Write-Up: Questionnaire

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

MG: It took me 4.5hrs to come up with this first basic version of solution and toughest part I felt was to model object's while already thinking about production ready design. Since it is drive by csv's and mostly getting executed with no involvement of upstream middleware components it was hard to switch gears from that my mindset to playground.

2. 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?

MG: I was thinking of templating the covenants with easily pluggable configurations that are frequently altered by facility admins using admin utilities. As loan processing engine these are very much like decision rules, so processor itself has to be made interactive to poll on seeded rules data store to refresh its current state while processing in flight events. I can think of something parallel to zookeper, etcd or elasticsearch or its complements that can be implemented fairly easily to achieve this objective. This is assuming the fact that eventual consistency of all nodes is something that is acceptable, hard cutover addressed in following question.

3. 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.

MG: Maximizing the profits are primary objective and since introducing facilities directly impacts yield i'm of the opinion to pause on loan stream processing by having subscribers re-synch the facilities from source data store where admin has on-boarded new. If pausing streams is not an option then based on business guidance and with appropriate acceptable circuit breaker / retry implementation on services integration can be flipped over to a newer facilities stack. It can be achieved by distributed properties mgmt, or api gateway CNAME switch or gradually draining of live servers using blue/green deployment of upstream components.

4. 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.

MG: API workflow documented in README
here https://github.com/checkmg/loanbks/blob/master/README.md .
LoanProcessingApi will be hosting an endpoint for users to post loan application that instruments LoanFundingProcessor upstream with external datasources and middleware to provide Funding status, and capacity information to users. To keep this highly scalable and lightweight backing the rest endpoints with event streams and webhooks would help enable guaranteed processing FIFO and ready available status.

5. 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.

MG: In batch we have a handle on all loan buckets upfront and there is no FIFO condition, so multiple dimensions of comparison to find optimal capacity facility that funds loan with max yield.

6. Because a number of facilities share the same interest rate, it's possible that there are a number of equally appealing options by the algorithm we recommended you use (assigning to the cheapest facility). How might you tiebreak facilities with equal interest rates, with the goal being to maximize the likelihood that future loans streaming in will be
assignable to some facility?

MG: Adjust logic to consider balance capacity maximizes profit.