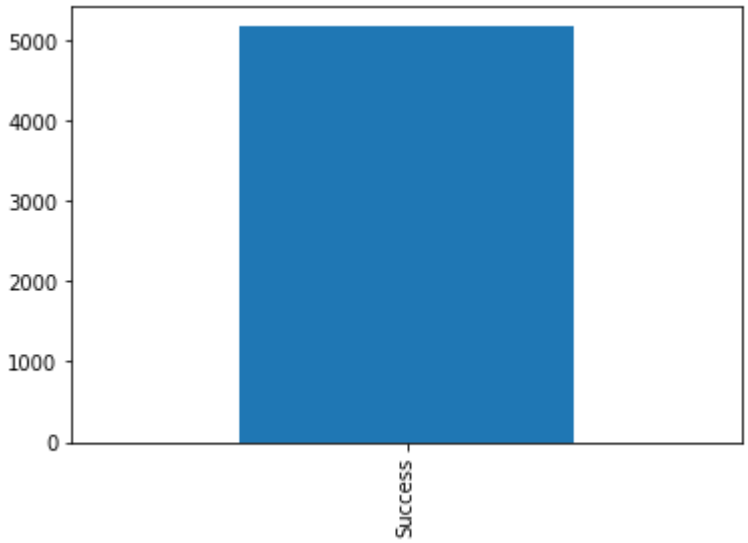
In [18]:
```
import pandas as pd
df = pd.read_csv("C:/Users/jain/Desktop/blockchain_interoperability/Finality Time Analysis/Mainnet.csv")
df.head()
```

Out[18]:

	Unnamed: 0	Transaction Hash	Status	Time Consumed (Seconds)	Value (Ether)	Transaction Fees (Ether)	Gas Price (Ether)
0	0	0x93666f74bcd9ca5df77db3c2f4a48407004251cb316f...	Success	30	48.796871	0.003129	1.490000e-07
1	1	0x1598196dfb6b1323e2ac5a9f4dd9efc4867e6b7e50a9...	Success	30	0.000000	0.003907	1.490000e-07
2	2	0x5941f5c381e75eb8421ea1d49cb36ca6ada6ff53074d...	Success	30	0.000000	0.006140	1.490000e-07
3	3	0xec991377dffb16f70cb943ef976a2a31476e86cc21f3...	Success	30	0.000000	0.006355	1.500000e-07
4	4	0x0663c86faf9888e61e5b751a29534734bfd13ed9908b...	Success	30	0.000000	0.028795	1.500000e-07

In [19]:
```
df['Status'].value_counts().plot.bar()
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15447d2bcf8>
```



In [20]:
```
import pandas as pd
df.drop(df.index[(df["Time Consumed (Seconds)"] == "Remove")],axis=0,inplace=True)
#pd.to_numeric(df['Time Consumed (Seconds)'])
#df['Time Consumed (Seconds)'].value_counts()
df['Time Consumed (Seconds)'] = df['Time Consumed (Seconds)'].astype(int)
```

In [21]:
```
df.describe()
```

Out[21]:

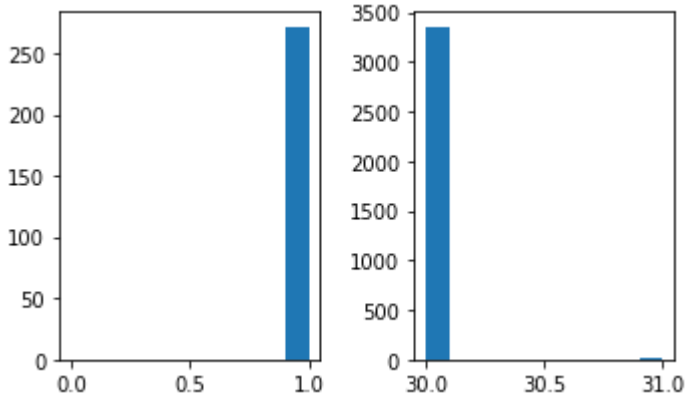
	Unnamed: 0	Time Consumed (Seconds)	Value (Ether)	Transaction Fees (Ether)	Gas Price (Ether)	Gas Limit	Gas Used
count	5169.000000	5169.000000	5169.000000	5169.000000	5.169000e+03	5.169000e+03	5.169000e+03
mean	2584.763591	49.946605	1.809618	0.011200	1.733460e-07	1.571153e+05	6.662727e+04
std	1492.618829	85.703142	26.361856	0.017238	1.273859e-07	3.530212e+05	1.042495e+05
min	0.000000	1.000000	0.000000	0.000021	1.000000e-09	2.100000e+04	1.496000e+04
25%	1293.000000	30.000000	0.000000	0.003780	1.480000e-07	3.473100e+04	2.100000e+04
50%	2585.000000	30.000000	0.000000	0.006696	1.640000e-07	8.100000e+04	4.120900e+04
75%	3877.000000	30.000000	0.060000	0.011938	1.790000e-07	2.000000e+05	6.709400e+04
max	5169.000000	2632.000000	1250.000000	0.609017	4.872186e-06	1.237656e+07	4.258861e+06

In [22]:
```
from matplotlib import pyplot as plt
df['Time Consumed (Seconds)'].value_counts().sort_values()
```

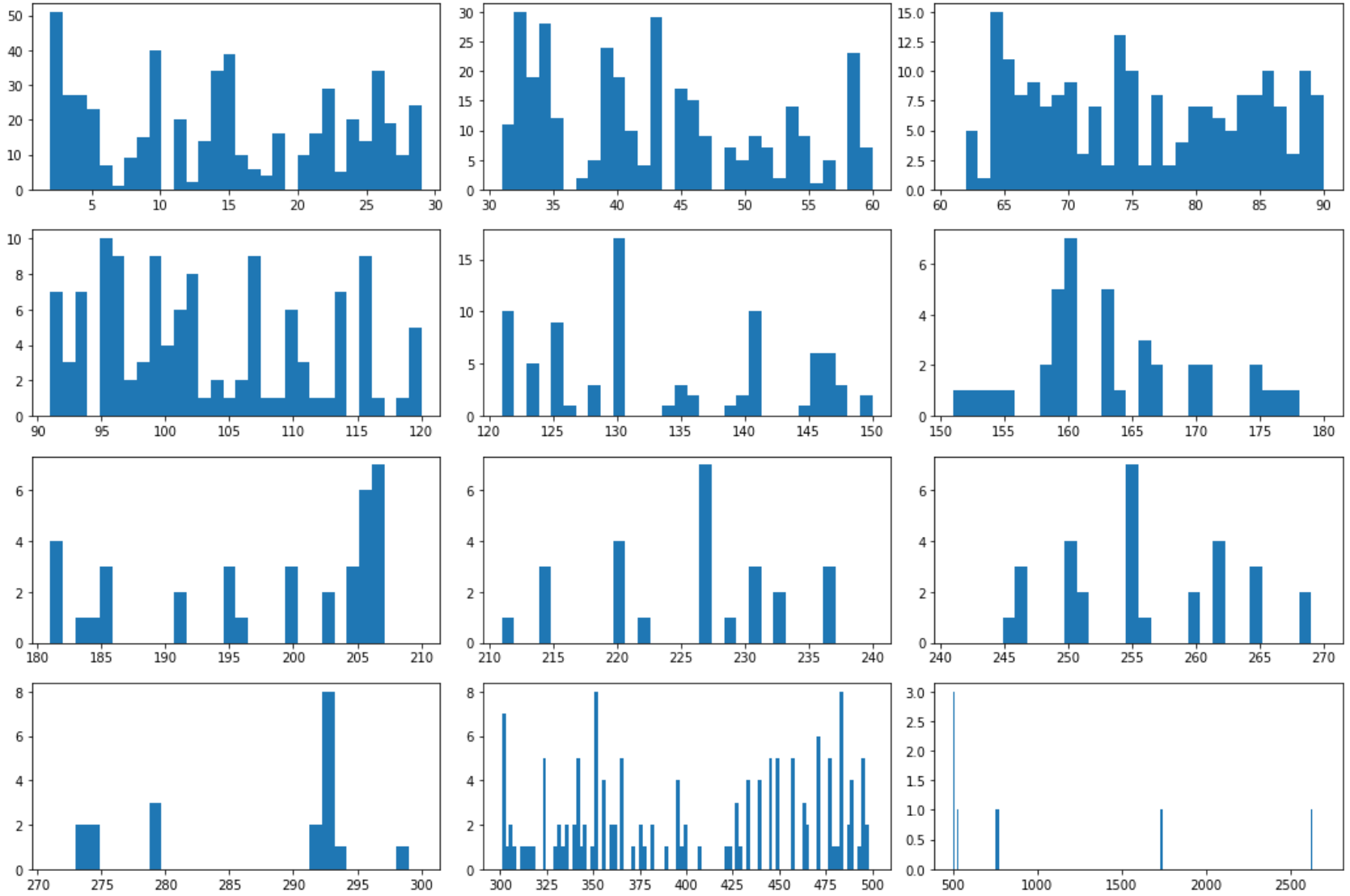
```
Out[22]: 1739      1
         7         1
         528       1
         494       1
         458       1
         390       1
         362       1
         350       1
         334       1
         330       1
         318       1
         314       1
         294       1
         222       1
         482       1
         480       1
         303       1
         344       1
         372       1
         312       1
         196       1
         360       1
         308       1
         112       1
         152       1
         108       1
         56        1
         184       1
         256       1
         176       1
                ...
         13       14
         9        15
         64       15
         46       15
         21       16
         19       16
         130      17
         45       17
         33       19
         40       19
         27       19
         24       20
```

```
11        20
5         23
59        23
39        24
29        24
4         27
3         27
34        28
22        29
43        29
32        30
26        34
14        34
15        39
10        40
2         51
1        271
30      3347
Name: Time Consumed (Seconds), Length: 250, dtype: int64
```

In [23]:
```
a = plt.subplots(1,2,figsize=(5,3))[1].ravel()
a[0].hist(df['Time Consumed (Seconds)'],range=[0,1])
a[1].hist(df['Time Consumed (Seconds)'],range=[30,31])
plt.tight_layout()
plt.show()
```



In [24]:
```
a = plt.subplots(4,3,figsize=(15,10))[1].ravel()
a[0].hist(df['Time Consumed (Seconds)'],bins=30,range=[2,29])
a[1].hist(df['Time Consumed (Seconds)'],bins=30,range=[31,60])
a[2].hist(df['Time Consumed (Seconds)'],bins=30,range=[61,90])
a[3].hist(df['Time Consumed (Seconds)'],bins=30,range=[91,120])
a[4].hist(df['Time Consumed (Seconds)'],bins=30,range=[121,150])
a[5].hist(df['Time Consumed (Seconds)'],bins=30,range=[151,180])
a[6].hist(df['Time Consumed (Seconds)'],bins=30,range=[181,210])
a[7].hist(df['Time Consumed (Seconds)'],bins=30,range=[211,240])
a[8].hist(df['Time Consumed (Seconds)'],bins=30,range=[241,270])
a[9].hist(df['Time Consumed (Seconds)'],bins=30,range=[271,300])
a[10].hist(df['Time Consumed (Seconds)'],bins=100,range=[301,500])
a[11].hist(df['Time Consumed (Seconds)'],bins=200,range=[501,2700])
#plt.tight_layout()
#plt.show()
#a[7].set_ylim([0,5])
plt.tight_layout()
plt.show()
```



In [25]:
```
v = df[df['Time Consumed (Seconds)'] > 150].count()/df.count()
v['Time Consumed (Seconds)']*100
```

Out[25]: 5.765138324627587

In [26]:
```
v = df[df['Time Consumed (Seconds)'] > 500 ].count()/df.count()
v['Time Consumed (Seconds)']*100
```

Out[26]:
```
0.15476881408396206
```

In [27]:
```
v= df[df['Time Consumed (Seconds)'].between(150,500, inclusive=True)].count()
print(v['Time Consumed (Seconds)'])
v = (df[df['Time Consumed (Seconds)'].between(150,500, inclusive=True)].count()/df.count())*100
print(v['Time Consumed (Seconds)'])
v = (df[df['Time Consumed (Seconds)'].between(0,150, inclusive=True)].count()/df.count())*100
print(v['Time Consumed (Seconds)'])
v = (df[df['Time Consumed (Seconds)'].between(501,2700, inclusive=True)].count()/df.count())*100
print(v['Time Consumed (Seconds)'])
```

```
292
5.649061714064616
94.23486167537241
0.15476881408396206
```

In [28]:
```
v = df[df['Time Consumed (Seconds)'] <= 30].count()/df.count()
v['Time Consumed (Seconds)']*100
```

Out[28]:
```
80.17024569549235
```

In [29]:
```
df[df['Time Consumed (Seconds)'].between(0,150, inclusive=True)].describe()
```

Out[29]:

	Unnamed: 0	Time Consumed (Seconds)	Value (Ether)	Transaction Fees (Ether)	Gas Price (Ether)	Gas Limit	Gas Used
count	4871.000000	4871.000000	4871.000000	4871.000000	4.871000e+03	4.871000e+03	4.871000e+03
mean	2601.823445	33.018887	1.858968	0.011183	1.750269e-07	1.591509e+05	6.584072e+04
std	1498.593868	22.208173	27.071288	0.016953	1.309868e-07	3.600422e+05	1.001020e+05
min	0.000000	1.000000	0.000000	0.000210	1.000000e-08	2.100000e+04	1.496000e+04
25%	1310.500000	30.000000	0.000000	0.003822	1.490000e-07	3.415250e+04	2.100000e+04
50%	2574.000000	30.000000	0.000000	0.006747	1.660000e-07	8.241800e+04	4.120900e+04
75%	3946.500000	30.000000	0.060209	0.011916	1.801800e-07	2.012555e+05	6.510700e+04
max	5169.000000	150.000000	1250.000000	0.609017	4.872186e-06	1.237656e+07	4.258861e+06

For the transaction having time upto 150 seconds, the mean is 33, standard deviation is 22 and amximum time taken is 150 seconds. Though, 80% of transactions completed upto 30 seconds. So, finality time for the bracket [0-150] seconds is as calculated ((Mean + Std. Deviation + Maximum value -150 seconds(As considerable amount of transactions))/2) = 103.

In [30]:
```
df[df['Time Consumed (Seconds)'].between(151,500, inclusive=True)].describe()
```

Out[30]:

	Unnamed: 0	Time Consumed (Seconds)	Value (Ether)	Transaction Fees (Ether)	Gas Price (Ether)	Gas Limit	Gas Used
count	290.000000	290.000000	290.000000	290.000000	2.900000e+02	2.900000e+02	2.900000e+02
mean	2300.751724	308.286207	1.026980	0.011293	1.457768e-07	1.223201e+05	7.834090e+04
std	1363.707670	108.856040	8.779158	0.021540	1.497629e-08	2.063668e+05	1.583340e+05
min	232.000000	151.000000	0.000000	0.000021	1.000000e-09	2.100000e+04	2.100000e+04
25%	441.250000	207.000000	0.000000	0.003396	1.345000e-07	3.660075e+04	2.128900e+04
50%	2683.500000	293.000000	0.000000	0.006185	1.370000e-07	6.000000e+04	4.120900e+04
75%	3500.750000	396.500000	0.036101	0.012955	1.600000e-07	1.522732e+05	9.342800e+04
max	4384.000000	497.000000	110.047884	0.311431	1.660000e-07	2.601641e+06	2.324111e+06

For the transaction having time upto 150 seconds, the mean is 33, standard deviation is 22 and amximum time taken is 150 seconds. Though, 80% of transactions completed upto 30 seconds. So, finality time for the bracket [0-150] seconds is as calculated ((Mean + Std. Deviation + Maximum value -150 seconds(As considerable amount of transactions))/2) = 457.

In [31]:
```
df[df['Time Consumed (Seconds)'].between(501,2700, inclusive=True)].describe()
```

Out[31]:

	Unnamed: 0	Time Consumed (Seconds)	Value (Ether)	Transaction Fees (Ether)	Gas Price (Ether)	Gas Limit	Gas Used
count	8.000000	8.000000	8.000000	8.000000	8.000000e+00	8.00000	8.000000
mean	2492.875000	992.000000	0.132651	0.018468	1.492500e-07	179005.37500	120924.500000
std	1441.603813	782.191244	0.188606	0.013958	1.258957e-08	102778.62817	88369.885874
min	279.000000	504.000000	0.000000	0.002835	1.342000e-07	21000.00000	21000.000000
25%	1963.000000	504.000000	0.000000	0.005686	1.352250e-07	130001.25000	36156.750000
50%	3036.500000	642.000000	0.008680	0.018567	1.542500e-07	174069.50000	120617.000000
75%	3529.500000	1011.500000	0.260387	0.027005	1.600000e-07	265766.25000	190224.750000
max	3727.000000	2632.000000	0.450000	0.037861	1.610000e-07	314240.00000	236634.000000

In [33]:
```
df[df['Time Consumed (Seconds)'].between(501,2700, inclusive=True)]
```

Out[33]:

	Unnamed: 0	Transaction Hash	Status	Time Consumed (Seconds)	Value (Ether)	Transaction Fees (Ether)	Gas Pr (Eth
279	279	0x7e0a8e3990159e4f3f56cb9c698cb8cdd8c1db009248...	Success	769	0.450000	0.018895	1.48500 07
280	280	0xae8ba6ae4274756b4de55bb669e4fe91f0c0766a3f5a...	Success	528	0.000000	0.002841	1.35300 07
2524	2524	0xc13b3162f1ac5f352ecb8c054935bafaa61cb21e909b...	Success	2632	0.223849	0.023792	1.34200 07
2545	2545	0x5edf1fdcb7742399065365ea808d5364c9ec92a8b15c...	Success	1739	0.001000	0.002835	1.35000 07
3528	3528	0x2101f4ba90a93a0b5803420c7458de0524986c81c1c1...	Success	504	0.370000	0.018239	1.60000 07
3529	3529	0x79f782190634b9aa74fe4759ecc30d55d859b69691be...	Success	504	0.016360	0.036645	1.60000 07
3531	3531	0xd83e502efad5eda0a030f7487271f79c53c34a6598c1...	Success	504	0.000000	0.037861	1.60000 07
3727	3727	0x9fbdb0eb4b20d21adf4d8bc60ba0fa973d5ddd3c8cec...	Success	756	0.000000	0.006635	1.61000 07

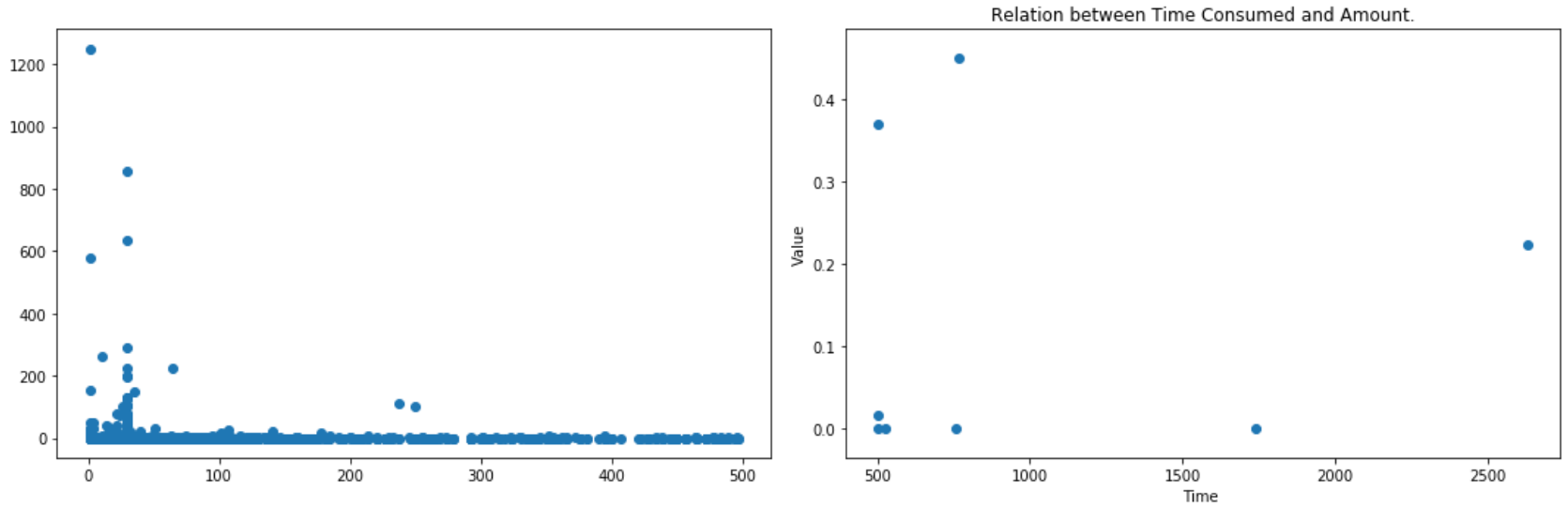
For the transaction having time upto 150 seconds, the mean is 33, standard deviation is 22 and amximum time taken is 150 seconds. Though, 80% of transactions completed upto 30 seconds. So, finality time for the bracket [0-150] seconds is as calculated ((Mean + Std. Deviation + Maximum value -150 seconds(As considerable amount of transactions))/2) = 2203.

Considering the above brackets of finality time, we are using ((0.94*103* + *0.05*457 + 0.0015*2203* = *127)* + *effective outliers value(0.94*150 + 0.05*497* + *0.00154*2632 = 169)/2) = 150 (approx) for ropsten network. For mainnet network, we took an average of outliers ranging between 500 and 2700 = (992 + 150)/2 = 571 seconds.

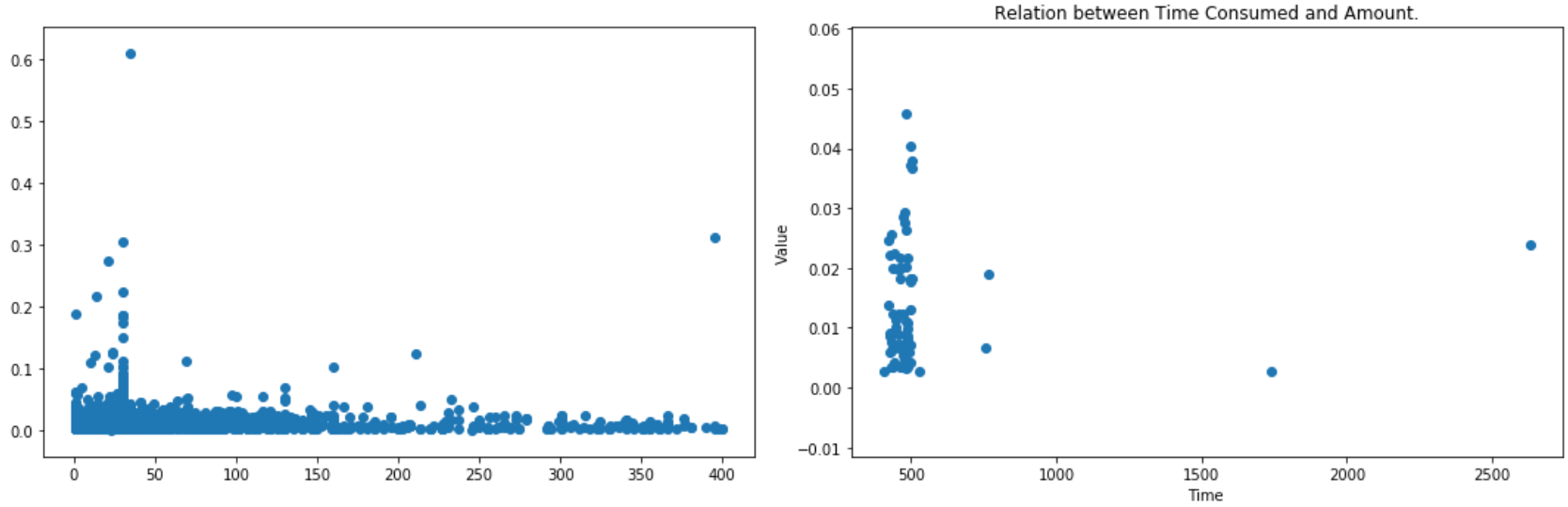
Analysis of relationship b/w Time Taken & different parameters.

1. Relationship of Time Consumed and amount value.
2. Relationship of Time Consumed and Transaction Fees.
3. Relationship of Time Consumed and Gas Price,Gas Limit, Gas Used.
4. Relationship of Gas Limit and Gas Used.
5. Relationship of value and Gas Price, Gas Limit.

In [60]:
```
a = plt.subplots(1,2,figsize=(15,5))[1].ravel()
v = df[df['Time Consumed (Seconds)'] <= 500]
v1 = df[df['Time Consumed (Seconds)'] > 500]
a[0].scatter(v['Time Consumed (Seconds)'], v['Value (Ether)'])
a[1].scatter(v1['Time Consumed (Seconds)'], v1['Value (Ether)'])
plt.xlabel('Time ')
plt.ylabel('Value')
plt.title("Relation between Time Consumed and Amount.")
plt.tight_layout()
plt.show()
```

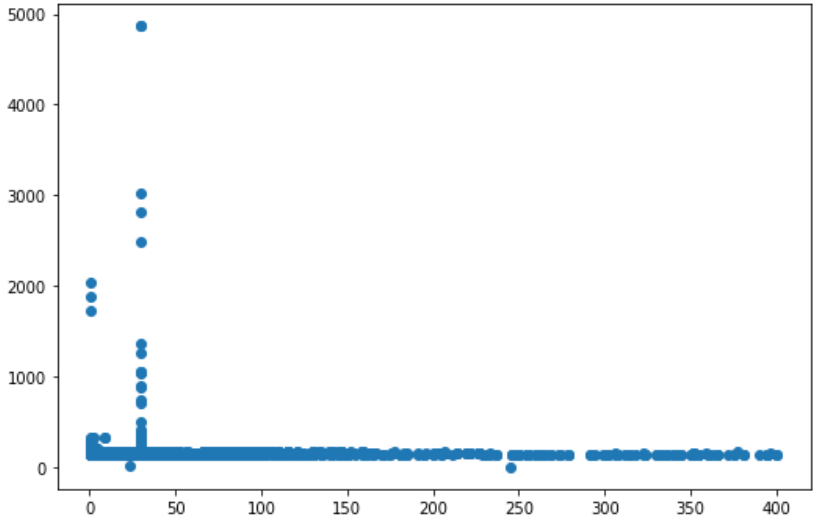


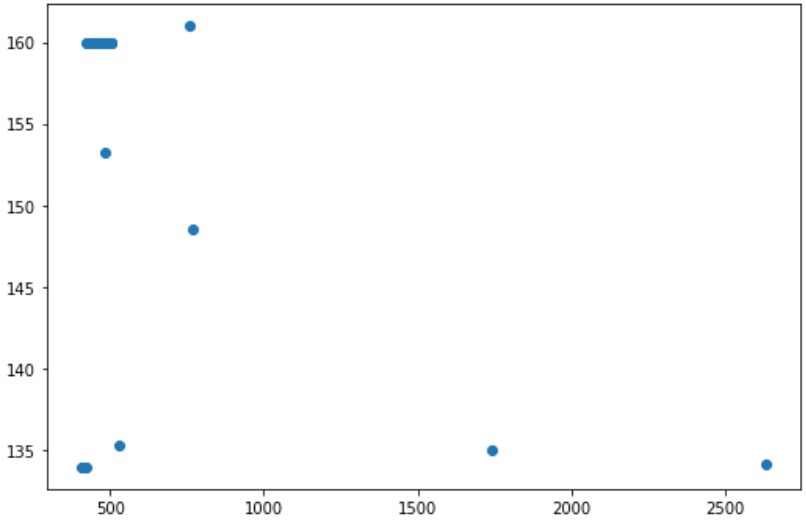
In [63]:
```
a = plt.subplots(1,2,figsize=(15,5))[1].ravel()
v = df[df['Time Consumed (Seconds)'] <= 400]
v1 = df[df['Time Consumed (Seconds)'] > 400]
a[0].scatter(v['Time Consumed (Seconds)'], v['Transaction Fees (Ether)'])
a[1].scatter(v1['Time Consumed (Seconds)'], v1['Transaction Fees (Ether)'])
plt.xlabel('Time ')
plt.ylabel('Value')
plt.title("Relation between Time Consumed and Transaction Fees.")
plt.tight_layout()
plt.show()
```

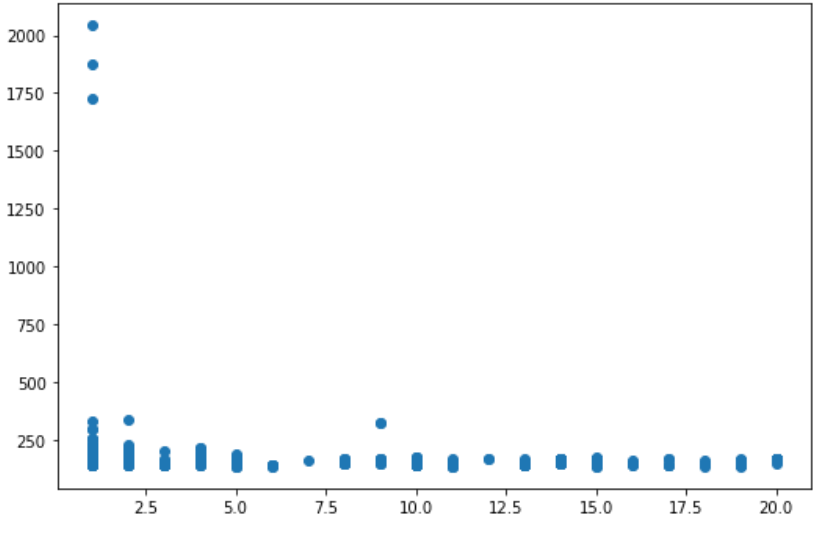


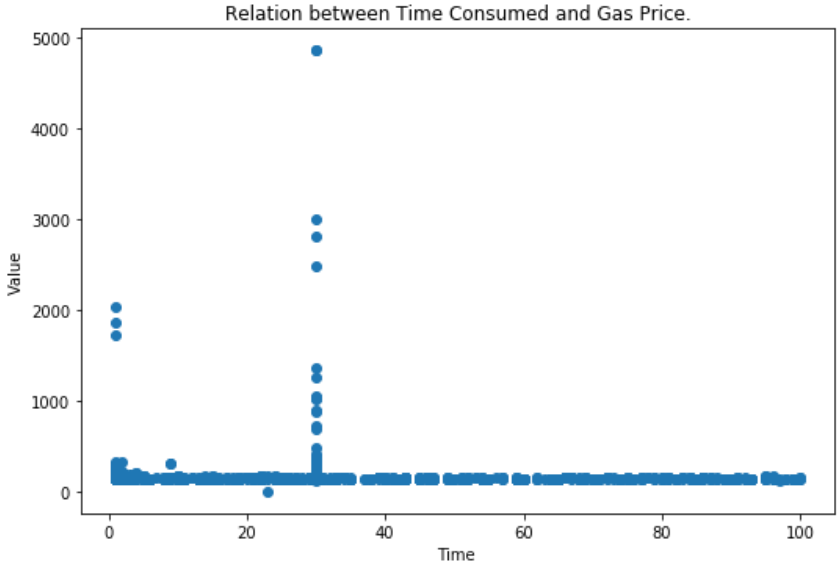
In [107]:
```
a = plt.subplots(2,2,figsize=(15,10))[1].ravel()
d = df[df['Time Consumed (Seconds)'] <= 20]
d['Gas Price (Ether)'] = d['Gas Price (Ether)']*1000000000
d1 = df[df['Time Consumed (Seconds)'] <= 100]
d1['Gas Price (Ether)'] = d1['Gas Price (Ether)']*1000000000
v = df[df['Time Consumed (Seconds)'] <= 400]
v['Gas Price (Ether)'] = v['Gas Price (Ether)']*1000000000
v1 = df[df['Time Consumed (Seconds)'] > 400]
v1['Gas Price (Ether)'] = v1['Gas Price (Ether)']*1000000000
a[0].scatter(v['Time Consumed (Seconds)'], v['Gas Price (Ether)'])
a[1].scatter(v1['Time Consumed (Seconds)'], v1['Gas Price (Ether)'])
a[2].scatter(d['Time Consumed (Seconds)'], d['Gas Price (Ether)'])
a[3].scatter(d1['Time Consumed (Seconds)'], d1['Gas Price (Ether)'])
plt.xlabel('Time ')
plt.ylabel('Value')
plt.title("Relation between Time Consumed and Gas Price.")
plt.tight_layout()
plt.show()
```

```
C:\Users\jain\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view
-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\jain\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view
-versus-copy
  """
C:\Users\jain\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view
-versus-copy
  import sys
C:\Users\jain\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view
-versus-copy
  if __name__ == '__main__':
```

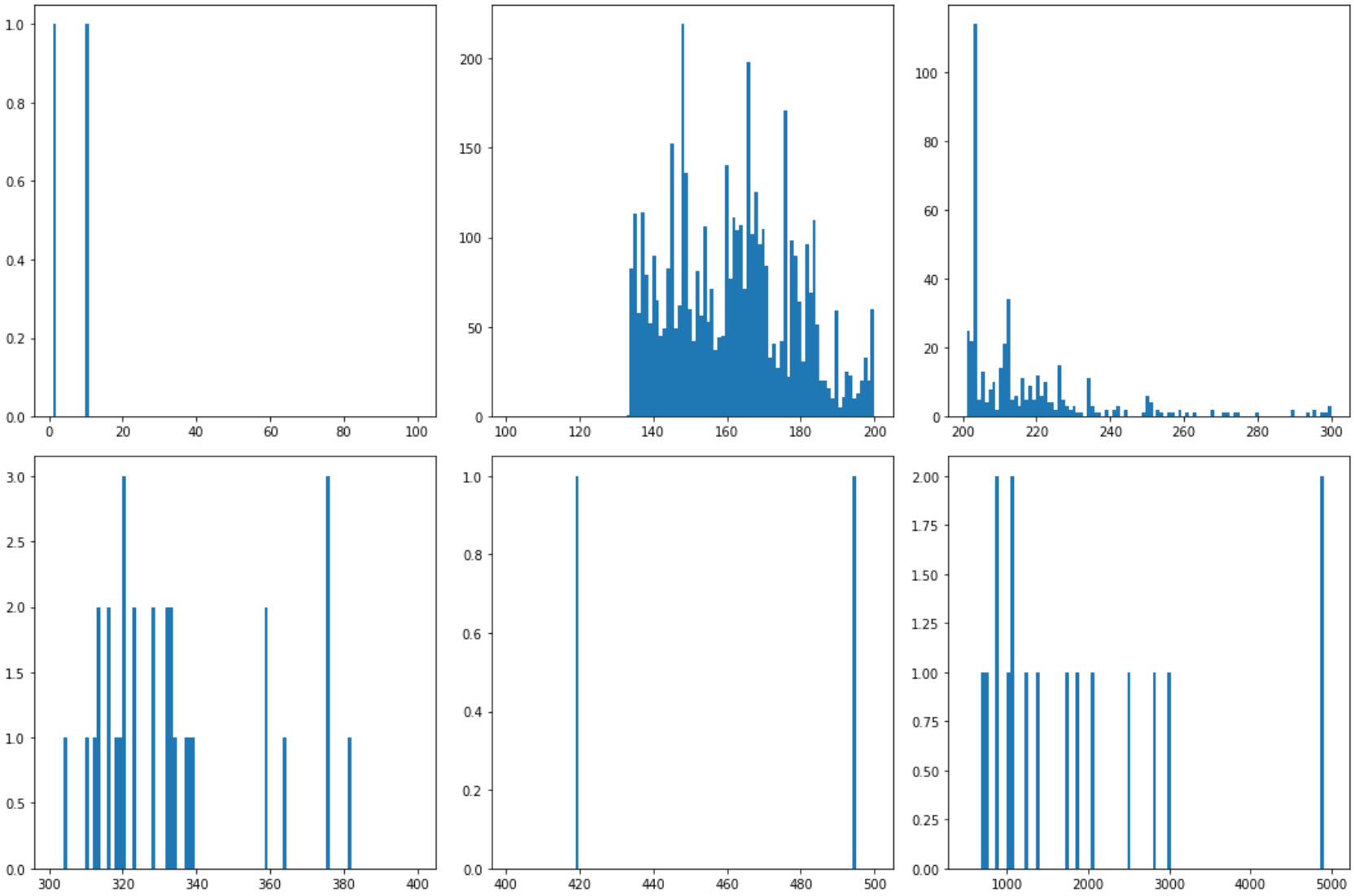

5000
4000
3000
2000
1000
0
0
50
100
150
200
250
300
350
400


160
155
150
145
140
135
500
1000
1500
2000
2500


2000
1750
1500
1250
1000
750
500
250
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Relation between Time Consumed and Gas Price.
Value
Time
5000
4000
3000
2000
1000
0
0
20
40
60
80
100

In [81]:
```
a = plt.subplots(2,3,figsize=(15,10))[1].ravel()
a[0].hist(df['Gas Price (Ether)']*1000000000,bins=100,range=[1,100])
a[1].hist(df['Gas Price (Ether)']*1000000000,bins=100,range=[101,200])
a[2].hist(df['Gas Price (Ether)']*1000000000,bins=100,range=[201,300])
a[3].hist(df['Gas Price (Ether)']*1000000000,bins=100,range=[301,400])
a[4].hist(df['Gas Price (Ether)']*1000000000,bins=100,range=[401,500])
a[5].hist(df['Gas Price (Ether)']*1000000000,bins=100,range=[501,5000])
#plt.tight_layout()
#plt.show()
#a[7].set_ylim([0,5])
plt.tight_layout()
plt.show()
```



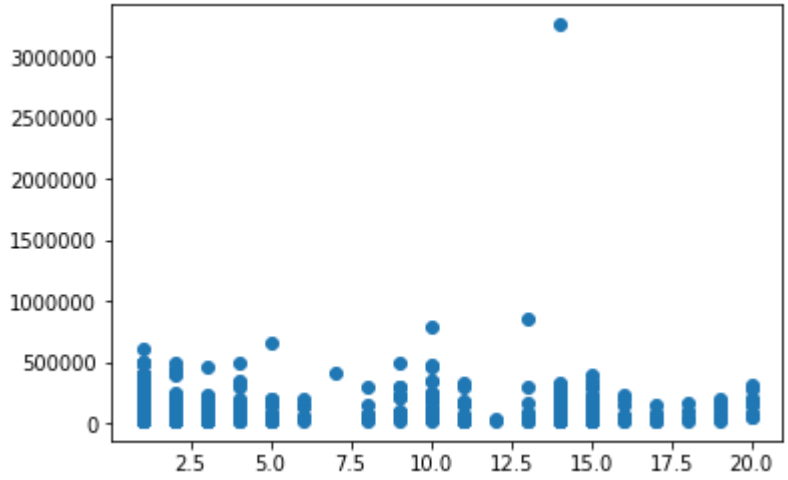
Analysis of Findings

1. **Gas Price is concentrated in the range of 130 to 220 Gwei. In normal cases, if user pays this gas price(near to 250 Gwei), the observed time taken would be <=500 seconds. For time greater than 500 seconds, the gas price is around maximum 170 Gwei.**
2. **There are some points where there is high gas price (around 5000 Gwei), the time taken is less than 40 seconds.**
3. **For the transactions completing below 20 seconds, the gas price is around 300 Gwei.**

In [111]:
```
v = df[df['Time Consumed (Seconds)'].between(0,20, inclusive=True)]
x = v.groupby('Gas Limit').count()
print(x['Status'].sort_values().tail(50))
plt.scatter(v['Time Consumed (Seconds)'], v['Gas Limit'])
plt.show()
```

Gas Limit	
87579	1
79038	1
65000	1
65630	1
65934	1
66033	1
86829	1
86790	1
85566	1
66127	1
43738	2
41209	2
21500	2
21408	2
140000	2
400000	2
57896	2
50165	2
61813	2
48714	2
48870	2
70000	2
213325	2
48959	2
50000	2
54485	2
95500	3
395317	3
55181	3
351400	3
210000	3
48977	3
48846	3
40000	3
42096	4
420000	4
75247	5
250000	6
64200	6
80000	6
60010	6
207128	7

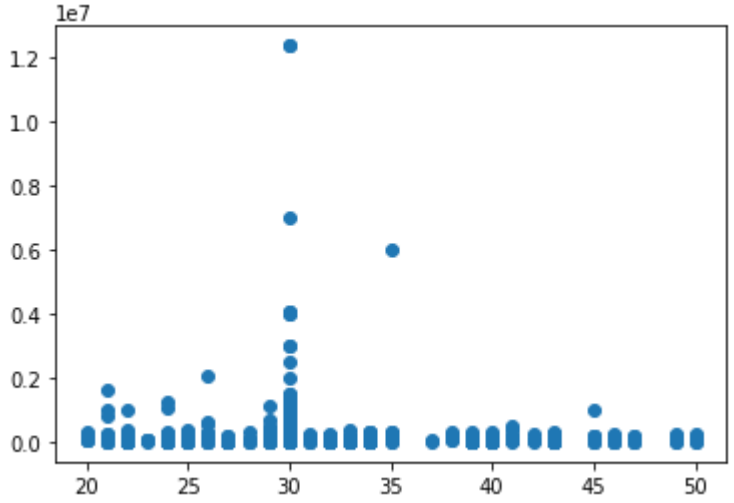
```
500000      8
300000     10
150000     13
90000      19
200000     29
100000     40
60000      45
21000     128
Name: Status, dtype: int64
```



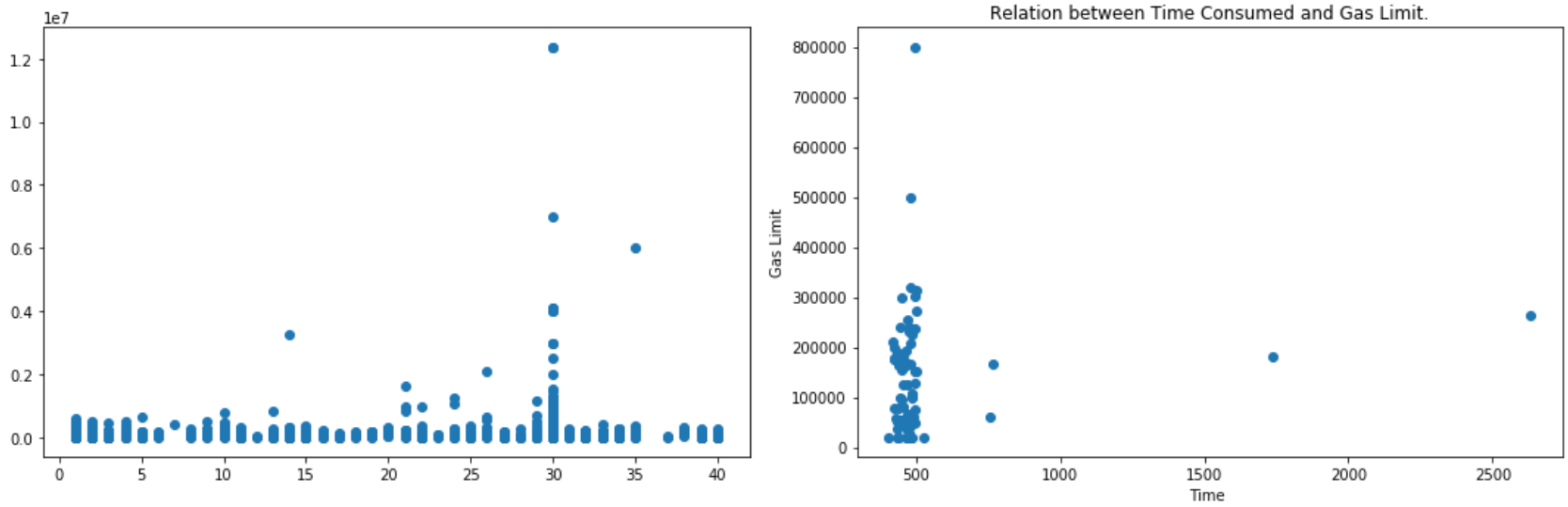
In [113]:
```
v = df[df['Time Consumed (Seconds)'].between(20,50, inclusive=True)]
x = v.groupby('Gas Limit').count()
print(x['Status'].sort_values().tail(50))
plt.scatter(v['Time Consumed (Seconds)'], v['Gas Limit'])
plt.show()
```

Gas Limit	
66853	5
84000	5
56836	5
75247	5
41197	5
350000	5
61813	6
67451	6
67918	6
48846	6
51179	6
395317	6
1170500	7
48959	7
35000	7
65000	7
56209	7
82442	8
60001	8
84313	9
60010	11
25200	12
45000	12
210000	12
82418	12
70000	13
30000	13
120000	13
105000	15
1000000	15
41209	15
180000	15
600000	18
49070	21
400000	22
42000	23
800000	25
50000	29
420000	32
80000	51
150000	56
300000	69

```
90000       78
200000      84
250000     100
500000     123
100000     139
60000      149
207128     215
21000      929
Name: Status, dtype: int64
```



In [114]:
```
a = plt.subplots(1,2,figsize=(15,5))[1].ravel()
v = df[df['Time Consumed (Seconds)'] <= 40]
v1 = df[df['Time Consumed (Seconds)'] > 400]
a[0].scatter(v['Time Consumed (Seconds)'], v['Gas Limit'])
a[1].scatter(v1['Time Consumed (Seconds)'], v1['Gas Limit'])
plt.xlabel('Time ')
plt.ylabel('Gas Limit')
plt.title("Relation between Time Consumed and Gas Limit.")
plt.tight_layout()
plt.show()
```



In [115]:
```
a = plt.subplots(1,2,figsize=(15,5))[1].ravel()
v = df[df['Time Consumed (Seconds)'].between(150,500, inclusive=True)]
print(v.groupby('Gas Limit').count().tail(25))
v1 = df[df['Time Consumed (Seconds)'] > 50]
a[0].scatter(v['Time Consumed (Seconds)'], v['Gas Limit'])
a[1].scatter(v1['Time Consumed (Seconds)'], v1['Gas Limit'])
```

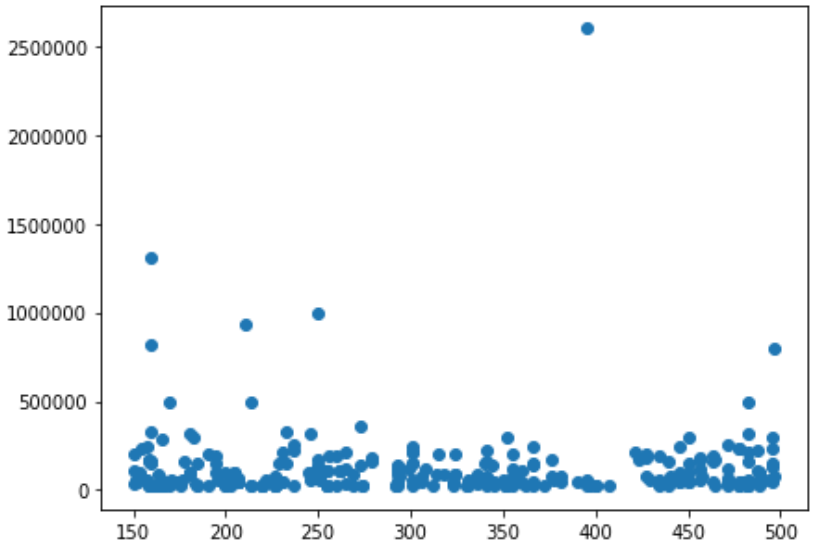
Gas Limit	Unnamed: 0	Transaction Hash	Status	Time Consumed (Seconds)
229125	1	1	1	1
230651	1	1	1	1
238434	1	1	1	1
239222	1	1	1	1
239452	1	1	1	1
240063	1	1	1	1
240639	1	1	1	1
250000	2	2	2	2
254733	1	1	1	1
254999	1	1	1	1
283152	1	1	1	1
300000	3	3	3	3
302112	1	1	1	1
315447	2	2	2	2
320140	1	1	1	1
326478	1	1	1	1
331604	1	1	1	1
355506	1	1	1	1
500000	3	3	3	3
800000	1	1	1	1
819200	1	1	1	1
938489	1	1	1	1
1000000	1	1	1	1
1312134	1	1	1	1
2601641	1	1	1	1

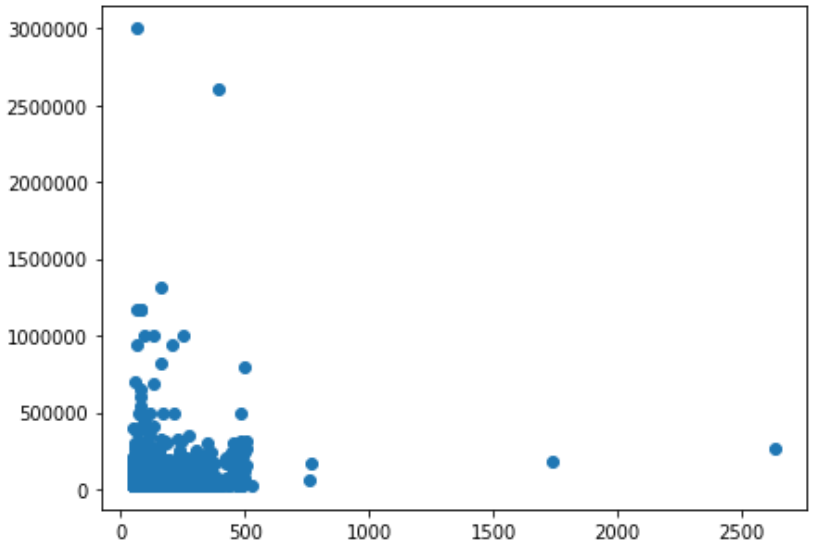
Gas Limit	Value (Ether)	Transaction Fees (Ether)	Gas Price (Ether)
229125	1	1	1
230651	1	1	1
238434	1	1	1
239222	1	1	1
239452	1	1	1
240063	1	1	1
240639	1	1	1
250000	2	2	2
254733	1	1	1
254999	1	1	1
283152	1	1	1
300000	3	3	3
302112	1	1	1

315447	2	2	2
320140	1	1	1
326478	1	1	1
331604	1	1	1
355506	1	1	1
500000	3	3	3
800000	1	1	1
819200	1	1	1
938489	1	1	1
1000000	1	1	1
1312134	1	1	1
2601641	1	1	1

Gas Limit	Gas Used
229125	1
230651	1
238434	1
239222	1
239452	1
240063	1
240639	1
250000	2
254733	1
254999	1
283152	1
300000	3
302112	1
315447	2
320140	1
326478	1
331604	1
355506	1
500000	3
800000	1
819200	1
938489	1
1000000	1
1312134	1
2601641	1

Out[115]: <matplotlib.collections.PathCollection at 0x154476e5d68>


2500000
2000000
1500000
1000000
500000
0
150
200
250
300
350
400
450
500


3000000
2500000
2000000
1500000
1000000
500000
0
0
500
1000
1500
2000
2500

In [91]:
```
a = plt.subplots(4,3,figsize=(15,10))[1].ravel()
a[0].hist(df['Gas Limit'],bins=30,range=[21000,25000])
a[1].hist(df['Gas Limit'],bins=30,range=[25000,35000])
a[2].hist(df['Gas Limit'],bins=30,range=[35000,55000])
a[3].hist(df['Gas Limit'],bins=30,range=[55000,80000])
a[4].hist(df['Gas Limit'],bins=30,range=[80000,100000])
a[5].hist(df['Gas Limit'],bins=30,range=[100000,150000])
a[6].hist(df['Gas Limit'],bins=30,range=[150000,250000])
a[7].hist(df['Gas Limit'],bins=30,range=[250000,500000])
a[8].hist(df['Gas Limit'],bins=30,range=[500000,600000])
a[9].hist(df['Gas Limit'],bins=30,range=[600000,700000])
a[10].hist(df['Gas Limit'],bins=30,range=[700000,1000000])
a[11].hist(df['Gas Limit'],bins=30,range=[1000000,1000000000])
#plt.tight_layout()
#plt.show()
#a[7].set_ylim([0,5])
plt.tight_layout()
plt.show()
```

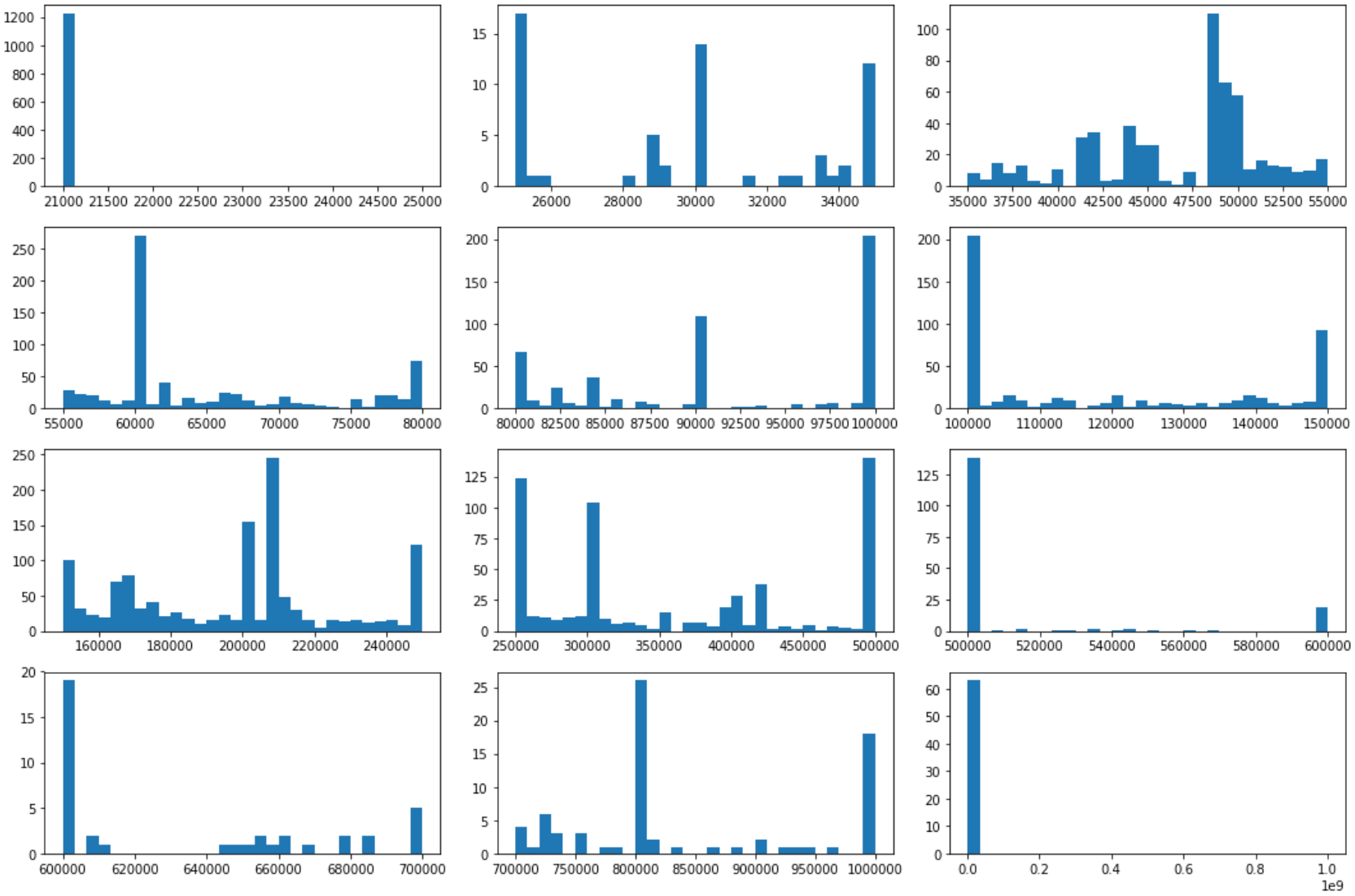


Table of Findings of Relation between Time Consumed with Gas Price and Gas Limit.

Gas Limit	Time Taken	Remarks	Gas Price
<= 3,00,000	0-20 sec	~42% (majority 21000 gas limit)	140-300(gas price in gwei), some high exceptions of gas price.
<= 5,00,000	20-400 sec	(majority 21000 gas limit)	140-200(gas price in gwei).
<= 3,00,000	40-2700 sec	Having wide array of values (sparsed)	130-175(gas price in gwei)

In [125]:
```
df['Gas Left'] = df['Gas Limit'] - df['Gas Used']
df['Gas Left'].describe()
#plt.hist(df['Gas Left'],bins=30,range=[0,10000])
```

Out[125]:
```
count    5.169000e+03
mean     9.048802e+04
std      3.243448e+05
min      0.000000e+00
25%      0.000000e+00
50%      3.000000e+04
75%      7.592900e+04
max      1.235556e+07
Name: Gas Left, dtype: float64
```

Findings on Gas Limit and Gas Used.

1. Around 25% of transactions consumed whole gas. ANd around 75% of trasnactions having gas unused <= 75929. There are few trasnactions having high unused gas value.