Data Exploration

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

data = pd.read_csv('CAR DETAILS FROM CAR DEKHO.csv')
```

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

```
                      name  year  selling_price  km_driven    fuel  \
0            Maruti 800 AC  2007          60000      70000  Petrol
1  Maruti Wagon R LXI Minor  2007         135000      50000  Petrol
2      Hyundai Verna 1.6 SX  2012         600000     100000  Diesel
3   Datsun RediGO T Option   2017         250000      46000  Petrol
4    Honda Amaze VX i-DTEC  2014         450000     141000  Diesel

  seller_type transmission         owner
0  Individual       Manual   First Owner
1  Individual       Manual   First Owner
2  Individual       Manual   First Owner
3  Individual       Manual   First Owner
4  Individual       Manual  Second Owner
```

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

```
              year  selling_price      km_driven
count  4340.000000   4.340000e+03    4340.000000
mean   2013.090783   5.041273e+05   66215.777419
std       4.215344   5.785487e+05   46644.102194
min    1992.000000   2.000000e+04       1.000000
25%    2011.000000   2.087498e+05   35000.000000
50%    2014.000000   3.500000e+05   60000.000000
75%    2016.000000   6.000000e+05   90000.000000
max    2020.000000   8.900000e+06  806599.000000
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           4340 non-null   object
 1   year           4340 non-null   int64
 2   selling_price  4340 non-null   int64
 3   km_driven      4340 non-null   int64
 4   fuel           4340 non-null   object
 5   seller_type    4340 non-null   object
 6   transmission   4340 non-null   object
 7   owner          4340 non-null   object
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
None
```

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

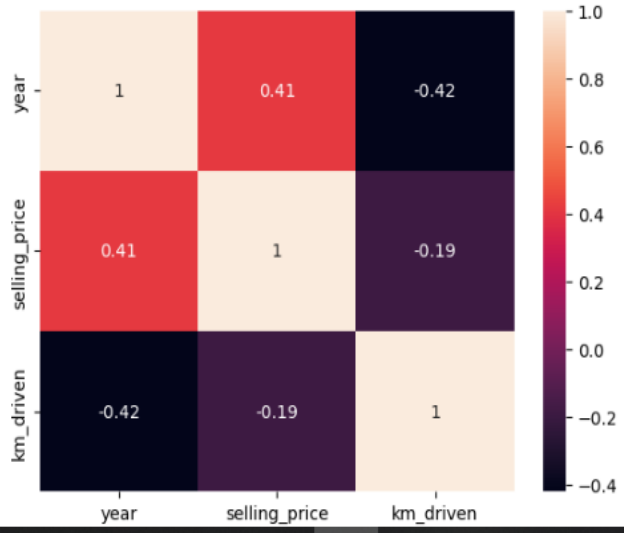
```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
dtype: int64
```

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_13800\2630443792.py:4: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = data.corr()
```



Feature Engineering

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

```
Index(['name', 'year', 'selling_price', 'km_driven', 'fuel', 'seller_type',
       'transmission', 'owner'],
      dtype='object')
```

In [8]:
```
categorical_columns = ['fuel', 'seller_type', 'transmission', 'owner','name']
data = pd.get_dummies(data, columns=categorical_columns)
```

In [9]:
```
data['age'] = 2024 - data['year']  # Example: Car age
```

In [10]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
numerical_columns = ['year', 'km_driven', 'age']  # Add other numerical columns if needed
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])
```

Model Selection and Training

In [11]:
```
from sklearn.model_selection import train_test_split

X = data.drop('selling_price', axis=1)  # Assuming 'selling_price' is the target
y = data['selling_price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [12]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(),
    'Gradient Boosting': GradientBoostingRegressor()
}
```

In [13]:
```
for name, model in models.items():
    model.fit(X_train, y_train)
    print(f'{name} model trained.')
```

```
Linear Regression model trained.
Random Forest model trained.
Gradient Boosting model trained.
```

Model Evaluation

In [14]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error

for name, model in models.items():
    predictions = model.predict(X_test)
    mae = mean_absolute_error(y_test, predictions)
    rmse = mean_squared_error(y_test, predictions, squared=False)
    print(f'{name} - MAE: {mae}, RMSE: {rmse}')
```

```
Linear Regression - MAE: 1206447987844883.2, RMSE: 4366519368862874.5
Random Forest - MAE: 119250.13070048825, RMSE: 361539.81072706083
Gradient Boosting - MAE: 168977.14176387936, RMSE: 375091.1426380031
```

Cross-validation

In [15]:
```
from sklearn.model_selection import cross_val_score

for name, model in models.items():
    scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')
    rmse_scores = (-scores) ** 0.5
    print(f'{name} - Cross-Validated RMSE: {rmse_scores.mean()}')
```

```
Linear Regression - Cross-Validated RMSE: 4.0336273887273544e+16
Random Forest - Cross-Validated RMSE: 284039.632066171
Gradient Boosting - Cross-Validated RMSE: 314886.7747224304
```

In [19]:
```
import matplotlib.pyplot as plt
import numpy as np

# Example RMSE scores
models = ['Linear Regression', 'Random Forest', 'Gradient Boosting']
rmse_scores = [4366519368862874.5, 361539.81072706083, 375091.1426380031]
```

```
# Plotting
plt.figure(figsize=(10, 6))
bars = plt.bar(models, rmse_scores, color=['blue', 'green', 'orange'])
# Adding labels and titles
plt.title('Model Evaluation: RMSE Comparison')
plt.xlabel('Models')
plt.ylabel('RMSE')
plt.yscale('log')  # Set y-axis to logarithmic scale
plt.grid(axis='y', linestyle='--', alpha=0.7)
# Adding the RMSE values on top of each bar
for bar, score in zip(bars, rmse_scores):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.1, f'{score:.2e}', ha='center', va='bottom', fontsize=10)

plt.tight_layout()
plt.show()
```