1. How long did you spend working on the problem? What did you find to be the most difficult part?
I spent about 3 hours working on this. The part that was the most difficult for me, was making sure I understood the directions correctly. I had initially thought that I needed to break up each covenant line with a banned state into two lines (one with a banned state, and one without) , but after checking my answers against the provided answers I ended up having to ignore that part of the directions to get the required answers. I am still a bit uncertain about this part, but also running out of time.

2. How would you modify your data model or code to account for an eventual introduction of new, as-of-yet unknown types of covenants, beyond just maximum default likelihood and state restrictions?
I think since I have the covenants encapsulated in it's own class, it would be not hard to add another property and then modify the "allow_loan" method to include this new logic.

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.

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.
The stakeholder would need to post all the information provided in the loans.csv line, and we would add the loan to a loan queue (assuming we must still go in order of loan received), we would initially return a promise.
The loan_stream_assignment method would need to be modified to also resolve the promise when we get to the loan, and return the facility (pre putting it into the facility_loan dictionary).

At this point, it would probably be best to turn our dictionaries into actual databases. With the current situation, one could call a GET with a facility id and get the facility.amount back, by looking it up by id in the facilities dictionary.

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.

If we were working in batches we could
-first calculated the expected yield for each loan for each possible facility (one that has covenants that would allow the loan from given state, default possibility and any other requirements)
- we could then order the loans by maximum possible yield

- we would try to fund the loans in this new order by yield, when the maximum yield is not possible (facility no longer has the capacity), move onto second possible yield facility for that loan

This new algorithm would ensure that loans with the biggest possible yields get funded first, maximizing profit.

6. Discuss your solution's runtime complexity.
-In the beginning we loop through every facility and covenant once. *Linear O(length of files)*
-My current solution loops through all the loans once, O(length of loans_files * (length_of_covenant + facilities)
-For every loan we select all possible covenants and facilities (once again having to loop though the covenants, and facilities) Linear
- the funding and recording facility is constant time
- writing the assignments output, will have to loop through the facility loan dictionary, in the best case all loans are funded by one facility in the worst case, this would once again loop through all the facilities , this also loops through all the loans
- writing the assignments will loop through facility loan dictionary, (once again in the best case all loans are funded by one facility…) and then loop through every loan, and then loop through loans to sum them up.

I could at optimize, by not having to loop through everything again when I write assignments and write them to the csv file as we calculate them (don't have time to actually make changes, though)