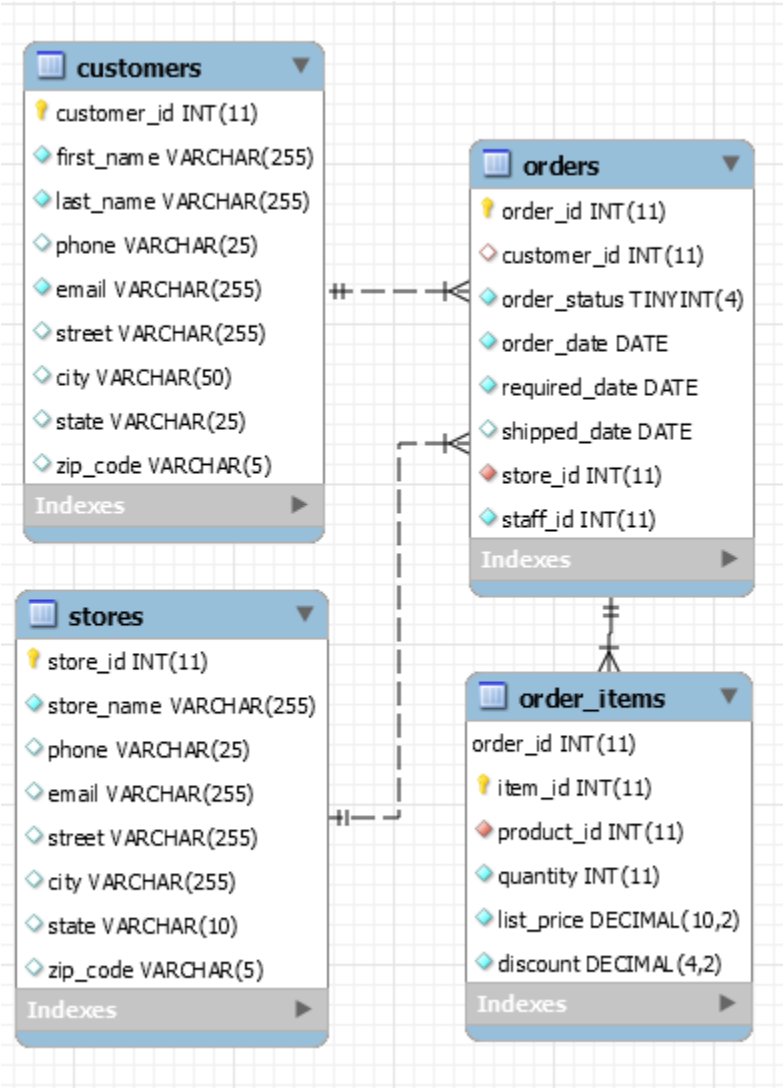
Project #03: Sales Database SQL training

Complete By: Tuesday April 14th @ noon
Assignment: Writing SQL Queries
Policy: Individual work only, late work *is* accepted (up to 24 hours late for a penalty of 10%)
Submission: via Gradescope

Overview

The project is to write a series of queries against a database containing sales data. This database contains 4 tables with data about customers, orders, stores, and items in the orders.

I've provided templated code for the files I'd like you to submit. Fill in the SQL in the individual files, and copy those queries into the Proj3ALL.sql file when you are done and ready to submit.



Requirements

You will be required to use an SQL **join** when retrieving data from more than one table. You can join tables by putting commas between them in the FROM clause, then including the condition that the primary key matches the foreign key in the WHERE clause. Alternatively, you can use other forms of joins like the ones discussed in class on 4/8, or nested queries.

Given the importance of database applications, there are numerous tools that generate SQL for you (e.g. from the database schema). Such tools cannot be used --- you are required to write your SQL queries yourself, from scratch.

Assignment

For each of the following problems, write a query which retrieves the data containing the answer in Prog3_##.sql, where ## is the number of the question, such as 01. The basic description of the query can be found here, and the detailed description of the fields and sample output can be found in the template code.

1. Generate a table containing nicely formatted customer information.

2. Find the contact information for a particular customer. The order will be provided in @oid, be sure that your query does not include the setting of this variable.

3. Filter out only the active orders. It is useful to have a table containing only the active orders so that the company can see what needs to be done.

4. Investigate how long it takes in general (average number of days) for an order to get completed (time from order to shipping).

5. Create a table similar to the one in question 3, limiting the results to only the orders exceeding the average time computed in question 4 and ordered by the time since ordering. The current day will be provided in the variable @today, to produce consistent results for what the output should be.
   Do not hard code the value for average time with the value in the data set.

6. For each store, report the Average turnaround (time from order to shipment).

7. For each store, list the percentage of orders which did not ship until after the required_date.

8. For each store and customer pair where more than one order has been made, list the number of orders that customer has made at that store.

9. List the receipt information about each order containing the total list price and discount applied.

10. Identify whether there may be some correlation between the amount ordered and the time it takes to ship the order. Include the average and std deviation of the ratio of these factors.

Electronic Submission and Grading

The grade reported by Gradescope will be a tentative one. After the due date, submissions will be re-evaluated against a new database to see that the correct answers are still being computed on a new set of data.

When you are ready to submit your program for grading, login to Gradescope and upload all your "Prog3_##.sql" source files, in addition to Prog3ALL.sql, either as a zip or individually. You have unlimited submissions, and Gradescope keeps a complete history of all submissions you have made. By default, Gradescope records the score of your last submission, but if that score is lower, you can click on "Submission history", select an earlier score, and click "Activate" to select it. The activated submission will be the score

that gets recorded, and the submission we grade. If you submit on-time and late, we'll grade the last submission (the late one) unless you activate an earlier submission.

Policy

Late work *is* accepted. You may submit as late as 24 hours after the deadline for a penalty of 10%. After 24 hours, no submissions will be accepted.

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them (e.g. your "iClicker teammates"), this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml .