CS 328 - Homework 1

Deadline

11:59 pm on **Friday, January 30, 2026**

Purpose

To make sure you have read over the course syllabus, to let me know whether you wish to build upon your CS 325 database this semester, and to start getting familiar with the "bookstore" database you also will be working with this semester.

How to submit

Problem 1 is completed **on the course Canvas site**.

For **Problem 2 onward**:

Each time you wish to submit, submit your files **using `~st10/328submit` on nrs-projects**, with a homework number of **1** .

Reminder: Instructions for using the tool ~st10/328submit

- If **not** already on nrs-projects, transfer your files to be submitted to a directory on nrs-projects.
  - You can do so by using `sftp` (secure file transfer), or an application supporting `sftp` such as WinSCP or FileZilla, and connecting to `nrs-projects-ssh.humboldt.edu`, and then transferring them.
  - (See the class handout: "**Useful details: `ssh` and `sftp`**" linked along with this homework handout.)
- Use `ssh` to connect to `nrs-projects-ssh.humboldt.edu` .
- For each nrs-projects directory containing files to submit, use `cd` to change to that directory -- for example:

  ```
  cd 328hw01
  ```

- For each nrs-projects directory containing files to submit, once you have `cd`'d to that directory, submit them by typing the command:

  ```
  ~st10/328submit
  ```

  ...then giving the number of the homework for which the files are being submitted when asked,

  ...and then answering that, `y`, you *do* want to submit all of the files with of-interest-to-328 suffixes in the current directory.

  - (Remember, I don't mind if extra files get submitted as well -- I'd rather receive too many files that too few, and typing in all of the file names for each assignment is just too error-prone!)

- you are expected to **carefully** check the list of files that the `~st10/328submit` tool believes have been submitted, and make sure all of the files you hoped to submit were indeed submitted!
  - (The most common error is to try to run `~st10/328submit` while in a different directory than where your files-to-be-submitted are.)

Problem 1 - 20 points

Problem 1 is correctly answering the "HW 1 - Problem 1 - required syllabus confirmation and reading questions" on the course Canvas site.

Problem 2

You will be building pieces atop a small bookstore database, described in Problem 3 below, during the semester.

You will **also** be building pieces atop a second database that you choose -- it could be your database from CS 325, or it could be another database with at least 7 related relations.

Because we don't have time to go through a proper full model/design/implement cycle for a new database in CS 328, I would like to know if -- and/or how many -- students might need or want a different database than the one they built in CS 325, so I can try to arrange for several alternatives for such students to choose from.

For this problem, you **do not** have to **commit** to a decision yet -- you **do** need to do the following:

- On nrs-projects, using `vim/emacs/nano`, create a file named **`second-db.txt`** that starts with your name.
- Then include a statement letting me know if you know yet if you wish to build atop your CS 325 database in CS 328 this semester.
  - (That is, you might *know* you want to,
  - or you might be *still deciding*,
  - or you might know you *cannot* or *do not wish* to,
  - or you might have a scenario near-and-dear to your heart and be committed to designing another database with at least 7 relations for this purpose for CS 328, for example.)

Submit your resulting file **`second-db.txt`**, containing your name and this statement.

Problem 3

We will be practicing many of the concepts and languages we will be using this semester on a little bookstore database, whose tables can be found in the scripts **`create-bks.sql`** and **`pop-bks.sql`** posted along with this homework handout. Since you will be spending a lot of time using this database, it will be useful for you to start getting familiar with it now.

CS 328's definition of relation structure form

Since terminology can vary: for CS 328, we'll define **relation structure form** as follows: when you

write a relation in this format:

- the name of the relation,
- followed by an opening parenthesis,
- then a comma-separated list of the relations's attributes, (ALL of its attributes, including foreign-key attributes),

  with the primary key attributes written in all-uppercase, and the other attributes written in all-lowercase,
- followed by a closing parenthesis,
- followed by, on the next lines, SQL-syntax foreign key clauses for each of the foreign key attributes.
- Put a blank line before starting the relation structure form for the next relation.
- CS 328 class style: it is OK to write these relation structures in any order.
  - (That is, even though, in SQL, you must create a table before a foreign key in another `create table` statement references it, we are not enforcing that ordering for relation structure form.)

This format is relatively compact, reasonable to type in plain text. and serves as a convenient reference when designing SQL queries.

For example, the relations created by the SQL script `set-up-ex-tbls.sql` (that you used in the Week 1 Lab Exercise) would be written in relation-structure form as:

```
empl(EMPL_NUM, empl_last_name, job_title, mgr, hiredate, salary,
     commission, dept_num)
    foreign key (dept_num) references dept,
    foreign key (mgr) references empl(empl_num)

dept(DEPT_NUM, dept_name, dept_loc)

customer(CUST_ID, cust_lname, cust_fname, empl_rep, cust_street,
         cust_city, cust_state, cust_zip, cust_balance)
    foreign key (empl_rep) references empl(empl_num)
```

Your task for Problem 3

As one means of getting familiar with this little bookstore database created by the script **`create-bks.sql`** and populated using the script **`pop-bks.sql`**:

- On nrs-projects, using `vim/emacs/nano`, create a file named **`design-bks.txt`** that starts with your name.
- Then, for each of the relations in the database created by the script **`create-bks.sql`**, write its **relation structure form** in `design-bks.txt`.

Submit your resulting file **`design-bks.txt`**, containing your name and the bookstore's database's relations written in relation structure form.

Problem 4

On nrs-projects, create your own copies of the bookstore database scripts **`create-bks.sql`** and **`pop-bks.sql`** using these commands:

```
cp ~st10/create-bks.sql .     # NOTE the space and the period!!
cp ~st10/pop-bks.sql .
```

...and run these in **`sqlplus`** to create and populate these tables.

Then, using `vim/emacs/nano`, write a script **`bks-play.sql`** that meets the following specifications:

- start it with comment(s) containing **`CS 328 - HW 1 - Problem 4`**, your name, and the last-modified date
- start spooling to a file **`bks-play-out.txt`** (and make sure you **`spool off`** at the script's end!)
- write a **`prompt`** command including your name
- CHOOSE **at least THREE** of the following, and

  precede each of your choices with a `prompt` command giving the problem part being answered, then your answer for that part.

Since this problem is intended both to help you get familiar with the bookstore database and review writing SQL queries, **I have included an example** `bks-play-out.pdf` **along with this homework handout, for comparison purposes.**

- This is both to let you know if you are on the right track, AND to encourage DEBUGGING of your SQL `select` statements if you see significant differences.

4 part a

Write a query projecting **just** two columns:

- one column displaying the names of publishers, and
- one column displaying the city of the publisher, a comma and a blank, and then the state of the publisher, with a tasteful column heading of your choice

...displaying the results in alphabetical order of publisher name.

4 part b

Write a query projecting **just** the title names and title prices for titles published by Benjamin/Cummings, displaying the results in alphabetical order of title name.

4 part c

Write a query projecting each publisher's name, the number of titles available from that publisher, and the average title price of a title from that publisher, displaying the results in order of average title price of a title from that publisher, and giving the number of titles available the displayed column heading **`# TITLES`** and the average title price of a title from that publisher the displayed column heading **`AVG PRICE`** .

4 part d

Write a query projecting the names of titles involved in order 11012, and the quantity of each title being ordered.

4 part e

Write a query projecting the name of the publisher(s) involved in the order line item(s) with the greatest order quantity, and project that greatest order quantity as well.

Make sure you turned spooling off in **`bks-play.sql`**, and submit your resulting files **`bks-play.sql`** and **`bks-play-out.txt`**.