In [1]:
```
import re
import exodata
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import exodata.astroquantities as aq
```

# Define Functions

In [2]:
```
def listToString(s):
    """pass in list, returns string"""
    str1 = ","
    return (str1.join(s))
```

In [3]:
```
def findOtherName(check, primary, alts):
    """extract other names & ID number and arrange them in formatting that matches curr
    ex: stars['altnames'] = ['11 com b', 'Gliese 234', 'HD137'] -> stars['HD'] = '137'"
    name = ""
    if primary.startswith(check):
        name = primary[len(check) + 1:]
    if name == "" and alts == "":
        return np.nan
    for i in range(len(alts)):
        if alts[i].startswith(check):
            name = alts[i][len(check) + 1:]
    if name == "":
        return np.nan
    while re.search("\D", name):
        name = name[:-1]
    return float(name)
```

In [4]:
```
def findGLName(primary, alts):
    """extract Gliese name & ID number and arrange them in formatting that matches curr
    ex: stars['altnames'] = ['11 com b', 'Gliese 234', 'HD137'] -> stars['GL'] = 'GL 23
    prefixes = ["GL ", "Gliese ", "NN ", "WO ", "GJ "]
    name = ""
    for i in range(len(prefixes)):
        if primary.startswith(prefixes[i]):
            name = primary
            break
    if name == "" and alts != "":
        for i in range(len(alts)):
            for j in range(len(prefixes)):
                if alts[i].startswith(prefixes[j]):
                    name = alts[i]
                    break
            if name != "":
                break
    if name == "":
        return np.nan
    if name.startswith("GL"):
        name = "Gl" + name[2:]
    elif name.startswith("WO"):
```

```
        name = "Wo" + name[2:]
    elif name.startswith("Gliese"):
        name = "GJ" + name[6:]
    return name
```

In [5]:
```
def getListOfNames(names):
    """Pass in a string from dataframe that has names surrounded by parentheses you'd l
    ex: stars['name'] = Star('11 com b') -> stars['name'] = '11 com b'"""
    temp = ""
    while len(names) > 2:
        start = names.find('(') + 2
        end = names.find(')') - 1
        temp += names[start:end] if len(temp) == 0 else ", " + names[start:end]
        names = names[end + 2 :]
    return temp
```

In [6]:
```
def isNaN(x):
    """pass in a value you'd like to check to see if it is not a number, not to be conf
    try:
        float(x)
    except:
        return True
    return False
```

In [7]:
```
def catagorize(columns, name):
    """Given a list of columns and a name, adds all columns into single column under ne
    data[name] = data[columns[0]]
    for item in columns[1:]:
        data[name] = data[name] + data[item]
```

# Read data

In [8]:
```
# Load the most current data from the Open Exoplanet Catalouge from db url for most up
exocat = exodata.load_db_from_url('https://github.com/OpenExoplanetCatalogue/oec_gzip/r
star_csv = pd.read_csv('data/hygdata_v3.csv')
```

```
WARNING: IllegalSecondWarning: 'second' was found  to be '60', which is not in range [0,
60). Treating as 0 sec, +1 min [astropy.coordinates.angle_utilities]
rejected duplicate satellite:
                                in Jupiter
rejected duplicate satellite:
                                in Jupiter
rejected duplicate satellite:
                                in Jupiter
rejected duplicate satellite:
                                in Saturn
rejected duplicate satellite:
                                in Saturn
rejected duplicate satellite:
                                in Saturn
rejected duplicate satellite:
                                in Saturn
rejected duplicate satellite:
                                in Saturn
```

```
rejected duplicate satellite:
                                in Uranus
rejected duplicate satellite:
                                in Uranus
rejected duplicate satellite:
                                in Uranus
rejected duplicate satellite:
                                in Uranus
```

In [9]:

```
# assign names to tree branches we'll be using
planets = exocat.planets
stars = exocat.stars
```

In [10]:

```
# to remove every bit of info from the xml and transfer it to a dataframe, first we'll
star_dict = {}

#then we iterate over each element in the branch and fill the dictionaries with the inf
#in both callable methods, and within a dictionary within one of those methods, which m
#also the module is written to raise errors instead of returning nothing, so everything
```

In [11]:

```
# don't seem to be able to parse through the xml without it being in try/excepts
i = 0
while i < 3505:
    star_dict[i] = {}
    try:
        star_dict[i]['spectral_type'] = stars[i].params['spectraltype']
    except:
        star_dict[i]['spectral_type'] = np.nan
    try:
        star_dict[i]['temp'] = stars[i].params['temperature']
    except:
        star_dict[i]['temp'] = np.nan
    try:
        star_dict[i]['metallicity'] = stars[i].params['metallicity']
    except:
        star_dict[i]['metallicity'] = np.nan
    try:
        star_dict[i]['altnamesstr'] = listToString(stars[i].params['altnames'])
    except:
        star_dict[i]['altnames'] = ''
    try:
        star_dict[i]['altnames'] = stars[i].params['altnames']
    except:
        star_dict[i]['altnames'] = np.nan
    try:
        star_dict[i]['mass'] = stars[i].params['mass']
    except:
        star_dict[i]['mass'] = np.nan
    try:
        star_dict[i]['magUltraviolet'] = stars[i].params['magU']
    except:
        star_dict[i]['magUltraviolet'] = np.nan
    try:
        star_dict[i]['magBlue'] = stars[i].params['magB']
    except:
        star_dict[i]['magBlue'] = np.nan
    try:
        star_dict[i]['magH_nearinfared'] = stars[i].params['magH']
```

```
        except:
            star_dict[i]['magH_nearinfared'] = np.nan
        try:
            star_dict[i]['magInfared'] = stars[i].params['magI']
        except:
            star_dict[i]['magInfared'] = np.nan
        try:
            star_dict[i]['magJ_nearinfared'] = stars[i].params['magJ']
        except:
            star_dict[i]['magJ_nearinfared'] = np.nan
        try:
            star_dict[i]['magK_nearinfared'] = stars[i].params['magK']
        except:
            star_dict[i]['magK_nearinfared'] = np.nan
        try:
            star_dict[i]['magVisual'] = stars[i].params['magV']
        except:
            star_dict[i]['magVisual'] = np.nan
        try:
            star_dict[i]['magL_nq_midinfared'] = stars[i].params['magL']
        except:
            star_dict[i]['magL_nq_midinfared'] = np.nan
        try:
            star_dict[i]['magM_midinfared'] = stars[i].params['magM']
        except:
            star_dict[i]['magM_midinfared'] = np.nan
        try:
            star_dict[i]['magN_midinfared'] = stars[i].params['magN']
        except:
            star_dict[i]['magN_midinfared'] = np.nan
        try:
            star_dict[i]['distance'] = stars[i].d
        except:
            star_dict[i]['distance'] = np.nan
        try:
            star_dict[i]['periastron'] = stars[i].params['periastron']
        except:
            star_dict[i]['periastron'] = np.nan
        try:
            star_dict[i]['right_ascension'] = stars[i].ra
        except:
            star_dict[i]['right_ascension'] = np.nan
        try:
            star_dict[i]['declination'] = stars[i].dec
        except:
            star_dict[i]['declination'] = np.nan
        try:
            star_dict[i]['parent_obj'] = getListOfNames(str(stars[i].parent))
        except:
            star_dict[i]['parent_obj'] = np.nan
        try:
            star_dict[i]['child_obj'] = getListOfNames(str(stars[i].children))
        except:
            star_dict[i]['child_obj'] = np.nan
        try:
            star_dict[i]['planet1type'] = stars[i].children[0].type()
        except:
            star_dict[i]['planet1type'] = np.nan
        try:
            star_dict[i]['planet2type'] = stars[i].children[1].type()
        except:
```

```
        star_dict[i]['planet2type'] = np.nan
    try:
        star_dict[i]['planet3type'] = stars[i].children[2].type()
    except:
        star_dict[i]['planet3type'] = np.nan
    try:
        star_dict[i]['planet4type'] = stars[i].children[3].type()
    except:
        star_dict[i]['planet4type'] = np.nan
    try:
        star_dict[i]['planet5type'] = stars[i].children[4].type()
    except:
        star_dict[i]['planet5type'] = np.nan
    try:
        star_dict[i]['planet6type'] = stars[i].children[5].type()
    except:
        star_dict[i]['planet6type'] = np.nan
    try:
        star_dict[i]['planet7type'] = stars[i].children[6].type()
    except:
        star_dict[i]['planet7type'] = np.nan
    try:
        star_dict[i]['planet8type'] = stars[i].children[7].type()
    except:
        star_dict[i]['planet8type'] = np.nan
    try:
        star_dict[i]['planet9type'] = stars[i].children[8].type()
    except:
        star_dict[i]['planet9type'] = np.nan
    star_dict[i]['proper'] = stars[i].name
    star_dict[i]['flags'] = stars[i].flags
    star_dict[i]['system'] = getListOfNames(str(stars[i].system))
    star_dict[i]['radius'] = stars[i].R
    star_dict[i]['age'] = stars[i].age
    star_dict[i]['hip'] = findOtherName("HIP ", star_dict[i]['proper'], star_dict[i]['a
    star_dict[i]['hd'] = findOtherName("HD ", star_dict[i]['proper'], star_dict[i]['alt
    star_dict[i]['hr'] = findOtherName("HR ", star_dict[i]['proper'], star_dict[i]['alt
    star_dict[i]['gl'] = findGLName(star_dict[i]['proper'], star_dict[i]['altnames'])
    i += 1
```

```
C:\ProgramData\anaconda3\envs\learn-env\lib\site-packages\quantities\quantity.py:424: Ru
ntimeWarning: invalid value encountered in greater
  return self.magnitude > other
C:\ProgramData\anaconda3\envs\learn-env\lib\site-packages\quantities\quantity.py:391: Ru
ntimeWarning: invalid value encountered in less
  return self.magnitude < other
```

In [12]:

```
# transform all of my dictionaries to dataframes so I can work with them in pandas
sdf = pd.DataFrame(star_dict)
```

In [13]:

```
#realign them so my columns are on top and rows go downward
stars = sdf.transpose()
```

# Examine data

Figure out what you might need to clean up

In [14]:
```
#clean up stars to correctly cast to dtypes
stars['radius'] = stars['radius'].str.strip(to_strip=" R_s")
stars['age'] = stars['age'].str.strip(" Gyr")
stars['mass'] = stars['mass'].str.strip(" M_s")
stars['distance'] = stars['distance'].str.strip(" pc")
stars['temp'] = stars['temp'].str.strip(" K")
```

In [15]:
```
# drop columns with little to no information
stars = stars.drop(['magL_nq_midinfared', 'magM_midinfared', 'magN_midinfared', 'perias
```

In [16]:
```
#recast everything as dtypes we can work with
stars['radius'] = stars['radius'].astype(float)
stars['age'] = stars['age'].astype(float)
stars['temp'] = stars['temp'].astype(float)
stars['mass'] = stars['mass'].astype(float)
stars['distance'] = stars['distance'].astype(float)
```

In [17]:
```
# counting the number of planets we have around each star
child = stars['child_obj'].astype(str)
childDict = {}
for i in range(len(child)):
    childDict[i] = {}
    if child[i] == '':
        childDict[i]['children'] = 0.0
    else:
        childDict[i]['children'] = float(child[i].count(',') + 1.0)
childDict = pd.DataFrame.from_dict(childDict, orient='index', dtype=float)
stars = childDict.merge(stars, right_index=True, left_index=True)
```

In [18]:
```
# I want to know how many total of each variety of planet each star has, to see if we c
# its features. I have a maximum of 8 planets in any solar system (excluding pluto whic
columns = {'Cold Jupiter':'CJ', 'Cold Neptune':'CN', 'Cold Super-Earth':'CE', 'Hot Jupi
           'Hot Neptune':'HN', 'Hot Super-Earth':'HSE', 'None Jupiter':'JUP', 'None Nep
           'None Super-Earth':'SE', 'Warm Jupiter':'WJ', 'Warm Neptune':'WN', 'Warm Sup

dumms1 = pd.get_dummies(stars['planet1type'])
dumms1 = dumms1.rename(columns=columns)
dumms2 = pd.get_dummies(stars['planet2type'])
dumms2 = dumms2.rename(columns=columns)
dumms3 = pd.get_dummies(stars['planet3type'])
dumms3 = dumms3.rename(columns=columns)
dumms4 = pd.get_dummies(stars['planet4type'])
dumms4 = dumms4.rename(columns=columns)
dumms5 = pd.get_dummies(stars['planet5type'])
dumms5 = dumms5.rename(columns=columns)
dumms6 = pd.get_dummies(stars['planet6type'])
dumms6 = dumms6.rename(columns=columns)
dumms7 = pd.get_dummies(stars['planet7type'])
dumms7 = dumms7.rename(columns=columns)
dumms8 = pd.get_dummies(stars['planet8type'])
dumms8 = dumms8.rename(columns=columns)
```

In [19]:
```
d1 = dumms1.merge(dumms2, left_index=True, right_index=True, suffixes=('1', '2'))
```

```
d2 = d1.merge(dummms3, left_index=True, right_index=True)
d3 = d2.merge(dummms4, left_index=True, right_index=True, suffixes=('3', '4'))
d4 = d3.merge(dummms5, left_index=True, right_index=True)
d5 = d4.merge(dummms6, left_index=True, right_index=True, suffixes=('6', '6'))
d6 = d5.merge(dummms7, left_index=True, right_index=True)
pln_types = d6.merge(dummms8, left_index=True, right_index=True, suffixes=('7', '8'))
pln_types = pln_types.astype(float)
```

In [20]:

```
# condense into catagories for each planet type
pln_types['CJ'] = pln_types['CJ1'] + pln_types['CJ2'] + pln_types['CJ3'] + pln_types['C
pln_types['CN'] = pln_types['CN1'] + pln_types['CN2'] + pln_types['CN3'] + pln_types['C
pln_types['CE'] = pln_types['CE1'] + pln_types['CE2'] + pln_types['CE3'] + pln_types['C
pln_types['HE'] = pln_types['HE']
pln_types['HN'] = pln_types['HN1'] + pln_types['HN2']
pln_types['HSE'] = pln_types['HSE1'] + pln_types['HSE2'] + pln_types['HSE3']
pln_types['JUP'] = pln_types['JUP1'] + pln_types['JUP2'] + pln_types['JUP3']
pln_types['NEP'] = pln_types['NEP1'] + pln_types['NEP2'] + pln_types['NEP']
pln_types['SE'] = pln_types['SE1'] + pln_types['SE2'] + pln_types['SE3'] + pln_types['S
pln_types['WJ'] = pln_types['WJ1'] + pln_types['WJ2'] + pln_types['WJ3'] + pln_types['W
pln_types['WN'] = pln_types['WN1'] + pln_types['WN2'] + pln_types['WN3'] + pln_types['W
pln_types['WSE'] = pln_types['WSE1'] + pln_types['WSE2'] + pln_types['WSE3'] + pln_type
```

In [21]:

```
# get rid of all of the excess planet types
pln_types = pln_types.drop(columns=['CJ1', 'CN1', 'CE1', 'HN1', 'HSE1', 'JUP1', 'NEP1',
                    'CJ2', 'CN2', 'CE2', 'HN2', 'HSE2', 'JUP2', 'NEP2', 'SE2', 'WJ2',
                    'CN3', 'CE3', 'HSE3', 'JUP3', 'SE3', 'WJ3', 'WN3', 'WSE3', 'CJ4',
                    'HSE4', 'JUP4', 'SE4', 'WJ4', 'WN4', 'WSE4', 'CJ6', 'CN_x', 'CE6',
                    'CE6', 'SE6', 'WSE6', 'CJ7', 'CN_y', 'WSE7', 'CJ8', 'WSE8'])
```

In [22]:

```
# merge your dummies frame with your main frame
stars = stars.merge(pln_types, right_index=True, left_index=True)
```

In [24]:

```
#  make seperate frames for each of your name types, will try to merge on each of them
HIP = stars.loc[stars['hip'] > 0]
HD = stars.loc[stars['hd'] > 0]
HR = stars.loc[stars['hr'] > 0]
GL = stars.loc[stars['gl'].notnull()]
proper = star_csv.loc[star_csv['proper'].notnull()]
```

In [25]:

```
# merge on Id for each name type, we don't have a match for every star so we only end u
GL_csv = GL.merge(star_csv, on='gl')
HD_csv = HD.merge(star_csv, on='hd')
HR_csv = HR.merge(star_csv, on='hr')
HIP_csv = HIP.merge(star_csv, on='hip')
stars_csv = stars.merge(proper, on='proper')
```

In [26]:

```
# add all of your frames back together & drop the duplicates (we know most stars have m
HIP_csv = HIP_csv.append(GL_csv)
HIP_csv = HIP_csv.append(HD_csv)
HIP_csv = HIP_csv.append(stars_csv)
HIP_csv = HIP_csv.append(HR_csv)
HIP_csv = HIP_csv.drop_duplicates('id')
```

In [27]:
```
# the readme that this data comes with says that it replaced null values with 1,000,000
HIP_csv['dist'] = HIP_csv['dist'].replace(100000, np.nan)
HIP_csv['dist'] = HIP_csv.fillna(HIP_csv['distance'])
```

In [28]:
```
# new estimates roughly 1 in 4 sunlike stars have planets, adjusting for how much data
data = HIP_csv.append(star_csv.sample(3000, random_state=42))
```

In [29]:
```
data
```

Out[29]:

| | children | spectral_type | temp | metallicity | altnamesstr | altnames | mass | magUlt |
|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | G8 III | NaN | -0.35 | 11 Comae Berenices,HD 107383,HIP 60202,TYC 144... | [11 Comae Berenices, HD 107383, HIP 60202, TYC... | NaN | |
| **1** | 1.0 | K4III | NaN | 0.04 | 11 Ursae Minoris,Pherkard,Pherkad Minor,HD 136... | [11 Ursae Minoris, Pherkard, Pherkad Minor, HD... | NaN | |
| **2** | 1.0 | K0III | NaN | -0.24 | 14 Andromedae,HD 221345,HIP 116076,TYC 3231-32... | [14 Andromedae, HD 221345, HIP 116076, TYC 323... | NaN | |
| **3** | 2.0 | K0 V | NaN | 0.43 | HD 145675,HIP 79248,TYC 3067-576-1,SAO 45933,G... | [HD 145675, HIP 79248, TYC 3067-576-1, SAO 459... | NaN | |
| **4** | 0.0 | G2V | NaN | 0.096 | 16 Cyg A,HD 186408,HIP 96895,TYC 3565-1524-1,S... | [16 Cyg A, HD 186408, HIP 96895, TYC 3565-1524... | NaN | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **117519** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **66049** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **37428** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **39674** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **54328** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

3979 rows × 92 columns

In [30]:

```
# now that we've added more data that didn't have these columns they'll all by NaN
list_ = ['HE', 'NEP', 'CE', 'CN', 'CJ', 'HN', 'HSE', 'JUP', 'SE', 'WJ', 'WN', 'WSE', 'c
for item in list_:
    data[item] = data[item].fillna(0.0)
```

In [31]:

```
# drop unnecessary columns & 'NaN' values, if you drop too many, just sample more rando
# relitively proportionate
data = data[['children', 'ra', 'dec', 'dist', 'pmra', 'pmdec', 'rv', 'mag', 'absmag', '
             'decrad', 'pmrarad', 'pmdecrad','comp_primary', 'lum','HE', 'NEP', 'CE', '
data = data.dropna()
```

In [32]:

```
# spectral type is another catagorical variable that we've got to do something about, u
# theyre a mess and written too inconsistently to easily make dummies of
data['temp_class'] = data['spect'].str[:1]
data['temp_class'] = data['temp_class'].fillna('nan')

data['heat_class'] = data['spect'].str[1:2]
data['heat_class'] = data['heat_class'].fillna('nan')

data['lum_class'] = data['spect'].str[2:]
data['lum_class'] = data['lum_class'].fillna('nan')
```

In [33]:

```
# making dummies adds a few hundred columns, most of which with only one star in it
data = data.reset_index()
data = pd.get_dummies(data, columns=['temp_class'], prefix='_y')
data = pd.get_dummies(data, columns=['heat_class'], prefix='_x')
data = pd.get_dummies(data, columns=['lum_class'], prefix='')
```

In [34]:

```
# to fix that, we'll be grouping all of our stars together by spectral type & splitting
to_lum = [
            ['I' ,
                 ['_.5Ib', '_/B8Ib', '_Ia', '_Ib', '_Ib/II', '_.5Iab:ne', '_Iab:var', '
                  '_Ib+...', '_-F8Ib', '_Ia/ab', '_Iab']],
            ['II',
                 ['_Ib/II', '_Ib-II', '_/B9II/III', '_/K0II', '_II', '_II-III', '_II/II
                    '_II/IIICN', '_IICN...', '_/2II/III+A', '_/F2III', '_II + A/F', '_
            ['III',
                 ['_.5III', '_.5III:', '_/G5III', '_/G8III', '_/B8III', '_/B9II/III', '
                  '_/K0III', '_/K0III:', '_/K1III', '_/K1III+..', '_/K2III', '_/K3III',
                  '_/M0III', '_/M1III', '_/M3III', '_:III:', '_III+...', '_II-III', '_/
                  '_/M2III', '_/2II/III+A', '_/K2III:', '_/M1III:', '_III comp', '_/F5I
                  '_III + (F)', '_III-IV', '_III...', '_III/IV', '_IIICNII', '_IIICNp..
                  '_/G6III', '_/K0III+..', '_/K0IIICNp', '_IIIb', '_IIIsp...', '_IIIp',
            ['IV',
                 ['_.5IV', '_/B9III/IV', '_/A3IV', '_/A3IV/V', '_/A7IV', '_/F2IV', '_/F
                  '_IV', '_IV:pe...', '_IVne+...', '_/F6IV/V', '_/F8IV+...', '_/G8IV/V'
                  '_IVn', '_.5IV-V', '_/A4IV', '_IV-V', '_/K1IV', '_III-IV', '_IV/V', '
                  '_IV: (+F/G)', '_/F3IV', '_/B3IV', '_/F3IV/V', '_/F5IV/V', '_/F5IV',
            ['V',
                 ['_ V', '_.5V', '_.5Ve', '_.5Vn', '_/A1V', '_/A2V', '_/A3IV/V', '_Vm',
                  '_/B3V', '_/F2IV/V', '_/G0V', '_/K0IV/V', '_V + G/K', '_Ve', '_Vvar',
                  '_/F3V', '_/G1V', '_/K0V',  '_Ve+...', '_/B5V', '_/F5V', '_/G2V', '_/
                  '_/G3V', '_V...', '_Vne', '_/F3IV/V', '_/K3V', '_/B8V', '_/F7IV/V', '
                  '_/B9V', '_Vp...', '_/F7V', '_/F8V', '_/G6V', '_/G8V', '_/B9.5V', '_:
                  '_Vw...', '_/F6IV/V', '_/G8IV/V', '_V comp', '_VCN...', '_.5IV-V']],
```

```
            ['VI',
                 ['_V-VI']]]

to_heat = [
            ['0',
                 ['_x_0', '_/G0V', '_/G0Vs...', '_/K0II', '_/K0III', '_/K0III:', '_/K0I
                  '_0', '_Ia0:', '_/K0III+..', '_/K0IIICNp']],
            ['1',
                 ['_x_1', '_/A1V', '_/G1V', '_/G2V', '_/K1III', '_/K1III+..', '_/K1IV',
            ['2',
                 ['_x_2', '_-G2Ie', '_/A2V', '_/F2IV', '_/F2IV/V', '_/K2III', '_/M2V',
                  '_/K2III:']],
            ['3',
                 ['_x_3', '_/A3III', '_/A3IV', '_/A3IV/V', '_/A3V', '_/A3V+...', '_/B3V
                  '_/K3V', '_/M3III', '_3', '_3   :', '_/F3IV', '_/B3IV', '_/F3IV/V', '
            ['4',
                 ['_x_4', '_/K4', '_/K4III', '_/K4III:', '_/A4IV']],
            ['5',
                 ['_x_5', '_/F5V', '_/G5III', '_/G5V', '_/G5Vw...', '_5', '_/F5IV', '_/
            ['6',
                 ['_x_6', '_/F6V', '_/G6V', '_6', '_6 (SB1)', '_/F6IV/V', '_/G6III']],
            ['7',
                 ['_x_7', '_/A7IV', '_/F7IV/V', '_/F7V', '_7', '_e-M7e']],
            ['8',
                 ['_x_8', '_/B8III', '_/B8Ib', '_/B8V', '_/F8V', '_/G8III', '_/G8III:',
                  '_e-M8e', '_/A8III', '_-F8Ib', '_/G8IV/V']],
            ['9',
                 ['_x_9', '_/B9II/III', '_/B9III/IV', '_/B9V', '_9', '_9?', '_e-M9e', '

to_temp = [
            ['A',
                 ['_y_A', '_x_A', '_/A1V', '_/A2V', '_/A3III', '_/A3IV', '_/A3IV/V', '_
                 ]],
            ['B',
                 ['_/B3V', '_/B8III', '_/B8Ib', '_/B8V', '_/B9II/III', '_/B9III/IV', '_
            ['C',
                 ['_y_C', '_x_C']],
            ['F',
                 ['_/F2IV', '_/F2IV/V', '_/F3', '_/F3V', '_/F5V', '_/F6V', '_/F6IV/V',
                  '_/F3IV', '_/F5IV', '_-F8Ib', '_/F2III', '_/F2V', '_/F3IV/V', '_/F5IV
                  '_/F5III', '_/F5IV/V', '_/F6IV/V', '_/F8IV+...', '_F:']],
            ['G',
                 ['_y_G', '_-G2Ie', '_/G0V', '_/G1V', '_/G2V', '_/G3V', '_/G5III', '_/G
                  '_/G8w...', '_G2', '_/G6III', '_/G8IV/V']],
            ['K',
                 ['_y_K', '_x_K', '_/K0II', '_/K0III', '_/K0III:', '_/K0IV/V', '_/K0V',
                  '_/K1IV', '_/K1V', '_/K2III', '_/K1IV/V:', '_/K3III', '_/K3III:', '_/
                  '_/K0IIICNp', '_/K2III:', '_/K3V:+...', '_/K0III+..']],
            ['N',
                 ['_y_N', '_x_N']],
            ['M',
                 ['_y_M', '_x_M', '_/M0III', '_/M1III', '_/M2V', '_/M3III', '_e-M8e', '
            ['m',
                 ['_m', '_m...', '_mp', '_Vm', '_x_m']],
            ['RD',
                 ['_y_R']],
            ['sd',
                 ['_y_s', '_x_d', '_V-VI']],
            ['W',
                 ['_y_W']],
            ['O',
```

```
                ['_y_O', '_/O8', '_O:', '_Ia0:']],
            ['D',
                ['_y_D']],
            ['p',
                ['_V:pe', '_Vp', '_Vp...', '_Vpe', '_p', '_p...', '_psh', '_sp...', '_:
                 '_/K0IIICNp']],
            ['n',
                ['_.5Vn', '_IVn', '_V:n', '_Vn', '_Vne', '_x_n', '_IVne+...', '_npe',
            ['e',
                ['_IV:pe...', '_V:pe', '_Ve', '_Ve+...', '_Vne', '_e', '_e-M7e', '_.5I
                 '_-G2Ie', '_.5Ve', '_ev']]]
to_symb = [['...',
                ['_+...', '_..', '_...', '_w...', '_sp...', '_V...', '_p...', '_m...',
                 '_IICNp...', '_:w...', '_III+...', '_/K0p...', '_/G0Vs...', '_IVne+..
                 '_:III:+...', '_:+...', '_/K1III+..', '_/G5Vw...', '_/A3V+...', '_/G8
                 '_V+...', '_VCN...', '_IIp...', '_Vp...', '_IICN...', '_IV...', '_/K0
           [':',
                ['_.5III:', '_/G8III:', '_/K0III:', '_:+...', '_:III:+...', '_:w...',
                 '_II/IIICNV:', '_III:', '_:III:', '_Ia0:', '_IV: (+F/G)', '_IV:pe...'
                 '_.5Iab:ne', '_/K2III:', '_/K3V:+...', '_/M1III:', '_:IVp', '_:Vw...'
           ['+',
                ['_+...', '_/A3V+...', '_/K0V + A/F', '_Ve+...', '_:III:+...', '_/2II/
                 '_/K3V:+...', '_II + A/F', '_V+...', '_IVne+...', '_Ib+...', '_/F8IV+
```

In [35]:

```
# turning our big ol lists into combined columns
to_cat = [to_temp, to_heat, to_lum, to_symb]

for item in to_cat:
    for i in range(len(item)):
                catagorize(item[i][1], item[i][0])
```

In [36]:

```
# Taking only the columns we need is easier than trying to figure out which few hundred
data = data[['children', 'ra', 'dec', 'dist', 'pmra', 'pmdec', 'rv', 'mag', 'absmag',
             'ci', 'x', 'y', 'z', 'vx', 'vy', 'vz', 'rarad', 'decrad', 'pmrarad', 'pmde
             'comp_primary', 'lum', 'HE', 'NEP', 'CE', 'CN', 'CJ', 'HN', 'HSE', 'JUP',
             'WJ', 'WN', 'WSE', 'I', 'II', 'III', 'IV', 'V', 'VI', '...', ':', '+', '0'
             '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'F', 'G', 'K', 'N',
             'M', 'RD', 'sd', 'W', 'O', 'D', 'n', 'e', 'p', 'm']].copy()
```

In [37]:

```
#convert everything to floats to get it ready to model
data['ra'] = data['ra'].astype(float)
data['dec'] = data['dec'].astype(float)
data['dist'] = data['dist'].astype(float)
data['pmra'] = data['pmra'].astype(float)
data['pmdec'] = data['pmdec'].astype(float)
data['rv'] = data['rv'].astype(float)
data['mag'] = data['mag'].astype(float)
data['absmag'] = data['absmag'].astype(float)
data['x'] = data['x'].astype(float)
data['y'] = data['y'].astype(float)
data['z'] = data['z'].astype(float)
data['vx'] = data['vx'].astype(float)
data['vz'] = data['vz'].astype(float)
data['vy'] = data['vy'].astype(float)
data['rarad'] = data['rarad'].astype(float)
data['decrad'] = data['decrad'].astype(float)
data['pmrarad'] = data['pmrarad'].astype(float)
```

```
data['pmdecrad'] = data['pmdecrad'].astype(float)
data['comp_primary'] = data['comp_primary'].astype(float)
data['lum'] = data['lum'].astype(float)
data['ci'] = data['ci'].astype(float)
```

In [38]:

```
#export so we don't have to keep re-running these cells!
data.to_csv('scrubbed.csv')
```