In [2]:
```
import pandas as pd
car = pd.read_csv("autos.csv", encoding="Latin-1")
```

In [3]:
```
car.columns
```

Out[3]:
```
Index(['dateCrawled', 'name', 'seller', 'offerType', 'price', 'abtest',
       'vehicleType', 'yearOfRegistration', 'gearbox', 'powerPS', 'model',
       'kilometer', 'monthOfRegistration', 'fuelType', 'brand',
       'notRepairedDamage', 'dateCreated', 'nrOfPictures', 'postalCode',
       'lastSeen'],
      dtype='object')
```

In [4]:
```
car.columns =["date_crawled","name","seller","offer_type","price","ab_test","vehi
                  "power_ps","model","kilometer","registration_month","fuel_type'
                  "picture_number","postalcode","last_seen"]
```

In [5]:
```
car
```

Out[5]:

	date_crawled	name	seller	offer_type	price	al
0	24-03-16 11:52	Golf_3_1.6	privat	Angebot	480.0	
1	24-03-16 10:58	A5_Sportback_2.7_Tdi	privat	Angebot	18300.0	
2	14-03-16 12:52	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800.0	
3	17-03-16 16:54	GOLF_4_1_4__3TÜRER	privat	Angebot	1500.0	
4	31-03-16 17:25	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600.0	
...	...	...	...	...	...	
371534	14-03-16 17:48	Suche_t4___vito_ab_6_sitze	privat	Angebot	2200.0	
371535	05-03-16 19:56	Smart_smart_leistungssteigerung_100ps	privat	Angebot	1199.0	
371536	19-03-16 18:57	Volkswagen_Multivan_T4_TDI_7DC_UY2	privat	Angebot	9200.0	
371537	20-03-16 19:41	VW_Golf_Kombi_1_9l_TDI	privat	Angebot	3400.0	
371538	07-03-16 19:39	BMW_M135i_vollausgestattet_NP_52.720____Euro	privat	Angebot	28990.0	c

371539 rows × 20 columns

In [6]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 371539 entries, 0 to 371538
Data columns (total 20 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   date_crawled         371539 non-null  object 
 1   name                 371539 non-null  object 
 2   seller               371538 non-null  object 
 3   offer_type           371538 non-null  object 
 4   price                371538 non-null  float64
 5   ab_test              371538 non-null  object 
 6   vehicle_test         333669 non-null  object 
 7   registration_year    371537 non-null  float64
 8   gear_box             351329 non-null  object 
 9   power_ps             371538 non-null  float64
 10  model                351054 non-null  object 
 11  kilometer            371538 non-null  object 
 12  registration_month   371537 non-null  float64
 13  fuel_type            338151 non-null  object 
 14  brand                371537 non-null  object 
 15  not_repaired_damage  299477 non-null  object 
 16  date_created         371537 non-null  object 
 17  picture_number       371537 non-null  float64
 18  postalcode           371537 non-null  float64
 19  last_seen            371537 non-null  object 
dtypes: float64(6), object(14)
memory usage: 56.7+ MB
```

In [7]:
```
car["seller"].value_counts()
```

Out[7]:
```
privat        371534
gewerblich         3
golf               1
Name: seller, dtype: int64
```

In [8]:
```
car["offer_type"].value_counts()
```

Out[8]:
```
Angebot    371525
Gesuch         12
150000          1
Name: offer_type, dtype: int64
```

In [ ]:
```
car["picture_number"].value_counts()
```

In [9]:
```
not_needed = ["seller", "offer_type", "picture_number"]
car_cleaning = car.drop(not_needed, axis=1)
car_cleaning.shape
```

Out[9]:
```
(371539, 17)
```

In [10]:
```
car_cleaning["price"].value_counts().sort_index(ascending=True).head(10)
```

Out[10]:
```
0.0     10778
1.0      1189
2.0        12
3.0         8
4.0         2
5.0        26
7.0         3
8.0         9
9.0         8
10.0       84
Name: price, dtype: int64
```

In [11]:
```
import math
cost = car_cleaning["price"]
cost_bool = cost == 0
percentage = (cost_bool.sum()) / car_cleaning.shape[0]
percentage = round(percentage, 2)
percentage
```

Out[11]:
```
0.03
```

In [12]:
```
import numpy as np
na = car_cleaning["price"]
na_bool = na == 0
car_cleaning.loc[na_bool, "price"] = np.nan
car_cleaning["price"].value_counts(dropna=False)
```

Out[12]:
```
NaN         10779
500.0        5670
1500.0       5394
1000.0       4649
1200.0       4594
            ...
349000.0        1
8889.0          1
3440.0          1
1997.0          1
10985.0         1
Name: price, Length: 5597, dtype: int64
```

In [18]:
```
nul = car_cleaning["price"].notnull()
cleaned_car = car_cleaning.loc[nul]
cleaned_car
```

Out[18]:

	date_crawled	name	price	ab_test	vehicle_test
0	24-03-16 11:52	Golf_3_1.6	480.0	test	NaN
1	24-03-16 10:58	A5_Sportback_2.7_Tdi	18300.0	test	coupe
2	14-03-16 12:52	Jeep_Grand_Cherokee_"Overland"	9800.0	test	suv
3	17-03-16 16:54	GOLF_4_1_4__3TÜRER	1500.0	test	kleinwagen
4	31-03-16 17:25	Skoda_Fabia_1.4_TDI_PD_Classic	3600.0	test	kleinwagen
...	...	...	...	...	...
371534	14-03-16 17:48	Suche_t4___vito_ab_6_sitze	2200.0	test	NaN
371535	05-03-16 19:56	Smart_smart_leistungssteigerung_100ps	1199.0	test	cabrio
371536	19-03-16 18:57	Volkswagen_Multivan_T4_TDI_7DC_UY2	9200.0	test	bus
371537	20-03-16 19:41	VW_Golf_Kombi_1_9l_TDI	3400.0	test	kombi
371538	07-03-16 19:39	BMW_M135i_vollausgestattet_NP_52.720____Euro	28990.0	control	limousine

360760 rows × 17 columns

In [19]:
```
cleaned_car["price"].value_counts(dropna=False)
```

Out[19]:
```
500.0      5670
1500.0     5394
1000.0     4649
1200.0     4594
2500.0     4438
           ...
31555.0       1
7675.0        1
12696.0       1
16555.0       1
10985.0       1
Name: price, Length: 5596, dtype: int64
```

In [24]:
```
cleaned_car["price"].value_counts().sort_index()
```

Out[24]:
```
1.000000e+00    1189
2.000000e+00      12
3.000000e+00       8
4.000000e+00       2
5.000000e+00      26
                ... 
3.254546e+07       1
7.418530e+07       1
9.900000e+07       1
1.000000e+08      15
2.147484e+09       1
Name: price, Length: 5596, dtype: int64
```

In [30]:
```
cleaned_car["price"].value_counts().sample(10)
```

Out[30]:
```
84993.0     1
21300.0    38
21100.0     3
10666.0     4
2660.0     11
23109.0     1
7660.0      2
47000.0    14
5409.0      1
1856.0      1
Name: price, dtype: int64
```

In [37]:
```
car_boolian = cleaned_car["price"].between(1, 350000)
the_cleaned_car = cleaned_car[car_boolian]
the_cleaned_car["price"].value_counts().sort_index(ascending=False).head(10)
```

Out[37]:
```
350000.0    4
349000.0    1
345000.0    1
323223.0    1
300000.0    1
299000.0    3
295000.0    1
294900.0    1
285000.0    1
284000.0    1
Name: price, dtype: int64
```

```
In [113]: the_cleaned_car["date_crawled"].value_counts().sort_index().tail(20)
```

```
Out[113]: 31-03-16 9:13     1
          31-03-16 9:25     6
          31-03-16 9:26     1
          31-03-16 9:32     2
          31-03-16 9:36    27
          31-03-16 9:37    30
          31-03-16 9:38    12
          31-03-16 9:39     1
          31-03-16 9:45     5
          31-03-16 9:47    10
          31-03-16 9:49     3
          31-03-16 9:50    38
          31-03-16 9:51    36
          31-03-16 9:52    34
          31-03-16 9:53    39
          31-03-16 9:54    41
          31-03-16 9:55    34
          31-03-16 9:56    33
          31-03-16 9:57    27
          ell               1
          Name: date_crawled, dtype: int64
```

In [114]:
```
el = the_cleaned_car["date_crawled"]
el_bool = el == "ell"
the_el = cleaned_car[el_bool]
the_el
```

```
C:\New\envs\snakes\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Bool
ean Series key will be reindexed to match DataFrame index.
  This is separate from the ipykernel package so we can avoid doing imports unt
il

---------------------------------------------------------------------------
IndexingError                             Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5964\2001960544.py in <module>
      1 el = the_cleaned_car["date_crawled"]
      2 el_bool = el == "ell"
----> 3 the_el = cleaned_car[el_bool]
      4 the_el

C:\New\envs\snakes\lib\site-packages\pandas\core\frame.py in __getitem__(self,
 key)
   3447         # Do we have a (boolean) 1d indexer?
   3448         if com.is_bool_indexer(key):
-> 3449             return self._getitem_bool_array(key)
   3450 
   3451         # We are left with two options: a single key, and a collection
 of keys,

C:\New\envs\snakes\lib\site-packages\pandas\core\frame.py in _getitem_bool_arra
y(self, key)
   3500         # check_bool_indexer will throw exception if Series key cannot
   3501         # be reindexed to match DataFrame rows
-> 3502         key = check_bool_indexer(self.index, key)
   3503         indexer = key.nonzero()[0]
   3504         return self._take_with_is_copy(indexer, axis=0)

C:\New\envs\snakes\lib\site-packages\pandas\core\indexing.py in check_bool_inde
xer(index, key)
   2387         if mask.any():
   2388             raise IndexingError(
-> 2389                 "Unalignable boolean Series provided as "
   2390                 "indexer (index of the boolean Series and of "
   2391                 "the indexed object do not match)."

IndexingError: Unalignable boolean Series provided as indexer (index of the boo
lean Series and of the indexed object do not match).
```

In [59]:
```
el = cleaned_car["date_crawled"]
el_bool = el == "ell"
cleaned_car.loc[el_bool, "date_crawled"] = np.nan
```

In [65]:
```
cleaned_car["date_crawled"].value_counts(dropna=False)
```

Out[65]:
```
05-03-16 14:25    66
05-03-16 14:26    60
05-03-16 17:49    58
05-03-16 15:48    57
20-03-16 11:50    55
                  ..
24-03-16 9:44      1
03-04-16 9:48      1
10-03-16 4:32      1
17-03-16 9:31      1
06-03-16 21:11     1
Name: date_crawled, Length: 15550, dtype: int64
```

In [115]:
```
the_cleaned_car["registration_year"].value_counts(dropna=False).sort_index().tail
```

Out[115]:
```
3000.0     6
3200.0     1
3700.0     1
3800.0     1
4000.0     3
4100.0     1
4500.0     2
4800.0     1
5000.0    17
5300.0     1
5555.0     2
5600.0     1
5900.0     1
5911.0     2
6000.0     6
6200.0     1
6500.0     1
7000.0     4
7100.0     1
7500.0     1
7777.0     1
7800.0     1
8000.0     2
8200.0     1
8500.0     1
8888.0     2
9000.0     4
9450.0     1
9999.0    18
NaN        1
Name: registration_year, dtype: int64
```

In [116]:
```
reg = the_cleaned_car["registration_year"].between(1900, 2016)
my_car = the_cleaned_car[reg]
my_car["registration_year"].value_counts().sort_index().tail(10)
```

Out[116]:
```
2007.0    17506
2008.0    16035
2009.0    15498
2010.0    12267
2011.0    11996
2012.0     9359
2013.0     6122
2014.0     4770
2015.0     2919
2016.0     9216
Name: registration_year, dtype: int64
```

In [117]:
```
my_car
```

Out[117]:

	date_crawled	name	price	ab_test	vehicle_test
0	24-03-16 11:52	Golf_3_1.6	480.0	test	NaN
1	24-03-16 10:58	A5_Sportback_2.7_Tdi	18300.0	test	coupe
2	14-03-16 12:52	Jeep_Grand_Cherokee_"Overland"	9800.0	test	suv
3	17-03-16 16:54	GOLF_4_1_4__3TÜRER	1500.0	test	kleinwagen
4	31-03-16 17:25	Skoda_Fabia_1.4_TDI_PD_Classic	3600.0	test	kleinwagen
...	...	...	...	...	...
371534	14-03-16 17:48	Suche_t4___vito_ab_6_sitze	2200.0	test	NaN
371535	05-03-16 19:56	Smart_smart_leistungssteigerung_100ps	1199.0	test	cabrio
371536	19-03-16 18:57	Volkswagen_Multivan_T4_TDI_7DC_UY2	9200.0	test	bus
371537	20-03-16 19:41	VW_Golf_Kombi_1_9l_TDI	3400.0	test	kombi
371538	07-03-16 19:39	BMW_M135i_vollausgestattet_NP_52.720____Euro	28990.0	control	limousine

346669 rows × 17 columns

In [118]:
```
my_car["kilometer"].value_counts().sort_values()
```

Out[118]:
```
10000        159
20000        442
30000        489
5000         503
40000        522
50000        622
60000        742
70000        793
80000        954
90000       1019
100000      1311
10000       1626
125000      3214
20000       4928
5000        5106
30000       5312
40000       5692
50000       6738
60000       7692
70000       8647
80000       9658
90000      10969
100000     13723
150000     19765
125000     32795
150000    203248
Name: kilometer, dtype: int64
```

```
In [119]: my_car["brand"].value_counts(normalize=True).sort_values(ascending=False)
```

```
Out[119]: volkswagen        0.211695
          bmw               0.109868
          opel              0.106407
          mercedes_benz     0.096850
          audi              0.089541
          ford              0.068919
          renault           0.047521
          peugeot           0.030153
          fiat              0.025690
          seat              0.018660
          skoda             0.015686
          mazda             0.015384
          smart             0.014331
          citroen           0.013950
          nissan            0.013598
          toyota            0.012935
          hyundai           0.009972
          sonstige_autos    0.009493
          mini              0.009384
          volvo             0.009147
          mitsubishi        0.008236
          honda             0.007532
          kia               0.006914
          suzuki            0.006363
          alfa_romeo        0.006309
          porsche           0.006211
          chevrolet         0.005022
          chrysler          0.003862
          dacia             0.002495
          jeep              0.002192
          land_rover        0.002169
          daihatsu          0.002161
          subaru            0.002117
          jaguar            0.001734
          saab              0.001465
          daewoo            0.001457
          trabant           0.001408
          lancia            0.001301
          rover             0.001272
          lada              0.000597
          Name: brand, dtype: float64
```

In [120]:
```
my_brand = my_car["brand"].value_counts(normalize=True).sort_values(ascending=Fal
my_bool = my_brand < 100
the_row = my_brand[my_bool]
index = the_row.index
index
```

Out[120]:
```
Index(['volkswagen', 'bmw', 'opel', 'mercedes_benz', 'audi', 'ford', 'renault',
       'peugeot', 'fiat', 'seat', 'skoda', 'mazda', 'smart', 'citroen',
       'nissan', 'toyota', 'hyundai', 'sonstige_autos', 'mini', 'volvo',
       'mitsubishi', 'honda', 'kia', 'suzuki', 'alfa_romeo', 'porsche',
       'chevrolet', 'chrysler', 'dacia', 'jeep', 'land_rover', 'daihatsu',
       'subaru', 'jaguar', 'saab', 'daewoo', 'trabant', 'lancia', 'rover',
       'lada'],
      dtype='object')
```

In [122]:
```
import math
car_mean = {}
for checks in index:
    brands = my_car["brand"]
    boo = brands == checks
    boo_1 = my_car.loc[boo, "price"]
    means = boo_1.mean()
    means = int(means)
    car_mean[checks] = means
car_mean
```

Out[122]:
```
{'volkswagen': 5400,
 'bmw': 8449,
 'opel': 2971,
 'mercedes_benz': 8558,
 'audi': 9086,
 'ford': 3696,
 'renault': 2437,
 'peugeot': 3267,
 'fiat': 2892,
 'seat': 4541,
 'skoda': 6530,
 'mazda': 4076,
 'smart': 3632,
 'citroen': 3734,
 'nissan': 4708,
 'toyota': 5340,
 'hyundai': 5567,
 'sonstige_autos': 14288,
 'mini': 10080,
 'volvo': 5238,
 'mitsubishi': 3407,
 'honda': 4005,
 'kia': 5855,
 'suzuki': 4044,
 'alfa_romeo': 4291,
 'porsche': 42258,
 'chevrolet': 7117,
 'chrysler': 4121,
 'dacia': 5922,
 'jeep': 11213,
 'land_rover': 17060,
 'daihatsu': 1775,
 'subaru': 4386,
 'jaguar': 13765,
 'saab': 3955,
 'daewoo': 1027,
 'trabant': 1900,
 'lancia': 3289,
 'rover': 1600,
 'lada': 3191}
```

```
In [123]: my_car["price"].value_counts().sort_values()
```

```
Out[123]: 10985.0       1
          30790.0       1
          18555.0       1
          3425.0        1
          5277.0        1
                     ...
          2500.0     4244
          1200.0     4332
          1000.0     4366
          1500.0     5093
          500.0      5468
          Name: price, Length: 5500, dtype: int64
```

```
In [ ]:
```