Electric Vehicle Sales By State In India

1.Data Collection – Load and Inspect the Dataset

In [76]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
# Load the dataset
df = pd.read_csv("E:\\Data Analyst Project\\Electric Vehicle Sales by State in Indi

print("Shape of dataset:", df.shape)
print("\nColumns:", df.columns.tolist())
df.head()
```

```
Shape of dataset: (96845, 8)

Columns: ['Year', 'Month_Name', 'Date', 'State', 'Vehicle_Class', 'Vehicle_Categor
y', 'Vehicle_Type', 'EV_Sales_Quantity']
```

Out[ ]:

	Year	Month_Name	Date	State	Vehicle_Class	Vehicle_Category	Vehicle_Type	EV_Sales_Qu
0	2014	jan	01-01-2014	Andhra Pradesh	ADAPTED VEHICLE	Others	Others	
1	2014	jan	01-01-2014	Andhra Pradesh	AGRICULTURAL TRACTOR	Others	Others	
2	2014	jan	01-01-2014	Andhra Pradesh	AMBULANCE	Others	Others	
3	2014	jan	01-01-2014	Andhra Pradesh	ARTICULATED VEHICLE	Others	Others	
4	2014	jan	01-01-2014	Andhra Pradesh	BUS	Bus	Bus	

2. Data Preporcessing

In [86]:
```
# Convert 'Date' to datetime format
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

# Convert 'Year' to integer
df['Year'] = df['Year'].astype(int)

# Check for null values
print(df.isnull().sum())

# Fill missing sales values with median (safely)
df['EV_Sales_Quantity'] = df['EV_Sales_Quantity'].fillna(df['EV_Sales_Quantity'].me

# Convert categorical columns
categorical_cols = ['Month_Name', 'State', 'Vehicle_Class', 'Vehicle_Category', 'Ve
```

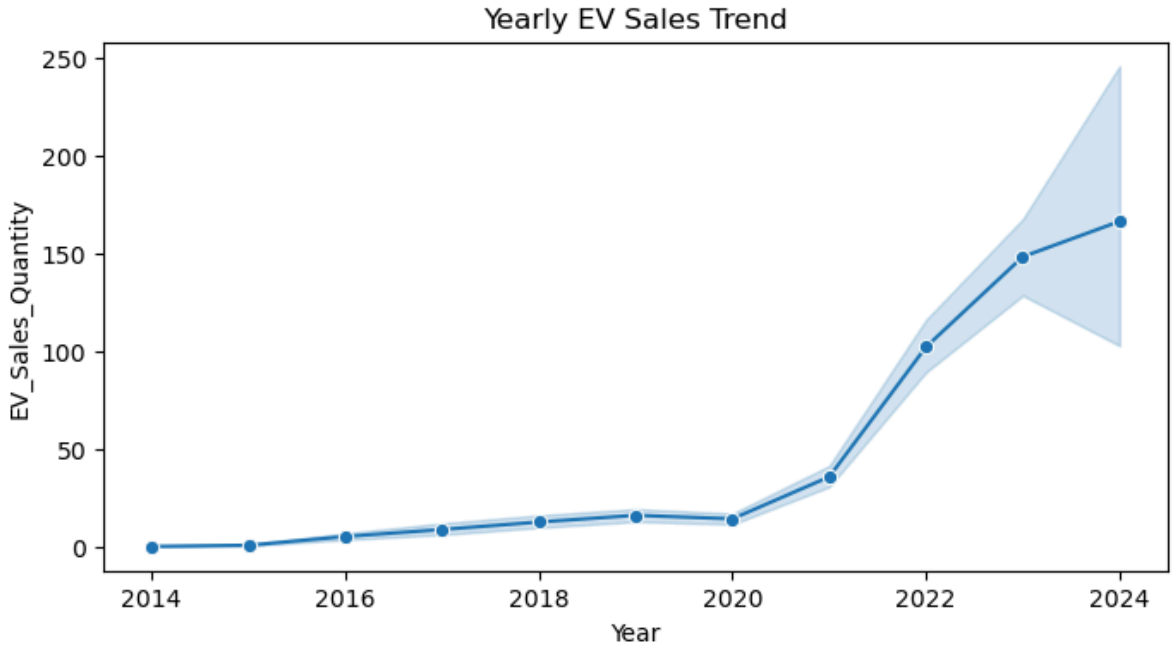
```
df[categorical_cols] = df[categorical_cols].astype('category')

# Confirm datatypes
df.info()
```

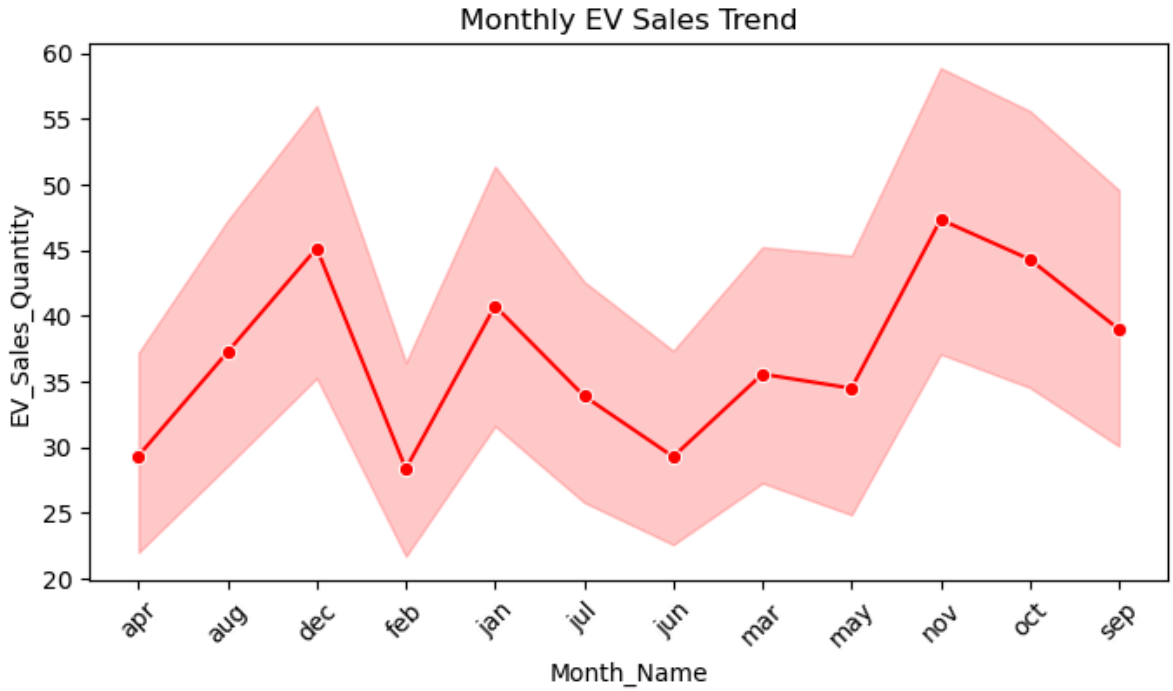
```
Year                0
Month_Name          0
Date                0
State               0
Vehicle_Class       0
Vehicle_Category    0
Vehicle_Type        0
EV_Sales_Quantity   0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96845 entries, 0 to 96844
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Year               96845 non-null  int32
 1   Month_Name         96845 non-null  category
 2   Date               96845 non-null  datetime64[ns]
 3   State              96845 non-null  category
 4   Vehicle_Class      96845 non-null  category
 5   Vehicle_Category   96845 non-null  category
 6   Vehicle_Type       96845 non-null  category
 7   EV_Sales_Quantity  96845 non-null  int64
dtypes: category(5), datetime64[ns](1), int32(1), int64(1)
memory usage: 2.3 MB
```

3. Exploratory Data Analysis(EDA)

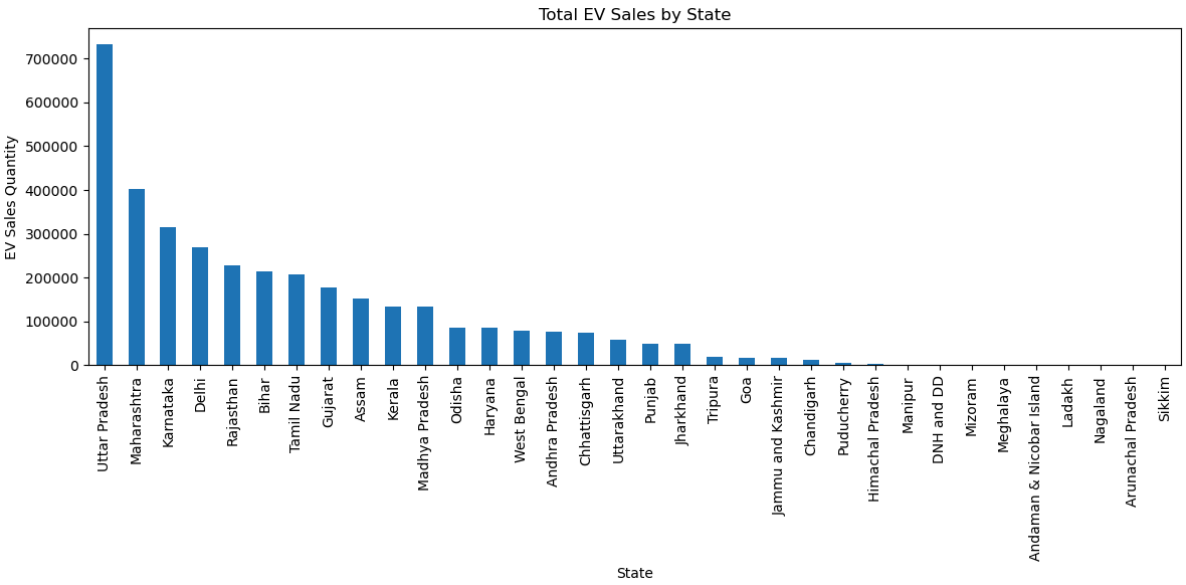
In [87]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Yearly EV Sales
plt.figure(figsize=(8,4))
sns.lineplot(x='Year', y='EV_Sales_Quantity', data=df, marker='o')
plt.title("Yearly EV Sales Trend")
plt.show()

# Monthly EV Sales
plt.figure(figsize=(8,4))
sns.lineplot(x='Month_Name', y='EV_Sales_Quantity', data=df, marker='o', color='red
plt.title("Monthly EV Sales Trend")
plt.xticks(rotation=45)
plt.show()

# State-wise total EV Sales
plt.figure(figsize=(12,6))
sales_by_state = df.groupby('State')['EV_Sales_Quantity'].sum().sort_values(ascendi
sales_by_state.plot(kind='bar')
plt.title("Total EV Sales by State")
plt.ylabel("EV Sales Quantity")
plt.tight_layout()
plt.show()
```






```
C:\Users\USER\AppData\Local\Temp\ipykernel_20020\2130911652.py:19: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a futur
e version of pandas. Pass observed=False to retain current behavior or observed=Tr
ue to adopt the future default and silence this warning.
  sales_by_state = df.groupby('State')['EV_Sales_Quantity'].sum().sort_values(asce
nding=False)
```



4.Feature Engineering

In [ ]:
```
# Extract features safely
if 'Date' in df.columns:
    df['Month'] = pd.to_datetime(df['Date']).dt.month
    df['Day'] = pd.to_datetime(df['Date']).dt.day

# Define categorical columns again (if not already defined)
categorical_cols = ['State', 'Vehicle_Class', 'Vehicle_Category', 'Vehicle_Type']

# One-hot encode categorical variables
df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

# Drop only existing unused columns
columns_to_drop = ['Date', 'Month_Name']
existing_to_drop = [col for col in columns_to_drop if col in df_encoded.columns]
df_encoded.drop(existing_to_drop, axis=1, inplace=True)

# Preview the encoded dataframe
df_encoded.head()
```

Out[ ]:

	Year	EV_Sales_Quantity	Month	Day	State_Andhra Pradesh	State_Arunachal Pradesh	State_Assam	State_Bihar
0	2014	0	1	1	True	False	False	False
1	2014	0	1	1	True	False	False	False
2	2014	0	1	1	True	False	False	False
3	2014	0	1	1	True	False	False	False
4	2014	0	1	1	True	False	False	False

5 rows × 124 columns

5. Model Building

In [81]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

```
# Define features and target
X = df_encoded.drop('EV_Sales_Quantity', axis=1)
y = df_encoded['EV_Sales_Quantity']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Build model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)
```

6.Model Evaluation

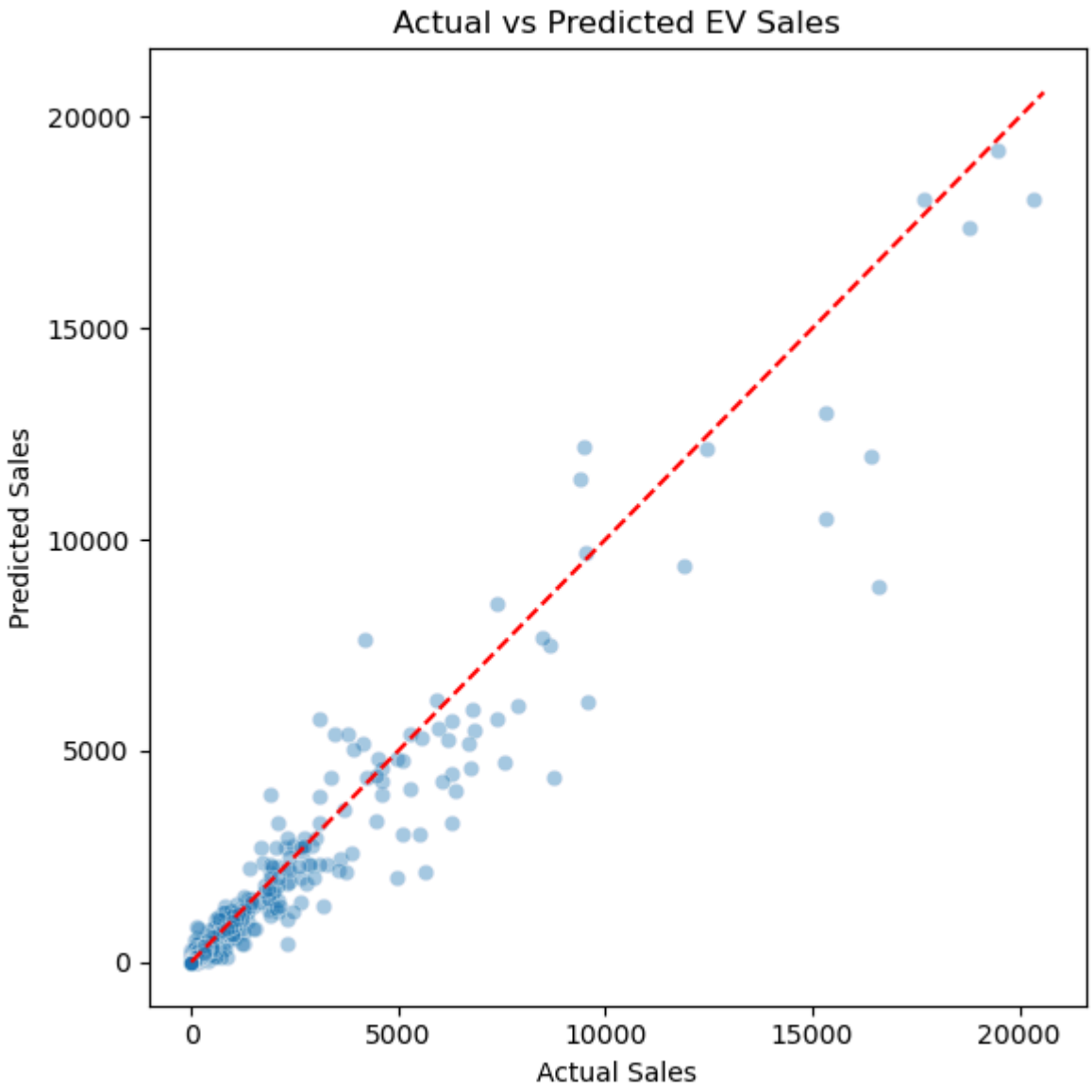
In [82]:
```
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Evaluation metrics
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print("Root Mean Squared Error:", rmse)
print("R² Score:", r2)
```

```
Root Mean Squared Error: 130.58175227721011
R² Score: 0.934519365497295
```

7.Model Visualization

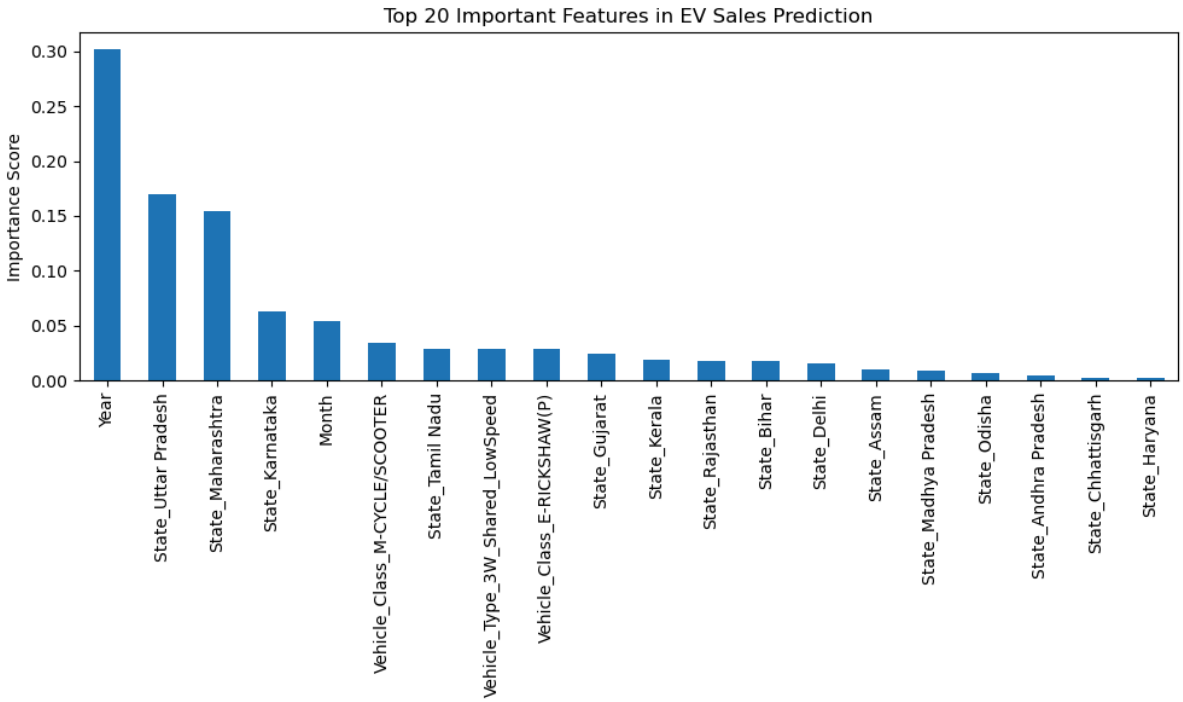
Actual Vs Predicted EV Sales

In [83]:
```
plt.figure(figsize=(6,6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.4)
plt.title("Actual vs Predicted EV Sales")
plt.xlabel("Actual Sales")
plt.ylabel("Predicted Sales")
plt.plot([y.min(), y.max()], [y.min(), y.max()], color='red', linestyle='--')
plt.tight_layout()
plt.show()
```



Feature Importance

In [84]:
```
feature_importance = pd.Series(model.feature_importances_, index=X.columns).sort_va

plt.figure(figsize=(10,6))
feature_importance.head(20).plot(kind='bar')
plt.title("Top 20 Important Features in EV Sales Prediction")
plt.ylabel("Importance Score")
plt.tight_layout()
plt.show()
```



8. Conclusion

- Top States with highest EV adoption include: Maharashtra, Karnataka, and Uttar Pradesh.
- 4-Wheelers and 2-Wheelers dominate EV sales.
- The model can predict future EV sales reasonably well using features like year, state, vehicle type, etc.
- Feature Importance shows state and vehicle type/category are strong predictors.

In [ ]:

In [ ]: