Assignment 3: DBMS

MIS No.: 612303083 (TY 1, ADS 3)

Q.1 Write the DDL and DML statements for the following. -

DDL - Solution queries:

• Code:

```
-- Each offering of a course (i.e. a section) can have many Teaching assistants; each teaching assistant is a student. Extend the existing schema(Add/Alter tables) to accommodate this requirement.

CREATE TABLE TA (

ID VARCHAR(5),

course_id VARCHAR(8),

sec_id VARCHAR(8),

semester VARCHAR(6) CHECK (semester IN ('Fall' , 'Winter', 'Spring', 'Summer')),

year NUMERIC(4 , 0 ) CHECK (year > 1701 AND year < 2100),

FOREIGN KEY (ID)

REFERENCES student (ID),

FOREIGN KEY (course_id , sec_id , semester , year)

REFERENCES section (course_id , sec_id , semester , year)
);
```

• Code:

```
-- According to the existing schema, one student can have only one advisor.
-- Alter the schema to allow a student to have multiple advisors and make sure that you are able to insert multiple advisors for a student.

DROP TABLE advisor;

CREATE TABLE advisor (
    s_ID VARCHAR(5),
    i_ID VARCHAR(5),
    PRIMARY KEY (s_ID, i_ID),
    FOREIGN KEY (i_ID) REFERENCES instructor(ID),
    FOREIGN KEY (s_ID) REFERENCES student(ID)
);
```

Q.2 Write SQL queries on the modified schema. You will need to insert data to ensure the query results are not empty -

DML and DQL - Solution queries:

• Code:

```
insert into instructor values ('30605', 'Ashok', 'Comp. Sci.', '75000');
insert into advisor values ('00128', '45565');
insert into advisor values ('00128', '98345');
insert into advisor values ('00128', '10101');
insert into advisor values ('00128', '76543');
insert into advisor values ('00128', '30605');
insert into advisor values ('12345', '10101');
insert into advisor values ('12345', '45565');
insert into advisor values ('76653', '98345');
insert into advisor values ('76653', '10101');
insert into advisor values ('76653', '22222');
insert into advisor values ('76653', '22222');
insert into advisor values ('76653', '76543');
```

• Code:

```
-- Find all students who have more than 3 advisors

SELECT s.name

FROM student s

JOIN advisor a ON s.ID = a.s_ID

GROUP BY s.name

HAVING COUNT(a.i_ID) > 3;
```

Output:



• Code:

```
-- Find all students who are co-advised by Prof. Srinivas and Prof. Ashok.

SELECT s.name

FROM student s

JOIN advisor a1 ON s.ID = a1.s_ID

JOIN instructor i1 ON a1.i_ID = i1.ID

JOIN advisor a2 ON s.ID = a2.s_ID

JOIN instructor i2 ON a2.i_ID = i2.ID

WHERE i1.name = 'Srinivasan' AND i2.name = 'Ashok';
```

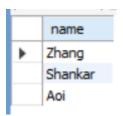
Output:



• Code:

```
-- Find students advised by instructors from different departments. etc.
SELECT DISTINCT s.name
FROM student s
JOIN advisor a1 ON s.ID = a1.s_ID
JOIN instructor i1 ON a1.i_ID = i1.ID
JOIN advisor a2 ON s.ID = a2.s_ID
JOIN instructor i2 ON a2.i_ID = i2.ID
WHERE i1.dept_name <> i2.dept_name;
```

• Output:



Q.3 Write SQL queries for the following -

DDL - Solution queries:

• Code:

```
-- Delete all information in the database which is more than 10 years old. Add data as necessary to verify your query.

DELETE FROM takes

WHERE year < YEAR(CURDATE()) - 10;

DELETE FROM teaches

WHERE year < YEAR(CURDATE()) - 10;

DELETE FROM section

WHERE year < YEAR(CURDATE()) - 10;
```

• Code:

```
-- Delete the course CS 101. Any course which has CS 101 as a prereq should remove CS 101 from its prereq set. Create a cascade constraint to enforce the above rule, and verify that it is working. drop table prereq;
```

```
CREATE TABLE prereq (
    course_id VARCHAR(8),
    prereq_id VARCHAR(8),
    PRIMARY KEY (course_id),
    FOREIGN KEY (course_id) REFERENCES course(course_id)
          ON DELETE CASCADE,
    FOREIGN KEY (prereq_id) REFERENCES course(course_id)
          ON DELETE SET NULL
);

DELETE FROM course WHERE course_id = 'CS101';
```