Part-I

Objective: Analysing IoT Data with Spark Sql

Author: Harshit Mehta

In [1]:
```
import numpy as np # linear algebra
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

In [2]:
```
import pyspark
from pyspark.sql import SparkSession
from pyspark import SparkConf
from pyspark import SparkContext
```

In [3]:
```
conf=pyspark.SparkConf().setAppName('SparkApp').setMaster('local')
sc=pyspark.SparkContext(conf=conf)
spark=SparkSession(sc)
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setL
ogLevel(newLevel).
23/07/16 22:24:04 WARN NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
23/07/16 22:24:10 WARN Utils: Service 'SparkUI' could not bind on port 404
0. Attempting port 4041.
```

Task 1

Read the data into a Dataframe

In [4]:
```
# Let's read the iot_devices.json into a dataframe
df = spark.read.json('iot_devices.json')
```

In [5]:
```
# Let's check the first row
df.show(n=1, vertical=True)
```

```
-RECORD 0----------------------------
 battery_level | 8
 c02_level     | 868
 cca2          | US
 cca3          | USA
 cn            | United States
 device_id     | 1
 device_name   | meter-gauge-1xbYRYcj
 humidity      | 51
 ip            | 68.161.225.1
 latitude      | 38.0
 lcd           | green
 longitude     | -97.0
 scale         | Celsius
 temp          | 34
 timestamp     | 1458444054093
only showing top 1 row
```

In [6]:
```
# Let's check the schema of the dataframe
df.printSchema()
```

```
root
 |-- battery_level: long (nullable = true)
 |-- c02_level: long (nullable = true)
 |-- cca2: string (nullable = true)
 |-- cca3: string (nullable = true)
 |-- cn: string (nullable = true)
 |-- device_id: long (nullable = true)
 |-- device_name: string (nullable = true)
 |-- humidity: long (nullable = true)
 |-- ip: string (nullable = true)
 |-- latitude: double (nullable = true)
 |-- lcd: string (nullable = true)
 |-- longitude: double (nullable = true)
 |-- scale: string (nullable = true)
 |-- temp: long (nullable = true)
 |-- timestamp: long (nullable = true)
```

In [7]:
```
# Let's get a statistical summary of the dataframe
df.describe().show(vertical=True)
```

```
[Stage 2:>                                                          (0 +
1) / 1]
```

```
-RECORD 0-----------------------------
 summary       | count
 battery_level | 198164
 c02_level     | 198164
 cca2          | 198164
 cca3          | 198164
 cn            | 198164
 device_id     | 198164
 device_name   | 198164
 humidity      | 198164
 ip            | 198164
 latitude      | 198164
 lcd           | 198164
 longitude     | 198164
 scale         | 198164
 temp          | 198164
 timestamp     | 198164
-RECORD 1-----------------------------
 summary       | mean
 battery_level | 4.4997678690377665
 c02_level     | 1199.7639429967098
 cca2          | null
 cca3          | null
 cn            | null
 device_id     | 99082.5
 device_name   | null
 humidity      | 61.99212773258513
 ip            | null
 latitude      | 36.521156062675466
 lcd           | null
 longitude     | -0.6459595082859193
 scale         | null
 temp          | 22.012787388223895
 timestamp     | 1.458444058246237...
-RECORD 2-----------------------------
 summary       | stddev
 battery_level | 2.8733916884106177
 c02_level     | 231.06002562900682
 cca2          | null
 cca3          | null
 cn            | null
 device_id     | 57205.1637092317
 device_name   | null
 humidity      | 21.67231306231426
 ip            | null
 latitude      | 17.907740712889645
 lcd           | null
 longitude     | 88.7275821792014
 scale         | null
 temp          | 7.209848253887028
 timestamp     | 1708.2257967415815
-RECORD 3-----------------------------
 summary       | min
 battery_level | 0
 c02_level     | 800
 cca2          | AD
 cca3          | ABW
 cn            |
 device_id     | 1
 device_name   | device-mac-100005...
```

```
 humidity      | 25
 ip            | 108.57.128.215
 latitude      | -51.75
 lcd           | green
 longitude     | -175.0
 scale         | Celsius
 temp          | 10
 timestamp     | 1458444054093
-RECORD 4-----------------------------
 summary       | max
 battery_level | 9
 c02_level     | 1599
 cca2          | ZW
 cca3          | ZWE
 cn            | Åland
 device_id     | 198164
 device_name   | therm-stick-99995...
 humidity      | 99
 ip            | 99.64.14.90
 latitude      | 72.0
 lcd           | yellow
 longitude     | 178.42
 scale         | Celsius
 temp          | 34
 timestamp     | 1458444061098
```

Task 2

Convert the Dataframe into a temporary view called iot

In [8]:
```
# Let's register the dataframe as a view to run sql on
df.createOrReplaceTempView("iot")
```

Task 3

Count how many devices are there from each country and display the output

In [9]:
```
result1 = spark.sql("SELECT cn as country, count(*) as count FROM iot GRO
result1.show()
```

```
+----------------+-----+
|         country|count|
+----------------+-----+
|    United States|68545|
|            China|14455|
|            Japan|12100|
|Republic of Korea|11879|
|          Germany| 7942|
|   United Kingdom| 6486|
|           Canada| 6041|
|           Russia| 5989|
|           France| 5305|
|           Brazil| 3224|
|        Australia| 3119|
|            Italy| 2915|
|           Sweden| 2880|
|           Poland| 2744|
|      Netherlands| 2488|
|            Spain| 2310|
|           Taiwan| 2128|
|            India| 1867|
|                 | 1810|
|   Czech Republic| 1507|
+----------------+-----+
only showing top 20 rows
```

Task 4

Display all the countries whose carbon dioxide level is more than 1400. Sort the output in descending order.

Calrification:

Since the wording of the question does not specify it, I have **ranked** the question in **3 different ways** as follows:

- result2_0 shows countries with C02_levels greater than 1400 AND countries ranked by maximum c02_levels
- result2_1 shows countries with C02_levels greater than 1400 AND countries ranked by mean C02_levels
- result2_2 shows countries with C02_levels greater than 1400 AND countries ranked by total number of devices in the country having C02 levels > 1400

So there are correspondingly 3 different queries, with following names:

- result2_0
- result2_1
- result2_2

What is different in these queries is the ranking methodology.

In [10]:
```
total_countries = spark.sql("SELECT count(DISTINCT cca3) as Total_Countri
total_countries.show()
```

```
+---------------+
|Total_Countries|
+---------------+
|            205|
+---------------+
```

Thus there are a total of **205** countries in the dataset

Countries with C02_levels greater than 1400 AND countries ranked by total number of devices in the country having C02 levels > 1400:

In [11]:
```
# Countries with C02_levels greater than 1400 AND countries ranked by tot
result2_2 = spark.sql("SELECT cca3 as country, count(c02_level) as total_
result2_2.show()
```

```
+-------+-------------+
|country|total_devices|
+-------+-------------+
|    USA|        17489|
|    CHN|         3616|
|    KOR|         2942|
|    JPN|         2935|
|    DEU|         1966|
|    GBR|         1660|
|    CAN|         1564|
|    RUS|         1508|
|    FRA|         1353|
|    BRA|          856|
|    AUS|          769|
|    SWE|          724|
|    ITA|          713|
|    POL|          664|
|    NLD|          646|
|    ESP|          586|
|    TWN|          542|
|    IND|          446|
|    NOR|          399|
|    UKR|          373|
+-------+-------------+
only showing top 20 rows
```

Countries with C02_levels greater than 1400 AND countries ranked by maximum C02_levels:

In [12]:
```
# Countries with max c02_levels greater than 1400 AND countries ranked by
result2_0 = spark.sql("SELECT cca3 as country, max(c02_level) as max_c02_
result2_0.show()
```

```
+-------+-------------+
|country|max_c02_level|
+-------+-------------+
|    NLD|         1599|
|    DEU|         1599|
|    TUR|         1599|
|    PER|         1599|
|    USA|         1599|
|    BRA|         1599|
|    VNM|         1599|
|    BOL|         1599|
|    KOR|         1599|
|    UKR|         1599|
|    ZAF|         1599|
|    ARE|         1599|
|    TWN|         1599|
|    POL|         1599|
|    MYS|         1599|
|    KNA|         1599|
|    CAN|         1599|
|    HUN|         1599|
|    RUS|         1599|
|    NOR|         1599|
+-------+-------------+
only showing top 20 rows
```

In [13]:
```
# Number of countries with max c02_levels greater than 1400
summary2_0 = spark.sql("SELECT count(*) as count FROM (SELECT cca3 as cou
summary2_0.show()
```

```
+-----+
|count|
+-----+
|  186|
+-----+
```

Thus there are 186 countries with maximum c02 levels detected greater than 1400

In [14]:
```
# Countries with max c02_levels less than 1400
clean_countries = spark.sql("SELECT DISTINCT cca3 as country FROM iot WHE
clean_countries.show()
```

```
+-------+
|country|
+-------+
|    TCA|
|    VAT|
|    GUY|
|    TON|
|    SYC|
|    COG|
|    MAF|
|    SUR|
|    MWI|
|    SLE|
|    GRL|
|    ASM|
|    AIA|
|    WSM|
|    MTQ|
|    GRD|
|    GIN|
|    COK|
|    BEN|
+-------+
```

Above 19 countries is where max_c02_level less than 1400

Countries with C02_levels greater than 1400 AND countries ranked by mean C02_levels:

(Note: While calculating the mean, only those devices whose C02 levels are greater than 1400 have been considered while calculating the mean)

In [15]:
```
# Countries with C02_levels greater than 1400 AND countries ranked by mea
result2_1 = spark.sql("SELECT cca3 as country, round(mean(c02_level),2) a
result2_1.show()
```

```
+-------+--------------+
|country|mean_c02_level|
+-------+--------------+
|    VCT|        1593.5|
|    SLB|        1588.0|
|    FSM|        1573.0|
|    RWA|        1560.5|
|    IOT|        1560.0|
|    ABW|        1559.0|
|    IMN|        1548.0|
|    GMB|        1544.5|
|    ALA|        1542.4|
|    AZE|       1538.38|
|    FRO|       1537.67|
|    LSO|        1537.5|
|    NCL|        1537.0|
|    AFG|        1536.0|
|    HTI|        1535.4|
|    CUB|        1534.8|
|    PRY|       1533.17|
|    VUT|        1532.0|
|    KNA|       1530.57|
|    MNG|       1529.78|
+-------+--------------+
only showing top 20 rows
```

In [16]:
```
# Countries with mean c02_levels less than 1400
clean_countries2_1 = spark.sql("SELECT DISTINCT cca3 as country FROM iot
clean_countries2_1.show()
```

```
+-------+
|country|
+-------+
|    TCA|
|    VAT|
|    GUY|
|    TON|
|    SYC|
|    COG|
|    MAF|
|    SUR|
|    MWI|
|    SLE|
|    GRL|
|    ASM|
|    AIA|
|    WSM|
|    MTQ|
|    GRD|
|    GIN|
|    COK|
|    BEN|
+-------+
```

The same 19 countries

Task 5

Select all countries' devices with high-levels of C02 and group by cca3 and order by device_ids (Hint: For high CO2 level, the LCD status will be RED).

In [17]:
```
total_lcd_states = spark.sql("SELECT DISTINCT lcd FROM iot")
total_lcd_states.show()
```

```
+------+
|   lcd|
+------+
| green|
|yellow|
|   red|
+------+
```

Interpretation: Display the total number of devices with high levels of CO2 across each country and display the count in its ascending order.

In [18]:
```
# ascending order
# Display the total number of devices with high levels of CO2 across each
result3 = spark.sql("SELECT cca3 as country, count(*) as total_devices FR
result3.show()
```

```
[Stage 44:>                                                         (0 +
1) / 1]
+-------+-------------+
|country|total_devices|
+-------+-------------+
|    TJK|            1|
|    FLK|            1|
|    REU|            1|
|    GIB|            1|
|    CIV|            1|
|    LBR|            1|
|    MHL|            1|
|    SLB|            1|
|    GAB|            1|
|    TKM|            1|
|    IOT|            1|
|    TLS|            1|
|    FSM|            1|
|    AND|            1|
|    GGY|            1|
|    YEM|            2|
|    LCA|            2|
|    VUT|            2|
|    ZMB|            2|
|    VCT|            2|
+-------+-------------+
only showing top 20 rows
```

In [19]:
```
# descending order
# Display the total number of devices with high levels of CO2 across each
```

```
result3 = spark.sql("SELECT cca3 as country, count(*) as total_devices FR
result3.show()
```

```
+-------+-------------+
|country|total_devices|
+-------+-------------+
|    USA|        17489|
|    CHN|         3616|
|    KOR|         2942|
|    JPN|         2935|
|    DEU|         1966|
|    GBR|         1660|
|    CAN|         1564|
|    RUS|         1508|
|    FRA|         1353|
|    BRA|          856|
|    AUS|          769|
|    SWE|          724|
|    ITA|          713|
|    POL|          664|
|    NLD|          646|
|    ESP|          586|
|    TWN|          542|
|    IND|          446|
|    NOR|          399|
|    UKR|          373|
+-------+-------------+
only showing top 20 rows
```

Task 6

Find out all devices in countries whose batteries need replacements

ASSUMPTION: I have assumed that devices with battery_level = 0 need replacement

In [20]:
```
result4 = spark.sql("SELECT cca3 as country, device_id FROM iot WHERE bat
result4.show()
```

```
+-------+---------+
|country|device_id|
+-------+---------+
|    AFG|   167828|
|    AFG|   162516|
|    AFG|   122629|
|    AGO|    65166|
|    AIA|   146980|
|    AIA|   122109|
|    ALA|   124665|
|    ALA|   197516|
|    ALA|    96404|
|    ALB|   138223|
|    ALB|    82389|
|    ALB|    12532|
|    ARE|    74301|
|    ARE|    81355|
|    ARE|    77008|
|    ARE|   107431|
|    ARE|   123314|
|    ARE|    62126|
|    ARE|   110925|
|    ARE|    32595|
+-------+---------+
only showing top 20 rows
```

Summary:

Task 1:

df = spark.read.json('iot_devices.json')

Task 2:

df.createOrReplaceTempView("iot")

Task 3:

result1 = spark.sql("**SELECT cn as country, count(*) as count FROM iot GROUP BY cn ORDER BY count desc**") result1.show()

Task 4:

Countries with C02_levels greater than 1400 AND countries ranked by total number of devices in the country having C02 levels > 1400:

result2_2 = spark.sql("**SELECT cca3 as country, count(c02_level) as total_devices FROM iot WHERE c02_level > 1400 GROUP BY cca3 ORDER BY total_devices desc**")

Countries with C02_levels greater than 1400 AND countries ranked by maximum c02_levels :

result2_0 = spark.sql("**SELECT cca3 as country, max(c02_level) as max_c02_level FROM iot WHERE c02_level > 1400 GROUP BY cca3 ORDER BY**

max_c02_level desc")

Countries with C02_levels greater than 1400 AND countries ranked by mean C02_levels :

result2_1 = spark.sql("**SELECT cca3 as country, round(mean(c02_level),2) as mean_c02_level FROM iot WHERE c02_level > 1400 GROUP BY cca3 ORDER BY mean_c02_level desc**")

Task 5:

Ascending order :
result3 = spark.sql("**SELECT cca3 as country, count(*) as total_devices FROM iot WHERE lcd = 'red' GROUP BY cca3 ORDER BY total_devices asc**")

Descending order :
result3 = spark.sql("**SELECT cca3 as country, count(*) as total_devices FROM iot WHERE lcd = 'red' GROUP BY cca3 ORDER BY total_devices desc**")

Task 6:

result4 = spark.sql("**SELECT cca3 as country, device_id FROM iot WHERE battery_level = 0 ORDER BY cca3**")