from astroquery.simbad import Simbad

import numpy as np

import pandas as pd

from astroquery.gaia import Gaia

from astropy.coordinates import SkyCoord

from astropy import units as u

from astroquery.vizier import Vizier

import warnings

# Configure query settings

Simbad.TIMEOUT = 1200

Simbad.ROW\_LIMIT = -1

Simbad.add\_votable\_fields('ids', 'otype', 'mesfe\_h') # Updated: 'fe\_h' → 'mesfe\_h'

Vizier.TIMEOUT = 1200

vizier = Vizier(columns=['TIC', 'GAIA', '\_RAJ2000', '\_DEJ2000'])

vizier.ROW\_LIMIT = -1

Gaia.MAIN\_GAIA\_TABLE = "gaiadr3.gaia\_source"

Gaia.ROW\_LIMIT = -1

# Suppress warnings from Simbad (optional, but recommended for clean output)

warnings.filterwarnings('ignore', category=DeprecationWarning, module='astroquery.simbad.core')

def get\_gaia\_and\_metallicity\_from\_simbad(target\_ids):

"""Query SIMBAD for Gaia DR3 IDs and metallicity (Fe/H) with robust error handling."""

s = Simbad.query\_objects(target\_ids)

gaia\_ids = np.full(len(target\_ids), np.nan, dtype=object)

fe\_h\_values = np.full(len(target\_ids), np.nan, dtype=float)

if s is None:

print('No Simbad match found for any target\_ids')

return gaia\_ids, fe\_h\_values

print("Available columns in SIMBAD results:", s.colnames)

for i, entry in enumerate(s):

try:

# Safely get IDS if available

ids = entry['IDS'].split('|') if 'IDS' in entry.colnames else []

# Extract Gaia DR3 ID

for id\_str in ids:

if 'Gaia DR3' in id\_str:

gaia\_ids[i] = id\_str.strip()

break # Take the first Gaia DR3 ID found

# Extract metallicity if available

if 'MESFE\_H' in entry.colnames and not np.isnan(entry['MESFE\_H']):

fe\_h\_values[i] = entry['MESFE\_H']

except Exception as e:

print(f"Error processing entry {i}: {str(e)}")

continue

return gaia\_ids, fe\_h\_values

# Load TOI data

cols1 = ['toi', 'tid', 'pl\_orbper', 'pl\_trandurh', 'pl\_trandep', 'pl\_rade', 'pl\_eqt', 'pl\_insol',

'st\_teff', 'st\_logg', 'st\_rad', 'ra', 'dec', 'tfopwg\_disp']

data = pd.read\_csv('TOI\_2025.08.02\_05.03.17.csv', comment='#', usecols=cols1)

# Count items before/after cleaning

original\_count = len(data)

data = data.dropna()

cleaned\_count = len(data)

print(f"Original CSV entries: {original\_count}")

print(f"Entries after dropna(): {cleaned\_count}")

# Process TOI data

raw\_toi\_df = data.reset\_index(drop=True)

raw\_toi\_df['tid'] = 'TIC ' + raw\_toi\_df['tid'].astype(str)

# Query SIMBAD for Gaia DR3 IDs and metallicity

gaia\_ids, fe\_h\_values = get\_gaia\_and\_metallicity\_from\_simbad(raw\_toi\_df['tid'])

raw\_toi\_df['Gaia\_DR3\_ID'] = gaia\_ids

raw\_toi\_df['Fe\_H'] = fe\_h\_values # Column name remains 'Fe\_H' for clarity

# Report missing Gaia IDs

missing\_gaia = raw\_toi\_df['Gaia\_DR3\_ID'].isnull().sum()

print(f"{missing\_gaia} out of {len(raw\_toi\_df)} TOIs lack Gaia DR3 IDs in SIMBAD")

# Report TOIs with Fe\_H data

fe\_h\_found = (~raw\_toi\_df['Fe\_H'].isnull()).sum()

print(f"{fe\_h\_found} out of {len(raw\_toi\_df)} TOIs have metallicity (Fe\_H) data in SIMBAD")

# Save to new CSV

output\_filename = "TOI\_2025.08.02\_FeH\_Gaia3ID.csv"

raw\_toi\_df.to\_csv(output\_filename, index=False)

print(f"Results saved to {output\_filename}")

**RESULTS:**

Original CSV entries: 7658

Entries after dropna(): 6586

Available columns in SIMBAD results: ['main\_id', 'ra', 'dec', 'coo\_err\_maj', 'coo\_err\_min', 'coo\_err\_angle', 'coo\_wavelength', 'coo\_bibcode', 'otype', 'ids', 'mesfe\_h.bibcode', 'mesfe\_h.catno', 'mesfe\_h.compstar', 'mesfe\_h.fe\_h', 'mesfe\_h.fe\_h\_prec', 'mesfe\_h.flag', 'mesfe\_h.log\_g', 'mesfe\_h.log\_g\_prec', 'mesfe\_h.mespos', 'mesfe\_h.teff', 'user\_specified\_id', 'object\_number\_id']

6586 out of 6586 TOIs lack Gaia DR3 IDs in SIMBAD

0 out of 6586 TOIs have metallicity (Fe\_H) data in SIMBAD

Results saved to TOI\_2025.08.02\_FeH\_Gaia3ID.csv