In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import re
from sklearn.cluster import KMeans
import plotly.express as ps
```

In [5]:
```
df=pd.read_csv(r"C:\Users\aksha\Downloads\Electric_Vehicle_Population_Data.csv")
```

In [6]:
```
df.head()
```

Out[6]:

	VIN (1-10)	County	City	State	ZIP Code	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District
0	WA1AAAGE2M	Kitsap	POULSBO	WA	98370	2021	AUDI	E-TRON	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	222	0	23.0
1	WBY8P2C00L	King	SEATTLE	WA	98122	2020	BMW	I3	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	153	0	37.0
2	5YJXCBE21K	Cowlitz	SILVERLAKE	WA	98645	2019	TESLA	MODEL X	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	289	0	20.0
3	1FTZR081XY	King	SEATTLE	WA	98117	2000	FORD	RANGER	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	58	0	36.0
4	WBY1Z6C55H	King	SEATTLE	WA	98119	2017	BMW	I3	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	81	0	36.0

In [7]:
```
df.columns.unique()
```

Out[7]:
```
Index(['VIN (1-10)', 'County', 'City', 'State', 'ZIP Code', 'Model Year',
       'Make', 'Model', 'Electric Vehicle Type',
       'Clean Alternative Fuel Vehicle (CAFV) Eligibility', 'Electric Range',
       'Base MSRP', 'Legislative District', 'DOL Vehicle ID',
       'Vehicle Location'],
      dtype='object')
```

In [8]:
```
df.shape
```

Out[8]:
```
(79767, 15)
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 79767 entries, 0 to 79766
Data columns (total 15 columns):
 #   Column                                             Non-Null Count  Dtype
---  ------                                             --------------  -----
 0   VIN (1-10)                                         79767 non-null  object
 1   County                                             79762 non-null  object
 2   City                                               79767 non-null  object
 3   State                                              79767 non-null  object
 4   ZIP Code                                           79767 non-null  int64
 5   Model Year                                         79767 non-null  int64
 6   Make                                               79767 non-null  object
 7   Model                                              79767 non-null  object
 8   Electric Vehicle Type                              79767 non-null  object
 9   Clean Alternative Fuel Vehicle (CAFV) Eligibility  79767 non-null  object
 10  Electric Range                                     79767 non-null  int64
 11  Base MSRP                                          79767 non-null  int64
 12  Legislative District                               79621 non-null  float64
 13  DOL Vehicle ID                                     79767 non-null  int64
 14  Vehicle Location                                   79763 non-null  object
dtypes: float64(1), int64(5), object(9)
memory usage: 9.1+ MB
```

In [10]:
```
df.isnull().sum().sort_values(ascending = False)
```

Out[10]:
```
Legislative District                               146
County                                               5
Vehicle Location                                     4
VIN (1-10)                                           0
City                                                 0
State                                                0
ZIP Code                                             0
Model Year                                           0
Make                                                 0
Model                                                0
Electric Vehicle Type                                0
Clean Alternative Fuel Vehicle (CAFV) Eligibility    0
Electric Range                                       0
Base MSRP                                            0
DOL Vehicle ID                                       0
dtype: int64
```

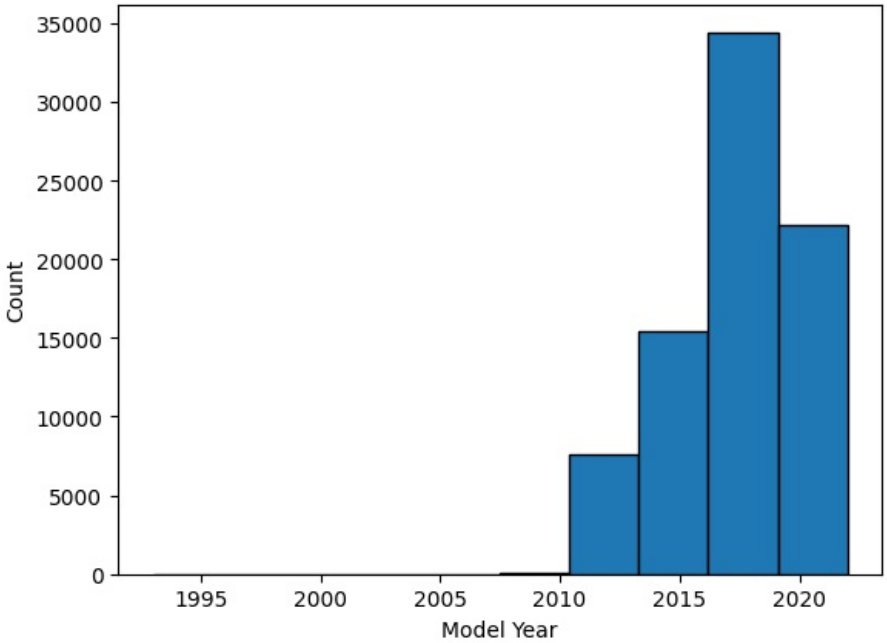
In [11]:
```
df.duplicated().sum()
```

Out[11]:
```
0
```

In [12]:
```
df['County'] = df['County'].fillna('Unknown')
df['Vehicle Location'] = df['Vehicle Location'].fillna('Unknown')
df['Legislative District'] = df['Legislative District'].fillna(00)
```

In [13]:
```
plt.hist(df["Model Year"],bins=10,edgecolor='black')
plt.xlabel("Model Year")
plt.ylabel("Count")
```

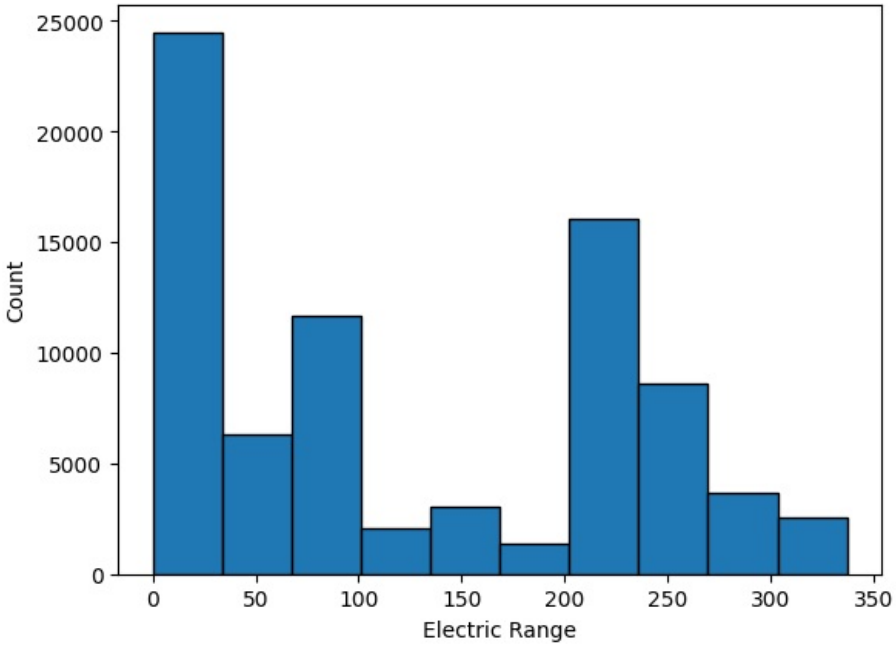
Out[13]:
```
Text(0, 0.5, 'Count')
```



In [14]:
```
plt.hist(df["Electric Range"],bins=10,edgecolor='black')
```

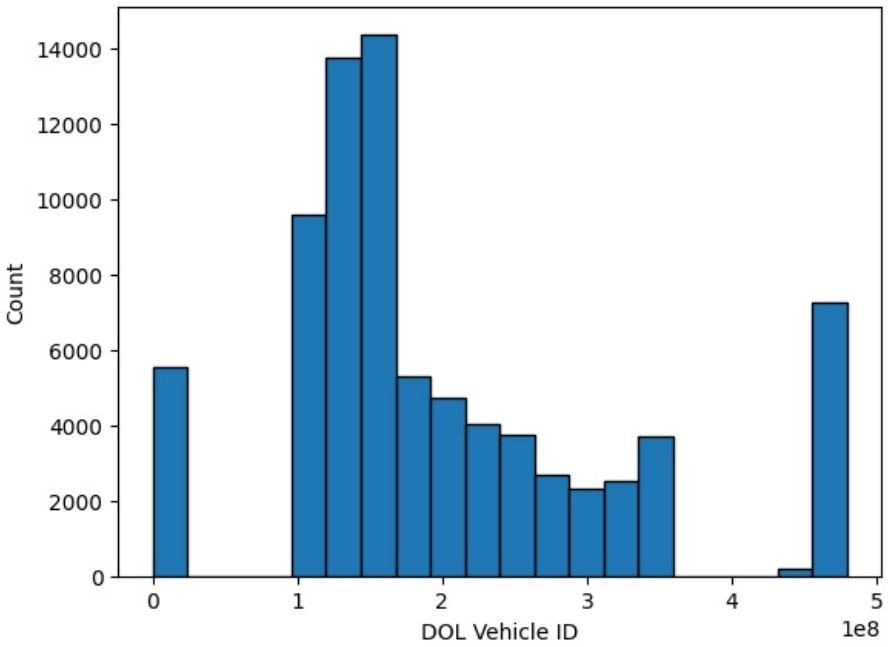
```
plt.xlabel("Electric Range")
plt.ylabel("Count")
```

Out[14]: Text(0, 0.5, 'Count')

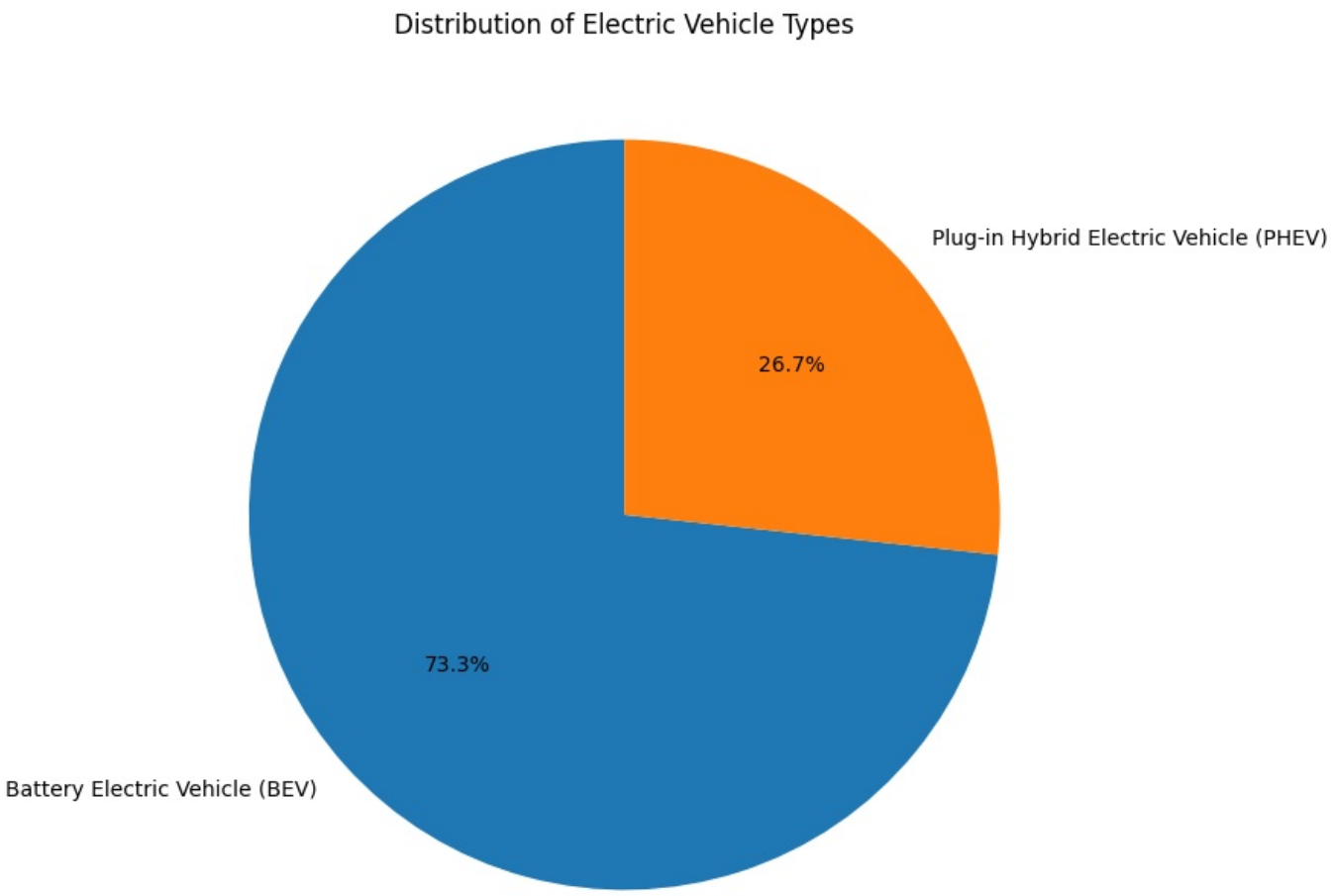


In [15]:
```
plt.hist(df["DOL Vehicle ID"],bins=20,edgecolor='black')
plt.xlabel("DOL Vehicle ID")
plt.ylabel("Count")
```

Out[15]: Text(0, 0.5, 'Count')

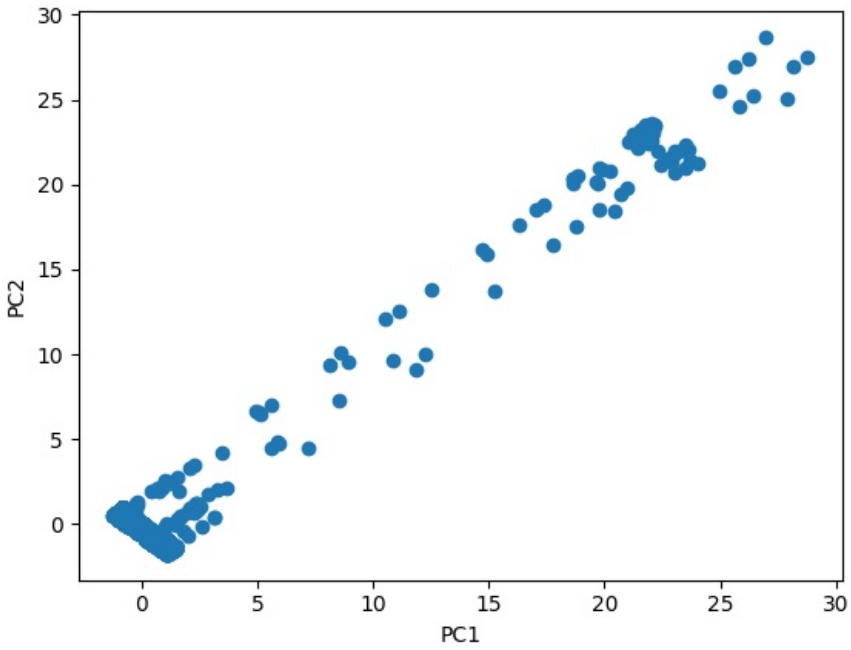


In [16]:
```
electric_vehicle_types_distribution = df['Electric Vehicle Type'].value_counts()

plt.figure(figsize=(8, 8))
plt.pie(electric_vehicle_types_distribution, labels=electric_vehicle_types_distribution.index, autopct='%1.1f%%
plt.title('Distribution of Electric Vehicle Types')
plt.show()
```



In [17]:
```
# Perform PCA
# Select the columns for PCA
columns_for_pca = ['Electric Range', 'ZIP Code']

# Create a new DataFrame with the selected columns
X = df[columns_for_pca]

# Standardize the data
scaler = StandardScaler()
X_std = scaler.fit_transform(X)

# Perform PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_std)
```

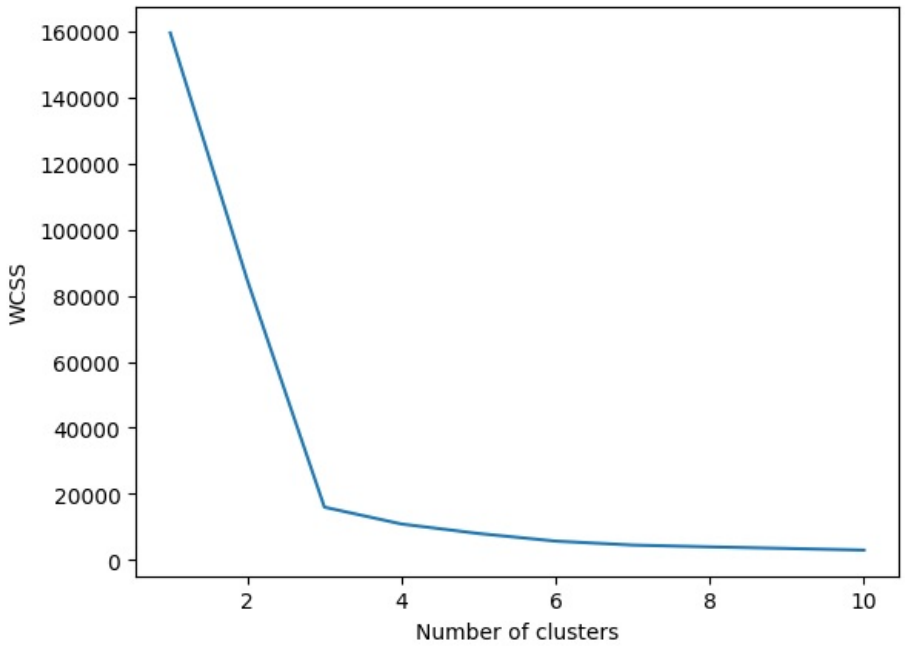
In [24]:
```
# Plot the PCA results
plt.scatter(X_pca[:, 0], X_pca[:, 1])
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```



In [25]:

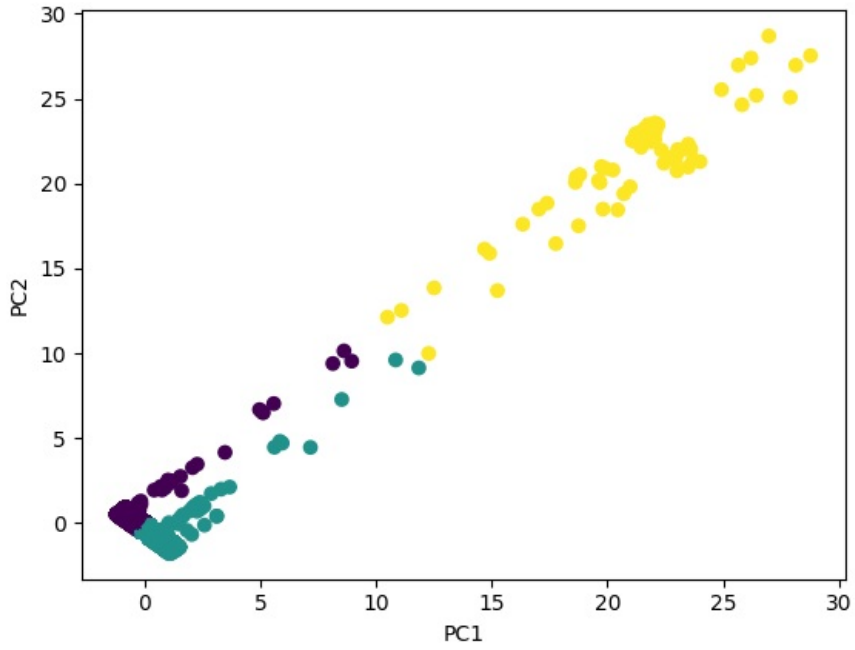
```
# Perform k-means clustering
# Determine the optimal number of clusters using the elbow method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X_std)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning

C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning

C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning

C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning

C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning

C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning

C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning

C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning

C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning

C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
```

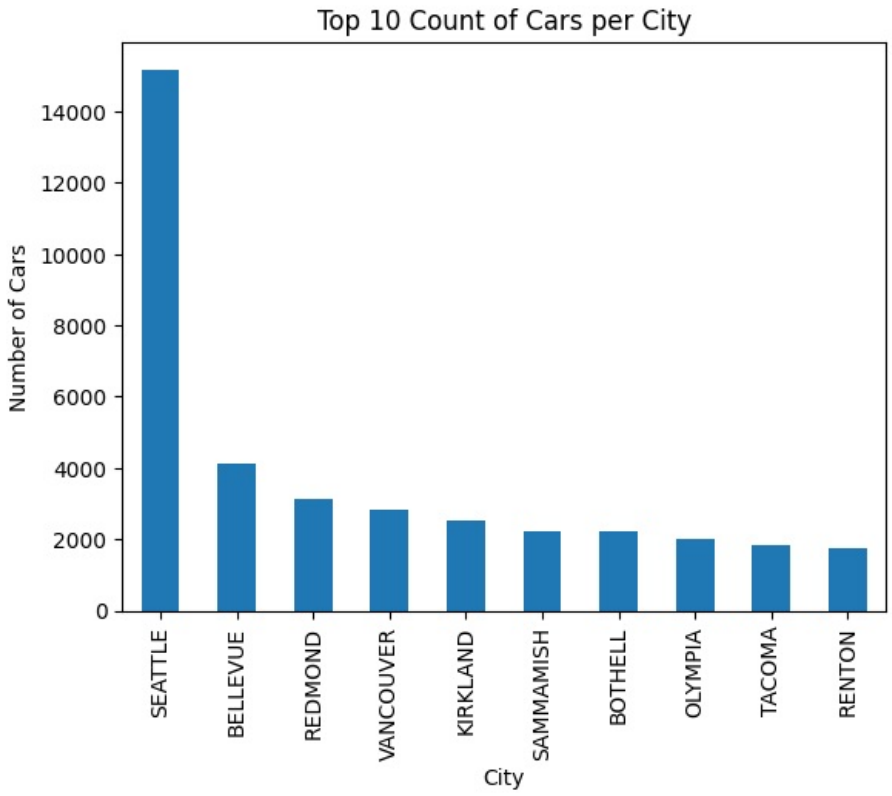


In [26]:
```
# Fit the k-means model with the optimal number of clusters
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
y_kmeans = kmeans.fit_predict(X_std)

# Plot the k-means results
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_kmeans)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```

```
C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
```

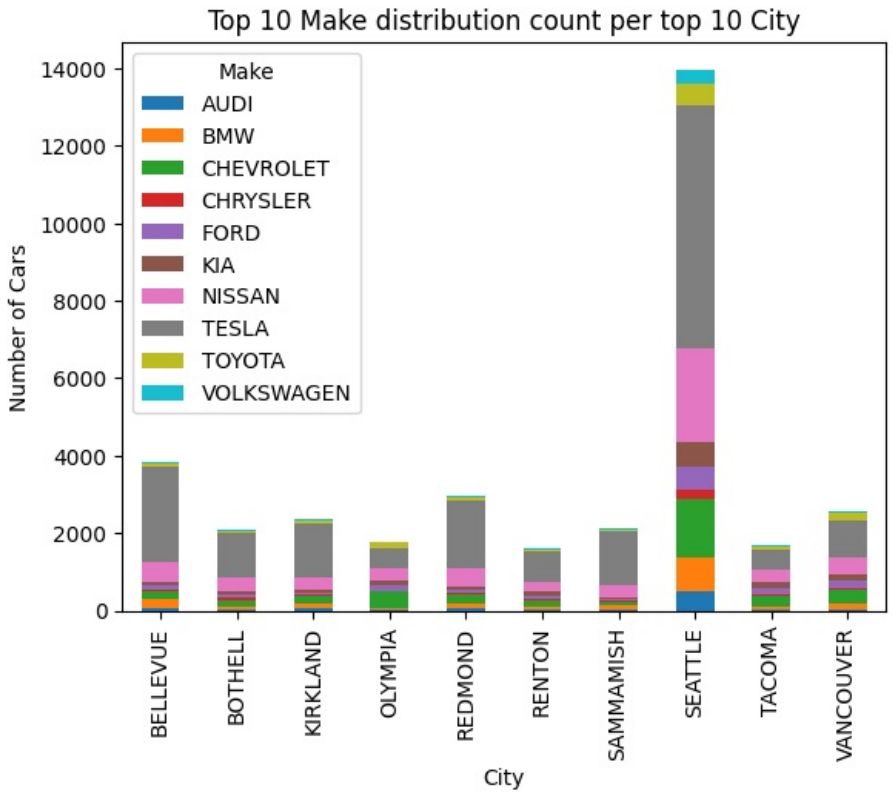


In [21]:
```
car_counts_Cty = df['City'].value_counts().nlargest(10)
# plot the counts
car_counts_Cty.plot(kind='bar')
plt.xlabel('City')
plt.ylabel('Number of Cars')
plt.title('Top 10 Count of Cars per City')
plt.show()
car_counts_cty_df = car_counts_Cty.to_frame()
```



In [22]:

```
cnt_MkCity = df.groupby(['City', 'Make']).size().reset_index(name='Count')
# Group the data by county and make, and sum the counts for each group
grouped_data_cty = cnt_MkCity.groupby(['City', 'Make'])['Count'].sum().reset_index()
# Group the data by county and sum the counts for each county
city_counts = grouped_data_cty.groupby('City')['Count'].sum().reset_index()
make_counts = grouped_data_cty.groupby('Make')['Count'].sum().reset_index()
# Sort the counties by count in descending order, and select the top 10
top_cities = city_counts.sort_values(by='Count', ascending=False).head(10)
top_makes = make_counts.sort_values(by='Count', ascending=False).head(10)
# Filter the data to only include the top 10 counties
filtered_data_Cty = grouped_data_cty[grouped_data_cty['City'].isin(top_cities['City']) & grouped_data_cty['Make
# Pivot the data to create a matrix with counties as rows, makes as columns, and counts as values
pivoted_data_cty = filtered_data_Cty.pivot(index='City', columns='Make',values='Count').fillna(0)
# Create a stacked bar plot of the pivoted data
pivoted_data_cty.plot(kind='bar', stacked=True)
# Set the title and axis labels
plt.title('Top 10 Make distribution count per top 10 City')
plt.xlabel('City')
plt.ylabel('Number of Cars')
# Show the plot
plt.show()
pivoted_data_cty.head()
```



Out[22]:

Make City	AUDI	BMW	CHEVROLET	CHRYSLER	FORD	KIA	NISSAN	TESLA	TOYOTA	VOLKSWAGEN
BELLEVUE	81	201	226	44	89	108	508	2448	97	43
BOTHELL	36	64	174	55	85	79	384	1120	41	36
KIRKLAND	68	124	176	33	71	86	317	1377	62	33
OLYMPIA	28	40	413	33	135	139	319	510	151	22
REDMOND	59	140	202	51	68	88	469	1767	82	39

In [23]:

```
year_wise_cars = df.groupby('Model Year')['VIN (1-10)'].count().reset_index()
year_wise_cars.columns = ['year','num_cars']
fig = ps.line(year_wise_cars,x="year", y="num_cars", title='Year Wise Number of Cars',markers=True)
fig.show()
```