Exercise Part 4:

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 exercise. 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 falling.

In [1]:
```
import pandas as pd
#this part of the code reads the csv needed and the first row in the dataframe
meteor = pd.read_csv('Meteorite_Landings.csv')
meteor.head(1)
```

Out[1]:

	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	50.775	6.08333	(50.775, 6.08333)

In [2]:
```
meteor_copy = meteor
#this part of the line takes the string that contains the year in the column year
meteor_copy["year"] = meteor_copy["year"].str.slice().str[6:10]
meteor_copy.head(1)
```

Out[2]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
0	Aachen	1	Valid	L5	21.0	Fell	1880	50.775	6.08333	(50.775, 6.08333)

In [3]:
```
#this part of the code drops the null entries in the year column
meteor_copy = meteor.dropna(subset=['year'])
#this part then converts the string year to an integer data type and replaces the c
meteor_copy['year'] = meteor_copy['year'].astype(int)
#the entries is then filtered out that are in the year 2005 to 2009
meteor_copy = meteor_copy[(meteor_copy['year']>=2005) & (meteor_copy['year']<=2009)
meteor_copy.head()
```

Out[3]:

```
                  name     id  nametype  recclass     mass  fall  year     reclat    reclong  Ge
                                                       (g)
 30           Almahata  48915     Valid  Ureilite-  3950.0  Fell  2008   20.74575   32.41275
                 Sitta                          an
 49          Ash Creek  48954     Valid         L6  9500.0  Fell  2009   31.80500  -97.01000
 82        Bassikounou  44876     Valid         H5 29560.0  Fell  2006   15.78333   -5.90000
101             Berduc  48975     Valid         L6   270.0  Fell  2008  -31.91000  -58.32833
148           Bunburra  48653     Valid    Eucrite   324.0  Fell  2007  -31.35000  129.19000
              Rockhole
```

In [4]:

```python
meteor_copy.pivot_table(index='year',columns='fall',values='mass (g)',aggfunc={'mas
```

Out[4]:

```
         <lambda_0>          count
fall     Fell       Found    Fell  Found
year
2005          NaN   4500.00   NaN   874.0
2006      25008.0   1600.50   5.0  2450.0
2007      89675.0   1126.90   8.0  1181.0
2008     106000.0   2274.80   9.0   948.0
2009       8333.4   1397.25   5.0  1492.0
```

2. 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 falling

In [5]:

```python
#this part uses the describe command to give a summary for the dataframe
meteor_copy1 = meteor.copy()
meteor_copy1.groupby(['fall'])['mass (g)'].describe()
```

Out[5]:

```
         count          mean            std  min     25%     50%      75%         max
fall
 Fell   1075.0  47070.715023  717067.125826  0.1  686.00  2800.0  10450.0  23000000.0
Found  44510.0  12461.922983  571105.752311  0.0    6.94    30.5    178.0  60000000.0
```

In [6]:

```
#this part also does the same in the previous code but in pivot table form
meteor_copy1.pivot_table(index='fall',values='mass (g)',aggfunc={'count', 'mean', '
```

Out[6]:

	<lambda_0>	<lambda_1>	count	max	mean	median	min	
fall								
Fell	686.00	10450.0	1075	23000000.0	47070.715023	2800.0	0.1	717067.
Found	6.94	178.0	44510	60000000.0	12461.922983	30.5	0.0	571105.

In [ ]:

In [ ]:

In [ ]:

Exercise Part 4: 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 [7]:

```
taxis = pd.read_csv('2019_Yellow_Taxi_Trip_Data.csv') # this line of code reads the
taxis.head()
```

Out[7]:

	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 [8]:

```
taxis_copy = taxis
taxis_copy = taxis_copy.set_index('tpep_dropoff_datetime') # this sets the index to
taxis_copy
```

Out[8]:

	vendorid	tpep_pickup_datetime	passenger_count	trip_distance	rat
tpep_dropoff_datetime					
2019-10-23T17:14:10.000	2	2019-10-23T16:39:42.000	1	7.93	
2019-10-23T16:45:26.000	1	2019-10-23T16:32:08.000	1	2.00	
2019-10-23T16:21:11.000	2	2019-10-23T16:08:44.000	1	1.36	
2019-10-23T16:43:26.000	2	2019-10-23T16:22:44.000	1	1.00	
2019-10-23T16:58:49.000	2	2019-10-23T16:45:11.000	1	1.96	
...	...	...	...	...	
2019-10-23T17:49:26.000	1	2019-10-23T17:39:59.000	2	1.30	
2019-10-23T18:00:45.000	1	2019-10-23T17:53:02.000	1	1.40	
2019-10-23T17:11:35.000	1	2019-10-23T17:07:16.000	1	0.70	
2019-10-23T17:49:28.000	1	2019-10-23T17:38:26.000	2	2.50	
2019-10-23T17:52:09.000	1	2019-10-23T17:22:14.000	1	3.00	

10000 rows × 17 columns

In [9]:

```
taxis_find = taxis_copy[['fare_amount','trip_distance','fare_amount','tolls_amount'
taxis_find
#this then finds the needed columns and is stored to another dataframe to avoid mis
```

Out[9]:

	fare_amount	trip_distance	fare_amount	tolls_amount	tip_amount
tpep_dropoff_datetime					
2019-10-23T17:14:10.000	29.5	7.93	29.5	6.12	7.98
2019-10-23T16:45:26.000	10.5	2.00	10.5	0.00	0.00
2019-10-23T16:21:11.000	9.5	1.36	9.5	0.00	2.00
2019-10-23T16:43:26.000	13.0	1.00	13.0	0.00	4.32
2019-10-23T16:58:49.000	10.5	1.96	10.5	0.00	0.50
...	...	...	...	...	...
2019-10-23T17:49:26.000	8.0	1.30	8.0	0.00	2.46
2019-10-23T18:00:45.000	8.0	1.40	8.0	0.00	0.00
2019-10-23T17:11:35.000	5.0	0.70	5.0	0.00	0.00
2019-10-23T17:49:28.000	10.0	2.50	10.0	0.00	0.00
2019-10-23T17:52:09.000	19.0	3.00	19.0	0.00	2.50

10000 rows × 5 columns

In [10]:

```
taxis_copy.nlargest(5, 'tip_amount')['tip_amount']
#this part finds the top 5 entries in the dataframe with the largest tip_amount alo
```

Out[10]:

```
tpep_dropoff_datetime
2019-10-23T16:30:00.000    43.00
2019-10-23T16:15:32.000    40.00
2019-10-23T18:58:16.000    37.25
2019-10-23T16:52:02.000    36.00
2019-10-23T16:11:22.000    30.03
Name: tip_amount, dtype: float64
```

In [ ]: