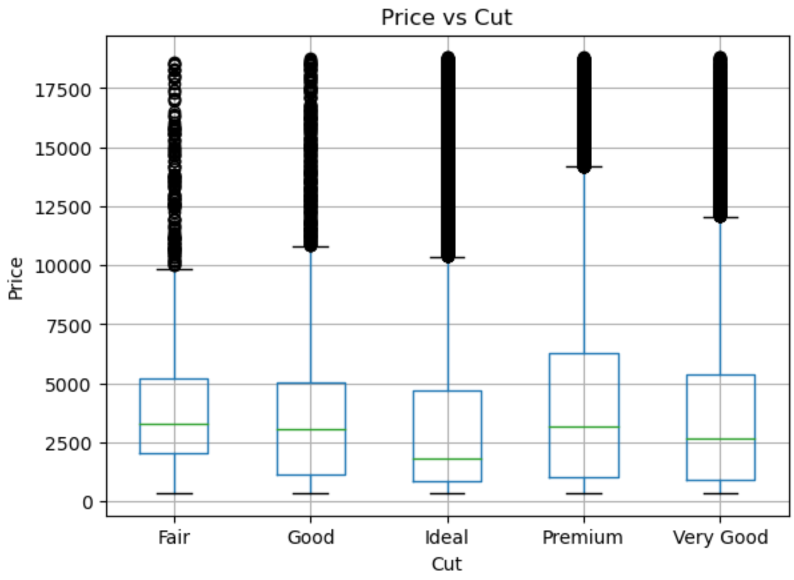
```
#Program 1

# Plot a boxplot for p̄rice̲␃vs c̄ut̲␃from the dataset d̄iamond.csv̲␃ Which of the categories under c̄ut̲␃have the highest median price?


import pandas as pd
import matplotlib.pyplot as plt

# Read the dataset
df = pd.read_csv("diamonds.csv")

print(df.head())

# Create the boxplot
plt.figure(figsize=(8, 6))
df.boxplot(column='price', by='cut')
plt.xlabel('Cut')
plt.ylabel('Price')
plt.title('Price vs Cut')
plt.suptitle('')  # Remove the default title
plt.show()
```

```
   Unnamed: 0  carat      cut color clarity  depth  table  price     x     y  \
0           1   0.23    Ideal     E     SI2   61.5   55.0    326  3.95  3.98
1           2   0.21  Premium     E     SI1   59.8   61.0    326  3.89  3.84
2           3   0.23     Good     E     VS1   56.9   65.0    327  4.05  4.07
3           4   0.29  Premium     I     VS2   62.4   58.0    334  4.20  4.23
4           5   0.31     Good     J     SI2   63.3   58.0    335  4.34  4.35

      z
0  2.43
1  2.31
2  2.31
3  2.63
4  2.75
<Figure size 800x600 with 0 Axes>
```



```
#Program 2

'''Create a frequency table (one-way table) for the variable c̄ut̲⍰from the dataset d̄iamond.csv̲⍰What is the frequency for
the cut type Ideal̲'''


import pandas as pd

# Read the dataset
df = pd.read_csv("diamonds.csv")

# Create the frequency table
frequency_table = df['cut'].value_counts()

# Print the frequency table
print(frequency_table)

# Access the frequency for the cut type "Ideal"
ideal_frequency = frequency_table['Ideal']
```

```
print("Frequency for cut type 'Ideal':", ideal_frequency)
```

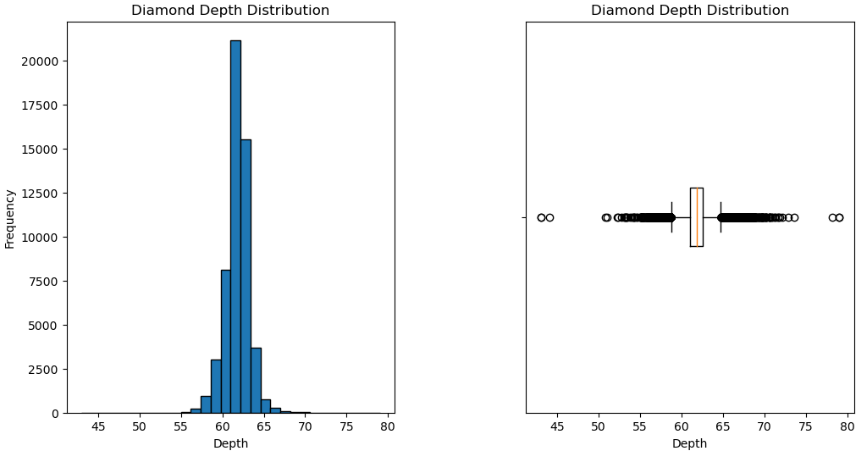
```
Ideal        21551
Premium      13791
Very Good    12082
Good          4906
Fair          1610
Name: cut, dtype: int64
Frequency for cut type 'Ideal': 21551
```

```
#Program 3

