Data Analyses

In [1]:
```
# Packages loader

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

In [2]:
```
# List all the datasets within seaboarn

sns.get_dataset_names()
```

Out[2]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

In [3]:
```
# Loading iris dataset

df_iris = sns.load_dataset('iris')
```

In [4]:
```
# Displaying headers of dataset

df_iris.head()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [5]:

```python
# Learning about the missing information, data-types, non-null row count

df_iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [6]:

```python
# Calculating descriptive statistics

print("Descriptive Statistics For Iris:\n")
df_iris.describe()
```

```
Descriptive Statistics For Iris:
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [7]:

```python
# Calculating statistics for categorical columns

print("Categorical Statistics For Iris:\n")
df_iris.describe(include=['object'])
```

```
Categorical Statistics For Iris:
```

Out[7]:

	species
count	150
unique	3
top	setosa
freq	50

In [8]:
```
# Calculating statistics for all the columns

print("All Statistics For Iris:\n")
df_iris.describe(include='all')
```

```
All Statistics For Iris:
```

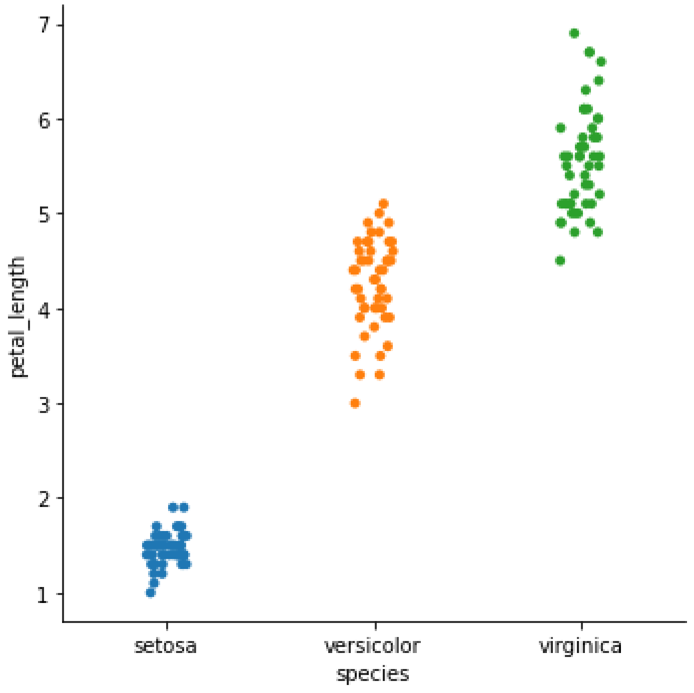
Out[8]:

	sepal_length	sepal_width	petal_length	petal_width	species
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	setosa
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.057333	3.758000	1.199333	NaN
std	0.828066	0.435866	1.765298	0.762238	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

In [ ]:
```

```

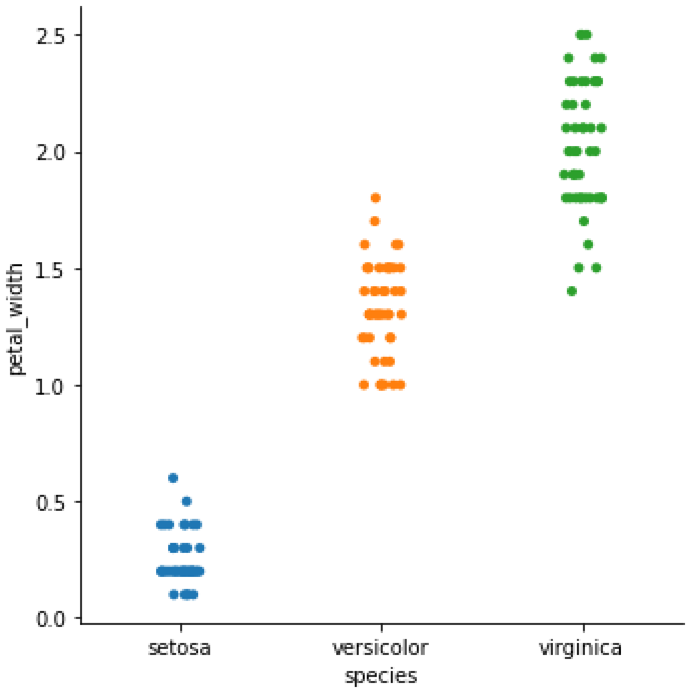
Visualization

In [9]:
```
# Catplot chart representing species against petal length

"""
https://seaborn.pydata.org/generated/seaborn.catplot.html

seaborn.catplot(data=None, *, x=None, y=None, hue=None, row=None, col=None, col_wrap=No
                errorbar=('ci', 95), n_boot=1000, units=None, seed=None, order=None, hu
                col_order=None, height=5, aspect=1, kind='strip', native_scale=False, f
                palette=None, hue_norm=None, legend='auto', legend_out=True, sharex=Tru
                facet_kws=None, ci='deprecated', **kwargs)
"""

catplot_chart = sns.catplot(x = "species", y = "petal_length", data = df_iris)
plt.show()
```



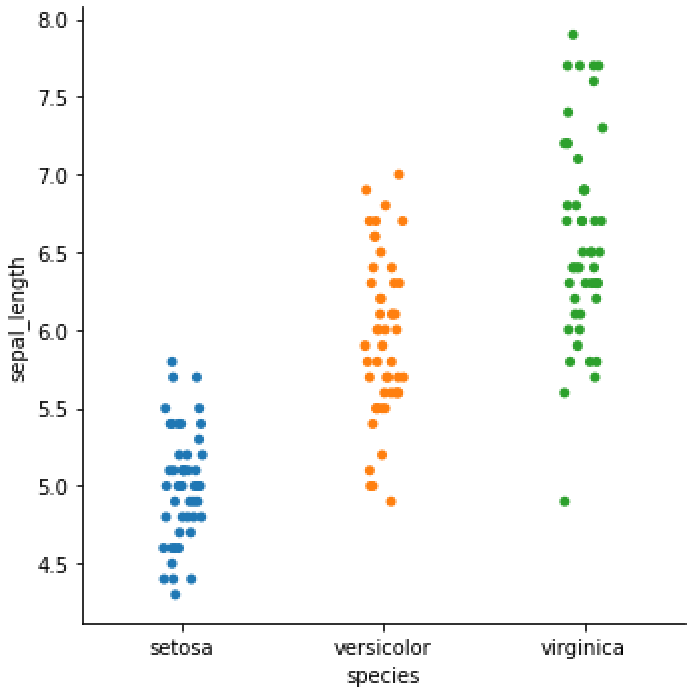
In [10]:

```
# Catplot chart representing petal width against species

"""
https://seaborn.pydata.org/generated/seaborn.catplot.html

seaborn.catplot(data=None, *, x=None, y=None, hue=None, row=None, col=None, col_wrap=No
                errorbar=('ci', 95), n_boot=1000, units=None, seed=None, order=None, hu
                col_order=None, height=5, aspect=1, kind='strip', native_scale=False, f
                palette=None, hue_norm=None, legend='auto', legend_out=True, sharex=Tru
                facet_kws=None, ci='deprecated', **kwargs)
"""

catplot_chart = sns.catplot(x = "species", y = "petal_width", data = df_iris)
plt.show()
```



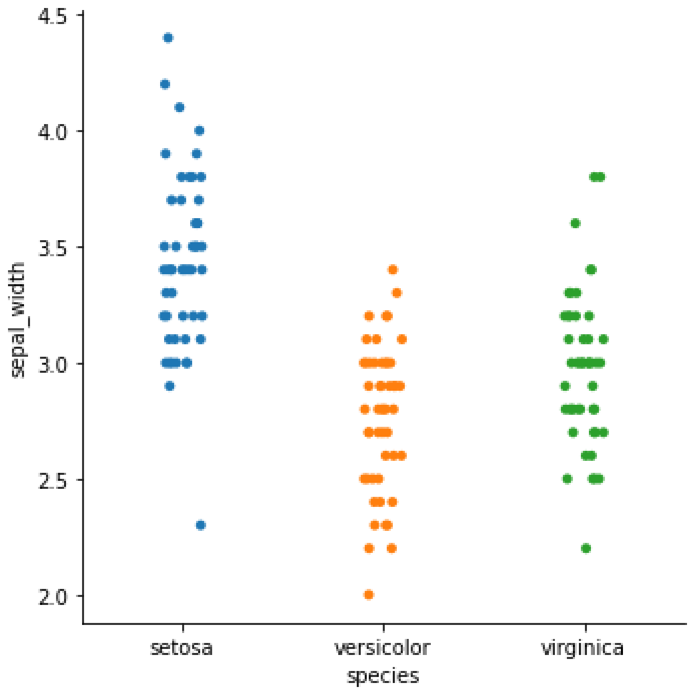
In [11]:

```
# Catplot chart representing sepal length against species

"""
https://seaborn.pydata.org/generated/seaborn.catplot.html

seaborn.catplot(data=None, *, x=None, y=None, hue=None, row=None, col=None, col_wrap=No
                errorbar=('ci', 95), n_boot=1000, units=None, seed=None, order=None, hu
                col_order=None, height=5, aspect=1, kind='strip', native_scale=False, f
                palette=None, hue_norm=None, legend='auto', legend_out=True, sharex=Tru
                facet_kws=None, ci='deprecated', **kwargs)
"""

catplot_chart = sns.catplot(x = "species", y = "sepal_length", data = df_iris)
plt.show()
```



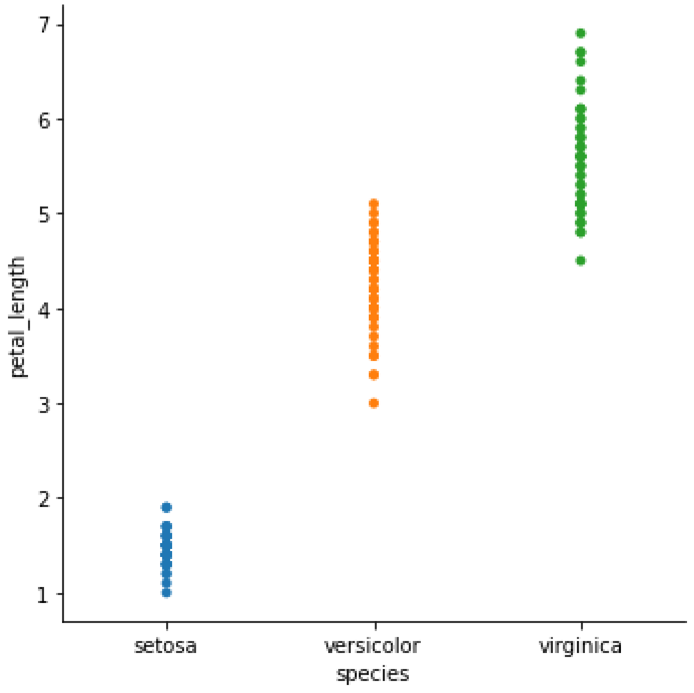
In [12]:

```
# Catplot chart representing sepal width against species

"""
https://seaborn.pydata.org/generated/seaborn.catplot.html

seaborn.catplot(data=None, *, x=None, y=None, hue=None, row=None, col=None, col_wrap=No
                errorbar=('ci', 95), n_boot=1000, units=None, seed=None, order=None, hu
                col_order=None, height=5, aspect=1, kind='strip', native_scale=False, f
                palette=None, hue_norm=None, legend='auto', legend_out=True, sharex=Tru
                facet_kws=None, ci='deprecated', **kwargs)
"""

catplot_chart = sns.catplot(x = "species", y = "sepal_width", data = df_iris)
plt.show()
```



In [13]:

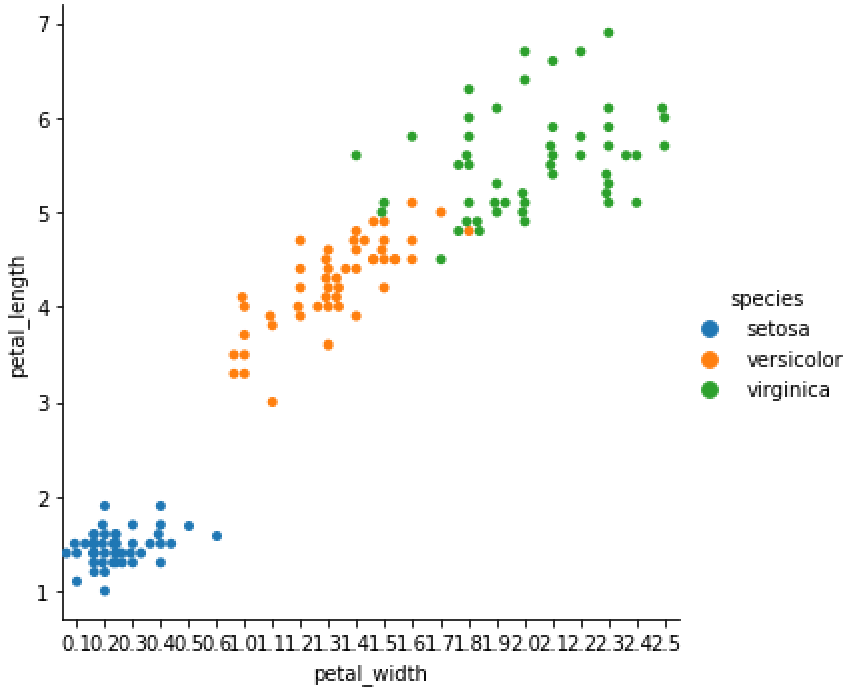
```
# Catplot chart representing petal length against species by removing jitter

"""
https://seaborn.pydata.org/generated/seaborn.catplot.html

seaborn.catplot(data=None, *, x=None, y=None, hue=None, row=None, col=None, col_wrap=No
                errorbar=('ci', 95), n_boot=1000, units=None, seed=None, order=None, hu
                col_order=None, height=5, aspect=1, kind='strip', native_scale=False, f
                palette=None, hue_norm=None, legend='auto', legend_out=True, sharex=Tru
                facet_kws=None, ci='deprecated', **kwargs)

The jitter parameter controls the magnitude of jitter or disables it altogether. jitter
along the horizontal axis, which is useful when there are large clusters of data points

"""

catplot_chart_withoutJitter = sns.catplot(x = "species", y = "petal_length", data = df_
plt.show()
```



In [14]:

```
# Catplot chart representing petal width to petal length defining hue as species

"""
https://seaborn.pydata.org/generated/seaborn.catplot.html

seaborn.catplot(data=None, *, x=None, y=None, hue=None, row=None, col=None, col_wrap=No
                errorbar=('ci', 95), n_boot=1000, units=None, seed=None, order=None, hu
                col_order=None, height=5, aspect=1, kind='strip', native_scale=False, f
                palette=None, hue_norm=None, legend='auto', legend_out=True, sharex=Tru
                facet_kws=None, ci='deprecated', **kwargs)

The jitter parameter controls the magnitude of jitter or disables it altogether. jitter
along the horizontal axis, which is useful when there are large clusters of data points

"""

catplot_chart_HueSwarm = sns.catplot(x = "petal_width", y = "petal_length", data = df_i
plt.show()
```

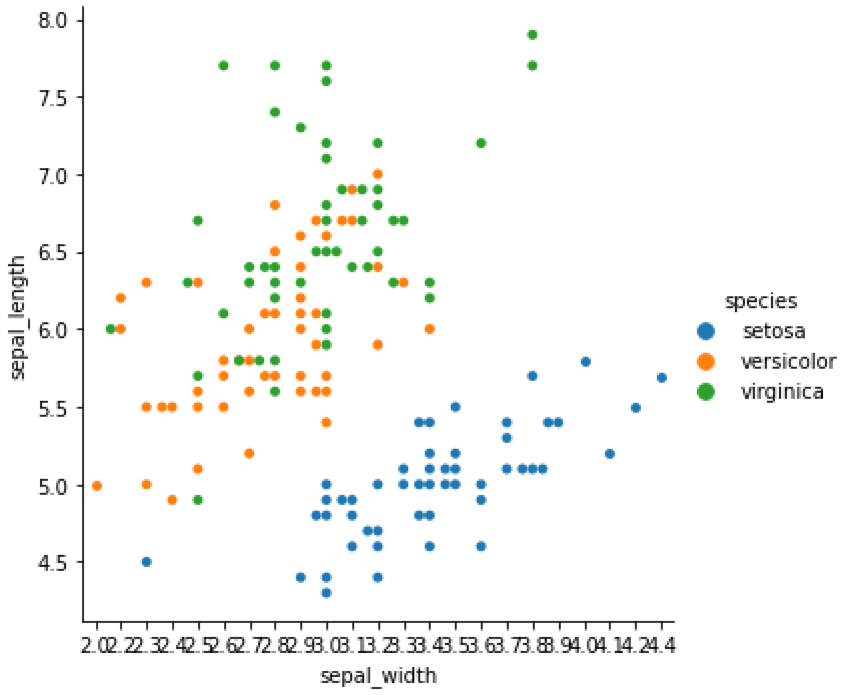
```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 4
8.3% of the points cannot be placed; you may want to decrease the size of the markers or
use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 1
4.3% of the points cannot be placed; you may want to decrease the size of the markers or
use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 1
6.7% of the points cannot be placed; you may want to decrease the size of the markers or
use stripplot.
  warnings.warn(msg, UserWarning)
```



In [15]:

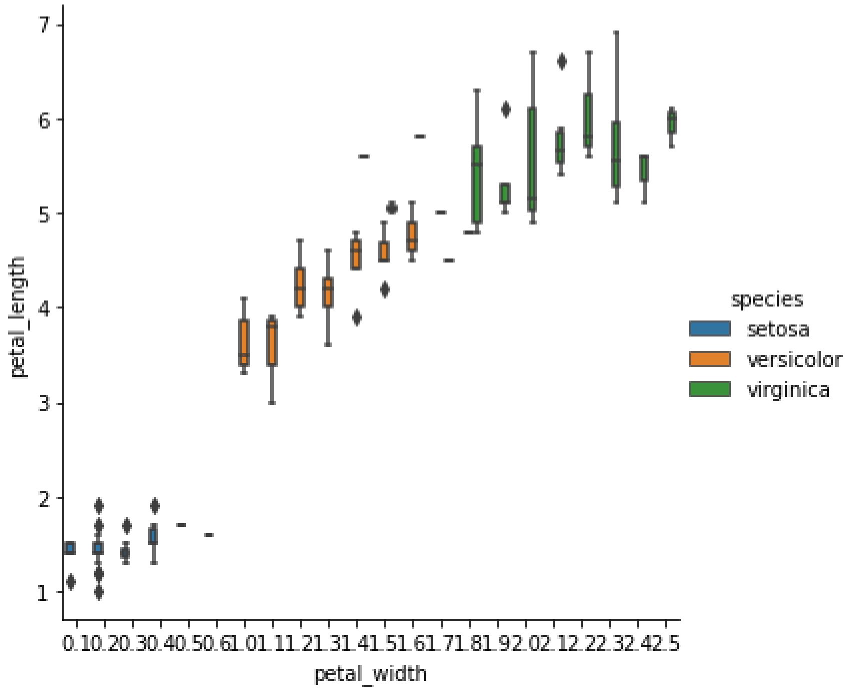
```
# Catplot chart representing sepal width to sepal length defining hue as species

"""
https://seaborn.pydata.org/generated/seaborn.catplot.html

seaborn.catplot(data=None, *, x=None, y=None, hue=None, row=None, col=None, col_wrap=No
                errorbar=('ci', 95), n_boot=1000, units=None, seed=None, order=None, hu
                col_order=None, height=5, aspect=1, kind='strip', native_scale=False, f
                palette=None, hue_norm=None, legend='auto', legend_out=True, sharex=Tru
                facet_kws=None, ci='deprecated', **kwargs)

The jitter parameter controls the magnitude of jitter or disables it altogether. jitter
along the horizontal axis, which is useful when there are large clusters of data points

"""

catplot_chart_HueSwarm = sns.catplot(x = "sepal_width", y = "sepal_length", data = df_i
plt.show()
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 1
1.1% of the points cannot be placed; you may want to decrease the size of the markers or
use stripplot.
  warnings.warn(msg, UserWarning)
```



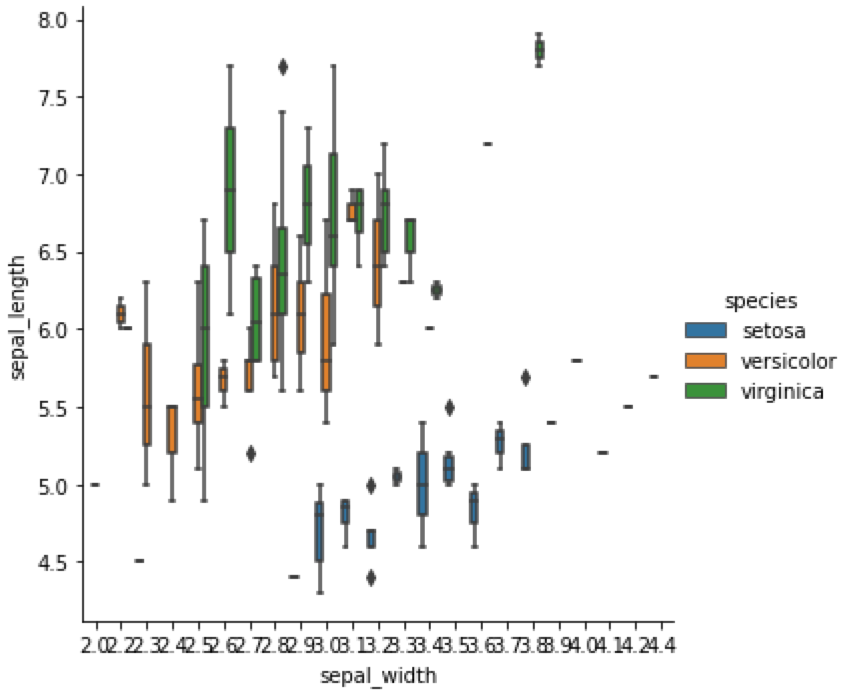
In [16]:

```
# Catplot chart representing petal width to petal length defining hue as species, kind

"""
https://seaborn.pydata.org/generated/seaborn.catplot.html

seaborn.catplot(data=None, *, x=None, y=None, hue=None, row=None, col=None, col_wrap=No
                errorbar=('ci', 95), n_boot=1000, units=None, seed=None, order=None, hu
                col_order=None, height=5, aspect=1, kind='strip', native_scale=False, f
                palette=None, hue_norm=None, legend='auto', legend_out=True, sharex=Tru
                facet_kws=None, ci='deprecated', **kwargs)

The jitter parameter controls the magnitude of jitter or disables it altogether. jitter
along the horizontal axis, which is useful when there are large clusters of data points

"""

catplot_chart_HueBox = sns.catplot(x = "petal_width", y = "petal_length", data = df_iri
plt.show()
```



In [17]:

```
# Catplot chart representing sepal width to sepal length defining hue as species, kind

"""
https://seaborn.pydata.org/generated/seaborn.catplot.html

seaborn.catplot(data=None, *, x=None, y=None, hue=None, row=None, col=None, col_wrap=No
                errorbar=('ci', 95), n_boot=1000, units=None, seed=None, order=None, hu
                col_order=None, height=5, aspect=1, kind='strip', native_scale=False, f
                palette=None, hue_norm=None, legend='auto', legend_out=True, sharex=Tru
                facet_kws=None, ci='deprecated', **kwargs)

The jitter parameter controls the magnitude of jitter or disables it altogether. jitter
along the horizontal axis, which is useful when there are large clusters of data points
"""

catplot_chart_HueBox = sns.catplot(x = "sepal_width", y = "sepal_length", data = df_iri
plt.show()
```

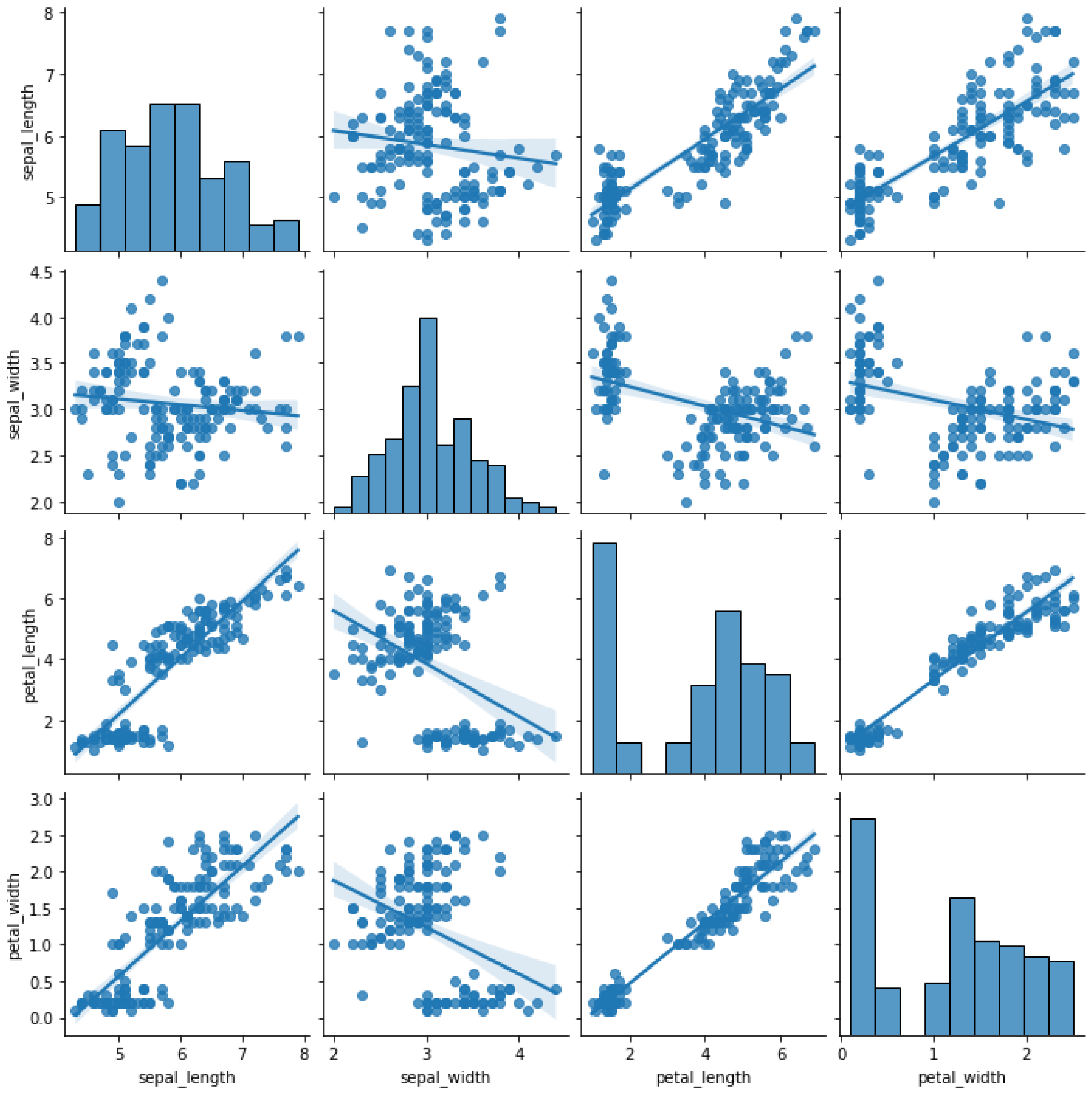


In [18]:

```
# Pairplotting all the columns of the iris dataset

"""
https://seaborn.pydata.org/generated/seaborn.pairplot.html

seaborn.pairplot(data, *, hue=None, hue_order=None, palette=None, vars=None, x_vars=Non
                 diag_kind='auto', markers=None, height=2.5, aspect=1, corner=False, dr
                 diag_kws=None, grid_kws=None, size=None)
"""

sns.pairplot(df_iris, kind="reg")
```

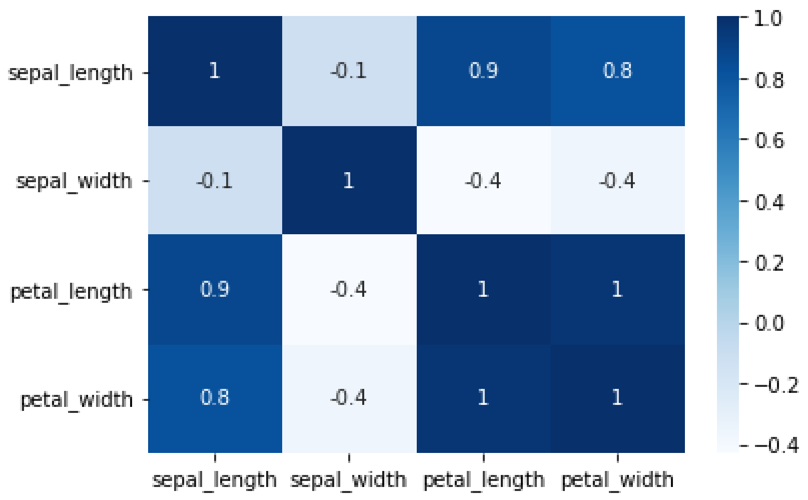
Out[18]:

```
<seaborn.axisgrid.PairGrid at 0x25e119ba3d0>
```



In [19]:

```
# Correlation heatmap for all the columns of the iris dataset

"""
https://seaborn.pydata.org/generated/seaborn.heatmap.html

seaborn.heatmap(data, *, vmin=None, vmax=None, cmap=None, center=None, robust=False, an
                linewidths=0, linecolor='white', cbar=True, cbar_kws=None, cbar_ax=None
                yticklabels='auto', mask=None, ax=None, **kwargs)
"""

heatmap = sns.heatmap(df_iris.corr(), annot=True, cmap="Blues", fmt='.1g')
```



In [ ]:

```

```

Hypotheses

Hypothesis - The average sepal length of all iris flowers is greater than 5.

Data

In [20]:

```
# To test this hypothesis, we can calculate the mean sepal length and compare it to 5.

mean_sepal_length = df_iris['sepal_length'].mean()
print(mean_sepal_length > 5)
```

```
True
```

Conclusion - Since, the mean sepal length is greater than 5, we accept the above stated hypothesis.

--

Hypothesis - The average petal width of all iris flowers is less than 1.2.

Data

In [21]:

```
# To test this hypothesis, we can calculate the mean petal width and compare it to 1.2.

mean_petal_width = df_iris['petal_width'].mean()
print(mean_petal_width < 1.2)
```

```
True
```

Conclusion - Since, the mean petal width is less than 1.2, we accept the above stated hypothesis.

--

Hypothesis - The standard deviation of sepal width for virginica species is less than that of setosa species.

Data

In [22]:

```
# To test this hypothesis, we can calculate the standard deviation of sepal width for e

std_sepal_width_virginica = df_iris[df_iris['species'] == 'virginica']['sepal_width'].s
std_sepal_width_setosa = df_iris[df_iris['species'] == 'setosa']['sepal_width'].std()
print(std_sepal_width_virginica < std_sepal_width_setosa)
```

```
True
```

Conclusion - Since the standard deviation of sepal width for virginica species is less than that of setosa species, we accept the above stated hypothesis.

Hypothesis - The median petal length of versicolor species is equal to the median petal length of virginica species.

Data

In [23]:

```
# To test this hypothesis, we can calculate the median petal length for each species an

median_petal_length_versicolor = df_iris[df_iris['species'] == 'versicolor']['petal_len
median_petal_length_virginica = df_iris[df_iris['species'] == 'virginica']['petal_lengt
print(median_petal_length_versicolor == median_petal_length_virginica)
```

```
False
```

Conclusion - Since the median petal length of versicolor species is different from the median petal length of virginica species, we reject the above stated hypothesis.

Hypothesis - The maximum sepal length among all species is lesser than or equal to 7.

Data

In [24]:

```
# To test this hypothesis, we can find the maximum sepal length and compare it to 7.

max_sepal_length = df_iris['sepal_length'].max()
print(max_sepal_length <= 7)
```

```
False
```

Conclusion - Since the maximum sepal length among atleast one species is greater than 7, we reject the above stated hypothesis.

Hypothesis - The 75th percentile of petal width for setosa species is greater than the 75th percentile of petal width for versicolor species.

Data

In [25]:
```
# To test this hypothesis, we can calculate the 75th percentile of petal width for each

percentile_75_setosa = df_iris[df_iris['species'] == 'setosa']['petal_width'].quantile(
percentile_75_versicolor = df_iris[df_iris['species'] == 'versicolor']['petal_width'].q
print(percentile_75_setosa > percentile_75_versicolor)
```

```
False
```

Conclusion - Since the 75th percentile of petal width for setosa species is greater than the 75th percentile of petal width for versicolor species, we reject the above stated hypothesis.

--

Hypothesis - The range of sepal width for virginica species is bigger than that of versicolor species.

Data

In [26]:
```
# To test this hypothesis, we can calculate the range of sepal width for each species a

range_sepal_width_virginica = df_iris[df_iris['species'] == 'virginica']['sepal_width']
df_iris[df_iris['species'] == 'virginica']['sepal_width'].min()
range_sepal_width_versicolor = df_iris[df_iris['species'] == 'versicolor']['sepal_width
df_iris[df_iris['species'] == 'versicolor']['sepal_width'].min()
print(range_sepal_width_virginica > range_sepal_width_versicolor)
```

```
True
```

Conclusion - Since the range of sepal width for virginica species is bigger than that of versicolor species, we accept the above stated hypothesis.

--

Hypothesis - The mean sepal length for setosa species is significantly different from the mean sepal length for virginica species.

Data

In [27]:
```
# To test this hypothesis, we can perform a t-test between the mean sepal length of the
# the mean sepal length of the virginica species.

import scipy.stats as stats

setosa_sepal_length = df_iris[df_iris['species'] == 'setosa']['sepal_length']
virginica_sepal_length = df_iris[df_iris['species'] == 'virginica']['sepal_length']

t_statistic, p_value = stats.ttest_ind(setosa_sepal_length, virginica_sepal_length)
print(p_value < 0.05)
```

```
True
```

Conclusion - Since the value of p is less than 0.05, we can reject the above stated hypothesis. This means that their means are not significantly different.

```
In [ ]:
```