Faculty of Engineering and Computer Science
Expectations of Originality

This form sets out the requirements for originality for work submitted by students in the Faculty of Engineering and Computer Science. Submissions such as assignments, lab reports, project reports, computer programs and take-home exams must conform to the requirements stated on this form and to the Academic Code of Conduct. The course outline may stipulate additional requirements for the course.

1. Your submissions must be your own original work. Group submissions must be the original work of the students in the group.
2. Direct quotations must not exceed 5% of the content of a report, must be enclosed in quotation marks, and must be attributed to the source by a numerical reference citation[1]. Note that engineering reports rarely contain direct quotations.
3. Material paraphrased or taken from a source must be attributed to the source by a numerical reference citation.
4. Text that is inserted from a web site must be enclosed in quotation marks and attributed to the web site by numerical reference citation.
5. Drawings, diagrams, photos, maps or other visual material taken from a source must be attributed to that source by a numerical reference citation.
6. No part of any assignment, lab report or project report submitted for this course can be submitted for any other course.
7. In preparing your submissions, the work of other past or present students cannot be consulted, used, copied, paraphrased or relied upon in any manner whatsoever.
8. Your submissions must consist entirely of your own or your group's ideas, observations, calculations, information and conclusions, except for statements attributed to sources by numerical citation.
9. Your submissions cannot be edited or revised by any other student.
10. For lab reports, the data must be obtained from your own or your lab group's experimental work.
11. For software, the code must be composed by you or by the group submitting the work, except for code that is attributed to its sources by numerical reference.

You must write one of the following statements on each piece of work that you submit:
For individual work: **"I certify that this submission is my original work and meets the Faculty's Expectations of Originality",** with your signature, I.D. #, and the date.
For group work: **"We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality",** with the signatures and I.D. #s of all the team members and the date.

A signed copy of this form must be submitted to the instructor at the beginning of the semester in each course.

I certify that I have read the requirements set out on this form, and that I am aware of these requirements. I certify that all the work I will submit for this course will comply with these requirements and with additional requirements stated in the course outline.

Course Number:	SOEN 363	Instructor:	Dr. Essam Mansour
Name:	Mohammad Ali Zahir	I.D. #	40077619
Signature:	Mohammad Ali Zahir	Date:	26 January 2022

[1] Rules for reference citation can be found in "Form and Style" by Patrich MacDonagh and Jack Bordan, fourth edition, May, 2000, available at http://www.encs.concordia.ca/scs/Forms/Form&Style.pdf.
Approved by the ENCS Faculty Council February 10, 2012

Mohammad Ali Zahir

ID: 40077619

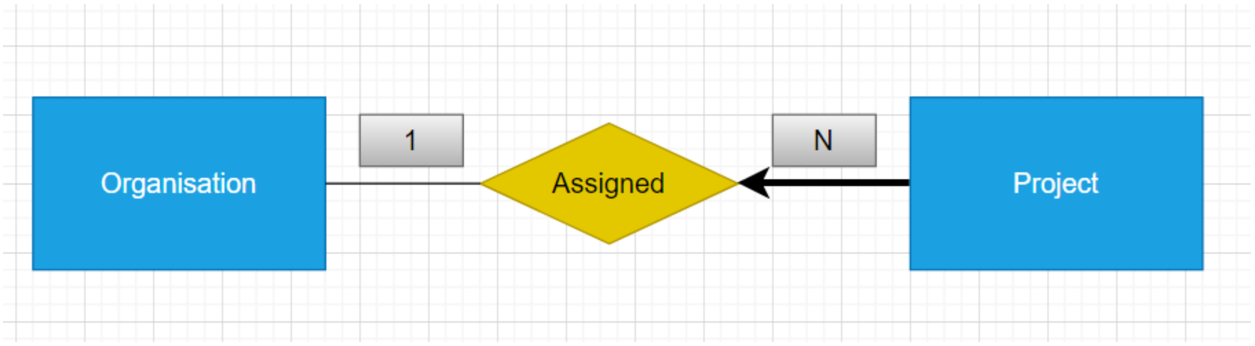
Date: 2022/01/25

SOEN 363: Data Systems for Software Engineers – Section S

SOEN 363: Assignment 1

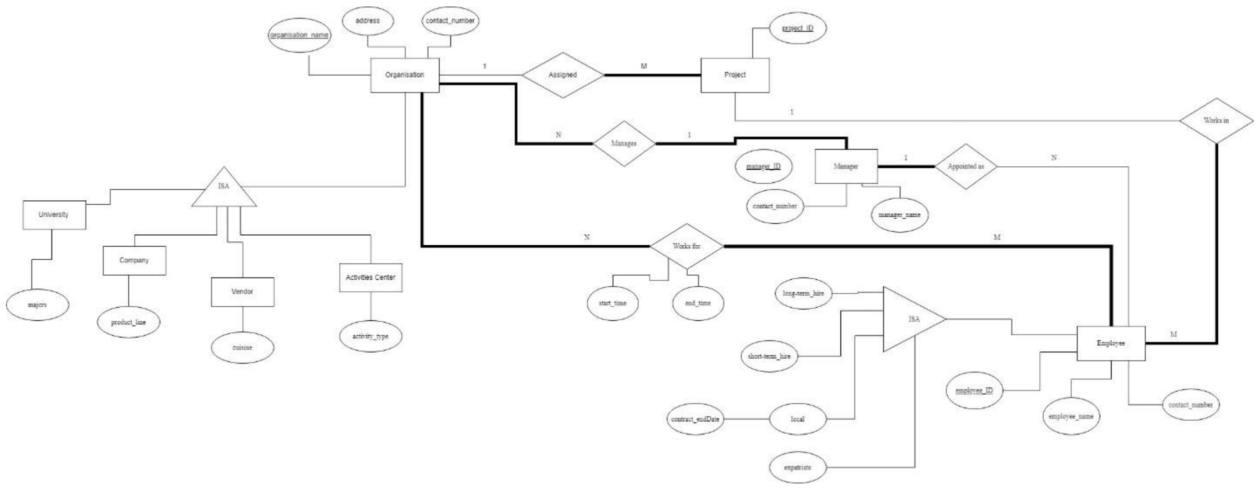
Question 1

a)



This must have a one-to-many relationship. Since one organization can be assigned to many projects. However, we would also need to add the partial, total participation since one project is assigned to one organisation only.

b)



Assumptions: Since a manager is an employee, I added the same fields for the manager which were needed for the employee.

To distinguish the different projects, I added a project_ID which is a primary key for Project.

c)

The organization ISA hierarchy has overlapping constraints. This is because an organization can be both a vendor and a company. However, this ISA hierarchy is not covering since we could have an organization which is not part of the four organization in this database (vendor, university, company or an activity center).

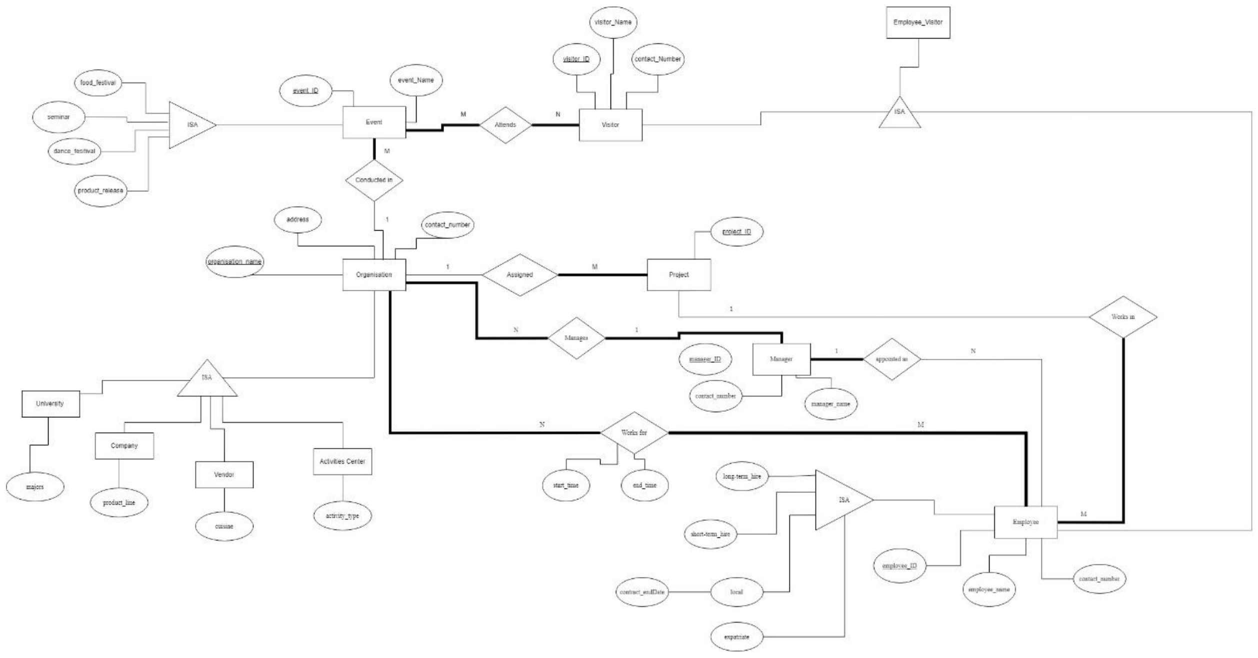
d) The employee ISA hierarchy has both covering and overlapping constraints. In terms of the covering constraints, all the employees will be short-term hires, long term hires, local or expatriate. In terms of the overlapping constraints, an employee can be a long-term hire as well as local, or another employee can be a long-term hire and an expatriate. Same would apply to the short-term hires.

e) The SQL code for the ER diagram above is:

```
CREATE TABLE Organization
(
 organisation_Name VARCHAR(30) NOT NULL UNIQUE,
 address VARCHAR(30) NOT NULL,
 contact_Number VARCHAR(10) NOT NULL,
 PRIMARY KEY (organisation_Name)
);

CREATE TABLE Employee
(
 employee_ID INT NOT NULL,
 employee_Name VARCHAR(20) NOT NULL,
 contact_Number VARCHAR(10) NOT NULL,
 PRIMARY KEY (employee_ID)
);

CREATE TABLE Works_For
(
 start_Time VARCHAR(10) NOT NULL,
 end_Time VARCHAR(10) NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (organisation_Name, employee_ID),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE University
(
 majors VARCHAR(30) NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 PRIMARY KEY (organisation_Name),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name)
);

CREATE TABLE Company
(
 product_line VARCHAR(20) NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (organisation_Name, employee_ID),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Vendor
(
 cuisine VARCHAR(20) NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (organisation_Name, employee_ID),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Activity_Center
(
 activity_types VARCHAR(20) NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (organisation_Name, employee_ID),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Manager
(
 manager_ID INT NOT NULL,
 manager_Name VARCHAR(30) NOT NULL,
 contact_Number VARCHAR(10) NOT NULL,
 PRIMARY KEY (manager_ID)
);

CREATE TABLE Project
(
 project_ID INT NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (project_ID),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE LongTerm_Hire
(
 employee_ID INT NOT NULL,
 PRIMARY KEY (employee_ID),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE ShortTerm_Hire
(
 employee_ID INT NOT NULL,
 PRIMARY KEY (employee_ID),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Local
(
 contract_EndDate VARCHAR(20) NOT NULL,
 employee_ID INT NOT NULL,
  PRIMARY KEY (employee_ID),
  FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Expatriate
(
  employee_ID INT NOT NULL,
  PRIMARY KEY (employee_ID),
  FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Works_in
(
  project_ID INT NOT NULL,
  PRIMARY KEY (project_ID),
  FOREIGN KEY (project_ID) REFERENCES Project(project_ID)
);

CREATE TABLE Assigned_To
(
  organisation_Name VARCHAR(30) NOT NULL,
  project_ID INT NOT NULL,
  PRIMARY KEY (organisation_Name, project_ID),
  FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
  FOREIGN KEY (project_ID) REFERENCES Project(project_ID)
);

CREATE TABLE Appointed_As
(
  employee_ID INT NOT NULL,
  manager_ID INT NOT NULL,
  PRIMARY KEY (employee_ID, manager_ID),
  FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID),
  FOREIGN KEY (manager_ID) REFERENCES Manager(manager_ID)
);
```

Question 2

a)



Assumptions: An employee_visitor is both a visitor and an employee, kind of like a visitor employee who is also a visitor.

Added the visitor_ID, to distinguish the different visitors.

b) The SQL code when for the above ER diagram is as follows (the part added in 2a are in bold)

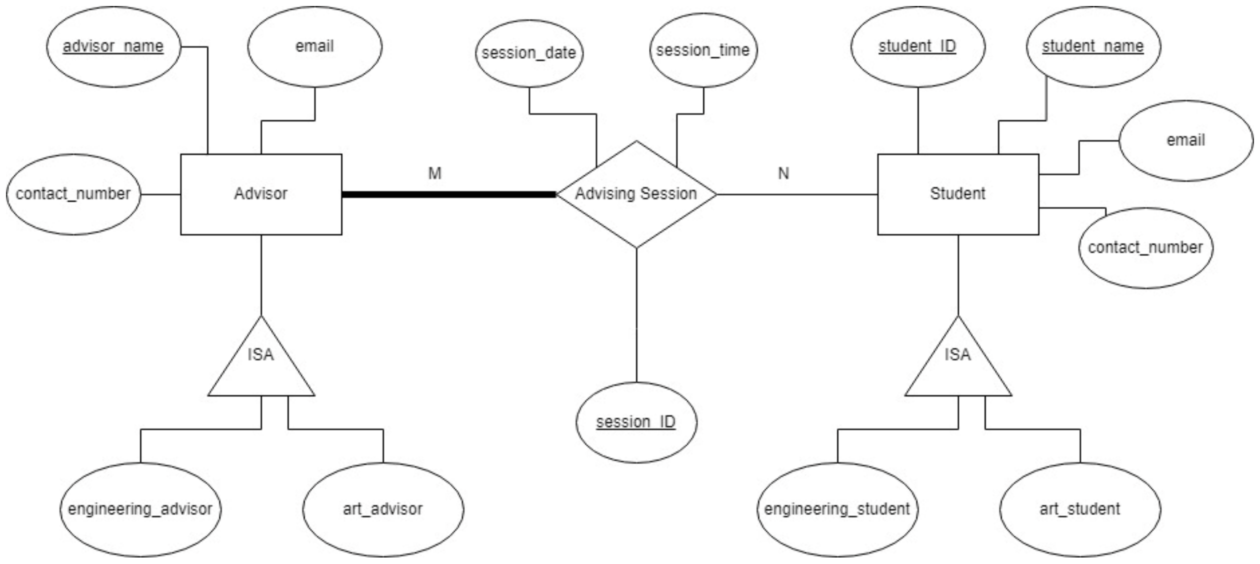
```
CREATE TABLE Organization
(
 organisation_Name VARCHAR(30) NOT NULL UNIQUE,
 address_ VARCHAR(30) NOT NULL,
 contact_Number VARCHAR(10) NOT NULL,
 PRIMARY KEY (organisation_Name)
);

CREATE TABLE Employee
(
 employee_ID INT NOT NULL,
 employee_Name_ VARCHAR(20) NOT NULL,
 contact_Number VARCHAR(10) NOT NULL,
 PRIMARY KEY (employee_ID)
);

CREATE TABLE Works_For
(
 start_Time VARCHAR(10) NOT NULL,
 end_Time VARCHAR(10) NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (organisation_Name, employee_ID),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE University
(
 majors VARCHAR(30) NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 PRIMARY KEY (organisation_Name),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name)
);

CREATE TABLE Company
(
 product_line VARCHAR(20) NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (organisation_Name, employee_ID),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Vendor
(
 cuisine VARCHAR(20) NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (organisation_Name, employee_ID),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Activity_Center
(
 activity_types VARCHAR(20) NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (organisation_Name, employee_ID),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);
```

```
CREATE TABLE Manager
(
 manager_ID INT NOT NULL,
 manager_Name VARCHAR(30) NOT NULL,
 contact_Number VARCHAR(10) NOT NULL,
 PRIMARY KEY (manager_ID)
);

CREATE TABLE Project
(
 project_ID INT NOT NULL,
 organisation_Name VARCHAR(30) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (project_ID),
 FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE LongTerm_Hire
(
 employee_ID INT NOT NULL,
 PRIMARY KEY (employee_ID),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE ShortTerm_Hire
(
 employee_ID INT NOT NULL,
 PRIMARY KEY (employee_ID),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Local
(
 contract_EndDate VARCHAR(20) NOT NULL,
 employee_ID INT NOT NULL,
 PRIMARY KEY (employee_ID),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Expatriate
(
 employee_ID INT NOT NULL,
 PRIMARY KEY (employee_ID),
 FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Works_in
(
 project_ID INT NOT NULL,
 PRIMARY KEY (project_ID),
 FOREIGN KEY (project_ID) REFERENCES Project(project_ID)
```

```
);

CREATE TABLE Assigned_To
(
  organisation_Name VARCHAR(30) NOT NULL,
  project_ID INT NOT NULL,
  PRIMARY KEY (organisation_Name, project_ID),
  FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name),
  FOREIGN KEY (project_ID) REFERENCES Project(project_ID)
);

CREATE TABLE Appointed_As
(
  employee_ID INT NOT NULL,
  manager_ID INT NOT NULL,
  PRIMARY KEY (employee_ID, manager_ID),
  FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID),
  FOREIGN KEY (manager_ID) REFERENCES Manager(manager_ID)
);

CREATE TABLE Event
(
  event_ID INT NOT NULL,
  event_Name VARCHAR(30) NOT NULL,
  organisation_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (event_ID, organisation_Name),
  FOREIGN KEY (organisation_Name) REFERENCES Organization(organisation_Name)
);

CREATE TABLE Food_Festival
(
  event_ID INT NOT NULL,
  organisation_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (event_ID, organisation_Name),
  FOREIGN KEY (event_ID, organisation_Name) REFERENCES Event(event_ID,
organisation_Name)
);

CREATE TABLE Dance_Festival
(
  event_ID INT NOT NULL,
  organisation_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (event_ID, organisation_Name),
  FOREIGN KEY (event_ID, organisation_Name) REFERENCES Event(event_ID,
organisation_Name)
);

CREATE TABLE Seminar
(
  event_ID INT NOT NULL,
  organisation_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (event_ID, organisation_Name),
  FOREIGN KEY (event_ID, organisation_Name) REFERENCES Event(event_ID, 
organisation_Name)
);

CREATE TABLE Product_Release
(
  event_ID INT NOT NULL,
  organisation_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (event_ID, organisation_Name),
  FOREIGN KEY (event_ID, organisation_Name) REFERENCES Event(event_ID, 
organisation_Name)
);

CREATE TABLE Visitor
(
  visitor_ID INT NOT NULL,
  visitor_Name VARCHAR(30) NOT NULL,
  contact_Number VARCHAR(10) NOT NULL,
  PRIMARY KEY (visitor_ID)
);

CREATE TABLE Employee_Visitor
(
  visitor_ID INT NOT NULL,
  employee_ID INT NOT NULL,
  PRIMARY KEY (visitor_ID, employee_ID),
  FOREIGN KEY (visitor_ID) REFERENCES Visitor(visitor_ID),
  FOREIGN KEY (employee_ID) REFERENCES Employee(employee_ID)
);

CREATE TABLE Attends
(
  visitor_ID INT NOT NULL,
  event_ID INT NOT NULL,
  organisation_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (visitor_ID, event_ID, organisation_Name),
  FOREIGN KEY (visitor_ID) REFERENCES Visitor(visitor_ID),
  FOREIGN KEY (event_ID, organisation_Name) REFERENCES Event(event_ID, 
organisation_Name)
);

CREATE TABLE Conducted_In
(
  event_ID INT NOT NULL,
  organisation_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (event_ID, organisation_Name),
  FOREIGN KEY (event_ID, organisation_Name) REFERENCES Event(event_ID, 
organisation_Name)
            );
```

Question 3

a)



Assumptions: Contact info was assumed to be email and contact number.

Since we know that there are only 2 types of advisors and students, we had two different ISA relationships stating these on the diagram.

We would also assume that the session_ID for the advising session is the same as the student_ID

b) The SQL code for the above diagram is as follows:

```
CREATE TABLE Advisor
(
  advisor_Name VARCHAR(30) NOT NULL,
  email VARCHAR(30) NOT NULL UNIQUE,
  contact_Number VARCHAR(10) NOT NULL,
  PRIMARY KEY (advisor_Name)
);

CREATE TABLE Student
```

```
(
  student_ID INT NOT NULL UNIQUE,
  student_Name VARCHAR(30) NOT NULL,
  email VARCHAR(30) NOT NULL UNIQUE,
  contact_Number VARCHAR(10) NOT NULL,
  PRIMARY KEY (student_ID, student_Name)
);

CREATE TABLE Advising_Session
(
  session_ID INT NOT NULL UNIQUE,
  session_Date INT NOT NULL,
  session_Time VARCHAR(20) NOT NULL,
  advisor_Name VARCHAR(30) NOT NULL,
  student_ID INT NOT NULL,
  student_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (session_ID, advisor_Name, student_ID, student_Name),
  FOREIGN KEY (advisor_Name) REFERENCES Advisor(advisor_Name),
  FOREIGN KEY (student_ID, student_Name) REFERENCES Student(student_ID,
student_Name)
);

CREATE TABLE Engineering_Advisor
(
  advisor_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (advisor_Name),
  FOREIGN KEY (advisor_Name) REFERENCES Advisor(advisor_Name)
);
```

```
CREATE TABLE Art_Advisor
(
  advisor_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (advisor_Name),
  FOREIGN KEY (advisor_Name) REFERENCES Advisor(advisor_Name)
);

CREATE TABLE Engineering_Student
(
  student_ID INT NOT NULL,
  student_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (student_ID, student_Name),
  FOREIGN KEY (student_ID, student_Name) REFERENCES Student(student_ID,
student_Name)
);

CREATE TABLE Art_Student
(
  student_ID INT NOT NULL,
  student_Name VARCHAR(30) NOT NULL,
  PRIMARY KEY (student_ID, student_Name),
  FOREIGN KEY (student_ID, student_Name) REFERENCES Student(student_ID,
student_Name)
);
```

c)

```
CREATE View [AdvisingHistory] AS

SELECT S.student_ID, S.student_Name, S.email, S.contact_Number, AD.session_Date, AD.session_Time

FROM Advising Session AD, Student S

WHERE S.student_ID = AD.session_ID;
```