**Business Requirements**

1. **Using Spark Core (*15 Marks*)**
2. Determine the total revenue earned by the e-commerce company, taking into account the price of products purchased and subtracting any applicable discounts. (*2 marks*)

val rdd1 = sc.textFile("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Customer Purchase History.csv")

val header1 = rdd1.first()

val rdd2 = rdd1.filter(r=>r!=header1).map(r=>(r.split(",")))

val rdd3 = sc.textFile("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Product Catalog.csv")

val header2 = rdd3.first()

val rdd4 = rdd3.filter(r=>r!=header2).map(r=>(r.split(",")))

case class Customer\_Purchase\_history (Customer\_ID:String, Purchase\_Date:String, Product\_ID:String, Discount\_Applied:Float)

case class Product\_Catalog(Product\_ID:String, Product\_Name:String, Product\_Category:String, Price:Float, Rating:Float)

val CusRDD = rdd2.map(r=>(r(2), Customer\_Purchase\_history(r(0), r(1), r(2), r(3).toFloat)))

val ProdRDD = rdd4.map(r=>(r(0), Product\_Catalog(r(0), r(1), r(2), r(3).toFloat, r(4).toFloat)))

val rdd5 = CusRDD.join(ProdRDD)

val Total\_Rev = rdd5.map(r=>(r.\_2.\_2.Price - (r.\_2.\_1.Discount\_Applied\*r.\_2.\_2.Price)/100)).reduce(\_+\_)

print("Total Revenue earned by the e-commerce company is " + Total\_Rev)

**Output:**



1. Compute the total discount given by the e-commerce company by summing up the Discount Applied field in the Customer Purchase History dataset. (*2 marks*)

val Discount = rdd5.map(r=>(r.\_2.\_1.Discount\_Applied\*r.\_2.\_2.Price)/100).reduce(\_+\_)

print("Discount given by e-commerce company is "+ Discount)

**Output:**



1. Perform an analysis to find the average purchase amount (excluding discounts) for each category of products by joining the customer purchase history and product catalog dataset. (*2 marks*)

val rdd5 = CusRDD.join(ProdRDD)

val rdd6 = rdd5.map(r=>(r.\_2.\_2.Product\_Category,1)).reduceByKey(\_+\_)

rdd6.foreach(println)

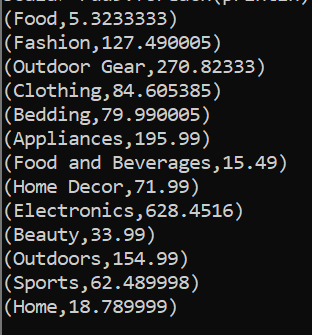
val rdd7 = rdd5.map(r=>(r.\_2.\_2.Product\_Category,r.\_2.\_2.Price)).reduceByKey(\_+\_)

val rdd8 = rdd6.join(rdd7)

val rdd9 = rdd8.map(r=>(r.\_1, r.\_2.\_2/r.\_2.\_1))

rdd9.foreach(println)

**Output:**



1. Identify the days with the highest bounce rate in the Website Traffic and User Behavior dataset. (*2 marks*)

val rdd10 = sc.textFile("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Website traffic and user behavior.csv")

val header3 = rdd10.first()

val rdd11 = rdd10.filter(r=>r!=header3).map(r=>(r.split(",")))

rdd11.collect()

val rdd12 = rdd11.map(r=>(r(0),(r(3).toInt\*r(1).toInt\*0.01, r(1).toInt)))

val rdd12\_1 = rdd12.reduceByKey((a,b) => (a.\_1 + b.\_1, a.\_2 + b.\_2)).map(r=>(r.\_1, (r.\_2.\_1/r.\_2.\_2\*100).round))

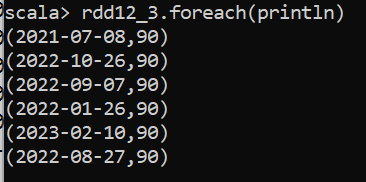
val rdd12\_2 = rdd12\_1.sortBy(r=>r.\_2,false,numPartitions=1)

val highest = rdd12\_2.first()

val rdd12\_3 = rdd12\_1.filter(r=> (r.\_2==highest.\_2))

rdd12\_3.foreach(println)

**Output:**



1. Calculate the average rating for each category in the Product Catalog dataset. (*2 marks*)

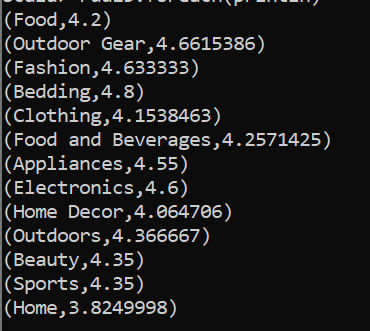
val rdd13 = rdd4.map(r=>(r(2),(r(4).toFloat, 1)))

val rdd14 = rdd13.reduceByKey{ case((a1,b1), (a2,b2)) =>(a1+a2, b1+b2)}

val rdd15 = rdd14.mapValues{ case(sum,count) =>(sum.toFloat/count)}

rdd15.foreach(println)

**Output:**



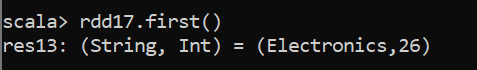
1. Determine the most popular product category based on the number of purchases in the customer purchase history dataset. (*2 marks*)

val rdd16 = rdd5.map(r=>(r.\_2.\_2.Product\_Category, 1)).reduceByKey(\_+\_)

val rdd17 = rdd16.sortBy(r=>r.\_2, false, numPartitions = 1)

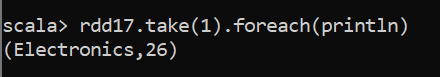
rdd17.first()

**Output:**



rdd17.take(1).foreach(println)

**Output:**

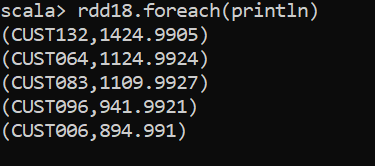


1. Identify the top 5 customers with the highest purchase amount by joining the customer purchase history and product catalog dataset. (*3 marks*)

val rdd18 = rdd5.map(r=>(r.\_2.\_1.Customer\_ID, r.\_2.\_2.Price - (r.\_2.\_1.Discount\_Applied\*r.\_2.\_2.Price)/100)).reduceByKey(\_+\_).sortBy(r=>r.\_2, false, numPartitions=1).take(5)

rdd18.foreach(println)

**Output:**



1. **Using Spark SQL (*15 Marks*)**
2. Find the product categories with the highest and lowest average rating in the product catalog dataset. (*3 marks*)

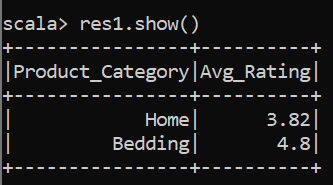
val df1 = spark.read.option("inferSchema", "true").option("header", "true").csv("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Product Catalog.csv").toDF("Product\_ID", "Product\_Name", "Product\_Category", "Price", "Rating")

df1.createOrReplaceTempView("Product\_Catalog");

val res1 = spark.sql("select Product\_Category, round(avg(Rating),2) as Avg\_Rating from Product\_Catalog group by Product\_Category having avg(Rating) == (select avg(Rating) from Product\_Catalog group by Product\_Category order by avg(Rating) limit 1) or avg(Rating) == (select avg(Rating) from Product\_Catalog group by Product\_Category order by avg(Rating) desc limit 1)")

res1.show()

**Output:**



1. Identify the customers who have made a purchase with a discount lesser than or equal to 10 in the customer purchase history dataset. (*2 marks*)

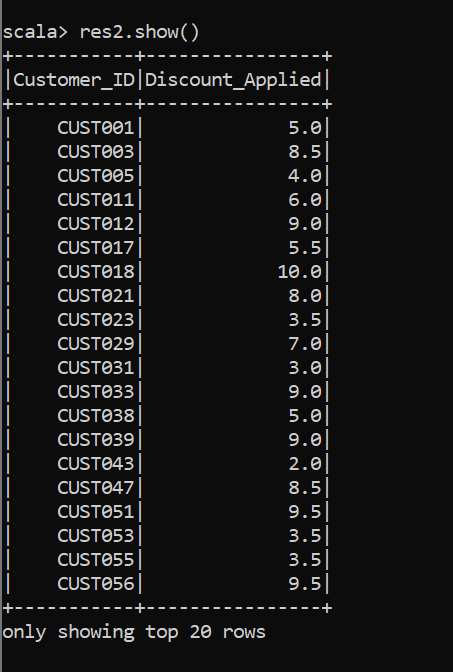
val df2 = spark.read.option("inferSchema", "true").option("header", "true").csv("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Customer Purchase History.csv").toDF("Customer\_ID", "Purchase\_Date", "Product\_ID", "Discount\_Applied")

df2.createOrReplaceTempView("Customer\_Pur")

val res2 = spark.sql("select Customer\_ID, Discount\_Applied from Customer\_Pur where Discount\_Applied <= 10")

res2.show()

**Output:**



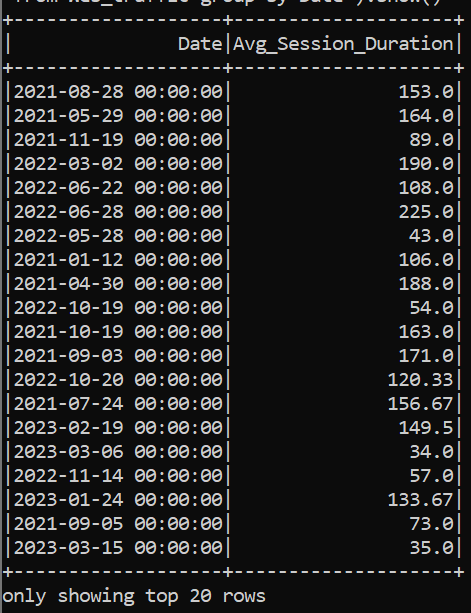
1. Determine the average session duration for each day in the website traffic and user behavior dataset. (*2 marks*)

val df3 = spark.read.option("header","true").option("inferSchema","true").csv("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Website traffic and user behavior.csv").toDF("Date","PageViews","UniqueVisitors","BounceRate","SessionDuration")

df3.createOrReplaceTempView("Web\_traffic")

spark.sql("select Date, round(avg(SessionDuration),2) as Avg\_Session\_Duration from Web\_traffic group by Date").show()

**Output:**



1. Find the product with the highest rating in each category in the product catalog dataset. (*2 marks*)

val rdd4 = spark.sql("select p.Product\_ID, p.Product\_Category, p.Product\_Name, p.Rating from Product\_Catalog p where p.Rating = (select max(prod.Rating) from Product\_Catalog prod where p.Product\_Category = prod.Product\_Category)")

rdd4.show()

**Output:**

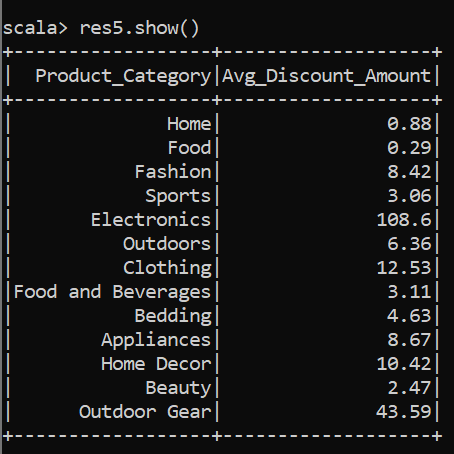


1. Determine the average discount amount applied on each product category by joining the customer purchase history and product catalog dataset. (*2 marks*)

val res5 = spark.sql("select p.Product\_Category, round(avg((c.Discount\_Applied\*p.Price)/100),2) as Avg\_Discount\_Amount from Customer\_Pur c join Product\_Catalog p on c.Product\_ID == p.Product\_ID group by p.Product\_Category")

res5.show()

**Output:**

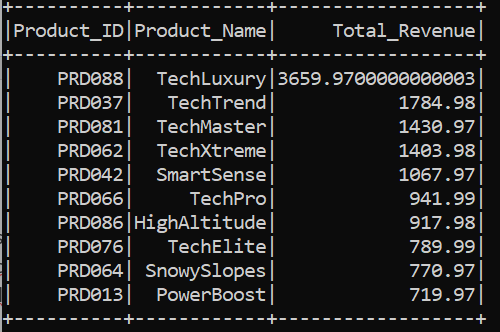


1. Identify the top 10 products with the highest total revenue generated by joining the customer purchase history and product catalog dataset. (*2 marks*)

val res6 = spark.sql("select p.Product\_ID, p.Product\_Name, sum(round(p.Price - (c.Discount\_Applied\*p.Price)/100, 2)) as Total\_Revenue from Customer\_Pur c join Product\_Catalog p on c.Product\_ID == p.Product\_ID group by p.Product\_ID, p.Product\_Name order by Total\_Revenue desc limit 10")

res6.show()

**Output:**

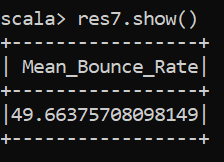


1. Determine the mean bounce rate for the month of March, 2022. (*2 marks*)

val res7 = spark.sql("select sum(BounceRate \* PageViews / 100)/sum(PageViews) \* 100 as Mean\_Bounce\_Rate from Web\_traffic where Date like '2022-03-%'")

res7.show()

Output:



1. **Using PySpark Core (*10 Marks*)**
2. Calculate the average session duration in seconds for each month in the year 2021 in the website traffic and user behavior dataset. (*2 marks*)

rdd1 = sc.textFile("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Website traffic and user behavior.csv")

header = rdd1.first()

rdd2 = rdd1.filter(lambda r: r!=header).map(lambda r:r.split(","))

rdd3 = rdd2.filter(lambda r: r[0].split("-")[0]=='2021').map(lambda r:(r[0].split("-")[1], (int(r[4]),1)))

rdd3.collect()

rdd4 = rdd3.reduceByKey(lambda a,b : (a[0]+b[0], a[1]+b[1]))

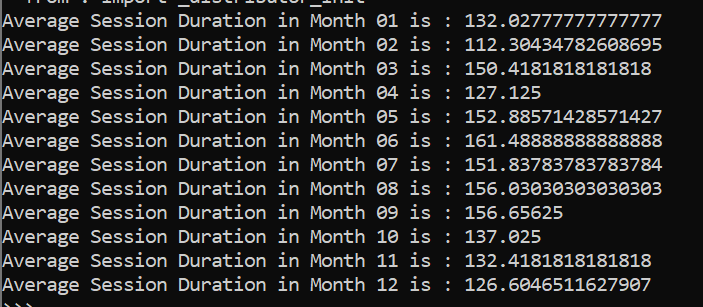
rdd5 = rdd4.mapValues(lambda a: (a[0]/a[1]))

rdd6 = rdd5.sortBy(lambda r: r[0], ascending=True)

for word in rdd6.collect():

print("Average Session Duration in Month "+ word[0] + " is : " + str(word[1]))

**Output:**



1. Find the date with the highest number of unique visitors from the website traffic and user behavior dataset. (*2 marks*)

word= rdd2.map(lambda r: (r[0], int(r[2]))).reduceByKey(lambda a,b : a+ b).sortBy(lambda r: r[1], False, numPartitions = 1).first()

print("Date is "+ word[0] + " and highest number of unique visitors is : "+str(word[1]))

**Output:**



1. Calculate the total number of page views, unique visitors, bounce rate, and session duration for each date from the website traffic and user behavior dataset. (2 marks)

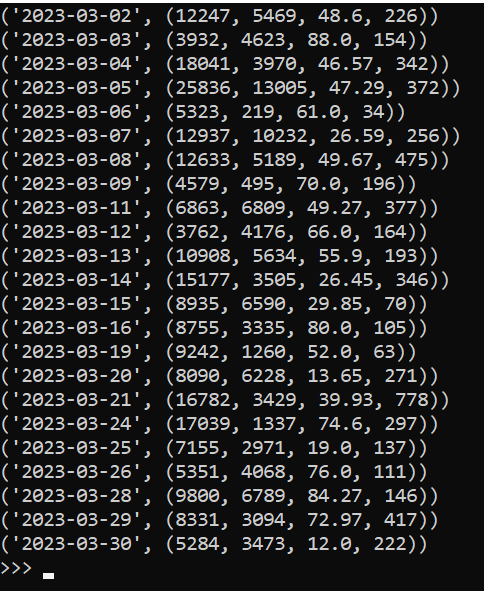
rdd8 = rdd2.map(lambda r: (r[0], (int(r[1]), int(r[2]), int(r[3])\*int(r[1])/100, int(r[4]))))

rdd9 = rdd8.reduceByKey(lambda a, b: (a[0]+b[0], a[1]+b[1], a[2]+b[2], a[3]+b[3])).sortBy(lambda r: r[0], ascending=True)

rdd9\_1 = rdd9.map(lambda r: (r[0], (r[1][0], r[1][1], round((r[1][2]/r[1][0])\*100,2), r[1][3])))

for word in rdd9\_1.collect():

print(word)



1. Filter the customer purchase history dataset to include only the purchases made on June 13, 2020. (*2 marks*)

rdd10 = sc.textFile("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Customer Purchase History.csv")

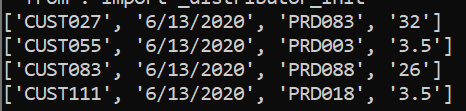
rdd11 = rdd10.map(lambda r: r.split(","))

rdd12 = rdd11.filter(lambda r:r[1]=='6/13/2020')

for word in rdd12.collect():

print(word)

**Output:**



1. Filter the product catalog dataset to include only products in the “Electronics” category and sort by price in ascending order. (*2 marks*)

rdd11 = sc.textFile("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Product Catalog.csv")

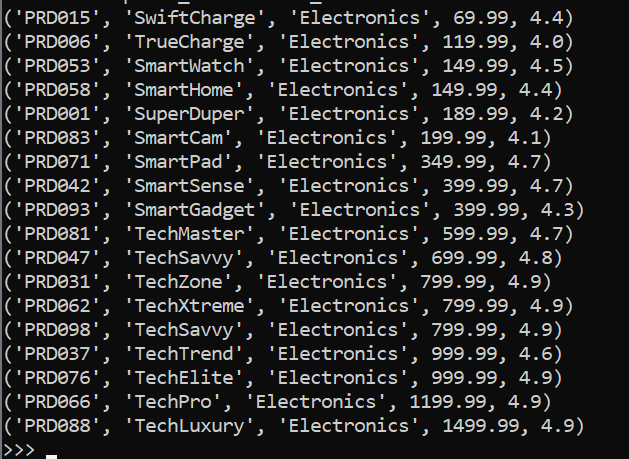
rdd12 = rdd11.map(lambda r:r.split(","))

rdd13 = rdd12.filter(lambda r:r[2]=="Electronics").map(lambda r: (r[0], r[1], r[2], float(r[3]), float(r[4]))).sortBy(lambda r: r[3], ascending=True,numPartitions=1)

for word in rdd13.collect():

print(word)

**Output:**



1. **Using PySpark SQL (*15 Marks*)**
2. Join the customer purchase history and product catalog datasets to create a new dataset with columns: Customer ID, Purchase Date, Product Name, Category, Amount Paid (Price after applying discount). (*3 marks*)

df1 = spark.read.csv("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Customer Purchase History.csv", inferSchema=True, header=True).toDF("Customer\_ID", "Purchase\_Date", "Product\_ID", "Discount\_Applied")

df1.createOrReplaceTempView("Customer\_Pur")

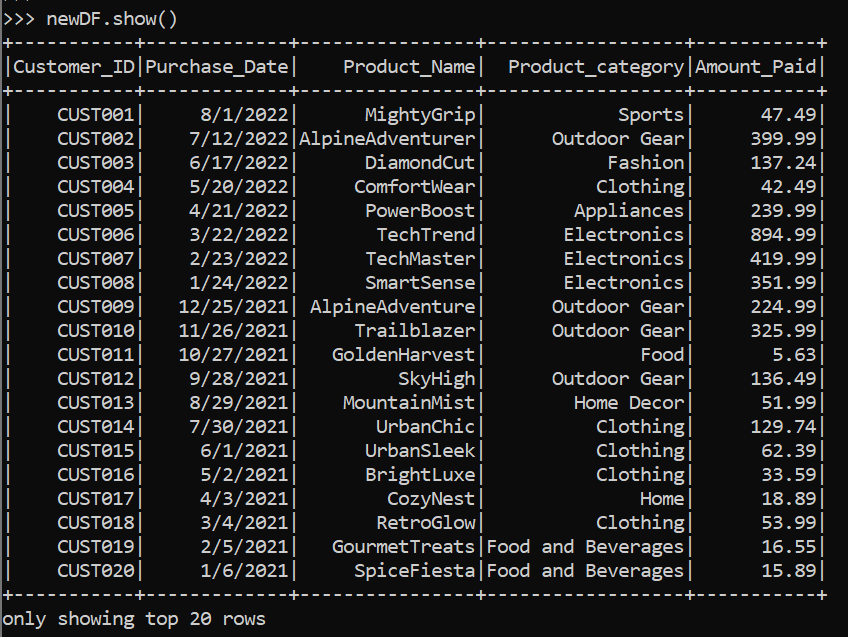
df2 = spark.read.csv("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Product Catalog.csv", inferSchema=True, header=True).toDF("Product\_ID", "Product\_Name", "Product\_Category", "Price", "Rating")

df2.createOrReplaceTempView("Product\_Catalog")

newDF = spark.sql("select c.Customer\_ID, c.Purchase\_Date, p.Product\_Name, p.Product\_category, round((p.Price-(p.Price\*c.Discount\_Applied)/100),2) as Amount\_Paid from Customer\_Pur c join Product\_Catalog p on c.Product\_ID == p.Product\_ID")

newDF.show()

**Output:**



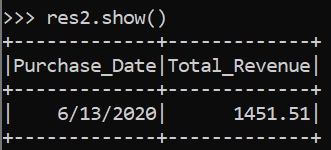
newDF.write.csv("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Solution\_1\_Pyspark-sql.csv")

1. Join the customer purchase history and product catalog datasets and filter to only include purchases made on June 13, 2020 to find the total revenue generated for that day. (*3 marks*)

res2 = spark.sql("select c.Purchase\_Date, sum(round((p.Price-(p.Price\*c.Discount\_Applied)/100),2)) as Total\_Revenue from Customer\_Pur c join Product\_Catalog p on c.Product\_ID == p.Product\_ID where c.Purchase\_Date == '6/13/2020' group by c.Purchase\_Date")

res2.show()

**Output:**



1. Join the customer purchase history and product catalog datasets and filter to include only purchases made by customers with a unique visitor count of 250 or more from the website traffic and user behavior dataset. Then, group the dataset by product name and find the total revenue generated for each product. (*4 marks*)

from collections import namedtuple

RDD1 = sc.textFile("C:/Users/ananya.shukla01/Desktop/FA3/FA3/Spark Dataset/Website traffic and user behavior.csv")

Web\_traffic=namedtuple('Web\_traffic',['Date','PageViews','UniqueVisitors','BounceRate','SessionDuration'])

header = RDD1.first()

df3 = RDD1.filter(lambda c: c!=header).map(lambda line :line.split(",")).map(lambda c : Web\_traffic (c[0],int(c[1]),int(c[2]),int(c[3]),int(c[4]))).toDF()

from pyspark.sql.functions import date\_format

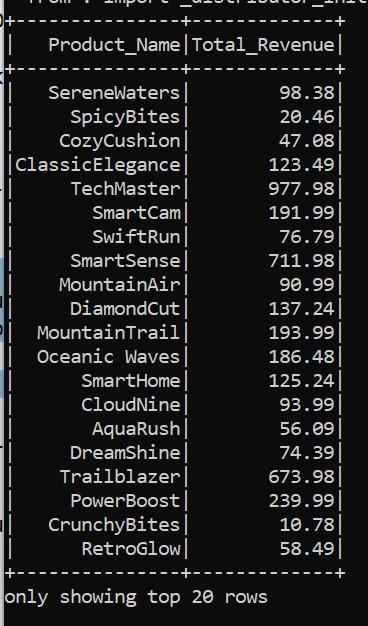
df = df3.withColumn('Date', date\_format(df3['Date'], 'M/dd/yyyy'))

df.createOrReplaceTempView("Web\_Traffic")

rdd3 = spark.sql("select p.Product\_Name, sum(round((p.Price-(p.Price\*c.Discount\_Applied)/100),2)) as Total\_Revenue from Customer\_Pur c join Product\_Catalog p on c.Product\_ID == p.Product\_ID where c.Purchase\_Date in (select Date from Web\_Traffic where UniqueVisitors >= 250) group by p.Product\_Name")

rdd3.show()

**Output:**

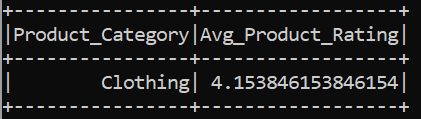


1. Find the average rating of products in the Clothing category using the product catalog dataset. (*2.5 marks*)

rdd4 = spark.sql("select Product\_Category, avg(Rating) as Avg\_Product\_Rating from Product\_Catalog where Product\_Category == 'Clothing' group by Product\_Category").show()

rdd4.show()

**Output:**



1. Identify the top 10 customers who spent the most money during the month of December 2021 using the customer purchase history dataset. (*2.5 marks*)

rdd5 = spark.sql("select c.Customer\_ID, c.Purchase\_Date, sum(round((p.Price-(p.Price\*c.Discount\_Applied)/100),2)) as Money\_Spent from Customer\_Pur c join Product\_Catalog p on c.Product\_ID == p.Product\_ID group by c.Purchase\_Date, c.Customer\_ID having c.Purchase\_Date like '12/%/2021' order by Money\_Spent desc limit 10")

rdd5.show()

**Output:**

