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


meteorites = pd.read_csv('Meteorite_Landings.csv',nrows=5) # pd.read_csv is a command to read the csv to a dataframe
meteorites
```

Out[1]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
0	Aachen	1	Valid	L5	21	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333	(50.775, 6.08333)
1	Aarhus	2	Valid	H6	720	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333	(56.18333, 10.23333)
2	Abee	6	Valid	EH4	107000	Fell	01/01/1952 12:00:00 AM	54.21667	-113.00000	(54.21667, -113.0)
3	Acapulco	10	Valid	Acapulcoite	1914	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000	(16.88333, -99.9)
4	Achiras	370	Valid	L6	780	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000	(-33.16667, -64.95)

In [2]:
```python
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]:
```python
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]:
```python
meteorites.index #this shows the number of indexes in the dataframe
```

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

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

response = requests.get('https://data.nasa.gov/resource/gh4g-9sfh.json',params=({'$limit':50_000})) #code to get the
```

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

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

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]:
```python
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	geolocation	:@computed_region_cbhk_
0	Aachen	1	Valid	L5	21	Fell	1880-01-01T00:00:00.000	50.775000	6.083330	{'latitude': '50.775', 'longitude': '6.08333'}	
1	Aarhus	2	Valid	H6	720	Fell	1951-01-01T00:00:00.000	56.183330	10.233330	{'latitude': '56.18333', 'longitude': '10.23333'}	
2	Abee	6	Valid	EH4	107000	Fell	1952-01-01T00:00:00.000	54.216670	-113.000000	{'latitude': '54.21667', 'longitude': '-113.0'}	

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


meteorites = pd.read_csv('Meteorite_Landings.csv') # pd.read_csv is a command to read the csv to a dataframe
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 default, can place inside the par
```

Out[11]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333	(50.775, 6.08333)
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333	(56.18333, 10.23333)
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667	-113.00000	(54.21667, -113.0)
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000	(16.88333, -99.9)
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000	(-33.16667, -64.95)
5	Adhi Kot	379	Valid	EH4	4239.0	Fell	01/01/1919 12:00:00 AM	32.10000	71.80000	(32.1, 71.8)
6	Adzhi-Bogdo (stone)	390	Valid	LL3-6	910.0	Fell	01/01/1949 12:00:00 AM	44.83333	95.16667	(44.83333, 95.16667)
7	Agen	392	Valid	H5	30000.0	Fell	01/01/1814 12:00:00 AM	44.21667	0.61667	(44.21667, 0.61667)
8	Aguada	398	Valid	L6	1620.0	Fell	01/01/1930 12:00:00 AM	-31.60000	-65.23333	(-31.6, -65.23333)
9	Aguila Blanca	417	Valid	L	1440.0	Fell	01/01/1920 12:00:00 AM	-30.86667	-64.55000	(-30.86667, -64.55)

In [12]:

```
meteorites.tail() # the tail() command shows the last 5 rows by default, can place inside the parenthesis to output s
```

Out[12]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990 12:00:00 AM	29.03700	17.01850	(29.037, 17.0185)
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999 12:00:00 AM	13.78333	8.96667	(13.78333, 8.96667)
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939 12:00:00 AM	49.25000	17.66667	(49.25, 17.66667)
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003 12:00:00 AM	49.78917	41.50460	(49.78917, 41.5046)
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976 12:00:00 AM	33.98333	-115.68333	(33.98333, -115.68333)

In [13]:

```
meteorites.info()
# this shows the information about the dataframe the column names, it shows non-null count
#if there are misssing entries in the dataframe, and the data types of the of each columns
```

```
<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 [17]:

```
#meteorites["name","year"]
#the comment above will result in an error because in accessing columns it must be in list so double brakets must be
```

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

Out[17]:

	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 [18]:

```python
meteorites[100:104] #this outputs the index with the specific number used until the 2nd number minus 1
#example [100:104] it first output the 100th index and it outputs the 103rd index due to 104 - 1 = 103
```

Out[18]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
100	Benton	5026	Valid	LL6	2840.0	Fell	01/01/1949 12:00:00 AM	45.95000	-67.55000	(45.95, -67.55)
101	Berduc	48975	Valid	L6	270.0	Fell	01/01/2008 12:00:00 AM	-31.91000	-58.32833	(-31.91, -58.32833)
102	Béréba	5028	Valid	Eucrite-mmict	18000.0	Fell	01/01/1924 12:00:00 AM	11.65000	-3.65000	(11.65, -3.65)
103	Berlanguillas	5029	Valid	L6	1440.0	Fell	01/01/1811 12:00:00 AM	41.68333	-3.80000	(41.68333, -3.8)

In [20]:

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

Out[20]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333	(50.775, 6.08333)
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000	(16.88333, -99.9)
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000	(-33.16667, -64.95)
6	Adzhi-Bogdo (stone)	390	Valid	LL3-6	910.0	Fell	01/01/1949 12:00:00 AM	44.83333	95.16667	(44.83333, 95.16667)

In [24]:

```
meteorites.iloc[100:104, [0,3,4,6]] #this shows the 100th to 100rd row as explain above, and the 0, 3, 4, 6 columns u
```

Out[24]:

	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 [21]:

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

Out[21]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333	(50.775, 6.08333)
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000	(16.88333, -99.9)
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000	(-33.16667, -64.95)
6	Adzhi-Bogdo (stone)	390	Valid	LL3-6	910.0	Fell	01/01/1949 12:00:00 AM	44.83333	95.16667	(44.83333, 95.16667)

In [23]:

```
meteorites.iloc[-1, -1] # iloc is used to take from the index given [-1,-1] shows the last row (last index is -1) and
```

Out[23]:

```
'(33.98333, -115.68333)'
```

In [31]:

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

Out[31]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
37	Northwest Africa 5815	50693	Valid	L5	256.80	Found	NaN	0.00000	0.00000	(0.0, 0.0)
757	Dominion Range 03239	32591	Valid	L6	69.50	Found	01/01/2002 12:00:00 AM	NaN	NaN	NaN
804	Dominion Range 03240	32592	Valid	LL5	290.90	Found	01/01/2002 12:00:00 AM	NaN	NaN	NaN
1111	Abajo	4	Valid	H5	331.00	Found	01/01/1982 12:00:00 AM	26.80000	-105.41667	(26.8, -105.41667)
1112	Abar al' Uj 001	51399	Valid	H3.8	194.34	Found	01/01/2008 12:00:00 AM	22.72192	48.95937	(22.72192, 48.95937)
...	...	...	...	...	...	...	...	...	...	...
45709	Zhongxiang	30406	Valid	Iron	100000.00	Found	01/01/1981 12:00:00 AM	31.20000	112.50000	(31.2, 112.5)
45710	Zillah 001	31355	Valid	L6	1475.00	Found	01/01/1990 12:00:00 AM	29.03700	17.01850	(29.037, 17.0185)
45711	Zillah 002	31356	Valid	Eucrite	172.00	Found	01/01/1990 12:00:00 AM	29.03700	17.01850	(29.037, 17.0185)
45714	Zubkovsky	31357	Valid	L6	2167.00	Found	01/01/2003 12:00:00 AM	49.78917	41.50460	(49.78917, 41.5046)
45715	Zulu Queen	30414	Valid	L3.7	200.00	Found	01/01/1976 12:00:00 AM	33.98333	-115.68333	(33.98333, -115.68333)

18854 rows × 10 columns

In [30]:

```
meteorites[(meteorites['mass (g)']> 1e6) & (meteorites.fall == 'Fell')] #this shows parts of the dataframe filtered c
```

Out[30]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
29	Allende	2278	Valid	CV3	2000000.0	Fell	01/01/1969 12:00:00 AM	26.96667	-105.31667	(26.96667, -105.31667)
419	Jilin	12171	Valid	H5	4000000.0	Fell	01/01/1976 12:00:00 AM	44.05000	126.16667	(44.05, 126.16667)
506	Kunya-Urgench	12379	Valid	H5	1100000.0	Fell	01/01/1998 12:00:00 AM	42.25000	59.20000	(42.25, 59.2)
707	Norton County	17922	Valid	Aubrite	1100000.0	Fell	01/01/1948 12:00:00 AM	39.68333	-99.86667	(39.68333, -99.86667)
920	Sikhote-Alin	23593	Valid	Iron, IIAB	23000000.0	Fell	01/01/1947 12:00:00 AM	46.16000	134.65333	(46.16, 134.65333)

In [33]:

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

Out[33]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
29	Allende	2278	Valid	CV3	2000000.0	Fell	01/01/1969 12:00:00 AM	26.96667	-105.31667	(26.96667, -105.31667)
419	Jilin	12171	Valid	H5	4000000.0	Fell	01/01/1976 12:00:00 AM	44.05000	126.16667	(44.05, 126.16667)
506	Kunya-Urgench	12379	Valid	H5	1100000.0	Fell	01/01/1998 12:00:00 AM	42.25000	59.20000	(42.25, 59.2)
707	Norton County	17922	Valid	Aubrite	1100000.0	Fell	01/01/1948 12:00:00 AM	39.68333	-99.86667	(39.68333, -99.86667)
920	Sikhote-Alin	23593	Valid	Iron, IIAB	23000000.0	Fell	01/01/1947 12:00:00 AM	46.16000	134.65333	(46.16, 134.65333)

In [34]:

```
meteorites.fall.value_counts() # this command returns the entries in the column grouped by the entry and outputs the
```

```
Out[34]: fall
Found    44609
Fell      1107
Name: count, dtype: int64
```

```
In [37]: meteorites.value_counts(subset=['nametype','fall'], normalize=False)
```

```
Out[37]: nametype  fall
Valid     Found    44534
          Fell      1107
Relict    Found       75
Name: count, dtype: int64
```

```
In [42]: type(meteorites['mass (g)'].mean()) #this takes the mean of the whole column entries in the
```

```
Out[42]: numpy.float64
```

```
In [43]: meteorites['mass (g)'].quantile([0.01,0.05,0.5,0.95,0.99]) #this shows the quantile of tje
```

```
Out[43]: 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 [45]: meteorites['mass (g)'].median() #this shows the median, or the 50th percentile
```

```
Out[45]: 32.6
```

```
In [46]: meteorites['mass (g)'].max() #this shows the maximum entry in the "mass (g)" column of the dataframe
```

```
Out[46]: 60000000.0
```

```
In [48]: meteorites.loc[meteorites['mass (g)'].idxmax()] # this shows the row where the maximum entry in the "mass (g)" is by
```

Out[48]:
```
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 [49]:
```python
meteorites.recclass.nunique() #this shows the number of values that are not unique
```

Out[49]:
```
466
```

In [54]:
```python
meteorites.name.nunique() #this shows the number of values that are not unique
```

Out[54]:
```
45716
```

In [58]:
```python
meteorites.recclass.unique()  #this shows the number of values that are  unique
```

```
Out[58]: 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 [56]:
```
meteorites.describe()
```

Out[56]:

	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 [57]:
```
meteorites.describe(include='all')
```

Out[57]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
count	45716	45716.000000	45716	45716	4.558500e+04	45716	45425	38401.000000	38401.000000	38401
unique	45716	NaN	2	466	NaN	2	266	NaN	NaN	17100
top	Aachen	NaN	Valid	L6	NaN	Found	01/01/2003 12:00:00 AM	NaN	NaN	(0.0, 0.0)
freq	1	NaN	45641	8285	NaN	44609	3323	NaN	NaN	6214
mean	NaN	26889.735104	NaN	NaN	1.327808e+04	NaN	NaN	-39.122580	61.074319	NaN
std	NaN	16860.683030	NaN	NaN	5.749889e+05	NaN	NaN	46.378511	80.647298	NaN
min	NaN	1.000000	NaN	NaN	0.000000e+00	NaN	NaN	-87.366670	-165.433330	NaN
25%	NaN	12688.750000	NaN	NaN	7.200000e+00	NaN	NaN	-76.714240	0.000000	NaN
50%	NaN	24261.500000	NaN	NaN	3.260000e+01	NaN	NaN	-71.500000	35.666670	NaN
75%	NaN	40656.750000	NaN	NaN	2.026000e+02	NaN	NaN	0.000000	157.166670	NaN
max	NaN	57458.000000	NaN	NaN	6.000000e+07	NaN	NaN	81.166670	354.473330	NaN

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 [59]:

```python
import pandas as pd

#1     Create a DataFrame by reading in the 2019_Yellow_Taxi_Trip_Data.csv file. Examine the first 5 rows.

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

Out[59]:

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

In [84]:

```python
#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 dataframe
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 [69]:
```python
#3
#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.

# the 'fare_amount', 'tip_amount', 'tolls_amount' is in double brackets to output those columns
#and the describe() command is then used to describe those columns with details
df[['fare_amount', 'tip_amount', 'tolls_amount']].describe()
```

Out[69]:

	fare_amount	tip_amount	tolls_amount
count	10000.000000	10000.000000	10000.000000
mean	15.106313	2.634494	0.623447
std	13.954762	3.409800	6.437507
min	-52.000000	0.000000	-6.120000
25%	7.000000	0.000000	0.000000
50%	10.000000	2.000000	0.000000
75%	16.000000	3.250000	0.000000
max	176.000000	43.000000	612.000000

In [81]:
```python
#4
#Isolate the fare_amount, tip_amount, tolls_amount, and total_amount for the longest trip by distance (trip_distance)


#this command finds row with the longest trip distance using loc and idxmax and using trip_distance as the one to fir
#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 with maximum trip+_distance
df.loc[df['trip_distance'].idxmax()][['fare_amount', 'tip_amount', 'tolls_amount', 'total_amount']]
```

Out[81]:
```
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.