```
# 2. Perform the following operations using Python on the Air quality and Heart Diseases data sets
# a. Data cleaning
# b. Data integration
# c. Data transformation
# d. Error correcting
# e. Data model building


import pandas as pd
import numpy as np
df =pd.read_csv("/content/heartdisease.csv")
df.head()
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHI
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No

```
df.isnull()
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
299	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
300	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
301	False	False	False	False	False	False	False	False	False	False	False	False	False	False	F
302	False	False	False	False	False	False	False	False	False	False	False	False	True	False	F

303 rows × 15 columns

```
df.isna().any()
```

```
Unnamed: 0    False
Age           False
Sex           False
ChestPain     False
RestBP        False
Chol          False
Fbs           False
RestECG       False
MaxHR         False
ExAng         False
Oldpeak       False
Slope         False
Ca             True
Thal           True
AHD           False
dtype: bool
```

```
df.isnull().sum()
```

```
Unnamed: 0    0
Age           0
Sex           0
ChestPain     0
RestBP        0
Chol          0
Fbs           0
RestECG       0
MaxHR         0
ExAng         0
Oldpeak       0
Slope         0
Ca            4
Thal          2
AHD           0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

```
a=df.drop_duplicates(subset=None,keep=False,ignore_index=True)
```

```
a
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	A
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	`
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	`
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	299	45	1	typical	110	264	0	0	132	0	1.2	2	0.0	reversable	`
299	300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.0	reversable	`
300	301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2	1.0	reversable	`
301	302	57	0	nontypical	130	236	0	2	174	0	0.0	2	1.0	normal	`
302	303	38	1	nonanginal	138	175	0	0	173	0	0.0	1	NaN	normal	

303 rows × 15 columns

```
a.isna().sum()
```

```
Unnamed: 0    0
Age           0
Sex           0
ChestPain     0
RestBP        0
Chol          0
Fbs           0
RestECG       0
MaxHR         0
ExAng         0
Oldpeak       0
Slope         0
Ca            4
Thal          2
```

```
AHD          0
dtype: int64
```

```
#data cleaning
mean_value = df['Ca'].mean()
df['Ca'].fillna(value=mean_value, inplace=True)
df
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Th
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.000000	fix
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.000000	norm
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.000000	reversab
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.000000	norm
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.000000	norm
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	299	45	1	typical	110	264	0	0	132	0	1.2	2	0.000000	reversab
299	300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.000000	reversab
300	301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2	1.000000	reversab
301	302	57	0	nontypical	130	236	0	2	174	0	0.0	2	1.000000	norm
302	303	38	1	nonanginal	138	175	0	0	173	0	0.0	1	0.672241	norm

303 rows × 15 columns

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

```
Unnamed: 0    0
Age           0
Sex           0
ChestPain     0
RestBP        0
Chol          0
Fbs           0
RestECG       0
MaxHR         0
ExAng         0
Oldpeak       0
Slope         0
Ca            0
Thal          2
AHD           0
dtype: int64
```

```
most_common_value = df['Thal'].mode()[0]
df['Thal'].fillna(most_common_value, inplace=True)
```

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

```
Unnamed: 0    0
Age           0
Sex           0
ChestPain     0
RestBP        0
Chol          0
Fbs           0
RestECG       0
MaxHR         0
ExAng         0
Oldpeak       0
Slope         0
Ca            0
```

```
Thal          0
AHD           0
dtype: int64
```

```
#data Transformation
```

```
df['ChestPain'].unique()
```

```
array(['typical', 'asymptomatic', 'nonanginal', 'nontypical'],
      dtype=object)
```

```
#data transformation
df['ChestPain'] = df['ChestPain'].replace('typical',0)
df['ChestPain'] = df['ChestPain'].replace('asymptomatic',1)
df['ChestPain'] = df['ChestPain'].replace('nonanginal',2)
df['ChestPain'] = df['ChestPain'].replace('nontypical',3)
df
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
0	1	63	1	0	145	233	1	2	150	0	2.3	3	0.000000	fixed
1	2	67	1	1	160	286	0	2	108	1	1.5	2	3.000000	normal
2	3	67	1	1	120	229	0	2	129	1	2.6	2	2.000000	reversable
3	4	37	1	2	130	250	0	0	187	0	3.5	3	0.000000	normal
4	5	41	0	3	130	204	0	2	172	0	1.4	1	0.000000	normal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	299	45	1	0	110	264	0	0	132	0	1.2	2	0.000000	reversable
299	300	68	1	1	144	193	1	0	141	0	3.4	2	2.000000	reversable
300	301	57	1	1	130	131	0	0	115	1	1.2	2	1.000000	reversable
301	302	57	0	3	130	236	0	2	174	0	0.0	2	1.000000	normal
302	303	38	1	2	138	175	0	0	173	0	0.0	1	0.672241	normal

303 rows × 15 columns

```
#data transformation
df['Sex'] = df['Sex'].replace(1,'M')
df['Sex'] = df['Sex'].replace(0,'F')
df
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal

```
#data transformation
df['AHD'] = df['AHD'].replace('Yes',1)
df['AHD'] = df['AHD'].replace('No',0)
df
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
0	1	63	M	0	145	233	1	2	150	0	2.3	3	0.000000	fixed
1	2	67	M	1	160	286	0	2	108	1	1.5	2	3.000000	normal
2	3	67	M	1	120	229	0	2	129	1	2.6	2	2.000000	reversable
3	4	37	M	2	130	250	0	0	187	0	3.5	3	0.000000	normal
4	5	41	F	3	130	204	0	2	172	0	1.4	1	0.000000	normal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	299	45	M	0	110	264	0	0	132	0	1.2	2	0.000000	reversable
299	300	68	M	1	144	193	1	0	141	0	3.4	2	2.000000	reversable
300	301	57	M	1	130	131	0	0	115	1	1.2	2	1.000000	reversable
301	302	57	F	3	130	236	0	2	174	0	0.0	2	1.000000	normal
302	303	38	M	2	138	175	0	0	173	0	0.0	1	0.672241	normal

303 rows × 15 columns

```
#data transformation
for i in range(len(df["Chol"])):
  if i>200:
    df['Chol']= df['Chol'].replace(i,"high")
  else:
    df['Chol']= df['Chol'].replace(i,"low")
df
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
0	1	63	M	0	145	high	1	2	150	0	2.3	3	0.000000	fixed
1	2	67	M	1	160	high	0	2	108	1	1.5	2	3.000000	normal
2	3	67	M	1	120	high	0	2	129	1	2.6	2	2.000000	reversable
3	4	37	M	2	130	high	0	0	187	0	3.5	3	0.000000	normal
4	5	41	F	3	130	high	0	2	172	0	1.4	1	0.000000	normal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	299	45	M	0	110	high	0	0	132	0	1.2	2	0.000000	reversable
299	300	68	M	1	144	low	1	0	141	0	3.4	2	2.000000	reversable
300	301	57	M	1	130	low	0	0	115	1	1.2	2	1.000000	reversable
301	302	57	F	3	130	high	0	2	174	0	0.0	2	1.000000	normal
302	303	38	M	2	138	low	0	0	173	0	0.0	1	0.672241	normal

303 rows × 15 columns

```
#data transformation by cubing
df["Oldpeak"]=df["Oldpeak"]*df["Oldpeak"]*df["Oldpeak"]
df
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal
0	1	63	M	0	145	high	1	2	150	0	12.167	3	0.000000	fixed
1	2	67	M	1	160	high	0	2	108	1	3.375	2	3.000000	normal
2	3	67	M	1	120	high	0	2	129	1	17.576	2	2.000000	reversable
3	4	37	M	2	130	high	0	0	187	0	42.875	3	0.000000	normal
4	5	41	F	3	130	high	0	2	172	0	2.744	1	0.000000	normal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	299	45	M	0	110	high	0	0	132	0	1.728	2	0.000000	reversable
299	300	68	M	1	144	low	1	0	141	0	39.304	2	2.000000	reversable
300	301	57	M	1	130	low	0	0	115	1	1.728	2	1.000000	reversable
301	302	57	F	3	130	high	0	2	174	0	0.000	2	1.000000	normal
302	303	38	M	2	138	low	0	0	173	0	0.000	1	0.672241	normal

303 rows × 15 columns

```
#data subsets
df2=df[['Age','Sex','ChestPain']].loc[0:10]
df3=df[['Age','Sex','ChestPain']].loc[11:20]


#data integration
m_df=pd.concat([df2,df3])
m_df
```

	Age	Sex	ChestPain
0	63	M	0
1	67	M	1
2	67	M	1
3	37	M	2
4	41	F	3
5	56	M	3
6	62	F	1
7	57	F	1
8	63	M	1
9	53	M	1
10	57	M	1
11	56	F	3
12	56	M	2
13	44	M	3
14	52	M	2
15	57	M	2
16	48	M	3
17	54	M	1
18	48	F	2
19	49	M	3
20	64	M	0

```
#data correction
df['Ca'].unique()
```

```
array([0.       , 3.       , 2.       , 1.       , 0.6722408])
```

```
df['Thal'].unique()
```

```
array(['fixed', 'normal', 'reversable'], dtype=object)
```

```
df1=pd.read_csv("/content/heartdisease.csv")
df1['Ca'].unique()
```

```
array([ 0.,  3.,  2.,  1., nan])
```

```
df1 = df1.fillna(df.mode().iloc[0])
```

```
df1['Ca'].unique()
```

```
array([0., 3., 2., 1.])
```

```
#if '?' value in the column then follow :
# df.loc[df['Ca']=="?", 'Ca']=df['Ca'].mode()
# df=df.fillna(df.mode().iloc[0])
# df["Ca"].unique()
```

```
array([0., 3., 2., 1.])
```

```
#data model building
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

feature_cols = ['Age','ChestPain','RestBP',"Fbs",'RestECG','MaxHR','Slope']
X = df[feature_cols] # Features
y = df.AHD # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
print("F1 score:",metrics.f1_score(y_test, y_pred))
print("Precesion:",metrics.precision_score(y_test, y_pred))
```

```
Accuracy: 0.6923076923076923
Recall: 0.5476190476190477
F1 score: 0.6216216216216217
Precesion: 0.71875
```

✓ 0s completed at 11:54 PM