Exploring Advanced Seaborn Techniques: A Comprehensive Tutorial

- Seaborn, a data visualization library built on Matplotlib, is renowned for its appealing visualizations and seamless integration with Pandas. In contrast to Matplotlib, where creating plots often involves multiple lines of code, Seaborn simplifies the process by making informed assumptions, allowing you to achieve the same plot with just one line of code.
    - To install Seaborn, you can use the Anaconda Environment tab or execute the following command in your terminal:

-- pip install seaborn

or

-- conda install seaborn

For a quick overview of available options, you can use 'Shift + Tab' after an attribute.

import

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


%matplotlib inline

# Auto reloads notebook when changes are made
%reload_ext autoreload
%autoreload 2
```

In [ ]:
```
import urllib.request
from urllib.request import urlopen
import ssl
import json
ssl._create_default_https_context = ssl._create_unverified_context
```

Import Data

In [ ]:
```
# You can import custom data
cs_df = pd.read_csv('ComputerSales.csv')

# Seaborn provides built in datasets
print(sns.get_dataset_names())

# Load a built in dataset based on US State car crash percentages
crash_df = sns.load_dataset('car_crashes')
```

```
['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamo
nds', 'dots', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'h
```

```
ealthexp', 'iris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips',
'titanic']
```

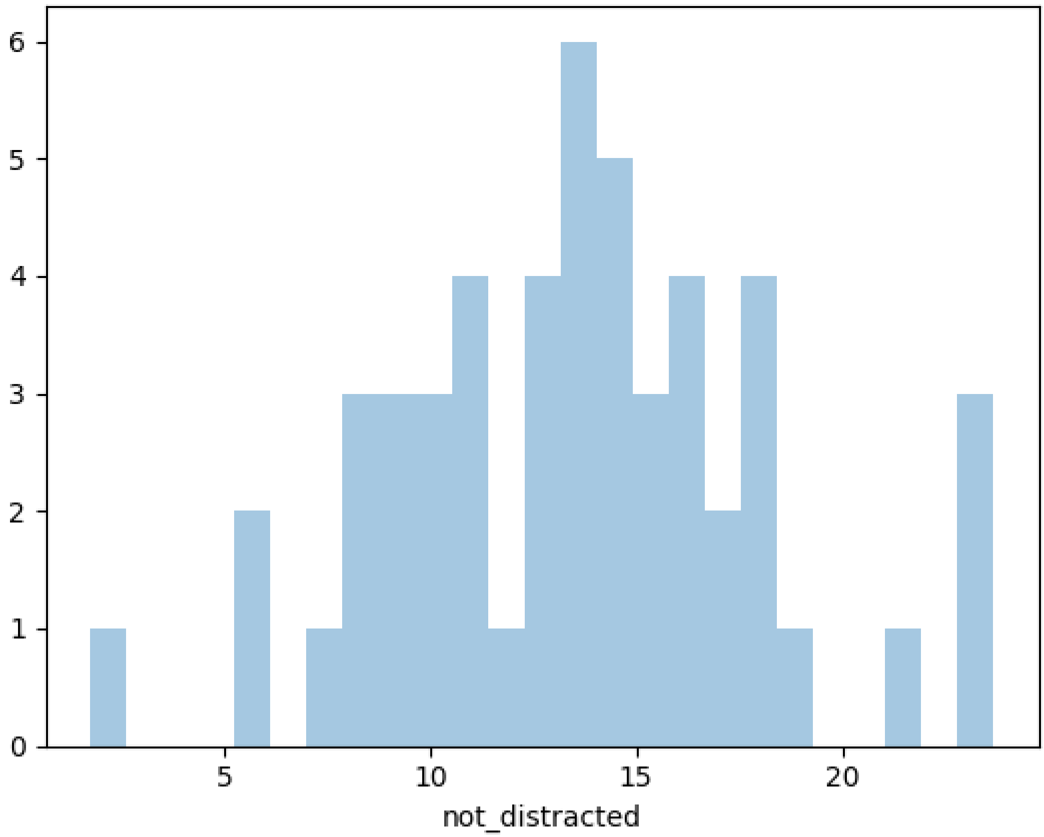
Distribution Plots

In [ ]:

```python
# Provides a way to look at a univariate distribution. A
# univeriate distribution provides a distribution for one variable
# Kernal Density Estimation with a Histogram is provided
# kde=False removes the KDE
# Bins define how many buckets to divide the data up into between intervals
# For example put all profits between $10 and $20 in this bucket
sns.distplot(crash_df['not_distracted'], kde=False, bins=25)
```

```
/var/folders/26/pvvz5dxx7lb978b1_d113_r40000gn/T/ipykernel_5290/777783273.py:
7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(crash_df['not_distracted'], kde=False, bins=25)
```

Out[ ]: `<Axes: xlabel='not_distracted'>`



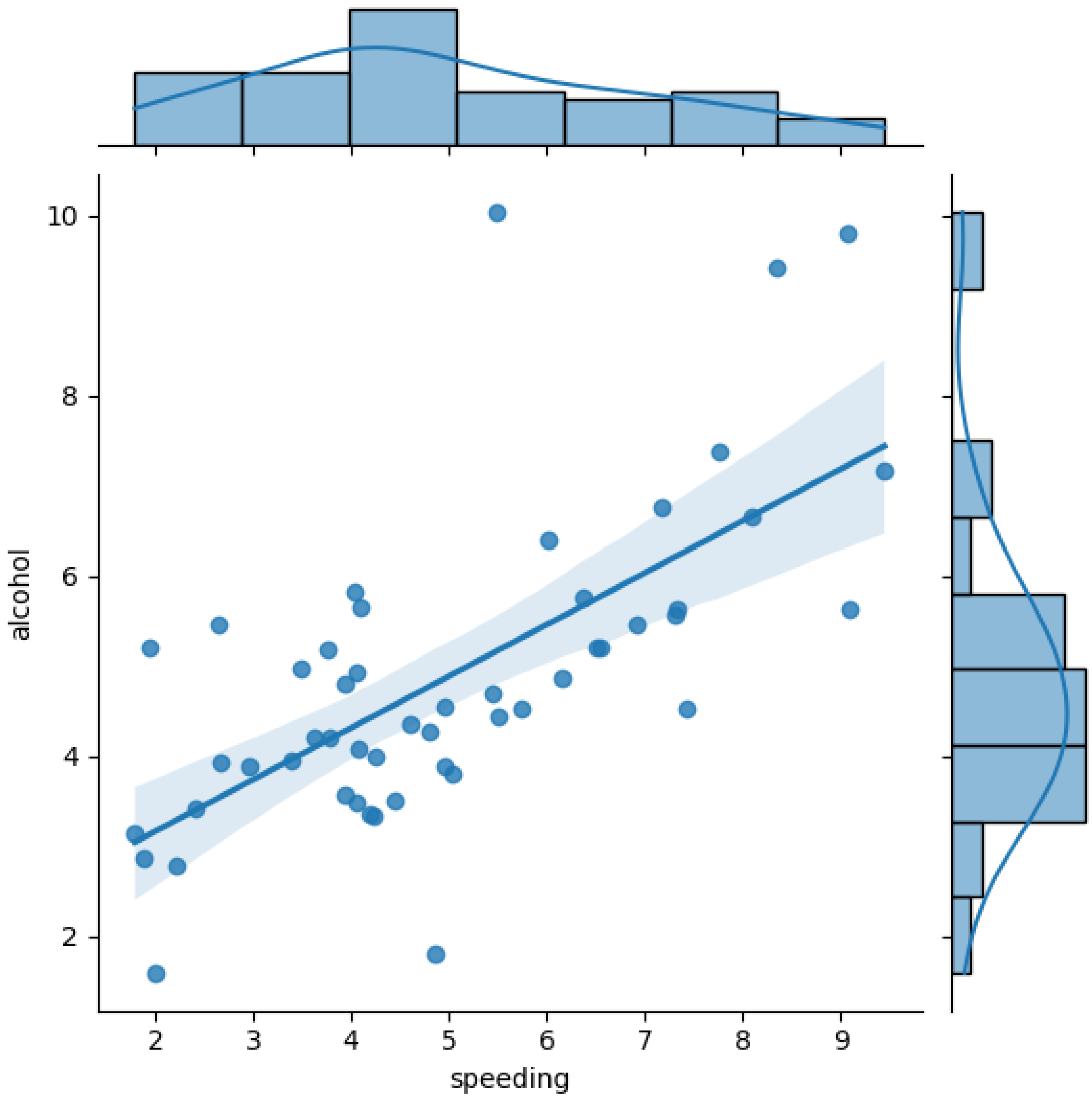
Joint Plot

In [ ]:

```python
# Jointplot compares 2 distributions and plots a scatter plot by default
# As we can see as people tend to speed they also tend to drink & drive
# With kind you can create a regression line with kind='reg'
# You can create a 2D KDE with kind='kde'
# Kernal Density Estimation estimates the distribution of data
```

