In [ ]:
```
""" Jonathan Zhang """
```

Checkpoint #1 Report:

Could not figure out how to make run.py run all my tests for Battery and Device Use.

Thought to compile an encompassing notebook to show you the data analysis we performed. Thank you!

Problem to investigate:

- "What parameters can help with battery health related issues?"

Based on the data/samples collected from the Battery DLL file, we found some parameters that we believe can influence battery health substantially.

- ACPI-BATTERY(1) - # of batteries detected
- ACPI-BATTERY(2) - Battery life %
- ACPI-BATTERY(6) - Chemistry
- ACPI-BATTERY(7) - Estimated runtime (s)
- ACPI-BATTERY(8) - Charge discharge rate
- ACPI-BATTERY(10) - Full charge capacity

In our subject domain, we utilize Intel's SDK (software development kit) for data collection. In doing so, we use their input libraries to extrapolate data from our computer's user wait time, processes, device and battery usage, specifically BATTERY and DEVICE USE.

In order to generate the data we needed to accomplish the tasks at hand, specifically performing data analysis on Battery and Device Use, we needed to use Intel's ESRV process which essentially gave us all the data we needed from our own computers. The output database file contained the timestamp at which the battery life was at a certain percentage, along with other various statistics and characteristics of our battery. We believe that in analyzing the parameters above, we can substantiate findings to see whether or not we can prolong battery life.

Checkpoint #1 Code for BATTERY:

Imports

In [36]:
```
import numpy as np
import pandas as pd
```

Data Preparation

In [52]:
```
# Data for BATTERY
data = pd.read_csv("../data/temp/counters_ull_time_data.csv")
data_2 = pd.read_csv("../data/temp/counters_double.csv")

# Data for DEVICE USE
du_data = pd.read_csv("../data/temp/du_counters_ull_time_data.csv")
du_data_2 = pd.read_csv("../data/temp/du_counters_string_time_data.csv")
```

COUNTERS_ULL_TIME_DATA.csv -- Data Analysis

In [41]:
```
# Data Demographic
    # 0 : ACDC
    # 1 : BATTERY COUNT
    # 2 : BATTERY LIFE
    # 3 : CAPABILITIES
    # 6 : ESTIMATED RUN TIME
    # 8 : THEORETICAL CAPACITY
    # 9 : FULL_CHARGE_CAPACITY
data['ID_INPUT'].value_counts()
```

Out[41]:
```
6    1403
2      38
9       1
8       1
3       1
1       1
0       1
Name: ID_INPUT, dtype: int64
```

In [67]:
```
# All the "VALUE(s)" presented when ID_INPUT is 2.
data.loc[(data['ID_INPUT'] == 2)]
```

Out[67]:

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_DATA
2	2020-11-07 00:51:22.229	2	63	0
34	2020-11-07 00:52:17.208	2	62	0
72	2020-11-07 00:53:43.188	2	61	0
111	2020-11-07 00:55:07.162	2	60	0
142	2020-11-07 00:56:17.142	2	59	0
177	2020-11-07 00:57:30.111	2	58	0
211	2020-11-07 00:58:43.090	2	57	0
256	2020-11-07 01:00:09.063	2	56	0
290	2020-11-07 01:01:29.039	2	55	0
331	2020-11-07 01:02:59.009	2	54	0
371	2020-11-07 01:04:21.989	2	53	0
410	2020-11-07 01:05:53.964	2	52	0
450	2020-11-07 01:07:20.934	2	51	0
488	2020-11-07 01:08:48.911	2	50	0
522	2020-11-07 01:10:18.875	2	49	0
564	2020-11-07 01:11:58.842	2	48	0
603	2020-11-07 01:13:20.816	2	47	0
642	2020-11-07 01:14:48.793	2	46	0
676	2020-11-07 01:16:21.756	2	45	0
716	2020-11-07 01:17:54.729	2	44	0
760	2020-11-07 01:19:23.699	2	43	0
800	2020-11-07 01:20:50.671	2	42	0
841	2020-11-07 01:22:20.645	2	41	0
875	2020-11-07 01:23:39.622	2	40	0
905	2020-11-07 01:25:01.590	2	39	0
958	2020-11-07 01:26:45.550	2	38	0
998	2020-11-07 01:28:19.511	2	37	0
1039	2020-11-07 01:29:50.481	2	36	0
1075	2020-11-07 01:31:18.448	2	35	0
1120	2020-11-07 01:32:54.426	2	34	0
1158	2020-11-07 01:34:23.391	2	33	0
1199	2020-11-07 01:35:55.368	2	32	0
1237	2020-11-07 01:37:27.351	2	31	0
1272	2020-11-07 01:38:43.321	2	30	0

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_DATA
1313	2020-11-07 01:40:08.295	2	29	0
1354	2020-11-07 01:41:40.268	2	28	0
1393	2020-11-07 01:43:14.235	2	27	0
1425	2020-11-07 01:44:36.204	2	26	0

In [43]:
```
# Max value when ID_INPUT is 2
data.loc[(data['ID_INPUT'] == 2)].max()
```

Out[43]:
```
MEASUREMENT_TIME    2020-11-07 01:44:36.204
ID_INPUT                                  2
VALUE                                    63
PRIVATE_DATA                              0
dtype: object
```

In [44]:
```
# Min value when ID_INPUT is 2
data.loc[(data['ID_INPUT'] == 2)].min()
```

Out[44]:
```
MEASUREMENT_TIME    2020-11-07 00:51:22.229
ID_INPUT                                  2
VALUE                                    26
PRIVATE_DATA                              0
dtype: object
```

In [78]:
```
# Average Battery Life % is 44.5%
data.loc[(data['ID_INPUT'] == 2)].mean()
```

Out[78]:
```
ID_INPUT         2.0
VALUE           44.5
PRIVATE_DATA     0.0
dtype: float64
```

COUNTERS_DOUBLE.csv -- Data Analysis

In [45]:
```
# Data Demographic
    # 7 : CHARGE_DISCHARGE_RATE
    # 10 : SYSTEM_POWER_IN_DC:WATTS:
data_2['ID_INPUT'].value_counts()
```

Out[45]:
```
7     1388
10     541
Name: ID_INPUT, dtype: int64
```

In [46]:
```
# All the "VALUE(s)" presented when ID_INPUT is 7.
data_2.loc[(data_2['ID_INPUT'] == 7)]
```

Out[46]:

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_DATA
0	2020-11-07 00:51:22.229	7	-19725.0	1
1	2020-11-07 00:51:23.228	7	-19319.0	1
2	2020-11-07 00:51:25.226	7	-19813.0	1
3	2020-11-07 00:51:26.225	7	-19846.0	1
4	2020-11-07 00:51:27.225	7	-19807.0	1
...	...	...	...	...
1922	2020-11-07 01:45:20.195	7	-19405.0	1
1924	2020-11-07 01:45:23.195	7	-19848.0	1
1925	2020-11-07 01:45:25.194	7	-19832.0	1
1927	2020-11-07 01:45:28.193	7	-19791.0	1
1928	2020-11-07 01:45:30.191	7	-19920.0	1

1388 rows × 4 columns

In [47]:
```
# All the "VALUE(s)" presented when ID_INPUT is 10.
data_2.loc[(data_2['ID_INPUT'] == 10)]
```

Out[47]:

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_DATA
5	2020-11-07 00:51:28.225	10	0.0	0
9	2020-11-07 00:51:34.223	10	0.0	0
13	2020-11-07 00:51:41.221	10	0.0	0
16	2020-11-07 00:51:46.217	10	0.0	0
21	2020-11-07 00:51:53.217	10	0.0	0
...	...	...	...	...
1914	2020-11-07 01:45:03.196	10	0.0	0
1918	2020-11-07 01:45:09.195	10	0.0	0
1920	2020-11-07 01:45:15.193	10	0.0	0
1923	2020-11-07 01:45:22.194	10	0.0	0
1926	2020-11-07 01:45:28.193	10	0.0	0

541 rows × 4 columns

In [48]:
```
# Max value when ID_INPUT is 7
    # Max value : -15621
data_2.loc[(data_2['ID_INPUT'] == 7)].max()
```

Out[48]:
```
MEASUREMENT_TIME    2020-11-07 01:45:30.191
ID_INPUT                                  7
VALUE                                 -15621
PRIVATE_DATA                               1
dtype: object
```

In [49]:
```
# Min value when ID_INPUT is 7
    # Min value : -25405
data_2.loc[(data_2['ID_INPUT'] == 7)].min()
```

Out[49]:
```
MEASUREMENT_TIME    2020-11-07 00:51:22.229
ID_INPUT                                  7
VALUE                                 -25405
PRIVATE_DATA                               1
dtype: object
```

In [50]:
```
# Max value when ID_INPUT is 10
    # Max value : 288.185
data_2.loc[(data_2['ID_INPUT'] == 10)].max()
```

Out[50]:
```
MEASUREMENT_TIME    2020-11-07 01:45:28.193
ID_INPUT                                 10
VALUE                                288.185
PRIVATE_DATA                               0
dtype: object
```

In [51]:
```
# Min value when ID_INPUT is 10
    # Min value : 0
data_2.loc[(data_2['ID_INPUT'] == 10)].min()
```

Out[51]:
```
MEASUREMENT_TIME    2020-11-07 00:51:28.225
ID_INPUT                                 10
VALUE                                     0
PRIVATE_DATA                              0
dtype: object
```

Checkpoint #1 Code for DEVICE USE:

In [70]:
```
# Real Numbers for VALUES
du_data.head(10)
```

Out[70]:

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_DATA
0	2020-11-07 01:50:13.775	3	4	1
1	2020-11-07 01:50:13.775	3	4	2
2	2020-11-07 01:50:13.775	3	4	3
3	2020-11-07 01:50:13.775	3	4	4
4	2020-11-07 01:50:13.775	3	4	5
5	2020-11-07 01:50:13.775	3	4	6
6	2020-11-07 01:50:13.775	3	4	7
7	2020-11-07 01:50:13.775	3	4	8
8	2020-11-07 01:50:13.775	3	4	9
9	2020-11-07 01:50:13.775	3	4	10

In [71]:
```
# Strings for VALUES
du_data_2.head(10)
```

Out[71]:

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_
0	2020-11-07 01:50:13.775	0	GUID_DEVINTERFACE_DISPLAY_ADAPTER	
1	2020-11-07 01:50:13.775	1	ROOT\BasicDisplay	
2	2020-11-07 01:50:13.775	2	Microsoft Basic Display Driver	
3	2020-11-07 01:50:13.775	0	GUID_DEVINTERFACE_DISPLAY_ADAPTER	
4	2020-11-07 01:50:13.775	1	PCI\VEN_8086&DEV_8A52&SUBSYS_385217AA&REV_07	
5	2020-11-07 01:50:13.775	2	Intel(R) Iris(R) Plus Graphics	
6	2020-11-07 01:50:13.775	0	GUID_DEVINTERFACE_MONITOR	
7	2020-11-07 01:50:13.775	1	MONITOR\BOE06F2	
8	2020-11-07 01:50:13.775	2	Generic PnP Monitor	
9	2020-11-07 01:50:13.775	0	GUID_DISPLAY_DEVICE_ARRIVAL	

In [74]:
```
# Device Use Data Demographic
    # 3 : Device Status
du_data["ID_INPUT"].value_counts()
```

Out[74]:
```
3    39
Name: ID_INPUT, dtype: int64
```

In [75]:
```
# Device Use Data Demographic
    # 0 : Device Guide
    # 1 : Device HW Name
    # 2 : Device Friendly Name
du_data_2['ID_INPUT'].value_counts()
```

Out[75]:
```
2    39
1    39
0    39
Name: ID_INPUT, dtype: int64
```