In [11]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
import matplotlib as plt
%matplotlib inline
```

In [12]:
```
url = "https://raw.githubusercontent.com/edyoda/data-science-complete-tutorial/master/Data/house_rental_data.csv.txt"
df1 = pd.read_csv(url)
df1.head()
```

Out[12]:

| | Unnamed: 0 | Sqft | Floor | TotalFloor | Bedroom | Living.Room | Bathroom | Price |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 1177.698 | 2 | 7 | 2 | 2 | 2 | 62000 |
| **1** | 2 | 2134.800 | 5 | 7 | 4 | 2 | 2 | 78000 |
| **2** | 3 | 1138.560 | 5 | 7 | 2 | 2 | 1 | 58000 |
| **3** | 4 | 1458.780 | 2 | 7 | 3 | 2 | 2 | 45000 |
| **4** | 5 | 967.776 | 11 | 14 | 3 | 2 | 2 | 45000 |

In [13]:
```
df1=df1.drop(["Unnamed: 0"], axis=1)
df1.head()
```

Out[13]:

| | Sqft | Floor | TotalFloor | Bedroom | Living.Room | Bathroom | Price |
|---|---|---|---|---|---|---|---|
| **0** | 1177.698 | 2 | 7 | 2 | 2 | 2 | 62000 |
| **1** | 2134.800 | 5 | 7 | 4 | 2 | 2 | 78000 |
| **2** | 1138.560 | 5 | 7 | 2 | 2 | 1 | 58000 |
| **3** | 1458.780 | 2 | 7 | 3 | 2 | 2 | 45000 |
| **4** | 967.776 | 11 | 14 | 3 | 2 | 2 | 45000 |

In [14]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 645 entries, 0 to 644
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Sqft         645 non-null    float64
 1   Floor        645 non-null    int64
 2   TotalFloor   645 non-null    int64
 3   Bedroom      645 non-null    int64
 4   Living.Room  645 non-null    int64
 5   Bathroom     645 non-null    int64
 6   Price        645 non-null    int64
dtypes: float64(1), int64(6)
memory usage: 35.4 KB
```

In [15]:
```
df1=df1.rename(columns = {"Living.Room" : "Living Room","TotalFloor":"Total Floor"})
df1.head()
```

Out[15]:

| | Sqft | Floor | Total Floor | Bedroom | Living Room | Bathroom | Price |
|---|---|---|---|---|---|---|---|
| **0** | 1177.698 | 2 | 7 | 2 | 2 | 2 | 62000 |
| **1** | 2134.800 | 5 | 7 | 4 | 2 | 2 | 78000 |
| **2** | 1138.560 | 5 | 7 | 2 | 2 | 1 | 58000 |
| **3** | 1458.780 | 2 | 7 | 3 | 2 | 2 | 45000 |
| **4** | 967.776 | 11 | 14 | 3 | 2 | 2 | 45000 |

In [16]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler, normalize
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
from scipy.cluster import hierarchy
from scipy.spatial.distance import cdist
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
from sklearn import metrics
from sklearn.metrics import silhouette_score
import scipy.cluster.hierarchy as shc
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df1)
pd.DataFrame(scaled_data).describe()
```

Out[16]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| **count** | 6.450000e+02 | 6.450000e+02 | 6.450000e+02 | 6.450000e+02 | 6.450000e+02 | 6.450000e+02 | 6.450000e+02 |
| **mean** | 7.289604e-17 | 1.056971e-17 | -6.024466e-18 | -1.684269e-16 | 1.984631e-16 | 9.191614e-17 | -8.451465e-17 |
| **std** | 1.000776e+00 | 1.000776e+00 | 1.000776e+00 | 1.000776e+00 | 1.000776e+00 | 1.000776e+00 | 1.000776e+00 |
| **min** | -1.523619e+00 | -1.272516e+00 | -1.974190e+00 | -1.819099e+00 | -3.926263e+00 | -2.651152e+00 | -1.569526e+00 |
| **25%** | -7.858412e-01 | -7.572786e-01 | -7.723470e-01 | -8.289563e-01 | 4.026936e-01 | -1.187117e+00 | -6.455621e-01 |
| **50%** | -1.362251e-01 | -2.420416e-01 | 2.291886e-01 | 1.611860e-01 | 4.026936e-01 | 2.769182e-01 | -3.366380e-01 |
| **75%** | 4.762700e-01 | 5.308140e-01 | 6.298029e-01 | 1.151328e+00 | 4.026936e-01 | 2.769182e-01 | 3.654621e-01 |
| **max** | 5.645358e+00 | 4.137473e+00 | 5.437174e+00 | 4.121755e+00 | 4.731650e+00 | 4.669023e+00 | 5.280163e+00 |

In [17]:
```
kmeans = KMeans(n_clusters = 4, init = 'k-means++')
print(kmeans.fit(scaled_data))
print(kmeans.inertia_)
```

```
KMeans(n_clusters=4)
2155.0675715411244
```

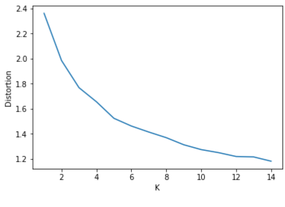
In [29]:
```
dist = []
inertias = []
map1 = {}
map2 = {}
K = range(1, 15)
for k in K:
    kmeanModel = KMeans(n_clusters=k).fit(scaled_data)
    kmeanModel.fit(scaled_data)

    dist.append(sum(np.min(cdist(scaled_data, kmeanModel.cluster_centers_,
                                 'euclidean'), axis=1)) / scaled_data.shape[0])
    inertias.append(kmeanModel.inertia_)

    map1[k] = sum(np.min(cdist(scaled_data, kmeanModel.cluster_centers_,
                               'euclidean'), axis=1)) / scaled_data.shape[0]
    map2[k] = kmeanModel.inertia_
```

```
C:\Users\subra\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\subra\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
```

In [35]:
```
for key,val in map1.items():
    print(str(key)+' : '+str(val))
ax=sns.lineplot(x =K, y =dist)
ax.set(xlabel="K", ylabel="Distortion")
```

```
1 : 2.3615265828331347
2 : 1.9847844864092707
3 : 1.7681728418731681
4 : 1.6555515931770937
5 : 1.523696258586769
6 : 1.4620086903154532
7 : 1.4139752595274477
8 : 1.36904965155053
9 : 1.3126348957549092
10 : 1.273909682828251
11 : 1.2495035817133983
12 : 1.2188928398342442
13 : 1.2156303972185416
14 : 1.181444847067608
```

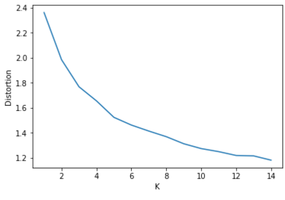
Out[35]:
```
[Text(0.5, 0, 'K'), Text(0, 0.5, 'Distortion')]
```




In [37]:
```
for key,val in map2.items():
    print(str(key)+' : '+str(val))
ax=sns.lineplot(x =K, y =dist)
ax.set(xlabel="K", ylabel="Distortion")
```

```
1 : 4515.0
2 : 3205.723708410711
3 : 2502.4632671880568
4 : 2154.163657347778
5 : 1880.8797891137028
6 : 1725.5905070198555
7 : 1595.096819289331
8 : 1519.5999644651401
9 : 1413.4392183979242
10 : 1338.766025679341
11 : 1287.2874121243965
12 : 1221.7915494412243
13 : 1182.0797078264034
14 : 1149.3742784698836
```

Out[37]:
```
[Text(0.5, 0, 'K'), Text(0, 0.5, 'Distortion')]
```




In [ ]: