Exercise (Part 4)

```
import pandas as pd
```

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

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
data = pd.read_csv("Meteorite_Landings.csv")
meteorites = pd.DataFrame(data)
meteorites.head()
```

	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	50.77500	6.08333	(50.775, 6.08333)
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333	(56.18333, 10.23333)
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667	-113.00000	(54.21667, -113.0)
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000	(16.88333, -99.9)
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000	(-33.16667, -64.95)

Next steps: View recommended plots New interactive sheet

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
name	object
id	int64
nametype	object
recclass	object
mass (g)	float64
fall	object
year	float64
reclat	float64
reclong	float64
GeoLocation	object

dtype: object

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

```
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

dtype: float64

```
meteorites_Found = meteorites_filtered[(meteorites_filtered["fall"] == "Found")]
meteorites_Found.iloc[:, 4].quantile([0.95])
```

	mass (g)
0.95	1841.64

dtype: float64

```
Number_1_1 = meteorites_filtered.groupby(["year", "fall"])[["mass (g)"]].quantile(0.95)
Number_1_1
```

year	fall	mass (g)
2005.0	**Found**	4500.00
2006.0	**Fell**	25008.00
	Found	1600.50
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
```

year	fall	name
2005.0	**Found**	875
2006.0	**Fell**	5
	Found	2451
2007.0	**Fell**	8
	Found	1181
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
Number_1_1
```

year	fall	mass (g)	95th Percentile	Count
2005.0	Found	4500.00	4500.00	875
2006.0	Fell	25008.00	25008.00	5
	Found	1600.50	1600.50	2451
2007.0	Fell	89675.00	89675.00	8
	Found	1126.90	1126.90	1181
2008.0	Fell	106000.00	106000.00	9
	Found	2274.80	2274.80	948
2009.0	Fell	8333.40	8333.40	5
	Found	1397.25	1397.25	1492

Next steps: View recommended plots New interactive sheet

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
```

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

Next steps: View recommended plots New interactive sheet

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

```
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
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
```

Hour	tip_amount
16	12249.32
17	12044.03
18	1907.64
15	75.10
19	25.74

dtype: float64