TIPS DATASET COMPLETE EDA

Importing the necessary libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing the dataset

In [2]:
```
df=pd.read_csv("tips.csv")
df
```

Out[2]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Numb
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	35603251686034
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	44780713797792
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	60118121129713
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	467613764768599
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	48327326186372
...	...	...	...	...	...	...	...	...	...	
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael	52960686060528

									Avila	
240	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	350680615556540
241	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	601189161874719
242	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	437522055095
243	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	351145162669813

244 rows × 11 columns

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

Out[3]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

In [4]:
```
df.shape
```

Out[4]:
```
(244, 11)
```

In [5]:
```
df.size
```

Out[5]:
```
2684
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   total_bill        244 non-null    float64
 1   tip               244 non-null    float64
 2   sex               244 non-null    object 
 3   smoker            244 non-null    object 
 4   day               244 non-null    object 
 5   time              244 non-null    object 
 6   size              244 non-null    int64  
 7   price_per_person  244 non-null    float64
 8   Payer Name        244 non-null    object 
 9   CC Number         244 non-null    int64  
 10  Payment ID        244 non-null    object 
dtypes: float64(3), int64(2), object(6)
memory usage: 21.1+ KB
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
total_bill           0
tip                  0
sex                  0
smoker               0
day                  0
time                 0
size                 0
price_per_person     0
Payer Name           0
CC Number            0
Payment ID           0
dtype: int64
```

In [8]:
```
df.describe()
```

Out[8]:

	total_bill	tip	size	price_per_person	CC Number
count	244.000000	244.000000	244.000000	244.000000	2.440000e+02
mean	19.785943	2.998279	2.569672	7.888197	2.563496e+15
std	8.902412	1.383638	0.951100	2.914234	2.369340e+15
min	3.070000	1.000000	1.000000	2.880000	6.040679e+10
25%	13.347500	2.000000	2.000000	5.800000	3.040731e+13
50%	17.795000	2.900000	2.000000	7.255000	3.525318e+15
75%	24.127500	3.562500	3.000000	9.390000	4.553675e+15
max	50.810000	10.000000	6.000000	20.270000	6.596454e+15

In [9]:
```
df.describe(include='O')
```

Out[9]:

	sex	smoker	day	time	Payer Name	Payment ID
count	244	244	244	244	244	244
unique	2	2	4	2	244	243
top	Male	No	Sat	Dinner	Christy Cunningham	Thur8084
freq	157	151	87	176	1	2

In [10]:
```
df.columns
```

Out[10]:
```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size',
       'price_per_person', 'Payer Name', 'CC Number', 'Payment ID'],
      dtype='object')
```

In [11]:
```
df[df.duplicated()].sum()
```

Out[11]:
```
total_bill           0.0
tip                  0.0
sex                    0
smoker                 0
day                    0
time                   0
size                   0
price_per_person     0.0
Payer Name             0
CC Number              0
Payment ID             0
dtype: object
```

In [12]:
```
cat_col=[col for col in df.columns if df[col].dtype=="O"]
```

```
cat_col
```

Out[12]:
```
['sex', 'smoker', 'day', 'time', 'Payer Name', 'Payment ID']
```

In [13]:
```
num_col=[col for col in df.columns if df[col].dtype!="O"]
num_col
```

Out[13]:
```
['total_bill', 'tip', 'size', 'price_per_person', 'CC Number']
```

In [14]:
```
df[num_col]=df[num_col].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   total_bill        244 non-null    int64 
 1   tip               244 non-null    int64 
 2   sex               244 non-null    object
 3   smoker            244 non-null    object
 4   day               244 non-null    object
 5   time              244 non-null    object
 6   size              244 non-null    int64 
 7   price_per_person  244 non-null    int64 
 8   Payer Name        244 non-null    object
 9   CC Number         244 non-null    int64 
 10  Payment ID        244 non-null    object
dtypes: int64(5), object(6)
memory usage: 21.1+ KB
```

Observations :

- There are no NAN values present in the dataset.
- No duplicate values in the dataset.
-

Univariate analysis

Numerical Variables

1.Total Bill

In [15]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Univariate Analysis for Numerical Variables
```

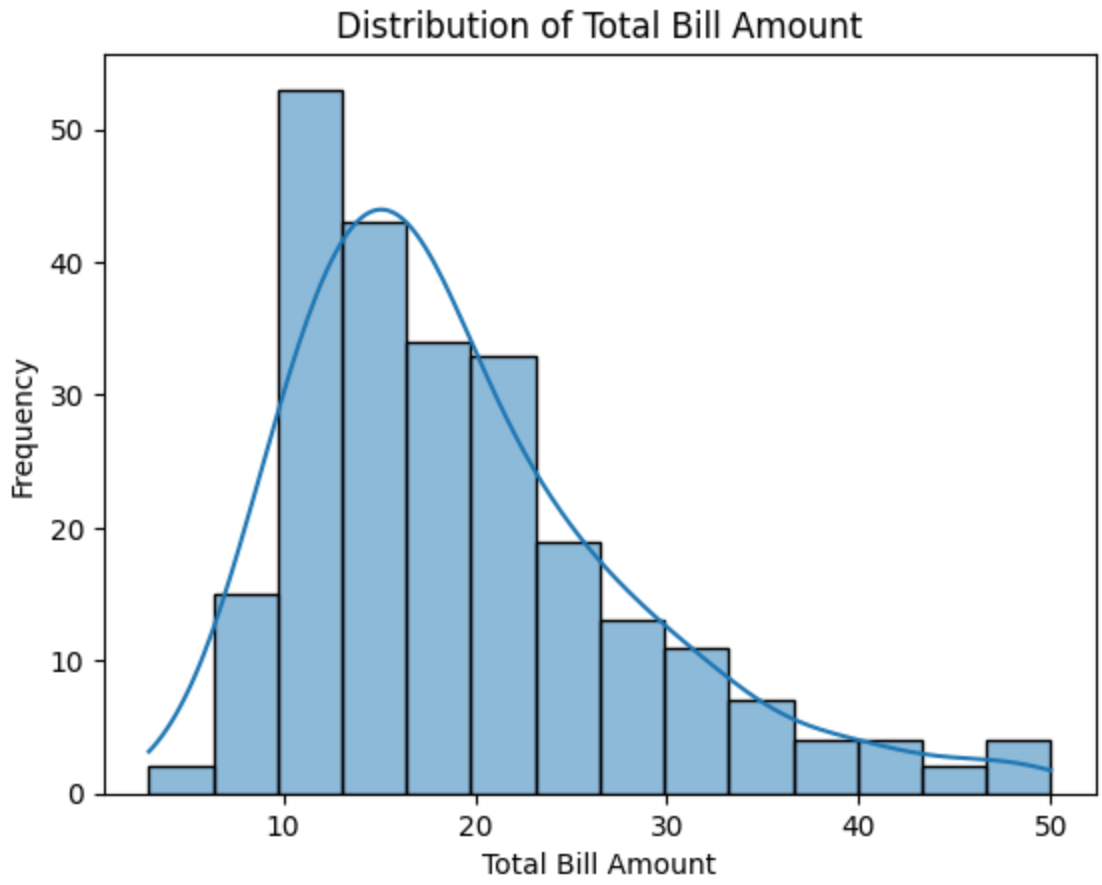
In [16]:
```
df['total_bill'].agg((max,min))
```

Out[16]:
```
max    50
min     3
Name: total_bill, dtype: int64
```

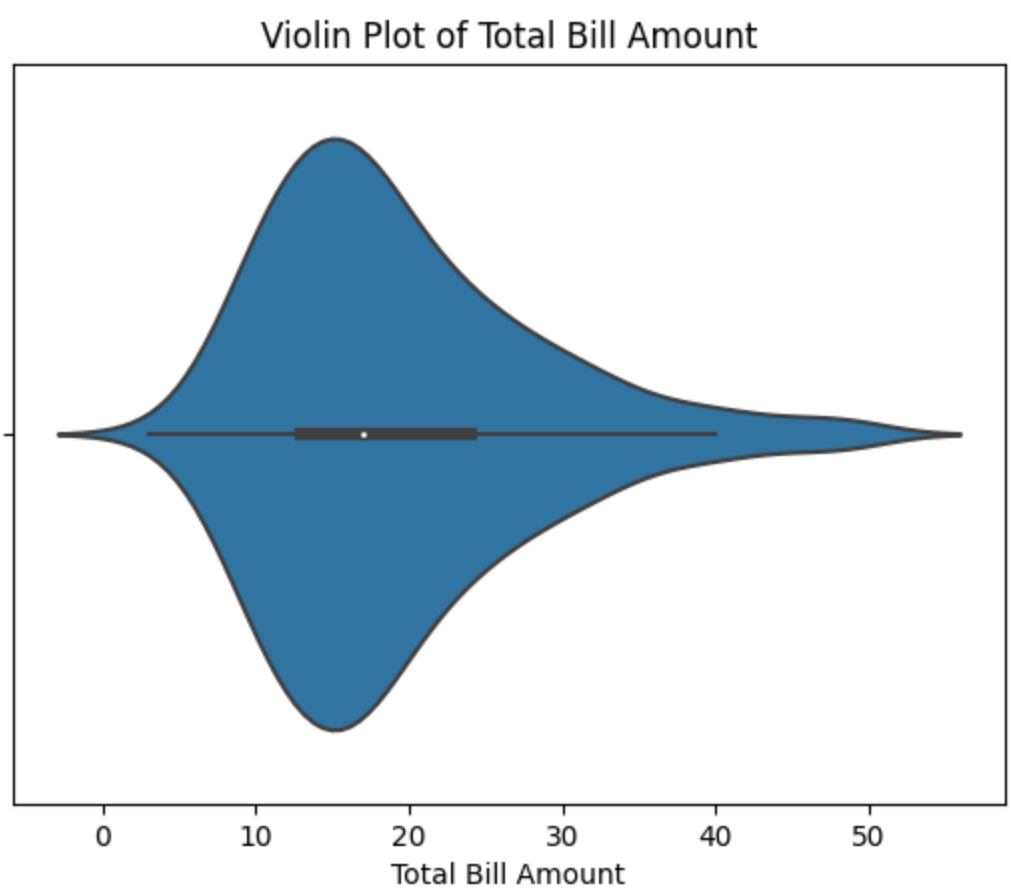
In [17]:
```
# Total Bill

sns.histplot(df['total_bill'],kde=True)

plt.title('Distribution of Total Bill Amount')
```

```
plt.xlabel('Total Bill Amount')
plt.ylabel('Frequency')
```

Out[17]:
```
Text(0, 0.5, 'Frequency')
```



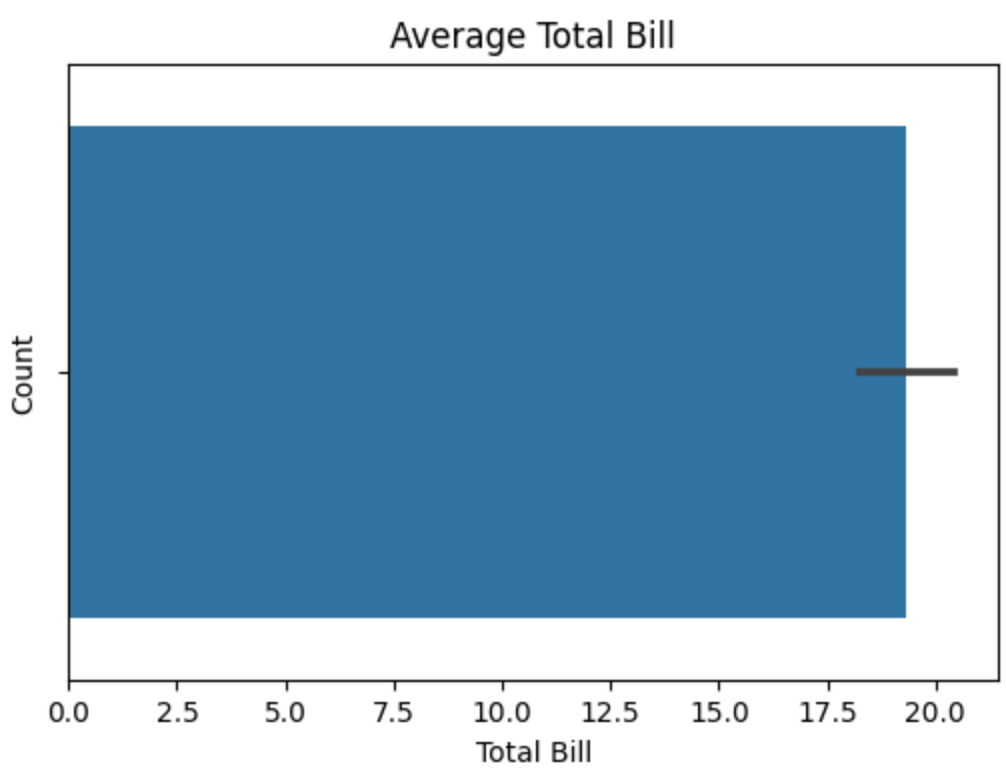
In [18]:
```
sns.violinplot(data=df, x='total_bill')
plt.title('Violin Plot of Total Bill Amount')
plt.xlabel('Total Bill Amount')
plt.show()
```



In [19]:
```
# Assuming your dataset is stored in a DataFrame called 'df'
average_total_bill = df['total_bill'].mean()
print(f"The average total bill is: ${average_total_bill:.2f}")
```

```
The average total bill is: $19.32
```

In [20]:
```
# Create a bar plot to visualize the average total bill
plt.figure(figsize=(6, 4))
sns.barplot(data=df, x='total_bill')
plt.title('Average Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Count')
plt.show()
```



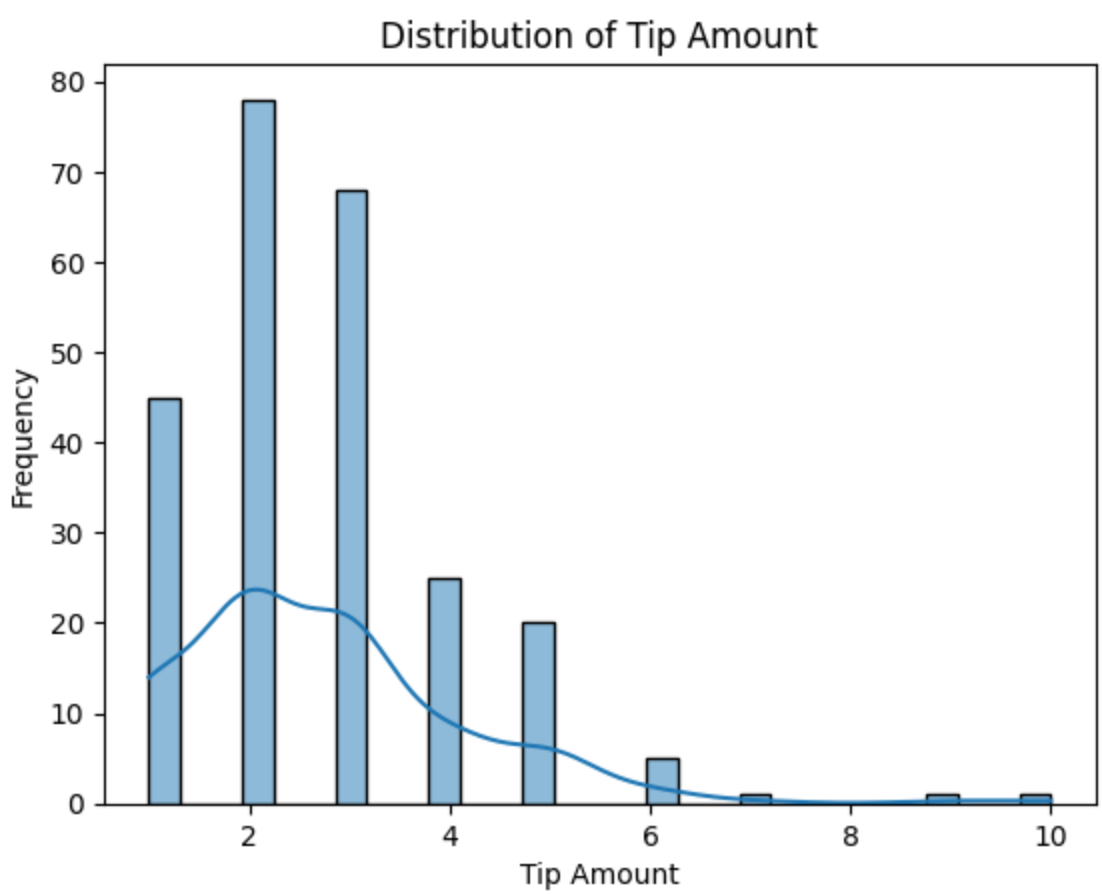
2.Tip

In [21]:
```
df['tip'].agg((max,min))
```

Out[21]:
```
max    10
min     1
Name: tip, dtype: int64
```

In [22]:
```
# Tip

sns.histplot(df['tip'], kde=True)
plt.title('Distribution of Tip Amount')
plt.xlabel('Tip Amount')
plt.ylabel('Frequency')
```

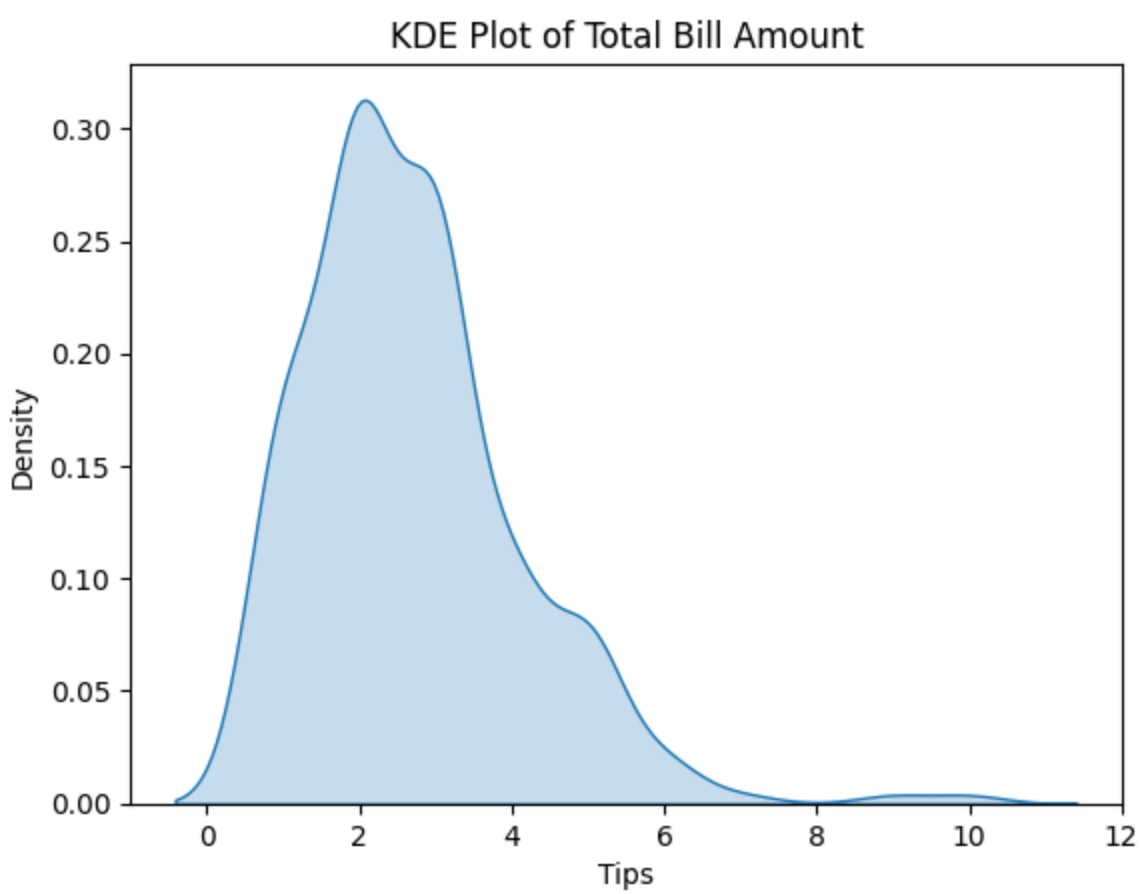
Out[22]:
```
Text(0, 0.5, 'Frequency')
```



In [23]:

```
# Tip

sns.kdeplot(data=df, x='tip', shade=True)
plt.title('KDE Plot of Total Bill Amount')
plt.xlabel('Tips')
plt.ylabel('Density')
plt.show()
```

