Corpuz, Micki Laurren B. | 03 April 2025

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 exercise.

In [162…
```
import pandas as pd
meteorites = pd.read_csv("Meteorite_Landings.csv")
```

In [163…
```
# Splice and Convert to numeric
meteorites['year'] = meteorites.year.str.slice(6, 11)
meteorites['year'] = pd.to_numeric(meteorites['year'], errors='coerce')
```

In [164…
```
meteorites = meteorites[(meteorites['year'] >= 2005) & (meteorites['year'] <= 2009)]
years = meteorites['year'].unique()
```

In [165…
```
# Create a pivot table using the built-in .pivot_table()
pivot_table = meteorites.pivot_table(
    index='year',
    columns='fall',
    values='mass (g)',
    aggfunc=['count', lambda x: x.quantile(0.95)]
)
```

In [166…
```
pivot_table.columns = ['No. Fell', 'No. Found', 'Mass 95th Fell', 'Mass 95th Found']
pivot_table
```

Out[166…

year	No. Fell	No. Found	Mass 95th Fell	Mass 95th Found
2005.0	NaN	874.0	NaN	4500.00
2006.0	5.0	2450.0	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

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 [167…
```
# Separately calculate the summary statistics for each category, without using built-in .groupby()
fell = meteorites[meteorites['fall'] == 'Fell']['mass (g)']
found = meteorites[meteorites['fall'] == 'Found']['mass (g)']
fell_stats = fell_meteorites.describe()
found_stats = found_meteorites.describe()
```

In [168…
```
summary = pd.DataFrame({'Fell': fell_stats, 'Found': found_stats})
summary
```

Out[168…

	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 [169…

```
# Provide the summary statistics of categories of 'fall' using built-in .groupby()
summaryStat = meteorites.groupby('fall')['mass (g)'].describe()
summaryStat
```

Out[169…

	count	mean	std	min	25%	50%	75%	max
fall								
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 [170…

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

In [171…

```
# Resample the data following panda's built-in .resample()

taxis['tpep_dropoff_datetime'] = pd.to_datetime(taxis['tpep_dropoff_datetime'])
new_taxis = taxis.resample('h', on = 'tpep_dropoff_datetime').sum()[['trip_distance', 'fare_amount', 'tolls_amount', 'tip_amou
```

In [172…

```
top_5_tips = new_taxis.nlargest(5, 'tip_amount')
top_5_tips
```

Out[172…

	trip_distance	fare_amount	tolls_amount	tip_amount
tpep_dropoff_datetime				
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 [ ]: