DASHBOARD PROJECT - GROUP 04 (DIAMONDS DATA SET)

Statistical Programming-ST 3011

P.G.D.K.Herath – s14923
M.Y.Kavinda – s15055
A.M.M.Fathir - S15109
A.D.W.S.V.A.Dasanayaka– s15032
M.M.P.K Perera - 14939

P.G.D.K.Herath – s14923

My contribution to the project can be broken down into two main areas:

1. Statistical analysis
2. Implementation via Python.

Regarding statistical analysis:

I was responsible for performing data cleaning and preparation. This involved identifying and handling missing values, removing duplicates, and dealing with outliers.

To handle missing values, I first used the **isna()** method to identify any columns with missing values, and since I could not find out any missing values I keep the variables as it is. This ensured that the dataset was complete.

To remove duplicates, I used the **duplicated()** method to identify any duplicate rows in the dataset, and then found out there are 146 duplicated rows in the dataset. Since the additional information are not given about the dataset (No primary key) I did not delete those duplicates. (There can be several diamonds which have the save attributes values as others.)

To handle outliers, I created boxplots for each numerical variable and identified the possibles outliers. Then using **Boolean masking** method, outliers were removed.

Regarding implementation via Python:

I was responsible for creating the callbacks for the dashboard. This involved connecting the dropdown and radio buttons with the graphs, although I did not create the actual graphs themselves.

To create the callbacks, I used the **@app.callback** decorator provided by Dash. I created a function for each callback, and used the input and output parameters to specify the components that should trigger the function and the components that should be updated by the function.

For example, to connect the dropdown menu with a graph, I created a function that takes the selected value of the dropdown menu as input, and updates the graph with the corresponding data. I used the dcc.Dropdown component as the input component, and the dcc.Graph component as the output component.

Similarly, to connect the radio buttons with a graph, I created a function that takes the selected value of the radio buttons as input, and updates the graph with the corresponding data. I used the dcc.RadioItems component as the input component, and the dcc.Graph component as the output component.

M.Y.Kavinda – s15055

For this dashboard project, I contributed my preliminary statistical analysis knowledge, Descriptive analysis knowledge, python coding knowledge and Graphical interface ideas with my group as my best and the best the team needs. So, below I explain the way I did my part of descriptive analysis and coding part.

Preliminary statistical analysis including descriptive analysis.

So, my part of Preliminary statistical analysis including descriptive analysis is to identify Quantitative variables and find the relationships of those quantitative variables. As well as I tried to find the most suitable graphical way to graph those relationships for the dashboard viewers. So, in our diamond data set there are 7 quantitative variables which are Carat, Depth, Table, Price, X, Y and Z. So, according to theoretical matters box plots and scatter plots are the most suitable graph. As well as through heatmap we can identify the correlation of each and every pair of quantitative variables. So, I decided to plot box plots for each and every quantitative variable with price variable as y axis for find relationship of those variables with price. As well as I plot boxplot for each and every quantitative variable for find five number summery of each variable and identify their variabilities. Then plot heatmap with including all quantitative variables.

The implementation via Python.

So, I wrote two codes for x, y, z variables with price and carat, depth, table variables with price for plot scatter plots using **px.scatter** with other features by naming x_y_z_graphs and carat_depth_table_graphs. The special thing is that in this both plots show the behavior of three variables in one graph and used separate colors to show separate variables. Then I wrote a code for boxplot for each and every quantitative variable by using **px.box** with other features. In here as well I plot only one boxplot and used radio buttons to change variable as viewer wish. So, for the radio button I used **dcc.RadioItems** and by option feature used separate labels for show all variables. Then the viewer can look any variable by just click the name of that variable. Then for heatmap I used **df.corr** with other features and used all quantitative variables to compare all of them withing one graph.

A.M.M.Fathir - S15109

As a group member in the project, my role was to analyze the qualitative variables present in the Diamonds dataset and provide valuable insights using Python and the Dash framework. The focus was on three ordinal variables: "cut" quality, "color" of the diamond, and "clarity" measurement.

To effectively analyze and communicate the findings, decided to create bar charts for each qualitative variable, highlighting their distributions and patterns within the dataset.

Cut Quality:

The cut quality is an important factor in determining a diamond's brilliance and overall appeal. To visualize this variable, a bar chart was created for different categories of cut quality, namely Fair, Good, Very Good, Premium, and Ideal, on the x-axis. The y-axis represents the count of diamonds falling into each category. This bar chart helps us understand the distribution of diamonds across different cut qualities, allowing us to assess the overall quality of the dataset.

Diamond Color:

Diamond color plays a significant role in determining its value and beauty. To present the distribution of diamond colors, another bar chart was constructed. The x-axis represents the various color categories, ranging from J (worst) to D (best), while the y-axis denotes the count of diamonds belonging to each color category. This bar chart provides a clear overview of the distribution of diamond colors in the dataset, enabling us to observe any predominant color trends.

Clarity:

Clarity refers to the visual appearance of inclusions or flaws within a diamond. To analyze the clarity variable, a bar chart was created to illustrate the different clarity categories: I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, and IF (best). The x-axis represents these clarity categories, while the y-axis represents the count of diamonds falling into each category. This bar chart allows us to visualize the distribution of clarity levels among the diamonds in the dataset and identify any patterns.

Overall, my contribution to the project centered around the analysis of qualitative variables using Python and the Dash framework. By creating insightful bar charts for the cut quality, diamond color, and clarity variables, I helped the team gain a deeper understanding of the dataset's qualitative aspects.

A.D.W.S.V.A.Dasanayaka– s15032

As a member of the group assigned to work on the diamond dataset, I have actively participated in the preliminary data analysis and the implementation of a dashboard using Python. My contributions are highlighted below.

Descriptive Statistical Analysis:

During the preliminary data analysis stage, I worked on identifying the distribution of numerical variables and the variation of categorical variables with the price. I conducted a thorough analysis of the dataset to understand the variables that are suitable for performing hypothesis testing. I also performed feature engineering to create new variables that can potentially improve the accuracy of the analysis. one example of feature engineering could be calculating the volume of the diamond using the dimensions.

Regarding implementation via Python:

I was responsible for creating the layout for the dashboard using Dash Bootstrap components in Python. This involved creating two pages, one for visualizing graphs and the other for showing the diamond dataset as a data table. I also included a navbar to navigate between the pages.

To create the navbar, I used the **dbc.Navbar** component and added links to each page.

On the graph page, I used the **dbc.Row** and **dbc.Col** components to create a grid layout for the graphs. I then used placeholders to show where the graphs would be placed. These placeholders would be updated with the actual graphs once they were created by other members of the group.

On the data table page, I used the **dbc.Table** component to create a table that displays the diamond dataset. I used the pandas library to read the dataset into a pandas dataframe, and then used the dbc.Table component to display the dataframe.

Overall, my contributions to the project have helped ensure that the dashboard has a user-friendly layout with easy navigation between pages. The placeholder components for the graphs and the data table components are ready to be updated with the actual graphs and data table created by other members of the group.

M.M.P.K Perera - 14939

As a member of this group project, my main contributions were in performing descriptive statistical analysis of the dataset and creating the cards for the dashboard that show the number of diamonds, max price, min price, and median price.

Descriptive Statistical Analysis:

In the descriptive statistical analysis part, I explored the dataset and performed the following analysis:

- Calculated the mean, median, and mode of the diamond prices.
- Calculated the range, variance, and standard deviation of the diamond prices.
- Created frequency tables for the categorical variables like cut, color, and clarity.

Dashboard Creation:

For the dashboard creation, I designed the layout for the cards to display the following information:

- Total number of diamonds in the dataset
- Maximum price of a diamond in the dataset
- Minimum price of a diamond in the dataset
- Median price of a diamond in the dataset

I used Dash Bootstrap components to create the layout for the data cards.

Codes and results for the preliminary analysis & descriptive analysis

In [46]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

In [47]:
```python
Initial_df = pd.read_csv("diamonds.csv")
df = Initial_df.iloc[:,1:]
```

In [48]:
```python
df.head()
```

Out[48]:

	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

Missing values

In [49]:
```python
df.isna().sum()
```

Out[49]:
```
carat       0
cut         0
color       0
clarity     0
depth       0
table       0
price       0
x           0
y           0
z           0
dtype: int64
```

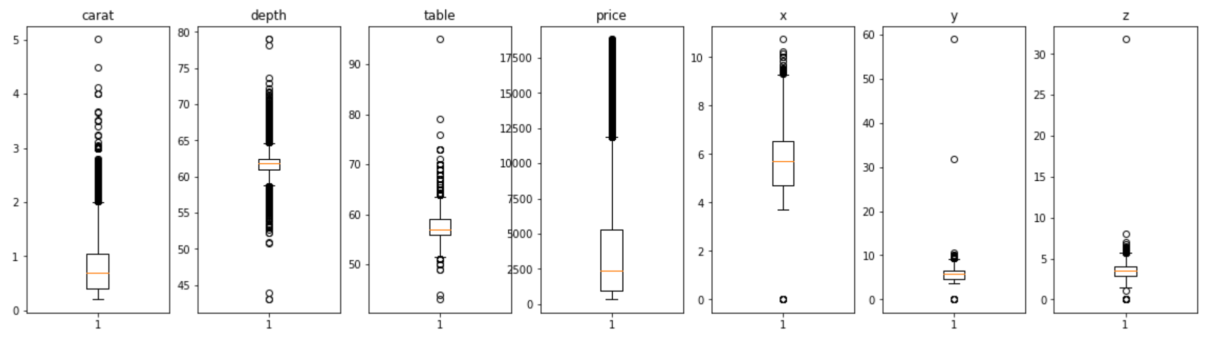
Duplicate rows

In [50]:
```python
df.duplicated().sum()
```

Out[50]:
```
146
```

Identifying outliers

In [51]:
```python
num_df = df.select_dtypes(include=['float64', 'int64'])

# Set the figure size
fig, axs = plt.subplots(ncols=len(num_df.columns), figsize=(20,5))

# Create boxplots for each numerical variable
for i, col in enumerate(num_df.columns):
    axs[i].boxplot(num_df[col])
    axs[i].set_title(col)

plt.show()
```



Removing outliers

In [59]:
```python
df = df[(df['y']>0) & (df['x']>0) & (df['z']>0)]
df = df[df['y']<20]
df = df[df['z']<10]
df = df[(df['depth']<75) & (df['depth']>50)]
df = df[(df['table']<80) & (df['table']>45)]
```

Descriptive analysis

In [60]:
```python
df.describe()
```

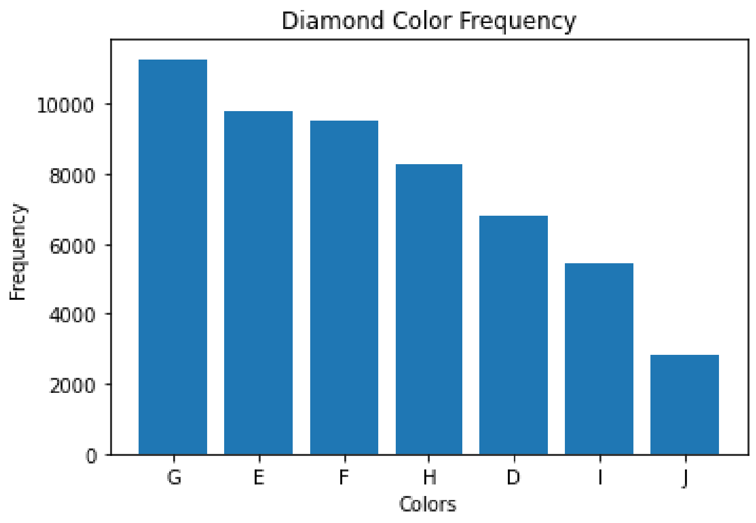
Out[60]:

	carat	depth	table	price	x	y	
count	53908.000000	53908.000000	53908.000000	53908.000000	53908.000000	53908.000000	53908.00(
mean	0.797663	61.749662	57.456366	3930.866996	5.731555	5.733384	3.53(
std	0.473774	1.420106	2.224566	3987.282422	1.119401	1.111271	0.69
min	0.200000	50.800000	49.000000	326.000000	3.730000	3.680000	1.07(
25%	0.400000	61.000000	56.000000	949.000000	4.710000	4.720000	2.91(
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.53(
75%	1.040000	62.500000	59.000000	5324.000000	6.540000	6.540000	4.04(
max	5.010000	73.600000	79.000000	18823.000000	10.740000	10.540000	6.98(

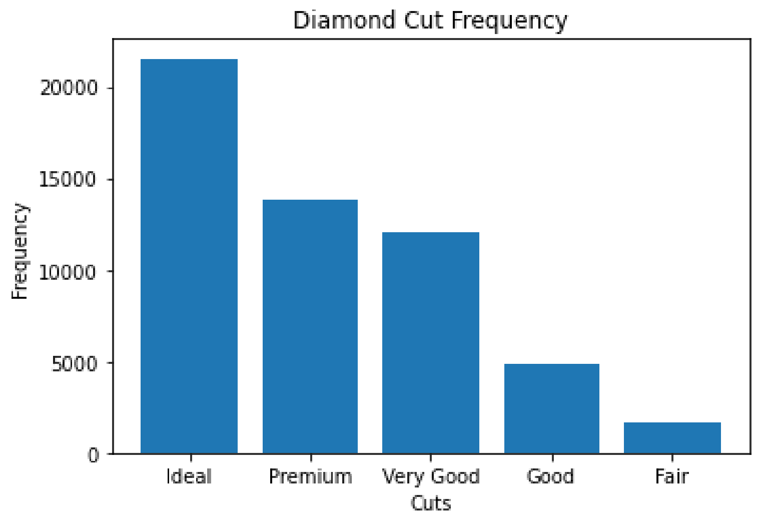
Categorical variables analysis

In [61]:

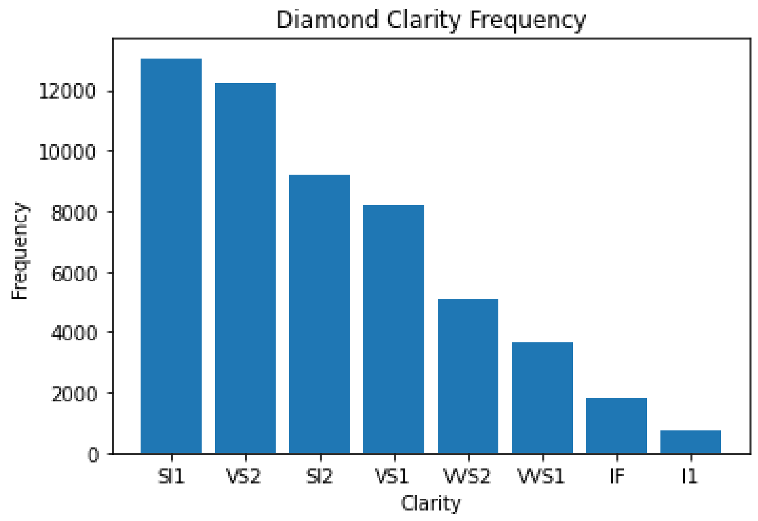
```
value2 = df.color.value_counts()
color_count = pd.DataFrame(value2)
color_count.reset_index(inplace=True)
color_count.rename(columns={"index":"color","color": "frequency"}, inplace=Tru

fig, ax = plt.subplots()
ax.bar(color_count["color"], color_count["frequency"])
ax.set_xlabel("Colors")
ax.set_ylabel("Frequency")
ax.set_title("Diamond Color Frequency")
plt.show()
```



In [62]:
```python
value3 = df.cut.value_counts()
cut_count = pd.DataFrame(value3)
cut_count.reset_index(inplace=True)
cut_count.rename(columns={"index":"cut","cut": "frequency"}, inplace=True)

fig, ax = plt.subplots()
ax.bar(cut_count["cut"], cut_count["frequency"])
ax.set_xlabel("Cuts")
ax.set_ylabel("Frequency")
ax.set_title("Diamond Cut Frequency")
plt.show()
```



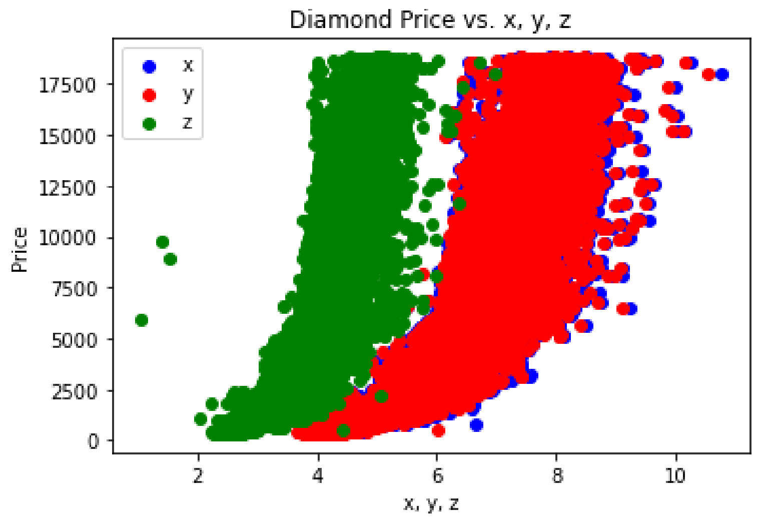
In [63]:
```python
value4 = df.clarity.value_counts()
clarity_count = pd.DataFrame(value4)
clarity_count.reset_index(inplace=True)
clarity_count.rename(columns={"index":"clarity","clarity": "frequency"}, inpla

fig, ax = plt.subplots()
ax.bar(clarity_count["clarity"], clarity_count["frequency"])
ax.set_xlabel("Clarity")
ax.set_ylabel("Frequency")
ax.set_title("Diamond Clarity Frequency")
plt.show()
```



Numerical varibles analysis

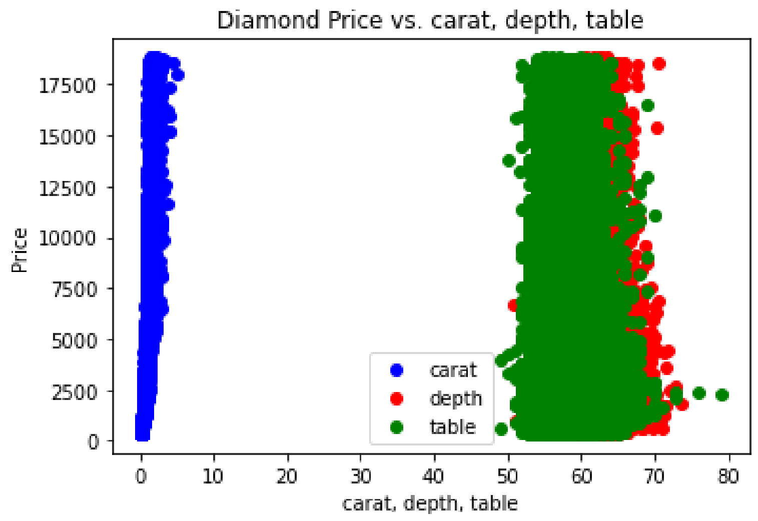
In [64]:

```
fig, ax = plt.subplots()
ax.scatter(df["x"], df["price"], color="blue", label="x")
ax.scatter(df["y"], df["price"], color="red", label="y")
ax.scatter(df["z"], df["price"], color="green", label="z")
ax.set_xlabel("x, y, z")
ax.set_ylabel("Price")
ax.set_title("Diamond Price vs. x, y, z")
ax.legend()
plt.show()
```

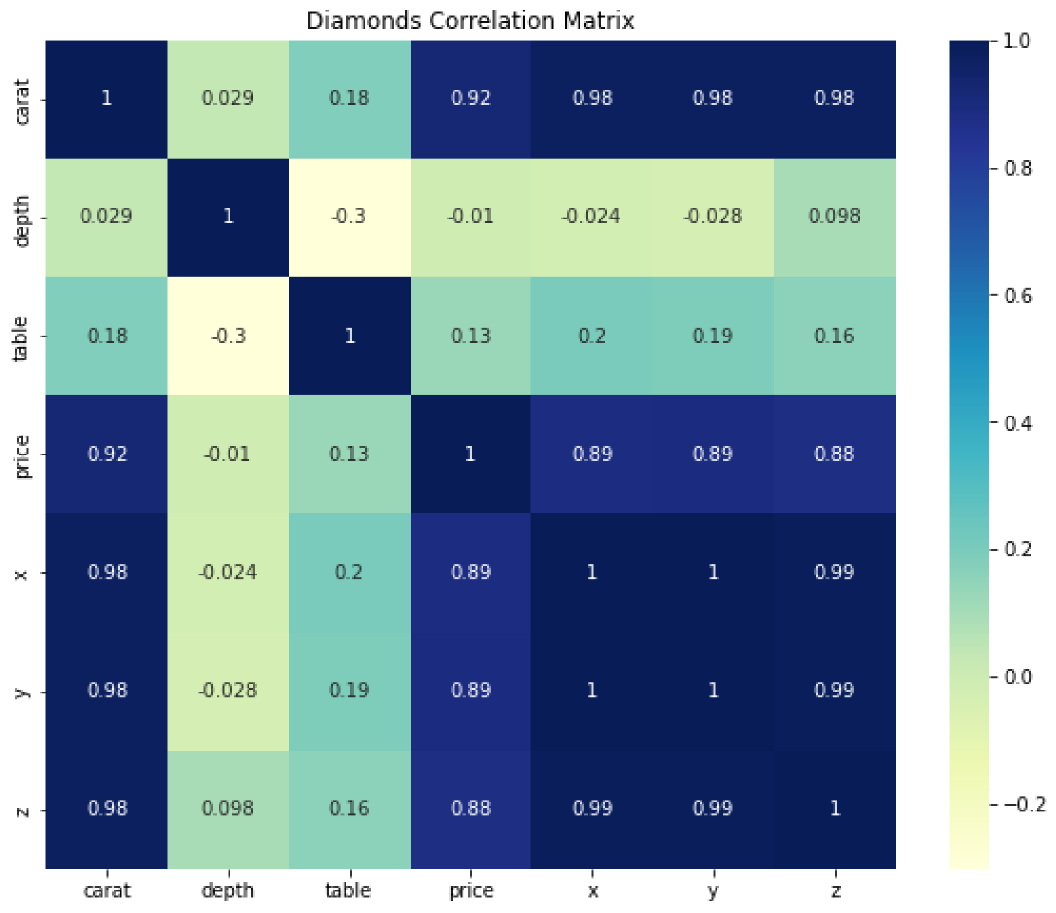


In [65]:

```
fig, ax = plt.subplots()
ax.scatter(df["carat"], df["price"], color="blue", label="carat")
ax.scatter(df["depth"], df["price"], color="red", label="depth")
ax.scatter(df["table"], df["price"], color="green", label="table")
ax.set_xlabel("carat, depth, table")
ax.set_ylabel("Price")
ax.set_title("Diamond Price vs. carat, depth, table")
ax.legend()
plt.show()
```

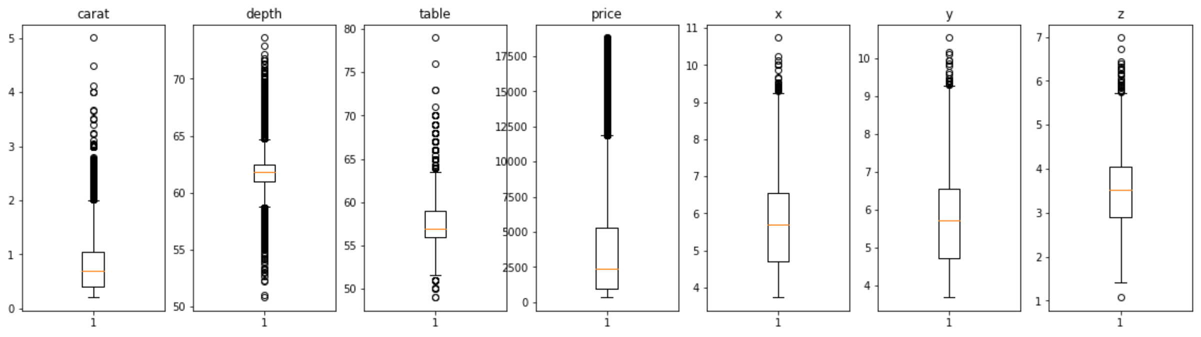


In [66]:
```python
corr_matrix = df.corr()

fig, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap="YlGnBu", ax=ax)
ax.set_title("Diamonds Correlation Matrix")
plt.show()
```



	carat	depth	table	price	x	y	z
carat	1	0.029	0.18	0.92	0.98	0.98	0.98
depth	0.029	1	-0.3	-0.01	-0.024	-0.028	0.098
table	0.18	-0.3	1	0.13	0.2	0.19	0.16
price	0.92	-0.01	0.13	1	0.89	0.89	0.88
x	0.98	-0.024	0.2	0.89	1	1	0.99
y	0.98	-0.028	0.19	0.89	1	1	0.99
z	0.98	0.098	0.16	0.88	0.99	0.99	1

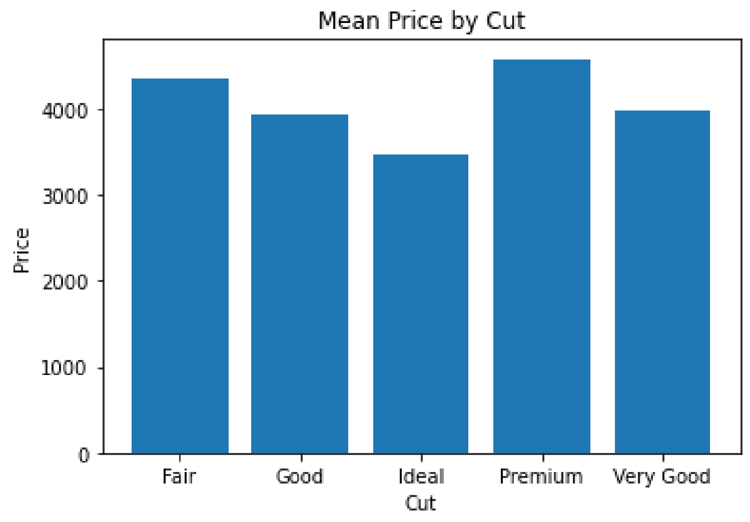
In [67]:
```python
num_df = df.select_dtypes(include=['float64', 'int64'])

# Set the figure size
fig, axs = plt.subplots(ncols=len(num_df.columns), figsize=(20,5))

# Create boxplots for each numerical variable
for i, col in enumerate(num_df.columns):
    axs[i].boxplot(num_df[col])
    axs[i].set_title(col)

plt.show()
```



Other informations

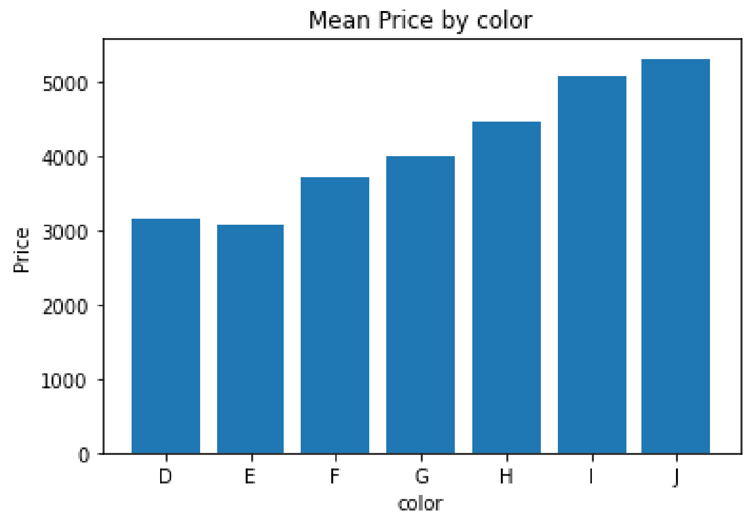
In [68]:

```python
bar_chart_data = df.groupby(['cut'])['price'].mean().reset_index()
fig, ax = plt.subplots()
ax.bar(bar_chart_data['cut'], bar_chart_data['price'])
ax.set_xlabel('Cut')
ax.set_ylabel('Price')
ax.set_title('Mean Price by Cut')
plt.show()
```



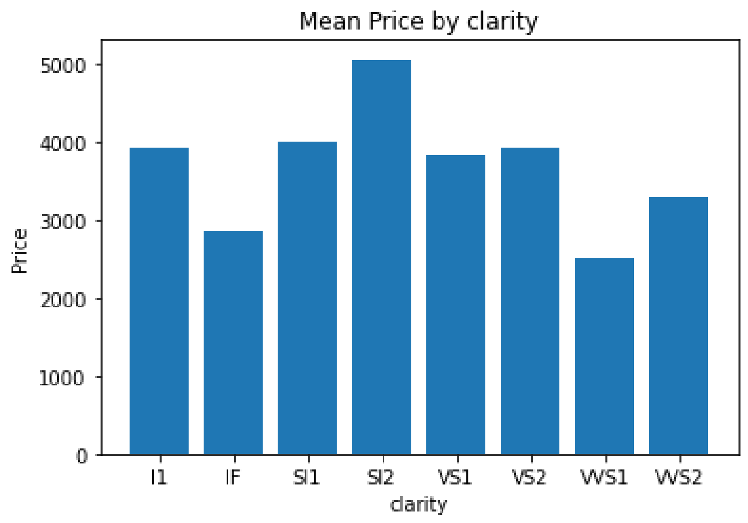
In [69]:

```python
bar_chart_data = df.groupby(['color'])['price'].mean().reset_index()
fig, ax = plt.subplots()
ax.bar(bar_chart_data['color'], bar_chart_data['price'])
ax.set_xlabel('color')
ax.set_ylabel('Price')
ax.set_title('Mean Price by color')
plt.show()
```



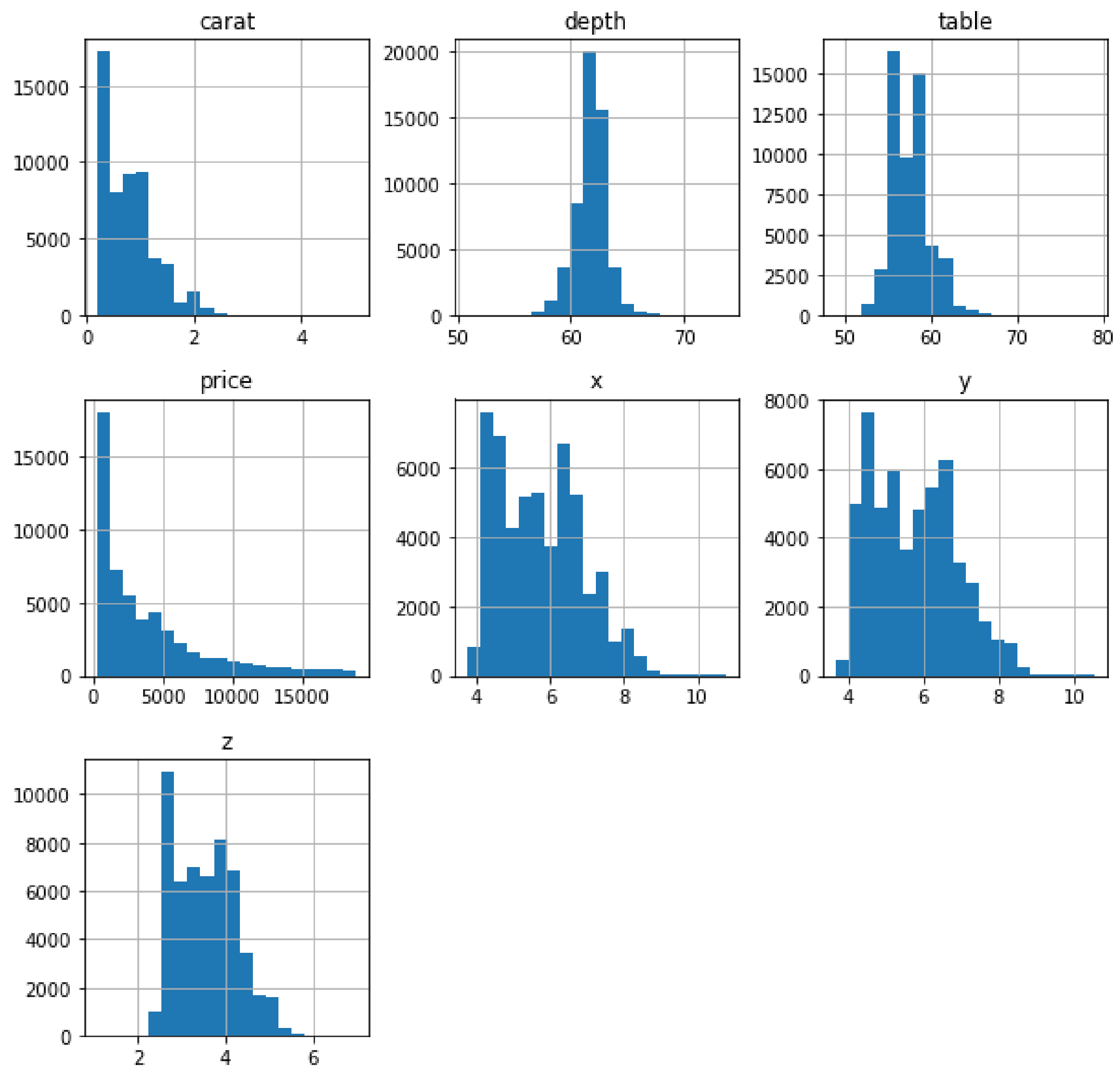
In [70]:

```python
bar_chart_data = df.groupby(['clarity'])['price'].mean().reset_index()
fig, ax = plt.subplots()
ax.bar(bar_chart_data['clarity'], bar_chart_data['price'])
ax.set_xlabel('clarity')
ax.set_ylabel('Price')
ax.set_title('Mean Price by clarity')
plt.show()
```



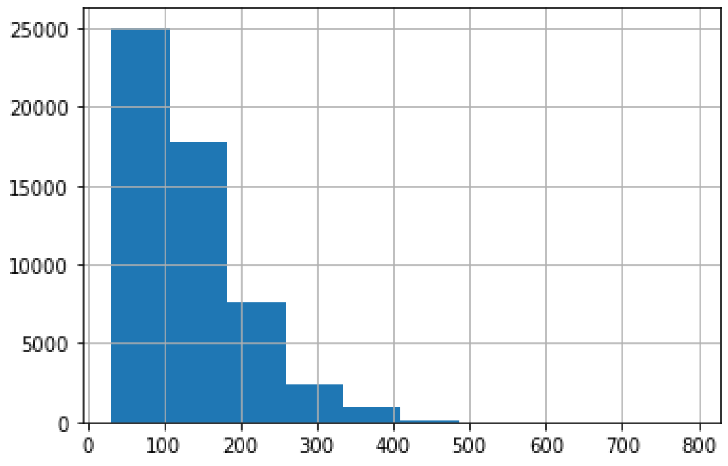
In [71]:

```
df.hist(bins=20, figsize=(10,10))
plt.show()
```



In [72]:
```
# perform feature engineering
volume = df['x'] * df['y'] * df['z']
volume.hist()
```

Out[72]:
```
<AxesSubplot:>
```


25000
20000
15000
10000
5000
0
0
100
200
300
400
500
600
700
800