In [49]:

```
import astropy
from astropy.io import fits
from astropy.table import Table
import matplotlib.pyplot as plt
import numpy as np
from astropy import constants as con
from astropy import units as u
from astropy.io import ascii
import pandas as pd
from scipy.interpolate import interp1d
from mpl_toolkits import mplot3d
from matplotlib import animation
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

In [50]:

```
yilun = ascii.read('spt_to_teff.txt')
np.unique(yilun[0])
```

Out[50]:

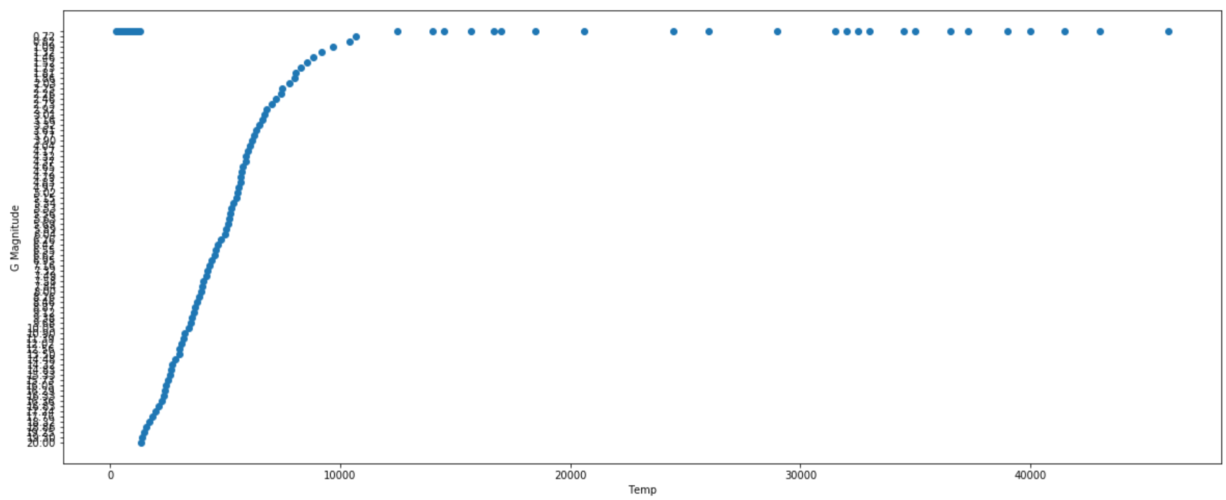
```
array([('O3V', 46000, 4.663, '5.80', '-9.75', '-4.05', '-5.7', '-0.3
2', '...', '...', '...', '...', '...', '.....', '-1.22', '...',
'.....', '.....', '.....', '.....', '...', '...', '...', '...',
'....', '....', '...', '...', '12.5', '....', 'O3V')],
      dtype=[('SpT', '<U5'), ('Teff', '<i8'), ('logT', '<f8'), ('log
L', '<U5'), ('Mbol', '<U5'), ('BCv', '<U5'), ('Mv', '<U5'), ('B-V', '<
U6'), ('Bt-Vt', '<U6'), ('G-V', '<U6'), ('Bp-Rp', '<U6'), ('G-Rp', '<U
6'), ('M_G', '<U5'), ('b-y', '<U6'), ('U-B', '<U6'), ('V-Rc', '<U6'),
('V-Ic', '<U6'), ('V-Ks', '<U6'), ('J-H', '<U6'), ('H-Ks', '<U6'), ('K
s-W1', '<U5'), ('W1-W2', '<U6'), ('W1-W3', '<U6'), ('W1-W4', '<U6'),
('M_J', '<U6'), ('M_Ks', '<U5'), ('i-z', '<U4'), ('z-Y', '<U4'), ('R_R
sun', '<U5'), ('Msun', '<U5'), ('#SpT', '<U5')])
```

In [51]:

```
#gband mag as a funct of temp
plt.figure(figsize=(20,8))
temp = yilun['Teff']
m_gorl = yilun['M_G']
plt.gca().invert_yaxis()
plt.scatter(temp, m_gorl)
plt.xlabel('Temp')
plt.ylabel('G Magnitude')
#plt.hlines(0,0,40000)
```

Out[51]:

```
Text(0, 0.5, 'G Magnitude')
```



In [52]:

```
len(temp)
```

Out[52]:

```
124
```

In [53]:

```
fixed_temp = np.array([])
fixed_mgorl = np.array([])
for i in np.arange(0, 124):
    if temp[i] > 1300:
        fixed_temp = np.append(fixed_temp, temp[i])
        fixed_mgorl = np.append(fixed_mgorl, m_gorl[i])
        #print(fixed_temp)
        #print(fixed_mgorl)
```

In [54]:

```
max(fixed_temp)
```

Out[54]:

```
46000.0
```

In [55]:

```
f = interp1d(fixed_mgorl, fixed_temp)
```

In [56]:

```
'''
plt.figure(figsize=(20,8))
plt.scatter(fixed_temp, fixed_mgorl)
plt.plot(np.linspace(1350, 4600, 1000), f(np.linspace(1350, 4600, 1000)))
plt.gca().invert_yaxis()
plt.xlabel('Effective Temperature, in K')
plt.ylabel('G Magnitude')
plt.title('Relation Between Teff and G Mag')
'''
```

Out[56]:

```
"\nplt.figure(figsize=(20,8))\nplt.scatter(fixed_temp, fixed_mgorl)\np
lt.plot(np.linspace(1350, 4600, 1000), f(np.linspace(1350, 4600, 100
0)))\nplt.gca().invert_yaxis()\nplt.xlabel('Effective Temperature, in
K')\nplt.ylabel('G Magnitude')\nplt.title('Relation Between Teff and G
Mag')\n"
```

In [57]:

```
np.sort(temp)
```

Out[57]:

<Column name='Teff' dtype='int64' length=124>

```
  250
  325
  350
  390
  420
  475
  530
  610
  700
  770
  840
  960
  ...
32000
32500
33000
34500
35000
36500
37300
39000
40000
41500
43000
46000
```

In [58]:

```
data = Table.read('asu.fit')
print(data)
```

```
  _r        _RAJ2000          _DEJ2000       Source   ...   Teff    Rad
Lum
arcmin        deg               deg                   ...    K     solRad
solLum
------ ----------------- ----------------- -------- ... ------- ------
------
0.0116   245.89664236128 -26.5255826857852  6631424 ...     0.0    0.0
0.0
0.0205 245.8971323101844 -26.5257618225833  6536960 ...     0.0    0.0
0.0
0.0265  245.896769646258 -26.5260997864899  6539264 ...     0.0    0.0
0.0
0.0301  245.896858746915  -26.525258143415  6631808 ...     0.0    0.0
0.0
0.0415 245.8973597661302 -26.5261756755971  6539392 ... 5095.63    0.0
0.0
0.0443 245.8971060573211 -26.5264169504993  6539648 ...     0.0    0.0
0.0
 0.046 245.8959742841654 -26.5255185055872  6631936 ... 5143.45    0.0
0.0
 0.046 245.8969773452325 -26.5264051264131  6538368 ...     0.0    0.0
0.0
0.0518 245.8975538856744 -26.5250953593613  6632192 ... 4508.75    0.0
0.0
0.0549 245.8961584044747 -26.5263737945874  6539008 ... 5143.45    0.0
0.0
   ...               ...               ...      ... ...     ...    ...
...
0.9945 245.8878103617531 -26.5401557475941 16593536 ...  5076.0    0.0
0.0
0.9945 245.8882688926921 -26.5403794827975  6361472 ...     0.0    0.0
0.0
0.9951 245.8827857078983 -26.5148357156183  7046656 ...     0.0    0.0
0.0
0.9954 245.9072999693412 -26.5393939638479  6175360 ...     0.0    0.0
0.0
0.9955  245.878491074076   -26.52827501348  7761792 ...  5000.0   0.99
0.557
 0.997 245.8840173611604 -26.5136539133512  7079424 ...     0.0    0.0
0.0
0.9982 245.8805703431035 -26.5337641229179 16642944 ... 5143.45    0.0
0.0
0.9983 245.9146830999414  -26.530264331309  6549376 ... 4650.03   1.62
1.108
0.9994 245.9126454428589  -26.516903343388  6606976 ...     0.0    0.0
0.0
0.9996 245.9154261835508 -26.5255252911882  6378880 ...  4920.0    0.0
0.0
Length = 1913 rows
```

In [59]:

```
np.unique(data[:0])
```

Out[59]:

```
array([],
      dtype=[('_r', '<f8'), ('_RAJ2000', '<f8'), ('_DEJ2000', '<f8'),
('Source', '<i4'), ('FG', '<f8'), ('e_FG', '<f8'), ('Gmag', '<f8'),
('e_Gmag', '<f8'), ('BPmag', '<f8'), ('e_BPmag', '<f8'), ('RPmag', '<f
8'), ('e_RPmag', '<f8'), ('BP-RP', '<f8'), ('Teff', '<f8'), ('Rad', '<
f8'), ('Lum', '<f8')])
```

In [60]:

```
bprp_arr1 = np.array(data['BP-RP'])
```

In [61]:

```
lum_arr = np.array(data['Lum'])
```

In [62]:

```
teff_arr = np.array(data['Teff'])
print(len(teff_arr))
```

```
1913
```

In [63]:

```
dat_arr = np.array(data['Gmag'])
```

In [64]:

```
bprp_array = np.array([])
gmag_array = np.array([])
lum_array = np.array([])
teff_array = np.array([])
for i in data['BP-RP']:
    if i != 0.0:
        bprp_array = np.append(bprp_array, i)

for k in np.arange(0, 1913):
    if bprp_arr1[k] != 0.0:
        gmag_array = np.append(gmag_array, dat_arr[k])
        lum_array = np.append(lum_array, lum_arr[k])
        teff_array = np.append(teff_array, teff_arr[k])

```

In [65]:

```
len(teff_array)
```

Out[65]:

```
841
```

In [66]:

```
finalteff_array = np.array([])
finalbprp_array = np.array([])
finalgmag_array = np.array([])
finallum_array = np.array([])
for k in np.arange(0, 841):
    if teff_array[k] != 0.0:
        finalgmag_array = np.append(finalgmag_array, gmag_array[k])
        finallum_array = np.append(finallum_array, lum_array[k])
        finalteff_array = np.append(finalteff_array, teff_array[k])
        finalbprp_array = np.append(finalbprp_array, bprp_array[k])
print(len(finalgmag_array))
```

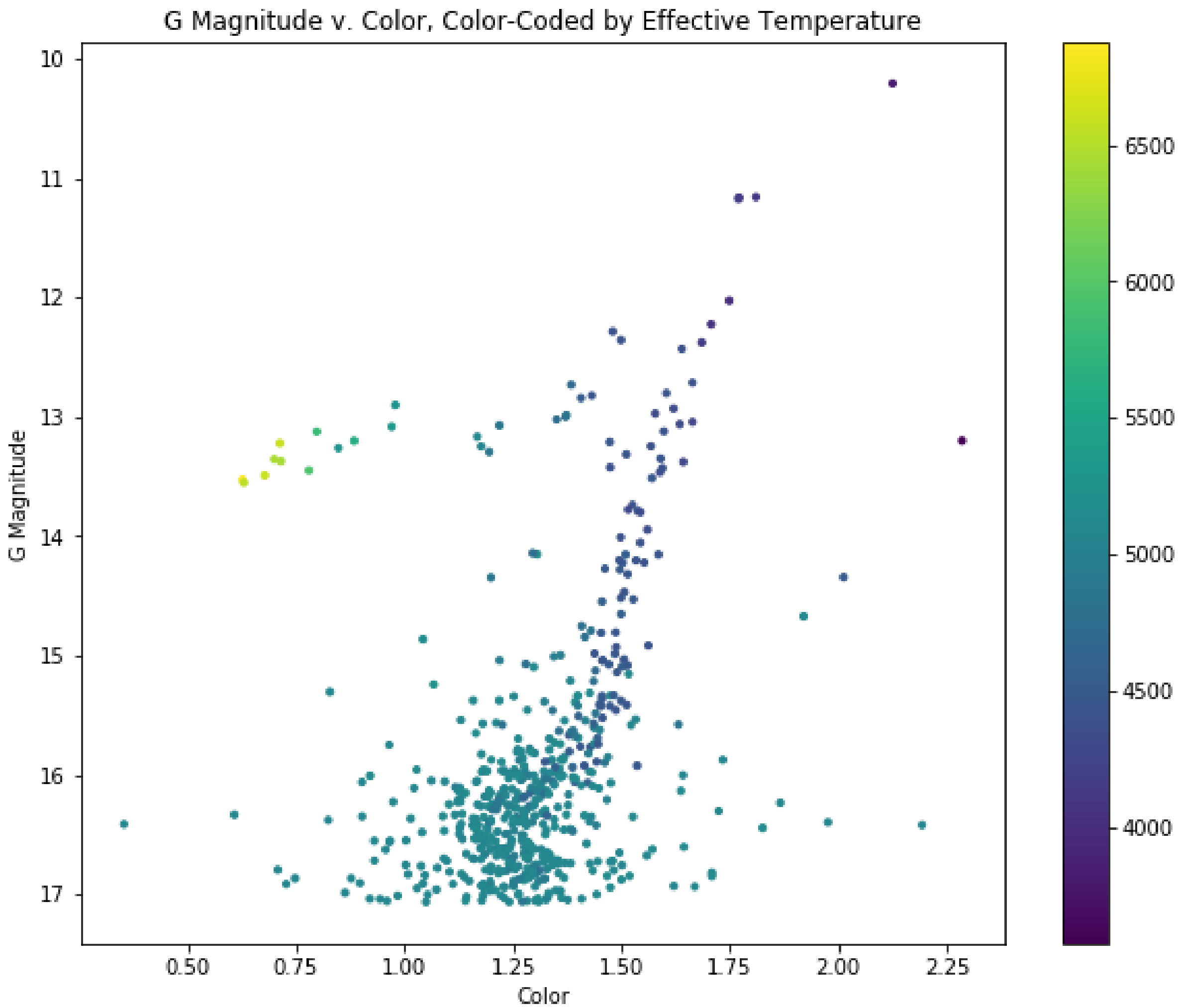
```
558
```

In [67]:

```
plt.figure(figsize=(10,8))
plt.scatter(finalbprp_array, finalgmag_array, c = finalteff_array, s = 10)
plt.gca().invert_yaxis()
plt.xlabel('Color')
plt.ylabel('G Magnitude')
plt.title('G Magnitude v. Color, Color-Coded by Effective Temperature')
plt.colorbar()
```

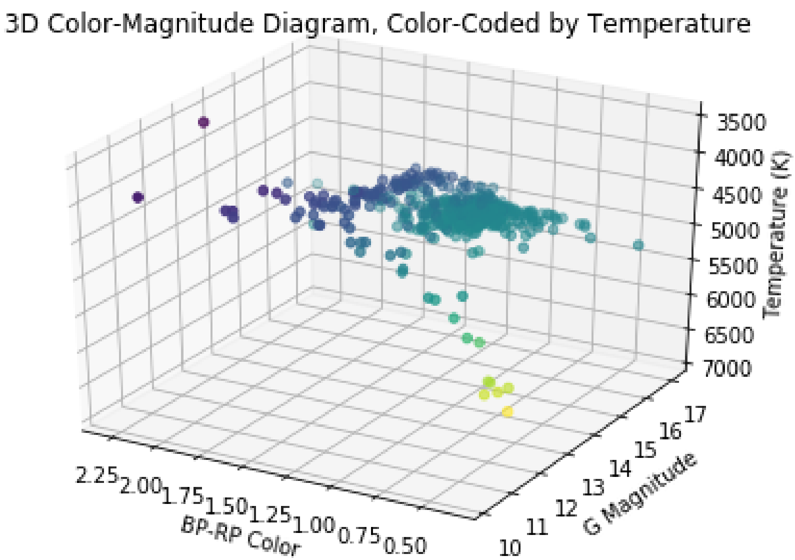
Out[67]:

```
<matplotlib.colorbar.Colorbar at 0x7f25fc7b1d60>
```



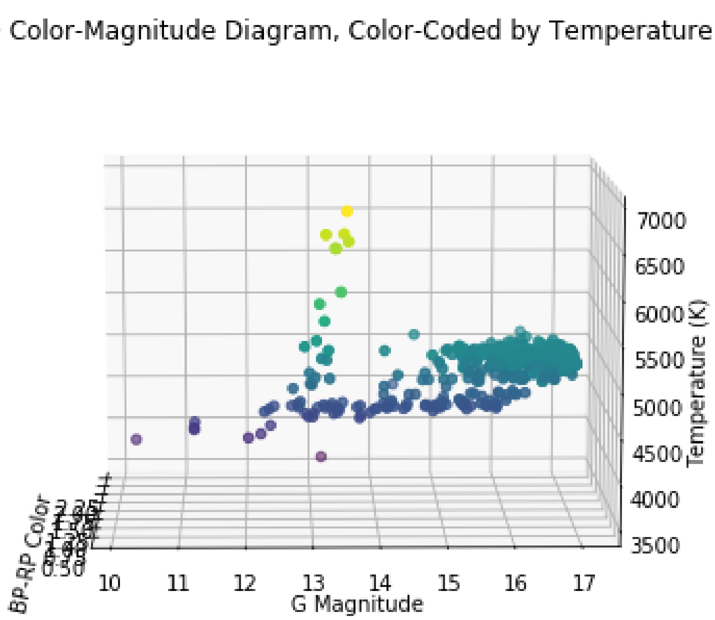
In [92]:

```
ax = plt.axes(projection ="3d")
ax.invert_yaxis()
ax.invert_zaxis()
ax.scatter3D(finalbprp_array, finalgmag_array, finalteff_array, c = finalteff_a
ax.set_xlabel('BP-RP Color')
ax.set_ylabel('G Magnitude')
ax.set_zlabel('Temperature (K)')
ax.set_title('3D Color-Magnitude Diagram, Color-Coded by Temperature')
plt.tight_layout()

```



In [95]:

```python

# Create a figure and a 3D Axes
fig = plt.figure()
ax = Axes3D(fig)

# Create an init function and the animate functions.
# Both are explained in the tutorial. Since we are changing
# the the elevation and azimuth and no objects are really
# changed on the plot we don't have to return anything from
# the init and animate function. (return value is explained
# in the tutorial.
def init():
    ax.invert_yaxis()
    ax.invert_zaxis()
    ax.scatter(finalbprp_array, finalgmag_array, finalteff_array, c = finalteff

    ax.set_xlabel('BP-RP Color')
    ax.set_ylabel('G Magnitude')
    ax.set_zlabel('Temperature (K)')
    ax.set_title('3D Color-Magnitude Diagram, Color-Coded by Temperature')
    return fig,

def animate(i):
    ax.view_init(elev=10., azim=i)
    return fig,

# Animate
anim = animation.FuncAnimation(fig, animate, init_func=init,
                               frames=360, interval=20, blit=True)
# Save
anim.save('finalproj_anim1.mp4', fps=30)
```

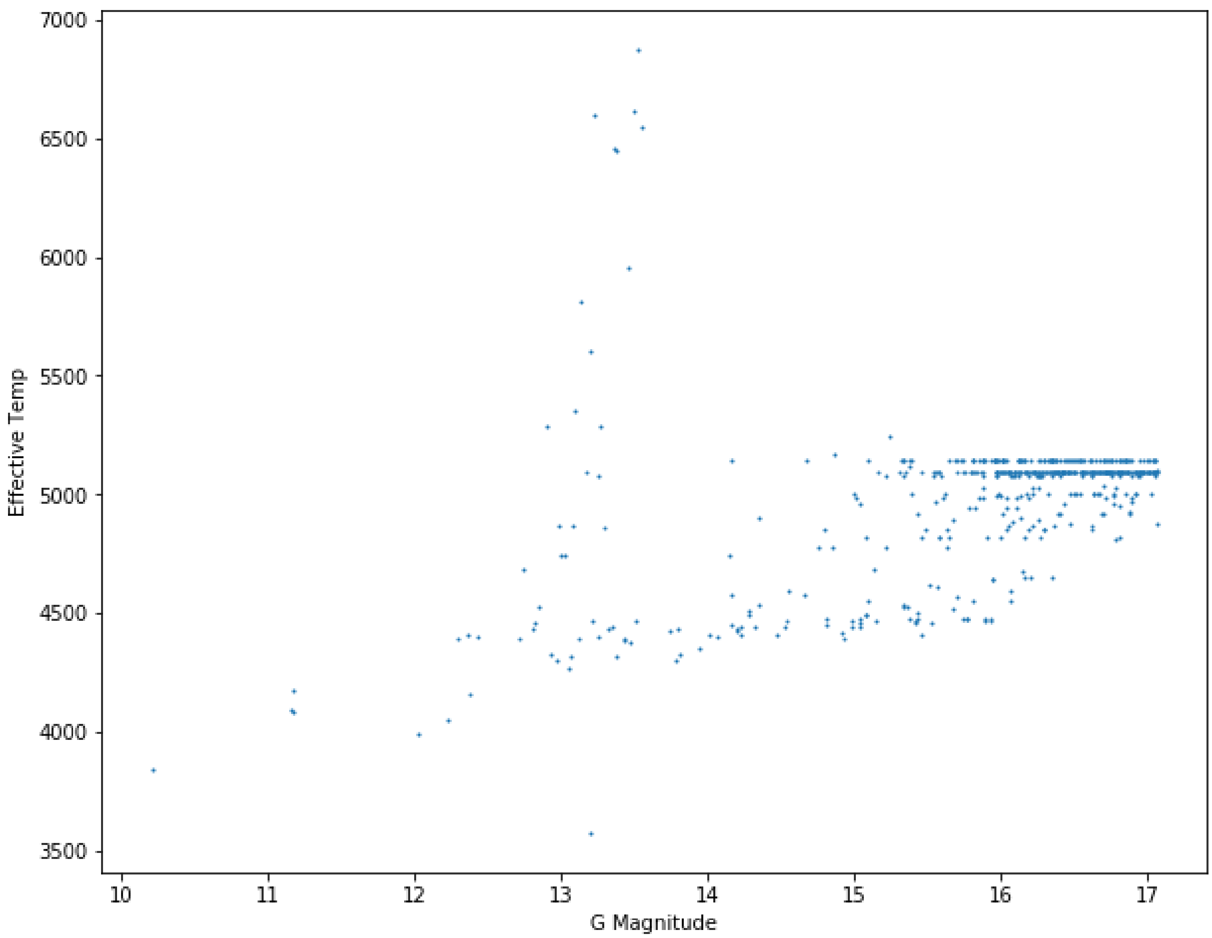


In [87]:

```
plt.figure(figsize=(10,8))
plt.scatter(finalgmag_array, finalteff_array, s=1)
plt.xlabel('G Magnitude')
plt.ylabel('Effective Temp')

```

Out[87]:

```
Text(0, 0.5, 'Effective Temp')
```



In [35]:

```
len(lum_array)
```

Out[35]:

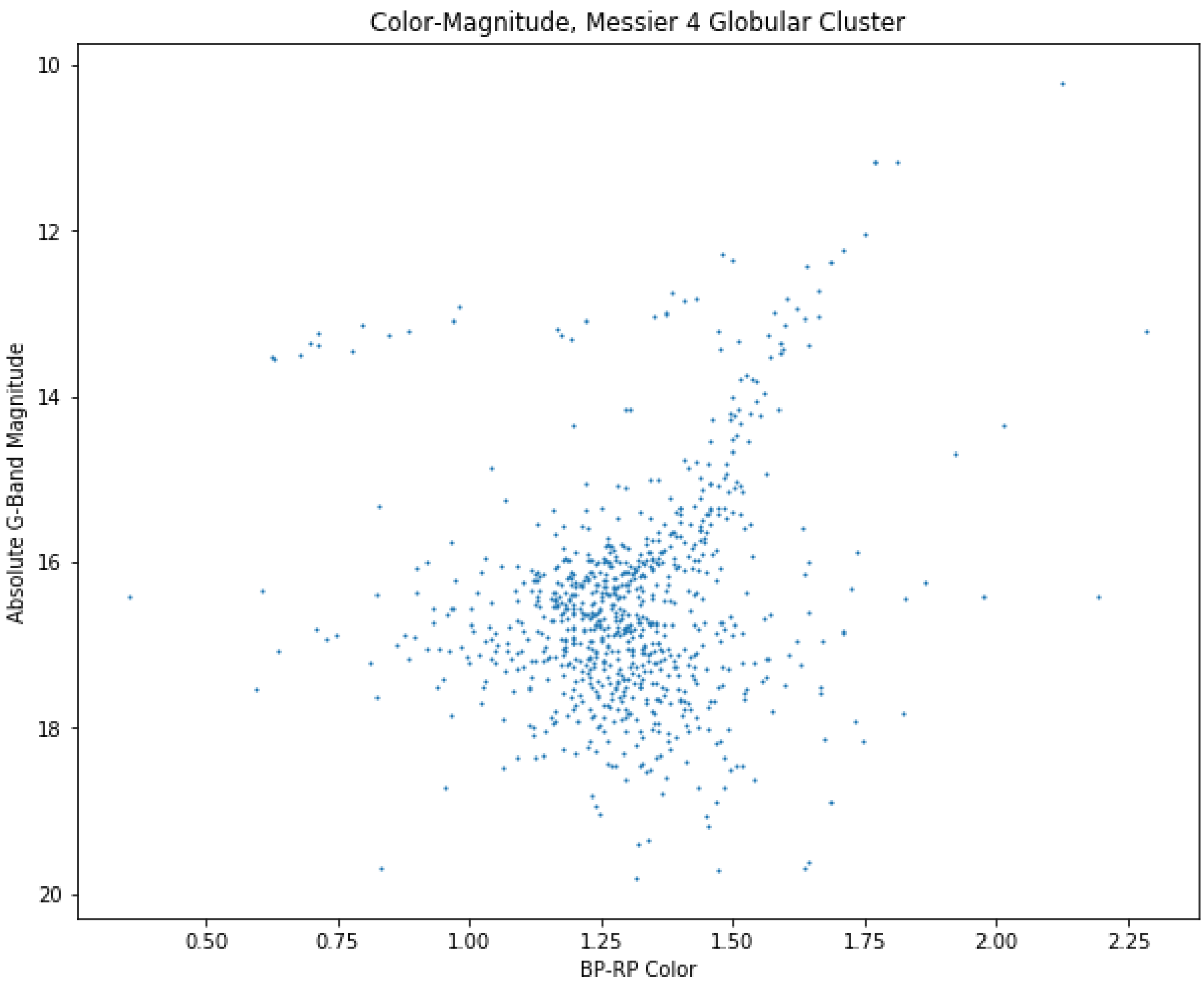
```
841
```

In [36]:

```
plt.figure(figsize=(10,8))
plt.gca().invert_yaxis()
plt.scatter(bprp_array, gmag_array, s=1)
plt.xlabel('BP-RP Color')
plt.ylabel('Absolute G-Band Magnitude')
plt.title('Color-Magnitude, Messier 4 Globular Cluster')

#try color coding by temp
```

Out[36]:

```
Text(0.5, 1.0, 'Color-Magnitude, Messier 4 Globular Cluster')
```

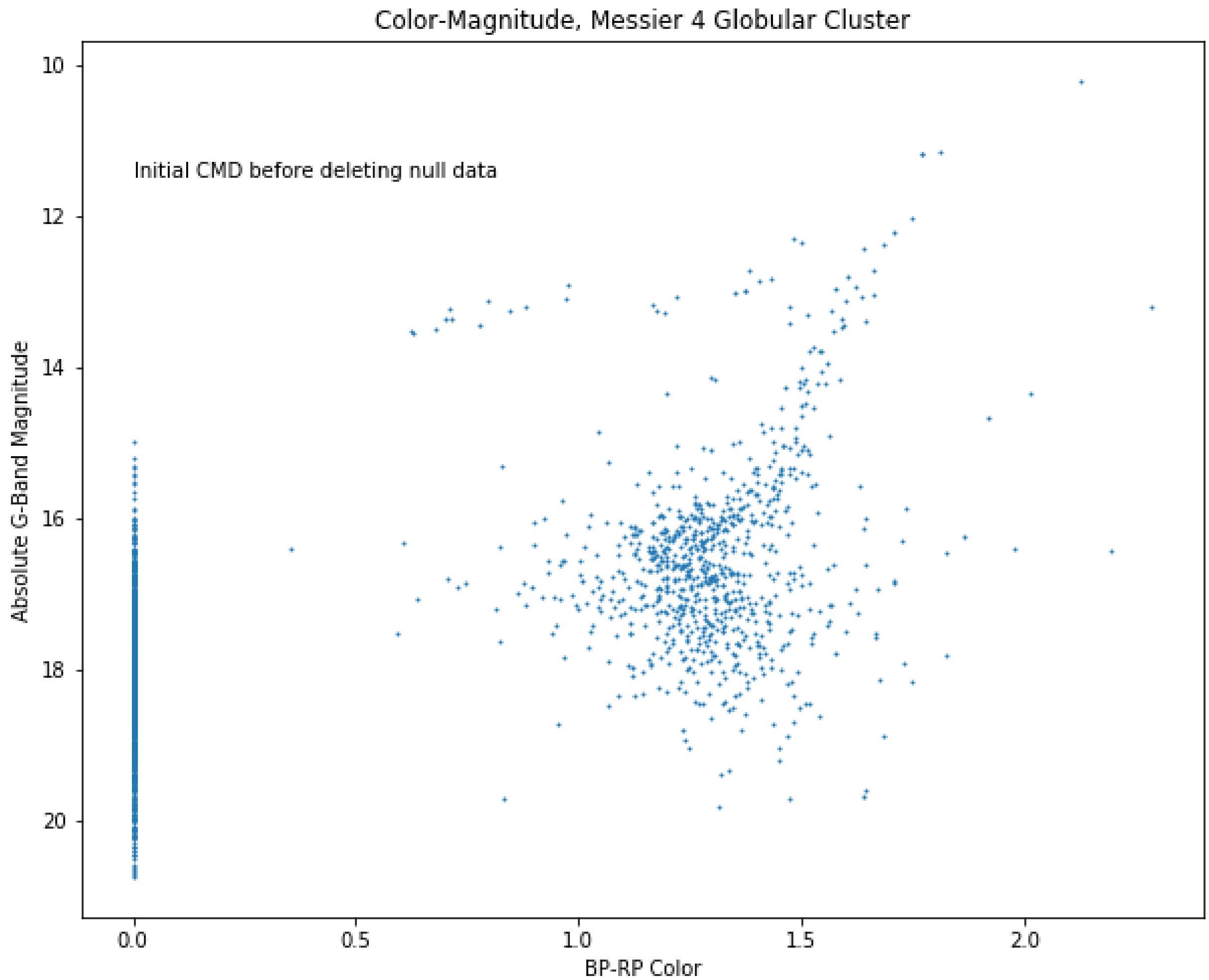


In [37]:

```
plt.figure(figsize=(10,8))
plt.gca().invert_yaxis()
plt.scatter(data['BP-RP'], data['Gmag'], s=1)
plt.xlabel('BP-RP Color')
plt.ylabel('Absolute G-Band Magnitude')
plt.title('Color-Magnitude, Messier 4 Globular Cluster')
plt.text(0, 11.5, 'Initial CMD before deleting null data')
```

Out[37]:

```
Text(0, 11.5, 'Initial CMD before deleting null data')
```



In [38]:

```
Teff_array = np.array(data['Teff'])
print(len(Teff_array))
```

```
1913
```

In [39]:

```
print(con.sigma_sb)
```

```
Name   = Stefan-Boltzmann constant
Value  = 5.670374419184431 4e-08
Uncertainty  = 0.0
Unit  = W / (K4 m2)
Reference = CODATA 2018
```

In [40]:

```
sigmasb = 5.670374419184431 4 * 10**(-8)
```

In [41]:

```
len(teff_array)
```

Out[41]:

```
841
```

In [42]:

```
teff = teff_array * u.K
oldlum = lum_array * con.L_sun
oldlum.to(u.W)
```

Out[42]:

$[0,\ 0,\ 0,\ \dots,\ 4.241424 \times 10^{26},\ 0,\ 0]\ \mathrm{W}$

In [43]:

```
lum = oldlum / u.W
print(lum)
for n in lum:
    if n != 0.00000000e+00:
        print(n)
```

```
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 1.36621320e+27 0.00000000e+00 0.00000000e+00
 3.85862400e+26 0.00000000e+00 1.24367892e+28 0.00000000e+00
 0.00000000e+00 0.00000000e+00 2.05257360e+27 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 4.15184880e+27 0.00000000e+00 4.34784240e+27 5.56744320e+27
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 3.35876376e+28 0.00000000e+00 0.00000000e+00 1.21730400e+26
 0.00000000e+00 0.00000000e+00 5.49700800e+26 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 5.25201600e+26
 0.00000000e+00 0.00000000e+00 1.90584636e+28 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
```

In [44]:

```
rad_array = np.array([])
for j in np.arange(0, 841):
    r = (lum_array[j] / (4 * np.pi * sigmasb * (teff_array[j]**4)))**0.5
    rad_array = np.append(rad_array, r)
#r = (lum_array / (4 * np.pi * con.sigma_sb * teff_array**4))**0.5
```

```
<ipython-input-44-650e08bd932c>:3: RuntimeWarning: invalid value encou
ntered in double_scalars
  r = (lum_array[j] / (4 * np.pi * sigmasb * (teff_array[j]**4)))**0.5
```

In [ ]:

```

```