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
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:
G=4.302e-9 # in Mpc/M⊙.(km/s)^2
c=float(c.to('km/s').value)
H_0=70
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 [3]:
```
df=pd.read_csv('Skyserver_SQL6_15_2025 8_20_03 AM.csv')
df
```

Out[3]:

objid	ra	dec	photoz	photozerr	specz	specze
1237671768542478711	257.82458	64.133257	0.079193	0.022867	0.08244731	1.657283
					0.08246633	1.43452
1237671768542478713	257.83332	64.126043	0.091507	0.014511	0.08121841	2.131049
1237671768542544090	257.85137	64.173247	0.081102	0.009898	0.07956107	2.217769
...	...	...	...	...	...	
1237671939804627464	258.44994	64.025909	0.081894	0.013624	0.0718016	2.505877
1237671939804627483	258.43205	64.123685	0.077184	0.010523	0.08079002	2.528532
1237671939804627518	258.46676	64.119499	0.088762	0.016934	0.08287635	5.564596
1237671939804627535	258.48871	64.111343	0.079721	0.009565	0.08089872	2.86031
					0.08089609	2.330272

140 rows × 1 columns

📊 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:

```
df_grouped = df.groupby("objid").agg({
    "specz": "mean",    # Average redshift for each galaxy
    "ra": "first",      # Keep the first instance of Right Ascension (RA)
    "dec": "first",     # Keep the first instance of Declination (DEC)
    "proj_sep": "first" # Keep the first projected separation value
}).reset_index()
```

In [4]:
```
df = pd.read_csv("Skyserver_SQL6_15_2025 8_20_03 AM.csv", header=1)
print(df.columns)
```

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

In [5]:
```
# Calculating the average specz for each id:
df_grouped = df.groupby("objid").agg({
    "specz": "mean",   # Average redshift for each galaxy
    "ra": "first",     # Keep the first instance of Right Ascension (RA)
    "dec": "first",    # Keep the first instance of Declination (DEC)
    "proj_sep": "first" # Keep the first projected separation value
}).reset_index()

mean_z = df_grouped["specz"].mean()
std_z = df_grouped["specz"].std()

#  Define the redshift limits using the 3σ rule
low_limit = mean_z - 3 * std_z
up_limit = mean_z + 3 * std_z

df_cluster_members = df_grouped[(df_grouped["specz"] >= low_limit) & (df_grouped["s

num_member = len(df_cluster_members)
print(f"Number of galaxies identified in the cluster: {num_member}")
```

```
Number of galaxies identified in the cluster: 91
```

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

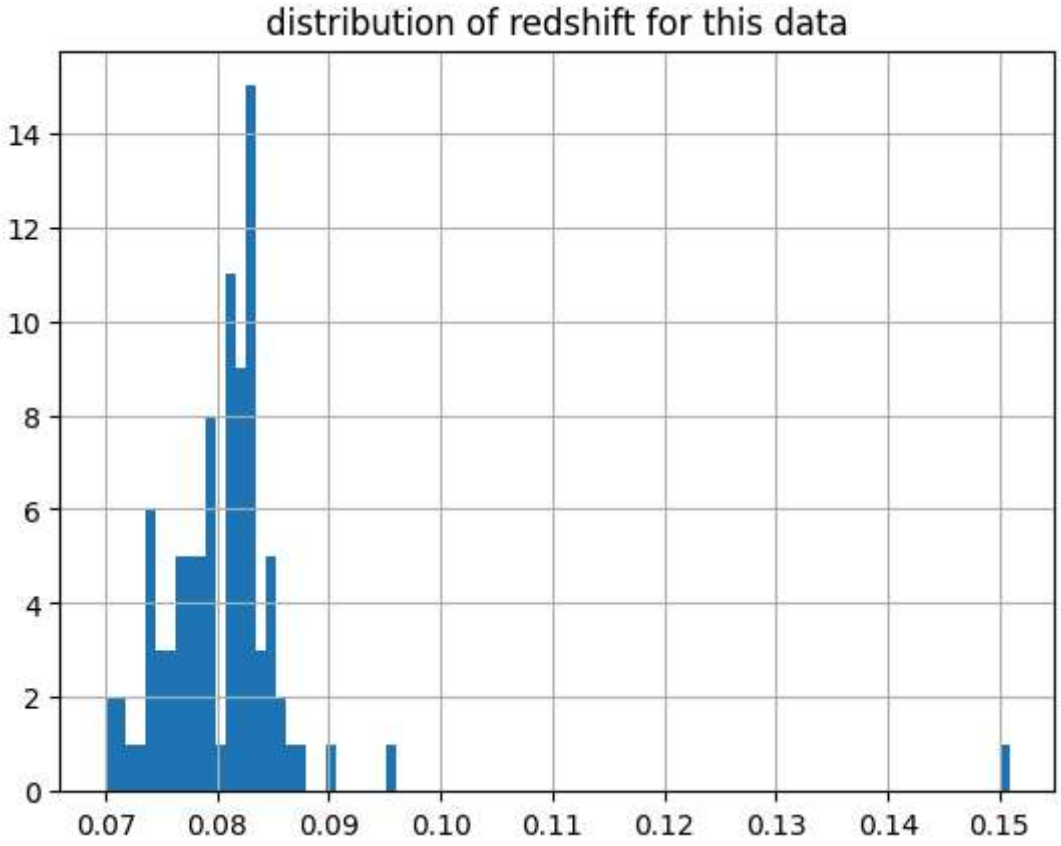
In [6]:
```
df_grouped.describe()['specz']
```

Out[6]:
```
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
```

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

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

In [7]:
```
plt.title("distribution of redshift for this data")
plt.hist(df_grouped['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 [8]:
```
# Filtering the data based on specz values, used 3 sigma deviation from mean as upp
```

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

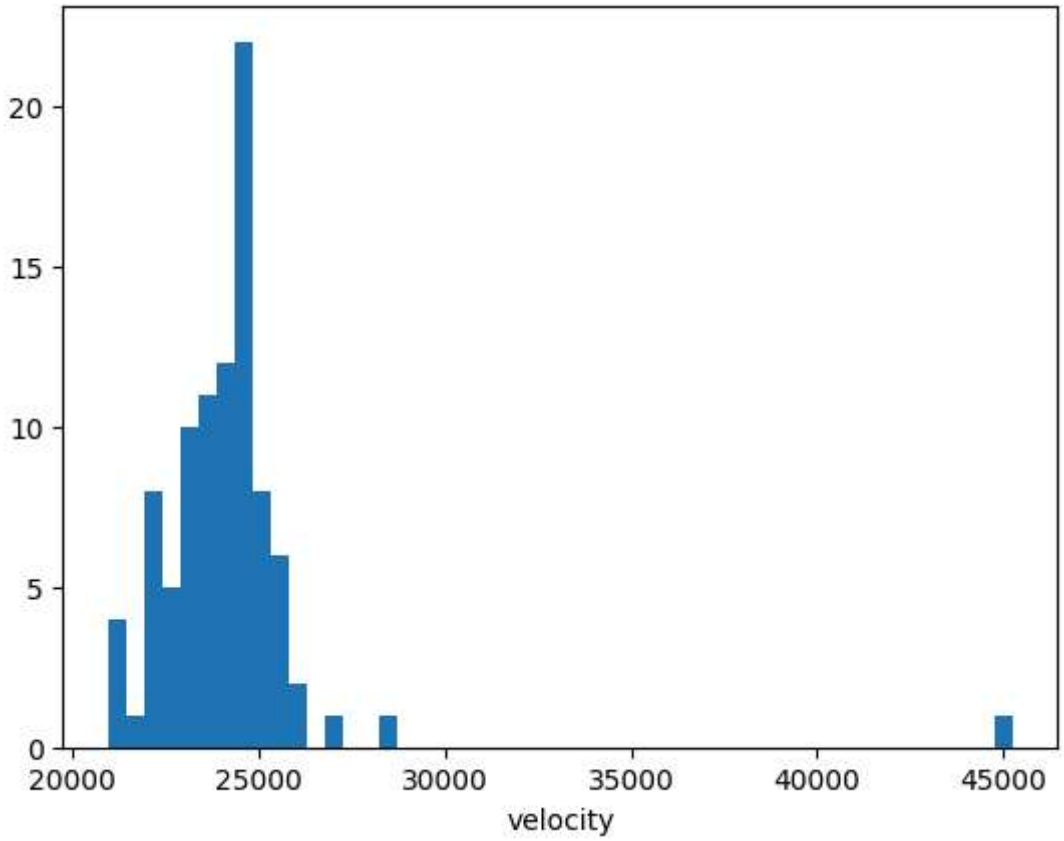
In [9]:
```
df_grouped['velocity']=df_grouped['specz']*c
df_grouped
```

Out[9]:

	objid	specz	ra	dec	proj_sep	velocity
0	1237671768542478711	0.082457	257.82458	64.133257	8.347733	24719.932747
1	1237671768542478713	0.081218	257.83332	64.126043	8.011259	24348.666769
2	1237671768542544090	0.079564	257.85137	64.173247	8.739276	23852.805546
3	1237671768542544107	0.080842	257.89303	64.141138	6.839642	24235.764929
4	1237671768542544127	0.084575	257.91585	64.107290	5.666108	25355.070050
...	...	...	...	...	...	...
87	1237671939804627462	0.082060	258.45078	64.020363	9.483937	24601.083025
88	1237671939804627464	0.071804	258.44994	64.025909	9.316140	21526.443054
89	1237671939804627483	0.080790	258.43205	64.123685	8.146154	24220.238678
90	1237671939804627518	0.082876	258.46676	64.119499	8.986029	24845.704677
91	1237671939804627535	0.080897	258.48871	64.111343	9.483374	24252.431891

92 rows × 6 columns

In [10]:
```
plt.hist(df_grouped['velocity'], bins=50)
plt.xlabel('velocity')
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!!

In [11]:
```
mean_velocity=df_grouped['velocity'].mean()
velocity_dispersion=float(np.sqrt(np.sum((df_grouped['velocity']-mean_velocity)**2)
velocity_dispersion
```

Out[11]:
```
2571.5039597430723
```

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()
```

In [12]:
```
z_cluster= df_cluster_members['specz'].mean()
z=df_cluster_members['specz']
a=(1+z)**2-(1+z_cluster)**2
b=(1+z)**2+(1+z_cluster)**2
v_relv=c*(a/b)
v_relv
```

Out[12]:
```
0      662.365302
1      319.185348
2     -139.779039
3      214.746305
4     1248.541035
          ...
87     552.549373
88   -2302.424337
89     200.381006
90     778.533295
91     230.166253
Name: specz, Length: 91, dtype: float64
```

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 [13]:
```
mean_vrelv=v_relv.mean()
N=len(v_relv) # or N= len(df_cluster_members)
disp=float(np.sqrt(np.sum((v_relv-mean_vrelv)**2)/N))
```

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

In [14]:
```
disp
```

Out[14]:
```
1211.7794338417104
```

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

```
The value of the cluster redshift = 0.08007
The characteristic value of velocity dispersion of the cluster along the line of sig
ht = 1.212e+03 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]:
```
df_grouped['proj_sep']
```

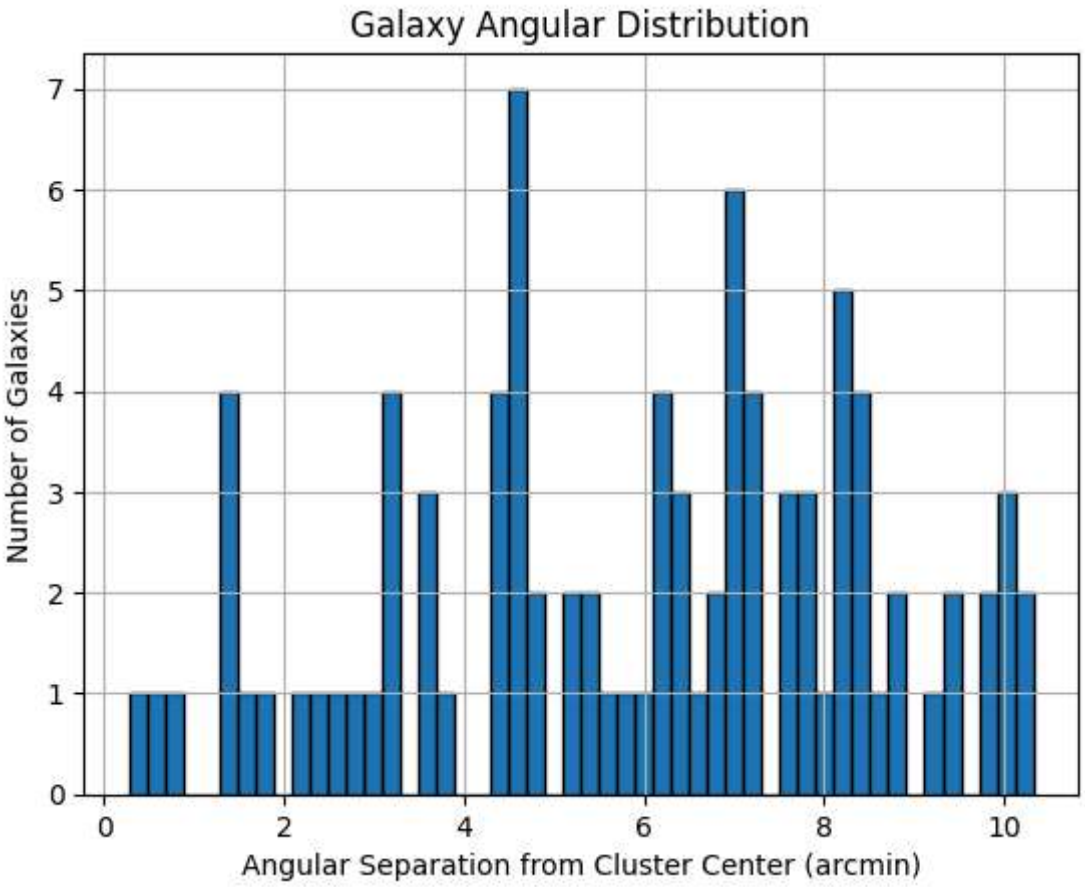
Out[16]:
```
0     8.347733
1     8.011259
2     8.739276
3     6.839642
4     5.666108
        ...
87    9.483937
88    9.316140
89    8.146154
90    8.986029
91    9.483374
Name: proj_sep, Length: 92, dtype: float64
```

In [17]:
```
ra_center = df_cluster_members['ra'].mean()
dec_center = df_cluster_members['dec'].mean()

from astropy.coordinates import SkyCoord
coords = SkyCoord(ra=df_cluster_members['ra'].values * u.deg,
                  dec=df_cluster_members['dec'].values * u.deg)
center = SkyCoord(ra=ra_center * u.deg, dec=dec_center * u.deg)
```

```
separations = coords.separation(center).arcminute
```

In [18]:
```
#Plot histogram for proj sep column
plt.hist(separations, bins=50, edgecolor='black')
plt.xlabel("Angular Separation from Cluster Center (arcmin)")
plt.ylabel("Number of Galaxies")
plt.title("Galaxy Angular Distribution")
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 [19]:
```
z_=z_cluster
q0=-0.534
r=float((c*z_/H_0)*(1-(z_/2)*(1+q0)))
D_A=float(r/(1+z_))
D_A
```

Out[19]:
```
311.5668238308137
```

In [20]:
```
theta_arcmin = float(separations.max())
theta_rad = np.deg2rad(theta_arcmin / 60)
diameter = float(D_A * theta_rad)
R = float(diameter/2)
R
```

Out[20]:
```
0.46763258560115856
```

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 [21]:
```
### Calculating the dynamical mass in solar masses:
M_dyn=(3*(disp**2)*R) / G
```

```
print(f"Dynamical mass: {M_dyn:.2e} M⊙")
```

```
Dynamical mass: 4.79e+14 M⊙
```

Calculations for comparing between dynamical mass and luminous mass

In [22]:
```
num_galaxies = 91

# Assumed average stellar mass per galaxy (in solar masses)
avg_stellar_mass = 1e11  # M⊙

# Total luminous mass
M_lum = num_galaxies * avg_stellar_mass
print(f"Luminous Mass: {M_lum:.2e} M⊙")
```

```
Luminous Mass: 9.10e+12 M⊙
```

In [23]:
```
mass_ratio = M_dyn / M_lum
print(f"Mass Ratio (M_dyn / M_lum): {mass_ratio:.2f}")
```

```
Mass Ratio (M_dyn / M_lum): 52.62
```

Only about 1.9% of the cluster's mass is luminous; the rest, approximately 98.1% can be dark matter and hot intracluster gas, a signature of the hidden universe. The total gravitational pull keeping the cluster together can't be explained by stuff that we can see. Even more intriguing, nearly all of the cluster's mass is invisible, making an important evidence for dark matter.

In [ ]:
```

```