**Bar-Beer-Drinker PLUS**

Your objective will be to extend significantly the famous db scheme with new attributes and new relations. It has to be realistic and interesting.

Imagine you are pitching a new company to investors. You have unique data (!) which will make a good business case to bar owners, drinkers or beer manufacturers or whoever else you want to add.

You should start from our Bar-Beer-Drinker ER diagram and the 6 tables we have. But add more tables to introduce more relationships such as facebook friends, detailed records of bar activity every day (several years ago someone even had the record of bar fights). You can include type of music played and live bands, you can add who was the bartender and measure their sales record. Finally, you look at return on advertising – assuming a bar decided to buy google ads or TV ads. You can assume any data including who ordered which beer at what time (i.e. all transactions) You can make it whatever you want but you should at least have 6 tables we have right now with many more tuples (this would be a C though 😊 if you just did this)

Once you settle of the db scheme, you will have to populate it with realistic tuples. By realistic, I mean names of bars, drinker names, dollar figures etc when appropriate. No a1, b1, c1! No drinker X and drinker Y! Generate and load your db with the large number of synthetic tuples, may be 10,000? May be more. It is your choice. But your instances should not be completely random. On the contrary – you should think of a few patterns which will have business significance. For example, the larger the bar, the less it charges for a beer (most of the time). The older the drinker, the less frequently s/he goes to a bar. Drinkers go to local bars, not like what we have in AWS database instance where New York based drinker frequents a bar in San Francisco. Evening beer sales are higher than afternoon beer sales. Embed a number of such patterns in your data and construct SQL queries which would prove that indeed your db satisfies such pattern. Your patterns should be intuitive and have actionable business value either for a bar or for a drinker.

CONSTRAINTS: Enforce a couple of functional dependencies, foreign keys, any other constraints which will be imposed when you update the data – there should be part of GUI which allows us to enter a new tuple to any of the tables – but only when **constraints are not violated**.

I suggest creating your initial data by a program which outputs a csv file and than loading the database with the csv files. But then you should allow updates from the GUI.

AUDIENCE: Can be drinkers, can be bar owners, can be beer manufacturers.

QUERIES: Must be valuable for your audience. You should have up to 5 valuable and nontrivial queries preferably involving multiple tables. Feel free to consider aggregate queries.