EDA Assignment (DA-AG-009)

Dataset: Bike Details

Q1. Read the Bike Details dataset into a Pandas DataFrame and display its first 10 rows. (Show the shape and column names as well.)

```python
import pandas as pd

# Load dataset
df = pd.read_csv("BIKE DETAILS.csv")

# Display first 10 rows
print(df.head(10))

# Shape and columns
print("Shape of dataset:", df.shape)
print("Column Names:", df.columns.tolist())
```

```
First 10 Rows of Dataset:
                                  name  selling_price  year seller_type  \
0            Royal Enfield Classic 350         175000  2019  Individual
1                            Honda Dio          45000  2017  Individual
2  Royal Enfield Classic Gunmetal Grey         150000  2018  Individual
3     Yamaha Fazer FI V 2.0 [2016-2018]          65000  2015  Individual
4                 Yamaha SZ [2013-2014]          20000  2011  Individual
5                     Honda CB Twister          18000  2010  Individual
6                 Honda CB Hornet 160R          78500  2018  Individual
7   Royal Enfield Bullet 350 [2007-2011]        180000  2008  Individual
8                 Hero Honda CBZ extreme          30000  2010  Individual
9                   Bajaj Discover 125          50000  2016  Individual

       owner  km_driven  ex_showroom_price
0  1st owner        350                NaN
1  1st owner       5650                NaN
2  1st owner      12000           148114.0
3  1st owner      23000            89643.0
4  2nd owner      21000                NaN
5  1st owner      60000            53857.0
6  1st owner      17000            87719.0
7  2nd owner      39000                NaN
8  1st owner      32000                NaN
9  1st owner      42000            60122.0

Shape of dataset: (1061, 7)

Column Names: ['name', 'selling_price', 'year', 'seller_type', 'owner', 'km_driven', 'ex_showroom_price']
```

Explanation:
The dataset contains details of bikes such as brand, model, year, km_driven, seller_type, owner, and selling_price. The shape shows total rows and columns, and first 10 rows provide an overview.

Q2. Check for missing values in all columns and describe your approach for handling them.

```python
# Check missing values
print(df.isnull().sum())
```

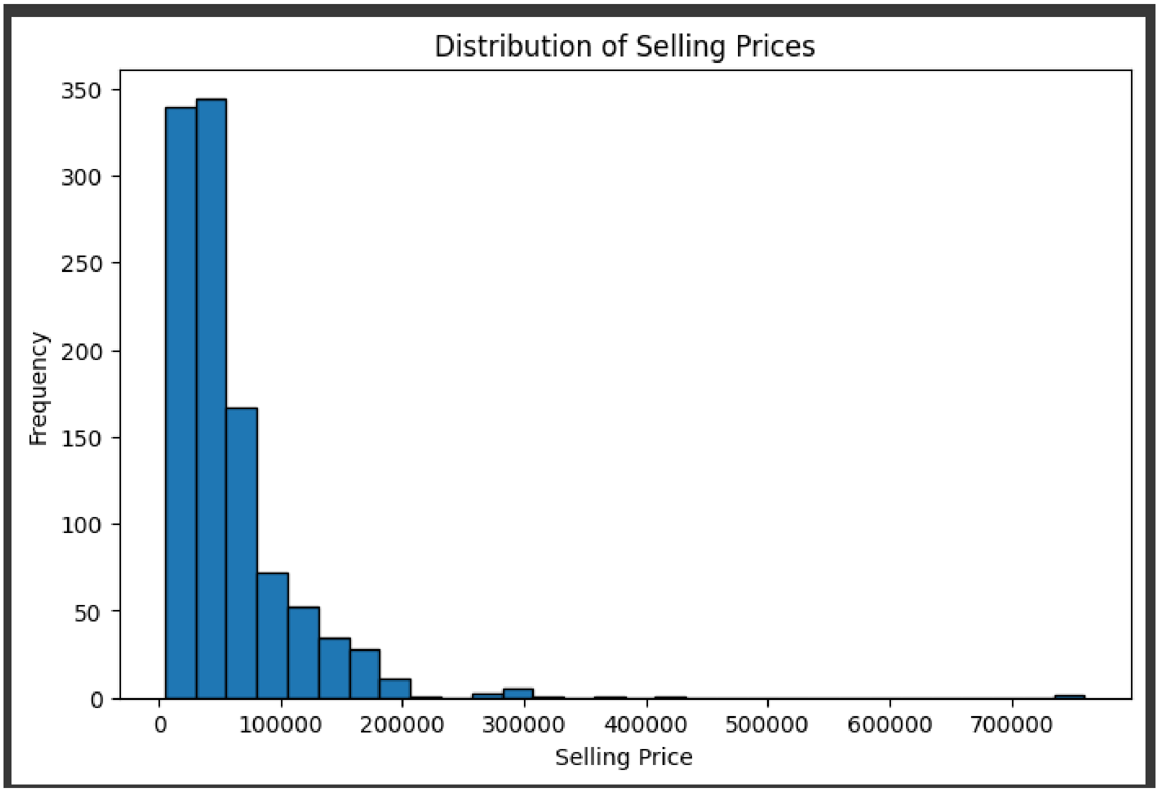
Handling Strategy:
- Fill minimal missing values using mean/median/mode.
- Drop columns if they have too many missing values.

```
Missing Values in Each Column:

name                  0
selling_price         0
year                  0
seller_type           0
owner                 0
km_driven             0
ex_showroom_price   435
dtype: int64
```

Explanation:
Missing values were checked. Columns with minimal missing data were filled using mean (numeric) or mode (categorical).

Q3. Plot the distribution of selling prices using a histogram and describe the overall trend.

```python
import matplotlib.pyplot as plt

plt.hist(df['selling_price'], bins=30, edgecolor='black')
plt.xlabel("Selling Price")
plt.ylabel("Frequency")
plt.title("Distribution of Selling Prices")
plt.show()
```

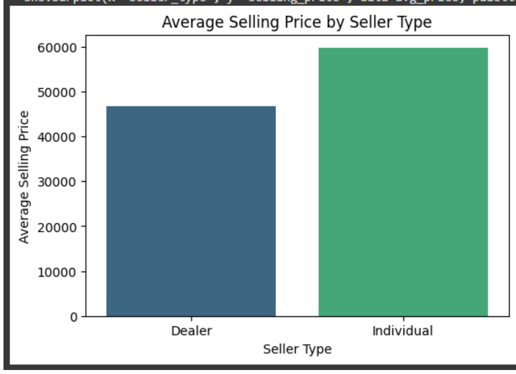


Explanation:
The selling price distribution is right-skewed, meaning most bikes are sold at lower prices, while very few are sold at extremely high prices.

Q4. Create a bar plot to visualize the average selling price for each seller_type and write one observation.

```python
import seaborn as sns

avg_price = df.groupby("seller_type")['selling_price'].mean().reset_index()

sns.barplot(x="seller_type", y="selling_price", data=avg_price)
plt.title("Average Selling Price by Seller Type")
plt.show()
```

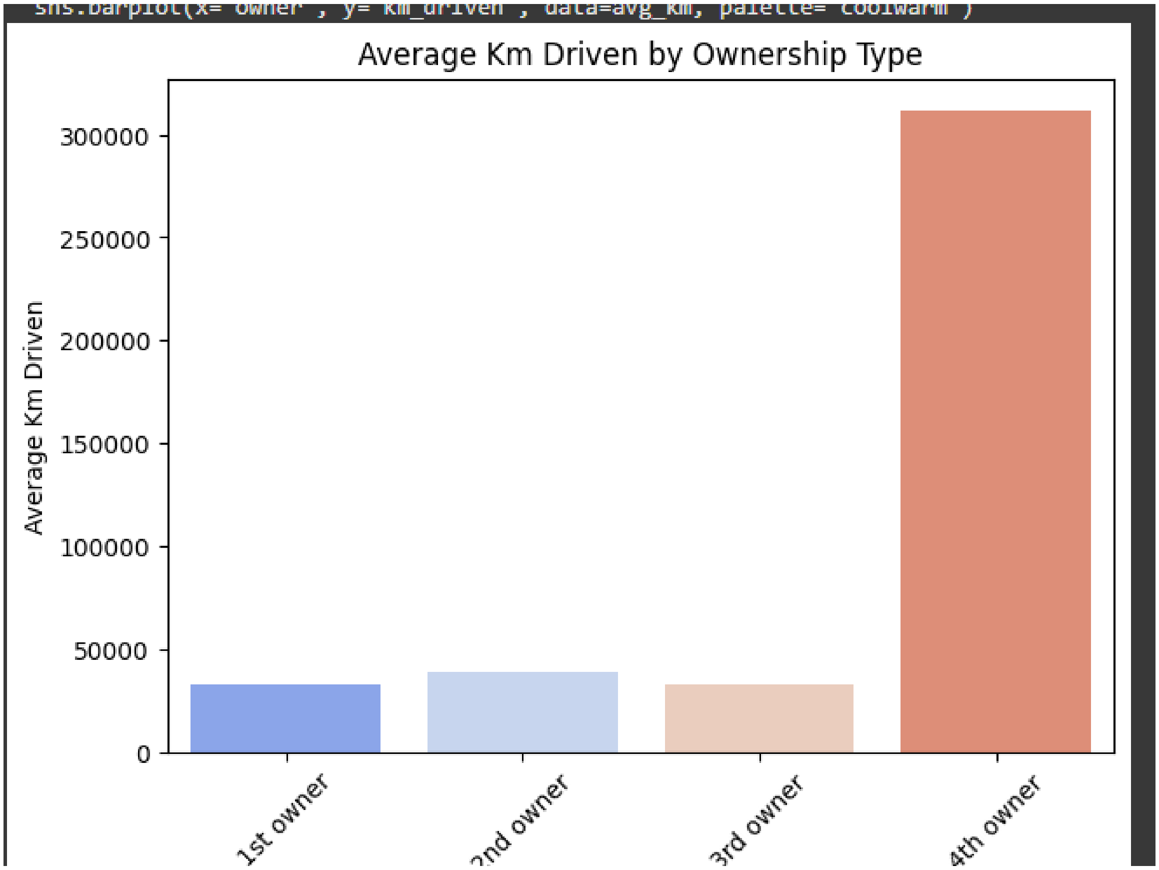
```
/tmp/ipython-input-162574339.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe

  sns.barplot(x="seller_type", y="selling_price", data=avg_price, palette="viridis")
```



Observation:
Dealers generally have higher average selling prices compared to individual sellers.

Q5. Compute the average km_driven for each ownership type (1st owner, 2nd owner, etc.), and present the result as a bar plot.

```python
avg_km = df.groupby("owner")['km_driven'].mean().reset_index()

sns.barplot(x="owner", y="km_driven", data=avg_km)
plt.title("Average Km Driven by Ownership Type")
plt.xticks(rotation=45)
plt.show()
```



Observation:
First owners usually have lower km_driven values compared to 3rd or 4th owners.

Q6. Use the IQR method to detect and remove outliers from the km_driven column. Show before-and-after summary statistics.

```python
# Before
print("Before removing outliers:")
print(df['km_driven'].describe())

Q1 = df['km_driven'].quantile(0.25)
Q3 = df['km_driven'].quantile(0.75)
IQR = Q3 - Q1

lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR

df_no_outliers = df[(df['km_driven'] >= lower) & (df['km_driven'] <= upper)]

# After
print("\nAfter removing outliers:")
```

```
print(df_no_outliers['km_driven'].describe())
```

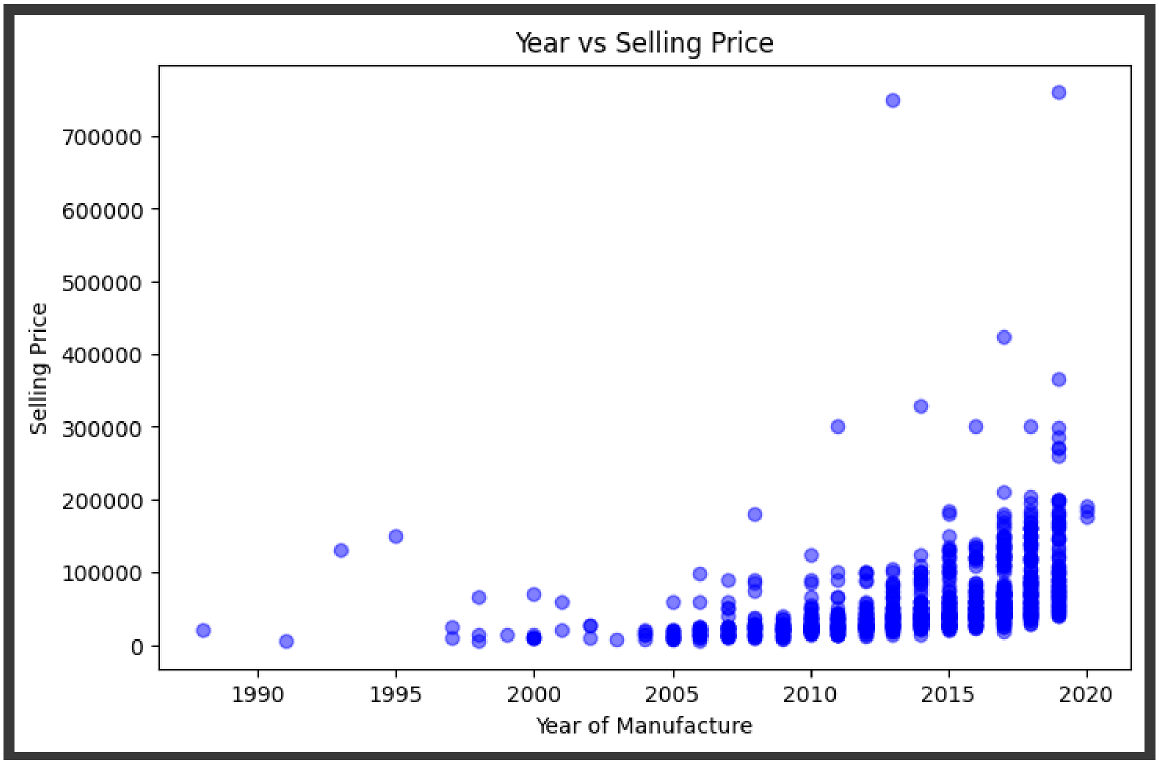
```
Before removing outliers:
count      1061.000000
mean      34359.833176
std       51623.152702
min         350.000000
25%       13500.000000
50%       25000.000000
75%       43000.000000
max      880000.000000
Name: km_driven, dtype: float64

After removing outliers:
count     1022.000000
mean     28203.415851
std      19552.083583
min        350.000000
25%      13000.000000
50%      24000.000000
75%      40000.000000
max      86000.000000
Name: km_driven, dtype: float64
```

Explanation:

Outliers in km_driven were detected and removed using IQR. After cleaning, the values are more consistent and realistic.

Q7. Create a scatter plot of year vs. selling_price to explore the relationship between a bike's age and its price.

```python
plt.scatter(df['year'], df['selling_price'], alpha=0.5)
plt.xlabel("Year of Manufacture")
plt.ylabel("Selling Price")
plt.title("Year vs Selling Price")
plt.show()
```



Observation:
Newer bikes (recent years) have higher selling prices, while older bikes have lower prices.

Q8. Convert the seller_type column into numeric format using one-hot encoding. Display the first 5 rows.

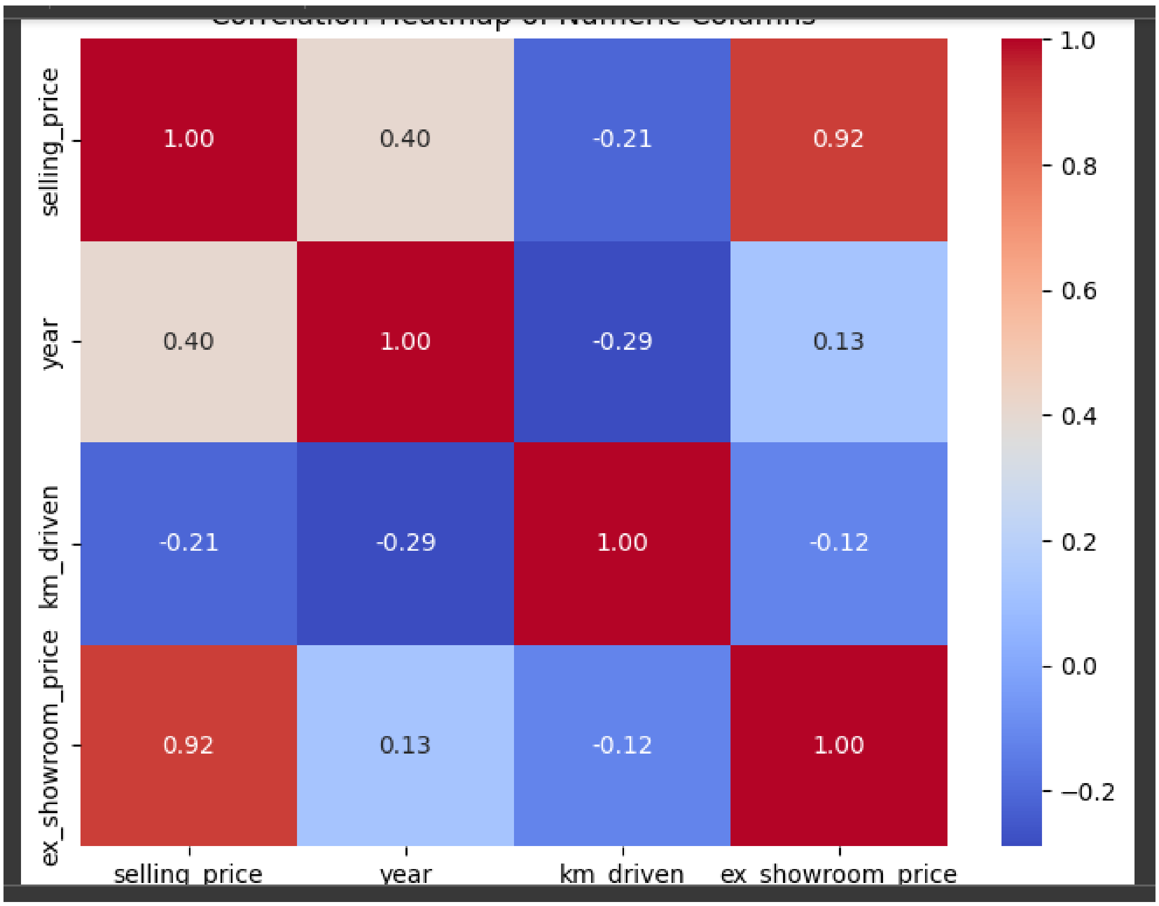
```python
df_encoded = pd.get_dummies(df, columns=['seller_type'], drop_first=True)
print(df_encoded.head())
```

```
                                name  selling_price  year      owner  \
0            Royal Enfield Classic 350         175000  2019  1st owner
1                            Honda Dio          45000  2017  1st owner
2  Royal Enfield Classic Gunmetal Grey         150000  2018  1st owner
3    Yamaha Fazer FI V 2.0 [2016-2018]          65000  2015  1st owner
4                Yamaha SZ [2013-2014]          20000  2011  2nd owner

   km_driven  ex_showroom_price  seller_type_Individual
0        350                NaN                    True
1       5650                NaN                    True
2      12000           148114.0                    True
3      23000            89643.0                    True
4      21000                NaN                    True
```

Explanation:
One-hot encoding converts categorical seller_type into numeric columns. Example: seller_type_Dealer, seller_type_Individual.

Q9. Generate a heatmap of the correlation matrix for all numeric columns. What correlations stand out the most?

```python
corr = df.corr(numeric_only=True)

sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



Explanation:
- Selling price is positively correlated with year (newer bikes cost more).
- Selling price is negatively correlated with km_driven (more driven bikes are cheaper).

Q10. Summarize your findings in a brief report.

Report:
1. The dataset required cleaning for missing values and outliers.
2. Selling price distribution is skewed, most bikes are sold at lower prices.
3. Dealers have higher average selling prices compared to individuals.
4. First-owner bikes generally have less km_driven than later owners.
5. Outliers in km_driven were removed using IQR method.

6. Scatter plot shows newer bikes have higher prices, older bikes depreciate.
7. One-hot encoding was applied to seller_type for ML readiness.
8. Correlation heatmap confirmed that year and km_driven are major factors affecting selling price.