Step 1: Importing Necessary Libraries

We begin by importing Python libraries commonly used in data analysis and visualization:

- `numpy` for numerical operations
- `matplotlib.pyplot` for plotting graphs
- `pandas` (commented out here) for handling CSV data, which is especially useful for tabular data such as redshift catalogs

> Tip: If you haven't used `pandas` before, it's worth learning as it offers powerful tools to manipulate and analyze structured datasets.

For reading big csv files, one can use numpy as well as something called "pandas". We suggest to read pandas for CSV file reading and use that

In [32]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from astropy.constants import G, c
from astropy.cosmology import Planck18 as cosmo
import astropy.units as u
import json
```

Before we begin calculations, we define key physical constants used throughout:

- H_0: Hubble constant, describes the expansion rate of the Universe.
- c : Speed of light.
- G: Gravitational constant.
- q_0 : Deceleration parameter, used for approximate co-moving distance calculations.

We will use **`astropy.constants`** to ensure unit consistency and precision.

In [33]:
```
# Constants:

H_0 =  cosmo.H(0) # Hubble constant in SI
c  # Speed of light in m/s
c = c.to(u.km/u.s) #converting to km/s
G  # Gravitational constant in pc kg^-1 (m/s)^2
q0=-0.534  # Deceleration parameter (assumed from Planck fit KEEP it as it is)
```

Read the csv data into the python using the method below

In [34]:
```
#the .csv file downnloaded loses the digits after the 15th digit, hence I used th
with open(r"F:\Skyserver_SQL6_21_2025 2_45_47 PM.json") as f:
    df = pd.DataFrame(json.load(f)[0]["Rows"])

#counting unique objects
print("Unique objids:", df['objid'].nunique())
```

```
Unique objids: 92
```

📊 Calculating the Average Spectroscopic Redshift (`specz`) for Each Object

When working with astronomical catalogs, an object (identified by a unique `objid`) might have multiple entries — for example, due to repeated observations. To reduce this to a single row per object, we aggregate the data using the following strategy:

```
averaged_df = df.groupby('objid').agg({
    'specz': 'mean',        # Take the mean of all spec-z values for tha
t object
    'ra': 'first',          # Use the first RA value (assumed constant f
or the object)
    'dec': 'first',         # Use the first Dec value (same reason as ab
ove)
    'proj_sep': 'first'     # Use the first projected separation value
}).reset_index()
```

In [35]:
```
# Calculating the average specz for each id:
averaged_df = df.groupby('objid').agg({'specz': 'mean',
                                       'ra': 'first',
                                       'dec': 'first',
                                       'proj_sep': 'first',
                                      }).reset_index()
averaged_df.describe()['specz']
```

Out[35]:
```
count    92.000000
mean      0.080838
std       0.008578
min       0.069976
25%       0.077224
50%       0.080961
75%       0.082797
max       0.150886
Name: specz, dtype: float64
```

To create a cut in the redshift so that a cluster can be identified. We must use some logic. Most astronomers prefer anything beyond 3*sigma away from the mean to be not part of the same group.

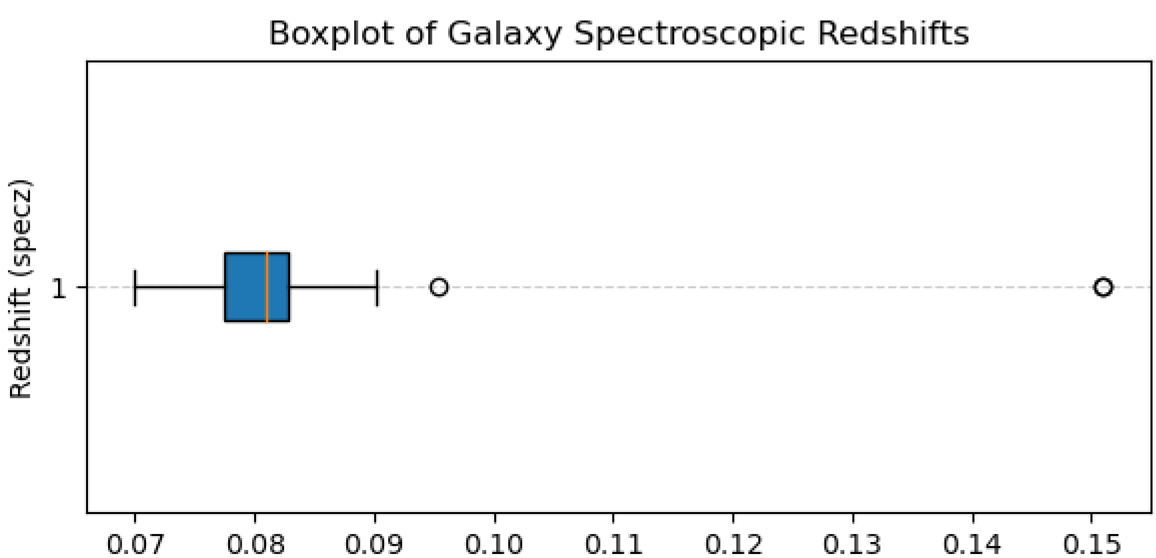
Find the mean, standard deviation and limits of the redshift from the data

In [36]:
```
specz = df['specz'].dropna()
mean = specz.mean()
std = specz.std()
min_val = specz.min()
max_val = specz.max()
lower_limit = mean - 3*std
upper_limit = mean + 3*std

# Print results
print(f"Mean redshift: {mean}")
print(f"Standard Deviation: {std}")
print(f"Min redshift: {min_val}")
print(f"Max redshift: {max_val}")
print(f"Limits: [{lower_limit}, {upper_limit}]")
```

```
Mean redshift: 0.08104694625899281
Standard Deviation: 0.009497709534680291
Min redshift: 0.06997444
Max redshift: 0.15091
Limits: [0.052553817654951936, 0.10954007486303369]
```

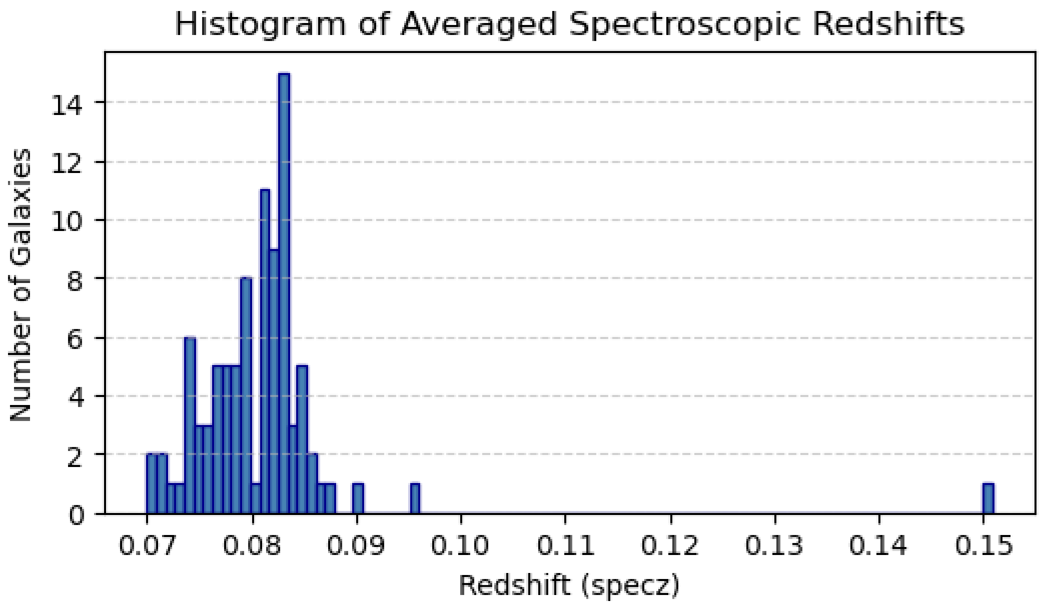
You can also use boxplot to visualize the overall values of redshift

In [37]:
```
# Plot the dsitribution of redshift as histogram and a boxplot

plt.figure(figsize=(6, 3))
plt.boxplot(df['specz'], vert=False, patch_artist=True)
plt.title("Boxplot of Galaxy Spectroscopic Redshifts", fontsize=12)
plt.ylabel("Redshift (specz)", fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.6)

plt.tight_layout()
plt.show()
```



But the best plot would be a histogram to see where most of the objects downloaded lie in terms of redshift value

In [38]:
```
# histogram of averaged redshifts
plt.figure(figsize=(6, 3))
plt.hist(averaged_df['specz'], bins=90, color='steelblue', edgecolor='darkblue')
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.xlabel("Redshift (specz)", fontsize=10)
plt.ylabel("Number of Galaxies", fontsize=10)
plt.title("Histogram of Averaged Spectroscopic Redshifts", fontsize=12)
plt.show()
```



Filter your data based on the 3-sigma limit of redshift. You should remove all data points which are 3-sigma away from mean of redshift

In [39]:
```
# Filtering the data based on specz values, used 3 sigma deviation from mean as u
# Limits are already calculated

filtered_df = averaged_df[
    (averaged_df['specz'] >= lower_limit) &
    (averaged_df['specz'] <= upper_limit)
]
```

Use the relation between redshift and velocity to add a column named velocity in the data. This would tell the expansion velocity at that redshift

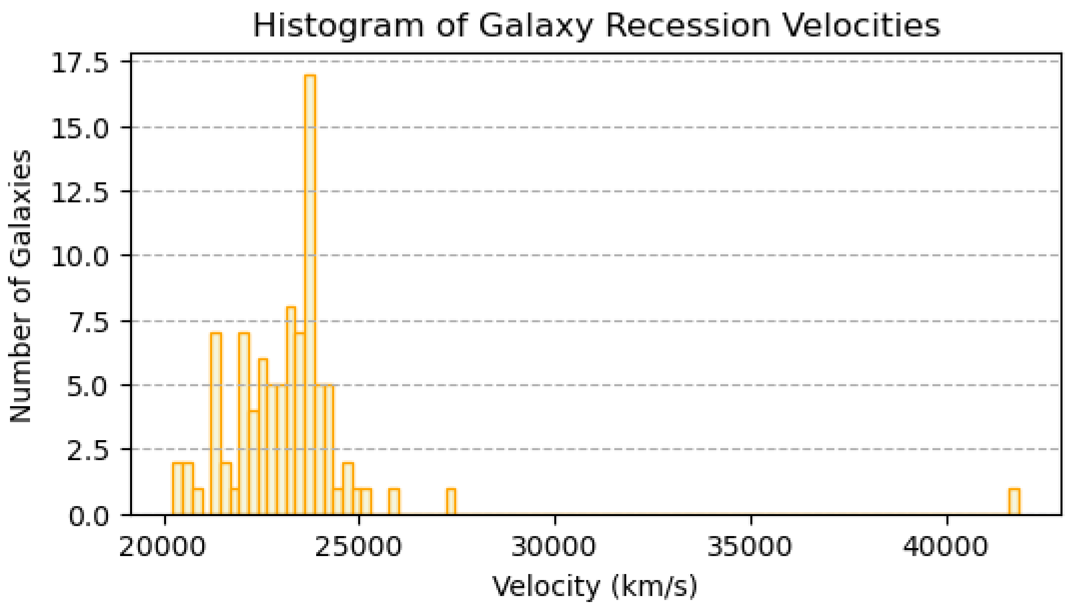
In [40]:
```
# Using the relativistic formula for each redshift
averaged_df['velocity'] = c * ((1 + averaged_df['specz'])**2 - 1) / ((1 + average

# Preview the updated DataFrame
print(averaged_df[['specz', 'velocity']].head())
```

```
      specz      velocity
0  0.082457  23703.959988
1  0.081218  23362.893831
2  0.079564  22906.662584
3  0.080842  23259.086161
4  0.084575  24286.386423
```

In [41]:
```
print(averaged_df['specz'].describe())
```

```
count    92.000000
mean      0.080838
std       0.008578
min       0.069976
25%       0.077224
50%       0.080961
75%       0.082797
max       0.150886
Name: specz, dtype: float64
```

In [42]:
```python
#plot the velocity column created as hist

# Plot the histogram of galaxy velocities
plt.figure(figsize=(6, 3))
plt.hist(averaged_df['velocity'], bins=90, color='beige', edgecolor='orange')

# Add labels and title
plt.title("Histogram of Galaxy Recession Velocities", fontsize=12)
plt.xlabel("Velocity (km/s)", fontsize=10)
plt.ylabel("Number of Galaxies", fontsize=10)
plt.grid(axis='y', linestyle='--')
plt.show()
```



use the dispersion equation to find something called velocity dispersion. You can even refer to wikipedia to know about the term [wiki link here (https://en.wikipedia.org/wiki/Velocity_dispersion#:~:text=In%20astronomy%2C%20the%20velocity%

It is the velocity dispersion value which tells us, some galaxies might be part of even larger groups!!

Step 2: Calculate Mean Redshift of the Cluster

We calculate the average redshift (`specz`) of galaxies that belong to a cluster. This gives us an estimate of the cluster's systemic redshift.

```
cluster_redshift = filtered_df['specz'].mean()
```

The velocity dispersion (v) of galaxies relative to the cluster mean redshift is computed using the relativistic Doppler formula:

$$v = c \cdot \frac{(1+z)^2 - (1+z_{\text{cluster}})^2}{(1+z)^2 + (1+z_{\text{cluster}})^2}$$

where:

- (v) is the relative velocity (dispersion),
- (z) is the redshift of the individual galaxy,
- (z_{cluster}) is the mean cluster redshift,
- (c) is the speed of light.

In [43]:
```
# Mean redshift of the cluster
z_cluster = filtered_df['specz'].mean()

# Relative velocities using relativistic Doppler formula
z = filtered_df['specz']
v_rel = c * (((1 + z)**2 - (1 + z_cluster)**2) / ((1 + z)**2 + (1 + z_cluster)**2

# Adding column and calculating velocity dispersion (std dev)
filtered_df = filtered_df.copy() # we are creating another dataframe to modify ar
filtered_df['v_rel'] = v_rel
velocity_dispersion = v_rel.std()

print(f"Cluster mean redshift: {z_cluster:.6f}")
print(f"Velocity dispersion of cluster: {velocity_dispersion:.2f} km/s")
```

```
Cluster mean redshift: 0.080068
Velocity dispersion of cluster: 1218.49 km/s
```

Pro tip: Check what the describe function of pandas does. Does it help to get quick look stats for your column of dispersion??

In [ ]:
```
# Yes, it gives the statistical values of the column in a csv file or the datafrc
```

In [44]:
```
cluster_redshift = z_cluster
disp = velocity_dispersion
print(f"The value of the cluster redshift = {cluster_redshift:.4}")
print(f"The characteristic value of velocity dispersion of the cluster along the
```

```
The value of the cluster redshift = 0.08007
The characteristic value of velocity dispersion of the cluster along the line o
f sight = 1.218e+03 km/s.
```

In [45]:
```python
from astropy.coordinates import SkyCoord

# Reference taken from <https://astronomy.stackexchange.com/questions/18713/centr
# for calcuation of the center of cluster

cluster_ra = df['ra'].mean()
cluster_dec = df['dec'].mean()
print(f"Cluster Center (mean): RA = {cluster_ra:.5f}, Dec = {cluster_dec:.5f}")

cluster_center = SkyCoord(ra=cluster_ra*u.deg, dec=cluster_dec*u.deg)

# Galaxy positions
galaxies = SkyCoord(ra=df['ra']*u.deg, dec=df['dec']*u.deg)

# calculating angular separations
separations = galaxies.separation(cluster_center)

# Converting to arcminutes
sep_arcmin = separations.arcminute

df['separation_deg'] = separations.deg
df['separation_arcmin'] = sep_arcmin

# Displaying position and separation of each Galaxy
print(df[['ra', 'dec', 'separation_deg', 'separation_arcmin']].head(92))
```

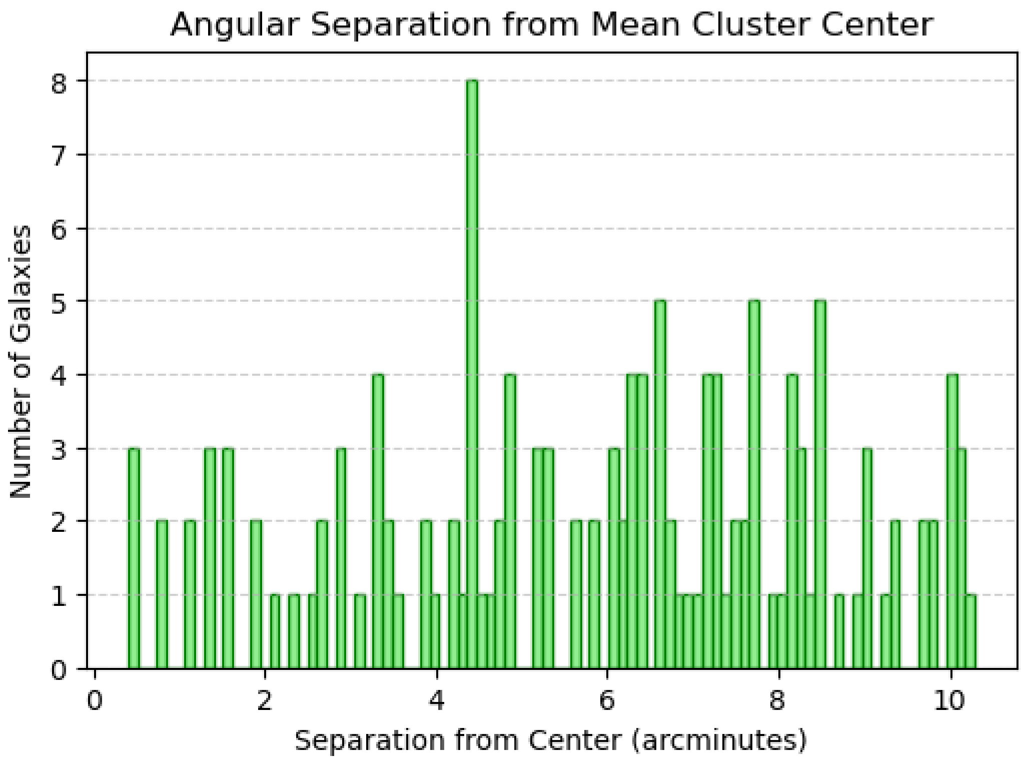
```
Cluster Center (mean): RA = 258.15423, Dec = 64.07246
           ra        dec  separation_deg  separation_arcmin
0   257.82458  64.133257        0.156284           9.377064
1   257.82458  64.133257        0.156284           9.377064
2   257.83332  64.126043        0.150067           9.004039
3   257.85137  64.173247        0.166220           9.973173
4   257.85137  64.173247        0.166220           9.973173
..        ...        ...             ...                ...
87  258.11872  63.963722        0.109847           6.590794
88  258.11872  63.963722        0.109847           6.590794
89  258.13516  64.002787        0.070173           4.210371
90  258.13516  64.002787        0.070173           4.210371
91  258.13222  63.997333        0.075744           4.544638

[92 rows x 4 columns]
```

Step 4: Visualizing Angular Separation of Galaxies

We plot a histogram of the projected (angular) separation of galaxies from the cluster center. This helps us understand the spatial distribution of galaxies within the cluster field.

- The x-axis represents the angular separation (in arcminutes or degrees, depending on units).
- The y-axis shows the number of galaxies at each separation bin.

In [46]:
```
#Plot histogram for proj sep column

plt.figure(figsize=(6, 4))
plt.hist(df['separation_arcmin'], bins=90, color='lightgreen', edgecolor='green')
plt.title("Angular Separation from Mean Cluster Center", fontsize=12)
plt.xlabel("Separation from Center (arcminutes)", fontsize=10)
plt.ylabel("Number of Galaxies", fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.show()
```



Determining size and mass of the cluster:

Step 5: Estimating Physical Diameter of the Cluster

We now estimate the **physical diameter** of the galaxy cluster using cosmological parameters.

- `r` is the **co-moving distance**, approximated using a Taylor expansion for low redshift:
$$r = \frac{cz}{H_0}\left(1 - \frac{z}{2}(1 + q_0)\right)$$
where q_0 is the deceleration parameter
- `ra` is the **angular diameter distance**, given by:
$$D_A = \frac{r}{1+z}$$
- Finally, we convert the observed angular diameter (in arcminutes) into physical size using:
$$\text{diameter (in Mpc)} = D_A \cdot \theta$$
where θ is the angular size in radians, converted from arcminutes.

This gives us a rough estimate of the cluster's size in megaparsecs (Mpc),

In [47]:
```
# Comoving distance
# z_cluster is the mean of cluster red shift
r = (c * z_cluster / H_0) * (1 - (z_cluster / 2) * (1 + q0))  # in Mpc

# Angular diameter distance
ra = r / (1 + z_cluster)

# Angular size / separation / diameter (in radians)
theta_arcmin = df['separation_arcmin'].max()
theta_rad = theta_arcmin * (np.pi / (180 * 60))  # arcmin to radians

# diameter in Mpc
diameter_mpc = ra * theta_rad

# Output
print(f"With the Mean redshift (z): {z_cluster:.5f}, we get the following physica
print('\n'+ f"Angular Diameter: {theta_arcmin:.2f} arcmin")
print(f"Angular Diameter (radians): {theta_rad:.5f} rad")
print(f"Comoving distance (r): {r:.2f}")
print(f"Angular diameter distance (D_A): {ra:.2f}")
print(f"Estimated Physical Diameter: {diameter_mpc:.2f}")
```

```
With the Mean redshift (z): 0.08007, we get the following physical Diameters.

Angular Diameter: 10.29 arcmin
Angular Diameter (radians): 0.00299 rad
Comoving distance (r): 348.15 Mpc
Angular diameter distance (D_A): 322.34 Mpc
Estimated Physical Diameter: 0.97 Mpc
```

Step 6: Calculating the Dynamical Mass of the Cluster

We now estimate the **dynamical mass** of the galaxy cluster using the virial theorem:

$$M_{\text{dyn}} = \frac{3\sigma^2 R}{G}$$

Where:

- σ is the **velocity dispersion** in m/s (`disp * 1000`),
- R is the **cluster radius** in meters (half the physical diameter converted to meters),
- G is the **gravitational constant** in SI units,
- The factor of 3 assumes an isotropic velocity distribution (common in virial estimates).

We convert the final result into **solar masses** by dividing by 2×10^{30} kg.

> This mass estimate assumes the cluster is in dynamical equilibrium and bound by gravity.

In [100]:
```
### Calculating the dynamical mass in solar masses:
M_dyn =(3*((velocity_dispersion*1000)**2)*(diameter_mpc*0.5*(3*10**22)))/(G*(2*1(

print(f"Dynamical Mass of the cluster is {M_dyn:.2e}, solar mass")
```

```
Dynamical Mass of the cluster is 4.83e+14 kg Mpc s2 / m3, solar mass
```

Calculation of Luminousity of the cluster. Using the Pogson's equation, distace modulus etc..

In [92]:
```
from astropy.cosmology import FlatLambdaCDM

M_sun_r = 4.42  # Solar absolute magnitude in SDSS r-band (AB system)
cosmo = FlatLambdaCDM(H0=70, Om0=0.3)

# Luminosity distance (in parsecs)
df['D_L_pc'] = cosmo.luminosity_distance(df['photoz'].values).to(u.pc).value

# Distance modulus
df['mu'] = 5 * np.log10(df['D_L_pc'] / 10)

# Absolute magnitude
df['M_r'] = df['rmag'] - df['mu']

# Luminosity in solar units
df['L_r'] = 10 ** (-0.4 * (df['M_r'] - M_sun_r))

# Total cluster luminosity
L_cluster = df['L_r'].sum()

# --- Output ---
print(f"\nTotal r-band luminosity of the cluster (members only): {L_cluster:.2e}
```

```
Total r-band luminosity of the cluster (members only): 3.26e+12 L_sun
```

In [103]:
```
a = 5
# normalization constant
b = 1.36
#Slope

L_scaled = L_cluster / 10**12  # Scale L as per formula, l-cluster is in tersm oj

M_lum_cluster = 1e14*(L_scaled / a) ** (1 / b)  # Result in solar masses (M_sun)
df['M_lum_gal'] = 1e14 *((df['L_r'] / 1e12) / a) ** (1 / b) # Luminous mass of ec

print(f"L = {L_cluster:.2e} L_sun")
print(f"Estimated luminous mass (M_lum) = {M_lum_cluster:.2e} M_sun")
```

```
L = 3.26e+12 L_sun
Estimated luminous mass (M_lum) = 7.29e+13 M_sun
```

While Calculating the luminous mass, i have use the scaling formula Scaling formula = (L/L_0 = a((M_lum)/(M_0*e15))**b

but was not able to get the values of a and b, hence there is only the estimated value of them, so as to get a optimal result.

Reference: [https://academic.oup.com/view-large/96947779 (https://academic.oup.com/view-large/96947779)](https://academic.oup.com/view-large/96947779) [https://academic.oup.com/view-large/96947779 (https://academic.oup.com/view-large/96947779)](https://academic.oup.com/view-large/96947779)

In [ ]: