**CSC263 Database Systems**

**Homework Assignment #4**

**Part I. Normalization**

1. **Answer the following questions.**

* What three data anomalies are likely to be the result of data redundancy? How can such anomalies be eliminated?

Three data anomalies that are likely to be the result of data redundancy are: insertion, deletion, or updating. An obvious major way is to normalize the table structuring.

* What is normalization?

Normalization is when we go through the tables of a database and reduce the number of redundancies that can occur.

* When is a table in 1NF?

A table is in 1NF when there are no repeating groups within the table, all attributes are dependent upon the primary key, and all of the attributes are defined.

* When is a table in 2NF?

A table is in 2NF when it no longer has any partial dependencies.

* When is a table in 3NF?

This would occur once a table no longer has any transitive dependencies.

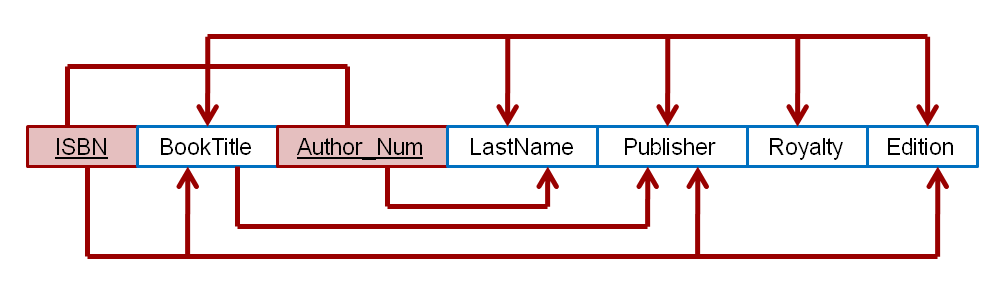
* When is a table in BCNF?

This would occur once a tables determinants are candidate keys.

* What is denormalization? Explain the pros and cons.

Denormalization is the process in which a database adds redundancies in order to increase efficiency. The advantage here is that less queries would have to be performed HOWEVER this heavily increases the access load and overhead associated with the database.

* The dependency diagram shown below indicates that authors are paid royalties for each book that they write for a publisher. The amount of the royalty can vary by author, by book, and by edition of the book.

****

* 1. Based on the dependency diagram, create a database whose tables are at least in 2NF, showing the dependency diagram for each table.
  2. Create a database whose tables are at least in 3NF, showing the dependency diagram for each table.

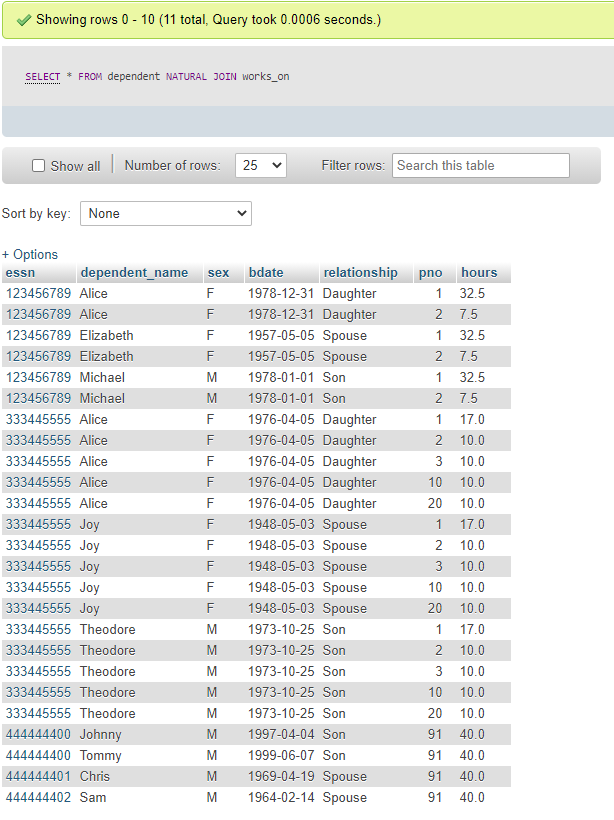
1. **Prove or verify that you have brought all your semester project’s database table structure to the 3rd normal form.**

I believe that my semester project is already in 3NF. All of my tables are each dependent upon their primary keys, all of the attributes are defined, there are no repeating groups, there are no partial dependencies among the tables, and there are no transitive dependencies. Each tables attributes completely rely on the primary key and without that key they would not function properly. Even in situations where I have tables completely consisting of foreign keys, those tables are completely reliant on the combination of those keys to accurately perform.

**Part 3. SQL operations**

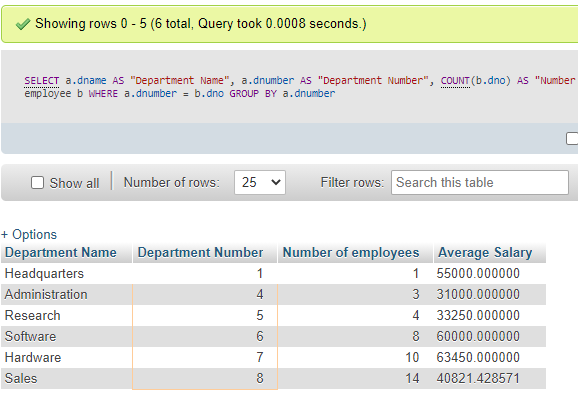
* **What are the different types of JOINs?** 
  + Natural JOIN

A natutal JOIN is one in which tables are joined but the common columns only appear once. I did this using “works\_on” and “dependent” from the COMPANY database because both tables included “essn” See below:



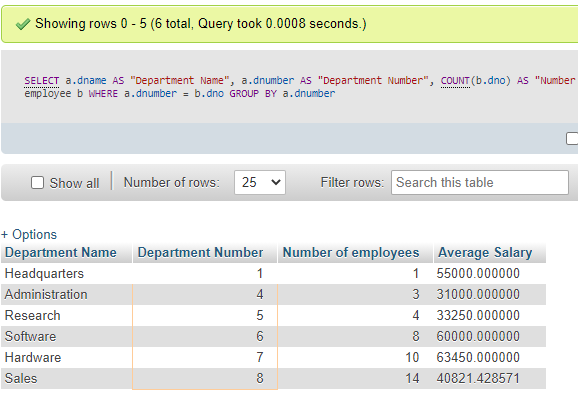
* + EquiJOIN

An EquiJOIN is when two tables are joined using eualities between them. In the example I give we use the “department” and “employee” tables from the COMPANY database to helpe show how many employees are in each department along with their salaries by equiJOINing.

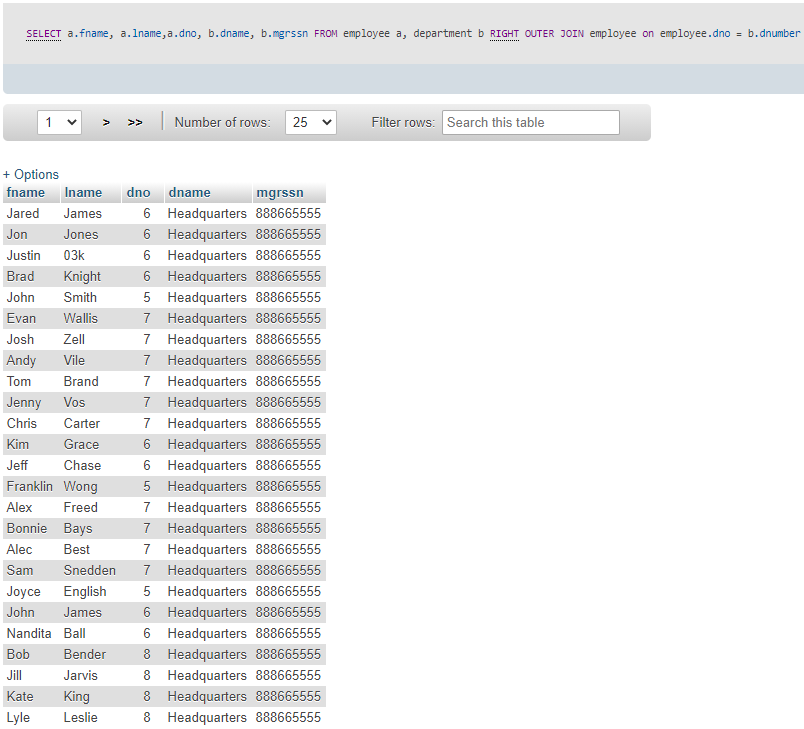


* + Inner JOIN

An Inner JOIN is done when two tables are joined using values that are the same within two tables. Again the above example I believe is an example of an Inner Join.



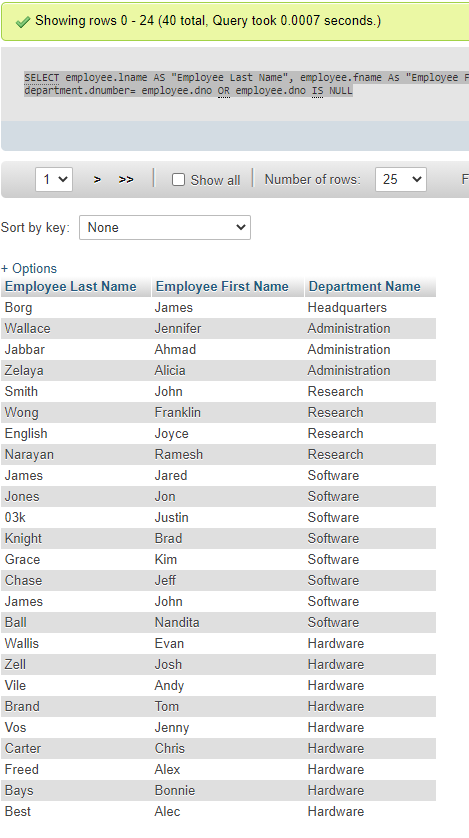
* + Outer JOIN

An Outer JOIN is a JOIN that returns both the attributes that satisfy the JOIN condition as well as those that don’t from the JOIN condition. Fror this, I used “employee” and “department” to right OUTER JOIN the two tables.

* + LEFT JOIN

A left JOIN is when a JOIN condition is set to join one table to another from the left. For this I used one of the questions from the previous lab to show how it is performed. Problem 27 from lab 6 asked us to left join the employee and department tables.

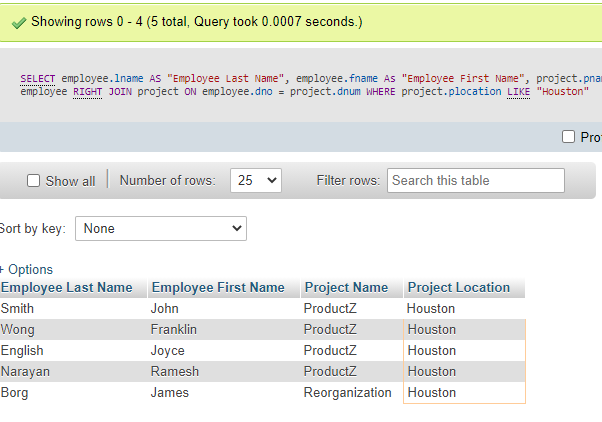
Query Used “SELECT employee.lname AS "Employee Last Name", employee.fname As "Employee First Name", department.dname AS "Department Name" FROM department LEFT JOIN employee ON department.dnumber= employee.dno OR employee.dno IS NULL”



* + RIGHT JOIN

A RIGHT JOIN does the same as a left JOIN however it places the table join on the right side of the original table. Again I used a problem from lab 6 that asked us to join the employee tabl and project tables based on a certain location.

Query Used “SELECT employee.lname AS "Employee Last Name", employee.fname As "Employee First Name", project.pname AS "Project Name", project.plocation AS "Project Location" FROM employee RIGHT JOIN project ON employee.dno = project.dnum WHERE project.plocation LIKE "Houston"”

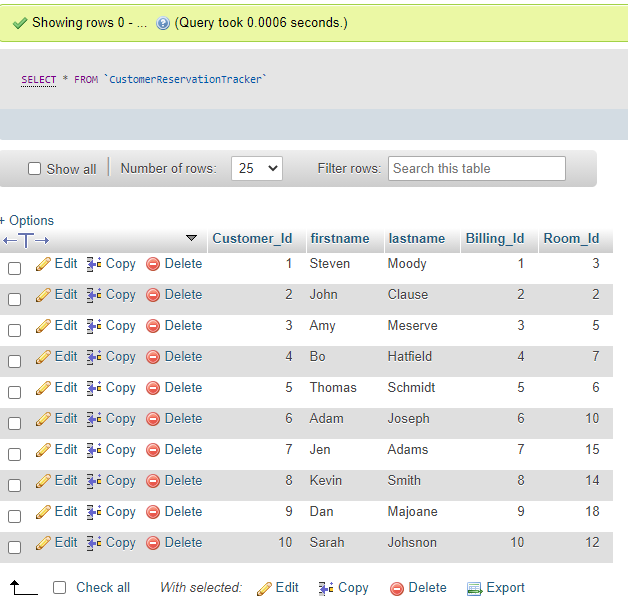


Issue real queries in MySQL to illustrate the use of EACH type of JOIN operations listed above. Queries must be issued using any database we have been working on during this semester in the lab activities as well as that you have been working on for your team projects. You need to show the queries and insert the screenshots that can show the results generated by the queries.

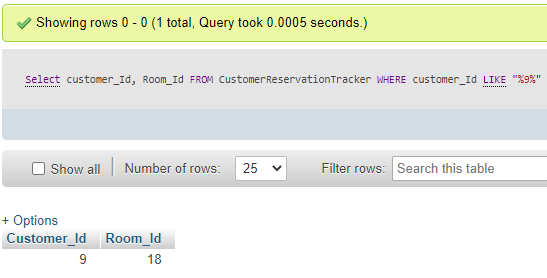
* **What is a VIEW in SQL? What is a temporary table in SQL?** Using the database you created for the semester project you are working on, show an example that uses a VIEW. You need to show the query that creates a view and the query that uses the view. Show the successful operations with enough screenshots.

A view is a special virtual table that is created within the database using result sets of stored query data that can then be queried just like any other table. A temporary table is a table that exists only temporarily within a database. I created a view that shows the customers along with the rooms they are reserving.

Query Used to create view: “CREATE VIEW CustomerReservationTracker AS SELECT a.Customer\_Id, a.firstname, a.lastname, b.Billing\_Id, b.Room\_Id FROM hotel\_Customer a, hotel\_Reserves b WHERE a.Customer\_Id = b. Customer\_Id”

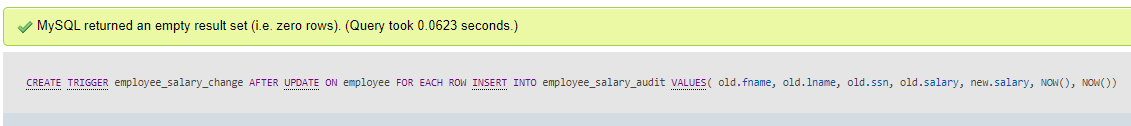


Query used to test view: Select customer\_Id, Room\_Id FROM CustomerReservationTracker WHERE customer\_Id LIKE "%9%"

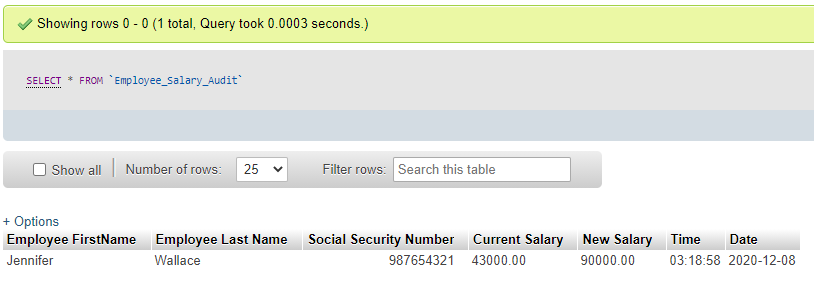


* **What is a TRIGGER in SQL? What are the pros and cons of using SQL Trigger?** Using the database you created for the semester project you are working on, show an example that uses a TRIGGER. You need to show the query that creates a trigger and the query that causes the trigger to fire. Show the successful operations with enough screenshots.

A TRIGGER in SQL is exactly what it sounds like. It is a query or update that occurs either before or after a set of conditions is met. The pros of a trigger are that we can create things like logs or verify things occur a certain way through using triggers but these triggers can be complicated, add a layer of overhead to the database and also can be difficult to view depending on their complexity. To show this I created the trigger from lab 6 using the following:



Then I performed a salary change to one of the employees.



* **What is a SQL Index? Why should we use Index?**

Using the database you created for the semester project you are working on, show one or more example that uses indice.

An index are used in order to obtain data more efficiently from within a table. These are primarily used to make searching more efficient. Here I added an Index on my table “hotel\_Customer” for the customers first names using the statement ALTER TABLE `hotel\_Customer` ADD INDEX( `firstname`);.See the screenshot below.

