INF1343, Winter 2012, Quiz 4

_____

write your name above

Galaxy Mansions is a real estate agency that helps wealthy clients buy and sell mansions around the Galaxy. The agency uses a database consisting of the following four tables:

seller (galactic_social_insurance_number, full_name, species, tax_status)

buyer (galactic_social_insurance_number, full_name, species, tax_status)

mansion_for_sale (mansion_id, date_listed, description, number_of_rooms, size_in_sq_m, number_of_bathrooms, planet_code, planet_name, address, price, planet_size, planet_population)

sale_transaction (mansion_id, seller_id, buyer_id, date_of_sale, amount_paid, commission_received)

a) Do any of the tables in the above database violate 1NF, 2NF, or 3NF? If so, explain which normal form is violated and why. If applicable, describe the functional dependency that creates the violation.

The most clear candidate for an offending violation is this dependency:

planet_code → planet_name, planet_size and planet_population

Whether this is a 2NF violation or a 3NF violation depends on what we assume to be the candidate keys in this table. If we assume that mansion_id uniquely identifies a mansion listing and is the only key, then we the dependency shown above is a *transitive* dependency and we have a 3NF violation. The same is true if we assume that mansion_id is not unique and other attributes, e.g., date_listed would also need to be a part of a key. If we assume, however, that the key has to include planet_code (as some of you did), then the dependency listed above is a *partial* dependency and we have a 2NF violation. I gave full credit for this answer as long as the assumption was stated clearly and there were no other shortcomings.

Notes:

- All tables satisfy 1NF. Strictly speaking, if we go by the definition used in class and in the book, a table that is implemented in SQL always satisfies 1NF. This is because a table that violates 1NF cannot be implemented in SQL! (What we usually have to watch out for are bad ways to get rid of 1NF violations, such as adding multiple columns – "child1", "child2". Note doing so *does* resolve 1NF, just not in a good way.)
- Having "a functional dependency" is not a problem per se. Some functional dependencies are good. 2NF and 3NF violations are caused by *specific kinds* of functional dependencies.
- 2NF and 3NF violations do not depend on which candidate keys you select as primary keys.
- "A key" is not the same thing as "the primary key." Also, being "a key" does not mean that an attribute would potentially make a *good* primary key.
- There are many other ways in which this database could be improved. However, not all of those imperfections are 2NF or 3NF violations. In other words, tables that fail 2NF or 3NF should be considered poorly designed, but not all poorly designed tables fail 2NF or 3NF. For example, yes, having separate tables for seller and buyer is bad design. It is not, however, a 2NF or 3NF violation. So, it has no relevance to this quiz.

b) If you identified a violation, explain what may be some of the practical consequences of this violation.

The transitive dependency shown above demonstrates that we are storing information about planets in a table

that is really above mansions. This means that we will not be able to add information about a new planet until a mansion on this planet is listed (an “insertion anomaly”). We also risk losing information about a planet when we delete a mansion (a “deletion anomaly”). And if we want to update planet population, we will need to go and change it in many different rows (an “update anomaly”).

Notes:

- Lack of normalization does not normally lead to performance problems. In fact, denormalized tables can sometimes allow for *faster* queries, since you don’t need a join. (This improvement in performance is usually not a good reason to avoid normalization.)