Used_car cleaning

February 3, 2026

1 Calling Library

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

2 Load data

[2]:
```
df = pd.read_csv('D:/projects/USED_CAR/vehicles.csv/vehicles.csv')
```

3 Display information about the data

[4]:
```
# Display the first five rows of data
df.head()
```

[4]:

	id	url	\
0	7222695916	https://prescott.craigslist.org/cto/d/prescott…	
1	7218891961	https://fayar.craigslist.org/ctd/d/bentonville…	
2	7221797935	https://keys.craigslist.org/cto/d/summerland-k…	
3	7222270760	https://worcester.craigslist.org/cto/d/west-br…	
4	7210384030	https://greensboro.craigslist.org/cto/d/trinit…	

	region	region_url	price	year	\
0	prescott	https://prescott.craigslist.org	6000	NaN	
1	fayetteville	https://fayar.craigslist.org	11900	NaN	
2	florida keys	https://keys.craigslist.org	21000	NaN	
3	worcester / central MA	https://worcester.craigslist.org	1500	NaN	
4	greensboro	https://greensboro.craigslist.org	4900	NaN	

	manufacturer	model	condition	cylinders	…	size	type	paint_color	\
0	NaN	NaN	NaN	NaN	…	NaN	NaN	NaN	
1	NaN	NaN	NaN	NaN	…	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	…	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	…	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	…	NaN	NaN	NaN	

```
  image_url description county state lat long posting_date
0       NaN         NaN    NaN    az NaN  NaN          NaN
1       NaN         NaN    NaN    ar NaN  NaN          NaN
2       NaN         NaN    NaN    fl NaN  NaN          NaN
3       NaN         NaN    NaN    ma NaN  NaN          NaN
4       NaN         NaN    NaN    nc NaN  NaN          NaN

[5 rows x 26 columns]
```

3.1 Identifying null values

[5]:
```
df.isna().sum()
```

[5]:
```
id                  0
url                 0
region              0
region_url          0
price               0
year             1205
manufacturer    17646
model            5277
condition      174104
cylinders      177678
fuel             3013
odometer         4400
title_status     8242
transmission     2556
VIN            161042
drive          130567
size           306361
type            92858
paint_color    130203
image_url          68
description        70
county         426880
state               0
lat              6549
long             6549
posting_date       68
dtype: int64
```

3.2 General description

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

[6]:

	id	price	year	odometer	county	\
count	4.268800e+05	4.268800e+05	425675.000000	4.224800e+05	0.0	
mean	7.311487e+09	7.519903e+04	2011.235191	9.804333e+04	NaN	
std	4.473170e+06	1.218228e+07	9.452120	2.138815e+05	NaN	
min	7.207408e+09	0.000000e+00	1900.000000	0.000000e+00	NaN	
25%	7.308143e+09	5.900000e+03	2008.000000	3.770400e+04	NaN	
50%	7.312621e+09	1.395000e+04	2013.000000	8.554800e+04	NaN	
75%	7.315254e+09	2.648575e+04	2017.000000	1.335425e+05	NaN	
max	7.317101e+09	3.736929e+09	2022.000000	1.000000e+07	NaN	

	lat	long
count	420331.000000	420331.000000
mean	38.493940	-94.748599
std	5.841533	18.365462
min	-84.122245	-159.827728
25%	34.601900	-111.939847
50%	39.150100	-88.432600
75%	42.398900	-80.832039
max	82.390818	173.885502

[7]:
```
# Display some important information about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 426880 entries, 0 to 426879
Data columns (total 26 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            426880 non-null  int64
 1   url           426880 non-null  object
 2   region        426880 non-null  object
 3   region_url    426880 non-null  object
 4   price         426880 non-null  int64
 5   year          425675 non-null  float64
 6   manufacturer  409234 non-null  object
 7   model         421603 non-null  object
 8   condition     252776 non-null  object
 9   cylinders     249202 non-null  object
 10  fuel          423867 non-null  object
 11  odometer      422480 non-null  float64
 12  title_status  418638 non-null  object
 13  transmission  424324 non-null  object
 14  VIN           265838 non-null  object
 15  drive         296313 non-null  object
 16  size          120519 non-null  object
 17  type          334022 non-null  object
 18  paint_color   296677 non-null  object
 19  image_url     426812 non-null  object
```

```
 20  description   426810 non-null  object
 21  county        0 non-null       float64
 22  state         426880 non-null  object
 23  lat           420331 non-null  float64
 24  long          420331 non-null  float64
 25  posting_date  426812 non-null  object
dtypes: float64(5), int64(2), object(19)
memory usage: 84.7+ MB
```

[8]:
```
# DESPLAY COLUMN NAMES
df.columns
```

[8]:
```
Index(['id', 'url', 'region', 'region_url', 'price', 'year', 'manufacturer',
       'model', 'condition', 'cylinders', 'fuel', 'odometer', 'title_status',
       'transmission', 'VIN', 'drive', 'size', 'type', 'paint_color',
       'image_url', 'description', 'county', 'state', 'lat', 'long',
       'posting_date'],
      dtype='object')
```

[9]:
```
# DESPLAY DATA TYPE FOR EVERY COLUMN
df.dtypes
```

[9]:
```
id               int64
url             object
region          object
region_url      object
price            int64
year           float64
manufacturer    object
model           object
condition       object
cylinders       object
fuel            object
odometer       float64
title_status    object
transmission    object
VIN             object
drive           object
size            object
type            object
paint_color     object
image_url       object
description     object
county         float64
state           object
lat            float64
long           float64
posting_date    object
```

```
dtype: object
```

4 Data cleaning

[10]:
```
# I deleted the columns that I didn't need in the study and that contained many␣
↪empty values.
df=df.drop(["id","url","image_url","region_url","lat","long","county"],axis = 1)
```

[12]:
```
# Text data was stored in a variable
df_object = df.select_dtypes(include = 'object').head(3)
# Non-text data was stored in a variable
df_num = df.select_dtypes(exclude = 'object').head(3)
```

[13]:
```
df.select_dtypes(exclude = 'object').isna().sum()
```

[13]:
```
price          0
year        1205
odometer    4400
dtype: int64
```

[14]:
```
df.select_dtypes(include = 'object').isna().sum()
```

[14]:
```
region              0
manufacturer    17646
model            5277
condition      174104
cylinders      177678
fuel             3013
title_status     8242
transmission     2556
VIN            161042
drive          130567
size           306361
type            92858
paint_color    130203
description        70
state               0
posting_date       68
dtype: int64
```

[15]:
```
# I deleted all rows containing empty values in these columns.

df.dropna(subset = ["model"],inplace = True)
df.dropna(subset = ["fuel"],inplace = True)
df.dropna(subset = ["transmission"],inplace = True)
df.dropna(subset = ["title_status"],inplace = True)
df.dropna(subset = ["posting_date"],inplace = True)
```

[20]:
```
# I delete missing values in a column when the values
# within that column are less than 7000,
# and deleting this data doesn't cause any harm in large datasets.
```

[17]:
```
# Display missing values in text data
df.select_dtypes(include = 'object').isna().sum()
```

[17]:
```
region              0
manufacturer    16764
model               0
condition      164850
cylinders      171086
fuel                0
title_status        0
transmission        0
VIN            157280
drive          122837
size           291571
type            89085
paint_color    119557
description         2
state               0
posting_date        0
dtype: int64
```

[18]:
```
# Display missing values, non-text data
df.select_dtypes(exclude = 'object').isna().sum()
```

[18]:
```
price          0
year         737
odometer    3508
dtype: int64
```

[19]:
```
# I deleted all missing values in the non-text data.
df.dropna(subset = ["year"],inplace = True)
df.dropna(subset = ["odometer"],inplace = True)
```

[21]:
```
# Cleaned of empty values
df.select_dtypes(exclude = 'object').isna().sum()
```

[21]:
```
price       0
year        0
odometer    0
dtype: int64
```

[22]:
```
# Due to the deletion of rows, the data index has changed. We will fix this␣
  ↪later.
df["price"].index
```

[22]:
```
Index([    27,     28,     29,     30,     31,     32,     33,     34,     35,
           36,
       …
        426870, 426871, 426872, 426873, 426874, 426875, 426876, 426877, 426878,
        426879],
      dtype='int64', length=405594)
```

[20]:
```
(405594 * (7.5/100))
```

[20]:
```
30419.55
```

[23]:
```
# I will delete the first 7.5% of the data from both sides because there are␣
 ↪outliers.
```

[24]:
```
df[df["price"] ==0].head(5)
```

[24]:

	region	price	year	manufacturer	model	condition	\
46	auburn	0	2011.0	jeep	compass	excellent	
126	auburn	0	2018.0	chevrolet	express cargo van	like new	
127	auburn	0	2019.0	chevrolet	express cargo van	like new	
128	auburn	0	2018.0	chevrolet	express cargo van	like new	
191	birmingham	0	2015.0	nissan	sentra	excellent	

	cylinders	fuel	odometer	title_status	transmission	VIN	\
46	NaN	gas	99615.0	clean	automatic	NaN	
126	6 cylinders	gas	68472.0	clean	automatic	1GCWGAFP8J1309579	
127	6 cylinders	gas	69125.0	clean	automatic	1GCWGAFP4K1214373	
128	6 cylinders	gas	66555.0	clean	automatic	1GCWGAFPXJ1337903	
191	4 cylinders	gas	99505.0	clean	automatic	3N1AB7AP8FY348505	

	drive	size	type	paint_color	\
46	NaN	full-size	SUV	NaN	
126	rwd	full-size	van	white	
127	rwd	full-size	van	white	
128	rwd	full-size	van	white	
191	fwd	NaN	sedan	silver	

	description	state	\
46	Call or text now 800-213-0336 Open 9:00-6:00pm…	al	
126	2018 Chevrolet Express Cargo Van RWD 2500 135"…	al	
127	2019 Chevrolet Express Cargo Van RWD 2500 135"…	al	
128	2018 Chevrolet Express Cargo Van RWD 2500 135"…	al	
191	2015 Nissan Sentra by Benton Nissan of Oxford…	al	

	posting_date
46	2021-04-30T16:35:11-0500
126	2021-04-12T11:20:35-0500

```
127   2021-04-12T11:20:00-0500
128   2021-04-12T11:19:58-0500
191   2021-05-04T11:00:42-0500
```

[25]:
```
top_30419 = df_num['price'].nlargest(30419)
bottom_100 = df_num['price'].nsmallest(30419)
```

[26]:
```
x=df[(df["price"] <= 3736928711) & (df["price"] >= 1111111)].index
y=df[df["price"] ==0].index
```

[27]:
```
df=df.drop(x,axis = 0)
```

[28]:
```
df=df.drop(y,axis = 0)
```

[29]:
```
df.describe()
```

[29]:

	price	year	odometer
count	375717.000000	375717.000000	3.757170e+05
mean	18935.740640	2011.096118	9.849603e+04
std	16144.694868	9.385032	1.976752e+05
min	1.000000	1900.000000	0.000000e+00
25%	7200.000000	2008.000000	3.812300e+04
50%	15499.000000	2013.000000	8.703700e+04
75%	27882.000000	2017.000000	1.356360e+05
max	1000000.000000	2022.000000	1.000000e+07

