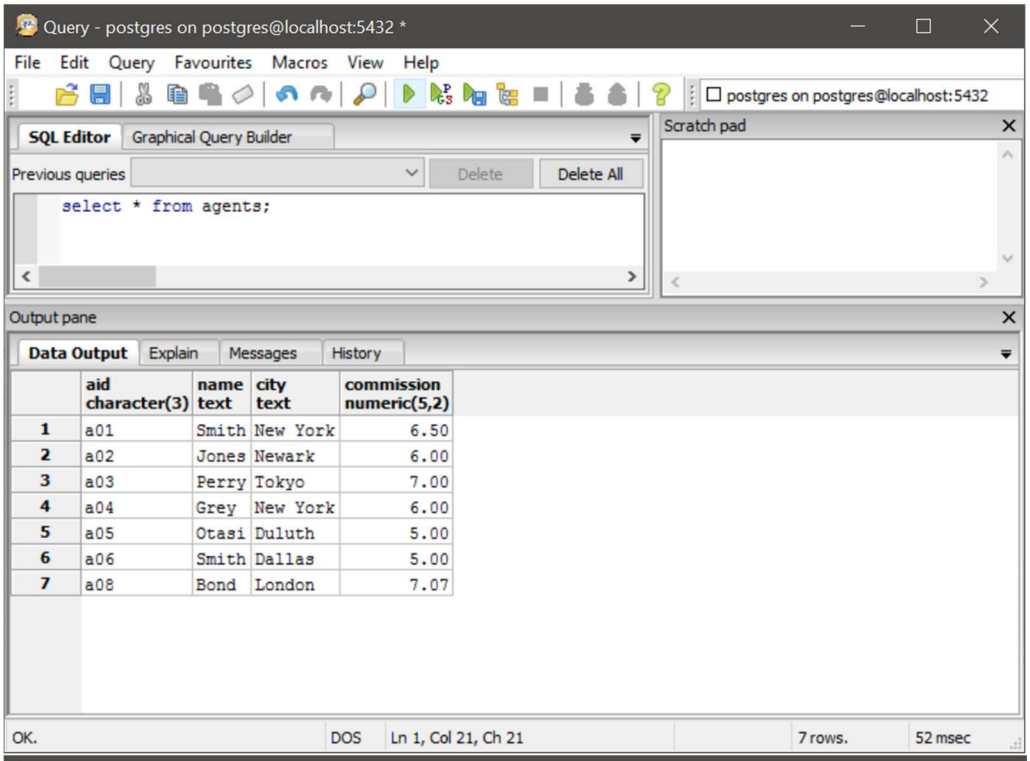
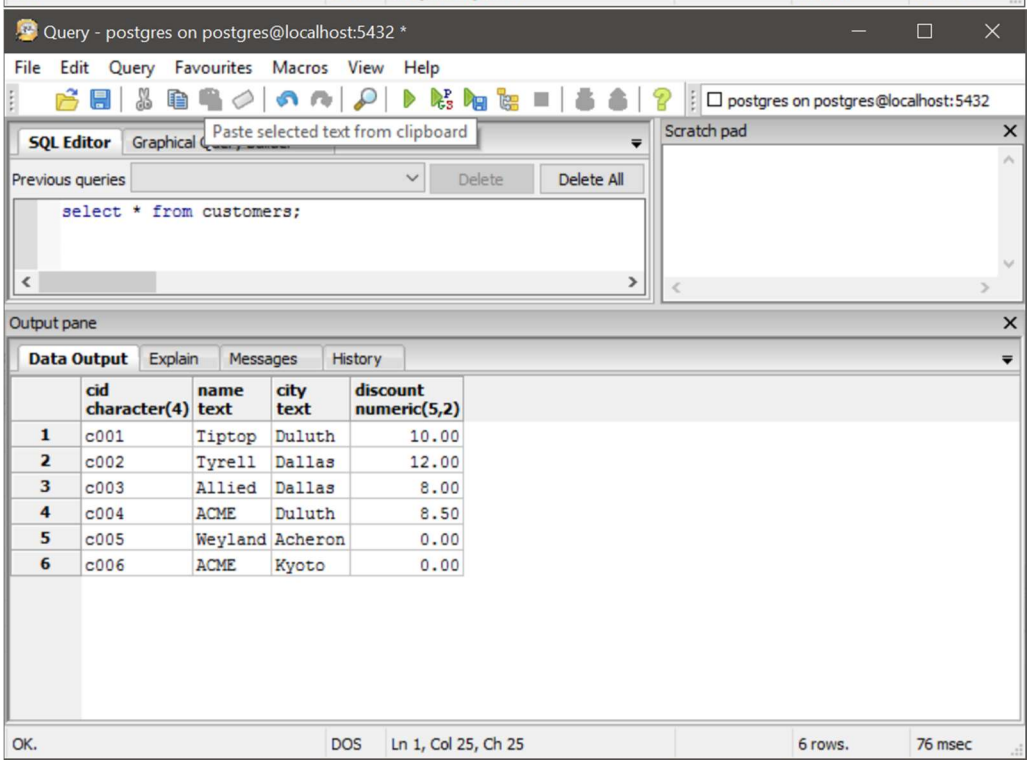
Michael Read

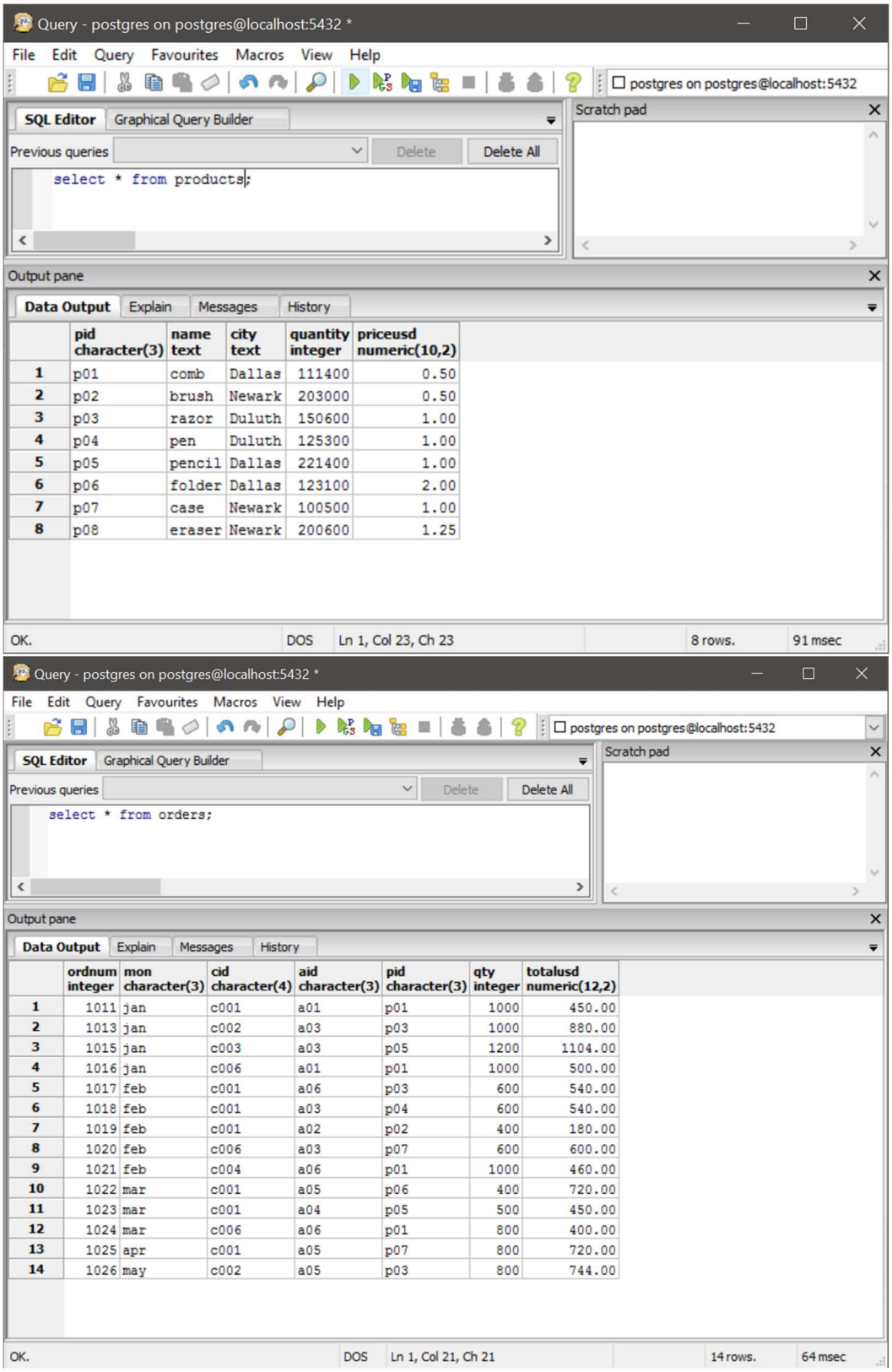
Lab 2

September 15, 2016

1) Query and results screen shots







The data in the results of the queries matched the data shown in the CAP snapshot

2) Distinctions between primary key, candidate key, and superkey

In relational databases, a superkey is a group of columns in a table which uniquely identifies any row in that table. This means that no two rows in a table will have the same data stored in the superkey, as knowing the data in the superkey tells you exactly which row in the table you are looking at. When you reduce the number of columns in the superkey to the minimum number of columns required to uniquely identify each row in a table, the superkey becomes the candidate key. If the candidate key for a table is a single column, it is called a primary key. For example, a superkey in a table holding data about airline flights might be the columns of departure_time, destination, and flight_number. However, maybe in order to uniquely identify each row, you only need flight_number. That means that you can simplify the superkey down to a candidate key of just flight_number. This candidate key is special, as it only contains one column, which means that the airline flights table has the flight_number column as its primary key.

3) Short essay on data types

In a database table holding information about patients in an Emergency Room, one might name the table ER_Patient. Some example columns in the ER_Patient table could be date_admitted, birth_date, name, weight_in_lbs, height_in_ft, house_address, occupation, place_of_work, contact_phone_number, contact_name, and contact_relationship. The fields contact_relationship, contact_name, contact_phone_number, place_of_work, occupation, house_address, and name would be of the data type text, because they are mostly composed of text, and the operations you would want to use on them would be those of type text. Both weight_in_lbs and height_in_ft would be of type double, as they would be decimal numbers, and used in numerical calculations making the type double the right choice. The column birth_date would be of type date, as it would record the date of birth, and the operations used on it would fit the type date. Finally, date_admitted would be of type timestamp, as both the date and time that the patient was admitted would be important, making timestamp the best option. The columns date_admitted, birth_date, name, weight_in_lbs, and height_in_ft would be not nullable, as those fields can be filled in for every patient admitted to the ER. The columns of house_address, occupation, place_of_work, contact_phone_number, contact_name, and contact_relationship are not all able to be filled in for every patient, as some may not have an address, some might be out of work, and others might not have anyone to contact. This means that those columns would be nullable.

4) Explain the 3 relational “rules”

1. The “first normal form” rule means that all values in the intersection of a column and a row should be atomic, or a single value. This means that no entry should contain a list of multiple values, and each column should be designed to only hold one value. An example of this is in a table storing data about students in a class, where one of the columns is called students_pet_name. While this would work fine if every student in the class only had a maximum of one pet, if there was a student in the class with multiple pets, then in this design, all of those pets names would have to be stored in the one column, which violates this rule. Making sure that all entries are atomic is important, because if they are not, then the structure of the database deteriorates, and searching for a single attribute becomes difficult, because if you find multiple values in one column, then you may not know which value is the one you want.
2. The “access rows by content only” rule means that the only way you should be looking up values in a table is by naming what column and what row you are looking in. Instead of querying for the data in row 2, column 4, you instead should query for what is in the superpower column of the Sean Connery row. The reason this is important is because databases are defined as sets of tables, which are in turn sets of columns and rows. Set theory tells us that elements in a set have no inherent order, which means that technically speaking, column 4 in the table could be superpowers, or it could be height_in_ft, both of which are valid for the table. In order to keep tables as sets of columns and rows, you must always query by content, and not by position.
3. The “all rows must be unique” rule means that no two rows in a table should have the same values for all of their columns. When you have multiple rows with the same information, you run into a duplication problem, where changing one piece of information means you have to change it in multiple places, and when you go searching for something, you might not know which row you should be looking in. For example, in a table of students, if you have two rows which both have the same information, you would have two instances of the same student in the table. If you then went to change something, but didn’t change it in both places, then when to search your table, you might not be sure which entry of that student you should use, as they would both be for the same student, but would have different information about that student.