```
# You can create a hexagon distribution with kind='hex'
sns.jointplot(x='speeding', y='alcohol', data=crash_df, kind='reg')
```

Out[ ]: <seaborn.axisgrid.JointGrid at 0x154845270>

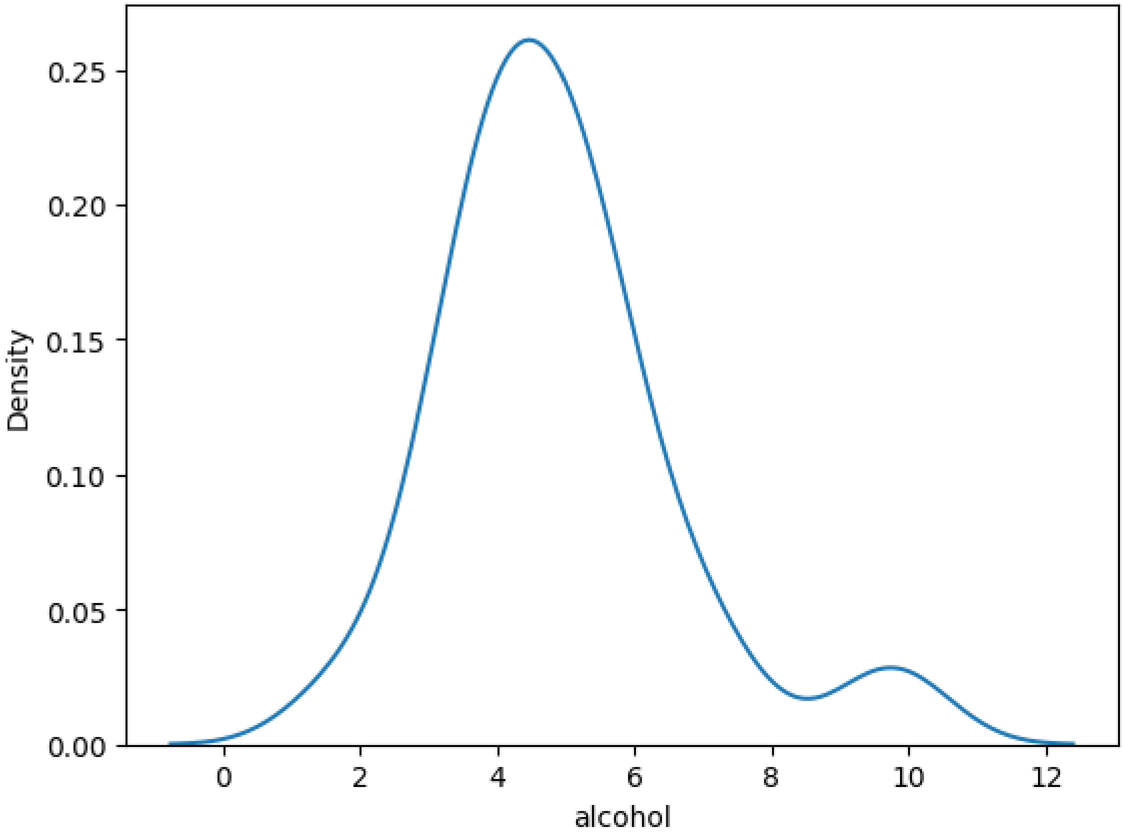


KDE Plot

In [ ]:

```
# Get just the KDE plot
sns.kdeplot(crash_df['alcohol'])
```

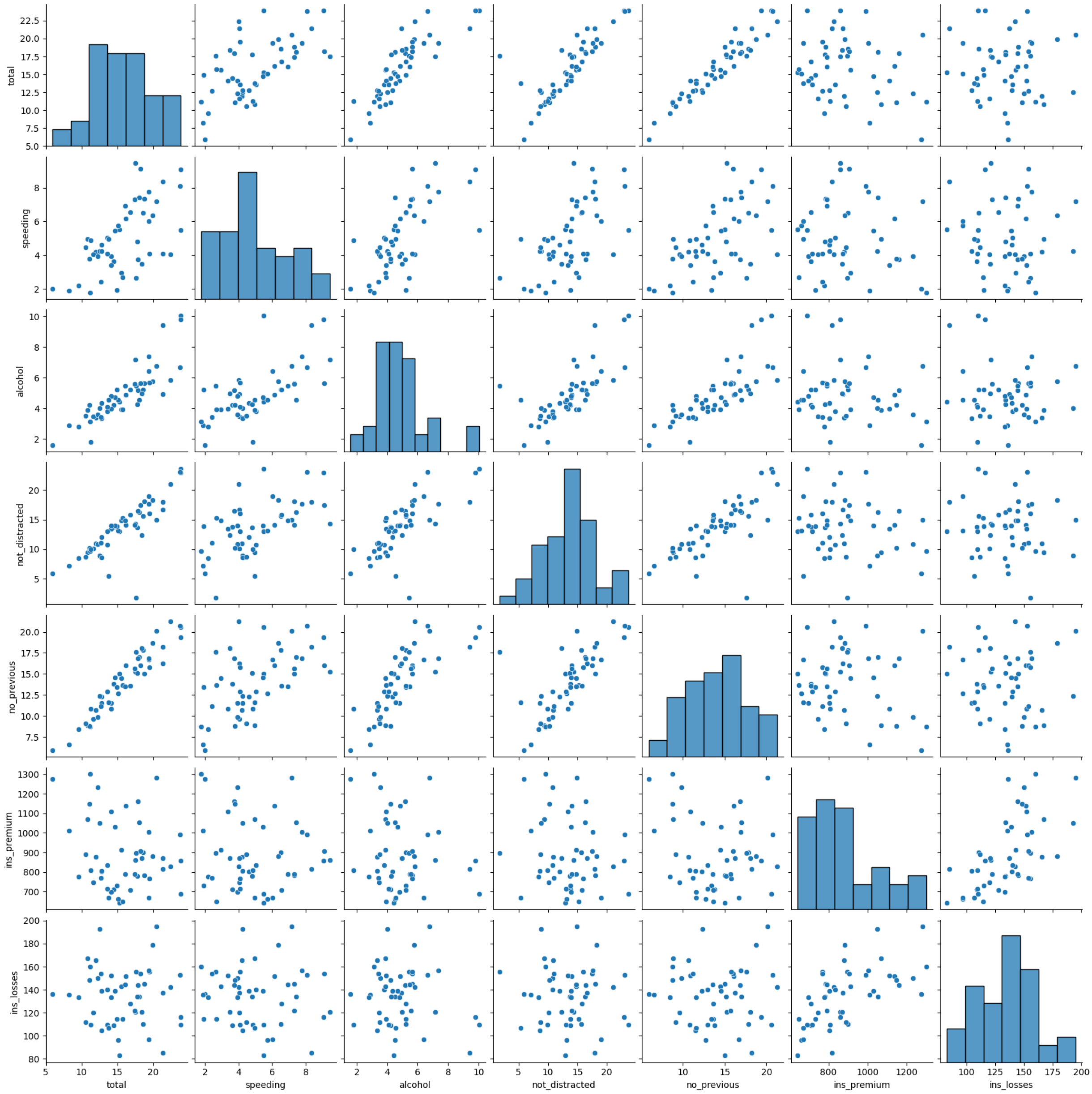
Out[ ]: <Axes: xlabel='alcohol', ylabel='Density'>



Pair Plots

In [ ]:

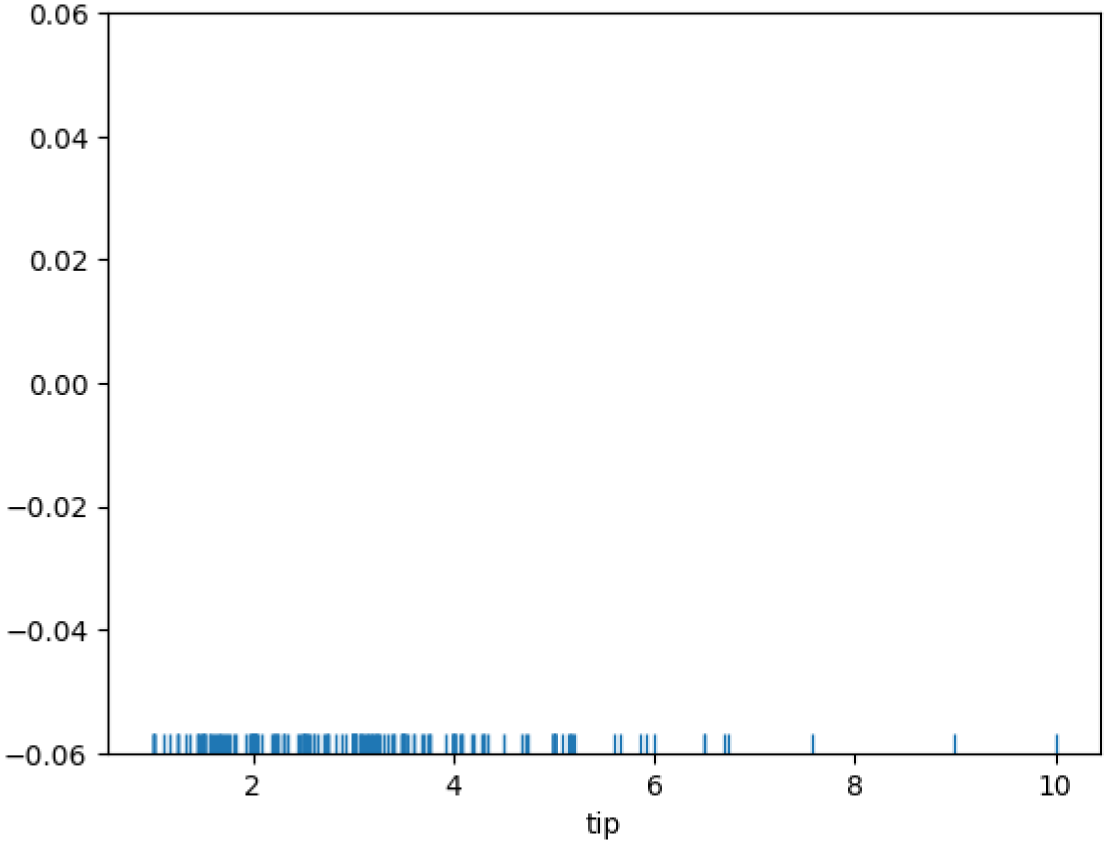
```
# Pair Plot plots relationships across the entire data frames numerical value
sns.pairplot(crash_df)

# Load data on tips
tips_df = sns.load_dataset('tips')

# With hue you can pass in a categorical column and the charts will be colori
# You can use color maps from Matplotlib to define what colors to use
# sns.pairplot(tips_df, hue='sex', palette='Blues')
```



Rug Plots

In [ ]:
```
# Plots a single column of datapoints in an array as sticks on an axis
# With a rug plot you'll see a more dense number of lines where the amount is
# most common. This is like how a histogram is taller where values are more c
sns.rugplot(tips_df['tip'])
```

Out[ ]: <Axes: xlabel='tip'>

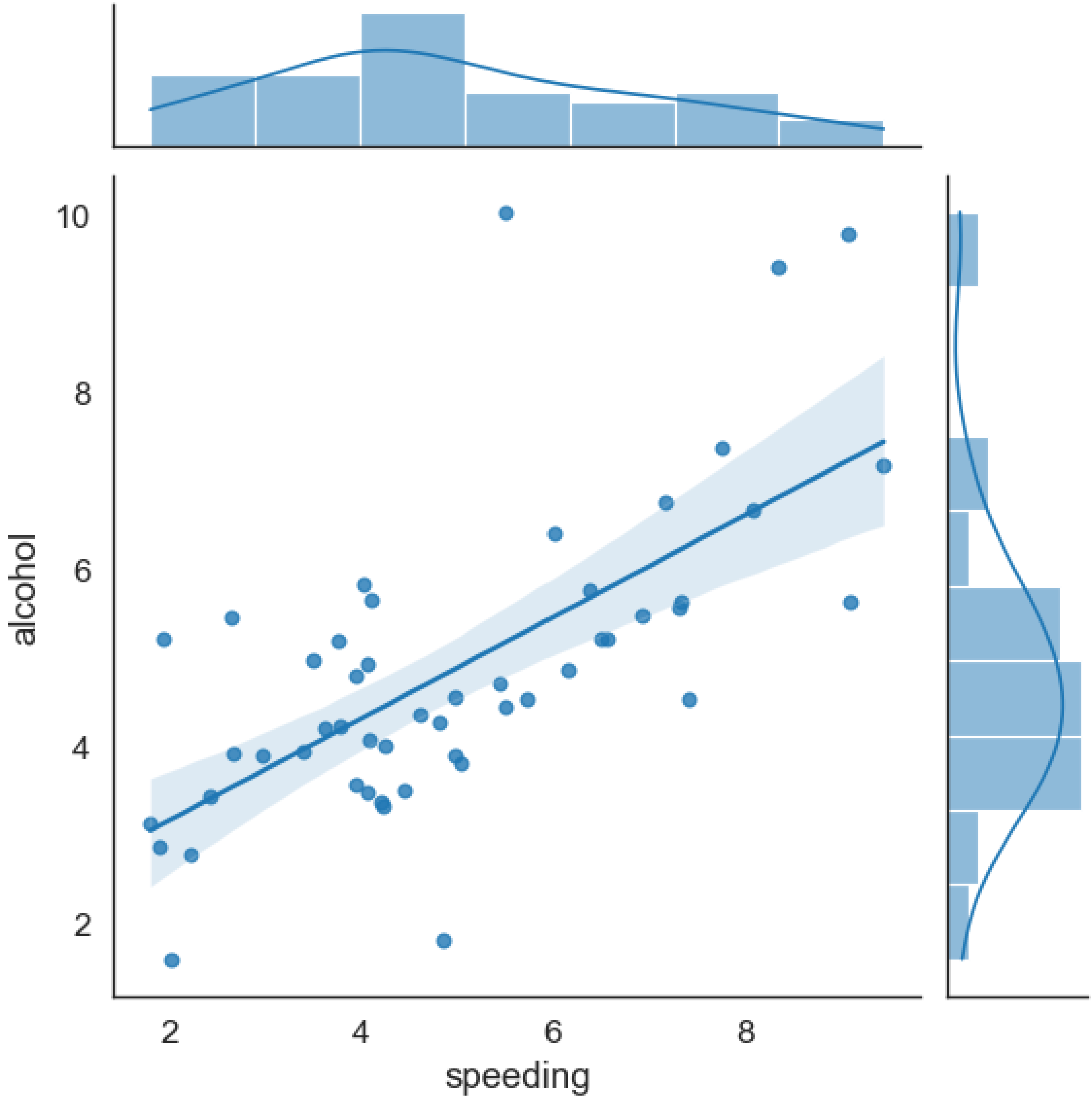


Styling

In [ ]:

```
# You can set styling for your axes and grids
# white, darkgrid, whitegrid, dark, ticks
sns.set_style('white')

# You can use figure sizing from Matplotlib
plt.figure(figsize=(8,4))

# Change size of lables, lines and other elements to best fit
# how you will present your data (paper, talk, poster)
sns.set_context('paper', font_scale=1.4)

sns.jointplot(x='speeding', y='alcohol', data=crash_df, kind='reg')

# Get rid of spines
# You can turn of specific spines with right=True, left=True
# bottom=True, top=True
sns.despine(left=False, bottom=False)
```

```
<Figure size 800x400 with 0 Axes>
```



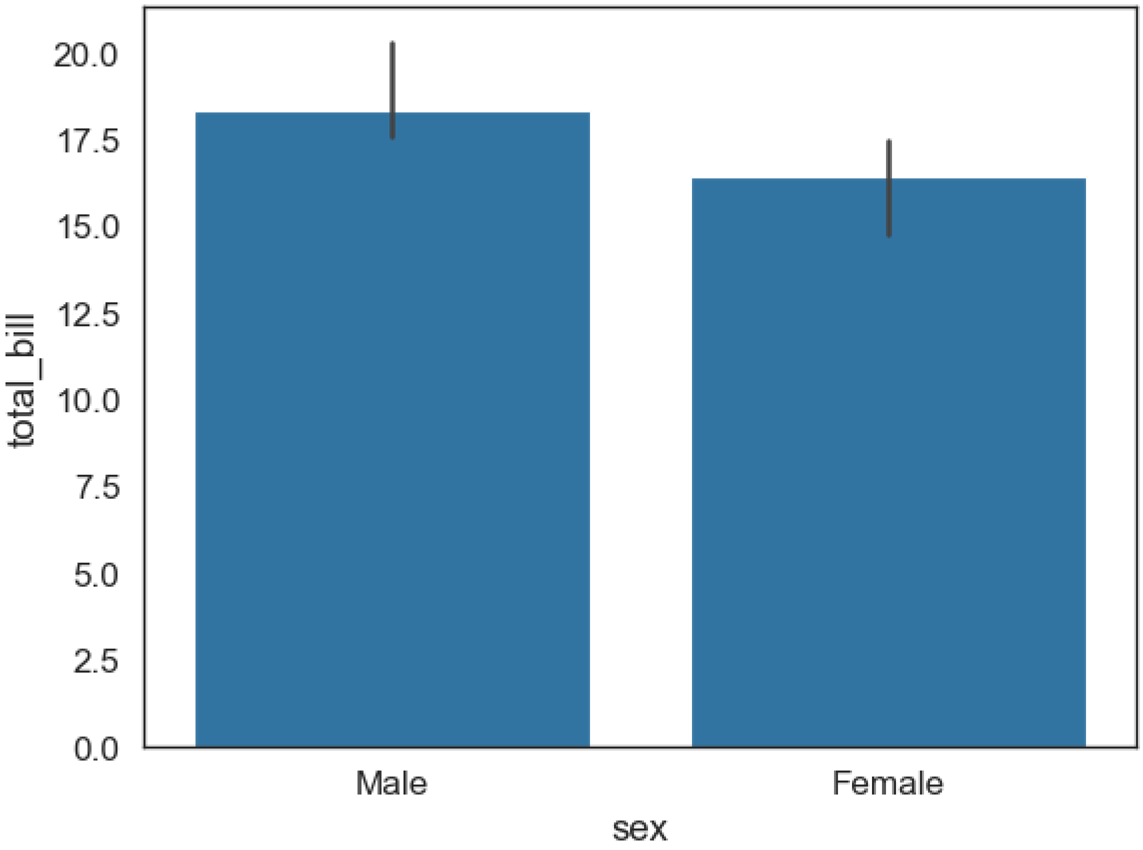
Categorical Plots

Bar Plots

In [ ]:

```
# Focus on distributions using categorical data in reference to one of the nu
# columns

# Aggregate categorical data based on a function (mean is the default)
# Estimate total bill amount based on sex
# With estimator you can define functions to use other than the mean like tho
# provided by NumPy : median, std, var, cov or make your own functions
sns.barplot(x='sex',y='total_bill',data=tips_df, estimator=np.median)
```

Out[ ]: <Axes: xlabel='sex', ylabel='total_bill'>

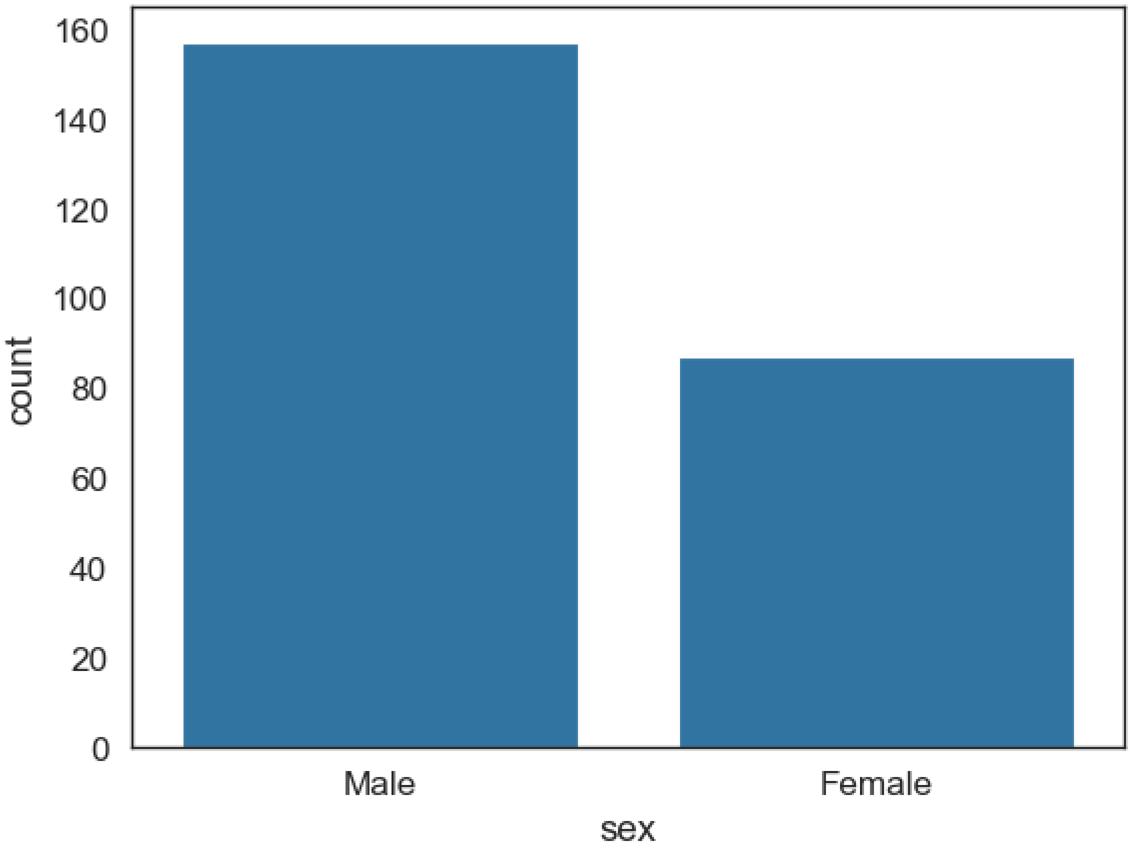


Count Plot

In [ ]:

```
# A count plot is like a bar plot, but the estimator is counting
# the number of occurances
sns.countplot(x='sex',data=tips_df)
```

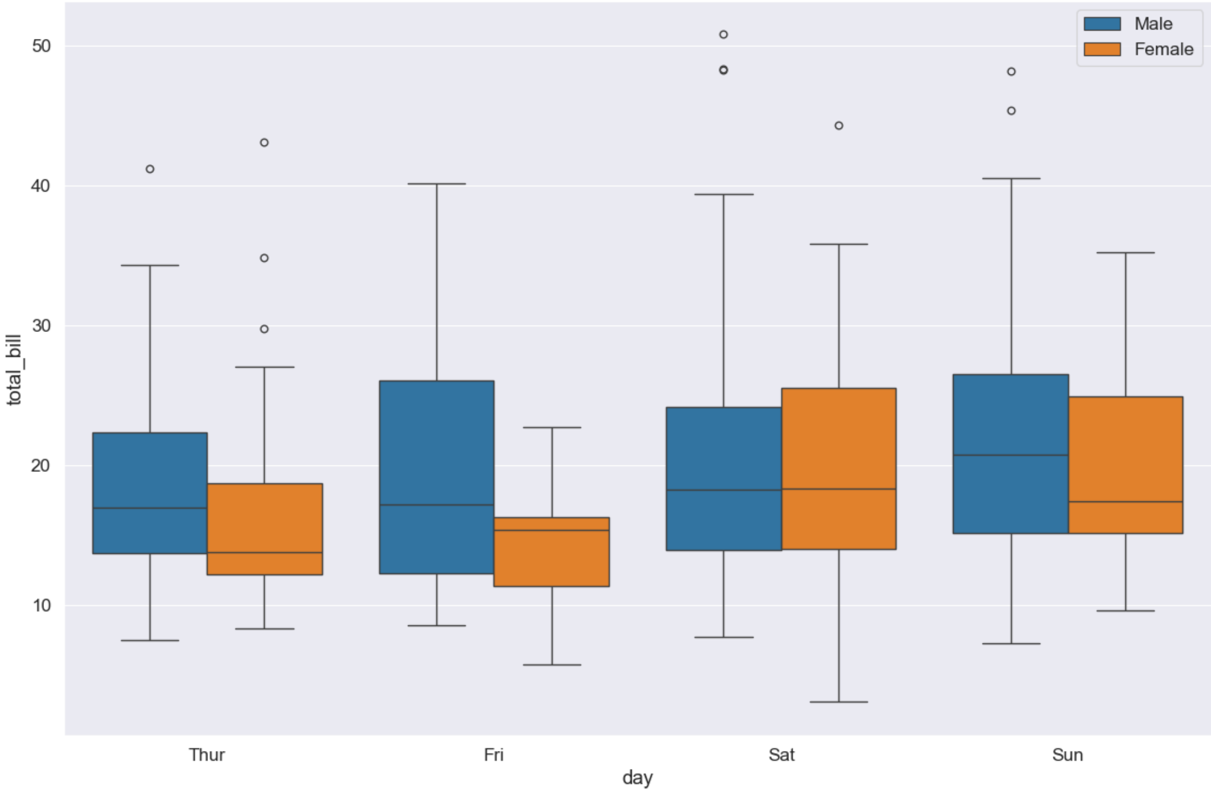
Out[ ]: <Axes: xlabel='sex', ylabel='count'>



Box Plot

In [ ]:
```
plt.figure(figsize=(14,9))
sns.set_style('darkgrid')

# A box plot allows you to compare different variables
# The box shows the quartiles of the data. The bar in the middle is the media
# the box extends 1 standard deviation from the median
# The whiskers extend to all the other data aside from the points that are co
# to be outliers
# Hue can add another category being sex
# We see men spend way more on Friday versus less than women on Saturday
sns.boxplot(x='day',y='total_bill',data=tips_df, hue='sex')

# Moves legend to the best position
plt.legend(loc=0)
```

Out[ ]: <matplotlib.legend.Legend at 0x16a8bc130>

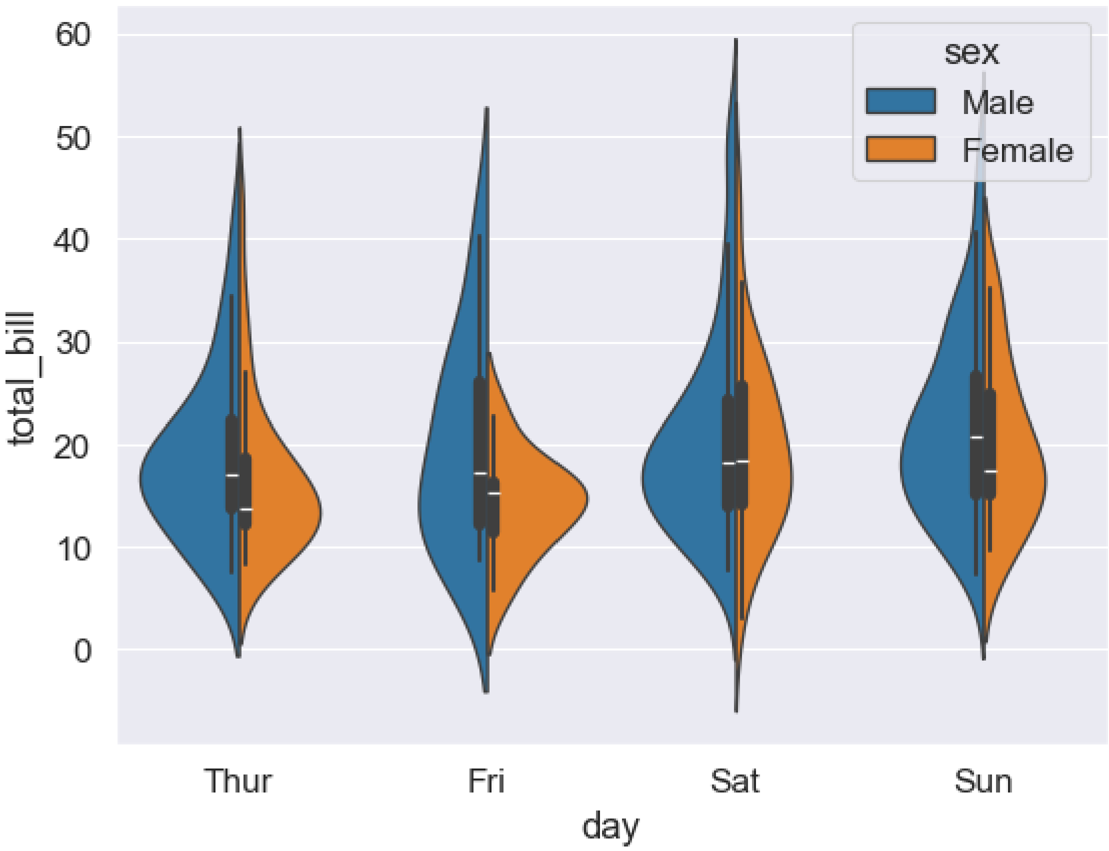


Violin Plot

In [ ]:

```
# Violin Plot is a combination of the boxplot and KDE
# While a box plot corresponds to data points, the violin plot uses the KDE e
# of the data points
# Split allows you to compare how the categories compare to each other
sns.violinplot(x='day',y='total_bill',data=tips_df, hue='sex',split=True)
```

Out[ ]: <Axes: xlabel='day', ylabel='total_bill'>

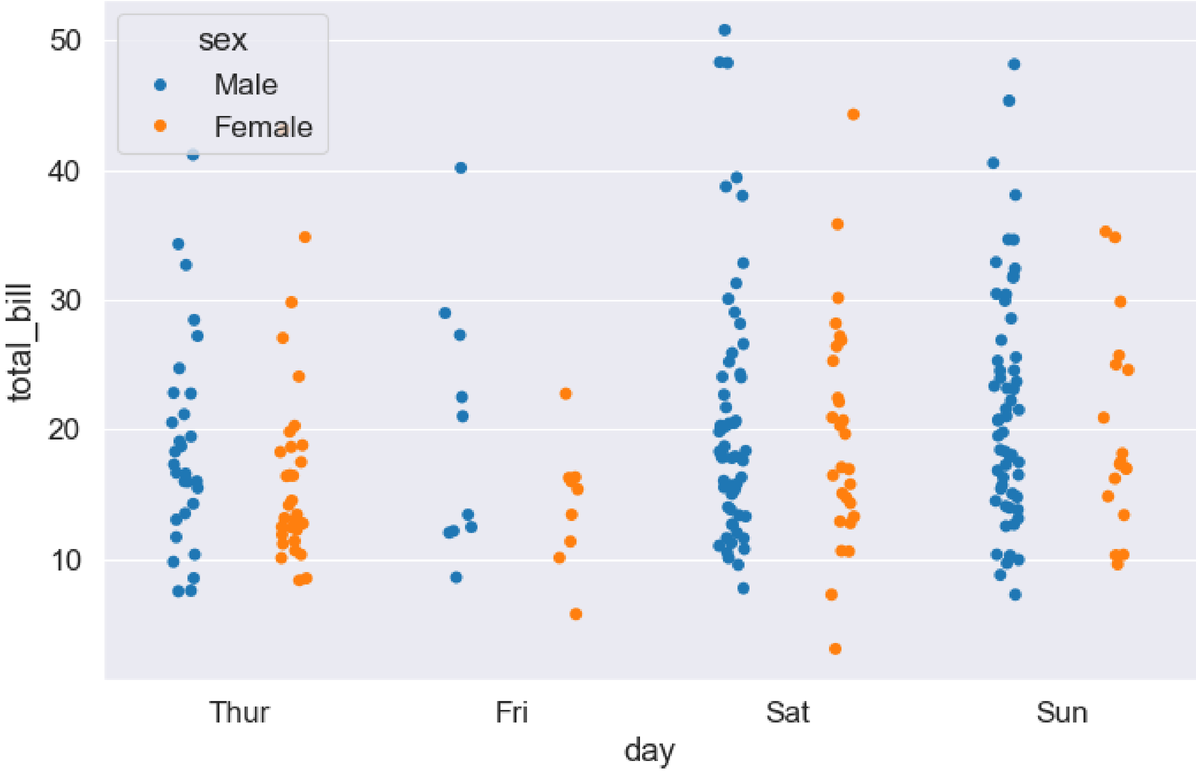


Strip Plot

In [ ]:

```
plt.figure(figsize=(8,5))

# The strip plot draws a scatter plot representing all data points where one
# variable is categorical. It is often used to show all observations with
# a box plot that represents the average distribution
# Jitter spreads data points out so that they aren't stacked on top of each o
# Hue breaks data into men and women
# Dodge separates the men and women data
sns.stripplot(x='day',y='total_bill',data=tips_df, jitter=True,
              hue='sex', dodge=True)
```

Out[ ]: <Axes: xlabel='day', ylabel='total_bill'>

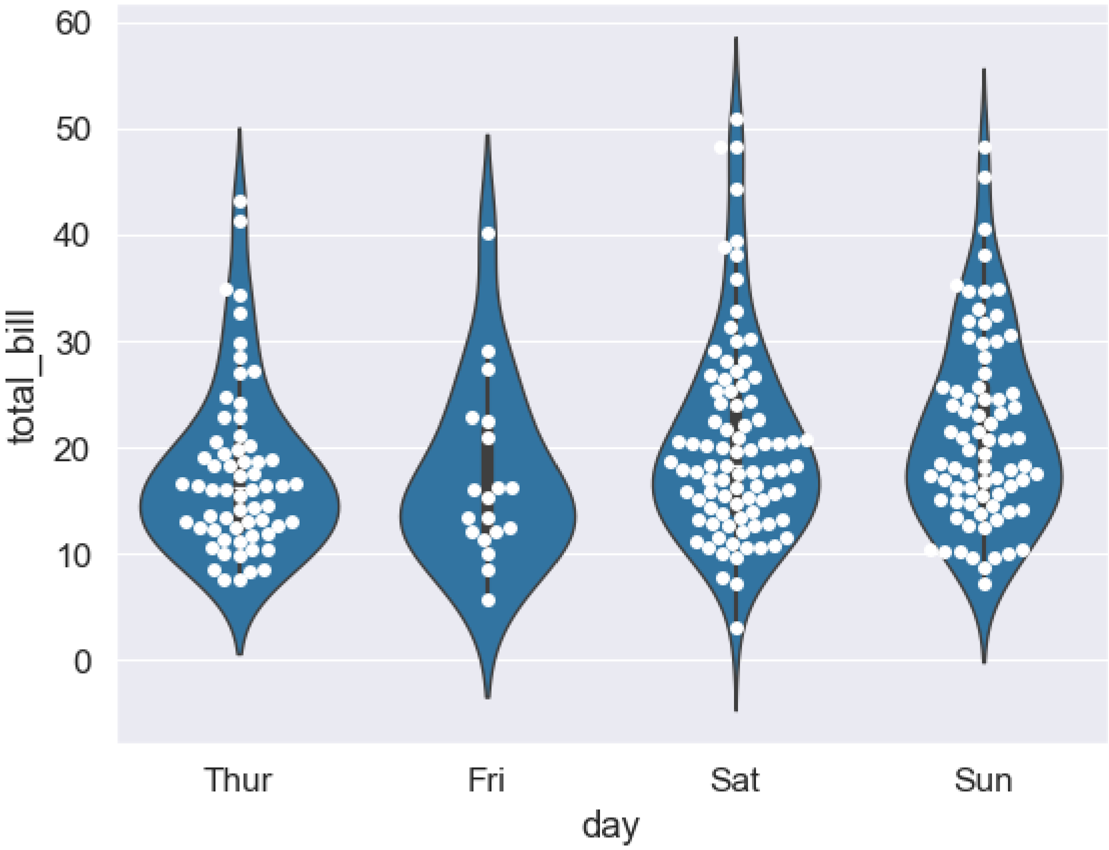


Swarm Plot

In [ ]:

```
# A swarm plot is like a strip plot, but points are adjusted so they don't ov
# It looks like a combination of the violin and strip plots
# sns.swarmplot(x='day',y='total_bill',data=tips_df)

# You can stack a violin plot with a swarm
sns.violinplot(x='day',y='total_bill',data=tips_df)
sns.swarmplot(x='day',y='total_bill',data=tips_df, color='white')
```

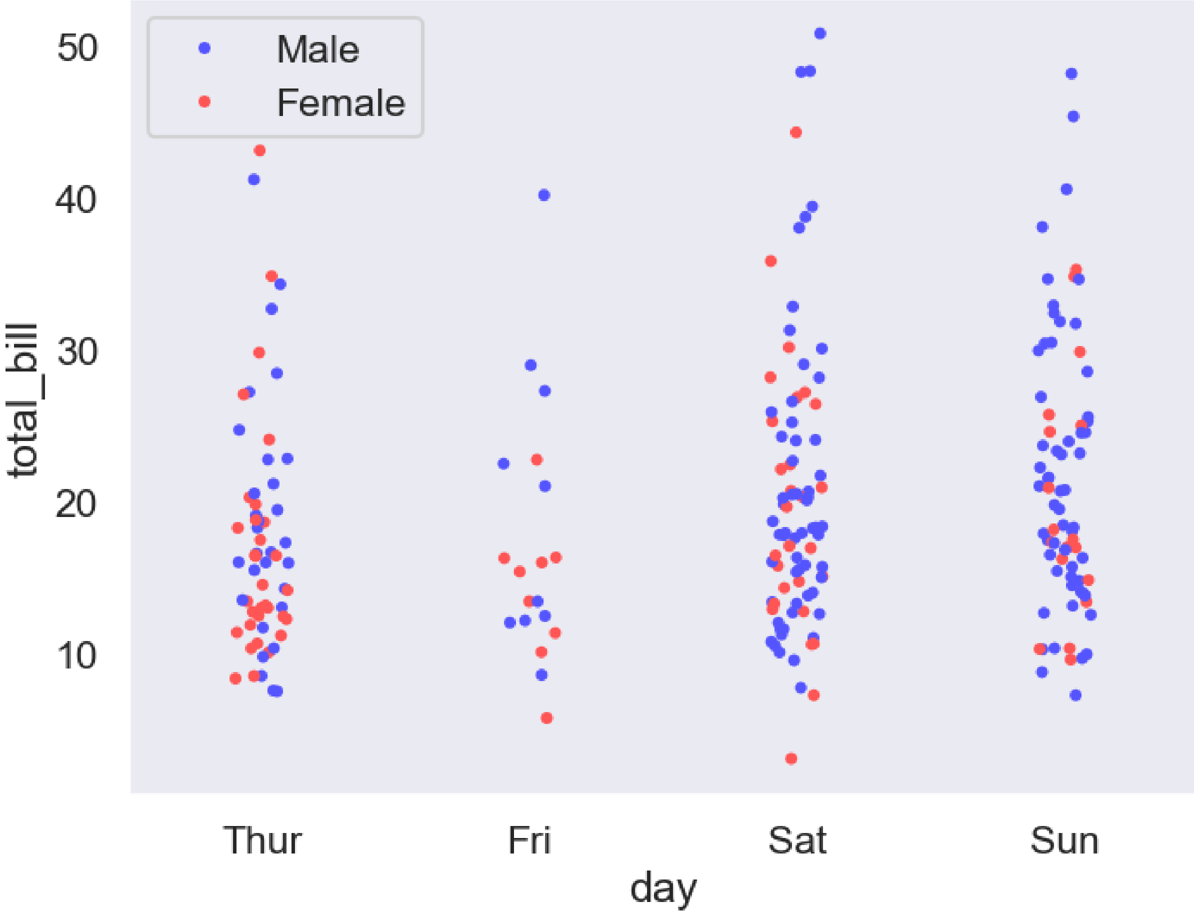
Out[ ]: <Axes: xlabel='day', ylabel='total_bill'>



Palettes

In [ ]:
```
plt.figure(figsize=(8,6))

sns.set_style('dark')

sns.set_context('talk')

# You can use Matplotlibs color maps for color styling
# https://matplotlib.org/3.3.1/tutorials/colors/colormaps.html
sns.stripplot(x='day',y='total_bill',data=tips_df, hue='sex',
             palette='seismic')

# Add the optional legend with a location number (best: 0,
# upper right: 1, upper left: 2, lower left: 3, lower right: 4,
# https://matplotlib.org/3.3.1/api/_as_gen/matplotlib.pyplot.legend.html)
# or supply a tuple of x & y from lower left
plt.legend(loc=0)
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x16c3eb790>
```



Matrix Plots

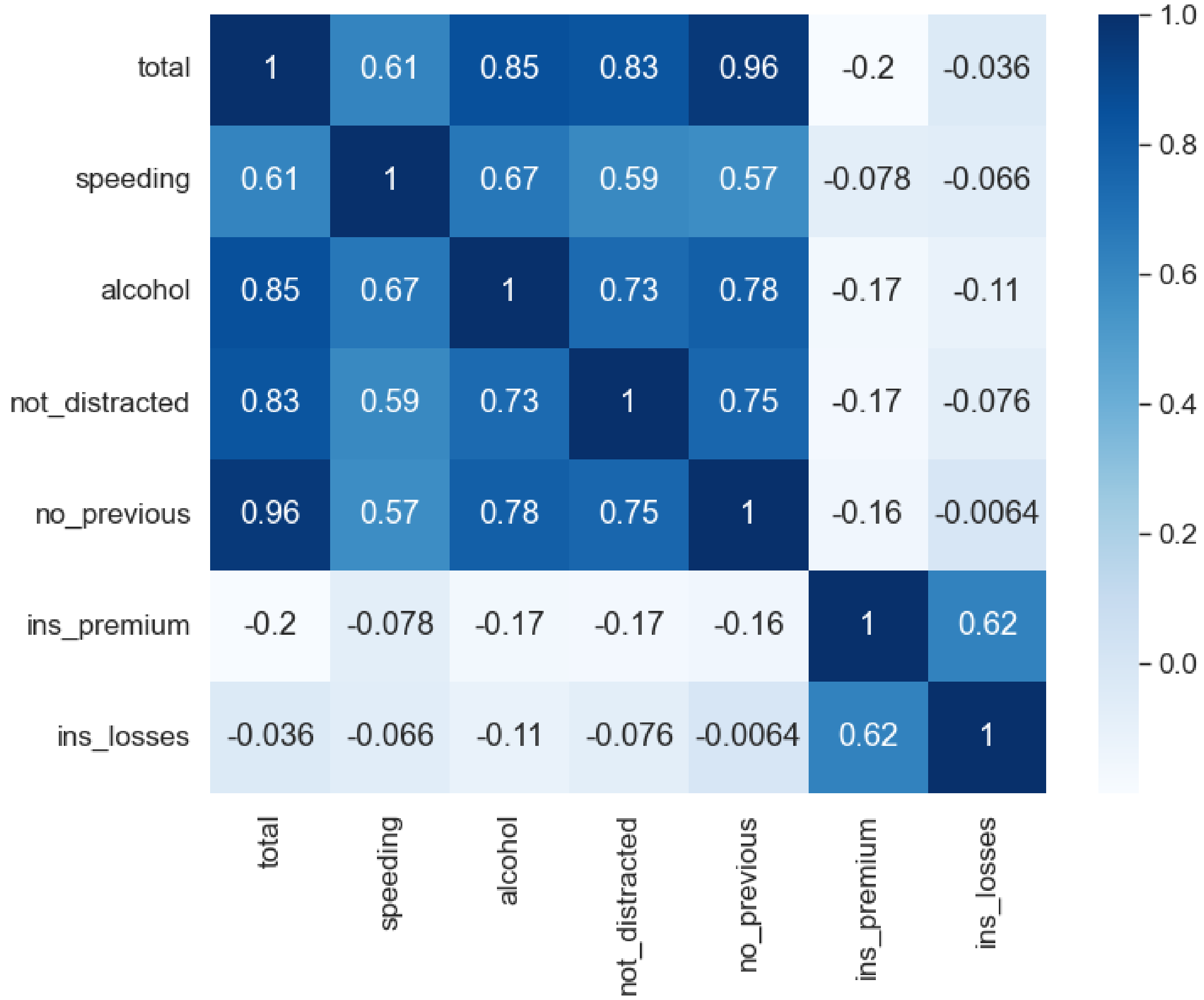
Heatmaps

In [ ]:

```
import os,sys
from scipy import stats
```

In [ ]:

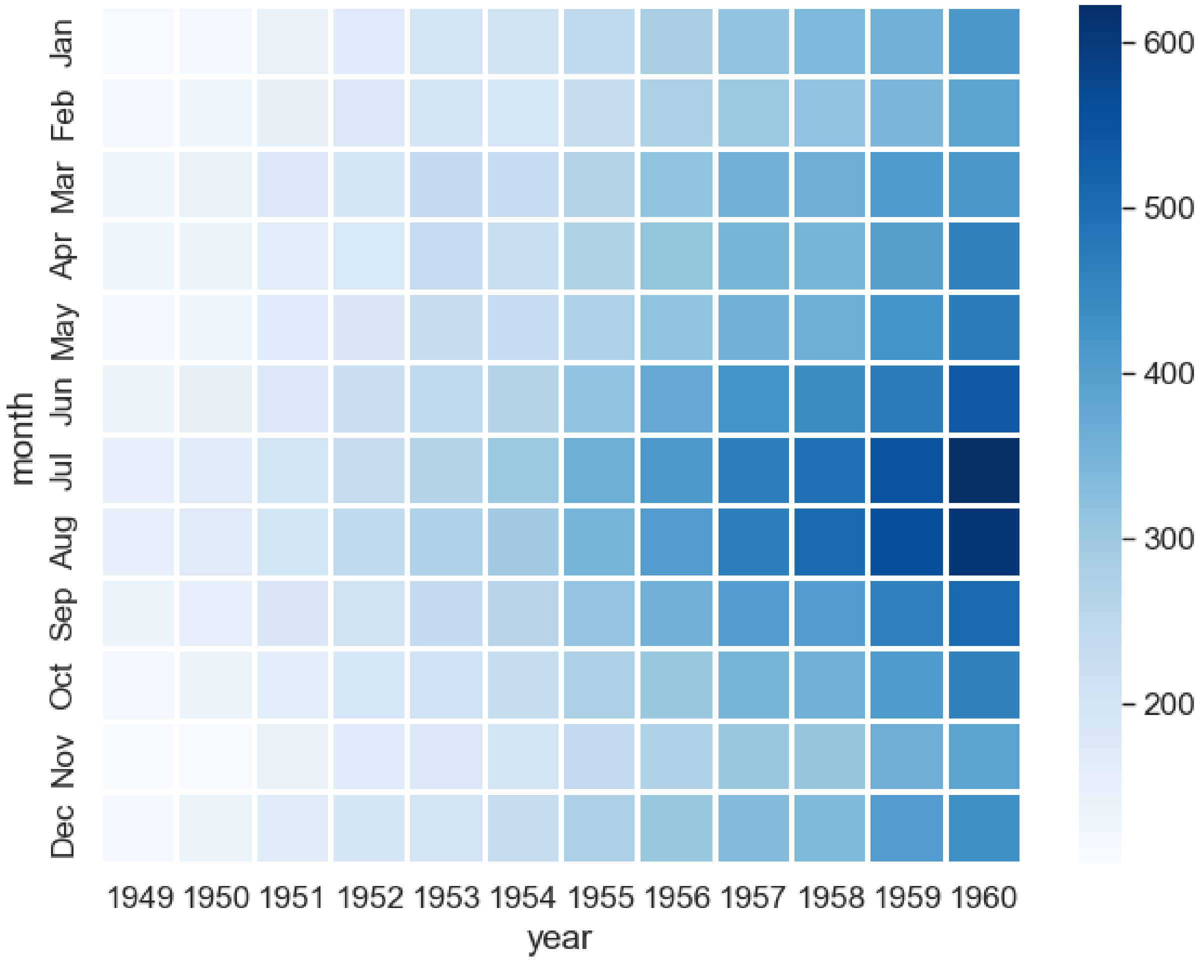
```
# Assuming you have a 'crash_df' DataFrame with mixed data types
# crash_df = ...

# Identify numeric columns in the DataFrame
numeric_columns = crash_df.select_dtypes(include=['float64', 'int64']).column

# Select only numeric columns for correlation calculation
numeric_df = crash_df[numeric_columns]

plt.figure(figsize=(8, 6))
sns.set_context('paper', font_scale=1.4)

# Calculate the correlation matrix for numeric columns
crash_mx = numeric_df.corr()

# Create the heatmap, add annotations, and choose a color map
sns.heatmap(crash_mx, annot=True, cmap='Blues')

# Show the plot
plt.show()
```



In [ ]:

```
plt.figure(figsize=(8,6))
sns.set_context('paper', font_scale=1.4)

# We can create a matrix with an index of month, columns representing years
# and the number of passengers for each
# We see that flights have increased over time and that most people travel in
# July and August
flights = sns.load_dataset("flights")
flights = flights.pivot_table(index='month', columns='year', values='passenge
# You can separate data with lines
sns.heatmap(flights, cmap='Blues', linecolor='white', linewidth=1)
```

Out[ ]: <Axes: xlabel='year', ylabel='month'>



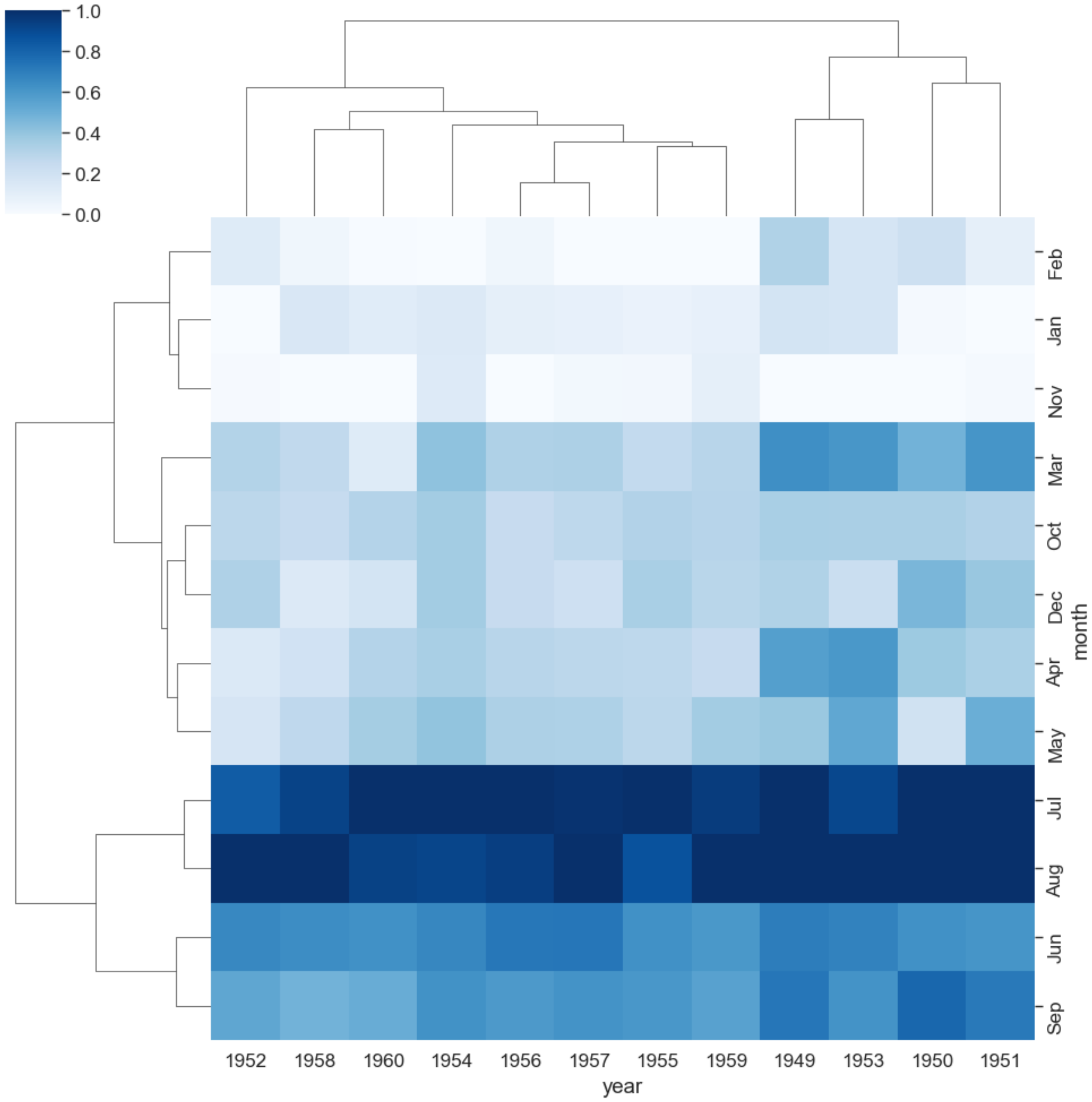
Cluster Map

In [ ]:

```
plt.figure(figsize=(8,6))
sns.set_context('paper', font_scale=1.4)

# A Cluster map is a hierarchically clustered heatmap
# The distance between points is calculated, the closest are joined, and this
# continues for the next closest (It compares columns / rows of the heatmap)
# This is data on iris flowers with data on petal lengths
iris = sns.load_dataset("iris")
# Return values for species
# species = iris.pop("species")
# sns.clustermap(iris)

# With our flights data we can see that years have been reoriented to place
# like data closer together
# You can see clusters of data for July & August for the years 59 & 60
# standard_scale normalizes the data to focus on the clustering
sns.clustermap(flights,cmap="Blues", standard_scale=1)
```

Out[ ]:

```
<seaborn.matrix.ClusterGrid at 0x16c6862f0>
<Figure size 800x600 with 0 Axes>
```



PairGrid

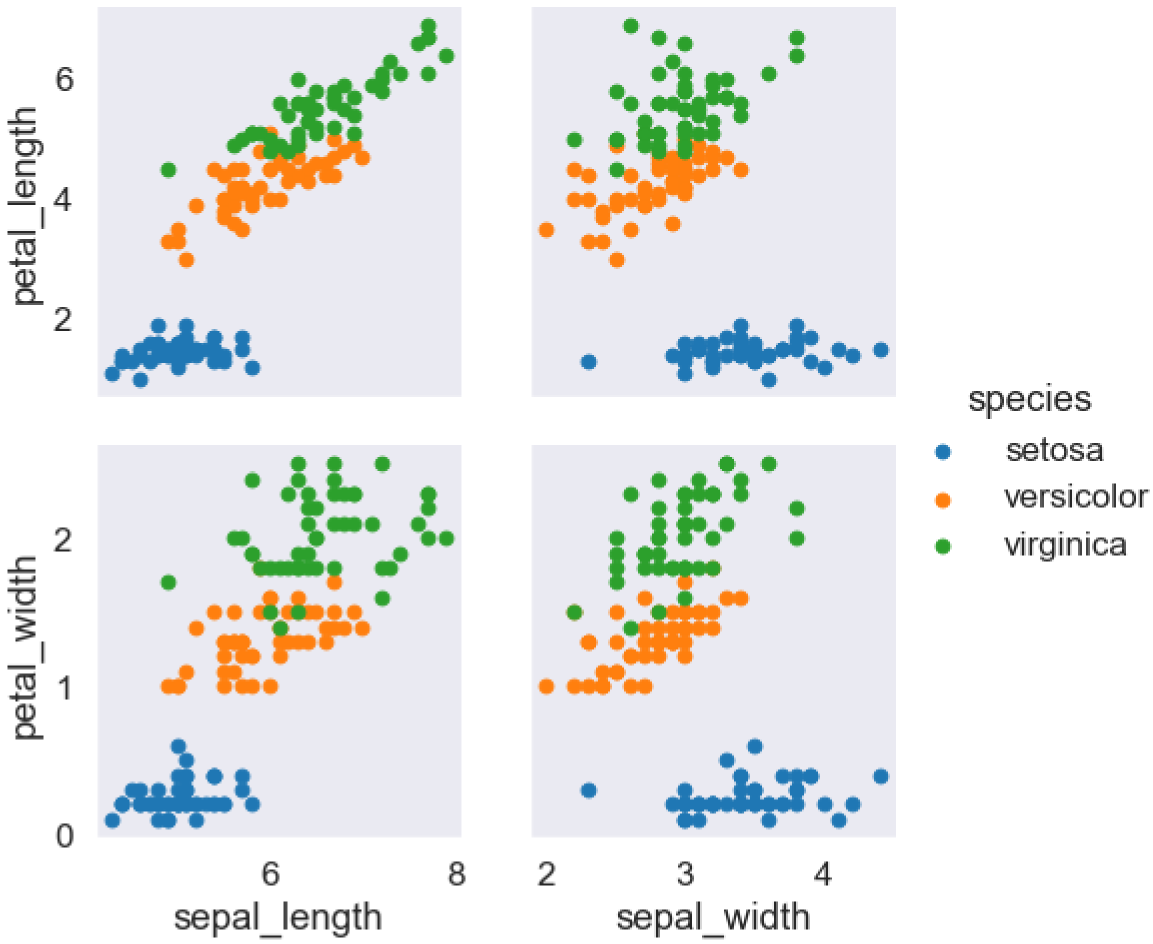
In [ ]:

```
plt.figure(figsize=(8,6))
sns.set_context('paper', font_scale=1.4)

# You can create a grid of different plots with complete control over what is
# Create the empty grid system using the provided data
# Colorize based on species
# iris_g = sns.PairGrid(iris, hue="species")

# Put a scatter plot across the upper, lower and diagonal
# iris_g.map(plt.scatter)

# Put a histogram on the diagonal
# iris_g.map_diag(plt.hist)
# And a scatter plot every place else
# iris_g.map_offdiag(plt.scatter)

# Have different plots in upper, lower and diagonal
# iris_g.map_upper(plt.scatter)
# iris_g.map_lower(sns.kdeplot)

# You can define define variables for x & y for a custom grid
iris_g = sns.PairGrid(iris, hue="species",
```

```
                              x_vars=["sepal_length", "sepal_width"],
                              y_vars=["petal_length", "petal_width"])

iris_g.map(plt.scatter)

# Add a legend last
iris_g.add_legend()
```

Out[ ]: <seaborn.axisgrid.PairGrid at 0x16c629de0>

<Figure size 800x600 with 0 Axes>



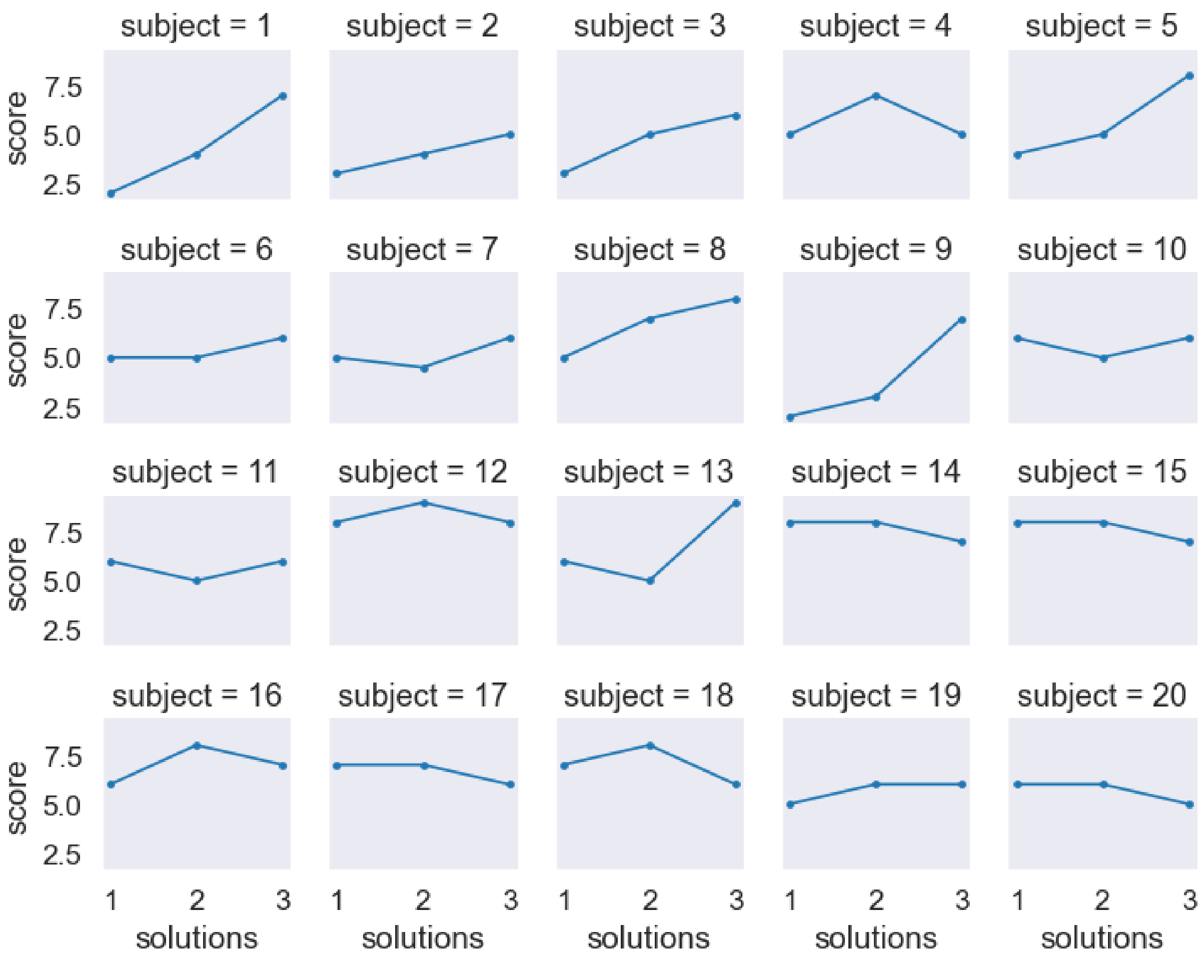
Facet Grid

In [ ]:

```
# Can also print multiple plots in a grid in which you define columns & rows
# Get histogram for smokers and non with total bill for lunch & dinner
# tips_fg = sns.FacetGrid(tips_df, col='time', row='smoker')

# You can pass in attributes for the histogram
# tips_fg.map(plt.hist, "total_bill", bins=8)

# Create a scatter plot with data on total bill & tip (You need to parameters
# tips_fg.map(plt.scatter, "total_bill", "tip")

# We can assign variables to different colors and increase size of grid
# Aspect is 1.3 x the size of height
# You can change the order of the columns
# Define the palette used
# tips_fg = sns.FacetGrid(tips_df, col='time', hue='smoker', height=4, aspect
#                        col_order=['Dinner', 'Lunch'], palette='Set1')
# tips_fg.map(plt.scatter, "total_bill", "tip", edgecolor='w')

# # Define size, linewidth and assign a color of white to markers
```

```
# kws = dict(s=50, linewidth=.5, edgecolor="w")
# # Define that we want to assign different markers to smokers and non
# tips_fg = sns.FacetGrid(tips_df, col='sex', hue='smoker', height=4, aspect=
#                         hue_order=['Yes','No'],
#                         hue_kws=dict(marker=['^', 'v']))
# tips_fg.map(plt.scatter, "total_bill", "tip", **kws)
# tips_fg.add_legend()

# This dataframe provides scores for different students based on the level
# of attention they could provide during testing
att_df = sns.load_dataset("attention")
# Put each person in their own plot with 5 per line and plot their scores
att_fg = sns.FacetGrid(att_df, col='subject', col_wrap=5, height=1.5)
att_fg.map(plt.plot, 'solutions', 'score', marker='.')
```

Out[ ]: `<seaborn.axisgrid.FacetGrid at 0x16caad480>`



Regression Plots

In [ ]:

```
# lmplot combines regression plots with facet grid
tips_df = sns.load_dataset('tips')
tips_df.head()
```

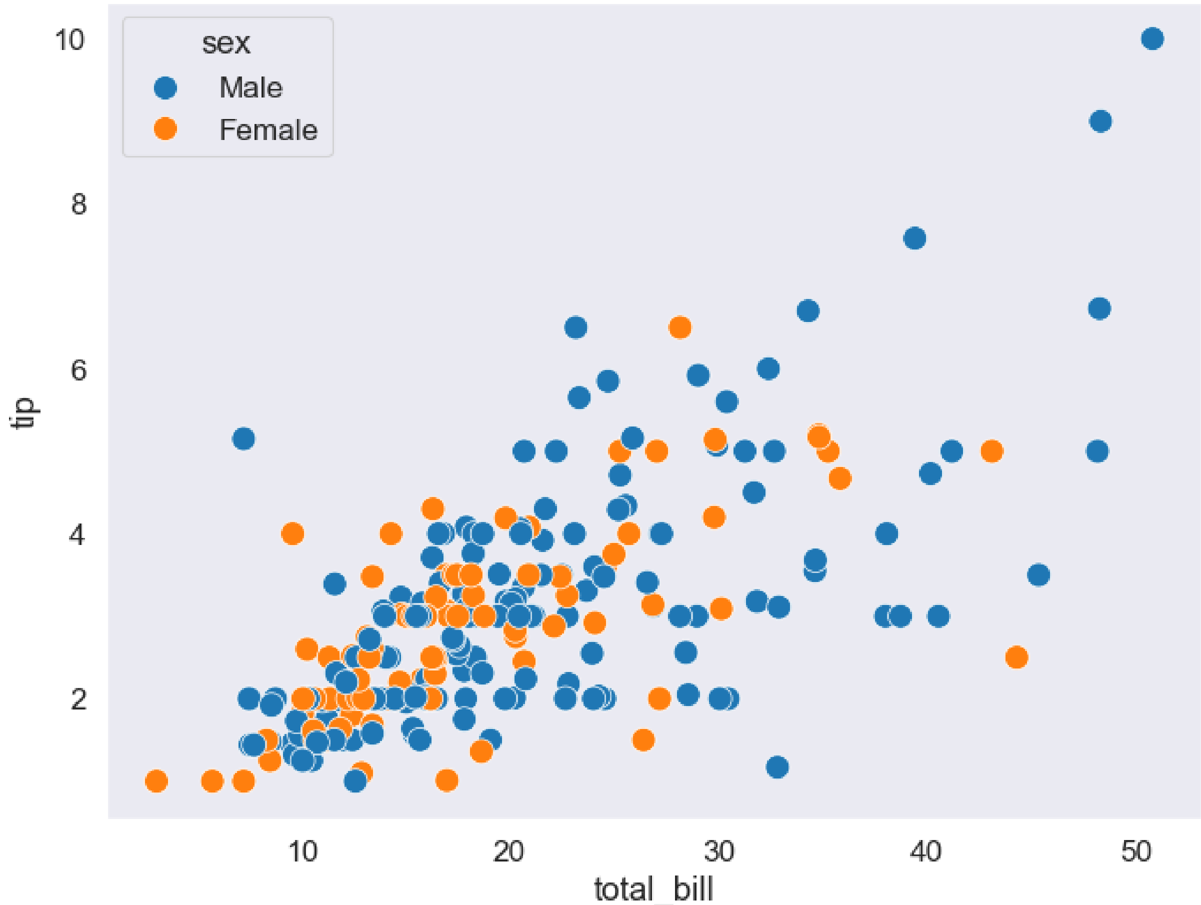
Out[ ]:

	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

	total_bill	tip	sex	smoker	day	time	size
4	24.59	3.61	Female	No	Sun	Dinner	4

In [ ]:

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

# Assuming you have a 'tips_df' DataFrame
# tips_df = ...

plt.figure(figsize=(8, 6))

# Create a scatter plot with a regression line to study the relationship betw
# Use 'hue' to show the separation based on the categorical data 'sex'
# Different markers ('o' for females, '^' for males) are defined for each cat
# Additional styling of the scatter plot is applied using the scatter_kws dic
sns.scatterplot(
    x='total_bill',
    y='tip',
    hue='sex',
    data=tips_df,
    markers=['o', '^'],
    s=100,  # Marker size
    linewidth=0.5,
    edgecolor='w'
)

# Display the plot
plt.show()
```



In [ ]:

```
# You can separate the data into separate columns for day data
# sns.lmplot(x='total_bill', y='tip', col='sex', row='time', data=tips_df)
```

```python
tips_df.head()

# Makes the fonts more readable
sns.set_context('poster', font_scale=1.4)

sns.lmplot(x='total_bill', y='tip', data=tips_df, col='day', hue='sex',
           height=8, aspect=0.6)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x16cff7850>