//Executed code in Scala

spark-shell

/\*

creating book table by loading file saved on hdfs

\*/

val bookRDD = sc.textFile("/cloud/greesh/book") // loading text file

val bookDF = bookRDD.map(book => { (book.split("\t")(0),book.split("\t")(1))}).toDF("isbn","name") // loading text in file into spark Data frame

bookDF.show() // displays file in form of data frame

bookDF.registerTempTable("book") //creating temporary table book

sqlContext.sql("select \* from book").show()

/\*

creating purchase table by loading file saved on hdfs

\*/

val purchaseRDD = sc.textFile("/cloud/greesh/purchase") // loading text file

val purchaseDF = purchaseRDD.map(purchase => { (purchase.split("\t")(0).toInt,book.split("\t")(1),purchase.split("\t")(2),purchase.split("\t")(3).purchase.split("\t")(4).toInt)}).toDF("year","cid","isbn","seller","price")

// loading text in file into spark Data frame

purchaseDF.show()

purchaseDF.registerTempTable("purchase") //creating temporary table book

sqlContext.sql("select \* from purchase").show()

val seller =sqlContext.sql("select isbn,cid,seller,price from purchase where seller = 'Amazon' order by isbn") //filtering out other sellers as we need only books sold by amazon

seller.registerTempTable("seller")

sqlContext.sql("select seller from seller").show()

val price =sqlContext.sql("select isbn,min(price) as price from purchase group by isbn order by price") //getting lowstest price for each isbn

price.registerTempTable("price")

sqlContext.sql("select price from price").show()

val lp = sqlContext.sql("select p.isbn,s.seller,p.price from price p left join seller s on p.price=s.price where seller is not null) // getting isbn of book's sold by amazon for lowest price

lp.registerTempTable("lp")

val eliminate=sqlContext.sql("select isbn,seller,price from purchase where seller !='Amazon' order by isbn) // we have both amazon and borders selling b1 for 90 so to eliminate b1 from list ,adding other seller's who sold books for lowest price list in a table

eliminate.registerTempTable("eliminate")

val equalremoving=sqlContext.sql("select lp.isbn,lp.seller,lp.price from lp lp left outer join eliminate e on e.price =lp.price where e.isbn is null) // removed b1 from the final list

equalremoving.registerTempTable("equalremoving")

val bookname=sqlContext.sql("select distinct b.name,b.isbn from book b join equalremoving e on e.isbn=b.isbn") // getting book details from table book

bookname.registerTempTable("bookname")

sqlContext.sql("select \* from bookname").show() // displaying final books that amazon sold for lowest price compared to other sellers.