```
/var/folders/rs/w1cppzrd0tg10lz07q7q90qm0000gn/T/ipykernel_2255/2797215799.py:3: FutureW
arning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data=df, x='tip', shade=True)
```



3.Size

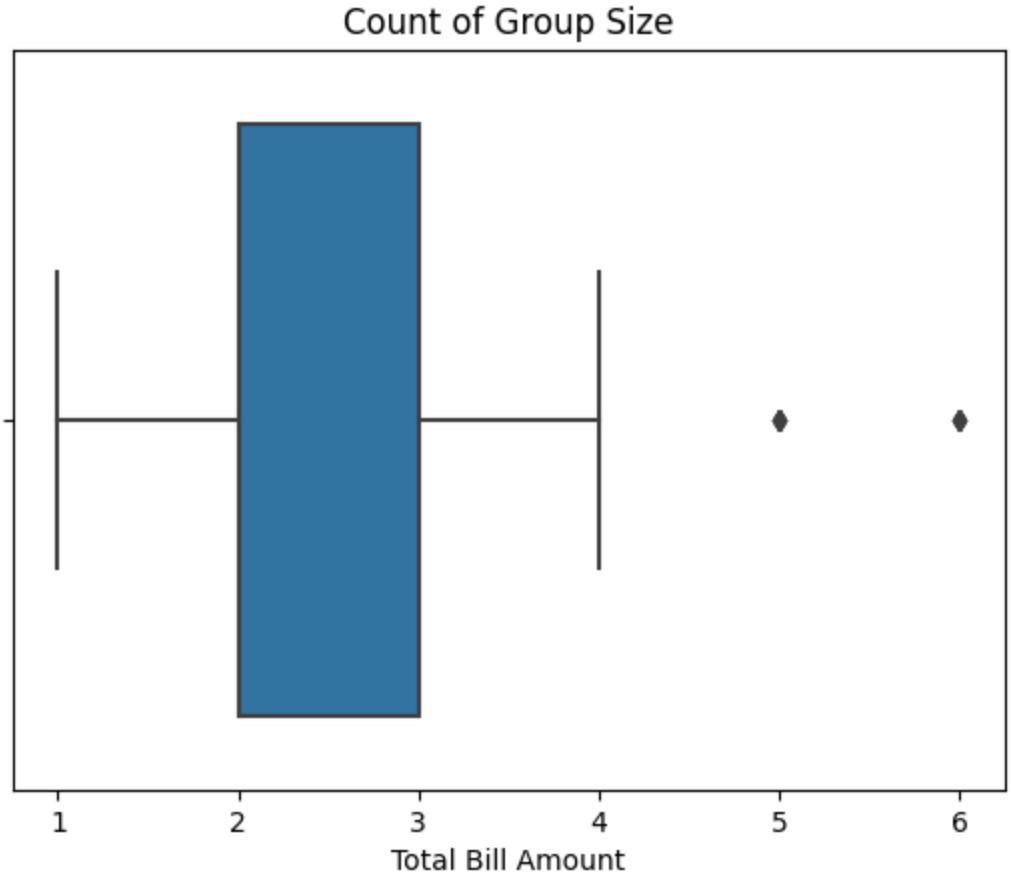
In [24]:
```
df['size'].value_counts()
```

Out[24]:
```
size
2    156
3     38
4     37
5      5
1      4
6      4
Name: count, dtype: int64
```

In [25]:
```
df['size'].agg((max,min))
```

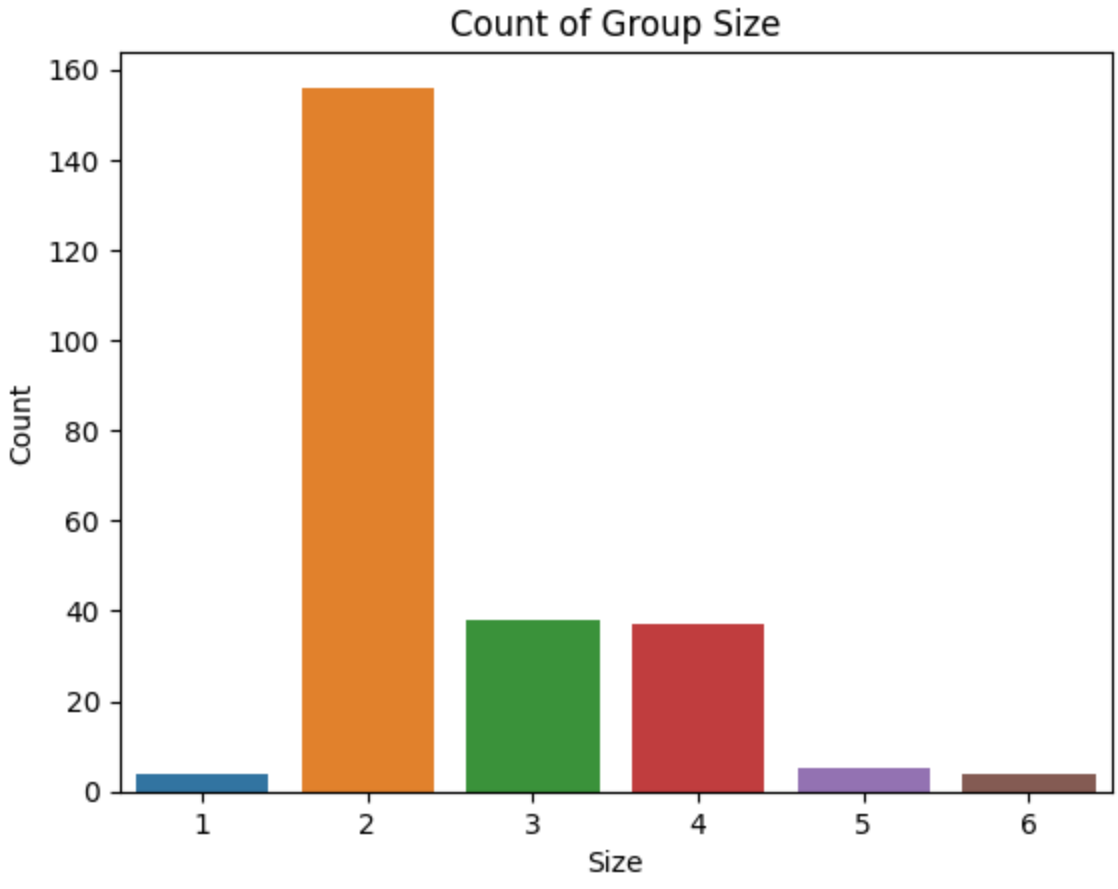
Out[25]:
```
max    6
min    1
Name: size, dtype: int64
```

In [26]:
```
sns.boxplot(data=df, x='size')
plt.title('Count of Group Size')
plt.xlabel('Total Bill Amount')
plt.show()
```



In [27]:
```
sns.countplot(data=df, x='size')
plt.title('Count of Group Size')
plt.xlabel('Size')
plt.ylabel('Count')
```

Out[27]:
```
Text(0, 0.5, 'Count')
```



4. price_per_person

In [28]:
```
df['price_per_person'].value_counts(ascending=True)
```

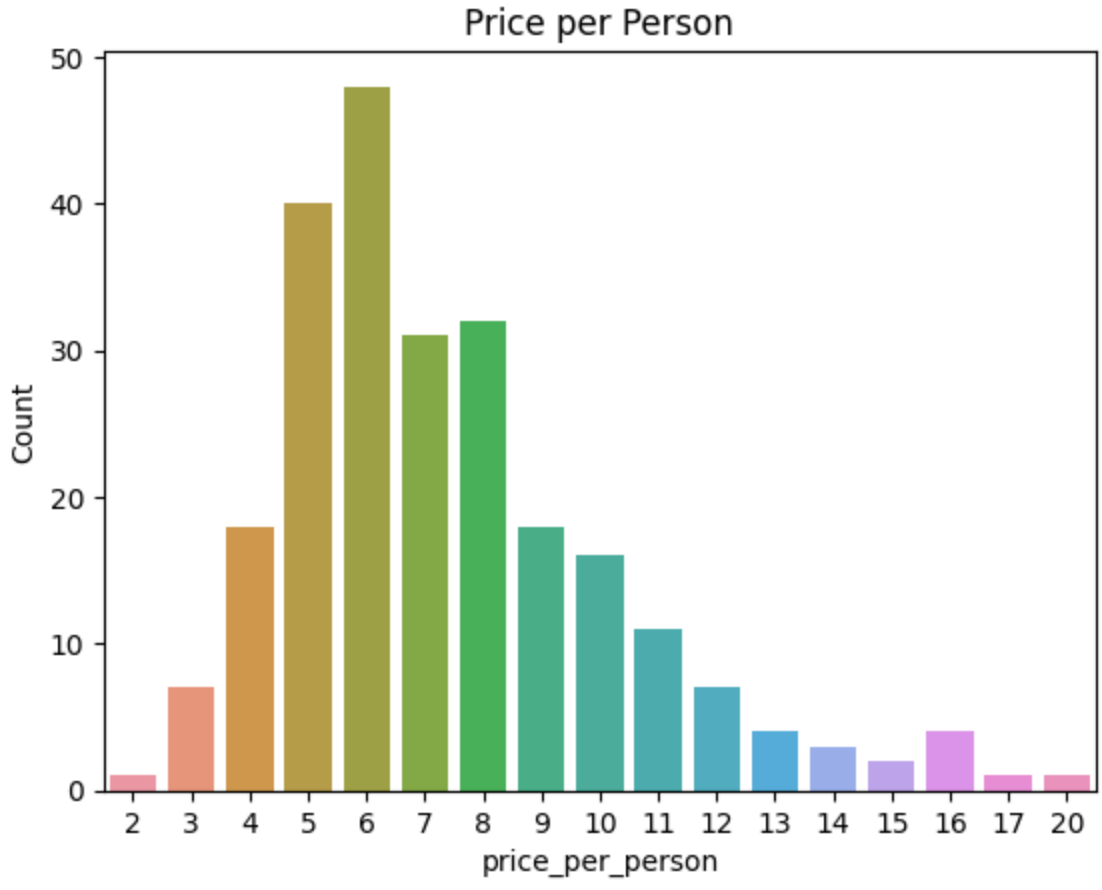
Out[28]:
```
price_per_person
20     1
2      1
17     1
15     2
14     3
16     4
13     4
3      7
12     7
11    11
10    16
9     18
4     18
7     31
8     32
5     40
6     48
Name: count, dtype: int64
```

In [29]:
```
df['price_per_person'].agg((max,min))
```

Out[29]:
```
max    20
min     2
Name: price_per_person, dtype: int64
```

In [30]:
```
sns.countplot(data=df, x='price_per_person')
plt.title('Price per Person')
plt.xlabel('price_per_person')
plt.ylabel('Count')
```

Out[30]:
```
Text(0, 0.5, 'Count')
```



Observations :

> . Max total_bill is 50 and minimum total_bill is 3.
>
> . The maximum tip given is 10.
>
> . From size column we found that mostly two people or couple comes to the restaurant.

Categorical Variable

1. Sex

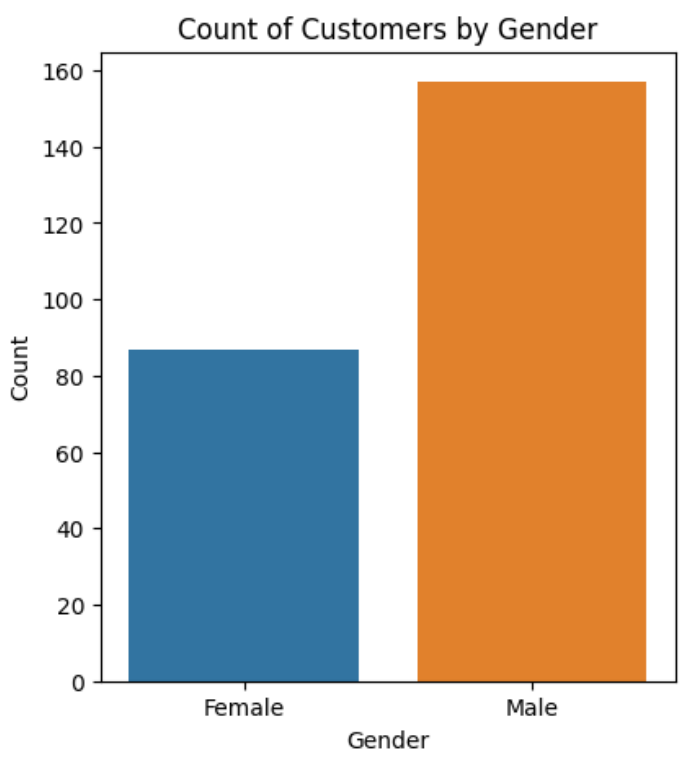
In [31]:
```
df['sex'].head()
```

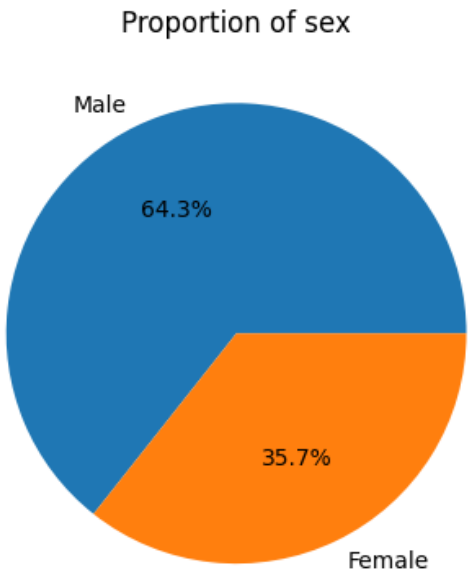
Out[31]:
```
0    Female
1      Male
2      Male
3      Male
4    Female
Name: sex, dtype: object
```

In [32]:
```
df['sex'].value_counts()
```

Out[32]:
```
sex
Male      157
Female     87
Name: count, dtype: int64
```

In [33]:
```
# Sex
plt.figure(figsize=(10, 5))

plt.subplot(121)
sns.countplot(data=df, x='sex')
plt.title('Count of Customers by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')

plt.subplot(122)
category_counts = df['sex'].value_counts()
plt.pie(category_counts, labels=category_counts.index, autopct='%1.1f%%')
plt.title('Proportion of sex')
plt.show()
```



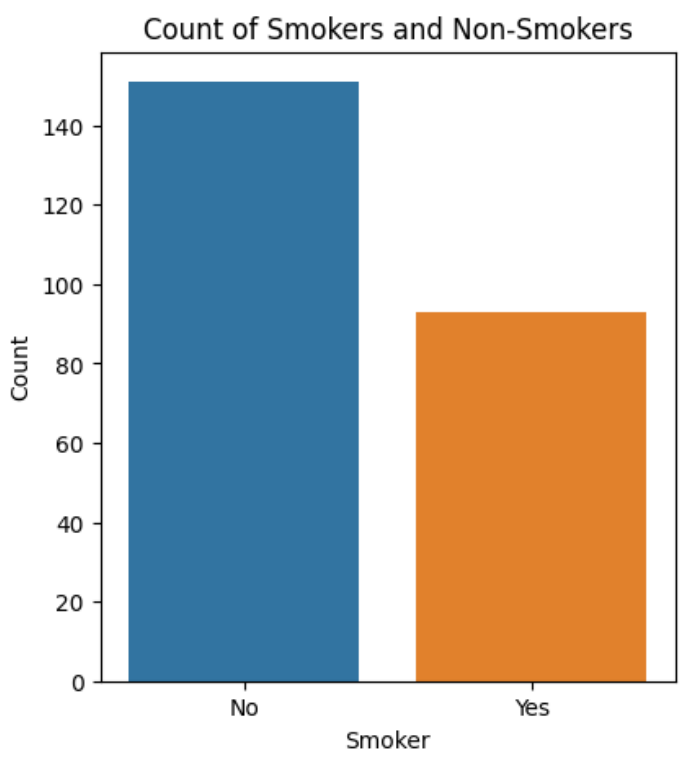


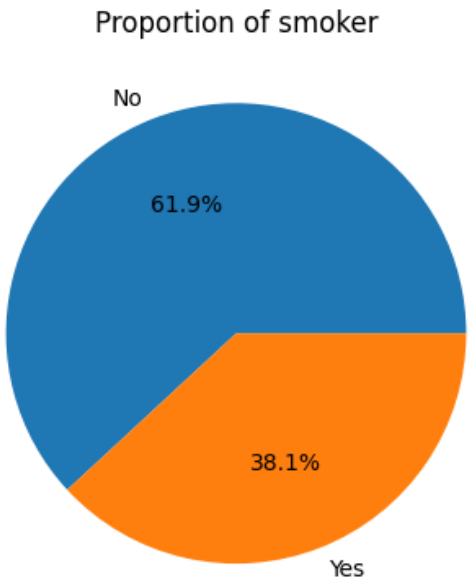
2. Smoker

In [34]:
```
df['smoker'].value_counts()
```

Out[34]:
```
smoker
No     151
Yes     93
Name: count, dtype: int64
```

In [35]:
```
plt.figure(figsize=(10, 5))

plt.subplot(121)
sns.countplot(data=df, x='smoker')
plt.title('Count of Smokers and Non-Smokers')
plt.xlabel('Smoker')
plt.ylabel('Count')

plt.subplot(122)
cat_counts = df['smoker'].value_counts()
plt.pie(cat_counts, labels=cat_counts.index, autopct='%1.1f%%')
plt.title('Proportion of smoker')
plt.show()
```





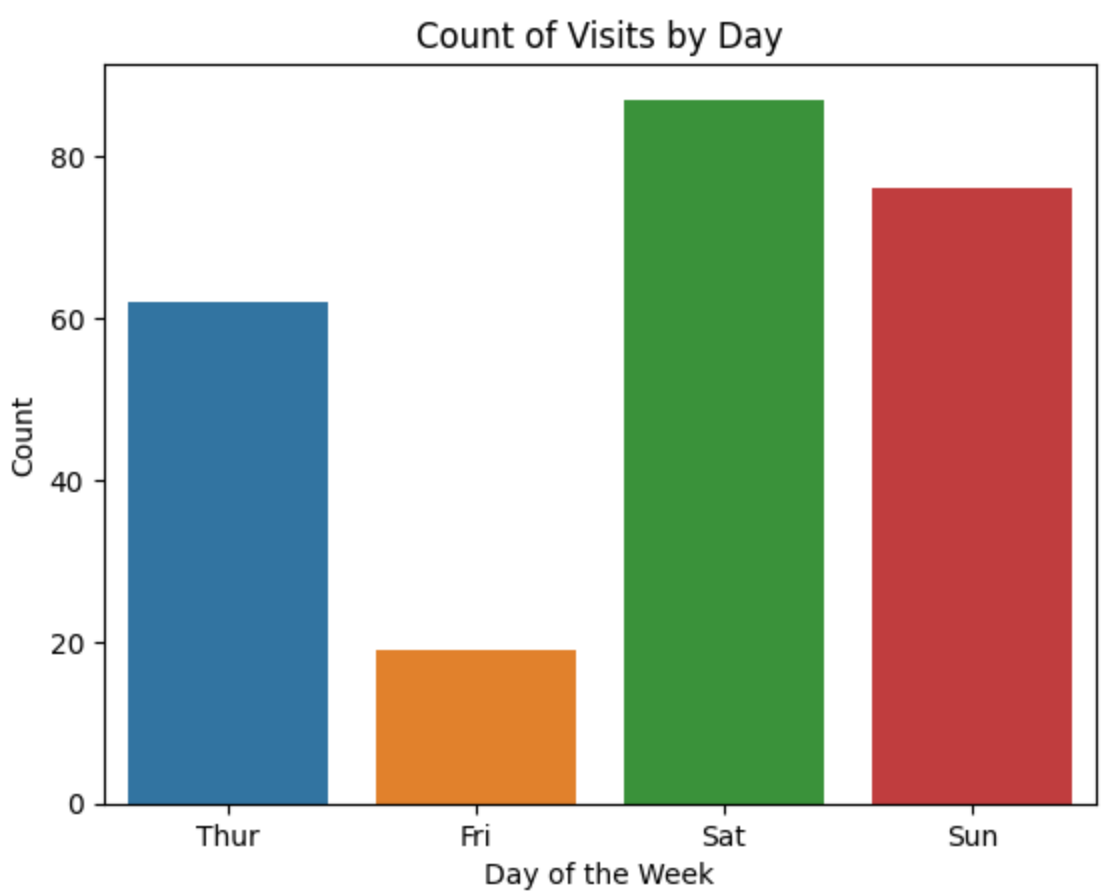
3. Day

In [36]:
```
df['day'].value_counts()
```

Out[36]:
```
day
Sat     87
Sun     76
Thur    62
Fri     19
Name: count, dtype: int64
```

In [37]:
```
sns.countplot(data=df, x='day', order=['Thur', 'Fri', 'Sat', 'Sun'])
plt.title('Count of Visits by Day')
plt.xlabel('Day of the Week')
plt.ylabel('Count')
```

Out[37]:
```
Text(0, 0.5, 'Count')
```

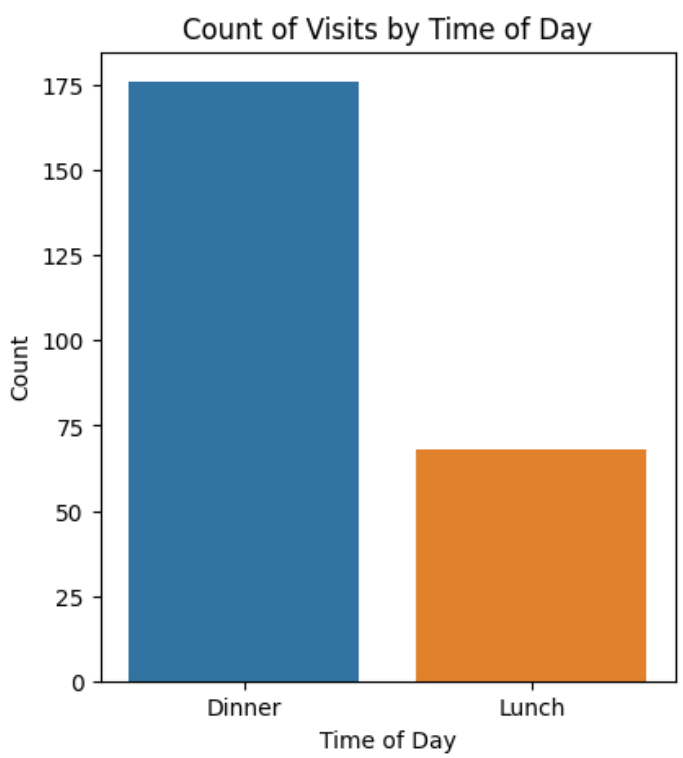


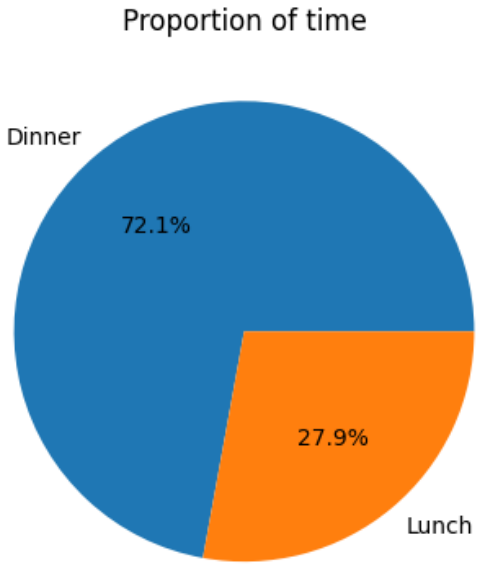
4. Time

In [38]:
```
df['time'].value_counts()
```

Out[38]:
```
time
Dinner    176
Lunch      68
Name: count, dtype: int64
```

In [39]:
```
# Time
plt.figure(figsize=(10, 5))
plt.subplot(121)
sns.countplot(data=df, x='time')
plt.title('Count of Visits by Time of Day')
plt.xlabel('Time of Day')
plt.ylabel('Count')


plt.subplot(122)
time_counts = df['time'].value_counts()
plt.pie(time_counts, labels=time_counts.index, autopct='%1.1f%%')
plt.title('Proportion of time')
plt.show()
```





Observations :

> .Mostly men comes to the restaurant and out of 100% of the population approximately 64% are men.
>
> .Out of total population 62% are smoker.
>
> .Maximum pepole prefer to come on weekends.
>
> .Most of the people come for dinner.

Bivariate Analysis

'

1. Let's analyze the relationship between the 'total_bill' and 'tip' amounts.

You can analyze the relationship between the total bill and the tip amount. Are there any tipping patterns or cultural factors influencing tips?

In [40]:
```
df['tip'].value_counts()
```

Out[40]:
```
tip
2     78
3     68
1     45
4     25
5     20
6      5
7      1
10     1
```

```
9      1
Name: count, dtype: int64
```

In [41]:
```
df['tip'].agg((max,min))
```

Out[41]:
```
max    10
min     1
Name: tip, dtype: int64
```

In [42]:
```
df['tip'].value_counts().agg((max,min))
```

Out[42]:
```
max    78
min     1
Name: count, dtype: int64
```

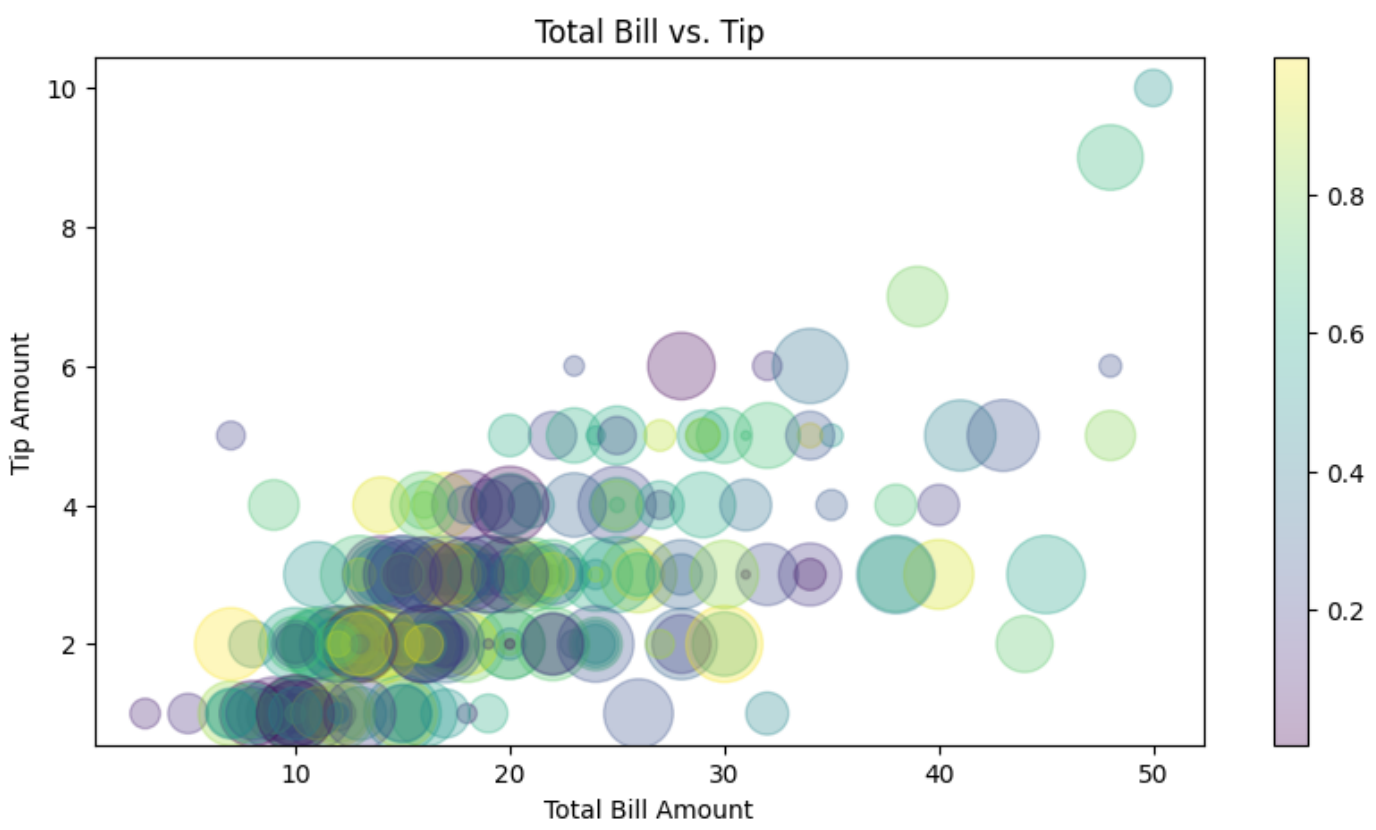
In [43]:
```
max_tip_count = df['tip'].value_counts().max()
tips_with_max_count = df['tip'].value_counts().idxmax()

print(f"The maximum tip count is {max_tip_count} for a tip value of ${tips_with_max_coun
```

```
The maximum tip count is 78 for a tip value of $2.00
```

In [44]:
```
min_tip_count = df['tip'].value_counts(ascending=True).min()
tips_with_min_count = df['tip'].value_counts().idxmin()

print(f"The minimum tip count is {min_tip_count} for a tip value of ${tips_with_min_coun
```

```
The minimum tip count is 1 for a tip value of $7.00
```

In [45]:
```
rng = np.random.RandomState(0)
colors = rng.rand(len(df))
sizes = 1000 * rng.rand(len(df))

plt.figure(figsize = (10, 5))
x = df['total_bill']
y = df['tip']
plt.scatter(x, y, c=colors[:], s=sizes, alpha=0.3,
                    cmap='viridis')

plt.title('Total Bill vs. Tip')
plt.xlabel('Total Bill Amount')
plt.ylabel('Tip Amount')

plt.colorbar();
```

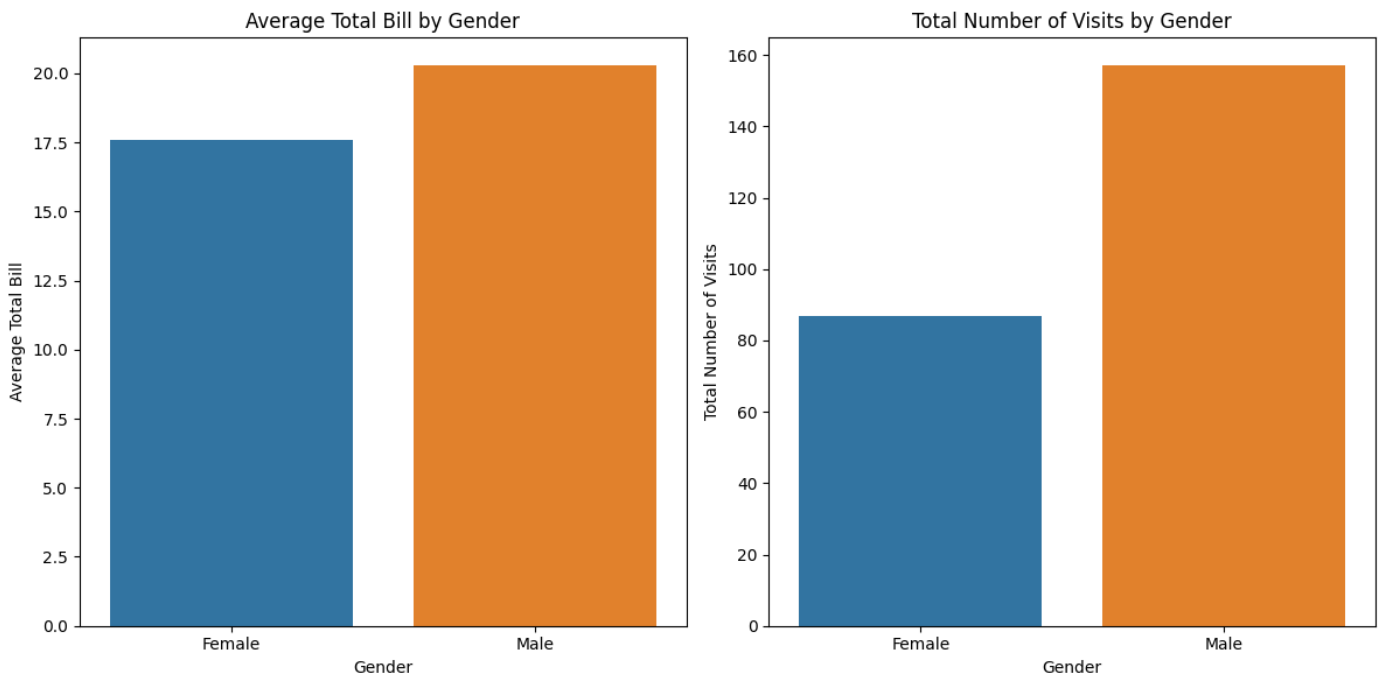


2. Gender spending paterns

By analyzing total bills based on gender, you can see if there are any differences in spending between males and females. Are there any noticeable patterns or variations?

In [46]:
```
# Assuming your dataset is stored in a DataFrame named 'df'
gender_spending_stats = df.groupby('sex').agg({
    'total_bill': ['mean', 'sum'],
    'tip': ['mean', 'sum'],
    'size': 'count'
}).reset_index()
```

In [47]:
```
# Create visualizations
plt.figure(figsize=(12, 6))

# Barplot for Average Total Bill
plt.subplot(1, 2, 1)
sns.barplot(data=gender_spending_stats, x='sex', y=('total_bill', 'mean'))
plt.title('Average Total Bill by Gender')
plt.xlabel('Gender')
plt.ylabel('Average Total Bill')

# Barplot for Total Number of Visits
plt.subplot(1, 2, 2)
sns.barplot(data=gender_spending_stats, x='sex', y=('size', 'count'))
plt.title('Total Number of Visits by Gender')
plt.xlabel('Gender')
plt.ylabel('Total Number of Visits')

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



3. relationship between the categorical variables sex and smoker

In [48]:
```
#totl number of men
df[df['sex']=='Male']['sex'].count()
```

Out[48]:
```
157
```

In [49]:
```
#totl number of female
df[df['sex']=='Female']['sex'].count()
```

Out[49]:
```
87
```

In [50]:
```
# number of women who smoke
df[(df['sex']=='Female') & (df['smoker']=='Yes')]['smoker'].count()
```

Out[50]:
```
33
```

In [51]:
```
#number of men who smoke
df[(df['sex']=='Male') & (df['smoker']=='Yes')]['smoker'].count()
```

Out[51]:
```
60
```

In [52]:
```
(60/157)*100 ## means 38% of men are smoker
```

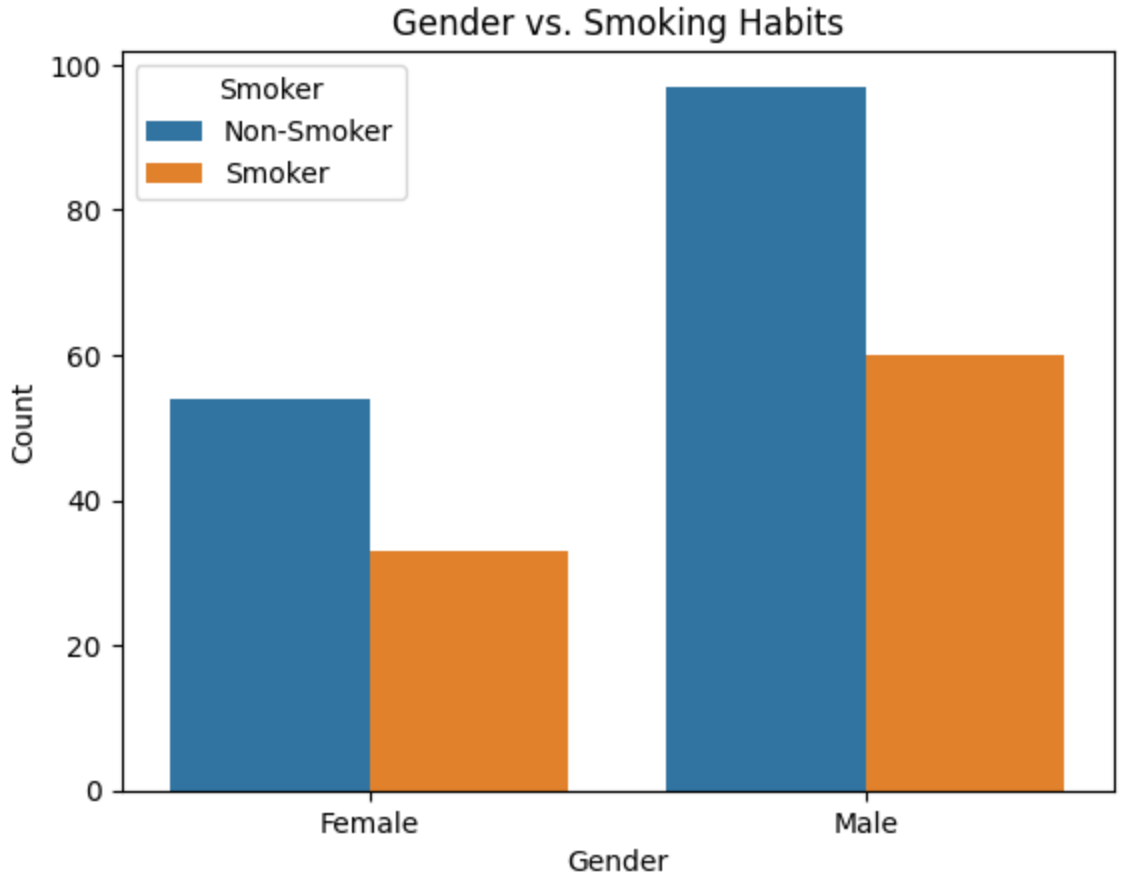
Out[52]:
```
38.21656050955414
```

In [53]:
```
(33/87)*100 ##means 38% women are smoker
```

Out[53]:
```
37.93103448275862
```

In [54]:
```
# Create a count plot to compare gender and smoking habits
sns.countplot(data=df, x='sex', hue='smoker')

# Add labels and a title
plt.title('Gender vs. Smoking Habits')
plt.xlabel('Gender')
```

```
plt.ylabel('Count')

# Add a legend to distinguish between smokers and non-smokers
plt.legend(title='Smoker', labels=['Non-Smoker', 'Smoker'])

# Show the plot
plt.show()
```



4. SALES PER DAY

In [55]:
```
df.groupby('day')['total_bill'].sum().reset_index()
```

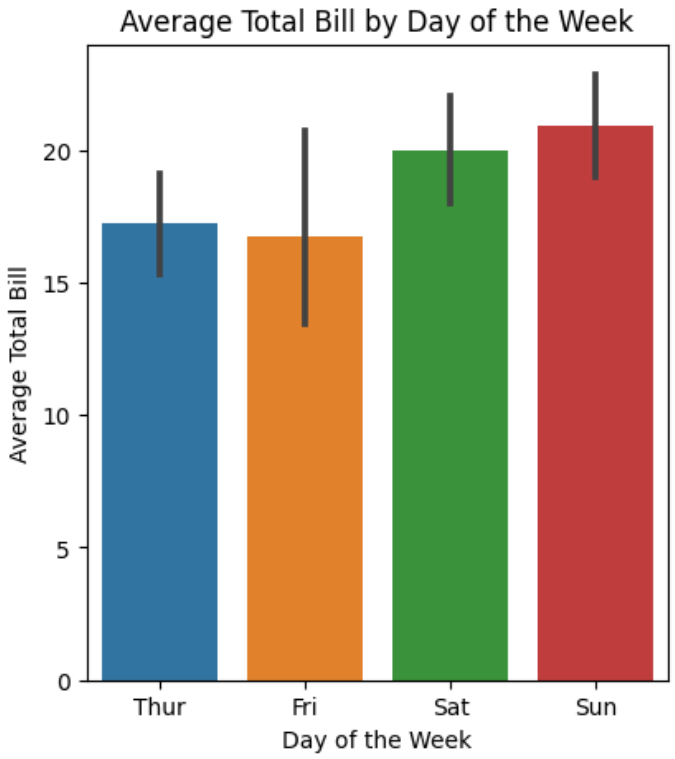
Out[55]:

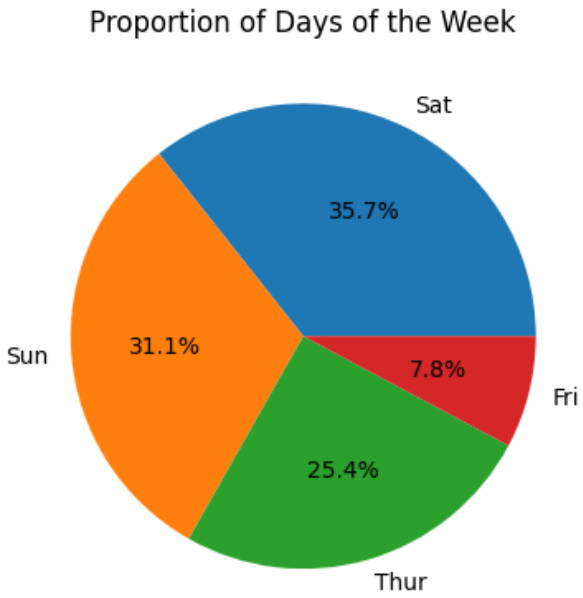
	day	total_bill
0	Fri	318
1	Sat	1739
2	Sun	1588
3	Thur	1069

In [56]:
```
# Group the data by day and sum the total bills to find the sales for each day
sales_by_day = df.groupby('day')['total_bill'].sum().reset_index()

# Find the day with the highest sales
highest_sales_day = sales_by_day[sales_by_day['total_bill'] == sales_by_day['total_bill'

print("Day with the highest sales:")
print(highest_sales_day)
```

```
Day with the highest sales:
   day  total_bill
1  Sat        1739
```

In [57]:
```
plt.figure(figsize=(10, 5))

plt.subplot(121)
sns.barplot(data=df, x='day', y='total_bill', order=['Thur', 'Fri', 'Sat', 'Sun'])
plt.title('Average Total Bill by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Average Total Bill')




plt.subplot(122)
category_counts = df['day'].value_counts()
plt.pie(category_counts, labels=category_counts.index, autopct='%1.1f%%')
plt.title('Proportion of Days of the Week')
plt.show()
```





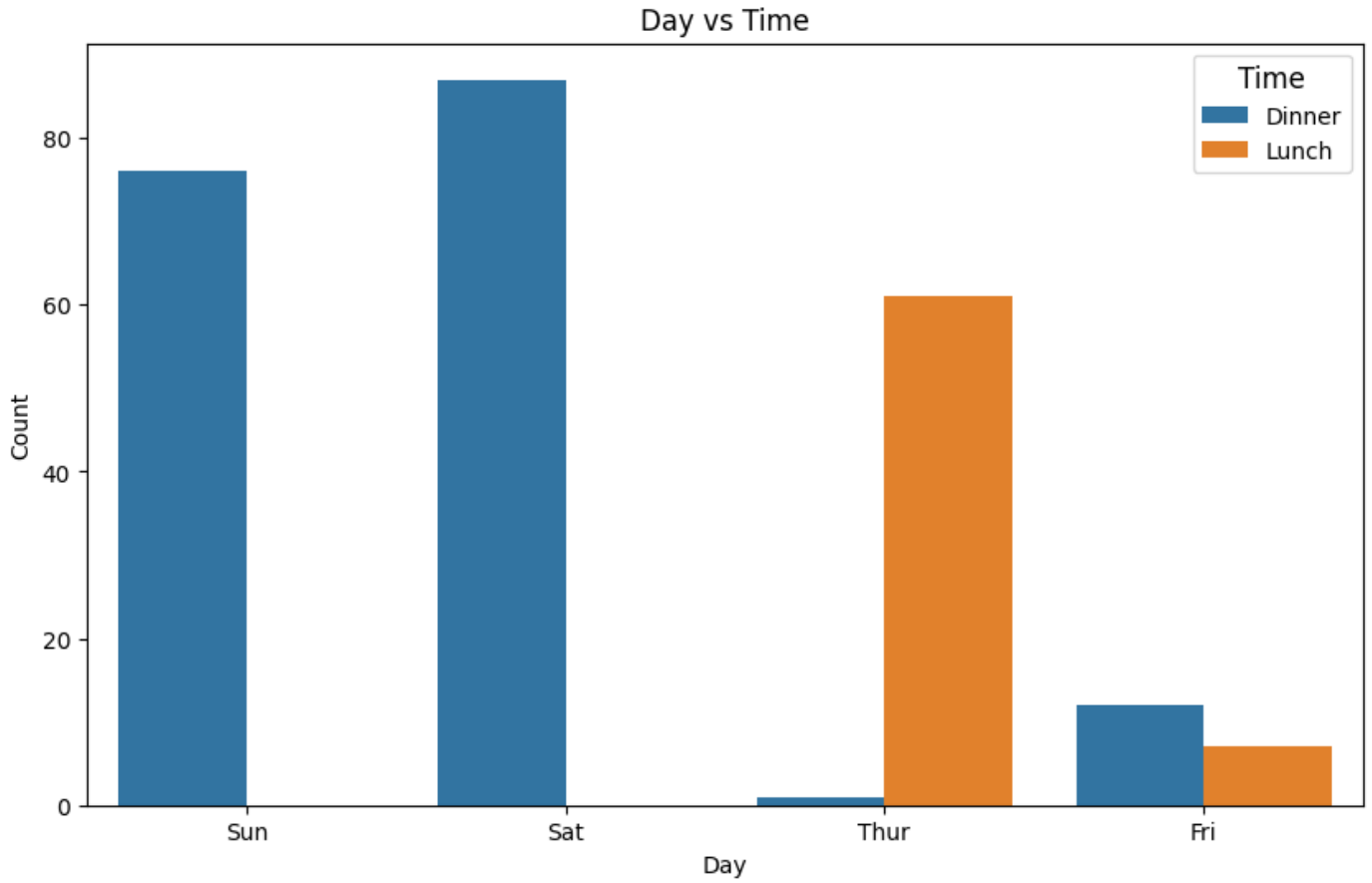
5.meal times by day of the week

In [58]:
```
df.groupby(['day', 'time']).size()
```

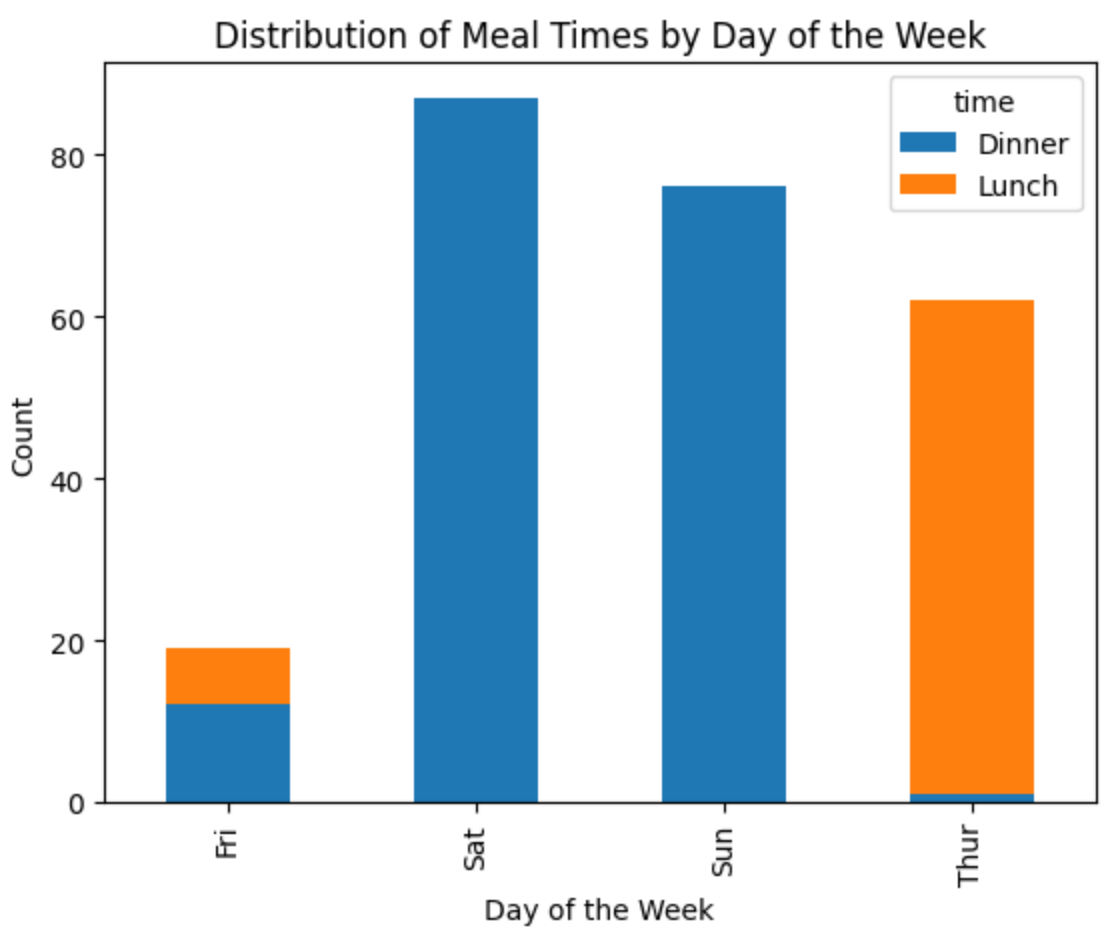
Out[58]:
```
day   time
Fri   Dinner    12
      Lunch      7
Sat   Dinner    87
Sun   Dinner    76
Thur  Dinner     1
      Lunch     61
dtype: int64
```

In [59]:
```
# Categorical-Categorical Bivariate Analysis: Day vs. Time

plt.figure(figsize=(10, 6))


sns.countplot(data=df, x='day', hue='time')
plt.title('Day vs Time')
plt.xlabel('Day')
plt.ylabel('Count')
```

```
plt.legend(title='Time', title_fontsize='12')
plt.show()
```



In [60]:
```
day_time_counts = df.groupby(['day', 'time']).size().unstack(fill_value=0)
day_time_counts.plot(kind='bar', stacked=True)
plt.title('Distribution of Meal Times by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Count')
plt.show()
```

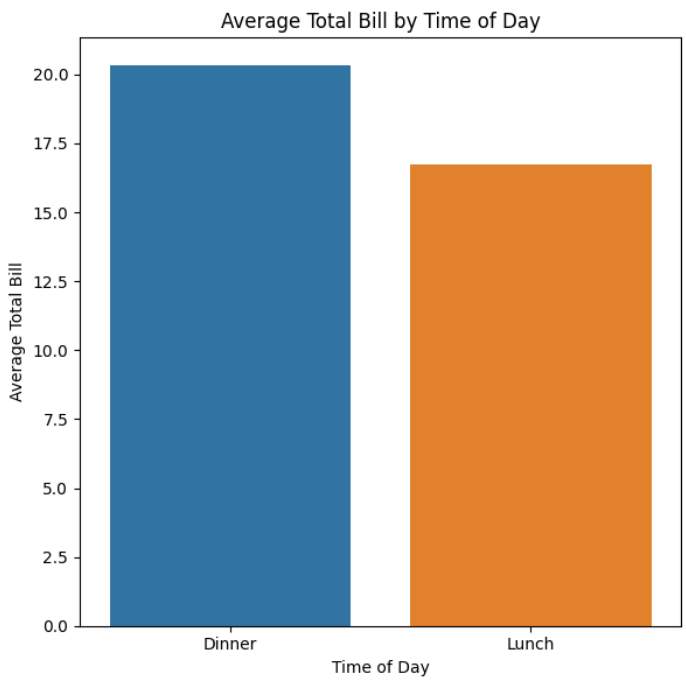


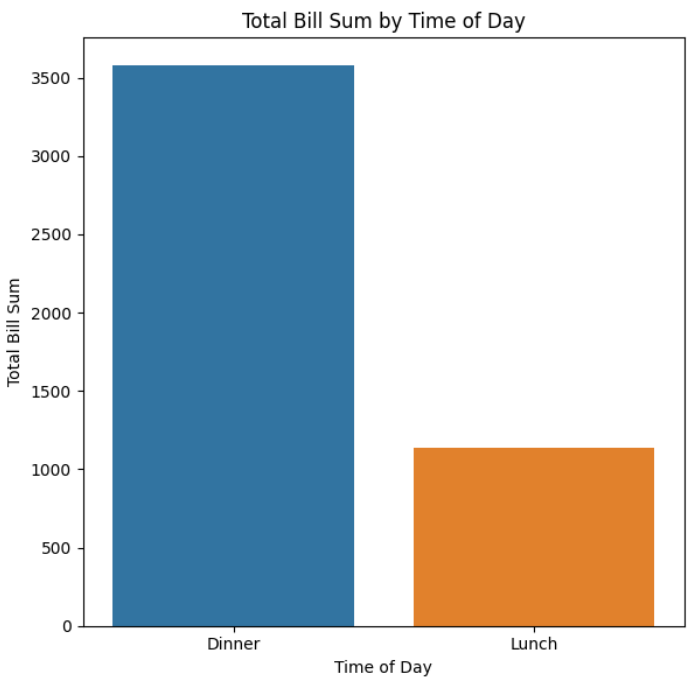
6. Peak Hours:

Determine the busiest hours of the day or the most popular day of the week. This can help with staffing and resource allocation.

