Problem Statement::

* Use retail\_db data set
* Problem Statement
  + Get daily revenue by product considering completed and closed orders.
  + Data need to be sorted in ascending order by date and then descending  
    order by revenue computed for each product for each day.
* Data for orders and order\_items is available in HDFS  
  /public/retail\_db/orders and /public/retail\_db/order\_items
* Data for products is available locally under /data/retail\_db/products
* Final output need to be stored under
  + HDFS location – avro format  
    /user/YOUR\_USER\_ID/daily\_revenue\_avro\_python
  + HDFS location – text format  
    /user/YOUR\_USER\_ID/daily\_revenue\_txt\_python
  + Local location /home/YOUR\_USER\_ID/daily\_revenue\_python
  + Solution need to be stored under  
    /home/YOUR\_USER\_ID/daily\_revenue\_python.txt

pyspark *--master yarn \*

*--conf spark.ui.port=12890 \*

*--num-executors 2 \*

*--executor-memory 512M \*

*--packages com.databricks:spark-avro\_2.10:2.0.1*

orders = sc.textFile("file:///F:/pySpark/data-master/data\_folder/retail\_db/orders")

>>> orderItems sc.textFile("file:///F:/pySpark/data-master/data\_folder/retail\_db/order\_items")

File "<stdin>", line 1

orderItems sc.textFile("file:///F:/pySpark/data-master/data\_folder/retail\_db/order\_items")

^

SyntaxError: invalid syntax

>>> orderItems = sc.textFile("file:///F:/pySpark/data-master/data\_folder/retail\_db/order\_items")

>>> for i in orders.take(10): print(i)

...

1,2013-07-25 00:00:00.0,11599,CLOSED

2,2013-07-25 00:00:00.0,256,PENDING\_PAYMENT

3,2013-07-25 00:00:00.0,12111,COMPLETE

4,2013-07-25 00:00:00.0,8827,CLOSED

5,2013-07-25 00:00:00.0,11318,COMPLETE

6,2013-07-25 00:00:00.0,7130,COMPLETE

7,2013-07-25 00:00:00.0,4530,COMPLETE

8,2013-07-25 00:00:00.0,2911,PROCESSING

9,2013-07-25 00:00:00.0,5657,PENDING\_PAYMENT

10,2013-07-25 00:00:00.0,5648,PENDING\_PAYMENT

>>> for i in orderItems.take(10): print(i)

...

1,1,957,1,299.98,299.98

2,2,1073,1,199.99,199.99

3,2,502,5,250.0,50.0

4,2,403,1,129.99,129.99

5,4,897,2,49.98,24.99

6,4,365,5,299.95,59.99

7,4,502,3,150.0,50.0

8,4,1014,4,199.92,49.98

9,5,957,1,299.98,299.98

10,5,365,5,299.95,59.99

>>> 20/08/26 13:09:47 WARN HeartbeatReceiver: Removing executor driver with no recent heartbeats: 757342 ms exceeds timeout 120000 ms

>>>

>>> orders.count()

68883

>>> orderItems.count()

172198

>>> for i in orders.map(lambda o: o.split(",")[3]).distinct().collect(): print(i)

...

CLOSED

CANCELED

PENDING\_PAYMENT

COMPLETE

PROCESSING

PAYMENT\_REVIEW

PENDING

ordersFiltered = orders.filter(lambda o: o.split(",")[3] in ["COMPLETE", "CLOSED"])

>>> for i in ordersFiltered.take(10): print(i)

...

1,2013-07-25 00:00:00.0,11599,CLOSED

3,2013-07-25 00:00:00.0,12111,COMPLETE

4,2013-07-25 00:00:00.0,8827,CLOSED

5,2013-07-25 00:00:00.0,11318,COMPLETE

6,2013-07-25 00:00:00.0,7130,COMPLETE

7,2013-07-25 00:00:00.0,4530,COMPLETE

12,2013-07-25 00:00:00.0,1837,CLOSED

15,2013-07-25 00:00:00.0,2568,COMPLETE

17,2013-07-25 00:00:00.0,2667,COMPLETE

18,2013-07-25 00:00:00.0,1205,CLOSED

ordersMap = ordersFiltered.map(lambda o: (int(o.split(",")[0]), o.split(",")[1]))

>>> orderItemsMap = orderItems.map(lambda oi: (int(oi.split(",")[1]), (int(oi.split(",")[2]) float(oi.split(","0[4]))))

File "<stdin>", line 1

orderItemsMap = orderItems.map(lambda oi: (int(oi.split(",")[1]), (int(oi.split(",")[2]) float(oi.split(","0[4]))))

^

SyntaxError: invalid syntax

>>> orderItemsMap = orderItems.map(lambda oi: (int(oi.split(",")[1]), (int(oi.split(",")[2]), float(oi.split(","0[4]))))

File "<stdin>", line 1

orderItemsMap = orderItems.map(lambda oi: (int(oi.split(",")[1]), (int(oi.split(",")[2]), float(oi.split(","0[4]))))

^

SyntaxError: invalid syntax

>>> orderItemsMap = orderItems.map(lambda oi: (int(oi.split(",")[1]), (int(oi.split(",")[2]), float(oi.split(",")[4]))))

>>> for i in ordersMap.TAKE(10): PRINT(I)

...

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'PipelinedRDD' object has no attribute 'TAKE'

>>> for i in ordersMap.take(10): print(i)

...

(1, '2013-07-25 00:00:00.0')

(3, '2013-07-25 00:00:00.0')

(4, '2013-07-25 00:00:00.0')

(5, '2013-07-25 00:00:00.0')

(6, '2013-07-25 00:00:00.0')

(7, '2013-07-25 00:00:00.0')

(12, '2013-07-25 00:00:00.0')

(15, '2013-07-25 00:00:00.0')

(17, '2013-07-25 00:00:00.0')

(18, '2013-07-25 00:00:00.0')

>>> for i in orderItemsMap.take(10): prin(i)

...

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'prin' is not defined

>>> for i in orderItemsMap.take(10): print(i)

...

(1, (957, 299.98))

(2, (1073, 199.99))

(2, (502, 250.0))

(2, (403, 129.99))

(4, (897, 49.98))

(4, (365, 299.95))

(4, (502, 150.0))

(4, (1014, 199.92))

(5, (957, 299.98))

(5, (365, 299.95))

orders orderItems ordersFiltered ordersMap orderItemsMap

>>> ordersjoin = ordersMap.join(orderIt

orderItems orderItemsMap

>>> ordersjoin = ordersMap.join(orderItemsMap)

>>> for i in ordersjoin.take(10): print(i)

...

(4, ('2013-07-25 00:00:00.0', (897, 49.98)))

(4, ('2013-07-25 00:00:00.0', (365, 299.95)))

(4, ('2013-07-25 00:00:00.0', (502, 150.0)))

(4, ('2013-07-25 00:00:00.0', (1014, 199.92)))

(12, ('2013-07-25 00:00:00.0', (957, 299.98)))

(12, ('2013-07-25 00:00:00.0', (134, 100.0)))

(12, ('2013-07-25 00:00:00.0', (1014, 149.94)))

(12, ('2013-07-25 00:00:00.0', (191, 499.95)))

(12, ('2013-07-25 00:00:00.0', (502, 250.0)))

(24, ('2013-07-25 00:00:00.0', (403, 129.99)))

>>> ordersjoinMap = ordersjoin.map(lambda oj: ((oj[1][0], oj[1][1][0]), oj[1][1][1]))

>>> for i in ordersjoinMap.take(10): print(i)

...

(('2013-07-25 00:00:00.0', 897), 49.98)

(('2013-07-25 00:00:00.0', 365), 299.95)

(('2013-07-25 00:00:00.0', 502), 150.0)

(('2013-07-25 00:00:00.0', 1014), 199.92)

(('2013-07-25 00:00:00.0', 957), 299.98)

(('2013-07-25 00:00:00.0', 134), 100.0)

(('2013-07-25 00:00:00.0', 1014), 149.94)

(('2013-07-25 00:00:00.0', 191), 499.95)

(('2013-07-25 00:00:00.0', 502), 250.0)

(('2013-07-25 00:00:00.0', 403), 129.99)

>>> from operator import add

>>> ordersjoinMap.reduceByKey(add)

PythonRDD[30] at RDD at PythonRDD.scala:52

>>> dailyRevenuePerProductId = ordersjoinMap.reduceByKey(add)

>>> for i in dailyRevenuePerProductId.take(10): print(i)

...

(('2013-07-25 00:00:00.0', 191), 5099.489999999999)

(('2013-07-25 00:00:00.0', 403), 1949.8500000000001)

(('2013-07-25 00:00:00.0', 627), 1079.73)

(('2013-07-25 00:00:00.0', 835), 63.98)

(('2013-07-26 00:00:00.0', 1004), 10799.460000000001)

(('2013-07-26 00:00:00.0', 792), 89.94)

(('2013-07-26 00:00:00.0', 828), 127.96)

(('2013-07-26 00:00:00.0', 116), 134.97)

(('2013-07-27 00:00:00.0', 828), 223.92999999999998)

(('2013-07-27 00:00:00.0', 1004), 9599.52)

from operator import add

>>> ordersjoinMap.reduceByKey(add)

PythonRDD[30] at RDD at PythonRDD.scala:52

>>> dailyRevenuePerProductId = ordersjoinMap.reduceByKey(add)

>>> for i in dailyRevenuePerProductId.take(10): print(i)

...

(('2013-07-25 00:00:00.0', 191), 5099.489999999999)

(('2013-07-25 00:00:00.0', 403), 1949.8500000000001)

(('2013-07-25 00:00:00.0', 627), 1079.73)

(('2013-07-25 00:00:00.0', 835), 63.98)

(('2013-07-26 00:00:00.0', 1004), 10799.460000000001)

(('2013-07-26 00:00:00.0', 792), 89.94)

(('2013-07-26 00:00:00.0', 828), 127.96)

(('2013-07-26 00:00:00.0', 116), 134.97)

(('2013-07-27 00:00:00.0', 828), 223.92999999999998)

(('2013-07-27 00:00:00.0', 1004), 9599.52)

Load products data and convert into RDD

productsRaw = open("F:/pySpark/data-master/data\_folder/retail\_db/products/part-00000").read().splitlines()

products = sc.parallelize(productsRaw)

for i in products.take(10): print(i)

...

1,2,Quest Q64 10 FT. x 10 FT. Slant Leg Instant U,,59.98,http://images.acmesports.sports/Quest+Q64+10+FT.+x+10+FT.+Slant+Leg+Instant+Up+Canopy

2,2,Under Armour Men's Highlight MC Football Clea,,129.99,http://images.acmesports.sports/Under+Armour+Men%27s+Highlight+MC+Football+Cleat

3,2,Under Armour Men's Renegade D Mid Football Cl,,89.99,http://images.acmesports.sports/Under+Armour+Men%27s+Renegade+D+Mid+Football+Cleat

4,2,Under Armour Men's Renegade D Mid Football Cl,,89.99,http://images.acmesports.sports/Under+Armour+Men%27s+Renegade+D+Mid+Football+Cleat

5,2,Riddell Youth Revolution Speed Custom Footbal,,199.99,http://images.acmesports.sports/Riddell+Youth+Revolution+Speed+Custom+Football+Helmet

6,2,Jordan Men's VI Retro TD Football Cleat,,134.99,http://images.acmesports.sports/Jordan+Men%27s+VI+Retro+TD+Football+Cleat

7,2,Schutt Youth Recruit Hybrid Custom Football H,,99.99,http://images.acmesports.sports/Schutt+Youth+Recruit+Hybrid+Custom+Football+Helmet+2014

8,2,Nike Men's Vapor Carbon Elite TD Football Cle,,129.99,http://images.acmesports.sports/Nike+Men%27s+Vapor+Carbon+Elite+TD+Football+Cleat

9,2,Nike Adult Vapor Jet 3.0 Receiver Gloves,,50.0,http://images.acmesports.sports/Nike+Adult+Vapor+Jet+3.0+Receiver+Gloves

10,2,Under Armour Men's Highlight MC Football Clea,,129.99,http://images.acmesports.sports/Under+Armour+Men%27s+Highlight+MC+Football+Cleat

for i in productsMap.take(10): print(i)

...

(1, 'Quest Q64 10 FT. x 10 FT. Slant Leg Instant U')

(2, "Under Armour Men's Highlight MC Football Clea")

(3, "Under Armour Men's Renegade D Mid Football Cl")

(4, "Under Armour Men's Renegade D Mid Football Cl")

(5, 'Riddell Youth Revolution Speed Custom Footbal')

(6, "Jordan Men's VI Retro TD Football Cleat")

(7, 'Schutt Youth Recruit Hybrid Custom Football H')

(8, "Nike Men's Vapor Carbon Elite TD Football Cle")

(9, 'Nike Adult Vapor Jet 3.0 Receiver Gloves')

(10, "Under Armour Men's Highlight MC Football Clea")

dailyRevenuePerProductIdMap = dailyRevenuePerProductId.map(lambda d: (d[0][1], (d[0][0], d[1])))

>>> for i in daily

dailyRevenuePerProductId dailyRevenuePerProductIdMap

>>> for i in dailyRevenuePerProductIdMap.take(10): print(i)

...

(191, ('2013-07-25 00:00:00.0', 5099.489999999999))

(403, ('2013-07-25 00:00:00.0', 1949.8500000000001))

(627, ('2013-07-25 00:00:00.0', 1079.73))

(835, ('2013-07-25 00:00:00.0', 63.98))

(1004, ('2013-07-26 00:00:00.0', 10799.460000000001))

(792, ('2013-07-26 00:00:00.0', 89.94))

(828, ('2013-07-26 00:00:00.0', 127.96))

(116, ('2013-07-26 00:00:00.0', 134.97))

(828, ('2013-07-27 00:00:00.0', 223.92999999999998))

(1004, ('2013-07-27 00:00:00.0', 9599.52))

Join and sort data

dailyRevenuePerProductjoin = dailyRevenuePerProductIdMap.join(productsMap)

>>> for i in daily

dailyRevenuePerProductId dailyRevenuePerProductIdMap dailyRevenuePerProductjoin

>>> for i in dailyRevenuePerProductjoin.take(10): print(i)

...

(792, (('2013-07-26 00:00:00.0', 89.94), "Hirzl Men's Hybrid Golf Glove"))

(792, (('2013-08-25 00:00:00.0', 74.95), "Hirzl Men's Hybrid Golf Glove"))

(792, (('2013-09-19 00:00:00.0', 44.97), "Hirzl Men's Hybrid Golf Glove"))

(792, (('2013-10-13 00:00:00.0', 104.93), "Hirzl Men's Hybrid Golf Glove"))

(792, (('2013-10-31 00:00:00.0', 59.96), "Hirzl Men's Hybrid Golf Glove"))

(792, (('2013-11-06 00:00:00.0', 104.93), "Hirzl Men's Hybrid Golf Glove"))

(792, (('2013-11-09 00:00:00.0', 74.95), "Hirzl Men's Hybrid Golf Glove"))

(792, (('2013-11-28 00:00:00.0', 74.95), "Hirzl Men's Hybrid Golf Glove"))

(792, (('2013-12-25 00:00:00.0', 29.98), "Hirzl Men's Hybrid Golf Glove"))

(792, (('2013-12-28 00:00:00.0', 14.99), "Hirzl Men's Hybrid Golf Glove"))

dailyRevenuePerProduct = dailyRevenuePerProductjoin.map(lambda d: ((d[1][0][0], -d[1][0][1]), d[1][0][0] + "," + str(d[1][0][1]) + "," + d[1][1]))

>>> for i in dailyRevenuePerProduct.take(10): print(i)

...

(('2013-07-26 00:00:00.0', -89.94), "2013-07-26 00:00:00.0,89.94,Hirzl Men's Hybrid Golf Glove")

(('2013-08-25 00:00:00.0', -74.95), "2013-08-25 00:00:00.0,74.95,Hirzl Men's Hybrid Golf Glove")

(('2013-09-19 00:00:00.0', -44.97), "2013-09-19 00:00:00.0,44.97,Hirzl Men's Hybrid Golf Glove")

(('2013-10-13 00:00:00.0', -104.93), "2013-10-13 00:00:00.0,104.93,Hirzl Men's Hybrid Golf Glove")

(('2013-10-31 00:00:00.0', -59.96), "2013-10-31 00:00:00.0,59.96,Hirzl Men's Hybrid Golf Glove")

(('2013-11-06 00:00:00.0', -104.93), "2013-11-06 00:00:00.0,104.93,Hirzl Men's Hybrid Golf Glove")

(('2013-11-09 00:00:00.0', -74.95), "2013-11-09 00:00:00.0,74.95,Hirzl Men's Hybrid Golf Glove")

(('2013-11-28 00:00:00.0', -74.95), "2013-11-28 00:00:00.0,74.95,Hirzl Men's Hybrid Golf Glove")

(('2013-12-25 00:00:00.0', -29.98), "2013-12-25 00:00:00.0,29.98,Hirzl Men's Hybrid Golf Glove")

(('2013-12-28 00:00:00.0', -14.99), "2013-12-28 00:00:00.0,14.99,Hirzl Men's Hybrid Golf Glove")

Sort

dailyRevenuePerProductId dailyRevenuePerProductIdMap dailyRevenuePerProductjoin dailyRevenuePerProduct

>>> dailyRevenuePerProductSorted = dailyRevenuePerProduct.sortByKey()

>>> for i in dail

dailyRevenuePerProductId dailyRevenuePerProductIdMap dailyRevenuePerProductjoin dailyRevenuePerProduct dailyRevenuePerProductSorted

>>> for i in dailyRevenuePerProductSorted.take(i): print(i)

...

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "C:\spark-2.3.4-bin-hadoop2.6\python\pyspark\rdd.py", line 1328, in take

while len(items) < num and partsScanned < totalParts:

TypeError: '<' not supported between instances of 'int' and 'tuple'

>>> for i in dailyRevenuePerProductSorted.take(10): print(i)

...

(('2013-07-25 00:00:00.0', -5599.72), '2013-07-25 00:00:00.0,5599.72,Field & Stream Sportsman 16 Gun Fire Safe')

(('2013-07-25 00:00:00.0', -5099.489999999999), "2013-07-25 00:00:00.0,5099.489999999999,Nike Men's Free 5.0+ Running Shoe")

(('2013-07-25 00:00:00.0', -4499.700000000001), "2013-07-25 00:00:00.0,4499.700000000001,Diamondback Women's Serene Classic Comfort Bi")

(('2013-07-25 00:00:00.0', -3359.44), '2013-07-25 00:00:00.0,3359.44,Perfect Fitness Perfect Rip Deck')

(('2013-07-25 00:00:00.0', -2999.8500000000004), '2013-07-25 00:00:00.0,2999.8500000000004,Pelican Sunstream 100 Kayak')

(('2013-07-25 00:00:00.0', -2798.88), "2013-07-25 00:00:00.0,2798.88,O'Brien Men's Neoprene Life Vest")

(('2013-07-25 00:00:00.0', -1949.8500000000001), "2013-07-25 00:00:00.0,1949.8500000000001,Nike Men's CJ Elite 2 TD Football Cleat")

(('2013-07-25 00:00:00.0', -1650.0), "2013-07-25 00:00:00.0,1650.0,Nike Men's Dri-FIT Victory Golf Polo")

(('2013-07-25 00:00:00.0', -1079.73), "2013-07-25 00:00:00.0,1079.73,Under Armour Girls' Toddler Spine Surge Runni")

(('2013-07-25 00:00:00.0', -599.99), '2013-07-25 00:00:00.0,599.99,Bowflex SelectTech 1090 Dumbbells')

>>> daily

dailyRevenuePerProductId dailyRevenuePerProductIdMap dailyRevenuePerProductjoin dailyRevenuePerProduct dailyRevenuePerProductSorted

>>> dailyRevenuePerProductName = daily

dailyRevenuePerProductId dailyRevenuePerProductIdMap dailyRevenuePerProductjoin dailyRevenuePerProduct dailyRevenuePerProductSorted

>>> dailyRevenuePerProductName = dailyRevenuePerProductSorted.map(lambda i: i[1])

>>> for i in daily

dailyRevenuePerProductId dailyRevenuePerProductIdMap dailyRevenuePerProductjoin dailyRevenuePerProduct dailyRevenuePerProductSorted dailyRevenuePerProductName

>>> for i in dailyRevenuePerProductName.take(10): print(i)

...

2013-07-25 00:00:00.0,5599.72,Field & Stream Sportsman 16 Gun Fire Safe

2013-07-25 00:00:00.0,5099.489999999999,Nike Men's Free 5.0+ Running Shoe

2013-07-25 00:00:00.0,4499.700000000001,Diamondback Women's Serene Classic Comfort Bi

2013-07-25 00:00:00.0,3359.44,Perfect Fitness Perfect Rip Deck

2013-07-25 00:00:00.0,2999.8500000000004,Pelican Sunstream 100 Kayak

2013-07-25 00:00:00.0,2798.88,O'Brien Men's Neoprene Life Vest

2013-07-25 00:00:00.0,1949.8500000000001,Nike Men's CJ Elite 2 TD Football Cleat

2013-07-25 00:00:00.0,1650.0,Nike Men's Dri-FIT Victory Golf Polo

2013-07-25 00:00:00.0,1079.73,Under Armour Girls' Toddler Spine Surge Runni

2013-07-25 00:00:00.0,599.99,Bowflex SelectTech 1090 Dumbbells

>>>

dailyRevenuePerProductName.saveAsTextFile("file:///F:/pySpark/data-master/data\_folder/retail\_db/dailyReveneuperProductName")

dailyRevenuePerProductName.coalesce(2).saveAsTextFile("file:///F:/pySpark/data-master/data\_folder/retail\_db/dailyReveneuperProductName2")

Saving file as Avro

dailyRevenuePerProduct = dailyRevenuePerProductjoin.map(lambda d: ((d[1][0][0], -d[1][0][1]), (d[1][0][0], round(d[1][0][1]), d[1][1])))

>>> for i in dailyRevenuePerProduct.take(10): print(i)

...

(('2013-07-26 00:00:00.0', -89.94), ('2013-07-26 00:00:00.0', 90, "Hirzl Men's Hybrid Golf Glove"))

(('2013-08-25 00:00:00.0', -74.95), ('2013-08-25 00:00:00.0', 75, "Hirzl Men's Hybrid Golf Glove"))

(('2013-09-19 00:00:00.0', -44.97), ('2013-09-19 00:00:00.0', 45, "Hirzl Men's Hybrid Golf Glove"))

(('2013-10-13 00:00:00.0', -104.93), ('2013-10-13 00:00:00.0', 105, "Hirzl Men's Hybrid Golf Glove"))

(('2013-10-31 00:00:00.0', -59.96), ('2013-10-31 00:00:00.0', 60, "Hirzl Men's Hybrid Golf Glove"))

(('2013-11-06 00:00:00.0', -104.93), ('2013-11-06 00:00:00.0', 105, "Hirzl Men's Hybrid Golf Glove"))

(('2013-11-09 00:00:00.0', -74.95), ('2013-11-09 00:00:00.0', 75, "Hirzl Men's Hybrid Golf Glove"))

(('2013-11-28 00:00:00.0', -74.95), ('2013-11-28 00:00:00.0', 75, "Hirzl Men's Hybrid Golf Glove"))

(('2013-12-25 00:00:00.0', -29.98), ('2013-12-25 00:00:00.0', 30, "Hirzl Men's Hybrid Golf Glove"))

(('2013-12-28 00:00:00.0', -14.99), ('2013-12-28 00:00:00.0', 15, "Hirzl Men's Hybrid Golf Glove"))

>>>

dailyRevenuePerProductSorted = dailyRevenuePerProduct.sortByKey()

>>> for i in dailyRevenuePerProductSorted.take(10): print(i)

...

(('2013-07-25 00:00:00.0', -5599.72), ('2013-07-25 00:00:00.0', 5600, 'Field & Stream Sportsman 16 Gun Fire Safe'))

(('2013-07-25 00:00:00.0', -5099.489999999999), ('2013-07-25 00:00:00.0', 5099, "Nike Men's Free 5.0+ Running Shoe"))

(('2013-07-25 00:00:00.0', -4499.700000000001), ('2013-07-25 00:00:00.0', 4500, "Diamondback Women's Serene Classic Comfort Bi"))

(('2013-07-25 00:00:00.0', -3359.44), ('2013-07-25 00:00:00.0', 3359, 'Perfect Fitness Perfect Rip Deck'))

(('2013-07-25 00:00:00.0', -2999.8500000000004), ('2013-07-25 00:00:00.0', 3000, 'Pelican Sunstream 100 Kayak'))

(('2013-07-25 00:00:00.0', -2798.88), ('2013-07-25 00:00:00.0', 2799, "O'Brien Men's Neoprene Life Vest"))

(('2013-07-25 00:00:00.0', -1949.8500000000001), ('2013-07-25 00:00:00.0', 1950, "Nike Men's CJ Elite 2 TD Football Cleat"))

(('2013-07-25 00:00:00.0', -1650.0), ('2013-07-25 00:00:00.0', 1650, "Nike Men's Dri-FIT Victory Golf Polo"))

(('2013-07-25 00:00:00.0', -1079.73), ('2013-07-25 00:00:00.0', 1080, "Under Armour Girls' Toddler Spine Surge Runni"))

(('2013-07-25 00:00:00.0', -599.99), ('2013-07-25 00:00:00.0', 600, 'Bowflex SelectTech 1090 Dumbbells'))

>>> dailyRevenuePerProductName = dailyRevenuePerProductSorted.map(lambda i: i[1])

>>> for i in dailyRevenuePerProductName.take(10): print(i)

...

('2013-07-25 00:00:00.0', 5600, 'Field & Stream Sportsman 16 Gun Fire Safe')

('2013-07-25 00:00:00.0', 5099, "Nike Men's Free 5.0+ Running Shoe")

('2013-07-25 00:00:00.0', 4500, "Diamondback Women's Serene Classic Comfort Bi")

('2013-07-25 00:00:00.0', 3359, 'Perfect Fitness Perfect Rip Deck')

('2013-07-25 00:00:00.0', 3000, 'Pelican Sunstream 100 Kayak')

('2013-07-25 00:00:00.0', 2799, "O'Brien Men's Neoprene Life Vest")

('2013-07-25 00:00:00.0', 1950, "Nike Men's CJ Elite 2 TD Football Cleat")

('2013-07-25 00:00:00.0', 1650, "Nike Men's Dri-FIT Victory Golf Polo")

('2013-07-25 00:00:00.0', 1080, "Under Armour Girls' Toddler Spine Surge Runni")

('2013-07-25 00:00:00.0', 600, 'Bowflex SelectTech 1090 Dumbbells')

>>>

dailyRevenuePerProductNameDF = dailyRevenuePerProductName.coalesce(2).toDF(schema=["order\_date", "revenue\_per\_product", "ProductName"])

20/08/28 13:01:27 WARN ObjectStore: Failed to get database global\_temp, returning NoSuchObjectException

dailyRevenuePerProductNameDF.show()

<https://stackoverflow.com/questions/33878433/spark-write-avro-file>

As part of this topic, we will get the data to the local file system

hadoop fs -get /user/dgadiraju/daily\_revenue\_txt\_python /home/dgadiraju/daily\_revenue\_python

hadoop fs -get /user/dgadiraju/daily\_revenue\_avro\_python /home/dgadiraju/daily\_revenue\_python

* This need not be relevant for the certification – but must to know
* Create a directory for the application
* Create src/main/python
* Create program file with py extension
* Ship the code to the cluster
* Run on the cluster using spark-submit