2cs317-pes2ug22cs325-pes2ug22cs910

November 18, 2023

DATASET-2: Diamond Prices

Features Description: • Price: price in US dollars
• Carat: is the diamond's physical weight measured in metric carats.
• Cut: quality of the cut
• Color: diamond color, from J (worst) to D (best) • Clarity: a measurement of how clear the diamond is
• X: length in mm • Y: width in mm • Z: depth in mm • Depth: total depth percentage = z / mean(x, y) = 2 * z / (x + y)

Table: width of the top of diamond relative to widest poin

[24]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Read the dataset
df = pd.read_csv('dataset_2_Diamonds Prices.csv')
print(df)
```

```
       index  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
…      …    …        …   …     …    …    …    …   …   …
53938  53939   0.86    Premium     H     SI2   61.0   58.0   2757  6.15  6.12
53939  53940   0.75      Ideal     D     SI2   62.2   55.0   2757  5.83  5.87
53940  53941   0.71    Premium     E     SI1   60.5   55.0   2756  5.79  5.74
53941  53942   0.71    Premium     F     SI1   59.8   62.0   2756  5.74  5.73
53942  53943   0.70  Very Good     E     VS2   60.5   59.0   2757  5.71  5.76

          z
0      2.43
1      2.31
2      2.31
3      2.63
```

```
4      2.75
…      …
53938  3.74
53939  3.64
53940  3.49
53941  3.43
53942  3.47

[53943 rows x 11 columns]
```

Questions: 1.

i) What is the shape of the dataset? (Specify rows and columns separately)
ii) List the column names and their data types?

iii) Delete ‘index’ column?

[25]:

```python
# Get the shape of the DataFrame
shape = df.shape
# Print the shape
print(f'The dataset has {shape[0]} rows and {shape[1]} columns.')
print()

# list of columns names and their data types
colTy = df.dtypes
# Print the column names and their data types
print(colTy)
print()

# remove the index column and print the new DataFrame
coldel = 'index'
df = df.drop(columns=[coldel])
print(df)
```

```
The dataset has 53943 rows and 11 columns.

index        int64
carat      float64
cut         object
color       object
clarity     object
depth      float64
table      float64
price        int64
x          float64
y          float64
z          float64
dtype: object
```

```
       carat        cut color clarity  depth  table  price     x     y     z
0       0.23      Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1       0.21    Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2       0.23       Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3       0.29    Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4       0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
…      …      …   …    …    …    …     …   …   …    …
53938   0.86    Premium     H     SI2   61.0   58.0   2757  6.15  6.12  3.74
53939   0.75      Ideal     D     SI2   62.2   55.0   2757  5.83  5.87  3.64
53940   0.71    Premium     E     SI1   60.5   55.0   2756  5.79  5.74  3.49
53941   0.71    Premium     F     SI1   59.8   62.0   2756  5.74  5.73  3.43
53942   0.70  Very Good     E     VS2   60.5   59.0   2757  5.71  5.76  3.47

[53943 rows x 10 columns]
```

Question 2.

Describe the summary statistics, min, max, mean, standard deviation for all numeric columns?

[26]:

```
summary = df.describe()
print(summary)

print()
minimum = summary.min()
print("Min:\n",minimum)

print()
maximum = summary.max()
print("Max:\n",maximum)

print()
mean = summary.loc['mean']
print("Mean:\n",mean)

print()
standard_deviation = summary.loc['std']
print("Standard Deviation:\n",standard_deviation)
```

```
              carat         depth         table         price             x  \
count  53943.000000  53935.000000  53943.000000  53943.000000  53931.000000
mean       0.797935     61.749426     57.457251   3932.734294      5.731166
std        0.473999      1.432672      2.234549   3989.338447      1.121819
min        0.200000     43.000000     43.000000    326.000000      0.000000
25%        0.400000     61.000000     56.000000    950.000000      4.710000
50%        0.700000     61.800000     57.000000   2401.000000      5.700000
75%        1.040000     62.500000     59.000000   5324.000000      6.540000
max        5.010000     79.000000     95.000000  18823.000000     10.740000

                  y             z
```

```
count  53934.000000  53933.000000
mean       5.734518      3.538768
std        1.142165      0.705728
min        0.000000      0.000000
25%        4.720000      2.910000
50%        5.710000      3.530000
75%        6.540000      4.040000
max       58.900000     31.800000

Min:
 carat      0.200000
depth      1.432672
table      2.234549
price    326.000000
x          0.000000
y          0.000000
z          0.000000
dtype: float64

Max:
 carat    53943.0
depth    53935.0
table    53943.0
price    53943.0
x        53931.0
y        53934.0
z        53933.0
dtype: float64

Mean:
 carat       0.797935
depth      61.749426
table      57.457251
price    3932.734294
x           5.731166
y           5.734518
z           3.538768
Name: mean, dtype: float64

Standard Deviation:
 carat       0.473999
depth       1.432672
table       2.234549
price    3989.338447
x           1.121819
y           1.142165
z           0.705728
Name: std, dtype: float64
```

Question 3.

List all distinct values and most frequent values in each column ‘cut, ‘colour’ and ‘clarity’?

[27]:

```python
# Assuming df is your DataFrame

# List all distinct values
cut_unique = df['cut'].unique()
color_unique = df['color'].unique()
clarity_unique = df['clarity'].unique()

# Get the most frequent values
cut_most_frequent = df['cut'].mode()[0]
color_most_frequent = df['color'].mode()[0]
clarity_most_frequent = df['clarity'].mode()[0]

# Print all distinct values
print(f"Distinct values in 'cut': {cut_unique}")
print(f"Distinct values in 'color': {color_unique}")
print(f"Distinct values in 'clarity': {clarity_unique}")

# Print the most frequent values
print(f"Most frequent value in 'cut': {cut_most_frequent}")
print(f"Most frequent value in 'color': {color_most_frequent}")
print(f"Most frequent value in 'clarity': {clarity_most_frequent}")
```

```
Distinct values in 'cut': ['Ideal' 'Premium' 'Good' 'Very Good' 'Fair']
Distinct values in 'color': ['E' 'I' 'J' 'H' 'F' 'G' 'D']
Distinct values in 'clarity': ['SI2' 'SI1' 'VS1' 'VS2' 'VVS2' 'VVS1' 'I1' 'IF']
Most frequent value in 'cut': Ideal
Most frequent value in 'color': G
Most frequent value in 'clarity': SI1
```

Question 4:

Identify and describe any data quality issues or inconsistencies within the data set. What steps would you take to clean and pre-processes the data to ensure its accuracy for further analysis

To clean and preprocess the data:

1. Handle missing values
2. Remove duplicates
3. Handle outliers
4. Convert data types

[28]:

```python
# Identify missing values
print("Missing values\n",df.isnull().sum())
# Identify duplicated records
print("Duplicated records\n",df.duplicated().sum())
# Identify outliers
```

```
print("Outliers\n",df.describe(include='all').loc['max'])
# Indetiify inconsistent data types
print("Inconsistent data types\n",df.dtypes)
```

```
Missing values
 carat       0
cut         0
color       0
clarity     0
depth       8
table       0
price       0
x          12
y           9
z          10
dtype: int64
Duplicated records
 149
Outliers
 carat         5.01
cut            NaN
color          NaN
clarity        NaN
depth         79.0
table         95.0
price      18823.0
x            10.74
y             58.9
z             31.8
Name: max, dtype: object
Inconsistent data types
 carat      float64
cut         object
color       object
clarity     object
depth      float64
table      float64
price        int64
x          float64
y          float64
z          float64
dtype: object
```

Question 5:

(i) Convert price in us dollar to rupees? (1 dollar = 80 rupees)

(ii) Create a new column called 'color_clarity_cut' and values are color+ '' *+clarity+* '' + cut? (Ex: E_ SI2_Ideal , E_ SI1_Premium)

```
[29]: # (i) Convert price in us dollar to rupees
      df['price'] = df['price'] * 80
      print(df)

      # (ii) Create a new column called 'color_clarity_cut'
      df['color_clarity_cut'] = df['color'] + '_' + df['clarity'] + '_' + df['cut']
      print(df)
```

```
       carat        cut color clarity  depth  table   price     x     y     z
0       0.23      Ideal     E     SI2   61.5   55.0   26080  3.95  3.98  2.43
1       0.21    Premium     E     SI1   59.8   61.0   26080  3.89  3.84  2.31
2       0.23       Good     E     VS1   56.9   65.0   26160  4.05  4.07  2.31
3       0.29    Premium     I     VS2   62.4   58.0   26720  4.20  4.23  2.63
4       0.31       Good     J     SI2   63.3   58.0   26800  4.34  4.35  2.75
…   …   … …  …  … …  … … … …
53938   0.86    Premium     H     SI2   61.0   58.0  220560  6.15  6.12  3.74
53939   0.75      Ideal     D     SI2   62.2   55.0  220560  5.83  5.87  3.64
53940   0.71    Premium     E     SI1   60.5   55.0  220480  5.79  5.74  3.49
53941   0.71    Premium     F     SI1   59.8   62.0  220480  5.74  5.73  3.43
53942   0.70  Very Good     E     VS2   60.5   59.0  220560  5.71  5.76  3.47

[53943 rows x 10 columns]
       carat        cut color clarity  depth  table   price     x     y     z  \
0       0.23      Ideal     E     SI2   61.5   55.0   26080  3.95  3.98  2.43
1       0.21    Premium     E     SI1   59.8   61.0   26080  3.89  3.84  2.31
2       0.23       Good     E     VS1   56.9   65.0   26160  4.05  4.07  2.31
3       0.29    Premium     I     VS2   62.4   58.0   26720  4.20  4.23  2.63
4       0.31       Good     J     SI2   63.3   58.0   26800  4.34  4.35  2.75
…   …   … …  …  … …  … … … …
53938   0.86    Premium     H     SI2   61.0   58.0  220560  6.15  6.12  3.74
53939   0.75      Ideal     D     SI2   62.2   55.0  220560  5.83  5.87  3.64
53940   0.71    Premium     E     SI1   60.5   55.0  220480  5.79  5.74  3.49
53941   0.71    Premium     F     SI1   59.8   62.0  220480  5.74  5.73  3.43
53942   0.70  Very Good     E     VS2   60.5   59.0  220560  5.71  5.76  3.47

      color_clarity_cut
0           E_SI2_Ideal
1         E_SI1_Premium
2            E_VS1_Good
3         I_VS2_Premium
4            J_SI2_Good
…                   …
53938     H_SI2_Premium
53939       D_SI2_Ideal
53940     E_SI1_Premium
53941     F_SI1_Premium
53942   E_VS2_Very Good
```

```
[53943 rows x 11 columns]
```

Question 6:

Check for any outliers in all numeric columns and then analyze carefully, how they should be addressed.

Outliers can be handled by :

1. Replaceing them with central tendencies.
2. Removing the outliers in case of extreme out of range.
3. Transforming the data by any of the methods such as dividing the number by larger number or applying log.