# Show the subplot of the diamond carat weight distribution.

import pandas as pd
import matplotlib.pyplot as plt

# Read the dataset
df = pd.read_csv("diamonds.csv")

print(df['carat'])

# Create a subplot with 1 row and 2 columns
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# Plot the histogram of carat weight on the first subplot
axs[0].hist(df['carat'], bins=30, edgecolor='black')
axs[0].set_xlabel('Carat Weight')
axs[0].set_ylabel('Frequency')
axs[0].set_title('Carat Weight Distribution')

# Plot the boxplot of carat weight on the second subplot
axs[1].boxplot(df['carat'], vert=False)
axs[1].set_yticklabels('')
axs[1].set_xlabel('Carat Weight')
axs[1].set_title('Carat Weight Distribution')

# Adjust the spacing between subplots
plt.subplots_adjust(wspace=0.3)

# Show the plot
plt.show()
```

```
0        0.23
1        0.21
2        0.23
```

```
#Program 4

# Show the subplot of diamond depth distribution.

import pandas as pd
import matplotlib.pyplot as plt

# Read the dataset
df = pd.read_csv("diamonds.csv")
print(df['depth'])

# Create a subplot with 1 row and 2 columns
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# Plot the histogram of diamond depth on the first subplot
axs[0].hist(df['depth'], bins=30, edgecolor='black')
axs[0].set_xlabel('Depth')
axs[0].set_ylabel('Frequency')
axs[0].set_title('Diamond Depth Distribution')

# Plot the boxplot of diamond depth on the second subplot
axs[1].boxplot(df['depth'], vert=False)
axs[1].set_yticklabels('')
axs[1].set_xlabel('Depth')
axs[1].set_title('Diamond Depth Distribution')

# Adjust the spacing between subplots
plt.subplots_adjust(wspace=0.4)

# Show the plot
plt.show()
```

```
0        61.5
1        59.8
2        56.9
3        62.4
4        63.3
         ...
53935    60.8
53936    63.1
53937    62.8
53938    61.0
53939    62.2
Name: depth, Length: 53940, dtype: float64
```



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

```
# Read the dataset
df = pd.read_csv("diamonds.csv")

# Define the input features (X) and target variable (y)
X = df[['carat', 'cut', 'clarity', 'depth']]
y = df['price']

# Convert categorical variables to numerical using one-hot encoding
X_encoded = pd.get_dummies(X)

print("X_encoded=",X_encoded)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

# Build the linear regression model
model = LinearRegression()

model.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = model.predict(X_test)

# Calculate the accuracy (R-squared score)
accuracy = r2_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
X_encoded=        carat  depth  cut_Fair  cut_Good  cut_Ideal  cut_Premium  \
0       0.23   61.5         0         0          1            0   
1       0.21   59.8         0         0          0            1   
2       0.23   56.9         0         1          0            0   
3       0.29   62.4         0         0          0            1   
4       0.31   63.3         0         1          0            0   
...      ...    ...       ...       ...        ...          ...   
53935   0.72   60.8         0         0          1            0   
53936   0.72   63.1         0         1          0            0   
53937   0.70   62.8         0         0          0            0   
53938   0.86   61.0         0         0          0            1   
53939   0.75   62.2         0         0          1            0   

       cut_Very Good  clarity_I1  clarity_IF  clarity_SI1  clarity_SI2  \
0                  0           0           0            0            1   
1                  0           0           0            1            0   
2                  0           0           0            0            0   
3                  0           0           0            0            0   
4                  0           0           0            0            1   
...              ...         ...         ...          ...          ...   
53935              0           0           0            1            0   
53936              0           0           0            1            0   
53937              1           0           0            1            0   
53938              0           0           0            0            1   
53939              0           0           0            0            1   

       clarity_VS1  clarity_VS2  clarity_VVS1  clarity_VVS2  
0                0            0             0             0  
1                0            0             0             0  
2                1            0             0             0  
3                0            1             0             0  
4                0            0             0             0  
...            ...          ...           ...           ...  
53935            0            0             0             0  
53936            0            0             0             0  
53937            0            0             0             0  
53938            0            0             0             0  
53939            0            0             0             0  

[53940 rows x 15 columns]
LinearRegression()
Accuracy: 0.8966651780887069
```