In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelBinarizer
```

In [3]:

```
# Load the dataset
df = pd.read_csv(r"C:\Users\aksha\Downloads\Indian automoble buying behavour study 1.0.csv")
```

In [4]:

```
df.head()
```

Out[4]:

	Age	Profession	Marrital Status	Education	No of Dependents	Personal loan	House Loan	Wife Working	Salary	Wife Salary	Total Salary	Make	Price
0	27	Salaried	Single	Post Graduate	0	Yes	No	No	800000	0	800000	i20	800000
1	35	Salaried	Married	Post Graduate	2	Yes	Yes	Yes	1400000	600000	2000000	Ciaz	1000000
2	45	Business	Married	Graduate	4	Yes	Yes	No	1800000	0	1800000	Duster	1200000
3	41	Business	Married	Post Graduate	3	No	No	Yes	1600000	600000	2200000	City	1200000
4	31	Salaried	Married	Post Graduate	2	Yes	No	Yes	1800000	800000	2600000	SUV	1600000

In [5]:

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

```
Age                 0
Profession          0
Marrital Status     0
Education           0
No of Dependents    0
Personal loan       0
House Loan          0
Wife Working        0
Salary              0
Wife Salary         0
Total Salary        0
Make                0
Price               0
dtype: int64
```

In [6]:

```
# Check the data types of the columns
print(df.dtypes)
```

```
Age                  int64
Profession          object
Marrital Status     object
Education           object
No of Dependents     int64
Personal loan       object
House Loan          object
Wife Working        object
Salary               int64
Wife Salary          int64
Total Salary         int64
Make                object
Price                int64
dtype: object
```

In [7]:

```
# Check the distribution of the numerical columns
print(df.describe())
```

```
             Age  No of Dependents        Salary   Wife Salary  Total Salary  \
count  99.000000         99.000000  9.900000e+01  9.900000e+01  9.900000e+01   
mean   36.313131          2.181818  1.736364e+06  5.343434e+05  2.270707e+06   
std     6.246054          1.335265  6.736217e+05  6.054450e+05  1.050777e+06   
min    26.000000          0.000000  2.000000e+05  0.000000e+00  2.000000e+05   
25%    31.000000          2.000000  1.300000e+06  0.000000e+00  1.550000e+06   
50%    36.000000          2.000000  1.600000e+06  5.000000e+05  2.100000e+06   
75%    41.000000          3.000000  2.200000e+06  9.000000e+05  2.700000e+06   
max    51.000000          4.000000  3.800000e+06  2.100000e+06  5.200000e+06   

              Price  
count  9.900000e+01  
mean   1.194040e+06  
std    4.376955e+05  
min    1.100000e+05  
25%    8.000000e+05  
50%    1.200000e+06  
75%    1.500000e+06  
max    3.000000e+06  
```

In [8]:
```
df.columns.unique()
```

Out[8]:
```
Index(['Age', 'Profession', 'Marrital Status', 'Education', 'No of Dependents',
       'Personal loan', 'House Loan', 'Wife Working', 'Salary', 'Wife Salary',
       'Total Salary', 'Make', 'Price'],
      dtype='object')
```

In [9]:
```
df.shape
```

Out[9]:
```
(99, 13)
```

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

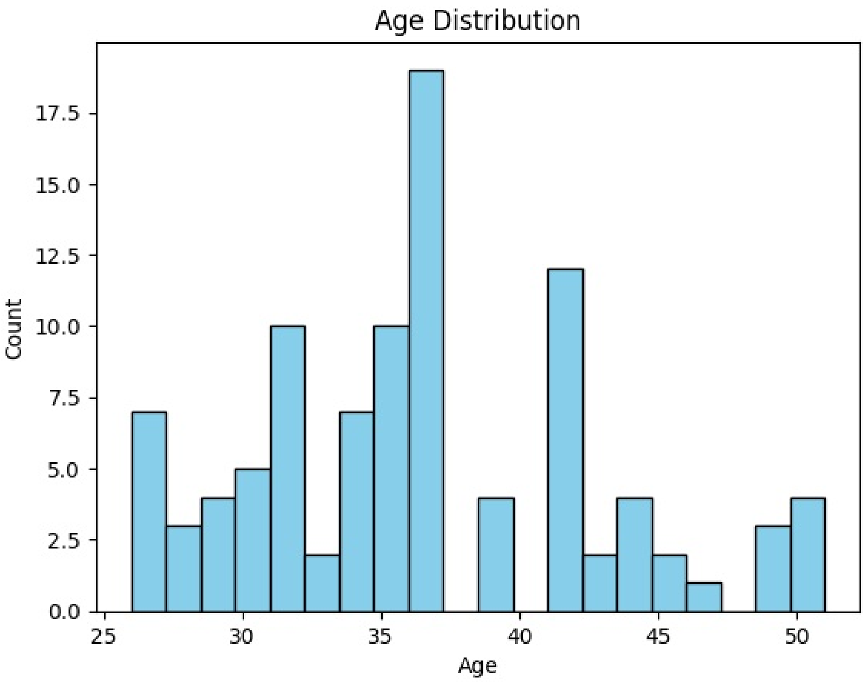
Out[10]:
```
Age                 0
Profession          0
Marrital Status     0
Education           0
No of Dependents    0
Personal loan       0
House Loan          0
Wife Working        0
Salary              0
Wife Salary         0
Total Salary        0
Make                0
Price               0
dtype: int64
```

In [11]:
```
# Check the distribution of the categorical columns
for col in df.select_dtypes(include=['object']):
    print(df[col].value_counts())
```

```
Profession
Salaried    64
Business    35
Name: count, dtype: int64
Marrital Status
Married    84
Single     15
Name: count, dtype: int64
Education
Post Graduate    56
Graduate         43
Name: count, dtype: int64
Personal loan
No     67
Yes    32
Name: count, dtype: int64
House Loan
No     62
Yes    37
Name: count, dtype: int64
Wife Working
Yes    52
No     46
m       1
Name: count, dtype: int64
Make
SUV        19
Baleno     19
Creata     14
i20        12
Ciaz       12
City       10
Duster      7
Verna       4
Luxuray     2
Name: count, dtype: int64
```

In [12]:

```
# Histogram of the 'Age' column
plt.hist(df['Age'], bins=20, color='skyblue', edgecolor='black')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



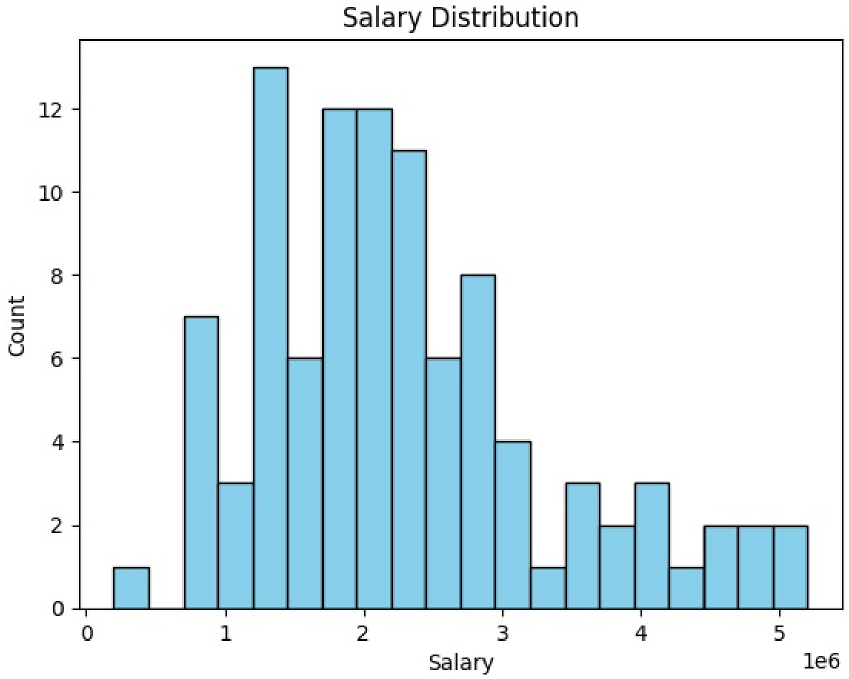
In [13]:

```
profession_column = df["Profession"]
encoder = LabelBinarizer()
one_hot_encoded_profession = encoder.fit_transform(profession_column)
df["Profession"] = one_hot_encoded_profession
print(df)
```

```
    Age  Profession Marrital Status      Education  No of Dependents  \
0    27           1          Single  Post Graduate                 0
1    35           1         Married  Post Graduate                 2
2    45           0         Married       Graduate                 4
3    41           0         Married  Post Graduate                 3
4    31           1         Married  Post Graduate                 2
.. ...         ...             ...            ...               ...
94   27           0          Single       Graduate                 0
95   50           1         Married  Post Graduate                 3
96   51           0         Married       Graduate                 2
97   51           1         Married  Post Graduate                 2
98   51           1         Married  Post Graduate                 2

   Personal loan House Loan Wife Working   Salary  Wife Salary  Total Salary  \
0            Yes         No           No   800000            0        800000
1            Yes        Yes          Yes  1400000       600000       2000000
2            Yes        Yes           No  1800000            0       1800000
3             No         No          Yes  1600000       600000       2200000
4            Yes         No          Yes  1800000       800000       2600000
..           ...        ...          ...      ...          ...           ...
94            No         No           No  2400000            0       2400000
95            No         No          Yes  3800000      1300000       5100000
96           Yes        Yes           No  2200000            0       2200000
97            No         No          Yes  2700000      1300000       4000000
98           Yes        Yes           No  2200000            0       2200000

      Make    Price
0      i20   800000
1     Ciaz  1000000
2   Duster  1200000
3     City  1200000
4      SUV  1600000
..     ...      ...
94     SUV  1600000
95     SUV  1600000
96    Ciaz  1100000
97  Creata  1500000
98    Ciaz  1100000

[99 rows x 13 columns]
```

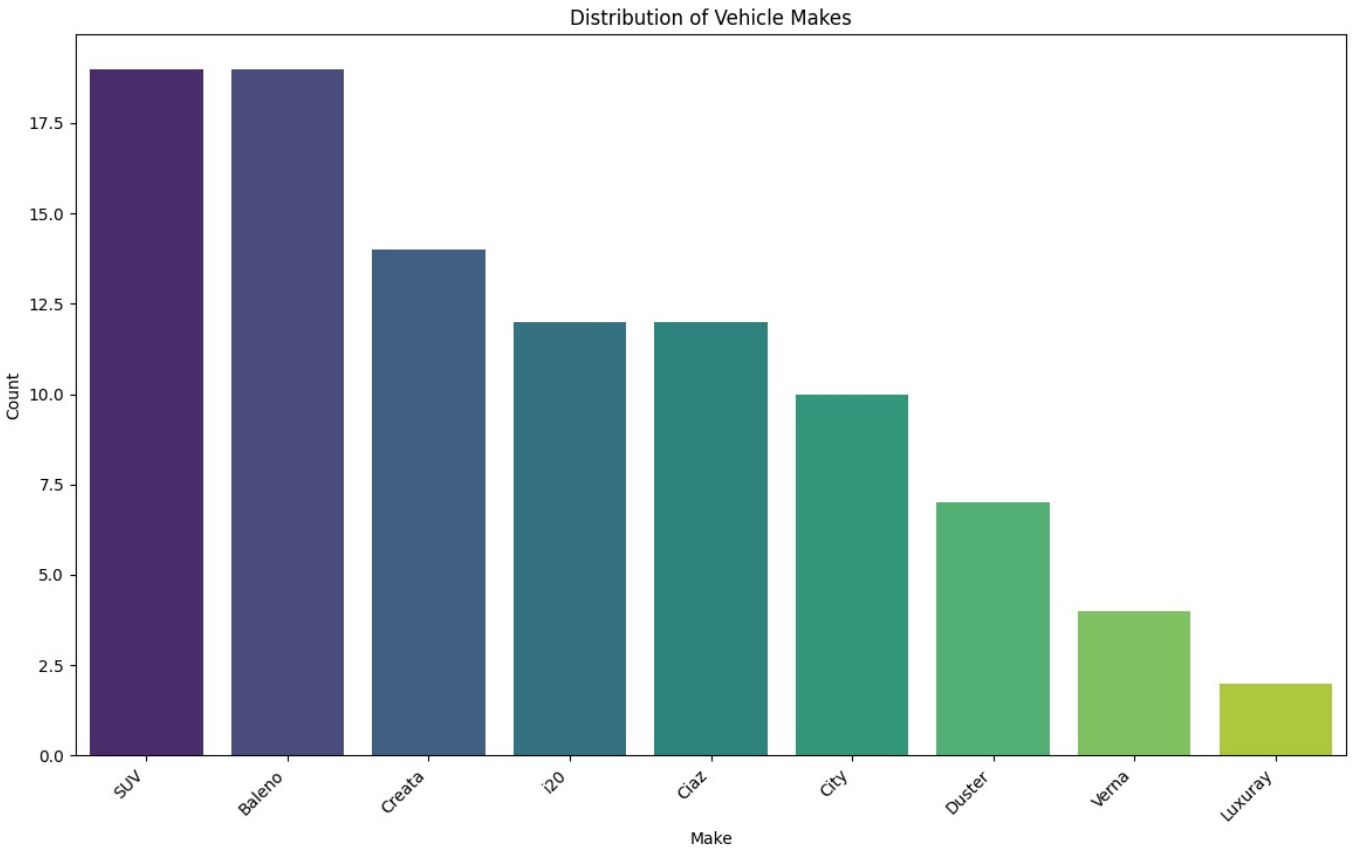
In [14]:

```
# Histogram of 'Salary'
plt.hist(df['Total Salary'], bins=20, color='skyblue', edgecolor='black')
plt.title('Salary Distribution')
plt.xlabel('Salary')
plt.ylabel('Count')
plt.show()
```



In [15]:

```
plt.figure(figsize=(14, 8))
sns.countplot(x='Make', data=df, order=df['Make'].value_counts().index, palette='viridis')
plt.title('Distribution of Vehicle Makes')
plt.xlabel('Make')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.show()
```

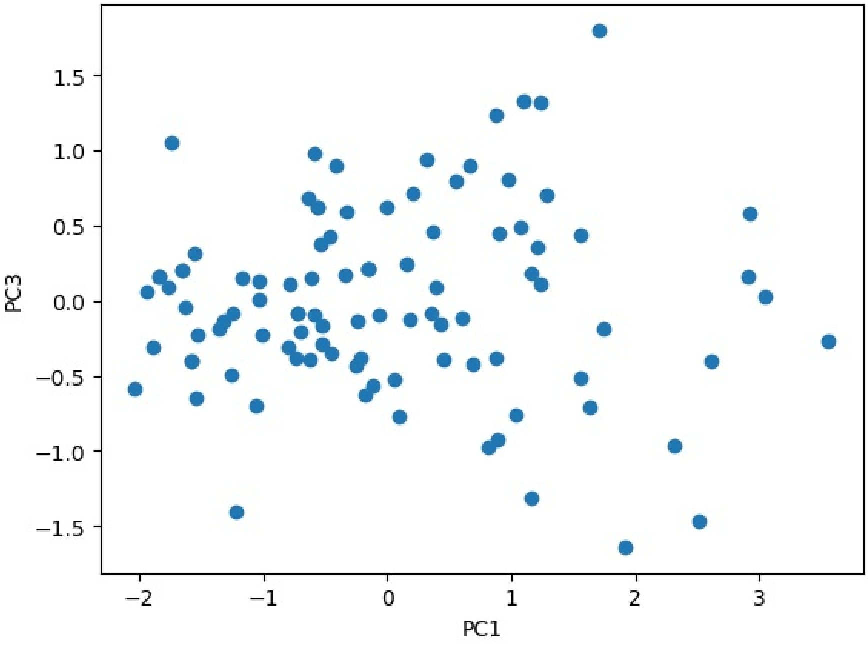


In [16]:

```
# Perform PCA
columns_for_pca = ['Age', 'Total Salary', 'Profession']
X = df[columns_for_pca]
scaler = StandardScaler()
X_std = scaler.fit_transform(X)
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_std)
```

In [17]:

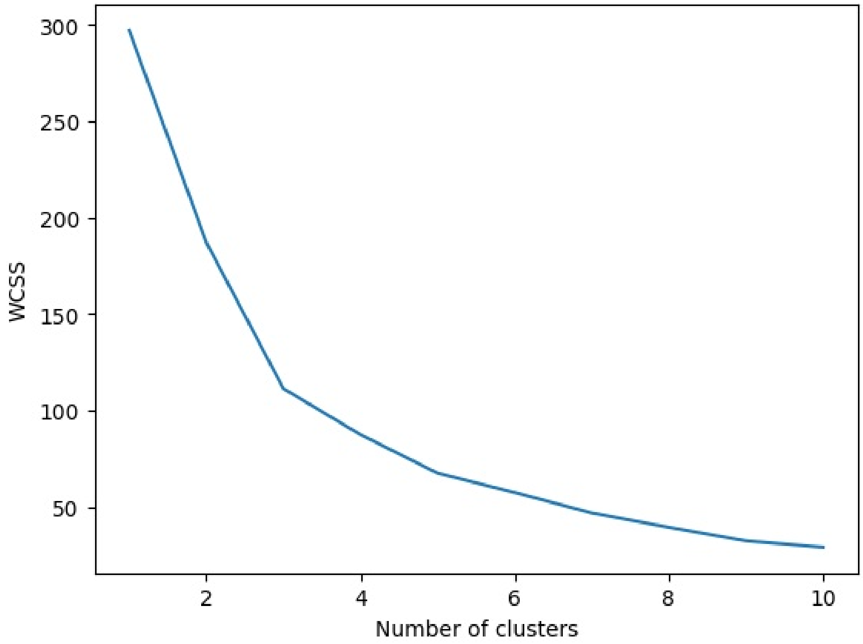
```
# Plot the PCA results
plt.scatter(X_pca[:, 0], X_pca[:, 2])
plt.xlabel('PC1')
plt.ylabel('PC3')
plt.show()
```



In [18]:

```
# Perform k-means clustering
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()
```

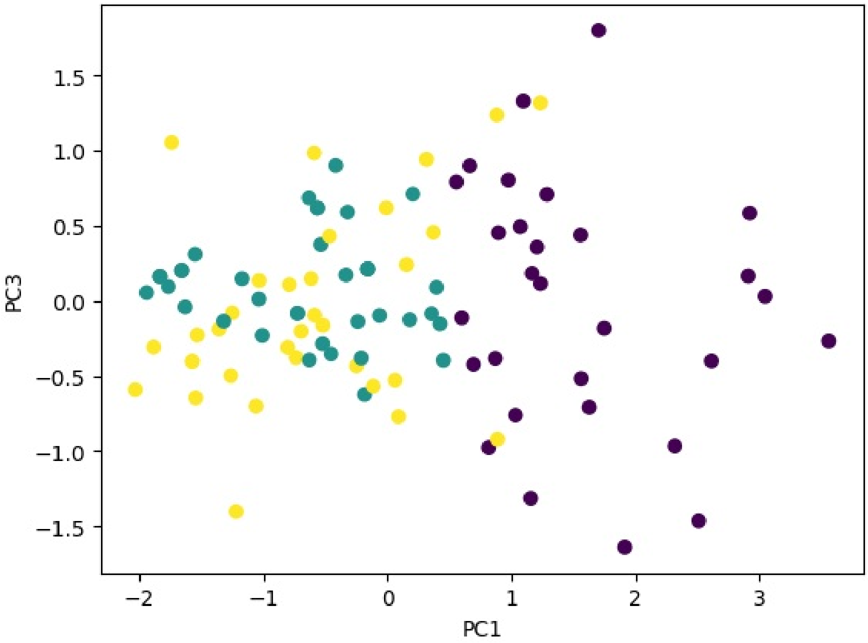
```
C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly
to suppress the warning
  warnings.warn(
C:\Users\aksha\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\_kmeans.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 suppress the warning
  warnings.warn(
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 suppress the warning
  warnings.warn(
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 suppress the warning
  warnings.warn(
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 suppress the warning
  warnings.warn(
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 suppress the warning
  warnings.warn(
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 suppress the warning
  warnings.warn(
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 suppress the warning
  warnings.warn(
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 suppress the warning
  warnings.warn(
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 suppress the warning
  warnings.warn(
```



In [19]:

```
# 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)
plt.scatter(X_pca[:, 0], X_pca[:, 2], c=y_kmeans)
plt.xlabel('PC1')
plt.ylabel('PC3')
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 suppress the warning
  warnings.warn(
```



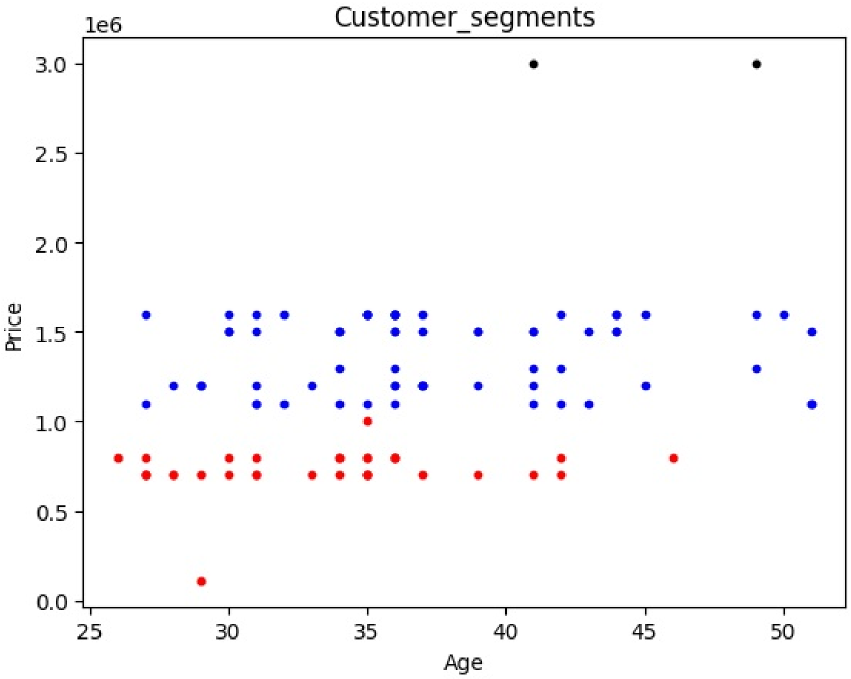
In [20]:

```
def plotseg(clus,Y, labels):
  plt.scatter(clus[Y==0,0], clus[Y==0,1], s=10, c='blue', label='Cluster 1')
  plt.scatter(clus[Y==1,0], clus[Y==1,1], s=10, c='red', label='Cluster 2')
  plt.scatter(clus[Y==2,0], clus[Y==2,1], s=10, c='black', label='Cluster 3')

  plt.title('Customer_segments')
  plt.xlabel(labels[0])
  plt.ylabel(labels[1])
  plt.show()
```

In [22]:

```
clus = df.loc[:,["Age","Price"]].values
kmeans = KMeans(n_clusters=3, init='k-means++')
Y = kmeans.fit_predict(clus)
plotseg(clus, Y, ["Age","Price"])
```

```
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 suppress the warning
  warnings.warn(
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js