Operations:

a. Data cleaning

b. Data integration

c. Data transformation

d. Error correcting

e. Data Model Backend

```
import pandas as pd
import numpy as np

df = pd.read_csv('airquality_data.csv', encoding='cp1252')
```

```
C:\Users\Gayatri Tagalpallewa\AppData\Local\Temp\
ipykernel_21076\2182913842.py:1: DtypeWarning: Columns (0) have mixed
types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('airquality_data.csv', encoding='cp1252')
```

```
df.head()
```

```
  stn_code       sampling_date            state   location agency  \
0    150.0  February - M021990  Andhra Pradesh  Hyderabad    NaN
1    151.0  February - M021990  Andhra Pradesh  Hyderabad    NaN
2    152.0  February - M021990  Andhra Pradesh  Hyderabad    NaN
3    150.0     March - M031990  Andhra Pradesh  Hyderabad    NaN
4    151.0     March - M031990  Andhra Pradesh  Hyderabad    NaN

                                 type  so2   no2  rspm  spm  \
0  Residential, Rural and other Areas  4.8  17.4   NaN  NaN
1                     Industrial Area  3.1   7.0   NaN  NaN
2  Residential, Rural and other Areas  6.2  28.5   NaN  NaN
3  Residential, Rural and other Areas  6.3  14.7   NaN  NaN
4                     Industrial Area  4.7   7.5   NaN  NaN

  location_monitoring_station  pm2_5        date
0                         NaN    NaN  1990-02-01
1                         NaN    NaN  1990-02-01
2                         NaN    NaN  1990-02-01
3                         NaN    NaN  1990-03-01
4                         NaN    NaN  1990-03-01
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435742 entries, 0 to 435741
```

```
Data columns (total 13 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   stn_code                      291665 non-null  object 
 1   sampling_date                 435739 non-null  object 
 2   state                         435742 non-null  object 
 3   location                      435739 non-null  object 
 4   agency                        286261 non-null  object 
 5   type                          430349 non-null  object 
 6   so2                           401096 non-null  float64
 7   no2                           419509 non-null  float64
 8   rspm                          395520 non-null  float64
 9   spm                           198355 non-null  float64
 10  location_monitoring_station  408251 non-null  object 
 11  pm2_5                         9314 non-null    float64
 12  date                          435735 non-null  object 
dtypes: float64(5), object(8)
memory usage: 43.2+ MB

df.columns

Index(['stn_code', 'sampling_date', 'state', 'location', 'agency',
'type',
       'so2', 'no2', 'rspm', 'spm', 'location_monitoring_station',
'pm2_5',
       'date'],
      dtype='object')
```

Data Cleaning

```
# Change data type from float64 to float32 for Space Complexity
df['so2'] = df['so2'].astype('float32')
df['no2'] = df['no2'].astype('float32')
df['rspm'] = df['rspm'].astype('float32')
df['spm'] = df['spm'].astype('float32')
df['date'] = df['date'].astype('string')

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 435742 entries, 0 to 435741
Data columns (total 13 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   stn_code                      291665 non-null  object 
 1   sampling_date                 435739 non-null  object 
 2   state                         435742 non-null  object 
 3   location                      435739 non-null  object 
 4   agency                        286261 non-null  object 
 5   type                          430349 non-null  object 
 6   so2                           401096 non-null  float32
 7   no2                           419509 non-null  float32
 8   rspm                          395520 non-null  float32
 9   spm                           198355 non-null  float32
 10  location_monitoring_station   408251 non-null  object
 11  pm2_5                         9314 non-null    float64
 12  date                          435735 non-null  string
dtypes: float32(4), float64(1), object(7), string(1)
memory usage: 36.6+ MB
```

```
df=df.drop_duplicates()
```

```
df.isna().sum()
```

```
stn_code                       144077
sampling_date                       3
state                               0
location                            3
agency                         149466
type                             5357
so2                             34632
no2                             16222
rspm                            40035
spm                            236908
location_monitoring_station     27303
pm2_5                          425754
date                                7
dtype: int64
```

```
percent_missing = df.isnull().sum() * 100 / len(df)
```

```
percent_missing.sort_values(ascending=False)
```

```
pm2_5                          97.859185
spm                            54.453097
agency                         34.354630
stn_code                       33.115973
rspm                            9.202010
so2                             7.960135
location_monitoring_station     6.275571
no2                             3.728613
type                            1.231302
date                            0.001609
sampling_date                   0.000690
location                        0.000690
state                           0.000000
dtype: float64
```

```
df=df.drop(['stn_code',
'agency','sampling_date','location_monitoring_station','pm2_5'], axis
= 1)
```

```
df.head()
```

```
            state   location                               type  so2
no2  \
0  Andhra Pradesh  Hyderabad  Residential, Rural and other Areas  4.8
17.4
1  Andhra Pradesh  Hyderabad                    Industrial Area  3.1
7.0
2  Andhra Pradesh  Hyderabad  Residential, Rural and other Areas  6.2
28.5
3  Andhra Pradesh  Hyderabad  Residential, Rural and other Areas  6.3
14.7
4  Andhra Pradesh  Hyderabad                    Industrial Area  4.7
7.5

   rspm  spm        date
0   NaN  NaN  1990-02-01
1   NaN  NaN  1990-02-01
2   NaN  NaN  1990-02-01
3   NaN  NaN  1990-03-01
4   NaN  NaN  1990-03-01
```

```
df.columns
```

```
Index(['state', 'location', 'type', 'so2', 'no2', 'rspm', 'spm',
'date'], dtype='object')
```

```
col_var = ['state', 'location', 'type','date']
col_num = ['so2','no2','rspm','spm']

for col in df.columns:
    if df[col].dtype == 'object' or df[col].dtype == 'string':
        df[col] = df[col].fillna(df[col].mode()[0])
    else:
        df[col] = df[col].fillna(df[col].mean())

df.isna().sum()
```

```
state       0
location    0
type        0
so2         0
no2         0
rspm        0
spm         0
date        0
dtype: int64
```

```
df
```

```
                              state   location  \
0                    Andhra Pradesh  Hyderabad
1                    Andhra Pradesh  Hyderabad
2                    Andhra Pradesh  Hyderabad
3                    Andhra Pradesh  Hyderabad
4                    Andhra Pradesh  Hyderabad
...                             ...        ...
435737                  West Bengal   ULUBERIA
435738                  West Bengal   ULUBERIA
435739  andaman-and-nicobar-islands   Guwahati
435740                  Lakshadweep   Guwahati
435741                      Tripura   Guwahati

                                      type        so2        no2
rspm  \
0       Residential, Rural and other Areas   4.800000  17.400000
108.871712
1                          Industrial Area   3.100000   7.000000
108.871712
2       Residential, Rural and other Areas   6.200000  28.500000
108.871712
3       Residential, Rural and other Areas   6.300000  14.700000
108.871712
4                          Industrial Area   4.700000   7.500000
108.871712
...                                    ...        ...        ...
...
435737                               RIRUO  22.000000  50.000000
143.000000
435738                               RIRUO  20.000000  46.000000
171.000000
435739  Residential, Rural and other Areas  10.830467  25.823299
108.871712
435740  Residential, Rural and other Areas  10.830467  25.823299
108.871712
435741  Residential, Rural and other Areas  10.830467  25.823299
108.871712

               spm        date
0       220.774796  1990-02-01
1       220.774796  1990-02-01
2       220.774796  1990-02-01
3       220.774796  1990-03-01
4       220.774796  1990-03-01
...            ...         ...
435737  220.774796  2015-12-24
435738  220.774796  2015-12-29
435739  220.774796  2015-03-19
435740  220.774796  2015-03-19
435741  220.774796  2015-03-19
```

```
[435068 rows x 8 columns]

df.isna().sum()

state       0
location    0
type        0
so2         0
no2         0
rspm        0
spm         0
date        0
dtype: int64
```

Data integration

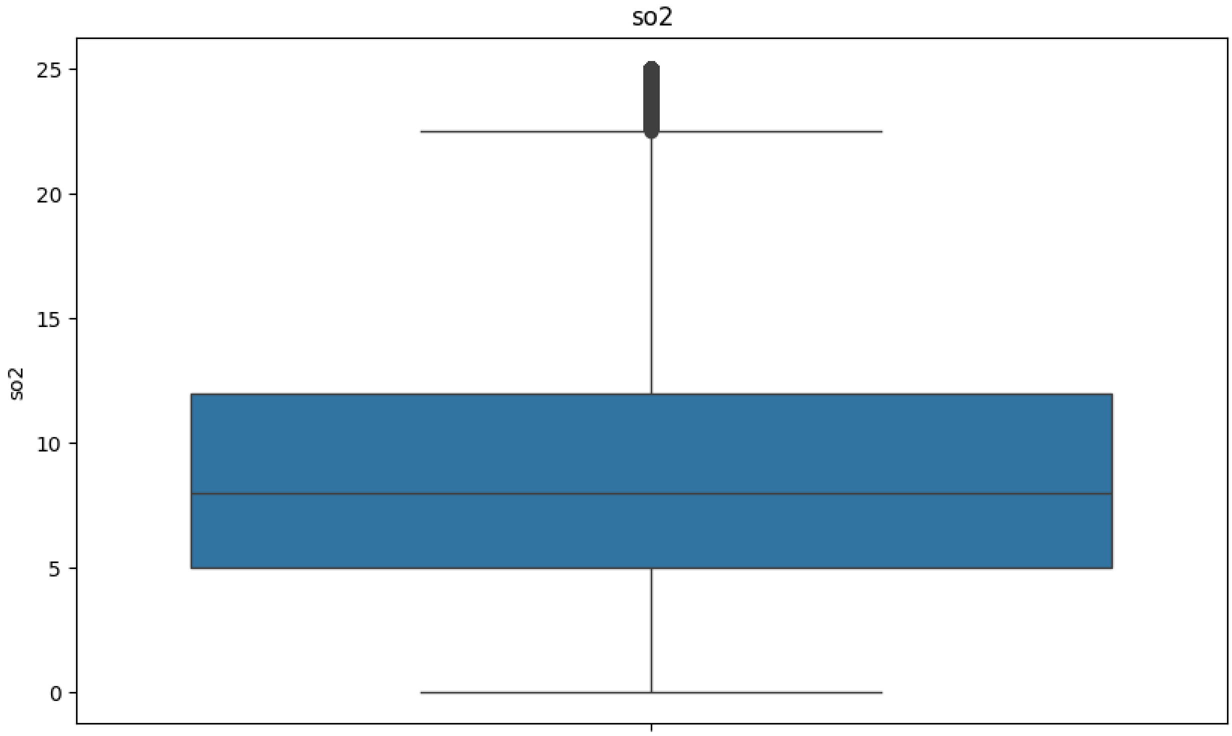
```
subSet1 = df[['state', 'type']]
subSet2 = df[['state','location']]

subSet1.head()

            state                                type
0  Andhra Pradesh  Residential, Rural and other Areas
1  Andhra Pradesh                     Industrial Area
2  Andhra Pradesh  Residential, Rural and other Areas
3  Andhra Pradesh  Residential, Rural and other Areas
4  Andhra Pradesh                     Industrial Area

subSet2.head()

            state   location
0  Andhra Pradesh  Hyderabad
1  Andhra Pradesh  Hyderabad
2  Andhra Pradesh  Hyderabad
3  Andhra Pradesh  Hyderabad
4  Andhra Pradesh  Hyderabad

concatenated_df = pd.concat([subSet1, subSet2], axis=1)

concatenated_df

                              state                                
type  \
0                    Andhra Pradesh  Residential, Rural and other 
Areas   
1                    Andhra Pradesh                     Industrial 
Area   
2                    Andhra Pradesh  Residential, Rural and other 
Areas
```

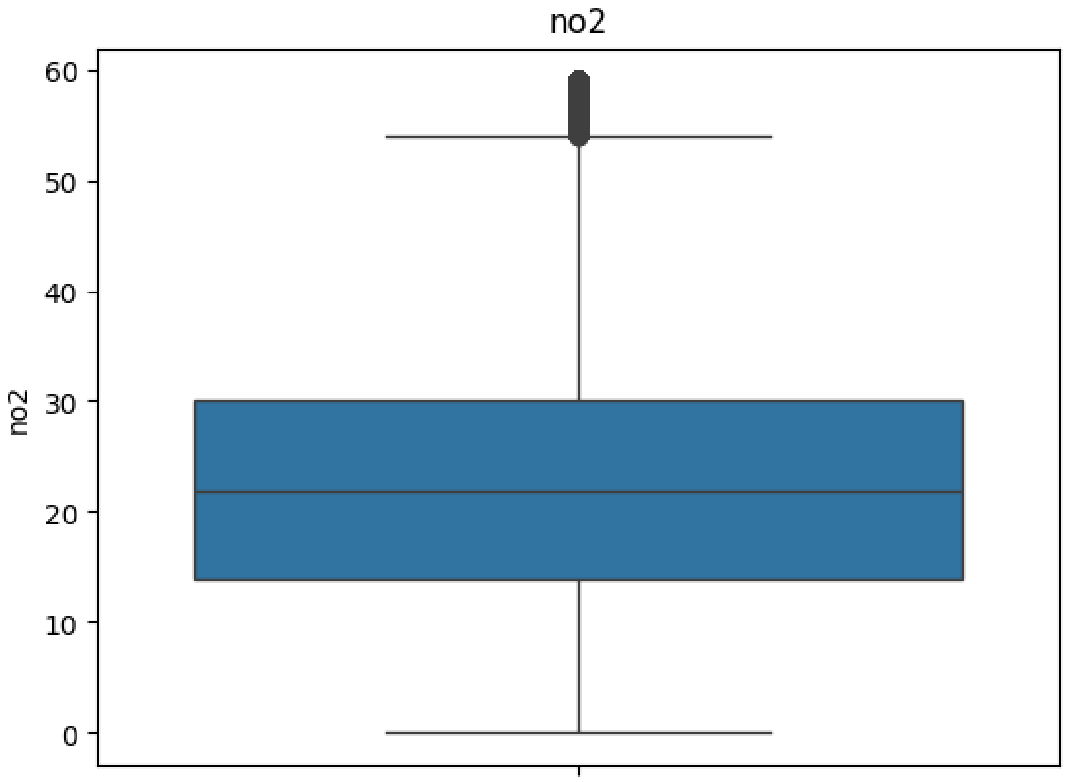
```
3                     Andhra Pradesh  Residential, Rural and other
Areas
4                     Andhra Pradesh                     Industrial
Area
...                             ...                               ..
.
435737                   West Bengal
RIRUO
435738                   West Bengal
RIRUO
435739  andaman-and-nicobar-islands  Residential, Rural and other
Areas
435740                   Lakshadweep  Residential, Rural and other
Areas
435741                       Tripura  Residential, Rural and other
Areas

                              state   location
0                    Andhra Pradesh  Hyderabad
1                    Andhra Pradesh  Hyderabad
2                    Andhra Pradesh  Hyderabad
3                    Andhra Pradesh  Hyderabad
4                    Andhra Pradesh  Hyderabad
...                             ...        ...
435737                  West Bengal   ULUBERIA
435738                  West Bengal   ULUBERIA
435739  andaman-and-nicobar-islands   Guwahati
435740                  Lakshadweep   Guwahati
435741                      Tripura   Guwahati

[435068 rows x 4 columns]
```

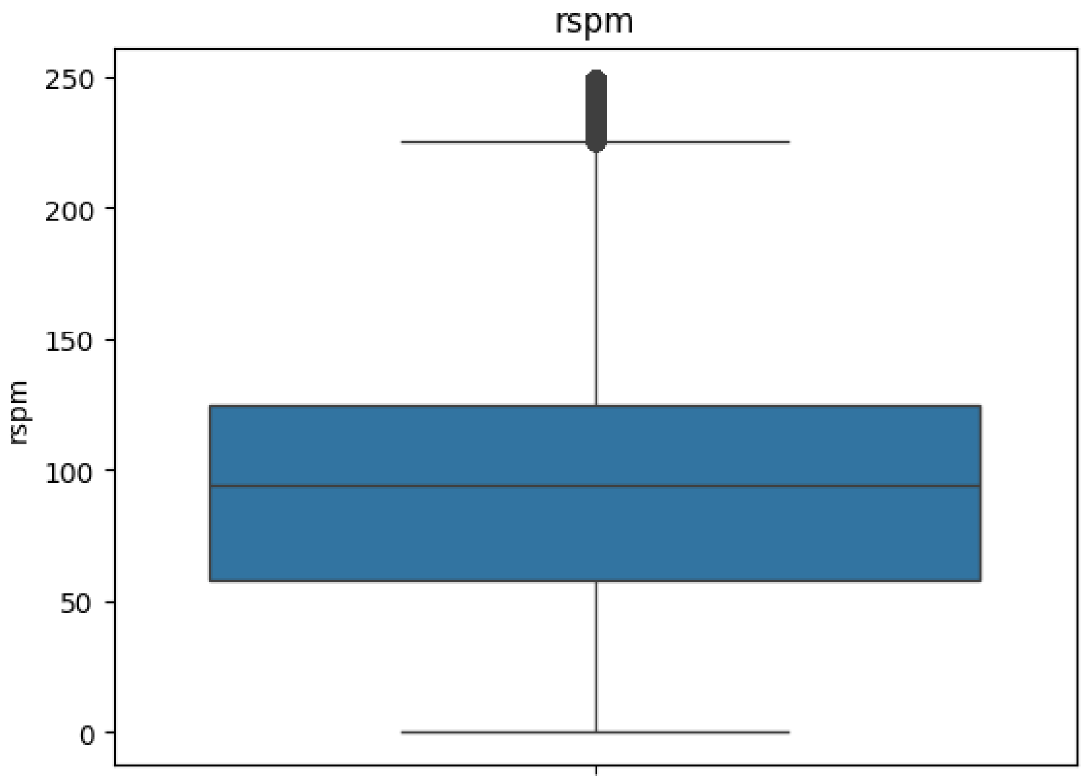
Error Correcting

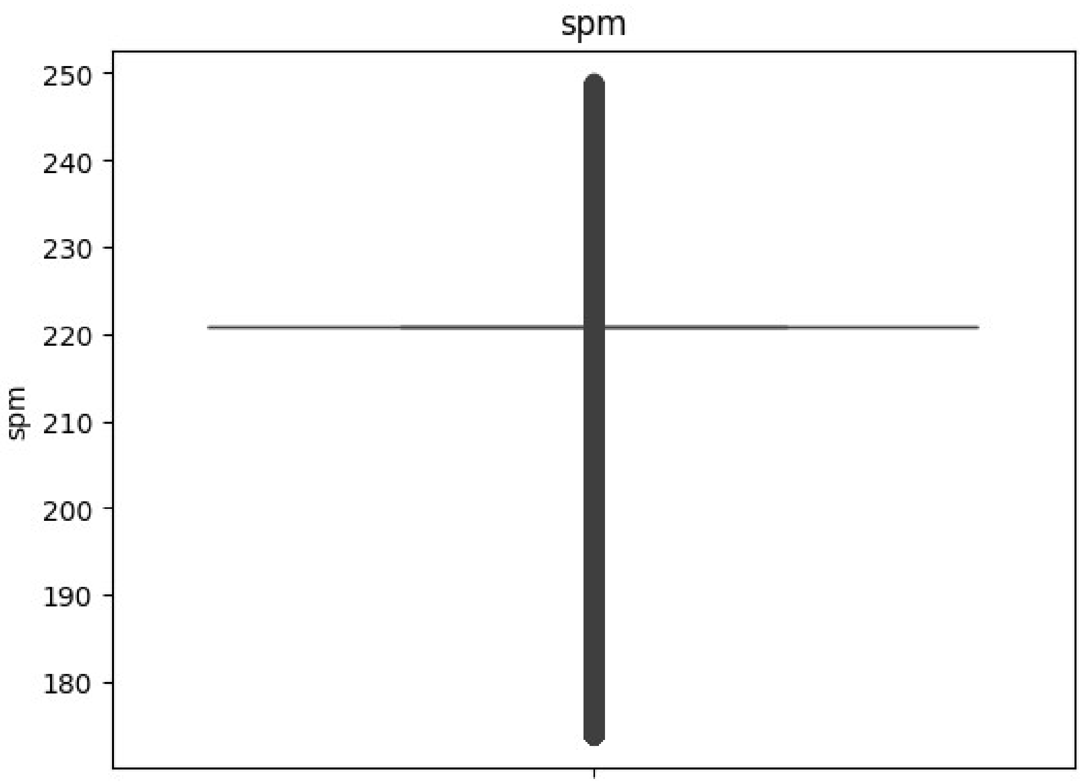
```
def remove_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    threshold = 1.5 * IQR
    outlier_mask = (column < Q1 - threshold) | (column > Q3 +
threshold)
    return column[~outlier_mask]

df.columns

Index(['state', 'location', 'type', 'so2', 'no2', 'rspm', 'spm',
'date'], dtype='object')

# Remove outliers for each column using a loop
col_name = ['so2', 'no2', 'rspm', 'spm']
```

```
for col in col_name:
    df[col] = remove_outliers(df[col])

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))  # Adjust the figure size if needed

for col in col_name:
    sns.boxplot(data=df[col])
    plt.title(col)
    plt.show()
```




no2
60
50
40
30
20
10
0
no2


rspm
250
200
150
100
50
0
rspm



Data Transform

```
from sklearn.preprocessing import LabelEncoder

col_label= ['state','location','type']
# Initialize LabelEncoder

encoder = LabelEncoder()
# Iterate over columns
for col in df.columns:
        # Fit and transform the column
        df[col] = encoder.fit_transform(df[col])

df
```

```
        state  location  type   so2   no2  rspm  spm  date
0           0       114     6   446  1489  2030  464   213
1           0       114     1   197   250  2030  464   213
2           0       114     6   790  3096  2030  464   213
3           0       114     6   823  1144  2030  464   214
4           0       114     1   427   301  2030  464   214
...       ...       ...   ...   ...   ...   ...  ...   ...
435737     35       282     3  2888  5307  2534  464  5059
435738     35       282     3  2809  5113  3098  464  5064
435739     36       100     6  1638  2696  2030  464  4779
435740     17       100     6  1638  2696  2030  464  4779
```

```
435741     31       100     6  1638  2696  2030  464  4779

[435068 rows x 8 columns]
```

```
# import pandas library
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.linear_model import LogisticRegression
import seaborn as sns
import matplotlib.pyplot as plt

# Reading csv file
df = pd.read_csv("Heart.csv")
df.head()

   age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak
slp  \
0   63    1   3     145   233    1        0       150     0      2.3
0
1   37    1   2     130   250    0        1       187     0      3.5
0
2   41    0   1     130   204    0        0       172     0      1.4
2
3   56    1   1     120   236    0        1       178     0      0.8
2
4   57    0   0     120   354    0        1       163     1      0.6
2

   caa  thall  output
0    0      1       1
1    0      2       1
2    0      2       1
3    0      2       1
4    0      2       1
```

Data Cleaning

```
df = df.drop_duplicates()

# Count ,min,max ,etc of each column
df.describe()

             age         sex          cp      trtbps        chol
fbs  \
count  302.00000  302.000000  302.000000  302.000000  302.000000
302.000000
mean    54.42053    0.682119    0.963576  131.602649  246.500000
0.149007
std      9.04797    0.466426    1.032044   17.563394   51.753489
```

```
0.356686
min     29.00000    0.000000    0.000000   94.000000  126.000000
0.000000
25%     48.00000    0.000000    0.000000  120.000000  211.000000
0.000000
50%     55.50000    1.000000    1.000000  130.000000  240.500000
0.000000
75%     61.00000    1.000000    2.000000  140.000000  274.750000
0.000000
max     77.00000    1.000000    3.000000  200.000000  564.000000
1.000000

          restecg    thalachh        exng     oldpeak         slp
caa  \
count  302.000000  302.000000  302.000000  302.000000  302.000000
302.000000
mean     0.526490  149.569536    0.327815    1.043046    1.397351
0.718543
std      0.526027   22.903527    0.470196    1.161452    0.616274
1.006748
min      0.000000   71.000000    0.000000    0.000000    0.000000
0.000000
25%      0.000000  133.250000    0.000000    0.000000    1.000000
0.000000
50%      1.000000  152.500000    0.000000    0.800000    1.000000
0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000
1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000
4.000000

            thall      output
count  302.000000  302.000000
mean     2.314570    0.543046
std      0.613026    0.498970
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

```
# Information about each column data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 302 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       302 non-null    int64
 1   sex       302 non-null    int64
 2   cp        302 non-null    int64
 3   trtbps    302 non-null    int64
 4   chol      302 non-null    int64
 5   fbs       302 non-null    int64
 6   restecg   302 non-null    int64
 7   thalachh  302 non-null    int64
 8   exng      302 non-null    int64
 9   oldpeak   302 non-null    float64
 10  slp       302 non-null    int64
 11  caa       302 non-null    int64
 12  thall     302 non-null    int64
 13  output    302 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 35.4 KB
```

```
#Finding null values in each column
df.isna().sum()
```

```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

Data Integration

```
df.head()
```

```
   age  sex  cp  trtbps  chol  fbs  restecg  thalachh  exng  oldpeak
slp  \
0   63    1   3     145   233    1        0       150     0      2.3
0
1   37    1   2     130   250    0        1       187     0      3.5
0
2   41    0   1     130   204    0        0       172     0      1.4
2
3   56    1   1     120   236    0        1       178     0      0.8
2
4   57    0   0     120   354    0        1       163     1      0.6
```

```
2
```

```
   caa  thall  output
0    0      1       1
1    0      2       1
2    0      2       1
3    0      2       1
4    0      2       1
```

```python
df.fbs.unique()
```

```
array([1, 0], dtype=int64)
```

```python
subSet1 = df[['age','cp','chol','thalachh']]
```

```python
subSet2 = df[['exng','slp','output']]
```

```python
merged_df = subSet1.merge(right=subSet2,how='cross')
merged_df.head()
```

```
   age  cp  chol  thalachh  exng  slp  output
0   63   3   233       150     0    0       1
1   63   3   233       150     0    0       1
2   63   3   233       150     0    2       1
3   63   3   233       150     0    2       1
4   63   3   233       150     1    2       1
```

Error Correcting

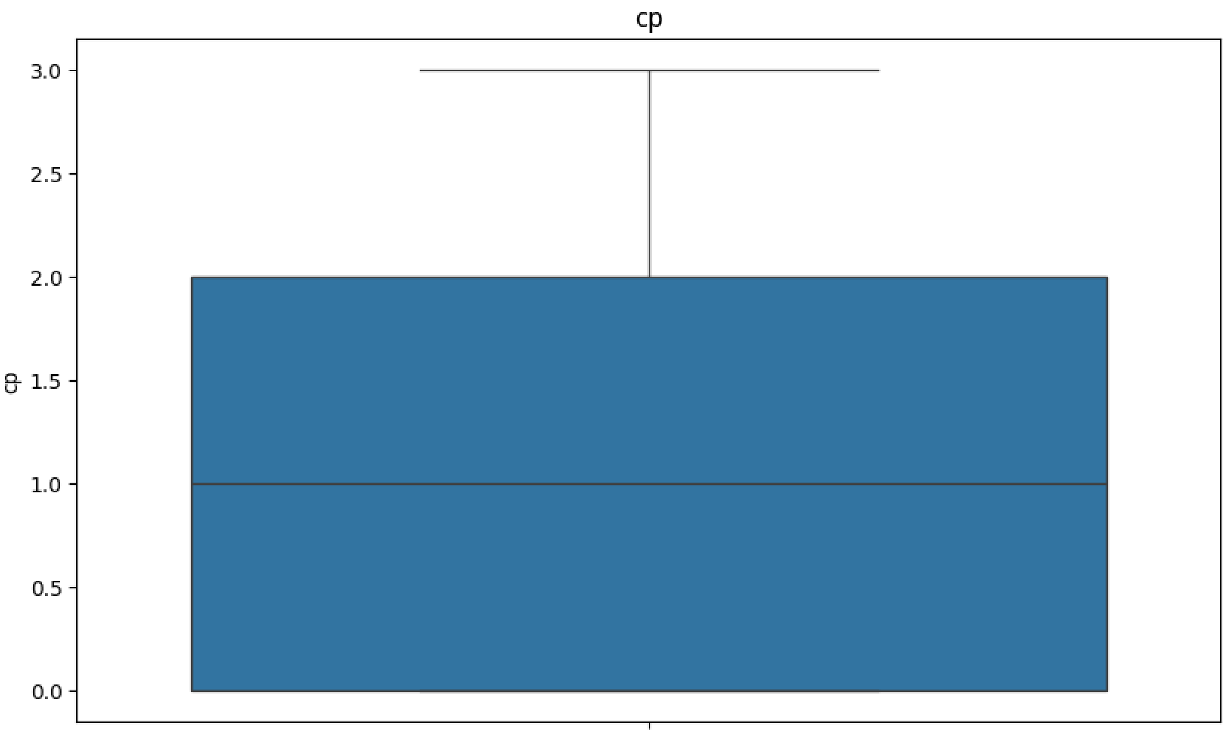
```python
df.columns
```

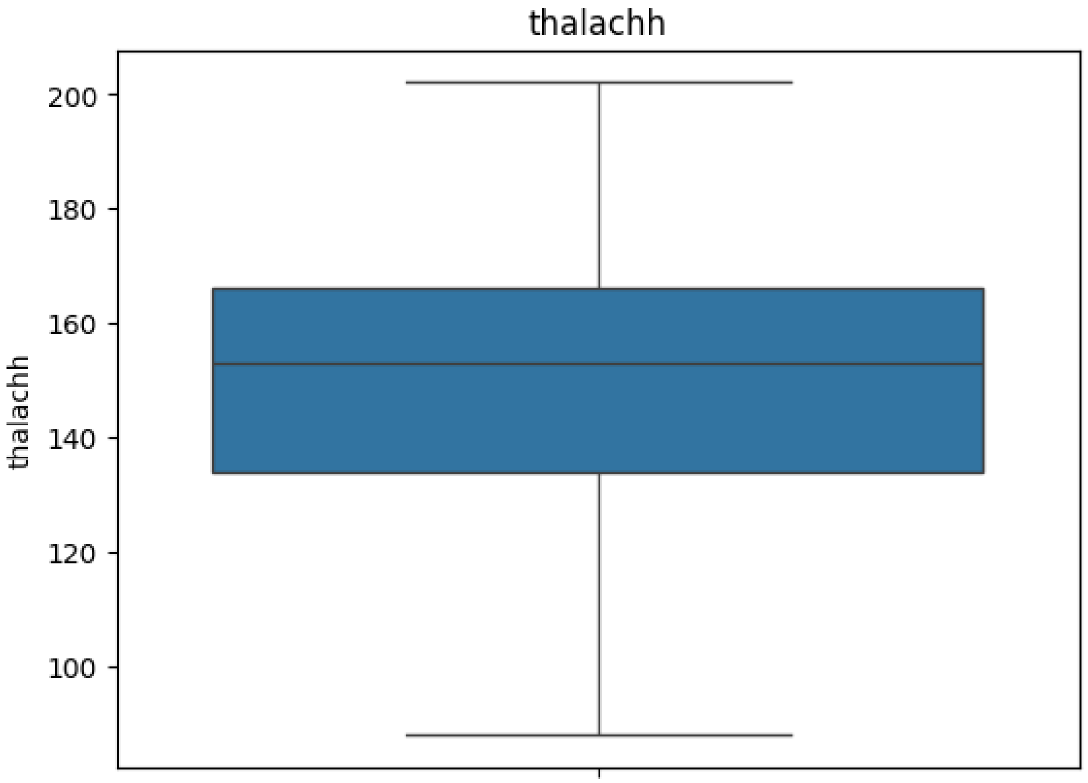
```
Index(['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg',
'thalachh',
       'exng', 'oldpeak', 'slp', 'caa', 'thall', 'output'],
      dtype='object')
```

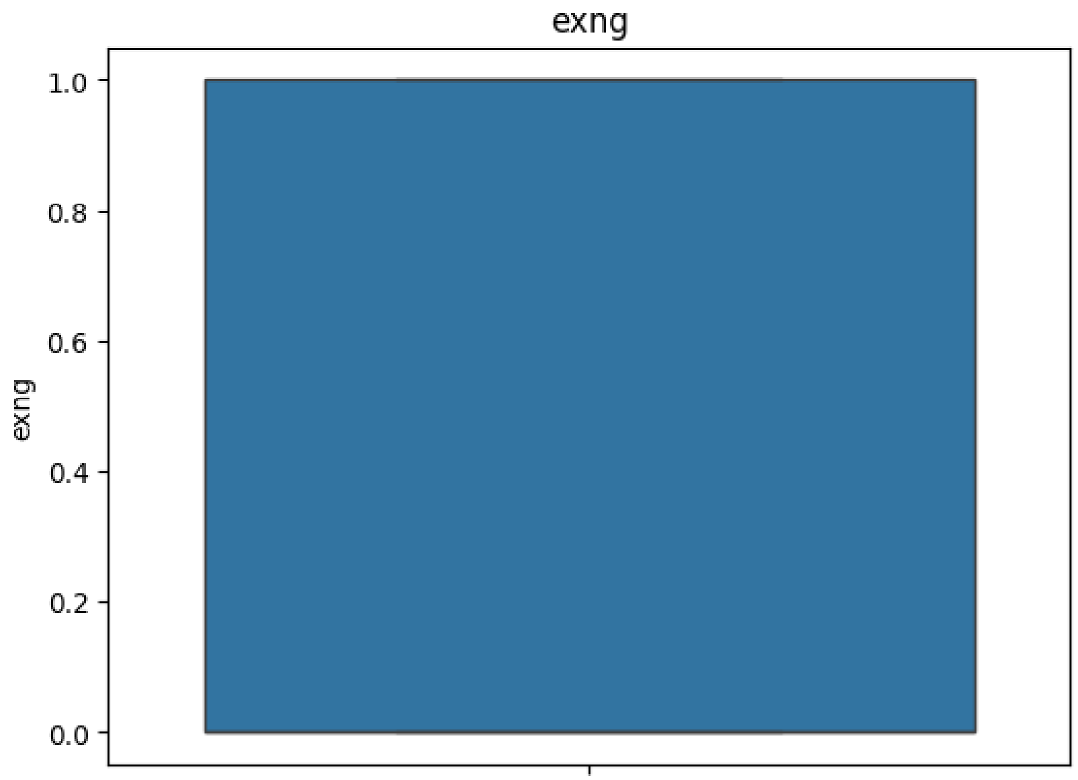
```python
def remove_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    threshold = 1.5 * IQR
    outlier_mask = (column < Q1 - threshold) | (column > Q3 +
threshold)
    return column[~outlier_mask]
```

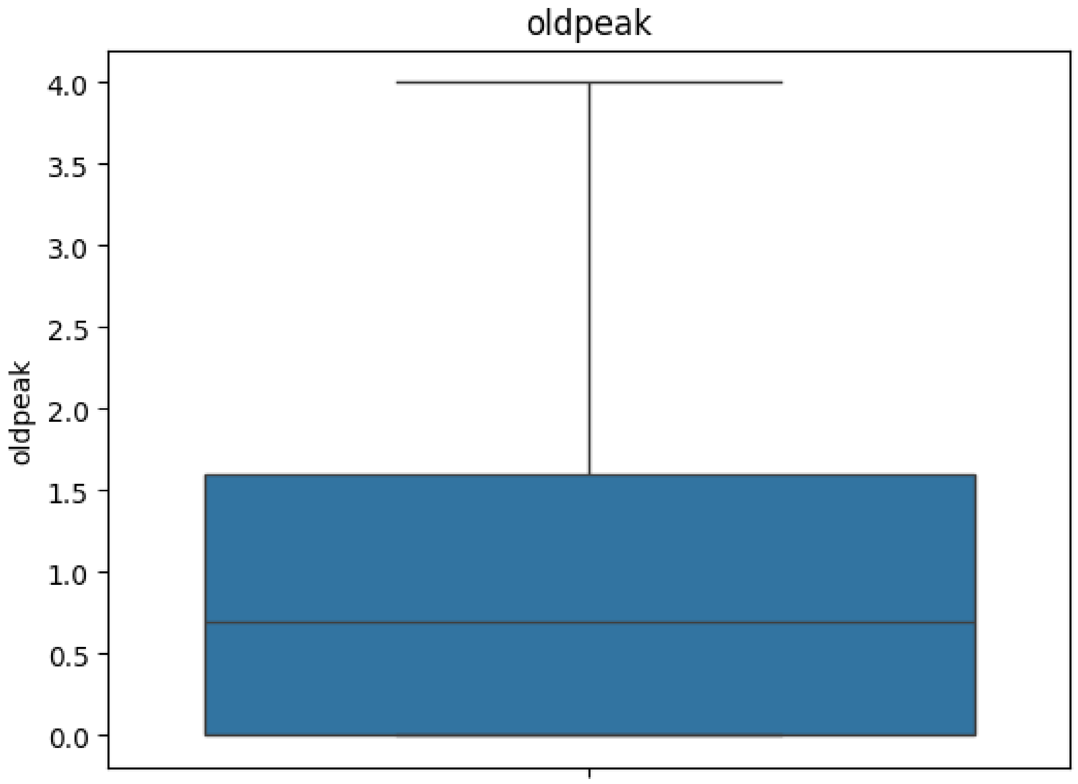
```python
# Remove outliers for each column using a loop
col_name = ['cp','thalachh','exng','oldpeak','slp','caa']
for col in col_name:
    df[col] = remove_outliers(df[col])
```

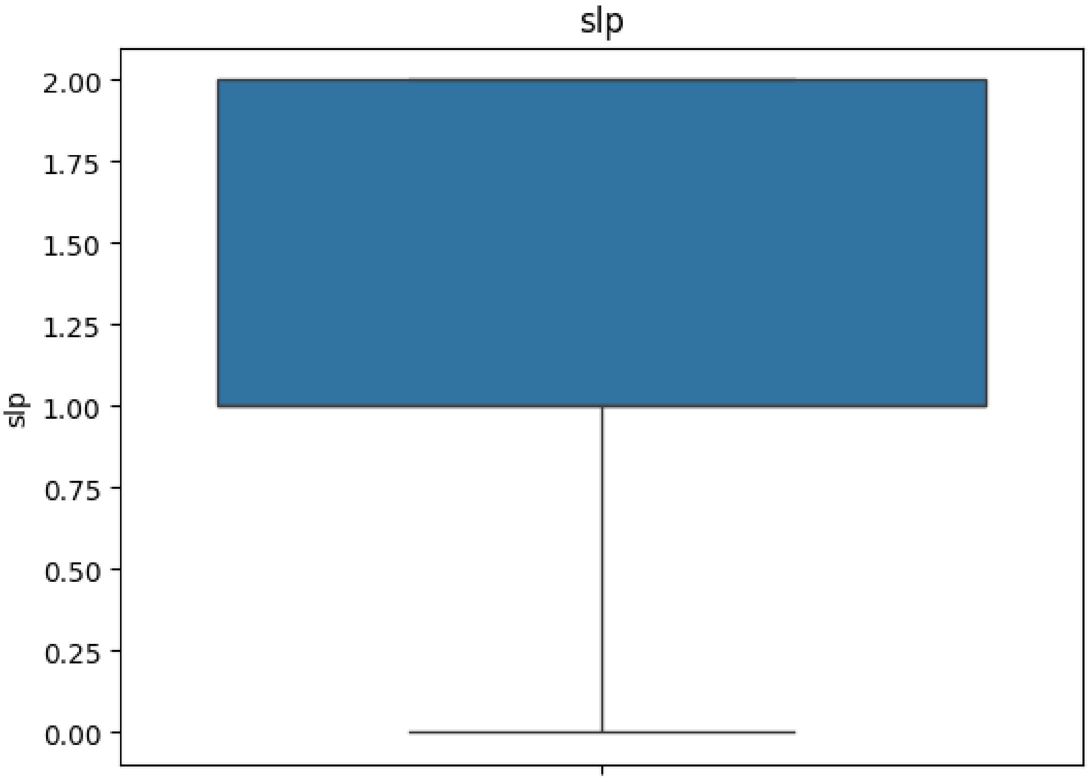
```python
plt.figure(figsize=(10, 6))  # Adjust the figure size if needed

for col in col_name:
    sns.boxplot(data=df[col])
    plt.title(col)
    plt.show()
```

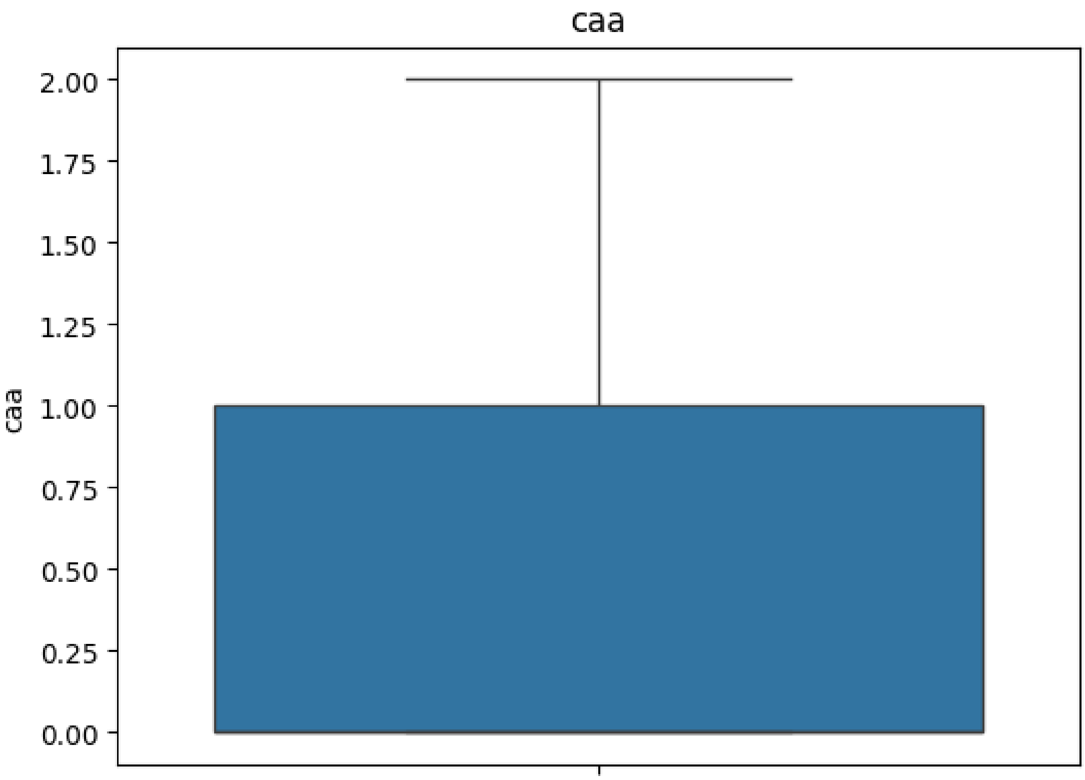



thalachh
thalachh
200
180
160
140
120
100


exng
exng
1.0
0.8
0.6
0.4
0.2
0.0


oldpeak
oldpeak
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0


slp
slp
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00



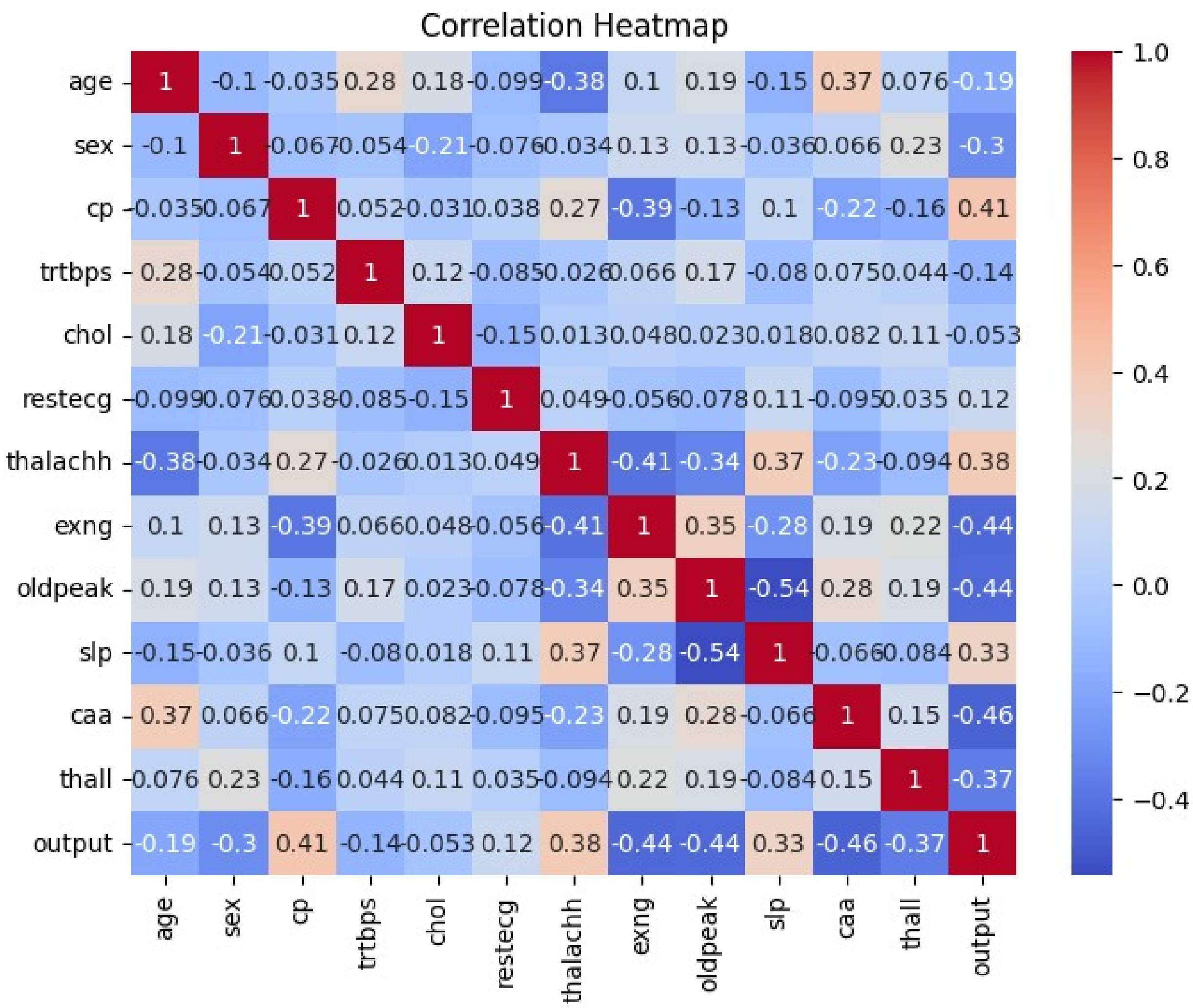
```
df = df.dropna()

df.isna().sum()
```

```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

```
df = df.drop('fbs',axis=1)

# Compute correlations between features and target
correlations = df.corr()['output'].drop('output')

# Print correlations
```

```
print("Correlation with the Target:")
print(correlations)
print()

# Plot correlation heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
Correlation with the Target:
age        -0.193798
sex        -0.303271
cp          0.410807
trtbps     -0.135238
chol       -0.052796
restecg     0.122071
thalachh    0.384609
exng       -0.444401
oldpeak    -0.437895
slp         0.329432
caa        -0.460816
thall      -0.366390
Name: output, dtype: float64
```



```
# df.isna().sum()
```

Data Split

```
# splitting data using train test split
x = df[['cp','thalachh','exng','oldpeak','slp','caa']]
y = df.output
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,rando
m_state=0)

x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((220, 6), (55, 6), (220,), (55,))
```

Data transformation

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()

x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

Data model building

```
y_train= np.array(y_train).reshape(-1, 1)
y_test= np.array(y_test).reshape(-1, 1)

y_train.shape
```

```
(220, 1)
```

```
model = LogisticRegression()
model.fit(x_train_scaled, y_train)

# Make predictions on the test set
y_pred = model.predict(x_test_scaled)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8363636363636363
```

```
C:\Users\Gayatri Tagalpallewa\AppData\Roaming\Python\Python312\site-packages\sklearn\utils\validation.py:1339: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
#Classification model using Decision Tree
from sklearn.tree import DecisionTreeClassifier
tc=DecisionTreeClassifier(criterion='entropy')
tc.fit(x_train_scaled,y_train)
y_pred=tc.predict(x_test_scaled)

print("Training Accuracy Score :",accuracy_score(y_pred,y_test))
print("Training Confusion Matrix  :",confusion_matrix(y_pred,y_test))
```

```
Training Accuracy Score : 0.7818181818181819
Training Confusion Matrix  : [[20  5]
 [ 7 23]]
```