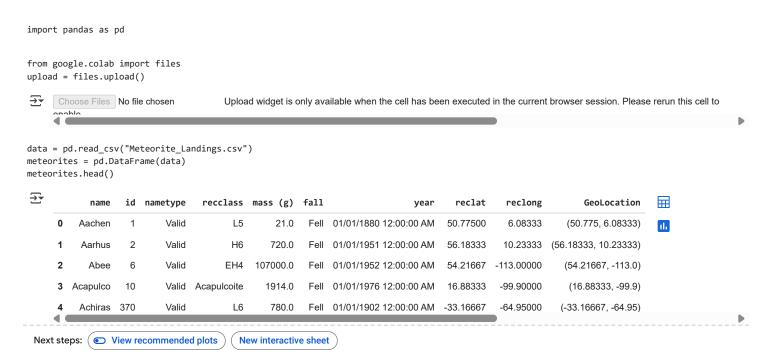
## Exercise (Part 4)



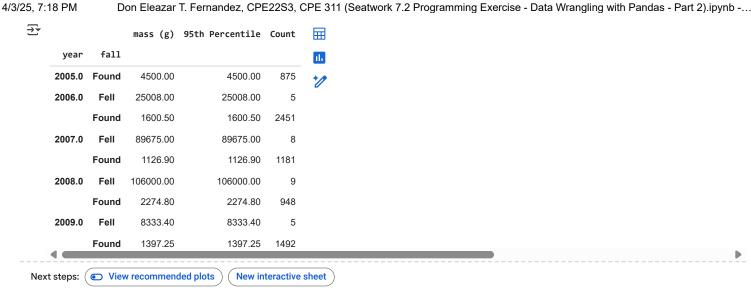
1. Using the meteorite data from the Meteorite\_Landings.csv file, create a pivot table that shows both the number of meteorites and the 95th percentile of meteorite mass for those that were found versus observed falling per year from 2005 through 2009 (inclusive). Hint: Be sure to convert the year column to a number as we did in the previous exercise.

```
filter = lambda x: x.str[6:10]
meteorites["year"] = filter(meteorites["year"])
meteorites["year"] = meteorites["year"].apply(pd.to_numeric)
meteorites.dtypes
₹
                         0
                     object
          name
                      int64
        nametype
                     object
        recclass
                     object
        mass (g)
                    float64
           fall
                     object
          year
                    float64
          reclat
                    float64
                    float64
         reclong
      GeoLocation
                    object
      dtuna, object
meteorites_filtered = meteorites[(meteorites["year"] >= 2005) & (meteorites["year"] <= 2009)]</pre>
meteorites_filtered = meteorites_filtered.set_index("year")
```

meteorites Fell = meteorites filtered[(meteorites filtered["fall"] == "Fell")]

meteorites\_Fell.iloc[:, 4].quantile([0.95])

```
₹
            mass (g)
      0.95
           100000.0
     dtune: floot64
meteorites_Found = meteorites_filtered[(meteorites_filtered["fall"] == "Found")]
meteorites_Found.iloc[:, 4].quantile([0.95])
₹
            mass (g)
      0.95
             1841.64
Number_1_1 = meteorites_filtered.groupby(["year", "fall"])[["mass (g)"]].quantile(0.95)
Number_1_1
₹
                                  \blacksquare
                      mass (g)
               fall
       year
                                  th
                       4500.00
      2005.0 Found
                       25008.00
      2006.0
               Fell
                       1600.50
              Found
      2007.0
              Fell
                      89675.00
              Found
                        1126.90
      2008.0
               Fell
                     106000.00
              Found
                        2274.80
      2009.0
               Fell
                        8333.40
              Found
                        1397.25
 Next steps: ( View recommended plots )
                                          New interactive sheet
Number_1_2 = meteorites_filtered.groupby(["year", "fall"])[["name"]].count()
Number_1_2
₹
                             name
               fall
       year
                             ılı.
      2005.0 Found
                      875
      2006.0
               Fell
                        5
              Found 2451
      2007.0
              Fell
                        8
                     1181
              Found
      2008.0
              Fell
                        9
              Found
                      948
      2009.0
              Fell
                        5
              Found
                     1492
 Next steps: ( View recommended plots
                                          New interactive sheet
Number_1_1["95th Percentile"] = Number_1_1
Number_1_1["Count"] = Number_1_2
{\tt Number\_1\_1}
```



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.

Number\_2 = meteorites\_filtered.groupby("fall")["mass (g)"].describe() Number\_2



## Exercise (Part 5)

from google.colab import files upload = files.upload()

Choose Files 2019 Yello...rip Data.csv 2019\_Yellow\_Taxi\_Trip\_Data.csv(text/csv) - 1000622 bytes, last modified: 4/3/2025 - 100% done wing 2010 Vallow Tavi Thin Data cov to 2010 Vallow Tavi Thin Data (1) cov

data1 = pd.read\_csv("2019\_Yellow\_Taxi\_Trip\_Data.csv") taxi = pd.DataFrame(data1) taxi.head()

₹		vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	ratecodeid	store_and_fwd_flag	pulocationid	ı
	0	2	2019-10- 23T16:39:42.000	2019-10- 23T17:14:10.000	1	7.93	1	N	138	
	1	1	2019-10- 23T16:32:08.000	2019-10- 23T16:45:26.000	1	2.00	1	N	11	
	2	2	2019-10- 23T16:08:44.000	2019-10-23T16:21:11.000	1	1.36	1	N	163	
	3	2	2019-10- 23T16:22:44.000	2019-10- 23T16:43:26.000	1	1.00	1	N	170	
	4	2	2019-10- 23T16:45:11.000	2019-10- 23T16:58:49.000	1	1.96	1	N	163	

Next steps: ( View recommended plots New interactive sheet 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.

```
# resample the data to an hourly frequency based on the dropoff time
taxi['tpep_dropoff_datetime'] = pd.to_datetime(taxi['tpep_dropoff_datetime'])
taxi.set_index("tpep_dropoff_datetime", inplace = True)
taxi.index = pd.to_datetime(taxi.index)
taxi['Hour'] = taxi.index.hour
# Calculate the total trip_distance, fare_amount, tolls_amount, and tip_amount, then find the 5 hours with the most tips.
taxi = taxi.groupby('Hour')[['trip_distance', 'fare_amount', 'tolls_amount', 'tip_amount']].sum()
taxi = taxi["tip_amount"].nlargest(5)
taxi
₹
            tip_amount
     Hour
       16
              12249.32
       17
              12044.03
               1907.64
       18
       15
                 75.10
                 25.74
       19
```

dtype: float64