

Case-Study Day 21.2(Sensor Data)

Objective 1:

1. Load HVAC.csv file into temporary table
2. Add a new column, tempchange - set to 1, if there is a change of greater than +/-5 between actual and target temperature

Ans:

Note: Program files are properly documented for a detailed description of each instruction used within the program.

1. ScreenShot:

The screenshot shows a Scala IDE with a project named 'sparktest-build'. The code in the editor is as follows:

```
61 println("Dataframe Registered as table !")
62
63 //Add a new column, tempchange - set to 1, if
```

The output window shows the following table:

Date	Time	TargetTemp	ActualTemp	System	SystemAge	BuildingId
6/1/13	0:00:01	66	58	13	20	4
6/2/13	1:00:01	69	68	3	20	17
6/3/13	2:00:01	70	73	17	20	18
6/4/13	3:00:01	67	63	2	23	15
6/5/13	4:00:01	68	74	16	9	3
6/6/13	5:00:01	67	56	13	28	4
6/7/13	6:00:01	70	58	12	24	2
6/8/13	7:00:01	70	73	20	26	16
6/9/13	8:00:01	66	69	16	9	9
6/10/13	9:00:01	65	57	6	5	12
6/11/13	10:00:01	67	70	10	17	15
6/12/13	11:00:01	69	62	2	11	7
6/13/13	12:00:01	69	73	14	2	15
6/14/13	13:00:01	65	61	3	2	6
6/15/13	14:00:01	67	59	19	22	20
6/16/13	15:00:01	65	56	19	11	8
6/17/13	16:00:01	67	57	15	7	6
6/18/13	17:00:01	66	57	12	5	13
6/19/13	18:00:01	69	58	8	22	4
6/20/13	19:00:01	67	55	17	5	7

only showing top 20 rows

Dataframe Registered as table !

2. ScreenShot:

spark-warehouse

src

main

scala

core

SparkSQLUseCase1

SparkSQLUseCase1

```
64 //Added a new column based upon a condition
65 al hvac1 = spark.sql( sqlText = "select *,IF((targettemp - actualtemp) > 5, '1', IF((targettemp - actualtemp) < -5, '1', 0)) AS tempchange"
66
```

Date	Time	TargetTemp	ActualTemp	System	SystemAge	BuildingId	tempchange
6/1/13	0:00:01	66	58	13	20	4	1
6/2/13	1:00:01	69	68	3	20	17	0
6/3/13	2:00:01	70	73	17	20	18	0
6/4/13	3:00:01	67	63	2	23	15	0
6/5/13	4:00:01	68	74	16	9	3	1
6/6/13	5:00:01	67	56	13	28	4	1
6/7/13	6:00:01	70	58	12	24	2	1
6/8/13	7:00:01	70	73	20	26	16	0
6/9/13	8:00:01	66	69	16	9	9	0
6/10/13	9:00:01	65	57	6	5	12	1
6/11/13	10:00:01	67	70	10	17	15	0
6/12/13	11:00:01	69	62	2	11	7	1
6/13/13	12:00:01	69	73	14	2	15	0
6/14/13	13:00:01	65	61	3	2	6	0
6/15/13	14:00:01	67	59	19	22	20	1
6/16/13	15:00:01	65	56	19	11	8	1
6/17/13	16:00:01	67	57	15	7	6	1
6/18/13	17:00:01	66	57	12	5	13	1
6/19/13	18:00:01	69	58	8	22	4	1
6/20/13	19:00:01	67	55	17	5	7	1

only showing top 20 rows

Data Frame Registered as HVAC1 table !

Objective 2:

1. Load building.csv file into temporary table

Ans:

Note: Program files are properly documented for a detailed description of each instruction used within the program.

ScreenShot:

The screenshot shows an IDE with a project named 'sparktest-build'. The file explorer on the left shows the directory structure: 'src' > 'main' > 'scala' > 'core'. The code editor displays the following code:

```
99 build.createOrReplaceTempView( viewName = "building")
100
101 println("Buildings data registered as building table")
102
```

The code is part of a file named 'SparkSQLUseCase1' with a main method that takes an array of strings as an argument. Below the code editor, a data table is displayed, showing the contents of the 'building' table. The table has columns: buildingid, buildmgr, buildAge, hvacproduct, and Country. The data is as follows:

buildingid	buildmgr	buildAge	hvacproduct	Country
1	M1	25	AC1000	USA
2	M2	27	FN39TG	France
3	M3	28	JDNS77	Brazil
4	M4	17	GG1919	Finland
5	M5	3	ACMAX22	Hong Kong
6	M6	9	AC1000	Singapore
7	M7	13	FN39TG	South Africa
8	M8	25	JDNS77	Australia
9	M9	11	GG1919	Mexico
10	M10	23	ACMAX22	China
11	M11	14	AC1000	Belgium
12	M12	26	FN39TG	Finland
13	M13	25	JDNS77	Saudi Arabia
14	M14	17	GG1919	Germany
15	M15	19	ACMAX22	Israel
16	M16	23	AC1000	Turkey
17	M17	11	FN39TG	Egypt
18	M18	25	JDNS77	Indonesia
19	M19	14	GG1919	Canada
20	M20	19	ACMAX22	Argentina

Below the table, the text 'Buildings data registered as building table' is displayed, which is a confirmation message from the program.

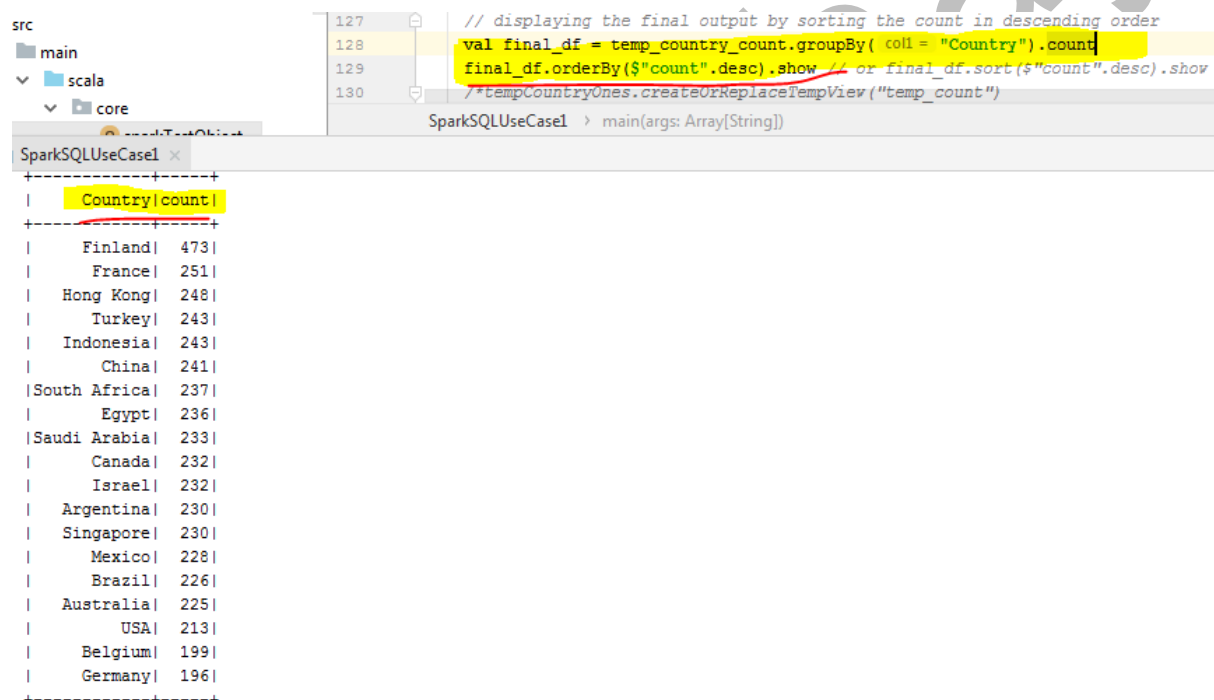
Objective 3:

1. Filter the rows where tempchange is 1 and count the number of occurrences for each country.

Ans:

Note: Program files are properly documented for a detailed description of each instruction used within the program.

ScreenShot:



```
src
├── main
│   └── scala
│       └── core
│           └── SparkSQLUseCase1.scala
└── TestObject

127 // displaying the final output by sorting the count in descending order
128 val final_df = temp_country_count.groupBy( col1 = "Country").count
129 final_df.orderBy($"count".desc).show // or final_df.sort($"count".desc).show
130 /*tempCountryOnes.createOrReplaceTempView("temp_count")

SparkSQLUseCase1 > main(args: Array[String])

+-----+-----+
| Country | count |
+-----+-----+
| Finland | 473   |
| France  | 251   |
| Hong Kong | 248   |
| Turkey  | 243   |
| Indonesia | 243   |
| China   | 241   |
| South Africa | 237   |
| Egypt   | 236   |
| Saudi Arabia | 233   |
| Canada  | 232   |
| Israel  | 232   |
| Argentina | 230   |
| Singapore | 230   |
| Mexico  | 228   |
| Brazil  | 226   |
| Australia | 225   |
| USA     | 213   |
| Belgium | 199   |
| Germany | 196   |
+-----+-----+
```

End
