

Tarpaulin aka Tarp

Sailing the ICs

Dylan Bartness, Carson Davis, Rodney Erickson, Carter Grove, Keiley Maahs

Phase 1

- A. The proposed system is an Online Learning Management System (OLMS) designed to facilitate the creation, management, and delivery of educational courses. The system provides a user-friendly interface for students and instructors, allowing efficient handling of course materials, assessments, and student progress tracking.
- B. There are two types of users: Students and Instructors. Users can either be a student or an instructor, but not both. Users have unique usernames, passwords, and their names.

Each student earns points based on their completed courses and grades. A student can rate instructors whom they have taken a course from. A student can enroll in courses and can rate courses they have taken. A student can take tests. A student can view a list of lectures for a course. A student can watch lectures. A student can view their grade report. A student can view a list of tests for a course. A student can search for courses using filters for course ratings, instructors, and topics. Students also have a grade report for each course they are enrolled in. The grade report consists of the student's grade, the start date for the course, and the date it was completed. Students can create and join communities. In the communities, students can view rankings which show how well they are doing compared to other students.

Instructors can create and make courses visible. In the courses, instructors can add new lectures and new tests. Tests are multiple choice. Each course has a QBoard where users can write questions and answers to each other.

Courses have a unique course number, course name, visibility, the instructor that created it, and can have a Q Board that both students who enrolled in the course and the instructor of the course can post questions on.

Q (question) Boards have any number of questions.

Lectures have a lecture ID, a title, a length, and video URL.

A community has a community ID and tracks the number of people in the community.

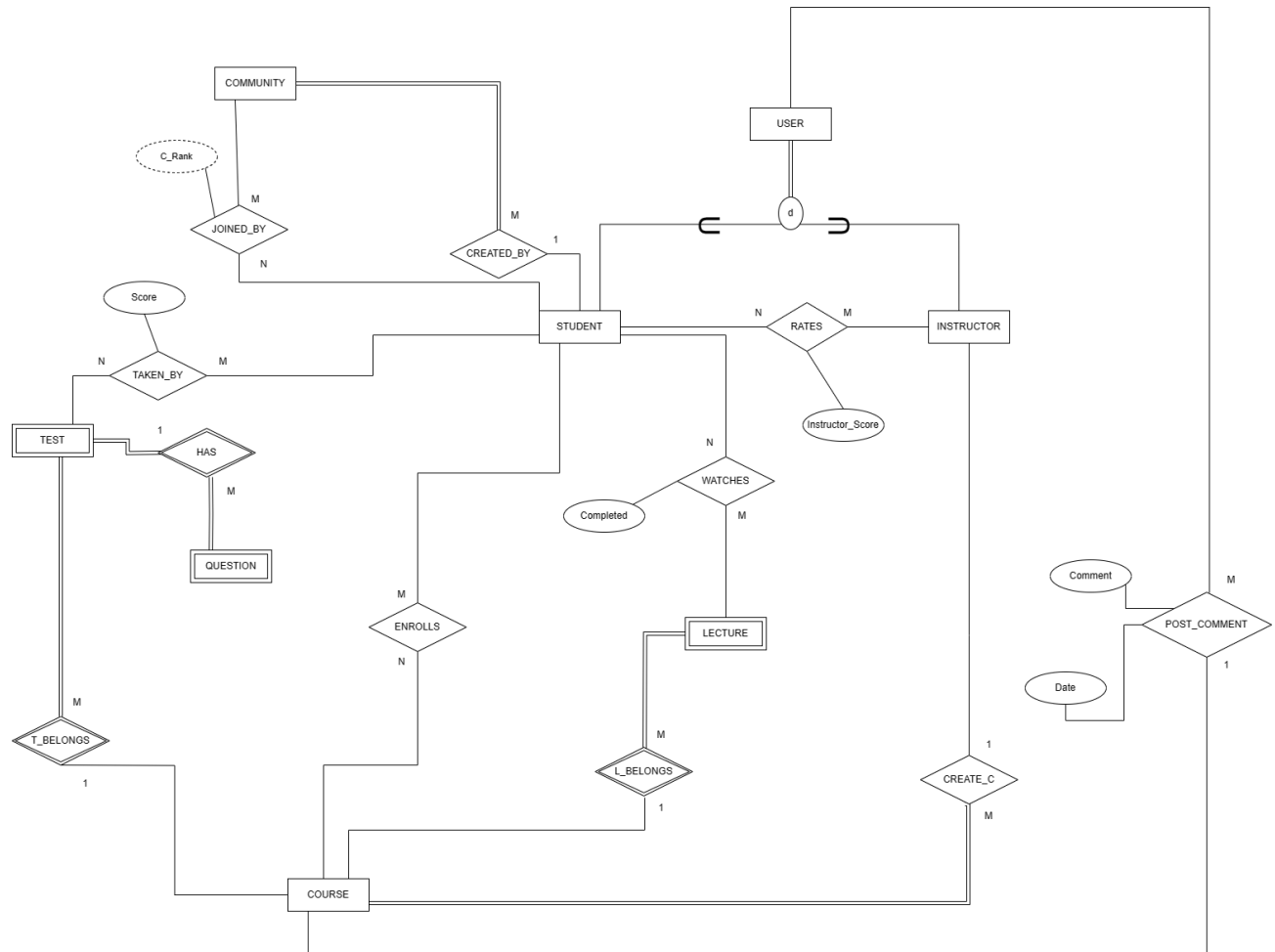
C. Table

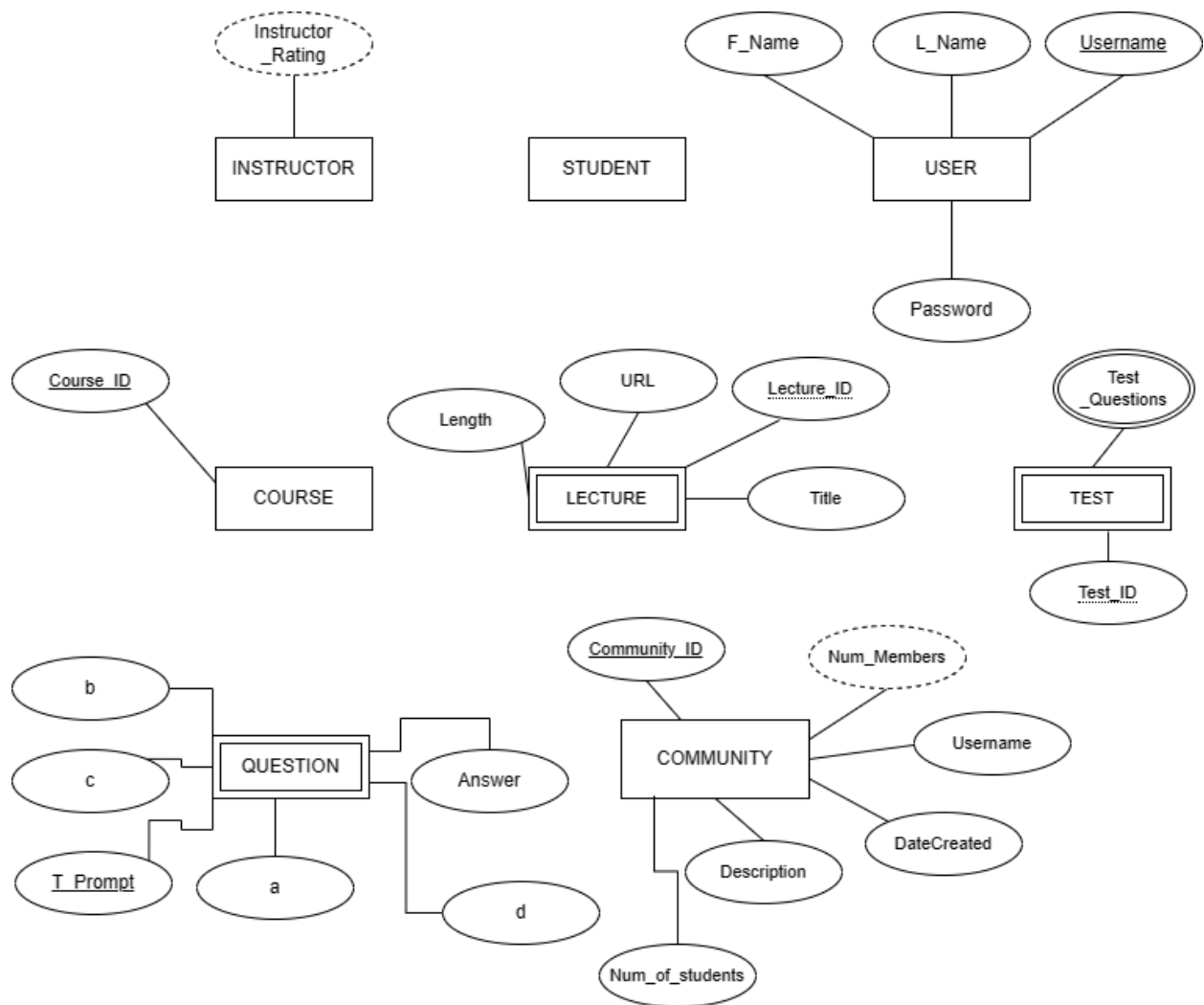
| Proposed Functionality | Member Responsible | Brief Description |
|--|---------------------------|---|
| Login | Dylan | Allow users to enter username and password and verify the credentials. |
| Create Courses (Instructor) | Carson | Before posting a course, an instructor creates a course not yet visible to students. |
| Modify Profile | Rodney | A user can modify their profile (update password, update username, etc.) |
| Delete Courses (Instructor) | Carter | An instructor can remove a course they have created. |
| Create lectures (Instructor) | Keiley | Instructors can create lectures for courses they have created. |
| Enroll in course (Student) | Dylan | Students can enroll themselves in a course to gain access to tests and lectures posted by the instructor |
| Create Community | Carson | Students can create a community for any student to join. |
| Delete Community | Rodney | The Student that created the Community can delete the community at any time. |
| Join Community | Carter | Students can join a community with other students to view each other's progress. |
| Rank Students in Community by points | Keiley | Click a button within Community page to view student rankings (largest to smallest number of points earned). |
| Add/Delete to the QBoard | Dylan | A user can add or delete comments on the QBoard (Question Board) for a course. |
| Complete tests (students) | Carson | A student can complete a test for a course and receive a grade. |
| View Grade Report for a course (Student) | Rodney | Students can view their current grade reports for a course they are enrolled in to see their progress. |
| Watch Lecture (Student) | Carter | Students can view lectures within courses they are enrolled in. |
| List Tests (Student) | Keiley | A student can view all the tests for a given course. |
| Add new test to course (Instructor) | Dylan | An instructor can add a test they have created to a course they have created, if the course has not yet been published. |
| List Lectures (Student) | Carson | A student can click a button to list all the lectures in a course they are enrolled in |

