In [61]:
```
#Paul Galvez
#DSC 680 - T301
#Milestone 4 Term Project Weeks 1-4
#Date: 9/24/2023
```

In [62]:
```
#Term Project Milestone 4
```

In [63]:
```
#Tourism is the primary source of income for the state of Hawaii and will most likely stay the
#primary source of revenue for the state's ability to run. One of the critical problems facing the touris
#industry in Hawaii is getting people to stay for extended periods. In other words, the length of time a
#stays in the state means the more money they spend and the more the state makes in the long and short te
#However, several factors affect the ability of tourists to stay longer. Factors such as the costs of
#hotels, cars, and airfare can play significant roles in the lengths of stays for people traveling to the
#business planning perspective, there are opportunities to apply data science to predict when tourists ar
#likely to stay longer in the state and where those tourists are booking those stays. The time of year he
#the data analysis because we know the traditional mindset for those in the travel and tourism industry i
#people travel in the summer and winter months. However, there could be more evidence to suggest that thi
#be the case. Using data science, we can address the problem of managing a business in the travel and tou
#industry of when to expect the most visitors and how to plan accordingly regarding resources. Being able
#and predicting the influx of crowds of tourists would allow businesses to gain actionable insights that
#clear and decisive action for the usage and deployment of resources. If we can predict visitor levels as
#the more effective we can be when they arrive, and we can deliver an elevated guest experience.
```

In [64]:
```
#importing the libraries needed:

import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
import pandas as pd
from cycler import cycler
```

In [65]:

```
#importing the data
#the data's date ranges are by year starting at 1999 - 2021

df_one = pd.read_csv("Hawaiidata.csv")
df_one
```

Out[65]:

	Group	Indicator	Units	1999	2000	2001	2002	2003	2004	2005	...	2012	2013	2014	2015	2016	2017	2018	201
0	All visitors by air	LOS Statewide	days	8.9	8.9	9.2	9.4	9.2	9.1	9.1	...	9.4	9.3	9.2	9.1	9.0	9.0	9.0	8.
1	Hotel-only visitors	LOS Statewide	days	7.4	7.2	7.5	7.7	7.7	7.3	7.3	...	7.3	7.2	7.2	7.2	7.2	7.3	7.3	7.
2	First-time visitors	LOS Statewide	days	8.1	7.9	8.4	8.5	8.5	8.2	8.3	...	8.4	8.3	8.3	8.3	8.2	8.4	8.5	8.
3	Honeymoon visitors	LOS Statewide	days	7.3	7.4	NaN	NaN	NaN	8.0	7.8	...	7.5	7.5	7.6	7.5	7.5	7.6	7.7	7.
4	All visitors by air	LOS on Oahu	days	6.4	6.6	6.8	7.0	6.9	6.9	6.9	...	7.3	7.0	6.8	6.8	6.8	6.7	6.9	6.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
117	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
118	Data is updated monthly by the Research & Econ...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
119	Source of Data: Hawaii Tourism Authority	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
120	Seasonally adjusted series are from DBEDT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal
121	Hotel performance data prior to March 2017 are...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Nal

122 rows × 26 columns

In [66]:
```
#Looking at the dataset we can see LOS = Length of Stay under the indicator column

df_one.head(10)
```

Out[66]:

	Group	Indicator	Units	1999	2000	2001	2002	2003	2004	2005	...	2012	2013	2014	2015	2016	2017	2018	2019
0	All visitors by air	LOS Statewide	days	8.9	8.9	9.2	9.4	9.2	9.1	9.1	...	9.4	9.3	9.2	9.1	9.0	9.0	9.0	8.8
1	Hotel-only visitors	LOS Statewide	days	7.4	7.2	7.5	7.7	7.7	7.3	7.3	...	7.3	7.2	7.2	7.2	7.2	7.3	7.3	7.1
2	First-time visitors	LOS Statewide	days	8.1	7.9	8.4	8.5	8.5	8.2	8.3	...	8.4	8.3	8.3	8.3	8.2	8.4	8.5	8.2
3	Honeymoon visitors	LOS Statewide	days	7.3	7.4	NaN	NaN	NaN	8.0	7.8	...	7.5	7.5	7.6	7.5	7.5	7.6	7.7	7.7
4	All visitors by air	LOS on Oahu	days	6.4	6.6	6.8	7.0	6.9	6.9	6.9	...	7.3	7.0	6.8	6.8	6.8	6.7	6.9	6.8
5	Hotel-only visitors	LOS on Oahu	days	5.6	5.7	5.9	6.2	6.1	5.9	6.0	...	6.2	6.0	5.9	5.9	5.9	6.0	6.1	6.0
6	First-time visitors	LOS on Oahu	days	5.9	5.9	6.3	6.4	6.3	6.2	6.3	...	6.5	6.3	6.1	6.2	6.2	6.2	6.4	6.3
7	Honeymoon visitors	LOS on Oahu	days	5.3	5.3	NaN	NaN	NaN	5.5	5.5	...	5.8	5.6	5.7	5.6	5.7	5.7	5.8	5.9
8	All visitors by air	LOS on Maui	days	6.7	6.8	6.9	7.2	7.3	7.5	7.5	...	8.1	8.2	8.2	8.2	8.1	8.0	8.1	7.9
9	Hotel-only visitors	LOS on Maui	days	5.6	5.4	5.7	5.9	6.3	6.2	6.3	...	6.3	6.3	6.5	6.5	6.6	6.6	6.7	6.7

10 rows × 26 columns

In [67]:
```
df_one.describe()
```

Out[67]:

	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	...	201
count	72.000000	72.000000	63.000000	63.000000	63.000000	81.000000	90.000000	81.000000	81.000000	90.000000	...	99.0000(
mean	6.155556	6.251389	6.577778	6.541270	6.376190	6.220988	6.070000	5.803704	5.776543	6.220000	...	6.43838
std	1.949568	1.964759	2.050448	2.150591	2.378649	2.403628	2.351927	2.290166	2.294955	2.422794	...	2.43728
min	2.900000	2.800000	3.000000	2.900000	2.500000	2.300000	2.100000	2.100000	2.000000	2.300000	...	2.3000(
25%	4.875000	4.875000	5.000000	4.800000	4.400000	4.100000	4.050000	3.700000	3.800000	4.025000	...	4.1000(
50%	5.750000	6.000000	6.100000	6.300000	6.300000	6.200000	6.200000	6.000000	5.900000	6.250000	...	6.5000(
75%	7.325000	7.475000	8.050000	8.000000	8.200000	8.100000	7.775000	7.400000	7.300000	7.900000	...	8.1000(
max	13.000000	12.200000	12.100000	12.600000	12.300000	12.500000	11.600000	11.000000	11.100000	12.000000	...	12.1000(

8 rows × 23 columns

In [68]:
```
df_one.tail(10)
```

Out[68]:

	Group	Indicator	Units	1999	2000	2001	2002	2003	2004	2005	...	2012	2013	2014	2015	2016	2017	2018	201
112	Family visitors	LOS on Hawaii Island	days	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	6.9	7.4	7.2	7.3	7.2	7.1	7.2	7.(
113	B&B visitors	LOS in Hilo	days	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	6.2	6.2	5.8	5.4	5.8	4.9
114	Family visitors	LOS in Hilo	days	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	3.4	3.8	3.7	3.8	3.7	3.8	3.7	3.5
115	B&B visitors	LOS in Kona	days	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	7.4	7.3	7.5	6.8	6.9	6.(
116	Family visitors	LOS in Kona	days	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	6.8	7.2	7.0	7.0	6.8	6.6	6.7	6.(
117	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
118	Data is updated monthly by the Research & Econ...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
119	Source of Data: Hawaii Tourism Authority	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
120	Seasonally adjusted series are from DBEDT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
121	Hotel performance data prior to March 2017 are...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

10 rows × 26 columns

In [69]:

```
#I dropped all NAN values that were found in the datasest

df_2 = df_one.dropna()
df_2
```

Out[69]:

