In [68]:
```
#Importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [28]:
```
#importing the data into the environment
ExptData = pd.read_excel("VES1.xlsx")
ExptData
```

Out[28]:

	S.no	current_electrode_distace(l/2)	potential_electrode_distance(b/2)	direct_voltmeter_reading(v1)	reverse_voltmeter_reading(v2)	direct_currentmeter_reading(i1)	reverse_currentmeter_reading(i2)
0	1	2	0.5	211	213	43.5	44.0
1	2	3	0.5	38	40	17.6	18.1
2	3	4	0.5	25	26	21.2	21.6
3	4	5	0.5	9	9	12.6	12.9
4	5	6	1.0	30	28	30.1	29.9
5	6	8	1.0	16	15	30.3	30.4
6	7	10	1.0	7	7	15.3	23.6
7	8	15	2.0	11	13	30.8	37.5
8	9	20	2.0	5	4	26.9	22.0
9	10	30	4.0	8	8	38.3	36.5
10	11	40	4.0	10	8	79.7	72.6
11	12	50	4.0	3	3	26.6	33.9

In [13]:
```
#Checking for missing values or null values from the data imported
pd.isnull(ExptData)
```

Out[13]:

	S.no	current_electrode_distace(l/2)	potential_electrode_distance(b/2)	direct_voltmeter_reading(v1)	reverse_voltmeter_reading(v2)	direct_currentmeter_reading(i1)	reverse_currentmeter_reading(i2)
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False

	S.no	current_electrode_distace(l/2)	potential_electrode_distance(b/2)	direct_voltmeter_reading(v1)	reverse_voltmeter_reading(v2)	direct_currentmeter_reading(i1)	reverse_currentmeter_reading(i2)
10	False	False	False	False	False	False	False
11	False	False	False	False	False	False	False

In [21]:
```
#Checking the type and dimensionalities
print(type(ExptData))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [24]:
```
print(len(ExptData.columns))
print(ExptData.shape)
```

```
7
(12, 7)
```

In [30]:
```
ExptData.describe() #Understanding the summary of the data
```

Out[30]:

	S.no	current_electrode_distace(l/2)	potential_electrode_distance(b/2)	direct_voltmeter_reading(v1)	reverse_voltmeter_reading(v2)	direct_currentmeter_reading(i1)	reverse_currentmeter_reading(i2)
count	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000
mean	6.500000	16.083333	1.750000	31.083333	31.166667	31.075000	31.916667
std	3.605551	15.882857	1.453835	57.683239	58.348223	17.784882	15.638230
min	1.000000	2.000000	0.500000	3.000000	3.000000	12.600000	12.900000
25%	3.750000	4.750000	0.500000	7.750000	7.750000	20.300000	21.900000
50%	6.500000	9.000000	1.000000	10.500000	11.000000	28.500000	30.150000
75%	9.250000	22.500000	2.500000	26.250000	26.500000	32.675000	36.750000
max	12.000000	50.000000	4.000000	211.000000	213.000000	79.700000	72.600000

In [33]:
```
#Renaming or assigning row names for the sake of mitigating ambiguity
ExptData = ExptData.rename(columns = {"current_electrode_distace(l/2)" : "current_ed", "potential_electrode_distance(b/2)" : "potential_ed",
                                      "direct_voltmeter_reading(v1)" : "vr_direct", "reverse_voltmeter_reading(v2)" : "vr_reverse",
                                      "direct_currentmeter_reading(i1)" : "cr_direct", "reverse_currentmeter_reading(i2)" : "cr_reverse"})
```

In [36]:
```
#Checking the first and last 5 rows of the DataFrame
ExptData.head() #first 5
```

Out[36]:

	S.no	current_ed	potential_ed	vr_direct	vr_reverse	cr_direct	cr_reverse
0	1	2	0.5	211	213	43.5	44.0
1	2	3	0.5	38	40	17.6	18.1
2	3	4	0.5	25	26	21.2	21.6
3	4	5	0.5	9	9	12.6	12.9
4	5	6	1.0	30	28	30.1	29.9

In [37]:
```
ExptData.tail() #bottom 5
```

Out[37]:

	S.no	current_ed	potential_ed	vr_direct	vr_reverse	cr_direct	cr_reverse
7	8	15	2.0	11	13	30.8	37.5
8	9	20	2.0	5	4	26.9	22.0
9	10	30	4.0	8	8	38.3	36.5
10	11	40	4.0	10	8	79.7	72.6
11	12	50	4.0	3	3	26.6	33.9

In [49]:
```
#Performing calculations required on the columns
#Mutating and adding additional columns to the existing dataframe

ExptData["Avg_pd"] = (ExptData["vr_direct"] + ExptData["vr_reverse"])/2
ExptData["Avg_curr"] = (ExptData["cr_direct"] + ExptData["cr_reverse"])/2
ExptData["R"] = ExptData["Avg_pd"]/ ExptData["Avg_curr"]
ExptData["K"] = ((np.pi * ((ExptData["current_ed"] ** 2)) - (ExptData["potential_ed"] ** 2))/ (ExptData["potential_ed"] * 2))
ExptData["Apparent_res"] = ExptData["K"] * ExptData["R"]
```

In [53]:
```
#Rounding the values upto 3 decimal places
ExptData = round(ExptData, 3)
```

In [55]:
```
ExptData #Viewing the table
```

Out[55]:

	S.no	current_ed	potential_ed	vr_direct	vr_reverse	cr_direct	cr_reverse	Avg_pd	Avg_curr	R	K	Apparent_res
0	1	2	0.5	211	213	43.5	44.0	212.0	43.75	4.846	12.316	59.682
1	2	3	0.5	38	40	17.6	18.1	39.0	17.85	2.185	28.024	61.230
2	3	4	0.5	25	26	21.2	21.6	25.5	21.40	1.192	50.015	59.598
3	4	5	0.5	9	9	12.6	12.9	9.0	12.75	0.706	78.290	55.263
4	5	6	1.0	30	28	30.1	29.9	29.0	30.00	0.967	56.049	54.180
5	6	8	1.0	16	15	30.3	30.4	15.5	30.35	0.511	100.031	51.087
6	7	10	1.0	7	7	15.3	23.6	7.0	19.45	0.360	156.580	56.353
7	8	15	2.0	11	13	30.8	37.5	12.0	34.15	0.351	175.715	61.745
8	9	20	2.0	5	4	26.9	22.0	4.5	24.45	0.184	313.159	57.637
9	10	30	4.0	8	8	38.3	36.5	8.0	37.40	0.214	351.429	75.172
10	11	40	4.0	10	8	79.7	72.6	9.0	76.15	0.118	626.319	74.023
11	12	50	4.0	3	3	26.6	33.9	3.0	30.25	0.099	979.748	97.165

In [59]:
```
#Maximum and minimum values of Apparent Resistivity
print(ExptData["Apparent_res"].max()) #max value
print(ExptData["Apparent_res"].min()) #min value
```

```
97.165
51.087
```

In [66]:
```
#Depth of the water table
min_value = ExptData["Apparent_res"].min()
depth = ExptData.loc[ExptData["Apparent_res"] == min_value, "current_ed"].iloc[0]
print("Depth of the water table is {} mts.".format(depth))
```

```
Depth of the water table is 8 mts.
```

In [85]:
```
#Plotting the graph between "Distance" and "Apparent resistivity"
plt.plot(ExptData["current_ed"], ExptData["Apparent_res"])

#Adding labels and plot title
plt.xlabel("Distance")
plt.ylabel("Apparent Resistivity")
plt.title("L/2 vs. Apparent Resistivity")

#Setting log-log scale
plt.xscale("log")
plt.yscale("log")

#Setting the limits for the plot
plt.xlim(1,1000)
plt.ylim(1,1000)

#Pointing the end point or minimum point on the graph
plt.axvline(x = depth, color = 'g')
plt.axhline(y = min(ExptData["Apparent_res"]), color = 'r')

#Displaying output graph
plt.show()
```