Market Segmentation

Task done as part of FeyNN Labs Internship Analysing the Electric Vehicle market in India using Segmentation analysis for an Electric Vehicles Startup and coming up with a feasible strategy to enter the market, targeting the segments most likely to use Electric vehicles.

Authors: Vaibhav Rai, Angadi Abhinay, Ayush Soni,

Importing Important Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Data Preprocessing

In [2]:
```
# Importing consumer buying behavior study dataset
df = pd.read_csv('Indian automoble buying behavour study 1.0.csv')
df.head()
```

Out[2]:

	Age	Profession	Marrital Status	Education	No of Dependents	Personal loan	House Loan	Wife Working	Salary
0	27	Salaried	Single	Post Graduate	0	Yes	No	No	800000
1	35	Salaried	Married	Post Graduate	2	Yes	Yes	Yes	1400000
2	45	Business	Married	Graduate	4	Yes	Yes	No	1800000
3	41	Business	Married	Post Graduate	3	No	No	Yes	1600000
4	31	Salaried	Married	Post Graduate	2	Yes	No	Yes	1800000

In [4]:
```
df.shape
```

Out[4]: (99, 13)

In [5]:
```
df.info
```

Out[5]:
```
<bound method DataFrame.info of     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
..  ...        ...             ...            ...               ...
94   27   Business          Single       Graduate                 0
95   50   Salaried         Married  Post Graduate                 3
96   51   Business         Married       Graduate                 2
97   51   Salaried         Married  Post Graduate                 2
98   51   Salaried         Married  Post Graduate                 2

   Personal loan House Loan Wife Working   Salary  Wife Salary  Total Sal
ary  \
0            Yes         No           No   800000            0        800
000
1            Yes        Yes          Yes  1400000       600000       2000
000
2            Yes        Yes           No  1800000            0       1800
000
3             No         No          Yes  1600000       600000       2200
000
4            Yes         No          Yes  1800000       800000       2600
000
..           ...        ...          ...      ...          ...
...
```

In [5]:
```
df.info
```

Out[5]:
```
<bound method DataFrame.info of     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
..  ...        ...             ...            ...               ...
94   27   Business          Single       Graduate                 0
95   50   Salaried         Married  Post Graduate                 3
96   51   Business         Married       Graduate                 2
97   51   Salaried         Married  Post Graduate                 2
98   51   Salaried         Married  Post Graduate                 2

   Personal loan House Loan Wife Working   Salary  Wife Salary  Total Sal
ary  \
0             Yes         No           No   800000            0        800
000
1             Yes        Yes          Yes  1400000       600000       2000
000
2             Yes        Yes           No  1800000            0       1800
000
3              No         No          Yes  1600000       600000       2200
000
4             Yes         No          Yes  1800000       800000       2600
000
..            ...        ...          ...      ...          ...
...
94             No         No           No  2400000            0       2400
000
95             No         No          Yes  3800000      1300000       5100
000
96            Yes        Yes           No  2200000            0       2200
000
97             No         No          Yes  2700000      1300000       4000
000
98            Yes        Yes           No  2200000            0       2200
000

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

Out[6]:

	Age	No of Dependents	Salary	Wife Salary	Total Salary	Price
count	99.000000	99.000000	9.900000e+01	9.900000e+01	9.900000e+01	9.900000e+01
mean	36.313131	2.787879	1.736364e+06	5.343434e+05	2.270707e+06	1.194040e+06
std	6.246054	1.335265	6.736217e+05	6.054450e+05	1.050777e+06	4.376955e+05
min	26.000000	0.000000	2.000000e+05	0.000000e+00	2.000000e+05	1.100000e+05
25%	31.000000	2.000000	1.300000e+06	0.000000e+00	1.550000e+06	8.000000e+05
50%	36.000000	3.000000	1.600000e+06	5.000000e+05	2.100000e+06	1.200000e+06
75%	41.000000	3.000000	2.200000e+06	9.000000e+05	2.700000e+06	1.500000e+06
max	51.000000	4.000000	3.800000e+06	2.100000e+06	5.200000e+06	3.000000e+06

In [8]:
```
# Observing unique value for object dtype columns
for col in ['Profession', 'Marrital Status', 'Education', 'Personal loan', 'Ho
  print(col,':',df[col].unique())
```

```
Profession : ['Salaried' 'Business']
Marrital Status : ['Single' 'Married']
Education : ['Post Graduate' 'Graduate']
Personal loan : ['Yes' 'No']
House Loan : ['No' 'Yes']
Wife Working : ['No' 'Yes' 'm']
Make : ['i20' 'Ciaz' 'Duster' 'City' 'SUV' 'Baleno' 'Verna' 'Luxuray' 'Cr
eata']
```

In [9]:
```
# Observing Column entries
for col in df.columns:
  print(df[col].value_counts())
```

```
Index(['Age', 'Profession', 'Marrital Status', 'Education', 'No of Depend
ents',
       'Personal loan', 'House Loan', 'Wife Working', 'Salary', 'Wife S
alary',
       'Total Salary', 'Make', 'Price'],
      dtype='object')
```

In [7]:
```
df.columns
```

Out[7]:
```
1400000     6
1100000     6
1600000     5
1900000     5
2200000     5
800000      4
2000000     4
3100000     4
1200000     3
1700000     3
2400000     3
2900000     2
2100000     2
```

	Age	Dependents	Salary	Wife Salary	Total Salary	Price
count	99.000000	99.000000	9.900000e+01	9.900000e+01	9.900000e+01	9.900000e+01
mean	36.313131	2.181818	1.736364e+06	5.343434e+05	2.270707e+06	1.194040e+06
std	6.246054	1.335265	6.736217e+05	6.054450e+05	1.050777e+06	4.376955e+05
min	26.000000	0.000000	2.000000e+05	0.000000e+00	2.000000e+05	1.100000e+05
25%	31.000000	2.000000	1.300000e+06	0.000000e+00	1.550000e+06	8.000000e+05
50%	36.000000	2.000000	1.600000e+06	5.000000e+05	2.100000e+06	1.200000e+06
75%	41.000000	3.000000	2.200000e+06	9.000000e+05	2.700000e+06	1.500000e+06
max	51.000000	4.000000	3.800000e+06	2.100000e+06	5.200000e+06	3.000000e+06

In [8]:
```
# Observing unique value for object dtype columns
for col in ['Profession','Marrital Status','Education','Personal loan','Ho
    print(col,':',df[col].unique())
```

```
Profession : ['Salaried' 'Business']
Marrital Status : ['Single' 'Married']
Education : ['Post Graduate' 'Graduate']
Personal loan : ['Yes' 'No']
House Loan : ['No' 'Yes']
Wife Working : ['No' 'Yes' 'm']
Make : ['i20' 'Ciaz' 'Duster' 'City' 'SUV' 'Baleno' 'Verna' 'Luxuray' 'Cr
eata']
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Age', 'Profession', 'Marrital Status', 'Education', 'No of Depend
ents',
       'Personal loan', 'House Loan', 'Wife Working', 'Salary', 'Wife S
alary',
       'Total Salary', 'Make', 'Price'],
      dtype='object')
```

In [9]:
```
# Observing Column entries
for col in df.columns:
    print(df[col].value_counts())
```

