CS348 - Project 3

Overview

In this project you are going to use PL/SQL (Oracle's procedural extension to SQL) to write a few functions and procedures to process data. You will need to use your Oracle account as you did in Project 2 for storing data.

Step 1: Setup

The schema for the database is the same as that you used in Project 2, i.e.:

*1. STUDENT(*snum: integer, sname: string, deptid: integer, slevel: string, age: integer)*
*2. CLASS(*cname: string, meets_at: date, room: string, fid: integer)*
*3. ENROLLED(*snum:integer, *cname: string)*
*4. FACULTY(*fid: integer, fname: string, deptid: integer)*
*5.DEPARTMENT (*deptid: integer, dname: string, location:string)*

If you have the tables in your database from the previous project, drop all five tables using the command "Drop Table X", where X is the name of the table. Keep in mind you will need to drop the tables in the correct order to avoid referential integrity errors (e.g. you will need to drop the table "Enrolled" before the table "Student", etc.).
Create the tables with all the key and referential integrity constraints necessary to model the application. Make sure that your field & table names correspond to those listed above. Then populate your database by executing the file data.sql provided with the project.

Step 2: PL/SQL

Create a file named procedures.sql. The first line of this file should be:

set serveroutput on size 32000

Your file should contain code to create and run five procedures: pro_department_report, pro_student_stats, pro_faculty_stats, pro_histogram, pro_enroll. The description of each procedure is provided below.

Your file should look something like this:

```
/* create the procedure */
create or replace procedure pro_department_report as
/* declarations */
begin
   /* code */
end;

/

/* actually run the procedure */
```

```
begin
   pro_department_report;
end;

/

create or replace procedure pro_faculty_stats as

begin
/*code*/
end;

/

begin
   pro_faculty_stats
end;

/
...
```

Procedures:

1. **pro_department_report:** Generate a report that lists, for each department, the students in that department. For each department, you should first print the department name on a line followed by the number of students in that department on the next line and a numbered list of student names in that department. The output should be modeled as follows: Sort by the department name (ascending and sort by student name (ascending) for each department.

Sample output:

Department: Computer Sciences
Total number of students: 3
---
1. Alice
2. Bob
3. Joe

Department: ECE
Total number of students: 2
---
1. Joyce
2. Sam

Department: Management
Total number of students: 0
----

2. **pro_student_stats:** Generate a report that contains statistics about students. Print out the number of classes that each student is taking; omit students taking no classes. Sort by student name.

Sample output:

Student Name # Classes
--------------- ----------
Bob 3
Joe 2
...

3. **pro_faculty_stats:** Generate a report about the total number of students each faculty teaches. Sort results by faculty name. The number of students for each faculty should be marked with an *X* under the corresponding bin.
   You should create 4 equal-sized bins for the number of students based on the minimum and maximum number of students any faculty teaches. For example, if the minimum number of students any faculty teaches is 0 and the maximum is 8, the bins created should be those in the sample output below. If the difference between the minimum and maximum number of students is not divisible by 4, your output should include an extra bin for the remainder of the numbers. For example, if the minimum is 0 and maximum is 7, the categories would be {**0}, {>0, <=1}, {>1, <=2}, {>2, <=3}, {>3, <=4}, {>4}**. If the minimum is 2 and maximum is 12, the categories would be **{2}, {>2, <=4}, {>4, <=6}, {>6, <=8}, {>8, <=10}, {>10}**. This means your output will have either 5 or 6 categories for the number of students. You can assume that the difference between the minimum and maximum number of students will always be greater than or equal to 4.
   Note that the number of students for each faculty should be calculated as the total number of students (NOT DISTINCT) in the classes that faculty is teaching. The maximum number of characters in a faculty name will not exceed 12 and the total number of students for each faculty will not have more than two digits, so you can format your output accordingly. Make sure the *X*'s in your output align with the bins of students corresponding to each faculty.

Sample output:

Faculty name	*# Students:*	*0*	*>0, <= 2*	*>2, <=4*	*>4, =<6*	*>6, <=8*
Alice		*X*				
Bob						*X*
Joe				*X*		

4. **pro_histogram:** Generate a histogram for the ages of students. Include all discrete integer values in the range min(age) .. max(age). Mark the statistical median on the graph (in the sample output below, it's 19). [Aside: Do you really know the definition of median? What if the size of your input set is even? Be careful on this one.] Only mark the median if it's an integer value.

Sample output:

Age | number of students
17 | 1
18 | 10
19 | 5 <-- median
20 | 2
21 | 8
22 | 3
23 | 0
24 | 1

5. **pro_enroll:** Write a procedure to enroll a student in a class. The input parameters will be as follows: sname_in, cname_in. You can assume that student names are unique (i.e. there is a single *snum* for every *sname* in the Student table). You can also assume that the given sname_in and cname_in already exist in the database. The result will be a new enrollment record added to the database.
   Execute this procedure to insert the enrollments (M.Lee, CS448) and (A.Smith, ENG320) to the database. Do a select * from Enrolled before and after this procedure is run (i.e. include a query that retrieves the whole content of the Enrolled table before and after the statements for executing this procedure to insert data).

Step 3: Run your procedures

You can run your procedures file by typing '@procedures.sql' after executing the *sqlplus* command in a terminal. If there are errors in creating the procedures, you can see the errors by typing the command "*show errors;".* Note that the sample outputs provided above are not the sample outputs for data.sql, they are just to give you an idea about the formatting.

Evaluation:

Your project will mostly be evaluated based on the correctness of your output for the procedures. Make sure you comply with the database schema provided above when creating the tables. We will test your procedures on a database populated with the correct tables and the table field names in your procedures have to match those given in the assignment to produce correct output.

Submission Instructions:

Please create a README file that contains identifying information. For example:

CS348 - Project 3

Author: John Doe
Login: jdoe
Email: jdoe@cs.purdue.edu

Include here anything you might want us to know when grading your project.
To turn in your project, ssh to *lore.cs.purdue.edu*, create a folder named *project3* in your home directory and copy your *procedures.sql* file and your README to that folder.

After copying your files in the folder project3, execute the following command in your home directory:

turnin -c cs348 -p proj3 project3

To verify the contents of your submission, execute the following command right after submission:

turnin -c cs348 -p proj3 –v

Resources:

You can find additional information about creating stored procedures/functions in PL/SQL on the following websites.

1. http://infolab.stanford.edu/~ullman/fcdb/oracle/or-plsql.html
2. http://www.unix.org.ua/orelly/oracle/prog2/index.htm