UE19CS322 Big Data Assignment 3

Analysis of Earth Surface Temperature using Spark

This is the third assignment for the UE19CS322 Big Data Course at PES University. The assignment consists of 2 tasks and focuses on using Spark and manipulating dataframes to analyse and obtain insights from Earth's surface temperature readings.

The files required for the assignment can be found here.

Assignment Objectives and Outcomes

1. This assignment will help students get familiar with Spark and DataFrames
2. At the end of this assignment, the student will be able to write and debug code to work with DataFrames and manipulate them using Spark

Ethical practices

Please submit original code only. You can discuss your approach with your friends but you must write original code. All solutions must be submitted through the portal. **We will perform a plagiarism check on the code and you will be penalised if your code is found to be plagiarised**.

The Dataset

The dataset consists of readings of the Earth's surface temperatures and was compiled and put together by Berkeley Earth, an organization which is focused on environmental data science and records data on Earth's climate.

There are **2 dataframes**, and you will be working with the following attributes in each of them respectively.

City.csv

This DataFrame contains **surface temperature readings captured everyday for every city** in the world from 1750 to present.

Attribute	Type	Description
dt	String	Date of the Record
City	String	Name of the city
Country	String	Name of the country
AverageTemperature	Float	Average Temperature of the city on that Date

Global.csv

This DataFrame contains **worldwide surface temperature** readings captured everyday from 1750 to present.

Attribute	Type	Description
dt	String	Date of the Record
LandAverageTemperature	Float	Global Land Average Temperature on that Date

Software/Languages to be used:

1. Python `3.8.x`
2. Spark `v3.1.2` only

Marks

Task 1: 2 marks

Task 2: 2 marks

Report: 1 mark

Tasks Overview:

1. Create `task1.py` and `task2.py` for Task 1 and Task 2 respectively
2. Run your code on the sample dataset until you get the right answer
3. Submit the files to the portal
4. Submit one page report based on the template and answer the questions on the report

Submission Link

Portal for Big Data Assignment Submissions

Submission Deadline

8th November, 11:59 PM

Submission Guidelines

You will need to make the following changes to your `task1.py` and `task2.py` scripts to run them on the portal

1. Make sure to always **fetch** the available `Spark Context` instead of creating a new one. This will prevent any errors while attempting to create a new `Spark Context` to connect to the Spark Cluster.

```
spark_context = SparkContext.getOrCreate()
```

2. Convert line breaks in DOS format to Unix format (**this is necessary if you are coding on Windows** - your code will not run on our portal otherwise)

```
dos2unix task1.py task2.py
```

Task Specifications

The following sample splits **(containing only the required columns for both Tasks)** will be used to explain the examples for each Task

City.csv

dt	AverageTemperature	City	Country
1876-02-01	20	Bangalore	India
1862-04-01	32	Bangalore	India
1876-02-01	28	Indore	India
1862-04-01	34	Kolkata	India
1856-01-01	32	Kolkata	India
1856-01-01	33	Delhi	India

dt	AverageTemperature	City	Country
1876-02-01	27	Frankfurt	Germany
1862-04-01	29	Frankfurt	Germany
1856-01-01	26	Frankfurt	Germany

Global.csv

dt	LandAverageTemperature
1876-02-01	20
1862-04-01	30
1856-01-01	25

Task 1

Problem Statement

Find the number of times where a city's average temperature on a day turned out to be higher than the city's average temperature throughout the dataset for a given country.

Description

Use the `city.csv` DataFrame to find the number of times a city's average temperature turned out to be higher than the city's overall average temperature for a given `country` and display each city along with its count on a newline.

Comments

1. The `country` and the path to the `city.csv` DataFrame will be provided as the two command line arguments for the Task.
2. You are *expected* to use as many Spark transformations as possible without using custom code snippets (loops and data structures to manipulate data) since these will speed up your implementation
3. **Important**: Never load the entire dataset into your memory!

Input Format

The input to the Task consists of two command line arguments in the following order:

`country path_to_city.csv`

Output Format

Display each city in that `country` with its corresponding count in sorted order of cities. The values are `\t` separated and newline delimited. Do not print the city if it does not have any instances where the required condition has been satisfied.

Example

Assuming the value of the command line argument `country` to be **India**, we obtain the average temperatures of each city in the given country as `26` for Bangalore, `33` for Kolkata, `28` for Indore, and `33` for Delhi.

There are exactly **two examples** in the given split where the average temperature of a city in the given country on any given date is higher than the city's average temperature throughout the dataset:

1. Bangalore (occurred on 1862-04-01 with average temperature of 32)
2. Kolkata (occurred on 1981-11-01 with average temperature of 34)

Hence, the output for this Task will be as follows.

```
Bangalore   1
Kolkata 1
```

Task 2

Problem Statement

Find the number of times where a country's maximum average temperature on a date turned out to be higher than the worldwide land average temperature on the same date.

Description

Use the `city.csv` DataFrame to find the maximum average temperature of a country on every date. Count the number of occurences where the country's maximum average temperature on a date turned out to be higher

than the worldwide land average temperature on the same date from the `global.csv` DataFrame and display the count for each country.

Comments

1. The maximum average temperature of a country is defined as the maximum of the average temperatures of all cities in that country on that date.
2. The path to the `city.csv` and `global.csv` DataFrames will be provided as the two command line arguments for the Task.
3. You are *expected* to use as many Spark transformations as possible without using custom code snippets (loops and data structures to manipulate data) since these will speed up your implementation
4. **Important**: Never load the entire datasets into your memory!

Input Format

The input to the Task consists of two command line arguments in the following order:

`path_to_city.csv path_to_global.csv`

Output Format

Display each country with its corresponding count in sorted order of countries. The values are `\t` separated and newline delimited. Do not print the country if it does not have any instances where the required condition has been satisfied.

Example

After computing the maximum average temperature for each country on every date, we obtain the following

1. `1876-02-01`
    1. India: `max(20, 28) = 28`
    2. Germany: `max(27) = 27`

    Since both countries have their maximum temperature greater than the land average temperature on this date (20), we increment the counter for both these countries.

    Current count: `India: 1, Germany: 1`

2. `1862-04-01`

    1. India: `max(32, 34) = 34`
    2. Germany: `max(29) = 29`

    Since only India's maximum temperature is greater than the land average temperature on this date(30), we increment only India's count.

    Current count: `India: 2, Germany: 1`

3. `1856-01-01` :

    1. India: `max(32, 33) = 33`
    2. Germany: `max(26) = 26`

    Since both countries have their maximum temperature greater than the land average temperature on this date (25), we increment the counter for both these countries.

    Current count: `India: 3, Germany: 2`

Hence, the final output will be as follows.

```
Germany 2
India   3
```