Optimization Process

At the start we first worked on the database structure. We changed all ints to smallints which took less space, we took a look into tinyints which is used in SQL, but is not applicable in Postgres. After that we made an overview of what we needed in every query, in order to get an idea of what we wanted to improve and what materialized views and indices to make. Testing the initial queries appeared to be more troubling than expected due to the huge time it took for queries to succeed (e.g. query 7 would take 42 minutes). We therefore tried to create a materialized view on the passed grades, since we would need this to find GPA, Active, Complete, and highest grades. This took about 4 minutes and creating an index took 2 minutes, and we did not have that time. Which is why we tried to pass this to the loading time by splitting CourseRegistrations horizontale in passed, failed and dropped out and afterwards recombine it with a view. This wasn't advised, and it was told that creating the views shouldn't take that long on the actual servers.

When the servers came online we tried to test our settings and only got 8000 points back, and we lost a lot of time trying to correct query 1, after contact it appeared we had a small mistake in our table creation which wasn't using the optimisation of space (char vs varchar, in Postgres there is no added benefit in char). After that we optimized per query and test every query on the server by using more materialized views and indices. Unfortunately, we did not succeed in getting the desired results for all the queries according to the chatbot.

Chosen Optimizations

1. Materialized view all_courses_passed, improved query 1, 2, 3, 4, 5, 6, 7
   a. Takes 2.4GB of space, q1 goes from 15s to 50ms on Thymen's computer.
   b. Index has been made on this view on (StudentDegree, DegreeId)
2. Materialized view gpa_active_complete, improves query 2, 3, 7
   a. Takes 300MB of space
3. Index on CourseRegistrations(Grade) to improve query 6