[31]:
```
# We observe that the mean and median are approaching each other, which means␣
  ↪that they are approaching a normal distribution.
print(df['price'].mean())
print(df['price'].median())
```

```
18935.740639896518
15499.0
```

[32]:
```
df.isna().sum()
```

[32]:
```
region              0
price               0
year                0
manufacturer    14714
model               0
condition      142243
cylinders      153971
fuel                0
odometer            0
title_status        0
transmission        0
VIN            147590
```

```
drive           112905
size            267055
type             82507
paint_color     106661
description          2
state                0
posting_date         0
dtype: int64
```

[33]:
```
# I will clean each column individually and fill in the empty values with the␣
↪most suitable values, God willing.
```

5 manufacturer

[34]:
```
# Non-duplicate values
df["manufacturer"].unique()
```

[34]:
```
array(['gmc', 'chevrolet', 'toyota', 'ford', 'jeep', 'nissan', 'ram',
       'mazda', 'cadillac', 'honda', 'dodge', 'lexus', 'jaguar', 'buick',
       'chrysler', 'volvo', 'audi', 'infiniti', 'lincoln', 'alfa-romeo',
       'subaru', nan, 'acura', 'hyundai', 'mercedes-benz', 'bmw',
       'mitsubishi', 'volkswagen', 'porsche', 'kia', 'ferrari', 'mini',
       'pontiac', 'fiat', 'rover', 'tesla', 'saturn', 'mercury',
       'harley-davidson', 'datsun', 'aston-martin', 'land rover'],
      dtype=object)
```

[35]:
```
df["manufacturer"].isna().sum()
```

[35]:
```
np.int64(14714)
```

[36]:
```
# Change the missing values to "unknown" (indicating an unknown value).
df["manufacturer"]=df["manufacturer"].fillna("Unknown")
```

[37]:
```
df["manufacturer"].isna().sum()
```

[37]:
```
np.int64(0)
```

[38]:
```
df["manufacturer"].unique()
```

[38]:
```
array(['gmc', 'chevrolet', 'toyota', 'ford', 'jeep', 'nissan', 'ram',
       'mazda', 'cadillac', 'honda', 'dodge', 'lexus', 'jaguar', 'buick',
       'chrysler', 'volvo', 'audi', 'infiniti', 'lincoln', 'alfa-romeo',
       'subaru', 'Unknown', 'acura', 'hyundai', 'mercedes-benz', 'bmw',
       'mitsubishi', 'volkswagen', 'porsche', 'kia', 'ferrari', 'mini',
       'pontiac', 'fiat', 'rover', 'tesla', 'saturn', 'mercury',
       'harley-davidson', 'datsun', 'aston-martin', 'land rover'],
      dtype=object)
```

6 condition

[40]:
```
# Change the missing values to "UN" (indicating an unknown value).
df["condition"]=df["condition"].fillna("UN")
df["condition"].isna().sum()
```

[40]:
```
np.int64(0)
```

[41]:
```
df["condition"].unique()
```

[41]:
```
array(['good', 'excellent', 'fair', 'like new', 'UN', 'new', 'salvage'],
      dtype=object)
```

[42]:
```
# I will create a loop with conditions inside to fill in the values with what␣
↪suits them. I know there are newer methods, but I will use this method␣
↪because everyone knows it.
```

[43]:
```
prices = list(df["price"])
condition = list(df["condition"])
```

[44]:
```
len(prices) == len(condition)
```

[44]:
```
True
```

[45]:
```
df["price"].describe()
```

[45]:
```
count     375717.000000
mean       18935.740640
std        16144.694868
min            1.000000
25%         7200.000000
50%        15499.000000
75%        27882.000000
max      1000000.000000
Name: price, dtype: float64
```

[46]:
```
df["condition"].info()
```

```
<class 'pandas.core.series.Series'>
Index: 375717 entries, 27 to 426879
Series name: condition
Non-Null Count   Dtype
--------------   -----
375717 non-null  object
dtypes: object(1)
memory usage: 5.7+ MB
```

[47]:
```
condition[230239]
```

[47]:
```
'good'
```

[48]:
```
o=0
for i in condition:

    if i == 'UN':
        if prices[o] <= 1000000.0:
            condition[o] = condition[o].replace('UN',"new")
        elif  prices[o] <=27882.00:
            condition[o] = condition[o].replace('UN',"like new")
        elif  prices[o] <= 18935.0:
            condition[o] = condition[o].replace('UN',"excellent")
        elif  prices[o] <=13995.0:
            condition[o] = condition[o].replace('UN',"good")
        elif  prices[o] <=5995.0:
            condition[o] = condition[o].replace('UN',"fair")
        else :
            condition[o] = condition[o].replace("UN","salvage")
    o+=1
```

[49]:
```
test_2=pd.Series(condition)
```

[50]:
```
test_2.unique()
```

[50]:
```
array(['good', 'excellent', 'fair', 'like new', 'new', 'salvage'],
      dtype=object)
```

[51]:
```
df=df.drop(["condition"],axis=1)
```

[52]:
```
df.isna().sum()
```

[52]:
```
region              0
price               0
year                0
manufacturer        0
model               0
cylinders       153971
fuel                0
odometer            0
title_status        0
transmission        0
VIN            147590
drive          112905
size           267055
type            82507
paint_color    106661
description         2
state               0
```

```
posting_date         0
dtype: int64
```

[53]:
```
index =list(df.index)
```

[54]:
```
test_3=pd.DataFrame(condition,index=index,columns=["condition"])
```

[55]:
```
df=pd.concat([df,test_3],axis=1)
```

[56]:
```
df.isna().sum()
```

[56]:
```
region              0
price               0
year                0
manufacturer        0
model               0
cylinders      153971
fuel                0
odometer            0
title_status        0
transmission        0
VIN            147590
drive          112905
size           267055
type            82507
paint_color    106661
description         2
state               0
posting_date        0
condition           0
dtype: int64
```

[57]:
```
# I will change the missing values in the appropriate column.
```

7 cylinders

[58]:
```
df["cylinders"]=df["cylinders"].fillna("UN")
df["cylinders"].isna().sum()
```

[58]:
```
np.int64(0)
```

[59]:
```
df["cylinders"].unique()
```

[59]:
```
array(['8 cylinders', '6 cylinders', 'UN', '4 cylinders', '5 cylinders',
       'other', '3 cylinders', '10 cylinders', '12 cylinders'],
      dtype=object)
```

[60]:
```
cylinders = list(df["cylinders"])
```

[61]:
```
len(prices) == len(cylinders)
```

[61]:
```
True
```

[62]:
```
o=0
for i in cylinders:

    if i == 'UN':
        if prices[o] <= 1000000.0:
            cylinders[o] = cylinders[o].replace('UN',"12 cylinders")
        elif  prices[o] <=27882.00:
            cylinders[o] = cylinders[o].replace('UN',"10 cylinders")
        elif  prices[o] <= 18935.0:
            cylinders[o] = cylinders[o].replace('UN',"8 cylinders")
        elif  prices[o] <=13995.0:
            cylinders[o] = cylinders[o].replace('UN',"6 cylinders")
        elif  prices[o] <=5995.0:
            cylinders[o] = cylinders[o].replace('UN',"4 cylinders")
        else :
            cylinders[o] = cylinders[o].replace("UN","3 cylinders")
    o+=1
```

[63]:
```
test_4=pd.Series(cylinders)
```

[64]:
```
test_4.unique()
```

[64]:
```
array(['8 cylinders', '6 cylinders', '12 cylinders', '4 cylinders',
       '5 cylinders', 'other', '3 cylinders', '10 cylinders'],
      dtype=object)
```

[65]:
```
df=df.drop(["cylinders"],axis=1)
```

[66]:
```
index =list(df.index)
```

[67]:
```
test_5=pd.DataFrame(cylinders,index=index,columns=["cylinders"])
```

[68]:
```
df=pd.concat([df,test_5],axis=1)
```

[69]:
```
df.isna().sum()
```

[69]:
```
region              0
price               0
year                0
manufacturer        0
model               0
fuel                0
```

```
odometer             0
title_status         0
transmission         0
VIN             147590
drive           112905
size            267055
type             82507
paint_color     106661
description          2
state                0
posting_date         0
condition            0
cylinders            0
dtype: int64
```

[70]:
```
df.sample(5)
```

[70]:

	region	price	year	manufacturer	model	fuel	odometer	title_status	transmission	VIN	drive	size	type	paint_color	description	state	posting_date	condition	cylinders
157265	des moines	22995	2014.0	chevrolet	silverado 1500	gas	94958.0	clean	automatic	1GCRCREC6EZ115114	rwd	NaN	truck	NaN	2014 CHEVROLET SILVERADO 1500 LT Truck \t\t…	ia	2021-04-26T15:36:22-0500	excellent	8 cylinders
268760	elmira-corning	44995	2014.0	ford	super duty f-550 drw	gas	18019.0	clean	automatic	1FDUF5HY7EEB27454	rwd	NaN	pickup	blue	Why Buy From Twin Work Vans?Twin Work Vans is …	ny			
29821	inland empire	4950	2009.0	nissan	versa s	gas	134000.0	clean	automatic	NaN	fwd	compact	hatchback	grey	4cyl 1.8L automatic with ABS, power steering p…	ca			
136715	spokane / coeur d'alene	5900	2006.0	mercedes-benz	benz ml350	gas	121000.0	rebuilt	automatic	NaN	NaN	NaN	NaN	NaN	06 Mercedes ML 350 AWD 3.5L V6 , automatic , …	id			
312905	bend	66747	2016.0	ford	super duty f-350 srw	diesel	24062.0	clean	automatic	1FT8W3BT2GEC23776	4wd	NaN	truck	NaN	Hillyer's Mid-City STOCK #: T1335A …	or			

```
268760  2021-04-27T18:50:36-0400       good  12 cylinders
29821   2021-05-04T12:11:18-0700  excellent   4 cylinders
136715  2021-05-01T19:02:34-0700       fair  12 cylinders
312905  2021-04-21T08:11:57-0700        new   8 cylinders
```

8 drive

[71]:
```
df["drive"].unique()
```

[71]:
```
array([nan, 'rwd', '4wd', 'fwd'], dtype=object)
```

[72]:
```
df["drive"]=df["drive"].fillna("UN")
df["drive"].isna().sum()
```

[72]:
```
np.int64(0)
```

[73]:
```
df["drive"].unique()
```

[73]:
```
array(['UN', 'rwd', '4wd', 'fwd'], dtype=object)
```

[74]:
```
drive = list(df["drive"])
```

[75]:
```
len(prices) == len(drive)
```

[75]:
```
True
```

[76]:
```
o=0
for i in drive:

    if i == 'UN':
        if prices[o] <= 1000000.0:
            drive[o] = drive[o].replace('UN',"4wd")
        elif  prices[o] <= 18935.0:
            drive[o] = drive[o].replace('UN',"rwd")
        else :
            drive[o] = drive[o].replace("UN","fwd")
    o+=1
```

[ ]:
```

```

[77]:
```
test_6=pd.Series(drive)
```

[78]:
```
test_6.unique()
```

[78]:
```
array(['4wd', 'rwd', 'fwd'], dtype=object)
```

[79]:
```
df=df.drop(["drive"],axis=1)
```

[80]:
```
index =list(df.index)
```

[81]:
```
test_7=pd.DataFrame(drive,index=index,columns=["drive"])
```

[82]:
```
df=pd.concat([df,test_7],axis=1)
```

[83]:
```
df.isna().sum()
```

[83]:
```
region              0
price               0
year                0
manufacturer        0
model               0
fuel                0
odometer            0
title_status        0
transmission        0
VIN            147590
size           267055
type            82507
paint_color    106661
description         2
state               0
posting_date        0
condition           0
cylinders           0
drive               0
dtype: int64
```

9 type

[84]:
```
df["type"].unique()
```

[84]:
```
array(['pickup', 'truck', 'other', nan, 'coupe', 'SUV', 'hatchback',
       'mini-van', 'sedan', 'offroad', 'bus', 'convertible', 'wagon',
       'van'], dtype=object)
```

[85]:
```
df["manufacturer"].unique()
```

[85]:
```
array(['gmc', 'chevrolet', 'toyota', 'ford', 'jeep', 'nissan', 'ram',
       'mazda', 'cadillac', 'honda', 'dodge', 'lexus', 'jaguar', 'buick',
       'chrysler', 'volvo', 'audi', 'infiniti', 'lincoln', 'alfa-romeo',
       'subaru', 'Unknown', 'acura', 'hyundai', 'mercedes-benz', 'bmw',
       'mitsubishi', 'volkswagen', 'porsche', 'kia', 'ferrari', 'mini',
       'pontiac', 'fiat', 'rover', 'tesla', 'saturn', 'mercury',
       'harley-davidson', 'datsun', 'aston-martin', 'land rover'],
      dtype=object)
```

[86]:
```
manufacturer_type_map = {
    'jeep': 'SUV',
    'land rover': 'SUV',
    'rover': 'SUV',
    'gmc': 'pickup',
    'ram': 'pickup',
    'harley-davidson': 'other',

    'ferrari': 'coupe',
    'aston-martin': 'coupe',
    'porsche': 'coupe',
    'alfa-romeo': 'coupe',


    'mini': 'hatchback',
    'fiat': 'hatchback',
    'subaru': 'wagon',


    'mercedes-benz': 'sedan',
    'bmw': 'sedan',
    'audi': 'sedan',
    'lexus': 'sedan',
    'cadillac': 'sedan',
    'lincoln': 'sedan',
    'jaguar': 'sedan',
    'infiniti': 'sedan',
    'acura': 'sedan',
    'volvo': 'sedan',
    'buick': 'sedan',
    'chrysler': 'sedan',


    'toyota': 'sedan',
    'honda': 'sedan',
    'nissan': 'sedan',
    'ford': 'sedan',
    'chevrolet': 'sedan',
    'dodge': 'sedan',
    'hyundai': 'sedan',
    'kia': 'sedan',
    'mazda': 'sedan',
    'volkswagen': 'sedan',
    'mitsubishi': 'sedan',
    'saturn': 'sedan',
    'mercury': 'sedan',
    'pontiac': 'sedan',
```

```
    'datsun': 'sedan',
    'tesla': 'sedan',

    'Unknown': 'other'
}
```

[87]:
```
df['type'] = df['type'].fillna(df['manufacturer'].map(manufacturer_type_map))
```

[88]:
```
df['type'].isna().sum()
```

[88]:
```
np.int64(0)
```

[89]:
```
df["type"].unique()
```

[89]:
```
array(['pickup', 'truck', 'other', 'SUV', 'coupe', 'hatchback',
       'mini-van', 'sedan', 'offroad', 'bus', 'convertible', 'wagon',
       'van'], dtype=object)
```

[90]:
```
df.isna().sum()
```

[90]:
```
region              0
price               0
year                0
manufacturer        0
model               0
fuel                0
odometer            0
title_status        0
transmission        0
VIN            147590
size           267055
type                0
paint_color    106661
description         2
state               0
posting_date        0
condition           0
cylinders           0
drive               0
dtype: int64
```

10 size

[91]:
```
df["size"].unique()
```

[91]:
```
array([nan, 'full-size', 'mid-size', 'compact', 'sub-compact'],
      dtype=object)
```

[92]:
```
type_size_map = {
    'sedan': 'mid-size',
    'coupe': 'compact',
    'hatchback': 'sub-compact',
    'SUV': 'full-size',
    'pickup': 'full-size',
    'truck': 'full-size',
    'van': 'full-size',
    'mini-van': 'mid-size',
    'wagon': 'mid-size',
    'bus': 'full-size',
    'offroad': 'full-size',
    'convertible': 'compact',
    'other' : 'UNKNOWN'
}
```

[93]:
```
df['size'] = df['size'].fillna(df['type'].map(type_size_map))
```

[94]:
```
df.isna().sum()
```

[94]:
```
region              0
price               0
year                0
manufacturer        0
model               0
fuel                0
odometer            0
title_status        0
transmission        0
VIN            147590
size                0
type                0
paint_color    106661
description         2
state               0
posting_date        0
condition           0
cylinders           0
drive               0
dtype: int64
```

11 paint_color

[95]:
```
df["paint_color"] = df["paint_color"].fillna("UNKNOWN")
```

[96]:
```
df.isna().sum()
```

[96]:
```
region              0
price               0
year                0
manufacturer        0
model               0
fuel                0
odometer            0
title_status        0
transmission        0
VIN            147590
size                0
type                0
paint_color         0
description         2
state               0
posting_date        0
condition           0
cylinders           0
drive               0
dtype: int64
```

12 description

[97]:
```
df.dropna(subset=["description"],inplace = True)
```

[98]:
```
df.isna().sum()
```

[98]:
```
region              0
price               0
year                0
manufacturer        0
model               0
fuel                0
odometer            0
title_status        0
transmission        0
VIN            147589
size                0
type                0
paint_color         0
description         0
state               0
posting_date        0
condition           0
cylinders           0
drive               0
```

```
dtype: int64
```

13 VIN

[99]:
```
df["VIN"] = df["VIN"].fillna("UNKNOWN")
```

[100]:
```
df.isna().sum()
```

[100]:
```
region          0
price           0
year            0
manufacturer    0
model           0
fuel            0
odometer        0
title_status    0
transmission    0
VIN             0
size            0
type            0
paint_color     0
description     0
state           0
posting_date    0
condition       0
cylinders       0
drive           0
dtype: int64
```

[101]:
```
df.sample(5)
```

[101]:

	region	price	year	manufacturer	model	fuel	odometer	title_status	transmission	VIN	size	type	paint_color	description	state
390983	fredericksburg	7995	2011.0	chevrolet	equinox	gas	137836.0	clean	automatic	2CNFLNECXB6387597	UNKNOWN	other	black	2011 CHEVROLET EQUINOX LT w/2LT	Offered by…	va
75203	colorado springs	17950	2014.0	gmc	acadia	gas	92916.0	clean	automatic	1GKKVRKD4EJ356956	full-size	SUV				
264268	albany	4950	2013.0	chevrolet	sonic lt	gas	134000.0	clean	automatic	UNKNOWN	mid-size	sedan				
424567	milwaukee	5500	1990.0	cadillac	allante	gas	119000.0	clean	automatic	UNKNOWN	compact	convertible				
264673	albany	38900	2017.0	ford	f-150	gas	52072.0	clean	automatic	1FTEW1EF7HFA06281	mid-size	truck				

```
75203       silver  2014 GMC Acadia AWD 4dr SLT1     Offered by: S…    co
264268       white  LT white 4cyl auto 4door fully loaded stereo c…    ny
424567         red  new battery, tires, convertible top motor, bel…    wi
264673      silver  2017 Ford F-150 XLT 4x4 4dr Supercrew 5.5 ft. …    ny

                    posting_date  condition     cylinders drive
390983  2021-04-23T05:43:29-0400        new  12 cylinders   4wd
75203   2021-04-07T10:36:07-0600        new  12 cylinders   4wd
264268  2021-04-25T11:45:07-0400       good   4 cylinders   fwd
424567  2021-04-18T18:38:08-0500        new   8 cylinders   fwd
264673  2021-04-20T13:25:30-0400  excellent   8 cylinders   4wd
```

14 VIN

15 description

16 posting_date

[102]:
```
df['posting_date'] = pd.to_datetime(df['posting_date'], utc=True)

df['posting_month'] = df['posting_date'].dt.month
df['posting_day'] = df['posting_date'].dt.day
df['posting_weekday'] = df['posting_date'].dt.weekday
df['posting_day'] = df['posting_date'].dt.day_name()
```

[103]:
```
df
```

[103]:
```
          region  price    year manufacturer                     model    fuel  \
27        auburn  33590  2014.0          gmc  sierra 1500 crew cab slt     gas
28        auburn  22590  2010.0    chevrolet            silverado 1500     gas
29        auburn  39590  2020.0    chevrolet       silverado 1500 crew     gas
30        auburn  30990  2017.0       toyota      tundra double cab sr     gas
31        auburn  15000  2013.0         ford                  f-150 xlt     gas
…            …    …      …              …                         …      …
426875  wyoming  23590  2019.0       nissan          maxima s sedan 4d     gas
426876  wyoming  30590  2020.0        volvo  s60 t5 momentum sedan 4d     gas
426877  wyoming  34990  2020.0     cadillac          xt4 sport suv 4d  diesel
426878  wyoming  28990  2018.0        lexus            es 350 sedan 4d     gas
426879  wyoming  30590  2019.0          bmw  4 series 430i gran coupe     gas

        odometer title_status transmission                VIN  …  \
27       57923.0        clean        other  3GTP1VEC4EG551563  …
28       71229.0        clean        other  1GCSCSE06AZ123805  …
29       19160.0        clean        other  3GCPWCED5LG130317  …
30       41124.0        clean        other  5TFRM5F17HX120972  …
31      128000.0        clean    automatic            UNKNOWN  …
```

```
…          …          …          …                 …  …
426875   32226.0        clean        other  1N4AA6AV6KC367801  …
426876   12029.0        clean        other  7JR102FKXLG042696  …
426877    4174.0        clean        other  1GYFZFR46LF088296  …
426878   30112.0        clean        other  58ABK1GG4JU103853  …
426879   22716.0        clean        other  WBA4J1C58KBM14708  …

       paint_color                                        description state  \
27           white  Carvana is the safer way to buy a car During t…    al
28            blue  Carvana is the safer way to buy a car During t…    al
29             red  Carvana is the safer way to buy a car During t…    al
30             red  Carvana is the safer way to buy a car During t…    al
31           black  2013 F-150 XLT V6 4 Door. Good condition. Leve…    al
…              …                                                 …   …
426875     UNKNOWN  Carvana is the safer way to buy a car During t…    wy
426876         red  Carvana is the safer way to buy a car During t…    wy
426877       white  Carvana is the safer way to buy a car During t…    wy
426878      silver  Carvana is the safer way to buy a car During t…    wy
426879     UNKNOWN  Carvana is the safer way to buy a car During t…    wy

                     posting_date  condition     cylinders drive posting_month  \
27     2021-05-04 17:31:18+00:00       good   8 cylinders   4wd             5
28     2021-05-04 17:31:08+00:00       good   8 cylinders   4wd             5
29     2021-05-04 17:31:25+00:00       good   8 cylinders   4wd             5
30     2021-05-04 15:41:31+00:00       good   8 cylinders   4wd             5
31     2021-05-03 19:02:03+00:00  excellent   6 cylinders   rwd             5
…                            …          …            …   …           …
426875 2021-04-04 09:21:31+00:00       good   6 cylinders   fwd             4
426876 2021-04-04 09:21:29+00:00       good  12 cylinders   fwd             4
426877 2021-04-04 09:21:17+00:00       good  12 cylinders   4wd             4
426878 2021-04-04 09:21:11+00:00       good   6 cylinders   fwd             4
426879 2021-04-04 09:21:07+00:00       good  12 cylinders   rwd             4

       posting_day  posting_weekday
27         Tuesday                1
28         Tuesday                1
29         Tuesday                1
30         Tuesday                1
31          Monday                0
…             …                …
426875      Sunday                6
426876      Sunday                6
426877      Sunday                6
426878      Sunday                6
426879      Sunday                6

[375715 rows x 22 columns]
```

[104]:
```
df['has_phone'] = df['description'].str.contains(r'\d{3}-\d{3}-\d{4}',␣
↪regex=True).astype(int)
df['desc_word_count'] = df['description'].apply(lambda x: len(str(x).split()))
df['has_warranty'] = df['description'].str.contains('warranty', case=False).
↪astype(int)
df['is_one_owner'] = df['description'].str.contains('one owner|1st owner',␣
↪case=False).astype(int)
```

[105]:
```
df["has_phone"].unique()
```

[105]:
```
array([1, 0])
```

[106]:
```
df["desc_word_count"].unique()
```

[106]:
```
array([ 681,  692,  690, …, 2870, 2781, 2934])
```

[107]:
```
df["has_warranty"].unique()
```

[107]:
```
array([0, 1])
```

[108]:
```
df["is_one_owner"].unique()
```

[108]:
```
array([0, 1])
```

[109]:
```
df['vin_country_code'] = df['VIN'].apply(lambda x: x[0] if x != "UNKNOWN" else␣
↪"UNKNOWN")
```

[110]:
```
country_map = {'1': 'USA', '4': 'USA', '5': 'USA', 'J': 'Japan', 'K': 'South␣
↪Korea', 'W': 'Germany'}
```

[111]:
```
df['origin_country'] = df['vin_country_code'].map(country_map).fillna('Other/
↪Unknown')
```

[112]:
```
df['origin_country']
```

[112]:
```
27        Other/Unknown
28                  USA
29        Other/Unknown
30                  USA
31        Other/Unknown
              …
426875              USA
426876    Other/Unknown
426877              USA
426878              USA
426879          Germany
Name: origin_country, Length: 375715, dtype: object
```

17 Columns Cleaning

[113]:
```
df.columns
```

[113]:
```
Index(['region', 'price', 'year', 'manufacturer', 'model', 'fuel', 'odometer',
       'title_status', 'transmission', 'VIN', 'size', 'type', 'paint_color',
       'description', 'state', 'posting_date', 'condition', 'cylinders',
       'drive', 'posting_month', 'posting_day', 'posting_weekday', 'has_phone',
       'desc_word_count', 'has_warranty', 'is_one_owner', 'vin_country_code',
       'origin_country'],
      dtype='object')
```

Region

[114]:
```
df["region"].sample(43)
```

[114]:
```
29508              imperial county
284600               new hampshire
385471               wichita falls
356099                   knoxville
276952               new york city
180546                      maine
234443                   charlotte
190497      western massachusetts
29123              humboldt county
113852               south florida
311510                       bend
316215                     eugene
292493                  cleveland
339941    scranton / wilkes-barre
204620                   kalamazoo
251981                  central NJ
6679           anchorage / mat-su
227060                 great falls
337214                  pittsburgh
338923                     reading
329580                  harrisburg
166379             kansas city, MO
5900           anchorage / mat-su
306249               oklahoma city
18882                    jonesboro
200753                grand rapids
325755                       salem
47848                      redding
322427                    portland
223583                    billings
165525             kansas city, MO
309560                       tulsa
```

```
332196              lehigh valley
111065              south florida
389429            charlottesville
84841                    hartford
160683     omaha / council bluffs
169010                   wichita
250671                central NJ
308941                     tulsa
366088                    austin
290624                cincinnati
171109                 lexington
Name: region, dtype: object
```

[115]:
```
df["region"]=df["region"].replace([",","/","'","-","  "]," ",regex=True)
```

[116]:
```
df["region"].sample(200)
```

[116]:
```
409662              seattle tacoma
282109           utica rome oneida
403000    kennewick pasco richland
54717                   san diego
182834                  baltimore
                   …
364665                     austin
375783                    houston
317697                     eugene
410409       skagit   island   SJI
138348     spokane   coeur d alene
Name: region, Length: 200, dtype: object
```

17.0.1 Price

[117]:
```
df["price"].dtypes
```

[117]:
```
dtype('int64')
```

17.0.2 Year

[118]:
```
df["year"].dtypes
```

[118]:
```
dtype('float64')
```

[119]:
```
df["year"] = df["year"].astype(int)
```

[120]:
```
df["year"].dtypes
```

[120]:
```
dtype('int64')
```

[121]:
```
df["year"].unique
```

[121]:
```
<bound method Series.unique of 27        2014
28        2010
29        2020
30        2017
31        2013
          …
426875    2019
426876    2020
426877    2020
426878    2018
426879    2019
Name: year, Length: 375715, dtype: int64>
```

17.0.3 Manufacturer

[122]:
```
df["manufacturer"].unique()
```

[122]:
```
array(['gmc', 'chevrolet', 'toyota', 'ford', 'jeep', 'nissan', 'ram',
       'mazda', 'cadillac', 'honda', 'dodge', 'lexus', 'jaguar', 'buick',
       'chrysler', 'volvo', 'audi', 'infiniti', 'lincoln', 'alfa-romeo',
       'subaru', 'Unknown', 'acura', 'hyundai', 'mercedes-benz', 'bmw',
       'mitsubishi', 'volkswagen', 'porsche', 'kia', 'ferrari', 'mini',
       'pontiac', 'fiat', 'rover', 'tesla', 'saturn', 'mercury',
       'harley-davidson', 'datsun', 'aston-martin', 'land rover'],
      dtype=object)
```

[123]:
```
df["manufacturer"] = df["manufacturer"].replace(["-"," ",regex=True)
```

[124]:
```
df["manufacturer"].unique()
```

[124]:
```
array(['gmc', 'chevrolet', 'toyota', 'ford', 'jeep', 'nissan', 'ram',
       'mazda', 'cadillac', 'honda', 'dodge', 'lexus', 'jaguar', 'buick',
       'chrysler', 'volvo', 'audi', 'infiniti', 'lincoln', 'alfa romeo',
       'subaru', 'Unknown', 'acura', 'hyundai', 'mercedes benz', 'bmw',
       'mitsubishi', 'volkswagen', 'porsche', 'kia', 'ferrari', 'mini',
       'pontiac', 'fiat', 'rover', 'tesla', 'saturn', 'mercury',
       'harley davidson', 'datsun', 'aston martin', 'land rover'],
      dtype=object)
```

[125]:
```
df["manufacturer"].dtypes
```

[125]:
```
dtype('O')
```

17.0.4 Model

[126]:
```
df["model"].unique()
```

[126]:
```
array(['sierra 1500 crew cab slt', 'silverado 1500',
       'silverado 1500 crew', …, 'gand wagoneer', '96 Suburban',
       'Paige Glenbrook Touring'], dtype=object)
```

[127]:
```
df["model"].sample(50)
```

[127]:
```
164125                          xc90
345829                 transit cargo
244358       frontier crew cab pro-4x
246216                     jetta tsi
222546            wrangler unlimited
149653                           500
400667                       tundra
224779          super duty f-350 srw
90893                            s70
308953         silverado 1500 regular
16270              wrangler unlimited
215119                       terrain
158541                        is 250
55350                     murano sl
105464                         f250
195676                         f250
157790                   civic sedan
164867                      escalade
214026                 accord hybrid
371988                grand cherokee
348969                         regal
203527                       odyssey
423002                         f-150
126033                    beetle bug
188826                      explorer
94652                       rogue sv
156178                     avalanche
44700         silverado 1500 crew cab
33746                      camry 2001
167471    focus electric hatchback 4d
313659                       2500 st
285746         silverado medium duty
343014                        maxima
24797                     challenger
415662        thunderbird convertible
258682                      explorer
230796         sierra 1500 double cab
356046                       transit
```

```
122602                            545i
370478                              q5
38713                             f150
398530                            cx-9
167000                         mustang
36295            super duty f-350 srw
93594                             c10
275654                            rdx
40243                          murano
153775            mustang gt coupe 2d
2993        a4 ultra premium sedan 4d
335801                           camry
Name: model, dtype: object
```

[128]:
```
df=df.drop(["model"],axis = 1)
```

[129]:
```
df.columns
```

[129]:
```
Index(['region', 'price', 'year', 'manufacturer', 'fuel', 'odometer',
       'title_status', 'transmission', 'VIN', 'size', 'type', 'paint_color',
       'description', 'state', 'posting_date', 'condition', 'cylinders',
       'drive', 'posting_month', 'posting_day', 'posting_weekday', 'has_phone',
       'desc_word_count', 'has_warranty', 'is_one_owner', 'vin_country_code',
       'origin_country'],
      dtype='object')
```

17.0.5 Fuel

[130]:
```
df["fuel"].unique()
```

[130]:
```
array(['gas', 'other', 'diesel', 'hybrid', 'electric'], dtype=object)
```

17.0.6 Odometer

[131]:
```
df["odometer"].dtypes
```

[131]:
```
dtype('float64')
```

[132]:
```
df["odometer"]=df["odometer"].round()
```

[133]:
```
df["odometer"] = df["odometer"].astype(int)
```

[134]:
```
df["odometer"].dtypes
```

[134]:
```
dtype('int64')
```

17.0.7 Title_status

[135]:
```
df["title_status"].unique()
```

[135]:
```
array(['clean', 'rebuilt', 'lien', 'salvage', 'missing', 'parts only'],
      dtype=object)
```

17.0.8 Transmission

[136]:
```
df["transmission"].unique()
```

[136]:
```
array(['other', 'automatic', 'manual'], dtype=object)
```

17.0.9 VIN

[137]:
```
df["VIN"].unique()
```

[137]:
```
array(['3GTP1VEC4EG551563', '1GCSCSE06AZ123805', '3GCPWCED5LG130317', …,
       '2HGES15535H620534', '1FDWF37P64EA24868', 'SAJGX2749VCOO8376'],
      dtype=object)
```

[138]:
```
df=df.drop(["VIN"],axis=1)
```

[139]:
```
df.columns
```

[139]:
```
Index(['region', 'price', 'year', 'manufacturer', 'fuel', 'odometer',
       'title_status', 'transmission', 'size', 'type', 'paint_color',
       'description', 'state', 'posting_date', 'condition', 'cylinders',
       'drive', 'posting_month', 'posting_day', 'posting_weekday', 'has_phone',
       'desc_word_count', 'has_warranty', 'is_one_owner', 'vin_country_code',
       'origin_country'],
      dtype='object')
```

17.0.10 Size

[140]:
```
df["size"].unique()
```

[140]:
```
array(['full-size', 'UNKNOWN', 'compact', 'sub-compact', 'mid-size'],
      dtype=object)
```

[141]:
```
df["size"]=df["size"].replace("-"," ",regex=True)
```

17.0.11 Type

[142]:
```
df["type"].unique()
```

[142]:
```
array(['pickup', 'truck', 'other', 'SUV', 'coupe', 'hatchback',
       'mini-van', 'sedan', 'offroad', 'bus', 'convertible', 'wagon',
       'van'], dtype=object)
```

[143]:
```
df["type"]=df["type"].replace("-"," ",regex=True)
```

17.0.12 Paint_color

[144]:
```
df["paint_color"].unique()
```

[144]:
```
array(['white', 'blue', 'red', 'black', 'silver', 'grey', 'UNKNOWN',
       'brown', 'yellow', 'orange', 'green', 'custom', 'purple'],
      dtype=object)
```

17.0.13 Description

[145]:
```
df=df.drop(["description"],axis=1)
```

[146]:
```
df.columns
```

[146]:
```
Index(['region', 'price', 'year', 'manufacturer', 'fuel', 'odometer',
       'title_status', 'transmission', 'size', 'type', 'paint_color', 'state',
       'posting_date', 'condition', 'cylinders', 'drive', 'posting_month',
       'posting_day', 'posting_weekday', 'has_phone', 'desc_word_count',
       'has_warranty', 'is_one_owner', 'vin_country_code', 'origin_country'],
      dtype='object')
```

17.0.14 State

[147]:
```
df["state"].unique()
```

[147]:
```
array(['al', 'ak', 'az', 'ar', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga',
       'hi', 'id', 'il', 'in', 'ia', 'ks', 'ky', 'la', 'me', 'md', 'ma',
       'mi', 'mn', 'ms', 'mo', 'mt', 'nc', 'ne', 'nv', 'nj', 'nm', 'ny',
       'nh', 'nd', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx',
       'ut', 'vt', 'va', 'wa', 'wv', 'wi', 'wy'], dtype=object)
```

17.0.15 Posting_date

[148]:
```
df=df.drop(["posting_date"],axis =1)
```

[149]:
```
df.columns
```

[149]:
```
Index(['region', 'price', 'year', 'manufacturer', 'fuel', 'odometer',
       'title_status', 'transmission', 'size', 'type', 'paint_color', 'state',
       'condition', 'cylinders', 'drive', 'posting_month', 'posting_day',
       'posting_weekday', 'has_phone', 'desc_word_count', 'has_warranty',
       'is_one_owner', 'vin_country_code', 'origin_country'],
      dtype='object')
```

17.0.16 Condition

[150]:
```
df["condition"].unique()
```

[150]:
```
array(['good', 'excellent', 'fair', 'like new', 'new', 'salvage'],
      dtype=object)
```

17.0.17 Cylinders

[151]:
```
df["cylinders"].unique()
```

[151]:
```
array(['8 cylinders', '6 cylinders', '12 cylinders', '4 cylinders',
       '5 cylinders', 'other', '3 cylinders', '10 cylinders'],
      dtype=object)
```

17.0.18 Drive

[152]:
```
df["drive"].unique()
```

[152]:
```
array(['4wd', 'rwd', 'fwd'], dtype=object)
```

17.0.19 posting_month

[153]:
```
df["posting_month"].unique()
```

[153]:
```
array([5, 4], dtype=int32)
```

17.0.20 Posting_day

[154]:
```
df["posting_day"].unique()
```

[154]:
```
array(['Tuesday', 'Monday', 'Sunday', 'Saturday', 'Friday', 'Thursday',
       'Wednesday'], dtype=object)
```

17.0.21 Posting_weekday

[155]:
```
df["posting_weekday"].unique()
```

[155]:
```
array([1, 0, 6, 5, 4, 3, 2], dtype=int32)
```

17.0.22 Has_phone

[156]:
```
df["has_phone"].unique()
```

[156]:
```
array([1, 0])
```

[157]:
```
df["has_phone"] =df["has_phone"].replace(0,"Not Has")
df["has_phone"] =df["has_phone"].replace(1,"Has")
```

[158]:
```
df["has_phone"].unique()
```

[158]:
```
array(['Has', 'Not Has'], dtype=object)
```

[159]:
```
df["has_phone"].dtypes
```

[159]:
```
dtype('O')
```

17.0.23 Desc_word_count

[160]:
```
df["desc_word_count"].unique()
```

[160]:
```
array([ 681,  692,  690, …, 2870, 2781, 2934])
```

[161]:
```
df["desc_word_count"].dtypes
```

[161]:
```
dtype('int64')
```

17.0.24 Has_warranty

[162]:
```
df["has_warranty"].unique()
```

[162]:
```
array([0, 1])
```

[163]:
```
df["has_warranty"] =df["has_warranty"].replace(0,"Not Warranty")
df["has_warranty"] =df["has_warranty"].replace(1,"Warranty")
```

[164]:
```
df["has_warranty"].unique()
```

[164]:
```
array(['Not Warranty', 'Warranty'], dtype=object)
```

[ ]:
```

```

17.0.25 Is_one_owner

[165]:
```
df["is_one_owner"].unique()
```

[165]:
```
array([0, 1])
```

[166]:
```
df["is_one_owner"] =df["is_one_owner"].replace(0,"Other")
df["is_one_owner"] =df["is_one_owner"].replace(1,"One Owner")
```

[167]:
```
df["is_one_owner"].unique()
```

[167]:
```
array(['Other', 'One Owner'], dtype=object)
```

17.0.26 Vin_country_code

[168]:
```
df["vin_country_code"].unique()
```

[168]:
```
array(['3', '1', '5', 'UNKNOWN', 'J', 'Z', '2', 'S', 'K', 'Y', '7', 'W',
       'L', '4', 'N', '0', 'B', 'M', '6', 'T', 'H', 'C', 'D', 'V', 'P',
       '8', 'I', 'F', 'A', 'U', '9', 'R', 'G', 'X', 'E', 'O'],
      dtype=object)
```

[169]:
```
df.columns
```

[169]:
```
Index(['region', 'price', 'year', 'manufacturer', 'fuel', 'odometer',
       'title_status', 'transmission', 'size', 'type', 'paint_color', 'state',
       'condition', 'cylinders', 'drive', 'posting_month', 'posting_day',
       'posting_weekday', 'has_phone', 'desc_word_count', 'has_warranty',
       'is_one_owner', 'vin_country_code', 'origin_country'],
      dtype='object')
```

17.0.27 Origin_country

[170]:
```
df["origin_country"].unique()
```

[170]:
```
array(['Other/Unknown', 'USA', 'Japan', 'South Korea', 'Germany'],
      dtype=object)
```

[171]:
```
df["origin_country"]=df["origin_country"].replace(["/Unknown"],"",regex= True)
```

[172]:
```
df["origin_country"].unique()
```

[172]:
```
array(['Other', 'USA', 'Japan', 'South Korea', 'Germany'], dtype=object)
```

[ ]:
```

```

18 Reset index

[ ]:
```

```

[173]:
```
df = df.reset_index(drop=True)
```

[174]:
```
df.sample()
```

[174]:
```
            region  price  year manufacturer fuel  odometer title_status  \
351667  winchester  10995  2016       toyota  gas    127161        clean

       transmission      size   type  … drive posting_month posting_day  \
351667    automatic  mid size  sedan  …   fwd             4      Friday

       posting_weekday has_phone  desc_word_count has_warranty  is_one_owner  \
```

```
351667               4   Not Has              433     Warranty         Other

       vin_country_code  origin_country
351667                2           Other

[1 rows x 24 columns]
```

[ ]:

[ ]:

19 Export New Data

[175]:
```
#df.to_csv("D:/projects/USED_CAR/vehicles.csv/New_Data.csv")
```

[176]:
```
df.columns
```

[176]:
```
Index(['region', 'price', 'year', 'manufacturer', 'fuel', 'odometer',
       'title_status', 'transmission', 'size', 'type', 'paint_color', 'state',
       'condition', 'cylinders', 'drive', 'posting_month', 'posting_day',
       'posting_weekday', 'has_phone', 'desc_word_count', 'has_warranty',
       'is_one_owner', 'vin_country_code', 'origin_country'],
      dtype='object')
```

20 ANALYSIS

Section 1 :

- What is the average price for each car brand? - What are the top 5 most expensive car brands in the dataset? - How does the price vary based on fuel type (Gasoline, Diesel, Hybrid, EV)? - Are there significant price outliers for the same car model? - What is the price difference between cars sold by dealers vs. private owners? - Do metallic colors command a higher price compared to matte colors? Section 2: Mileage & Usage Analysis - What is the correlation between mileage and price? - What is the average mileage for cars that are 5 years old? - Do high-mileage cars sell faster due to their lower prices? - Is there a specific mileage threshold where the price drops significantly?

Section 3: Technical Specifications - Do cars with larger engine capacities sell for less due to rising fuel costs? - What is the most frequently occurring engine size in the market? - Does the presence of a sunroof significantly impact the final selling price? - What is the price gap between Front-Wheel Drive (FWD) and Rear-Wheel Drive (RWD) cars in the same category? - Do turbocharged engines hold their value better than naturally aspirated ones?

Section 4: Age & Depreciation - What is the depreciation rate of a car within its first 3 years? - Which brands have the highest resale value retention over time? - Are current-year used cars (like-new) ever priced higher than their brand-new counterparts? - What is the oldest model year in the dataset, and is it considered a classic/antique or just old? - How do the prices of pre-2010 models compare to post-2010 models?

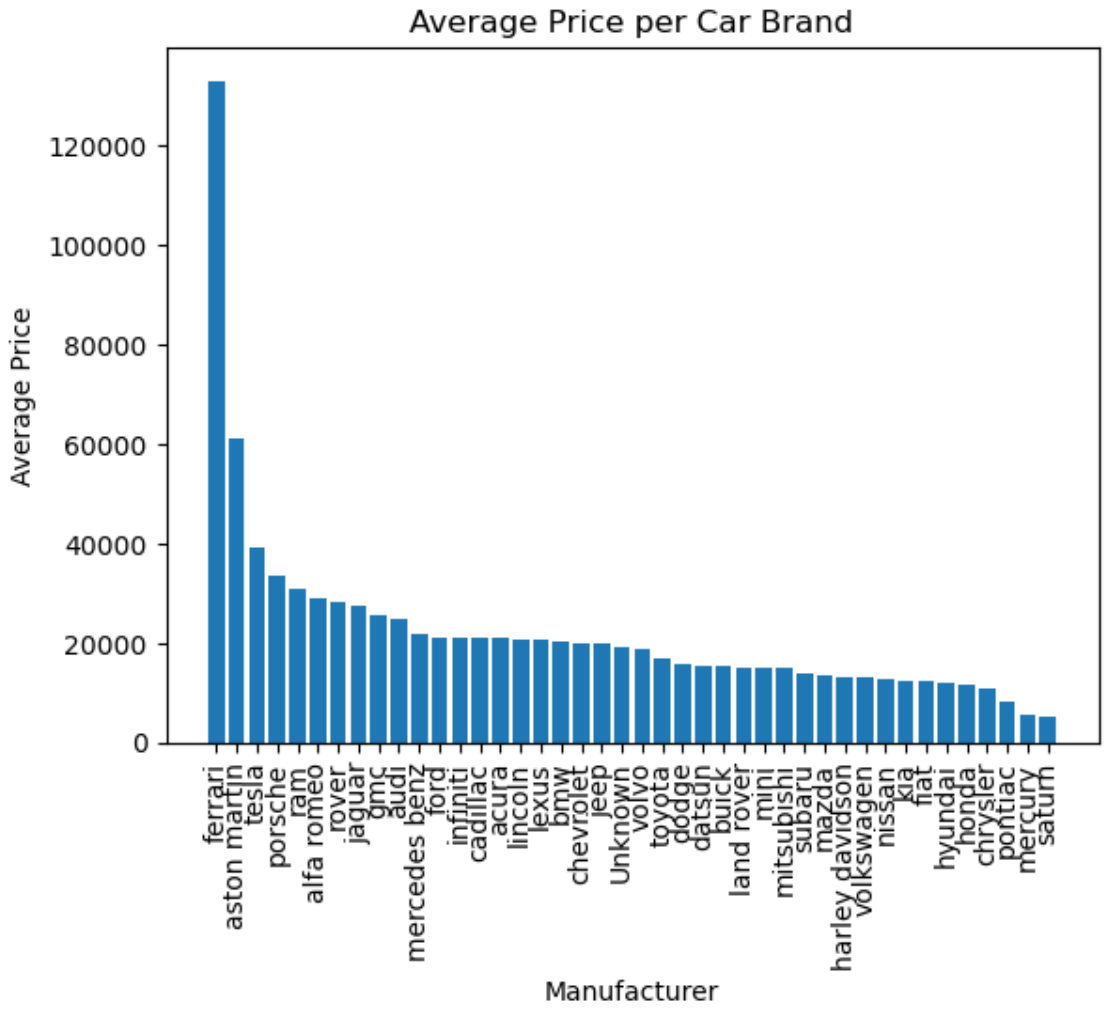
20.1 Section 1

20.1.1 - What is the average price for each car brand?

[212]:
```
average_prices_sorted = df.groupby("manufacturer")["price"].mean()
average_prices_sorted = average_prices_sorted.sort_values(ascending=False)
```

[213]:
```
plt.bar(average_prices_sorted.index,average_prices_sorted.values)
plt.xticks(rotation=90)
plt.xlabel("Manufacturer")
plt.ylabel("Average Price")
plt.title("Average Price per Car Brand")

plt.savefig("average_price_by_brand.png")
```



20.1.2 What are the top 5 most expensive car brands in the dataset?

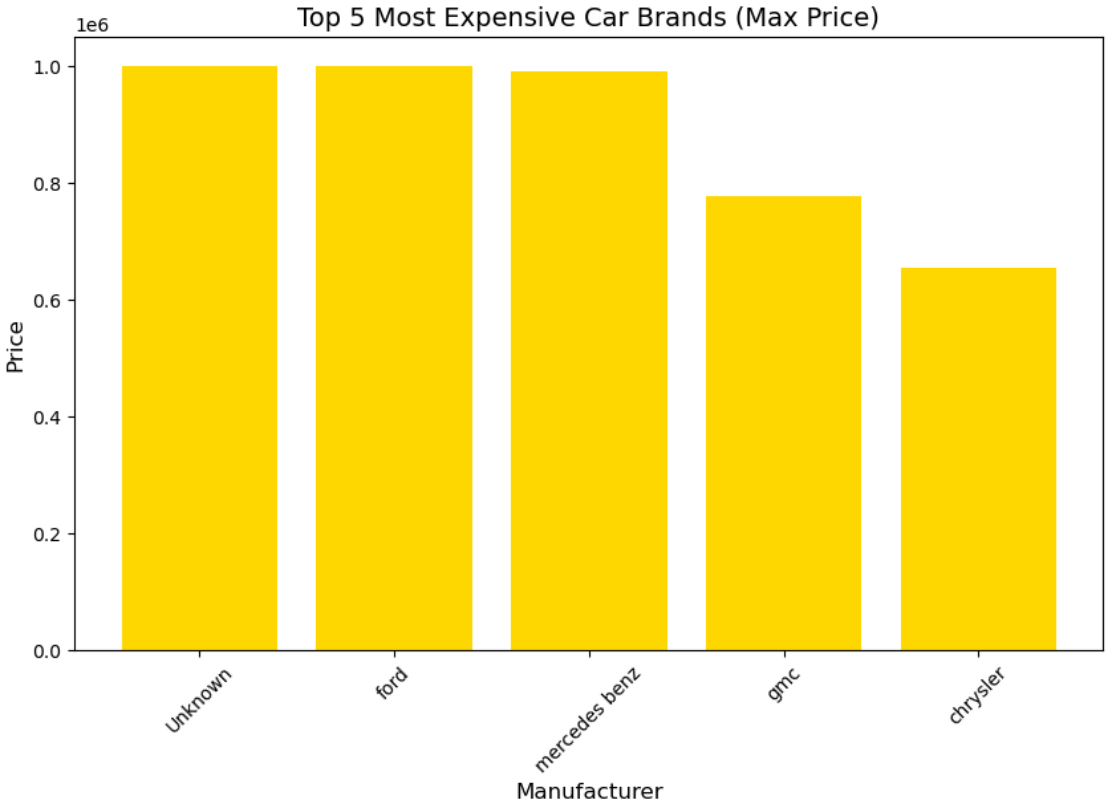
[214]:
```
# 1.            ) 5 (
top_5_expensive = df.groupby("manufacturer")["price"].max().
 ↪sort_values(ascending=False).head(5)

# 2.
plt.figure(figsize=(10, 6)) #
plt.bar(top_5_expensive.index, top_5_expensive.values, color='gold') #
 ↪

# 3.                (Labels)
plt.title("Top 5 Most Expensive Car Brands (Max Price)", fontsize=14)
plt.xlabel("Manufacturer", fontsize=12)
plt.ylabel("Price", fontsize=12)

# 4.
plt.xticks(rotation=45)

# 5.
plt.show()
```



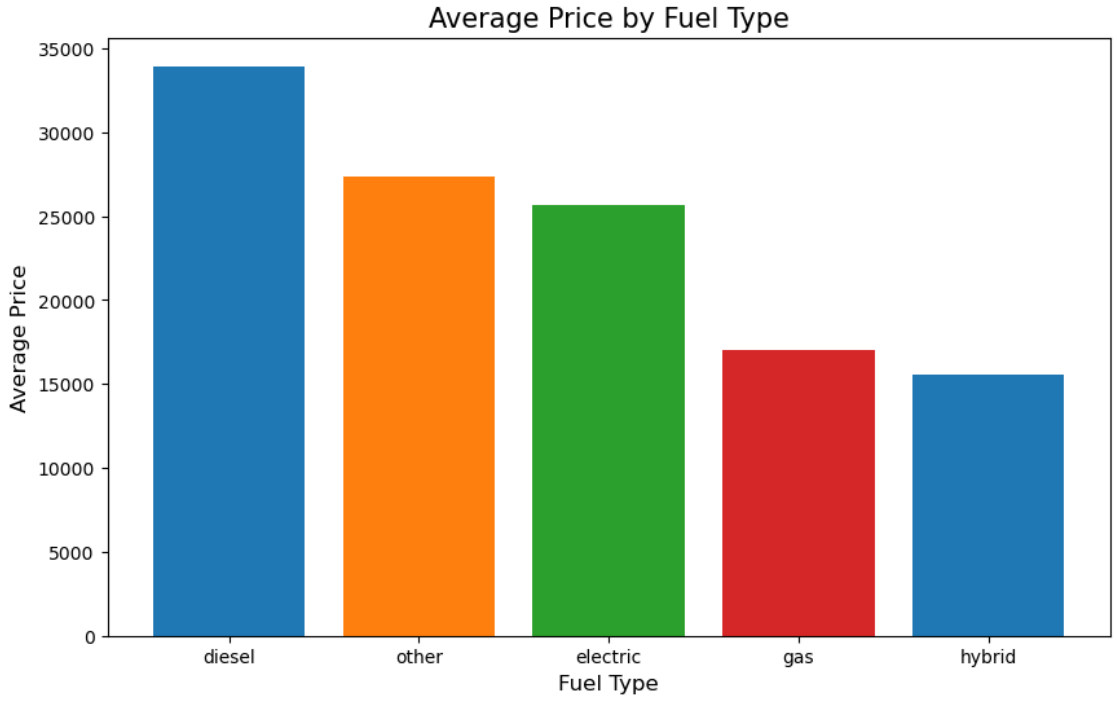
20.1.3 How does the price vary based on fuel type (Gasoline, Diesel, Hybrid, EV)?

[185]:
```
# 1.
fuel_price_analysis = df.groupby("fuel")["price"].mean().
  ↪sort_values(ascending=False)

# 2.
plt.figure(figsize=(10, 6))
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728'] #
plt.bar(fuel_price_analysis.index, fuel_price_analysis.values, color=colors)

# 3.          (Labels)
plt.title("Average Price by Fuel Type", fontsize=15)
plt.xlabel("Fuel Type", fontsize=12)
plt.ylabel("Average Price", fontsize=12)

# 4.
plt.show()
```

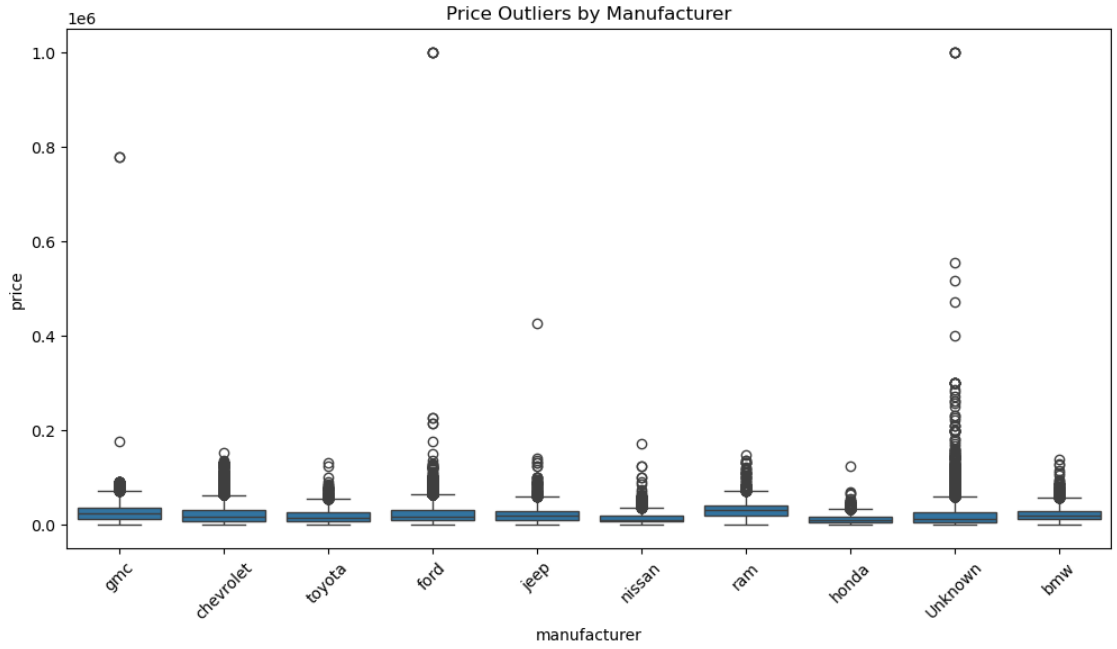


20.1.4 Are there significant price outliers for the same car model?

[187]:
```
#        10
top_10_brands = df['manufacturer'].value_counts().nlargest(10).index
df_subset = df[df['manufacturer'].isin(top_10_brands)]

plt.figure(figsize=(12, 6))
sns.boxplot(x='manufacturer', y='price', data=df_subset)

plt.title("Price Outliers by Manufacturer")
plt.xticks(rotation=45)
plt.show()
```



[188]:
```
#
def find_outliers(group):
    Q1 = group.quantile(0.25)
    Q3 = group.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    #
    return group[(group < lower_bound) | (group > upper_bound)]

#
outliers = df.groupby('manufacturer')['price'].apply(find_outliers)
```

```
print("                :")
print(outliers.count())

#       10
print(outliers.head(10))
```

```
                :
6689
manufacturer
Unknown       670      63990
              2443     80000
              4467    145000
              4724     80000
              6616     85000
              6995    145000
              7916    112900
              8398    229500
              8845     75000
              9052     65750
Name: price, dtype: int64
```

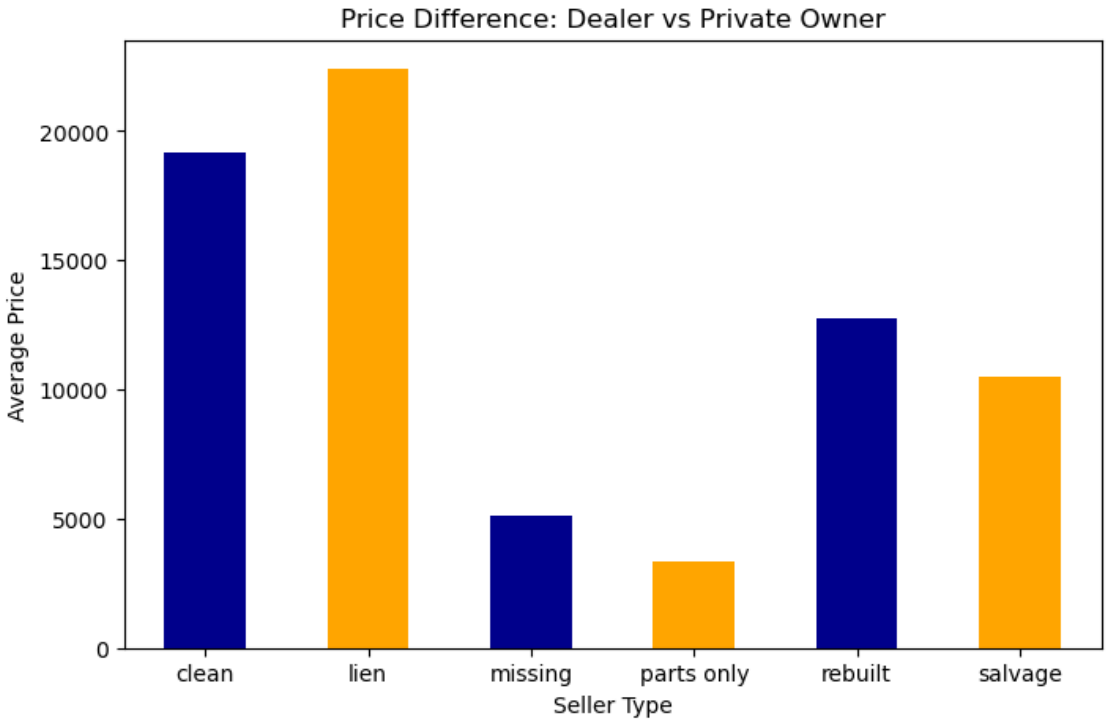
20.1.5 What is the price difference between cars sold by dealers vs. private owners?

[190]:
```
#
#    :     'title_status'
dealer_vs_private = df.groupby('title_status')['price'].mean()
```

[191]:
```
#
plt.figure(figsize=(8, 5))
dealer_vs_private.plot(kind='bar', color=['darkblue', 'orange'])

plt.title("Price Difference: Dealer vs Private Owner")
plt.ylabel("Average Price")
plt.xlabel("Seller Type")
plt.xticks(rotation=0) #

plt.show()
```

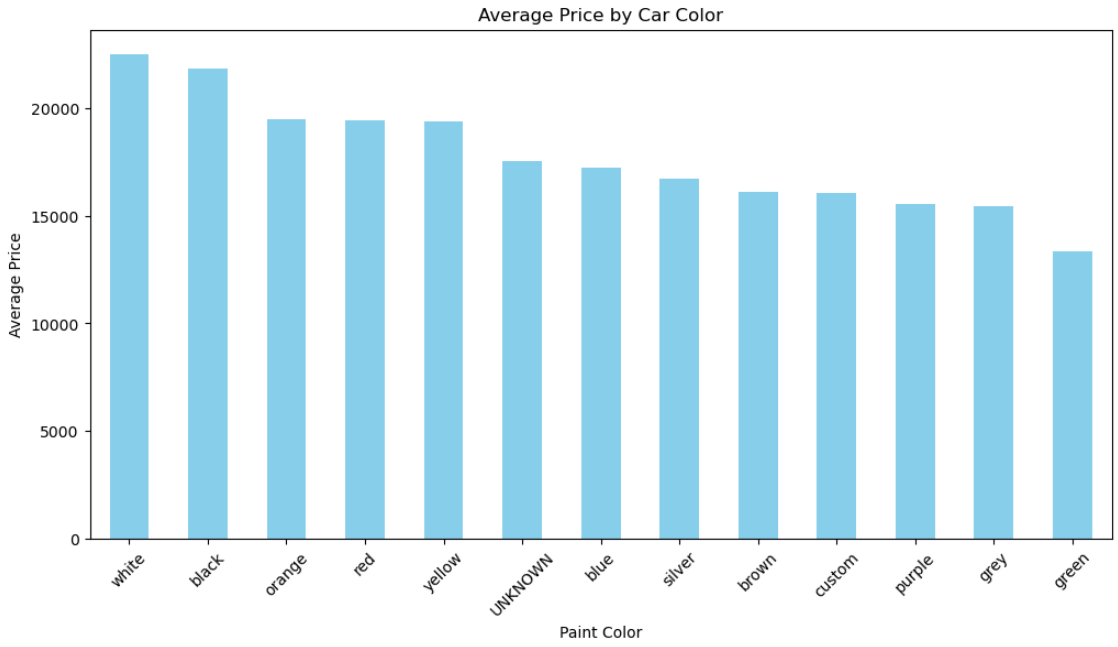


20.1.6 Do metallic colors command a higher price compared to matte colors?

[193]:
```
# 1.
color_prices = df.groupby('paint_color')['price'].mean().
 ↪sort_values(ascending=False)

# 2.
plt.figure(figsize=(12, 6))
color_prices.plot(kind='bar', color='skyblue')

plt.title("Average Price by Car Color")
plt.xlabel("Paint Color")
plt.ylabel("Average Price")
plt.xticks(rotation=45)
plt.show()
```



[194]:
```
#
df['is_metallic'] = df['paint_color'].str.contains('metallic', case=False,␣
 ↪na=False)

#
metallic_comparison = df.groupby('is_metallic')['price'].mean()
print(metallic_comparison)
```

```
is_metallic
False    18935.625873
Name: price, dtype: float64
```

20.2 Section 2 Mileage & Usage Analysis

20.2.1 What is the correlation between mileage and price?

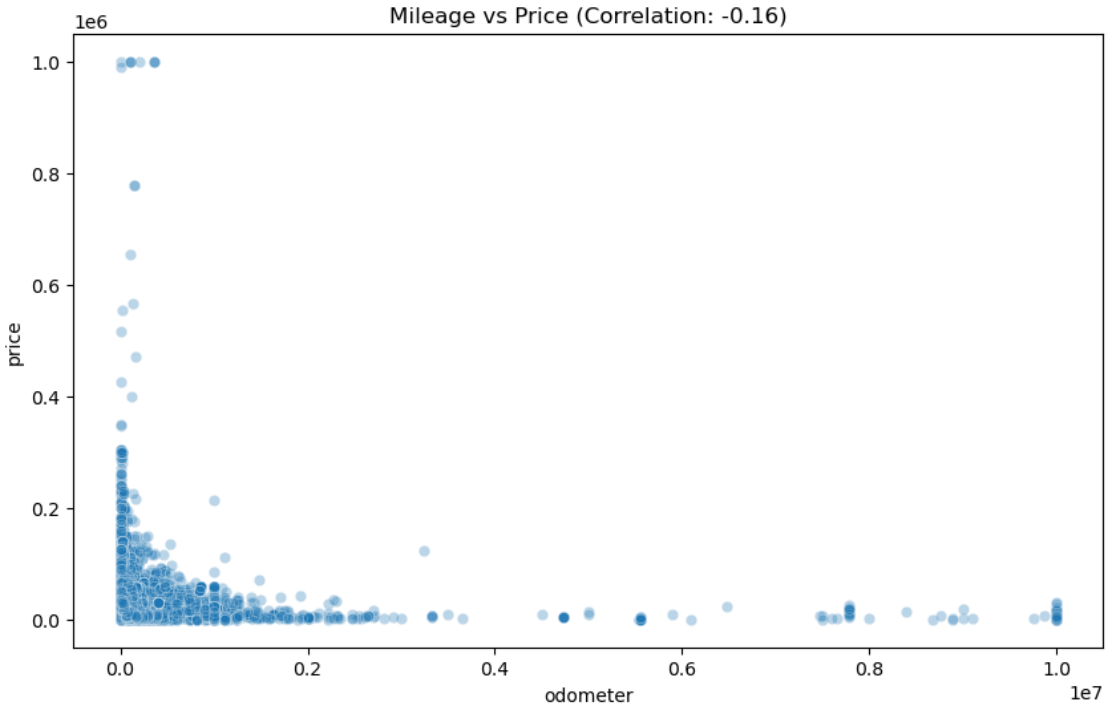
[197]:
```
#
correlation = df['odometer'].corr(df['price'])
print(f"Correlation between mileage and price: {correlation}")

#              (Scatter Plot)
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='odometer', y='price', alpha=0.3)
plt.title(f"Mileage vs Price (Correlation: {correlation:.2f})")
```

```
plt.show()
```

```
Correlation between mileage and price: -0.16338281431469398
```



20.2.2 What is the average mileage for cars that are 5 years old?

[199]:
```
#                ) 5                (
target_year = df['year'].max() - 5
avg_mileage_5y = df[df['year'] == target_year]['odometer'].mean()

print(f"Average mileage for {target_year} models: {avg_mileage_5y:.2f} miles")
```

```
Average mileage for 2017 models: 54223.88 miles
```

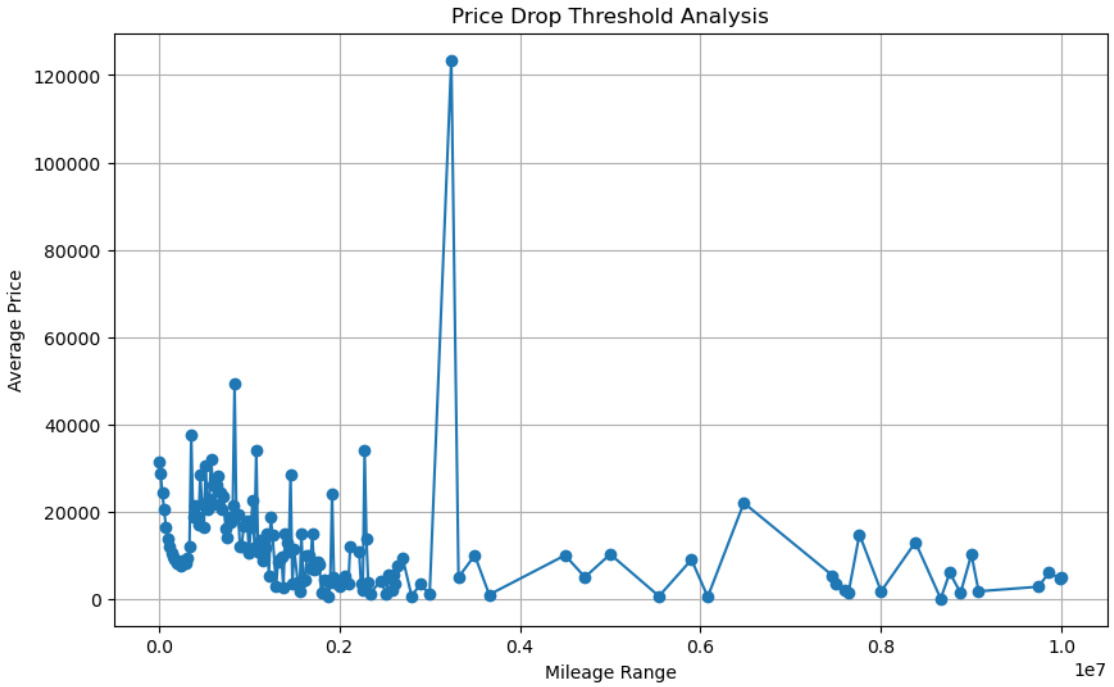
20.2.3 Is there a specific mileage threshold where the price drops significantly?

[201]:
```
#                )   20   (
df['mileage_range'] = (df['odometer'] // 20000) * 20000

#
threshold_analysis = df.groupby('mileage_range')['price'].mean()

#
plt.figure(figsize=(10, 6))
threshold_analysis.plot(kind='line', marker='o')
```

```
plt.title("Price Drop Threshold Analysis")
plt.xlabel("Mileage Range")
plt.ylabel("Average Price")
plt.grid(True)
plt.show()
```



20.3 Section 3: Technical Specifications

20.3.1 - Do cars with larger engine capacities sell for less due to rising fuel costs?

[204]:
```
#
engine_impact = df.groupby('cylinders')['price'].mean().
  ↪sort_values(ascending=False)
print(engine_impact)
```

```
cylinders
8 cylinders     23904.411236
10 cylinders    21384.268072
12 cylinders    20794.286546
other           20440.480533
6 cylinders     18441.828986
3 cylinders     13085.836522
4 cylinders     11069.877609
5 cylinders      8073.489191
Name: price, dtype: float64
```

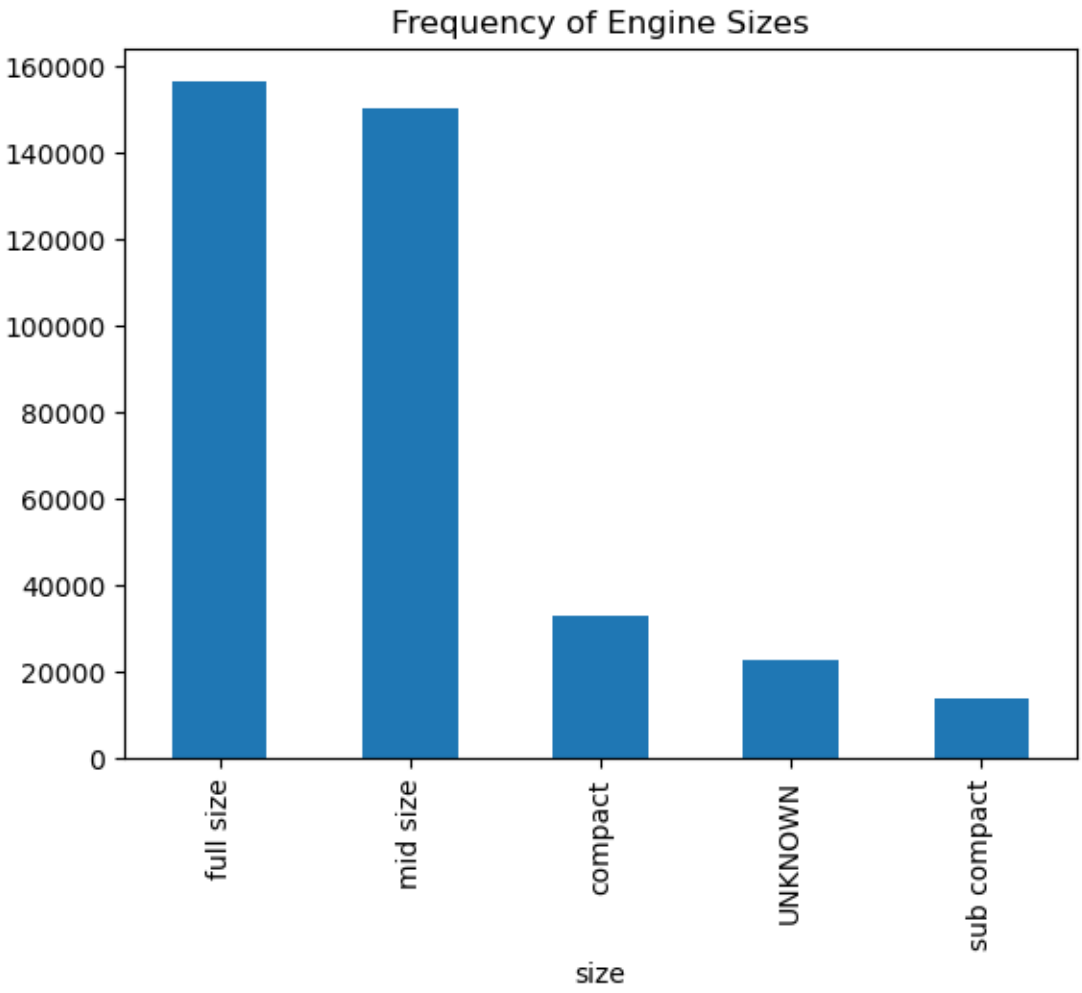
20.3.2 - What is the most frequently occurring engine size in the market?

[216]:
```
#
most_common_size = df['size'].mode()[0]
print(f"The most frequently occurring engine size is: {most_common_size}")

#
df['size'].value_counts().plot(kind='bar', title="Frequency of Engine Sizes")
```

```
The most frequently occurring engine size is: full size
```

[216]:
```
<Axes: title={'center': 'Frequency of Engine Sizes'}, xlabel='size'>
```



[217]:
```
#
drive_comparison = df[df['drive'].isin(['fwd', 'rwd'])].
 ↪groupby('drive')['price'].mean()
```

```
price_gap = drive_comparison['rwd'] - drive_comparison['fwd']
print(f"Price Gap: RWD cars are on average ${price_gap:.2f} more expensive than␣
 ↪FWD")
```

```
Price Gap: RWD cars are on average $8719.89 more expensive than FWD
```

[ ]:

[218]:
```
#                                    )   10 (
old_cars = df[df['year'] < (df['year'].max() - 10)]
retention = old_cars.groupby('manufacturer')['price'].mean().
 ↪sort_values(ascending=False)
print(retention.head(5))
```

```
manufacturer
ferrari          99000.333333
aston martin     48592.777778
porsche          22913.995106
alfa romeo       15678.100000
datsun           15393.725806
Name: price, dtype: float64
```

[219]:
```
oldest_year = df['year'].min()
oldest_car = df[df['year'] == oldest_year][['manufacturer', 'year', 'price']]
print(f"Oldest year in dataset: {oldest_year}")
print(oldest_car)
```

```
Oldest year in dataset: 1900
       manufacturer  year  price
25817       Unknown  1900      1
27583       Unknown  1900      1
36222         acura  1900  38250
38162       Unknown  1900      1
81419          ford  1900      1
82889          ford  1900      1
83483          ford  1900      1
107561      Unknown  1900   4500
110960      Unknown  1900      1
135534      Unknown  1900     75
238215      Unknown  1900    998
350470        dodge  1900    500
```

[211]:
```
#
pre_2010 = df[df['year'] < 2010]['price'].mean()
post_2010 = df[df['year'] >= 2010]['price'].mean()

print(f"Average price Pre-2010: ${pre_2010:,.2f}")
print(f"Average price Post-2010: ${post_2010:,.2f}")
```

```
Average price Pre-2010: $9,253.28
Average price Post-2010: $23,063.39
```

[ ]: