In [56]:
```
import pandas as pd
import numpy as np
```

In [60]:
```
meteorite = pd.read_csv("Meteorite_Landings.csv", nrows = 5) # dito ang nasa loob l
meteorite
```

Out[60]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong
0	Aachen	1	Valid	L5	21	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333
1	Aarhus	2	Valid	H6	720	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333
2	Abee	6	Valid	EH4	107000	Fell	01/01/1952 12:00:00 AM	54.21667	-113.00000
3	Acapulco	10	Valid	Acapulcoite	1914	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000
4	Achiras	370	Valid	L6	780	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000

In [61]:
```
meteorites = pd.read_csv("Meteorite_Landings.csv") # dito kinuha ko na yung kabuoha
```

In [7]:
```
meteorites
```

Out[7]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong
0	Aachen	1	Valid	L5	21	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333
1	Aarhus	2	Valid	H6	720	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333
2	Abee	6	Valid	EH4	107000	Fell	01/01/1952 12:00:00 AM	54.21667	-113.00000
3	Acapulco	10	Valid	Acapulcoite	1914	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000
4	Achiras	370	Valid	L6	780	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000

In [13]:
```
meteorites["name"] # checking for a series of a specific column
```

Out[13]:
```
0      Aachen
1      Aarhus
2        Abee
3    Acapulco
4     Achiras
Name: name, dtype: object
```

In [14]:
```
meteorites.name #checking for a series of a specific column other way
```

Out[14]:
```
0      Aachen
1      Aarhus
2        Abee
3    Acapulco
4     Achiras
Name: name, dtype: object
```

In [12]:
```
meteorites.columns # checking for the names of every columns
```

Out[12]:
```
Index(['name', 'id', 'nametype', 'recclass', 'mass (g)', 'fall', 'year',
       'reclat', 'reclong', 'GeoLocation'],
      dtype='object')
```

In [15]:
```
meteorites.index #checking the index of the dataframe
```

Out[15]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [25]:
```
#using data from an API (correct response)

import requests
```

```python
response = requests.get(
    'https://data.nasa.gov/resource/gh4g-9sfh.json',
    params = {'$limit': 50_000}

)

if response.ok:
    payload = response.json()
else:
    print(f'Request was not successful and returned code: {response.status_code}.')
    payload = None
```

In [26]:

```python
payload[:20]
```

Out[26]:


```
[{'name': 'Aachen',
  'id': '1',
  'nametype': 'Valid',
  'recclass': 'L5',
  'mass': '21',
  'fall': 'Fell',
  'year': '1880-01-01T00:00:00.000',
  'reclat': '50.775000',
  'reclong': '6.083330',
  'geolocation': {'latitude': '50.775', 'longitude': '6.08333'}},
 {'name': 'Aarhus',
  'id': '2',
  'nametype': 'Valid',
  'recclass': 'H6',
  'mass': '720',
  'fall': 'Fell',
  'year': '1951-01-01T00:00:00.000',
  'reclat': '56.183330',
  'reclong': '10.233330',
  'geolocation': {'latitude': '56.18333', 'longitude': '10.23333'}},
 {'name': 'Abee',
  'id': '6',
  'nametype': 'Valid',
  'recclass': 'EH4',
  'mass': '107000',
  'fall': 'Fell',
  'year': '1952-01-01T00:00:00.000',
  'reclat': '54.216670',
  'reclong': '-113.000000',
  'geolocation': {'latitude': '54.21667', 'longitude': '-113.0'}},
 {'name': 'Acapulco',
  'id': '10',
  'nametype': 'Valid',
  'recclass': 'Acapulcoite',
  'mass': '1914',
  'fall': 'Fell',
  'year': '1976-01-01T00:00:00.000',
  'reclat': '16.883330',
  'reclong': '-99.900000',
  'geolocation': {'latitude': '16.88333', 'longitude': '-99.9'}},
 {'name': 'Achiras',
  'id': '370',
  'nametype': 'Valid',
  'recclass': 'L6',
  'mass': '780',
  'fall': 'Fell',
  'year': '1902-01-01T00:00:00.000',
  'reclat': '-33.166670',
  'reclong': '-64.950000',
  'geolocation': {'latitude': '-33.16667', 'longitude': '-64.95'}},
 {'name': 'Adhi Kot',
  'id': '379',
  'nametype': 'Valid',
  'recclass': 'EH4',
  'mass': '4239',
  'fall': 'Fell',
```

'year': '1919-01-01T00:00:00.000',
'reclat': '32.100000',
'reclong': '71.800000',
'geolocation': {'latitude': '32.1', 'longitude': '71.8'}},
{'name': 'Adzhi-Bogdo (stone)',
'id': '390',
'nametype': 'Valid',
'recclass': 'LL3-6',
'mass': '910',
'fall': 'Fell',
'year': '1949-01-01T00:00:00.000',
'reclat': '44.833330',
'reclong': '95.166670',
'geolocation': {'latitude': '44.83333', 'longitude': '95.16667'}},
{'name': 'Agen',
'id': '392',
'nametype': 'Valid',
'recclass': 'H5',
'mass': '30000',
'fall': 'Fell',
'year': '1814-01-01T00:00:00.000',
'reclat': '44.216670',
'reclong': '0.616670',
'geolocation': {'latitude': '44.21667', 'longitude': '0.61667'}},
{'name': 'Aguada',
'id': '398',
'nametype': 'Valid',
'recclass': 'L6',
'mass': '1620',
'fall': 'Fell',
'year': '1930-01-01T00:00:00.000',
'reclat': '-31.600000',
'reclong': '-65.233330',
'geolocation': {'latitude': '-31.6', 'longitude': '-65.23333'}},
{'name': 'Aguila Blanca',
'id': '417',
'nametype': 'Valid',
'recclass': 'L',
'mass': '1440',
'fall': 'Fell',
'year': '1920-01-01T00:00:00.000',
'reclat': '-30.866670',
'reclong': '-64.550000',
'geolocation': {'latitude': '-30.86667', 'longitude': '-64.55'}},
{'name': 'Aioun el Atrouss',
'id': '423',
'nametype': 'Valid',
'recclass': 'Diogenite-pm',
'mass': '1000',
'fall': 'Fell',
'year': '1974-01-01T00:00:00.000',
'reclat': '16.398060',
'reclong': '-9.570280',
'geolocation': {'latitude': '16.39806', 'longitude': '-9.57028'}},
{'name': 'Aïr',
'id': '424',

```
 'nametype': 'Valid',
 'recclass': 'L6',
 'mass': '24000',
 'fall': 'Fell',
 'year': '1925-01-01T00:00:00.000',
 'reclat': '19.083330',
 'reclong': '8.383330',
 'geolocation': {'latitude': '19.08333', 'longitude': '8.38333'}},
{'name': 'Aire-sur-la-Lys',
 'id': '425',
 'nametype': 'Valid',
 'recclass': 'Unknown',
 'fall': 'Fell',
 'year': '1769-01-01T00:00:00.000',
 'reclat': '50.666670',
 'reclong': '2.333330',
 'geolocation': {'latitude': '50.66667', 'longitude': '2.33333'}},
{'name': 'Akaba',
 'id': '426',
 'nametype': 'Valid',
 'recclass': 'L6',
 'mass': '779',
 'fall': 'Fell',
 'year': '1949-01-01T00:00:00.000',
 'reclat': '29.516670',
 'reclong': '35.050000',
 'geolocation': {'latitude': '29.51667', 'longitude': '35.05'}},
{'name': 'Akbarpur',
 'id': '427',
 'nametype': 'Valid',
 'recclass': 'H4',
 'mass': '1800',
 'fall': 'Fell',
 'year': '1838-01-01T00:00:00.000',
 'reclat': '29.716670',
 'reclong': '77.950000',
 'geolocation': {'latitude': '29.71667', 'longitude': '77.95'}},
{'name': 'Akwanga',
 'id': '432',
 'nametype': 'Valid',
 'recclass': 'H',
 'mass': '3000',
 'fall': 'Fell',
 'year': '1959-01-01T00:00:00.000',
 'reclat': '8.916670',
 'reclong': '8.433330',
 'geolocation': {'latitude': '8.91667', 'longitude': '8.43333'}},
{'name': 'Akyumak',
 'id': '433',
 'nametype': 'Valid',
 'recclass': 'Iron, IVA',
 'mass': '50000',
 'fall': 'Fell',
 'year': '1981-01-01T00:00:00.000',
 'reclat': '39.916670',
 'reclong': '42.816670',
```

```
  'geolocation': {'latitude': '39.91667', 'longitude': '42.81667'}},
 {'name': 'Al Rais',
  'id': '446',
  'nametype': 'Valid',
  'recclass': 'CR2-an',
  'mass': '160',
  'fall': 'Fell',
  'year': '1957-01-01T00:00:00.000',
  'reclat': '24.416670',
  'reclong': '39.516670',
  'geolocation': {'latitude': '24.41667', 'longitude': '39.51667'}},
 {'name': 'Al Zarnkh',
  'id': '447',
  'nametype': 'Valid',
  'recclass': 'LL5',
  'mass': '700',
  'fall': 'Fell',
  'year': '2001-01-01T00:00:00.000',
  'reclat': '13.660330',
  'reclong': '28.960000',
  'geolocation': {'latitude': '13.66033', 'longitude': '28.96'}},
 {'name': 'Alais',
  'id': '448',
  'nametype': 'Valid',
  'recclass': 'CI1',
  'mass': '6000',
  'fall': 'Fell',
  'year': '1806-01-01T00:00:00.000',
  'reclat': '44.116670',
  'reclong': '4.083330',
  'geolocation': {'latitude': '44.11667', 'longitude': '4.08333'}}]
```


In [23]:

```python
#using data from an API (unsuccessful response)

import requests #API LIBRARY

response = requests.get(
    'https://data.nasa.gov/gh4g-9sfh.json',
    params = {'$limit': 50_000}

)

if response.ok:
    payload = response.json()
else:
    print(f'Request was not successful and returned code: {response.status_code}.')
    payload = None
```

```
Request was not successful and returned code: 404.
```

In [29]:

```python
df = pd.DataFrame(payload) # transfer the json into pandas
df.head(5)
```

Out[29]:

	name	id	nametype	recclass	mass	fall	year	reclat	recl
0	Aachen	1	Valid	L5	21	Fell	1880-01-01T00:00:00.000	50.775000	6.083
1	Aarhus	2	Valid	H6	720	Fell	1951-01-01T00:00:00.000	56.183330	10.233
2	Abee	6	Valid	EH4	107000	Fell	1952-01-01T00:00:00.000	54.216670	-113.000
3	Acapulco	10	Valid	Acapulcoite	1914	Fell	1976-01-01T00:00:00.000	16.883330	-99.900
4	Achiras	370	Valid	L6	780	Fell	1902-01-01T00:00:00.000	-33.166670	-64.950

In [39]:
```
meteorites.shape #shape the size of rows and columns of the data frame
```

Out[39]:
```
(5, 10)
```

In [31]:
```
meteorites.columns # command to check all the names of every columns
```

Out[31]:
```
Index(['name', 'id', 'nametype', 'recclass', 'mass (g)', 'fall', 'year',
       'reclat', 'reclong', 'GeoLocation'],
      dtype='object')
```

In [32]:
```
meteorites.dtypes # determine the datatypes of each columns
```

Out[32]:
```
name            object
id               int64
nametype        object
recclass        object
mass (g)         int64
fall            object
year            object
reclat         float64
reclong        float64
GeoLocation     object
dtype: object
```

In [54]:
```
meteorites.head(10) # printing the first 10 data of the data frame
```

Out[54]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667	-113.00000
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000
5	Adhi Kot	379	Valid	EH4	4239.0	Fell	01/01/1919 12:00:00 AM	32.10000	71.80000
6	Adzhi-Bogdo (stone)	390	Valid	LL3-6	910.0	Fell	01/01/1949 12:00:00 AM	44.83333	95.16667
7	Agen	392	Valid	H5	30000.0	Fell	01/01/1814 12:00:00 AM	44.21667	0.61667
8	Aguada	398	Valid	L6	1620.0	Fell	01/01/1930 12:00:00 AM	-31.60000	-65.23333
9	Aguila Blanca	417	Valid	L	1440.0	Fell	01/01/1920 12:00:00 AM	-30.86667	-64.55000

In [55]:

```
meteorites.tail(5) # printing the last 5 data of the dataframe
```

Out[55]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	r
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990 12:00:00 AM	29.03700	17
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999 12:00:00 AM	13.78333	8
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939 12:00:00 AM	49.25000	17
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003 12:00:00 AM	49.78917	41
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976 12:00:00 AM	33.98333	-115

In [64]:
```
meteorites.info() # checking for the numbers of data of every columns and show thei
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45716 entries, 0 to 45715
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   name         45716 non-null  object
 1   id           45716 non-null  int64
 2   nametype     45716 non-null  object
 3   recclass     45716 non-null  object
 4   mass (g)     45585 non-null  float64
 5   fall         45716 non-null  object
 6   year         45425 non-null  object
 7   reclat       38401 non-null  float64
 8   reclong      38401 non-null  float64
 9   GeoLocation  38401 non-null  object
dtypes: float64(3), int64(1), object(6)
memory usage: 3.5+ MB
```

In [74]:
```
meteorites # loading the data
```

Out[74]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667
...	...	...	...	...	...	...	...	...
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990 12:00:00 AM	29.03700
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999 12:00:00 AM	13.78333
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939 12:00:00 AM	49.25000
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003 12:00:00 AM	49.78917
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976 12:00:00 AM	33.98333

45716 rows × 10 columns

In [71]:
```
meteorites["name","recclass"] #this will result into a error message because this i
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('name', 'recclass')

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[71], line 1
----> 1 meteorites["name","recclass"]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__getitem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: ('name', 'recclass')
```

In [73]:
```
meteorites[["name","recclass"]] # this is the correct way of calling multiple colum
```

Out[73]:

	name	recclass
0	Aachen	L5
1	Aarhus	H6
2	Abee	EH4
3	Acapulco	Acapulcoite
4	Achiras	L6
...	...	...
45711	Zillah 002	Eucrite
45712	Zinder	Pallasite, ungrouped
45713	Zlin	H4
45714	Zubkovsky	L6
45715	Zulu Queen	L3.7

45716 rows × 2 columns

SELECTING DATAS USING INDEXING

In [81]:

```
# selecting row of data using indexing
meteorites[100:104]
# instances [starting : ending] referring to row of data
```

Out[81]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclon
100	Benton	5026	Valid	LL6	2840.0	Fell	01/01/1949 12:00:00 AM	45.95000	-67.5500
101	Berduc	48975	Valid	L6	270.0	Fell	01/01/2008 12:00:00 AM	-31.91000	-58.3283
102	Béréba	5028	Valid	Eucrite-mmict	18000.0	Fell	01/01/1924 12:00:00 AM	11.65000	-3.6500
103	Berlanguillas	5029	Valid	L6	1440.0	Fell	01/01/1811 12:00:00 AM	41.68333	-3.8000

In [88]:

```
# selecting using iloc
meteorites.iloc[100:104,[0,3,4,6]]
# instace = iloc[index of rows(you can use splicing like i did),columns[index of co
```

Out[88]:

	name	recclass	mass (g)	year
100	Benton	LL6	2840.0	01/01/1949 12:00:00 AM
101	Berduc	L6	270.0	01/01/2008 12:00:00 AM
102	Béréba	Eucrite-mmict	18000.0	01/01/1924 12:00:00 AM
103	Berlanguillas	L6	1440.0	01/01/1811 12:00:00 AM

BE MINDFUL YOU CANT USE INDEXING IN LOC

In [85]:
```
meteorites.loc[100:104,'mass (g)':'year']
# instance = loc[index of rows(you can use splicing like i did),name of column lite
```

Out[85]:

	mass (g)	fall	year
100	2840.0	Fell	01/01/1949 12:00:00 AM
101	270.0	Fell	01/01/2008 12:00:00 AM
102	18000.0	Fell	01/01/1924 12:00:00 AM
103	1440.0	Fell	01/01/1811 12:00:00 AM
104	960.0	Fell	01/01/2004 12:00:00 AM

In [87]:
```
meteorites.iloc[[-1],[-1]] # example of calling the last row and last column of the
```

Out[87]:

	GeoLocation
45715	(33.98333, -115.68333)

In [92]:
```
(meteorites["mass (g)"] > 50) & (meteorites.fall == 'Found')
# you can do conditions
```

Out[92]:
```
0        False
1        False
2        False
3        False
4        False
         ...
45711     True
45712    False
45713    False
45714     True
45715     True
Length: 45716, dtype: bool
```

In [94]:
```
# and pass it to dataframe so you can see the row of data that passes the condition
meteorites[(meteorites["mass (g)"] > 50) & (meteorites.fall == 'Found')]
```

Out[94]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	
37	Northwest Africa 5815	50693	Valid	L5	256.80	Found	NaN	0.00000	
757	Dominion Range 03239	32591	Valid	L6	69.50	Found	01/01/2002 12:00:00 AM	NaN	
804	Dominion Range 03240	32592	Valid	LL5	290.90	Found	01/01/2002 12:00:00 AM	NaN	
1111	Abajo	4	Valid	H5	331.00	Found	01/01/1982 12:00:00 AM	26.80000	-1
1112	Abar al' Uj 001	51399	Valid	H3.8	194.34	Found	01/01/2008 12:00:00 AM	22.72192	
...	...	...	...	...	...	...	...	...	
45709	Zhongxiang	30406	Valid	Iron	100000.00	Found	01/01/1981 12:00:00 AM	31.20000	1
45710	Zillah 001	31355	Valid	L6	1475.00	Found	01/01/1990 12:00:00 AM	29.03700	
45711	Zillah 002	31356	Valid	Eucrite	172.00	Found	01/01/1990 12:00:00 AM	29.03700	
45714	Zubkovsky	31357	Valid	L6	2167.00	Found	01/01/2003 12:00:00 AM	49.78917	
45715	Zulu Queen	30414	Valid	L3.7	200.00	Found	01/01/1976 12:00:00 AM	33.98333	-1

18854 rows × 10 columns

In [95]:

```
#
meteorites.query("`mass (g)` > 1e6 and fall == 'Fell'" )
```

Out[95]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclon
29	Allende	2278	Valid	CV3	2000000.0	Fell	01/01/1969 12:00:00 AM	26.96667	-105.3166
419	Jilin	12171	Valid	H5	4000000.0	Fell	01/01/1976 12:00:00 AM	44.05000	126.1666
506	Kunya-Urgench	12379	Valid	H5	1100000.0	Fell	01/01/1998 12:00:00 AM	42.25000	59.2000
707	Norton County	17922	Valid	Aubrite	1100000.0	Fell	01/01/1948 12:00:00 AM	39.68333	-99.8666
920	Sikhote-Alin	23593	Valid	Iron, IIAB	23000000.0	Fell	01/01/1947 12:00:00 AM	46.16000	134.6533

In [97]:

```
meteorites.fall.value_counts()
# conts the number of row base on different elements inside the columns
```

Out[97]:

```
fall
Found    44609
Fell      1107
Name: count, dtype: int64
```

In [100…

```
meteorites.value_counts(subset = ["nametype","fall"], normalize = True) # use norma
```

Out[100…

```
nametype  fall
Valid     Found    0.974145
          Fell     0.024215
Relict    Found    0.001641
Name: proportion, dtype: float64
```

In [101…

```
meteorites.value_counts(subset = ["nametype","fall"], normalize = False)
# normalize = false it print the counts of every unique values
```

Out[101…

```
nametype  fall
Valid     Found    44534
          Fell      1107
Relict    Found       75
Name: count, dtype: int64
```

In [103…

```
round(meteorite['mass (g)'].mean(),2)
```

Out[103…

```
22087.0
```

In [105…

```
type(meteorite['mass (g)'].mean())
```

Out[105…

```
numpy.float64
```

In [107…
```
meteorites['mass (g)'].quantile([0.01,0.05,0.5,0.95,0.99])
```

Out[107…
```
0.01        0.44
0.05        1.10
0.50       32.60
0.95     4000.00
0.99    50600.00
Name: mass (g), dtype: float64
```

In [111…
```
meteorites['mass (g)'].median() # get the middle value of the data
```

Out[111…
```
32.6
```

In [112…
```
meteorites['mass (g)'].max() # get the highest value of the column
```

Out[112…
```
60000000.0
```

In [110…
```
meteorites.loc[meteorites['mass (g)'].idxmax()] # the idxmax shows the index of the
```

Out[110…
```
name                            Hoba
id                             11890
nametype                       Valid
recclass                   Iron, IVB
mass (g)                  60000000.0
fall                           Found
year           01/01/1920 12:00:00 AM
reclat                      -19.58333
reclong                      17.91667
GeoLocation     (-19.58333, 17.91667)
Name: 16392, dtype: object
```

In [113…
```
meteorites.recclass.nunique() # shows the number of unique values of recclass colum
```

Out[113…
```
466
```

In [120…
```
meteorites.name.nunique() # same as here in name column
```

Out[120…
```
45716
```

In [116…
```
meteorites.recclass.unique()[:14] #show
```

Out[116…
```
array(['L5', 'H6', 'EH4', 'Acapulcoite', 'L6', 'LL3-6', 'H5', 'L',
       'Diogenite-pm', 'Unknown', 'H4', 'H', 'Iron, IVA', 'CR2-an'],
      dtype=object)
```

In [118…
```
meteorites.describe()
```

Out[118…

	id	mass (g)	reclat	reclong
count	45716.000000	4.558500e+04	38401.000000	38401.000000
mean	26889.735104	1.327808e+04	-39.122580	61.074319
std	16860.683030	5.749889e+05	46.378511	80.647298
min	1.000000	0.000000e+00	-87.366670	-165.433330
25%	12688.750000	7.200000e+00	-76.714240	0.000000
50%	24261.500000	3.260000e+01	-71.500000	35.666670
75%	40656.750000	2.026000e+02	0.000000	157.166670
max	57458.000000	6.000000e+07	81.166670	354.473330

In [121…

```
meteorites.describe(include = 'all')
```

Out[121…

	name	id	nametype	recclass	mass (g)	fall	year	
count	45716	45716.000000	45716	45716	4.558500e+04	45716	45425	3840
unique	45716	NaN	2	466	NaN	2	266	
top	Aachen	NaN	Valid	L6	NaN	Found	01/01/2003 12:00:00 AM	
freq	1	NaN	45641	8285	NaN	44609	3323	
mean	NaN	26889.735104	NaN	NaN	1.327808e+04	NaN	NaN	-39
std	NaN	16860.683030	NaN	NaN	5.749889e+05	NaN	NaN	46
min	NaN	1.000000	NaN	NaN	0.000000e+00	NaN	NaN	-87
25%	NaN	12688.750000	NaN	NaN	7.200000e+00	NaN	NaN	-76
50%	NaN	24261.500000	NaN	NaN	3.260000e+01	NaN	NaN	-71
75%	NaN	40656.750000	NaN	NaN	2.026000e+02	NaN	NaN	0
max	NaN	57458.000000	NaN	NaN	6.000000e+07	NaN	NaN	81

EXERCISE PART1

Using the 2019_Yellow_Taxi_Trip_Data.csv dataset, accomplish the following items and submit a PDF of the notebook:

```
1. Create a DataFrame by reading in the
2019_Yellow_Taxi_Trip_Data.csv file. Examine the first 5 rows.
2. Find the dimensions (number of rows and number of columns) in
the data.
3. Using the data in the 2019_Yellow_Taxi_Trip_Data.csv file,
calculate summary statistics for the fare_amount, tip_amount,
tolls_amount, and total_amount columns.
4. Isolate the fare_amount, tip_amount, tolls_amount, and
total_amount for the longest trip by distance (trip_distance).
```

In [123…

```python
# Create a DataFrame by reading in the 2019_Yellow_Taxi_Trip_Data.csv file. Examine
import pandas as pd
df = pd.read_csv('2019_Yellow_Taxi_Trip_Data.csv')
```

In [153…

```python
# Examine the first 5 rows.
df.head(5)
```

Out[153…

	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 [126…

```python
# Find the dimensions (number of rows and number of columns) in the data.

df.shape
```

Out[126…

```
(10000, 18)
```

In [131…

```python
# Using the data in the 2019_Yellow_Taxi_Trip_Data.csv file, calculate summary stat

df[["fare_amount","tip_amount","tolls_amount","total_amount"]].describe()
```

Out[131…

	fare_amount	tip_amount	tolls_amount	total_amount
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	15.106313	2.634494	0.623447	22.564659
std	13.954762	3.409800	6.437507	19.209255
min	-52.000000	0.000000	-6.120000	-65.920000
25%	7.000000	0.000000	0.000000	12.375000
50%	10.000000	2.000000	0.000000	16.300000
75%	16.000000	3.250000	0.000000	22.880000
max	176.000000	43.000000	612.000000	671.800000

In [152…

```
# Isolate the fare_amount, tip_amount, tolls_amount, and total_amount for the longe
df.loc[df['trip_distance'].idxmax()][["fare_amount","tip_amount","tolls_amount","to
```

Out[152…

```
fare_amount      176.0
tip_amount       18.29
tolls_amount      6.12
total_amount    201.21
Name: 8338, dtype: object
```

REFLECTION

- After doing the activity, I was able to learn the basics of pandas starting from importing the csv into a dataframe into applying statistical analysis to the dataframe. During the early part of the activity some of the tasks are easy to follow and when it comes to last part of the activity for me it is difficult since im starting to familiarize myself with the syntax of the pandas wherein im doing a trial and error in every code inorder to do the tasks. For me the hardest part is the indexing since I always forget the name of the column wherein I need to go back to the output of dataframes just to check the name of the column that i need to use.

In [ ]:

```
```

DATA WRANGLING

In [23]:

```
import pandas as pd

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

In [24]:

```
taxis
```

Out[24]:

	vendorid	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distan
0	2	2019-10-23T16:39:42.000	2019-10-23T17:14:10.000	1	7.
1	1	2019-10-23T16:32:08.000	2019-10-23T16:45:26.000	1	2.
2	2	2019-10-23T16:08:44.000	2019-10-23T16:21:11.000	1	1.
3	2	2019-10-23T16:22:44.000	2019-10-23T16:43:26.000	1	1.
4	2	2019-10-23T16:45:11.000	2019-10-23T16:58:49.000	1	1.
...	...	...	...	...	
9995	1	2019-10-23T17:39:59.000	2019-10-23T17:49:26.000	2	1.
9996	1	2019-10-23T17:53:02.000	2019-10-23T18:00:45.000	1	1.
9997	1	2019-10-23T17:07:16.000	2019-10-23T17:11:35.000	1	0.
9998	1	2019-10-23T17:38:26.000	2019-10-23T17:49:28.000	2	2.
9999	1	2019-10-23T17:22:14.000	2019-10-23T17:52:09.000	1	3.

10000 rows × 18 columns

In [66]:
```
mask = taxis.columns.str.contains('id$|store_and_fwd_flag', regex = True)
columns_to_drop = taxis.columns[mask]
columns_to_drop
# here we save all the columns that has name that contains id or store_and_fwd_flag
```

Out[66]:
```
Index([], dtype='object')
```

In [67]:
```
taxis = taxis.drop(columns = columns_to_drop) # here we remove the columns that we
```

In [12]:
```
taxis
```

Out[12]:

	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	payme
0	2019-10-23T16:39:42.000	2019-10-23T17:14:10.000	1	7.93	
1	2019-10-23T16:32:08.000	2019-10-23T16:45:26.000	1	2.00	
2	2019-10-23T16:08:44.000	2019-10-23T16:21:11.000	1	1.36	
3	2019-10-23T16:22:44.000	2019-10-23T16:43:26.000	1	1.00	
4	2019-10-23T16:45:11.000	2019-10-23T16:58:49.000	1	1.96	
...	...	...	...	...	
9995	2019-10-23T17:39:59.000	2019-10-23T17:49:26.000	2	1.30	
9996	2019-10-23T17:53:02.000	2019-10-23T18:00:45.000	1	1.40	
9997	2019-10-23T17:07:16.000	2019-10-23T17:11:35.000	1	0.70	
9998	2019-10-23T17:38:26.000	2019-10-23T17:49:28.000	2	2.50	
9999	2019-10-23T17:22:14.000	2019-10-23T17:52:09.000	1	3.00	

10000 rows × 13 columns

In [27]:

```
# renaming columns
# inorder for us to rename column we can use the function 'rename' inside the paret
taxis = taxis.rename(
    columns = {
        'tpep_pickup_datetime' : 'pickup',
        'tpep_dropoff_datetime' : 'dropoff'
    }
)

taxis.columns
```

Out[27]:

```
Index(['pickup', 'dropoff', 'passenger_count', 'trip_distance', 'payment_type',
       'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount',
       'improvement_surcharge', 'total_amount', 'congestion_surcharge'],
      dtype='object')
```

In [28]:

```
taxis
```

Out[28]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_a
0	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	1	
2	2019-10-23T16:08:44.000	2019-10-23T16:21:11.000	1	1.36	1	
3	2019-10-23T16:22:44.000	2019-10-23T16:43:26.000	1	1.00	1	
4	2019-10-23T16:45:11.000	2019-10-23T16:58:49.000	1	1.96	1	
...	...	...	...	...	...	
9995	2019-10-23T17:39:59.000	2019-10-23T17:49:26.000	2	1.30	1	
9996	2019-10-23T17:53:02.000	2019-10-23T18:00:45.000	1	1.40	2	
9997	2019-10-23T17:07:16.000	2019-10-23T17:11:35.000	1	0.70	2	
9998	2019-10-23T17:38:26.000	2019-10-23T17:49:28.000	2	2.50	1	
9999	2019-10-23T17:22:14.000	2019-10-23T17:52:09.000	1	3.00	1	

10000 rows × 13 columns

In [68]:
```
taxis[['pickup','dropoff']] = taxis[['pickup','dropoff']].apply(pd.to_datetime)
# here we apply a datatype to our columns pickup and dropoff with data type datetim
```

In [69]:
```
taxis.dtypes # check for the updated data type of our dataframe
```

Out[69]:
```
pickup                    datetime64[ns]
dropoff                   datetime64[ns]
passenger_count                    int64
trip_distance                    float64
payment_type                       int64
fare_amount                      float64
extra                            float64
mta_tax                          float64
tip_amount                       float64
tolls_amount                     float64
improvement_surcharge            float64
total_amount                     float64
congestion_surcharge             float64
elapsed_time             timedelta64[ns]
dtype: object
```

In [72]:
```
taxis
```

Out[72]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra
0	2019-10-23 16:39:42	2019-10-23 17:14:10	1	7.93	1	29.5	1.0
1	2019-10-23 16:32:08	2019-10-23 16:45:26	1	2.00	1	10.5	1.0
2	2019-10-23 16:08:44	2019-10-23 16:21:11	1	1.36	1	9.5	1.0
3	2019-10-23 16:22:44	2019-10-23 16:43:26	1	1.00	1	13.0	1.0
4	2019-10-23 16:45:11	2019-10-23 16:58:49	1	1.96	1	10.5	1.0
...	...	...	...	...	...	...	...
9995	2019-10-23 17:39:59	2019-10-23 17:49:26	2	1.30	1	8.0	3.5
9996	2019-10-23 17:53:02	2019-10-23 18:00:45	1	1.40	2	8.0	3.5
9997	2019-10-23 17:07:16	2019-10-23 17:11:35	1	0.70	2	5.0	1.0
9998	2019-10-23 17:38:26	2019-10-23 17:49:28	2	2.50	1	10.0	1.0
9999	2019-10-23 17:22:14	2019-10-23 17:52:09	1	3.00	1	19.0	3.5

10000 rows × 14 columns

In [35]:
```
taxis['elapsed_time'] = taxis['dropoff'] - taxis['pickup']
```

In [74]:
```
taxis = taxis.assign(
    elapsed_time = lambda x: x.dropoff - x.pickup,
    cost_before_tip = lambda x: x.total_amount - x.tip_amount,
    tip_pct = lambda x: x.tip_amount - x.cost_before_tip,
    fees = lambda x: x.cost_before_tip - x.cost_before_tip,
    avg_speed = lambda x: x.trip_distance.div(x.elapsed_time.dt.total_seconds()/60/
)
```

In [75]:
```
taxis.sort_values(['passenger_count','pickup'],ascending = [False,True]).head()
#
```

Out[75]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra
5997	2019-10-23 15:55:19	2019-10-23 16:08:25	6	1.58	2	10.0	1.0
443	2019-10-23 15:56:59	2019-10-23 16:04:33	6	1.46	2	7.5	1.0
8722	2019-10-23 15:57:33	2019-10-23 16:03:34	6	0.62	1	5.5	1.0
4198	2019-10-23 15:57:38	2019-10-23 16:05:07	6	1.18	1	7.0	1.0
8238	2019-10-23 15:58:31	2019-10-23 16:29:29	6	3.23	2	19.5	1.0

In [37]:
```
taxis.sort_values(['fare_amount','tip_amount'], ascending = [False, True]).head()
```

Out[37]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra
8338	2019-10-23 16:50:53	2019-10-24 15:32:55	1	38.11	1	176.00	0.0
853	2019-10-23 16:07:39	2019-10-23 17:37:05	3	19.09	2	160.00	0.0
4714	2019-10-23 16:33:17	2019-10-23 17:56:49	2	26.30	1	111.75	0.0
9758	2019-10-23 17:20:50	2019-10-23 18:58:16	1	19.50	1	96.00	1.0
3354	2019-10-23 16:23:19	2019-10-23 17:10:00	1	10.01	1	95.00	0.0

In [38]:
```
taxis.sort_values(['fare_amount']).head() #sort the dataframe based on the fare amo
```

Out[38]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra
822	2019-10-23 16:52:52	2019-10-23 16:52:54	3	0.02	3	-52.0	-4.5
7586	2019-10-23 16:52:06	2019-10-23 17:29:50	1	10.86	2	-52.0	-4.5
8804	2019-10-23 16:50:16	2019-10-23 17:06:08	2	0.53	4	-10.5	-1.0
6585	2019-10-23 16:20:03	2019-10-23 16:34:47	1	0.87	3	-10.0	-1.0
2103	2019-10-23 16:41:17	2019-10-23 16:56:35	1	0.85	3	-10.0	-1.0

In [39]:

```
taxis.sort_values(['fare_amount','tip_amount'], ascending = [True, False]).head()
# when we are sorting values
```

Out[39]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra
822	2019-10-23 16:52:52	2019-10-23 16:52:54	3	0.02	3	-52.0	-4.5
7586	2019-10-23 16:52:06	2019-10-23 17:29:50	1	10.86	2	-52.0	-4.5
8804	2019-10-23 16:50:16	2019-10-23 17:06:08	2	0.53	4	-10.5	-1.0
2103	2019-10-23 16:41:17	2019-10-23 16:56:35	1	0.85	3	-10.0	-1.0
6585	2019-10-23 16:20:03	2019-10-23 16:34:47	1	0.87	3	-10.0	-1.0

In [41]:

```
taxis.nlargest(3,'elapsed_time') #the nlargest command takes the number of row of t
```

Out[41]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra
7576	2019-10-23 16:52:51	2019-10-24 16:51:44	1	3.75	1	17.5	1.0
6902	2019-10-23 16:51:42	2019-10-24 16:50:22	1	11.19	2	39.5	1.0
4975	2019-10-23 16:18:51	2019-10-24 16:17:30	1	0.70	2	7.0	1.0

In [42]:

```
taxis.nlargest(3,'trip_distance') # lets try it with largest trip distance
```

Out[42]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra
8338	2019-10-23 16:50:53	2019-10-24 15:32:55	1	38.11	1	176.0	0.0
9965	2019-10-23 17:34:29	2019-10-23 18:48:00	1	37.86	2	52.0	4.5
1656	2019-10-23 16:04:45	2019-10-23 19:11:40	3	37.57	1	52.0	4.5

Exercise 2

Read in the meteorit data from the Meteorite_Landing.csv file, rename the mass (g) column to mass, and drop all the latitude and longitude columns, sort the result by mass in descending order.

In [44]:

```
import pandas as pd

meteorite = pd.read_csv("Meteorite_Landings.csv")

meteorite
```

Out[44]:

	name	id	nametype	recclass	mass (g)	fall	year	reclat
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667
...	...	...	...	...	...	...	...	...
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990 12:00:00 AM	29.03700
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999 12:00:00 AM	13.78333
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939 12:00:00 AM	49.25000
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003 12:00:00 AM	49.78917
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976 12:00:00 AM	33.98333

45716 rows × 10 columns

In [62]:

```
# rename the mass (g) column to mass

meteorite = meteorite.rename(
    columns = {
        'mass (g)' : 'mass'
    }

)
```

In [64]:
```
#  and drop all the latitude and longitude columns
meteor = meteorite.columns.str.contains('lat|long',regex = True)
columns_drop = meteorite.columns[meteor]
```

In [56]:
```
meteorite.drop(columns = columns_drop,inplace =True)
```

In [61]:
```
meteorite
```

Out[61]:

	name	id	nametype	recclass	mass (g)	fall	year	GeoLocatio
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	(50.775 6.08333
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	(56.18333 10.23333
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	(54.21667 -113.0
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	(16.88333 -99.9
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	(-33.16667 -64.95
...	...	...	...	...	...	...	...	.
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990 12:00:00 AM	(29.037 17.0185
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999 12:00:00 AM	(13.78333 8.96667
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939 12:00:00 AM	(49.25 17.66667
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003 12:00:00 AM	(49.78917 41.5046
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976 12:00:00 AM	(33.98333 -115.68333

45716 rows × 8 columns

In [65]:
```
# sort the result by mass in descending order.
meteorite.sort_values(['mass'],ascending = False)
```

Out[65]:

	name	id	nametype	recclass	mass	fall	year	GeoLocati
16392	Hoba	11890	Valid	Iron, IVB	60000000.0	Found	01/01/1920 12:00:00 AM	(-19.5833 17.9166
5373	Cape York	5262	Valid	Iron, IIIAB	58200000.0	Found	01/01/1818 12:00:00 AM	(76.1333 -64.9333
5365	Campo del Cielo	5247	Valid	Iron, IAB-MG	50000000.0	Found	12/22/1575 12:00:00 AM	(-27.4666 -60.5833
5370	Canyon Diablo	5257	Valid	Iron, IAB-MG	30000000.0	Found	01/01/1891 12:00:00 AM	(35.0 -111.0333
3455	Armanty	2335	Valid	Iron, IIIE	28000000.0	Found	01/01/1898 12:00:00 AM	(47.0, 88
...	...	...	...	...	...	...	...	
38282	Wei-hui-fu (a)	24231	Valid	Iron	NaN	Found	01/01/1931 12:00:00 AM	Na
38283	Wei-hui-fu (b)	24232	Valid	Iron	NaN	Found	01/01/1931 12:00:00 AM	Na
38285	Weiyuan	24233	Valid	Mesosiderite	NaN	Found	01/01/1978 12:00:00 AM	(35.2666 104.3166
41472	Yamato 792768	28117	Valid	CM2	NaN	Found	01/01/1979 12:00:00 AM	(-71 35.6666
45698	Zapata County	30393	Valid	Iron	NaN	Found	01/01/1930 12:00:00 AM	(27.0, -99

45716 rows × 8 columns

In [ ]:

In [81]:
```
taxis = taxis.set_index("pickup")
# here instead of having a index of number we set the our index as the pickup colum
# its already sorted that why it output an error
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_6280\1820723686.py in ?()
----> 1 taxis = taxis.set_index("pickup")
      2 # here instead of having a index of number we set the our index as the picku
p column
      3 # its already sorted that why it output an error

C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py in ?(self, keys, dro
p, append, inplace, verify_integrity)
   6118                    if not found:
   6119                        missing.append(col)
   6120
   6121        if missing:
-> 6122            raise KeyError(f"None of {missing} are in the columns")
   6123
   6124        if inplace:
   6125            frame = self

KeyError: "None of ['pickup'] are in the columns"
```

In [77]:
```
taxis
```

Out[77]:

pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra	mta_
2019-10-23 16:39:42	2019-10-23 17:14:10	1	7.93	1	29.5	1.0	
2019-10-23 16:32:08	2019-10-23 16:45:26	1	2.00	1	10.5	1.0	
2019-10-23 16:08:44	2019-10-23 16:21:11	1	1.36	1	9.5	1.0	
2019-10-23 16:22:44	2019-10-23 16:43:26	1	1.00	1	13.0	1.0	
2019-10-23 16:45:11	2019-10-23 16:58:49	1	1.96	1	10.5	1.0	
...	...	...	...	...	...	...	
2019-10-23 17:39:59	2019-10-23 17:49:26	2	1.30	1	8.0	3.5	
2019-10-23 17:53:02	2019-10-23 18:00:45	1	1.40	2	8.0	3.5	
2019-10-23 17:07:16	2019-10-23 17:11:35	1	0.70	2	5.0	1.0	
2019-10-23 17:38:26	2019-10-23 17:49:28	2	2.50	1	10.0	1.0	
2019-10-23 17:22:14	2019-10-23 17:52:09	1	3.00	1	19.0	3.5	

10000 rows × 17 columns

In [78]:
```
taxis.sort_index(inplace = True)
```

In [79]:
```
taxis
```

Out[79]:

pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra	mta_
2019-10-23 07:05:34	2019-10-23 08:03:16	3	14.68	1	50.0	1.0	
2019-10-23 07:48:58	2019-10-23 07:52:09	1	0.67	2	4.5	1.0	
2019-10-23 08:02:09	2019-10-24 07:42:32	1	8.38	1	32.0	1.0	
2019-10-23 08:18:47	2019-10-23 08:36:05	1	2.39	2	12.5	1.0	
2019-10-23 09:27:16	2019-10-23 09:33:13	2	1.11	2	6.0	1.0	
...	...	...	...	...	...	...	
2019-10-24 07:23:52	2019-10-24 08:08:52	1	0.00	1	36.2	0.0	
2019-10-24 07:29:52	2019-10-24 07:33:24	1	0.54	2	4.0	0.0	
2019-10-24 07:58:31	2019-10-24 08:47:05	1	0.00	1	22.2	0.0	
2019-10-24 08:07:45	2019-10-24 08:07:50	2	0.00	2	52.0	0.0	
2019-10-24 08:19:11	2019-10-24 09:00:35	0	13.20	2	42.0	0.0	

10000 rows × 17 columns

In [97]:
```
taxis.sort_index(axis = 0) #arrange the row
```

Out[97]:

pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra	mta_
2019-10-23 07:05:34	2019-10-23 08:03:16	3	14.68	1	50.0	1.0	
2019-10-23 07:48:58	2019-10-23 07:52:09	1	0.67	2	4.5	1.0	
2019-10-23 08:02:09	2019-10-24 07:42:32	1	8.38	1	32.0	1.0	
2019-10-23 08:18:47	2019-10-23 08:36:05	1	2.39	2	12.5	1.0	
2019-10-23 09:27:16	2019-10-23 09:33:13	2	1.11	2	6.0	1.0	
...	...	...	...	...	...	...	
2019-10-24 07:23:52	2019-10-24 08:08:52	1	0.00	1	36.2	0.0	
2019-10-24 07:29:52	2019-10-24 07:33:24	1	0.54	2	4.0	0.0	
2019-10-24 07:58:31	2019-10-24 08:47:05	1	0.00	1	22.2	0.0	
2019-10-24 08:07:45	2019-10-24 08:07:50	2	0.00	2	52.0	0.0	
2019-10-24 08:19:11	2019-10-24 09:00:35	0	13.20	2	42.0	0.0	

10000 rows × 17 columns

In [96]:

```
taxis.sort_index(axis = 1) # axis 1 arranges the dataframe wherein it arranges colu
```

Out[96]:

pickup	avg_speed	congestion_surcharge	cost_before_tip	dropoff	elapsed_time	extra
2019-10-23 07:05:34	15.265165	0.0	51.8	2019-10-23 08:03:16	0 days 00:57:42	1.0
2019-10-23 07:48:58	12.628272	2.5	8.8	2019-10-23 07:52:09	0 days 00:03:11	1.0
2019-10-23 08:02:09	0.353989	2.5	36.3	2019-10-24 07:42:32	0 days 23:40:23	1.0
2019-10-23 08:18:47	8.289017	2.5	16.8	2019-10-23 08:36:05	0 days 00:17:18	1.0
2019-10-23 09:27:16	11.193277	0.0	7.8	2019-10-23 09:33:13	0 days 00:05:57	1.0
...	...	...	...	...	...	...
2019-10-24 07:23:52	0.000000	0.0	37.0	2019-10-24 08:08:52	0 days 00:45:00	0.0
2019-10-24 07:29:52	9.169811	0.0	4.8	2019-10-24 07:33:24	0 days 00:03:32	0.0
2019-10-24 07:58:31	0.000000	0.0	23.0	2019-10-24 08:47:05	0 days 00:48:34	0.0
2019-10-24 08:07:45	0.000000	2.5	55.3	2019-10-24 08:07:50	0 days 00:00:05	0.0
2019-10-24 08:19:11	19.130435	0.0	42.8	2019-10-24 09:00:35	0 days 00:41:24	0.0

10000 rows × 17 columns

In [91]:

```
taxis["2019-10-23 07:45":"2019-10-23 08"] # show the rows base on the given index r
```

Out[91]:

pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra	mta_
2019-10-23 07:48:58	2019-10-23 07:52:09	1	0.67	2	4.5	1.0	
2019-10-23 08:02:09	2019-10-24 07:42:32	1	8.38	1	32.0	1.0	
2019-10-23 08:18:47	2019-10-23 08:36:05	1	2.39	2	12.5	1.0	

In [93]:

```
taxis["2019-10-23":] #show the rows
```

Out[93]:

pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra	mta_
2019-10-23 07:05:34	2019-10-23 08:03:16	3	14.68	1	50.0	1.0	
2019-10-23 07:48:58	2019-10-23 07:52:09	1	0.67	2	4.5	1.0	
2019-10-23 08:02:09	2019-10-24 07:42:32	1	8.38	1	32.0	1.0	
2019-10-23 08:18:47	2019-10-23 08:36:05	1	2.39	2	12.5	1.0	
2019-10-23 09:27:16	2019-10-23 09:33:13	2	1.11	2	6.0	1.0	
...	...	...	...	...	...	...	
2019-10-24 07:23:52	2019-10-24 08:08:52	1	0.00	1	36.2	0.0	
2019-10-24 07:29:52	2019-10-24 07:33:24	1	0.54	2	4.0	0.0	
2019-10-24 07:58:31	2019-10-24 08:47:05	1	0.00	1	22.2	0.0	
2019-10-24 08:07:45	2019-10-24 08:07:50	2	0.00	2	52.0	0.0	
2019-10-24 08:19:11	2019-10-24 09:00:35	0	13.20	2	42.0	0.0	

10000 rows × 17 columns

In [92]:

```
taxis.loc['2019-10-23 08']
```

Out[92]:

pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra	mta_
2019-10-23 08:02:09	2019-10-24 07:42:32	1	8.38	1	32.0	1.0	
2019-10-23 08:18:47	2019-10-23 08:36:05	1	2.39	2	12.5	1.0	

In [95]:
```
taxis.reset_index().head()
```

Out[95]:

	pickup	dropoff	passenger_count	trip_distance	payment_type	fare_amount	extra	m
0	2019-10-23 07:05:34	2019-10-23 08:03:16	3	14.68	1	50.0	1.0	
1	2019-10-23 07:48:58	2019-10-23 07:52:09	1	0.67	2	4.5	1.0	
2	2019-10-23 08:02:09	2019-10-24 07:42:32	1	8.38	1	32.0	1.0	
3	2019-10-23 08:18:47	2019-10-23 08:36:05	1	2.39	2	12.5	1.0	
4	2019-10-23 09:27:16	2019-10-23 09:33:13	2	1.11	2	6.0	1.0	

EXERCISE 3

using the meteorite

In [190…
```
import pandas as pd

df1 = pd.read_csv("Meteorite_Landings.csv")
df1.head()
```

Out[190…

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667	-113.00000
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000

In [191…

```
# update the year column into year only and take the first list only
df1["year"] = df1["year"].str.split().str[0]
```

In [192…

```
df1
```

Out[192…

	name	id	nametype	recclass	mass (g)	fall	year	reclat
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880	50.77500
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951	56.18333
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952	54.21667
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976	16.88333
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902	-33.16667
...	...	...	...	...	...	...	...	...
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990	29.03700
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999	13.78333
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939	49.25000
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003	49.78917
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976	33.98333

45716 rows × 10 columns

In [193…

```
df1["year"] = df1["year"].str.split("/").str[2]
# next we split the text of date again based on "/" and get the index 2 since the y
```

In [194…

```
df1
#check the dataframe
```

Out[194…

	name	id	nametype	recclass	mass (g)	fall	year	reclat	recl
0	Aachen	1	Valid	L5	21.0	Fell	1880	50.77500	6.08
1	Aarhus	2	Valid	H6	720.0	Fell	1951	56.18333	10.23
2	Abee	6	Valid	EH4	107000.0	Fell	1952	54.21667	-113.00
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	1976	16.88333	-99.90
4	Achiras	370	Valid	L6	780.0	Fell	1902	-33.16667	-64.95
...	...	...	...	...	...	...	...	...	
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	1990	29.03700	17.01
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	1999	13.78333	8.96
45713	Zlin	30410	Valid	H4	3.3	Found	1939	49.25000	17.66
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	2003	49.78917	41.50
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	1976	33.98333	-115.68

45716 rows × 10 columns

In [195…

```
df1.isnull().sum()
#check if there are null values check the total number of null values
```

Out[195…

```
name              0
id                0
nametype          0
recclass          0
mass (g)        131
fall              0
year            291
reclat         7315
reclong        7315
GeoLocation    7315
dtype: int64
```

In [196…

```
df1['year'] = df1['year'].fillna(0)
# fill the null values of column year with 0
```

In [197…

```
df1
```

Out[197…

	name	id	nametype	recclass	mass (g)	fall	year	reclat	recl
0	Aachen	1	Valid	L5	21.0	Fell	1880	50.77500	6.08
1	Aarhus	2	Valid	H6	720.0	Fell	1951	56.18333	10.23
2	Abee	6	Valid	EH4	107000.0	Fell	1952	54.21667	-113.00
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	1976	16.88333	-99.90
4	Achiras	370	Valid	L6	780.0	Fell	1902	-33.16667	-64.95
...	...	...	...	...	...	...	...	...	
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	1990	29.03700	17.01
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	1999	13.78333	8.96
45713	Zlin	30410	Valid	H4	3.3	Found	1939	49.25000	17.66
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	2003	49.78917	41.50
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	1976	33.98333	-115.68

45716 rows × 10 columns

In [198…

```
df1.isnull().sum()
# check if there are still a missing values in column year
```

Out[198…

```
name              0
id                0
nametype          0
recclass          0
mass (g)        131
fall              0
year              0
reclat         7315
reclong        7315
GeoLocation    7315
dtype: int64
```

In [199…

```
df1['year'] = df1.year.astype('int64')
# now that we dont have any missing values we change the data type of clumn year in
```

In [200…

```
df1.dtypes
```

Out[200…

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

In [201…

```
#create a new columns indicating whether the meteorite was observed falling before
# here we set  a condition that if the year as lesser 1970 and fall column is Fell
df1['observed_before_1970'] = (df1["year"] < 1970) & (df1['fall'] == 'Fell')
```

In [203…

```
df1.dtypes
```

Out[203…

```
name                     object
id                        int64
nametype                 object
recclass                 object
mass (g)                float64
fall                     object
year                      int64
reclat                  float64
reclong                 float64
GeoLocation              object
observed_before_1970       bool
dtype: object
```

In [204…

```
# set the index to the id column and extract all the rows with IDs between 10,036 a
# we set the id column as our index
df1 = df1.set_index('id')
```

In [205…

```
df1
```

Out[205…

id	name	nametype	recclass	mass (g)	fall	year	reclat	reclong	G
1	Aachen	Valid	L5	21.0	Fell	1880	50.77500	6.08333	
2	Aarhus	Valid	H6	720.0	Fell	1951	56.18333	10.23333	
6	Abee	Valid	EH4	107000.0	Fell	1952	54.21667	-113.00000	
10	Acapulco	Valid	Acapulcoite	1914.0	Fell	1976	16.88333	-99.90000	
370	Achiras	Valid	L6	780.0	Fell	1902	-33.16667	-64.95000	
...	...	...	...	...	...	...	...	...	
31356	Zillah 002	Valid	Eucrite	172.0	Found	1990	29.03700	17.01850	
30409	Zinder	Valid	Pallasite, ungrouped	46.0	Found	1999	13.78333	8.96667	
30410	Zlin	Valid	H4	3.3	Found	1939	49.25000	17.66667	
31357	Zubkovsky	Valid	L6	2167.0	Found	2003	49.78917	41.50460	
30414	Zulu Queen	Valid	L3.7	200.0	Found	1976	33.98333	-115.68333	

45716 rows × 10 columns

In [220…

```
df1 = df1.sort_index()
df1.loc[10036:10040]
# inorder to find the given range of location we first sort the index to arrange it
# then we use loc to locate the desire range of index
```

Out[220…

id	name	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoL
10036	Enigma	Valid	H4	94.0	Found	1967	31.33333	-82.31667	(3 -8
10037	Enon	Valid	Iron, ungrouped	763.0	Found	1883	39.86667	-83.95000	(3
10038	Enshi	Valid	H5	8000.0	Fell	1974	30.30000	109.50000	(30.
10039	Ensisheim	Valid	LL6	127000.0	Fell	1491	47.86667	7.35000	(4

BONUS QUESTION TRY LANG di nalagyan comment

In [ ]:
```
# there a data entry in the year column. Can you find it?
```

In [223…
```
import pandas as pd

df2 = pd.read_csv("Meteorite_Landings.csv")
```

In [208…
```
df2.isnull().sum()
```

Out[208…

```
vendorid                 0
tpep_pickup_datetime     0
tpep_dropoff_datetime    0
passenger_count          0
trip_distance            0
ratecodeid               0
store_and_fwd_flag       0
pulocationid             0
dolocationid             0
payment_type             0
fare_amount              0
extra                    0
mta_tax                  0
tip_amount               0
tolls_amount             0
improvement_surcharge    0
total_amount             0
congestion_surcharge     0
dtype: int64
```

In [224…

```
df2
```

Out[224…

	name	id	nametype	recclass	mass (g)	fall	year	reclat
0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500
1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333
2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333
4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667
...	...	...	...	...	...	...	...	...
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	01/01/1990 12:00:00 AM	29.03700
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	01/01/1999 12:00:00 AM	13.78333
45713	Zlin	30410	Valid	H4	3.3	Found	01/01/1939 12:00:00 AM	49.25000
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	01/01/2003 12:00:00 AM	49.78917
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	01/01/1976 12:00:00 AM	33.98333

45716 rows × 10 columns

In [225…

```
df2['year']
```

Out[225…

```
0        01/01/1880 12:00:00 AM
1        01/01/1951 12:00:00 AM
2        01/01/1952 12:00:00 AM
3        01/01/1976 12:00:00 AM
4        01/01/1902 12:00:00 AM
                   ...
45711    01/01/1990 12:00:00 AM
45712    01/01/1999 12:00:00 AM
45713    01/01/1939 12:00:00 AM
45714    01/01/2003 12:00:00 AM
45715    01/01/1976 12:00:00 AM
Name: year, Length: 45716, dtype: object
```

In [227…

```
df2['year'].unique()
```

Out[227…

```
array(['01/01/1880 12:00:00 AM', '01/01/1951 12:00:00 AM',
       '01/01/1952 12:00:00 AM', '01/01/1976 12:00:00 AM',
       '01/01/1902 12:00:00 AM', '01/01/1919 12:00:00 AM',
       '01/01/1949 12:00:00 AM', '01/01/1814 12:00:00 AM',
       '01/01/1930 12:00:00 AM', '01/01/1920 12:00:00 AM',
       '01/01/1974 12:00:00 AM', '01/01/1925 12:00:00 AM',
       '01/01/1769 12:00:00 AM', '01/01/1838 12:00:00 AM',
       '01/01/1959 12:00:00 AM', '01/01/1981 12:00:00 AM',
       '01/01/1957 12:00:00 AM', '01/01/2001 12:00:00 AM',
       '01/01/1806 12:00:00 AM', '01/01/1766 12:00:00 AM',
       '01/01/2002 12:00:00 AM', '01/01/1835 12:00:00 AM',
       '01/01/1873 12:00:00 AM', '01/01/1860 12:00:00 AM',
       '01/01/1900 12:00:00 AM', '01/01/1883 12:00:00 AM',
       '01/01/1899 12:00:00 AM', '01/01/1969 12:00:00 AM',
       '01/01/2008 12:00:00 AM', '01/01/1977 12:00:00 AM',
       '01/01/1895 12:00:00 AM', '01/01/1898 12:00:00 AM',
       '01/01/1939 12:00:00 AM', nan, '01/01/1822 12:00:00 AM',
       '01/01/1869 12:00:00 AM', '01/01/1942 12:00:00 AM',
       '01/01/1971 12:00:00 AM', '01/01/1984 12:00:00 AM',
       '01/01/1914 12:00:00 AM', '01/01/1803 12:00:00 AM',
       '01/01/1954 12:00:00 AM', '01/01/1932 12:00:00 AM',
       '01/01/1950 12:00:00 AM', '01/01/1805 12:00:00 AM',
       '01/01/2009 12:00:00 AM', '01/01/1923 12:00:00 AM',
       '01/01/1886 12:00:00 AM', '01/01/1896 12:00:00 AM',
       '01/01/1933 12:00:00 AM', '01/01/1945 12:00:00 AM',
       '01/01/1836 12:00:00 AM', '01/01/1865 12:00:00 AM',
       '01/01/1842 12:00:00 AM', '01/01/1858 12:00:00 AM',
       '01/01/1908 12:00:00 AM', '01/01/1855 12:00:00 AM',
       '01/01/1968 12:00:00 AM', '01/01/1938 12:00:00 AM',
       '01/01/1934 12:00:00 AM', '01/01/1929 12:00:00 AM',
       '01/01/1922 12:00:00 AM', '01/01/1907 12:00:00 AM',
       '01/01/1993 12:00:00 AM', '01/01/1871 12:00:00 AM',
       '01/01/1892 12:00:00 AM', '01/01/1913 12:00:00 AM',
       '01/01/1790 12:00:00 AM', '01/01/1704 12:00:00 AM',
       '01/01/1904 12:00:00 AM', '01/01/1910 12:00:00 AM',
       '01/01/1965 12:00:00 AM', '01/01/2006 12:00:00 AM',
       '01/01/1994 12:00:00 AM', '01/01/2012 12:00:00 AM',
       '01/01/1989 12:00:00 AM', '01/01/1916 12:00:00 AM',
       '01/01/1893 12:00:00 AM', '01/01/1961 12:00:00 AM',
       '01/01/1937 12:00:00 AM', '01/01/1798 12:00:00 AM',
       '01/01/2004 12:00:00 AM', '01/01/1943 12:00:00 AM',
       '01/01/1924 12:00:00 AM', '01/01/1811 12:00:00 AM',
       '01/01/1859 12:00:00 AM', '01/01/1921 12:00:00 AM',
       '01/01/1877 12:00:00 AM', '01/01/1940 12:00:00 AM',
       '01/01/1905 12:00:00 AM', '01/01/1827 12:00:00 AM',
       '01/01/1887 12:00:00 AM', '01/01/1999 12:00:00 AM',
       '01/01/1843 12:00:00 AM', '01/01/1796 12:00:00 AM',
       '01/01/1941 12:00:00 AM', '01/01/1906 12:00:00 AM',
       '01/01/1909 12:00:00 AM', '01/01/1833 12:00:00 AM',
       '01/01/1804 12:00:00 AM', '01/01/1962 12:00:00 AM',
       '01/01/1808 12:00:00 AM', '01/01/1894 12:00:00 AM',
       '01/01/1852 12:00:00 AM', '01/01/1812 12:00:00 AM',
       '01/01/1823 12:00:00 AM', '01/01/2003 12:00:00 AM',
       '01/01/2011 12:00:00 AM', '01/01/1847 12:00:00 AM',
       '01/01/1956 12:00:00 AM', '01/01/1960 12:00:00 AM',
       '01/01/1964 12:00:00 AM', '01/01/2007 12:00:00 AM',
```

'01/01/1990 12:00:00 AM', '01/01/1946 12:00:00 AM',
'01/01/1863 12:00:00 AM', '01/01/1861 12:00:00 AM',
'01/01/1870 12:00:00 AM', '01/01/1991 12:00:00 AM',
'01/01/1866 12:00:00 AM', '01/01/1973 12:00:00 AM',
'01/01/1846 12:00:00 AM', '01/01/1874 12:00:00 AM',
'01/01/1791 12:00:00 AM', '01/01/1848 12:00:00 AM',
'01/01/1583 12:00:00 AM', '01/01/1810 12:00:00 AM',
'01/01/1988 12:00:00 AM', '01/01/1840 12:00:00 AM',
'01/01/1998 12:00:00 AM', '01/01/1885 12:00:00 AM',
'01/01/1834 12:00:00 AM', '01/01/1815 12:00:00 AM',
'01/01/1841 12:00:00 AM', '01/01/2013 12:00:00 AM',
'01/01/1901 12:00:00 AM', '01/01/1966 12:00:00 AM',
'01/01/1978 12:00:00 AM', '01/01/1979 12:00:00 AM',
'01/01/1917 12:00:00 AM', '01/01/1890 12:00:00 AM',
'01/01/1844 12:00:00 AM', '01/01/1936 12:00:00 AM',
'01/01/1878 12:00:00 AM', '01/01/1868 12:00:00 AM',
'01/01/1829 12:00:00 AM', '01/01/1897 12:00:00 AM',
'01/01/1911 12:00:00 AM', '01/01/1967 12:00:00 AM',
'01/01/1884 12:00:00 AM', '01/01/1903 12:00:00 AM',
'01/01/1864 12:00:00 AM', '01/01/1995 12:00:00 AM',
'01/01/1970 12:00:00 AM', '01/01/1853 12:00:00 AM',
'01/01/1872 12:00:00 AM', '01/01/1947 12:00:00 AM',
'01/01/1785 12:00:00 AM', '12/24/1399 12:00:00 AM',
'12/23/1491 12:00:00 AM', '01/01/1889 12:00:00 AM',
'01/01/1837 12:00:00 AM', '01/01/1879 12:00:00 AM',
'01/01/1875 12:00:00 AM', '01/01/1996 12:00:00 AM',
'01/01/1944 12:00:00 AM', '01/01/1882 12:00:00 AM',
'01/01/1654 12:00:00 AM', '01/01/1826 12:00:00 AM',
'01/01/2000 12:00:00 AM', '01/01/1918 12:00:00 AM',
'01/01/1881 12:00:00 AM', '01/01/1983 12:00:00 AM',
'01/01/1980 12:00:00 AM', '01/01/1891 12:00:00 AM',
'01/01/1972 12:00:00 AM', '01/01/1982 12:00:00 AM',
'01/01/1851 12:00:00 AM', '01/01/1817 12:00:00 AM',
'01/01/1628 12:00:00 AM', '01/01/1857 12:00:00 AM',
'01/01/1912 12:00:00 AM', '01/01/1825 12:00:00 AM',
'01/01/1963 12:00:00 AM', '01/01/1751 12:00:00 AM',
'01/01/2010 12:00:00 AM', '01/01/1975 12:00:00 AM',
'01/01/1928 12:00:00 AM', '01/01/1926 12:00:00 AM',
'01/01/1621 12:00:00 AM', '01/01/1819 12:00:00 AM',
'01/01/1997 12:00:00 AM', '01/01/1876 12:00:00 AM',
'01/01/1821 12:00:00 AM', '01/01/1955 12:00:00 AM',
'01/01/1850 12:00:00 AM', '01/01/1787 12:00:00 AM',
'01/01/1867 12:00:00 AM', '01/01/1809 12:00:00 AM',
'01/01/1986 12:00:00 AM', '01/01/1931 12:00:00 AM',
'01/01/1985 12:00:00 AM', '01/01/1987 12:00:00 AM',
'01/01/1830 12:00:00 AM', '01/01/1845 12:00:00 AM',
'01/01/1813 12:00:00 AM', '01/01/1854 12:00:00 AM',
'01/01/1839 12:00:00 AM', '01/01/1820 12:00:00 AM',
'01/01/1935 12:00:00 AM', '01/01/1768 12:00:00 AM',
'01/01/1753 12:00:00 AM', '01/01/1927 12:00:00 AM',
'01/01/1948 12:00:00 AM', '01/01/1801 12:00:00 AM',
'01/01/1992 12:00:00 AM', '01/01/1953 12:00:00 AM',
'01/01/1915 12:00:00 AM', '01/01/1862 12:00:00 AM',
'01/01/1632 12:00:00 AM', '01/01/1849 12:00:00 AM',
'01/01/1637 12:00:00 AM', '01/01/1795 12:00:00 AM',
'12/27/0920 12:00:00 AM', '01/01/1750 12:00:00 AM',

```
       '12/28/0860 12:00:00 AM', '01/01/1662 12:00:00 AM',
       '01/01/1741 12:00:00 AM', '01/01/1958 12:00:00 AM',
       '12/22/1519 12:00:00 AM', '01/01/1671 12:00:00 AM',
       '01/01/1856 12:00:00 AM', '01/01/1775 12:00:00 AM',
       '01/01/1779 12:00:00 AM', '01/01/1723 12:00:00 AM',
       '01/01/1740 12:00:00 AM', '01/01/1824 12:00:00 AM',
       '01/01/1828 12:00:00 AM', '12/23/1490 12:00:00 AM',
       '01/01/1636 12:00:00 AM', '01/01/1688 12:00:00 AM',
       '01/01/1715 12:00:00 AM', '01/01/1773 12:00:00 AM',
       '01/01/1818 12:00:00 AM', '01/01/1794 12:00:00 AM',
       '01/01/1647 12:00:00 AM', '01/01/1623 12:00:00 AM',
       '01/01/1807 12:00:00 AM', '01/01/1668 12:00:00 AM',
       '12/23/1495 12:00:00 AM', '01/01/1831 12:00:00 AM',
       '01/01/2005 12:00:00 AM', '01/01/1888 12:00:00 AM',
       '01/01/1784 12:00:00 AM', '12/22/1575 12:00:00 AM',
       '01/01/1793 12:00:00 AM', '01/01/1749 12:00:00 AM',
       '01/01/1781 12:00:00 AM', '01/01/1600 12:00:00 AM',
       '01/01/1970 12:33:23 AM', '01/01/2101 12:00:00 AM',
       '01/01/1797 12:00:00 AM', '01/01/1716 12:00:00 AM',
       '01/01/1724 12:00:00 AM', '01/01/1776 12:00:00 AM',
       '01/01/1832 12:00:00 AM', '01/01/1792 12:00:00 AM'], dtype=object)
```

In [228…

```
df2['year'] = df2['year'].str.split().str[0]
```

In [229…

```
df2['year'] = df2['year'].str.split('/').str[2]
```

In [230…

```
df2
```

Out[230…

	name	id	nametype	recclass	mass (g)	fall	year	reclat	recl
0	Aachen	1	Valid	L5	21.0	Fell	1880	50.77500	6.08
1	Aarhus	2	Valid	H6	720.0	Fell	1951	56.18333	10.23
2	Abee	6	Valid	EH4	107000.0	Fell	1952	54.21667	-113.00
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	1976	16.88333	-99.90
4	Achiras	370	Valid	L6	780.0	Fell	1902	-33.16667	-64.95
...	...	...	...	...	...	...	...	...	
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	1990	29.03700	17.01
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	1999	13.78333	8.96
45713	Zlin	30410	Valid	H4	3.3	Found	1939	49.25000	17.66
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	2003	49.78917	41.50
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	1976	33.98333	-115.68

45716 rows × 10 columns

In [231…

```
copy_m = df2.copy()
```

In [232…

```
copy_m.dropna(subset = ["year"], inplace = True)
```

In [233…

```
copy_m
```

Out[233…

	name	id	nametype	recclass	mass (g)	fall	year	reclat	recl
0	Aachen	1	Valid	L5	21.0	Fell	1880	50.77500	6.08
1	Aarhus	2	Valid	H6	720.0	Fell	1951	56.18333	10.23
2	Abee	6	Valid	EH4	107000.0	Fell	1952	54.21667	-113.00
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	1976	16.88333	-99.90
4	Achiras	370	Valid	L6	780.0	Fell	1902	-33.16667	-64.95
...	...	...	...	...	...	...	...	...	
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	1990	29.03700	17.01
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	1999	13.78333	8.96
45713	Zlin	30410	Valid	H4	3.3	Found	1939	49.25000	17.66
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	2003	49.78917	41.50
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	1976	33.98333	-115.68

45425 rows × 10 columns

In [234…

```
copy_m['year'] = copy_m.year.astype('int64')
```

In [239…

```
copy_m['year'].sort_values()
```

Out[239…

```
704       860
679       920
278      1399
856      1490
283      1491
         ...
30775    2013
30774    2013
30762    2013
30730    2013
30682    2101
Name: year, Length: 45425, dtype: int64
```

In [240…

```
copy_m['year'].sort_values(ascending = False)
```

Out[240…

```
30682    2101
194      2013
30730    2013
30763    2013
30774    2013
         ...
283      1491
856      1490
278      1399
679       920
704       860
Name: year, Length: 45425, dtype: int64
```

In [241…

```
copy_m['year'].nlargest()
```

Out[241…

```
30682    2101
194      2013
30730    2013
30762    2013
30763    2013
Name: year, dtype: int64
```

In [242…

```
copy_m['year'].nsmallest(3)
```

Out[242…

```
704     860
679     920
278    1399
Name: year, dtype: int64
```

In [ ]: