

Group A Lab 3D SparkSql

Participants:

1. Aadarsha chapagain
2. Jyoti shukla
3. Rishi Phaneendra Varma
4. Priti Bhale
5. Sreya Treesa Johny
6. Piyush Bhatia

Part1: choose any 5 Data Frame operation

1. Collect As List

`val collectDf = filteredDf.collectAsList`

The screenshot shows a Jupyter Notebook interface with a terminal window at the bottom. The terminal displays the execution of Spark SQL commands. The first command is `root@sandbox hdp:~#`. The second command is `scala> filteredDf.head(30)`, which returns a list of 30 rows of data. The data is displayed in a table format with columns: ProductId, PaymentType, CustomerName, City, Region, and Country. The third command is `scala> val collectDf = filteredDf.collectAsList`, which returns a list of 30 rows of data. The data is displayed in a table format with columns: ProductId, PaymentType, CustomerName, City, Region, and Country. The fourth command is `scala> collectDf`, which returns a list of 30 rows of data. The data is displayed in a table format with columns: ProductId, PaymentType, CustomerName, City, Region, and Country.

```
root@sandbox hdp:~#  
[ProductId, PaymentType, CustomerName, City, Region, Country]  
[Product1, 1200, Mastercard, carolina, Basildon, England, United Kingdom]  
[Product2, 3600, Visa, Sabine, London, England, United Kingdom]  
[Product1, 1200, Visa, Jeremy, Manchester, England, United Kingdom]  
[Product1, 1200, Visa, Chris, London, England, United Kingdom]  
[Product1, 1200, Mastercard, Tamar, Headley, England, United Kingdom]  
[Product1, 1200, Mastercard, Bernadett, Southampton, England, United Kingdom]  
[Product1, 1200, Mastercard, Anaisula, Helens Bay, Northern Ireland, United Kingdom]  
[Product1, 1200, Mastercard, Kevin, Cheltenham, England, United Kingdom]  
[Product1, 1200, Mastercard, Anne, Bournemouth, England, United Kingdom]  
[Product1, 1200, Visa, Grace, York, England, United Kingdom]  
[Product1, 1200, Visa, Cheryl, New Malden, England, United Kingdom]  
[Product1, 1200, Visa, Nicole, Basingstoke, England, United Kingdom]  
[Product2, 3600, Visa, Cheryl, New Malden, England, United Kingdom]  
[Product1, 1200, Visa, Doug, Bishop Auckland, England, United Kingdom]  
[Product1, 1200, Visa, Melissa, Aberdeen, Scotland, United Kingdom]  
[Product1, 1200, Visa, Randy, Wigan, England, United Kingdom]  
[Product1, 1200, Visa, Jose, Albury Town, England, United Kingdom]  
[Product1, 1200, Visa, Genevieve, Milton Keynes, England, United Kingdom]  
[Product1, 1200, Visa, Kevin, London, England, United Kingdom]  
only showing top 20 rows  
scala> filteredDf.head(30)  
res0: Array[(org.apache.spark.sql.Row) = Array([Product1,1200,Mastercard,carolina,Basildon,England,United Kingdom], [Product2,3600,Visa,Sabine,London,England,United Kingdom], [Product1,1200,Visa,Jeremy,Manchester,England,United Kingdom], [Product1,1200,Visa,Chris,London,England,United Kingdom], [Product1,1200,Mastercard,Tamar,Headley,England,United Kingdom], [Product1,1200,Mastercard,Bernadett,Southampton,England,United Kingdom], [Product1,1200,Mastercard,Anaisula,Helens Bay,Northern Ireland,United Kingdom], [Product1,1200,Mastercard,Kevin,Cheltenham,England,United Kingdom], [Product1,1200,Mastercard,Anne,Bournemouth,England,United Kingdom], [Product1,1200,Visa,Grace,York,England,United Kingdom], [Product1,1200,Visa,Cheryl,New Malden,England,United Kingdom], [Product1,1200,Visa,Nicole,Basingstoke,England,United Kingdom], [Product2,3600,Visa,Cheryl,New Malden,England,United Kingdom], [Product1,1200,Visa,Doug,Bishop Auckland,England,United Kingdom], [Product1,1200,Visa,Melissa,Aberdeen,Scotland,United Kingdom], [Product1,1200,Visa,Randy,Wigan,England,United Kingdom], [Product1,1200,Visa,Jose,Albury Town,England,United Kingdom], [Product1,1200,Visa,Genevieve,Milton Keynes,England,United Kingdom], [Product1,1200,Visa,Kevin,London,England,United Kingdom], [Product1,1200,Visa,Grace...]  
scala> val collectDf = filteredDf.collectAsList  
res1: java.util.List[(org.apache.spark.sql.Row) = [(Product1,1200,Mastercard,carolina,Basildon,England,United Kingdom), (Product2,3600,Visa,Sabine,London,England,United Kingdom), (Product1,1200,Visa,Jeremy,Manchester,England,United Kingdom), (Product1,1200,Visa,Chris,London,England,United Kingdom), (Product1,1200,Mastercard,Tamar,Headley,England,United Kingdom), (Product1,1200,Mastercard,Bernadett,Southampton,England,United Kingdom), (Product1,1200,Mastercard,Anaisula,Helens Bay,Northern Ireland,United Kingdom), (Product1,1200,Mastercard,Kevin,Cheltenham,England,United Kingdom), (Product1,1200,Mastercard,Anne,Bournemouth,England,United Kingdom), (Product1,1200,Visa,Grace,York,England,United Kingdom), (Product1,1200,Visa,Cheryl,New Malden,England,United Kingdom), (Product1,1200,Visa,Nicole,Basingstoke,England,United Kingdom), (Product2,3600,Visa,Cheryl,New Malden,England,United Kingdom), (Product1,1200,Visa,Doug,Bishop Auckland,England,United Kingdom), (Product1,1200,Visa,Melissa,Aberdeen,Scotland,United Kingdom), (Product1,1200,Visa,Randy,Wigan,England,United Kingdom), (Product1,1200,Visa,Jose,Albury Town,England,United Kingdom), (Product1,1200,Visa,Genevieve,Milton Keynes,England,United Kingdom), (Product1,1200,Visa,Kevin,London,England,United Kingdom), (Product1,1200,Visa,Grace...]  
scala> collectDf  
res2: List[(Product1,1200,Mastercard,carolina,Basildon,England,United Kingdom), (Product2,3600,Visa,Sabine,London,England,United Kingdom), (Product1,1200,Visa,Jeremy,Manchester,England,United Kingdom), (Product1,1200,Visa,Chris,London,England,United Kingdom), (Product1,1200,Mastercard,Tamar,Headley,England,United Kingdom), (Product1,1200,Mastercard,Bernadett,Southampton,England,United Kingdom), (Product1,1200,Mastercard,Anaisula,Helens Bay,Northern Ireland,United Kingdom), (Product1,1200,Mastercard,Kevin,Cheltenham,England,United Kingdom), (Product1,1200,Mastercard,Anne,Bournemouth,England,United Kingdom), (Product1,1200,Visa,Grace,York,England,United Kingdom), (Product1,1200,Visa,Cheryl,New Malden,England,United Kingdom), (Product1,1200,Visa,Nicole,Basingstoke,England,United Kingdom), (Product2,3600,Visa,Cheryl,New Malden,England,United Kingdom), (Product1,1200,Visa,Doug,Bishop Auckland,England,United Kingdom), (Product1,1200,Visa,Melissa,Aberdeen,Scotland,United Kingdom), (Product1,1200,Visa,Randy,Wigan,England,United Kingdom), (Product1,1200,Visa,Jose,Albury Town,England,United Kingdom), (Product1,1200,Visa,Genevieve,Milton Keynes,England,United Kingdom), (Product1,1200,Visa,Kevin,London,England,United Kingdom), (Product1,1200,Visa,Grace...]  
Responda: Sahas009  
0
```

2. Head

Head is used to show top specified elements

`filteredDf.head(30)`

The screenshot shows a Jupyter Notebook interface with a terminal window at the bottom. The terminal displays the execution of Spark SQL commands. The first command is `root@sandbox hdp:~#`. The second command is `scala> filteredDf.show`, which returns a list of 30 rows of data. The data is displayed in a table format with columns: ProductId, PaymentType, CustomerName, City, Region, and Country. The third command is `scala> filteredDf.head(30)`, which returns a list of 30 rows of data. The data is displayed in a table format with columns: ProductId, PaymentType, CustomerName, City, Region, and Country. The fourth command is `scala> filteredDf.head(30)`, which returns a list of 30 rows of data. The data is displayed in a table format with columns: ProductId, PaymentType, CustomerName, City, Region, and Country.

```
root@sandbox hdp:~#  
[ProductId, PaymentType, CustomerName, City, Region, Country]  
[Product1, 1200, Mastercard, carolina, Basildon, England, United Kingdom]  
[Product2, 3600, Visa, Sabine, London, England, United Kingdom]  
[Product1, 1200, Visa, Jeremy, Manchester, England, United Kingdom]  
[Product1, 1200, Visa, Chris, London, England, United Kingdom]  
[Product1, 1200, Mastercard, Tamar, Headley, England, United Kingdom]  
[Product1, 1200, Mastercard, Bernadett, Southampton, England, United Kingdom]  
[Product1, 1200, Mastercard, Anaisula, Helens Bay, Northern Ireland, United Kingdom]  
[Product1, 1200, Mastercard, Kevin, Cheltenham, England, United Kingdom]  
[Product1, 1200, Mastercard, Anne, Bournemouth, England, United Kingdom]  
[Product1, 1200, Visa, Grace, York, England, United Kingdom]  
[Product1, 1200, Visa, Cheryl, New Malden, England, United Kingdom]  
[Product1, 1200, Visa, Nicole, Basingstoke, England, United Kingdom]  
[Product2, 3600, Visa, Cheryl, New Malden, England, United Kingdom]  
[Product1, 1200, Visa, Doug, Bishop Auckland, England, United Kingdom]  
[Product1, 1200, Visa, Melissa, Aberdeen, Scotland, United Kingdom]  
[Product1, 1200, Visa, Randy, Wigan, England, United Kingdom]  
[Product1, 1200, Visa, Jose, Albury Town, England, United Kingdom]  
[Product1, 1200, Visa, Genevieve, Milton Keynes, England, United Kingdom]  
[Product1, 1200, Visa, Kevin, London, England, United Kingdom]  
only showing top 20 rows  
scala> filteredDf.show  
res0: Array[(org.apache.spark.sql.Row) = Array([Product1,1200,Mastercard,carolina,Basildon,England,United Kingdom], [Product2,3600,Visa,Sabine,London,England,United Kingdom], [Product1,1200,Visa,Jeremy,Manchester,England,United Kingdom], [Product1,1200,Visa,Chris,London,England,United Kingdom], [Product1,1200,Mastercard,Tamar,Headley,England,United Kingdom], [Product1,1200,Mastercard,Bernadett,Southampton,England,United Kingdom], [Product1,1200,Mastercard,Anaisula,Helens Bay,Northern Ireland,United Kingdom], [Product1,1200,Mastercard,Kevin,Cheltenham,England,United Kingdom], [Product1,1200,Mastercard,Anne,Bournemouth,England,United Kingdom], [Product1,1200,Visa,Grace,York,England,United Kingdom], [Product1,1200,Visa,Cheryl,New Malden,England,United Kingdom], [Product1,1200,Visa,Nicole,Basingstoke,England,United Kingdom], [Product2,3600,Visa,Cheryl,New Malden,England,United Kingdom], [Product1,1200,Visa,Doug,Bishop Auckland,England,United Kingdom], [Product1,1200,Visa,Melissa,Aberdeen,Scotland,United Kingdom], [Product1,1200,Visa,Randy,Wigan,England,United Kingdom], [Product1,1200,Visa,Jose,Albury Town,England,United Kingdom], [Product1,1200,Visa,Genevieve,Milton Keynes,England,United Kingdom], [Product1,1200,Visa,Kevin,London,England,United Kingdom], [Product1,1200,Visa,Grace...]  
scala> filteredDf.head(30)  
res1: java.util.List[(org.apache.spark.sql.Row) = [(Product1,1200,Mastercard,carolina,Basildon,England,United Kingdom), (Product2,3600,Visa,Sabine,London,England,United Kingdom), (Product1,1200,Visa,Jeremy,Manchester,England,United Kingdom), (Product1,1200,Visa,Chris,London,England,United Kingdom), (Product1,1200,Mastercard,Tamar,Headley,England,United Kingdom), (Product1,1200,Mastercard,Bernadett,Southampton,England,United Kingdom), (Product1,1200,Mastercard,Anaisula,Helens Bay,Northern Ireland,United Kingdom), (Product1,1200,Mastercard,Kevin,Cheltenham,England,United Kingdom), (Product1,1200,Mastercard,Anne,Bournemouth,England,United Kingdom), (Product1,1200,Visa,Grace,York,England,United Kingdom), (Product1,1200,Visa,Cheryl,New Malden,England,United Kingdom), (Product1,1200,Visa,Nicole,Basingstoke,England,United Kingdom), (Product2,3600,Visa,Cheryl,New Malden,England,United Kingdom), (Product1,1200,Visa,Doug,Bishop Auckland,England,United Kingdom), (Product1,1200,Visa,Melissa,Aberdeen,Scotland,United Kingdom), (Product1,1200,Visa,Randy,Wigan,England,United Kingdom), (Product1,1200,Visa,Jose,Albury Town,England,United Kingdom), (Product1,1200,Visa,Genevieve,Milton Keynes,England,United Kingdom), (Product1,1200,Visa,Kevin,London,England,United Kingdom), (Product1,1200,Visa,Grace...]  
scala> filteredDf.head(30)  
res2: List[(Product1,1200,Mastercard,carolina,Basildon,England,United Kingdom), (Product2,3600,Visa,Sabine,London,England,United Kingdom), (Product1,1200,Visa,Jeremy,Manchester,England,United Kingdom), (Product1,1200,Visa,Chris,London,England,United Kingdom), (Product1,1200,Mastercard,Tamar,Headley,England,United Kingdom), (Product1,1200,Mastercard,Bernadett,Southampton,England,United Kingdom), (Product1,1200,Mastercard,Anaisula,Helens Bay,Northern Ireland,United Kingdom), (Product1,1200,Mastercard,Kevin,Cheltenham,England,United Kingdom), (Product1,1200,Mastercard,Anne,Bournemouth,England,United Kingdom), (Product1,1200,Visa,Grace,York,England,United Kingdom), (Product1,1200,Visa,Cheryl,New Malden,England,United Kingdom), (Product1,1200,Visa,Nicole,Basingstoke,England,United Kingdom), (Product2,3600,Visa,Cheryl,New Malden,England,United Kingdom), (Product1,1200,Visa,Doug,Bishop Auckland,England,United Kingdom), (Product1,1200,Visa,Melissa,Aberdeen,Scotland,United Kingdom), (Product1,1200,Visa,Randy,Wigan,England,United Kingdom), (Product1,1200,Visa,Jose,Albury Town,England,United Kingdom), (Product1,1200,Visa,Genevieve,Milton Keynes,England,United Kingdom), (Product1,1200,Visa,Kevin,London,England,United Kingdom), (Product1,1200,Visa,Grace...]  
Responda: Sahas009  
0
```

Filter is used to filter according to given criteria it is similar to where clause

```
Val filteresDf = salesRecords.filter("Country = "united kingdom")
```

Filter is used to filter according to given criteria it is similar to where clause

```
Val filteresDf = salesRecords.filter("Country = "united kingdom")
```

The screenshot shows a Jupyter Notebook window titled "root@sandbox-hdp-". The notebook contains several code cells:

- A Scala code cell defining a filtered DataFrame:


```
val filteredDf = salesRecords.filter("Country"='United Kingdom')
```
- An error message from the console:


```
>console>!! error: unclosed character literal
val filteredDf = salesRecords.filter("Country"='United Kingdom')
```
- A Scala code cell showing the result of filtering:


```
scala> val filteredDf = salesRecords.filter("Country"='United Kingdom')
filteredDf: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Id: string, SalesAmt: int ... 5 more fields]
```
- A Scala code cell displaying the first 20 rows of the filtered DataFrame:


```
scala> filteredDf.show
```

The output of the last code cell is a table with 7 columns: Id, SalesAmt, PaymentType, CustomerName, City, Region, and Country. It lists 20 transactions where the country is 'United Kingdom'.

Id	SalesAmt	PaymentType	CustomerName	City	Region	Country
Product1	1200	Mastercard	carolina	Basilton	England/United Kingdom	
Product1	3600	Visa	Sabine	London	England/United Kingdom	
Product1	1200	Visa	Jeremy	Manchester	England/United Kingdom	
Product1	1200	Visa	chris	London	England/United Kingdom	
Product1	1200	Mastercard	Sari	Newbury	England/United Kingdom	
Product1	1200	Visa	Gindy	Rambley	England/United Kingdom	
Product1	1200	Mastercard	Tamar	Headley	England/United Kingdom	
Product1	1200	Mastercard	Bernadett	Southampton	England/United Kingdom	
Product1	1200	Mastercard	Anastasia	Helena Bay	Northern Ireland/United Kingdom	
Product1	1200	Mastercard	Kevin	Cheltenham	England/United Kingdom	
Product1	1200	Mastercard	anne	Bournemouth	England/United Kingdom	
Product1	1200	Visa	Grace	York	England/United Kingdom	
Product1	1200	Diners	Nicole	Basingstoke	England/United Kingdom	
Product2	3600	Visa	Cheryl	New Malden	England/United Kingdom	
Product1	1200	Visa	Doug/Bishop	Auckland	England/United Kingdom	
Product1	1200	Visa	Melissa	Aberdeen	Scotland/United Kingdom	
Product1	1200	Visa	Randy	Wigan	England/United Kingdom	
Product1	1200	Visa	Jessica	Abbey Town	England/United Kingdom	
Product1	1200	Visa	Genevieve	Milton Keynes	England/United Kingdom	
Product1	1200	Visa	Kevin	London	England/United Kingdom	

Below the table, it says "only showing top 20 rows".

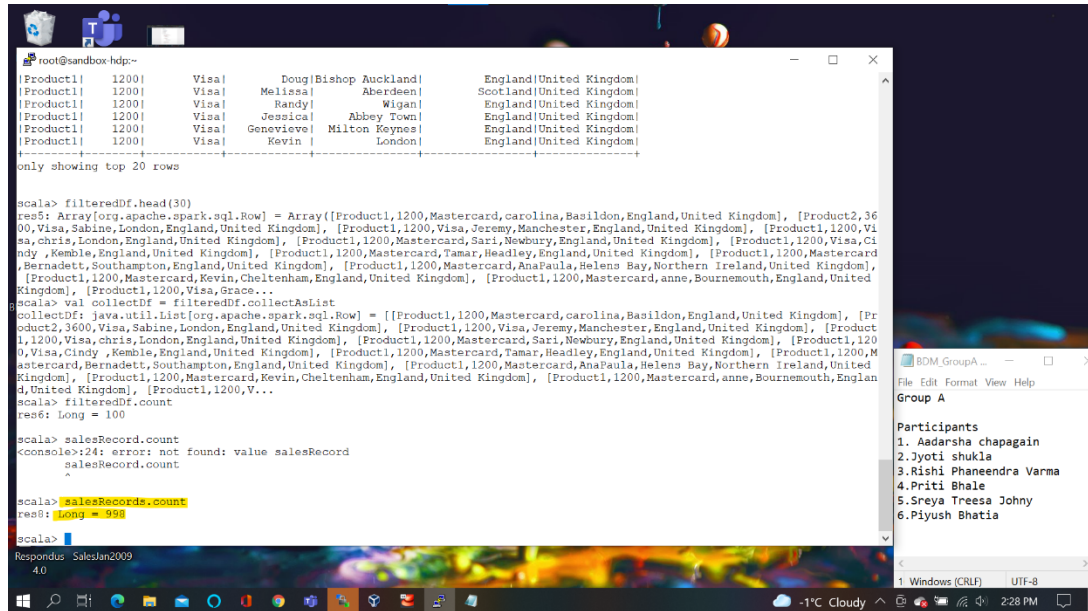
In the bottom right corner, there is a sidebar for a file explorer named "BDM_GroupA_...". It shows a tree view with folders like "File Edit Format View Help" and "Group A", and a list of participants including Adarscha chapagain, Jyoti shukla, Rishi Phaneendra Varma, Priti Bhale, Sreyas Treasa Johnny, and Piyush Bhatia.

4. Select

Select is used for projection

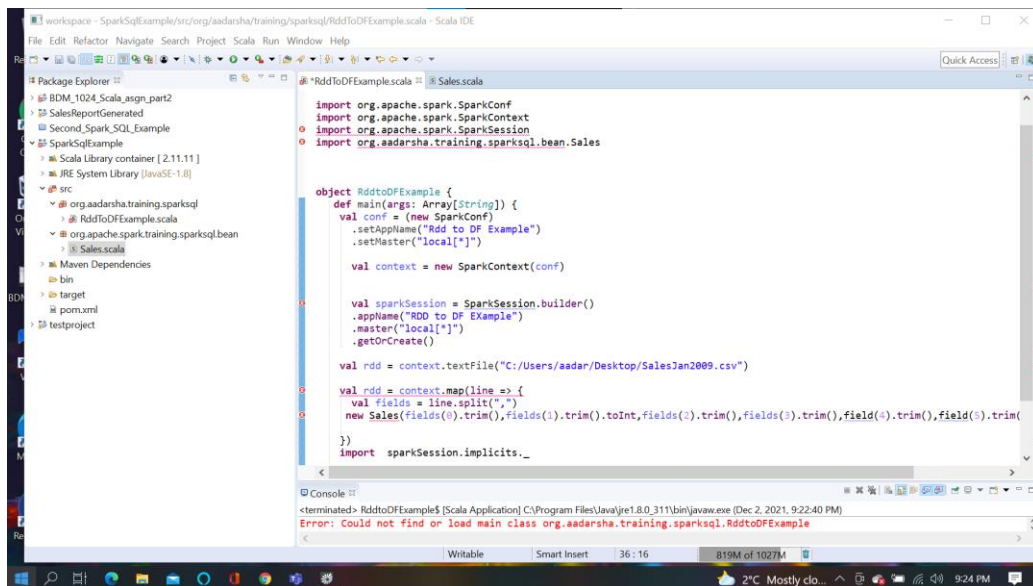
[illegible]

5. Count



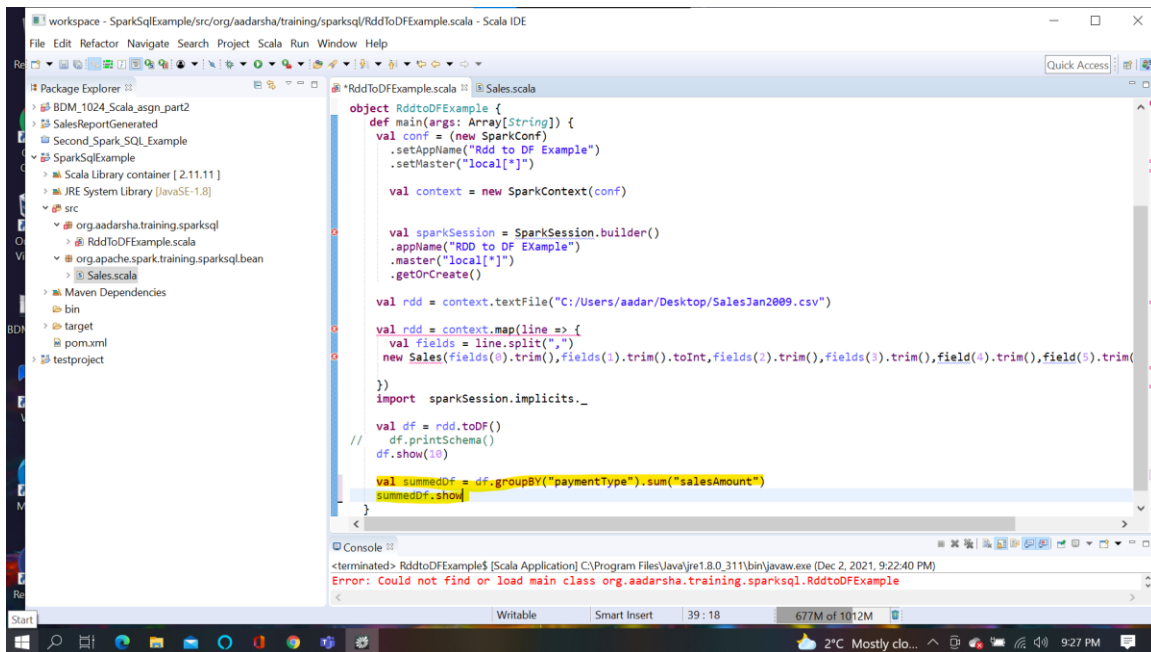
```
root@sandbox-hdp:~#  
+-----+-----+-----+-----+-----+-----+  
|ProductID|ProductID|ProductID|ProductID|ProductID|ProductID|  
+-----+-----+-----+-----+-----+-----+  
|1200|1200|1200|1200|1200|1200|  
+-----+-----+-----+-----+-----+-----+  
|Visa|Visa|Visa|Visa|Visa|Visa|  
+-----+-----+-----+-----+-----+-----+  
|Doug|Doug|Doug|Doug|Doug|Doug|  
+-----+-----+-----+-----+-----+-----+  
|Bishop|Bishop|Bishop|Bishop|Bishop|Bishop|  
+-----+-----+-----+-----+-----+-----+  
|Auckland|Auckland|Auckland|Auckland|Auckland|Auckland|  
+-----+-----+-----+-----+-----+-----+  
|England|England|England|England|England|England|  
+-----+-----+-----+-----+-----+-----+  
|United|United|United|United|United|United|  
+-----+-----+-----+-----+-----+-----+  
|Kingdom|Kingdom|Kingdom|Kingdom|Kingdom|Kingdom|  
+-----+-----+-----+-----+-----+-----+  
only showing top 20 rows  
  
scala> filteredDf.head(30)  
res5: Array[org.apache.spark.sql.Row] = Array([Product1,1200,Mastercard,carolina,Basildon,England,United Kingdom], [Product2,3600,Visa,Sabine,London,England,United Kingdom], [Product1,1200,Visa,Jeremy,Manchester,England,United Kingdom], [Product1,1200,Visa,Chris,London,England,United Kingdom], [Product1,1200,Mastercard,Sari,Newbury,England,United Kingdom], [Product1,1200,Visa,Cindy,Kemble,England,United Kingdom], [Product1,1200,Mastercard,Tamar,Headley,England,United Kingdom], [Product1,1200,Mastercard,Bernadett,Southampton,England,United Kingdom], [Product1,1200,Mastercard,AnaPaula,Helens Bay,Northern Ireland,United Kingdom], [Product1,1200,Mastercard,Kevin,Cheltenham,England,United Kingdom], [Product1,1200,Mastercard,anne,Bournemouth,England,United Kingdom], [Product1,1200,Visa,Grace...]  
scala> val collectDF = filteredDf.collectAsList  
collectDF: java.util.List[org.apache.spark.sql.Row] = [[Product1,1200,Mastercard,carolina,Basildon,England,United Kingdom], [Product2,3600,Visa,Sabine,London,England,United Kingdom], [Product1,1200,Visa,Jeremy,Manchester,England,United Kingdom], [Product1,1200,Visa,Chris,London,England,United Kingdom], [Product1,1200,Mastercard,Sari,Newbury,England,United Kingdom], [Product1,1200,Visa,Cindy,Kemble,England,United Kingdom], [Product1,1200,Mastercard,Tamar,Headley,England,United Kingdom], [Product1,1200,Mastercard,Bernadett,Southampton,England,United Kingdom], [Product1,1200,Mastercard,AnaPaula,Helens Bay,Northern Ireland,United Kingdom], [Product1,1200,Mastercard,Kevin,Cheltenham,England,United Kingdom], [Product1,1200,Mastercard,anne,Bournemouth,England,United Kingdom], [Product1,1200,Visa,Grace...]  
scala> filteredDf.count  
res6: Long = 100  
  
scala> salesRecord.count  
<console>:24: error: not found: value salesRecord  
salesRecord.count  
      ^  
scala> salesRecords.count  
res8: Long = 998  
scala> |  
Respondus SalesJan2009  
4.0
```

Slide 83:



```
workspace - SparkSqlExample/src/org/aadarsha/training/sparksql/RddToDFExample.scala - Scala IDE  
File Edit Refactor Navigate Search Project Scala Run Window Help  
Package Explorer  
BDM_1024_Scala.src.part2  
SalesReportGenerated  
Second_Spark_SQL_Example  
SparkSqlExample  
Scala Library container [2.11.11]  
JRE System Library [JavaSE-1.8]  
src  
org.aadarsha.training.sparksql  
RddToDFExample.scala  
org.apache.spark.training.sparksql.been  
Sales.scala  
Maven Dependencies  
bin  
target  
pom.xml  
testproject  
RddToDFExample.scala  
import org.apache.spark.SparkConf  
import org.apache.spark.SparkContext  
import org.apache.spark.SparkSession  
import org.aadarsha.training.sparksql.been.Sales  
  
object RddToDFExample {  
  def main(args: Array[String]) {  
    val conf = (new SparkConf)  
      .setAppName("Rdd to DF Example")  
      .setMaster("local[*"])  
  
    val context = new SparkContext(conf)  
  
    val sparkSession = SparkSession.builder()  
      .appName("RDD to DF Example")  
      .master("local[*"])  
      .getOrCreate()  
  
    val rdd = context.textFile("C:/Users/aadar/Desktop/SalesJan2009.csv")  
  
    val rdd = context.map(line => {  
      val fields = line.split(",")  
      new Sales(fields(0).trim(), fields(1).trim().toInt, fields(2).trim(), fields(3).trim(), fields(4).trim(), fields(5).trim()  
    })  
    import sparkSession.implicits._  
  }  
}
```

Slide 85:



The screenshot shows an IDE window titled "workspace - SparkSqlExample/src/org/aadarsha/training/sparksql/RddToDFExample.scala - Scala IDE". The left sidebar displays the Package Explorer with a project structure including "org.aadarsha.training.sparksql" and "Sales.scala". The main editor shows the following Scala code:

```
object RddToDFExample {  
  def main(args: Array[String]) {  
    val conf = (new SparkConf)  
      .setAppName("Rdd to DF Example")  
      .setMaster("local[*]")  
  
    val context = new SparkContext(conf)  
  
    val sparkSession = SparkSession.builder()  
      .appName("RDD to DF Example")  
      .master("local[*]")  
      .getOrCreate()  
  
    val rdd = context.textFile("C:/Users/aadar/Desktop/SalesJan2009.csv")  
  
    val rdd = context.map(line => {  
      val fields = line.split(",")  
      new Sales(fields(0).trim(), fields(1).trim().toInt, fields(2).trim(), fields(3).trim(), field(4).trim(), field(5).trim()  
    })  
    import sparkSession.implicits._  
  
    val df = rdd.toDF()  
    // df.printSchema()  
    df.show(10)  
  
    val summedDf = df.groupBy("paymentType").sum("salesAmount")  
    summedDf.show()  
  }  
}
```

The bottom console shows the following error message:

```
<terminated> RddtoDFExample$ [Scala Application] C:\Program Files\Java\jre1.8.0_311\bin\javaw.exe (Dec 2, 2021, 9:22:40 PM)  
Error: Could not find or load main class org.aadarsha.training.sparksql.RddtoDFExample
```