```
1400000    6
1100000    6
1600000    5
1900000    5
2200000    5
800000     4
2000000    4
3100000    4
1200000    3
1700000    3
2400000    3
2900000    2
2100000    2
1500000    2
2500000    2
200000     1
2600000    1
2300000    1
2800000    1
3800000    1
```

Cleaning Data

In [10]:
```
## Double checking the percentage of empty entries column wise
df.isnull().sum() / df.shape[0] * 100.00
```

Out[10]:
```
Age                 0.0
Profession          0.0
Marrital Status     0.0
Education           0.0
No of Dependents    0.0
Personal loan       0.0
House Loan          0.0
Wife Working        0.0
Salary              0.0
Wife Salary         0.0
Total Salary        0.0
Make                0.0
Price               0.0
dtype: float64
```

In [11]:
```
df.loc[df['Wife Working'] == 'm']
```

Out[11]:

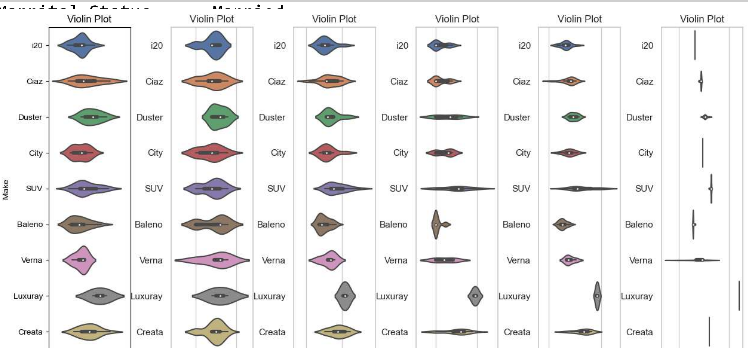
	Age	Profession	Marrital Status	Education	No of Dependents	Personal loan	House Loan	Wife Working	Salary
11	35	Salaried	Married	Graduate	4	Yes	Yes	m	1400000

In [12]:
```
df=df.replace(to_replace ="m", value ="No")
df.loc[11]
```

Out[12]:
```
Age                35
Profession         Salaried
Marrital Status    Married
```

Behavioral and Psychographic Analysis

In [13]:
```
plt.figure(1,figsize=(15,7))
n = 0
for cols in ['Age','No of Dependents','Salary','Wife Salary','Total Salary
    n += 1
    plt.subplot(1,6,n)
    sns.set(style = 'whitegrid')
    plt.subplots_adjust(hspace=0.5,wspace=0.5)
    sns.violinplot(x= cols, y = 'Make', data=df)
    plt.ylabel("Make" if n==1 else '')
    plt.title('Violin Plot')
```



In [11]:
```
df.loc[df['Wife Working'] == 'm']
```

Out[11]:

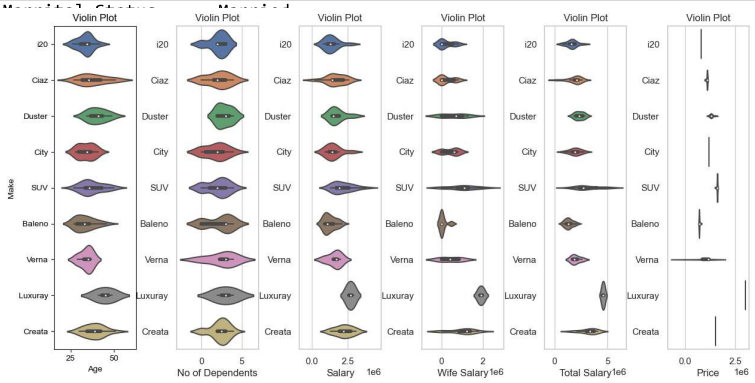
	Age	Profession	Marrital Status	Education	No of Dependents	Personal loan	House Loan	Wife Working	Salary
11	35	Salaried	Married	Graduate	4	Yes	Yes	m	1400000

Behavioral and Psychographic Analysis

In [13]:
```
plt.figure(1,figsize=(15,7))
n = 0
for cols in ['Age','No of Dependents','Salary','Wife Salary','Total Salary','Price']:
    n += 1
    plt.subplot(1,6,n)
    sns.set(style = 'whitegrid')
    plt.subplots_adjust(hspace=0.5,wspace=0.5)
    sns.violinplot(x= cols, y = 'Make', data=df)
    plt.ylabel("Make" if n==1 else '')
    plt.title('Violin Plot')
```

In [12]:
```
df=df.replace(to_replace = "m", value ="No")
df.loc[11]
```

Out[12]:
```
Age                    35
Profession         Salaried
Marrital Status     Married
Education          Graduate
No of Dependents          4
Personal loan           Yes
House Loan              Yes
Wife Working             No
Salary              1400000
Name: 11, dtype: object
```



Observations:

Age: Younger consumers tend to purchase more affordable vehicles.

The choice of vehicle among consumers is significantly influenced by their age. Younger consumers, often characterized by limited financial resources, tend to opt for less expensive vehicles. This preference can be attributed to several factors, including budget constraints, cost-consciousness, and the desire for practicality over luxury. It's common to see younger individuals favoring smaller, more fuel-efficient cars or used vehicles to keep costs down.

Number of Dependents: Consumers with a greater number of dependents prefer vehicles with more seats, often opting for SUVs.

The number of dependents in a consumer's household plays a crucial role in the type of vehicle they choose. Families or individuals responsible for more dependents typically require larger vehicles with more seating capacity. As a result, they are more inclined to select SUVs, minivans, or other spacious models. These vehicles are better suited to accommodate the needs of a larger family, such as transporting children, carpooling, or taking family trips.

Salary: A direct relationship between consumers' salaries and the price of the vehicles they purchase.

A pronounced and direct relationship exists between consumers' salaries and the price of the vehicles they purchase. When you overlay the normalized salary data with the price plot, you'll notice that the median of the salary violin plot closely aligns with the median of the vehicle price plot. This alignment indicates that individuals with higher salaries tend to buy more expensive vehicles, while those with lower incomes gravitate towards more affordable options.

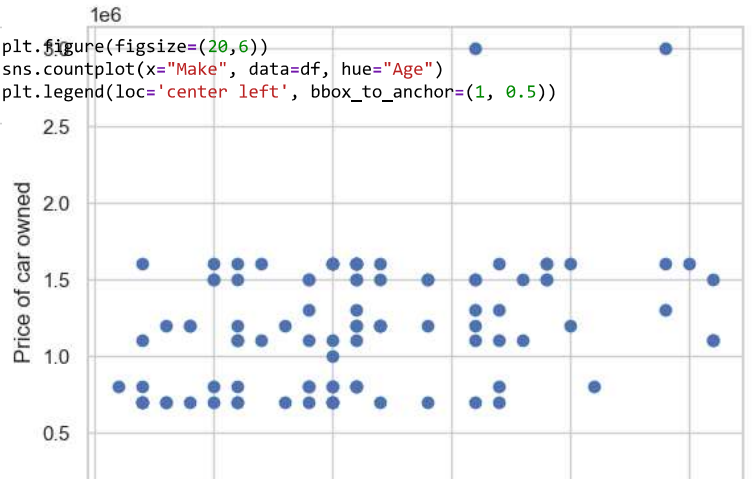
In summary, consumers' age, the number of dependents they have, and their salary significantly influence the type and price of the vehicles they tend to purchase. Younger consumers often choose less expensive vehicles, families or individuals with more dependents prefer larger vehicles like SUVs, and consumers with higher salaries tend to invest in more costly and premium vehicles. These observations underscore the importance of demographics and financial factors in shaping consumer preferences in the automotive market.

In [15]:
```
plt.xlabel('Age')
plt.ylabel('Price of car owned')
plt.scatter(df['Age'],df['Price'])
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x2b5c909bdc0>
```

In [17]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Age")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[17]:



plot. This alignment indicates that individuals with higher salaries tend to buy more expensive vehicles, while those with lower incomes gravitate towards more affordable options.

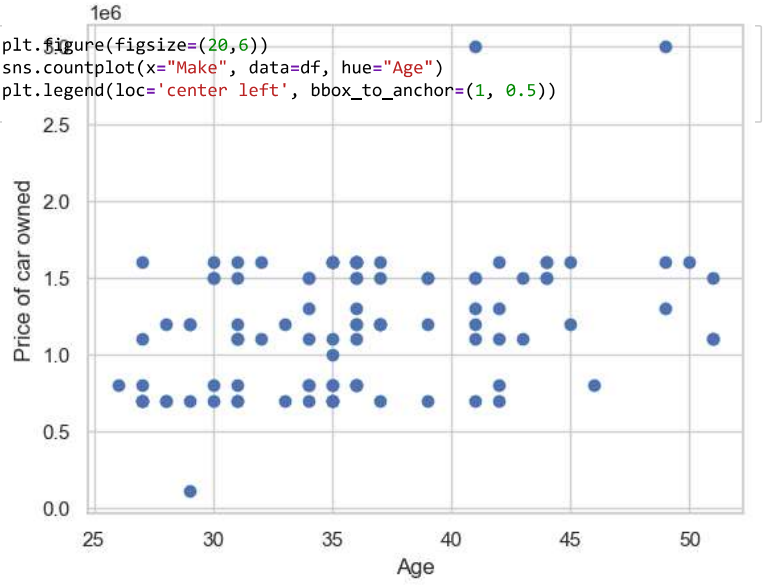
In summary, consumers' age, the number of dependents they have, and their salary significantly influence the type and price of the vehicles they tend to purchase. Younger consumers often choose less expensive vehicles, families or individuals with more dependents prefer larger vehicles like SUVs, and consumers with higher salaries tend to invest in more costly and premium vehicles. These observations underscore the importance of demographics and financial factors in shaping consumer preferences in the automotive market.

In [15]:
```
plt.xlabel('Age')
plt.ylabel('Price of car owned')
plt.scatter(df['Age'],df['Price'])
```

Out[15]: <matplotlib.collections.PathCollection at 0x2b9c9099bc0>

In [17]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Age")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[17]:

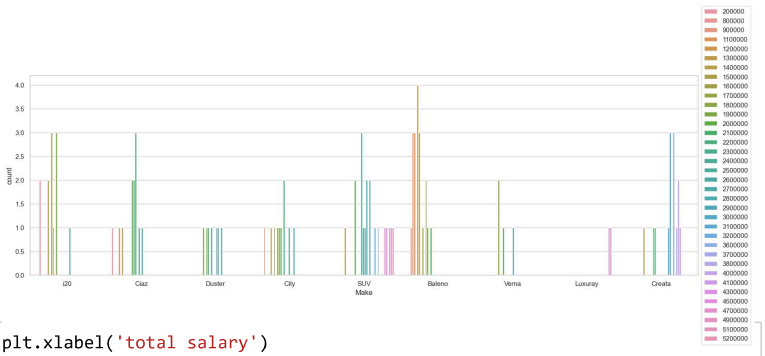


2. Relation between consumers' total salary and the vehicles they tend to purchase

Make of vehicles they tend to purchase

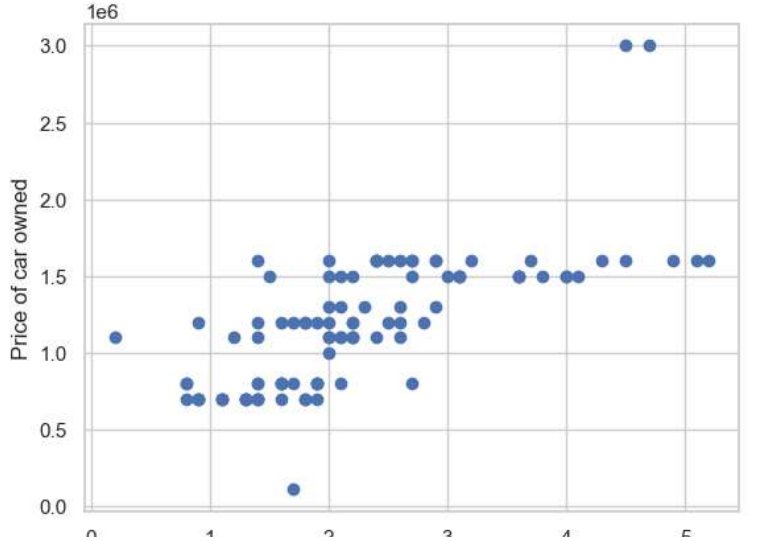
In [16]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Total Salary")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[16]: <matplotlib.legend.Legend at 0x2b9c90b27d0>



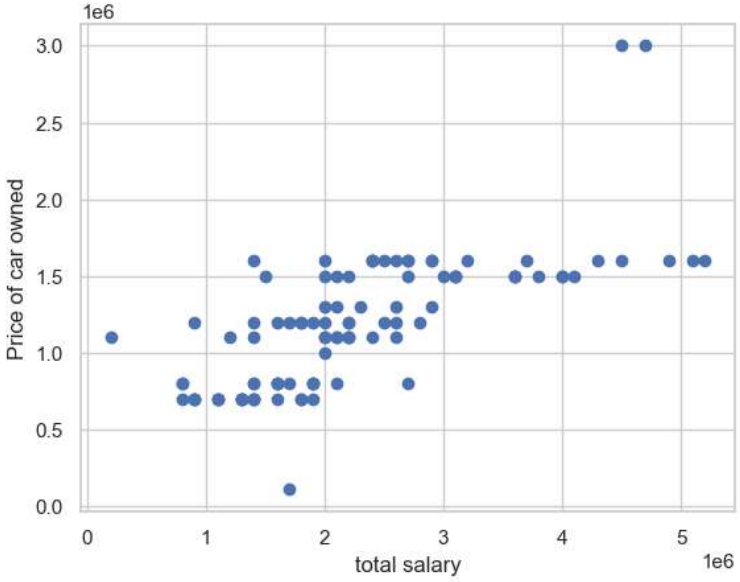
In [18]:
```
plt.xlabel('total salary')
plt.ylabel('Price of car owned')
plt.scatter(df['Total Salary'],df['Price'])
```

Out[18]: <matplotlib.collections.PathCollection at 0x2b9c702c160>



In [18]:
```
plt.xlabel('total salary')
plt.ylabel('Price of car owned')
plt.scatter(df['Total Salary'],df['Price'])
```

Out[18]: <matplotlib.collections.PathCollection at 0x2b9c702c160>

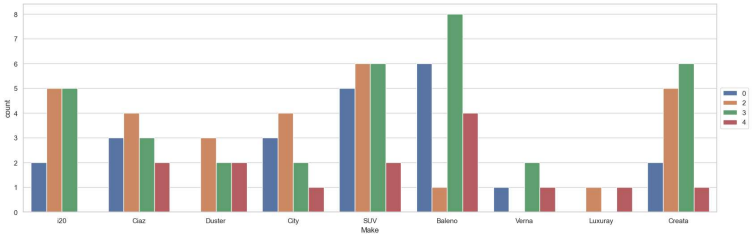


3. Relation between number of dependents on a consumer and the vehicles they tend to purchase

Make of vehicles they tend to purchase

In [19]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="No of Dependents")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[19]: <matplotlib.legend.Legend at 0x2b9c96b7490>

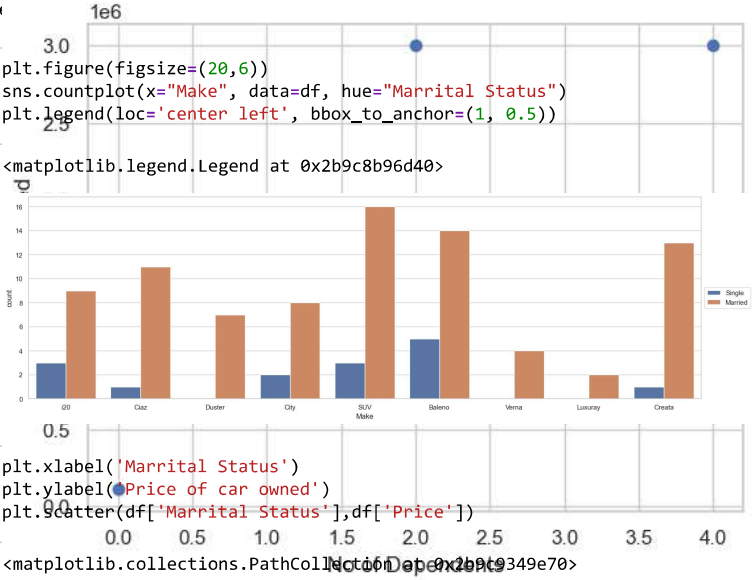


In [20]:
```
plt.xlabel('No of Dependents')
plt.ylabel('Price of car owned')
plt.scatter(df['No of Dependents'],df['Price'])
```

Out[20]: <matplotlib.collections.PathCollection at 0x2b9c973e280>

4. Relation between consumers' marital status and the vehicles they tend to purchase

Mak



In [21]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Marrital Status")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[21]: <matplotlib.legend.Legend at 0x2b9c8b96d40>

In [22]:
```
plt.xlabel('Marrital Status')
plt.ylabel('Price of car owned')
plt.scatter(df['Marrital Status'],df['Price'])
```

Out[22]: <matplotlib.collections.PathCollection at 0x2b9c9349e70>

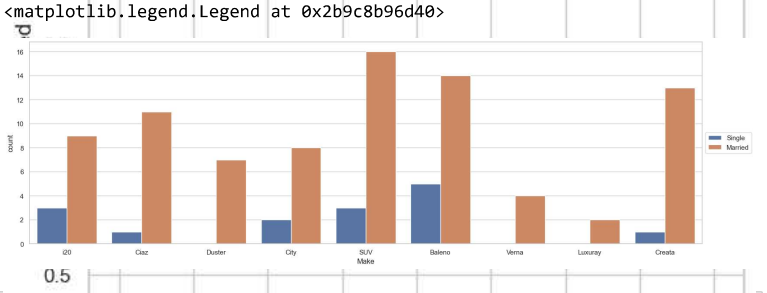
Out[20]: <matplotlib.collections.PathCollection at 0x2b9c973e280>

4. Relation between consumers' marital status and the vehicles they tend to purchase

Make of vehicles they tend to purchase

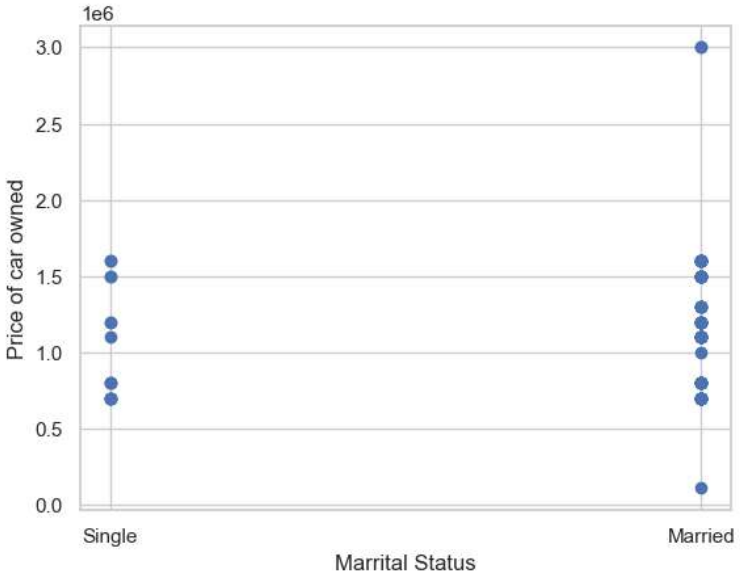
In [21]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Marrital Status")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[21]: <matplotlib.legend.Legend at 0x2b9c8b96d40>



In [22]:
```
plt.xlabel('Marrital Status')
plt.ylabel('Price of car owned')
plt.scatter(df['Marrital Status'],df['Price'])
```

Out[22]: <matplotlib.collections.PathCollection at 0x2b9c9349e70>



5. Relation between consumers profession and the vehicles they tend to purchase

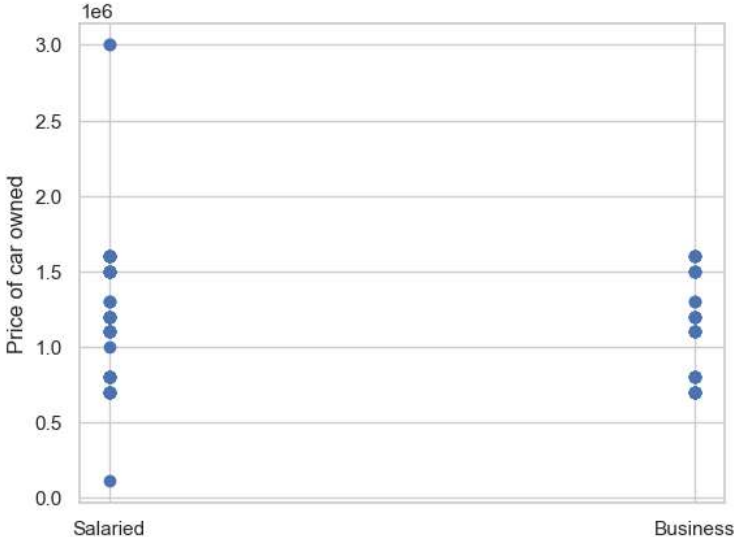
Make of vehicles they tend to purchase

In [23]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Profession")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

In [24]:
```
plt.xlabel('Profession')
plt.ylabel('Price of car owned')
plt.scatter(df['Profession'],df['Price'])
```

Out[23]: <matplotlib.legend.Legend at 0x2b9c6fd1ed0>

Out[24]: <matplotlib.collections.PathCollection at 0x2b9c945d2d0>

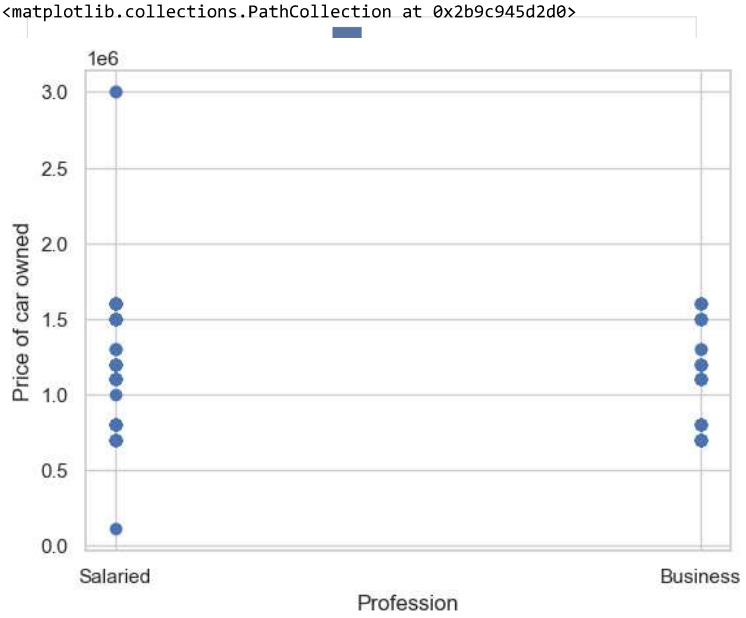


In [23]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Profession")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.ylabel('Price of car owned')
plt.scatter(df['Profession'],df['Price'])
```

Out[23]: <matplotlib.legend.Legend at 0x2b9c6fd1ed0>

In [24]:

Out[24]: <matplotlib.collections.PathCollection at 0x2b9c945d2d0>

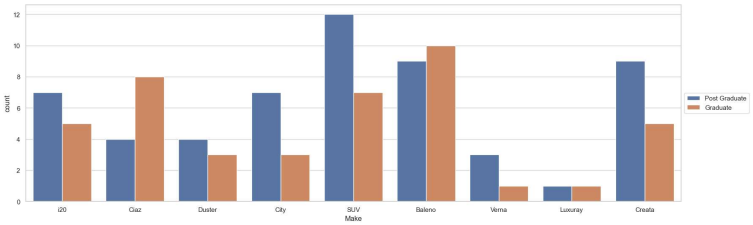


6. Relation between consumers education and the vehicles they tend to purchase

Make of vehicles they tend to purchase

In [25]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Education")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

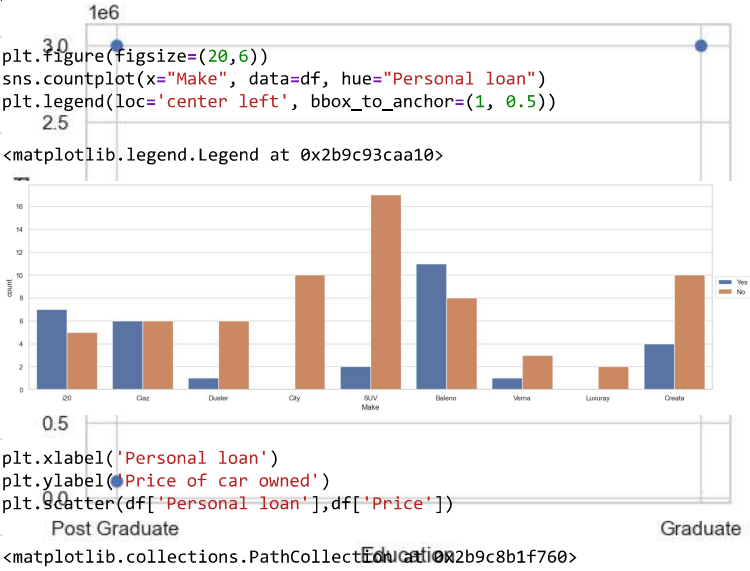
Out[25]: <matplotlib.legend.Legend at 0x2b9c945ff70>



In [26]:
```
plt.xlabel('Education')
plt.ylabel('Price of car owned')
plt.scatter(df['Education'],df['Price'])
```

Out[26]: <matplotlib.collections.PathCollection at 0x2b9c98886a0>

Mak



In [27]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Personal loan")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[27]: <matplotlib.legend.Legend at 0x2b9c93caa10>

In [28]:
```
plt.xlabel('Personal loan')
plt.ylabel('Price of car owned')
plt.scatter(df['Personal loan'],df['Price'])
```

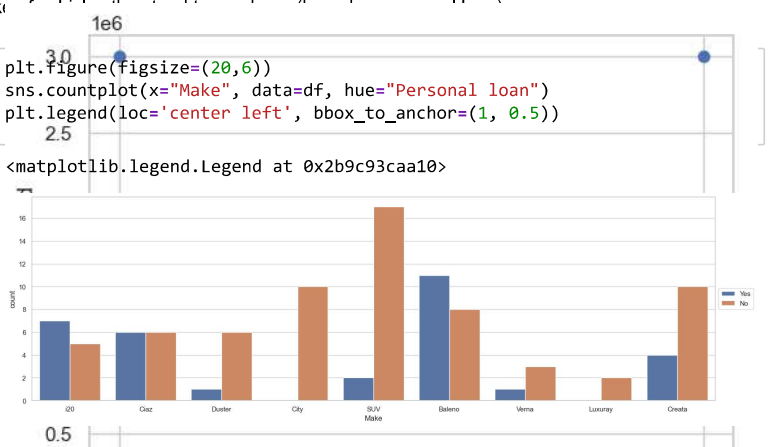
Out[28]: <matplotlib.collections.PathCollection at 0x2b9c8b1f760>

7.

Mak

In [27]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Personal loan")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

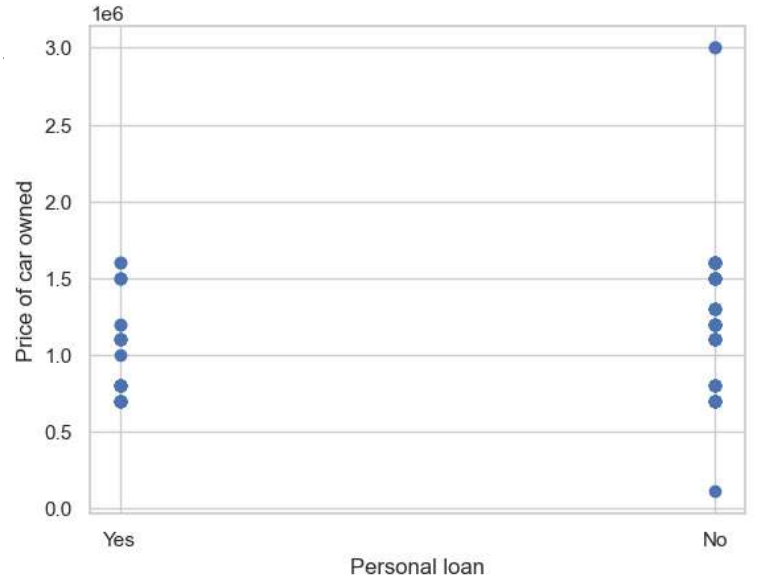
Out[27]: <matplotlib.legend.Legend at 0x2b9c93caa10>



In [28]:
```
plt.xlabel('Personal loan')
plt.ylabel('Price of car owned')
plt.scatter(df['Personal loan'],df['Price'])
```

Out[28]: <matplotlib.collections.PathCollection at 0x2b9c8b1f760>

7.



Make of vehicles they tend to purchase (based on house loan)

In [29]:
```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="House Loan")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[29]: <matplotlib.legend.Legend at 0x2b9c6fb0340>

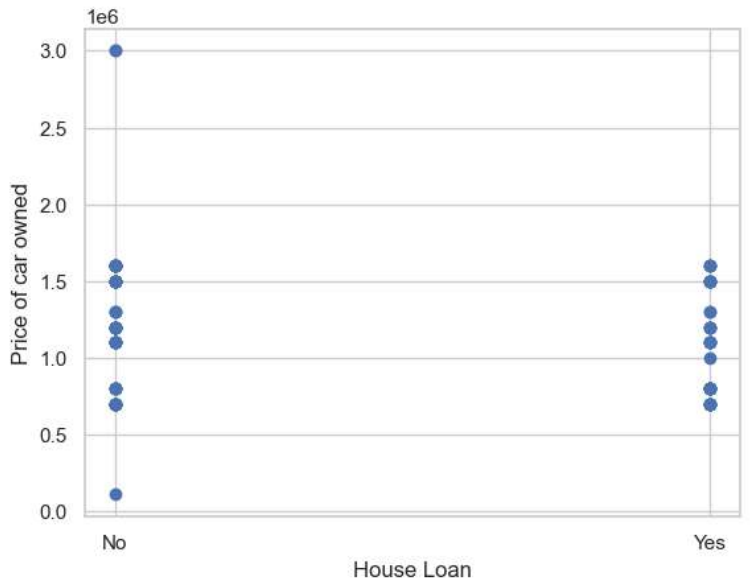
In [30]:
```
plt.xlabel('House Loan')
plt.ylabel('Price of car owned')
plt.scatter(df['House Loan'],df['Price'])
```

Out[30]: <matplotlib.collections.PathCollection at 0x2b9c902ae60>

Out[29]: <matplotlib.legend.Legend at 0x2b9c6fb0340>

In [30]:
```
plt.xlabel('House Loan')
plt.ylabel('Price of car owned')
plt.scatter(df['House Loan'],df['Price'])
```

Out[30]: <matplotlib.collections.PathCollection at 0x2b9c902ae60>



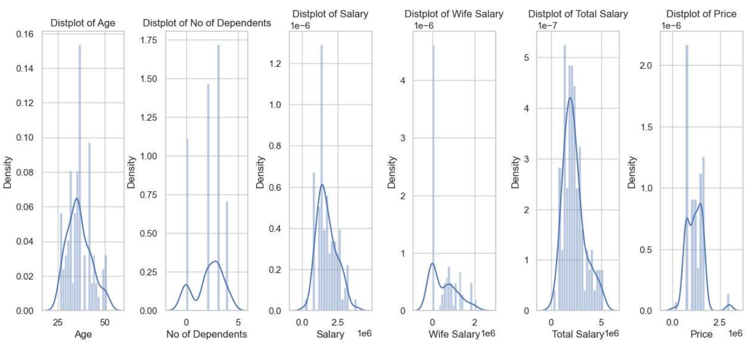
Demographic Analysis

In [32]:
```
plt.figure(1, figsize=(15, 6))
n = 0

# Correct the column names to match your DataFrame
for x in ['Age', 'No of Dependents', 'Salary', 'Wife Salary', 'Total Salar
    n += 1
    plt.subplot(1, 6, n)
    plt.subplots_adjust(hspace=0.5, wspace=0.5)
    sns.distplot(df[x], bins=20)
    plt.title('Distplot of {}'.format(x))

plt.show()
```

In [32]:
```
plt.figure(1, figsize=(15, 6))
n = 0

# Correct the column names to match your DataFrame
for x in ['Age', 'No of Dependents', 'Salary', 'Wife Salary', 'Total Salar
    n += 1
    plt.subplot(1, 6, n)
    plt.subplots_adjust(hspace=0.5, wspace=0.5)
    sns.distplot(df[x], bins=20)
    plt.title('Distplot of {}'.format(x))

plt.show()
```

```
C:\Users\Rv\AppData\Local\Temp\ipykernel_11868\2659763312.py:9: UserWarni
ng:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df[x], bins=20)
C:\Users\Rv\AppData\Local\Temp\ipykernel_11868\2659763312.py:9: UserWarni
ng:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)
```

```
C:\Users\Rv\AppData\Local\Temp\ipykernel_11868\2659763312.py:9: UserWarni
ng:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df[x], bins=20)
C:\Users\Rv\AppData\Local\Temp\ipykernel_11868\2659763312.py:9: UserWarni
ng:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df[x], bins=20)
C:\Users\Rv\AppData\Local\Temp\ipykernel_11868\2659763312.py:9: UserWarni
ng:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df[x], bins=20)
C:\Users\Rv\AppData\Local\Temp\ipykernel_11868\2659763312.py:9: UserWarni
ng:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)
  sns.distplot(df[x], bins=20)
C:\Users\Rv\AppData\Local\Temp\ipykernel_11868\2659763312.py:9: UserWarni
ng:
  sns.distplot(df[x], bins=20)
```

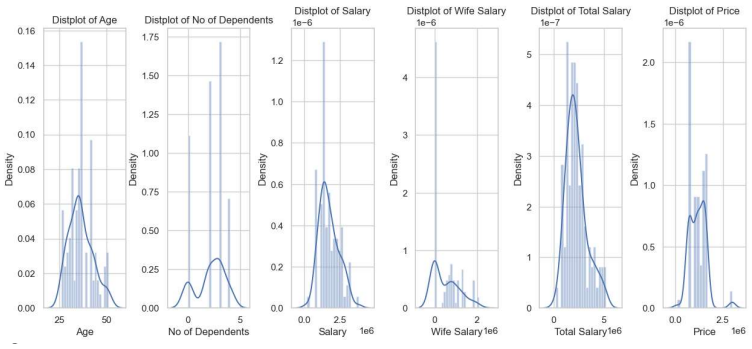



```
`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).
```

Observations:

1. The majority of consumers in the market fall within the age range of 25 to 50, making up the largest segment of buyers.
2. Vehicles are most commonly purchased by individuals with an average total salary in the vicinity of 30 lakhs.
3. The typical expenditure for vehicles predominantly falls within the range of 10 to 20 lakhs.

```
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (http
s://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)
  sns.distplot(df[x], bins=20)
C:\Users\Rv\AppData\Local\Temp\ipykernel_11868\2659763312.py:9: UserWarni
ng:
sns.distplot(df[x], bins=20)
```




```
`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.
Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).
  with sns.distplot.
```

Observations:

1. The majority of consumers in the market fall within the age range of 25 to 50, making up the largest segment of buyers.
2. Vehicles are most commonly purchased by individuals with an average total salary in the vicinity of 30 lakhs.
3. The typical expenditure for vehicles predominantly falls within the range of 10 to 20 lakhs.

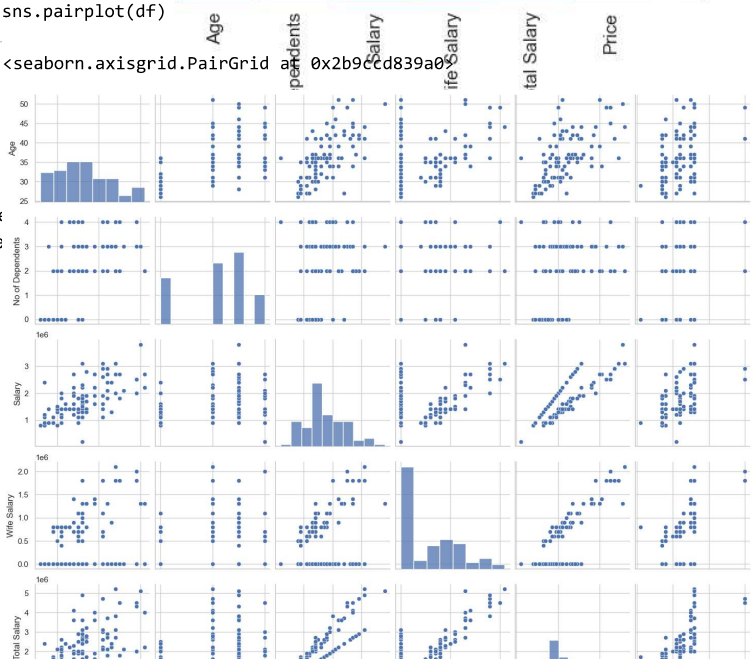
In [33]:
```
# Heatmap of Correlation
sns.heatmap(df.corr(), annot=True)
```

```
C:\Users\Rv\AppData\Local\Temp\ipykernel_11868\3903709160.py:2: FutureWar
ning: 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.
  sns.heatmap(df.corr(), annot=True)
```

Out[33]: <Axes: >

In [34]:
```
# Pair Plot
sns.pairplot(df)
```

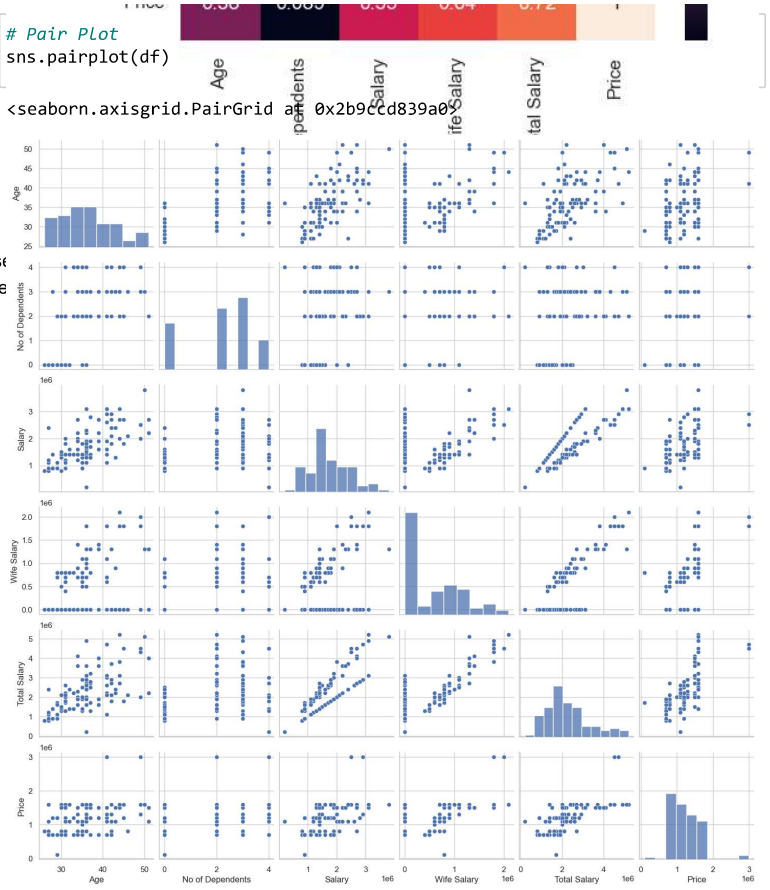
Out[34]: <seaborn.axisgrid.PairGrid at 0x2b9ccd839a0>



Obse
have

In [34]:
```
# Pair Plot
sns.pairplot(df)
```

Out[34]: <seaborn.axisgrid.PairGrid at 0x2b9ccd839a0>



Obse
have

Geographic Analysis

In [35]:
```
# Importing state-wise sales dataset
data = pd.read_csv('EVStats.csv')
data
```

Out[35]:

	Sl. No	State	Two Wheelers (Category L1 & L2 as per Central Motor Vehicles Rules	Two Wheelers (Category L2 (CMVR))	Two Wheelers (Max power not exceeding 250 Watts)	Three Wheelers (Category L5 slow speed as per CMVR)	Three Wheelers (Category L5 as per CMVR)	Passenger Cars (Category M1 as per CMVR)	Bu
0	1	Meghalaya	0	0	0	0	0	6	
1	2	Nagaland	0	20	3	0	0	1	
2	3	Manipur	16	8	11	0	5	12	
3	4	Tripura	28	9	36	0	0	8	
4	5	Andaman & Nicobar islands	0	0	0	0	0	82	
5	6	Himachal Pradesh	0	0	0	0	0	98	
6	7	Jammu & Kashmir	2	76	152	0	0	208	
7	8	Goa	0	0	0	0	0	513	
		Dadra and							

In [35]:
```
# Importing state-wise sales dataset
data = pd.read_csv('EVStats.csv')
data
```

Out[35]:

	Sl. No	State	Two Wheelers (Category L1 & L2 as per Central Motor Vehicles Rules	Two Wheelers (Category L2 (CMVR))	Two Wheelers (Max power not exceeding 250 Watts)	Three Wheelers (Category L5 slow speed as per CMVR)	Three Wheelers (Category L5 as per CMVR)	Passenger Cars (Category M1 as per CMVR)	Bu
0	1	Meghalaya	0	0	0	0	0	6	
1	2	Nagaland	0	20	3	0	0	1	
2	3	Manipur	16	8	11	0	5	12	
3	4	Tripura	28	9	36	0	0	8	
4	5	Andaman & Nicobar islands	0	0	0	0	0	82	
5	6	Himachal Pradesh	0	0	0	0	0	98	
6	7	Jammu & Kashmir	2	76	152	0	0	208	
7	8	Goa	0	0	0	0	0	513	
8	9	Dadra and Nagar Haveli	4	0	9	0	0	803	
9	10	Jharkhand	75	228	736	9	7	655	
10	11	Assam	463	138	1006	0	117	151	
11	12	Chandigarh	612	18	896	0	0	974	
12	13	Bihar	252	430	2148	6	64	271	
13	14	Odisha	377	824	2031	0	37	594	
14	15	Uttarkhand	423	168	3239	45	38	265	
15	16	Chhattisgarh	613	382	2078	58	106	997	
16	17	Madhya Pradesh	503	378	2904	8	106	2562	
17	18	Punjab	698	300	1968	0	5	3567	
18	19	Telangana	535	711	2256	2	0	5530	
19	20	Andhra Pradesh	431	692	4689	0	0	3680	
20	21	Kerala	432	78	4961	1	0	5729	
21	22	Karnataka	784	1104	3252	2	0	8242	
22	23	West Bengal	1451	65	10781	3	0	1840	
23	24	Rajasthan	2036	1153	8375	19	64	4116	
24	25	Tamil Nadu	491	863	8260	0	0	7132	
25	26	Delhi	1395	251	5018	0	1	12695	
26	27	Haryana	3162	1504	13908	113	24	4878	
27	28	Uttar Pradesh	2934	2355	15199	117	139	5445	
28	29	Gujarat	7182	217	8476	0	4	15388	
29	30	Maharashtra	2630	2097	10146	6	3	19129	

In [37]:
```
X = df.iloc[:, df.columns!='Make']
X.head()
```

Out[37]:

	Age	Profession	Marrital Status	Education	No of Dependents	Personal loan	House Loan	Wife Working	Salary
0	27	Salaried	Single	Post Graduate	0	Yes	No	No	800000
1	35	Salaried	Married	Post Graduate	2	Yes	Yes	Yes	1400000
2	45	Business	Married	Graduate	4	Yes	Yes	No	1800000
3	41	Business	Married	Post Graduate	3	No	No	Yes	1600000
4	31	Salaried	Married	Post Graduate	2	Yes	No	Yes	1800000

Top 5 states for each category of Electric Vehicles

for y in ['Two Wheelers (Category L1 & L2 as per Central Motor Vehicles Rules','Two Wheelers (Category L2 (CMVR))','Two Wheelers (Max power not exceeding 250 Watts)','Three Wheelers (Category L5 slow speed as per CMVR)','Three Wheelers (Category L5 as per CMVR)','Passenger Cars (Category M1 as per CMVR)','Buses','Total in state']: ax = data

In [ ]:
```
# encoding = {"Profession":{"Salaried": 0, "Business": 1},
            "Marrital Status":{"Single": 0, "Married": 1},
            "Education":{"Graduate": 0, "Post Graduate": 1},
            "Personal loan":{"No": 0, "Yes": 1},
            "House Loan":{"No": 0, "Yes": 1},
            "Wife Working":{"No": 0, "Yes": 1}
            }
```

Observations: Based on the type of electric vehicle, states with higher number of electric vehicle can be targeted as people in these states are more likely to purchase them.

Model Deployment

K-Means Clustering

In [39]:
```
obj_df = X.replace(encoding)
```

27	28	Uttar Pradesh	2954	2355	15199	117	139	5445
28	29	Gujarat	7182	217	8476	0	4	15388
29	30	Maharashtra	2630	2097	10146	6	3	19129

Top 5 states for each category of Electric Vehicles

for y in ['Two Wheelers (Category L1 & L2 as per Central Motor Vehicles Rules','Two Wheelers (Category L2 (CMVR))','Two Wheelers (Max power not exceeding 250 Watts)','Three Wheelers (Category L5 slow speed as per CMVR)','Three Wheelers (Category L5 as per CMVR)','Passenger Cars (Category M1 as per CMVR)','Buses','Total in state']: ax = data.sort_values(y, ascending = True).tail(5).plot.barh(x='State', ax = ax)

Observations: Based on the type of electric vehicle, states with higher number of electric vehicle can be targeted as people in these states are more likely to purchase them.

Model Deployment

K-Means Clustering

In [37]:
```python
X = df.iloc[:,df.columns != 'Make']
X.head()
```

Out[37]:

	Age	Profession	Marrital Status	Education	No of Dependents	Personal loan	House Loan	Wife Working	Salary
0	27	Salaried	Single	Post Graduate	0	Yes	No	No	800000
1	35	Salaried	Married	Post Graduate	2	Yes	Yes	Yes	1400000
2	45	Business	Married	Graduate	4	Yes	Yes	No	1800000
3	41	Business	Married	Post Graduate	3	No	No	Yes	1600000
4	31	Salaried	Married	Graduate	2	Yes	No	Yes	1800000

In [ ]:
```python
# encoding = {"Profession":{"Salaried": 0, "Business": 1},
              "Marrital Status":{"Single": 0, "Married": 1},
              "Education":{"Graduate": 0, "Post Graduate": 1},
              "Personal loan":{"No": 0, "Yes": 1},
              "House Loan":{"No": 0, "Yes": 1},
              "Wife Working":{"No": 0, "Yes": 1}
              }
```

In [39]:
```python
obj_df = X.replace(encoding)
obj_df.head()
```

Out[39]:

	Age	Profession	Marrital Status	Education	No of Dependents	Personal loan	House Loan	Wife Working	Salary
0	27	0	0	1	0	1	0	0	800000
1	35	0	1	1	2	1	1	1	1400000
2	45	1	1	0	4	1	1	0	1800000
3	41	1	1	1	3	0	0	1	1600000
4	31	0	1	1	2	1	0	1	1800000

K - Means Algorithm

In [40]:
```python
# Importing Important Libraries
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

In [41]:
```python
X_scaled = StandardScaler().fit_transform(obj_df)
X_scaled = pd.DataFrame(X_scaled,columns=['Age', 'Profession', 'Marrital S
                                          'Personal loan', 'House Loan', '
                                          'Total Salary','Price'])

x = X_scaled.to_numpy()
X_scaled
```

Out[41]:

	Age	Profession	Marrital Status	Education	No of Dependents	Personal loan	House Loan	Wif Workin
0	-1.498630	-0.739510	-2.366432	0.876275	-1.642313	1.446980	-0.772512	-1.05184
1	-0.211304	-0.739510	0.422577	0.876275	-0.136859	1.446980	1.294479	0.95070
2	1.397855	1.352247	0.422577	-1.141195	1.368594	1.446980	1.294479	-1.05184
3	0.754191	1.352247	0.422577	0.876275	0.615867	-0.691095	-0.772512	0.95070
4	-0.854967	-0.739510	0.422577	0.876275	-0.136859	1.446980	-0.772512	0.95070
...	...	...	...	...	...	...	...	...
94	-1.498630	1.352247	-2.366432	-1.141195	-1.642313	-0.691095	-0.772512	-1.05184
95	2.202434	-0.739510	0.422577	0.876275	0.615867	-0.691095	-0.772512	0.95070
96	2.363350	1.352247	0.422577	-1.141195	-0.136859	1.446980	1.294479	-1.05184
97	2.363350	-0.739510	0.422577	0.876275	-0.136859	-0.691095	-0.772512	0.95070
98	2.363350	-0.739510	0.422577	0.876275	-0.136859	1.446980	1.294479	-1.05184

99 rows × 12 columns

In [42]:
```python
wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)
```

```
C:\Users\Rv\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, w
hen there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Rv\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, w
hen there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Rv\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, w
hen there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Rv\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, w
hen there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Rv\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, w
```

In [42]:
```
wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
                    max_iter = 300, n_init = 10, random_state = 42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)
```

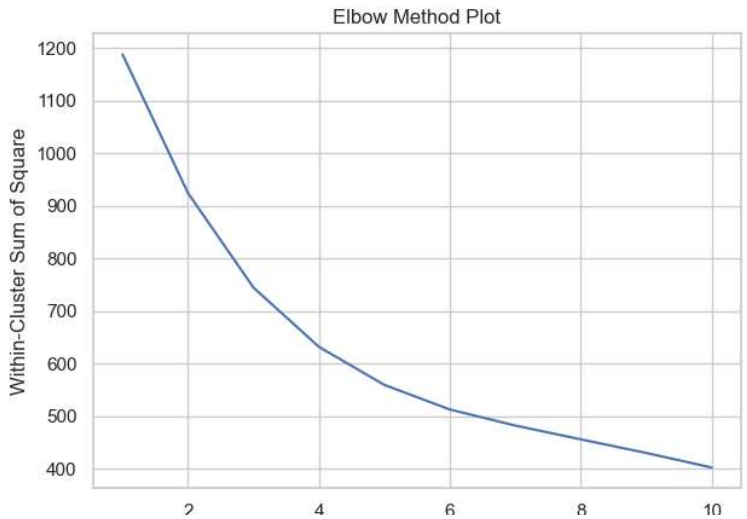
Out[41]:

	Age	Profession	Marrital Status	Education	No of Dependents	Personal loan	House Loan	Wif Workin
0	-1.498630	-0.739510	2.366432	0.876275	-1.643121	1.446980	-0.772512	-1.05184
1	-0.211304	-0.739510	0.422577	0.876275	-0.136859	1.446980	1.294479	0.95070
2	1.397855	1.352247	0.422577	-1.141195	1.368594	1.446980	1.294479	-1.05184
3	0.734191	1.352247	0.422577	0.876275	0.615867	-0.691095	-0.772512	0.95070
4	-0.854967	-0.739510	0.422577	0.876275	-0.136859	1.446980	-0.772512	0.95070
...	...	...	...	...	...	...	...	...
94	-1.409080	1.352247	2.366432	-1.141195	-1.643121	-0.691095	-0.772512	-1.05184
95	2.202434	-0.739510	0.422577	0.876275	0.615867	-0.691095	-0.772512	0.95070
96	-1.363350	1.352247	0.422577	-1.141195	-1.643121	1.446980	1.294479	-1.05184
97	2.363350	-0.739510	0.422577	0.876275	-0.136859	-0.691095	-0.772512	0.95070
98	-1.363350	-0.739510	0.422577	0.876275	-0.136859	1.446980	1.294479	-1.05184

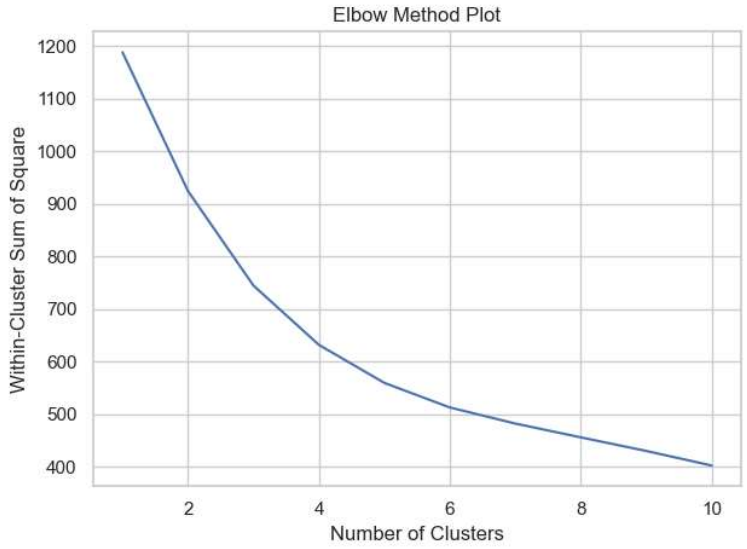
99 rows × 12 columns

```
C:\Users\Rv\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, w
hen there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```
(repeated 10 times)

In [43]:
```
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method Plot')
plt.xlabel('Number of Clusters')
plt.ylabel('Within-Cluster Sum of Square') # Within cluster sum of squares
plt.tight_layout()
plt.show()
```



In [43]:
```
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method Plot')
plt.xlabel('Number of Clusters')
plt.ylabel('Within-Cluster Sum of Square') # Within cluster sum of squares
plt.tight_layout()
plt.show()
```



K = 3

In [44]:
```
kmeans = KMeans(n_clusters = 3, init = 'k-means++',
                max_iter = 300, n_init = 10,random_state = 42)
kmeans.fit(X_scaled)
```

```
C:\Users\Rv\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, w
hen there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[44]:
```
KMeans(n_clusters=3, n_init=10, random_state=42)
```

In [45]:
```
y = kmeans.predict(X_scaled)
y_df = pd.DataFrame(y,columns=['Class'])
```

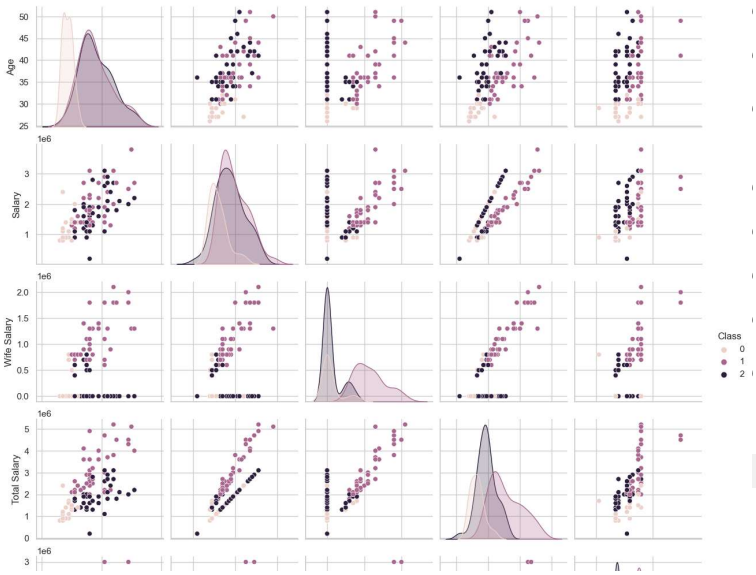
In [46]:
```
final_data = pd.concat([df,y_df],axis=1)
final_data
```

Out[46]:

	Age	Profession	Marrital Status	Education	No of Dependents	Personal loan	House Loan	Wife Working	Salary	Post[
0	27	Salaried	Single		0	Yes	No	No	800000	
1	35	Salaried	Married	Post Graduate	2	Yes	Yes	Yes	1400000	

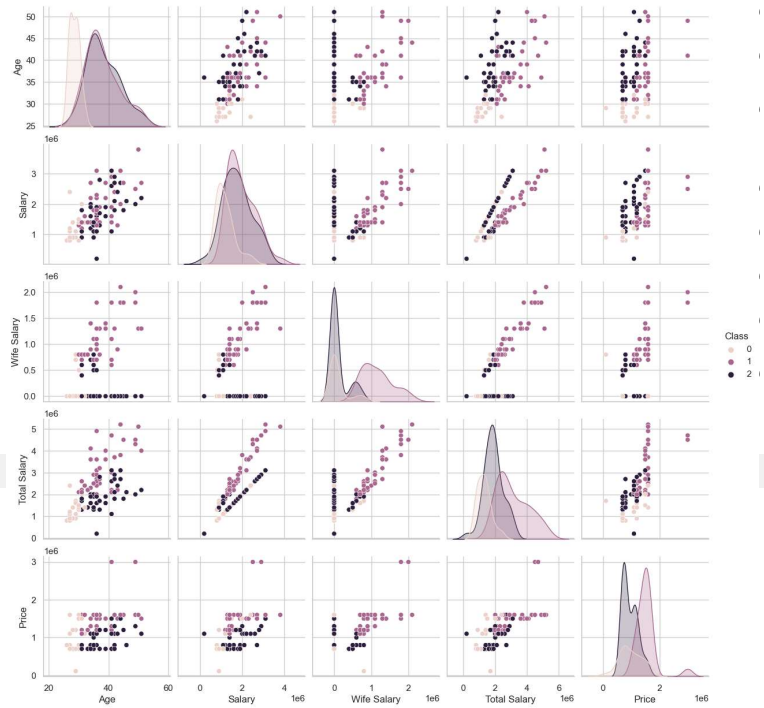
In [47]:
```
sns.pairplot(final_data,x_vars=['Age','Salary', 'Wife Salary','Total Sal
```

Out[47]:
```
<seaborn.axisgrid.PairGrid at 0x2b9d320bfa0>
```



In [47]:
```
sns.pairplot(final_data,x_vars=['Age','Salary', 'Wife Salary','Total Sal
```

Out[47]:
```
<seaborn.axisgrid.PairGrid at 0x2b9d320bfa0>
```



In [48]:
```
kmeans1 = KMeans(n_clusters = 5, init = 'k-means++',
                 max_iter = 300, n_init = 10,random_state = 42)
kmeans1.fit(X_scaled)
```

```
C:\Users\Rv\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, w
hen there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[48]:
```
KMeans(n_clusters=5, n_init=10, random_state=42)
```

In [49]:
```
y1 = kmeans1.predict(X_scaled)
y1_df = pd.DataFrame(y1,columns=['Class'])
```

In [50]:
```
final_data1 = pd.concat([df,y1_df],axis=1)
final_data1
```

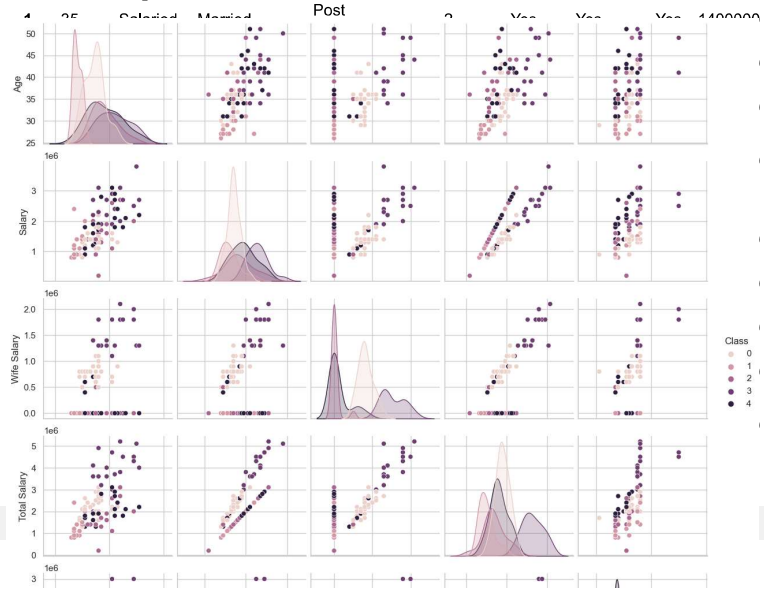
Out[50]:

	Age	Profession	Marrital Status	Education	No of Dependents	Personal Loan	House Loan	Wife Working	Salary	Wife Salary	Total Salary	Price	Class
0	27	Salaried	Single	Post Graduate	0	Yes	No	No	800000	0	800000	2600000	0
1	35	Salaried	Married	Post Graduate	2	Yes	Yes	Yes	1400000	1400000	2800000	4500000	1
2	45	Business	Married	Graduate	4	No	Yes	Yes	1800000	1800000	3600000	6000000	2
3	41	Salaried	Married	Graduate	3	No	Yes	Yes	1600000	1600000	3200000	5200000	2
4	31	Salaried	Married	Graduate	2	Yes	Yes	Yes	1200000	1200000	2400000	3900000	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	29	Salaried	Married	Graduate	2	Yes	Yes	Yes	1200000	1200000	2400000	3900000	1
1996	30	Salaried	Married	Graduate	2	Yes	Yes	Yes	1200000	1200000	2400000	3900000	1
1997	30	Salaried	Married	Graduate	2	Yes	Yes	Yes	1200000	1200000	2400000	3900000	1
1998	30	Salaried	Married	Graduate	2	Yes	Yes	Yes	1200000	1200000	2400000	3900000	1
1999	30	Salaried	Married	Graduate	2	Yes	Yes	Yes	1200000	1200000	2400000	3900000	1

2000 rows × 13 columns

In [51]:
```
Salary', 'Wife Salary','Total Salary','Price'], y_vars = ['Age','Salary',
```

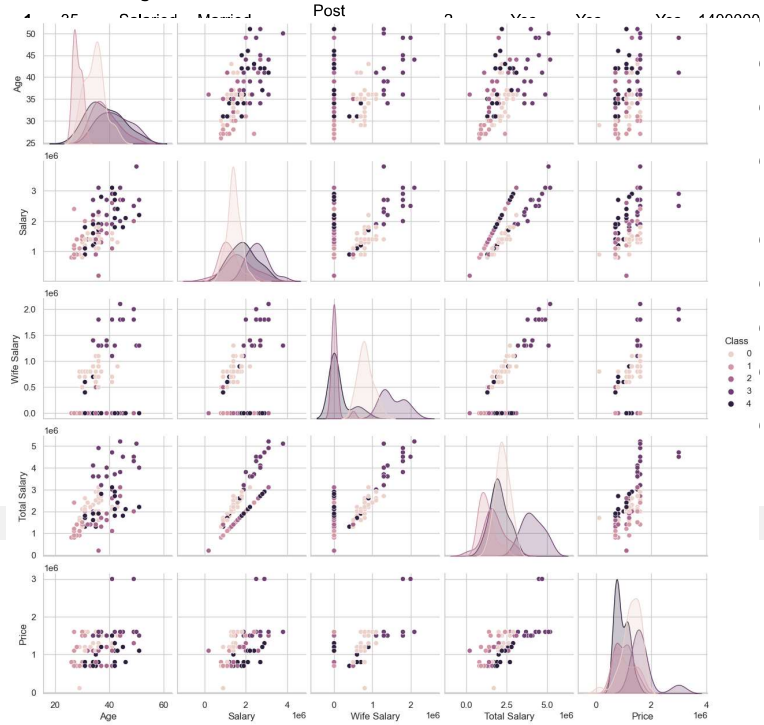
Out[51]:
```
<seaborn.axisgrid.PairGrid at 0x2b9d0b11f60>
```



In [51]:
```
Salary', 'Wife Salary', 'Total Salary', 'Price'], y_vars = ['Age', 'Salary',
```

Age	Profession	Marrital Status	Education	No of Dependents	Personal Loan	House Loan	Wife Working	Salary
			Graduate					
			Post					

Out[51]: <seaborn.axisgrid.PairGrid at 0x2b9d0b11f60>



In [ ]: