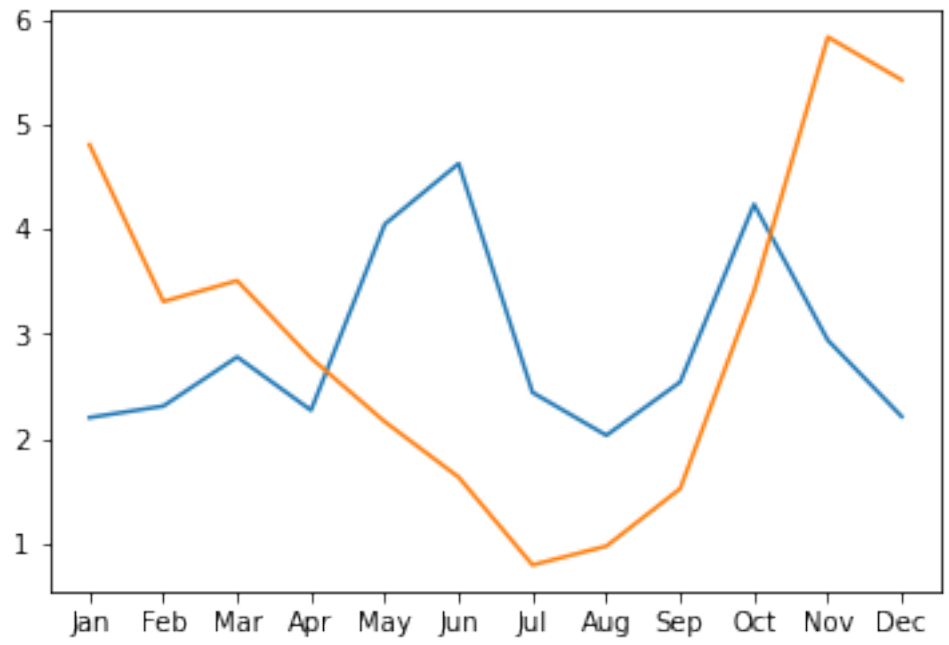
4. Data Visualization

July 12, 2022

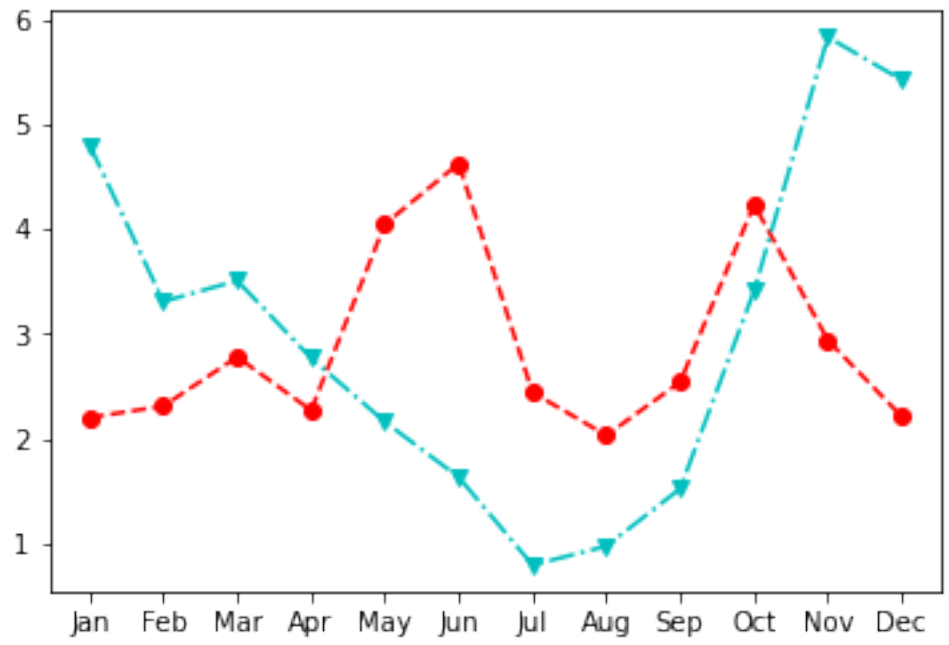
1 INTRODUCCIÓN A MATPLOTLIB

1.1 INTRODUCCIÓN A MATPLOTLIB

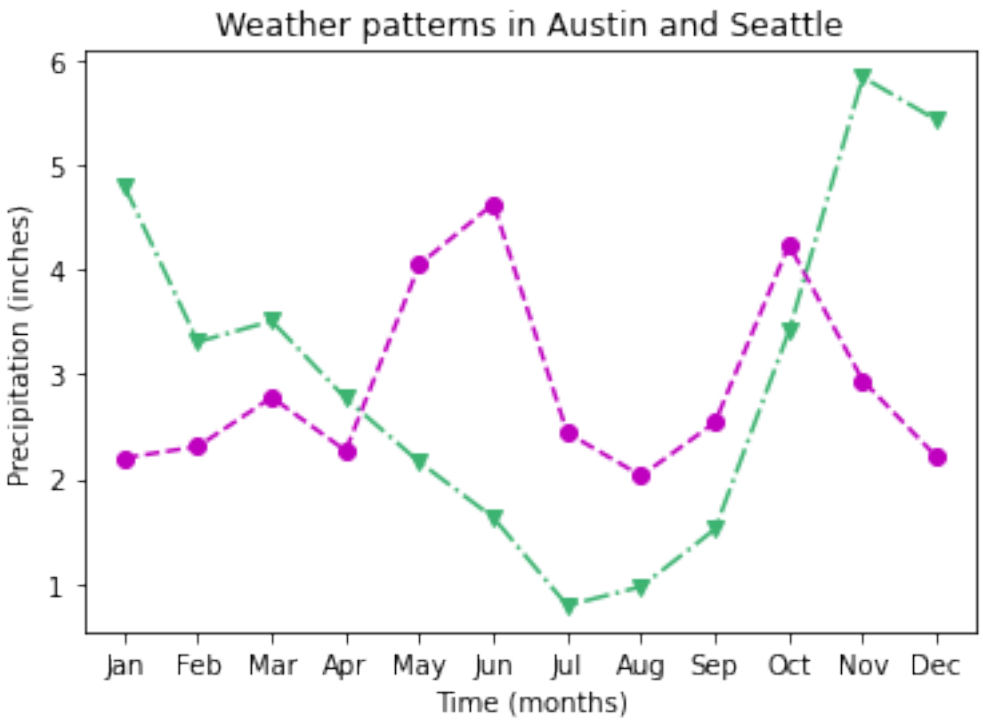
[1]:
```
import matplotlib.pyplot as plt
import pandas as pd

seattle_weather = pd.read_csv("https://assets.datacamp.com/production/
 ↪repositories/3634/datasets/6fd451508ecce0d63354fad86704236592eed8ca/
 ↪seattle_weather.csv", index_col ='DATE')
austin_weather = pd.read_csv("https://assets.datacamp.com/production/
 ↪repositories/3634/datasets/e76b460b41dc7ff286d78246daf3a8c324cb5587/
 ↪austin_weather.csv", index_col ='DATE')

seattle_weather['MONTH'] = pd.to_datetime(seattle_weather.index, format='%m').
 ↪month_name().str.slice(stop=3)
austin_weather['MONTH'] = pd.to_datetime(austin_weather.index, format='%m').
 ↪month_name().str.slice(stop=3)

seattle_weather = seattle_weather[seattle_weather['STATION'] == 'USW00094290']

# Create a Figure and an Axes with plt.subplots
fig, ax = plt.subplots()

seattle_weather = seattle_weather[seattle_weather["STATION"] == "USW00094290"]

ax.plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-NORMAL"])
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-NORMAL"])
plt.show()
```



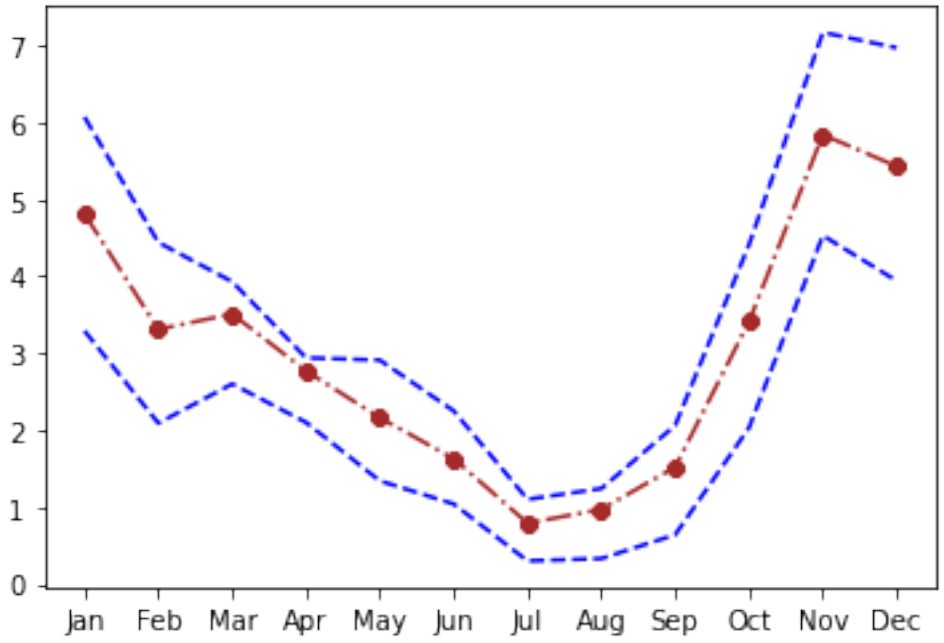
[2]:
```
# Marcadores, tipos de línea y color

# https://matplotlib.org/stable/api/markers_api.html
# https://matplotlib.org/stable/gallery/lines_bars_and_markers/linestyles.html

fig, ax = plt.subplots()

ax.plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-NORMAL"], marker =␣
 ↪"o", linestyle = "--", color = "r")
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-NORMAL"], marker =␣
 ↪"v", linestyle = "dashdot", color = "c")
plt.show()
```



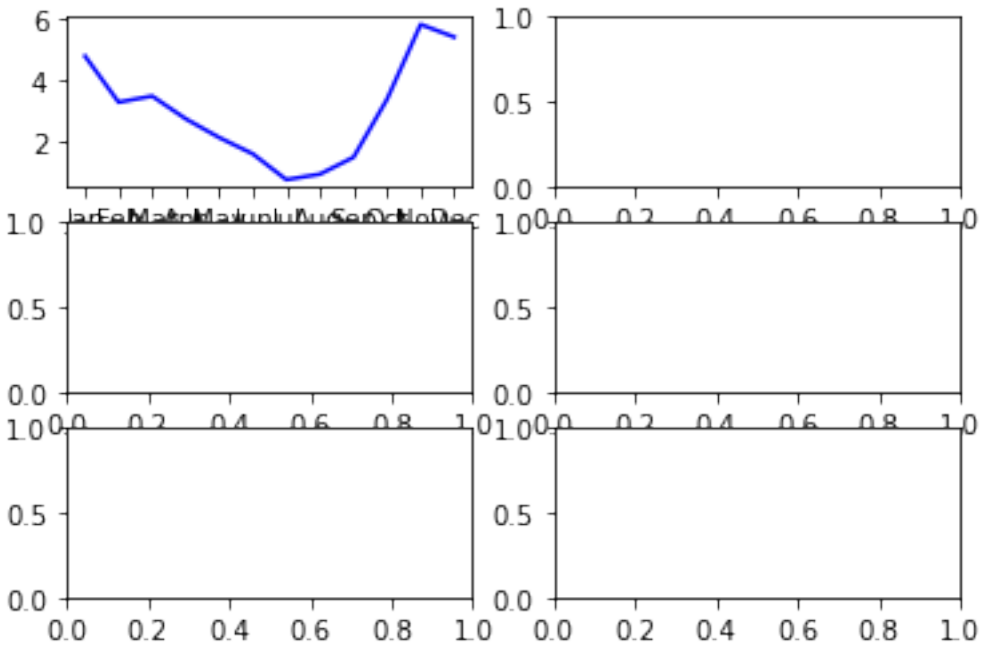
[3]:
```
# Etiquetas

fig, ax = plt.subplots()

ax.plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-NORMAL"], marker =␣
 ↪"o", linestyle = "--", color = "m")
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-NORMAL"], marker =␣
 ↪"v", linestyle = "dashdot", color = "mediumseagreen")
ax.set_xlabel("Time (months)")
ax.set_ylabel("Precipitation (inches)")
ax.set_title("Weather patterns in Austin and Seattle")
plt.show()
```



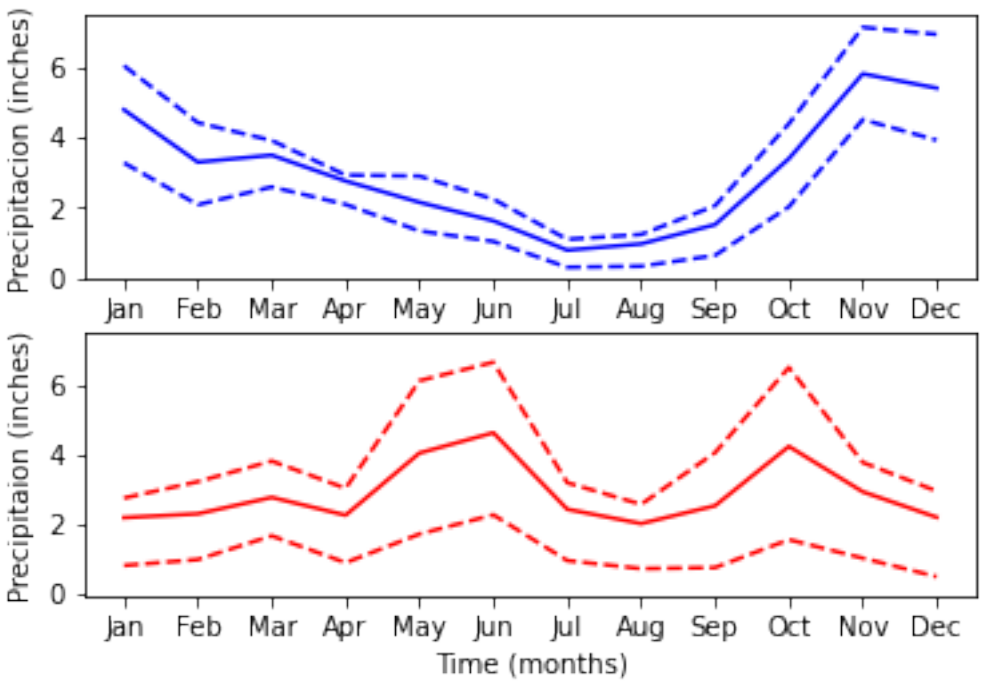
[4]:
```
# Más datos

fig, ax = plt.subplots()

ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-NORMAL"], marker =␣
↪"8", linestyle = "dashdot", color = "brown")
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-25PCTL"], linestyle␣
↪= "--", color = "blue")
ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-75PCTL"], linestyle␣
↪= "--", color = "blue")

plt.show()
```



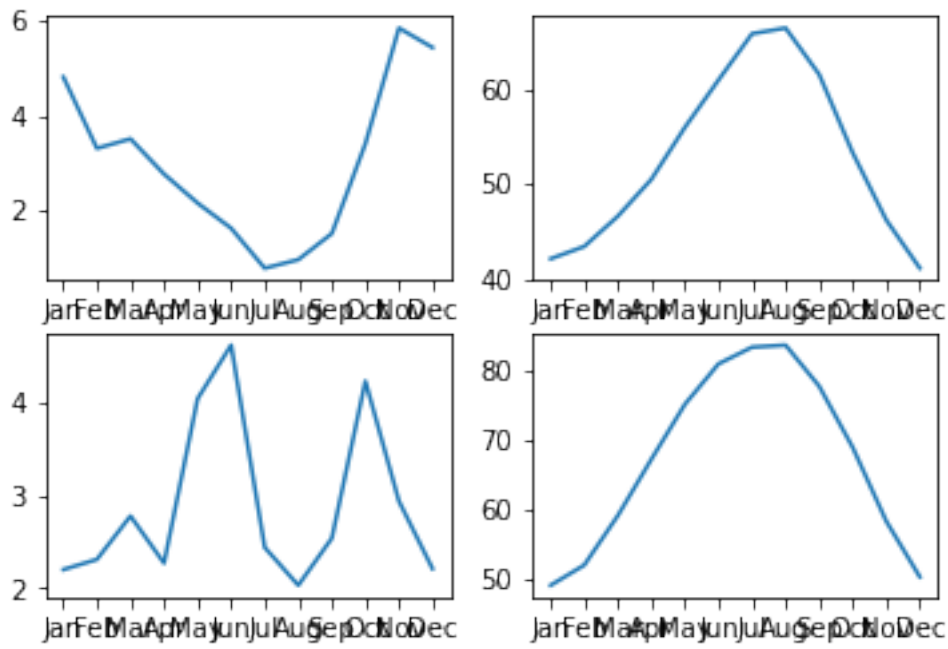
[5]:
```
# Si son muchas series, es mejor usar subplots:

fig, ax = plt.subplots(3, 2)


# ax es ahora como una "matriz":

ax[0,0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-NORMAL"],␣
 ↪color = "b")
plt.show()
```



[6]:
```
fig, ax = plt.subplots(2, 1, sharey = True) # sharey es para que compartan la misma magnitud del eje vertical

ax[0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-NORMAL"], color = "b")
ax[0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-25PCTL"], color = "b", linestyle = "--")
ax[0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-75PCTL"], color = "b", linestyle = "--")

ax[1].plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-NORMAL"], color = "r")
ax[1].plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-25PCTL"], color = "r", linestyle = "--")
ax[1].plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-75PCTL"], color = "r", linestyle = "--")

ax[0].set_ylabel("Precipitacion (inches)")
ax[1].set_ylabel("Precipitaion (inches)")
ax[1].set_xlabel("Time (months)")

plt.show()
```



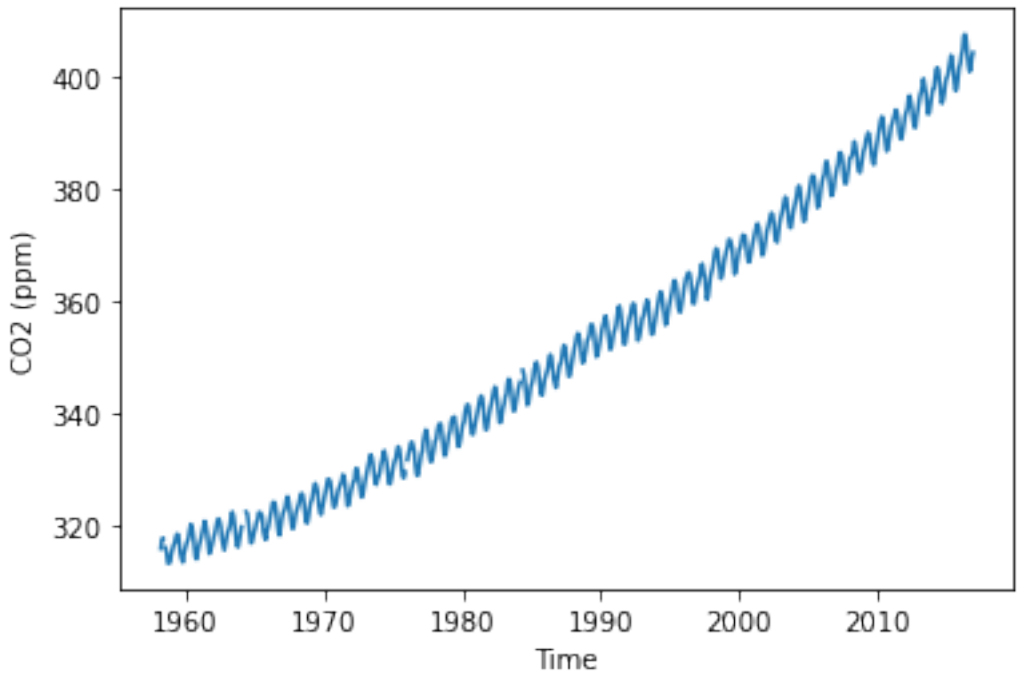
[7]:
```
# Ejemplo

# Create a Figure and an array of subplots with 2 rows and 2 columns
fig, ax = plt.subplots(2, 2)

# Addressing the top left Axes as index 0, 0, plot month and Seattle␣
 ↪precipitation
ax[0, 0].plot(seattle_weather["MONTH"], seattle_weather["MLY-PRCP-NORMAL"])

# In the top right (index 0,1), plot month and Seattle temperatures
ax[0, 1].plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])

# In the bottom left (1, 0) plot month and Austin precipitations
ax[1, 0].plot(austin_weather["MONTH"], austin_weather["MLY-PRCP-NORMAL"])

# In the bottom right (1, 1) plot month and Austin temperatures
ax[1, 1].plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])
plt.show()
```



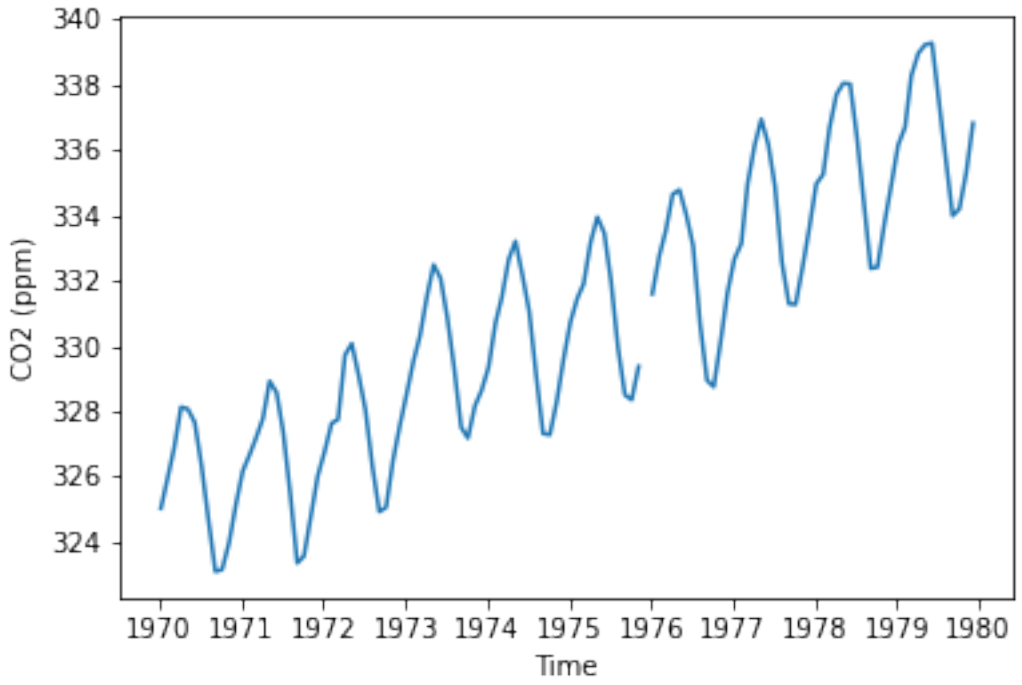
1.2 SERIES DE TIEMPO

Hay que indicar la columna que será la fecha.

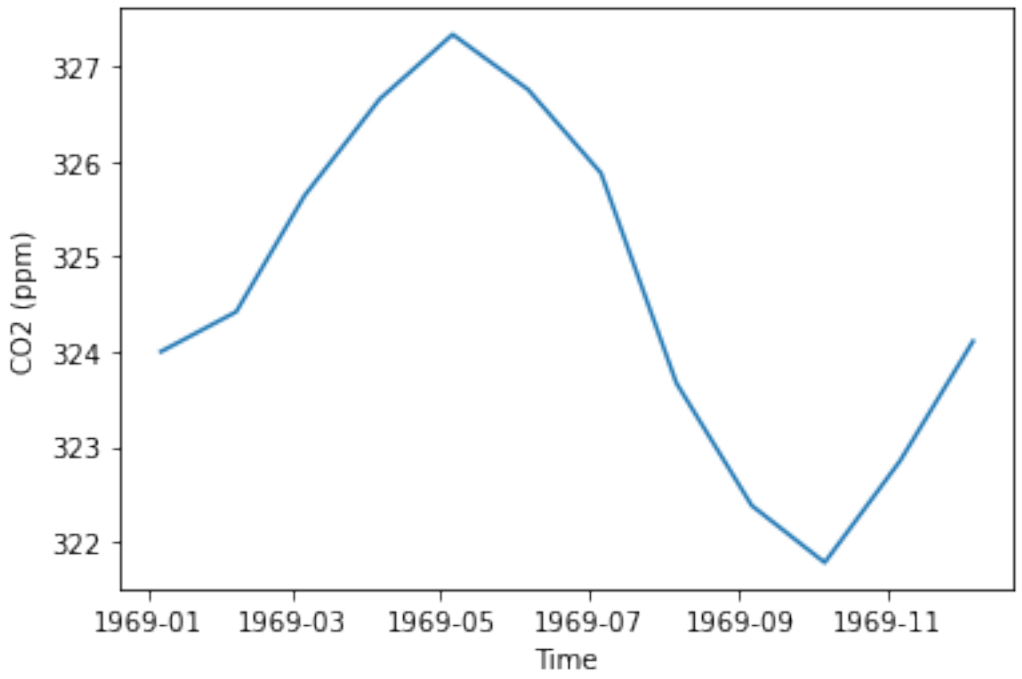
[8]:
```
climate_change = pd.read_csv('https://assets.datacamp.com/production/
 ↪repositories/3634/datasets/411add3f8570d5adf891127fd64095020210711b/
 ↪climate_change.csv', parse_dates=["date"], index_col="date")

fig, ax = plt.subplots()

ax.plot(climate_change.index, climate_change["co2"])
ax.set_xlabel("Time")
ax.set_ylabel("CO2 (ppm)")
plt.show()
```



[9]:
```
# Se puede seleccionar solo un periodo de tiempo:

seventies = climate_change["1970-01-01":"1979-12-31"]

fig, ax = plt.subplots()

ax.plot(seventies.index, seventies["co2"])
ax.set_xlabel("Time")
ax.set_ylabel("CO2 (ppm)")
plt.show()
```

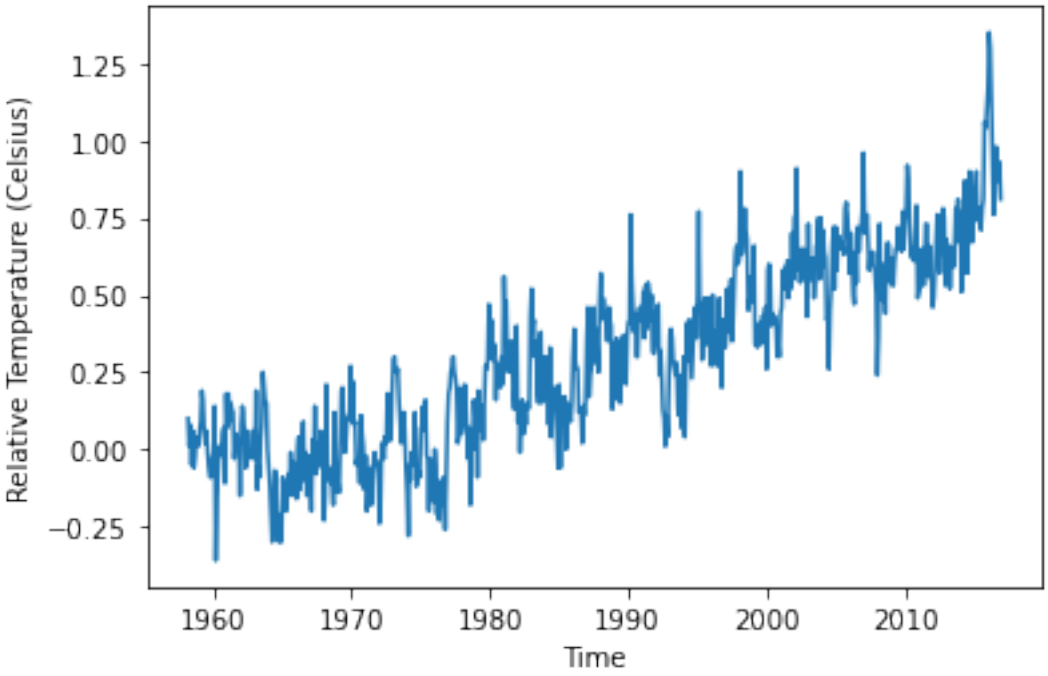


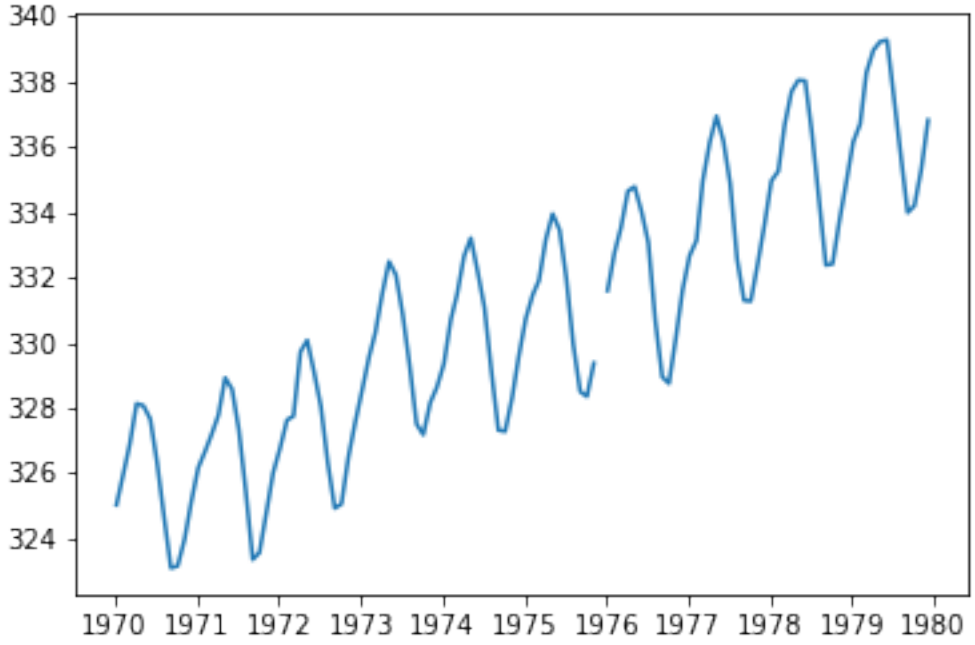
[10]:
```
sixty_nine = climate_change["1969-01-01":"1969-12-31"]

fig, ax = plt.subplots()

ax.plot(sixty_nine.index, sixty_nine["co2"])
ax.set_xlabel("Time")
ax.set_ylabel("CO2 (ppm)")
plt.show()
```



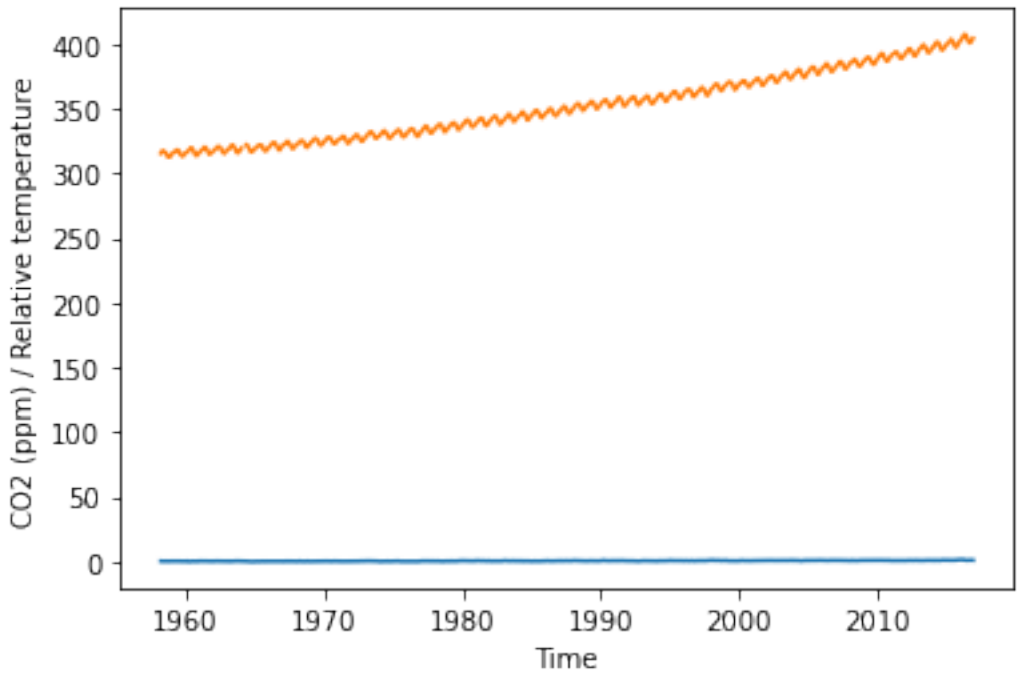
[11]:
```
# Ejemplo

fig, ax = plt.subplots()

# Add the time-series for "relative_temp" to the plot
ax.plot(climate_change.index, climate_change["relative_temp"])

# Set the x-axis label
ax.set_xlabel("Time")

# Set the y-axis label
ax.set_ylabel("Relative Temperature (Celsius)")

# Show the figure
plt.show()

###

# Use plt.subplots to create fig and ax
fig, ax = plt.subplots()

# Create variable seventies with data from "1970-01-01" to "1979-12-31"
seventies = climate_change["1970-01-01":"1979-12-31"]
```

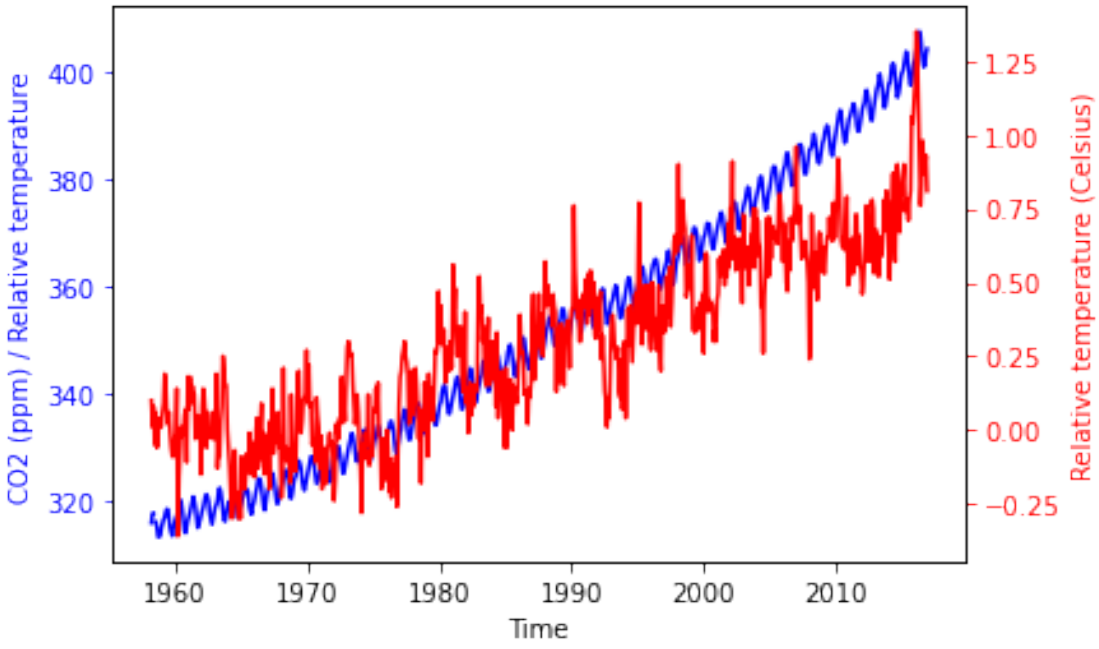
```
# Add the time-series for "co2" data from seventies to the plot
ax.plot(seventies.index, seventies["co2"])

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



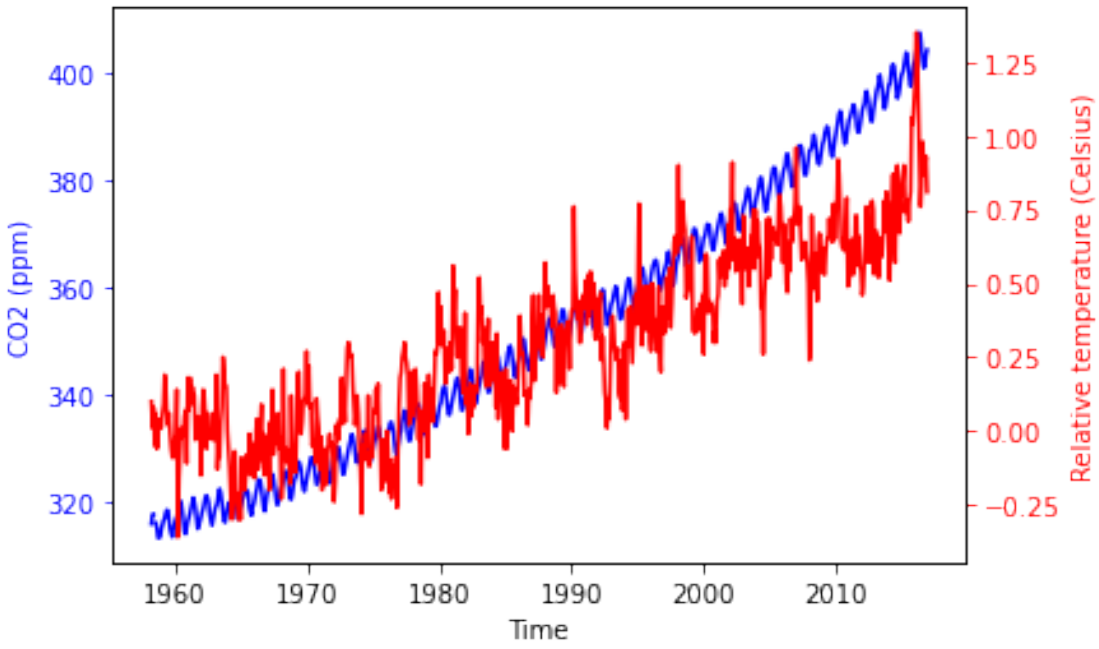


[12]:
```
### Dos series de tiempo simultáneas

fig, ax = plt.subplots()

ax.plot(climate_change.index, climate_change["relative_temp"])
ax.plot(climate_change.index, climate_change["co2"])
ax.set_xlabel("Time")
ax.set_ylabel("CO2 (ppm) / Relative temperature")
plt.show()

# Donde es evidente que hay que modificar la escala:

fig, ax = plt.subplots()

ax.plot(climate_change.index, climate_change["co2"], color = "blue")
ax.set_xlabel("Time")
ax.set_ylabel("CO2 (ppm) / Relative temperature", color = "blue")
ax.tick_params("y", colors = "blue")
ax2 = ax.twinx() # crea un eje Y gemelo
ax2.plot(climate_change.index, climate_change["relative_temp"], color = "red")
ax2.set_ylabel("Relative temperature (Celsius)", color = "red")
ax2.tick_params("y", colors = "red")
plt.show()
```





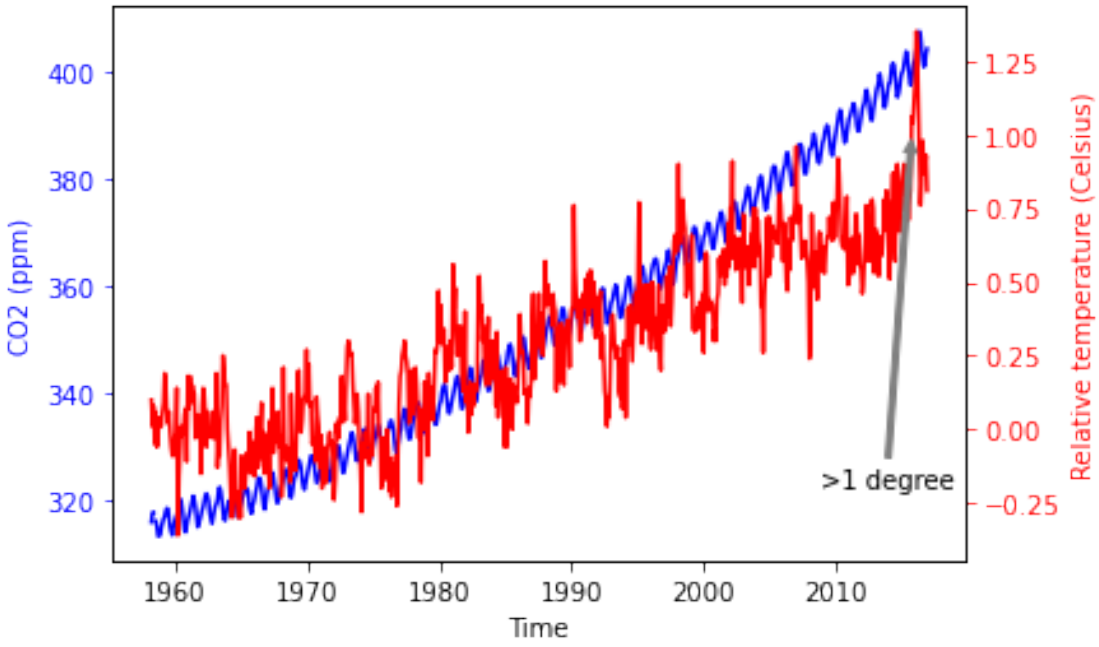
[13]:
```
# Lo cual se puede automatizar:

def plot_timeseries(axes, x, y, color, xlabel, ylabel):
    axes.plot(x, y, color = color)
    axes.set_xlabel(xlabel)
    axes.set_ylabel(ylabel, color = color)
    axes.tick_params("y", colors = color)

fig, ax = plt.subplots()
plot_timeseries(ax, climate_change.index, climate_change["co2"], "blue",␣
 ↪"Time", "CO2 (ppm)")

ax2 = ax.twinx()

plot_timeseries(ax2, climate_change.index, climate_change["relative_temp"],␣
 ↪"red", "Time", "Relative temperature (Celsius)")
```



1.2.1 Anotaciones

[14]:
```
def plot_timeseries(axes, x, y, color, xlabel, ylabel):
    axes.plot(x, y, color = color)
    axes.set_xlabel(xlabel)
    axes.set_ylabel(ylabel, color = color)
    axes.tick_params("y", colors = color)

fig, ax = plt.subplots()
plot_timeseries(ax, climate_change.index, climate_change["co2"], "blue",␣
↪"Time", "CO2 (ppm)")

ax2 = ax.twinx()

plot_timeseries(ax2, climate_change.index, climate_change["relative_temp"],␣
↪"red", "Time", "Relative temperature (Celsius)")

ax2.annotate(">1 degree", xy = (pd.Timestamp("2015-10-06"), 1), xytext = (pd.
↪Timestamp("2008-10-06"), -0.2), \
             arrowprops = {"arrowstyle":"simple", "color":"gray"})

# https://matplotlib.org/2.0.2/users/annotations.html#:~:
↪text=The%20annotate()%20function%20in,two%20points%20on%20the%20plot.
↪&text=This%20annotates%20a%20point%20at,at%20xytext%20given%20in%20textcoords%20.
↪
```

[14]: Text(2008-10-06 00:00:00, -0.2, '>1 degree')

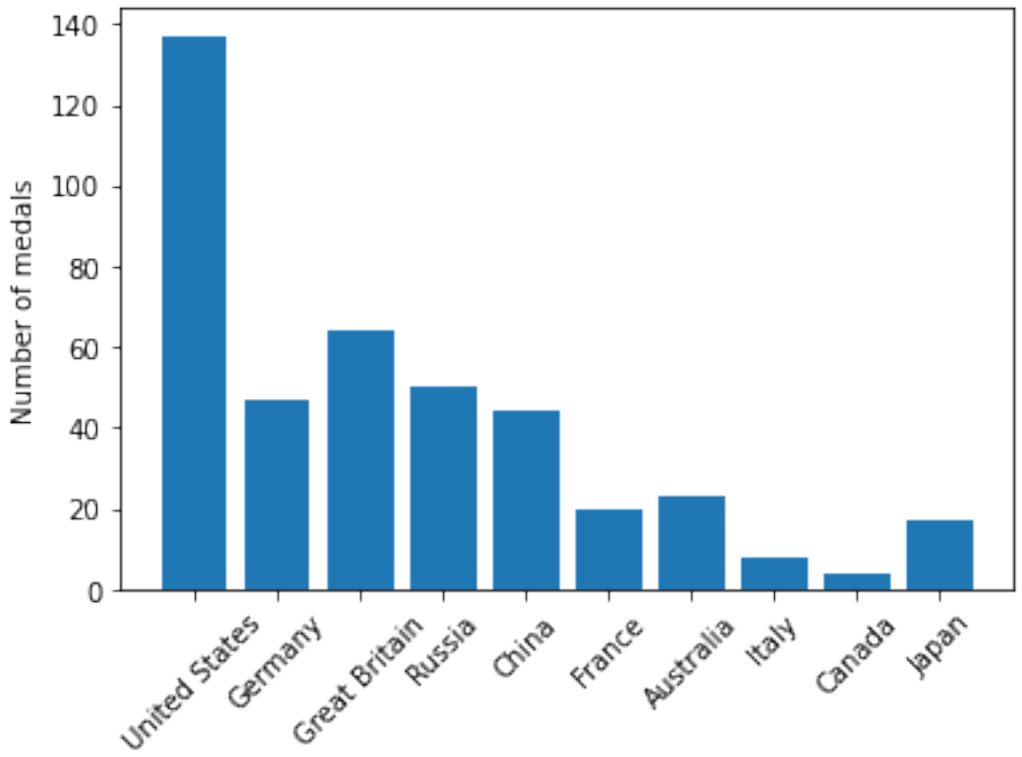


1.3 COMPARACIONES CUANTITATIVAS Y VISUALIZACIONES ESTADÍSTICAS

1.3.1 Gráficos de barras

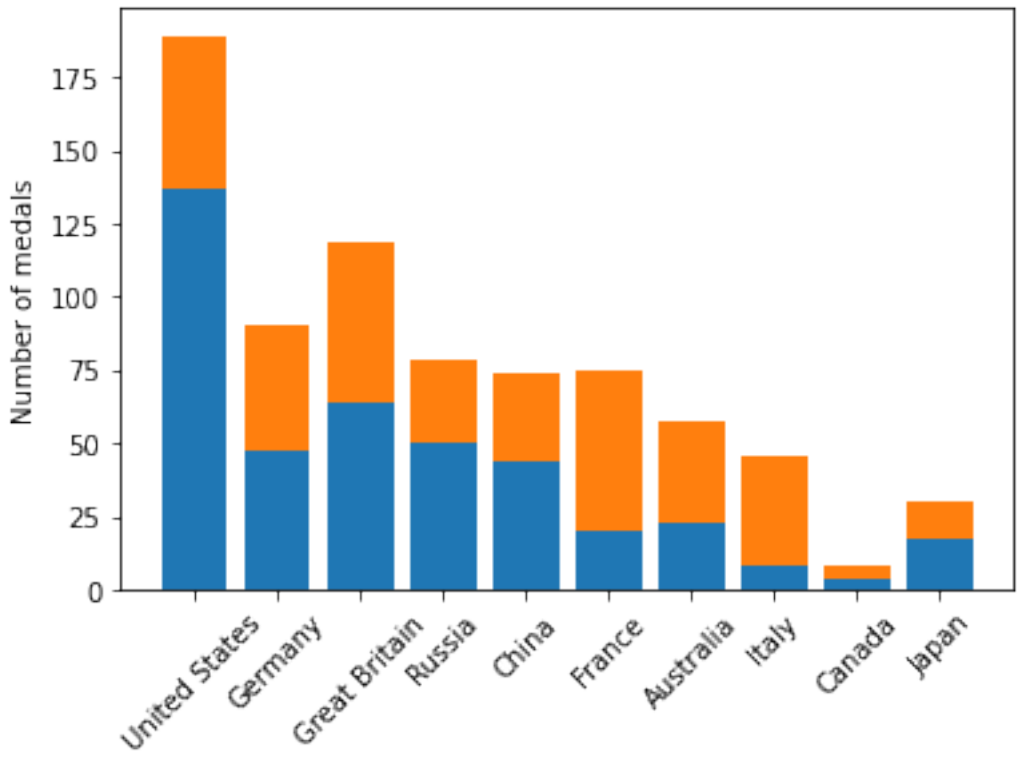
[15]:
```
medals = pd.read_csv('https://assets.datacamp.com/production/repositories/3634/
 ↪datasets/ec663f9f509bf633d40932f65bd4cc51205689e2/medals_by_country_2016.
 ↪csv', index_col = "Unnamed: 0")

fig, ax = plt.subplots()

ax.bar(medals.index, medals["Gold"])
ax.set_xticklabels(medals.index, rotation = 45)
ax.set_ylabel("Number of medals")
plt.show()
```

```
<ipython-input-15-8808e9e368af>:6: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels(medals.index, rotation = 45)
```



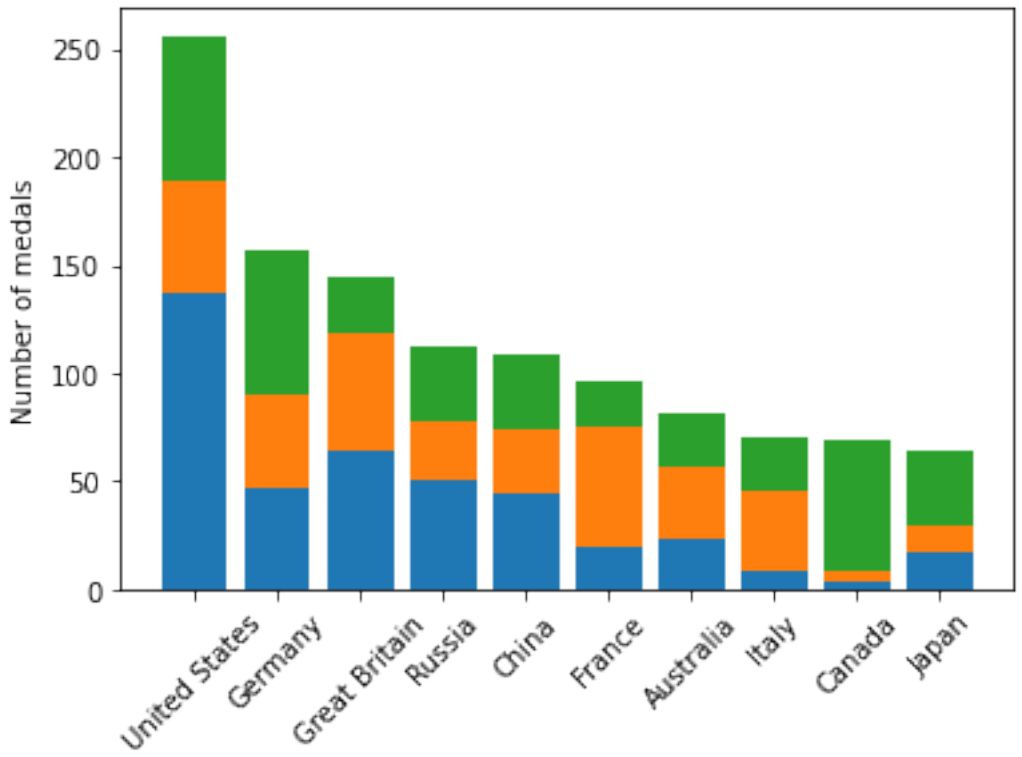
[16]:
```
# Gráfico apilado

fig, ax = plt.subplots()

ax.bar(medals.index, medals["Gold"])
ax.bar(medals.index, medals["Silver"], bottom = medals["Gold"])
ax.set_xticklabels(medals.index, rotation = 45)
ax.set_ylabel("Number of medals")
plt.show()
```

```
<ipython-input-16-2fbfb3afce17>:7: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels(medals.index, rotation = 45)
```



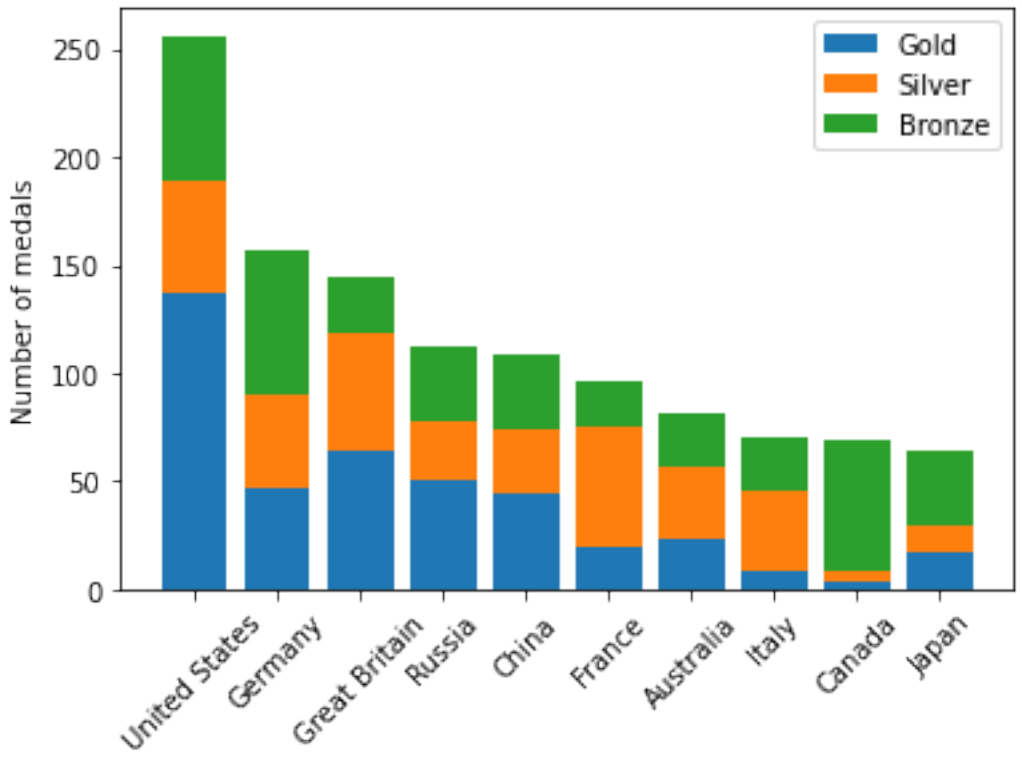
[17]:
```
# Gráfico apilado

fig, ax = plt.subplots()

ax.bar(medals.index, medals["Gold"])
ax.bar(medals.index, medals["Silver"], bottom = medals["Gold"])
ax.bar(medals.index, medals["Bronze"], bottom = medals["Gold"] + ␣
 ↪medals["Silver"])
ax.set_xticklabels(medals.index, rotation = 45)
ax.set_ylabel("Number of medals")
plt.show()
```

```
<ipython-input-17-8e10a8f97519>:8: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels(medals.index, rotation = 45)
```



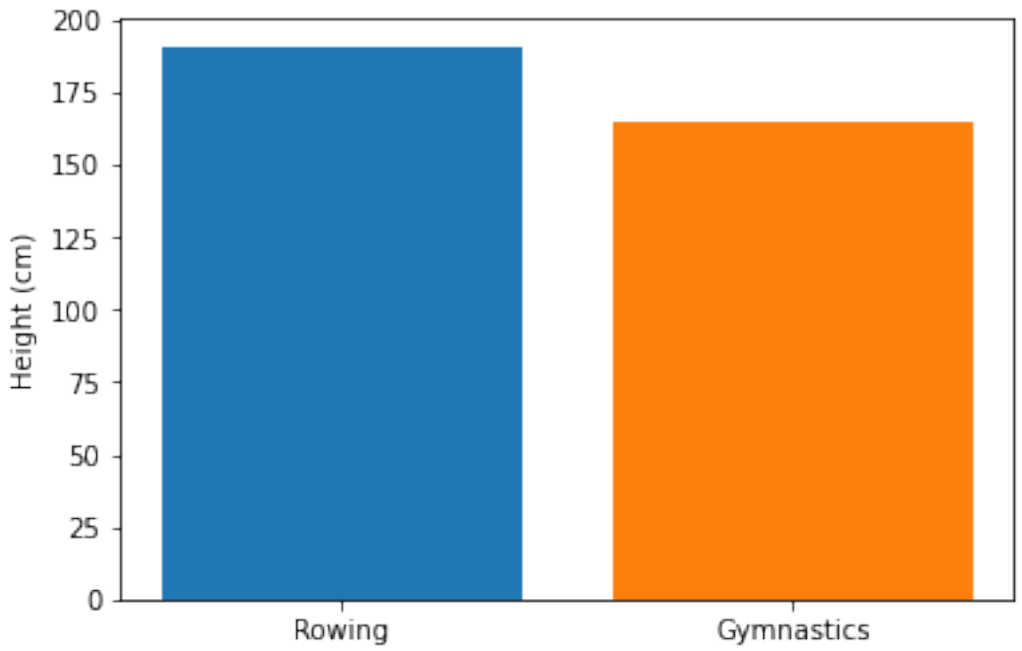
[18]:
```
# Etiquetas

fig, ax = plt.subplots()

ax.bar(medals.index, medals["Gold"], label = "Gold")
ax.bar(medals.index, medals["Silver"], bottom = medals["Gold"], label =␣
 ↪"Silver")
ax.bar(medals.index, medals["Bronze"], bottom = medals["Gold"] + ␣
 ↪medals["Silver"], label = "Bronze")
ax.set_xticklabels(medals.index, rotation = 45)
ax.set_ylabel("Number of medals")
ax.legend()
plt.show()
```

```
<ipython-input-18-18d243bd0f07>:8: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels(medals.index, rotation = 45)
```

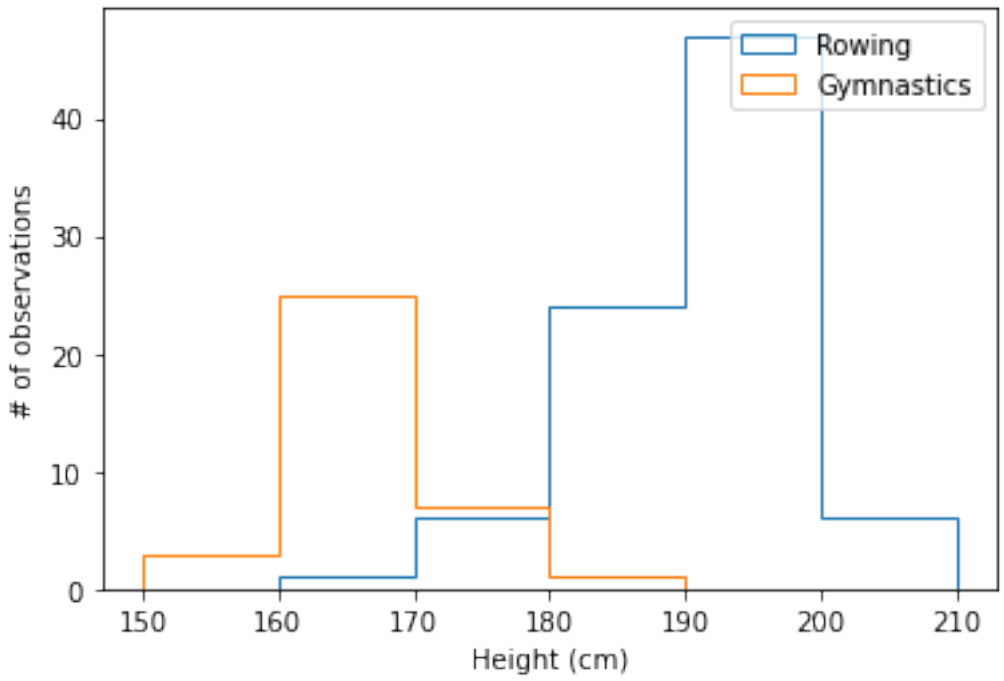


1.3.2 Histogramas

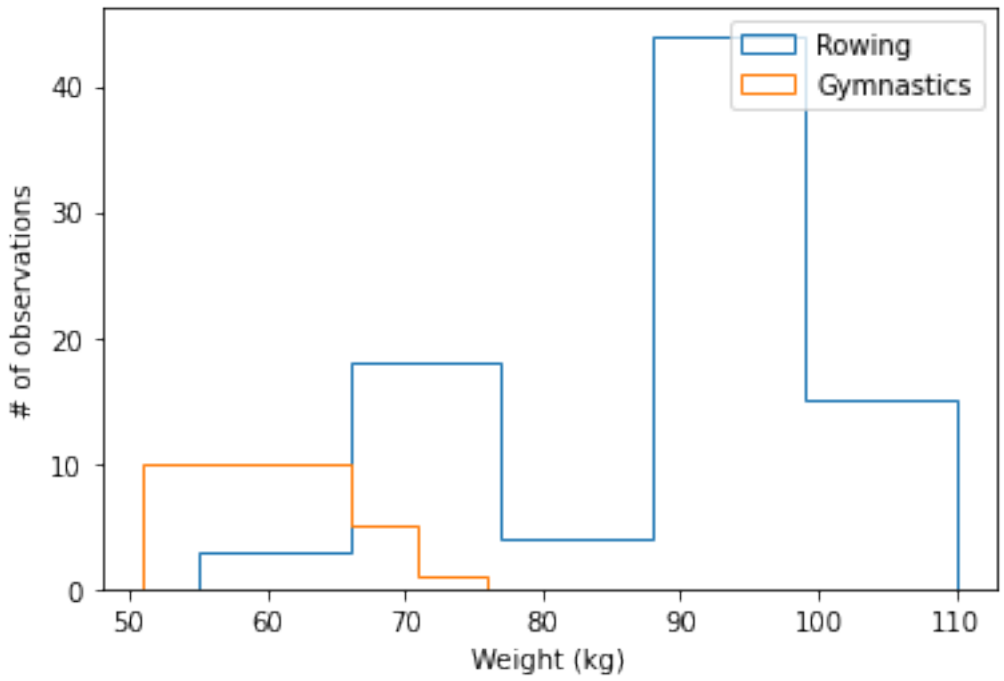
[19]:
```
summer_2016_medals = pd.read_csv('https://assets.datacamp.com/production/
 ↪repositories/3634/datasets/67d7344085ace400d612275b87738615698127a3/
 ↪summer2016.csv')
mens_rowing = summer_2016_medals[(summer_2016_medals['Sport'] == 'Rowing') &␣
 ↪(summer_2016_medals['Sex'] == 'M')]
mens_gymnastics = summer_2016_medals[(summer_2016_medals['Sport'] ==␣
 ↪'Gymnastics') & (summer_2016_medals['Sex'] == 'M')]

###

fig, ax = plt.subplots()

ax.bar("Rowing", mens_rowing["Height"].mean())
ax.bar("Gymnastics", mens_gymnastics["Height"].mean())

ax.set_ylabel("Height (cm)")

plt.show()
```



[20]:
```
fig, ax = plt.subplots()

ax.hist(mens_rowing["Height"], label = "Rowing", bins = [150, 160, 170, 180,␣
↪190, 200, 210], histtype = "step")
ax.hist(mens_gymnastics["Height"], label = "Gymnastics", bins = [150, 160, 170,␣
↪180, 190, 200, 210], histtype = "step")
ax.set_xlabel("Height (cm)")
ax.set_ylabel("# of observations")
ax.legend()

plt.show()
```

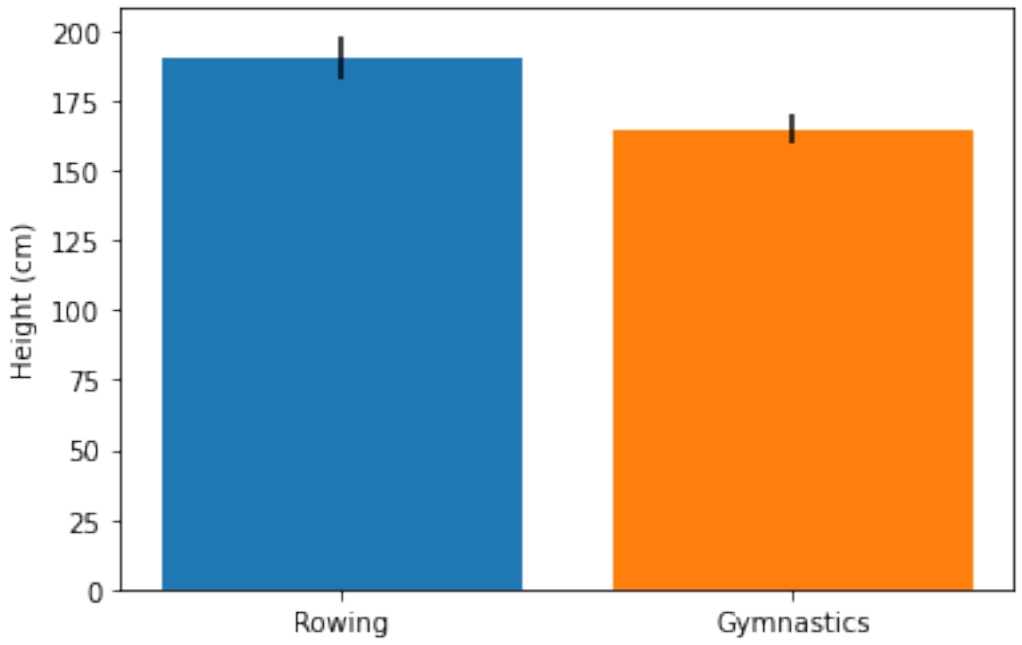


[21]:
```
fig, ax = plt.subplots()

# Plot a histogram of "Weight" for mens_rowing
ax.hist(mens_rowing["Weight"], histtype='step', label="Rowing", bins=5)

# Compare to histogram of "Weight" for mens_gymnastics
ax.hist(mens_gymnastics["Weight"], histtype='step', label="Gymnastics", bins=5)

ax.set_xlabel("Weight (kg)")
ax.set_ylabel("# of observations")

# Add the legend and show the Figure
ax.legend()
plt.show()
```

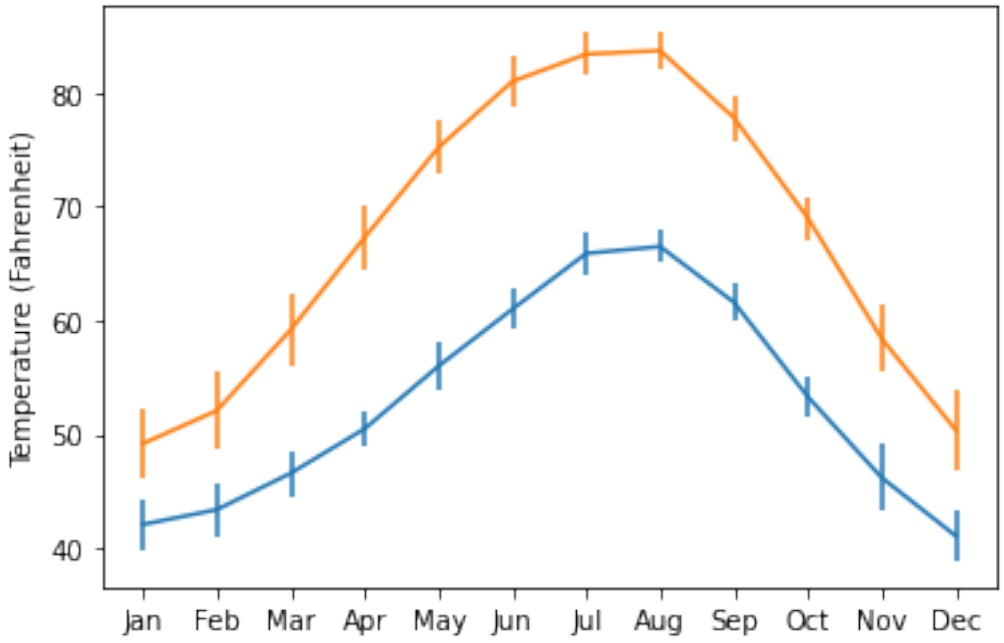


1.3.3 Graficación estadística

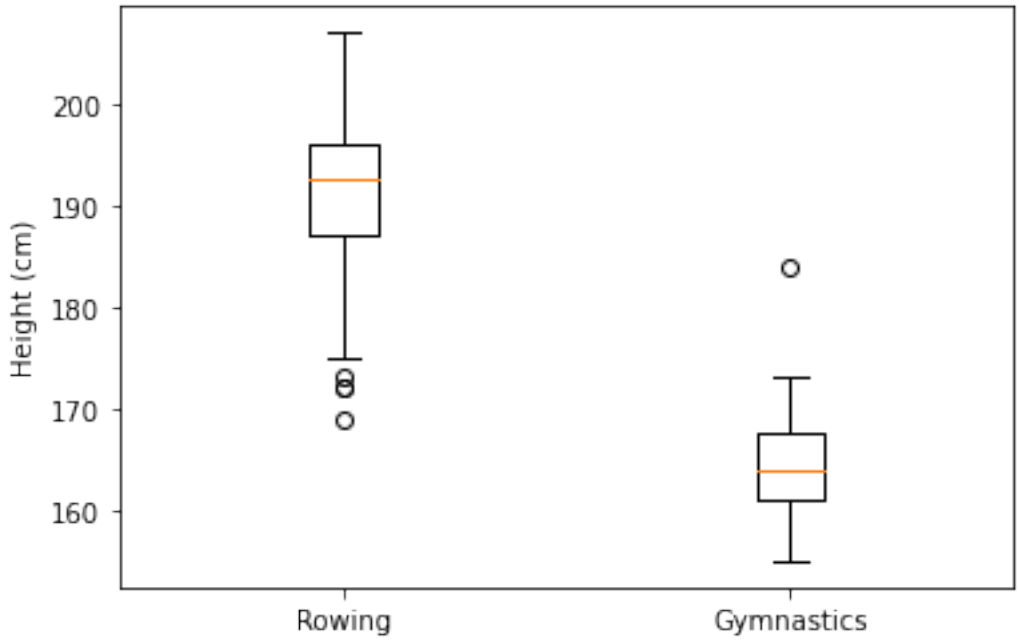
[22]:
```
# Barras de error

fig, ax = plt.subplots()

# Add a bar for the rowing "Height" column mean/std
ax.bar("Rowing", mens_rowing["Height"].mean(), yerr=mens_rowing["Height"].std())
ax.bar("Gymnastics", mens_gymnastics["Height"].mean(),␣
 ↪yerr=mens_gymnastics["Height"].std())

ax.set_ylabel("Height (cm)")

plt.show()
```



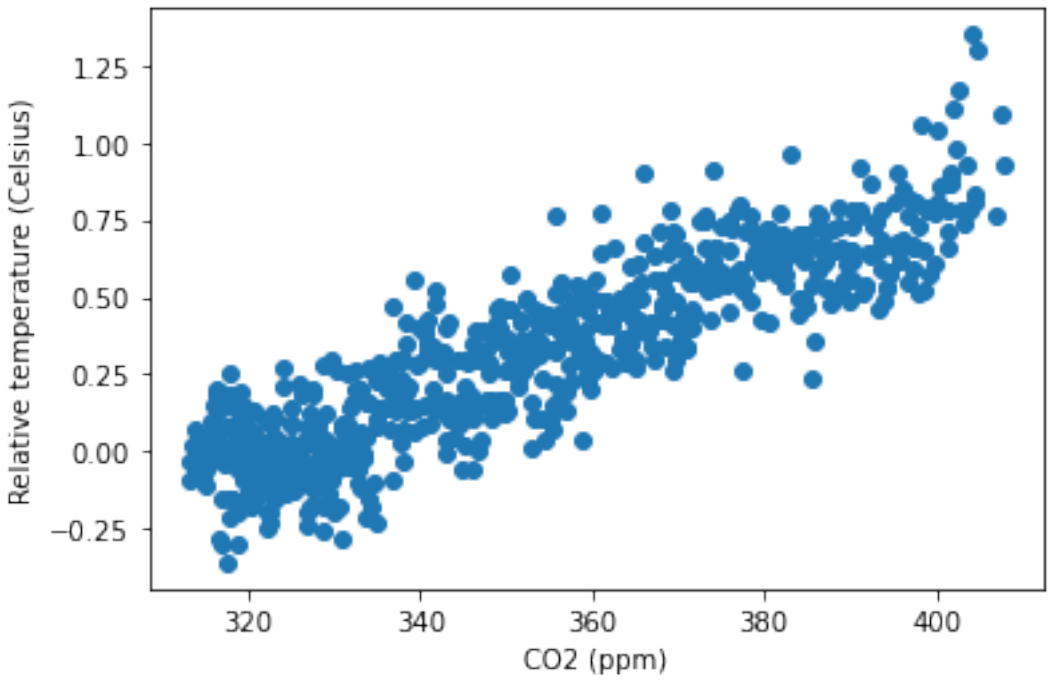
[23]:
```
fig, ax = plt.subplots()

ax.errorbar(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"],␣
 ↪yerr=seattle_weather["MLY-TAVG-STDDEV"])
ax.errorbar(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"],␣
 ↪yerr=austin_weather["MLY-TAVG-STDDEV"])

ax.set_ylabel("Temperature (Fahrenheit)")

plt.show()
```



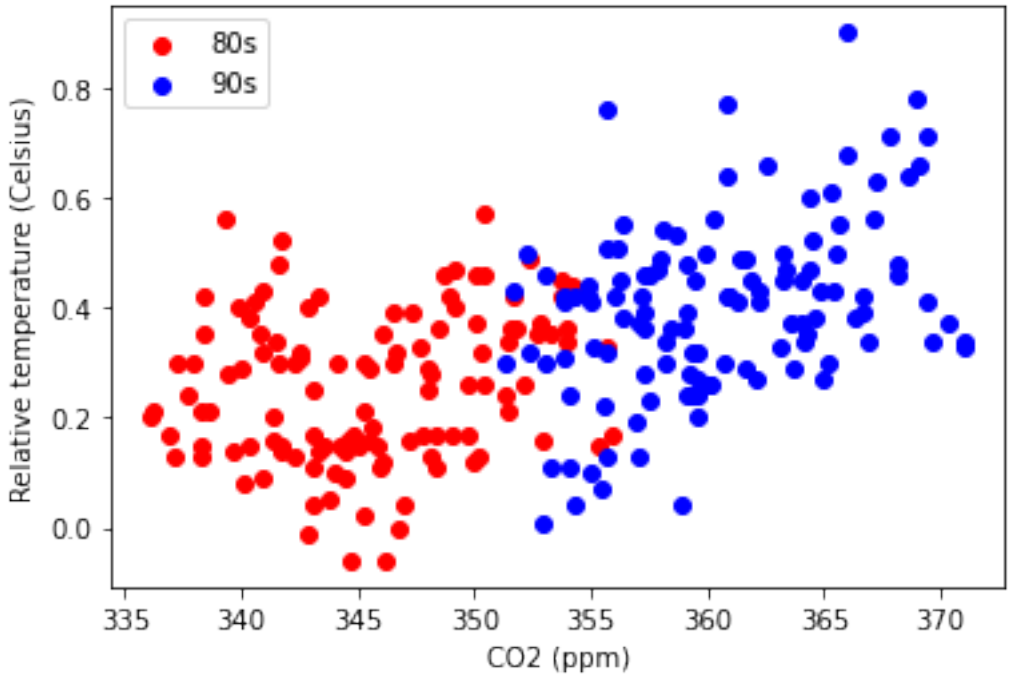
[24]:
```
# Boxplots

fig, ax = plt.subplots()

ax.boxplot([mens_rowing["Height"], mens_gymnastics["Height"]])
ax.set_xticklabels(["Rowing", "Gymnastics"])
ax.set_ylabel("Height (cm)")

plt.show()
```



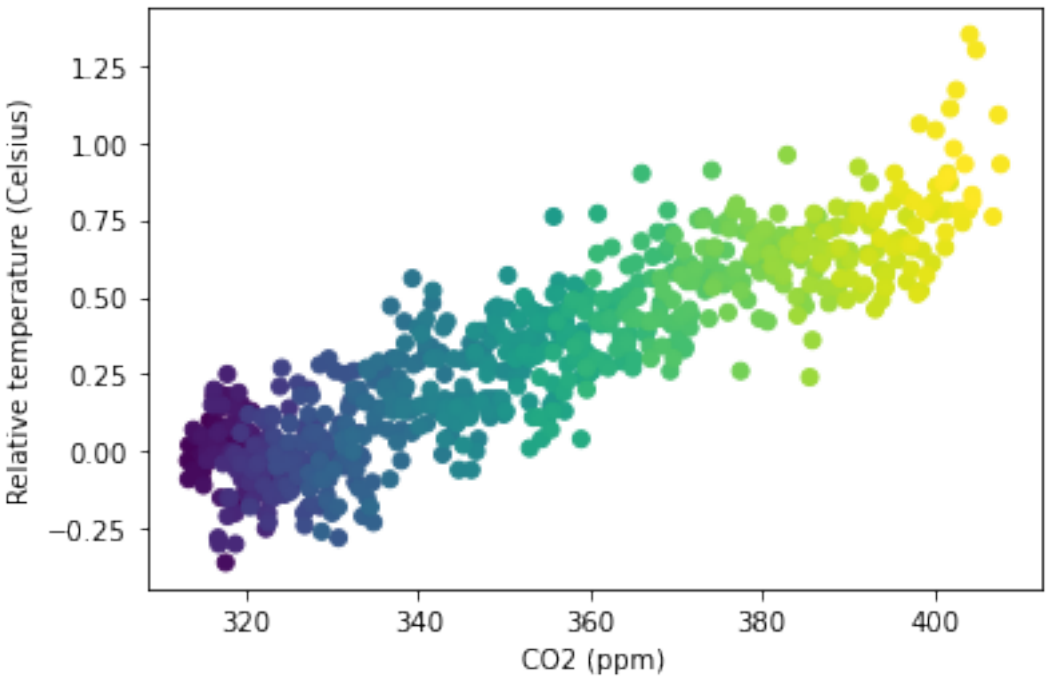
[25]:
```
# Scatterplots

fig, ax = plt.subplots()

ax.scatter(climate_change["co2"], climate_change["relative_temp"])

ax.set_xlabel("CO2 (ppm)")
ax.set_ylabel("Relative temperature (Celsius)")

plt.show()
```



[26]:
```
# Dos simultáneos

eighties = climate_change["1980-01-01":"1989-12-31"]
nineties = climate_change["1990-01-01":"1999-12-31"]

fig, ax = plt.subplots()

ax.scatter(eighties["co2"], eighties["relative_temp"], color = "red", label =␣
↪"80s")
ax.scatter(nineties["co2"], nineties["relative_temp"], color = "blue", label =␣
↪"90s")

ax.legend()

ax.set_xlabel("CO2 (ppm)")
ax.set_ylabel("Relative temperature (Celsius)")

plt.show()
```

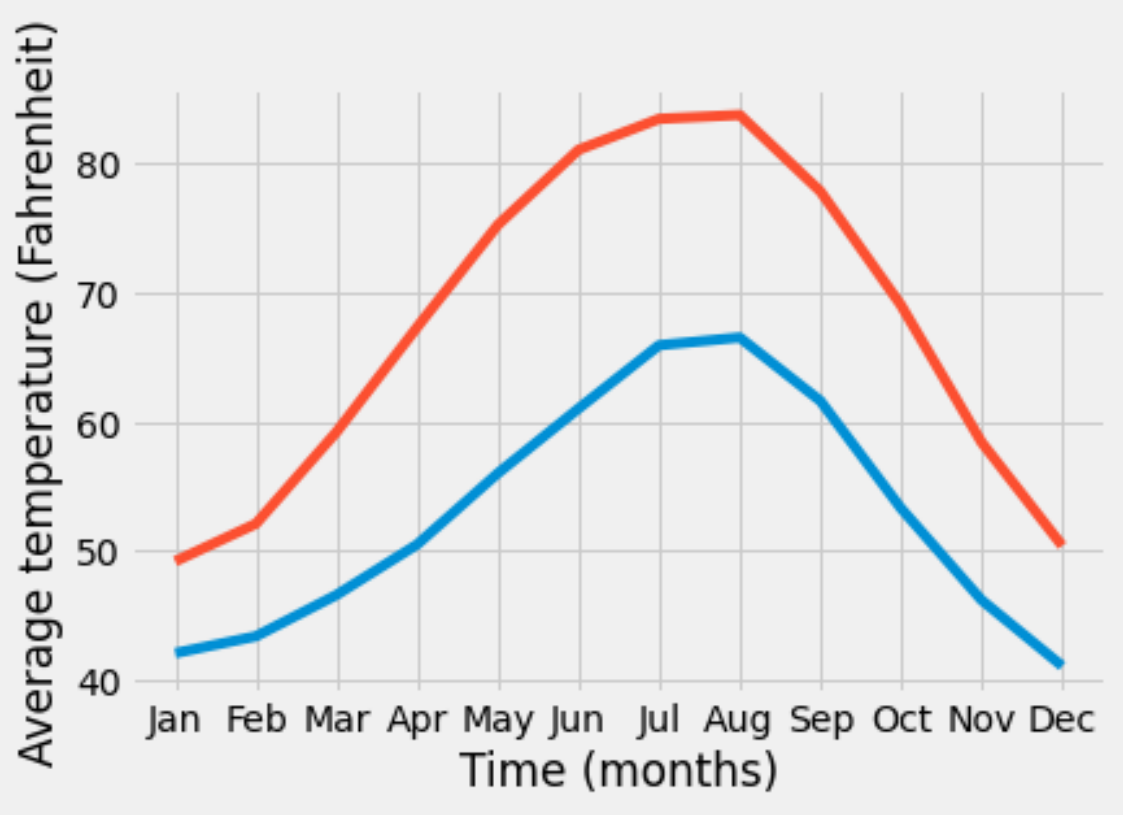


[27]:
```
# Codificación de una 3era variable

fig, ax = plt.subplots()

ax.scatter(climate_change["co2"], climate_change["relative_temp"], c =␣
↪climate_change.index)

ax.set_xlabel("CO2 (ppm)")
ax.set_ylabel("Relative temperature (Celsius)")

plt.show() # Donde la escala viridis indica la fecha
```

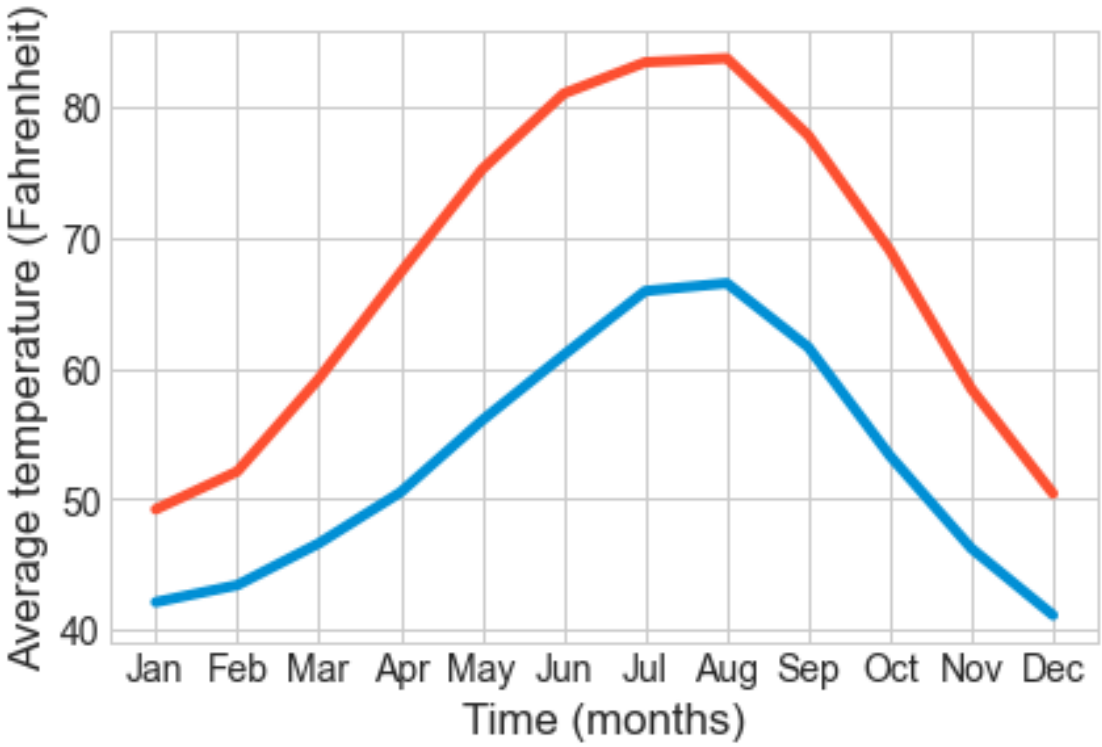


1.4 COMPARTIENDO VISUALIZACIONES

[28]:
```
# Estilo de gráficos

import matplotlib.pyplot as plt

plt.style.use("fivethirtyeight")

fig, ax = plt.subplots()

ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])

ax.set_xlabel("Time (months)")
ax.set_ylabel("Average temperature (Fahrenheit)")

plt.show()

# https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html
```



[29]:
```
plt.style.use("seaborn-whitegrid")

fig, ax = plt.subplots()

ax.plot(seattle_weather["MONTH"], seattle_weather["MLY-TAVG-NORMAL"])
ax.plot(austin_weather["MONTH"], austin_weather["MLY-TAVG-NORMAL"])

ax.set_xlabel("Time (months)")
ax.set_ylabel("Average temperature (Fahrenheit)")

plt.show()

# https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html
```

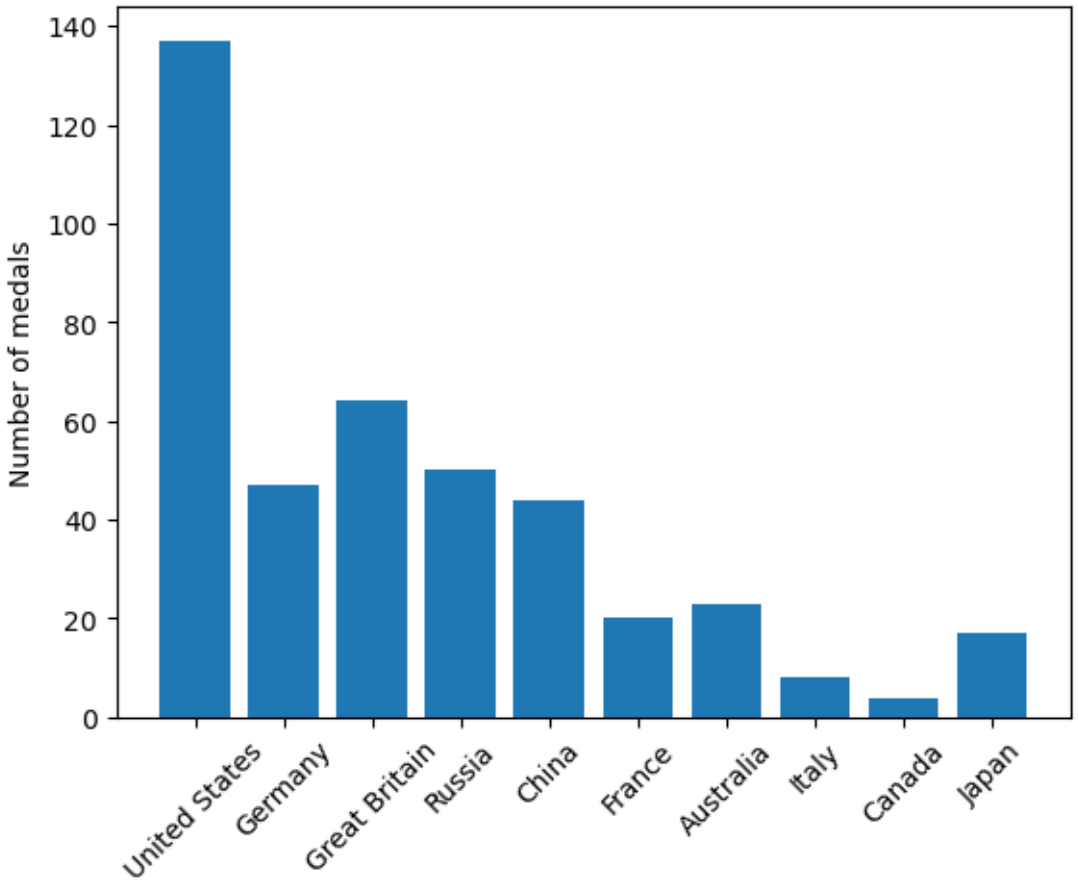


1.4.1 Guardando visualizaciones

[30]:
```
plt.style.use('default')
# Show the figure
fig, ax = plt.subplots()

# Plot a bar-chart of gold medals as a function of country
ax.bar(medals.index, medals["Gold"])

ax.set_xticklabels(medals.index, rotation=45)

# Set the y-axis label
ax.set_ylabel("Number of medals")

plt.show()

# PARA GUARDAR:

fig.savefig("Img/gold_medals.png", dpi = 900) # se puede usar dpi o quality(0,␣
↪100)

# Se puede guardar en PNG, JPG o vectores (SVG)
```

```
# PARA CAMBIAR EL TAMAÑO DE LA IMAGEN:

fig.set_size_inches([6, 3.5])
fig.savefig("Img/gold_medals.png", dpi = 900) # se puede usar dpi o quality(0,␣
↪100)
```

```
<ipython-input-30-7ebb287ffb6c>:8: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels(medals.index, rotation=45)
```



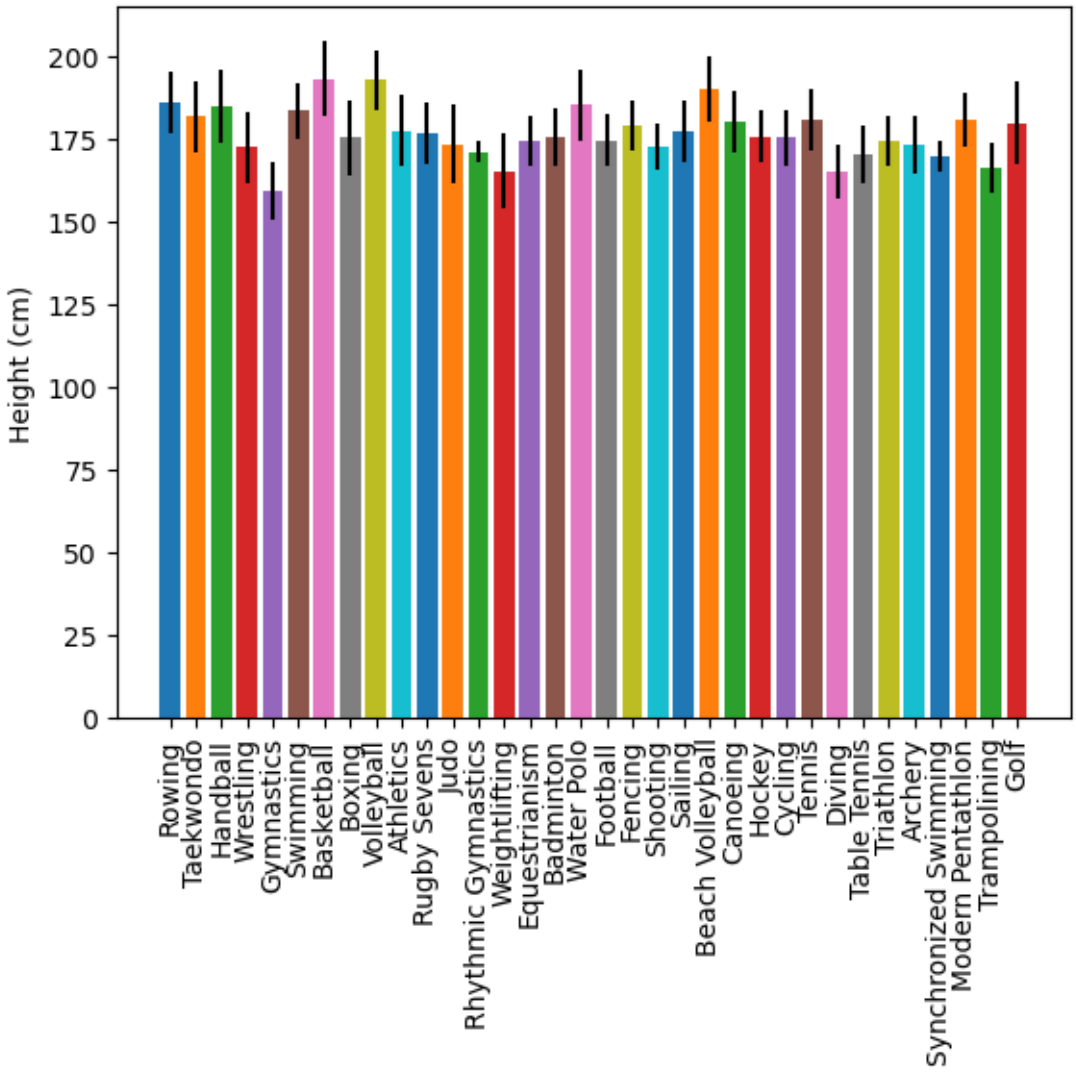
1.4.2 Automatización de figuras

[31]:
```
sports = summer_2016_medals["Sport"].unique()

print(sports)

fig, ax = plt.subplots()
```

```
for sport in sports:
    sport_df = summer_2016_medals[summer_2016_medals["Sport"] == sport]
    ax.bar(sport, sport_df["Height"].mean(),yerr = sport_df["Height"].std())

ax.set_ylabel("Height (cm)")
ax.set_xticklabels(sports, rotation = 90)

plt.show()
```

```
['Rowing' 'Taekwondo' 'Handball' 'Wrestling' 'Gymnastics' 'Swimming'
 'Basketball' 'Boxing' 'Volleyball' 'Athletics' 'Rugby Sevens' 'Judo'
 'Rhythmic Gymnastics' 'Weightlifting' 'Equestrianism' 'Badminton'
 'Water Polo' 'Football' 'Fencing' 'Shooting' 'Sailing' 'Beach Volleyball'
 'Canoeing' 'Hockey' 'Cycling' 'Tennis' 'Diving' 'Table Tennis'
 'Triathlon' 'Archery' 'Synchronized Swimming' 'Modern Pentathlon'
 'Trampolining' 'Golf']
```

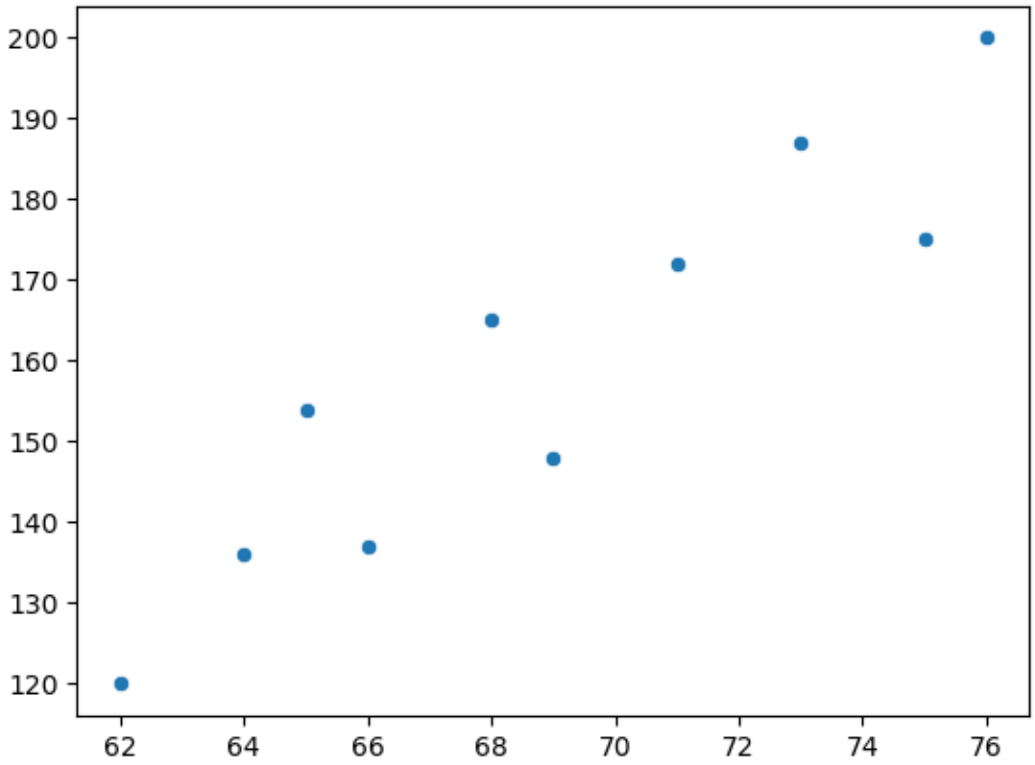
```
<ipython-input-31-3935be168410>:12: UserWarning: FixedFormatter should only be
used together with FixedLocator
  ax.set_xticklabels(sports, rotation = 90)
```



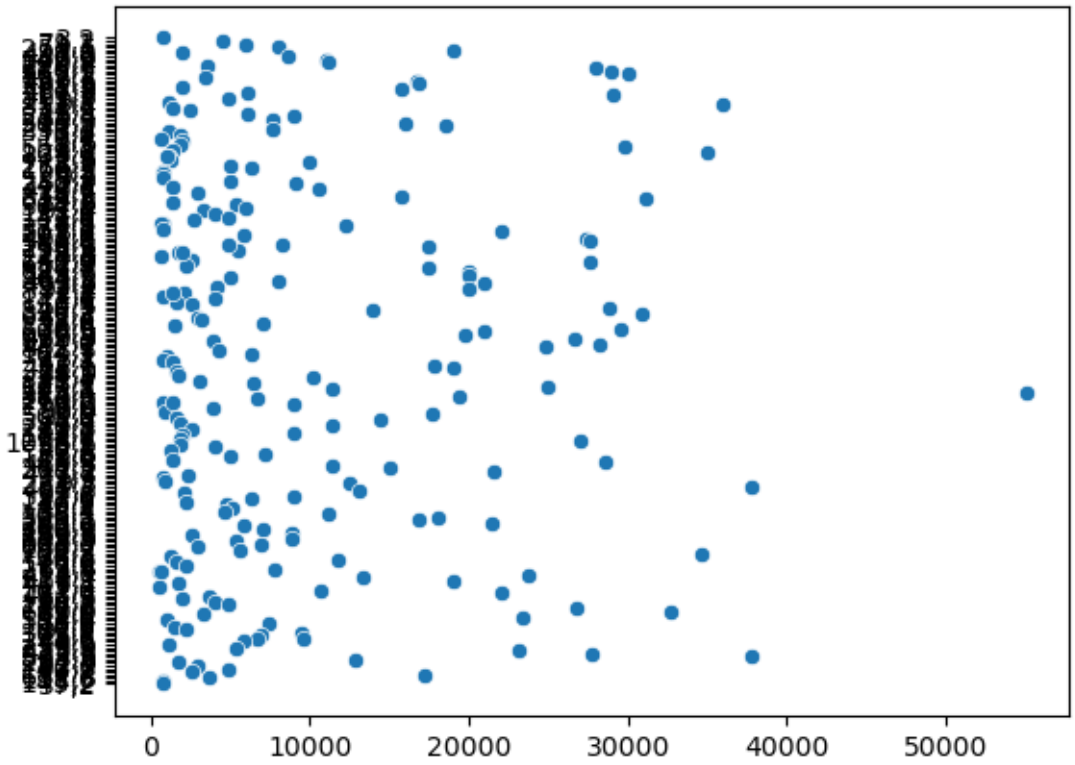
2 INTRODUCCIÓN A SEABORN

2.1 INTRODUCCIÓN A SEABORN

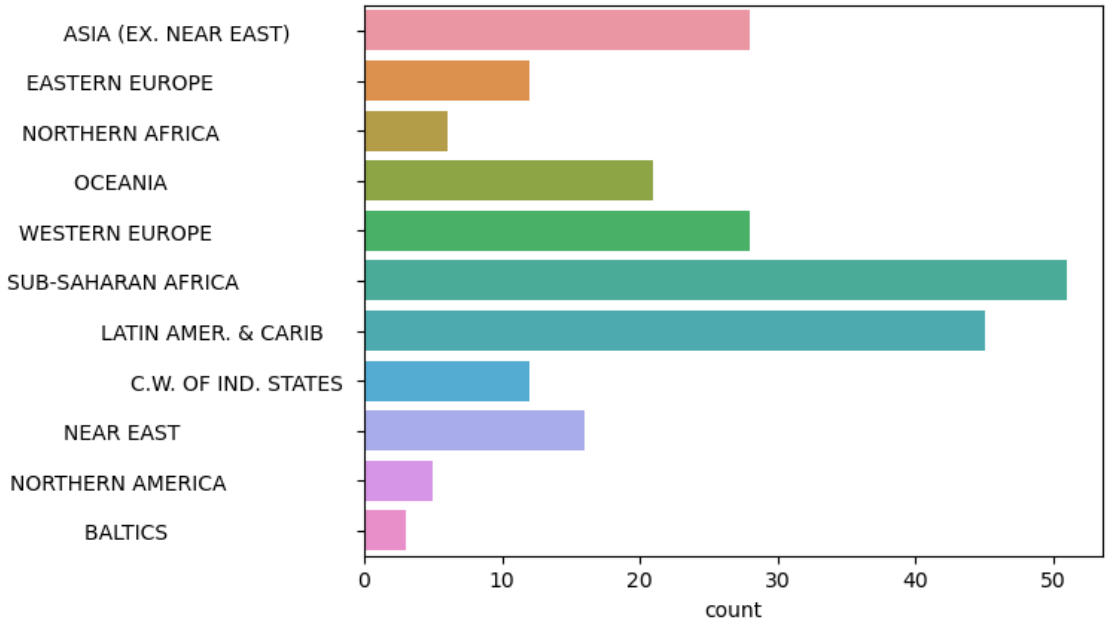
[32]:
```
import seaborn as sns

import matplotlib.pyplot as plt

height = [62,64,69,75,66,68,65,71,76,73]
weight = [120,136,148,175,137,165,154,172,200,187]

sns.scatterplot(x = height, y = weight)
plt.show()
```



[33]:
```
import pandas as pd

countries = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
 ↪countries-of-the-world.csv")

gdp = countries['GDP ($ per capita)'].tolist()
phones = countries['Phones (per 1000)'].tolist()
percent_literate = countries['Literacy (%)'].tolist()
region = countries['Region'].tolist()

sns.scatterplot(x = gdp, y = phones)
plt.show()
```



[34]:
```
# Import Matplotlib and Seaborn
import matplotlib.pyplot as plt
import seaborn as sns


# Create count plot with region on the y-axis
sns.countplot(y=region)

# Show plot
plt.show()
```

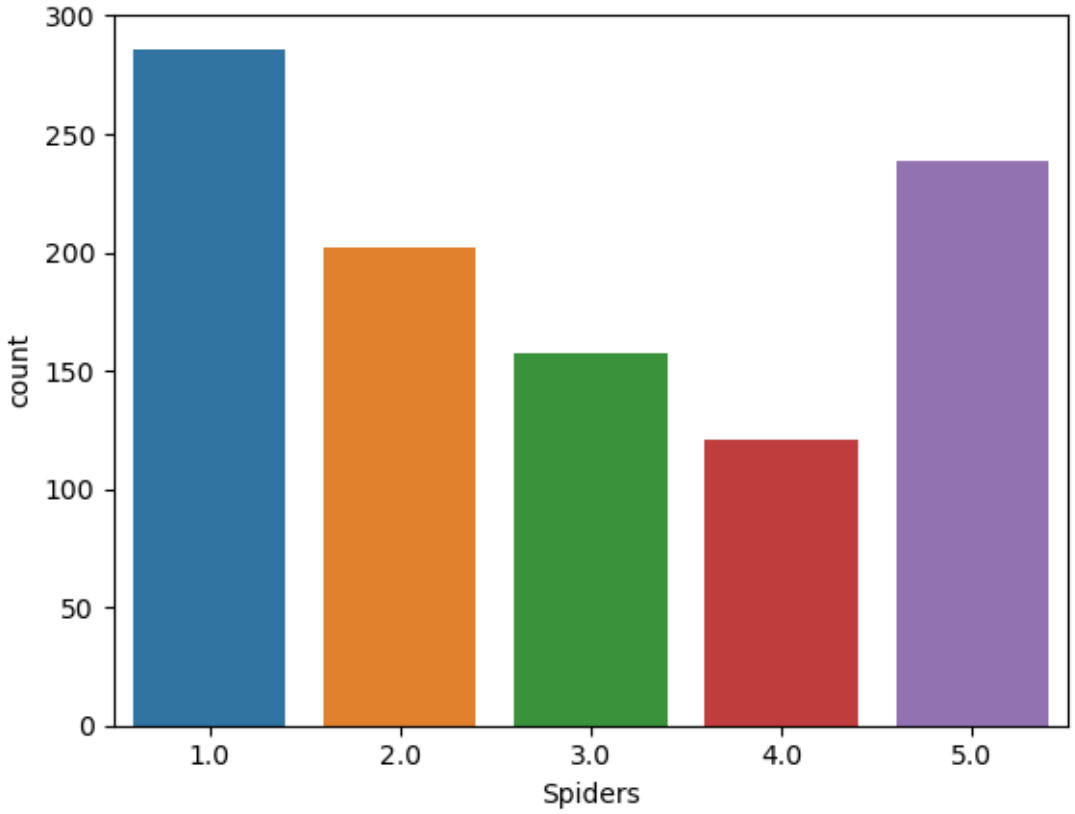


2.1.1 Pandas y Seaborn

[35]:
```
# Para hacer un countplot sin una lista:

survey_data = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
 ↪young-people-survey-responses.csv", index_col = 0)

print(survey_data.head())

# Create a count plot with "Spiders" on the x-axis
sns.countplot(x = 'Spiders', data=survey_data)

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

```
   Music  Techno  Movies  History  Mathematics  Pets  Spiders  Loneliness  \
0    5.0     1.0     5.0      1.0          3.0   4.0      1.0         3.0
1    4.0     1.0     5.0      1.0          5.0   5.0      1.0         2.0
2    5.0     1.0     5.0      1.0          5.0   5.0      1.0         5.0
3    5.0     2.0     5.0      4.0          4.0   1.0      5.0         5.0
4    5.0     2.0     5.0      3.0          2.0   1.0      1.0         3.0

   Parents' advice   Internet usage  Finances   Age  Siblings  Gender  \
0              4.0  few hours a day       3.0  20.0       1.0  female
1              2.0  few hours a day       3.0  19.0       2.0  female
2              3.0  few hours a day       2.0  20.0       2.0  female
```

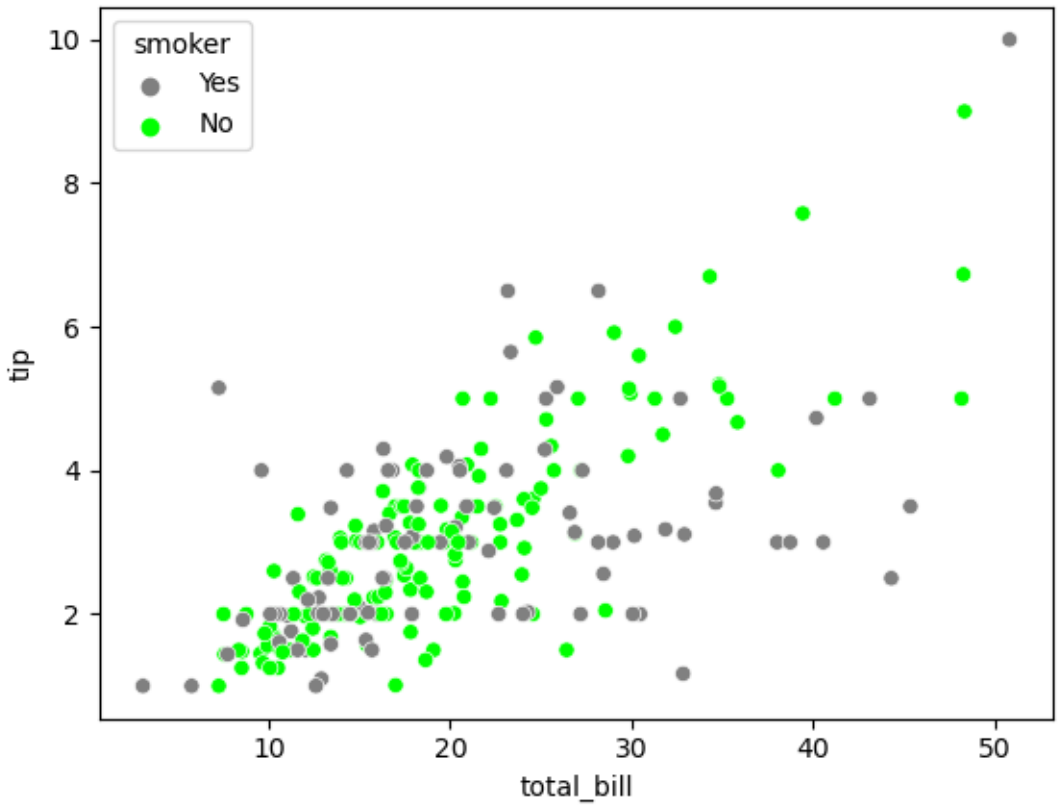
```
3              2.0  most of the day       2.0  22.0       1.0  female
4              3.0  few hours a day       4.0  20.0       1.0  female

  Village - town
0        village
1           city
2           city
3           city
4        village
```



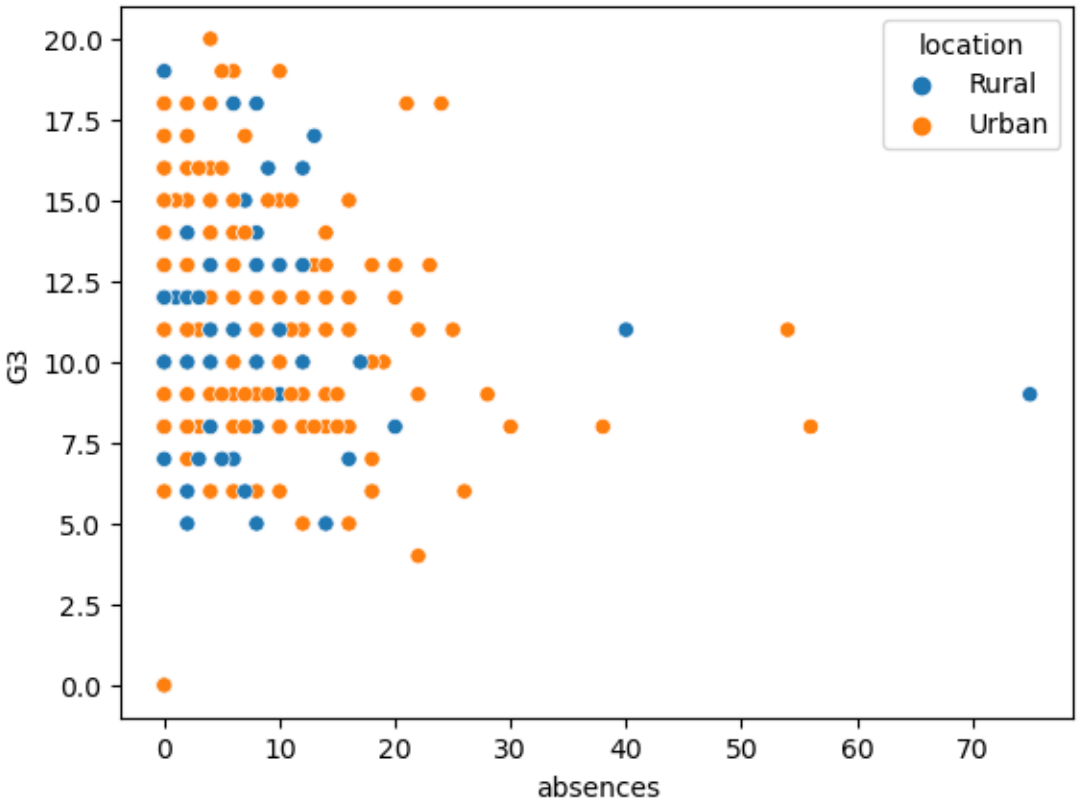
[36]:
```
# Codificando una tercera variable

tips = sns.load_dataset("tips")

print(tips.head())

hue_colors = {"Yes": "#808080",
              "No": "#00FF00"}

sns.scatterplot(x = "total_bill", y = "tip", data = tips, hue = "smoker",␣
 ↪hue_order=["Yes", "No"], palette = hue_colors)
```

```
plt.show()
```

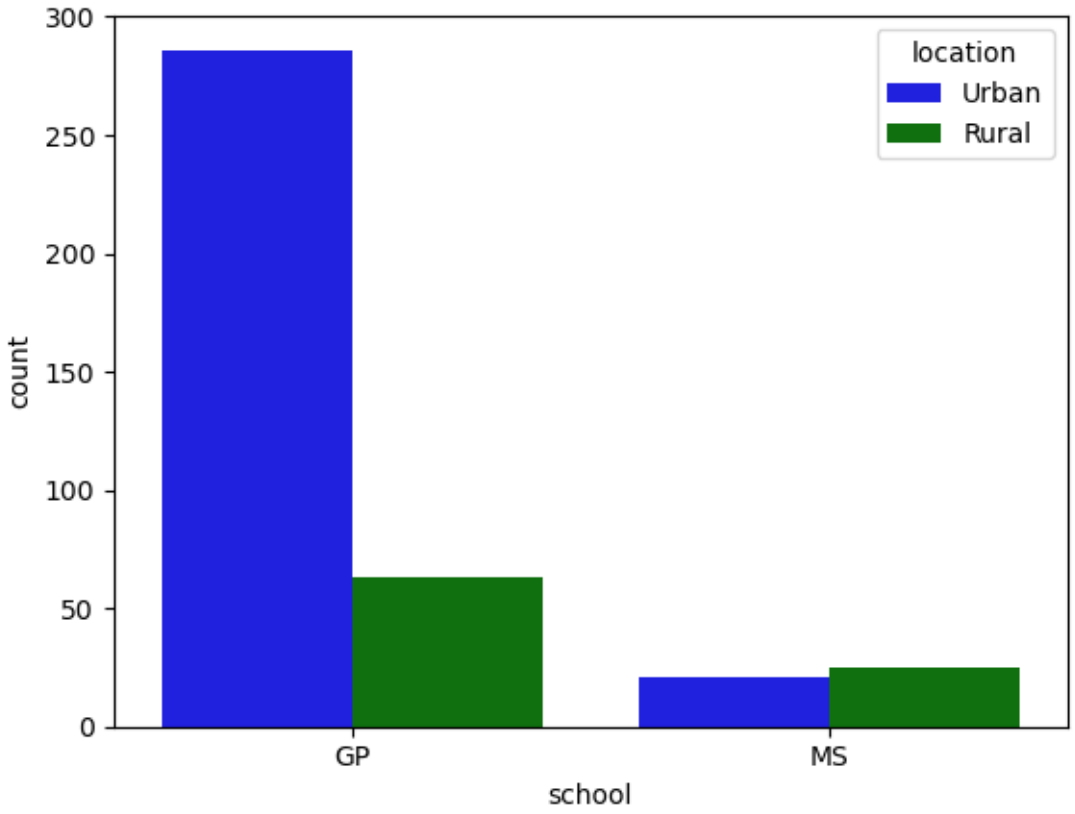
```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female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



[37]:
```
student_data = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
 ↪student-alcohol-consumption.csv")

# Create a scatter plot of absences vs. final grade
sns.scatterplot(x = "absences", y = "G3", data = student_data, hue =␣
 ↪"location", hue_order = (["Rural", "Urban"]))

# Show plot
plt.show()
```



[38]:
```
# Create a dictionary mapping subgroup values to colors
palette_colors = {"Rural": "green", "Urban": "blue"}

# Create a count plot of school with location subgroups
sns.countplot(x = "school", data = student_data, hue = "location", palette =␣
↪palette_colors)

# Display plot
plt.show()
```

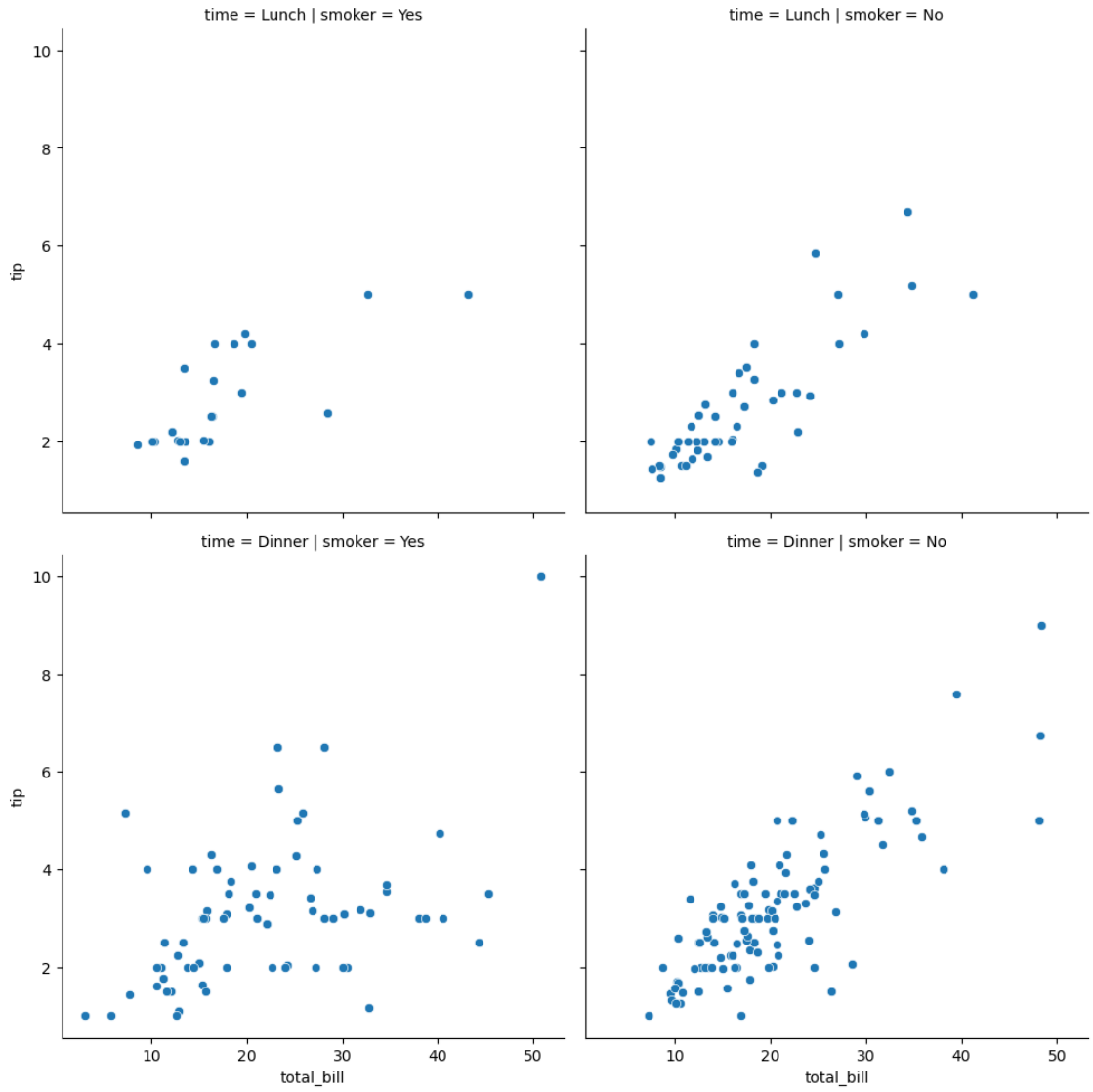


2.2 DOS VARIABLES CUANTITATIVAS

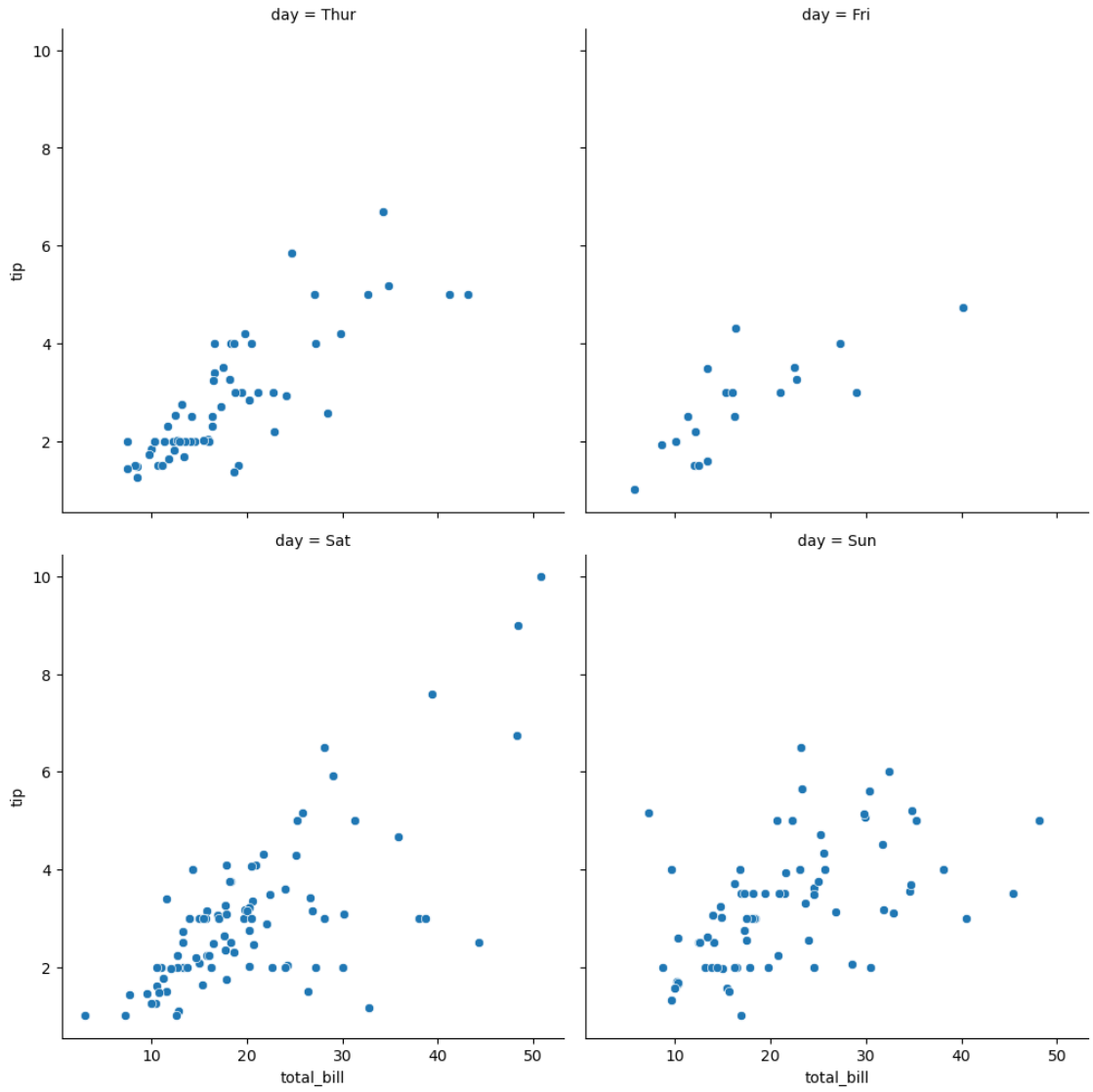
relplot() permite visualizar la relación entre dos varibales cuantitativas usando scatterplots o diagramas de línea.

2.2.1 Scatterplots

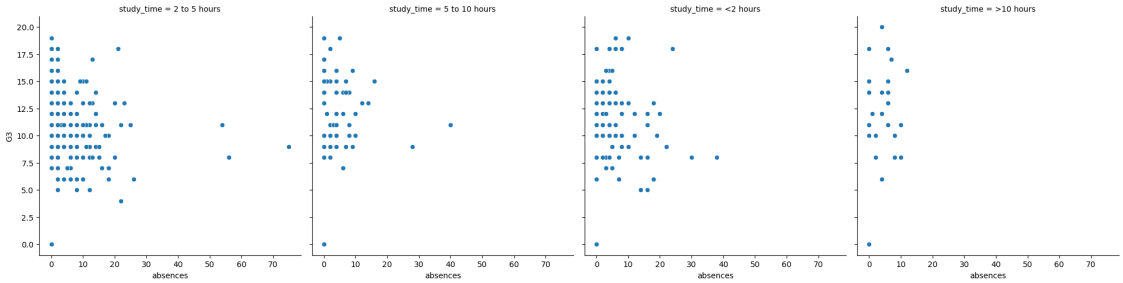
[39]:
```
sns.relplot(x = "total_bill", y = "tip", data = tips, kind = "scatter", col =␣
↪"smoker", row = "time")
# donde col hará el arreglo horizontal, row vertical
plt.show()
```



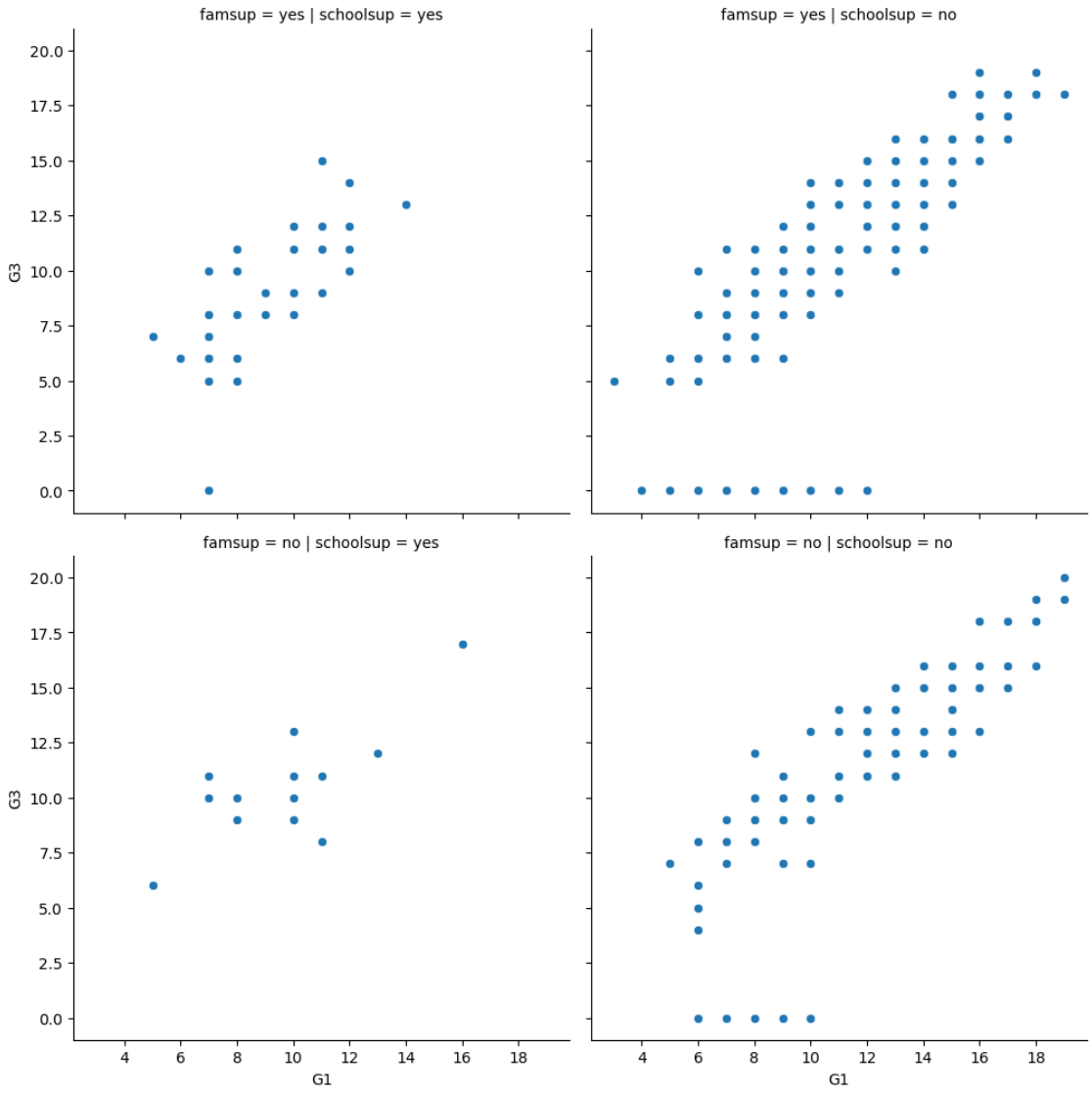
[40]:
```
sns.relplot(x = "total_bill", y = "tip", data = tips, kind = "scatter", col =␣
↪"day", col_wrap = 2, \
            col_order = ["Thur", "Fri", "Sat", "Sun"])
# col_wrap organiza el display de gráficos si son muchos y col_order los ordena
plt.show()
```



[41]:
```
# Change to make subplots based on study time
sns.relplot(x="absences", y="G3",
            data=student_data,
            kind="scatter", col = "study_time")

# Show plot
plt.show()
```



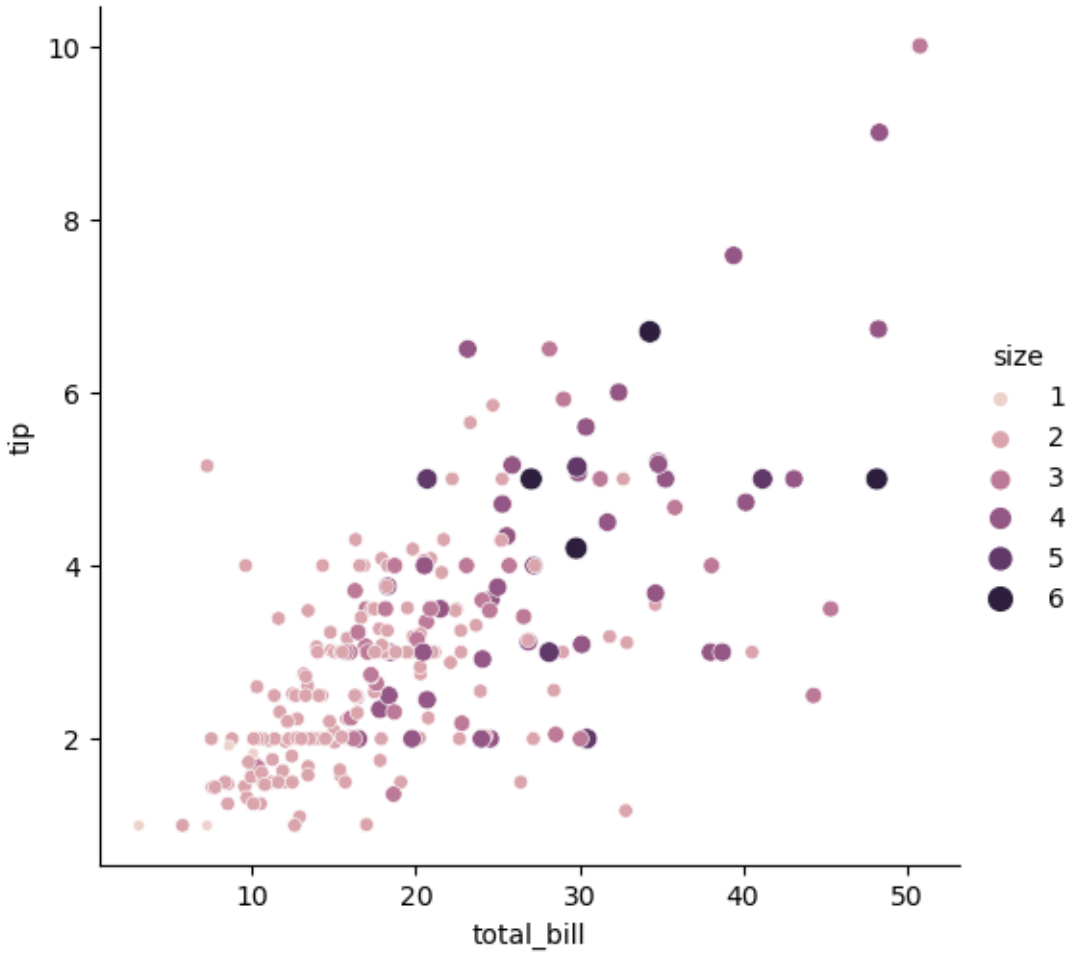
[42]:
```
# Adjust further to add subplots based on family support
sns.relplot(x="G1", y="G3",
            data=student_data,
            kind="scatter",
            col="schoolsup",
            row = "famsup",
            col_order=["yes", "no"],
            row_order = ["yes", "no"])

# Show plot
plt.show()
```



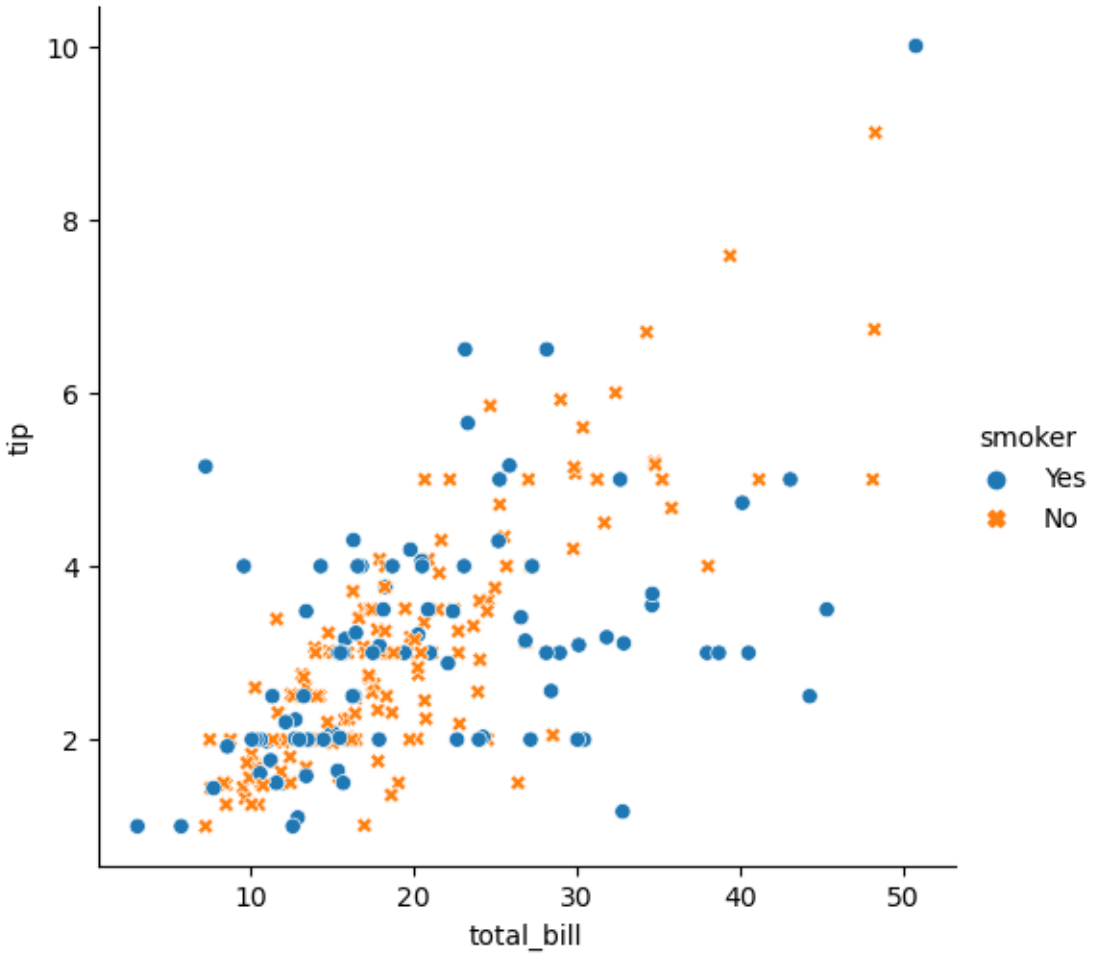
2.2.2 Personalización

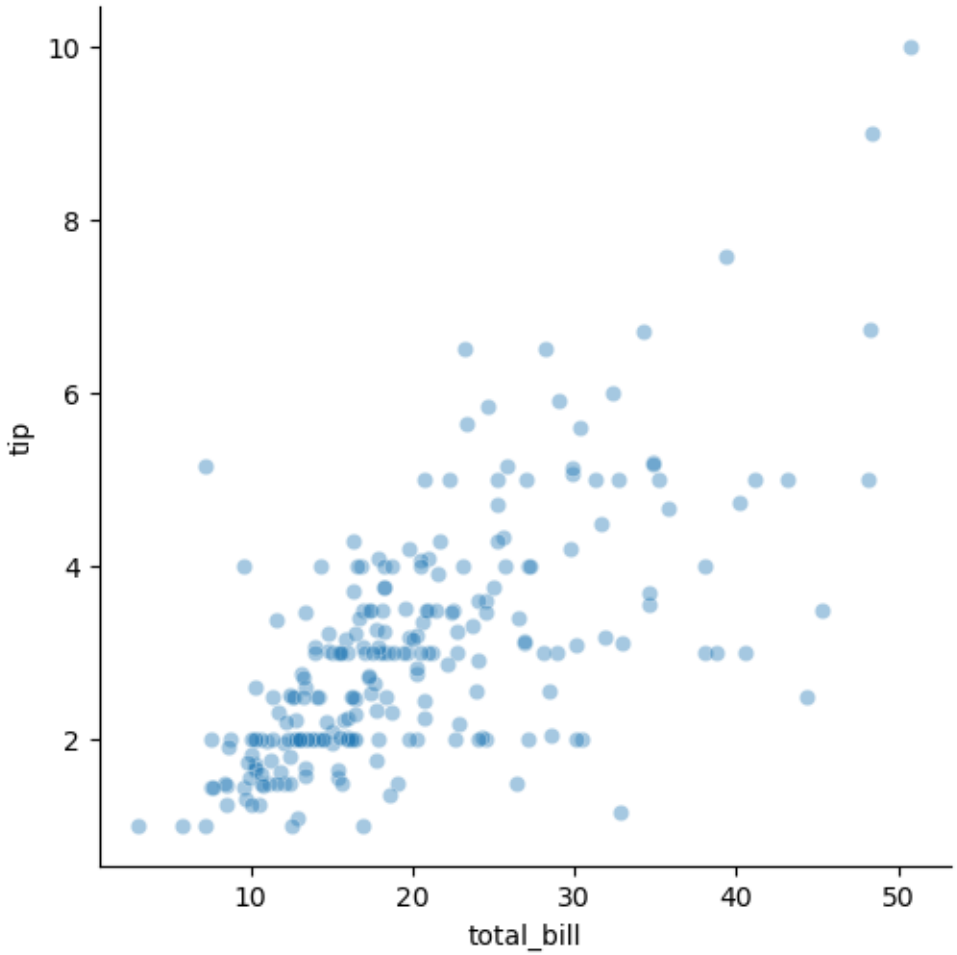
[43]:
```
# Tamaño del punto

sns.relplot(x = "total_bill", y = "tip", data = tips, kind = "scatter", size =␣
↪"size", hue = "size")
plt.show()

# Estilo del punto

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

```
# Transparencia
sns.relplot(x = "total_bill", y = "tip", data = tips, kind = "scatter", alpha =␣
↪0.4)
```

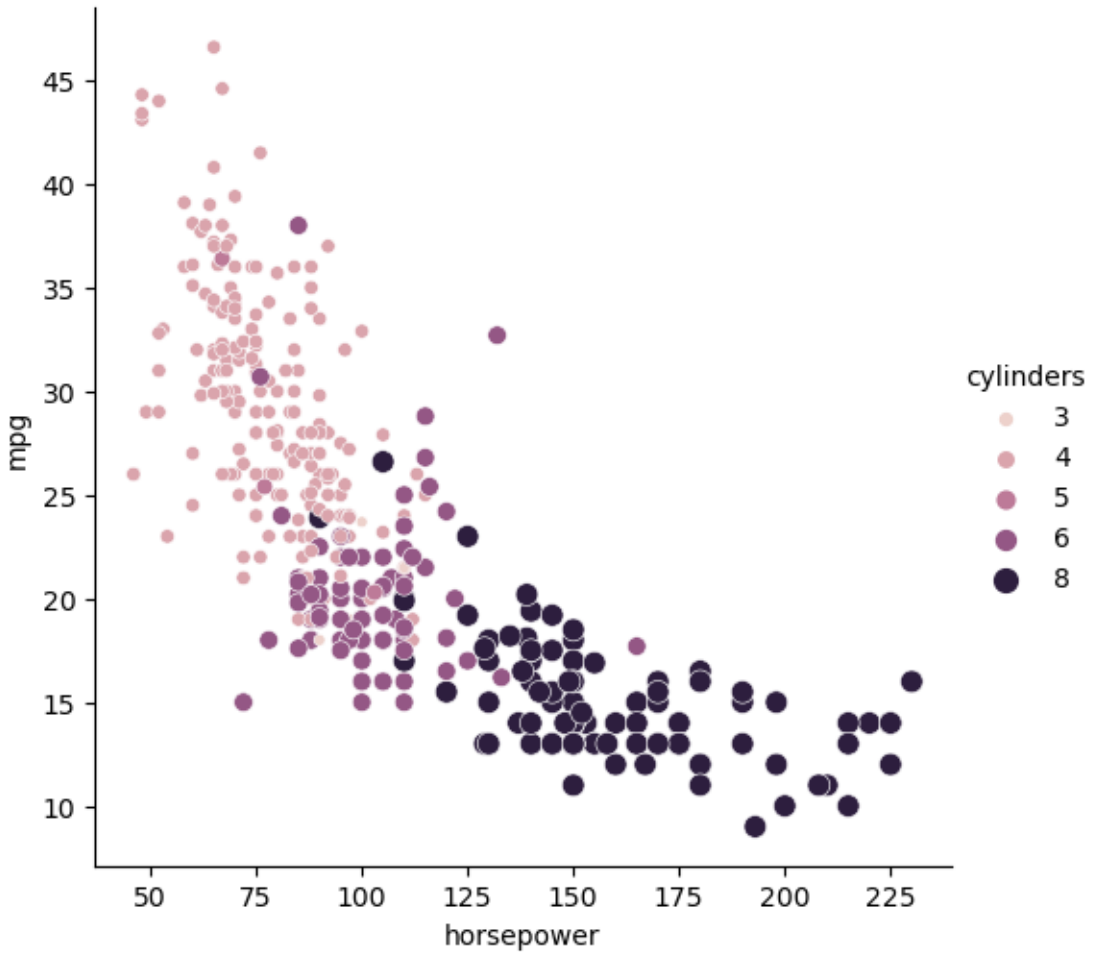


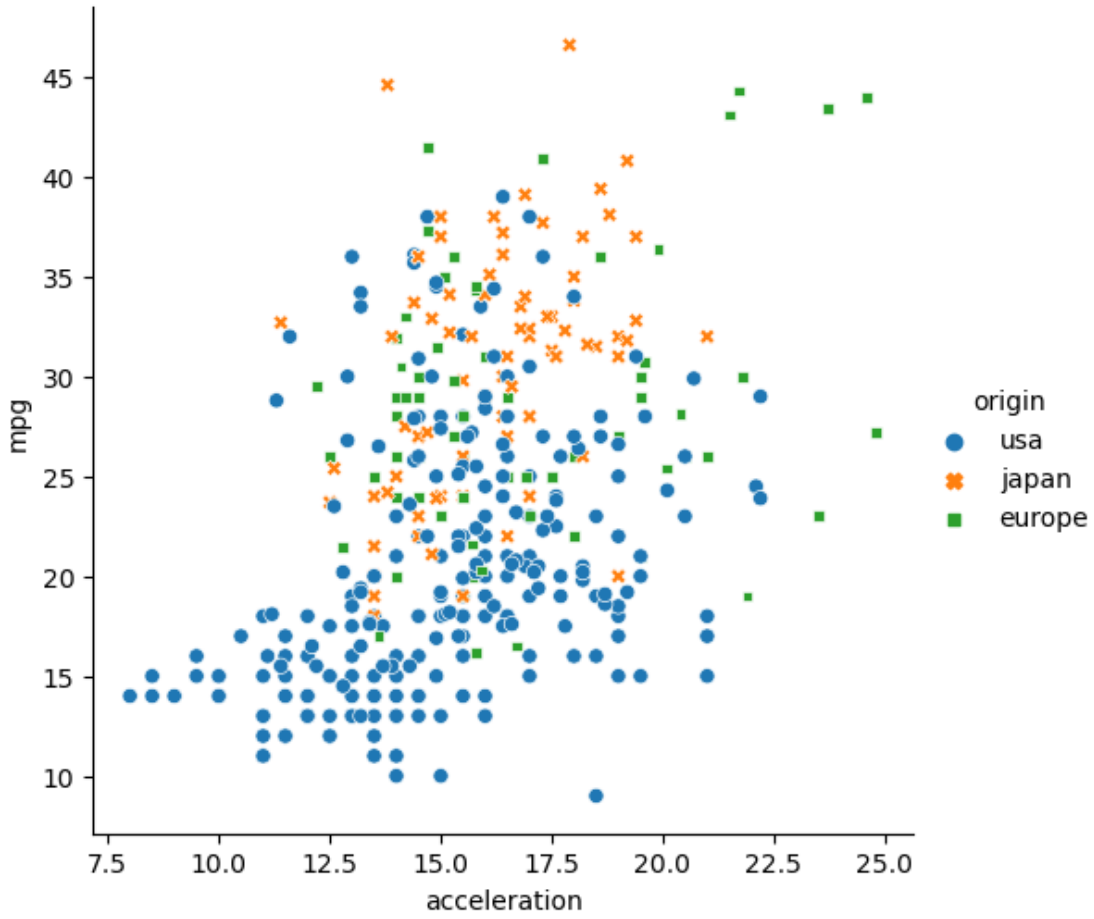
[43]: <seaborn.axisgrid.FacetGrid at 0x1bd2fb7e8e0>


10
8
6
4
2
tip
10
20
30
40
50
total_bill
smoker
Yes
No



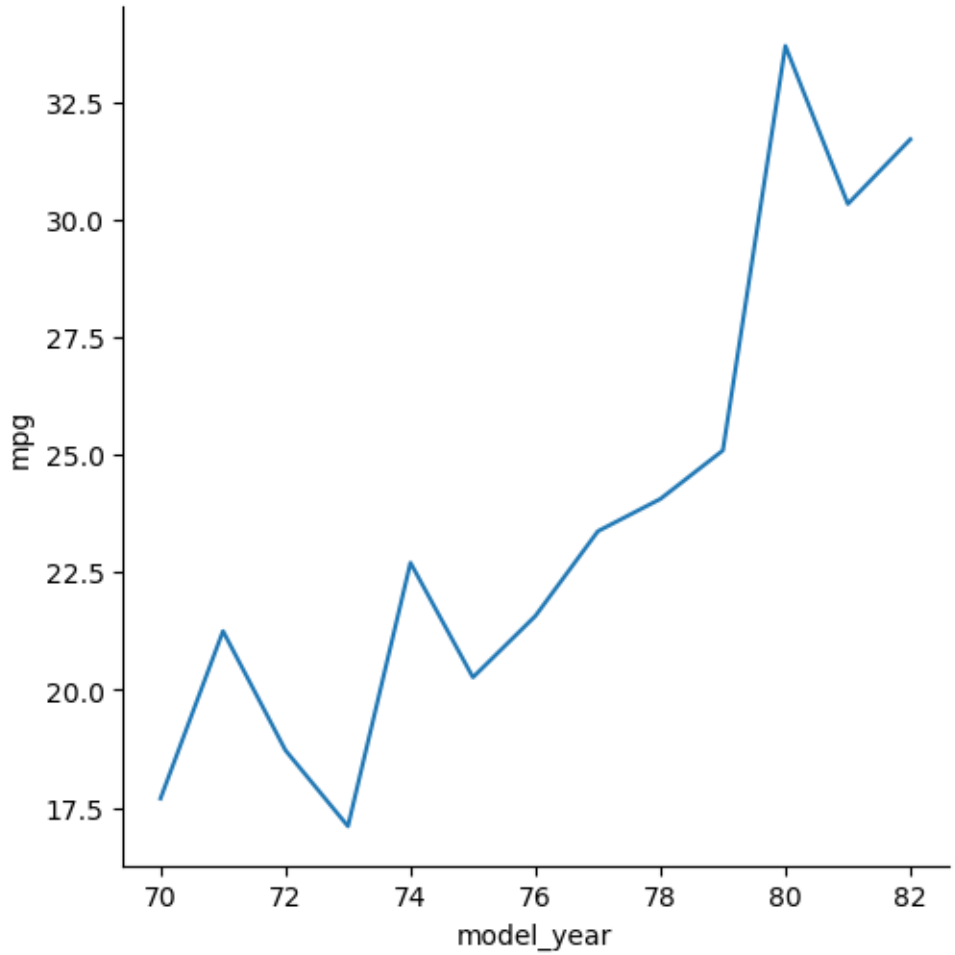
[44]:
```
# Ejemplo

mpg = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/mpg.csv")
sns.relplot(x="horsepower", y="mpg", data=mpg, kind="scatter",␣
 ↪size="cylinders", hue = "cylinders")
plt.show()

###
sns.relplot(x = "acceleration", y = "mpg", data = mpg, kind = "scatter", style␣
 ↪= "origin", hue = "origin")
plt.show()
```

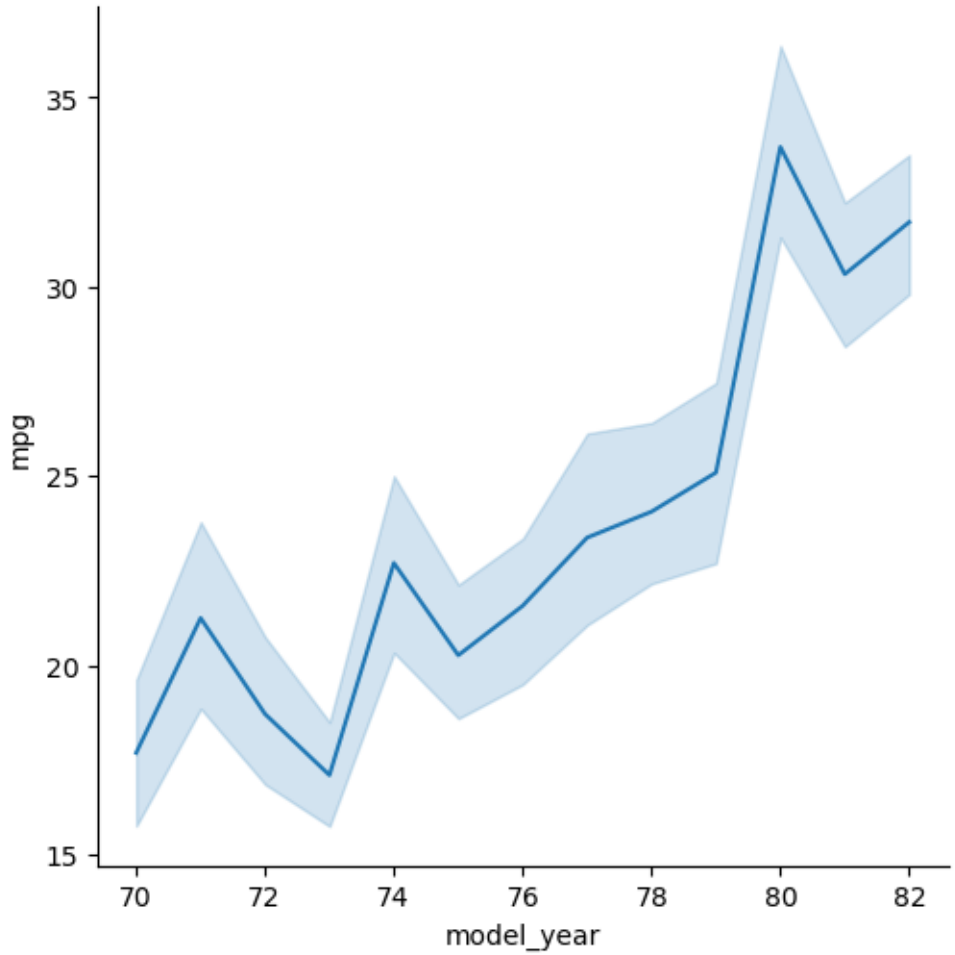

mpg
horsepower
cylinders
3
4
5
6
8
45
40
35
30
25
20
15
10
50
75
100
125
150
175
200
225

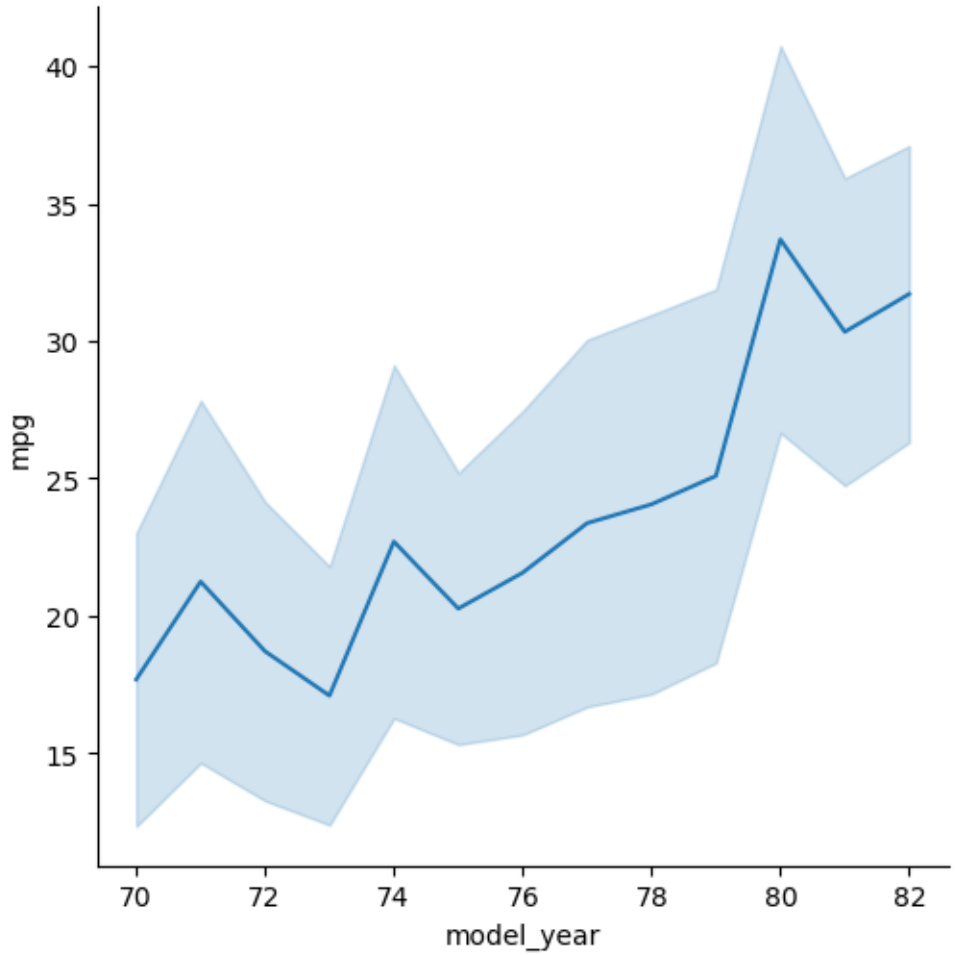


2.2.3 Lineplots

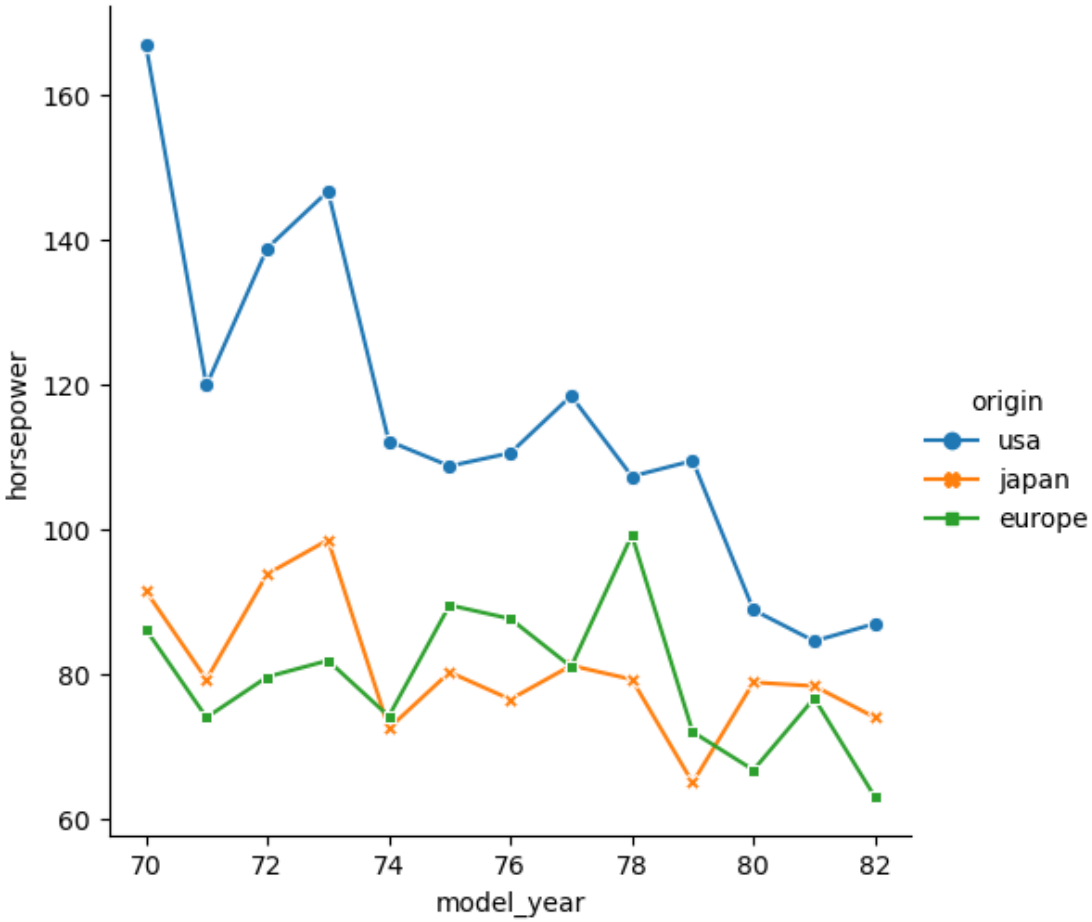
[45]:
```
# En estos ejemplos, cada x tiene varios valores

sns.relplot(x='model_year', y='mpg', data=mpg, kind='line', ci = None) #␣
 ↪desactiva el intervalo de confianza
plt.show()

sns.relplot(x='model_year', y='mpg', data=mpg, kind='line')
plt.show()

# Y para mostrar la desviación estándar:
sns.relplot(x="model_year", y="mpg", data=mpg, kind="line", ci='sd');
```


32.5
30.0
27.5
25.0
22.5
20.0
17.5
mpg
70
72
74
76
78
80
82
model_year


35
30
25
20
15
mpg
70
72
74
76
78
80
82
model_year



[46]:
```
# Change to create subgroups for country of origin
sns.relplot(x="model_year", y="horsepower", data=mpg, kind="line", ci=None,␣
 ↪style="origin", \
            hue="origin", markers = True, dashes = False)

# Show plot
plt.show()
```

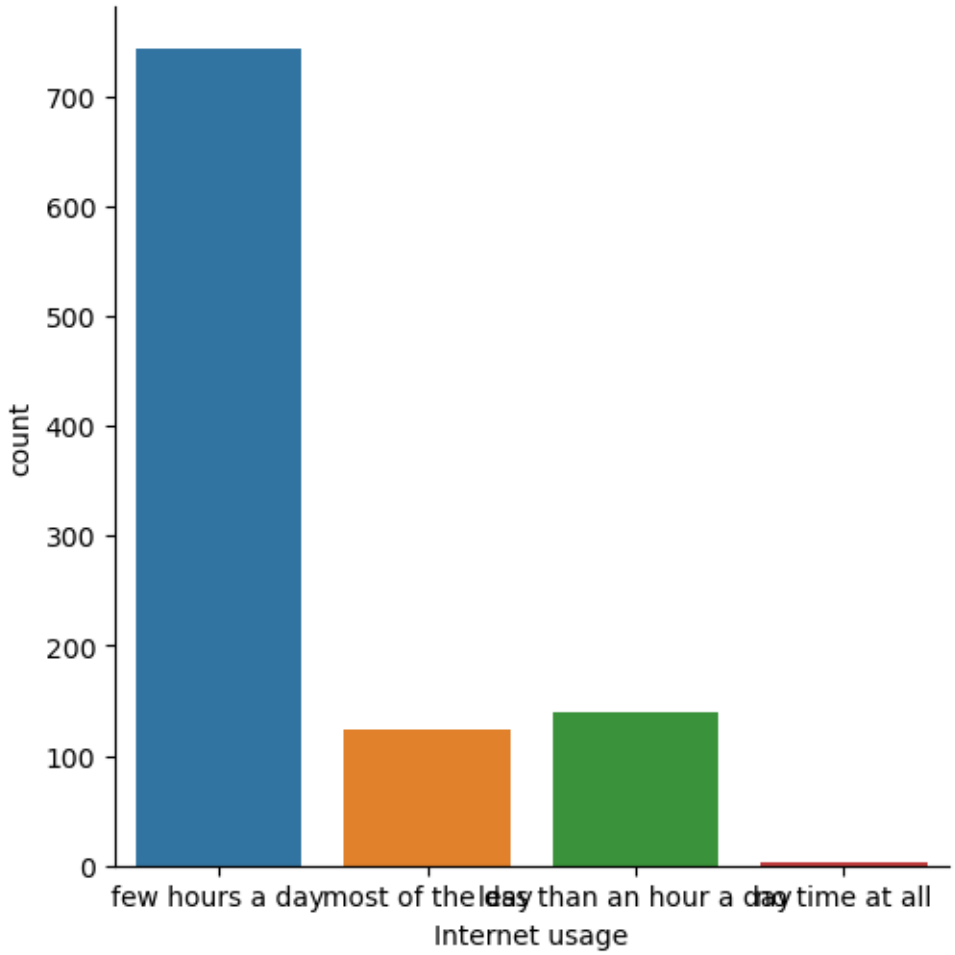


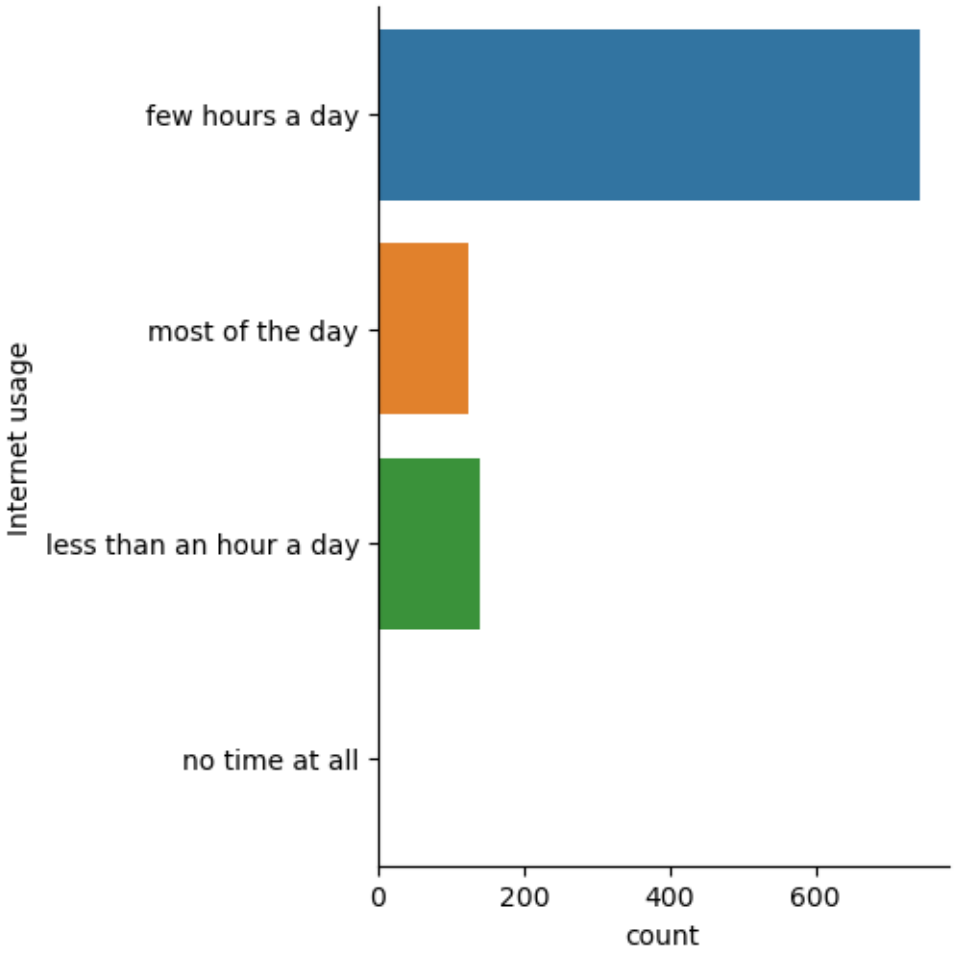
2.3 VARIABLES CATEGÓRICAS Y CUANTITATIVAS

catplot() es el equivalente de relplot() para variables categóricas

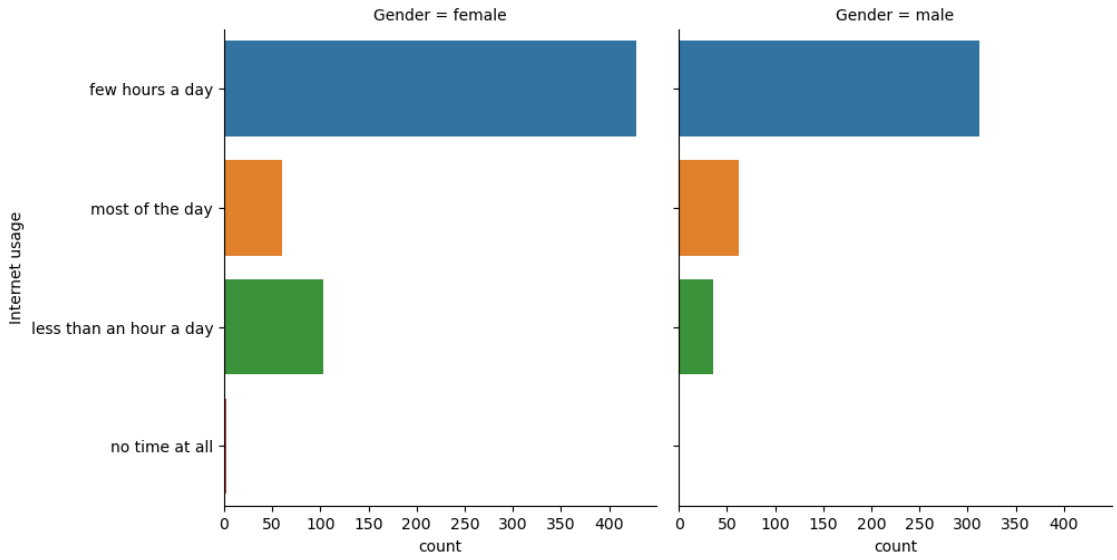
2.3.1 Countplots

[47]:
```
sns.catplot(x='Internet usage', data=survey_data, kind='count', aspect=1)
plt.show()

###

sns.catplot(y="Internet usage", data=survey_data,kind="count")
plt.show()

###

sns.catplot(y="Internet usage", data=survey_data, kind="count", col='Gender')
```


700
600
500
400
300
200
100
0
count
few hours a day
most of the day
less than an hour a day
not time at all
Internet usage

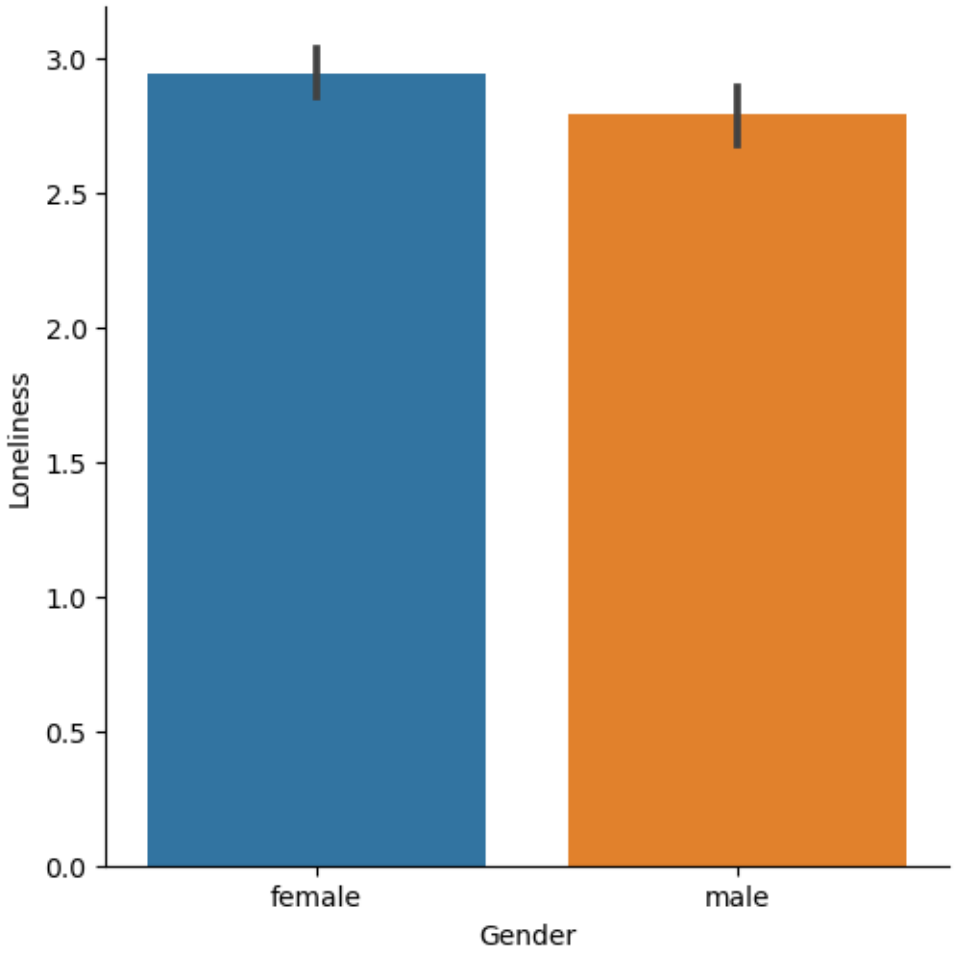


[47]: <seaborn.axisgrid.FacetGrid at 0x1bd2fbc2400>

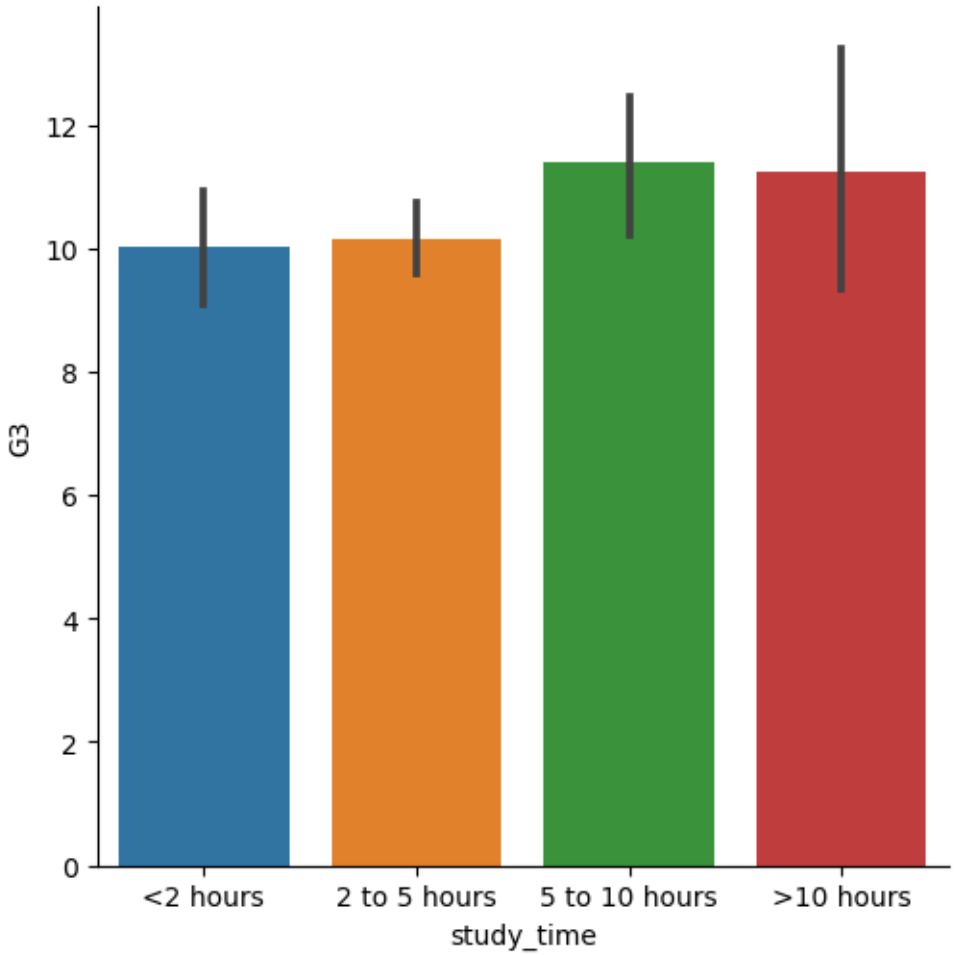


2.3.2 Barplots

[48]:
```
# Create a bar plot of interest in math, separated by gender
sns.catplot(x = "Gender", y = "Loneliness", data = survey_data, kind = "bar")
# Show plot
plt.show()
```



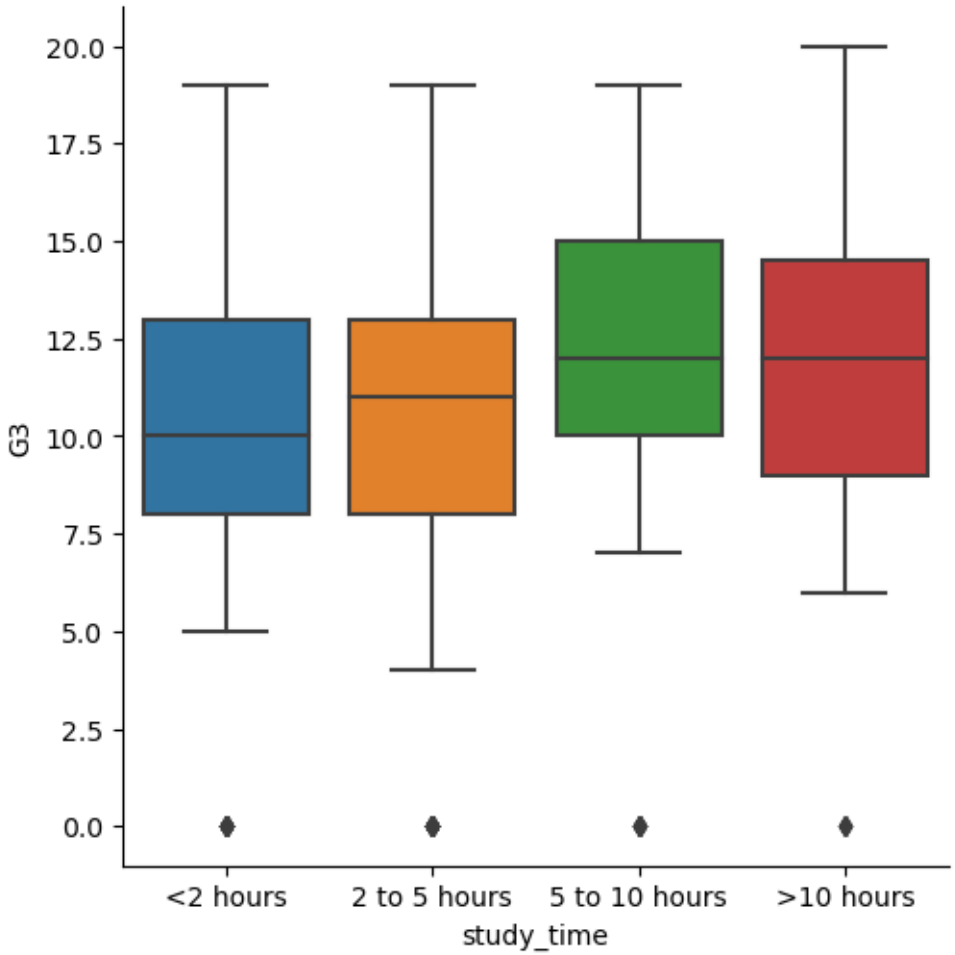
[49]:
```
# Create bar plot of average final grade in each study category
sns.catplot(x="study_time", y="G3", data=student_data, kind="bar",
            order = ['<2 hours', '2 to 5 hours', '5 to 10 hours', '>10 hours']);
# Show plot
plt.show()
```



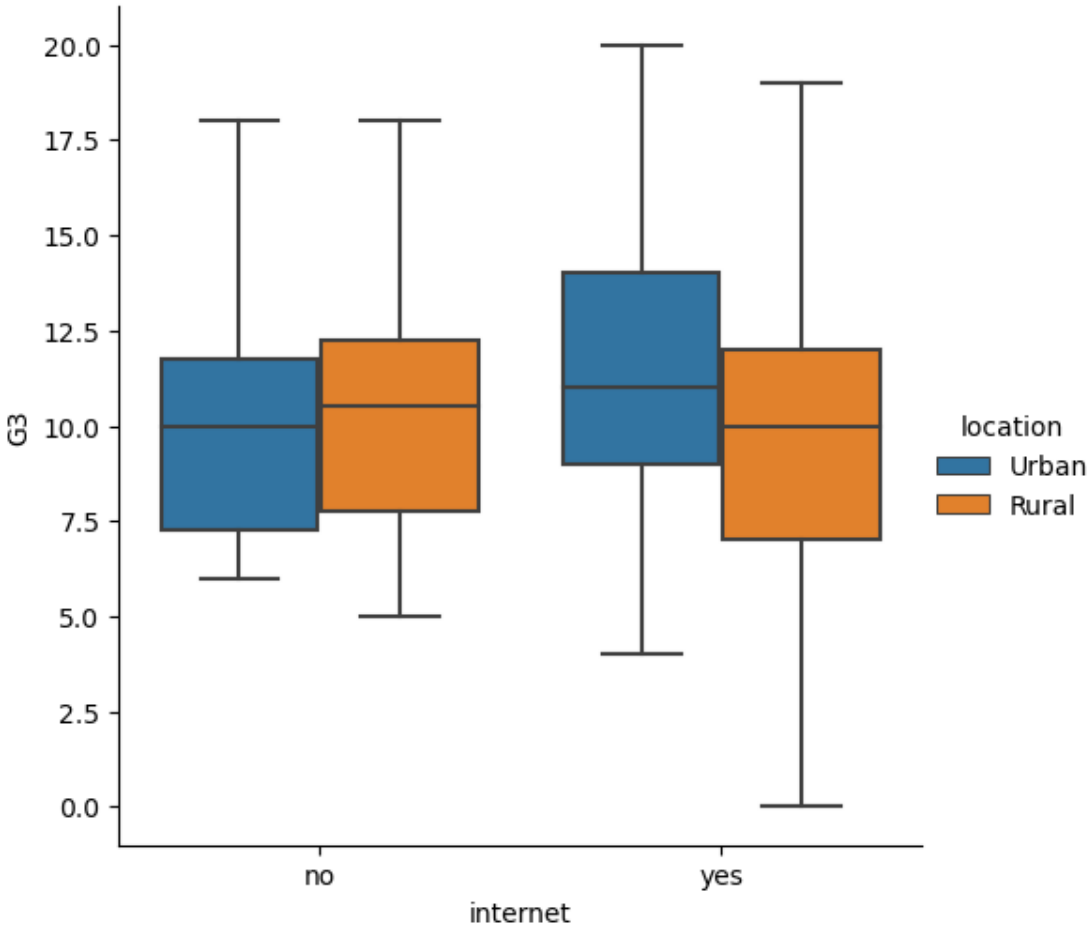
2.3.3 Boxplots

[50]:
```
# Specify the category ordering
study_time_order = ["<2 hours", "2 to 5 hours",
                    "5 to 10 hours", ">10 hours"]

# Create a box plot and set the order of the categories
sns.catplot(x='study_time', y='G3',
            data=student_data,
            kind='box',
            order=study_time_order)
```

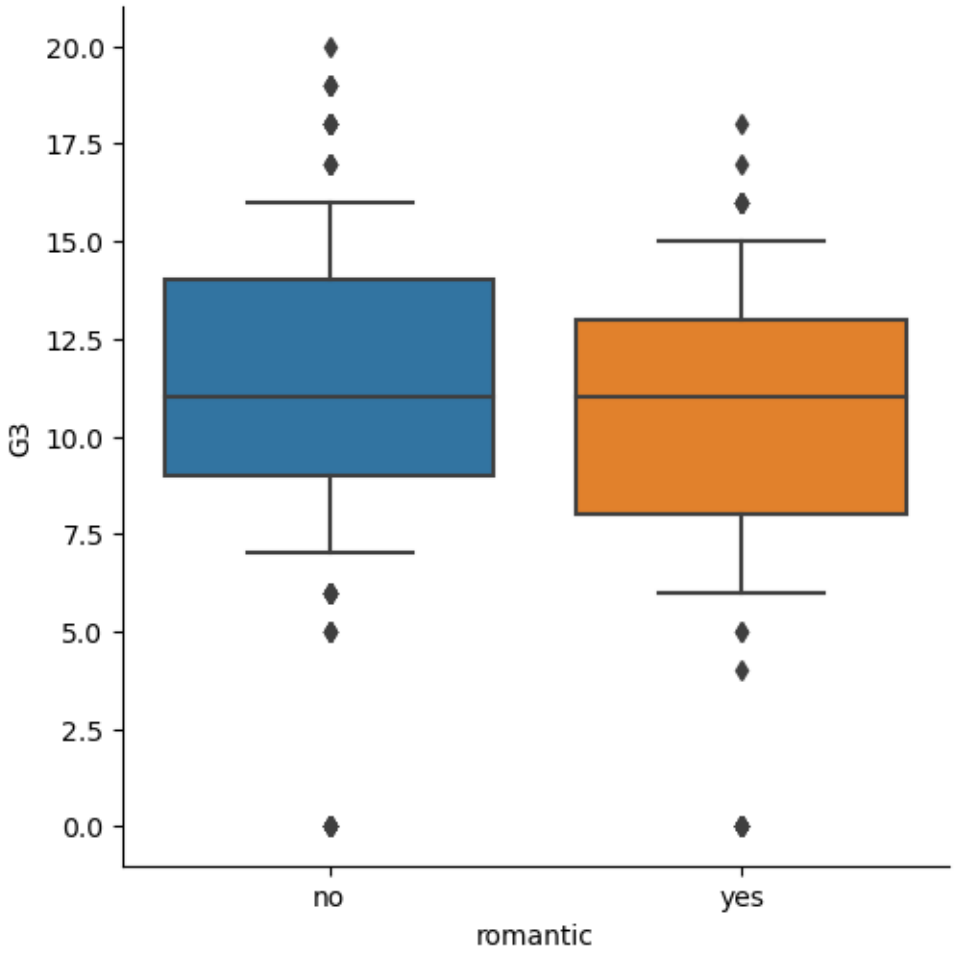
[50]: <seaborn.axisgrid.FacetGrid at 0x1bd2dbc3280>



[51]:
```
# Create a box plot with subgroups and omit the outliers
sns.catplot(x = "internet",
            y = "G3",
            data = student_data,
            kind = "box",
            hue = "location",
            sym = "")

# Show plot
plt.show()
```



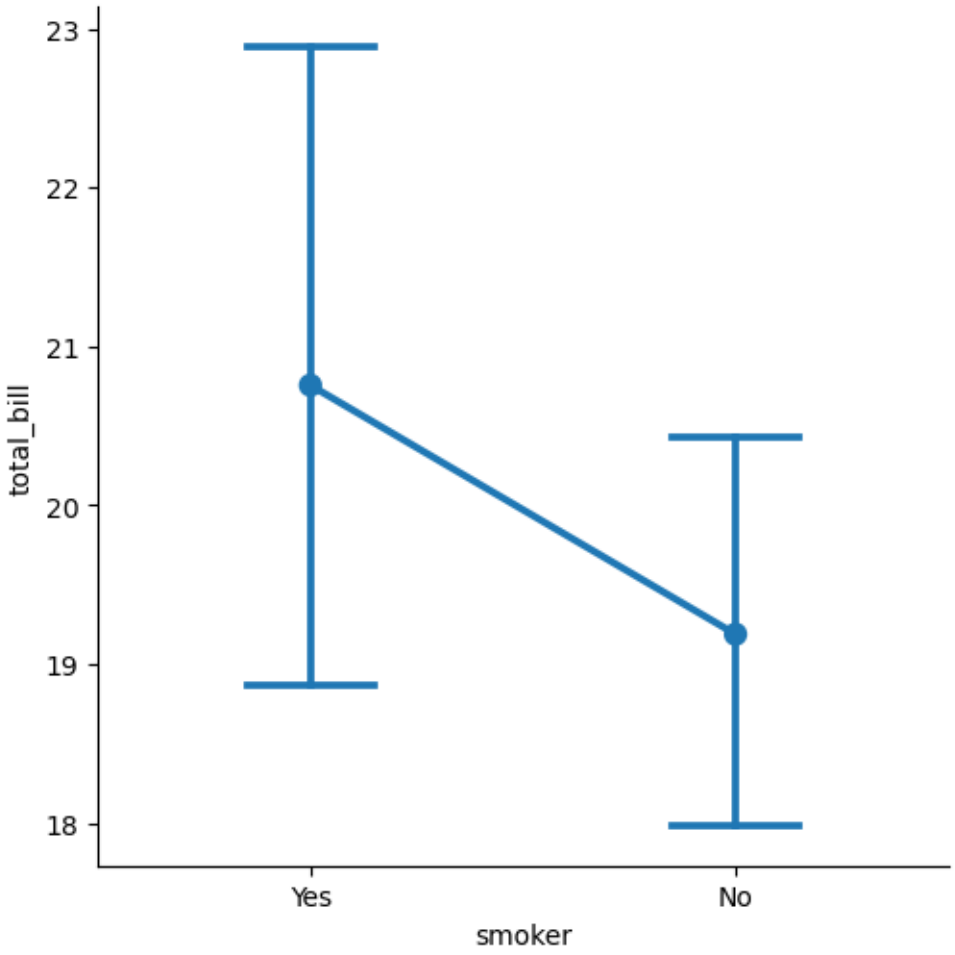
[52]:
```
# Set the whiskers to 0.5 * IQR
sns.catplot(x="romantic", y="G3",
            data=student_data,
            kind="box",
            whis=0.5);
```



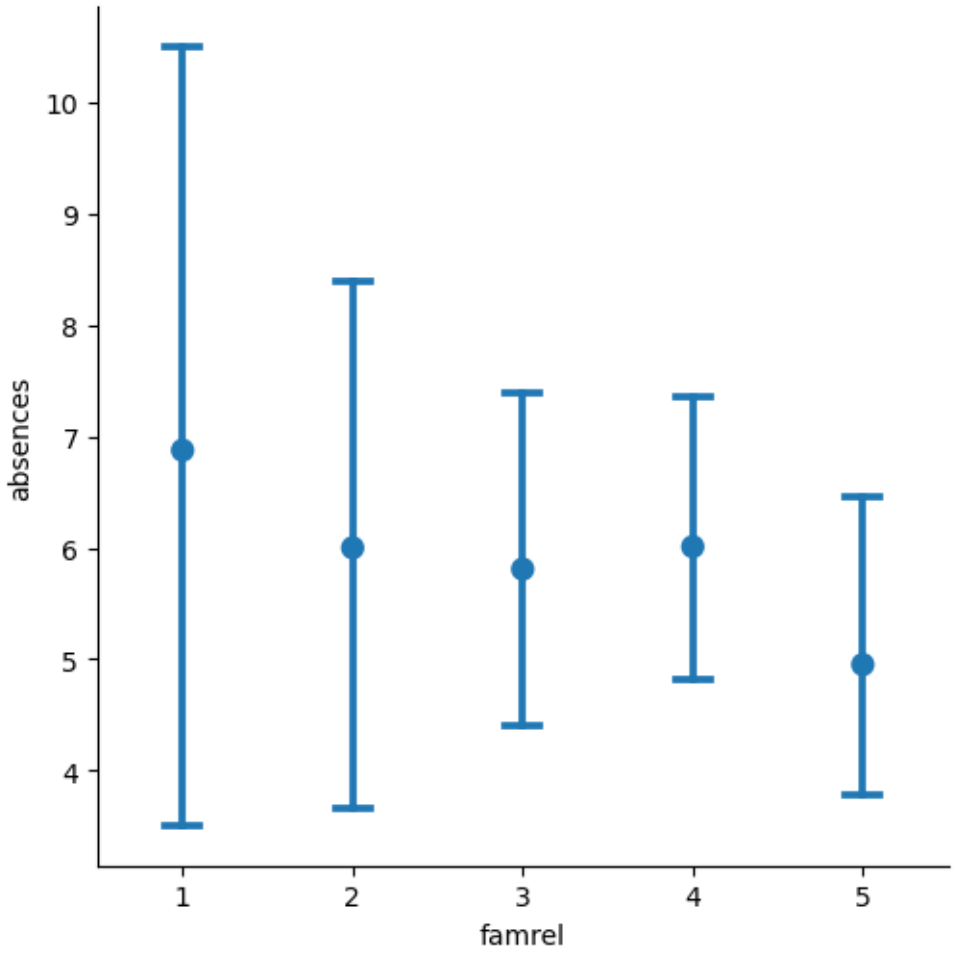
2.3.4 Pointplots

A diferencia de los lineplots, en donde ambas variables suelen ser cuantitativas, en un pointplot una variable es categórica (normalmente la x).

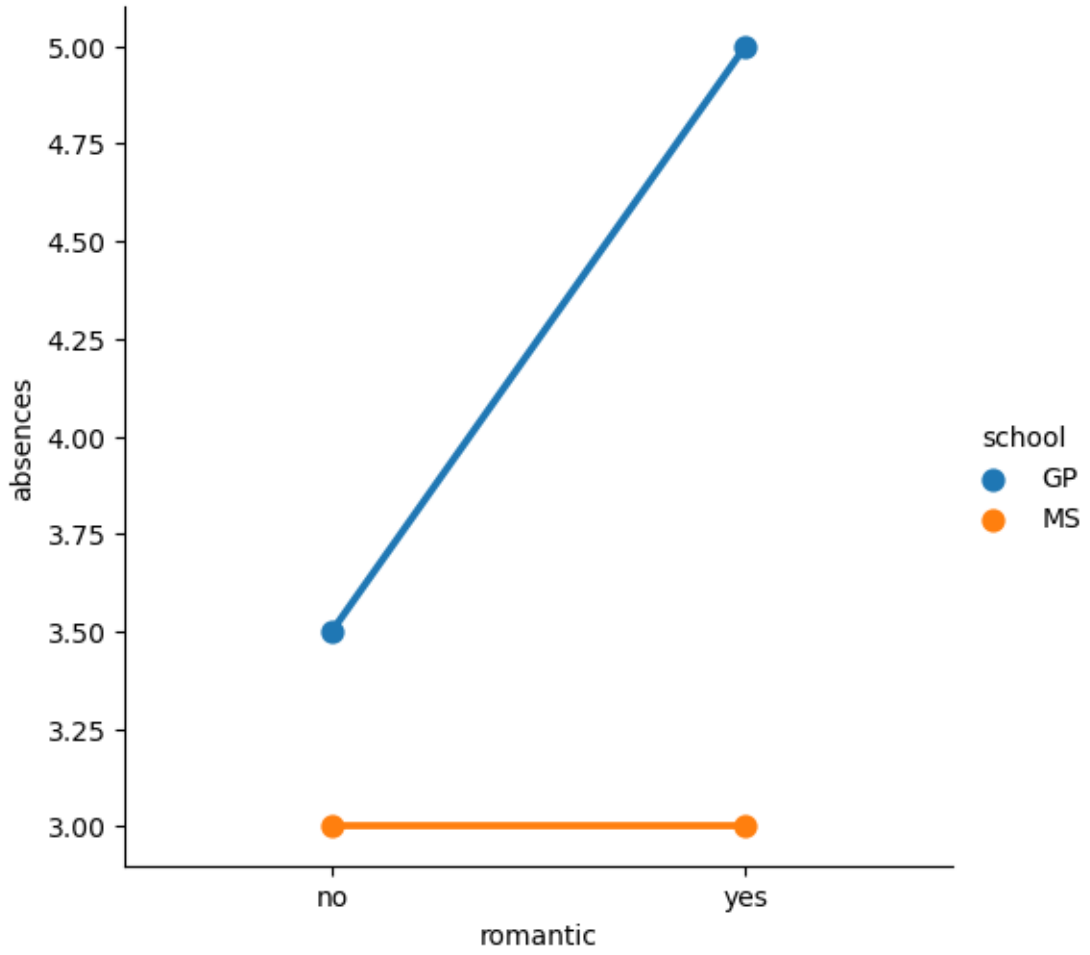
[53]:
```
sns.catplot(x = "smoker", y = "total_bill", data = tips, kind = "point",␣
↪capsize = 0.3)
plt.show()
```



[54]:
```
sns.catplot(x="famrel", y="absences",
            data=student_data,
            kind="point",
            capsize=0.2, join = False)

# Show plot
plt.show()
```



[55]:
```
# Import median function from numpy
import numpy as np

sns.catplot(x="romantic", y="absences",
            data=student_data,
            kind="point",
            hue="school",
            ci=None,
            estimator = np.median, )

plt.show()
```

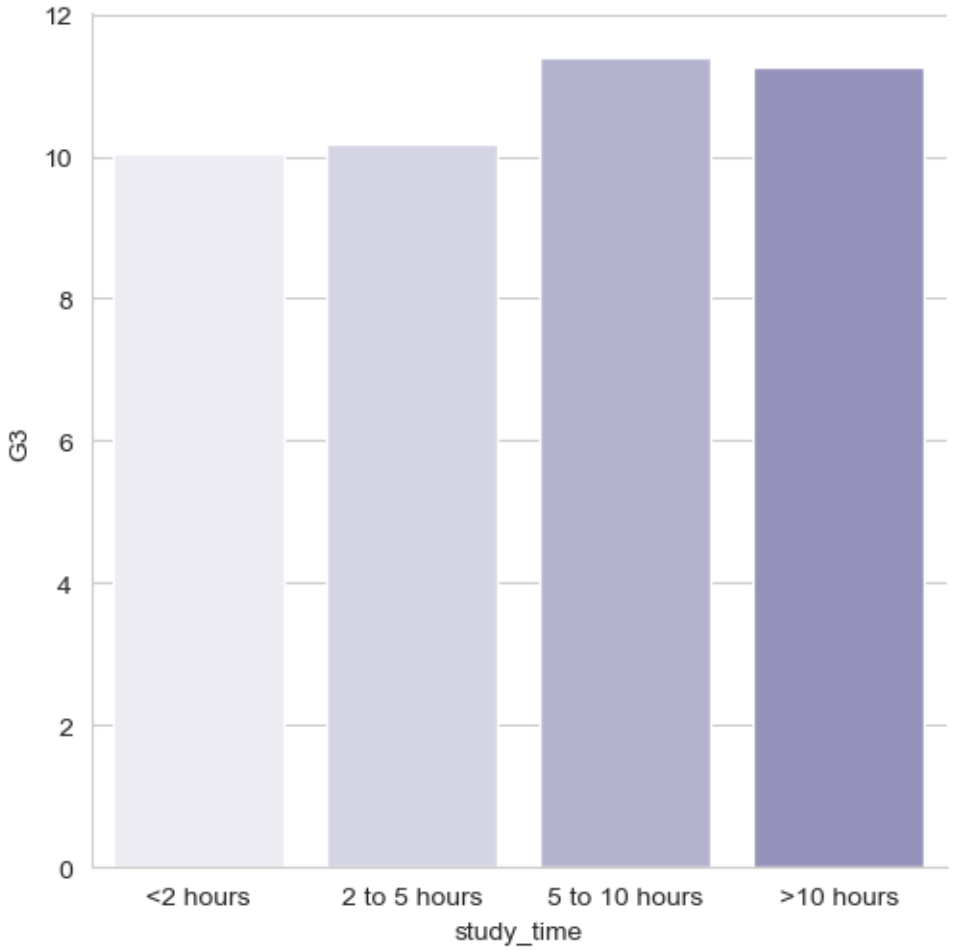


2.4 PERSONALIZACIÓN

2.4.1 Estilo, paletas y contexto

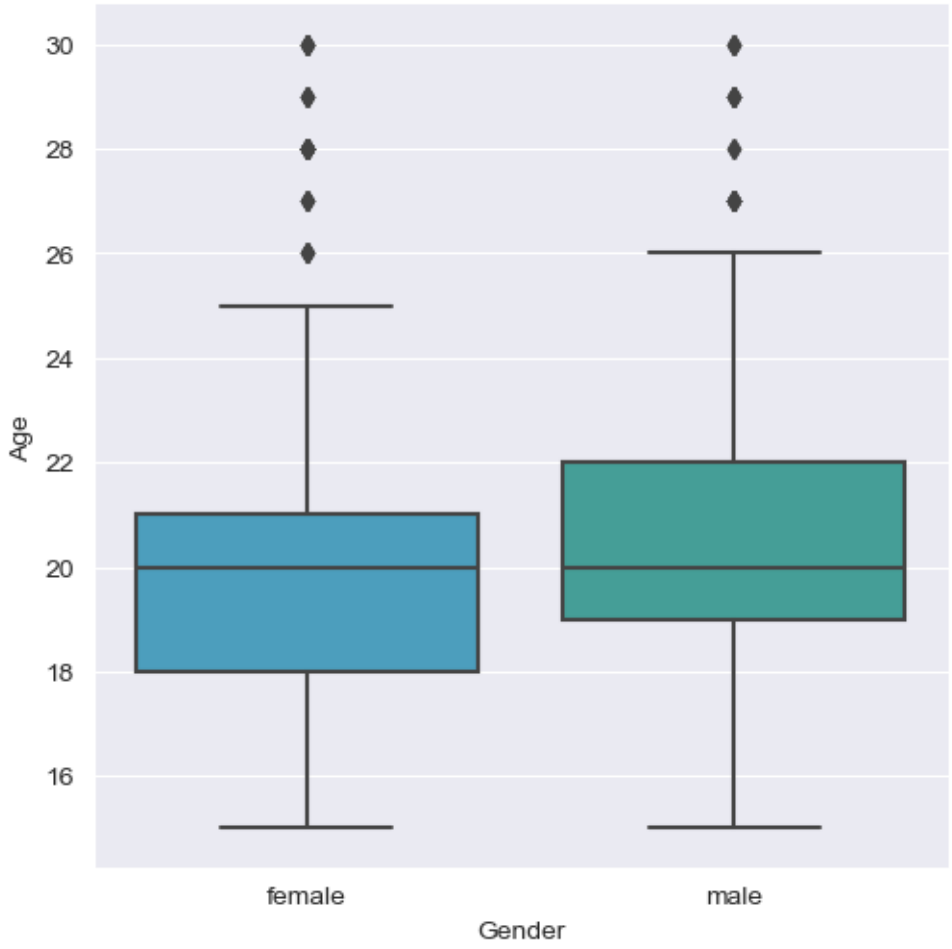
[56]:
```
# Set the color palette to "Purples"
sns.set_style("whitegrid")
sns.set_palette("Purples")
# Plot the relation between student study time and their grades using barplots␣
 ↪and arrange the bar in ascending order w.r.t to study time
# Turn off the confidence intervals
sns.catplot(x="study_time", y="G3",
            data=student_data,
            kind="bar",
            order=["<2 hours",
                   "2 to 5 hours",
                   "5 to 10 hours",
                   ">10 hours"],ci=False)
```

```
# Show plot
plt.show()

# STYLES: https://seaborn.pydata.org/generated/seaborn.set_style.html
# PALETTES: https://medium.com/@morganjonesartist/
 ↪color-guide-to-seaborn-palettes-da849406d44f
# CONTEXTS: https://seaborn.pydata.org/generated/seaborn.set_context.html
```

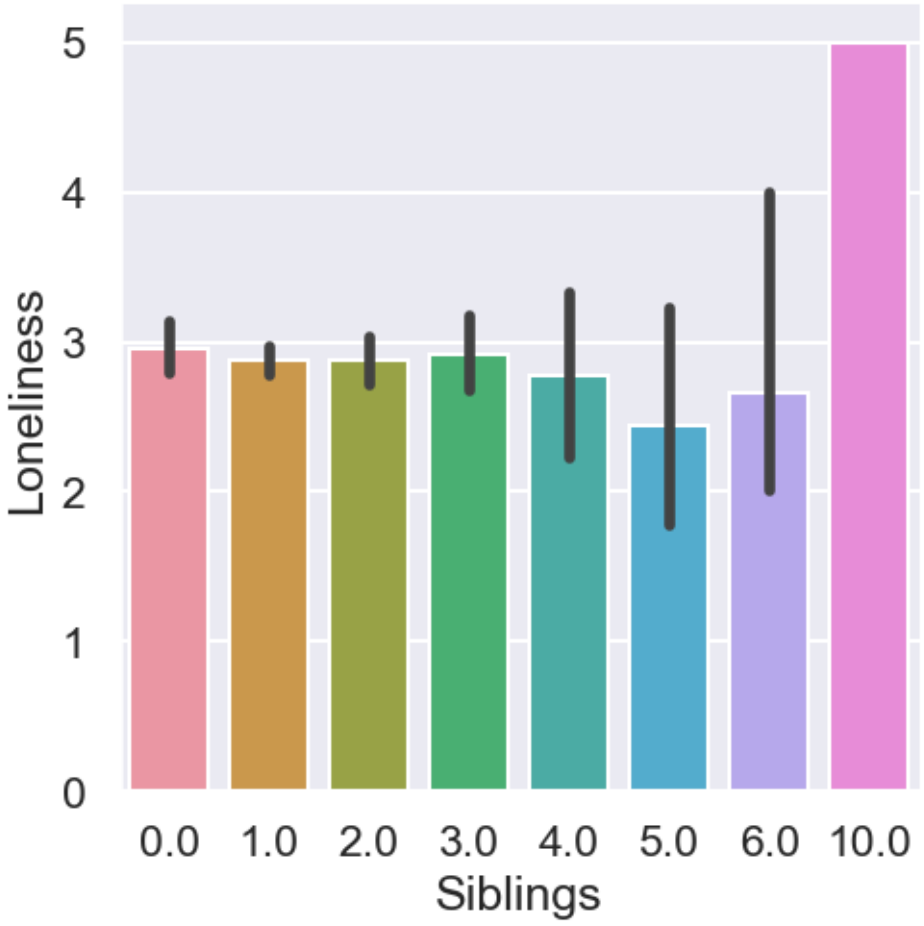


[57]:
```
# Set the style to "darkgrid"
sns.set_style("darkgrid")
sns.set_palette(["#39A7D0","#36ADA4"])

# Set a custom color palette


# Create the box plot of age distribution by gender
```

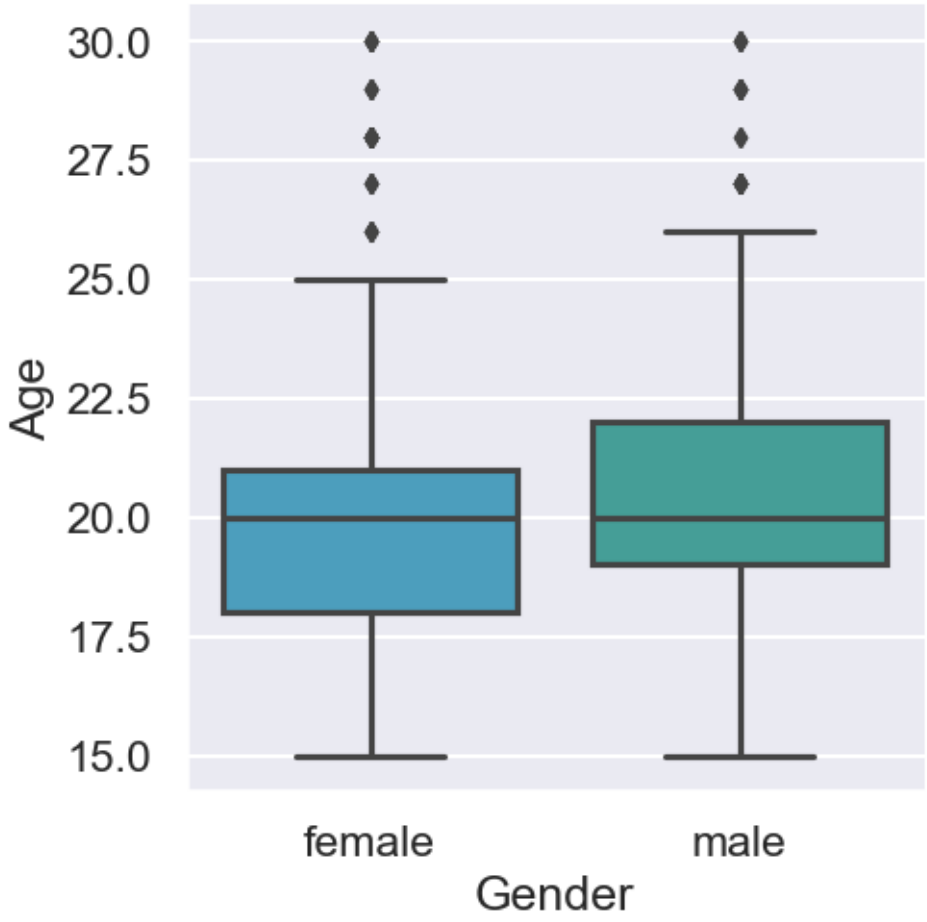
```
sns.catplot(x="Gender", y="Age",
            data=survey_data, kind="box")

# Show plot
plt.show()
```



[58]:
```
# Set the context to "paper"
sns.set_context('talk')

# Create bar plot
sns.catplot(x="Siblings", y="Loneliness",
            data=survey_data, kind="bar");
```



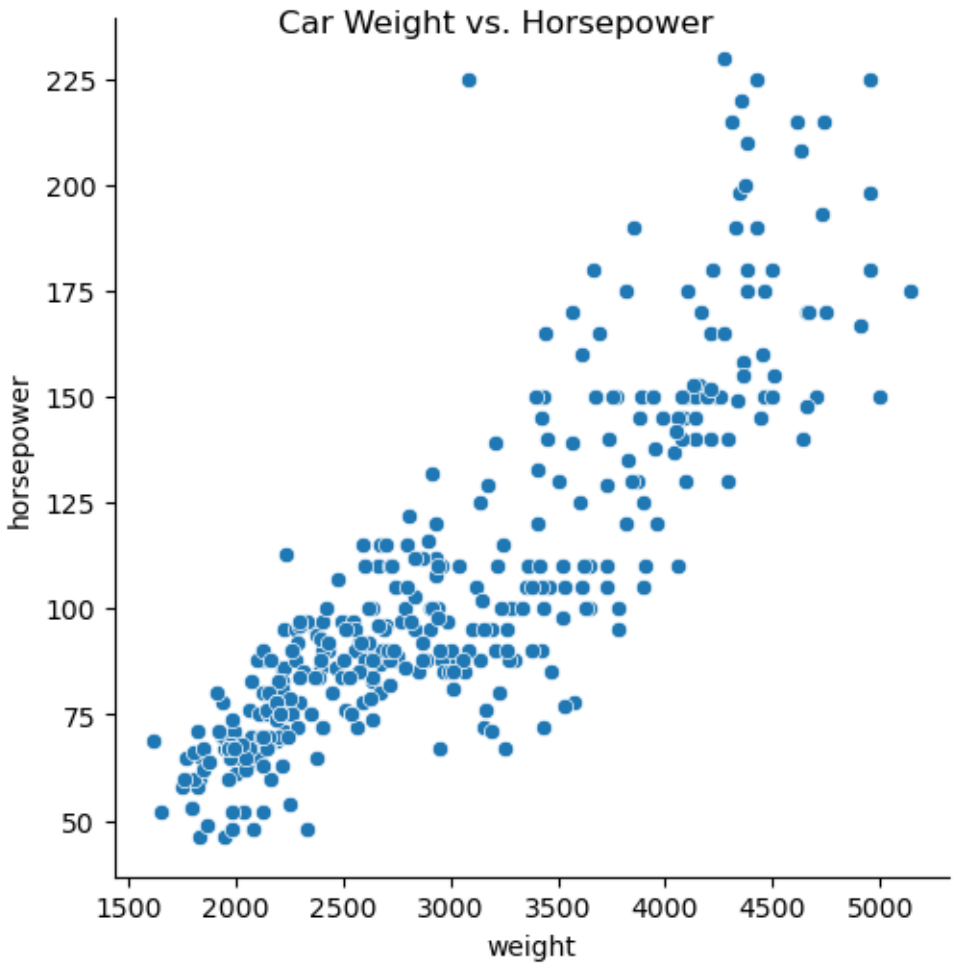
[59]:
```
# Set the style to "darkgrid"
sns.set_style('darkgrid')

# Set a custom color palette
sns.set_palette(['#39A7D0', '#36ADA4'])

# Create the box plot of age distribution by gender
sns.catplot(x="Gender", y="Age",
            data=survey_data, kind="box")

# Show plot
plt.show()
```



2.4.2 Títulos y etiquetas

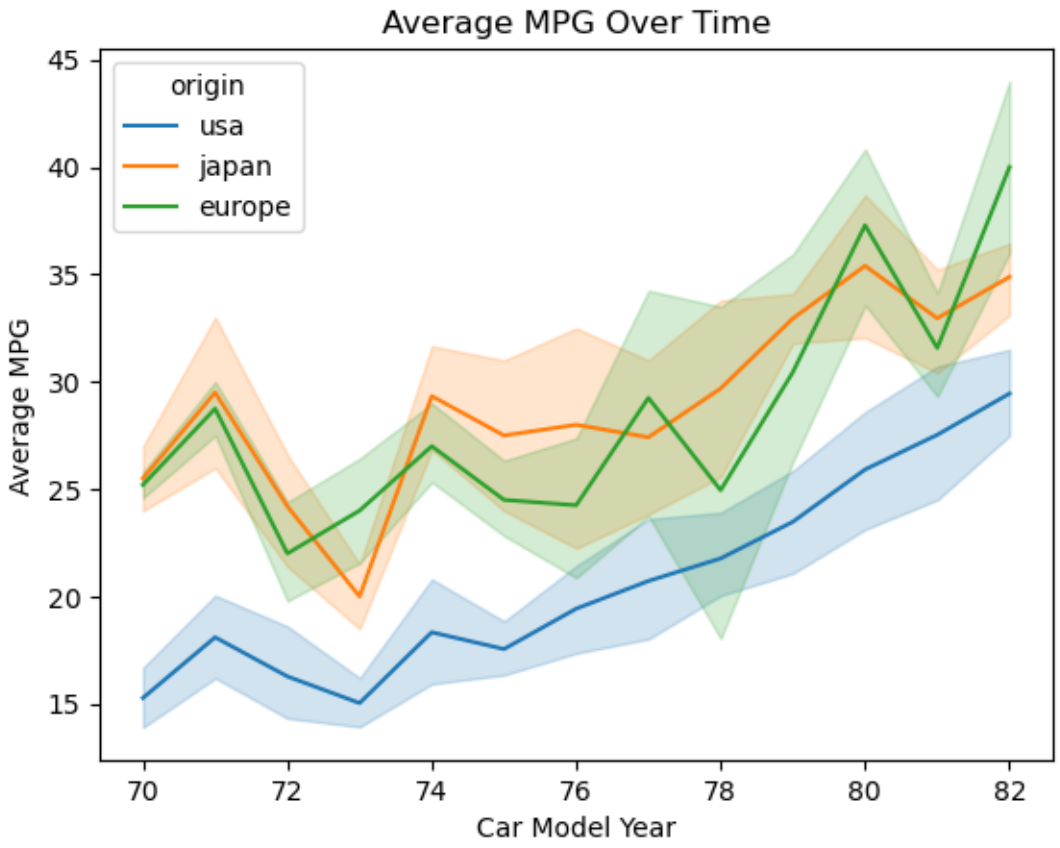
[60]:
```
import matplotlib as mpl
mpl.rcParams.update(mpl.rcParamsDefault)

# Create scatter plot
g = sns.relplot(x="weight",
                y="horsepower",
                data=mpg,
                kind="scatter");

# Add a title "Car Weight vs. Horsepower"
g.fig.suptitle('Car Weight vs. Horsepower')
```

[60]:
```
Text(0.5, 0.98, 'Car Weight vs. Horsepower')
```

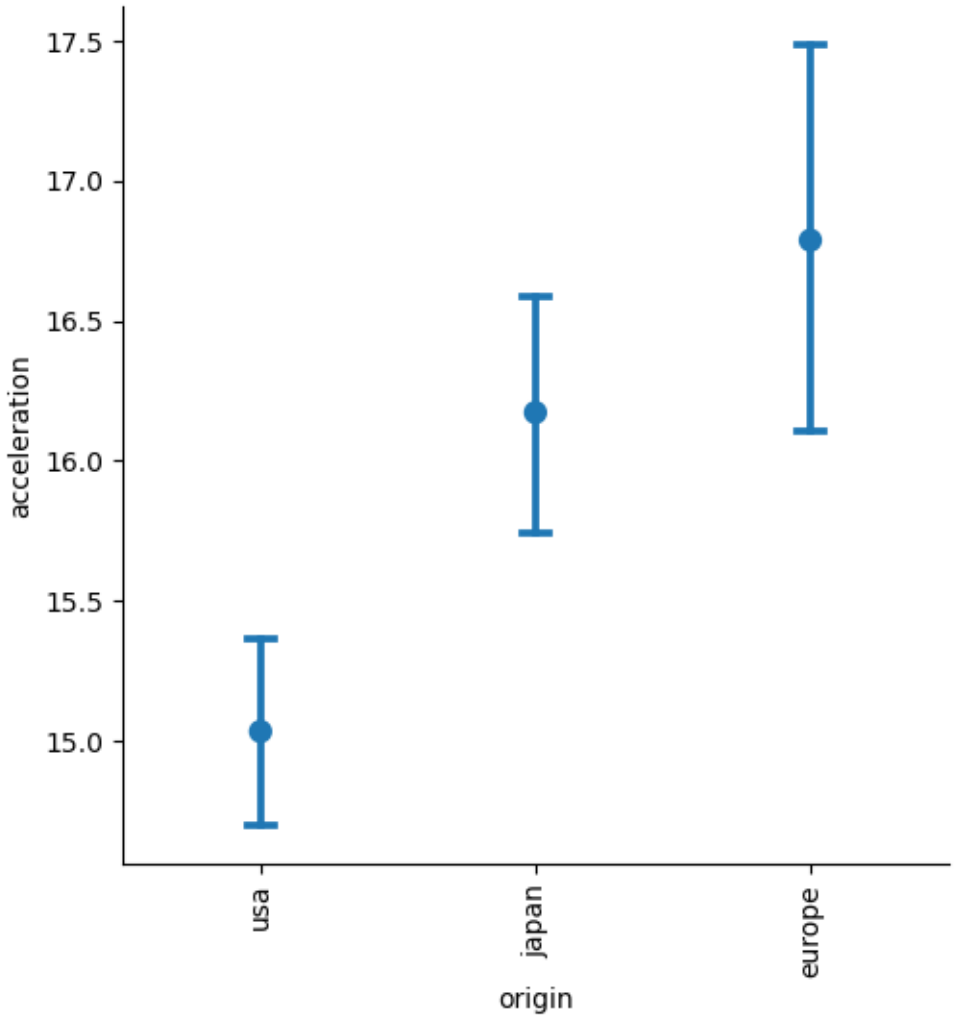


[61]:
```
# Create line plot
g = sns.lineplot(x="model_year", y="mpg",
                 data=mpg,
                 hue="origin")

# Add a title "Average MPG Over Time"
g.set_title("Average MPG Over Time")

# Add x-axis and y-axis labels
g.set(xlabel="Car Model Year",
      ylabel="Average MPG")
```

[61]: [Text(0.5, 0, 'Car Model Year'), Text(0, 0.5, 'Average MPG')]



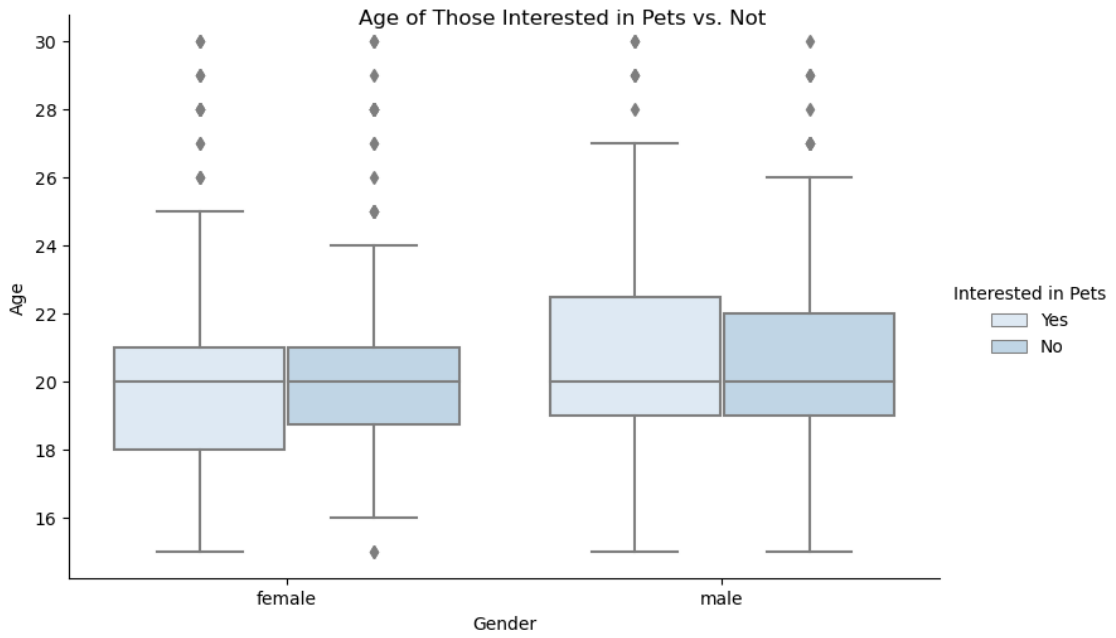
[62]:
```
# Create point plot
sns.catplot(x="origin",
            y="acceleration",
            data=mpg,
            kind="point",
            join=False,
            capsize=0.1);

# Rotate x-tick labels
plt.xticks(rotation=90);
```



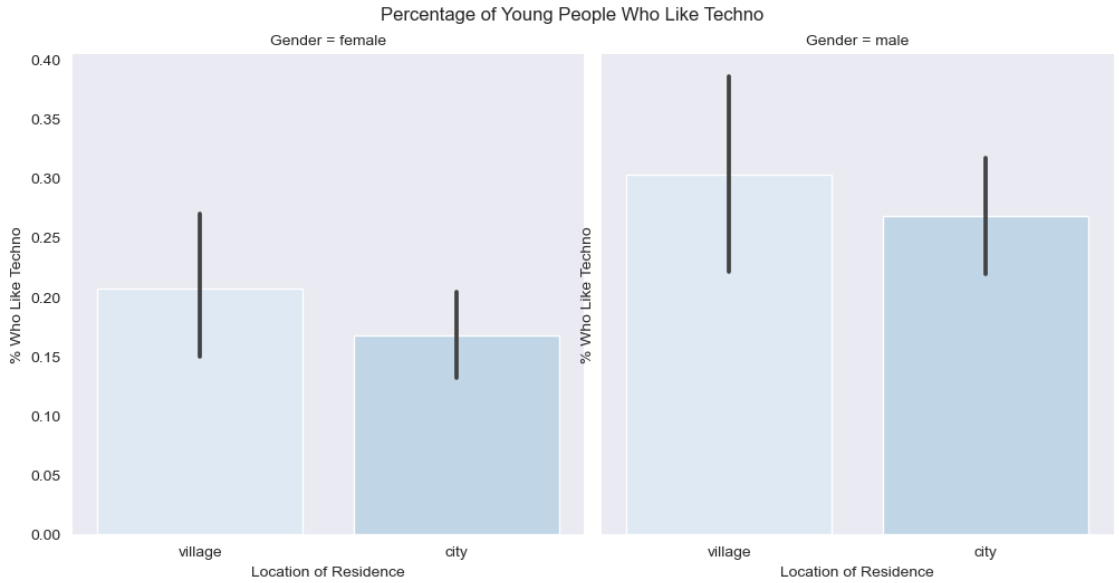
2.4.3 Resumen

[63]:
```
survey_data['Interested in Pets'] = survey_data['Pets'].apply(lambda x: 'Yes'␣
↪if x >= 4.0 else 'No')

# Set palette to "Blues"
sns.set_palette("Blues")

# Adjust to add subgroups based on "Interested in Pets"
g = sns.catplot(x="Gender",
                y="Age", data=survey_data,
                kind="box", hue='Interested in Pets', aspect = 1.5)

# Set title to "Age of Those Interested in Pets vs. Not"
g.fig.suptitle("Age of Those Interested in Pets vs. Not")
```

```
plt.show()
```



[64]:
```
survey_data['Likes Techno'] = survey_data['Techno'].apply(lambda x: True if x␣
↪>= 4.0 else False)

# Set the figure style to "dark"
sns.set_style("dark")

# Adjust to add subplots per gender
g = sns.catplot(x="Village - town", y="Likes Techno",
                data=survey_data, kind="bar",
                col='Gender')

# Add title and axis labels
g.fig.suptitle("Percentage of Young People Who Like Techno", y=1.02);
g.set(xlabel="Location of Residence",
      ylabel="% Who Like Techno")

plt.show()
```

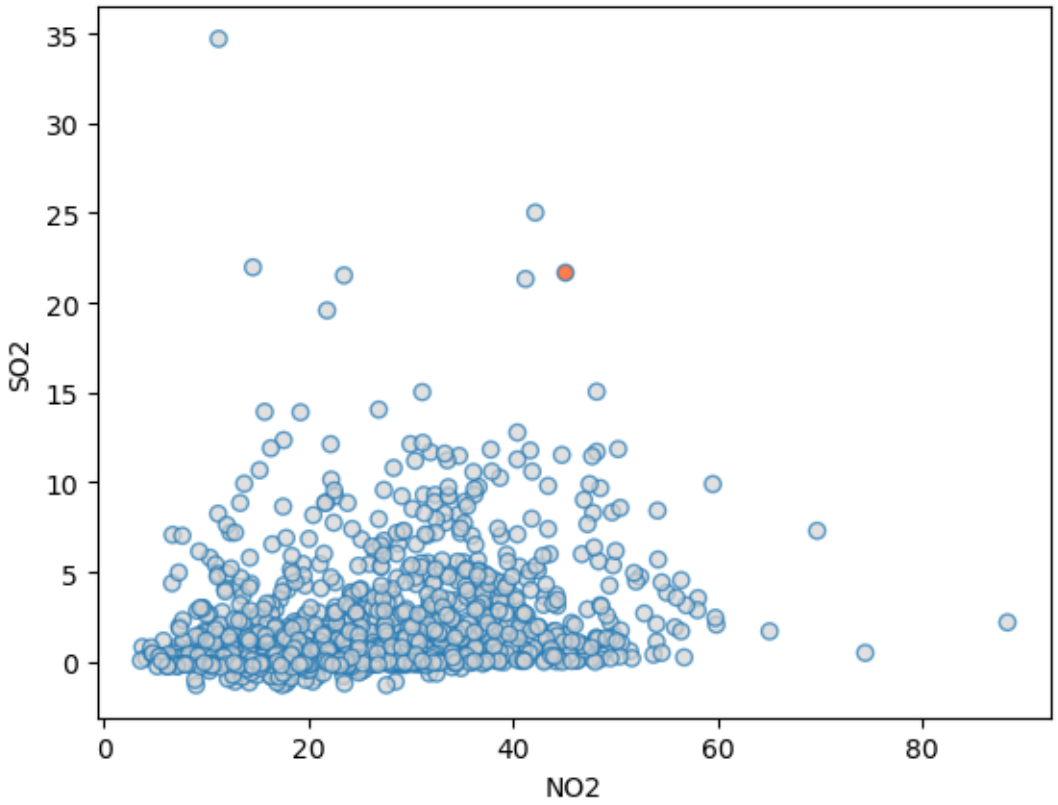


3 MEJORANDO VISUALIZACIONES EN PYTHON

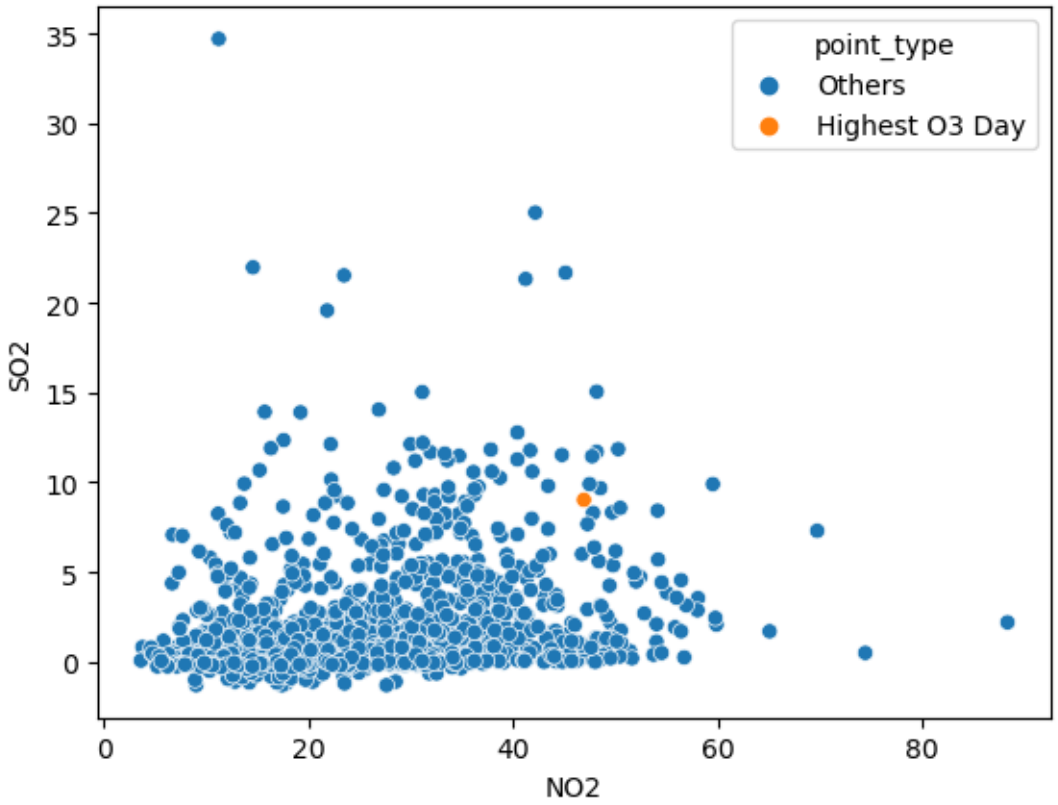
3.1 RESALTANDO DATOS

3.1.1 Highlights

[65]:
```
pollution = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
 ↪pollution_wide.csv")
mpl.rcParams.update(mpl.rcParamsDefault)

###

houston_pollution = pollution[pollution.city  ==  'Houston']

# Make array orangred for day 330 of year 2014, otherwise lightgray
houston_colors = ['orangered' if (day  ==  330) & (year  ==  2014) else␣
 ↪'lightgray'
                  for day,year in zip(houston_pollution.day, houston_pollution.
 ↪year)]

sns.regplot(x = 'NO2',
            y = 'SO2',
            data = houston_pollution,
            fit_reg = False,
            # Send scatterplot argument to color points
            scatter_kws = {'facecolors': houston_colors, 'alpha': 0.7})
plt.show()
```

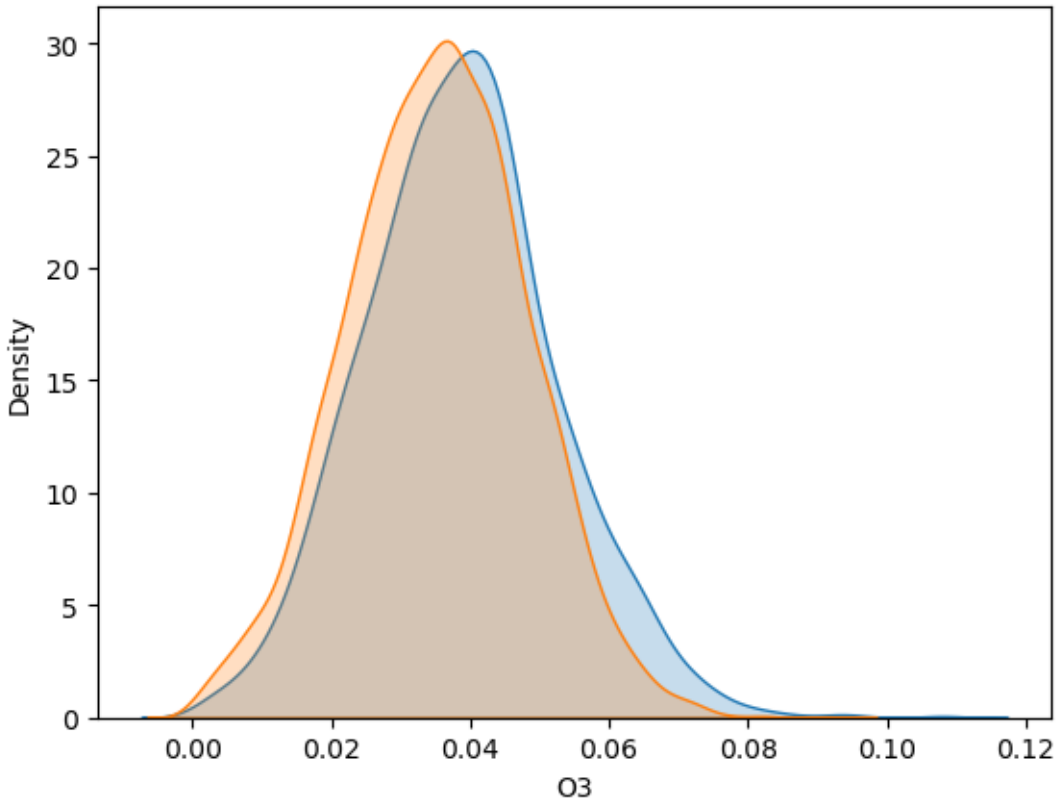


[66]:
```
houston_pollution = pollution[pollution.city  ==  'Houston'].copy()

# Find the highest observed O3 value
max_O3 = houston_pollution.O3.max()

# Make a column that denotes which day had highest O3
houston_pollution['point_type'] = ['Highest O3 Day' if O3  ==  max_O3 else␣
 ↪'Others' for O3 in houston_pollution.O3]

# Encode the hue of the points with the O3 generated column
sns.scatterplot(x = 'NO2',
                y = 'SO2',
                hue = 'point_type',
                data = houston_pollution)
plt.show()
```



3.1.2 KDEs

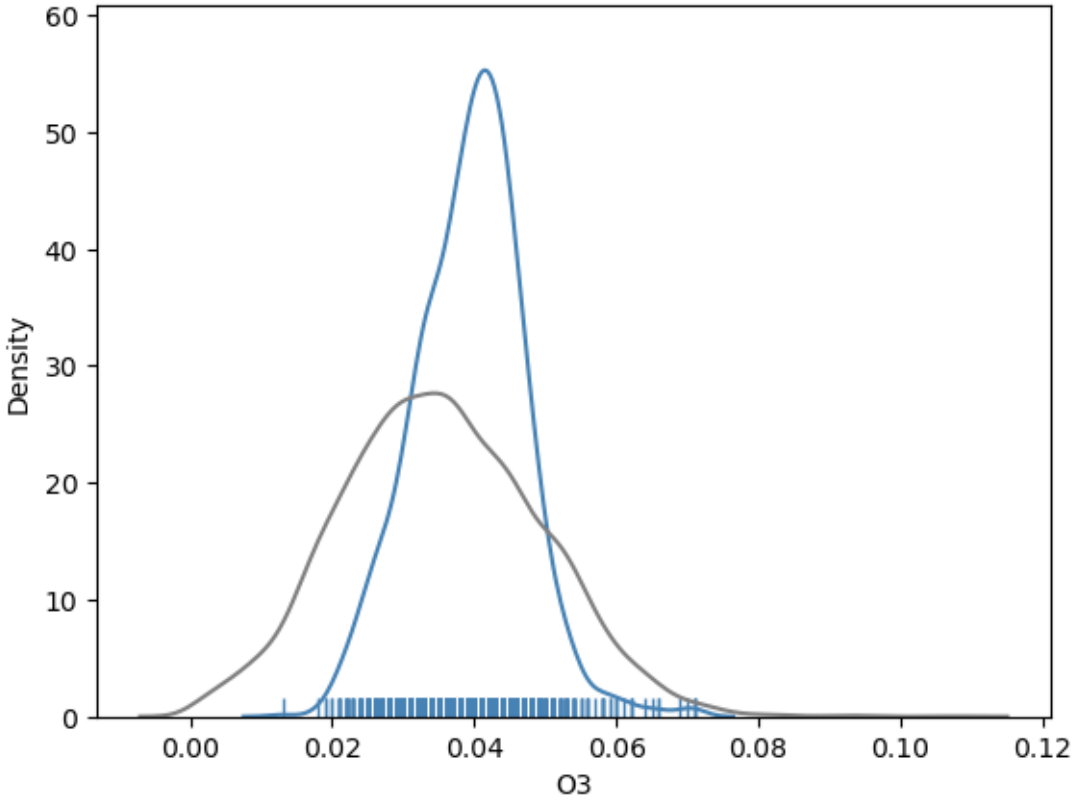
[67]:
```
# Filter dataset to the year 2012
sns.kdeplot(pollution[pollution.year == 2012].O3,
            # Shade under kde and add a helpful label
            shade = True,
            label = '2012')

# Filter dataset to everything except the year 2012
sns.kdeplot(pollution[pollution.year != 2012].O3,
            # Again, shade under kde and add a helpful label
            shade = True,
            label= 'other years')
plt.show()
```



[68]:
```
sns.distplot(pollution[pollution.city == 'Vandenberg Air Force Base'].O3,
             label = 'Vandenberg',
                # Turn of the histogram and color blue to stand out
               hist = False,
             color = 'steelblue',
               # Turn on rugplot
               rug = True)

sns.distplot(pollution[pollution.city != 'Vandenberg Air Force Base'].O3,
             label = 'Other cities',
               # Turn off histogram and color gray
               hist = False,
             color = 'gray')
plt.show()
```

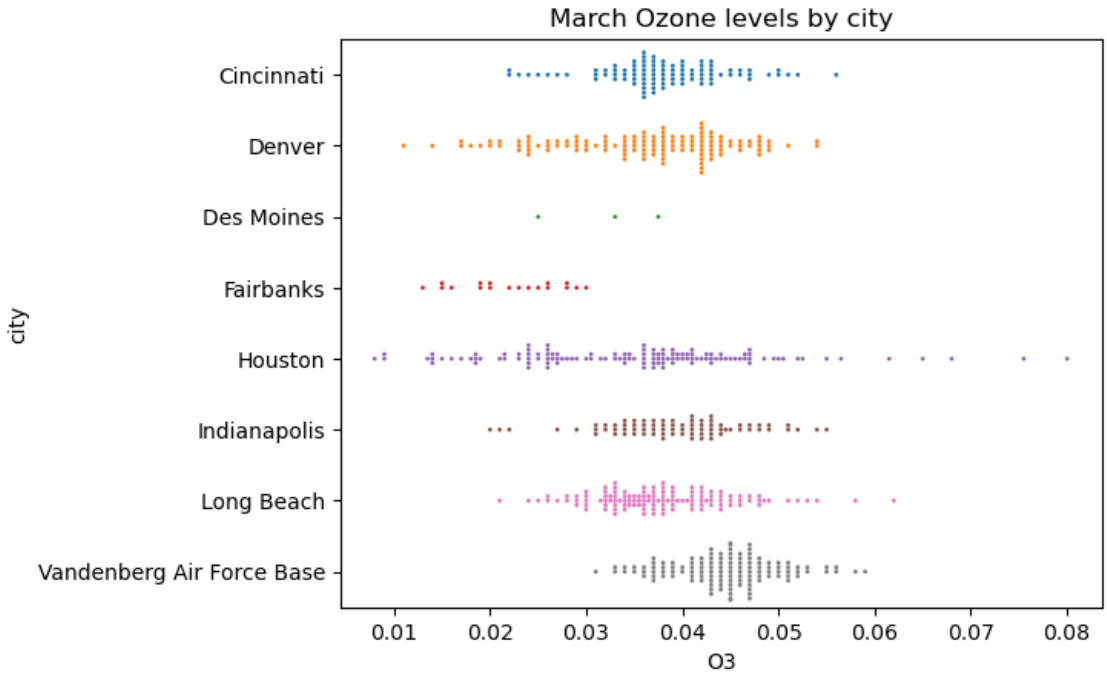
```
C:\Users\marco\anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `kdeplot` (an axes-level function for
kernel density plots).
  warnings.warn(msg, FutureWarning)
```

```
C:\Users\marco\anaconda3\lib\site-packages\seaborn\distributions.py:2056:
FutureWarning: The `axis` variable is no longer used and will be removed.
Instead, assign variables directly to `x` or `y`.
  warnings.warn(msg, FutureWarning)
C:\Users\marco\anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `kdeplot` (an axes-level function for
kernel density plots).
  warnings.warn(msg, FutureWarning)
```



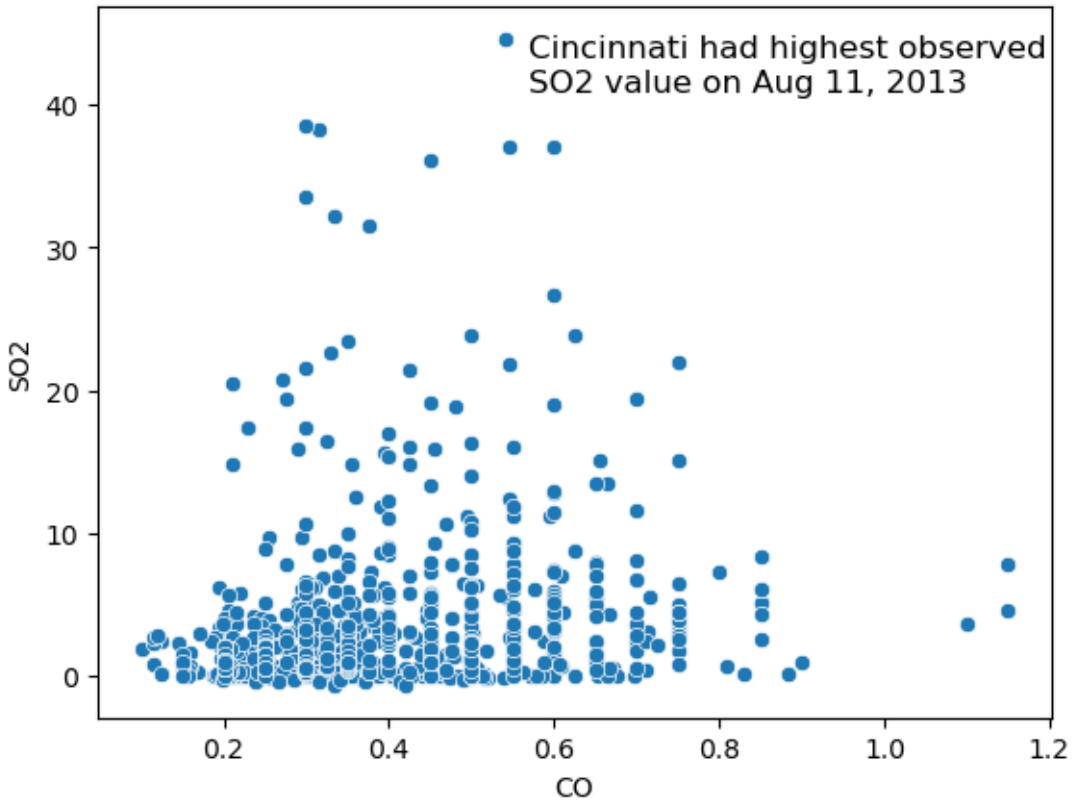
[69]:
```
# Filter data to just March
pollution_mar = pollution[pollution.month == 3]

# Plot beeswarm with x as O3
sns.swarmplot(y = "city",
              x = "O3",
              data = pollution_mar,
              # Decrease the size of the points to avoid crowding
              size = 2)
```

```
# Give a descriptive title
plt.title('March Ozone levels by city')
plt.show()
```

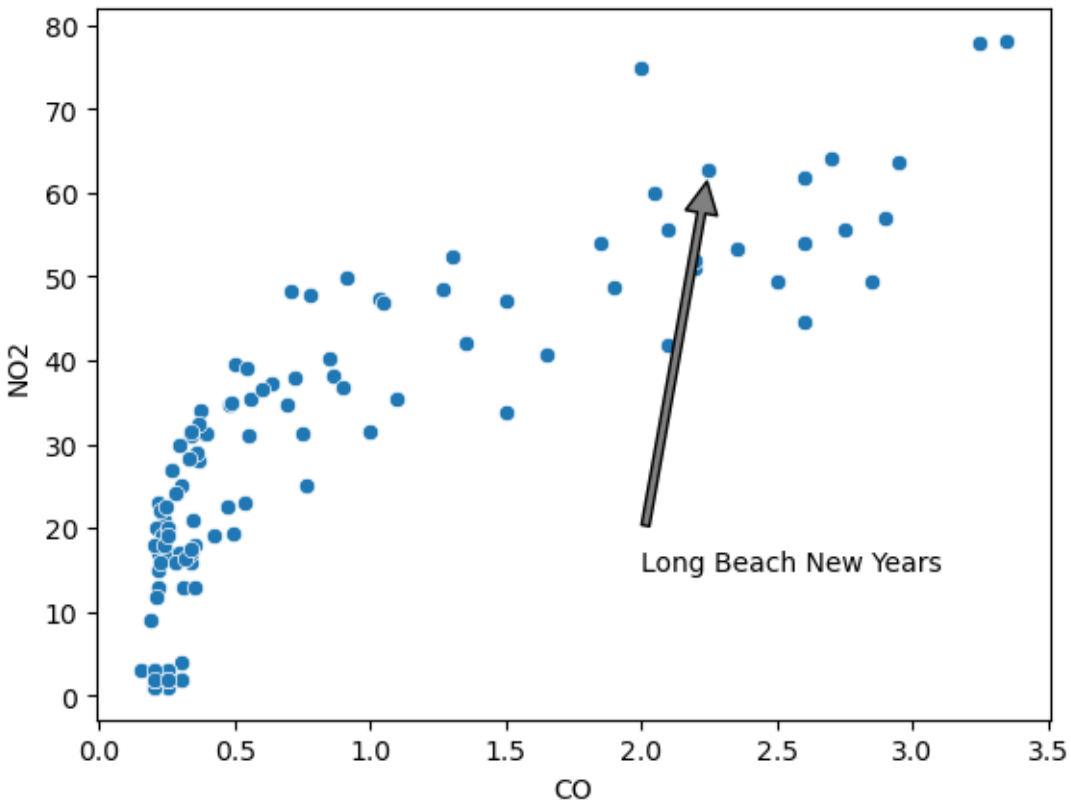


3.1.3 Anotaciones

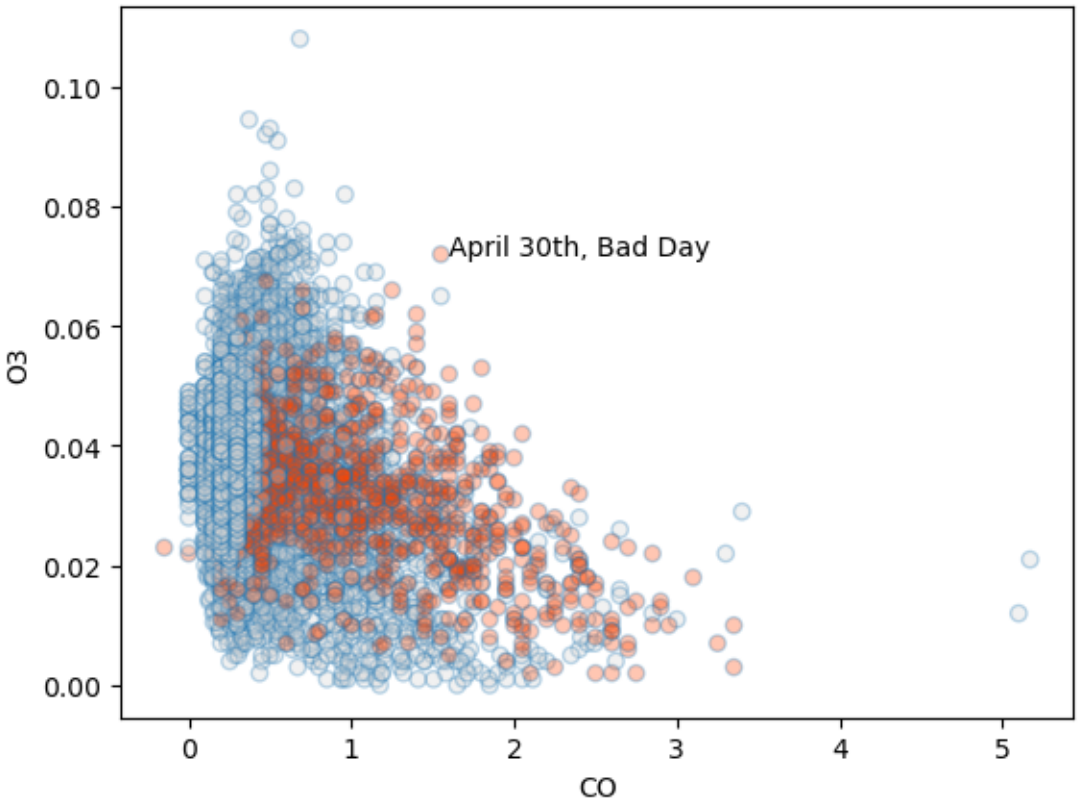
[70]:
```
# Draw basic scatter plot of pollution data for August
sns.scatterplot(x = 'CO', y = 'SO2', data = pollution[pollution.month  ==  8])

# Label highest SO2 value with text annotation
plt.text(0.57, 41,
         'Cincinnati had highest observed\nSO2 value on Aug 11, 2013',
         # Set the font to large
         fontdict = {'ha': 'left', 'size': 'large'})
plt.show()
```



[71]:
```
# Query and filter to New Years in Long Beach
jan_pollution = pollution.query("(month  ==  1) & (year  ==  2012)")
lb_newyears = jan_pollution.query("(day  ==  1) & (city  ==  'Long Beach')")

sns.scatterplot(x = 'CO', y = 'NO2',
                data = jan_pollution)

# Point arrow to lb_newyears & place text in lower left
plt.annotate('Long Beach New Years',
             xy = (lb_newyears.CO, lb_newyears.NO2),
             xytext = (2, 15),
             # Shrink the arrow to avoid occlusion
             arrowprops = {'facecolor':'gray', 'width': 3, 'shrink': 0.03},
             backgroundcolor = 'white')
plt.show()
```

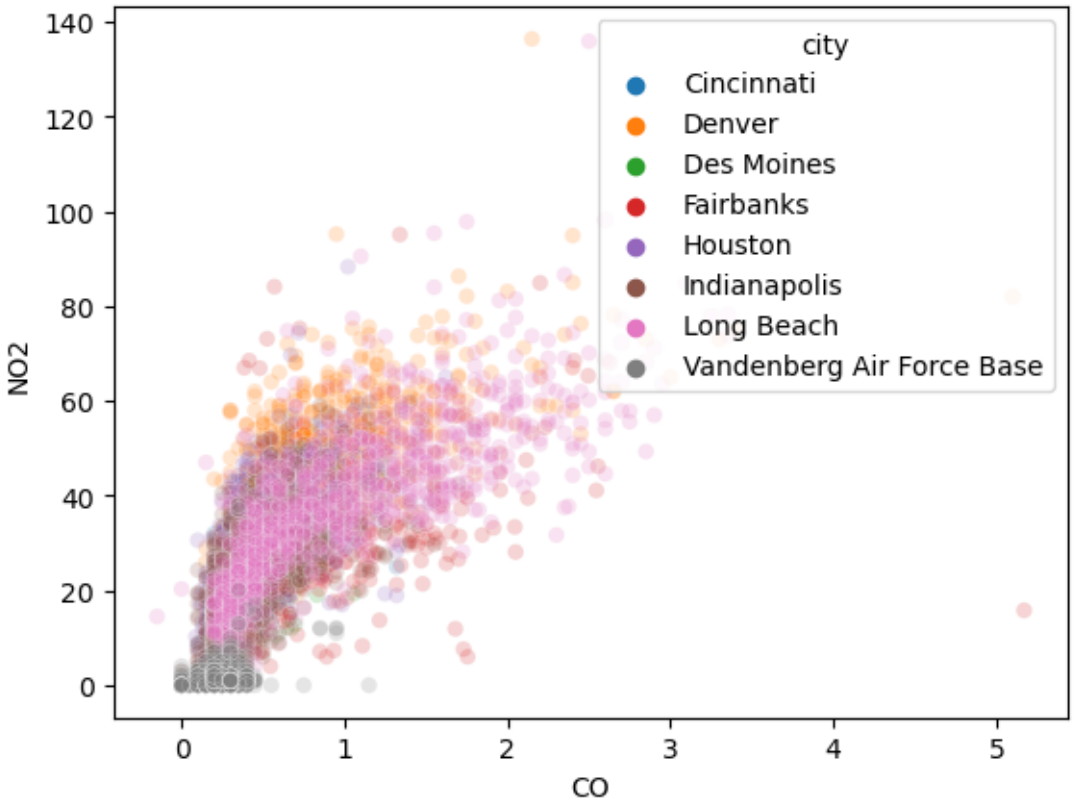


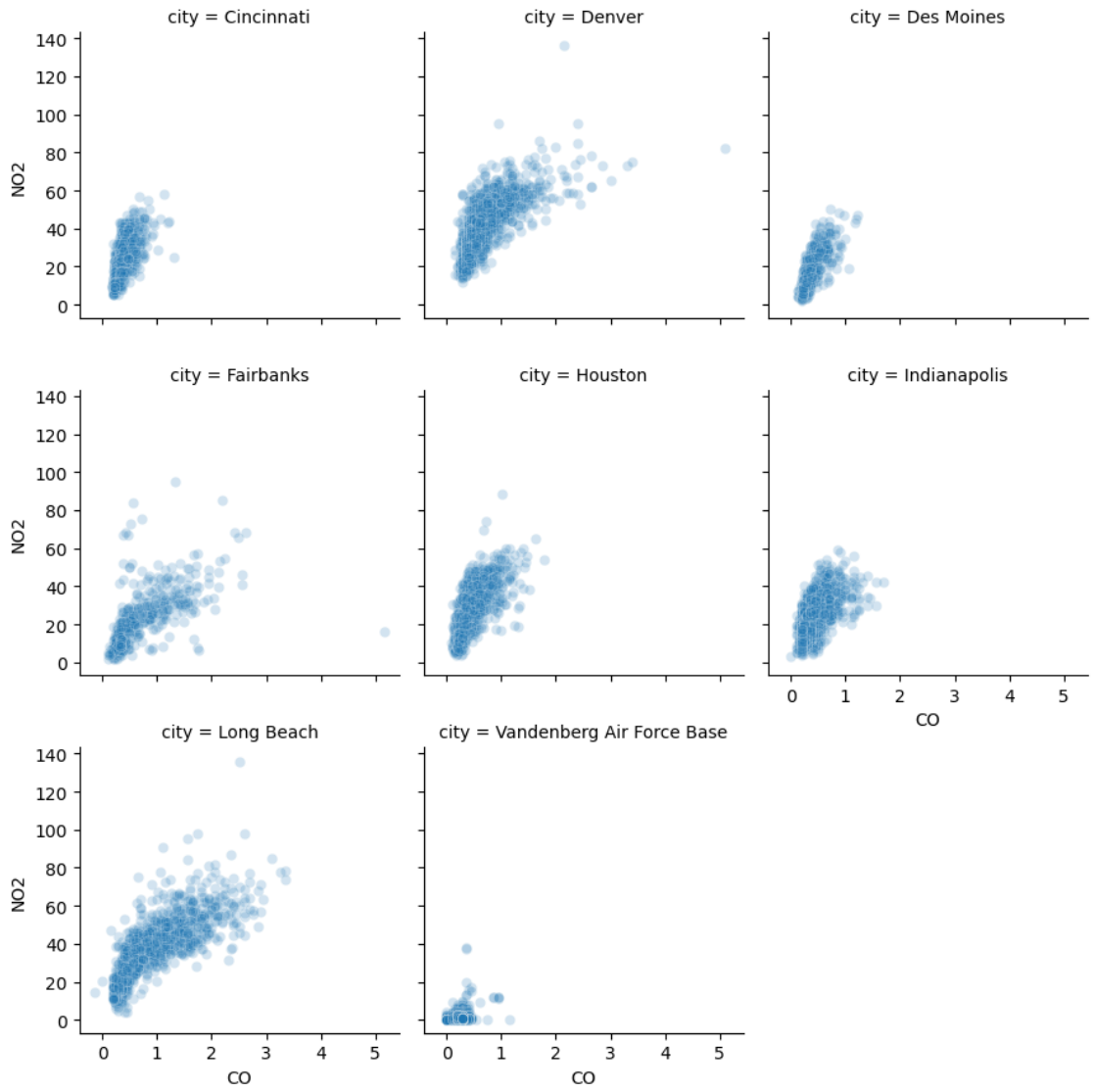
[72]:
```
# Make a vector where Long Beach is orangered; else lightgray
is_lb = ['orangered' if city  ==  'Long Beach' else 'lightgray' for city in␣
 ↪pollution['city']]

# Map facecolors to the list is_lb and set alpha to 0.3
sns.regplot(x = 'CO',
            y = 'O3',
            data = pollution,
            fit_reg = False,
            scatter_kws = {'facecolors':is_lb, 'alpha': 0.3})

# Add annotation to plot
plt.text(1.6, 0.072, "April 30th, Bad Day")
plt.show()
```



3.2 USANDO COLORES

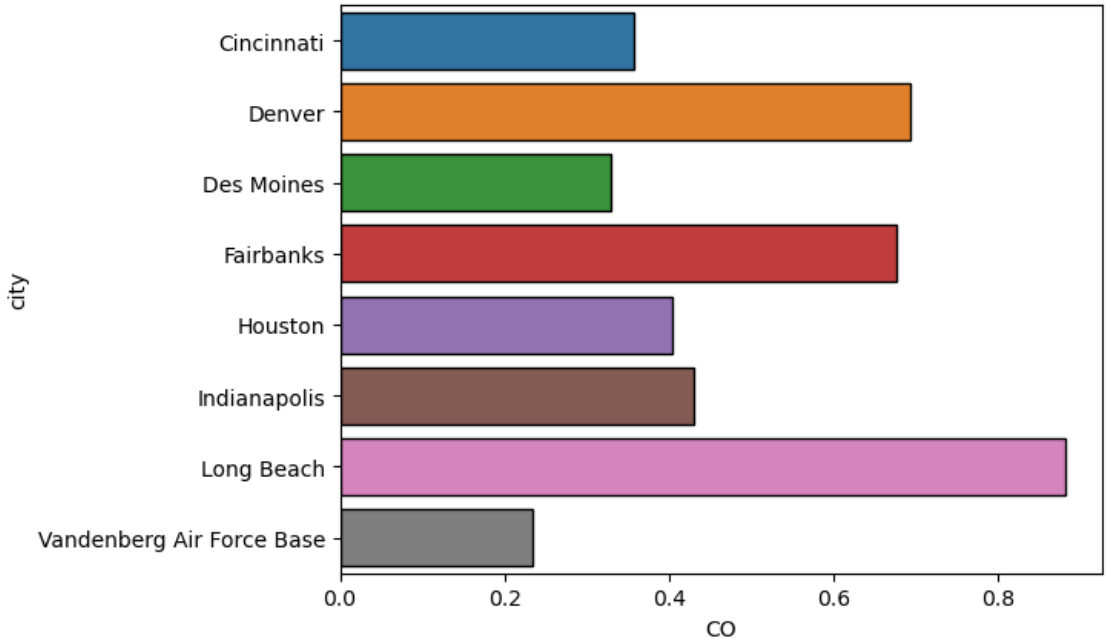
[73]:
```
 sns.scatterplot(x = 'CO', y ='NO2',
                 alpha = 0.2,
                 hue = 'city',
                 data = pollution)
plt.show()

# Pero es mejor separar las ciudades:

# Setup a facet grid to separate the cities apart
g = sns.FacetGrid(data = pollution,
                  col = 'city',
                  col_wrap = 3)

# Map sns.scatterplot to create separate city scatter plots
g.map(sns.scatterplot, 'CO', 'NO2', alpha = 0.2)
plt.show()
```

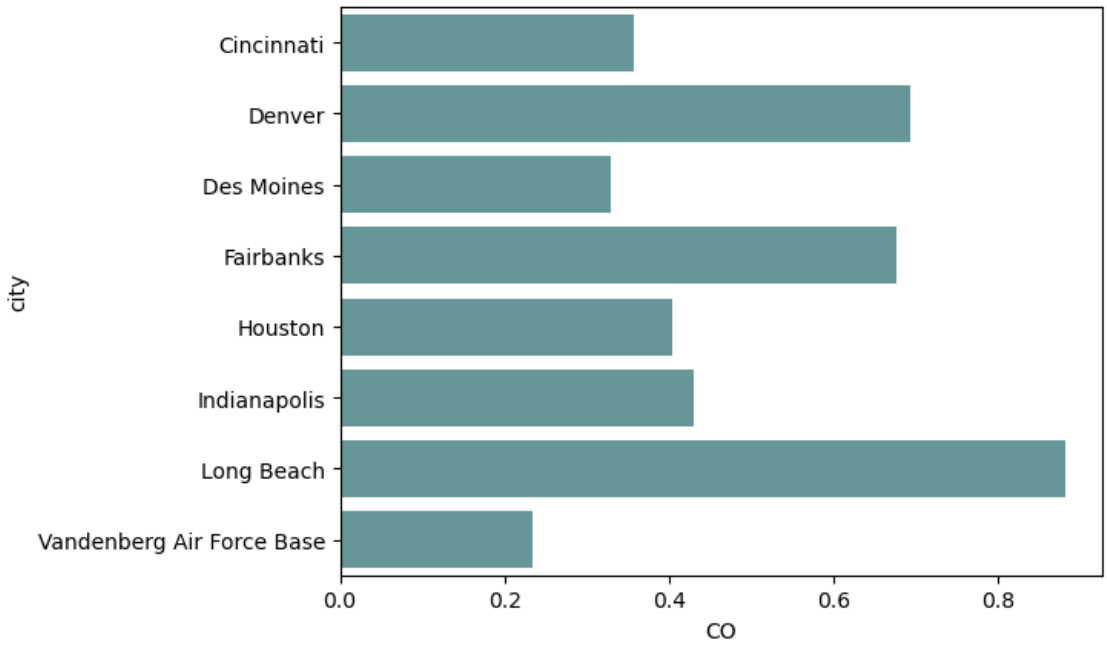

city
Cincinnati
Denver
Des Moines
Fairbanks
Houston
Indianapolis
Long Beach
Vandenberg Air Force Base
NO2
CO
0
20
40
60
80
100
120
140
0
1
2
3
4
5



[74]:
```
# Cuando hay demasiados colores:

import numpy as np

sns.barplot(y = 'city', x = 'CO',
            estimator = np.mean,
            ci = False,
            data = pollution,
            # Add a border to the bars
            edgecolor = "black")
plt.show()

###
```

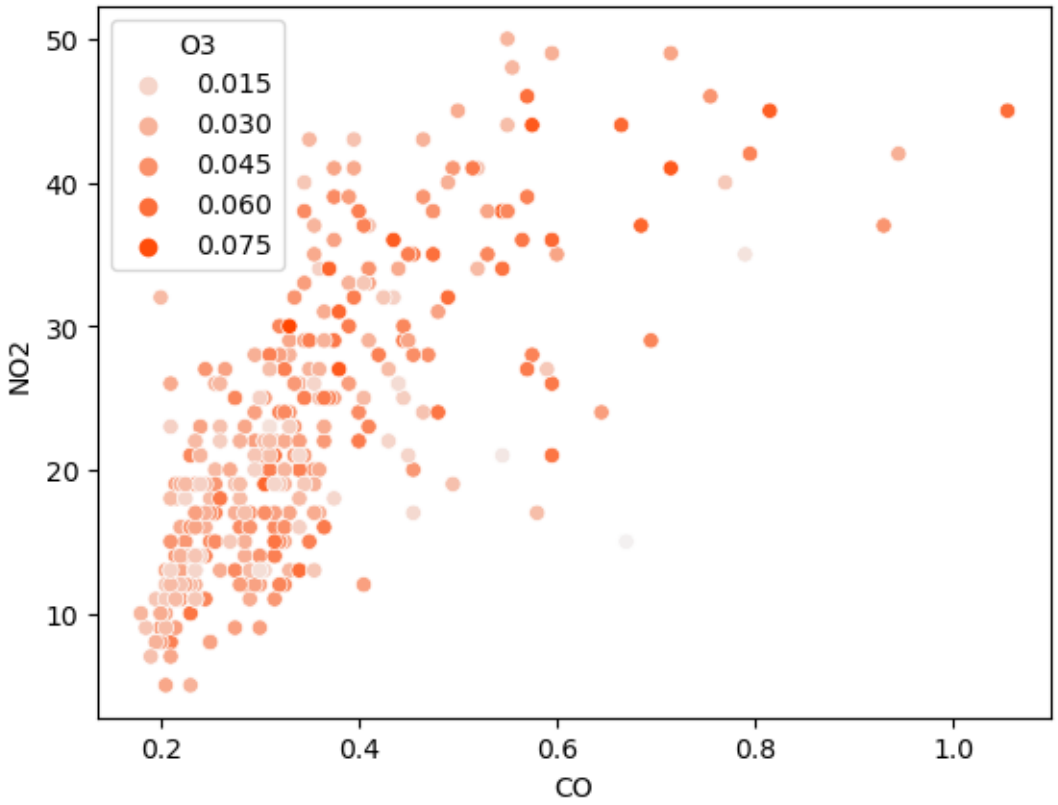
```
sns.barplot(y = 'city', x = 'CO',
            estimator = np.mean,
            ci = False,
            data = pollution,
            # Replace border with bar colors
            color = "cadetblue")
plt.show()
```





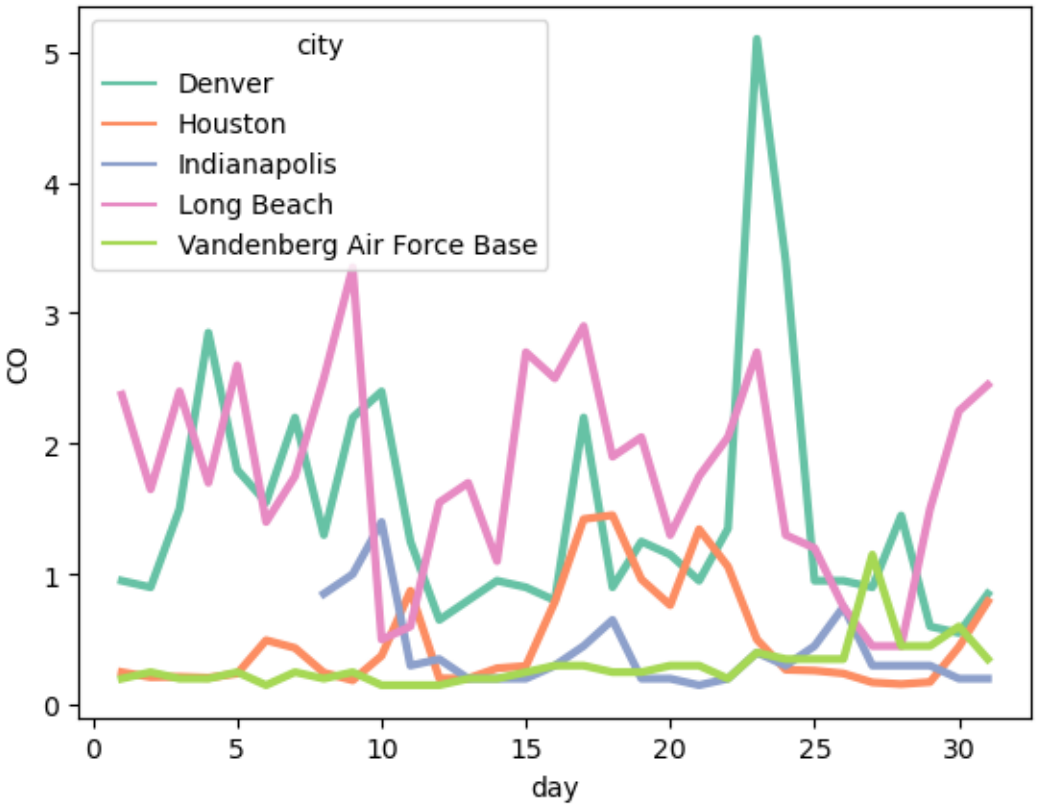
3.2.1 Paletas continuas

[75]:
```
# Paleta continua personalizada:

# Filter the data
cinci_2014 = pollution.query("city  ==  'Cincinnati' & year  ==  2014")

# Define a custom continuous color palette
color_palette = sns.light_palette('orangered',
                        as_cmap = True)

# Plot mapping the color of the points with custom palette
sns.scatterplot(x = 'CO',
                y = 'NO2',
                hue = 'O3',
                data = cinci_2014,
                palette = color_palette)
plt.show()
```

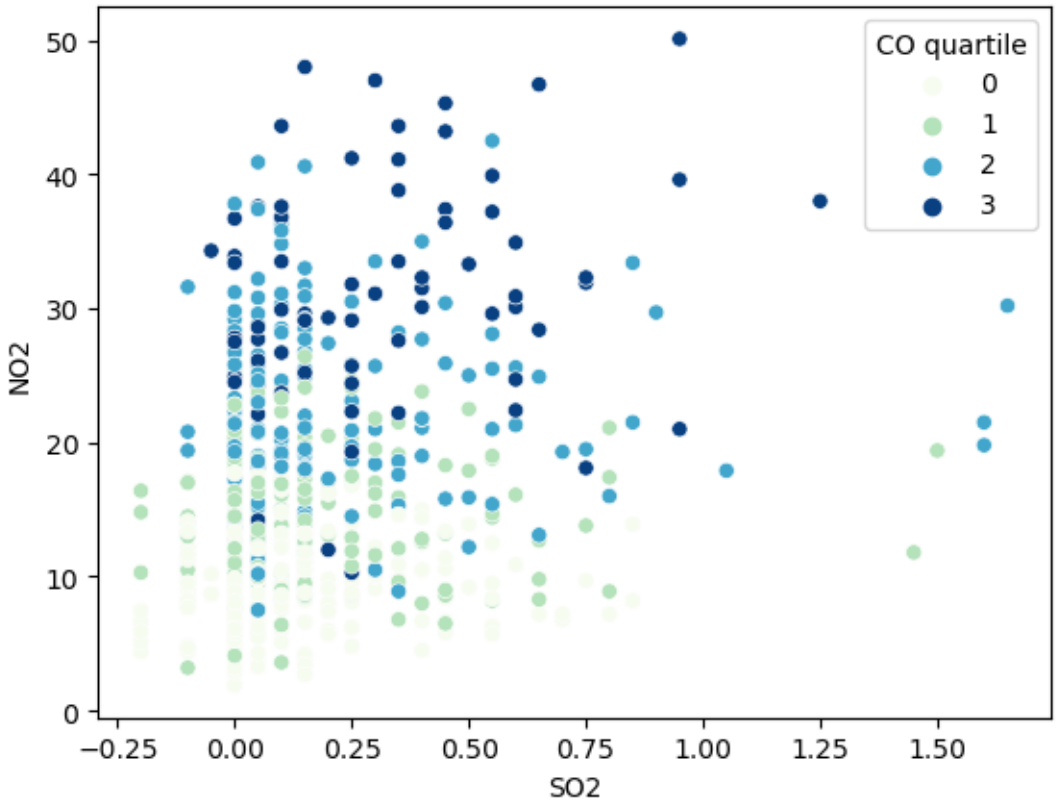


3.2.2 Paletas categóricas

[76]:
```
# Paleta predeterminada:
mpl.rcParams.update(mpl.rcParamsDefault)

# Filter data to Jan 2013
pollution_jan13 = pollution.query('year  ==  2013 & month  ==  1')

# Color lines by the city and use custom ColorBrewer palette
sns.lineplot(x = "day",
             y = "CO",
             hue = "city",
             palette = "Set2",
             linewidth = 3,
             data = pollution_jan13)
plt.show()
```



[77]:
```
# Ccategorías ordinales:


# Divide CO into quartiles
pollution['CO quartile'] = pd.qcut(pollution['CO'], q = 4, labels = False)

# Filter to just Des Moines
des_moines = pollution.query("city  ==  'Des Moines'")

# Color points with by quartile and use ColorBrewer palette
sns.scatterplot(x = 'SO2',
                y = 'NO2',
                hue = 'CO quartile',
                  data = des_moines,
                palette = 'GnBu')
plt.show()
```

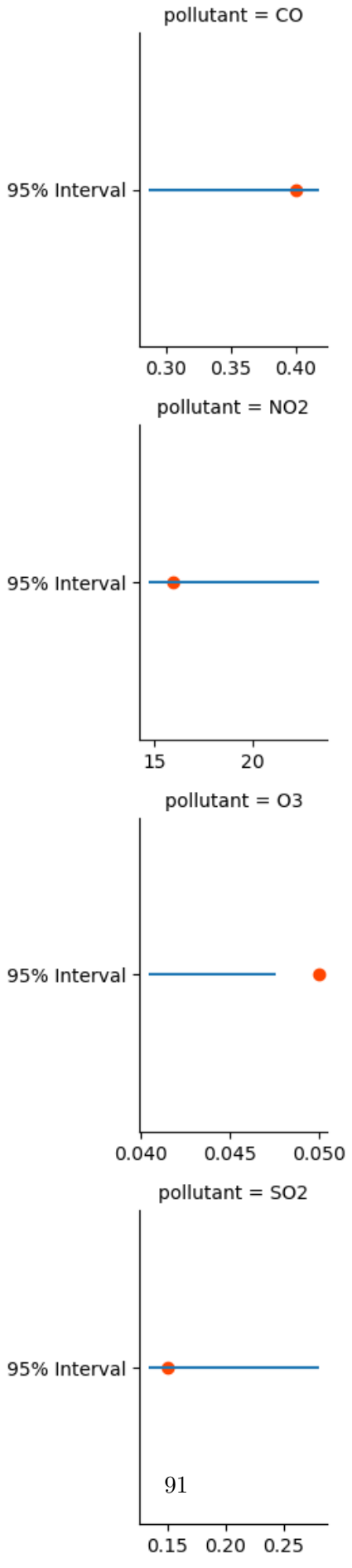


3.3 INCERTIDUMBRE

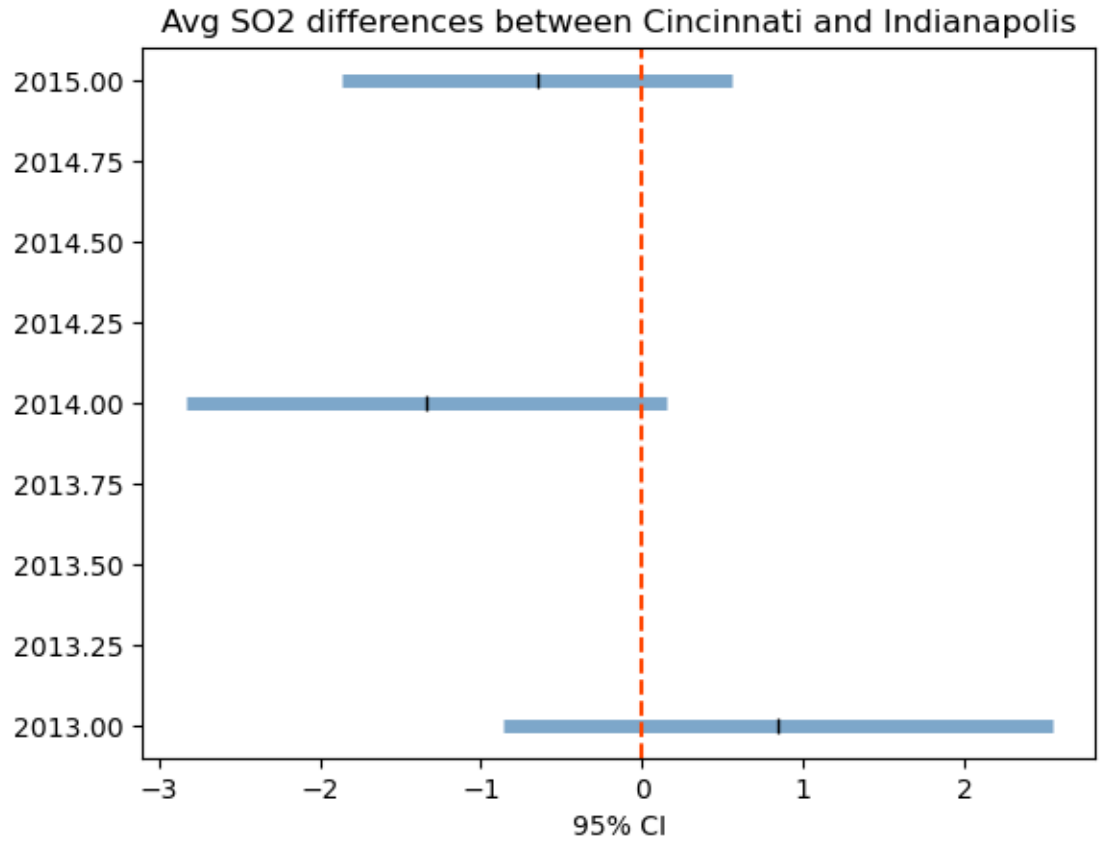
3.3.1 Intervalos de confianza

[78]:
```
average_ests = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
 ↪average_ests.csv", index_col = 0)

# Construct CI bounds for averages
average_ests['lower'] = average_ests['mean'] - 1.96*average_ests['std_err']
average_ests['upper'] = average_ests['mean'] + 1.96*average_ests['std_err']

# Setup a grid of plots, with non-shared x axes limits
g = sns.FacetGrid(average_ests, row = 'pollutant', sharex = False)

# Plot CI for average estimate
g.map(plt.hlines, 'y', 'lower', 'upper')

# Plot observed values for comparison and remove axes labels
g.map(plt.scatter, 'seen', 'y', color = 'orangered').set_ylabels('').
 ↪set_xlabels('')
```

```
plt.show()
```


pollutant = CO
95% Interval
0.30
0.35
0.40
pollutant = NO2
95% Interval
15
20
pollutant = O3
95% Interval
0.040
0.045
0.050
pollutant = SO2
95% Interval
0.15
0.20
0.25

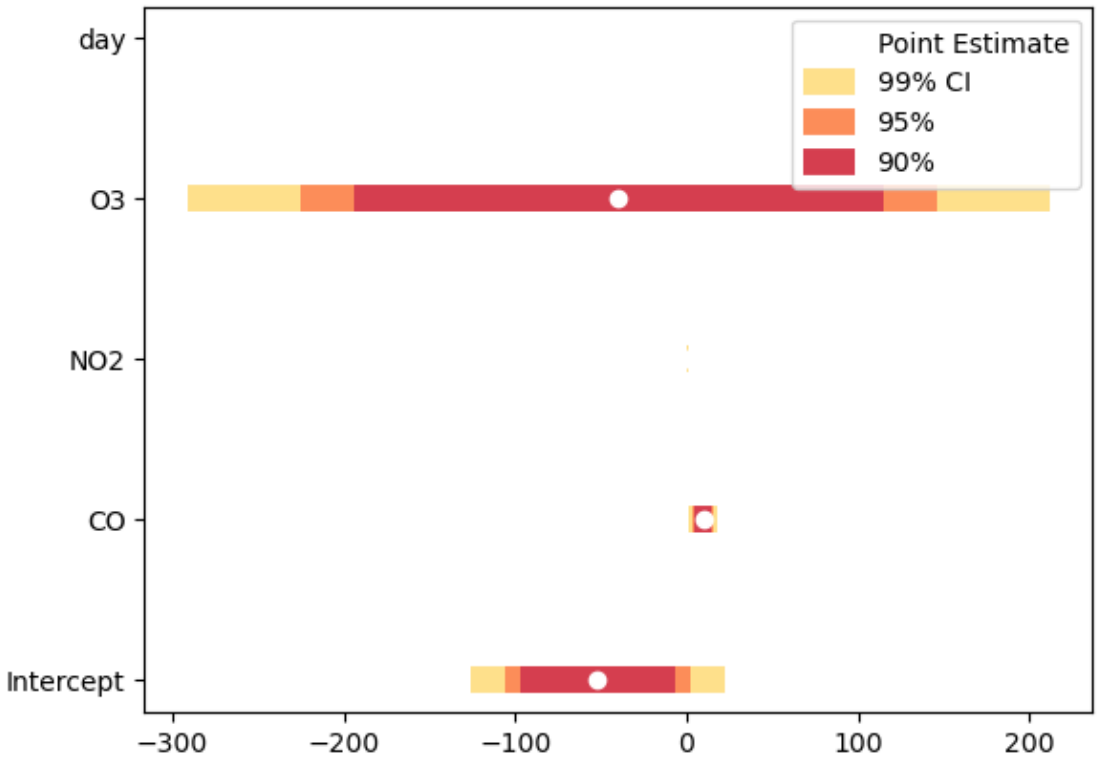
[79]:
```
# Set start and ends according to intervals

diffs_by_year = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
 ↪diffs_by_year.csv", index_col = 0)
# Make intervals thicker
plt.hlines(y = 'year', xmin = 'lower', xmax = 'upper',
           linewidth = 5, color = 'steelblue', alpha = 0.7,
           data = diffs_by_year)
# Point estimates
plt.plot('mean', 'year', 'k|', data = diffs_by_year)

# Add a 'null' reference line at 0 and color orangered
plt.axvline(x = 0, color = 'orangered', linestyle = '--')

# Set descriptive axis labels and title
plt.xlabel('95% CI')
plt.title('Avg SO2 differences between Cincinnati and Indianapolis')
plt.show()
```



3.3.2 Point estimates

[80]:
```
from statsmodels.formula.api import ols
pollution = pollution.query("city == 'Fairbanks' & year == 2014 & month == 11")
pollution_model = ols(formula='SO2 ~ CO + NO2 + O3 + day', data=pollution)

res = pollution_model.fit()
# Add interval percent widths
alphas = [     0.01,  0.05,   0.1]
widths = [ '99% CI', '95%', '90%']
colors = ['#fee08b','#fc8d59','#d53e4f']

for alpha, color, width in zip(alphas, colors, widths):
    # Grab confidence interval
    conf_ints = res.conf_int(alpha)

    # Pass current interval color and legend label to plot
    plt.hlines(y = conf_ints.index, xmin = conf_ints[0], xmax = conf_ints[1],
               colors = color, label = width, linewidth = 10)

# Draw point estimates
plt.plot(res.params, res.params.index, 'wo', label = 'Point Estimate')

plt.legend(loc = 'upper right')
```

[80]:
```
<matplotlib.legend.Legend at 0x1bd2f209430>
```

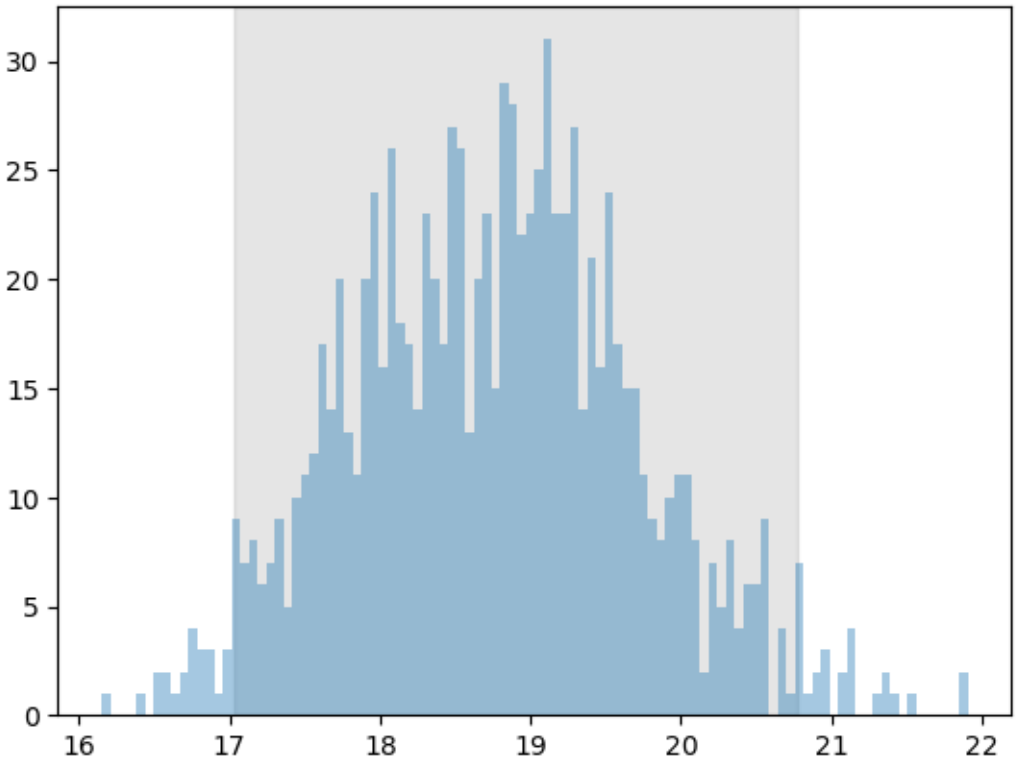


3.3.3 Bootstrap

[81]:
```
# Perform bootstrapped mean on a vector
def bootstrap(data, n_boots):
    return [np.mean(np.random.choice(data,len(data))) for _ in range(n_boots)]

pollution = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
 ↪pollution_wide.csv")

cinci_may_NO2 = pollution.query("city  ==  'Cincinnati' & month  ==  5").NO2

# Generate bootstrap samples
boot_means = bootstrap(cinci_may_NO2, 1000)

# Get lower and upper 95% interval bounds
lower, upper = np.percentile(boot_means, [2.5, 97.5])

# Plot shaded area for interval
plt.axvspan(lower, upper, color = 'gray', alpha = 0.2)

# Draw histogram of bootstrap samples
sns.distplot(boot_means, bins = 100, kde = False)
```

```
plt.show()
```

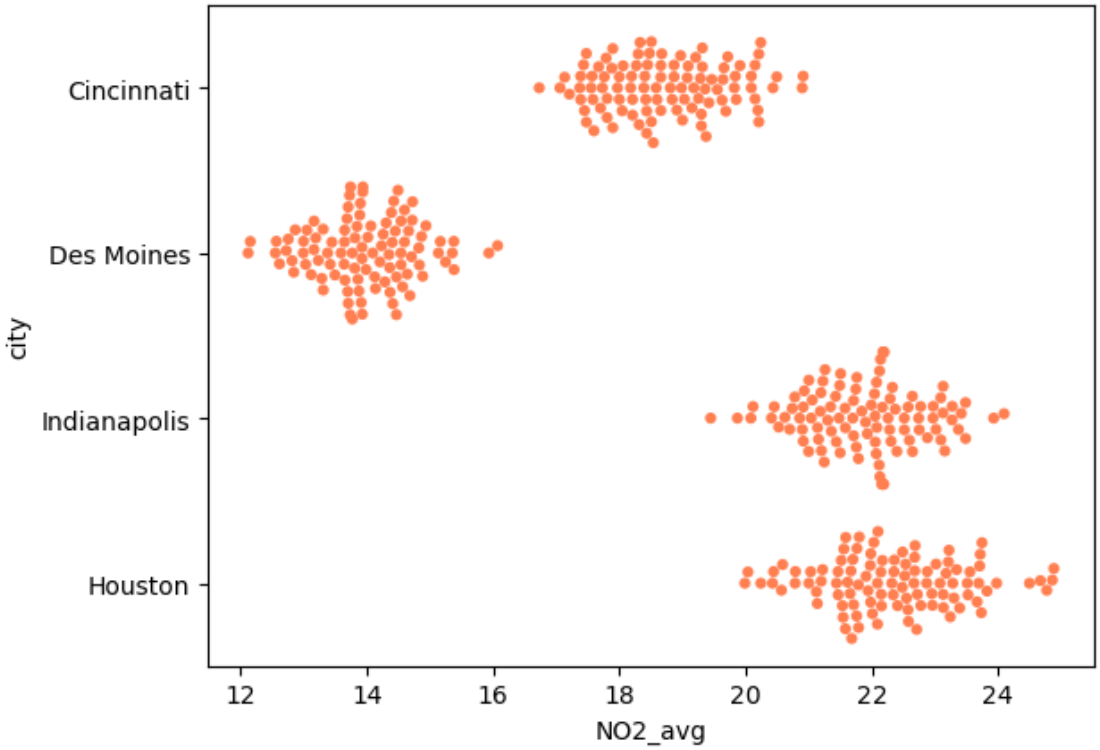
```
C:\Users\marco\anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



[82]:
```
pollution_may = pollution.query("month == 5")

# Initialize a holder DataFrame for bootstrap results
city_boots = pd.DataFrame()

for city in ['Cincinnati', 'Des Moines', 'Indianapolis', 'Houston']:
    # Filter to city
    city_NO2 = pollution_may[pollution_may.city  ==  city].NO2
    # Bootstrap city data & put in DataFrame
    cur_boot = pd.DataFrame({'NO2_avg': bootstrap(city_NO2, 100), 'city': city})
    # Append to other city's bootstraps
    city_boots = pd.concat([city_boots,cur_boot])
```

```
# Beeswarm plot of averages with citys on y axis
sns.swarmplot(y = "city", x = "NO2_avg", size = 4.5, data = city_boots, color =␣
↪'coral');
```

```
C:\Users\marco\anaconda3\lib\site-packages\seaborn\categorical.py:1296:
UserWarning: 6.0% of the points cannot be placed; you may want to decrease the
size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



3.4 PROYECTO DATA SCIENCE

[83]:
```
markets = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/markets_cleaned.
↪csv", index_col =0)

markets['num_items_sold']=markets.iloc[:,-29:].sum(axis=1)

state_pop = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
↪census-state-populations.csv", index_col =0)

markets = markets.merge(state_pop, on = "state", how = "left")
```

```
markets.head()
```

[83]:
```
                                             name      city     county  \
0  Caledonia Farmers Market Association - Danville  Danville  Caledonia
1                Stearns Homestead Farmers' Market     Parma   Cuyahoga
2                106 S. Main Street Farmers Market  Six Mile        NaN
3              10th Steet Community Farmers Market     Lamar     Barton
4                              112st Madison Avenue  New York   New York

            state        lat        lon  months_open  Bakedgoods  Beans  \
0         Vermont -72.140337  44.411036          3.0           1      1
1            Ohio -81.733939  41.374801          4.0           1      0
2  South Carolina -82.818700  34.804200          NaN           0      0
3        Missouri -94.274619  37.495628          8.0           1      0
4        New York -73.949300  40.793900          5.0           1      0

   Cheese  …  Prepared  Seafood  Soap  Tofu  Trees  Vegetables  \
0       1  …         1        0     1     0      1           1
1       0  …         0        0     1     0      0           1
2       0  …         0        0     0     0      0           0
3       0  …         1        0     1     0      0           1
4       0  …         1        0     1     0      0           1

   WildHarvested  Wine  num_items_sold  pop_est_2014
0              0     0              20      626562.0
1              0     0              12    11594163.0
2              0     0               0     4832482.0
3              0     0              13     6063589.0
4              0     0              10    19746227.0

[5 rows x 38 columns]
```

[84]:
```
### PRIMERA APROXIMACIÓN A LOS DATOS

# Print first three rows of data and transpose
first_rows = markets.head(3).transpose()
print(first_rows)

# Get descriptions of every column
col_descriptions = markets.describe(include = 'all',
                                    percentiles = [0.5]).transpose()
print(col_descriptions)
```

```
                                                          0  \
name            Caledonia Farmers Market Association - Danville
city                                                Danville
county                                             Caledonia
state                                                Vermont
```

```
lat                                                -72.140337
lon                                                 44.411036
months_open                                               3.0
Bakedgoods                                                  1
Beans                                                       1
Cheese                                                      1
Coffee                                                      1
Crafts                                                      1
Eggs                                                        1
Flowers                                                     1
Fruits                                                      1
Grains                                                      0
Herbs                                                       1
Honey                                                       1
Jams                                                        1
Juices                                                      0
Maple                                                       1
Meat                                                        1
Mushrooms                                                   1
Nursery                                                     0
Nuts                                                        0
PetFood                                                     1
Plants                                                      0
Poultry                                                     1
Prepared                                                    1
Seafood                                                     0
Soap                                                        1
Tofu                                                        0
Trees                                                       1
Vegetables                                                  1
WildHarvested                                               0
Wine                                                        0
num_items_sold                                             20
pop_est_2014                                           626562.0

                                                    1  \
name            Stearns Homestead Farmers' Market
city                                           Parma
county                                       Cuyahoga
state                                            Ohio
lat                                        -81.733939
lon                                         41.374801
months_open                                       4.0
Bakedgoods                                          1
Beans                                               0
Cheese                                              0
Coffee                                              0
Crafts                                              1
```

```
Eggs                                    1
Flowers                                 1
Fruits                                  1
Grains                                  0
Herbs                                   1
Honey                                   1
Jams                                    1
Juices                                  0
Maple                                   1
Meat                                    0
Mushrooms                               0
Nursery                                 0
Nuts                                    0
PetFood                                 0
Plants                                  0
Poultry                                 1
Prepared                                0
Seafood                                 0
Soap                                    1
Tofu                                    0
Trees                                   0
Vegetables                              1
WildHarvested                           0
Wine                                    0
num_items_sold                         12
pop_est_2014                     11594163.0

                                        2
name            106 S. Main Street Farmers Market
city                                 Six Mile
county                                    NaN
state                           South Carolina
lat                                  -82.8187
lon                                   34.8042
months_open                               NaN
Bakedgoods                              0
Beans                                   0
Cheese                                  0
Coffee                                  0
Crafts                                  0
Eggs                                    0
Flowers                                 0
Fruits                                  0
Grains                                  0
Herbs                                   0
Honey                                   0
Jams                                    0
Juices                                  0
```

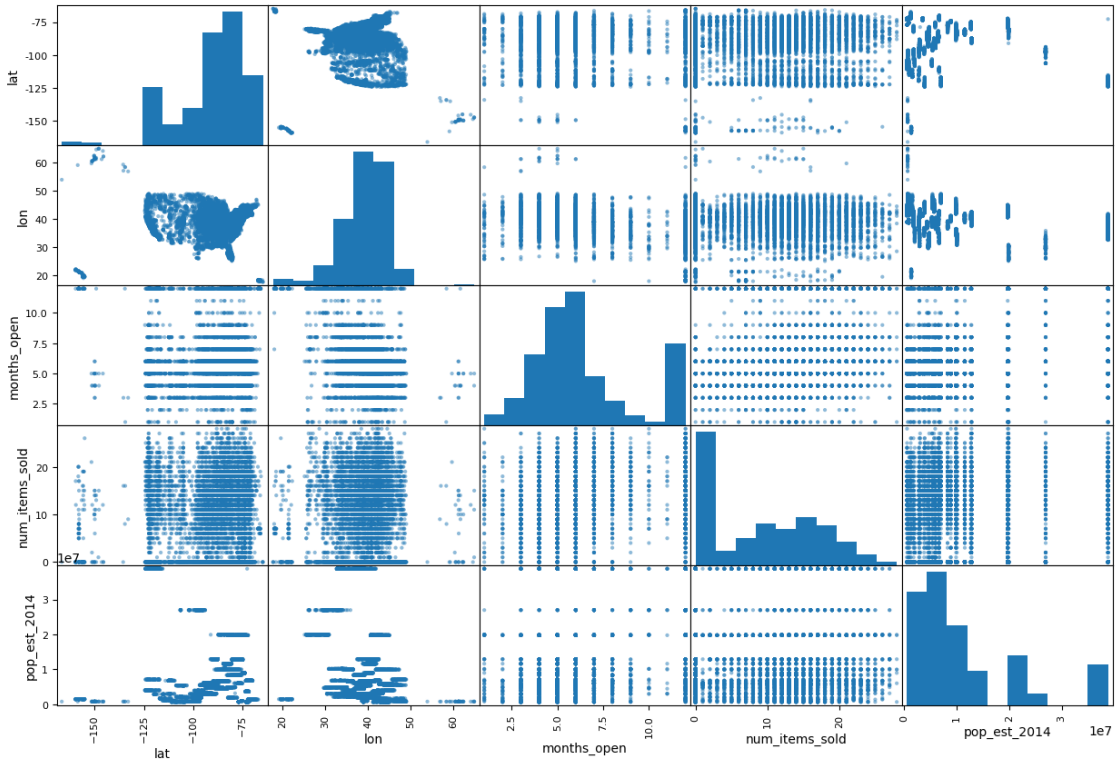
```
Maple                                    0
Meat                                     0
Mushrooms                                0
Nursery                                  0
Nuts                                     0
PetFood                                  0
Plants                                   0
Poultry                                  0
Prepared                                 0
Seafood                                  0
Soap                                     0
Tofu                                     0
Trees                                    0
Vegetables                               0
WildHarvested                            0
Wine                                     0
num_items_sold                           0
pop_est_2014                     4832482.0
                 count unique                top freq            mean  \
name              8739   8158  El Mercado Familiar   33             NaN
city              8699   4697              Chicago   62             NaN
county            8228   1503          Los Angeles  121             NaN
state             8739     53           California  760             NaN
lat             8710.0    NaN                  NaN  NaN      -90.925689
lon             8710.0    NaN                  NaN  NaN       39.133725
months_open     5452.0    NaN                  NaN  NaN        6.456346
Bakedgoods      8739.0    NaN                  NaN  NaN        0.592516
Beans           8739.0    NaN                  NaN  NaN        0.098638
Cheese          8739.0    NaN                  NaN  NaN        0.330015
Coffee          8739.0    NaN                  NaN  NaN        0.240417
Crafts          8739.0    NaN                  NaN  NaN        0.416867
Eggs            8739.0    NaN                  NaN  NaN        0.503948
Flowers         8739.0    NaN                  NaN  NaN        0.458748
Fruits          8739.0    NaN                  NaN  NaN        0.549949
Grains          8739.0    NaN                  NaN  NaN        0.097265
Herbs           8739.0    NaN                  NaN  NaN        0.523286
Honey           8739.0    NaN                  NaN  NaN        0.546401
Jams            8739.0    NaN                  NaN  NaN        0.542625
Juices          8739.0    NaN                  NaN  NaN        0.166266
Maple           8739.0    NaN                  NaN  NaN        0.216501
Meat            8739.0    NaN                  NaN  NaN         0.37018
Mushrooms       8739.0    NaN                  NaN  NaN        0.160659
Nursery         8739.0    NaN                  NaN  NaN        0.037075
Nuts            8739.0    NaN                  NaN  NaN        0.199222
PetFood         8739.0    NaN                  NaN  NaN        0.129763
Plants          8739.0    NaN                  NaN  NaN        0.439295
Poultry         8739.0    NaN                  NaN  NaN        0.304383
Prepared        8739.0    NaN                  NaN  NaN        0.411031
```

Seafood	8739.0	NaN	NaN	NaN	0.166495
Soap	8739.0	NaN	NaN	NaN	0.45646
Tofu	8739.0	NaN	NaN	NaN	0.026319
Trees	8739.0	NaN	NaN	NaN	0.187779
Vegetables	8739.0	NaN	NaN	NaN	0.652935
WildHarvested	8739.0	NaN	NaN	NaN	0.098066
Wine	8739.0	NaN	NaN	NaN	0.116489
num_items_sold	8739.0	NaN	NaN	NaN	9.039593
pop_est_2014	8693.0	NaN	NaN	NaN	11135572.386288

	std	min	50%	max
name	NaN	NaN	NaN	NaN
city	NaN	NaN	NaN	NaN
county	NaN	NaN	NaN	NaN
state	NaN	NaN	NaN	NaN
lat	17.366333	-166.54	-86.249268	-64.7043
lon	5.286239	17.7099	40.017753	64.86275
months_open	2.743168	1.0	6.0	12.0
Bakedgoods	0.491394	0.0	1.0	1.0
Beans	0.298193	0.0	0.0	1.0
Cheese	0.470245	0.0	0.0	1.0
Coffee	0.427361	0.0	0.0	1.0
Crafts	0.493069	0.0	0.0	1.0
Eggs	0.500013	0.0	1.0	1.0
Flowers	0.498324	0.0	0.0	1.0
Fruits	0.497527	0.0	1.0	1.0
Grains	0.296335	0.0	0.0	1.0
Herbs	0.499486	0.0	1.0	1.0
Honey	0.497871	0.0	1.0	1.0
Jams	0.498208	0.0	1.0	1.0
Juices	0.372341	0.0	0.0	1.0
Maple	0.411883	0.0	0.0	1.0
Meat	0.48288	0.0	0.0	1.0
Mushrooms	0.367237	0.0	0.0	1.0
Nursery	0.188957	0.0	0.0	1.0
Nuts	0.399438	0.0	0.0	1.0
PetFood	0.336062	0.0	0.0	1.0
Plants	0.49633	0.0	0.0	1.0
Poultry	0.460172	0.0	0.0	1.0
Prepared	0.492049	0.0	0.0	1.0
Seafood	0.372546	0.0	0.0	1.0
Soap	0.498129	0.0	0.0	1.0
Tofu	0.160091	0.0	0.0	1.0
Trees	0.390558	0.0	0.0	1.0
Vegetables	0.476064	0.0	1.0	1.0
WildHarvested	0.297421	0.0	0.0	1.0
Wine	0.320829	0.0	0.0	1.0
num_items_sold	7.84534	0.0	10.0	28.0

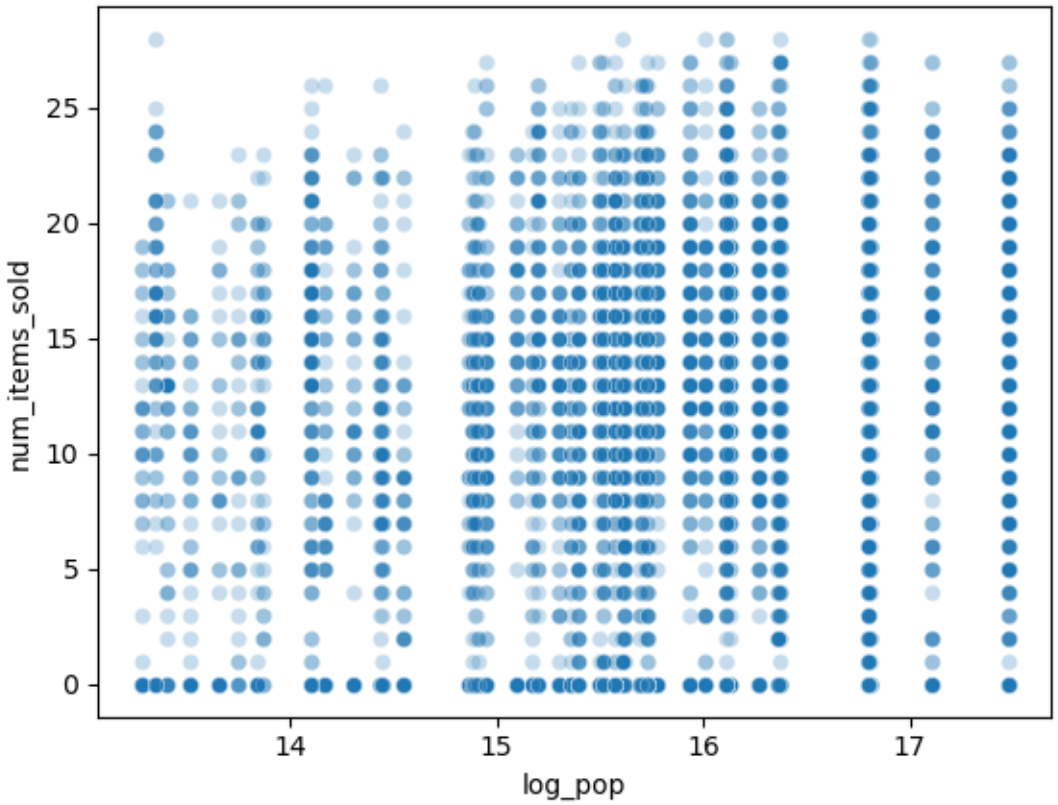
```
pop_est_2014     10479682.324487  584153.0  6745408.0  38802500.0
```

[85]:
```
### Matriz scatter de columnas numéricas

# Select just the numeric columns (exluding individual goods)
numeric_columns = ['lat', 'lon', 'months_open', 'num_items_sold',␣
 ↪'pop_est_2014']

# Make a scatter matrix of numeric columns
pd.plotting.scatter_matrix(markets[numeric_columns],
                           # Make figure larget to show details
                           figsize=(15, 10),
                            # Lower point opacity to show overlap
                            alpha=0.5,
                           );
```



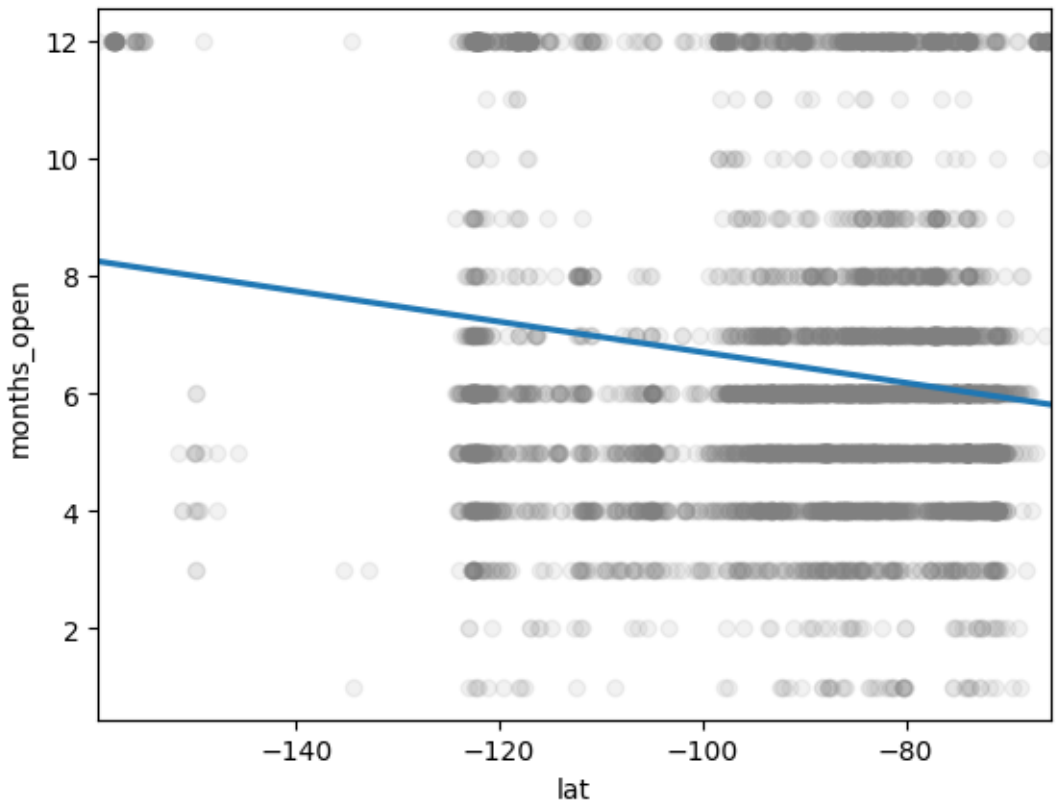
[86]:
```
# Create a new logged population column
markets['log_pop'] = np.log(markets['pop_est_2014'])

# Draw a scatterplot of log-population to # of items sold
sns.scatterplot(x = 'log_pop',
                y = 'num_items_sold',
                # Reduce point opacity to show overlap
```

```
                    alpha = 0.25,
                    data = markets)

plt.show()
```



[87]:
```
# ¿LA LATITUD SE RELACIONA CON LOS MESES ABIERTOS?
sns.regplot(x='lat',
            y='months_open',
            # Set scatter point opacity & color
            scatter_kws={'alpha':0.1, 'color':'gray'},
            # Disable confidence band
            ci=False,
            data=markets)
```

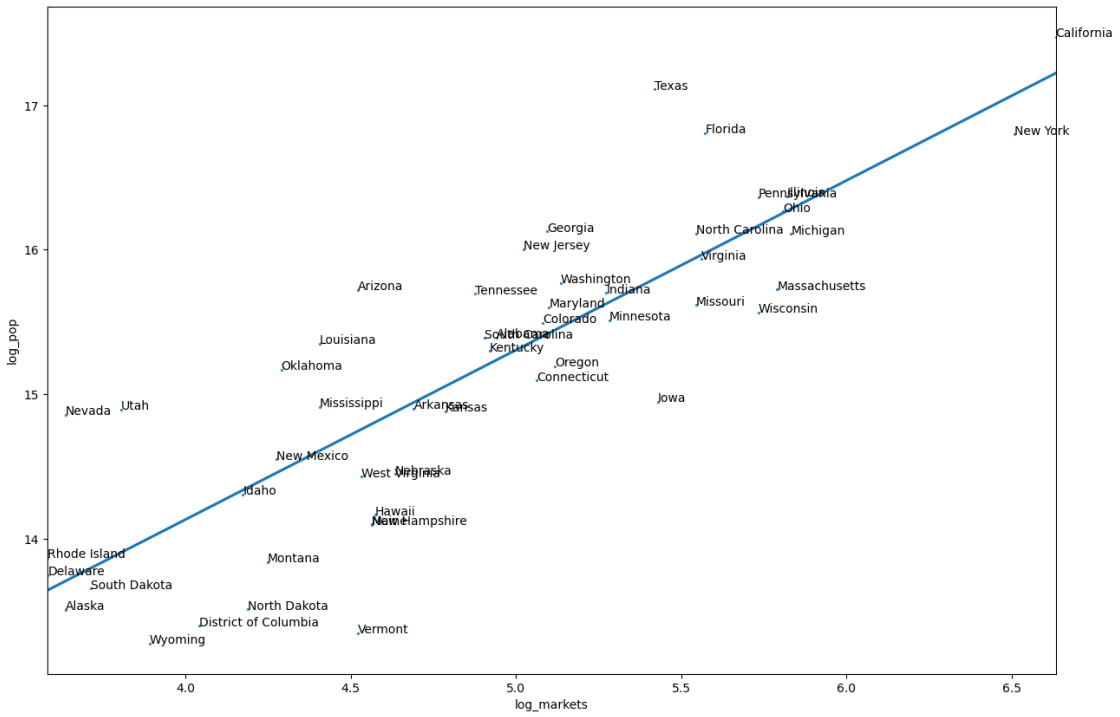
[87]: <AxesSubplot:xlabel='lat', ylabel='months_open'>



[88]:
```
# ¿QUÉ ESTADO ES EL MÁS MARKET-FRIENDLY?
markets_and_pop = (markets.groupby('state', as_index = False).agg({'name':␣
 ↪lambda d: np.log(len(d)),
       'pop_est_2014': lambda d: np.log(d.iloc[0])})
    .rename(columns = {
        'name': 'log_markets',
        'pop_est_2014': 'log_pop' }))

fig, ax = plt.subplots(figsize=(15, 10));
g = sns.regplot(
    "log_markets", "log_pop",
    ci = False,
    # Shrink scatter plot points
    scatter_kws = {'s':2},
    data = markets_and_pop,
    ax=ax
)

# Iterate over the rows of the data
for _, row in markets_and_pop.iterrows():
    state, log_markets, log_pop = row
```

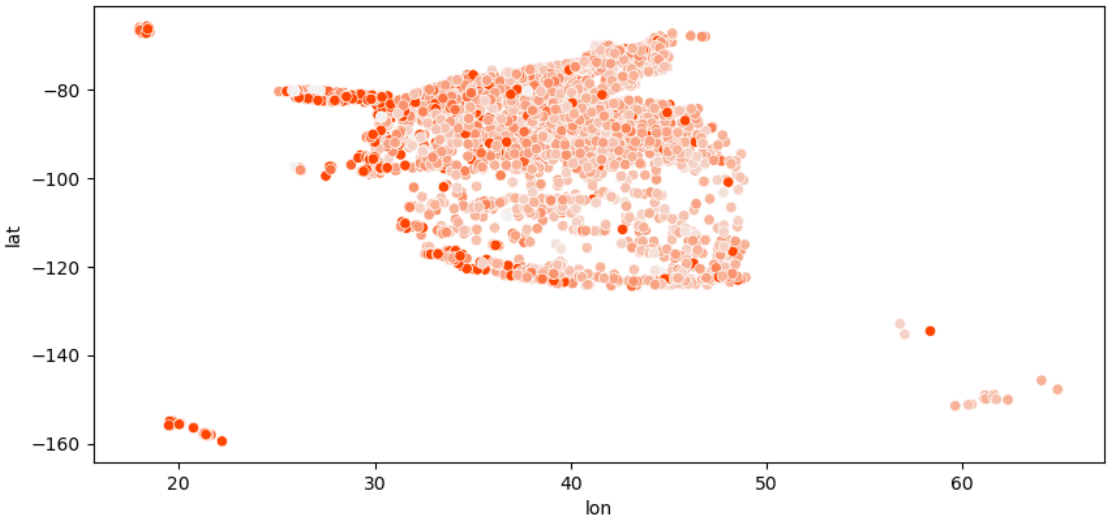
```
    # Place annotation and reduce size for clarity
    g.annotate(state, (log_markets, log_pop), size=10);
```

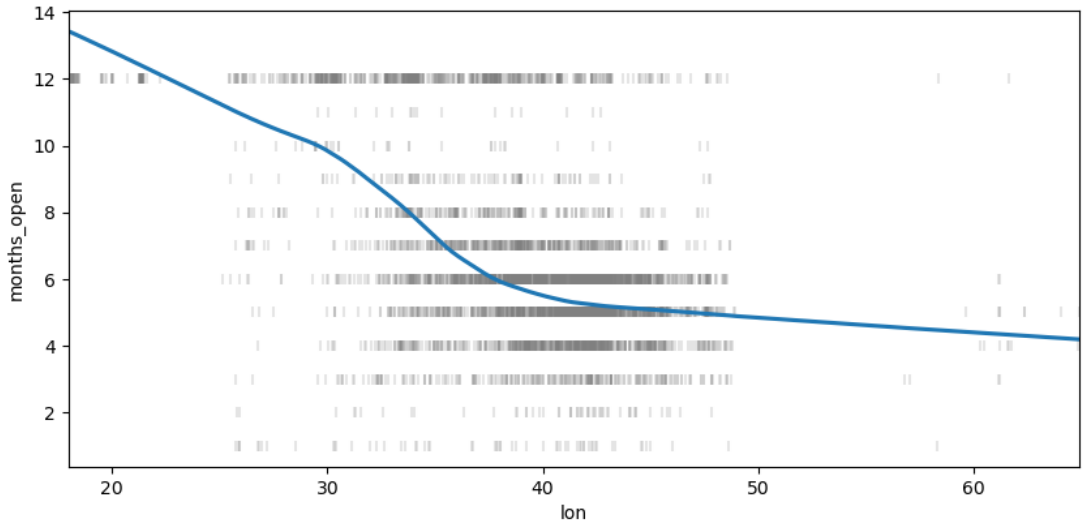
```
C:\Users\marco\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[89]:
```
# VISUALIZACIONES DIFERENTES SIMULTÁNEAS

# Setup two stacked plots
_, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 10))

# Draw location scatter plot on first plot
sns.scatterplot(x = "lon", y = "lat", hue = "months_open",
                palette=sns.light_palette("orangered", n_colors=12),
                legend=False, data=markets,
                ax=ax1);

# Plot a regression plot on second plot
sns.regplot(x = "lon", y = "months_open",
            scatter_kws={'alpha':0.2, 'color':'gray', 'marker': '|'},
```

```
            lowess=True,
            marker='|', data=markets,
            ax=ax2);
```



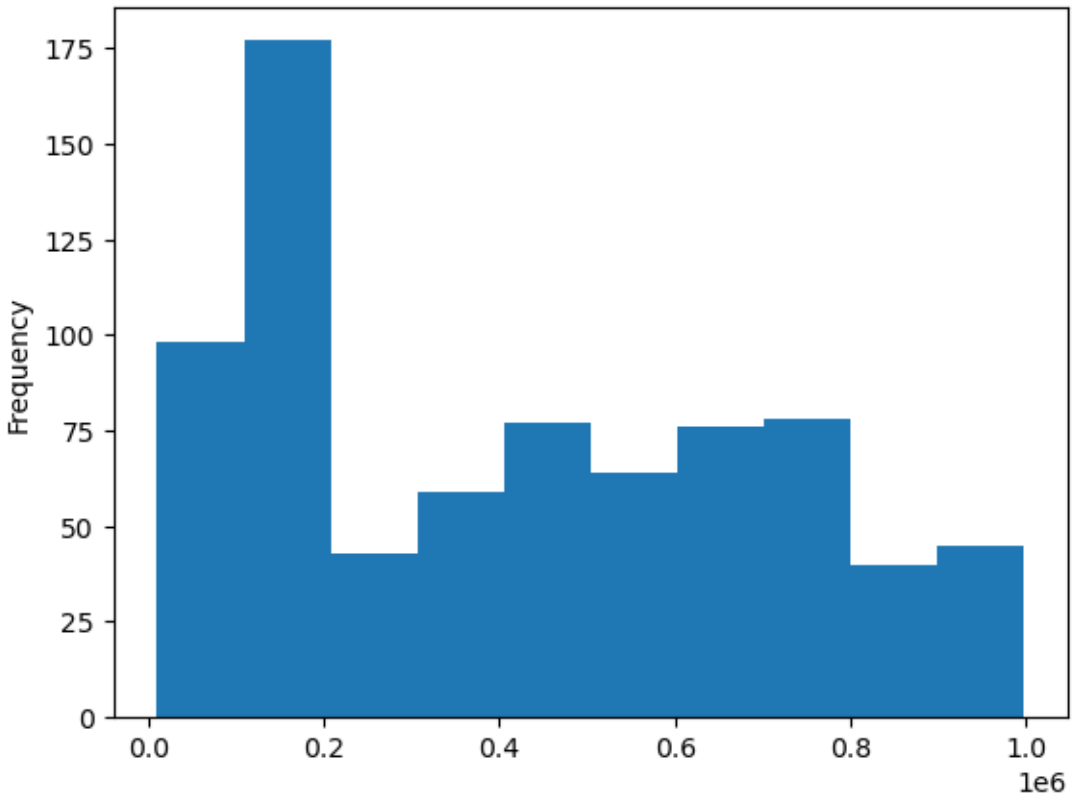


4 SEABORN INTERMEDIO

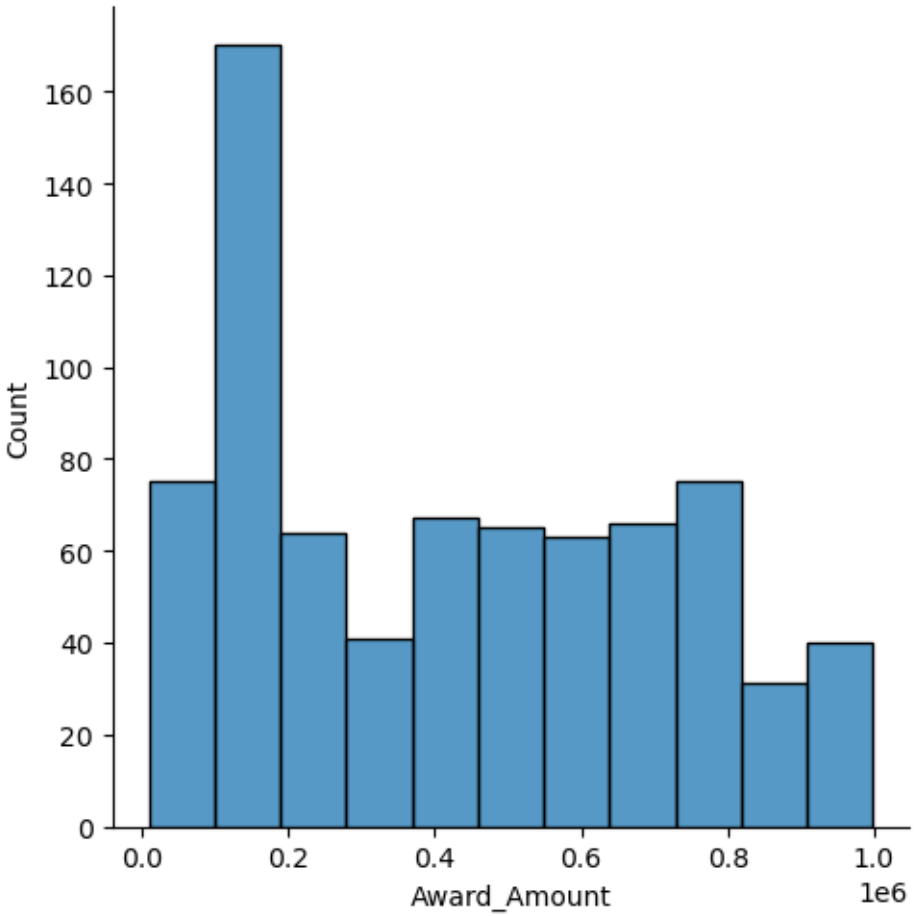
4.1 INTRODUCCIÓN A SEABORN

[90]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

```
grants = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
 ↪schoolimprovement2010grants.csv")

# Display pandas histogram
grants['Award_Amount'].plot.hist()
plt.show()

# Clear out the pandas histogram
plt.clf()

# Display a Seaborn displot
sns.displot(grants['Award_Amount'])
plt.show()

# Clear the displot
plt.clf()
```



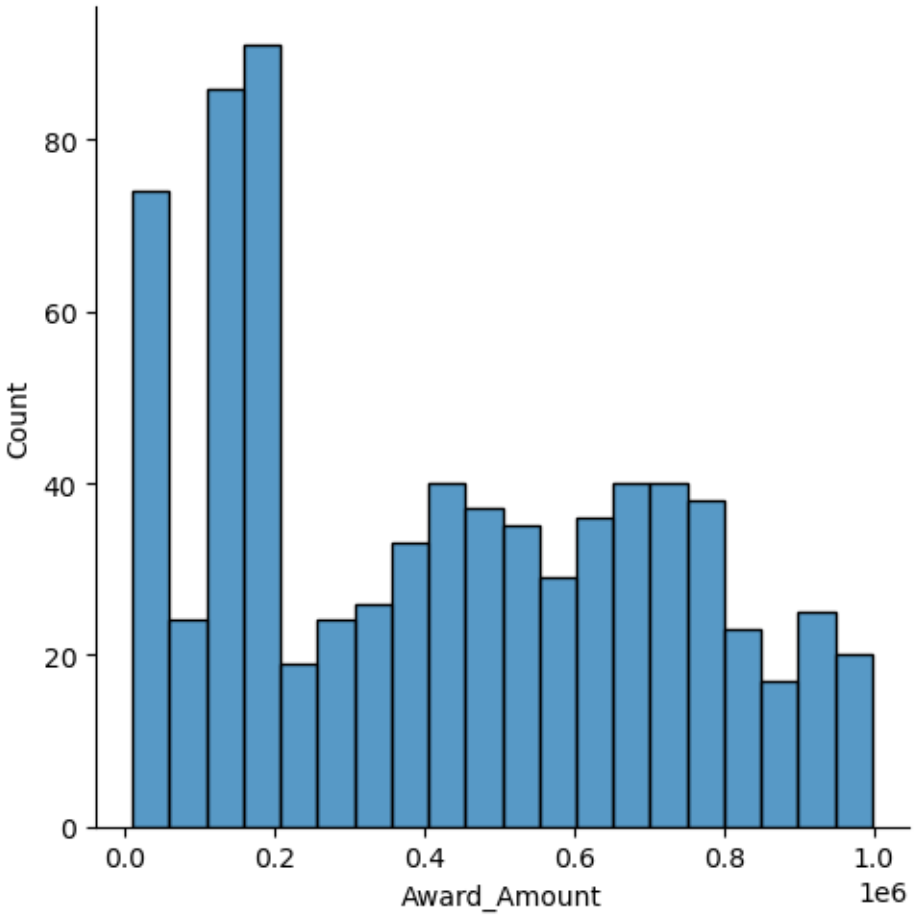

```
<Figure size 640x480 with 0 Axes>
```



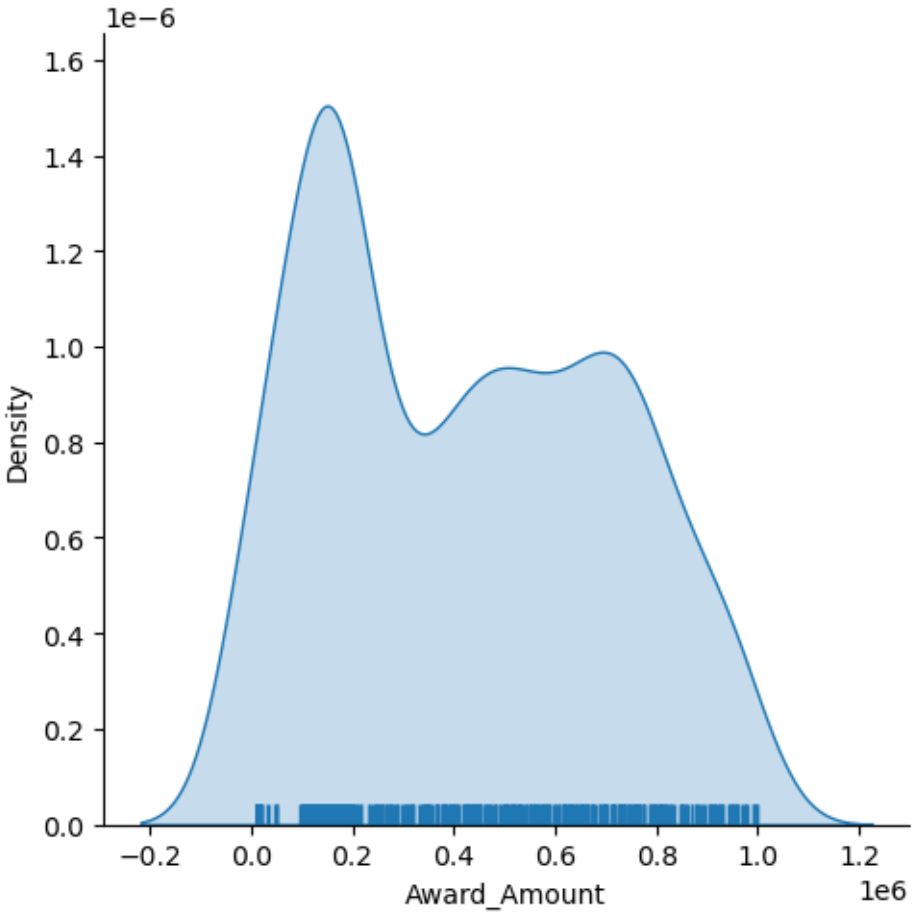

```
<Figure size 640x480 with 0 Axes>
```

4.1.1 Displot

[91]:
```
# Create a displot
sns.displot(grants['Award_Amount'],
            bins=20)

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

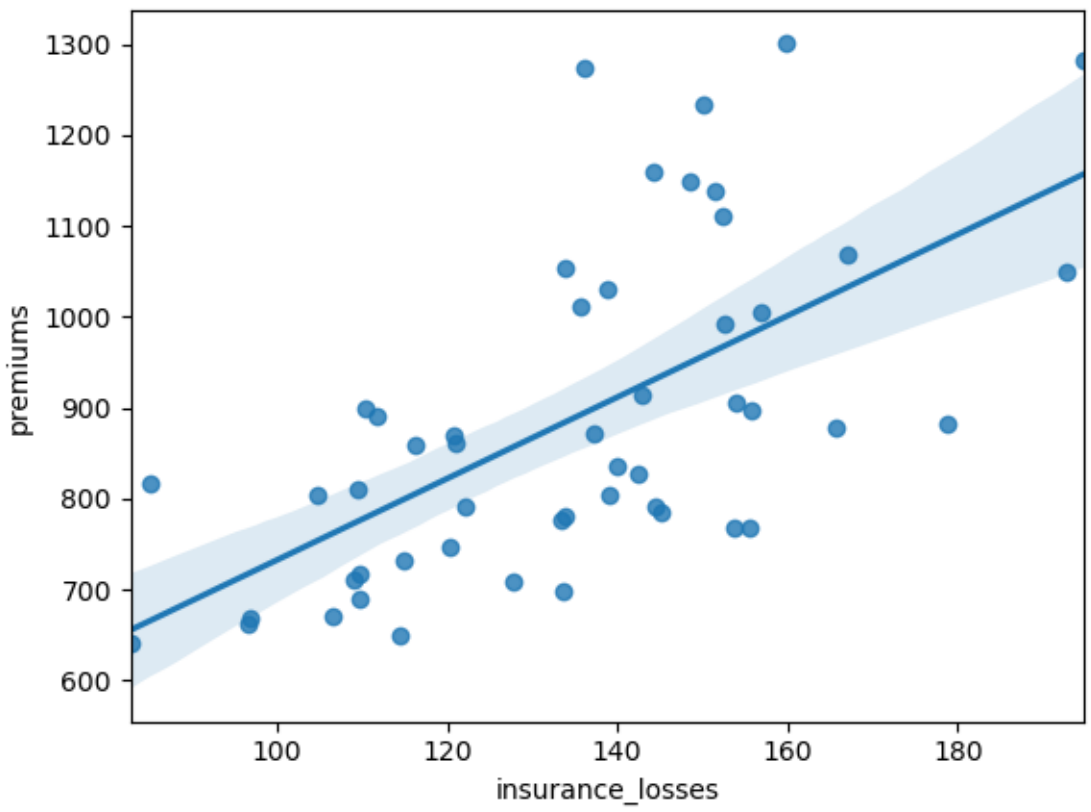


[92]:
```
# Create a displot of the Award Amount
sns.displot(grants['Award_Amount'],
            kind='kde',
            rug=True,
            fill=True)

# Plot the results
plt.show()
```

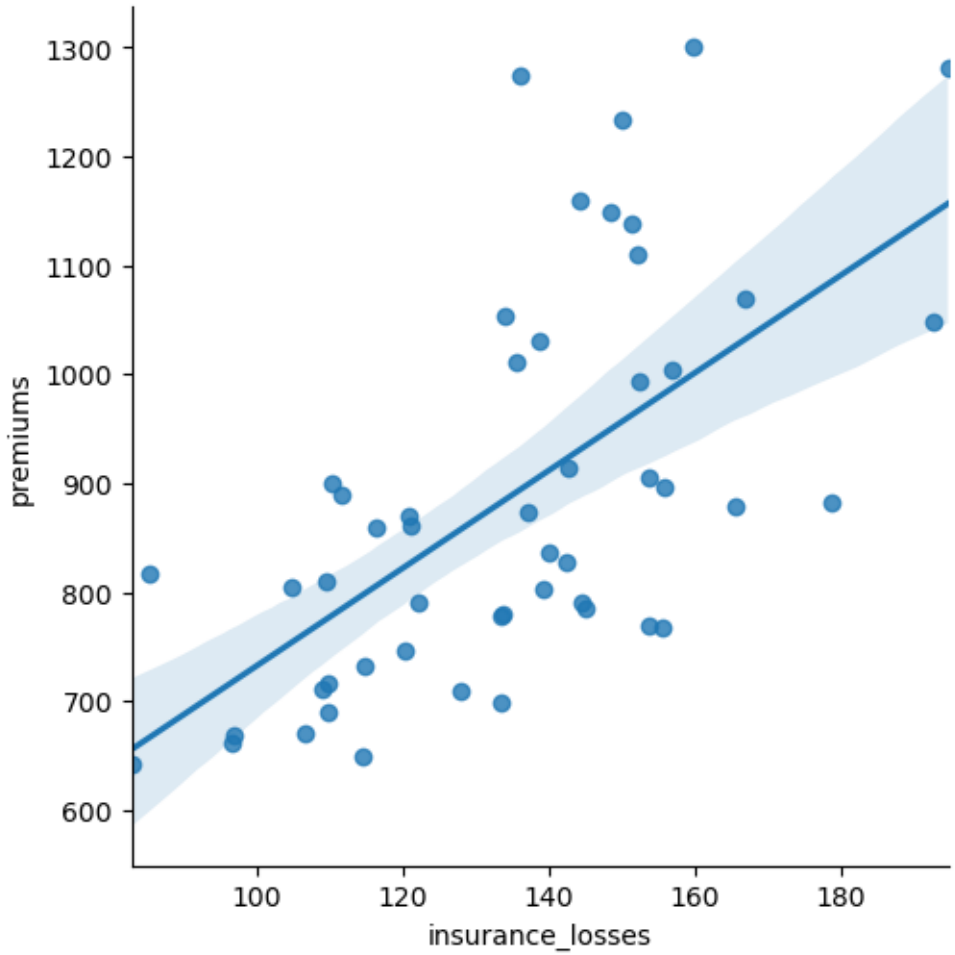


4.1.2 Regression plots

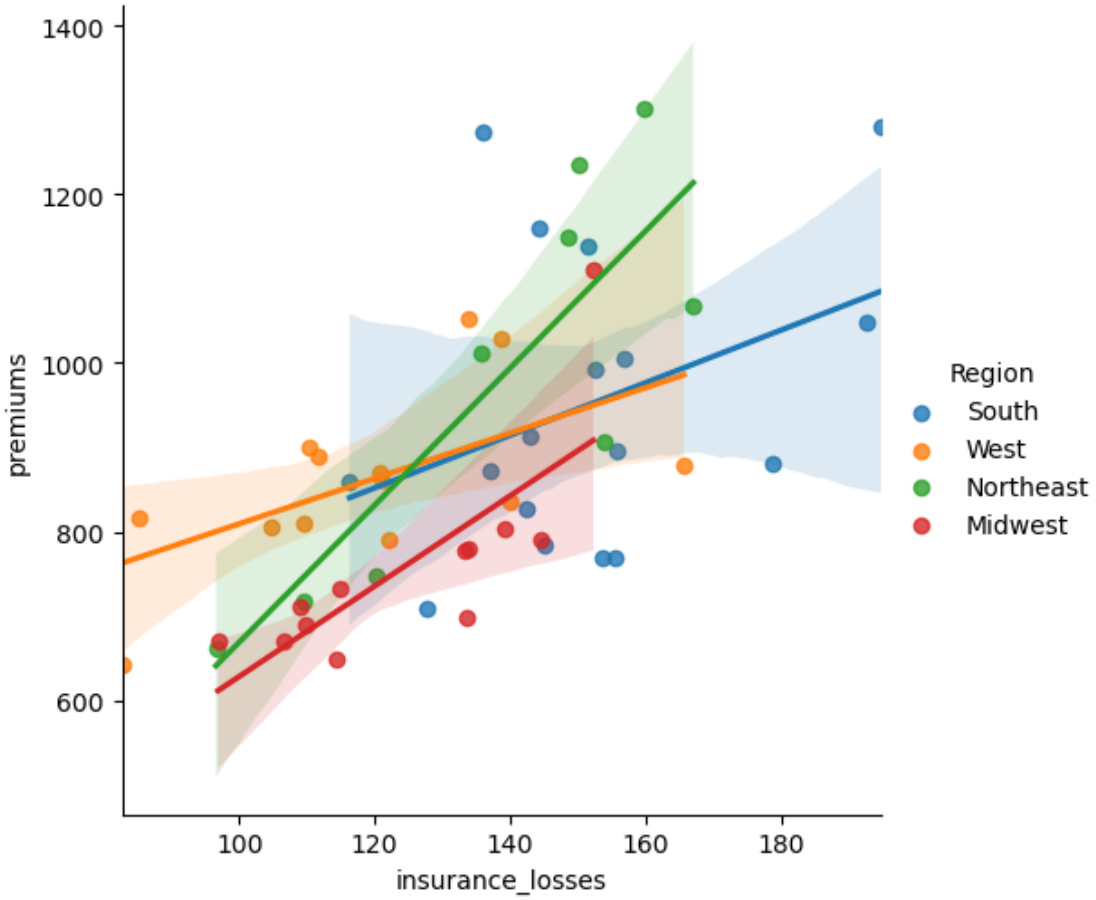
[93]:
```
insurance = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
 ↪insurance_premiums.csv")

# Create a regression plot of premiums vs. insurance_losses
sns.regplot(x = "insurance_losses",
            y = "premiums",
            data = insurance)

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



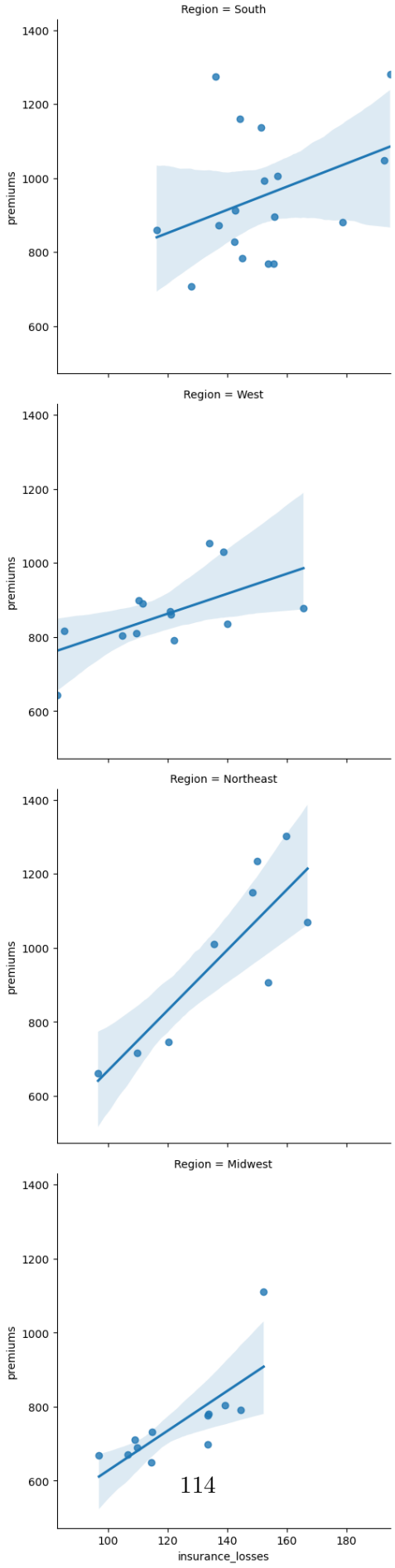
[94]:
```
# Create an lmplot of premiums vs. insurance_losses
sns.lmplot(x = "insurance_losses",
           y = "premiums",
           data = insurance)

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



[95]:
```
# Create a regression plot using hue
sns.lmplot(data=insurance,
           x="insurance_losses",
           y="premiums",
           hue="Region")

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

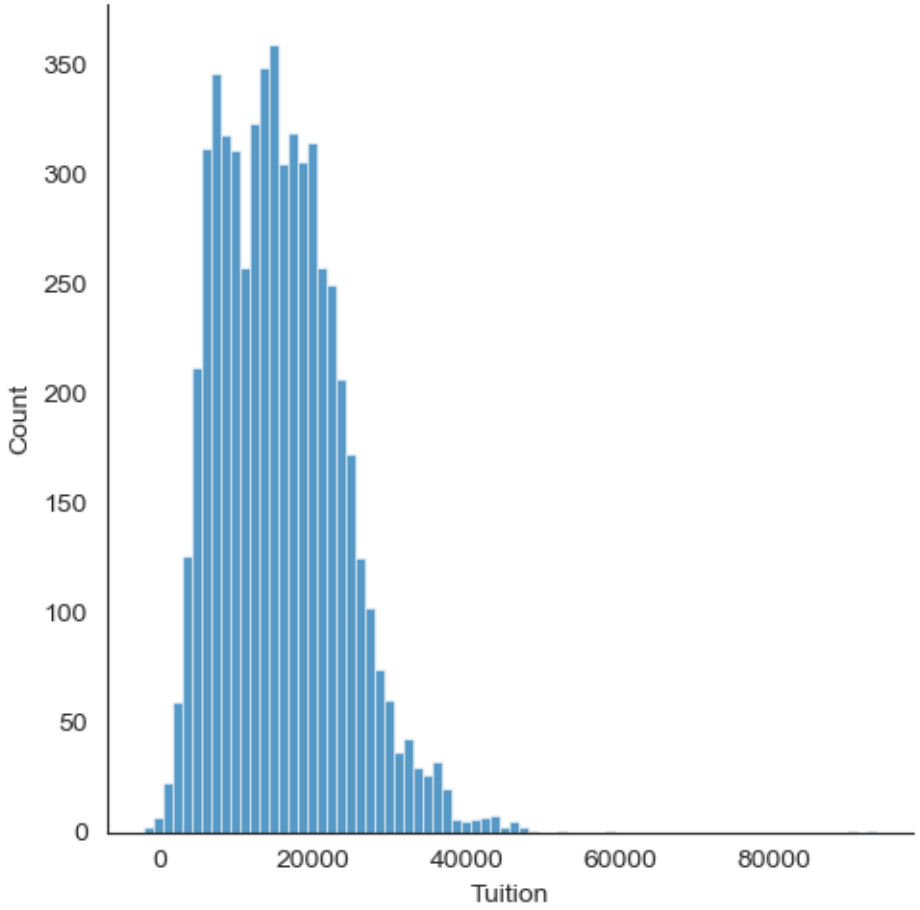


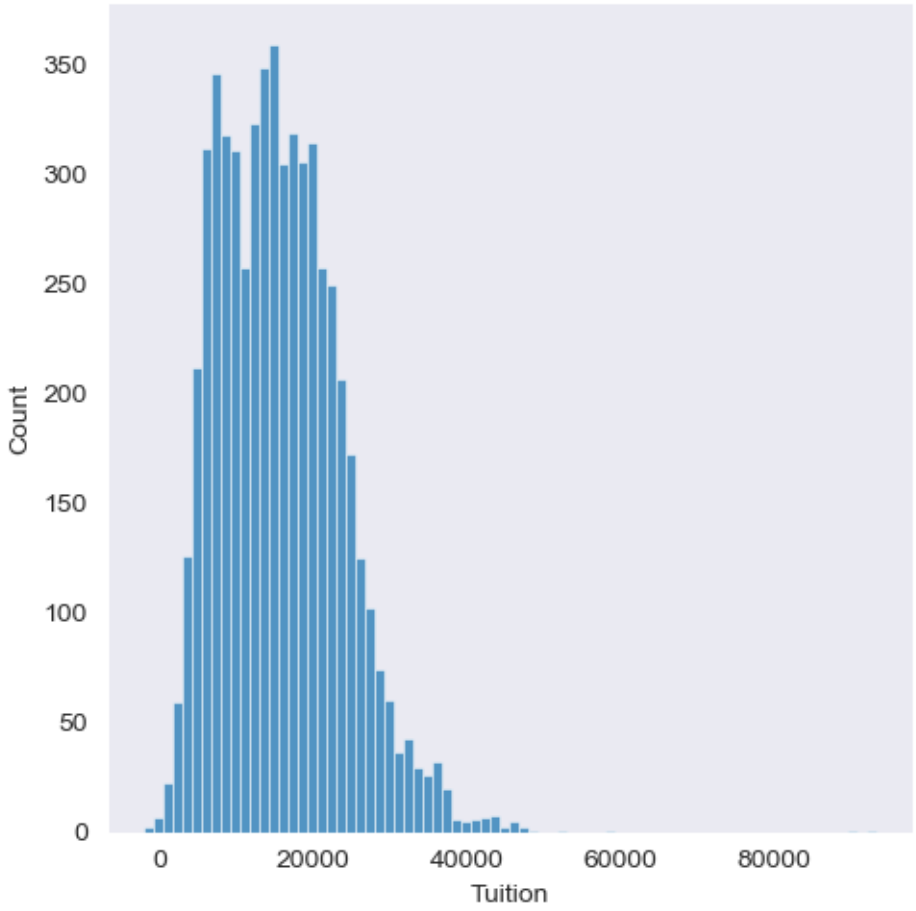
[96]:
```
# Create a regression plot with multiple rows
sns.lmplot(data=insurance,
           x="insurance_losses",
           y="premiums",
           row="Region")

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

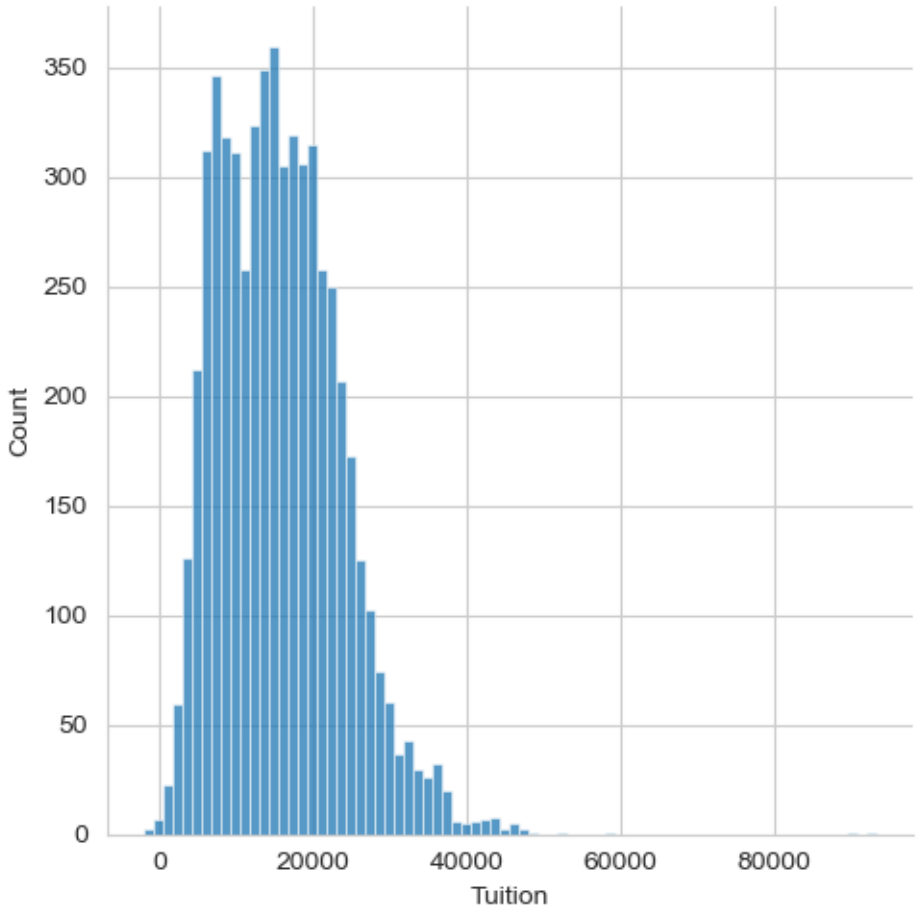

Region = South
Region = West
Region = Northeast
Region = Midwest
premiums
insurance_losses
1400
1200
1000
800
600
100
120
140
160
180

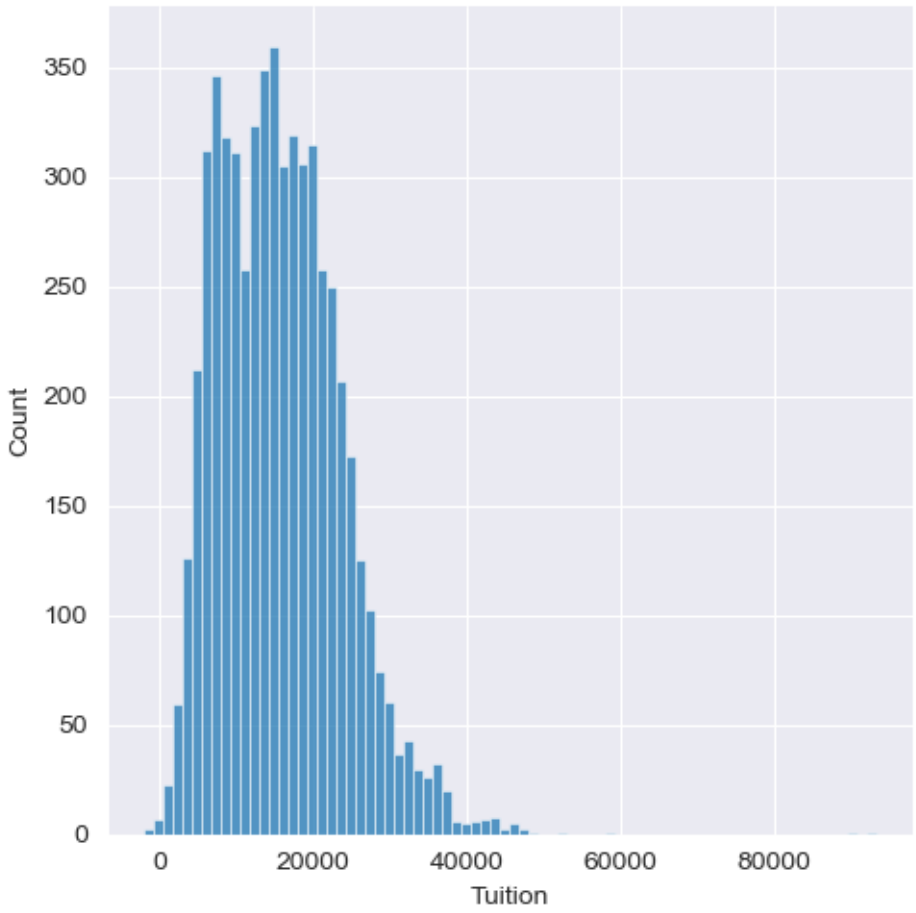
4.2 PERSONALIZACIÓN

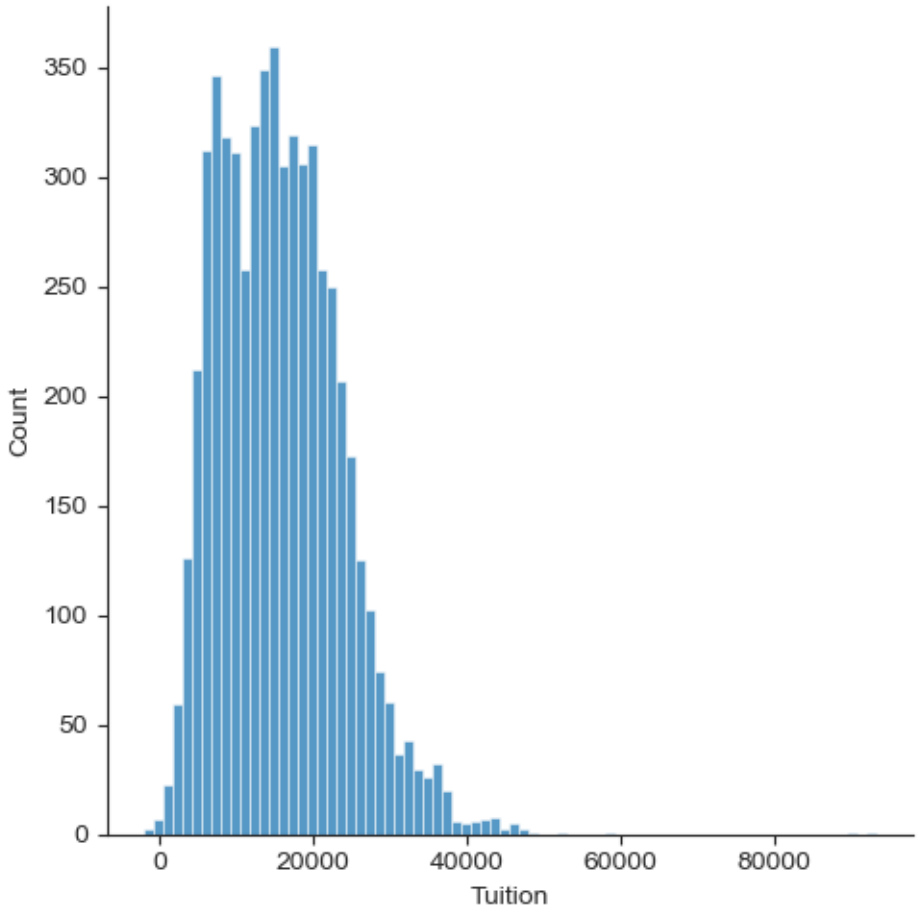
[97]:
```
college = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/college_datav3.
 ↪csv")

### TIP PARA CHECAR VARIOS ESTILOS:

for style in ["white", "dark", "whitegrid", "darkgrid", "ticks"]:
    sns.set_style(style)
    sns.displot(college["Tuition"])
    plt.show()
```



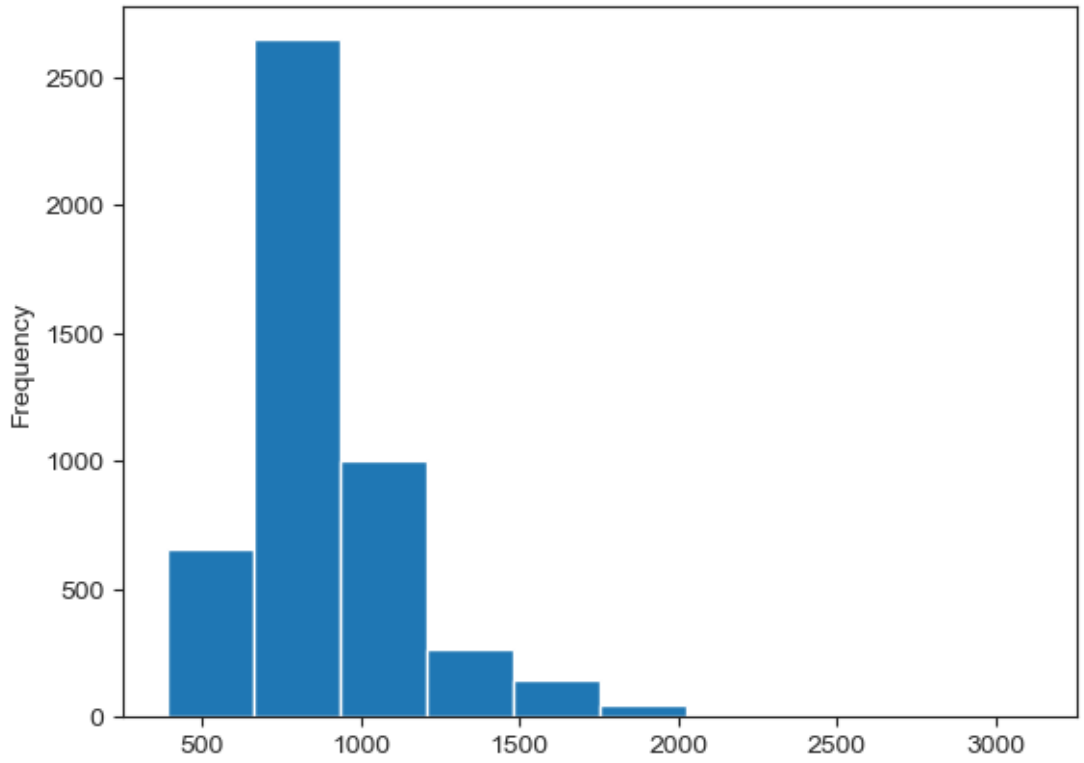

350
300
250
200
150
100
50
0
Count
0
20000
40000
60000
80000
Tuition

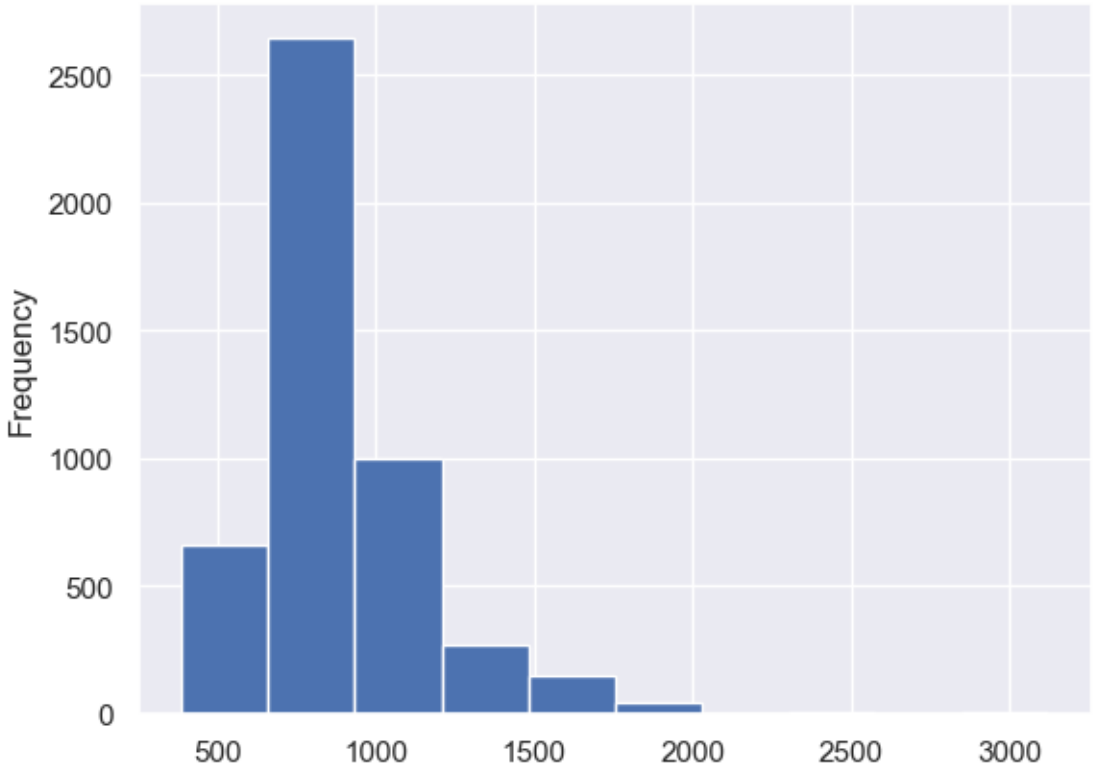

350
300
250
200
150
100
50
0
Count
0
20000
40000
60000
80000
Tuition


350
300
250
200
150
100
50
0
Count
0
20000
40000
60000
80000
Tuition



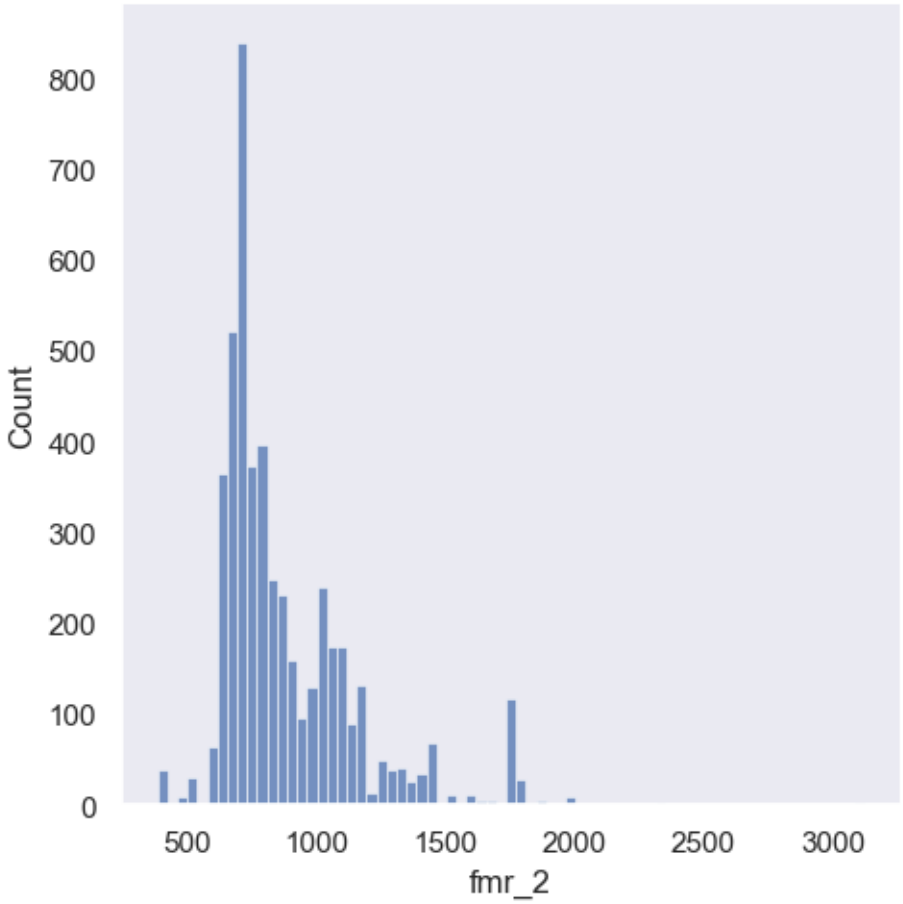
[98]:
```
fmr = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/FY18_4050_FMRs.csv")

# Plot the pandas histogram
fmr['fmr_2'].plot.hist()
plt.show()
plt.clf()

# Set the default seaborn style
sns.set()

# Plot the pandas histogram again
fmr['fmr_2'].plot.hist()
plt.show()
plt.clf()

### RESETEAR ESTILO:

# plt.clf()
```


Frequency
2500
2000
1500
1000
500
0
500
1000
1500
2000
2500
3000


Frequency
2500
2000
1500
1000
500
0
500
1000
1500
2000
2500
3000

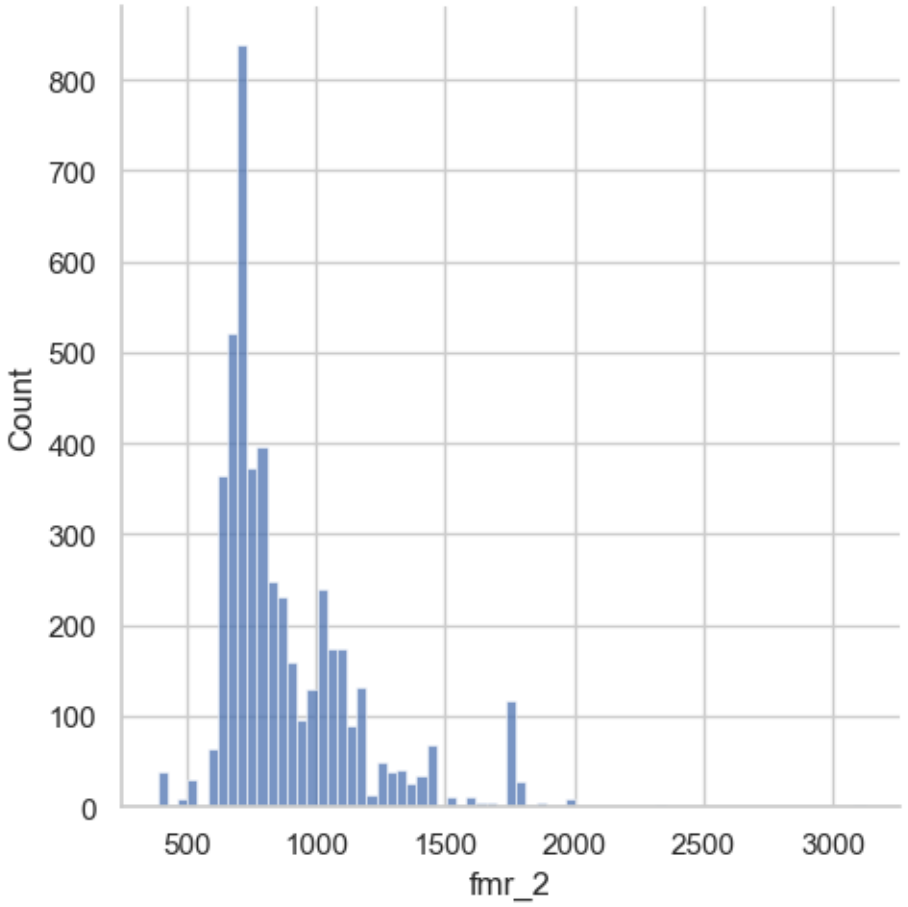
```
<Figure size 640x480 with 0 Axes>
```

[99]:
```
sns.set_style('dark')
sns.displot(fmr['fmr_2'])
plt.show()
plt.clf()
```



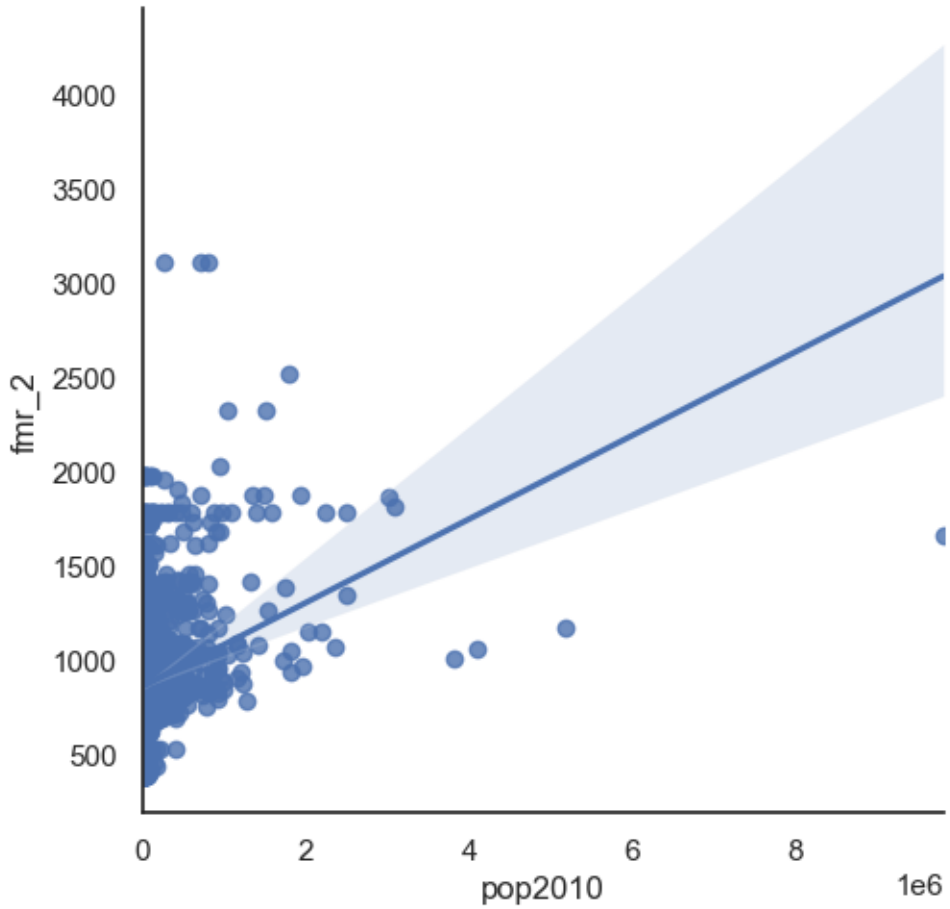

```
<Figure size 640x480 with 0 Axes>
```

[100]:
```
sns.set_style('whitegrid')
sns.displot(fmr['fmr_2'])
plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

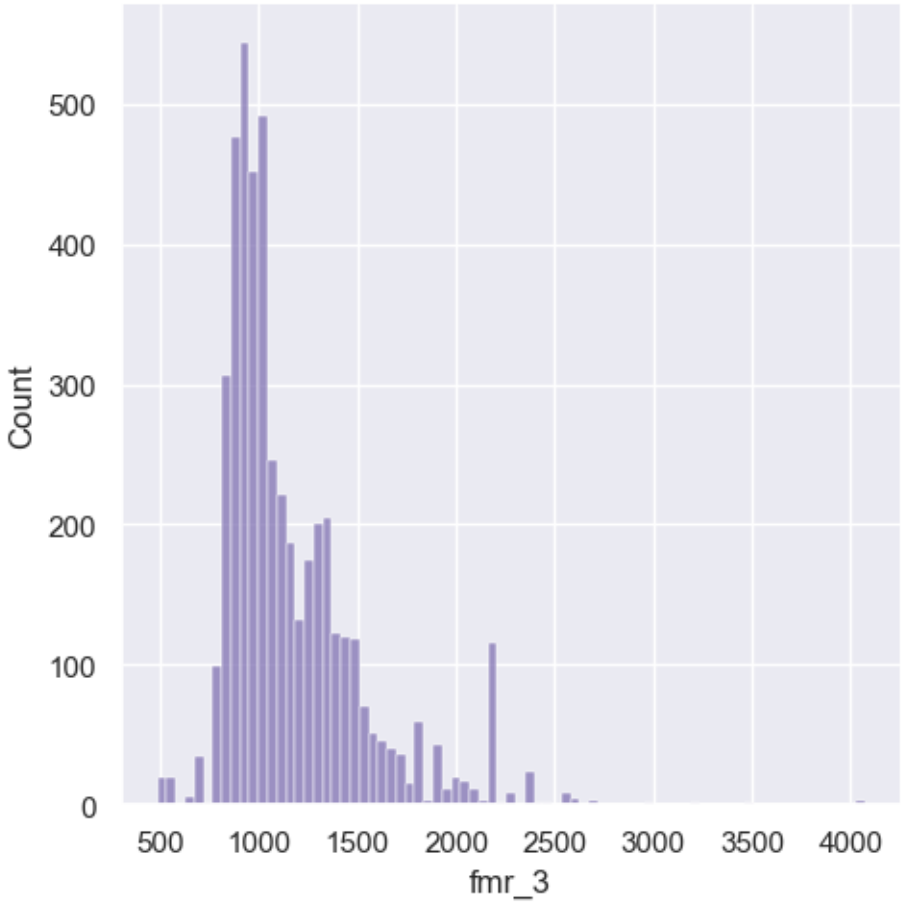
[101]:
```
# Set the style to white
sns.set_style('white')

# Create a regression plot
sns.lmplot(data=fmr,
           x='pop2010',
           y='fmr_2')

# Remove the spines
sns.despine()

# Show the plot and clear the figure
plt.show()
plt.clf()
```



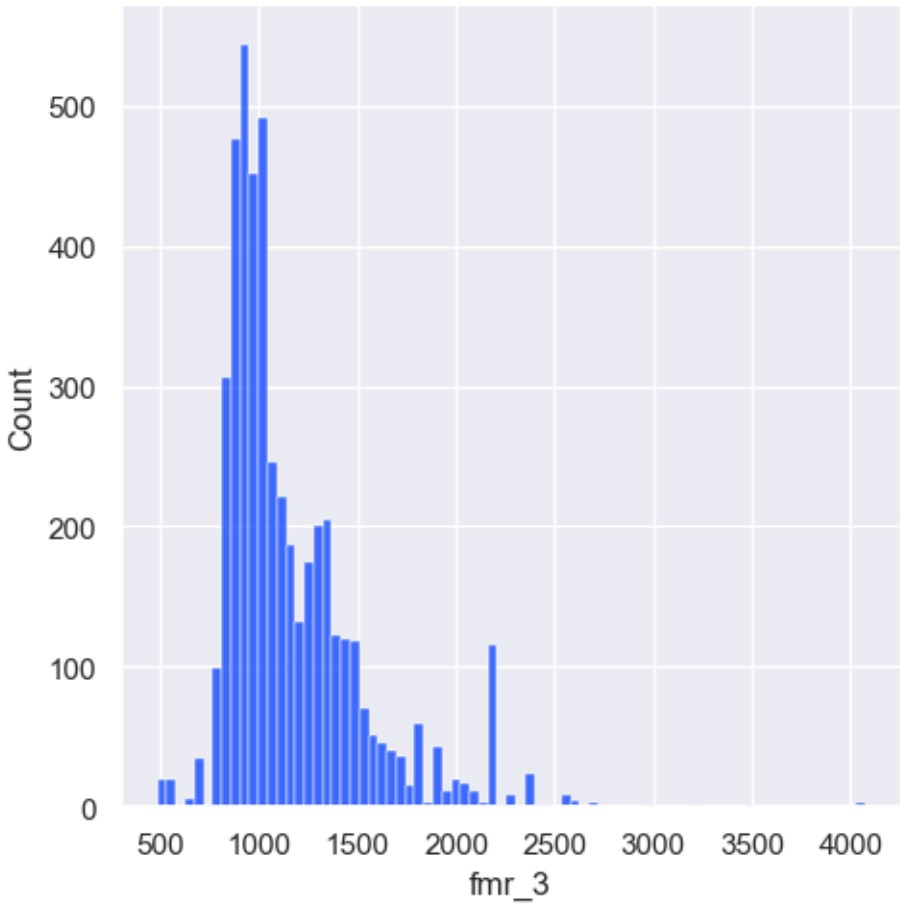

```
<Figure size 640x480 with 0 Axes>
```

4.2.1 Colores

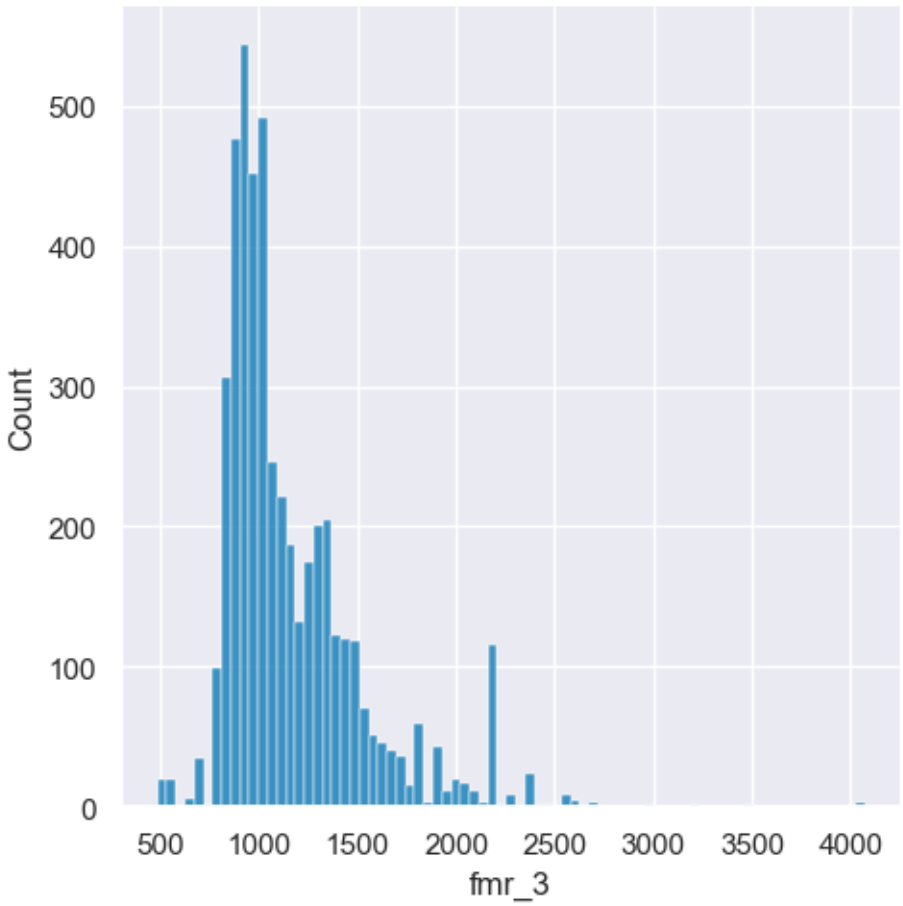
[102]:
```
# Set style, enable color code, and create a magenta distplot
sns.set(color_codes = True)
sns.displot(fmr['fmr_3'],
            color = 'm')

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



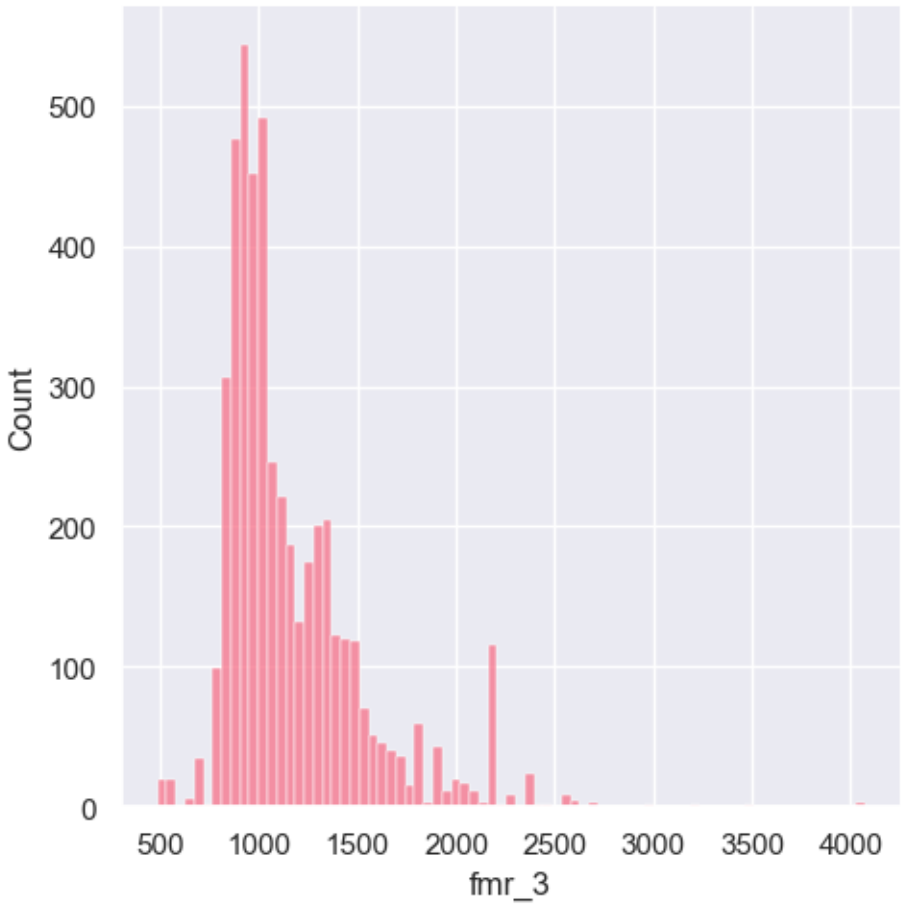
[106]:
```
### TIP PARA CHECAR COLORES

# Loop through differences between bright and colorblind palettes
for p in ['bright', 'colorblind', "husl", "Set2", "flare", "pastel"]:
    sns.set_palette(p)
    sns.displot(fmr['fmr_3'])
    plt.show()

    # Clear the plots
    plt.clf()
```



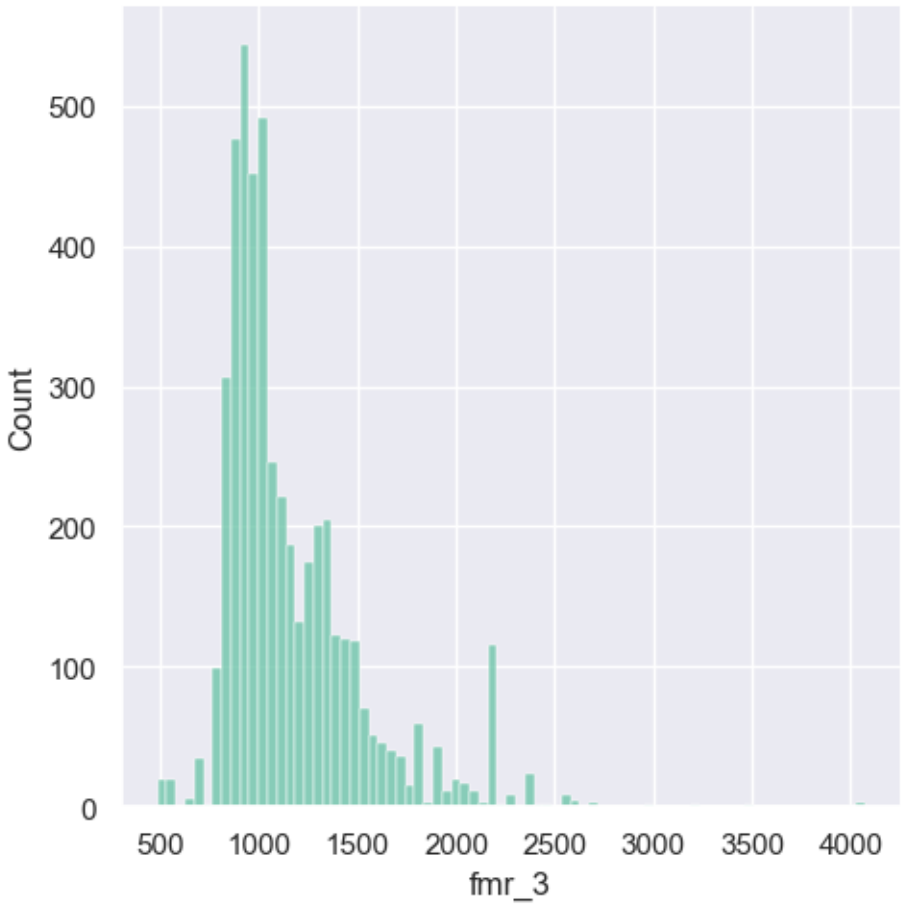

```
<Figure size 640x480 with 0 Axes>
```



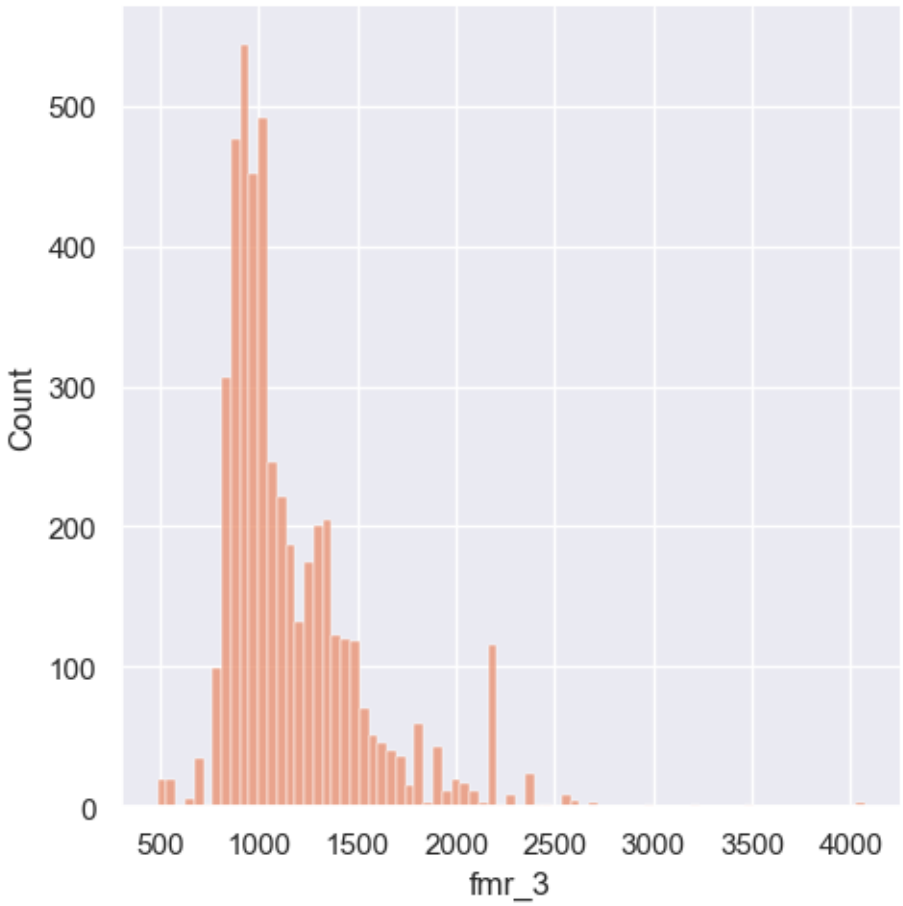

```
<Figure size 640x480 with 0 Axes>
```



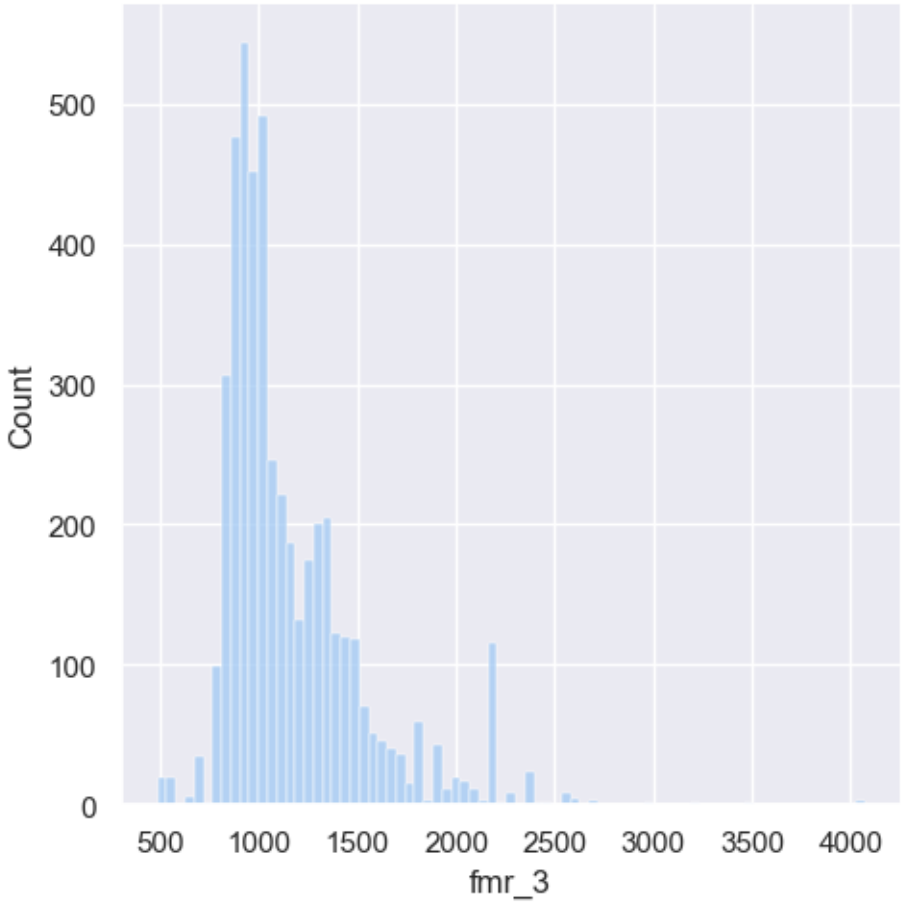

```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```



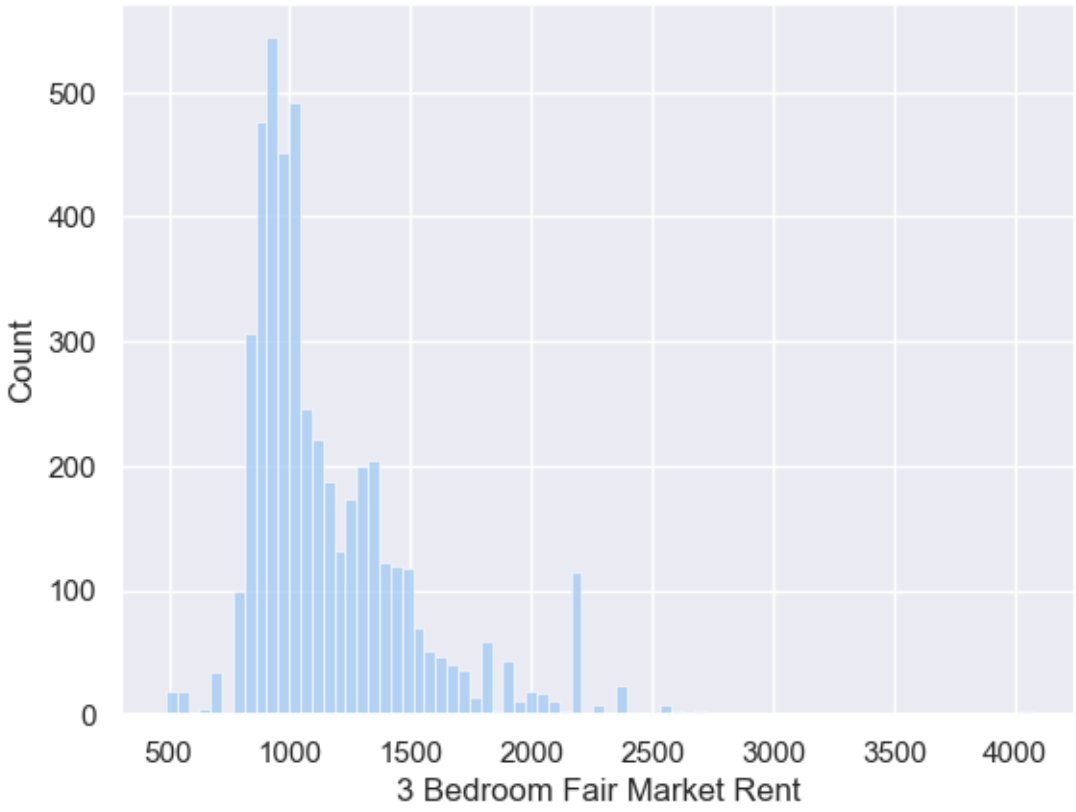

```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```

4.2.2 Ejes

[109]:
```
# Create a figure and axes
fig, ax = plt.subplots()

# Plot the distribution of data
sns.histplot(fmr['fmr_3'],
             ax = ax)

# Create a more descriptive x axis label
ax.set(xlabel = "3 Bedroom Fair Market Rent")

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



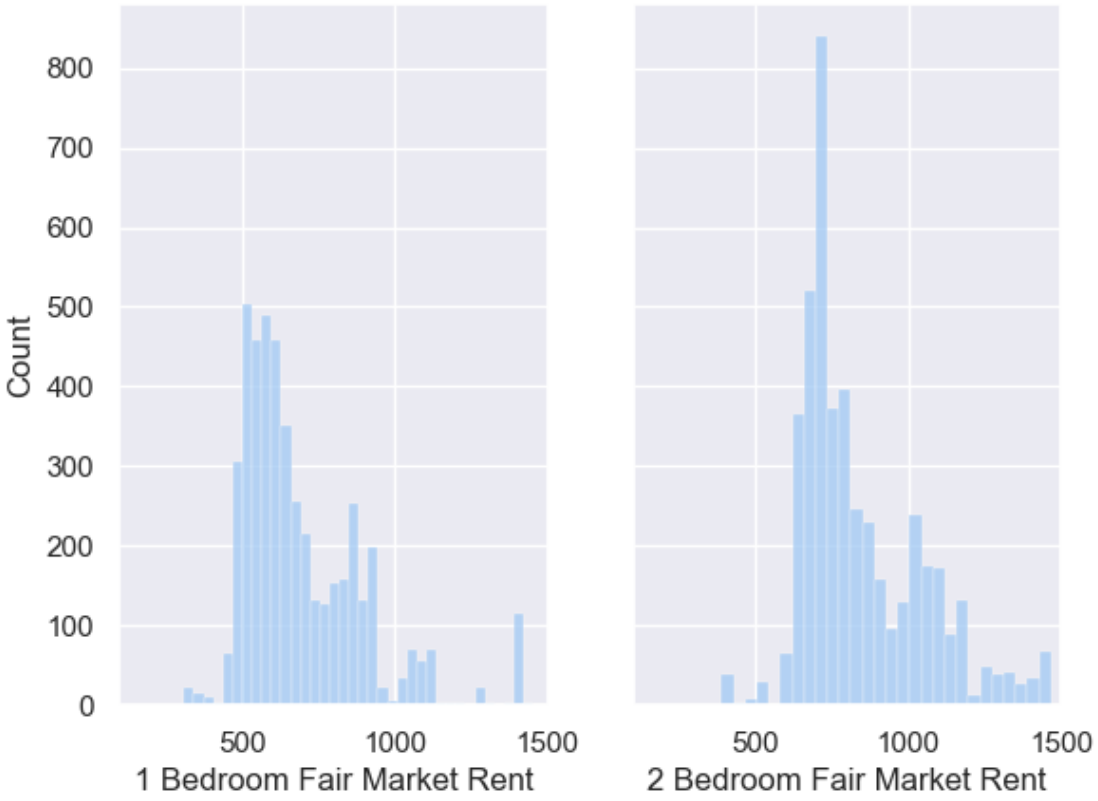
[111]:
```
# Create a figure and axes
fig, ax = plt.subplots()

# Plot the distribution of 1 bedroom rents
sns.histplot(fmr['fmr_1'], ax=ax)

# Modify the properties of the plot
ax.set(xlabel="1 Bedroom Fair Market Rent",
       xlim=(100,1500),
       title="US Rent")

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

[114]:
```
# Create a figure and axes. Then plot the data
fig, ax = plt.subplots()
sns.histplot(fmr['fmr_1'],
             ax = ax)

# Customize the labels and limits
ax.set(xlabel = "1 Bedroom Fair Market Rent",
       xlim = (100,1500),
       title = "US Rent")

# Add vertical lines for the median and mean
ax.axvline(x = fmr['fmr_1'].median(),
           color = 'm',
           label = 'Median',
           linestyle = '--',
           linewidth = 2)

ax.axvline(x = fmr['fmr_1'].mean(),
           color='b',
```

```
            label='Mean',
            linestyle='-',
            linewidth=2)

# Show the legend and plot the data
ax.legend()
plt.show()
```

[115]:
```
# Create a plot with 1 row and 2 columns that share the y axis label
fig, (ax0, ax1) = plt.subplots(nrows=1, ncols=2, sharey=True)

# Plot the distribution of 1 bedroom apartments on ax0
sns.histplot(fmr['fmr_1'], ax=ax0)
ax0.set(xlabel="1 Bedroom Fair Market Rent", xlim=(100,1500))

# Plot the distribution of 2 bedroom apartments on ax1
sns.histplot(fmr['fmr_2'], ax=ax1)
ax1.set(xlabel="2 Bedroom Fair Market Rent", xlim=(100,1500))
```

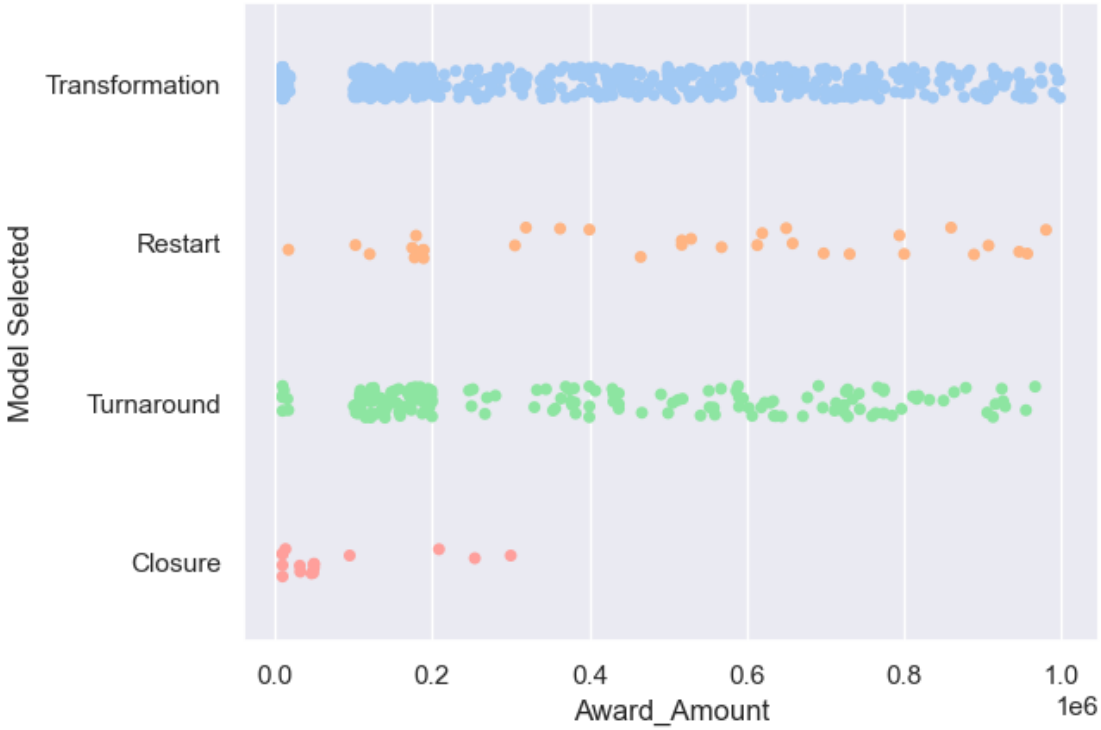
```
# Display the plot
plt.show()
```



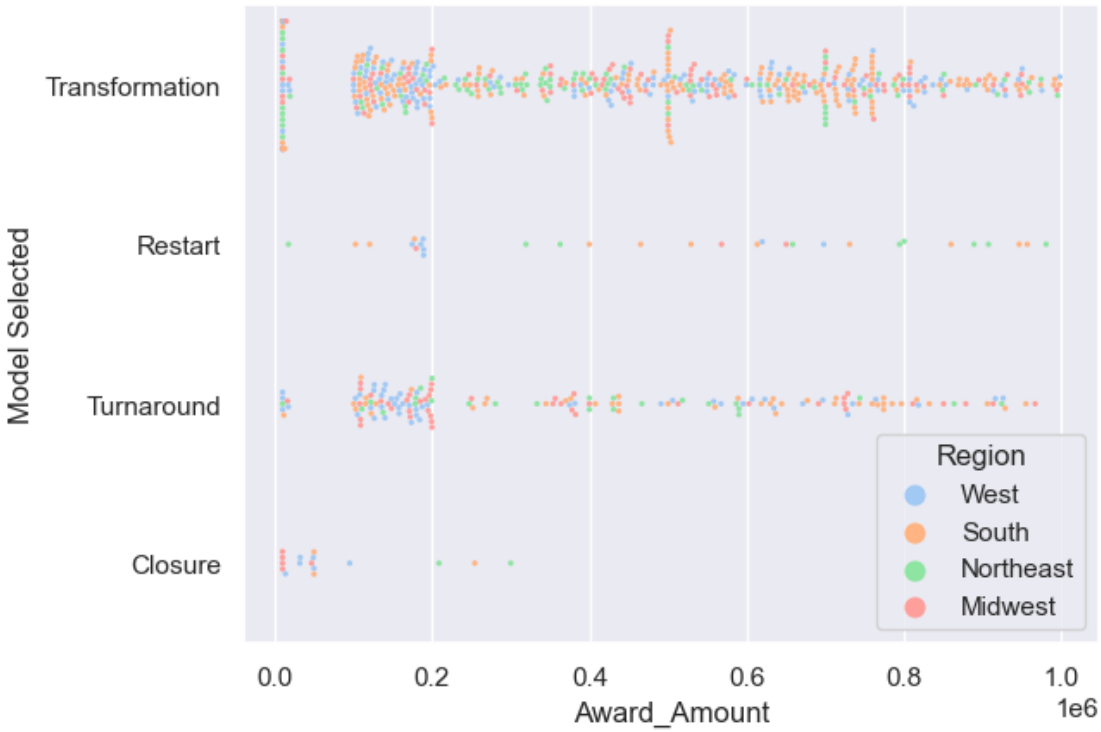
4.3 GRÁFICOS CATEGÓRICOS

4.3.1 Básicos

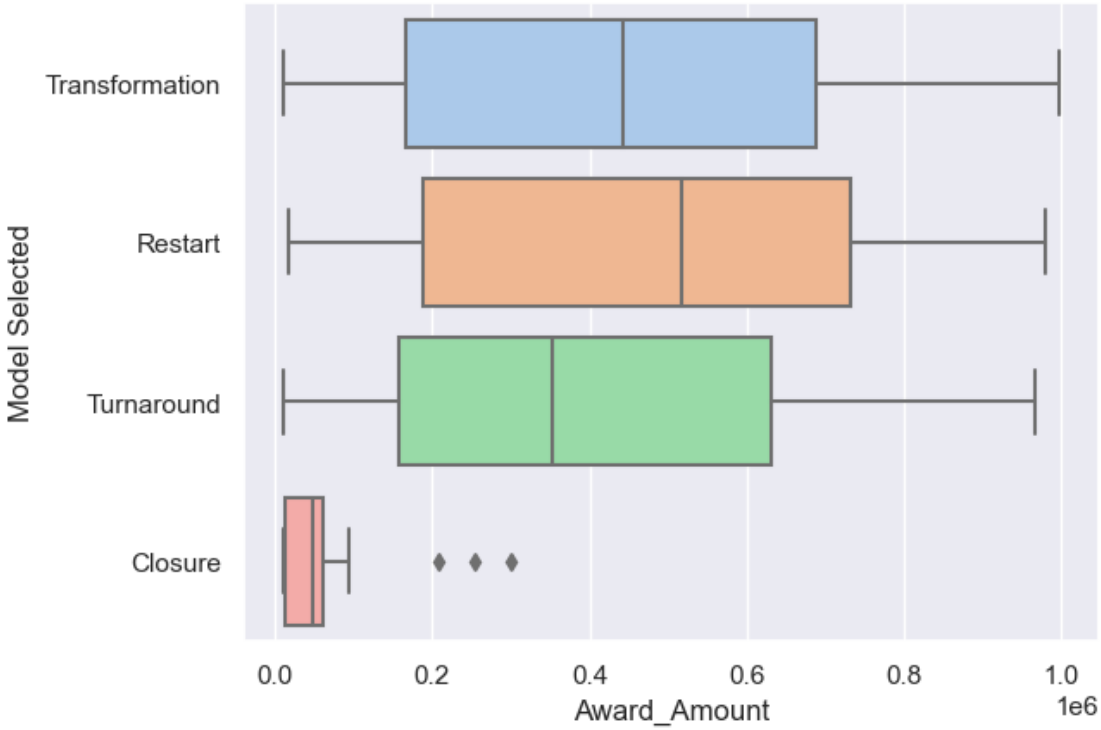
[116]:
```
# Create the stripplot
sns.stripplot(data = grants,
              x = 'Award_Amount',
              y = 'Model Selected',
              jitter = True)

plt.show()
```



[123]:
```
# Create and display a swarmplot with hue set to the Region
sns.swarmplot(data = grants,
              x = 'Award_Amount',
              y = 'Model Selected',
              hue = 'Region', size = 2.3)

plt.show()
```

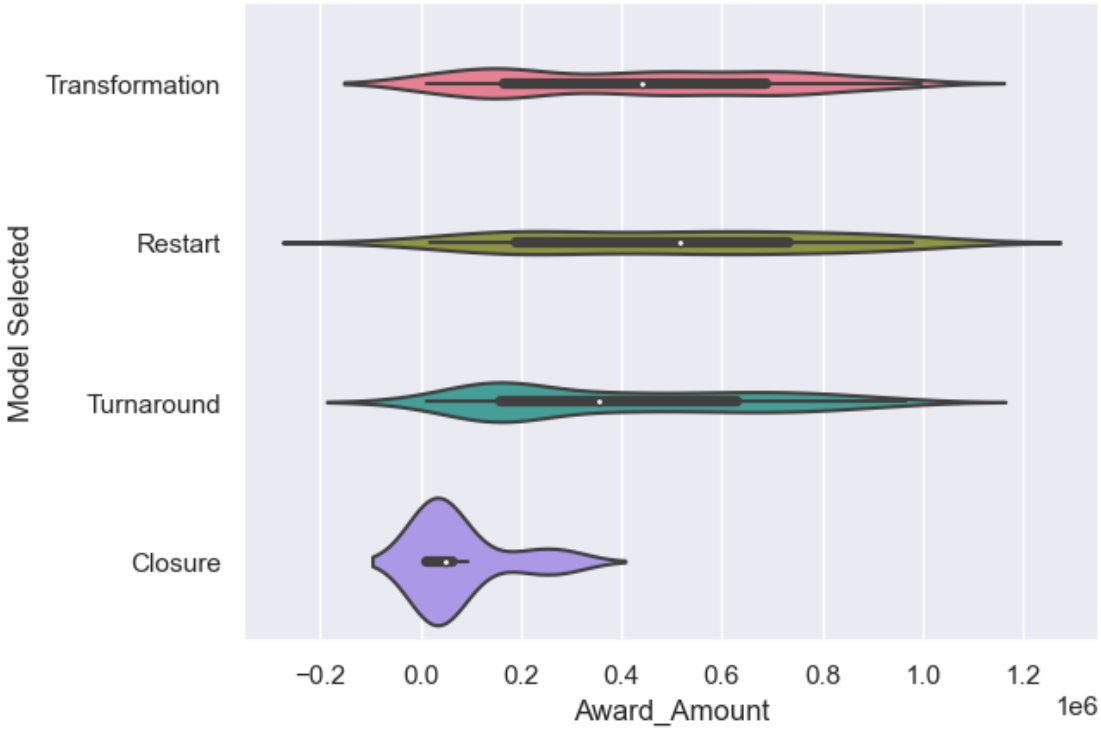


[124]:
```
# Create a boxplot
sns.boxplot(data = grants,
            x = 'Award_Amount',
            y = 'Model Selected')

plt.show()
plt.clf()
```



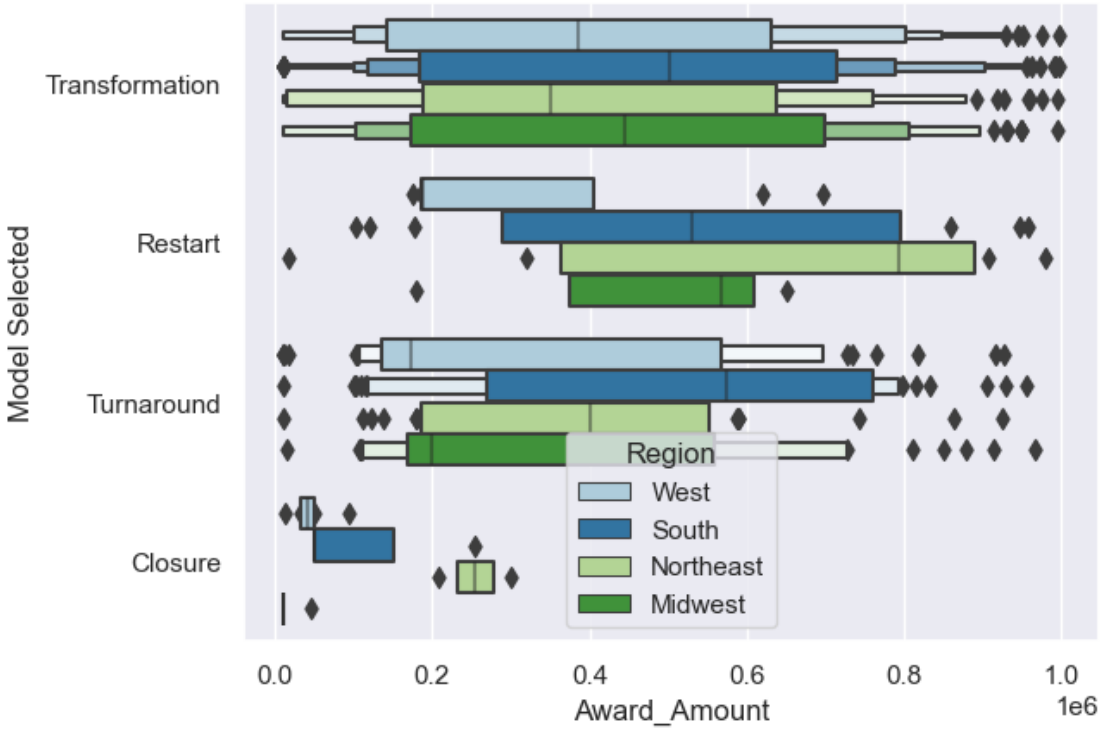

```
<Figure size 640x480 with 0 Axes>
```

[125]:
```
# Create a violinplot with the husl palette
sns.violinplot(data = grants,
               x = 'Award_Amount',
               y = 'Model Selected',
               palette = 'husl')

plt.show()
plt.clf()
```



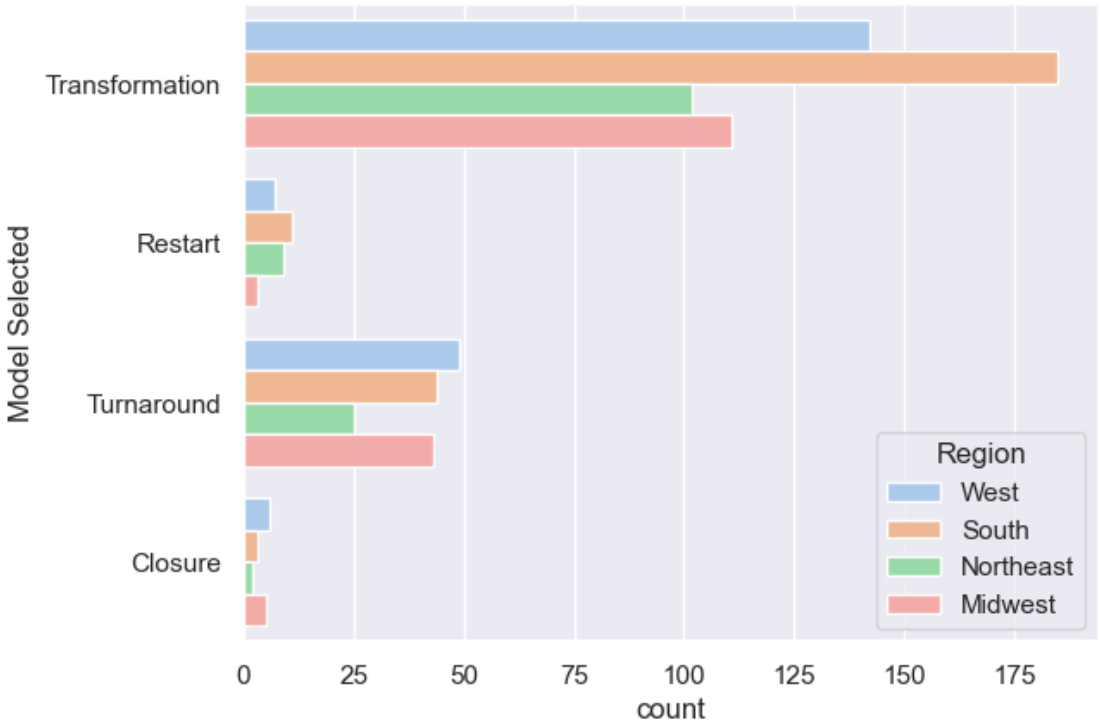

```
<Figure size 640x480 with 0 Axes>
```

[131]:
```
# Create a boxenplot with the Paired palette and the Region column as the hue
sns.boxenplot(data=grants,
              x='Award_Amount',
              y='Model Selected',
              palette='Paired',
              hue='Region')

plt.show()
plt.clf()
```



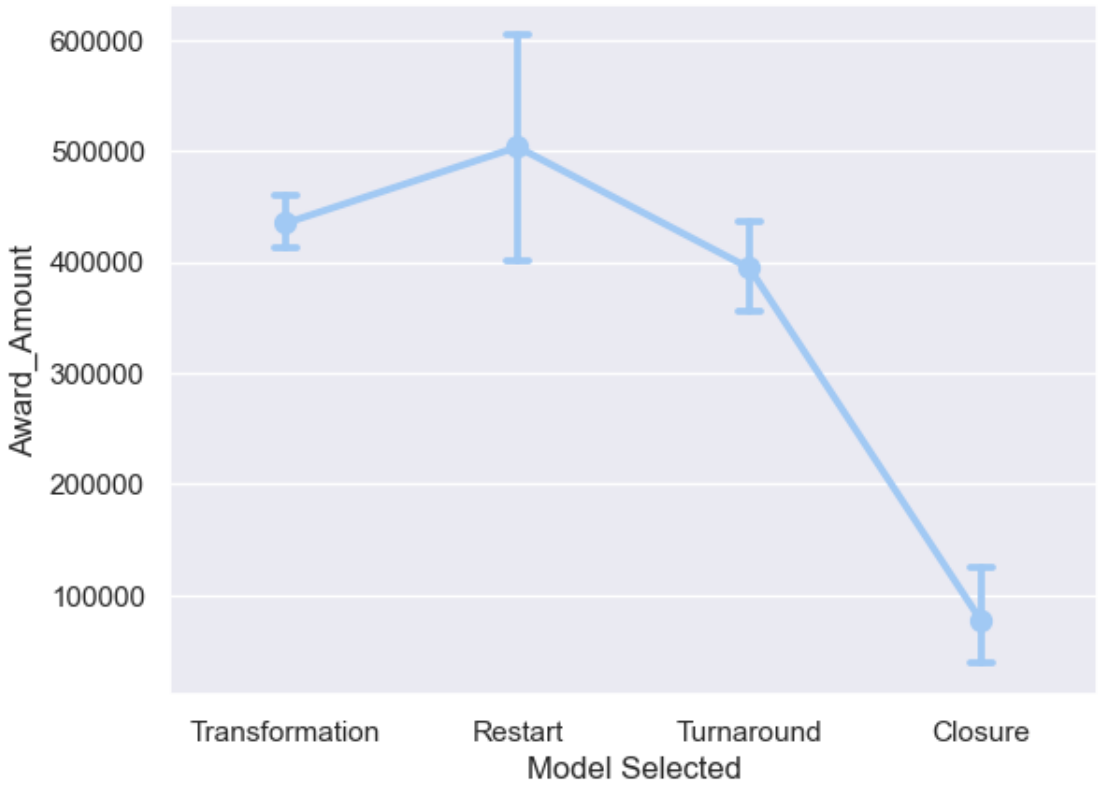

```
<Figure size 640x480 with 0 Axes>
```

[127]:
```
# Show a countplot with the number of models used with each region a different␣
↪color
sns.countplot(data = grants,
              y = "Model Selected",
              hue = "Region")

plt.show()
plt.clf()
```



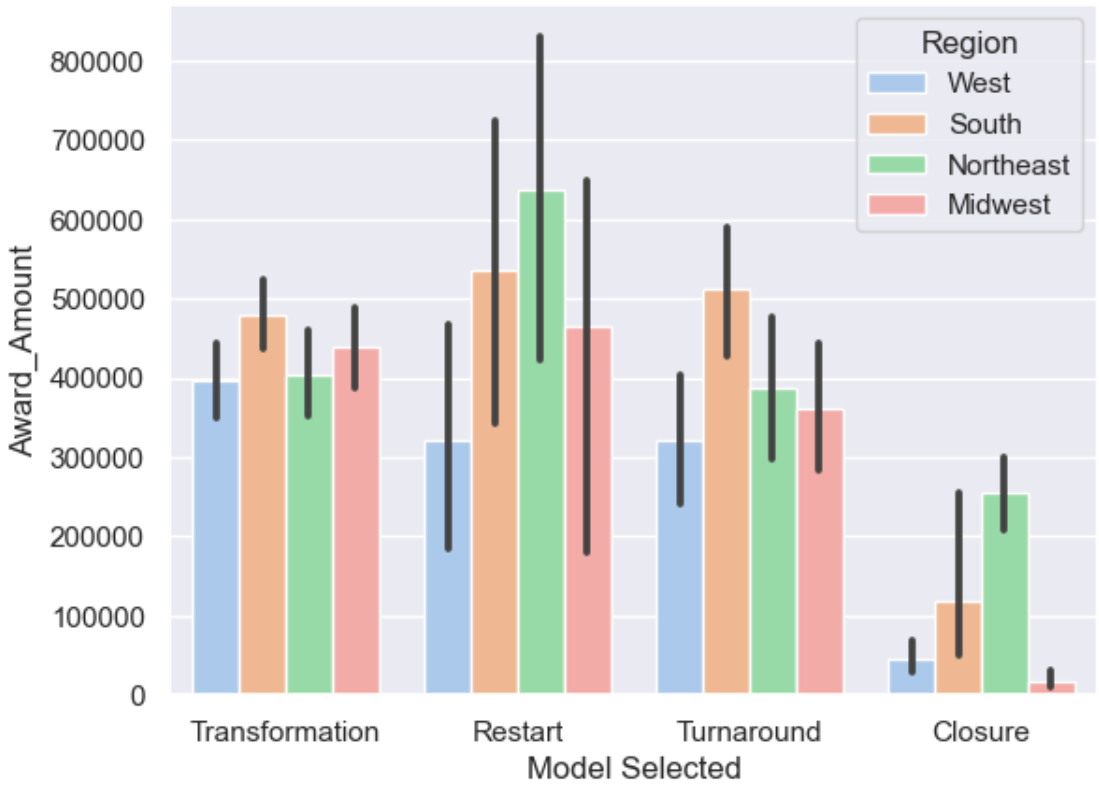

```
<Figure size 640x480 with 0 Axes>
```

[129]:
```
# Create a pointplot and include the capsize in order to show bars on the␣
 ↪confidence interval
sns.pointplot(data = grants,
              y = 'Award_Amount',
              x = 'Model Selected',
              capsize = .1)

plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

[130]:
```
# Create a barplot with each Region shown as a different color
sns.barplot(data = grants,
            y = 'Award_Amount',
            x = 'Model Selected',
            hue = 'Region')

plt.show()
plt.clf()
```



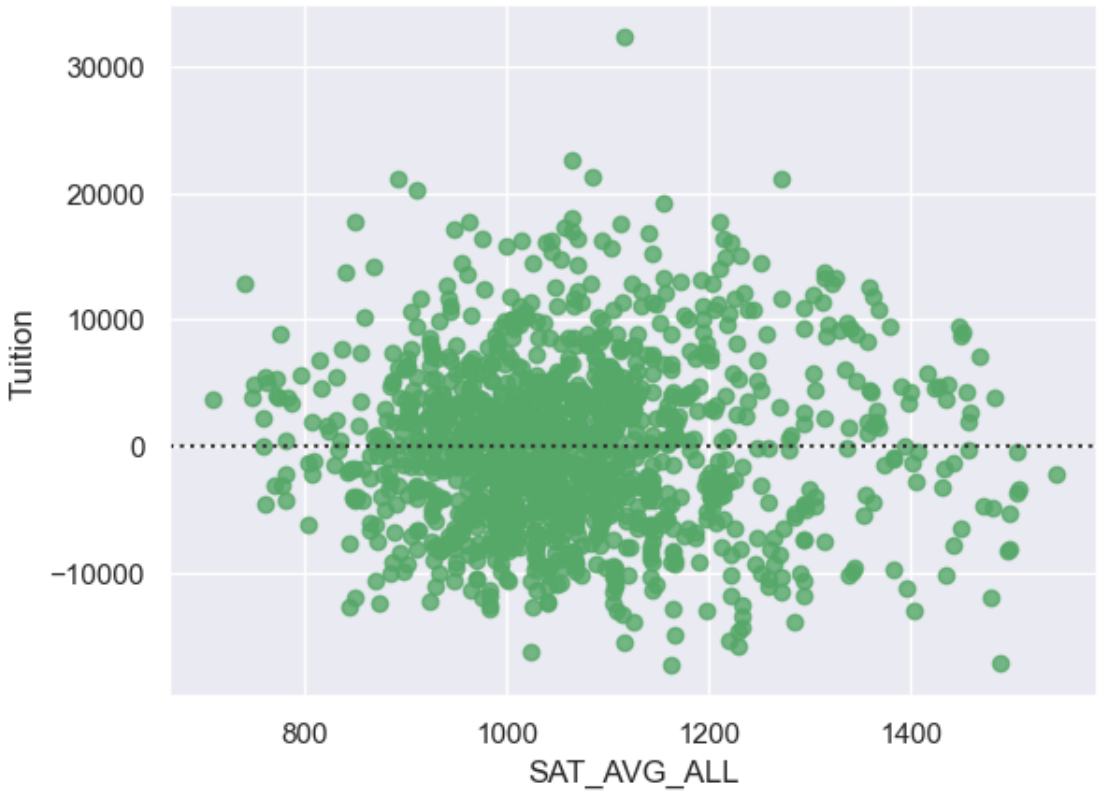

```
<Figure size 640x480 with 0 Axes>
```

4.3.2 Regression plots

[132]:
```
# Display a regression plot for Tuition
sns.regplot(data = college,
            y = 'Tuition',
            x = "SAT_AVG_ALL",
            marker = '^',
            color = 'g')

plt.show()
plt.clf()
```

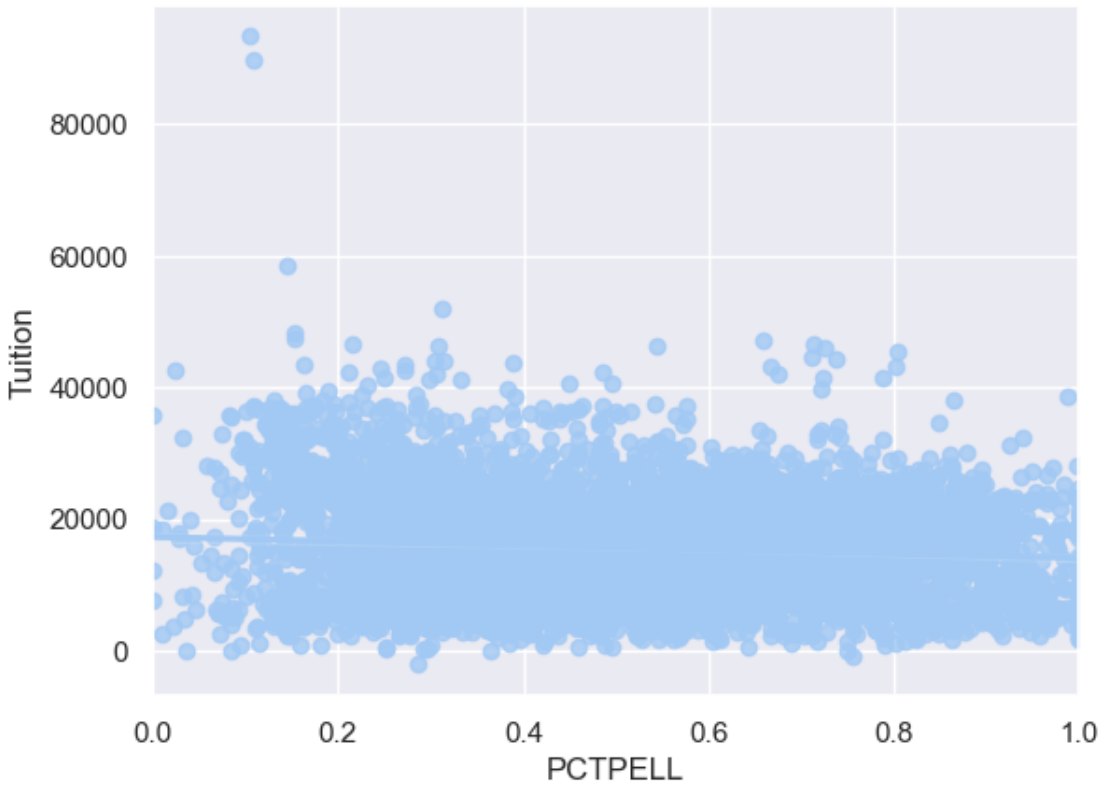

```
<Figure size 640x480 with 0 Axes>
```

[133]:
```
# Display the residual plot
sns.residplot(data = college,
              y = 'Tuition',
              x = "SAT_AVG_ALL",
              color = 'g')

plt.show()
plt.clf()
```



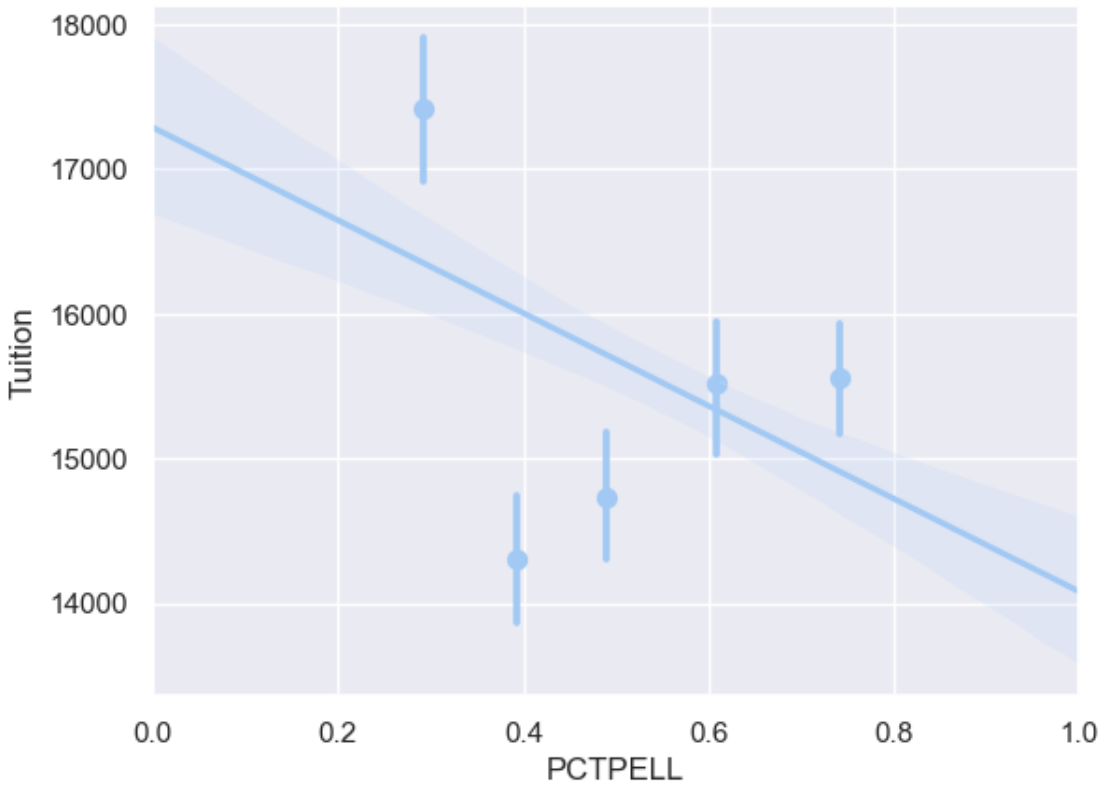

```
<Figure size 640x480 with 0 Axes>
```

[134]:
```
# Plot a regression plot of Tuition and the Percentage of Pell Grants
sns.regplot(data = college,
            y = 'Tuition',
            x = "PCTPELL")

plt.show()
plt.clf()
```



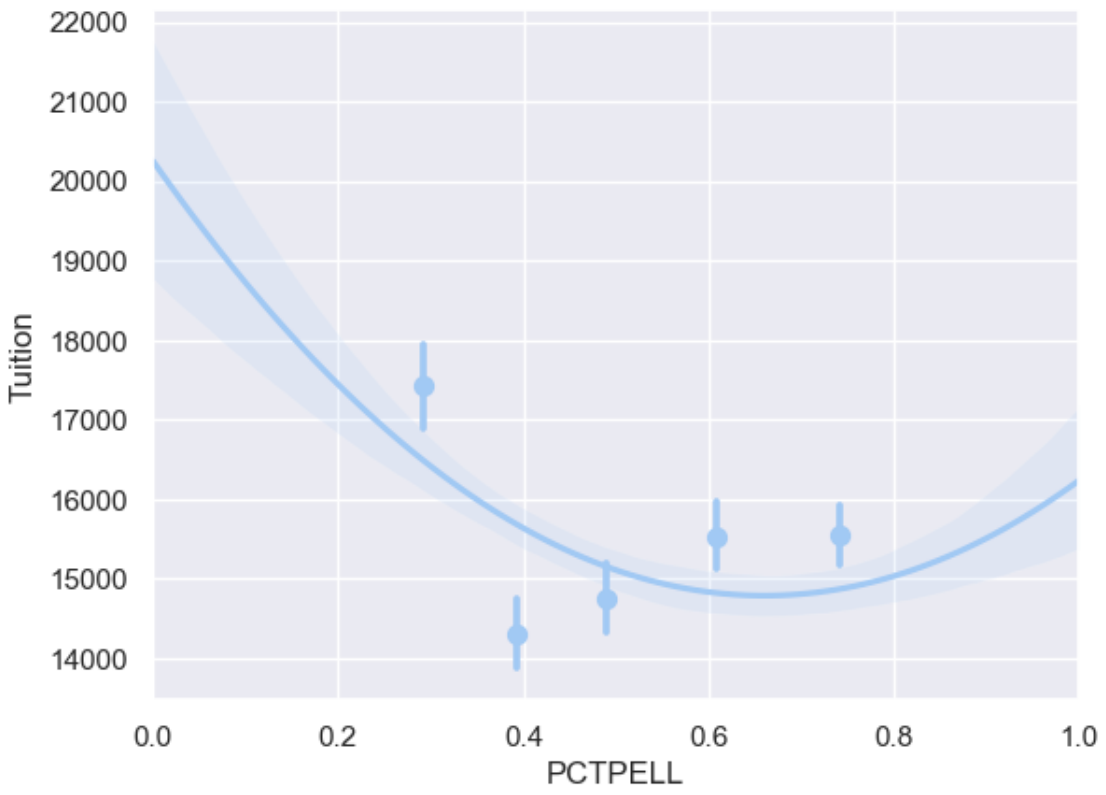

```
<Figure size 640x480 with 0 Axes>
```

[135]:
```
# Create another plot that estimates the tuition by PCTPELL
sns.regplot(data = college,
            y = 'Tuition',
            x = "PCTPELL",
            x_bins = 5)

plt.show()
plt.clf()
```



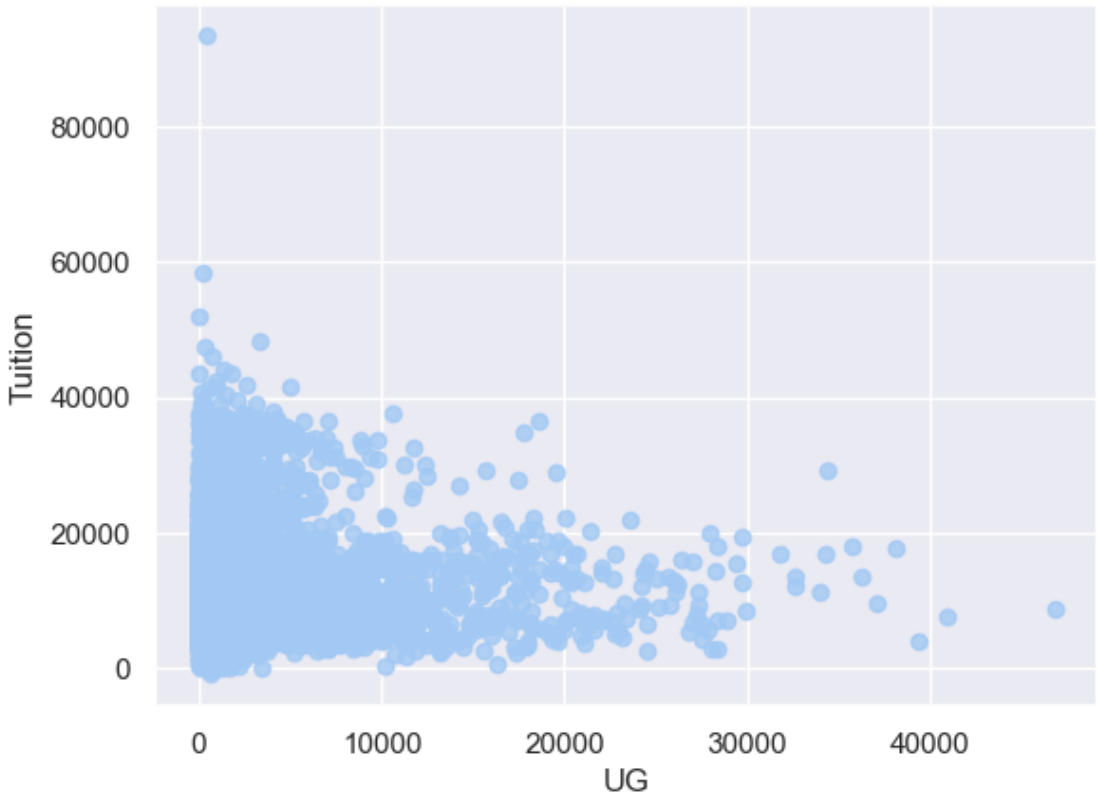

```
<Figure size 640x480 with 0 Axes>
```

[136]:
```
# The final plot should include a line using a 2nd order polynomial
sns.regplot(data = college,
            y = 'Tuition',
            x = "PCTPELL",
            x_bins = 5,
            order = 2)

plt.show()
plt.clf()
```



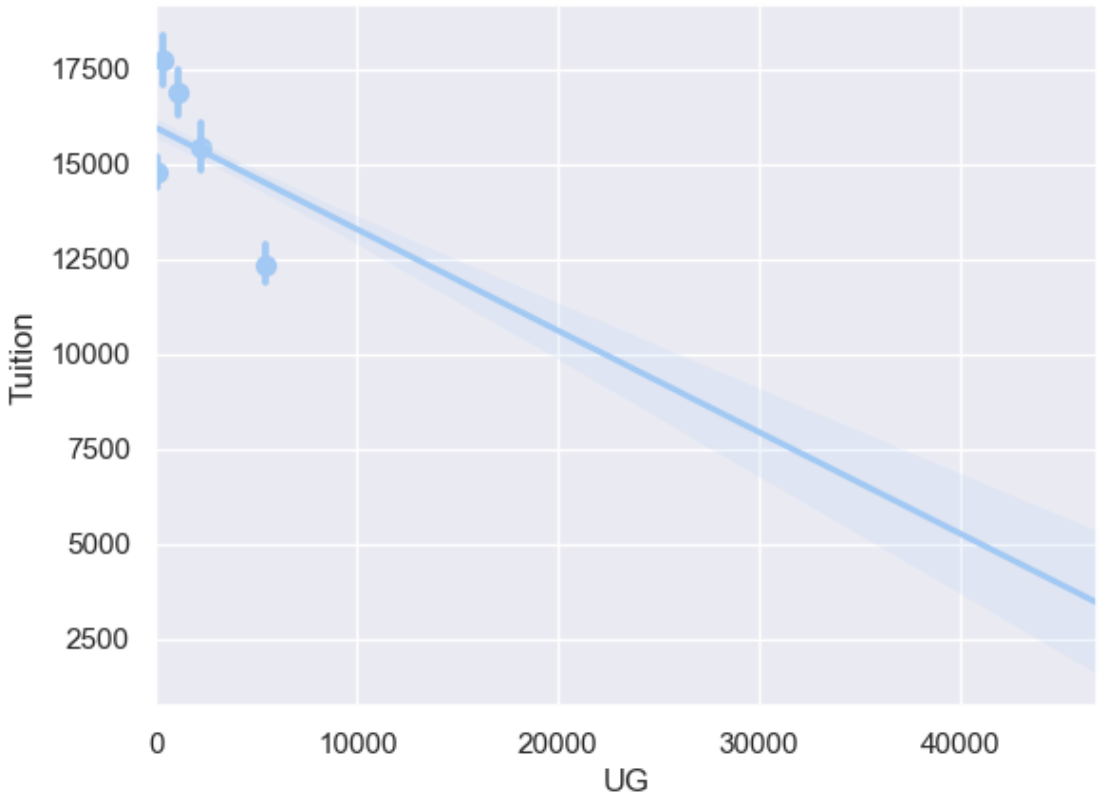

```
<Figure size 640x480 with 0 Axes>
```

[137]:
```
# Create a scatter plot by disabling the regression line
sns.regplot(data = college,
            y = 'Tuition',
            x = "UG",
            fit_reg = False)

plt.show()
plt.clf()
```



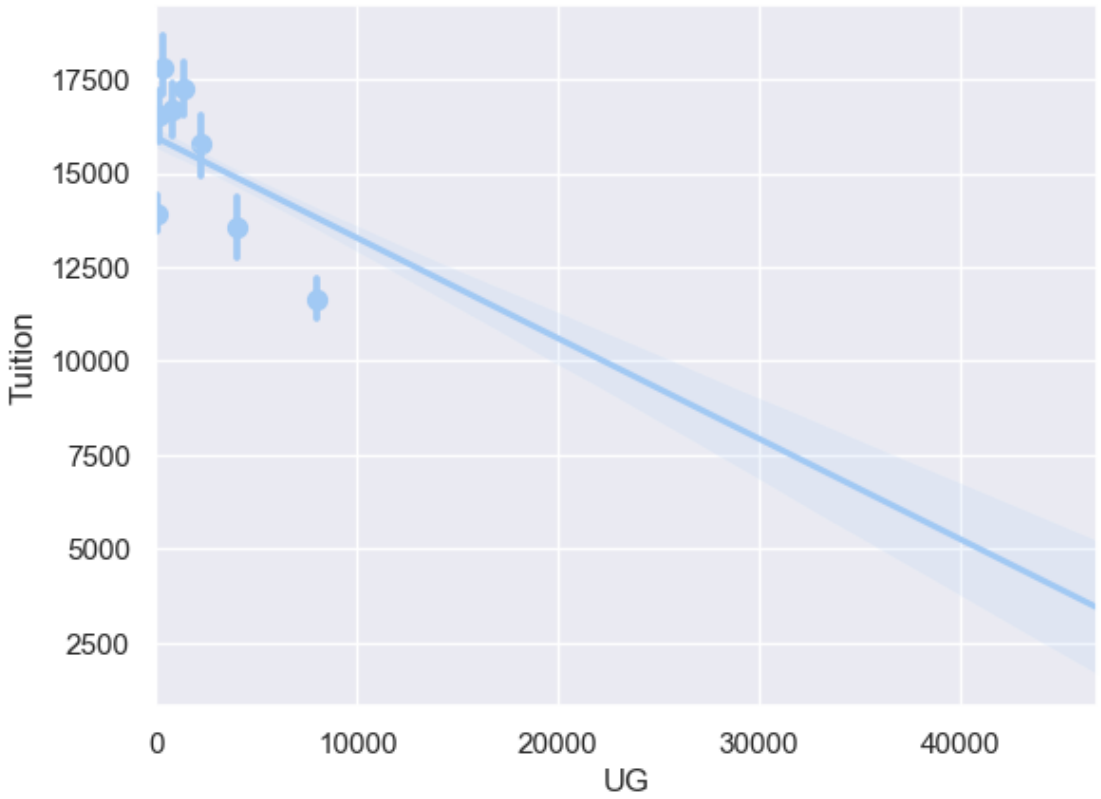

```
<Figure size 640x480 with 0 Axes>
```

[138]:
```
# Create a scatter plot and bin the data into 5 bins
sns.regplot(data = college,
            y = 'Tuition',
            x = "UG",
            x_bins = 5)

plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

[139]:
```
# Create a regplot and bin the data into 8 bins
sns.regplot(data = college,
            y = 'Tuition',
            x = "UG",
            x_bins = 8)

plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

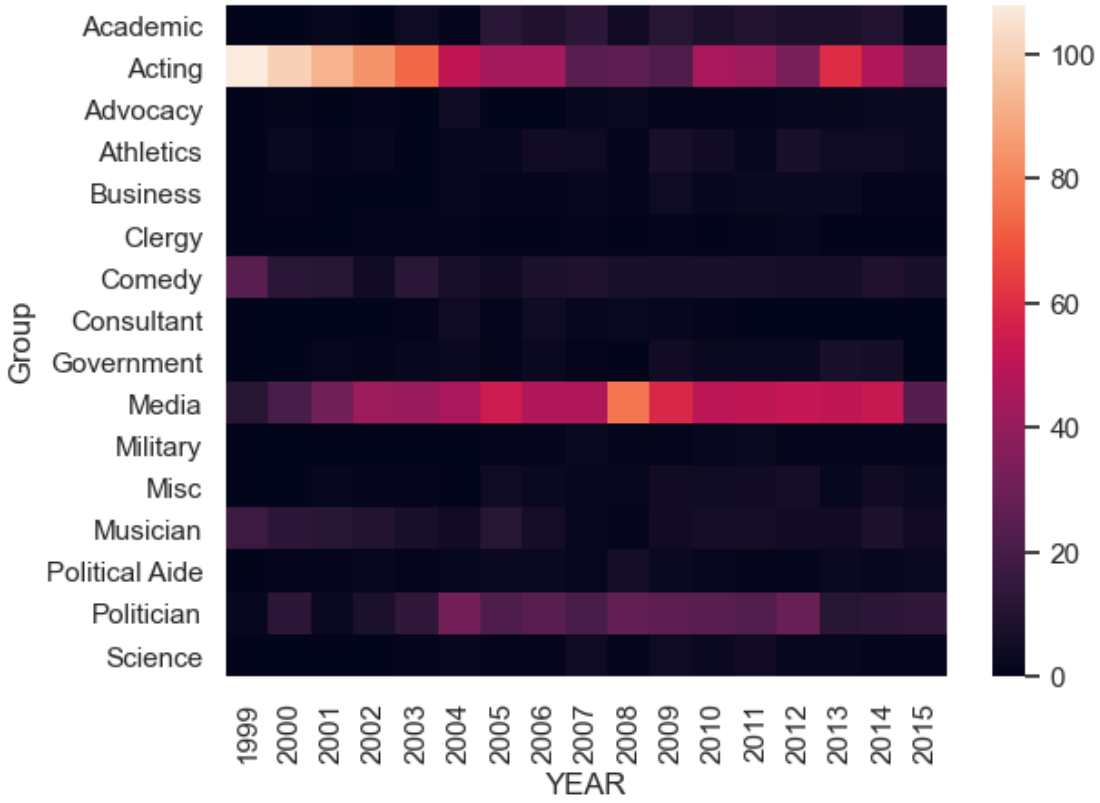
4.3.3 Matrix plots

[140]:
```
show = pd.read_csv("C:/Users/marco/Data Camp Python/Datasets/
 ↪daily_show_guests_cleaned.csv")

# Create a crosstab table of the data
pd_crosstab = pd.crosstab(show["Group"], show["YEAR"])
print(pd_crosstab)

# Plot a heatmap of the table
sns.heatmap(pd_crosstab)

# Rotate tick marks for visibility
plt.yticks(rotation = 0)
plt.xticks(rotation = 90)

plt.show()
```

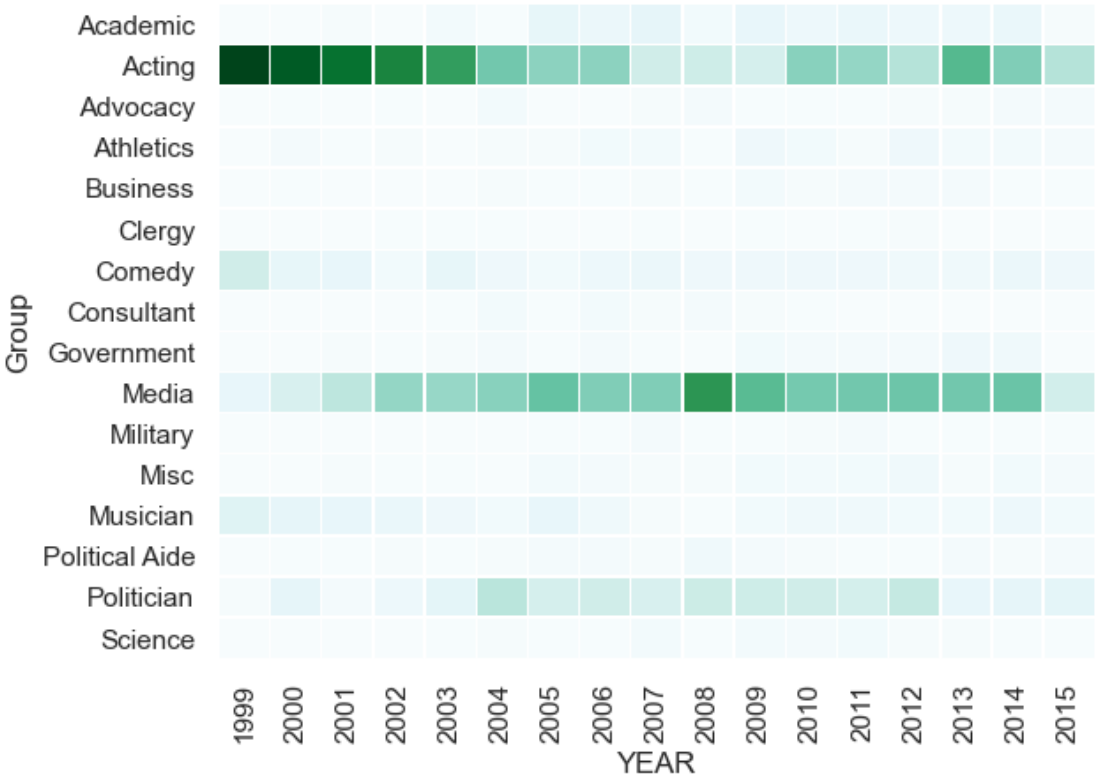
```
YEAR            1999  2000  2001  2002  2003  2004  2005  2006  2007  2008  \
Group
```

Academic	0	0	2	0	4	1	12	9	13	5
Acting	108	100	92	84	74	51	44	44	25	26
Advocacy	0	1	0	1	0	4	0	0	2	3
Athletics	0	3	1	2	0	2	2	5	4	1
Business	0	1	0	0	0	2	1	1	2	1
Clergy	0	0	0	1	1	1	0	0	1	0
Comedy	25	12	11	5	12	7	5	8	9	7
Consultant	0	0	0	0	1	4	1	4	2	3
Government	0	0	2	1	2	3	1	3	1	0
Media	11	21	31	42	41	45	54	47	47	77
Military	0	0	0	0	0	0	1	1	3	1
Misc	0	0	2	1	1	0	4	3	2	2
Musician	17	13	11	10	7	5	11	6	2	1
Political Aide	0	1	1	2	1	2	3	3	2	6
Politician	2	13	3	8	14	32	22	25	21	27
Science	0	0	0	0	1	2	1	1	4	1

YEAR	2009	2010	2011	2012	2013	2014	2015
Group							
Academic	11	8	10	8	8	10	2
Acting	22	45	42	33	60	47	33
Advocacy	1	1	1	2	2	3	3
Athletics	7	5	2	7	4	4	3
Business	4	2	3	3	3	1	1
Clergy	1	0	1	2	0	0	0
Comedy	7	7	7	6	6	9	7
Consultant	2	1	0	0	0	0	0
Government	5	3	3	3	7	6	0
Media	59	50	51	52	51	53	24
Military	1	2	3	1	1	1	1
Misc	5	4	5	6	2	5	3
Musician	5	6	6	5	5	8	5
Political Aide	3	2	1	1	3	2	3
Politician	26	25	23	29	11	13	14
Science	4	3	5	2	2	1	1



[141]:
```
# Create the crosstab DataFrame
pd_crosstab = pd.crosstab(show["Group"], show["YEAR"])

# Plot a heatmap of the table with no color bar and using the BuGn palette
sns.heatmap(pd_crosstab,
            cbar = False,
            cmap = "BuGn",
            linewidths = 0.3)

# Rotate tick marks for visibility
plt.yticks(rotation = 0)
plt.xticks(rotation = 90)

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

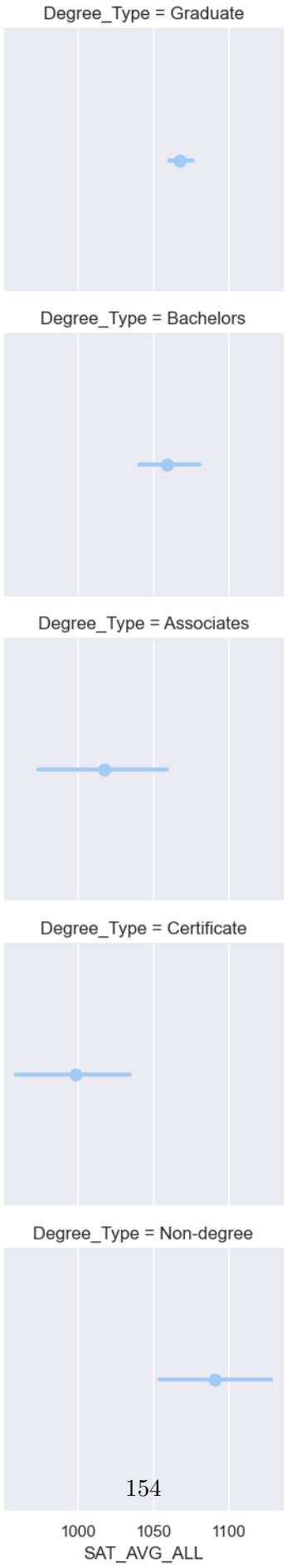



```
<Figure size 640x480 with 0 Axes>
```

4.4 GRÁFICOS MÚLTIPLES

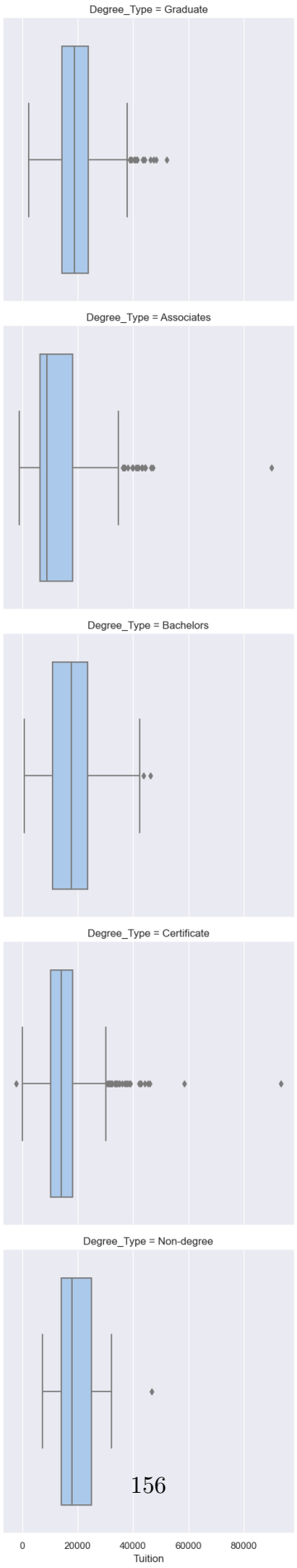
4.4.1 FacetGrid, catplot e lmplot

[145]:
```
# Create FacetGrid with Degree_Type and specify the order of the rows using␣
 ↪row_order
g2 = sns.FacetGrid(college,
                   row="Degree_Type",
                   row_order=['Graduate',
                              'Bachelors',
                              'Associates',
                              'Certificate',
                              'Non-degree'])

# Map a pointplot of SAT_AVG_ALL onto the grid
g2.map(sns.pointplot, 'SAT_AVG_ALL', order = None)

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


Degree_Type = Graduate
Degree_Type = Bachelors
Degree_Type = Associates
Degree_Type = Certificate
Degree_Type = Non-degree
1000
1050
1100
SAT_AVG_ALL

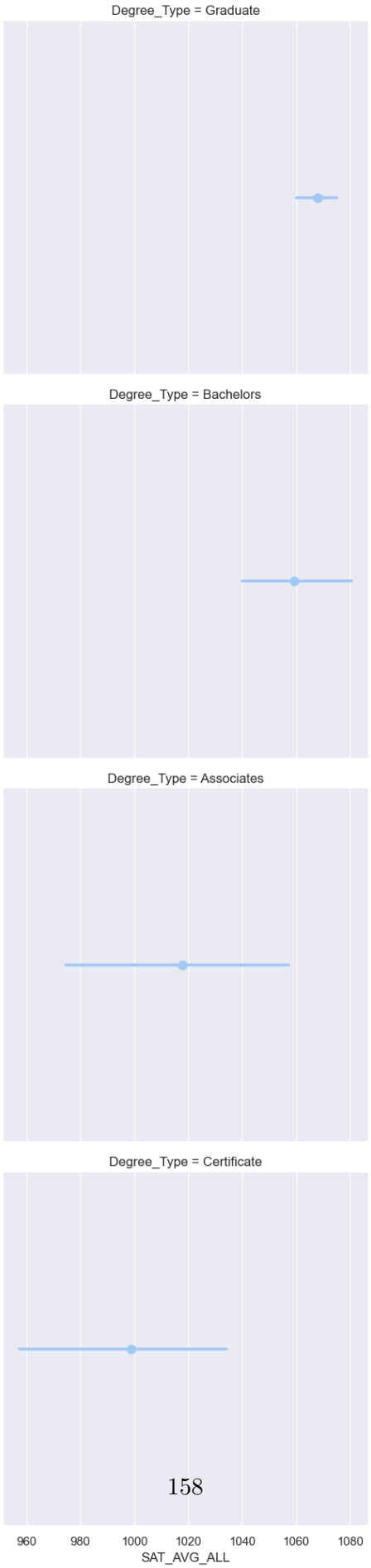
```
<Figure size 640x480 with 0 Axes>
```

[147]:
```
# Create a factor plot that contains boxplots of Tuition values
sns.catplot(data=college,
            x='Tuition',
            kind='box',
            row='Degree_Type')

plt.show()
plt.clf()
```


Degree_Type = Graduate
Degree_Type = Associates
Degree_Type = Bachelors
Degree_Type = Certificate
Degree_Type = Non-degree
0
20000
40000
60000
80000
Tuition

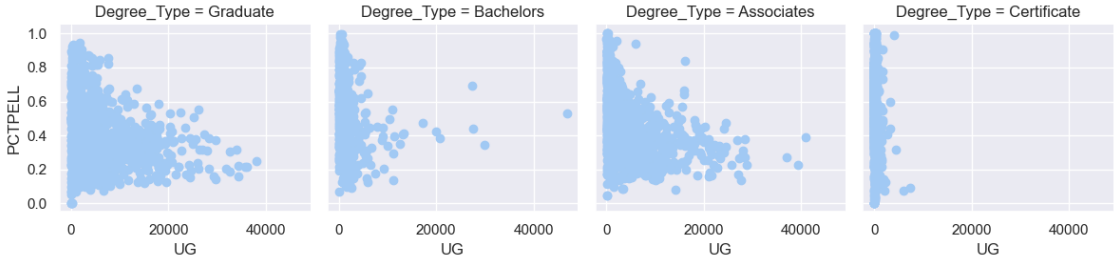
```
<Figure size 640x480 with 0 Axes>
```

[148]:
```
# Create a facetted pointplot of Average SAT_AVG_ALL scores facetted by Degree␣
↪Type
sns.catplot(data=college,
        x='SAT_AVG_ALL',
        kind='point',
        row='Degree_Type',
        row_order=['Graduate', 'Bachelors', 'Associates', 'Certificate'])

plt.show()
plt.clf()
```


Degree_Type = Graduate
Degree_Type = Bachelors
Degree_Type = Associates
Degree_Type = Certificate
960
980
1000
1020
1040
1060
1080
SAT_AVG_ALL

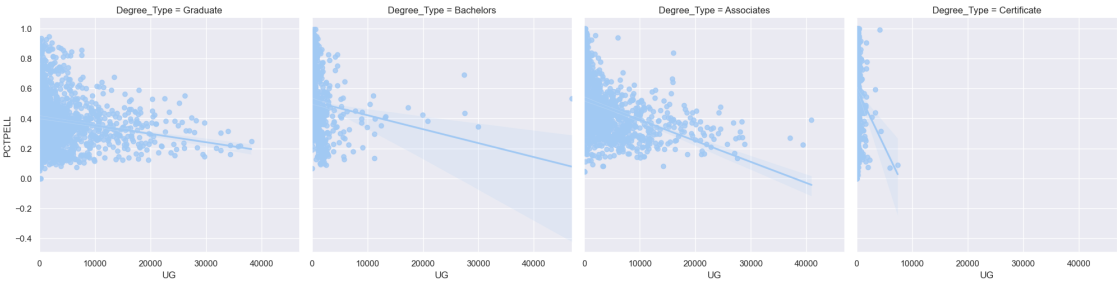
```
<Figure size 640x480 with 0 Axes>
```

[154]:
```
# Create a FacetGrid varying by column and columns ordered with the␣
 ↪degree_order variable
degree_ord = ['Graduate', 'Bachelors', 'Associates', 'Certificate']

g = sns.FacetGrid(college, col="Degree_Type", col_order=degree_ord)

# Map a scatter plot of Undergrad Population compared to PCTPELL
g.map(plt.scatter, 'UG', 'PCTPELL')

plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

[155]:
```
# Re-create the previous plot as an lmplot
sns.lmplot(data=college,
           x='UG',
           y='PCTPELL',
           col="Degree_Type",
           col_order=degree_ord)

plt.show()
plt.clf()
```



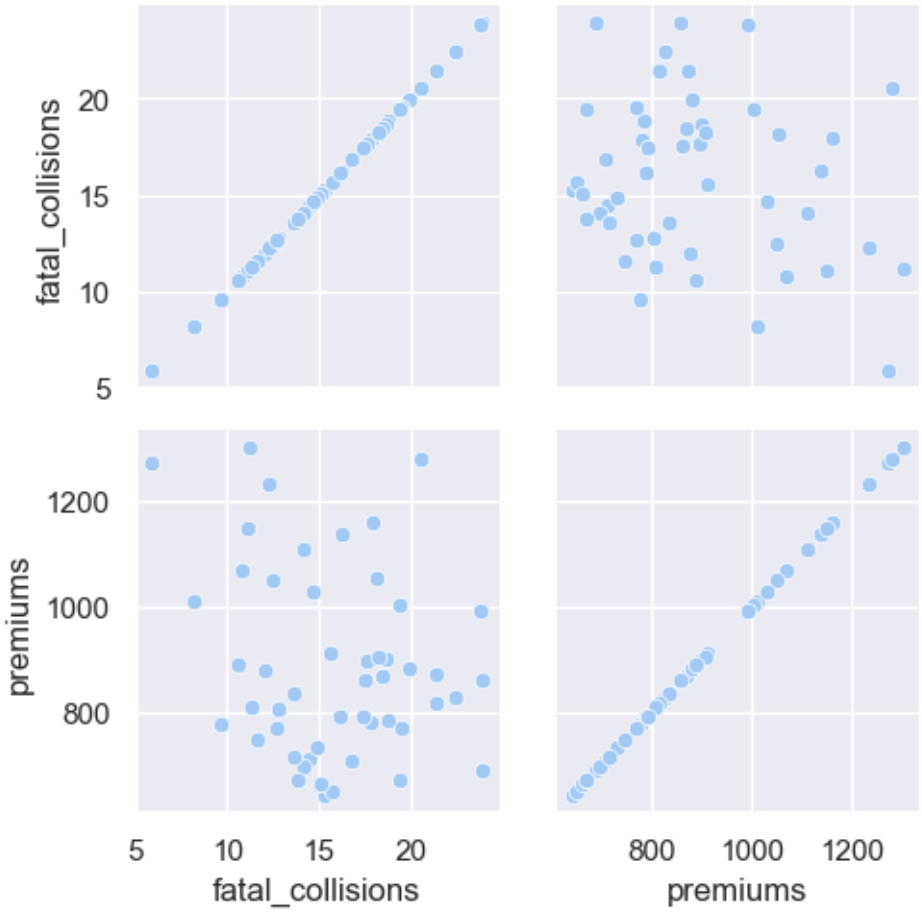
```
<Figure size 640x480 with 0 Axes>
```

[157]:
```
# Create an lmplot that has a column for Ownership, a row for Degree_Type and␣
 ↪hue based on the WOMENONLY column
sns.lmplot(data=college,
           x='SAT_AVG_ALL',
           y='Tuition',
           col="Ownership",
           row='Degree_Type',
           row_order=['Graduate', 'Bachelors'],
           hue='WOMENONLY',
           col_order=['Public', 'Private non-profit'])

plt.show()
plt.clf()
```

```
<Figure size 640x480 with 0 Axes>
```

4.4.2 PairGrid y pairplot

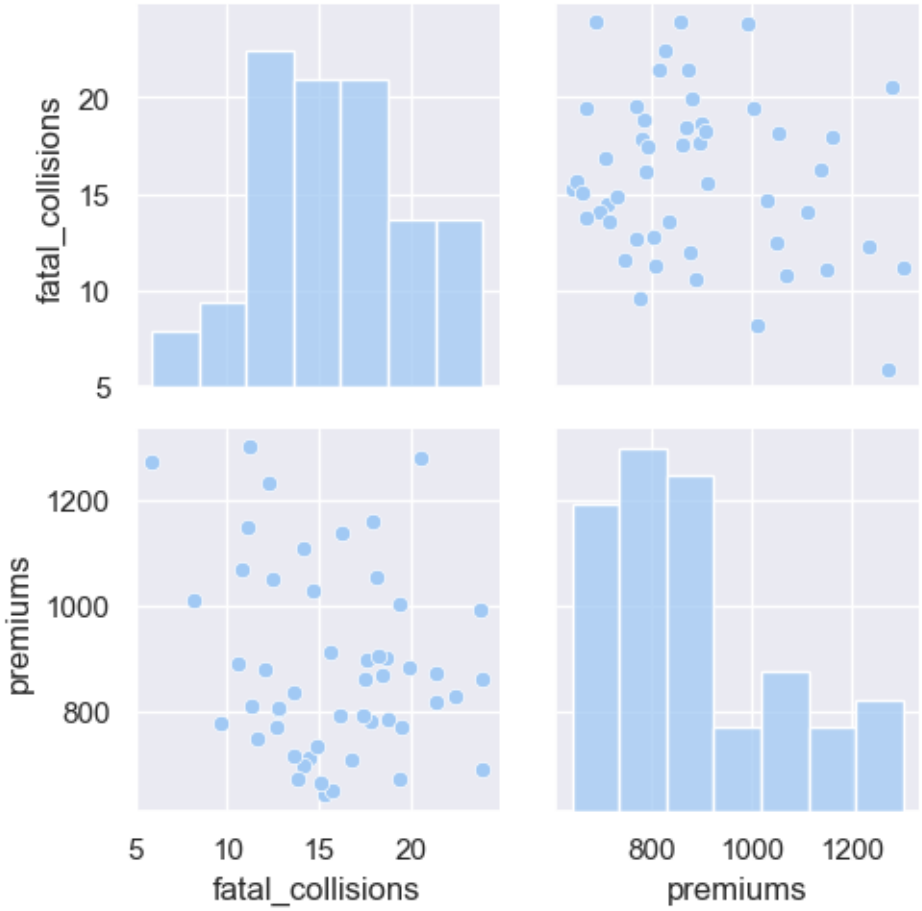
[163]:
```
# Create a PairGrid with a scatter plot for fatal_collisions and premiums
g = sns.PairGrid(insurance,
                 vars=["fatal_collisions", "premiums"])

g2 = g.map(sns.scatterplot)

plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

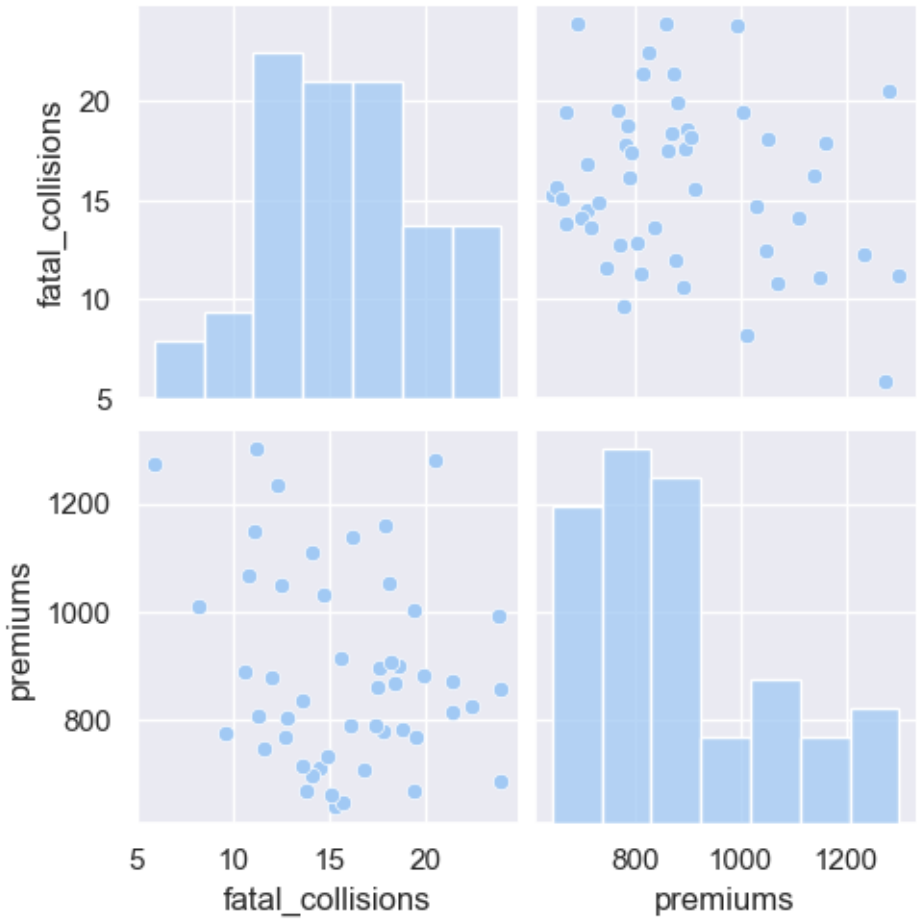
[164]:
```
# Create the same PairGrid but map a histogram on the diag
g = sns.PairGrid(insurance, vars=["fatal_collisions", "premiums"])
```

```
g2 = g.map_diag(sns.histplot)
g3 = g2.map_offdiag(sns.scatterplot)

plt.show()
plt.clf()
```



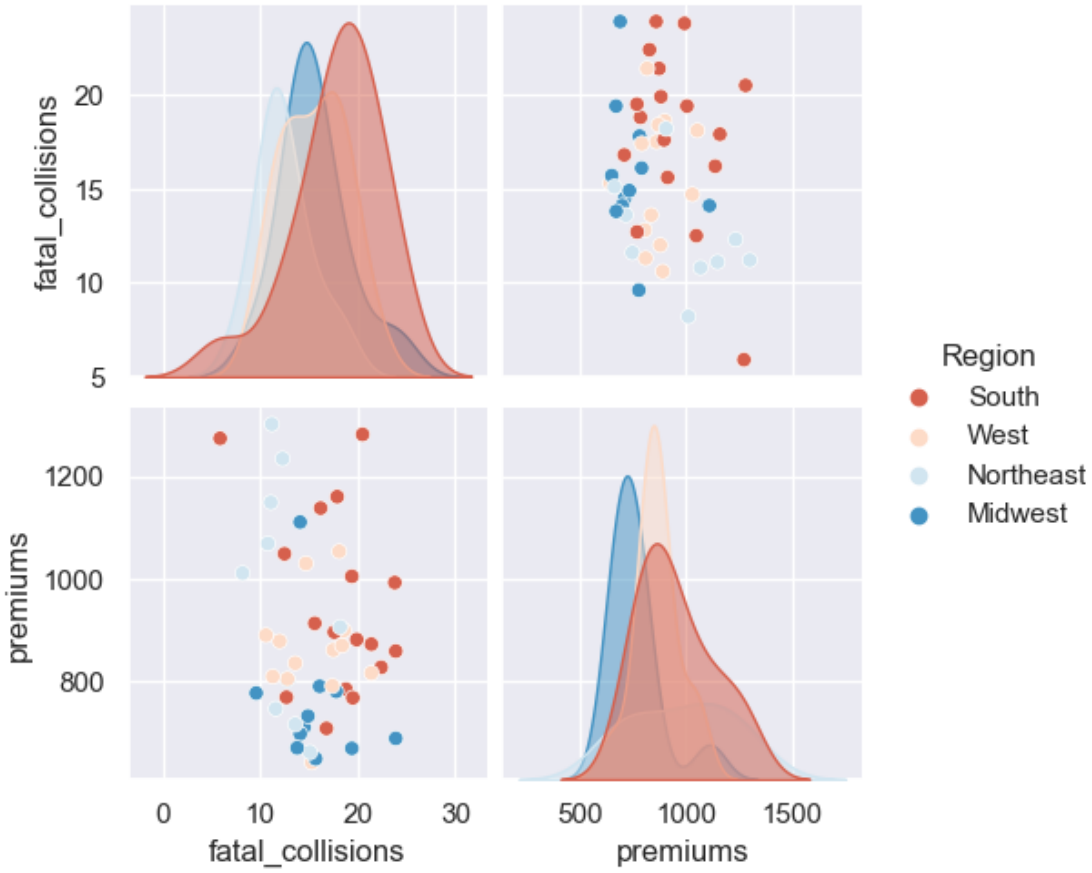

```
<Figure size 640x480 with 0 Axes>
```

[160]:
```
# Create a pairwise plot of the variables using a scatter plot
sns.pairplot(data=insurance,
             vars=["fatal_collisions", "premiums"],
             kind='scatter')

plt.show()
plt.clf()
```



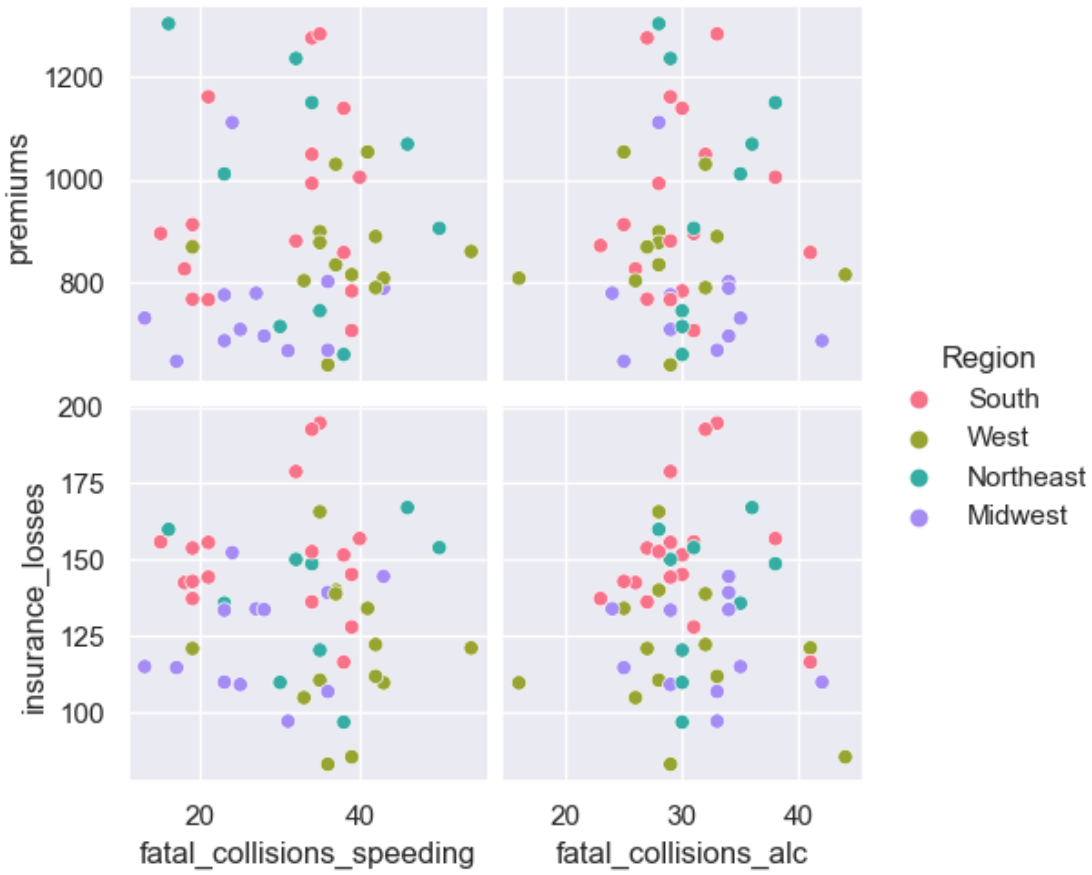

```
<Figure size 640x480 with 0 Axes>
```

[165]:
```
# Plot the same data but use a different color palette and color code by Region
sns.pairplot(data=insurance,
             vars=["fatal_collisions", "premiums"],
             kind='scatter',
             hue='Region',
             palette='RdBu',
             diag_kws={'alpha':.5})

plt.show()
plt.clf()
```



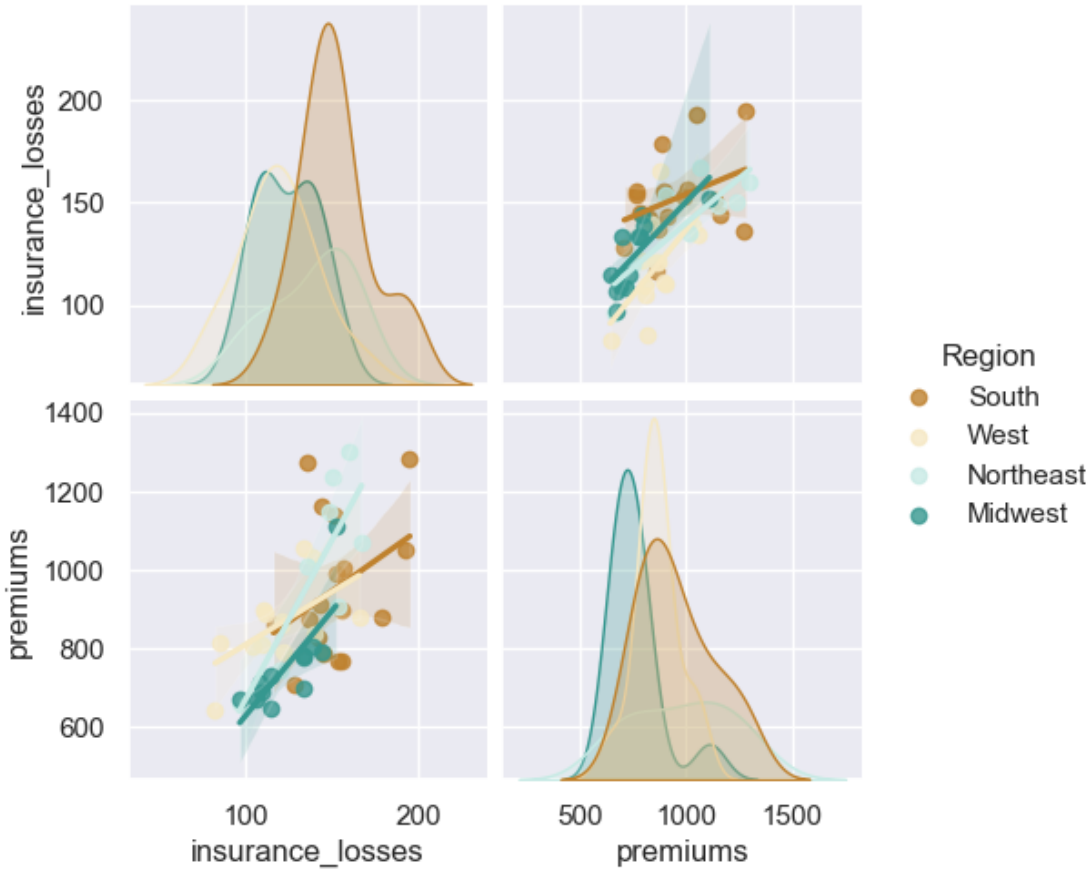

```
<Figure size 640x480 with 0 Axes>
```

[166]:
```
# Build a pairplot with different x and y variables
sns.pairplot(data=insurance,
             x_vars=["fatal_collisions_speeding", "fatal_collisions_alc"],
             y_vars=['premiums', 'insurance_losses'],
             kind='scatter',
             hue='Region',
             palette='husl')

plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

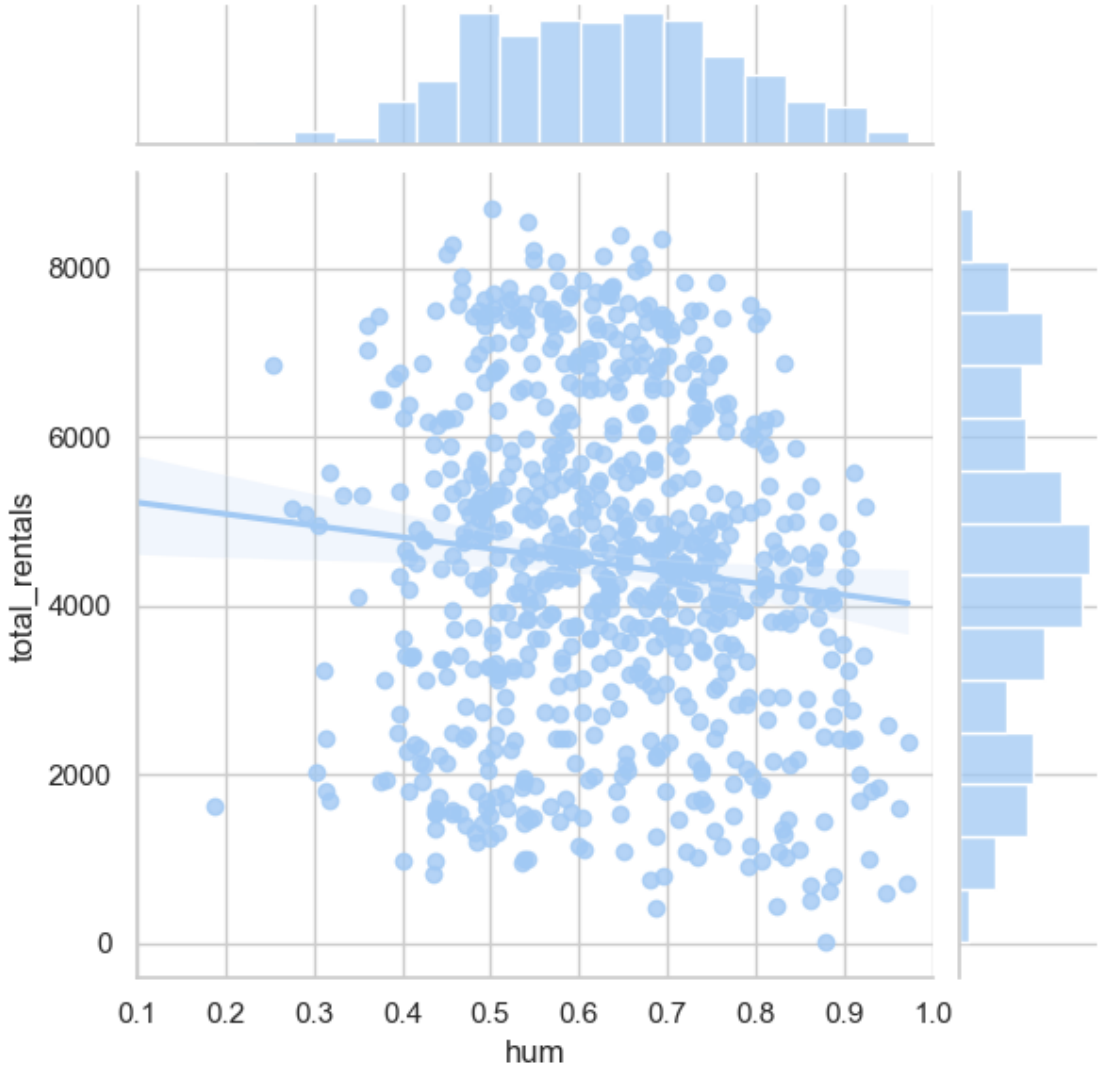
[167]:
```
# plot relationships between insurance_losses and premiums
sns.pairplot(data=insurance,
             vars=["insurance_losses", "premiums"],
             kind='reg',
             palette='BrBG',
             diag_kind = 'kde',
             hue='Region')

plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

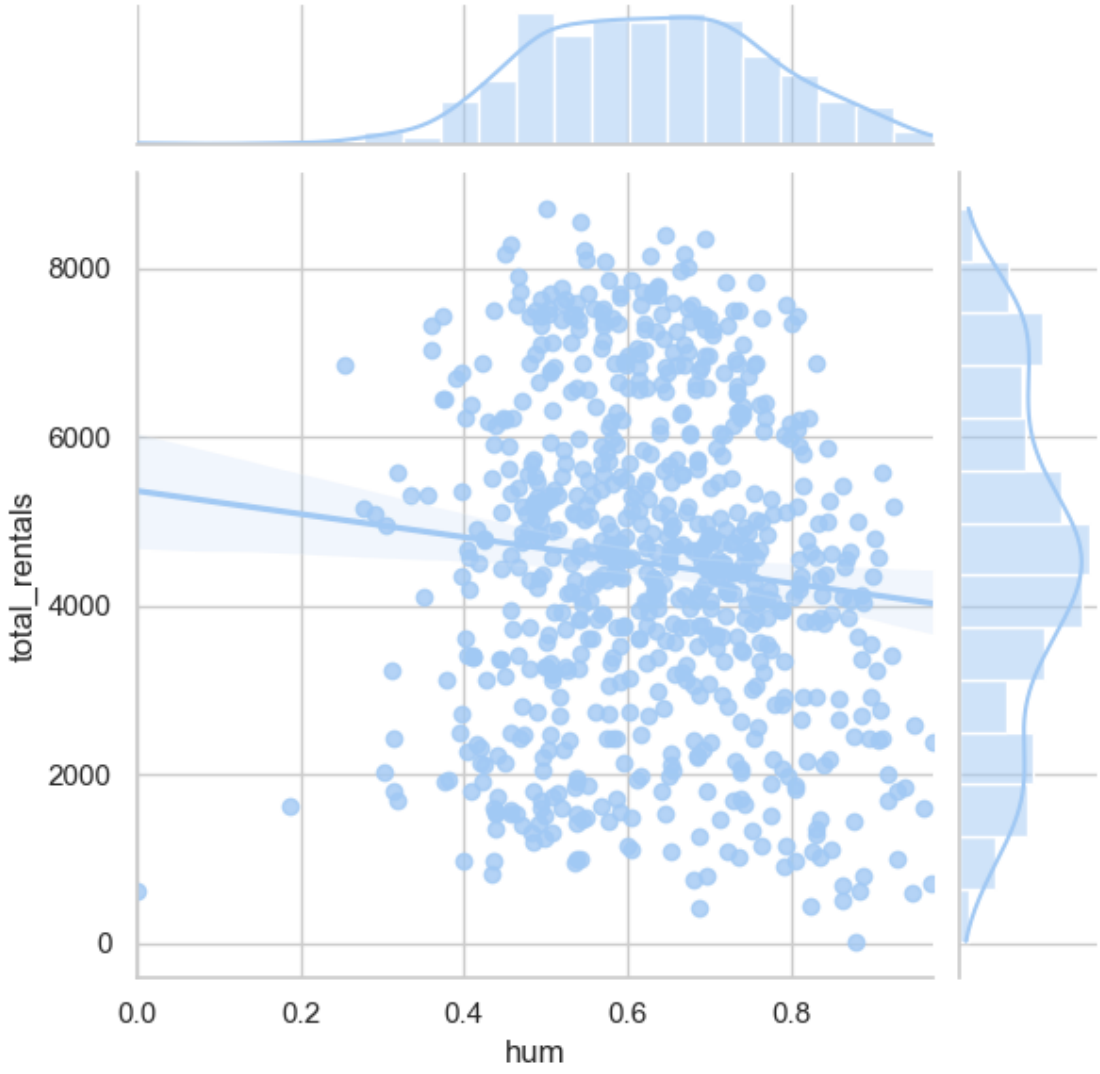
4.4.3 JointGrid y jointplot

[173]:
```
# Build a JointGrid comparing humidity and total_rentals
sns.set_style("whitegrid")
g = sns.JointGrid(x="hum",
                  y="total_rentals",
                  data=bikes,
                  xlim=(0.1, 1.0))

g.plot(sns.regplot, sns.histplot)

plt.show()
plt.clf()
```



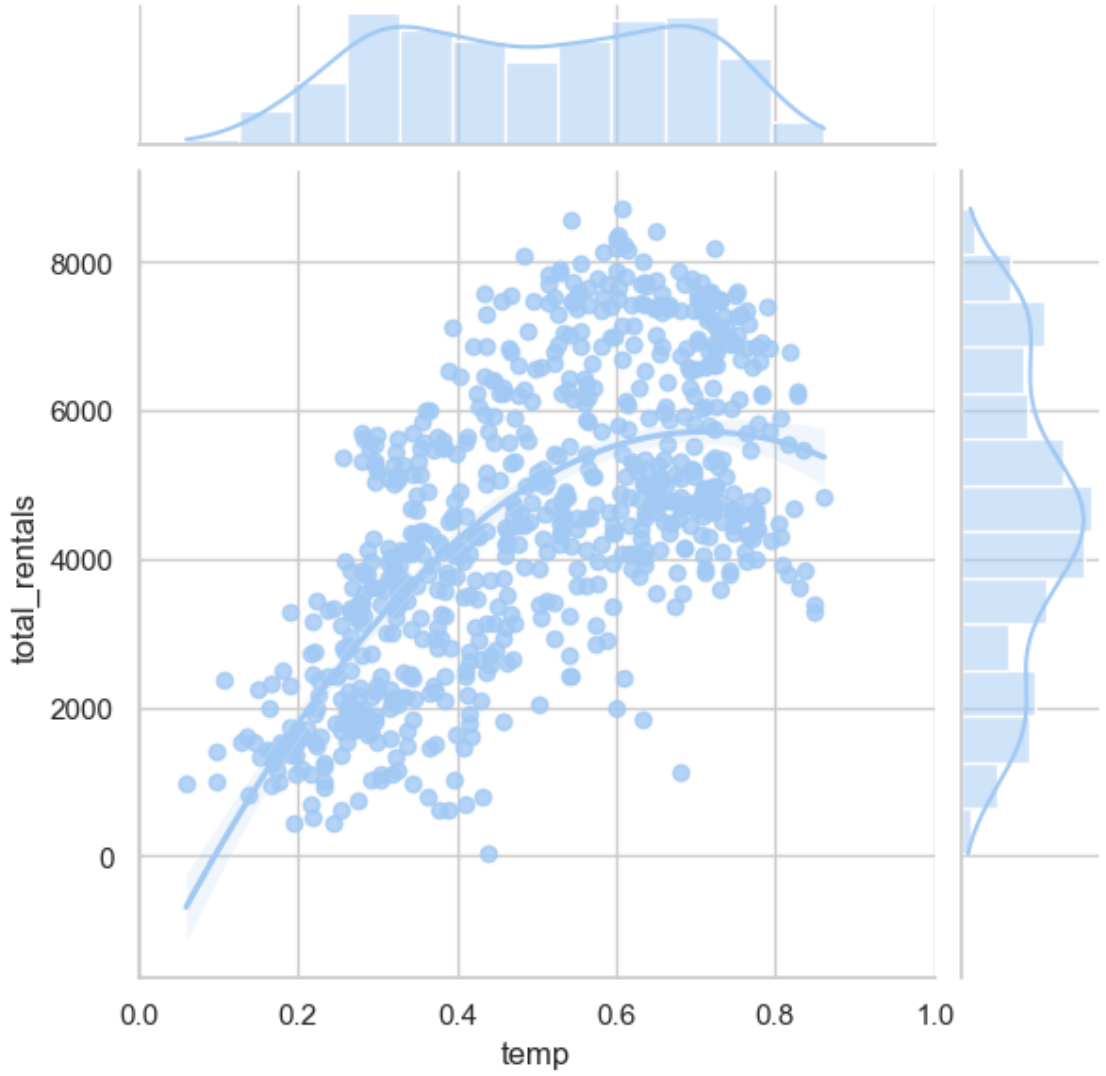

```
<Figure size 640x480 with 0 Axes>
```

[175]:
```
# Create a jointplot similar to the JointGrid
sns.jointplot(x="hum",
        y="total_rentals",
        kind='reg',
        data=bikes)

plt.show()
plt.clf()
```



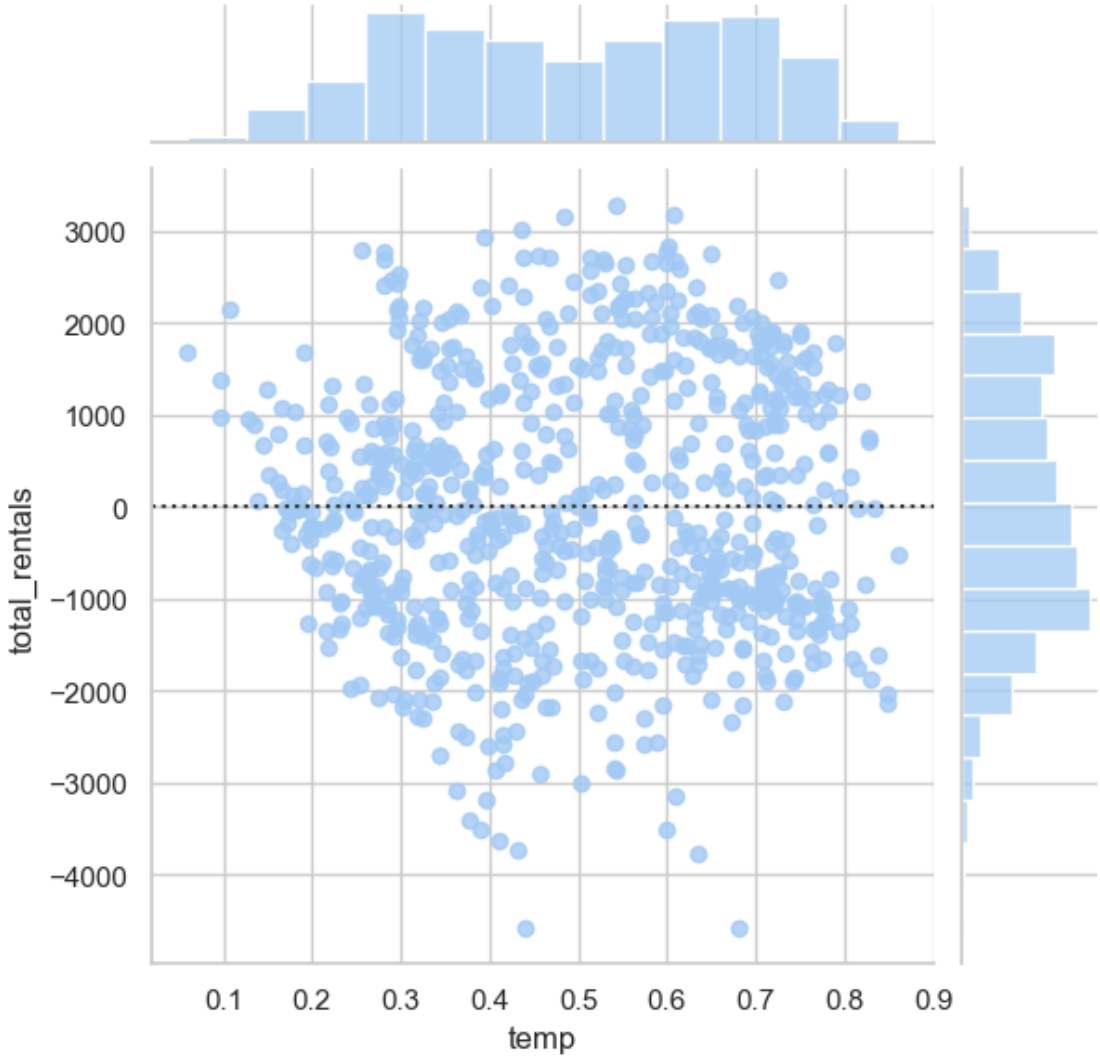
<Figure size 640x480 with 0 Axes>

[176]:
```
# Plot temp vs. total_rentals as a regression plot
sns.jointplot(x="temp",
              y="total_rentals",
              kind='reg',
              data=bikes,
              order=2,
              xlim=(0, 1))

plt.show()
plt.clf()
```



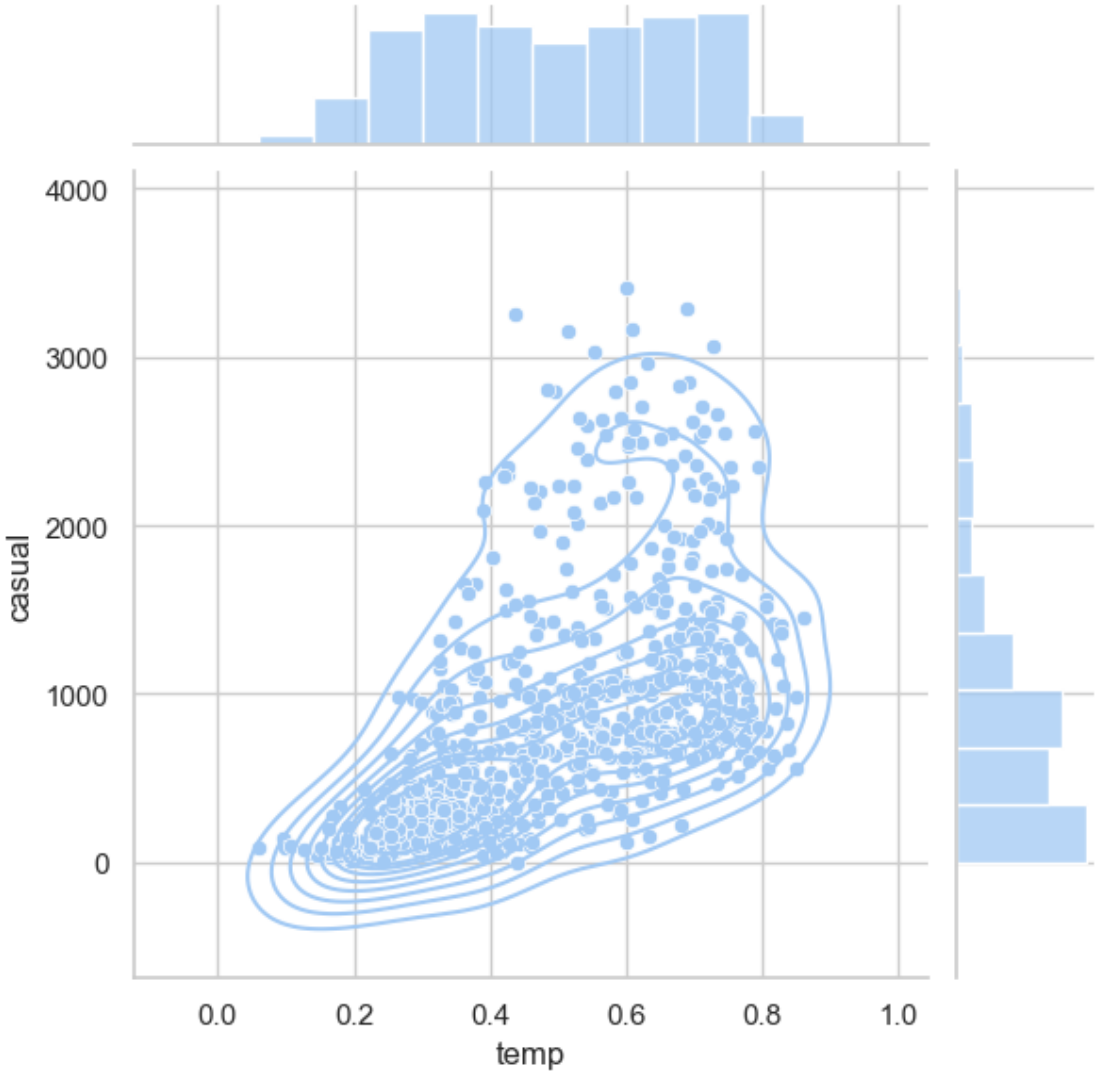

```
<Figure size 640x480 with 0 Axes>
```

[177]:
```
# Plot a jointplot showing the residuals
sns.jointplot(x="temp",
        y="total_rentals",
        kind='resid',
        data=bikes,
        order=2)

plt.show()
plt.clf()
```



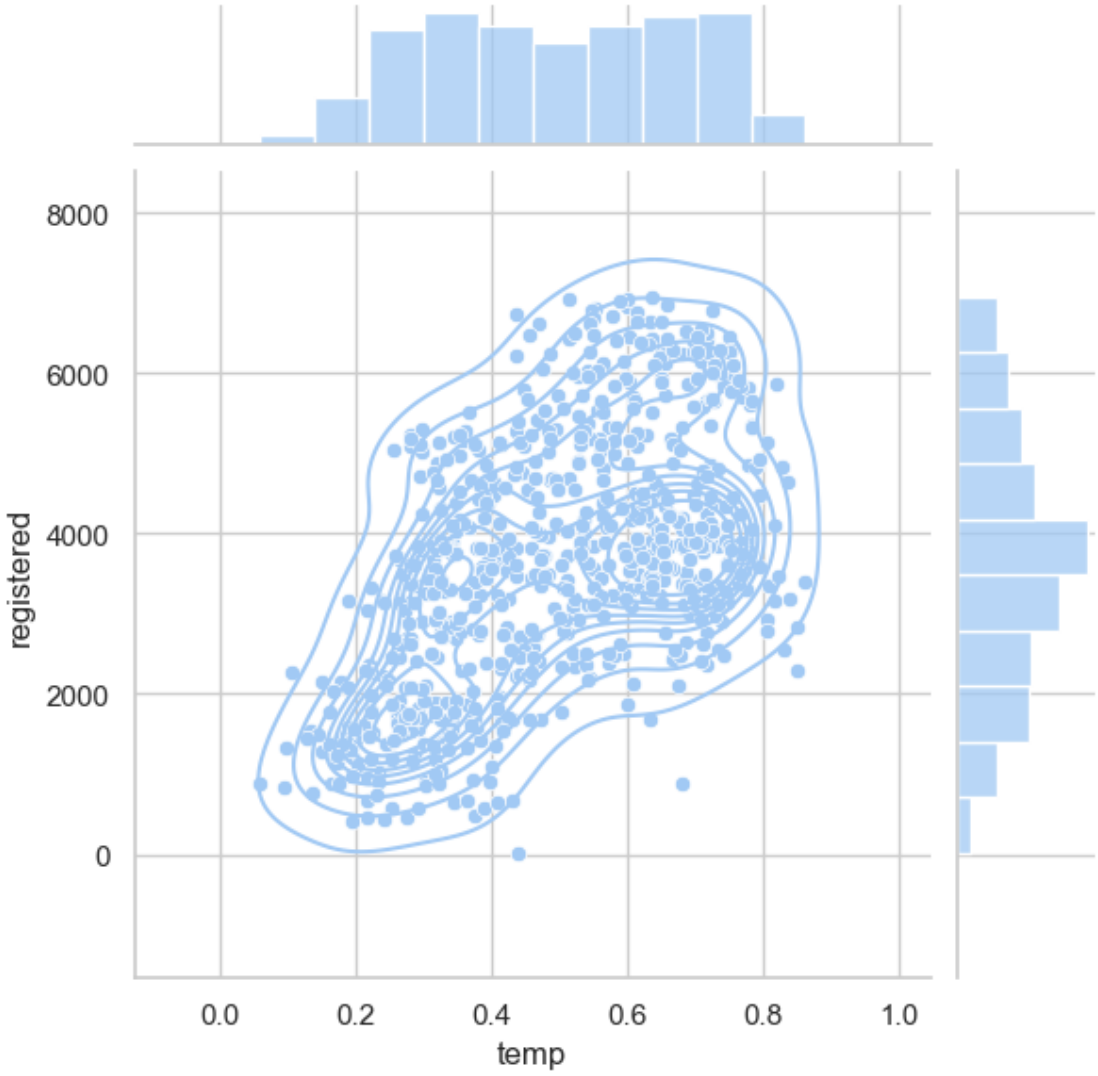

```
<Figure size 640x480 with 0 Axes>
```

[178]:
```
# Create a jointplot of temp vs. casual riders
# Include a kdeplot over the scatter plot
g = sns.jointplot(x="temp",
                  y="casual",
                  kind='scatter',
                  data=bikes,
                  marginal_kws=dict(bins=10))
g.plot_joint(sns.kdeplot)

plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

[179]:
```
# Replicate the above plot but only for registered riders
g = sns.jointplot(x="temp",
                  y="registered",
                  kind='scatter',
                  data=bikes,
                  marginal_kws=dict(bins=10))
g.plot_joint(sns.kdeplot)

plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```