

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('/content/drive/MyDrive/AI CODE/BMW.CSV')
df.head()
```

	Year	Month	Region	Model	Units_Sold	Avg_Price_EUR	Revenue_EUR	BEV_Share	Premium_Share
0	2018	1	Europe	3 Series	7822	47482	371404204	0.011	19.7
1	2018	1	Europe	5 Series	10280	61685	634121800	0.019	19.7
2	2018	1	Europe	X3	3105	58433	181434465	0.022	19.7
3	2018	1	Europe	X5	7420	67955	504226100	0.021	19.7
4	2018	1	Europe	X7	8474	92300	782150200	0.035	19.7

Next steps:

[Generate code with df](#)

[New interactive sheet](#)

df.dtypes

	0
Year	int64
Month	int64
Region	object
Model	object
Units_Sold	int64
Avg_Price_EUR	int64
Revenue_EUR	int64
BEV_Share	float64
Premium_Share	float64
GDP_Growth	float64
Fuel_Price_Index	float64

dtype: object

df.columns

```
Index(['Year', 'Month', 'Region', 'Model', 'Units_Sold', 'Avg_Price_EUR',
      'Revenue_EUR', 'BEV_Share', 'Premium_Share', 'GDP_Growth',
      'Fuel_Price_Index'],
      dtype='object')
```

```
df.isnull().sum()
```

	0
Year	0
Month	0
Region	0
Model	0
Units_Sold	0
Avg_Price_EUR	0
Revenue_EUR	0
BEV_Share	0
Premium_Share	0
GDP_Growth	0
Fuel_Price_Index	1065

```
dtype: int64
```

```
missing_percentage = df.isnull().mean() * 100
threshold = 30
# Filter columns with more than the threshold percentage of missing values
columns_to_drop = missing_percentage[missing_percentage > threshold].index
# Drop the columns
df = df.drop(columns=columns_to_drop)
# Display the resulting dataframe after removing columns with too many missing values
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")
```

```
Columns dropped: ['Fuel_Price_Index']
Remaining columns: Index(['Year', 'Month', 'Region', 'Model', 'Units_Sold', 'Avg_Price_EUR',
      'Revenue_EUR', 'BEV_Share', 'Premium_Share', 'GDP_Growth'],
      dtype='object')
```

```
#Pre-processing on dataset
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for column in df.select_dtypes(include=['object']).columns:
    df[column] = label_encoder.fit_transform(df[column])
df.head()
```

	Year	Month	Region	Model	Units_Sold	Avg_Price_EUR	Revenue_EUR	BEV_Share	Premium_Share
0	2018	1	1	0	7822	47482	371404204	0.011	19.1
1	2018	1	1	1	10280	61685	634121800	0.019	19.1
2	2018	1	1	3	3105	58433	181434465	0.022	19.1
3	2018	1	1	4	7420	67955	504226100	0.021	19.1
4	2018	1	1	5	8474	92300	782150200	0.035	19.1

Next steps:

[Generate code with df](#)
[New interactive sheet](#)

```
# Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean() # Get the mode (most frequent value)
    df[column] = df[column].fillna(mean_value)
# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0] # Get the mode (most frequent value)
    df[column] = df[column].fillna(mode_value)
df.isnull().sum()
```

	0
Year	0
Month	0
Region	0
Model	0
Units_Sold	0
Avg_Price_EUR	0
Revenue_EUR	0
BEV_Share	0
Premium_Share	0
GDP_Growth	0

dtype: int64

```
#Using IQR (Interquartile Range):
data=df.copy()
def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Return a boolean series indicating if the value is an outlier
    return (column < lower_bound) | (column > upper_bound)

# Apply the IQR method to each feature in the DataFrame
for col in data.columns:
    data[f'{col}_Outlier'] = detect_outliers_iqr(data[col])
```

```
# Filter the DataFrame to display rows where any outlier flag is True
outliers_only = data[data.filter(like='_Outlier').any(axis=1)]
# Display the filtered DataFrame with outliers
outliers_only
```

	Year	Month	Region	Model	Units_Sold	Avg_Price_EUR	Revenue_EUR	BEV_Share	Premium_5
452	2019	3	1	5	13295	91058	1210616110	0.046	
836	2020	3	1	5	13645	91053	1242418185	0.059	
1124	2020	12	1	5	13404	93509	1253394636	0.074	
1316	2021	6	1	5	13799	90011	1242061789	0.104	
1428	2021	9	3	5	14215	93386	1327481990	0.092	
1516	2021	12	0	5	14353	91024	1306467472	0.095	
1524	2021	12	3	5	13944	90129	1256758776	0.078	
1700	2022	6	1	5	14100	91334	1287809400	0.112	
1744	2022	7	3	0	10314	47968	494741952	0.130	
1745	2022	7	3	1	3711	61105	226760655	0.133	
1746	2022	7	3	3	4258	57425	244515650	0.112	
1747	2022	7	3	4	9739	69440	676276160	0.134	
1748	2022	7	3	5	9954	93539	931087206	0.128	
1749	2022	7	3	6	6365	66275	421840375	0.117	
1750	2022	7	3	7	5044	76350	385109400	0.140	
1751	2022	7	3	2	7885	40403	318577655	0.123	
1820	2022	9	2	5	13417	91272	1224596424	0.118	
1892	2022	12	1	5	13906	90555	1259257830	0.117	
2108	2023	6	2	5	13568	92488	1254877184	0.151	
2404	2024	4	1	5	13423	92633	1243412759	0.187	
2436	2024	5	1	5	13376	91495	1223837120	0.199	
2444	2024	5	0	5	13421	93816	1259104536	0.166	
2500	2024	7	1	5	13218	92869	1227542442	0.157	
2572	2024	9	0	5	15061	90157	1357854577	0.160	
2612	2024	10	3	5	13352	91830	1226114160	0.156	
2668	2024	12	0	5	14810	92812	1374545720	0.173	
2764	2025	3	0	5	15448	92794	1433481712	0.205	
2852	2025	6	1	5	13947	90006	1255313682	0.197	
2876	2025	6	2	5	14786	92278	1364422508	0.198	
2956	2025	9	0	5	14802	90096	1333600992	0.204	
3044	2025	12	1	5	14561	93587	1362720307	0.197	

Next steps:

[Generate code with outliers_only](#)

[New interactive sheet](#)

```

data=df.copy()

def replace_outliers_with_nan(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Replace outliers with NaN
    return column.where((column >= lower_bound) & (column <= upper_bound), np.nan)

# Replace outliers with NaN for each feature
for col in data.columns:
    data[col] = replace_outliers_with_nan(data[col])

null_counts = data.isnull().sum()
print("Sum of null values in each column:")
print(null_counts)

```

Sum of null values in each column:

```

Year          0
Month         0
Region        0
Model         0
Units_Sold    0
Avg_Price_EUR 0
Revenue_EUR   23
BEV_Share     0
Premium_Share 0
GDP_Growth    8
dtype: int64

```

```

# Using the Interquartile Range (IQR) Method
data=df.copy()

def remove_outliers_iqr(data):
    # Calculate Q1 (25th percentile) and Q3 (75th percentile) for each feature
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    # Define lower and upper bounds
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Filter the dataset to remove rows with outliers in any column
    data_cleaned = data[~((data < lower_bound) | (data > upper_bound)).any(axis=1)]
    return data_cleaned

cleaned_data = remove_outliers_iqr(data)
print("Original Data:")
print(len(data))
print("\nCleaned Data (without outliers):")
print(len(cleaned_data))

```

Original Data:
3072

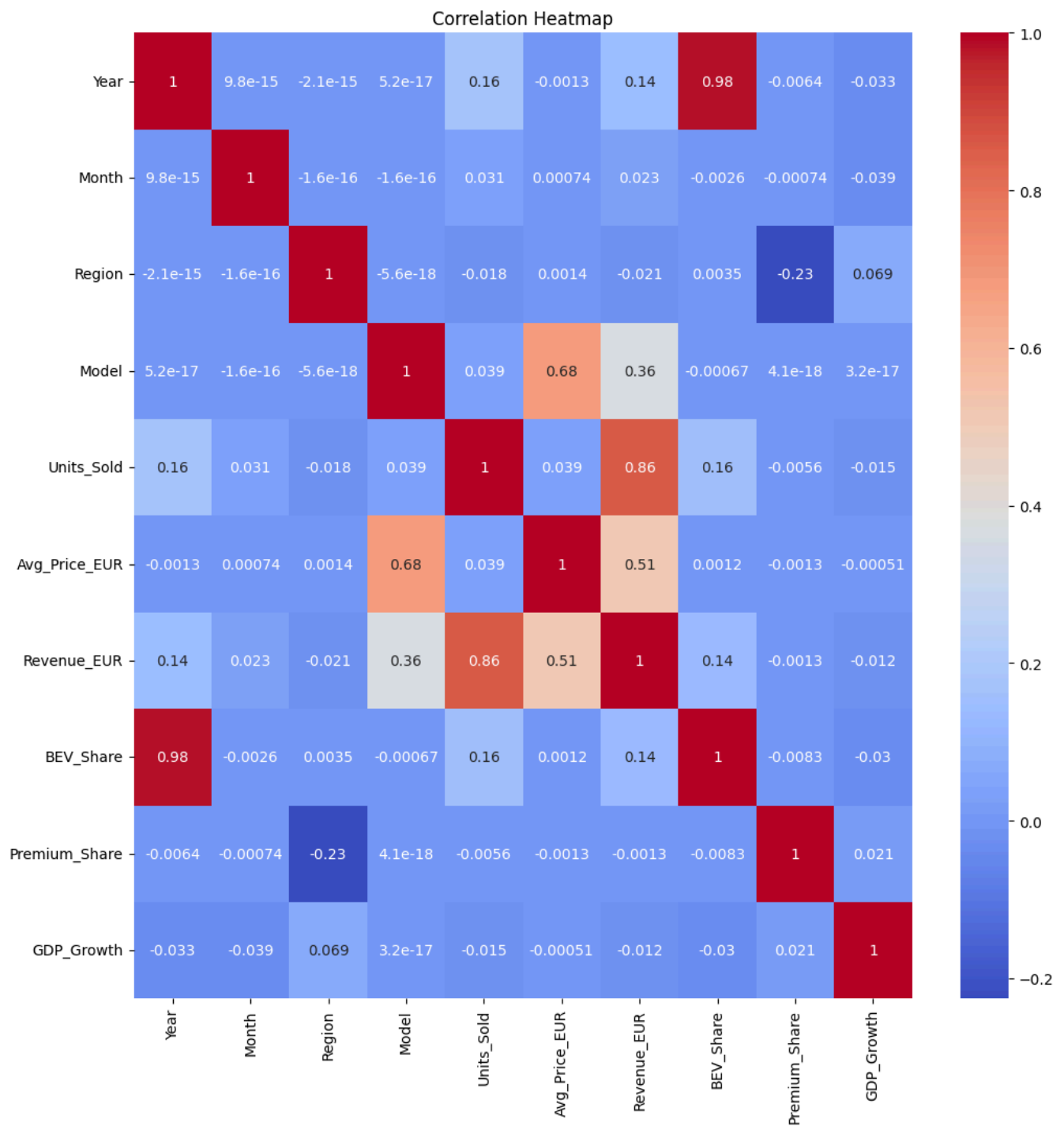
Cleaned Data (without outliers):
3041

```

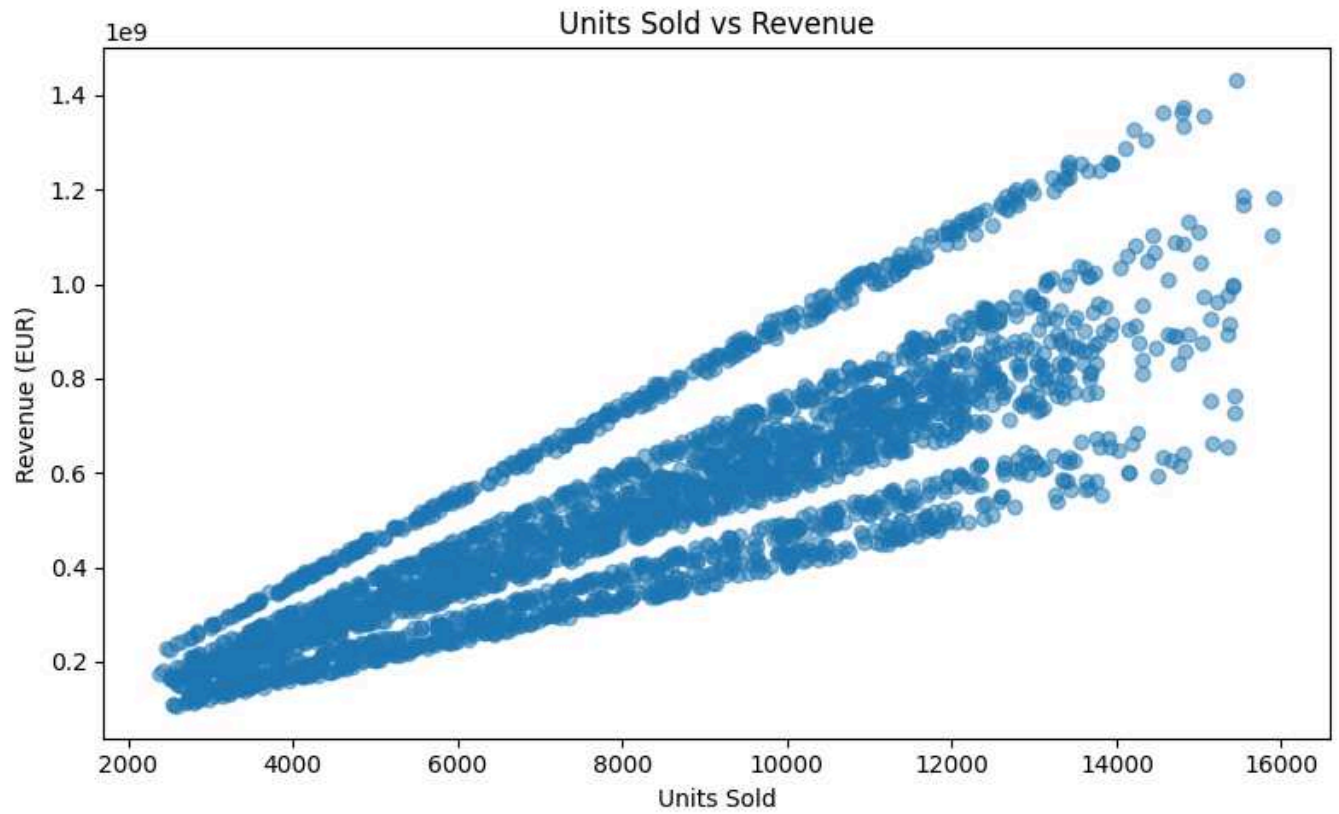
plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')

```

```
plt.show()
```

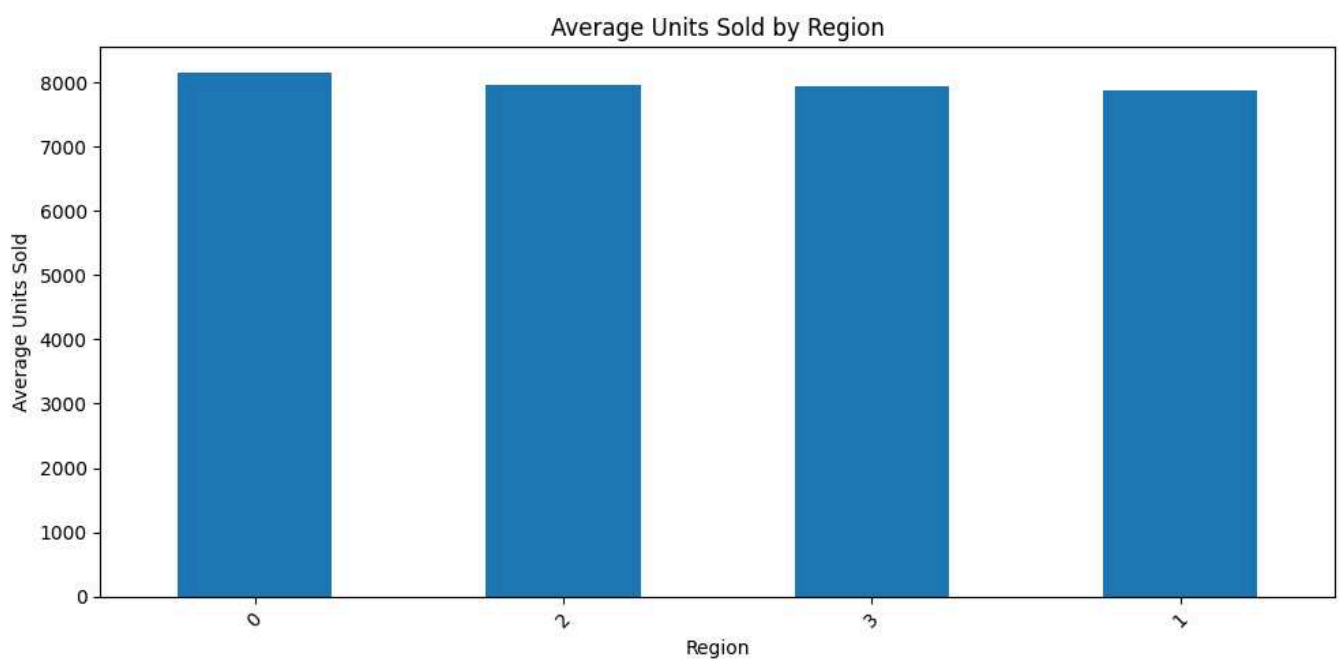


```
plt.figure(figsize=(8, 5))
plt.scatter(df["Units_Sold"], df["Revenue_EUR"], alpha=0.5)
plt.title("Units Sold vs Revenue")
plt.xlabel("Units Sold")
plt.ylabel("Revenue (EUR)")
plt.tight_layout()
plt.show()
```



```
region_sales = df.groupby("Region")["Units_Sold"].mean().sort_values(ascending=False)

plt.figure(figsize=(10, 5))
region_sales.plot(kind="bar")
plt.title("Average Units Sold by Region")
plt.xlabel("Region")
plt.ylabel("Average Units Sold")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



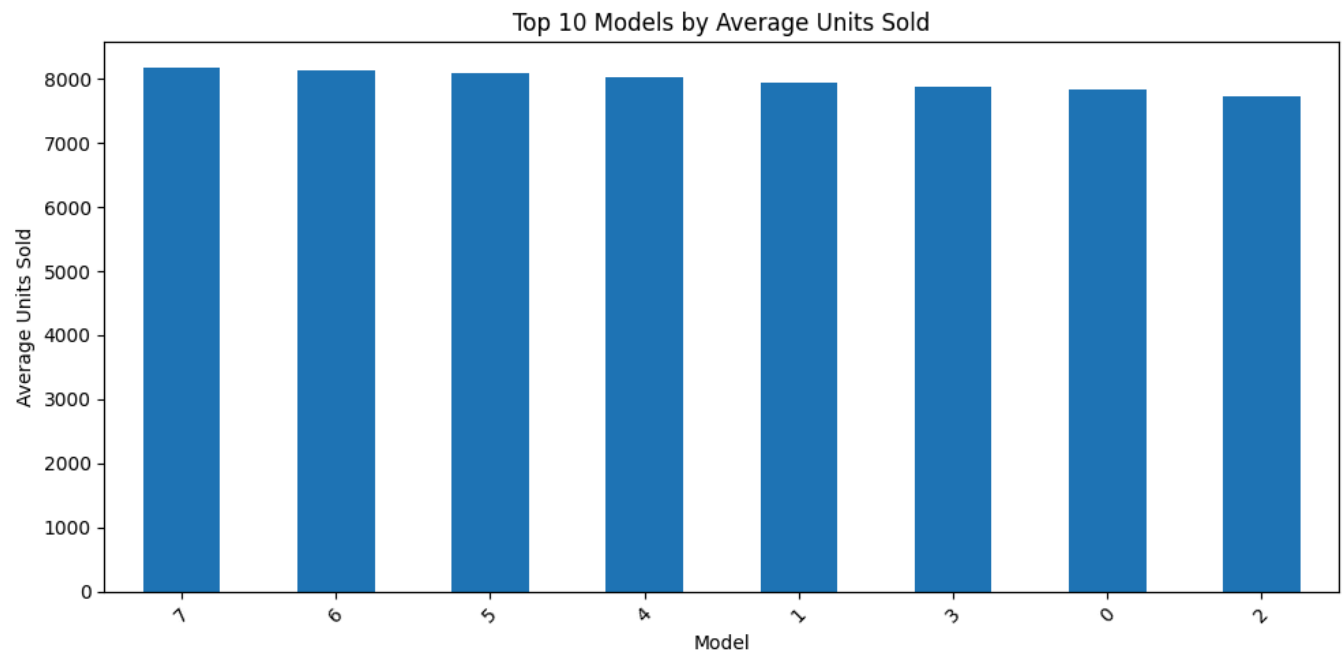
```
model_sales = df.groupby("Model")["Units_Sold"].mean().sort_values(ascending=False).head(10)

plt.figure(figsize=(10, 5))
```

```

model_sales.plot(kind="bar")
plt.title("Top 10 Models by Average Units Sold")
plt.xlabel("Model")
plt.ylabel("Average Units Sold")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

```



```

numeric_df = df.select_dtypes(include=[np.number])

corr = numeric_df.corr()

plt.figure(figsize=(10, 8))
plt.imshow(corr, interpolation='nearest', aspect='auto')
plt.colorbar()
plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
plt.yticks(range(len(corr.columns)), corr.columns)
plt.title("Correlation Heatmap")
plt.tight_layout()
plt.show()

```

