# 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.**

# Importing Libraries

In [1]:

```
# Importing Important Libraries
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 [6]:

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

Out[6]:

| | 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 [7]:

```
df.shape
```

Out[7]:

```
(99, 13)
```

In [8]:

```
df.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
```

In [9]:

```
df.describe()
```

Out[9]:

| | 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 | 2.181818 | 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 [10]:

```
df.columns
```

Out[10]:

```
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 [11]:

```
# Observing unique value for object dtype columns
for col in ['Profession','Marrital Status','Education','Personal loan','House Loan','Wife Working','Make']:
  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' 'Creata']
```

In [12]:

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

```
Age
36    13
```

```
35    10
31     8
41     7
34     7
27     6
37     6
42     5
30     5
39     4
44     4
29     4
51     3
49     3
28     3
43     2
33     2
32     2
45     2
46     1
50     1
26     1
Name: count, dtype: int64
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
No of Dependents
3    34
2    29
0    22
4    14
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
Salary
1400000    17
900000      8
1800000     7
2700000     6
1300000     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
Name: count, dtype: int64
Wife Salary
0          48
800000      8
1300000     7
700000      6
600000      5
1100000     5
900000      5
1800000     5
500000      3
1400000     3
400000      1
2000000     1
1000000     1
2100000     1
Name: count, dtype: int64
Total Salary
1400000    8
2000000    7
2200000    6
1900000    5
2100000    5
1600000    5
1800000    4
2600000    4
900000     4
1300000    4
2400000    4
2700000    4
800000     3
1100000    3
3100000    3
3600000    3
2900000    3
1700000    3
2500000    2
4500000    2
4000000    2
1500000    1
2800000    1
4900000    1
4100000    1
5200000    1
3200000    1
3000000    1
1200000    1
4700000    1
3800000    1
4300000    1
200000     1
2300000    1
3700000    1
5100000    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
```

```
Price
1600000    18
700000     18
1500000    16
800000     13
1200000    13
1100000    12
1300000     5
3000000     2
1000000     1
110000      1
Name: count, dtype: int64
```

# Cleaning Data

In [13]:

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

Out[13]:

```
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
```

**There are no null entries.**

In [14]:

```
df.loc[df['Wife Working'] == 'm']
```

Out[14]:

| | Age | Profession | Marrital Status | Education | No of Dependents | Personal loan | House Loan | Wife Working | Salary | Wife Salary | Total Salary | Make | Price |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 11 | 35 | Salaried | Married | Graduate | 4 | Yes | Yes | m | 1400000 | 0 | 1400000 | Baleno | 700000 |

**We can see that Wife Salary has been mentioned as 0, so it is safe to change 'm' with 'no' under Wife Working for simplication of data.**

In [15]:

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

Out[15]:

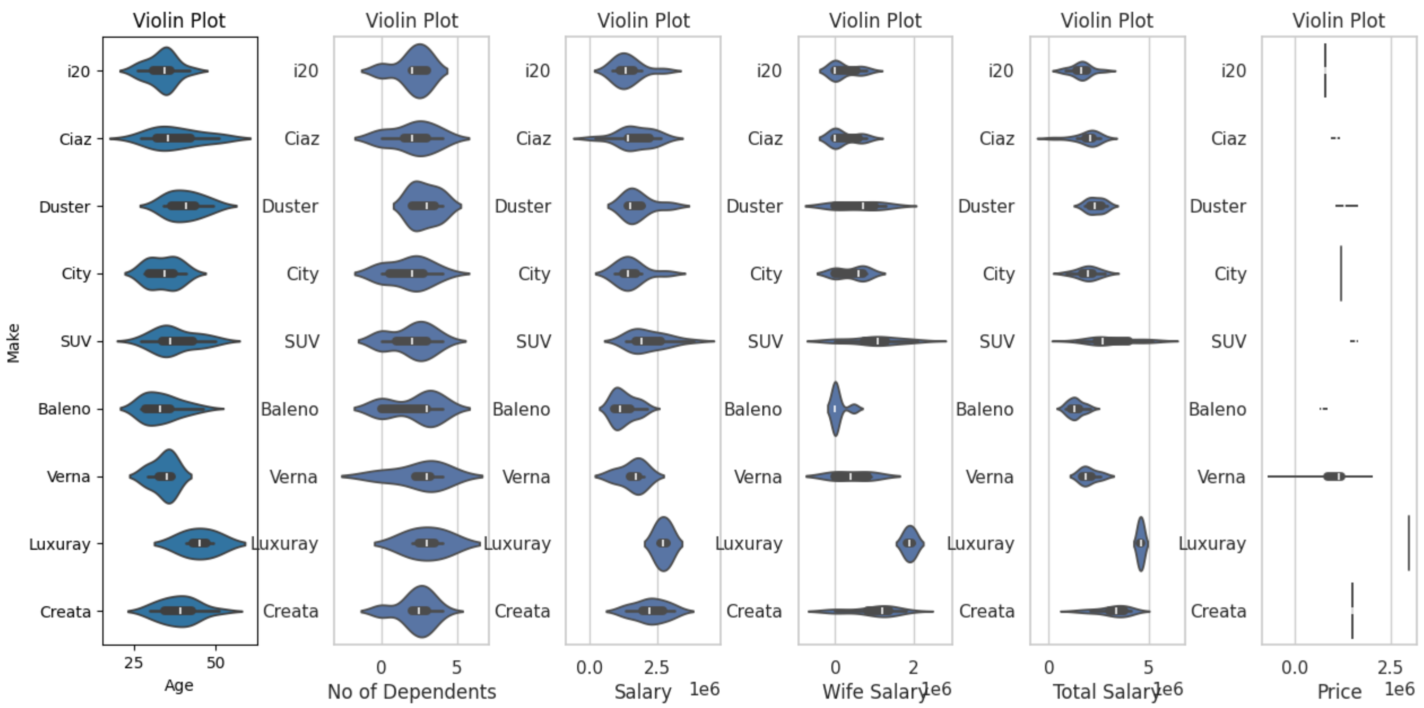
```
Age                        35
Profession           Salaried
Marrital Status       Married
Education            Graduate
No of Dependents            4
Personal loan             Yes
House Loan                Yes
Wife Working               No
Salary                1400000
```

```
Wife Salary                0
Total Salary         1400000
Make                  Baleno
Price                 700000
Name: 11, dtype: object
```

# Behavioral and Psychographic Analysis

In [16]:

```
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')
```




**Observations:**

- **Age: Younger consumers purchase less expensive vehicles.**
- **Number of Dependents: Greater number of dependents makes the consumer buy a vehicle with more seats and so they prefer SUVs.**
- **Salary: If you overlap the normalised salary plots with price plot, you would observe the median of salary violin plot matches that of the price of the vehicle indicating a very direct relationship.**

# 1. Relation between consumers' age and the vehicles they tend to purchase

- **Make of vehicles they tend to purchase**

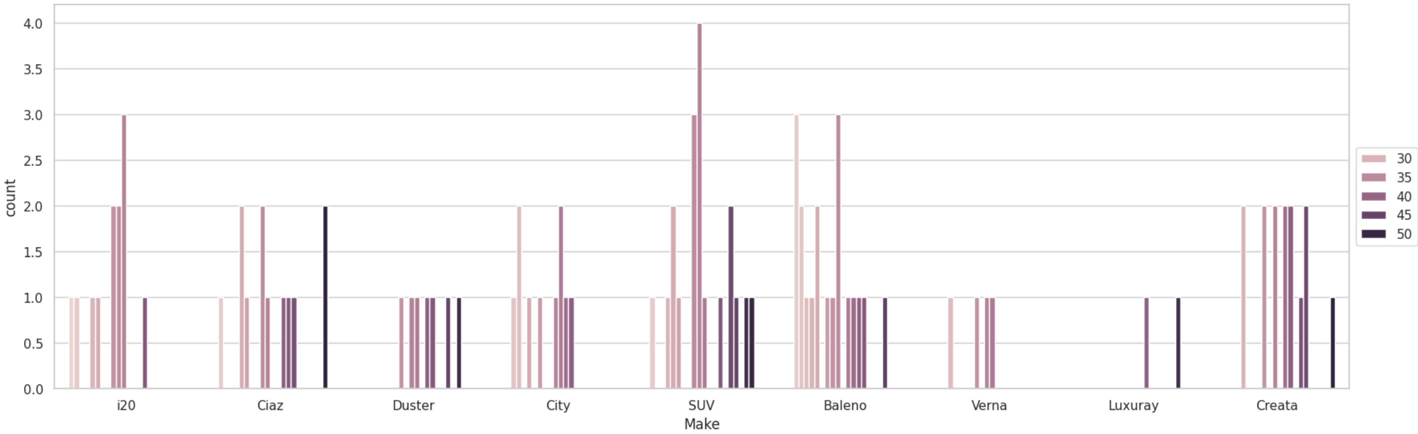
In [17]:

```
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]:

Out[17]:

<matplotlib.legend.Legend at 0x7d61947f3280>



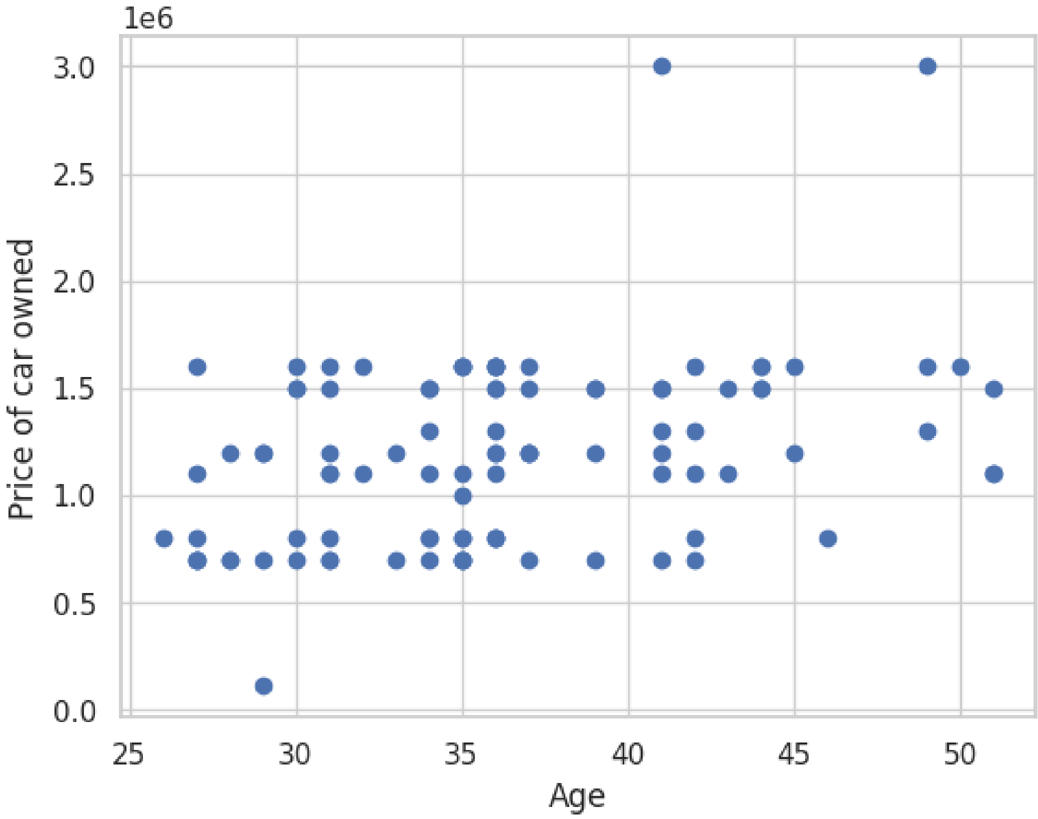

- **Price of vehicle owned**

In [18]:

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

Out[18]:

<matplotlib.collections.PathCollection at 0x7d6194c9fd30>




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

- **Make of vehicles they tend to purchase**

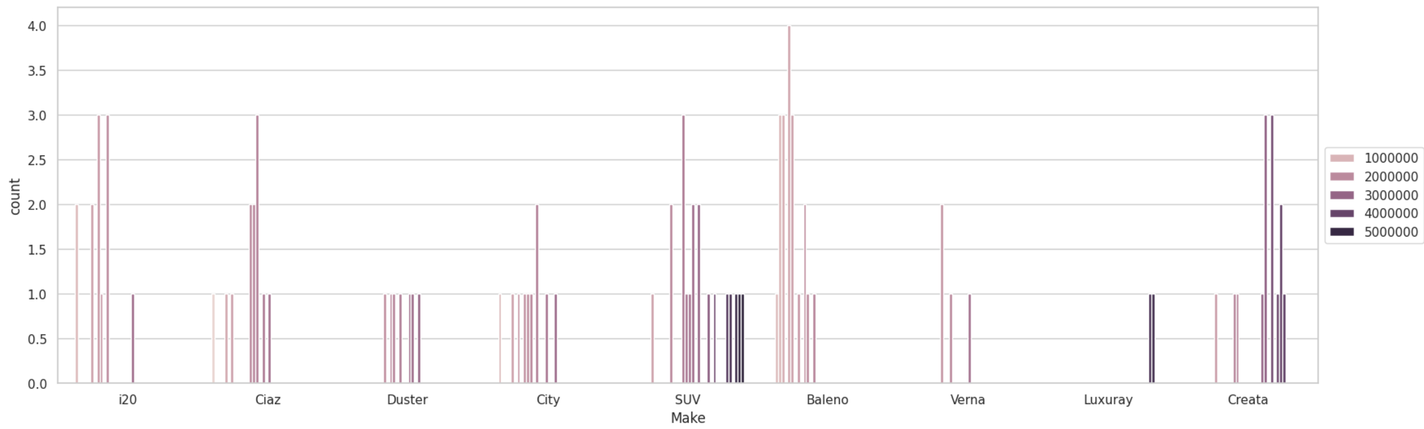
In [19]:

```
plt.figure(figsize=(20,6))
sns.countplot(x="Make", data=df, hue="Total Salary")
```

```
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

Out[19]:

```
<matplotlib.legend.Legend at 0x7d6194b047c0>
```



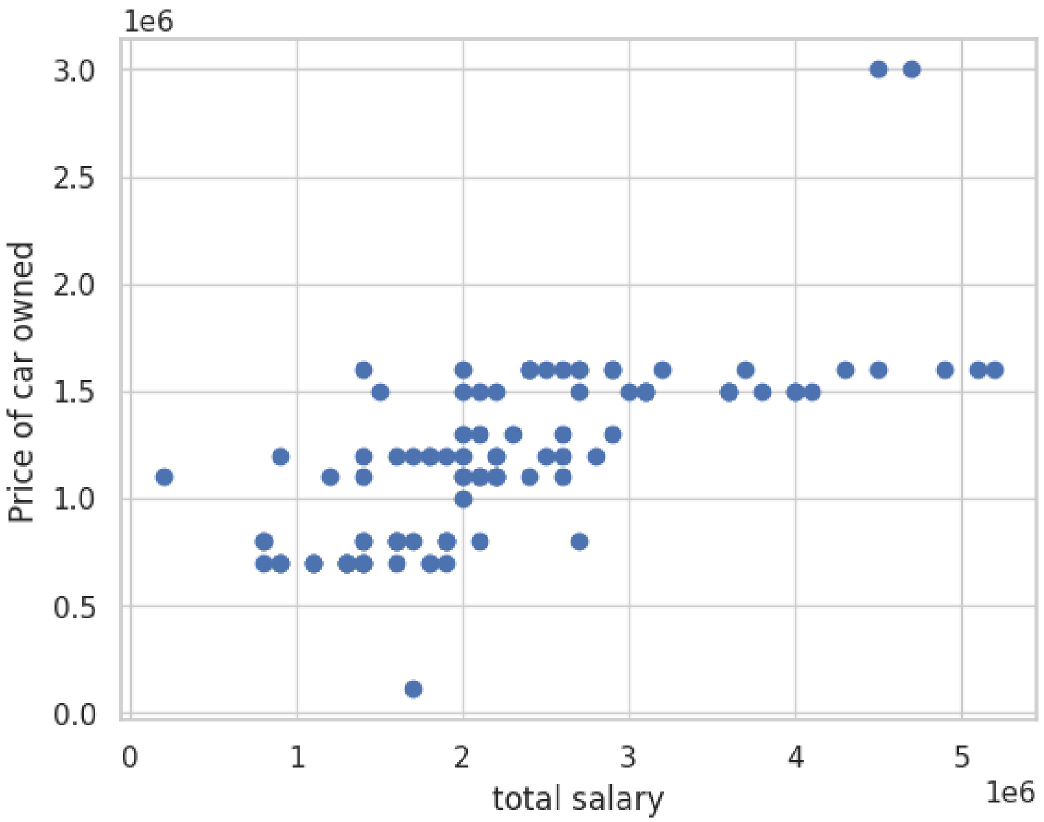

- **Price of vehicle owned**

In [20]:

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

Out[20]:

```
<matplotlib.collections.PathCollection at 0x7d6194a75150>
```




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

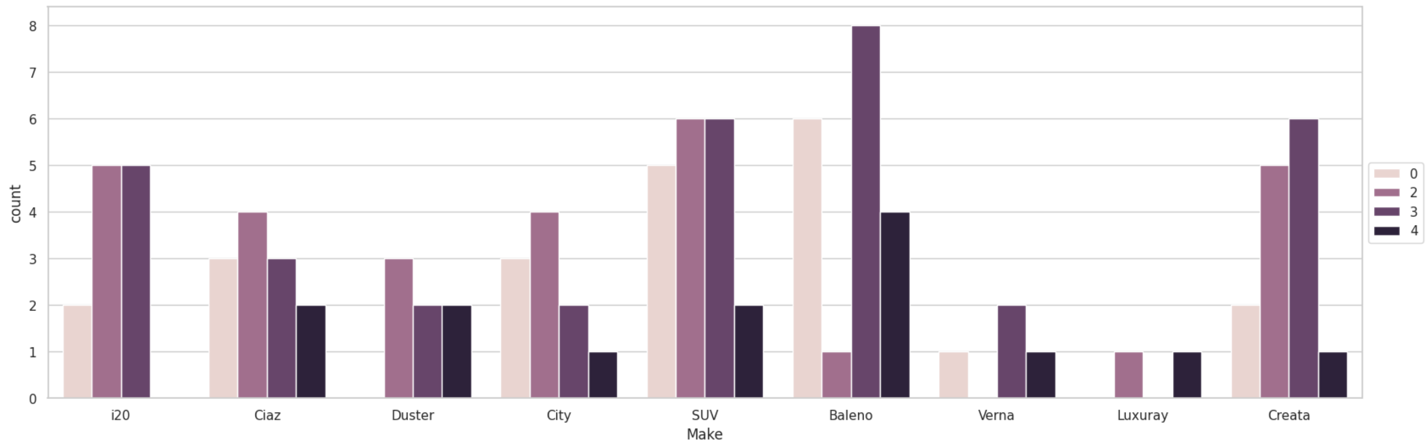
- **Make of vehicles they tend to purchase**

In [21]:

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

Out[21]:

<matplotlib.legend.Legend at 0x7d6194ab24a0>



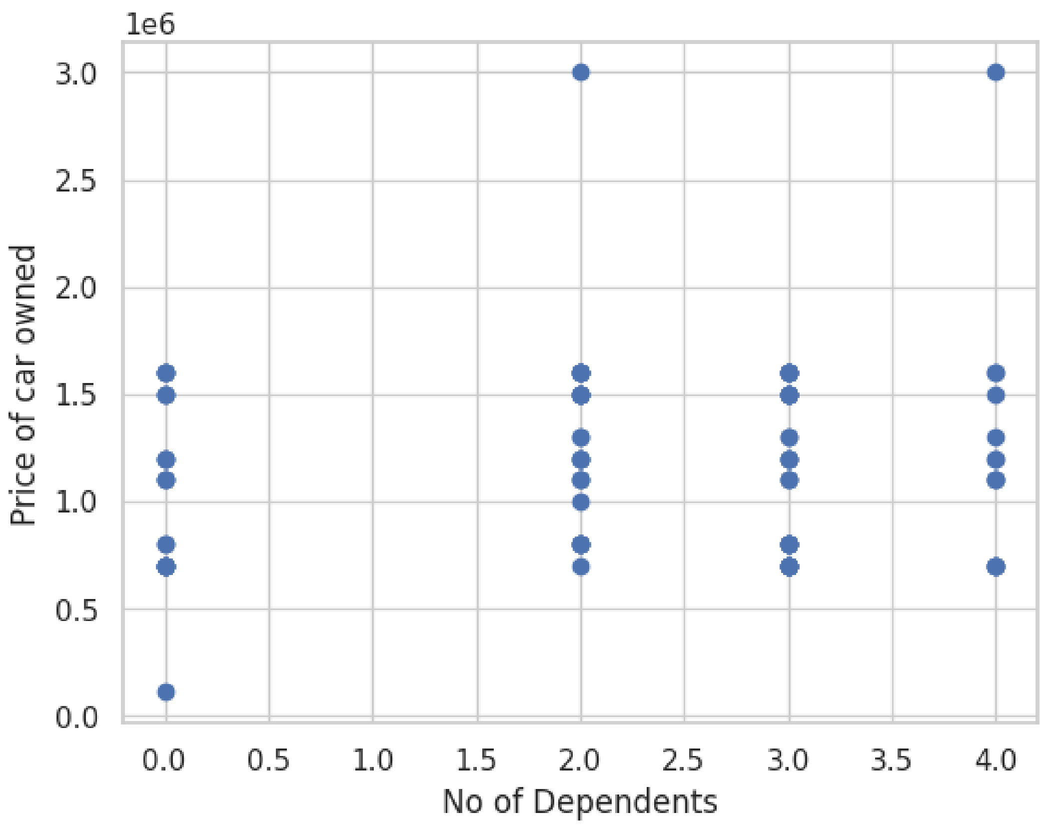

- **Price of vehicle owned**

In [22]:

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

Out[22]:

<matplotlib.collections.PathCollection at 0x7d619493d690>




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

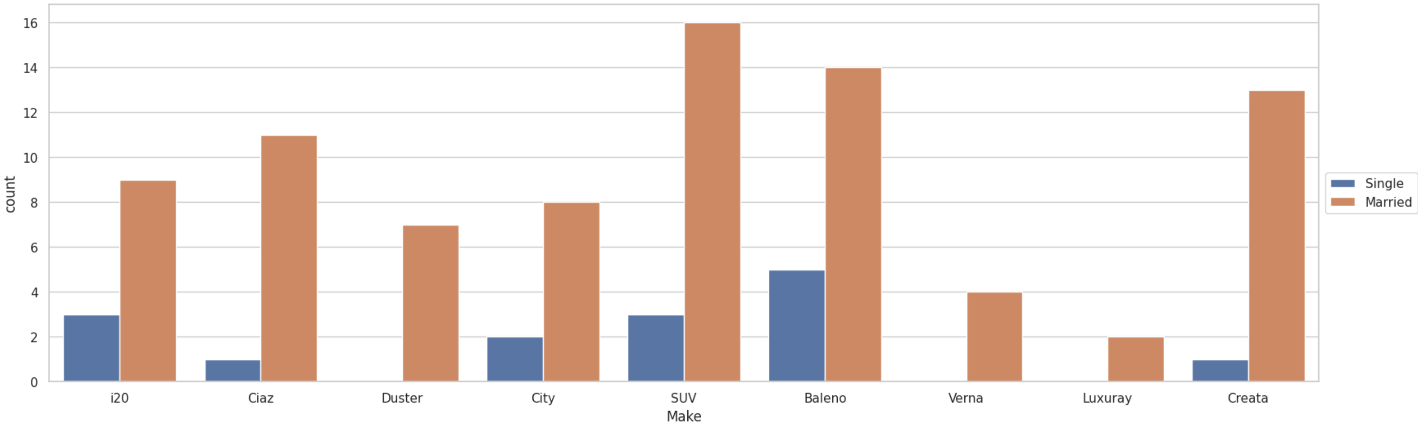
- **Make of vehicles they tend to purchase**

In [23]:

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

Out[23]:

```
<matplotlib.legend.Legend at 0x7d6194c4c040>
```



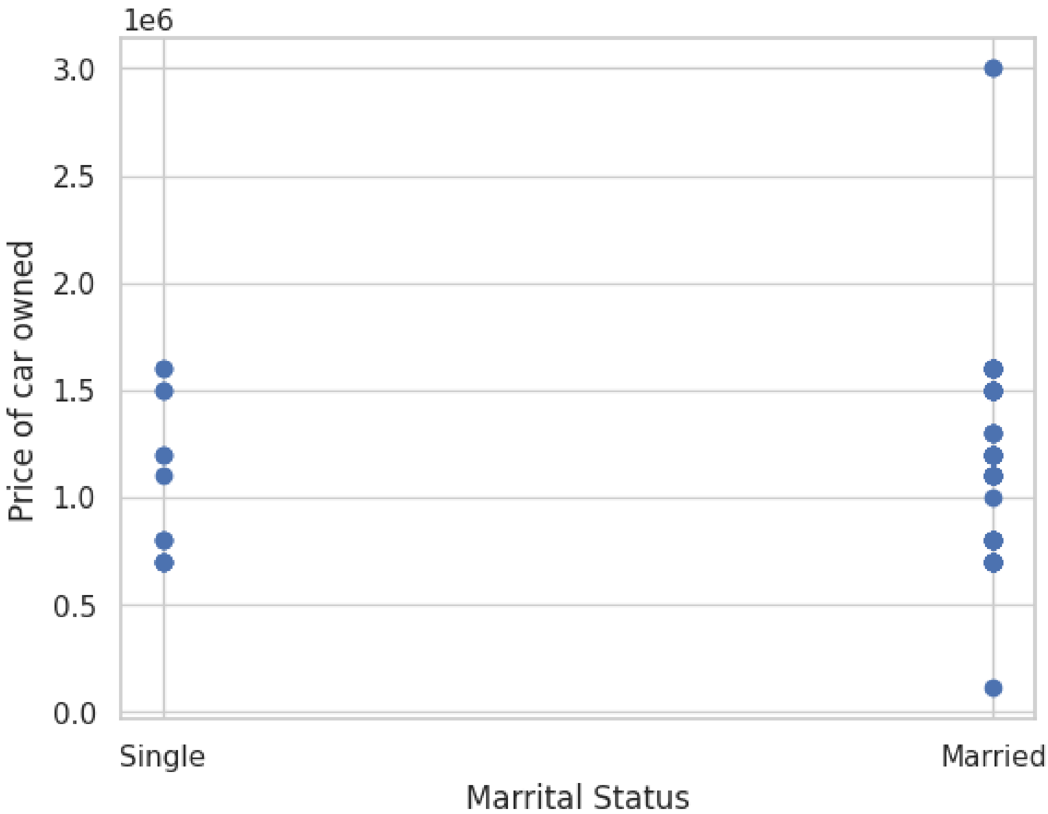

- **Price of vehicle owned**

In [24]:

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

Out[24]:

```
<matplotlib.collections.PathCollection at 0x7d6192c8a9b0>
```




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

- Make of vehicles they tend to purchase

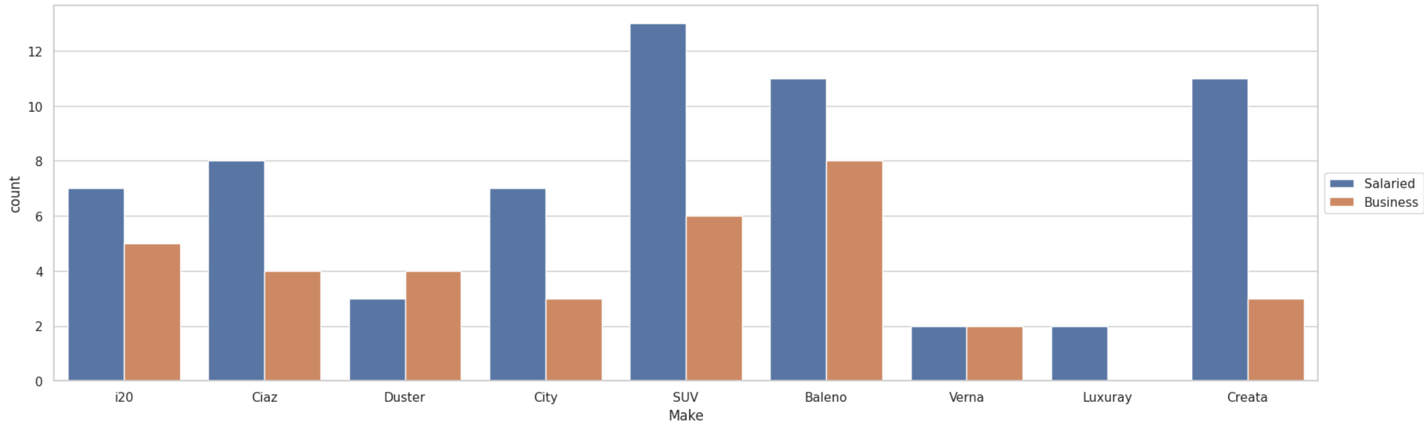
- Make of vehicles they tend to purchase

In [25]:

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

Out[25]:

<matplotlib.legend.Legend at 0x7d6192adc040>



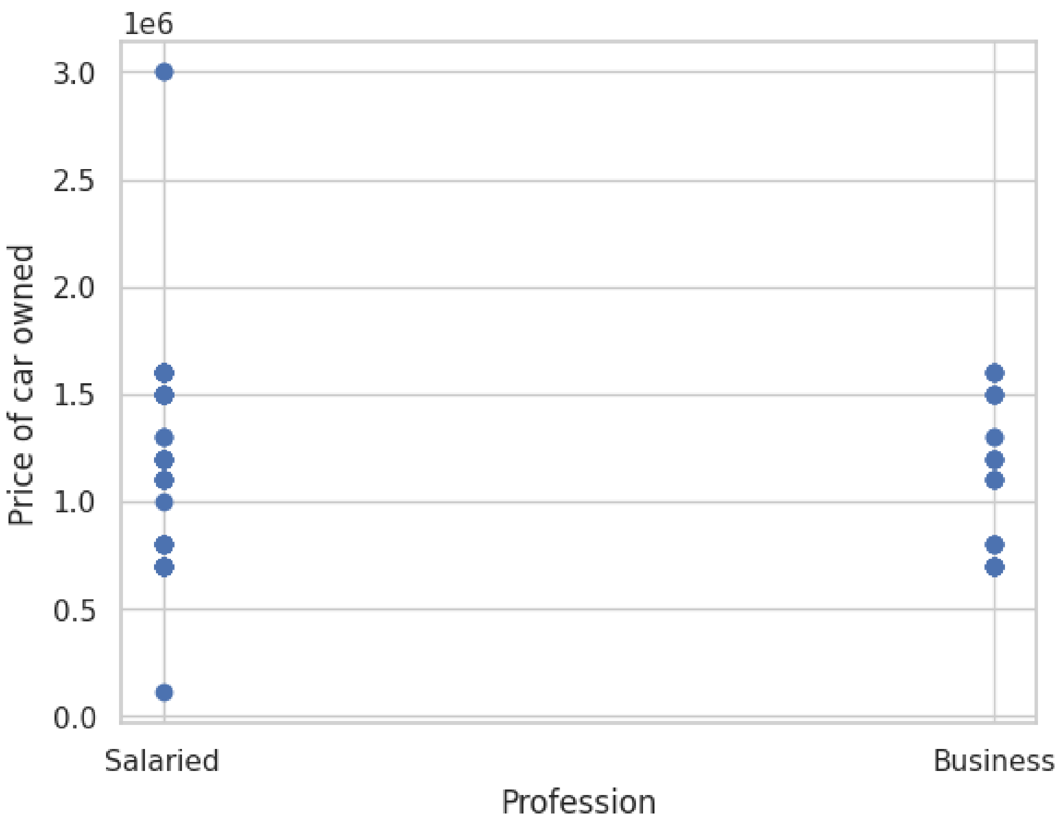

- **Price of vehicle owned**

In [26]:

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

Out[26]:

<matplotlib.collections.PathCollection at 0x7d6192b9fdc0>




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

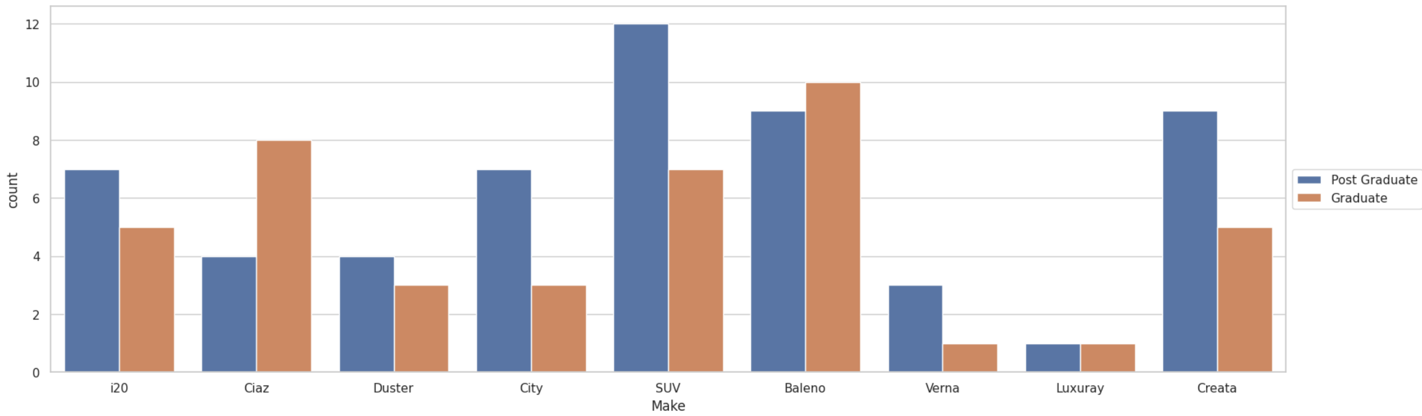
- **Make of vehicles they tend to purchase**

In [27]:

```
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[27]:

```
<matplotlib.legend.Legend at 0x7d6192b3ad10>
```



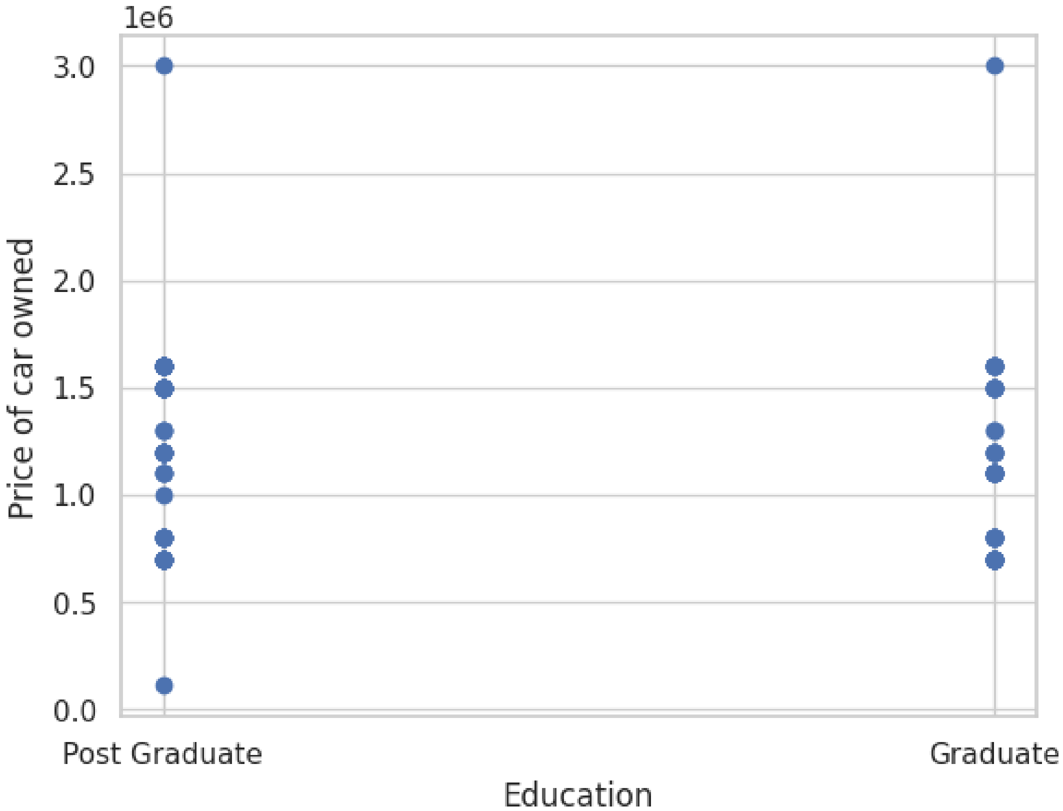

- **Price of vehicle owned**

In [28]:

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

Out[28]:

```
<matplotlib.collections.PathCollection at 0x7d6192aaeb30>
```




# 7. Relation between consumers loan status (indicator of

## purchasing power) and the vehicles they tend to purchase

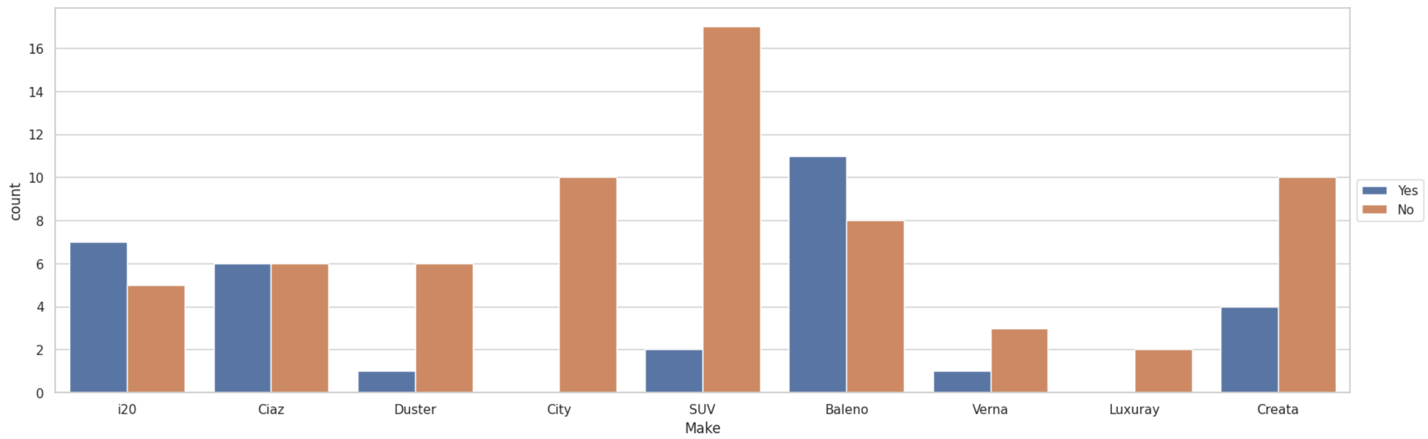
- **Make of vehicles they tend to purchase (based on personal loan)**

In [29]:

```
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[29]:

```
<matplotlib.legend.Legend at 0x7d61928f8460>
```



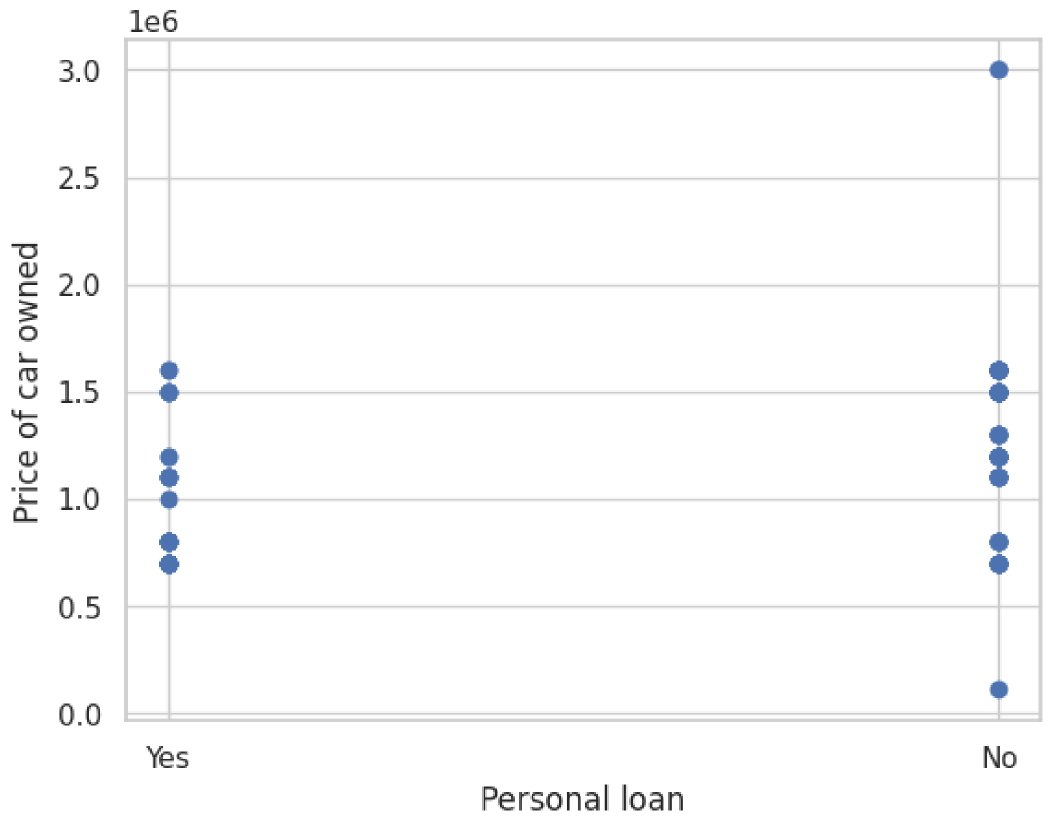

- **Price of vehicle owned (based on personal loan)**

In [30]:

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

Out[30]:

```
<matplotlib.collections.PathCollection at 0x7d61927dd1b0>
```



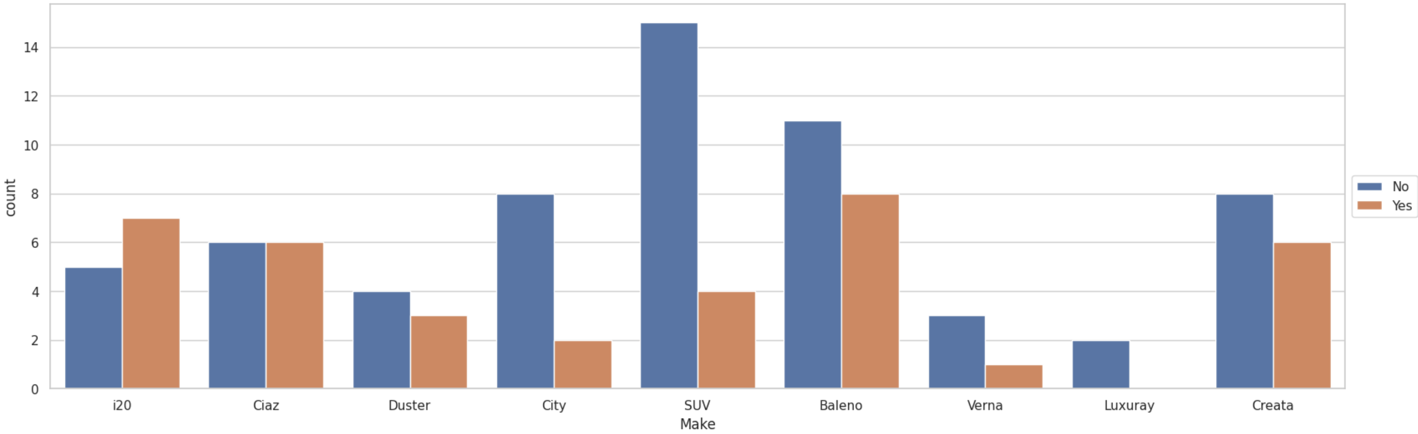
- **Make of vehicles they tend to purchase (based on house loan)**

In [31]:

```
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[31]:

<matplotlib.legend.Legend at 0x7d6192806a40>



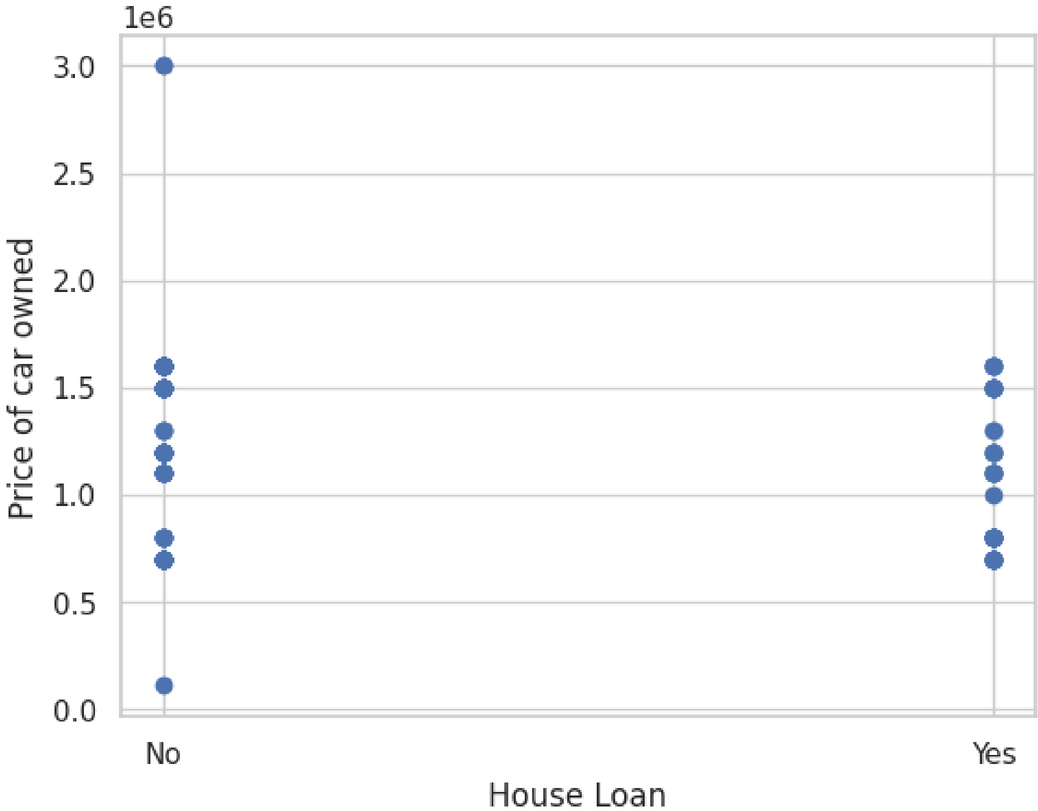

- **Price of vehicle owned (based on house loan)**

In [32]:

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

Out[32]:

<matplotlib.collections.PathCollection at 0x7d61926e6560>



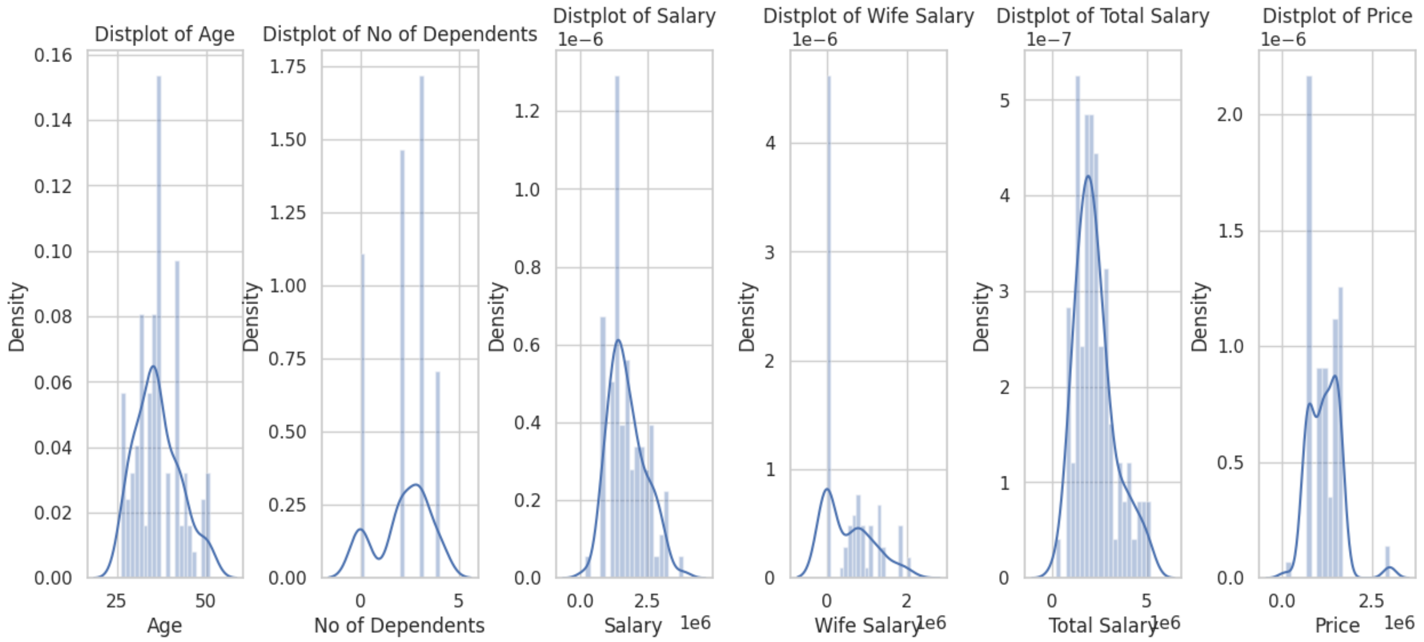

# Demographic Analysis

In [33]:

```
# Plotting for int64 dtype columns
plt.figure(1, figsize=(15,6))
n=0
for x in ['Age', 'No of Dependents' ,'Salary'  ,'Wife Salary'  ,'Total Salary'  ,'Price'
]:
  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()
```

```
<ipython-input-33-528108f5ef9c>:8: UserWarning:

`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 with
similar flexibility) or `histplot` (an axes-level function for histograms).

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

  sns.distplot(df[x], bins = 20)
<ipython-input-33-528108f5ef9c>:8: UserWarning:

`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 with
similar flexibility) or `histplot` (an axes-level function for histograms).

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

  sns.distplot(df[x], bins = 20)
<ipython-input-33-528108f5ef9c>:8: UserWarning:

`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 with
similar flexibility) or `histplot` (an axes-level function for histograms).

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

  sns.distplot(df[x], bins = 20)
<ipython-input-33-528108f5ef9c>:8: UserWarning:

`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 with
similar flexibility) or `histplot` (an axes-level function for histograms).

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

  sns.distplot(df[x], bins = 20)
<ipython-input-33-528108f5ef9c>:8: UserWarning:

`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 with
similar flexibility) or `histplot` (an axes-level function for histograms).

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

  sns.distplot(df[x], bins = 20)
<ipython-input-33-528108f5ef9c>:8: UserWarning:

`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 with
similar flexibility) or `histplot` (an axes-level function for histograms).

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

  sns.distplot(df[x], bins = 20)
```




**Observations:**

- **People between the age group of 25 to 50 create the most of the consumer market.**
- **Most people having an average total salary of around 30 lakh tend to purchase vehicles more.**
- **Most people spent around 10 to 20 lakhs for vehicles.**

In [34]:

```
# Heatmap of Correlation
sns.heatmap(df.corr(), annot=True)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-34-9c70fce8a3a5> in <cell line: 2>()
      1 # Heatmap of Correlation
----> 2 sns.heatmap(df.corr(), annot=True)

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in corr(self, method, min_pe
riods, numeric_only)
  10052         cols = data.columns
  10053         idx = cols.copy()
> 10054         mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10055
  10056         if method == "pearson":

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in to_numpy(self, dtype, cop
y, na_value)
   1836         if dtype is not None:
   1837             dtype = np.dtype(dtype)
-> 1838         result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1839         if result.dtype is not dtype:
   1840             result = np.array(result, dtype=dtype, copy=False)

/usr/local/lib/python3.10/dist-packages/pandas/core/internals/managers.py in as_array(sel
f, dtype, copy, na_value)
   1730                 arr.flags.writeable = False
   1731         else:
-> 1732             arr = self._interleave(dtype=dtype, na_value=na_value)
   1733             # The underlying data was copied within _interleave, so no need
   1734             # to further copy if copy=True or setting na_value
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/internals/managers.py in _interleave(
self, dtype, na_value)
   1792             else:
   1793                 arr = blk.get_values(dtype)
-> 1794             result[rl.indexer] = arr
   1795             itemmask[rl.indexer] = 1
   1796

ValueError: could not convert string to float: 'Salaried'
```

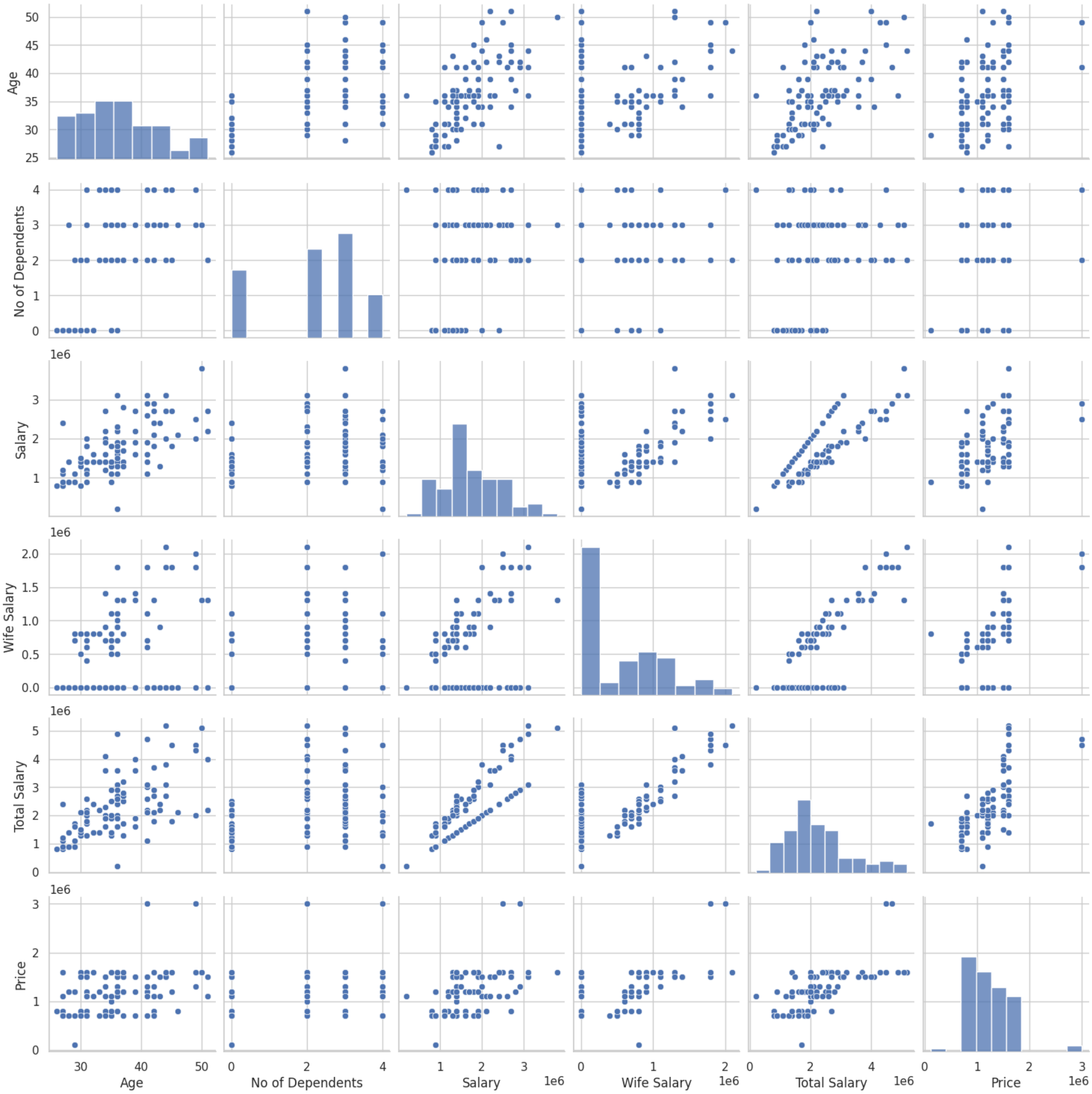
**Observations: There isn't any striking new relation found, but it confirms our previous observations.**

In [35]:

```
# Pair Plot
sns.pairplot(df)
```

Out[35]:

```
<seaborn.axisgrid.PairGrid at 0x7d619289fdf0>
```




# Geographic Analysis

# Geographic Analysis

In [36]:

```
# Importing state-wise sales dataset
data = pd.read_csv('Indian automoble buying behavour study 1.0.csv')
data
```

Out[36]:

| | Age | Profession | Marrital Status | Education | No of Dependents | Personal loan | House Loan | Wife Working | Salary | Wife Salary | Total Salary | Make | Pric |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 27 | Salaried | Single | Post Graduate | 0 | Yes | No | No | 800000 | 0 | 800000 | i20 | 80000 |
| 1 | 35 | Salaried | Married | Post Graduate | 2 | Yes | Yes | Yes | 1400000 | 600000 | 2000000 | Ciaz | 10000 |
| 2 | 45 | Business | Married | Graduate | 4 | Yes | Yes | No | 1800000 | 0 | 1800000 | Duster | 12000 |
| 3 | 41 | Business | Married | Post Graduate | 3 | No | No | Yes | 1600000 | 600000 | 2200000 | City | 12000 |
| 4 | 31 | Salaried | Married | Post Graduate | 2 | Yes | No | Yes | 1800000 | 800000 | 2600000 | SUV | 16000 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 94 | 27 | Business | Single | Graduate | 0 | No | No | No | 2400000 | 0 | 2400000 | SUV | 16000 |
| 95 | 50 | Salaried | Married | Post Graduate | 3 | No | No | Yes | 3800000 | 1300000 | 5100000 | SUV | 16000 |
| 96 | 51 | Business | Married | Graduate | 2 | Yes | Yes | No | 2200000 | 0 | 2200000 | Ciaz | 11000 |
| 97 | 51 | Salaried | Married | Post Graduate | 2 | No | No | Yes | 2700000 | 1300000 | 4000000 | Creata | 15000 |
| 98 | 51 | Salaried | Married | Post Graduate | 2 | Yes | Yes | No | 2200000 | 0 | 2200000 | Ciaz | 11000 |

99 rows × 13 columns

# Model Deployment

**K-Means Clustering**

In [49]:

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

Out[49]:

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

In [50]:

```
encoding = {"Profession":{"Salaried": 0, "Business": 1}}
```

```
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 [51]:

```
obj_df = X.replace(encoding)
obj_df.head()
```

Out[51]:

| | Age | Profession | Marrital Status | Education | No of Dependents | Personal loan | House Loan | Wife Working | Salary | Wife Salary | Total Salary | Price |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 27 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 800000 | 0 | 800000 | 800000 |
| 1 | 35 | 0 | 1 | 1 | 2 | 1 | 1 | 1 | 1400000 | 600000 | 2000000 | 1000000 |
| 2 | 45 | 1 | 1 | 0 | 4 | 1 | 1 | 0 | 1800000 | 0 | 1800000 | 1200000 |
| 3 | 41 | 1 | 1 | 1 | 3 | 0 | 0 | 1 | 1600000 | 600000 | 2200000 | 1200000 |
| 4 | 31 | 0 | 1 | 1 | 2 | 1 | 0 | 1 | 1800000 | 800000 | 2600000 | 1600000 |

# K - Means Algorithm

In [52]:

```
# Importing Important Libraries
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

In [53]:

```
X_scaled = StandardScaler().fit_transform(obj_df)
X_scaled = pd.DataFrame(X_scaled,columns=['Age', 'Profession', 'Marrital Status', 'Educa
tion', 'No of Dependents',
                                          'Personal loan', 'House Loan', 'Wife Working',
'Salary', 'Wife Salary',
                                          'Total Salary','Price'])
x = X_scaled.to_numpy()
X_scaled
```

Out[53]:

| | Age | Profession | Marrital Status | Education | No of Dependents | Personal loan | House Loan | Wife Working | Salary | Wife Salary | Total Salary | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -1.498630 | -0.739510 | -2.366432 | 0.876275 | -1.642313 | 1.446980 | -0.772512 | -1.051847 | -1.397118 | -0.887055 | -1.406760 | 0.904 |
| 1 | -0.211304 | -0.739510 | 0.422577 | 0.876275 | -0.136859 | 1.446980 | 1.294479 | 0.950708 | -0.501877 | 0.108995 | -0.258937 | 0.445 |
| 2 | 1.397855 | 1.352247 | 0.422577 | -1.141195 | 1.368594 | 1.446980 | 1.294479 | -1.051847 | 0.094950 | -0.887055 | -0.450240 | 0.013 |
| 3 | 0.754191 | 1.352247 | 0.422577 | 0.876275 | 0.615867 | -0.691095 | -0.772512 | 0.950708 | -0.203464 | 0.108995 | -0.067633 | 0.013 |
| 4 | -0.854967 | -0.739510 | 0.422577 | 0.876275 | -0.136859 | 1.446980 | -0.772512 | 0.950708 | 0.094950 | 0.441012 | 0.314975 | 0.932 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 94 | -1.498630 | 1.352247 | -2.366432 | -1.141195 | -1.642313 | -0.691095 | -0.772512 | -1.051847 | 0.990190 | -0.887055 | 0.123671 | 0.932 |
| 95 | 2.202434 | -0.739510 | 0.422577 | 0.876275 | 0.615867 | -0.691095 | -0.772512 | 0.950708 | 3.079085 | 1.271054 | 2.706274 | 0.932 |

| | Age | Profession | Marital Status | Education | No of Dependents | Personal loan | House Loan | Wife Working | Salary | Wife Salary | Total Salary | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 96 | 2.363350 | -1.352247 | 0.422577 | -1.141195 | -0.136859 | 1.446980 | 1.294479 | -1.051847 | 0.691777 | -0.887055 | -0.067633 | 0.215 |
| 97 | 2.363350 | -0.739510 | 0.422577 | 0.876275 | -0.136859 | -0.691095 | -0.772512 | 0.950708 | 1.437811 | 1.271054 | 1.654102 | 0.702 |
| 98 | 2.363350 | -0.739510 | 0.422577 | 0.876275 | -0.136859 | 1.446980 | 1.294479 | -1.051847 | 0.691777 | -0.887055 | -0.067633 | 0.215 |

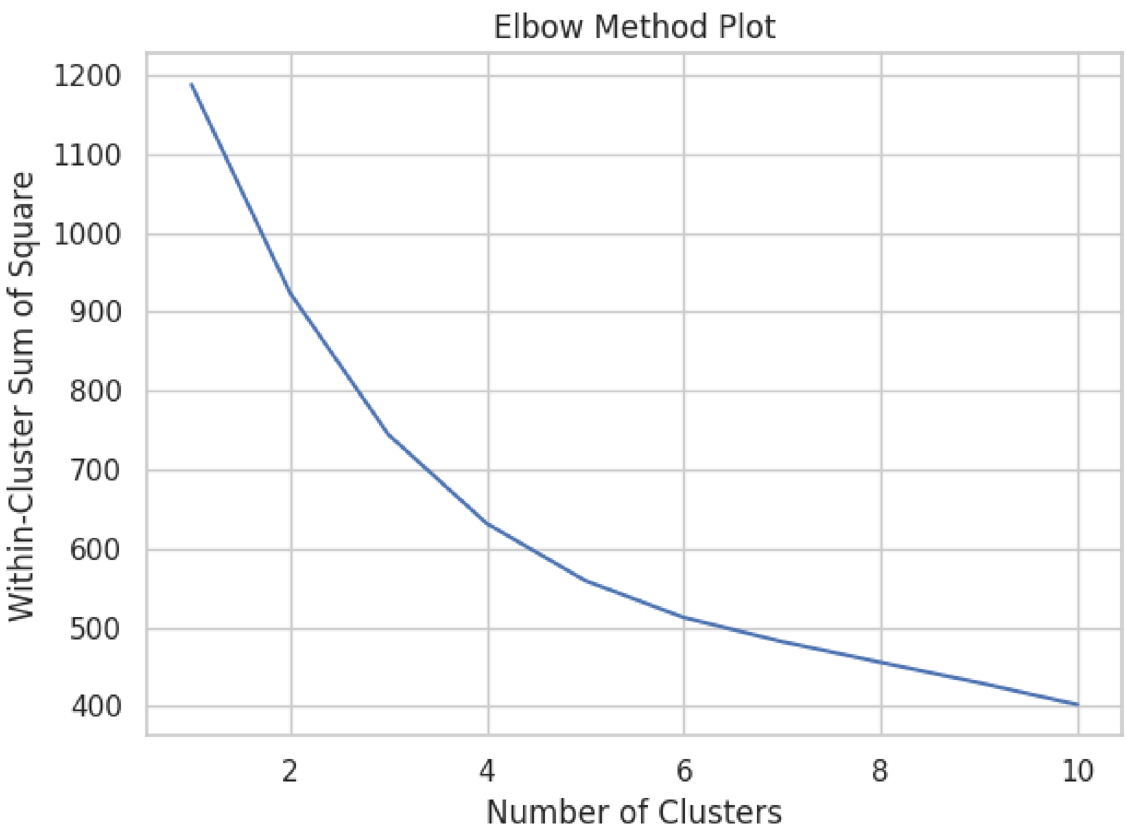
**99 rows × 12 columns**

In [54]:

```
wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
                    max_iter = 300, n_init = 10,random_state = 0)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)
```

In [55]:

```
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 [56]:

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

Out[56]:

```
▾                    KMeans
KMeans(n_clusters=3, n_init=10, random_state=42)
```

```
KMeans(n_clusters=3, n_init=10, random_state=42)
```

In [57]:

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

In [58]:

```python
final_data = pd.concat([df,y_df],axis=1)
final_data
```

Out[58]:

| | Age | Profession | Marrital Status | Education | No of Dependents | Personal loan | House Loan | Wife Working | Salary | Wife Salary | Total Salary | Make | Pric |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 27 | Salaried | Single | Post Graduate | 0 | Yes | No | No | 800000 | 0 | 800000 | i20 | 80000 |
| 1 | 35 | Salaried | Married | Post Graduate | 2 | Yes | Yes | Yes | 1400000 | 600000 | 2000000 | Ciaz | 100000 |
| 2 | 45 | Business | Married | Graduate | 4 | Yes | Yes | No | 1800000 | 0 | 1800000 | Duster | 120000 |
| 3 | 41 | Business | Married | Post Graduate | 3 | No | No | Yes | 1600000 | 600000 | 2200000 | City | 120000 |
| 4 | 31 | Salaried | Married | Post Graduate | 2 | Yes | No | Yes | 1800000 | 800000 | 2600000 | SUV | 160000 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 94 | 27 | Business | Single | Graduate | 0 | No | No | No | 2400000 | 0 | 2400000 | SUV | 160000 |
| 95 | 50 | Salaried | Married | Post Graduate | 3 | No | No | Yes | 3800000 | 1300000 | 5100000 | SUV | 160000 |
| 96 | 51 | Business | Married | Graduate | 2 | Yes | Yes | No | 2200000 | 0 | 2200000 | Ciaz | 110000 |
| 97 | 51 | Salaried | Married | Post Graduate | 2 | No | No | Yes | 2700000 | 1300000 | 4000000 | Creata | 150000 |
| 98 | 51 | Salaried | Married | Post Graduate | 2 | Yes | Yes | No | 2200000 | 0 | 2200000 | Ciaz | 110000 |

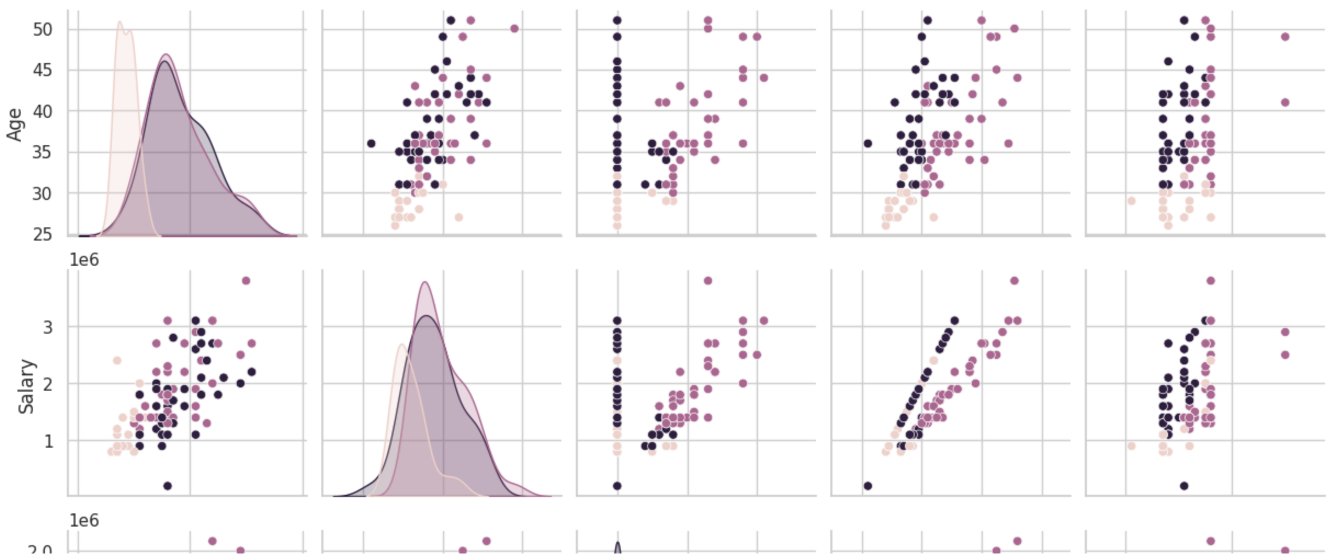
**99 rows × 14 columns**

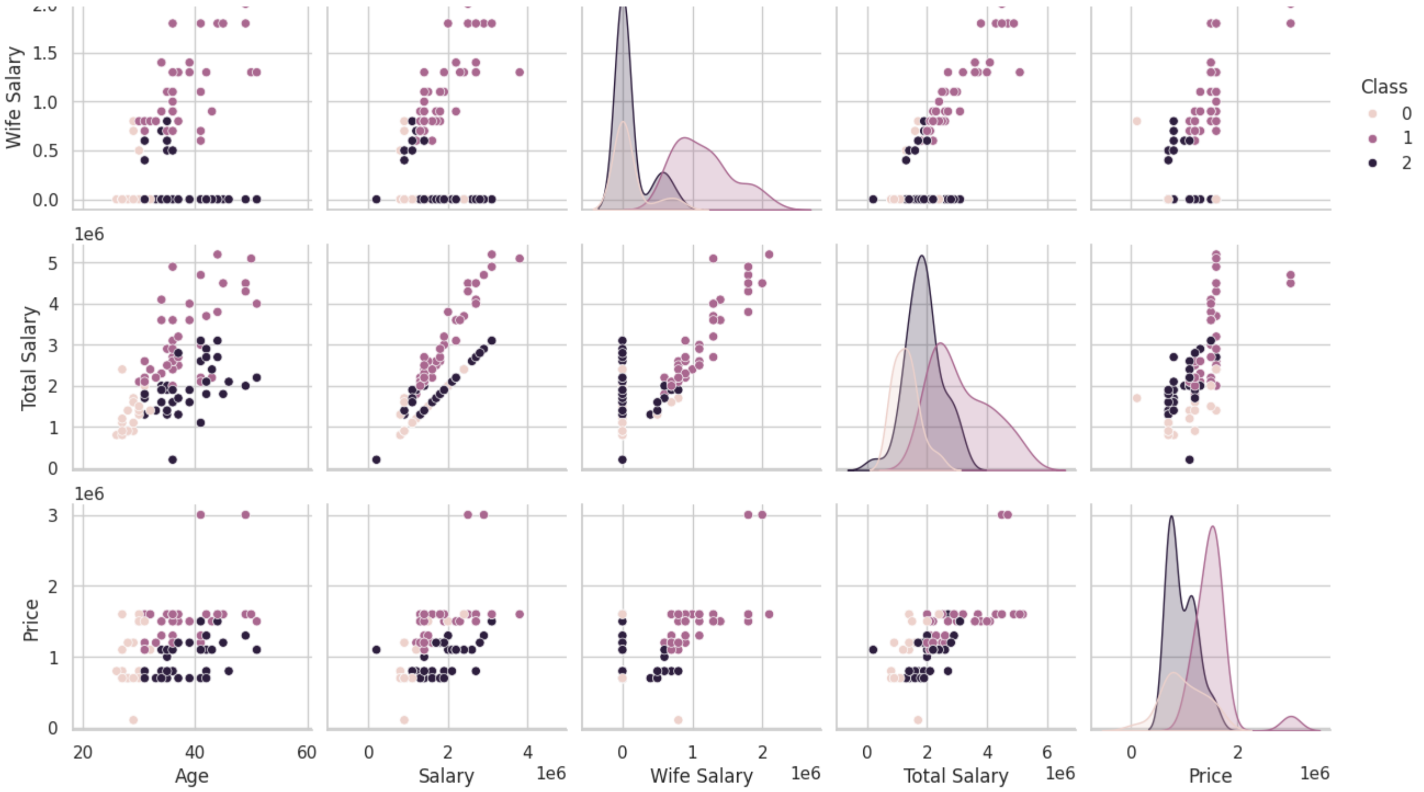
In [59]:

```python
sns.pairplot(final_data,x_vars = ['Age','Salary', 'Wife Salary','Total Salary','Price'],
y_vars = ['Age','Salary', 'Wife Salary','Total Salary','Price'], hue='Class')
```

Out[59]:

```
<seaborn.axisgrid.PairGrid at 0x7d618a127010>
```






**K = 5**

In [60]:

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

Out[60]:

```
▾                    KMeans
KMeans(n_clusters=5, n_init=10, random_state=42)
```

In [61]:

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

In [62]:

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

Out[62]:

| | Age | Profession | Marrital Status | Education | No of Dependents | Personal loan | House Loan | Wife Working | Salary | Wife Salary | Total Salary | Make | Pric |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 27 | Salaried | Single | Post Graduate | 0 | Yes | No | No | 800000 | 0 | 800000 | i20 | 80000 |
| **1** | 35 | Salaried | Married | Post Graduate | 2 | Yes | Yes | Yes | 1400000 | 600000 | 2000000 | Ciaz | 100000 |
| **2** | 45 | Business | Married | Graduate | 4 | Yes | Yes | No | 1800000 | 0 | 1800000 | Duster | 120000 |
| **3** | 41 | Business | Married | Post Graduate | 3 | No | No | Yes | 1600000 | 600000 | 2200000 | City | 120000 |
| **4** | 31 | Salaried | Married | Post Graduate | 2 | Yes | No | Yes | 1800000 | 800000 | 2600000 | SUV | 160000 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **94** | 27 | Business | Single | Graduate | 0 | No | No | No | 2400000 | 0 | 2400000 | SUV | 160000 |

| | Age | Profession | Marrital Status | Education | No of Dependents | Personal loan | House Loan | Wife Working | Salary | Wife Salary | Total Salary | Make | Pric |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 95 | 50 | Salaried | Married | Post Graduate | 3 | No | No | Yes | 3800000 | 1300000 | 5100000 | SUV | 160000 |
| 96 | 51 | Business | Married | Graduate | 2 | Yes | Yes | No | 2200000 | 0 | 2200000 | Ciaz | 110000 |
| 97 | 51 | Salaried | Married | Post Graduate | 2 | No | No | Yes | 2700000 | 1300000 | 4000000 | Creata | 150000 |
| 98 | 51 | Salaried | Married | Post Graduate | 2 | Yes | Yes | No | 2200000 | 0 | 2200000 | Ciaz | 110000 |

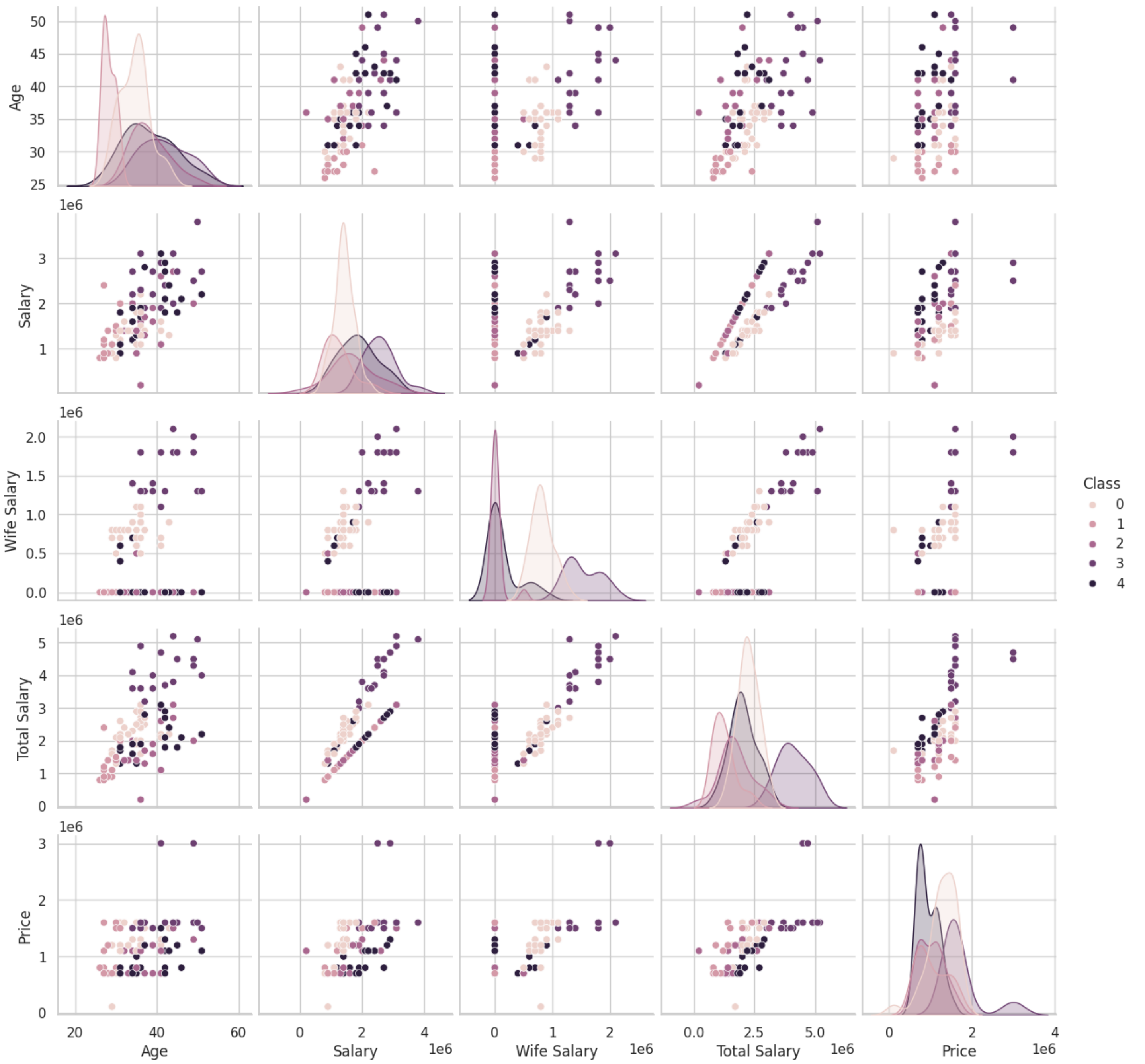
99 rows × 14 columns

In [63]:

```
sns.pairplot(final_data1,x_vars = ['Age','Salary', 'Wife Salary','Total Salary','Price'],
y_vars = ['Age','Salary', 'Wife Salary','Total Salary','Price'], hue='Class')
```

Out[63]:

```
<seaborn.axisgrid.PairGrid at 0x7d618967edd0>
```




In [ ]: