Manasa Vadrevu

Shutterfly Inc. | Redwood City, ca

**Customer LTV**

A Prototype

Table of Contents

[**INTRODUCTION** 2](#_Toc471674449)

[**PURPOSE OF THE APPLICATION** 2](#_Toc471674450)

[**TECHNICAL ASCPECTS** 3](#_Toc471674451)

[**HIGH LEVEL DESIGN** 3](#_Toc471674452)

[**INPUT FILE** 3](#_Toc471674453)

[**OUTPUT FILE** 4](#_Toc471674454)

[**JAVA PROGRAM** 4](#_Toc471674455)

[**UNIT TEST AND DATA QUALITY TESTS** 6](#_Toc471674456)

[**RECOMMENDATIONS** 6](#_Toc471674457)

[**APPENDIX** 7](#_Toc471674458)

# INTRODUCTION

**Customer Lifetime Value (CLV):**

Customer Lifetime Value is used to assess the monetary value of each customer. It is basically an extrapolation of the net profits a company can derive from a customer based on the present value of cash flows from the customer. It not only provides the financial value but also suggests who a customer can influence - which leads to the concept of customer lifetime influence value.

It is an important metric that impacts business decisions related to sales, marketing, product support, eventually impacting the growth and profitability of a business. For example:

1. CLV Improves customer profiling and segmentation to offer interest-based pricing strategies to boost revenues per customer.
2. The metric can help predict future product demand to serve customers better.
3. It assists in building a better brand that impacts customer loyalty and increases customer retention.
4. CLV helps to target specific markets and customers, thereby saving resources.

This application is primarily focused on calculating the customer lifetime value for Shutterfly. This is the projected revenue that customers will generate during their association with Shutterfly Inc.

A simple LTV can be calculated using the following equation:

**52 (a) (t)**

Where a is the average customer value per week (customer expenditures per visit (USD) x number of site visits per week)

t is the average customer lifespan. For Shutterfly, it is 10 years.

# PURPOSE OF THE APPLICATION

Current Application aims at achieving the following:

* Data Ingestion for the event e
* Calculating Simple Lifetime Value for Top X customers.

# TECHNICAL ASPECTS

**Programming Language:** Java

**Libraries/ Dependencies**: json-simple-1.1.1.jar

# HIGH-LEVEL DESIGN

Java Program to Ingest Event Data D and Calculate Customer LTV Values for D

Input Text File (JSON Format)

Output Text File with Calculated LTV for Top X Customers

# INPUT FILE

**Location:** Data used for this application is located in the input directory of the code repository

**Points to note:**

* The input text file will be in JSON Format. The entire file is a JSON array and each entry is a JSON object.
* It contains data for 1 week from 1/1/2017 to 1/7/2017
* This application assumes that input data frequency is 1 week. Any data out of this range should not be sourced or processed by the applications
* This application is scalable to handle data of varying periods from weekly to yearly or more based on the frequency at which the business wants to analyse.
* Amount supports float values and no Text should be supplied in the data. All amounts should be converted to USD.

There are four events captured in this data – Customer, Site Visits, Image Upload, and Orders. Please [**refer appendix**](#_APPENDIX)for detail information.

**Event scenarios captured in the data:**

* New Customers profile capture
* Profile updates by existing customers
* Site visits by new customers
* Site visits by existing customers whenever they update their profile or visit site for shopping
* Image Uploads
* New orders and updates on existing orders

**Additional scenarios covered in testing for unidentified data:**

* Invalid verbs
* Invalid events
* Null values

# OUTPUT FILE

The output file contains Simple Lifetime Values for Top X customers, where X is the number of customers.

**File format:** comma separated text file

**Fields:** CustomerID, Last Name, SimpleLTV values

**Location:** Output directory in the code repository

# JAVA PROGRAM

**Functionalities Implemented:**

1. **Ingest(e,D)**

Given event e, update data D.

Event e is a single JSON Object from the input file. Details of events are stored as <key,value> pairs which are parsed by the java program using json-simple-1.1.1.jar.

Based on the TYPE and VERB in each object, the event is identified and additional data is retrieved and ingested accordingly. Data is stored using in-memory data structure.

Please refer to the source code for details.

1. **TopXSimpleLTVCustomers(x, D)**

Explanation of the formula:

Simple LTV = **52 (a) (t)**

t = 10 years

a = a is the average customer value per week

= customer expenditures per visit (USD) x number of site visits per week

customer expenditures per visit (USD):

sum of the total amount each customer spent in the entire week/ no. of orders placed by the customer for the week

number of site visits per week:

These are captured by the site\_visits data, which give the total number of visits made by the customer for the week which includes first time visits, update profile visits, visits which lead to a sale and also visits which no not lead to a sale.

**Note:**

* The number of orders placed by the customer for the week does not double count any updates on the orders. It uniquely counts the orders placed by the customer
* Total amount spent by the customer only considers the latest amount updated by the customers. It ensures that old order details are not used in the calculation.
* The same formula can be used for different periods either monthly or quarterly or yearly. The number of weeks needs to be determined for the period and (a) value needs to be adjusted accordingly.

For example, we have data for 1 month which has 4 weeks:

a = a is the average customer value per week

= customer expenditures per visit (USD) x number of site visits per week

= [sum(total\_amount for 1 month) / total no. of order placed in the month]\*[total number of site visits for 1 month/ 4]

The output of this functionality gives SimpleLTV values for top X individual customers for one week.

(Note: For easy execution, X is set to 10, we can easily parameterize the program to take input from the user).

# UNIT TEST AND DATA QUALITY TESTS

Please see the document attached here which has main test cases and data quality test conducted on the program.



(**Note:** The code is tested for other test cases as well which are not listed here for simplicity).

# RECOMMENDATIONS

* The current data structure effectively supports the functions implemented, but in future, if more than one analytic method needs to be computed this data structure needs to be re-designed to give the best performance and also support various data types.
* The analytical functions give the best performance when they are computed only when required. As the business would have a large number of customer and transactions, computing these results every time there is transaction would not give the best performance. Unless the business users want to have a real-time analytic system, computing these functions periodically is suggested.
* Currently, the application is written in such a way that the data is not persisted. It can be expanded to store historic transactions as well.
* A NoSQL database can be used to easily work with JSON data. This database can be sourced by the program to handle large amounts of data.
* Scripts can be developed to automatically trigger the program and compute results for the current period and before next period starts. For example, a script can be made to automatically executed on 1/7/2017 at 11:59 PM for the period 1/1/2017 to 1/7/2017.
* The program is written in such a way that it can be easily scalable for
* A data model can be designed for the given data in such a way it will be easy for the business users to analyse data.
* A visualisation tool can be integrated to use the computed values and analyse customer behaviour and market segments to easily make decisions.

# APPENDIX

**Events**

Sample events the Data Warehouse collects from Shutterfly’s public sites. All events have a**key** and **event\_time** but are received with no guaranteed order and with fluctuating frequency.

**Customer**

* type
  + CUSTOMER
* verb
  + NEW
  + UPDATE
* Additional Data
  + key(customer\_id)
  + event\_time
  + last\_name
  + adr\_city
  + adr\_state

**Site Visit**

* type
  + SITE\_VISIT
* verb
  + NEW
* Additional Data
  + key(page\_id)
  + event\_time
  + customer\_id
  + tags

**Image Upload**

* type
  + IMAGE
* verb
  + UPLOAD
* Additional Data
  + key(image\_id)
  + event\_time
  + customer\_id
  + camera\_make
  + camera\_model

**Order**

* type
  + ORDER
* verb
  + NEW
  + UPDATE
* Additional Data
  + key(order\_id)
  + event\_time
  + customer\_id
  + total\_amount