

Part 1:

Scenario

A middle school is building a database to organize the information of the teachers and students, which including the courses, lectures and projects they are involved in.

It is assumed that a teacher cannot be a student as well in this scenario. And every student and teacher have to belong to just one department. Some students and teachers are involved in projects. There are various kinds of projects, and some small projects might actually be part of a big project. If a project is a part of another projects, it cannot also be part of any other projects, which means one project cannot be a part of different projects at the same time. Teachers instruct students on doing the projects. It is assumed that every project can be instructed by more than one teacher and be implemented by more than students. Also, it also assumed that every teacher and student might involve in more than one projects.

Some teachers teach courses and every student in the system at least one course. In this scenario, a teacher might teach more than one course, and every course might be taught by different instructors.

Students might choose to participate in one or more student organizations according to their interests. It assumed that one student can join several student organizations.

Some of student organizations invite certain teachers to give lectures from time to time in order to arise the students' interest of various field knowledge. It assumed that every lecture has only one topic which is also unqiue. Sudents attend the lectures they are interested in. It is also assumed that each lecture can only be presented by one lecturer, and students might attend several lectures.

The types of queries I expect:

- (1) List the last name and first name of the students who attend the lecture whose topic is X and also involved in project Y at the same time.
- (2) List all depatment name whose teacher ever present a lecture.
- (3) Find the number of girl who have ever taken the Z course.

- (4) List the name of the students who both take the course A and the Lecture whose topic is B.
- (5) List the name of teachers who are involved in Project C and also have ever gave a lecture.

Part 2. Draw an EER diagram that represents the information that should be in the database.

Please another attached file.

Part 3: Relational database design:

Step-by-Step Solution:

FK: Foreign key constraint

*: This table is revised in subsequent steps

JOB Indicates an entity, relationship or attribute depicted on the EER diagram

Step 1 – Map out the strong entities.

Leave the super/subclass entities until step 8

PROJECT (P_ID, createDT, description, isInside) *

STUDENT_ORG (Stu_org_ID, Stu_org_Name, Stu_org_des)

DEPARTMENT (D_ID, D_Name, State, ,City, Street, Zip)

COURSE (C_ID, Course_NM, Credits, Location, Year, Semester, Description)

LECTURE (L_ID, Location, Date, Description)

Step 2 – Map out weak entities.

There is no weak entity in this EER.

Step 3 – 1 to 1 relationships.

There isn't any.

Step 4 – 1 to N relationships.

The *BELONG_TO* and *PRESENT* relationships are 1:N, but one of the entities in each of these relationships is a subclass that we haven't yet mapped. The only 1:N relationship that we can map at this time is the *PART_OF* recursive relationship.

PROJECT (P_ID, createDT, description, isInside, parentProjectID)
FK: PROJECT.parentProjectID → PROJECT.P_ID

Step 5 – M:N relationships.

The relationships *TEACH*, *INSTUCTED*, *TAKE*, *PARTICIPAT*, and *ATTEND* are N:M, but one or two of the entities in each of these is a subclass that we haven't yet mapped. The only M:N relationship that we can map at this time is the *ORGANIZE* relationship.

ORGANIZE(Stu_org_ID, L_ID)
FK: ORGANIZE.Stu_org_ID → STUDENT_ORG.Stu_org_ID
ORGANIZE.L_ID → LECTURE.L_ID

Step 6 – Multi-valued attributes.

There isn't any.

Step 7 – N-Ary relationships.

Leave *INSTRUCT* until we have mapped out the super/subclass

Step 8 – Superclass/Subclass Entities.

Superclass/Subclasses	Choices Available	
PERSON/TEACHER, STUDENTS	A,B,C	Using A will make the table much more concise compared to B.

The super/subclass mappings are:

PERSON (P_ID, LastNM, FirstNM, Phone, Email, State, City, Street, Zip, BirthDT, Genre, D_ID)

FK: PERSON.D_ID → DEPARTMENT.D_ID

TEACHER (P_ID, Position, off_building, off_roomNB)

FK: TEACHER.P_ID → PERSON.P_ID

STUDENT (P_ID, GPA)

FK: STUDENT.P_ID → PERSON.P_ID

LECTURE (L_ID, Location, Date, Topic, P_ID)

FK: LECTURE.P_ID → TEACHER.P_ID

PATICIPATE (P_ID, Stu_org_ID)

FK: PATICIPATE.P_ID → STUDENT.P_ID

PARTICIPATE.Stu_org_ID → STUDENT_ORG.Stu_org_ID

ATTEND (P_ID, L_ID)

FK: ATTEND.P_ID → STUDENT.P_ID

ATTEND.L_ID → LECTURE.L_ID

INSTRUCT (Tech_ID, Stu_ID, L_ID)

FK: INSTRUCT.Tech_ID → TEACHER.P_ID

INSTRUCT.Stu_ID → STUDENT.P_ID

INSTRUCT.L_ID → LECTURE.L_ID

TEACH (P_ID, C_ID)

FK: TEACH.P_ID → TEACHER.P_ID

TEACH.C_ID → COURSE.C_ID

TAKE (P_ID, C_ID)

FK: TAKE.P_ID → STUDENT.P_ID

TAKE.C_ID → COURSE.C_ID

Final database schema (tables) for database. And indicate all the primary keys, foreign keys, and data types.

(1)

STUDENT ORG (Stu_org_ID, Stu_org Name, Stu_org des)

<u>Stu_org_ID</u> (NUMBER)	Stu_org_Name (VARCHAR2)	Stu_org_des (VARCHAR2)
-------------------------------	----------------------------	---------------------------

(2)

DEPARTMENT (D_ID, D_Name, State, City, Street, Zip)

<u>D_ID</u> (NUMBER)	D_Name (VARCHAR2)	State (VARCHAR2)	City (VARCHAR2)	Street (VARCHAR2)	Zip (NUMBER)
-------------------------	----------------------	---------------------	--------------------	----------------------	-----------------

(3)

COURSE (C_ID, Course_NM, Credits, Location, Year, Semester, Description)

<u>C_ID</u> (NUMBER)	Course_NM (VARCHAR2)	Credits (NUMBER)	Location (VARCHAR2)	Year (NUMBER)	Semester (VARCHAR2)	Description (VARCHAR2)
-------------------------	-------------------------	---------------------	------------------------	------------------	------------------------	---------------------------

(4)

LECTURE (L_ID, Location, Date, Topic, P_ID)

FK: LECTURE.P_ID → TEACHER.P_ID

<u>L_ID</u> (NUMBER)	Location (VARCHAR2)	Date (DATE)	Topic (VARCHAR2)	P_ID (NUMBER)
-------------------------	------------------------	----------------	---------------------	------------------

(5)

PROJECT (P_ID, createDT, description, isInside, parentProjectID)

FK: PROJECT.parentProjectID → PROJECT.P_ID

<u>P_ID</u> (NUMBER)	createDT (DATE)	description (VARCHAR2)	isInside (VARCHAR2)	parentProjectID (VARCHAR2)
-------------------------	--------------------	---------------------------	------------------------	-------------------------------

(6)

ORGANIZE (Stu_org_ID, L_ID)

FK: ORGANIZE.Stu_org_ID → STUDENT_ORG.Stu_org_ID

<u>Stu_org_ID</u> (NUMBER)	<u>L_ID</u> (NUMBER)
-------------------------------	-------------------------

(7)

PERSON (P_ID, LastNM, FirstNM, Phone, Email, State, City, Street, Zip, BirthDT, Genre, D_ID)

FK: PERSON.D_ID → DEPARTMENT.D_ID

<u>P_ID</u> (NUMBER)	LastNM (VARCHAR2)	FirstNM (VARCHAR2)	Phone (NUMBER)	Email (VARCHAR2)	State (VARCHAR2)
City (VARCHAR2)	Street (VARCHAR2)	Zip (NUMBER)	BirthDT (DATE)	Genre (VARCHAR2)	D_ID (NUMBER)

(8)

TEACHER (P_ID, Position, off_building, off_roomNB)

FK: TEACHER.P_ID → PERSON.P_ID

<u>P_ID</u> (NUMBER)	Position (VARCHAR2)	off_building (VARCHAR2)	off_roomNB (VARCHAR2)
-------------------------	------------------------	----------------------------	--------------------------

(9)

STUDENT (P_ID, GPA)

FK: STUDENT.P_ID → PERSON.P_ID

<u>P_ID</u> (NUMBER)	GPA (NUMBER)
-------------------------	-----------------

(10)

PATICIPATE (P_ID, Stu_org_ID)

FK: PATICIPATE.P_ID → STUDENT.P_ID

PARTICIPATE.Stu_org_ID → STUDENT_ORG.Stu_org_ID

<u>P_ID</u> (NUMBER)	<u>Stu_org_ID</u> (NUMBER)
-------------------------	-------------------------------

(11)

ATTEND (P_ID, L_ID)

FK: ATTEND.P_ID → STUDENT.P_ID

ATTEND.L_ID → LECTURE.L_ID

<u>P_ID</u> (NUMBER)	<u>L_ID</u> (NUMBER)
-------------------------	-------------------------

(12)

INSTRUCT (Tech_ID, Stu_ID, L_ID)
 FK: INSTRUCT.Tech_ID → TEACHER.P_ID
 INSTRUCT.Stu_ID → STUDENT.P_ID
 INSTRUCT.L_ID → LECTURE.L_ID

<u>Tech_ID</u> (NUMBER)	<u>Stu_ID</u> (NUMBER)	<u>L_ID</u> (NUMBER)
----------------------------	---------------------------	-------------------------

(13)

TEACH (P_ID, C_ID)
 FK: TEACH.P_ID → TEACHER.P_ID
 TEACH.C_ID → COURSE.C_ID

<u>P_ID</u> (NUMBER)	<u>C_ID</u> (NUMBER)
-------------------------	-------------------------

(14)

TAKE (P_ID, C_ID, Grade, C_Date)
 FK: TAKE.P_ID → STUDENT.P_ID
 TAKE.C_ID → COURSE.C_ID

<u>P_ID</u> (NUMBER)	<u>C_ID</u> (NUMBER)	<u>Grade</u> (FLOAT)	<u>C_Date</u> (DATE)
-------------------------	-------------------------	-------------------------	-------------------------

Part 4:

Final Relational database design and their functional dependencies:

Check functional dependency of each relationship, if they are not in 3NF then normalize it.

(1)

STUDENT_ORG (Stu_org_ID, Stu_org_Name, Stu_org_des)

FD:

{Stu_org_ID} → {Stu_org_Name, Stu_org_des },
 {Stu_org_Name} → {Stu_org_des },

Stu_org_des is a nonprime attribute of *STUDENT_ORG* which is transitively functionally dependent on the key --*Stu_org_ID* by

functionally depending on *Stu_org_Name*. So this is not in 3NF.
Normalize it:

STUDENT_ORG (Stu_org_ID, Stu_org_Name)

FD:

{Stu_org_ID} \rightarrow {Stu_org_Name}

STUDENT_ORG_DES(Stu_org_Name, Stu_org_des)

FK: STUDENT_ORG_DES. Stu_org_Name \rightarrow STUDENT_ORG.
Stu_org_Name

FD: {Stu_org_Name} \rightarrow {Stu_org_des}

(2)

DEPARTMENT (D_ID, D_Name, State, City, Street, Zip)

FD:

{D_ID} \rightarrow {D_Name, D_Phone, State, City, Street},

{City} \rightarrow {Zip}

Zip is a nonprime attribute, and it is transitively functionally dependent on the primary key -- *D_ID* by functionally depend on *City*. So this is not in 3NF. However, it seems it make more sense that *City* is in this relationship, since it together with *State*, *City*, *Street* form the address of the department. As a result, I don't further normalize it into 3NF.

(3)

COURSE (C_ID, Course_NM, Credits, Location, Year, Semester, Description)

FD:

{C_ID} \rightarrow {C_ID, Course_NM, Credits, Location, Year, Semester, Description}

Description is a nonprime attribute of *COURSE* which is transitively functionally dependent on primary key by depending on *Course_NM*. So this is not in 3NF. Normalize it:

COURSE (C_ID, Course_NM, Credits, Location, Year, Semester)

FD:

{C_ID} \rightarrow {C_ID, Course_NM, Credits, Location, Year, Semester}

COURSE_DES (Course_NM, Description)

FK: COURSE_DES. Course_NM \rightarrow COURSE. Course_NM

FD:

{Course_NM} \rightarrow {Description}

(4)

LECTURE (L_ID, Location, Date, Topic, P_ID)

FK: LECTURE.P_ID \rightarrow TEACHER.P_ID

FD:

{L_ID} \rightarrow {Location, L_Date, Topic, P_ID }

It is in 3NF.

(5)

PROJECT (P_ID, createDT, description, isInside, parentProjectID)

FK: PROJECT.parentProjectID \rightarrow PROJECT.P_ID

FD:

{ P_ID } \rightarrow {createDT, description, isInside, parentProjectID}

It is in 3NF.

(6)

ORGANIZE (Stu_org_ID, L_ID)

FK: ORGANIZE.Stu_org_ID \rightarrow STUDENT_ORG.Stu_org_ID

ORGANIZE.L_ID \rightarrow LECTURE.L_ID

There is no functional dependencies.

(7)

PERSON (P_ID, LastNM, FirstNM, Phone, Email, State, City, Street, Zip, BirthDT, Genre, D_ID)

FK: PERSON.D_ID \rightarrow DEPARTMENT.D_ID

FD:

{P_ID} \rightarrow { LastNM, FirstNM, Phone, Email, State, City, Street, BirthDT, Genre, D_ID }

{City} \rightarrow {Zip}

It is not in 3NF, but I don't normalize it. The reason is the same as the reason of (2).

(8)

TEACHER (P_ID, Position, off_building, off_roomNB)

FK: TEACHER.P_ID \rightarrow PERSON.P_ID

FD:

$\{\underline{P_ID}\} \rightarrow \{\text{Position, off_building, off_roomNB}\}$

It is in 3NF.

(9)

STUDENT (P_ID, GPA)

FK: STUDENT.P_ID \rightarrow PERSON.P_ID

FD:

$\{\underline{P_ID}\} \rightarrow \{GPA\}$

It is in 3NF.

(10)

PATICIPATE (P_ID, Stu_org_ID)

FK: PATICIPATE.P_ID \rightarrow STUDENT.P_ID

PATICIPATE.Stu_org_ID \rightarrow STUDENT_ORG.Stu_org_ID

There is no functional dependencies.

(11)

ATTEND (P_ID, L_ID)

FK: ATTEND.P_ID \rightarrow STUDENT.P_ID

ATTEND.L_ID \rightarrow LECTURE.L_ID

There is no functional dependencies.

(12)

INSTRUCT (Tech_ID, Stu_ID, L_ID)

FK: INSTRUCT.Tech_ID \rightarrow TEACHER.P_ID

INSTRUCT.Stu_ID \rightarrow STUDENT.P_ID

INSTRUCT.L_ID \rightarrow LECTURE.L_ID

There is no functional dependencies.

(13)

TEACH (P_ID, C_ID)

FK: TEACH.P_ID \rightarrow TEACHER.P_ID

TEACH.C_ID \rightarrow COURSE.C_ID

There is no functional dependencies.

(14)

TAKE (P_ID, C_ID, Grade, C_Date)

FK: TAKE.P_ID \rightarrow STUDENT.P_ID

TAKE.C_ID → COURSE.C_ID
There is no functional dependencies.

Part 5:

SQL: Create the database:

```
CREATE TABLE A8_STUDENT_ORG  
(Stu_org_ID NUMBER(9) NOT NULL,  
Stu_org_Name VARCHAR2(40) NOT NULL,  
PRIMARY KEY (Stu_org_ID));
```

```
CREATE TABLE A8_STUDENT_ORG_DES  
(Stu_org_Name VARCHAR2(40) NOT NULL,  
Stu_org_des VARCHAR2(200),  
PRIMARY KEY (Stu_org_Name));
```

```
CREATE TABLE A8_DEPARTMENT  
(D_ID NUMBER(9) NOT NULL,  
D_Name VARCHAR2(40) NOT NULL,  
D_Phone NUMBER,  
State VARCHAR2(20),  
City VARCHAR2(20),  
Street VARCHAR2(20),  
Zip NUMBER,  
PRIMARY KEY(D_ID));
```

```
CREATE TABLE A8_COURSE  
(C_ID NUMBER(9) NOT NULL,  
Course_NM VARCHAR2(40) NOT NULL,  
Credits NUMBER,  
Location VARCHAR2(40),  
Year NUMBER(4),  
Semester VARCHAR2(20),  
PRIMARY KEY(C_ID));
```

```
CREATE TABLE A8_COURSE_DES  
(Course_NM VARCHAR2(20) NOT NULL,  
Description VARCHAR2(200),
```

PRIMARY KEY (Course_NM));

```
CREATE TABLE A8_LECTURE
(L_ID NUMBER(9) NOT NULL,
Location VARCHAR2(40),
L_Date DATE,
Topic VARCHAR2(40),
PRIMARY KEY(L_ID));
```

```
CREATE TABLE A8_PROJECT
(P_ID NUMBER(9) NOT NULL,
createDT DATE,
description VARCHAR2(40),
isInside VARCHAR2(10),
parentProjectID NUMBER(9),
PRIMARY KEY(P_ID),
FOREIGN KEY (parentProjectID) REFERENCES A8_PROJECT (P_ID)
ON DELETE CASCADE);
```

```
CREATE TABLE A8_PERSON
(P_ID NUMBER(9) NOT NULL,
LastNM VARCHAR2(40),
FirstNM VARCHAR2(20),
Phone NUMBER,
Email VARCHAR2(20),
State VARCHAR2(20),
City VARCHAR2(20),
Street VARCHAR2(20),
Zip NUMBER,
BirthDT DATE,
Genre VARCHAR2(20),
D_ID NUMBER(9),
PRIMARY KEY(P_ID),
FOREIGN KEY (D_ID) REFERENCES A8_DEPARTMENT(D_ID)
ON DELETE CASCADE);
```

```
CREATE TABLE A8_STUDENT
(P_ID NUMBER(9) NOT NULL,
GPA FLOAT,
```

```
PRIMARY KEY(P_ID),  
FOREIGN KEY (P_ID)REFERENCES A8_PERSON(P_ID)  
ON DELETE CASCADE);
```

```
CREATE TABLE A8_ORGANIZE  
(Stu_org_ID NUMBER(9) NOT NULL,  
L_ID NUMBER(9),  
PRIMARY KEY(Stu_org_ID,L_ID),  
FOREIGN KEY (Stu_org_ID) REFERENCES  
A8_STUDENT_ORG(Stu_org_ID)  
ON DELETE CASCADE,  
FOREIGN KEY (L_ID) REFERENCES A8_LECTURE(L_ID)  
ON DELETE CASCADE);
```

```
CREATE TABLE A8_TEACHER  
(P_ID NUMBER(9) NOT NULL,  
Position VARCHAR2(20),  
off_building VARCHAR2(40),  
off_roomNB VARCHAR2(40),  
PRIMARY KEY (P_ID),  
FOREIGN KEY (P_ID)REFERENCES A8_PERSON(P_ID)  
ON DELETE CASCADE);
```

```
CREATE TABLE A8_PARTICIPATE  
(P_ID NUMBER(9) NOT NULL,  
Stu_org_ID NUMBER(9) NOT NULL,  
PRIMARY KEY(P_ID, Stu_org_ID),  
FOREIGN KEY (P_ID)REFERENCES A8_STUDENT(P_ID)  
ON DELETE CASCADE,  
FOREIGN KEY (Stu_org_ID) REFERENCES  
A8_STUDENT_ORG(Stu_org_ID)  
ON DELETE CASCADE);
```

```
CREATE TABLE A8_ATTEND(  
P_ID NUMBER(9) NOT NULL,  
L_ID NUMBER(9) NOT NULL,  
PRIMARY KEY(P_ID, L_ID),  
FOREIGN KEY (P_ID)REFERENCES A8_STUDENT(P_ID)  
ON DELETE CASCADE,
```

```
FOREIGN KEY (L_ID) REFERENCES A8_LECTURE(L_ID)
ON DELETE CASCADE);
```

```
CREATE TABLE A8_INSTRUCT(
Tech_ID NUMBER(9) NOT NULL,
Stu_ID NUMBER(9) NOT NULL,
P_ID NUMBER(9) NOT NULL,
PRIMARY KEY(Tech_ID, Stu_ID,P_ID),
FOREIGN KEY (Tech_ID) REFERENCES A8_TEACHER(P_ID)
ON DELETE CASCADE,
FOREIGN KEY(Stu_ID) REFERENCES A8_STUDENT(P_ID)
ON DELETE CASCADE,
FOREIGN KEY (P_ID) REFERENCES A8_PROJECT(P_ID)
ON DELETE CASCADE);
```

```
ALTER TABLE A8_DEPARTMENT DROP COLUMN D_Phone;
```

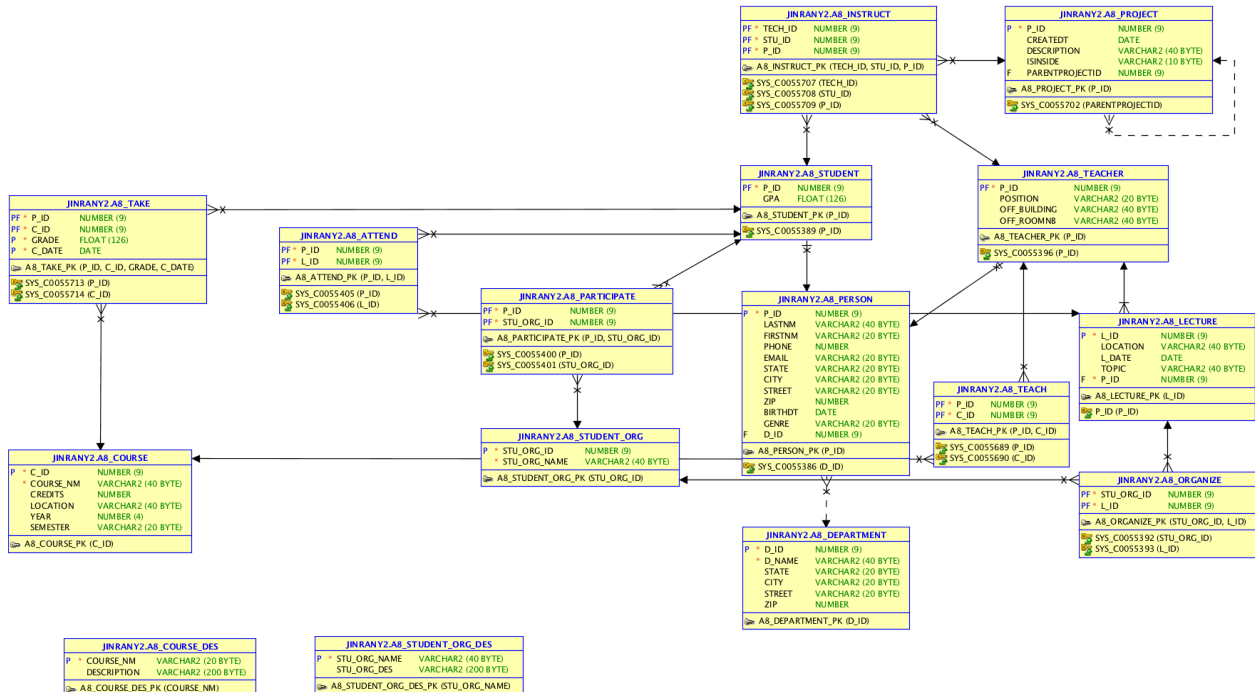
```
ALTER TABLE A8_LECTURE
ADD P_ID NUMBER(9) NOT NULL;
ALTER TABLE A8_LECTURE
ADD CONSTRAINT P_ID
FOREIGN KEY (P_ID) REFERENCES A8_TEACHER(P_ID);
```

```
CREATE TABLE A8_TEACH(
P_ID NUMBER(9) NOT NULL,
C_ID NUMBER(9) NOT NULL,
PRIMARY KEY (P_ID, C_ID),FOREIGN KEY(P_ID) REFERENCES
A8_TEACHER(P_ID)
ON DELETE CASCADE,
FOREIGN KEY(C_ID) REFERENCES A8_COURSE(C_ID)
ON DELETE CASCADE);
```

```
CREATE TABLE A8_TAKE(
P_ID NUMBER(9) NOT NULL,
C_ID NUMBER(9) NOT NULL,
Grade FLOAT,
C_Date DATE,
PRIMARY KEY (P_ID, C_ID, Grade,C_Date ),
FOREIGN KEY(P_ID) REFERENCES A8_STUDENT(P_ID)
```

ON DELETE CASCADE,
FOREIGN KEY(C_ID) REFERENCES A8_COURSE(C_ID)
ON DELETE CASCADE);

Screen shot of the relationships byusing the Data Modeler:



Add representative records:

```
INSERT INTO A8_DEPARTMENT
VALUES('001','Information Science',' ',' ','78909');
INSERT INTO A8_DEPARTMENT
VALUES('002','Business',' ',' ','89898');
INSERT INTO A8_DEPARTMENT
VALUES('003','Statistics',' ',' ','89898');
INSERT INTO A8_DEPARTMENT
VALUES('004','Computer Science',' ',' ','76789');
INSERT INTO A8_DEPARTMENT
VALUES('005','Mechanical Engineering',' ',' ','98765');
```

```
INSERT INTO A8_COURSE
VALUES('001','Data Science','4',' ','2017','Fall ');
INSERT INTO A8_COURSE
VALUES('002','Database','4',' ','2017','Spring');
INSERT INTO A8_COURSE
VALUES('003','Data Storytelling','4',' ','2017','Fall');
```

```

INSERT INTO A8_PERSON
VALUES ('001', 'Alex','Feng','2178987653',' ',' ',' ',
','78987',TO_DATE('1960-10-1','YYYY-MM-DD'),'M','001');
INSERT INTO A8_PERSON
VALUES ('002', 'Payton','Antonsen','2849098789',' ',' ',' ',
','78732',TO_DATE('1975-12-1','YYYY-MM-DD'),'F','002');
INSERT INTO A8_PERSON
VALUES ('003', 'Geogri','Ma', '2938495098',' ',' ',' ',
','67321',TO_DATE('1970-1-1','YYYY-MM-DD'),'M','003');
INSERT INTO A8_PERSON
VALUES ('004', 'Matas','Howse', '2347890345',' ',' ',' ',
','67221',TO_DATE('1973-1-1','YYYY-MM-DD'),'F','004');
INSERT INTO A8_PERSON
VALUES ('005', 'Sachin','Harper','2378590932',' ',' ',' ',
','67121',TO_DATE('1990-12-13','YYYY-MM-DD'),'M','001');
INSERT INTO A8_PERSON
VALUES ('006', 'Kwame','James', '2378909432',' ',' ',' ',
','67221',TO_DATE('1995-1-1','YYYY-MM-DD'),'F','001');
INSERT INTO A8_PERSON
VALUES ('007', 'Holger','Shea', '2384980987',' ',' ',' ',
','67221',TO_DATE('1997-1-1','YYYY-MM-DD'),'M','002');
INSERT INTO A8_PERSON
VALUES ('008', 'Alex','Wang', '2178927653',' ',' ',' ',
','67221',TO_DATE('2000-1-1','YYYY-MM-DD'),'F','002');
INSERT INTO A8_PERSON
VALUES ('009', 'Alex','Yang', '2178337653',' ',' ',' ',
','67221',TO_DATE('2005-1-1','YYYY-MM-DD'),'F','003');
INSERT INTO A8_PERSON
VALUES ('010', 'Alex','Ceng','2178937653',' ',' ',' ',
','67221',TO_DATE('2001-1-1','YYYY-MM-DD'),'F','003');
INSERT INTO A8_PERSON
VALUES ('011','Alex','Seng','2173987653',' ',' ',' ',
','67221',TO_DATE('2003-1-1','YYYY-MM-DD'),'M','005');

INSERT INTO A8_TEACHER
VALUES ('001','Assistant Professor ',' ',' ');
INSERT INTO A8_TEACHER
VALUES ('002','Assistant Professor ',' ',' ');

```



```
INSERT INTO A8_TEACHER
VALUES ('003','Associate Professor ',' ',' ');
INSERT INTO A8_TEACHER
VALUES ('004','Assistant Professor ',' ',' ');
INSERT INTO A8_TEACHER
VALUES ('005','Associate Professor ',' ',' ');
```

```
INSERT INTO A8_LECTURE
VALUES ('001', '100 Mechanical Building', TO_DATE('2017-10-1','YYYY-MM-DD'),'Peace','001');
INSERT INTO A8_LECTURE
VALUES ('002', '101 Mechanical Building', TO_DATE('2017-11-1','YYYY-MM-DD'),'Humanity','002');
INSERT INTO A8_LECTURE
VALUES ('003', '103 Mechanical Building', TO_DATE('2017-11-2','YYYY-MM-DD'),'Algorithms','003');
```

```
INSERT INTO A8_PROJECT
VALUES ('001',TO_DATE('2010-10-1','YYYY-MM-DD'),' ','NO',"") ;
INSERT INTO A8_PROJECT
VALUES ('002',TO_DATE('2011-10-1','YYYY-MM-DD'),' ','NO',"") ;
INSERT INTO A8_PROJECT
VALUES ('003',TO_DATE('2013-10-1','YYYY-MM-DD'),' ','YES','001') ;
INSERT INTO A8_PROJECT
VALUES ('004',TO_DATE('2012-10-1','YYYY-MM-DD'),' ','YES','001') ;
```

```
INSERT INTO A8_STUDENT
VALUES ('006', '2.7');
INSERT INTO A8_STUDENT
VALUES ('007', '3.0');
INSERT INTO A8_STUDENT
VALUES ('008', '3.1');
INSERT INTO A8_STUDENT
VALUES ('009', '3.2');
INSERT INTO A8_STUDENT
VALUES ('010', '3.6');
INSERT INTO A8_STUDENT
VALUES ('011', '4.0');
```

```
INSERT INTO A8_ATTEND
VALUES ('006','001');
INSERT INTO A8_ATTEND
VALUES ('007','001');
INSERT INTO A8_ATTEND
VALUES ('008','001');
INSERT INTO A8_ATTEND
VALUES ('009','001');
INSERT INTO A8_ATTEND
VALUES ('009','002');
INSERT INTO A8_ATTEND
VALUES ('010','003');
INSERT INTO A8_ATTEND
VALUES ('011','003');
INSERT INTO A8_ATTEND
VALUES ('009','003');
INSERT INTO A8_ATTEND
VALUES ('011','002');
```

```
INSERT INTO A8_INSTRUCT
VALUES ('001','006','001');
INSERT INTO A8_INSTRUCT
VALUES ('001','007','001');
INSERT INTO A8_INSTRUCT
VALUES ('002','008','002');
INSERT INTO A8_INSTRUCT
VALUES ('003','009','002');
INSERT INTO A8_INSTRUCT
VALUES ('003','009','003');
INSERT INTO A8_INSTRUCT
VALUES ('003','010','003');
```

```
INSERT INTO A8_TAKE
VALUES ('006', '001','99.2',TO_DATE('2017-1-13','YYYY-MM-DD'));
INSERT INTO A8_TAKE
VALUES ('007', '001','91.2',TO_DATE('2017-1-13','YYYY-MM-DD'));
INSERT INTO A8_TAKE
VALUES ('008', '001','92.2',TO_DATE('2017-1-13','YYYY-MM-DD'));
INSERT INTO A8_TAKE
```

```

VALUES ('009', '001', '93.2', TO_DATE('2017-8-23', 'YYYY-MM-DD'));
INSERT INTO A8_TAKE
VALUES ('006', '002', '98.2', TO_DATE('2017-8-23', 'YYYY-MM-DD'));
INSERT INTO A8_TAKE
VALUES ('007', '002', '60.2', TO_DATE('2017-1-13', 'YYYY-MM-DD'));
INSERT INTO A8_TAKE
VALUES ('008', '003', '70.9', TO_DATE('2017-1-13', 'YYYY-MM-DD'));
INSERT INTO A8_TAKE
VALUES ('009', '003', '70.8', TO_DATE('2017-1-13', 'YYYY-MM-DD'));
INSERT INTO A8_TAKE
VALUES ('010', '001', '98', TO_DATE('2017-1-13', 'YYYY-MM-DD'));
INSERT INTO A8_TAKE
VALUES ('011', '003', '67.4', TO_DATE('2017-1-13', 'YYYY-MM-DD'));

```

Part 6: Write 5 queries

(1)

Question 1:

List the last name and first name of the students who attend the lecture whose topic is peace and also involved in project whose ID is 002 at the same time.
(Using explicit join in this query)

```

SELECT A8_PERSON.LastNM, A8_PERSON.FirstNM
FROM (((A8_PERSON INNER JOIN A8_STUDENT
      ON A8_PERSON.P_ID = A8_STUDENT.P_ID)
     INNER JOIN A8_INSTRUCT
      ON A8_PERSON.P_ID = A8_INSTRUCT.Stu_ID)
    INNER JOIN A8_ATTEND
      ON A8_PERSON.P_ID = A8_ATTEND.P_ID)
     INNER JOIN A8_LECTURE
      ON A8_ATTEND.L_ID = A8_LECTURE.L_ID)
WHERE A8_INSTRUCT.P_ID = '002' AND A8_LECTURE.Topic
='Peace';

```

(2)

Question 2:

List all department name whose teacher ever present a lecture.

Also list the topic of the lectures they present.(Using explicit join in this query)

```
SELECT A8_DEPARTMENT.D_Name, A8_LECTURE.Topic
FROM ((A8_DEPARTMENT INNER JOIN A8_PERSON
      ON A8_DEPARTMENT.D_ID = A8_PERSON.D_ID)
      INNER JOIN A8_LECTURE
      ON A8_PERSON.P_ID = A8_LECTURE.P_ID);
```

(3)

Question 3:

Find the number of girl who have ever taken the Data Science course.
(Using explicit join in this query; Using aggregation function--COUNT))

```
SELECT COUNT (A8_PERSON.Genre)
FROM ((A8_PERSON INNER JOIN A8_TAKE
      ON A8_PERSON.P_ID = A8_TAKE.P_ID)
      INNER JOIN A8_COURSE
      ON A8_TAKE.C_ID = A8_COURSE.C_ID)
WHERE A8_COURSE.C_ID = '001' AND A8_PERSON.Genre = 'F';
```

(4)

Question 4:

List the name of the students who both take the course 'Data Storytelling'
and the Lecture whose topic is 'Algorithms'.
(Using implicit join)

```
SELECT A8_PERSON.LastNM, A8_PERSON.FirstNM,
A8_LECTURE.Topic, A8_COURSE.Course_NM
FROM A8_PERSON, A8_TAKE, A8_ATTEND, A8_LECTURE,
A8_COURSE
WHERE A8_PERSON.P_ID = A8_TAKE.P_ID
AND A8_TAKE.C_ID = A8_COURSE.C_ID
AND A8_COURSE.Course_NM = 'Data Storytelling'
AND A8_TAKE.P_ID = A8_ATTEND.P_ID
AND A8_ATTEND.L_ID = A8_LECTURE.L_ID
AND A8_LECTURE.Topic = 'Algorithms'
```

(5)

Question 5:

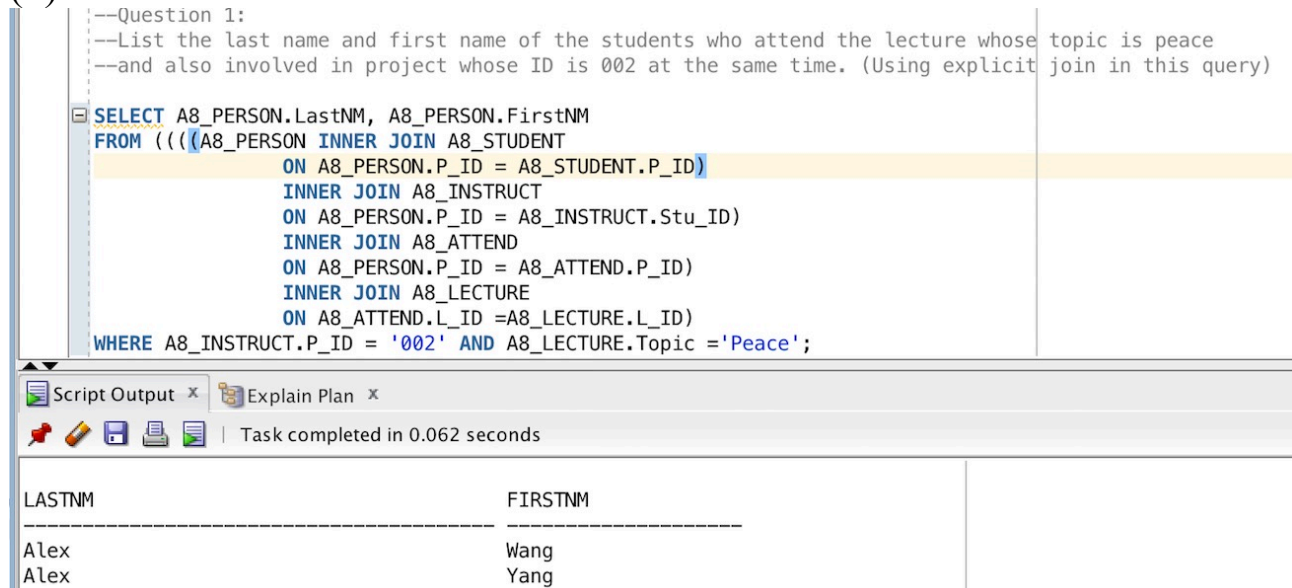
List the name of teachers who are involved in Project 'oo2' and also have ever gave a lecture.

(Using implicit join)

```
SELECT A8_PERSON.LastNM, A8_PERSON.FirstNM
FROM A8_PERSON, A8_INSTRUCT, A8_LECTURE
WHERE A8_PERSON.P_ID = A8_INSTRUCT.Tech_ID
AND A8_INSTRUCT.P_ID = '002'
AND A8_INSTRUCT.Tech_ID = A8_LECTURE.P_ID
```

Part 6: Run the 5 SQL queries and show the results

(1)



The screenshot shows a SQL IDE interface. At the top, there is a comment block:
--Question 1:
--List the last name and first name of the students who attend the lecture whose topic is peace
--and also involved in project whose ID is 002 at the same time. (Using explicit join in this query)

Below the comments is the SQL query:

```
SELECT A8_PERSON.LastNM, A8_PERSON.FirstNM
FROM (((A8_PERSON INNER JOIN A8_STUDENT
        ON A8_PERSON.P_ID = A8_STUDENT.P_ID)
      INNER JOIN A8_INSTRUCT
        ON A8_PERSON.P_ID = A8_INSTRUCT.Stu_ID)
      INNER JOIN A8_ATTEND
        ON A8_PERSON.P_ID = A8_ATTEND.P_ID)
      INNER JOIN A8_LECTURE
        ON A8_ATTEND.L_ID = A8_LECTURE.L_ID)
WHERE A8_INSTRUCT.P_ID = '002' AND A8_LECTURE.Topic = 'Peace';
```

Below the query editor, there is a status bar that says "Task completed in 0.062 seconds".

At the bottom, there is a table with two columns: LASTNM and FIRSTNM. The table contains two rows of data:

LASTNM	FIRSTNM
Alex	Wang
Alex	Yang

(2)

```
--Question 2:
--List all department name whose teacher ever present a lecture.
--Also list the topic of the lectures they present.(Using explicit join in this query)

SELECT A8_DEPARTMENT.D_Name, A8_LECTURE.Topic
FROM ((A8_DEPARTMENT INNER JOIN A8_PERSON
      ON A8_DEPARTMENT.D_ID = A8_PERSON.D_ID)
      INNER JOIN A8_LECTURE
      ON A8_PERSON.P_ID = A8_LECTURE.P_ID);
```

Script Output x

Task completed in 1.463 seconds

D_NAME	TOPIC
Information Science	Peace
Business	Humanity
Statistics	Algorithms

(3)

```
--Question 3:
--Find the number of girl who have ever taken the Data Science course.
--(Using explicit join in this query; Using aggregation function--COUNT))
```

```
SELECT COUNT (A8_PERSON.Genre)
FROM ((A8_PERSON INNER JOIN A8_TAKE
      ON A8_PERSON.P_ID = A8_TAKE.P_ID)
      INNER JOIN A8_COURSE
      ON A8_TAKE.C_ID = A8_COURSE.C_ID)
WHERE A8_COURSE.C_ID = '001' AND A8_PERSON.Genre = 'F';
```

Script Output x

Task completed in 0.179 seconds

COUNT(A8_PERSON.GENRE)
4

(4)

```
--Question 4:
--List the name of the students who both take the course 'Data Storytelling'
--and the Lecture whose topic is 'Algorithms'.
--(Using implicit join)
```

```
SELECT A8_PERSON.LastNM, A8_PERSON.FirstNM, A8_LECTURE.Topic, A8_COURSE.Course_NM
FROM A8_PERSON, A8_TAKE, A8_ATTEND, A8_LECTURE, A8_COURSE
WHERE A8_PERSON.P_ID = A8_TAKE.P_ID
AND A8_TAKE.C_ID = A8_COURSE.C_ID
AND A8_COURSE.Course_NM = 'Data Storytelling'
AND A8_TAKE.P_ID = A8_ATTEND.P_ID
AND A8_ATTEND.L_ID = A8_LECTURE.L_ID
AND A8_LECTURE.Topic = 'Algorithms'
```

Script Output x
Task completed in 0.119 seconds

LASTNM	FIRSTNM	TOPIC	COURSE_NM
Alex	Yang	Algorithms	Data Storytelling
Alex	Seng	Algorithms	Data Storytelling

(5)

```
--Question 5:
--List the name of teachers who are invloved in Project '002' and also have ever gave a lecture.
--(Using implicit join)
```

```
SELECT A8_PERSON.LastNM, A8_PERSON.FirstNM
FROM A8_PERSON, A8_INSTRUCT, A8_LECTURE
WHERE A8_PERSON.P_ID = A8_INSTRUCT.Tech_ID
AND A8_INSTRUCT.P_ID = '002'
AND A8_INSTRUCT.Tech_ID = A8_LECTURE.P_ID
```

Script Output x
Task completed in 0.257 seconds

LASTNM	FIRSTNM
Payton	Antonsen
Geogri	Ma