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

Car Price Analysis

- Read in the dataset `car_prices.csv`.
- Run the info and describe DataFrame methods to get a quick understanding of the data.

In [2]:
```
cars = pd.read_csv("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 [5]:
```
cars.describe()
```

Out[5]:

	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

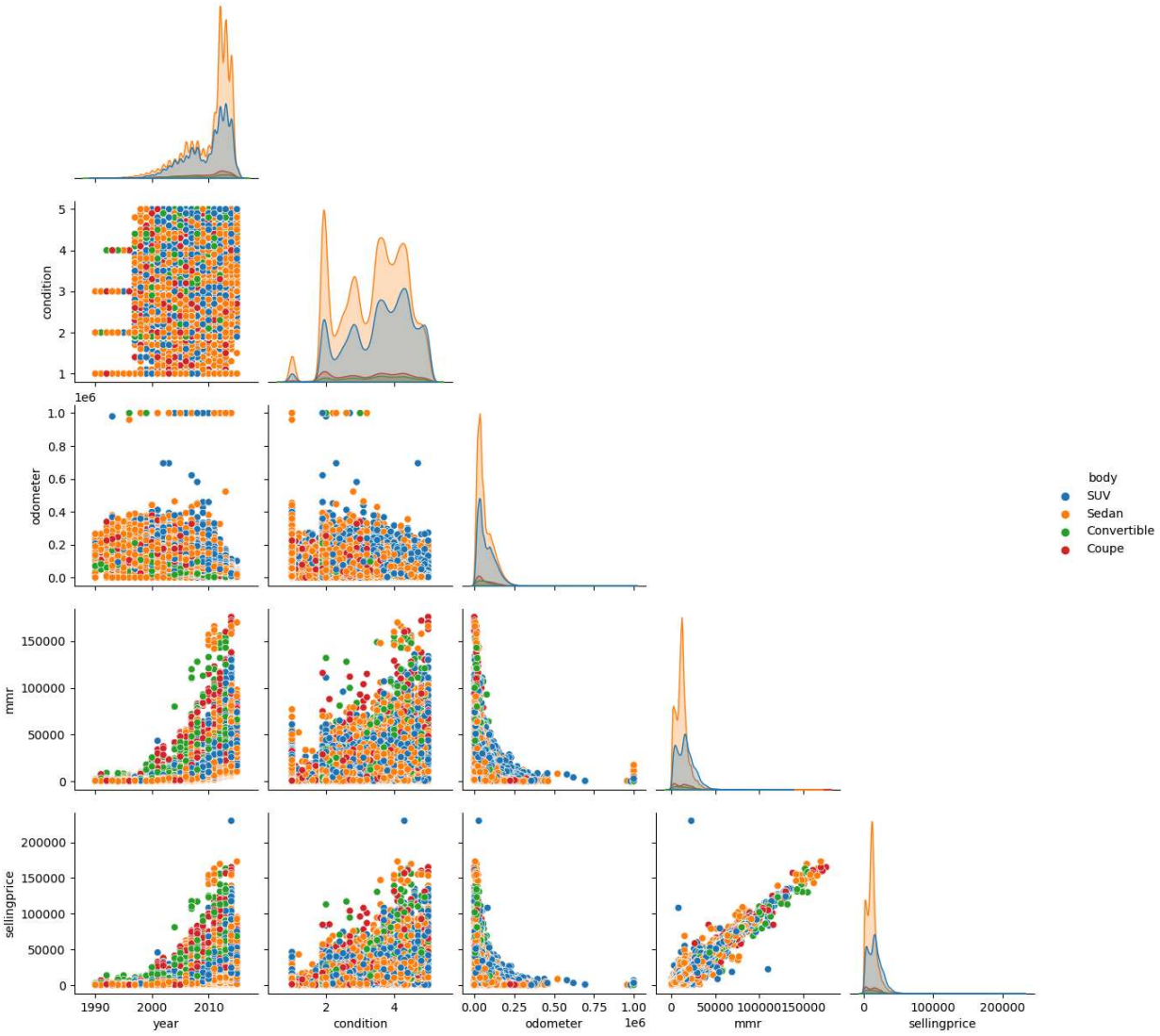
Numeric Variable Relationships

Use a pairplot to get a good sense for the relationships between the numeric variables in the dataset.

- Why do you think the relationship between `sellingprice` and `mmr` (the recommended selling price) is so strong?
- What is the relationship between `sellingprice` and `year` (the year the car was manufactured). Does this make sense?
- Once you've looked at the two relationships above, filter your dataframe down to the `body` styles ['SUV', 'Sedan', 'Convertible', and 'Coupe']. Color the scatterplot based on values in the `body` column.

In [6]:
```
style_list = ["SUV", "Sedan", "Convertible", "Coupe"]

sns.pairplot(data = cars.loc[cars["body"].isin(style_list)], hue = "body", corner = Tr
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x1a9087ebb20>
```

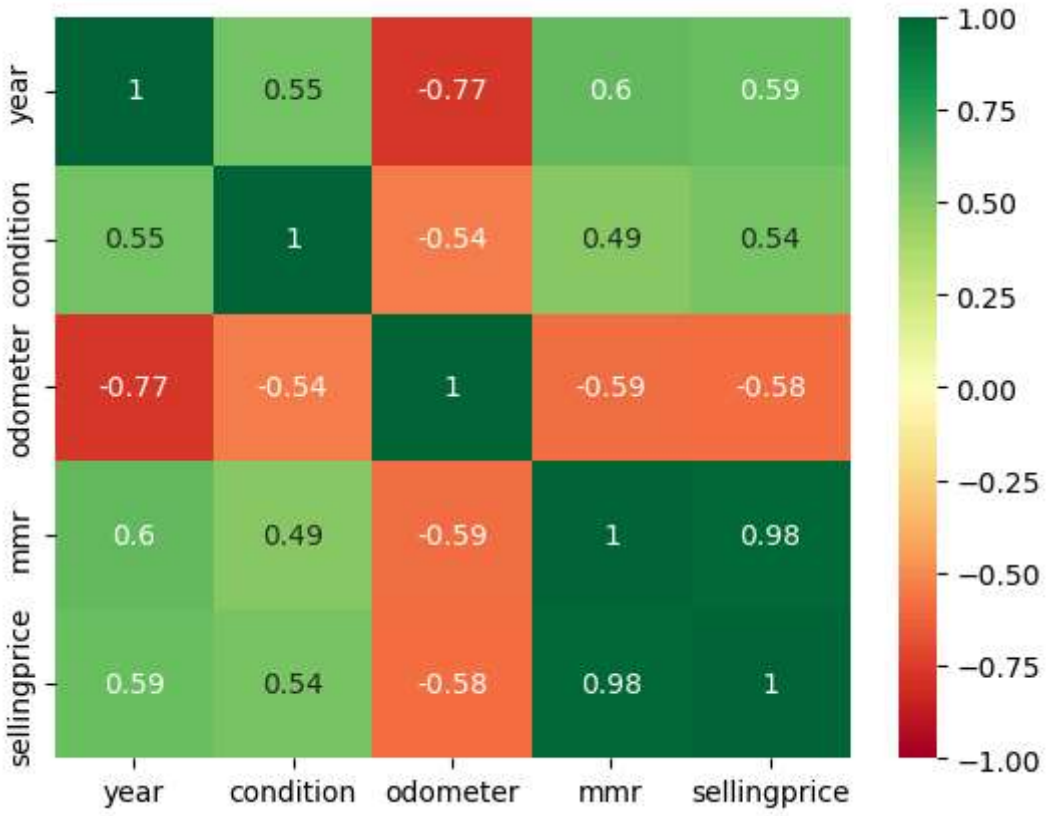


Correlation Heat Map

- Plot a heatmap of the correlation between the numeric variables. Does it make sense year and condition and odomoter are negatively correlated?

In [7]:
```
sns.heatmap(cars.corr(numeric_only = True), annot = True, vmin = -1, vmax = 1, cmap =
```

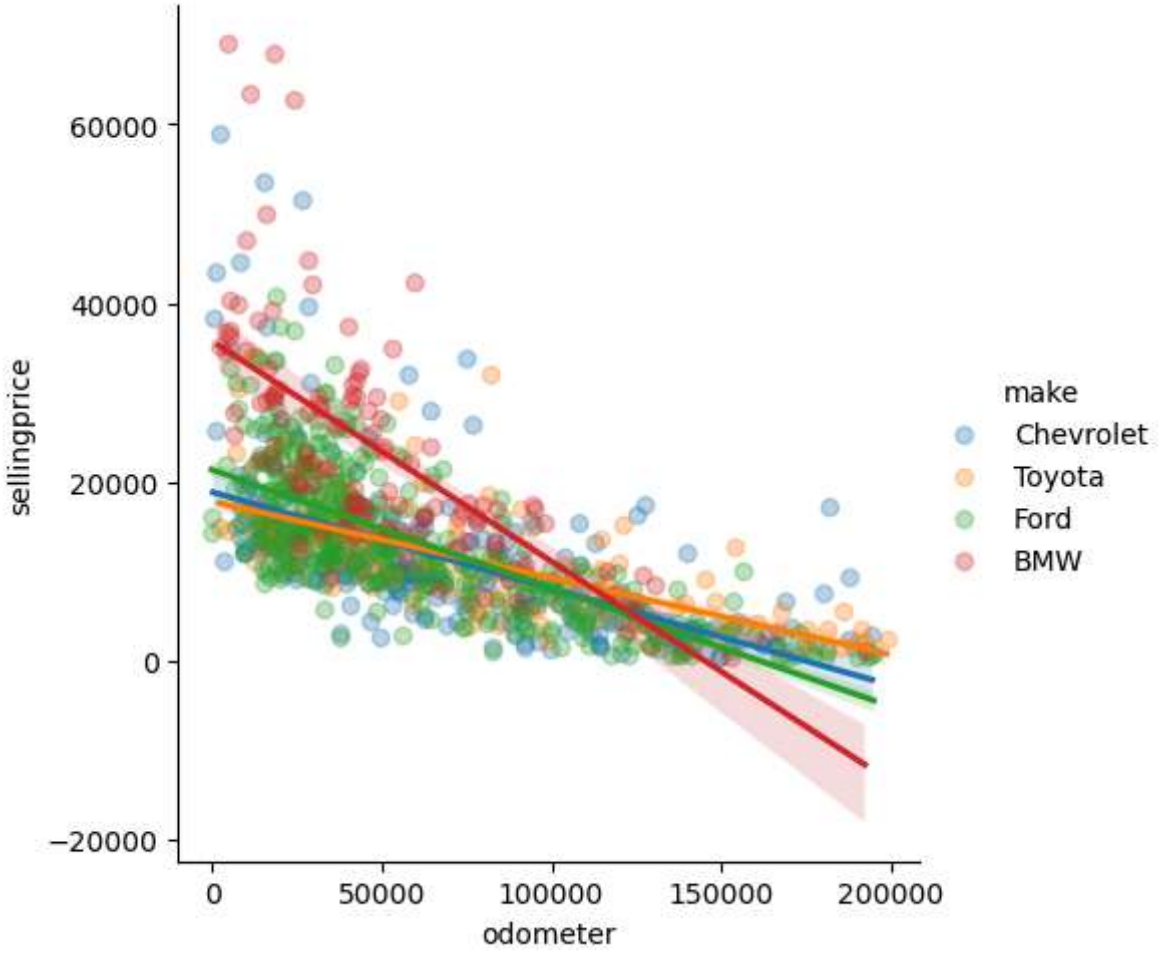
Out[7]:
```
<Axes: >
```



Value Retention

- Filter your data down to the following values of `make` (brand): ['Ford', 'BMW', 'Toyota', 'Chevy']
- Build a linear relationship plot comparing `odometer` (distance car has travelled) vs. `sellingprice`, broken out by `make`.
- If you want to fit a regression with scipy.stats, use dropna() on the dataframe first.

In [8]:
```
brand_list = ['Ford', 'BMW', 'Toyota', 'Chevrolet']

sns.lmplot(
    x = "odometer",
    y = "sellingprice",
    hue = "make",
    data = (cars
            .loc[(cars["make"].isin(brand_list))
                 & (cars["body"].isin(style_list))
                 & (cars["odometer"] <= 200000)]
            .sample(1000)
           ),
    scatter_kws = {"alpha": 0.3}
)

plt.show()
```

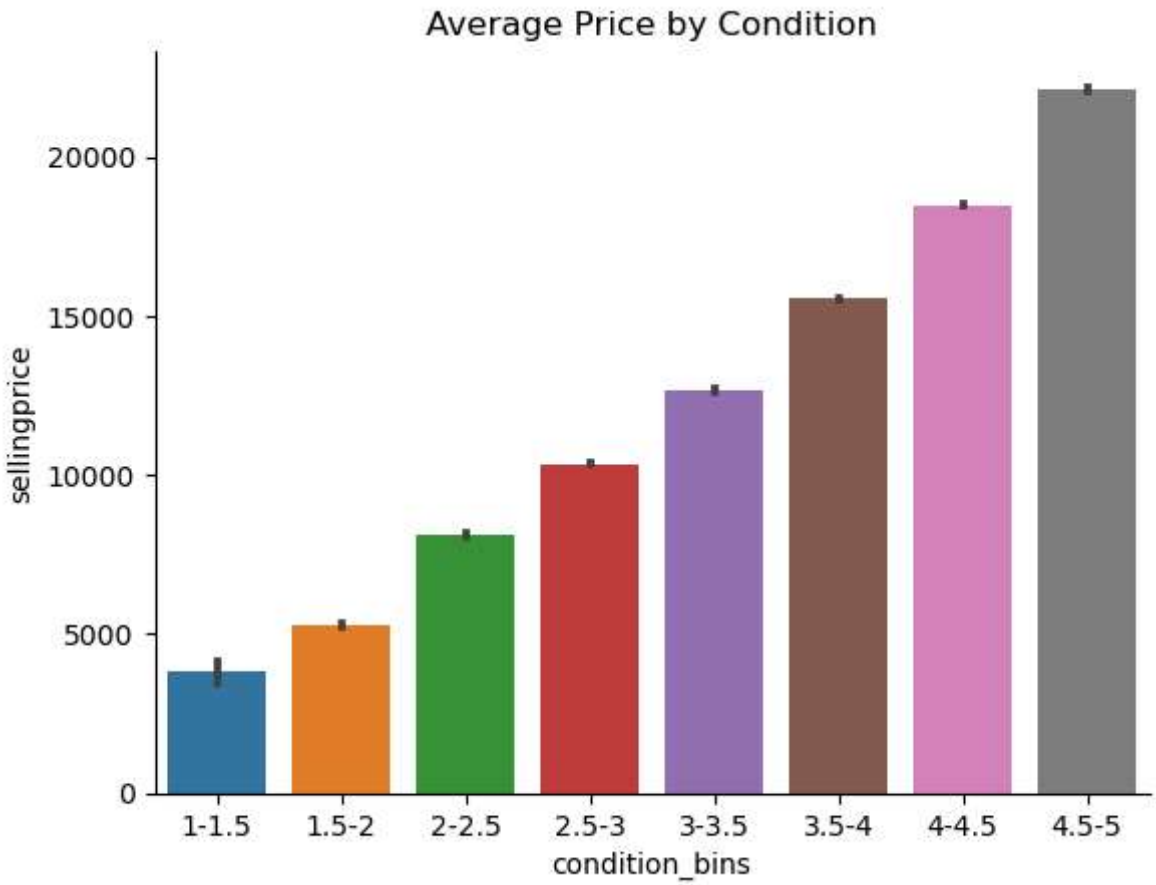


Categorical Variable Relationships

- Bin the `condition` variable into increments of .5. For example, the first bin will be '1-1.5', the second '1.5-2', and so on up until the maximum condition value of 5. Use pd.cut() to create the bins.
- Build a barplot of the average price by condition bin.

In [9]:
```
cars["condition_bins"] = pd.cut(
    x = cars["condition"],
    bins = [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5],
    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"]
)
```

In [10]:
```
sns.barplot(x = "condition_bins", y = "sellingprice", data = cars)
sns.despine()
plt.title("Average Price by Condition")
plt.show()
```



Price by make and condition

- Create a pivot table with the top 10 most common car brands `make` column as rows, and condition bins as columns. The values in the table should be the average selling price for each make/condition combination. Consider using value_counts to get the top 10 most common brands.
- Once you've created the table, build a heatmap from the data. Which brands and quality levels have the highest selling prices? Does this match your expecations?

In [11]:
```
top10 = cars["make"].value_counts().iloc[:10].index
```

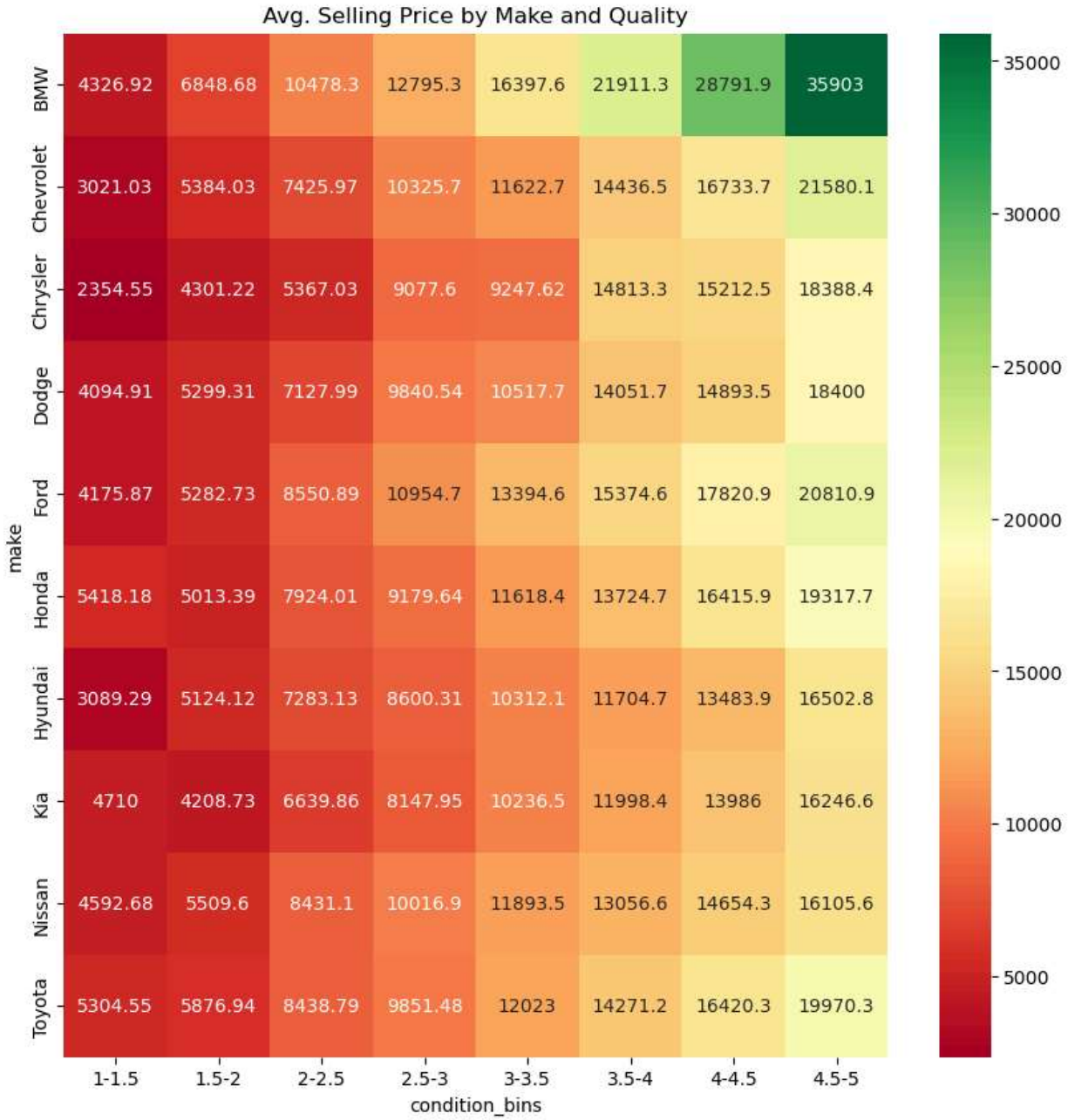
In [12]:
```
top10_pivot = (cars
               .loc[cars["make"].isin(top10)]
               .pivot_table(
                   index = "make",
                   columns = "condition_bins",
                   values = "sellingprice",
                   aggfunc = "mean"
               )
              )
top10_pivot.head()
```

Out[12]:

condition_bins	1-1.5	1.5-2	2-2.5	2.5-3	3-3.5	3.5-4	
make							
BMW	4326.923077	6848.680865	10478.320029	12795.277669	16397.610281	21911.286118	287
Chevrolet	3021.031746	5384.030601	7425.967945	10325.668781	11622.675195	14436.514288	167
Chrysler	2354.545455	4301.224656	5367.031437	9077.602013	9247.621069	14813.315207	152
Dodge	4094.907407	5299.306603	7127.988641	9840.543230	10517.710952	14051.693213	148
Ford	4175.866667	5282.734146	8550.886379	10954.656807	13394.647490	15374.553012	178

In [13]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(top10_pivot, annot = True, fmt = "g", cmap = "RdYlGn")
plt.title("Avg. Selling Price by Make and Quality")
```

Out[13]:
```
Text(0.5, 1.0, 'Avg. Selling Price by Make and Quality')
```



Ford F-150 Deep Dive

Our client is mostly interested in purchasing work trucks - specifically the Ford F-150, the highest selling automobile for decades in the US.

- Filter your data down to rows where `model` is "F-150".
- Then build a histogram of sellingprice with 10 bins.

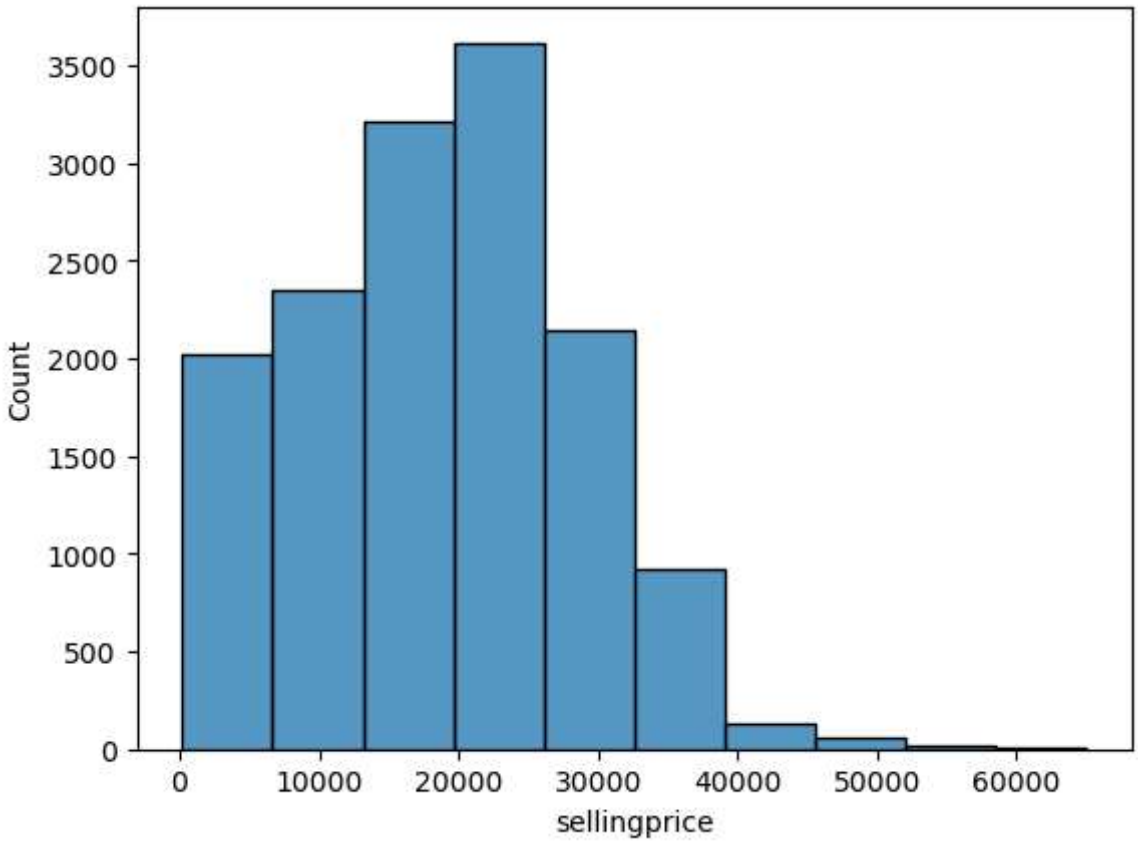
In [14]:
```
f150s = cars[cars["model"] == "F-150"].copy()

f150s.head()
```

Out[14]:

	year	make	model	trim	body	transmission	vin	state	condition	odor
983	2012	Ford	F-150	XLT	SuperCrew	NaN	1ftew1cm9ckd05952	ca	4.6	51
1013	2012	Ford	F-150	FX2	SuperCrew	automatic	1ftfw1ct0cfb64807	ca	4.6	28
1052	2012	Ford	F-150	XLT	SuperCrew	automatic	1ftfw1et3ckd61619	ca	3.9	27
1054	2012	Ford	F-150	XLT	SuperCrew	automatic	1ftfw1ef9cfc79834	ca	3.5	93
1074	2012	Ford	F-150	Lariat	SuperCab	automatic	1ftfx1ef6cfc80260	ca	4.1	46

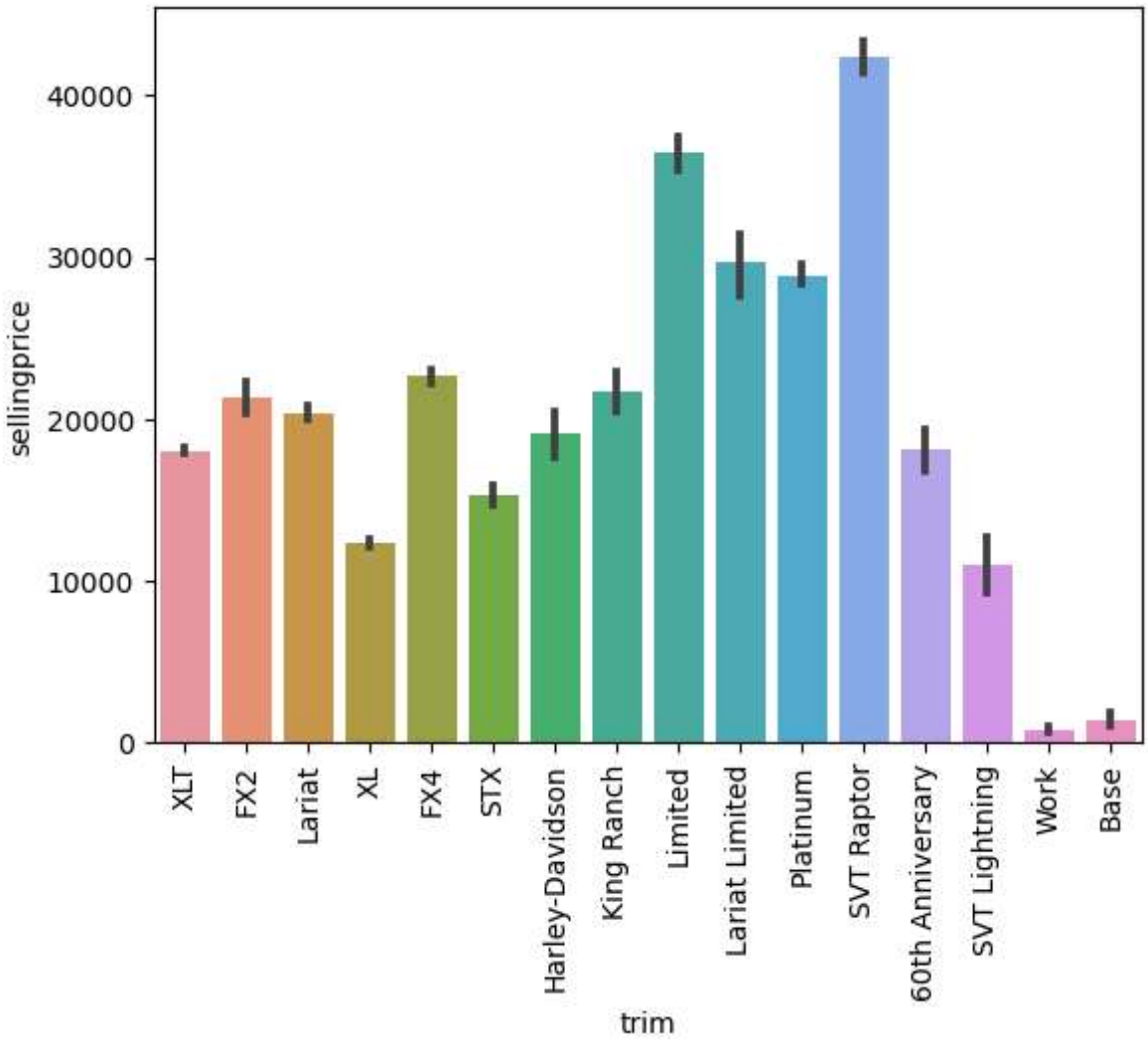
In [15]:
```
sns.histplot(f150s["sellingprice"], bins = 10)
```

Out[15]:
```
<Axes: xlabel='sellingprice', ylabel='Count'>
```



- Build a barplot of `sellingprice` by `trim` for the F150s.

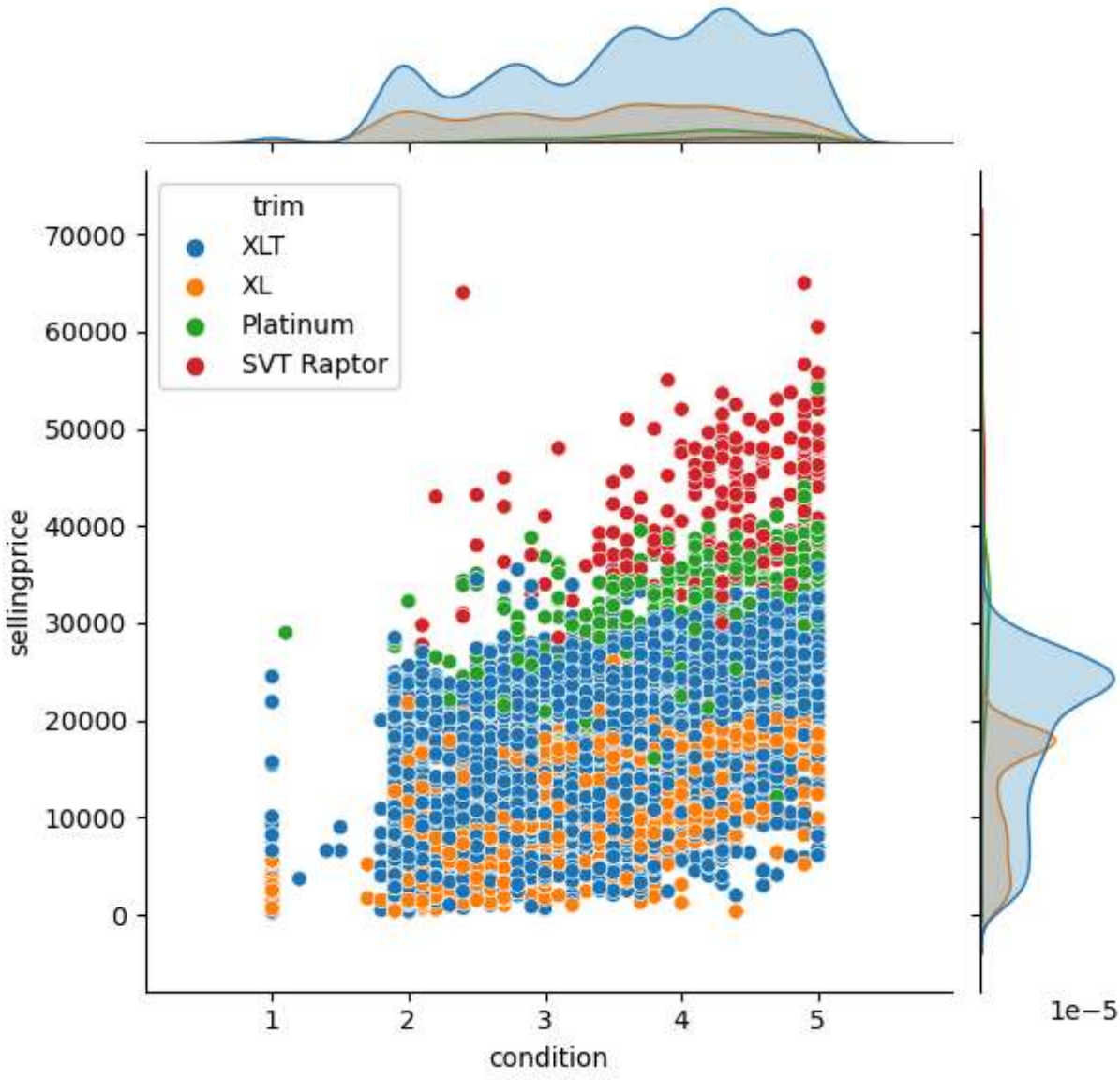
In [16]:
```
sns.barplot(
    x = "trim",
    y = "sellingprice",
    data = f150s
)
plt.xticks(rotation = 90)
plt.show()
```



- Filter the dataset down to the trim levels ["XL", "XLT", "Platinum", "SVT Raptor"].
- Then, build a jointplot of `condition` (not binned) by `sellingprice`.

In [17]:
```
trim_list = ['XL', 'XLT', 'Platinum', 'SVT Raptor']

sns.jointplot(
    x = "condition",
    y = "sellingprice",
    hue = "trim",
    data = f150s.loc[f150s["trim"].isin(trim_list)]
)
```

Out[17]:
```
<seaborn.axisgrid.JointGrid at 0x1a9165bb970>
```

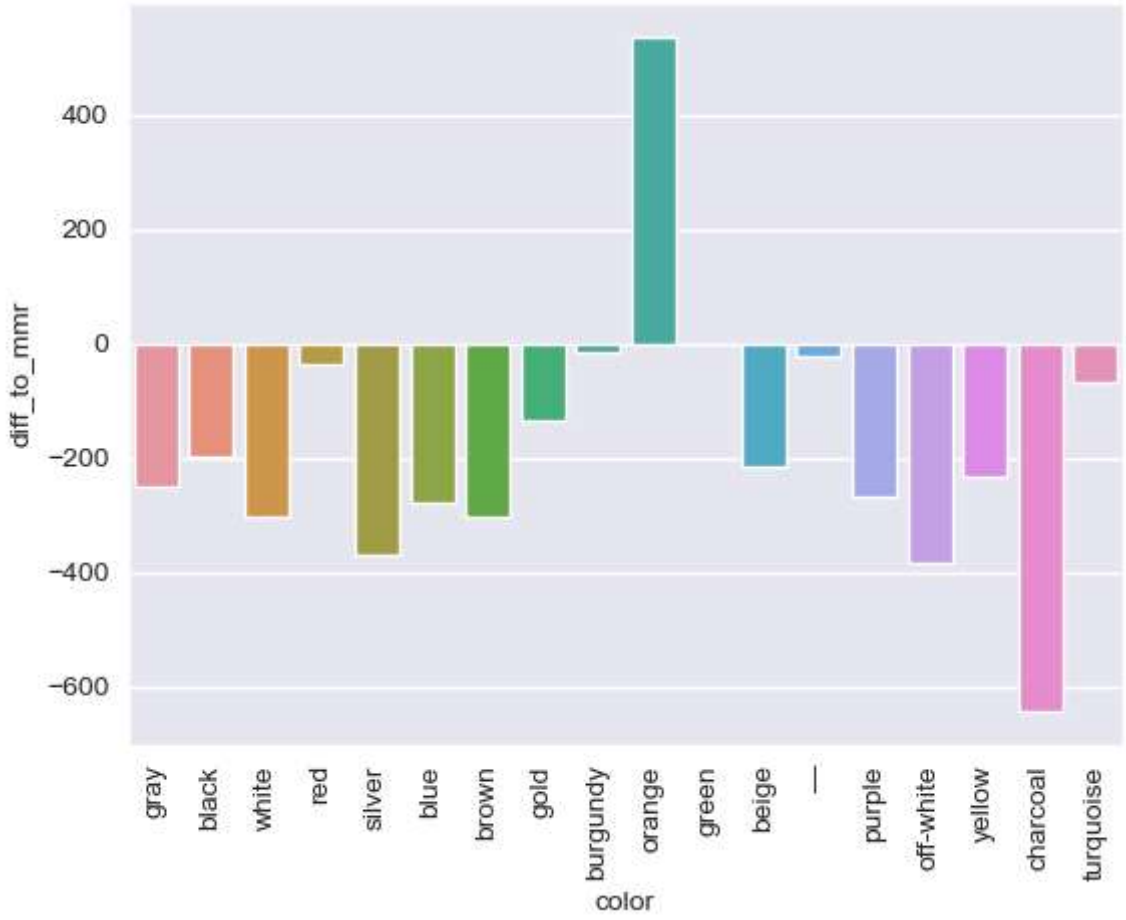


Are there any "Deals"?

Ok, thanks to the charts above we're getting closer to honing in on the trucks to purchase.

- Create a column `diff_to_mmr` which is the difference between `sellingprice` and `mmr`.
- Then build a barplot looking at mean `diff_to_mmr` by `color`.
- Which color sells for more than the recommended price?

In [18]:
```
sns.set_style("darkgrid")

f150s["diff_to_mmr"] = f150s["sellingprice"] - f150s["mmr"]

sns.barplot(x = "color", y = "diff_to_mmr", data = f150s, errorbar = None)

plt.xticks(rotation = 90)
plt.show()
```



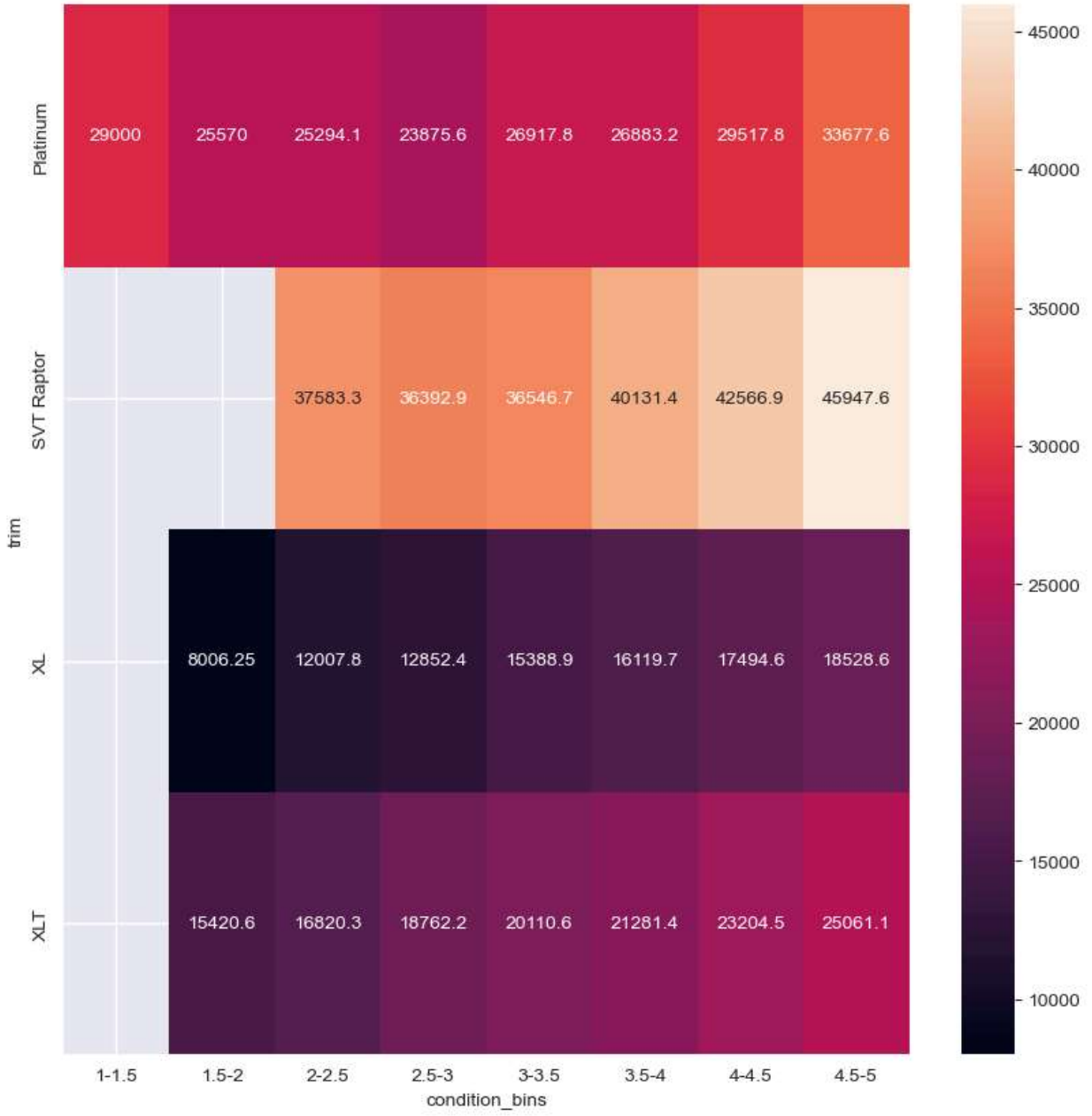
- Further filter the data to the trims below. Exclude the color 'orange' and look only at trucks from years between 2009 and 2014.

In [19]:
```
f150s_pivot = (f150s
               .loc[(f150s["trim"].isin(trim_list))
                    & (f150s["color"] != "orange")
                    & (f150s["year"] >= 2009)
                    & (f150s["year"] <= 2014)]
               .pivot_table(
                   index = "trim",
                   columns = "condition_bins",
                   values = "sellingprice",
                   aggfunc = "mean"
               )
              )
f150s_pivot.round(2)
```

Out[19]:

condition_bins	1-1.5	1.5-2	2-2.5	2.5-3	3-3.5	3.5-4	4-4.5	4.5-5
trim								
Platinum	29000.0	25570.00	25294.12	23875.56	26917.80	26883.15	29517.81	33677.63
SVT Raptor	NaN	NaN	37583.33	36392.86	36546.67	40131.43	42566.95	45947.62
XL	NaN	8006.25	12007.77	12852.36	15388.94	16119.67	17494.63	18528.62
XLT	NaN	15420.61	16820.26	18762.22	20110.63	21281.36	23204.48	25061.06

In [20]:
```
plt.figure(figsize = (10, 10))

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

plt.show()
```



Best state to buy trucks in?

- Ok, we've decided on the XLT model, it has more features than XL but is still quite affordable. Also filter out any trucks with a quality of less than 3.5. Your DataFrame should only have trucks with a quality of 3.5 or greater.
- Build a subplot with a bar chart of `state` by `diff_to_mmr` and `state` by `count` to find which states sell XLT models below mmr and have ample quantity.

In [21]:
```
selected = f150s.loc[(f150s["trim"] == "XLT")
                     & (f150s["condition"] >= 3.5)
```

```
                    & (f150s["year"] >= 2009)
                    & (f150s["year"] <= 2014)]

selected.head()
```

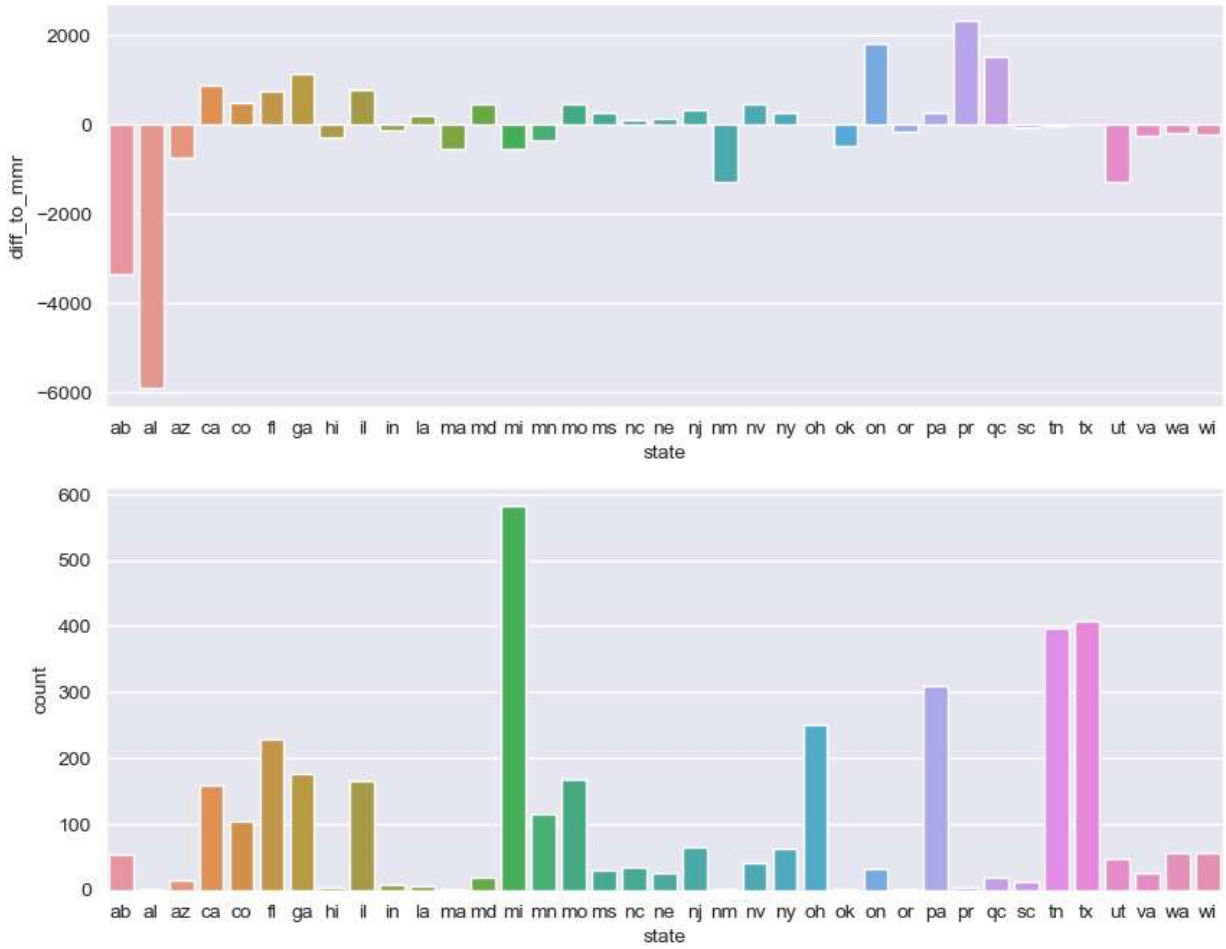
Out[21]:

	year	make	model	trim	body	transmission	vin	state	condition	odom
983	2012	Ford	F-150	XLT	SuperCrew	NaN	1ftew1cm9ckd05952	ca	4.6	511
1052	2012	Ford	F-150	XLT	SuperCrew	automatic	1ftfw1et3ckd61619	ca	3.9	274
1054	2012	Ford	F-150	XLT	SuperCrew	automatic	1ftfw1ef9cfc79834	ca	3.5	938
2079	2011	Ford	F-150	XLT	SuperCab	automatic	1ftex1cm3bfd14408	ca	4.5	286
2107	2011	Ford	F-150	XLT	SuperCrew	automatic	1ftfw1ef6bfa44290	ca	4.1	719

In [22]:

```
import matplotlib.gridspec as gridspec

fig = plt.figure(figsize = (10, 8))

gs = gridspec.GridSpec(nrows = 2, ncols = 1)

ax1 = fig.add_subplot(gs[0])
sns.barplot(
    x = "state",
    y = "diff_to_mmr",
    data = selected.sort_values("state"),
    errorbar = None,
    ax = ax1
)

ax2 = fig.add_subplot(gs[1])
```

```
sns.countplot(
    x = "state",
    data = selected.sort_values("state"),
    ax = ax2
)

plt.show()
```



- Ok, looks like Utah (our client has an office nearby) has a good amount of quantity and a low average price. Take a look at the average diff_to_mmr and count of cars by saledate - feel free to return just a table.

In [23]:
```
(selected
 .loc[selected["state"] == "ut"]
 .groupby("saledate")
 .agg({"diff_to_mmr":["mean", "count"]})
)
```

Out[23]:

	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)	-1787.500000	12
Wed Feb 18 2015 03:30:00 GMT-0800 (PST)	-377.777778	9
Wed Feb 25 2015 03:30:00 GMT-0800 (PST)	-500.000000	6
Wed Jan 07 2015 11:30:00 GMT-0800 (PST)	-2325.000000	2
Wed Jan 14 2015 03:30:00 GMT-0800 (PST)	-1900.000000	1
Wed Jan 21 2015 03:30:00 GMT-0800 (PST)	-900.000000	2
Wed Jun 03 2015 04:30:00 GMT-0700 (PDT)	-1625.000000	2
Wed Jun 17 2015 04:30:00 GMT-0700 (PDT)	-1957.142857	7
Wed Mar 04 2015 03:30:00 GMT-0800 (PST)	-1237.500000	4