Big Data Analytics Assignment 2

PART 1:- DATA ANALYSIS USING PYSPARK

PySpark will be used in this assignment to analyze a GPS trajectory dataset that was collected by 100+ people in the Geolife project (Microsoft Research Asia) over five years. Each GPS trajectory in this dataset is represented as a sequence of time-stamped points.

As the entire data set is too large, I will find relevant subsets where selected individual trajectory data were combined into a single CSV file.

The first line of this file contains a header:

UserID,Latitude,Longitude,AllZero,Altitude,Timestamp,Date,Time

Pyspark code

Task 1:- Usually data analysis starts with data cleaning (in fact, it usually takes most of the time). In this case, we need to convert all dates/times from GMT to Beijing time, where many of these trajectory data were collected. To do this, we need to move dates, times, and timestamps by 8 hours ahead.

Solution:-

First of all we have to import all the important packages for pyspark and all other packages which is required.
We have to define one function expression that is F.expr () is a SQL function that executes SQL-like expressions and uses a DataFrame column value as an expression argument to Pyspark's built-in functions

.

```
!pip install pyspark
```

```
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_ti
mestamp
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
import numpy as np



 #Task 1 - Adding 8 hours to Time and Date
print("Cleansed Dataframe")
df=df.withColumn('Time',date_format(to_timestamp(concat(df.Date,lit(" "),df.Time)+
F.expr('INTERVAL 8 HOURS')),'HH:mm:ss'))
df=df.withColumn('Date',to_date(concat(df.Date,lit(" "),df.Time)+ F.expr('INTERVAL 8
HOURS')))
print(df)
df.show()
```

Above is the Output of loading the database

```
>>> dataframe=spark.read.csv("/home/comp529/Downloads/df.csv",header=True)
>>> dataframe.show()
+------+-----------+-----------+-----------+-----------+----------+--------+
|UserID|   Latitude|  Longitude|   Altitude|  Timestamp|      Date|    Time|
+------+-----------+-----------+-----------+-----------+----------+--------+
|   100|39.97440892|116.3035221|480.2873556|40753.53069|29-07-2011|12:44:12|
|   100|39.97439708|116.3035269|480.1211516|40753.53071|29-07-2011|12:44:13|
|   100|39.97398252|116.3036218|478.4994554|40753.53073|29-07-2011|12:44:15|
|   100|39.97394329|116.3036326|479.1769882|40753.53074|29-07-2011|12:44:16|
|   100|39.97393715|116.3036397|479.1294324|40753.53075|29-07-2011|12:44:17|
|   100|39.97391672|116.3036385|479.6152789|40753.53076|29-07-2011|12:44:18|
|   100|39.97389226|116.3036449|480.5060269|40753.53078|29-07-2011|12:44:19|
|   100| 39.9738674|116.3036471|481.3875098|40753.53079|29-07-2011|12:44:20|
|   100|39.97383646|116.3036502| 482.008727| 40753.5308|29-07-2011|12:44:21|
|   100| 39.9738212|116.3036494|482.3258169|40753.53081|29-07-2011|12:44:22|
|   100|39.97380714| 116.303643|482.2894226|40753.53082|29-07-2011|12:44:23|
|   100|39.97379151|116.3036381|482.3143537|40753.53083|29-07-2011|12:44:24|
|   100|39.97378222|116.3036262|482.3627133|40753.53084|29-07-2011|12:44:25|
|   100|39.97377404|116.3036194|482.4723458|40753.53086|29-07-2011|12:44:26|
|   100| 39.9737646|116.3036122|482.7167979|40753.53087|29-07-2011|12:44:27|
|   100|39.97375425|116.3036151|482.7825295|40753.53088|29-07-2011|12:44:28|
|   100|39.97373654|116.3036126|483.0864042|40753.53089|29-07-2011|12:44:29|
|   100|39.97372628|116.3036159|483.3964862| 40753.5309|29-07-2011|12:44:30|
|   100|39.97371755|116.3036143|483.6241667|40753.53091|29-07-2011|12:44:31|
|   100|39.97370145|116.3036225| 484.271414|40753.53093|29-07-2011|12:44:32|
+------+-----------+-----------+-----------+-----------+----------+--------+
only showing top 20 rows
```

Here we are converting all dates/times from GMT to Beijing time.

OUTPUT:-

```
+------+-----------+------------+-------+---------------+---------------+----------+--------+
|UserID|   Latitude|   Longitude|AllZero|       Altitude|      Timestamp|      Date|    Time|
+------+-----------+------------+-------+---------------+---------------+----------+--------+
|   100|39.974408918|116.303522101|      0|480.287355643045|40753.5306944444|2011-07-29|20:44:12|
|   100|39.974397078|116.303526932|      0|480.121151574803|40753.5307060185|2011-07-29|20:44:13|
|   100|39.973982524|116.303621837|      0|478.499455380577|40753.5307291667|2011-07-29|20:44:15|
|   100|39.973943291|116.303632641|      0|479.176988188976|40753.5307407407|2011-07-29|20:44:16|
|   100|39.973937148|116.303639667|      0|479.129432414698|40753.5307523148|2011-07-29|20:44:17|
|   100|39.973916715| 116.30363848|      0|479.615278871391|40753.5307638889|2011-07-29|20:44:18|
|   100|39.973892264|116.303644867|      0|480.506026902887| 40753.530775463|2011-07-29|20:44:19|
|   100|39.973867401|116.303647142|      0| 481.38750984252| 40753.530787037|2011-07-29|20:44:20|
|   100|39.973836462| 116.30365019|      0|482.008727034121|40753.5307986111|2011-07-29|20:44:21|
|   100|39.973821199|116.303649412|      0|482.325816929134|40753.5308101852|2011-07-29|20:44:22|
|   100|39.973807136|116.303642951|      0|482.289422572178|40753.5308217593|2011-07-29|20:44:23|
|   100|39.973791514|116.303638069|      0|482.314353674541|40753.5308333333|2011-07-29|20:44:24|
|   100|39.973782219|116.303626231|      0|482.362713254593|40753.5308449074|2011-07-29|20:44:25|
|   100|39.973774037|116.303619373|      0|482.472345800525|40753.5308564815|2011-07-29|20:44:26|
|   100|39.973764604|116.303612174|      0|482.716797900262|40753.5308680556|2011-07-29|20:44:27|
|   100|39.973754251|116.303615089|      0|482.782529527559|40753.5308796296|2011-07-29|20:44:28|
|   100|39.973736535|116.303612592|      0|483.086404199475|40753.5308912037|2011-07-29|20:44:29|
|   100|39.973726284|116.303615942|      0|483.396486220472|40753.5309027778|2011-07-29|20:44:30|
|   100|39.973717545|116.303614266|      0|483.624166666667|40753.5309143519|2011-07-29|20:44:31|
|   100|39.973701448|116.303622536|      0|484.271414041995|40753.5309259259|2011-07-29|20:44:32|
+------+-----------+------------+-------+---------------+---------------+----------+--------+
only showing top 20 rows
```

#Task 2 – Number of days data was recorded for each user and top 5.

Solution:-

df.createOrReplaceTempView("dataset")
df7=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")
df7.show(5)
The number of days data can be recorded by selecting the userid , date from dataset

OUTPUT:-

Above output is for calculating for each person on how many days the data was recorded and displaying first 5 used id's.

#Task 3 - Number of days there were more than 100 data points recorded for each user's.

df8=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")
df8.show()

OUTPUT :-

We haveto calculate for each person on how many days there were more than 100 data points and displaying them

#Task 4 – Determine the highest altitude that each person has reached. List the top five user IDs by this measure, their value, and the day that they achieved it (in case of a tie, list the earliest such a day).

Solutiuon :-

```
from pyspark.sql.functions import col
from pyspark.sql.functions import rank
from pyspark.sql.window import Window
window = Window.partitionBy(df['UserId']).orderBy(df['Altitude'].desc())
df.select('*', rank().over(window).alias('rank')).filter(col('rank') <= 5).show(5)
```

OUTPUT:-

```
>>> from pyspark.sql.functions import col
>>> window = Window.partitionBy(df['UserId']).orderBy(df['Altitude'].desc())
>>> dataframe.select('*', rank().over(window).alias('rank')).filter(col('rank') <= 5).show(5)
+------+---------+----------+--------+-----------+----------+--------+----+
|UserID| Latitude| Longitude|Altitude|  Timestamp|      Date|    Time|rank|
+------+---------+----------+--------+-----------+----------+--------+----+
|   125|25.738228|112.732518|     999|39688.21181|28-08-2008|05:05:00|   1|
|   125|25.737866|112.729258|     999|39688.21325|28-08-2008|05:07:05|   1|
|   125|25.737717|112.729269|     999|39688.21343|28-08-2008|05:07:20|   1|
|   125|25.737124|112.728733|     999|39688.22222|28-08-2008|05:20:00|   1|
|   125|25.737146|112.728738|     999|39688.22228|28-08-2008|05:20:05|   1|
+------+---------+----------+--------+-----------+----------+--------+----+
only showing top 5 rows
```

#Task 5 - Find for each person the difference between the highest and lowest time stamp of their observation, i.e.,
the time span of their observation.

Solution :-

```
 -ts_df = df4.groupBy("UserID").agg(f.max("Timestamp")-
F.min("Timestamp")).sort('(max(Timestamp) - min(Timestamp))', ascending = False)

ts_df.show(5)
```

OUTPUT:-

```
>>> ts_df = dataframe.groupBy("UserID").agg(F.max("Timestamp")-F.min("Timestamp")).sort('(max(Timestamp) - min(Timestamp))', ascending = False)
>>> ts_df.show(5)
+------+-------------------------------+
|UserID|(max(Timestamp) - min(Timestamp))|
+------+-------------------------------+
|   114|              963.8455900000044|
|   111|               838.7832199999975|
|   115|               506.6880400000009|
|   104|             317.21798000000126|
|   101|             212.69595000000118|
+------+-------------------------------+
only showing top 5 rows
```

#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.

Solution:-

```
dfwithDay = df2.withcolumn(“day”,f.dayofmonth(df2.datetimeBj))
w_task6 = Window().partitionBy("UserId").orderBy("DateTime")
task6 = dfWithDay.withColumn("dist", dist(
    "Longitude", "Latitude",
    lag("Longitude", 1).over(w_task6), lag("Latitude", 1).over(w_task6)
).alias("dist"))
task6_c = task6.select("UserId", "dist", "Date").groupBy("UserId",
"Date").agg(f.sum("dist")).filter(F.col("sum(dist)") != "NaN")
question6_a.show()
```

```
#For each user output the (earliest) day they travelled the most
w_6_c = Window.partitionBy('UserId')
task6_c = task6_a.withColumn('maxB', F.max('sum(dist)').over(w_6_b))\
    .where(F.col('sum(dist)') == F.col('maxB'))\
    .drop('maxB')

task6_c.show()
```

OUTPUT:-

```
    #Find the total distance by all the users
    question6_c = question6_a.select(f.sum("sum(dist)").alias("total_sum"))
    question6_c.show()
```

```
+------------------+
|         total_sum|
+------------------+
|124208.62254385433|
+------------------+
```

```
|UserId|      Date|          sum(dist)|
+------+----------+-------------------+
|   108|2007-10-02|1.6587260860085606|
|   108|2007-10-03|43.631893458311964|
|   108|2007-10-04| 147.0055120203384|
|   108|2007-10-06|121.43545197781773|
|   108|2007-10-07| 7.560496310794932|
|   108|2007-10-08|3.5475681716161547|
|   108|2007-10-09| 1.526404310495542|
|   101|2007-11-30| 35.71357885259294|
|   101|2007-12-02| 26.28155622300305|
|   101|2007-12-03|13.946825605235945|
|   101|2007-12-07|21.582506892854884|
|   101|2007-12-11|1.2158358355356826|
|   101|2007-12-12| 5.240018538952616|
|   101|2007-12-13| 131.2705465948174|
|   101|2007-12-15| 134.2261667257604|
|   101|2007-12-19| 157.9404104628446|
|   101|2007-12-22| 222.8093068237573|
|   101|2007-12-23| 8.639073137599118|
|   101|2007-12-26|2.4209762057765114|
|   101|2007-12-27|3.9078701419516726|
```

UserId	Date	sum(dist)
108	2007-10-04	147.0055120203384
101	2008-01-25	912.3501366350881
115	2007-11-28	2097.446018079143
126	2008-05-01	372.51247632567714
103	2008-09-19	29.44931227567783
128	2009-02-22	10090.016973407062
122	2009-07-31	1967.2757652846492
111	2007-09-05	2462.021045854465
117	2007-06-22	26.30900937760673
112	2008-02-02	1078.383461221913
127	2008-10-05	1028.5007633041885
107	2007-10-07	8.659731775734203
114	2010-05-28	46.56970415564099
100	2011-07-29	10.965117553721749
130	2009-07-12	103.34148374177562
129	2008-05-02	317.7130265707075
102	2011-12-31	31.239379907177888
113	2010-05-20	19.666718577249753
121	2009-10-05	12.850327012071368
125	2008-08-27	1597.3327329740112