Task 1 : The following code opens an astronomical data file and divides it into 5 arrays.

In [48]:

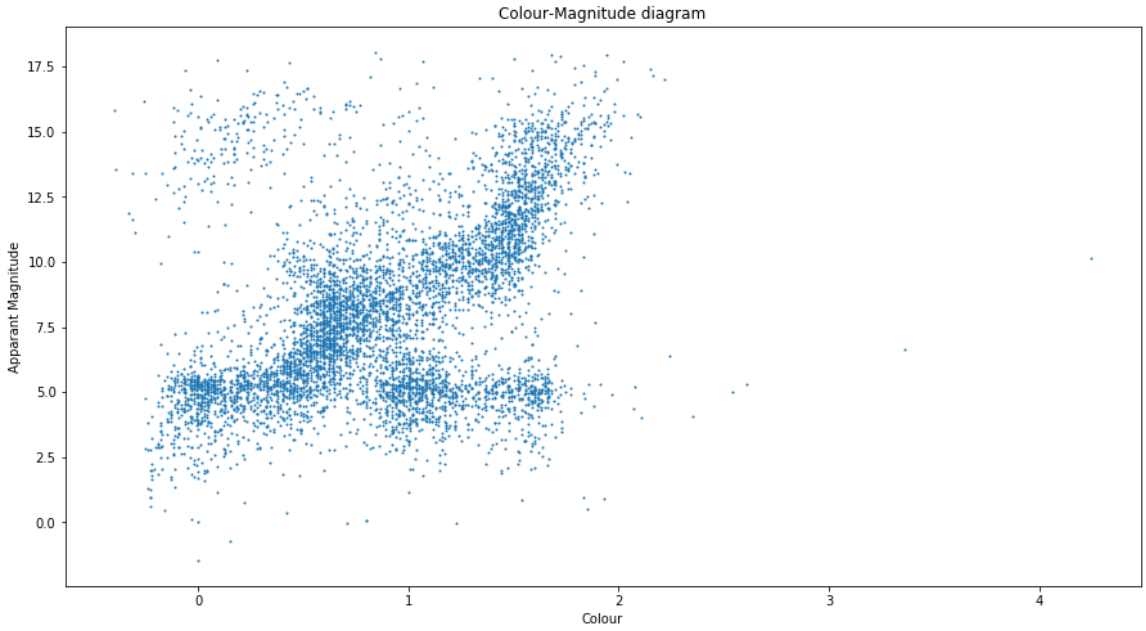
```
#Open the text file as StarData and split it into separate lines
with open('StarData.txt') as StarData:
    lines = StarData.read().splitlines()

StarID = [] #Star ID number
AppMag = [] #Apparant magnitude
Colour = [] #Observed B-V Colour
Parallax = [] #Observed Parallax
DeltaP = [] #Uncertainty in Parallax

#Split each line into individual numbers and assign them to the relevant array
for line in lines:
    Columns = line.split()
    StarID.append(eval(Columns[0]))
    AppMag.append(eval(Columns[1]))
    Colour.append(eval(Columns[2]))
    Parallax.append(eval(Columns[3]))
    #DeltaP.append(eval(Columns[4]))

#for loop to check that the data in the relevant arrays match the data in the StarData
 file
for i in range(0, len(AppMag), 500):
    print(StarID[i], "AppMag:",AppMag[i],"Colour: ", Colour[i])
```

```
2.0 AppMag: 4.61 Colour:  1.04
412.01 AppMag: 11.03 Colour:  1.54
886.01 AppMag: 15.28 Colour:  1.93
1326.01 AppMag: 14.14 Colour:  1.46
1827.0 AppMag: 11.2 Colour:  1.59
2301.0 AppMag: 6.24 Colour:  1.25
2764.0 AppMag: 8.94 Colour:  0.8
3198.1 AppMag: 11.25 Colour:  1.45
3626.0 AppMag: 5.41 Colour:  0.6
4071.0 AppMag: 10.65 Colour:  1.22
4588.0 AppMag: 8.83 Colour:  0.67
5106.1 AppMag: 9.56 Colour:  0.64
5608.0 AppMag: 3.9 Colour:  1.02
```

Task 2: A scatter plot is made, showing apparent magnitude vs colour

In [47]:

```
import matplotlib.pyplot as plt

#Plot Apparant Magnitude of the Star vs the B-V Colour
plt.figure(figsize = (15, 8))
plt.scatter(Colour, AppMag, s = 1)
plt.xlabel("Colour")
plt.ylabel("Apparant Magnitude")
plt.title("Colour-Magnitude diagram")
plt.show()
```



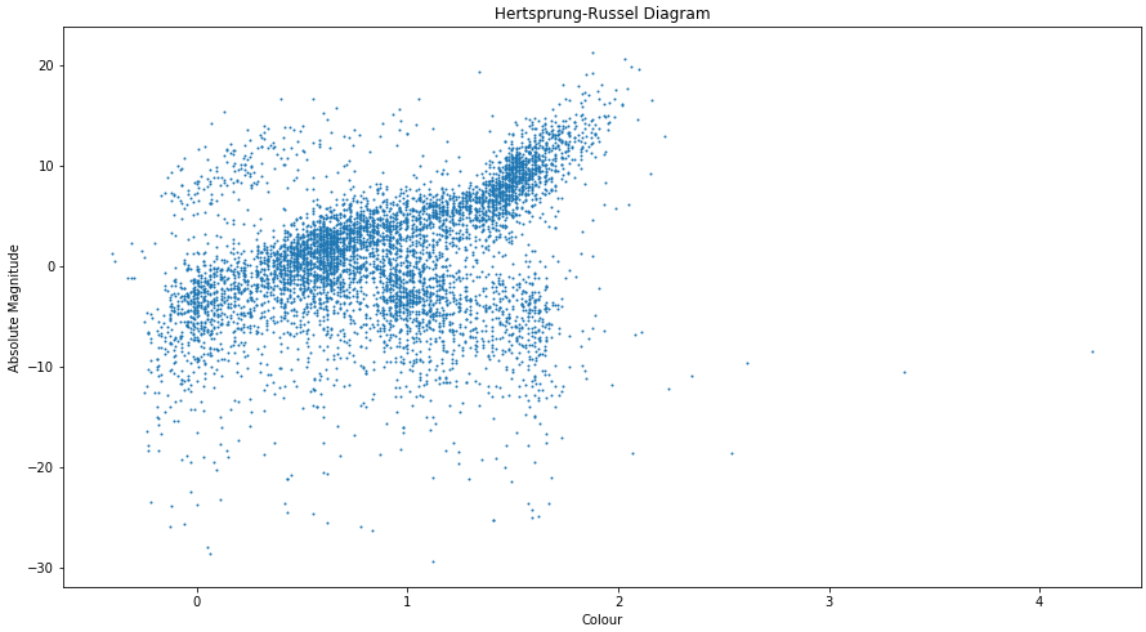
Task 3: Manipulate the data to plot the Hertzsprung-Russell diagram

In [37]:

```
import numpy as np

dist = [] #Array for distances
AbsMag = [] #Array for absolute magnitudes of stars
LenParallax = len(Parallax)

#Every distance value is equal to the inverse of the parallax
for i in range (LenParallax):
    dist.append(1/Parallax[i])

#compute the absolute magnitude values
for i in range (LenParallax):
    AbsMag.append(AppMag[i] - 5 * np.log(dist[i]/10))

#PLot Hertz-Russell diagram
plt.figure(figsize = (15, 8))
plt.title("Hertsprung-Russel Diagram")
plt.xlabel("Colour")
plt.ylabel("Absolute Magnitude")
plt.scatter(Colour, AbsMag, s = 1)
```

Out[37]:

```
<matplotlib.collections.PathCollection at 0x299ab74ec50>
```



The above plot does not resemble the Hert-Russull diagram, so I will create a function that would count the numbers of the various star types. I will test the function using two hypothetical star types.

In [44]:

```
#function to calculate the number of a certain star category. Input expected colour and
magnitude ranges that you expect the star type to occupy
def category(colour1, colour2, magnitude1, magnitude2):
    NumStar = 0 #Number of stars of a certain type

    #If a value in the colour array and absolute magnitude array are within the expecte
d parameter, add 1 to NumStar
    for i in range(LenParallax):
        if(colour1 < Colour[i] < colour2  and magnitude1 < AbsMag[i] < magnitude2 ):
            NumStar += 1
    return NumStar

#Test the function using 2 hypothetical star types
print("Star A: ", category(0, 1, 0, 10)) #Star found between 0 and 1 Colour value and 0
to 10 AbsMag values
print("Star B: ", category(0, 1, 10, 20))
```

```
Star A:  1600
Star B:  87
```