In [80]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [81]:

```
lobsterland = pd.read_csv("lobsterland_2021.csv")
```

In [82]:

```
lobsterland.head()
```

Out[82]:

	Date	Day.of.Week	Max	Average	Min	Precip	DayPass	UniqueVisitor	AvgDuration	ParkingRev
0	2021-05-31	Monday	59	53.5	47	0.90	3261	4024	328	15212.86
1	2021-06-01	Tuesday	77	63.2	47	0.02	2263	2646	376	9323.06
2	2021-06-02	Wednesday	77	64.7	49	0.00	2731	2936	318	11252.19
3	2021-06-03	Thursday	68	62.7	58	NaN	2879	3196	485	11861.30
4	2021-06-04	Monday	93	80.5	64	0.00	2118	2334	240	8727.01

Question#2 A.Called head() function B.5 rows are visible

In [83]:

```
lobsterland.shape
```

Out[83]:

```
(99, 20)
```

C.There are 99 rows and 20 column

In [84]:

```
SpecialE = lobsterland[['Spec_Event']]
SpecialE.describe()
```

Out[84]:

	Spec_Event
count	99.000000
mean	3.575758
std	1.761788
min	1.000000
25%	2.000000
50%	4.000000
75%	5.000000
max	6.000000

D. a) categorical b) Described data are listed above

```
In [85]: lobsterland[['Spec_Event']]=pd.Categorical(lobsterland.Spec_Event)
         lobsterland[['Spec_Event']].dtypes
```

```
Out[85]: Spec_Event    category
         dtype: object
```

```
In [ ]:
```

```
In [ ]:
```

C)Covert Special events variable into categorical variable

```
In [86]: SpecialE.value_counts()
```

```
Out[86]: Spec_Event
         5             29
         1             18
         2             15
         3             15
         6             14
         4              8
         dtype: int64
```

d)showed above e)step b showed std,mean,min,max,25%,50%,75%, these are the information that Python calculated. Step d counts all the days that have 5(1,2,3,6,4) events.In another word, it shows us that 29 days have 5 events, 18 days have 1 event and so on.

```
In [87]: lobsterland.isnull().sum()
```

```
Out[87]: Date              0
         Day.of.Week       0
         Max               0
         Average           0
         Min               0
         Precip            6
         DayPass           0
         UniqueVisitor     0
         AvgDuration       0
         ParkingRev        0
         SnackShackRev     0
         LobsteramaRev     0
         GoldZoneRev       0
         MerchRev          0
         StaffHours        0
         Sign_Ups2022      0
         Fireworks         0
         Spec_Event        0
         DailyGrossRev     0
         day_type          0
         dtype: int64
```

```
In [88]: Precip=lobsterland[['Precip']]
         lobsterland['Precip']=lobsterland['Precip'].fillna(0)
```

```
lobsterland.isnull().sum()
```

Out[88]:

```
Date              0
Day.of.Week       0
Max               0
Average           0
Min               0
Precip            0
DayPass           0
UniqueVisitor     0
AvgDuration       0
ParkingRev        0
SnackShackRev     0
LobsteramaRev     0
GoldZoneRev       0
MerchRev          0
StaffHours        0
Sign_Ups2022      0
Fireworks         0
Spec_Event        0
DailyGrossRev     0
day_type          0
dtype: int64
```

E a)As shown above, using isnull().sum() b)It assesses the amount of rainfall in that day, the average of all the data under Precip has no meaning, replacing null with 0 won't cause misunderstanding on the data. Moreover, Weather is unpredictable by just using described data such as mean, sd, min, and max, when we have null under Precip, we can assume the day without rainfall, and not affecting overall data analysis.

In [89]:

```
lobsterland[['Min']] = lobsterland[['Min']].apply(lambda x: [Min if Min >= 51 else 51 f
lobsterland[['Min']]
```

Out[89]:

	Min
0	51
1	51
2	51
3	58
4	64
...	...
94	61
95	54
96	55
97	51
98	60

99 rows × 1 columns

G.numbers are less than 51 are converted to 51

In [90]:

```
LDE=lobsterland.iloc[-4:]
LDE
```

Out[90]:

	Date	Day.of.Week	Max	Average	Min	Precip	DayPass	UniqueVisitor	AvgDuration	ParkingRev
95	2021-09-03	Friday	72	63.40	54	0.00	4494	5108	289	18514.43
96	2021-09-04	Saturday	75	64.50	55	0.00	4200	5066	375	17304.08
97	2021-09-05	Sunday	68	60.70	51	0.00	4424	5482	412	18226.30
98	2021-09-06	Monday	76	66.58	60	0.19	5112	5570	471	18407.56

In [91]:

```
lobsterland.mean()
```

Out[91]:

```
Max                  76.797980
Average              68.191717
Min                  60.515152
Precip                0.156465
DayPass            3241.111111
UniqueVisitor      3757.696970
AvgDuration         337.141414
ParkingRev        13344.433939
SnackShackRev     16233.011010
LobsteramaRev     28292.904646
GoldZoneRev       26749.907273
MerchRev          32051.397475
StaffHours          808.894626
Sign_Ups2022         44.636364
Fireworks             0.252525
DailyGrossRev    120135.184444
dtype: float64
```

H. a)Comparing to the overall average of the dataset, the Labor Day Effect brought Daypass up in a significant number. Average Dayapss is about3241, but over the last four days of labor-day, day pass increased to above 4000, and on the last day of the holiday, its day pass number even increased to 5112. Unique visitors are also increased dramatically. Because of the increasing of visitors, all the revenues shown above such as parking revenue, snack shack rev, and so on are increased dramatically, staff hours also increased because more visitors needed to be served,and more people in the park also drives up sighn_ups 2022.

b)2022 sign-ups seem to stand out the most. First of all, when more people are in the park, more people would know about sign-ups in 2022 and purchase them. Second of all, the Herd effect also affects the sign-ups 2022, people are more willing to sign up when they see many others are signing up. Third of all, During the holiday, people are more willing to spend money.

In [92]:

```
Daytype=lobsterland[['day_type']]
Daytype.value_counts()
```

Out[92]:

```
day_type
Overcast         17
```

```
Cloudy           16
Partly Sunny     15
Partly Cloudy    13
Rainy            11
Very Sunny       10
Sunny             9
Very Rainy        8
dtype: int64
```

In [93]:

```
lobsterland[['day_type']] = lobsterland[['day_type']].replace(['Overcast','Partly Cloud
lobsterland[['day_type']] = lobsterland[['day_type']].replace(['Partly Sunny','Very Sun
lobsterland[['day_type']] = lobsterland[['day_type']].replace(['Rainy','Very Rainy'],'R
Daytype.value_counts()
```

Out[93]:

```
day_type
Overcast         17
Cloudy           16
Partly Sunny     15
Partly Cloudy    13
Rainy            11
Very Sunny       10
Sunny             9
Very Rainy        8
dtype: int64
```

I. a)are shown above b) Firstly, it is better for user to extract the information they need. Some people only care if it is raining, so they can bring umbrellas, some care about sunshine,they can put on sunscream before leaving home, and etc. Secondly, it is better for visulazition, less level of factor variable can be read easier and more clearly.

In [94]:

```
del lobsterland['Max']
```

In [95]:

```
lobsterland
```

Out[95]:

	Date	Day.of.Week	Average	Min	Precip	DayPass	UniqueVisitor	AvgDuration	ParkingRev	Snac
0	2021-05-31	Monday	53.50	51	0.90	3261	4024	328	15212.86	
1	2021-06-01	Tuesday	63.20	51	0.02	2263	2646	376	9323.06	
2	2021-06-02	Wednesday	64.70	51	0.00	2731	2936	318	11252.19	
3	2021-06-03	Thursday	62.70	58	0.00	2879	3196	485	11861.30	
4	2021-06-04	Monday	80.50	64	0.00	2118	2334	240	8727.01	
...	...	...	...	...	...	...	...	...	...	
94	2021-09-02	Sunday	64.50	61	0.00	4653	5425	347	19170.07	
95	2021-09-03	Friday	63.40	54	0.00	4494	5108	289	18514.43	

	Date	Day.of.Week	Average	Min	Precip	DayPass	UniqueVisitor	AvgDuration	ParkingRev	Snac
96	2021-09-04	Saturday	64.50	55	0.00	4200	5066	375	17304.08	
97	2021-09-05	Sunday	60.70	51	0.00	4424	5482	412	18226.30	
98	2021-09-06	Monday	66.58	60	0.19	5112	5570	471	18407.56	

99 rows × 19 columns

J.Max has been removed

In [96]:
```
del lobsterland['Min']
```

In [97]:
```
lobsterland
```

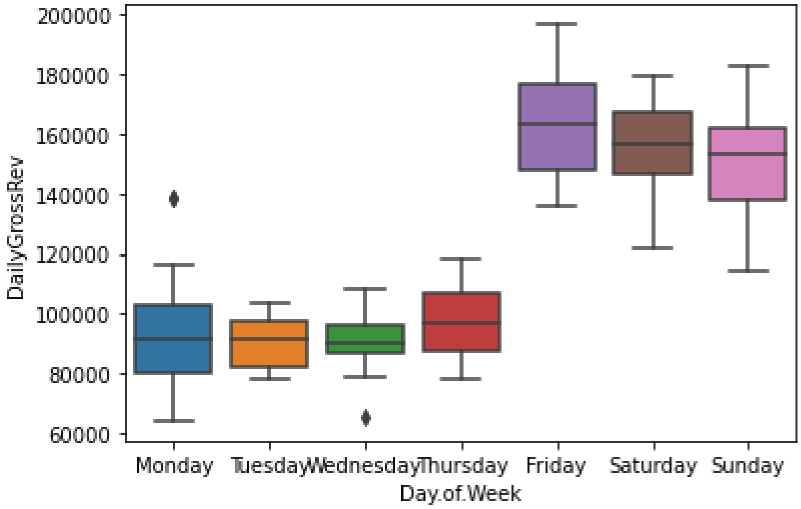
Out[97]:

	Date	Day.of.Week	Average	Precip	DayPass	UniqueVisitor	AvgDuration	ParkingRev	SnackShac
0	2021-05-31	Monday	53.50	0.90	3261	4024	328	15212.86	185
1	2021-06-01	Tuesday	63.20	0.02	2263	2646	376	9323.06	113
2	2021-06-02	Wednesday	64.70	0.00	2731	2936	318	11252.19	137
3	2021-06-03	Thursday	62.70	0.00	2879	3196	485	11861.30	144
4	2021-06-04	Monday	80.50	0.00	2118	2334	240	8727.01	106
...	...	...	...	...	...	...	...	...	
94	2021-09-02	Sunday	64.50	0.00	4653	5425	347	19170.07	233
95	2021-09-03	Friday	63.40	0.00	4494	5108	289	18514.43	225
96	2021-09-04	Saturday	64.50	0.00	4200	5066	375	17304.08	210
97	2021-09-05	Sunday	60.70	0.00	4424	5482	412	18226.30	222
98	2021-09-06	Monday	66.58	0.19	5112	5570	471	18407.56	198

99 rows × 18 columns

K.Min has been removed

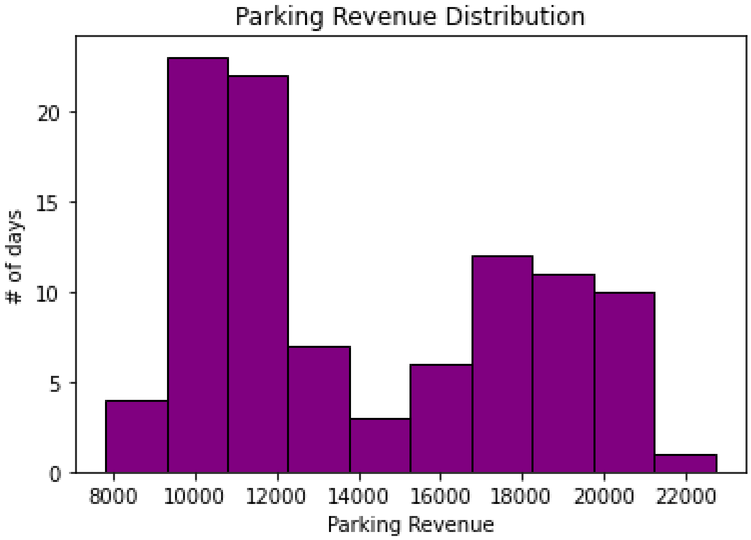
In [98]:
```
sns.boxplot( y=lobsterland["DailyGrossRev"], x=lobsterland["Day.of.Week"] );
plt.show()
```



L.a)On Friday, lobster land has the highest average revenue, and during the weekend the average revenues are higher than the weekday. On Friday, after they have been through a tough week, people tend to have fun, therefore, the average revenue on Friday is the highest. On Sunday, some people need to prepare for the work/school day on Monday, therefore, Sunday has the lowest total revenue among these three days.

In [99]:
```
plt.hist(lobsterland['ParkingRev'], align='right', color='purple', edgecolor='black')
plt.xlabel('Parking Revenue')
plt.ylabel('# of days')
plt.title('Parking Revenue Distribution')
```

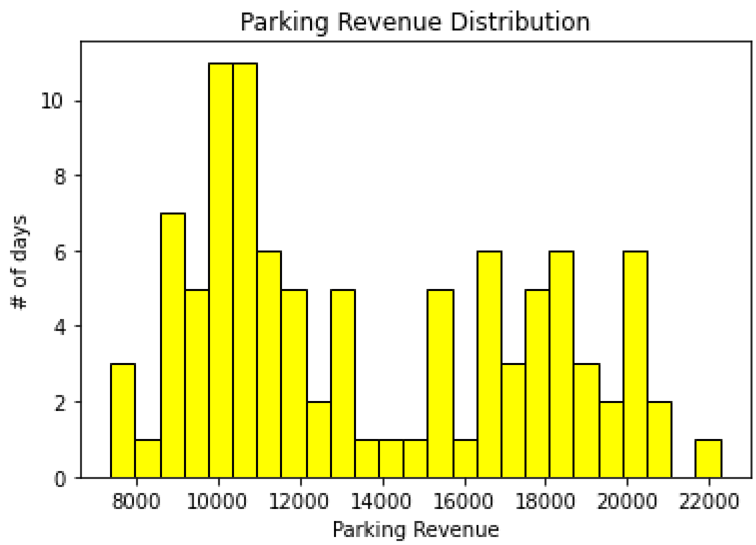
Out[99]:
```
Text(0.5, 1.0, 'Parking Revenue Distribution')
```



M.Distribution chart is shown above

In [100…
```
plt.hist(lobsterland['ParkingRev'],bins=25, align='right', color='Yellow', edgecolor='b
plt.xlabel('Parking Revenue')
plt.ylabel('# of days')
plt.title('Parking Revenue Distribution')
```

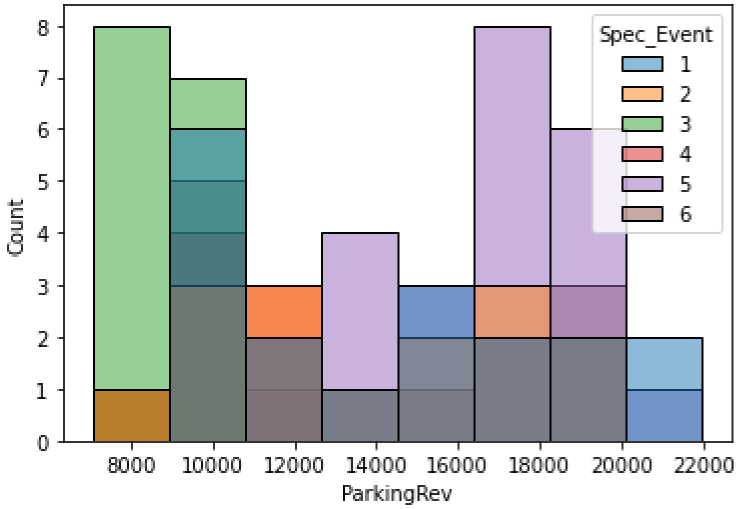
```
Out[100…  Text(0.5, 1.0, 'Parking Revenue Distribution')
```



N.a)The second diagram shows a shorter rectangle, Some ranges have no rectangle. The graph with more bins can show greater detail. Moreover, but it can be difficult to discern the signal from the noise, it can prevent users from discovering useful patterns.

```
In [101…  sns.histplot(data=lobsterland, x='ParkingRev', hue="Spec_Event")
```

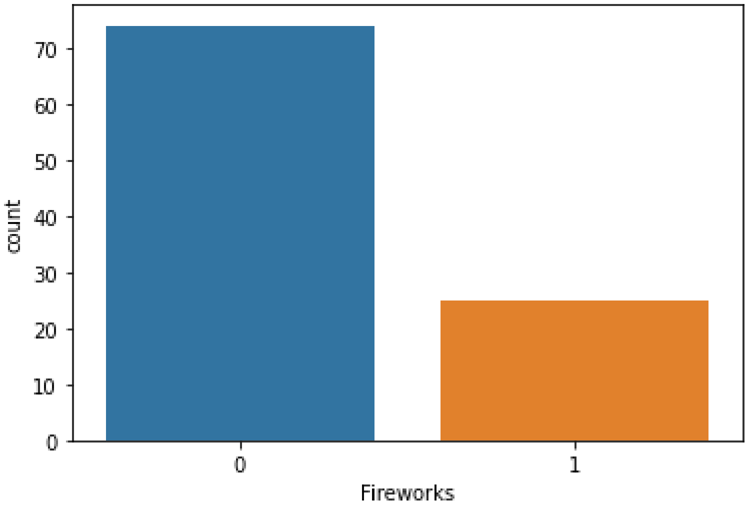
```
Out[101…  <AxesSubplot:xlabel='ParkingRev', ylabel='Count'>
```



N.b)When Comedy show events are held in the park, less parking revenue are made at the most time. When there are no events held in the park, parking revenue is really high in a range between 20000 and 22000.

```
In [105…  sns.countplot(x ='Fireworks', data = lobsterland)
```

```
Out[105…  <AxesSubplot:xlabel='Fireworks', ylabel='count'>
```



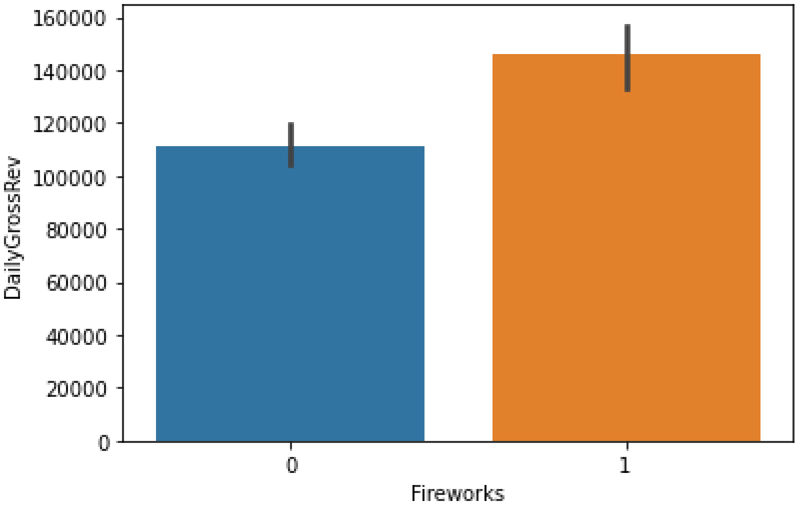
O.It shows the days that park has fireworks and the days that don't have fireworks. During the summer 2020, more than 70 days the park doesn't have fireworks, and more than 20 days and less than 30 days, the park has fireworks.

In [106…

```
sns.barplot(x="Fireworks", y="DailyGrossRev", data=lobsterland)
```

Out[106…

```
<AxesSubplot:xlabel='Fireworks', ylabel='DailyGrossRev'>
```



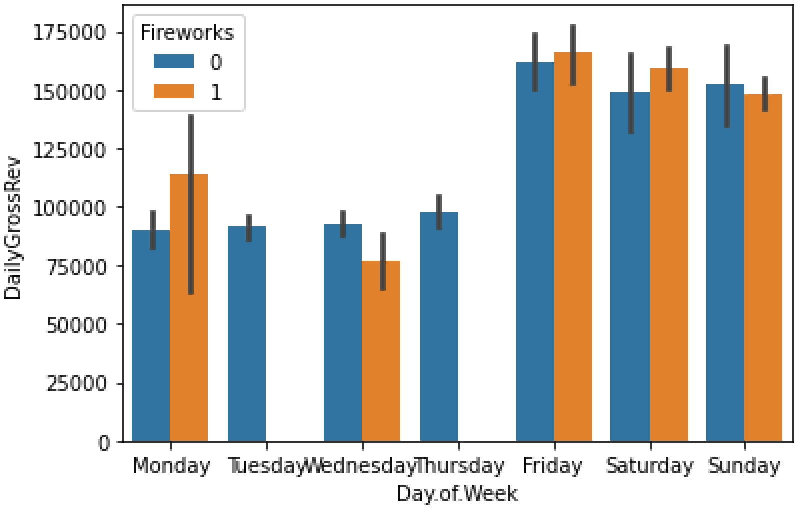
P.It compares the profitability between the park with fireworks and the park without fireworks.Apparently, Fireworks can bring the park more revenue.

In [107…

```
sns.barplot(x="Day.of.Week", y="DailyGrossRev",hue='Fireworks', data=lobsterland)
```

Out[107…

```
<AxesSubplot:xlabel='Day.of.Week', ylabel='DailyGrossRev'>
```



Q.It shows the daily gross revenue of each day of the week with fireworks and the daily gross revenue of each day of the week without fireworks. On Friday, Saturday, Sunday, and Monday, the park make more revenue with firework shows. On Wednesday, The parks make less revenue with fireworks. On Tuesday and Thursday, the park has no fireworks. This graph can help the decision-maker to decide which day they should have fireworks to increase revenue, and which day they should cancel the fireworks. The previous diagram can not reveal this insight.

In [ ]: