```
from pyspark.sql import SQLContext
from pyspark.sql import SparkSession

from pyspark.sql.functions import acos, cos, sin, lit, toRadians,radians
from pyspark.sql.types import*
import pyspark.sql.functions as F
from pyspark.sql.functions import desc,count
from pyspark.sql.functions import date_format
from pyspark.sql.functions import to_date,datediff,date_sub
from pyspark.sql.functions import to_timestamp
from pyspark.sql.functions import concat, split, lit, from_unixtime, unix_timestamp
from pyspark.sql import functions as F
import pyspark.sql.functions as fn
from pyspark.sql.functions import row_number
from pyspark.sql.window import Window
from pyspark.sql.functions import lag
from pyspark.sql.functions import rank
from pyspark.sql.window import Window
from pyspark.sql.functions import col
from pyspark.sql import functions as F

import numpy as np

#Importing the dataset

dataframe=spark.read.csv("/home/comp529/Desktop/dataset.csv",header=True)
dataframe.show()
#Task 1
#converting all dates/times from GMT to Beijing time

df=dataframe.withColumn('Time',date_format(to_timestamp(concat(df.Date,lit("
"),df.Time)+ F.expr('INTERVAL 8 HOURS')),'HH:mm:ss'))
df=dataframe.withColumn('Date',to_date(concat(df.Date,lit(" "),df.Time)+
F.expr('INTERVAL 8 HOURS')))
df.show()
#Task 2
#Calculating for each person on how many days the data was recorded and displaying
first 5 used id's

dataframe.createOrReplaceTempView("dataset")
df1=spark.sql("select b.UserID,count(UserID) from (select UserID,Date from dataset
where Latitude is NOT NULL group by UserID,Date)b group by UserID order by
count(UserID) desc" )
df1.show(5)

#Task 3
#Calculating for each person on how many days there were more than 100 data points
and displaying them
df2=spark.sql("select b.UserID,count(UserID) from (select UserID,Date,count(Date)
as Daycount from dataset where Latitude is NOT NULL group by UserID,Date)b where
Daycount>100 group by UserID")
df2.show()

#Task 4
#Calculating for each person the highest altitude they reached and displaying top 5
according to its measure

window = Window.partitionBy(df['UserId']).orderBy(df['Altitude'].desc())
dataframe.select('*', rank().over(window).alias('rank')).filter(col('rank') <=
5).show(5)

#Task 5
#Calculating for each person, the timespan of the observation, i.e., the difference
between the highest timestamp of his/her observation and the lowest one and
displaying top 5

ts_df = dataframe.groupBy("UserID").agg(F.max("Timestamp")-
F.min("Timestamp")).sort('(max(Timestamp) - min(Timestamp))', ascending = False)
ts_df.show(5)

#Task 6
#Based on the UserdId, order the data based on the DateTime, and use the lag
function, we have to calculate the distance travelled by the users.


w_task6 = Window().partitionBy("UserId").orderBy("DateTime")
task6 = dfWithDay.withColumn("dist", dist(
    "Longitude", "Latitude",
    lag("Longitude", 1).over(w_question6), lag("Latitude", 1).over(w_question6)
).alias("dist"))
task6_a = task6.select("UserId", "dist", "Date").groupBy("UserId",
"Date").agg(f.sum("dist")).filter(F.col("sum(dist)") != "NaN")
task6_c.show()


#For each user output the (earliest) day they travelled the most
w_6_b = Window.partitionBy('UserId')
task6_b = task6_c.withColumn('maxB', F.max('sum(dist)').over(w_6_b))\
    .where(F.col('sum(dist)') == F.col('maxB'))\
    .drop('maxB')

task6_c.show()
```