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


lobster = pd.read_csv("lobsterland_2021.csv")
lobster.head()
```

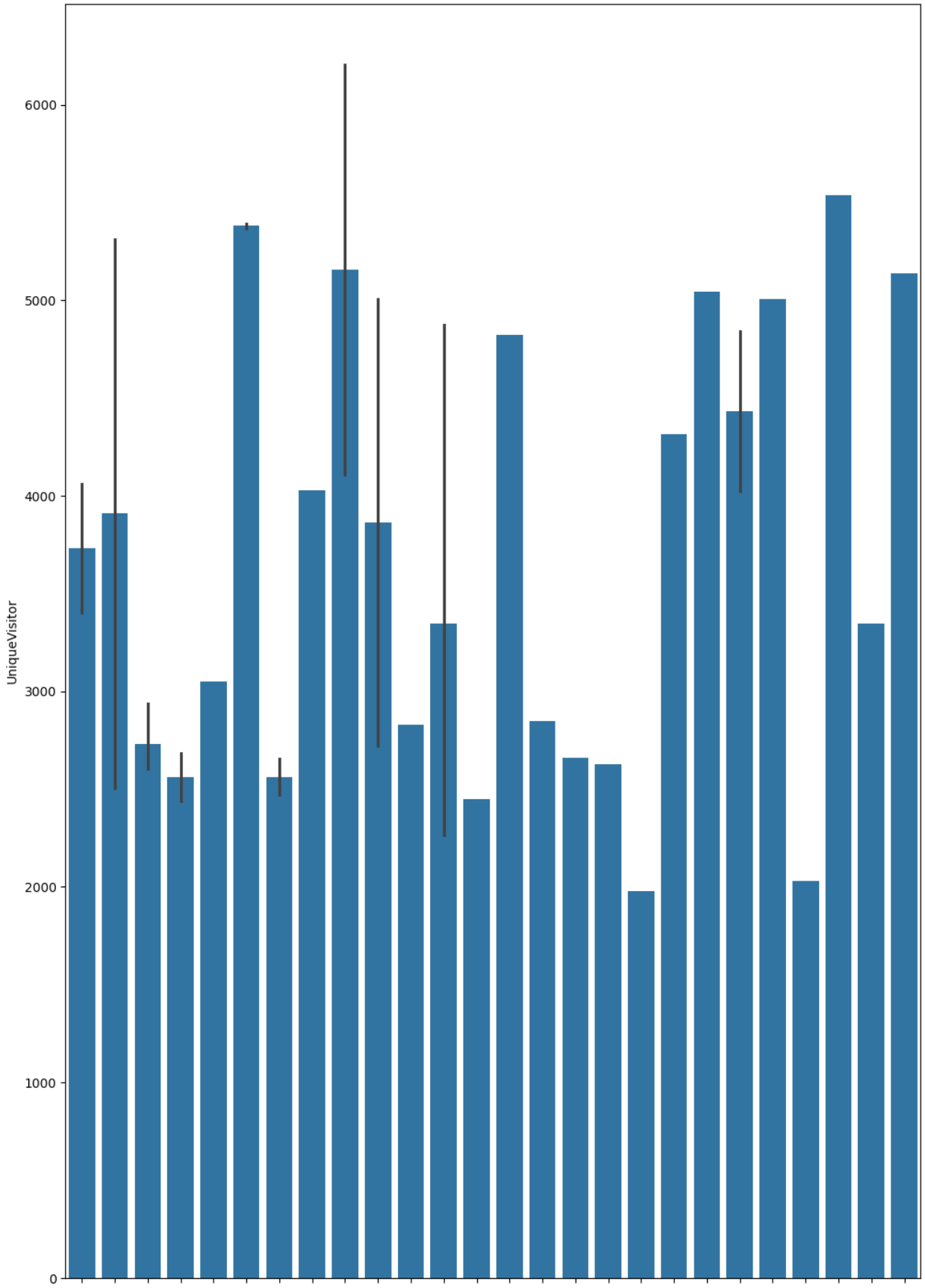
```
{"repr_error":"'str' object has no attribute
'empty'","type":"dataframe","variable_name":"lobster"}
```

```
lobster.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           99 non-null     object
 1   Day.of.Week    99 non-null     object
 2   Max            99 non-null     int64
 3   Average        99 non-null     float64
 4   Min            99 non-null     int64
 5   Precip         93 non-null     float64
 6   DayPass        99 non-null     int64
 7   UniqueVisitor  99 non-null     int64
 8   AvgDuration    99 non-null     int64
 9   ParkingRev     99 non-null     float64
 10  SnackShackRev  99 non-null     float64
 11  LobsteramaRev  99 non-null     float64
 12  GoldZoneRev    99 non-null     float64
 13  MerchRev       99 non-null     float64
 14  StaffHours     99 non-null     float64
 15  Sign_Ups2022   99 non-null     int64
 16  Fireworks      99 non-null     int64
 17  Spec_Event     99 non-null     int64
 18  DailyGrossRev  99 non-null     float64
 19  day_type       99 non-null     object
dtypes: float64(9), int64(8), object(3)
memory usage: 15.6+ KB
```

```
lobster.shape
(99,20)
plt.figure(figsize=(12,18))
sns. barplot (data= lobster, x='Precip',y='UniqueVisitor')
```

```
<Axes: xlabel='Precip', ylabel='UniqueVisitor'>
```


6000
5000
4000
3000
2000
1000
0
UniqueVisitor

```
lobster.columns
```

```
Index(['Date', 'Day.of.Week', 'Max', 'Average', 'Min', 'Precip',
'DayPass',
       'UniqueVisitor', 'AvgDuration', 'ParkingRev', 'SnackShackRev',
       'LobsteramaRev', 'GoldZoneRev', 'MerchRev', 'StaffHours',
       'Sign_Ups2022', 'Fireworks', 'Spec_Event', 'DailyGrossRev',
'day_type'],
      dtype='object')
```

```
print(lobster['Spec_Event'].head())
```

```
0    1
1    1
2    4
3    4
4    3
Name: Spec_Event, dtype: int64
```

```
print(lobster['Spec_Event'].describe())
```

```
count    99.000000
mean      3.575758
std       1.761788
min       1.000000
25%       2.000000
50%       4.000000
75%       5.000000
max       6.000000
Name: Spec_Event, dtype: float64
```

```
lobster['Spec_Event'] = lobster['Spec_Event'].astype('category')
```

```
print(lobster['Spec_Event'].value_counts())
```

```
5    29
1    18
2    15
3    15
6    14
4     8
Name: Spec_Event, dtype: int64
```

In Step 2, we used the describe() function to get an overview of our data, which includes things like how many values are there, how many unique values, what's the most common value, and how often it appears.

In Step 3, we used the value_counts() function to specifically look at one categorical variable called "SpecEvent". This helped us understand better how often each different event happened. It's like counting how many times each type of event occurred, which gives us a clearer picture of what's happening with that variable.

```
m_values = lobster.isnull().sum()
print("Missing values in the dataset:")
print(m_values)
lobster['Precip'].fillna(0, inplace=True)
```

```
Missing values in the dataset:
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
```

I used a method called isnull().sum() to find out how many missing values there are in each column of the DataFrame. If the sum for a column is greater than zero, it means there are missing values in that column. Specifically, I looked for missing values in the 'Precip' column by checking if there were any NaN values present.

```
lobster['Min'] = lobster['Min'].apply(lambda x: max(x, 51))
subset = lobster.iloc[-4:]
print(subset)
```

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

    AvgDuration  ParkingRev  SnackShackRev  LobsteramaRev  GoldZoneRev
```

```
\
95          289    18514.43       22563.10       40984.81     40982.92

96          375    17304.08       21087.59       38308.28     38304.65

97          412    18226.30       22211.07       40346.49     40348.37

98          471    18407.56       19829.31       55447.87     40739.78


    MerchRev   StaffHours  Sign_Ups2022  Fireworks Spec_Event
DailyGrossRev  \
95  51518.10  1123.555994            93          0          1
165643.57
96  48504.75  1050.101320            77          1          2
154805.40
97  52718.03  1106.061363            83          0          5
163065.68
98  49912.88  1033.587580           129          1          1
138250.61

        day_type
95      Overcast
96  Partly Sunny
97        Cloudy
98    Very Sunny

lobster_stats = lobster.describe()
subset_stats = subset.describe()
print(lobster_stats)
print(subset_stats)

             Max    Average        Min     Precip      DayPass
UniqueVisitor  \
count  99.000000  99.000000  99.000000  99.000000    99.000000
99.000000
mean   76.797980  68.191717  60.515152   0.156465  3241.111111
3757.696970
std     8.162857   6.246385   5.812309   0.387470   993.167484
1261.573462
min    59.000000  53.500000  51.000000   0.000000  1713.000000
1976.000000
25%    70.500000  63.650000  56.000000   0.000000  2374.500000
2649.500000
50%    77.000000  67.100000  61.000000   0.000000  2879.000000
3145.000000
75%    82.000000  72.350000  64.000000   0.100000  4159.500000
5055.500000
max    97.000000  85.600000  76.000000   2.240000  5333.000000
6206.000000
```

```
       AvgDuration    ParkingRev  SnackShackRev  LobsteramaRev
GoldZoneRev  \
count    99.000000     99.000000      99.000000      99.000000
99.000000
mean    337.141414  13344.433939   16233.011010   28292.904646
26749.907273
std     100.879804   4053.768314    4913.066475   10324.998317
11131.686649
min     131.000000   7055.620000    8599.750000   13903.530000
12197.150000
25%     259.500000   9782.520000   11918.040000   19281.950000
16906.915000
50%     333.000000  11861.300000   14452.550000   23377.000000
20499.360000
75%     402.000000  17136.625000   20647.465000   37935.720000
37934.240000
max     612.000000  21971.140000   26771.860000   55447.870000
48637.010000

           MerchRev   StaffHours  Sign_Ups2022  Fireworks
DailyGrossRev
count     99.000000    99.000000     99.000000  99.000000
99.000000
mean   32051.397475   808.894626     44.636364   0.252525
120135.184444
std    11056.811566   245.058530     27.776974   0.436672
35276.835836
min    16137.340000   428.275710     11.000000   0.000000
64159.660000
25%    22367.805000   593.567811     21.000000   0.000000
90302.660000
50%    27120.900000   719.770323     28.000000   0.000000
107836.250000
75%    42093.305000  1032.079076     69.500000   0.500000
151602.205000
max    53970.000000  1333.313863    129.000000   1.000000
196572.640000
             Max    Average        Min  Precip      DayPass
UniqueVisitor  \
count   4.000000   4.000000   4.000000  4.0000     4.000000
4.000000
mean   72.750000  63.795000  55.000000  0.0475  4557.500000
5306.500000
std     3.593976   2.448694   3.741657  0.0950   390.354967
256.563832
min    68.000000  60.700000  51.000000  0.0000  4200.000000
5066.000000
25%    71.000000  62.725000  53.250000  0.0000  4368.000000
```

```
5097.500000
50%    73.500000  63.950000  54.500000  0.0000  4459.000000
5295.000000
75%    75.250000  65.020000  56.250000  0.0475  4648.500000
5504.000000
max    76.000000  66.580000  60.000000  0.1900  5112.000000
5570.000000

       AvgDuration    ParkingRev  SnackShackRev  LobsteramaRev
GoldZoneRev  \
count     4.000000      4.000000        4.00000       4.000000
4.000000
mean    386.750000  18113.092500    21422.76750   43771.862500
40093.930000
std      76.220623    552.298247     1234.66997    7867.246941
1221.159022
min     289.000000  17304.080000    19829.31000   38308.280000
38304.650000
25%     353.500000  17995.745000    20773.02000   39836.937500
39837.440000
50%     393.500000  18316.930000    21649.33000   40665.650000
40544.075000
75%     426.750000  18434.277500    22299.07750   44600.575000
40800.565000
max     471.000000  18514.430000    22563.10000   55447.870000
40982.920000

           MerchRev   StaffHours  Sign_Ups2022  Fireworks
DailyGrossRev
count      4.000000     4.000000      4.000000    4.00000
4.000000
mean   50663.440000  1078.326564     95.500000    0.50000
155441.315000
std     1841.653841    43.255680     23.288051    0.57735
12357.746221
min    48504.750000  1033.587580     77.000000    0.00000
138250.610000
25%    49560.847500  1045.972885     81.500000    0.00000
150666.702500
50%    50715.490000  1078.081342     88.000000    0.50000
158935.540000
75%    51818.082500  1110.435021    102.000000    1.00000
163710.152500
max    52718.030000  1123.555994    129.000000    1.00000
165643.570000

diff = subset_stats.loc["mean"] - lobster_stats.loc["mean"]
print(diff)
```

```
Max                 -4.047980
Average             -4.396717
Min                 -5.515152
Precip              -0.108965
DayPass           1316.388889
UniqueVisitor     1548.803030
AvgDuration         49.608586
ParkingRev        4768.658561
SnackShackRev     5189.756490
LobsteramaRev    15478.957854
GoldZoneRev      13344.022727
MerchRev         18612.042525
StaffHours         269.431939
Sign_Ups2022        50.863636
Fireworks            0.247475
DailyGrossRev    35306.130556
Name: mean, dtype: float64
```

```
pct_diff = diff / lobster_stats.loc["mean"] * 100
print(pct_diff)
```

```
Max                -5.270946
Average            -6.447582
Min                -9.113671
Precip            -69.641704
DayPass            40.615358
UniqueVisitor      41.216816
AvgDuration        14.714474
ParkingRev         35.735188
SnackShackRev      31.970387
LobsteramaRev      54.709681
GoldZoneRev        49.884370
MerchRev           58.069364
StaffHours         33.308657
Sign_Ups2022      113.951120
Fireworks          98.000000
DailyGrossRev      29.388668
Name: mean, dtype: float64
```

Based on what we found, two things really stand out: the "Precip" and "DayPass" variables. During the Labor Day long weekend, there was no rain at all, which is a big difference compared to the average rainfall of 0.11 inches. This suggests the weather was perfect for going to Lobster Land, so more people probably visited the park.

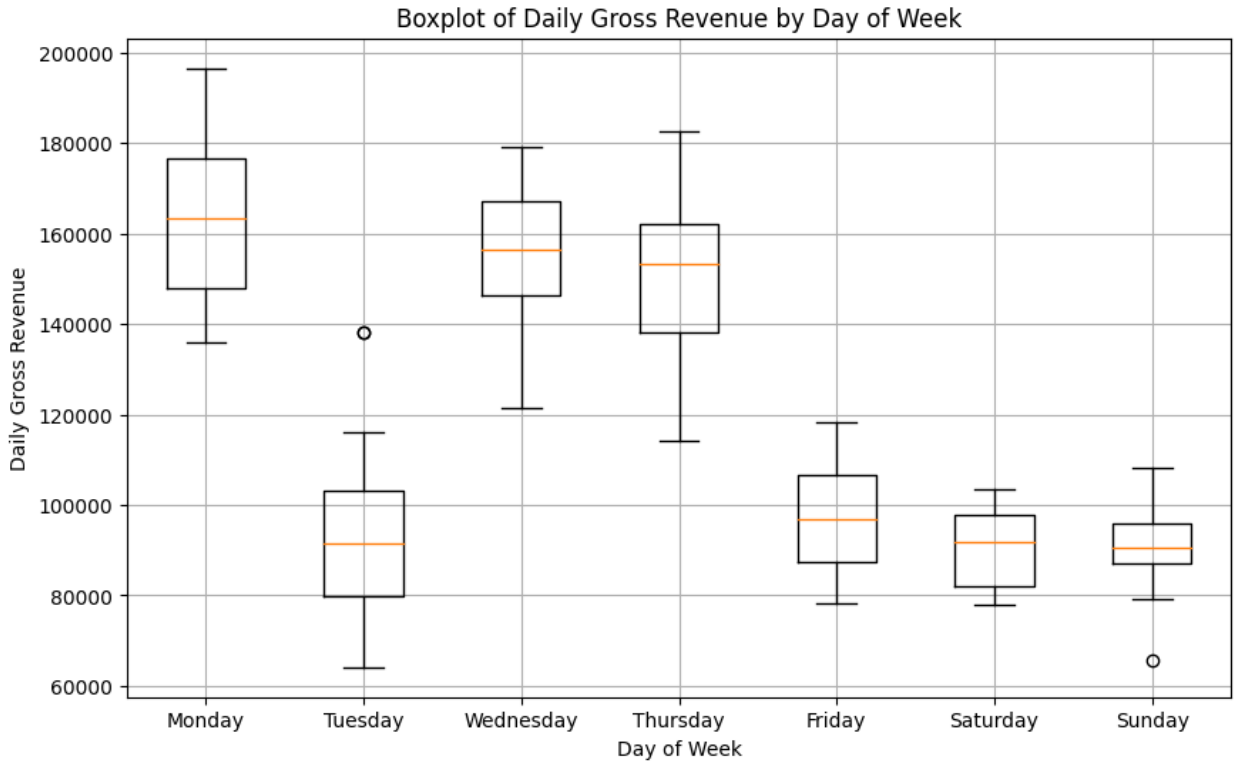
Also, we noticed that there were a lot more day passes sold during the Labor Day weekend compared to the usual. This could be because the weather was great and people wanted to enjoy the end of summer at the park. It could also be because Lobster Land offered special deals on day passes for the holiday weekend.

So, in short, the awesome weather and maybe some special offers likely brought more people to Lobster Land over Labor Day weekend, leading to higher sales of day passes.

```
mapping = {
    'Weekday': 'Weekday',
    'Saturday': 'Weekend',
    'Sunday': 'Weekend',
    'Memorial Day': 'Holiday',
    'Independence Day': 'Holiday',
    'Labor Day': 'Holiday',
    'Other': 'Other',
    'Summer': 'Other'
}
lobster['day_type'] = lobster['day_type'].map(mapping)
```

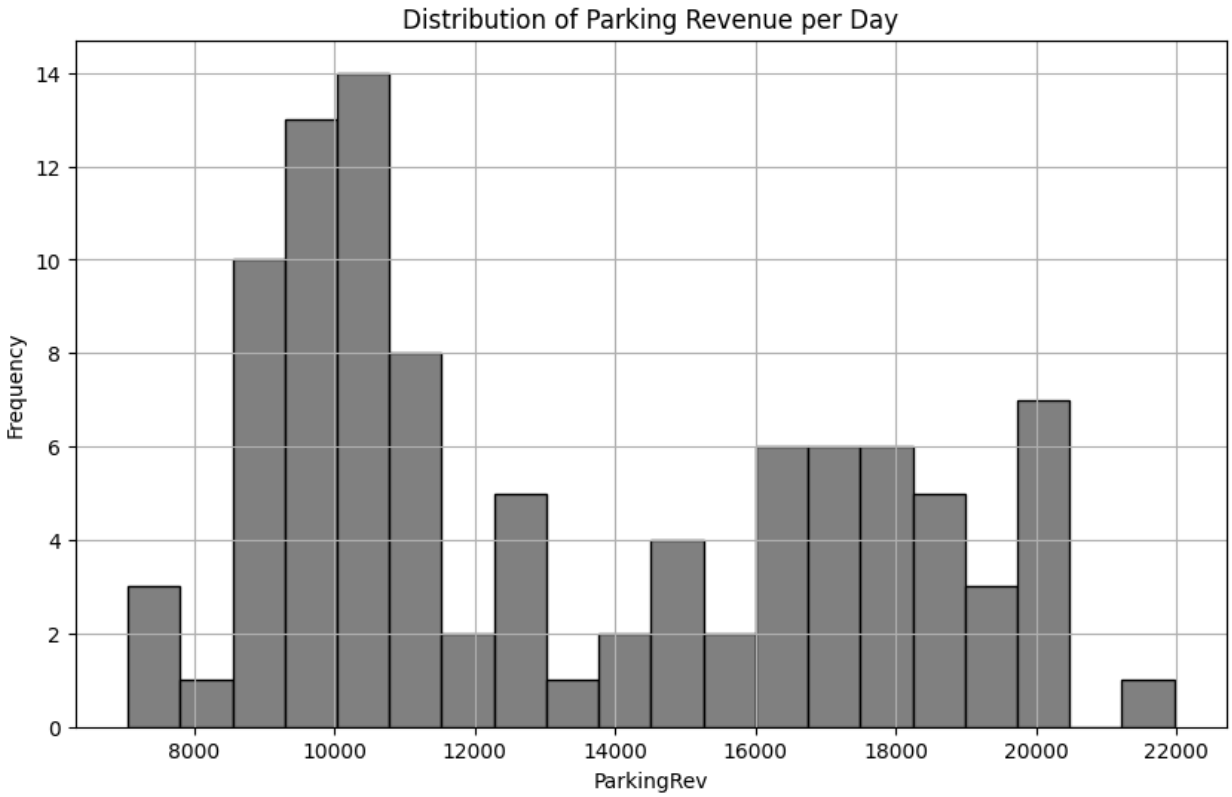
Making analysis and understanding data can be easier if we group things together. For example, instead of looking at each holiday separately like Memorial Day, Independence Day, and Labor Day, we can just call them all "Holiday". This helps us see patterns and behaviors related to holidays more easily. Also, when we simplify how we look at data like this, it can help computer programs work better when they're trying to predict things based on the data. So, grouping similar things together can make things simpler and help us make better predictions.

```
lobster.rename(columns={'Average': 'Avg'}, inplace=True)

days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday', 'Sunday']

# Boxplot
plt.figure(figsize=(10, 6))
plt.boxplot(grouped_data.values)
plt.xlabel('Day of Week')
plt.ylabel('Daily Gross Revenue')
plt.title('Boxplot of Daily Gross Revenue by Day of Week')
plt.xticks(ticks=np.arange(1, 8), labels=days_of_week)
plt.grid(True)
plt.show()
```

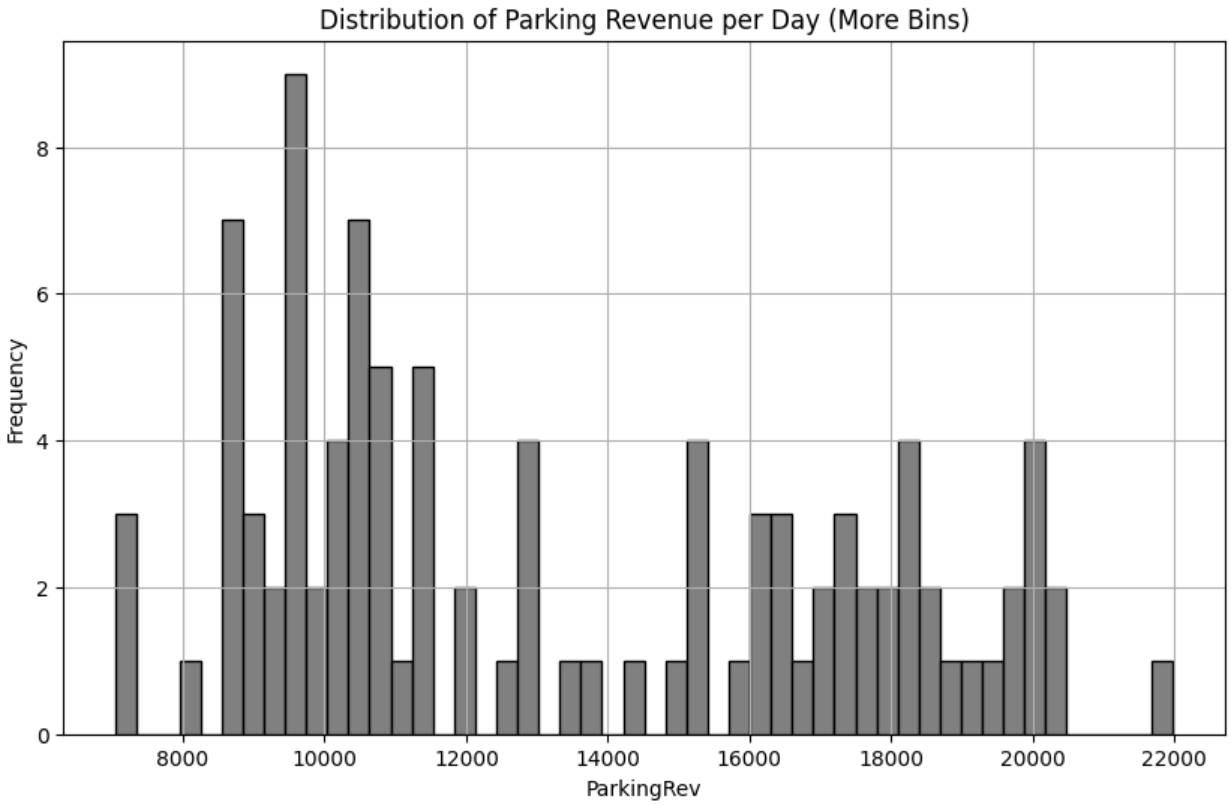



The boxplot shows how the daily gross revenue varies for each day of the week. The widest box, which represents Sunday, means that the revenue on Sundays has the most ups and downs, or variability. On the other hand, Tuesday has the narrowest box, indicating the revenue doesn't change much on that day. Saturdays and Sundays have the highest middle revenue values compared to other days, while Thursdays have the lowest. So, if we look at the middle 50% of revenue values, Saturdays and Sundays are the best days, while Thursdays are not as good.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.hist(lobster['ParkingRev'], bins=20, color='grey',
edgecolor='black')
plt.xlabel('ParkingRev')
plt.ylabel('Frequency')
plt.title('Distribution of Parking Revenue per Day')
plt.grid(True)
```

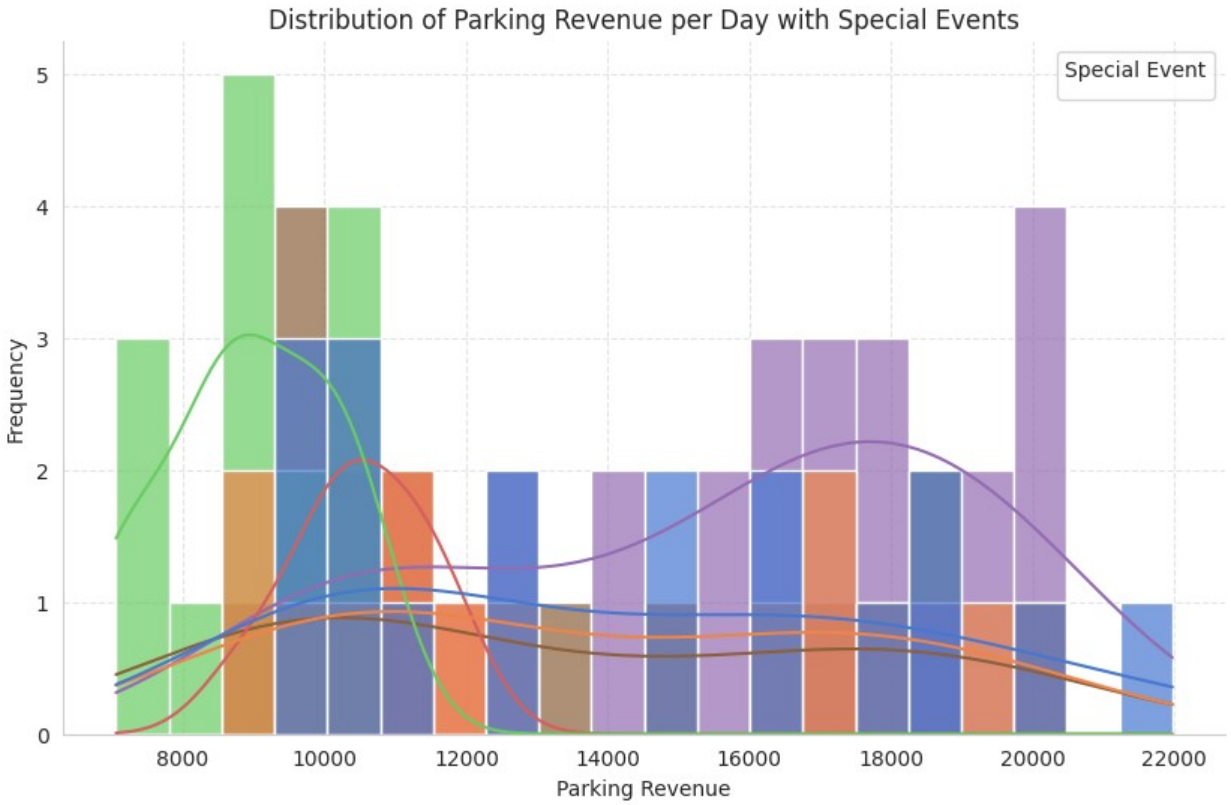



```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.hist(lobster['ParkingRev'], bins=50, color='grey',
edgecolor='black')
plt.xlabel('ParkingRev')
plt.ylabel('Frequency')
plt.title('Distribution of Parking Revenue per Day (More Bins)')
plt.grid(True)
```




```
sns.set_style("whitegrid")

custom_palette = sns.color_palette("muted")

# Plot
plt.figure(figsize=(10, 6))
sns.histplot(data=lobster, x='ParkingRev', bins=20, hue='Spec_Event',
kde=True, palette=custom_palette, alpha=0.7)
plt.xlabel('Parking Revenue')
plt.ylabel('Frequency')
plt.title('Distribution of Parking Revenue per Day with Special
Events')
plt.legend(title='Special Event', loc='upper right')
plt.grid(True, linestyle='--', alpha=0.5)
sns.despine()
plt.show()
```

```
<ipython-input-42-415e0727edfa>:7: UserWarning: The palette list has
more values (10) than needed (6), which may not be intended.
  sns.histplot(data=lobster, x='ParkingRev', bins=20,
hue='Spec_Event', kde=True, palette=custom_palette, alpha=0.7)
WARNING:matplotlib.legend:No artists with labels found to put in
legend.  Note that artists whose label start with an underscore are
ignored when legend() is called with no argument.
```



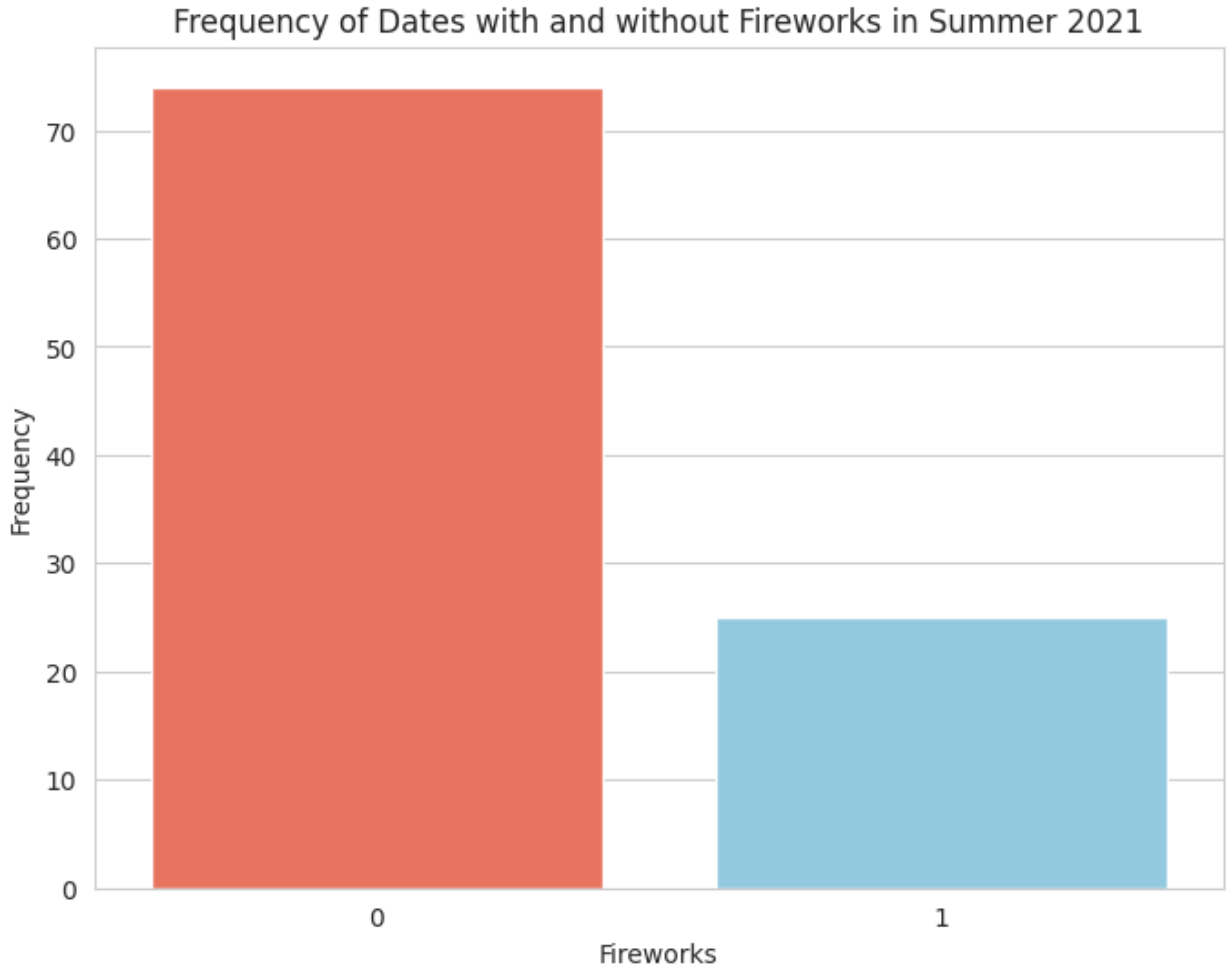

A)Imagine a histogram like a bar chart that shows how many times different values appear in a set of data. By increasing the number of bins in the histogram, we can get a clearer picture of how the data is spread out. When we have fewer bins, the bars are wider, so it gives us a broad idea of the data's distribution. But when we have more bins, the bars are narrower, which gives us a more detailed look at the data. So, the first histogram, with fewer bins, gives us a general overview, while the second one, with more bins, helps us see the smaller differences and patterns in the data more clearly.

B)Adding Special Events as a hue variable in the histogram helps us see how different types of events relate to parking revenue visually. Think of it as coloring the bars in the histogram based on the type of event. This way, we can easily see if certain types of events bring in more or less parking revenue compared to others. By looking at the colored bars, we can understand how events impact parking revenue at LobsterLand. This information can be really helpful for LobsterLand to plan events better and manage parking effectively, as it shows which events attract more visitors and generate more parking revenue.

```
custom_palette = sns.color_palette(["#FF6347", "#87CEEB"])

# Plot
plt.figure(figsize=(8, 6))
sns.countplot(data=lobster, x='Fireworks', palette=custom_palette)
plt.xlabel('Fireworks')
plt.ylabel('Frequency')
plt.title('Frequency of Dates with and without Fireworks in Summer
```

```
2021')
plt.show()
```

```
<ipython-input-52-76c64c9e3d3c>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(data=lobster, x='Fireworks', palette=custom_palette)
```

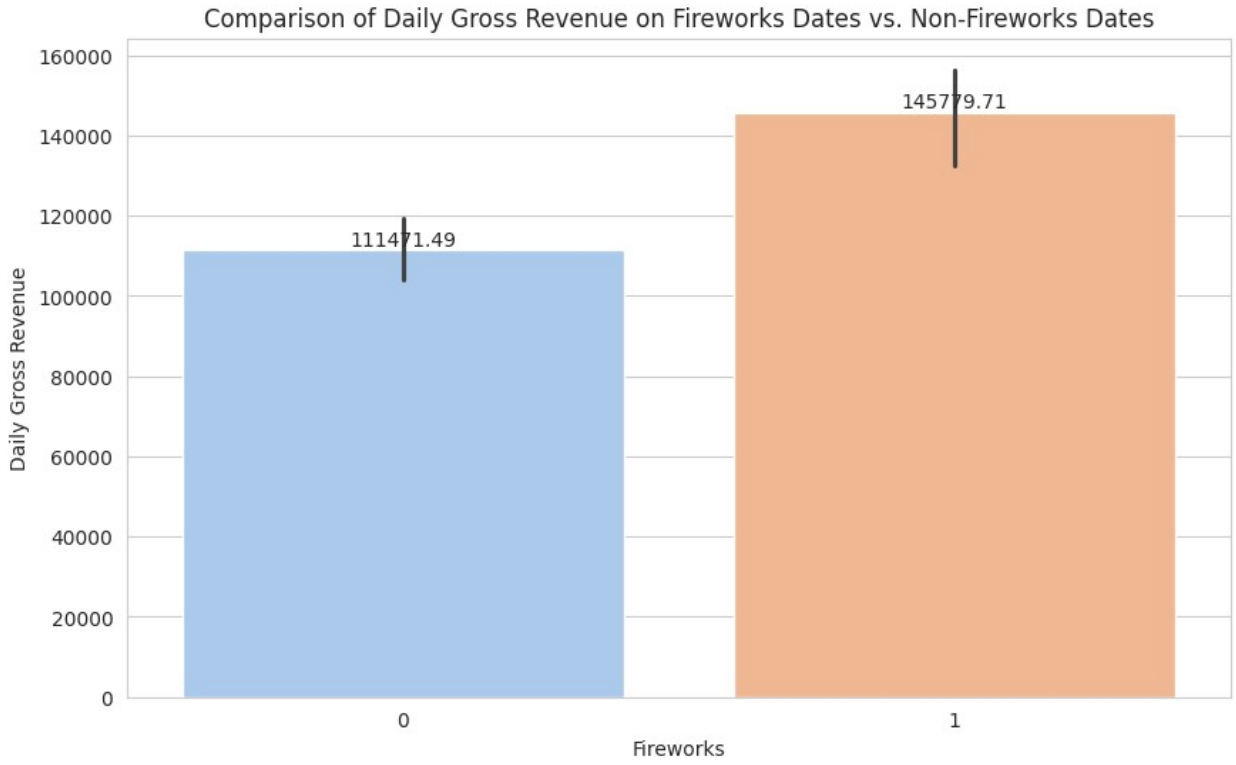



The graph tells us that during summer 2021, there were more days when fireworks happened compared to days when there were no fireworks. It seems like there were about twice as many days with fireworks as there were without. However, remember that this data might not represent all places or all of summer 2021.

```
sns.set_style("whitegrid")
custom_palette = sns.color_palette("pastel")

# Plot
```

```
plt.figure(figsize=(10, 6))
sns.barplot(data=lobster, x='Fireworks', y='DailyGrossRev',
palette=custom_palette)
plt.xlabel('Fireworks')
plt.ylabel('Daily Gross Revenue')
plt.title('Comparison of Daily Gross Revenue on Fireworks Dates vs.
Non-Fireworks Dates')

for i, bar in enumerate(plt.gca().patches):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height,
f'{height:.2f}', ha='center', va='bottom')

plt.show()
```

```
<ipython-input-48-28d982c76e85>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(data=lobster, x='Fireworks', y='DailyGrossRev',
palette=custom_palette)
<ipython-input-48-28d982c76e85>:6: UserWarning: The palette list has
more values (10) than needed (2), which may not be intended.
  sns.barplot(data=lobster, x='Fireworks', y='DailyGrossRev',
palette=custom_palette)
```




The graph compares how much money Lobster Land makes each day when there are fireworks versus when there are no fireworks. It looks like Lobster Land makes more money on days when there are fireworks compared to days when there aren't. Here are some details from the graph:

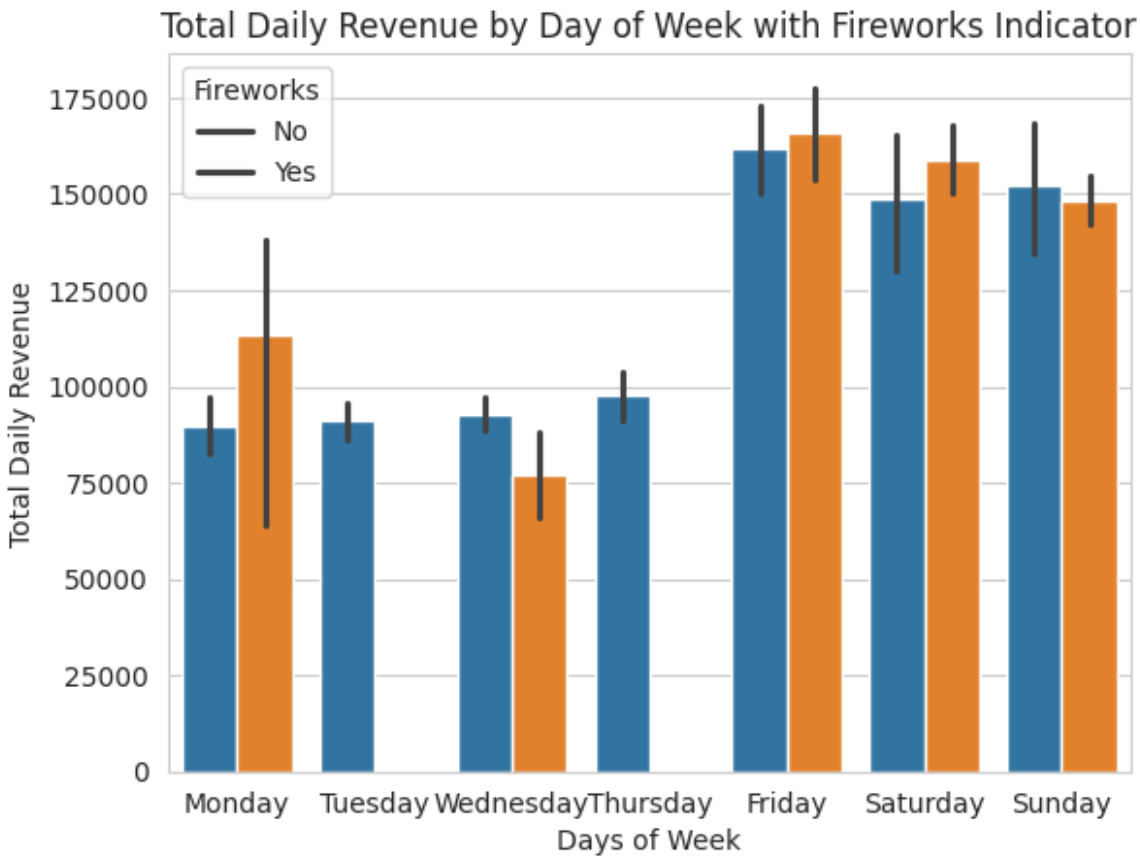
On the best days with fireworks, Lobster Land makes around $160,000.

On the worst days with fireworks, Lobster Land still makes around $111,000.

On the best days without fireworks, Lobster Land makes around $146,000.

On the worst days without fireworks, Lobster Land makes around $20,000

```
sns.barplot(data=lobster, x='Day.of.Week', y='DailyGrossRev',
hue='Fireworks')
plt.xlabel('Days of Week')
plt.ylabel('Total Daily Revenue')
plt.title('Total Daily Revenue by Day of Week with Fireworks
Indicator')
plt.legend(title='Fireworks', loc='upper left', labels=['No', 'Yes'])
plt.show()
```




The graph tells us how much money Lobster Land makes each day of the week and whether there were fireworks on those days. Fridays and Saturdays bring in the most money, with both days making over $150,000.

On the other hand, Sundays and Mondays make the least, each around $50,000. Interestingly, it seems like having fireworks boosts the revenue. Every day that had fireworks made more money compared to days without fireworks. So, it looks like fireworks are connected to higher revenue at Lobster Land.

1. a Increasing Daily Revenue: main aim is to boost the amount of money Lobster Land makes each day by 15% in the next year. This will involve running targeted marketing campaigns, making sure customers have better experiences, and finding ways to make our operations more efficient to hit our revenue target.

b Improving Visitor Satisfaction: goal is to make sure visitors are super happy, aiming for a satisfaction score of 4.5 out of 5 in surveys. work on things like making rides safer, keeping the park clean, and providing top-notch customer service. If want to achieve this high satisfaction score in the next three months.

c Increasing Visitor Engagement: want people to stay longer at Lobster Land, so we're aim for a 20% increase in how long they hang around. To make this happen in the next six months, we'll bring in new rides, entertainment options, and fun stuff for visitors to do.

d Boosting Merchandise Sales: target is to sell more stuff from shops, aiming for a 12% increase in merchandise sales compared to last year.do this by running special promotions, offering more products, and arranging our shop displays to encourage people to buy more.

e Improving Ride Waiting Times: Nobody likes waiting in line, so work to cut down how long people have to wait for rides by 25% in the next three months. This involves things like better planning for when rides are open, having more staff when it's busy, and making queues work smoother.