Spark Developer Training - 3 Days

Manaranjan Pradhan
manaranjan@enablecloud.com

Lab: 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

In [1]:

```
sc
```

Out[1]:

```
<pyspark.context.SparkContext at 0x7f509021d1d0>
```

This Spark Application is running on YARN. So, open the YARN Resource manager UI and verify if the application is running

- To open resource manager web UI, enter http://hadooplab.bigdataleap.com:8088/ (http://hadooplab.bigdataleap.com:8088/)

Initialize SQLContext from SparkContext

In [2]:

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

Reading JSON file from HDFS

In [3]:

```
## Read the json file from HDFS
txns = sqlContext.read.json(
    "hdfs://hadooplab.bigdataleap.com/sparklab/txnjsonsmall")
```

Display the first 10 records

In [4]:

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

```
+------------+------------------+---------+--------------------+-----
--------+---------+---------+--------+
|CashOrCredit|      creditCardNo|customerNo|           lineItems| merc
hantCity|    state|    tDate|   txnNo|
+------------+------------------+---------+--------------------+-----
--------+---------+---------+--------+
|      credit|4971-xxxx-xxxx-5769|   4004819|[[015.82,Team Spo...|  Bro
wnsville|    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...|   De
s 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...|Overl
and 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
```

The line items are nested structure in each transaction. Display the lineitems of first 5 transactions

In [5]:

```
txns.select( "lineItems" ).take( 5 )
```

Out[5]:

```
[Row(lineItems=[Row(amount='015.82', category='Team Sports', product='C
heerleading'), Row(amount='086.47', category='Water Sports', product='W
hitewater Rafting'), Row(amount='063.08', category='Exercise & Fitnes
s', product='Gym Mats'), Row(amount='068.80', category='Exercise & Fitn
ess', product='Weightlifting Machine Accessories'), Row(amount='092.4
9', category='Team Sports', product='Lacrosse'), Row(amount='083.92', c
ategory='Outdoor Recreation', product='Lawn Games')]),
 Row(lineItems=[Row(amount='089.28', category='Water Sports', product
='Water Tubing'), Row(amount='042.38', category='Water Sports', product
='Surfing'), Row(amount='062.80', category='Team Sports', product='Chee
rleading')]),
 Row(lineItems=[Row(amount='067.51', category='Exercise & Fitness', pro
duct='Exercise Bands'), Row(amount='154.57', category='Team Sports', pr
oduct='Rugby'), Row(amount='100.18', category='Outdoor Recreation', pro
duct='Skateboarding'), Row(amount='190.52', category='Exercise & Fitnes
s', product='Foam Rollers'), Row(amount='054.35', category='Water Sport
s', product='Kitesurfing')]),
 Row(lineItems=[Row(amount='043.38', category='Water Sports', product
='Boating'), Row(amount='106.27', category='Team Sports', product='Rugb
y'), Row(amount='164.86', category='Combat Sports', product='Fencing'),
Row(amount='164.94', category='Racquet Sports', product='Tennis'), Row
(amount='007.36', category='Exercise & Fitness', product='Gym Mats'), R
ow(amount='110.56', category='Outdoor Recreation', product='Skateboardi
ng')]),
 Row(lineItems=[Row(amount='193.65', category='Outdoor Recreation', pro
duct='Deck Shuffleboard'), Row(amount='135.98', category='Winter Sport
s', product='Snowshoeing'), Row(amount='063.27', category='Racquet Spor
ts', product='Racquetball'), Row(amount='151.53', category='Dancing', p
roduct='Ballet Bars'), Row(amount='088.69', category='Gymnastics', prod
uct='Balance Beams'), Row(amount='070.75', category='Outdoor Play Equip
ment', product='Swing Sets')])]
```

Import the explode function to flatten the records

In [6]:

```
from pyspark.sql.functions import *
```

In [7]:

```
## Explode and flatten the nested structure into a set of columns
txns_new = txns.select( 'txnNo', 'tDate', 'customerNo', 'merchantCity',
                        'state', 'item.category',
                        'item.product', 'item.amount',
                        explode( txns.lineItems ).alias( 'item' ) ).drop( 'item')
```

In [8]:

```
# Show 10 records
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
```

Register the new table as temporary (in memory) table, so that we can run SQL Queries on it

In [9]:

```
# Register the dataframe as an temporary sql table into memory..
# so that we can go and run some sql query
txns_new.registerTempTable("txnrecords")
```

Find revenue generated by state and product

In [10]:

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

In [11]:

```
# show the first 10 records
revenue_by_state.show( 10 )
```

```
+---------+--------------------+------------------+
|    state|             product|             total|
+---------+--------------------+------------------+
|Minnesota|          Lawn Games|            245.45|
| Kentucky|          Bobsledding|232.84999999999997|
| Maryland|          Playhouses|            122.14|
|   Oregon|Camping & Backpac...|             88.38|
|  Florida|        Foam Rollers|            387.27|
|   Oregon|Scuba Diving & Sn...|            264.55|
| Nebraska|   Deck Shuffleboard|            317.43|
|   Oregon|               Rugby|             261.4|
|     Utah|           Wrestling|207.79000000000002|
|    Texas|          Parachutes|            706.58|
+---------+--------------------+------------------+
only showing top 10 rows
```

In [12]:

```
# We can also register the result sets as temporary tables
revenue_by_state.registerTempTable('state_revenue')
```

Write an UDF (User defined function) to extract week day name from the date field

In [13]:

```
# Define a user defined function to be invoked from sql query.
# For example, deriving weekday name from the date field
import datetime
def getDayOfWeek( date):
    return datetime.datetime.strptime(date, "%m-%d-%Y").strftime('%A')
```

Register the function to SQL Context as new UDF

In [14]:

```
# Register the function
from pyspark.sql.types import StringType
sqlContext.udf.register("getDayOfWeek", lambda date: getDayOfWeek( date ),
                        StringType() )
```

Invoke the UDF from the SQL

In [15]:

```
# Write a query to invoke the user defined function..
# Calculate the total revenue by different weekdays...
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''' )
```

In [16]:

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

```
+---------+--------+
|  weekday|   total|
+---------+--------+
| Thursday| 94549.2|
|Wednesday|85091.56|
|   Monday|81712.77|
|   Sunday|79634.08|
|  Tuesday|79594.51|
| Saturday|78114.84|
|   Friday| 71809.1|
+---------+--------+
```

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;

Using jdbc to read from mysql table

In [18]:

```
## Read customer information from mysql table....
cust_df = sqlContext.read.format('jdbc')                                          \
    .options(url='jdbc:mysql://localhost/retail?user=root&password=hadoop123',
    dbtable='customers').load()
```

In [19]:

```
cust_df.show( 10 )
```

```
+-------+---------+----------+---+--------------------+
| CustID|FirstName|  LastName|Age|          Profession|
+-------+---------+----------+---+--------------------+
|4000001| Kristina|     Chung| 55|               Pilot|
|4000002|    Paige|      Chen| 74|             Teacher|
|4000003|   Sherri|    Melton| 34|         Firefighter|
|4000004| Gretchen|      Hill| 66|Computer hardware...|
|4000005|    Karen|   Puckett| 74|              Lawyer|
|4000006|  Patrick|      Song| 42|        Veterinarian|
|4000007|    Elsie|  Hamilton| 43|               Pilot|
|4000008|    Hazel|    Bender| 63|           Carpenter|
|4000009|  Malcolm|    Wagner| 39|              Artist|
|4000010|  Dolores|McLaughlin| 60|              Writer|
+-------+---------+----------+---+--------------------+
only showing top 10 rows
```

Finding total money spent by each customers

In [20]:

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

In [21]:

```
top_10_custs.registerTempTable( "top_10_custs" )
cust_df.registerTempTable( "custs_rec" )
```

Joining with customer table to find top 10 customers based on total money spent

In [22]:

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

In [23]:

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

```
+-------+---------+---------+---+-------+
| 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|
+-------+---------+---------+---+-------+
```

Exercises

- Find out top 5 selling products in each category
- Find top 10 customers for every month

Make a note of things you learnt in the exercise.