```
[30]:
# Select numeric columns
numeric_cols = df.select_dtypes(include=[np.number])

# Calculate IQR for each numeric column
Q1 = numeric_cols.quantile(0.25)
Q3 = numeric_cols.quantile(0.75)
IQR = Q3 - Q1

# Define a threshold for outliers
threshold = 1.5

# Identify outliers
outliers = ((numeric_cols < (Q1 - threshold * IQR)) | (numeric_cols > (Q3 +␣
 ↪threshold * IQR))).any(axis=1)

# Print outliers
print(df[outliers])

# Remove outliers
df = df[~outliers]
print(df)
```

```
       carat        cut color clarity  depth  table   price     x     y     z  \
2       0.23       Good     E     VS1   56.9   65.0   26160  4.05  4.07  2.31
8       0.22       Fair     E     VS2   65.1   61.0   26960  3.87  3.78  2.49
24      0.31  Very Good     J     SI1   58.1   62.0   28240  4.44  4.47  2.59
35      0.23       Good     F     VS1   58.2   59.0   32160  4.06  4.08  2.37
42      0.26       Good     D     VS2   65.2   56.0   32240  3.99  4.02  2.61
…        …          …    …      …     …     …      …      …     …     …
53882   0.71       Fair     D     VS1   65.4   59.0  219760  5.62  5.58  3.66
53886   0.70       Good     D     VS2   58.0   62.0  219920  5.78  5.87  3.38
53890   0.73       Good     E     SI1   57.9   55.0  219920  6.00  5.96  3.46
53895   0.70       Good     F     VS1   57.8   61.0  220080  5.83  5.79  3.36
53927   0.79       Good     F     SI1   58.1   59.0  220480  6.06  6.13  3.54
```

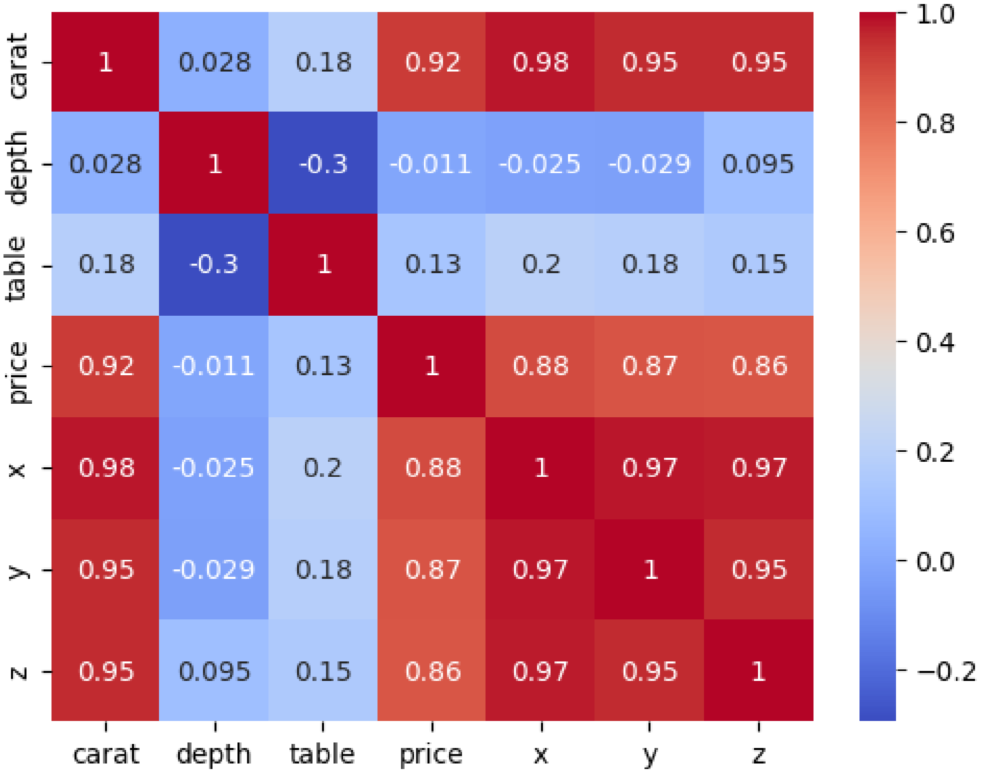
```
      color_clarity_cut
2             E_VS1_Good
8             E_VS2_Fair
24       J_SI1_Very Good
35            F_VS1_Good
42            D_VS2_Good
…                  …
53882         D_VS1_Fair
53886         D_VS2_Good
53890         E_SI1_Good
53895         F_VS1_Good
53927         F_SI1_Good

[6416 rows x 11 columns]
       carat        cut color clarity  depth  table   price     x     y     z  \
0       0.23      Ideal     E     SI2   61.5   55.0   26080  3.95  3.98  2.43
1       0.21    Premium     E     SI1   59.8   61.0   26080  3.89  3.84  2.31
3       0.29    Premium     I     VS2   62.4   58.0   26720  4.20  4.23  2.63
4       0.31       Good     J     SI2   63.3   58.0   26800  4.34  4.35  2.75
5       0.24  Very Good     J    VVS2   62.8   57.0   26880  3.94  3.96  2.48
…      …          …   …     …    …    …      …   …   …   …
53938   0.86    Premium     H     SI2   61.0   58.0  220560  6.15  6.12  3.74
53939   0.75      Ideal     D     SI2   62.2   55.0  220560  5.83  5.87  3.64
53940   0.71    Premium     E     SI1   60.5   55.0  220480  5.79  5.74  3.49
53941   0.71    Premium     F     SI1   59.8   62.0  220480  5.74  5.73  3.43
53942   0.70  Very Good     E     VS2   60.5   59.0  220560  5.71  5.76  3.47

      color_clarity_cut
0           E_SI2_Ideal
1         E_SI1_Premium
3         I_VS2_Premium
4            J_SI2_Good
5      J_VVS2_Very Good
…                  …
53938     H_SI2_Premium
53939       D_SI2_Ideal
53940     E_SI1_Premium
53941     F_SI1_Premium
53942   E_VS2_Very Good

[47527 rows x 11 columns]
```

Question 7:

Calculate the correlation (Using heat map) between price and all other numeric columns and list them in descending order and identify the highest and lowest correlation?

[31]:

```python
# Calculate correlation between numeric features
corr = numeric_cols.corr()

# Plot the correlation heatmap
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.show()

# List the correlation of all features with price in descending order
print(corr['price'].sort_values(ascending=False))

# Identify the highest and lowest correlation
print(f"Highest correlation: {corr['price'].max()}")
print(f"Lowest correlation: {corr['price'].min()}")
```



```
price    1.000000
carat    0.921591
x        0.884457
y        0.865413
z        0.861251
table    0.127118
depth   -0.010666
Name: price, dtype: float64
```

```
Highest correlation: 1.0
Lowest correlation: -0.01066551076228034
```

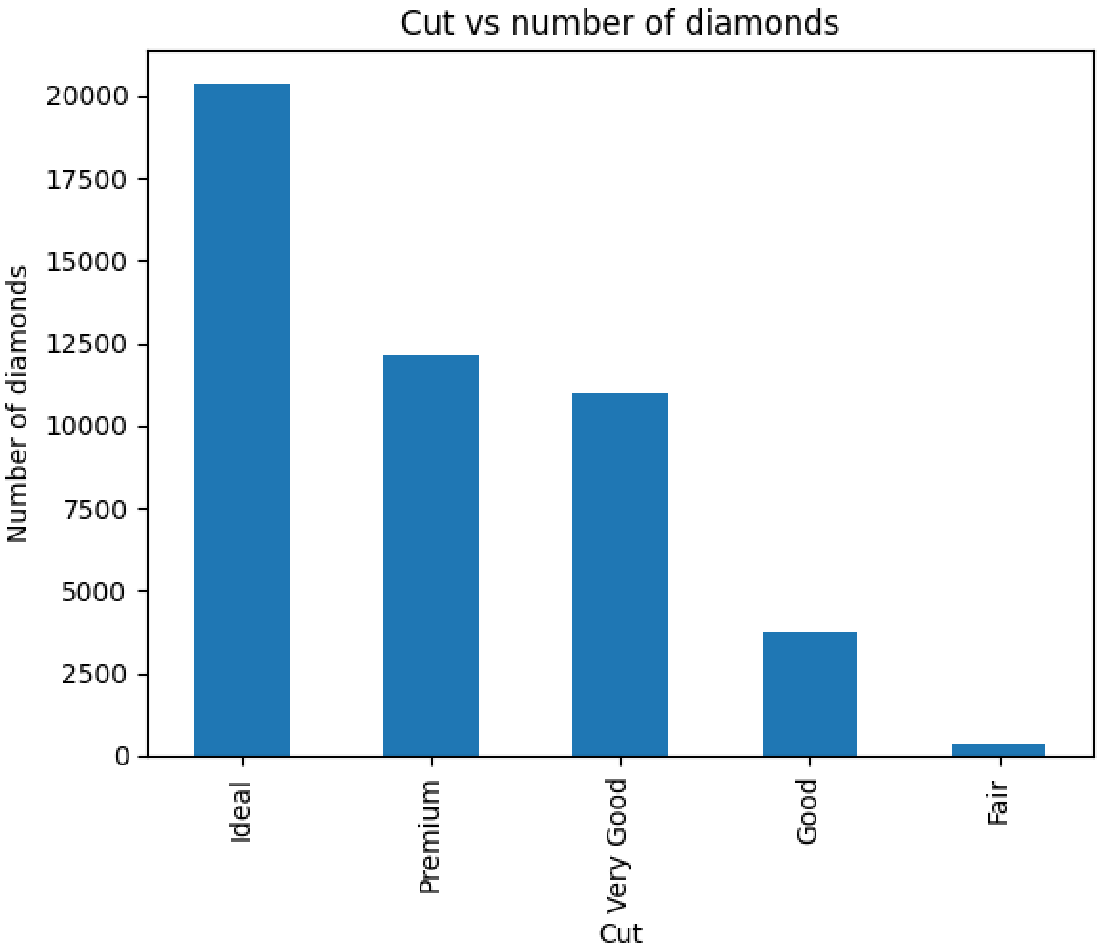
Question 8:

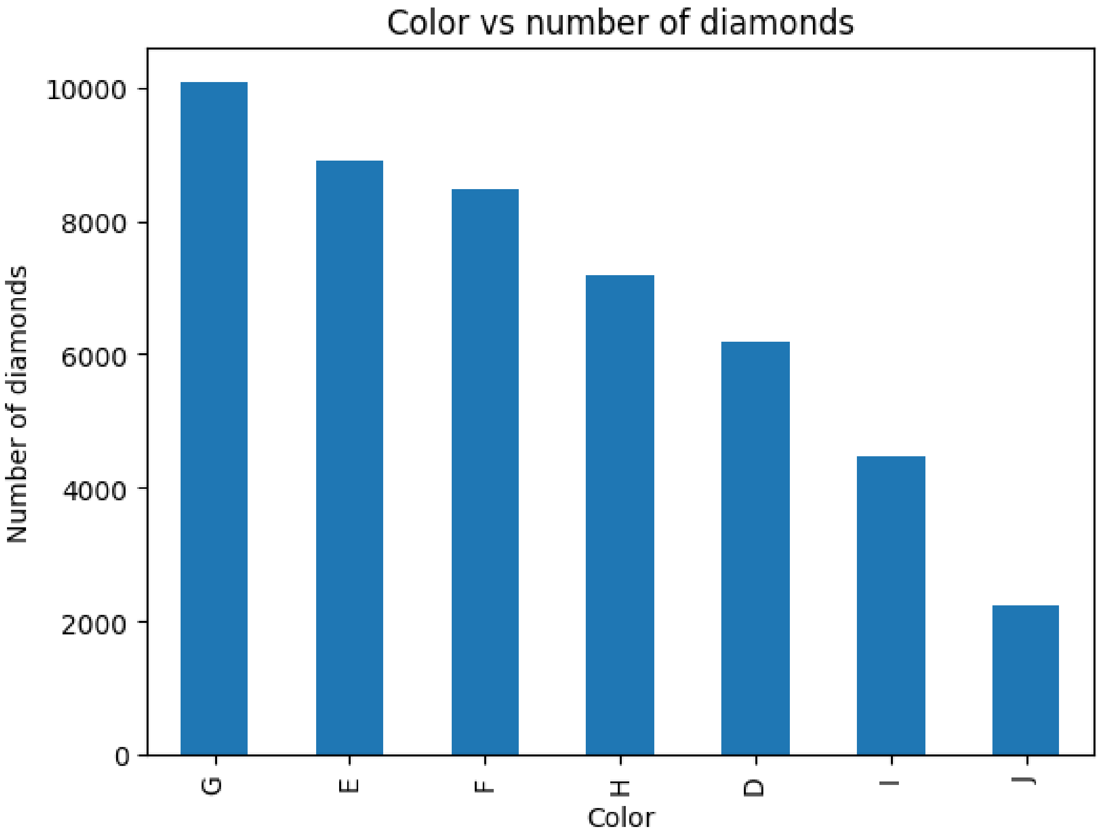
Draw bar plots, visualize and also indicate any insights can be obtained by taking X-axis vs Y-axis as:

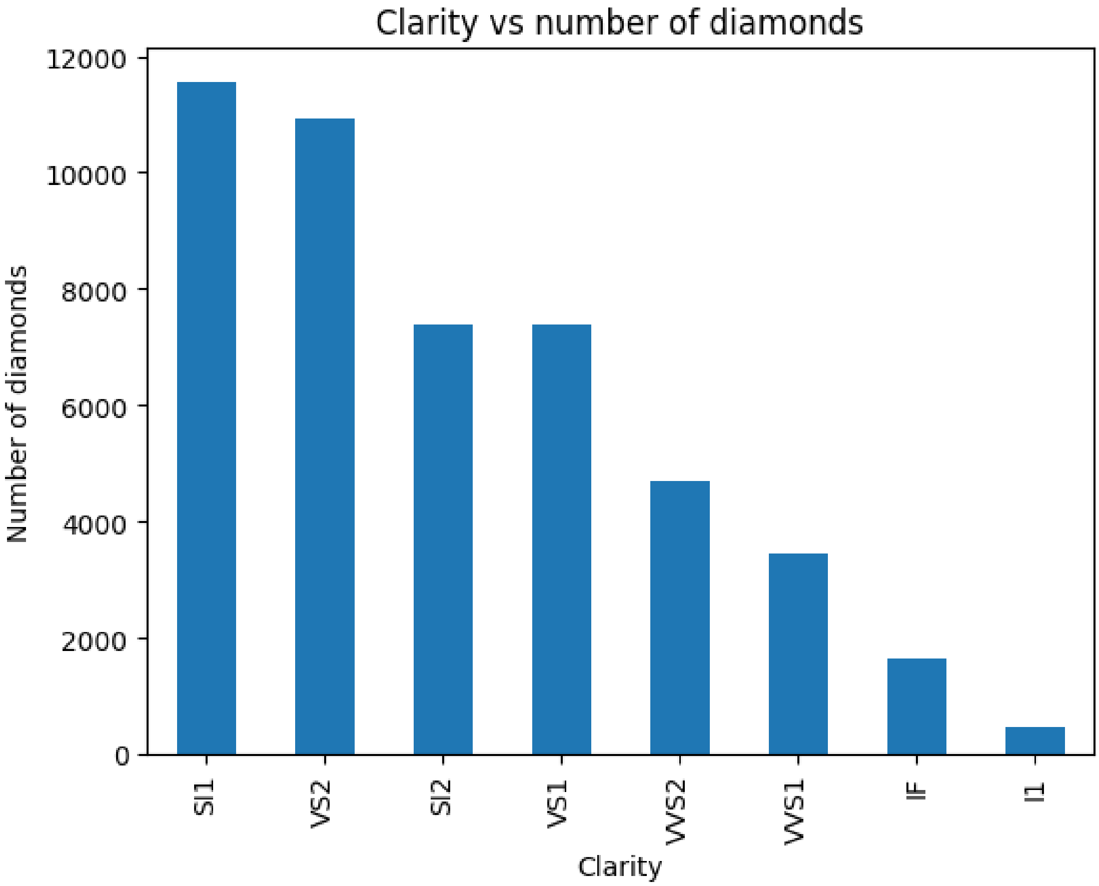
- Cut vs no.of diamonds
- Color vs no.of diamonds
- Clarity vs no.of diamonds

[32]:

```python
import matplotlib.pyplot as plt

# Cut vs number of diamonds
df['cut'].value_counts().plot(kind='bar')
plt.title('Cut vs number of diamonds')
plt.xlabel('Cut')
plt.ylabel('Number of diamonds')
plt.show()

# Color vs number of diamonds
df['color'].value_counts().plot(kind='bar')
plt.title('Color vs number of diamonds')
plt.xlabel('Color')
plt.ylabel('Number of diamonds')
plt.show()

# Clarity vs number of diamonds
df['clarity'].value_counts().plot(kind='bar')
plt.title('Clarity vs number of diamonds')
plt.xlabel('Clarity')
plt.ylabel('Number of diamonds')
plt.show()
```


Cut vs number of diamonds
Number of diamonds
20000
17500
15000
12500
10000
7500
5000
2500
0
Ideal
Premium
Very Good
Good
Fair
Cut


Color vs number of diamonds
Number of diamonds
10000
8000
6000
4000
2000
0
G
E
F
H
D
I
J
Color

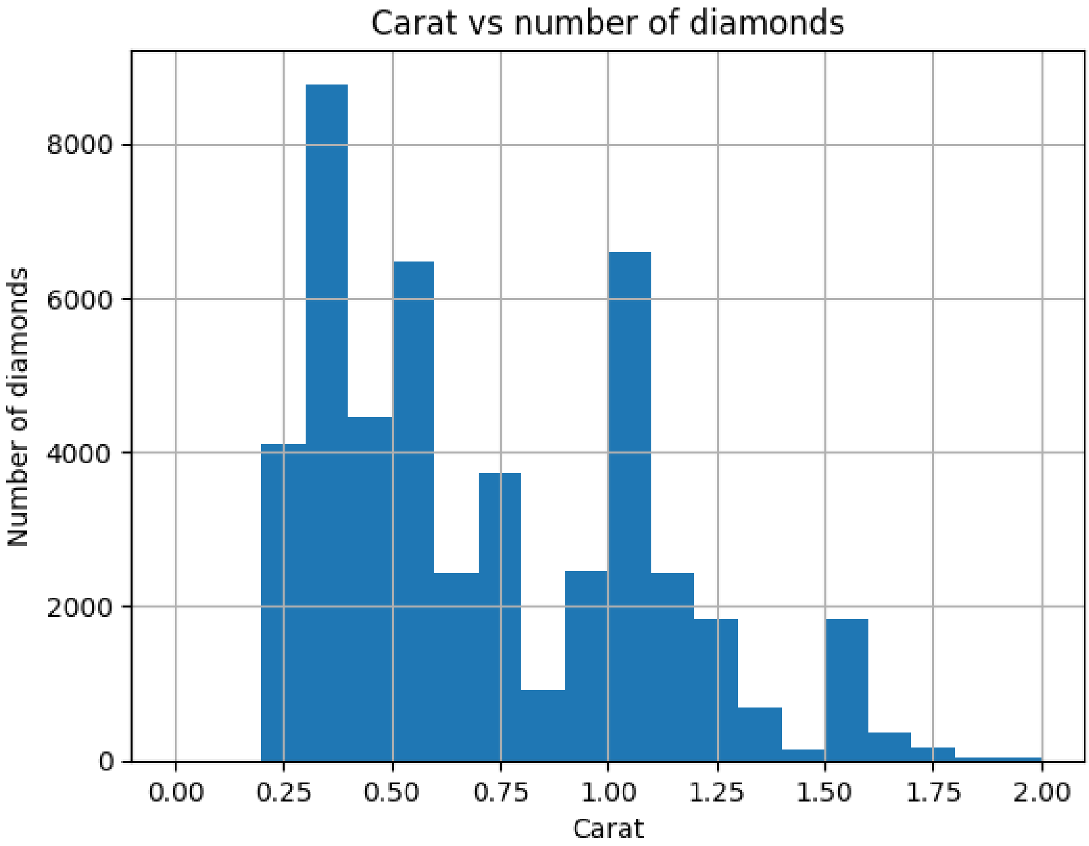


Question 9:

Draw a histogram where X-axis-> carat with interval size 0.1 and
Y-aixs-> no.of diamonds? and comment on it

[33]:

```
import numpy as np

# Draw a histogram
df['carat'].hist(bins=np.arange(0, df['carat'].max() + 0.1, 0.1))
plt.title('Carat vs number of diamonds')
plt.xlabel('Carat')
plt.ylabel('Number of diamonds')
plt.show()
```



The histogram will show the distribution of diamonds across different carat sizes.

Observations:

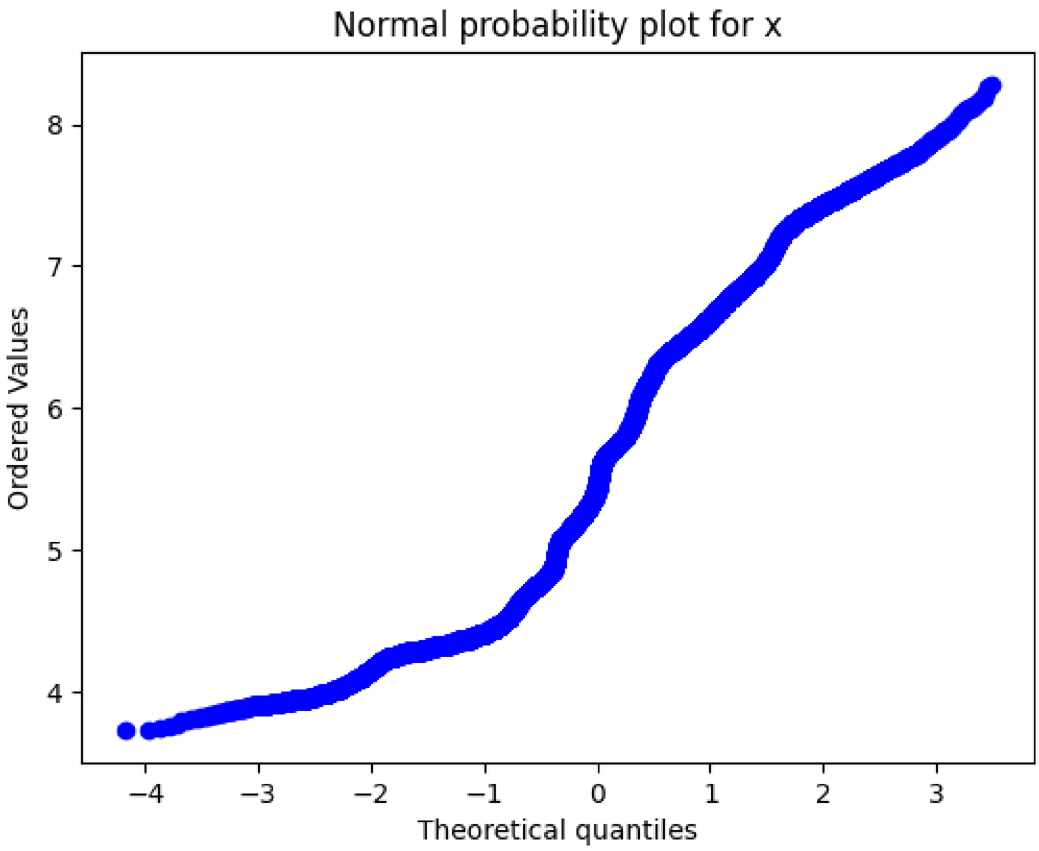
- The most common carat sizes among the diamonds.
- Whether the distribution is skewed towards smaller or larger carat sizes.

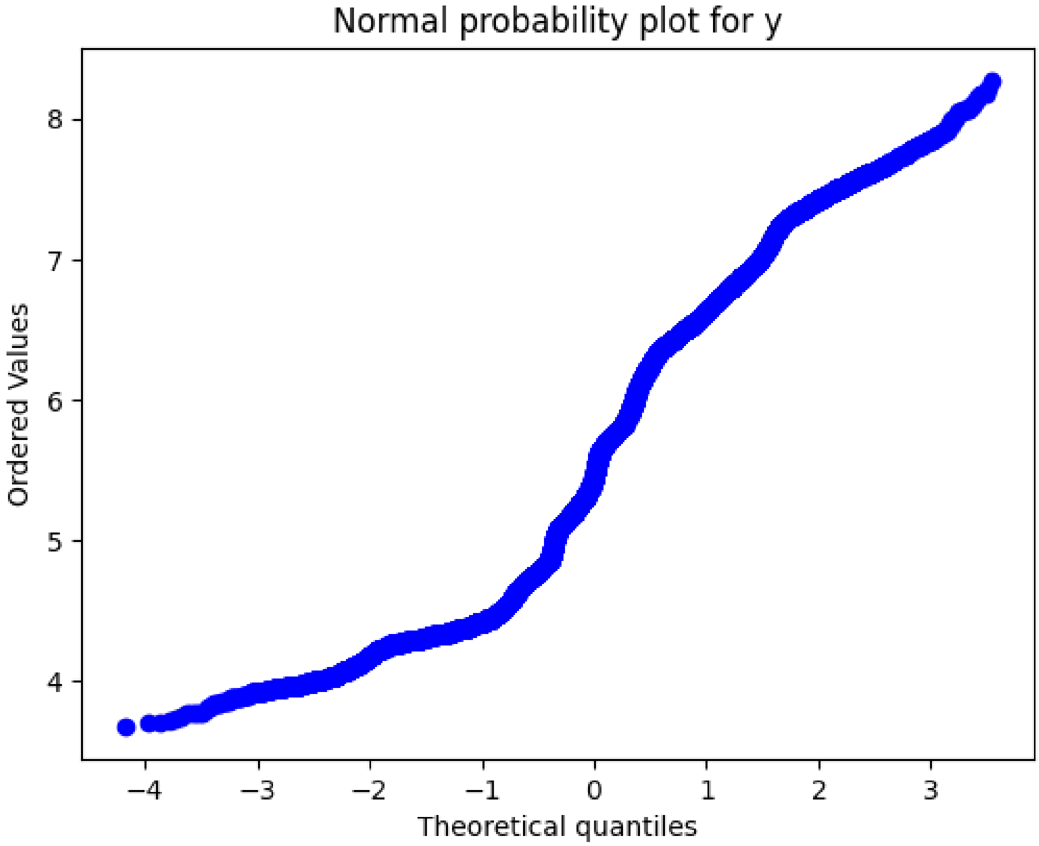
Question 10:

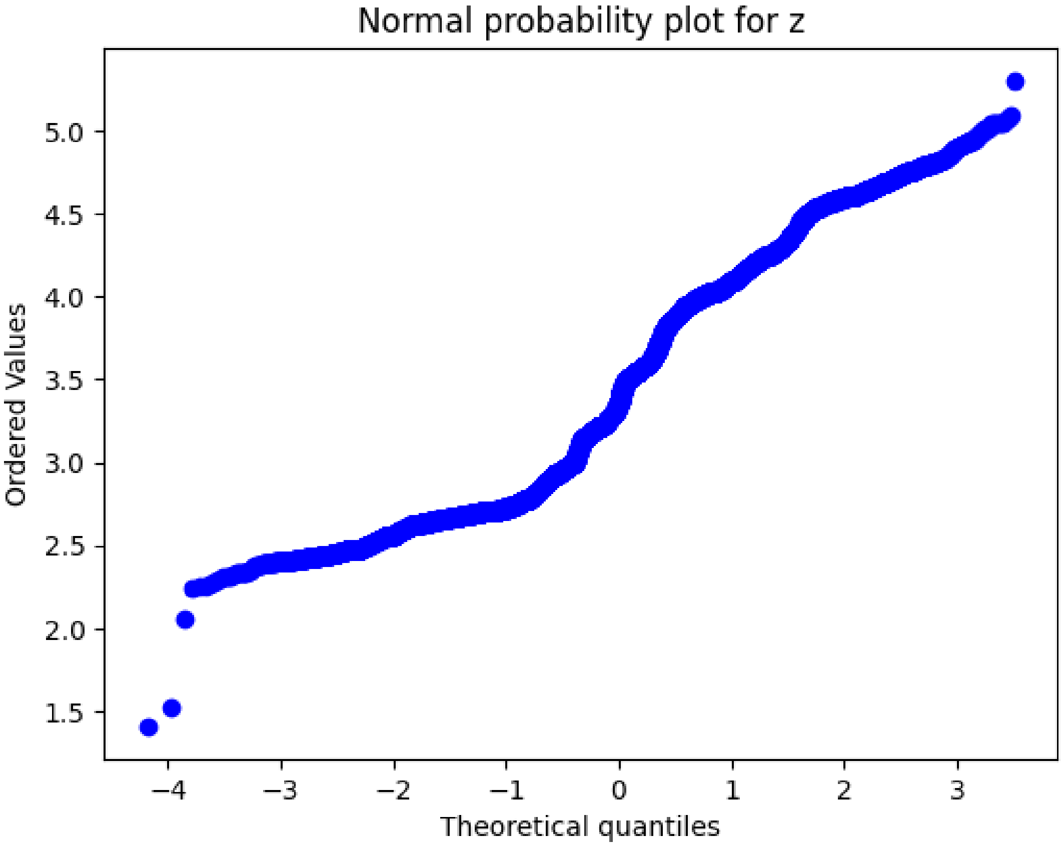
Draw a normal probability plot on X or Y or z? Based on the shape and trend of the plot? Is any conclusion can be drawn, if yes what it is?

[34]:
```
import scipy.stats as stats
import matplotlib.pyplot as plt

# Assuming df is your DataFrame

# Draw a normal probability plot for 'x'
stats.probplot(df['x'], plot=plt)
plt.title('Normal probability plot for x')
plt.show()
```

```
# Draw a normal probability plot for 'y'
stats.probplot(df['y'], plot=plt)
plt.title('Normal probability plot for y')
plt.show()

# Draw a normal probability plot for 'z'
stats.probplot(df['z'], plot=plt)
plt.title('Normal probability plot for z')
plt.show()
```




Normal probability plot for y
Ordered Values
Theoretical quantiles
8
7
6
5
4
−4
−3
−2
−1
0
1
2
3


Normal probability plot for z
Ordered Values
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
−4
−3
−2
−1
0
1
2
3
Theoretical quantiles