**Featurespace Machine Learning Engineer Interview Task**

**Task Description**

In this task you are asked to implement a primitive anomaly detection system for credit card transactions.

You are provided with a .csv file which mimics a real-time credit card purchase feed. The transactions have the following format:

AccountId, MerchantId, TransactionAmount

Notice that sometimes there is data corruption, and a meaningless extra column might appear.

Your goal is to flag up anomalous transactions in real-time for further inspection by an expert. You decided to approach this problem by quantifying how much an incoming transaction deviates from the typical transaction for the customer in question.

This exercise also tests your software engineering skills, so we recommend sticking to best practices, such as clear and document code and a sufficient number of automated tests. We suggest you use Python, but you may use C++ or Java, if you prefer.

**Step 1: Data health check**

* Is your data in good shape? Note down how serious the data corruption is, and describe how you deal with this problem.

**Step 2: Simple statistics**

* To build up knowledge of the typical transaction, you accumulate some simple statistics about each customer. Write a program that reads event data from a file, tracks the mean transaction amount (η) for each customer and its standard deviation (σ), and prints it to standard output after every transaction.
* Suppose you observe a new transaction for *d* dollars. You can use η and σ to detect whether it’s an anomaly by checking if [d differs from η by at least 3 standard deviations](https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule). Update your program with this simple detector, and output any warnings into a log.
* Is your detector working reliably when it has seen very few transactions per customer? One approach to mitigate the initial lack of data is to fall back on the global statistics. Implement tracking mean transaction amount and its variance *globally* (for all customers) and update your detector to use this information when needed.

**Step 3: More statistics (optional)**

* This simple set of statistics can be extended to track the minimum and maximum transaction value, as well as the ratio between the current value and the historical minimum. Extend your program to track these 3 quantities.

**Please submit:**

* Your program source code