Mahima Panchal

Visualizing Chipotle's Data

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import datetime
from datetime import datetime, date
from collections import Counter
# set this so the graphs open internally
%matplotlib inline
```

In [2]:
```
chipo = pd.read_csv('Downloads/chipotle (1).tsv', delimiter='\t')
```

In [3]:
```
chipo
```

Out[3]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98
...	...	...	...	...	...
4617	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	$11.75
4618	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	$11.75
4619	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$11.25
4620	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	$8.75
4621	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$8.75

4622 rows × 5 columns

Step 4. See the first 10 entries

In [4]:
```
chipo.head(10)
```

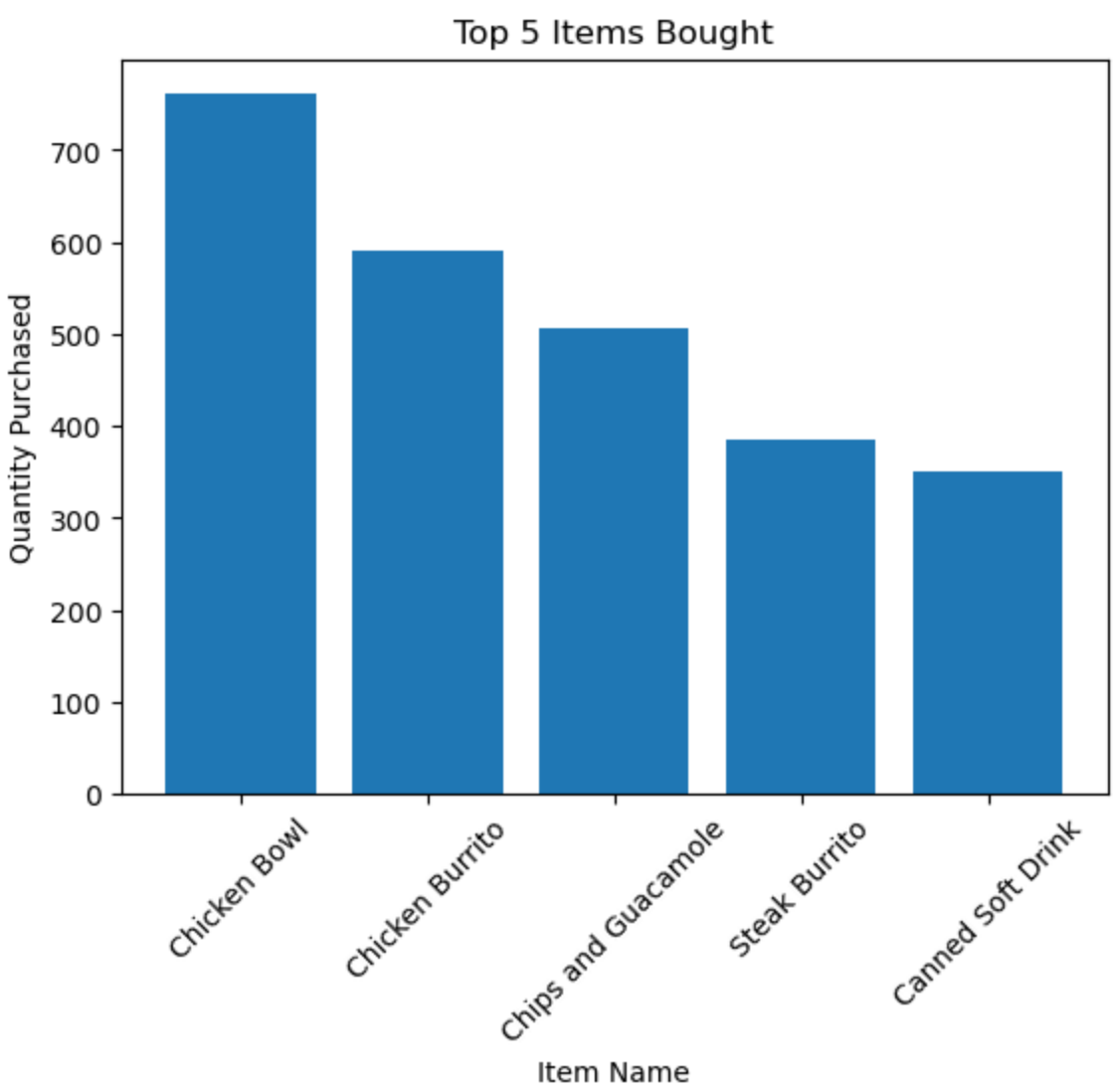
Out[4]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	$10.98
6	3	1	Side of Chips	NaN	$1.69
7	4	1	Steak Burrito	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	$11.75
8	4	1	Steak Soft Tacos	[Tomatillo Green Chili Salsa, [Pinto Beans, Ch...	$9.25
9	5	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Pinto...	$9.25

Step 5. Create a histogram of the top 5 items bought

In [5]:

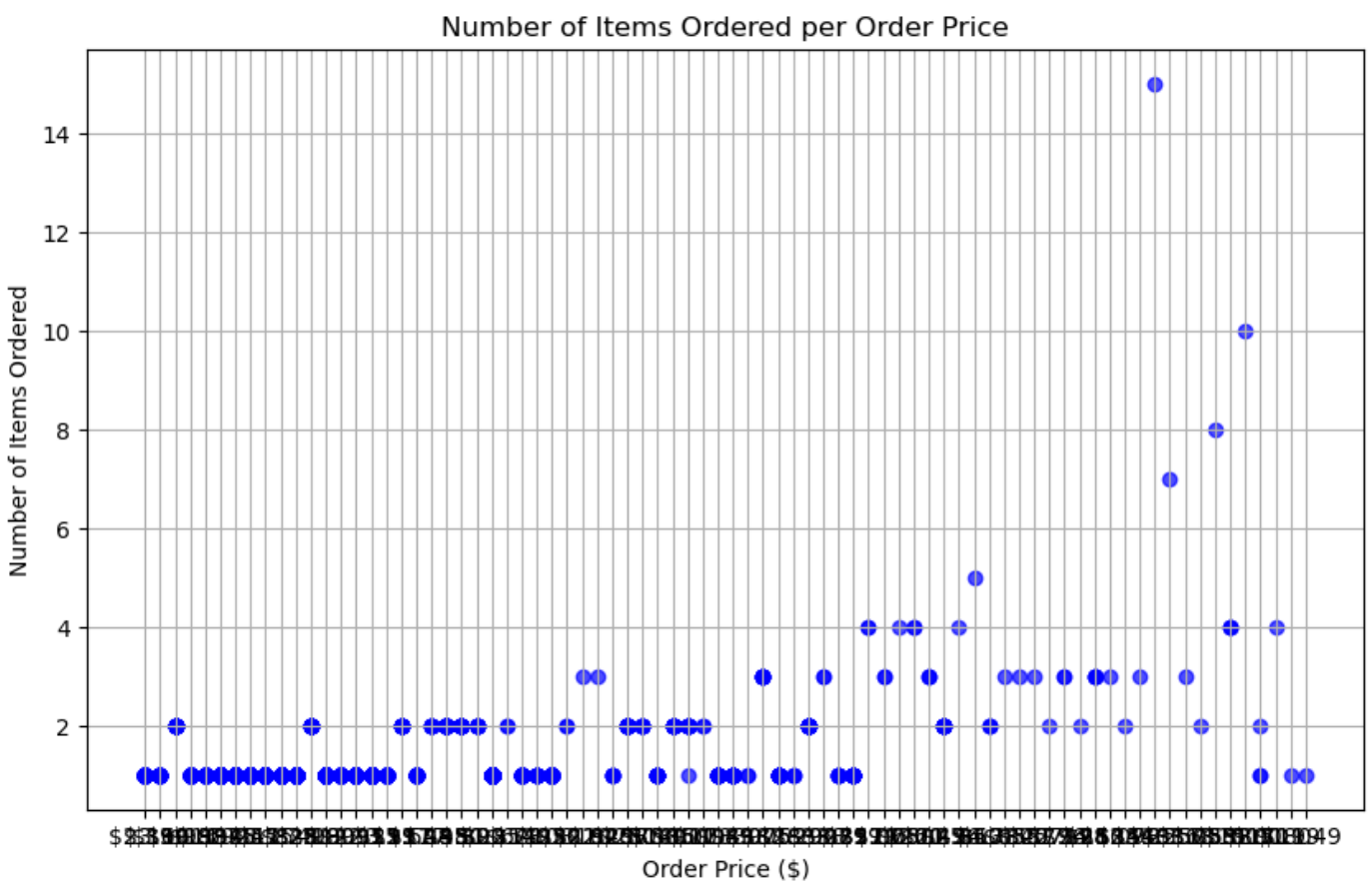
```
item_counts = chipo.groupby('item_name')['quantity'].sum()
top_5_items = item_counts.sort_values(ascending=False).head(5)

# Create a bar chart (histogram) for the top 5 items
plt.bar(top_5_items.index, top_5_items.values)
plt.xlabel('Item Name')
plt.ylabel('Quantity Purchased')
plt.title('Top 5 Items Bought')
plt.xticks(rotation=45)  # Rotate the x-axis labels for readability
plt.show()
```



Step 6. Create a scatterplot with the number of items orderered per order price

Hint: Price should be in the X-axis and Items ordered in the Y-axis

In [6]:
```
plt.figure(figsize=(10, 6))
plt.scatter(chipo['item_price'], chipo['quantity'], alpha=0.7, marker='o', color='b')
plt.xlabel('Order Price ($)')
plt.ylabel('Number of Items Ordered')
plt.title('Number of Items Ordered per Order Price')
plt.grid(True)
plt.show()
```



Online Retails Purchase

In [7]:
```
online_rt = pd.read_csv("Downloads/Online_Retail.txt", 'r',  encoding='ISO-8859-1')
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_12508\3119798623.py:1: FutureWarning: In a fu
ture version of pandas all arguments of read_csv except for the argument 'filepath_or_bu
ffer' will be keyword-only.
  online_rt = pd.read_csv("Downloads/Online_Retail.txt", 'r',  encoding='ISO-8859-1')
C:\Users\Dell\AppData\Local\Temp\ipykernel_12508\3119798623.py:1: DtypeWarning: Columns
(3,4) have mixed types. Specify dtype option on import or set low_memory=False.
  online_rt = pd.read_csv("Downloads/Online_Retail.txt", 'r',  encoding='ISO-8859-1')
```

In [8]:
```
online_rt
```

Out[8]:

	InvoiceNo,StockCode,Desc	iption,Quantity,InvoiceDate,UnitP	ice,Custome	ID,Count	y
0	536365,85123A,WHITE HANGING HEART T-LIGHT HOLD...	NaN	NaN	NaN	NaN
1	536365,71053,WHITE METAL LANTERN,6,12/1/10 8:2...	NaN	NaN	NaN	NaN
2	536365,84406B,CREAM CUPID HEARTS COAT HANGER,8...	NaN	NaN	NaN	NaN
3	536365,84029G,KNITTED UNION FLAG HOT WATER BOT...	NaN	NaN	NaN	NaN
4	536365,84029E,RED WOOLLY HOTTIE WHITE HEART.,6...	NaN	NaN	NaN	NaN
...	...	...	...	...	...
541904	581587,22613,PACK OF 20 SPACEBOY NAPKINS,12,12...	ance	NaN	NaN	NaN
541905	581587,22899,CHILDREN'S APRON DOLLY GIRL ,6,12...	ance	NaN	NaN	NaN
541906	581587,23254,CHILDRENS CUTLERY DOLLY GIRL ,4,1...	ance	NaN	NaN	NaN
541907	581587,23255,CHILDRENS CUTLERY CIRCUS PARADE,4...	ance	NaN	NaN	NaN
541908	581587,22138,BAKING SET 9 PIECE RETROSPOT ,3,1...	ance	NaN	NaN	NaN

541909 rows × 5 columns

In [9]:
```
online_rt.columns
```

Out[9]:
```
Index(['InvoiceNo,StockCode,Desc', 'iption,Quantity,InvoiceDate,UnitP',
       'ice,Custome', 'ID,Count', 'y'],
      dtype='object')
```

Scores

Step 2. Create the DataFrame that should look like the one below.

In [10]:
```
data = {
    'first_name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
    'last_name': ['Miller', 'Jacobson', 'Ali', 'Milner', 'Cooze'],
    'age': [42, 52, 36, 24, 73],
    'female': [0, 1, 1, 0, 1],
    'preTestScore': [4, 24, 31, 2, 3],
    'postTestScore': [25, 94, 57, 62, 70]
}

df = pd.DataFrame(data)

print(df)
```

```
  first_name last_name  age  female  preTestScore  postTestScore
0      Jason    Miller   42       0             4             25
1      Molly  Jacobson   52       1            24             94
2       Tina       Ali   36       1            31             57
3       Jake    Milner   24       0             2             62
4        Amy     Cooze   73       1             3             70
```

In [11]:
```
df
```

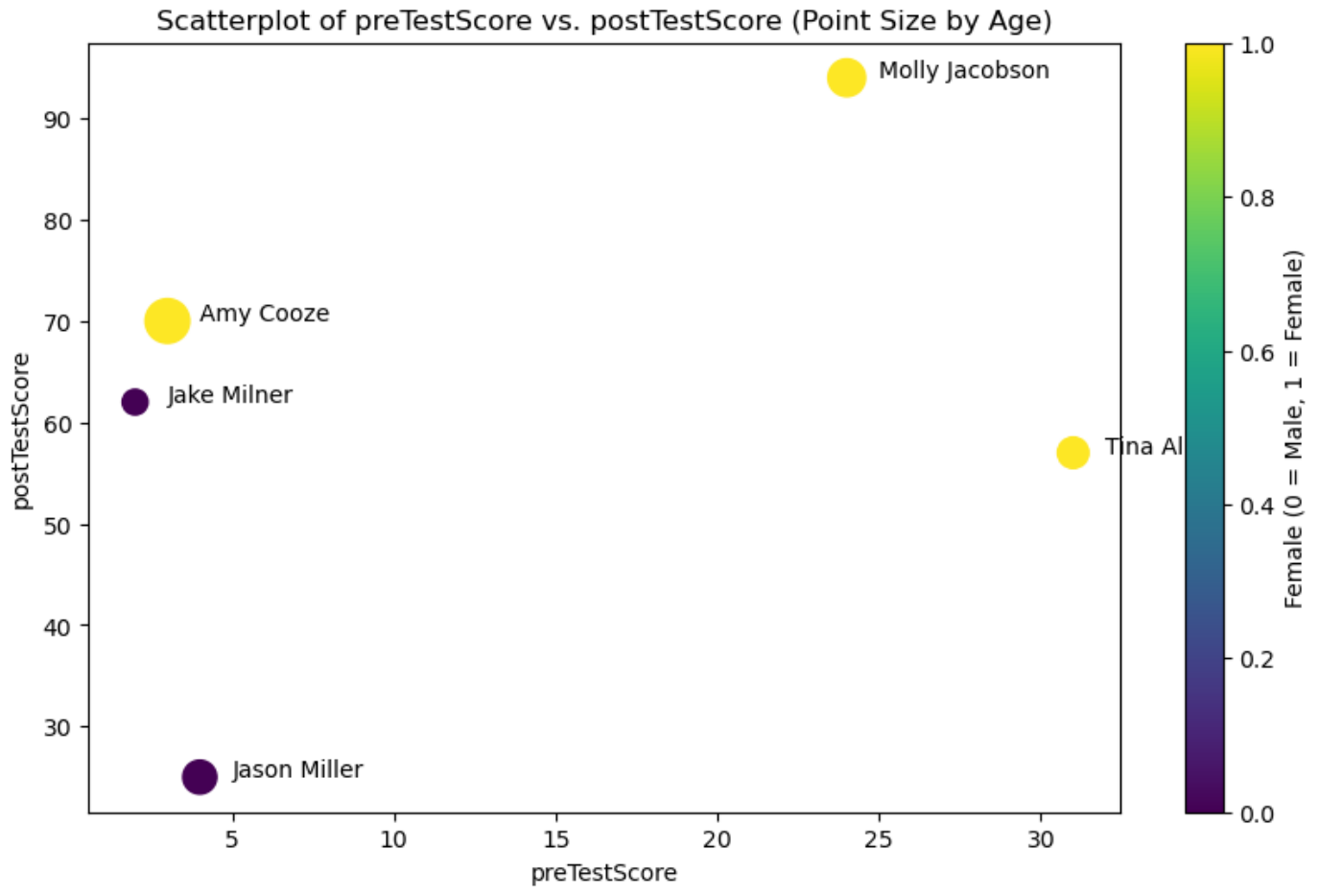
Out[11]:

	first_name	last_name	age	female	preTestScore	postTestScore
0	Jason	Miller	42	0	4	25
1	Molly	Jacobson	52	1	24	94
2	Tina	Ali	36	1	31	57
3	Jake	Milner	24	0	2	62
4	Amy	Cooze	73	1	3	70

Step 3. Create a Scatterplot of preTestScore and postTestScore, with the size of each point determined by age

Hint: Don't forget to place the labels

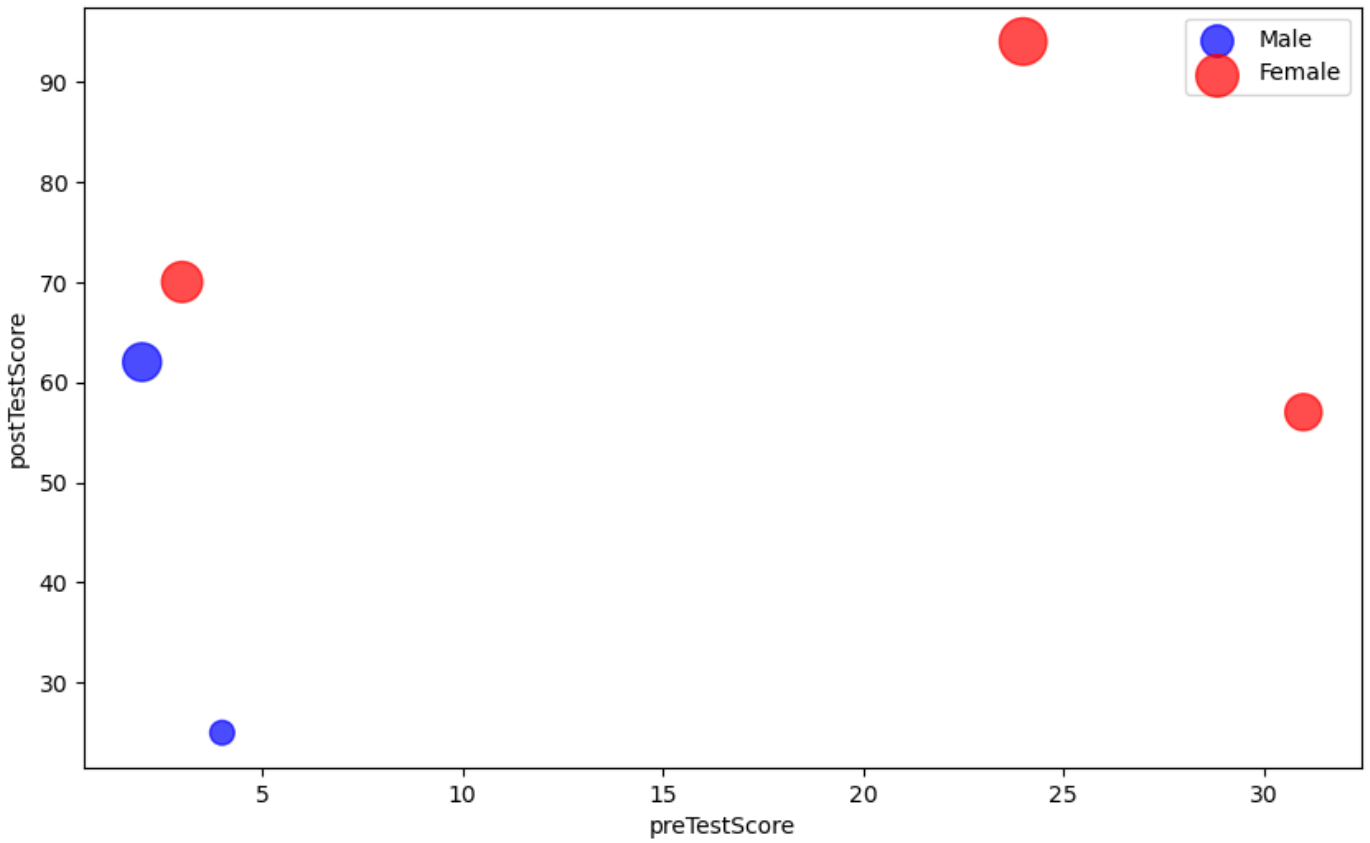
In [12]:
```
# Create a scatterplot
plt.figure(figsize=(10, 6))
scatter = plt.scatter(df['preTestScore'], df['postTestScore'], s=df['age'] * 5, c=df['fe

# Add labels for each point
for i in range(len(df)):
    plt.text(df['preTestScore'][i] + 1, df['postTestScore'][i], f"{df['first_name'][i]}

# Add labels and a colorbar
plt.xlabel('preTestScore')
plt.ylabel('postTestScore')
plt.title('Scatterplot of preTestScore vs. postTestScore (Point Size by Age)')
plt.colorbar(scatter, label='Female (0 = Male, 1 = Female)')
plt.show()
```



Step 4. Create a Scatterplot of preTestScore and postTestScore.

This time the size should be 4.5 times the postTestScore and the color determined by sex

In [13]:
```
plt.figure(figsize=(10, 6))

# Define colors for Male (0) and Female (1)
colors = {0: 'blue', 1: 'red'}

marker_sizes = df['postTestScore'] * 4.5

for sex, group in df.groupby('female'):
    plt.scatter(
        group['preTestScore'],
        group['postTestScore'],
        s=marker_sizes.loc[group.index],
        c=colors[sex],
        label='Female' if sex == 1 else 'Male',
        alpha=0.7
    )

plt.xlabel('preTestScore')
plt.ylabel('postTestScore')
plt.legend()

plt.show()
```



Tips

In [14]:
```
tips = pd.read_csv("Downloads/tips.txt")
```

In [15]:
```
tips
```

Out[15]:

	Unnamed: 0	total_bill	tip	sex	smoker	day	time	size
0	0	16.99	1.01	Female	No	Sun	Dinner	2
1	1	10.34	1.66	Male	No	Sun	Dinner	3
2	2	21.01	3.50	Male	No	Sun	Dinner	3
3	3	23.68	3.31	Male	No	Sun	Dinner	2
4	4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...	...
239	239	29.03	5.92	Male	No	Sat	Dinner	3
240	240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	242	17.82	1.75	Male	No	Sat	Dinner	2
243	243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 8 columns

Step 4. Delete the Unnamed 0 column

In [16]:
```
del tips['Unnamed: 0']
```

In [17]:
```
tips
```

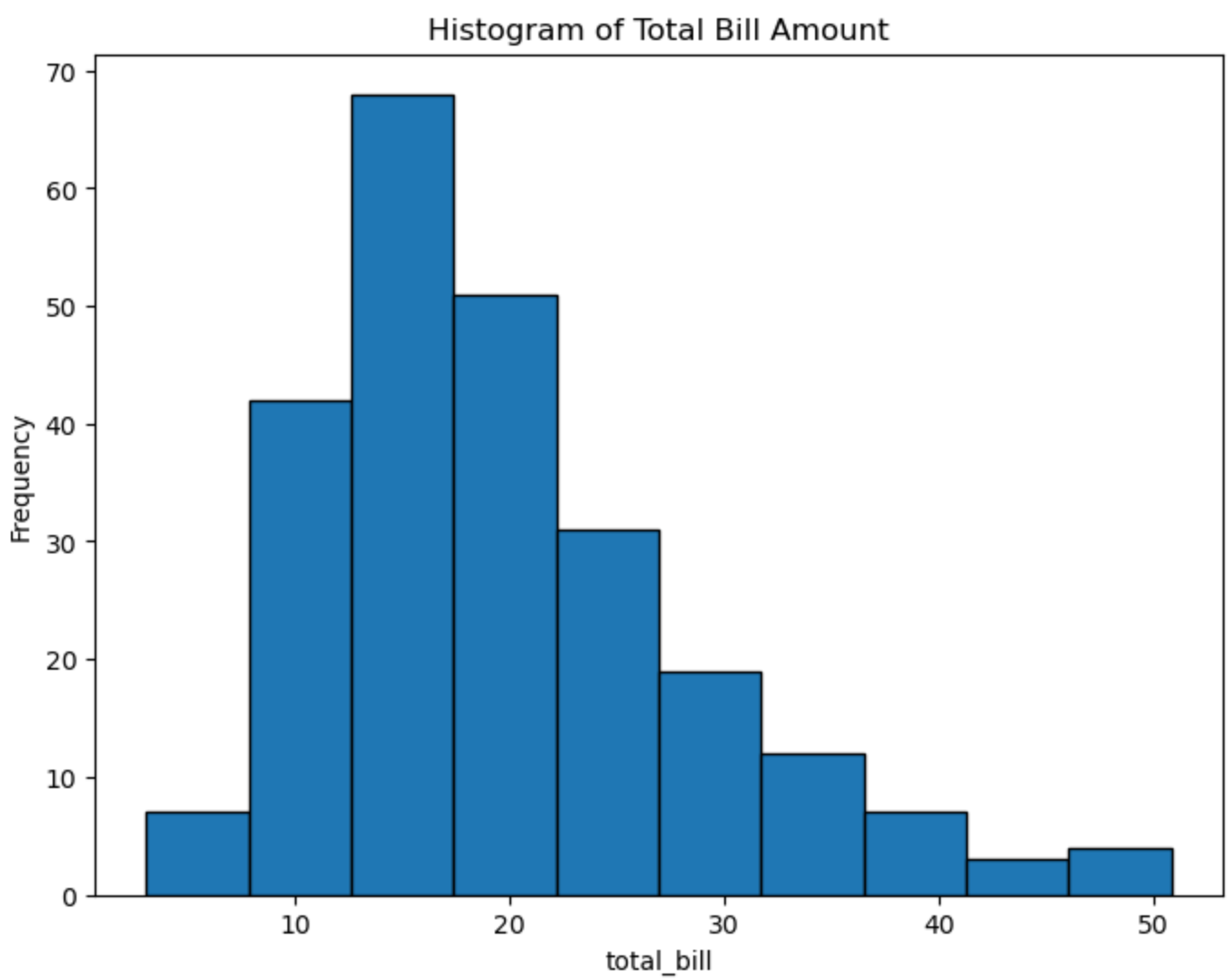
Out[17]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

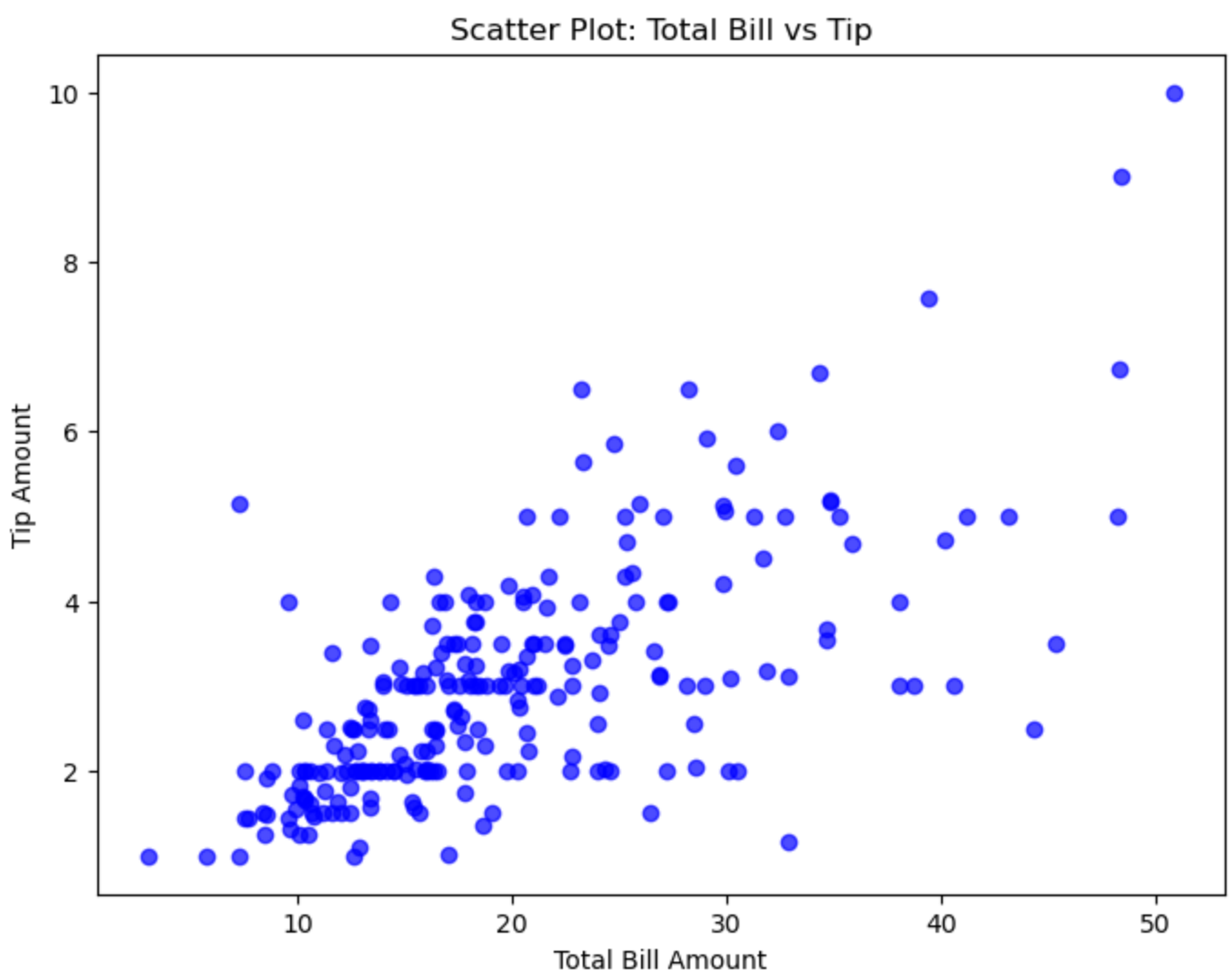
Step 5. Plot the total_bill column histogram

In [18]:
```
plt.figure(figsize=(8, 6))
plt.hist(tips['total_bill'], bins=10, edgecolor='black')
plt.xlabel('total_bill')
plt.ylabel('Frequency')
plt.title('Histogram of Total Bill Amount')
plt.show()
```



Step 6. Create a scatter plot presenting the relationship between total_bill and tip

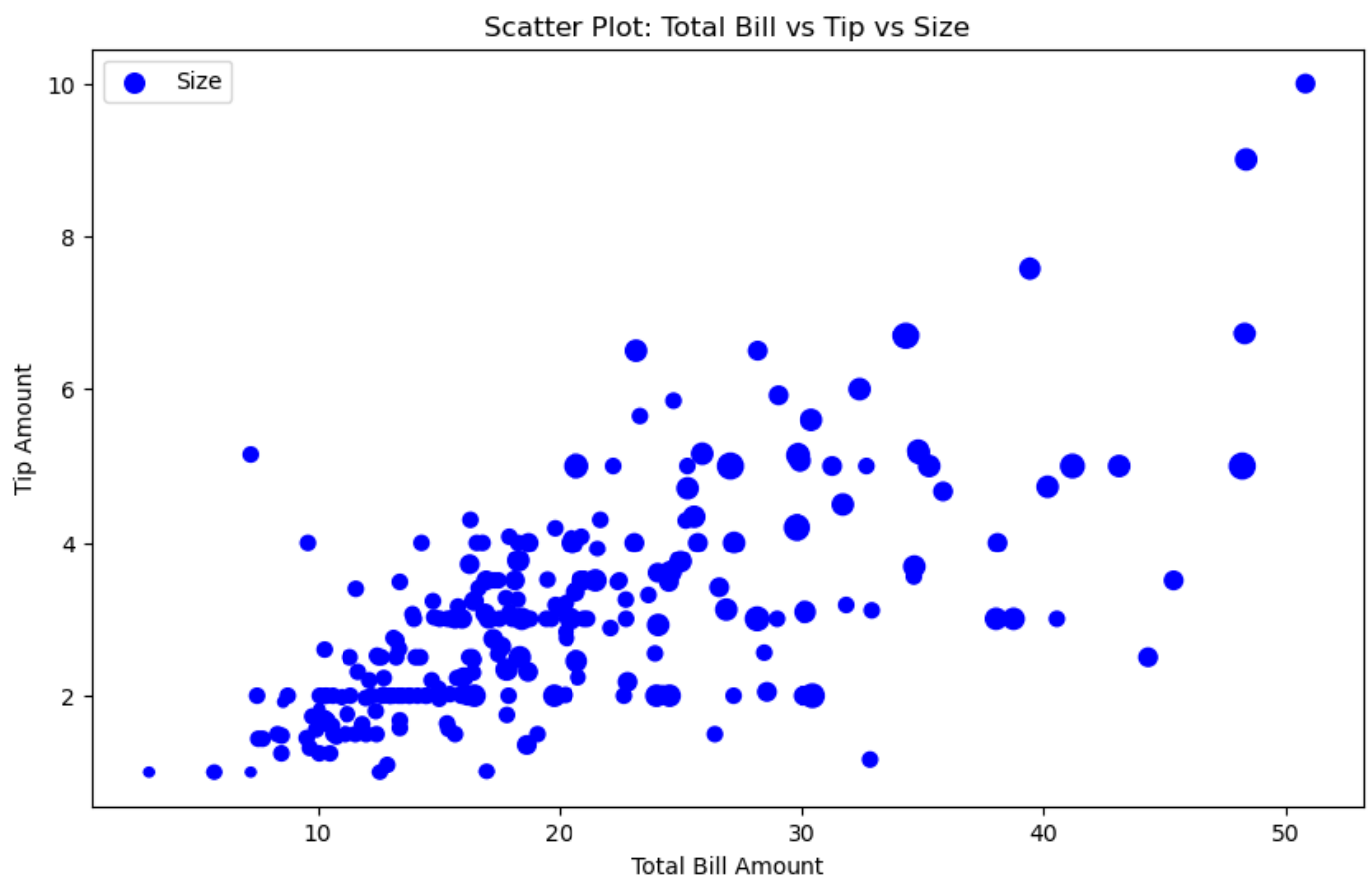
In [19]:
```
plt.figure(figsize=(8, 6))
plt.scatter(tips['total_bill'], tips['tip'], c='blue', alpha=0.7)

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

plt.show()
```



Step 7. Create one image with the relationship of total_bill, tip and size.

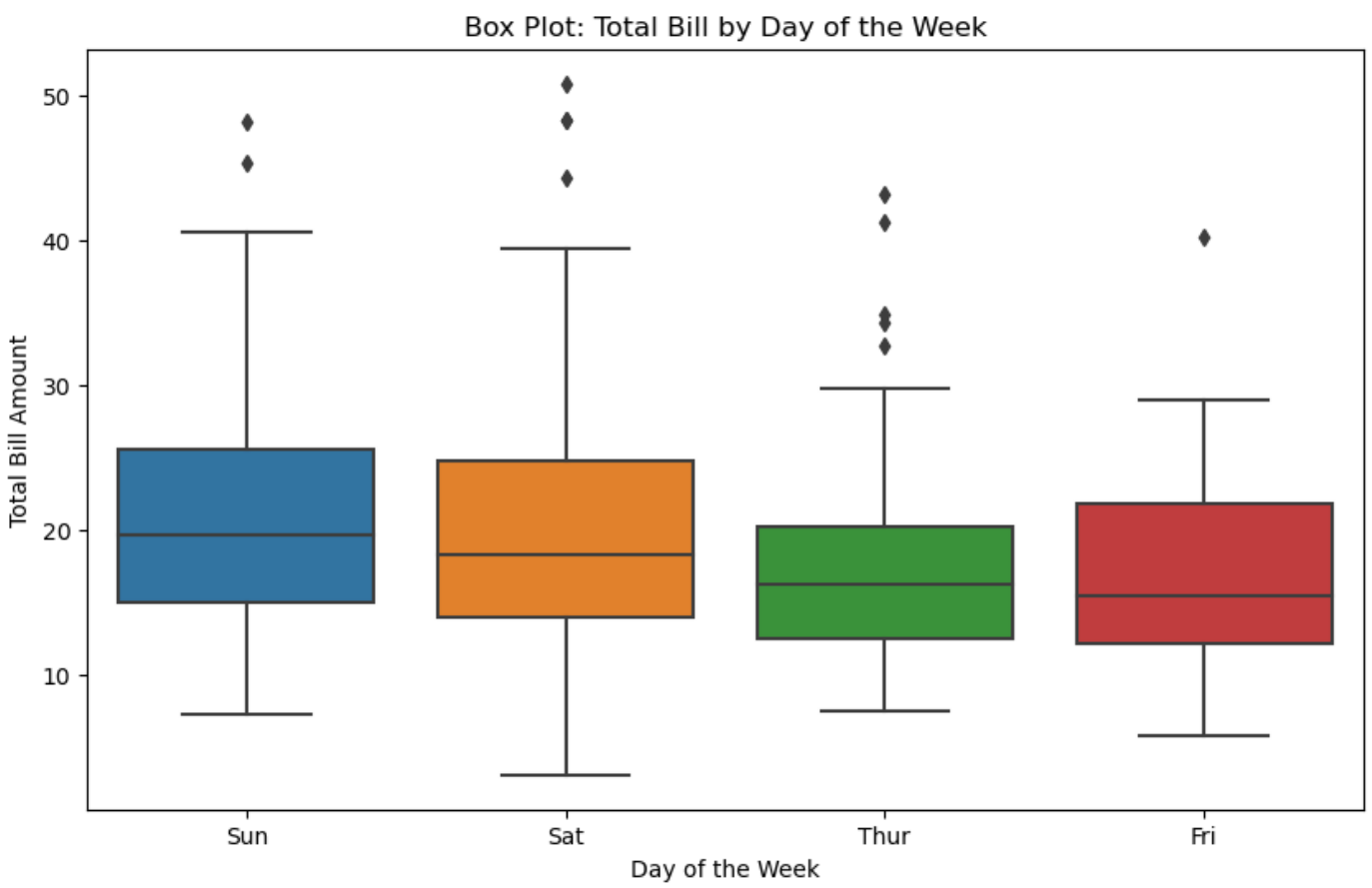
Hint: It is just one function.

In [20]:
```
plt.figure(figsize=(10, 6))

plt.scatter(tips['total_bill'], tips['tip'], s=tips['size'] * 20, c='blue', label='Size'

plt.xlabel('Total Bill Amount')
plt.ylabel('Tip Amount')
plt.title('Scatter Plot: Total Bill vs Tip vs Size')
plt.legend()

plt.show()
```



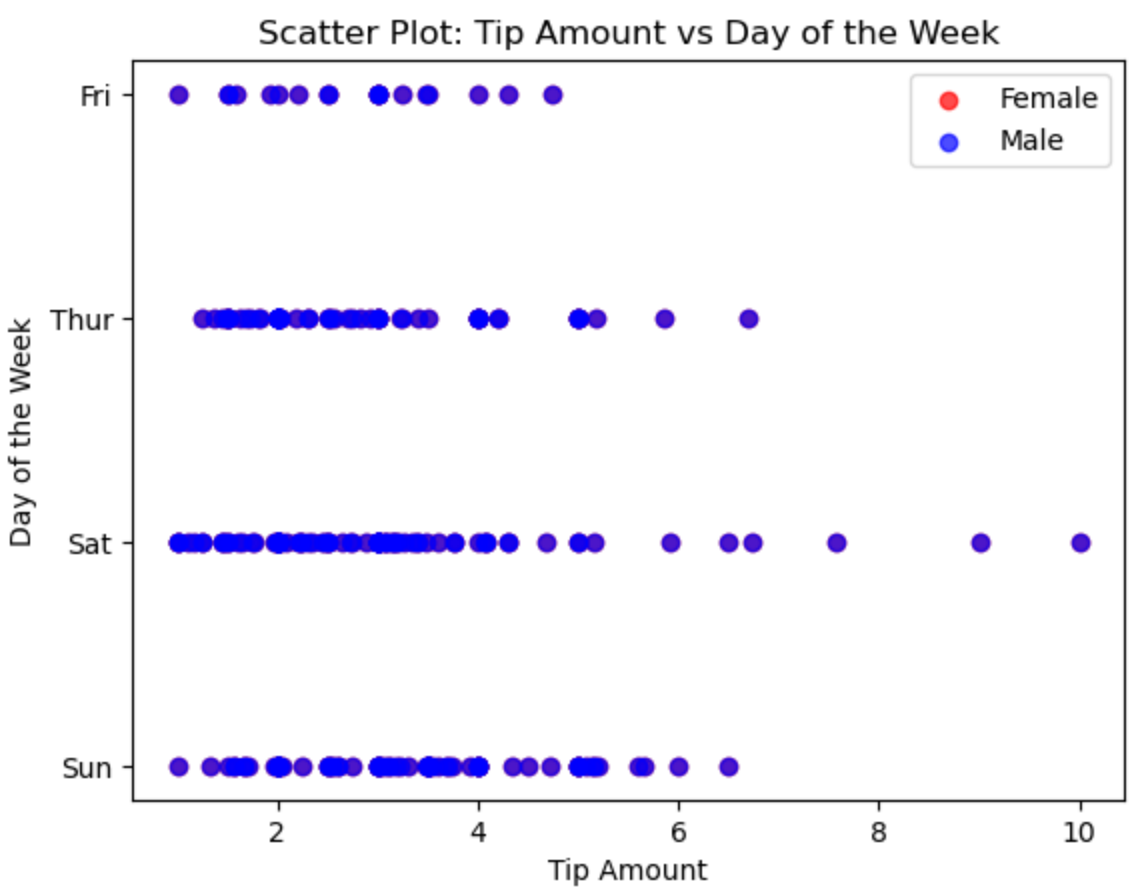
Step 8. Present the relationship between days and total_bill value.

In [21]:
```
# BOX PLOT

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

sns.boxplot(x='day', y='total_bill', data=tips)

# Add labels and a title
plt.xlabel('Day of the Week')
plt.ylabel('Total Bill Amount')
plt.title('Box Plot: Total Bill by Day of the Week')

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



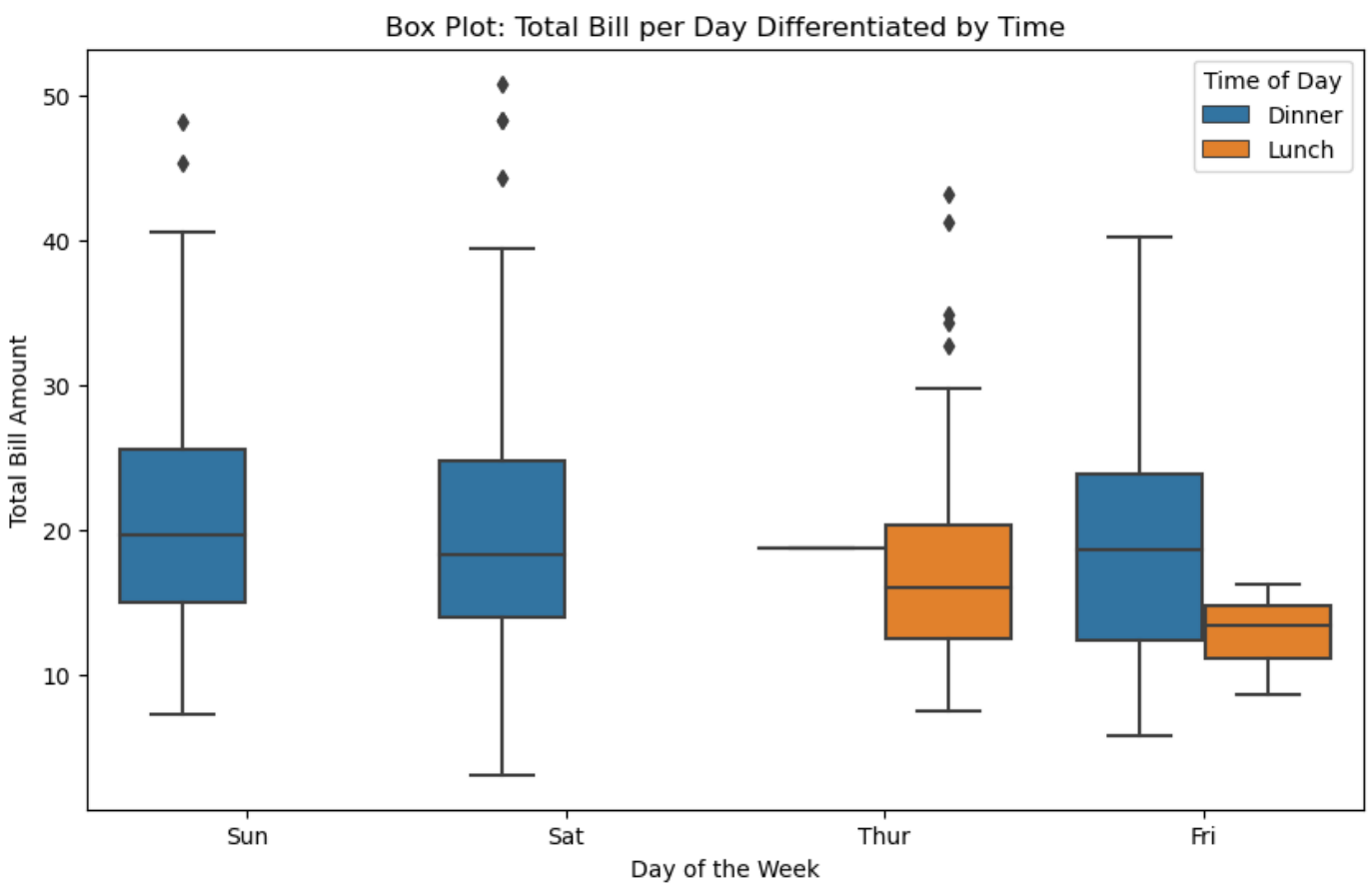
Step 9. Create a scatter plot with the day as the y-axis and tip as the x-axis, differ the dots by sex

In [22]:
```
colors = {'Male': 'blue', 'Female': 'red'}

for sex, group in tips.groupby('sex'):
    plt.scatter(
        tips['tip'],
        tips['day'],
        c=colors[sex],
        label=sex,
        alpha=0.7
    )

plt.xlabel('Tip Amount')
plt.ylabel('Day of the Week')
plt.title('Scatter Plot: Tip Amount vs Day of the Week')
plt.legend()

plt.show()
```



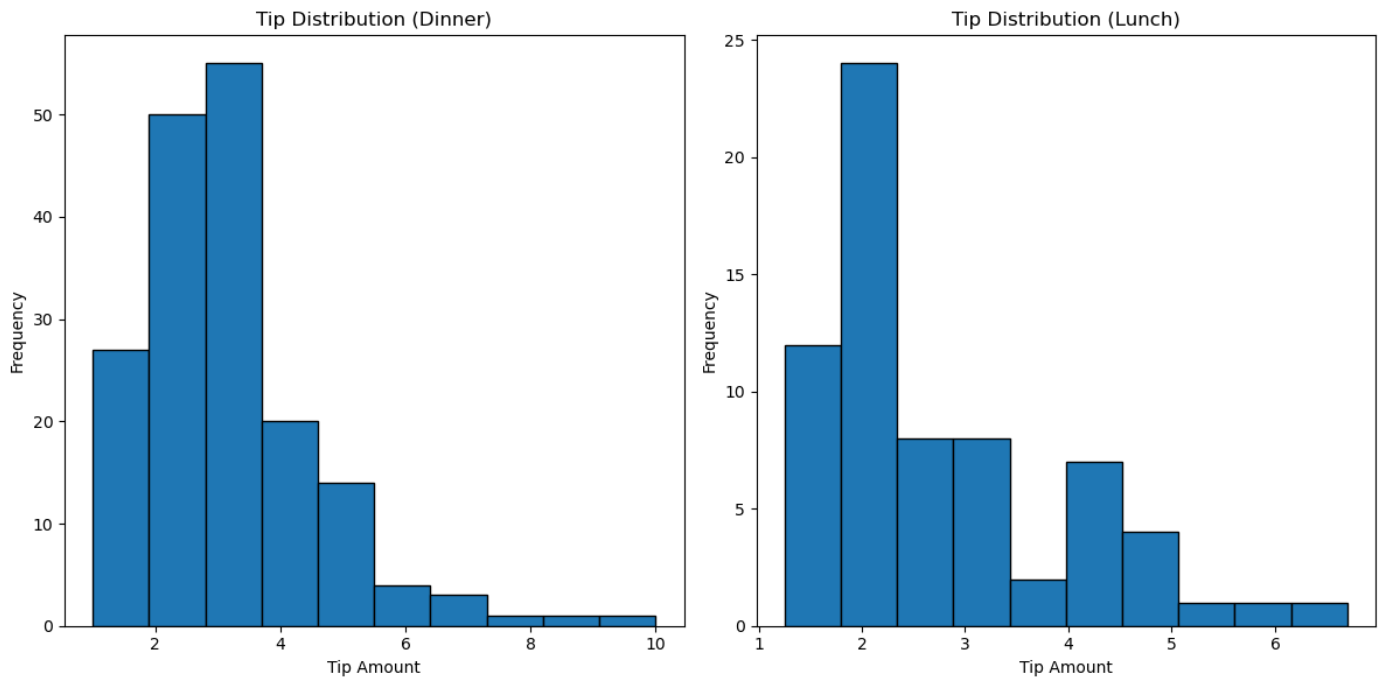
Step 10. Create a box plot presenting the total_bill per day differetiation the time (Dinner or Lunch).

In [23]:
```
plt.figure(figsize=(10, 6))

sns.boxplot(x='day', y='total_bill', hue='time', data=tips)

plt.xlabel('Day of the Week')
plt.ylabel('Total Bill Amount')
plt.title('Box Plot: Total Bill per Day Differentiated by Time')

plt.legend(title='Time of Day')

plt.show()
```



Step 11. Create two histograms of the tip value based for Dinner and Lunch. They must be side by side.

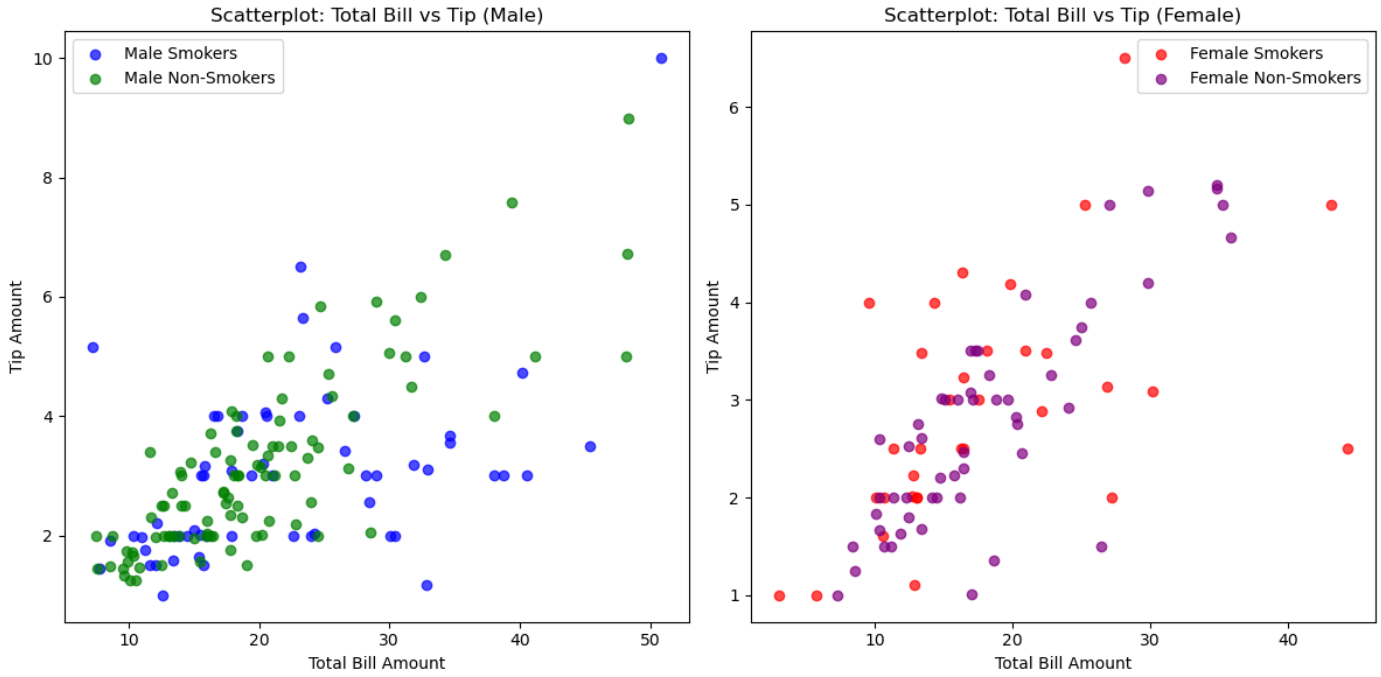
In [24]:
```
plt.figure(figsize=(12, 6))

# Histogram for 'Dinner'
plt.subplot(1, 2, 1)
plt.hist(tips[tips['time'] == 'Dinner']['tip'], bins=10, edgecolor='black')
plt.title('Tip Distribution (Dinner)')
plt.xlabel('Tip Amount')
plt.ylabel('Frequency')

# Histogram for 'Lunch'
plt.subplot(1, 2, 2)
plt.hist(tips[tips['time'] == 'Lunch']['tip'], bins=10, edgecolor='black')
plt.title('Tip Distribution (Lunch)')
plt.xlabel('Tip Amount')
plt.ylabel('Frequency')

plt.tight_layout()

plt.show()
```



Step 12. Create two scatterplots graphs, one for Male and another for Female, presenting the total_bill value and tip relationship, differing by smoker or no smoker They must be side by side.

In [25]:
```
plt.figure(figsize=(12, 6))

# Scatterplot for Male
plt.subplot(1, 2, 1)
plt.scatter(
    tips[(tips['sex'] == 'Male') & (tips['smoker'] == 'Yes')]['total_bill'],
    tips[(tips['sex'] == 'Male') & (tips['smoker'] == 'Yes')]['tip'],
    label='Male Smokers',
    c='blue',
    alpha=0.7
)
plt.scatter(
    tips[(tips['sex'] == 'Male') & (tips['smoker'] == 'No')]['total_bill'],
    tips[(tips['sex'] == 'Male') & (tips['smoker'] == 'No')]['tip'],
    label='Male Non-Smokers',
    c='green',
    alpha=0.7
)
plt.title('Scatterplot: Total Bill vs Tip (Male)')
plt.xlabel('Total Bill Amount')
plt.ylabel('Tip Amount')
plt.legend()

# Scatterplot for Female
plt.subplot(1, 2, 2)
plt.scatter(
    tips[(tips['sex'] == 'Female') & (tips['smoker'] == 'Yes')]['total_bill'],
    tips[(tips['sex'] == 'Female') & (tips['smoker'] == 'Yes')]['tip'],
    label='Female Smokers',
    c='red',
    alpha=0.7
)
plt.scatter(
    tips[(tips['sex'] == 'Female') & (tips['smoker'] == 'No')]['total_bill'],
    tips[(tips['sex'] == 'Female') & (tips['smoker'] == 'No')]['tip'],
    label='Female Non-Smokers',
    c='purple',
```

```
    alpha=0.7
)
plt.title('Scatterplot: Total Bill vs Tip (Female)')
plt.xlabel('Total Bill Amount')
plt.ylabel('Tip Amount')
plt.legend()

plt.tight_layout()

plt.show()
```



Visualizing the Titanic Disaster

In [26]:
```
titanic = pd.read_csv("Downloads/train.txt")
```

In [27]:
```
titanic
```

Out[27]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	

891 rows × 12 columns

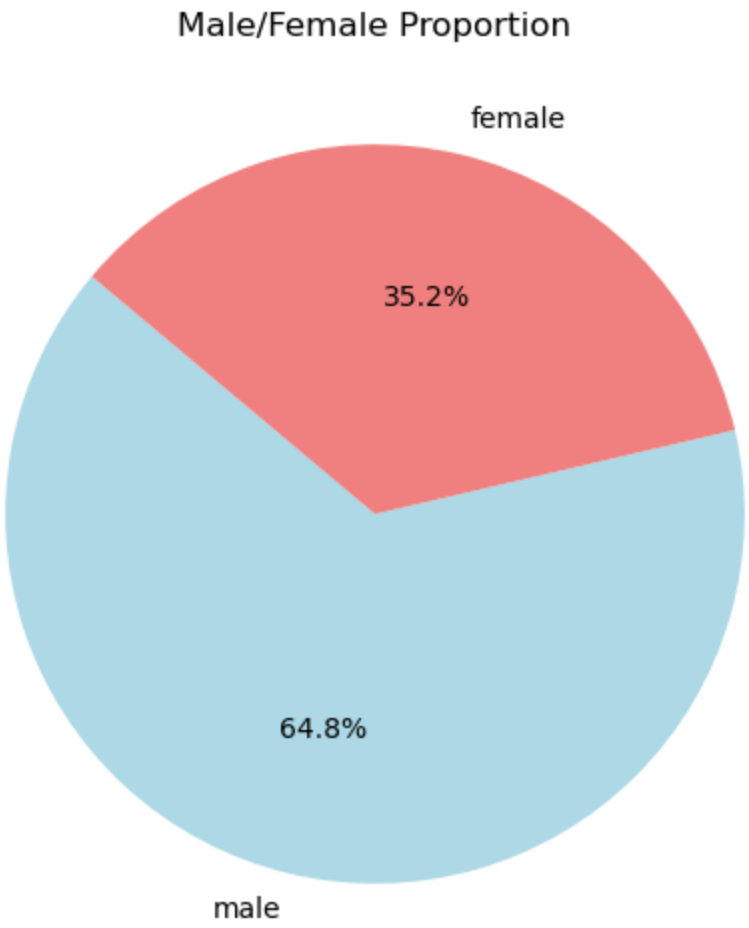
Step 4. Set PassengerId as the index

In [28]:
```
titanic.set_index('PassengerId', inplace=True)
```

Step 5. Create a pie chart presenting the male/female proportion

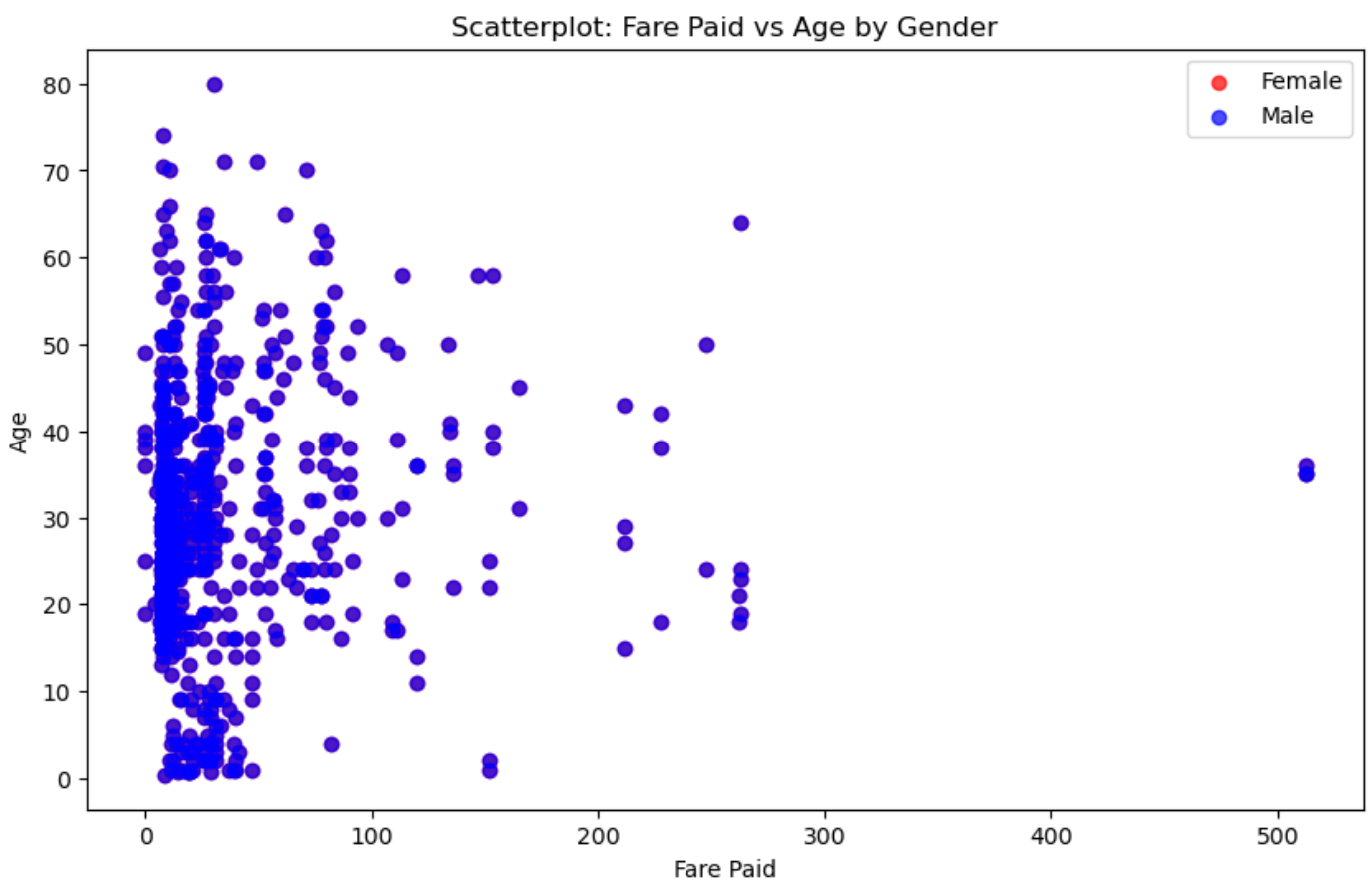
In [29]:
```
gender_counts = titanic['Sex'].value_counts()

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

colors = ['lightblue', 'lightcoral']

plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', colors=colors, sta
```

```
plt.title('Male/Female Proportion')

plt.show()
```



Step 6. Create a scatterplot with the Fare payed and the Age, differ the plot color by gender

In [30]:
```
colors = {'male': 'blue', 'female': 'red'}

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

for gender, group in titanic.groupby('Sex'):
    plt.scatter(
        titanic['Fare'],
        titanic['Age'],
        c=colors[gender.lower()],
        label=gender.capitalize(),
        alpha=0.7
    )

plt.xlabel('Fare Paid')
plt.ylabel('Age')
plt.title('Scatterplot: Fare Paid vs Age by Gender')
plt.legend()

plt.show()
```



Step 7. How many people survived?

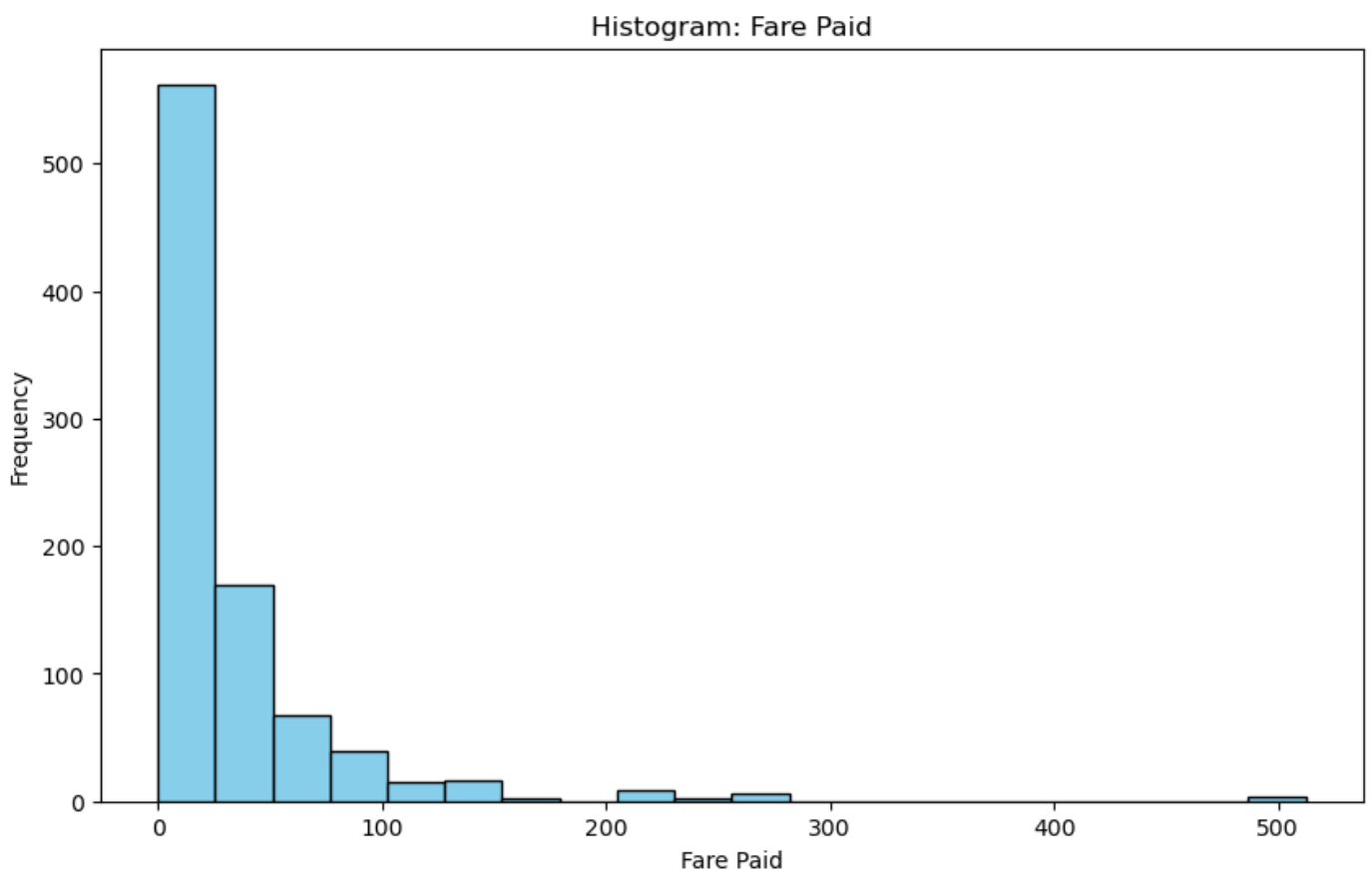
In [31]:
```
survived_count = titanic['Survived'].sum()
```

In [32]:
```
survived_count
```

Out[32]:
```
342
```

Step 8. Create a histogram with the Fare payed

In [33]:
```
plt.figure(figsize=(10, 6))

plt.hist(titanic['Fare'], bins=20, edgecolor='black', color='skyblue')

plt.xlabel('Fare Paid')
plt.ylabel('Frequency')
plt.title('Histogram: Fare Paid')

plt.show()
```



Pokemon

Step 3: Assign the data dictionary to a variable called 'pokemon'

In [34]:
```
data = {
    'name': ['Ivysaur', 'Charmeleon', 'Wartortle', 'Metapod'],
    'hp': [45, 39, 44, 45],
    'evolution': ['Bulbasaur', 'Charmander', 'Squirtle', 'Caterpie'],
    'pokedex': ['yes', 'no', 'yes', 'no'],
    'type': ['grass', 'fire', 'water', 'bug']
}

pokemon = pd.DataFrame(data)
```

In [35]:
```
pokemon
```

Out[35]:

	name	hp	evolution	pokedex	type
0	Ivysaur	45	Bulbasaur	yes	grass
1	Charmeleon	39	Charmander	no	fire
2	Wartortle	44	Squirtle	yes	water
3	Metapod	45	Caterpie	no	bug

Step 4. Ops...it seems the DataFrame columns are in alphabetical order. Place the order of the columns as name, type, hp, evolution, pokedex

In [36]:
```
pokemon = pokemon[['name', 'type', 'hp', 'evolution', 'pokedex']]

print(pokemon)
```

```
         name   type  hp   evolution pokedex
0     Ivysaur  grass  45   Bulbasaur     yes
1  Charmeleon   fire  39  Charmander      no
2   Wartortle  water  44    Squirtle     yes
3     Metapod    bug  45    Caterpie      no
```

In [37]:
```
pokemon
```

Out[37]:

	name	type	hp	evolution	pokedex
0	Ivysaur	grass	45	Bulbasaur	yes
1	Charmeleon	fire	39	Charmander	no
2	Wartortle	water	44	Squirtle	yes
3	Metapod	bug	45	Caterpie	no

Step 5. Add another column called place, and insert what you have in mind.

In [38]:
```
pokemon['place'] = ['forest', 'volcano', 'ocean', 'meadow']
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_12508\3995600451.py:1: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  pokemon['place'] = ['forest', 'volcano', 'ocean', 'meadow']
```

In [39]:
```
pokemon
```

Out[39]:

	name	type	hp	evolution	pokedex	place
0	Ivysaur	grass	45	Bulbasaur	yes	forest
1	Charmeleon	fire	39	Charmander	no	volcano
2	Wartortle	water	44	Squirtle	yes	ocean
3	Metapod	bug	45	Caterpie	no	meadow

Step 6. Present the type of each column

In [40]:
```
column_types = pokemon.dtypes
```

In [41]:
```
column_types
```

Out[41]:
```
name         object
type         object
hp            int64
evolution    object
pokedex      object
place        object
dtype: object
```

Apple Stock

In [42]:
```
apple = pd.read_csv("Downloads/appl_1980_2014.txt")
```

In [43]:
```
apple
```

Out[43]:

	Date	Open	High	Low	Close	Volume	Adj Close
0	2014-07-08	96.27	96.80	93.92	95.35	65130000	95.35
1	2014-07-07	94.14	95.99	94.10	95.97	56305400	95.97
2	2014-07-03	93.67	94.10	93.20	94.03	22891800	94.03
3	2014-07-02	93.87	94.06	93.09	93.48	28420900	93.48
4	2014-07-01	93.52	94.07	93.13	93.52	38170200	93.52
...	...	...	...	...	...	...	...
8460	1980-12-18	26.63	26.75	26.63	26.63	18362400	0.41
8461	1980-12-17	25.87	26.00	25.87	25.87	21610400	0.40
8462	1980-12-16	25.37	25.37	25.25	25.25	26432000	0.39
8463	1980-12-15	27.38	27.38	27.25	27.25	43971200	0.42
8464	1980-12-12	28.75	28.87	28.75	28.75	117258400	0.45

8465 rows × 7 columns

Step 4. Check out the type of the columns

In [44]:
```
column_types = apple.dtypes
```

In [45]:
```
column_types
```

Out[45]:
```
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Volume         int64
Adj Close    float64
dtype: object
```

Step 5. Transform the Date column as a datetime type

In [46]:
```
apple['Date'] = pd.to_datetime(apple['Date'])

print("Data Type of the 'Date' Column:")
print(apple['Date'].dtype)
```

```
Data Type of the 'Date' Column:
datetime64[ns]
```

Step 6. Set the date as the index

In [47]:
```
apple.set_index('Date', inplace=True)
print(apple.head())
```

```
             Open   High    Low  Close    Volume  Adj Close
Date
2014-07-08  96.27  96.80  93.92  95.35  65130000      95.35
2014-07-07  94.14  95.99  94.10  95.97  56305400      95.97
2014-07-03  93.67  94.10  93.20  94.03  22891800      94.03
2014-07-02  93.87  94.06  93.09  93.48  28420900      93.48
2014-07-01  93.52  94.07  93.13  93.52  38170200      93.52
```

In [48]:
```
apple.head()
```

Out[48]:

Date	Open	High	Low	Close	Volume	Adj Close
2014-07-08	96.27	96.80	93.92	95.35	65130000	95.35
2014-07-07	94.14	95.99	94.10	95.97	56305400	95.97
2014-07-03	93.67	94.10	93.20	94.03	22891800	94.03
2014-07-02	93.87	94.06	93.09	93.48	28420900	93.48
2014-07-01	93.52	94.07	93.13	93.52	38170200	93.52

Step 7. Is there any duplicate dates?

In [49]:
```
duplicate_dates = apple.index.duplicated().any()

if duplicate_dates:
    print("There are duplicate dates in the index.")
else:
    print("There are no duplicate dates in the index.")
```

```
There are no duplicate dates in the index.
```

Step 8. Ops...it seems the index is from the most recent date.

Make the first entry the oldest date.

In [50]:
```
apple.sort_index(ascending=True, inplace=True)
```

In [51]:
```
apple.head()
```

Out[51]:

Date	Open	High	Low	Close	Volume	Adj Close
1980-12-12	28.75	28.87	28.75	28.75	117258400	0.45
1980-12-15	27.38	27.38	27.25	27.25	43971200	0.42
1980-12-16	25.37	25.37	25.25	25.25	26432000	0.39
1980-12-17	25.87	26.00	25.87	25.87	21610400	0.40
1980-12-18	26.63	26.75	26.63	26.63	18362400	0.41

Step 9. Get the last business day of each month

In [52]:
```
last_business_days = apple.index.to_period('M').to_timestamp('M') + pd.offsets.BMonthEnd
```

In [53]:
```
last_business_days
```

Out[53]:
```
DatetimeIndex(['1981-01-30', '1981-01-30', '1981-01-30', '1981-01-30',
               '1981-01-30', '1981-01-30', '1981-01-30', '1981-01-30',
               '1981-01-30', '1981-01-30',
               ...
               '2014-07-31', '2014-07-31', '2014-07-31', '2014-07-31',
               '2014-07-31', '2014-08-29', '2014-08-29', '2014-08-29',
               '2014-08-29', '2014-08-29'],
              dtype='datetime64[ns]', name='Date', length=8465, freq=None)
```

Step 10. What is the difference in days between the first day and the oldest

In [54]:
```
first_date = apple.index.min()
oldest_date = apple.index.max()
difference_in_days = (oldest_date - first_date).days
```

In [55]:
```
difference_in_days
```

Out[55]:
```
12261
```

Step 11. How many months in the data we have?

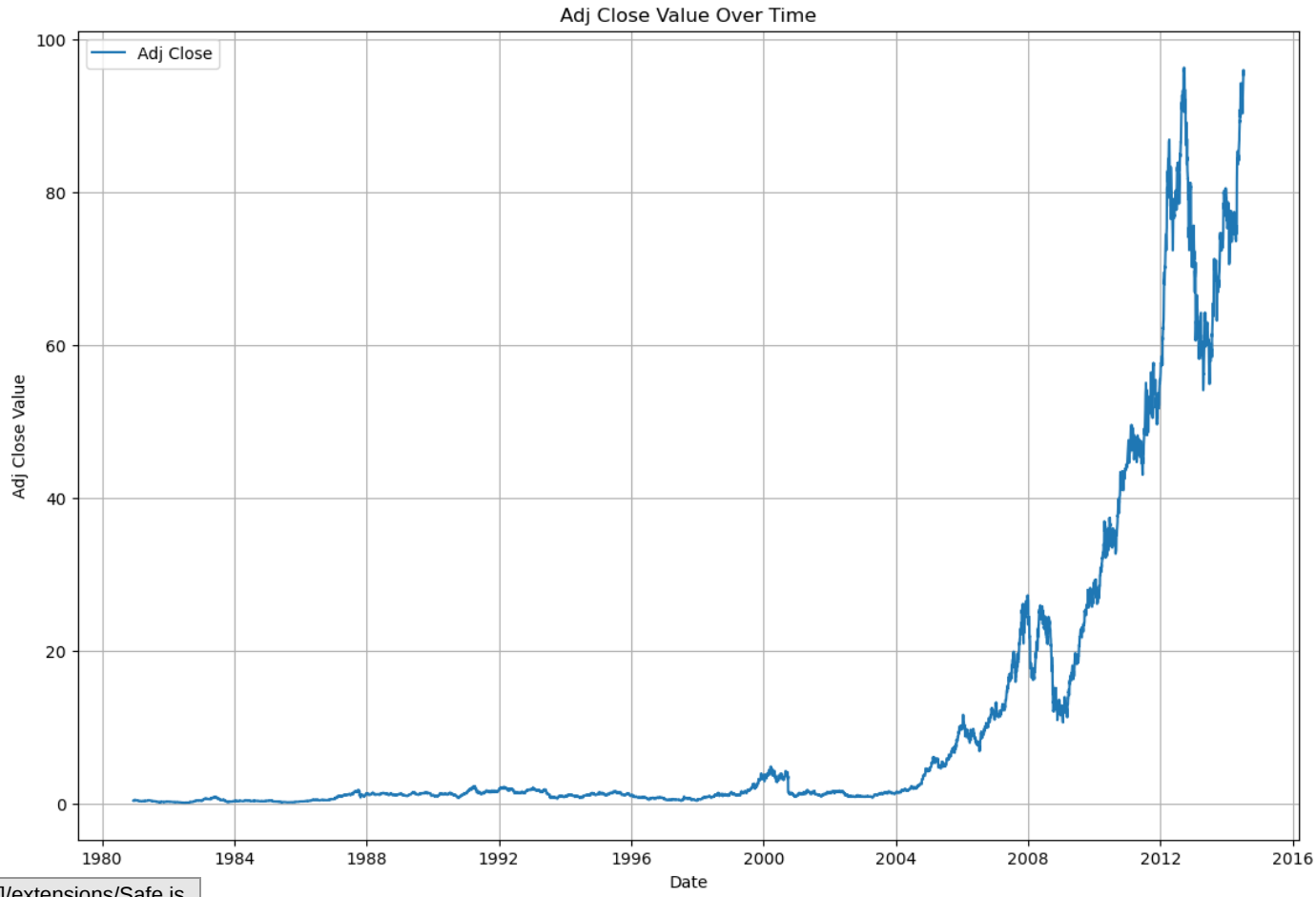
In [56]:
```
unique_months = apple.index.to_period('M').nunique()
```

In [57]:
```
unique_months
```

Out[57]:
```
404
```

Step 12. Plot the 'Adj Close' value. Set the size of the figure to 13.5 x 9 inches

In [58]:
```
plt.figure(figsize=(13.5, 9))

plt.plot(apple.index, apple['Adj Close'], label='Adj Close')
plt.xlabel('Date')
plt.ylabel('Adj Close Value')
plt.title('Adj Close Value Over Time')
plt.legend()
plt.grid(True)

plt.show()
```



Getting Financial Data - Pandas Datareader

Step 2. Create your time range (start and end variables). The start date should be 01/01/2015 and the end should today (whatever your today is).

In [59]:
```
start_date = datetime(2015, 1, 1)

end_date = date.today()
```

In [60]:
```
start_date
```

Out[60]:
```
datetime.datetime(2015, 1, 1, 0, 0)
```

In [61]:
```
end_date
```

Out[61]:
```
datetime.date(2023, 9, 23)
```

Investor - Flow of Funds - US

In [62]:
```
weekly = pd.read_csv("Downloads/weekly.txt")
```

In [63]:
```
weekly
```

Out[63]:

	Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
0	2012-12-05	-7426	-6060	-1367	-74	5317	4210	1107	-2183
1	2012-12-12	-8783	-7520	-1263	123	1818	1598	219	-6842
2	2012-12-19	-5496	-5470	-26	-73	103	3472	-3369	-5466
3	2012-12-26	-4451	-4076	-375	550	2610	3333	-722	-1291
4	2013-01-02	-11156	-9622	-1533	-158	2383	2103	280	-8931
5	2013-01-09	14817	7995	6821	2888	9766	7311	2455	27471
6	2014-04-02	3155	938	2217	265	3379	3129	250	6799
7	2014-04-09	5761	2080	3681	1482	1609	1448	161	8852
8	2014-04-16	2286	634	1652	1186	633	604	29	4105
9	2014-04-23	3530	1392	2138	1239	1984	1453	531	6753
10	2014-04-30	-3890	-3996	106	759	888	559	329	-2242
11	2014-05-07	632	-2006	2639	-340	5493	4417	1076	5785
12	2014-05-14	-1079	-2321	1242	1188	4037	3141	897	4146
13	2014-05-21	697	-1790	2487	1216	2196	1398	798	4109
14	2014-05-28	-2453	-2603	150	1108	2041	1236	805	696
15	2014-06-04	2098	-1148	3246	1123	188	-470	658	3409
16	2014-06-11	1236	-1840	3075	1159	2112	1587	524	4506
17	2014-06-18	-922	-2204	1282	1060	4159	3740	419	4297
18	2014-06-25	-93	-1354	1262	1246	3256	2694	562	4409
19	2014-07-02	-7835	-8887	1052	636	2979	2704	276	-4220
20	2014-07-09	666	-1070	1736	1006	2721	3203	-482	4393
21	2014-07-30	118	-1171	1290	1024	1806	1119	687	2949
22	2014-08-06	-471	-3073	2602	-375	-8193	-8658	465	-9040
23	2014-08-13	320	-974	1294	496	1436	539	897	2252
24	2014-08-20	2671	738	1933	821	4999	4185	814	8490

	Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
25	2014-08-27	-577	-2199	1623	943	3655	2921	734	4021
26	2014-09-03	-4024	-5305	1281	544	2430	1768	661	-1050
27	2014-09-10	1257	-1291	2548	1055	1554	711	843	3866
28	2014-11-05	-32	-1634	1602	-176	5813	5284	529	5604
29	2014-11-12	1464	61	1403	963	3596	2703	893	6023
30	2014-11-19	-3010	-3622	611	99	2529	1758	771	-383
31	2014-11-25	-1175	-2044	869	-157	2590	1821	769	1258
32	2015-01-07	-3913	-5438	1525	-1057	-3403	-4729	1326	-8373
33	2015-01-14	1774	-37	1811	248	3549	2582	967	5572
34	2015-01-21	1267	856	411	790	1258	220	1038	3315
35	2015-01-28	4343	3455	888	1748	5964	4689	1275	12055
36	2015-02-04	4240	3536	703	793	3237	2274	963	8270
37	2015-02-11	1268	-27	1296	959	5862	5169	693	8089
38	2015-03-04	999	-1933	2932	528	4984	4309	675	6511
39	2015-03-11	3911	-7	3918	851	1298	999	298	6059
40	2015-03-18	1948	-1758	3706	912	452	258	194	3312
41	2015-03-25	-1167	-4478	3311	538	2404	1701	703	1775
42	2015-04-01	-1527	-3307	1780	720	-1296	-1392	96	-2103
43	2015-04-08	1906	-1321	3227	250	1719	1906	-187	3875

Step 4. What is the frequency of the dataset?

In [64]:
```
weekly.value_counts()
```

Out[64]:

Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total	
2012-12-05	-7426	-6060	-1367	-74	5317	4210	1107	-2183	1
2012-12-12	-8783	-7520	-1263	123	1818	1598	219	-6842	1
2014-08-20	2671	738	1933	821	4999	4185	814	8490	1
2014-08-27	-577	-2199	1623	943	3655	2921	734	4021	1
2014-09-03	-4024	-5305	1281	544	2430	1768	661	-1050	1
2014-09-10	1257	-1291	2548	1055	1554	711	843	3866	1
2014-11-05	-32	-1634	1602	-176	5813	5284	529	5604	1
2014-11-12	1464	61	1403	963	3596	2703	893	6023	1
2014-11-19	-3010	-3622	611	99	2529	1758	771	-383	1
2014-11-25	-1175	-2044	869	-157	2590	1821	769	1258	1
2015-01-07	-3913	-5438	1525	-1057	-3403	-4729	1326	-8373	1
2015-01-14	1774	-37	1811	248	3549	2582	967	5572	1
2015-01-21	1267	856	411	790	1258	220	1038	3315	1
2015-01-28	4343	3455	888	1748	5964	4689	1275	12055	1
2015-02-04	4240	3536	703	793	3237	2274	963	8270	1
2015-02-11	1268	-27	1296	959	5862	5169	693	8089	1
2015-03-04	999	-1933	2932	528	4984	4309	675	6511	1
2015-03-11	3911	-7	3918	851	1298	999	298	6059	1
2015-03-18	1948	-1758	3706	912	452	258	194	3312	1
2015-03-25	-1167	-4478	3311	538	2404	1701	703	1775	1
2015-04-01	-1527	-3307	1780	720	-1296	-1392	96	-2103	1
2014-08-13	320	-974	1294	496	1436	539	897	2252	1
2014-08-06	-471	-3073	2602	-375	-8193	-8658	465	-9040	1
2014-07-30	118	-1171	1290	1024	1806	1119	687	2949	1
2014-04-30	-3890	-3996	106	759	888	559	329	-2242	1
2012-12-19	-5496	-5470	-26	-73	103	3472	-3369	-5466	1
2012-12-26	-4451	-4076	-375	550	2610	3333	-722	-1291	1
2013-01-02	-11156	-9622	-1533	-158	2383	2103	280	-8931	1
2013-01-09	14817	7995	6821	2888	9766	7311	2455	27471	1
2014-04-02	3155	938	2217	265	3379	3129	250	6799	1
2014-04-09	5761	2080	3681	1482	1609	1448	161	8852	1

```
2014-04-16   2286          634              1652          1186    633          604
29             4105     1
2014-04-23   3530          1392             2138          1239    1984         1453
531            6753     1
2014-05-07   632           -2006            2639         -340     5493         4417
1076           5785     1
2014-07-09   666           -1070            1736          1006    2721         3203
-482            4393     1
2014-05-14  -1079          -2321            1242          1188    4037         3141
897            4146     1
2014-05-21   697           -1790            2487          1216    2196         1398
798            4109     1
2014-05-28  -2453          -2603            150           1108    2041         1236
805            696      1
2014-06-04   2098          -1148            3246          1123    188         -470
658            3409     1
2014-06-11   1236          -1840            3075          1159    2112         1587
524            4506     1
2014-06-18  -922           -2204            1282          1060    4159         3740
419            4297     1
2014-06-25  -93            -1354            1262          1246    3256         2694
562            4409     1
2014-07-02  -7835          -8887            1052          636     2979         2704
276           -4220     1
2015-04-08   1906          -1321            3227          250     1719         1906
-187            3875     1
dtype: int64
```

In [65]:
```
weekly.describe()
```

Out[65]:

	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	
count	44.000000	44.000000	44.000000	44.000000	44.000000	44.000000	44.000000	44
mean	-161.727273	-1815.931818	1654.227273	684.227273	2452.613636	1931.977273	520.613636	2975
std	4318.401639	3223.717184	1533.770151	670.325983	2729.245012	2574.644536	786.337321	6228
min	-11156.000000	-9622.000000	-1533.000000	-1057.000000	-8193.000000	-8658.000000	-3369.000000	-9040
25%	-1758.500000	-3385.750000	883.250000	249.500000	1524.500000	1089.000000	279.000000	-549
50%	476.000000	-1774.000000	1563.500000	791.500000	2417.000000	1863.500000	668.000000	3948
75%	1916.500000	-22.000000	2561.500000	1072.000000	3610.750000	3235.500000	855.500000	5844
max	14817.000000	7995.000000	6821.000000	2888.000000	9766.000000	7311.000000	2455.000000	27471

Step 5. Set the column Date as the index.

In [ ]:
```

```

In [66]:
```
weekly
```

Out[66]:

	Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
0	2012-12-05	-7426	-6060	-1367	-74	5317	4210	1107	-2183
1	2012-12-12	-8783	-7520	-1263	123	1818	1598	219	-6842
2	2012-12-19	-5496	-5470	-26	-73	103	3472	-3369	-5466
3	2012-12-26	-4451	-4076	-375	550	2610	3333	-722	-1291
4	2013-01-02	-11156	-9622	-1533	-158	2383	2103	280	-8931
5	2013-01-09	14817	7995	6821	2888	9766	7311	2455	27471
6	2014-04-02	3155	938	2217	265	3379	3129	250	6799
7	2014-04-09	5761	2080	3681	1482	1609	1448	161	8852
8	2014-04-16	2286	634	1652	1186	633	604	29	4105
9	2014-04-23	3530	1392	2138	1239	1984	1453	531	6753
10	2014-04-30	-3890	-3996	106	759	888	559	329	-2242
11	2014-05-07	632	-2006	2639	-340	5493	4417	1076	5785
12	2014-05-14	-1079	-2321	1242	1188	4037	3141	897	4146
13	2014-05-21	697	-1790	2487	1216	2196	1398	798	4109
14	2014-05-28	-2453	-2603	150	1108	2041	1236	805	696
15	2014-06-04	2098	-1148	3246	1123	188	-470	658	3409
16	2014-06-11	1236	-1840	3075	1159	2112	1587	524	4506
17	2014-06-18	-922	-2204	1282	1060	4159	3740	419	4297
18	2014-06-25	-93	-1354	1262	1246	3256	2694	562	4409
19	2014-07-02	-7835	-8887	1052	636	2979	2704	276	-4220
20	2014-07-09	666	-1070	1736	1006	2721	3203	-482	4393
21	2014-07-30	118	-1171	1290	1024	1806	1119	687	2949
22	2014-08-06	-471	-3073	2602	-375	-8193	-8658	465	-9040
23	2014-08-13	320	-974	1294	496	1436	539	897	2252
24	2014-08-20	2671	738	1933	821	4999	4185	814	8490

	Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
25	2014-08-27	-577	-2199	1623	943	3655	2921	734	4021
26	2014-09-03	-4024	-5305	1281	544	2430	1768	661	-1050
27	2014-09-10	1257	-1291	2548	1055	1554	711	843	3866
28	2014-11-05	-32	-1634	1602	-176	5813	5284	529	5604
29	2014-11-12	1464	61	1403	963	3596	2703	893	6023
30	2014-11-19	-3010	-3622	611	99	2529	1758	771	-383
31	2014-11-25	-1175	-2044	869	-157	2590	1821	769	1258
32	2015-01-07	-3913	-5438	1525	-1057	-3403	-4729	1326	-8373
33	2015-01-14	1774	-37	1811	248	3549	2582	967	5572
34	2015-01-21	1267	856	411	790	1258	220	1038	3315
35	2015-01-28	4343	3455	888	1748	5964	4689	1275	12055
36	2015-02-04	4240	3536	703	793	3237	2274	963	8270
37	2015-02-11	1268	-27	1296	959	5862	5169	693	8089
38	2015-03-04	999	-1933	2932	528	4984	4309	675	6511
39	2015-03-11	3911	-7	3918	851	1298	999	298	6059
40	2015-03-18	1948	-1758	3706	912	452	258	194	3312
41	2015-03-25	-1167	-4478	3311	538	2404	1701	703	1775
42	2015-04-01	-1527	-3307	1780	720	-1296	-1392	96	-2103
43	2015-04-08	1906	-1321	3227	250	1719	1906	-187	3875

Step 6. What is the type of the index?

In [67]:
```
type(weekly.index)
```

Out[67]:
```
pandas.core.indexes.range.RangeIndex
```

In [68]:
```
weekly.index.dtype
```

Out[68]:
```
dtype('int64')
```

Step 8. Change the frequency to monthly, sum the values and assign it to monthly.

In [69]:
```
weekly['Date'] = pd.to_datetime(weekly['Date'])

weekly.set_index('Date', inplace=True)

monthly = weekly.resample('M').sum()
```

In [70]:
```
monthly
```

Out[70]:

Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
2012-12-31	-26156	-23126	-3031	526	9848	12613	-2765	-15782
2013-01-31	3661	-1627	5288	2730	12149	9414	2735	18540
2013-02-28	0	0	0	0	0	0	0	0
2013-03-31	0	0	0	0	0	0	0	0
2013-04-30	0	0	0	0	0	0	0	0
2013-05-31	0	0	0	0	0	0	0	0
2013-06-30	0	0	0	0	0	0	0	0
2013-07-31	0	0	0	0	0	0	0	0
2013-08-31	0	0	0	0	0	0	0	0
2013-09-30	0	0	0	0	0	0	0	0
2013-10-31	0	0	0	0	0	0	0	0
2013-11-30	0	0	0	0	0	0	0	0
2013-12-31	0	0	0	0	0	0	0	0
2014-01-31	0	0	0	0	0	0	0	0
2014-02-28	0	0	0	0	0	0	0	0
2014-03-31	0	0	0	0	0	0	0	0
2014-04-30	10842	1048	9794	4931	8493	7193	1300	24267
2014-05-31	-2203	-8720	6518	3172	13767	10192	3576	14736
2014-06-30	2319	-6546	8865	4588	9715	7551	2163	16621
2014-07-31	-7051	-11128	4078	2666	7506	7026	481	3122
2014-08-31	1943	-5508	7452	1885	1897	-1013	2910	5723
2014-09-30	-2767	-6596	3829	1599	3984	2479	1504	2816
2014-10-31	0	0	0	0	0	0	0	0
2014-11-30	-2753	-7239	4485	729	14528	11566	2962	12502

Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
2014-12-31	0	0	0	0	0	0	0	0
2015-01-31	3471	-1164	4635	1729	7368	2762	4606	12569
2015-02-28	5508	3509	1999	1752	9099	7443	1656	16359
2015-03-31	5691	-8176	13867	2829	9138	7267	1870	17657
2015-04-30	379	-4628	5007	970	423	514	-91	1772

Step 9. You will notice that it filled the dataFrame with months that don't have any data with NaN. Let's drop these rows.

In [71]:
```
monthly.dropna(inplace=True)
```

In [72]:
```
monthly
```

Out[72]:

Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
2012-12-31	-26156	-23126	-3031	526	9848	12613	-2765	-15782
2013-01-31	3661	-1627	5288	2730	12149	9414	2735	18540
2013-02-28	0	0	0	0	0	0	0	0
2013-03-31	0	0	0	0	0	0	0	0
2013-04-30	0	0	0	0	0	0	0	0
2013-05-31	0	0	0	0	0	0	0	0
2013-06-30	0	0	0	0	0	0	0	0
2013-07-31	0	0	0	0	0	0	0	0
2013-08-31	0	0	0	0	0	0	0	0
2013-09-30	0	0	0	0	0	0	0	0
2013-10-31	0	0	0	0	0	0	0	0
2013-11-30	0	0	0	0	0	0	0	0
2013-12-31	0	0	0	0	0	0	0	0
2014-01-31	0	0	0	0	0	0	0	0
2014-02-28	0	0	0	0	0	0	0	0
2014-03-31	0	0	0	0	0	0	0	0
2014-04-30	10842	1048	9794	4931	8493	7193	1300	24267
2014-05-31	-2203	-8720	6518	3172	13767	10192	3576	14736
2014-06-30	2319	-6546	8865	4588	9715	7551	2163	16621
2014-07-31	-7051	-11128	4078	2666	7506	7026	481	3122
2014-08-31	1943	-5508	7452	1885	1897	-1013	2910	5723
2014-09-30	-2767	-6596	3829	1599	3984	2479	1504	2816
2014-10-31	0	0	0	0	0	0	0	0
2014-11-30	-2753	-7239	4485	729	14528	11566	2962	12502

Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
2014-12-31	0	0	0	0	0	0	0	0
2015-01-31	3471	-1164	4635	1729	7368	2762	4606	12569
2015-02-28	5508	3509	1999	1752	9099	7443	1656	16359
2015-03-31	5691	-8176	13867	2829	9138	7267	1870	17657
2015-04-30	379	-4628	5007	970	423	514	-91	1772

In [73]:
```
weekly[~(weekly == 0).all(axis=1)]
```

Out[73]:

Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
2012-12-05	-7426	-6060	-1367	-74	5317	4210	1107	-2183
2012-12-12	-8783	-7520	-1263	123	1818	1598	219	-6842
2012-12-19	-5496	-5470	-26	-73	103	3472	-3369	-5466
2012-12-26	-4451	-4076	-375	550	2610	3333	-722	-1291
2013-01-02	-11156	-9622	-1533	-158	2383	2103	280	-8931
2013-01-09	14817	7995	6821	2888	9766	7311	2455	27471
2014-04-02	3155	938	2217	265	3379	3129	250	6799
2014-04-09	5761	2080	3681	1482	1609	1448	161	8852
2014-04-16	2286	634	1652	1186	633	604	29	4105
2014-04-23	3530	1392	2138	1239	1984	1453	531	6753
2014-04-30	-3890	-3996	106	759	888	559	329	-2242
2014-05-07	632	-2006	2639	-340	5493	4417	1076	5785
2014-05-14	-1079	-2321	1242	1188	4037	3141	897	4146
2014-05-21	697	-1790	2487	1216	2196	1398	798	4109
2014-05-28	-2453	-2603	150	1108	2041	1236	805	696
2014-06-04	2098	-1148	3246	1123	188	-470	658	3409
2014-06-11	1236	-1840	3075	1159	2112	1587	524	4506
2014-06-18	-922	-2204	1282	1060	4159	3740	419	4297
2014-06-25	-93	-1354	1262	1246	3256	2694	562	4409
2014-07-02	-7835	-8887	1052	636	2979	2704	276	-4220
2014-07-09	666	-1070	1736	1006	2721	3203	-482	4393
2014-07-30	118	-1171	1290	1024	1806	1119	687	2949
2014-08-06	-471	-3073	2602	-375	-8193	-8658	465	-9040
2014-08-13	320	-974	1294	496	1436	539	897	2252

Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
2014-08-20	2671	738	1933	821	4999	4185	814	8490
2014-08-27	-577	-2199	1623	943	3655	2921	734	4021
2014-09-03	-4024	-5305	1281	544	2430	1768	661	-1050
2014-09-10	1257	-1291	2548	1055	1554	711	843	3866
2014-11-05	-32	-1634	1602	-176	5813	5284	529	5604
2014-11-12	1464	61	1403	963	3596	2703	893	6023
2014-11-19	-3010	-3622	611	99	2529	1758	771	-383
2014-11-25	-1175	-2044	869	-157	2590	1821	769	1258
2015-01-07	-3913	-5438	1525	-1057	-3403	-4729	1326	-8373
2015-01-14	1774	-37	1811	248	3549	2582	967	5572
2015-01-21	1267	856	411	790	1258	220	1038	3315
2015-01-28	4343	3455	888	1748	5964	4689	1275	12055
2015-02-04	4240	3536	703	793	3237	2274	963	8270
2015-02-11	1268	-27	1296	959	5862	5169	693	8089
2015-03-04	999	-1933	2932	528	4984	4309	675	6511
2015-03-11	3911	-7	3918	851	1298	999	298	6059
2015-03-18	1948	-1758	3706	912	452	258	194	3312
2015-03-25	-1167	-4478	3311	538	2404	1701	703	1775
2015-04-01	-1527	-3307	1780	720	-1296	-1392	96	-2103
2015-04-08	1906	-1321	3227	250	1719	1906	-187	3875

Step 10. Good, now we have the monthly data. Now change the frequency to year.

In [74]:
```
yearly = monthly.resample('Y').sum()
```

In [75]:
```
yearly
```

Out[75]:

Date	Total Equity	Domestic Equity	World Equity	Hybrid	Total Bond	Taxable Bond	Municipal Bond	Total
2012-12-31	-26156	-23126	-3031	526	9848	12613	-2765	-15782
2013-12-31	3661	-1627	5288	2730	12149	9414	2735	18540
2014-12-31	330	-44689	45021	19570	59890	44994	14896	79787
2015-12-31	15049	-10459	25508	7280	26028	17986	8041	48357

Iris

In [76]:
```
iris = pd.read_csv("Downloads/iris.data")
```

In [77]:
```
iris
```

Out[77]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

Step 4. Create columns for the dataset

In [78]:
```
iris.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
```

In [79]:
```
iris
```

Out[79]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

Step 5. Is there any missing value in the dataframe?

In [80]:
```
iris.isnull().sum()
```

Out[80]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
class           0
dtype: int64
```

Step 6. Lets set the values of the rows 10 to 29 of the column 'petal_length' to NaN

In [81]:
```
iris.loc[10:29, 'petal_length'] = np.nan
```

In [82]:
```
iris
```

Out[82]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

Step 7. Good, now lets substitute the NaN values to 1.0

In [83]:
```
iris['petal_length'].fillna(1.0, inplace=True)
```

In [84]:
```
iris
```

Out[84]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

Step 8. Now let's delete the column class

In [85]:
```
iris.drop(columns=['class'], inplace=True)
```

In [86]:
```
iris
```

Out[86]:

	sepal_length	sepal_width	petal_length	petal_width
0	4.9	3.0	1.4	0.2
1	4.7	3.2	1.3	0.2
2	4.6	3.1	1.5	0.2
3	5.0	3.6	1.4	0.2
4	5.4	3.9	1.7	0.4
...	...	...	...	...
144	6.7	3.0	5.2	2.3
145	6.3	2.5	5.0	1.9
146	6.5	3.0	5.2	2.0
147	6.2	3.4	5.4	2.3
148	5.9	3.0	5.1	1.8

149 rows × 4 columns

Step 9. Set the first 3 rows as NaN

In [87]:
```
iris.loc[0:2] = np.nan
```

In [88]:
```
iris
```

Out[88]:

	sepal_length	sepal_width	petal_length	petal_width
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	5.0	3.6	1.4	0.2
4	5.4	3.9	1.7	0.4
...	...	...	...	...
144	6.7	3.0	5.2	2.3
145	6.3	2.5	5.0	1.9
146	6.5	3.0	5.2	2.0
147	6.2	3.4	5.4	2.3
148	5.9	3.0	5.1	1.8

149 rows × 4 columns

Step 10. Delete the rows that have NaN

In [89]:
```
iris.dropna(inplace=True)
```

In [90]:
```
iris
```

Out[90]:

	sepal_length	sepal_width	petal_length	petal_width
3	5.0	3.6	1.4	0.2
4	5.4	3.9	1.7	0.4
5	4.6	3.4	1.4	0.3
6	5.0	3.4	1.5	0.2
7	4.4	2.9	1.4	0.2
...	...	...	...	...
144	6.7	3.0	5.2	2.3
145	6.3	2.5	5.0	1.9
146	6.5	3.0	5.2	2.0
147	6.2	3.4	5.4	2.3
148	5.9	3.0	5.1	1.8

146 rows × 4 columns

Step 11: Reset the index so it begins with 0 again

In [91]:
```
iris.reset_index(drop=True, inplace=True)
```

In [92]:
```
iris
```

Out[92]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.0	3.6	1.4	0.2
1	5.4	3.9	1.7	0.4
2	4.6	3.4	1.4	0.3
3	5.0	3.4	1.5	0.2
4	4.4	2.9	1.4	0.2
...	...	...	...	...
141	6.7	3.0	5.2	2.3
142	6.3	2.5	5.0	1.9
143	6.5	3.0	5.2	2.0
144	6.2	3.4	5.4	2.3
145	5.9	3.0	5.1	1.8

146 rows × 4 columns

Wine

In [93]:
```
wine = pd.read_csv("Downloads/wine.data")
```

In [94]:
```
wine
```

Out[94]:

	1	14.23	1.71	2.43	15.6	127	2.8	3.06	.28	2.29	5.64	1.04	3.92	1065
0	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
1	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
2	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
3	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
4	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
172	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740
173	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750
174	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835
175	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840
176	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560

177 rows × 14 columns

Step 4. Delete the first, fourth, seventh, nineth, eleventh, thirteenth and fourteenth columns

In [95]:
```
columns_to_delete = [0, 3, 6, 8, 10, 12, 13]
wine.drop(wine.columns[columns_to_delete], axis=1, inplace=True)
```

In [96]:
```
wine
```

Out[96]:

	14.23	1.71	15.6	127	3.06	2.29	1.04
0	13.20	1.78	11.2	100	2.76	1.28	1.05
1	13.16	2.36	18.6	101	3.24	2.81	1.03
2	14.37	1.95	16.8	113	3.49	2.18	0.86
3	13.24	2.59	21.0	118	2.69	1.82	1.04
4	14.20	1.76	15.2	112	3.39	1.97	1.05
...	...	...	...	...	...	...	...
172	13.71	5.65	20.5	95	0.61	1.06	0.64
173	13.40	3.91	23.0	102	0.75	1.41	0.70
174	13.27	4.28	20.0	120	0.69	1.35	0.59
175	13.17	2.59	20.0	120	0.68	1.46	0.60
176	14.13	4.10	24.5	96	0.76	1.35	0.61

177 rows × 7 columns

Step 5. Assign the columns as below: The attributes are (donated by Riccardo Leardi, riclea '@' anchem.unige.it):

1) alcohol 2) malic_acid 3) alcalinity_of_ash 4) magnesium 5) flavanoids 6) proanthocyanins 7) hue

In [97]:
```
wine.columns = ['alcohol','malic_acid','alcalinity_of_ash','magnesium','flavanoids','pro
```

In [98]:
```
wine
```

Out[98]:

	alcohol	malic_acid	alcalinity_of_ash	magnesium	flavanoids	proanthocyanins	hue
0	13.20	1.78	11.2	100	2.76	1.28	1.05
1	13.16	2.36	18.6	101	3.24	2.81	1.03
2	14.37	1.95	16.8	113	3.49	2.18	0.86
3	13.24	2.59	21.0	118	2.69	1.82	1.04
4	14.20	1.76	15.2	112	3.39	1.97	1.05
...	...	...	...	...	...	...	...
172	13.71	5.65	20.5	95	0.61	1.06	0.64
173	13.40	3.91	23.0	102	0.75	1.41	0.70
174	13.27	4.28	20.0	120	0.69	1.35	0.59
175	13.17	2.59	20.0	120	0.68	1.46	0.60
176	14.13	4.10	24.5	96	0.76	1.35	0.61

177 rows × 7 columns

Step 6. Set the values of the first 3 rows from alcohol as NaN

In [104…
```
wine.loc[0:2, 'alcohol'] = np.nan
```

In [105…
```
wine
```

Out[105]:

	alcohol	malic_acid	alcalinity_of_ash	magnesium	flavanoids	proanthocyanins	hue
0	NaN	1.78	11.2	100	2.76	1.28	1.05
1	NaN	2.36	18.6	101	3.24	2.81	1.03
2	NaN	1.95	16.8	113	3.49	2.18	0.86
3	13.24	2.59	21.0	118	2.69	1.82	1.04
4	14.20	1.76	15.2	112	3.39	1.97	1.05
...	...	...	...	...	...	...	...
172	13.71	5.65	20.5	95	0.61	1.06	0.64
173	13.40	3.91	23.0	102	0.75	1.41	0.70
174	13.27	4.28	20.0	120	0.69	1.35	0.59
175	13.17	2.59	20.0	120	0.68	1.46	0.60
176	14.13	4.10	24.5	96	0.76	1.35	0.61

177 rows × 7 columns

Step 7. Now set the value of the rows 3 and 4 of magnesium as NaN

In [106…

```
wine.loc[2:3, 'magnesium'] = np.nan
```

In [107…

```
wine
```

Out[107]:

	alcohol	malic_acid	alcalinity_of_ash	magnesium	flavanoids	proanthocyanins	hue
0	NaN	1.78	11.2	100.0	2.76	1.28	1.05
1	NaN	2.36	18.6	101.0	3.24	2.81	1.03
2	NaN	1.95	16.8	NaN	3.49	2.18	0.86
3	13.24	2.59	21.0	NaN	2.69	1.82	1.04
4	14.20	1.76	15.2	112.0	3.39	1.97	1.05
...	...	...	...	...	...	...	...
172	13.71	5.65	20.5	95.0	0.61	1.06	0.64
173	13.40	3.91	23.0	102.0	0.75	1.41	0.70
174	13.27	4.28	20.0	120.0	0.69	1.35	0.59
175	13.17	2.59	20.0	120.0	0.68	1.46	0.60
176	14.13	4.10	24.5	96.0	0.76	1.35	0.61

177 rows × 7 columns

Step 8. Fill the value of NaN with the number 10 in alcohol and 100 in magnesium

In [113…

```
wine['alcohol'].fillna(10, inplace=True)
wine['magnesium'].fillna(100, inplace=True)
```

In [114…

```
wine
```

Out[114]:

	alcohol	malic_acid	alcalinity_of_ash	magnesium	flavanoids	proanthocyanins	hue
0	10.00	1.78	11.2	100.0	2.76	1.28	1.05
1	10.00	2.36	18.6	101.0	3.24	2.81	1.03
2	10.00	1.95	16.8	100.0	3.49	2.18	0.86
3	13.24	2.59	21.0	100.0	2.69	1.82	1.04
4	14.20	1.76	15.2	112.0	3.39	1.97	1.05
...	...	...	...	...	...	...	...
172	13.71	5.65	20.5	95.0	0.61	1.06	0.64
173	13.40	3.91	23.0	102.0	0.75	1.41	0.70
174	13.27	4.28	20.0	120.0	0.69	1.35	0.59
175	13.17	2.59	20.0	120.0	0.68	1.46	0.60
176	14.13	4.10	24.5	96.0	0.76	1.35	0.61

177 rows × 7 columns

In [116…
```
wine.iloc[10]
```

Out[116]:
```
alcohol              14.12
malic_acid            1.48
alcalinity_of_ash    16.80
magnesium            95.00
flavanoids            2.43
proanthocyanins       1.57
hue                   1.17
Name: 10, dtype: float64
```

Step 9: Count the number of missing values

In [117…
```
wine.isnull().sum().sum()
```

Out[117]:
```
0
```

Step 10: Create an array of 10 random numbers up until 10

In [119…
```
random_numbers = np.random.randint(0, 10, 10)
```

In [125…
```
random_numbers
```

Out[125]:
```
array([6, 0, 7, 7, 8, 9, 6, 9, 2, 4])
```

Step 11: Use random numbers as an index and assign NaN value to each cell

In [126…
```
wine.iloc[random_numbers, :] = np.nan
```

In [127…
```
wine
```

Out[127]:

	alcohol	malic_acid	alcalinity_of_ash	magnesium	flavanoids	proanthocyanins	hue
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	10.00	2.36	18.6	101.0	3.24	2.81	1.03
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	13.24	2.59	21.0	100.0	2.69	1.82	1.04
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...
172	13.71	5.65	20.5	95.0	0.61	1.06	0.64
173	13.40	3.91	23.0	102.0	0.75	1.41	0.70
174	13.27	4.28	20.0	120.0	0.69	1.35	0.59
175	13.17	2.59	20.0	120.0	0.68	1.46	0.60
176	14.13	4.10	24.5	96.0	0.76	1.35	0.61

177 rows × 7 columns

Step 12. How many missing values do we have?

In [128…
```
wine.isnull().sum().sum()
```

Out[128]:
```
49
```

Step 13: Delete rows that contain missing values

In [129…
```
wine.dropna(inplace=True)
```

In [130…
```
wine.isnull().sum().sum()
```

Out[130]:
```
0
```

Step 14: Print only the non-null values in 'alcohol'

In [131…
```
print(wine['alcohol'].dropna())
```

```
1      10.00
3      13.24
5      14.39
10     14.12
11     13.75
       ...
172    13.71
173    13.40
174    13.27
175    13.17
176    14.13
Name: alcohol, Length: 170, dtype: float64
```

Step 15: Reset the index

In [132…
```
wine.reset_index(drop=True, inplace=True)
```

In [133…
```
wine
```

Out[133]:

	alcohol	malic_acid	alcalinity_of_ash	magnesium	flavanoids	proanthocyanins	hue
0	10.00	2.36	18.6	101.0	3.24	2.81	1.03
1	13.24	2.59	21.0	100.0	2.69	1.82	1.04
2	14.39	1.87	14.6	96.0	2.52	1.98	1.02
3	14.12	1.48	16.8	95.0	2.43	1.57	1.17
4	13.75	1.73	16.0	89.0	2.76	1.81	1.15
...	...	...	...	...	...	...	...
165	13.71	5.65	20.5	95.0	0.61	1.06	0.64
166	13.40	3.91	23.0	102.0	0.75	1.41	0.70
167	13.27	4.28	20.0	120.0	0.69	1.35	0.59
168	13.17	2.59	20.0	120.0	0.68	1.46	0.60
169	14.13	4.10	24.5	96.0	0.76	1.35	0.61

170 rows × 7 columns

In [ ]: