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.

In [53]:
```
import pandas as pd
meteorites = pd.read_csv("Meteorite_Landings.csv")
```

In [54]:
```
#splicing
meteorites['year'] = meteorites['year'].str.slice(6,10)
```

In [55]:
```
# convert it to a  numeric data type

meteorites['year'] = pd.to_numeric(meteorites['year'], errors='coerce')
```

In [56]:
```
# meteorites falling per year from 2005 through 2009

meteorites = meteorites[(meteorites['year'] <= 2009) & (meteorites['year'] >= 2005)]
meteorites
```

Out[56]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat
30	Almahata Sitta	48915	Valid	Ureilite-an	3950.000	Fell	2008.0	20.74575
49	Ash Creek	48954	Valid	L6	9500.000	Fell	2009.0	31.80500
82	Bassikounou	44876	Valid	H5	29560.000	Fell	2006.0	15.78333
101	Berduc	48975	Valid	L6	270.000	Fell	2008.0	-31.91000
148	Bunburra Rockhole	48653	Valid	Eucrite	324.000	Fell	2007.0	-31.35000
...	...	...	...	...	...	...	...	...
38396	Yabrin 003	48974	Valid	Acapulcoite	21.048	Found	2008.0	23.31522
45664	Yaringie Hill	48950	Valid	H5	5750.000	Found	2006.0	-32.08287
45668	Yarle Lakes 004	52945	Valid	CK4	4.600	Found	2009.0	-30.50000
45674	Yelland Dry Lake	52641	Valid	H4	76000.000	Found	2007.0	39.35067
45685	Youxi	55793	Valid	Mesosiderite-C	218000.000	Found	2006.0	26.06000

6974 rows × 10 columns

In [73]:
```
# used a built-in function that generates pivot-style spreadsheet into DataFrame

pivot_table = meteorites.pivot_table(
    index='year',
    columns='fall',
    values='mass (g)',
    aggfunc={'mass (g)': lambda x: x.quantile(0.95), 'fall': 'count'}
)
```

In [74]:
```
# rename the columns for clarification

pivot_table.columns = ['count_fell', 'count_found', 'mass_95th_fell', 'mass_95th_fo

# display the table
pivot_table
```

Out[74]:

year	count_fell	count_found	mass_95th_fell	mass_95th_found
2005.0	NaN	875.0	NaN	4500.00
2006.0	5.0	2451.0	25008.0	1600.50
2007.0	8.0	1181.0	89675.0	1126.90
2008.0	9.0	948.0	106000.0	2274.80
2009.0	5.0	1492.0	8333.4	1397.25

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 [81]:
```
# without using .groupby

fell_meteorites = meteorites[meteorites['fall'] == 'Fell']['mass (g)']
found_meteorites = meteorites[meteorites['fall'] == 'Found']['mass (g)']

# compute summary statistics
fell_stats = fell_meteorites.describe()
found_stats = found_meteorites.describe()

# combine results into a DataFrame for comparison
summary_stats = pd.DataFrame({'Fell': fell_stats, 'Found': found_stats})

summary_stats
```

Out[81]:

	Fell	Found
count	27.000000	6.945000e+03
mean	19029.665185	1.573986e+03
std	34081.623779	4.202089e+04
min	18.410000	0.000000e+00
25%	410.000000	7.500000e+00
50%	3950.000000	3.450000e+01
75%	8206.500000	1.970000e+02
max	110000.000000	3.000000e+06

In [80]:
```
# statistics using .groupby
summary_stats = meteorites.groupby('fall')['mass (g)'].describe()
summary_stats
```

Out[80]:

fall	count	mean	std	min	25%	50%	75%	max
Fell	27.0	19029.665185	34081.623779	18.41	410.0	3950.0	8206.5	110000.0
Found	6945.0	1573.986245	42020.893987	0.00	7.5	34.5	197.0	3000000.0

Exercise Part 5

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 [78]:
```
import pandas as pd

taxis = pd.read_csv('2019_Yellow_Taxi_Trip_Data.csv')

# dropoff time column to datetime format convertion
taxis['tpep_dropoff_datetime'] = pd.to_datetime(taxis['tpep_dropoff_datetime'])

# resample data to hourly frequency based on dropoff time
hourly_data = taxis.resample('h', on='tpep_dropoff_datetime').sum()[['trip_distance

# top 5 hours with the highest tip amounts
most_tips = hourly_data.nlargest(5, 'tip_amount')

most_tips
```

Out[78]:

tpep_dropoff_datetime	trip_distance	fare_amount	tolls_amount	tip_amount
2019-10-23 16:00:00	10676.95	67797.76	699.04	12228.64
2019-10-23 17:00:00	16052.83	70131.91	4044.04	12044.03
2019-10-23 18:00:00	3104.56	11565.56	1454.67	1907.64
2019-10-23 15:00:00	14.34	213.50	0.00	51.75
2019-10-23 19:00:00	98.59	268.00	24.48	25.74

In [ ]:
```

```