import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

try:

df = pd.read\_csv('car\_prices.csv')

except FileNotFoundError:

print("Error: 'car\_prices.csv' not found. Please ensure the file is in the same directory.")

# Display the first 5 rows and data types

print("First 5 rows of the DataFrame:")

print(df.head())

print("\nDataFrame Info:")

df.info()

# Check and handle duplicates

print("\nNumber of duplicate rows:")

print(df.duplicated().sum())

df.drop\_duplicates(inplace=True)

print("Duplicate rows dropped.")

# Check for and handle missing values

print("\nNull values per column:")

print(df.isnull().sum())

df['transmission'].fillna('missing', inplace=True)

df.drop('saledate', axis=1, inplace=True)

print("Missing values handled.")

# 2.1 Total number of cars sold

total\_cars\_sold = len(df)

print(f"Total cars sold: {total\_cars\_sold}")

# 2.2 Top 10 most popular car makes

top\_10\_makes = df['make'].value\_counts().head(10)

print("\nTop 10 most popular car makes:")

print(top\_10\_makes)

# 2.3 Average selling price

avg\_price = df['sellingprice'].mean()

print(f"\nAverage selling price: ${avg\_price:,.2f}")

# 2.4 Number of cars by transmission type

transmission\_counts = df['transmission'].value\_counts()

print("\nCars by transmission type:")

print(transmission\_counts)

# 2.5 Average selling price for automatic cars

avg\_price\_auto = df[df['transmission'] == 'automatic']['sellingprice'].mean()

print(f"\nAverage price for automatic cars: ${avg\_price\_auto:,.2f}")

# 2.6 Make with the highest average selling price

highest\_avg\_price\_make = df.groupby('make')['sellingprice'].mean().idxmax()

print(f"\nMake with highest average price: {highest\_avg\_price\_make}")

# 2.7 Minimum selling price by interior type

min\_price\_by\_interior = df.groupby('interior')['sellingprice'].min()

print("\nMinimum selling price by interior:")

print(min\_price\_by\_interior)

# 2.8 Highest odometer reading per year

highest\_odometer\_per\_year = df.groupby('year')['odometer'].max().sort\_values(ascending=False)

print("\nHighest odometer reading per year:")

print(highest\_odometer\_per\_year)

# 2.9 Create a 'car\_age' column

df['car\_age'] = 2025 - df['year']

print("\n'car\_age' column created.")

# 2.10 Number of cars with high condition and odometer

high\_condition\_odometer\_cars = df[(df['condition'] >= 48) & (df['odometer'] > 90000)].shape[0]

print(f"\nCars with condition >= 48 and odometer > 90000: {high\_condition\_odometer\_cars}")

# 2.11 State with highest prices for newer cars (year > 2013)

highest\_price\_state\_newer\_cars = df[df['year'] > 2013].groupby('state')['sellingprice'].mean().idxmax()

print(f"\nState with highest prices for newer cars: {highest\_price\_state\_newer\_cars}")

# 2.12 Value-for-money makes (excellent condition, lowest avg price)

excellent\_condition\_threshold = df['condition'].quantile(0.8)

excellent\_condition\_cars = df[df['condition'] >= excellent\_condition\_threshold]

lowest\_price\_makes = excellent\_condition\_cars.groupby('make')['sellingprice'].mean().sort\_values().head(5)

print("\nValue-for-money makes in excellent condition:")

print(lowest\_price\_makes)

# 3.1 Correlation of numerical features

numerical\_df = df.select\_dtypes(include=['int64', 'float64'])

correlation\_matrix = numerical\_df.corr()

plt.figure(figsize=(12, 10))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Matrix of Numerical Features')

plt.show()

# 3.2 Average selling price by year (Bar plot used for discrete years)

avg\_price\_by\_year = df.groupby('year')['sellingprice'].mean()

plt.figure(figsize=(12, 6))

avg\_price\_by\_year.plot(kind='bar')

plt.title('Average Selling Price by Year')

plt.xlabel('Year')

plt.ylabel('Average Selling Price')

plt.show()

# 3.3 Average selling price by odometer (Scatter plot)

plt.figure(figsize=(12, 6))

plt.scatter(df['odometer'], df['sellingprice'], alpha=0.5)

plt.title('Selling Price vs. Odometer Reading')

plt.xlabel('Odometer Reading')

plt.ylabel('Selling Price')

plt.show()

# 3.4 Number of cars sold in each state (Bar plot)

cars\_by\_state = df['state'].value\_counts()

plt.figure(figsize=(15, 7))

cars\_by\_state.plot(kind='bar')

plt.title('Number of Cars Sold per State')

plt.xlabel('State')

plt.ylabel('Number of Cars Sold')

plt.xticks(rotation=45)

plt.show()

# 3.5 Average selling price by condition score ranges (size 5)

df['condition\_range\_5'] = pd.cut(df['condition'], bins=range(0, 55, 5), right=False)

avg\_price\_by\_condition = df.groupby('condition\_range\_5')['sellingprice'].mean()

plt.figure(figsize=(12, 6))

avg\_price\_by\_condition.plot(kind='bar')

plt.title('Average Selling Price by Condition Score Range (Size 5)')

plt.xlabel('Condition Score Range')

plt.ylabel('Average Selling Price')

plt.xticks(rotation=45)

plt.show()

# 3.6 Number of cars sold by condition ranges (size 10)

df['condition\_range\_10'] = pd.cut(df['condition'], bins=range(0, 60, 10), right=False)

cars\_by\_condition\_range = df['condition\_range\_10'].value\_counts().sort\_index()

plt.figure(figsize=(12, 6))

cars\_by\_condition\_range.plot(kind='bar')

plt.title('Number of Cars Sold by Condition Score Range (Size 10)')

plt.xlabel('Condition Score Range')

plt.ylabel('Number of Cars Sold')

plt.xticks(rotation=45)

plt.show()

# 3.7 Box plot of selling prices grouped by color

plt.figure(figsize=(15, 8))

sns.boxplot(x='color', y='sellingprice', data=df)

plt.title('Distribution of Selling Price by Color (with outliers)')

plt.xlabel('Color')

plt.ylabel('Selling Price')

plt.xticks(rotation=45)

plt.show()

# Box plot with outliers removed (top 1%)

price\_99\_percentile = df['sellingprice'].quantile(0.99)

df\_no\_outliers = df[df['sellingprice'] <= price\_99\_percentile]

plt.figure(figsize=(15, 8))

sns.boxplot(x='color', y='sellingprice', data=df\_no\_outliers)

plt.title('Distribution of Selling Price by Color (Outliers Removed)')

plt.xlabel('Color')

plt.ylabel('Selling Price')

plt.xticks(rotation=45)

plt.show()