Personal Database Table Setup

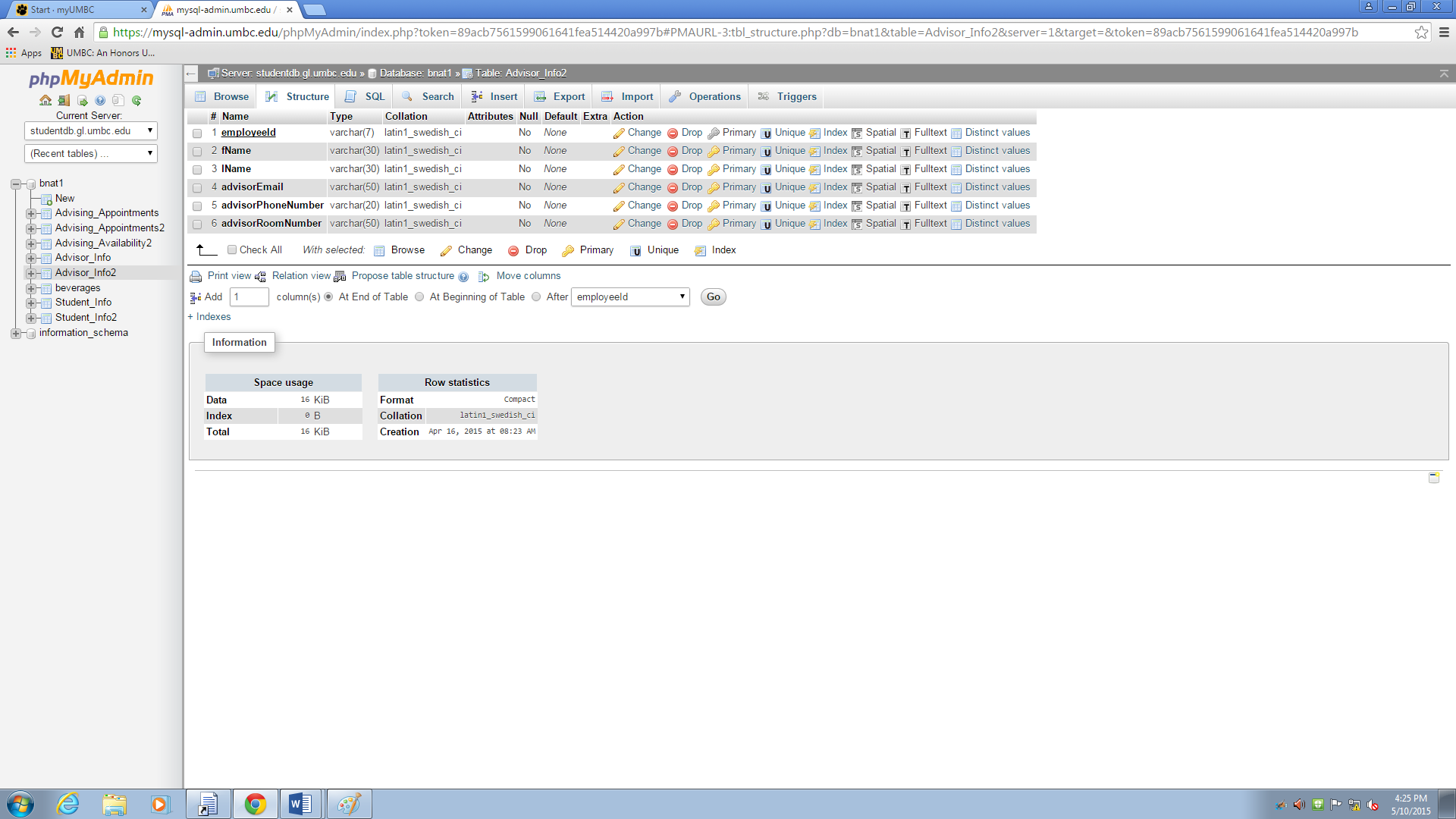
Nat’s Tables from the first project had a setup that was very easy to use, to create the desired functionality of the second project. We just had to create one more table, and one more column for the current functionality. In the first project, the database tables included Student\_Info, Advisor\_Info, and Advising\_Appointments. The student info table held all of the student’s information that they enter: first name, last name, major, student id, and major. The Advisor info table had just the Advisors’ names, and fake id numbers. We treated group appointments like a separate advisor, and this made the project easier to implement, instead of using every advisor for a group appointment. The advising appointments table held the student’s id, the advisor’s id, and the date and time of the appointment. The student Id and employee id were the primary keys for Student\_Info and Advisor\_Info, respectively.

That is how the tables were set up at first for project 1, and we only had to change a few things to implement project 2. We created the table Advising\_Availability so that when an advisor chooses a time for an appointment, this table holds the entry. The columns in the advising availability table were the advisor’s id, the datetime of the available time, and a column with default value of NULL: major. The major column is only used for group appointments, so an advisor can specify certain group times for certain majors. The only other change to the tables from project 1 were adding the email column to Student\_Info, and adding more of the advisor’s information as columns in Advisor\_Info, such as email, room number, and phone number. All of the ID columns that are not in the Student\_Info or Advisor\_Info tables are foreign keys to the IDs in Student\_Info and Advisor\_Info, so that an appointment doesn’t get damaged if something were to happen in to a student or advisors’ entry. When we create an appointment with a new student, the student’s info is entered into Student\_Info2. But if the student already exists in the database, their information is not updated; we used ‘ON DUPLICATE KEY UPDATE `studentId` = `studentId`. This pretty much makes the first part of the query not execute. If there’s a duplicate entry, it sets the key to itself. In order to support a new advisor, the administrator of the database would just have to insert a new row with the advisor’s information into Advisor\_Info.

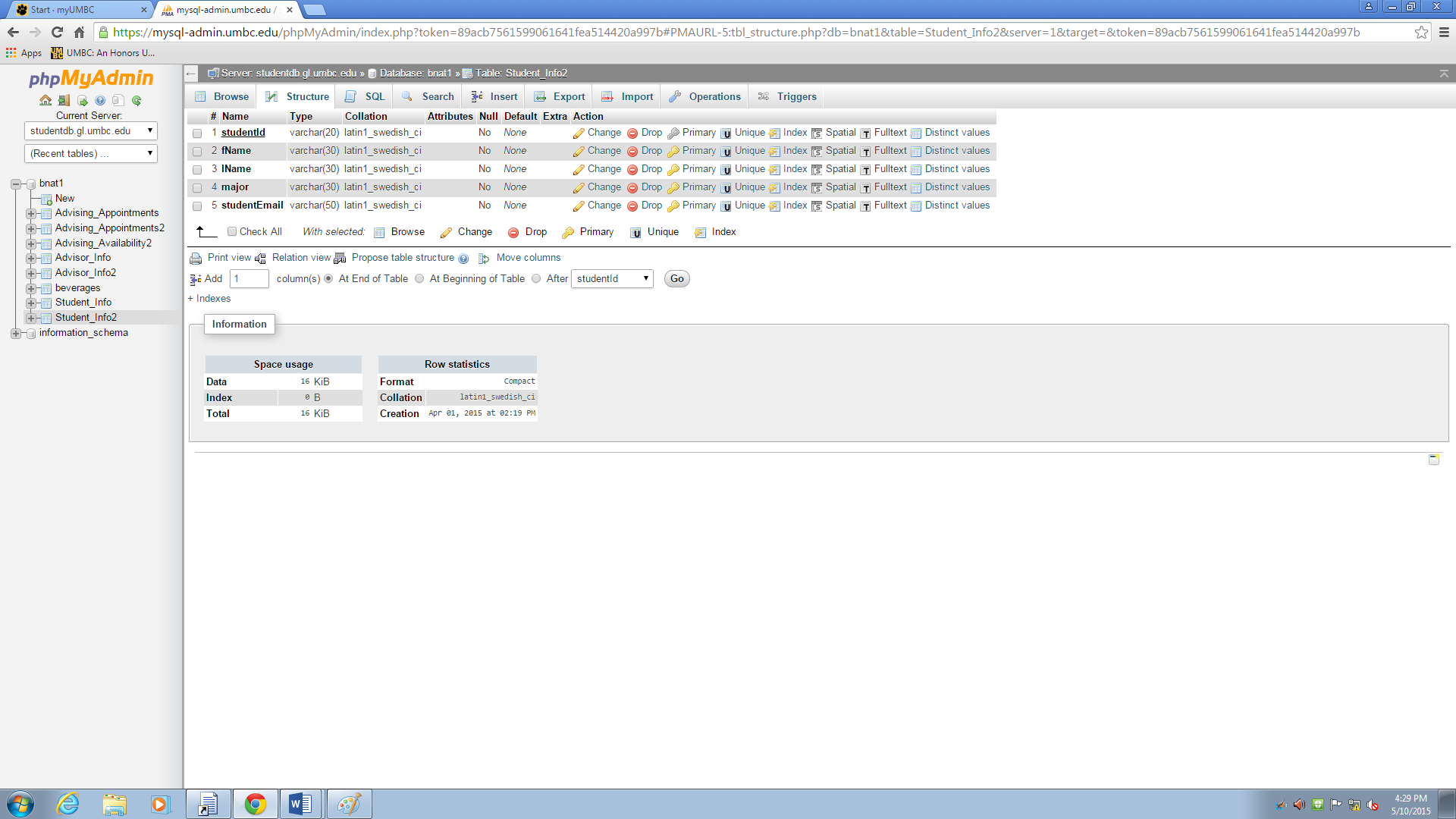
We decided to label the table names with a ‘2’ at the end to distinguish the tables from the first project’s tables, since we used only Nat’s database as our database to include all of these tables.

Screen Captures

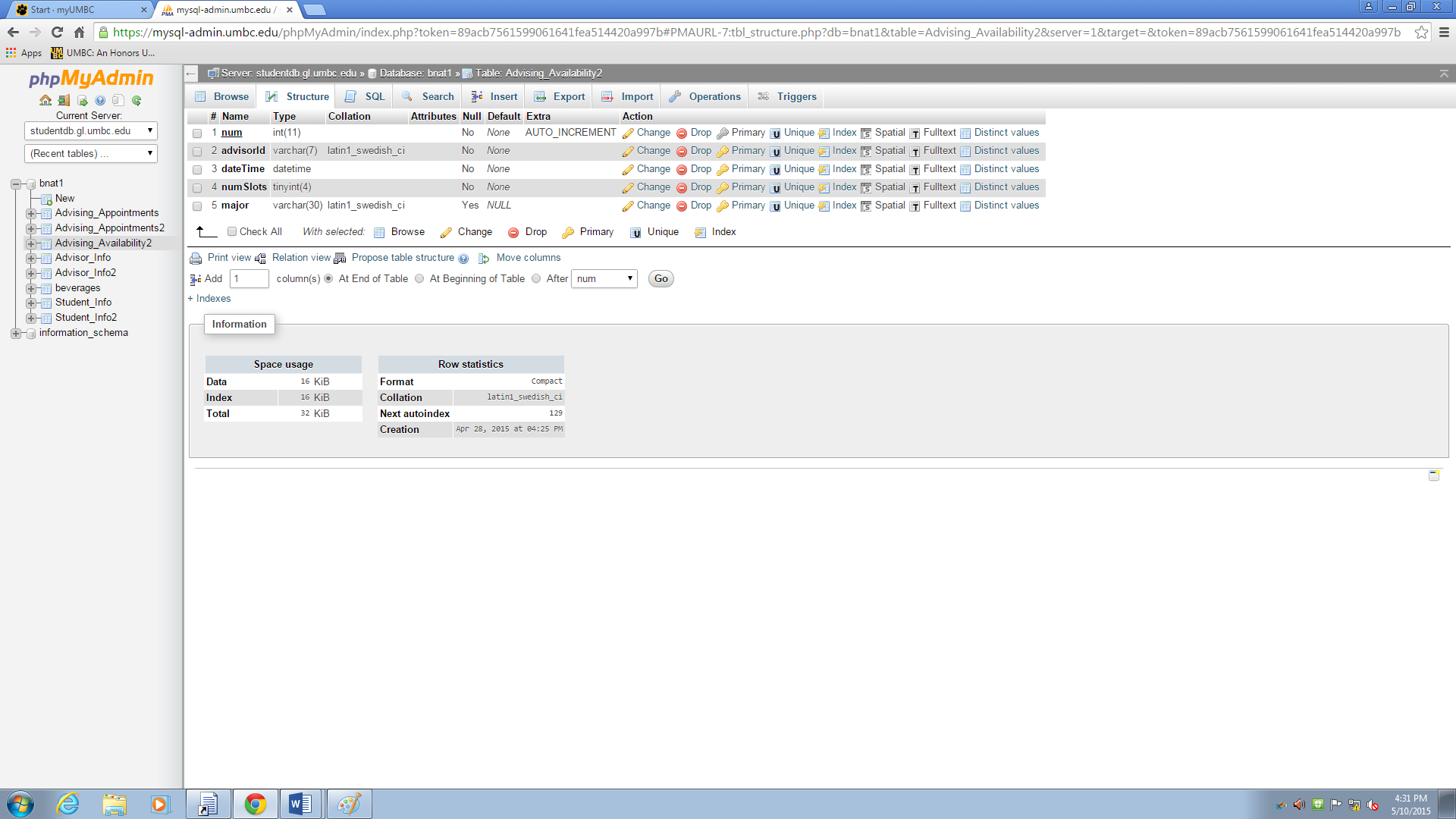
Structure view of Advisor\_Info2



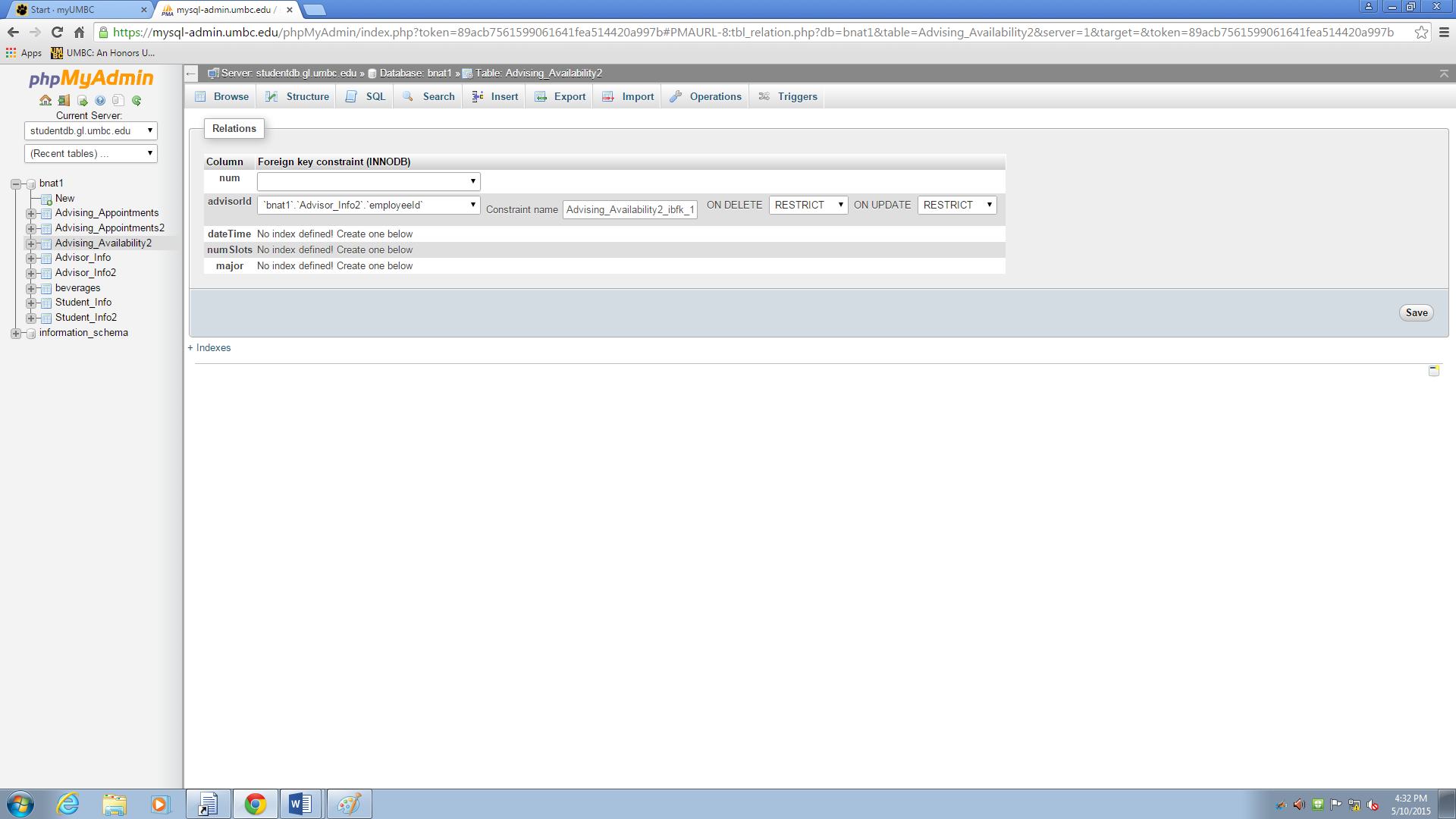
Structure view of Student\_Info2



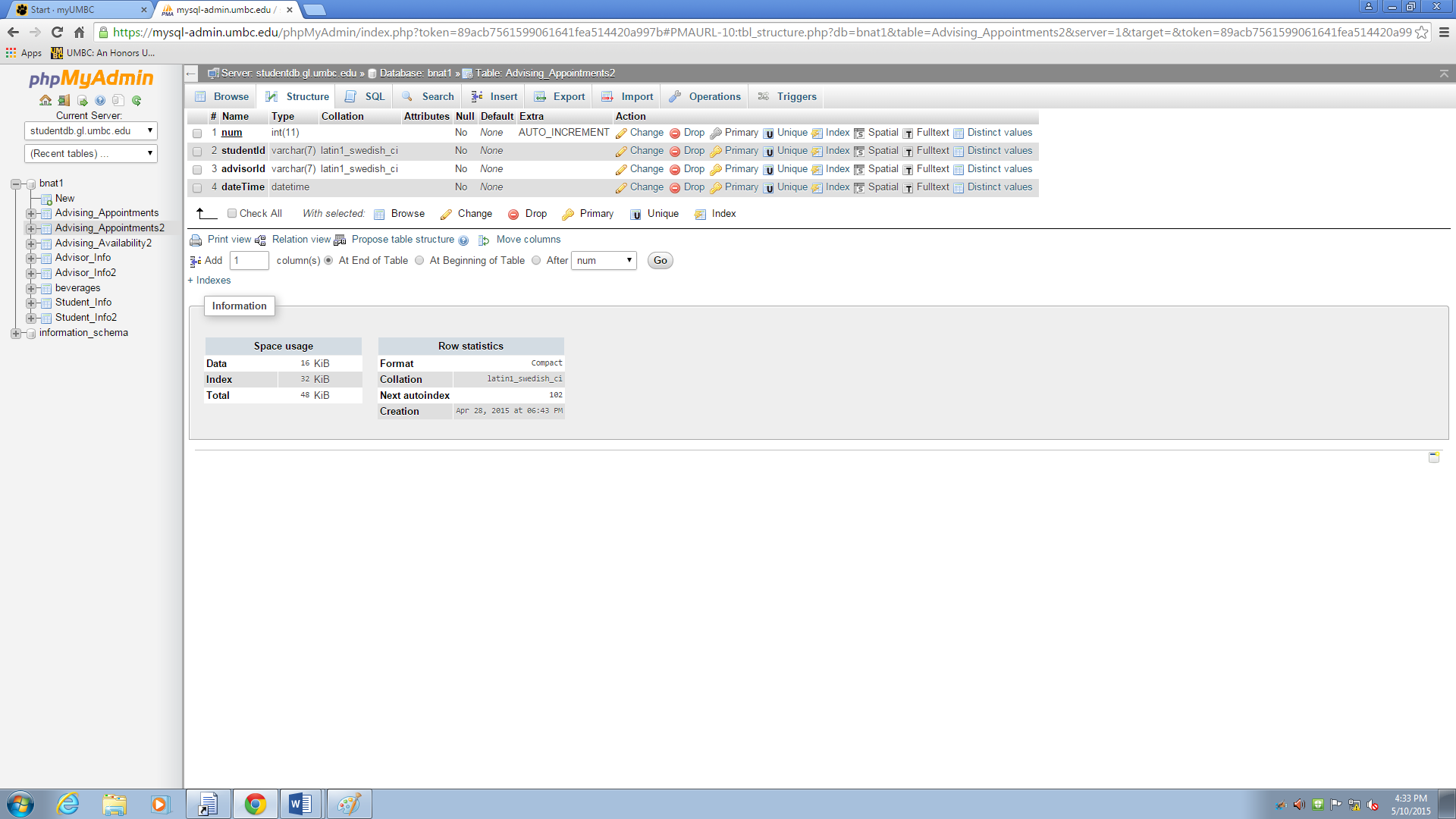
Structure view of Advising\_Availability2



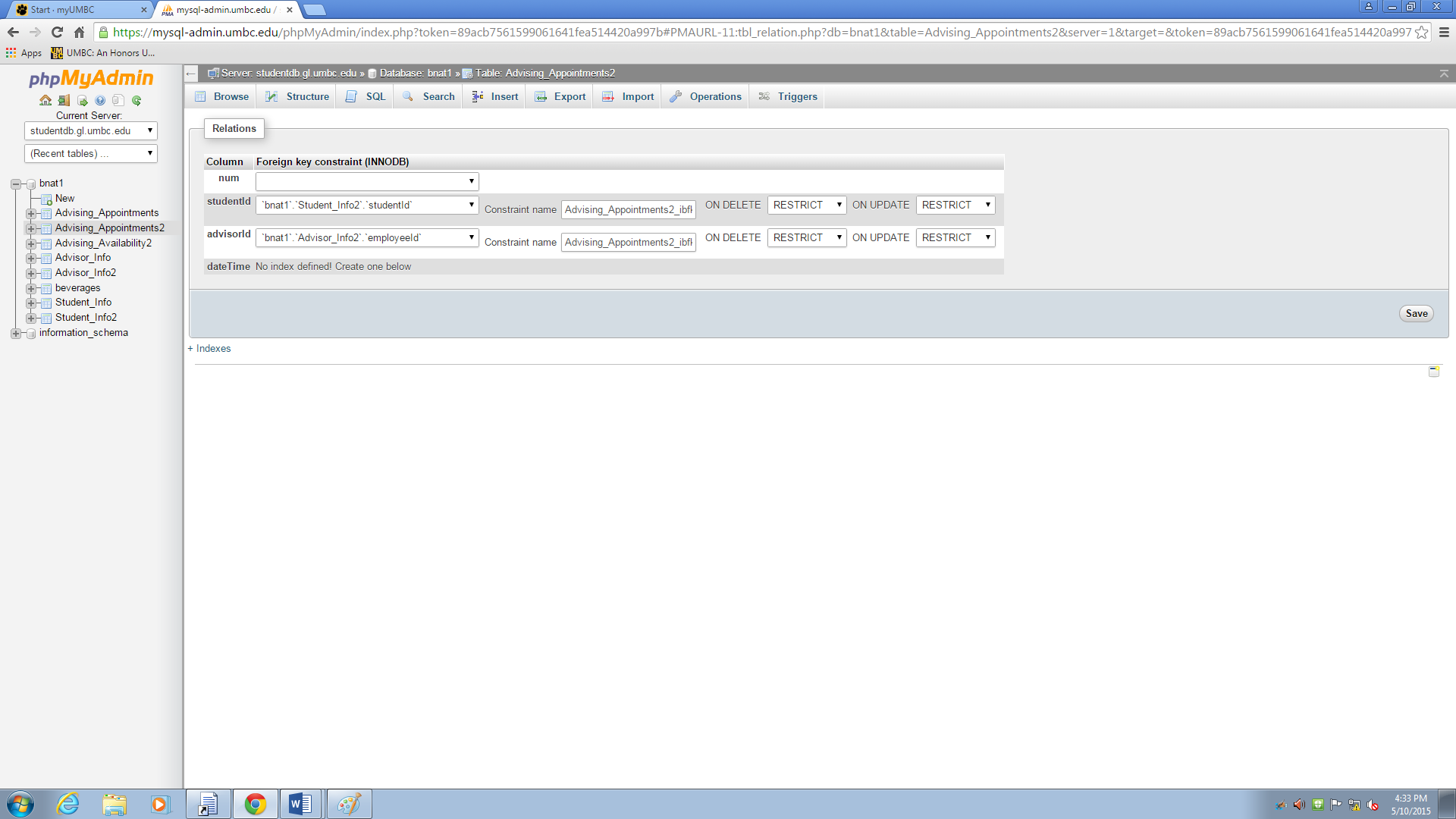
Relation view of Advising\_Availability2



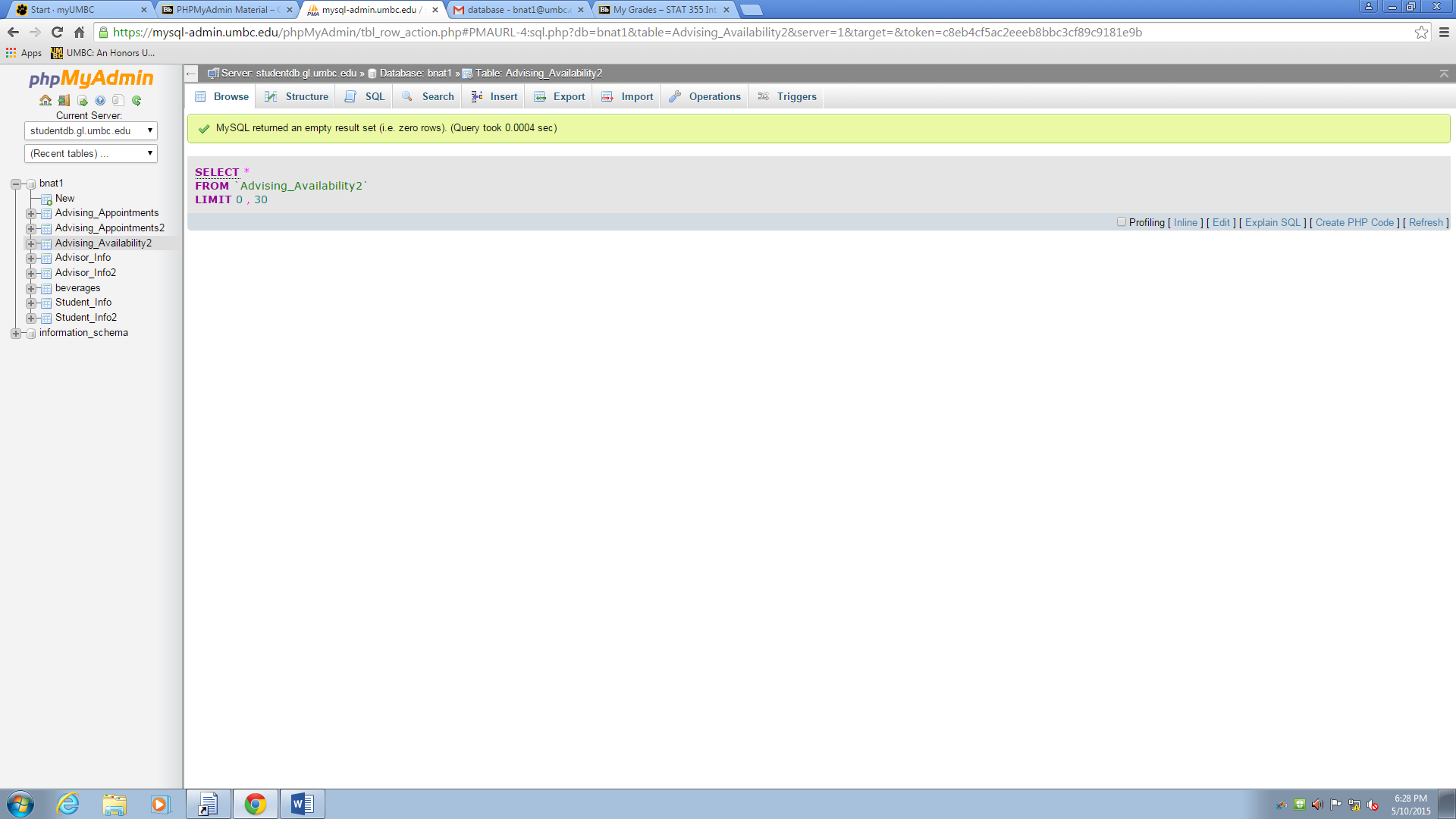
Structure view of Advising\_Appointments2



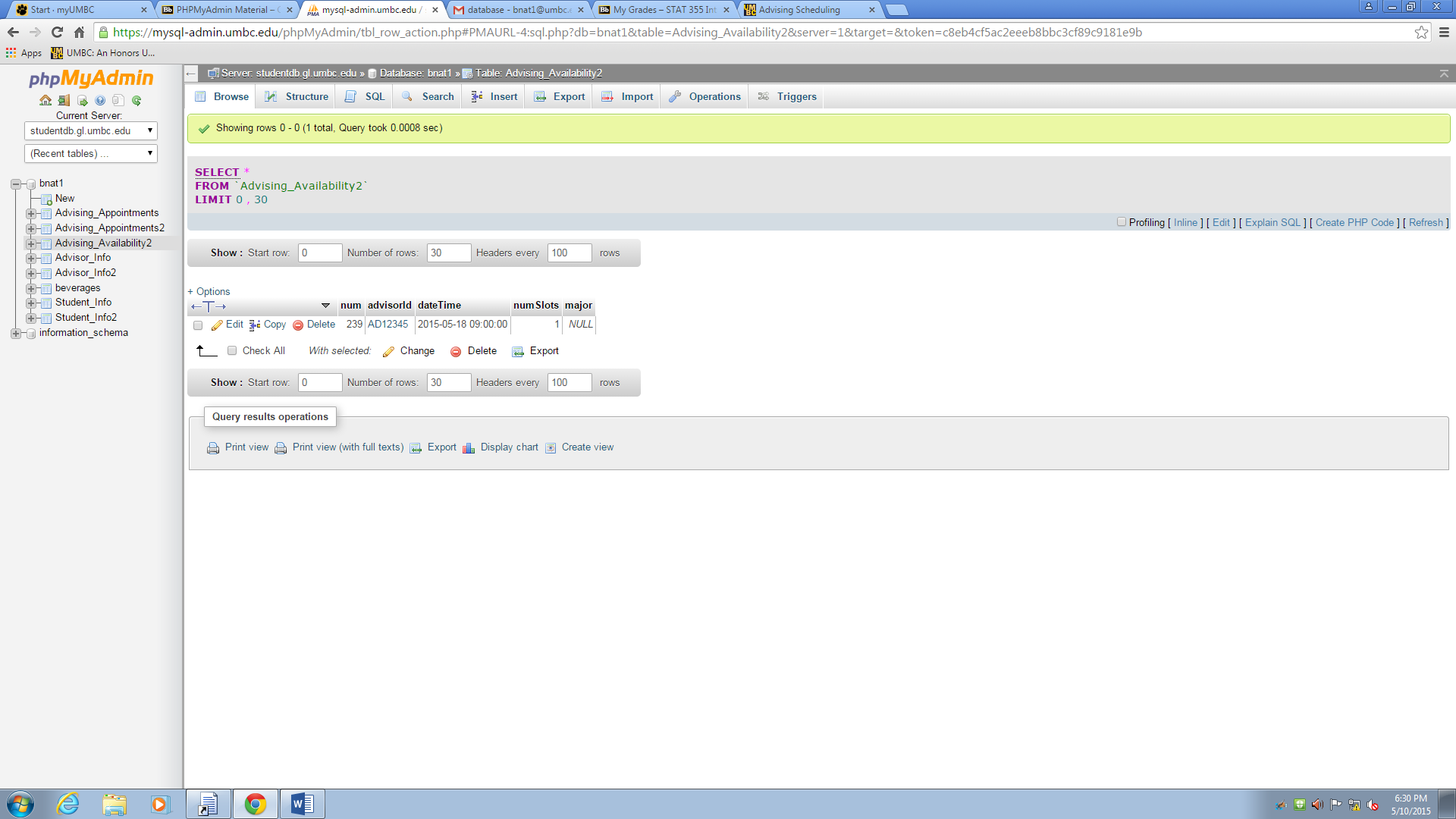
Relation view of Advising\_Appointments2



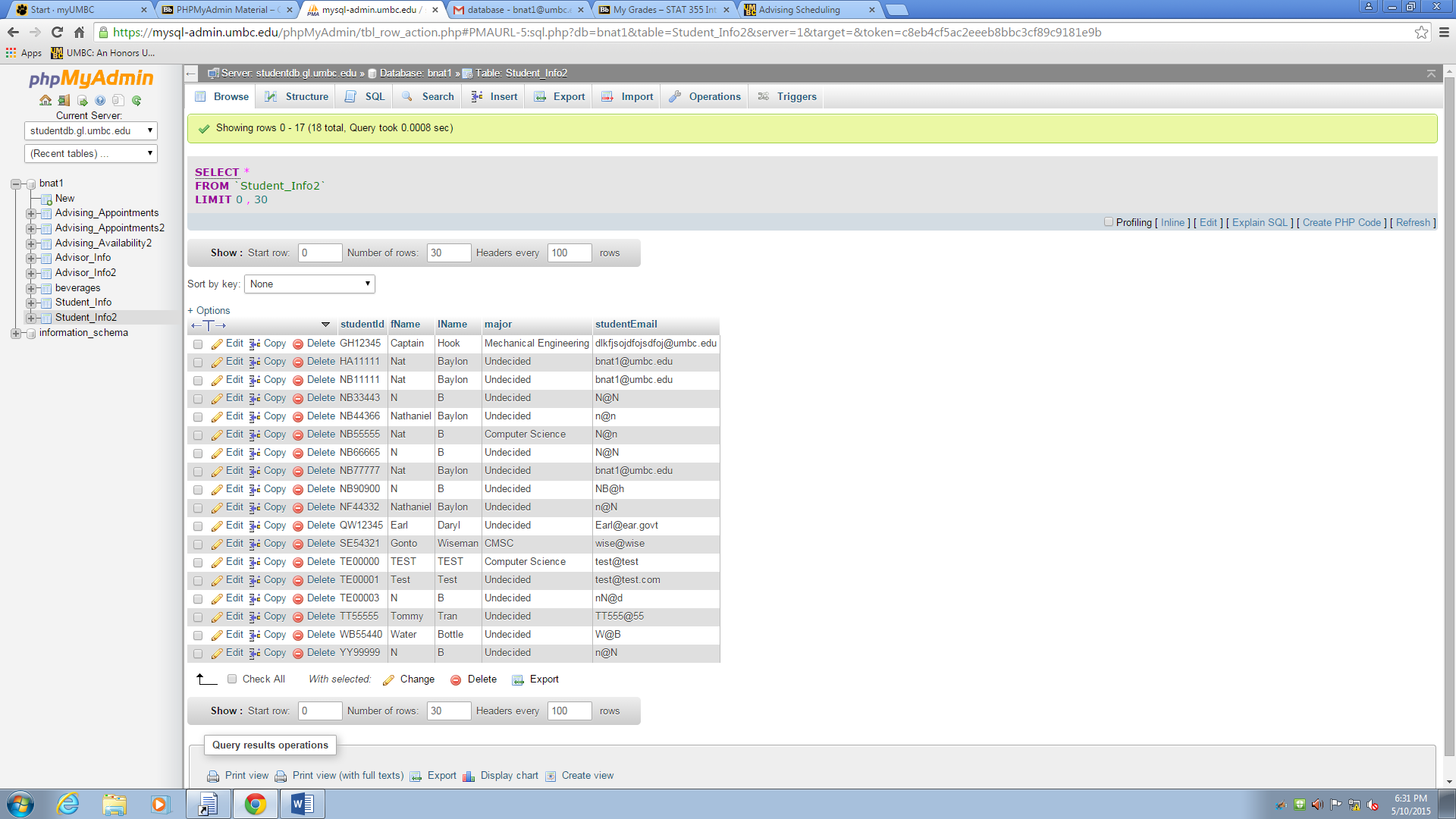
Advising\_Availability2: Browse view before making an individual appointment time available



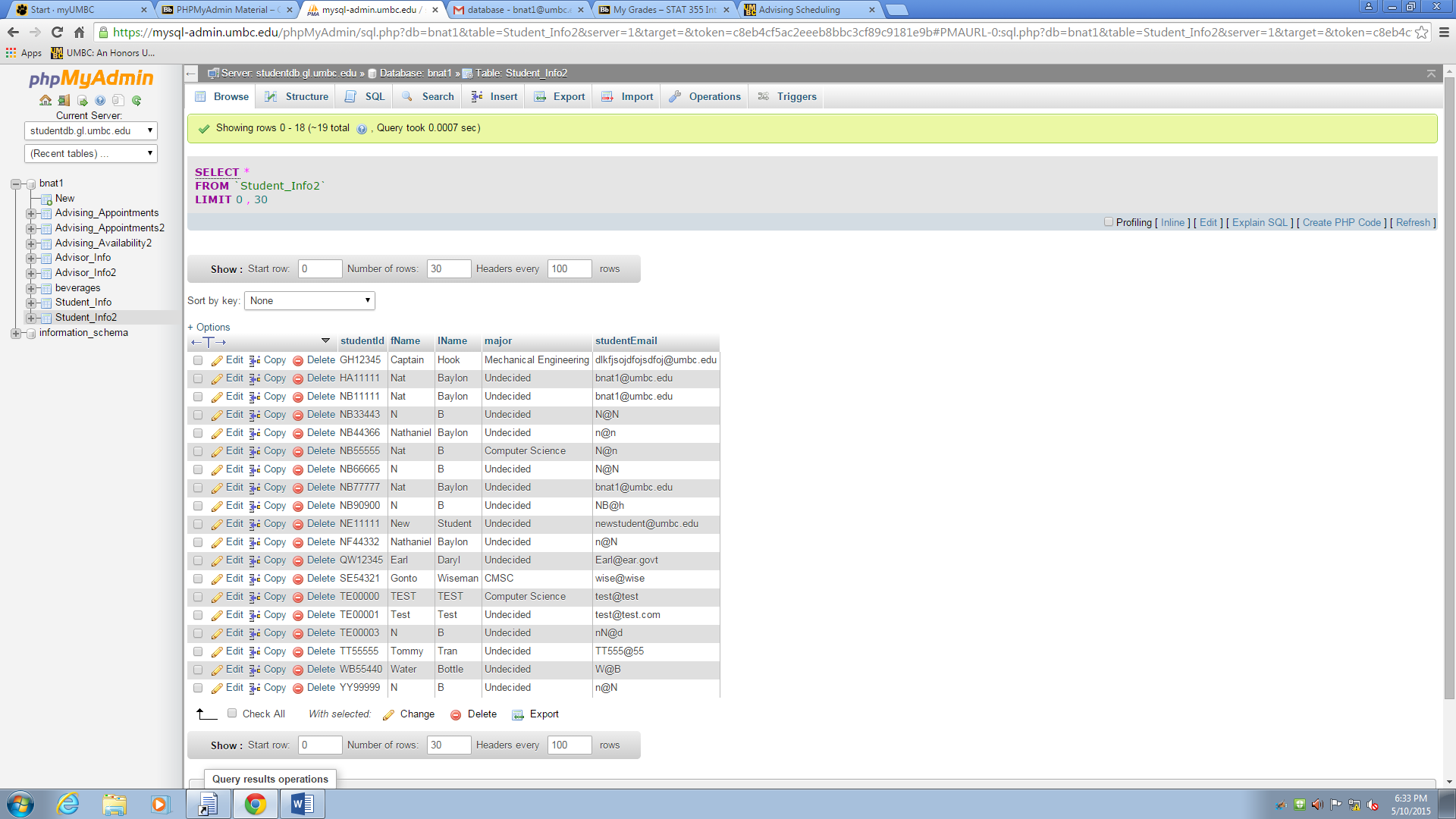
Advising\_Availability2: Browse view after making an individual appointment time available



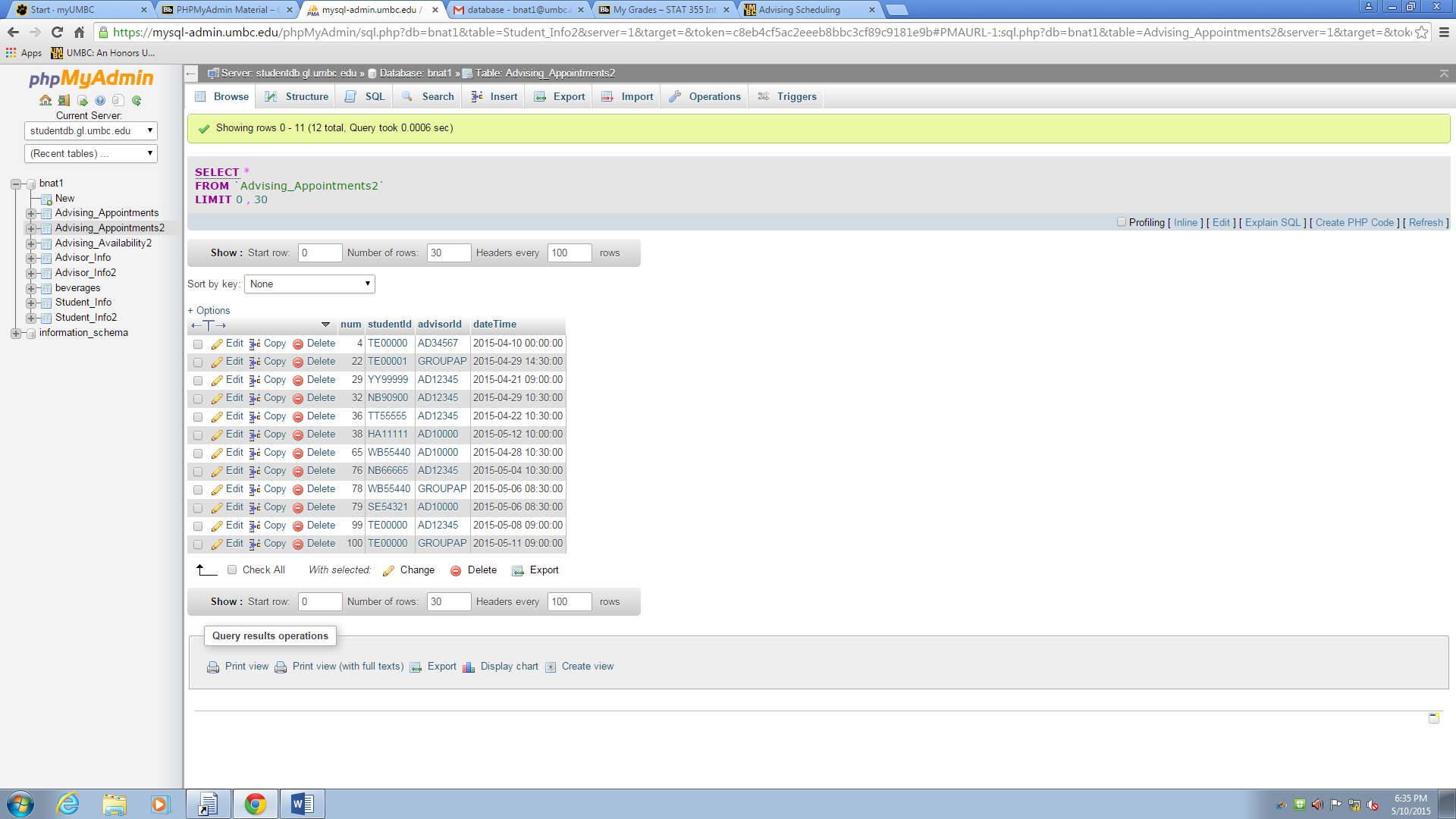
Student\_Info2: Browse view before making appointment with new student with name: New Student



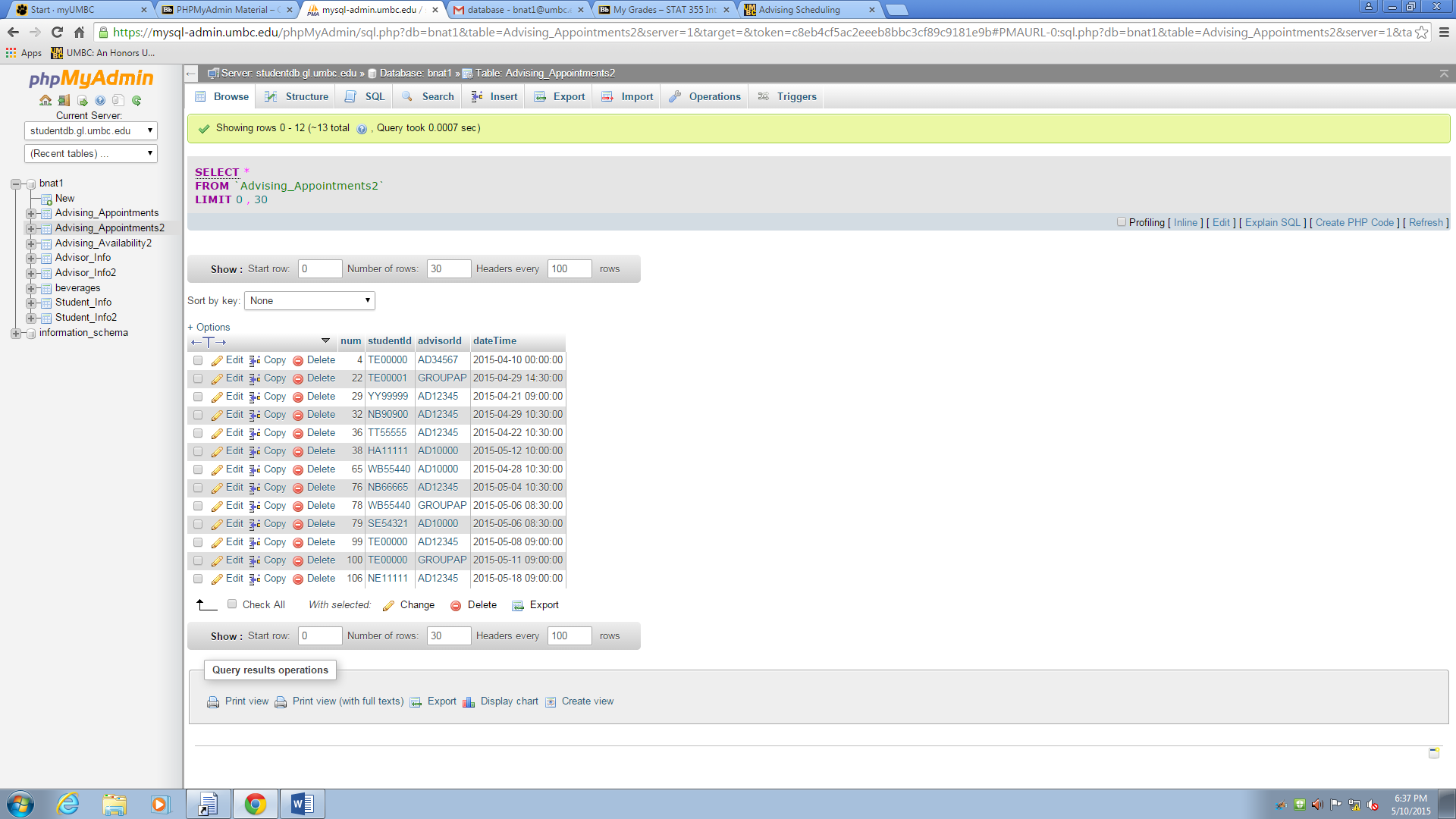
Student\_Info2: Browse view after making appointment with New Student



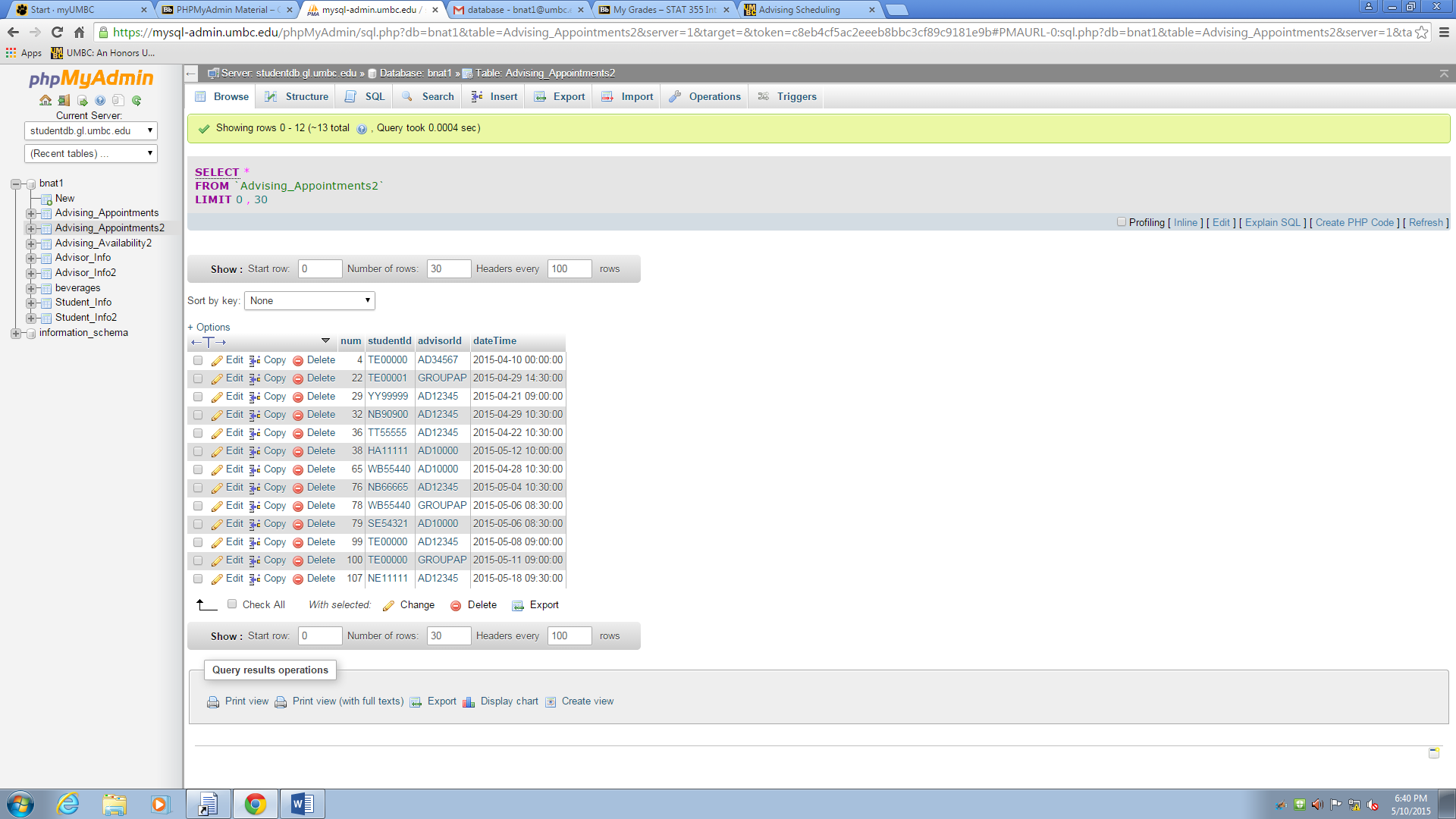
Advising\_Appointments2: Browse view before creating appointment with New Student



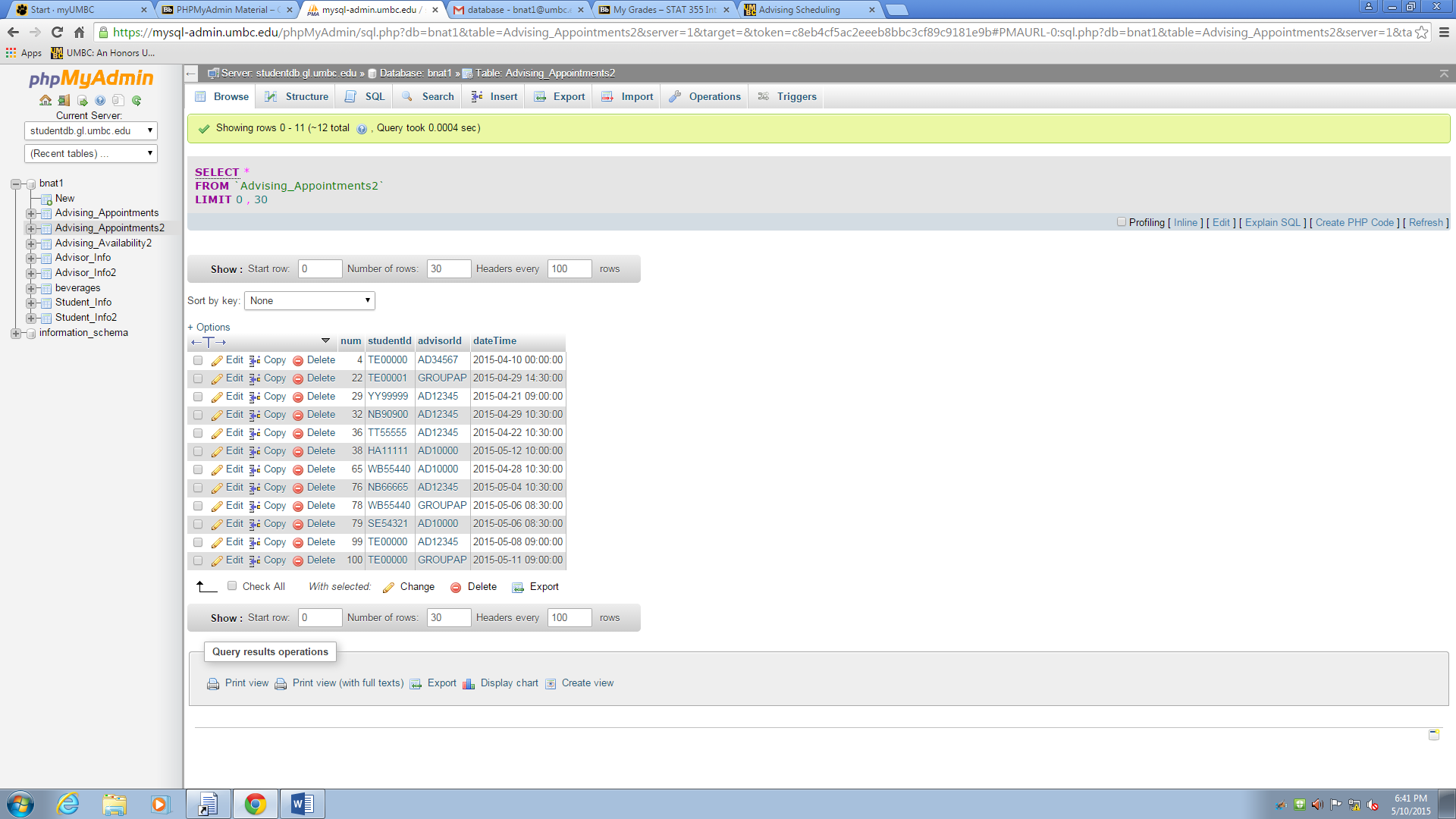
Advising\_Appointments2: Browse view after creating appointment with New Student



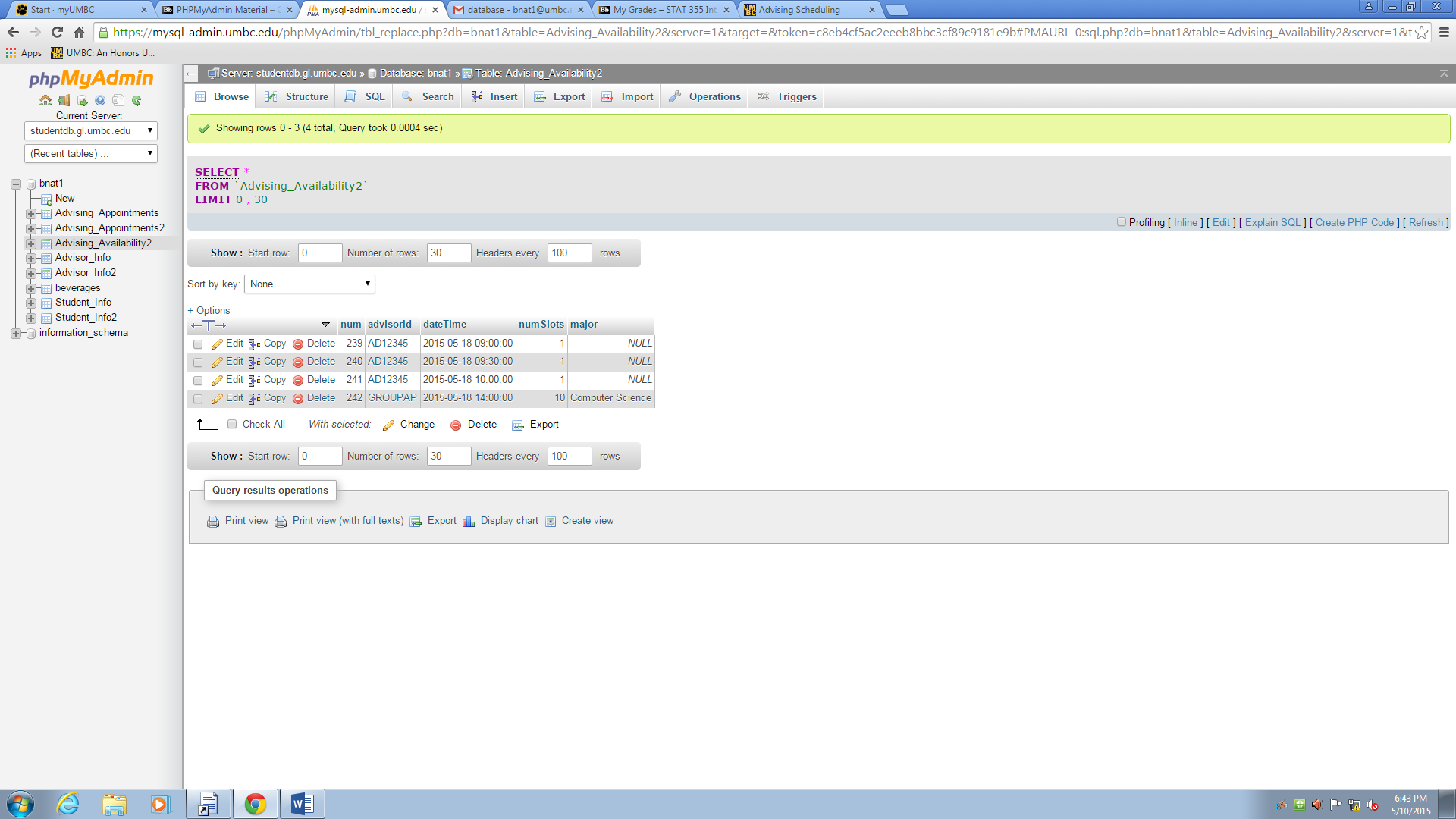
Advising\_Appointments2: Browse view after changing appointment time



Advising\_Appointments2: Browse view after cancelling appointment



Advising\_Availability2: Browse view after making a group time available



Advising\_Availability2: after canceling the group appointment time

