Brandon Buchanan

CSCI 2020 Database Fundamentals
Lab8

Submit as a text document called *LastnameFirstnameLab8.sql* to the drop box.
Print out a hard copy, staple this sheet to the front, and submit to your instructor before leaving the lab.

Notes:

- **Be sure all style guidelines are complied with**. (SQL style guidelines are available in the Additional Material section of Content in D2L).
- Your file should include a **header block of comments** as shown in the SQL Style Guidelines.
- **Be sure to add a comment above the solution to each task**. The comment should include the task number and the task itself, and should also explain in common language how your solution accomplishes the task.

  Ex. 1) List all contents of the Instructor table.

  The code should look like:

```
/*  1 – list all contents of the Instructor table --
        The * operator in the select clause will
        select all columns from the specified table.
 */
select * from INSTRUCTOR;
```

- To make your code more readable, I would prefer that you **format your select statements as follows**: (note the from, where and order by are all on separate lines indented under the select)

```
select COLUMNS_HERE
  from TABLE_HERE
  where CONDITION1_HERE
        or CONDITION2_HERE
  order by COLUMN_HERE;
```

You are submitting a file that has the sql code in it, not the results. I must be able to run your .sql file in its entirety. Make sure that comments are added properly.

TASKS

1) Create a view called PLACEHOLDER that lists the IDs and names of courses that have never had a section scheduled. Then do a select * from the view to show all rows.

2) Create a view called WARNING that lists the names and average ***current*** grades for students who ***currently*** have less than a 2.5 GPA (***not high school GPA)***. Then do a select * from the view to show all rows.

3) Create a view called OVERWORKED that lists the instructor's name, ID, and number of courses taught for the instructor who taught the most courses in the Spring 2008 semester (term 081). Then do a select * from the view to show all rows.

4) **Use the view created in Task 3** to do the following.
   a. Show the query used to find the instructor's salary before the raise. What is the instructor's salary (answer in comments)?
   b. Show the query used to give the instructor a 5% raise. What is the instructor's salary now (answer in comments)?

5) Create a new table called TASK with the following columns
   i. ID – an integer that will serve as the primary key
   ii. NAME – a varying length character string of no more than 20 characters
   iii. BUDGET – a number up to 999,999.99
   iv. SPONSOR – a foreign key to the INSTRUCTOR table

   b. Add 2 rows to your new TASK table. Use values less than 500,000 for BUDGET.

   c. Do a select * from the table to show the rows you just added.

   d. Add a constraint to TASK to limit the values for the BUDGET column to be between 0 and 500,000 (inclusive). Add this constraint to the existing table definition, do not rebuild the entire table.

   e. Remove the TASK table from your database.

6) If an instructor has a PhD and no given name, give them the name 'Doctor'.
   a. Query used to change the instructor's name.
   b. Query used to verify the changes.

7) Add the following courses to the COURSE table
   i. MSCI1250 – Basic Sand People Tracking, a 3 hour MSCI course.
   ii. EDUC4957 – Jawa Bargaining Techniques, a 3 hour EDUC course.

   b. Verify the rows have been added to the table.

7.5) Undo all changes to your database.

```
/* BuchananBrandonLab8.sql
-------------------------------------------------------------------------
Programmer's Name: Brandon Buchanan
Course: CSCI2020                              Section Number: 002
Creation Date: 11/02/2011          Date of Last Modification: 11/02/2011
E-mail address: zbwb8@goldmail.etsu.edu
-------------------------------------------------------------------------
Purpose -
 Lab 8
-------------------------------------------------------------------------
Notes on specifications, special algorithms, and assumptions:
Designed to work on the database created by the schooldb.sql
script
*/

/* 1. -  Create a PLACEHOLDER view taht displays the IDs and names of courses
         that have never had a section scheduled.  The first query returns
         all courses, while the second query returns all course that have
         been scheduled at least once.  Then the second query is subtracted
         from the first to leave on the course that have never been taught.  */

drop view PLACEHOLDER;

create view PLACEHOLDER as
  select distinct COURSE.ID, COURSE.NAME
    from COURSE
  minus
  select distinct SECTION.COURSE, COURSE.NAME
    from SECTION
    join COURSE
      on SECTION.COURSE = COURSE.ID;

select * from PLACEHOLDER;


/* 2. -  Create a WARNING view that shows students and GPA's for those with a
         current GPA less tahn 2.5.  The view displays students' first names
         and last names along with their average.  The output only displays
         a student when their GPA is less than 2.5.  */

 drop view WARNING;

 create view WARNING as
   select STUDENT.GIVEN_NAME, STUDENT.FAMILY_NAME, avg(TAKES.GRADE) as AVERAGE
     from STUDENT
     join TAKES
       on STUDENT.ID = TAKES.STUDENT
     group by STUDENT.GIVEN_NAME, STUDENT.FAMILY_NAME
     having avg(TAKES.GRADE) < 2.5;

select * from WARNING;
```

93

```
/* 3. - Create a view that displays the instructors' names, IDs, and number of
        classes taught during the Spring 2008 for the instructors who taught
        the most classes.  The outer query displays the names, ID, and number of
        classes taught by each instructor during Spring 2008, while the inner
        query finds the maximum number of classes taught by each instructor
        during the Spring 2008 semester.  */

drop view OVERWORKED;

create view OVERWORKED as
  select INSTRUCTOR.GIVEN_NAME, INSTRUCTOR.FAMILY_NAME, INSTRUCTOR.ID,
         count(*) as SECTIONS_TAUGHT
    from INSTRUCTOR
    join SECTION
      on INSTRUCTOR.ID = SECTION.INSTRUCTOR
    where SECTION.TERM = '081'
    group by INSTRUCTOR.GIVEN_NAME, INSTRUCTOR.FAMILY_NAME, INSTRUCTOR.ID
    having count(*) = (
      select max(count(*))
      from INSTRUCTOR
      join SECTION
        on INSTRUCTOR.ID = SECTION.INSTRUCTOR
      where SECTION.TERM = '081'
      group by INSTRUCTOR.GIVEN_NAME, INSTRUCTOR.FAMILY_NAME, INSTRUCTOR.ID
      );

select * from OVERWORKED;

/* 4.  */

/*  Displays hardest working instructor's salary = $49,500 */
select INSTRUCTOR.SALARY
  from OVERWORKED
  join INSTRUCTOR
    on OVERWORKED.ID = INSTRUCTOR.ID;

/*  Display hardest working instructor's salary with 5% raise = $51,975 */
select INSTRUCTOR.SALARY * 1.05
  from OVERWORKED
  join INSTRUCTOR
    on OVERWORKED.ID = INSTRUCTOR.ID;

/* 5. -  Create a new table called TASK with 4 columns.  */

/* drop table TASK; */

/* a.  Create the table with specified column data types */
create table TASK(
  ID number(8,0) primary key,
  NAME varchar(20),
  BUDGET number(8,2),
  SPONSOR number(8,0) references INSTRUCTOR
```

```
  );

/* b.  Add 2 rows to the table */
insert into TASK values
  (10,'Build C3P0','25000','1957');
insert into TASK values
  (20,'Upgrade blaster','10000','107');

/* c.  Select all from the table  */
select * from TASK;

/* d.  Alter the BUDGER column so that it only allows input between 0 and
       500000 inclusive.  */
alter table TASK add
  (constraint BUDGETRANGE
    check(BUDGET > 0  and BUDGET < 500000 )
  );

/* e.  Remove TASK table */
drop table TASK;


/* 6a. - Change instructor's name to Doctor when that instructor has a PhD. and
         doesn't have a first name.  */

update INSTRUCTOR
  set INSTRUCTOR.GIVEN_NAME = 'Doctor'
  where INSTRUCTOR.DEGREE_TITLE = 'PhD'
    and INSTRUCTOR.GIVEN_NAME is null;

/* 6b. - Verify changes of 6a.  */

select INSTRUCTOR.GIVEN_NAME
  from INSTRUCTOR
  where INSTRUCTOR.DEGREE_TITLE = 'PhD'
    and INSTRUCTOR.GIVEN_NAME = 'Doctor';

/* 7a. -  Insert two courses into the database.   */

insert into COURSE values
  ('MSCI1250','Basic Sand People Tracking','3','MSCI');
insert into COURSE values
  ('EDUC4957','Jawa Bargaining Techniques','3','EDUC');

/* 7b. - Verify changes of 7a, displays two rows entered. */

select * from COURSE
  where COURSE.ID = 'MSCI1250'
     or COURSE.ID = 'EDUC4957';

/* 7.5. -  Undo changes to the database.  */
rollback;
```