In [3]:
```python
import pandas as pd

df = pd.read_csv('diamonds.csv')
df.head()
```

Out[3]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

Dataset values

In [23]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  53940 non-null  int64  
 1   carat       53940 non-null  float64
 2   cut         53940 non-null  object 
 3   color       53940 non-null  object 
 4   clarity     53940 non-null  object 
 5   depth       53940 non-null  float64
 6   table       53940 non-null  float64
 7   price       53940 non-null  int64  
 8   x           53940 non-null  float64
 9   y           53940 non-null  float64
 10  z           53940 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 4.5+ MB
```

No null values, 6 float values, 2 int values and 3 object values

In [24]:
```python
df.describe()
```

Out[24]:

	Unnamed: 0	carat	depth	table	price	
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000
mean	26970.500000	0.797940	61.749405	57.457184	3932.799722	5.731
std	15571.281097	0.474011	1.432621	2.234491	3989.439738	1.121
min	1.000000	0.200000	43.000000	43.000000	326.000000	0.000
25%	13485.750000	0.400000	61.000000	56.000000	950.000000	4.710
50%	26970.500000	0.700000	61.800000	57.000000	2401.000000	5.700
75%	40455.250000	1.040000	62.500000	59.000000	5324.250000	6.540
max	53940.000000	5.010000	79.000000	95.000000	18823.000000	10.740

Description of numeric values

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
Unnamed: 0    0
carat         0
cut           0
color         0
clarity       0
depth         0
table         0
price         0
x             0
y             0
z             0
dtype: int64
```

No null values

In [7]:
```
print("Rows",df.shape[0]);
print("Columns",df.shape[1]);
print("Column names:", df.columns.tolist());
```

```
Rows 53940
Columns 11
Column names: ['Unnamed: 0', 'carat', 'cut', 'color', 'clarity', 'depth', 'tabl
e', 'price', 'x', 'y', 'z']
```

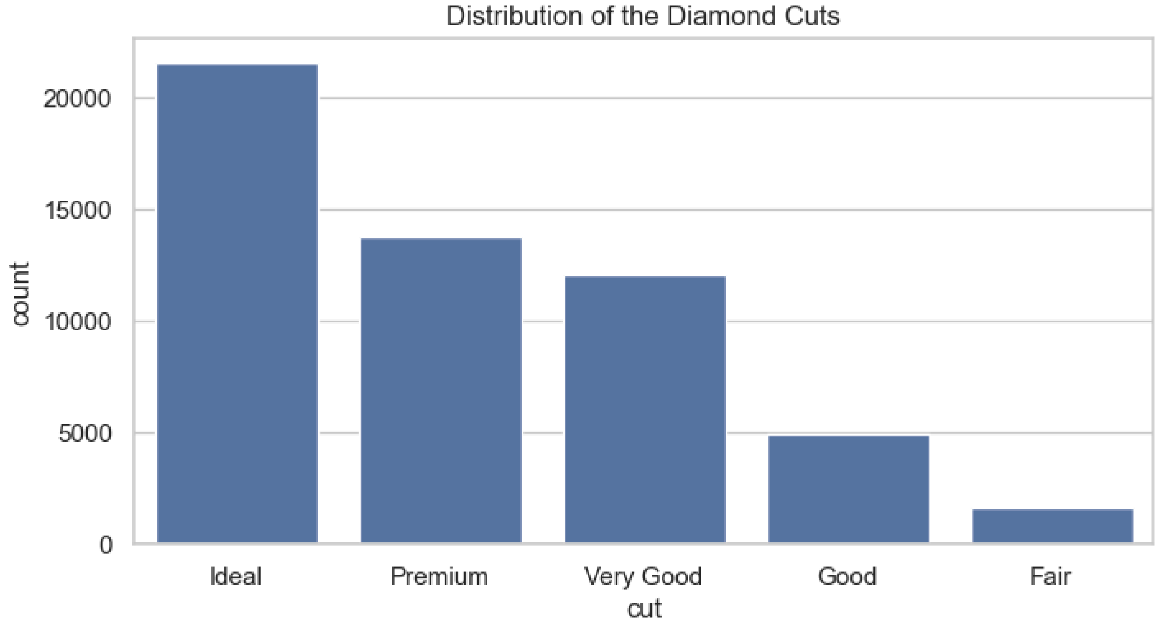
Number of rows and columns with names of the columns