| | | |
|---------------------------------|--------|--|
| | | |
| Search for courses (Student) | Rodney | A student can search for courses (filtered by topic or rating) |
| Rate Course (Student) | Carter | Students can rate courses they are enrolled in |
| Rate Instructor (Student) | Keiley | Students can rate how well the instructor taught the course as long as they are enrolled |

Phase 2

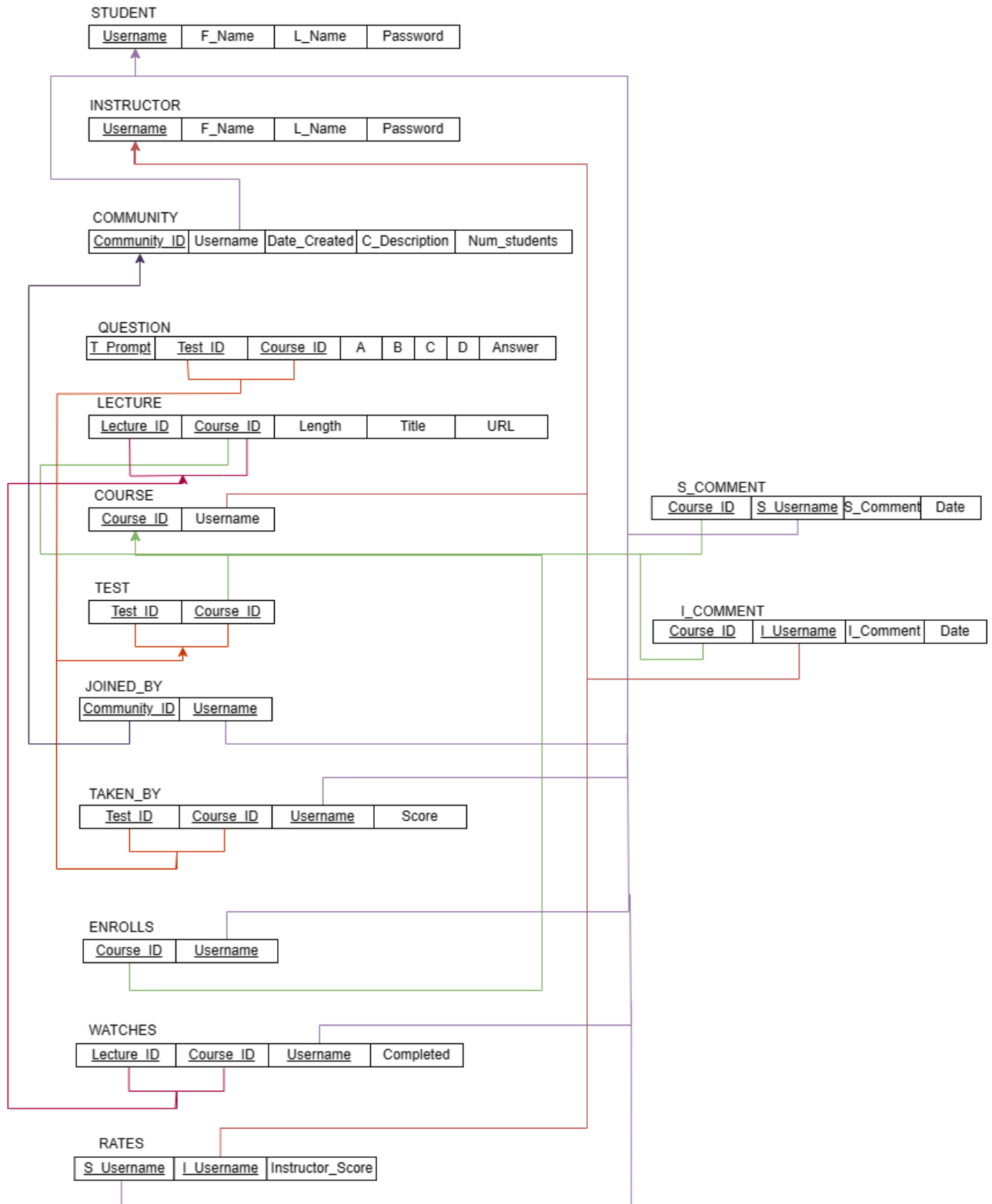
(1) Conceptual database design—High-level data modeling via ER/EER diagrams:





(2) Logical database design–ER/EER to relational mapping:

We decided to map the USER, STUDENT, and INSTRUCTOR specialization as multiple relations – subtype relations only. We did this because the superclass requires total participation, and the specialization is disjoint. For the POST_COMMENT relationship between USER and COURSE, we made lookup relations S_COMMENT and I_COMMENT because the USER side had partial participation and null values were expected. For the RATES relationship between STUDENT and INSTRUCTOR we decided to map this construct as a lookup table called RATES because the participation is many to many and neither side has total participation. For the CREATED_BY relationship between STUDENT and COMMUNITY we decided to map this as a foreign key because there is total participation on the many side of the relationship. For the JOINED_BY relationship between STUDENT and COMMUNITY we decided to map this as a lookup table called JOINED_BY because the participation is many to many and neither side has total participation. For the TAKEN_BY relationship between TEST and STUDENT we decided to map this as another lookup table called TAKEN_BY because the participation is many to many and neither side has total participation. For the ENROLLS relationship between STUDENT and COURSE we mapped this as a lookup table (called ENROLLS) as well because the participation is many to many and neither side has total participation. For the WATCHES relationship between STUDENT and LECTURE we mapped this as a lookup table called WATCHES because the participation is many to many and neither side has total participation. For the HAS relationship between TEST and QUESTION we mapped this as a foreign key because the many side of the relationship has total participation. We also mapped QUESTION and TEST as lookup tables because they are weak entities. We mapped the T_BELONGS relationship between TEST and COURSE as a foreign key because the many side of the relationship has total participation. The relation L_BELONGS between COURSE and LECTURE was mapped as a foreign key because there is total participation on the many side of the relationship. LECTURE was also mapped as its own relation because it is a weak entity. The CREATE_C relationship between COURSE and INSTRUCTOR was mapped as a foreign key because there is total participation on the many side of the relationship.



(3) Physical database design—relational to Oracle mapping:

Creating tables:

```
1  -- tarp_student
2  DROP TABLE tarp_student CASCADE CONSTRAINTS;
3  CREATE TABLE tarp_student (
4      username  varchar(20),
5      f_name    varchar(20),
6      l_name    varchar(20),
7      password  varchar(20),
8      primary key (username)
9  );
10
11 -- tarp_instructor
12 DROP TABLE tarp_instructor CASCADE CONSTRAINTS;
13 CREATE TABLE tarp_instructor (
14     username  varchar(20),
15     f_name    varchar(20),
16     l_name    varchar(20),
17     password  varchar(20),
18     primary key (username)
19 );
20
21 -- tarp_community
22 DROP TABLE tarp_community CASCADE CONSTRAINTS;
23 CREATE TABLE tarp_community (
24     comm_id  varchar(20),
25     username  varchar(20),
26     date_created  date,
27     c_description  varchar(50),
28     num_students  int,
29     primary key (comm_id),
30     Constraint FK_TARP_COMMUNITY foreign key (username) references tarp_student(username)
31 );
32
33 -- tarp_course
34 DROP TABLE tarp_course CASCADE CONSTRAINTS;
35 CREATE TABLE tarp_course (
36     course_id  varchar(20),
37     username  varchar(20),
38     primary key (course_id),
39     Constraint FK_TARP_COURSE foreign key (username) references tarp_instructor(username)
40 );
41
42 -- tarp_s_comment
43 DROP TABLE tarp_s_comment CASCADE CONSTRAINTS;
44 CREATE TABLE tarp_s_comment (
45     course_id  varchar(20),
46     s_username  varchar(20),
47     s_comment  varchar(200),
48     cdate  date,
49     primary key (course_id, s_username),
50     Constraint FK_TARP_S_COMMENT1 foreign key (course_id) references tarp_course(course_id),
51     Constraint FK_TARP_S_COMMENT2 foreign key (s_username) references tarp_student(username)
52 );
53
54 -- tarp_i_comment
55 DROP TABLE tarp_i_comment CASCADE CONSTRAINTS;
56 CREATE TABLE tarp_i_comment (
57     course_id  varchar(20),
58     i_username  varchar(20),
59     i_comment  varchar(200),
60     cdate  date,
61     primary key (course_id, i_username),
62     Constraint FK_TARP_I_COMMENT1 foreign key (course_id) references tarp_course(course_id),
63     Constraint FK_TARP_I_COMMENT2 foreign key (i_username) references tarp_instructor(username)
64 );
65
```


Creating Tables Continued:

```
65
66 -- tarp_lecture
67 DROP TABLE tarp_lecture CASCADE CONSTRAINTS;
68 CREATE TABLE tarp_lecture (
69     lecture_id varchar(20),
70     course_id varchar(20),
71     length decimal(3,1),
72     url varchar(100),
73     primary key (lecture_id, course_id),
74     Constraint FK_TARP_Lecture foreign key (course_id) references tarp_course(course_id)
75 );
76
77 -- tarp_test
78 DROP TABLE tarp_test CASCADE CONSTRAINTS;
79 CREATE TABLE tarp_test (
80     test_id varchar(20),
81     course_id varchar(20),
82     primary key (test_id, course_id),
83     Constraint FK_TARP_TEST foreign key (course_id) references tarp_course(course_id)
84 );
85
86 -- tarp_question
87 DROP TABLE tarp_question CASCADE CONSTRAINTS;
88 CREATE TABLE tarp_question (
89     t_prompt varchar(60),
90     a varchar(30),
91     b varchar(30),
92     c varchar(30),
93     d varchar(30),
94     answer varchar(30),
95     test_id varchar(20),
96     course_id varchar(20),
97     primary key (t_prompt, test_id, course_id),
98     Constraint FK_TARP_QUESTION foreign key (test_id, course_id) references tarp_test(test_id, course_id)
99 );
100
101 -- tarp_joined_by
102 DROP TABLE tarp_joined_by CASCADE CONSTRAINTS;
103 CREATE TABLE tarp_joined_by (
104     community_id varchar(20),
105     username varchar(20),
106     primary key (community_id, username),
107     Constraint FK_TARP_JOINED_BY1 foreign key (community_id) references tarp_community(comm_id),
108     Constraint FK_TARP_JOINED_BY2 foreign key (username) references tarp_student(username)
109 );
110
111 -- tarp_taken_by
112 DROP TABLE tarp_taken_by CASCADE CONSTRAINTS;
113 CREATE TABLE tarp_taken_by (
114     test_id varchar(20),
115     course_id varchar(20),
116     username varchar(20),
117     score int,
118     primary key (test_id, course_id, username),
119     Constraint FK_TARP_TAKEN_BY1 foreign key (course_id) references tarp_course(course_id),
120     Constraint FK_TARP_TAKEN_BY2 foreign key (username) references tarp_student(username)
121 );
122
123 -- tarp_enrolls
124 DROP TABLE tarp_enrolls CASCADE CONSTRAINTS;
125 CREATE TABLE tarp_enrolls (
126     course_id varchar(20),
127     username varchar(20),
128     primary key (course_id, username),
129     Constraint FK_TARP_ENROLLS1 foreign key (course_id) references tarp_course(course_id),
130     Constraint FK_TARP_ENROLLS2 foreign key (username) references tarp_student(username)
131 );
132
```

Creating Tables Continued:

```
132
133 -- tarp_watches
134 DROP TABLE tarp_watches CASCADE CONSTRAINTS;
135 CREATE TABLE tarp_watches (
136     lecture_id varchar(20),
137     course_id varchar(20),
138     username varchar(20),
139     completed CHAR(1),
140     primary key (lecture_id, course_id, username),
141     Constraint FK_TARP_WATCHES1 foreign key (course_id, lecture_id) references tarp_lecture(course_id, lecture_id),
142     Constraint FK_TARP_WATCHES2 foreign key (username) references tarp_student(username)
143 );
144
145 -- tarp_rates
146 DROP TABLE tarp_rates CASCADE CONSTRAINTS;
147 CREATE TABLE tarp_rates (
148     s_username varchar(20),
149     i_username varchar(20),
150     instructor_score int,
151     primary key (s_username, i_username),
152     Constraint FK_TARP_RATES1 foreign key (s_username) references tarp_student(username),
153     Constraint FK_TARP_RATES2 foreign key (i_username) references tarp_instructor(username)
154 );
155
```

Insert Data:

```
1 ALTER TABLE tarp_community DISABLE CONSTRAINT FK_TARP_COMMUNITY;
2 ALTER TABLE tarp_course DISABLE CONSTRAINT FK_TARP_COURSE;
3 ALTER TABLE tarp_s_comment DISABLE CONSTRAINT FK_TARP_S_COMMENT1;
4 ALTER TABLE tarp_s_comment DISABLE CONSTRAINT FK_TARP_S_COMMENT2;
5 ALTER TABLE tarp_i_comment DISABLE CONSTRAINT FK_TARP_I_COMMENT1;
6 ALTER TABLE tarp_i_comment DISABLE CONSTRAINT FK_TARP_I_COMMENT2;
7 ALTER TABLE tarp_lecture DISABLE CONSTRAINT FK_TARP_LECTURE;
8 ALTER TABLE tarp_test DISABLE CONSTRAINT FK_TARP_TEST;
9 ALTER TABLE tarp_question DISABLE CONSTRAINT FK_TARP_QUESTION1;
10 ALTER TABLE tarp_joined_by DISABLE CONSTRAINT FK_TARP_JOINED_BY1;
11 ALTER TABLE tarp_joined_by DISABLE CONSTRAINT FK_TARP_JOINED_BY2;
12 ALTER TABLE tarp_taken_by DISABLE CONSTRAINT FK_TARP_TAKEN_BY1;
13 ALTER TABLE tarp_taken_by DISABLE CONSTRAINT FK_TARP_TAKEN_BY2;
14 ALTER TABLE tarp_enrolls DISABLE CONSTRAINT FK_TARP_ENROLLS1;
15 ALTER TABLE tarp_enrolls DISABLE CONSTRAINT FK_TARP_ENROLLS2;
16 ALTER TABLE tarp_watches DISABLE CONSTRAINT FK_TARP_WATCHES1;
17 ALTER TABLE tarp_watches DISABLE CONSTRAINT FK_TARP_WATCHES2;
18 ALTER TABLE tarp_rates DISABLE CONSTRAINT FK_TARP_RATES1;
19 ALTER TABLE tarp_rates DISABLE CONSTRAINT FK_TARP_RATES2;
20
21 --data
22
23 INSERT INTO tarp_student (username, f_name, l_name, password) VALUES ('erickson25', 'Rodney', 'Erickson', 'password1');
24 INSERT INTO tarp_student (username, f_name, l_name, password) VALUES ('maahs35', 'Keiley', 'Maahs', 'password2');
25 INSERT INTO tarp_student (username, f_name, l_name, password) VALUES ('davis12', 'Carson', 'Davis', 'password3');
26 INSERT INTO tarp_student (username, f_name, l_name, password) VALUES ('grove5', 'Carter', 'Grove', 'password4');
27 INSERT INTO tarp_student (username, f_name, l_name, password) VALUES ('bartness2', 'Dylan', 'Bartness', 'password5');
28
29
30 INSERT INTO tarp_instructor (username, f_name, l_name, password) VALUES ('imad2000', 'Imad', 'Rahal', 'password6');
31 INSERT INTO tarp_instructor (username, f_name, l_name, password) VALUES ('john2', 'John', 'Davidson', 'password7');
32 INSERT INTO tarp_instructor (username, f_name, l_name, password) VALUES ('bill34', 'Bill', 'Billy', 'password8');
33 INSERT INTO tarp_instructor (username, f_name, l_name, password) VALUES ('erin1234', 'Erin', 'Bert', 'password9');
34
35
36 INSERT INTO tarp_community (comm_id, username, date_created, c_description, num_students) VALUES ('csbsju', 'davis12', TIMESTAMP '2024-03-29 10:30:00', 'This group is for CSBSJU students!', 0);
37 INSERT INTO tarp_community (comm_id, username, date_created, c_description, num_students) VALUES ('MN', 'maahs35', TIMESTAMP '2024-03-29 10:30:00', 'This group is for in-state students!', 0);
38
39
40 INSERT INTO tarp_course (course_id, username) VALUES ('CSCI101', 'imad2000');
41 INSERT INTO tarp_course (course_id, username) VALUES ('CSCI331', 'imad2000');
42 INSERT INTO tarp_course (course_id, username) VALUES ('MATH101', 'bill34');
43 INSERT INTO tarp_course (course_id, username) VALUES ('MATH231', 'bill34');
44 INSERT INTO tarp_course (course_id, username) VALUES ('THEO111', 'erin1234');
45
46
47 INSERT INTO tarp_s_comment (course_id, s_username, s_comment, cdate) VALUES ('CSCI101', 'grove5', 'Great course!', TIMESTAMP '2024-03-28 10:30:00');
48 INSERT INTO tarp_s_comment (course_id, s_username, s_comment, cdate) VALUES ('CSCI101', 'erickson25', 'Awesome course!', TIMESTAMP '2024-05-28 10:30:00');
49
50
51 INSERT INTO tarp_i_comment (course_id, i_username, i_comment, cdate) VALUES ('CSCI101', 'imad2000', 'Cool course!', TIMESTAMP '2024-02-28 10:30:00');
52 INSERT INTO tarp_i_comment (course_id, i_username, i_comment, cdate) VALUES ('MATH101', 'bill34', 'Good job everyone!!', TIMESTAMP '2024-05-28 10:30:00');
53
54
55 INSERT INTO tarp_lecture (lecture_id, course_id, length, url) VALUES ('Lecture 1', 'CSCI101', 20.0, 'https://example.com/lecture1/CSCI101');
56 INSERT INTO tarp_lecture (lecture_id, course_id, length, url) VALUES ('Lecture 2', 'CSCI101', 30.0, 'https://example.com/lecture2/CSCI101');
57 INSERT INTO tarp_lecture (lecture_id, course_id, length, url) VALUES ('Lecture 1', 'MATH101', 25.0, 'https://example.com/lecture1/MATH101');
58 INSERT INTO tarp_lecture (lecture_id, course_id, length, url) VALUES ('Lecture 2', 'MATH101', 15.0, 'https://example.com/lecture2/MATH101');
59 INSERT INTO tarp_lecture (lecture_id, course_id, length, url) VALUES ('Lecture 1', 'THEO111', 20.0, 'https://example.com/lecture1/THEO111');
60
61
62 INSERT INTO tarp_test (test_id, course_id) VALUES ('test 1', 'CSCI101');
63 INSERT INTO tarp_test (test_id, course_id) VALUES ('test 2', 'CSCI101');
64 INSERT INTO tarp_test (test_id, course_id) VALUES ('test 3', 'CSCI101');
65 INSERT INTO tarp_test (test_id, course_id) VALUES ('test 1', 'THEO111');
66 INSERT INTO tarp_test (test_id, course_id) VALUES ('test 2', 'THEO111');
67 INSERT INTO tarp_test (test_id, course_id) VALUES ('test 1', 'MATH101');
68 INSERT INTO tarp_test (test_id, course_id) VALUES ('test 2', 'MATH101');
69
```

Insert Table Continued:

```
70
71 INSERT INTO tarp_question (t_prompt, a, b, c, d, answer, test_id, course_id) VALUES ('What is 3 + 1?', '3', '4', '5', '6', 'B', 'test 1', 'MATH101');
72 INSERT INTO tarp_question (t_prompt, a, b, c, d, answer, test_id, course_id) VALUES ('What is 3 * 4?', '13', '14', '25', '12', 'D', 'test 1', 'MATH101');
73 INSERT INTO tarp_question (t_prompt, a, b, c, d, answer, test_id, course_id) VALUES ('What is 3 - 3?', '3', '14', '0', '6', 'C', 'test 1', 'MATH101');
74 INSERT INTO tarp_question (t_prompt, a, b, c, d, answer, test_id, course_id) VALUES ('What is 3 / 1?', '33', '4', '3', '6', 'C', 'test 1', 'MATH101');
75
76
77 INSERT INTO tarp_joined_by (community_id, username) VALUES ('csbsju', 'davis12');
78 INSERT INTO tarp_joined_by (community_id, username) VALUES ('csbsju', 'maahs35');
79 INSERT INTO tarp_joined_by (community_id, username) VALUES ('csbsju', 'grove5');
80 INSERT INTO tarp_joined_by (community_id, username) VALUES ('csbsju', 'bartness2');
81 INSERT INTO tarp_joined_by (community_id, username) VALUES ('csbsju', 'erickson25');
82 INSERT INTO tarp_joined_by (community_id, username) VALUES ('MN', 'davis12');
83
84
85 INSERT INTO tarp_taken_by (test_id, course_id, username, score) VALUES ('test 1', 'MATH101', 'grove5', 90);
86 INSERT INTO tarp_taken_by (test_id, course_id, username, score) VALUES ('test 2', 'MATH101', 'grove5', 80);
87 INSERT INTO tarp_taken_by (test_id, course_id, username, score) VALUES ('test 1', 'MATH101', 'maahs35', 88);
88 INSERT INTO tarp_taken_by (test_id, course_id, username, score) VALUES ('test 1', 'MATH101', 'davis12', 65);
89 INSERT INTO tarp_taken_by (test_id, course_id, username, score) VALUES ('test 1', 'MATH101', 'bartness2', 95);
90 INSERT INTO tarp_taken_by (test_id, course_id, username, score) VALUES ('test 1', 'MATH101', 'erickson25', 80);
91
92
93 INSERT INTO tarp_enrolls (course_id, username) VALUES ('MATH101', 'grove5');
94 INSERT INTO tarp_enrolls (course_id, username) VALUES ('MATH101', 'maahs35');
95 INSERT INTO tarp_enrolls (course_id, username) VALUES ('MATH101', 'davis12');
96 INSERT INTO tarp_enrolls (course_id, username) VALUES ('MATH101', 'bartness2');
97 INSERT INTO tarp_enrolls (course_id, username) VALUES ('MATH101', 'erickson25');
98
99
100 INSERT INTO tarp_watches (lecture_id, course_id, username, completed) VALUES ('Lecture 1', 'MATH101', 'bartness2', 'y');
101 INSERT INTO tarp_watches (lecture_id, course_id, username, completed) VALUES ('Lecture 2', 'MATH101', 'bartness2', 'y');
102 INSERT INTO tarp_watches (lecture_id, course_id, username, completed) VALUES ('Lecture 1', 'MATH101', 'grove5', 'y');
103 INSERT INTO tarp_watches (lecture_id, course_id, username, completed) VALUES ('Lecture 2', 'MATH101', 'grove5', 'n');
104 INSERT INTO tarp_watches (lecture_id, course_id, username, completed) VALUES ('Lecture 1', 'MATH101', 'maahs35', 'y');
105 INSERT INTO tarp_watches (lecture_id, course_id, username, completed) VALUES ('Lecture 2', 'MATH101', 'maahs35', 'n');
106 INSERT INTO tarp_watches (lecture_id, course_id, username, completed) VALUES ('Lecture 1', 'MATH101', 'davis12', 'y');
107 INSERT INTO tarp_watches (lecture_id, course_id, username, completed) VALUES ('Lecture 2', 'MATH101', 'davis12', 'n');
108 INSERT INTO tarp_watches (lecture_id, course_id, username, completed) VALUES ('Lecture 1', 'MATH101', 'erickson25', 'n');
109 INSERT INTO tarp_watches (lecture_id, course_id, username, completed) VALUES ('Lecture 2', 'MATH101', 'erickson25', 'n');
110
111
112 INSERT INTO tarp_rates (s_username, i_username, instructor_score) VALUES ('davis12', 'bill34', 4);
113 INSERT INTO tarp_rates (s_username, i_username, instructor_score) VALUES ('erickson25', 'bill34', 7);
114 INSERT INTO tarp_rates (s_username, i_username, instructor_score) VALUES ('maahs35', 'bill34', 10);
115
116
117
118 ALTER TABLE tarp_community ENABLE CONSTRAINT FK_TARP_COMMUNITY;
119 ALTER TABLE tarp_course ENABLE CONSTRAINT FK_TARP_COURSE;
120 ALTER TABLE tarp_s_comment ENABLE CONSTRAINT FK_TARP_S_COMMENT1;
121 ALTER TABLE tarp_s_comment ENABLE CONSTRAINT FK_TARP_S_COMMENT2;
122 ALTER TABLE tarp_i_comment ENABLE CONSTRAINT FK_TARP_I_COMMENT1;
123 ALTER TABLE tarp_i_comment ENABLE CONSTRAINT FK_TARP_I_COMMENT2;
124 ALTER TABLE tarp_lecture ENABLE CONSTRAINT FK_TARP_Lecture;
125 ALTER TABLE tarp_test ENABLE CONSTRAINT FK_TARP_TEST;
126 ALTER TABLE tarp_question ENABLE CONSTRAINT FK_TARP_QUESTION1;
127 ALTER TABLE tarp_joined_by ENABLE CONSTRAINT FK_TARP_JOINED_BY1;
128 ALTER TABLE tarp_joined_by ENABLE CONSTRAINT FK_TARP_JOINED_BY2;
129 ALTER TABLE tarp_taken_by ENABLE CONSTRAINT FK_TARP_TAKEN_BY1;
130 ALTER TABLE tarp_taken_by ENABLE CONSTRAINT FK_TARP_TAKEN_BY2;
131 ALTER TABLE tarp_enrolls ENABLE CONSTRAINT FK_TARP_ENROLLS1;
132 ALTER TABLE tarp_enrolls ENABLE CONSTRAINT FK_TARP_ENROLLS2;
133 ALTER TABLE tarp_watches ENABLE CONSTRAINT FK_TARP_WATCHES1;
134 ALTER TABLE tarp_watches ENABLE CONSTRAINT FK_TARP_WATCHES2;
135 ALTER TABLE tarp_rates ENABLE CONSTRAINT FK_TARP_RATES1;
136 ALTER TABLE tarp_rates ENABLE CONSTRAINT FK_TARP_RATES2;
137
```

(4) SQL Routines:

Query/Routine 1 (As seen in table of part 5):

```
-- DESCRIPTION: Trigger to update the number of students in a
-- community after an insertion/deletion from TARP_JOINED_BY
```

```
-- CODE
CREATE or REPLACE TRIGGER COMMUNITY_INSERT_DELETE_TRIGGER
AFTER INSERT OR DELETE ON TARP_JOINED_BY
For Each Row
Begin
    IF INSERTING THEN --Add one to students if insertion
        UPDATE TARP_COMMUNITY
        SET     NUM_STUDENTS = NUM_STUDENTS + 1
        WHERE  COMM_ID=:NEW.COMMUNITY_ID;
    ELSE --Subtract one from students if deletion
        UPDATE TARP_COMMUNITY
        SET     NUM_STUDENTS = NUM_STUDENTS - 1
        WHERE  COMM_ID=:OLD.COMMUNITY_ID;
    END IF;
End ;
```

```
-- TEST COMMUNITY_INSERT_DELETE_TRIGGER (1)
```

```
SELECT * FROM TARP_COMMUNITY;
INSERT INTO tarp_joined_by (community_id, username) VALUES('MN','grove5');
SELECT * FROM TARP_COMMUNITY;
```

```
-- OUTPUT
--COMM_ID      USERNAME      DATE_CREA  C_DESCRIPTION      NUM_STUDENTS
-----
--csbsju       davis12       29-MAR-24  This group is for CSBSJU students!      5
--MN           maahs35       29-MAR-24  This group is for in-state students!      1
--
--1 row inserted.
--
--COMM_ID      USERNAME      DATE_CREA  C_DESCRIPTION      NUM_STUDENTS
-----
--csbsju       davis12       29-MAR-24  This group is for CSBSJU students!      5
--MN           maahs35       29-MAR-24  This group is for in-state students!      2
```

Query/Routine 2 (As seen in table of part 5):

```
-- DESCRIPTION: This procedure creates a new lecture using 4 parameters
-- and inserts it into TARP_Lecture
|
-- CODE
CREATE OR REPLACE PROCEDURE create_lecture(p_lecture_id VARCHAR, p_course_id VARCHAR, p_length DECIMAL, p_url VARCHAR)
AS
BEGIN
    INSERT INTO tarp_lecture (lecture_id, course_id, length, url)
    VALUES (p_lecture_id, p_course_id, p_length, p_url);
END;

-- TEST CREATE_Lecture (2)

SELECT * FROM tarp_lecture WHERE course_id = 'CSCI101';
EXEC create_lecture('SQL Queries', 'CSCI101', 25.0, 'https://example.com/SQLQueries/CSCI101');
SELECT * FROM tarp_lecture WHERE course_id = 'CSCI101';

-- OUTPUT
--LECTURE_ID      COURSE_ID      LENGTH URL
-----
--Lecture 1      CSCI101      20 https://example.com/lecture1/CSCI101
--Lecture 2      CSCI101      30 https://example.com/lecture2/CSCI101
--
--
--PL/SQL procedure successfully completed.
--
--LECTURE_ID      COURSE_ID      LENGTH URL
-----
--Lecture 1      CSCI101      20 https://example.com/lecture1/CSCI101
--Lecture 2      CSCI101      30 https://example.com/lecture2/CSCI101
--SQL Queries    CSCI101      25 https://example.com/SQLQueries/CSCI101
```

Query/Routine 3 (As seen in table of part 5):

```
-- DESCRIPTION: This procedure deletes the
-- specified course from the TARP_COURSE table

-- CODE
CREATE OR REPLACE PROCEDURE delete_course(p_course_id VARCHAR)
AS
BEGIN
    DELETE FROM tarp_course
    WHERE course_id = p_course_id;
END;
```

```
-- TEST DELETE_COURSE (3)

SELECT * FROM tarp_course;
INSERT INTO tarp_course (course_id, username) VALUES ('CSCI222', 'imad2000');
SELECT * FROM tarp_course;
EXEC delete_course('CSCI222');
SELECT * FROM tarp_course;

-- OUTPUT
-- COURSE_ID          USERNAME
-----
--CSCI101             imad2000
--CSCI331             imad2000
--MATH101             bill34
--MATH231             bill34
--THE0111            erin1234
--CSCI222             imad2000
--
--6 rows selected.
--
--
--PL/SQL procedure successfully completed.
--
--
-- COURSE_ID          USERNAME
-----
--CSCI101             imad2000
--CSCI331             imad2000
--MATH101             bill34
--MATH231             bill34
--THE0111            erin1234
```

Query/Routine 4 (As seen in table of part 5):

```
-- DESCRIPTION: This procedure enrolls a student in the specified course.

-- CODE
CREATE OR REPLACE PROCEDURE enroll_in_course(p_course_id VARCHAR, p_username VARCHAR)
AS
BEGIN
    INSERT INTO tarp_enrolls (course_id, username)
    VALUES (p_course_id, p_username);
END;

no rows selected

PL/SQL procedure successfully completed.

USERNAME
-----
erickson25
```

Query/Routine 5 (As seen in table of part 5):

```
1 --DESCRIPTION: Procedure to modify a student's password
2 --given the username and their new password
3
4 --CODE
5 CREATE OR REPLACE PROCEDURE modify_student_profile(p_username VARCHAR, p_new_password VARCHAR)
6 AS
7 BEGIN
8     UPDATE tarp_student
9     SET password = p_new_password
10    WHERE username = p_username;
11 END;
```

-- TEST MODIFY_STUDENT_PROFILE (5)

```
SELECT * FROM tarp_student where username = 'erickson25';
EXEC modify_student_profile('erickson25','passwordChanged');
SELECT * FROM tarp_student where username = 'erickson25';
```

-- OUTPUT

| --USERNAME | F_NAME | L_NAME | PASSWORD |
|--|--------|----------|-----------------|
| --erickson25 | Rodney | Erickson | password1 |
| -- | | | |
| -- | | | |
| --PL/SQL procedure successfully completed. | | | |
| -- | | | |
| -- | | | |
| --USERNAME | F_NAME | L_NAME | PASSWORD |
| --erickson25 | Rodney | Erickson | passwordChanged |

Query/Routine 6 (As seen in table of part 5):

```
1 --DESCRIPTION: Procedure to create a new community given the name (id) of the community,
2 --the username of the creator, the date created,
3 --a description, and the number of students in the community (updated by a trigger).
4
5 --CODE
6 CREATE OR REPLACE PROCEDURE create_community(p_comm_id VARCHAR,
7 p_username VARCHAR, p_date_created TIMESTAMPTZ, p_c_description VARCHAR, p_num_students INT)
8 AS
9 BEGIN
10     INSERT INTO tarp_community (comm_id, username, date_created, c_description, num_students)
11     VALUES (p_comm_id, p_username, p_date_created, p_c_description, p_num_students);
12 END;
```

```
-- TEST CREATE_COMMUNITY (6)

SELECT * FROM tarp_community;
EXEC create_community('USA', 'erickson25', TIMESTAMP '2024-01-29 10:30:00', 'This group is for USA students!', 0);
SELECT * FROM tarp_community;

-- OUTPUT
-- COMM_ID      USERNAME      DATE_CREA  C_DESCRIPTION      NUM_STUDENTS
-----
--csbsju        davis12       29-MAR-24  This group is for CSBSJU students!      5
--MN            maahs35       29-MAR-24  This group is for in-state students!     2
--
--PL/SQL procedure successfully completed.
--
-- COMM_ID      USERNAME      DATE_CREA  C_DESCRIPTION      NUM_STUDENTS
-----
--csbsju        davis12       29-MAR-24  This group is for CSBSJU students!      5
--MN            maahs35       29-MAR-24  This group is for in-state students!     2
--USA           erickson25    29-JAN-24  This group is for USA students!         0
```

Query/Routine 9 (As seen in table of part 5):

```
-- DESCRIPTION: Function to return a student's grade (progress) for a specific course
-- Lectures count for 2 points and tests count for 5

-- CODE
Create or Replace Function view_grade(uname VARCHAR, c_id VARCHAR) Return NUMBER
AS
    courseTotal int := 0;
    studentTotal int := 0;
    courseLectures int := 0;
    courseTests int := 0;
    studentLectures int := 0;
    studentTests int := 0;
Begin
    Select Count(*) * 2 into courseLectures from TARP_Lecture where course_id = c_id;

    Select Count(*) * 2 into studentLectures from TARP_WATCHES
    where uname = username and course_id = c_id and completed = 'y';

    Select Count(*) * 5 into courseTests from TARP_TEST where course_id = c_id;
    Select Count(*) * 5 into studentTests from TARP_TAKEN_BY

    where uname = username and course_id = c_id;
    courseTotal := courseLectures + courseTests;

    studentTotal := studentLectures + studentTests;
    Return studentTotal/courseTotal * 100;
End ;

-- TEST VIEW_GRADE (9)

Select VIEW_GRADE('grove5', 'MATH101') from dual;

-- OUTPUT
-- VIEW_GRADE('GROVE5', 'MATH101')
--
-- 85.7142857
```


Query/Routine 10 (As seen in table of part 5):

```
1  --DESCRIPTION: A view to see all comments made by both students and instructors
2
3  --CODE
4  CREATE OR REPLACE VIEW view_comments AS
5  SELECT course_id, s_username, s_comment, cdate
6  FROM tarp_s_comment
7  UNION
8  SELECT course_id, i_username AS s_username, i_comment AS s_comment, cdate
9  FROM tarp_i_comment
10 ORDER BY cdate;
11
```

-- TEST VIEW_COMMENTS (10)

```
SELECT *
FROM view_comments
WHERE course_id = 'CSCI101';
```

| --COURSE_ID | S_USERNAME | S_COMMENT |
|-------------|------------|-----------------|
| --CSCI101 | imad2000 | Cool course! |
| --CSCI101 | grove5 | Great course! |
| --CSCI101 | erickson25 | Awesome course! |

Query/Routine 11 (As seen in table of part 5):

```
CREATE OR REPLACE PROCEDURE watch_lecture(
    p_lecture_id VARCHAR,
    p_course_id VARCHAR,
    p_username VARCHAR
)
AS
BEGIN
    UPDATE tarp_watches
    SET completed = 'y'
    WHERE lecture_id = p_lecture_id
        AND course_id = p_course_id
        AND username = p_username;
END;
```

```
--TEST watch_lecture (11)
SELECT * FROM tarp_watches WHERE username = 'maahs35' and lecture_id = 'Lecture 2' and course_id = 'MATH101';
EXEC watch_lecture('Lecture 2', 'MATH101', 'maahs35');
SELECT * FROM tarp_watches WHERE username = 'maahs35' and lecture_id = 'Lecture 2' and course_id = 'MATH101';

--OUTPUT
--
--LECTURE_ID          COURSE_ID          USERNAME          C
-----
--Lecture 2          MATH101          maahs35          n
--
--
--PL/SQL procedure successfully completed.
--
--LECTURE_ID          COURSE_ID          USERNAME          C
-----
--Lecture 2          MATH101          maahs35          y
```

Query/Routine 12 (As seen in table of part 5):

```
1  --DESCRIPTION: Procedure to add an instructor's comment to a course given the course_id,
2  --the username of the instructor, and the comment to be added.
3
4  --CODE
5  CREATE OR REPLACE PROCEDURE Add_Comment_I(p_course_id VARCHAR, p_username VARCHAR, p_comment VARCHAR)
6  AS
7  BEGIN
8      INSERT INTO tarp_i_comment (course_id, i_username, i_comment, cdate)
9      VALUES (p_course_id, p_username, p_comment, SYSDATE);
10 END;
11

-- TEST ADD_COMMENT_I (12)

SELECT * FROM tarp_i_comment;
EXEC add_comment_i('CSCI101', 'imad2000', 'Keep up the good work!');
SELECT * FROM tarp_i_comment;

-- OUTPUT
--COURSE_ID          I_USERNAME          I_COMMENT
-----
--CSCI101          imad2000          Cool course!
--MATH101          bill34          Good job everyone!!
--
--
--PL/SQL procedure successfully completed.
--
--COURSE_ID          I_USERNAME          I_COMMENT
-----
--CSCI101          imad2000          Cool course!
--MATH101          bill34          Good job everyone!!
--CSCI101          imad2000          Keep up the good work!
```

Query/Routine 13 (As seen in table of part 5):

```
1 --DESCRIPTION: Function returning 1 if correct answer and 0 if incorrect given the test prompt,
2 --the test_id, course_id, and the submitted answer
3
4 --CODE
5 Create or Replace Function CORRECT_ANSWER(f_prompt varchar,
6 f_test_id varchar, f_course_id varchar, submission varchar) Return int
7 AS
8   c_answer varchar(30);
9   toReturn int := 0;
10  Begin
11    Select answer into c_answer from TARP_QUESTION
12    where f_prompt = t_prompt and f_test_id = test_id and f_course_id = course_id;
13    If (c_answer = submission) Then
14      toReturn := 1;
15    End If;
16    Return toReturn;
17  End ;

-- TEST CORRECT_ANSWER (13)

Select CORRECT_ANSWER('What is 3 + 1?', 'test 1', 'MATH101', 'B') from dual;
Select CORRECT_ANSWER('What is 3 + 1?', 'test 1', 'MATH101', 'C') from dual;

--OUTPUT
--CORRECT_ANSWER('WHATIS3+1?', 'TEST1', 'MATH101', 'B')
-----
--
--
--
--CORRECT_ANSWER('WHATIS3+1?', 'TEST1', 'MATH101', 'C')
-----
--
--
```

Query/Routine 14 (As seen in table of part 5):

```
1 --DESCRIPTION: A function to determine if a user successfully logged in given username
2 --and password. Returns 1 if successful, 0 otherwise.
3
4 --CODE
5 Create or Replace Function Login(p_username varchar, p_password varchar) Return int
6 AS
7   toReturn int := 0;
8   i_exists int := 0;
9   s_exists int := 0;
10  Begin
11    Select Count(*) into i_exists from TARP_INSTRUCTOR T
12    where T.username = p_username and T.password = p_password;
13
14    Select Count(*) into s_exists from TARP_STUDENT T
15    where T.username = p_username and T.password = p_password;
16
17    If (i_exists = 1 OR s_exists = 1) Then
18      toReturn := 1;
19    End If;
20    Return toReturn;
21  End ;
```

```
-- TEST LOGIN (14)

Select Login('grove5', 'password4') from dual;
Select Login('groove5', 'password4') from dual;
Select Login('grove5', 'password3') from dual;

-- OUTPUT
--LOGIN('GROVE5', 'PASSWORD4')
-----
--                               1
--
--
--LOGIN('GROOVE5', 'PASSWORD4')
-----
--                               0
--
--
--LOGIN('GROVE5', 'PASSWORD3')
-----
--                               0
```

Query/Routine 16 (As seen in table of part 5):

-- DESCRIPTION: This procedure creates a new user using the provided parameters

```
-- CODE
CREATE OR REPLACE PROCEDURE create_new_user(uname VARCHAR, f VARCHAR, l VARCHAR, pass VARCHAR, typeUser VARCHAR)
AS
BEGIN
    IF typeUser = 'S' THEN
        INSERT INTO tarp_student (username, f_name, l_name, password)
        VALUES (uname, f, l, pass);
    ELSE
        INSERT INTO tarp_instructor (username, f_name, l_name, password)
        VALUES (uname, f, l, pass);
    END IF;
END;
```

```
-- TEST CREATE_NEW_USER (16)
```

```
SELECT * FROM tarp_student;
```

```
EXEC create_new_user('kerry8','Kerry', 'Maahs', 'kerryPass', 'S');
```

```
SELECT * FROM tarp_student;
```

```
-- OUTPUT
```

| --USERNAME | F_NAME | L_NAME | PASSWORD |
|--------------|--------|----------|-----------|
| --erickson25 | Rodney | Erickson | password1 |
| --maahs35 | Keiley | Maahs | password2 |
| --davis12 | Carson | Davis | password3 |
| --grove5 | Carter | Grove | password4 |
| --bartness2 | Dylan | Bartness | password5 |

```
--
```

```
--
```

```
--PL/SQL procedure successfully completed.
```

```
--
```

```
--
```

| --USERNAME | F_NAME | L_NAME | PASSWORD |
|--------------|--------|----------|-----------|
| --erickson25 | Rodney | Erickson | password1 |
| --maahs35 | Keiley | Maahs | password2 |
| --davis12 | Carson | Davis | password3 |
| --grove5 | Carter | Grove | password4 |
| --bartness2 | Dylan | Bartness | password5 |
| --kerry8 | Kerry | Maahs | kerryPass |

Query/Routine 19 (As seen in table of part 5):

```
--DESCRIPTION: Procedure to count how many lectures a student has completed
```

```
-- CODE
```

```
Create or Replace Function count_student_completed_lectures(uname VARCHAR) Return int  
AS
```

```
    numLectures int := 0;
```

```
    Begin
```

```
        Select Count(*) into numLectures from TARP_WATCHES T
```

```
        where uname = T.username and T.completed = 'y';
```

```
        Return numLectures;
```

```
    End ;
```

```
-- TEST COUNT_STUDENT_COMPLETED_LECTURES (19)

Select Count(*) from TARP_WATCHES T where 'grove5' = T.username and T.completed = 'y';
SELECT count_student_completed_lectures('grove5') from DUal;
Select Count(*) from TARP_WATCHES T where 'bartness2' = T.username and T.completed = 'y';
SELECT count_student_completed_lectures('bartness2') from DUal;

--
-- COUNT(*)
-- -----
--          1
--
--
-- COUNT_STUDENT_COMPLETED_LECTURES(' GROVES' )
-- -----
--                               1
--
--
-- COUNT(*)
-- -----
--          2
--
--
-- COUNT_STUDENT_COMPLETED_LECTURES(' BARTNESS2' )
-- -----
--                               2
```

Query/Routine 20 (As seen in table of part 5):

```
CREATE OR REPLACE PROCEDURE rate_instructor(
    p_course_id VARCHAR,
    p_student_username VARCHAR,
    p_rating INT
)
AS
    v_instructor_username VARCHAR(20);
BEGIN
    SELECT username INTO v_instructor_username
    FROM tarp_course
    WHERE course_id = p_course_id;

    INSERT INTO tarp_rates (s_username, i_username, instructor_score)
    VALUES (p_student_username, v_instructor_username, p_rating);

END RATE_INSTRUCTOR;
```

```
--TEST rate_instructor (20)
SELECT * FROM tarp_rates WHERE I_username = 'bill34';
EXEC rate_instructor('MATH101', 'grove5', 8);
SELECT * FROM tarp_rates WHERE I_username = 'bill34';

--OUTPUT
--
--S_USERNAME      I_USERNAME      INSTRUCTOR_SCORE
-----
--davis12         bill34          4
--erickson25      bill34          7
--maahs35         bill34          10
--
--
--PL/SQL procedure successfully completed.
--
--S_USERNAME      I_USERNAME      INSTRUCTOR_SCORE
-----
--davis12         bill34          4
--erickson25      bill34          7
--grove5          bill34          8
--maahs35         bill34          10
```

(5) Data Processing:

Query/Routine 7 (As seen in table of part 5):

```
-- TEST view_score

-- DESCRIPTION: View the score for a student

-- CODE

SELECT
    e.username,
    (count_student_completed_lectures(e.username) * 10)
    + sum_student_tests(e.username) AS total_score
FROM
    tarp_enrolls e
INNER JOIN
    tarp_course c ON e.course_id = c.course_id
WHERE
    e.username = 'grove5';

--OUTPUT

--USERNAME          TOTAL_SCORE
-----
--grove5              180
```

Query/Routine 8 (As seen in table of part 5):

```
-- TEST view_courses_taking

-- DESCRIPTION: View the courses a student is enrolled in

-- CODE

SELECT course_id
FROM tarp_enrolls
WHERE username = 'erickson25';

--OUTPUT

--COURSE_ID
-----
--MATH101
```


Query/Routine 15 (As seen in table of part 5):

```
-- TEST VIEW_COURSES TAUGHT (15)

-- DESCRIPTION: View courses that an instructor teaches

-- CODE

SELECT course_id
FROM TARP_COURSE tp
WHERE tp.username = 'bill34';

-- OUTPUT
-- COURSE_ID
-----
-- MATH101
-- MATH231
```

Query/Routine 17 (As seen in table of part 5):

```
-- TEST list_lectures

-- DESCRIPTION: View the lectures for a student based on a course

-- CODE

SELECT lecture_id, length, url
FROM tarp_lecture
WHERE course_id IN (
  SELECT course_id
  FROM tarp_enrolls
  WHERE username = 'erickson25' and course_id = 'MATH101'
);

-- OUTPUT
-- LECTURE_ID          LENGTH URL
-----
-- Lecture 1           25 https://example.com/lecture1/MATH101
-- Lecture 2           15 https://example.com/lecture2/MATH101
```

Query/Routine 18 (As seen in table of part 5):

```
-- TEST VIEW_ALL_GRADES (18)

-- DESCRIPTION: View the grades (progress percentage)
-- for each course a specific student is taking.

-- CODE

SELECT COURSE_ID, VIEW_GRADE('grove5', TE.course_id) AS GRADE
FROM TARP_ENROLLS TE
WHERE username = 'grove5';

-- COURSE_ID          GRADE
-----
-- MATH101             85.7142857
-- THE0111             41.6666667
```

Table of Query/Routines:

| | Query/Routine Name | Brief Description | |
|----|---------------------------------------|---|--------|
| 1 | COMMUNITY INSERT DELETE TRIGGER | Trigger to update number of students in community table when a student joins/leaves. | RE |
| 2 | CREATE LECTURE (Procedure) | Procedure to create a lecture for a course. | CG |
| 3 | DELETE COURSE (Procedure) | Procedure to delete a course. | DB |
| 4 | ENROLL IN COURSE (Procedure) | Procedure for a student to enroll in a course. | DB |
| 5 | MODIFY STUDENT PROFILE (Procedure) | Procedure to update the password of a student. | CD |
| | MODIFY INSTRUCTOR PROFILE (Procedure) | Procedure to update the password of an instructor. | - |
| 6 | CREATE COMMUNITY (Procedure) | Procedure to create a new community. | CD |
| - | DELETE COMMUNITY (Procedure) | Procedure to delete a community. | - |
| - | JOIN COMMUNITY (Procedure) | Procedure to add user to community. | - |
| 7 | VIEW SCORE (Statement) | View to see the current ranking of a student in a community. | DB |
| 8 | VIEW COURSES TAKING (Statement) | View to see the courses that a student is taking. | CD |
| 9 | VIEW GRADE (Function) | Students can view their current grade for a specific course they are enrolled in to see their progress. | K M |
| 10 | VIEW COMMENTS (View) | View comments for a course ordered by timestamp. | RE |
| 11 | WATCH LECTURE (Procedure) | Statement that marks lectures as watched. | CD |
| 12 | ADD COMMENT I (Procedure) | Procedure for an instructor commenting on a course. | CG |
| - | ADD COMMENT S (Procedure) | Procedure for a student commenting on a course. | - |
| 13 | CORRECT ANSWER (Function) | Function returns 1 if a student correctly answered a test question and 0 otherwise. | CG |
| 14 | LOGIN (Function) | Function to return 1 if a user entered a valid username/password login. | CG |
| 15 | VIEW COURSES TAUGHT (Statement) | View the courses that an instructor is teaching. | K M |
| 16 | CREATE NEW USER (Procedure) | A person can create an account as either a student or instructor. | DB |
| 17 | LIST LECTURES (Statement) | A student can click a button to list all the lectures in a course they are enrolled in | RE |

| | | | |
|----|---|---|--------|
| 18 | VIEW_ALL_GRADES (Statement) | A student can view the grades of all courses they are enrolled in | K M |
| 19 | COUNT_STUDENT_COMPLETED_LECTURES (Function) | Count number of lectures completed by a student. | K M |
| - | SUM_STUDENT_TESTS (Function) | Sum the total points from all tests a student has completed. | - |
| 20 | RATE_INSTRUCTOR (Procedure) | This procedure allows a student to rate an instruct with a score of 0-10. | RE |

(6) Sample User Interfaces

1)

The Best Community on Tarp

This is a great community where any student is invited to compete with one another to see who is the best student.

123 Members
Your Rank: 2nd

[Invite Another Student](#) [Leave Community](#)

Leaderboard

1. userHere
2. **You**
3. anotheruser
4. someoneElse
5. person

2)

Name

URL

[Confirm](#)

3)

Terrible Course

★★★★☆ (2.1)

12 students

[View](#) [Edit](#) [QBoard](#) [Delete](#)

4)

CSCI101


[Overview](#) [Lectures](#) [Tests](#) [Q Board](#)

[Enroll](#)

Professor Prof ★★★★★ (3.2)

Description
This is a very good courses. You should enroll.

5)



Your Profile

View or update your account information

Username

aUser

First Name

John


Last Name

Doe

Email

john.doe001@csbsju.edu

Password

.....

6)

Create a Course

Name

CSCI123

Description

This is a course in which you can learn about computers.

Create

7)

The Best Community on Tarp

This is a great community where any student is invited to compete with one another to see who is the best student.

123 Members
Your Rank: 2nd

Invite Another Student

Leave Community

Leaderboard

- userHere
- You**
- anotheruser
- sopmeoneElse
- person

8)

Your Enrolled Courses

CSCI 321
by Bruce the Prof
Progress: 50%

[View](#) [QBoard](#)

HIST 111
by Doe Johnson
Progress: 30%

[View](#) [QBoard](#)

CSCI 355
by John Doeson
Progress: 80%

[View](#) [QBoard](#)

9)

CSCI 123 - Grades

[Back to Course](#)

| | |
|--------------------------|------------|
| Total: | 92% |
| Intro to Data Types Exam | 90% |
| Data Types Part 2 Exam | 98% |
| Unit 3 Exam | 100% |

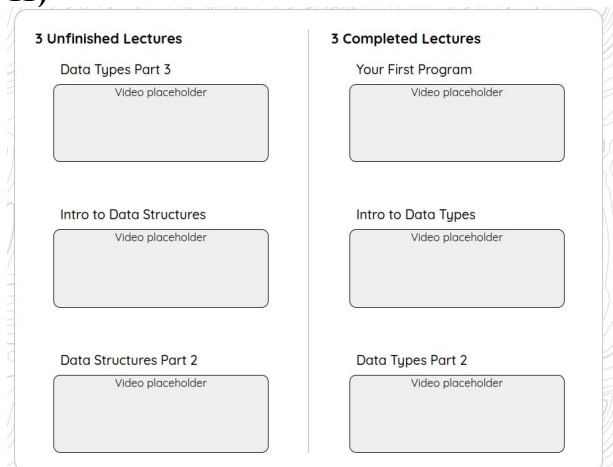
10)

Bruce The Prof (instructor) 01/22/24 at 2:30pm
Thanks for all of the feedback everyone!

userHere 01/21/24 at 1:23pm
Great course.

anotherUser 01/21/24 at 1:20pm
I learned a lot of usefull stuff.

11)



12)

comment

Comment

13)

The question prompt could go here.

- + ☒ Option A
- ☐ Option B
- ☐ Option C
- ☐ Option D

Another question prompt could possibly go here.

- ☐ Option A
- ☒ Option B
- ☐ Option C
- + ☐ Option D

14)

Welcome Back!

Email

email

Password

password

Login

15)

John Doeson

★★★★ (4.5)

Rate

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Submit

Courses

CSCI 123

★★★★ (4.5)

This is a really cool and good course.

CSCI 101

★★ (2.3)

Learn about cool computer stuff.

CSCI 321

★★★★★ (5.0)

Data stuff.

16)

Let's Get Started!

Full Name

John

Username

Doe

Email

john.doe001@csbsju.edu

Password



Sign Up

17)

3 Unfinished Lectures

Data Types Part 3

Video placeholder

Intro to Data Structures

Video placeholder

Data Structures Part 2

Video placeholder

3 Completed Lectures

Your First Program

Video placeholder

Intro to Data Types

Video placeholder

Data Types Part 2

Video placeholder

18)

Cumulative Grade Report

| | |
|-------------|--------------|
| GPA: | 3.923 |
| CSCI 123 | 92% |
| HIST 111 | 98% |
| CSCI 125 | 85% |

19)

John Doeson

★★★★ (4.5)

Rate

0 1 2 3 4 5 Submit

Courses

CSCI 123

★★★★ (4.5)

This is a really cool and good course.

CSCI 101

★★ (2.3)

Learn about cool computer stuff.

CSCI 321

★★★★★ (5.0)

Data stuff.

20)

John Doeson

★★★★ (4.5)

Rate

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Submit