Documentation

At first we only worked with the very basic tables we were given. We did not index or create views. We only created simple primary keys, these generally were the IDs in a table (studentID, teacherID, courseID, etc.), since these were used a lot to join tables. Our queries ran super slow because of this and we were required to optimize our database.
We experimented with different indexes, but the only ones we got significant differences in speed from was by creating a hash index on StudentID in the StudentRegistrationsToDegrees table.
After that we decided that we needed our data to be more accessible because for each question we had to join several tables. Therefore, we decided to add the coursenames to the CourseOffers table and to add studentID and courseID to the CourseRegistrations table.
Also, we decided to make some materialized views in our database, of which we will explain final versions in part two of the documentation, where we will make use of them. We first created a materialized view with the amount of credits per student, there was nothing specifically wrong with this, but we thought of more information we could put in this view. next we created one that stored all data of passed courses of students, the problem with this was that it took too much disk space. We also had a materialized view that contained the max grade of each course, however, this was more than we needed as we only needed the ones from quartile 1 in 2018. We also had one that contained all students that never failed a course, however this was slow and could be sped up by doing it later in the process, since it would have to be merged later anyways.

Like we stated in the first part, at the beginning we tried to make the queries work without making changes to our database (views, indexes). This caused the queries to take too long, which made us start from scratch after we made changes to our database. We will go through the queries one by one:
For **q1** we started with joining two tables together, and then selecting the proper studentregistrationID, as we realized that joining all of this together was a big task. We did however see that we only needed to return values from one table, so we decided that we look up the value of sregID in one table, and return them from the other, which ended up being significantly faster (about 5 seconds per query).
For **q2** we made a materialized view with all students that got their degree without ever failing a course. We did this by first making an intermediate view with all students that got their degree (14MB), indexed on studentregistrationID, the view contained GPA and the sum of ECTs of students. This cost about 2 minutes to make, which is long, but it improves the performance of the query significantly, we started at a running time of about 10 minutes, down to 3 seconds.
For **q3** we made a materialized view with all active students and combined it with the table students. Creating this materialized view takes about 2 and a half minutes on our VM's and is 270MB.
For **q4** we join 3 tables together and execute the query, since this was already fast enough we did not do any further optimizations.
For **q5** we were unable to come up with better optimisations for this query, but the query runs and we were pleased with it taking less than a minute, so we left it as it is. Right now this is our slowest query and there is quite some room for improvement for this.
For **q6** we create a materialized view containing the max grades(192kb) of 2018 quartile 1 grouped by courseofferID, which takes about 25 seconds to make. Also, we use a subquery for this where we count the amount of courses where a student is excellent, as this is faster than using a having-clause.
For **q7** we tried reusing the activestudent view, however this ended up in an error. Therefore we made a virtual table (view) that's called activestudents. We also use a view that contains the GPA and the total ECTs of each student, which was also used to make the view in q2, it takes up 581 MB and costs about 2 minutes to make.
For **q8** we made several views, one with the amount of students per course, one with the amount of student assistants per course and one that combines them looking at which of them has too many people. Joining together many tables and then querying in them is not efficient so we decided that making views was our best option. This query only takes 9 seconds. Every version that didn't use views ran into a memory error so we are happy with this result.