In [1]:
```
import pandas as pd


meteorites = pd.read_csv('Meteorite_Landings.csv',nrows=5) # pd.read_csv is a comma
meteorites
```

Out[1]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	G
0	Aachen	1	Valid	L5	21	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333	
1	Aarhus	2	Valid	H6	720	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333	
2	Abee	6	Valid	EH4	107000	Fell	01/01/1952 12:00:00 AM	54.21667	-113.00000	
3	Acapulco	10	Valid	Acapulcoite	1914	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000	
4	Achiras	370	Valid	L6	780	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000	

In [2]:
```
meteorites.name # it shows the entries under the column named "name"
```

Out[2]:
```
0      Aachen
1      Aarhus
2        Abee
3    Acapulco
4     Achiras
Name: name, dtype: object
```

In [3]:
```
meteorites.columns # this shows the names of the columns in the dataframe
```

Out[3]:
```
Index(['name', 'id', 'nametype', 'recclass', 'mass (g)', 'fall', 'year',
       'reclat', 'reclong', 'GeoLocation'],
      dtype='object')
```

In [4]:
```
meteorites.index #this shows the number of indexes in the dataframe
```

Out[4]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [5]:
```
import requests #requests is the library that gets the data using API

response = requests.get('https://data.nasa.gov/resource/gh4g-9sfh.json',params=({'$

if response.ok:
```

```
    payload = response.json()
else:
    print(f'Request was not succesful and returned code: {response.status_code}.')
    payload = None
```

In [6]:
```
payload[1] #the payload result in in list, so when calling an entry use list type c
```

Out[6]:
```
{'name': 'Aarhus',
 'id': '2',
 'nametype': 'Valid',
 'recclass': 'H6',
 'mass': '720',
 'fall': 'Fell',
 'year': '1951-01-01T00:00:00.000',
 'reclat': '56.183330',
 'reclong': '10.233330',
 'geolocation': {'latitude': '56.18333', 'longitude': '10.23333'}}
```

In [7]:
```
import pandas as pd
df = pd.DataFrame(payload) # the list is then placed as a dataframe
df.head(3)
```

Out[7]:

	name	id	nametype	recclass	mass	fall	year	reclat	reclong	g
0	Aachen	1	Valid	L5	21	Fell	1880-01-01T00:00:00.000	50.775000	6.083330	
1	Aarhus	2	Valid	H6	720	Fell	1951-01-01T00:00:00.000	56.183330	10.233330	
2	Abee	6	Valid	EH4	107000	Fell	1952-01-01T00:00:00.000	54.216670	-113.000000	

In [8]:
```
import pandas as pd


meteorites = pd.read_csv('Meteorite_Landings.csv') # pd.read_csv is a command to re
meteorites


meteorites.shape # this shows the rows and columns of the dataframe
```

Out[8]:
```
(45716, 10)
```

In [9]:
```
meteorites.columns # this shows the names of the columns in the dataframe
```

Out[9]:
```
Index(['name', 'id', 'nametype', 'recclass', 'mass (g)', 'fall', 'year',
       'reclat', 'reclong', 'GeoLocation'],
      dtype='object')
```

In [10]:
```
meteorites.dtypes #this shows the data types that each column contain
```

Out[10]:
```
name            object
id               int64
nametype        object
recclass        object
mass (g)       float64
fall            object
year            object
reclat         float64
reclong        float64
GeoLocation     object
dtype: object
```

In [11]:
```
meteorites.head(10) # the head() command shows the first 5 rows in the dataframe by
```

Out[11]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667	-113.00000
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000
5	Adhi Kot	379	Valid	EH4	4239.0	Fell	01/01/1919 12:00:00 AM	32.10000	71.80000
6	Adzhi-Bogdo (stone)	390	Valid	LL3-6	910.0	Fell	01/01/1949 12:00:00 AM	44.83333	95.16667
7	Agen	392	Valid	H5	30000.0	Fell	01/01/1814 12:00:00 AM	44.21667	0.61667
8	Aguada	398	Valid	L6	1620.0	Fell	01/01/1930 12:00:00 AM	-31.60000	-65.23333
9	Aguila Blanca	417	Valid	L	1440.0	Fell	01/01/1920 12:00:00 AM	-30.86667	-64.55000

In [12]:
```
meteorites.tail() # the tail() command shows the last 5 rows by default, can place
```

Out[12]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	r
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990 12:00:00 AM	29.03700	17
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999 12:00:00 AM	13.78333	8
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939 12:00:00 AM	49.25000	17
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003 12:00:00 AM	49.78917	41
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976 12:00:00 AM	33.98333	-115

In [13]:
```
meteorites.info()
# this shows the information about the dataframe the column names, it shows non-nul
#if there are misssing entries in the dataframe, and the data types of the of each
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45716 entries, 0 to 45715
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   name         45716 non-null  object 
 1   id           45716 non-null  int64  
 2   nametype     45716 non-null  object 
 3   recclass     45716 non-null  object 
 4   mass (g)     45585 non-null  float64
 5   fall         45716 non-null  object 
 6   year         45425 non-null  object 
 7   reclat       38401 non-null  float64
 8   reclong      38401 non-null  float64
 9   GeoLocation  38401 non-null  object 
dtypes: float64(3), int64(1), object(6)
memory usage: 3.5+ MB
```

In [14]:
```
#meteorites["name","year"]
#the comment above will result in an error because in accessing columns it must be

meteorites[["name","year"]] # this is the correct command
```

Out[14]:

	name	year
0	Aachen	01/01/1880 12:00:00 AM
1	Aarhus	01/01/1951 12:00:00 AM
2	Abee	01/01/1952 12:00:00 AM
3	Acapulco	01/01/1976 12:00:00 AM
4	Achiras	01/01/1902 12:00:00 AM
...	...	...
45711	Zillah 002	01/01/1990 12:00:00 AM
45712	Zinder	01/01/1999 12:00:00 AM
45713	Zlin	01/01/1939 12:00:00 AM
45714	Zubkovsky	01/01/2003 12:00:00 AM
45715	Zulu Queen	01/01/1976 12:00:00 AM

45716 rows × 2 columns

In [15]:
```
meteorites[100:104] #this outputs the index with the specific number used until the
#example [100:104] it first output the 100th index and it outputs the 103rd index d
```

Out[15]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclon
100	Benton	5026	Valid	LL6	2840.0	Fell	01/01/1949 12:00:00 AM	45.95000	-67.5500
101	Berduc	48975	Valid	L6	270.0	Fell	01/01/2008 12:00:00 AM	-31.91000	-58.3283
102	Béréba	5028	Valid	Eucrite-mmict	18000.0	Fell	01/01/1924 12:00:00 AM	11.65000	-3.6500
103	Berlanguillas	5029	Valid	L6	1440.0	Fell	01/01/1811 12:00:00 AM	41.68333	-3.8000

In [16]:
```
meteorites.iloc[[0,3,4,6]] #this outputs the 0, 3, 4, 6 columns using iloc
```

Out[16]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	Ge
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333	
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000	
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000	
6	Adzhi-Bogdo (stone)	390	Valid	LL3-6	910.0	Fell	01/01/1949 12:00:00 AM	44.83333	95.16667	

In [17]:
```
meteorites.iloc[100:104, [0,3,4,6]] #this shows the 100th to 100rd row as explain a
```

Out[17]:

	name	recclass	mass (g)	year
100	Benton	LL6	2840.0	01/01/1949 12:00:00 AM
101	Berduc	L6	270.0	01/01/2008 12:00:00 AM
102	Béréba	Eucrite-mmict	18000.0	01/01/1924 12:00:00 AM
103	Berlanguillas	L6	1440.0	01/01/1811 12:00:00 AM

In [18]:
```
meteorites.loc[[0,3,4,6]]# meteorites.iloc[[0,3,4,6]] # works the same as iloc
```

Out[18]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	Ge
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333	
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000	
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000	
6	Adzhi-Bogdo (stone)	390	Valid	LL3-6	910.0	Fell	01/01/1949 12:00:00 AM	44.83333	95.16667	

In [19]:
```
meteorites.iloc[-1, -1] # iloc is used to take from the index given [-1,-1] shows t
```

Out[19]: '(33.98333, -115.68333)'

In [20]:
```
meteorites[(meteorites['mass (g)']> 50) & (meteorites.fall == 'Found')] #this shows
```

Out[20]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	
37	Northwest Africa 5815	50693	Valid	L5	256.80	Found	NaN	0.00000	
757	Dominion Range 03239	32591	Valid	L6	69.50	Found	01/01/2002 12:00:00 AM	NaN	
804	Dominion Range 03240	32592	Valid	LL5	290.90	Found	01/01/2002 12:00:00 AM	NaN	
1111	Abajo	4	Valid	H5	331.00	Found	01/01/1982 12:00:00 AM	26.80000	-1
1112	Abar al' Uj 001	51399	Valid	H3.8	194.34	Found	01/01/2008 12:00:00 AM	22.72192	
...	...	...	...	...	...	...	...	...	
45709	Zhongxiang	30406	Valid	Iron	100000.00	Found	01/01/1981 12:00:00 AM	31.20000	1
45710	Zillah 001	31355	Valid	L6	1475.00	Found	01/01/1990 12:00:00 AM	29.03700	
45711	Zillah 002	31356	Valid	Eucrite	172.00	Found	01/01/1990 12:00:00 AM	29.03700	
45714	Zubkovsky	31357	Valid	L6	2167.00	Found	01/01/2003 12:00:00 AM	49.78917	
45715	Zulu Queen	30414	Valid	L3.7	200.00	Found	01/01/1976 12:00:00 AM	33.98333	-1

18854 rows × 10 columns

In [21]:
```
meteorites[(meteorites['mass (g)']> 1e6) & (meteorites.fall == 'Fell')] #this shows
```

Out[21]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclon
29	Allende	2278	Valid	CV3	2000000.0	Fell	01/01/1969 12:00:00 AM	26.96667	-105.3166
419	Jilin	12171	Valid	H5	4000000.0	Fell	01/01/1976 12:00:00 AM	44.05000	126.1666
506	Kunya-Urgench	12379	Valid	H5	1100000.0	Fell	01/01/1998 12:00:00 AM	42.25000	59.2000
707	Norton County	17922	Valid	Aubrite	1100000.0	Fell	01/01/1948 12:00:00 AM	39.68333	-99.8666
920	Sikhote-Alin	23593	Valid	Iron, IIAB	23000000.0	Fell	01/01/1947 12:00:00 AM	46.16000	134.6533

In [22]:
```
meteorites.query("`mass (g)`> 1e6 and fall == 'Fell'") # the query is
```

Out[22]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclon
29	Allende	2278	Valid	CV3	2000000.0	Fell	01/01/1969 12:00:00 AM	26.96667	-105.3166
419	Jilin	12171	Valid	H5	4000000.0	Fell	01/01/1976 12:00:00 AM	44.05000	126.1666
506	Kunya-Urgench	12379	Valid	H5	1100000.0	Fell	01/01/1998 12:00:00 AM	42.25000	59.2000
707	Norton County	17922	Valid	Aubrite	1100000.0	Fell	01/01/1948 12:00:00 AM	39.68333	-99.8666
920	Sikhote-Alin	23593	Valid	Iron, IIAB	23000000.0	Fell	01/01/1947 12:00:00 AM	46.16000	134.6533

In [23]:
```
meteorites.fall.value_counts() # this command returns the entries in the column gro
```

Out[23]:
```
fall
Found     44609
Fell       1107
Name: count, dtype: int64
```

In [24]:
```
meteorites.value_counts(subset=['nametype','fall'], normalize=False)
```

Out[24]:
```
nametype  fall
Valid     Found    44534
          Fell      1107
Relict    Found       75
Name: count, dtype: int64
```

In [25]:
```
type(meteorites['mass (g)'].mean()) #this takes the mean of the whole column entrie
```

Out[25]:
```
numpy.float64
```

In [26]:
```
meteorites['mass (g)'].quantile([0.01,0.05,0.5,0.95,0.99]) #this shows the quantile
```

Out[26]:
```
0.01        0.44
0.05        1.10
0.50       32.60
0.95     4000.00
0.99    50600.00
Name: mass (g), dtype: float64
```

In [27]:
```
meteorites['mass (g)'].median() #this shows the median, or the 50th percentile
```

Out[27]:
```
32.6
```

In [28]:
```
meteorites['mass (g)'].max() #this shows the maximum entry in the "mass (g)" column
```

Out[28]:
```
60000000.0
```

In [29]:
```
meteorites.loc[meteorites['mass (g)'].idxmax()] # this shows the row where the maxi
```

Out[29]:
```
name                          Hoba
id                           11890
nametype                     Valid
recclass                 Iron, IVB
mass (g)                60000000.0
fall                         Found
year           01/01/1920 12:00:00 AM
reclat                    -19.58333
reclong                    17.91667
GeoLocation     (-19.58333, 17.91667)
Name: 16392, dtype: object
```

In [30]:
```
meteorites.recclass.nunique() #this shows the number of values that are not unique
```

Out[30]:
```
466
```

In [31]:
```
meteorites.name.nunique() #this shows the number of values that are not unique
```

Out[31]:
```
45716
```

In [32]:
```
meteorites.recclass.unique()  #this shows the number of values that are  unique
```

Out[32]:
```
array(['L5', 'H6', 'EH4', 'Acapulcoite', 'L6', 'LL3-6', 'H5', 'L',
       'Diogenite-pm', 'Unknown', 'H4', 'H', 'Iron, IVA', 'CR2-an', 'LL5',
       'CI1', 'L/LL4', 'Eucrite-mmict', 'CV3', 'Ureilite-an',
       'Stone-uncl', 'L3', 'Angrite', 'LL6', 'L4', 'Aubrite',
       'Iron, IIAB', 'Iron, IAB-sLL', 'Iron, ungrouped', 'CM2', 'OC',
       'Mesosiderite-A1', 'LL4', 'C2-ung', 'LL3.8', 'Howardite',
       'Eucrite-pmict', 'Diogenite', 'LL3.15', 'LL3.9', 'Iron, IAB-MG',
       'H/L3.9', 'Iron?', 'Eucrite', 'H4-an', 'L/LL6', 'Iron, IIIAB',
       'H/L4', 'H4-5', 'L3.7', 'LL3.4', 'Martian (chassignite)', 'EL6',
       'H3.8', 'H3-5', 'H5-6', 'Mesosiderite', 'H5-7', 'L3-6', 'H4-6',
       'Ureilite', 'Iron, IID', 'Mesosiderite-A3/4', 'CO3.3', 'H3',
       'EH3/4-an', 'Iron, IIE', 'L/LL5', 'H3.7', 'CBa', 'H4/5', 'H3/4',
       'H?', 'H3-6', 'L3.4', 'Iron, IAB-sHL', 'L3.7-6', 'EH7-an', 'Iron',
       'CR2', 'CO3.2', 'K3', 'L5/6', 'CK4', 'Iron, IIE-an', 'L3.6',
       'LL3.2', 'Pallasite', 'CO3.5', 'Lodranite', 'Mesosiderite-A3',
       'L3-4', 'H5/6', 'Pallasite, PMG', 'Eucrite-cm', 'L5-6', 'CO3.6',
       'Martian (nakhlite)', 'LL3.6', 'C3-ung', 'H3-4', 'CO3.4', 'EH3',
       'Iron, IAB-ung', 'Winonaite', 'LL', 'Eucrite-br', 'Iron, IIF',
       'R3.8-6', 'L4-6', 'EH5', 'LL3.00', 'H3.4', 'Martian (shergottite)',
       'Achondrite-ung', 'LL3.3', 'C', 'H/L3.6', 'Iron, IIIAB-an', 'LL7',
       'Mesosiderite-B2', 'LL4-6', 'CO3.7', 'L/LL6-an',
       'Iron, IAB complex', 'Pallasite, PMG-an', 'H3.9/4', 'L3.8',
       'LL5-6', 'LL3.8-6', 'L3.9', 'L4-5', 'L3-5', 'LL4/5', 'L4/5',
       'H3.9', 'H3.6-6', 'H3.8-5', 'H3.8/4', 'H3.9-5', 'CH3', 'R3.8-5',
       'L3.9/4', 'E4', 'CO3', 'Chondrite-ung', 'H~5', 'H~6', 'L/LL3.10',
       'EL5', 'LL3', 'L~6', 'L~3', 'H~4', 'L(LL)3.5-3.7', 'Iron, IIIE-an',
       'H3.6', 'L3.4-3.7', 'L3.5', 'CM1/2', 'Martian (OPX)', 'Brachinite',
       'LL7(?)', 'LL6(?)', 'Eucrite-Mg rich', 'H3.5-4', 'EL3', 'R3.6',
       'H3.5', 'CM1', 'L/LL3', 'H7', 'L(?)3', 'L3.2', 'L3.7-3.9',
       'Mesosiderite-B1', 'Eucrite-unbr', 'LL3.7', 'CO3.0', 'LL3.5',
       'L3.7-4', 'CV3-an', 'Lunar (anorth)', 'L3.3', 'Iron, IAB-sLM',
       'Lunar', 'Iron, IC', 'Iron, IID-an', 'Iron, IIIE', 'Iron, IVA-an',
       'CK6', 'L3.1', 'CK5', 'H3.3', 'H3.7-6', 'E6', 'H3.0', 'H3.1',
       'L3.0', 'L/LL3.4', 'C6', 'LL3.0', 'Lunar (gabbro)', 'R4', 'C4',
       'Iron, IIG', 'Iron, IIC', 'C1-ung', 'H5-an', 'EH4/5', 'Iron, IIIF',
       'R3-6', 'Mesosiderite-B4', 'L6/7', 'Relict H', 'L-imp melt', 'CK3',
       'H3-an', 'Iron, IVB', 'R3.8', 'L~5', 'Mesosiderite-an',
       'Mesosiderite-A2', 'Pallasite, PES', 'C4-ung', 'Iron, IAB?',
       'Mesosiderite-A', 'R3.5-6', 'H3.9-6', 'Ureilite-pmict', 'LL~6',
       'CK4/5', 'EL4', 'Lunar (feldsp. breccia)', 'L3.9-6', 'H-an',
       'L/LL3-6', 'L/LL3-5', 'H/L3.5', 'H/L3', 'R3-4', 'CK3-an', 'LL4-5',
       'H/L6', 'L3/4', 'H-imp melt', 'CR', 'Chondrite-fusion crust',
       'Iron, IAB-sLH', 'H(L)3-an', 'L(LL)3', 'H(L)3', 'R3', 'L7',
       'CM-an', 'L/LL~6', 'L/LL~5', 'L~4', 'L/LL~4', 'LL(L)3', 'H3.2',
       'L-melt breccia', 'H6-melt breccia', 'H5-melt breccia',
       'H-melt rock', 'Eucrite-an', 'Lunar (bas/anor)', 'LL5/6', 'LL3/4',
       'H3.4/3.5', 'Lunar (basalt)', 'H/L5', 'H(5?)', 'LL-imp melt',
       'Mesosiderite?', 'H~4/5', 'L6-melt breccia', 'L3.5-3.7',
       'Iron, IIAB-an', 'L3.3-3.7', 'L3.2-3.6', 'L3.3-3.6',
       'Acapulcoite/Lodranite', 'Mesosiderite-B', 'CK5/6', 'L3.05', 'C2',
       'C4/5', 'L/LL3.2', 'Iron, IIIAB?', 'L3.5-5', 'L/LL(?)3', 'H4(?)',
       'Iron, IAB-sHH', 'Relict iron', 'EL4/5', 'L5-7', 'Diogenite-an',
       'L-melt rock', 'CR1', 'H5 ', 'L5 ', 'H4 ', 'L4 ', 'E', 'L6 ',
       'H3 ', 'LL6 ', 'H-metal', 'H6 ', 'L-metal', 'Relict OC', 'EH',
       'Mesosiderite-A4', 'L/LL5/6', 'H3.8-4', 'CBb', 'EL6/7', 'EL7',
       'CH/CBb', 'CO3.8', 'H/L~4', 'Mesosiderite-C2', 'R5', 'H4/6',
```

```
'H3.7-5', 'LL3.7-6', 'H3.7/3.8', 'L3.7/3.8', 'EH-imp melt', 'R',
'Fusion crust', 'Aubrite-an', 'R6', 'LL-melt rock', 'L3.5-3.9',
'L3.2-3.5', 'L3.3-3.5', 'L3.0-3.7', 'E3-an', 'K', 'E3',
'Acapulcoite/lodranite', 'CK4-an', 'L(LL)3.05', 'L3.10', 'CB',
'Diogenite-olivine', 'EL-melt rock', 'EH6', 'Pallasite, ungrouped',
'L/LL4/5', 'L3.8-an', 'Iron, IAB-an', 'C5/6-ung', 'CV2',
'Iron, IC-an', 'Lunar (bas. breccia)', 'L3.8-6', 'R3/4', 'R3.9',
'CK', 'LL3.10', 'R4/5', 'L3.8-5', 'Mesosiderite-C', 'Enst achon',
'H/L3-4', 'L(H)3', 'LL6/7', 'LL3.1', 'OC3', 'R3.7', 'CO3 ', 'CH3 ',
'LL~4', 'LL~4/5', 'L(LL)~4', 'H3.05', 'H3.10',
'Impact melt breccia', 'LL3-5', 'H/L3.7', 'LL3-4', 'CK3/4',
'Martian', 'CO3.1', 'Lunar (bas/gab brec)', 'Achondrite-prim',
'LL<3.5', 'CK3.8', 'L/LL-melt rock', 'H6/7', 'EL6 ',
'Iron, IAB-sHL-an', 'CM2-an', 'R3-5', 'L4-melt rock',
'L6-melt rock', 'H/L4/5', 'EL3/4', 'H/L6-melt rock',
'Enst achon-ung', 'L3-7', 'R3.4', 'LL3.05', 'LL4/6', 'LL3.8-4',
'H3.15', 'C3.0-ung', 'LL-melt breccia', 'LL6-melt breccia',
'L5-melt breccia', 'LL(L)3.1', 'LL6-an', 'L4-melt breccia',
'Howardite-an', 'H4-melt breccia', 'Martian (basaltic breccia)',
'L3-melt breccia', 'L~4-6', 'LL~5', 'R3.5-4', 'CR7',
'H-melt breccia', 'Lunar (norite)', 'L3.00', 'H3.0-3.4', 'L/LL4-6',
'CM', 'EH7', 'L4-an', 'E-an', 'H3.8/3.9', 'L3.9-5', 'H3.8-6',
'H3.4-5', 'L3.0-3.9', 'L3.5-3.8', 'H3.2-3.7', 'L3.6-4',
'Iron, IIE?', 'C3/4-ung', 'L/LL3.5', 'L/LL3.6/3.7', 'H/L4-5',
'LL~3', 'Pallasite?', 'LL5-7', 'LL3.9/4', 'H3.8-an', 'CR-an',
'L/LL5-6', 'L(LL)5', 'L(LL)6', 'LL3.1-3.5', 'E5', 'Lodranite-an',
'H3.2-6', 'H(?)4', 'E5-an', 'H3.2-an', 'EH6-an', 'Stone-ung',
'C1/2-ung', 'L/LL'], dtype=object)
```

In [33]:
```
meteorites.describe()
```

Out[33]:

	id	mass (g)	reclat	reclong
count	45716.000000	4.558500e+04	38401.000000	38401.000000
mean	26889.735104	1.327808e+04	-39.122580	61.074319
std	16860.683030	5.749889e+05	46.378511	80.647298
min	1.000000	0.000000e+00	-87.366670	-165.433330
25%	12688.750000	7.200000e+00	-76.714240	0.000000
50%	24261.500000	3.260000e+01	-71.500000	35.666670
75%	40656.750000	2.026000e+02	0.000000	157.166670
max	57458.000000	6.000000e+07	81.166670	354.473330

In [34]:
```
meteorites.describe(include='all')
```

Out[34]:

	name	id	nametype	recclass	mass (g)	fall	year	
count	45716	45716.000000	45716	45716	4.558500e+04	45716	45425	3840
unique	45716	NaN	2	466	NaN	2	266	
top	Aachen	NaN	Valid	L6	NaN	Found	01/01/2003 12:00:00 AM	
freq	1	NaN	45641	8285	NaN	44609	3323	
mean	NaN	26889.735104	NaN	NaN	1.327808e+04	NaN	NaN	-3
std	NaN	16860.683030	NaN	NaN	5.749889e+05	NaN	NaN	4
min	NaN	1.000000	NaN	NaN	0.000000e+00	NaN	NaN	-8
25%	NaN	12688.750000	NaN	NaN	7.200000e+00	NaN	NaN	-7
50%	NaN	24261.500000	NaN	NaN	3.260000e+01	NaN	NaN	-7
75%	NaN	40656.750000	NaN	NaN	2.026000e+02	NaN	NaN	
max	NaN	57458.000000	NaN	NaN	6.000000e+07	NaN	NaN	8

In [ ]:

Excercise (Part 1)

Seatwork 6.2 Programming Exercise: Getting Started with Pandas!

""" Using the 2019_Yellow_Taxi_Trip_Data.csv dataset, accomplish the following items and submit a PDF of the notebook:

```
Create a DataFrame by reading in the 2019_Yellow_Taxi_Trip_Data.csv
file. Examine the first 5 rows.
Find the dimensions (number of rows and number of columns) in the
data.
Using the data in the 2019_Yellow_Taxi_Trip_Data.csv file,
calculate summary statistics for the fare_amount, tip_amount,
tolls_amount, and total_amount columns.
Isolate the fare_amount, tip_amount, tolls_amount, and total_amount
for the longest trip by distance (trip_distance).
```

"""

In [35]:
```
import pandas as pd

#1     Create a DataFrame by reading in the 2019_Yellow_Taxi_Trip_Data.csv file. Ex
```

```
#the pd.read_csv is a command to read the csv file, the dataframe is then stored in
df = pd.read_csv('2019_Yellow_Taxi_Trip_Data.csv')
df.head()
```

Out[35]:

	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance
0	2	2019-10-23T16:39:42.000	2019-10-23T17:14:10.000	1	7.93
1	1	2019-10-23T16:32:08.000	2019-10-23T16:45:26.000	1	2.00
2	2	2019-10-23T16:08:44.000	2019-10-23T16:21:11.000	1	1.36
3	2	2019-10-23T16:22:44.000	2019-10-23T16:43:26.000	1	1.00
4	2	2019-10-23T16:45:11.000	2019-10-23T16:58:49.000	1	1.96

In [36]:
```
#2     Find the dimensions (number of rows and number of columns) in the data.

#using the shape command this outputs the number of rows and columns in the datafra
a,x = df.shape
print(f'The number of rows is {a}')
print(f'The number of columns is {x}')
```

```
The number of rows is 10000
The number of columns is 18
```

In [37]:
```
#3
#Using the data in the 2019_Yellow_Taxi_Trip_Data.csv file, calculate summary stati
#tip_amount, tolls_amount, and total_amount columns.

# the 'fare_amount', 'tip_amount', 'tolls_amount' is in double brackets to output t
#and the describe() command is then used to describe those columns with details
df[['fare_amount', 'tip_amount', 'tolls_amount','total_amount']].describe()
```

Out[37]:

	fare_amount	tip_amount	tolls_amount	total_amount
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	15.106313	2.634494	0.623447	22.564659
std	13.954762	3.409800	6.437507	19.209255
min	-52.000000	0.000000	-6.120000	-65.920000
25%	7.000000	0.000000	0.000000	12.375000
50%	10.000000	2.000000	0.000000	16.300000
75%	16.000000	3.250000	0.000000	22.880000
max	176.000000	43.000000	612.000000	671.800000

In [38]:
```
#4
#Isolate the fare_amount, tip_amount, tolls_amount, and total_amount for the longes


#this command finds row with the longest trip distance using loc and idxmax and usi
#the "[['fare_amount', 'tip_amount', 'tolls_amount', 'total_amount']]" is then used
#to output the fare_amount, tip_amount, tolls_amount, and total_amount of the row w
df.loc[df['trip_distance'].idxmax()][['fare_amount', 'tip_amount', 'tolls_amount',
```

Out[38]:
```
fare_amount      176.0
tip_amount       18.29
tolls_amount      6.12
total_amount    201.21
Name: 8338, dtype: object
```

Conclusion / Reflection

```
After doing this activity, I learned many ways to visualize data,
by using the appropriate commands to output the output necessary,
there are many functions
```

to be used in making the specific outputs, other than taking data from your own directory in a device, you can also take data from the internet, this is done by using the "requests" library, but this activity is more focused on the "pandas" library visualizing a dataframe. The some syntax is quite easy to follow the data statistics is quite hard because I don't quite understand the syntax being used. I still need more time to practice those syntax to do a better visualization of the data.

Data wrangling

In [39]:
```
import pandas as pd
taxis = pd.read_csv('2019_Yellow_Taxi_Trip_Data.csv') # load the csv file using pan
taxis.head()
```

Out[39]:

	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance
0	2	2019-10-23T16:39:42.000	2019-10-23T17:14:10.000	1	7.93
1	1	2019-10-23T16:32:08.000	2019-10-23T16:45:26.000	1	2.00
2	2	2019-10-23T16:08:44.000	2019-10-23T16:21:11.000	1	1.36
3	2	2019-10-23T16:22:44.000	2019-10-23T16:43:26.000	1	1.00
4	2	2019-10-23T16:45:11.000	2019-10-23T16:58:49.000	1	1.96

In [40]:
```
#this code store the columns that ends with "id" and has the word "store_and_fwd_fl
mask = taxis.columns.str.contains('id$|store_and_fwd_flag', regex = True)
columns_to_drop = taxis.columns[mask]
columns_to_drop
```

Out[40]:
```
Index(['vendorid', 'ratecodeid', 'store_and_fwd_flag', 'pulocationid',
       'dolocationid'],
      dtype='object')
```

In [41]:
```
#this part removes the columns that contain the stored columns in the code above
taxis=taxis.drop(columns = columns_to_drop)
taxis.head()
```

Out[41]:

	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	payment_t
0	2019-10-23T16:39:42.000	2019-10-23T17:14:10.000	1	7.93	
1	2019-10-23T16:32:08.000	2019-10-23T16:45:26.000	1	2.00	
2	2019-10-23T16:08:44.000	2019-10-23T16:21:11.000	1	1.36	
3	2019-10-23T16:22:44.000	2019-10-23T16:43:26.000	1	1.00	
4	2019-10-23T16:45:11.000	2019-10-23T16:58:49.000	1	1.96	

In [42]:
```
taxis = taxis.rename(columns={'tpep_pickup_datetime':'pickup','tpep_dropoff_datetim
taxis.columns
#renames the column "tpep_pickup_datetime" to "pickup" and the column "tpep_dropoff
```

Out[42]:
```
Index(['pickup', 'dropoff', 'passenger_count', 'trip_distance', 'payment_type',
       'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount',
       'improvement_surcharge', 'total_amount', 'congestion_surcharge'],
      dtype='object')
```

In [43]:
```
taxis.dtypes #this outputs the datatypes of each column
```

Out[43]:
```
pickup                  object
dropoff                 object
passenger_count          int64
trip_distance          float64
payment_type             int64
fare_amount            float64
extra                  float64
mta_tax                float64
tip_amount             float64
tolls_amount           float64
improvement_surcharge  float64
total_amount           float64
congestion_surcharge   float64
dtype: object
```

In [44]:
```
taxis[['pickup','dropoff']] =  taxis[['pickup','dropoff']].apply(pd.to_datetime)
taxis.dtypes
#this rewrites the dataframe with the changes pickup and dropoff datatype
```

Out[44]:
```
pickup                  datetime64[ns]
dropoff                 datetime64[ns]
passenger_count                  int64
trip_distance                  float64
payment_type                     int64
fare_amount                    float64
extra                          float64
mta_tax                        float64
tip_amount                     float64
tolls_amount                   float64
improvement_surcharge          float64
total_amount                   float64
congestion_surcharge           float64
dtype: object
```

Creating new columns

In [45]:
```
taxis['elapsed_time'] = taxis['dropoff'] - taxis['pickup']#making a new column in t
```

In [46]:
```
taxis['elapsed_time']
```

Out[46]:
```
0       0 days 00:34:28
1       0 days 00:13:18
2       0 days 00:12:27
3       0 days 00:20:42
4       0 days 00:13:38
              ...
9995    0 days 00:09:27
9996    0 days 00:07:43
9997    0 days 00:04:19
9998    0 days 00:11:02
9999    0 days 00:29:55
Name: elapsed_time, Length: 10000, dtype: timedelta64[ns]
```

In [47]:
```
taxis = taxis.assign(cost_before_tip=lambda x: x.total_amount - x.tip_amount,
                     tip_pct = lambda x: x.tip_amount/x.cost_before_tip,
                     fees=lambda x: x.cost_before_tip - x.fare_amount,
                     avg_speed = lambda x: x.trip_distance.div(x.elapsed_time.dt.to
taxis.head()

#this is another way of making new columns
```

Out[47]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra	m
0	2019-10-23 16:39:42	2019-10-23 17:14:10	1	7.93	1	29.5	1.0	
1	2019-10-23 16:32:08	2019-10-23 16:45:26	1	2.00	1	10.5	1.0	
2	2019-10-23 16:08:44	2019-10-23 16:21:11	1	1.36	1	9.5	1.0	
3	2019-10-23 16:22:44	2019-10-23 16:43:26	1	1.00	1	13.0	1.0	
4	2019-10-23 16:45:11	2019-10-23 16:58:49	1	1.96	1	10.5	1.0	

In [48]:
```
taxis.sort_values(['passenger_count', 'pickup'], ascending=[False,True]).head() # t
```

Out[48]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra
5997	2019-10-23 15:55:19	2019-10-23 16:08:25	6	1.58	2	10.0	1.0
443	2019-10-23 15:56:59	2019-10-23 16:04:33	6	1.46	2	7.5	1.0
8722	2019-10-23 15:57:33	2019-10-23 16:03:34	6	0.62	1	5.5	1.0
4198	2019-10-23 15:57:38	2019-10-23 16:05:07	6	1.18	1	7.0	1.0
8238	2019-10-23 15:58:31	2019-10-23 16:29:29	6	3.23	2	19.5	1.0

In [49]:
```
taxis.nlargest(3,'elapsed_time') #this code finds the 3 largest elapsed_time
```

Out[49]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra
7576	2019-10-23 16:52:51	2019-10-24 16:51:44	1	3.75	1	17.5	1.0
6902	2019-10-23 16:51:42	2019-10-24 16:50:22	1	11.19	2	39.5	1.0
4975	2019-10-23 16:18:51	2019-10-24 16:17:30	1	0.70	2	7.0	1.0

In [50]:
```
taxis.nsmallest(3,'elapsed_time') #this does the same but the mallest ones
```

Out[50]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra
729	2019-10-23 16:52:27	2019-10-23 16:52:27	1	0.0	2	52.0	4.5
3912	2019-10-23 16:21:26	2019-10-23 16:21:26	1	0.0	2	8.0	3.5
4716	2019-10-23 16:07:59	2019-10-23 16:07:59	1	0.0	2	80.0	0.0

Exercise (Part 2)

In [51]:
```
meteorite =pd.read_csv("Meteorite_Landings.csv")
meteorite # this reads the csv file
```

Out[51]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667
...	...	...	...	...	...	...	...	...
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990 12:00:00 AM	29.03700
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999 12:00:00 AM	13.78333
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939 12:00:00 AM	49.25000
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003 12:00:00 AM	49.78917
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976 12:00:00 AM	33.98333

45716 rows × 10 columns

In [54]:
```
meteorite = meteorite.rename(columns = {'mass (g)':"mass"})
meteorite
#.rename command renames the column
```

Out[54]:

	name	id	nametype	recclass	mass	fall	year	reclat
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667
...	...	...	...	...	...	...	...	...
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990 12:00:00 AM	29.03700
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999 12:00:00 AM	13.78333
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939 12:00:00 AM	49.25000
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003 12:00:00 AM	49.78917
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976 12:00:00 AM	33.98333

45716 rows × 10 columns

In [63]:

```
coordinates = meteorite.columns.str.contains('lat|long', regex = True)
to_drop = meteorite.columns[coordinates]
meteorite = meteorite.drop(columns = to_drop)
meteorite
#this drops the latitude and longitude
```

Out[63]:

	name	id	nametype	recclass	mass	fall	year
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM
...	...	...	...	...	...	...	...
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990 12:00:00 AM
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999 12:00:00 AM
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939 12:00:00 AM
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003 12:00:00 AM
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976 12:00:00 AM

45716 rows × 7 columns

In [64]:
```
meteorite.sort_values(['mass'], ascending=[False])# sorts the mass column descendin
```

Out[64]:

	name	id	nametype	recclass	mass	fall	year
16392	Hoba	11890	Valid	Iron, IVB	60000000.0	Found	01/01/1920 12:00:00 AM
5373	Cape York	5262	Valid	Iron, IIIAB	58200000.0	Found	01/01/1818 12:00:00 AM
5365	Campo del Cielo	5247	Valid	Iron, IAB-MG	50000000.0	Found	12/22/1575 12:00:00 AM
5370	Canyon Diablo	5257	Valid	Iron, IAB-MG	30000000.0	Found	01/01/1891 12:00:00 AM
3455	Armanty	2335	Valid	Iron, IIIE	28000000.0	Found	01/01/1898 12:00:00 AM
...	...	...	...	...	...	...	...
38282	Wei-hui-fu (a)	24231	Valid	Iron	NaN	Found	01/01/1931 12:00:00 AM
38283	Wei-hui-fu (b)	24232	Valid	Iron	NaN	Found	01/01/1931 12:00:00 AM
38285	Weiyuan	24233	Valid	Mesosiderite	NaN	Found	01/01/1978 12:00:00 AM
41472	Yamato 792768	28117	Valid	CM2	NaN	Found	01/01/1979 12:00:00 AM
45698	Zapata County	30393	Valid	Iron	NaN	Found	01/01/1930 12:00:00 AM

45716 rows × 7 columns

In [ ]: