Instructor 2 - Seaborn 2022

June 20, 2022

[66]:
```
import numpy as py
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
# %matplotlib is a magic function in IPython.

df = sns.load_dataset("diamonds")
diamonds = sns.load_dataset("diamonds")
tips = sns.load_dataset("tips")
penguins = sns.load_dataset("penguins")
flights = sns.load_dataset("flights")

df.shape
```

[66]:
```
(53940, 10)
```

[2]:
```
# sns.get_dataset_names()
```

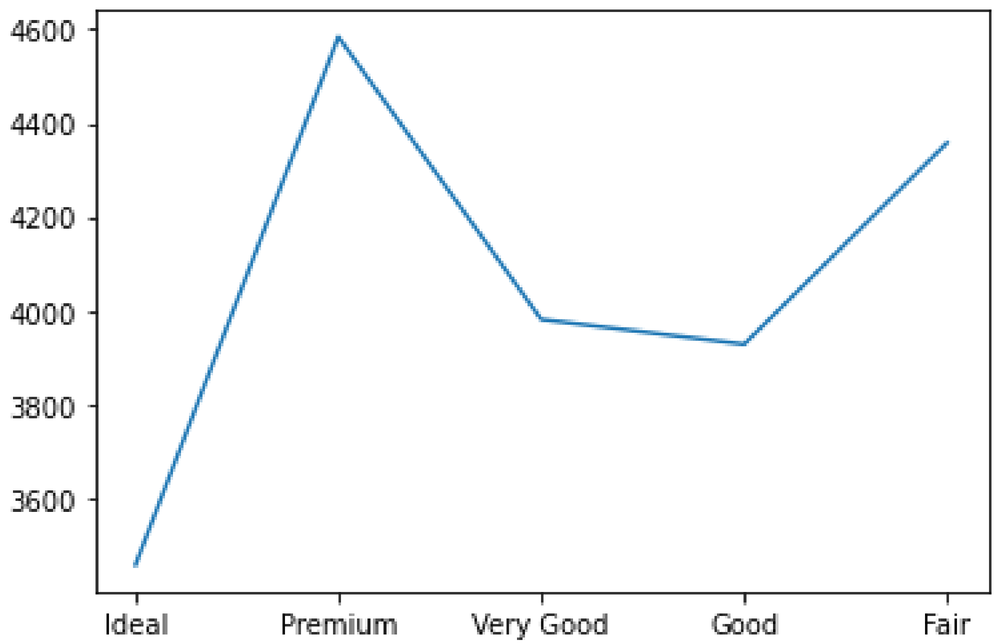
1 Matplotlib to Pandas to Seaborn

[3]:
```
# NOTICE: One y value per X value

x = ['Ideal','Premium','Very Good','Good','Fair',]
y = [3457,4584,3981,3928,4358]
```

[4]:
```
type(x)
```

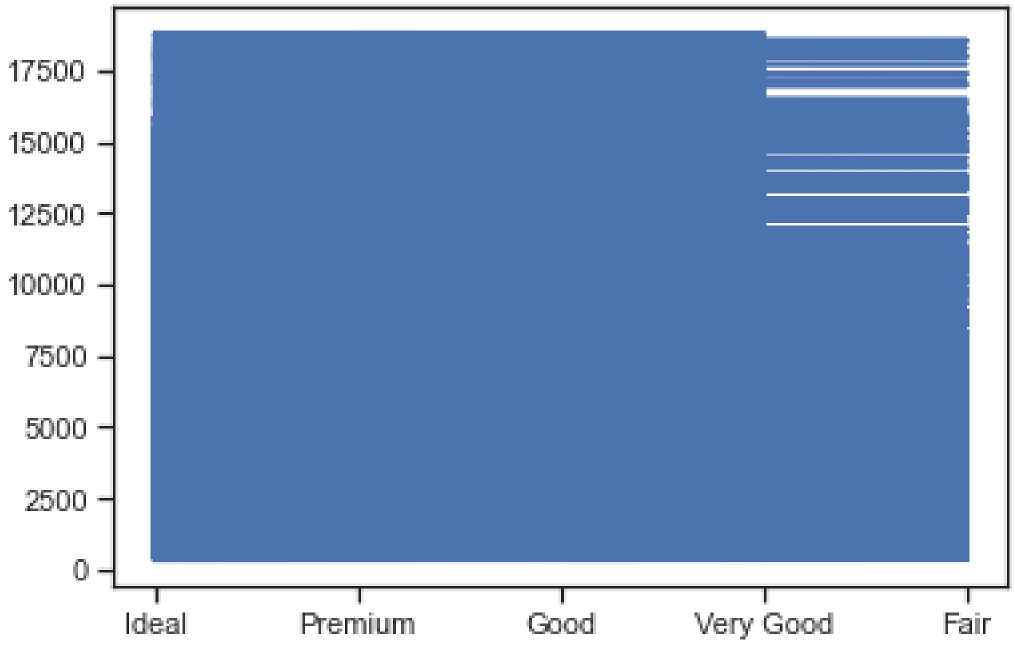
[4]:
```
list
```

[5]:
```
# Matplotlib
plt.plot(x, y);
```



[64]:
```
# This works but data is not aggregated
plt.plot(df.cut, df.price)
```

[64]:
```
[<matplotlib.lines.Line2D at 0x7f77c979fdc0>]
```

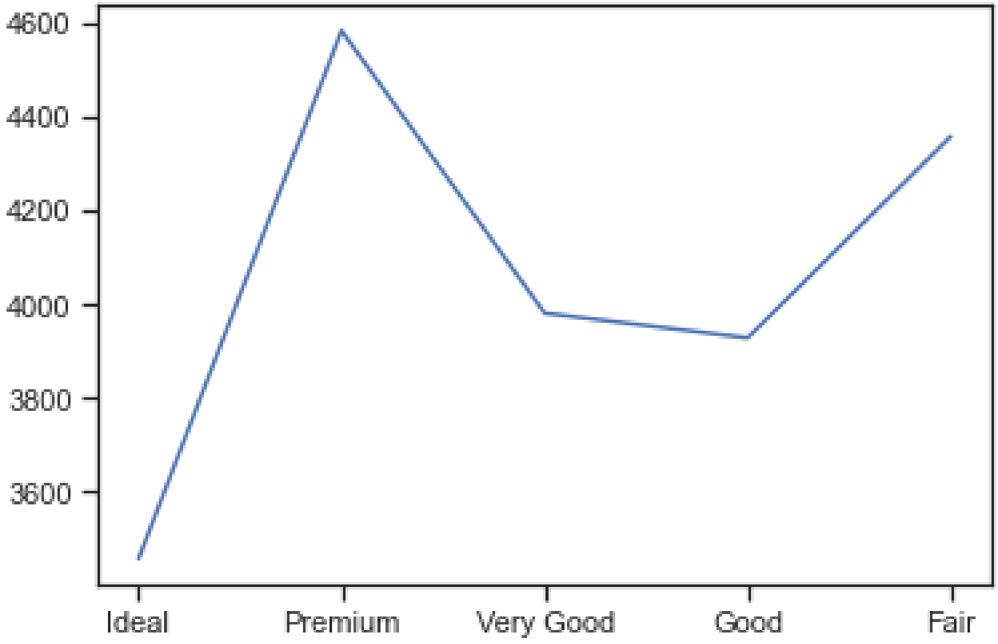


[65]:
```
df2 = df.groupby('cut').mean().reset_index() # creating a smaller dataset
df2
```

[65]:
```
         cut     carat      depth      table        price         x         y  \
0      Ideal  0.702837  61.709401  55.951668  3457.541970  5.507451  5.520080
1    Premium  0.891955  61.264673  58.746095  4584.257704  5.973887  5.944879
2  Very Good  0.806381  61.818275  57.956150  3981.759891  5.740696  5.770026
3       Good  0.849185  62.365879  58.694639  3928.864452  5.838785  5.850744
4       Fair  1.046137  64.041677  59.053789  4358.757764  6.246894  6.182652

          z
0  3.401448
1  3.647124
2  3.559801
3  3.639507
4  3.982770
```

[69]:
```
plt.plot(df2.cut, df2.price)
```

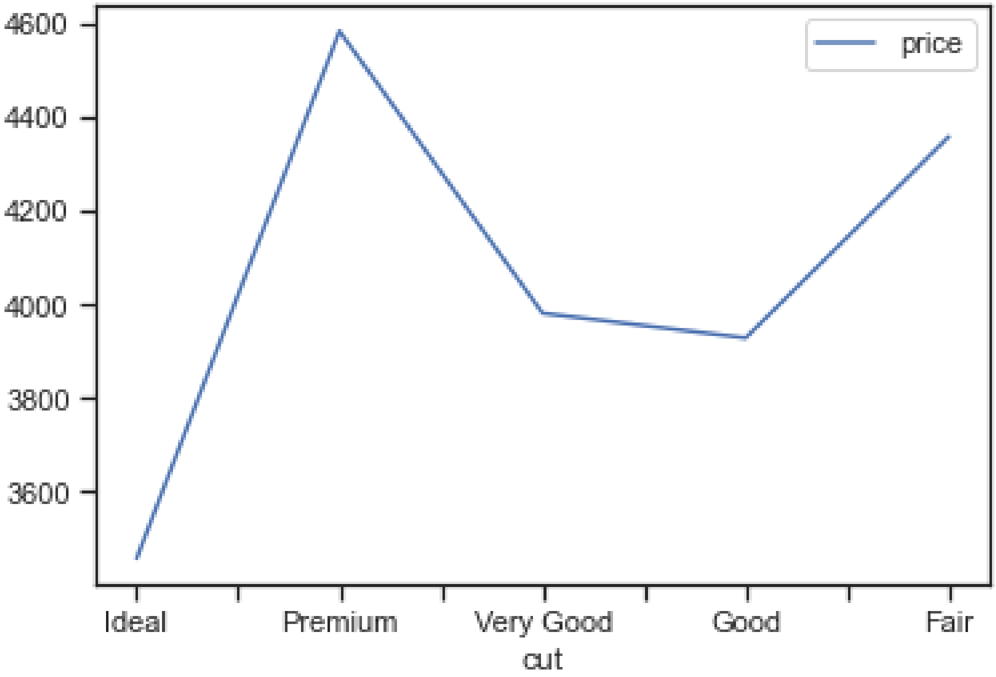
[69]:
```
[<matplotlib.lines.Line2D at 0x7f782783e730>]
```



Below … **pandas** ploting functions (e.g., .plot) uses matplotlib

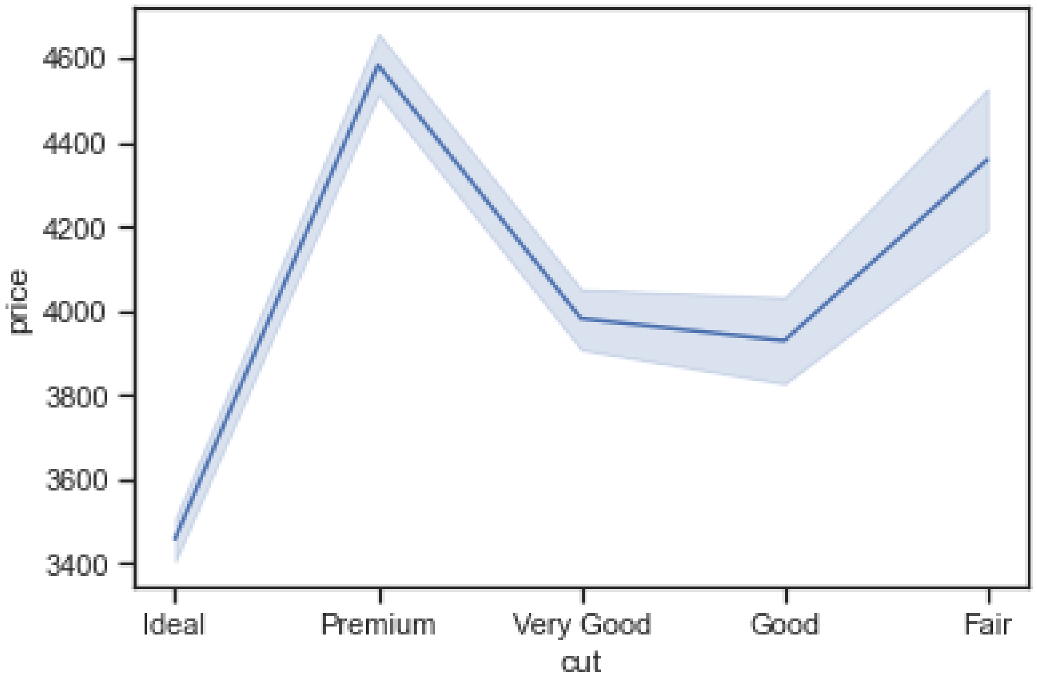
[72]:
```
# This is a pandas plot of the raw data (df)
#df.plot(x='cut', y = 'price')
df2.plot(x='cut', y = 'price')
```

```
[72]: <AxesSubplot:xlabel='cut'>
```



```
[70]: # This is a seaborn plot of df (not grouped) - automatic aggregation
sns.lineplot(data=df, x="cut", y="price")
```

```
[70]: <AxesSubplot:xlabel='cut', ylabel='price'>
```

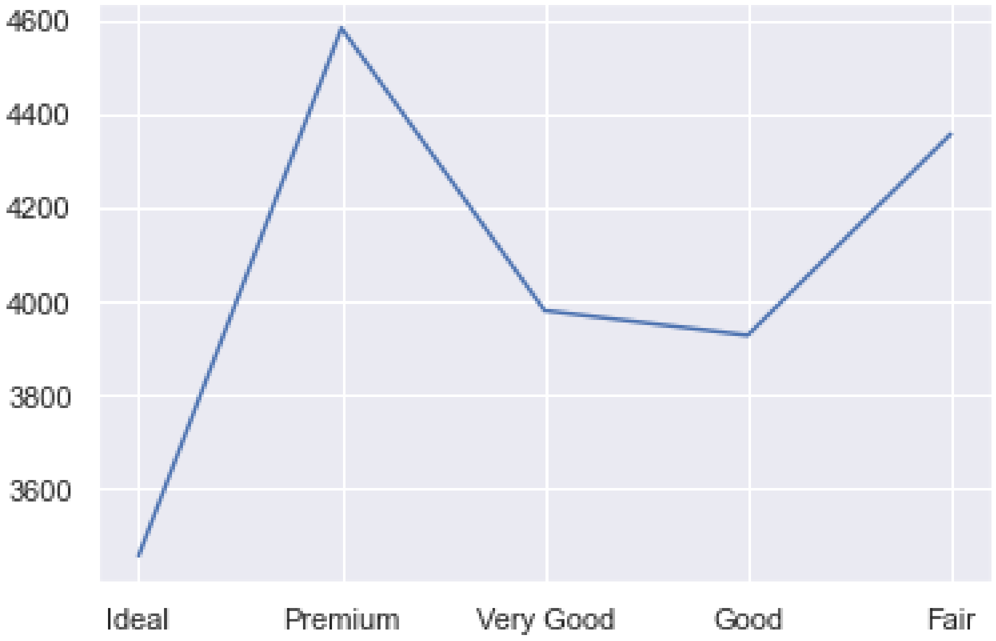


[10]:

```
# Using seaborn's set_theme method
# Seaborn is updating Matplotlib's rc parameters.  rc params are the default style settings
# rc = runtime configuration.
# https://matplotlib.org/stable/tutorials/introductory/customizing.html

sns.set_theme()
```

[11]:

```
# Matplotlib
plt.plot(x, y);
```



2 Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and **statistical aggregation** to produce informative plots.

The **declarative** API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

2.1 Figure & Axes Level Plotting Functions

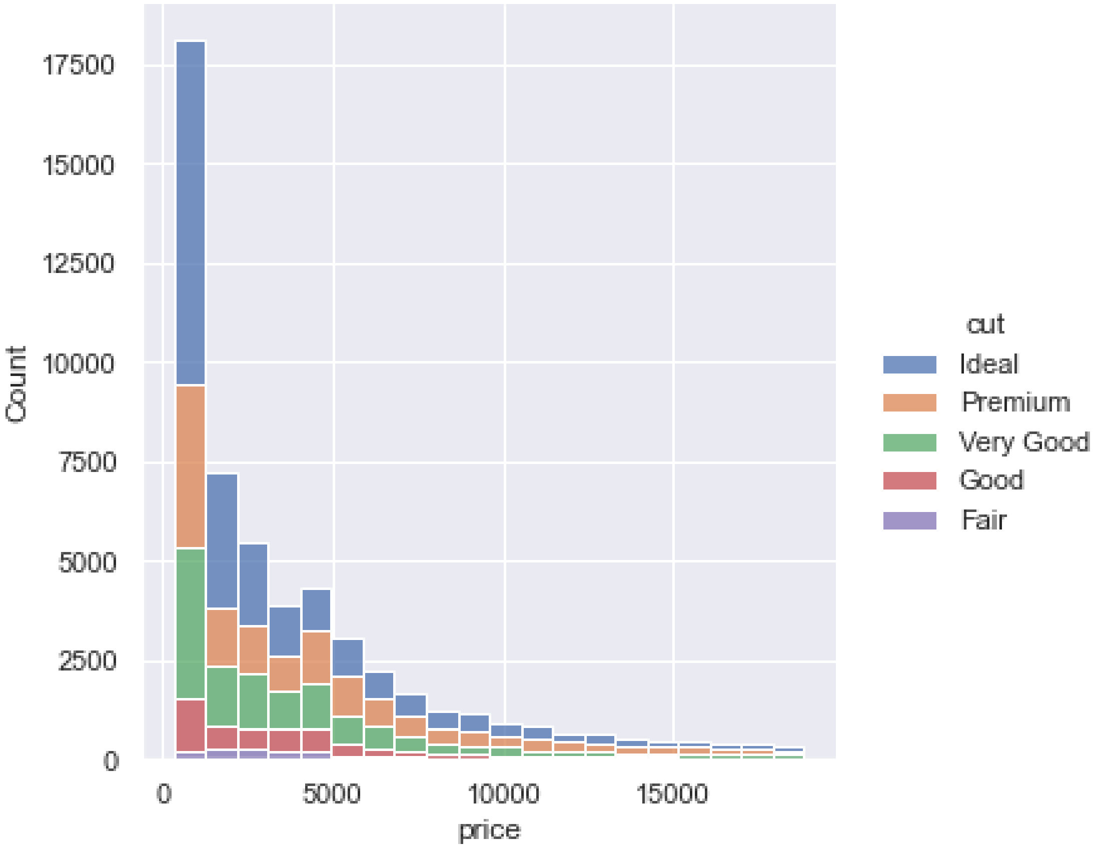
http://seaborn.pydata.org/tutorial.html

2.2 What's the difference between figure and axes level options?

Axes-level functions make self-contained plots The axes-level functions are written to act like drop-in replacements for matplotlib functions. While they add axis labels and legends automatically, they don't modify anything beyond the axes that they are drawn into. That means they can be composed into arbitrarily-complex matplotlib figures with predictable results.

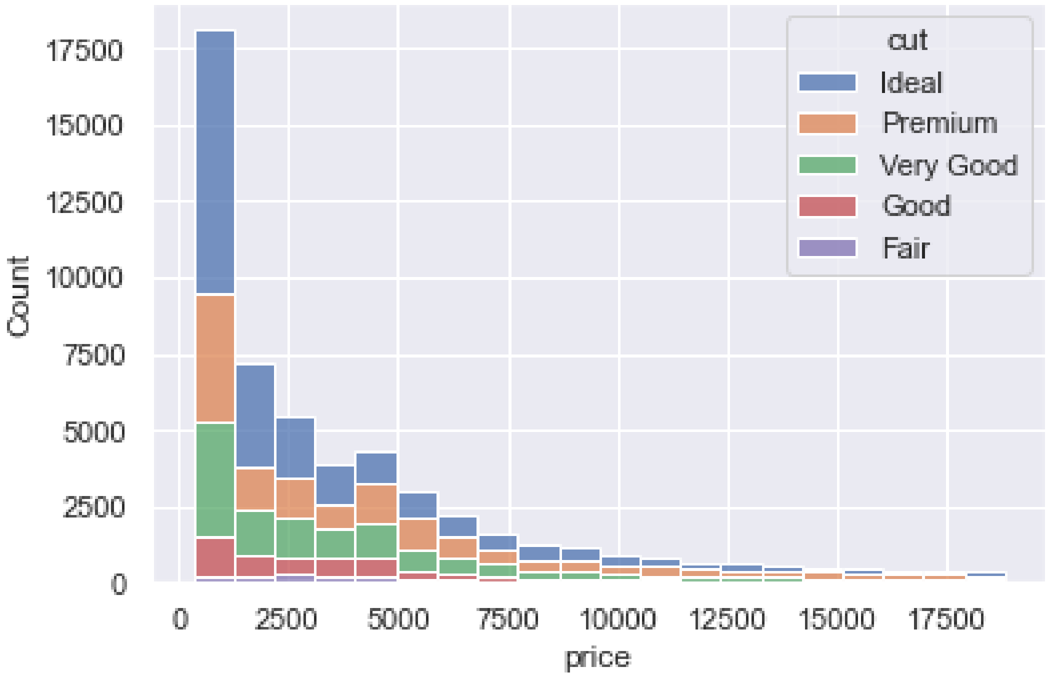
[12]:
```
# Figure level
sns.displot( data=df, x="price", hue="cut", multiple = "stack", kind = 'hist',␣
↪bins = 20)
```

```
[12]: <seaborn.axisgrid.FacetGrid at 0x7f78263dc940>
```



```
[13]: # Axes level
      sns.histplot(data=df, x='price', hue='cut', multiple = 'stack', bins = 20)
```

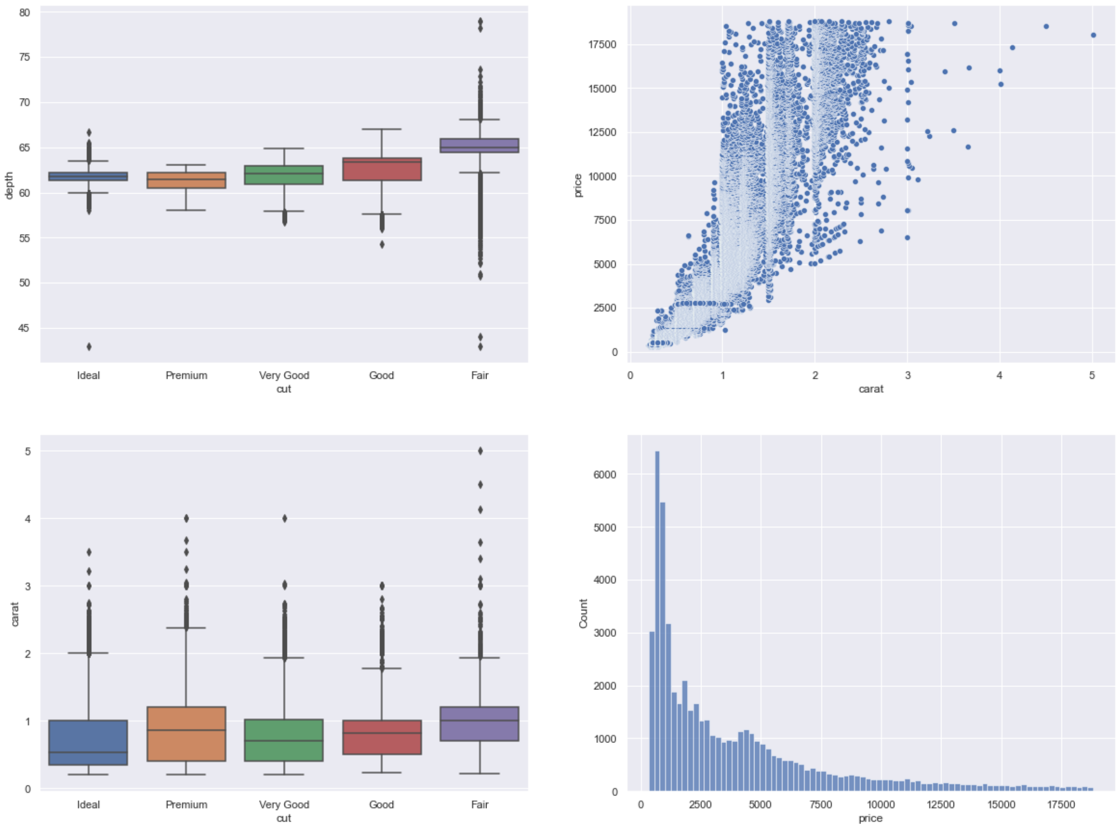
```
[13]: <AxesSubplot:xlabel='price', ylabel='Count'>
```



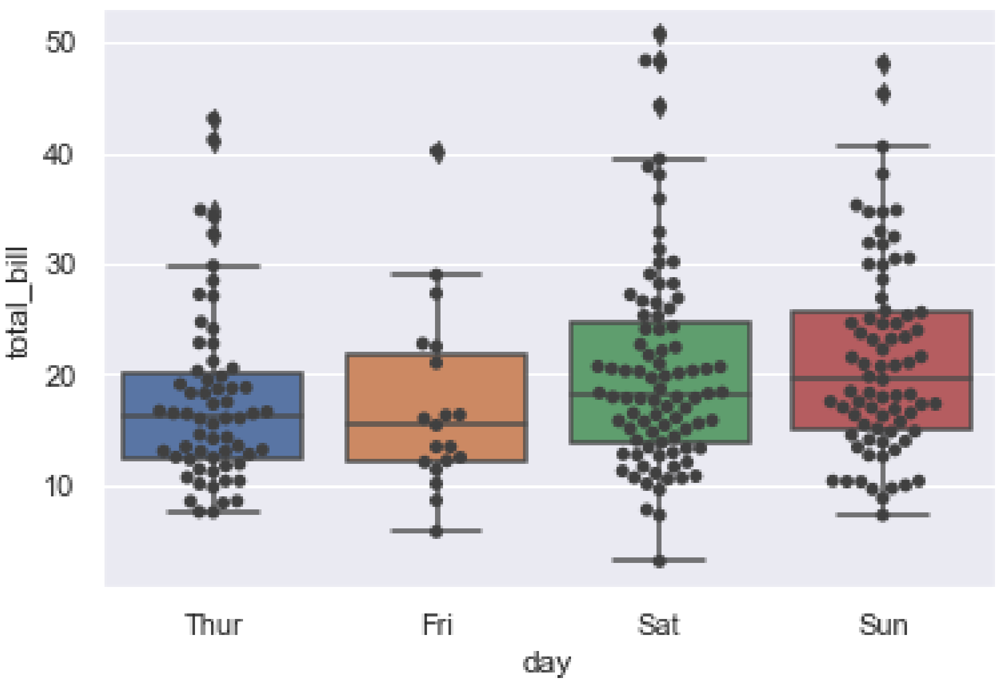
2.3 Using sns and matplotlib together

[14]:
```
plt.figure(figsize=(20, 15))
plt.subplot(2,2,1)
sns.boxplot(y = 'depth', x = 'cut', data = df)
plt.subplot(2,2,2)
sns.scatterplot(y = 'price', x = 'carat', data = df)
plt.subplot(2,2,3)
sns.boxplot(y = 'carat', x = 'cut', data = df)
plt.subplot(2,2,4)
sns.histplot(x = 'price', data = df)
```

[14]:
```
<AxesSubplot:xlabel='price', ylabel='Count'>
```



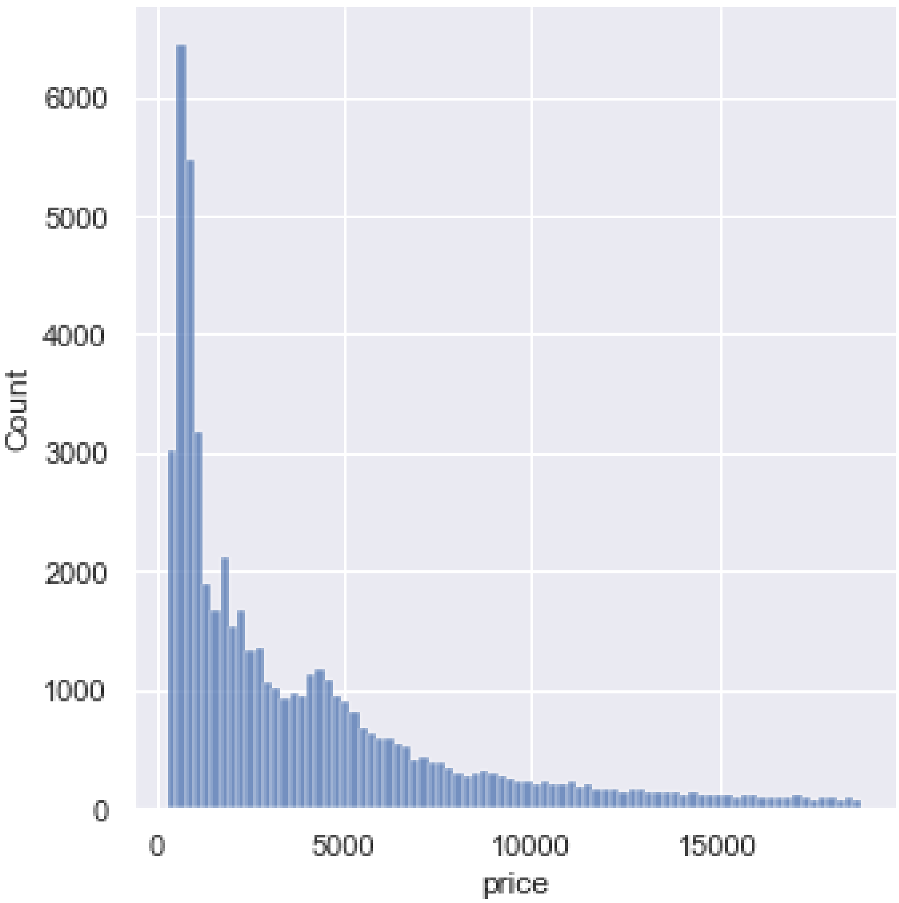
```
[15]: # From Seaborn docuentation
ax = sns.boxplot(x="day", y="total_bill", data=tips)
ax = sns.swarmplot(x="day", y="total_bill", data=tips, color=".25")
```



2.4 Figure level

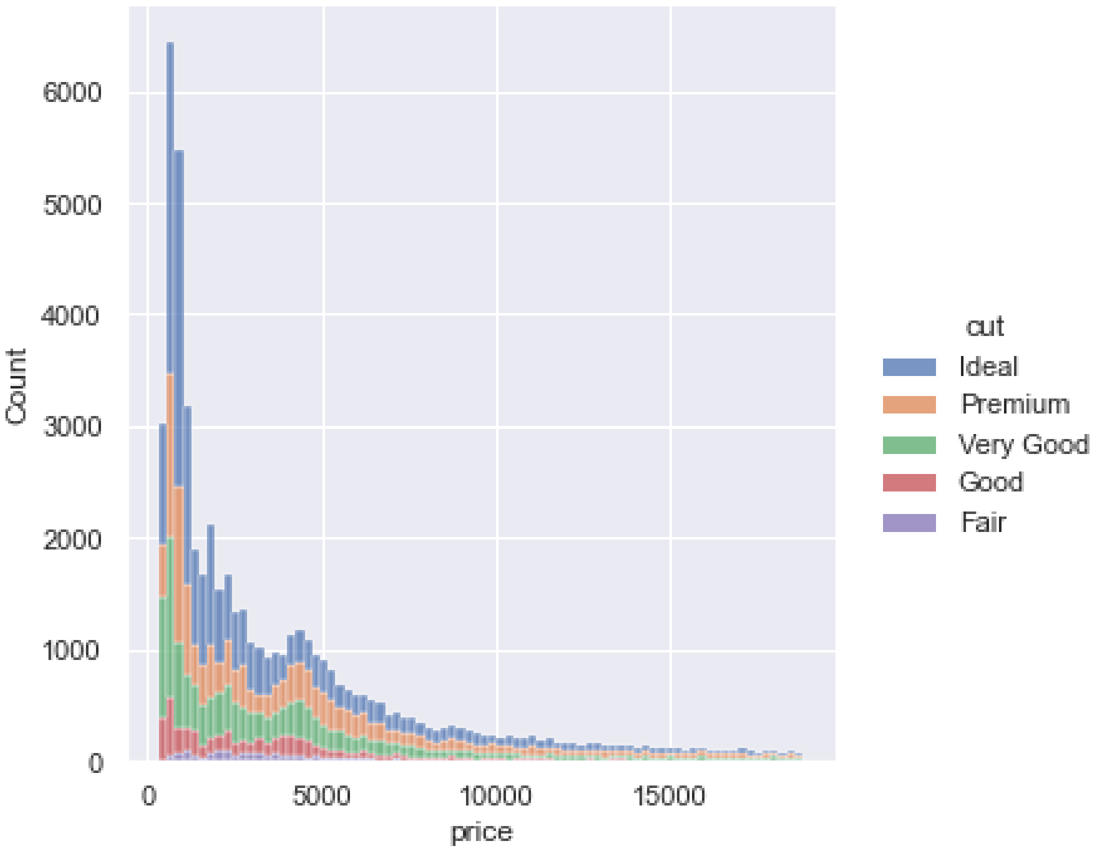
- Figure-level functions interface with matplotlib through a seaborn object, usually a FacetGrid
- Each module (relational, distributions, categorical) has a single figure-level function

[16]:
```
# The default for distplot is a histogram

sns.displot(data=df, x="price")
plt.savefig('save_as_a_png.png')

# sns.displot(data=df, x="price",height=8, aspect=15/8)  Use height and aspect␣
 ↪to change the size of the figure.
```



[17]:
```
sns.displot(data=df, x="price", hue="cut", multiple="stack")
```

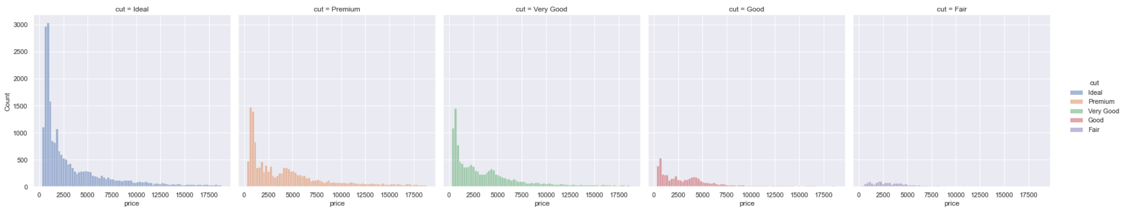
[17]:
```
<seaborn.axisgrid.FacetGrid at 0x7f7827591be0>
```



2.5 Change the plot type with kind = …

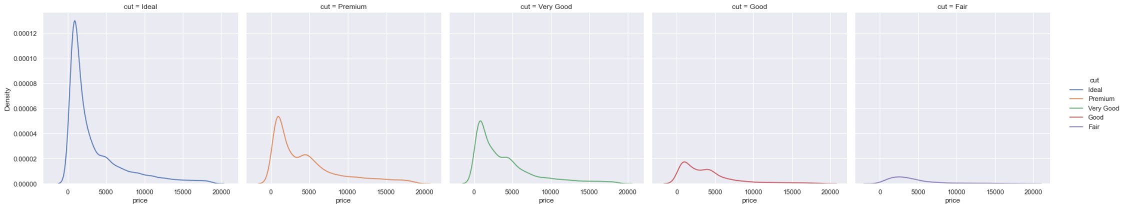
[18]:
```
sns.displot(data=df, x="price", hue="cut", col="cut", kind = 'hist')
```

[18]:
```
<seaborn.axisgrid.FacetGrid at 0x7f7809b6fe50>
```



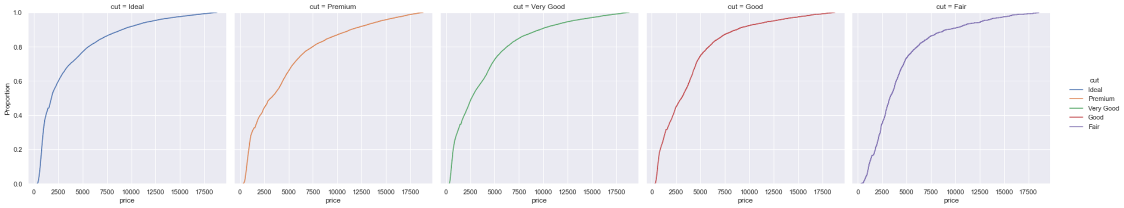
[19]:
```
sns.displot(data=df, x="price", hue="cut", col="cut", kind = 'kde')

# kernel density estimation
```

[19]:
```
<seaborn.axisgrid.FacetGrid at 0x7f782640a400>
```



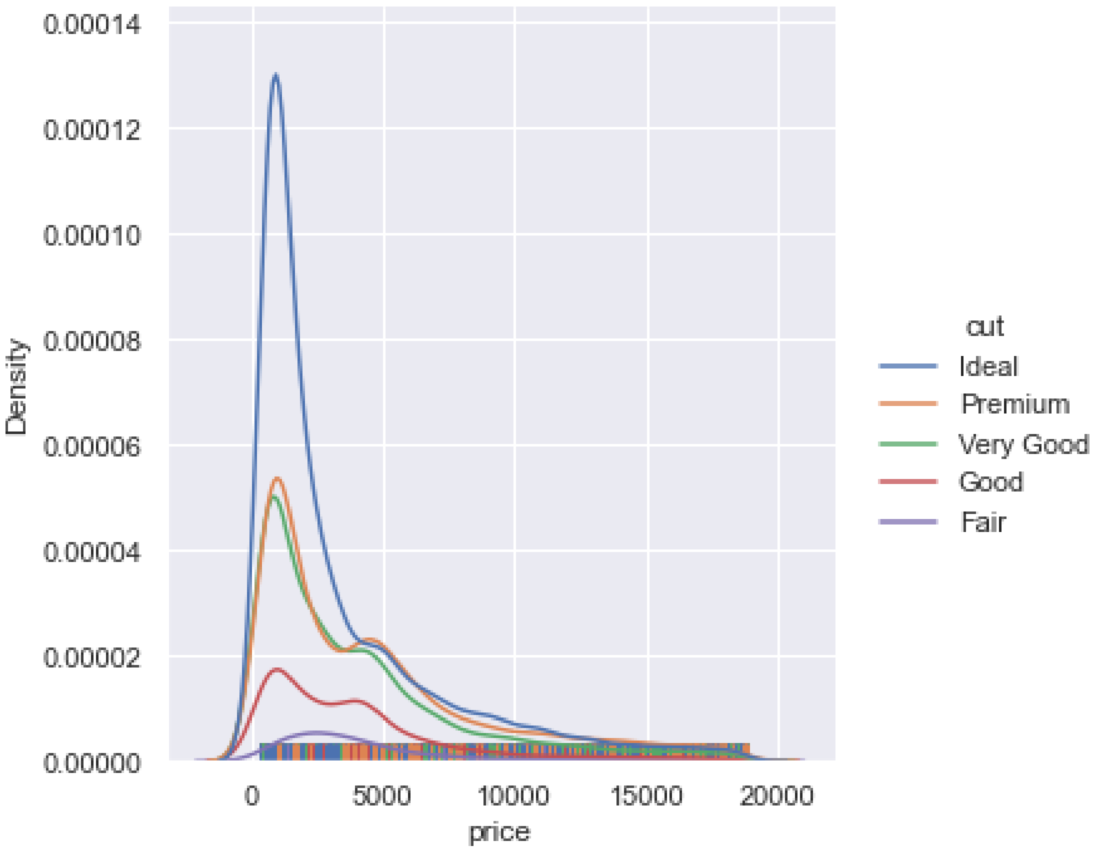
[20]:
```
sns.displot(data=df, x="price", hue="cut", col="cut", kind = 'ecdf')

# empirical cumulative distribution functions
```

[20]:
```
<seaborn.axisgrid.FacetGrid at 0x7f780a3dcf70>
```



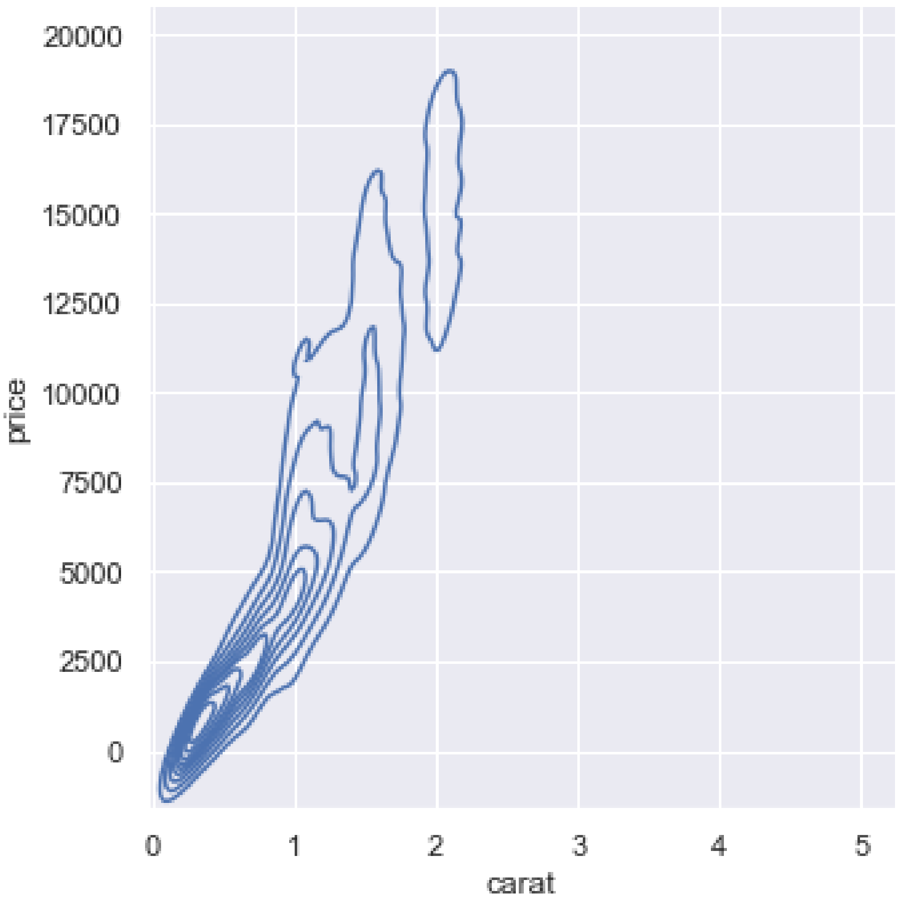
[21]:
```
sns.displot(data=df, x="price", hue="cut", kind = 'kde', rug = True)
```

[21]:
```
<seaborn.axisgrid.FacetGrid at 0x7f780bdbca00>
```



[22]:
```
# This one might take a minute to run.

sns.displot(data=df, x="carat", y='price', kind ='kde')
```

[22]:
```
<seaborn.axisgrid.FacetGrid at 0x7f780c82f610>
```



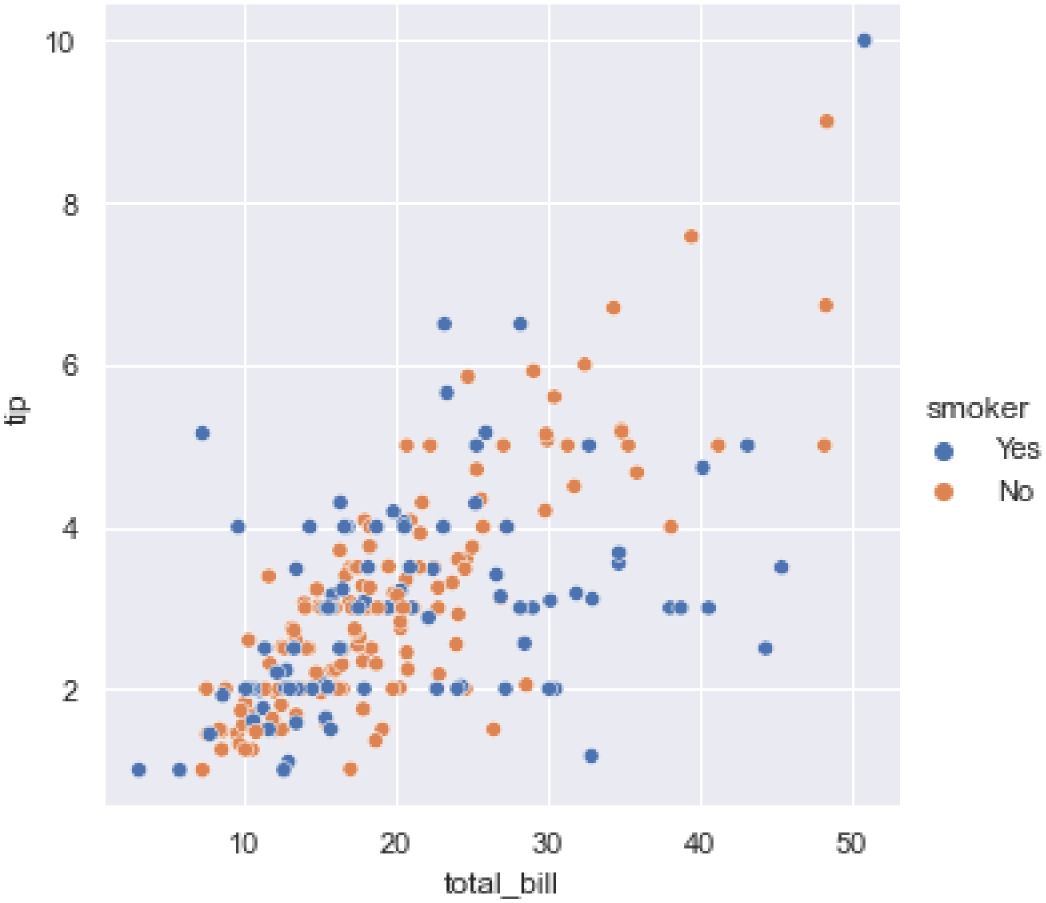
2.6 Seaborn Exercise 1 - 10 minutes

- Use the relational (relplot) figure-level function to create two charts. First a scatterplot and second a line chart.
- Use the 'tips' data set.
- For the scatterplot, determine if tips increasewith the bill amount. Try to show a distinction between data points based on time of day.
- For the line chart, show how tips change based on size of the party.

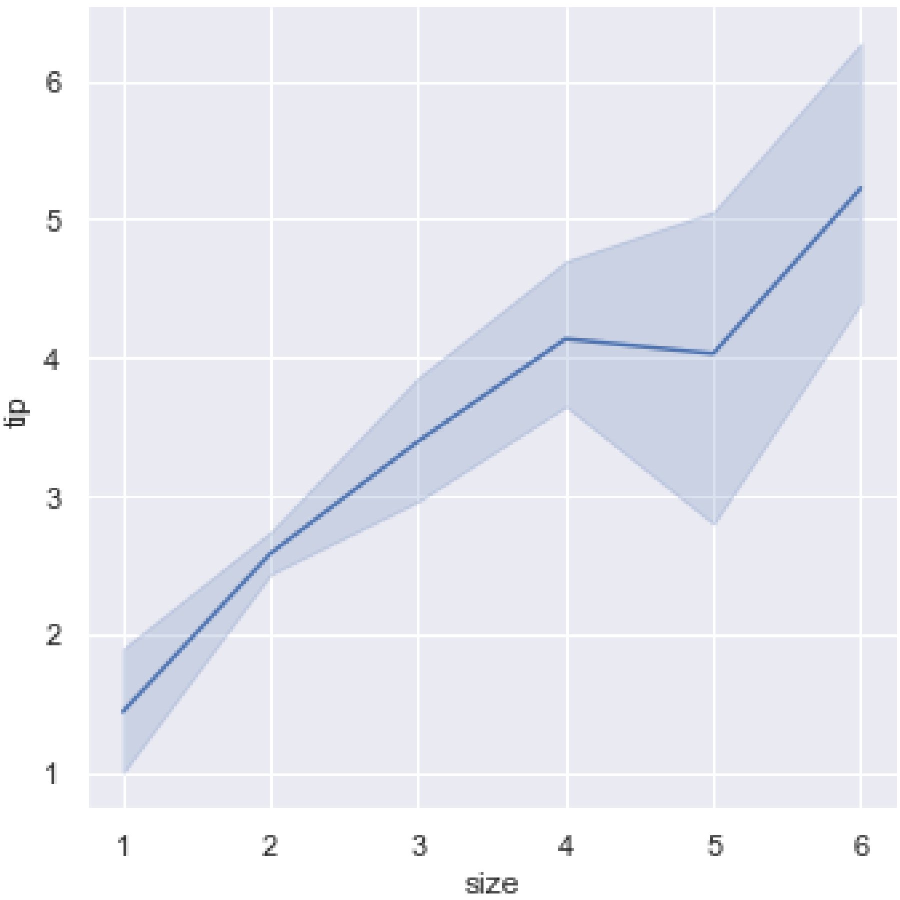
```
[23]: tips.head()
```

```
[23]:    total_bill   tip     sex smoker  day    time  size
      0       16.99  1.01  Female     No  Sun  Dinner     2
      1       10.34  1.66    Male     No  Sun  Dinner     3
      2       21.01  3.50    Male     No  Sun  Dinner     3
      3       23.68  3.31    Male     No  Sun  Dinner     2
      4       24.59  3.61  Female     No  Sun  Dinner     4
```

[24]:
```
# Place scatterplot here

sns.relplot(x="total_bill", y="tip", data=tips, kind = 'scatter', hue =␣
↪'smoker');
```

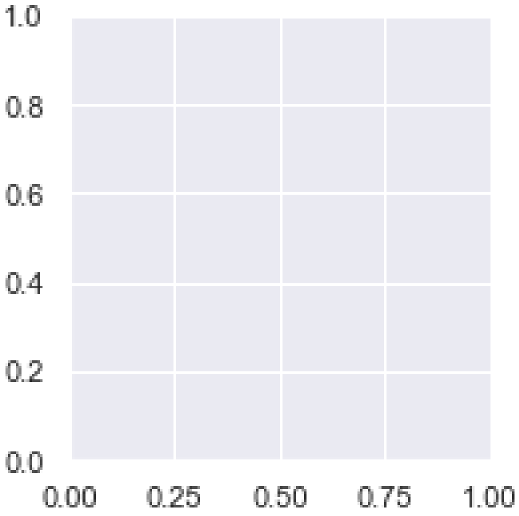


[25]:
```
# Place line chart here

sns.relplot(x="size", y="tip", data=tips, kind = 'line');
```

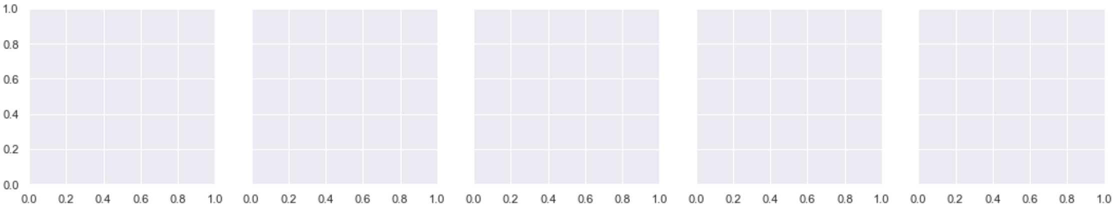


3 Facet Grids - Creating Small Multiples

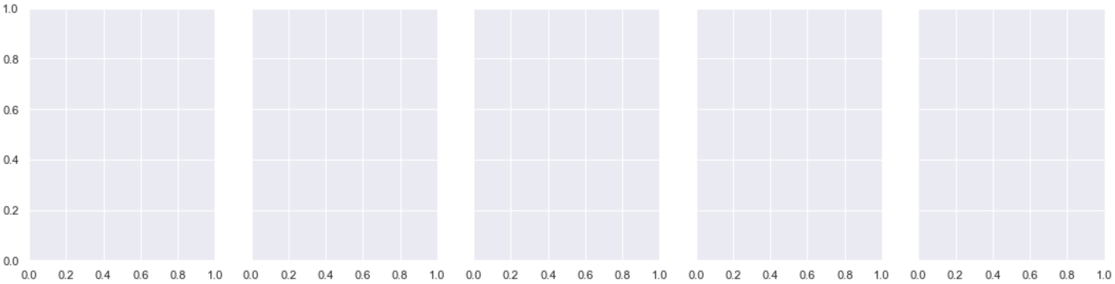
[26]:
```
p = sns.FacetGrid(df) # p is the facet grid
```



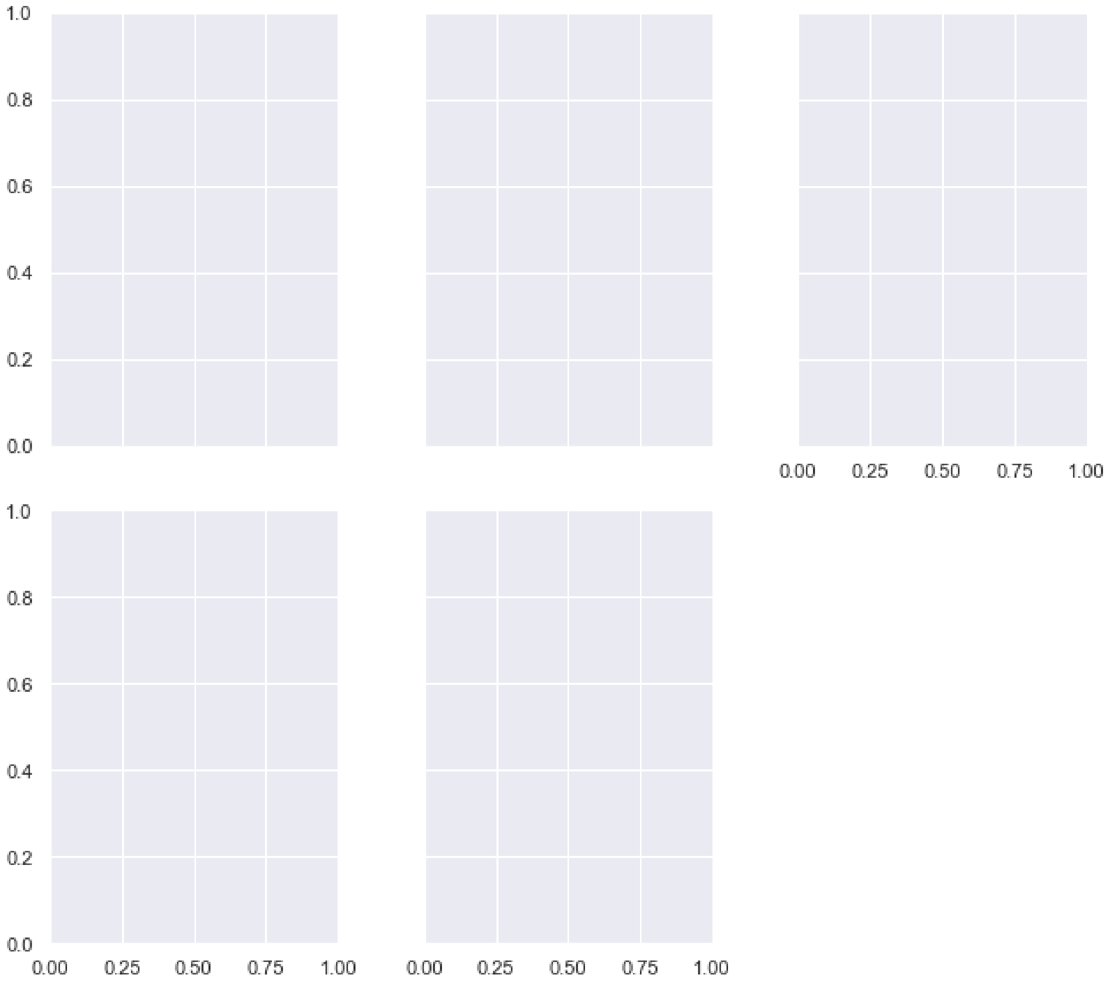
[27]:
```
p = sns.FacetGrid(df, col = 'cut') # 1 column for each facet (value) of cut.

# matplotlib will squeeze the 5 plots into the orginal size.
```



[28]:
```
p = sns.FacetGrid(df, col = 'cut', height = 4, aspect = 0.75)
# Aspect ratio of each facet, so that aspect * height gives the width of each facet.
```

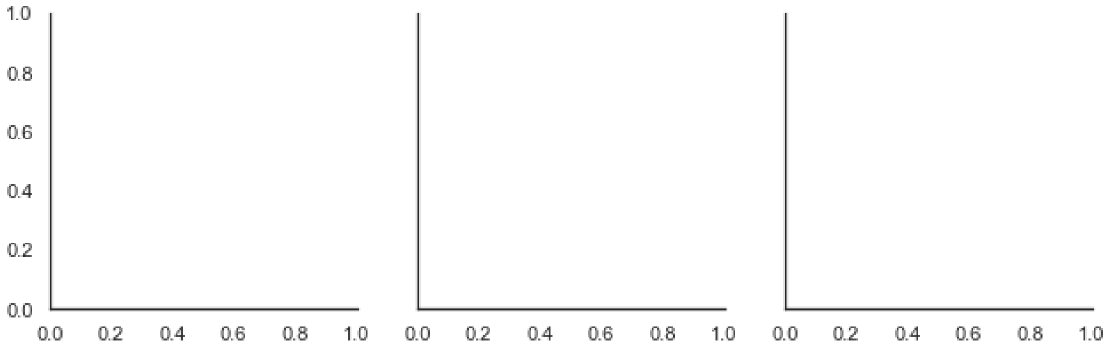


[29]:
```
p = sns.FacetGrid(df, col = 'cut', height = 4, aspect = 0.75, col_wrap = 3)
# Aspect ratio of each facet, so that aspect * height gives the width of each facet.
```

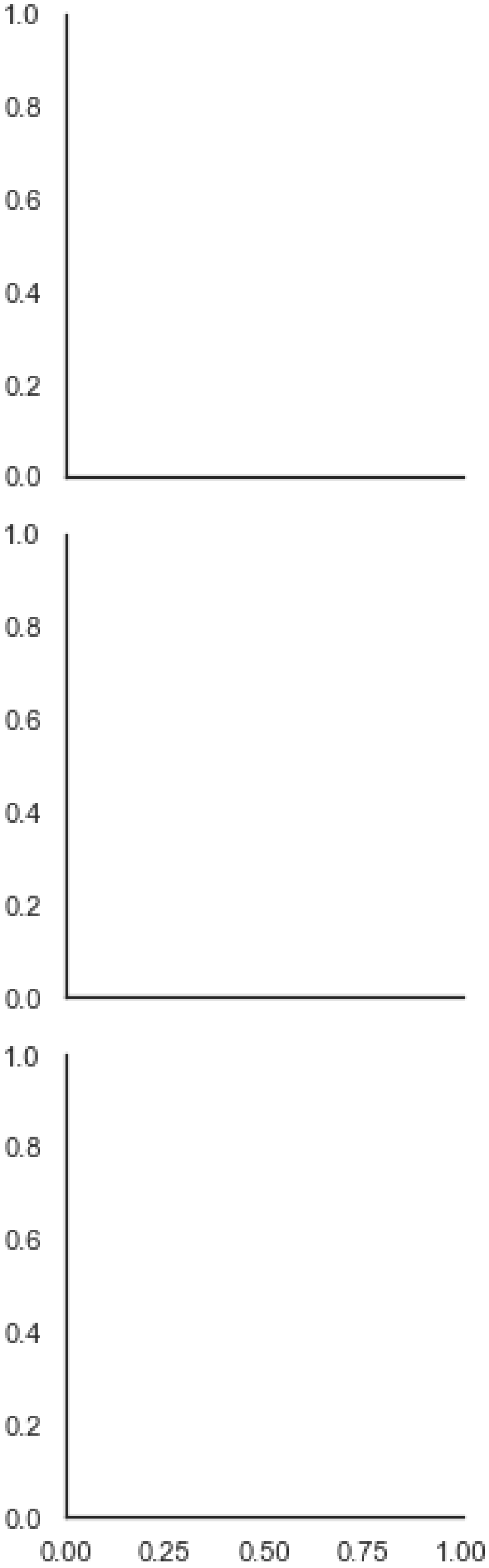


[30]:
```
sns.set_style('white')
penguins = sns.load_dataset("penguins")
```

[31]:
```
p = sns.FacetGrid(penguins, col='island');
```



[32]:
```
p = sns.FacetGrid(penguins, row='island');
```


1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00

[33]:

```
type(p)
```

[33]:

```
seaborn.axisgrid.FacetGrid
```

3.1 Managing the Facet Grid

[34]:

```
penguins.head()
```

[34]:

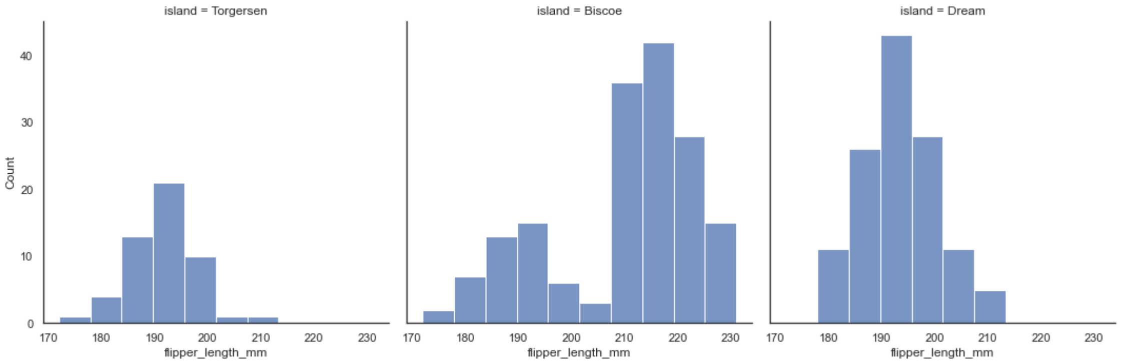
```
  species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0
1  Adelie  Torgersen            39.5           17.4              186.0
2  Adelie  Torgersen            40.3           18.0              195.0
3  Adelie  Torgersen             NaN            NaN                NaN
4  Adelie  Torgersen            36.7           19.3              193.0

   body_mass_g     sex
0       3750.0    Male
1       3800.0  Female
2       3250.0  Female
3          NaN     NaN
4       3450.0  Female
```

[35]:

```
sns.displot(data=penguins, x="flipper_length_mm", col="island", kind = 'hist')
```

[35]:

```
<seaborn.axisgrid.FacetGrid at 0x7f780bde5490>
```



3.2 Using methods of FacetGrid

These will be used when an axes level plot is used.

Three steps: - set up the FacetGrid - identify the plot type using .map or .map_dataframe - customize

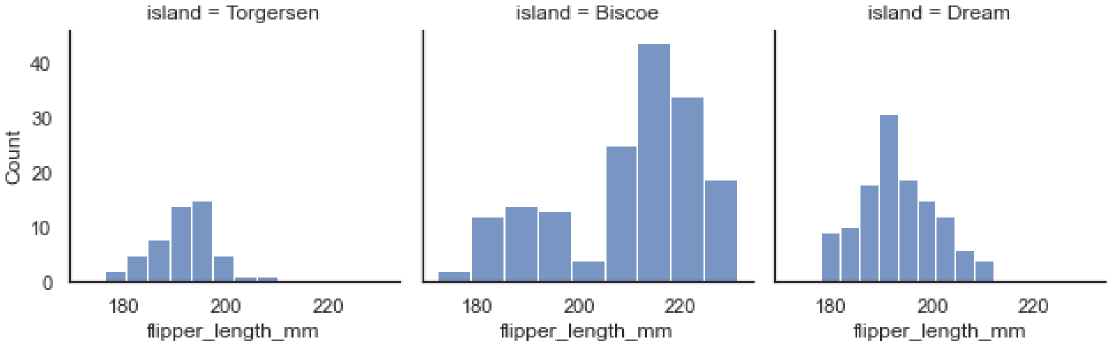
3.2.1 .map()

1. Set up the facet grid (format the facets)
2. Describe what should be plotted in the grids
3. Add extras - labels, titles, etc.

To draw a plot on every facet, pass a function and the name of one or more columns in the dataframe to FacetGrid.map()

[36]:

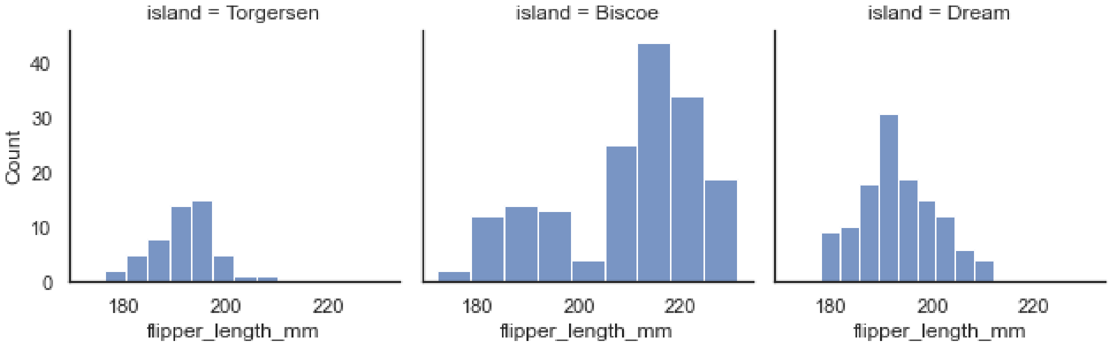
```
p = sns.FacetGrid(penguins, col='island')
p.map(sns.histplot, 'flipper_length_mm'); # Requires positional arguements, not␣
 ↪named (x = 'flipper_length_mm')
```



3.2.2 .map_dataframe()

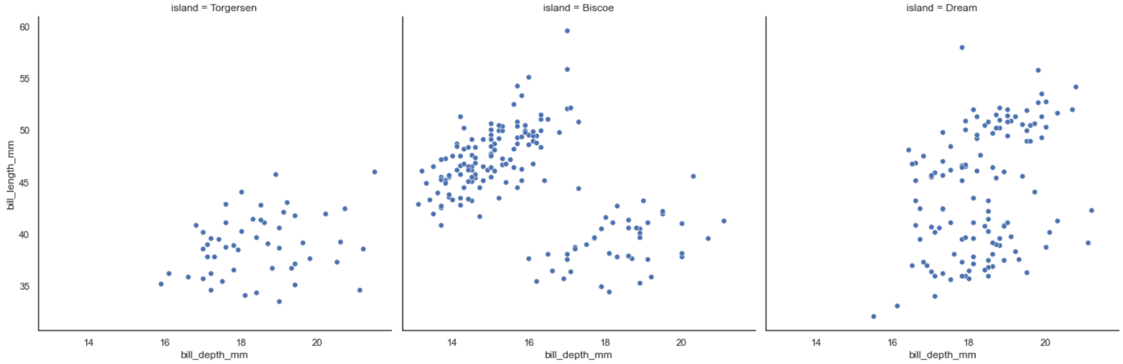
[37]:

```
p = sns.FacetGrid(penguins, col='island')
p.map_dataframe(sns.histplot, x = 'flipper_length_mm'); # It is the x= that is␣
 ↪different.  This is the named arguement
```



[38]:

```
p = sns.FacetGrid(penguins, col='island', height = 6, aspect =1)
p.map_dataframe(sns.scatterplot,  y='bill_length_mm',x='bill_depth_mm');
```



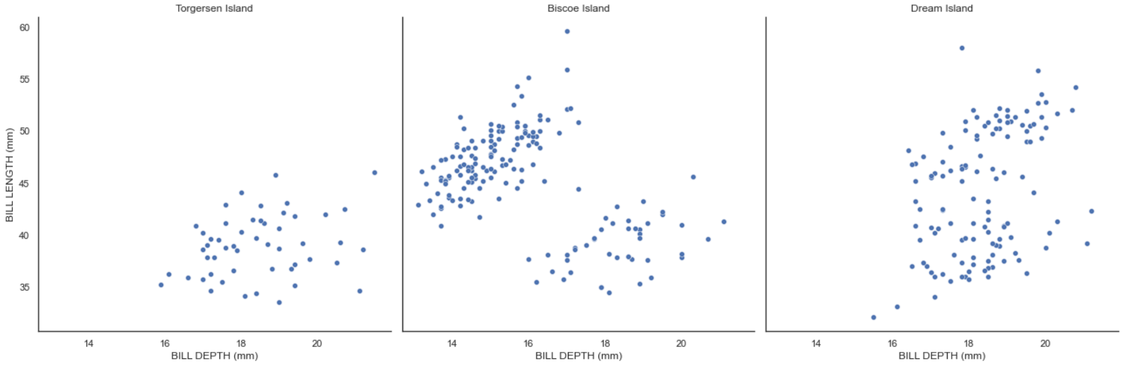
[ ]:

```

```

3.2.3 .set_axis_labels(), .set_titles(), sharey, ylim

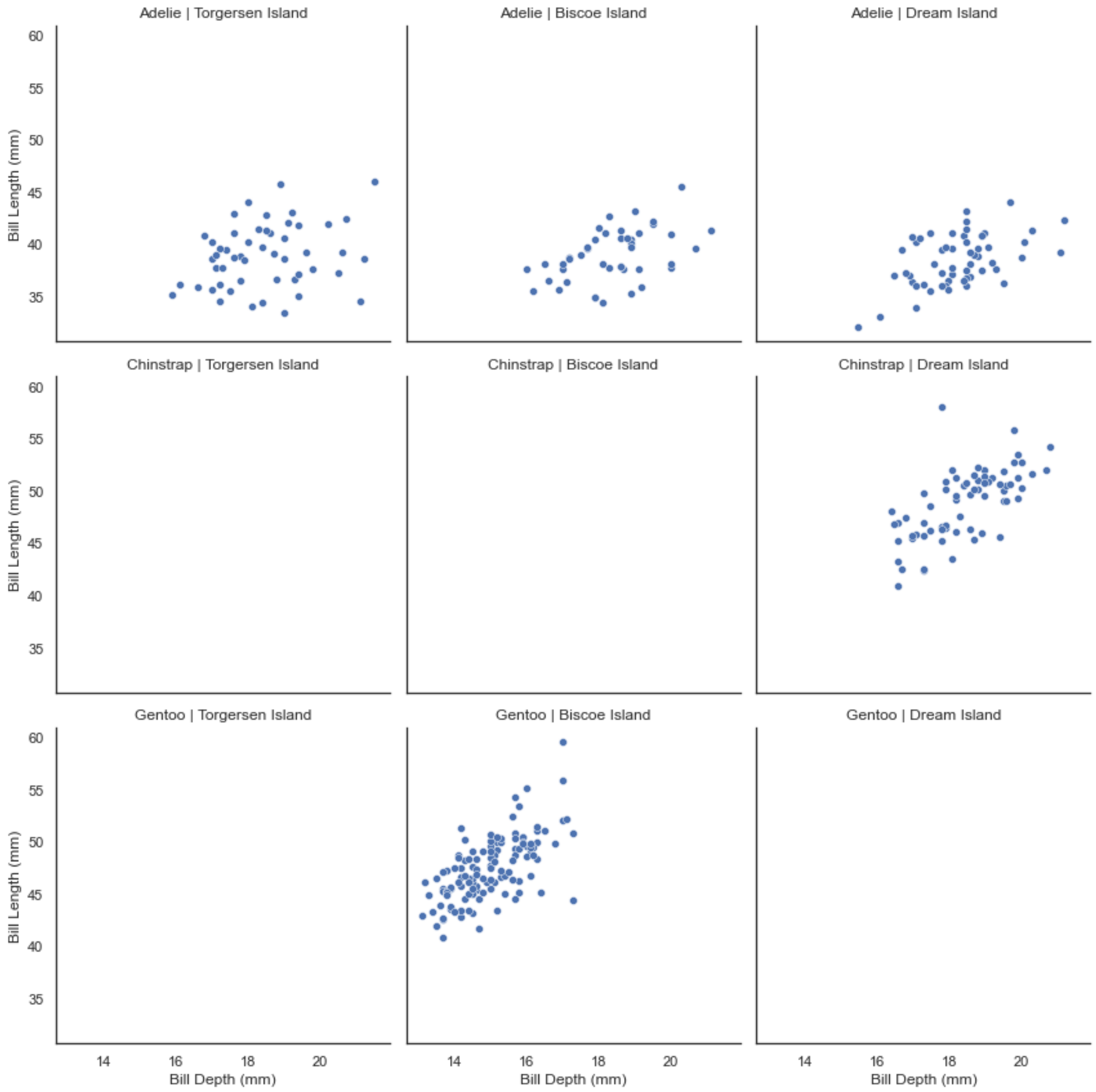
[39]:

```
p = sns.FacetGrid(penguins, col='island', height = 6, aspect =1)
p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm')

p.set_axis_labels('BILL DEPTH (mm)', 'BILL LENGTH (mm)'); # if the LABELS needs␣
 ↪to be changed
p.set_titles(col_template='{col_name} Island'); # if the TITLE needs to be␣
 ↪changed
```



[40]:

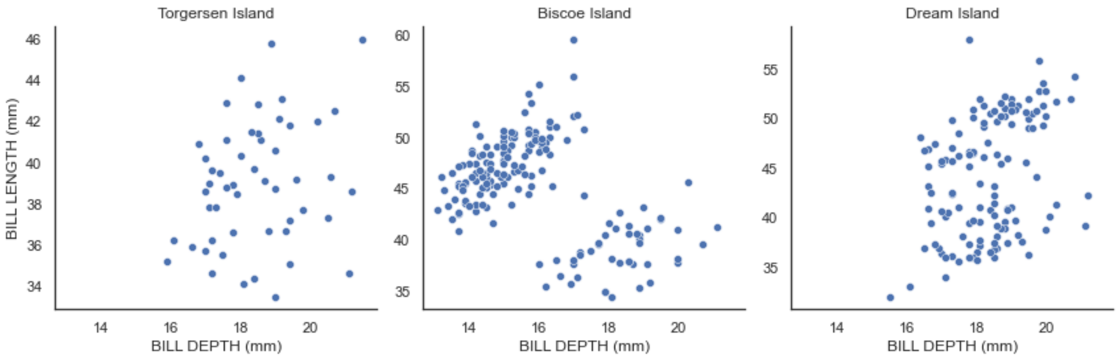
```
p = sns.FacetGrid(penguins, col='island', row='species', height = 4, aspect =1)
p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm')
p.set_axis_labels('Bill Depth (mm)', 'Bill Length (mm)')
p.set_titles(row_template='{row_name}', col_template='{col_name} Island');
```



- sharey: False means the y-axis will not be shared and each plot will get its own y-axis.
- ylim: Sets a specified range for all y-axes shown

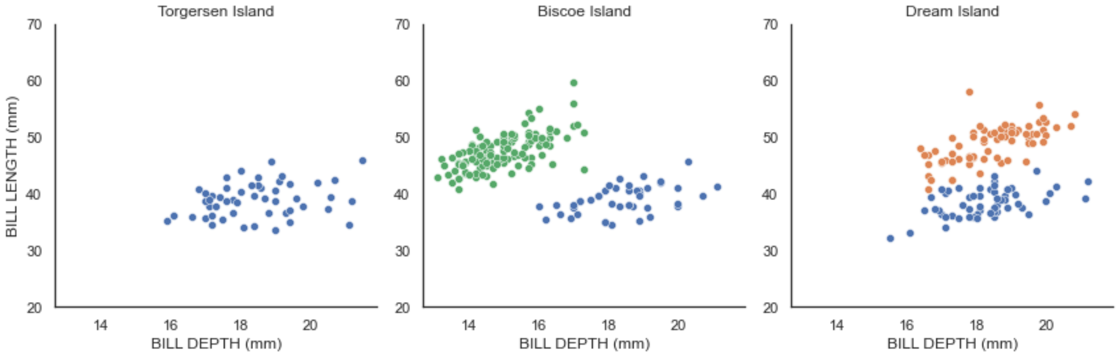
sharey = False

```
[41]: p = sns.FacetGrid(penguins, col='island', height = 4, aspect =1, sharey=False)

#p = sns.FacetGrid(penguins, col='island', height = 4, aspect =1, sharey=False,␣
 ↪ylim=(20, 70))

p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm');
p.set_axis_labels('BILL DEPTH (mm)', 'BILL LENGTH (mm)');
p.set_titles(col_template='{col_name} Island');
```



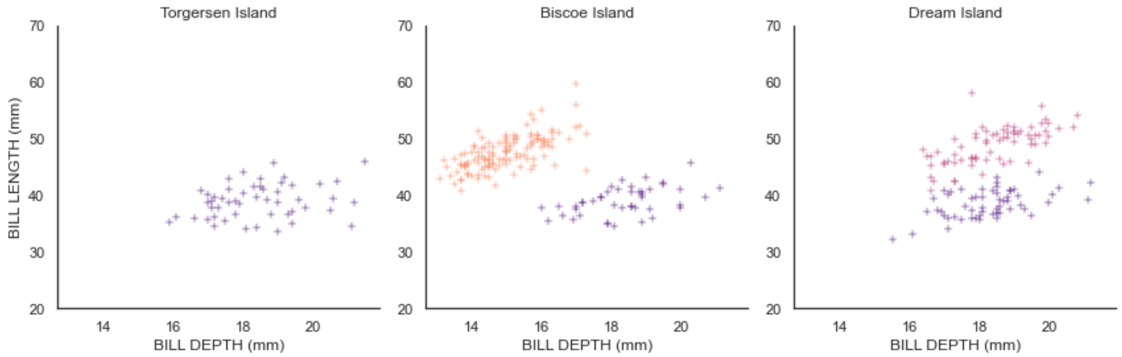
3.2.4 hue & pallette

```
[42]: p = sns.FacetGrid(penguins, col='island', height = 4, aspect =1, sharey=False,␣
       ↪ylim=(20, 70), hue = 'species')
      p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm');
      p.set_axis_labels('BILL DEPTH (mm)', 'BILL LENGTH (mm)');
      p.set_titles(col_template='{col_name} Island');
```



```
[43]: p = sns.FacetGrid(penguins,
                        col='island',
                        height = 4,
                        aspect =1,
                        sharey=False,
                        ylim=(20, 70),
                        hue = 'species',
                        palette = 'magma'
                        #palette = ['grey','blue','red']
                       )
```

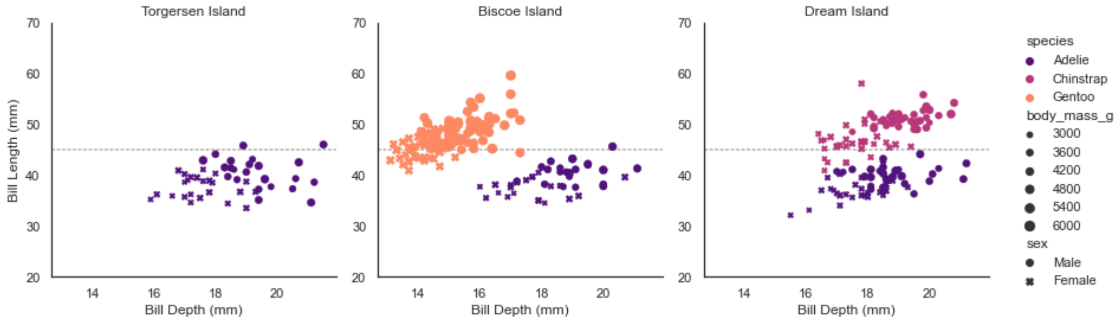
```
p.map_dataframe(sns.scatterplot, x='bill_depth_mm', y='bill_length_mm', marker␣
↪= '+');
p.set_axis_labels('BILL DEPTH (mm)', 'BILL LENGTH (mm)');
p.set_titles(col_template='{col_name} Island');
```



3.2.5 Accomplish the same without defining the facet grid first

[44]:
```
p = sns.relplot(data=penguins, x='bill_depth_mm', y='bill_length_mm', kind =␣
↪'scatter',
            col='island', height = 4, aspect =1, hue = 'species', palette =␣
↪'magma',
            marker = '+',
            size = 'body_mass_g',
            style = 'sex',
            facet_kws={'sharey': False, 'sharex': True, 'ylim':(20,70)}
               #   sharey=False, ylim=(20, 70), #palette = ['grey','blue','red']
           )

p.map(plt.axhline,
        y=45, color=".7",
        dashes=(2, 1),
        zorder=0)
p.set_axis_labels('Bill Depth (mm)', 'Bill Length (mm)')
p.set_titles(row_template='{row_name}', col_template='{col_name} Island')
```

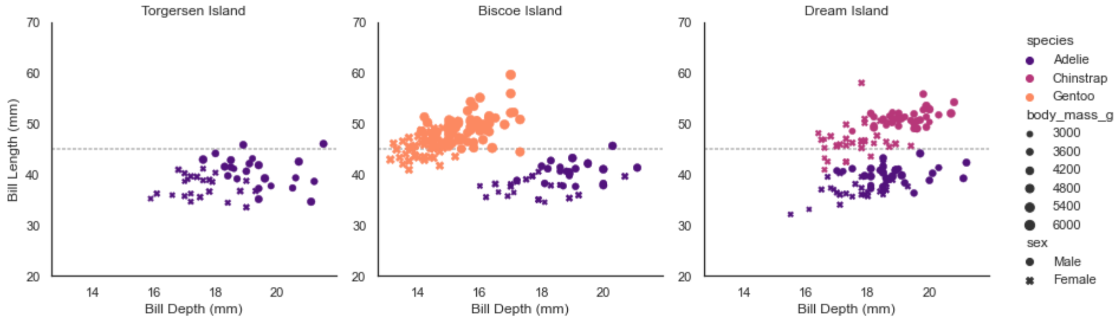
[44]:
```
<seaborn.axisgrid.FacetGrid at 0x7f7811063d60>
```



3.2.6 Method Chaining

[45]:
```
p = sns.relplot(data=penguins, x='bill_depth_mm', y='bill_length_mm', kind =␣
↪'scatter',
             col='island', height = 4, aspect =1, hue = 'species', palette =␣
↪'magma'
             ,marker = '+',
             size = 'body_mass_g',
             style = 'sex',
             facet_kws={'sharey': False, 'sharex': True, 'ylim':(20,70)}
            )

(
p.map(plt.axhline,
        y=45, color=".7",
        dashes=(2, 1),
        zorder=0)
 .set_axis_labels('Bill Depth (mm)', 'Bill Length (mm)')
 .set_titles(row_template='{row_name}', col_template='{col_name} Island'))
```

[45]:
```
<seaborn.axisgrid.FacetGrid at 0x7f78102cca60>
```



3.3 Seaborn Exercise 2 - 10 minutes

Using the flights info, create a visualization that plots - for each month - the number of passengers by year.
There should be one plot per month.

[46]:

```
flights.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   year        144 non-null    int64
 1   month       144 non-null    category
 2   passengers  144 non-null    int64
dtypes: category(1), int64(2)
memory usage: 2.9 KB
```

[47]:

```
flights.head(20)
```

[47]:

```
    year month  passengers
0   1949   Jan         112
1   1949   Feb         118
2   1949   Mar         132
3   1949   Apr         129
4   1949   May         121
5   1949   Jun         135
6   1949   Jul         148
7   1949   Aug         148
8   1949   Sep         136
9   1949   Oct         119
10  1949   Nov         104
11  1949   Dec         118
12  1950   Jan         115
13  1950   Feb         126
14  1950   Mar         141
15  1950   Apr         135
16  1950   May         125
17  1950   Jun         149
18  1950   Jul         170
19  1950   Aug         170
```

[48]:

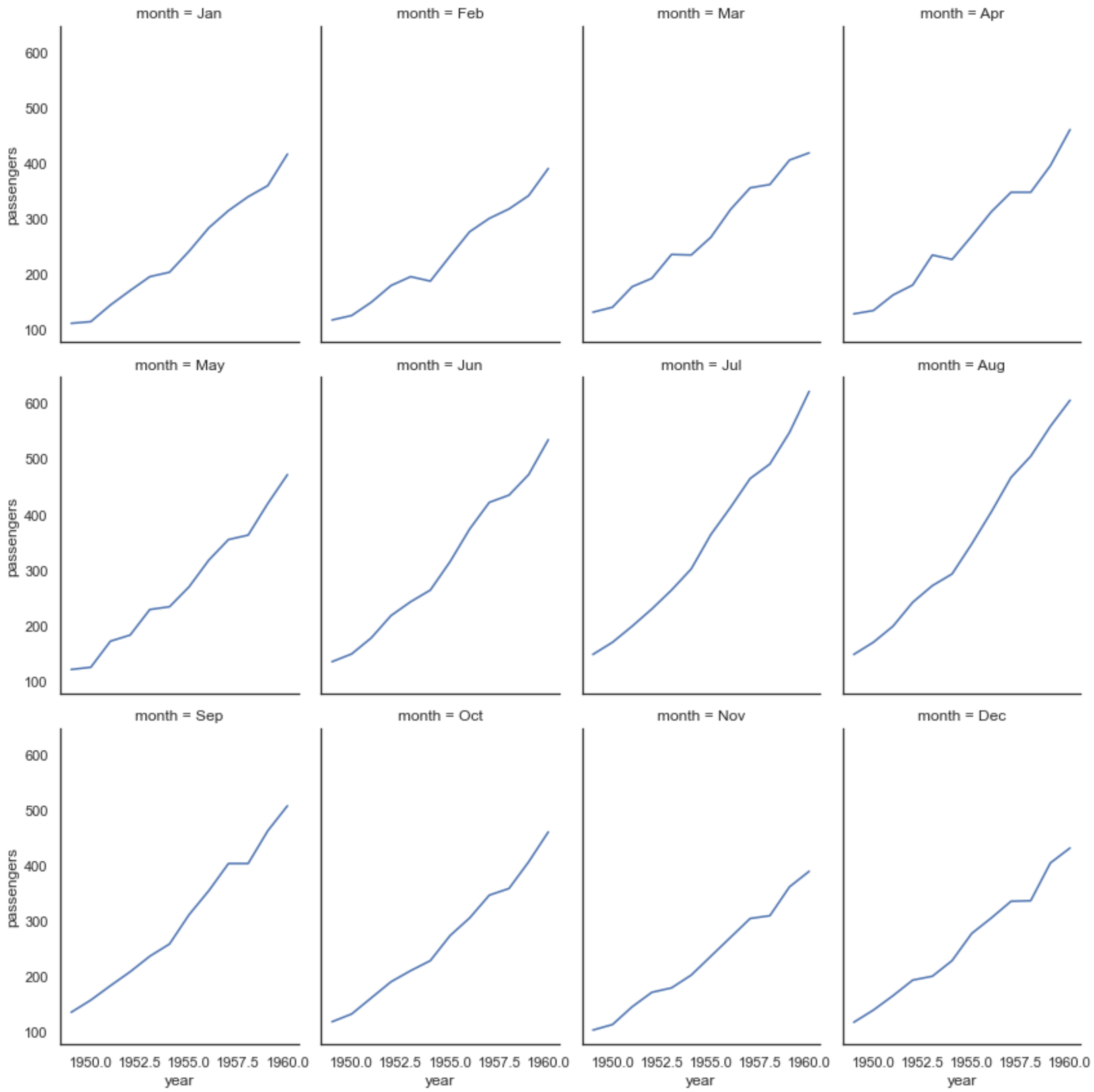
```
flights.shape
```

[48]:

```
(144, 3)
```

[49]:

```
# SNS Exercise 2 solution here.
p = sns.FacetGrid(flights, col = 'month', height = 4, aspect = 0.75, col_wrap = 4)
p.map_dataframe(sns.lineplot, x='year', y='passengers');
```



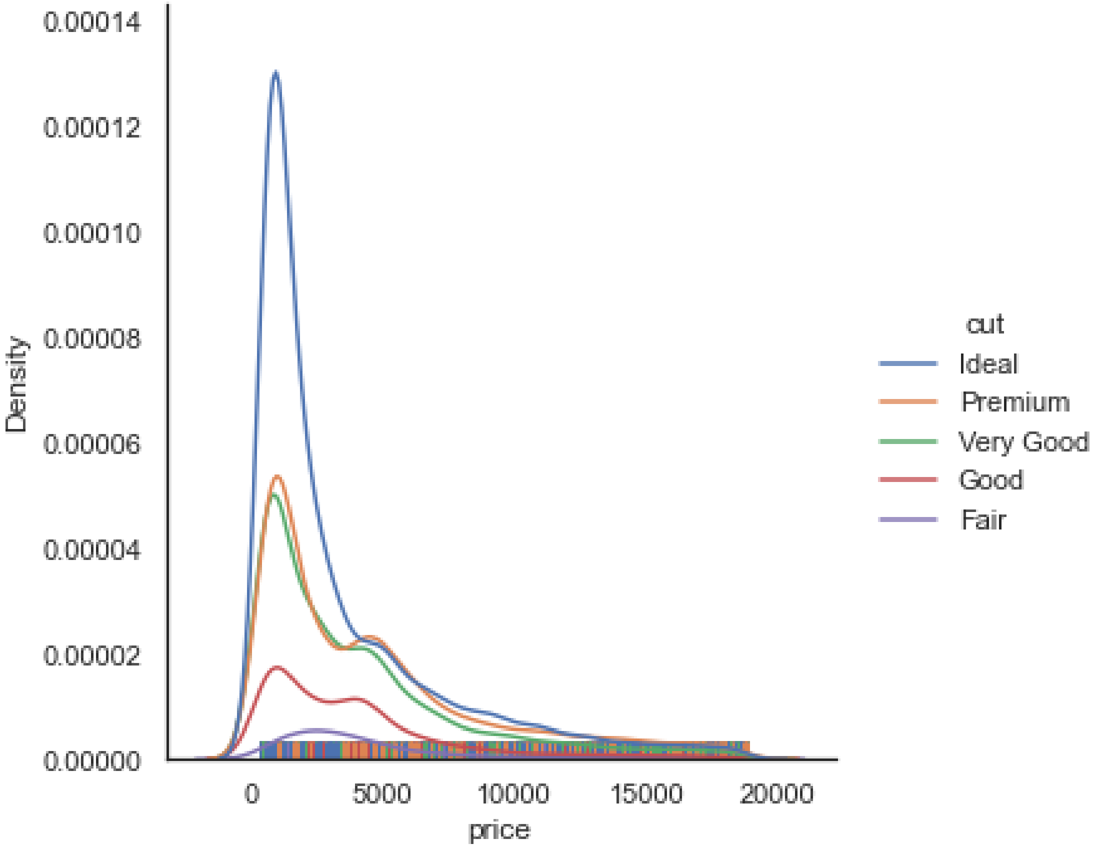
3.4 Seaborn Exercise 3 - 15 minutes

The distplot below is quick 'one-liner' plot. Take a little more time to create an axes for each cut and the axes are one above the other.

[50]:

```
# This is the chart on the left with diamond data.
sns.displot(data=df, x="price", hue="cut", kind = 'kde', rug = True)
```

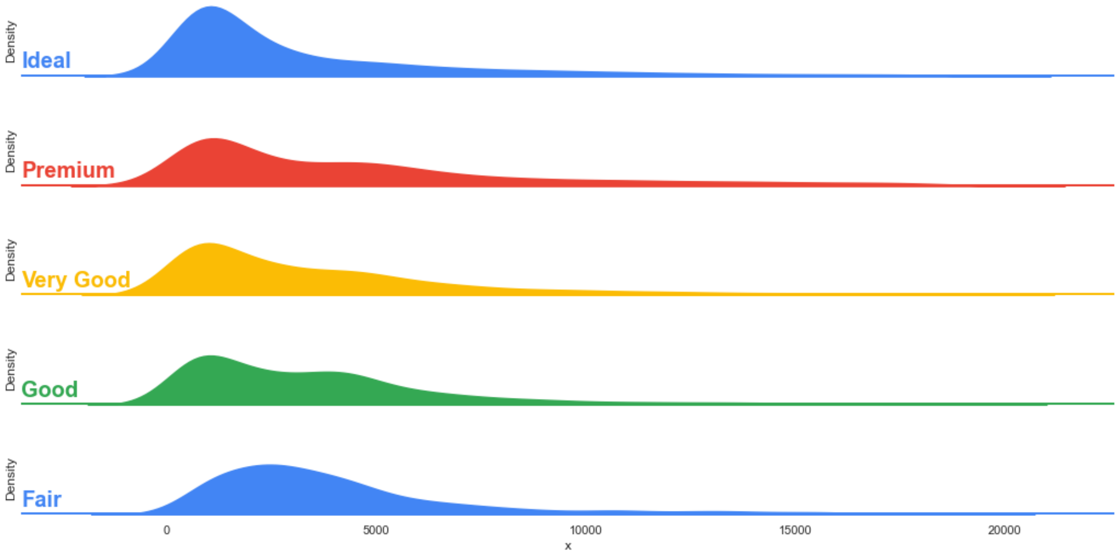
```
[50]: <seaborn.axisgrid.FacetGrid at 0x7f7812fb3b20>
```



```
[51]: # Place Exercise 3 solution here.

      # https://towardsdatascience.com/
       ↪sorry-but-sns-distplot-just-isnt-good-enough-this-is-though-ef2ddbf28078
      df = sns.load_dataset("diamonds")

      sns.set_style('white')
      g = sns.FacetGrid(df, #the dataframe to pull from
                        row="cut", #define the column for each subplot row to be␣
       ↪differentiated by
                        hue="cut", #define the column for each subplot color to be␣
       ↪differentiated by
                        aspect=10, #aspect * height = width
                        height=1.5, #height of each subplot
                        palette=['#4285F4','#EA4335','#FBBC05','#34A853'] #google␣
       ↪colors
                       )
```

```
#shade: True/False, shade area under curve or not
#alpha: transparency, lw: line width, bw: kernel shape specification


#g.map(sns.kdeplot, "price", lw=4, bw_method=0.2)   Same as below but no fill
g.map(sns.kdeplot, "price", shade=True, alpha=1, lw=1.5, bw_method=0.2)
g.map(plt.axhline, y=0, lw=4)

def label(x, color, label):
    ax = plt.gca() #get the axes of the current object
    ax.text(0, .2, #location of text
            label, #text label
            fontweight="bold", color=color, size=20, #text attributes
            ha="left", va="center", #alignment specifications
            transform=ax.transAxes) #specify axes of transformation

g.map(label, "x") #the function counts as a plotting object!


g.set_titles("") #set title to blank
g.set(yticks=[]) #set y ticks to blank
g.despine(bottom=True, left=True) #remove 'spines'
```

```
[51]: <seaborn.axisgrid.FacetGrid at 0x7f7825def7c0>
```

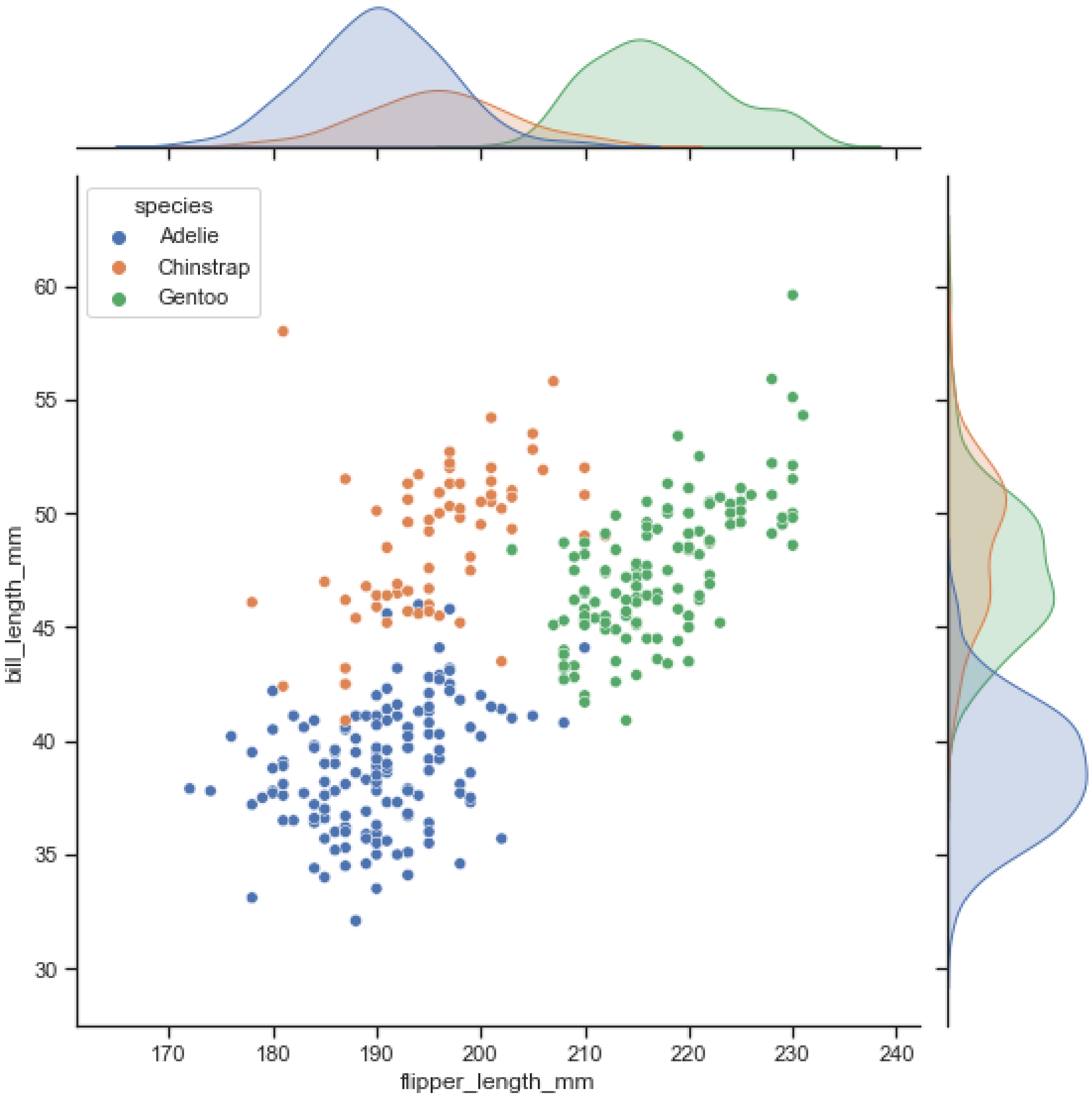


4 Multiple Views

4.1 Jointplot

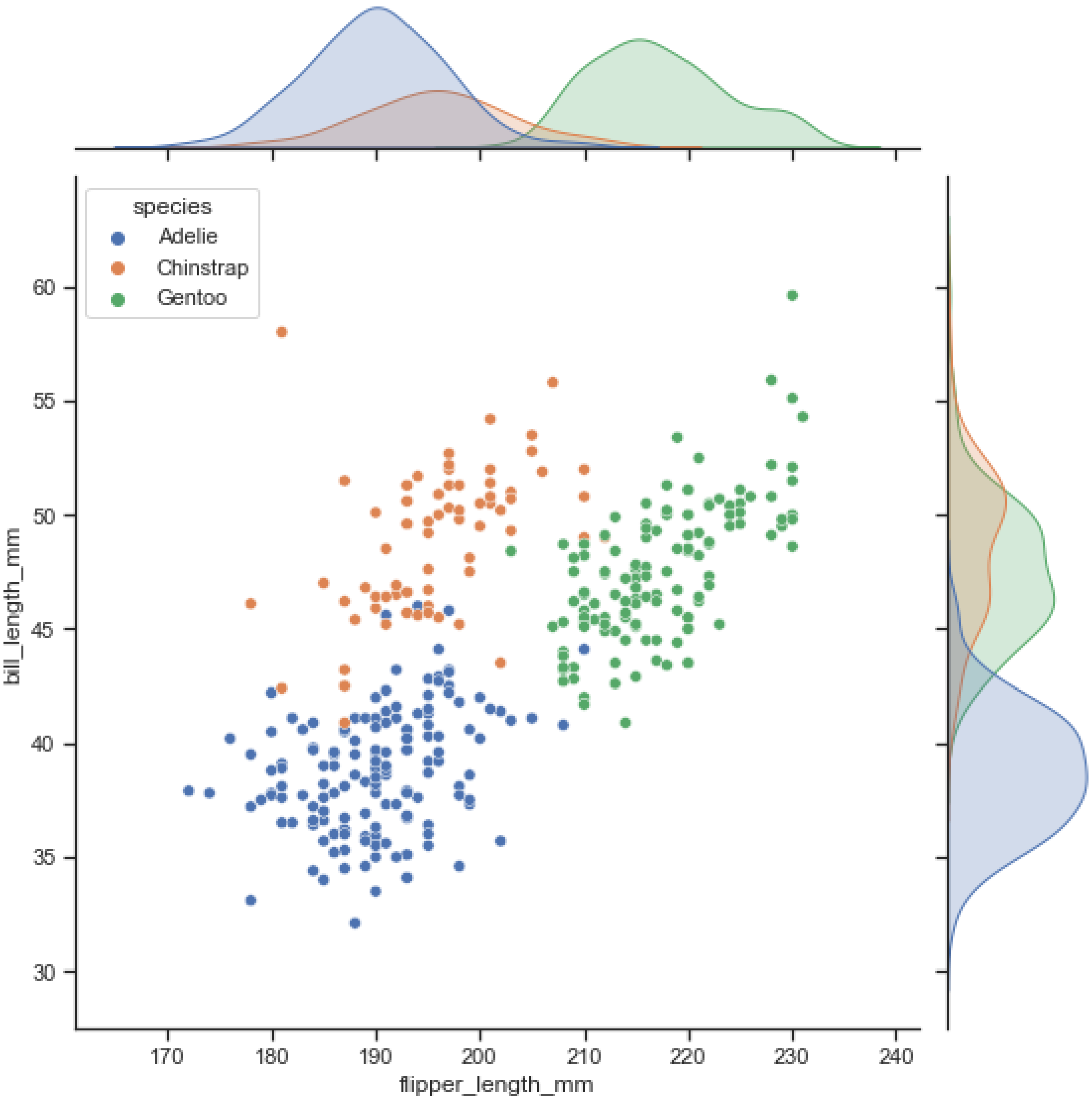
```
[52]: sns.set_style("ticks")
      sns.jointplot(data = penguins, x="flipper_length_mm", y="bill_length_mm",␣
       ↪hue="species", height = 8 )
```

```
[52]: <seaborn.axisgrid.JointGrid at 0x7f7814eaf100>
```



```
[53]: sns.set_style("ticks")
      sns.jointplot(data = penguins, x="flipper_length_mm", y="bill_length_mm",␣
       ↪hue="species", height = 8 )
```

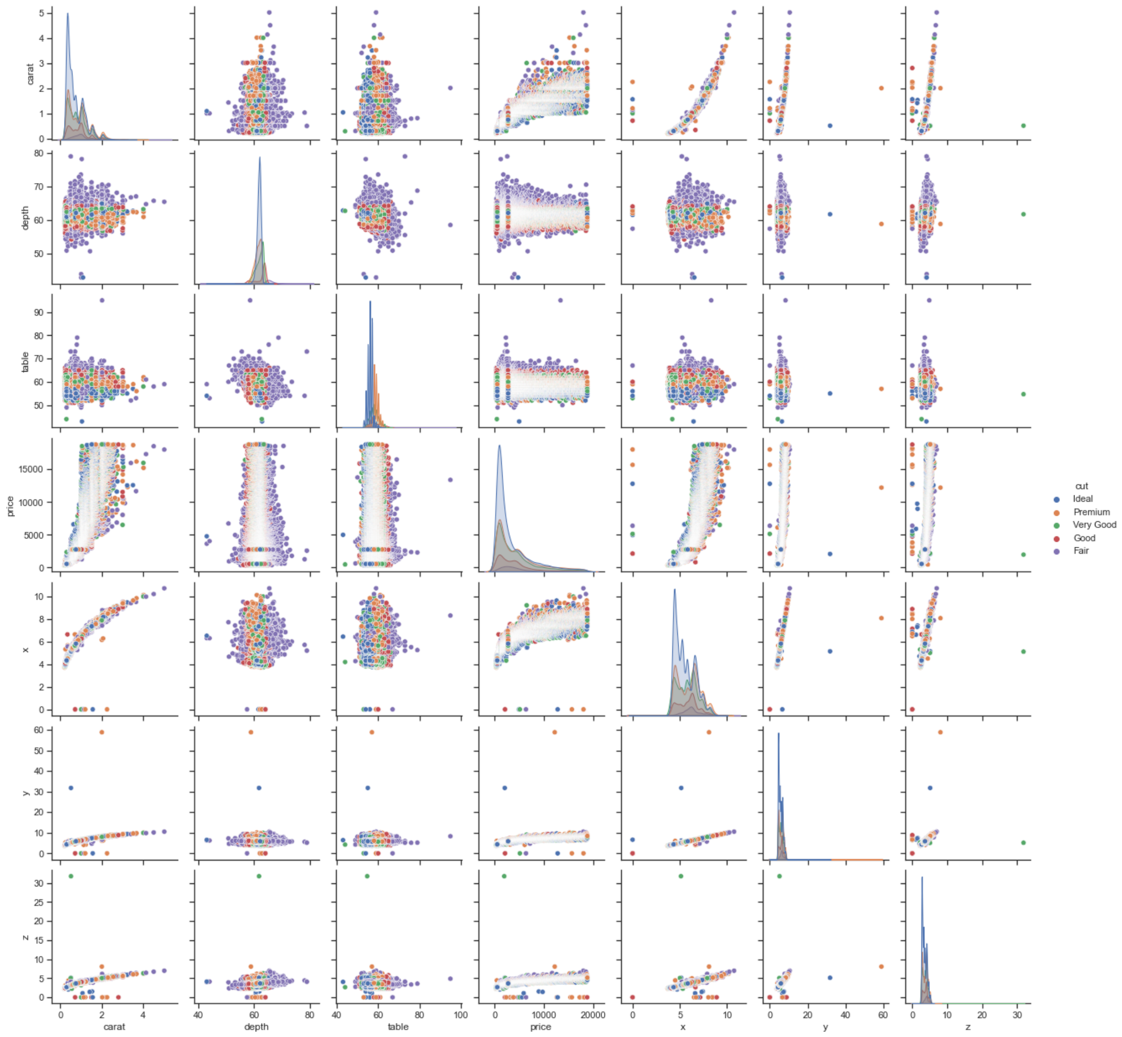
```
[53]: <seaborn.axisgrid.JointGrid at 0x7f7812258550>
```



4.2 Pairplot

```
[54]: sns.pairplot(data = df, hue = 'cut')
```

```
[54]: <seaborn.axisgrid.PairGrid at 0x7f7815107550>
```

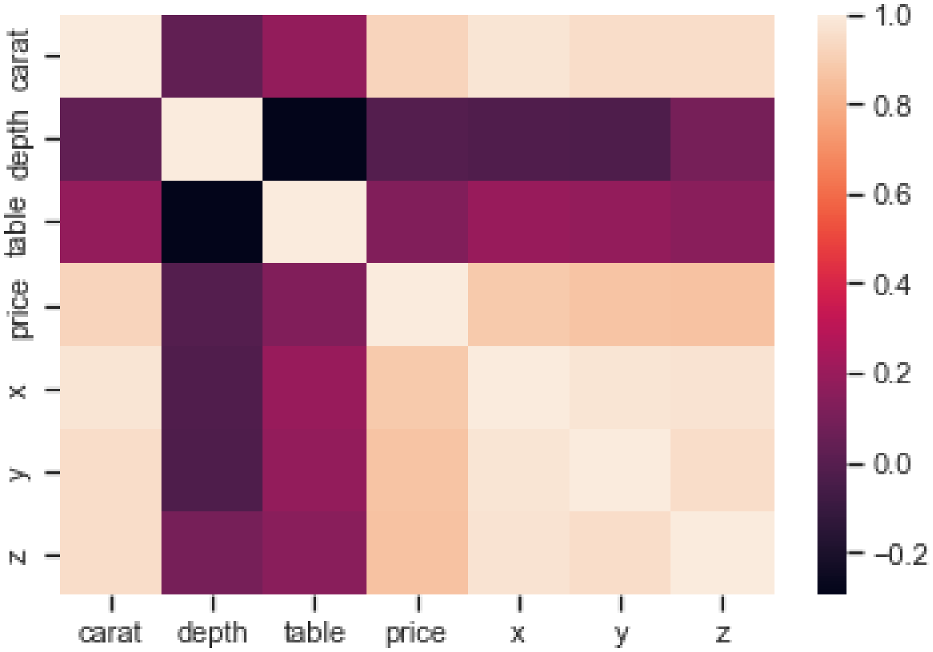


[55]:
```
xyz = df.corr()
xyz
```

[55]:
```
          carat     depth     table     price         x         y         z
carat  1.000000  0.028224  0.181618  0.921591  0.975094  0.951722  0.953387
depth  0.028224  1.000000 -0.295779 -0.010647 -0.025289 -0.029341  0.094924
table  0.181618 -0.295779  1.000000  0.127134  0.195344  0.183760  0.150929
price  0.921591 -0.010647  0.127134  1.000000  0.884435  0.865421  0.861249
x      0.975094 -0.025289  0.195344  0.884435  1.000000  0.974701  0.970772
y      0.951722 -0.029341  0.183760  0.865421  0.974701  1.000000  0.952006
z      0.953387  0.094924  0.150929  0.861249  0.970772  0.952006  1.000000
```

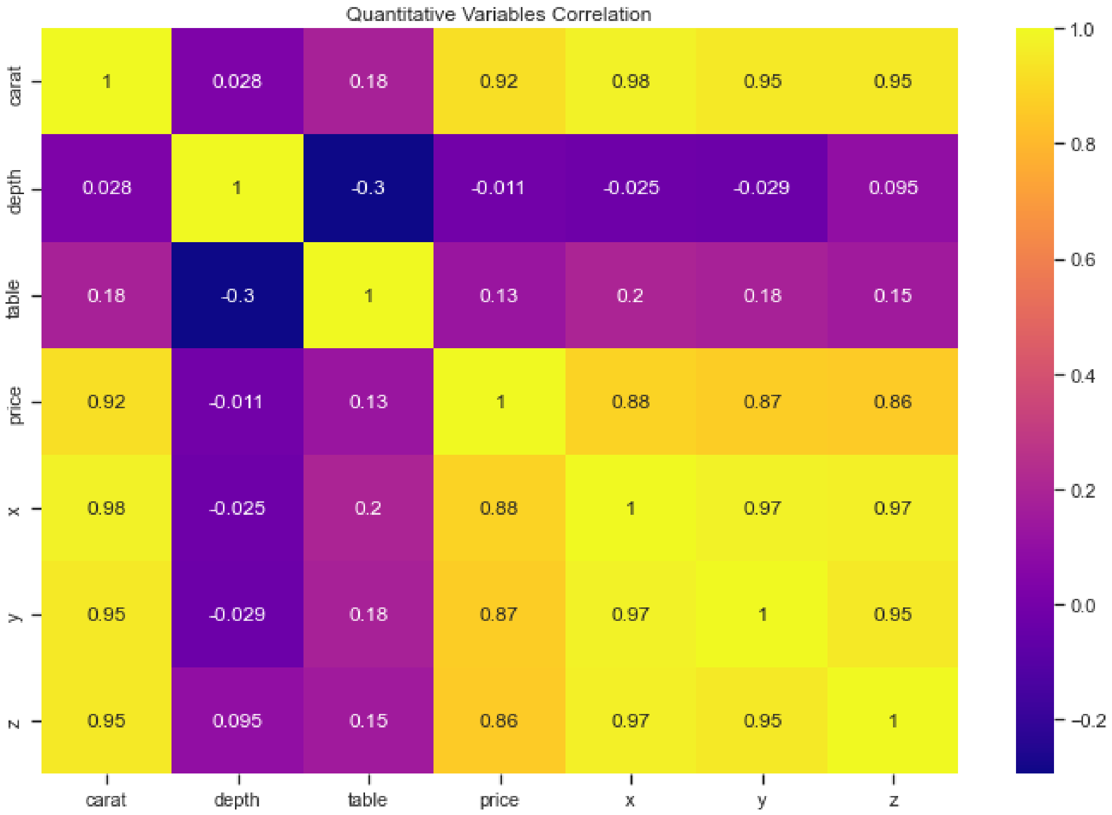
[56]:
```
sns.heatmap(xyz, annot=False)
```

[56]:
```
<AxesSubplot:>
```



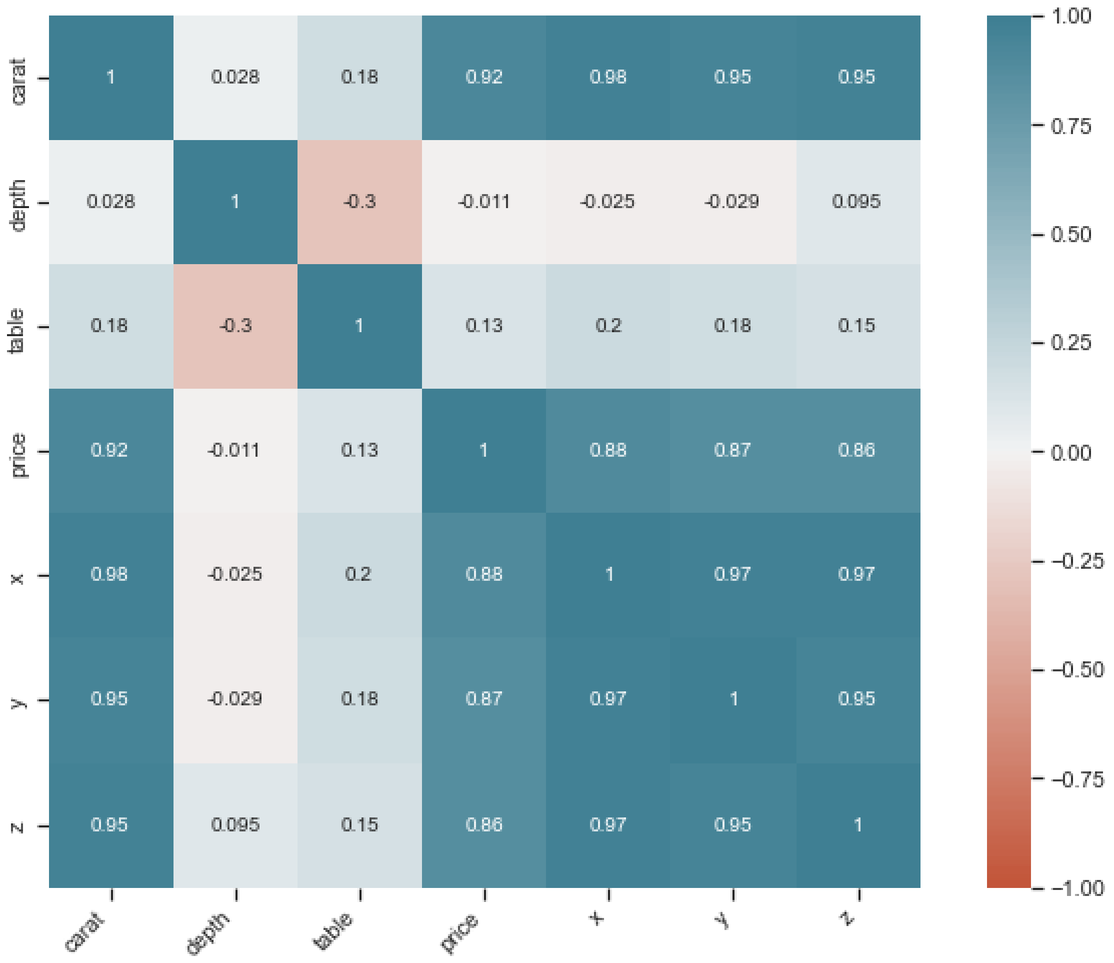
[57]:
```
# Calculate correlations
corr = df.corr()
plt.figure(figsize=(12,8))
plt.title('Quantitative Variables Correlation')

# Heatmap
sns.heatmap(corr,cmap='plasma',annot=True)
```

[57]:
```
<AxesSubplot:title={'center':'Quantitative Variables Correlation'}>
```



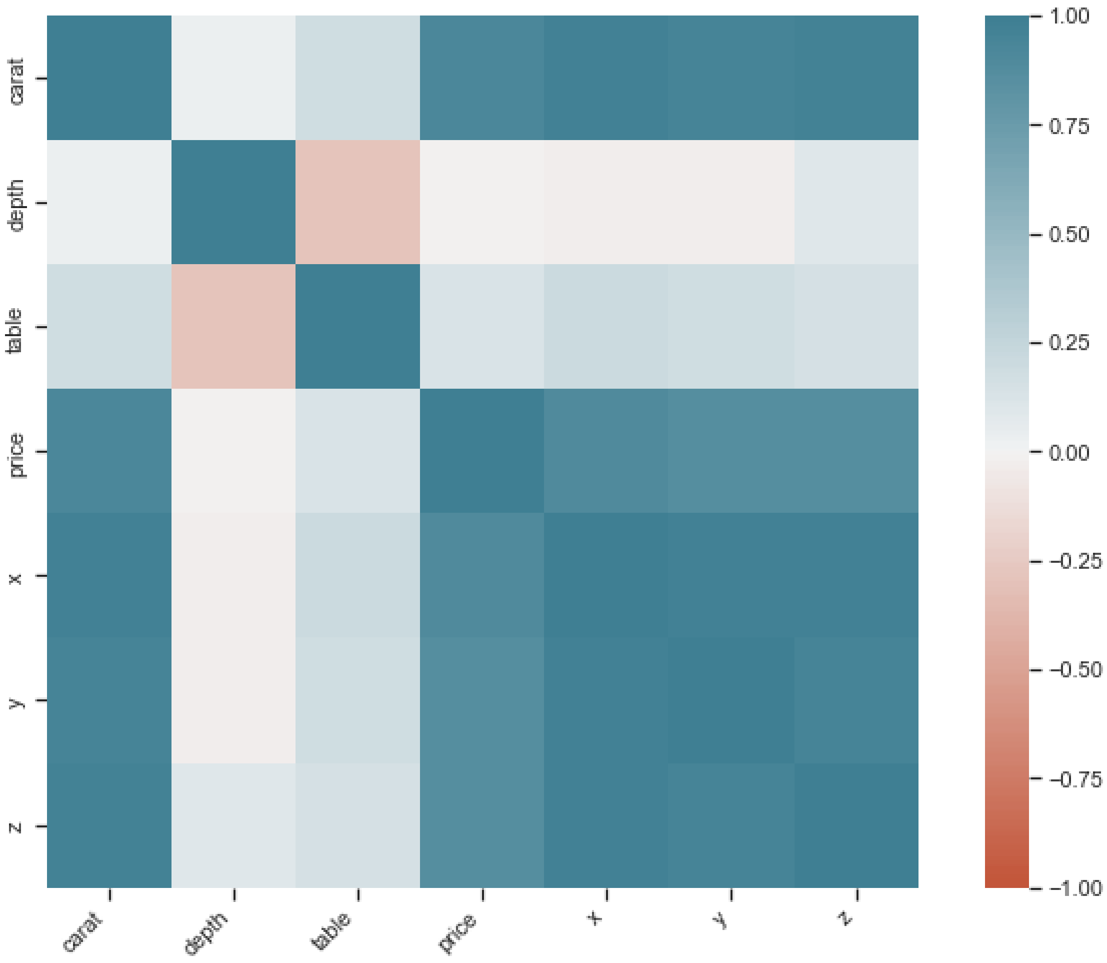
[58]:

```
plt.figure(figsize=(12,8))
corr = df.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True,
    annot=True, annot_kws={"size":10}
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
```



[59]:

```
plt.figure(figsize=(12,8))
corr = df.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
    square=True,
    annot=False, annot_kws={"size":20}
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
```


carat
depth
table
price
x
y
z
carat
depth
table
price
x
y
z
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00