='v
plt.title("FastingBS, Resting BP")
plt.show()
```

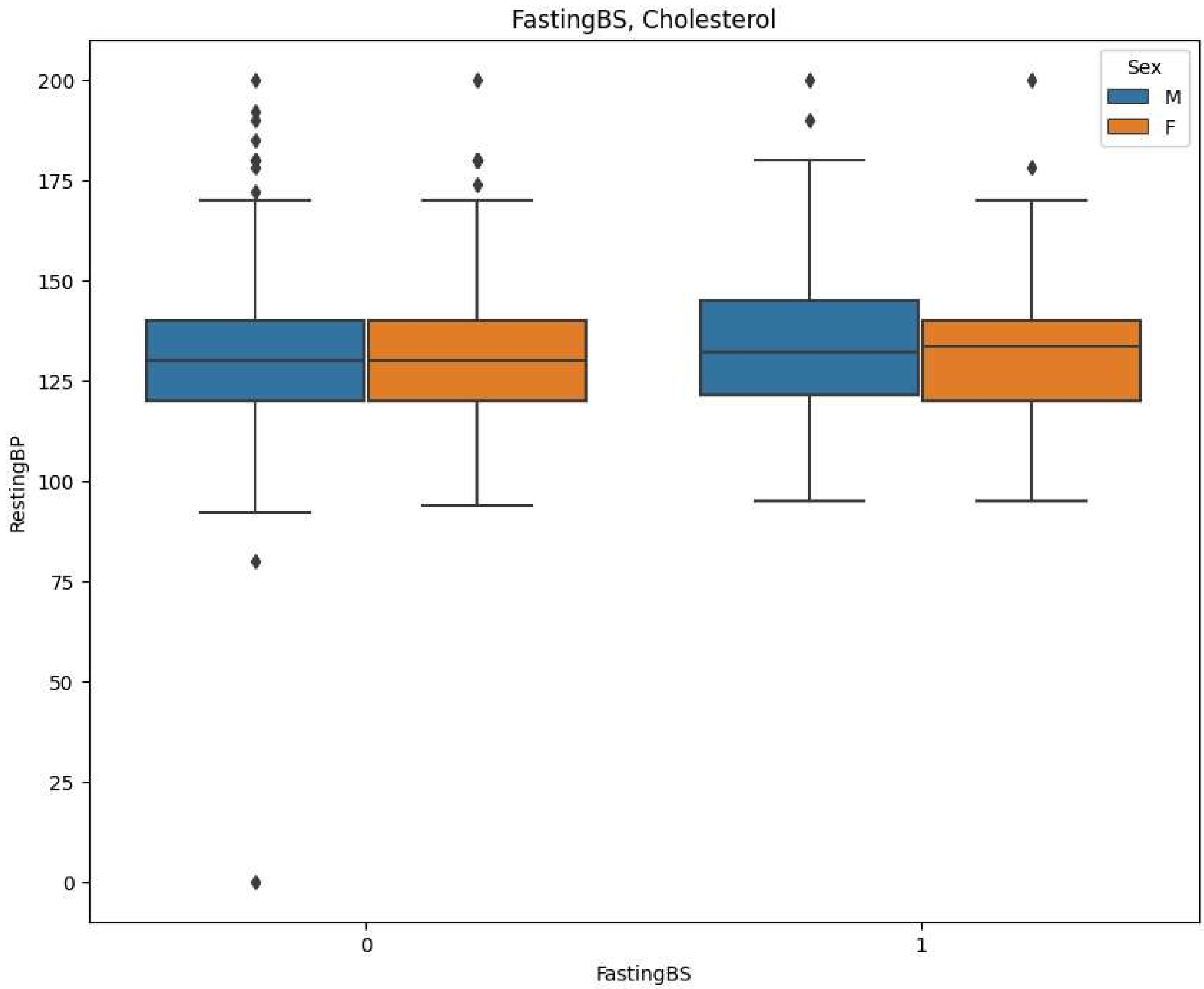


In [ ]:

```
ANSWER: When comparing if a patient was doing Fasting Blood Sugar had limited impac
Blood Pressure.
```

In [78]:

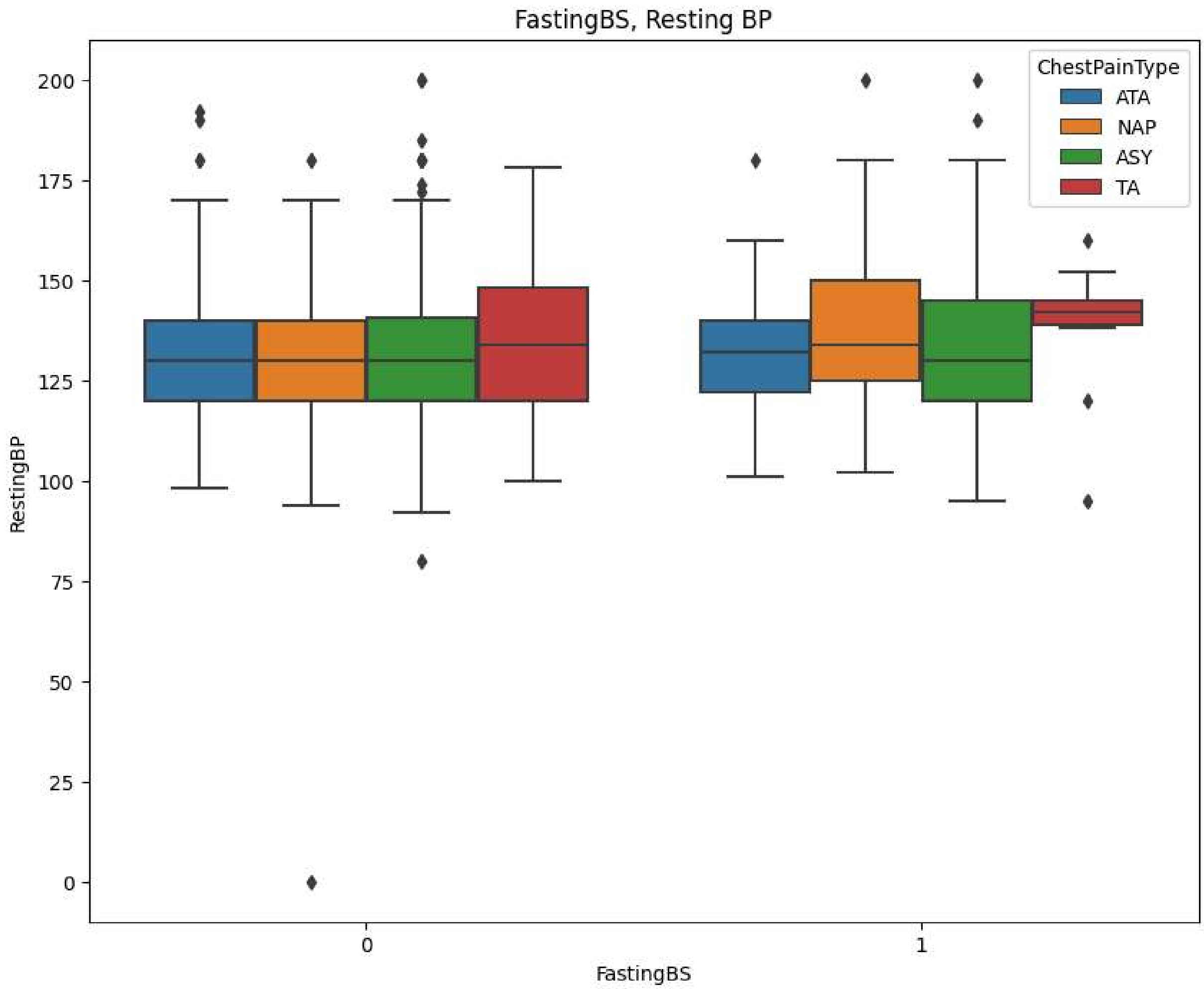
```
plt.figure(figsize=(10, 8))
sns.boxplot(data=bp_df, x='FastingBS', y='RestingBP', hue='Sex', orient='v')
plt.title("FastingBS, Resting BP")
plt.show()
```



```
In [ ]: ANSWER:  Doing the same analysis as above, there doesn't seem to be a major
        difference when evaluating Sex (i.e. Male versus Female)
```

```
In [84]: # g.) Create several useful or informative boxplots of continuous variables by cate
         #  Seaborn or PlotNine.   Find an interesting result or contrast among the variable
         #  what you think it means or implies

         plt.figure(figsize=(10, 8))
         sns.boxplot(data=bp_df, x='FastingBS', y='RestingBP', hue='ChestPainType', orient='
         plt.title("FastingBS, Resting BP")
         plt.show()
```



In [ ]:

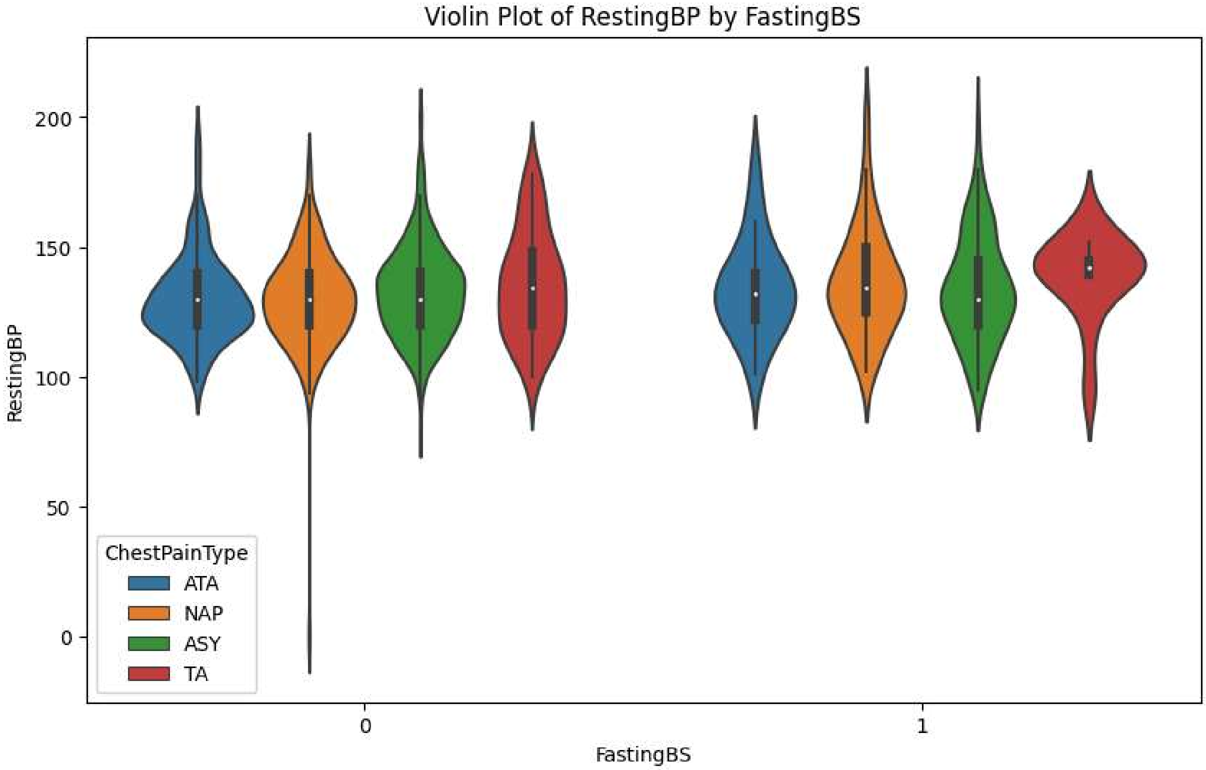
```
ANSWER:  I went through several boxplots and thought this one was interesting.  Peo
have fasting blood sugar had a higher resting blood pressure and also had typical a
```

In [93]:

```
# h.) Create violin plots of these same results


def create_violin_plot(bp_df, FastingBS, RestingBP, hue_col= 'ChestPainType'):
    plt.figure(figsize=(10, 6))
    sns.violinplot(x='FastingBS', y='RestingBP', hue='ChestPainType', data=bp_df)
    plt.title(f'Violin Plot of {RestingBP} by {FastingBS}')
    plt.show()

# Call the function to actually create and display the plot
create_violin_plot(bp_df, 'FastingBS', 'RestingBP', 'ChestPainType')
```


Violin Plot of RestingBP by FastingBS
RestingBP
200
150
100
50
0
FastingBS
0
1
ChestPainType
ATA
NAP
ASY
TA

```
In [ ]:
```