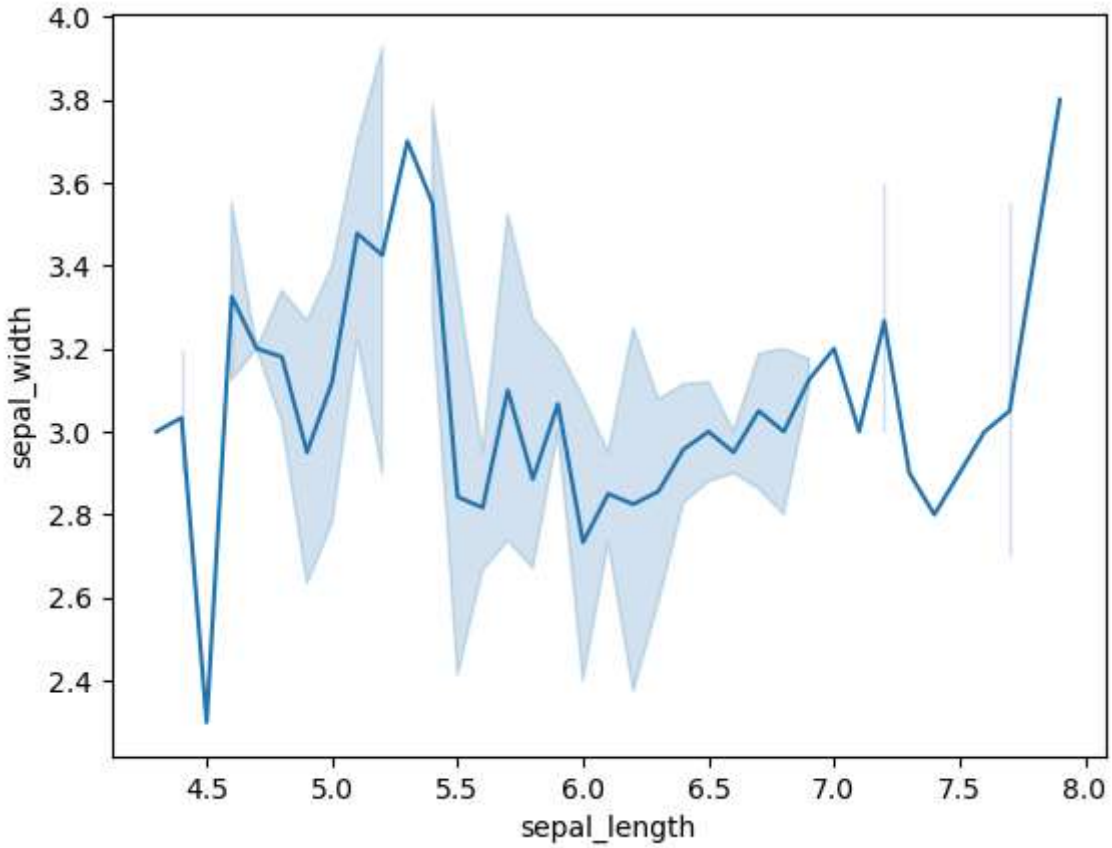
In [1]:
```
# VIVEK-CHAUHAN-ADVANCED-DATA-ANALYTICS-SEABORN-BASICS-WITH-LINE-BAR-CHARTS
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

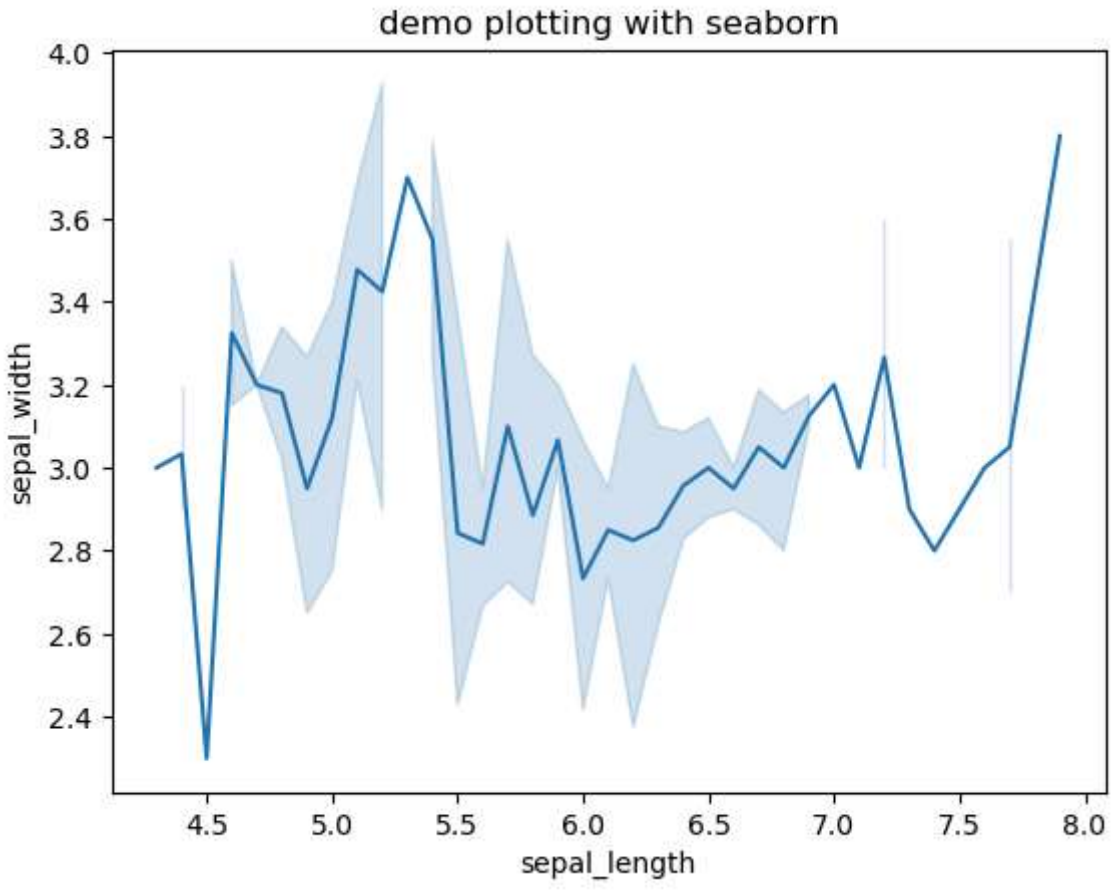
In [5]:
```
# we plot the default dataset which name is iris so iris is a flower plant

data = sns.load_dataset("iris") # loading dataset
print(data)
```

```
     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
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

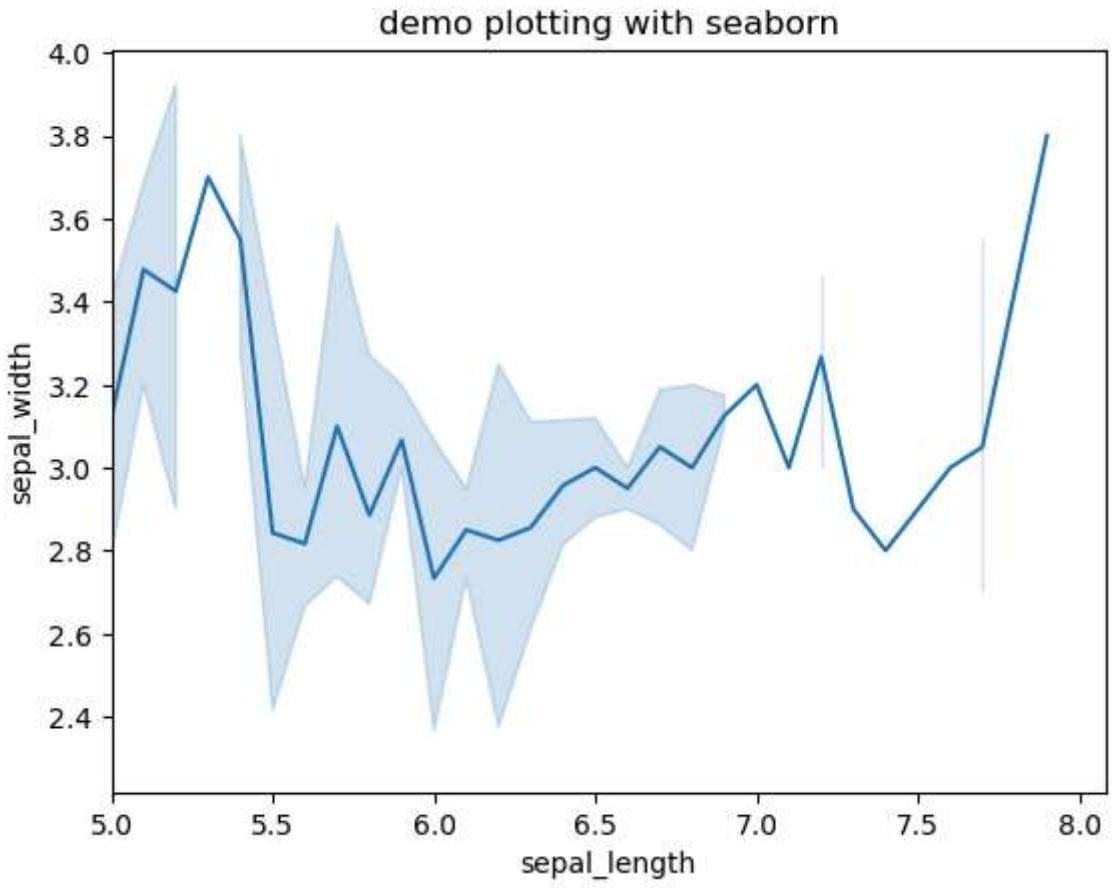
In [9]:
```
# draw a lineplot

a = sns.lineplot(x="sepal_length",y="sepal_width",data = data)
```



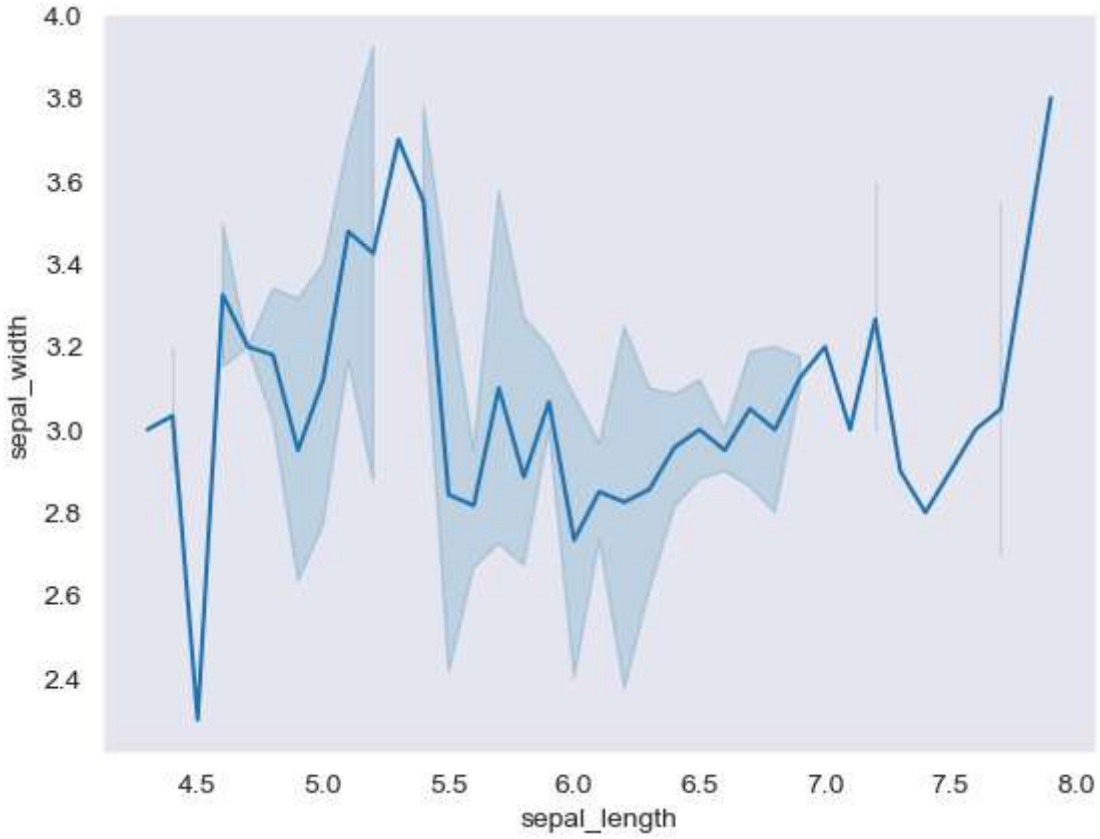
In [13]:
```
# we can create the seaborn with matplotlib also
# draw a lineplot

a = sns.lineplot(x="sepal_length",y="sepal_width",data = data)
plt.title("demo plotting with seaborn")
plt.show()
```



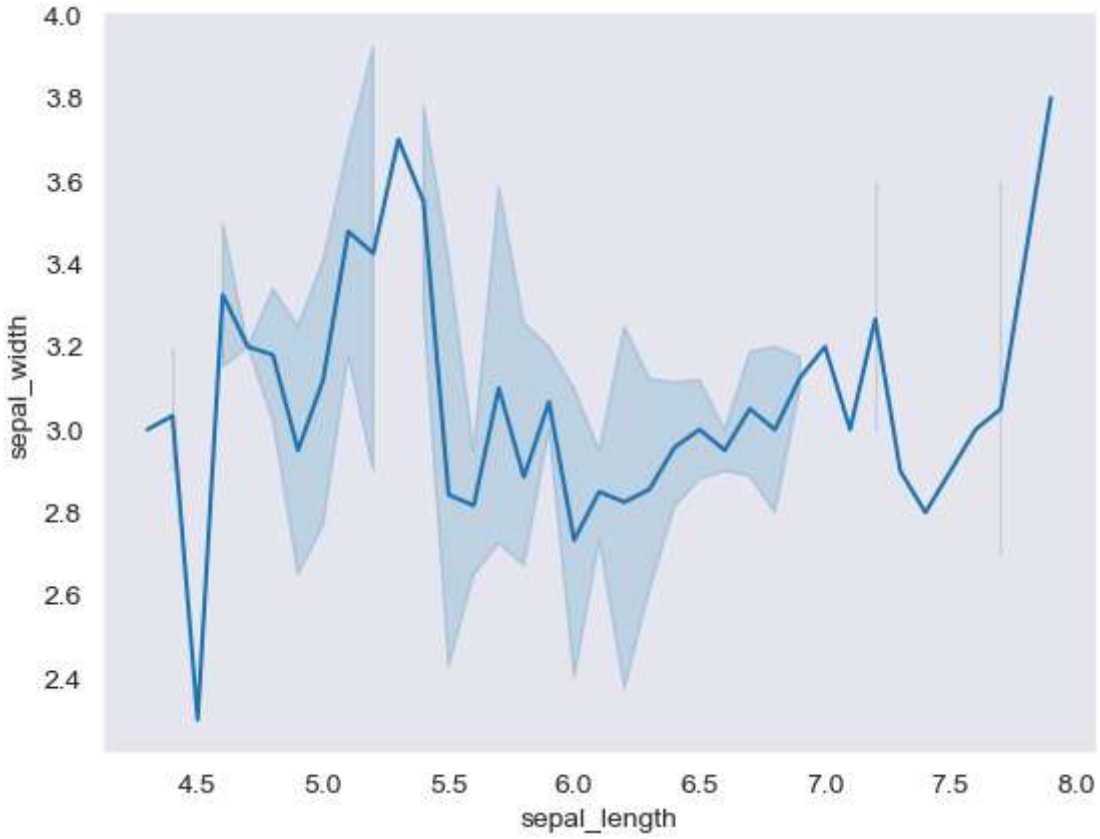
In [11]:
```
# we can also give xlim and ylim same as matplot
# we can create the seaborn with matplotlib also
# draw a lineplot

a = sns.lineplot(x="sepal_length",y="sepal_width",data = data)
plt.title("demo plotting with seaborn")
plt.xlim(5)
plt.show()
```



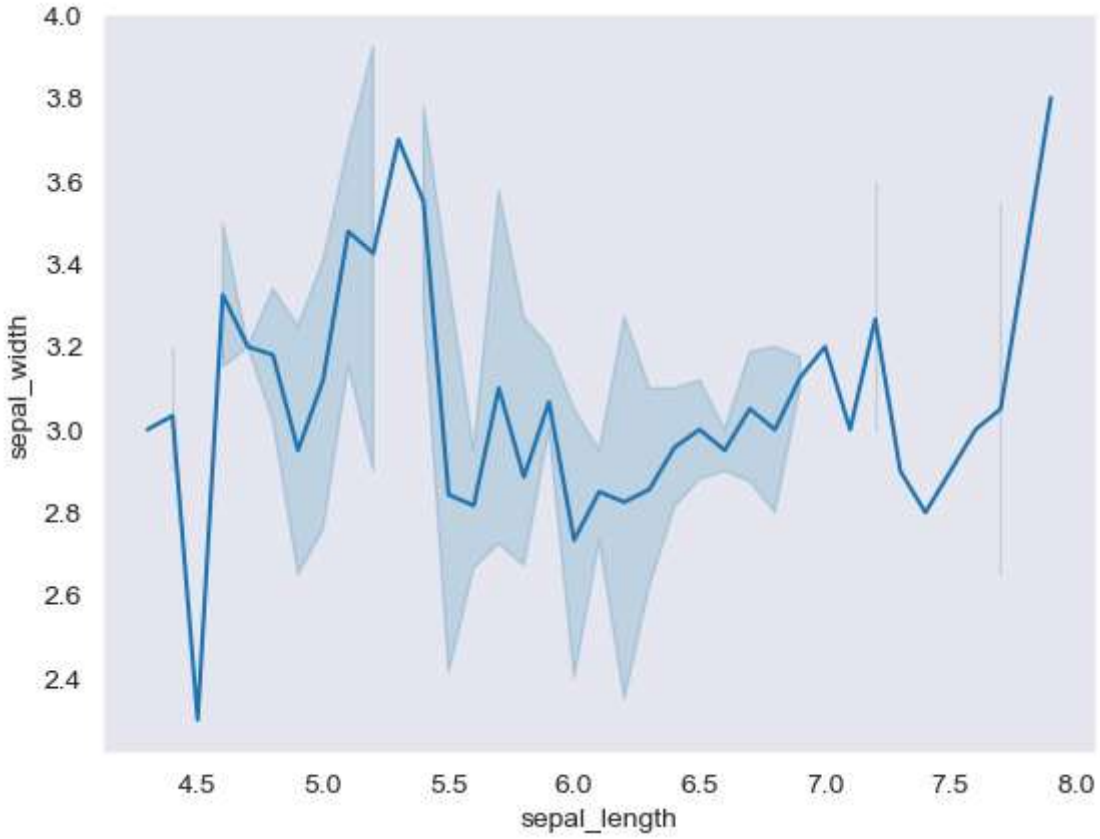
In [31]:
```
# using set style we can give the customised background options

a = sns.lineplot(x="sepal_length",y="sepal_width",data = data)
sns.set_style("dark")
```



In [27]:
```
# for removing spines using despine()

a = sns.lineplot(x="sepal_length",y="sepal_width",data = data)
sns.set_style("dark")
sns.despine()# by default top,right line will be removed if mention left=true then
```



In [55]:
```
# we can change the figure size by using matplotlib figure function

sns.lineplot(x="sepal_length",y="sepal_width",data = data)
sns.set_style("dark")
plt.figure(figsize = (0.10,0.10))
plt.show()
```



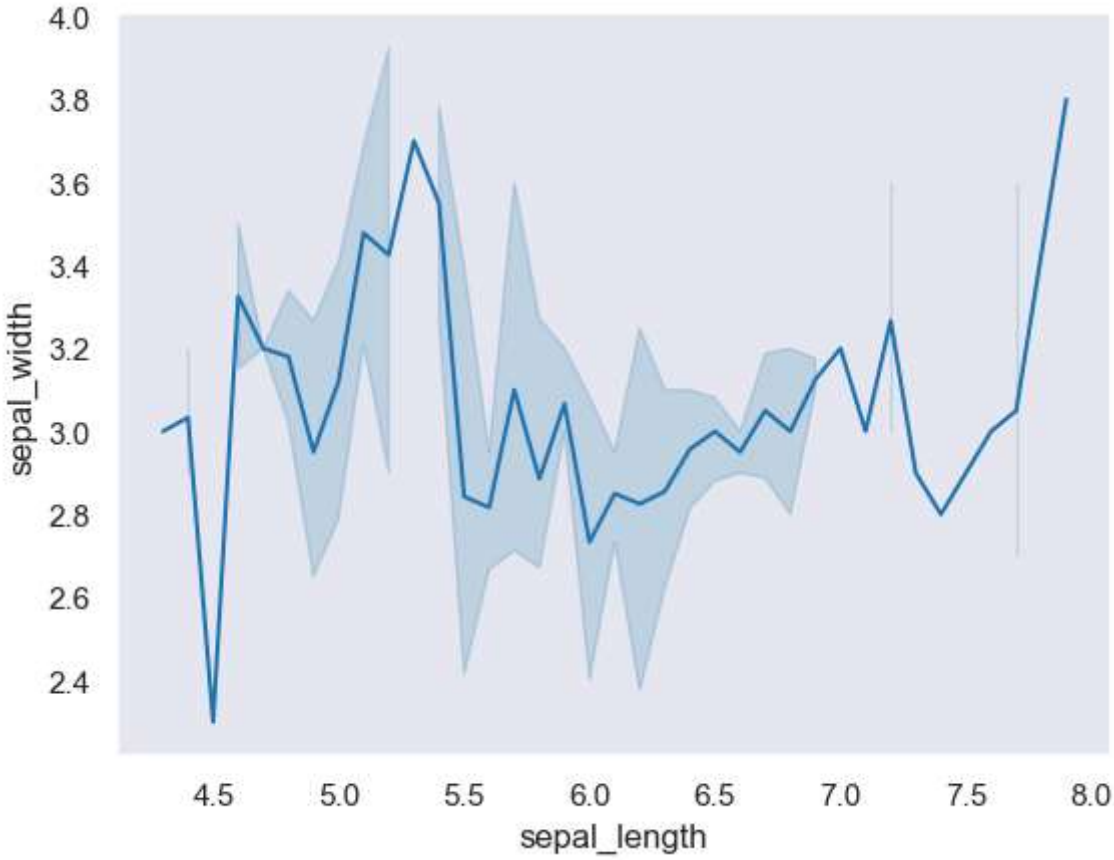

```
<Figure size 10x10 with 0 Axes>
```

In [59]:
```
# for color pallette() we use that function

a = sns.color_palette()
sns.palplot(a)
plt.show()
```

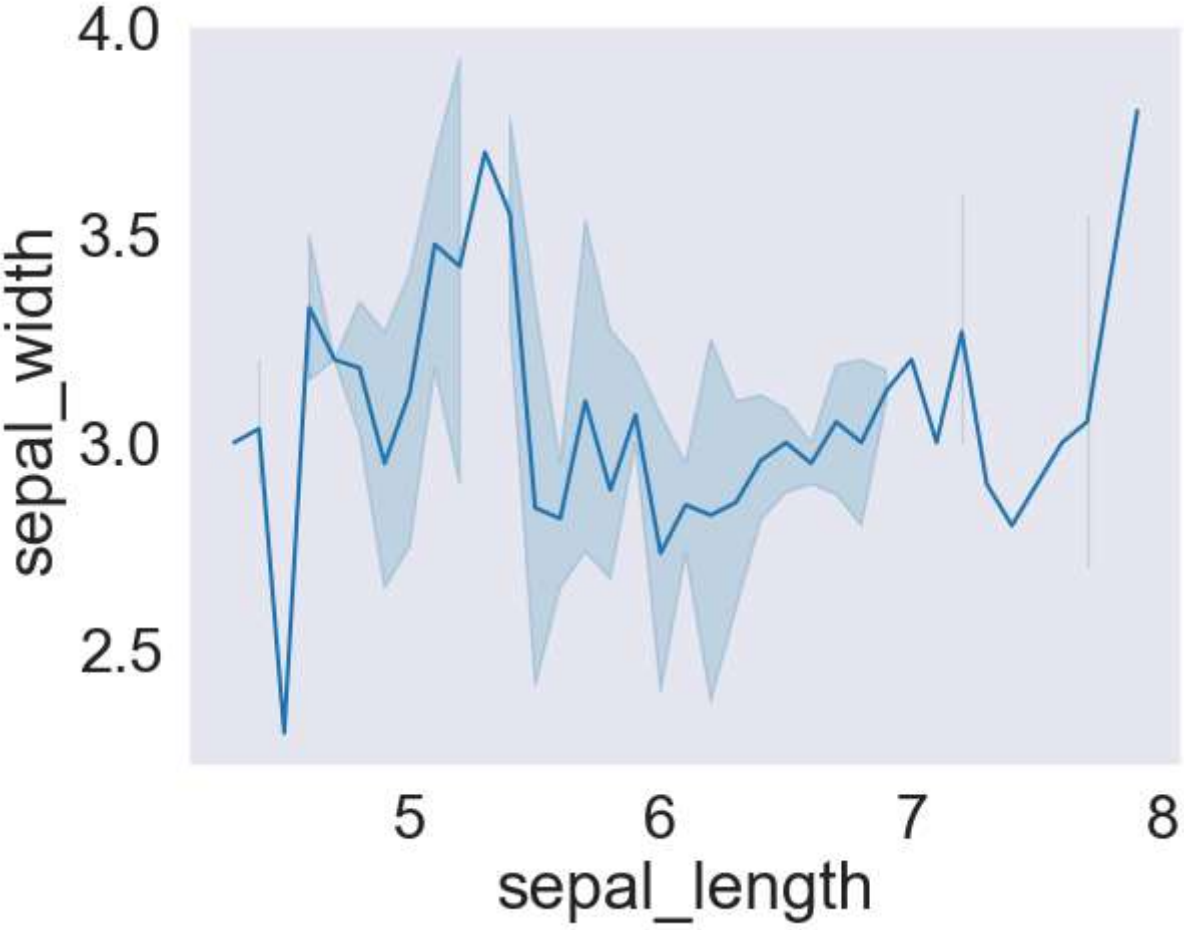
In [36]:
```
# we can change the shade of the colors by usign customisation
# for color palette() we use that function

a = sns.color_palette("Greens",11) # the name is case-sensitive
sns.palplot(a) # palplot() is used for the colors of the palette
plt.show()
```

In [46]:
```
# to change the look like the whole plot just write the set_context() it will chang
# set_context() is paper,notebook,talk,poster
# font_scale = font size it will change the text property

a = sns.load_dataset("iris")
```

```
sns.lineplot(x="sepal_length",y="sepal_width",data=a)
sns.set_context("notebook")
```



In [64]:
```
# font_scale = font size it will change the text property
a = sns.load_dataset("iris")
sns.lineplot(x="sepal_length",y="sepal_width",data=a)
sns.set_context("notebook",font_scale=2) # font_scale = font size temperarily.
```



In [68]:
```
# seaborn color_palette() classify in 3 different ways qualitative,sequential,diver
current_palette = sns.color_palette()
sns.palplot(current_palette)
```

In [74]:
```
sns.color_palette()
```

Out[74]:

In [86]:
```
# sequential color paletter for the sequence data
# default it will take the grade from lighter to brighter
color = sns.color_palette("Greys")
sns.palplot(color)
```

In [92]:
```
# diverging palette for the positive and negative paletter for better visualization
color = sns.color_palette("deep",5)
sns.palplot(color)
```

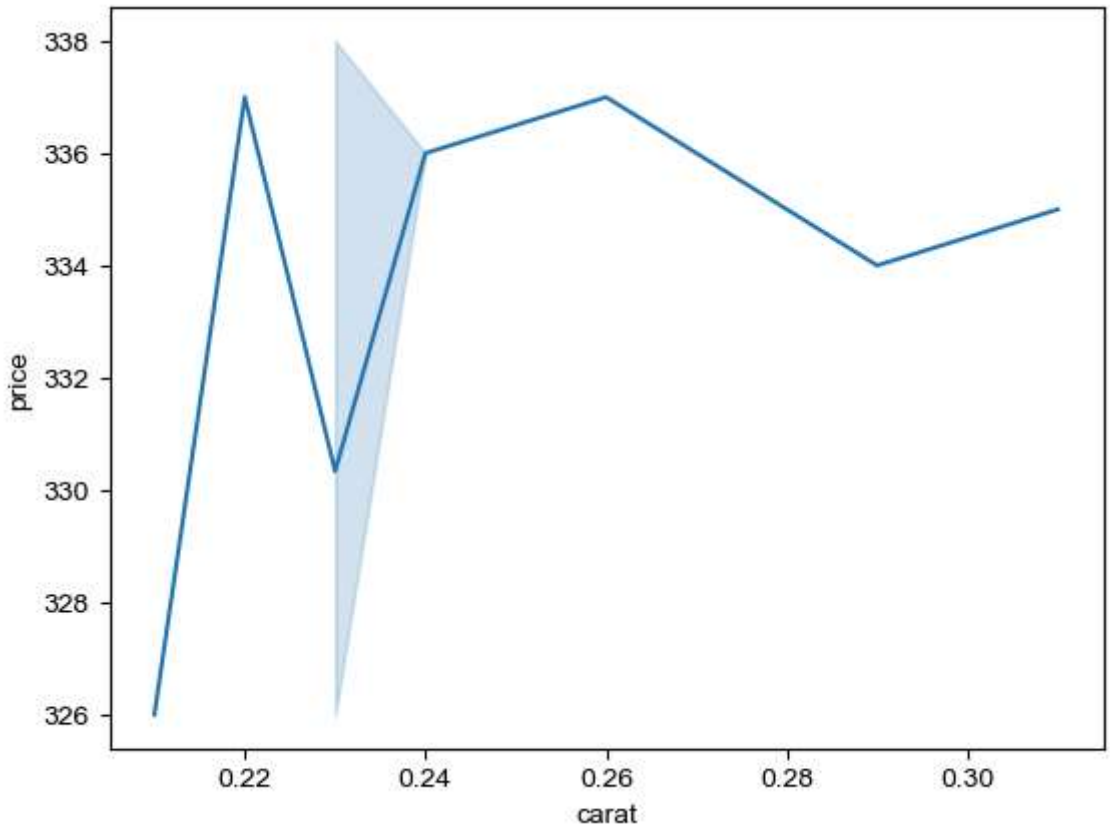
In [13]:
```
# line plot using seaborn
```

In [11]:
```
# load the dataset by using sns.load_dataset Diamonds which is default dataset
a = sns.load_dataset("diamonds")
print(a)
```

```
       carat        cut color clarity  depth  table  price     x     y     z
0       0.23      Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1       0.21    Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2       0.23       Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3       0.29    Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4       0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
...      ...        ...   ...     ...    ...    ...    ...   ...   ...   ...
53935   0.72      Ideal     D     SI1   60.8   57.0   2757  5.75  5.76  3.50
53936   0.72       Good     D     SI1   63.1   55.0   2757  5.69  5.75  3.61
53937   0.70  Very Good     D     SI1   62.8   60.0   2757  5.66  5.68  3.56
53938   0.86    Premium     H     SI2   61.0   58.0   2757  6.15  6.12  3.74
53939   0.75      Ideal     D     SI2   62.2   55.0   2757  5.83  5.87  3.64

[53940 rows x 10 columns]
```

In [13]:
```
# now this time to make the line plot

slicing = a.iloc[0:10,:] # we make the line plot only 10 data
print(slicing)

data = a.head(10) # we have to done this via head() cause the slicing will raise an

line = sns.lineplot(x = "carat",y = "price",data=data)
sns.set_style("darkgrid") # lets give some background attributes using set() functi
plt.show() # its not necessary but write it
```

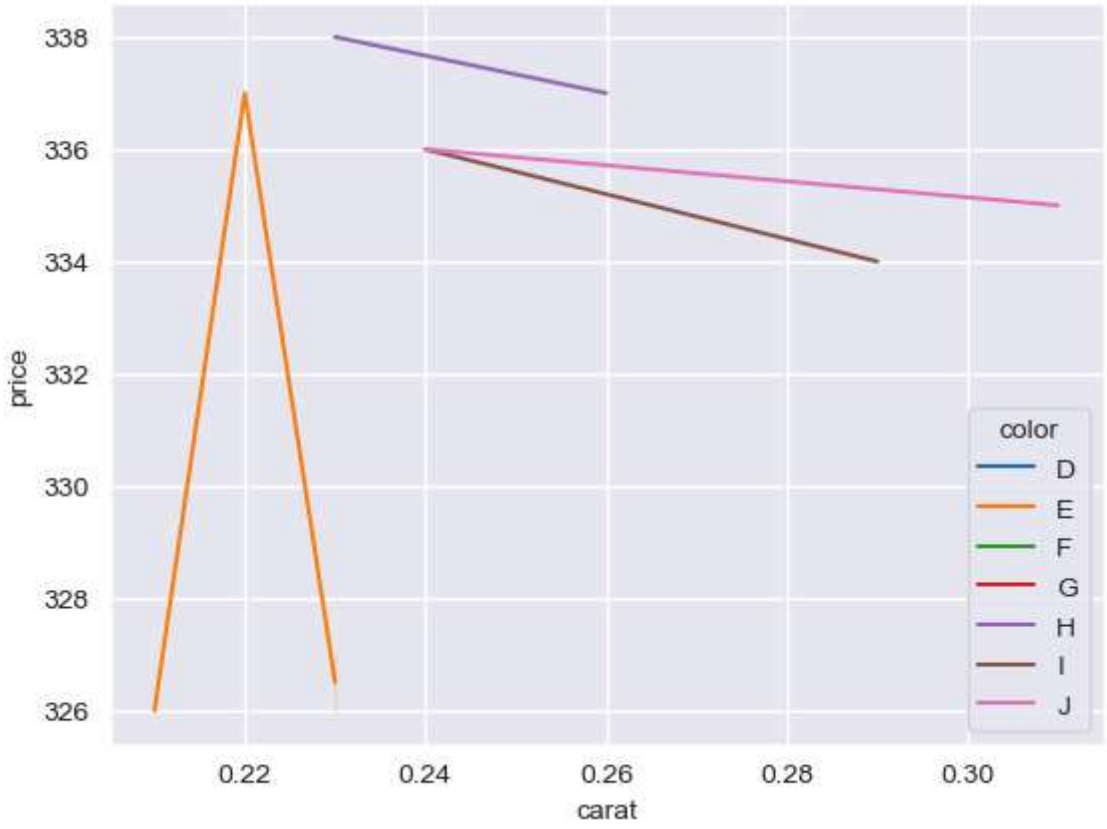
```
   carat        cut color clarity  depth  table  price     x     y     z
0   0.23      Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1   0.21    Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2   0.23       Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3   0.29    Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4   0.31       Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
5   0.24  Very Good     J    VVS2   62.8   57.0    336  3.94  3.96  2.48
6   0.24  Very Good     I    VVS1   62.3   57.0    336  3.95  3.98  2.47
7   0.26  Very Good     H     SI1   61.9   55.0    337  4.07  4.11  2.53
8   0.22       Fair     E     VS2   65.1   61.0    337  3.87  3.78  2.49
9   0.23  Very Good     H     VS1   59.4   61.0    338  4.00  4.05  2.39
```



In [127…

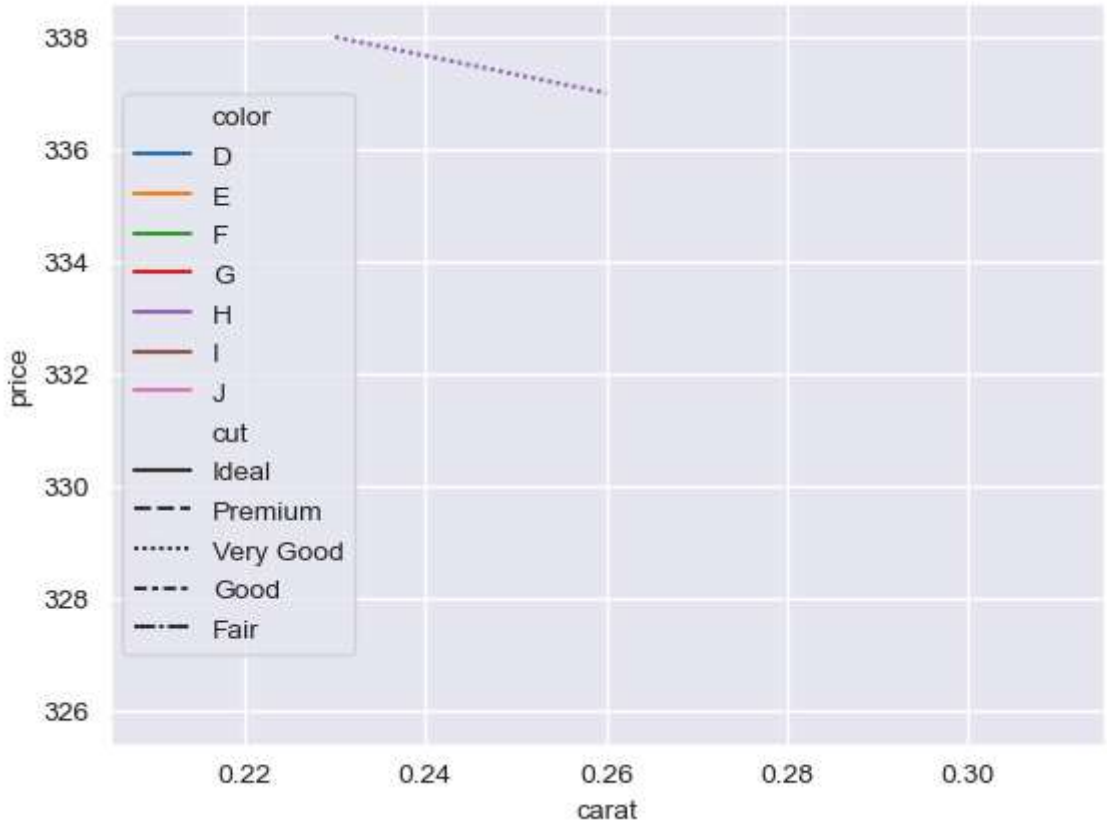
```
# hue is used to visualize the data in the different categories in one plot
# now this time to make the line plot

slicing = a.iloc[0:10,:] # we make the line plot only 10 data
print(slicing)

data = a.head(10) # we have to done this via head() cause the slicing will raise an

line = sns.lineplot(x = "carat",y = "price",data=data,hue="color")
sns.set_style("darkgrid") # lets give some background attributes using set() functi
plt.show() # its not necessary but write it
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
5	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
6	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
7	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
8	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
9	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39



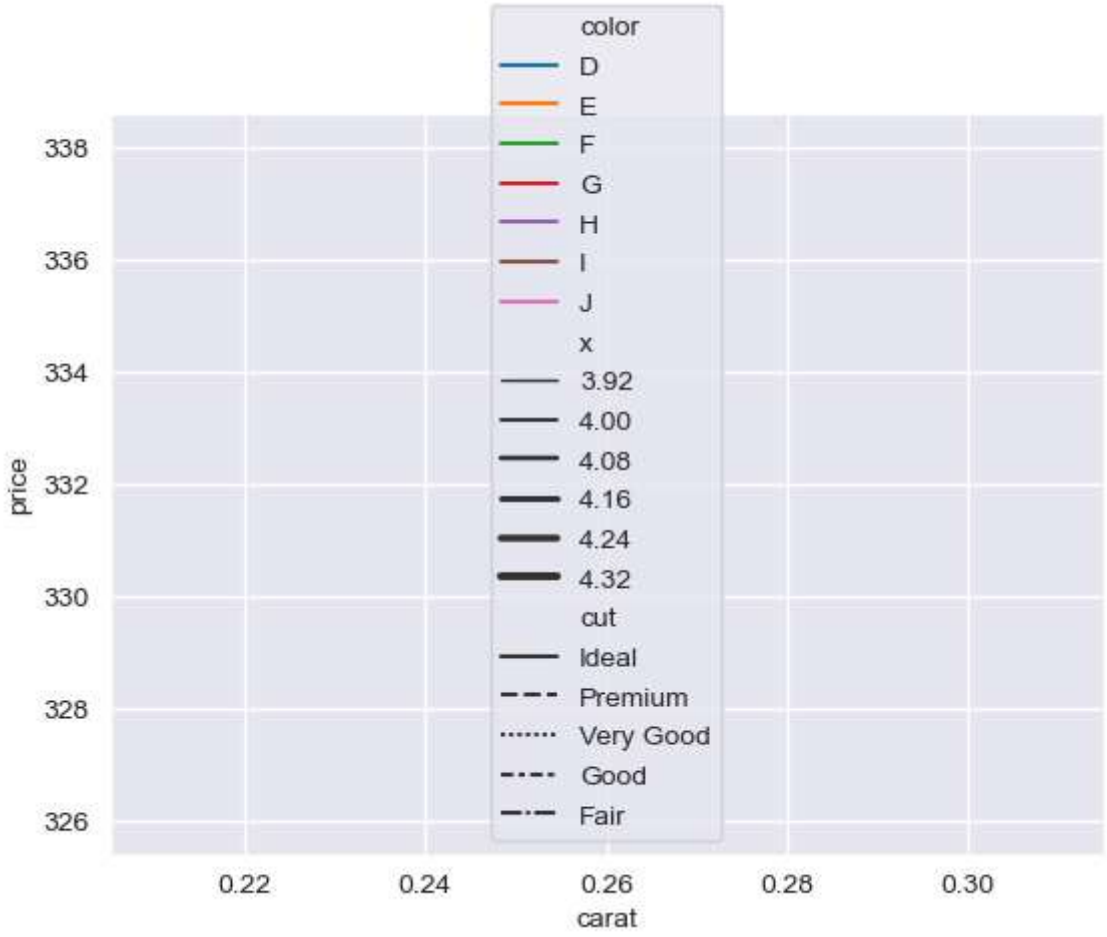
In [15]:
```
# line style decides the which data is suppose to be displayed
# hue is used to visualize the data in the different categories in one plot
# now this time to make the line plot

slicing = a.iloc[0:10,:] # we make the line plot only 10 data
print(slicing)

data = a.head(10) # we have to done this via head() cause the slicing will raise an

line = sns.lineplot(x = "carat",y = "price",data=data,hue="color",style="cut") # he
sns.set_style("darkgrid") # lets give some background attributes using set() functi
plt.show() # its not necessary but write it
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
5	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
6	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
7	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
8	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
9	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39



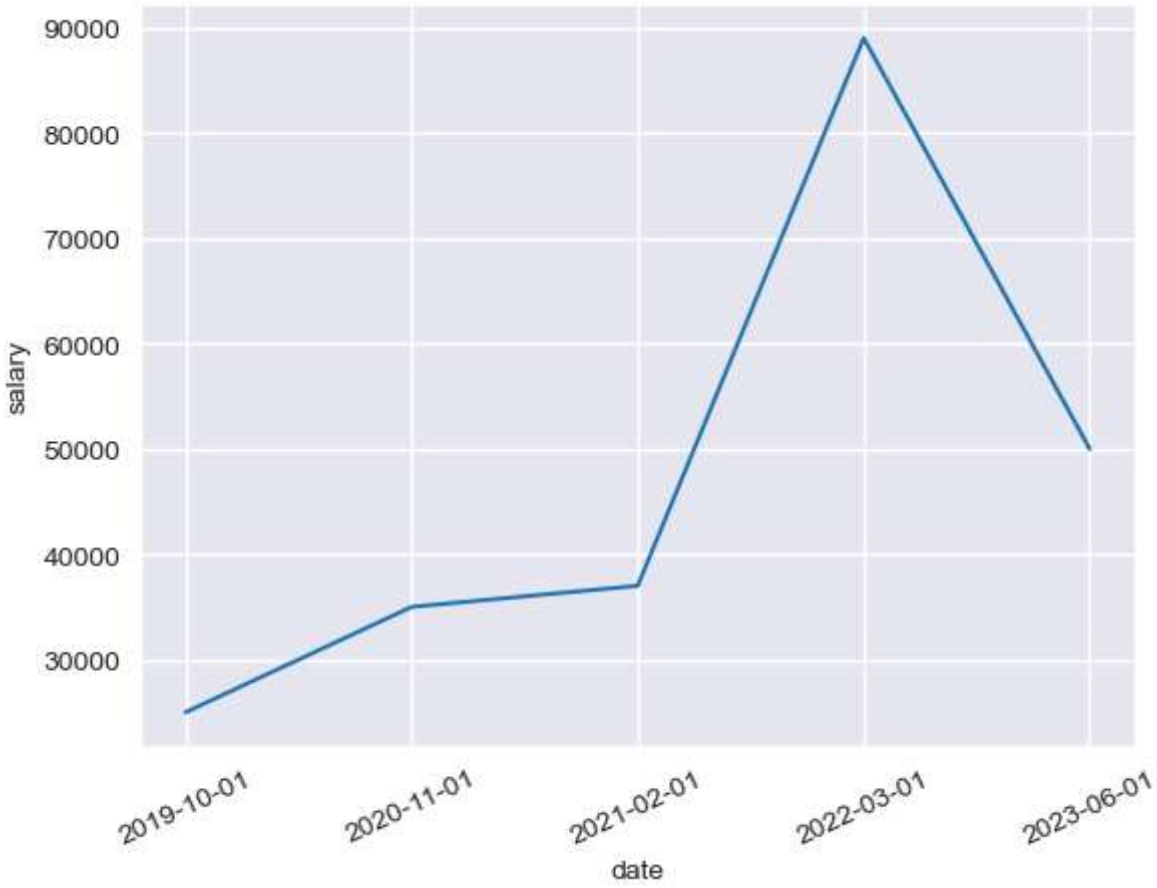
In [17]:
```
# we can differentite using the size
# line style decides the which data is suppose to be displayed
# hue is used to visualize the data in the different categories in one plot
# now this time to make the line plot

slicing = a.iloc[0:10,:] # we make the line plot only 10 data
print(slicing)

data = a.head(10) # we have to done this via head() cause the slicing will raise an

line = sns.lineplot(x = "carat",y = "price",data=data,hue="color",style="cut",size=
sns.set_style("darkgrid") # lets give some background attributes using set() functi
plt.show() # its not necessary but write it
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
5	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
6	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
7	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
8	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
9	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39



In [91]:
```
# now we learn time-series means the data will be change over the time
# first we create a demo dataframe to visualize effectively

employee = {
    "salary":[25000,35000,37000,89000,50000],
    "date" : ['2019-10-01','2020-11-01','2021-02-01','2022-03-01','2023-06-01']
}

dataframe = pd.DataFrame(employee)
print(dataframe)

# lets visualize the timeseries plot

sns.lineplot(x="date",y="salary",data=dataframe)
# rotation is matplot library so plt.show() is required
plt.xticks(rotation=25)# here the rotation of xticks for better visualization and r
plt.show()
```

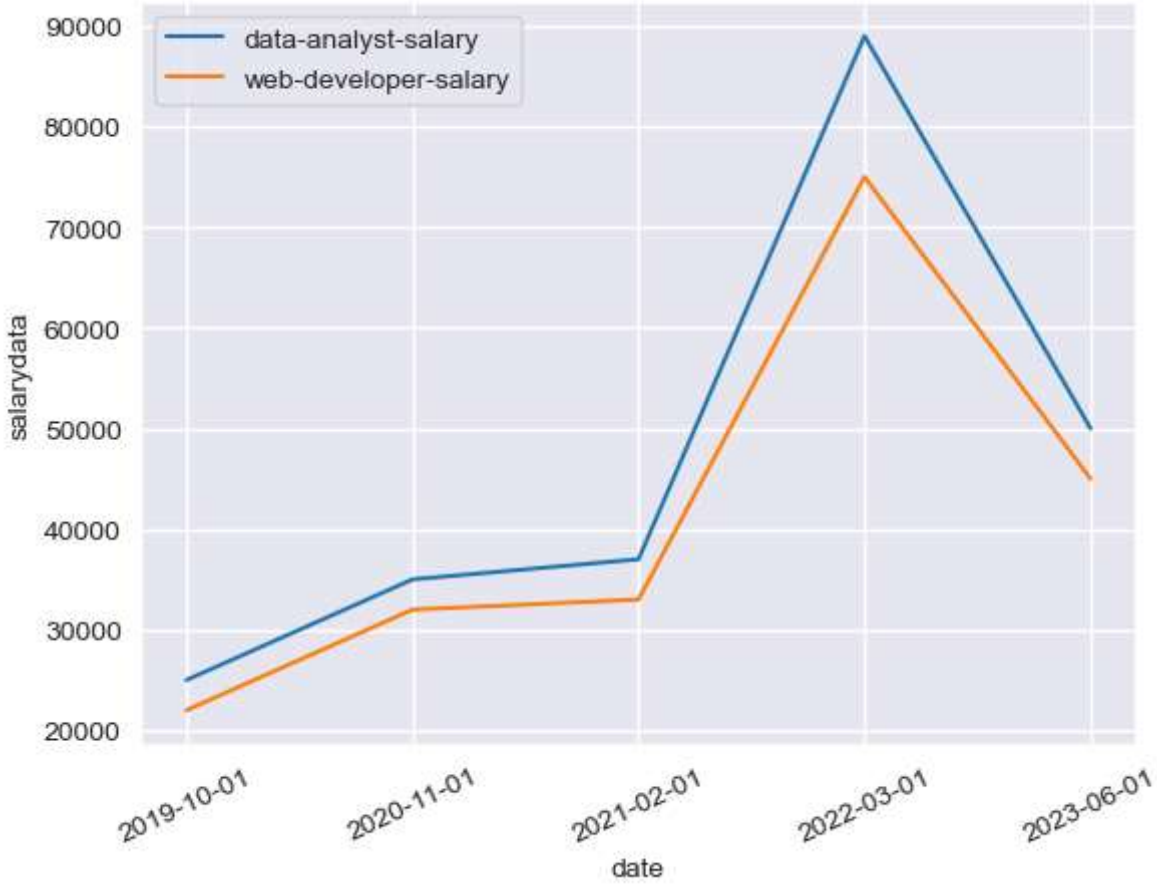
```
   salary        date
0   25000  2019-10-01
1   35000  2020-11-01
2   37000  2021-02-01
3   89000  2022-03-01
4   50000  2023-06-01
```



In [114…

```
# we can create the multiple line plots in the timeseries
# now we learn time-series means the data will be change over the time
# first we create a demo dataframe to visualize effectively

employee = {
    "salarydata":[25000,35000,37000,89000,50000],
    "date" : ['2019-10-01','2020-11-01','2021-02-01','2022-03-01','2023-06-01'],
    "salaryweb":[22000,32000,33000,75000,45000],
}

dataframe = pd.DataFrame(employee)
print(dataframe)

# lets visualize the timeseries plot

sns.lineplot(x="date",y="salarydata",data=dataframe,label="data-analyst-salary")
sns.lineplot(x="date",y="salaryweb",data=dataframe,label="web-developer-salary")
# rotation is matplot library so plt.show() is required
plt.xticks(rotation=25)# here the rotation of xticks for better visualization and r
plt.show()
```

```
   salarydata        date  salaryweb
0       25000  2019-10-01      22000
1       35000  2020-11-01      32000
2       37000  2021-02-01      33000
3       89000  2022-03-01      75000
4       50000  2023-06-01      45000
```



In [120…

```
# make a time-series lineplot using rolling average concept

petrol_price = {
    "date":['1959-01-01','1960-02-01','1961-03-01','1962-04-01',
       '1963-05-01','1964-06-01','1965-07-01','1966-08-01',
       '1967-09-01','1968-10-01'],
    "rate" : [25,26,27,28,29,30,31,32,33,35]
}

dataframe = pd.DataFrame(petrol_price)
print(dataframe)

# lets make a rolling avg plot for better avg values.
# lets make a new column to store the rolling avg values.
dataframe["rolling-average-price"] = dataframe.rate.rolling(5).mean()
print(dataframe)
```

```
         date  rate
0  1959-01-01    25
1  1960-02-01    26
2  1961-03-01    27
3  1962-04-01    28
4  1963-05-01    29
5  1964-06-01    30
6  1965-07-01    31
7  1966-08-01    32
8  1967-09-01    33
9  1968-10-01    35
         date  rate  rolling-average-price
0  1959-01-01    25                    NaN
1  1960-02-01    26                    NaN
2  1961-03-01    27                    NaN
3  1962-04-01    28                    NaN
4  1963-05-01    29                   27.0
5  1964-06-01    30                   28.0
6  1965-07-01    31                   29.0
7  1966-08-01    32                   30.0
8  1967-09-01    33                   31.0
9  1968-10-01    35                   32.2
```

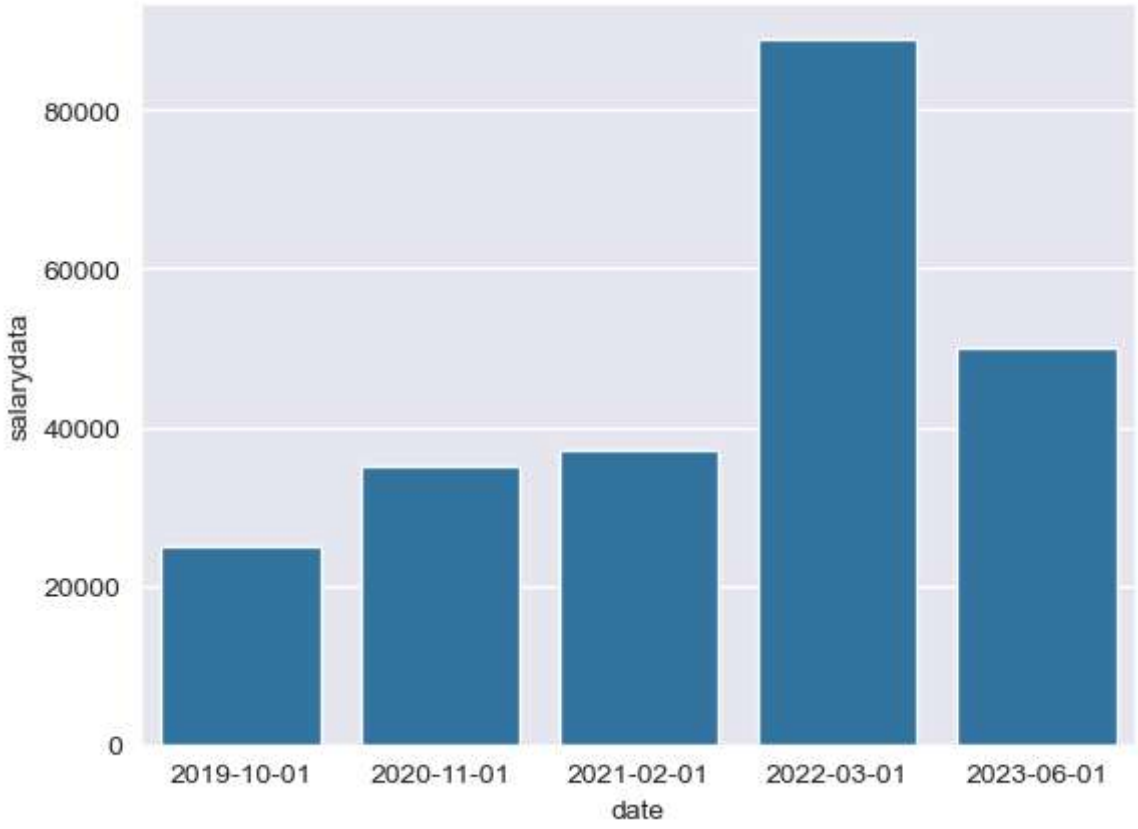
In [128…

```
# lets create the barplot using the sns.barplot()
employee = {
    "salarydata":[25000,35000,37000,89000,50000],
    "date" : ['2019-10-01','2020-11-01','2021-02-01','2022-03-01','2023-06-01'],
    "salaryweb":[22000,32000,33000,75000,45000],
}

dataframe = pd.DataFrame(employee)
print(dataframe)

sns.barplot(x="date",y="salarydata",data=dataframe)
```

```
   salarydata        date  salaryweb
0       25000  2019-10-01      22000
1       35000  2020-11-01      32000
2       37000  2021-02-01      33000
3       89000  2022-03-01      75000
4       50000  2023-06-01      45000
```

Out[128…

```
<Axes: xlabel='date', ylabel='salarydata'>
```

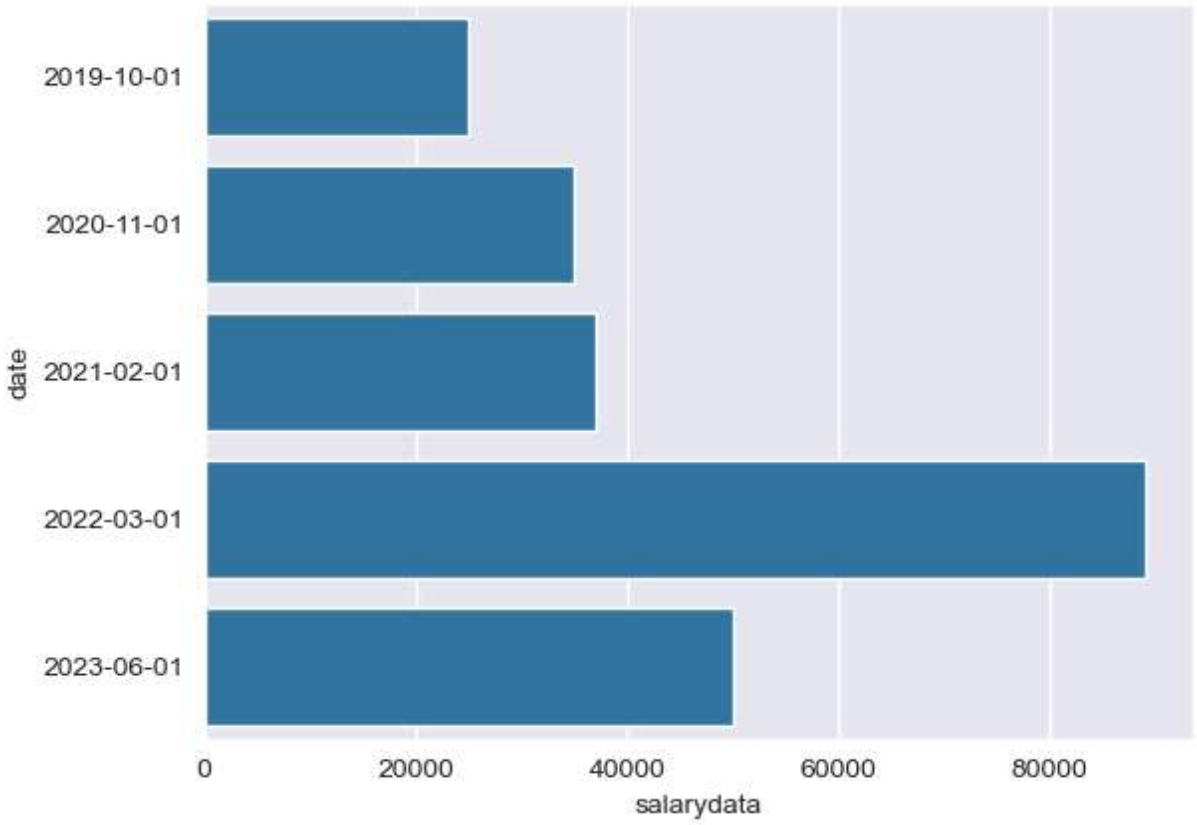


In [130…

```
# for horizontal bar-plot just change the data on x & y axes
# lets create the barplot using the sns.barplot()
employee = {
    "salarydata":[25000,35000,37000,89000,50000],
    "date" : ['2019-10-01','2020-11-01','2021-02-01','2022-03-01','2023-06-01'],
    "salaryweb":[22000,32000,33000,75000,45000],
}

dataframe = pd.DataFrame(employee)
print(dataframe)

sns.barplot(y="date",x="salarydata",data=dataframe)
```

```
   salarydata        date  salaryweb
0       25000  2019-10-01      22000
1       35000  2020-11-01      32000
2       37000  2021-02-01      33000
3       89000  2022-03-01      75000
4       50000  2023-06-01      45000
```

Out[130… <Axes: xlabel='salarydata', ylabel='date'>

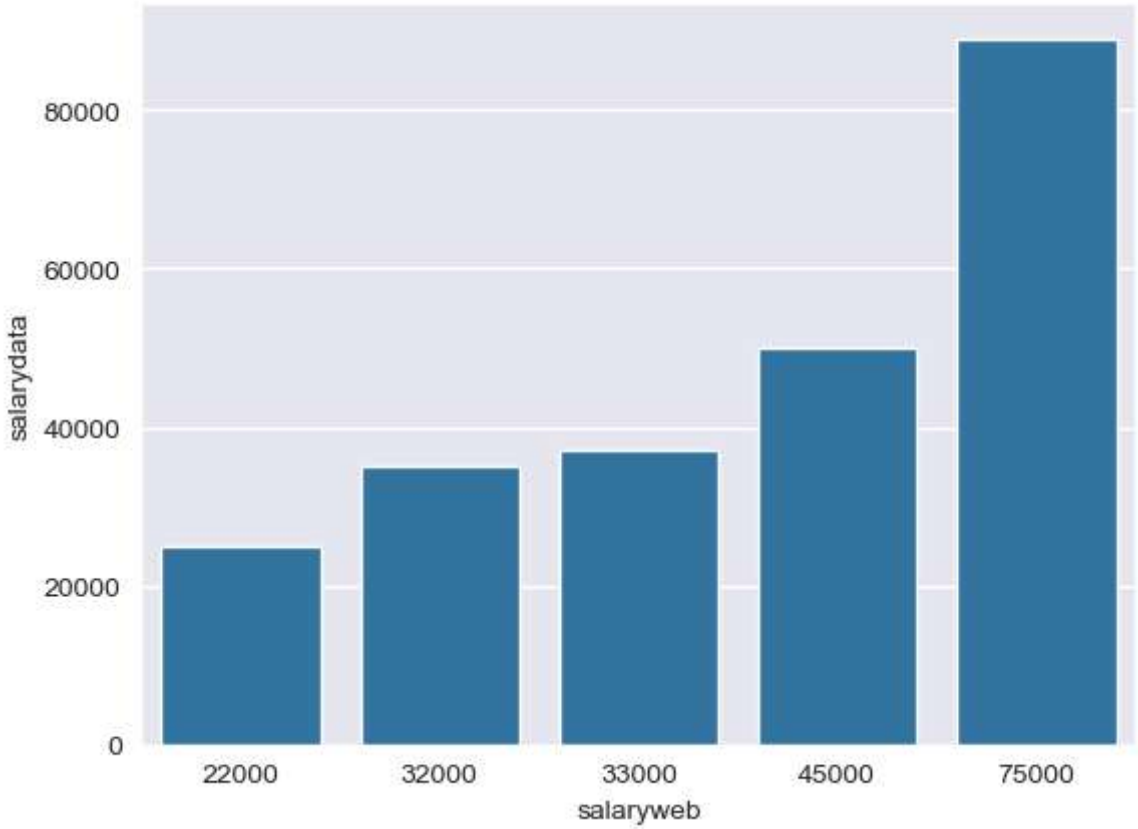


In [125…

```
# plot the bar in the given order
# lets create the barplot using the sns.barplot()
employee = {
    "salarydata":[25000,35000,37000,89000,50000],
    "date" : ['2019-10-01','2020-11-01','2021-02-01','2022-03-01','2023-06-01'],
    "salaryweb":[22000,32000,33000,75000,45000],
    "orders":["fresher","jr.data-analyst","sr.data-analyst","head","boss"]
}

dataframe = pd.DataFrame(employee)
print(dataframe)

sns.barplot(x="salaryweb",y="salarydata",data=dataframe)
```

```
   salarydata        date  salaryweb           orders
0       25000  2019-10-01      22000          fresher
1       35000  2020-11-01      32000  jr.data-analyst
2       37000  2021-02-01      33000  sr.data-analyst
3       89000  2022-03-01      75000             head
4       50000  2023-06-01      45000             boss
```

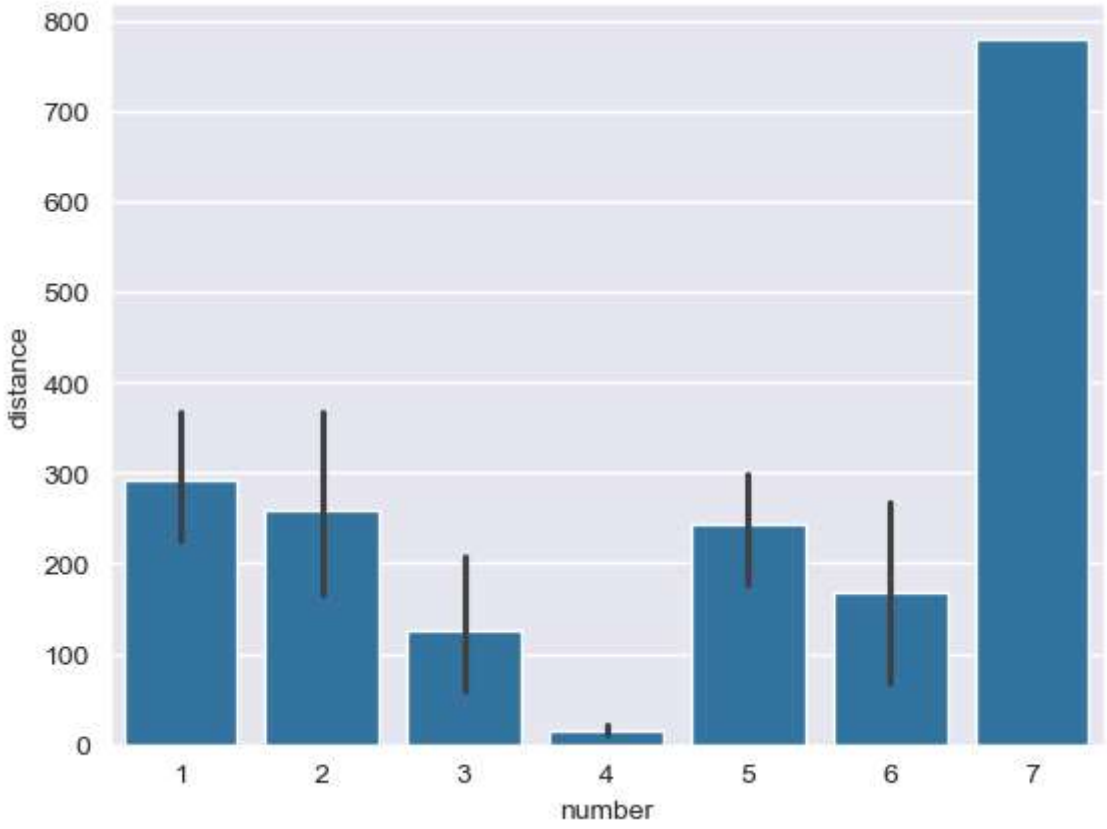
Out[125… <Axes: xlabel='salaryweb', ylabel='salarydata'>



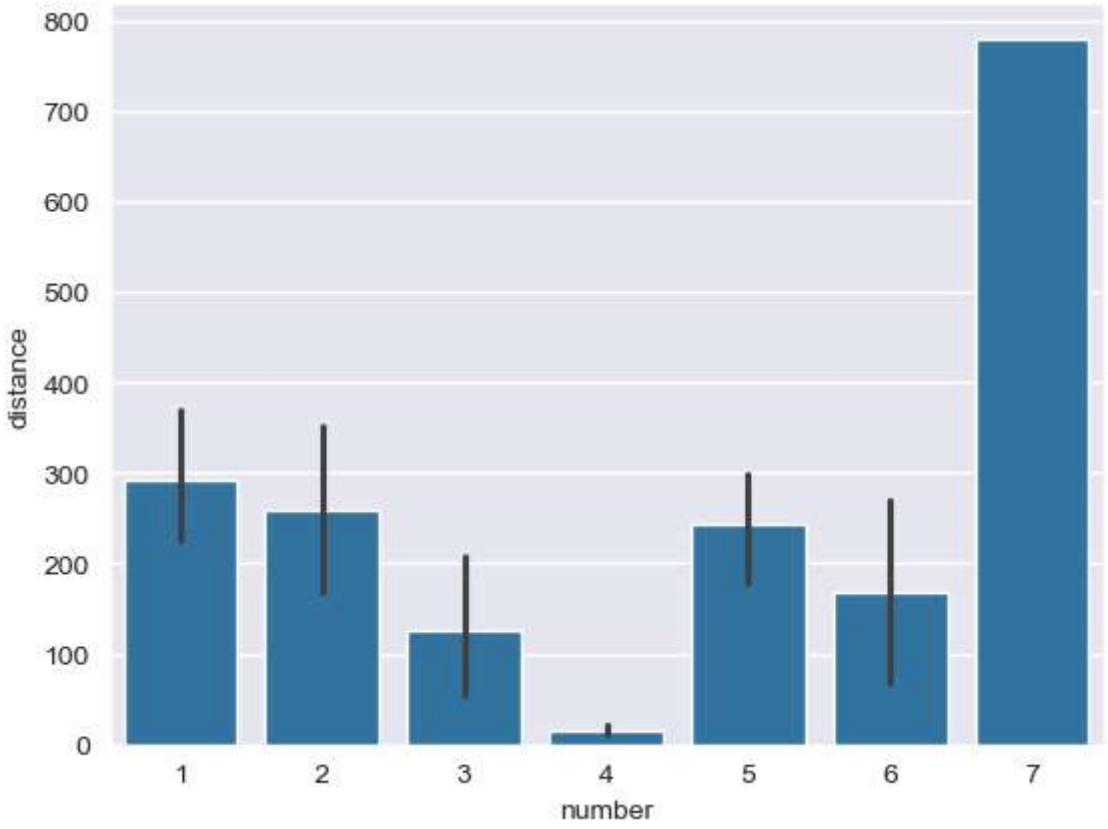
In [37]:
```
# lets create the barplot first of all load the dataset

df = sns.load_dataset("planets")
print(df)
```

```
             method  number  orbital_period   mass  distance  year
0     Radial Velocity       1       269.300000   7.10     77.40  2006
1     Radial Velocity       1       874.774000   2.21     56.95  2008
2     Radial Velocity       1       763.000000   2.60     19.84  2011
3     Radial Velocity       1       326.030000  19.40    110.62  2007
4     Radial Velocity       1       516.220000  10.50    119.47  2009
...               ...     ...              ...    ...       ...   ...
1030          Transit       1         3.941507    NaN    172.00  2006
1031          Transit       1         2.615864    NaN    148.00  2007
1032          Transit       1         3.191524    NaN    174.00  2007
1033          Transit       1         4.125083    NaN    293.00  2008
1034          Transit       1         4.187757    NaN    260.00  2008

[1035 rows x 6 columns]
```

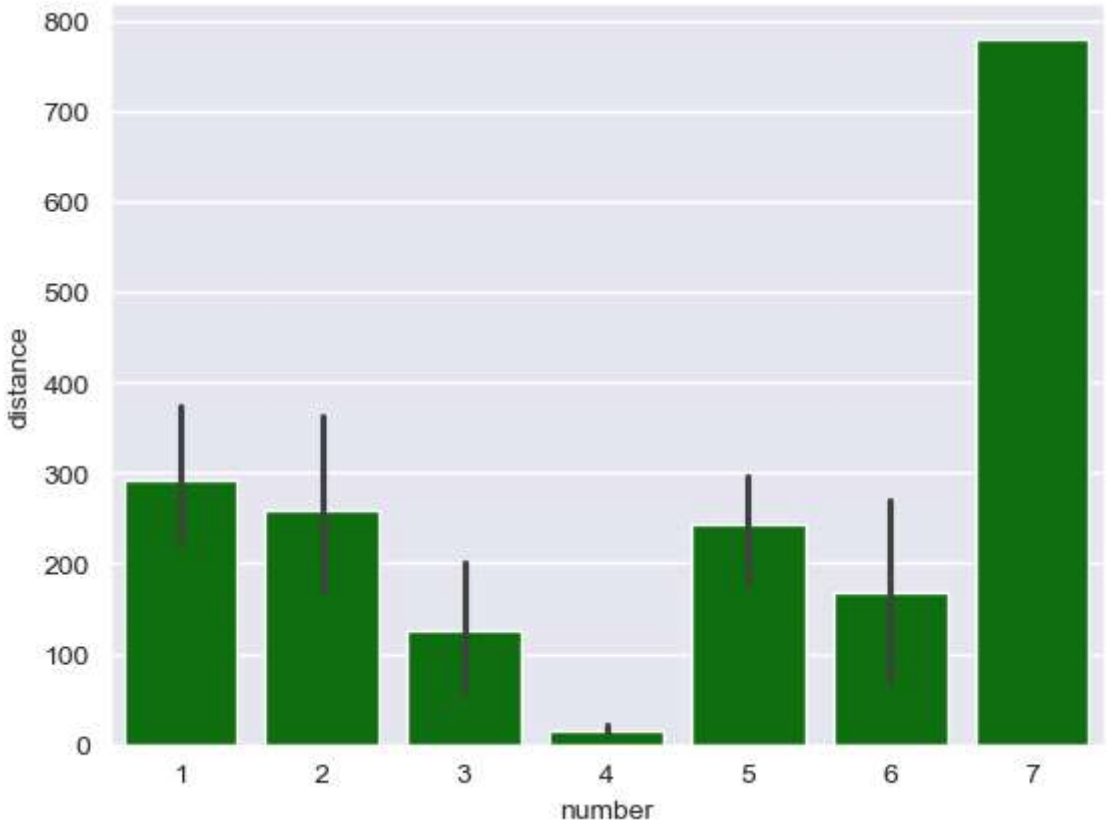
In [41]:
```
# create the bar plot

sns.barplot(x="number",y="distance",data=df)
plt.show()
```



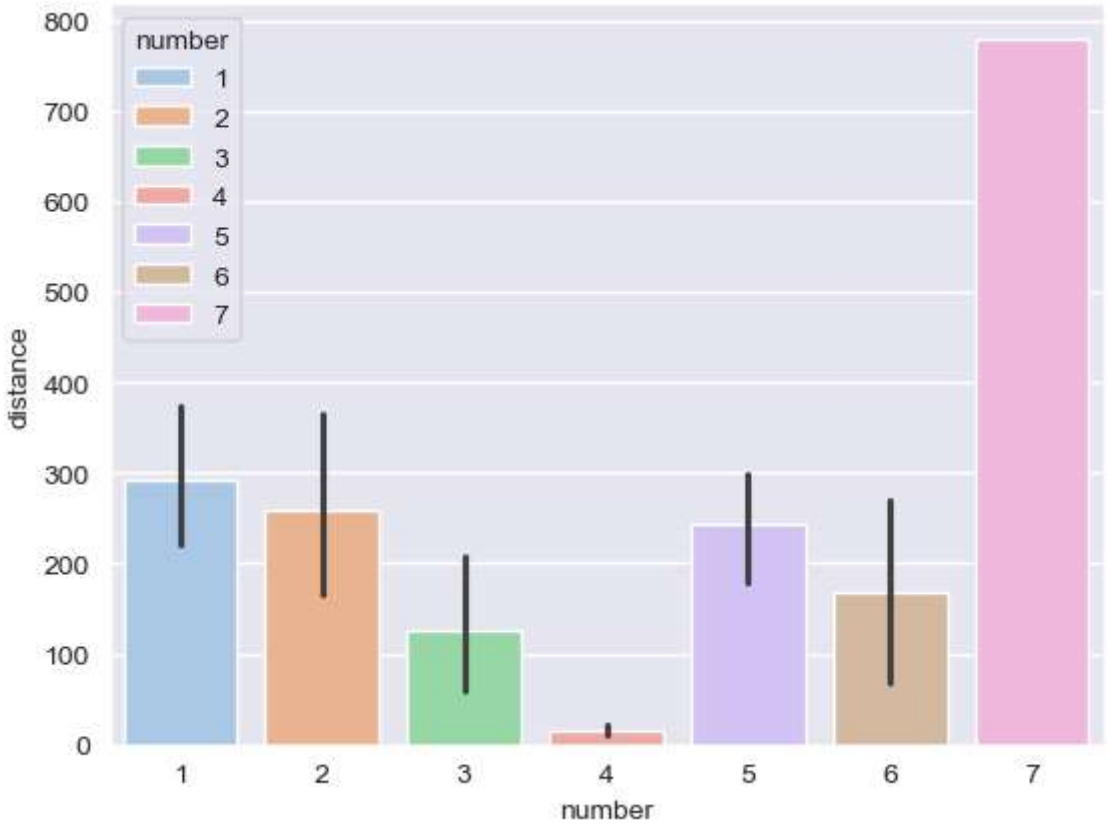
In [83]:
```
# lets differentiate with the hue
# create the bar plot

sns.barplot(x="number",y="distance",data=df)
plt.show()
```



In [71]:
```
# ploting all the bars in single color
# create the bar plot

sns.barplot(x="number",y="distance",data=df,color="green")
plt.show()
```

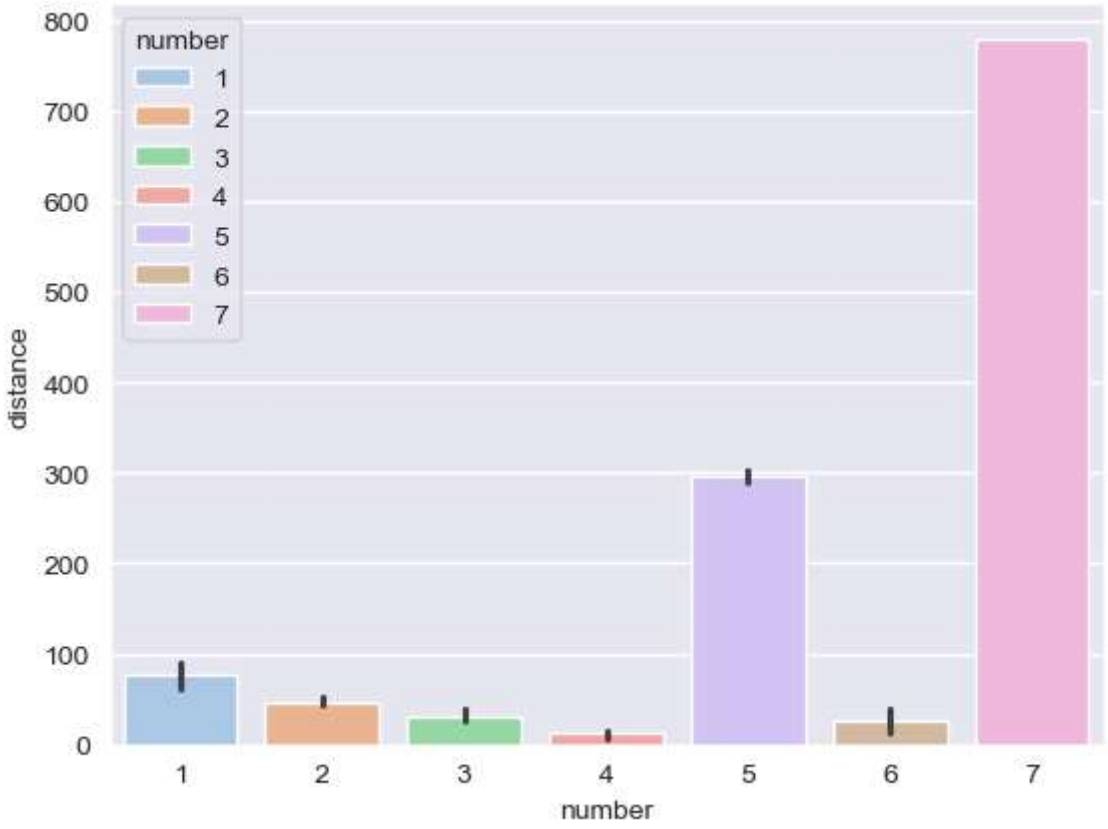


In [85]:
```
# we can create the bars with the different level of colors
# create the bar plot

sns.barplot(x="number",y="distance",data=df,palette="pastel",hue="number")
plt.show()
```



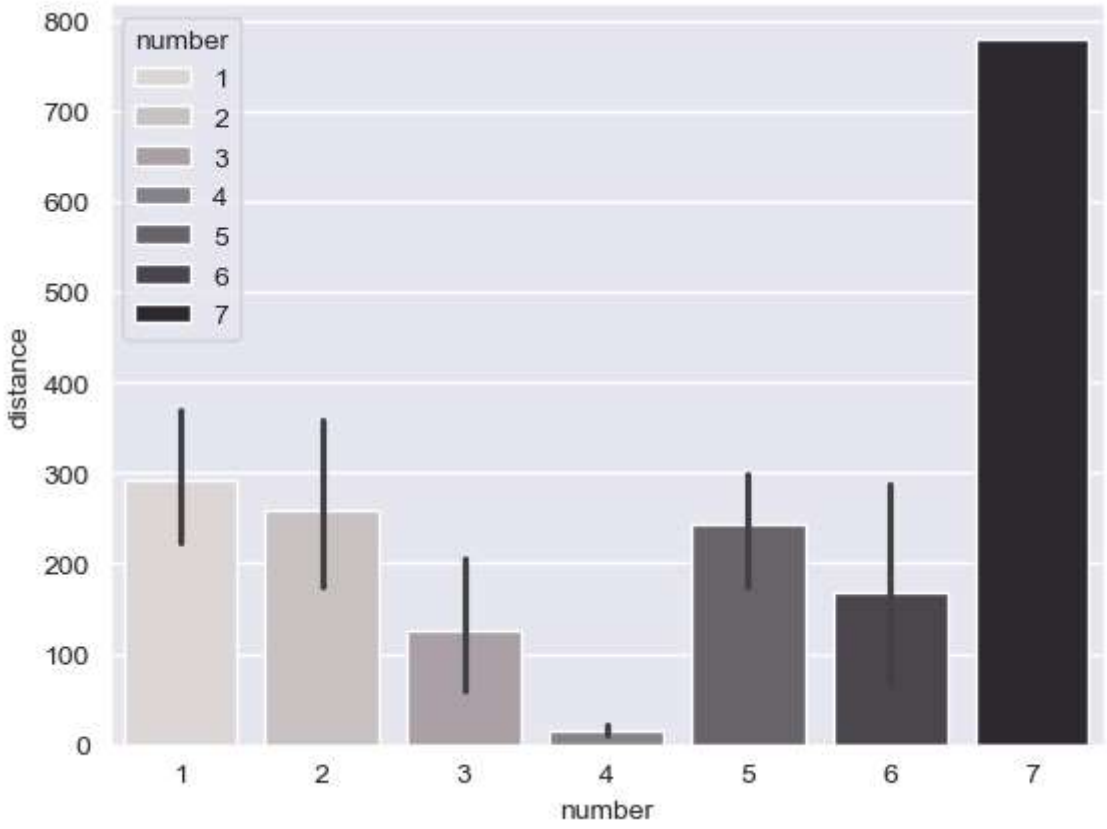
In [101…

```
# we can give the statistical function like mean and median for each category
from numpy import median

sns.barplot(x="number",y="distance",data=df,palette="pastel",hue="number",estimator
plt.show()
```



In [111…

```
# we can give the saturation parameter to looks better

# ploting all the bars in single color
# create the bar plot

sns.barplot(x="number",y="distance",data=df,hue="number",saturation=0.1)
plt.show()
```


number
1
2
3
4
5
6
7
distance
800
700
600
500
400
300
200
100
0
1
2
3
4
5
6
7
number

In [ ]: