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 [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd # type: ignore
from astropy.constants import G, c
from astropy.cosmology import Planck18 as cosmo
import astropy.units as u
```

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 [2]:
```
# Constants:

H_0 = cosmo.H0.to(u.m/u.s/u.pc).value  # Hubble constant in m/s/pc
c = c.value  # Speed of light in m/s
G = G.to(u.m**3/u.kg/u.s**2).value  # Gravitational constant in m^3 kg^-1 s^-2
q0 = -0.534  # Deceleration parameter
```

Read the csv data into the python using the method below

In [5]:
```
df = pd.read_csv('sdss_cluster_data.csv', comment='#')
print(df.columns)
```

```
Index(['objid', 'ra', 'dec', 'photoz', 'photozerr', 'specz', 'speczerr',
       'proj_sep', 'umag', 'umagerr', 'gmag', 'gmagerr', 'rmag', 'rmagerr',
       'obj_type'],
      dtype='object')
```

In [6]:
```python
print (df.head())
```

```
                 objid         ra        dec    photoz  photozerr     specz  \
0  1237671768542478711  257.82458  64.133257  0.079193   0.022867  0.082447
1  1237671768542478711  257.82458  64.133257  0.079193   0.022867  0.082466
2  1237671768542478713  257.83332  64.126043  0.091507   0.014511  0.081218
3  1237671768542544090  257.85137  64.173247  0.081102   0.009898  0.079561
4  1237671768542544090  257.85137  64.173247  0.081102   0.009898  0.079568

   speczerr  proj_sep      umag   umagerr      gmag   gmagerr      rmag  \
0  0.000017  8.347733  18.96488  0.043377  17.49815  0.005672  16.75003
1  0.000014  8.347733  18.96488  0.043377  17.49815  0.005672  16.75003
2  0.000021  8.011259  20.22848  0.072019  18.38334  0.007763  17.46793
3  0.000022  8.739276  19.21829  0.050135  17.18970  0.004936  16.22043
4  0.000019  8.739276  19.21829  0.050135  17.18970  0.004936  16.22043

    rmagerr  obj_type
0  0.004708         3
1  0.004708         3
2  0.005828         3
3  0.003769         3
4  0.003769         3
```

📊 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:

```python
averaged_df = df.groupby('objid').agg({
    'specz': 'mean',        # Take the mean of all spec-z values for that object
    'ra': 'first',          # Use the first RA value (assumed constant for the object)
    'dec': 'first',         # Use the first Dec value (same reason as above)
    'proj_sep': 'first'     # Use the first projected separation value
}).reset_index()
```

In [7]:
```python
# Calculating the average specz for each id:
averaged_df = df.groupby('objid').agg({'specz': 'mean','ra': 'first','dec': 'first'
averaged_df.describe()['specz']
```

Out[7]:
```
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.

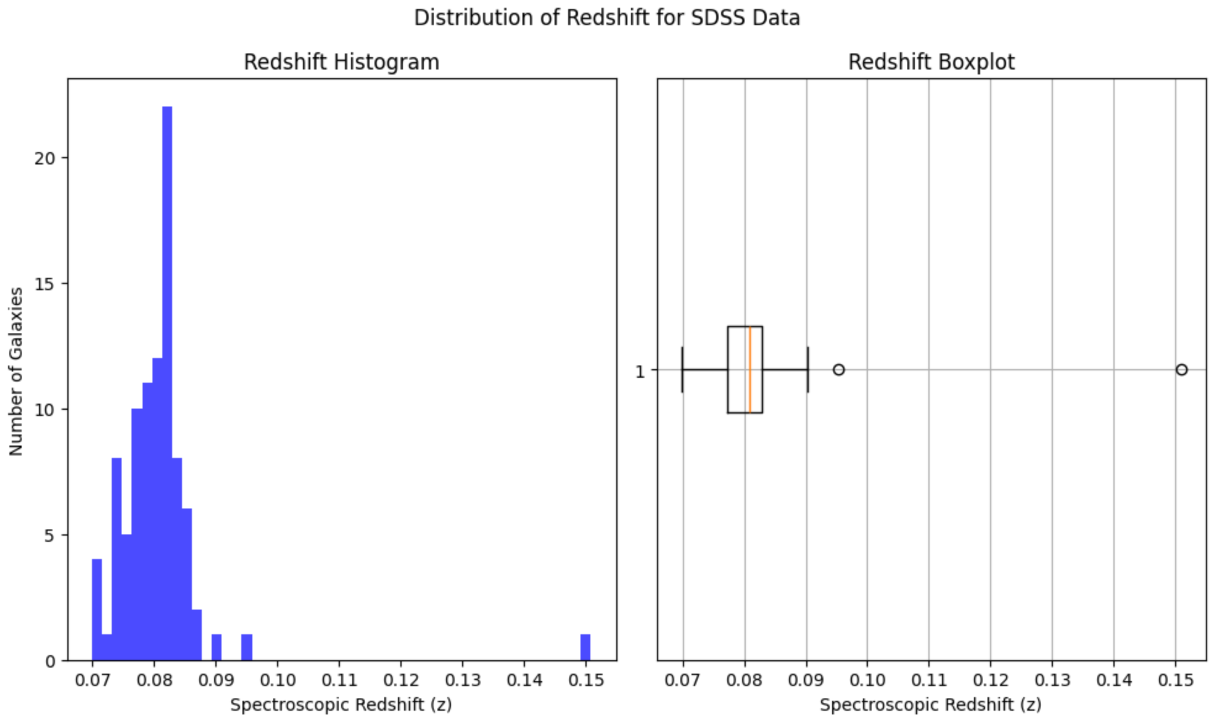
Find the mean, standard deviation and limits of the redshift from the data

In [8]:
```
# Calculate mean and standard deviation of specz
z_mean= averaged_df['specz'].mean()
z_std= averaged_df['specz'].std()
z_lower= z_mean - 3 * z_std
z_upper= z_mean + 3 * z_std

print(f"Mean redshift: {z_mean:.4f}")
print(f"Standard deviation: {z_std:.4f}")
print(f"3-sigma redshift range: {z_lower:.4f},{z_upper:.4f}")
```

```
Mean redshift: 0.0808
Standard deviation: 0.0086
3-sigma redshift range: 0.0551,0.1066
```

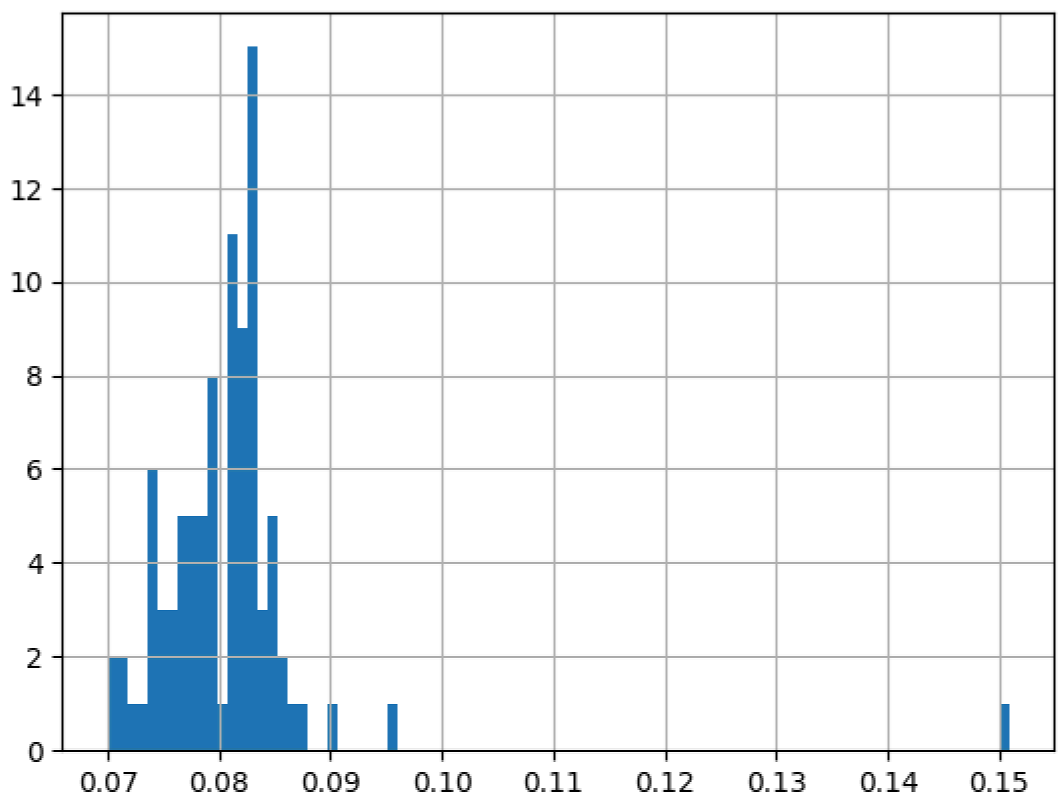
You can also use boxplot to visualize the overall values of redshift

In [9]:
```
# Plot the dsitribution of redshift as histogram and a boxplot
plt.figure(figsize=(10,6))
plt.subplot(1,2,1)
plt.hist(averaged_df['specz'],bins=50,color='blue',alpha=0.7)
plt.xlabel('Spectroscopic Redshift (z)')
plt.ylabel('Number of Galaxies')
plt.title('Redshift Histogram')

plt.subplot(1,2,2)
plt.boxplot(averaged_df['specz'],vert=False)
plt.xlabel('Spectroscopic Redshift (z)')
plt.title('Redshift Boxplot')
plt.grid(True)

plt.suptitle('Distribution of Redshift for SDSS Data')
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 [10]:
```
plt.hist(averaged_df['specz'],bins=90)
plt.grid()
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

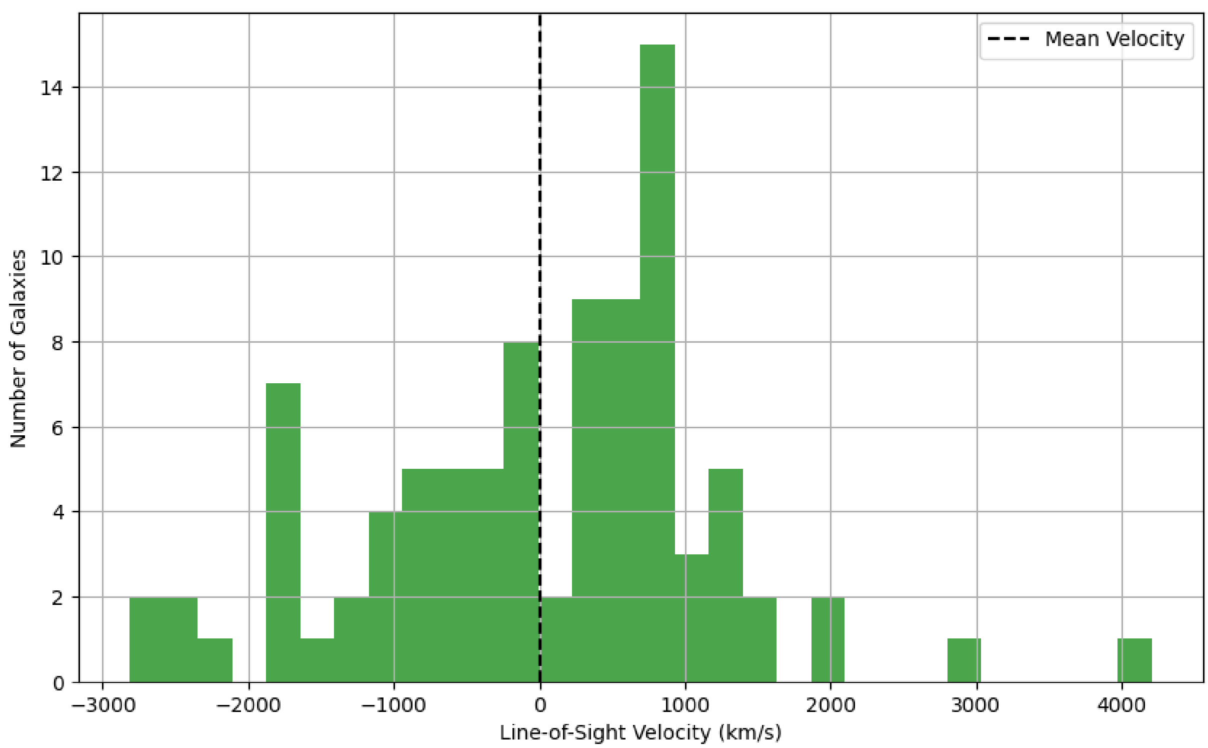
In [11]:
```
# Filtering the data based on specz values, used 3 sigma deviation from mean as upp
filtered_df= averaged_df[(averaged_df['specz'] >= z_lower) & (averaged_df['specz']
print(f"Number of cluster members: {len(filtered_df)}")
```

```
Number of cluster members: 91
```

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 [12]:
```
cluster_redshift = filtered_df['specz'].mean()
filtered_df = filtered_df.copy()  # Avoid SettingWithCopyWarning
filtered_df.loc[:, 'velocity'] = c * ((1 + filtered_df['specz'])**2 - (1 + cluster_
filtered_df.loc[:, 'velocity'] /= 1000  # Convert m/s to km/s
```

In [13]:
```
#plot the velocity column created as hist
plt.figure(figsize=(10,6))
plt.hist(filtered_df['velocity'], bins=30,color='green',alpha=0.7)
plt.axvline(0,color='black',linestyle='--',label='Mean Velocity')
plt.xlabel('Line-of-Sight Velocity (km/s)')
plt.ylabel('Number of Galaxies')
plt.legend()
plt.grid(True)
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

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 [14]:
```
# Calculate velocity dispersion
disp= filtered_df['velocity'].std(ddof=1)
print(f"Velocity dispersion: {disp:.2f} km/s")

#Quick stats using describe
print(f"\nVelocity ststistics:")
print(filtered_df['velocity'].describe())
```

```
Velocity dispersion: 1218.49 km/s

Velocity ststistics:
count      91.000000
mean       -2.449331
std      1218.492945
min     -2814.230840
25%      -806.606785
50%       237.179091
75%       754.977576
max      4206.136789
Name: velocity, dtype: float64
```

In [15]:
```
print(f"The value of the cluster redshift = {cluster_redshift:.4f}")
print(f"The characteristic value of velocity dispersion of the cluster along the li
```

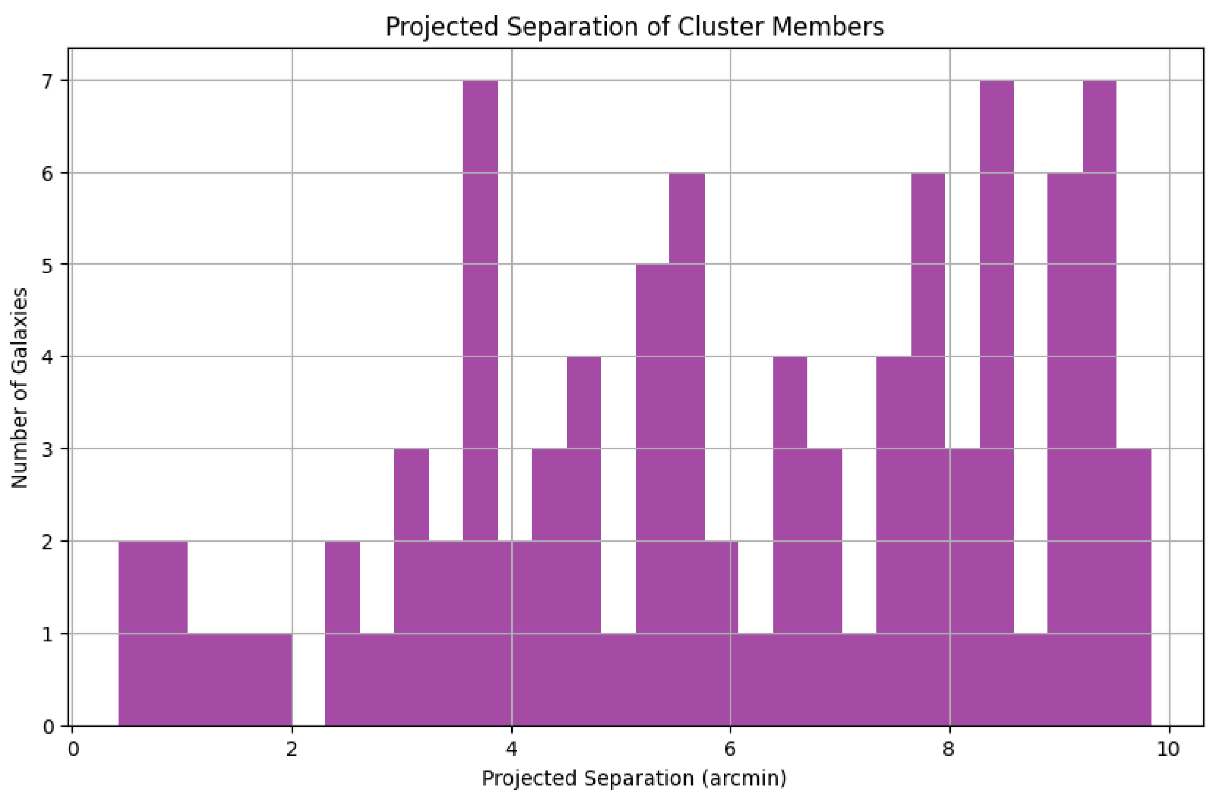
```
The value of the cluster redshift = 0.0801
The characteristic value of velocity dispersion of the cluster along the line of sig
ht = 1218.49 km/s.
```

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 [16]:
```
#Plot histogram for proj sep column
plt.figure(figsize=(10,6))
plt.hist(filtered_df['proj_sep'],bins=30,color='purple',alpha=0.7)
plt.xlabel('Projected Separation (arcmin)')
plt.ylabel('Number of Galaxies')
plt.title('Projected Separation of Cluster Members')
plt.grid(True)
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), assuming a flat ΛCDM cosmology.

In [17]:
```
# Co-moving distance
r= (c / H_0) * cluster_redshift * (1-(cluster_redshift / 2) * (1 + q0))

# Angular diameter distance
D_A= r/ (1 + cluster_redshift)

#Physical diameter(use median proj_sep as angular size)
theta = np.median(filtered_df['proj_sep']) * (np.pi / (180 * 60))        # Convert
diameter= D_A * theta / 1e6          # Convert to Mpc (since D_A is in pc)

print(f"Cluster diameter: {diameter:.2f} Mpc")
```

```
Cluster diameter: 0.59 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 $2 \times 10^{30}\,\text{kg}$.

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

In [18]:
```
### Calculating the dynamical mass in solar masses:
R = (diameter * 0.5) * 3.0857e22  # Radius in meters
sigma = disp * 1000  # m/s
M_sun = 1.989e30  # kg
M_dyn = 3 * (sigma**2) * R / G / M_sun

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

```
Dynamical Mass of the cluster is 3.07e+14 solar mass
```

In [19]:
```
print(filtered_df.columns)
filtered_df['gmag'] = df.loc[filtered_df.index, 'gmag']
```

```
Index(['objid', 'specz', 'ra', 'dec', 'proj_sep', 'velocity'], dtype='object')
```

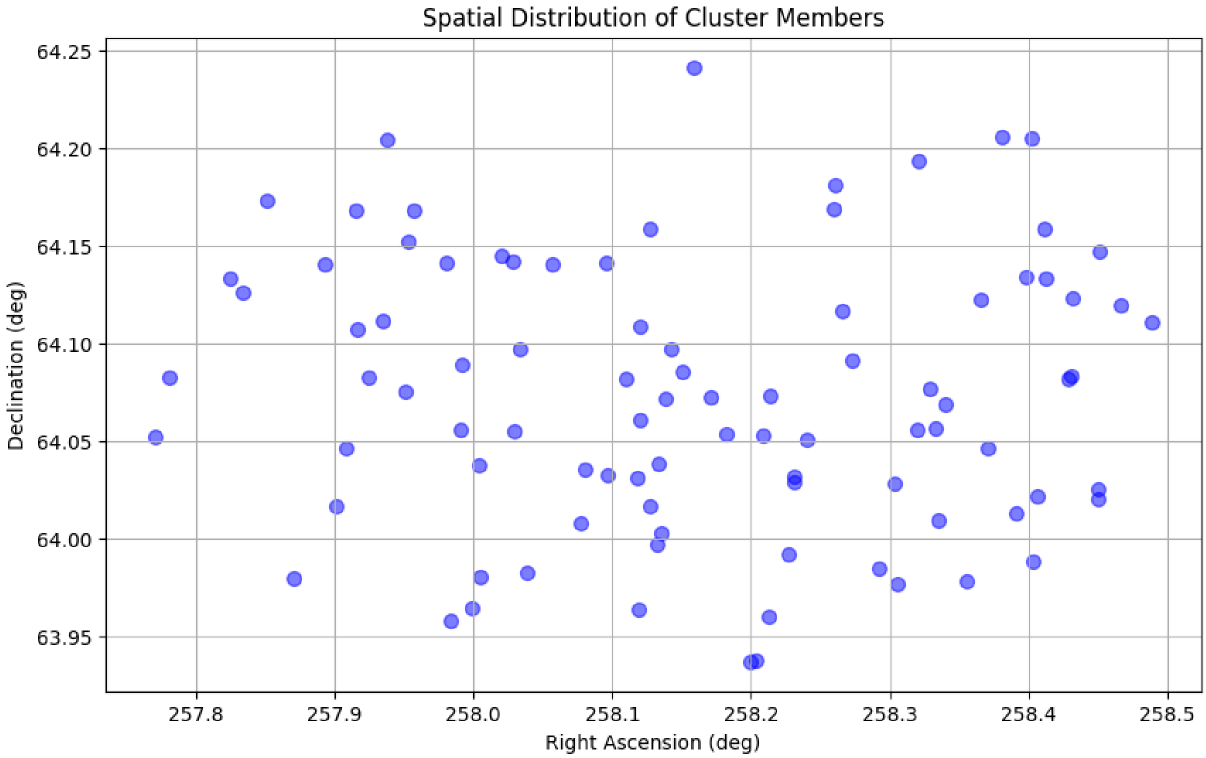
In [20]:
```
D_L = cosmo.luminosity_distance(cluster_redshift).to(u.pc).value
M_g = filtered_df['gmag'] - 5 * np.log10(D_L / 10)
M_g_sun = 5.1
L = 10**(-0.4 * (M_g - M_g_sun))
upsilon = 5
M_lum = upsilon * L
M_lum_total = np.sum(M_lum)
print(f"Luminous mass: {M_lum_total:.2e} M_sun")
print(f"Mass-to-light ratio: {M_dyn / np.sum(L):.2f} M_sun / L_sun")
```

```
Luminous mass: 8.54e+12 M_sun
Mass-to-light ratio: 179.82 M_sun / L_sun
```

In [21]:
```
#Print all key results together for verification
print(f"Cluster redshift: {cluster_redshift:.4f}")
print(f"Velocity dispersion: {disp:.2f} km/s")
print(f"Cluster diameter: {diameter:.2f} Mpc")
print(f"Dynamical mass: {M_dyn:.2e} M_sun")
print(f"Luminous mass: {M_lum_total:.2e} M_sun")
print(f"Mass-to-light ratio: {M_dyn / np.sum(L):.2f} M_sun / L_sun")
```

```
Cluster redshift: 0.0801
Velocity dispersion: 1218.49 km/s
Cluster diameter: 0.59 Mpc
Dynamical mass: 3.07e+14 M_sun
Luminous mass: 8.54e+12 M_sun
Mass-to-light ratio: 179.82 M_sun / L_sun
```

In [22]:
```
### RA vs. Dec Scatter Plot
plt.figure(figsize=(10,6))
plt.scatter(filtered_df['ra'],filtered_df['dec'], s=50 , c='blue' , alpha=0.5)
plt.xlabel('Right Ascension (deg)')
plt.ylabel('Declination (deg)')
plt.title('Spatial Distribution of Cluster Members')
plt.grid(True)
plt.show()
```


Spatial Distribution of Cluster Members
64.25
64.20
64.15
64.10
64.05
64.00
63.95
Declination (deg)
257.8
257.9
258.0
258.1
258.2
258.3
258.4
258.5
Right Ascension (deg)

```
In [ ]:
```