Our project wasn't usual from start, because while others were trying to create projects that were satisfying project complexity by creating applications around the data, we decided to create a project that consists of the queries complexity itself. So basically, all we just needed is any huge database that we can play around and work with, and here was the first place where we had made a mistake: we had chosen too complicated database, and were suffering with creating UML and ER diagrams around it, so diagrams became a mess. Second mistake was using PostgreSQL database, as it had been discovered, some participants of the group faced a lot of challenges with installing and importing data to this database, furthermore, when we had been creating the frontend and backend part of our project, we understood that it's easy to start new project in Django with postgresql, but it was actually hard to migrate existing database with postsgresql to Django project. So basically what we did is just used Django helpers for migrating existing database with Sqlite, and wrote code around Sqlite database.

Now we have a project - place were people can solve the sql puzzles by running queries on website, but how it actually works? The hardest part was thinking about how to create sql compiler that can accept any SQL dialect and compile it, to check if users answer is correct. We ended up with the naive SQL compiler implementation, where we store in our database the right query, and backend can only receive Sqlite SQL answer dialect. When users presses submit, we are catching submitted sql answer, using inner python tools we are trying to run raw sql on our database, then we are running correct sql answer stored for the question, and comparing the answer strings, if they match user had submitted the correct answer, if not, he had made a mistake. Creating the contests was complete headache and still under the progress, as far as I see it should work like this, someone presses create contest -> he chooses the questions from the list that contest contains of -> we create the contest record and generate unique link by which users can connect to the contest and solve problems while contest session exists, there is no any Auth system in the project, so basically anyone can join.

Now we need to fill our website with the queries that can satisfy project complexity, so basically we were going through Hackerrank SQL questions, and were adopting them for our database in the most cases. At the same time we had 15 really complex queries that we created in the phase 5, that we should create questions of. Problem there was that question titles that we created on phase 3 were too abstract. For example: "Can we determine patterns in gaining "Best manager" awards?", there is no only one absolutely right sql query that can answer this questions, so we've needed to play around and find our own definitions, what means to be Best manager in our database.

There also were some difficulties with translating RA operators, and calculating the queries speed performance, but fortunately, we successfully overcame this phases.