**Calculating Simple LTV for top X Customers**

**Steps Taken:**

1. I have coded in python, which will run from command line. This requires two command line arguments
   1. Top X Customer which should be > 0
   2. File Name (Consisting events)
2. There are three python snippets
   1. Input.py - This is the main code which kicks off all the supporting methods.
   2. Calculate.py – This snippet is used for implementing Ingest and Calculating LTV for top X customers.
   3. Classes.py – This is used to store all the event attributes.

**Assumptions:**

1. I have considered Monday to Sunday as one week.
2. If value for top X customer is more than the customers in the event file, then I am printing all the customer’s LTV value who are in the file.
3. I am assuming that the event time in the CUSTOMER event as the start time.

**Design:**

1. I have used python dictionary (dict) to store the event details of each customer.
2. Before loading to dictionary, I have created classes for each event type and all the attributes are loaded to the class.
3. I have implemented two methods,
   1. Ingest(data, dict):

This method takes data (which is nothing but events in the file) and loads everything to my dictionary (dict). This method is implemented in calculate.py file.

* 1. topXSimpleLTVCustomers (top\_Customer, dict)

This method uses the dictionary (dict) and goes through all the customers and calculates the LTV value and prints the top X Customer Id and their corresponding LTV value.

**Error Handling:**

1. Divide by zero is handled while calculating the number of visits.

2. If the argument value for top X customer is negative then meaningful message will be displayed and it will exit the program.

3. If the number of argument passed is less than expected, then meaningful message will be displayed and it will exit the program.

**Performance:**

Since I am using dictionaries which are nothing but hash tables, these are really fast in performance. Each method would run in O(N) time.

**Improvement**:

Code could be improved by handling few more exceptions for value error and IO error. Thoughtful error messages when there is missing value/incorrect value in the events.

Also a staging table could be created to store all the events and then then could be used for calculating purpose.

**Output**:

Customer ID LTV

96f55c7d8f42 85904.0

96f55c7d8f44 65447.2

96f55c7d8f43 10524.8

I have considered 3 customers where, one customer visited 3 times, and the other customer visited 2 times and one more customer visited only once.

1. Customer\_Id: 96f55c7d8f42 made three orders 12.34+52.88+99.98 and the visits were made. Hence the LTV value is largest among other customers.
2. Customer\_Id: 96f55c7d8f44 made order amount of 125.86 and number of visit is just one.
3. Customer id: 96f55c7d8f43 made two orders with amount 6.24 and 14.00 and number of visit is 2, since amount spent is low, customer ended up with lesser LTV compared to other two.