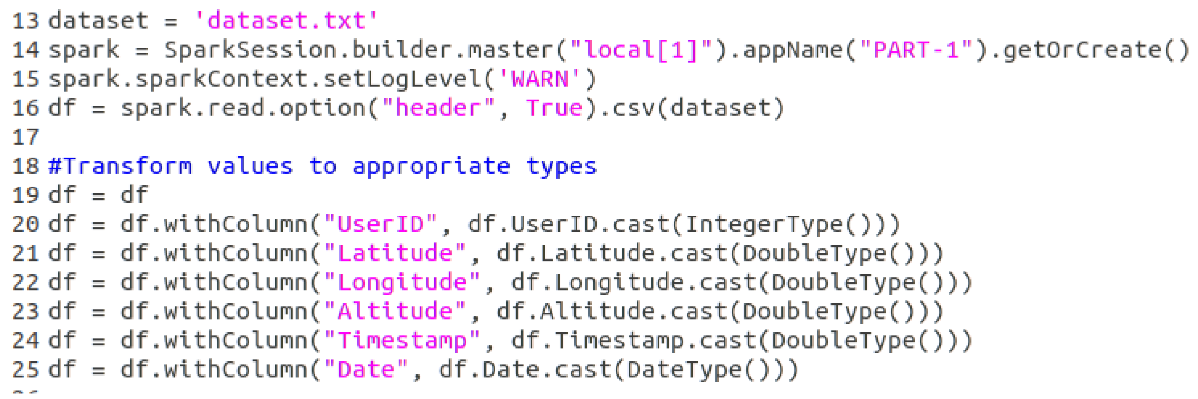
Computer Science Project using Spark, operating on a given dataset.

This file contains the code with comments and results printed out by this code.

In the beginning, I prepare the environment to solve needed exercises. I make sure that data in the provided data frame is in the correct types.

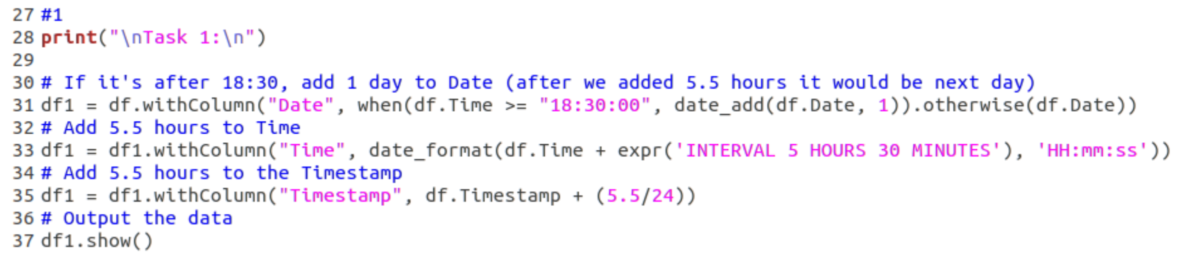


```python
dataset = 'dataset.txt'
spark = SparkSession.builder.master("local[1]").appName("PART-1").getOrCreate()
spark.sparkContext.setLogLevel('WARN')
df = spark.read.option("header", True).csv(dataset)

#Transform values to appropriate types
df = df
df = df.withColumn("UserID", df.UserID.cast(IntegerType()))
df = df.withColumn("Latitude", df.Latitude.cast(DoubleType()))
df = df.withColumn("Longitude", df.Longitude.cast(DoubleType()))
df = df.withColumn("Altitude", df.Altitude.cast(DoubleType()))
df = df.withColumn("Timestamp", df.Timestamp.cast(DoubleType()))
df = df.withColumn("Date", df.Date.cast(DateType()))
```

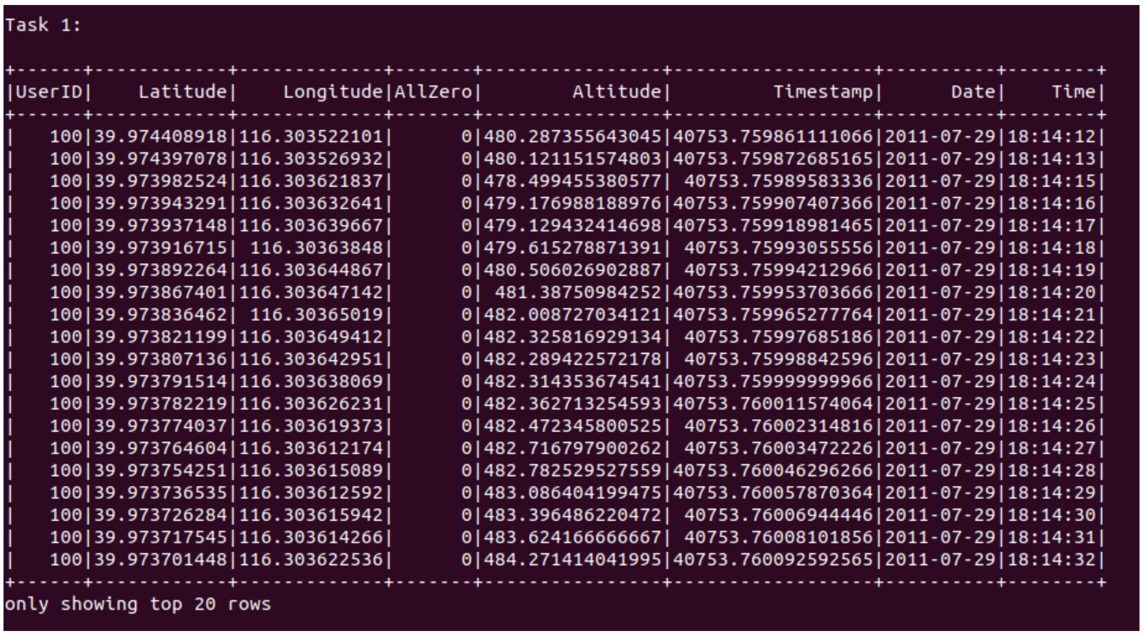
Task 1:

Even though it wasn't required, I printed the result of task 1 to illustrate what it looks like and make sure that all needed values were shifted by 5.5hours or a day.
I first check if it's after or exactly 18:30, so I know if it's going to be the next day after adding 5.5 hours. I add (5.5/24) to Timestamp as it is represented in days. 1day = 24hours



```python
#1
print("\nTask 1:\n")

# If it's after 18:30, add 1 day to Date (after we added 5.5 hours it would be next day)
df1 = df.withColumn("Date", when(df.Time >= "18:30:00", date_add(df.Date, 1)).otherwise(df.Date))
# Add 5.5 hours to Time
df1 = df1.withColumn("Time", date_format(df.Time + expr('INTERVAL 5 HOURS 30 MINUTES'), 'HH:mm:ss'))
# Add 5.5 hours to the Timestamp
df1 = df1.withColumn("Timestamp", df.Timestamp + (5.5/24))
# Output the data
df1.show()
```



```
Task 1:

+------+------------+-------------+-------+---------------+-----------------+----------+--------+
|UserID|    Latitude|    Longitude|AllZero|       Altitude|        Timestamp|      Date|    Time|
+------+------------+-------------+-------+---------------+-----------------+----------+--------+
|   100|39.974408918|116.303522101|      0|480.287355643045|40753.759861111066|2011-07-29|18:14:12|
|   100|39.974397078|116.303526932|      0|480.121151574803|40753.759872685165|2011-07-29|18:14:13|
|   100|39.973982524|116.303621837|      0|478.499455380577| 40753.75989583336|2011-07-29|18:14:15|
|   100|39.973943291|116.303632641|      0|479.176988188976|40753.759907407366|2011-07-29|18:14:16|
|   100|39.973937148|116.303639667|      0|479.129432414698|40753.759918981465|2011-07-29|18:14:17|
|   100|39.973916715| 116.30363848|      0|479.615278871391| 40753.75993055556|2011-07-29|18:14:18|
|   100|39.973892264|116.303644867|      0|480.506026902887| 40753.75994212966|2011-07-29|18:14:19|
|   100|39.973867401|116.303647142|      0| 481.38750984252|40753.759953703666|2011-07-29|18:14:20|
|   100|39.973836462| 116.30365019|      0|482.008727034121|40753.759965277764|2011-07-29|18:14:21|
|   100|39.973821199|116.303649412|      0|482.325816929134| 40753.75997685186|2011-07-29|18:14:22|
|   100|39.973807136|116.303642951|      0|482.289422572178| 40753.75998842596|2011-07-29|18:14:23|
|   100|39.973791514|116.303638069|      0|482.314353674541|40753.759999999966|2011-07-29|18:14:24|
|   100|39.973782219|116.303626231|      0|482.362713254593|40753.760011574064|2011-07-29|18:14:25|
|   100|39.973774037|116.303619373|      0|482.472345800525| 40753.76002314816|2011-07-29|18:14:26|
|   100|39.973764604|116.303612174|      0|482.716797900262| 40753.76003472226|2011-07-29|18:14:27|
|   100|39.973754251|116.303615089|      0|482.782529527559|40753.760046296266|2011-07-29|18:14:28|
|   100|39.973736535|116.303612592|      0|483.086404199475|40753.760057870364|2011-07-29|18:14:29|
|   100|39.973726284|116.303615942|      0|483.396486220472| 40753.76006944446|2011-07-29|18:14:30|
|   100|39.973717545|116.303614266|      0|483.624166666667| 40753.76008101856|2011-07-29|18:14:31|
|   100|39.973701448|116.303622536|      0|484.271414041995|40753.760092592565|2011-07-29|18:14:32|
+------+------------+-------------+-------+---------------+-----------------+----------+--------+
only showing top 20 rows
```

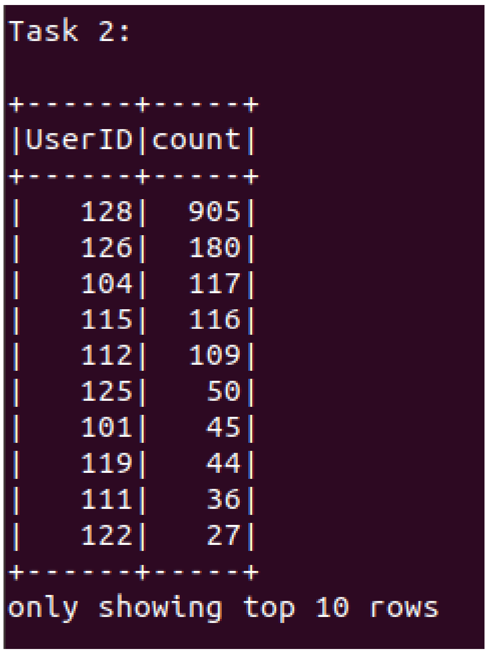
Task 2:

Group data by UserID and Date and count the number of rows.
Filter the data, so that we only have users and days where at least 2 points were registered (count of rows >=2).
Group by UserID and count the number of different days for them in the data frame.
Sort and output the data.

```python
#2
print("\nTask 2:\n")

# Count how many times data was recorded for the same day for the same user
df2 = df1.groupBy("UserID", "Date").count()
# Select UserIDs and Days that have at least 2 data points recorded on a day
df2 = df2.where(col("count") >= 2)
# For each user, count on how many days at least 2 data points were recorded
# Sort by the number of count and UserID number
df2 = df2.groupBy("UserID").count().sort(desc("count"), desc("UserID"))
# Output top 10 rows
df2.show(10)

```

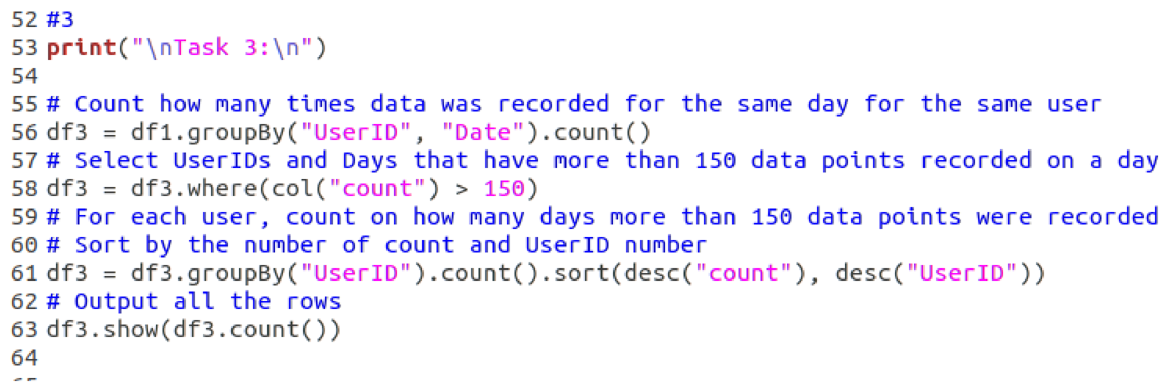


```
Task 2:

+------+-----+
|UserID|count|
+------+-----+
|   128|  905|
|   126|  180|
|   104|  117|
|   115|  116|
|   112|  109|
|   125|   50|
|   101|   45|
|   119|   44|
|   111|   36|
|   122|   27|
+------+-----+
only showing top 10 rows
```

Task 3:

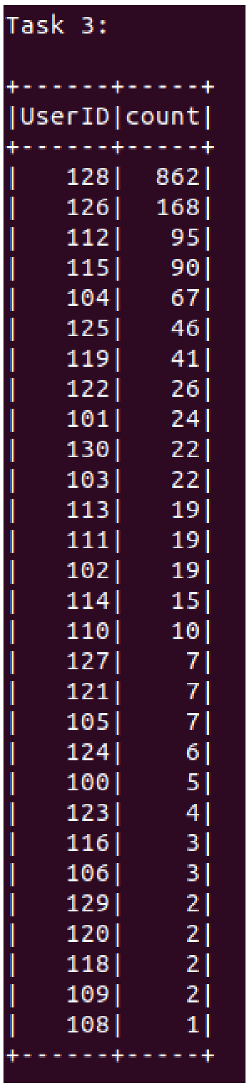
Group data by UserID and Date and count the number of rows.
Filter the data, so that we only have users and days where more than 150 points were registered (count of rows > 150).
Group by UserID and count the number of different days for them in the data frame.
Sort and output the data.



```
#3
print("\nTask 3:\n")

# Count how many times data was recorded for the same day for the same user
df3 = df1.groupBy("UserID", "Date").count()
# Select UserIDs and Days that have more than 150 data points recorded on a day
df3 = df3.where(col("count") > 150)
# For each user, count on how many days more than 150 data points were recorded
# Sort by the number of count and UserID number
df3 = df3.groupBy("UserID").count().sort(desc("count"), desc("UserID"))
# Output all the rows
df3.show(df3.count())

```

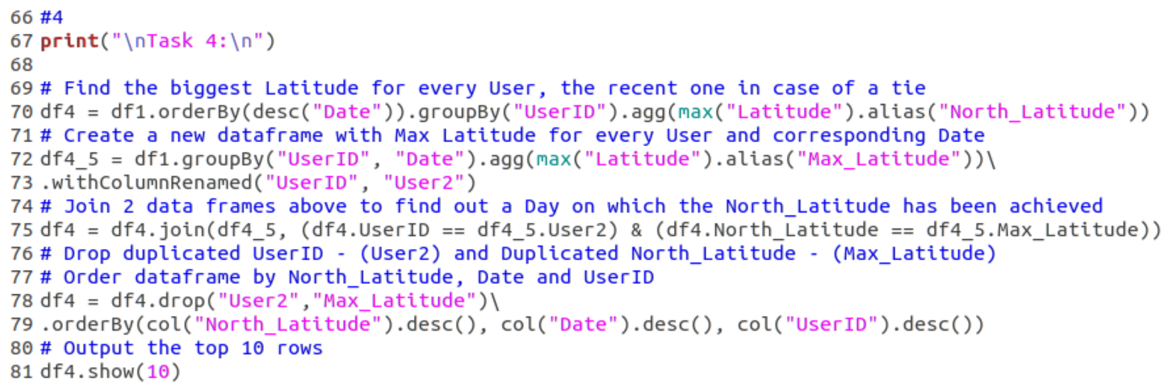


```
Task 3:

+------+-----+
|UserID|count|
+------+-----+
|   128|  862|
|   126|  168|
|   112|   95|
|   115|   90|
|   104|   67|
|   125|   46|
|   119|   41|
|   122|   26|
|   101|   24|
|   130|   22|
|   103|   22|
|   113|   19|
|   111|   19|
|   102|   19|
|   114|   15|
|   110|   10|
|   127|    7|
|   121|    7|
|   105|    7|
|   124|    6|
|   100|    5|
|   123|    4|
|   116|    3|
|   106|    3|
|   129|    2|
|   120|    2|
|   118|    2|
|   109|    2|
|   108|    1|
+------+-----+
```

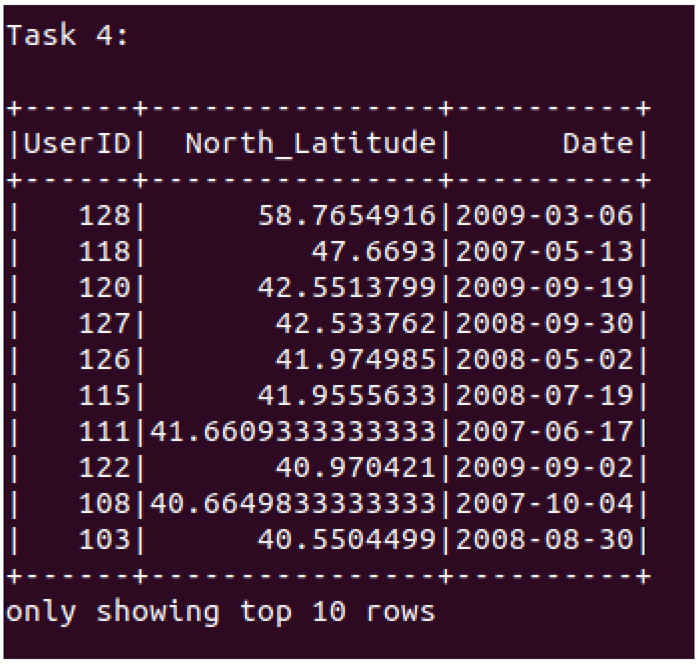
Task 4:

Find the largest Latitude for every UserID.
Create a new data frame with the largest Latitude for each Day.
Join both data frames to find a Day on which max Latitude has been achieved.
Drop duplicated columns, order the results, and print them out.



```python
#4
print("\nTask 4:\n")

# Find the biggest Latitude for every User, the recent one in case of a tie
df4 = df1.orderBy(desc("Date")).groupBy("UserID").agg(max("Latitude").alias("North_Latitude"))
# Create a new dataframe with Max Latitude for every User and corresponding Date
df4_5 = df1.groupBy("UserID", "Date").agg(max("Latitude").alias("Max_Latitude"))\
.withColumnRenamed("UserID", "User2")
# Join 2 data frames above to find out a Day on which the North_Latitude has been achieved
df4 = df4.join(df4_5, (df4.UserID == df4_5.User2) & (df4.North_Latitude == df4_5.Max_Latitude))
# Drop duplicated UserID - (User2) and Duplicated North_Latitude - (Max_Latitude)
# Order dataframe by North_Latitude, Date and UserID
df4 = df4.drop("User2","Max_Latitude")\
.orderBy(col("North_Latitude").desc(), col("Date").desc(), col("UserID").desc())
# Output the top 10 rows
df4.show(10)
```



```
Task 4:

+------+----------------+----------+
|UserID|  North_Latitude|      Date|
+------+----------------+----------+
|   128|      58.7654916|2009-03-06|
|   118|         47.6693|2007-05-13|
|   120|      42.5513799|2009-09-19|
|   127|       42.533762|2008-09-30|
|   126|       41.974985|2008-05-02|
|   115|      41.9555633|2008-07-19|
|   111|41.6609333333333|2007-06-17|
|   122|       40.970421|2009-09-02|
|   108|40.6649833333333|2007-10-04|
|   103|      40.5504499|2008-08-30|
+------+----------------+----------+
only showing top 10 rows
```

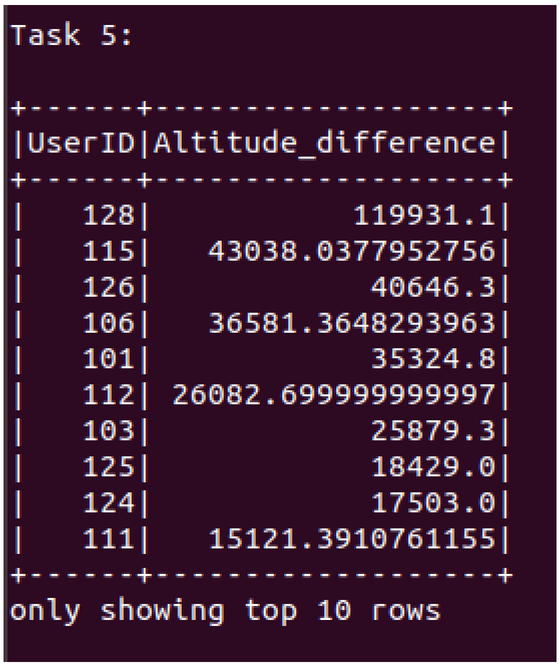
Task 5:

Find the maximum and minimum Altitude ever achieved by each User.
Calculate the difference between Max and Min altitude and assign the value to a new column. Drop the unnecessary columns, order the data and print it out.

```python
#5

print("\nTask 5:\n")

# Find the max and min altitude for every UserID
df5 = df1.groupBy("UserID").agg(max("Altitude").alias("Max_altitude"),min("Altitude").alias("Min_altitude"))
# Find for each user the difference between min and max altitudes achieved by them
df5 = df5.withColumn("Altitude_difference", df5.Max_altitude - df5.Min_altitude)
# Drop the unneccesary columns and sort by Altitude_difference, UserID
df5 = df5.drop("Max_altitude","Min_altitude")\
.orderBy(col("Altitude_difference").desc(), col("UserID").desc())
# Output the top 10 rows
df5.show(10)
```



```
Task 5:

+------+-------------------+
|UserID|Altitude_difference|
+------+-------------------+
|   128|           119931.1|
|   115|   43038.0377952756|
|   126|            40646.3|
|   106|   36581.3648293963|
|   101|            35324.8|
|   112| 26082.699999999997|
|   103|            25879.3|
|   125|            18429.0|
|   124|            17503.0|
|   111|   15121.3910761155|
+------+-------------------+
only showing top 10 rows
```

Task 6:

Divided into three parts. The 'zero' part is used in both solutions (first part and second).

Partition the data by User and order by the timestamp.
Create a new column with an Altitude value from the next row (lag, over the partition).
Calculate the difference between two altitudes and drop the NULL values (when there was no next element to get a lag altitude value).
Filter the data so we only have the Altitude difference when the user was going up.
Sum up the altitude climbed by every user every day.

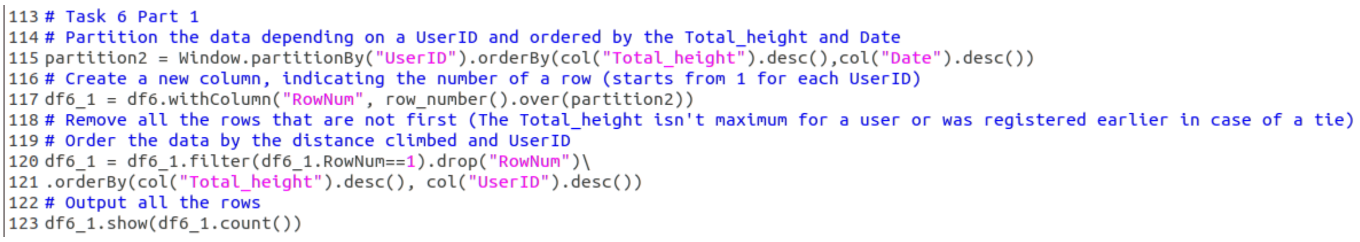
```python
#6

print("\nTask 6:\n")

# Partition the data for every UserID and order it by Timestamp (ASC)
partition = Window.partitionBy("UserID").orderBy("Timestamp")
# Create a lag on Altitude over the partition above
df6 = df1.withColumn("Lag_alti",lag("Altitude",1).over(partition))
# Create a new colum that is the difference between registered altitude and altitude registered at the next measure (from lag), drop the NULLS
df6 = df6.withColumn("High_difference", df6.Altitude - df6.Lag_alti).dropna()
# Drop all the values that are less or equal to 0. We don't count going down and 0s are unneccesary.
df6 = df6.filter(df6.High_difference>0)
# Group the data by UserID and Data, sum up the distance climbed on each day for a User.
df6 = df6.groupBy("UserID", "Date").agg(sum("High_difference").alias("Total_height"))
```

__For Part 1 (Task 6 part 1):__

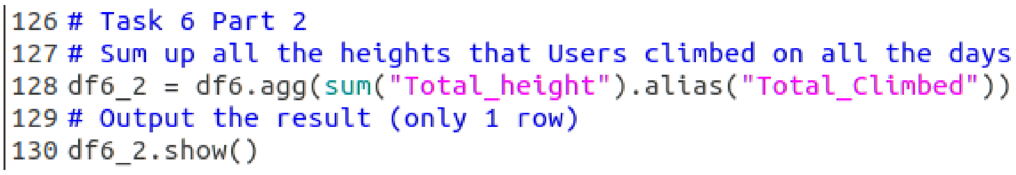
Partition the data once again by the UserID and sort it by the altitude climbed.
Assign row numbers to the data frame and drop all but the first ones (keep the day when the user climbed the largest distance).
Drop the row number column, sort the data and output all the results.



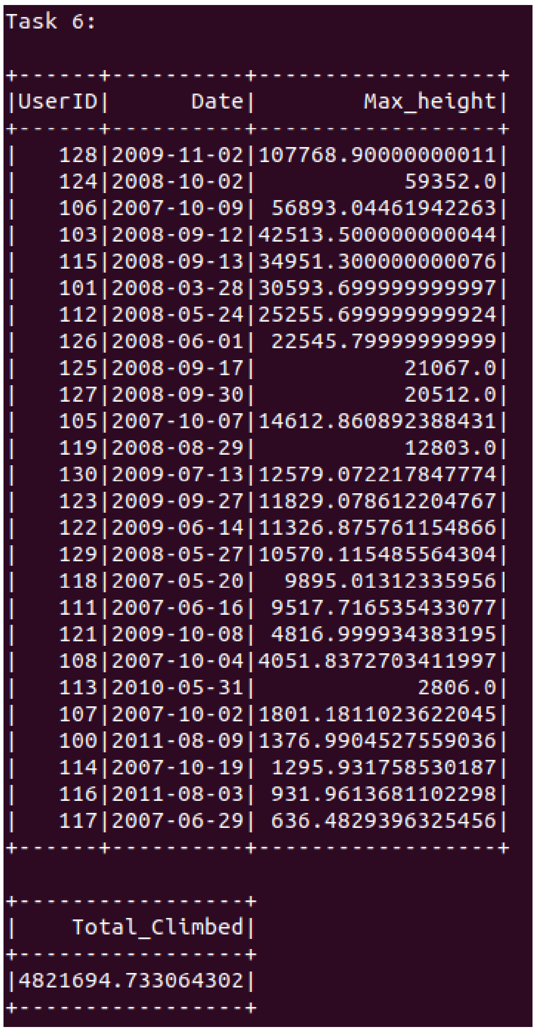
```
# Task 6 Part 1
# Partition the data depending on a UserID and ordered by the Total_height and Date
partition2 = Window.partitionBy("UserID").orderBy(col("Total_height").desc(),col("Date").desc())
# Create a new column, indicating the number of a row (starts from 1 for each UserID)
df6_1 = df6.withColumn("RowNum", row_number().over(partition2))
# Remove all the rows that are not first (The Total_height isn't maximum for a user or was registered earlier in case of a tie)
# Order the data by the distance climbed and UserID
df6_1 = df6_1.filter(df6_1.RowNum==1).drop("RowNum")\
.orderBy(col("Total_height").desc(), col("UserID").desc())
# Output all the rows
df6_1.show(df6_1.count())
```

__For Part 2 (Task 6 part 2):__

Sum up all the heights climbed on all the days and output the result.



```
# Task 6 Part 2
# Sum up all the heights that Users climbed on all the days
df6_2 = df6.agg(sum("Total_height").alias("Total_Climbed"))
# Output the result (only 1 row)
df6_2.show()
```



```
Task 6:

+------+----------+-----------------+
|UserID|      Date|       Max_height|
+------+----------+-----------------+
|   128|2009-11-02|107768.90000000011|
|   124|2008-10-02|          59352.0|
|   106|2007-10-09| 56893.04461942263|
|   103|2008-09-12|42513.500000000044|
|   115|2008-09-13|34951.300000000076|
|   101|2008-03-28|30593.699999999997|
|   112|2008-05-24|25255.699999999924|
|   126|2008-06-01| 22545.79999999999|
|   125|2008-09-17|          21067.0|
|   127|2008-09-30|          20512.0|
|   105|2007-10-07|14612.860892388431|
|   119|2008-08-29|          12803.0|
|   130|2009-07-13|12579.072217847774|
|   123|2009-09-27|11829.078612204767|
|   122|2009-06-14|11326.875761154866|
|   129|2008-05-27|10570.115485564304|
|   118|2007-05-20|  9895.01312335956|
|   111|2007-06-16| 9517.716535433077|
|   121|2009-10-08| 4816.999934383195|
|   108|2007-10-04|4051.8372703411997|
|   113|2010-05-31|           2806.0|
|   107|2007-10-02|1801.1811023622045|
|   100|2011-08-09|1376.9904527559036|
|   114|2007-10-19| 1295.931758530187|
|   116|2011-08-03| 931.9613681102298|
|   117|2007-06-29| 636.4829396325456|
+------+----------+-----------------+

+-----------------+
|    Total_Climbed|
+-----------------+
|4821694.733064302|
+-----------------+
```