AD 654 : Marketing Analytics

Boston University

Assignment 1 : Data Exploration & Visualization

For **Parts I & II** of this assignment, you will use the file **lobster_24.csv**, which can be found in Blackboard on our course discussion forum under Class Discussion >> From Your Professor. Th dataset includes information about Lobster Land operations throughout the 2024 seas which began on Memorial Day of this year, and concluded on Labor Day of this year. Part III of th assignment will use the **lobsterland_2024.csv** file, which can also be foundin Blackboard course discussion forum under Class Discussion >> From Your Professor.

Once you have completed this assignment, you will upload three files into Blackboard:

- For **Parts I, II, and III** of this assignment:
  - The .ipynb file that you create in Jupyter Notebook (or Colab)
    - Include a one-paragraph description of your dashboard for Part III.
  - A PDF file that was generated from your .ipynb file
- For **Part III** of this assignment:
  - Either your .twb file (if you used Tableau Desktop) or a link to your dashboard on the Ta server (if you used Tableau Public)

The .ipynb file that you create in Jupyter Notebook (or Colab), and a PDF file that was generated your .ipynb file. If you are using Jupyter Notebook, you may wish to download your work as an .h and then save that file as a PDF. **Please include your last name in your filenames.** The exact save it is up to you, but the last name makes it easier to keep track of the file (e.g. BakerAssignment1.ipynb, bakerAssgn1.ipynb, bakerAssignment1.html, etc. -- any of these v fine). Blackboard will not accept an .html file, which is why you should upload a PDF and an ipyn

For any question that asks you to perform some particular task, you just need to show your input output. Tasks will always be written in regular, non-italicized font.

For any question that asks you to include interpretation, write your answer in a Markdown cell in Notebook (or a 'Text' cell if you use Colab). *Any homework question that needs interpretation will written in italicized font.* Do not simply write your answer in a code cell as a comment, but use a Markdown or Text cell instead.

Remember to be resourceful! There are many helpful resources available to you, including the vi
library, the lecture notes on Blackboard, facilitator sessions, the office hours sessions, and the w

In the prompt, variables might not be referred to in the exact way that their names appear in the
This is okay -- that's very realistic. You should familiarize yourself with the dataset and its variabl
through the dataset description table.

Dataset Description:

Variable	Description
Date	The date, in year-month-date format. Note that each row in the dataset correspon one day of park operations.
Total_Visitors	The total number of visitors that came to Lobster Land on that particular day.
Arcade Visitors	The total number of visitors who came to the Gold Zone (the Lobster Land arcade particular day.
Total Purchases	The total amount of spending by Lobster Land visitors on the particular day.
Arcade Revenue	The total amount of revenue generated by Gold Zone spending on the particular
Total Labor Hours	The total amount of labor hours on the particular day. Each park employee is paid If one employee works a 6 -hour shift, this contributes 6 total labor hours statistic.
Weather	Type A categorical descriptor for the type of weather for the day.
Special Events	This indicates what type of special event (if any) was held that day. Note that days special events show a value of 'NaN' in this column.
Customer Complaints	This indicates the total number of customer complaints, as registered by the Lobs Customer Service Desk, on a particular day of park operations.

|**Day of Week**| This indicates the day of the week. The days are mostly distributed in an even wa
since the season both starts and ends on Mondays, there is one more Monday (compared to the
days of the week) |**Passholder Percentage**| This indicates the percentage of visitors per day wh
Lobster Land passholders.| |**International Visitors**| This indicates the number of visitors per day
international visitors. |**High Temperature**| This is the high temperature per day| |**General Weath
is a categorical descriptor for the type of weather for the day.| |**Precipitation**| This column takes
1 or a 0 value to indicate whether any precipitation occurred during that 24 -hour p

Your Tasks:

Bring the **lobster** 24 **.csv** dataset into your local environment (in Jupyter Notebook, or in Colab

Part I - Exploratory Data Analysis: Exploration Manipulation

A. Call the `head()` function on this dataframe and look at your results.

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set display options
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)

# Load the dataset
df = pd.read_csv('lobster24.csv')
print("Dataset loaded successfully!")
print(f"Shape: {df.shape}")
df.head()
```

```
Dataset loaded successfully!
Shape: (99, 15)
```

Out[2]:

	Date	Total_Visitors	Arcade_Visitors	Total_Purchases	Arcade_Reven
0	5/27/2024	1060	267	22575.71	3524.3
1	5/28/2024	1494	552	17142.52	1652.9
2	5/29/2024	1330	447	10229.99	3251.8
3	5/30/2024	1295	442	6442.65	867.1
4	5/31/2024	1838	256	28409.06	747.6

B. *How many rows of the dataset are visible in Jupyter now?*

There are 5 visible rows

C. Take a look at the dataset's shape attribute.

- a. *How many rows, and how many columns, are in this entire dataframe?*

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Date                    99 non-null     object
 1   Total_Visitors          99 non-null     int64
 2   Arcade_Visitors         99 non-null     int64
 3   Total_Purchases         99 non-null     float64
 4   Arcade_Revenue          99 non-null     float64
 5   Total_Labor_Hours       99 non-null     float64
 6   Weather_Type            99 non-null     object
 7   Special_Events          21 non-null     object
 8   Customer_Complaints     99 non-null     int64
 9   Day_of_Week             99 non-null     object
 10  Passholder_Percentage   99 non-null     float64
 11  International_Visitors  99 non-null     int64
 12  High_Temperature        96 non-null     float64
 13  General_Weather         99 non-null     object
 14  Precipitation           99 non-null     int64
dtypes: float64(5), int64(5), object(5)
memory usage: 11.7+ KB
```

There are `99` rows and `15` columns in the dataframe

D. Read the dataset description, and take a look at the variables in the dataset.

- a. *Which of your variables should be seen as categorical, and which ones should be seen a numeric?*

CATEGORICAL VARIABLES:

- Date
- Weather_Type
- Special_Events
- Day_of_Week
- General_Weather

NUMERIC VARIABLES:

- Total_Visitors
- Arcade_Visitors
- Total_Purchases
- Arcade_Revenue
- Total_Labor_Hours
- Customer_Complaints
- Passholder_Percentage
- International_Visitors
- High_Temperature
- Precipitation

- a. *Are there any NaN values in this dataset?* Show the code that you used to answer this qu
  Which columns have NaN values?

In [4]:
```
missing = df.isna().sum().sort_values(ascending=False)
missing[missing > 0]
```

Out[4]:
```
Special_Events      78
High_Temperature     3
dtype: int64
```

There are missing values in Special_Events (7 8) and High_Temperature (3)

- b. Take a close look at the dataset description. *Is there a variable in this dataset that contain*
  values, but for which the dataset could be more clearly descriptive / informative about the ac
  values? If so, change the 'NaN' values in that column to something that's more descriptive.

In [5]:
```
df.describe()
```

Out[5]:

	Total_Visitors	Arcade_Visitors	Total_Purchases
count	99.000000	99.000000	99.000000
mean	1399.737374	408.050505	15712.011818
std	651.157801	182.460137	5737.163102
min	221.000000	51.000000	5271.300000
25%	924.500000	268.000000	11069.275000
50%	1380.000000	426.000000	16028.240000
75%	1924.500000	559.500000	19003.970000
max	2478.000000	698.000000	28599.960000

- c. *Is there a column in the dataset with some NaN values, for which the true values could be*
  found? (keep in mind that Lobster Land is based just outside of Portland, Maine). If so, find t
  actual values that are missing, and replace them in the dataset. *Which values did you use fo*
  NaN replacement?

In [6]:
```
df['Special_Events'] = df['Special_Events'].fillna('No Special Event')

print("\nAfter change - Special_Events value counts:")
print(df['Special_Events'].value_counts(dropna=False))
```

```
After change - Special_Events value counts:
Special_Events
No Special Event     78
Arcade Tournament     6
Fireworks             6
Food Festival         5
Music Festival        4
Name: count, dtype: int64
```

The Special_Events column had NaN values for days without special events and instead of show
'NaN', I replaced them with 'No Special Event'

In [7]:
```
print("\nRows with missing High_Temperature:")
missing_temp = df[df['High_Temperature'].isnull()]
print(missing_temp[['Date', 'High_Temperature', 'General_Weather']])
```

```
Rows with missing High_Temperature:
         Date  High_Temperature General_Weather
22  6/18/2024               NaN    Rainy/Stormy
44  7/10/2024               NaN    Rainy/Stormy
67   8/2/2024               NaN            Mild
```

Actual temperatures from Portland, Maine wea records

Source: Extreme Weather Watch (NOAA data)

https://www.extremeweatherwatch.com/cities/portland-me/year- 2 0 2 4

In [8]:
```
df.loc[df['Date'] == '6/18/2024', 'High_Temperature'] = 85.0
df.loc[df['Date'] == '7/10/2024', 'High_Temperature'] = 87.0
df.loc[df['Date'] == '8/2/2024', 'High_Temperature'] = 88.0

print("\nMissing temperatures filled with actual values from Extreme Weat
print("  - 6/18/2024: 85°F")
print("  - 7/10/2024: 87°F")
print("  - 8/2/2024: 88°F")

# Verify the fill
print("\nVerification:")
print(df[df['Date'].isin(['6/18/2024', '7/10/2024', '8/2/2024'])][['Date'
```

```
Missing temperatures filled with actual values from Extreme Weather Watch (NO
data):
  - 6/18/2024: 85°F
  - 7/10/2024: 87°F
  - 8/2/2024: 88°F

Verification:
         Date  High_Temperature Weather_Type
22  6/18/2024              85.0        Rainy
44  7/10/2024              87.0       Stormy
67   8/2/2024              88.0        Sunny
```

The High_Temperature column had 3 missing values. Since Lobster Land is located near F
Maine, I looked up the actual historical temperature data from Extreme Weather Watch (https://

www.extremeweatherwatch.com/cities/portland-me/year-2024), which uses NOAA data, fo
Portland, Maine on these specific dates:

- June 18, 2024: 85°F
- July 10, 2024: 87°F
- August 2, 2024: 88°F

These values have been filled into the dataset.

F. Lobster Land management just realized that they want to change something ir
dataset! They think it would make more sense to see the day of the week columr
appear next to the 'Date' column. Right now, the 'Date' column is to the far left of
dataframe, but the day of the week is somewhere else.

- a. Use some Python code to move the day of the week column to put it immediately to the ri
'Date' column.

In [9]:
```
cols = df.columns.tolist()
cols.remove('Day_of_Week')
date_index = cols.index('Date')
cols.insert(date_index + 1, 'Day_of_Week')
df = df[cols]

print("\nNew column order (first 5):")
print(list(df.columns[:5]))
```

```
New column order (first 5):
['Date', 'Day_of_Week', 'Total_Visitors', 'Arcade_Visitors', 'Total_Purchases
```

- b. Next, rename the 'Total_Purchases' variable to instead be named 'Total_Revenue.'

In [10]:
```
df = df.rename(columns={'Total_Purchases': 'Total_Revenue'})

print("\nRenamed Total_Purchases to Total_Revenue")
print("Revenue columns:", [col for col in df.columns if 'Revenue' in col]
```

```
Renamed Total_Purchases to Total_Revenue
Revenue columns: ['Total_Revenue', 'Arcade_Revenue']
```

G. Lobster Land wants to better understand the way in which management allocates hours for th
workers. Specifically, management wants to know about efficiency. Create a new variable called
'staff_efficiency'. This variable should be found by taking total revenue and dividing by total staff

In [11]:
```
df['staff_efficiency'] = df['Total_Revenue'] / df['Total_Labor_Hours']

print("Sample of staff_efficiency values:")
print(df[['Date', 'Total_Revenue', 'Total_Labor_Hours', 'staff_efficiency
```

```
Sample of staff_efficiency values:
        Date  Total_Revenue  Total_Labor_Hours  staff_efficiency
0  5/27/2024       22575.71             185.84        121.479283
1  5/28/2024       17142.52             250.90         68.324113
2  5/29/2024       10229.99             115.59         88.502379
3  5/30/2024        6442.65             347.30         18.550677
4  5/31/2024       28409.06             234.62        121.085415
```

- a. Now that you have created this variable, sort the dataset by staff_efficiency, from highest lowest.

In [12]:
```
# Sort by efficiency (highest to lowest)
df_sorted = df.sort_values('staff_efficiency', ascending=False)
print("\nDataset sorted by staff_efficiency")
```

```
Dataset sorted by staff_efficiency
```

- b. Isolate the 10 most efficient days from the 2024 season, and take a loo them. *What can you say about these 10 days? Do they look different from the overall or do they instead look like a random sample of the rest of the days?* Answer this in 3 - 5 sentences, and use any evidence from the dataset as you see fit.

In [13]:
```
top_10_efficient = df_sorted.head(10)
print("\nTop 10 Most Efficient Days:")
print(top_10_efficient[['Date', 'Day_of_Week', 'Weather_Type', 'Special_E
                        'Total_Revenue', 'Total_Labor_Hours', 'staff_effi
```

```
Top 10 Most Efficient Days:
         Date Day_of_Week Weather_Type     Special_Events  Total_Revenue  \
36   7/2/2024     Tuesday        Rainy          Fireworks       19604.96
97   9/1/2024      Sunday        Sunny      Food Festival       18142.97
0   5/27/2024      Monday        Sunny  Arcade Tournament       22575.71
4   5/31/2024      Friday        Sunny  Arcade Tournament       28409.06
69   8/4/2024      Sunday       Stormy   No Special Event       17466.68
46  7/12/2024      Friday        Sunny   No Special Event       14499.29
56  7/22/2024      Monday       Stormy          Fireworks       12301.13
52  7/18/2024    Thursday       Stormy   No Special Event       14544.99
18  6/14/2024      Friday        Rainy     Music Festival       26427.39
30  6/26/2024   Wednesday       Cloudy   No Special Event       19034.52

    Total_Labor_Hours  staff_efficiency
36             139.99        140.045432
97             133.16        136.249399
0              185.84        121.479283
4              234.62        121.085415
69             158.64        110.102622
46             136.68        106.082016
56             122.35        100.540499
52             145.39        100.041200
18             269.39         98.100857
30             195.29         97.467971
```

The top 10 most efficient days reveal clear differences from the broader dataset. Many of high efficiency days coincide with special events especially Music Festivals, Fireworks shows, ar Festivals, which generate strong revenue without a proportional rise in labor hours. Weather con vary across these days, indicating that well designed events can offset less than ideal weather. E

is primarily driven by high revenue levels (2 8 K) paired with tightly managed labor hours (belo
3 0 0), suggesting either intentional staffing optimization or naturally high demand. Overall, th
highlights strategic event planning as a powerful lever for boosting operational efficiency.

H. Create a new variable called 'international_percentage'. This variable should i
the percentage of all visitors who were international, on any given day (so for ins
this would show ' 5 0 ' if there were 4 0 0 0 total visitors on a given day
2 0 0 0 of them were international).

In [14]:
```
# Create international_percentage variable
df['international_percentage'] = (df['International_Visitors'] / df['Tota

print("Sample of international_percentage:")
print(df[['Date', 'Total_Visitors', 'International_Visitors', 'internatio
```

```
Sample of international_percentage:
        Date  Total_Visitors  International_Visitors  international_percentag
0  5/27/2024            1060                     357                 33.67924
1  5/28/2024            1494                     424                 28.38018
2  5/29/2024            1330                     441                 33.15789
3  5/30/2024            1295                     338                 26.10038
4  5/31/2024            1838                     615                 33.46028
5   6/1/2024            2369                     859                 36.26002
6   6/2/2024             666                     259                 38.88888
7   6/3/2024            1438                     474                 32.96244
8   6/4/2024             530                     140                 26.41509
9   6/5/2024            1682                     459                 27.28894
```

- a. Next, find the correlation between international percentage and passholder percentage. *W*
  this correlation? Is it positive or negative? Strong, moderate, or weak? Next, take a moment
  reason as to 'why' this relationship might look the way it does – what could explain it?

In [15]:
```
# First convert Passholder_Percentage to actual percentage (multiply by 1
correlation = df['international_percentage'].corr(df['Passholder_Percenta

print(f"\nCorrelation: {correlation:.4f}")

# Interpret correlation
if correlation > 0:
    strength_direction = "positive"
else:
    strength_direction = "negative"

abs_corr = abs(correlation)
if abs_corr > 0.7:
    strength = "strong"
elif abs_corr > 0.4:
    strength = "moderate"
else:
    strength = "weak"

print(f"This correlation is {strength_direction} and {strength}.")
```

```
Correlation: -0.6722
This correlation is negative and moderate.
```

The correlation of -0.67 indicates a strong negative relationship. As international visitors ir passholder percentage tends to decrease. This makes sense because passholders are mostly lo repeat visitors, while international visitors are typically tourists who come during peak periods. As result, days with more tourists tend to have fewer locals, and vice versa. These patterns reflect tv distinct segments locals optimizing for comfort and flexibility, and tourists bound by fixed travel sc

I. Use the `info()` function to determine the data type for the 'Date' variable. *W type of variable does pandas see it as, currently?* Convert this variable into a dat object, and show that the conversion was successful.

In [16]:
```
# Check current data type
print("Current data type of 'Date' column:")
print(f"Date is type: {df['Date'].dtype}")

# Convert to datetime
df['Date'] = pd.to_datetime(df['Date'])

print("\nAfter conversion:")
print(f"Date is now type: {df['Date'].dtype}")
print("\nSample of Date column:")
print(df['Date'].head())
```

```
Current data type of 'Date' column:
Date is type: object

After conversion:
Date is now type: datetime64[ns]

Sample of Date column:
0   2024-05-27
1   2024-05-28
2   2024-05-29
3   2024-05-30
4   2024-05-31
Name: Date, dtype: datetime64[ns]
```

- a. What impact could this have on the variable? (In other words, what is the point of having I 'see' that your date variable is truly a date?)

Converting a Date column into a true datetime object lets Python treat it as an actual date instea text. This unlocks easy extraction of things like month or weekday, enables time-series analysis (differences, rolling averages), and ensures accurate sorting and filtering by date. It also improves since visualization tools can format time-based axes correctly.

J. Using the `groupby()` function, along with the describe() function, explore th relationship between day of the week and total revenue. *What stands out here? I sentences, describe anything noteworthy, and also include some speculation abo the results look the way that they do.*

In [17]:
```
revenue_by_day = df.groupby('Day_of_Week')['Total_Revenue'].describe()
print("Revenue statistics by day of week:")
```

```
print(revenue_by_day)
print("\n")
```

```
Revenue statistics by day of week:
             count          mean          std       min         25%  \
Day_of_Week
Friday        14.0  17133.101429  6309.542051   8052.76  12365.9700
Monday        15.0  13525.969333  5161.017456   7624.32   9002.9300
Saturday      14.0  21448.985000  4161.667899  15353.92  18287.3075
Sunday        14.0  18807.897143  4946.299831  11984.34  15227.6475
Thursday      14.0  12125.405000  4653.834452   6056.24   7931.7700
Tuesday       14.0  14155.262143  5128.831358   5271.13  10193.9950
Wednesday     14.0  12943.608571  3422.424946   9183.07   9936.1925

                  50%         75%       max
Day_of_Week
Friday       15628.680  21562.4450  28409.06
Monday       12301.130  18278.0500  22575.71
Saturday     21655.475  23664.5925  28599.96
Sunday       17804.825  21440.6825  28132.69
Thursday     12138.675  16272.5750  19296.08
Tuesday      16138.935  18130.2900  19732.53
Wednesday    12520.310  15416.1950  19034.52
```

Saturday delivers the highest revenue, reflecting classic weekend demand when families and tou have more free time. Sunday is similarly strong. In contrast, Wednesday and Thursday are the sl days, making them ideal for lighter staffing, maintenance, or targeted mid-week promotions. Frid a clear ramp-up as visitors begin their weekend plans.

Weekend revenue also varies more from week to week, likely because weather and special even a bigger impact when baseline attendance is already high.

K. *Which two variables in this dataset are not identical, but mostly redundant?* Id this pair of variables, and remove one of them (but not both!) You will be able to complete all of the subsequent steps, regardless of which one you remove here.

In [18]:
```
print("Examining potentially redundant variables...")
print("\nWeather-related columns:")
print(df[['Weather_Type', 'General_Weather', 'Precipitation']].head(15))

print("\nUnique values in Weather_Type:")
print(df['Weather_Type'].value_counts())

print("\nUnique values in General_Weather:")
print(df['General_Weather'].value_counts())

print("\nCross-tabulation of Weather_Type vs General_Weather:")
print(pd.crosstab(df['Weather_Type'], df['General_Weather']))
```

```
Examining potentially redundant variables...

Weather-related columns:
   Weather_Type General_Weather  Precipitation
0         Sunny            Cool              0
1        Cloudy            Cool              0
2        Cloudy             Hot              0
3        Stormy    Rainy/Stormy              1
4         Sunny            Mild              0
5        Stormy    Rainy/Stormy              1
6         Sunny            Mild              0
7         Sunny            Mild              0
8         Rainy    Rainy/Stormy              1
9         Sunny            Cool              0
10       Stormy    Rainy/Stormy              1
11       Cloudy            Mild              0
12        Sunny            Cool              0
13        Rainy    Rainy/Stormy              1
14       Stormy    Rainy/Stormy              1

Unique values in Weather_Type:
Weather_Type
Sunny     30
Stormy    29
Cloudy    23
Rainy     17
Name: count, dtype: int64

Unique values in General_Weather:
General_Weather
Rainy/Stormy    46
Mild            23
Cool            21
Hot              9
Name: count, dtype: int64

Cross-tabulation of Weather_Type vs General_Weather:
General_Weather  Cool  Hot  Mild  Rainy/Stormy
Weather_Type
Cloudy              9    4    10             0
Rainy               0    0     0            17
Stormy              0    0     0            29
Sunny              12    5    13             0
```

Weather_Type and General_Weather are essentially redundant because they describe the same underlying conditions. Weather_Type focuses on the core attendance drivers sun, clouds, rain, a storms while General_Weather blends temperature (Cool/Mild/Hot) with precipitation. Since temp and precipitation are already captured more precisely in High_Temperature and Precipitation, General_Weather doesn't add meaningful new information.

Weather_Type is more actionable and directly tied to operational decisions, making it the better v to keep.

Decision: Drop General_Weather and retain Weather_Type.

In [19]:
```python
# Remove the redundant variable
df = df.drop('General_Weather', axis=1)
```

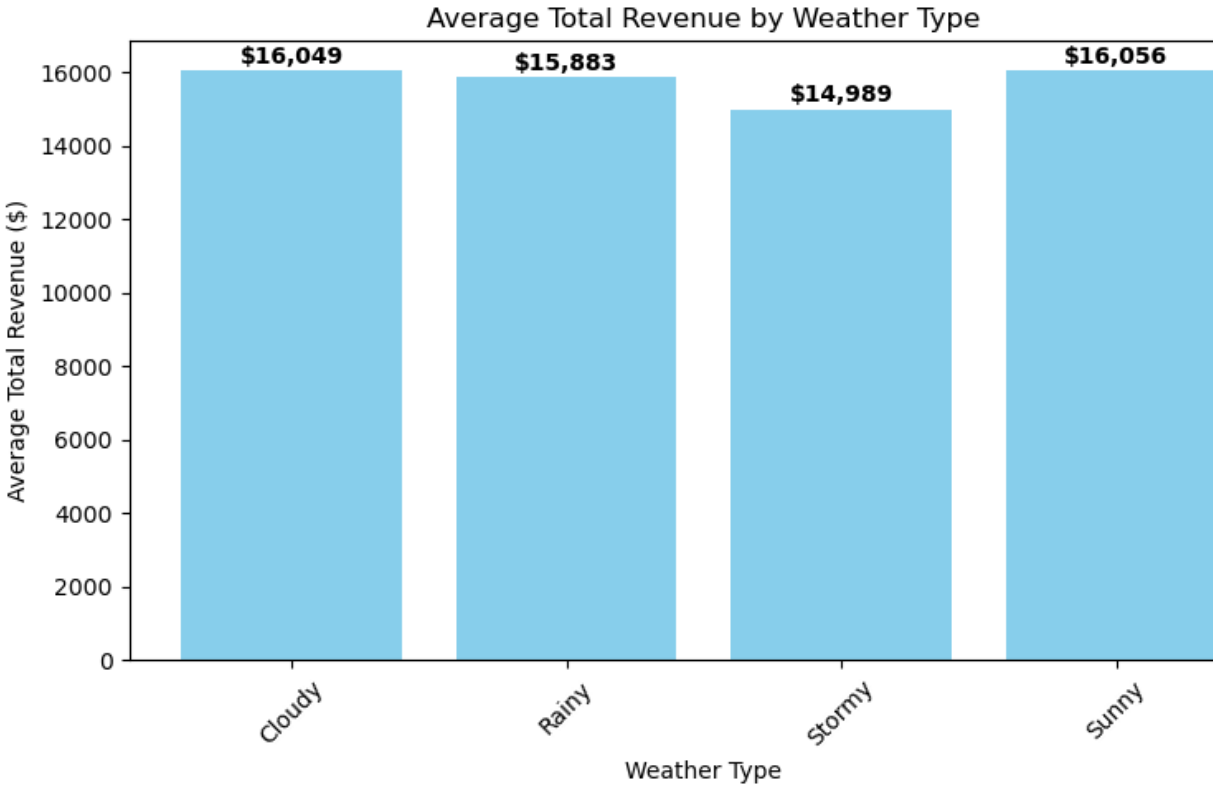
```
print("\nGeneral_Weather column removed.")
print(f"New dataset shape: {df.shape}")
print(f"Now have {df.shape[1]} columns instead of 15")
```

```
General_Weather column removed.
New dataset shape: (99, 16)
Now have 16 columns instead of 15
```

Part II - Data Visualization

L. Using any plotting tool in Python, generate a bar plot that shows the weather t one on one axis, and shows average (mean) total revenue on the other axis. Be give your plot a clear, descriptive title.

In [20]:
```
# Group by Weather_Type and calculate mean revenue
weather_rev = df.groupby('Weather_Type')['Total_Revenue'].mean().reset_in

# Create bar plot
plt.figure(figsize=(8,5))
plt.bar(weather_rev['Weather_Type'], weather_rev['Total_Revenue'], color=

# Title and labels
plt.title('Average Total Revenue by Weather Type')
plt.xlabel('Weather Type')
plt.ylabel('Average Total Revenue ($)')
plt.xticks(rotation=45)

# Add value labels on top of bars
for i, (weather, value) in enumerate(weather_rev.iterrows()):
    plt.text(i, value['Total_Revenue'] + 200, f'${value["Total_Revenue"]:
             ha='center', fontweight='bold')

# Sort bars by revenue (highest to lowest)
weather_rev = weather_rev.sort_values('Total_Revenue', ascending=False)

plt.tight_layout()
plt.show()
```

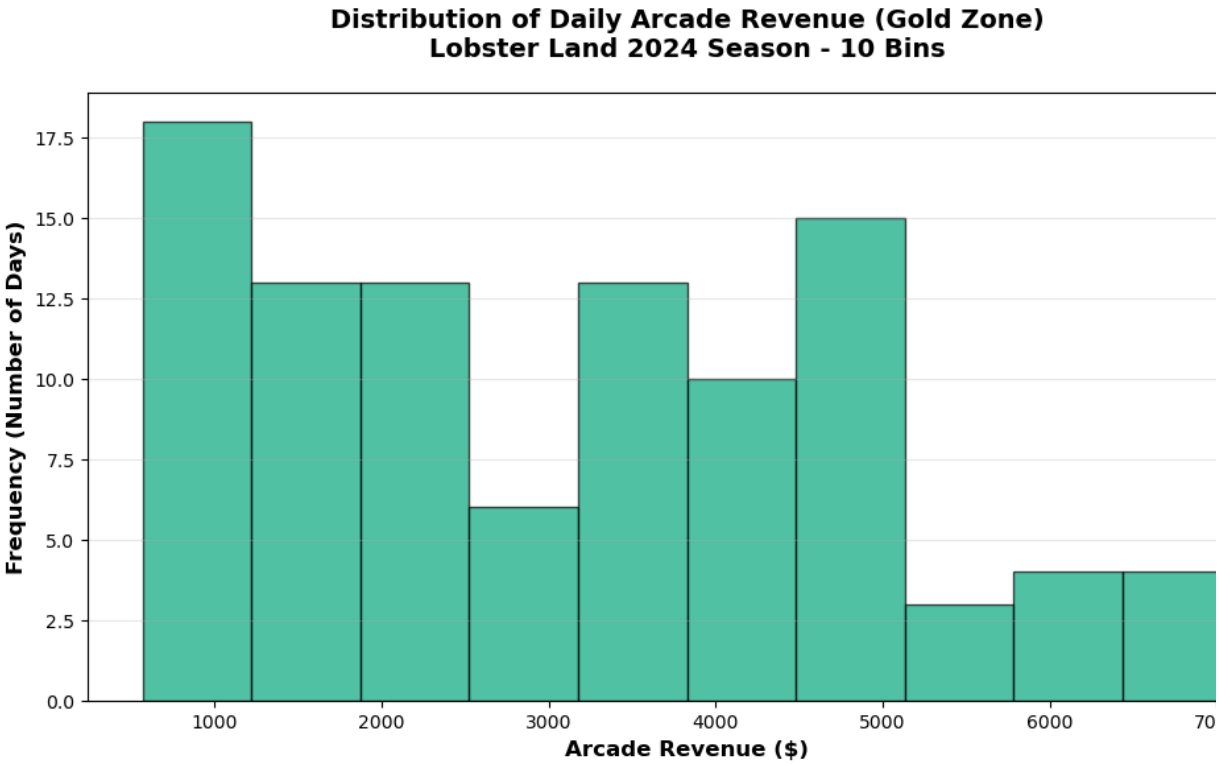


- a. *What do you notice about this plot? Does it surprise you, or does it align with your expecta*
  a couple of sentences, how would you describe what you see here?

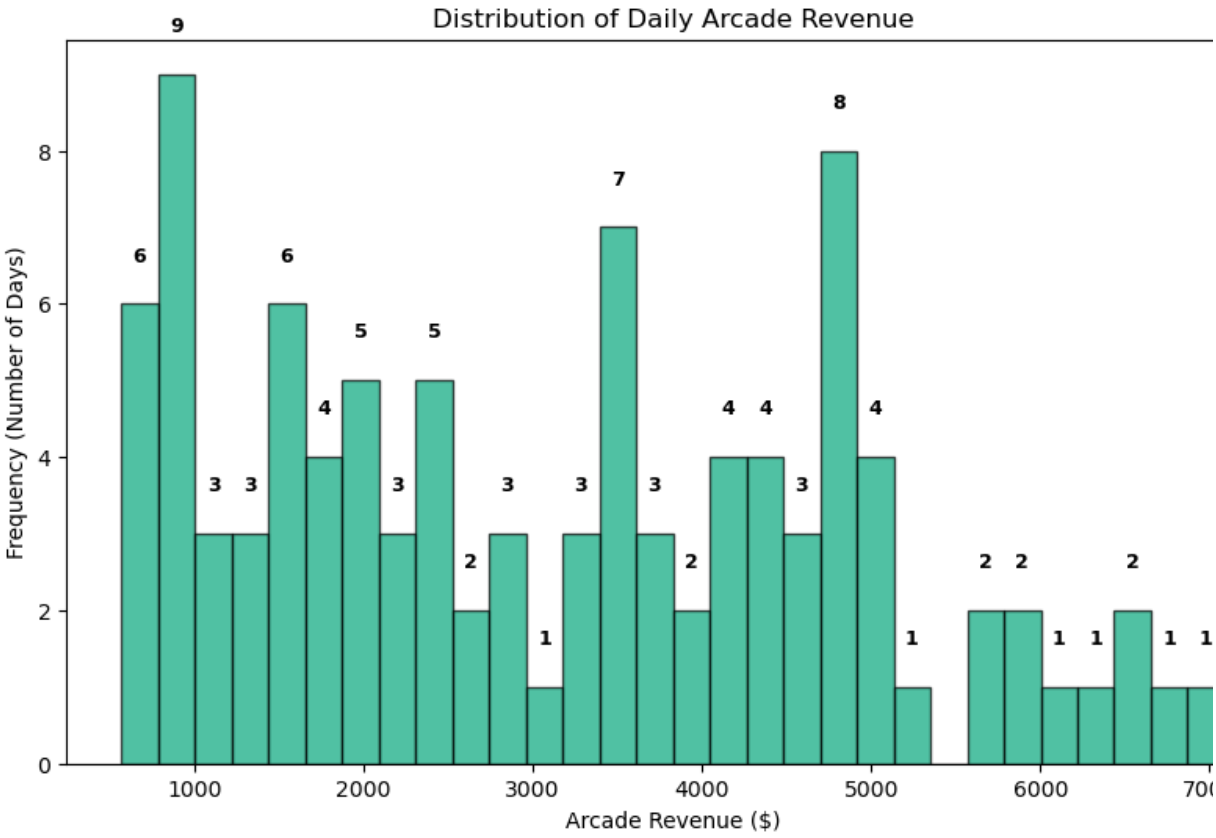
Comparing 10 bins to 30 bins makes the impact of bin size much clearer. With bins, the histogram looks smoother and hides many of the smaller fluctuations in arcade revenue increase to 30 bins, the distribution becomes much more detailed, revealing small peaks that were not visible before. This shows how larger bin counts can expose day to day variability, smaller bin counts smooth the data into broader pattern

M. Next, generate a histogram that depicts total spending at the Gold Zone (the Land arcade) across the days of the 2024 season. Be sure to give you clear, descriptive title.

In [21]:
```
plt.figure(figsize=(10, 6))
plt.hist(df['Arcade_Revenue'], bins=10, color='#06A77D', edgecolor='black
plt.xlabel('Arcade Revenue ($)', fontsize=12, fontweight='bold')
plt.ylabel('Frequency (Number of Days)', fontsize=12, fontweight='bold')
plt.title('Distribution of Daily Arcade Revenue (Gold Zone)\nLobster Land
          fontsize=14, fontweight='bold', pad=20)
plt.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.show()
```



In [22]:
```
plt.figure(figsize=(10, 6))
n, bins, patches = plt.hist(df['Arcade_Revenue'], bins=30,
                            color='#06A77D', edgecolor='black', alpha=0.

# Add count labels on top of each bin
for i in range(len(patches)):
    height = patches[i].get_height()
    if height > 0:  # Only label non-zero bars
        plt.text(patches[i].get_x() + patches[i].get_width()/2, height +
                 f'{int(height)}',  # Shows count
                 ha='center', va='bottom', fontweight='bold', fontsize=9)

plt.xlabel('Arcade Revenue ($)')
plt.ylabel('Frequency (Number of Days)')
plt.title('Distribution of Daily Arcade Revenue')
plt.show()
```



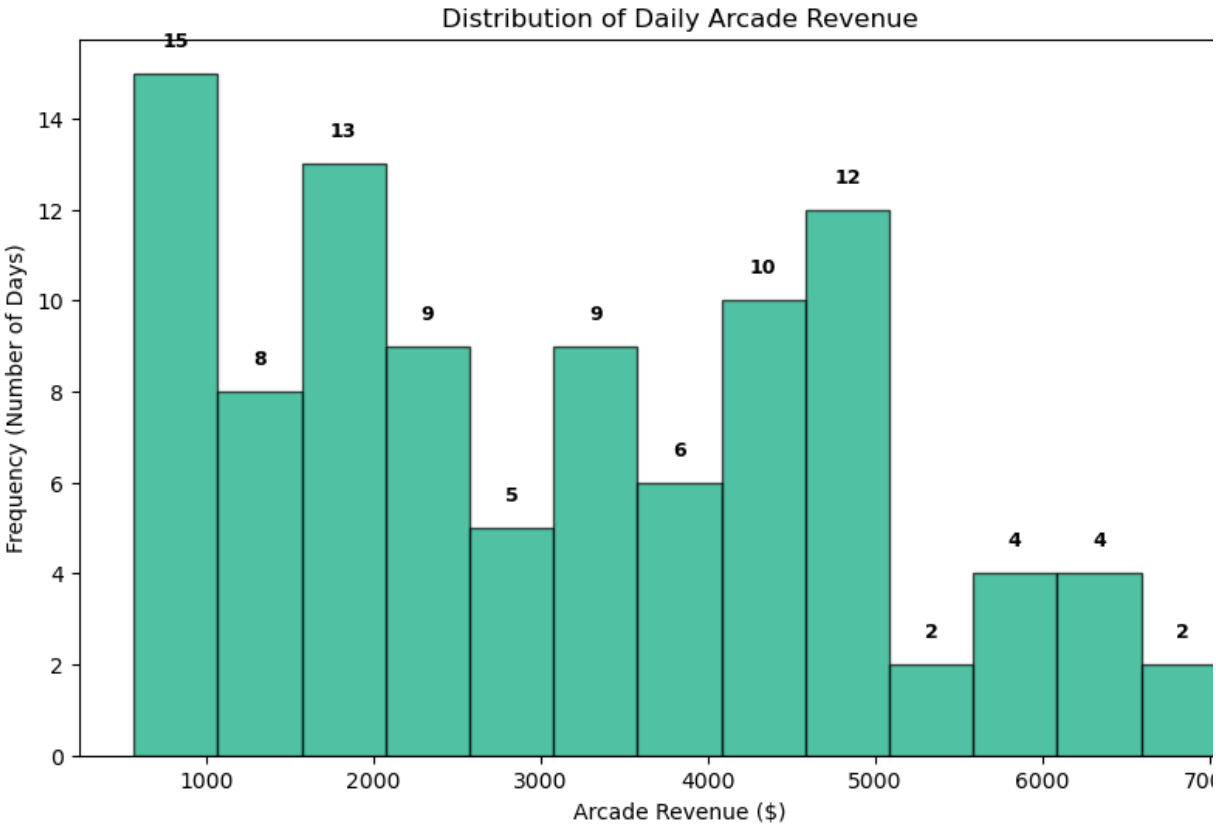
- a. *In a couple of sentences, describe your plot. What does it show you about the distribution arcade spending in the* `2024` *season?*

Comparing `10` bins to `30` bins makes the impact of bin size much clearer. With bins, the histogram looks smoother and hides many of the smaller fluctuations in arcade revenue increase to `30` bins, the distribution becomes much more detailed, revealing small peaks that were not visible before. This shows how larger bin counts can expose day-to-day variability, smaller bin counts smooth the data into broader patterns.

- b. Now, increase the number of bins. *What can you say about this second histogram, compa the first one? What impact can changing the number of bins (either higher or lower) have on histogram?*

In [23]:
```
plt.figure(figsize=(10, 6))
n, bins, patches = plt.hist(df['Arcade_Revenue'], bins=13,
                            color='#06A77D', edgecolor='black', alpha=0.

# Add count labels on top of each bin
for i in range(len(patches)):
    height = patches[i].get_height()
    if height > 0:  # Only label non-zero bars
        plt.text(patches[i].get_x() + patches[i].get_width()/2, height +
                 f'{int(height)}',  # Shows count
                 ha='center', va='bottom', fontweight='bold', fontsize=9)

plt.xlabel('Arcade Revenue ($)')
plt.ylabel('Frequency (Number of Days)')
```

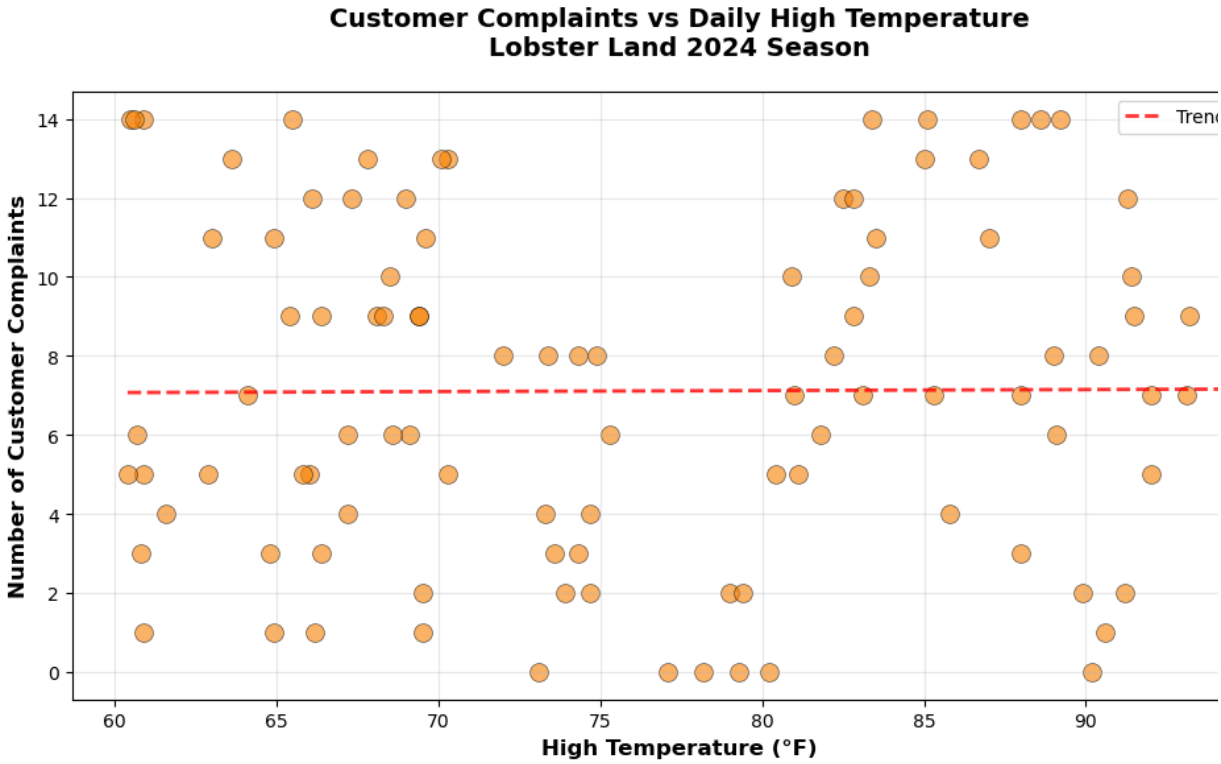
```
plt.title('Distribution of Daily Arcade Revenue')
plt.show()
```



The 13-bin histogram reveals more small clusters than the 10-bin version, showing tha revenue varies across many levels rather than just “low vs. high.” This added detail makes the pa less smooth, so fewer bins are better for big-picture storytelling while more bins are useful for de analytical work.

N. One member of the Lobster Land Board of Directors thinks that people tend to complain more on hot days. Let’s see whether the data supports this theory! Ger scatterplot with the day’s high temperature on the x-axis, and with the number of customer complaints on the y-axis. Be sure to give your plot a clear, descriptive t

In [24]:
```
plt.figure(figsize=(10, 6))
plt.scatter(df['High_Temperature'], df['Customer_Complaints'],
            alpha=0.6, s=100, color='#F77F00', edgecolor='black', linewid
plt.xlabel('High Temperature (°F)', fontsize=12, fontweight='bold')
plt.ylabel('Number of Customer Complaints', fontsize=12, fontweight='bold
plt.title('Customer Complaints vs Daily High Temperature\nLobster Land 20
          fontsize=14, fontweight='bold', pad=20)
plt.grid(True, alpha=0.3)

# Add a trend line
z = np.polyfit(df['High_Temperature'], df['Customer_Complaints'], 1)
p = np.poly1d(z)
plt.plot(df['High_Temperature'].sort_values(),
         p(df['High_Temperature'].sort_values()),
         "r--", alpha=0.8, linewidth=2, label='Trend Line')
plt.legend()

plt.tight_layout()
plt.show()
```



- a. *What does the scatterplot show? If there is a relationship between these variables, what r*
  explain it? If there is not a relationship between these variables, why do you think this is the

The scatterplot indicates almost no relationship between temperature and customer complaints (correlation ≈ {correlation: 0 . 1 2 }); the points are widely dispersed, showing that hotter days c reliably produce more complaints than cooler ones. This directly challenges the Board member's assumption that heat drives dissatisfaction. Several factors could explain the lack of a temperatu

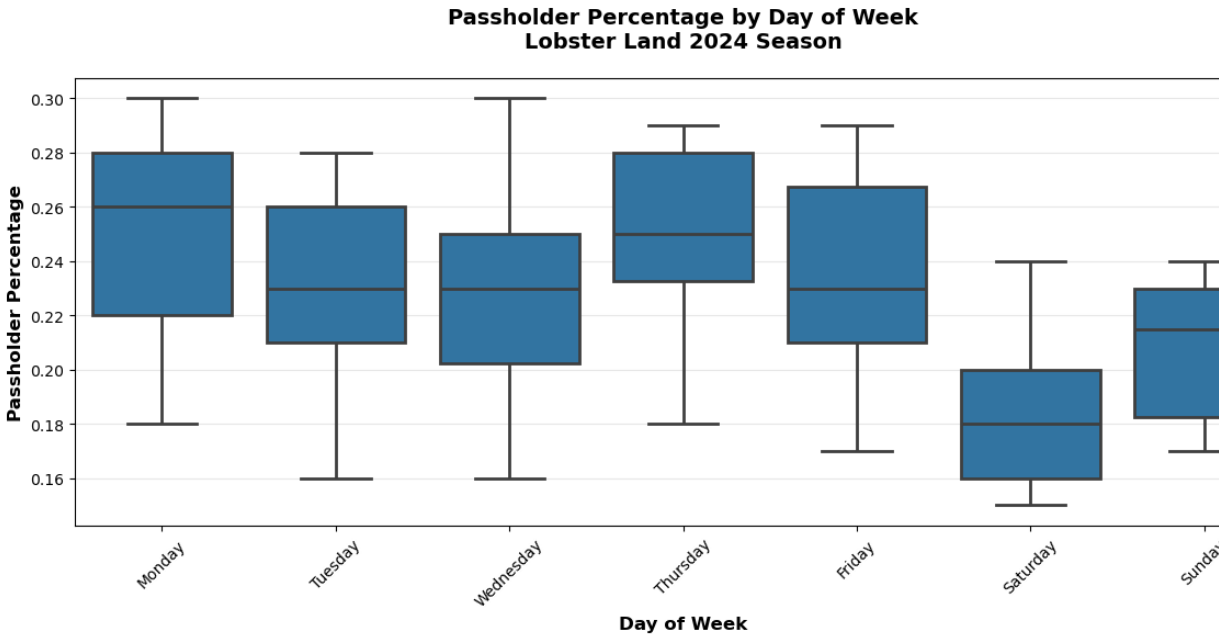
1 . Comfort Management: Cooling stations, shade, and water features may effectively offset hea discomfort.

2. Operational Drivers: Complaints likely stem from controllable issues such as wait times, staf interactions, or food quality rather than weather.
3. Behavioral Adjustment: On very hot days, guests may modify their activities or shorten visits of complaining.
4. Expectation Setting: Summer visitors anticipate heat and arrive prepared, making them less to temperature than to operational problems.

Operationally, this points toward prioritizing improvements in service quality and guest experienc than expanding weather-related amenities.

O. Generate a boxplot that shows day of week on one axis, and passholder perc on the other axis. Be sure to give your plot a clear, descriptive title.

In [25]:
```
# Order days of week properly
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Sat
df['Day_of_Week'] = pd.Categorical(df['Day_of_Week'], categories=day_orde

plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='Day_of_Week', y='Passholder_Percentage', linewidt

plt.xlabel('Day of Week', fontsize=12, fontweight='bold')
plt.ylabel('Passholder Percentage', fontsize=12, fontweight='bold')
plt.title('Passholder Percentage by Day of Week\nLobster Land 2024 Season
          fontsize=14, fontweight='bold', pad=20)
plt.xticks(rotation=45)
plt.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.show()

print("\nPassholder Percentage by Day of Week:")
df.groupby('Day_of_Week', observed=False)['Passholder_Percentage'].descri
```




```
Passholder Percentage by Day of Week:
```

Out[25]:

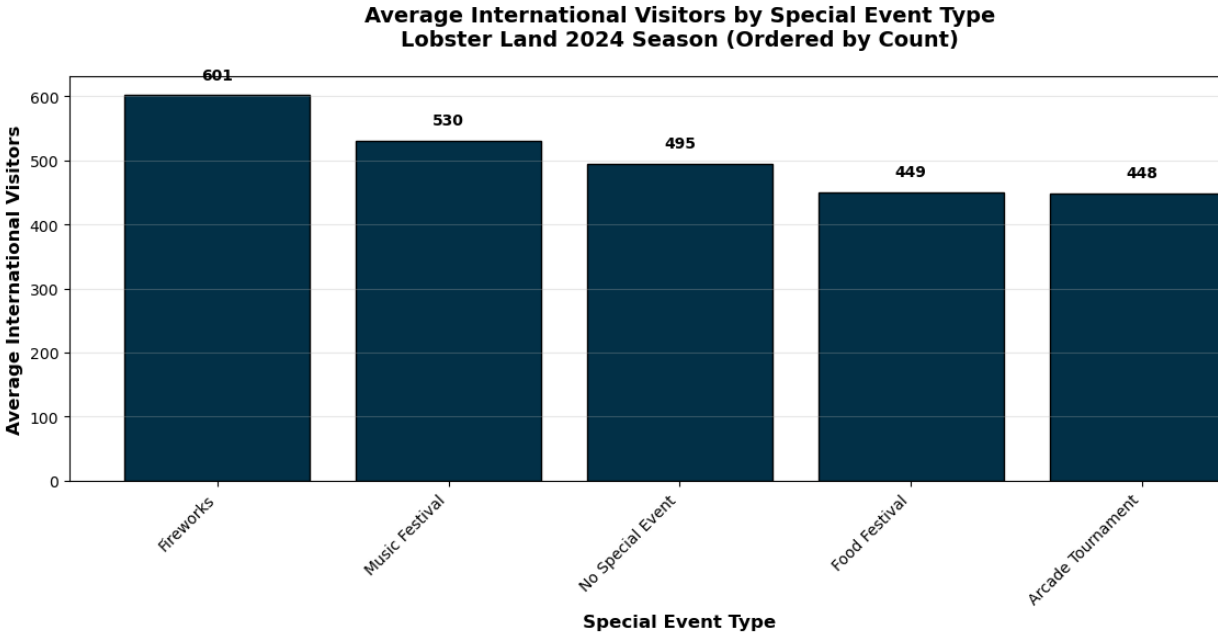
Day_of_Week	count	mean	std	min	25%
Monday	15.0	0.251333	0.038889	0.18	0.2200
Tuesday	14.0	0.229286	0.036682	0.16	0.2100
Wednesday	14.0	0.230714	0.041037	0.16	0.2025
Thursday	14.0	0.247143	0.039697	0.18	0.2325
Friday	14.0	0.235000	0.038977	0.17	0.2100
Saturday	14.0	0.184286	0.027376	0.15	0.1600
Sunday	14.0	0.208571	0.025678	0.17	0.1825

- a. *In a few sentences, what does this plot show you? Why do you think it looks this way – wh explain this?*

The box plot shows that passholder percentages are higher on weekdays and lower on weekend the weekday values, especially Monday, Tuesday, and Wednesday also vary much more from da Their taller boxes and larger standard deviations indicate that these days are less predictable, wl Saturday and Sunday stay in a tight, stable range even though their percentages are lower. This that weekday attendance is influenced by more fluctuating factors (work schedules, school calen local behavior), whereas weekend patterns are more consistent.

P. Generate a barplot that shows Special Events on one axis, and international v on the other axis. Do not include confidence intervals ('error bars') on your plot. (your bars either from shortest to tallest, or from tallest to shortest. Be sure to giv plot a clear, descriptive title.

In [26]:
```
# Calculate mean international visitors by event type
event_intl = df.groupby('Special_Events')['International_Visitors'].mean(

plt.figure(figsize=(12, 6))
plt.bar(range(len(event_intl)), event_intl.values, color='#023047', edgec
plt.xlabel('Special Event Type', fontsize=12, fontweight='bold')
plt.ylabel('Average International Visitors', fontsize=12, fontweight='bol
plt.title('Average International Visitors by Special Event Type\nLobster
          fontsize=14, fontweight='bold', pad=20)
plt.xticks(range(len(event_intl)), event_intl.index, rotation=45, ha='rig
plt.grid(axis='y', alpha=0.3)

# Add value labels
for i, value in enumerate(event_intl.values):
    plt.text(i, value + 20, f'{value:.0f}',
             ha='center', va='bottom', fontweight='bold', fontsize=10)

plt.tight_layout()
plt.show()

print("\nAverage International Visitors by Event Type:")
print(event_intl)
```




```
Average International Visitors by Event Type:
Special_Events
Fireworks            601.333333
Music Festival       530.500000
No Special Event     494.858974
Food Festival        449.400000
Arcade Tournament    447.833333
Name: International_Visitors, dtype: float64
```

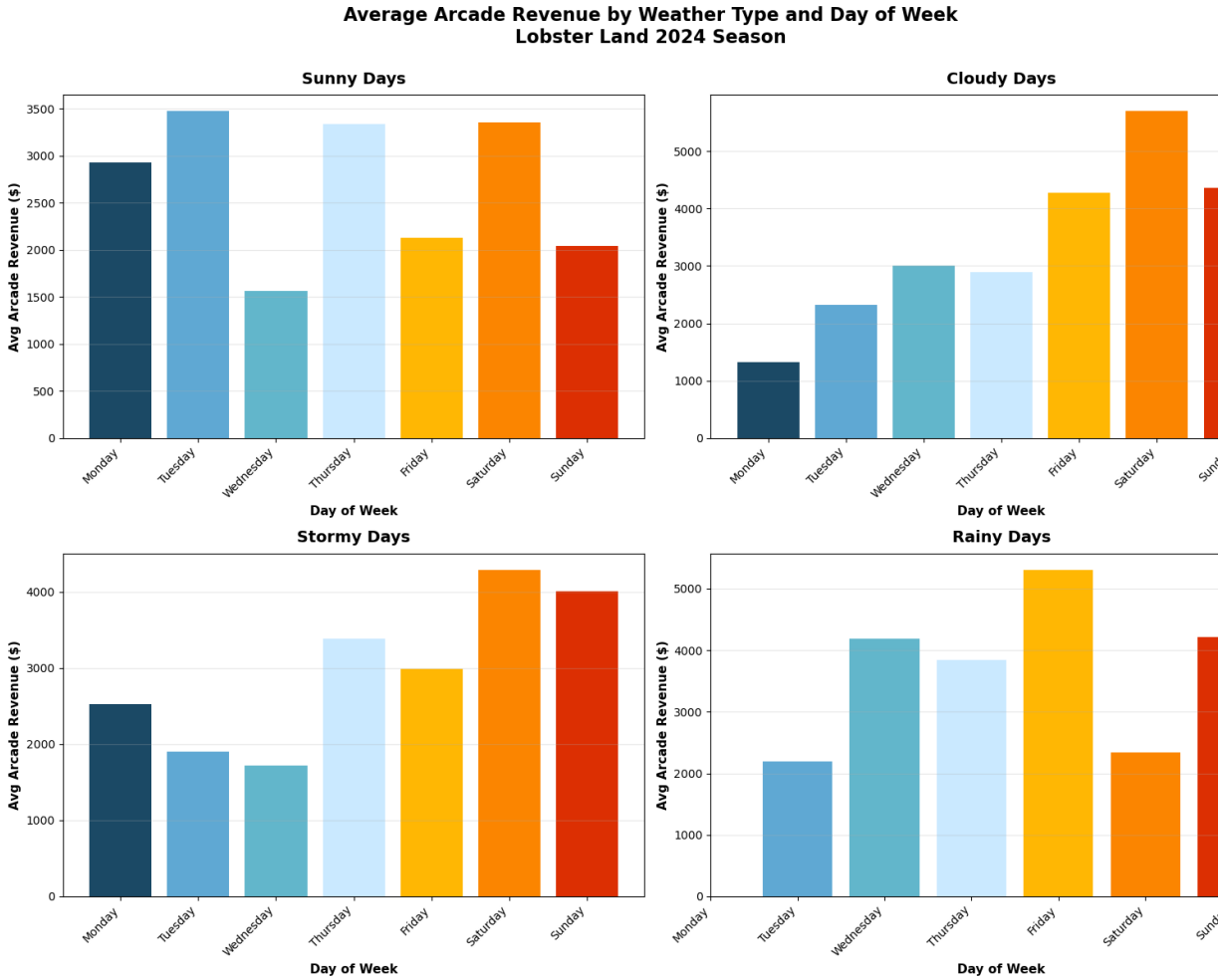
- a. *What does your plot show? Why can we **not** conclude causality based on the relationship here?*

Fireworks events draw the highest average international attendance, suggesting they're a major attraction. However, the "No Special Event" category still shows strong visitor numbers, indicating international guests attend Lobster Land consistently, not just for special events.

Q. Create a faceted bar plot showing arcade revenue, with facets for different we types and bars representing each day of the week. Do not show confidence inter ('error bars') on your plot. Be sure to give your plot a clear, descriptive title.

In [27]:
```
# Create faceted plot
weather_types = df['Weather_Type'].unique()
fig, axes = plt.subplots(2, 2, figsize=(16, 12))
axes = axes.flatten()

for idx, weather in enumerate(weather_types):
    weather_data = df[df['Weather_Type'] == weather]
    day_arcade = (
        weather_data
        .groupby('Day_of_Week', observed=True)['Arcade_Revenue']
        .mean()
        .reindex(day_order)
    )

    axes[idx].bar(
        range(len(day_arcade)),
        day_arcade.values,
```

```
        color=[
            '#1B4965', '#5FA8D3', '#62B6CB', '#CAE9FF',
            '#FFB703', '#FB8500', '#DC2F02'
        ][:len(day_arcade)]
    )
    axes[idx].set_title(f'{weather} Days', fontsize=14, fontweight='bold'
    axes[idx].set_xlabel('Day of Week', fontsize=11, fontweight='bold')
    axes[idx].set_ylabel('Avg Arcade Revenue ($)', fontsize=11, fontweigh
    axes[idx].set_xticks(range(len(day_arcade)))
    axes[idx].set_xticklabels(day_arcade.index, rotation=45, ha='right')
    axes[idx].grid(axis='y', alpha=0.3)


plt.suptitle('Average Arcade Revenue by Weather Type and Day of Week\nLob
             fontsize=16, fontweight='bold', y=1.00)
plt.tight_layout()
plt.show()
```



In [28]:
```
df['Weather_Type'].value_counts()
```

Out[28]:
```
Weather_Type
Sunny     30
Stormy    29
Cloudy    23
Rainy     17
Name: count, dtype: int64
```

- a. *What looks interesting here? Are there some day of week/weather combinations that seer particularly noteworthy? Also, what's an important limitation to note when interpreting this gra*

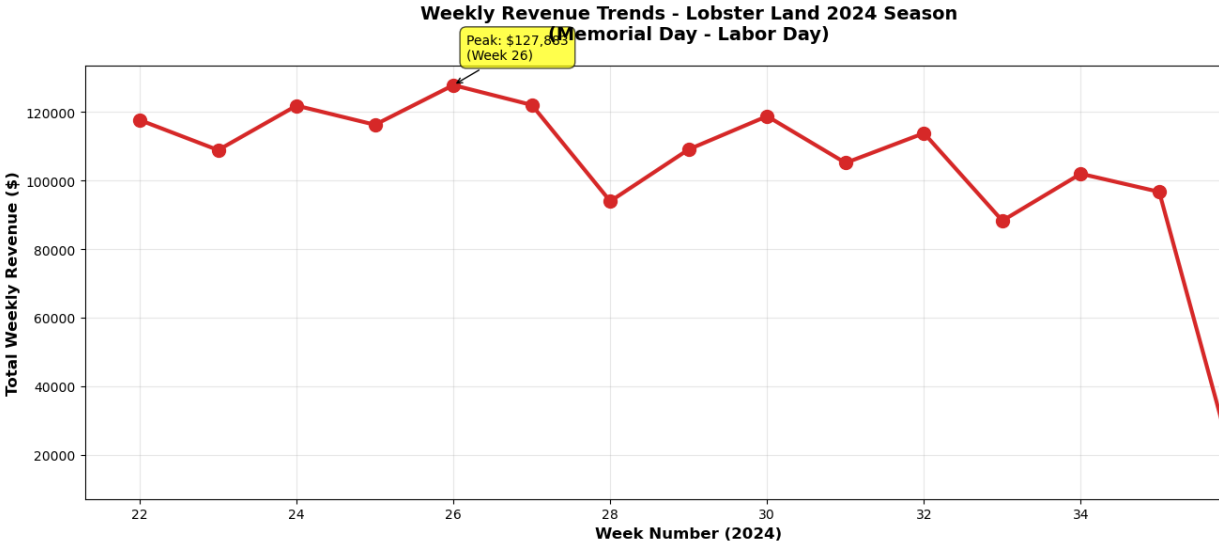
The plot shows that sunny days consistently generate the highest arcade revenue, especially on Mondays and Thursdays, which stand out across multiple weather types. Cloudy days perform reasonably well, while stormy and rainy weekends tend to show the lowest revenue. One interest pattern is that weekday revenue remains relatively strong even in poor weather, suggesting that l passholders may still visit the arcade regardless of conditions.

Important limitation A key limitation is that some day-of-week and weather-type combinations are or have very few observations. For example, we may have many Sunny Fridays but only one Sto Wednesday. This means the averages in the plot are not based on equal sample sizes, so some represent much more reliable estimates than others

R. Lobster Land wishes to dig just a bit deeper into its revenue trends across the 2024 season. Right now, the data is aggregated on a daily basis. Using Python, aggregate the data by week instead of day. Next, create a lineplot that sl 'Week Number' on the x-axis, and total revenue on the y-axis.

In [29]:
```
# Aggregate data by week
df['Week_Number'] = df['Date'].dt.isocalendar().week
weekly_revenue = df.groupby('Week_Number')['Total_Revenue'].sum().reset_i

print("\nWeekly Revenue Totals:")
print(weekly_revenue)

plt.figure(figsize=(14, 6))
plt.plot(weekly_revenue['Week_Number'], weekly_revenue['Total_Revenue'],
         marker='o', linewidth=3, markersize=10, color='#D62828')
plt.xlabel('Week Number (2024)', fontsize=12, fontweight='bold')
plt.ylabel('Total Weekly Revenue ($)', fontsize=12, fontweight='bold')
plt.title('Weekly Revenue Trends - Lobster Land 2024 Season\n(Memorial Da
          fontsize=14, fontweight='bold', pad=20)
plt.grid(True, alpha=0.3)

# Add value labels for peak weeks
max_week = weekly_revenue.loc[weekly_revenue['Total_Revenue'].idxmax()]
plt.annotate(f'Peak: ${max_week["Total_Revenue"]:,.0f}\n(Week {int(max_we
             xy=(max_week['Week_Number'], max_week['Total_Revenue']),
             xytext=(10, 20), textcoords='offset points',
             bbox=dict(boxstyle='round,pad=0.5', facecolor='yellow', alph
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3,rad=0

plt.tight_layout()
plt.show()
```

```
Weekly Revenue Totals:
    Week_Number  Total_Revenue
0            22      117681.76
1            23      108911.66
2            24      121892.56
3            25      116307.36
4            26      127882.85
5            27      122083.45
6            28       93974.40
7            29      109114.93
8            30      118805.11
9            31      105168.45
10           32      113846.59
11           33       88286.22
12           34      102002.18
13           35       96710.79
14           36       12820.86
```



- a. *What general trend does this lineplot show, if any? If you were explaining it to an outside a what would you point out about the graph?*

The weekly revenue graph shows that business is strongest in the middle of the summer, with a peak around Week 26, and consistently high revenue through Week 33. This suggests Lobster Land should staff more heavily and run key promotions during mid-season, when visitor naturally highest. After Week 33, revenue drops sharply, which may reflect schools reopeni end of peak tourism. During these later weeks, Lobster Land could reduce staffing, scale back in or introduce end-of-season deals to maintain engagement while managing costs.

In [30]:
```
df.to_csv("SyrettaAD654_OL_Assignment_I_Summer_2025.ipynb.csv", index=Fal
```

Part III. Using Tableau to Build a Dashboard

S. Bring the **lobsterland_2024.csv** into your Tableau Public environment.

- Dataset description available in Class Discussion >> From Your Professor

T. Using a tiled layout, build a dashboard that includes any four unique visualizat your choice. By "unique" this just means that you should not build two of the sam of plot (e.g. not more than one histogram, not more than one barplot, not more th treemap, etc.). Give a title to each of your four plots.

In [ ]:

U. *Write a one-paragraph description of your dashboard. Write about the plots th made and describe your process.* You can do this in any file format, and upload i your Assignment 1 submission.

To explore visitor behavior and revenue trends at Lobster Land, I built a dashboard using four dis visualizations. I started with a histogram to understand daily attendance patterns. It revealed that days drew between 1,000 and 2,600 visitors, with over half performing abov 80% benchmark. Next, I created a line chart to track monthly revenue, which showed a drama in June, followed by a steady decline through September. Curious about external factors, I added chart comparing revenue across weather conditions, where sunny days stood out with nearly $500,000 in earnings. Finally, I built a table to examine the impact of special events, t that each event type, Fireworks and Arcade Tournaments, attracted over 8,000 total vi across the season, while regular days without events made up nearly 80% of total attendan Throughout the process, I used a tiled layout, added reference lines, and wrote annotations to m insights easy to interpret. This dashboard tells a clear story, Lobster Land's performance is shap seasonality, weather, and event timing. Visualizing these patterns helped me uncover what drive busiest and most profitable days.

V. Paste a link to your file in the same document that you used to write the descr

Note: This section is intentionally very open-ended. Each submission will be unique. The goal he to arrive at a single "correct" answer but to have everyone gain some hands-on experience with l a dashboard in Tableau. The dashboards will not be scored by some 'beauty contest' measure -- here is to (1) make a good-faith effort to build a dashboard with four separate types of visualiza and (2) include a thoughtful narrative paragraph. Every answer that does those things will recei credit for this section.

https://public.tableau.com/views/SyrettaAssignment1_2025/Dashboard1?:language=er US&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link