Washington State University

School of Electrical Engineering and Computer Science

CptS 451 – Introduction to Database Systems

Summer 2020

Dr. Sakire Arslan Ay

Homework-4

Due Date: Monday, June 8th 11:59pm

Name: ______________________________

Student Number:______________________

Question:	Max points:	Score:
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
9	10	
10	10	
ExtraCredit	5	
Total	**100**	

Consider the following database schema (also in **HW4schema.sql**):

```
CREATE TABLE UserTable (
    userID CHAR(8),
    firstName VARCHAR,
    lastName VARCHAR,
    email VARCHAR(50),
    PRIMARY KEY (userID)
);

CREATE TABLE Instructor(
    instructorID CHAR(8),
    title VARCHAR,
    PRIMARY KEY (instructorID),
    FOREIGN KEY (instructorID) REFERENCES UserTable(userID)
);

CREATE TABLE Student(
    studentID CHAR(8),
    major VARCHAR,
    PRIMARY KEY (studentID),
    FOREIGN KEY (studentID) REFERENCES UserTable(userID)
);

CREATE TABLE Course (
    major VARCHAR,
    courseNum CHAR(3),
    title VARCHAR,
    PRIMARY KEY (major,courseNum)
);

CREATE TABLE Prerequisite (
    major VARCHAR,
    courseNum CHAR(3),
    prereqMajor VARCHAR,
    prereqCourseNum CHAR(3),
    PRIMARY KEY (major,courseNum,prereqMajor,prereqCourseNum),
    FOREIGN KEY (major,courseNum) REFERENCES Course (major,courseNum),
    FOREIGN KEY (prereqMajor,prereqCourseNum) REFERENCES Course (major,courseNum)
);

CREATE TABLE Class (
    classID VARCHAR,
    major VARCHAR NOT NULL,
    courseNum CHAR(3) NOT NULL,
    semester VARCHAR(10),
    year CHAR(4),
    instructorID CHAR(8) NOT NULL,
    enrollmentlimit INTEGER,
    PRIMARY KEY (classID),
    FOREIGN KEY (major,courseNum) REFERENCES Course(major,courseNum),
    FOREIGN KEY (instructorID) REFERENCES Instructor(instructorID)
);

CREATE TABLE Enroll (
    studentID CHAR(8),
    classID VARCHAR,
    grade INTEGER,
    PRIMARY KEY (classID,studentID),
    FOREIGN KEY (classID) REFERENCES Class(classID),
    FOREIGN KEY (studentID) REFERENCES Student(studentID)
);
```

```
CREATE TABLE Assignment (
    classID VARCHAR,
    assignmentNo INTEGER,
    title VARCHAR,
    weight INTEGER,
    deadline DATE,
    PRIMARY KEY (classID, assignmentNo),
    FOREIGN KEY (classID) REFERENCES Class(classID)
);
CREATE TABLE Submit (
    studentID CHAR(8),
    classID VARCHAR,
    assignmentNo INTEGER,
    score INTEGER,
    submissionDate DATE,
    PRIMARY KEY (studentID, classID, assignmentNo),
    FOREIGN KEY (classID, assignmentNo) REFERENCES Assignment(classID, assignmentNo),
    FOREIGN KEY (studentID) REFERENCES Student(studentID)
);
CREATE TABLE Post (
    postID INTEGER,
    userID CHAR(8) NOT NULL,
    kind VARCHAR,
    timestamp DATE,
    content VARCHAR,
    popularity INTEGER,  --
    PRIMARY KEY (postID),
    FOREIGN KEY (userID) REFERENCES UserTable(userID)
);
CREATE TABLE PostAbout (
    postID INTEGER,
    classID VARCHAR,
    assignmentNo INTEGER,
    PRIMARY KEY (postID,classID,assignmentNo),
    FOREIGN KEY (postID) REFERENCES Post(postID),
    FOREIGN KEY (classID, assignmentNo) REFERENCES Assignment(classID, assignmentNo)
);
```

Please complete the following before you answer the homework questions:

1. On PostgreSQL Server create a database named `hw4`.
2. Download and extract HW4-DB.zip – attached to HW4 on Blackboard. There are *two* sql script files in this archive: `HW4DB.sql` and `HW4schema.sql`.
3. Create the tables `UserTable, Instructor, Student, Course, Prerequisite, Class, Enroll, Assignment, Submit, Post, PostAbout` by running the CREATE TABLE statements in `HW4schema.sql` file. Make sure to create the tables in the given order, otherwise you will get errors due to foreign key definitions.
4. Populate your DB by running the `HW4schema.sql.` This file contains the INSERT statements for the above tables.
    - See Appendix-1 for instructions on how to execute script files on command line.
    - You may alternatively copy and paste the INSERT statements on the DBMS client (pgAdmin or command line) and execute them.

5. Check if the data is inserted correctly by running a “select * from” on each table. The tables sizes are given below:
    - `UserTable: 22 ; Instructor: 10 ; Student: 16 ; Course: 17; Prerequisite: 11;  Class: 21,  Enroll: 40; Assignment: 17; Submit: 67 ; Post: 20; PostAbout: 20`

Write the following queries in SQL for the above database schema:

1. Find the distinct instructors who taught a 'CptS' course in the 'Spring' '2020' semester. Return the firstname, lastname, and title of those instructors.

```
 firstname |  lastname  |        title
-----------+------------+---------------------
 Diane     | Cook       | Professor
 Sakire    | ArslanAy   | Associate Professor
 Venera    | Arnaoudova | Assistant Professor
(3 rows)
```

2. Find the classes that have more than 10 students enrolled in them. Return the classID, major, courseNum of each class and the number of students enrolled.

```
    classid     | major | coursenum | numstudents
----------------+-------+-----------+-------------
 2019S01CptS451 | CptS  | 451       |          14
 2019F02STAT360 | STAT  | 360       |          12
 2019S01CptS437 | CptS  | 437       |          14
(3 rows)
```

3. Find the courses which do not have any prerequisites. Return the major, courseNum, and title of each such course.

```
 major | coursenum |               title
-------+-----------+-----------------------------------
 CptS  | 132       | Data Structures - Java
 CptS  | 223       | Advanced Data Structures
 CptS  | 581       | Software Maintenance
 CptS  | 582       | Software Testing
 MATH  | 171       | Calculus I
 MATH  | 415       | Some MATH Class
 STAT  | 360       | Probability and Statistics
 STAT  | 412       | Statistical Methods in Research I
(8 rows)
```

4. Find the students who posted more than one post on the same day.
Return the userID, the date for the posts, and the number of posts that the user posted on that date.

```
  userid  | timestamp  | count
----------+------------+-------
 8        | 2019-01-15 |     2
(1 row)
```

5. Find the course which has the highest enrollment limit among all classes.
   Return the classID, major, courseNum, semester, year, and enrollment limit of the course.

```
    classid     | major | coursenum | semester | year | enrollmentlimit
----------------+-------+-----------+----------+------+-----------------
 2019S01CptS321 | CptS  | 321       | Spring   | 2019 |              30
(1 row)
```

6. Find the courses that has the highest enrollment limit among the courses of each major.
   Return the classID, major, courseNum, semester, year, and enrollment limit of those courses.

```
    classid     | major | coursenum | semester | year | enrollmentlimit
----------------+-------+-----------+----------+------+-----------------
 2019S01CptS321 | CptS  | 321       | Spring   | 2019 |              30
 2019F02STAT360 | STAT  | 360       | Fall     | 2019 |              15
 2019S01MATH172 | MATH  | 172       | Spring   | 2020 |              20
(3 rows)
```

7. Find the students whose overall GPAs are less than their GPAs in the CptS classes (i.e., the average of the grades in the CptS classes they took). If a student didn't take any CptS courses, the query result should not include those students. Return the firstname, lastname, studentID, CptS GPA and, overall GPA of those courses.

```
firstname | lastname | studentid | csgpa | gpa
-----------+----------+-----------+-------+------
 Travis    | Person   | 16        |  2.50 | 2.25
 Ben       | Hill     | 20        |  3.67 | 3.00
 Tyler     | Walker   | 5         |  4.00 | 3.40
 Lucas     | Mason    | 8         |  3.67 | 3.40
 Noel      | Sam      | 9         |  3.33 | 3.00
(5 rows)
```

8. Find the pairs of students who submitted a 'CptS' '451' assignment on the same date and earned the same score for that assignment. Give the first names of those students, the classIDs of the assignments, the submission dates , and the scores earned.

```
firstname | firstname |  classid  | assignmentno | submissiondate |
score
----------+-----------+-----------+--------------+--------------+-------
 Andy      | Tyler    | 2019S01CptS451 |       1 | 2019-02-05     |    75
 Andy      | Bob      | 2019S01CptS451 |       3 | 2019-02-25     |    75
 Andy      | Rachel   | 2019S01CptS451 |       4 | 2019-03-05     |    75
 Andy      | Rachel   | 2019S01CptS451 |       5 | 2019-03-15     |    75
 Andy      | Noel     | 2019S01CptS451 |       5 | 2019-03-15     |    75
 Andy      | Bob      | 2019S01CptS451 |       6 | 2019-04-05     |    75
 Tazin     | Noel     | 2019S01CptS451 |       1 | 2019-02-05     |    85
 Tazin     | Tyler    | 2019S01CptS451 |       2 | 2019-02-15     |    85
 Rea       | Tyler    | 2019S01CptS451 |       3 | 2019-02-25     |    95
 Rea       | Noel     | 2019S01CptS451 |       4 | 2019-03-05     |    95
 Rea       | Noel     | 2019S01CptS451 |       6 | 2019-04-05     |    95
 Rachel    | Bob      | 2019S01CptS451 |       2 | 2019-02-15     |    75
 Rachel    | Noel     | 2019S01CptS451 |       5 | 2019-03-15     |    75
 Rachel    | Tazin    | 2019S01CptS451 |       6 | 2019-04-05     |    85
 Tyler     | Noel     | 2019F01CptS355 |       5 | 2019-11-15     |   100
(15 rows)
```

9. Find the classes whose enrollments exceed their enrollment limits (i.e., the total enrollment is greater than the enrollment limit of the class).
   Return the class major, courseNumber, enrollment limit, and the actual enrollment for those courses.

```
major | coursenum | enrollmentlimit | numstudents
-------+-----------+-----------------+-------------
 CptS  | 451       |               8 |          14
(1 row)
```

10. Students who wrote a post about an assignment but didn't submit that assignment. Return the studentID, classID, and assignmentNo for the assignment, and the content of the post.

```
studentid | classid  | assignmentno |           content
---------+------------+-------------+---------------------------------------------
 15   | 2019S01CptS451 |   2 | Clarification on problem-2
 16   | 2019F01CptS355 |   2 | Is it possible to submit HW1 a little bit late?
 18   | 2019F01CptS355 |   2 | Should we include the query output in our solution.
 4    | 2019F01CptS355 |   2 | Yes.!
 20   | 2019F01CptS355 |   5 | Can we handwrite the solution?
(5 rows)
```

Extra Credit: (5pts)

Find the students who took <u>all the courses</u> required by his/her track in his/her major.

```
studentid |    classid
-----------+----------------
 8         | 2019F01CptS355
 5         | 2019S01CptS451
 9         | 2019S01CptS451
 10        | 2019S01CptS451
 5         | 2019F01CptS355
 8         | 2019S01CptS451
 4         | 2019S01CptS451
(7 rows)
```

<u>Submission Instructions:</u>

HW4 will be submitted online on Blackboard.

- Please include all your SQL queries in a text file (in order) and save it as `HW4.sql`
- Please include the question numbers as comments. Also, include your name and the list of the students you collaborated with in the beginning of the file.
- Before you submit, make sure that the complete file can be run on the command line with the following command. Make sure to include ; at the end of each query.
  `psql -U postgres –d hw4  <  HW4.sql`
- Submit your `HW4.sql` file to “Homework-4 -Dropbox” under Homeworks on Blackboard.

Appendix

Running an SQL Script file on PostgreSQL Command Line:

Start a console window (on Windows: run cmd), and browse to the directory where the .sql script file is located.

Run the following in command line :

```
psql -U postgres –d hw4  <  your_script_file.sql
```

If the database hw4 doesn’t exists, you need to create it first.
If you would be running PostgreSQL client with another username (other than postgres), replace postgres with that username. You will be asked to enter your password for the username you specify.

How to run the script and create the output file

Let’s assume you have created your sql script file and saved it as `your_script_file.sql`.
If you would like to save query results into a file, then run a statement like the following in the command line. This would save the query results into `your_output_file.txt`:

```
psql -U postgres –d db_name < your_script_file.sql > your_output_file.txt
```