Electric vehicle market segmentation analysis

February 22, 2024

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

[2]:
```
data=pd.read_csv("people_data.csv")
```

[3]:
```
data.head()
```

[3]:
```
   Age Profession Marrital Status      Education  No of Dependents  \
0   27   Salaried          Single  Post Graduate                 0
1   35   Salaried         Married  Post Graduate                 2
2   45   Business         Married       Graduate                 4
3   41   Business         Married  Post Graduate                 3
4   31   Salaried         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

     Make    Price
0     i20   800000
1    Ciaz  1000000
2  Duster  1200000
3    City  1200000
4     SUV  1600000
```

[4]:
```
data.shape
```

[4]:
```
(99, 13)
```

[5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Age               99 non-null     int64
 1   Profession        99 non-null     object
 2   Marrital Status   99 non-null     object
 3   Education         99 non-null     object
 4   No of Dependents  99 non-null     int64
 5   Personal loan     99 non-null     object
 6   House Loan        99 non-null     object
 7   Wife Working      99 non-null     object
 8   Salary            99 non-null     int64
 9   Wife Salary       99 non-null     int64
 10  Total Salary      99 non-null     int64
 11  Make              99 non-null     object
 12  Price             99 non-null     int64
dtypes: int64(6), object(7)
memory usage: 10.2+ KB
```

[6]:
```
data.describe()
```

[6]:
```
             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
```

[7]:
```
data.isnull().sum()
```

[7]:
```
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
```

[8]:
```
# There are no null values in the dataset
```

[9]:
```
data.dtypes
```

[9]:
```
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
```

[10]:
```
data.rename(columns={"Marrital Status": "Marital Status"}, inplace=True)
```

[11]:
```
data.columns
```

[11]:
```
Index(['Age', 'Profession', 'Marital Status', 'Education', 'No of Dependents',
       'Personal loan', 'House Loan', 'Wife Working', 'Salary', 'Wife Salary',
       'Total Salary', 'Make', 'Price'],
      dtype='object')
```

[12]:
```
# There was a spelling mistake in the column marital status, hence it is renamed
```

[13]:
```
data.rename(columns={'Personal loan':'Car_Loan'},inplace=True)
data.rename(columns={'Price':'Car_Price'},inplace=True)
```

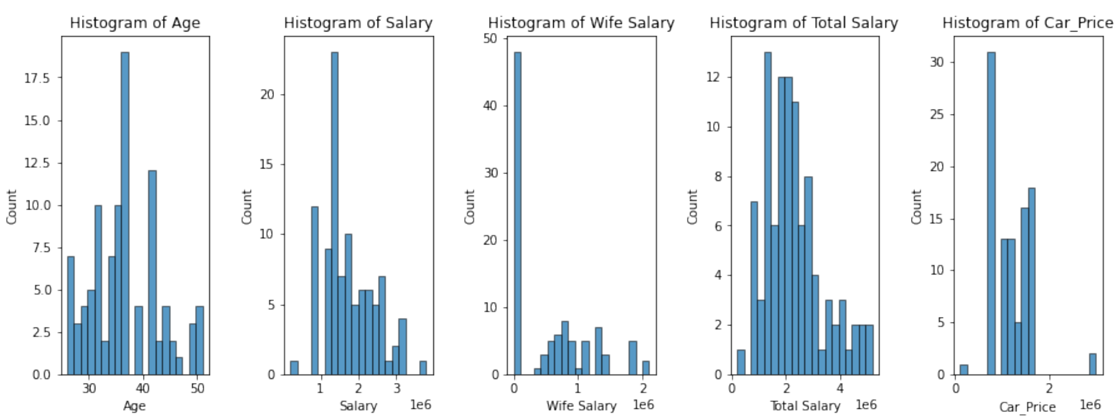
[14]:
```
data.head()
```

[14]:

```
   Age Profession Marital Status       Education  No of Dependents Car_Loan  \
0   27   Salaried         Single   Post Graduate                 0      Yes
1   35   Salaried        Married   Post Graduate                 2      Yes
2   45   Business        Married        Graduate                 4      Yes
3   41   Business        Married   Post Graduate                 3       No
4   31   Salaried        Married   Post Graduate                 2      Yes

  House Loan Wife Working   Salary  Wife Salary  Total Salary    Make  \
0         No           No   800000            0        800000     i20
1        Yes          Yes  1400000       600000       2000000    Ciaz
2        Yes           No  1800000            0       1800000  Duster
3         No          Yes  1600000       600000       2200000    City
4         No          Yes  1800000       800000       2600000     SUV

   Car_Price
0     800000
1    1000000
2    1200000
3    1200000
4    1600000
```

[15]:

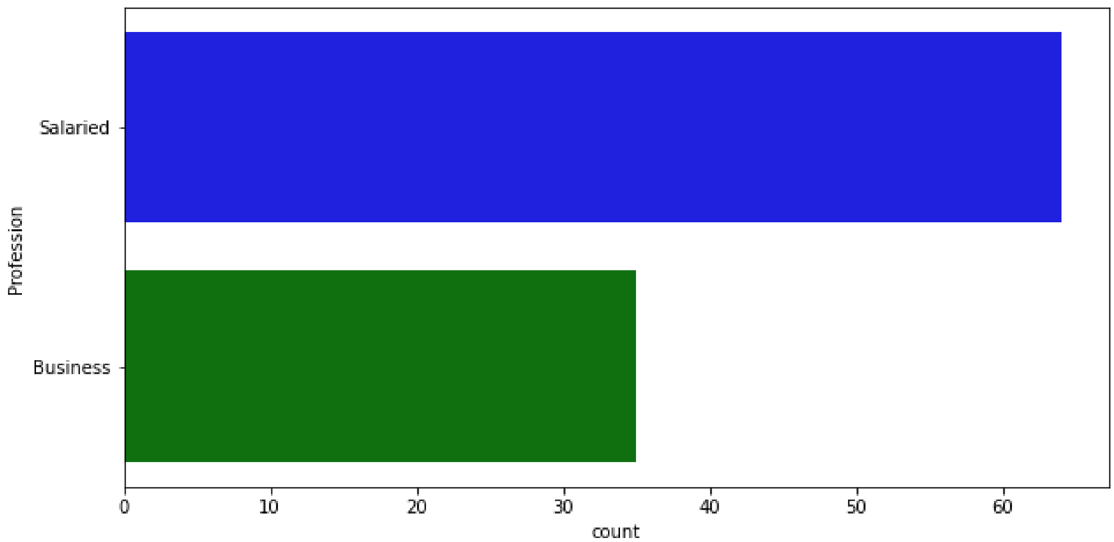
```
plt.figure(1, figsize=(15, 5))
n = 0
for x in ["Age", "Salary", "Wife Salary", "Total Salary", "Car_Price"]:
    n += 1
    plt.subplot(1, 5, n)
    plt.subplots_adjust(hspace=0.5, wspace=0.5)
    sns.histplot(data[x], bins=20)
    plt.title("Histogram of {}".format(x))
plt.show()
```



[16]:
```
profession = data['Profession'].value_counts()
profession
```

[16]:
```
Salaried    64
Business    35
Name: Profession, dtype: int64
```

[17]:
```
plt.figure(figsize=(10, 5))
sns.countplot(y="Profession", data=data, palette={'Salaried': 'blue',␣
↪'Business': 'green'})
plt.show()
```

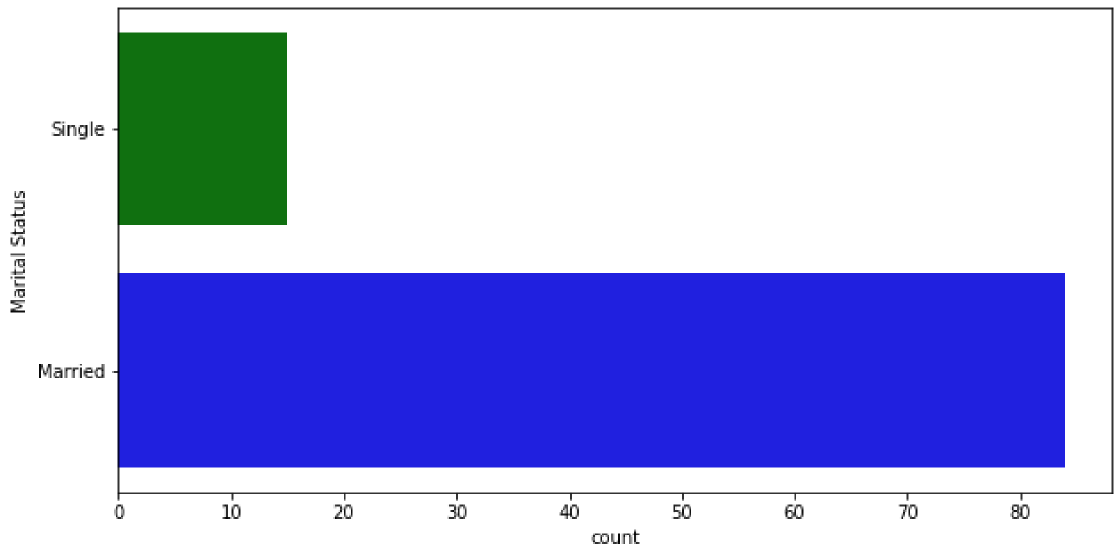


[18]:
```
# There are 64 salaried people and 35 business people-there are approximately 2␣
↪salaried people for every 1 business person.
```

[19]:
```
marital_status = data['Marital Status'].value_counts()
marital_status
```

[19]:
```
Married    84
Single     15
Name: Marital Status, dtype: int64
```

[20]:
```
plt.figure(figsize=(10, 5))
sns.countplot(y="Marital Status", data=data, palette={'Married': 'blue',␣
↪'Single': 'green'})
plt.show()
```

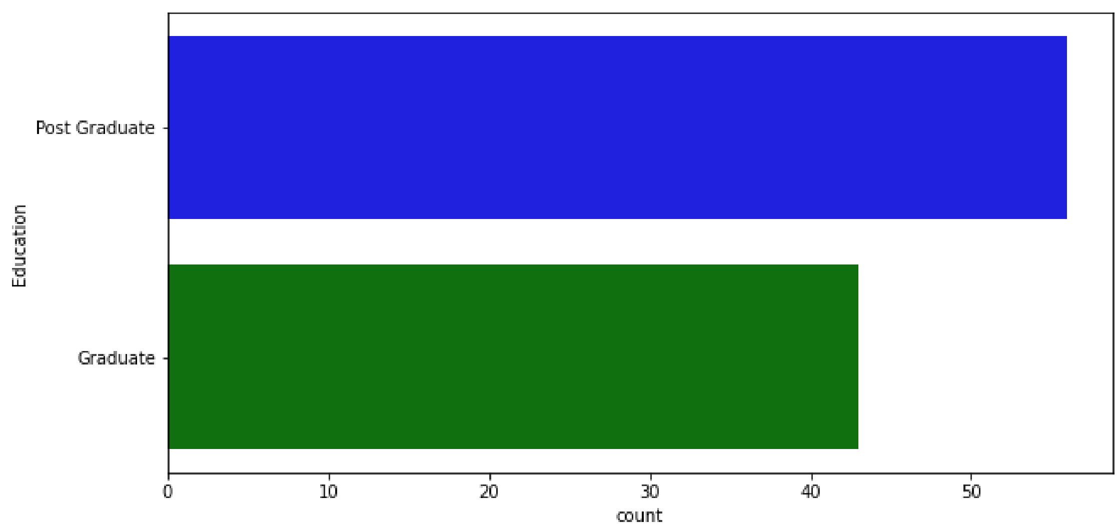


[21]:
```
# The number of married people are way more than the unmarried
```

[22]:
```
education = data['Education'].value_counts()
education
```

[22]:
```
Post Graduate    56
Graduate         43
Name: Education, dtype: int64
```

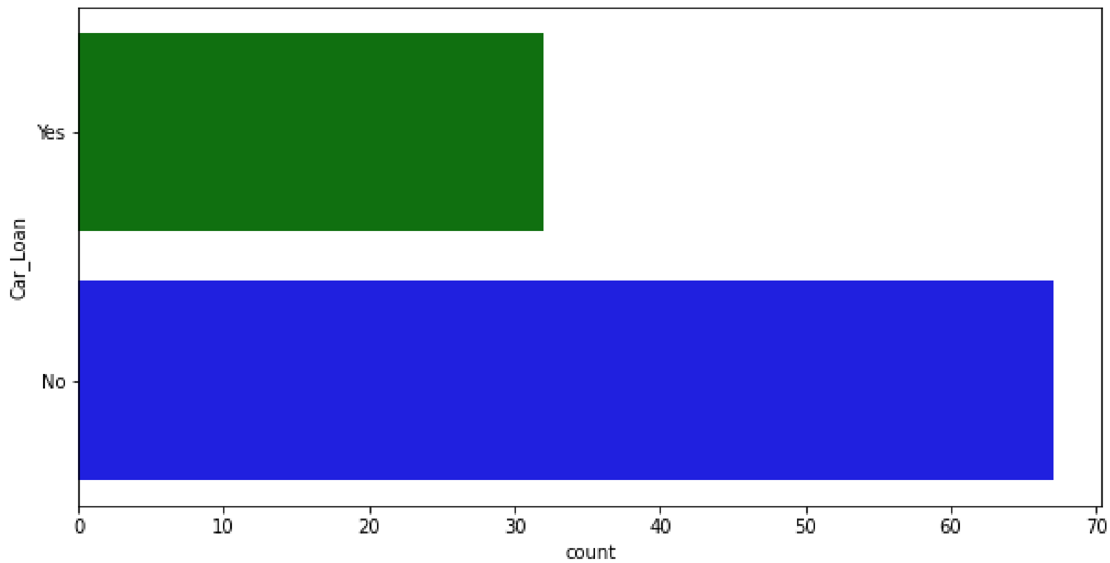
[23]:
```
plt.figure(figsize=(10,5))
sns.countplot(y="Education", data=data, palette= {"Post Graduate":"blue",␣
↪"Graduate":"green"})
plt.show()
```



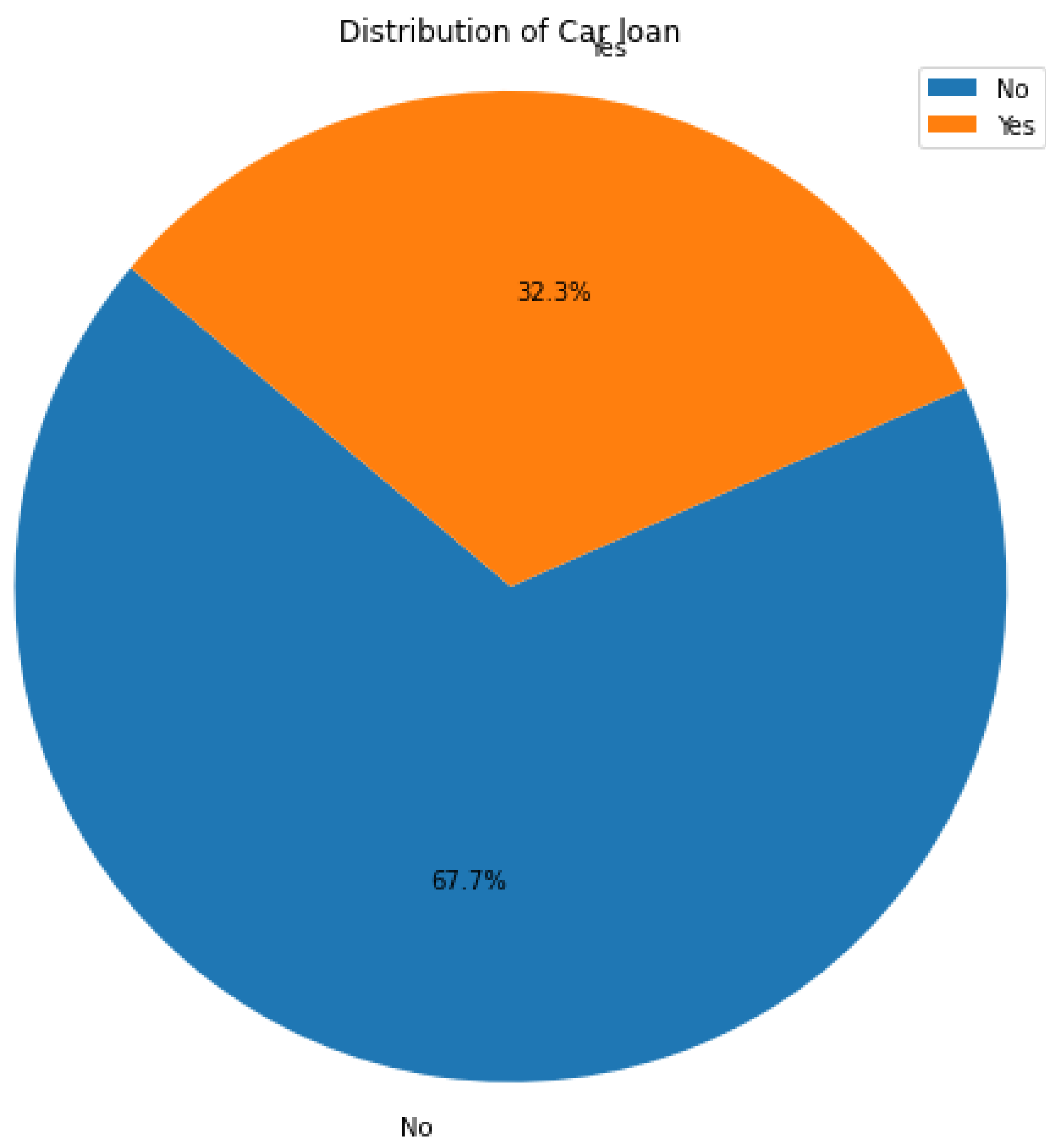
[24]:
```
loan_car = data["Car_Loan"].value_counts()
loan_car
```

[24]:
```
No     67
Yes    32
Name: Car_Loan, dtype: int64
```

[25]:
```
plt.figure(figsize=(10,5))
sns.countplot(y="Car_Loan", data= data, palette= {"Yes":"green", "No":"blue"})
plt.show()
```



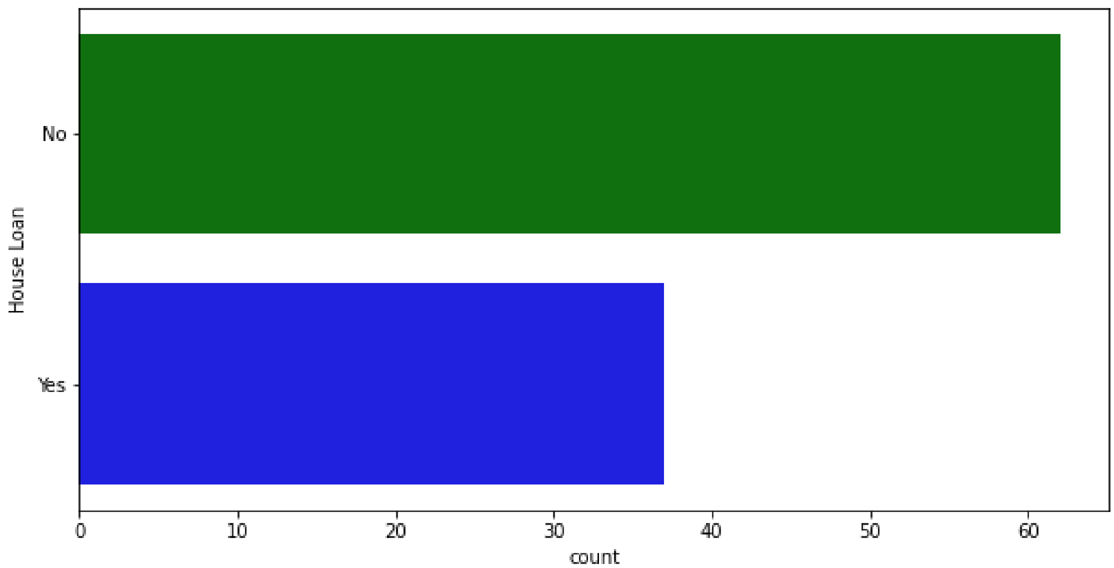
[26]:
```
personal_loan_count = data['Car_Loan'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(personal_loan_count , labels=personal_loan_count .index, autopct='%1.
 ↪1f%%', startangle=140)
plt.title('Distribution of Car loan')
plt.legend(loc='upper right')
plt.axis('equal')
plt.show()
```



[27]:
```
loan_house= data["House Loan"].value_counts()
loan_house
```

[27]:
```
No     62
Yes    37
Name: House Loan, dtype: int64
```

[28]:
```
plt.figure(figsize=(10,5))
sns.countplot(y="House Loan", data=data, palette={"No":"green","Yes":"blue"})
plt.show()
```



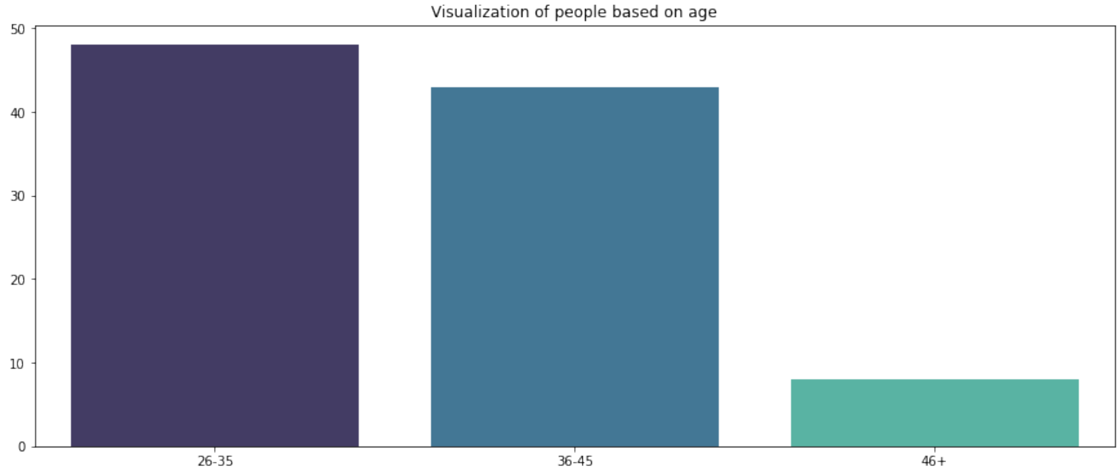
[29]:
```
# From the above countplots and pie chart it is clear that there are more␣
↪number of people with house loans as well as car loans.
```

[30]:
```
data["Age"].agg(["min", "max"])
```

[30]:
```
min    26
max    51
Name: Age, dtype: int64
```

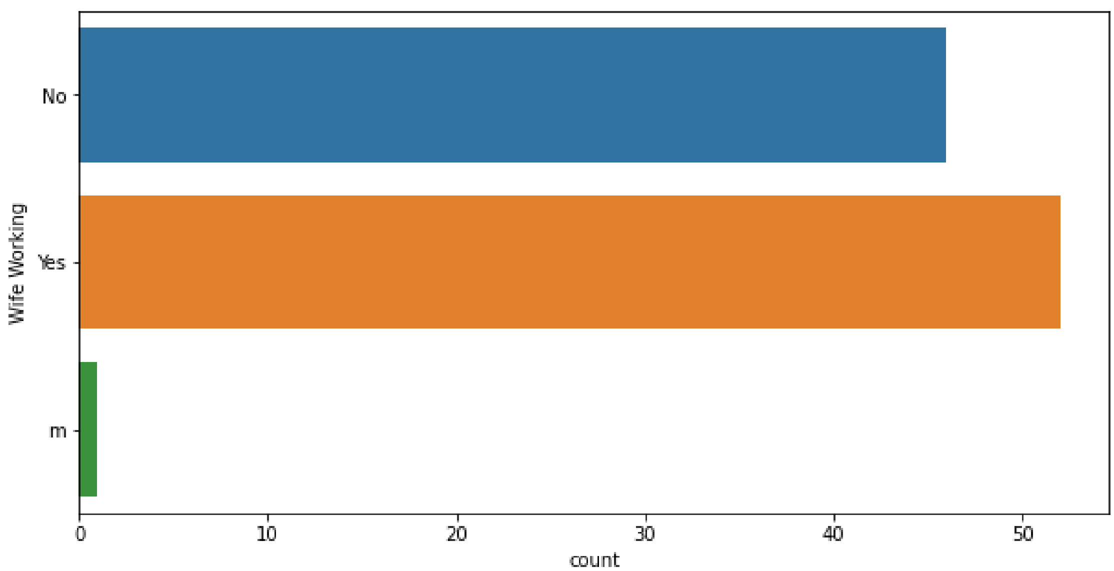
[31]:
```
age_26_35 = data.Age[(data.Age >= 26) & (data.Age <= 35)]
age_36_45 = data.Age[(data.Age >= 36) & (data.Age <= 45)]
age_above_46 = data.Age[(data.Age > 45)]
```

[32]:
```
age_x = [ "26-35", "36-45", "46+"]
age_y = [len(age_26_35.values), len(age_36_45.values), len(age_above_46.values)]
plt.figure(figsize = (15,6))
sns.barplot(x= age_x, y = age_y, palette= "mako")
plt.title("Visualization of people based on age")
plt.xlabel=("Age")
plt.ylabel=("Number of customers")
plt.show()
```




```
[33]: # There are more people in the age range of 26-35 and people of age 46 or more
      ↪is quite less
```

```
[34]: plt.figure(figsize=(10,5))
      sns.countplot(y="Wife Working", data=data )
      plt.show()
```




```
[35]: # From this output we can see that there is an undesirable entry called m which
      ↪needs to be removed
```

```
[36]: data.loc[data["Wife Working"]=="m"]
```

[36]:
```
    Age Profession Marital Status Education  No of Dependents Car_Loan  \
11   35   Salaried        Married  Graduate                 4      Yes

   House Loan Wife Working   Salary  Wife Salary  Total Salary    Make  \
11        Yes            m  1400000            0       1400000  Baleno

    Car_Price
11     700000
```

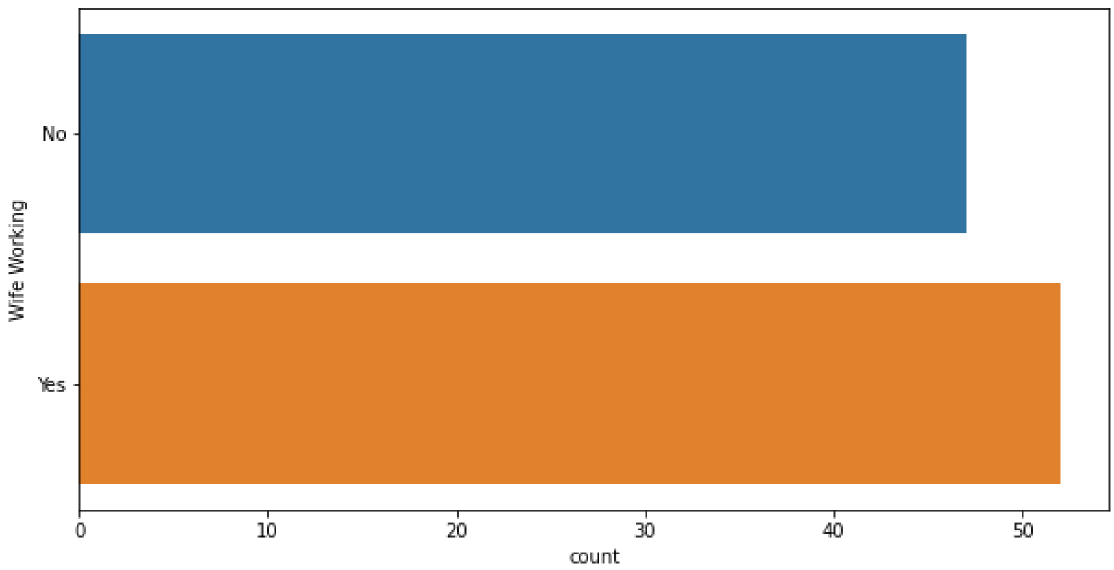
[37]:
```
data=data.replace(to_replace="m", value="No")
```

[38]:
```
row_11 = data.iloc[11]
print(row_11)
```

```
Age                        35
Profession           Salaried
Marital Status        Married
Education            Graduate
No of Dependents            4
Car_Loan                  Yes
House Loan                Yes
Wife Working               No
Salary                1400000
Wife Salary                 0
Total Salary          1400000
Make                   Baleno
Car_Price              700000
Name: 11, dtype: object
```

[39]:
```
# the letter m in 11th row is replaced with No as the column wife salary is␣
 ↪zero.
```

[40]:
```
plt.figure(figsize=(10,5))
sns.countplot(y="Wife Working", data=data)
plt.show()
```

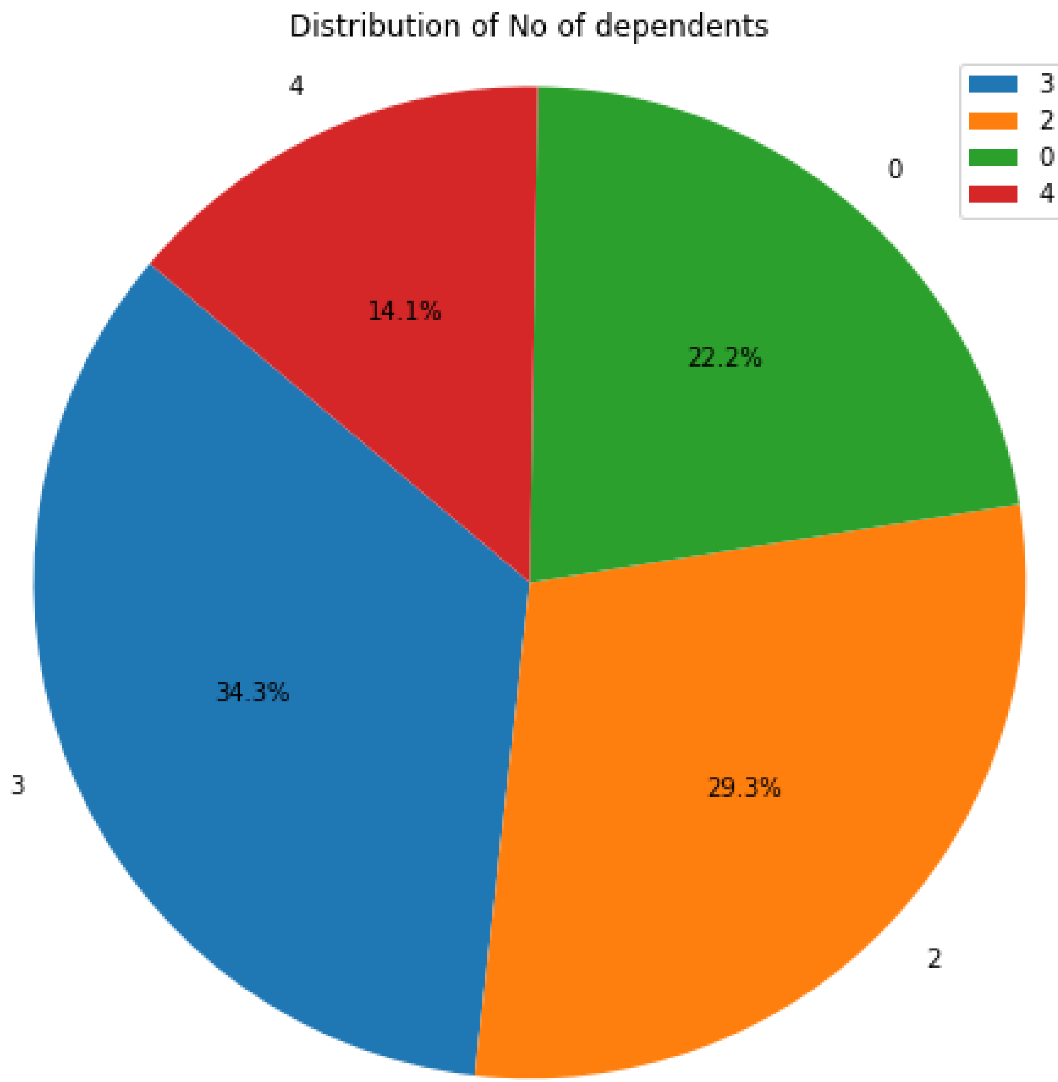


[41]:
```
# The number of working women are more than non-working women.
```

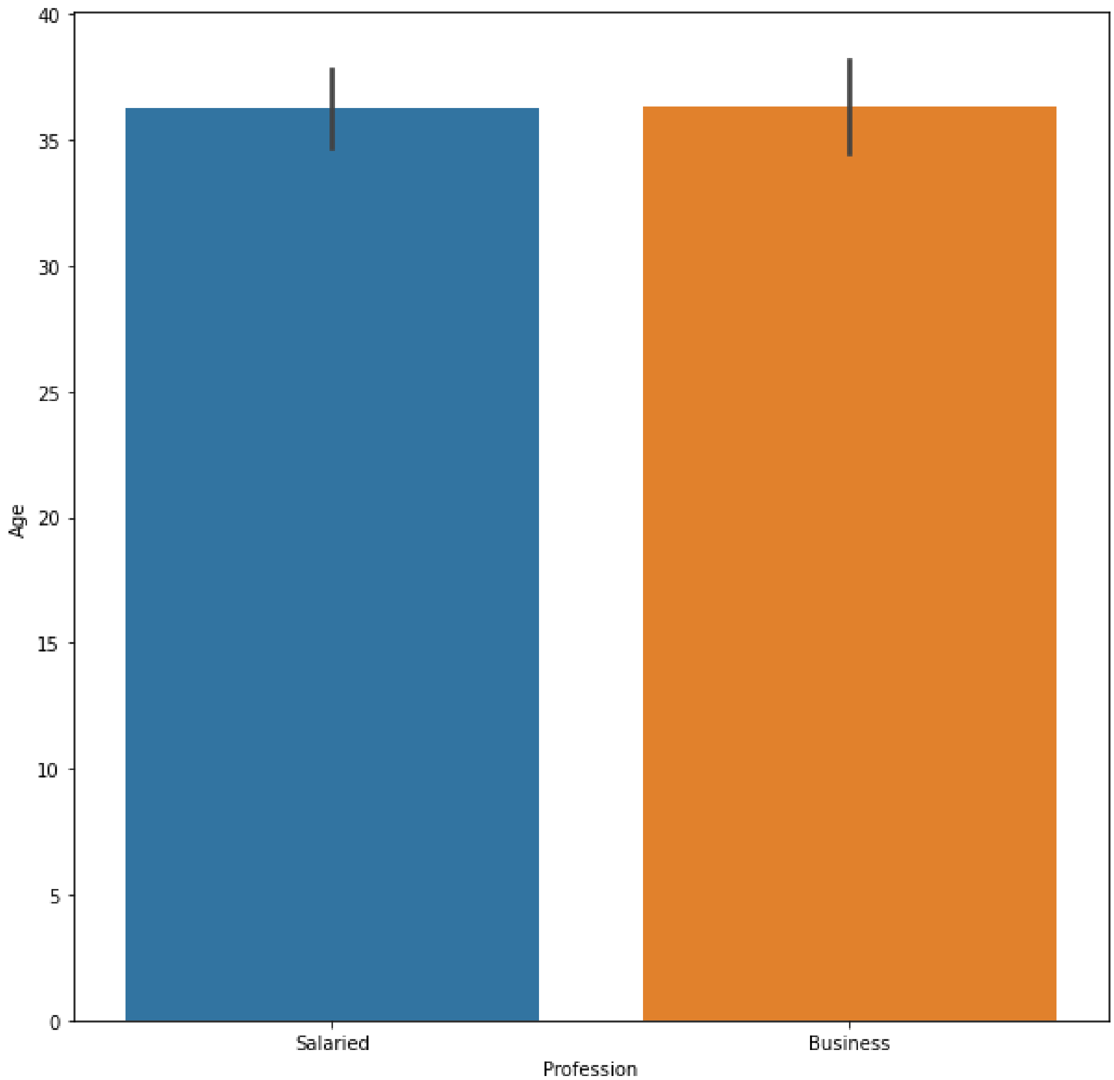
[42]:
```
no_of_dependents= data["No of Dependents"].value_counts()
no_of_dependents
```

[42]:
```
3    34
2    29
0    22
4    14
Name: No of Dependents, dtype: int64
```

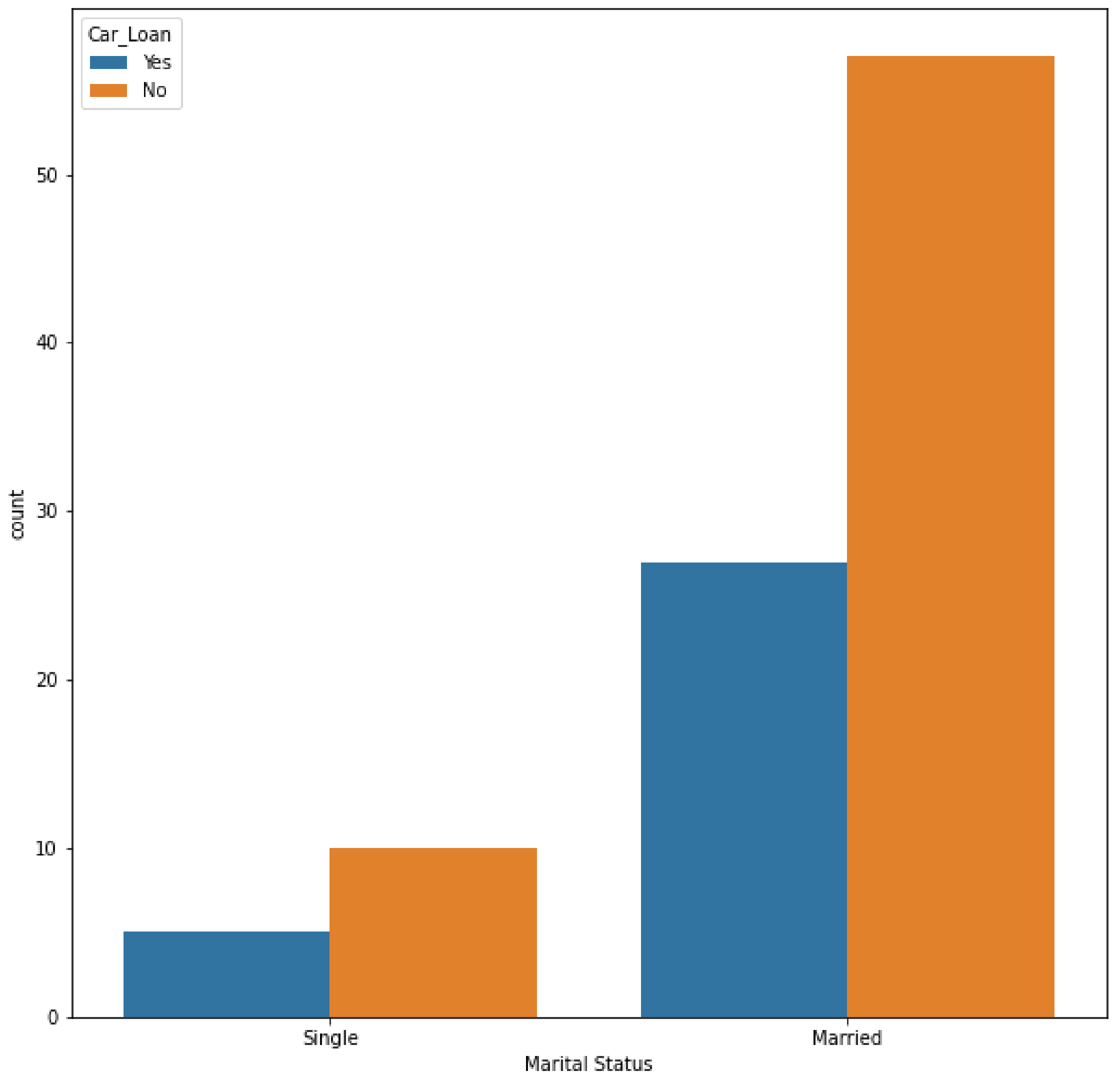
[43]:
```
plt.figure(figsize=(8, 8))
plt.pie(no_of_dependents , labels=no_of_dependents .index, autopct='%1.1f%%',␣
 ↪startangle=140)
plt.title('Distribution of No of dependents')
plt.legend(loc='upper right')
plt.axis('equal')
plt.show()
```



[44]:
```
plt.figure(figsize=(10, 10))
sns.barplot(x='Profession',y='Age',data=data)
plt.show()
```

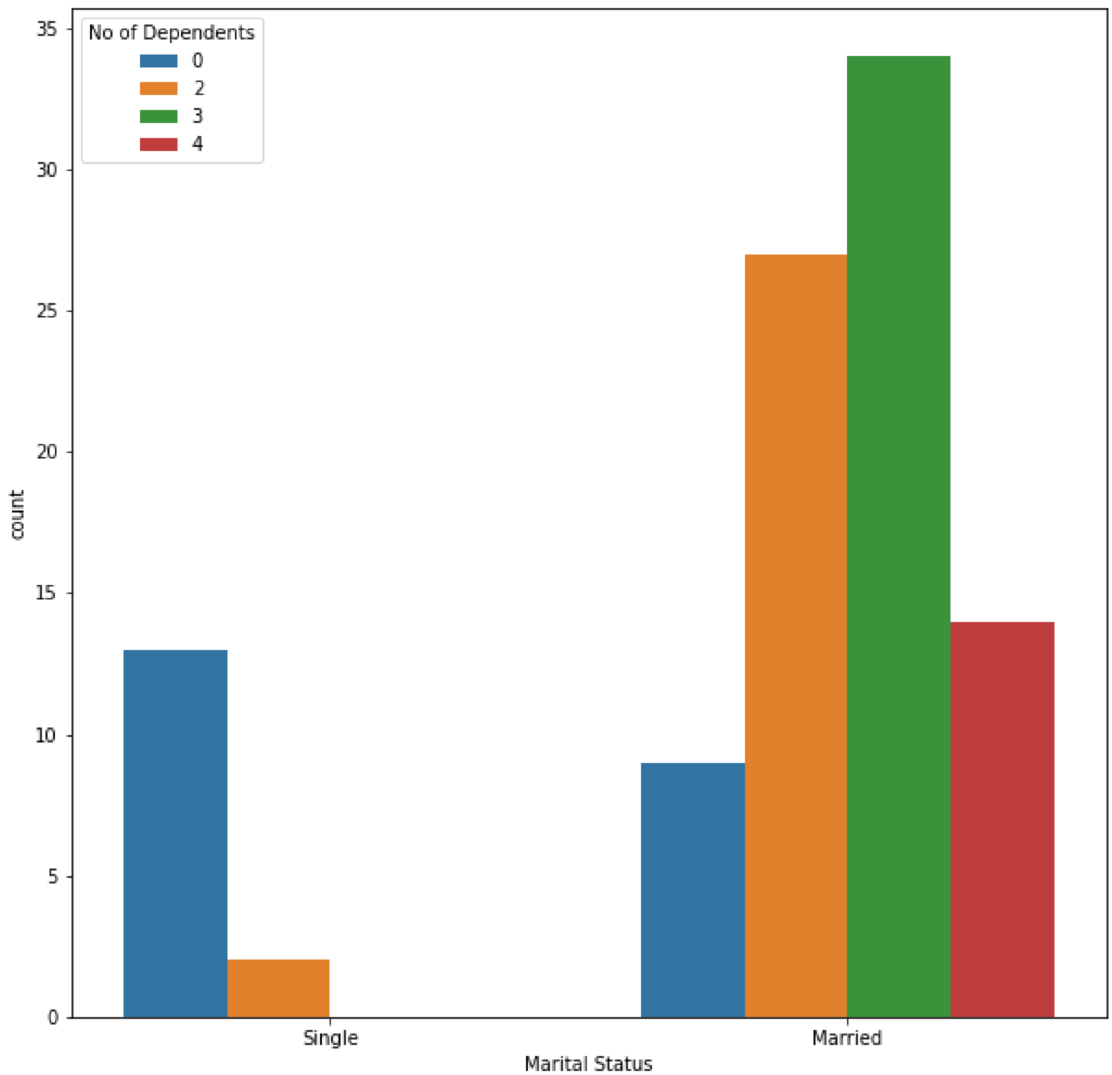


[45]:
```
plt.figure(figsize=(10, 10))
sns.countplot(x='Marital Status',hue='Car_Loan',data=data)
plt.show()
```



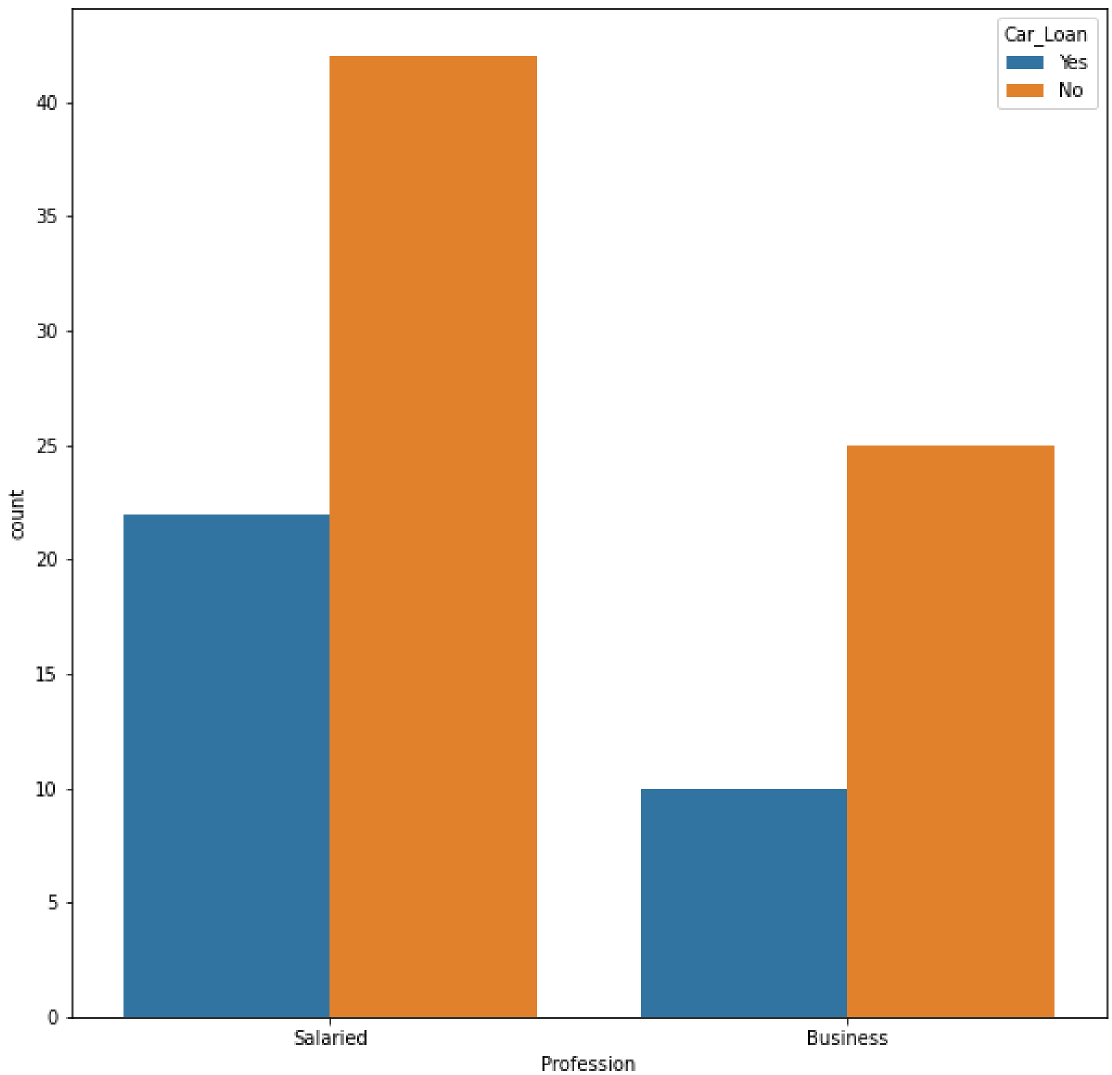
[46]:
```
# The percentage of people not opting for loan is more in both the categories.␣
↪But the percentage of married people looking for loans is higher than␣
↪unmarried people
```

[47]:
```
plt.figure(figsize=(10, 10))
sns.countplot(x='Marital Status',hue='No of Dependents',data=data)
plt.show()
```



[48]:
```
# Married people having more no of dependednts compared to the singles.
```

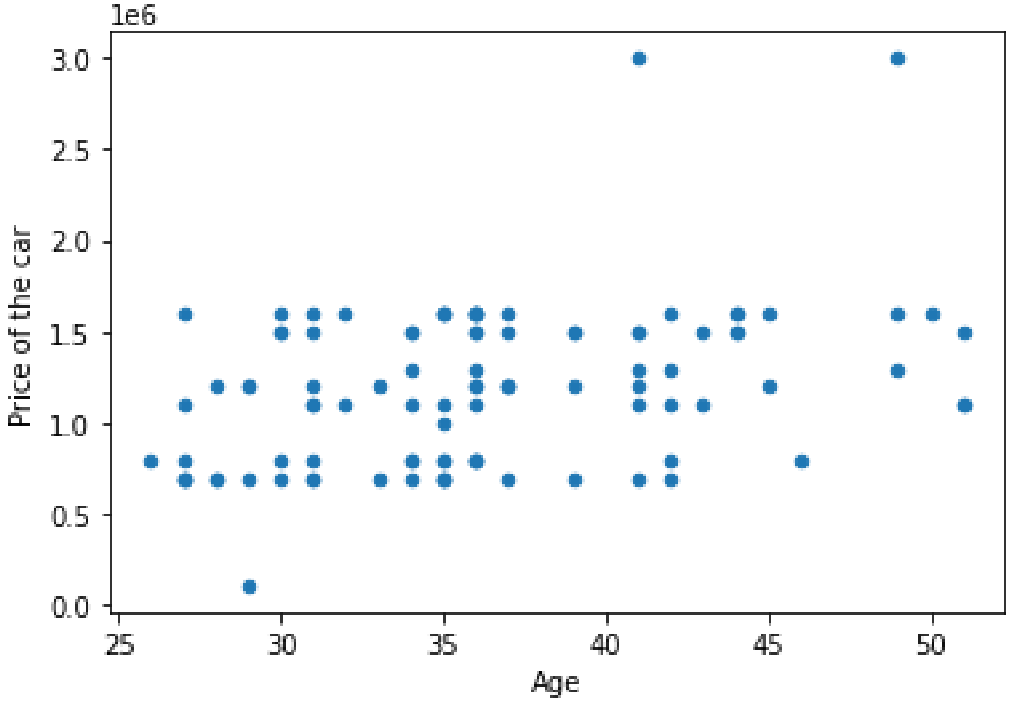
[49]:
```
plt.figure(figsize=(10, 10))
sns.countplot(x='Profession',hue='Car_Loan',data=data)
plt.show()
```



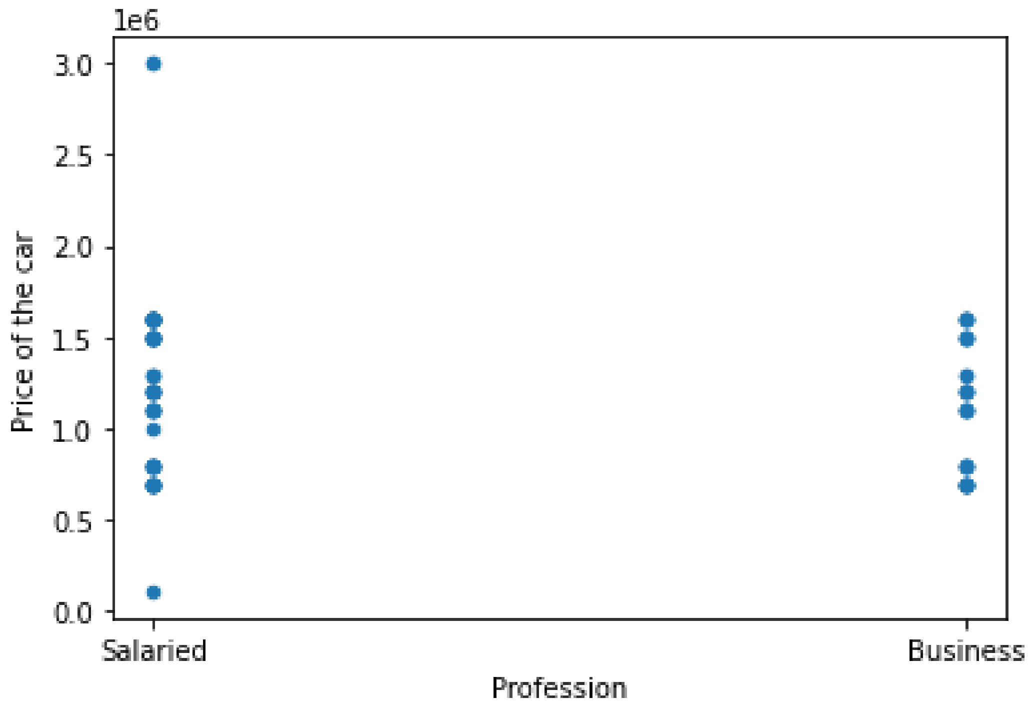

```
[50]: # But the percentage of salaried people looking for loans is higher than␣
      ↪businessman.
```

```
[51]: scatter_plot=data.plot(kind='scatter',x='Total Salary',y='Car_Price')
      scatter_plot.set_xlabel('Total Salary')
      scatter_plot.set_ylabel('Price of the car')
      plt.show()
```

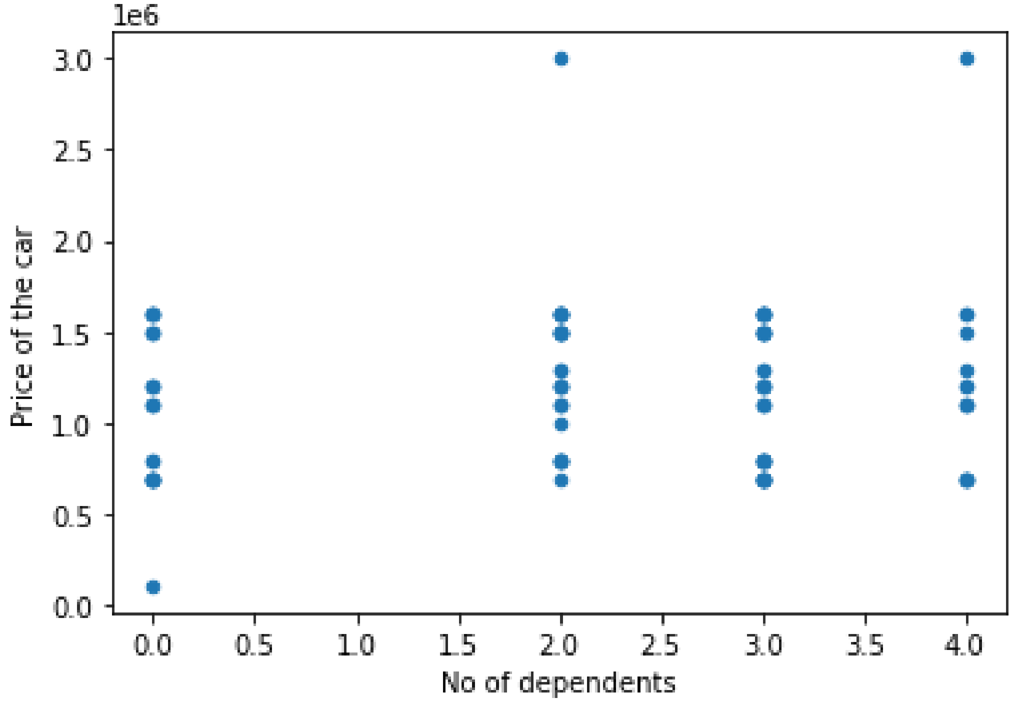
[52]:
```
scatter_plot=data.plot(kind='scatter',x='Age',y='Car_Price')
scatter_plot.set_xlabel('Age')
scatter_plot.set_ylabel('Price of the car')
plt.show()
```



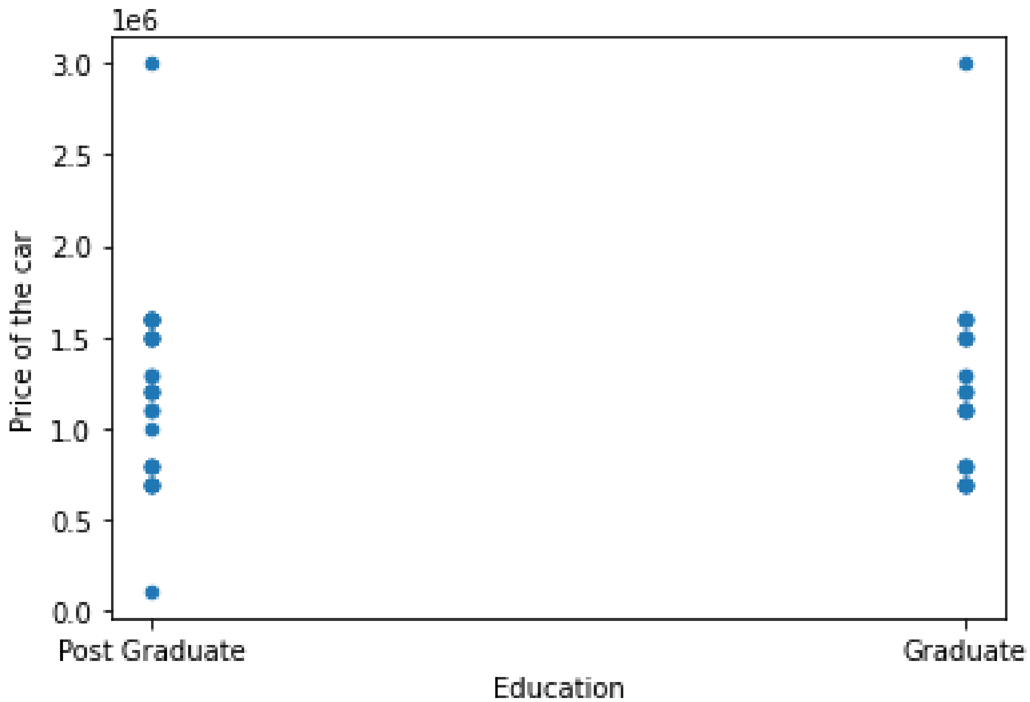
[53]:
```
scatter_plot=data.plot(kind='scatter',x='Profession',y='Car_Price')
scatter_plot.set_xlabel('Profession')
scatter_plot.set_ylabel('Price of the car')
plt.show()
```



[54]:
```
scatter_plot=data.plot(kind='scatter',x='No of Dependents',y='Car_Price')
scatter_plot.set_xlabel('No of dependents')
scatter_plot.set_ylabel('Price of the car')
plt.show()
```

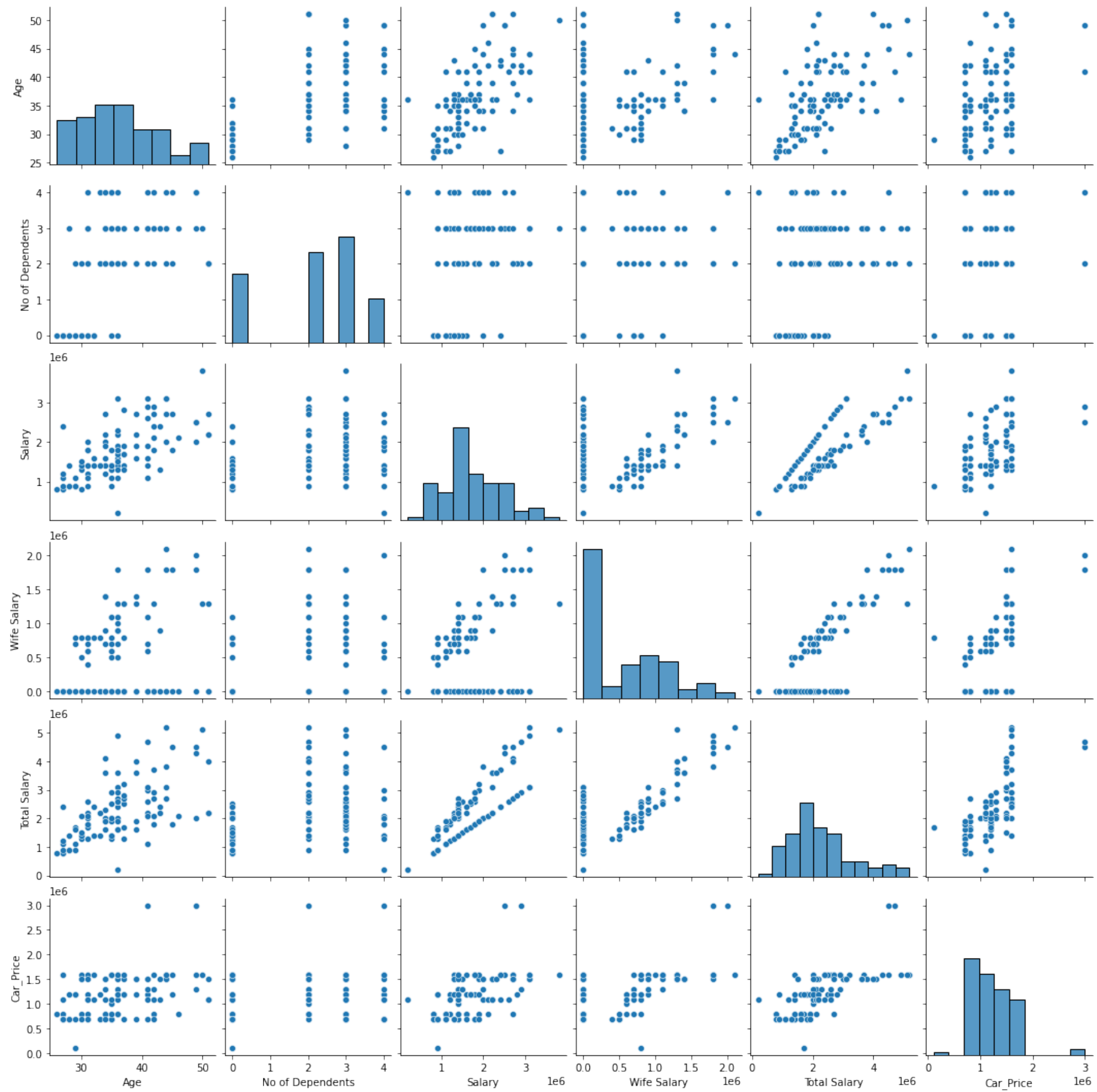


[55]:
```
scatter_plot=data.plot(kind='scatter',x='Education',y='Car_Price')
scatter_plot.set_xlabel('Education')
scatter_plot.set_ylabel('Price of the car')
plt.show()
```



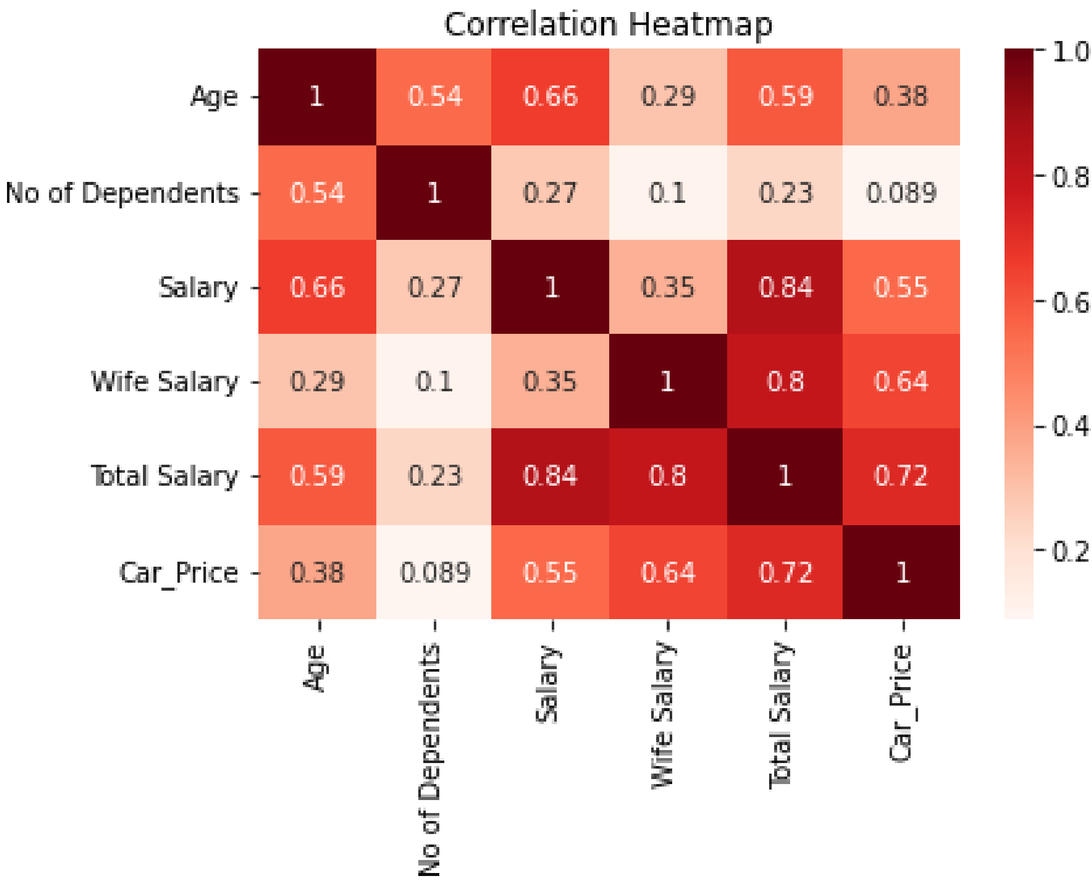
[56]:
```
sns.pairplot(data)
```

[56]: <seaborn.axisgrid.PairGrid at 0x7f3a4147b4f0>



[57]:
```
data_corr=data.corr()
sns.heatmap(data_corr,cmap='Reds',annot=True)
plt.title('Correlation Heatmap')
plt.show()
```

```
/tmp/ipykernel_378/429290499.py:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  data_corr=data.corr()
```



[79]:
```
# CLUSTERING
```

[59]:
```
data.head()
```

[59]:
```
   Age Profession Marital Status      Education  No of Dependents Car_Loan  \
0   27   Salaried         Single  Post Graduate                 0      Yes
1   35   Salaried        Married  Post Graduate                 2      Yes
2   45   Business        Married       Graduate                 4      Yes
3   41   Business        Married  Post Graduate                 3       No
4   31   Salaried        Married  Post Graduate                 2      Yes

  House Loan Wife Working   Salary  Wife Salary  Total Salary    Make  \
0         No           No   800000            0        800000     i20
1        Yes          Yes  1400000       600000       2000000    Ciaz
2        Yes           No  1800000            0       1800000  Duster
3         No          Yes  1600000       600000       2200000    City
4         No          Yes  1800000       800000       2600000     SUV

   Car_Price
```

```
0     800000
1    1000000
2    1200000
3    1200000
4    1600000
```

[60]:
```
columns_to_drop=['House Loan', 'Wife Working', 'Wife Salary', 'Make']
clustering_data = data.drop(columns=columns_to_drop)
clustering_data.head()
```

[60]:
```
   Age Profession Marital Status      Education  No of Dependents Car_Loan  \
0   27   Salaried         Single  Post Graduate                 0      Yes
1   35   Salaried        Married  Post Graduate                 2      Yes
2   45   Business        Married       Graduate                 4      Yes
3   41   Business        Married  Post Graduate                 3       No
4   31   Salaried        Married  Post Graduate                 2      Yes

    Salary  Total Salary  Car_Price
0   800000        800000     800000
1  1400000       2000000    1000000
2  1800000       1800000    1200000
3  1600000       2200000    1200000
4  1800000       2600000    1600000
```

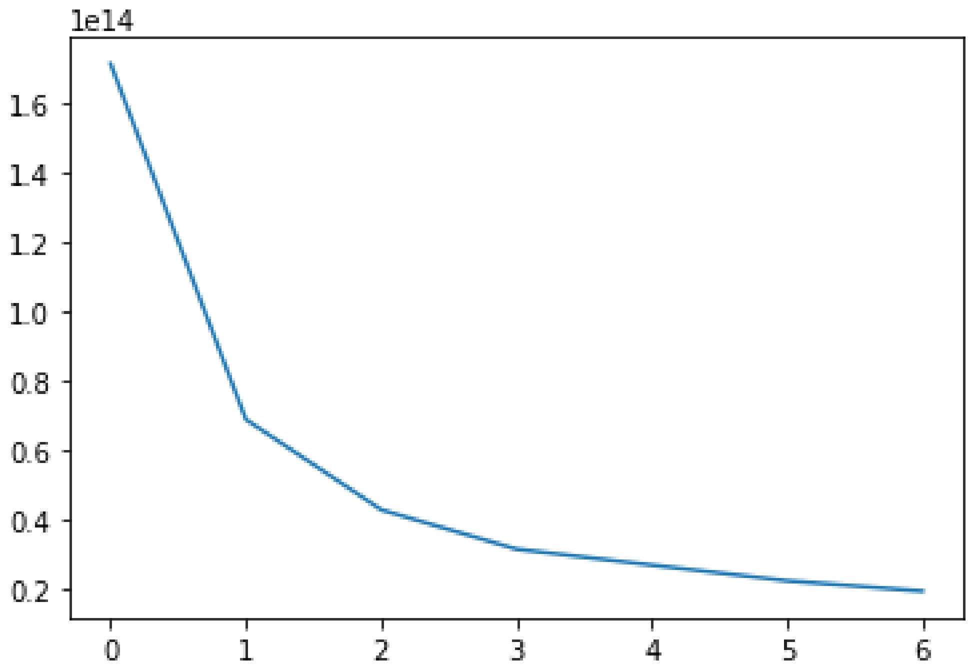
[61]:
```
!pip install kmodes
from kmodes.kprototypes import KPrototypes
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: kmodes in ./.local/lib/python3.10/site-packages
(0.12.2)
Requirement already satisfied: numpy>=1.10.4 in /usr/local/lib/python3.10/site-
packages (from kmodes) (1.23.5)
Requirement already satisfied: scikit-learn>=0.22.0 in
/usr/local/lib/python3.10/site-packages (from kmodes) (1.3.1)
Requirement already satisfied: scipy>=0.13.3 in /usr/local/lib/python3.10/site-
packages (from kmodes) (1.9.3)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/site-
packages (from kmodes) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/site-packages (from scikit-learn>=0.22.0->kmodes)
(3.1.0)

[notice] A new release of pip is
available: 23.3 -> 24.0
[notice] To update, run:
pip install --upgrade pip
```

[62]:
```
cluster_features = list(clustering_data.columns)
clusters = clustering_data[cluster_features].values
```

[63]:
```
# Finding optimal number of clusters for KPrototypes

cost = []
for num_clusters in list(range(1,8)):
    kproto = KPrototypes(n_clusters=num_clusters, init='Cao')
    kproto.fit_predict(clusters, categorical=[1,2,3,5])
    cost.append(kproto.cost_)

plt.plot(cost)
```

[63]:
```
[<matplotlib.lines.Line2D at 0x7f3a3fb85270>]
```



[64]:
```
cost
```

[64]:
```
[171448752145958.78,
 69038756991260.4,
 43067483317073.38,
 31741612025561.22,
 27151033164178.4,
 22623629761379.164,
 19648779937913.5]
```

[65]:

```
# fitting data to clusters

kproto = KPrototypes(n_clusters=2, verbose=2,max_iter=20)
cluster = kproto.fit_predict(clusters, categorical=[1,2,3,5])
```

```
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 1, iteration: 1/20, moves: 3, ncost: 69086465524510.375
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 2, iteration: 1/20, moves: 1, ncost: 69086465524510.375
Run: 2, iteration: 2/20, moves: 0, ncost: 69086465524510.375
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 3, iteration: 1/20, moves: 2, ncost: 69038756991260.4
Run: 3, iteration: 2/20, moves: 0, ncost: 69038756991260.4
Init: initializing centroids
Init: initializing clusters
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 4, iteration: 1/20, moves: 8, ncost: 69112824603304.516
Run: 4, iteration: 2/20, moves: 1, ncost: 69038756991260.4
Run: 4, iteration: 3/20, moves: 0, ncost: 69038756991260.4
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 5, iteration: 1/20, moves: 15, ncost: 71237403504718.31
Run: 5, iteration: 2/20, moves: 6, ncost: 69224234927689.06
Run: 5, iteration: 3/20, moves: 2, ncost: 69038756991260.4
Run: 5, iteration: 4/20, moves: 0, ncost: 69038756991260.4
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 6, iteration: 1/20, moves: 14, ncost: 70694577025036.27
Run: 6, iteration: 2/20, moves: 5, ncost: 69224234927689.06
Run: 6, iteration: 3/20, moves: 2, ncost: 69038756991260.4
Run: 6, iteration: 4/20, moves: 0, ncost: 69038756991260.4
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 7, iteration: 1/20, moves: 20, ncost: 80276263048062.12
Run: 7, iteration: 2/20, moves: 16, ncost: 71237403504718.31
```

```
Run: 7, iteration: 3/20, moves: 6, ncost: 69224234927689.06
Run: 7, iteration: 4/20, moves: 2, ncost: 69038756991260.4
Run: 7, iteration: 5/20, moves: 0, ncost: 69038756991260.4
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 8, iteration: 1/20, moves: 23, ncost: 75728107622613.34
Run: 8, iteration: 2/20, moves: 10, ncost: 70694577025036.27
Run: 8, iteration: 3/20, moves: 5, ncost: 69224234927689.06
Run: 8, iteration: 4/20, moves: 2, ncost: 69038756991260.4
Run: 8, iteration: 5/20, moves: 0, ncost: 69038756991260.4
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 9, iteration: 1/20, moves: 16, ncost: 70227299447362.375
Run: 9, iteration: 2/20, moves: 4, ncost: 69224234927689.06
Run: 9, iteration: 3/20, moves: 2, ncost: 69038756991260.4
Run: 9, iteration: 4/20, moves: 0, ncost: 69038756991260.4
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run: 10, iteration: 1/20, moves: 36, ncost: 69779122783252.78
Run: 10, iteration: 2/20, moves: 3, ncost: 69224234927689.06
Run: 10, iteration: 3/20, moves: 2, ncost: 69038756991260.4
Run: 10, iteration: 4/20, moves: 0, ncost: 69038756991260.4
Best run was number 3
```

[66]:
```
# Appending the cluster data

clustering_data['Cluster'] = cluster
```

[67]:
```
#Average of car price in clustering data
clustering_data.Car_Price.mean()
```

[67]: 1194040.4040404041

[68]:
```
# Average cost of a car in first segment
clustering_data.Car_Price[clustering_data.Cluster==0].mean()
```

[68]: 1030142.8571428572

[69]:
```
clustering_data['Car_Price'][clustering_data.Cluster==1].max()
```

[69]: 3000000

[70]:
```
# Average cost of a car in second segment
clustering_data.Car_Price[clustering_data.Cluster==1].mean()
```

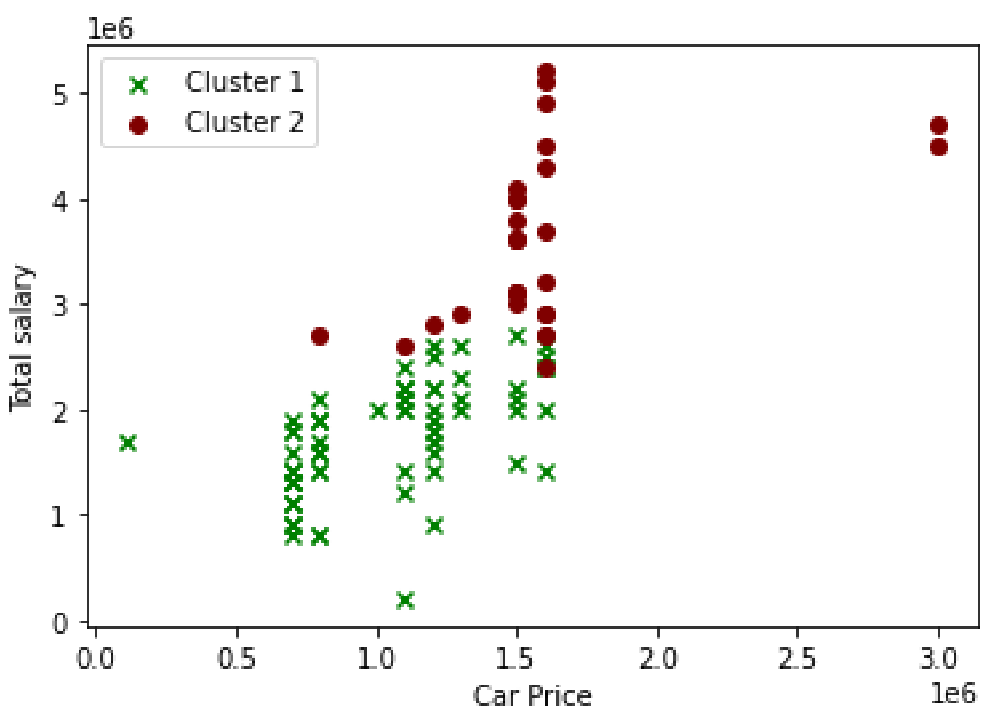
[70]: 1589655.1724137932

[71]:
```
clustering_data['Cluster'].value_counts(normalize=True) * 100
```

[71]:
```
0    70.707071
1    29.292929
Name: Cluster, dtype: float64
```

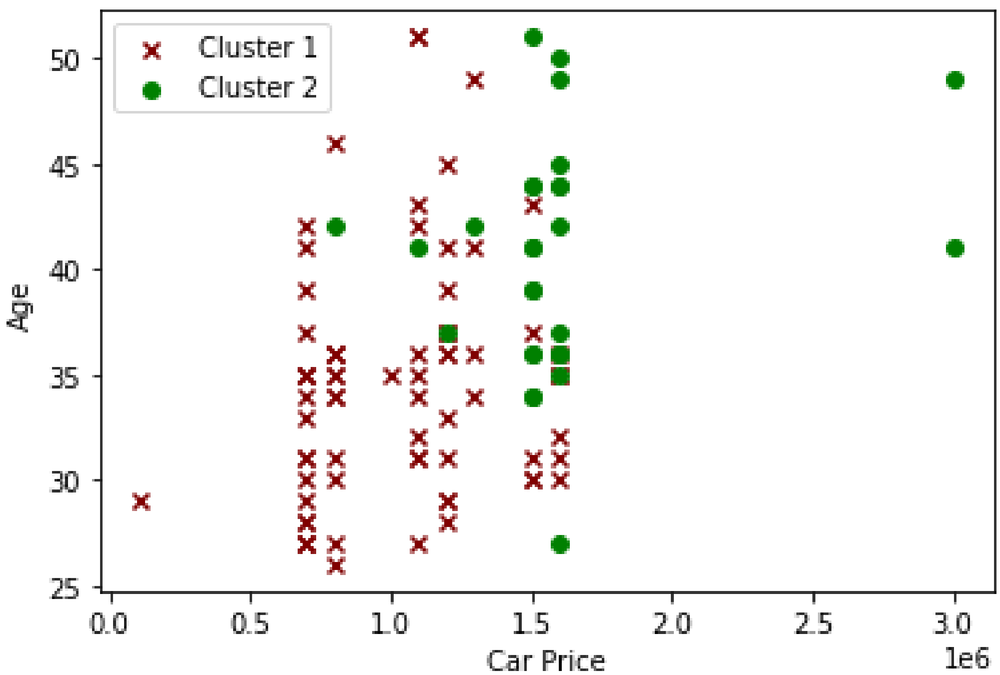
[72]:
```
# Seggregrating each cluster
Cluster_0 = clustering_data[clustering_data.Cluster==0]
Cluster_1 = clustering_data[clustering_data.Cluster==1]
```

[73]:
```
# plotting the effect of salary and ev price on cluster data

plt.scatter(Cluster_0.Car_Price, Cluster_0['Total Salary'],color='green',␣
 ↪marker = 'x', label = 'Cluster 1')
plt.scatter(Cluster_1.Car_Price, Cluster_1['Total Salary'],color='maroon',␣
 ↪label = 'Cluster 2')
plt.legend(loc="upper left")

plt.gca().set_xlabel('Car Price')
plt.gca().set_ylabel('Total salary')
plt.show()
```



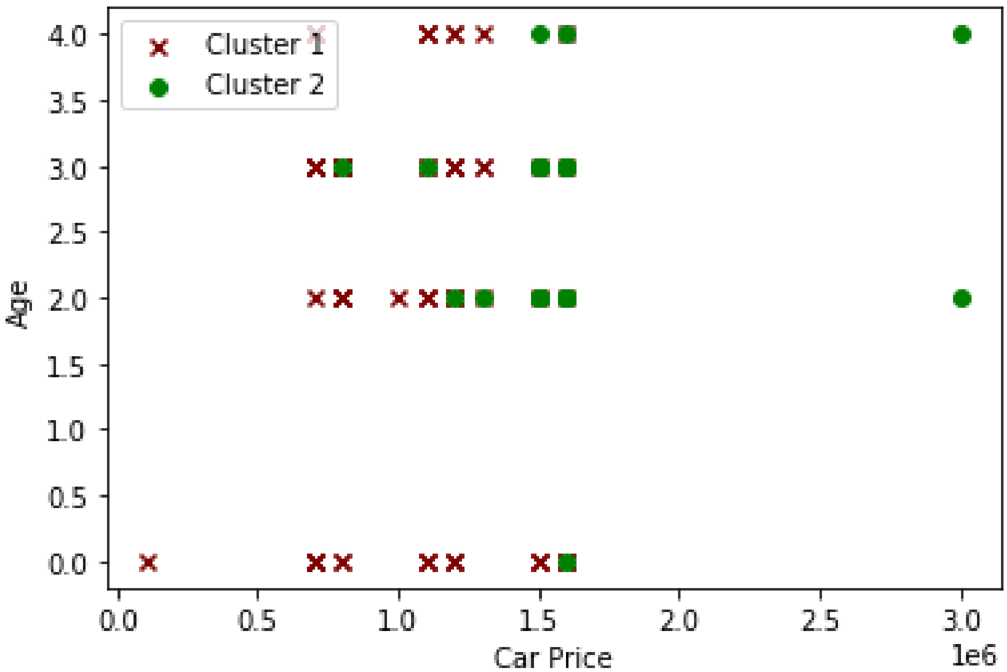
[74]:

```
plt.scatter(Cluster_0.Car_Price, Cluster_0['Age'],color='maroon', marker = 'x',␣
↪label = 'Cluster 1')
plt.scatter(Cluster_1.Car_Price, Cluster_1['Age'],color='green', label =␣
↪'Cluster 2')
plt.legend(loc = "upper left")

plt.gca().set_xlabel('Car Price')
plt.gca().set_ylabel('Age')
plt.show()
```



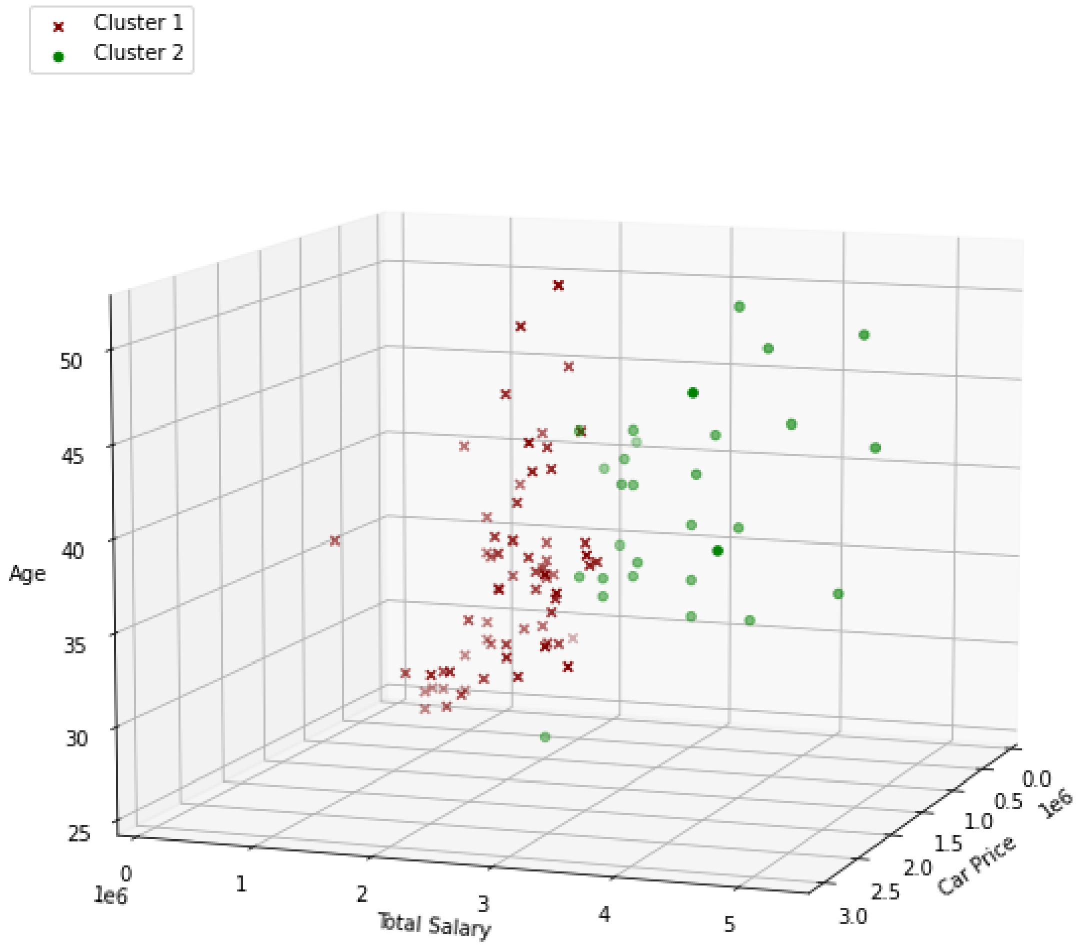
[75]:

```
plt.scatter(Cluster_0.Car_Price, Cluster_0['No of Dependents'],color='maroon',␣
↪marker = 'x', label = 'Cluster 1')
plt.scatter(Cluster_1.Car_Price, Cluster_1['No of Dependents'],color='green',␣
↪label = 'Cluster 2')
plt.legend(loc = "upper left")

plt.gca().set_xlabel('Car Price')
plt.gca().set_ylabel('Age')
plt.show()
```



[76]:
```
from mpl_toolkits.mplot3d import Axes3D
```

[77]:
```
# analyzing the influence of age

fig = plt.figure(figsize=(10,10))

ax = fig.add_subplot(111, projection='3d')

ax.scatter(Cluster_0.Car_Price, Cluster_0['Total Salary'], Cluster_0['Age'],␣
 ↪color='maroon', marker = 'x', label = 'Cluster 1')
ax.scatter(Cluster_1.Car_Price, Cluster_1['Total Salary'],Cluster_1['Age'],␣
 ↪color='green', label = 'Cluster 2')
plt.legend(loc = 'upper left')

ax.view_init(10, 20)

plt.gca().set_xlabel("Car Price")
plt.gca().set_ylabel("Total Salary")
ax.set_zlabel('Age')
plt.show()
```




```
[78]: # plotting influence of no.of dependents
fig = plt.figure(figsize=(8,8))

ax = fig.add_subplot(111, projection='3d')

ax.scatter(Cluster_0.Car_Price, Cluster_0['Total Salary'], Cluster_0['No of␣
 ↪Dependents'], color='maroon', marker = 'x', label = 'Cluster 1')
ax.scatter(Cluster_1.Car_Price, Cluster_1['Total Salary'],Cluster_1['No of␣
 ↪Dependents'], color='green', label = 'Cluster 2')
plt.legend(loc = 'upper left')

ax.view_init(10, 20)
```

```
plt.gca().set_xlabel("Car Price")
plt.gca().set_ylabel("Total Salary")
ax.set_zlabel('No_of Dependents')
plt.show()
```