In [1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
cars = pd.read_csv("../Maven Python Course/car_prices.csv" , on_bad_lines='skip')
```

In [3]:
```
cars.head()
```

Out[3]:

| | year | make | model | trim | body | transmission | vin | state | condition | odometer |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2015 | Kia | Sorento | LX | SUV | automatic | 5xyktca69fg566472 | ca | 5.0 | 16639.0 |
| **1** | 2015 | Kia | Sorento | LX | SUV | automatic | 5xyktca69fg561319 | ca | 5.0 | 9393.0 |
| **2** | 2014 | BMW | 3 Series | 328i SULEV | Sedan | automatic | wba3c1c51ek116351 | ca | 4.5 | 1331.0 |
| **3** | 2015 | Volvo | S60 | T5 | Sedan | automatic | yv1612tb4f1310987 | ca | 4.1 | 14282.0 |
| **4** | 2014 | BMW | 6 Series Gran Coupe | 650i | Sedan | automatic | wba6b2c57ed129731 | ca | 4.3 | 2641.0 |

In [4]:
```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 558811 entries, 0 to 558810
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   year          558811 non-null  int64
 1   make          548510 non-null  object
 2   model         548412 non-null  object
 3   trim          548160 non-null  object
 4   body          545616 non-null  object
 5   transmission  493458 non-null  object
 6   vin           558811 non-null  object
 7   state         558811 non-null  object
 8   condition     547017 non-null  float64
 9   odometer      558717 non-null  float64
 10  color         558062 non-null  object
 11  interior      558062 non-null  object
 12  seller        558811 non-null  object
 13  mmr           558811 non-null  int64
 14  sellingprice  558811 non-null  int64
 15  saledate      558811 non-null  object
dtypes: float64(2), int64(3), object(11)
memory usage: 68.2+ MB
```

In [4]:
```
cars.describe()
```

Out[4]:

| | year | condition | odometer | mmr | sellingprice |
|---|---|---|---|---|---|
| **count** | 558811.000000 | 547017.000000 | 558717.000000 | 558811.000000 | 558811.000000 |
| **mean** | 2010.038696 | 3.424512 | 68323.195797 | 13769.324646 | 13611.262461 |
| **std** | 3.966812 | 0.949439 | 53397.752933 | 9679.874607 | 9749.656919 |
| **min** | 1982.000000 | 1.000000 | 1.000000 | 25.000000 | 1.000000 |
| **25%** | 2007.000000 | 2.700000 | 28374.000000 | 7100.000000 | 6900.000000 |
| **50%** | 2012.000000 | 3.600000 | 52256.000000 | 12250.000000 | 12100.000000 |
| **75%** | 2013.000000 | 4.200000 | 99112.000000 | 18300.000000 | 18200.000000 |
| **max** | 2015.000000 | 5.000000 | 999999.000000 | 182000.000000 | 230000.000000 |

In [ ]:
```
style_list = ["SUV" , "Sedan" , "Convertible" , "Coupe"]

sns.pairplot(
             cars.query("body in @style_list"),
             corner=True,
             hue='body'
)
```

In [ ]:
```
sns.heatmap(
    cars.corr(),
    cmap="RdYlGn",
    annot=True
)
```

In [ ]:
```
brand_list = ['Ford','BMW','Toyota', 'Chevrolet']
```

```
sns.Implot(
    x="odometer",
    y="sellingprice",
    data=cars.query("make in @brand_list and sellingprice <150000 and odometer < 20000
    hue="make",
    scatter.kws=({"alpha": .3}),
)
```

In [ ]:
```
import scipy.stats as stats

cars_regress = cars.dropna()
stats.linregress(x=cars_regress["odometer"] , y=cars_regress["sellingprice"])
```

In [ ]:
```
for brand in brand_list:
    data=cars_regress.loc[cars_regress["make"]==brand]
    print(f"{brand}: {stats.linregress(data['odometer'],data['sellingprice'])[0:2]}")
```

In [ ]:
```
fig, ax = plt.subplots(figsize=(10,10))

bins=np.linspace(1,5,9)
labels= ["1-1.5" , "1.5-2" , "2-2.5" , "2.5-3" , "3-3.5" , "3.5-4" , "4-4.5" , "4.5-5"

cars["condition_bins"] = pd.cut(cars["condition"] , bins=bins , labels=labels )

sns.barplot(
    x="condition_bins",
    y="sellingprice",
    data=cars

)
ax.set_title("Average Price By Condition")
sns.despine()
```

In [ ]:
```
cars["make"].value_counts()
```

In [ ]:
```
top10=cars["make"].value_counts().iloc[:10].index

avgs= ( cars
       .loc[cars["make"].isin(top10)]
       .pivot_table(index="make",
                    columns="condition_bins",
                    values={"sellingprice":"mean"}
                    )
       .droplevel(0,axis=1)
)
avgs.head()
```

In [ ]:
```
fig, ax = plt.subplots(figsize=(10,10))

plt.title("Avg.selling price by make and quality" , fontsize=16)

sns.heatmap(
    avgs,
    annot=True,
    cmap="RdYlGn",
    fmt="g"
)
```

In [ ]:
```
f150s=cars[cars["model"]=="F-150"]

f150s.head()
```

In [ ]:
```
sns.histplot(x="sellingprice",
             bins=10,
             data=f150s)
```

In [ ]:
```
fig, ax =plt.subplots()
sns.barplot(x="trim" , y="sellingprice" , data=f150s)
plt.xticks(rotation=90)
plt.show()
```

In [ ]:
```
f150s_reduced = f150s.query("trim in ['XL','XLT','Platinum','SVT Raptor']")
```

In [ ]:
```
sns.jointplot(
    x="condition",
    y="sellingprice",
    kind="scatter",
    data=f150s_reduced,
    hue="trim"
)
```

In [ ]:
```
sns.set_style("darkgrid")

f150s = f150s.assign(diff_to_mmr = f150s["sellingprice"]-f150s["mmr"])

fig, ax = plt.subplots(figsize=(8,6))

sns.barplot(
    x="color",
    y="diff_to_mmr",
    data=f150s,
    errorbar=None
)
plt.xticks(rotation=45)
plt.show()
```

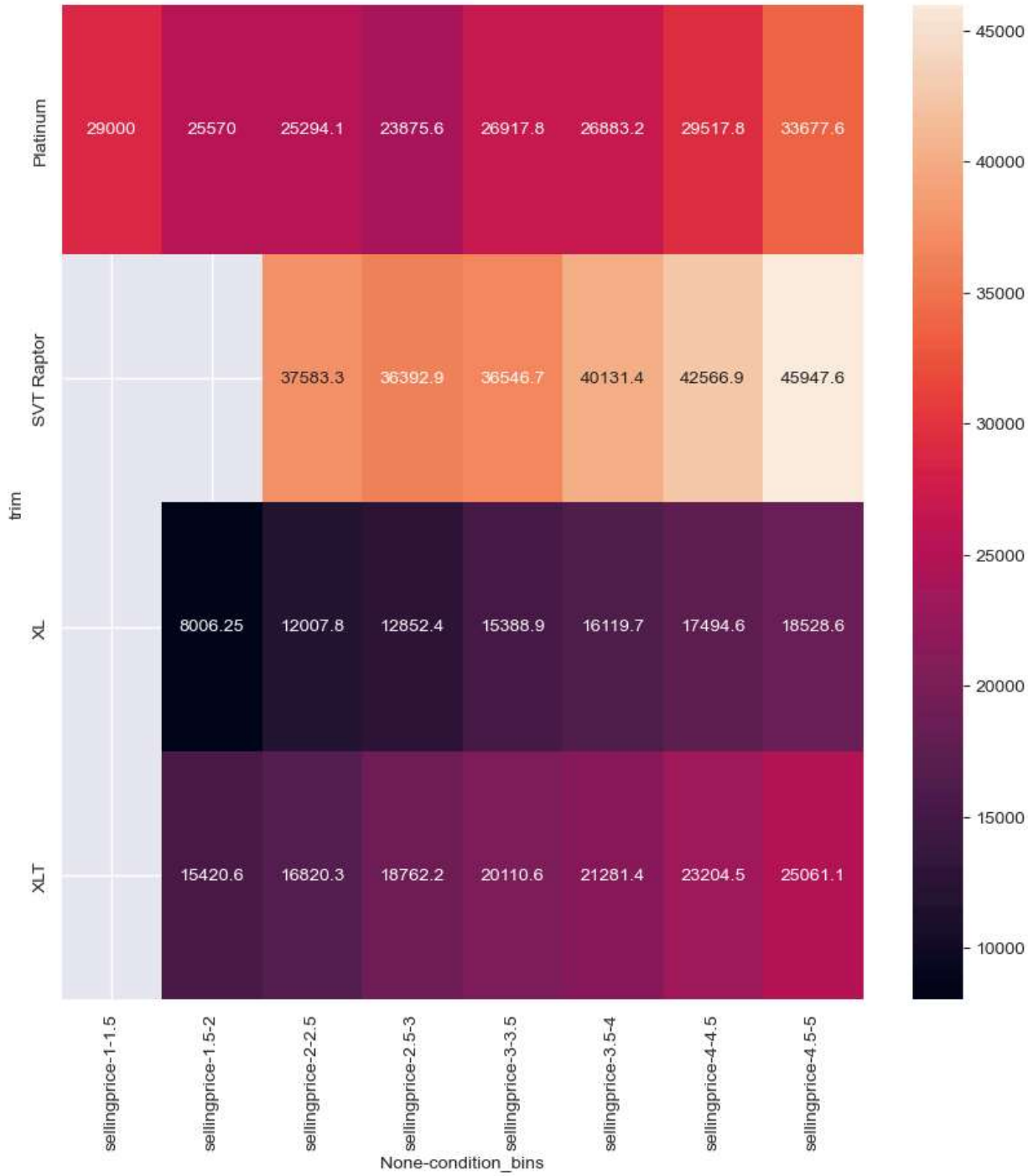
In [ ]:
```
f150s.head()
```

In [ ]:
```
trim_list=['XL' , 'XLT','Platinum','SVT Raptor']

f150s_reduced = f150s.query("color != 'orange' and trim in @trim_list")

f150s_pivot = (
    f150s_reduced
    .loc[f150s_reduced["year"].between(2009 , 2014)]
    .pivot_table(
        index="trim",
        columns="condition_bins",
        values={"sellingprice":"mean"}
    )
)
```

In [50]:
```
fig, ax = plt.subplots(figsize=(10,10))
```

```
sns.heatmap(f150s_pivot,annot=True,fmt="g")
```

Out[50]: <Axes: xlabel='None-condition_bins', ylabel='trim'>

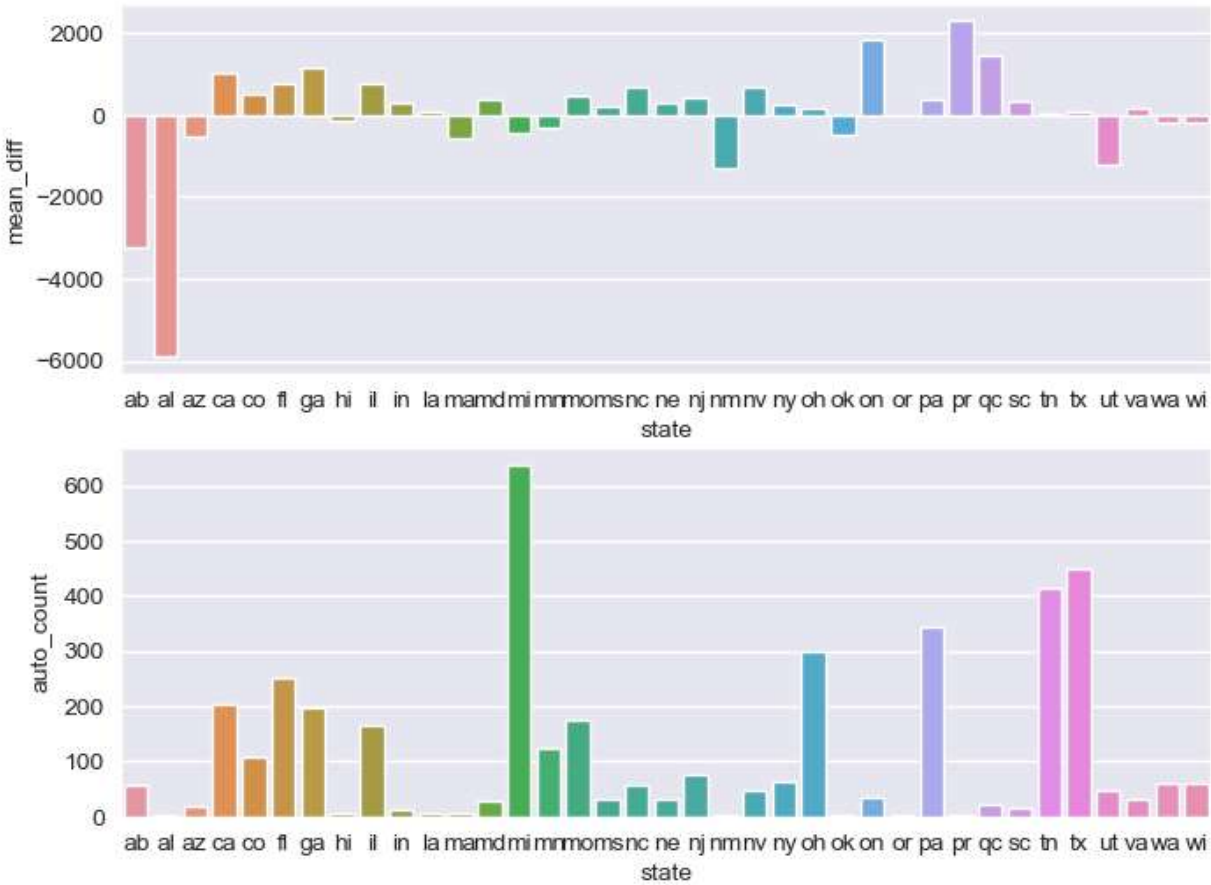



In [49]:

```
f150s_XLT = (f150s_reduced
             .query("trim == 'XLT' and condition >=3.5")
             .groupby(['state'] , as_index=False)
             .agg(
                 mean_diff=("diff_to_mmr" , 'mean'),
                 auto_count=("diff_to_mmr" , "count")
             )
)
fig, ax = plt.subplots(2,figsize=(8,6))

sns.barplot(
```

```
    x="state",
    y="mean_diff",
    data=f150s_XLT,
    ax=ax[0]
)

sns.barplot(
    x="state",
    y="auto_count",
    data=f150s_XLT,
    ax=ax[1]
)
```

Out[49]: <Axes: xlabel='state', ylabel='auto_count'>




In [51]:
```
f150s.query("trim=='XLT' and state=='ut' and condition >3.5").groupby("saledate").agg(
```

Out[51]:

| | diff_to_mmr | |
|---|---|---|
| | mean | count |
| saledate | | |
| Wed Feb 04 2015 03:30:00 GMT-0800 (PST) | -1400.000000 | 2 |
| Wed Feb 11 2015 03:30:00 GMT-0800 (PST) | -1605.000000 | 10 |
| Wed Feb 18 2015 03:30:00 GMT-0800 (PST) | -542.857143 | 7 |
| Wed Feb 25 2015 03:30:00 GMT-0800 (PST) | -950.000000 | 4 |
| Wed Jan 07 2015 11:30:00 GMT-0800 (PST) | -2325.000000 | 2 |
| Wed Jan 21 2015 03:30:00 GMT-0800 (PST) | -900.000000 | 2 |
| Wed Jun 17 2015 04:30:00 GMT-0700 (PDT) | -2120.000000 | 5 |
| Wed Mar 04 2015 03:30:00 GMT-0800 (PST) | -600.000000 | 5 |

In [ ]: