Initial Data Exploration

This notebook will be used to explore the dataset and to check key values, changes to be made, overall quality of the data. The following hidden cell is where the data is loaded into the workspace and the csv transformed into a dataFrame.

In [8]:
```
# The code was removed by Watson Studio for sharing.
```

The dataFrame is renamed and the Id column renamed so that it can be known what that column is.

In [9]:
```
#Change the dataFrame name
data_df = df_data_1

#Rename Unnamed: 0 to Id for readability
data_df.rename(columns = {data_df.columns[0]: 'Id'}, inplace = True)
data_df.head()
```

Out[9]:

	Id	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_05	sensor_06
0	0	2018-04-01 00:00:00	2.465394	47.09201	53.2118	46.310760	634.3750	76.45975	13.41146
1	1	2018-04-01 00:01:00	2.465394	47.09201	53.2118	46.310760	634.3750	76.45975	13.41146
2	2	2018-04-01 00:02:00	2.444734	47.35243	53.2118	46.397570	638.8889	73.54598	13.32465
3	3	2018-04-01 00:03:00	2.460474	47.09201	53.1684	46.397568	628.1250	76.98898	13.31742
4	4	2018-04-01 00:04:00	2.445718	47.13541	53.2118	46.397568	636.4583	76.58897	13.35359

In [3]:
```
#Check the types of each of the dataFrames
data_df.dtypes
```

Out[3]:
```
Id                 int64
timestamp         object
sensor_00        float64
sensor_01        float64
sensor_02        float64
sensor_03        float64
sensor_04        float64
sensor_05        float64
sensor_06        float64
sensor_07        float64
sensor_08        float64
sensor_09        float64
sensor_10        float64
sensor_11        float64
sensor_12        float64
sensor_13        float64
sensor_14        float64
sensor_15        float64
sensor_16        float64
sensor_17        float64
sensor_18        float64
sensor_19        float64
sensor_20        float64
sensor_21        float64
sensor_22        float64
sensor_23        float64
sensor_24        float64
sensor_25        float64
sensor_26        float64
sensor_27        float64
sensor_28        float64
sensor_29        float64
sensor_30        float64
sensor_31        float64
sensor_32        float64
sensor_33        float64
sensor_34        float64
sensor_35        float64
sensor_36        float64
sensor_37        float64
sensor_38        float64
sensor_39        float64
sensor_40        float64
sensor_41        float64
sensor_42        float64
sensor_43        float64
sensor_44        float64
sensor_45        float64
sensor_46        float64
sensor_47        float64
sensor_48        float64
sensor_49        float64
sensor_50        float64
sensor_51        float64
```

```
machine_status     object
dtype: object
```

The data types seem to make sense. All the sensors are float values and the machine_status is object. The machine_status will eventually need to be turned to int and the timestamp to a datetime, if timestamp column is needed.

In [4]:
```
#Pandas describe is called to check the mean and other key statistics in the data
data_df.describe()
```

Out[4]:

	Id	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04
count	220320.000000	210112.000000	219951.000000	220301.000000	220301.000000	220301.000000
mean	110159.500000	2.372221	47.591611	50.867392	43.752481	590.673936
std	63601.049991	0.412227	3.296666	3.666820	2.418887	144.023912
min	0.000000	0.000000	0.000000	33.159720	31.640620	2.798032
25%	55079.750000	2.438831	46.310760	50.390620	42.838539	626.620400
50%	110159.500000	2.456539	48.133678	51.649300	44.227428	632.638916
75%	165239.250000	2.499826	49.479160	52.777770	45.312500	637.615723
max	220319.000000	2.549016	56.727430	56.032990	48.220490	800.000000

In [5]:
```
#Due to inconsistent numbers in the describe function, Nan amounts are needed.
data_df.isna().sum()
```

Out[5]:
```
Id                 0
timestamp          0
sensor_00      10208
sensor_01        369
sensor_02         19
sensor_03         19
sensor_04         19
sensor_05         19
sensor_06       4798
sensor_07       5451
sensor_08       5107
sensor_09       4595
sensor_10         19
sensor_11         19
sensor_12         19
sensor_13         19
sensor_14         21
sensor_15     220320
sensor_16         31
sensor_17         46
sensor_18         46
sensor_19         16
sensor_20         16
sensor_21         16
sensor_22         41
sensor_23         16
sensor_24         16
sensor_25         36
sensor_26         20
sensor_27         16
sensor_28         16
sensor_29         72
sensor_30        261
sensor_31         16
sensor_32         68
sensor_33         16
sensor_34         16
sensor_35         16
sensor_36         16
sensor_37         16
sensor_38         27
sensor_39         27
sensor_40         27
sensor_41         27
sensor_42         27
sensor_43         27
sensor_44         27
sensor_45         27
sensor_46         27
sensor_47         27
sensor_48         27
sensor_49         27
sensor_50      77017
sensor_51      15383
```

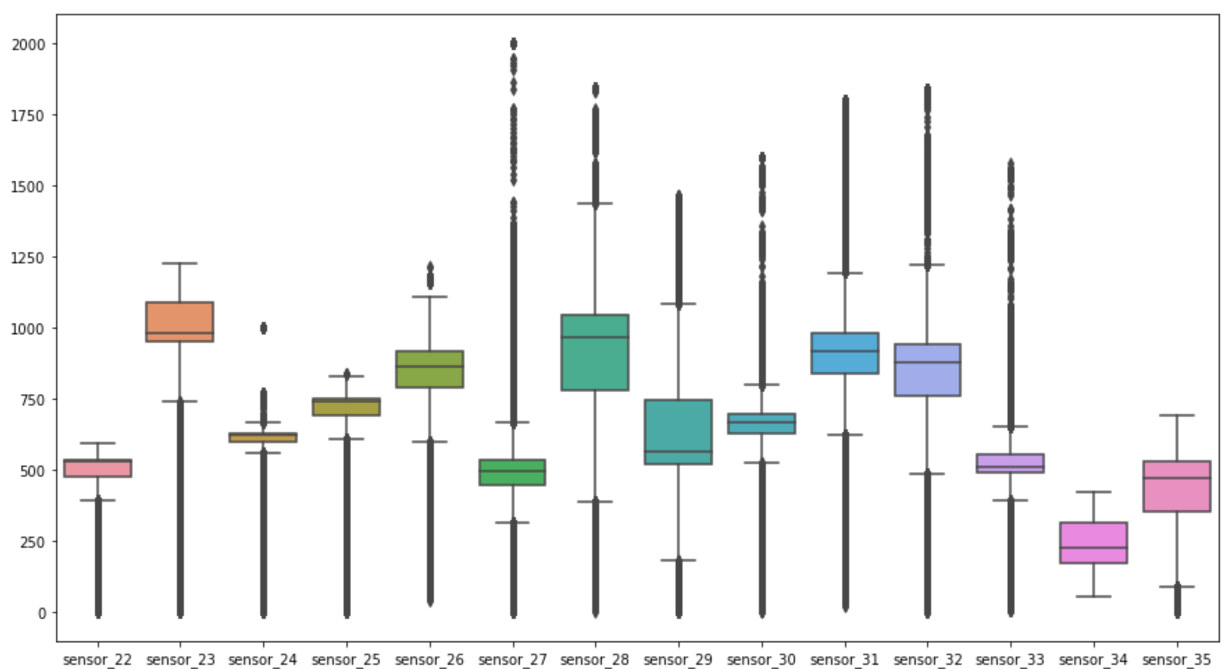
```
machine_status        0
dtype: int64
```

As shown by the previous cell their is a decent amount of nans. Sensor_15 will need to be removed as it has no data and will not benefit from being replaced by zeros. The others will need to have to be replaced by zeros or more likely the mean of the sensor data.

In [6]:
```
import seaborn as sns
from matplotlib import rcParams

# figure size in inches
rcParams['figure.figsize'] = 15,8.27
box_data = data_df.loc[:, 'sensor_22':'sensor_35']
print(box_data.head())
sns.boxplot(data = box_data)
```

```
   sensor_22  sensor_23  sensor_24  sensor_25  sensor_26  sensor_27  \
0   498.8926   975.9409   627.6740   741.7151   848.0708   429.0377   
1   498.8926   975.9409   627.6740   741.7151   848.0708   429.0377   
2   501.3617   982.7342   631.1326   740.8031   849.8997   454.2390   
3   499.0430   977.7520   625.4076   739.2722   847.7579   474.8731   
4   498.5383   979.5755   627.1830   737.6033   846.9182   408.8159   

   sensor_28  sensor_29  sensor_30  sensor_31  sensor_32  sensor_33  \
0   785.1935   684.9443   594.4445   682.8125   680.4416   433.7037   
1   785.1935   684.9443   594.4445   682.8125   680.4416   433.7037   
2   778.5734   715.6266   661.5740   721.8750   694.7721   441.2635   
3   779.5091   690.4011   686.1111   754.6875   683.3831   446.2493   
4   785.2307   704.6937   631.4814   766.1458   702.4431   433.9081   

   sensor_34  sensor_35  
0   171.9375   341.9039  
1   171.9375   341.9039  
2   169.9820   343.1955  
3   166.4987   343.9586  
4   164.7498   339.9630  
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe742c5a810>



The above figure shows off some of the sensors in a boxplot graph. As shown by the plots their is a decent amount of outliers within the data, but with the large amount of data there is definitely going to be a large amount of outliers as well.

In [7]:
```
#Check to see all of the categorical data bins.
data_df.machine_status.value_counts()
```

Out[7]:
```
NORMAL        205836
RECOVERING     14477
BROKEN             7
Name: machine_status, dtype: int64
```

In the final dataset, the broken will be merged with recovering as the goal is to check if the pump is in normal or not normal, recovering and broken, operation.