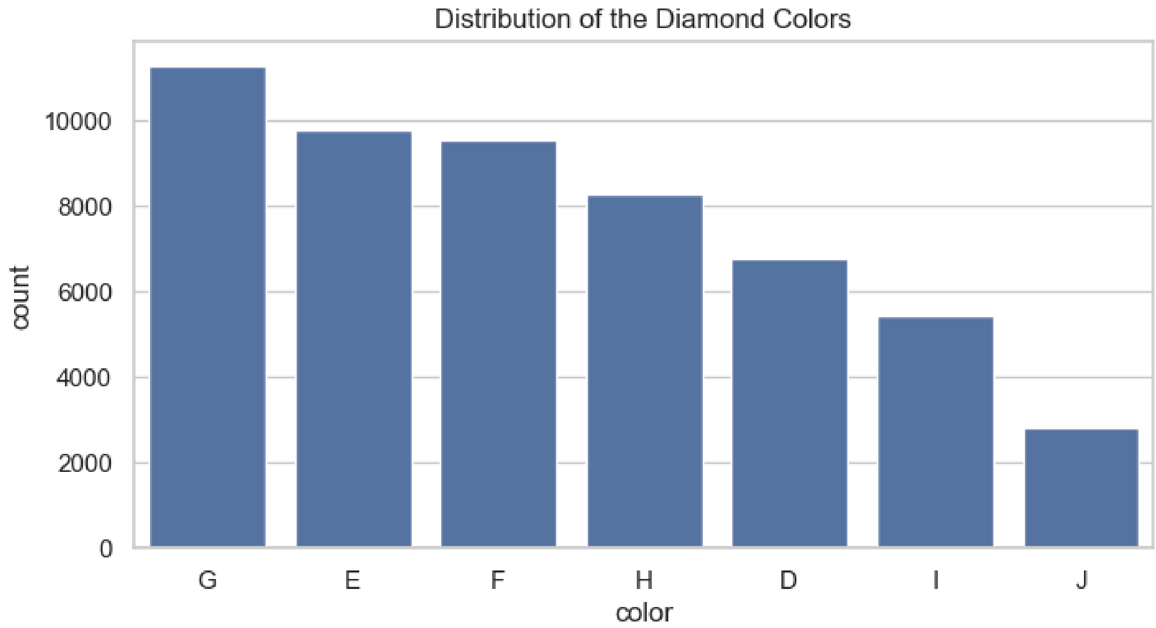
In [8]:
```
print("Cut\n",df['cut'].value_counts(),"\n");
print("Color\n",df['color'].value_counts(),"\n");
print("Clarity\n",df['clarity'].value_counts(),"\n");
```

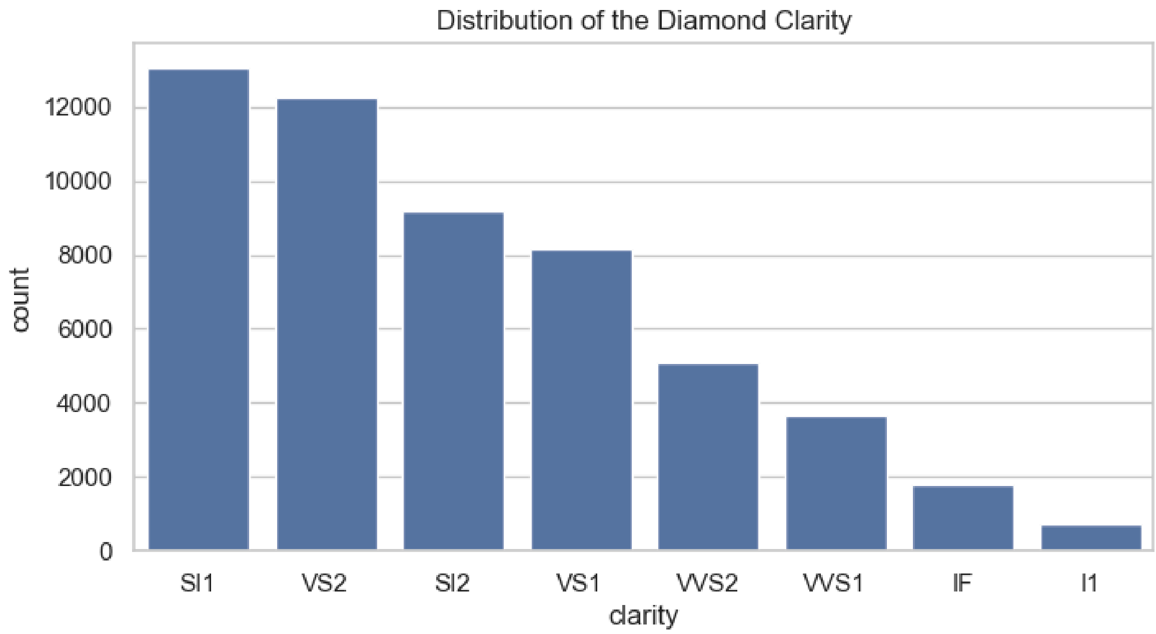
```
Cut
 cut
Ideal        21551
Premium      13791
Very Good    12082
Good          4906
Fair          1610
Name: count, dtype: int64

Color
 color
G    11292
E     9797
F     9542
H     8304
D     6775
I     5422
J     2808
Name: count, dtype: int64

Clarity
 clarity
SI1     13065
VS2     12258
SI2      9194
VS1      8171
VVS2     5066
VVS1     3655
IF       1790
I1        741
Name: count, dtype: int64
```

Count of different diamonds based on types cut,color and clarity

In [10]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style="whitegrid");

plt.figure(figsize=(8,4));
sns.countplot(x='cut',data=df,order=df['cut'].value_counts().index);
plt.title("Distribution of the Diamond Cuts");
plt.show();

plt.figure(figsize=(8,4));
sns.countplot(x='color',data=df,order=df['color'].value_counts().index);
plt.title("Distribution of the Diamond Colors");
plt.show();

plt.figure(figsize=(8,4));
sns.countplot(x='clarity',data=df,order=df['clarity'].value_counts().index);
plt.title("Distribution of the Diamond Clarity");
plt.show();
```

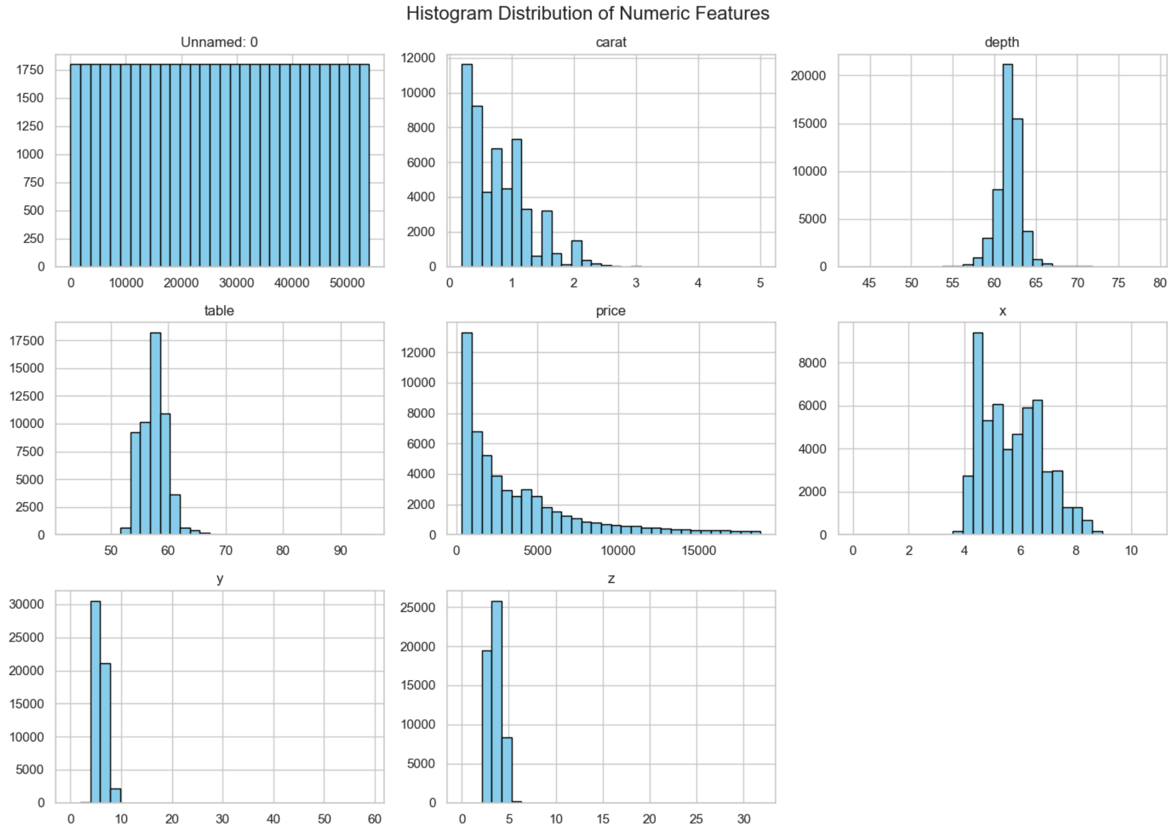

Distribution of the Diamond Cuts
count
20000
15000
10000
5000
0
Ideal
Premium
Very Good
Good
Fair
cut


Distribution of the Diamond Colors
count
10000
8000
6000
4000
2000
0
G
E
F
H
D
I
J
color

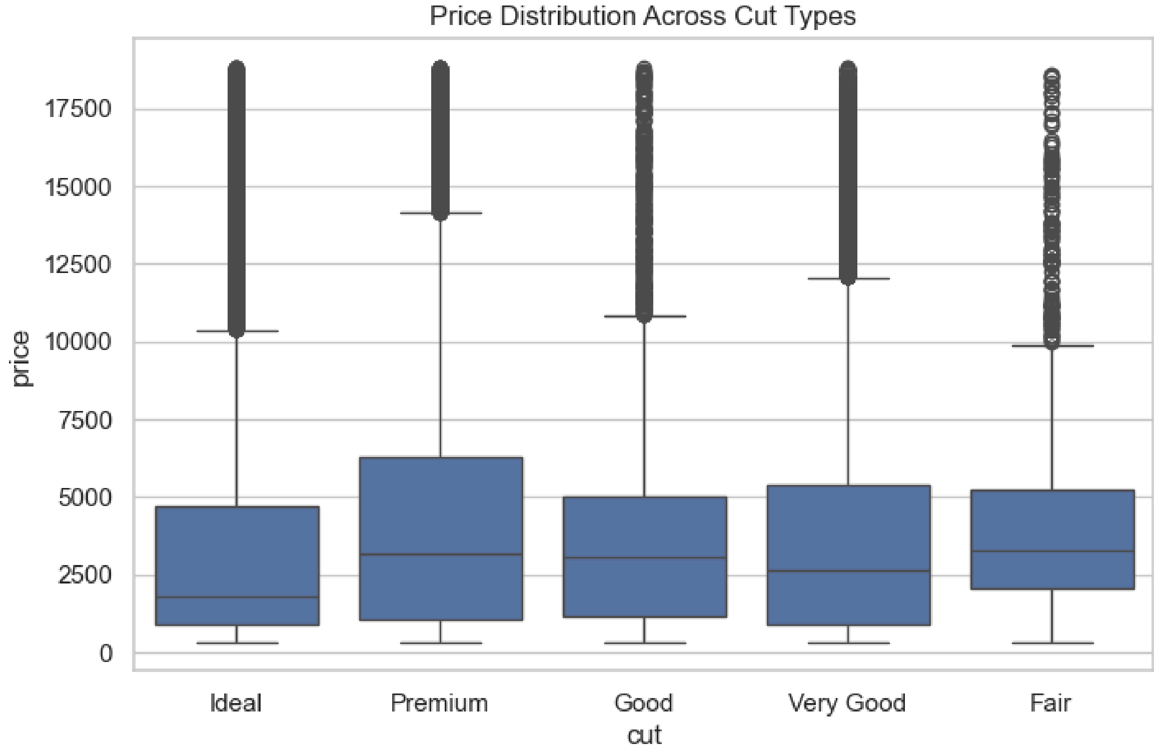

Distribution of the Diamond Clarity
count
12000
10000
8000
6000
4000
2000
0
SI1
VS2
SI2
VS1
VVS2
VVS1
IF
I1
clarity

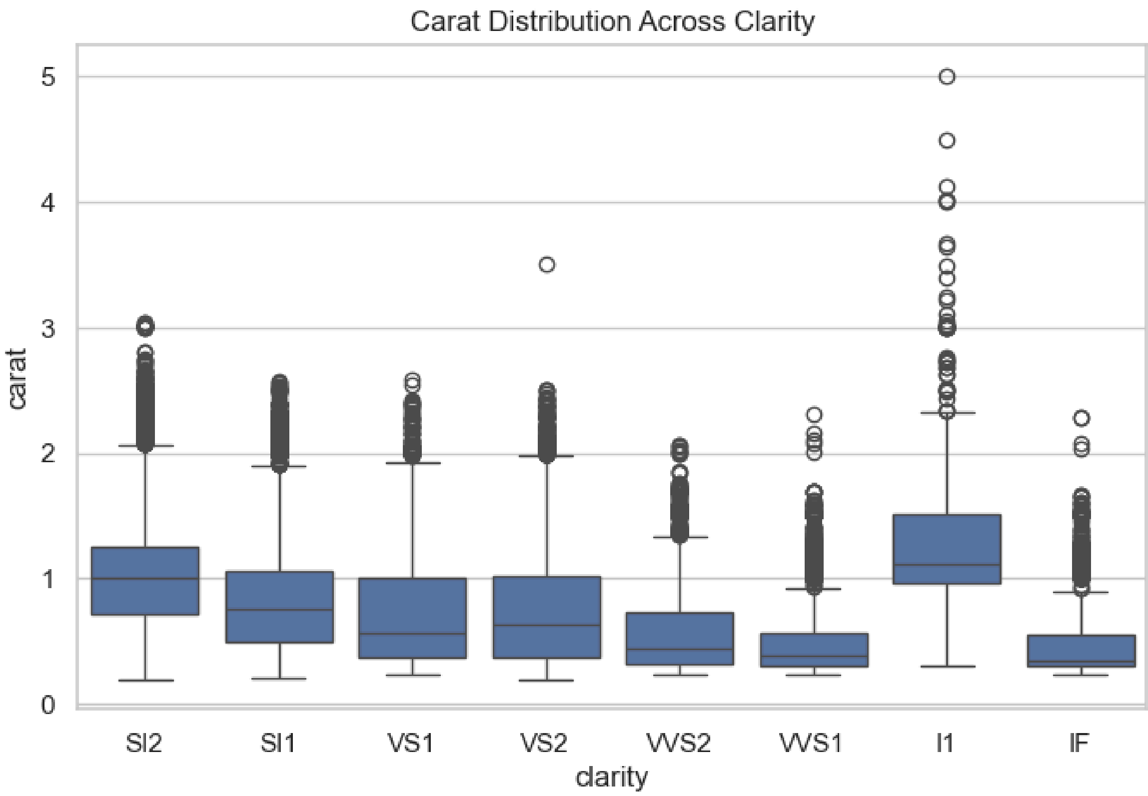
- Ideal is the highest diamond cut by a margin
- G leads in color with E following
- SI1 has got the highest count in clarity of diamond with VS2 following close

In [14]:
```
df.hist(bins=30, figsize=(14, 10), color='skyblue', edgecolor='black')
plt.suptitle('Histogram Distribution of Numeric Features', fontsize=16)
plt.tight_layout()
plt.show()
```



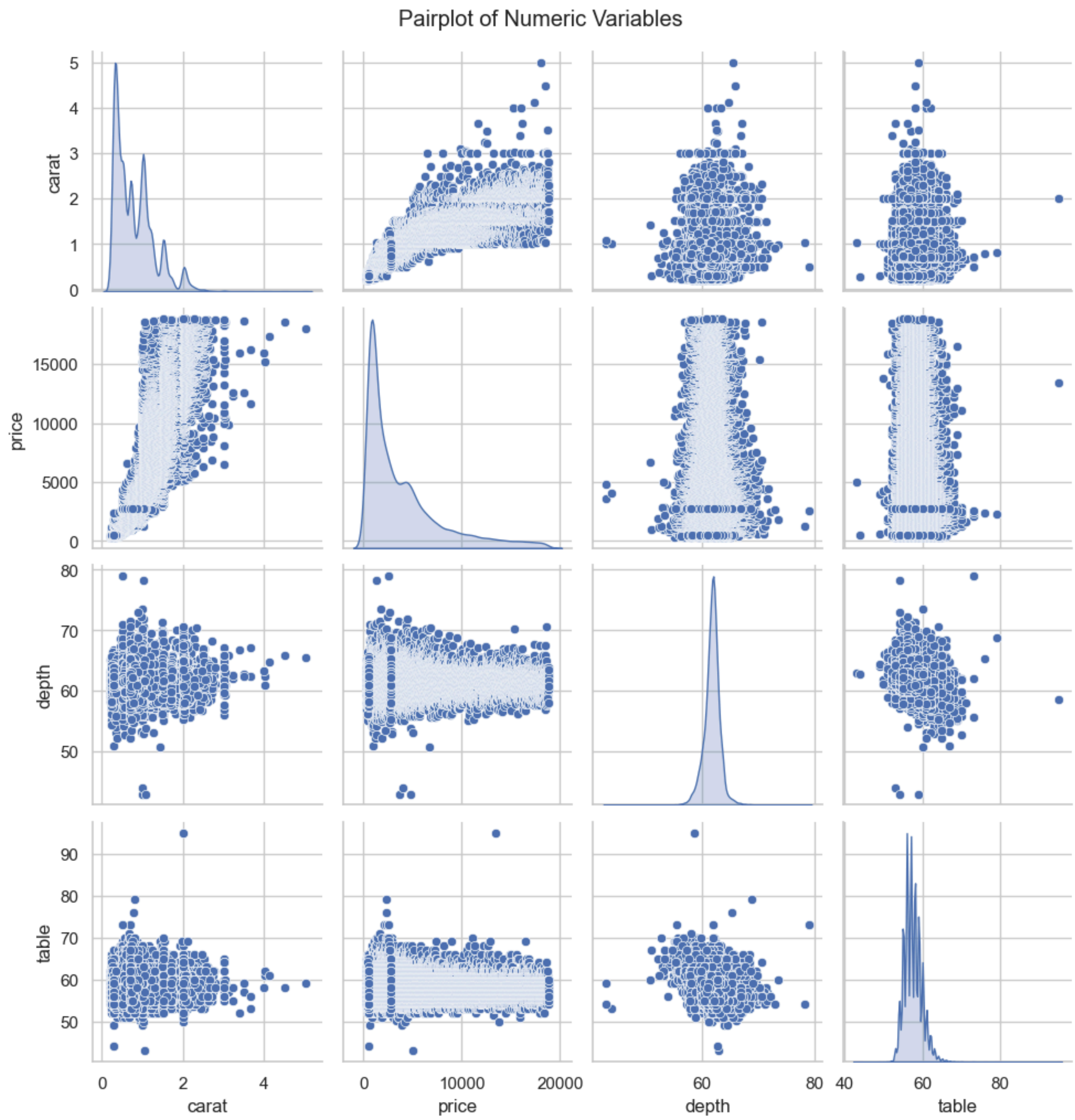
In [15]:
```
plt.figure(figsize=(8, 5))
sns.boxplot(data=df, x='cut', y='price')
plt.title('Price Distribution Across Cut Types')
plt.show()

plt.figure(figsize=(8, 5))
sns.boxplot(data=df, x='clarity', y='carat')
plt.title('Carat Distribution Across Clarity')
plt.show()
```





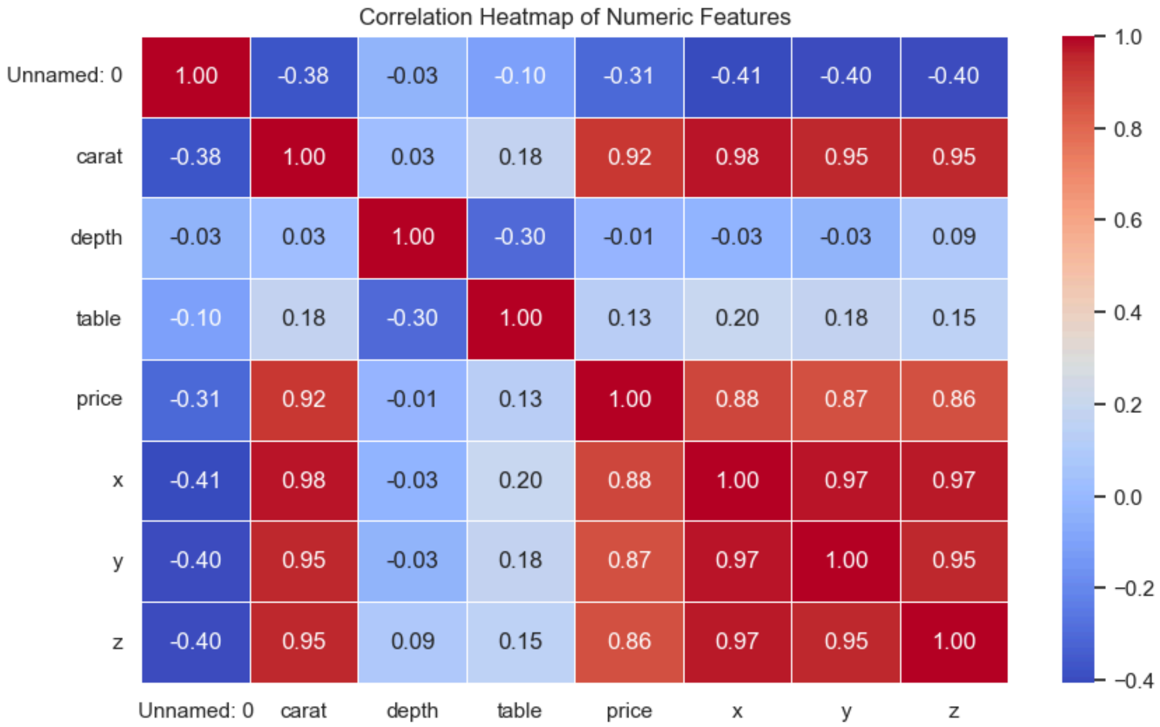
In [16]:

```
# Pairplot for key features
sns.pairplot(df[['carat', 'price', 'depth', 'table']], diag_kind='kde')
plt.suptitle('Pairplot of Numeric Variables', y=1.02)
plt.show()
```



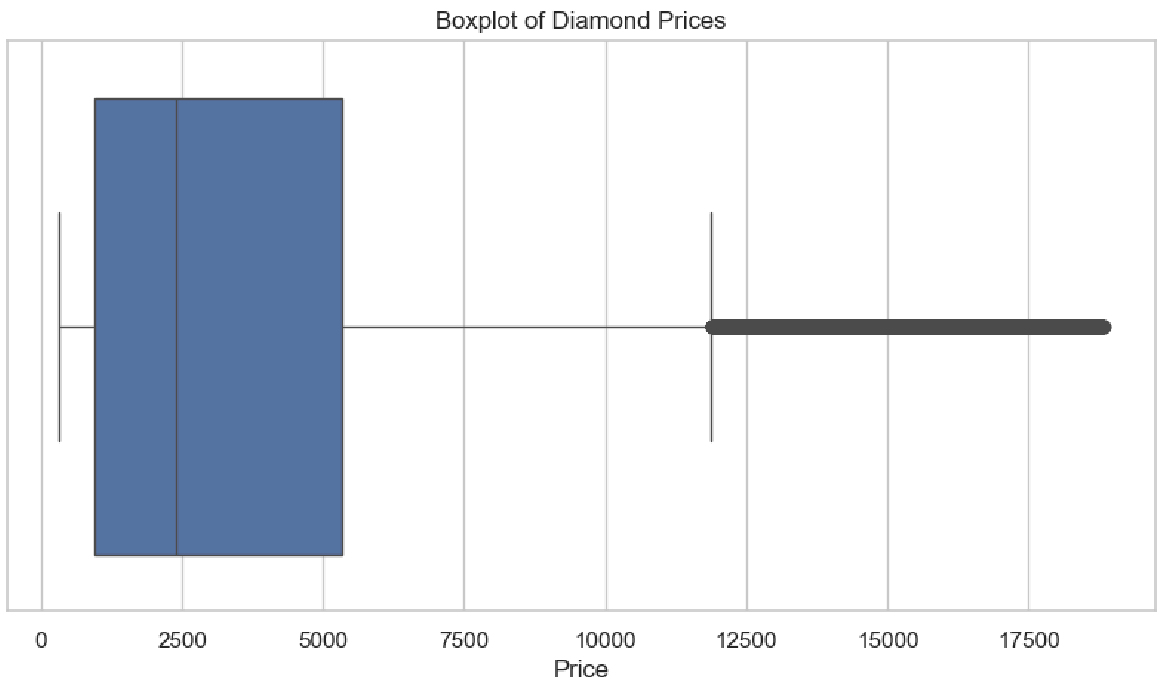
- Carat and price are right skewed and have a positive relation
- Both price and carat are seen to have ouliers
- Most of the distributions even in pairs are right skewed

In [17]:
```
corr_matrix = df.corr(numeric_only=True)

plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of Numeric Features')
plt.show()
```



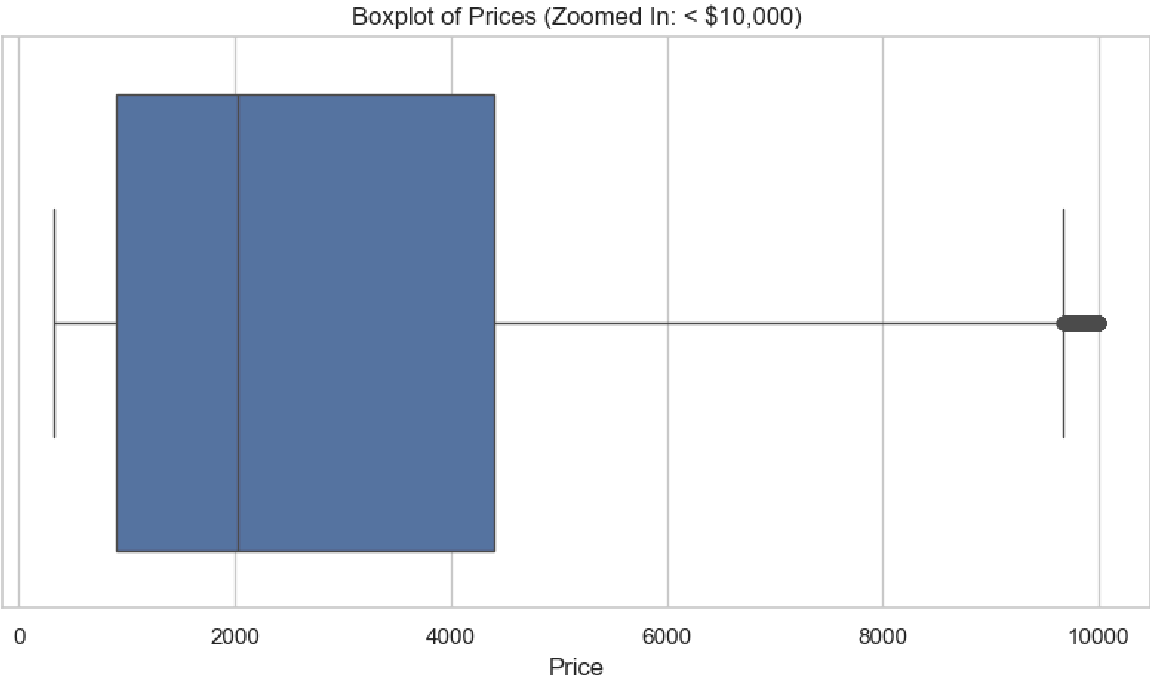
- Values +1 or -1 denotes strong relationship
- Thus can be inferred carat and price have strong coreation, so does carart and x,y,z
- Other factors like depth and table are weakly corealted with price as well as carat

In [18]:
```
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['price'])
plt.title('Boxplot of Diamond Prices')
plt.xlabel('Price')
plt.show()
```



In [19]:
```
plt.figure(figsize=(10, 5))
sns.boxplot(x=df[df['price'] < 10000]['price'])
plt.title('Boxplot of Prices (Zoomed In: < $10,000)')
```

```
plt.xlabel('Price')
plt.show()
```



In [20]:
```
plt.figure(figsize=(8, 5))
sns.scatterplot(data=df, x='carat', y='price', hue='cut', alpha=0.6)
plt.title('Carat vs Price Colored by Cut')
plt.show()
```

In [21]:
```
plt.figure(figsize=(8, 5))
sns.lineplot(data=df.groupby('cut')['price'].mean().reset_index(), x='cut', y='p
plt.title('Average Price by Cut Type')
plt.show()
```

- Box plot shows very high amounts of outliers
- But outliers tend to reduce when price comes below $10000
- Ideal Cut and Fair Cut seems to have the better pricing, with some fluctuations by the end
- Average price is increasing non linearly with reducing from fair to good to ideal and then going back up to premium to come back again at very good
- Ideal Cut seems to have the lowest price even with great quaity while Premium Cut has the highest price as expected

Final Data Summary

The exploratory data analysis on diamond dataset revealed the following:

- **Carat** is the price maker of the diamonds
- *Cut*,*Color* and *calrity* plays it's role but not as much as carat
- Outliers does exist especially when price goes into premium ranges (above $10k)
- Ideal cut are the most commom type of diamond given the better price at a great quality thus showing great market preference
- For further quality analysis dataplots (x,y,z) can be validated