# Creating a single historical dataset to be used for dashboarding in AWS QuickSight

## Creating the historical dataset

#### *Creating a dataframe for graph numbers to join with the historical dataset to create final set for the dashboard*

Importing and creating the SQL context

In [70]:

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

Importing the various functions for further usage.

In [71]:

```
from pyspark.sql.types import StructType, StringType
from pyspark import SparkConf, SparkContext
from pyspark import sql
from pyspark.sql.functions import lit
import numpy as np
import pandas as pd
```

Declaring a empty struct to later create an empty dataframe.

In [72]:

```
schema = StructType([])
```

Creating an empty dataframe to hold the graph numbers to include in the historical dataset to use in the dashboard.

In [73]:
```
empty = sqlContext.createDataFrame(sc.emptyRDD(), schema)
```

Declaring an array with the graph numbers

In [74]:
```
graph = np.array([1,2,3,4,5])
```

Creating the dataframe for the above created array

In [75]:
```
graphdf = pd.DataFrame(graph, columns = ['Graph'])
```

In [76]:
```
graphDF = sqlContext.createDataFrame(graphdf)
```

Registering the graph dataframe as a temp table to be used later for joining with historical datasets

In [77]:
```
graphDF.registerTempTable("grph_tbl")
```

In [78]:
```
graphDF.show()
```

```
+-----+
|Graph|
+-----+
|    1|
|    2|
|    3|
|    4|
|    5|
+-----+
```

***Reading and formatting the different historical datasets to be used in the dashboard***

Reading the historical top bottom 5 cities dataset from S3 bucket

```
In [79]: city = sqlContext.read.format('com.databricks.spark.csv') \
             .option("inferSchema",True).option("header",True).load('s3://bigdat
         aprjct/historical_data/top_bottom_5_city/top_bottom_5_city.csv')
```

Transforming the read city dataframe into RDD

```
In [80]: cityDataRDD = city.rdd
```

Caching the city RDD for faster processing and avoid reading from the bucket everytime

```
In [81]: cityDataRDD.cache()

Out[81]: MapPartitionsRDD[277] at javaToPython at NativeMethodAccessorImpl.java:
         0
```

Converting the city data RDD into a dataframe and providing the column names to the various columns a to the requirement for dashboarding

```
In [82]: cityDataDF = cityDataRDD.toDF(['Date_chr1','City1','CitySales1','TopBot
         tom1','Rank1'])
```

Adding column for Graph# according to the Graph that will be created using this portion of the data and joining with the temp graph table created above.

```
In [83]: cityDataDF = cityDataDF.withColumn('Graph',lit(1))
```

Registering the above created city dataframe as a temp table to join with the Graph table created earlier

```
In [84]: cityDataDF.registerTempTable("city_g1")
```

Viewing the schema to check if it looks correct

In [85]:
```
cityDataDF.printSchema()
```

```
root
 |-- Date_chr1: string (nullable = true)
 |-- City1: string (nullable = true)
 |-- CitySales1: double (nullable = true)
 |-- TopBottom1: string (nullable = true)
 |-- Rank1: long (nullable = true)
 |-- Graph: integer (nullable = false)
```

Checking the data in the city dataframe

In [17]:
```
cityDataDF.show()
```

```
+---------+-------------+----------+----------+-----+-----+
|Date_chr1|        City1|CitySales1|TopBottom1|Rank1|Graph|
+---------+-------------+----------+----------+-----+-----+
| 8/1/2015|       Playas|  7880.676|       bot|    1|    1|
| 8/1/2015|       Ibarra|  9941.613|       bot|    2|    1|
| 8/1/2015|     Riobamba|  10851.08|       bot|    3|    1|
| 8/1/2015|    El Carmen| 10930.531|       bot|    4|    1|
| 8/1/2015|      Salinas| 11353.155|       bot|    5|    1|
| 8/1/2015|      Machala| 36612.688|       top|    5|    1|
| 8/1/2015|       Cuenca| 39333.275|       top|    4|    1|
| 8/1/2015|Santo Domingo| 50046.034|       top|    3|    1|
| 8/1/2015|    Guayaquil|128751.848|       top|    2|    1|
| 8/1/2015|        Quito|521673.092|       top|    1|    1|
| 8/2/2015|     Riobamba|  7221.975|       bot|    1|    1|
| 8/2/2015|       Ibarra|   7279.21|       bot|    2|    1|
| 8/2/2015|      Salinas|  8084.883|       bot|    3|    1|
| 8/2/2015|       Playas|  8940.829|       bot|    4|    1|
| 8/2/2015|     Guaranda|  9285.041|       bot|    5|    1|
| 8/2/2015|       Ambato| 35782.769|       top|    5|    1|
| 8/2/2015|      Machala| 37985.512|       top|    4|    1|
| 8/2/2015|Santo Domingo| 59205.064|       top|    3|    1|
| 8/2/2015|   Guayaquil|135270.187|       top|    2|    1|
| 8/2/2015|       Quito|525404.305|       top|    1|    1|
+---------+------------+----------+----------+-----+-----+
only showing top 20 rows
```

Joining the above formed city table with graph table

In [86]:

```
g1_data = sqlContext.sql("""
    SELECT gr.*, c1.Date_chr1, c1.City1, c1.CitySales1, c1.TopBottom1,
 c1.Rank1
    from grph_tbl gr
    LEFT JOIN city_g1 c1
    ON gr.Graph = c1.Graph
""")

g1_data.printSchema()
```

```
root
 |-- Graph: long (nullable = true)
 |-- Date_chr1: string (nullable = true)
 |-- City1: string (nullable = true)
 |-- CitySales1: double (nullable = true)
 |-- TopBottom1: string (nullable = true)
 |-- Rank1: long (nullable = true)
```

Looking at the above created dataset

In [87]:

```
g1_data.show()
```

```
+-----+---------+------------+----------+----------+-----+
|Graph|Date_chr1|       City1|CitySales1|TopBottom1|Rank1|
+-----+---------+------------+----------+----------+-----+
|    5|     null|        null|      null|      null| null|
|    1| 8/1/2015|      Playas|  7880.676|       bot|    1|
|    1| 8/1/2015|      Ibarra|  9941.613|       bot|    2|
|    1| 8/1/2015|    Riobamba|  10851.08|       bot|    3|
|    1| 8/1/2015|    El Carmen| 10930.531|       bot|    4|
|    1| 8/1/2015|      Salinas| 11353.155|       bot|    5|
|    1| 8/1/2015|      Machala| 36612.688|       top|    5|
|    1| 8/1/2015|       Cuenca| 39333.275|       top|    4|
|    1| 8/1/2015|Santo Domingo| 50046.034|       top|    3|
|    1| 8/1/2015|    Guayaquil|128751.848|       top|    2|
|    1| 8/1/2015|        Quito|521673.092|       top|    1|
|    1| 8/2/2015|     Riobamba|  7221.975|       bot|    1|
|    1| 8/2/2015|       Ibarra|   7279.21|       bot|    2|
|    1| 8/2/2015|      Salinas|  8084.883|       bot|    3|
|    1| 8/2/2015|       Playas|  8940.829|       bot|    4|
|    1| 8/2/2015|     Guaranda|  9285.041|       bot|    5|
|    1| 8/2/2015|       Ambato| 35782.769|       top|    5|
|    1| 8/2/2015|      Machala| 37985.512|       top|    4|
|    1| 8/2/2015|Santo Domingo| 59205.064|       top|    3|
|    1| 8/2/2015|    Guayaquil|135270.187|       top|    2|
+-----+---------+-------------+----------+----------+-----+
only showing top 20 rows
```

Registering the above dataframe as a temp table to be joined later with other data parts.

In [88]:

```
g1_data.registerTempTable("g1_c")
```

## Reading the historical items data for Graph 2

***Reading the historical data for top and bottom items from S3 bucket for further processing***

In [89]:

```
item = sqlContext.read.format('com.databricks.spark.csv') \
    .option("inferSchema",True).option("header",True).load('s3://bigdataprjct/historical_data/top_bottom_5_item/top_bottom_5_item.csv')
```

Converting the iteam dataset read into RDD for further processsing

In [90]:

```
itemDataRDD = item.rdd
```

Caching the read item RDD for faster processing

In [91]:

```
itemDataRDD.cache()
```

Out[91]:

```
MapPartitionsRDD[306] at javaToPython at NativeMethodAccessorImpl.java:0
```

Converting the RDD to dataframe and renaming the column as per the requirement for dashboarding

In [93]:

```
itemDataDF = itemDataRDD.toDF(['Date_chr2','Family2','ItemSales2','TopBottom','Rank2'])
```

Adding the column Graph number and declaring all row values in the column to be 2 since item level graph in dashboard in number 2 and will further be joined with the above created final city dataset.

In [94]:

```
itemDataDF = itemDataDF.withColumn('Graph',lit(2))
```

Registering a temp table for items data to be further joined with above creaated final city dataset

In [95]:

```
itemDataDF.registerTempTable("item_g2")
```

Joining the above created item table with the final city datset created earlier which also has the graph number data

In [96]:

```
g2_data = sqlContext.sql("""
    SELECT g1.*, ig.Date_chr2, ig.Family2, ig.ItemSales2, ig.TopBottom, ig.Rank2
    from g1_c g1
    LEFT JOIN item_g2 ig
    ON g1.Graph = ig.Graph
""")

g2_data.printSchema()
```

```
root
 |-- Graph: long (nullable = true)
 |-- Date_chr1: string (nullable = true)
 |-- City1: string (nullable = true)
 |-- CitySales1: double (nullable = true)
 |-- TopBottom1: string (nullable = true)
 |-- Rank1: long (nullable = true)
 |-- Date_chr2: string (nullable = true)
 |-- Family2: string (nullable = true)
 |-- ItemSales2: double (nullable = true)
 |-- TopBottom: string (nullable = true)
 |-- Rank2: long (nullable = true)
```

Checking if the data was populated as we wanted into the above created dataset. Now there is data from Graph 1 and 2 in the dataframe while 3, 4 and 5 are still empty.

```
In [97]: g2_data.show()
```

```
+-----+---------+------------+----------+----------+-----+---------+--
-----+----------+---------+-----+
|Graph|Date_chr1|       City1|CitySales1|TopBottom1|Rank1|Date_chr2|Fa
mily2|ItemSales2|TopBottom|Rank2|
+-----+---------+------------+----------+----------+-----+---------+--
-----+----------+---------+-----+
|    5|     null|        null|      null|      null| null|     null|
null|      null|     null| null|
|    1| 8/1/2015|      Playas|  7880.676|       bot|    1|     null|
null|      null|     null| null|
|    1| 8/1/2015|      Ibarra|  9941.613|       bot|    2|     null|
null|      null|     null| null|
|    1| 8/1/2015|    Riobamba|  10851.08|       bot|    3|     null|
null|      null|     null| null|
|    1| 8/1/2015|   El Carmen| 10930.531|       bot|    4|     null|
null|      null|     null| null|
|    1| 8/1/2015|      Salinas| 11353.155|       bot|    5|     null|
null|      null|     null| null|
|    1| 8/1/2015|      Machala| 36612.688|       top|    5|     null|
null|      null|     null| null|
|    1| 8/1/2015|       Cuenca| 39333.275|       top|    4|     null|
null|      null|     null| null|
|    1| 8/1/2015|Santo Domingo| 50046.034|       top|    3|     null|
null|      null|     null| null|
|    1| 8/1/2015|    Guayaquil|128751.848|       top|    2|     null|
null|      null|     null| null|
|    1| 8/1/2015|        Quito|521673.092|       top|    1|     null|
null|      null|     null| null|
|    1| 8/2/2015|     Riobamba|  7221.975|       bot|    1|     null|
null|      null|     null| null|
|    1| 8/2/2015|       Ibarra|   7279.21|       bot|    2|     null|
null|      null|     null| null|
|    1| 8/2/2015|      Salinas|  8084.883|       bot|    3|     null|
null|      null|     null| null|
|    1| 8/2/2015|       Playas|  8940.829|       bot|    4|     null|
null|      null|     null| null|
|    1| 8/2/2015|     Guaranda|  9285.041|       bot|    5|     null|
null|      null|     null| null|
|    1| 8/2/2015|       Ambato| 35782.769|       top|    5|     null|
null|      null|     null| null|
|    1| 8/2/2015|      Machala| 37985.512|       top|    4|     null|
null|      null|     null| null|
|    1| 8/2/2015|Santo Domingo| 59205.064|       top|    3|     null|
null|      null|     null| null|
|    1| 8/2/2015|    Guayaquil|135270.187|       top|    2|     null|
null|      null|     null| null|
+-----+---------+-------------+----------+----------+-----+---------+--
-----+----------+---------+-----+
only showing top 20 rows
```

Registering a temp table for above created dataframe so that it can be further used for processing.

In [98]:

```
g2_data.registerTempTable("g2_ci")
```

## Reading the historical store level data

***Reading the historical store level data from AWS S3 bucket***

In [99]:

```
store = sqlContext.read.format('com.databricks.spark.csv') \
    .option("inferSchema",True).option("header",True).load('s3://bigdat
aprjct/historical_data/top_bottom_5_store/top_bottom_5_store.csv')
```

Transforming the read dataset into RDD

In [100]:

```
storeDataRDD = store.rdd
```

Caching the RDD to avoid re-read from source and for faster processing

In [101]:

```
storeDataRDD.cache()
```

Out[101]:

```
MapPartitionsRDD[342] at javaToPython at NativeMethodAccessorImpl.java:
0
```

Converting the RDD to dataframe and renaming the column so that it can be used in the dashboard with ease

In [102]:

```
storeDataDF =
storeDataRDD.toDF(['Date_chr3','StoreNbr3','ItemSales3','TopBottom3','R
ank3'])
```

Adding the Graph number to the store dataframe and populating the column with numeric 3 since the graph number for store level data is 3.

In [103]:

```
storeDataDF = storeDataDF.withColumn('Graph',lit(3))
```

Registering the store dataframe as a temp table to join it with the earlier created city and item level dataset and graph number

In [104]:

```
storeDataDF.registerTempTable("store_g3")
```

Joining the store data to the earlier created city and item dataset based on the graph number

In [105]:

```
g3_data = sqlContext.sql("""
    SELECT g1.*, sg.Date_chr3, sg.StoreNbr3, sg.ItemSales3, sg.TopBotto
m3, sg.Rank3
    from g2_ci g1
    LEFT JOIN store_g3 sg
    ON g1.Graph = sg.Graph
""")

g3_data.printSchema()
```

```
root
 |-- Graph: long (nullable = true)
 |-- Date_chr1: string (nullable = true)
 |-- City1: string (nullable = true)
 |-- CitySales1: double (nullable = true)
 |-- TopBottom1: string (nullable = true)
 |-- Rank1: long (nullable = true)
 |-- Date_chr2: string (nullable = true)
 |-- Family2: string (nullable = true)
 |-- ItemSales2: double (nullable = true)
 |-- TopBottom: string (nullable = true)
 |-- Rank2: long (nullable = true)
 |-- Date_chr3: string (nullable = true)
 |-- StoreNbr3: long (nullable = true)
 |-- ItemSales3: double (nullable = true)
 |-- TopBottom3: string (nullable = true)
 |-- Rank3: long (nullable = true)
```

Checking if the data is present in the format that we need for dashboarding. The graph number 1, 2 and 3 now have data corresponding to the city, item and store while 4 and 5 have blank rows

In [106]:
```
g3_data.show()
```

```
+-----+---------+-------------+----------+----------+-----+---------+--
-----+----------+---------+-----+---------+---------+----------+-------
---+-----+
|Graph|Date_chr1|        City1|CitySales1|TopBottom1|Rank1|Date_chr2|Fa
mily2|ItemSales2|TopBottom|Rank2|Date_chr3|StoreNbr3|ItemSales3|TopBott
om3|Rank3|
+-----+---------+-------------+----------+----------+-----+---------+--
-----+----------+---------+-----+---------+---------+----------+-------
---+-----+
|    5|     null|         null|      null|      null| null|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/1/2015|       Playas|  7880.676|       bot|    1|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/1/2015|       Ibarra|  9941.613|       bot|    2|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/1/2015|     Riobamba|  10851.08|       bot|    3|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/1/2015|    El Carmen| 10930.531|       bot|    4|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/1/2015|      Salinas| 11353.155|       bot|    5|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/1/2015|      Machala| 36612.688|       top|    5|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/1/2015|       Cuenca| 39333.275|       top|    4|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/1/2015|Santo Domingo| 50046.034|       top|    3|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/1/2015|    Guayaquil|128751.848|       top|    2|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/1/2015|        Quito|521673.092|       top|    1|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/2/2015|     Riobamba|  7221.975|       bot|    1|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/2/2015|       Ibarra|   7279.21|       bot|    2|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/2/2015|      Salinas|  8084.883|       bot|    3|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/2/2015|       Playas|  8940.829|       bot|    4|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/2/2015|     Guaranda|  9285.041|       bot|    5|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/2/2015|       Ambato| 35782.769|       top|    5|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/2/2015|      Machala| 37985.512|       top|    4|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/2/2015|Santo Domingo| 59205.064|       top|    3|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
|    1| 8/2/2015|    Guayaquil|135270.187|       top|    2|     null|
null|      null|     null| null|     null|     null|      null|      nu
ll| null|
+-----+---------+-------------+----------+----------+-----+---------+--
-----+----------+---------+-----+---------+---------+----------+-------
---+-----+
only showing top 20 rows
```

Registering the above created final datset as a temp table for further processing

In [107]:
```
g3_data.registerTempTable('g3_cis')
```

### Reading the historical day level transaction data for 4th dashboard

***Reading the historical day level transaction data from the S3 bucket to join with previously created dataset***

In [145]:
```
date = sqlContext.read.format('com.databricks.spark.csv') \
    .option("inferSchema",True).option("header",True).load('s3://bigdat
aprjct/historical_data/date_lvl/date_lvl.csv')
```

Transforming the read day level transaction data into RDD

In [109]:
```
dateDataRDD = date.rdd
```

Caching the data read above for faster processing and avoiding re-read

In [110]:
```
dateDataRDD.cache()
```

Out[110]:
```
MapPartitionsRDD[385] at javaToPython at NativeMethodAccessorImpl.java:
0
```

Converting the RDD into dataframe and providing it column names according to the dashboarding requirements.

In [144]:
```
dateDataDF =
dateDataRDD.toDF(['Date_chr4','StoreNbr4','Item4','Sales4','ItemCount4'
coil4','HolidayFlg4','TrnsCount4'])
```

Adding the Graph number column to the above dataframe and populating the rows with numeric value 4 since the day level transaction data will be used for creating the 4th graph in the dashbaord.

In [112]:

```
dateDataDF = dateDataDF.withColumn('Graph',lit(4))
```

Registering a temp table for above created dataframe to be used further

In [113]:

```
dateDataDF.registerTempTable("date_g4")
```

***Since the dashbaord 4th graph uses only last 14 days data we process the above formed dataframe to contain only last 14 days data***

Converting the date column in the above dataframe to teh date format and taking only the required sales column used for dashbaording from the data.

In [114]:

```
g4_data = sqlContext.sql("""
    SELECT sg.Graph,sg.Date_chr4, TO_DATE(CAST(UNIX_TIMESTAMP(sg.Date_c
hr4, 'MM/dd/yyyy') AS TIMESTAMP)) AS Date_dt4,sg.Sales4
    FROM date_g4 sg
""")

g4_data.printSchema()
```

```
root
 |-- Graph: integer (nullable = false)
 |-- Date_chr4: string (nullable = true)
 |-- Date_dt4: date (nullable = true)
 |-- Sales4: double (nullable = true)
```

Checking if teh got the data as we wanted

```
In [115]: g4_data.show()
```

```
+-----+---------+----------+-----------+
|Graph|Date_chr4|  Date_dt4|     Sales4|
+-----+---------+----------+-----------+
|    4| 8/1/2015|2015-08-01| 1044894.79|
|    4| 8/2/2015|2015-08-02|1043495.475|
|    4| 8/3/2015|2015-08-03| 811119.834|
|    4| 8/4/2015|2015-08-04| 726613.358|
|    4| 8/5/2015|2015-08-05| 724346.594|
|    4| 8/6/2015|2015-08-06| 583375.212|
|    4| 8/7/2015|2015-08-07| 663326.518|
|    4| 8/8/2015|2015-08-08| 784099.216|
|    4| 8/9/2015|2015-08-09| 679410.125|
|    4|8/10/2015|2015-08-10| 785477.934|
|    4|8/11/2015|2015-08-11| 668280.358|
|    4|8/12/2015|2015-08-12| 678064.732|
|    4|8/13/2015|2015-08-13| 557859.478|
|    4|8/14/2015|2015-08-14| 734484.205|
|    4|8/15/2015|2015-08-15| 936339.026|
|    4|8/16/2015|2015-08-16| 913846.625|
|    4|8/17/2015|2015-08-17| 694747.162|
|    4|8/18/2015|2015-08-18| 629475.643|
|    4|8/19/2015|2015-08-19|  653733.36|
|    4|8/20/2015|2015-08-20| 569443.704|
+-----+---------+----------+-----------+
only showing top 20 rows
```

Registering the above transformed and filtered dataset for further transformation that is filtering the last 14 days data.

```
In [116]: g4_data.registerTempTable("date_sg")
```

Filtering the above data to contain only the last 14 days data

In [117]:

```
final_dt_lvl = sqlContext.sql("""
     SELECT sg.Graph, sg.Date_chr4, sg.Sales4
    FROM date_sg sg
    WHERE sg.Date_dt4 >= (SELECT date_sub(MAX(Date_dt4),13) FROM date_s
g)
""")
```

Check if we got the data we needed

In [118]:

```
final_dt_lvl.show()
```

```
+-----+---------+-----------+
|Graph|Date_chr4|     Sales4|
+-----+---------+-----------+
|    4| 8/1/2017| 988527.763|
|    4| 8/2/2017| 964712.016|
|    4| 8/3/2017| 728068.485|
|    4| 8/4/2017| 827775.686|
|    4| 8/5/2017|  965693.65|
|    4| 8/6/2017|1049559.164|
|    4| 8/7/2017| 797464.964|
|    4| 8/8/2017| 717766.349|
|    4| 8/9/2017| 734139.674|
|    4|8/10/2017| 651386.912|
|    4|8/11/2017| 826373.722|
|    4|8/12/2017| 792630.535|
|    4|8/13/2017| 865639.677|
|    4|8/14/2017| 760922.406|
+-----+---------+-----------+
```

Registering the above created dataframe to be joined with the previously created dataset for Graphs 1, 2 and 3.

In [119]:

```
final_dt_lvl.registerTempTable("date_lvl_4")
```

Joining the data for Graph 4 with the previous data created for Graph 1, 2 and 3.

In [120]:

```
dtlvljoin = sqlContext.sql("""
        SELECT g1.*, dtl.Date_chr4, dtl.Sales4
    from g3_cis g1
    LEFT JOIN date_lvl_4 dtl
    ON g1.Graph = dtl.Graph
        """)

dtlvljoin.printSchema()
```

```
root
 |-- Graph: long (nullable = true)
 |-- Date_chr1: string (nullable = true)
 |-- City1: string (nullable = true)
 |-- CitySales1: double (nullable = true)
 |-- TopBottom1: string (nullable = true)
 |-- Rank1: long (nullable = true)
 |-- Date_chr2: string (nullable = true)
 |-- Family2: string (nullable = true)
 |-- ItemSales2: double (nullable = true)
 |-- TopBottom: string (nullable = true)
 |-- Rank2: long (nullable = true)
 |-- Date_chr3: string (nullable = true)
 |-- StoreNbr3: long (nullable = true)
 |-- ItemSales3: double (nullable = true)
 |-- TopBottom3: string (nullable = true)
 |-- Rank3: long (nullable = true)
 |-- Date_chr4: string (nullable = true)
 |-- Sales4: double (nullable = true)
```

Registering the data created till Graph 4 into a temp table for further processing

In [121]:

```
dtlvljoin.registerTempTable("g4F")
```

## Reading the data for graph 5 - linear regression showing relation

### between lag days transaction amounts

***Reading the liner regression graph data from S3 bucket.***

In [122]:

```
reg = sqlContext.read.format('com.databricks.spark.csv') \
    .option("inferSchema",True).option("header",True).load('s3://bigdat
aprjct/historical_data/linear_reg/Regression_Historical.csv')
```

Registering the above read data into a RDD

In [123]:

```
regRDD = reg.rdd
```

Caching the RDD for easier and faster processing further

In [124]:

```
regRDD.cache()
```

Out[124]:

```
MapPartitionsRDD[414] at javaToPython at NativeMethodAccessorImpl.java:
0
```

Transforming the above RDD into dataframe and renaming teh column as requirement for the Graph in the dashboard.

In [125]:

```
regDF = regRDD.toDF(['Date_chr5','Term5','Variable5','Value5'])
```

Adding the column for graph number to the above dataframe and populating the rows with value 5 since the graph for this on the dashboard is graph 5

In [126]:

```
regDF = regDF.withColumn('Graph',lit(5))
```

Registering the above dataframe as temp table for further processing

In [127]:

```
regDF.registerTempTable("reg_g5")
```

Joining the above created graph 5 data with the previously created data for graphs 1-4.

In [128]:
```
g5DF = sqlContext.sql("""
    SELECT g1.*, rg.Date_chr5, rg.Term5, rg.Variable5, rg.Value5
    from g4F g1
    LEFT JOIN reg_g5 rg
    ON g1.Graph = rg.Graph
""")

g5DF.printSchema()
```

```
root
 |-- Graph: long (nullable = true)
 |-- Date_chr1: string (nullable = true)
 |-- City1: string (nullable = true)
 |-- CitySales1: double (nullable = true)
 |-- TopBottom1: string (nullable = true)
 |-- Rank1: long (nullable = true)
 |-- Date_chr2: string (nullable = true)
 |-- Family2: string (nullable = true)
 |-- ItemSales2: double (nullable = true)
 |-- TopBottom: string (nullable = true)
 |-- Rank2: long (nullable = true)
 |-- Date_chr3: string (nullable = true)
 |-- StoreNbr3: long (nullable = true)
 |-- ItemSales3: double (nullable = true)
 |-- TopBottom3: string (nullable = true)
 |-- Rank3: long (nullable = true)
 |-- Date_chr4: string (nullable = true)
 |-- Sales4: double (nullable = true)
 |-- Date_chr5: string (nullable = true)
 |-- Term5: string (nullable = true)
 |-- Variable5: string (nullable = true)
 |-- Value5: double (nullable = true)
```

Registering the above created final dataset as temp table for further processing.

In [129]:

```
g5DF.registerTempTable("g5F")
```

Replacing and putting the columns in the place as required for the final dashbaording.

In [139]:

```
final5 = sqlContext.sql("""
        SELECT Graph, COALESCE(Date_chr1, Date_chr2, Date_chr3,Date_chr
4,Date_chr5) AS Date,
        COALESCE(Date_chr1, Date_chr2, Date_chr3,Date_chr4, Date_chr5)
 AS Date_chr,
        City1, CitySales1, TopBottom1, Rank1, Family2, ItemSales2, TopB
ottom, Rank2,
        StoreNbr3, ItemSales3, TopBottom3, Rank3,
        Sales4, Term5, Variable5, Value5
        FROM g5F
        """)
```

In [140]:

```
final5.printSchema()
```

```
root
 |-- Graph: long (nullable = true)
 |-- Date: string (nullable = true)
 |-- Date_chr: string (nullable = true)
 |-- City1: string (nullable = true)
 |-- CitySales1: double (nullable = true)
 |-- TopBottom1: string (nullable = true)
 |-- Rank1: long (nullable = true)
 |-- Family2: string (nullable = true)
 |-- ItemSales2: double (nullable = true)
 |-- TopBottom: string (nullable = true)
 |-- Rank2: long (nullable = true)
 |-- StoreNbr3: long (nullable = true)
 |-- ItemSales3: double (nullable = true)
 |-- TopBottom3: string (nullable = true)
 |-- Rank3: long (nullable = true)
 |-- Sales4: double (nullable = true)
 |-- Term5: string (nullable = true)
 |-- Variable5: string (nullable = true)
 |-- Value5: double (nullable = true)
```

Dropping the unnecessary columns from the above dataframe

In [141]:
```
from functools import reduce
from pyspark.sql import DataFrame

final_his = reduce(DataFrame.drop,
['Date_chr1','Date_chr2','Date_chr3', 'Date_chr4','Date_chr5'], final5)
```

## Clearing the space on S3 where we are going to write this historical dataset

In [142]:
```
import os

cmd="hdfs dfs -rm -r -skipTrash s3n://bigdataprjct/Miscellaneous/historical"

os.system(cmd)
```

Out[142]:
```
0
```

***Writing the final dataset to a folder on S3 bucket.***

In [143]:
```
final_his\
    .coalesce(1)\
    .write.format("com.databricks.spark.csv")\
    .option("header", "true")\
    .save("s3n://bigdataprjct/Miscellaneous/historical")
```

***Linear Reg***