Retail Analysis

default

```
%md

## Working with HDFS & Yarn - Retail Data Analysis

#### Things to learn
* Reading from HDFS
* Reading from MySql ( from RDBMS using JDBC )
* Working with JSON Data Reading and Parsing
* Working with Spark SQLs
* Applying data transformaton using Spark SQL Statements
```

FINISHED

Working with HDFS & Yarn - Retail Data Analysis

Things to learn

- Reading from HDFS
- Reading from MySql (from RDBMS using JDBC)
- Working with JSON Data Reading and Parsing
- Working with Spark SQLs
- Applying data transformaton using Spark SQL Statements

Took 0 seconds

```
sc
```

FINISHED

```
res2: org.apache.spark.SparkContext = org.apache.spark.SparkContext@6ee55f6e
```

Took 2 seconds

```
sc.master
```

FINISHED

```
res4: String = yarn-client
```

Took 2 seconds

Zeppelin Notebook Interpreter

Search in your notebooks

READY Connected

Retail Analysis

default

READY

FINISHED

```
sqlContext
```

```
res6: org.apache.spark.sql.SQLContext = org.apache.spark.sql.hive.HiveContext@184b4499
```

Took 3 seconds

FINISHED

```
// Read the json file from hdfs
var txns = sqlContext.read.json("/sparklab/txnjsonsmall")
```

```
txns: org.apache.spark.sql.DataFrame = [CashOrCredit: string, creditCardNo: string, customerNo: string, lineItems: array<struct<amount:string,categor
y:string,product:string>>, merchantCity: string, state: string, tDate: string, txnNo: string]
```

Took 22 seconds

FINISHED

```
txns.show( 10 )
```

```
+------------+------------------+----------+--------------------+------------+----------+----------+--------+
|CashOrCredit|      creditCardNo|customerNo|           lineItems| merchantCity|     state|     tDate|   txnNo|
+------------+------------------+----------+--------------------+------------+----------+----------+--------+
|      credit|4971-xxxx-xxxx-5769|   4004819|[[015.82,Team Spo...|  Brownsville|     Texas|06-27-2011|00000000|
|      credit|3787-xxxx-xxxx-6017|   4003459|[[089.28,Water Sp...|      Houston|     Texas|02-07-2011|00000001|
|      credit|5951-xxxx-xxxx-4036|   4009112|[[067.51,Exercise...|       Eugene|    Oregon|03-02-2011|00000002|
|      credit|3793-xxxx-xxxx-3180|   4009376|[[043.38,Water Sp...|     Paterson|New Jersey|01-23-2011|00000003|
|      credit|3913-xxxx-xxxx-4556|   4006758|[[193.65,Outdoor ...|      Gresham|    Oregon|05-07-2011|00000004|
|      credit|4629-xxxx-xxxx-3692|   4000951|[[104.47,Exercise...|   Des Moines|      Iowa|12-07-2011|00000005|
|      credit|4032-xxxx-xxxx-1996|   4002494|[[093.97,Jumping,...|  St. Louis  |  Missouri|05-02-2011|00000006|
|      credit|3551-xxxx-xxxx-0696|   4000599|[[197.33,Exercise...|      Phoenix|   Arizona|06-02-2011|00000007|
|      credit|3282-xxxx-xxxx-5190|   4007057|[[128.98,Winter S...|Overland Park|    Kansas|03-06-2011|00000008|
|      credit|4621-xxxx-xxxx-9258|   4005366|[[074.57,Water Sp...|      Fremont|California|06-22-2011|00000009|
+------------+------------------+----------+--------------------+------------+----------+----------+--------+
only showing top 10 rows
```

Took 25 seconds

```
// This json has a nested structure.. the element lineitems itself has sub elements
txns.select( "lineItems" ).take( 5 )
```

FINISHED

```
res13: Array[org.apache.spark.sql.Row] = Array([WrappedArray([015.82,Team Sports,Cheerleading], [086.47,Water Sports,Whitewater Rafting], [063.08,Exercise & Fitness,Gym Mats], [068.80,Exercise & Fitness,Weightlifting Machine Accessories], [092.49,Team Sports,Lacrosse], [083.92,Outdoor Recreation,Lawn Games])], [WrappedArray([089.28,Water Sports,Water Tubing], [042.38,Water Sports,Surfing], [062.80,Team Sports,Cheerleading])], [WrappedArray([067.51,Exercise & Fitness,Exercise Bands], [154.57,Team Sports,Rugby], [100.18,Outdoor Recreation,Skateboarding], [190.52,Exercise & Fitness,Foam Rollers], [054.35,Water Sports,Kitesurfing])], [WrappedArray([043.38,Water Sports,Boating], [106.27,Team Sports,Rugby], [164.86,Combat Sports,Fencing], [164.94,Racquet Sports,Tennis], [007.36,Exercise & Fit...
```

Took 9 seconds

```
import org.apache.spark.sql.functions._
```

FINISHED

```
import org.apache.spark.sql.functions._
```

Took 5 seconds

```
// Explode and flatten the nested structure into a set of columns
var txns_exploded = txns.select( txns.col("txnNo"),
                                 txns.col("tDate"),
                                 txns.col("customerNo"),
                                 txns.col("merchantCity"),
                                 txns.col("state"),
                                 explode( txns.col("lineItems") ).as( "item" ) )
```

FINISHED

```
txns_exploded: org.apache.spark.sql.DataFrame = [txnNo: string, tDate: string, customerNo: string, merchantCity: string, state: string, item: struct<amount:string,category:string,product:string>]
```

Took 2 seconds

```
txns_exploded.show( 5 )
```

FINISHED

```
+--------+----------+----------+------------+-----+--------------------+
|   txnNo|     tDate|customerNo|merchantCity|state|                item|
+--------+----------+----------+------------+-----+--------------------+
|00000000|06-27-2011|   4004819| Brownsville|Texas|[015.82,Team Spor...|
|00000000|06-27-2011|   4004819| Brownsville|Texas|[086.47,Water Spo...|
|00000000|06-27-2011|   4004819| Brownsville|Texas|[063.08,Exercise ...|
|00000000|06-27-2011|   4004819| Brownsville|Texas|[068.80,Exercise ...|
|00000000|06-27-2011|   4004819| Brownsville|Texas|[092.49,Team Spor...|
+--------+----------+----------+------------+-----+--------------------+
```

```
only showing top 5 rows
```

Took 4 seconds

FINISHED

```
var txns_new = txns_exploded.select( txns_exploded.col("txnNo"),
                                     txns_exploded.col("tDate"),
                                     txns_exploded.col("customerNo"),
                                     txns_exploded.col("merchantCity"),
                                     txns_exploded.col("state"),
                                     txns_exploded.col("item").getField("category").as("category"),
                                     txns_exploded.col("item").getField("product").as("product"),
                                     txns_exploded.col("item").getField("amount").as("amount") )
```

```
txns_new: org.apache.spark.sql.DataFrame = [txnNo: string, tDate: string, customerNo: string, merchantCity: string, state: string, category: string,
 product: string, amount: string]
```

Took 3 seconds

FINISHED

```
txns_new.show( 10 )
```

```
+--------+----------+----------+------------+------+-----------------+--------------------+------+
|   txnNo|     tDate|customerNo|merchantCity| state|         category|             product|amount|
+--------+----------+----------+------------+------+-----------------+--------------------+------+
|00000000|06-27-2011|   4004819| Brownsville| Texas|      Team Sports|        Cheerleading|015.82|
|00000000|06-27-2011|   4004819| Brownsville| Texas|     Water Sports|  Whitewater Rafting|086.47|
|00000000|06-27-2011|   4004819| Brownsville| Texas|Exercise & Fitness|            Gym Mats|063.08|
|00000000|06-27-2011|   4004819| Brownsville| Texas|Exercise & Fitness|Weightlifting Mac...|068.80|
|00000000|06-27-2011|   4004819| Brownsville| Texas|      Team Sports|            Lacrosse|092.49|
|00000000|06-27-2011|   4004819| Brownsville| Texas|Outdoor Recreation|          Lawn Games|083.92|
|00000001|02-07-2011|   4003459|     Houston| Texas|     Water Sports|        Water Tubing|089.28|
|00000001|02-07-2011|   4003459|     Houston| Texas|     Water Sports|             Surfing|042.38|
|00000001|02-07-2011|   4003459|     Houston| Texas|      Team Sports|        Cheerleading|062.80|
|00000002|03-02-2011|   4009112|      Eugene|Oregon|Exercise & Fitness|      Exercise Bands|067.51|
+--------+----------+----------+------------+------+-----------------+--------------------+------+
only showing top 10 rows
```

Took 3 seconds

FINISHED

```
import sqlContext.implicits._

// cache the data frame before registering as table. This will prevent from parsting the record everytime the sql is run
```

```
txns_new.cache()

txns_new.registerTempTable( "txnrecords" )
```

```
import sqlContext.implicits._
res25: org.apache.spark.sql.DataFrame = [txnNo: string, tDate: string, customerNo: string, merchantCity: string, state: string, category: string, pro
duct: string, amount: string]
```

Took 5 seconds

```
var revenue_by_state = sqlContext.sql( "select state, product, sum( amount ) as total from txnrecords group by state, product" )
```

FINISHED

```
revenue_by_state: org.apache.spark.sql.DataFrame = [state: string, product: string, total: double]
```

Took 7 seconds

```
revenue_by_state.show( 10 )
```

FINISHED

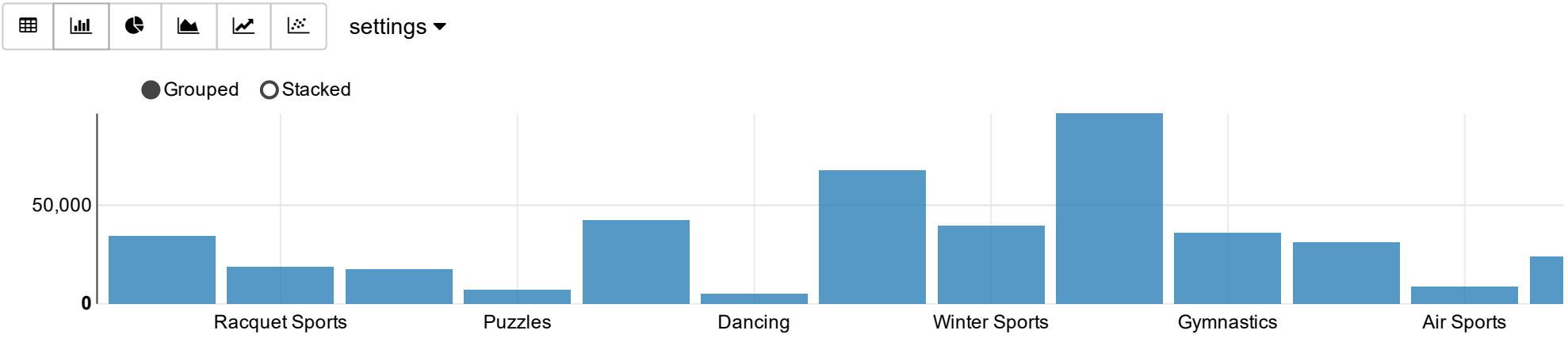
```
+-------------+--------------------+------------------+
|        state|             product|             total|
+-------------+--------------------+------------------+
|       Oregon|               Rugby|             261.4|
|        Texas|          Parachutes|            706.58|
|       Oregon|Scuba Diving & Sn...|            264.55|
|         Utah|           Wrestling|207.79000000000002|
|     Kentucky|         Bobsledding|232.84999999999997|
|      Florida|        Foam Rollers|            387.27|
|Massachusetts|          Air Hockey|            120.66|
|      Alabama|         Windsurfing|296.17999999999995|
|      Arizona|      Jumping Stilts| 96.00999999999999|
| Pennsylvania|           Disc Golf|             28.08|
+-------------+--------------------+------------------+
only showing top 10 rows
```

Took 16 seconds

```
%sql

select category, sum( amount ) as total from txnrecords group by category
```

FINISHED



Took 19 seconds

FINISHED

```
// Register the result sets as temporary tables
revenue_by_state.cache()
revenue_by_state.registerTempTable("state_revenue")
```

```
res33: org.apache.spark.sql.DataFrame = [state: string, product: string, total: double]
```

Took 10 seconds

FINISHED

```
//Write an UDF ( User defined function ) to extract week day name from the date field
import java.util.Calendar
import java.text.SimpleDateFormat

def getDayOfWeek(tDate: String ): Int = {

    val dateFormat = new SimpleDateFormat("MM-dd-yyyy")
    val c = Calendar.getInstance()
    c.setTime(dateFormat.parse( tDate ) )
    val dayOfWeek = c.get(Calendar.DAY_OF_WEEK)

    return dayOfWeek
}
```

```
import java.util.Calendar
import java.text.SimpleDateFormat
getDayOfWeek: (tDate: String)Int
```

Took 2 seconds

FINISHED

```
// Verify if the above function is working
getDayOfWeek( "01-01-2011")
```

```
res40: Int = 7
```

Took 3 seconds

FINISHED

```
sqlContext.udf.register("getDayOfWeek", (s: String) => getDayOfWeek(s) )
```

```
res42: org.apache.spark.sql.UserDefinedFunction = UserDefinedFunction(<function1>,IntegerType,List(StringType))
```

Took 5 seconds

FINISHED

```
// Write a query to invoke the user defined function. Calculate the total revenue by different weekdays.
var revenue_by_state = sqlContext.sql( """select weekday as weekday,
                                    round( sum( amount ), 2 ) as total
                                    from ( select getDayOfWeek( tDate ) as weekday,
                                    amount from txnrecords ) txns
                                    group by weekday order by total desc""" )
```

```
revenue_by_state: org.apache.spark.sql.DataFrame = [weekday: int, total: double]
```

Took 7 seconds

FINISHED

```
revenue_by_state.show()
```

```
+-------+--------+
|weekday|   total|
+-------+--------+
|      5| 94549.2|
|      4|85091.56|
|      2|81712.77|
|      1|79634.08|
|      3|79594.51|
|      7|78114.84|
```

```
|      6| 71809.1|
+-------+--------+
```

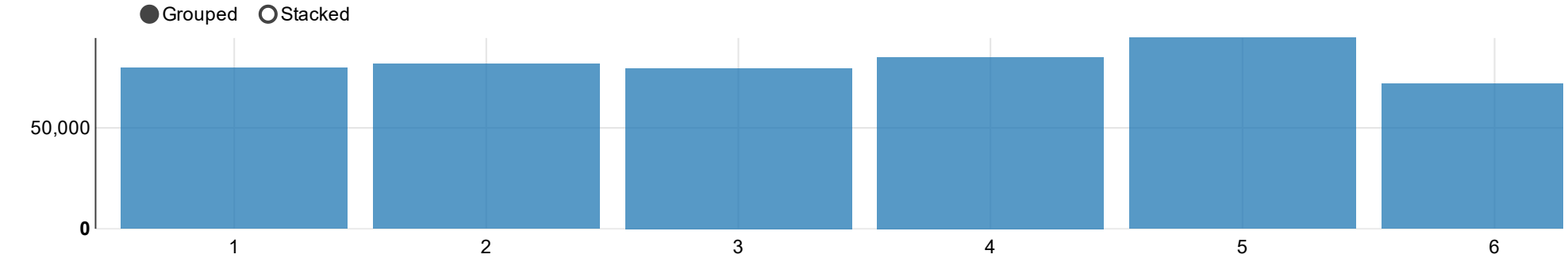
Took 12 seconds

FINISHED

```
%sql

select weekday as weekday,round( sum( amount ), 2 ) as total from ( select getDayOfWeek( tDate ) as weekday, amount from txnrecords ) txns group by
```

settings



Took 14 seconds

FINISHED

```
%md

## Reading data from MySql
#### Check MySql Table
* Go to linux prompt of your VM
* Enter "mysql -u root p"
* Enter password
* Select Database retail
    * use retail;
* Show tables
```

```
    * show tables;
    * describe customers;
* Load data into customers table
LOAD DATA LOCAL INFILE '/home/hadoop/lab/data/custs' INTO TABLE customers FIELDS
TERMINATED BY ',' LINES TERMINATED BY '\r\n';
* List some of the records
select * from customers limit 100;
```

Reading data from MySql

Check MySql Table

- Go to linux prompt of your VM
- Enter “mysql -u root p”
- Enter password
- Select Database retail
    - use retail;
- Show tables
    - show tables;
    - describe customers;
- Load data into customers table
  LOAD DATA LOCAL INFILE '/home/hadoop/lab/data/custs' INTO TABLE customers FIELDS
  TERMINATED BY ',' LINES TERMINATED BY '\r\n';
- List some of the records
  select * from customers limit 100;

Took 4 seconds

FINISHED

```
val driver = "com.mysql.jdbc.Driver"
val url = "jdbc:mysql://localhost/retail?user=root&password=hadoop123"

val jdbcDF = sqlContext.load("jdbc", Map(
  "url" -> url,
  "driver" -> driver,
  "dbtable" -> "customers"))

 jdbcDF.registerTempTable("customers")
```

```
driver: String = com.mysql.jdbc.Driver
url: String = jdbc:mysql://localhost/retail?user=root&password=hadoop123
```

```
warning: there were 1 deprecation warning(s); re-run with -deprecation for details
jdbcDF: org.apache.spark.sql.DataFrame = [CustID: string, FirstName: string, LastName: string, Age: int, Profession: string]
```

Took 9 seconds

FINISHED

```
%sql

select * from customers limit 10
```

CustID	FirstName	LastName	Age	Profession
4,000,002	Paige	Chen	74	Teacher
4,000,003	Sherri	Melton	34	Firefighter
4,000,004	Gretchen	Hill	66	Computer hardware engineer
4,000,005	Karen	Puckett	74	Lawyer
4,000,006	Patrick	Song	42	Veterinarian
4,000,007	Elsie	Hamilton	43	Pilot
4,000,008	Hazel	Bender	63	Carpenter
4,000,009	Malcolm	Wagner	39	Artist
4,000,010	Dolores	McLaughlin	60	Writer

Took 7 seconds

FINISHED

```
var top_10_custs = sqlContext.sql( """select customerNo, round( sum( amount ), 2 ) as total
                                      from txnrecords group by customerNo order by total desc limit 10""")

top_10_custs: org.apache.spark.sql.DataFrame = [customerNo: string, total: double]
```

Took 2 seconds

FINISHED

```
top_10_custs.registerTempTable( "top_10_custs" )
```

Took 2 seconds

```
var top_10_cust_names = sqlContext.sql( """select a.CustID, a.FirstName, a.LastName,
                                        a.Age, b.total from customers a join top_10_custs b
                                        on a.CustID = b.customerNo order by b.total desc""" )
```

FINISHED

```
top_10_cust_names: org.apache.spark.sql.DataFrame = [CustID: string, FirstName: string, LastName: string, Age: int, total: double]
```

Took 0 seconds

```
top_10_cust_names.show( 10 )
```

FINISHED

```
+-------+---------+---------+---+-------+
| CustID|FirstName| LastName|Age|  total|
+-------+---------+---------+---+-------+
|4007510|  Kristin|    Levin| 73|2204.79|
|4003293|   Martha|   Warner| 45|2024.67|
|4003971|   Donald|     Lamm| 34|1869.43|
|4004260| Courtney|    Rubin| 54|1869.12|
|4001058|   Gloria| Matthews| 53|1791.99|
|4008914| Samantha|Batchelor| 41|1652.06|
|4004491|     Rita|    Parks| 44|1649.74|
|4007168|  Carolyn|      Han| 52|1610.76|
|4001253|    Peter| McNamara| 74|1516.77|
|4009672|   Samuel|     Kidd| 61|1485.23|
+-------+---------+---------+---+-------+
```

Took 9 seconds

READY