Solution to Homework #5

Peter Mackenzie-Helnwein

November 16, 2015

Setup

```
/* *******************************************
 *                                           *
 *          CEE 505 - Fall 2015              *
 *                                           *
 *    Assignment #5: SQL Database access     *
 *                                           *
 * *******************************************/

-- adjusting settings for documentation --
.open Scores.db
.mode columns
.headers on
.output solution.txt
```

Question 1

```
-- 1.) Output a list of students born between June 16, 1991 and September 15, 1996
.print "\n\nQuestion␣1\n
    ================================================================================\n"
SELECT s.name FROM Students AS s
    WHERE s.DOB >= '1991-06-16'
    AND   s.DOB <= '1996-09-15';
```

```

Question 1
================================================================================

name
----------
F4242
F4243
F4248
F424B
F424F
F4252
F4255
F425A
F425B
F425D
F425F
F4264
F426B
F426E
F4273
F4275
F4282
F4283
F4285
F4287
F428B
F428D
```

F428F
F4291
F4293
F4294
F4295
F4298
F429E
F42A1
F42A2
F42A4
F42A8
F42A9
F42AA
F42AB
F42B0
F42B4
F42B6
F42B7
F42B9
F42BD
F42BE
F42BF
F42C0
F42C1
F42C7
F42C8
F42CB
F42D1
F42D5
F42D7
F42DA
F42DC
F42DE
F42DF
F42E0
F42E3
F42E7
F42EB
F42EC
F42EE
F42F1
F42F2
F42F4
F42F5
F42F6
F42F7
F42F8
F42FE
F42FF
F4304
F4305
F4306
F4308
F430B
F4312
F4315
F4316
F4318
F4319
F431B
F431C
F431E
F4325
F432A
F432E

Question 2

```
-- 2.) Output the number of students born between June 16, 1991 and September 15, 1996
.print "\n\nQuestion 2\n
    ==============================================================================\n"
SELECT count(s.name) FROM Students AS s
    WHERE s.DOB >= '1991-06-16'
    AND   s.DOB <= '1996-09-15';
```

```
Question 2
================================================================================

count(s.name)
-------------
87
```

Question 3

```
-- 3.) Output a list of students who have missed one or more labs
--     (Score <= 0.1 to avoid numeric truncation errors)
.print "\n\nQuestion 3\n
    ====================================================================================\n"
SELECT DISTINCT s.name, sc.Score, a.name, t.name as type
FROM Students AS s, Scores AS sc, Types as t, Assignments as a
    WHERE sc.Score < 0.01
    AND sc.StudentID = s.ID
    AND t.typeID = a.typeID
    AND a.ID = sc.AssignmentID
    AND t.name LIKE 'Lab%'
    ;
```

```
Question 3
================================================================================

name          Score         name                  type
----------    ----------    ------------------    ----------
F4251         0.0           Lab #3 (2786810)      Labs
F4253         0.0           Lab #1 (2829219)      Labs
F425D         0.0           Lab #2 (2786809)      Labs
F425D         0.0           Lab #7 (2786813)      Labs
F426A         0.0           Lab #1 (2829219)      Labs
F4276         0.0           Lab #2 (2786809)      Labs
F427A         0.0           Lab #6 (2786812)      Labs
F4280         0.0           Lab #10 (2786815      Labs
F4281         0.0           Lab #10 (2786815      Labs
F4287         0.0           Lab #6 (2786812)      Labs
F4291         0.0           Lab #10 (2786815      Labs
F429B         0.0           Lab #10 (2786815      Labs
F429D         0.0           Lab #3 (2786810)      Labs
F42AF         0.0           Lab #1 (2829219)      Labs
F42AF         0.0           Lab #10 (2786815      Labs
F42B5         0.0           Lab #1 (2829219)      Labs
F42B6         0.0           Lab #3 (2786810)      Labs
F42C4         0.0           Lab #1 (2829219)      Labs
F42EC         0.0           Lab #1 (2829219)      Labs
F42EC         0.0           Lab #2 (2786809)      Labs
F4303         0.0           Lab #1 (2829219)      Labs
F4305         0.0           Lab #1 (2829219)      Labs
F430B         0.0           Lab #8 (2870743)      Labs
F4311         0.0           Lab #2 (2786809)      Labs
F4311         0.0           Lab #7 (2786813)      Labs
F431B         0.0           Lab #8 (2870743)      Labs
F4324         0.0           Lab #8 (2870743)      Labs
F4325         0.0           Lab #1 (2829219)      Labs
```

Question 4

```
-- 4.) Output the name of the student with the best score at the final
.print "\n\nQuestion 4\n
    ==========================================================================================\n"
-- let's start finding the maximum score:
SELECT MAX(sc.Score) AS maxGPA
    FROM Scores AS sc, Types as t, Assignments as a
    WHERE t.typeID = a.typeID
      AND a.ID = sc.AssignmentID
      AND t.name LIKE 'Final%'
    ;

-- find the actual answer using nested SELECT statements
DROP TABLE IF EXISTS sol4;
CREATE TABLE sol4 AS
    SELECT s.name, sc.Score
        FROM Students as s, Scores as sc,
             (
            SELECT MAX(sc.Score) AS maxGPA
                FROM Scores AS sc, Types as t, Assignments as a
                WHERE t.typeID = a.typeID
                  AND a.ID = sc.AssignmentID
                  AND t.name LIKE 'Final%'
             ) AS aa
        WHERE s.ID = sc.StudentID
          AND aa.maxGPA = sc.Score
        ;
SELECT * FROM sol4;
```

```
Question 4
==========================================================================================

maxGPA
----------
96.0
name        Score
----------  ----------
F42DC       96.0
F430D       96.0
```

Question 5

```
-- 5.) Output the name of the student closest to the average score of midterm 1
.print "\n\nQuestion 5\n
    ===========================================================================\n"

-- here is how to find the average score:
DROP VIEW IF EXISTS final;
CREATE VIEW final AS
    SELECT s.name, sc.Score
        FROM Scores AS sc, Students AS s, Assignments AS a
        WHERE s.ID = sc.StudentID
          AND a.ID = sc.AssignmentID
          AND a.name LIKE 'Midterm 1%'
        ;
DROP VIEW IF EXISTS stats;
CREATE VIEW stats AS
    SELECT AVG(sc.Score) AS avgGPA, MAX(sc.Score) AS maxGPA
        FROM Scores AS sc, Assignments as a
        WHERE a.ID = sc.AssignmentID
          AND a.name LIKE 'Midterm 1%'
        ;
SELECT * FROM stats;

-- now look for a list of students closest to the average score
DROP VIEW IF EXISTS difference;
CREATE VIEW difference AS
    SELECT f.name, f.Score, ABS(f.Score - sts.avgGPA) AS diff
        FROM final AS f, stats AS sts
        ORDER by diff
        ;
--SELECT * FROM difference;
DROP VIEW IF EXISTS sol5;
CREATE VIEW sol5 AS
    SELECT f.name AS AvgStudents, f.Score AS Score
        FROM difference AS f,
             (
             SELECT MIN(df.diff) AS  minDev
                FROM difference as df
             ) as df
        WHERE f.diff = df.minDev
    ;
SELECT * FROM sol5;
```

```
Question 5
===========================================================================

avgGPA              maxGPA
----------------    ----------
66.2238493723849    100.0
AvgStudents   Score
-----------   ----------
F4274         66.0
F4296         66.0
F42C0         66.0
```

Question 6

There are many way to answer Question 6 using SQL statements. I will provide three (3) different solutions, all of which use results from previous searches without entering them manually into a WHERE clause. The provided solutions will work even if the input Scores.db would change. Keep thi sin mind for futur design of your SELECT statements.

Solution #1 is the closest to what I expect you to come up with. It's a little bit lengthy and will fail if more than 6 students satisfy conditions 4 and 5:

```
-- 6.) Output the accumulated homework score (sum of all assignment-type score)
--     for the students identified in 4. and 5., respectively.
.print "\n\nQuestion␣6\n
    ==================================================================================\n"

-- setup
DROP TABLE IF EXISTS targetStudents;
DROP VIEW IF EXISTS AssmntScore;
DROP VIEW IF EXISTS LabScore;
DROP VIEW IF EXISTS MidtermScore;
DROP VIEW IF EXISTS FinalScore;

-- create a list of students
CREATE TABLE targetStudents (
    name     text,
    Score    double
    );
INSERT INTO targetStudents (name, Score)
    SELECT * FROM sol4;
INSERT INTO targetStudents (name, Score)
    SELECT * FROM sol5;
DROP VIEW IF EXISTS firstStudent;
CREATE VIEW firstStudent AS
    SELECT * FROM targetStudents ORDER BY ROWID ASC LIMIT 1;


-- <1> homework score total
CREATE VIEW AssmntScore AS
    SELECT SUM(sc.Score) AS TotalAssignmentScore
        FROM Scores AS sc, Students AS s, Assignments AS a, Types as t
        WHERE sc.StudentID = s.ID
          AND sc.AssignmentID = a.ID
          AND a.typeID = t.typeID
          AND t.name LIKE 'Assign%'
          AND s.name IN (SELECT name FROM firstStudent)
        ;
-- <2> Labs score total
CREATE VIEW LabScore AS
    SELECT SUM(sc.Score) AS TotalLabScore
        FROM Scores AS sc, Students AS s, Assignments AS a, Types as t
        WHERE sc.StudentID = s.ID
          AND sc.AssignmentID = a.ID
          AND a.typeID = t.typeID
          AND t.name LIKE 'Lab%'
          AND s.name IN (SELECT name FROM firstStudent)
        ;
-- <3> midterm score total
CREATE VIEW MidtermScore AS
    SELECT SUM(sc.Score) AS TotalMidtermScore
        FROM Scores AS sc, Students AS s, Assignments AS a, Types as t
        WHERE sc.StudentID = s.ID
          AND sc.AssignmentID = a.ID
          AND a.typeID = t.typeID
          AND t.name LIKE 'Mid%'
          AND s.name IN (SELECT name FROM firstStudent)
        ;
-- <4> final score total
CREATE VIEW finalScore AS
    SELECT SUM(sc.Score) AS TotalFinalScore
```

```
        FROM Scores AS sc, Students AS s, Assignments AS a, Types as t
        WHERE sc.StudentID = s.ID
          AND sc.AssignmentID = a.ID
          AND a.typeID = t.typeID
          AND t.name LIKE 'Fin%'
          AND s.name IN (SELECT name FROM firstStudent)
        ;
-- create a table to hold the answers to Question 6
DROP TABLE IF EXISTS summary;
CREATE TEMPORARY TABLE summary (
    name            text,
    assmntScore     double,
    labScore        double,
    midtermScore    double,
    finalScore      double
    );

INSERT INTO summary (name, assmntScore, labScore, midtermScore, finalScore)
    SELECT f.name,
           TotalAssignmentScore,
           TotalLabScore,
           TotalMidtermScore,
           TotalFinalScore
    FROM firstStudent as f, assmntScore, labScore, midtermScore, finalScore
    ;
DELETE FROM targetStudents
    WHERE name = (SELECT name FROM firstStudent);

INSERT INTO summary (name, assmntScore, labScore, midtermScore, finalScore)
    SELECT f.name,
           TotalAssignmentScore,
           TotalLabScore,
           TotalMidtermScore,
           TotalFinalScore
    FROM firstStudent as f, assmntScore, labScore, midtermScore, finalScore
    ;
DELETE FROM targetStudents
    WHERE name = (SELECT name FROM firstStudent);

INSERT INTO summary (name, assmntScore, labScore, midtermScore, finalScore)
    SELECT f.name,
           TotalAssignmentScore,
           TotalLabScore,
           TotalMidtermScore,
           TotalFinalScore
    FROM firstStudent as f, assmntScore, labScore, midtermScore, finalScore
    ;
DELETE FROM targetStudents
    WHERE name = (SELECT name FROM firstStudent);

INSERT INTO summary (name, assmntScore, labScore, midtermScore, finalScore)
    SELECT f.name,
           TotalAssignmentScore,
           TotalLabScore,
           TotalMidtermScore,
           TotalFinalScore
    FROM firstStudent as f, assmntScore, labScore, midtermScore, finalScore
    ;
DELETE FROM targetStudents
    WHERE name = (SELECT name FROM firstStudent);

INSERT INTO summary (name, assmntScore, labScore, midtermScore, finalScore)
    SELECT f.name,
           TotalAssignmentScore,
           TotalLabScore,
           TotalMidtermScore,
           TotalFinalScore
    FROM firstStudent as f, assmntScore, labScore, midtermScore, finalScore
    ;
```

```
DELETE FROM targetStudents
    WHERE name = (SELECT name FROM firstStudent);

INSERT INTO summary (name, assmntScore, labScore, midtermScore, finalScore)
    SELECT f.name,
           TotalAssignmentScore,
           TotalLabScore,
           TotalMidtermScore,
           TotalFinalScore
    FROM firstStudent as f, assmntScore, labScore, midtermScore, finalScore
    ;
DELETE FROM targetStudents
    WHERE name = (SELECT name FROM firstStudent);

SELECT * FROM summary;
DROP TABLE summary;
```

```
Question 6
==========================================================================

name        assmntScore   labScore    midtermScore   finalScore
----------  ------------  ----------  -------------  ----------
F42DC       606.0         103.0       179.0          96.0
F430D       563.0         102.0       179.0          81.5
F4274       561.0         103.0       165.0          69.0
F4296       606.0         101.5       151.0          80.5
F42C0       606.0         101.0       142.5          71.5
```

Alternative solutions to Question 6

This solution to Question 6 is the most compact I could come up with. It will work for different Score.db as well as any arbitrary number of students satisfying conditions 4 and 5. This solution would also allow a larger number of types of assignments if properly defined in the Types TABLE. The key to this solution is the GROUP BY statement, which works a bit like a loop when computing student specific sums of scores.

```
-- 6.) Output the accumulated homework score (sum of all assignment-type score)
--     for the students identified in 4. and 5., respectively.
.print "\n\nQuestion 6 - alternative solution\n
    ==========================================================================\n"
-- setup
DROP TABLE IF EXISTS targetStudents;

-- create a list of students
CREATE TABLE targetStudents (
     name     text,
     Score    double
     );
INSERT INTO targetStudents (name, Score)
     SELECT * FROM sol4;
INSERT INTO targetStudents (name, Score)
     SELECT * FROM sol5;

-- unknown number of types
SELECT s.name, SUM(sc.Score) AS groupTotal, t.name as assignment, t.typeID
    FROM Scores AS sc, Students AS s, Assignments AS a, Types as t
    WHERE sc.StudentID = s.ID
      AND sc.AssignmentID = a.ID
      AND a.typeID = t.typeID
      AND s.name IN (SELECT name FROM targetStudents)
    GROUP BY s.name, t.typeID
    ORDER BY s.name, t.typeID ;

-- known number of types
DROP VIEW IF EXISTS altAssignments;
```

```
Question 6 - alternative solution
==========================================================================


name        groupTotal  assignment  typeID
----------  ----------  ----------  ----------
F4274       561.0       Assignment  1
F4274       103.0       Labs        2
F4274       165.0       Midterm     3
F4274       69.0        Final       4
F4296       606.0       Assignment  1
F4296       101.5       Labs        2
F4296       151.0       Midterm     3
F4296       80.5        Final       4
F42C0       606.0       Assignment  1
F42C0       101.0       Labs        2
F42C0       142.5       Midterm     3
F42C0       71.5        Final       4
F42DC       606.0       Assignment  1
F42DC       103.0       Labs        2
F42DC       179.0       Midterm     3
F42DC       96.0        Final       4
F430D       563.0       Assignment  1
F430D       102.0       Labs        2
F430D       179.0       Midterm     3
F430D       81.5        Final       4
```

Alternative solutions to Question 6

```
-- 6.) Output the accumulated homework score (sum of all assignment-type score)
--      for the students identified in 4. and 5., respectively.
.print "\n\nQuestion 6 - alternative solution\n
       ====================================================================================\n"
-- setup
DROP TABLE IF EXISTS targetStudents;

-- create a list of students
CREATE TABLE targetStudents (
     name     text,
     Score    double
     );
INSERT INTO targetStudents (name, Score)
     SELECT * FROM sol4;
INSERT INTO targetStudents (name, Score)
     SELECT * FROM sol5;

         FROM Scores AS sc, Students AS s
         WHERE sc.StudentID = s.ID
           AND s.name IN (SELECT name FROM targetStudents) ;

SELECT s.name,
        sc1.total AS Homework,
        sc2.total AS Labs,
        sc3.total AS Midterms,
        sc4.total AS Final
    FROM
           (SELECT SUM(Score) AS total, StudentID
            FROM reducedScores AS rsc, altAssignments AS aa
            WHERE rsc.AssignmentID = aa.ID
              AND aa.class LIKE "_ssignment%"
            GROUP BY StudentID
           ) AS sc1,
           (SELECT SUM(Score) AS total, StudentID
            FROM reducedScores AS rsc, altAssignments AS aa
            WHERE rsc.AssignmentID = aa.ID
              AND aa.class LIKE "Lab%"
            GROUP BY StudentID
           ) AS sc2,
           (SELECT SUM(Score) AS total, StudentID
            FROM reducedScores AS rsc, altAssignments AS aa
            WHERE rsc.AssignmentID = aa.ID
              AND aa.class LIKE "_idterm%"
            GROUP BY StudentID
           ) AS sc3,
           (SELECT SUM(Score) AS total, StudentID
            FROM reducedScores AS rsc, altAssignments AS aa
            WHERE rsc.AssignmentID = aa.ID
              AND aa.class LIKE "_inal%"
            GROUP BY StudentID
           ) AS sc4,
           Students AS s
    WHERE sc1.StudentID = s.ID
      AND sc2.StudentID = s.ID
      AND sc3.StudentID = s.ID
      AND sc4.StudentID = s.ID
    ORDER BY s.name ;
```

```
name          Homework      Labs          Midterms      Final
----------    ----------    ----------    ----------    ----------
F4274         561.0         103.0         165.0         69.0
F4296         606.0         101.5         151.0         80.5
F42C0         606.0         101.0         142.5         71.5
F42DC         606.0         103.0         179.0         96.0
F430D         563.0         102.0         179.0         81.5
```

Question 7

```
-- 7.) Create a VIEW named altAssignments, listing Assignment.ID, Assignment.name,
--     Type.name, and sorted by Type.name.  For reporting use
--         sqlite3> SELECT * FROM altAssignments;
--     and provide the output of
--         sqlite3> .schema altAssignment
--     and explain what it tells you.
.print "\n\nQuestion␣7\n
    ====================================================================================\n"

DROP VIEW IF EXISTS altAssignments;
CREATE VIEW altAssignments AS
    SELECT a.ID AS ID, a.name AS name, t.name as TypeName
    FROM Assignments as a, Types as t
    WHERE a.typeID = t.typeID;
SELECT * FROM altAssignments;
```

```
Question 7
=================================================================================

ID          name                                TypeName
----------  ----------------------------------  ----------
1           Homework Assignment #1 (2786783)    Assignment
2           Lab #1 (2829219)                    Labs
3           Homework Assignment #2 (2786789)    Assignment
4           Lab #2 (2786809)                    Labs
5           Homework Assignment #3 (2786785)    Assignment
6           Lab #3 (2786810)                    Labs
7           Homework Assignment #4 (2786784)    Assignment
8           Lab #4 (2786811)                    Labs
9           Homework Assignment #5 - Quick A    Assignment
10          Homework Assignment #5 - Problem    Assignment
11          Lab #5 (2856765)                    Labs
12          Midterm 1 (2786796)                 Midterm
13          Homework Assignment #6 (2786791)    Assignment
14          Lab #6 (2786812)                    Labs
15          Homework Assignment #7 (2786790)    Assignment
16          Lab #7 (2786813)                    Labs
17          Homework Assignment #8 (2786787)    Assignment
18          Midterm 2 (2786797)                 Midterm
19          Lab #8 (2870743)                    Labs
20          Lab #9 - Beam Lab (2786814)         Labs
21          Lab #10 (2786815)                   Labs
22          Final Exam (2786798)                Final
23          Bonus Assignment #9 (2786795)       Assignment
```

```
.schema altAssignment
```

```
CREATE VIEW altAssignment AS
    SELECT a.*, t.typeID
                FROM Assignments as a, Types as t
                WHERE a.typeID = t.typeID;
```

This output shows that the VIEW is not a table but rather a stores SELECT statement to be executed when accessing the view as if it were a table.

Usefull extension of Question 7

```
-- for practice (not part of the Assignment), create a similar VIEW but for Labs only
DROP VIEW IF EXISTS Labs;
CREATE VIEW Labs AS
SELECT * FROM altAssignments AS a
    WHERE a.TypeName LIKE 'Labs%';
SELECT * FROM Labs;
```

```
ID            name                  TypeName
----------    ------------------    ----------
2             Lab #1 (2829219)      Labs
4             Lab #2 (2786809)      Labs
6             Lab #3 (2786810)      Labs
8             Lab #4 (2786811)      Labs
11            Lab #5 (2856765)      Labs
14            Lab #6 (2786812)      Labs
16            Lab #7 (2786813)      Labs
19            Lab #8 (2870743)      Labs
20            Lab #9 - Beam La      Labs
21            Lab #10 (2786815      Labs
```

Question 8

```
-- 8.) Create a series of INSERT statements that create a user entry for yourself,
--     full score on all homeworks, 80% on Midterm 1, 90% on Midterm 2, and
--     99% on the Final.
--     Show all the newly added information through SELECT statements on the
--     respective tables (make sure to design those SELECT statements to filter
--     only those showing data for your record)
.print "\n\nQuestion 8\n
    ==========================================================================\n"

-- use a large ID to make sure we are dealing with a unique ID.
-- I achieve this by DELETE-ing any entry that might exist for that ID:
DELETE FROM Students WHERE ID = 654987;
DELETE FROM Scores WHERE StudentID = 654987;

INSERT INTO Students (ID, name, DOB) VALUES (654987, 'Peter Mackenzie', '1965-11-18');

-- working with an alternative Scores table to eliminate the key creation issue
DROP TABLE IF EXISTS MyScores;
CREATE TABLE MyScores (
   itemID               integer primary key autoincrement,
   AssignmentID         int,
   StudentID            int,
   Score                double
   );

-- copy Scores to MyScores
INSERT INTO MyScores SELECT * from Scores;

-- now add Homework Assignment Scores for Peter Mackenzie
INSERT INTO MyScores (StudentID, AssignmentID, Score)
    SELECT s.ID, a.ID, a.targetScore
        FROM Students as s, Assignments as a
        WHERE s.name LIKE '%Mackenz%'
          AND a.typeID = 1;

-- now add Midterm Scores for Peter Mackenzie
INSERT INTO MyScores (StudentID, AssignmentID, Score)
    SELECT s.ID, a.ID, 0.80*a.targetScore
        FROM Students as s, Assignments as a
        WHERE s.name LIKE '%Mackenz%'
          AND a.typeID = 3
          AND (a.name LIKE '%idterm 1%' OR a.name LIKE '%idterm1%' OR a.name LIKE '%idterm #1%
              ' )
          ;
INSERT INTO MyScores (StudentID, AssignmentID, Score)
    SELECT s.ID, a.ID, 0.90*a.targetScore
        FROM Students as s, Assignments as a
        WHERE s.name LIKE '%Mackenz%'
          AND a.typeID = 3
          AND (a.name LIKE '%idterm 2%' OR a.name LIKE '%idterm2%' OR a.name LIKE '%idterm #2%
              ' )
          ;

-- now add the score for the final for Peter Mackenzie
INSERT INTO MyScores (StudentID, AssignmentID, Score)
    SELECT s.ID, a.ID, 0.99*a.targetScore
        FROM Students as s, Assignments as a
        WHERE s.name LIKE '%Mackenz%'
          AND a.typeID = 4;

-- verify all scores for Peter Mackenzie
SELECT a.name, sc.Score, a.targetScore, 100*sc.Score/a.targetScore
    FROM Assignments AS a, MyScores AS sc, Students AS s
    WHERE s.ID = sc.StudentID
      AND s.name LIKE '%Mackenzie%'
      AND a.ID = sc.AssignmentID;
```

```
Question 8
================================================================================

name                                Score       targetScore   100*sc.Score/a.targetScore
----------------------------------  ----------  ------------  ----------------------------
Homework Assignment #1 (2786783)    60.0        60.0          100.0
Homework Assignment #2 (2786789)    60.0        60.0          100.0
Homework Assignment #3 (2786785)    70.0        70.0          100.0
Homework Assignment #4 (2786784)    80.0        80.0          100.0
Homework Assignment #5 - Quick A    21.0        21.0          100.0
Homework Assignment #5 - Problem    50.0        50.0          100.0
Homework Assignment #6 (2786791)    70.0        70.0          100.0
Homework Assignment #7 (2786790)    60.0        60.0          100.0
Homework Assignment #8 (2786787)    60.0        60.0          100.0
Bonus Assignment #9 (2786795)       20.0        20.0          100.0
Midterm 1 (2786796)                 80.0        100.0         80.0
Midterm 2 (2786797)                 90.0        100.0         90.0
Final Exam (2786798)                99.0        100.0         99.0
```