Data Exploration

Importing Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading the data

In [2]:
```
data = pd.read_excel("Inconel_Compiled_Data_New_elems.xlsx")
```

Display First 5 rows

In [3]:
```
data.head(5)
```

Out[3]:

	Type	Ni%	Cr%	Fe%	Mn%	Cu%	Al%	Si%	C%	S%	...	Ni% + Co%	W%	La%	Zr%	(Am
0	Inconel 600	72.0	15.5	8.0	1.0	0.5	0.5	0.15	0.015	0.0	...	0.0	0	0	0	130
1	Inconel 600	72.0	15.5	8.0	1.0	0.5	0.5	0.15	0.015	0.0	...	0.0	0	0	0	120
2	Inconel 600	72.0	15.5	8.0	1.0	0.5	0.5	0.15	0.015	0.0	...	0.0	0	0	0	120
3	Inconel 600	72.0	15.5	8.0	1.0	0.5	0.5	0.15	0.015	0.0	...	0.0	0	0	0	100
4	Inconel 600	72.0	15.5	8.0	1.0	0.5	0.5	0.15	0.015	0.0	...	0.0	0	0	0	120

5 rows × 26 columns

Display last 5 rows

In [4]:
```
data.tail(5)
```

Out[4]:

	Type	Ni%	Cr%	Fe%	Mn%	Cu%	Al%	Si%	C%	S%	...	Ni% + Co%	W%	La%	Zr%	(A
722	Inconel 825	42.0	21.5	33.0	1.0	2.25	0.2	0.5	0.05	0.03	...	0.0	0	0	0	
723	Inconel 825	42.0	21.5	33.0	1.0	2.25	0.2	0.5	0.05	0.03	...	0.0	0	0	0	
724	Inconel 825	42.0	21.5	33.0	1.0	2.25	0.2	0.5	0.05	0.03	...	0.0	0	0	0	
725	Inconel 825	42.0	21.5	33.0	1.0	2.25	0.2	0.5	0.05	0.03	...	0.0	0	0	0	
726	Inconel 825	42.0	21.5	33.0	1.0	2.25	0.2	0.5	0.05	0.03	...	0.0	0	0	0	

5 rows × 26 columns

Shape of Data

In [5]:

```
data.shape
```

Out[5]:

```
(727, 26)
```

Columns in Data

In [6]:

```
data.columns
```

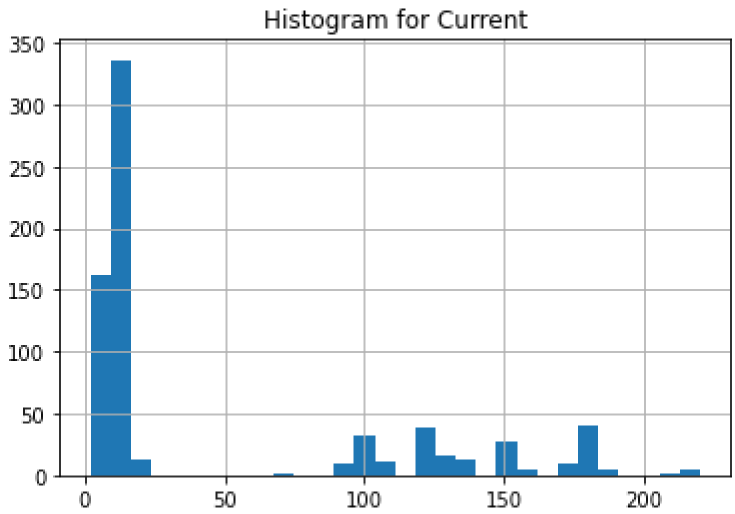
Out[6]:

```
Index(['Type', 'Ni%', 'Cr%', 'Fe%', 'Mn%', 'Cu%', 'Al%', 'Si%', 'C%', 'S%',
       'Mo%', 'Ti%', 'Co%', 'B%', 'P%', 'Nb & Ta%', 'Ni% + Co%', 'W%', 'La%',
       'Zr%', 'IP (Amp)', 'Ton (µS)', 'Toff  (µS)', 'Voltage (Volts)',
       'Surface Roughness (µm)', 'MRR (mm3/min)'],
      dtype='object')
```

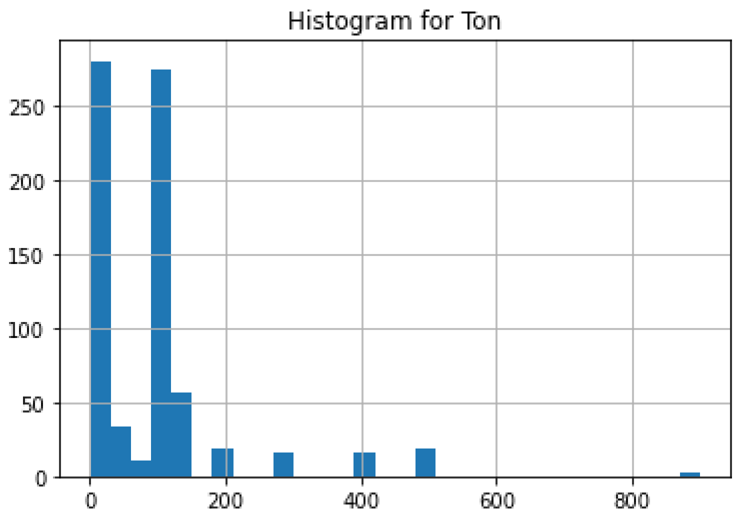
Distribution of Variables in Dataset

In [7]:

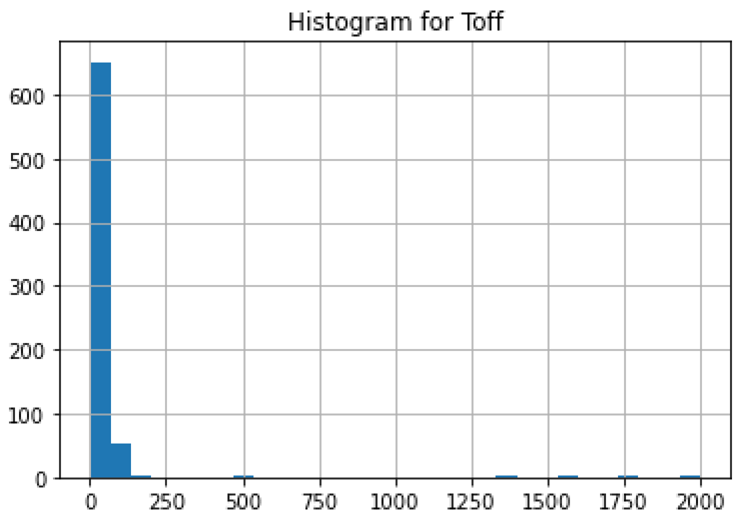
```
data.hist(column = 'IP (Amp)', bins =30, );
plt.title("Histogram for Current");
plt.savefig("Histogram for Current.jpg")
```



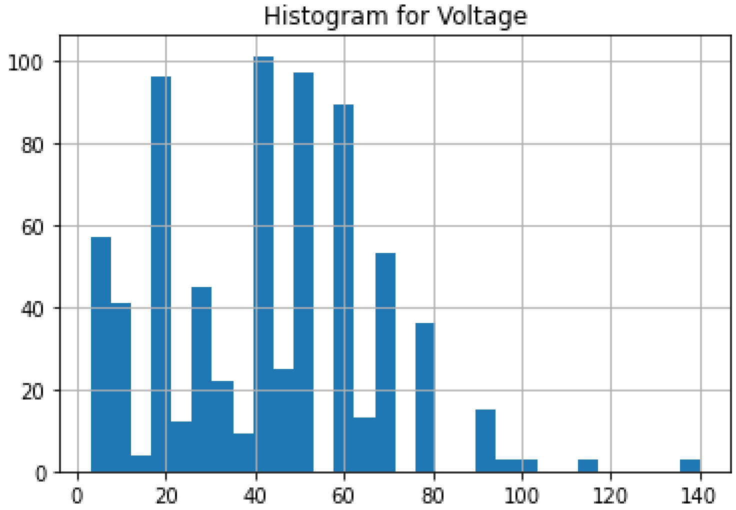

```
In [8]: data.hist(column = 'Ton (μS)', bins =30, );
        plt.title("Histogram for Ton");
        plt.savefig("Histogram for Ton.jpg")
```



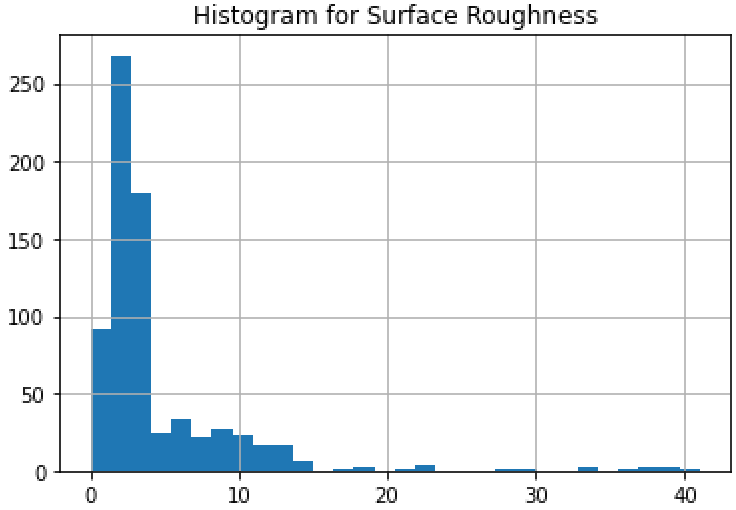

```
In [9]: data.hist(column = 'Toff  (μS)', bins =30, );
        plt.title("Histogram for Toff");
        plt.savefig("Histogram for Toff.jpg")
```



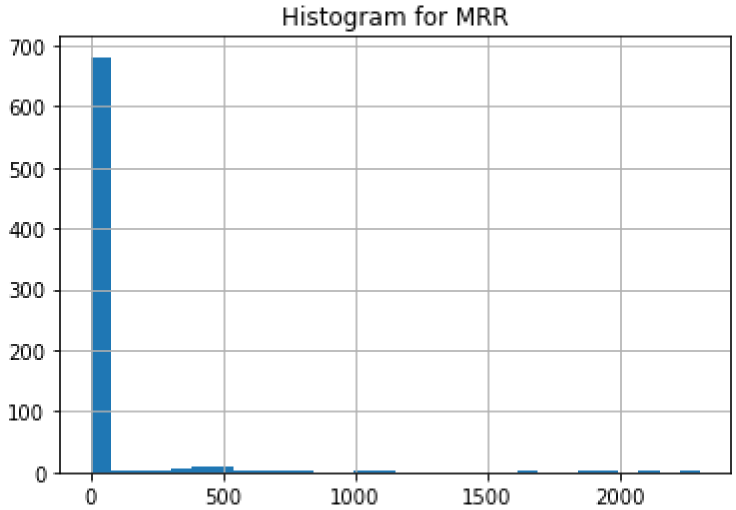

```
In [10]: data.hist(column = 'Voltage (Volts)', bins =30, );
         plt.title("Histogram for Voltage");
         plt.savefig("Histogram for VOltage.jpg")
```



In [11]:
```
data.hist(column = 'Surface Roughness (µm)', bins =30, );
plt.title("Histogram for Surface Roughness");
plt.savefig("Histogram for Surface Roughness.jpg")
```



In [12]:
```
data.hist(column = 'MRR (mm3/min)', bins =30, );
plt.title("Histogram for MRR");
plt.savefig("Histogram for MRR.jpg")
```



Vizualizing Outliers

In [13]:
```python
data.iloc[:,-6:].describe()
```

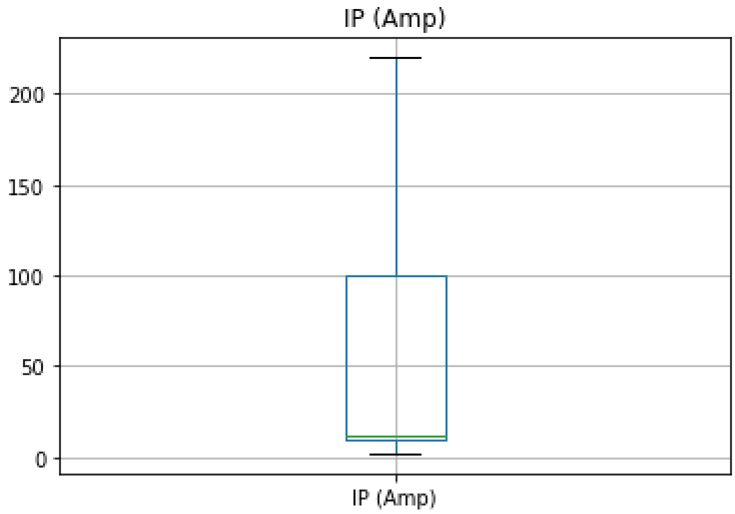
Out[13]:

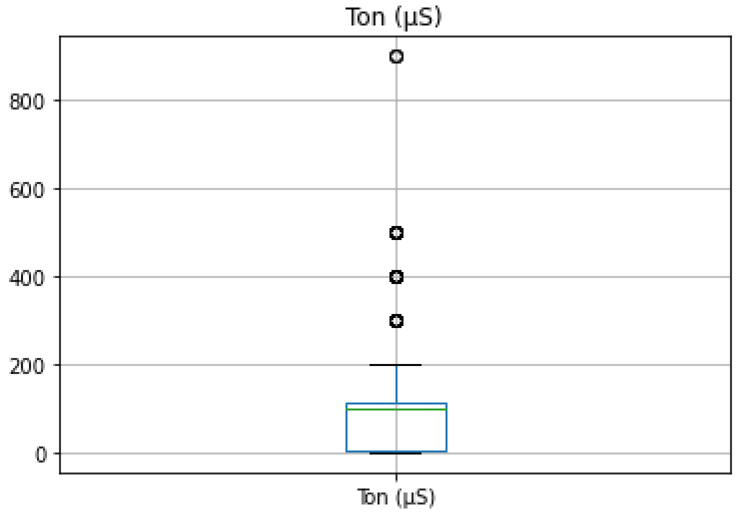
	IP (Amp)	Ton (μS)	Toff (μS)	Voltage (Volts)	Surface Roughness (μm)	MRR (mm3/min)
count	727.000000	727.000000	727.000000	727.000000	7.270000e+02	727.000000
mean	48.738990	94.145503	79.360326	42.474796	4.368155e+00	57.191972
std	62.119464	116.743802	248.672805	23.937343	5.154001e+00	245.127583
min	2.000000	0.350000	1.000000	3.045000	7.500000e-07	0.003200
25%	10.000000	3.000000	20.000000	20.000000	1.851500e+00	2.181462
50%	12.000000	100.000000	45.000000	40.000000	2.750000e+00	6.732200
75%	100.000000	115.000000	56.000000	60.000000	4.286490e+00	13.118396
max	220.000000	900.000000	2000.000000	140.000000	4.100000e+01	2302.000000

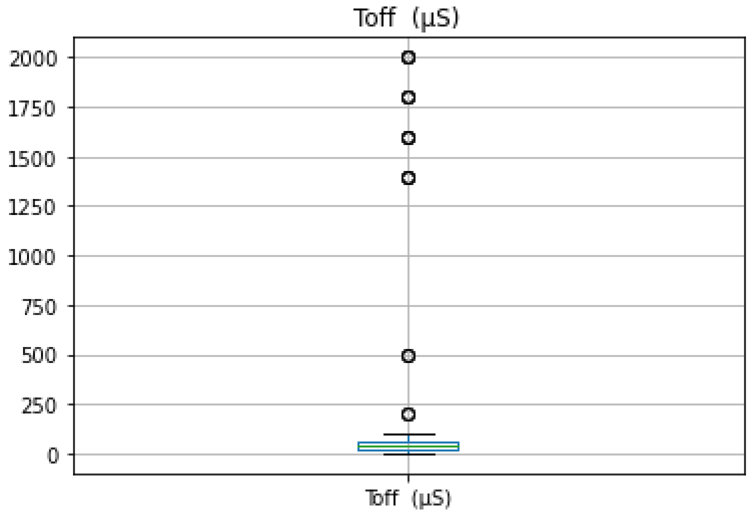
In [14]:
```python
data.columns[-6:]
```

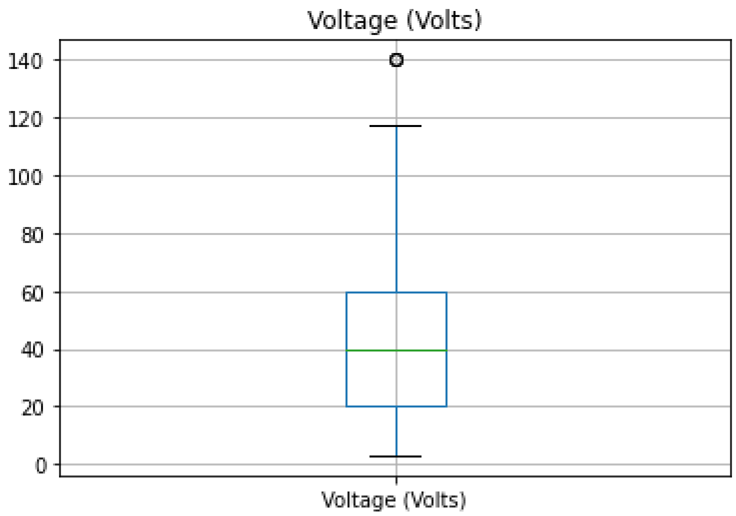
Out[14]:
```
Index(['IP (Amp)', 'Ton (μS)', 'Toff  (μS)', 'Voltage (Volts)',
       'Surface Roughness (μm)', 'MRR (mm3/min)'],
      dtype='object')
```

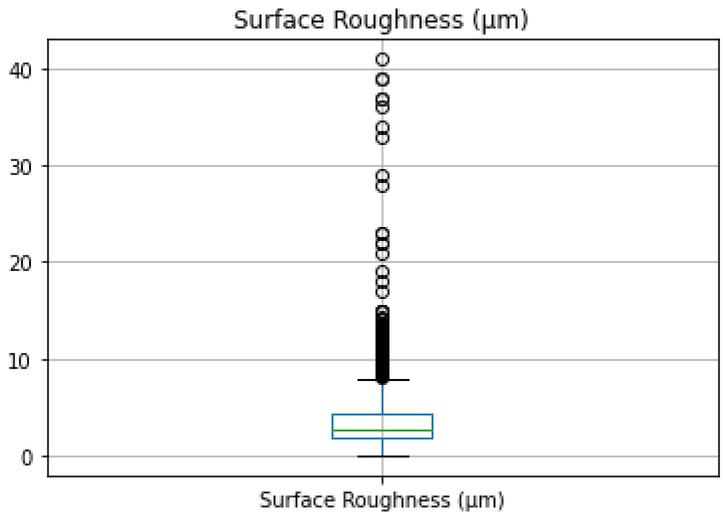
In [15]:
```python
for i in data.iloc[:,-6:].columns:

    data.boxplot(column = [i])
    plt.title(i)
    plt.show()
```

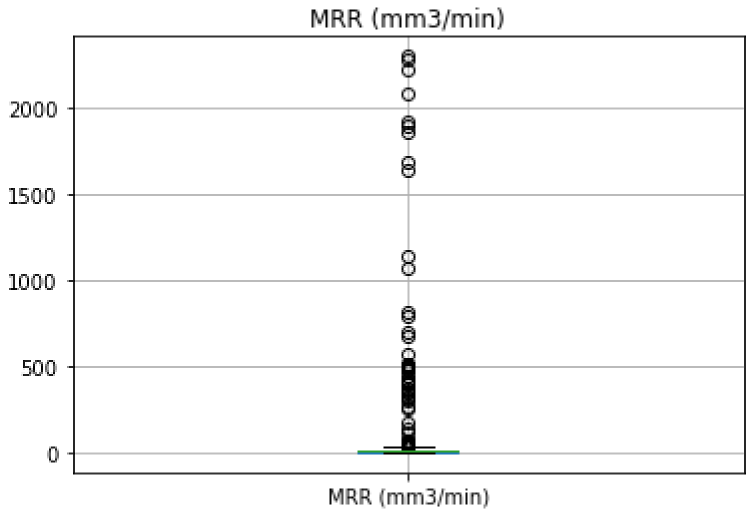



Ton (µS)
800
600
400
200
0
Ton (µS)


Toff (µS)
2000
1750
1500
1250
1000
750
500
250
0
Toff (µS)


Voltage (Volts)
140
120
100
80
60
40
20
0
Voltage (Volts)





Creating Correlation Matrix

In [10]:
```
df2 = data.drop(['W%', 'La%', 'Zr%'], axis =1) # Dropping these, because they are :
```

In [7]:
```
corr_matrix = df2.corr()
```

In [8]:
```
corr_matrix.to_csv("corr_matrix.csv")
corr_matrix
```

Out[8]:

	Ni%	Cr%	Fe%	Mn%	Cu%	Al%	Si%	C%
Ni%	1.000000	-0.241449	-0.815426	0.204475	0.021585	0.230782	0.339167	0.052003
Cr%	-0.241449	1.000000	-0.014964	-0.190396	0.054960	-0.462056	0.013952	-0.079402
Fe%	-0.815426	-0.014964	1.000000	0.134946	0.273079	-0.067373	-0.457861	-0.169615
Mn%	0.204475	-0.190396	0.134946	1.000000	0.357460	0.298055	-0.258610	-0.235116
Cu%	0.021585	0.054960	0.273079	0.357460	1.000000	0.183165	-0.061639	-0.349432
Al%	0.230782	-0.462056	-0.067373	0.298055	0.183165	1.000000	0.018397	0.178207
Si%	0.339167	0.013952	-0.457861	-0.258610	-0.061639	0.018397	1.000000	0.821803
C%	0.052003	-0.079402	-0.169615	-0.235116	-0.349432	0.178207	0.821803	1.000000
S%	0.056967	0.201795	-0.028094	0.253340	0.379147	0.738443	0.062966	0.099800
Mo%	-0.072141	0.148388	-0.328783	-0.463502	-0.497284	-0.189033	0.216808	0.376378
Ti%	0.061289	-0.563572	0.055017	0.109540	0.083928	0.278294	0.062803	0.077767
Co%	0.237943	-0.285657	-0.254343	-0.281380	-0.259914	0.036526	0.883519	0.867593
B%	-0.180025	-0.266436	0.268173	-0.665306	-0.173550	0.050856	0.186171	0.345573
P%	-0.451251	-0.032814	0.104567	-0.781121	-0.499575	-0.099699	-0.117033	0.147583
Nb & Ta%	-0.359502	-0.132501	0.150208	-0.775616	-0.424330	-0.020294	-0.152228	0.116958
Ni% + Co%	-0.579911	-0.120456	0.307148	-0.087913	-0.016033	-0.038819	-0.031168	-0.059952
IP (Amp)	-0.175781	0.117116	0.244051	0.073742	-0.114565	-0.171451	-0.184460	-0.062141
Ton (µS)	0.065846	-0.106596	-0.062890	0.097710	0.207097	-0.081461	-0.010333	-0.126843
Toff (µS)	0.184125	-0.182887	-0.117384	0.191872	0.071253	0.099888	0.029446	-0.034582
Voltage (Volts)	-0.170655	0.004346	0.272676	-0.033262	0.146685	-0.237947	-0.227864	-0.236877
Surface Roughness (µm)	0.141494	-0.200684	-0.077889	0.113928	0.097405	0.117861	0.061043	0.044878
MRR (mm3/min)	0.266048	-0.242102	-0.119109	0.246977	0.070156	0.021460	-0.173212	-0.290396

22 rows × 22 columns

Correlation of SR

In [19]:

```
corr_matrix["Surface Roughness (µm)"].sort_values(ascending = False)
```

Out[19]:
```
Surface Roughness (µm)    1.000000
Ti%                       0.381024
Voltage (Volts)           0.296748
Ton (µS)                  0.205931
IP (Amp)                  0.168366
Ni%                       0.141494
Al%                       0.117861
Mn%                       0.113928
Cu%                       0.097405
Toff  (µS)                0.068431
Si%                       0.061043
C%                        0.044878
Mo%                       0.023675
Co%                       0.017308
Nb & Ta%                  0.003342
P%                       -0.050528
MRR (mm3/min)            -0.054053
B%                       -0.061827
S%                       -0.076050
Fe%                      -0.077889
Ni% + Co%                -0.094954
Cr%                      -0.200684
Name: Surface Roughness (µm), dtype: float64
```

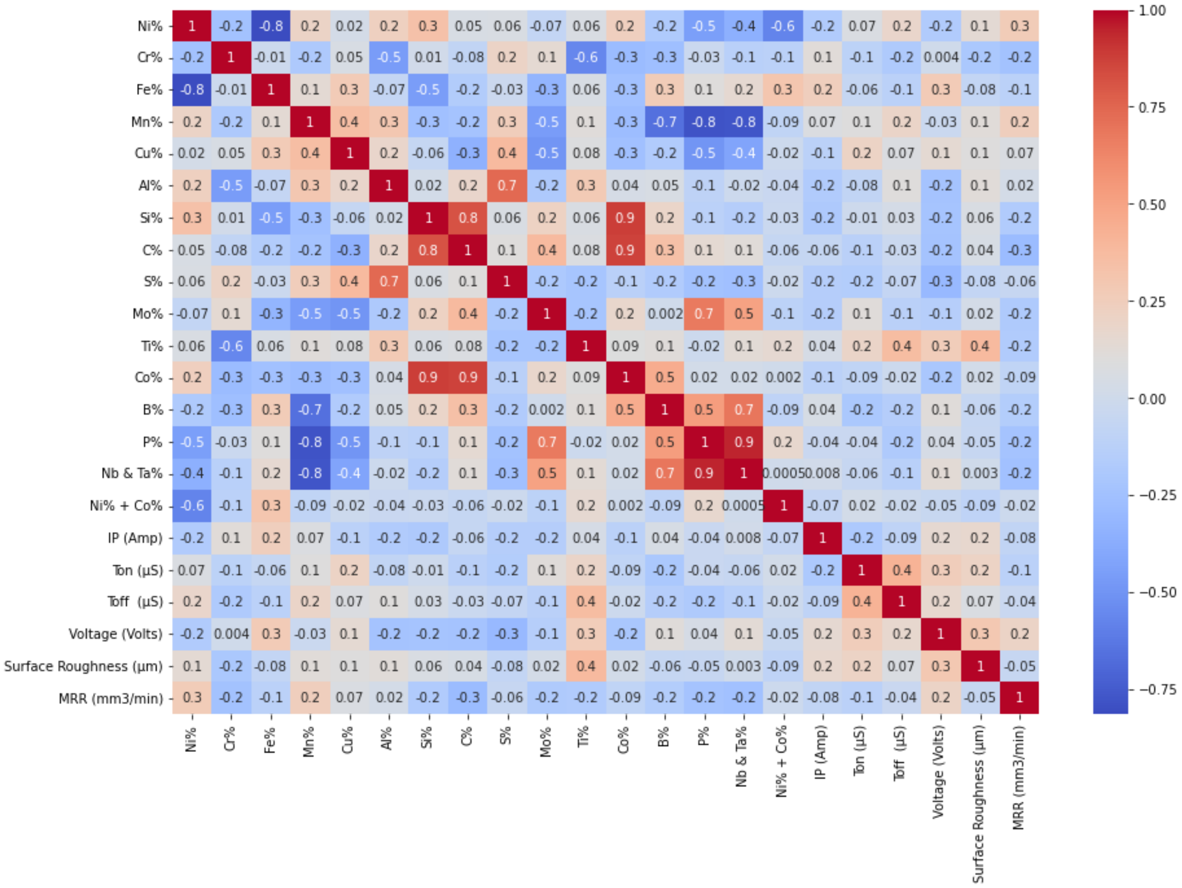
Correlation of MRR

In [9]:
```
corr_matrix["MRR (mm3/min)"].sort_values(ascending = False)
```

Out[9]:
```
MRR (mm3/min)             1.000000
Ni%                       0.266048
Mn%                       0.246977
Voltage (Volts)           0.203338
Cu%                       0.070156
Al%                       0.021460
Ni% + Co%                -0.016570
Toff  (µS)               -0.042217
Surface Roughness (µm)   -0.054053
S%                       -0.055673
IP (Amp)                 -0.078989
Co%                      -0.086475
Ton (µS)                 -0.118201
Fe%                      -0.119109
B%                       -0.160543
Ti%                      -0.170182
Si%                      -0.173212
Mo%                      -0.180036
P%                       -0.237810
Cr%                      -0.242102
Nb & Ta%                 -0.249457
C%                       -0.290396
Name: MRR (mm3/min), dtype: float64
```

Vizualizing Correlation Matrix

In [19]:
```
plt.figure(figsize=(15,10))
g = sns.heatmap(corr_matrix,  annot = True, fmt='.1g', cmap= 'coolwarm')
plt.savefig("Correlation_Matrix.jpg")
```

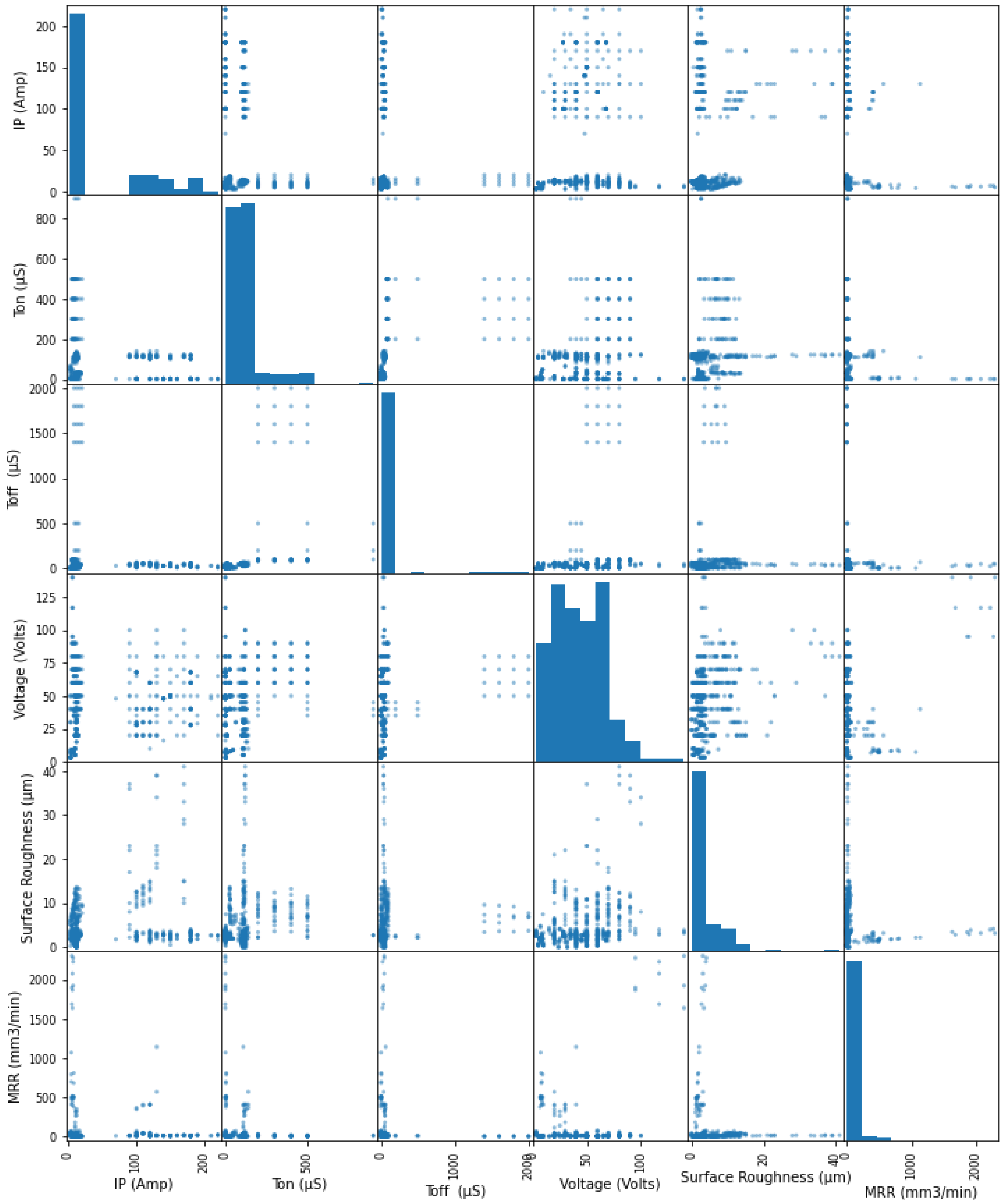


In [21]:
```python
from pandas.plotting import scatter_matrix
```

In [22]:
```python
attributes = ['IP (Amp)',
         'Ton (µS)',
         'Toff  (µS)',
         'Voltage (Volts)',
         'Surface Roughness (µm)',
         'MRR (mm3/min)'
              ]
```

Plotting Scatter Matrix

In [23]:
```python
scatter_matrix(data[attributes], figsize=(12, 15));
plt.savefig("scatter_matrix.jpg")
```



Check For Multicolinearity

In [24]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):

    # Calculating VIF
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])

    return(vif)
```

In [25]:
```
df3 = data.iloc[:,-6:-2]
```

In [26]:
```
calc_vif(df3)
```

Out[26]:

	variables	VIF
0	IP (Amp)	1.612607
1	Ton (μS)	2.077908
2	Toff (μS)	1.347022
3	Voltage (Volts)	2.555894

Since all the VIF values are below 5, there is very low multicolinearity between independent variabes.

source: https://www.analyticsvidhya.com/blog/2020/03/what-is-multicollinearity/