**CSCI 4125/5125 Course Project**

**Data Models and Database Systems**

**Fall 2023**

**Course Project**

**Phase 6: Intermediate SQL (10/4)**

**Due: Thursday, 10/12 @ 11:59pm**

**Reading:** Silberschatz Chapters 3.5, 3.7, 4.1, 4.2

**Submission Guidelines:**

1. This assignment is worth 60 points for all students.

2. It is your responsibility to make sure all files are readable and submitted on time.

**Submission:**

- Task 1 requires you to submit a single .sql file and a single .txt file worth 36 points.

- Task 2 requires you to submit a single Java file and a screenshot worth 24 points.

**Task 1. Intermediate Retrieval Queries (36 points, 9 points each)**

Write a SQL query for each of the following problems. Unless values are specified in the problem, do not hardcode values in queries. In other words, your query should still answer the problem if the data changes. **Submit**: your answers in a single .sql file (e.g., phase6.sql) and the outputs from each of your queries in a .txt file (e.g., phase6.sql).

--1. Find the names of all customers who have at least one address

--2. For each product id, list the total quantity sold

--3. Find the names of all products that do not have any reviews

--4. Find the names of all customers who were referred by 'Margot Robbie'. Note: You must use the string ‘Margot Robbie’ and not hardcode the ID.

e ID.

**Task 2. Understanding Query Processing (24 points, 12 points each)**

One of benefits of SQL is the program-data independence supported by the relational model (or the DBMS). However, to fully understand the power of SQL, it is helpful to know how the equivalent could be implemented in the flat file model. In this task, you will write a Java method, not SQL, that is the flat file model solution for Queries #1 and #2 in Task 1. Each method, should include the following:

* Read the relevant .txt file(s) given in Phase 4. You can hardcode the file names.
* Process the input to compute the query result. When you process the input, you will now have to hardcode the column positions.
* Print the results to the console. As a sanity check, your results should be the same as in Task 1.
* None of your code will use SQL.
* *Hint*: We have used some useful methods in previous phases of the project, e.g., trim(), split(), etc.
* *Hint*: It may be useful to store results in a data structure, such as a HashMap, that you can later print after you after done all of the processing.
* *Hint*: One way to process a JOIN or a subquery is a double for-loop.

**Submit:** A single Java file containing a method that solves only Queries #1 and #2 in Task 1, i.e., a total of two methods besides main() and a screenshot of the output printed to the console.