



```
scala> val df =
sqlContext.read.format("com.databricks.spark.csv").option("header",
"true").option("inferSchema", "true").load("train.csv")

scala> df.columns
Output: res0: Array[String] = Array(User_ID, Product_ID, Gender, Age, Occupation,
City_Category, Stay_In_Current_City_Years, Marital_Status, Product_Category_1,
Product_Category_2, Product_Category_3, Purchase)

scala> df.count()
Output: res1: Long = 550068

scala> df.printSchema()
Output: root
 |-- User_ID: integer (nullable = true)
 |-- Product_ID: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: string (nullable = true)
 |-- Occupation: integer (nullable = true)
 |-- City_Category: string (nullable = true)
 |-- Stay_In_Current_City_Years: string (nullable = true)
 |-- Marital_Status: integer (nullable = true)
 |-- Product_Category_1: integer (nullable = true)
 |-- Product_Category_2: integer (nullable = true)
 |-- Product_Category_3: integer (nullable = true)
 |-- Purchase: integer (nullable = true)

scala> df.show(2)
Output:
Output:
+-----+-----+-----+-----+-----+-----+-----+-----+
--+-----+-----+-----+-----+-----+-----+-----+-----+
--+
|User_ID|Product_ID|Gender|
Age|Occupation|City_Category|Stay_In_Current_City_Years|Marital_Status|Product_Ca
tegory_1|Product_Category_2|Product_Category_3|Purchase|
+-----+-----+-----+-----+-----+-----+-----+-----+
--+-----+-----+-----+-----+-----+-----+-----+-----+
--+
|1000001| P00069042| F|0-17| 10| A| 2| 0| 3| null| null| 8370|
|1000001| P00248942| F|0-17| 10| A| 2| 0| 1| 6| 14| 15200|
+-----+-----+-----+-----+-----+-----+-----+-----+
--+-----+-----+-----+-----+-----+-----+-----+-----+
--+
only showing top 2 rows

scala> df.select("Age").show(10)
Output:
+-----+
| Age|
+-----+
| 0-17|
| 0-17|
| 0-17|
| 0-17|
| 55+|
```

```
| 26-35 |
| 46-50 |
| 46-50 |
| 46-50 |
| 26-35 |
+-----+
```

only showing top 10 rows

```
scala> df.filter(df("Purchase") >= 10000).select("Purchase").show(10)
```

Output:

```
+-----+
| Purchase |
+-----+
| 15200 |
| 15227 |
| 19215 |
| 15854 |
| 15686 |
| 15665 |
| 13055 |
| 11788 |
| 19614 |
| 11927 |
+-----+
```

only showing top 10 rows

```
scala> val df1 = df.select("User_ID", "Occupation", "Marital_Status", "Purchase")
```

```
scala> import org.apache.spark.ml.feature.RFormula
```

```
scala> val formula = new RFormula().setFormula("Purchase ~
User_ID+Occupation+Marital_Status").setFeaturesCol("features").setLabelCol("label")
```

```
scala> val train = formula.fit(df1).transform(df1)
```

```
scala> import org.apache.spark.ml.regression.LinearRegression
```

```
scala> val lr = new
LinearRegression().setMaxIter(10).setRegParam(0.3).setElasticNetParam(0.8)
```

```
scala> val lrModel = lr.fit(train)
```

```
scala> println(s"Coefficients: ${lrModel.coefficients} Intercept:
${lrModel.intercept}")
```

Output: Coefficients: [0.015092115630330033,16.12117786898672,-10.520580986444338] Intercept: -5999.754797883323

```
scala> val trainingSummary = lrModel.summary
```

```
scala> trainingSummary.residuals.show(10)
```

Output:

```
+-----+
| residuals |
+-----+
| -883.5877032522076 |
| 5946.412296747792 |
| -7831.587703252208 |
```



```
| -8196.587703252208 |  
| -1381.3298625817588 |  
| 5892.776223171599 |  
| 10020.251134994305 |  
| 6659.251134994305 |  
| 6491.251134994305 |  
| -1533.3392694181512 |  
+-----+
```

only showing top 10 rows

```
scala> println(s"RMSE: ${trainingSummary.rootMeanSquaredError}")
```

Output: RMSE: 5021.899441991144