In [61]:
```
# Assuming your dataset is stored in a DataFrame named 'df'
peak_hour_stats = df.groupby('time').agg({
    'total_bill': ['mean', 'sum'],
    'tip': ['mean', 'sum'],
    'size': 'count'
}).reset_index()
```

In [62]:
```
# Create visualizations
plt.figure(figsize=(12, 6))

# Barplot for Average Total Bill by Time of Day
plt.subplot(1, 2, 1)
sns.barplot(data=peak_hour_stats, x='time', y=('total_bill', 'mean'))
plt.title('Average Total Bill by Time of Day')
plt.xlabel('Time of Day')
plt.ylabel('Average Total Bill')

# Barplot for Total Bill Sum by Time of Day
plt.subplot(1, 2, 2)
sns.barplot(data=peak_hour_stats, x='time', y=('total_bill', 'sum'))
plt.title('Total Bill Sum by Time of Day')
plt.xlabel('Time of Day')
plt.ylabel('Total Bill Sum')

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



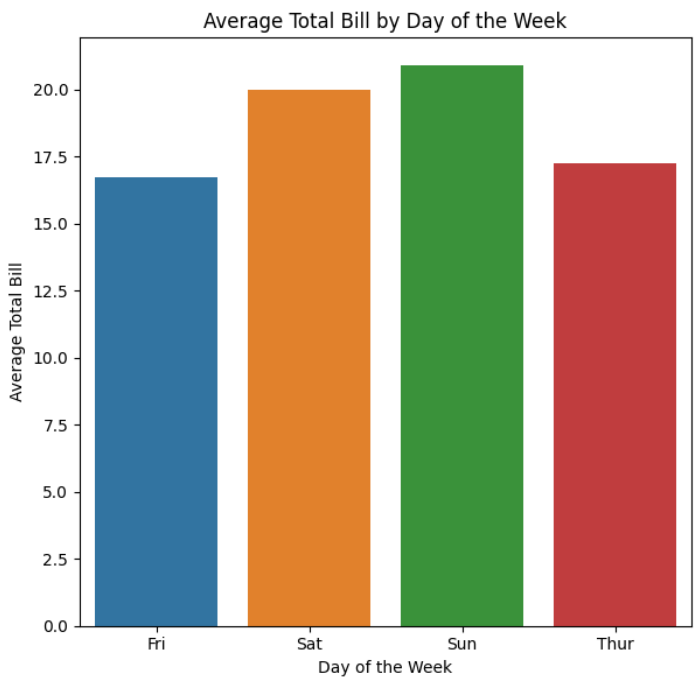


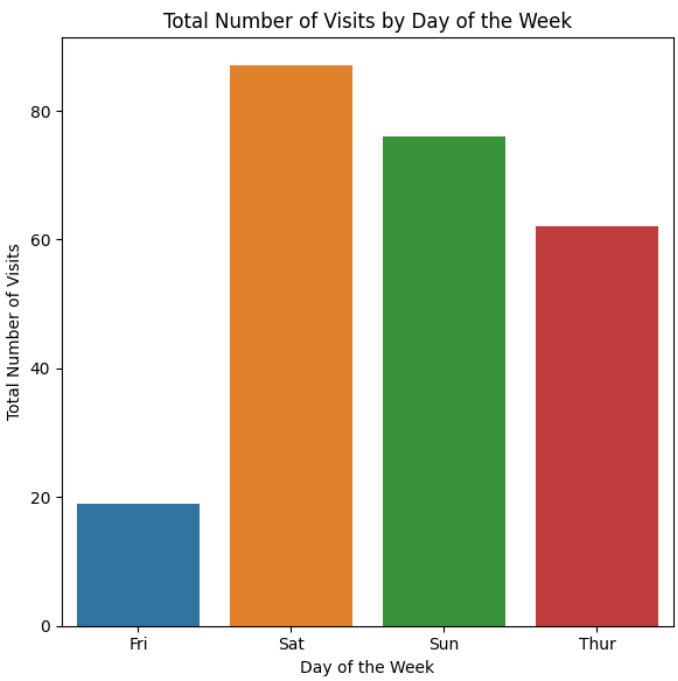
7. Day-of-the-Week Patterns

Are there specific days of the week when the restaurant has higher or lower sales? You can look for trends related to days of the week.

In [63]:
```
# Assuming your dataset is stored in a DataFrame named 'df'
day_of_week_stats = df.groupby('day').agg({
    'total_bill': ['mean', 'sum'],
    'tip': ['mean', 'sum'],
    'size': 'count'
}).reset_index()
```

In [64]:
```
# Create visualizations
plt.figure(figsize=(12, 6))

# Barplot for Average Total Bill
plt.subplot(1, 2, 1)
sns.barplot(data=day_of_week_stats, x='day', y=('total_bill', 'mean'))
plt.title('Average Total Bill by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Average Total Bill')

# Barplot for Total Number of Visits
plt.subplot(1, 2, 2)
sns.barplot(data=day_of_week_stats, x='day', y=('size', 'count'))
plt.title('Total Number of Visits by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Total Number of Visits')

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



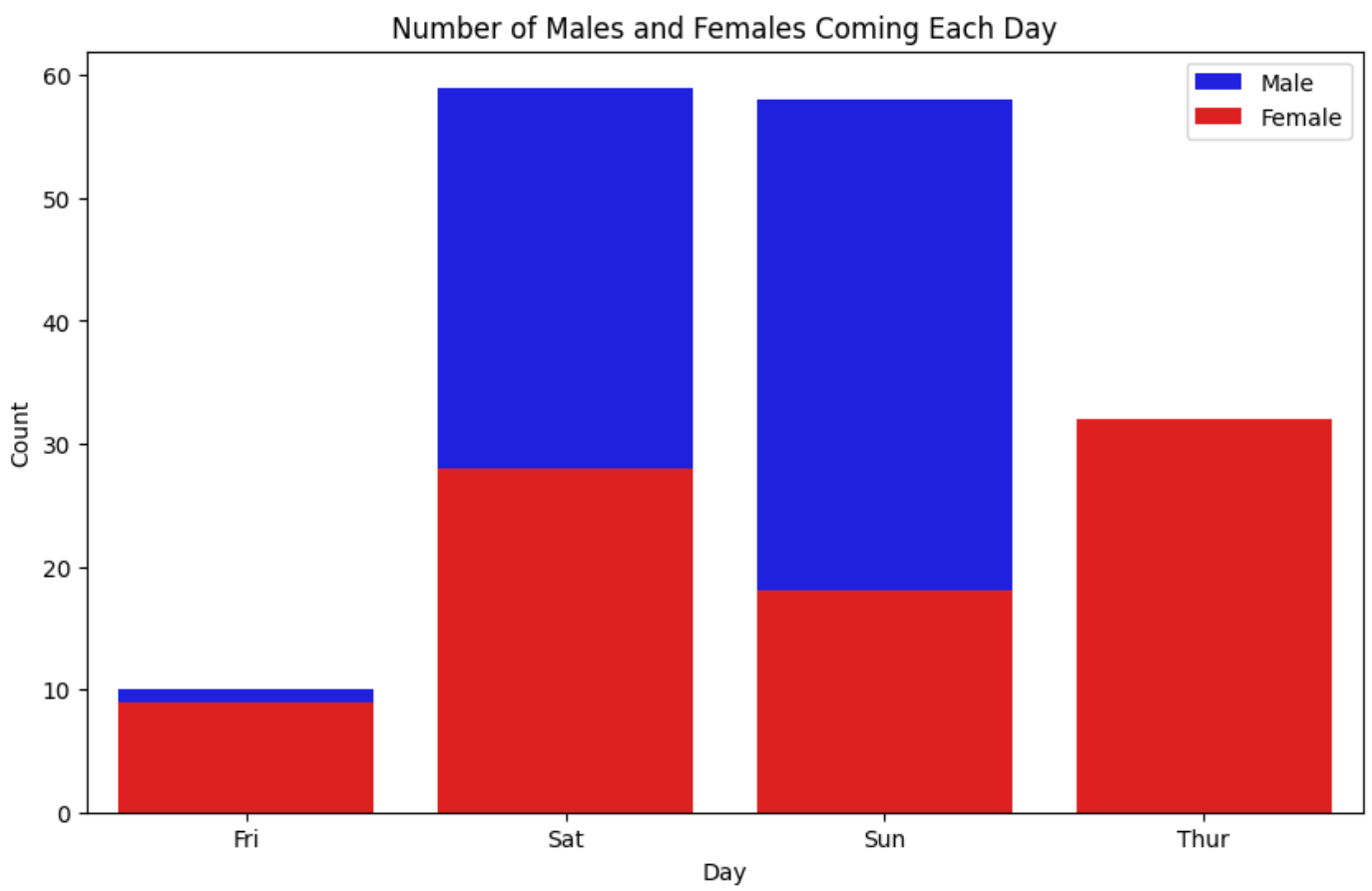


8. Find the number of male and female per day

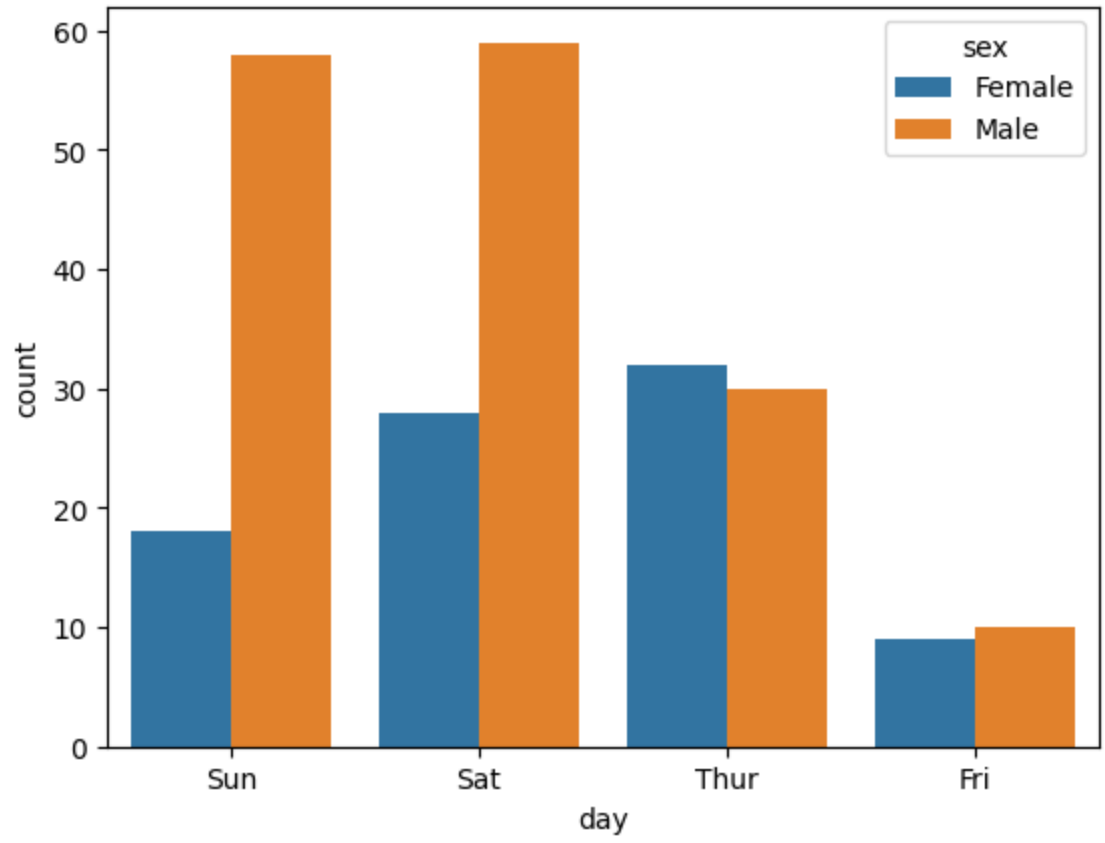
In [65]:
```
# Group the data by "day" and "sex" and count the occurrences
gender_per_day = df.groupby(['day', 'sex']).size().unstack()

# Fill any missing values with 0 (in case there are no occurrences of a particular gende
gender_per_day = gender_per_day.fillna(0)

# Display the result
print(gender_per_day)
```

```
sex    Female  Male
day
Fri         9    10
Sat        28    59
Sun        18    58
Thur       32    30
```

In [66]:
```
# Create a bar plot to visualize the number of males and females coming each day
plt.figure(figsize=(10, 6))
sns.barplot(data=gender_per_day, x=gender_per_day.index, y='Male', label='Male', color='
sns.barplot(data=gender_per_day, x=gender_per_day.index, y='Female', label='Female', col
plt.title('Number of Males and Females Coming Each Day')
plt.xlabel('Day')
plt.ylabel('Count')
plt.legend()
plt.show()
```



In [67]:
```
sns.countplot(x='day',data=df,hue='sex')
plt.show()
```



9. total_bill for genders per day

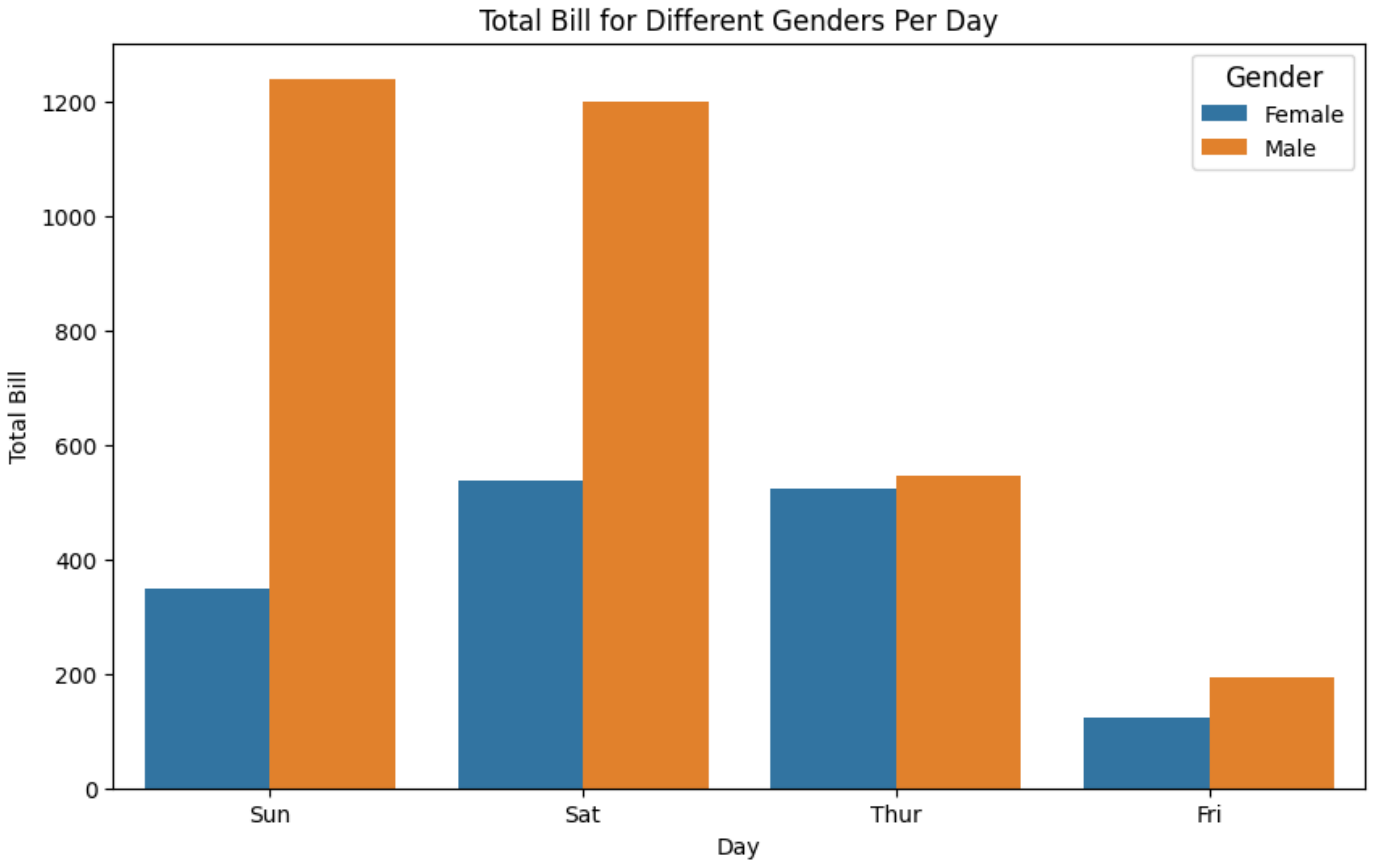
In [68]:
```
# Group the data by "day" and "sex" and sum the total bill for each group
```

```
total_bill_per_gender_per_day = df.groupby(['day', 'sex'])['total_bill'].sum().unstack()

# Fill any missing values with 0 (in case there are no occurrences of a particular gende
total_bill_per_gender_per_day = total_bill_per_gender_per_day.fillna(0)

# Display the result
print(total_bill_per_gender_per_day)
```

```
sex    Female  Male
day
Fri       123   195
Sat       538  1201
Sun       348  1240
Thur      522   547
```

In [69]:
```
# Create a bar plot to visualize the total bill for different genders per day
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='day', y='total_bill', hue='sex', estimator=sum, ci=None)
plt.title('Total Bill for Different Genders Per Day')
plt.xlabel('Day')
plt.ylabel('Total Bill')
plt.legend(title='Gender', title_fontsize='12')
plt.show()
```

```
/var/folders/rs/w1cppzrd0tg10lz07q7q90qm0000gn/T/ipykernel_2255/1431877828.py:3: FutureW
arning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='day', y='total_bill', hue='sex', estimator=sum, ci=None)
```

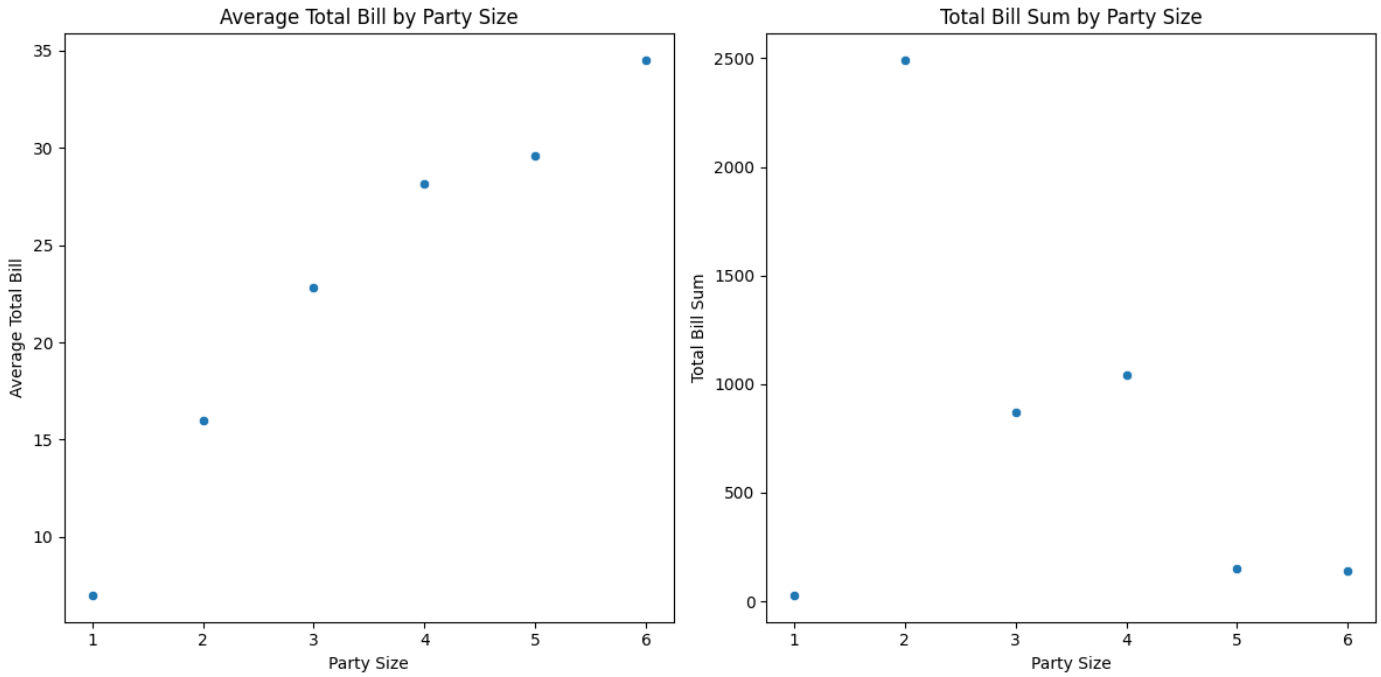


10. Party Size and Total Bill:

Investigate whether the size of the party correlates with the total bill. Do larger groups tend to spend more?

In [70]:
```
# Assuming your dataset is stored in a DataFrame named 'df'
party_size_stats = df.groupby('size').agg({
    'total_bill': ['mean', 'sum'],
    'tip': ['mean', 'sum']
}).reset_index()
```

In [71]:
```
# Create visualizations
plt.figure(figsize=(12, 6))

# Scatterplot for Average Total Bill by Party Size
plt.subplot(1, 2, 1)
sns.scatterplot(data=party_size_stats, x='size', y=('total_bill', 'mean'))
plt.title('Average Total Bill by Party Size')
plt.xlabel('Party Size')
plt.ylabel('Average Total Bill')

# Scatterplot for Total Bill Sum by Party Size
plt.subplot(1, 2, 2)
sns.scatterplot(data=party_size_stats, x='size', y=('total_bill', 'sum'))
plt.title('Total Bill Sum by Party Size')
plt.xlabel('Party Size')
plt.ylabel('Total Bill Sum')

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



Insights :

> .Maximum time people give 2 dollar tips and highest tips given is 10 dollar.
>
> .Mostly men pay the bill and less women come in compare to men.
>
> .Intersetingly 38% of women out of total women and 38% of men out of total men are smoker.
>
> .approximately 70% of sale done on weekends and out mostly from diner time.

In [73]:
```
jupyter nbconvert TIPS_DATASET_COMPLETE_EDA.ipynb --to pdf
```

```
  Cell In[73], line 1
    jupyter nbconvert TIPS_DATASET_COMPLETE_EDA.ipynb --to pdf
            ^
SyntaxError: invalid syntax
```

In [ ]: