Market Segmentation Analysis

Of EV Vehicles

-Shruthi Nanditha Ganesh

Dataset-
https://github.com/shruthi1210/marketanalysisofEV/blob/main/3_ev_market_india_dataset.xlsx

Data Pre processing

Data preprocessing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure. It has traditionally been an important preliminary step for the data mining process.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
```

In [15]:
```
df = pd.read_excel(r"C:\Users\shruthi\Desktop\fenny labs\3_ev_market_india_dataset.xlsx")
df.head()
```

Out[15]:

	Brand	Model	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	RapidCharge	PowerTrain	PlugType	BodyStyle	Segment	Se
0	Tesla	Model 3 Long Range Dual Motor	4.6	233	450	161	940	Yes	AWD	Type 2 CCS	Sedan	D	
1	Volkswagen	ID.3 Pure	10.0	160	270	167	250	No	RWD	Type 2 CCS	Hatchback	C	
2	Polestar	2	4.7	210	400	181	620	Yes	AWD	Type 2 CCS	Liftback	D	
3	BMW	iX3	6.8	180	360	206	560	Yes	RWD	Type 2 CCS	SUV	D	
4	Honda	e	9.5	145	170	168	190	Yes	RWD	Type 2 CCS	Hatchback	B	

In [16]:
```
d= df.describe(include="all")
display(d)
```

	Brand	Model	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	RapidCharge	PowerTrain	PlugType	BodyStyle	Segment
count	103	103	103.000000	103.000000	103.000000	103.000000	103.000000	103	103	103	103	103
unique	33	102	NaN	NaN	NaN	NaN	NaN	2	3	4	9	8
top	Tesla	e-Soul 64 kWh	NaN	NaN	NaN	NaN	NaN	Yes	AWD	Type 2 CCS	SUV	C
freq	13	2	NaN	NaN	NaN	NaN	NaN	77	41	90	45	30
mean	NaN	NaN	7.396117	179.194175	338.786408	189.165049	444.271845	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	3.017430	43.573030	126.014444	29.566839	203.949253	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	2.100000	123.000000	95.000000	104.000000	170.000000	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	5.100000	150.000000	250.000000	168.000000	260.000000	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	7.300000	160.000000	340.000000	180.000000	440.000000	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	9.000000	200.000000	400.000000	203.000000	555.000000	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	22.400000	410.000000	970.000000	273.000000	940.000000	NaN	NaN	NaN	NaN	NaN

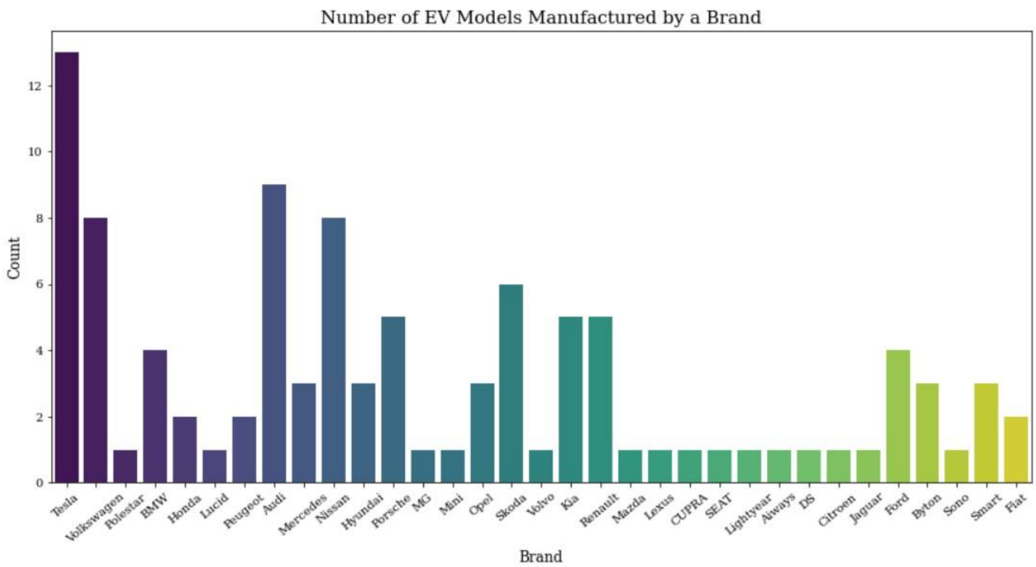
```
In [17]: print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Brand            103 non-null    object 
 1   Model            103 non-null    object 
 2   AccelSec         103 non-null    float64
 3   TopSpeed_KmH     103 non-null    int64  
 4   Range_Km         103 non-null    int64  
 5   Efficiency_WhKm  103 non-null    int64  
 6   FastCharge_KmH   103 non-null    int64  
 7   RapidCharge      103 non-null    object 
 8   PowerTrain       103 non-null    object 
 9   PlugType         103 non-null    object 
 10  BodyStyle        103 non-null    object 
 11  Segment          103 non-null    object 
 12  Seats            103 non-null    int64  
 13  PriceEuro        103 non-null    int64  
dtypes: float64(1), int64(6), object(7)
memory usage: 11.4+ KB
None
```

Exploratory Data Analysis

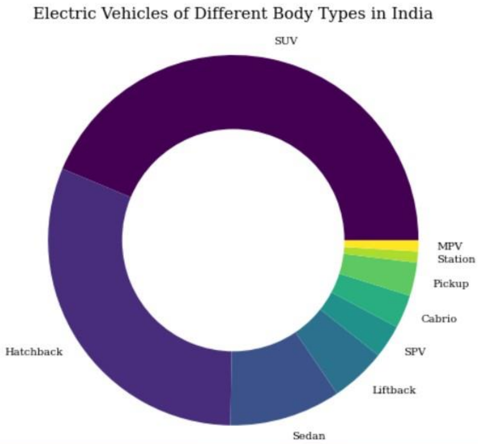
Exploratory Data Analysis, popularly abbreviated as EDA, is one of the most important steps in the data science pipeline. It is the process of gaining the information present inside the data with the help of summary statistics and visual representations. Keys features of this technique are presented in the below image.

```
In [19]: # brand-wise count of EV models
sns.catplot(data=df, x='Brand', kind='count', palette='viridis', height=6, aspect=2)
sns.despine(right=False, top=False)
plt.tick_params(axis='x', rotation=40)
plt.xlabel('Brand',family='serif', size=12)
plt.ylabel('Count', family='serif', size=12)
plt.xticks(family='serif')
plt.yticks(family='serif')
plt.title('Number of EV Models Manufactured by a Brand', family='serif', size=15)
plt.show()
```



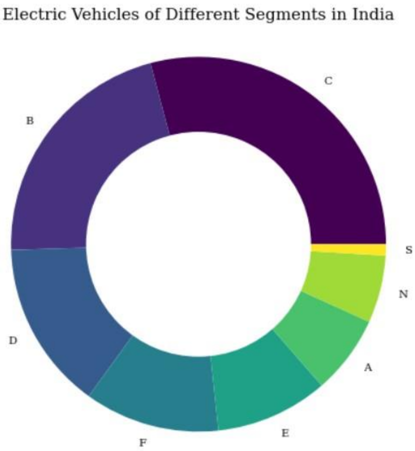
Anaysis of different body types of EVs
Observation: SUV and Hatchback body types form the majority while Station and MPV the minority

In [20]:
```
x = df['BodyStyle'].value_counts().plot.pie(radius=2, cmap='viridis', startangle=0, textprops=dict(family='serif'))
plt.pie(x=[1], radius=1.2, colors='white')
plt.title(label='Electric Vehicles of Different Body Types in India', family='serif', size=15, pad=100)
plt.ylabel('')
plt.show()
```



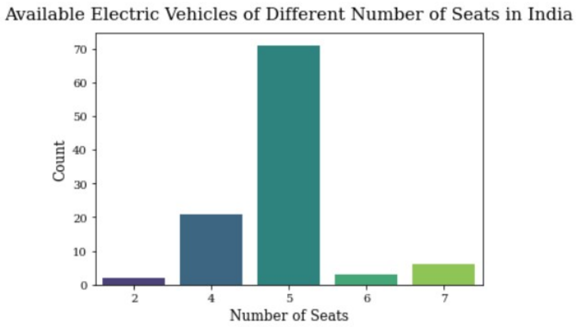
Analysis of different segments of EVs
Observation: B and C body segments form the majority while S and A the minority.

In [21]:
```
x = df['Segment'].value_counts().plot.pie(radius=2, cmap='viridis', startangle=0, textprops=dict(family='serif'), pctdistance=.5)
plt.pie(x=[1], radius=1.2, colors='white')
plt.title(label='Electric Vehicles of Different Segments in India', family='serif', size=15, pad=100)
plt.ylabel('')
plt.show()
```



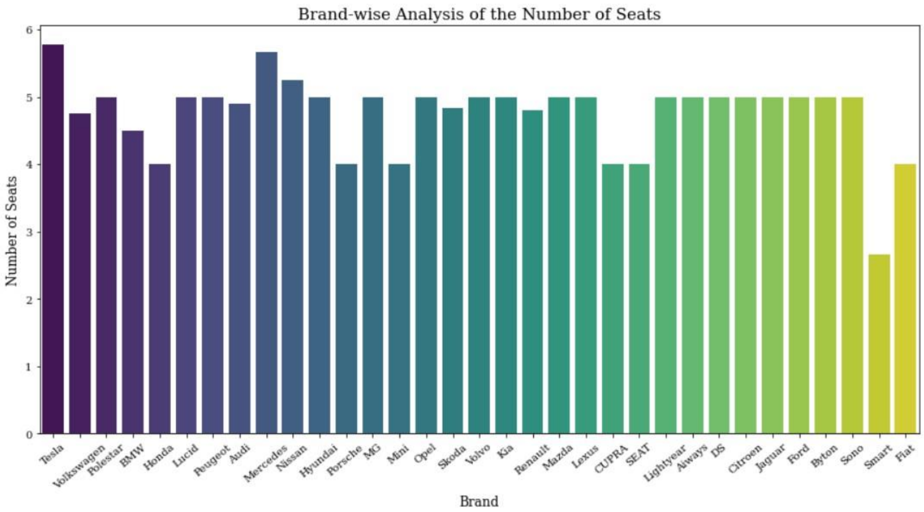
In [ ]:
```
#Analysis of EVs of different number of seats
#Observation: EVs with 5 sitters dominate the market while EVs with 2 sitters are less in number.
```

In [23]:
```
sns.countplot(data=df, x='Seats', palette='viridis')
plt.xlabel('Number of Seats', family='serif', size=12)
plt.ylabel('Count', family='serif', size=12)
plt.xticks(family='serif')
plt.yticks(family='serif')
plt.title(label='Available Electric Vehicles of Different Number of Seats in India', family='serif', size=15, pad=12)
plt.show()
```



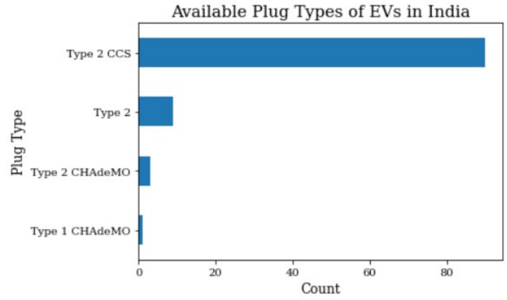
In [ ]:
```
#Analysis of the number of seats by each brand
#Observation: Based on the number of seats, Tesla, Mercedes and Nissan have the maximum number of seats and Smart the minimum.
```

In [24]:
```
sns.catplot(kind='bar', data=df, x='Brand', y='Seats', palette='viridis', ci=None, height=6, aspect=2)
sns.despine(right=False, top=False)
plt.tick_params(axis='x', rotation=40)
plt.xlabel('Brand',family='serif', size=12)
plt.ylabel('Number of Seats', family='serif', size=12)
plt.xticks(family='serif')
plt.yticks(family='serif')
plt.title('Brand-wise Analysis of the Number of Seats', family='serif', size=15);
```

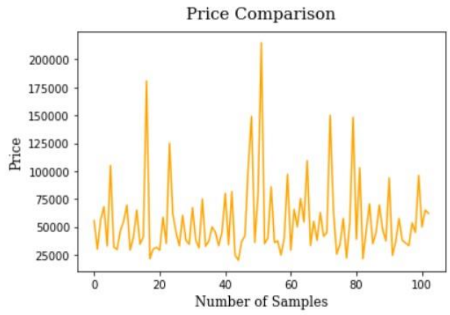


```
In [ ]: #Analysis of different plug types
        #Observation: EVs with plus type of 'Type 2 CCS' seem to dominate the market.
```

```
In [25]: df['PlugType'].value_counts().sort_values(ascending=True).plot.barh()
         plt.xlabel('Count', family='serif', size=12)
         plt.ylabel('Plug Type', family='serif', size=12)
         plt.xticks(family='serif')
         plt.yticks(family='serif')
         plt.title('Available Plug Types of EVs in India', family='serif', size=15)
         plt.show()
```

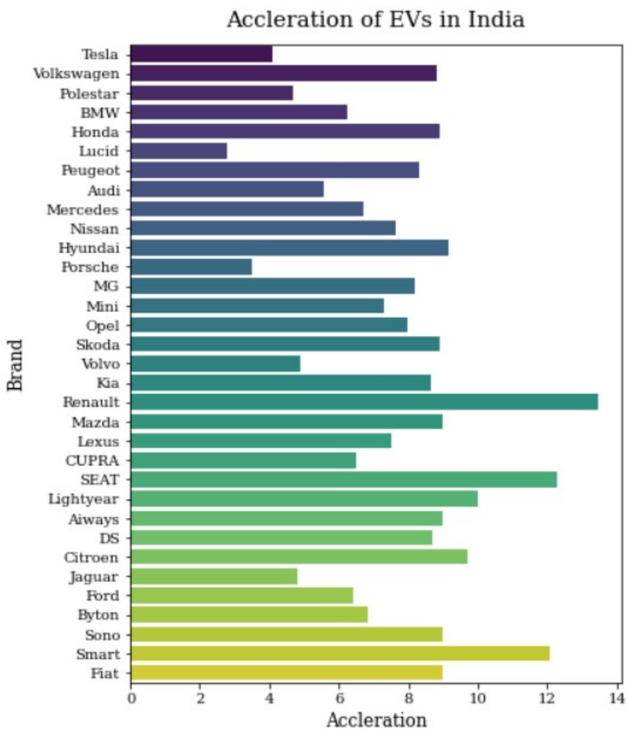



```
In [26]: plt.plot(df['PriceEuro'], color='orange')
         plt.xlabel('Number of Samples', family='serif', size=12)
         plt.ylabel('Price', family='serif', size=12)
         plt.title('Price Comparison', family='serif', size=15, pad=12);
```



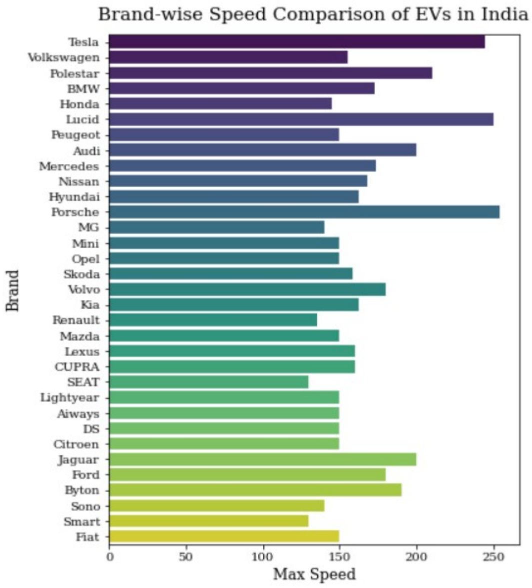
#Analysis of EVs based on accleration
#Observation: Based on accleration, EVs from Renault, Seat and Smart are the top performers while Tesla, Lucid and Porsche dont make it to the same.

```
In [27]: plt.figure(figsize=(6, 8))
         sns.barplot(data=df, y='Brand', x='AccelSec', ci=None, palette='viridis')
         plt.xticks(family='serif')
         plt.yticks(family='serif')
         plt.xlabel('Accleration', family='serif', size=12)
         plt.ylabel('Brand', family='serif', size=12)
         plt.title(label='Accleration of EVs in India', family='serif', size=15, pad=12)
         plt.show()
```

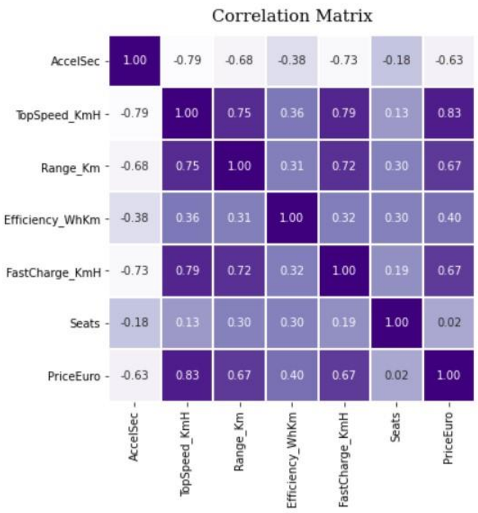



```
#Analysis of EVs based on speed
#Observation: Based on speed parameter, EVs from Tesla, Lucid and Porsche are the top performers while Renault, Smart and SEAT dont make it to the same.
```

In [28]:
```
plt.figure(figsize=(6, 8))
sns.barplot(data=df, x='TopSpeed_KmH', y='Brand', ci=None, palette='viridis')
plt.xticks(family='serif')
plt.yticks(family='serif')
plt.xlabel('Max Speed', family='serif', size=12)
plt.ylabel('Brand', family='serif', size=12)
plt.title(label='Brand-wise Speed Comparison of EVs in India', family='serif', size=15, pad=12)
plt.show()
```

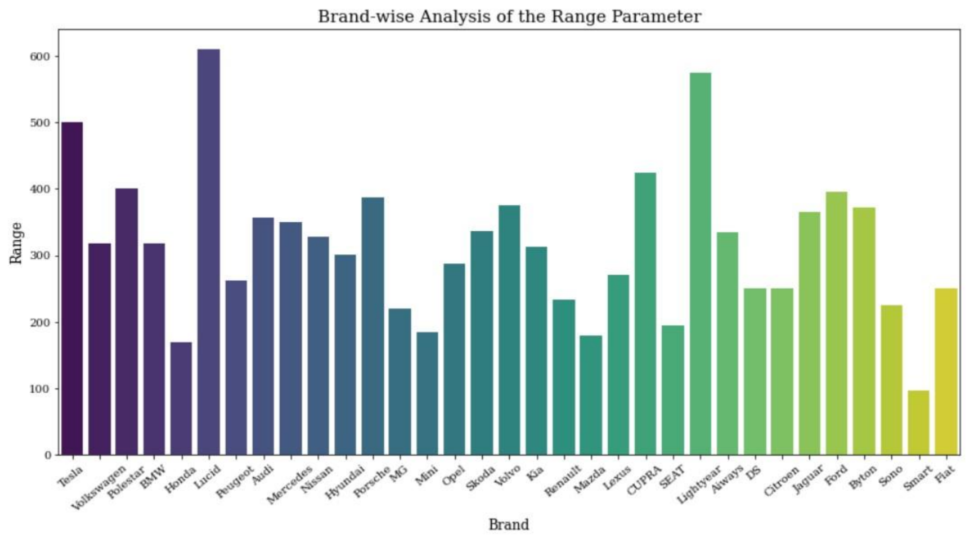


In [29]:
```
plt.figure(figsize=(6,6))
sns.heatmap(data=df.corr(), annot=True, cmap='Purples', cbar=False, square=True, fmt='.2f', linewidths=.3)
plt.title('Correlation Matrix', family='serif', size=15, pad=12);
```



In [ ]:
```
#Analysis of EVs based on the range parameter
#Observation: Based on range (Km), Lucid, Lightyear and Tesla have the highest range and Smart the lowest.
```

In [30]:
```
sns.catplot(kind='bar', data=df, x='Brand', y='Range_Km', palette='viridis', ci=None, height=6, aspect=2)
sns.despine(right=False, top=False)
plt.tick_params(axis='x', rotation=40)
plt.xlabel('Brand',family='serif', size=12)
plt.ylabel('Range', family='serif', size=12)
plt.xticks(family='serif')
plt.yticks(family='serif')
plt.title('Brand-wise Analysis of the Range Parameter', family='serif', size=15);
```



Segmentation Approaches

Clustering

Clustering is an unsupervised machine learning technique of grouping similar data points into clusters. The sole objective of this technique is to segregate datapoints with similar traits and place them into different clusters. There are several algorithms to perform clustering on data such as k-means clustering, hierarchical clustering, density-based clustering etc.

K-Means Clustering

K-Means Clustering is an unsupervised learning algorithm whose job is to group the unlabelled dataset into different clusters where each datapoint belongs to only one cluster. Here, K is the number of clusters that need to be created in the process. The algorithm finds its applicability into a variety of use cases including market segmentation, image segmentation, image compression, document clustering etc. The below image is the results of clustering on one of our datasets.

Principle Component Analysis

Principal component analysis (PCA) is a linear dimensionality-reduction technique that is used to reduce the dimensionality of large data sets by transforming a large set of variables into a smaller one while preserving most of the information present in the large set.

Elbow Method

The Elbow method is a way of determining the optimal number of clusters (k) in K-Means Clustering. It is based on calculating the Within Cluster Sum of Squared Errors (WCSS) for a different number of clusters (k) and selecting the k for which change in WCSS first starts to diminish. When you plot its graph, at one point the line starts to run parallel to the X-axis and that point, known as the Elbow Point, is considered as the best value for the k .

In [ ]:
```
#Model Building Using K-Means Clusteing
```

In [31]:
```
# encoding the categorical features

# PowerTrain feature
df['PowerTrain'].replace(to_replace=['RWD','FWD','AWD'],value=[0, 1, 2],inplace=True)

# RapidCharge feature
df['RapidCharge'].replace(to_replace=['No','Yes'],value=[0, 1],inplace=True)


# selecting features for building a model
X = df[['AccelSec','TopSpeed_KmH','Efficiency_WhKm','FastCharge_KmH', 'Range_Km', 'RapidCharge', 'Seats', 'PriceEuro','PowerTrair


# feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


# applying Principle Component Analysis (PCA)
pca = PCA(n_components=9)
X_pca = pca.fit_transform(X_scaled)
df_pca = pd.DataFrame(X_pca, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9'])
df_pca.head()
```

Out[31]:

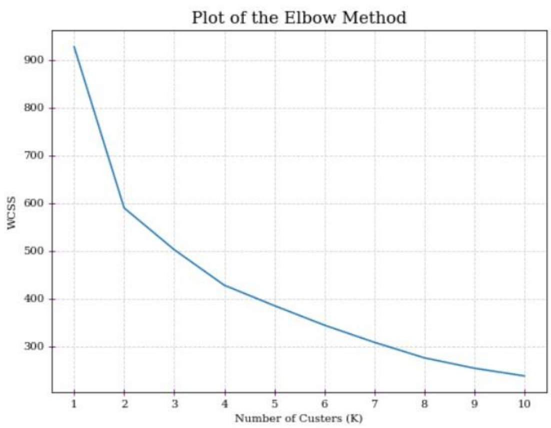
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
0	2.429225	-0.554599	-1.147772	-0.882791	0.839988	-0.959297	0.998880	0.711148	-0.396662
1	-2.322483	-0.345449	0.896473	-1.305529	0.079598	0.235116	-0.213678	-0.544135	-0.181867
2	1.587851	0.008899	-0.650523	0.041024	0.593537	-0.698248	0.058718	0.248837	-0.202775
3	0.291018	-0.000150	-0.307702	-0.514196	-1.608861	0.291624	0.364999	-0.235543	0.261663
4	-2.602679	-0.626489	-0.888088	0.585294	-0.802108	0.027387	-0.084955	-0.507790	-0.049904

In [32]:
```
# plotting the results of Elbow

wcss = []

for i in range(1, 11):
  kmean = KMeans(n_clusters=i, init='k-means++', random_state=90)
  kmean.fit(X_pca)
  wcss.append(kmean.inertia_)

plt.figure(figsize=(8,6))
plt.title('Plot of the Elbow Method', size=15, family='serif')
plt.plot(range(1, 11), wcss)
plt.xticks(range(1, 11), family='serif')
plt.yticks(family='serif')
plt.xlabel('Number of Custers (K)', family='serif')
plt.ylabel('WCSS', family='serif')
plt.grid()
plt.tick_params(axis='both', direction='inout', length=6, color='purple', grid_color='lightgray', grid_linestyle='--')
plt.show()
```



In [33]:
```
# training the model using k=4 as rendered by the above plot
kmean = KMeans(n_clusters=4, init='k-means++', random_state=90)
kmean.fit(X_pca)
```

Out[33]:

```
KMeans
KMeans(n_clusters=4, random_state=90)
```

In [34]:

```
# check the labels assigned to each data point
print(kmean.labels_)
```

```
[0 3 2 1 1 0 3 3 1 2 2 1 1 2 3 1 0 1 3 1 1 2 1 0 0 1 1 2 3 3 2 1 1 2 1 1 1
 3 3 2 0 1 2 1 1 1 1 0 0 3 2 0 1 1 2 1 1 3 1 0 3 2 2 2 3 0 1 2 3 2 1 2 0 2
 1 1 2 3 2 0 1 2 3 1 2 1 2 2 2 1 2 3 3 2 1 1 1 3 1 2 2 2 2]
```

In [35]:

```
# check the size of clusters
pd.Series(kmean.labels_).value_counts()
```

Out[35]:

```
1    39
2    32
3    19
0    13
dtype: int64
```

In [36]:

```
df['clusters'] = kmean.labels_
# visualizing clusters
plt.figure(figsize=(7,5))
sns.scatterplot(data=df_pca, x='PC1', y='PC9', s=70, hue=kmean.labels_, palette='viridis', zorder=2, alpha=.9)
plt.scatter(x=kmean.cluster_centers_[:,0], y=kmean.cluster_centers_[:,1], marker="*", c="r", s=80, label="centroids")
plt.xlabel('PC1', family='serif', size=12)
plt.ylabel('PC9', family='serif', size=12)
plt.xticks(family='serif')
plt.yticks(family='serif')
plt.grid()
plt.tick_params(grid_color='lightgray', grid_linestyle='--', zorder=1)
plt.legend(title='Labels', fancybox=True, shadow=True)
plt.title('K-Means Clustering Results', family='serif', size=15)
plt.show()
```