Data Exploration

In [1]:
```
import pandas as pd

data = pd.read_csv('laptopPrice.csv')
```

In [2]:
```
print(data.head())
```

```
    brand processor_brand processor_name processor_gnrtn ram_gb ram_type  \
0    ASUS           Intel        Core i3            10th   4 GB     DDR4
1  Lenovo           Intel        Core i3            10th   4 GB     DDR4
2  Lenovo           Intel        Core i3            10th   4 GB     DDR4
3    ASUS           Intel        Core i5            10th   8 GB     DDR4
4    ASUS           Intel   Celeron Dual   Not Available   4 GB     DDR4

      ssd      hdd       os  os_bit graphic_card_gb  weight     warranty  \
0    0 GB  1024 GB  Windows  64-bit            0 GB  Casual  No warranty
1    0 GB  1024 GB  Windows  64-bit            0 GB  Casual  No warranty
2    0 GB  1024 GB  Windows  64-bit            0 GB  Casual  No warranty
3  512 GB     0 GB  Windows  32-bit            2 GB  Casual  No warranty
4    0 GB   512 GB  Windows  64-bit            0 GB  Casual  No warranty

  Touchscreen msoffice  Price   rating  Number of Ratings  Number of Reviews
0          No       No  34649  2 stars                  3                  0
1          No       No  38999  3 stars                 65                  5
2          No       No  39999  3 stars                  8                  1
3          No       No  69990  3 stars                  0                  0
4          No       No  26990  3 stars                  0                  0
```

```
 18  Number of Reviews  823 non-null    int64
dtypes: int64(3), object(16)
memory usage: 122.3+ KB
None
```

In [5]:
```
print(data.isnull().sum())
```

```
brand               0
processor_brand     0
processor_name      0
processor_gnrtn     0
ram_gb              0
ram_type            0
ssd                 0
hdd                 0
os                  0
os_bit              0
graphic_card_gb     0
weight              0
warranty            0
Touchscreen         0
msoffice            0
Price               0
rating              0
Number of Ratings   0
Number of Reviews   0
dtype: int64
```

In [3]:
```
print(data.describe())
```

```
              Price  Number of Ratings  Number of Reviews
count     823.000000         823.000000         823.000000
mean    76745.177400         315.301337          37.609964
std     45101.790525        1047.382654         121.728017
min     16990.000000           0.000000           0.000000
25%     46095.000000           0.000000           0.000000
50%     64990.000000          17.000000           2.000000
75%     89636.000000         139.500000          18.000000
max    441990.000000       15279.000000        1947.000000
```

In [4]:
```
print(data.info())
```

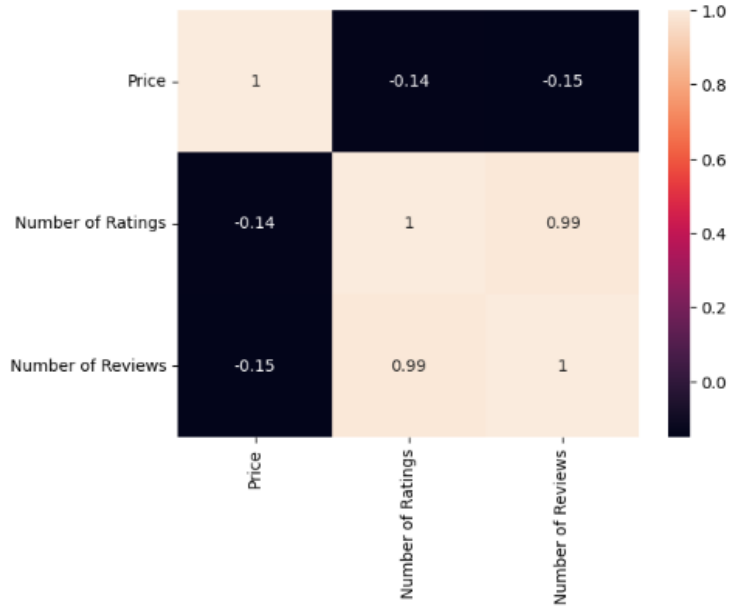
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 823 entries, 0 to 822
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   brand              823 non-null    object
 1   processor_brand    823 non-null    object
 2   processor_name     823 non-null    object
 3   processor_gnrtn    823 non-null    object
 4   ram_gb             823 non-null    object
 5   ram_type           823 non-null    object
 6   ssd                823 non-null    object
 7   hdd                823 non-null    object
 8   os                 823 non-null    object
 9   os_bit             823 non-null    object
 10  graphic_card_gb    823 non-null    object
 11  weight             823 non-null    object
 12  warranty           823 non-null    object
 13  Touchscreen        823 non-null    object
 14  msoffice           823 non-null    object
 15  Price              823 non-null    int64 
 16  rating             823 non-null    object
 17  Number of Ratings  823 non-null    int64 
 18  Number of Reviews  823 non-null    int64 
```

Correlation Analysis

In [6]:
```
import seaborn as sns
import matplotlib.pyplot as plt

correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True)
plt.show()
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_12424\2630443792.py:4: FutureWarning: The default value of numeric_only in DataFrame.c
orr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_onl
y to silence this warning.
  correlation_matrix = data.corr()
```



Feature Engineering

In [7]:
```
print(data.columns)
```

```
Index(['brand', 'processor_brand', 'processor_name', 'processor_gnrtn',
       'ram_gb', 'ram_type', 'ssd', 'hdd', 'os', 'os_bit', 'graphic_card_gb',
       'weight', 'warranty', 'Touchscreen', 'msoffice', 'Price', 'rating',
       'Number of Ratings', 'Number of Reviews'],
      dtype='object')
```

In [8]:
```
data = pd.get_dummies(data)
```

Model Selection and Training

In [9]:
```
from sklearn.model_selection import train_test_split

X = data.drop('Price', axis=1)  # 'Price' instead of 'price' based on your output
y = data['Price']  # 'Price' instead of 'price' based on your output

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [10]:
```
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
```

In [11]:
```
# Train the models
rf_model.fit(X_train, y_train)
gb_model.fit(X_train, y_train)
```

Out[11]:
```
GradientBoostingRegressor
GradientBoostingRegressor(random_state=42)
```

In [12]:
```
rf_y_pred = rf_model.predict(X_test)
```

In [13]:
```
gb_y_pred = gb_model.predict(X_test)
```

Model Evaluation

In [14]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
# Calculate evaluation metrics for Random Forest
rf_mae = mean_absolute_error(y_test, rf_y_pred)
rf_rmse = np.sqrt(mean_squared_error(y_test, rf_y_pred))
rf_r2 = r2_score(y_test, rf_y_pred)

print('Random Forest Regressor Metrics:')
print(f'Mean Absolute Error (MAE): {rf_mae:.2f}')
print(f'Root Mean Squared Error (RMSE): {rf_rmse:.2f}')
print(f'R-squared (R2 Score): {rf_r2:.2f}')
```

```
Random Forest Regressor Metrics:
Mean Absolute Error (MAE): 13262.89
Root Mean Squared Error (RMSE): 26259.54
R-squared (R2 Score): 0.65
```

In [15]:
```
# Calculate evaluation metrics for Gradient Boosting
gb_mae = mean_absolute_error(y_test, gb_y_pred)
gb_rmse = np.sqrt(mean_squared_error(y_test, gb_y_pred))
gb_r2 = r2_score(y_test, gb_y_pred)

print('\nGradient Boosting Regressor Metrics:')
print(f'Mean Absolute Error (MAE): {gb_mae:.2f}')
print(f'Root Mean Squared Error (RMSE): {gb_rmse:.2f}')
print(f'R-squared (R2 Score): {gb_r2:.2f}')
```

```
Gradient Boosting Regressor Metrics:
Mean Absolute Error (MAE): 13407.56
Root Mean Squared Error (RMSE): 25325.84
R-squared (R2 Score): 0.67
```

Cross-validation

In [16]:
```
from sklearn.model_selection import cross_val_score

# Perform cross-validation for Random Forest Regressor
rf_cv_scores = cross_val_score(rf_model, X, y, cv=5, scoring='neg_mean_squared_error')
rf_cv_rmse_scores = np.sqrt(-rf_cv_scores)
rf_mean_rmse = rf_cv_rmse_scores.mean()
rf_std_rmse = rf_cv_rmse_scores.std()

print('Random Forest Regressor Cross-validation RMSE:')
print(f'Mean RMSE: {rf_mean_rmse:.2f}')
print(f'Standard Deviation RMSE: {rf_std_rmse:.2f}')
```

```
# Perform cross-validation for Gradient Boosting Regressor
gb_cv_scores = cross_val_score(gb_model, X, y, cv=5, scoring='neg_mean_squared_error')
gb_cv_rmse_scores = np.sqrt(-gb_cv_scores)
gb_mean_rmse = gb_cv_rmse_scores.mean()
gb_std_rmse = gb_cv_rmse_scores.std()

print('\nGradient Boosting Regressor Cross-validation RMSE:')
print(f'Mean RMSE: {gb_mean_rmse:.2f}')
print(f'Standard Deviation RMSE: {gb_std_rmse:.2f}')
```

```
Random Forest Regressor Cross-validation RMSE:
Mean RMSE: 28462.62
Standard Deviation RMSE: 16814.39

Gradient Boosting Regressor Cross-validation RMSE:
Mean RMSE: 28667.33
Standard Deviation RMSE: 15391.00
```

In [17]:
```
import joblib
joblib.dump(rf_model, 'random_forest_laptop_price_prediction_model.pkl')
joblib.dump(gb_model, 'gradient_boosting_laptop_price_prediction_model.pkl')
```

Out[17]:
```
['gradient_boosting_laptop_price_prediction_model.pkl']
```

In [18]:
```
import matplotlib.pyplot as plt

# Example RMSE scores
models = ['Random Forest', 'Gradient Boosting']
rmse_scores = [26259.54, 25325.84]

# Plotting
plt.figure(figsize=(10, 6))
bars = plt.bar(models, rmse_scores, color=['green', 'orange'])
```

```
# Adding labels and titles
plt.title('Model Evaluation: RMSE Comparison')
plt.xlabel('Models')
plt.ylabel('RMSE')
# Removing logarithmic scale for better visibility
plt.grid(axis='y', linestyle='--', alpha=0.7)
# Adding the RMSE values on top of each bar with adjusted positions for readability
for bar, score in zip(bars, rmse_scores):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.005 * max(rmse_scores), f'{score:.2f}', ha='center', va='bo
plt.ylim(0, max(rmse_scores) * 1.2)  # Set y-axis limit slightly above maximum RMSE for better visualization
plt.tight_layout()
plt.show()
```