Week 6: Student-Mentor Project

1. Define your database

   a. the database, its purpose, why it was created, who it was created for, who the users are, and what type of processes and functions it will support.

      Our database includes 8 tables to support Student Mentor Sessions.

   b. List the entities of your database.

      Entities are:

	Entity	Description
1	**Mentor**	**A Mentor is an expert on topic(s)**
2	**T_Session**	**Talk Session / Meeting facilitated by U4S includes Mentor(s) & students**
3	**School**	**School Information where Student is enrolled**
4	**Student**	**Student entity represents any prospective mentee studying in a school.**
5	**Topic**	**Topics are Specific titles of interest of the Students. Ex. Placement in IT Industry**
6	Mentor_Session	Session(s) by Mentor(s)
7	Student_Session	Session(s) signed up by Students
8	Student_Topic	Topics Students are interested in
9	Mentor_Topic	Topics Mentors are expert of/ interested in

Mentor
M_ID (PK)
M_Name
M_Org
Address
City
State
Phone
Email

T_Session
Session_ID (PK)
Topic_ID
Start_Time
End_Time
Status
Venue
Delivery_Platform

School
School_ID (PK)
Sch_Name
Sch_Address
Sch_City
Sch_State
Sch_Phone
Sch_Email

Student
Student_ID (PK)
FirstName
LastName
Address
City
State
Phone
Email
School_ID (FK)

Mentor_Session
M_ID
Session_ID

Student_Session
Session_ID
Student_ID

Topic
Topic_ID
Primary_Subject_Area
Sec_Subject_Area
Description

Student_Topic
Student_ID
Topic_ID

Mentor_Topic
M_ID
Topic_ID

c. Present the business rules that determine connectivity, using the format reviewed in Week 2.

Directly-related entity type pairs (Relationships)

Student - School

An Student can be enrolled in one & only one school (Student can be enrolled in one or more colleges but only one of them is mapped in Student table)

School can have zero or many students

Student - Topic

An Student can be interested in at least one and may be many Topics

A Topic can be of interest of one or many students

T_Session - Topic

A Session can cover only one Topic

A Topic can have zero or many sessions

Mentor - Topic

A Mentor can be expert on one or many Topics

A Topic can have one or many Mentors/Experts

Mentor - T_Session

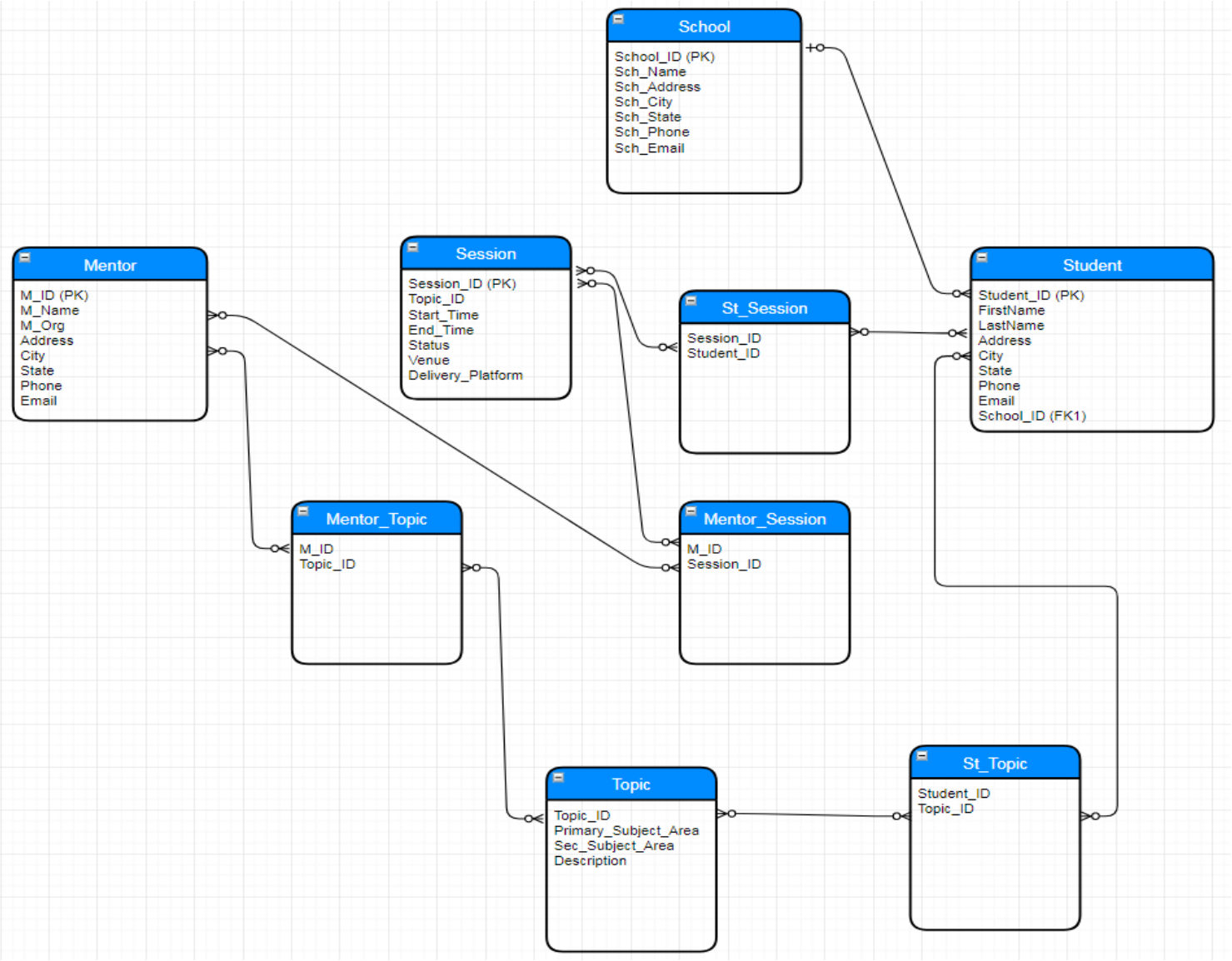
A Mentor can deliver one or more session

A Session can have talk from more than one mentor

Student - T_Session

Student can attend many sessions

d. Draw the ERD; copy the drawing into your Word project document.



2. Create the database and tables for the database. Show all SQL statements. Include primary and foreign keys.

```
CREATE TABLE school
  (
     school_id   INTEGER NOT NULL,
     sch_name    VARCHAR(30),
     sch_address VARCHAR(30),
     sch_city    VARCHAR(30),
     sch_state   VARCHAR(30),
     sch_zip     VARCHAR(10),
     sch_phone   VARCHAR(30),
     sch_email   VARCHAR(30),
     CONSTRAINT pk_school PRIMARY KEY (school_id)
  );

CREATE TABLE student
  (
     student_id INTEGER NOT NULL,
     firstname  VARCHAR(30),
     lastname   VARCHAR(30),
     address    VARCHAR(30),
     city       VARCHAR(30),
     state      VARCHAR(30),
     zip        VARCHAR(10),
     phone      VARCHAR(30),
     email      VARCHAR(30),
     school_id  INTEGER,
     CONSTRAINT pk_student PRIMARY KEY (student_id),
     CONSTRAINT fk_studentschool FOREIGN KEY (school_id) REFERENCES school(
     school_id)
  );

CREATE TABLE mentor
  (
     m_id    INTEGER NOT NULL,
     m_fname VARCHAR(30),
     m_lname VARCHAR(30),
     m_org   VARCHAR(30),
     address VARCHAR(30),
     city    VARCHAR(30),
     state   VARCHAR(30),
     zip     VARCHAR(10),
     phone   VARCHAR(30),
     email   VARCHAR(30),
     CONSTRAINT pk_mentor PRIMARY KEY (m_id)
  );
```

```
CREATE TABLE topic
  (
     topic_id             INTEGER NOT NULL,
     primary_subject_area VARCHAR(30),
     sec_subject_area     VARCHAR(30),
     description          VARCHAR(100),
     CONSTRAINT pk_topic PRIMARY KEY (topic_id)
  );

CREATE TABLE t_session
  (
     session_id INTEGER NOT NULL,
     topic_id   INTEGER NOT NULL,
     start_time TIMESTAMP,
     end_time   TIMESTAMP,
     status     VARCHAR(10),
     venue      VARCHAR(100),
     platform   VARCHAR(30),
     CONSTRAINT pk_session PRIMARY KEY (session_id),
     CONSTRAINT fk_t1 FOREIGN KEY (topic_id) REFERENCES topic(topic_id)
  );

CREATE TABLE st_topic
  (
     student_id INTEGER NOT NULL,
     topic_id   INTEGER NOT NULL,
     CONSTRAINT pk_st_topic PRIMARY KEY (student_id, topic_id),
     CONSTRAINT fk_st1 FOREIGN KEY (student_id) REFERENCES student(student_id),
     CONSTRAINT fk_st2 FOREIGN KEY (topic_id) REFERENCES topic(topic_id)
  );

CREATE TABLE mentor_topic
  (
     m_id     INTEGER NOT NULL,
     topic_id INTEGER NOT NULL,
     CONSTRAINT pk_m_topic PRIMARY KEY (m_id, topic_id),
     CONSTRAINT fk_mt1 FOREIGN KEY (m_id) REFERENCES mentor(m_id),
     CONSTRAINT fk_mt2 FOREIGN KEY (topic_id) REFERENCES topic(topic_id)
  );

CREATE TABLE mentor_session
  (
     m_id       INTEGER NOT NULL,
     session_id INTEGER NOT NULL,
     CONSTRAINT pk_m_session PRIMARY KEY (m_id, session_id),
     CONSTRAINT fk_ms1 FOREIGN KEY (m_id) REFERENCES mentor(m_id),
     CONSTRAINT fk_ms2 FOREIGN KEY (session_id) REFERENCES t_session(session_id)
  );

CREATE TABLE student_session
```

```
(
   student_id INTEGER NOT NULL,
   session_id INTEGER NOT NULL,
   CONSTRAINT pk_st_session PRIMARY KEY (student_id, session_id),
   CONSTRAINT fk_ss1 FOREIGN KEY (student_id) REFERENCES student(student_id),
   CONSTRAINT fk_ss2 FOREIGN KEY (session_id) REFERENCES t_session(session_id)
);
```

3. Insert data into each table. Show select statements and display the output of each table. **Note:** Student's name must be inserted into table as part of the data!

```
INSERT INTO school
VALUES      (1,
             'WALDEN UNIVERSITY',
             '100 S Washington Ave #900',
             'Minneapolis',
             'MN',
             '55401',
             '(866) 492-5336',
             'help@waldenu.edu');

INSERT INTO student
VALUES     (1001,
            'PAYAL',
            'MOHNANI',
            '123 Main St',
            'Salt lake City',
            'UT',
            '84101',
            '8019019911',
            'payal.mohnani@waldenu.edu',
            1);

INSERT INTO student
VALUES     (1002,
            'CHRISTINA',
            'ARMSTRONG',
            '234 State St',
            'Salt lake City',
            'UT',
            '84101',
            '8019019922',
            'chrissy1@waldenu.edu',
            1);

INSERT INTO mentor
VALUES     (9001,
            'BILL',
```

```
            'GATES',
            'Bill Foundation',
            '123 Main St',
            'Seattle',
            'WA',
            '98101',
            '2220001111',
            'bill@bill.com');

INSERT INTO mentor
VALUES     (9002,
            'LINUS',
            'TORVALDS',
            'Linux Foundation',
            '777 Linux St',
            'Seattle',
            'WA',
            '98102',
            '3330001111',
            'linus@linus.com');

INSERT INTO mentor
VALUES     (9003,
            'LARRY',
            'ELLISSON',
            'Oracle Corporation',
            '1 Oracle Way',
            'Redwood shores',
            'CA',
            '94086',
            '4084442211',
            'larry@larry.com');

INSERT INTO mentor
VALUES     (9004,
            'RICHARD',
            'STALLMAN',
            'Linux Foundation',
            '777 Linux St',
            'Seattle',
            'WA',
            '98102',
            '3330001111',
            'rich@rich.com');
```

```
INSERT INTO mentor
VALUES     (9005,
            'TERRY',
            'LABRUM',
            'Open Source Foundation',
            '222 Main St',
            'Salt Lake City',
            'UT',
            '84121',
            '8018018888',
            'terry@terry.com');


INSERT INTO topic
VALUES     (11,
            'Computer Science',
            'Android OS',
            'Linux Kernel based operating system with JAVA API');

INSERT INTO topic
VALUES     (12,
            'Computer Science',
            'Apple iOS',
            'Apple Proprietory OS with C API');

INSERT INTO topic
VALUES     (13,
            'Computer Science',
            'Exadata Database',
            'Oracle Proprietory database hardware');

INSERT INTO topic
VALUES     (14,
            'Career Dev & Placement',
            'Startup Placement',
            'N/A');

INSERT INTO topic
VALUES     (15,
            'Career Dev & Placement',
            'First Job as Intern',
            'N/A');

INSERT INTO t_session
VALUES     (51,
            11,
            '25-OCT-2019 1:00:00.00 PM',
            '25-OCT-2019 3:00:00.00 PM',
            'PLANNED',
            'ONLINE',
```

```
            'GOTOMEETING');

INSERT INTO t_session
VALUES     (52,
            11,
            '26-OCT-2019 1:00:00.00 PM',
            '26-OCT-2019 3:00:00.00 PM',
            'PLANNED',
            'ONLINE',
            'GOTOMEETING');

INSERT INTO t_session
VALUES     (53,
            13,
            '26-OCT-2019 3:30:00.00 PM',
            '26-OCT-2019 5:30:00.00 PM',
            'PLANNED',
            'ONLINE',
            'GOTOMEETING');

INSERT INTO st_topic VALUES     (1001,             11);

INSERT INTO st_topic VALUES     (1001,             12);

INSERT INTO st_topic VALUES     (1001,             13);

INSERT INTO st_topic VALUES     (1002,             11);

INSERT INTO st_topic VALUES     (1002,             13);

INSERT INTO st_topic VALUES     (1002,             15);

INSERT INTO mentor_topic VALUES     (9001,             11);

INSERT INTO mentor_topic VALUES     (9002,             11);

INSERT INTO mentor_topic VALUES     (9003,             13);

INSERT INTO mentor_session VALUES     (9001,             51);

INSERT INTO mentor_session VALUES     (9002,             52);

INSERT INTO mentor_session VALUES     (9004,             52);

INSERT INTO mentor_session VALUES     (9003,             53);

INSERT INTO student_session VALUES      (1001,             51);

INSERT INTO student_session VALUES      (1001,             52);
```

```
INSERT INTO student_session VALUES      (1001,              53);

INSERT INTO student_session VALUES      (1002,              51);

INSERT INTO student_session VALUES      (1002,              52);

INSERT INTO student_session VALUES      (1002,              53);
```

4. Perform the SQL below:

   a. . Query one table and use WHERE to filter the results. The SELECT clause should have a column list, not an asterisk (*). State the purpose of the query; show the query and the output.

Query: Below query filters all the mentors who are not from UTAH.

```
SELECT m_id,
       m_fname,
       m_lname,
       m_org,
       address,
       city,
       state,
       zip,
       phone,
       email
FROM   mentor;
```

Output

M_ID	M_FNAME	M_LNAME	M_ORG	ADDRESS	CITY	STATE	ZIP	PHONE	EMAIL
9001	BILL	GATES	Bill Foundation	123 Main St	Seattle	WA	98101	2220001111	bill@bill.com
9002	LINUS	TORVALDS	Linux Foundation	777 Linux St	Seattle	WA	98102	3330001111	linus@linus.com
9003	LARRY	ELLISSON	Oracle Corporation	1 Oracle Way	Redwood shores	CA	94086	4084442211	larry@larry.com
9004	RICHARD	STALLMAN	Linux Foundation	777 Linux St	Seattle	WA	98102	3330001111	rich@rich.com

M_ID	M_FNAME	M_LNAME	M_ORG	ADDRESS	CITY	STATE	ZIP	PHONE	EMAIL
9001	BILL	GATES	Bill Foundation	123 Main St	Seattle	WA	98101	2220001111	bill@bill.com
9002	LINUS	TORVALDS	Linux Foundation	777 Linux St	Seattle	WA	98102	3330001111	linus@linus.com
9003	LARRY	ELLISSON	Oracle Corporation	1 Oracle Way	Redwood shores	CA	94086	4084442211	larry@larry.com
9004	RICHARD	STALLMAN	Linux Foundation	777 Linux St	Seattle	WA	98102	3330001111	rich@rich.com

   b. Get information from at least **3** tables in one statement, and provide the output using the Join operator. Use ANSI Join syntax. State the purpose of the query; show the query and the output. Add a screen shot of SS Management Studio showing the query and results.

Query : Below query joins 3 or more tables and fetches Information (Id, First Name & Last Name) of Students, the Session date/time attended by students and the Mentor names giving Talk in those sessions.

```
SELECT student.student_id,
       student.firstname,
```

```
        student.lastname,
        t_session.start_time,
        t_session.end_time,

        mentor.m_fname,
        mentor.m_lname
FROM    student
        join student_session
          ON student.student_id = student_session.student_id
        join t_session
          ON t_session.session_id = student_session.session_id
        join mentor_session
          ON mentor_session.session_id = student_session.session_id
        join mentor
          ON mentor_session.m_id = mentor.m_id
```

Output

STUDENT_ID	FIRSTNAME	LASTNAME	START_TIME	END_TIME	M_FNAME	M_LNAME
1001	PAYAL	MOHNANI	25-OCT-19 01.00.00.000000 PM	25-OCT-19 03.00.00.000000 PM	BILL	GATES
1002	CHRISTINA	ARMSTRONG	25-OCT-19 01.00.00.000000 PM	25-OCT-19 03.00.00.000000 PM	BILL	GATES
1001	PAYAL	MOHNANI	26-OCT-19 01.00.00.000000 PM	26-OCT-19 03.00.00.000000 PM	LINUS	TORVALDS
1002	CHRISTINA	ARMSTRONG	26-OCT-19 01.00.00.000000 PM	26-OCT-19 03.00.00.000000 PM	LINUS	TORVALDS
1001	PAYAL	MOHNANI	26-OCT-19 01.00.00.000000 PM	26-OCT-19 03.00.00.000000 PM	RICHARD	STALLMAN
1002	CHRISTINA	ARMSTRONG	26-OCT-19 01.00.00.000000 PM	26-OCT-19 03.00.00.000000 PM	RICHARD	STALLMAN
1001	PAYAL	MOHNANI	26-OCT-19 03.30.00.000000 PM	26-OCT-19 05.30.00.000000 PM	LARRY	ELLISSON
1002	CHRISTINA	ARMSTRONG	26-OCT-19 03.30.00.000000 PM	26-OCT-19 05.30.00.000000 PM	LARRY	ELLISSON

c. Get information from **2** tables in one statement, and provide the output using the Left Outer Join operator. State the purpose of the query; show the query and the output. The outer join should be designed to retrieve information from the left table that has no matches in the right table. If that is not possible for your database, explain why.

Query:

This query lists all mentors and their sessions using left join , if they have no session ,then Session id shows as null due to left join.

```
SELECT mentor.*,
       mentor_session.session_id
FROM   mentor
       left join mentor_session
              ON mentor.m_id = mentor_session.m_id
```

M_ID	M_FNAME	M_LNAME	M_ORG	ADDRESS	CITY	STATE	ZIP	PHONE	EMAIL	SESSION_ID
9001	BILL	GATES	Bill Foundation	123 Main St	Seattle	WA	98101	2220001111	bill@bill.com	51
9002	LINUS	TORVALDS	Linux Foundation	777 Linux St	Seattle	WA	98102	3330001111	linus@linus.com	52
9003	LARRY	ELLISSON	Oracle Corporation	1 Oracle Way	Redwood shores	CA	94086	4084442211	larry@larry.com	53
9004	RICHARD	STALLMAN	Linux Foundation	777 Linux St	Seattle	WA	98102	3330001111	rich@rich.com	52
9005	TERRY	LABRUM	Open Source Foundation	222 Main St	Salt Lake City	UT	84121	8018018888	terry@terry.com	-

M_ID	M_FNAME	M_LNAME	M_ORG	ADDRESS	CITY	STATE	ZIP	PHONE	EMAIL	SESSION_ID
9001	BILL	GATES	Bill Foundation	123 Main St	Seattle	WA	98101	2220001111	bill@bill.com	51
9002	LINUS	TORVALDS	Linux Foundation	777 Linux St	Seattle	WA	98102	3330001111	linus@linus.com	52
9003	LARRY	ELLISSON	Oracle Corporation	1 Oracle Way	Redwood shores	CA	94086	4084442211	larry@larry.com	53
9004	RICHARD	STALLMAN	Linux Foundation	777 Linux St	Seattle	WA	98102	3330001111	rich@rich.com	52
9005	TERRY	LABRUM	Open Source Foundation	222 Main St	Salt Lake City	UT	84121	8018018888	terry@terry.com	-

d. Create a query using the IN keyword with a sub query. State the purpose of the query; show the query and the output.

Query: This query fetches all the mentors who have session in next 12 days from today (Oct13)

```
SELECT m_id,
       m_fname,
       m_lname,
       m_org,
       address,
       city,
       state,
       zip,
       phone,
       email
FROM   mentor
WHERE  m_id IN (SELECT mentor_session.m_id
                FROM   mentor_session
                       join t_session
                ON mentor_session.session_id = t_session.session_id
                WHERE  t_session.start_time <= (SELECT SYSDATE + 12
                                                FROM   dual));
```

M_ID	M_FNAME	M_LNAME	M_ORG	ADDRESS	CITY	STATE	ZIP	PHONE	EMAIL
9001	BILL	GATES	Bill Foundation	123 Main St	Seattle	WA	98101	2220001111	bill@bill.com

e. Create a query using an aggregate function (i.e., min, max, avg, sum, count) and the GROUP BY command. State the purpose of the query; show the query and the output.

Query: This Query gives counts of Sessions signed up by each student ID

```
SELECT student_id,
       Count(session_id) Sessions_Signed
FROM   student_session
GROUP  BY student_id;
```

output

STUDENT_ID	Sessions_Signed
1001	3
1002	3

f. Create a query using an aggregate function (i.e., min, max, avg, sum, count) and the GROUP BY command using the HAVING clause to filter the aggregate results. State the purpose of the query; show the query and the output.

Query: It identifies if there is any mentor without any Session scheduled

```
SELECT mentor.*,
       Count(mentor_session.session_id)
FROM   mentor
       left join mentor_session
              ON mentor.m_id = mentor_session.m_id
GROUP  BY mentor.m_id,
          mentor.m_fname,
          mentor.m_lname,
          mentor.m_org,
          mentor.address,
          mentor.city,
          mentor.state,
          mentor.zip,
          mentor.phone,
          mentor.email
HAVING Count(mentor_session.session_id) = 0;
```

M_ID	M_FNAME	M_LNAME	M_ORG	ADDRESS	CITY	STATE	ZIP	PHONE	EMAIL	COUNT(MENTOR_SESSION.SESSION_ID)
9005	TERRY	LABRUM	Open Source Foundation	222 Main St	Salt Lake City	UT	84121	8018018888	terry@terry.com	0

g. Update one row. State the purpose of the query; show the result set for the row(s) before the update; show the query; show the row(s) after the update.

Query to Update: This update statement modifies tech platform for Session Id 53

```
UPDATE t_session
SET    platform = 'SKYPE'
WHERE  session_id = 53;
```

Before:

```
SELECT * FROM   t_session;
```

SESSION_ID	TOPIC_ID	START_TIME	END_TIME	STATUS	VENUE	PLATFORM
51	11	25-OCT-19 01.00.00.000000 PM	25-OCT-19 03.00.00.000000 PM	PLANNED	ONLINE	GOTOMEETING
52	11	26-OCT-19 01.00.00.000000 PM	26-OCT-19 03.00.00.000000 PM	PLANNED	ONLINE	GOTOMEETING
53	13	26-OCT-19 03.30.00.000000 PM	26-OCT-19 05.30.00.000000 PM	PLANNED	ONLINE	GOTOMEETING

After:

SESSION_ID	TOPIC_ID	START_TIME	END_TIME	STATUS	VENUE	PLATFORM
51	11	25-OCT-19 01.00.00.000000 PM	25-OCT-19 03.00.00.000000 PM	PLANNED	ONLINE	GOTOMEETING
52	11	26-OCT-19 01.00.00.000000 PM	26-OCT-19 03.00.00.000000 PM	PLANNED	ONLINE	GOTOMEETING
53	13	26-OCT-19 03.30.00.000000 PM	26-OCT-19 05.30.00.000000 PM	PLANNED	ONLINE	SKYPE

h. Delete one row. State the purpose of the query; show the result set before the delete; show the query; show the result set after the delete.

Query: This query removes a Topic from Student-Topic table

```
SELECT * FROM   st_topic;

DELETE FROM st_topic WHERE  topic_id = 15;

SELECT * FROM   st_topic;
```

Before

STUDENT_ID	TOPIC_ID
1001	11
1001	12
1001	13
1002	11
1002	13
1002	15

6 rows selected.

1 row(s) deleted.

After

STUDENT_ID	TOPIC_ID
1001	11
1001	12
1001	13
1002	11
1002	13

Download CSV

5 rows selected.

References:

Oppel, A. J. (2009). *Databases: A beginner's guide*. New York, NY: McGraw-Hill.

Laureate Education (Producer). (2011a). *Designing a database* [Video file]. Baltimore, MD: Author