Scenario

“80% of your sales come from 20% of your customers” - The Pareto Principle

Customer segmentation has been a marketing tactic in use for years to identify these “best customers” and once such technique is RFM analysis

RFM analysis is a simple statistical measure of the “value” of a customer. It is based on 3 scores from a series of customer transaction data:

Recency: how recently a customer has purchased
Frequency: how often they purchase
Monetary Value: how much the customer spends

A local marketing company has been collecting transaction data from a number of small businesses in an enterprise data warehouse. They would like to sell an RFM modeling service to their clients to supplement their data warehousing capabilities.
They would like to hire you to prototype this service.

For simplicity for this prototype, the results will be stored to file instead of an online store like the would typically be in production (HBase/Cassandra/Redis,etc…) We will cover this once we talk about landing data.

Assignment

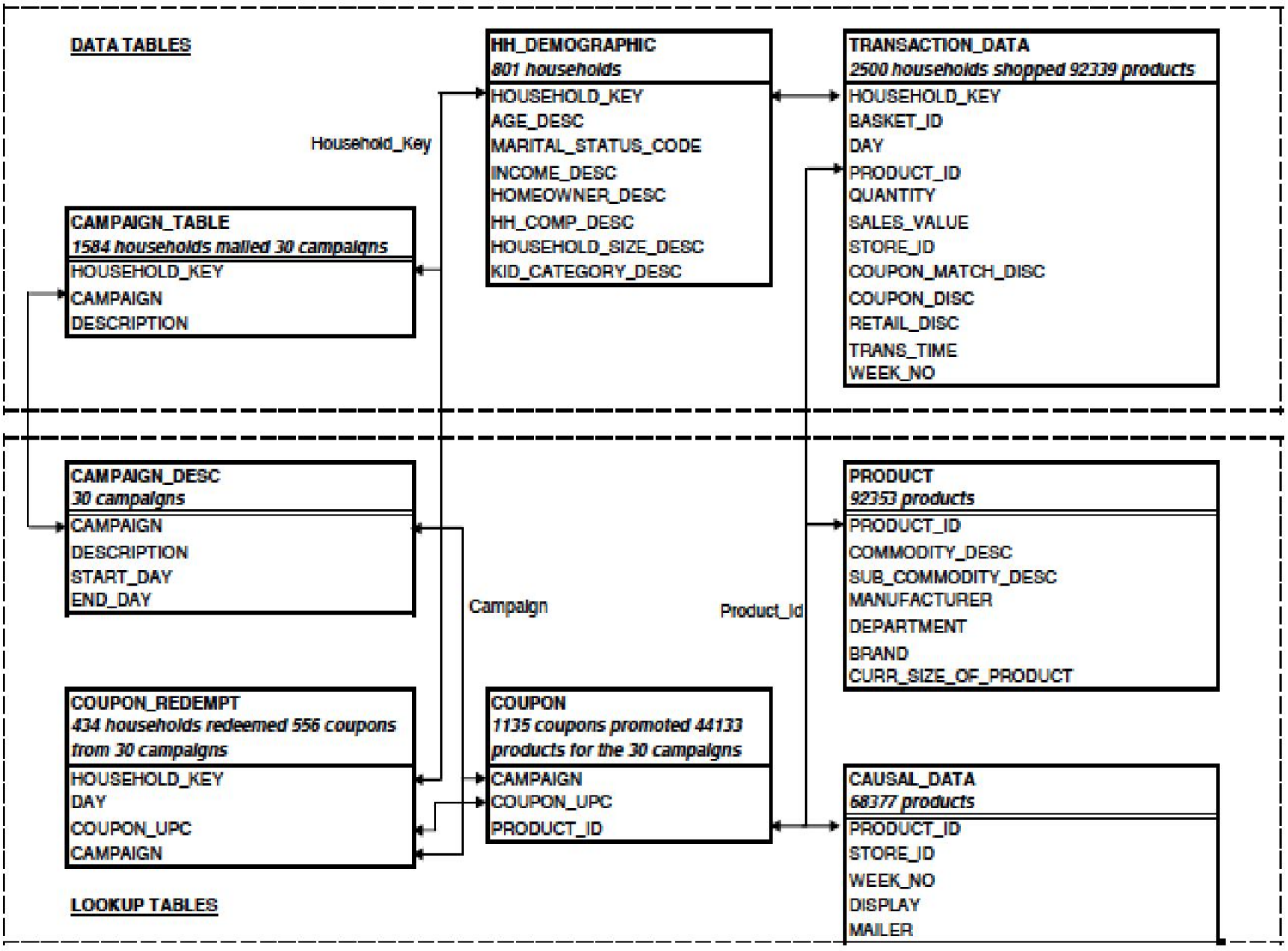
Using the provided demonstration transactional DB, extract the necessary data from the warehouse and calculate an RFM table in Spark for all households. Additionally the “pilot” customer would then like a report of only scores for homeowners (HOMEOWNER_DESC = “Homeowner”).

Input:

The data currently exists in a relational database. Connection details (Note that you should only be able to connect to the database from your UW provided VM i.e. not personal computer)

Username: bigdata
Password: 5Zgv\6;8rM
Host: bigdata220w18.database.windows.net
Port: 1443
Database: week3
JDBC URL: jdbc:sqlserver://bigdata220w18.database.windows.net:1433;database=week3

The ERM diagram is as follows:



There are 3 tables of interest:

hh_demographic
This table contains demographic information for a portion of households. Due to nature of the data, the demographic information is not available for all households

Variable	Description
HOUSEHOLD_KEY	Uniquely identifies each household
AGE_DESC	Estimated age range
MARITAL_STATUS_CODE	Marital Status (A - Married, B- Single, U - Unknown)
INCOME_DESC	Household income
HOMEOWNER_DESC	Homeowner, renter, etc.
HH_COMP_DESC	Household composition
HOUSEHOLD_SIZE_DESC	Size of household up to 5+
KID_CATEGORY_DESC	Number of children present up to 3+

transaction_data
This table contains all products purchased by households within this study. Each line found in this table is essentially the same line that would be found on a store receipt.

Variable	Description
HOUSEHOLD_KEY	Uniquely identifies each household
BASKET_ID	Uniquely identifies a purchase occasion
DAY	Day when transaction occurred
PRODUCT_ID	Uniquely identifies each product
QUANTITY	Number of the products purchased during the trip
SALES_VALUE	Amount of dollars retailer receives from sale
STORE_ID	Identifies unique stores
COUPON_MATCH_DISC	Discount applied due to retailer's match of manufacturer coupon
COUPON_DISC	Discount applied due to manufacturer coupon
RETAIL_DISC	Discount applied due to retailer's loyalty card program
TRANS_TIME	Time of day when the transaction occurred
WEEK_NO	Week of the transaction. Ranges 1 - 102

product
This table contains information on each product sold such as type of product, national or private label and a brand identifier.

Variable	Description
PRODUCT_ID	Number that uniquely identifies each product
DEPARTMENT	Groups similar products together
COMMODITY_DESC	Groups similar products together at a lower level
SUB_COMMODITY_DESC	Groups similar products together at the lowest level
MANUFACTURER	Code that links products with same manufacturer together
BRAND	Indicates Private or National label brand
CURR_SIZE_OF_PRODUCT	Indicates package size (not available for all products)

Use Sqoop to de-normalize these 3 tables joining on household_key and product_id and stage in HDFS as a single data source in Parquet format. See this weeks slides for info on sqoop and sqoop documentation for details on the arguments used at http://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html#_literal_sqoop_import_literal

You will need to download the Azure SQL Server driver from:
https://bigdata220w18.blob.core.windows.net/blobs/sqljdbc42.jar

The driver class name is `com.microsoft.sqlserver.jdbc.SQLServerDriver`

Copy that JAR file (cp command) to the `/usr/hdp/current/sqoop-client/lib` directory in your HDP sandbox. You will use sqoop from inside the sandbox to insert the data into Parquet format into HDFS.

You will need to add the "driver" flag to Sqoop and add the driver name in order to connect to the Azure SQL Server database like discussed in class:

```
sqoop import --driver com.microsoft.sqlserver.jdbc.SQLServerDriver
…..
```

The RFM metrics are defined as follows:

Recency: The most recent "day" from TRANSACTION DATA (can disregard weeks and time)
Frequency: The number of unique "basket_id" from TRANSACTION_DATA
Monetary Value: The sum of all "sales_value" from TRANSACTION_DATA

RFM is also the order of importance, so the results should be sorted by recency, then frequency, then monetary value

Output:

There should be two output files generated. A CSV file containing RFM calculations for each household, and a CSV file containing only homeowners.

Each output file record should have the following format:
Household key, Recency, Frequency, Monetary

Submission Guidelines:

The sqoop command to pull over the data, and the Spark code (Scala, Python or Java) to generate the requested output to fulfill the assignment.

A notebook with all of this is preferred (Zeppelin or Jupyter) but individual files are acceptable.

Example

Given the following (truncated) data:

```
household_key |  basket_id  | day | product_id | quantity | sales_value | store_id |
blanks
--------------+-------------+-----+------------+----------+-------------+----------+----------
--
         2375 | 26984851472 | 711 |    1033142 |        1 |           0 |      364 |
0
         2375 | 26984851472 | 711 |    1082185 |        1 |           1 |      364 |
0
         2375 | 26984851516 | 555 |     826249 |        2 |           1 |      364 |
0
         2375 | 26984851516 | 555 |    1085983 |        1 |           2 |      364 |
0
         2375 | 26984851516 | 555 |    6423775 |        1 |           2 |      364 |
0
         1364 | 26984896261 | 686 |     842930 |        1 |           2 |    31742 |
0
         1364 | 26984896261 | 686 |     920955 |        1 |           3 |    31742 |
0
         1364 | 26984896261 | 686 |     981760 |        1 |           0 |    31742 |
0
         1130 | 26984905972 | 620 |     866950 |        2 |           0 |    31642 |
0
         1130 | 26984905972 | 620 |    1048462 |        1 |           1 |    31642 |
0
         1173 | 26984945254 | 123 |     824399 |        2 |           1 |      412 |
0
         1173 | 26984945604 | 500 |    1131351 |        1 |           4 |      412 |
0
           98 | 26984951769 | 401 |     965138 |        2 |           3 |      337 |
0
           98 | 26984951769 | 401 |    1082185 |        1 |           0 |      337 |
0
         1172 | 26985025264 | 203 |     877180 |        1 |           2 |      396 |
0
         1172 | 26985026664 | 683 |     930917 |        2 |           2 |      396 |
0
         1172 | 26985031269 | 700 |     981760 |        1 |           0 |      396 |
0
```

The corresponding RFM table would be:

```
 household_key | recency | frequency | monetary
---------------+---------+-----------+---------
          2375 |     711 |         2 |       6
          1172 |     700 |         3 |       4
          1364 |     686 |         1 |       5
```

```
1130 |      620 |          1 |        1
1173 |      500 |          2 |        5
98   |      401 |          1 |        3
```

Bonus Exercise #1

The company for which you are prototyping this service has some clients that refuse to consider alcohol sales in their marketing campaigns. Also provide the code to calculate RFM metrics, excluding sales of products with the department of “SPIRITS”.

Bonus Exercise #2

Once the RFM data is generated, in order to be most useful, each customer is typically assigned a “score” based on the values of their individual metrics. You then would choose categories for each value to fall into, typically 3 or 5. This is usually either done with business rules or statistical quantiles (more common). Then it is very easy to identify groups of customers based on these “scores”. For example, with 3 quantiles, the highest ranked customers would score 222.

Add additional logic to your Spark application to calculate the full RFM table with scores and answer the following question:

How many customers score is “222”?

Take the RFM table generated from the base exercise and add the quantile each customer falls into. We will use 3 quantiles in a uniform split, so the ranges would be [0, 33, 66, 100].

So given:

```
household_key | recency | frequency | monetary
```

```
---------------+---------+-----------+---------
          2375 |     711 |         2 |       6
          1172 |     700 |         3 |       4
          1364 |     686 |         1 |       5
          1130 |     620 |         1 |       1
          1173 |     500 |         2 |       5
          98   |     401 |         1 |       3
```

Add the following might look like:

```
 household_key | recency | frequency | monetary | r_score | f_score | m_score
---------------+---------+-----------+----------+---------+---------+--------
          2375 |     711 |         2 |       6  |        2|        1|       2
          1172 |     700 |         3 |       4  |        2|        2|       2
          1364 |     686 |         1 |       5  |        1|        0|       1
          1130 |     620 |         1 |       1  |        1|        0|       1
          1173 |     500 |         2 |       5  |        0|        1|       0
          98   |     401 |         1 |       3  |        0|        0|       0
```

Hint: You will need to use Spark SQL stats and Spark MLLib to reliably calculate this