	Group	Indicator	Units	1999	2000	2001	2002	2003	2004	2005	...	2012	2013	2014	2015	2016	2017	2018	2019	20
0	All visitors by air	LOS Statewide	days	8.9	8.9	9.2	9.4	9.2	9.1	9.1	...	9.4	9.3	9.2	9.1	9.0	9.0	9.0	8.8	10
1	Hotel-only visitors	LOS Statewide	days	7.4	7.2	7.5	7.7	7.7	7.3	7.3	...	7.3	7.2	7.2	7.2	7.2	7.3	7.3	7.1	7
4	All visitors by air	LOS on Oahu	days	6.4	6.6	6.8	7.0	6.9	6.9	6.9	...	7.3	7.0	6.8	6.8	6.8	6.7	6.9	6.8	8
5	Hotel-only visitors	LOS on Oahu	days	5.6	5.7	5.9	6.2	6.1	5.9	6.0	...	6.2	6.0	5.9	5.9	5.9	6.0	6.1	6.0	6
8	All visitors by air	LOS on Maui	days	6.7	6.8	6.9	7.2	7.3	7.5	7.5	...	8.1	8.2	8.2	8.2	8.1	8.0	8.1	7.9	9
9	Hotel-only visitors	LOS on Maui	days	5.6	5.4	5.7	5.9	6.3	6.2	6.3	...	6.3	6.3	6.5	6.5	6.6	6.6	6.7	6.7	7
12	All visitors by air	LOS on Molokai	days	5.0	5.1	4.3	4.6	3.7	4.3	4.2	...	4.9	4.8	4.6	4.4	4.8	4.8	4.8	4.5	6
13	Hotel-only visitors	LOS on Molokai	days	4.1	4.4	3.8	3.4	2.5	2.6	2.6	...	2.7	2.5	2.5	2.4	2.6	2.5	2.6	2.4	2
16	All visitors by air	LOS on Lanai	days	4.3	4.7	4.7	4.7	3.4	3.3	3.1	...	3.6	3.5	3.3	2.8	3.5	3.4	3.4	3.2	4
17	Hotel-only visitors	LOS on Lanai	days	4.4	4.8	4.8	4.4	3.5	3.4	3.4	...	3.7	3.8	3.2	2.4	3.5	3.3	3.6	3.3	4
20	All visitors by air	LOS on Kauai	days	6.1	6.1	6.1	6.3	6.7	6.8	6.6	...	7.5	7.6	7.7	7.6	7.7	7.5	7.5	7.4	8
21	Hotel-only visitors	LOS on Kauai	days	5.0	4.7	4.8	5.0	5.3	5.3	5.4	...	5.9	5.9	6.0	6.0	6.1	6.2	6.2	6.1	6

	Group	Indicator	Units	1999	2000	2001	2002	2003	2004	2005	...	2012	2013	2014	2015	2016	2017	2018	2019	20
24	All visitors by air	LOS on Hawaii Island	days	6.3	6.3	6.5	6.5	6.6	6.7	6.6	...	7.3	7.4	7.6	7.6	7.5	7.3	7.5	7.4	10
25	Hotel-only visitors	LOS on Hawaii Island	days	5.2	5.0	5.1	5.1	5.3	5.1	5.2	...	5.1	5.1	5.3	5.5	5.6	5.5	5.6	5.6	6
28	All visitors by air	LOS in Hilo	days	4.2	4.0	4.0	3.8	3.8	3.6	3.5	...	3.9	4.1	4.1	4.1	4.2	4.1	4.2	3.9	6
29	Hotel-only visitors	LOS in Hilo	days	3.4	3.1	3.2	2.9	2.8	2.8	2.7	...	2.5	2.6	2.8	2.9	3.0	3.2	3.2	3.0	3
32	All visitors by air	LOS in Kona	days	5.9	5.9	6.0	6.2	6.4	6.3	6.2	...	7.1	7.1	7.2	7.1	7.0	6.8	6.9	6.8	9
33	Hotel-only visitors	LOS in Kona	days	5.1	4.8	4.8	5.0	5.3	5.1	5.3	...	5.4	5.4	5.5	5.6	5.6	5.4	5.4	5.5	6
36	Condo-only visitors	LOS Statewide	days	10.5	11.0	11.4	11.4	11.4	11.2	11.3	...	11.6	11.5	11.5	11.2	11.1	10.9	10.7	10.4	13
39	Condo-only visitors	LOS on Oahu	days	7.3	8.5	9.3	9.0	9.3	9.6	10.0	...	10.5	9.7	9.5	9.3	9.3	8.8	8.9	8.6	11
42	Condo-only visitors	LOS on Maui	days	9.2	9.4	9.7	10.0	10.2	10.2	10.2	...	10.8	10.9	10.9	10.7	10.6	10.5	10.3	10.0	12
45	Condo-only visitors	LOS on Molokai	days	7.3	7.7	5.7	6.2	5.2	6.7	7.4	...	7.8	7.1	7.7	6.8	8.2	7.9	7.5	7.4	11
48	Condo-only visitors	LOS on Lanai	days	4.9	5.3	5.9	4.2	3.4	3.3	2.9	...	3.5	2.7	3.3	2.9	3.3	3.4	3.5	2.9	4
51	Condo-only visitors	LOS on Kauai	days	8.2	8.6	8.4	8.8	9.0	9.2	9.3	...	9.8	9.7	9.9	9.8	9.8	9.8	9.4	9.4	11
54	Condo-only visitors	LOS on Hawaii Island	days	8.8	9.0	9.2	9.5	9.6	9.7	9.7	...	10.5	10.4	10.7	10.6	10.3	10.0	9.9	10.0	13

	Group	Indicator	Units	1999	2000	2001	2002	2003	2004	2005	...	2012	2013	2014	2015	2016	2017	2018	2019	20
57	Condo-only visitors	LOS in Hilo	days	5.2	5.0	5.4	4.6	4.0	4.1	3.9	...	4.1	4.0	3.9	4.2	4.0	3.9	4.0	3.9	5
60	Condo-only visitors	LOS in Kona	days	8.6	8.9	9.1	9.5	9.7	9.7	9.9	...	10.6	10.4	10.8	10.6	10.3	10.0	9.7	9.8	13
64	MCI visitors	LOS Statewide	days	7.9	7.9	8.0	8.3	8.2	8.1	7.9	...	7.8	7.7	7.7	7.5	7.3	7.5	7.4	7.2	7
66	MCI visitors	LOS on Oahu	days	5.8	6.1	6.1	6.4	6.3	6.3	6.4	...	6.2	6.0	5.9	5.7	5.5	5.8	5.7	5.7	6
68	MCI visitors	LOS on Maui	days	6.3	6.3	6.6	6.8	6.7	6.6	6.5	...	7.0	7.1	7.2	7.1	7.0	7.0	7.1	6.9	7
70	MCI visitors	LOS on Molokai	days	5.1	5.1	4.0	4.8	2.8	3.7	2.6	...	3.2	3.0	3.8	2.3	3.8	3.7	4.4	3.6	4
72	MCI visitors	LOS on Lanai	days	4.3	5.1	5.0	4.4	4.1	3.8	3.5	...	4.1	4.0	3.9	2.4	4.5	3.8	4.9	3.7	4
74	MCI visitors	LOS on Kauai	days	5.7	5.5	6.1	6.0	5.9	6.1	5.8	...	6.2	6.5	7.0	6.6	6.6	6.6	6.6	6.6	7
76	MCI visitors	LOS on Hawaii Island	days	6.1	6.3	6.7	6.6	6.2	6.5	6.2	...	6.5	6.4	6.6	6.4	6.6	6.3	6.8	6.7	7
78	MCI visitors	LOS in Hilo	days	3.9	4.2	4.7	3.8	3.4	3.6	3.4	...	3.2	3.4	3.7	3.4	3.5	3.5	4.0	3.6	4
80	MCI visitors	LOS in Kona	days	5.9	6.1	6.5	6.3	6.1	6.2	6.0	...	6.5	6.4	6.6	6.4	6.6	6.3	6.4	6.5	6

36 rows × 26 columns

In [70]:
```
df_2.head(15)
```

Out[70]:

	Group	Indicator	Units	1999	2000	2001	2002	2003	2004	2005	...	2012	2013	2014	2015	2016	2017	2018	2019	202
0	All visitors by air	LOS Statewide	days	8.9	8.9	9.2	9.4	9.2	9.1	9.1	...	9.4	9.3	9.2	9.1	9.0	9.0	9.0	8.8	10
1	Hotel-only visitors	LOS Statewide	days	7.4	7.2	7.5	7.7	7.7	7.3	7.3	...	7.3	7.2	7.2	7.2	7.2	7.3	7.3	7.1	7
4	All visitors by air	LOS on Oahu	days	6.4	6.6	6.8	7.0	6.9	6.9	6.9	...	7.3	7.0	6.8	6.8	6.8	6.7	6.9	6.8	8
5	Hotel-only visitors	LOS on Oahu	days	5.6	5.7	5.9	6.2	6.1	5.9	6.0	...	6.2	6.0	5.9	5.9	5.9	6.0	6.1	6.0	6
8	All visitors by air	LOS on Maui	days	6.7	6.8	6.9	7.2	7.3	7.5	7.5	...	8.1	8.2	8.2	8.2	8.1	8.0	8.1	7.9	9
9	Hotel-only visitors	LOS on Maui	days	5.6	5.4	5.7	5.9	6.3	6.2	6.3	...	6.3	6.3	6.5	6.5	6.6	6.6	6.7	6.7	7
12	All visitors by air	LOS on Molokai	days	5.0	5.1	4.3	4.6	3.7	4.3	4.2	...	4.9	4.8	4.6	4.4	4.8	4.8	4.8	4.5	6
13	Hotel-only visitors	LOS on Molokai	days	4.1	4.4	3.8	3.4	2.5	2.6	2.6	...	2.7	2.5	2.5	2.4	2.6	2.5	2.6	2.4	2
16	All visitors by air	LOS on Lanai	days	4.3	4.7	4.7	4.7	3.4	3.3	3.1	...	3.6	3.5	3.3	2.8	3.5	3.4	3.4	3.2	4
17	Hotel-only visitors	LOS on Lanai	days	4.4	4.8	4.8	4.4	3.5	3.4	3.4	...	3.7	3.8	3.2	2.4	3.5	3.3	3.6	3.3	4
20	All visitors by air	LOS on Kauai	days	6.1	6.1	6.1	6.3	6.7	6.8	6.6	...	7.5	7.6	7.7	7.6	7.7	7.5	7.5	7.4	8
21	Hotel-only visitors	LOS on Kauai	days	5.0	4.7	4.8	5.0	5.3	5.3	5.4	...	5.9	5.9	6.0	6.0	6.1	6.2	6.2	6.1	6

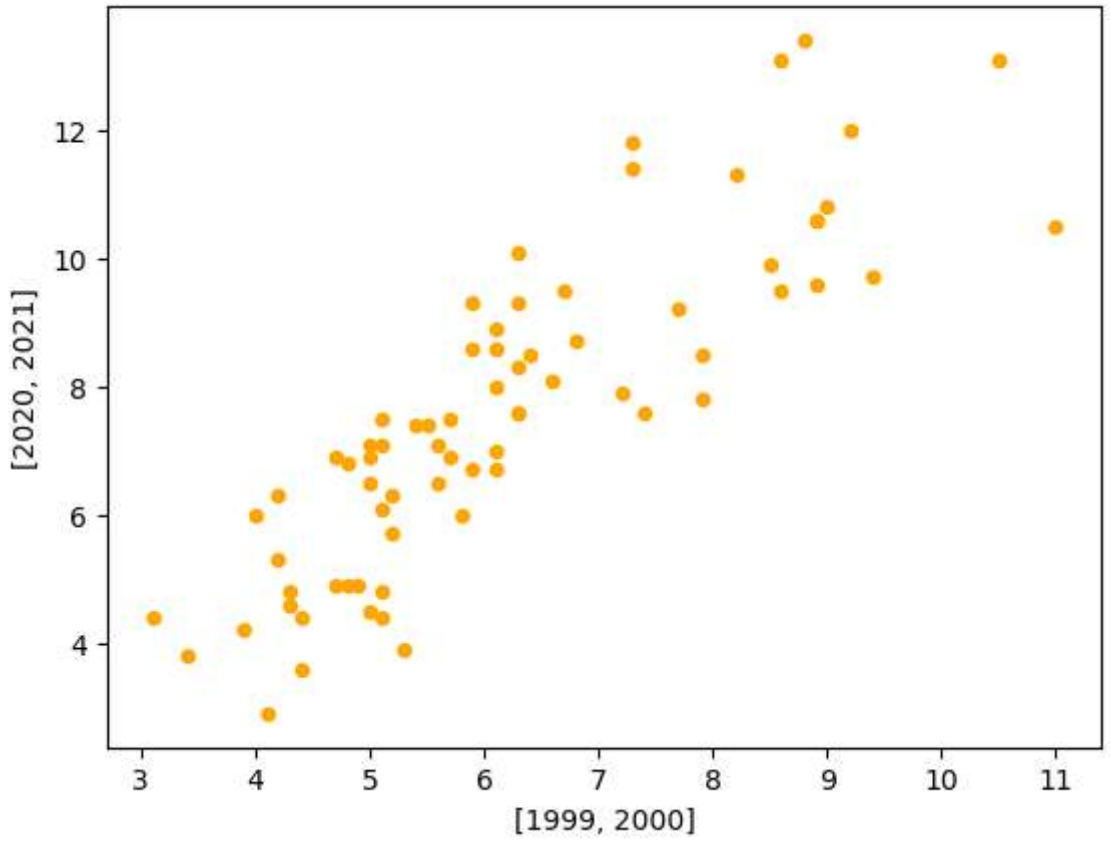
	Group	Indicator	Units	1999	2000	2001	2002	2003	2004	2005	...	2012	2013	2014	2015	2016	2017	2018	2019	202
24	All visitors by air	LOS on Hawaii Island	days	6.3	6.3	6.5	6.5	6.6	6.7	6.6	...	7.3	7.4	7.6	7.6	7.5	7.3	7.5	7.4	10
25	Hotel-only visitors	LOS on Hawaii Island	days	5.2	5.0	5.1	5.1	5.3	5.1	5.2	...	5.1	5.1	5.3	5.5	5.6	5.5	5.6	5.6	6
28	All visitors by air	LOS in Hilo	days	4.2	4.0	4.0	3.8	3.8	3.6	3.5	...	3.9	4.1	4.1	4.1	4.2	4.1	4.2	3.9	6

15 rows × 26 columns

In [71]:
```
import hvplot.pandas
```

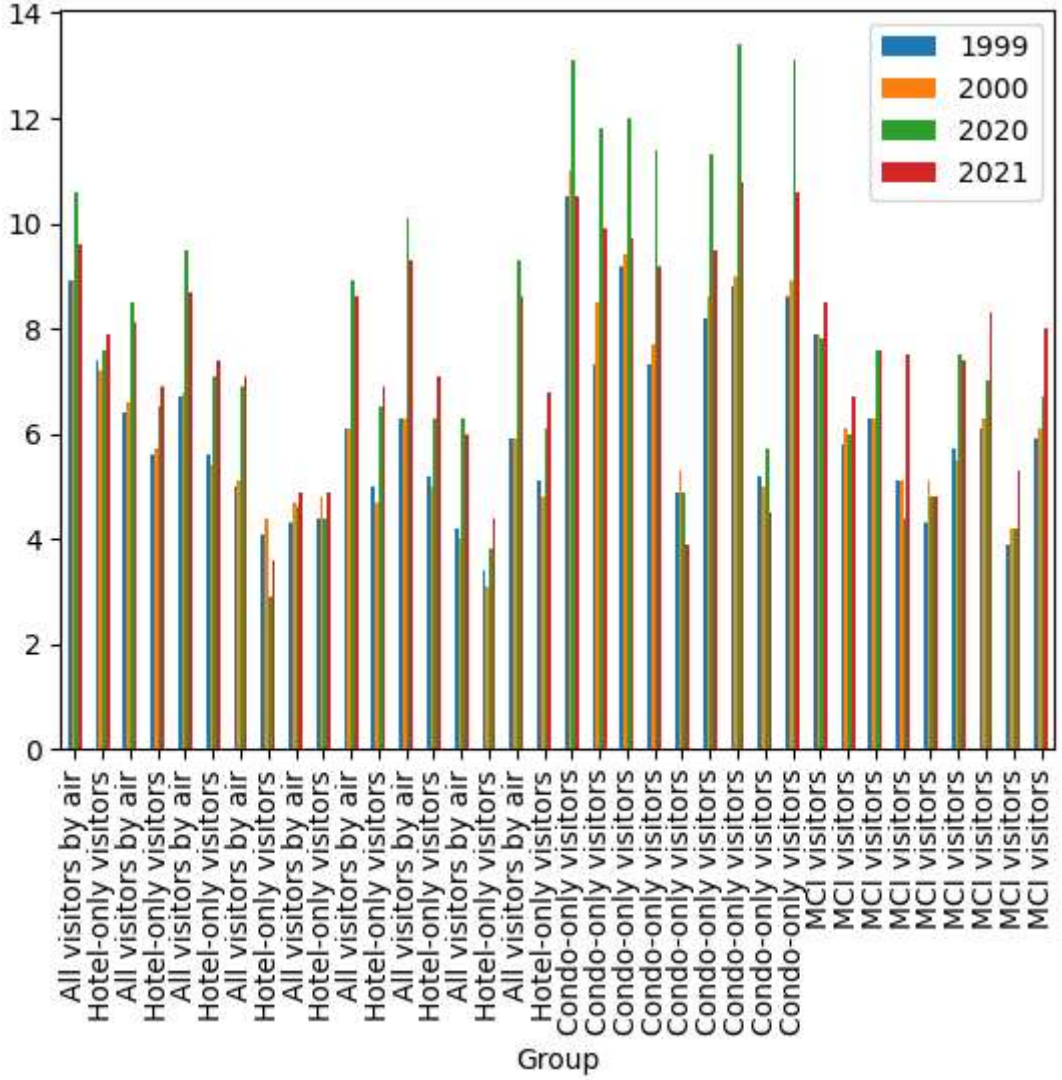
In [72]:
```
#the scatter plot below shows the first and last two years of data 1999 is on the x axis while the y axis
#the number of days are shown per data point. We can see there are similar numbers between the years.
#There aren't outliers in terms of a data point exceeding the number of days by a significant amount

df_2.plot.scatter(x=['1999', '2000'], color='orange', y=['2020', '2021'])
```

Out[72]: <Axes: xlabel='[1999, 2000]', ylabel='[2020, 2021]'>



In [73]:
```
#the bar graph below shows the amount of days by the variable 'group'. Within in 'group' column we can se
#rows that are visitors by air, hotel only visitors, condo only visitors, and MCI visitors. Again, I want
#first and lat two years of data. In almost all rows of variables, we can see 2020 was a a good year for
#was also the year of the pandemic. So one would think 2020 should have been a much slower year. And the
#However, we can assume that people that did travel to Hawaii during those years, were most likely stayin
#heavy travel restrictions that dictated the affected the length of stays for guests.

df_2.plot(x='Group', y=['1999', '2000', '2020', '2021'], kind='bar')
```

Out[73]:
```
<Axes: xlabel='Group'>
```


14
12
10
8
6
4
2
0
1999
2000
2020
2021
All visitors by air
Hotel-only visitors
All visitors by air
Hotel-only visitors
All visitors by air
Hotel-only visitors
All visitors by air
Hotel-only visitors
All visitors by air
Hotel-only visitors
All visitors by air
Hotel-only visitors
All visitors by air
Hotel-only visitors
All visitors by air
Hotel-only visitors
Condo-only visitors
Condo-only visitors
Condo-only visitors
Condo-only visitors
Condo-only visitors
Condo-only visitors
Condo-only visitors
Condo-only visitors
Condo-only visitors
MCI visitors
MCI visitors
MCI visitors
MCI visitors
MCI visitors
MCI visitors
MCI visitors
MCI visitors
MCI visitors
Group

In [74]:
```
#I wanted to find the max number of each column or year

max_x = df_2.loc[df_2['1999'].idxmax()]
max_x
```

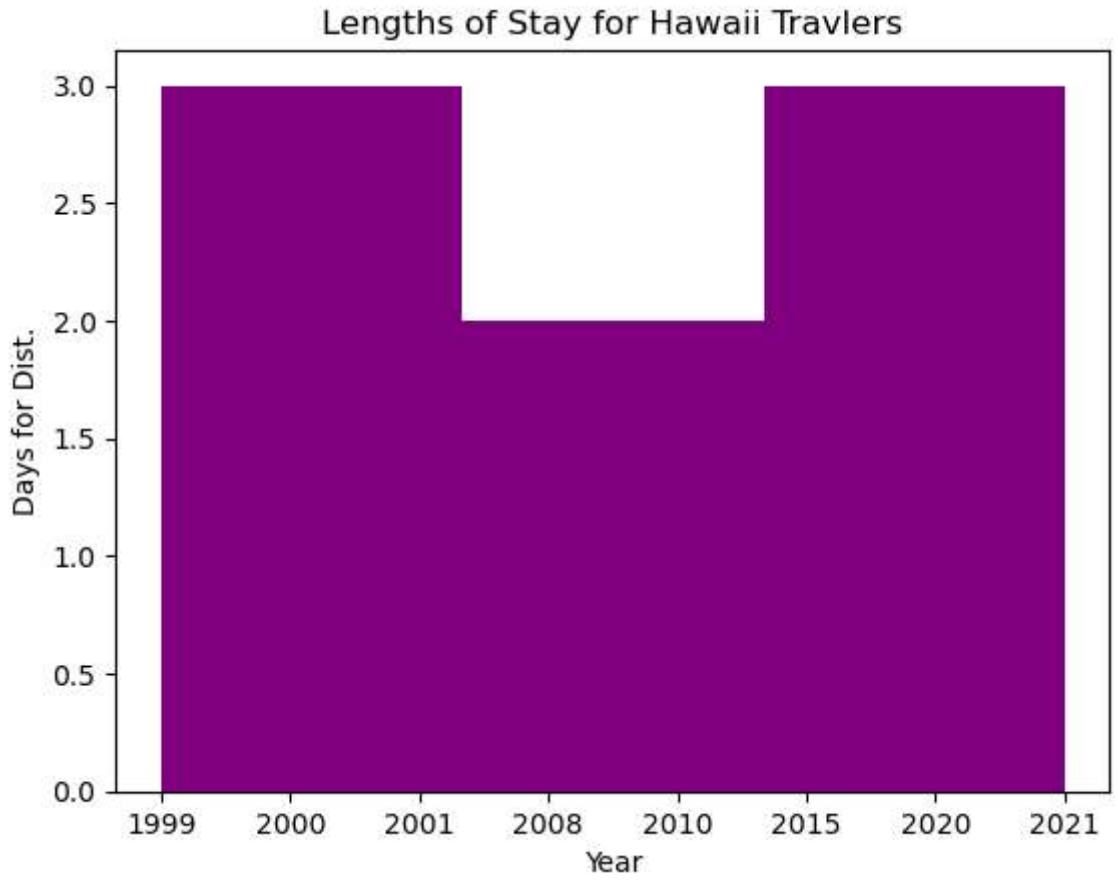
Out[74]:
```
Group        Condo-only visitors
Indicator          LOS Statewide
Units                        days
1999                         10.5
2000                         11.0
2001                         11.4
2002                         11.4
2003                         11.4
2004                         11.2
2005                         11.3
2006                         11.0
2007                         11.1
2008                         11.6
2009                         11.6
2010                         11.5
2011                         11.5
2012                         11.6
2013                         11.5
2014                         11.5
2015                         11.2
2016                         11.1
2017                         10.9
2018                         10.7
2019                         10.4
2020                         13.1
2021                         10.5
Name: 36, dtype: object
```

In [75]:
```
#create lib. for max vlaues found and comparing the same four years as the previous graphs with some adde

my_lib = {'1999':10.5, '2000':11.0, '2001':11.4, '2008':11.6, '2010':11.5, '2015':11.2, '2020':13.1, '202
my_lib
```

Out[75]:
```
{'1999': 10.5,
 '2000': 11.0,
 '2001': 11.4,
 '2008': 11.6,
 '2010': 11.5,
 '2015': 11.2,
 '2020': 13.1,
 '2021': 10.5}
```

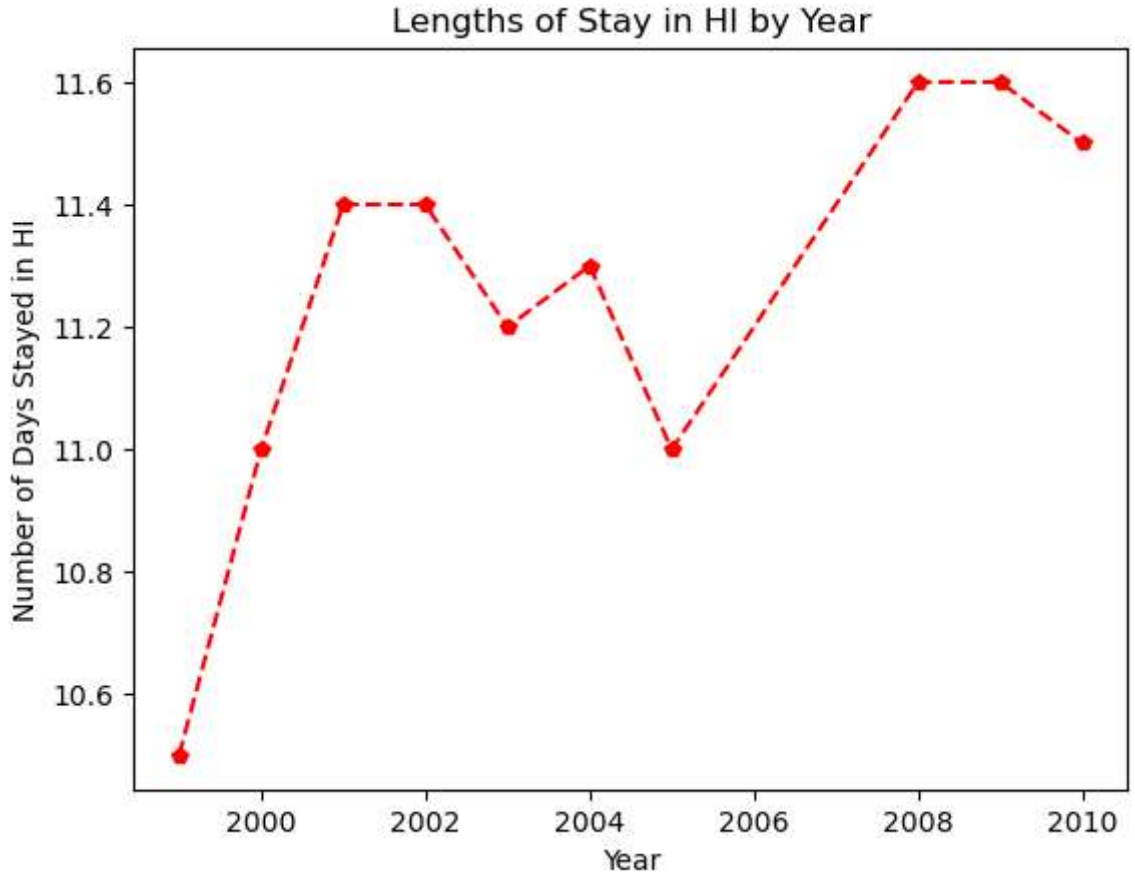
In [76]:
```
#the histogram shows there was some slow down from 2008 to around 2013/2014. We know there was a depressi
#years and the slow down in spending on amenities such as travel. We can see there are still people trave
#those slow economic periods, there is an obvious difference in length of stay in days in Hawaii.

plt.hist(my_lib, 3, color='purple')
plt.xlabel('Year')
plt.ylabel('Days for Dist.')
plt.title('Lengths of Stay for Hawaii Travlers')
plt.show
```

Out[76]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



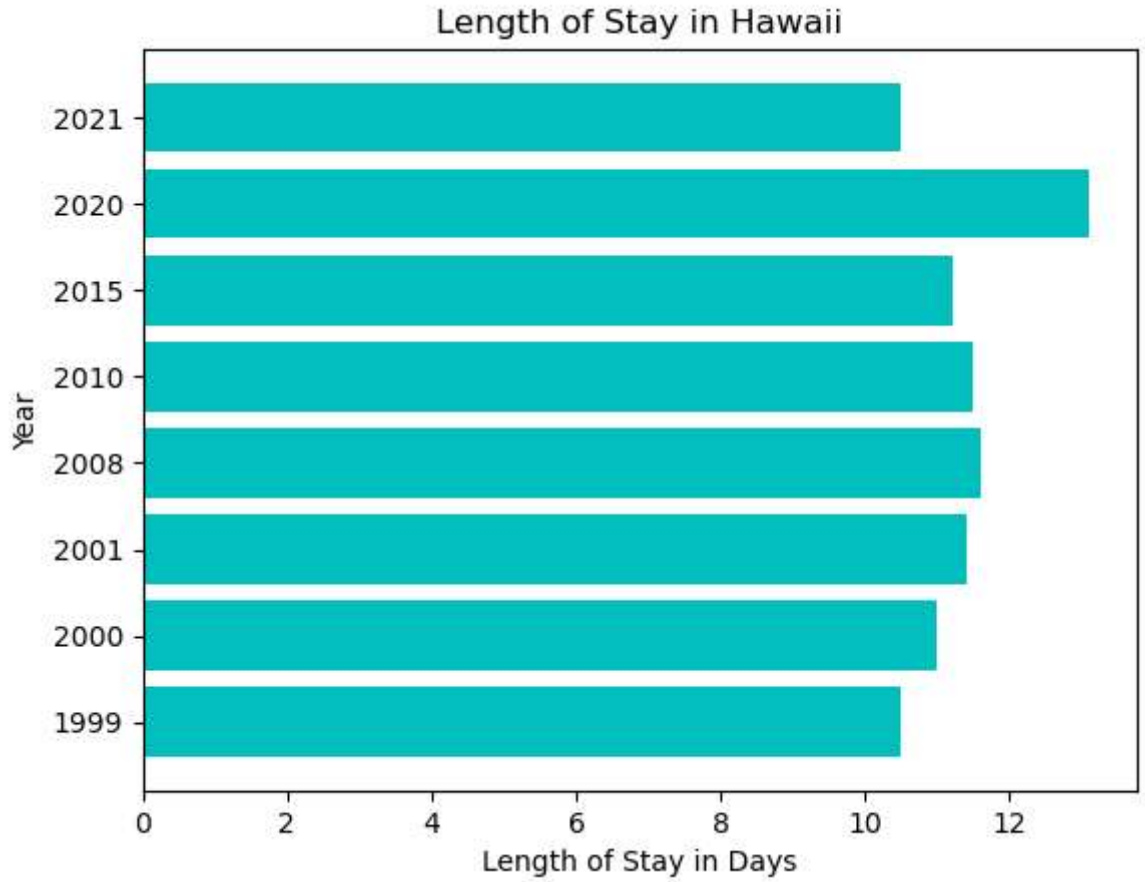
In [77]:
```
#the line plot below is laid out in the format that leds the data into the 2008 economic downturn. We can
#says the economic downturn does impact the LOS for visitors. The question the data does ask is how does
#country affects HI travelers. Also,does the number of visitors and LOS that much impacted by internation

fig, ax = plt.subplots()
ax.plot([1999, 2000, 2001, 2002, 2003, 2004, 2005, 2008, 2009, 2010], [10.5, 11.0, 11.4, 11.4, 11.2, 11.3
                                                                       11.0, 11.6, 11.6, 11.5], '--pr')
plt.xlabel('Year')
plt.ylabel('Number of Days Stayed in HI')
plt.title('Lengths of Stay in HI by Year')
```

Out[77]:
```
Text(0.5, 1.0, 'Lengths of Stay in HI by Year')
```



In [78]:
```
#the side bars are easier to read for the sake of the data being able to be broken down to avoid confusio

fig, ax = plt.subplots()
data = my_lib
yr_name = list(data.values())
days_tik = list(data.keys())

ax.barh(days_tik, yr_name, color='c')
plt.xlabel('Length of Stay in Days')
plt.ylabel('Year')
plt.title('Length of Stay in Hawaii')
```

Out[78]: Text(0.5, 1.0, 'Length of Stay in Hawaii')



In [79]:
```
min_x = df_2.loc[df_2['1999'].idxmin()]
min_x
```

Out[79]:
```
Group        Hotel-only visitors
Indicator             LOS in Hilo
Units                        days
1999                          3.4
2000                          3.1
2001                          3.2
2002                          2.9
2003                          2.8
2004                          2.8
2005                          2.7
2006                          2.7
2007                          2.5
2008                          2.6
2009                          2.6
2010                          2.6
2011                          2.8
2012                          2.5
2013                          2.6
2014                          2.8
2015                          2.9
2016                          3.0
2017                          3.2
2018                          3.2
2019                          3.0
2020                          3.8
2021                          4.4
Name: 29, dtype: object
```

In [80]:
```
#The final analysis of the graphs would suggest the max length of stays are generally steady. However, th
#oustanding questions regarding the type of visitors that travel to Hawaii. There are a number of indicat
#isn't a significant impact in terms of economic down turns. Also, the grpahs for the min or smaller numb
#be interesting to look at as well to compare max numbers.
```

In [81]:
```
#Looking for duplicate data. I already dropped the NAN values during milestone 1 to help better plot the

df_2.duplicated
```

Out[81]:
```
<bound method DataFrame.duplicated of                  Group            Indicator Units  1999  2000
2001  2002  \
0   All visitors by air         LOS Statewide  days   8.9   8.9   9.2   9.4
1   Hotel-only visitors         LOS Statewide  days   7.4   7.2   7.5   7.7
4   All visitors by air           LOS on Oahu  days   6.4   6.6   6.8   7.0
5   Hotel-only visitors           LOS on Oahu  days   5.6   5.7   5.9   6.2
8   All visitors by air           LOS on Maui  days   6.7   6.8   6.9   7.2
9   Hotel-only visitors           LOS on Maui  days   5.6   5.4   5.7   5.9
12  All visitors by air        LOS on Molokai  days   5.0   5.1   4.3   4.6
13  Hotel-only visitors        LOS on Molokai  days   4.1   4.4   3.8   3.4
16  All visitors by air          LOS on Lanai  days   4.3   4.7   4.7   4.7
17  Hotel-only visitors          LOS on Lanai  days   4.4   4.8   4.8   4.4
20  All visitors by air          LOS on Kauai  days   6.1   6.1   6.1   6.3
21  Hotel-only visitors          LOS on Kauai  days   5.0   4.7   4.8   5.0
24  All visitors by air  LOS on Hawaii Island  days   6.3   6.3   6.5   6.5
25  Hotel-only visitors  LOS on Hawaii Island  days   5.2   5.0   5.1   5.1
28  All visitors by air           LOS in Hilo  days   4.2   4.0   4.0   3.8
29  Hotel-only visitors           LOS in Hilo  days   3.4   3.1   3.2   2.9
32  All visitors by air           LOS in Kona  days   5.9   5.9   6.0   6.2
```

In [82]:
```
#dropping duplicates from the data
#it is more helpful to the proejct to have the data that tells the most accurate story and information

df_2 = df_2.drop_duplicates()
df_2
```

Out[82]:

	Group	Indicator	Units	1999	2000	2001	2002	2003	2004	2005	...	2012	2013	2014	2015	2016	2017	2018	2019
0	All visitors by air	LOS Statewide	days	8.9	8.9	9.2	9.4	9.2	9.1	9.1	...	9.4	9.3	9.2	9.1	9.0	9.0	9.0	8.8
1	Hotel-only visitors	LOS Statewide	days	7.4	7.2	7.5	7.7	7.7	7.3	7.3	...	7.3	7.2	7.2	7.2	7.2	7.3	7.3	7.1
4	All visitors by air	LOS on Oahu	days	6.4	6.6	6.8	7.0	6.9	6.9	6.9	...	7.3	7.0	6.8	6.8	6.8	6.7	6.9	6.8
5	Hotel-only visitors	LOS on Oahu	days	5.6	5.7	5.9	6.2	6.1	5.9	6.0	...	6.2	6.0	5.9	5.9	5.9	6.0	6.1	6.0
8	All visitors by air	LOS on Maui	days	6.7	6.8	6.9	7.2	7.3	7.5	7.5	...	8.1	8.2	8.2	8.2	8.1	8.0	8.1	7.9

In [83]:
```
import os
import requests
from bs4 import BeautifulSoup
```

In [84]:
```
#rename several columns for better reabability

df_2 = df_2.rename(columns={'Group': 'Visitor Type', 'Indicator': 'Location Of Stay', 'Units': 'Length of
df_2
```

Out[84]:

	Visitor Type	Location Of Stay	Length of Stay In	1999	2000	2001	2002	2003	2004	2005	...	2012	2013	2014	2015	2016	2017	2018	201
0	All visitors by air	LOS Statewide	days	8.9	8.9	9.2	9.4	9.2	9.1	9.1	...	9.4	9.3	9.2	9.1	9.0	9.0	9.0	8.
1	Hotel-only visitors	LOS Statewide	days	7.4	7.2	7.5	7.7	7.7	7.3	7.3	...	7.3	7.2	7.2	7.2	7.2	7.3	7.3	7.
4	All visitors by air	LOS on Oahu	days	6.4	6.6	6.8	7.0	6.9	6.9	6.9	...	7.3	7.0	6.8	6.8	6.8	6.7	6.9	6.
5	Hotel-only visitors	LOS on Oahu	days	5.6	5.7	5.9	6.2	6.1	5.9	6.0	...	6.2	6.0	5.9	5.9	5.9	6.0	6.1	6.
	All	LOS on																	

In [85]:
```
df_2.groupby('1999').aggregate('mean')
```

```
C:\Users\paul_\AppData\Local\Temp\ipykernel_11596\2621612885.py:1: FutureWarning: The default value o
f numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default
to False. Either specify numeric_only or select only columns which should be valid for the function.
  df_2.groupby('1999').aggregate('mean')
```

Out[85]:

1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	...	2012	2013	2014	2015	2016	2017	2018
3.4	3.10	3.20	2.90	2.80	2.80	2.70	2.70	2.50	2.60	2.60	...	2.50	2.60	2.80	2.90	3.00	3.20	3.20
3.9	4.20	4.70	3.80	3.40	3.60	3.40	3.30	3.00	3.10	3.30	...	3.20	3.40	3.70	3.40	3.50	3.50	4.00
4.1	4.40	3.80	3.40	2.50	2.60	2.60	2.60	2.70	2.40	2.70	...	2.70	2.50	2.50	2.40	2.60	2.50	2.60
4.2	4.00	4.00	3.80	3.80	3.60	3.50	3.20	3.10	3.60	3.70	...	3.90	4.10	4.10	4.10	4.20	4.10	4.20
4.3	4.90	4.85	4.55	3.75	3.55	3.30	3.45	3.80	3.65	4.05	...	3.85	3.75	3.60	2.60	4.00	3.60	4.15
4.4	4.80	4.80	4.40	3.50	3.40	3.40	3.60	3.90	3.90	4.10	...	3.70	3.80	3.20	2.40	3.50	3.30	3.60

In [86]:
```
df_2.groupby('2021').aggregate('mean')
```

```
C:\Users\paul_\AppData\Local\Temp\ipykernel_11596\1889895722.py:1: FutureWarning: The default value o
f numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default
to False. Either specify numeric_only or select only columns which should be valid for the function.
  df_2.groupby('2021').aggregate('mean')
```

Out[86]:

2021	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	...	2011	2012	2013	2014	2015	2016	2017
3.6	4.10	4.40	3.80	3.40	2.50	2.60	2.60	2.60	2.70	2.40	...	2.40	2.70	2.50	2.50	2.40	2.60	2.50
3.9	4.90	5.30	5.90	4.20	3.40	3.30	2.90	2.90	3.00	2.80	...	3.10	3.50	2.70	3.30	2.90	3.30	3.40
4.4	3.40	3.10	3.20	2.90	2.80	2.80	2.70	2.70	2.50	2.60	...	2.80	2.50	2.60	2.80	2.90	3.00	3.20
4.5	5.20	5.00	5.40	4.60	4.00	4.10	3.90	4.10	3.90	3.80	...	4.30	4.10	4.00	3.90	4.20	4.00	3.90
4.8	4.30	5.10	5.00	4.40	4.10	3.80	3.50	3.90	4.40	3.80	...	4.30	4.10	4.00	3.90	2.40	4.50	3.80
4.9	4.35	4.75	4.75	4.55	3.45	3.35	3.25	3.30	3.55	3.70	...	3.50	3.65	3.65	3.25	2.60	3.50	3.35

In [87]:
```
#I wanted to see the mean of LOS for the first and last year of the dataset. We can see the
#mean for LOS is roughly 6 days in the year of 1999

df_2.groupby('Length of Stay In')['1999'].mean()
```

Out[87]:
```
Length of Stay In
days    6.127778
Name: 1999, dtype: float64
```

In [88]:
```
#the mean for the year of 2021 is rought 7.5 days
#the LOS could be on any island throughout the state during the year.

df_2.groupby('Length of Stay In')['2021'].mean()
```

Out[88]:
```
Length of Stay In
days    7.486111
Name: 2021, dtype: float64
```

In [90]:
```
df_2.shape
```

Out[90]:
```
(36, 26)
```

In [91]:
```
df_2.columns
```

Out[91]:
```
Index(['Visitor Type', 'Location Of Stay', 'Length of Stay In', '1999', '2000',
       '2001', '2002', '2003', '2004', '2005', '2006', '2007', '2008', '2009',
       '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018',
       '2019', '2020', '2021'],
      dtype='object')
```

In [92]:
```
df_2.describe()[['1999','2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007', '2008', '2009', '
```

Out[92]:

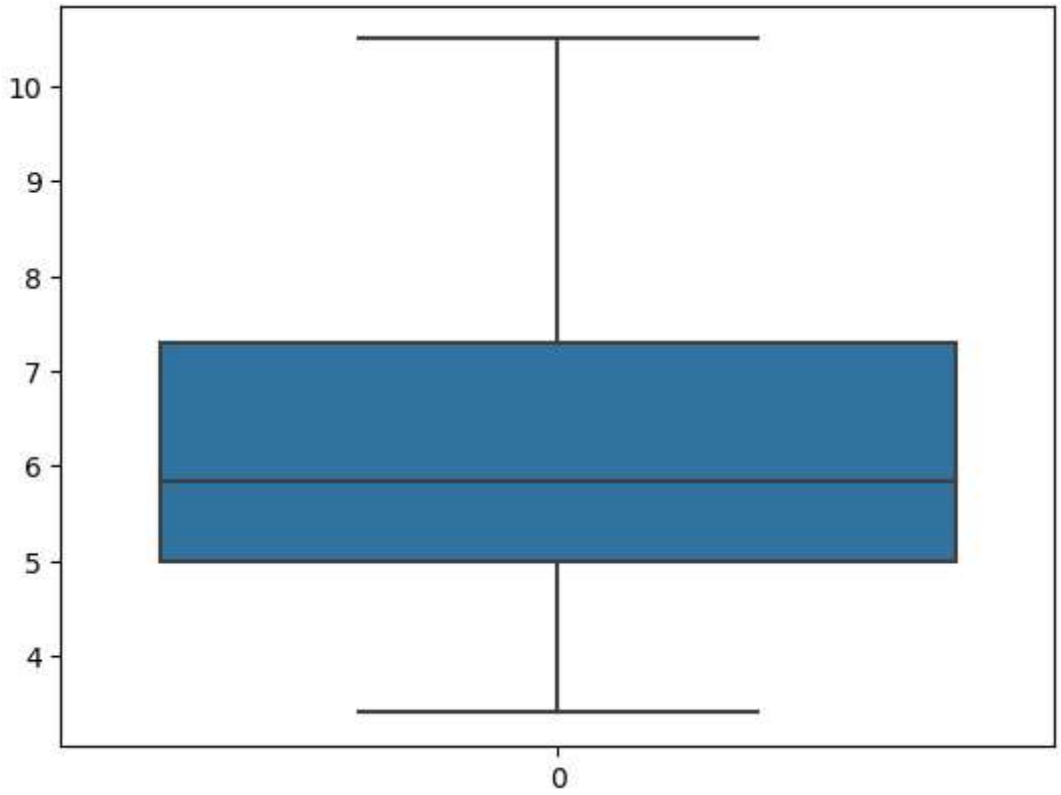
	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
count	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000	36.000000
mean	6.127778	6.263889	6.330556	6.330556	6.119444	6.188889	6.133333	6.091667	6.094444	6.263889	6.472222
std	1.694463	1.790635	1.915175	2.075272	2.355276	2.321672	2.445638	2.401830	2.360017	2.476536	2.441968
min	3.400000	3.100000	3.200000	2.900000	2.500000	2.600000	2.600000	2.300000	2.500000	2.400000	2.600000
25%	5.000000	5.000000	4.800000	4.675000	3.950000	4.025000	3.800000	3.975000	4.200000	3.875000	4.400000
50%	5.850000	6.000000	6.050000	6.200000	6.150000	6.250000	6.200000	6.100000	6.100000	6.250000	6.450000
75%	7.300000	7.325000	7.050000	7.325000	7.400000	7.350000	7.425000	7.525000	7.225000	7.500000	7.700000
max	10.500000	11.000000	11.400000	11.400000	11.400000	11.200000	11.300000	11.000000	11.100000	11.600000	11.600000

In [93]:
```
from scipy import stats
```

In [94]:
```
import seaborn as sns
```

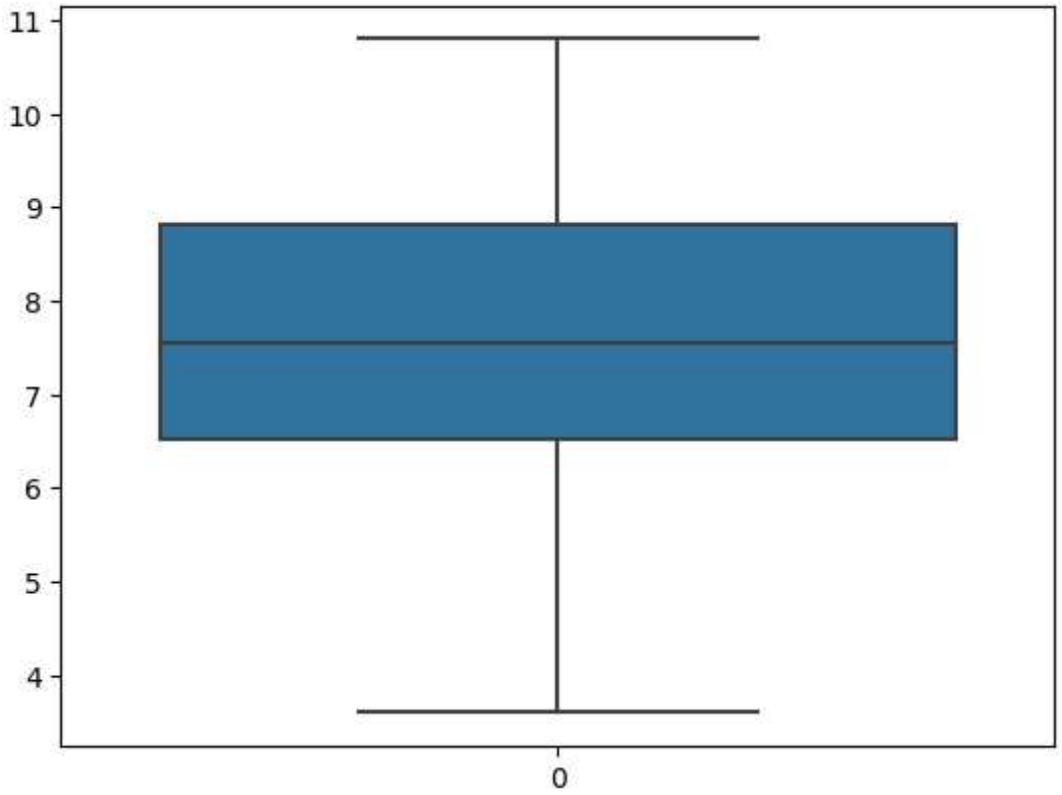
In [95]:
```
#for the foucus on the first and last years of the dataset, there are no outliers according to the boxplo
#I was trying to find outliers for 1999

sns.boxplot(df_2['1999'])
```

Out[95]: <Axes: >



In [96]:
```
#There are no outliers for the year of 2021

sns.boxplot(df_2['2021'])
```

Out[96]: <Axes: >



In [97]:
```
z_score_1999 = np.abs(stats.zscore(df_2['1999']))
print(z_score_1999)
```

```
0     1.659255
1     0.761462
4     0.162933
5     0.315890
8     0.342491
9     0.315890
12    0.675007
13    1.213683
16    1.093978
17    1.034125
20    0.016626
21    0.675007
24    0.103080
25    0.555302
28    1.153830
29    1.632653
32    0.136332
33    0.615155
36    2.616901
39    0.701609
42    1.838813
45    0.701609
48    0.734860
51    1.240285
54    1.599402
57    0.555302
60    1.479696
64    1.060726
66    0.196184
68    0.103080
70    0.615155
72    1.093978
74    0.256037
76    0.016626
78    1.333389
80    0.136332
Name: 1999, dtype: float64
```

In [98]:
```
z_score_2021 = np.abs(stats.zscore(df_2['2021']))
print(z_score_2021)
```

```
0     1.080527
1     0.211562
4     0.313793
5     0.299594
8     0.620486
9     0.044016
12    0.197363
13    1.986408
16    1.321906
17    1.321906
20    0.569371
21    0.299594
24    0.927180
25    0.197363
28    0.759634
29    1.577484
32    0.569371
33    0.350710
36    1.540567
```

In [99]:
```
df_2.dtypes
```

Out[99]:
```
Visitor Type          object
Location Of Stay      object
Length of Stay In     object
1999                 float64
2000                 float64
2001                 float64
2002                 float64
2003                 float64
2004                 float64
2005                 float64
2006                 float64
2007                 float64
2008                 float64
2009                 float64
2010                 float64
2011                 float64
2012                 float64
2013                 float64
2014                 float64
2015                 float64
2016                 float64
2017                 float64
2018                 float64
2019                 float64
2020                 float64
2021                 float64
dtype: object
```

In [100]:
```
from scipy.stats import zscore
```

In [101]:

```
#I applied the z score function to the dataset and there no outliers that jump out

numeric_cols = df_2.select_dtypes(include=[np.number]).columns
df_2[numeric_cols].apply(zscore)
```

Out[101]:

	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	...	2012	
0	1.659255	1.493049	1.519520	1.500037	1.326491	1.271672	1.230251	1.312510	1.334570	1.284293	...	1.160878	1.16
1	0.761462	0.530197	0.619282	0.669247	0.680589	0.485371	0.483807	0.510225	0.475097	0.424306	...	0.319436	0.32
4	0.162933	0.190368	0.248595	0.327157	0.336108	0.310638	0.317930	0.341323	0.303203	0.342402	...	0.319436	0.23
5	-0.315890	-0.319377	-0.228002	-0.063802	-0.008373	-0.126196	-0.055292	-0.038707	-0.083560	-0.067115	...	-0.121319	-0.16
8	0.342491	0.303644	0.301550	0.424897	0.508349	0.572738	0.566745	0.552451	0.518071	0.629065	...	0.639985	0.72
9	-0.315890	-0.489292	-0.333912	-0.210412	0.077747	0.004854	0.069115	0.087970	0.045361	0.014788	...	-0.081250	-0.04
12	-0.675007	-0.659207	-1.075285	-0.845722	-1.041816	-0.825131	-0.801737	-0.883217	-0.771138	-0.763295	...	-0.642211	-0.65
13	-1.213683	-1.055675	-1.340061	-1.432162	-1.558538	-1.567749	-1.465243	-1.474374	-1.458716	-1.582331	...	-1.523722	-1.58
16	-1.093978	-0.885760	-0.863464	-0.796852	-1.170997	-1.261965	-1.257898	-1.305472	-1.243848	-1.131861	...	-1.163104	-1.17
17	-1.034125	-0.829122	-0.810509	-0.943462	-1.127936	-1.218282	-1.133490	-1.052119	-0.943032	-0.968054	...	-1.123035	-1.05
20	-0.016626	-0.092824	-0.122091	-0.014932	0.249988	0.266954	0.193523	0.087970	0.045361	0.342402	...	0.399573	0.48
21	-0.675007	-0.885760	-0.810509	-0.650242	-0.352854	-0.388297	-0.304107	-0.249834	-0.212481	-0.230922	...	-0.241525	-0.20
24	0.103080	0.020453	0.089730	0.082807	0.206928	0.223271	0.193523	0.130195	0.088335	0.260499	...	0.319436	0.40
25	-0.555302	-0.715845	-0.651643	-0.601372	-0.352854	-0.475664	-0.387045	-0.376511	-0.470322	-0.476633	...	-0.562074	-0.52
28	-1.153830	-1.282228	-1.234151	-1.236682	-0.998756	-1.130915	-1.092021	-1.221021	-1.286821	-1.090909	...	-1.042898	-0.93
29	-1.632653	-1.791973	-1.657792	-1.676512	-1.429357	-1.480382	-1.423774	-1.432149	-1.544663	-1.500427	...	-1.603859	-1.54
32	-0.136332	-0.206100	-0.175046	-0.063802	0.120808	0.048537	0.027646	-0.038707	-0.083560	0.137643	...	0.239299	0.27
33	-0.615155	-0.829122	-0.810509	-0.650242	-0.352854	-0.475664	-0.345576	-0.334285	-0.341402	-0.353778	...	-0.441868	-0.40
36	2.616901	2.682453	2.684535	2.477437	2.273814	2.189024	2.142573	2.072569	2.151069	2.185232	...	2.042388	2.05
39	0.701609	1.266495	1.572475	1.304557	1.369552	1.490089	1.603474	1.396961	1.420517	1.570955	...	1.601633	1.33
42	1.838813	1.776240	1.784296	1.793257	1.757093	1.752190	1.686412	1.692540	1.721332	1.775714	...	1.721839	1.81
45	0.701609	0.813389	-0.333912	-0.063802	-0.395914	0.223271	0.525276	0.763578	0.475097	0.465258	...	0.519779	0.27
48	-0.734860	-0.545930	-0.228002	-1.041202	-1.170997	-1.261965	-1.340836	-1.347698	-1.329795	-1.418524	...	-1.203172	-1.50
51	1.240285	1.323134	1.095879	1.206817	1.240371	1.315356	1.313190	1.354735	1.334570	1.284293	...	1.321152	1.33
54	1.599402	1.549687	1.519520	1.548907	1.498732	1.533773	1.479066	1.481412	1.592412	1.407148	...	1.601633	1.61
57	-0.555302	-0.715845	-0.492778	-0.845722	-0.912636	-0.912498	-0.926144	-0.840992	-0.943032	-1.009006	...	-0.962761	-0.97

	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	...	2012	
60	1.479696	1.493049	1.466565	1.548907	1.541792	1.533773	1.562005	1.481412	1.635385	1.448100	...	1.641701	1.61
64	1.060726	0.926666	0.884058	0.962467	0.895890	0.834838	0.732622	0.763578	0.732939	0.670016	...	0.519779	0.52
66	-0.196184	-0.092824	-0.122091	0.033937	0.077747	0.048537	0.110584	0.045744	0.002387	-0.026164	...	-0.121319	-0.16
68	0.103080	0.020453	0.142685	0.229417	0.249988	0.179587	0.152054	0.299097	0.217256	0.219547	...	0.199230	0.27
70	-0.615155	-0.659207	-1.234151	-0.747982	-1.429357	-1.087231	-1.465243	-1.601051	-1.286821	-1.295668	...	-1.323378	-1.37
72	-1.093978	-0.659207	-0.704598	-0.943462	-0.869576	-1.043548	-1.092021	-0.925443	-0.728164	-1.009006	...	-0.962761	-0.97
74	-0.256037	-0.432654	-0.122091	-0.161542	-0.094493	-0.038830	-0.138231	-0.080932	-0.040586	-0.230922	...	-0.121319	0.03
76	-0.016626	0.020453	0.195640	0.131677	0.034687	0.135904	0.027646	0.045744	0.002387	0.014788	...	-0.001113	-0.00
78	-1.333389	-1.168952	-0.863464	-1.236682	-1.170997	-1.130915	-1.133490	-1.178796	-1.329795	-1.295668	...	-1.323378	-1.21
80	-0.136332	-0.092824	0.089730	-0.014932	-0.008373	0.004854	-0.055292	-0.038707	0.002387	-0.026164	...	-0.001113	-0.00

36 rows × 23 columns

In [102]:
```
#Overall the dataset was clean and didn't require a lot of cleaning and I did some removal of uneeded dat
#milestone one. I wanted to focus on getting the dataset organized in a manner were outliers were not an
#negatively impact future steps.
```

In [103]:
```
#Dropped the string columns so I can focus on the years as the main feature of my model. Given the inform
#previous milestones, I know the colimns below were consistent across the years. So the movement of tours
#dataset from 1999 to 2021 was more critical to model.

df_2 = df_2.drop(['Visitor Type', 'Location Of Stay', 'Length of Stay In'], axis=1)
```

In [104]:
```
#my target is 2018 from the dataset.

X = df_2.drop(['2018'], axis=1)
y = df_2['2018']
```

In [105]:
```
#Loading Linear Regression model library. I wanted to try to do one supervised and one unsupervised learn
#to start of with making predictions from my dataset that were more helpful using a lot of quantative inf

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
```

In [106]:
```
#setting the train test and split dataset

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

In [107]:
```
model_linear_regression = LinearRegression()
model_linear_regression.fit(X_train, y_train)
y_pred = model_linear_regression.predict(X_test)
```

In [108]:
```
#the scores from the model are below: we can see the RMSE value is .91 - close to 0 and the R2 score is r

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
print(f'The RMSE value is: {rmse}')
print(f'The R2 value is: {r2}')
```

```
The RMSE value is: 0.9129080835744351
The R2 value is: 0.7865516436886774
```

In [109]:
```
model_linear_regression.coef_
```

Out[109]:
```
array([ 0.37789506, -0.22579409,  0.21595734,  0.16699906, -0.57814808,
       -0.37917292,  1.84077336, -0.34368128, -0.65788377, -0.27971429,
       -0.9289576 ,  0.37895519, -0.20002191,  0.06075708,  1.60689441,
       -0.70616205, -0.3423159 , -0.08698725, -0.16164235,  1.46032167,
       -0.12167824,  0.04860311])
```

In [110]:
```
model_linear_regression.intercept_
```

Out[110]: -0.693595635653284

In [111]:
```
#using the model, we can see the number of visitors to the islands will be roughly the same or 79% likely
#with the total number of tourists of all types from all locations. If we know there is an almost 79% cha
#same number of toursists from 2018, I can apply that forecast to my business to better plan and allocate
#hotel would be able to plan for the travel seasons that see increased foot traffic such as winter and su
#a business leader, I would trust the model given the good accuracy R2 score of the model.
```

In [112]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import preprocessing
from sklearn import utils
```

In [113]:
```
lab = preprocessing.LabelEncoder()
y_transformed = lab.fit_transform(y)
```

In [114]:
```
print(y_transformed)
```

```
[26 21 19 13 24 17  8  0  2  4 23 14 23 11  6  1 19 10 31 25 30 23  3 27
 29  5 28 22 12 20  7  9 16 18  5 15]
```

In [115]:
```
#I also targeted 2004 and early year from the dataset to compare with 2018.

X = df_2.drop(['2004'], axis=1)
y = df_2['2004']
```

In [116]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

In [117]:
```
model_linear_regression = LinearRegression()
model_linear_regression.fit(X_train, y_train)
y_pred = model_linear_regression.predict(X_test)
```

In [118]:
```
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
print(f'The RMSE value is: {rmse}')
print(f'The R2 value is: {r2}')
```

```
The RMSE value is: 0.5458125128336484
The R2 value is: 0.9366555383115835
```

In [119]:
```
model_linear_regression.coef_
```

Out[119]:
```
array([ 0.16649004, -0.20932942,  0.33455827,  0.19284356, -0.23480493,
        1.05297728,  0.17560567, -0.60405141, -0.54572713,  0.06245127,
        0.23162298, -0.2077242 ,  0.15508769,  0.77618148, -0.42833826,
        0.02525683,  0.02723449,  0.04986419, -0.42133678,  0.53341956,
       -0.12458163,  0.10023698])
```

In [120]:
```
model_linear_regression.intercept_
```

Out[120]:
```
-0.7723555748926438
```