Mini-Project 1 Ryan Neighbor ANA 500

In [87]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
!pip install openpyxl
```

```
Requirement already satisfied: openpyxl in c:\users\nanoo\anaconda3\lib\site-package
s (3.1.5)
Requirement already satisfied: et-xmlfile in c:\users\nanoo\anaconda3\lib\site-packa
ges (from openpyxl) (1.1.0)
```

In [89]:
```
# 1a Acquire - identify data sets, retrieve data, query data
# Designate File Path
file_path = r"C:\Users\Nanoo\OneDrive\Desktop\ANA 500\heart disease.xlsx"

# Use read_excel for .xlsx files
df = pd.read_excel(file_path)
```

In [91]:
```
# 1b Acquire - identify data sets, retrieve data, query data
# Check First 5 entrys
df.head()
```

Out[91]:

	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	

In [93]:
```
# 1c Acquire - identify data sets, retrieve data, query data
# Missing Value Check
df.isnull().sum()
```

Out[93]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

In [95]:
```
# 1d Acquire - identify data sets, retrieve data, query data
# DF Info Check
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

In [97]:
```
# 1e Acquire - identify data sets, retrieve data, query data
# Dissect Categorical Variable Data Types by looping thhrough each Column and Print
categorical_cols = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slo

for col in categorical_cols:
    print(f"\nColumn: {col}")
    print(df[col].value_counts())
```

```
Column: Sex
Sex
M    725
F    193
Name: count, dtype: int64

Column: ChestPainType
ChestPainType
ASY    496
NAP    203
ATA    173
TA      46
Name: count, dtype: int64

Column: RestingECG
RestingECG
Normal    552
LVH       188
ST        178
Name: count, dtype: int64

Column: ExerciseAngina
ExerciseAngina
N    547
Y    371
Name: count, dtype: int64

Column: ST_Slope
ST_Slope
Flat    460
Up      395
Down     63
Name: count, dtype: int64
```

In [99]:

```python
# 1e Acquire - identify data sets, retrieve data, query data
# Dissect Categorical Variable Data Types (Autodetect Method)
categorical_cols = df.select_dtypes(include=['object', 'category']).columns

for col in categorical_cols:
    print(f"\nColumn: {col}")
    print(df[col].value_counts())
```

```
Column: Sex
Sex
M    725
F    193
Name: count, dtype: int64

Column: ChestPainType
ChestPainType
ASY    496
NAP    203
ATA    173
TA      46
Name: count, dtype: int64

Column: RestingECG
RestingECG
Normal    552
LVH       188
ST        178
Name: count, dtype: int64

Column: ExerciseAngina
ExerciseAngina
N    547
Y    371
Name: count, dtype: int64

Column: ST_Slope
ST_Slope
Flat    460
Up      395
Down     63
Name: count, dtype: int64
```

In [101…

```
# 1f Acquire - identify data sets, retrieve data, query data

# List View of Columns/Variables for better Understanding
print(df.columns.tolist())
```

```
['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingEC
G', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDisease']
```

In [103…

```
# 2a Prepare - explore and pre-process

# Summary Statistics
df.describe
```

Out[103…

```
<bound method NDFrame.describe of      Age Sex ChestPainType  RestingBP  Cholester
ol  FastingBS RestingECG  \
0     40   M           ATA        140          289          0     Normal
1     49   F           NAP        160          180          0     Normal
2     37   M           ATA        130          283          0         ST
3     48   F           ASY        138          214          0     Normal
4     54   M           NAP        150          195          0     Normal
..   ...  ..           ...        ...          ...        ...        ...
913   45   M            TA        110          264          0     Normal
914   68   M           ASY        144          193          1     Normal
915   57   M           ASY        130          131          0     Normal
916   57   F           ATA        130          236          0        LVH
917   38   M           NAP        138          175          0     Normal

     MaxHR ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0      172              N      0.0       Up             0
1      156              N      1.0     Flat             1
2       98              N      0.0       Up             0
3      108              Y      1.5     Flat             1
4      122              N      0.0       Up             0
..     ...            ...      ...      ...           ...
913    132              N      1.2     Flat             1
914    141              N      3.4     Flat             1
915    115              Y      1.2     Flat             1
916    174              N      0.0     Flat             1
917    173              N      0.0       Up             0

[918 rows x 12 columns]>
```

In [105…

```
# 2b Prepare – explore and pre-process

# Summary Statistics
df[['Age','Cholesterol','MaxHR', 'FastingBS', 'Oldpeak']].describe()
```

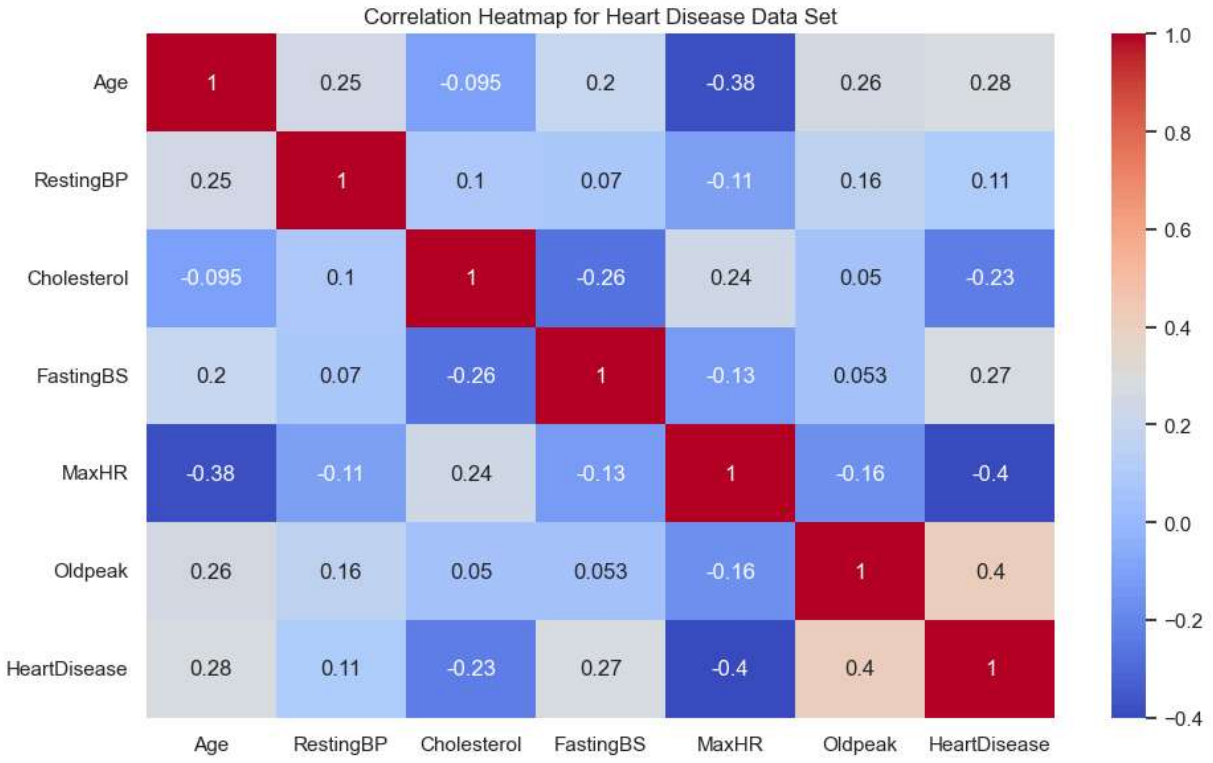
Out[105…

	Age	Cholesterol	MaxHR	FastingBS	Oldpeak
count	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	198.799564	136.809368	0.233115	0.887364
std	9.432617	109.384145	25.460334	0.423046	1.066570
min	28.000000	0.000000	60.000000	0.000000	-2.600000
25%	47.000000	173.250000	120.000000	0.000000	0.000000
50%	54.000000	223.000000	138.000000	0.000000	0.600000
75%	60.000000	267.000000	156.000000	0.000000	1.500000
max	77.000000	603.000000	202.000000	1.000000	6.200000

In [107…

```
# 2c Prepare – explore and pre-process
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#Correlation Heatmap for Continuous Data
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap for Heart Disease Data Set")
plt.tight_layout()
plt.show()
```



In [131…
```
# 2d Prepare – explore and pre-process

# Exploratory Filtering
df_filtered = df[(df['Age'] > 50) & (df['Cholesterol'] < 200)]
```

In [133…
```
# 2e Prepare – explore and pre-process
# Summary Statistics
df_filtered[['Age','Cholesterol']].describe()
```

Out[133…

	Age	Cholesterol
count	222.000000	222.000000
mean	59.243243	65.504505
std	5.823523	85.598862
min	51.000000	0.000000
25%	55.000000	0.000000
50%	59.000000	0.000000
75%	63.000000	170.750000
max	77.000000	199.000000

In [135…
```
# Remove rows where cholesterol is zero

df_clean = df[df['Cholesterol'] != 0]
```

In [137…
```
# Summary stats for cleaned cholesterol

print(df_clean['Cholesterol'].describe())
```

```
count    746.000000
mean     244.635389
std       59.153524
min       85.000000
25%      207.250000
50%      237.000000
75%      275.000000
max      603.000000
Name: Cholesterol, dtype: float64
```

In [139…
```
# 2f Prepare - explore and pre-process

# Normalize Continuous Values
# List of numeric columns to normalize
numeric_cols = ['Age', 'Cholesterol', 'MaxHR', 'FastingBS', 'Oldpeak']

# Apply normalization
df_normalized = df_clean[numeric_cols].apply(lambda x: round((x - x.min()) / (x.max

# Added normalized columns back to the original DataFrame
for col in df_normalized.columns:
    df[f'{col}_normalized'] = df_normalized[col]
```

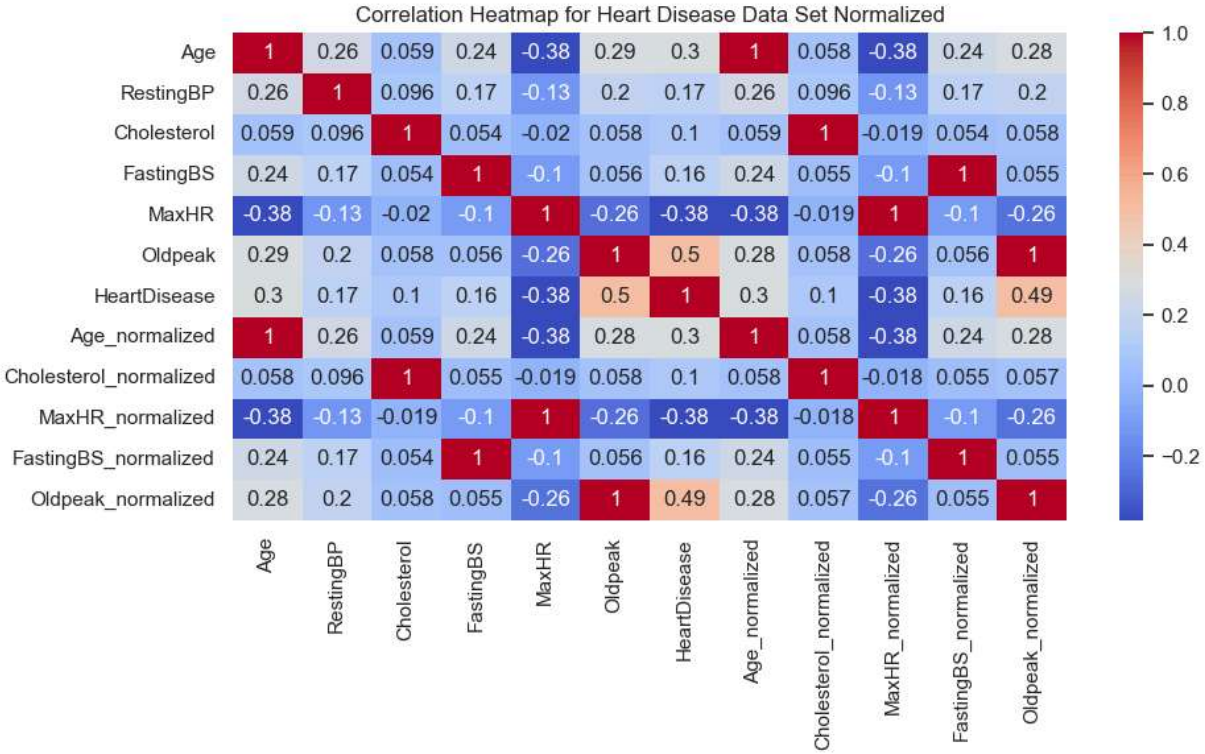
In [141…
```
# 2e Prepare - explore and pre-process

# Summary Statistics Normalized Dataframe
df_normalized.describe()
```

Out[141…

	Age	Cholesterol	MaxHR	FastingBS	Oldpeak
count	746.000000	746.000000	746.000000	746.000000	746.000000
mean	0.507627	0.308190	0.535094	0.167560	0.159960
std	0.193804	0.114254	0.184083	0.373726	0.168317
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.370000	0.240000	0.400000	0.000000	0.020000
50%	0.530000	0.290000	0.530000	0.000000	0.100000
75%	0.630000	0.370000	0.680000	0.000000	0.250000
max	1.000000	1.000000	1.000000	1.000000	1.000000

In [143…
```
# 2f Prepare - explore and pre-process
```

```
#Correlation Heatmap for Normalized Data
plt.figure(figsize=(10,6))
sns.heatmap(df_clean.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap for Heart Disease Data Set Normalized")
plt.tight_layout()
plt.show()
```



When comparing both non & normalized data for correlation: Age vs MaxHR shows promise Heart Disease vs MaxHR shows promise Cholesterol vs fastingBS shows promise Cholesterol vs MaxHR shows promise

In [146…

```
# 2g Prepare – explore and pre-process


# Bin Age into 4 equal groups
df_clean['AgeGroup'] = pd.cut(df['Age'], bins=[0, 25, 50, 75, 100],
                        labels=['0-25', '26-50', '51-75', '76-100'],
                        right=True)

# Group by AgeGroups and summarize Cholesterol

df_clean.groupby('AgeGroup')['Cholesterol'].agg(['mean', 'max', 'min', 'count'])
```

```
C:\Users\Nanoo\AppData\Local\Temp\ipykernel_37292\825285292.py:5: SettingWithCopyWar
ning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\Nanoo\AppData\Local\Temp\ipykernel_37292\825285292.py:11: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or observed=True t
o adopt the future default and silence this warning.
```

Out[146…

AgeGroup	mean	max	min	count
0–25	NaN	NaN	NaN	0
26–50	239.099291	529.0	117.0	282
51–75	248.450000	603.0	85.0	460
76–100	196.250000	304.0	113.0	4

In [148…

```python
# Correlation matrix for numeric columns only
correlation_table = df_clean.corr(numeric_only=True)

# Display the table
print(correlation_table)
```

```
                            Age  RestingBP  Cholesterol  FastingBS     MaxHR  \
Age                    1.000000   0.259865     0.058758   0.241338 -0.382112
RestingBP              0.259865   1.000000     0.095939   0.173765 -0.125774
Cholesterol            0.058758   0.095939     1.000000   0.054012 -0.019856
FastingBS              0.241338   0.173765     0.054012   1.000000 -0.102710
MaxHR                 -0.382112  -0.125774    -0.019856  -0.102710  1.000000
Oldpeak                0.286006   0.198575     0.058488   0.055568 -0.259533
HeartDisease           0.298617   0.173242     0.103866   0.160594 -0.377212
Age_normalized         0.999891   0.259282     0.058620   0.241411 -0.381745
Cholesterol_normalized 0.057729   0.095589     0.999673   0.054893 -0.019159
MaxHR_normalized      -0.381828  -0.125503    -0.018615  -0.102954  0.999887
FastingBS_normalized   0.241338   0.173765     0.054012   1.000000 -0.102710
Oldpeak_normalized     0.284762   0.198391     0.057586   0.054520 -0.256571

                         Oldpeak  HeartDisease  Age_normalized  \
Age                     0.286006      0.298617        0.999891
RestingBP               0.198575      0.173242        0.259282
Cholesterol             0.058488      0.103866        0.058620
FastingBS               0.055568      0.160594        0.241411
MaxHR                  -0.259533     -0.377212       -0.381745
Oldpeak                 1.000000      0.495696        0.284949
HeartDisease            0.495696      1.000000        0.297437
Age_normalized          0.284949      0.297437        1.000000
Cholesterol_normalized  0.057765      0.103287        0.057606
MaxHR_normalized       -0.259723     -0.377171       -0.381448
FastingBS_normalized    0.055568      0.160594        0.241411
Oldpeak_normalized      0.999854      0.494206        0.283682

                        Cholesterol_normalized  MaxHR_normalized  \
Age                                   0.057729         -0.381828
RestingBP                             0.095589         -0.125503
Cholesterol                           0.999673         -0.018615
FastingBS                             0.054893         -0.102954
MaxHR                                -0.019159          0.999887
Oldpeak                               0.057765         -0.259723
HeartDisease                          0.103287         -0.377171
Age_normalized                        0.057606         -0.381448
Cholesterol_normalized                1.000000         -0.017935
MaxHR_normalized                     -0.017935          1.000000
FastingBS_normalized                  0.054893         -0.102954
Oldpeak_normalized                    0.056868         -0.256768

                        FastingBS_normalized  Oldpeak_normalized
Age                                 0.241338            0.284762
RestingBP                           0.173765            0.198391
Cholesterol                         0.054012            0.057586
FastingBS                           1.000000            0.054520
MaxHR                              -0.102710           -0.256571
Oldpeak                             0.055568            0.999854
HeartDisease                        0.160594            0.494206
Age_normalized                      0.241411            0.283682
Cholesterol_normalized              0.054893            0.056868
MaxHR_normalized                   -0.102954           -0.256768
FastingBS_normalized                1.000000            0.054520
Oldpeak_normalized                  0.054520            1.000000
```

Analyze Data

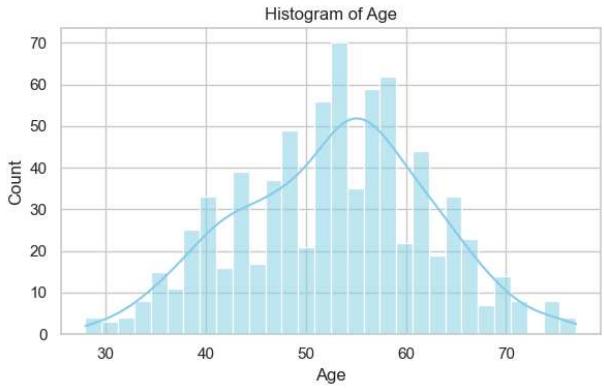
In [151…
```
import seaborn as sns
```

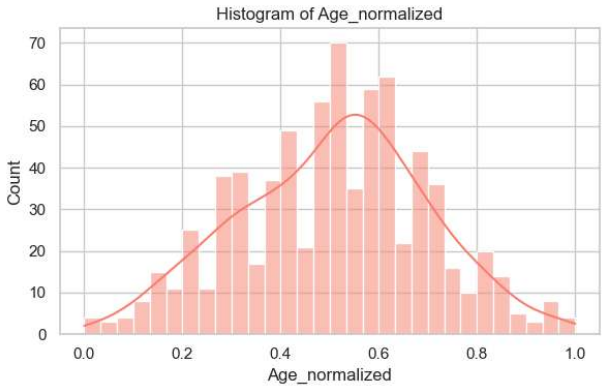
In [153…
```
df_clean.info()
```

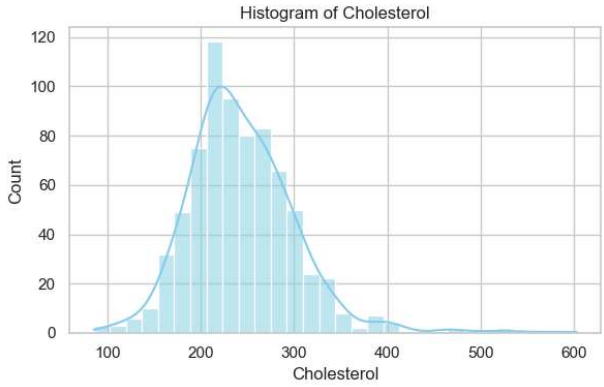
```
<class 'pandas.core.frame.DataFrame'>
Index: 746 entries, 0 to 917
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     746 non-null    int64
 1   Sex                     746 non-null    object
 2   ChestPainType           746 non-null    object
 3   RestingBP               746 non-null    int64
 4   Cholesterol             746 non-null    int64
 5   FastingBS               746 non-null    int64
 6   RestingECG              746 non-null    object
 7   MaxHR                   746 non-null    int64
 8   ExerciseAngina          746 non-null    object
 9   Oldpeak                 746 non-null    float64
 10  ST_Slope                746 non-null    object
 11  HeartDisease            746 non-null    int64
 12  Age_normalized          746 non-null    float64
 13  Cholesterol_normalized  746 non-null    float64
 14  MaxHR_normalized        746 non-null    float64
 15  FastingBS_normalized    746 non-null    float64
 16  Oldpeak_normalized      746 non-null    float64
 17  AgeGroup                746 non-null    category
dtypes: category(1), float64(6), int64(6), object(5)
memory usage: 122.0+ KB
```

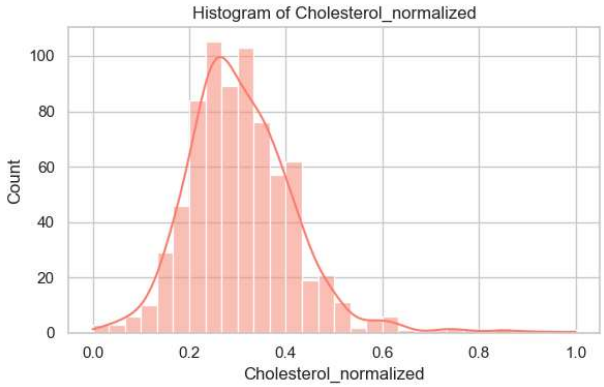
In [155…
```
# Set plot style
sns.set(style="whitegrid")

# Variables to plot
variables = ['Age', 'Cholesterol', 'MaxHR', 'Oldpeak']

# Histograms considering original and normalized variables
for var in variables:
    fig, axes = plt.subplots(1, 2, figsize=(12, 4))

    # Original variable
    sns.histplot(df_clean[var], bins=30, kde=True, ax=axes[0], color='skyblue')
    axes[0].set_title(f'Histogram of {var}')
    axes[0].set_xlabel(var)

    # Normalized variable
    norm_var = f"{var}_normalized"
    if norm_var in df_clean.columns:
        sns.histplot(df_clean[norm_var], bins=30, kde=True, ax=axes[1], color='salm
        axes[1].set_title(f'Histogram of {norm_var}')
        axes[1].set_xlabel(norm_var)
    else:
        axes[1].text(0.5, 0.5, f'{norm_var} not found', ha='center', va='center')
```

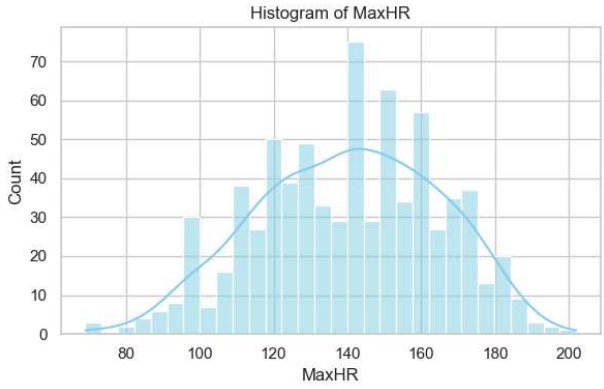
```
        axes[1].set_title(f'{norm_var} Missing')
        axes[1].axis('off')

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

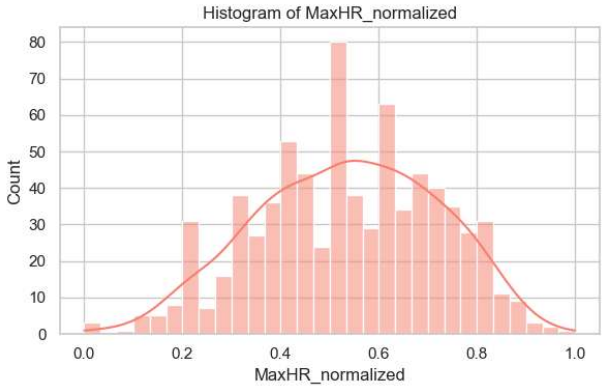


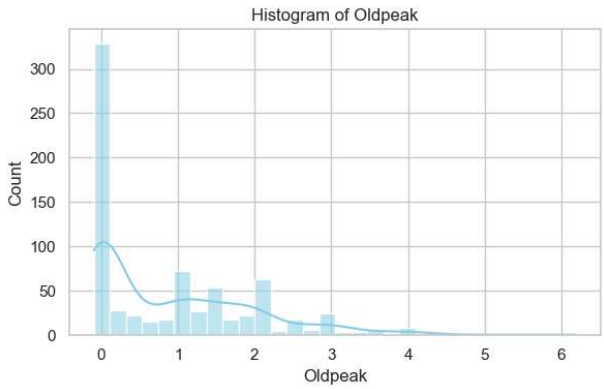


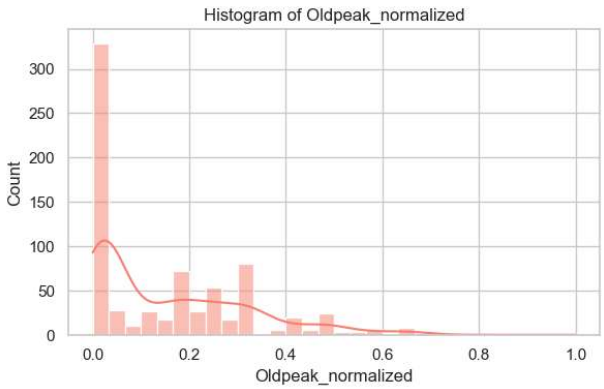










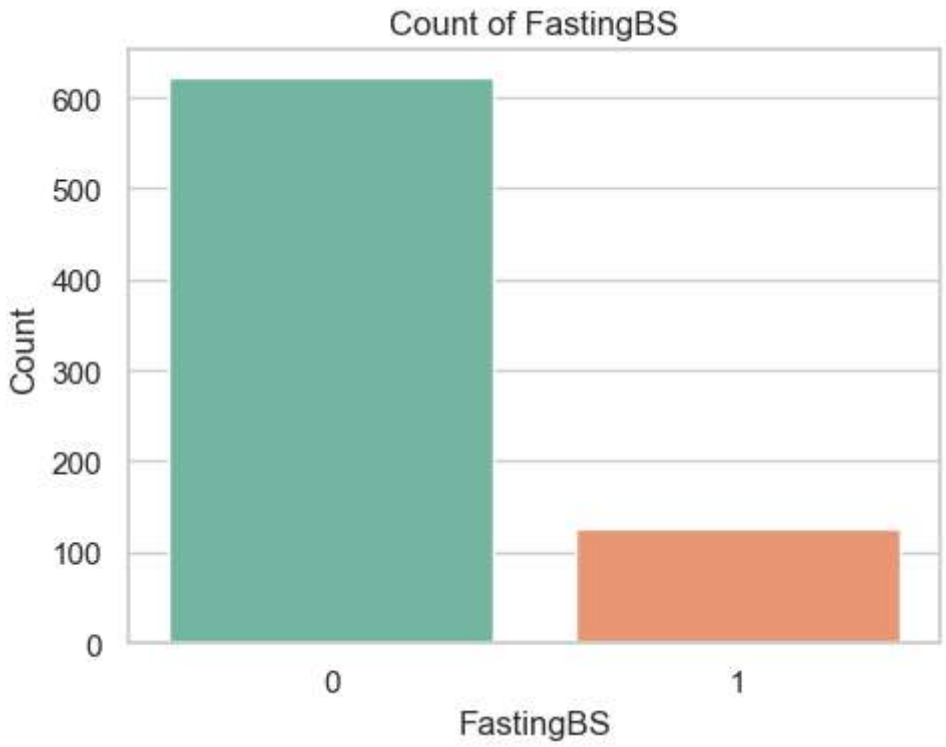


In [157…

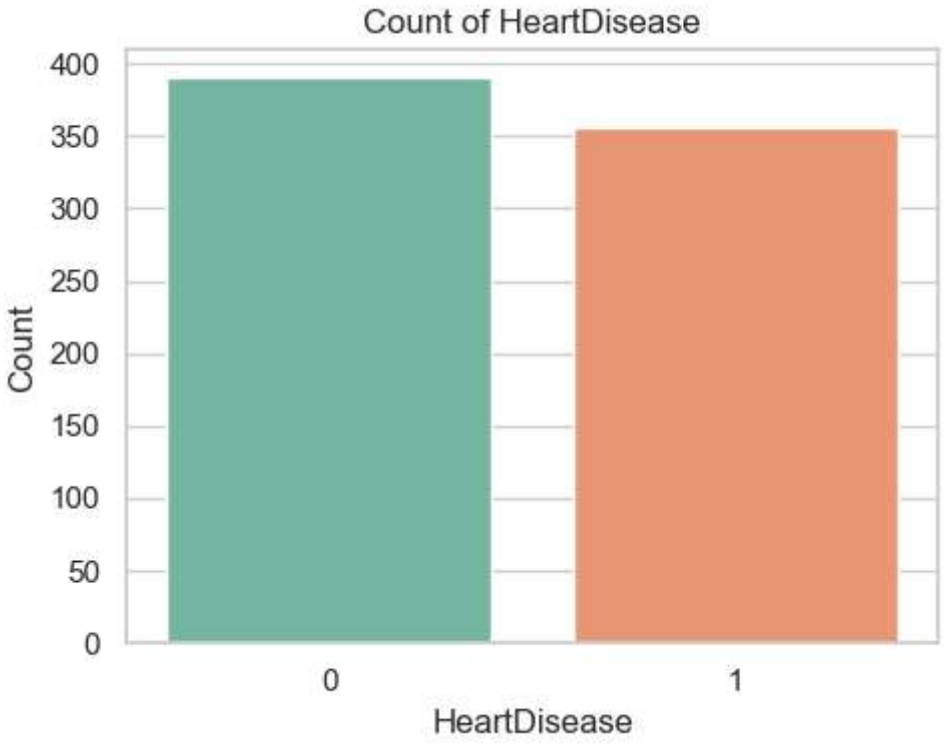
```
# Binary categorical variables (bar plot)
binary_vars = ['FastingBS', 'HeartDisease']
```

```
for var in binary_vars:
    plt.figure(figsize=(5, 4))
    sns.countplot(x=var, data=df_clean, palette='Set2')
    plt.title(f'Count of {var}')
    plt.xlabel(var)
    plt.ylabel('Count')
    plt.tight_layout()
    plt.show()
```

```
C:\Users\Nanoo\AppData\Local\Temp\ipykernel_37292\4214232699.py:6: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

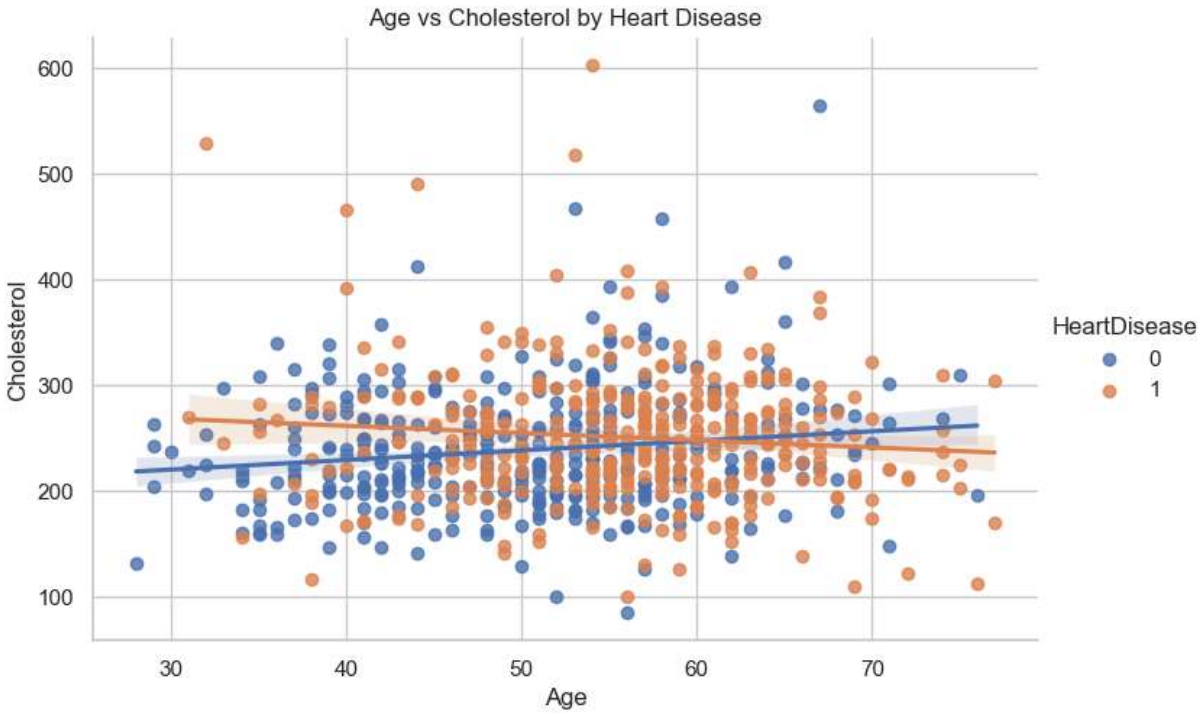


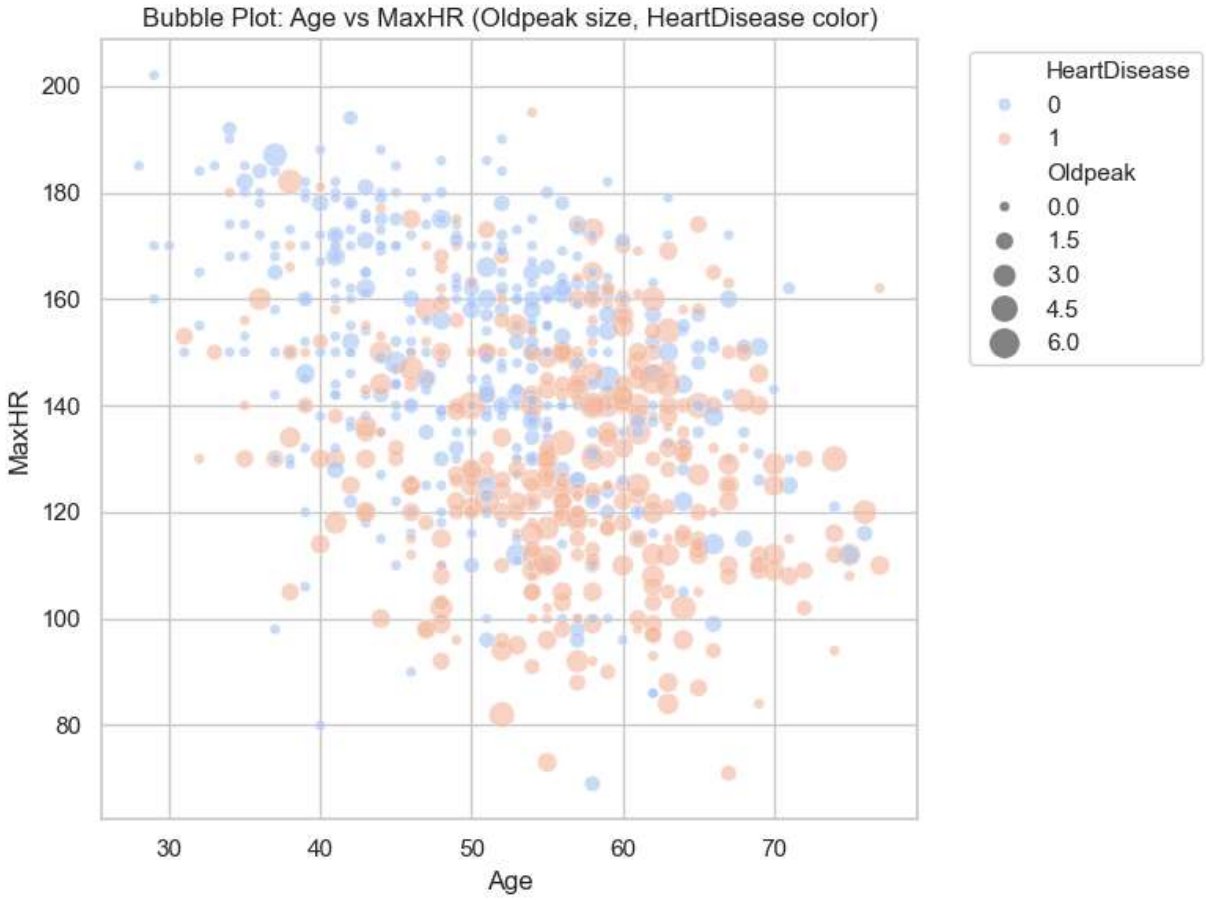

```
C:\Users\Nanoo\AppData\Local\Temp\ipykernel_37292\4214232699.py:6: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```



In [159…

```
# Scatter plot with regression line
sns.lmplot(x='Age', y='Cholesterol', data=df_clean, hue='HeartDisease', aspect=1.5)
plt.title('Age vs Cholesterol by Heart Disease')
plt.show()

# Bubble plot: Age vs MaxHR, bubble size = Oldpeak, color = HeartDisease
plt.figure(figsize=(8, 6))
sns.scatterplot(
    x='Age', y='MaxHR',
    size='Oldpeak', hue='HeartDisease',
    data=df_clean, sizes=(20, 200), alpha=0.6, palette='coolwarm'
)
plt.title('Bubble Plot: Age vs MaxHR (Oldpeak size, HeartDisease color)')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2)
plt.tight_layout()
plt.show()
```





In [161…

```
!pip install -U plotly
import plotly.express as px
import statsmodels.api as sm
!pip install -U kaleido
```

```
Requirement already satisfied: plotly in c:\users\nanoo\anaconda3\lib\site-packages
(6.3.0)
Requirement already satisfied: narwhals>=1.15.1 in c:\users\nanoo\anaconda3\lib\site
-packages (from plotly) (2.1.2)
Requirement already satisfied: packaging in c:\users\nanoo\anaconda3\lib\site-packag
es (from plotly) (24.1)
Requirement already satisfied: kaleido in c:\users\nanoo\anaconda3\lib\site-packages
(1.0.0)
Requirement already satisfied: choreographer>=1.0.5 in c:\users\nanoo\anaconda3\lib
\site-packages (from kaleido) (1.0.9)
Requirement already satisfied: logistro>=1.0.8 in c:\users\nanoo\anaconda3\lib\site-
packages (from kaleido) (1.1.0)
Requirement already satisfied: orjson>=3.10.15 in c:\users\nanoo\anaconda3\lib\site-
packages (from kaleido) (3.11.2)
```

In [162…

```python
import plotly.express as px
import statsmodels.api as sm


# Fit regression lines separately for each HeartDisease group
def add_regression(df, x, y):
    lines = []
    for label, group in df.groupby('HeartDisease'):
        model = sm.OLS(group[y], sm.add_constant(group[x])).fit()
        pred = model.predict(sm.add_constant(group[x]))
        lines.append(pd.DataFrame({
            x: group[x],
            'Predicted': pred,
            'HeartDisease': label
        }))
    return pd.concat(lines)

# Add regression predictions
df_reg = add_regression(df_clean, 'Age', 'Cholesterol')

# Scatter plot with regression overlay
fig = px.scatter(
    df_clean, x='Age', y='Cholesterol',
    color='HeartDisease',
    title='Age vs Cholesterol by Heart Disease',
    labels={'HeartDisease': 'Heart Disease'},
    hover_data=['Age', 'Cholesterol']
)

# Add regression lines
for label in df_reg['HeartDisease'].unique():
    group = df_reg[df_reg['HeartDisease'] == label]
    fig.add_scatter(x=group['Age'], y=group['Predicted'],
                    mode='lines', name=f'Regression (HD={label})')

fig.update_layout(legend_title_text='Heart Disease')
fig.write_image("age_vs_cholesterol.png")  # Save for PowerPoint
fig.show()
```

In [171…

```
# Normalized contiuous variables
normalized_vars = [
    'Cholesterol_normalized',
    'MaxHR_normalized',
    'Oldpeak_normalized'
]

# Melt the dataframe to long format
df_long = df_clean[['Age'] + normalized_vars].melt(
    id_vars='Age',
    value_vars=normalized_vars,
    var_name='Metric',
    value_name='Normalized_Value'
)

# Clean up metric names
df_long['Metric'] = df_long['Metric'].str.replace('_normalized', '')

# Create scatter plot
fig = px.scatter(
    df_long,
    x='Age',
    y='Normalized_Value',
    color='Metric',
    title='Normalized Health Metrics Over Age',
    labels={'Normalized_Value': 'Normalized Value (0-1)'},
    opacity=0.7,
    template='plotly_white'
)

fig.write_image("normalized_health_metrics_scatter.png")
fig.show()
```

In [174…

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score
```

In [182…

```
# Logistic Regression
# Define features and target
X = df_clean[['Cholesterol_normalized', 'MaxHR_normalized', 'Oldpeak_normalized']]
y = df_clean['HeartDisease']

# Split into train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [184…

```
# Initialize and train
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[184…

LogisticRegression

```
LogisticRegression()
```

In [186…

```
# Predict and evaluate p\Performance
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]

print(classification_report(y_test, y_pred))
print("AUC-ROC:", roc_auc_score(y_test, y_prob))
```

```
              precision    recall  f1-score   support

           0       0.71      0.85      0.77        71
           1       0.83      0.70      0.76        79

    accuracy                           0.77       150
   macro avg       0.77      0.77      0.77       150
weighted avg       0.78      0.77      0.77       150

AUC-ROC: 0.8541629523979318
```

In [188…

```
# Coefficients and odds ratios
coeffs = pd.Series(model.coef_[0], index=X.columns)
odds_ratios = coeffs.apply(lambda x: round(np.exp(x), 2))

print("Coefficients:\n", coeffs)
print("Odds Ratios:\n", odds_ratios)
```

```
Coefficients:
 Cholesterol_normalized    0.668682
MaxHR_normalized          -3.150142
Oldpeak_normalized         4.687230
dtype: float64
Odds Ratios:
 Cholesterol_normalized      1.95
MaxHR_normalized             0.04
Oldpeak_normalized         108.55
dtype: float64
```

In [ ]: