Report of Exam Data Engineering :

Meteorite data recordings ¶

TU Zhekun
01/04/2021
This report is using **Python**, **Markdown** and $\LaTeX$ **inline mathematical expressions**.

For original codes

You will find the origin `.ipynb` file at
https://github.com/TU-Zhekun/Exam_DataEngineering_March_2021/blob/main/notebook/TU_Zhekun_Exam_DataEngineering_March_2021.ipynb (https://github.com/TU-Zhekun/Exam_DataEngineering_March_2021/blob/main/notebook/TU_Zhekun_Exam_DataEngineering_March_2021.ipynb)

For github homepage of this Exam

https://github.com/TU-Zhekun/Exam_DataEngineering_March_2021 (https://github.com/TU-Zhekun/Exam_DataEngineering_March_2021)

This first part is used for myself to have a preliminary understanding of the data set.

In [1]:
```
import pandas as pd
df = pd.read_csv('../data/Meteorite_Landings.csv')
df.head(20) #View first 10 data rows
```

Out[1]:

	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)
5	Adhi Kot	379	Valid	EH4	4239.0	Fell	01/01/1919 12:00:00 AM	32.10000	71.80000	(32.1, 71.8)
6	Adzhi-Bogdo (stone)	390	Valid	LL3-6	910.0	Fell	01/01/1949 12:00:00 AM	44.83333	95.16667	(44.83333, 95.16667)
7	Agen	392	Valid	H5	30000.0	Fell	01/01/1814 12:00:00 AM	44.21667	0.61667	(44.21667, 0.61667)
8	Aguada	398	Valid	L6	1620.0	Fell	01/01/1930 12:00:00 AM	-31.60000	-65.23333	(-31.6, -65.23333)
9	Aguila Blanca	417	Valid	L	1440.0	Fell	01/01/1920 12:00:00 AM	-30.86667	-64.55000	(-30.86667, -64.55)
10	Aioun el Atrouss	423	Valid	Diogenite-pm	1000.0	Fell	01/01/1974 12:00:00 AM	16.39806	-9.57028	(16.39806, -9.57028)
11	Aïr	424	Valid	L6	24000.0	Fell	01/01/1925 12:00:00 AM	19.08333	8.38333	(19.08333, 8.38333)
12	Aire-sur-la-Lys	425	Valid	Unknown	NaN	Fell	01/01/1769 12:00:00 AM	50.66667	2.33333	(50.66667, 2.33333)
13	Akaba	426	Valid	L6	779.0	Fell	01/01/1949 12:00:00 AM	29.51667	35.05000	(29.51667, 35.05)
14	Akbarpur	427	Valid	H4	1800.0	Fell	01/01/1838 12:00:00 AM	29.71667	77.95000	(29.71667, 77.95)
15	Akwanga	432	Valid	H	3000.0	Fell	01/01/1959 12:00:00 AM	8.91667	8.43333	(8.91667, 8.43333)
16	Akyumak	433	Valid	Iron, IVA	50000.0	Fell	01/01/1981 12:00:00 AM	39.91667	42.81667	(39.91667, 42.81667)
17	Al Rais	446	Valid	CR2-an	160.0	Fell	01/01/1957 12:00:00 AM	24.41667	39.51667	(24.41667, 39.51667)
18	Al Zarnkh	447	Valid	LL5	700.0	Fell	01/01/2001 12:00:00 AM	13.66033	28.96000	(13.66033, 28.96)
19	Alais	448	Valid	CI1	6000.0	Fell	01/01/1806 12:00:00 AM	44.11667	4.08333	(44.11667, 4.08333)

In [2]:
```
df.describe()
```

Out[2]:

	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 [3]:
```
df.info()
```

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

Data cleaning

This part is used to wash Data set to make it usable later.

In [17]:
```
df = df.dropna()
df = df.drop(df[df["mass (g)"] == 0].index)
df = df.drop(df[(df["reclat"] == 0) & (df["reclong"] == 0)].index)
df.to_csv('../data/washed_Meteorite_Landings.csv')
```

In [22]:
```
df.head(10)
```

Out[22]:

	Unnamed: 0	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
0	0	Aachen	1	Valid	L5	21.0	Fell	01/01/1880 12:00:00 AM	50.77500	6.08333	(50.775, 6.08333)
1	1	Aarhus	2	Valid	H6	720.0	Fell	01/01/1951 12:00:00 AM	56.18333	10.23333	(56.18333, 10.23333)
2	2	Abee	6	Valid	EH4	107000.0	Fell	01/01/1952 12:00:00 AM	54.21667	-113.00000	(54.21667, -113.0)
3	3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	01/01/1976 12:00:00 AM	16.88333	-99.90000	(16.88333, -99.9)
4	4	Achiras	370	Valid	L6	780.0	Fell	01/01/1902 12:00:00 AM	-33.16667	-64.95000	(-33.16667, -64.95)
5	5	Adhi Kot	379	Valid	EH4	4239.0	Fell	01/01/1919 12:00:00 AM	32.10000	71.80000	(32.1, 71.8)
6	6	Adzhi-Bogdo (stone)	390	Valid	LL3-6	910.0	Fell	01/01/1949 12:00:00 AM	44.83333	95.16667	(44.83333, 95.16667)
7	7	Agen	392	Valid	H5	30000.0	Fell	01/01/1814 12:00:00 AM	44.21667	0.61667	(44.21667, 0.61667)
8	8	Aguada	398	Valid	L6	1620.0	Fell	01/01/1930 12:00:00 AM	-31.60000	-65.23333	(-31.6, -65.23333)
9	9	Aguila Blanca	417	Valid	L	1440.0	Fell	01/01/1920 12:00:00 AM	-30.86667	-64.55000	(-30.86667, -64.55)

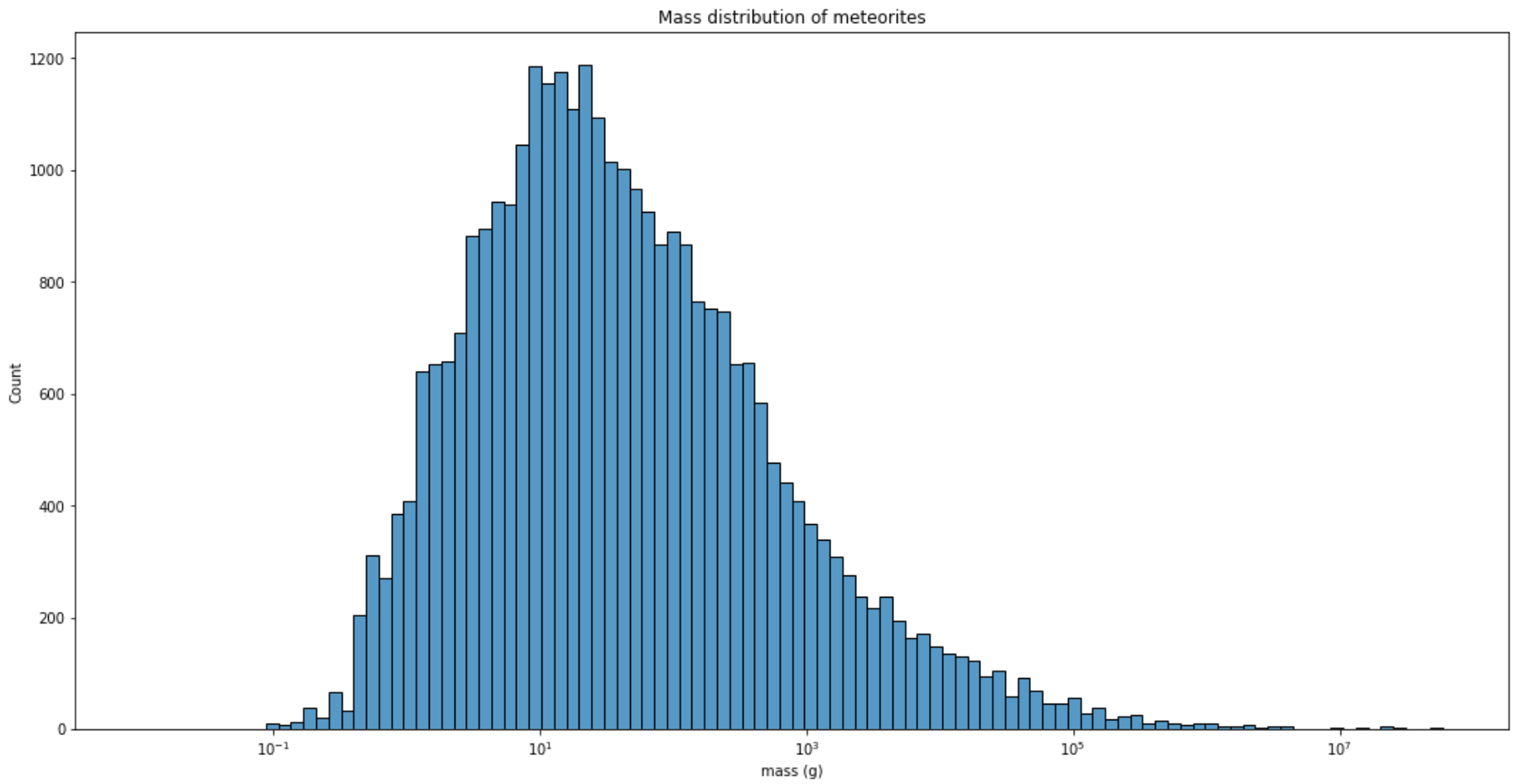
Questions

1. Make an histogram of the mass distribution of meteorites. Do it again for the meteorites having a mass less or equal to 50 000 grams.

In [5]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
df = pd.read_csv('../data/washed_Meteorite_Landings.csv')
```

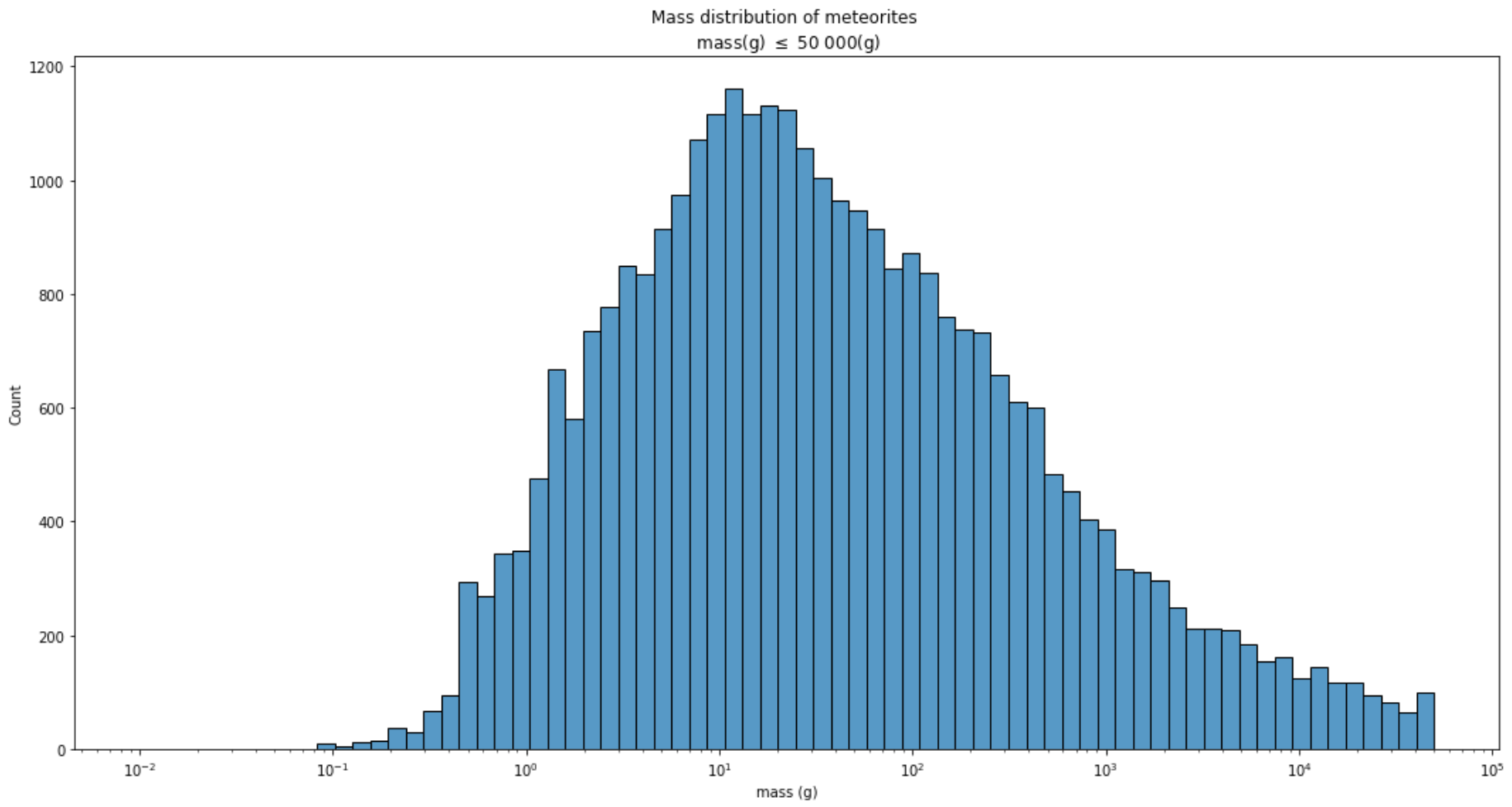
In [6]:
```
plt.figure(figsize=(18,9))
sns.histplot(data=df, x="mass (g)", log_scale=True).set_title("Mass distribution of meteorites")
```

Out[6]: Text(0.5, 1.0, 'Mass distribution of meteorites')



In [7]:
```
plt.figure(figsize=(18,9))
sns.histplot(data=df[df["mass (g)"] <= 50000], x="mass (g)", log_scale=True).set_title("Mass distribution of meteorites \n mass(g) $\leq$ 50 000(g)")
```

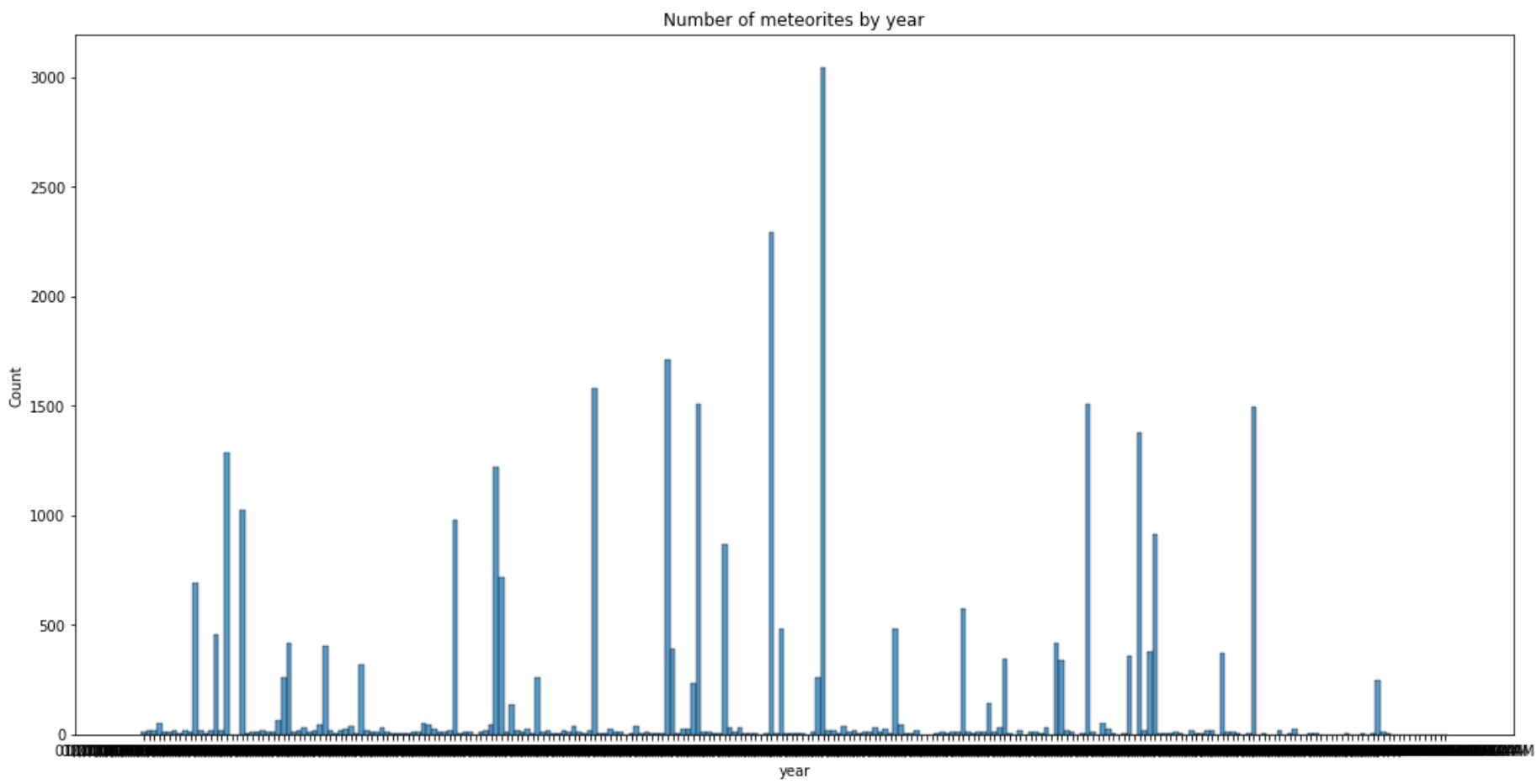
Out[7]: Text(0.5, 1.0, 'Mass distribution of meteorites \n mass(g) $\\leq$ 50 000(g)')



2. Make a plot of the number of meteorites as a function of time (by year).Find a linear $(y = ax + b)$ that approximates the trend of the curve.Using this function say what would be the number of landing meteorites next year. Is this approach of prediction scientifically robust?

In [8]:
```
plt.figure(figsize=(18,9))
sns.histplot(data=df, x="year").set_title("Number of meteorites by year")
```

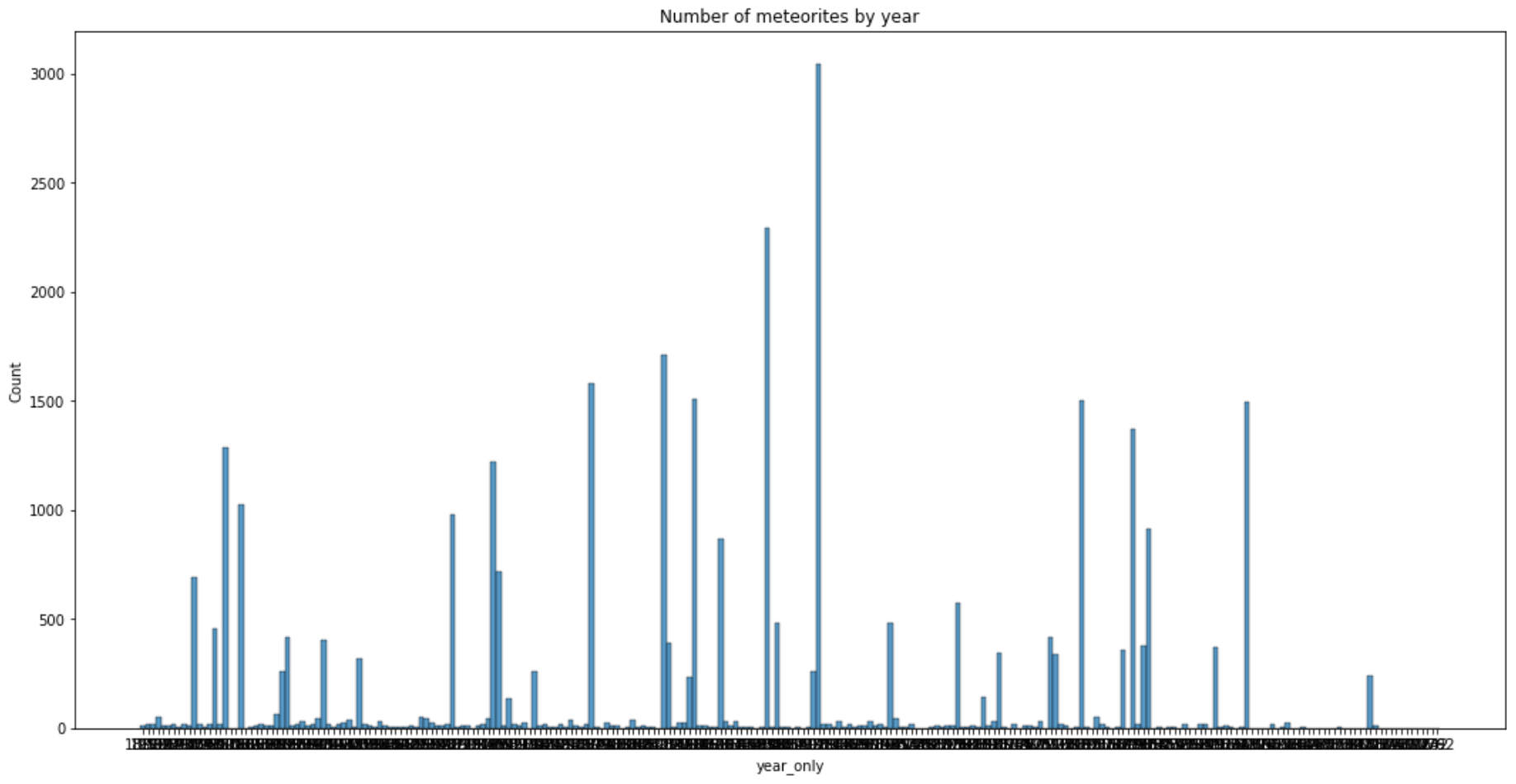
Out[8]: Text(0.5, 1.0, 'Number of meteorites by year')



In [ ]:
```
# insert an indepandent columns which contains only year infomation(not all timestamp)
df.insert(df.shape[1],"year_only", df["year"].str[6:10])
```

In [10]:
```
plt.figure(figsize=(18,9))
sns.histplot(data=df, x="year_only").set_title("Number of meteorites by year")
```

Out[10]: Text(0.5, 1.0, 'Number of meteorites by year')



Here we make a new data frame, which contains only year infomation and the frequency of events in this year.

In [68]:
```
tab_yearCounts = df["year_only"]
tab_yearCounts = pd.DataFrame(tab_yearCounts.value_counts())
tab_yearCounts.to_csv('../data/year_counts.csv')
tab_yearCounts = pd.read_csv('../data/year_counts.csv')
tab_yearCounts.columns = ["year","counts"]
tab_yearCounts.to_csv('../data/year_counts.csv')
```

Out[68]:

	year	counts
0	1979	3044
1	1988	2295
2	2003	1713
3	1999	1578
4	1990	1506
...	...	...
247	2013	1
248	1491	1
249	1628	1
250	1775	1
251	1399	1

252 rows × 2 columns

In [11]:
```
#reforme the dataframe
tab_yearCounts = pd.read_csv('../data/year_counts.csv')
tab_yearCounts.drop(columns=["Unnamed: 0"], inplace=True)
tab_yearCounts
```

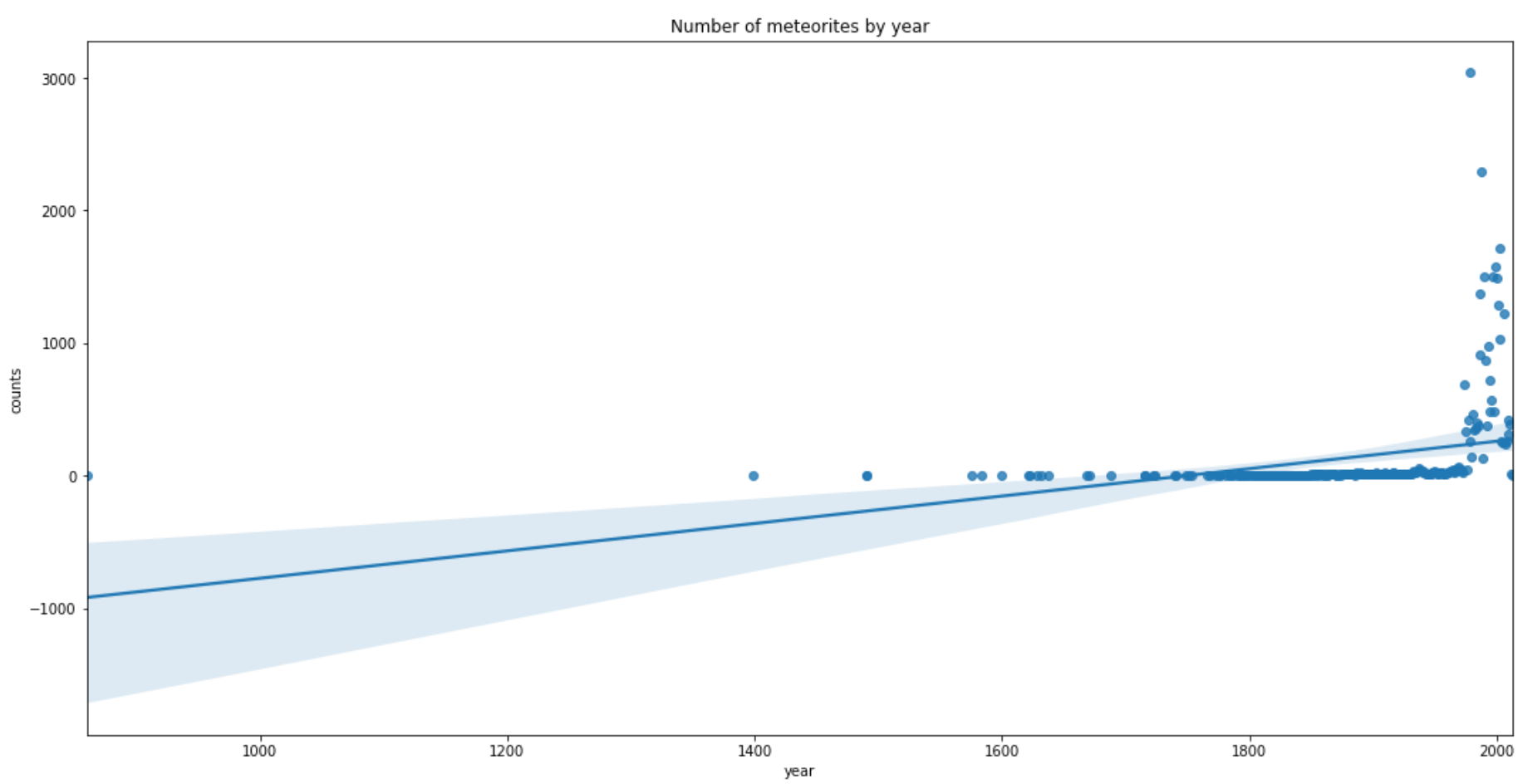
Out[11]:

	year	counts
0	1979	3044
1	1988	2295
2	2003	1713
3	1999	1578
4	1990	1506
...	...	...
247	2013	1
248	1491	1
249	1628	1
250	1775	1
251	1399	1

252 rows × 2 columns

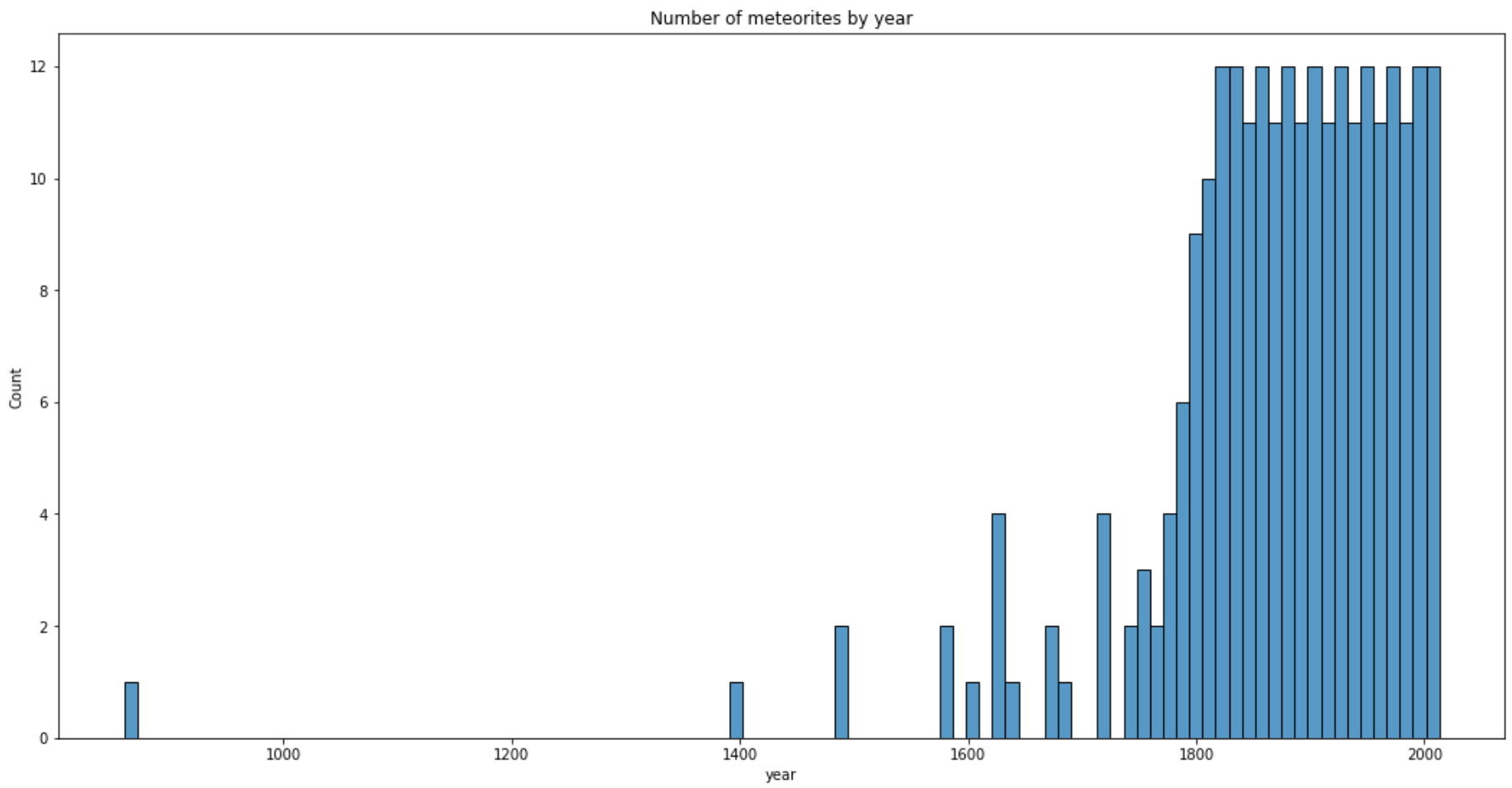
In [12]:
```
plt.figure(figsize=(18,9))
sns.regplot(data=tab_yearCounts, x="year", y="counts" ).set_title("Number of meteorites by year")
```

Out[12]: Text(0.5, 1.0, 'Number of meteorites by year')



In [13]:
```
plt.figure(figsize=(18,9))
sns.histplot(data=tab_yearCounts, x="year", bins = 100).set_title("Number of meteorites by year")
```

Out[13]: Text(0.5, 1.0, 'Number of meteorites by year')



Find a linear fit (y = ax + b) that approximates the trend of the curve.

In [2]:
```
from sklearn import datasets
from sklearn.linear_model import LinearRegression
```

In [79]:
```
x_train = tab_yearCounts["year"].values.reshape(-1, 1)
y_train = tab_yearCounts["counts"].values.reshape(-1, 1)
```

In [37]:
```
model = LinearRegression()
model.fit(x_train,y_train)
a = model.coef_
b = model.intercept_
```

In [77]:
```
print("Y="+str(a[0,0])+"*X"+str(b[0]))
```

Y=1.0330193660211613*X-1807.0050316331262

Finally, the expression of our linear fit is as below:

$$Y = 1.0330193660211613 \times X - 1807.0050316331262$$

Using this function say what would be the number of landing meteorites next year.

In [82]:
```
model.predict([[2022]])
```

Out[82]: array([[281.76012646]])

Our model predict that the number of landing meteorites next year(year 2022) will be:

$281.76012646 \approx 282$

3. We will concentrate now in the case of Oman. Create a plot of this country with different points representing the spatial distribution of the landing sites.

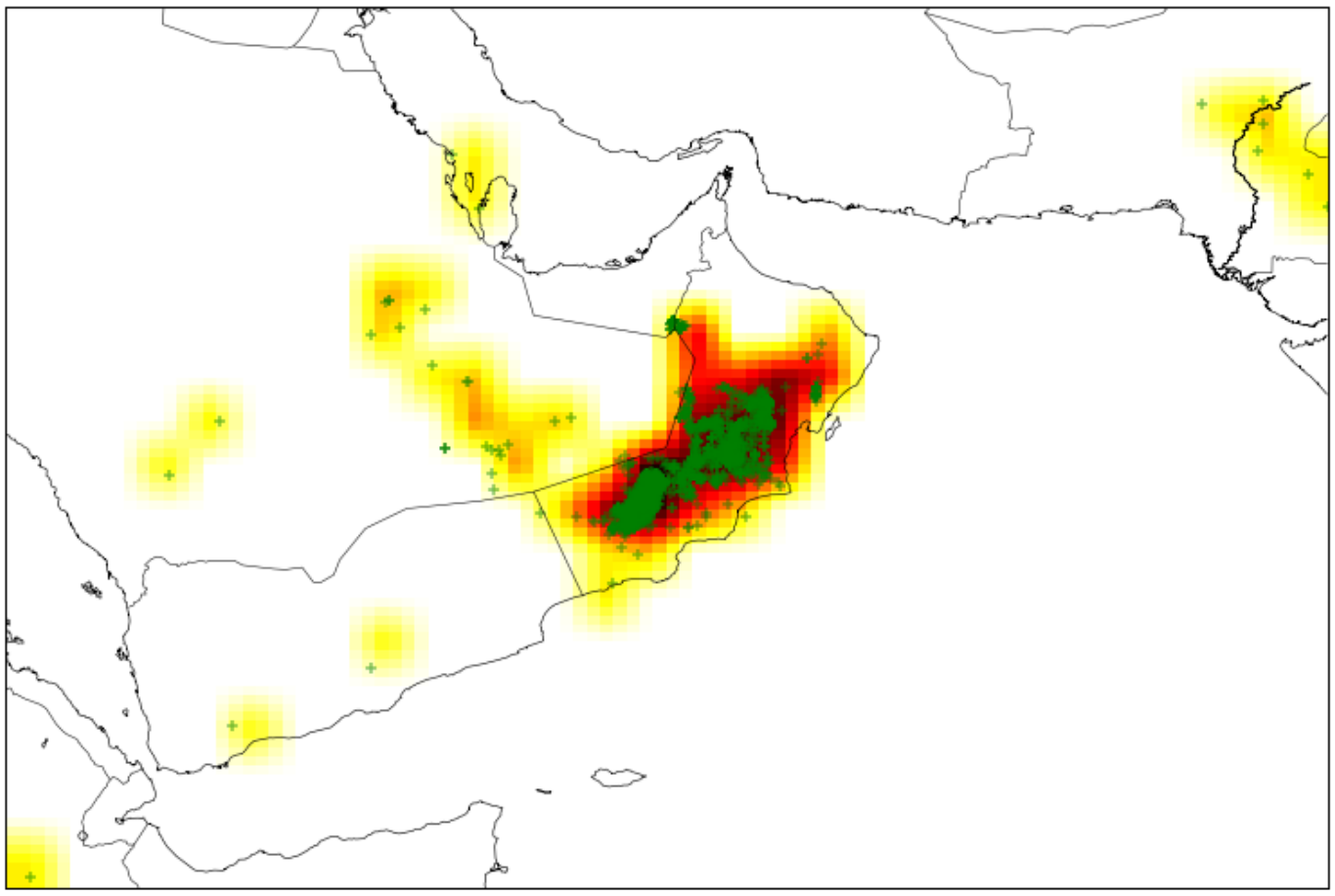
In [14]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from mpl_toolkits.basemap import Basemap
```

In [15]:
```
#We can easily focus on a specific area of the map, let's look at Oman :
plt.figure(figsize=(18,9))
# map = Basemap(projection='cyl',llcrnrlat=10.0,urcrnrlat=30.0,llcrnrlon=40.0,urcrnrlon=70.0,resolution='i')
map = Basemap(projection='cyl',
              llcrnrlat=10.0,
              urcrnrlat=30.0,
              llcrnrlon=40.0,
              urcrnrlon=70.0,
              resolution='i')
map.drawmapboundary(fill_color='w')
map.drawcoastlines(linewidth=0.6)
map.drawcountries()

h = plt.hist2d(df.reclong,df.reclat,bins=(np.arange(40,71,1.0),np.arange(10,31,1)))
X,Y = np.meshgrid(h[1][:-1]+0.5,h[2][:-1]+0.5)
data_interp, x, y = map.transform_scalar(np.log10(h[0].T+0.1), X[0], Y[:,0], 101, 101, returnxy=True)
map.pcolormesh(x, y, data_interp,cmap='hot_r')
map.scatter(df.reclong,df.reclat,marker='+', alpha=0.5, edgecolor='None', color='g')
```

```
d:\codewriting\softwares\python\lib\site-packages\ipykernel_launcher.py:17: MatplotlibDeprecationWarning: shading='flat' when X and Y have the same dimensions as C is deprecated since 3.3.  Either specify the corners of the quadrilaterals with X and Y, or pass shading='auto', 'neares
t' or 'gouraud', or set rcParams['pcolor.shading'].  This will become an error two minor releases later.
d:\codewriting\softwares\python\lib\site-packages\ipykernel_launcher.py:18: UserWarning: You passed a edgecolor/edgecolors ('None') for an unf
illed marker ('+').  Matplotlib is ignoring the edgecolor in favor of the facecolor.  This behavior may change in the future.
```

Out[15]: <matplotlib.collections.PathCollection at 0x2555d1b84c8>

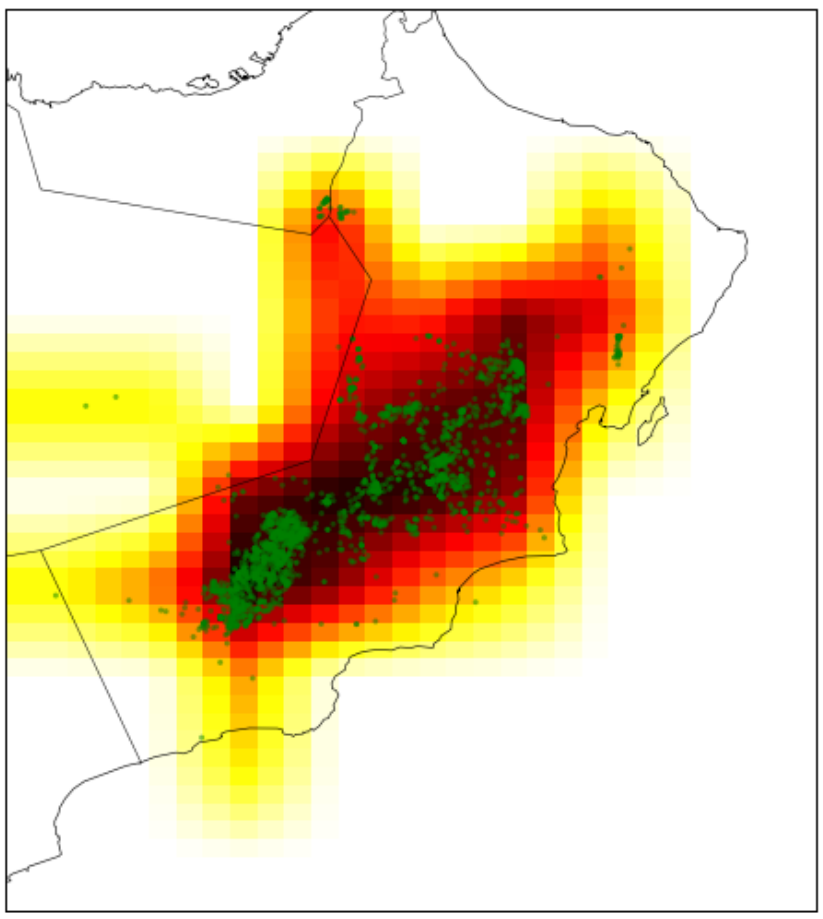


In [16]:
```
#We can easily focus on a specific area of the map, let's look at Oman :
plt.figure(figsize=(18,9))
# map = Basemap(projection='cyl',llcrnrlat=10.0,urcrnrlat=30.0,llcrnrlon=40.0,urcrnrlon=70.0,resolution='i')
map = Basemap(projection='cyl',
              llcrnrlat=10.0,
              urcrnrlat=30.0,
              llcrnrlon=40.0,
              urcrnrlon=70.0,
              resolution='i')
map.drawmapboundary(fill_color='w')
map.drawcoastlines(linewidth=0.6)
map.drawcountries()

h = plt.hist2d(df.reclong,df.reclat,bins=(np.arange(51.6,61,1.0),np.arange(15,26,1)))
X,Y = np.meshgrid(h[1][:-1]+0.5,h[2][:-1]+0.5)
data_interp, x, y = map.transform_scalar(np.log10(h[0].T+0.1), X[0], Y[:,0], 101, 101, returnxy=True)
map.pcolormesh(x, y, data_interp,cmap='hot_r')
map.scatter(df.reclong,df.reclat,marker='.', alpha=0.5, edgecolor='None', color='g')
```

```
d:\codewriting\softwares\python\lib\site-packages\ipykernel_launcher.py:17: MatplotlibDeprecationWarning: shading='flat' when X and Y have the same dimensions as C is deprecated since 3.3.  Either specify the corners of the quadrilaterals with X and Y, or pass shading='auto', 'neares
t' or 'gouraud', or set rcParams['pcolor.shading'].  This will become an error two minor releases later.
```

Out[16]: <matplotlib.collections.PathCollection at 0x25562b3dbc8>



Filter out the data in the oman range based on the latitude and longitude range introduced above

In [17]:
```
oman = df[(df["reclong"]>51.6)&(df["reclong"]<61)]
```

In [18]:
```
oman = oman[(oman["reclat"]>15)&(oman["reclat"]<26)]
oman
```

Out[18]:

	Unnamed: 0	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation	year_only
1516	1516	Al Huqf 001	434	Valid	L4	41.50	Found	01/01/2000 12:00:00 AM	19.42000	57.18833	(19.42, 57.18833)	2000
1517	1517	Al Huqf 002	435	Valid	H5	81.40	Found	01/01/2000 12:00:00 AM	19.27167	57.38667	(19.27167, 57.38667)	2000
1518	1518	Al Huqf 003	436	Valid	L4	115.42	Found	01/01/2002 12:00:00 AM	19.50695	57.11952	(19.50695, 57.11952)	2002
1519	1519	Al Huqf 004	437	Valid	L5	260.22	Found	01/01/2002 12:00:00 AM	19.54482	57.09807	(19.54482, 57.09807)	2002
1520	1520	Al Huqf 005	438	Valid	L6	763.29	Found	01/01/2002 12:00:00 AM	19.84057	57.00848	(19.84057, 57.00848)	2002
...	...	...	...	...	...	...	...	...	...	...	...	...
26551	26551	United Arab Emirates 026	51048	Valid	L6	97.00	Found	01/01/2009 12:00:00 AM	22.89183	55.18142	(22.89183, 55.18142)	2009
26552	26552	United Arab Emirates 027	51049	Valid	L6	107.00	Found	01/01/2009 12:00:00 AM	22.89406	55.17050	(22.89406, 55.1705)	2009
26553	26553	United Arab Emirates 028	51050	Valid	L6	39.00	Found	01/01/2009 12:00:00 AM	22.89092	55.17989	(22.89092, 55.17989)	2009
26558	26558	Uruq al Hadd 001	24127	Valid	LL5	2335.64	Found	01/01/2003 12:00:00 AM	18.44432	52.97805	(18.44432, 52.97805)	2003
26559	26559	Uruq al Hadd 002	24128	Valid	H3	3700.00	Found	01/01/1996 12:00:00 AM	18.50000	52.16667	(18.5, 52.16667)	1996

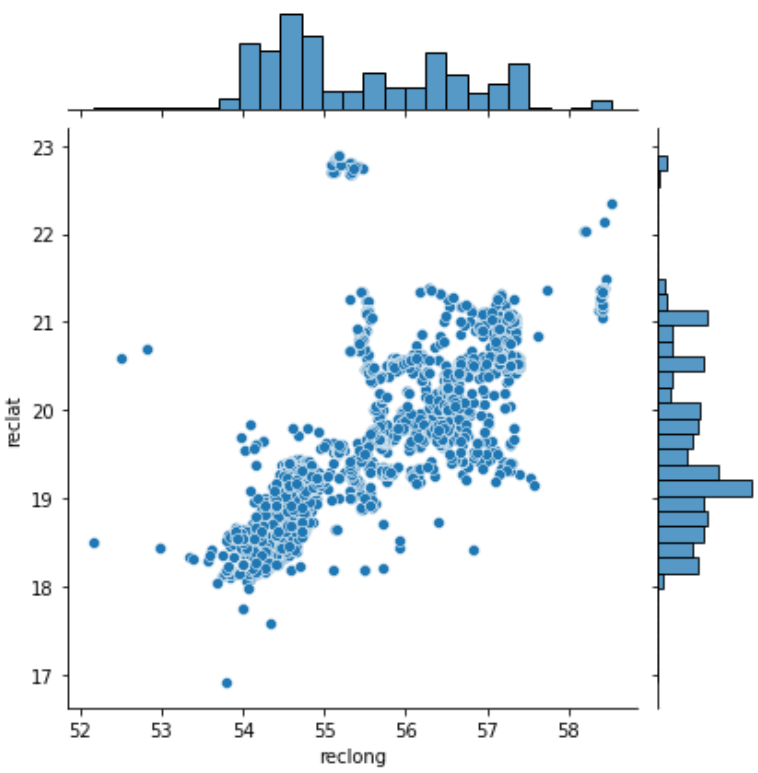
3026 rows × 12 columns

I will now use seaborn.jointplot to help me analyse the distribution.

In [19]:
```
plt.figure(figsize=(18,9))
sns.jointplot(data=oman, x="reclong",y="reclat",kind="scatter")
```

Out[19]:
```
<seaborn.axisgrid.JointGrid at 0x25562e39748>

<Figure size 1296x648 with 0 Axes>
```

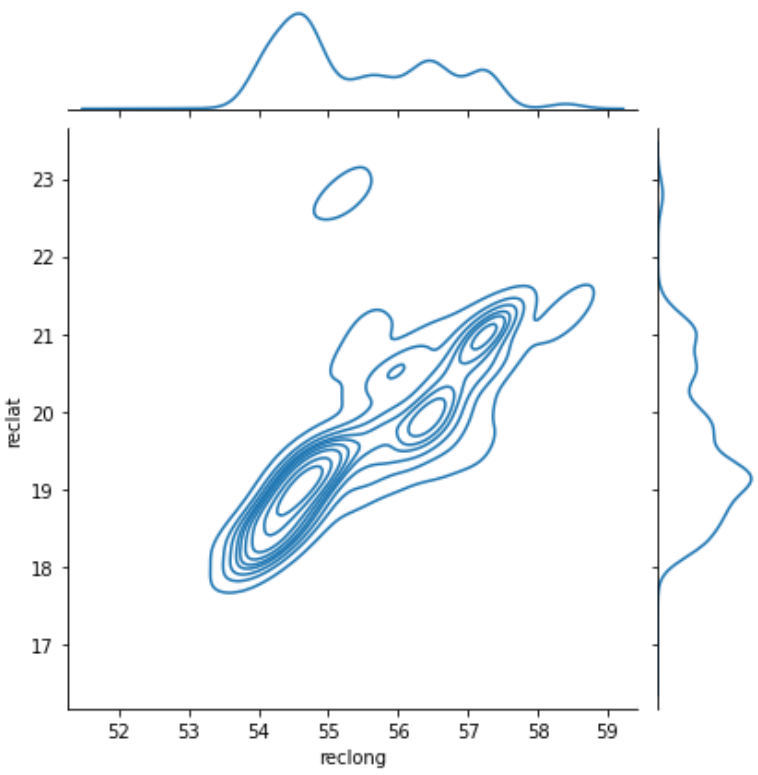


Use KernelDensity plot to make it chear for me to observe the distribution.

In [20]:
```
plt.figure(figsize=(18,9))
sns.jointplot(data=oman, x="reclong",y="reclat",kind="kde")
```

Out[20]:
```
<seaborn.axisgrid.JointGrid at 0x25560697108>

<Figure size 1296x648 with 0 Axes>
```



4. Propose a distribution (uniform, gaussian, cobinaison of different ones...) in order to describe the distribution of meteorite landing sites in Oman.

Due to the previous plot, we will use the **gaussian distribution** to describe meteorite landing sites in Oman.

5. Based on that distribution compute the probability that a meteorite land in the circle of center $\{latitude, longitude\} = \{18.9644, 53.9555\}$ and radius equals to 100 Kilometers.

I am quite confused in the question 5 and I do not think I found the right answer. But since I do not have anymore time, I'll just show you what I have done here.

In [23]:
```
x_train_oman = oman[["reclat","reclong"]].values
x_train_oman
```

Out[23]:
```
array([[19.42   , 57.18833],
       [19.27167, 57.38667],
       [19.50695, 57.11952],
       ...,
       [22.89092, 55.17989],
       [18.44432, 52.97805],
       [18.5    , 52.16667]])
```

In [28]:
```
from sklearn.neighbors.kde import KernelDensity
kde = KernelDensity(kernel='gaussian', bandwidth=0.1).fit(x_train_oman)
```

In [29]:
```
kde.score_samples([[18.9644, 53.9555]])
```

Out[29]:
```
array([-5.36877362])
```

In [31]:
```
kde.score([[17.9644, 52.9555],[19.9644, 54.9555]])
```

Out[31]:
```
-23.564696239663387
```

The probability that a meteorite land in the circle of center $\{latitude, longitude\} = \{18.9644, 53.9555\}$and radius equals to 100 Kilometers.

$\mathrm{P}(\{latitude, longitude\} = \{18.9644, 53.9555\}) = log_e(-23.56)$