Data Challenge @ RTR

Problem

Your goal is to created a minimal viable product (MVP) for a fit personalization algorithm: can you rank inventory by how likely it is to fit a given client?

Attached are a relational database schematic (db_schematic.pdf) and a link to zip file containing csv files for each table shown in that schematic (data.zip). The tables contain information relating to a subset of our subscription clients (users), inventory that is available for these clients (styles and skus), and how clients interacted with that inventory (which skus did a user rent?, how did they rate their experience?, etc.).

Questions

1. Discuss different approaches you could take to estimate which inventory fits a given client. Explain which data sources you considered and which approaches/algorithms are possible. Bonus points for using descriptive statistics to understand trade-offs between different approaches.
2. Choose the approach you think is best suited for an MVP. The goal is to find an algorithm that is relatively easy to implement, and still provides significant value for our customers. Implement the algorithm as a function that returns an orderd list of styles for each user `fit_reco(user_id) = ["CEL13_1", "VIN69_XS", ...]`. In that ordered list, the first sku is our best fit recommendation for the client, and skus lower on the list are expected to fit less well.
3. [Optional] Discuss how you would implement this algorithm in production. How would you decide when to show a row with this recommendation? How would you populate that row? Where would you place the row relative to other rows?
4. [Optional] If you had to improve on your MVP where would you start? Would you want different inputs (how would you collect them?) or make improvements to your approach or algorithm?

Format of solution

Both programming language (R, python, etc) and presentation format (Rmd, jupyter notebook, pdf file, scripts hosted on github) are completely up to you. Please describe your thought process as part of your solution (as text or comments). Bonus points for making your code easily reproducible.