/\*

Lab 4 – queries

Ryan Owens

9/29/16

PostgreSQL is the best!

\*/

-- Get the cities of agents booking an order for a customer whose cid is 'c006'.

Select city

from agents

where aid in

   (select aid

   from orders

   where cid = 'c006');

-- Get the ids of products ordered through any agent who takes at least one order    from a    customer in Kyoto, sorted by pid from highest to lowest. (This is not    the same as asking for    ids of products    ordered by customers    in Kyoto.)

Select pid

from orders

where aid in

   (select aid

   from orders

   where cid in

      (select cid

       from customers

       where city = 'Kyoto'))

order by pid desc;

-- Get the ids and names of customers who did not place an order through agent a03.

Select cid, name

from customers

where cid not in

   (select cid

    from orders

    where aid in

        (select aid

        from agents

        where city = 'Tokyo'));

-- Get the ids of customers who ordered both product p01 and p07.

(select cid

from orders

where pid = 'p01')

   INTERSECT

(select cid

from orders

where pid = 'p07');

-- Get the ids of products not ordered by any customers who placed any order through agent a08 in pid order from highest to lowest

Select pid

from orders

where pid not in

   (select pid

    from orders

    where aid in

        (select aid

        from agents

        where city = 'London'))

order by pid asc;

-- Get the name, discounts, and city    for all customers who place orders through agents in Dallas or New York.

Select name, discount, city

from customers

where cid in

   (select cid

    from orders

    where aid in

        (select aid

        from agents

        where city = 'Dallas' Or city = 'New York'));

-- Get all customers who have the same discount as that  of any customers in Dallas or London

Select \*

from customers

where discount in

   (Select discount

    from customers

    where city = 'Dallas'

Or city = ‘London’);

-- Tell me about check constraints: What are they? What are they good for? What’s the advantage of putting that sort    of thing    inside    the database?    Make up some    examples of good uses of check    constraints and some    examples of bad uses    of check constraints. Explain    the differences in your examples    and argue your case.

/\*  
Check constraints are constraints of integrity in SQL, specifying a requirement that must be met by each row in a table. The constraint specified must be a predicate, and can refer to either a single, or multiple columns of a table. These constraints are good for limiting column(s) or table values to a specified range of possibilities. The advantage of utilizing this check constraint within the database is that you can eliminate the potential for values out of scope for the table, column, or row entries and create a guideline for the data that can be entered. Some good examples would be columns holding id values, a check constraint would eliminate the possibility of assigning an id of 0 by limiting the range to a minimum value of 1 and a maximum value of x. Another good example would be age, it would not make sense to allow someone to have an age of 0 or an age of 130, because we do not state the age of infants as 0 and people do not currently live long enough to reach an age of 130. A bad example would be limiting all id’s to integers starting at 1, when multiple id’s are used for individual tables and a table uses multiple id’s as foreign keys which prevents uniqueness. The difference between these examples is that the good examples limit the range of values to an acceptable range based on protocols for creating id’s or a allowing for reasonable age ranges. The bad example allows all id’s to use and start at the same range which eliminates uniqueness when you have multiple departments or other users such as clients, being assigned id’s in their own table that will be used to reference a table with multiple id columns that correlate to multiple tables but each id starts with 1 and increments per user.

\*/