EV MARKET SEGMENTATION ANALYSIS

An electric vehicle (EV) is a type of vehicle that is powered by one or more electric motors using electricity stored in batteries or obtained from an external source such as a charging station.

Unlike conventional vehicles that rely on internal combustion engines fueled by gasoline or diesel, electric vehicles use electricity to generate the energy needed to propel the vehicle.

HERE IS THE FULL ANALYSIS OF EV INDIAN AUTOMOBILE AND EV MARKET

Q 1. WHICH COMPANIES ARE DOMINATING IN INDIAN MARKET ?

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
```

In [2]:
```
cars_data = pd.read_csv("cars_ds_final.csv")
```

In [3]:
```
cars_data
```

Out[3]:

	Unnamed: 0	Make	Model	Variant	Ex-Showroom_Price	Displacement	Cylinders	Valves_Per_
0	0	Tata	Nano Genx	Xt	Rs. 2,92,667	624 cc	2.0	
1	1	Tata	Nano Genx	Xe	Rs. 2,36,447	624 cc	2.0	
2	2	Tata	Nano Genx	Emax Xm	Rs. 2,96,661	624 cc	2.0	
3	3	Tata	Nano Genx	Xta	Rs. 3,34,768	624 cc	2.0	
4	4	Tata	Nano Genx	Xm	Rs. 2,72,223	624 cc	2.0	
...	...	...	...	...	...	...	...	...

	Unnamed: 0	Make	Model	Variant	Ex-Showroom_Price	Displacement	Cylinders	Valves_Per
1271	1271	Honda	City	Vx Mt Diesel	Rs. 13,02,000	1498 cc	4.0	
1272	1272	Honda	City	Zx Mt Diesel	Rs. 14,21,000	1498 cc	4.0	
1273	1273	Honda	City	Zx Cvt Petrol	Rs. 14,31,000	1497 cc	4.0	
1274	1274	Honda	City	V Cvt Petrol	Rs. 12,01,000	1497 cc	4.0	
1275	1275	Mitsubishi	Montero	3.2 At	Rs. 68,62,560	3200 cc	4.0	

1276 rows × 141 columns

Number of unique manufacutes and models

In [4]:
```
cars_data.nunique()
```

Out[4]:
```
Unnamed: 0           1276
Make                   39
Model                 263
Variant              1064
Ex-Showroom_Price    1179
                     ...
USB_Ports               3
Heads-Up_Display        1
Welcome_Lights          1
Battery                 5
Electric_Range          8
Length: 141, dtype: int64
```

In [5]:
```
cars_data["Make"].count()
```

Out[5]:
```
1201
```

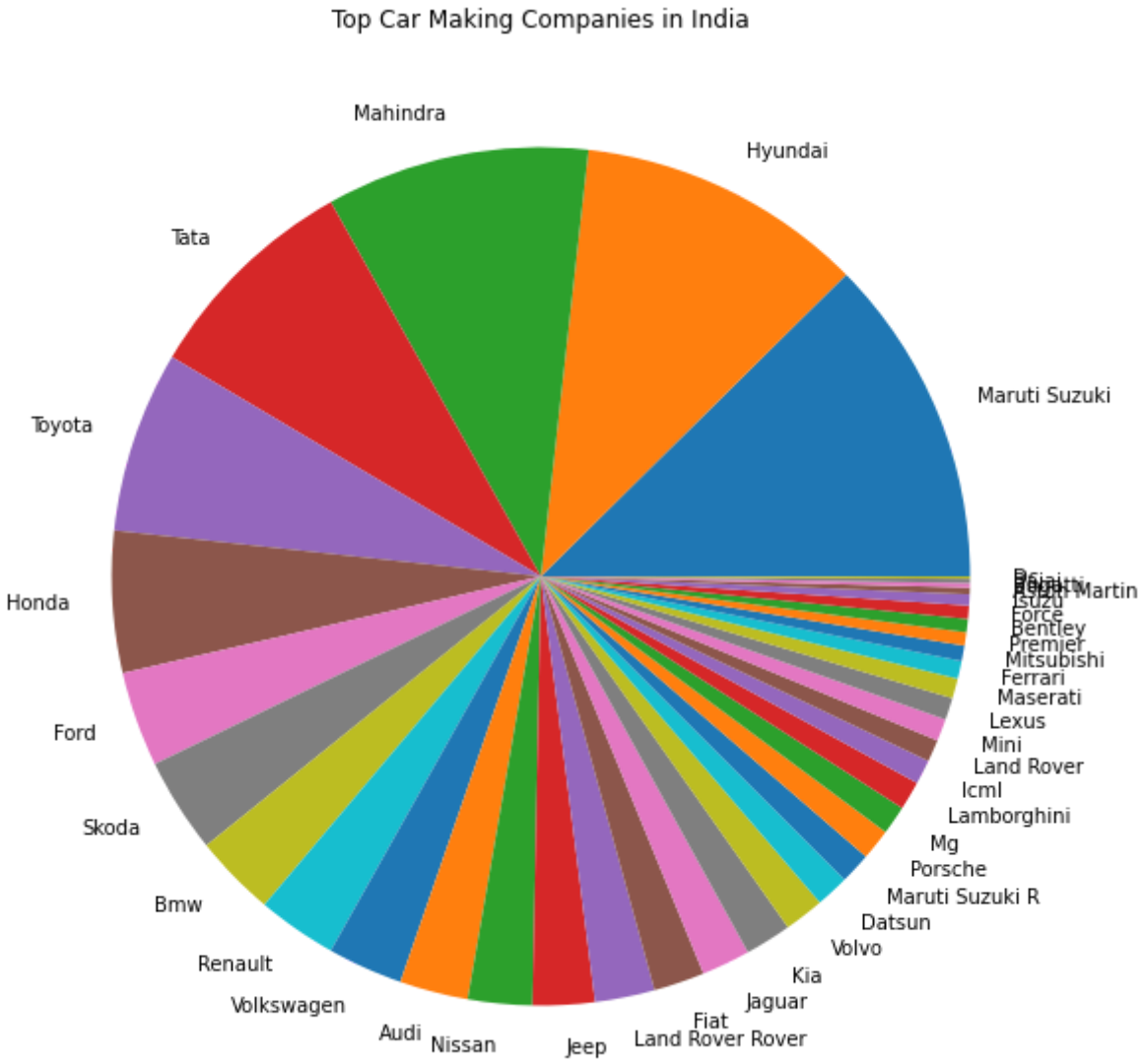
Major car manufactures in india

In [6]:
```
x = cars_data.iloc[:,1]
```

In [7]:
```
x.value_counts()
```

Out[7]:
```
Maruti Suzuki       149
Hyundai             130
Mahindra            119
Tata                100
Toyota               82
Honda                64
Ford                 43
Skoda                43
Bmw                  37
Renault              36
Volkswagen           34
Audi                 31
Nissan               29
Jeep                 28
Land Rover Rover     27
Fiat                 23
Jaguar               22
Kia                  21
Volvo                18
Datsun               15
Maruti Suzuki R      14
Porsche              14
Mg                   13
Lamborghini          13
Icml                 11
Land Rover           10
Mini                 10
Lexus                10
Maserati              9
Ferrari               8
Mitsubishi            7
Premier               6
Bentley               6
Force                 6
Isuzu                 5
Aston Martin          3
Bugatti               2
Bajaj                 2
Dc                    1
Name: Make, dtype: int64
```

In [8]:
```
fig = plt.figure(figsize = (10,10))
ax = fig.subplots()
x.value_counts().plot(ax=ax, kind='pie')
ax.set_ylabel("")
ax.set_title("Top Car Making Companies in India")
plt.show()
```



Hence more than 50% cars in india are manufactured by top 7 Companies

Q2. WHAT IS THE MOST DEMANDED CAR BODY TYPE IN INDIA ?

Most Demanded Body types in Car

In [9]:
```
body_types = cars_data.iloc[:,18]
```

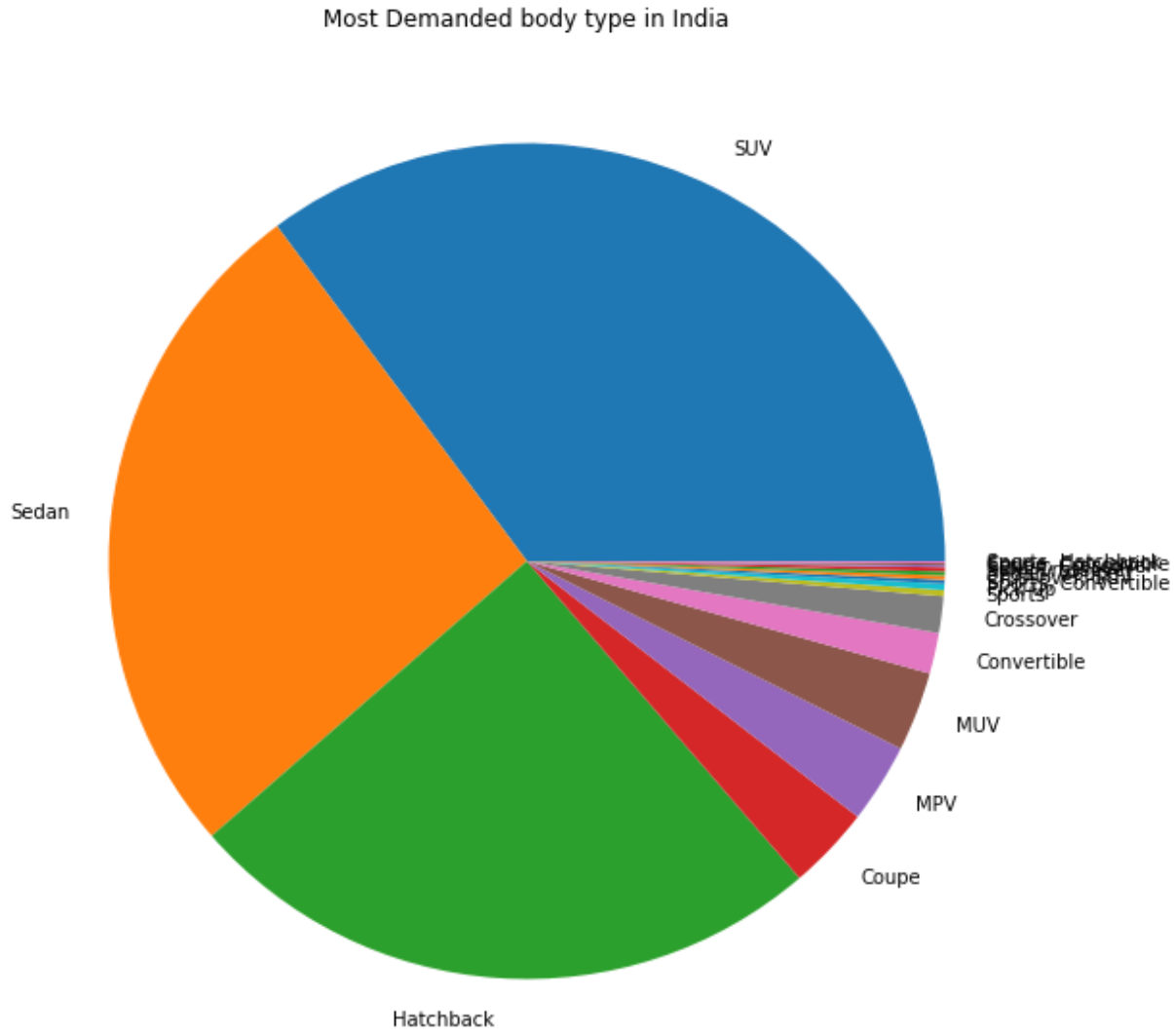
In [10]:
```
body_types.value_counts()
```

Out[10]:
```
SUV            447
Sedan          333
Hatchback      316
Coupe           41
MPV             39
MUV             39
Convertible     20
Crossover       18
Sports           3
```

```
Pick-up                 3
Sports, Convertible     2
Crossover, SUV          2
Sedan, Coupe            2
SUV, Crossover          2
Sedan, Crossover        1
Coupe, Convertible      1
Sports, Hatchback       1
Name: Body_Type, dtype: int64
```

In [11]:

```
fig = plt.figure(figsize = (10,10))
ax = fig.subplots()
body_types.value_counts().plot(ax=ax, kind='pie')
ax.set_ylabel("")
ax.set_title("Most Demanded body type in India")
plt.show()
```



Hence Most demanded body type in india is SUV,SEDAN AND HATCHBACKS

Q3.WHAT IS THE AVERAGE PRICE OF THESE MOST DEMANDES BODY TYPE IN INDIA ?

HatchBack

In [12]:
```
hatch = cars_data[cars_data["Body_Type"] == "Hatchback"]
```

In [13]:
```
hatch.count()
```

Out[13]:
```
Unnamed: 0           316
Make                 314
Model                316
Variant              316
Ex-Showroom_Price    316
                    ...
USB_Ports              0
Heads-Up_Display       3
Welcome_Lights         6
Battery                2
Electric_Range         2
Length: 141, dtype: int64
```

In [14]:
```
hatch_price = hatch["Ex-Showroom_Price"]
```

In [15]:
```
price = []
for i in range (0,142):
    l = hatch_price.iloc[i]
    l = l.split(" ")

    price.append(int(l[1].replace(",","")))
```

In [16]:
```
hatch_price = pd.DataFrame(price)
```

In [17]:
```
hatch_price.describe()
```

Out[17]:

	0
count	142.000000
mean	602380.577465
std	169335.451070
min	236447.000000
25%	480771.000000
50%	592932.000000
75%	721848.250000
max	988500.000000

In [18]:
```
hatch_mean = int(hatch_price.mean())
print(hatch_mean)
```

```
602380
```

SUV

In [19]:
```
suv = cars_data[cars_data["Body_Type"] == "SUV"]
```

In [20]:
```
suv_price = suv["Ex-Showroom_Price"]
```

In [21]:
```
price = []
for i in range (0,142):
    l = suv_price.iloc[i].split(" ")

    price.append(int(l[1].replace(",","")))
```

In [22]:
```
suv_price = pd.DataFrame(price)
```

In [23]:
```
suv_mean = int(suv_price.mean())
print(suv_mean)
```

```
4354507
```

Sedan

In [24]:
```
sedan = cars_data[cars_data["Body_Type"] == "Sedan"]
```

In [25]:
```
sedan_price = sedan["Ex-Showroom_Price"]
```

In [26]:
```
price = []
for i in range (0,142):
    l = sedan_price.iloc[i].split(" ")

    price.append(int(l[1].replace(",","")))
```

In [27]:
```
sedan_price = pd.DataFrame(price)
```

In [28]:
```
sedan_price.describe()
```

Out[28]:

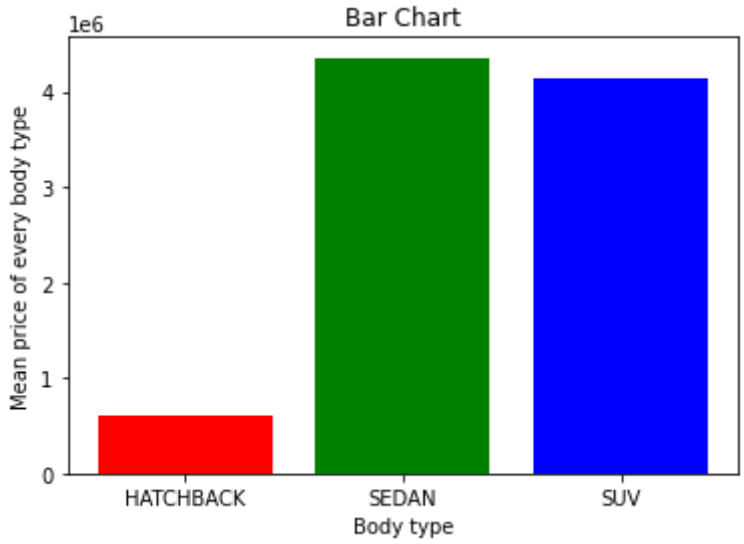
	0
count	1.420000e+02
mean	4.131652e+06
std	8.708078e+06
min	5.368590e+05
25%	7.132750e+05
50%	8.195065e+05
75%	3.166400e+06
max	5.324720e+07

In [29]:
```
sedan_mean = int(sedan_price.mean())
print(sedan_mean)
```

```
4131652
```

BAR-CHART

In [30]:
```
p = ["HATCHBACK","SEDAN","SUV"]
h = [hatch_mean, suv_mean,sedan_mean]
plt.bar(p,h,color = ("red","green","blue"))
plt.title('Bar Chart')
plt.xlabel("Body type")
plt.ylabel("Mean price of every body type")
plt.show()
```



As we can conclude the Hatchbacks are cheapers than Suv and Sedans

Q4. WHICH FUEL TYPES IS MOST DEMANDED IN INDIAN MARKET ?

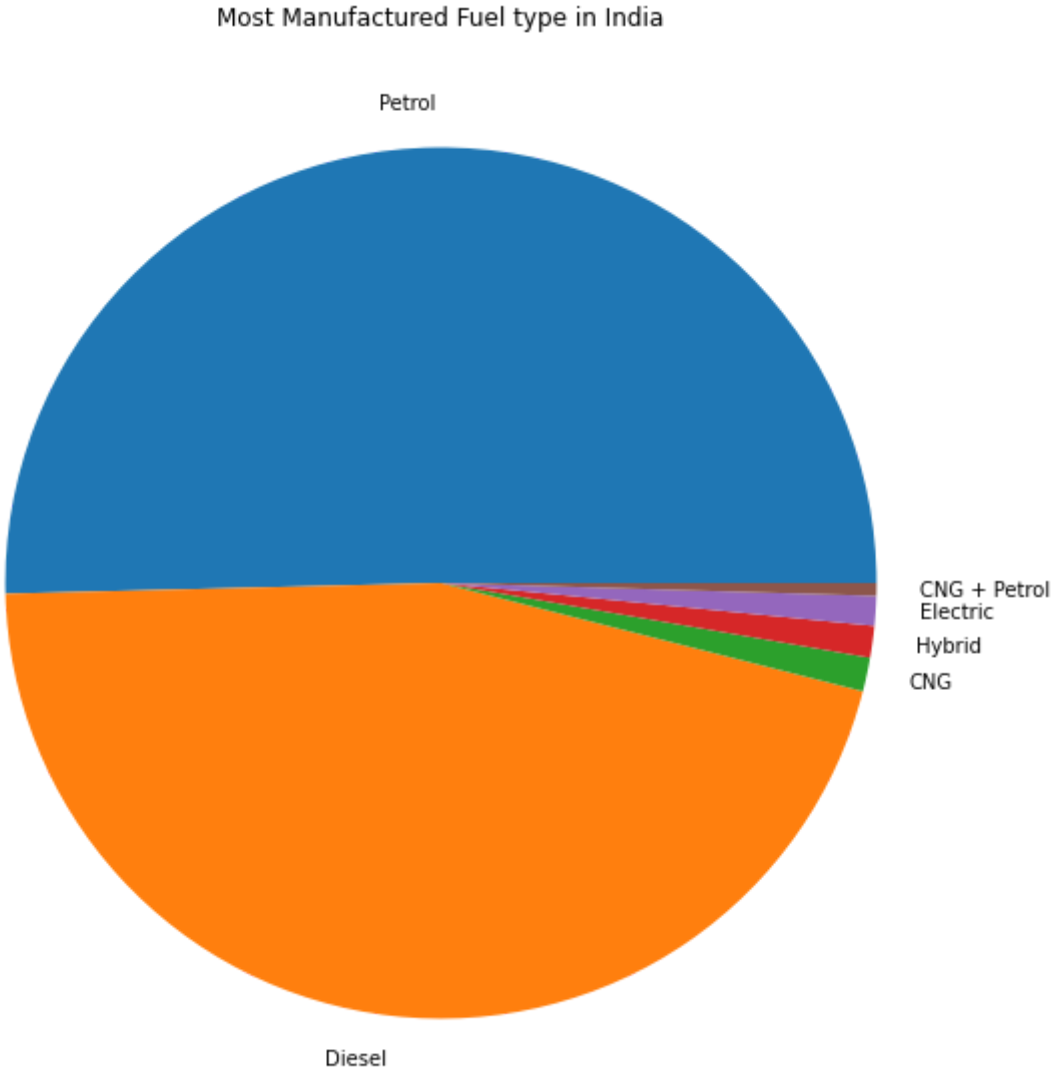
FUEL TYPE

In [31]:
```
fuel_types = cars_data.iloc[:,14]
```

In [32]:
```
fuel_types.value_counts()
```

Out[32]:
```
Petrol          643
Diesel          582
CNG              16
Hybrid           15
Electric         14
CNG + Petrol      6
Name: Fuel_Type, dtype: int64
```

In [33]:
```
fig = plt.figure(figsize = (10,10))
ax = fig.subplots()
fuel_types.value_counts().plot(ax=ax, kind='pie')
ax.set_ylabel("")
ax.set_title("Most Manufactured Fuel type in India")
plt.show()
```



In [34]:
```
fuel_types.count()
```

Out[34]: 1276

Out of 1276 models only 14 vechiles are made in elctric segments i.e approx 1%

Thats why There is strong opportunity and space to enter in EV market Segment

Q5. HOW INDIAN AUTOMOBILE IS PERFORMING RECENTLY ?

CAR SALES IN INDIAN

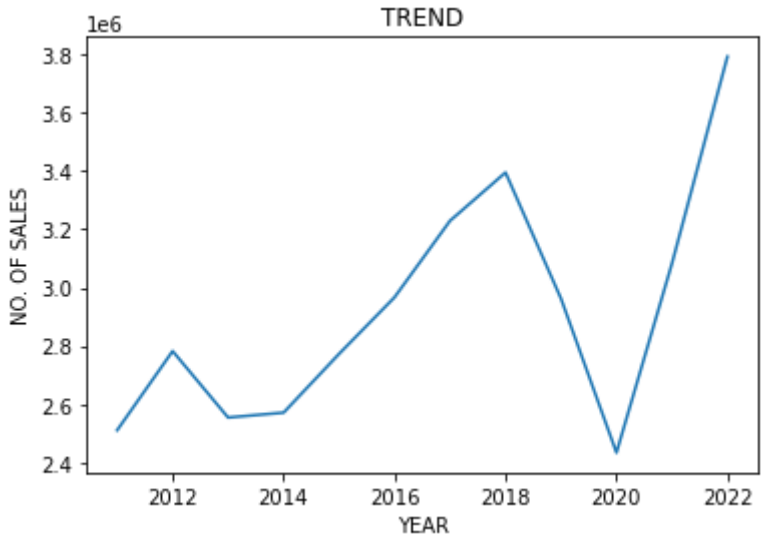
In [35]:
```
car_sales = pd.read_csv("car_sales.csv")
```

In [36]:
```
date = list(car_sales.iloc[:12,0])
```

In [37]:
```
sale = list(car_sales.iloc[1:12,1])
```

In [38]:
```
sales = [3792356]

for item in sale :
    sales.append(int(item))
```

In [39]:
```
plt.plot(date,sales)
plt.title('TREND')
plt.xlabel("YEAR")
plt.ylabel("NO. OF SALES")
plt.show()
```



LAST 5 MONTHS CAR SALES TREND

In [40]:
```
last_month_data = pd.read_csv("Car_sales_last_5_months.csv")
```

In [41]:
```
last_month_data
```

Out[41]:

	Make	Model Name	Nov-22	Dec-22	Jan-23	Feb-23	Mar-23	Apr-23	Unnamed: 8
0	Mahindra	Marazzo	201	171	164	171	490	0	NaN
1	Mahindra	Scorpio Classic	6455	7003	8715	6950	8788	9617	NaN
2	Mahindra	Thar	3987	3374	4410	5004	5008	5302	NaN
3	Mahindra	XUV300	5903	4850	5390	3809	5228	5062	NaN
4	Mahindra	eVerito	28	0	0	0	0	0	NaN

	Make	Model Name	Nov-22	Dec-22	Jan-23	Feb-23	Mar-23	Apr-23	Unnamed: 8
...	...	...	...	...	...	...	...	...	...
59	Maruti Nexa	XL6	2988	3364	2582	2108	1754	2860	NaN
60	Skoda	Superb	160	110	96	90	104	121	NaN
61	Skoda	Kodiaq	138	107	196	189	416	140	NaN
62	Skoda	Kushaq	2009	2186	2013	1783	2252	2162	NaN
63	Skoda	Slavia	2022	2257	1413	1274	1574	1586	NaN

64 rows × 9 columns

In [42]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
```

In [43]:
```
print(last_month_data.isnull().sum())
```

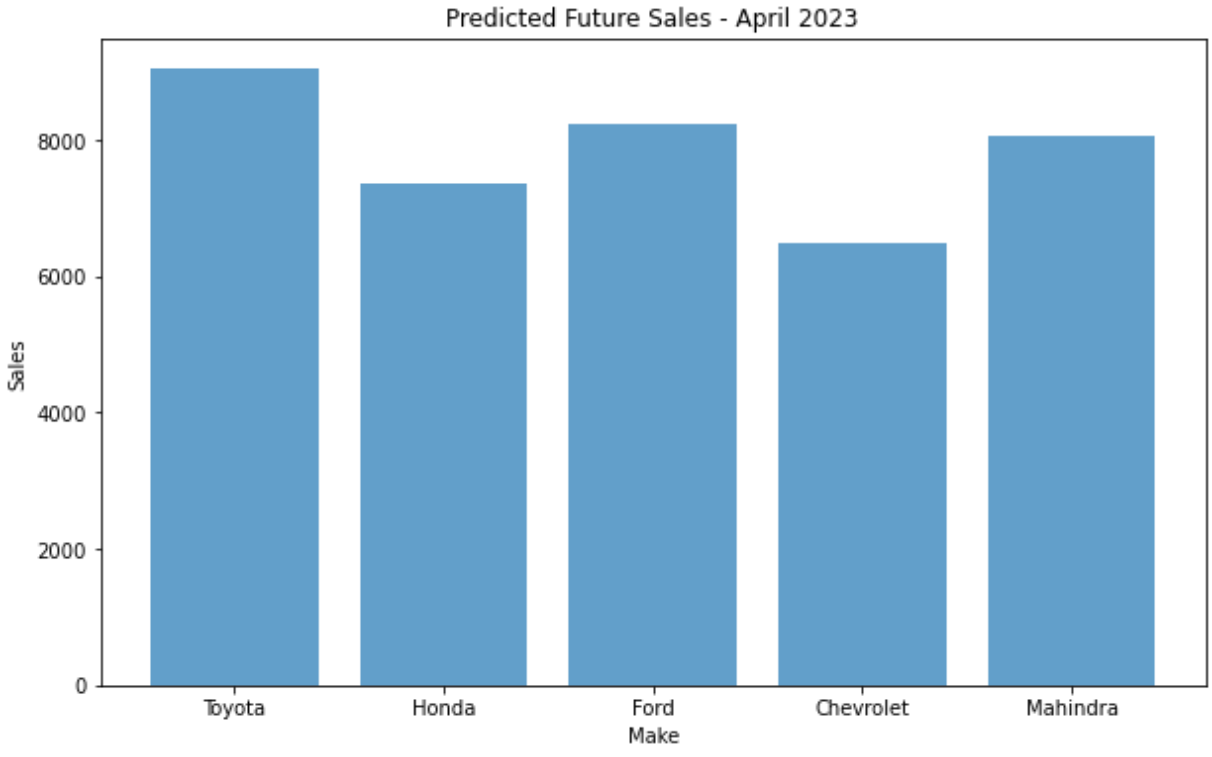
```
Make           0
Model Name     0
Nov-22         0
Dec-22         0
Jan-23         0
Feb-23         0
Mar-23         0
Apr-23         0
Unnamed: 8    64
dtype: int64
```

FUTURE SALES PREDICTIONS

In [44]:
```
features = ['Make', 'Nov-22', 'Dec-22', 'Jan-23', 'Feb-23', 'Mar-23']
target = 'Apr-23'  # Adjust the target column name based on your dataset

# Drop rows with missing values in the target column
last_month_data = last_month_data.dropna(subset=[target])

# Handle missing values in the feature columns
imputer = SimpleImputer(strategy='mean')
last_month_data[features[1:]] = imputer.fit_transform(last_month_data[features[1:]])

# Perform one-hot encoding on the 'Make' column
encoder = OneHotEncoder(sparse=False, handle_unknown='ignore')
encoded_features = encoder.fit_transform(last_month_data[['Make']])

# Combine the encoded features with the other numerical features
encoded_feature_names = encoder.get_feature_names(['Make'])
all_features = np.concatenate((encoded_feature_names, features[1:]))
X = np.concatenate((encoded_features, last_month_data[features[1:]]), axis=1)
y = last_month_data[target]
```

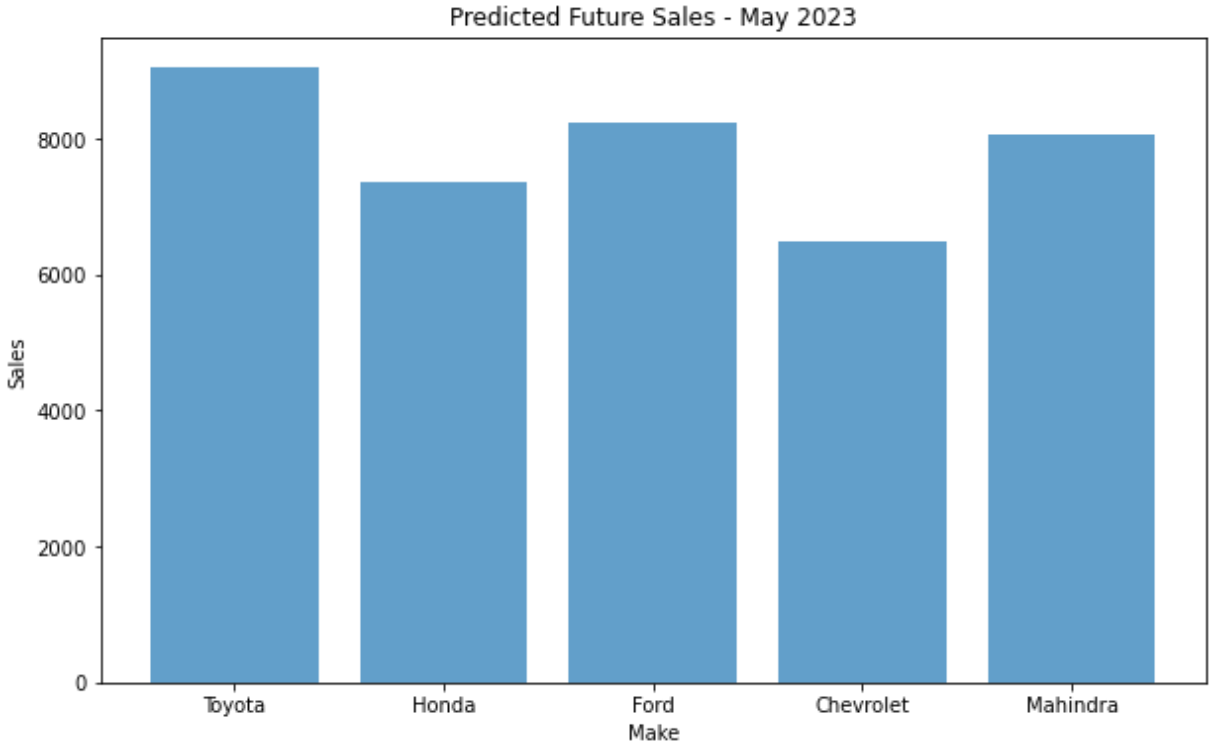
In [45]:
```
# Create the linear regression model
model = LinearRegression()
```

```
# Train the model
model.fit(X, y)

# Make predictions for future sales
future_data = pd.DataFrame({
    'Make': ['Toyota', 'Honda', 'Ford', 'Chevrolet', 'Mahindra'],  # Add the makes f
    'Nov-22': [10000, 8000, 9000, 7000, 8500],  # Add the corresponding sales values
    'Dec-22': [12000, 9000, 10000, 8000, 9500],  # Add the corresponding sales value
    'Jan-23': [11000, 8500, 9500, 7500, 9000],  # Add the corresponding sales values
    'Feb-23': [10500, 8200, 9200, 7200, 8800],  # Add the corresponding sales values
    'Mar-23': [11500, 8800, 9800, 7800, 9300]   # Add the corresponding sales values
})


# Handle missing values in the future data
future_data[features[1:]] = imputer.transform(future_data[features[1:]])

# Perform one-hot encoding on the future data
encoded_future_features = encoder.transform(future_data[['Make']])

# Make predictions for future sales
future_X = np.concatenate((encoded_future_features, future_data[features[1:]]), axis
future_predictions = model.predict(future_X)

# Create a DataFrame with the predicted sales
future_sales = pd.DataFrame({
    'Make': future_data['Make'],
    'Apr-23': future_predictions
})

# Visualize the predicted sales
plt.figure(figsize=(10, 6))
x_pos = np.arange(len(future_sales['Make']))
plt.bar(x_pos, future_sales['Apr-23'], align='center', alpha=0.7)
plt.xticks(x_pos, future_sales['Make'])
plt.xlabel('Make')
plt.ylabel('Sales')
plt.title('Predicted Future Sales - April 2023')
plt.show()
```



In [46]:

```
# Create a DataFrame with the predicted sales
future_sales = pd.DataFrame({
    'Make': future_data['Make'],
    'May-23': future_predictions
})

# Visualize the predicted sales
plt.figure(figsize=(10, 6))
x_pos = np.arange(len(future_sales['Make']))
plt.bar(x_pos, future_sales['May-23'], align='center', alpha=0.7)
plt.xticks(x_pos, future_sales['Make'])
plt.xlabel('Make')
plt.ylabel('Sales')
plt.title('Predicted Future Sales - May 2023')
plt.show()
```



In [47]:

```
# Calculate total sales volume for each model
sales_by_model = last_month_data.groupby('Model Name')['Nov-22', 'Dec-22', 'Jan-23',

# Sort the models based on total sales volume in descending order
top_selling_models = sales_by_model.sum(axis=1).sort_values(ascending=False)

# Display the top-selling models
print(top_selling_models)
```

```
Model Name
Balena          88994.0
Swift           79625.0
Wagon R         79552.0
Nexon           72174.0
Brezza          68897.0
                 ...
Hilux             281.0
EV6               262.0
Kona EV           255.0
Land Cruiser       88.0
eVerito            28.0
Length: 64, dtype: float64
```

```
<ipython-input-47-1c7b9c138beb>:2: FutureWarning: Indexing with multiple keys (impli
citly converted to a tuple of keys) will be deprecated, use a list instead.
  sales_by_model = last_month_data.groupby('Model Name')['Nov-22', 'Dec-22', 'Jan-2
3', 'Feb-23', 'Mar-23'].sum()
```

In [48]:

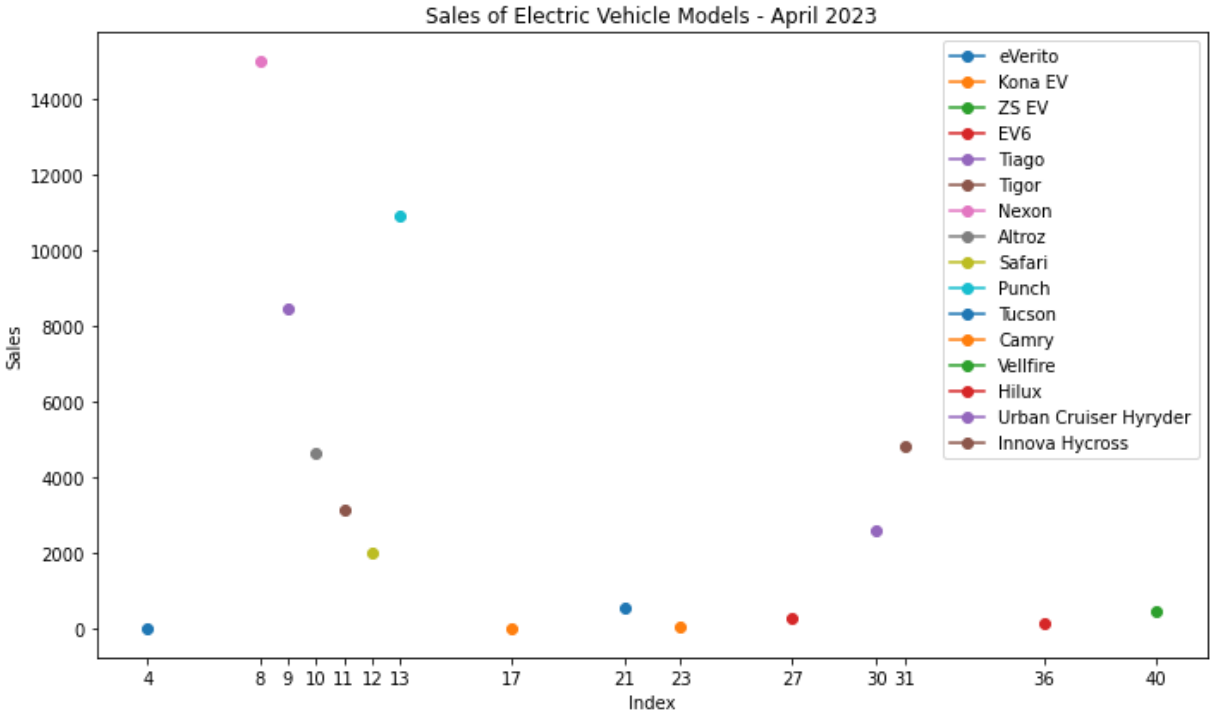
```
# Prepare the data
X = pd.get_dummies(last_month_data['Model Name'])  # Convert model names to numerica
y = last_month_data['Apr-23']  # Sales for April 2023

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat

# Train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', mse)

# Predict future sales
future_sales = model.predict(X)  # Provide the input features for future predictions

# Print the predicted sales for each model
for model_name, sales in zip(X.columns, future_sales):
    print('Model:', model_name, 'Predicted Sales:', sales)
```

```
Mean Squared Error: 28352310.460259166
Model: Alcazar Predicted Sales: 6243.916945279213
Model: Altroz Predicted Sales: 9617.000000000007
Model: Astor Predicted Sales: 5302.000000000002
Model: Aura Predicted Sales: 5061.999999999996
Model: Balena Predicted Sales: 0.0
Model: Bolero Neo Predicted Sales: 6243.916945279214
Model: Brezza Predicted Sales: 4757.000000000004
Model: Camry Predicted Sales: 2782.9999999999986
Model: Carens Predicted Sales: 15001.999999999996
Model: Carnival Predicted Sales: 6243.916945279214
Model: Celerio Predicted Sales: 4658.000000000004
Model: Ciaz Predicted Sales: 3154.0
Model: Creta Predicted Sales: 6243.916945279214
Model: Dzire Predicted Sales: 10934.000000000016
Model: EV6 Predicted Sales: 14185.999999999993
Model: Eeco Predicted Sales: 6472.000000000007
Model: Ertiga Predicted Sales: 6243.916945279212
Model: Fortuner     Predicted Sales: 1.8189894035458565e-12
Model: Glanza Predicted Sales: 4001.0000000000005
Model: Gloster Predicted Sales: 5085.000000000004
Model: Grand Vitara Predicted Sales: 2037.0000000000173
Model: Grand i10 Nios Predicted Sales: 549.9999999999955
Model: Harrier Predicted Sales: 10342.000000000004
Model: Hector Predicted Sales: 62.999999999991815
Model: Hilux Predicted Sales: 2577.9999999999886
Model: Ignis Predicted Sales: 6243.916945279208
Model: Innova Crysta Predicted Sales: 145.99999999999363
Model: Innova Hycross Predicted Sales: 269.00000000000455
Model: Kiger Predicted Sales: -9.094947017729282e-13
Model: Kodiaq Predicted Sales: 3652.9999999999973
Model: Kona EV Predicted Sales: 2615.9999999999977
Model: Kushaq Predicted Sales: 4836.999999999996
Model: Kwid Predicted Sales: 7212.9999999999945
Model: Land Cruiser Predicted Sales: -1.8189894035458565e-12
Model: Marazzo Predicted Sales: 9744.000000000004
Model: Nexon Predicted Sales: 6106.9999999999945
Model: Punch Predicted Sales: 6243.916945279209
```

```
Model: S-Presso Predicted Sales: 3103.0000000000027
Model: Safari Predicted Sales: 281.00000000000273
Model: Scorpio Classic Predicted Sales: 703.9999999999918
Model: Seltos Predicted Sales: 6243.916945279215
Model: Slavia Predicted Sales: 1520.0
Model: Sonet Predicted Sales: 31.00000000000182
Model: Superb Predicted Sales: 1480.9999999999955
Model: Swift Predicted Sales: 6243.916945279215
Model: Taigun Predicted Sales: 1161.9999999999882
Model: Thar Predicted Sales: 1082.000000000001
Model: Tiago Predicted Sales: 10132.000000000005
Model: Tigor Predicted Sales: 10503.999999999993
Model: Tiguan Predicted Sales: 18753.000000000007
Model: Triber Predicted Sales: 4889.999999999997
Model: Tucson Predicted Sales: 20879.000000000007
Model: Urban Cruiser Hyryder Predicted Sales: 6243.916945279218
Model: Vellfire     Predicted Sales: 5531.999999999998
Model: Venue Predicted Sales: 2561.9999999999995
Model: Verna Predicted Sales: 1017.0000000000082
Model: Virtus Predicted Sales: 6243.916945279216
Model: Wagon R Predicted Sales: 7742.0
Model: XL6 Predicted Sales: 6243.916945279216
Model: XUV300 Predicted Sales: 2860.0000000000005
Model: XUV700 7-Seater Predicted Sales: 120.99999999999545
Model: ZS EV Predicted Sales: 6243.916945279213
Model: eVerito Predicted Sales: 2161.9999999999995
Model: i20 Predicted Sales: 1586.000000000001
```

In [49]:

```
# Select the EV models and their corresponding sales data
ev_models = ['eVerito', 'Kona EV', 'ZS EV', 'EV6', 'Tiago', 'Tigor', 'Nexon', 'Altro
ev_sales = last_month_data[last_month_data['Model Name'].isin(ev_models)]

# Plotting the line chart
plt.figure(figsize=(10, 6))
for model in ev_models:
    model_sales = ev_sales[ev_sales['Model Name'] == model]
    plt.plot(model_sales.index, model_sales['Apr-23'], marker='o', label=model)
plt.xticks(ev_sales.index)
plt.xlabel('Index')
plt.ylabel('Sales')
plt.title('Sales of Electric Vehicle Models - April 2023')
plt.legend()
plt.tight_layout()
plt.show()
```



In [50]:

```
# Select the EV models and their corresponding sales data
ev_models = ['eVerito', 'Kona EV', 'ZS EV', 'EV6', 'Tiago', 'Tigor', 'Nexon', 'Altro
ev_sales = last_month_data[last_month_data['Model Name'].isin(ev_models)]

# Prepare the training data
months = ev_sales.columns[2:-1]
X = [[int(month.split('-')[1])] for month in months]  # Months as the input feature

future_months = ['May-23', 'Jun-23', 'Jul-23']
future_months_numeric = [int(month.split('-')[1]) for month in future_months]

# Train the linear regression model and predict future sales for each EV model
for model_name in ev_models:
    y = ev_sales[ev_sales['Model Name'] == model_name].values[0, 2:-1]

    # Check if the model has sufficient sales data
    if len(y) > 0:
        model = LinearRegression()
        model.fit(X, y.reshape(-1, 1))
        future_sales = model.predict([[month] for month in future_months_numeric])
        print(f"Model: {model_name}, Future Sales: {future_sales}")
    else:
        print(f"Model: {model_name}, No sufficient sales data.")
```

```
Model: eVerito, Future Sales: [[0.]
 [0.]
 [0.]]
Model: Kona EV, Future Sales: [[10.]
 [10.]
 [10.]]
Model: ZS EV, Future Sales: [[498.5]
 [498.5]
 [498.5]]
Model: EV6, Future Sales: [[38.]
 [38.]
 [38.]]
Model: Tiago, Future Sales: [[8076.25]
 [8076.25]
 [8076.25]]
Model: Tigor, Future Sales: [[3007.25]
 [3007.25]
```

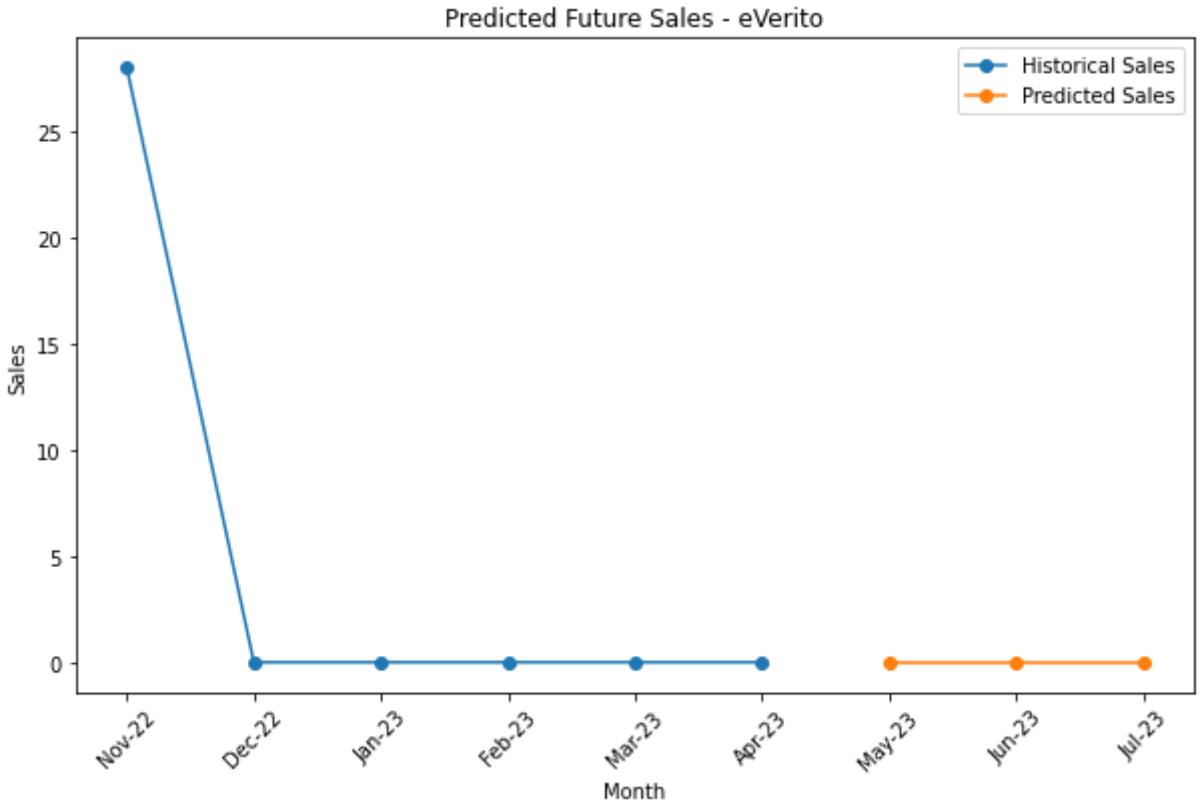
```
 [3007.25]]
Model: Nexon, Future Sales: [[14813.]
 [14813.]
 [14813.]]
Model: Altroz, Future Sales: [[4537.5]
 [4537.5]
 [4537.5]]
Model: Safari, Future Sales: [[1550.75]
 [1550.75]
 [1550.75]]
Model: Punch, Future Sales: [[11250.75]
 [11250.75]
 [11250.75]]
Model: Tucson, Future Sales: [[452.25]
 [452.25]
 [452.25]]
Model: Camry, Future Sales: [[71.75]
 [71.75]
 [71.75]]
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-50-6f43a48de368> in <module>
     12 # Train the linear regression model and predict future sales for each EV mod
el
     13 for model_name in ev_models:
---> 14     y = ev_sales[ev_sales['Model Name'] == model_name].values[0, 2:-1]
     15
     16     # Check if the model has sufficient sales data

IndexError: index 0 is out of bounds for axis 0 with size 0
```

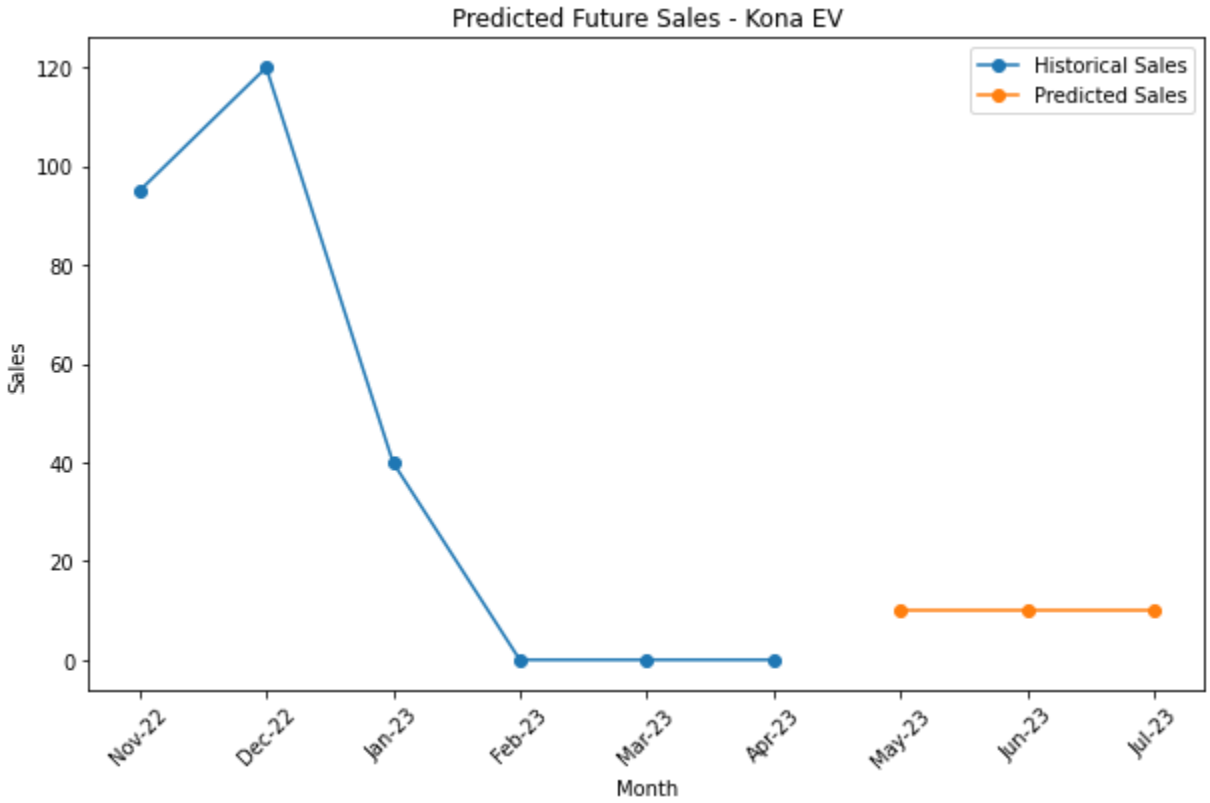
FUTURE SALES PREDICTIONS OF EV CARS IN INDIA

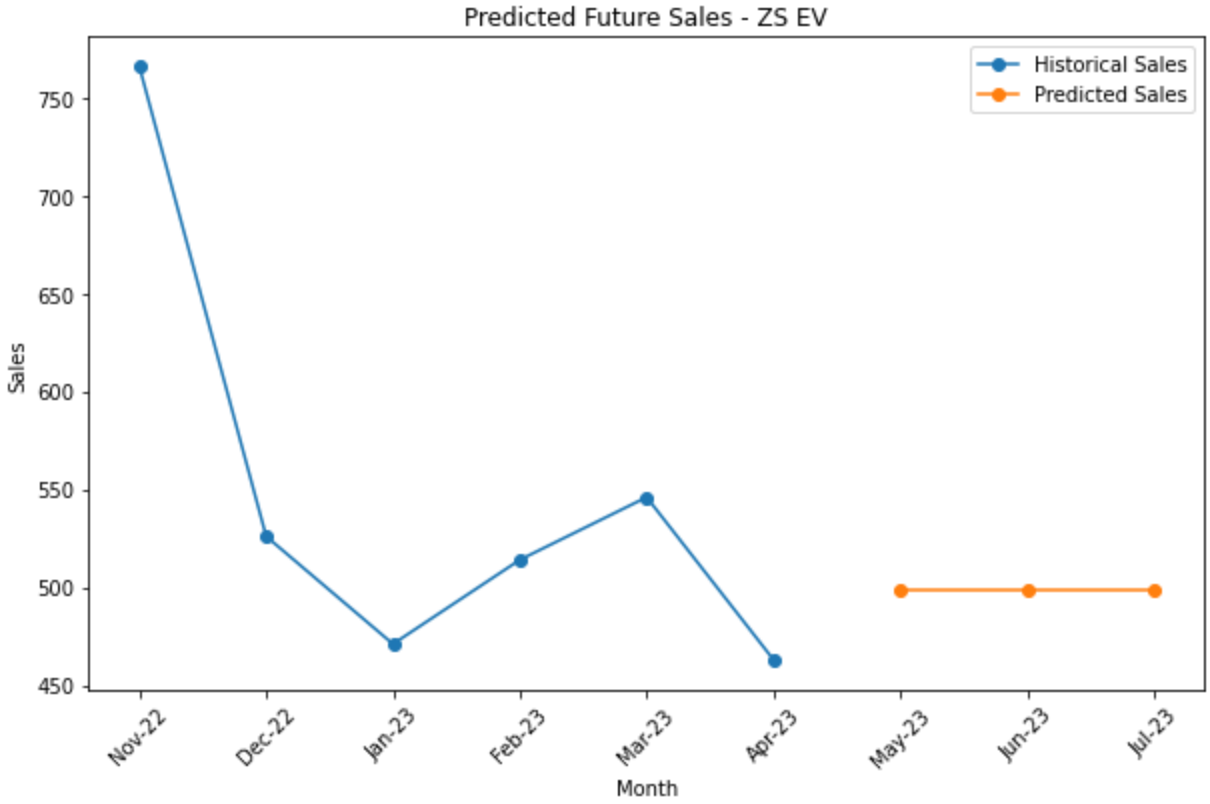
In [51]:

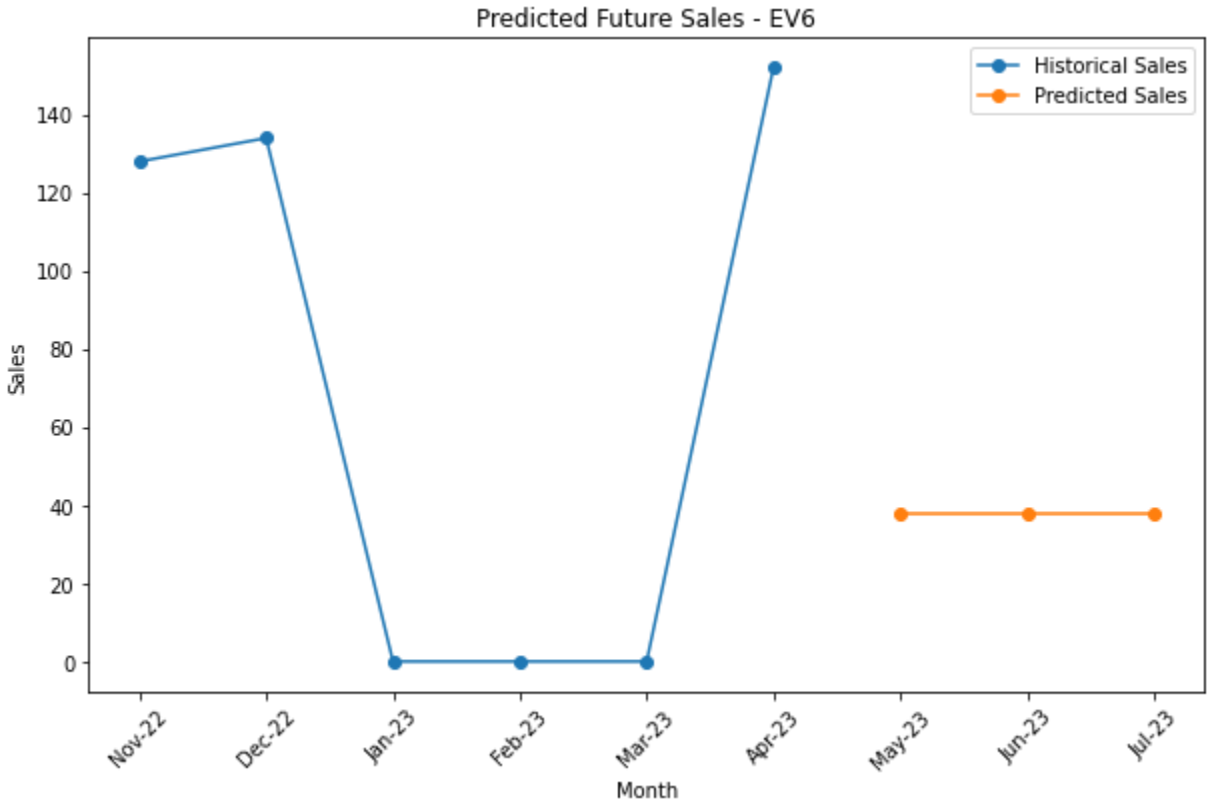
```
# Select the EV models and their corresponding sales data
ev_models = ['eVerito', 'Kona EV', 'ZS EV', 'EV6', 'Tiago', 'Tigor', 'Nexon', 'Altro
ev_sales = last_month_data[last_month_data['Model Name'].isin(ev_models)]

# Prepare the training data
months = ev_sales.columns[2:-1]
X = [[int(month.split('-')[1])] for month in months]  # Months as the input feature

future_months = ['May-23', 'Jun-23', 'Jul-23']
future_months_numeric = [int(month.split('-')[1]) for month in future_months]

# Train the linear regression model and predict future sales for each EV model
for model_name in ev_models:
    y = ev_sales[ev_sales['Model Name'] == model_name].values[0, 2:-1]

    # Check if the model has sufficient sales data
    if len(y) > 0:
        model = LinearRegression()
        model.fit(X, y.reshape(-1, 1))
        future_sales = model.predict([[month] for month in future_months_numeric])

         # Create a line chart for the predicted future sales
        plt.figure(figsize=(10, 6))
        plt.plot(months, y, marker='o', label='Historical Sales')
        plt.plot(future_months, future_sales.flatten(), marker='o', label='Predicted
        plt.xlabel('Month')
        plt.ylabel('Sales')
        plt.title(f'Predicted Future Sales - {model_name}')
        plt.legend()
        plt.xticks(rotation=45)
        plt.show()
```

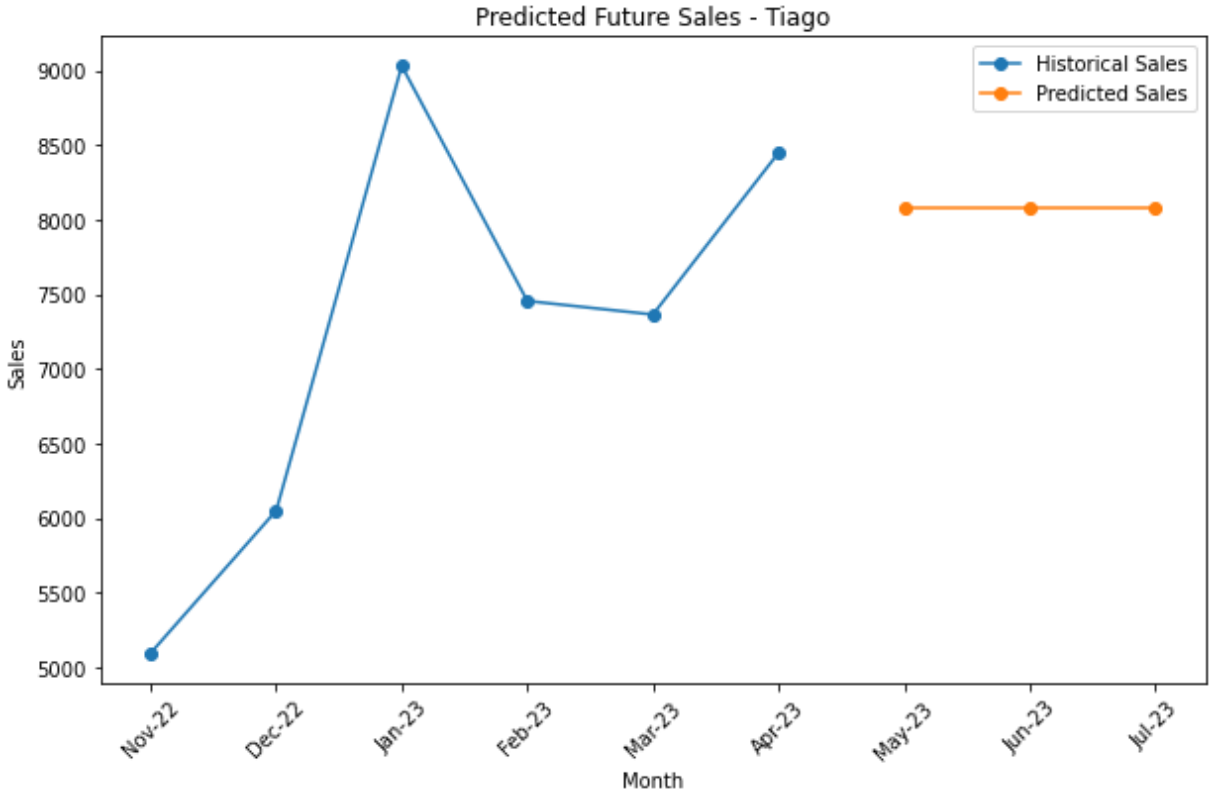
```
    else:
        print(f"Model: {model_name}, No sufficient sales data.")
```

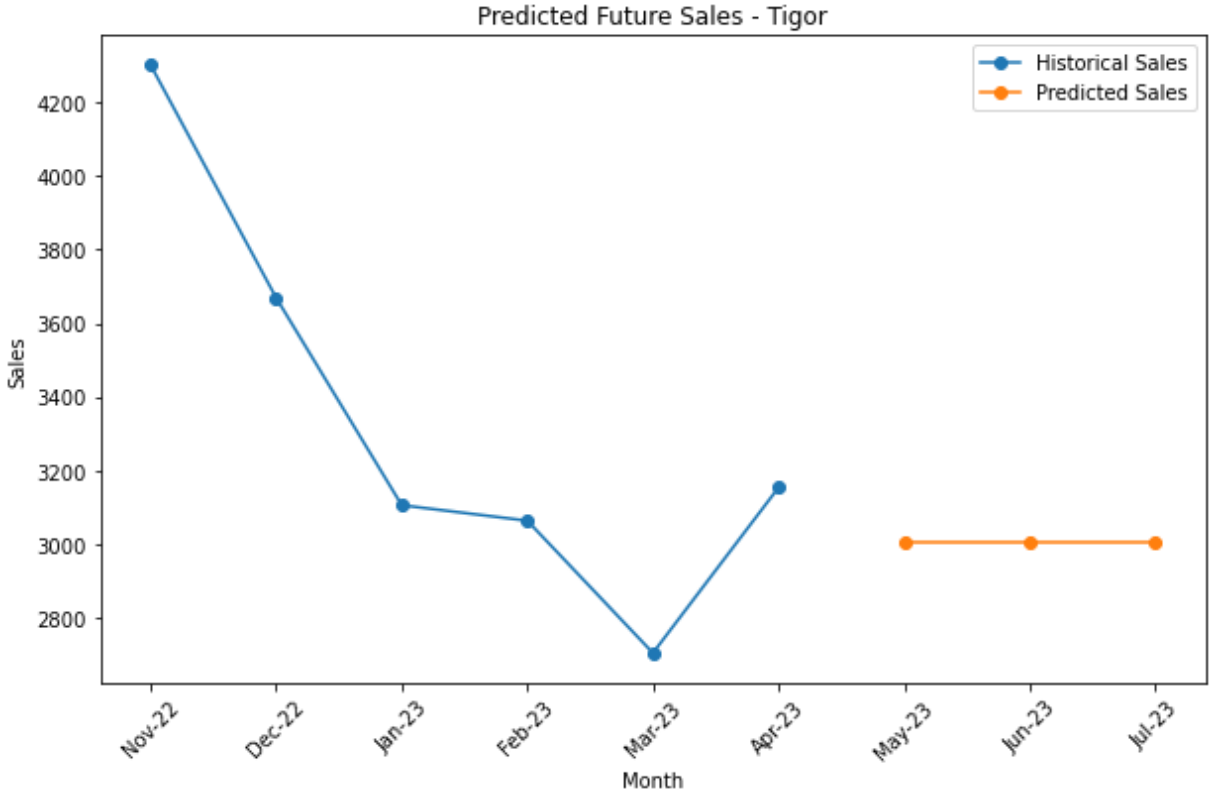


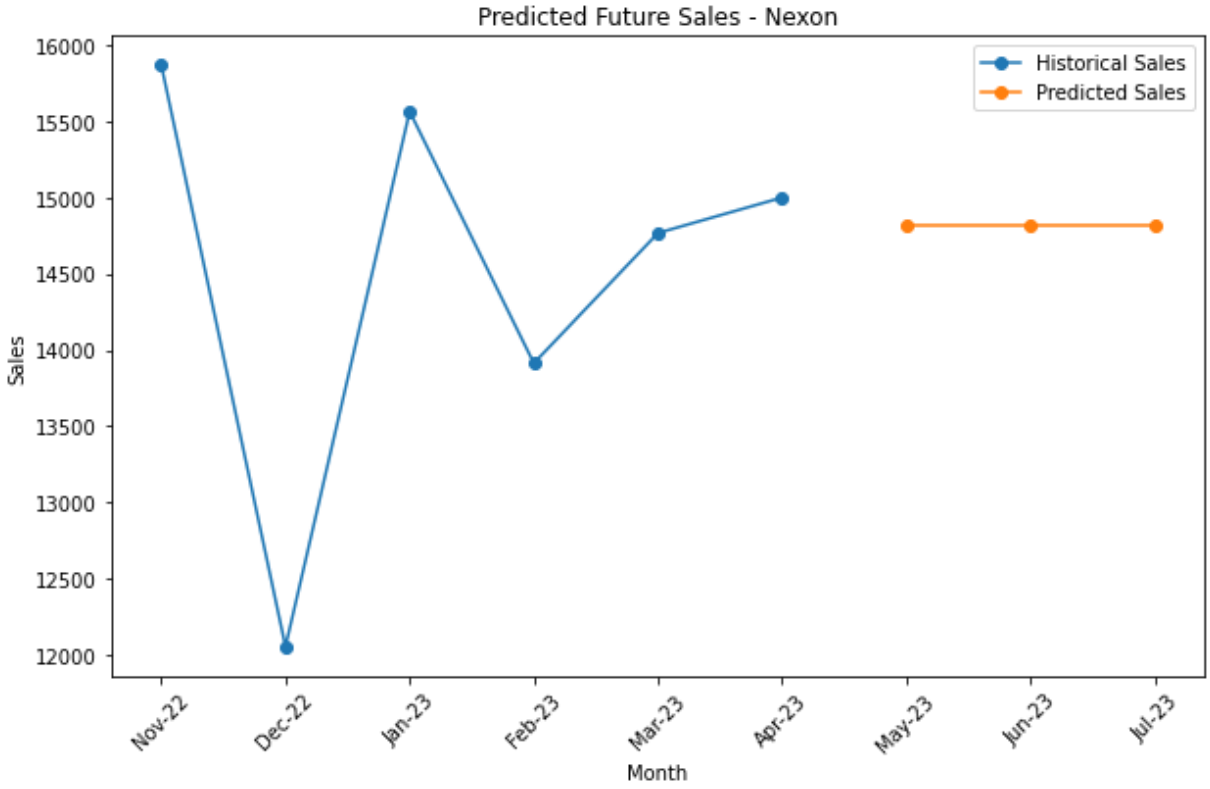


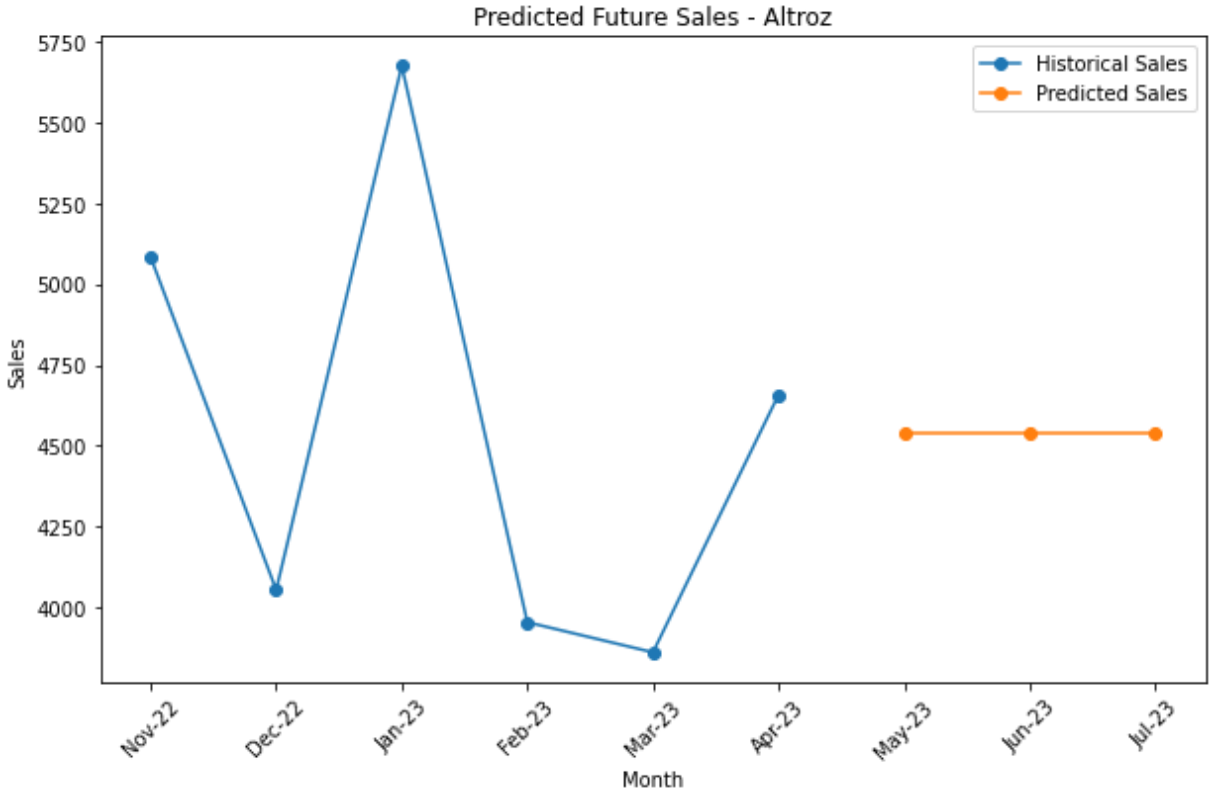

Predicted Future Sales - ZS EV
Historical Sales
Predicted Sales
Sales
750
700
650
600
550
500
450
Nov-22
Dec-22
Jan-23
Feb-23
Mar-23
Apr-23
May-23
Jun-23
Jul-23
Month

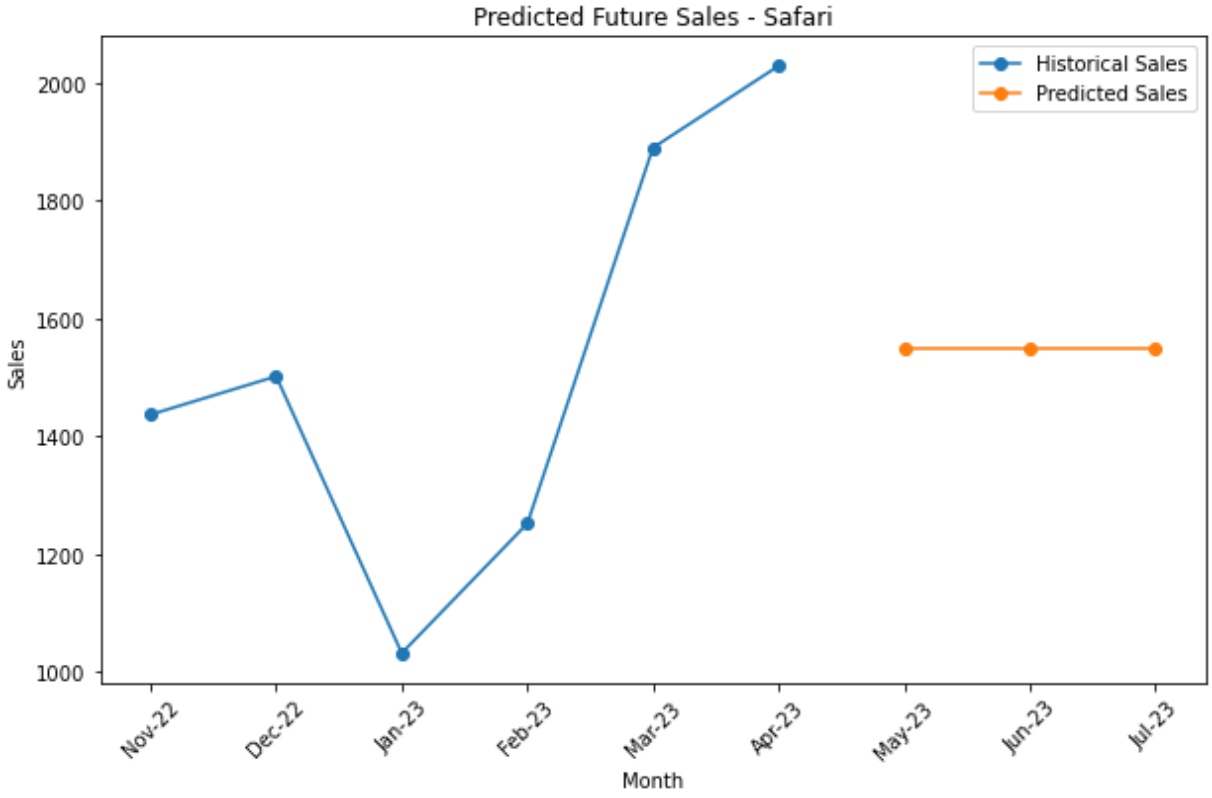

Predicted Future Sales - EV6
Historical Sales
Predicted Sales
Sales
140
120
100
80
60
40
20
0
Nov-22
Dec-22
Jan-23
Feb-23
Mar-23
Apr-23
May-23
Jun-23
Jul-23
Month

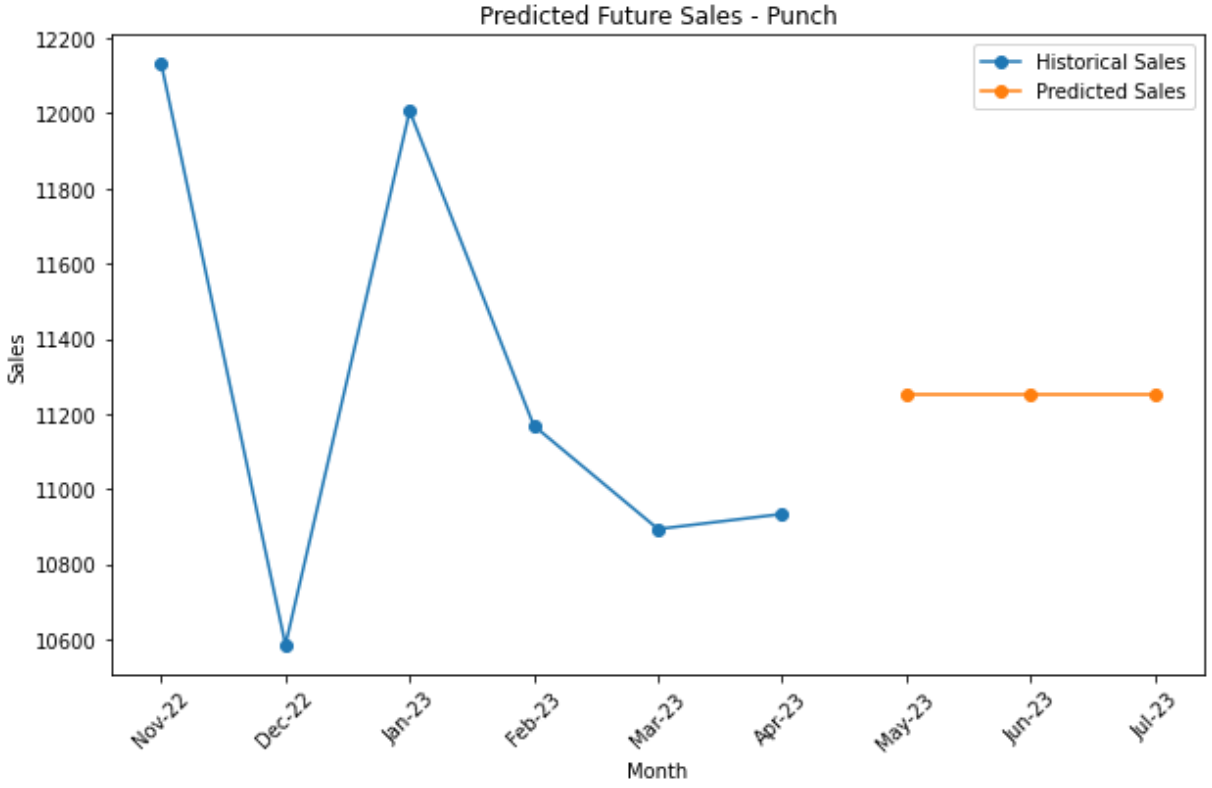

Predicted Future Sales - Tiago
Historical Sales
Predicted Sales
Sales
9000
8500
8000
7500
7000
6500
6000
5500
5000
Nov-22
Dec-22
Jan-23
Feb-23
Mar-23
Apr-23
May-23
Jun-23
Jul-23
Month

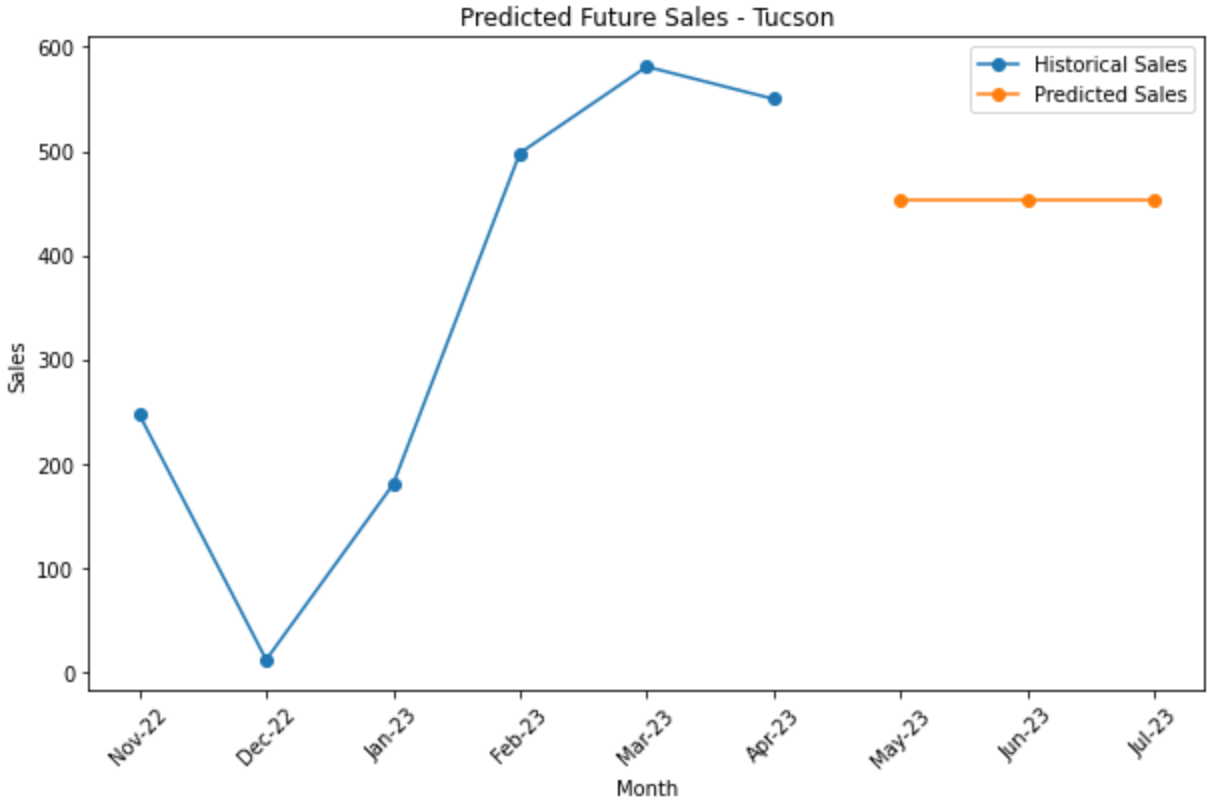

Predicted Future Sales - Tigor
Historical Sales
Predicted Sales
Sales
4200
4000
3800
3600
3400
3200
3000
2800
Nov-22
Dec-22
Jan-23
Feb-23
Mar-23
Apr-23
May-23
Jun-23
Jul-23
Month

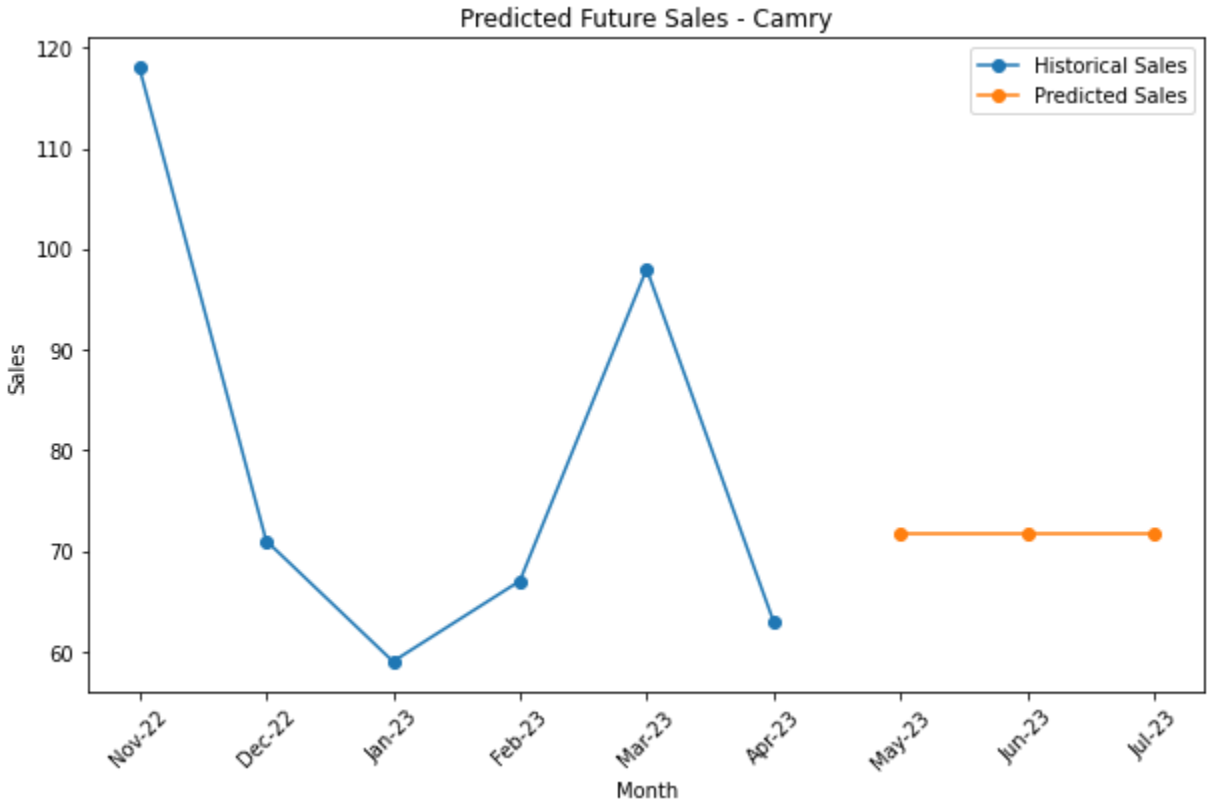

Predicted Future Sales - Nexon
Historical Sales
Predicted Sales
Sales
16000
15500
15000
14500
14000
13500
13000
12500
12000
Nov-22
Dec-22
Jan-23
Feb-23
Mar-23
Apr-23
May-23
Jun-23
Jul-23
Month


Predicted Future Sales - Altroz
Historical Sales
Predicted Sales
Sales
5750
5500
5250
5000
4750
4500
4250
4000
Nov-22
Dec-22
Jan-23
Feb-23
Mar-23
Apr-23
May-23
Jun-23
Jul-23
Month


Predicted Future Sales - Safari
Historical Sales
Predicted Sales
Sales
2000
1800
1600
1400
1200
1000
Nov-22
Dec-22
Jan-23
Feb-23
Mar-23
Apr-23
May-23
Jun-23
Jul-23
Month


Predicted Future Sales - Punch
Historical Sales
Predicted Sales
Sales
12200
12000
11800
11600
11400
11200
11000
10800
10600
Nov-22
Dec-22
Jan-23
Feb-23
Mar-23
Apr-23
May-23
Jun-23
Jul-23
Month






```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-51-d413b11a0357> in <module>
     12 # Train the linear regression model and predict future sales for each EV mod
el
     13 for model_name in ev_models:
---> 14     y = ev_sales[ev_sales['Model Name'] == model_name].values[0, 2:-1]
     15
     16     # Check if the model has sufficient sales data

IndexError: index 0 is out of bounds for axis 0 with size 0
```

Q6. WHAT SHOULD BE THE PRICE RANGE OF OUR VECHILE TO ATTRACT MOST NUMBER OF COSTUMERS ?

TARGET PRICE RANGE

In [52]:
```
dataset = pd.read_csv("Cars_age_Data.csv")
```

In [53]:
```
dataset
```

Out[53]:

	Company	Model	Buyer Gender	Buyer Age	Country	Color	New Car	Sale Price	Discount	Resell Price	Sp
0	Suzuki	Vitara	Female	51	India	Yellow	NaN	54806.14	0.2467	33858.32	2(
1	Honda	S2000	Male	30	India	Crimson	NaN	51826.30	0.3147	2989.28	1!
2	BMW	Z4	Female	54	India	Khaki	NaN	82929.14	0.5414	35049.16	1⁴
3	Toyota	Tacoma	Male	68	India	Puce	NaN	56928.66	0.0850	8236.15	1!
4	Ford	Festiva	Male	70	India	Yellow	NaN	77201.26	0.1642	32765.76	1;
...	...	...	...	...	...	...	...	...	...	...	
9995	Mazda	Tribute	Male	44	India	Pink	NaN	58580.65	0.1611	42640.82	2⁴
9996	GMC	Sierra 2500	Male	40	India	Puce	NaN	75229.74	0.2691	21115.58	1⁴
9997	Mercury	Mariner	Male	37	India	Blue	NaN	34755.44	0.2493	2731.25	1!
9998	Daewoo	Leganza	Female	21	India	Teal	1.0	98725.42	0.4654	18718.58	2(
9999	Toyota	Sequoia	Male	21	India	Pink	NaN	96769.78	0.5446	4545.95	2⁴

10000 rows × 11 columns

In [54]:
```
price_point = dataset[["Buyer Age","Sale Price"]]
```

In [55]:
```
price_point
```

Out[55]:

	Buyer Age	Sale Price
0	51	54806.14
1	30	51826.30
2	54	82929.14
3	68	56928.66
4	70	77201.26
...	...	...
9995	44	58580.65

	Buyer Age	Sale Price
9996	40	75229.74
9997	37	34755.44
9998	21	98725.42
9999	21	96769.78

10000 rows × 2 columns

USING K-MEANS

ELBOW METHOD

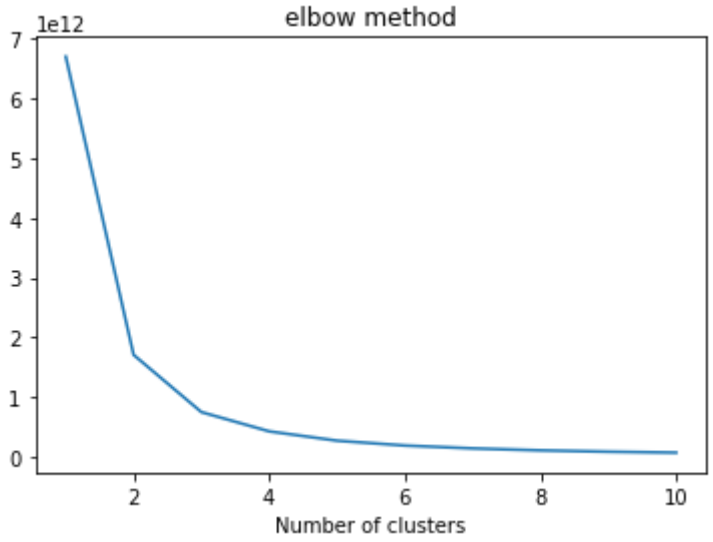
In [56]:
```
from sklearn.cluster import KMeans

wcss = []

for i in range (1,11):
    kmeans = KMeans(n_clusters=i,init = "k-means++")
    kmeans.fit(price_point)
    wcss.append(kmeans.inertia_)
```

In [57]:
```
a = np.arange(1,11)
print(a)
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

In [58]:
```
plt.plot(a,wcss)
plt.title("elbow method")
plt.xlabel("Number of clusters")
plt.ylable("WCSS")
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-58-9020d34d981b> in <module>
      2 plt.title("elbow method")
      3 plt.xlabel("Number of clusters")
----> 4 plt.ylable("WCSS")
      5 plt.show()

AttributeError: module 'matplotlib.pyplot' has no attribute 'ylable'
```



OPTIMAL NUMBER OF CLUSTER IS 3

In [59]:
```
kmeans = KMeans(n_clusters=3,init = "k-means++")
y_kmeans = kmeans.fit_predict(price_point)
```

In [60]:
```
y_kmeans
```

Out[60]:
```
array([0, 0, 1, ..., 2, 1, 1])
```

In [61]:
```
price_point = pd.DataFrame(price_point)
```

In [62]:
```
price_point
```

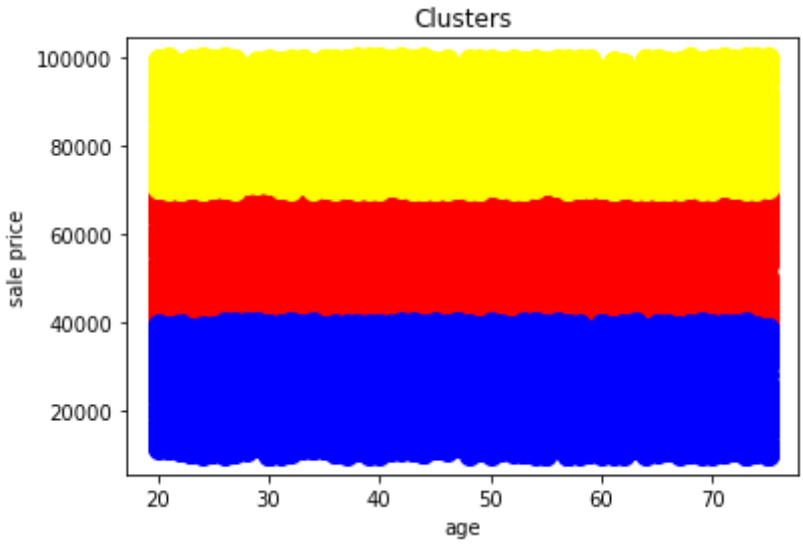
Out[62]:

	Buyer Age	Sale Price
0	51	54806.14
1	30	51826.30
2	54	82929.14
3	68	56928.66
4	70	77201.26
...	...	...
9995	44	58580.65
9996	40	75229.74
9997	37	34755.44
9998	21	98725.42
9999	21	96769.78

10000 rows × 2 columns

In [63]:
```
for i in range(0,10000):
    if y_kmeans[i]==0:
        plt.scatter(price_point.iloc[i,0], price_point.iloc[i,1], s = 100, c = "red"
```

```
for i in range(0,10000):
    if y_kmeans[i]==1:
        plt.scatter(price_point.iloc[i,0],price_point.iloc[i,1], s = 100, c = "yello
for i in range(0,10000) :
    if y_kmeans[i]==2:
        plt.scatter(price_point.iloc[i,0],price_point.iloc[i,1] , s = 100, c = "blue


#plt.scatter(x_train,y_train,c="red")
#plt.plot(x_train,a,c="blue")
plt.title('Clusters')
plt.xlabel('age')
plt.ylabel('sale price')
#plt.legend()
plt.show()
```



HENCE WE CAN CONCLUDE THAT MORE PEOPLE BUYS CAR UNDER PRICE RANGE OF 65,000 dollars

HENCE WE WILL TRY TO INTRODUCE OUR CAR UNDER 65,000 DOLLAR IN INDIAN MARKET

Q7. WHICH STATE WE SHOULD TARGET FIRST ?

TARGET STATES OF INDIA

In [64]:
```
data = pd.read_csv("data_states.csv")
```

In [65]:
```
data
```

Out[65]:

	State_Name	Two_Wheeler	Three_Wheeler	Four_Wheeler	Goods_Vehicles	Public_Service_Vehi
0	Andaman and Nicobar Island	1	30	81	0	

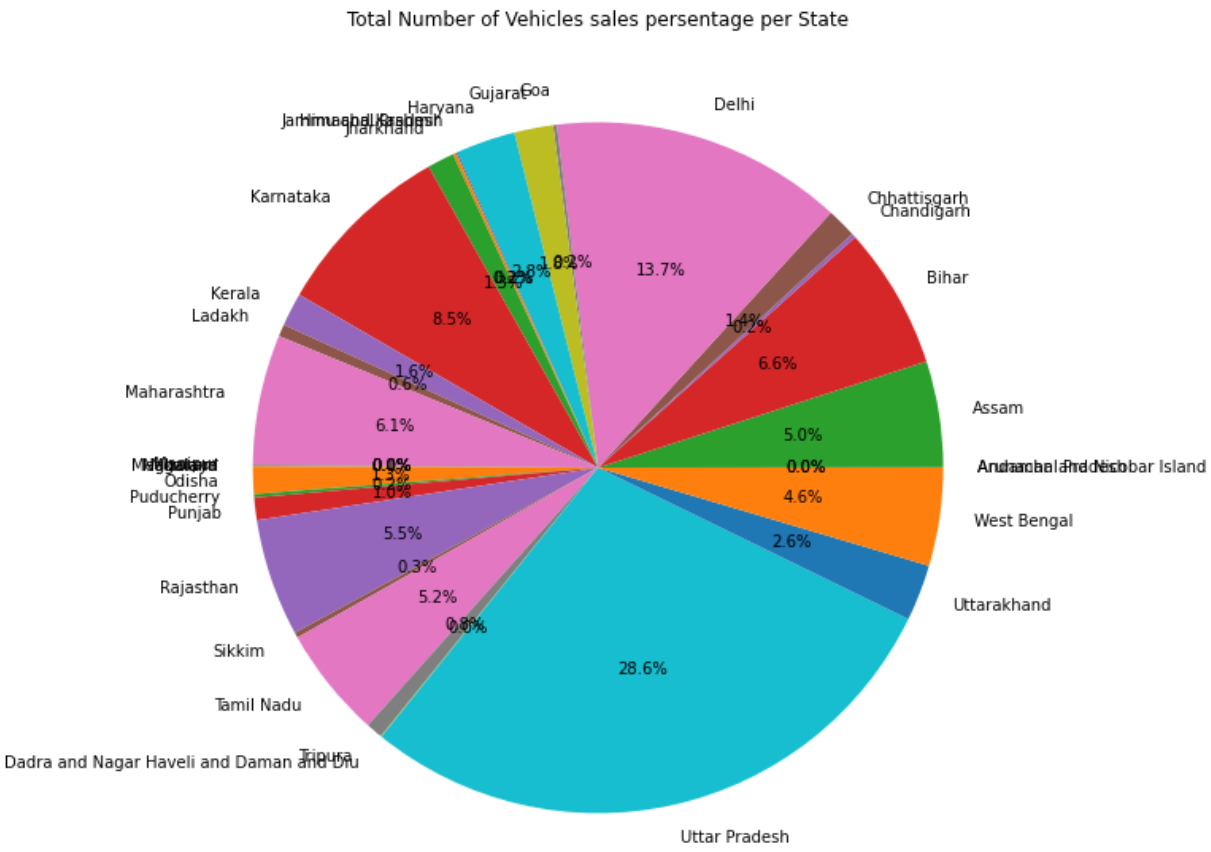
	State_Name	Two_Wheeler	Three_Wheeler	Four_Wheeler	Goods_Vehicles	Public_Service_Vehi
1	Arunachal Pradesh	14	0	5	0	
2	Assam	721	47041	161	7	
3	Bihar	5003	59079	114	11	
4	Chandigarh	298	1410	182	0	
5	Chhattisgarh	6424	5341	117	1077	
6	Delhi	14730	112831	3051	49	
7	Goa	1314	28	289	13	
8	Gujarat	13662	1869	1309	28	2
9	Haryana	7777	18595	186	122	
10	Himachal Pradesh	368	167	37	7	
11	Jammu and Kashmir	1417	43	10	6	
12	Jharkhand	2961	8986	139	24	
13	Karnataka	56737	16478	7212	153	
14	Kerala	10299	2115	2524	43	
15	Ladakh	12	0	5484	0	
16	Maharashtra	51149	6155	2	30	8
17	Manipur	86	443	9	1	
18	Meghalaya	16	6	3	3	
19	Mizoram	9	1	4	2	
20	Nagaland	44	0	121	3	
21	Odisha	10329	1808	75	21	
22	Puducherry	1429	32	124	9	
23	Punjab	6408	2878	798	35	
24	Rajasthan	23446	29631	12	25	
25	Sikkim	1	0	2414	1	
26	Tamil Nadu	44302	4470	13	1281	
27	Tripura	67	7510	14	1	
28	Dadra and Nagar Haveli and Daman and Diu	69	36	153	2	
29	Uttar Pradesh	18295	257159	368	53	3
30	Uttarakhand	2614	22096	709	1	
31	West Bengal	2540	40948	615	28	

SALES OF EV IN DIFFERENT STATES IN INDIA

In [66]:
```
data['Total_Vehicles'] = data.iloc[:, 1:-1].sum(axis=1)
total_vehicles_per_state = data[['State_Name', 'Grand_Total']]
print(total_vehicles_per_state)
state_data = data[['State_Name', 'Total_Vehicles']]
plt.figure(figsize=(10,10))
plt.pie(state_data['Total_Vehicles'] , labels=state_data['State_Name'] ,autopct='%1.
plt.title('Total Number of Vehicles sales persentage per State')
```

	State_Name	Grand_Total
0	Andaman and Nicobar Island	159
1	Arunachal Pradesh	20
2	Assam	47947
3	Bihar	64241
4	Chandigarh	1931
5	Chhattisgarh	13428
6	Delhi	132302
7	Goa	1686
8	Gujarat	17593
9	Haryana	26780
10	Himachal Pradesh	711
11	Jammu and Kashmir	1527
12	Jharkhand	12171
13	Karnataka	82046
14	Kerala	15022
15	Ladakh	5496
16	Maharashtra	58815
17	Manipur	540
18	Meghalaya	28
19	Mizoram	20
20	Nagaland	171
21	Odisha	12282
22	Puducherry	1614
23	Punjab	10142
24	Rajasthan	53141
25	Sikkim	2425
26	Tamil Nadu	50296
27	Tripura	7593
28	Dadra and Nagar Haveli and Daman and Diu	277
29	Uttar Pradesh	276217
30	Uttarakhand	25451
31	West Bengal	44291

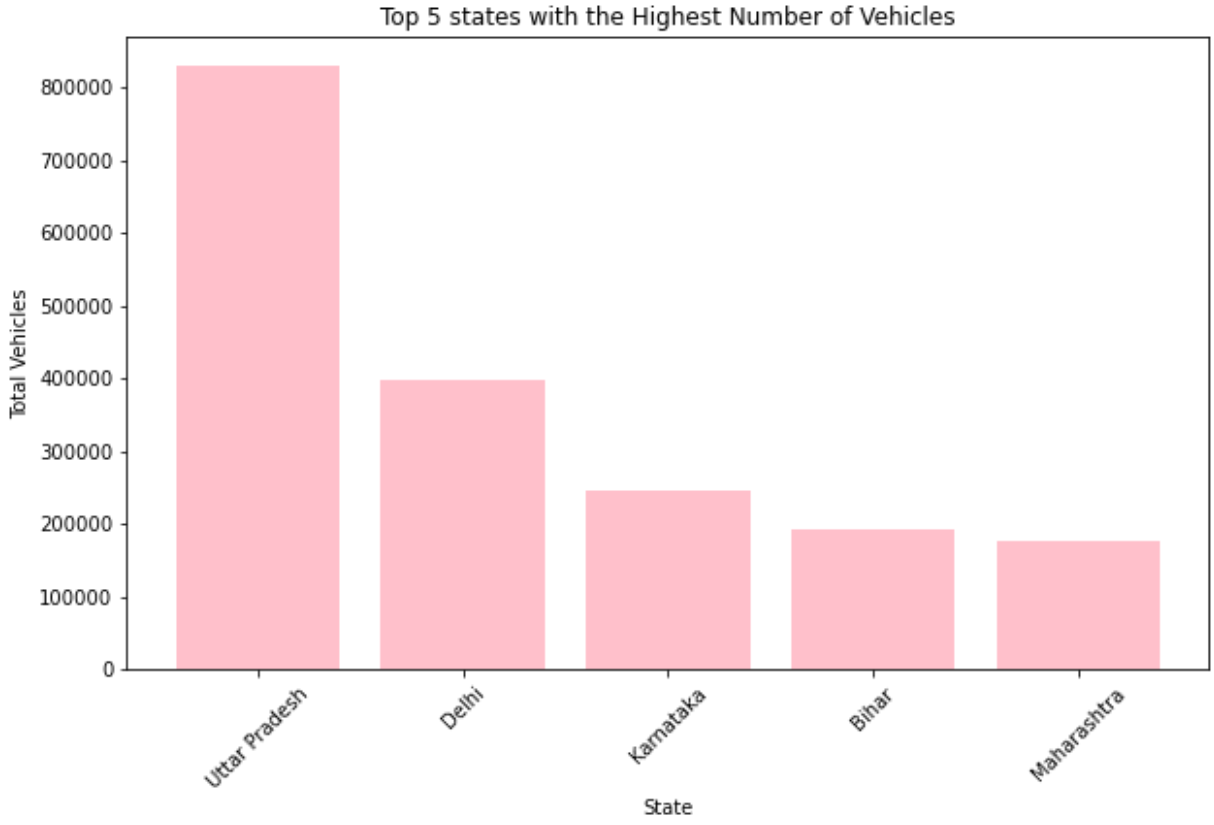
Out[66]: Text(0.5, 1.0, 'Total Number of Vehicles sales persentage per State')



TOP 5 STATES WITH HEIGHEST NUMBER OF SALES

In [67]:

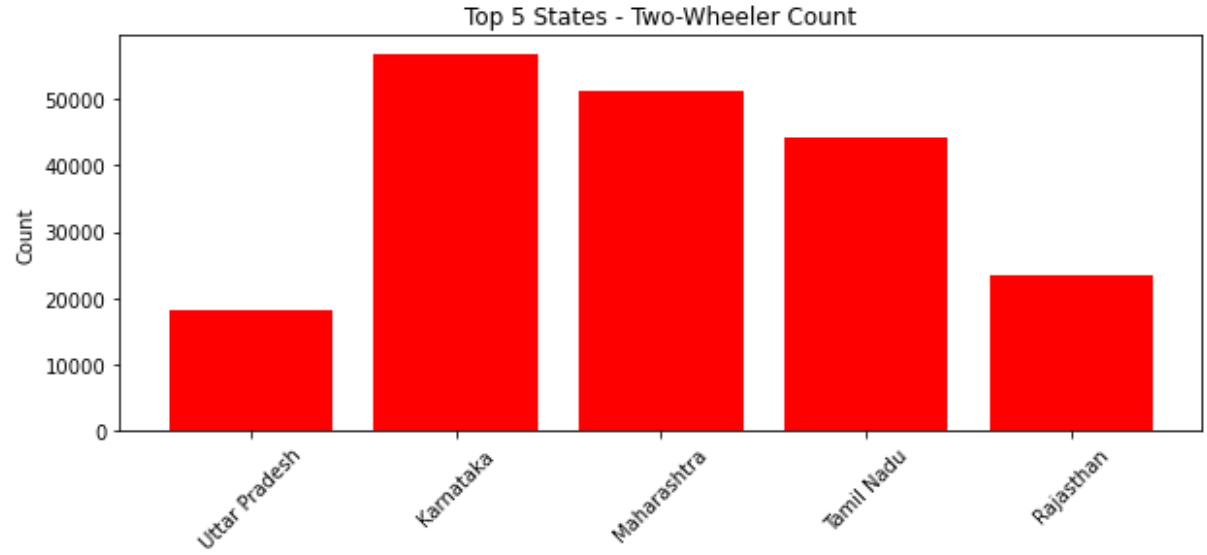
```
data['Total_Vehicles'] = data.iloc[:, 1:].sum(axis=1)

# Get the top 5 states with the highest number of vehicles
top_states = data.nlargest(5, 'Total_Vehicles')

# Plot the bar graph
plt.figure(figsize=(10, 6))
plt.bar(top_states['State_Name'], top_states['Total_Vehicles'],color = "pink")
plt.xlabel('State')
plt.ylabel('Total Vehicles')
plt.title('Top 5 states with the Highest Number of Vehicles')
plt.xticks(rotation=45)
plt.show()
```

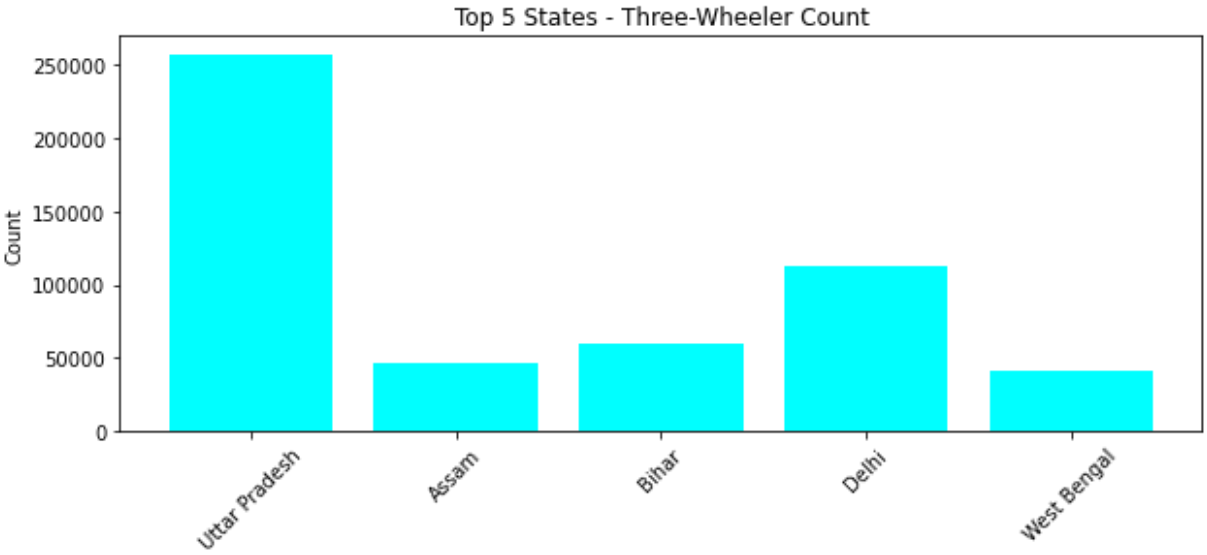


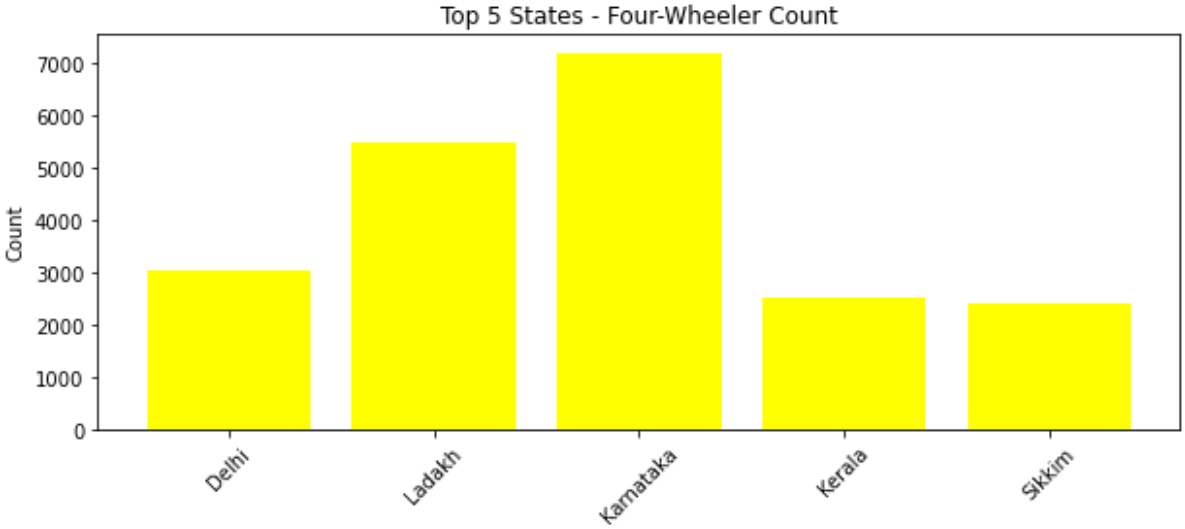
States with highest number of Two, Three and Four wheeler vehicles sales

In [68]:

```
two_wheeler_data = {
    'State_Name': ['Uttar Pradesh', 'Karnataka', 'Maharashtra', 'Tamil Nadu', 'Rajas
    'Two_Wheeler': [18295, 56737, 51149, 44302, 23446]}

three_wheeler_data = {
    'State_Name': ['Uttar Pradesh', 'Assam', 'Bihar', 'Delhi', 'West Bengal'],
    'Three_Wheeler': [257159, 47041, 59079, 112831, 40948]}

four_wheeler_data = {
    'State_Name': ['Delhi', 'Ladakh','Karnataka', 'Kerala', 'Sikkim'],
    'Four_Wheeler': [3051, 5484, 7212, 2524, 2414]}

two_wheeler_df = pd.DataFrame(two_wheeler_data)
three_wheeler_df = pd.DataFrame(three_wheeler_data)
four_wheeler_df = pd.DataFrame(four_wheeler_data)

fig, axs = plt.subplots(3, 1, figsize=(10, 15))

axs[0].bar(two_wheeler_df['State_Name'], two_wheeler_df['Two_Wheeler'],color = "red"
axs[0].set_ylabel('Count')
axs[0].set_title('Top 5 States - Two-Wheeler Count')
axs[0].tick_params(axis='x', rotation=45)

axs[1].bar(three_wheeler_df['State_Name'], three_wheeler_df['Three_Wheeler'],color =
axs[1].set_ylabel('Count')
axs[1].set_title('Top 5 States - Three-Wheeler Count')
axs[1].tick_params(axis='x', rotation=45)

axs[2].bar(four_wheeler_df['State_Name'], four_wheeler_df['Four_Wheeler'],color = "y
axs[2].set_ylabel('Count')
axs[2].set_title('Top 5 States - Four-Wheeler Count')
axs[2].tick_params(axis='x', rotation=45)

plt.subplots_adjust(hspace=0.5)
plt.show()
```



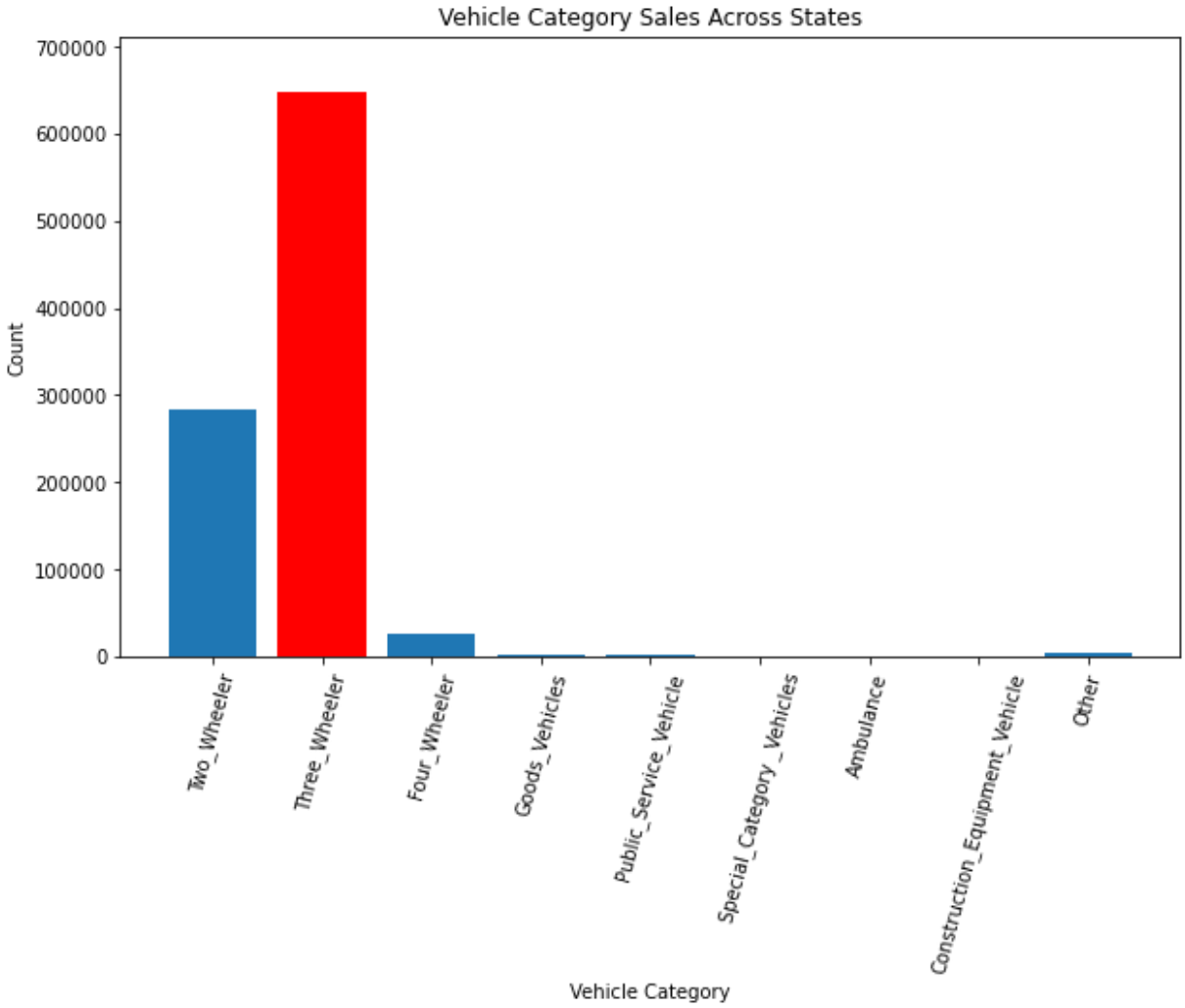




Q8. WHICH IS THE HIGHEST SELLING VECHILE CATEGORY IN EV ?

The highest selling vehicle category in all States combined

In [69]:
```
vehicle_categories = ['Two_Wheeler', 'Three_Wheeler', 'Four_Wheeler', 'Goods_Vehicle
                      'Special_Category _Vehicles', 'Ambulance', 'Construction_Equip
category_counts = data[vehicle_categories].sum()
highest_selling_category = category_counts.idxmax()
plt.figure(figsize=(10, 6))
plt.bar(vehicle_categories, category_counts)
plt.xlabel('Vehicle Category')
plt.ylabel('Count')
plt.title('Vehicle Category Sales Across States')
plt.xticks(rotation=75)
plt.ylim(top=max(category_counts) * 1.1)
max_index = vehicle_categories.index(highest_selling_category)
plt.bar(vehicle_categories[max_index], category_counts[max_index], color='red')
plt.show()

print("The highest selling vehicle category in all states combined is:", highest_sel
```




```
The highest selling vehicle category in all states combined is: Three_Wheeler
```

HENCE WE CAN CONCLUDE THE ELECTRIC SCOTERS AND E-RICKSHAW IS MORE DOMINANT IN INDIAN MARKET

SO IF WE WANT TO INTRODUCE OUR E-CAR THEN WE HAVE TO ATTRACT MORE COSTUMERS TOWARDS ELECTRIC CAR AS IT HAS MORE SCOPE TO GROW

Q9. WHERE IS THE MOST NUMBER OF CHARGING STATIONS ARE AVAILABLE

EV CHARGING STATIONS

In [70]:
```
dataset = pd.read_csv("CHARGING_STATION.csv")
```

In [71]:
```
dataset
```

Out[71]:

	Sl. No	Category	Expressways/Highways	EV Charging Stations Sanctioned
0	1	Expressways	Mumbai - Pune	10
1	2	Expressways	Ahmadabad - Vadodara	10
2	3	Expressways	Delhi Agra Yamuna	20
3	4	Expressways	Bengaluru Mysore	14
4	5	Expressways	Bangaluru-Chennai	30
5	6	Expressways	Surat-Mumbai	30
6	7	Expressways	Agra-Lucknow	40
7	8	Expressways	Eastern Peripheral (A)	14
8	9	Expressways	Hyderabad ORR	16
9	1	Highways	Delhi - Srinagar	80
10	2	Highways	Delhi – Kolkata	160
11	3	Highways	Agra - Nagpur	80
12	4	Highways	Meerut to GangotriDham	44
13	5	Highways	Mumbai - Delhi	124
14	6	Highways	Mumbai-Panaji	60
15	7	Highways	Mumbai - Nagpur	70
16	8	Highways	Mumbai - Bengaluru	100
17	9	Highways	Kolkata - Bhubaneswar	44
18	10	Highways	Kolkata - Nagpur	120
19	11	Highways	Kolkata - Gangtok	76
20	12	Highways	Chennai-Bhubaneswar	120
21	13	Highways	Chennai - Trivendram	74
22	14	Highways	Chennai-Ballary	62
23	15	Highways	Chennai - Nagpur	114
24	16	Highways	Mangaldai - Wakro	64
25	Total	Total	Total	1576

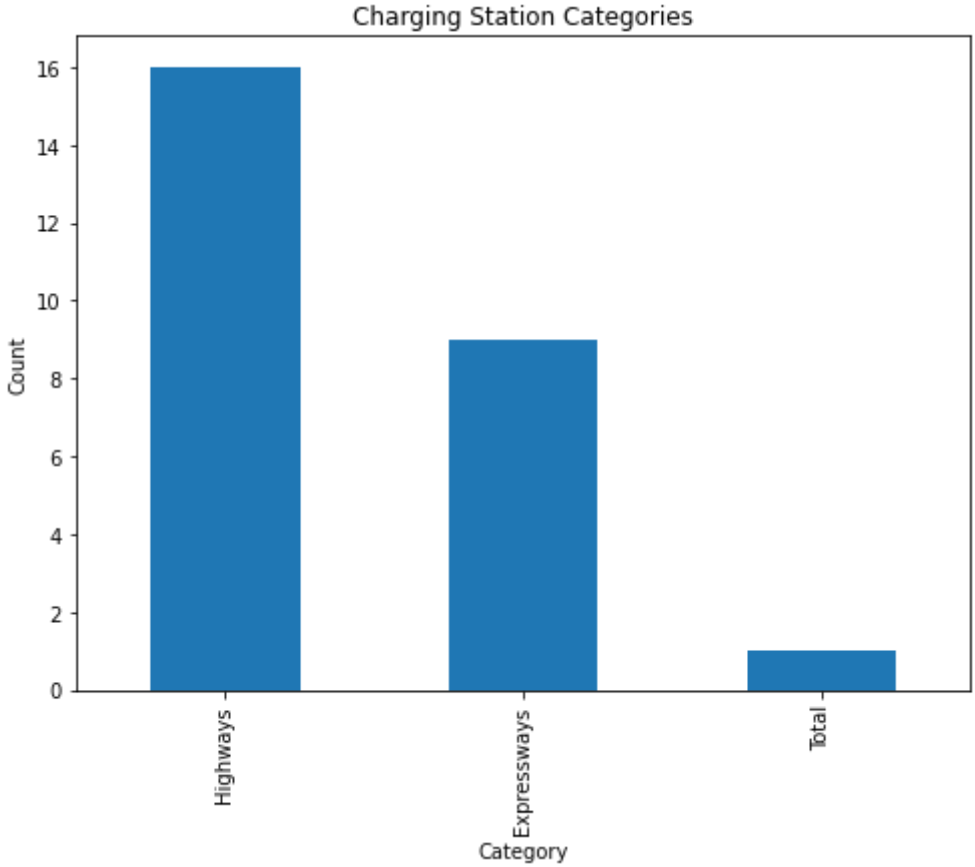
Count the number of charging stations in each category

In [72]:
```
category_counts = dataset['Category'].value_counts()
print(category_counts)
```

```
Highways       16
Expressways     9
Total           1
Name: Category, dtype: int64
```

Plot a bar chart for charging station categories

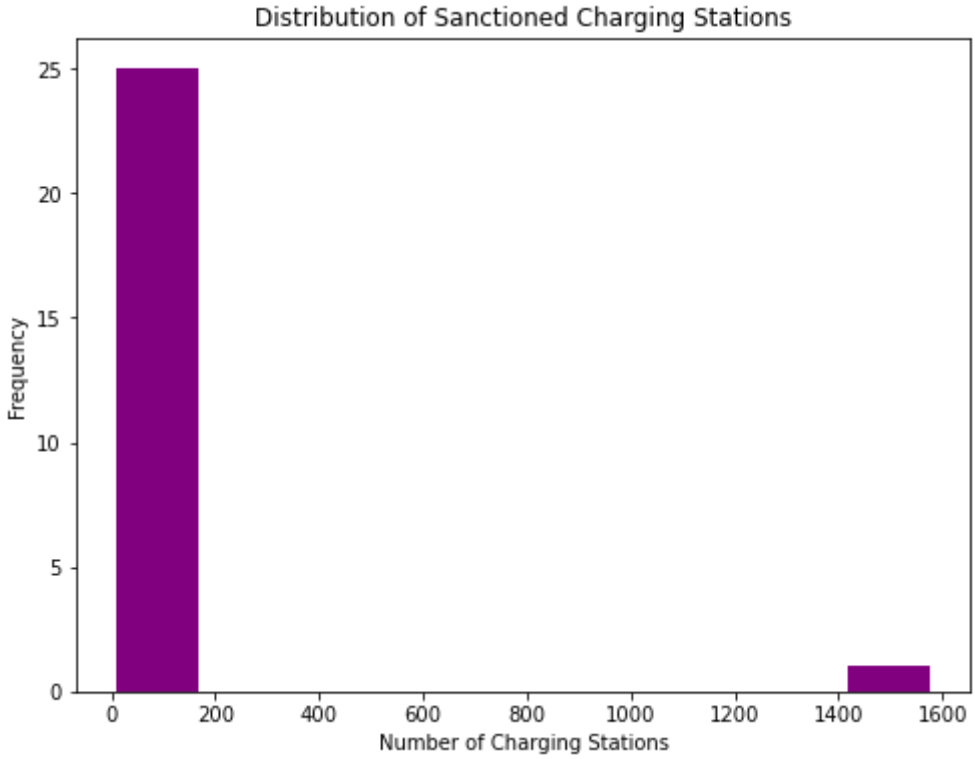
In [73]:
```
plt.figure(figsize=(8, 6))
category_counts.plot(kind='bar')
plt.title('Charging Station Categories')
plt.xlabel('Category')
plt.ylabel('Count')
plt.savefig('charging_station_categories.png')  # Save the chart as an image file
plt.show()
```



Q10. HOW MANY CHARGING STATIONS ARE SANCTIONED ?

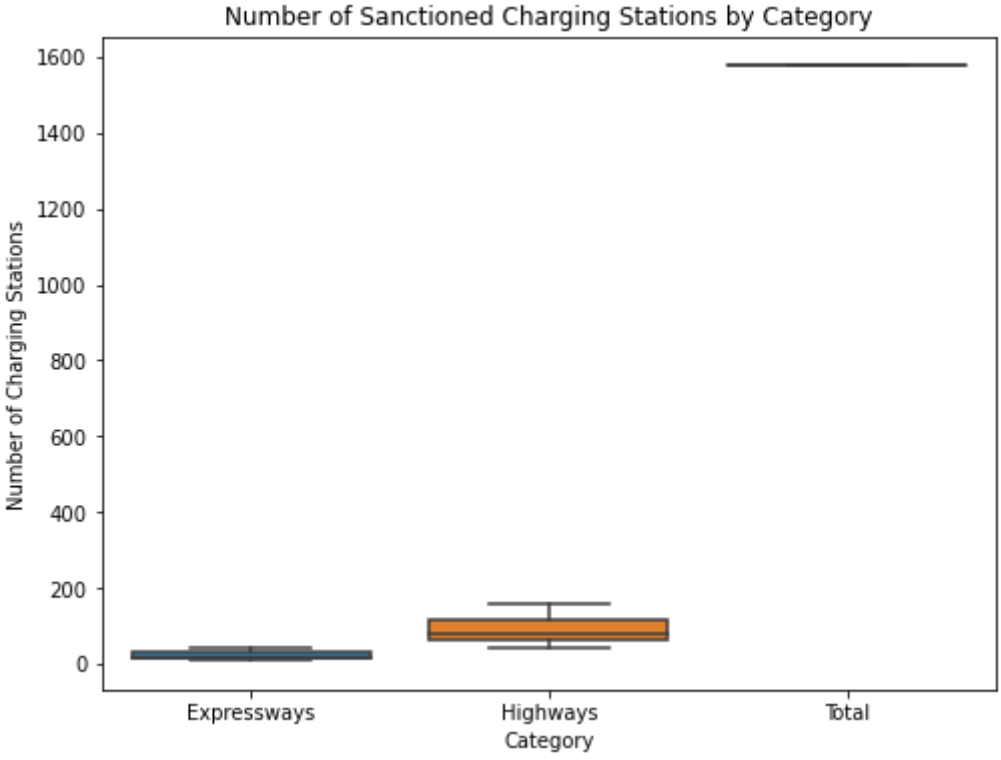
Plot a histogram for the number of sanctioned charging stations

In [74]:
```
plt.figure(figsize=(8, 6))
plt.hist(dataset['EV Charging Stations Sanctioned'], color = "purple",bins=10)
plt.title('Distribution of Sanctioned Charging Stations')
plt.xlabel('Number of Charging Stations')
plt.ylabel('Frequency')
plt.savefig('charging_station_distribution.png')  # Save the chart as an image file
plt.show()
```



Box plot of the number of sanctioned charging stations by category

In [75]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Category', y='EV Charging Stations Sanctioned', data=dataset)
plt.title('Number of Sanctioned Charging Stations by Category')
plt.xlabel('Category')
plt.ylabel('Number of Charging Stations')
plt.show()
```



CONCLUSION

BY THIS EXPLORATORY DATA ANALYSIS WE CAN CONCLUDE SEVERAL POINTS

1. MORE THAN 50% OF INDIAN CAR MARKET IS CAPTURED BY TOP 7 COMPANIES.

MARUTI SUZUKI, HYUNDAI, MAHINDRA,TATA,TOYOTA,HONDA, SKODA.

2. IN INDIA THE MOST DEMANDED BODY TYPES IS SUV , SEDAN AND HATCHBACKS.

3. EV MARKET IS STILL NEW IN INDIAN PEOPLE STILL PREFER PETROL AND DIESEL CAR OVER ELECTRIC VECHILES.

SO, WE NEW TO ADVERTISE OUR PRODUCT ADEQUATELY.

4. INDIAN AUTOMOBILE AND EV SALES IS GROWING RAPIDLY SO, IS TTHE GREAT TIME TO ENTER IN THE MARKETT .

5. VECHILE PRICE SHOULD BE UNDER 60,000 DOLLAR TO ATTARACT MOST NUMBER OF BUYERS

6. WE SHOULD TARGET TO 5 STATES WHICH HAVE HIGHEST SALES OF EV RECENTLY

UTTAR PRADESDH, KARNATAKA,DELHI,MAHARASTRA,BIHAR

7. KARNATAK SHOULD BE OUR PRIME TARGET AS IT HAS MOST E-CAR/BIKES SALES RECENTLY

8. THREE WHEELER ARE MOST POPULAR THEN TWO WHEELER AND THEN 4- WHEELERS IN INDIAN EV MARKET.

9. MOST NUMBER OF EV CHARGING STATIONS ARE SANCTIONED IN HIGHWAYS AND EXPRESSWAY

SO, WE CAN ALSO PLANT SOME CHARGING STATION INSIDE THE CITY TO EARN REVENUE

10. EV MARKET IS GROWING WORLDWIDE AND IN INDIA AS WELL SO, THIS IS MOST RAPIDLY CHANGING MARKET AND IF KEEP THESE POINT IN MIND THEN WE CAN MAKE A GOOD ENTRY IN EV MARKET.

In [ ]: