In [7]:

```
import pandas as pd
```

Exercise Part 4:

1.

Using the meteorite data from the Meteorite_Landings.csv file, create a pivot table that shows both the number of meteorites and the 95th percentile of meteorite mass for those that were found versus observed falling per year from 2005 through 2009 (inclusive). Hint: Be sure to convert the year column to a number as we did in the previous exercis e2. Using the meteorite data from the Meteorite_Landings.csv file, compare summary statistics of the mass column for the meteorites that were found versus observed fallng.ing.st tips.

In [31]:

```
meteor= pd.read_csv("Meteorite_Landings.csv")
meteor.head()
```

Out[31]:

	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

In [365…

```
# number 1
meteors = meteor.copy() # make a copy
meteors["year"]=meteors["year"].str.slice(6,11) # make the year column only year
meteors["year"]=pd.to_numeric(meteors['year']) # convert it to numeric type
meteors= meteors.rename(columns={'mass (g)' : 'mass'})# convert mass (g) to mass fo
meteors= meteors.fillna(0)# fill all empty with 0
```

In [323…

```
between_2005_2009 = meteors[(meteors["year"] >= 2005) & (meteors["year"] <= 2009)]
```

```
between_2005_2009["fall"].count()
```

Out[323…

```
6974
```

In [318…

```
found = (between_2005_2009["mass"][between_2005_2009["fall"] == "Found"]).quantile(
found
```

Out[318…

```
1839.9599999999969
```

In [320…

```
fell = (between_2005_2009["mass"][between_2005_2009["fall"] == "Fell"]).quantile(0.
fell
```

Out[320…

```
100000.0
```

In [355…

```
Found_vs_Fell = pd.DataFrame({
    "Count": ["Found", "Fell"],
    "95%": [found, fell]
})
Found_vs_Fell
```

Out[355…

	Count	95%
0	Found	1839.96
1	Fell	100000.00

In [549…

```
found_vs_fell_pivoted = Found_vs_Fell.pivot( columns = 'Count', values = '95%' )
found_vs_fell_pivoted
```

Out[549…

Count	Fell	Found
0	NaN	1839.96
1	100000.0	NaN

In [399…

```
# 2
meteorites = meteor.copy()
meteorites = meteorites.rename(columns={'mass (g)' : 'mass'})

found2 = meteorites.mass[meteorites[ "fall"] == "Found"].describe()
found2
```

Out[399…

```
count    4.451000e+04
mean     1.246192e+04
std      5.711058e+05
min      0.000000e+00
25%      6.940000e+00
50%      3.050000e+01
75%      1.780000e+02
max      6.000000e+07
Name: mass, dtype: float64
```

In [391…

```
fell2 = meteorites.mass[meteorites[ "fall"] == "Fell"].describe()
```

```
fell2
```

Out[391…

```
count    1.075000e+03
mean     4.707072e+04
std      7.170671e+05
min      1.000000e-01
25%      6.860000e+02
50%      2.800000e+03
75%      1.045000e+04
max      2.300000e+07
Name: mass, dtype: float64
```

In [405…

```
found2.compare(fell2) # comparrison between found 2 and fell2
```

Out[405…

	self	**other**
count	4.451000e+04	1.075000e+03
mean	1.246192e+04	4.707072e+04
std	5.711058e+05	7.170671e+05
min	0.000000e+00	1.000000e-01
25%	6.940000e+00	6.860000e+02
50%	3.050000e+01	2.800000e+03
75%	1.780000e+02	1.045000e+04
max	6.000000e+07	2.300000e+07

Exercise Part 5:

1. Using the taxi trip data in the 2019_Yellow_Taxi_Trip_Data.csv file, resample the data to an hourly frequency based on the dropoff time. Calculate the total trip_distance, fare_amount, tolls_amount, and tip_amount, then find the 5 hours with the most tips.

In [625…

```
taxi = pd.read_csv("2019_Yellow_Taxi_Trip_Data.csv")
taxis=taxi.copy()
```

In [627…

```
taxis=taxis.rename(columns={'tpep_dropoff_datetime' : 'drop_off'})
taxis=taxis.rename(columns={'tpep_pickup_datetime' : 'datetime'})

taxis.head()
```

Out[627…

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

In [629…

```
taxis["datetime"] = taxis["datetime"].apply(lambda x: x.split('T')[1].split('.')[0]
taxis["drop_off"]=taxis["drop_off"].apply(lambda x: x.split('T')[1].split('.')[0].r
```

In [631…

```
taxis["Sum"] = taxis["trip_distance"] + taxis["fare_amount"] + taxis["tolls_amount"
```

In [635…

```
taxis["datetime"]=pd.to_numeric(taxis["datetime"])
taxis["drop_off"]=pd.to_numeric(taxis["drop_off"])
```

In [637…

```
taxis.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   vendorid               10000 non-null  int64  
 1   datetime               10000 non-null  int64  
 2   drop_off               10000 non-null  int64  
 3   passenger_count        10000 non-null  int64  
 4   trip_distance          10000 non-null  float64
 5   ratecodeid             10000 non-null  int64  
 6   store_and_fwd_flag     10000 non-null  object 
 7   pulocationid           10000 non-null  int64  
 8   dolocationid           10000 non-null  int64  
 9   payment_type           10000 non-null  int64  
 10  fare_amount            10000 non-null  float64
 11  extra                  10000 non-null  float64
 12  mta_tax                10000 non-null  float64
 13  tip_amount             10000 non-null  float64
 14  tolls_amount           10000 non-null  float64
 15  improvement_surcharge  10000 non-null  float64
 16  total_amount           10000 non-null  float64
 17  congestion_surcharge   10000 non-null  float64
 18  Sum                    10000 non-null  float64
dtypes: float64(10), int64(8), object(1)
memory usage: 1.4+ MB
```

In [641…

```
taxis
```

Out[641…

	vendorid	datetime	drop_off	passenger_count	trip_distance	ratecodeid	store_and_
0	2	163942	171410	1	7.93	1	
1	1	163208	164526	1	2.00	1	
2	2	160844	162111	1	1.36	1	
3	2	162244	164326	1	1.00	1	
4	2	164511	165849	1	1.96	1	
...	...	...	...	...	...	...	
9995	1	173959	174926	2	1.30	1	
9996	1	175302	180045	1	1.40	1	
9997	1	170716	171135	1	0.70	1	
9998	1	173826	174928	2	2.50	1	
9999	1	172214	175209	1	3.00	1	

10000 rows × 19 columns

In [673…

```
(taxis["drop_off"] - taxis["datetime"]) == 5
```

Out[673…

```
0       False
1       False
2       False
3       False
4       False
        ...
9995    False
9996    False
9997    False
9998    False
9999    False
Length: 10000, dtype: bool
```

In [ ]: