Step 1->Creating Dataframe

In [82]:

```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
df = pd.read_csv('CAR DETAILS FROM CAR DEKHO.csv')
df
```

Out[82]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
...	...	...	...	...	...	...	...	...
4335	Hyundai i20 Magna 1.4 CRDi (Diesel)	2014	409999	80000	Diesel	Individual	Manual	Second Owner
4336	Hyundai i20 Magna 1.4 CRDi	2014	409999	80000	Diesel	Individual	Manual	Second Owner
4337	Maruti 800 AC BSIII	2009	110000	83000	Petrol	Individual	Manual	Second Owner
4338	Hyundai Creta 1.6 CRDi SX Option	2016	865000	90000	Diesel	Individual	Manual	First Owner
4339	Renault KWID RXT	2016	225000	40000	Petrol	Individual	Manual	First Owner

4340 rows × 8 columns

Step 2->Performing Exploratory Data Analysis (EDA)

In [83]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   name           4340 non-null   object
 1   year           4340 non-null   int64 
 2   selling_price  4340 non-null   int64 
 3   km_driven      4340 non-null   int64 
 4   fuel           4340 non-null   object
 5   seller_type    4340 non-null   object
 6   transmission   4340 non-null   object
 7   owner          4340 non-null   object
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
```

In [84]:

```
# Checking the dimensions of the dataset
df.shape
```

Out[84]:

```
(4340, 8)
```

In [85]:

```
# Checking number of entries in the dataset
df.size
```

Out[85]:

```
34720
```

In [86]:

```
df.describe()
```

Out[86]:

	year	selling_price	km_driven
count	4340.000000	4.340000e+03	4340.000000
mean	2013.090783	5.041273e+05	66215.777419
std	4.215344	5.785487e+05	46644.102194
min	1992.000000	2.000000e+04	1.000000
25%	2011.000000	2.087498e+05	35000.000000
50%	2014.000000	3.500000e+05	60000.000000
75%	2016.000000	6.000000e+05	90000.000000
max	2020.000000	8.900000e+06	806599.000000

In [87]:

```
# Checking for null values in our data set
df.isnull().sum()
```

Out[87]:

```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
dtype: int64
```

In [88]:

```
print("Unique Values for the following dataset:")
print("name: "+str(df["name"].nunique()))
print("year: "+str(df["year"].nunique()))
print("selling_price : "+str(df["selling_price"].nunique()))
print("km_driven: "+str(df["km_driven"].nunique()))
print("fuel: "+str(df["fuel"].nunique()))
print("seller_type : "+str(df["seller_type"].nunique()))
print("transmission : "+str(df["transmission"].nunique()))
print("owner: "+str(df["owner"].nunique()))
```

```
Unique Values for the following dataset:
name: 1491
year: 27
selling_price : 445
km_driven: 770
fuel: 5
seller_type : 3
transmission : 2
owner: 5
```

In [89]:

```
print("Values that are unique: ")
print("name: ",df["name"].unique())
print("year: ",df["year"].unique())
print("selling_price : ",df["selling_price"].unique())
print("km_driven: ",df["km_driven"].unique())
print("fuel: ",df["fuel"].unique())
print("seller_type : ",df["seller_type"].unique())
print("transmission : ",df["transmission"].unique())
print("owner: ",df["owner"].unique())
```

```
Values that are unique:
name:  ['Maruti 800 AC' 'Maruti Wagon R LXI Minor' 'Hyundai Verna 1.6 SX' ...
 'Mahindra Verito 1.5 D6 BSIII'
 'Toyota Innova 2.5 VX (Diesel) 8 Seater BS IV'
 'Hyundai i20 Magna 1.4 CRDi']
year:  [2007 2012 2017 2014 2016 2015 2018 2019 2013 2011 2010 2009 2006 1996
 2005 2008 2004 1998 2003 2002 2020 2000 1999 2001 1995 1997 1992]
selling_price :  [  60000  135000  600000  250000  450000  140000  550000  240000  85000
0
  365000  260000 1650000  585000 1195000  390000 1964999 1425000  975000
 1190000  930000  525000 1735000 1375000  900000 1300000 1400000  229999
 1550000 1250000  625000 1050000  560000  290000  275000  411000  150000
  500000  100000  725000  401000  750000  310000  665000  465000  160000
  675000  300000   70000  151000  280000  350000  570000  125000  130000
  925000  200000  248000   80000  650000  495000  371000 1025000 8150000
  325000 1470000 2800000  210000 1150000 4500000 2750000 1975000  175000
 2500000  628000  399000  315000  780000  434000  690000  555000  120000
  165000   95000  800000  840000  490000  400000 1000000  530000   40000
   75000  540000  700000  430000   65000  195000  170000  225000   99000
  620000 2550000  320000  810000  282000   72000  640000  380000 1500000
  434999  190000 2900000  425000  265000  890000  685000  940000  590000
  385000 2000000  235000   52000   89999  180000  285000 1075000   90000
  220000  110000  880000  115999  360000  680000  860000  270000  395000
  624000  345000  106000 1800000  575000  370000   50000   55000  755000
  720000 1100000  159000  335000  185000  470000  145000  595000 1600000
  105000  409999  215000  475000  330000 1044999 1350000  420000  760000
   43000 1850000 1125000  133000  352000  520000  509999  556000  484999
  565000  295000 2050000 1475000 4400000  670000  770000  775000 1725000
 2150000 3800000 1580000 4950000  535000  239000 2600000  114999  200999
  710000  969999  155000  138000  311000   58000  183000  825000  299000
  639000  415000 1199000  699000  269000  249000 1549000  254999  211000
  599000 4000000 1200000   98000  790000 1700000   68000  875000 1330000
  919999  611000  711000  851000  610000  744000  480000  950000   85000
  615000  227000  222000  735000  271000 1490000  455000  421000 2700000
 4700000 1900000 1770000  660000  716000  147000 1140000 3050000  375000
 1950000  340000 3100000  245000  715000 1750000 3500000  835000 2490000
 1015000   91200 2400000  635000  302000  204999  341000  819999  351000
  630000 1085000  580000   78000 3200000  695000  355000  619000   81000
  486000  802000 2300000  287000  250999   45000 1485000 1825000 3256000
  451000  149000  163000  419000  990000  346000  509000   69000 1380000
  256000   97000  199000 2595000  730000  368000  545000  641000  784000
  324000 2100000  305000  221000  828999 1119000  746000 1030000 1334000
  811999 1331000  852000  830000  213000   35000  869999  178000  515000
  312000  111000  774000  148000   57000  284000  349000  458000  381000
  751000  782000  321000   92800  291000   73000  655000  263000  217000
  539000  142000  910000  740000  164000  999000   56000 3899000  440000
  238000 1295000  541000  894999  844999  288000 1225000 1010000   30000
  396000  281000   93000  459999   88000   22000   79000  198000  182000
  861999  836000  696000  596000  612000   20000   61000  511000 1230000
  426000   62000 1450000   71000 2200000 1249000 1240000 1068000 1189000
  363000  821000  815000  738000  765000  516000  134000  347000 2650000
 2675000  359000  980000  707000  471000  377000  763000  701000  277000
  936999   82000  799000 1451000 1575000   78692  479000   48000  121000
  785000  173000 4800000  587000  123000 1290000  193000  721000 1040000
 2349000 1165000   42000 1680000  231999  841000 1280000 1090000  449000
  724000  126000  795000 2575000 1035000 1260000 8900000 1860000 4200000
 5500000  430999 1151000  927999   51111  212000  428000  219000  749000
  233000  614000   37500  865000]
km_driven:  [ 70000  50000 100000  46000 141000 125000  25000  60000  78000  35000
  24000   5000  33000  28000  59000   4500 175900  14500  15000  33800
 130400  80000  10000 119000  75800  40000  74000  64000 120000  79000
  18500  10200  29000  90000  73300  92000  66764 350000 230000  31000
  39000 166000 110000  54000  63000  76000  11958  20000   9000   6500
```

39000 166000 110000 54000 63000 76000 11958 20000 9000 6500
58000 62200 34000 53000 49000 63500 9800 13000 21000 29173
48000 30000 87000 16000 79350 81000 3600 14272 49213 57000
3240 114000 53772 140000 175000 36000 155500 23000 22155 78380
150000 80362 55000 1136 43000 2650 115962 65000 56000 213000
139000 160000 163000 32000 52000 11240 66000 26500 72000 44000
130000 195000 155000 4000 41000 10832 14681 51000 200000 19600
46730 21170 167223 141440 212814 88635 149674 8000 68000 38000
75000 98000 81925 82080 97000 52047 62009 33100 220000 45000
180000 22000 80577 127500 40903 22288 61690 64484 75976 85962
57035 72104 164000 124439 77000 1250 17152 24005 149000 19000
109000 61000 27633 12586 38083 55328 81632 155201 93283 217871
90165 101504 86017 85036 91086 160254 125531 82000 84000 560000
14365 61083 66363 11700 7104 45974 55340 61585 39415 29654
64672 54634 66521 23974 1000 86000 52600 19890 11918 10510
47162 49824 58500 56580 46507 11451 172000 66508 29900 3000
85000 7900 17500 206500 88600 186000 11000 138000 27974 18000
1400 124000 42000 28205 32670 30093 56228 59319 39503 35299
51687 76259 44049 45087 41125 42215 54206 52547 59110 54565
47564 45143 61624 132000 10980 20629 69782 63654 59385 70378
55425 78413 40890 34823 55545 56541 43700 27483 56207 1440
91195 63657 97248 89000 12000 12997 26430 24600 28481 41988
30375 7658 34400 28942 53600 53652 106000 205000 79500 197000
9161 19077 128000 21302 10500 107000 55300 74300 48781 87620
40219 11473 8352 9745 9748 20694 31080 37605 55850 58850
23839 45454 46957 190000 1500 47000 116000 26350 71042 167870
133564 23038 43608 11212 49217 28838 135000 19571 29600 13500
48600 127643 102354 62237 21394 32686 1001 53261 14000 39895
73000 17000 18591 26766 300000 27620 223000 161327 6000 71000
144000 37000 26000 27000 13250 101000 8500 90246 60400 70950
1100 31491 107143 46412 107500 43826 55838 112880 30300 80659
81324 127884 66755 123084 806599 95851 234000 170000 96000 19014
23262 35925 40771 30500 55800 66569 81358 82695 68293 190621
64700 88470 126000 74183 1 192000 83411 13270 88000 7000
13770 102000 143000 115000 136906 133000 28689 80322 61658 185000
30600 235000 67000 74500 118700 223660 2000 73756 16400 41395
71014 181000 89550 149500 83000 44800 156000 146000 99000 37516
25880 136000 2020 94000 88500 52536 1950 118400 6480 32077
19107 18469 28217 72787 31063 79641 58692 54784 64156 9500
81366 244000 312000 145000 7300 72539 101 52328 91505 20500
154000 41723 68745 27289 24662 28245 27005 39227 31367 35008
100005 45264 39093 45241 2769 43128 22255 59213 1010 1111
48965 5166 76290 45766 78771 79357 76736 92645 101849 155836
63230 1758 1452 35122 92621 92198 152000 78322 54309 34600
38217 77073 16584 81257 3917 69069 59059 39039 33033 55168
41041 67067 66066 82082 70070 63063 9528 135200 50300 151624
74820 129000 66778 63400 157000 38500 103921 14825 43377 102307
245244 68500 5007 49600 43100 10171 41123 20118 52517 99117
3700 43500 137250 5400 11200 93000 62000 5800 267000 250000
28635 32114 95149 68458 105546 104000 132343 26134 52895 42324
60236 10300 142000 28643 7600 47253 4432 68523 80251 34500
42743 93900 55766 113600 138925 121764 105429 23122 44500 13599
5200 12700 95000 45839 74510 87293 156040 93415 101159 68519
55130 65239 58182 91245 102989 108000 178000 75118 4637 42655
69000 117000 105000 182000 24585 13900 17563 173000 151000 117780
81595 9700 221000 28740 48500 148620 270000 41090 296823 89255
168000 5550 1700 45217 44440 91365 90010 31800 59100 31200
22700 50900 2417 65500 140300 10211 260000 32933 54551 57112
41025 53122 64111 78892 74113 84775 20778 64441 43192 44416
79991 62601 89600 60800 69111 20969 20194 34982 44588 57904
59258 60826 1300 31707 115992 109052 90658 25552 40700 11174
72500 76600 97700 37500 23800 44077 210000 9422 240000 17100
224642 222435 159000 101100 1200 134444 238000 165000 63700 74800
60516 76731 63840 76400 31489 295000 158000 400000 19495 62668
85710 63356 129627 4400 14987 25061 42494 44875 89741 347089
222252 55250 12500 162000 22038 2500 89126 134000 42500 131365
48980 98900 13800 99700 49654 45457 39221 48220 11114 60208
98600 85441 64541 16267 71500 12999 14700 92686 49359 108731
29976 30646 23600 71318 78098 18054 38406 54350 32260 58231
59858 73350 88473 96987 77350 61187 68350 81150 280000 105700
37091 38900 9400 14100 37555 56600 67580 48238 38365 23670
49834 57353 68308 63240 64916 37161 118000 50852 53500 51500
79800 6590 49957 43235 50699 140730 256000 218000 66782 1121981

```
 79800   6590  49957  45255  50699 140750 256000 218000  66782 112198]
fuel:  ['Petrol' 'Diesel' 'CNG' 'LPG' 'Electric']
seller_type :  ['Individual' 'Dealer' 'Trustmark Dealer']
transmission :  ['Manual' 'Automatic']
owner:  ['First Owner' 'Second Owner' 'Fourth & Above Owner' 'Third Owner'
 'Test Drive Car']
```

In [90]:

```
#Count of occurences for every unique element in "name" column
df.groupby("name",sort=False).size()
```

Out[90]:

```
name
Maruti 800 AC                                 23
Maruti Wagon R LXI Minor                      24
Hyundai Verna 1.6 SX                          15
Datsun RediGO T Option                         7
Honda Amaze VX i-DTEC                          9
                                              ..
Maruti Swift LDI                               1
Tata Nano XM                                   1
Mahindra Verito 1.5 D6 BSIII                   1
Toyota Innova 2.5 VX (Diesel) 8 Seater BS IV   1
Hyundai i20 Magna 1.4 CRDi                     1
Length: 1491, dtype: int64
```

In [91]:

```
#Count of occurences for every unique element in "year" column
df.groupby("year",sort=False).size()
```

Out[91]:

```
year
2007    134
2012    415
2017    466
2014    367
2016    357
2015    421
2018    366
2019    195
2013    386
2011    271
2010    234
2009    193
2006    110
1996      2
2005     85
2008    145
2004     42
1998     12
2003     23
2002     21
2020     48
2000     12
1999     10
2001     20
1995      1
1997      3
1992      1
dtype: int64
```

In [92]:

```
#Count of occurences for every unique element in "km_driven" column
df.groupby("km_driven",sort=False).size()
```

Out[92]:

```
km_driven
70000     236
```

```
70000     236
50000     222
100000    180
46000      16
141000      2
         ...
140730      1
256000      1
218000      2
66782       1
112198      1
Length: 770, dtype: int64
```

In [93]:

```
#Count of occurences for every unique element in "fuel" column
df.groupby("fuel",sort=False).size()
```

Out[93]:

```
fuel
Petrol      2123
Diesel      2153
CNG           40
LPG           23
Electric       1
dtype: int64
```

In [94]:

```
#Count of occurences for every unique element in "seller_type" column
df.groupby("seller_type",sort=False).size()
```

Out[94]:

```
seller_type
Individual          3244
Dealer               994
Trustmark Dealer     102
dtype: int64
```

In [95]:

```
#Count of occurences for every unique element in "transmission" column
df.groupby("transmission",sort=False).size()
```

Out[95]:

```
transmission
Manual       3892
Automatic     448
dtype: int64
```

In [96]:

```
#Count of occurences for every unique element in "owner" column
df.groupby("owner",sort=False).size()
```

Out[96]:

```
owner
First Owner             2832
Second Owner            1106
Fourth & Above Owner      81
Third Owner              304
Test Drive Car            17
dtype: int64
```

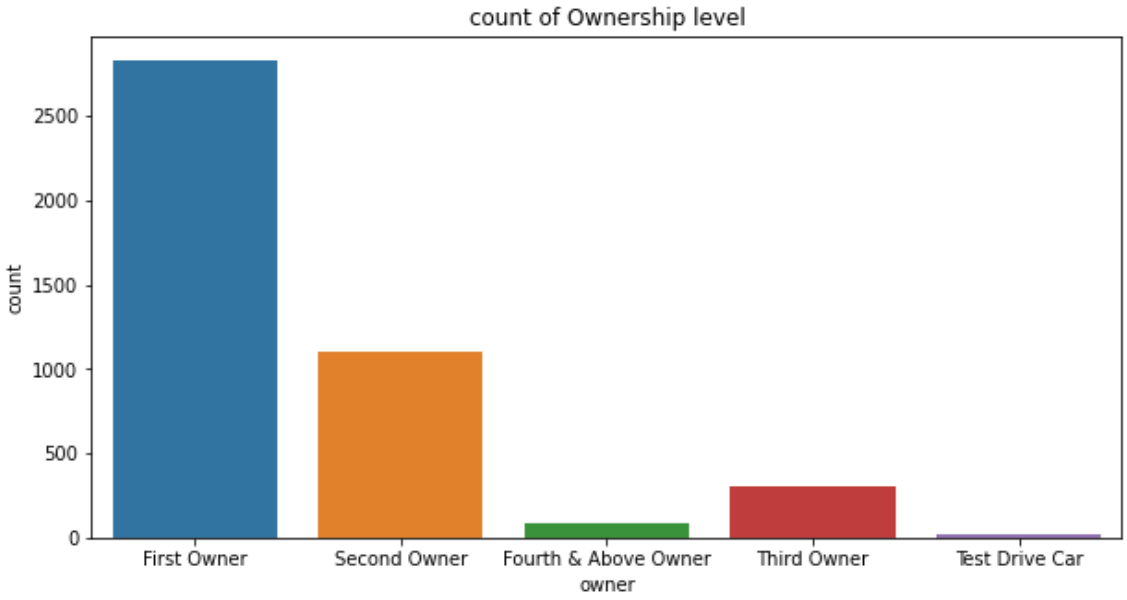
Step 3. Data Visualization

In [97]:

```
#Plot for Count of different Ownership Levels of Car
plt.figure(figsize=(10,5))
sb.countplot(x='owner',data=df)
plt.title("count of Ownership level")
```

Out[97]:

```
Text(0.5, 1.0, 'count of Ownership level')
```

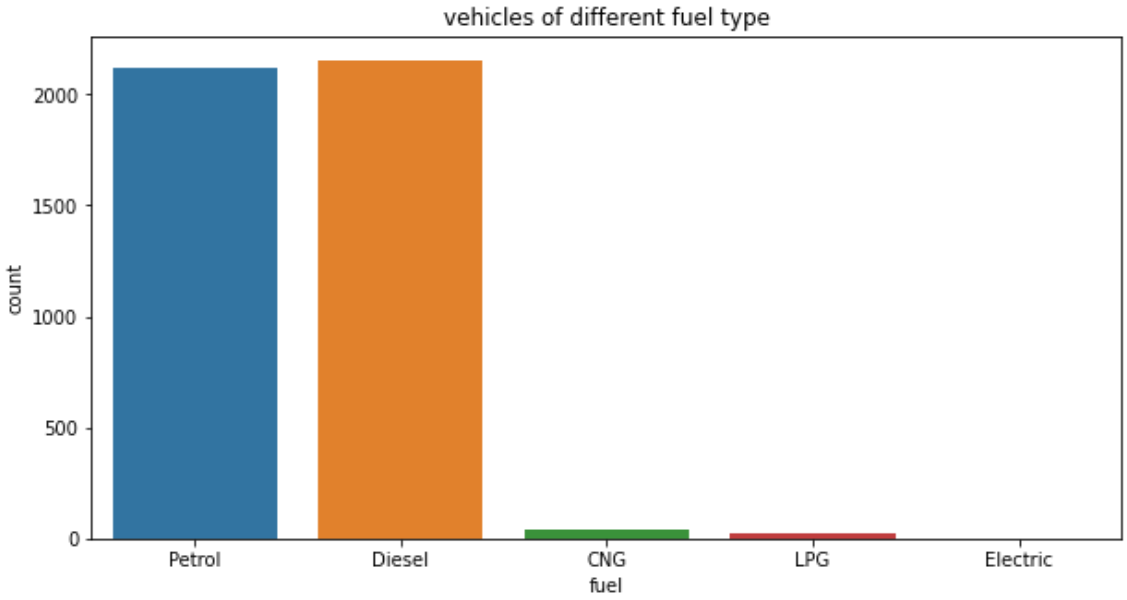


In [98]:

```
#Plot for Vehicles with different fuel types
plt.figure(figsize=(10,5))
sb.countplot(x='fuel',data=df)
plt.title("vehicles of different fuel type")
```

Out[98]:

```
Text(0.5, 1.0, 'vehicles of different fuel type')
```

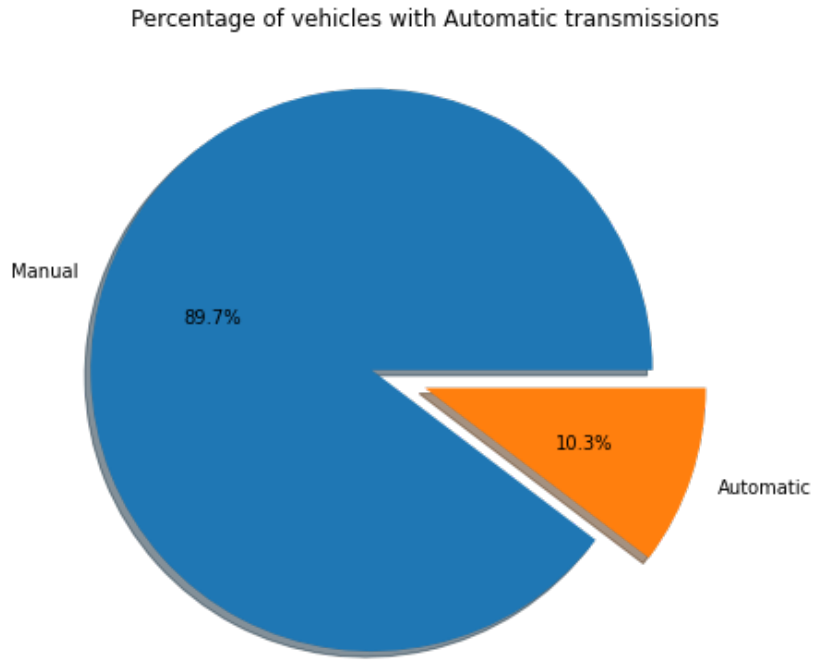


In [99]:

```
# Pie chart for type of vehicle transmission
trans = df.groupby('transmission',sort = False).size()
labels = ["Manual", "Automatic"]
explode = [0.2, 0]
plt.figure(figsize=(7,7))
plt.pie(trans, labels=labels, explode=explode, shadow=True, autopct="%1.1f%%")
plt.title("Percentage of vehicles with Automatic transmissions")
```

Out[99]:

```
Text(0.5, 1.0, 'Percentage of vehicles with Automatic transmissions')
```

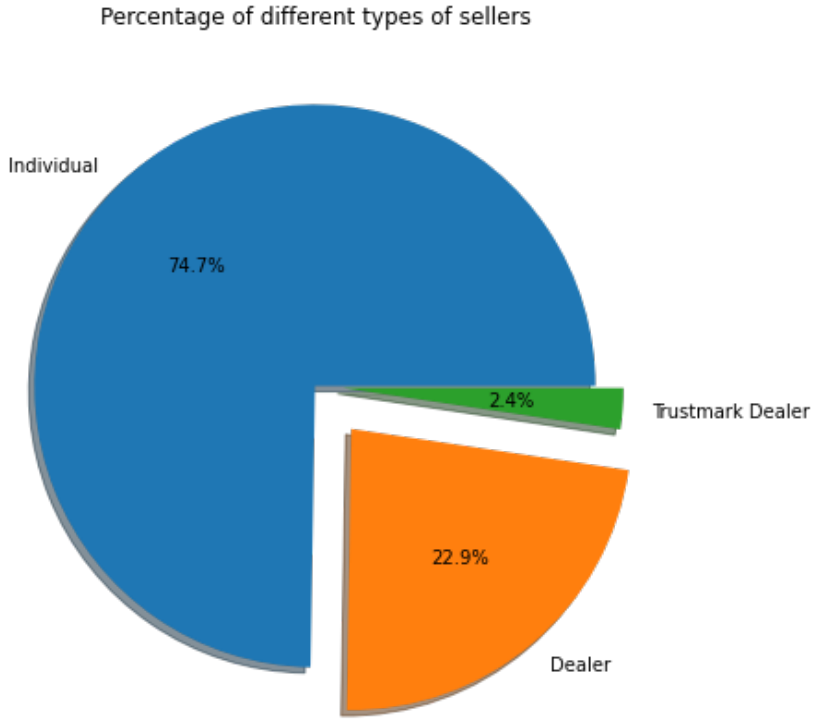


In [100]:

```
#Pie Chart for percentage of different types of seller
seller = df.groupby('seller_type',sort = False).size()
labels = ["Individual", "Dealer", "Trustmark Dealer"]
explode = [0, 0.2, 0.1]
plt.figure(figsize=(7,7))
plt.pie(seller, labels=labels, explode=explode, shadow=True, autopct="%1.1f%%")
plt.title("Percentage of different types of sellers")
```

Out[100]:

```
Text(0.5, 1.0, 'Percentage of different types of sellers')
```



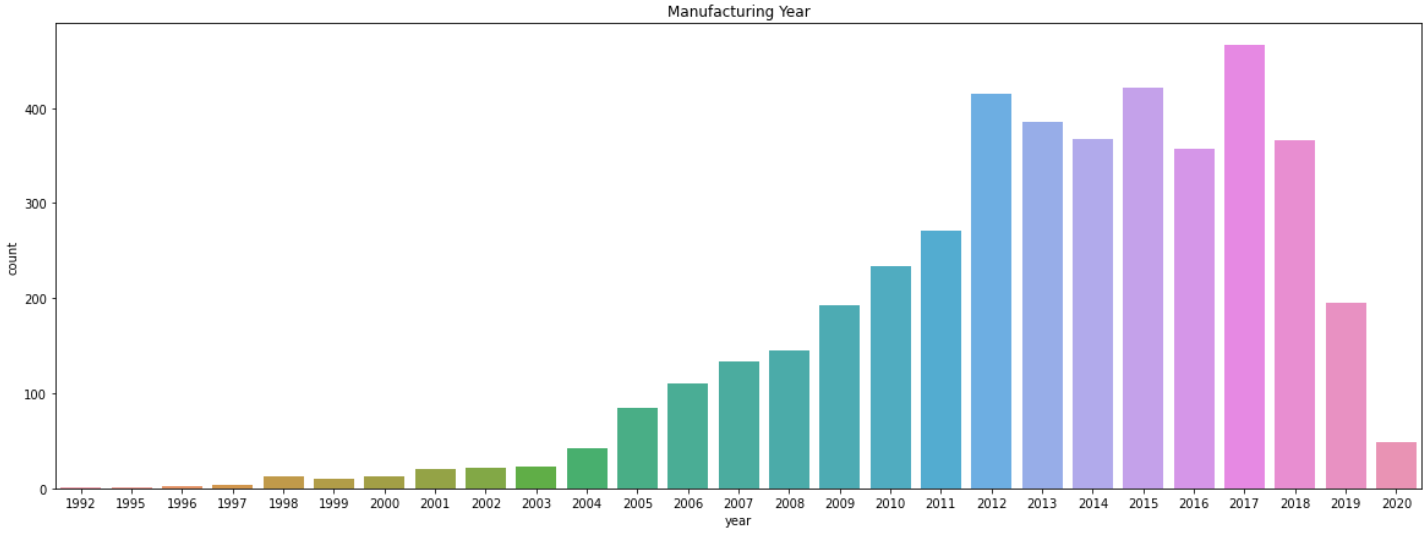
In [101]:

```
#Plot for Count of Vehicle manufactured in different years
plt.figure(figsize=(20,7))
sb.countplot(x='year',data=df)
```

```
plt.title("Manufacturing Year")
```

Out[101]:

```
Text(0.5, 1.0, 'Manufacturing Year')
```

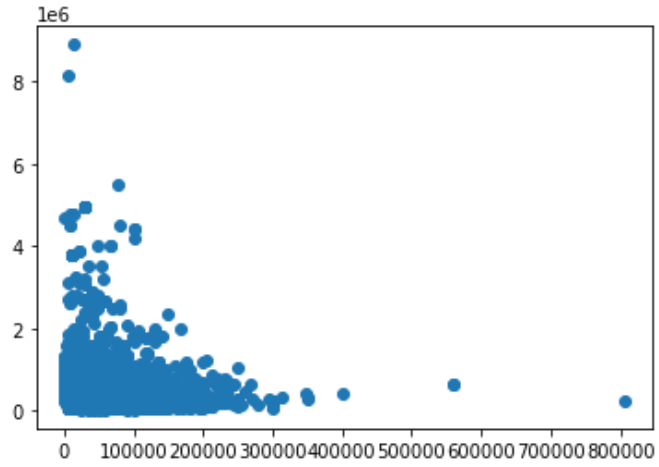


In [102]:

```
#Scatter Plot for the selling price of vehicle for a particular km_driven reading
plt.scatter(df['km_driven'],df['selling_price'])
```

Out[102]:

```
<matplotlib.collections.PathCollection at 0x7fdc989bef10>
```



Step 4->Divide data into input and output

In [103]:

```
# We will take km_driven column as the input for our model
x = df.iloc[:4300,3:4].values #.values converts the column into an array
x
```

Out[103]:

```
array([[ 70000],
       [ 50000],
       [100000],
       ...,
       [  5007],
       [ 30000],
       [ 70000]])
```

In [104]:

```
# We will take selling_price column as the output for our model
```

```
y= df.iloc[:4300,2].values #.values converts the column into an array
y
```

Out[104]:

```
array([ 60000, 135000, 600000, ..., 484999, 164000, 140000])
```

Step 5->Training and Testing data

In [105]:

```
# train_test_split(4 variables -i/p training,i/p testing,o/p train,o/p test)
# x_train contains all the input values of km_driven for training the model
# y_train contains all the values of selling_price(output) for training the model
from sklearn.model_selection import train_test_split
#95% of 4300 rows are used for training purpose and rest 5% for testing purpose
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15, random_state =
0)
```

In [106]:

```
print(x.shape)
print(x_train.shape)
print(x_test.shape)
```

```
(4300, 1)
(3655, 1)
(645, 1)
```

In [107]:

```
print(y.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4300,)
(3655,)
(645,)
```

Step 6. Running a Regressor

In [108]:

```
#Linear Regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

Step 7. Fitting the model

In [109]:

```
model.fit(x,y)
```

Out[109]:

```
LinearRegression()
```

Step 8. Predict the Output

In [110]:

```
y_pred = model.predict(x)
y_pred
```

Out[110]:

```
array([494688.66086409, 541960.96419385, 423780.20586945, ...,
       648307.10137966, 589233.26752362, 494688.66086409])
```

In [111]:

```
y
```

Out[111]:

```
array([ 60000, 135000, 600000, ..., 484999, 164000, 140000])
```

In [112]:

```
# We will find the estimated selling price of the vehicle with km reading=110000 km
model.predict([[110000]])
```

Out[112]:

```
array([400144.05420457])
```

In [113]:

```
m = model.coef_
m
```

Out[113]:

```
array([-2.36361517])
```

In [114]:

```
C = model.intercept_
C
```

Out[114]:

```
660141.7225182625
```

In [115]:

```
m * 110000 + C
```

Out[115]:

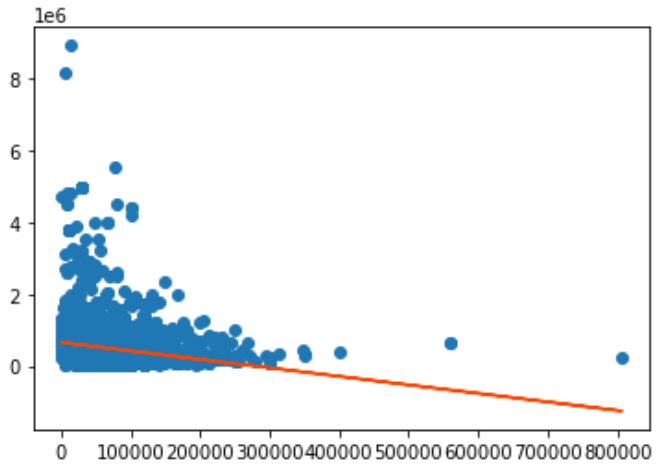
```
array([400144.05420457])
```

In [116]:

```
plt.scatter(x,y)
plt.plot(x,y_pred,c ='orangered')
```

Out[116]:

```
[<matplotlib.lines.Line2D at 0x7fdc988caf50>]
```



Step 9. Accuracy and Testing

Step 6: Accuracy and Testing

In [117]:

```
# defining functions to calculate the accuracy for different hyperparameters.
from sklearn.metrics import explained_variance_score
def get_model(df, n, test_size=0.05, random_state=0):
  x = df.iloc[:n, 3:4] if n != -1 else df.iloc[:, 3:4]
  y = df.iloc[:n, 2] if n != -1 else df.iloc[:, 2]
  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_
state=random_state)
  model = LinearRegression()
  model.fit(x_train, y_train)
  y_pred = model.predict(x_test)
  var_score = explained_variance_score(y_test, y_pred)
  return model, var_score, n, random_state
```

In [118]:

```
models = []
n_rows = [4300, -1] # taking 1. 4123 rows and 2. all rows
random_states = [0, 10, 15, 20, 25, 30, 35, 40] # different random_states
test_size = 0.05 # we will always take 95% of rows for training.
for n in n_rows:
  for random_state in random_states:
    models.append(get_model(df, n, test_size, random_state))
```

In [119]:

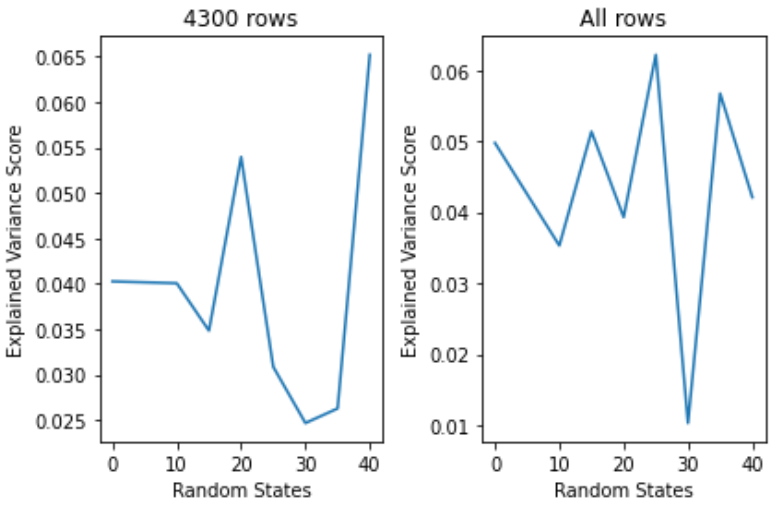
```
for i in range(len(models)):
  print(f'#{i+1} Var-Score: {models[i][1]} N-Rows: {models[i][2]} Random State: {models[i
][3]}')
```

```
#1 Var-Score: 0.040263820512726145 N-Rows: 4300 Random State: 0
#2 Var-Score: 0.04006121211955316 N-Rows: 4300 Random State: 10
#3 Var-Score: 0.03484280021980202 N-Rows: 4300 Random State: 15
#4 Var-Score: 0.053960544234787355 N-Rows: 4300 Random State: 20
#5 Var-Score: 0.03085391183457642 N-Rows: 4300 Random State: 25
#6 Var-Score: 0.024672983901840917 N-Rows: 4300 Random State: 30
#7 Var-Score: 0.026279050353976596 N-Rows: 4300 Random State: 35
#8 Var-Score: 0.06520746540256772 N-Rows: 4300 Random State: 40
#9 Var-Score: 0.0497806261364947 N-Rows: -1 Random State: 0
#10 Var-Score: 0.03531484433205612 N-Rows: -1 Random State: 10
#11 Var-Score: 0.05137492398379351 N-Rows: -1 Random State: 15
#12 Var-Score: 0.03931705567699417 N-Rows: -1 Random State: 20
#13 Var-Score: 0.06220475522744795 N-Rows: -1 Random State: 25
#14 Var-Score: 0.010291260555289394 N-Rows: -1 Random State: 30
#15 Var-Score: 0.056747236588216565 N-Rows: -1 Random State: 35
#16 Var-Score: 0.04212282114175 27 N-Rows: -1 Random State: 40
```

In [125]:

```
# Plotting the Random state v/s Var-Score for 4300 rows and for all Rows
random_states = [[model[3] for model in models if model[2] == 4300], [model[3] for model
in models if model[2] == -1]]
var_scores = [[model[1] for model in models if model[2] == 4300], [model[1] for model in
models if model[2] == -1]]

plt.subplot(1,2,1)

plt.plot(random_states[0], var_scores[0])
plt.xlabel('Random States')
plt.ylabel('Explained Variance Score')

plt.title('4300 rows')

plt.subplot(1,2,2)
plt.plot(random_states[1], var_scores[1])
plt.xlabel('Random States')
plt.ylabel('Explained Variance Score')

plt.title('All rows')
```

```
plt.tight_layout()
plt.show()
```



In [121]:

```
var_scores_all = [model[1] for model in models]
final_model = models[var_scores_all.index(max(var_scores_all))]
```

In [122]:

```
final_model
```

Out[122]:

```
(LinearRegression(), 0.06520746540256772, 4300, 40)
```

Hence our final/best model has the score 0.065 with all rows and random state=15