## Assignment 5\_Aparna Suresh

### Initial Setup

**Start Docker:**   
1.Open docker app

A screenshot of a computer

Description automatically generated

In the cmd as admin:

2.docker start db43bc55d3fe

3.docker exec -it db43bc55d3fe /bin/bash

4.Copy the input file to docker in cmd:  
docker cp iot\_devices.json hadoop-cluster:/input/

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Start pyspark**, with Hive support

5.pyspark --conf spark.sql.catalogImplementation=hive

A screenshot of a computer program

Description automatically generated

### Question:Read the iot\_devices.json file.

#### Answer:

iot\_devices = spark.read.json("input/iot\_devices.json")

A screenshot of a computer program

Description automatically generated

### Question: Create a temporary view for it.

Answer:

iot\_devices.createOrReplaceTempView("iot\_devices\_view")

A screenshot of a computer program

Description automatically generated

### Question: What percent of devices are showing red on the LCD? Use only SQL (not DataFrame) to come up with the answer to this question.

#### Answer:

spark.sql("""SELECT (COUNT(CASE WHEN lcd = 'red' THEN 1 END) \* 100.0 / COUNT(\*)) AS red\_percentage FROM iot\_devices\_view""").show()

### Write to text file:

red\_percentage= spark.sql("""SELECT (COUNT(CASE WHEN lcd = 'red' THEN 1 END) \* 100.0 / COUNT(\*)) AS red\_percentage FROM iot\_devices\_view""")

Text file didn’t support decimal so convert to string:

from pyspark.sql.functions import col

red\_percentage\_str = red\_percentage.withColumn("red\_percentage", col("red\_percentage").cast("string"))

red\_percentage\_str.coalesce(1).write \

.mode("overwrite") \

.format("text") \

.option("header", "false") \

.save("file:/output/percentage\_devices\_red.txt")

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

### Question: Among those "red" devices, do a **SQL query** that shows the count breakdown by country (sorted descending). What is the country with the most red devices and how many of them does it have?(REDO)

#### Answer:

spark.sql("""

SELECT cn AS country, COUNT(\*) AS red\_device\_count

FROM iot\_devices\_view

WHERE lcd = 'red'

GROUP BY cn

ORDER BY red\_device\_count DESC

LIMIT 1

""").show()

A screenshot of a computer

Description automatically generated

All countries with red device count:

spark.sql("""

SELECT cn AS country, COUNT(\*) AS red\_device\_count

FROM iot\_devices\_view

WHERE lcd = 'red'

GROUP BY cn

ORDER BY red\_device\_count DESC

""").show()

A screenshot of a computer

Description automatically generated

### Write it to a file:

result = spark.sql("""SELECT cn AS country, COUNT(\*) AS red\_device\_count FROM iot\_devices\_view WHERE lcd = 'red' GROUP BY cn ORDER BY red\_device\_count DESC LIMIT 1""")

# Collect the result to the driver

result\_data = result.collect()

# Open a text file for writing

with open("output/top\_country\_red\_device.txt", "w") as f:

for row in result\_data:

f.write(f"{row['country']}: {row['red\_device\_count']}\n")

A screenshot of a computer

Description automatically generated

Write all country and red device count to file:

result\_1 = spark.sql("""

SELECT cn AS country, COUNT(\*) AS red\_device\_count

FROM iot\_devices\_view

WHERE lcd = 'red'

GROUP BY cn

ORDER BY red\_device\_count DESC

""")

# Collect the result to the driver

result1\_data = result\_1.collect()

# Open a text file for writing

with open("output/country\_red\_device.txt", "w") as f:

for row in result\_data:

f.write(f"{row['country']}: {row['red\_device\_count']}\n")

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Question: Create a managed table for this query

Answer:

spark.sql("""CREATE TABLE TOP\_COUNTRY\_RED\_DEVICE AS SELECT cn AS country, COUNT(\*) AS red\_device\_count FROM iot\_devices\_view WHERE lcd = 'red' GROUP BY cn ORDER BY red\_device\_count DESC LIMIT 1""")

spark.sql("SHOW TABLES").show()

spark.sql("SELECT \* FROM TOP\_COUNTRY\_RED\_DEVICE").show()

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

spark.sql("""CREATE TABLE country\_red\_device AS SELECT cn AS country, COUNT(\*) AS red\_device\_count FROM iot\_devices\_view WHERE lcd = 'red' GROUP BY cn ORDER BY red\_device\_count DESC""")

spark.sql("SHOW TABLES").show()

spark.sql("SELECT \* FROM country\_red\_device").show()

A screenshot of a computer

Description automatically generated

### 6.Drop table:

spark.sql("DROP TABLE IF EXISTS country\_red\_device")

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Copy files to local:

docker cp hadoop-cluster:/output/ "A:/SJSU/Sem-2/DATA-228-Big Data/Assignment\_5/docker\_output"

**7. Check whether the original file was deleted:**

1. The iot\_devices.json file still exists in its original location.
2. **Explanation:**
   * Dropping a managed table deletes both the table metadata and the data file because the table is under Spark’s control.
   * However, since the iot\_devices.json file was initially read as an external file (not created by Spark), dropping the managed table does not delete the original file. The file remains untouched unless explicitly deleted by a Spark operation.