**Affirm LoanBalancer Write-up**

**How long did you spend working on the problem? What did you find to be the most difficult part?**

It took me about 1-2 hours to understand the problem with all scenarios

And 2 hours to code minimal working solution and took additional hour to clean up the code and test.

**How would you modify your data model or code to account for an eventual introduction of new, as­of­yet unknown types of covenants, beyond just maximum default likelihood and state restrictions?**

I will go with Rules Engine to maintain all covenants

**How would you architect your solution as a production service wherein new facilities can be introduced at arbitrary points in time? Assume these facilities become available by the finance team emailing your team and describing the addition with a new set of CSVs.**

I will create a lambda function to get the file then store it in S3 bucket and create generic mapping (canonical model) with all fields. Another lambda function to validate(null/mandatory field checks, format checks, size checks etc), transform, persist all these fields in database(preferably elastic search).

**Your solution most likely simulates the streaming process by directly calling a method in your code to process the loans inside of a for loop. What would a REST API look like for this same service? Stakeholders using the API will need, at a minimum, to be able to request a loan be assigned to a facility, and read the funding status of a loan, as well as query the capacities remaining in facilities.**

Rest API swagger contract is available in github

**How might you improve your assignment algorithm if you were permitted to assign loans in batch rather than streaming? We are not looking for code here, but pseudo code or description of a revised algorithm appreciated.**

To maximize the profit, I will go with best fit algorithm

Pesudo code steps

* Input list of facility and loans
* Initialize all loans as free.
* Start by picking each loan and find the

minimum facility capacity that can be assigned to

current loan i.e., find min(facilityCapacity[1],

facilityCapacity[2],.....facilityCapacity[n]) >

facilityCapacity[current], if found then assign it to the current loan.

If not, then leave that loan and keep checking the further loan.

I will persist this data into elastic indexes and search the respective facility for all the loans. This avoids all for loop logic.

**Discuss your solution’s runtime complexity.**

O(m+n+k) where m is list of loans, n is list of facilities and k is list of covenants