BDA2Eric

May 16, 2023

0.1 Assignments

0.1.1 Same as BDA1, but use built-in API functions for all the 5 exercises.

0.1.2 Q1

year, station with the max, maxValue ORDER BY maxValue DESC

year, station with the min, minValue ORDER BY minValue DESC

[ ]:
```
#Q1
from pyspark import SparkContext
from pyspark.sql import SparkSession, SQLContext
from pyspark.sql.types import StructType, StructField, StringType, FloatType
from pyspark.sql import functions as F

sc = SparkContext(appName="exercise 1")
spark = SparkSession(sc)
sqlContext = SQLContext(sc)

temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

year_temperature = lines.map(lambda x: (x[1][0:4], x[0], float(x[3])))

schema = StructType([
    StructField("year", StringType(), True),
    StructField("station", StringType(), True),
    StructField("value", FloatType(), True)
])

year_temperature_df = sqlContext.createDataFrame(year_temperature, schema)

filtered_year_temperature = year_temperature_df.
 ↪filter((year_temperature_df["year"] >= "1950") &␣
 ↪(year_temperature_df["year"] <= "2014"))

max_temperatures = filtered_year_temperature.groupBy('year', 'station').agg(F.
 ↪max('value').alias('maxValue')).orderBy(['year', 'station', 'maxValue'],␣
 ↪ascending=[False, False, True])
```

```
max_temperatures_combine = max_temperatures.rdd.coalesce(1)
max_temperatures_combine = max_temperatures_combine.sortBy(lambda x: x[2],␣
 ↪ascending=False)
max_temperatures_combine.saveAsTextFile("BDA/output/l2max")



min_temperatures = filtered_year_temperature.groupBy('year', 'station').agg(F.
 ↪min('value').alias('minValue')).orderBy(['year', 'station', 'minValue'],␣
 ↪ascending=[False, False, True])
min_temperatures_combine = min_temperatures.rdd.coalesce(1)
min_temperatures_combine = min_temperatures_combine.sortBy(lambda x: x[2],␣
 ↪ascending=False)
min_temperatures_combine.saveAsTextFile("BDA/output/l2min")
```

0.1.3 output of l2min

Row(year=u'2010', station=u'95530', minValue=15.199999809265137)
Row(year=u'1979', station=u'99090', minValue=13.100000381469727)
Row(year=u'1984', station=u'53220', minValue=12.0)
Row(year=u'2001', station=u'117160', minValue=8.0)
Row(year=u'2010', station=u'89560', minValue=7.900000095367432)
Row(year=u'1998', station=u'104390', minValue=7.5)
Row(year=u'1986', station=u'84390', minValue=5.300000190734863)
Row(year=u'2009', station=u'71140', minValue=4.900000095367432)
Row(year=u'1970', station=u'107530', minValue=4.099999904632568)
Row(year=u'1955', station=u'65640', minValue=3.4000000953674316)

0.1.4 output of l2max

Row(year=u'1975', station=u'86200', maxValue=36.099998474121094)
Row(year=u'1975', station=u'95160', maxValue=35.79999923706055)
Row(year=u'1975', station=u'96550', maxValue=35.599998474121094)
Row(year=u'1975', station=u'106100', maxValue=35.5)
Row(year=u'1992', station=u'63600', maxValue=35.400001525878906)
Row(year=u'1975', station=u'75240', maxValue=35.400001525878906)
Row(year=u'1992', station=u'63050', maxValue=35.20000076293945)
Row(year=u'1992', station=u'85040', maxValue=35.0)
Row(year=u'1992', station=u'76000', maxValue=35.0)
Row(year=u'1992', station=u'75240', maxValue=35.0)

0.1.5 Q2

year, month, value ORDER BY value DESC

year, month, value ORDER BY value DESC

[ ]:
```
#Q2
from pyspark import SparkContext
from pyspark.sql import SparkSession, SQLContext
from pyspark.sql.types import StructType, StructField, StringType, FloatType
from pyspark.sql import functions as F

sc = SparkContext(appName="exercise 1")
spark = SparkSession(sc)
sqlContext = SQLContext(sc)

temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
lines = temperature_file.map(lambda line: line.split(";"))

year_month_temperature = lines.map(lambda x: (x[1][0:4],x[1][5:
 ↪7],x[0],float(x[3])))

schema = StructType([
    StructField("year", StringType(), True),
    StructField("month", StringType(), True),
    StructField("station", StringType(), True),
    StructField("value", FloatType(), True)
])

year_month_temperature_df = sqlContext.createDataFrame(year_month_temperature,␣
 ↪schema)

filtered_year_month_temperature = year_month_temperature_df.
 ↪filter((year_month_temperature_df["year"] >= "1950") &␣
 ↪(year_month_temperature_df["year"] <= "2014")&␣
 ↪(year_month_temperature_df["value"] > 10))

count_temp = filtered_year_month_temperature.groupBy(["year", "month"]).count()
sort_count = count_temp.sort("count", ascending = False)
sort_count_combine = sort_count.rdd.coalesce(1)
sort_count_combine = sort_count_combine.sortBy(lambda x: x[2], ascending=False)
sort_count_combine.saveAsTextFile("BDA/output/l2_not_distinct")

distinct_temp = filtered_year_month_temperature.select(["year", "month",␣
 ↪"station"]).distinct()
count_dist_temp = distinct_temp.groupBy(["year", "month"]).count()
sort_count_dist = count_dist_temp.sort("count", ascending = False)
sort_count_dist = sort_count_dist.rdd.coalesce(1)
sort_count_dist = sort_count_dist.sortBy(lambda x: x[2], ascending=False)
sort_count_dist.saveAsTextFile("BDA/output/l2_distinct")
```

0.1.6 output of l2_not_distinct

Row(year=u'2014', month=u'07', count=147681)
Row(year=u'2011', month=u'07', count=146656)
Row(year=u'2010', month=u'07', count=143419)
Row(year=u'2012', month=u'07', count=137477)
Row(year=u'2013', month=u'07', count=133657)
Row(year=u'2009', month=u'07', count=133008)
Row(year=u'2011', month=u'08', count=132734)
Row(year=u'2009', month=u'08', count=128349)
Row(year=u'2013', month=u'08', count=128235)
Row(year=u'2003', month=u'07', count=128133)

0.1.7 output of l2_distinct

Row(year=u'1972', month=u'10', count=378)
Row(year=u'1973', month=u'05', count=377)
Row(year=u'1973', month=u'06', count=377)
Row(year=u'1972', month=u'08', count=376)
Row(year=u'1973', month=u'09', count=376)
Row(year=u'1972', month=u'06', count=375)
Row(year=u'1972', month=u'09', count=375)
Row(year=u'1971', month=u'08', count=375)
Row(year=u'1972', month=u'05', count=375)
Row(year=u'1971', month=u'06', count=374)

0.1.8 Q4

station, maxTemp, maxDailyPrecipitation ORDER BY station DESC

[ ]:
```
#Q4
from pyspark import SparkContext
from pyspark.sql import SparkSession, SQLContext
from pyspark.sql.types import StructType, StructField, StringType, FloatType
from pyspark.sql import functions as F

sc = SparkContext(appName="exercise 1")
spark = SparkSession(sc)
sqlContext = SQLContext(sc)

temperature_file = sc.textFile("BDA/input/temperature-readings.csv")
temperature_lines = temperature_file.map(lambda line: line.split(";"))
get_temperature = temperature_lines.map(lambda x: (x[0],float(x[3])))
tempschema = StructType([
    StructField("station", StringType(), True),
    StructField("temp", FloatType(), True)
])
temperature_df = sqlContext.createDataFrame(get_temperature, tempschema)
```

```
station_max_temp = temperature_df.groupBy("station").agg(F.max("temp").
 ↪alias('temp'))
filter_temp = station_max_temp.filter((station_max_temp["temp"] >= "25") &␣
 ↪(station_max_temp["temp"] <= "30"))

precipitation_file = sc.textFile("BDA/input/precipitation-readings.csv")
precipitation_file = precipitation_file.map(lambda line: line.split(";"))
get_percipitation = precipitation_file.map(lambda x: (x[0],float(x[3])))
precschema = StructType([
    StructField("station", StringType(), True),
    StructField("prec", FloatType(), True)
])
percipitation_df = sqlContext.createDataFrame(get_percipitation, precschema)
station_max_perc = percipitation_df.groupBy("station").agg(F.max("prec").
 ↪alias('prec'))
filter_perc = station_max_perc.filter((station_max_perc["prec"] >= "100") &␣
 ↪(station_max_perc["prec"] <= "200"))

combine_temp_perc = filter_temp.join(filter_perc.alias('perc'), 'station',␣
 ↪'inner')

#output
combine_temp_perc_combine = combine_temp_perc.rdd.coalesce(1)
filter_temp_combine = combine_temp_perc_combine.sortBy(lambda x: x[0],␣
 ↪ascending=False)
filter_temp_combine.saveAsTextFile("BDA/output/l2_perc_temp")


#Testoutput
#filter_temp_combine = filter_temp.rdd.coalesce(1)
#filter_temp_combine = filter_temp_combine.sortBy(lambda x: x[0],␣
 ↪ascending=False)
#filter_temp_combine.saveAsTextFile("BDA/output/l2_temptest")

#filter_perc_combine = filter_perc.rdd.coalesce(1)
#filter_perc_combine = filter_perc_combine.sortBy(lambda x: x[0],␣
 ↪ascending=False)
#filter_perc_combine.saveAsTextFile("BDA/output/l2_perctest")

#combine_temp_perc_combine = combine_temp_perc.rdd.coalesce(1)
#filter_temp_combine = combine_temp_perc_combine.sortBy(lambda x: x[0],␣
 ↪ascending=False)
#filter_temp_combine.saveAsTextFile("BDA/output/l2_combtest")
```

0.1.9 output of l2_perc_temp is empty, since no data meet the criteria