Estimating the Dynamical Mass of a Galaxy Cluster

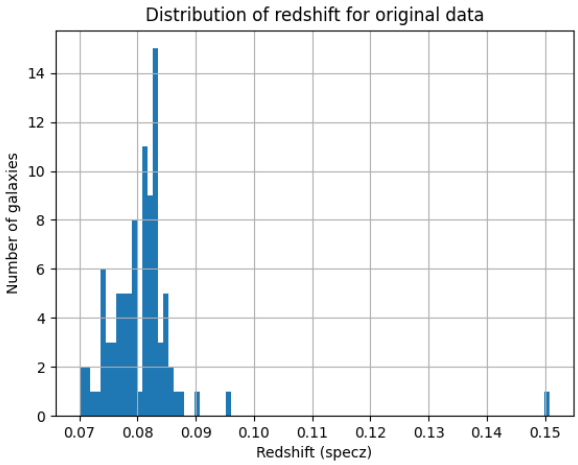
Aim: To estimate the dynamical mass of a galaxy cluster, which is the total mass of the cluster independent of light

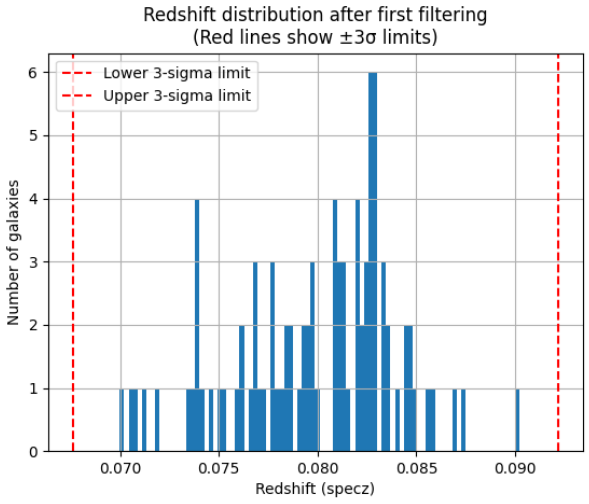
Data extraction: Downloaded galaxy data (RA, Dec, redshift, magnitudes, etc.) from SDSS in FITS format using the SQL Query provided in the project handout.

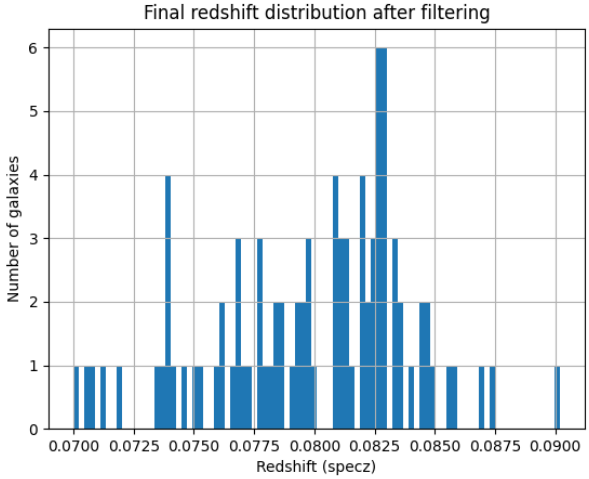
Project Details: Since the handout was divided into five subparts, I will explain my project following the same structure.

1. <u>Identifying galaxies that may be members of a cluster</u>

   The histogram of the averaged redshifts is plotted. It is observed that most of the data lies between values 0.07 and 0.09, with the mode being around 8.30. Outliers such as the values 0.15 and 0.095 are filtered out of the data to prevent skewing of mean and standard deviation. As the D'Agostino test p-value for this data is 0.2597411373218642 (greater than 0.05), the data can be considered to roughly have a normal distribution. Hence, a 3σ filter can be employed to further filter the data. However, this second filtration is not of much use, as all the values of the filtered data, lie within the lower and upper limits (0.06758568 and 0.092210904 respectively).







From the Relativistic Doppler Shift equation, we get:

$$v = c\,\frac{((1+z)^2-1)}{((1+z)^2+1)}$$

where:

v is the expansion velocity,

c is the speed of light and

z is the redshift of the galaxy.

From this point onward, the symbol c will refer to the speed of light in all subsequent formulas.

From this equation the histogram of the expansion velocities of the individual galaxies is plotted.

The expansion velocities range from 20,000 to 26,000 km/sec, with average expansion velocity being 22996.37km/s.

2. Determining the cluster redshift and estimating the characteristic velocity dispersion of galaxies

The cluster redshift is estimated by taking the mean of the redshifts of all the galaxies in the cluster. It was found to have a value of 0.0799.

The dispersion velocity of each galaxy is calculated using the formula provided in the handout, which is also derived using the Relativistic Doppler formula. The formula is:

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

where,

v is the dispersion velocity,

z is the redshift of the galaxy and

$\mathrm{z_{cluster}}$ is the redshift of the cluster

The characteristic velocity dispersion, or the standard deviation of the dispersion velocities, is found to have a value of 1140. 26 km/s.

3. Estimating the characteristic size of the cluster in Mpc

The physical diameter of the galaxy cluster is estimated using cosmological parameters.

The comoving distance is found using this formula provided in the handout:

$$r = c\frac{z\_cluster}{H_0}\,(1 - z_{cluster}\frac{(1+q_0)}{2})$$

where,

r is the comoving distance,

H_0 is the Hubble constant,

q_0 is a deceleration constant and

z_cluster is the redshift of the cluster

Using the comoving distance, the angular diameter distance, D_A, given by

$$D_A = \frac{r}{(1+z_{cluster})}$$

is calculated. Finally, we convert the observed angular diameter into physical size using

$$diameter\ (in\ Mpc) = D_A.\theta$$

where θ is the maximum projected separation in radians. The diameter was found to be 0.921 Mpc.

4. Estimating the dynamical mass of the cluster and quoting the value in units of solar mass

The dynamical mass of the galaxy cluster is estimated using the virial theorem:

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

where:

σ is the velocity dispersion in m/s,

R is the cluster radius in meters (half the physical diameter converted to meters) and

G is the gravitational constant in SI units.

The factor of 3 assumes an isotropic velocity distribution (common in virial estimates).

The final result is converted into solar masses by dividing by 2×10^{30}kg.
The dynamical mass of the galaxy cluster was found to be 1.35e+14 solar masses.

5. Estimating the luminous mass and explaining any inconsistencies with the dynamical mass.

First, the luminosity distance is calculated in parsecs using the luminosity distance() function from the Astropy library in Python.
Using the luminosity distance, the absolute magnitude for the r-band, M_r, is computed using the formula

$$M_r = m_r - 5\log_{10}\left(\frac{luminosity\ distance}{10}\right) - 1$$

where,
M_r is the absolute magnitude and
m_r is the observed magnitude.
The r-band luminosity of a galaxy, L_r , in terms of solar luminosity can be calculated using

$$L_r = 10^{0.4(M_r sun - M_r)}$$

Here $M_r sun$ is the absolute magnitude of the Sun in the r-band and its value is known to be around 4.67.

Next, the luminous-mass (≈ stellar mass) to light ratio, $\frac{M}{L_r}$, is calculated using

$$\log_{10}\left(\frac{M}{L_r}\right) = -0.306 + 1.097(g - r)$$

where,

g is the observed g-band magnitude and

r is the observed r-band magnitude.

Using this ratio the luminous mass is finally calculated using

$$M_{stellar} \approx M_{luminous} = Lr.(\frac{M}{L_r})$$

The luminous mass is found to have value 1.12e+13 solar masses.

So the luminous mass is only 8.29% of the dynamical mass. At first glance, it may seem that the mass unaccounted for in the luminous mass could simply be made up of non-luminous objects like rocks, gas, or planets. However, observations show that even after accounting for all visible and invisible baryonic matter (stars, gas, dust), there's still not enough mass to explain the gravitational effects. This strongly suggests the presence of non-baryonic dark matter — a different form of matter that doesn't emit or interact with light, but exerts gravitational influence.

References:

1. Section 3.2 of 'The Optical and Near-Infrared Properties of Galaxies. I. Luminosity and Stellar Mass Functions' by Eric F. Bell

2.https://phys.libretexts.org/Bookshelves/University_Physics/University_Physics_(OpenStax)/University_Physics_III_-_Optics_and_Modern_Physics_(OpenStax)/05%3A__Relativity/5.08%3A_Doppler_Effect_for_Light

3. https://en.wikipedia.org/wiki/Absolute_magnitude

4. https://github.com/supremeKAI40/ISA-Summer_School_2025/blob/main/projects/dynamical_mass/1_dynamical_mass.ipynb

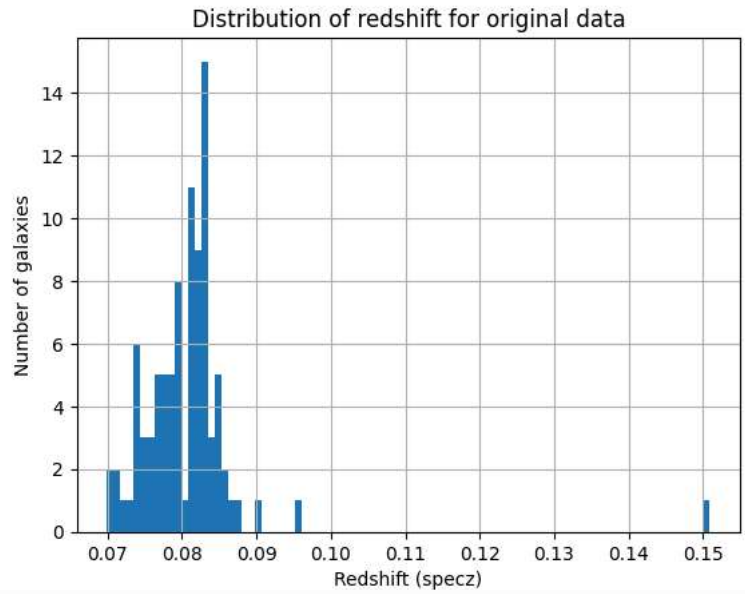
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from astropy.io import fits
from astropy.constants import G, c
from astropy.cosmology import Planck18 as cosmo
import astropy.units as u

H_0 = cosmo.H0  # Hubble constant
c_kms = c.to('km/s').value  # speed of light in km/s
q0 = -0.534  # deceleration parameter (assumed)



from google.colab import files

uploaded = files.upload('C:/Users/Welcome-Pc/OneDrive/Desktop/python/cluster.fits')  # Opens a file chooser to upload files
```

Choose Files cluster.fits

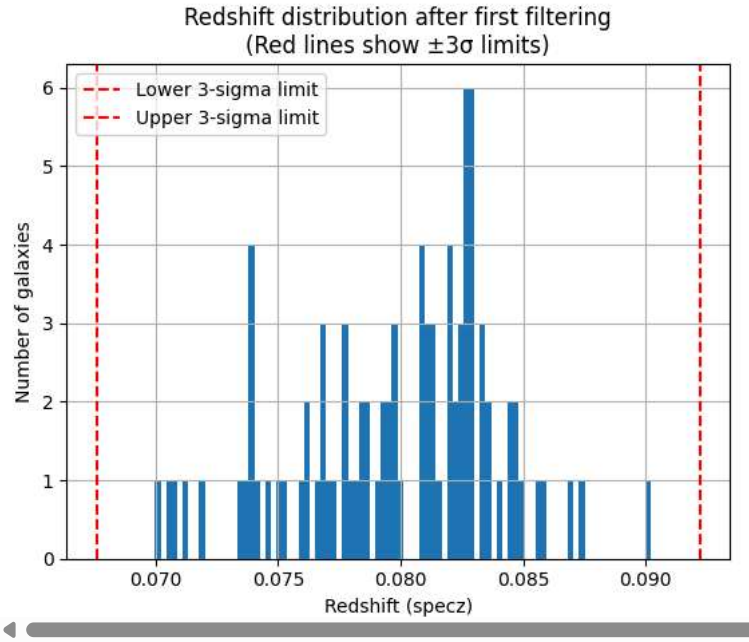
- **cluster.fits**(n/a) - 20160 bytes, last modified: 6/30/2025 - 100% done

Saving cluster.fits to C:/Users/Welcome-Pc/OneDrive/Desktop/python/cluster.fits/cluster.fits

```
# Opening FITS file and extracting data
file = fits.open('C:/Users/Welcome-Pc/OneDrive/Desktop/python/cluster.fits/cluster.fits')
data = file[1].data

def to_native(arr):#function to check whether byteorder is correct
    if arr.dtype.byteorder not in ('=', '|'):
        return arr.byteswap().view(arr.dtype.newbyteorder('='))
    return arr

# Extracting columns with correct byte order
specz = to_native(data['specz'])
objid = to_native(data['objid'])
ra = to_native(data['ra'])
dec = to_native(data['dec'])
proj_sep = to_native(data['proj_sep'])
rmag = to_native(data['rmag'])
gmag = to_native(data['gmag'])

# Creating DataFrame
df = pd.DataFrame({
    'specz': specz,
    'objid': objid,
    'ra': ra,
    'dec': dec,
    'proj_sep': proj_sep,
    'gmag': gmag,
    'rmag': rmag
})

# Grouping and averaging values by objid
averaged_df = df.groupby('objid').agg({
    'specz': 'mean',
    'rmag': 'mean',
    'gmag': 'mean',
    'ra': 'first',
    'dec': 'first',
    'proj_sep': 'first'
}).reset_index()


# Histogram of original redshift distribution
plt.hist(averaged_df['specz'], bins=90)
plt.grid()
plt.title("Distribution of redshift for original data")
plt.xlabel("Redshift (specz)")
plt.ylabel("Number of galaxies")
plt.show()
```

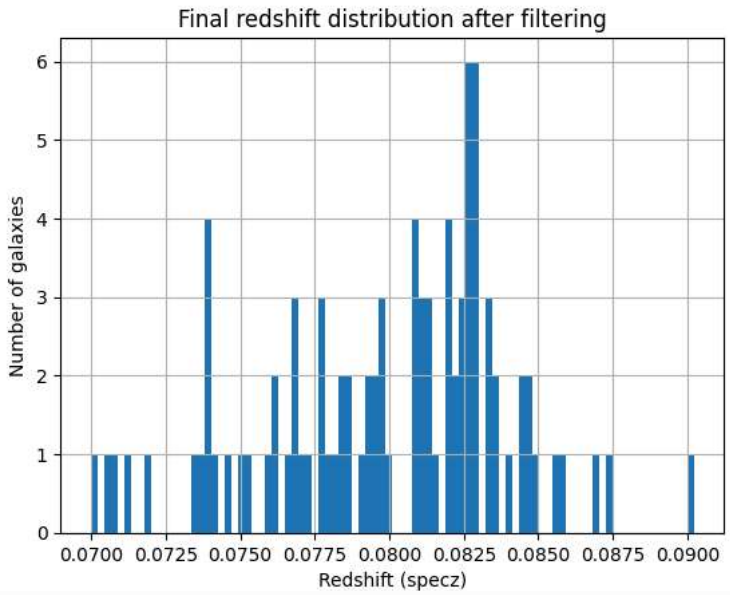



```
# Initial filtering by redshift outliers like 0.15 so that standard deviation does not get affected
filtered_df = averaged_df[(averaged_df['specz'] <= 0.095)].copy()
mean_z = filtered_df['specz'].mean()
std_z = filtered_df['specz'].std()
lower_limit = mean_z - 3 * std_z
upper_limit = mean_z + 3 * std_z

# Histogram after first filtering with 3-sigma lines
plt.hist(filtered_df['specz'], bins=90)
plt.grid()
plt.axvline(lower_limit, color='r', linestyle='--', label='Lower 3-sigma limit')
plt.axvline(upper_limit, color='r', linestyle='--', label='Upper 3-sigma limit')
plt.title("Redshift distribution after first filtering\n(Red lines show ±3σ limits)")
plt.xlabel("Redshift (specz)")
plt.ylabel("Number of galaxies")
plt.legend()
plt.show()
```

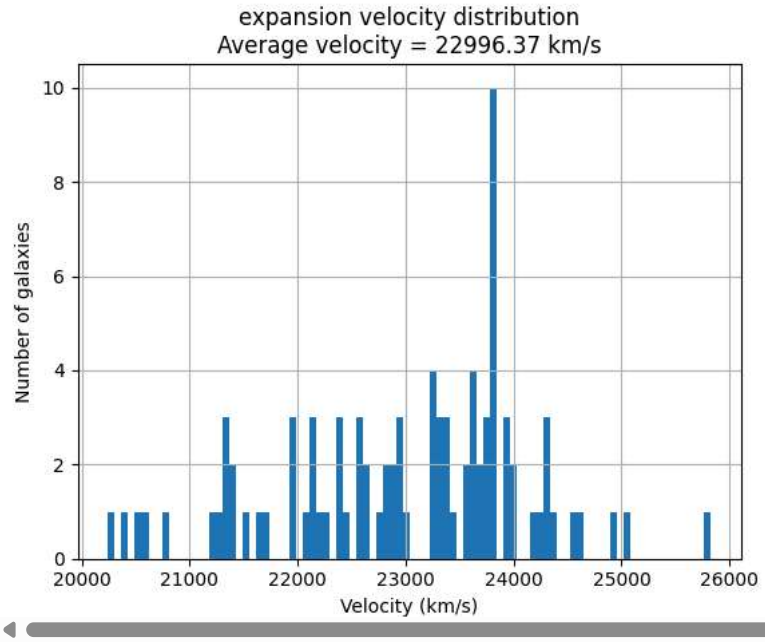



```
# Final filtering with 3-sigma limits(not really needed because as seen in histogram, all values lie within the limits)
filtered_df = averaged_df[(averaged_df['specz'] >= lower_limit) & (averaged_df['specz'] <= upper_limit)].copy()

# Histogram after final filtering
plt.hist(filtered_df['specz'], bins=90)
plt.grid()
plt.title("Final redshift distribution after filtering")
```

```
plt.xlabel("Redshift (specz)")
plt.ylabel("Number of galaxies")
plt.show()
```



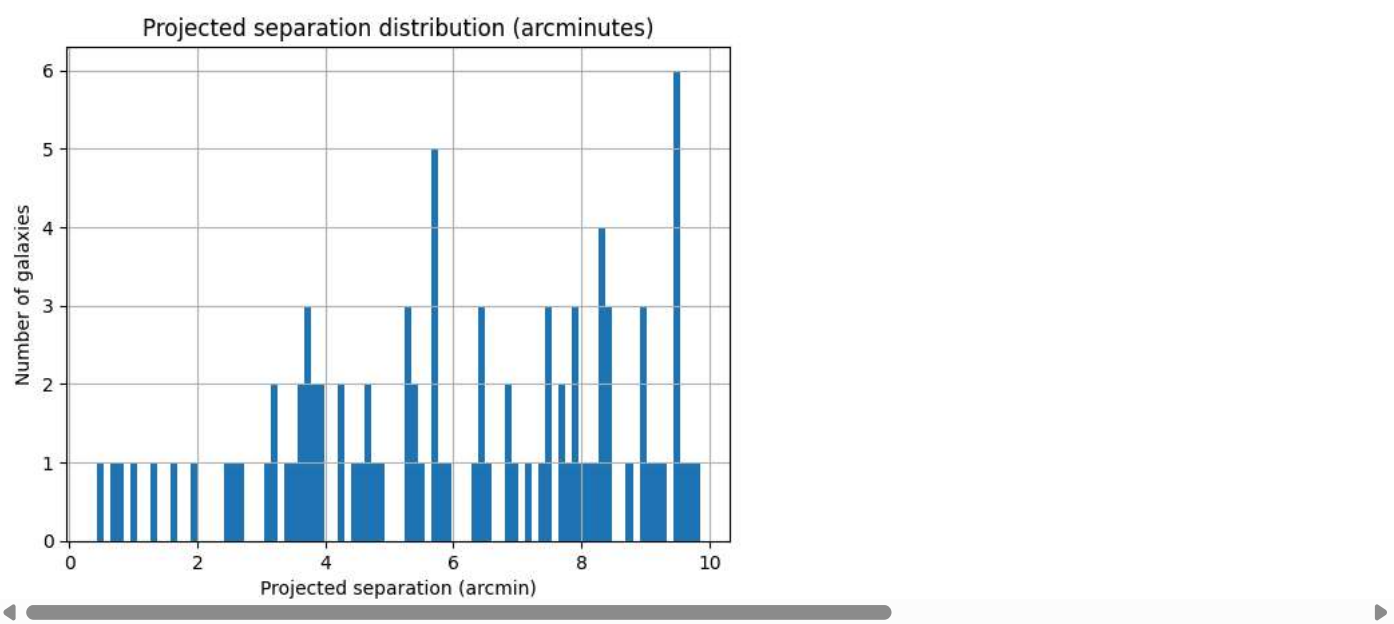

```
# Calculating expansion velocity and add to dataframe using .loc
filtered_df.loc[:, 'velocity'] = c_kms * ((1 + filtered_df['specz'])**2 - 1) / ((1 + filtered_df['specz'])**2 + 1)

# Histogram of expansion velocities
plt.hist(filtered_df['velocity'], bins=90)
plt.grid()
plt.title(f"expansion velocity distribution\nAverage velocity = {filtered_df['velocity'].mean():.2f} km/s")
plt.xlabel("Velocity (km/s)")
plt.ylabel("Number of galaxies")
plt.show()
```




```
# Calculating cluster redshift mean and velocity dispersion
cluster_redshift = filtered_df['specz'].mean()
filtered_df.loc[:, 'disp'] = c_kms * ((1 + filtered_df['specz'])**2 - (1 + cluster_redshift)**2) / ((1 + filtered_df['specz'])**2 + (1 + clu
velocity_disp = filtered_df['disp'].std()

print(f"Cluster redshift = {cluster_redshift:.4f}")
print(f"Velocity dispersion (line-of-sight) = {velocity_disp:.2f} km/s")
```

```
Cluster redshift = 0.0799
Velocity dispersion (line-of-sight) = 1140.26 km/s
```

```
# Histogram of projected separation
plt.hist(filtered_df['proj_sep'], bins=90)
plt.grid()
plt.title("Projected separation distribution (arcminutes)")
plt.xlabel("Projected separation (arcmin)")
plt.ylabel("Number of galaxies")
plt.show()
```




```
# Convert projected separation max to radians
theta_rad = (filtered_df['proj_sep'].max()) * (np.pi / 180) / 60  # arcmin to radians

# Comoving distance and angular diameter distance in Mpc
r = (c_kms * cluster_redshift / H_0.value) * (1 - cluster_redshift * (1 + q0) / 2)
DA = r / (1 + cluster_redshift)
diameter = DA * theta_rad

# Convert velocity dispersion to m/s and diameter to meters
velocity_disp_ms = (velocity_disp * u.km / u.s).to(u.m / u.s).value
diameter_m = diameter * 3.0e22  # 1 Mpc in meters
print(f'Diameter of the cluster is {diameter} Mpc')
# Calculate dynamical mass in kg, then solar masses
M_dyn_kg = (velocity_disp_ms**2 * diameter_m) / (2 * G.value)#dividing by 2 because we want radius not diameter
M_dyn_solar = M_dyn_kg / 2.0e30
print(f"Dynamical mass of the cluster = {M_dyn_solar:.2e} solar masses")
```

```
Diameter of the cluster is 0.9213036653881527 Mpc
Dynamical mass of the cluster = 1.35e+14 solar masses
```

```
# Calculating luminosity distance in parsecs (float)
filtered_df.loc[:, 'ld'] = cosmo.luminosity_distance(filtered_df['specz']).to('pc').value

# Calculating absolute magnitude in r-band
filtered_df.loc[:, 'abs_rmag'] = filtered_df['rmag'] - 5 * np.log10(filtered_df['ld'] / 10)
M_r_sun = 4.67  # Sun's absolute magnitude in r-band

# Calculating luminosity in solar units
filtered_df.loc[:, 'L_Lsun'] = 10 ** (0.4 * (M_r_sun - filtered_df['abs_rmag']))

# Calculating stellar mass-to-light ratio (log scale)
filtered_df.loc[:, 'logratio'] = -0.306 + 1.097 * (filtered_df['gmag'] - filtered_df['rmag'])

# Converting log ratio to ratio
filtered_df.loc[:, 'ratio'] = 10 ** filtered_df['logratio']

# Calculating stellar mass per galaxy
filtered_df.loc[:, 'stell_mass'] = filtered_df['ratio'] * filtered_df['L_Lsun']

# Total stellar mass of cluster
total_stellar_mass = filtered_df['stell_mass'].sum()
print(f"Total stellar/luminous mass of the cluster = {total_stellar_mass:.2e} solar masses")
```

```
print("Total stellar/luminous mass of the cluster = {total_stellar_mass:.2e} solar masses")
```

```
Total stellar/luminous mass of the cluster = 1.12e+13 solar masses
```

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from astropy.io import fits
from astropy.constants import G, c
from astropy.cosmology import Planck18 as cosmo
import astropy.units as u

H_0 = cosmo.H0  # Hubble constant
c_kms = c.to('km/s').value  # speed of light in km/s
q0 = -0.534  # deceleration parameter (assumed)

# Opening FITS file and extracting data
file = fits.open('C:/Users/Welcome-
Pc/OneDrive/Desktop/python/cluster.fits')
data = file[1].data

def to_native(arr):#function to check whether byteorder is correct
    if arr.dtype.byteorder not in ('=', '|'):
        return arr.byteswap().view(arr.dtype.newbyteorder('='))
    return arr

# Extracting columns with correct byte order
specz = to_native(data['specz'])
objid = to_native(data['objid'])
ra = to_native(data['ra'])
dec = to_native(data['dec'])
proj_sep = to_native(data['proj_sep'])
rmag = to_native(data['rmag'])
gmag = to_native(data['gmag'])

# Creating DataFrame
df = pd.DataFrame({
    'specz': specz,
    'objid': objid,
    'ra': ra,
    'dec': dec,
    'proj_sep': proj_sep,
    'gmag': gmag,
    'rmag': rmag
})

# Grouping and averaging values by objid
averaged_df = df.groupby('objid').agg({
    'specz': 'mean',
    'rmag': 'mean',
    'gmag': 'mean',
    'ra': 'first',
    'dec': 'first',
    'proj_sep': 'first'
}).reset_index()

# Histogram of original redshift distribution
plt.hist(averaged_df['specz'], bins=90)
plt.grid()
plt.title("Distribution of redshift for original data")
plt.xlabel("Redshift (specz)")
plt.ylabel("Number of galaxies")
plt.show()
```

```
# Initial filtering by redshift outliers like 0.15 so that standard
deviation does not get affected
filtered_df = averaged_df[(averaged_df['specz'] <= 0.095)].copy()
mean_z = filtered_df['specz'].mean()
std_z = filtered_df['specz'].std()
lower_limit = mean_z - 3 * std_z
upper_limit = mean_z + 3 * std_z

# Histogram after first filtering with 3-sigma lines
plt.hist(filtered_df['specz'], bins=90)
plt.grid()
plt.axvline(lower_limit, color='r', linestyle='--', label='Lower 3-sigma
limit')
plt.axvline(upper_limit, color='r', linestyle='--', label='Upper 3-sigma
limit')
plt.title("Redshift distribution after first filtering\n(Red lines show ±3σ
limits)")
plt.xlabel("Redshift (specz)")
plt.ylabel("Number of galaxies")
plt.legend()
plt.show()

# Final filtering with 3-sigma limits(not really needed because as seen in
histogram, all values lie within the limits)
filtered_df = averaged_df[(averaged_df['specz'] >= lower_limit) &
(averaged_df['specz'] <= upper_limit)].copy()

# Histogram after final filtering
plt.hist(filtered_df['specz'], bins=90)
plt.grid()
plt.title("Final redshift distribution after filtering")
plt.xlabel("Redshift (specz)")
plt.ylabel("Number of galaxies")
plt.show()
# Calculating expansion velocity and add to dataframe using .loc
filtered_df.loc[:, 'velocity'] = c_kms * ((1 + filtered_df['specz'])**2 -
1) / ((1 + filtered_df['specz'])**2 + 1)

# Histogram of expansion velocities
plt.hist(filtered_df['velocity'], bins=90)
plt.grid()
plt.title(f"expansion velocity distribution\nAverage velocity =
{filtered_df['velocity'].mean():.2f} km/s")
plt.xlabel("Velocity (km/s)")
plt.ylabel("Number of galaxies")
plt.show()

# Calculating cluster redshift mean and velocity dispersion
cluster_redshift = filtered_df['specz'].mean()
filtered_df.loc[:, 'disp'] = c_kms * ((1 + filtered_df['specz'])**2 - (1 +
cluster_redshift)**2) / ((1 + filtered_df['specz'])**2 + (1 +
cluster_redshift)**2)
velocity_disp = filtered_df['disp'].std()

print(f"Cluster redshift = {cluster_redshift:.4f}")
print(f"Velocity dispersion (line-of-sight) = {velocity_disp:.2f} km/s")

# Histogram of projected separation
plt.hist(filtered_df['proj_sep'], bins=90)
plt.grid()
plt.title("Projected separation distribution (arcminutes)")
```

```
plt.xlabel("Projected separation (arcmin)")
plt.ylabel("Number of galaxies")
plt.show()

# Convert projected separation max to radians
theta_rad = (filtered_df['proj_sep'].max()) * (np.pi / 180) / 60  # arcmin
to radians

# Comoving distance and angular diameter distance in Mpc
r = (c_kms * cluster_redshift / H_0.value) * (1 - cluster_redshift * (1 +
q0) / 2)
DA = r / (1 + cluster_redshift)
diameter = DA * theta_rad

# Convert velocity dispersion to m/s and diameter to meters
velocity_disp_ms = (velocity_disp * u.km / u.s).to(u.m / u.s).value
diameter_m = diameter * 3.0e22  # 1 Mpc in meters

# Calculate dynamical mass in kg, then solar masses
M_dyn_kg = (velocity_disp_ms**2 * diameter_m) / (2 * G.value)#dividing by 2
because we want radius not diameter
M_dyn_solar = M_dyn_kg / 2.0e30
    print(f"Dynamical mass of the cluster = {M_dyn_solar:.2e} solar
masses")

# Calculating luminosity distance in parsecs (float)
filtered_df.loc[:, 'ld'] =
cosmo.luminosity_distance(filtered_df['specz']).to('pc').value

# Calculating absolute magnitude in r-band
filtered_df.loc[:, 'abs_rmag'] = filtered_df['rmag'] - 5 *
np.log10(filtered_df['ld'] / 10)
M_r_sun = 4.67  # Sun's absolute magnitude in r-band

# Calculating luminosity in solar units
filtered_df.loc[:, 'L_Lsun'] = 10 ** (0.4 * (M_r_sun -
filtered_df['abs_rmag']))

# Calculating stellar mass-to-light ratio (log scale)
filtered_df.loc[:, 'logratio'] = -0.306 + 1.097 * (filtered_df['gmag'] -
filtered_df['rmag'])

# Converting log ratio to ratio
filtered_df.loc[:, 'ratio'] = 10 ** filtered_df['logratio']

# Calculating stellar mass per galaxy
filtered_df.loc[:, 'stell_mass'] = filtered_df['ratio'] *
filtered_df['L_Lsun']

# Total stellar mass of cluster
total_stellar_mass = filtered_df['stell_mass'].sum()
print(f"Total stellar/luminous mass of the cluster =
{total_stellar_mass:.2e} solar masses")
```