# Normalize all of your tables to third normal form. Make any necessary changes to the EER. Explain why these changes needed to be made.

TRACK (name, dept-abbreviation)

In TRACK table, name —> dept-abbreviation and name is primary key. It follows the definition of 3NF. Then TRACK is in 3NF.

ADVICE (prof-net-id, student-net-id)

In ADVISE table, primary key contains all attributes. Any functional dependency in this table will follows the condition in 3NF (The attributes dependent on will be part of primary key). Thus ADVICE is in 3NF.

SECTION\_HAS\_TA (ta-net-id, course-number, section-number, year, semester, workload)

In SECTION\_HAS\_TA table, non-prime attribute workload dependent on primary key. It follows conditions in 3NF. There are no other functional dependencies. Thus SECTION\_HAS\_TA is in 3NF.

TRACK\_CORE\_COURSE (track-name, course-number)

In TRACK\_CORE\_COURSE table, primary key contains all attributes. Any functional dependency in this table will follows the condition in 3NF (The attributes dependent on will be part of primary key). Thus TRACK\_CORE\_COURSE is in 3NF.

STUDENT\_PREREQUISITE (student-net-id, course-number)

In STUDENT\_PREREQUISITE table, primary key contains all attributes. Any functional dependency in this table will follows the condition in 3NF (The attributes dependent on will be part of primary key). Thus STUDENT\_PREREQUISITE is in 3NF.

HIRE (dept-abbreviation, net-id)

In HIRE table, primary key contains all attributes. Any functional dependency in this table will follows the condition in 3NF (The attributes dependent on will be part of primary key). Thus HIRE is in 3NF.

RUN (prof-net-id, building-abbreviation, room-number)

In RUN table, primary key contains all attributes. Any functional dependency in this table will follows the condition in 3NF (The attributes dependent on will be part of primary key). Thus RUN is in 3NF.

TAKE(student-net-id, course-number, section-number, year, semester, grade)

In TAKE table, non-prime attribute grade dependent on primary key. It follows conditions in 3NF. There are no other functional dependencies. Thus TAKE is in 3NF.

# Draw a dependency diagram for each table.

# Write SQL statements to create database, tables and all other structures. Primary keys and foreign keys must be defined appropriately. The quantity constraints of the relation between the entities, which should be described in EER diagram, are not required.

TRACK (name, dept-abbreviation)

CREATE TABLE TRACK

(

name VARCHAR(255) NOT NULL,

dept\_abbreviation VARCHAR(255) NOT NULL,

CONSTRAINT pk\_track PRIMARY KEY (name)

);

ADVICE (prof-net-id, student-net-id)

CREATE TABLE ADVICE

(

prof\_net\_id VARCHAR(255) NOT NULL,

student\_net\_id VARCHAR(255) NOT NULL,

CONSTRAINT pk\_advice PRIMARY KEY (prof\_net\_id, student\_net\_id)

);

SECTION\_HAS\_TA (ta-net-id, course-number, section-number, year, semester, workload)

CREATE TABLE SECTION\_HAS\_TA

(

ta\_net\_id VARCHAR(255) NOT NULL,

course\_number INTEGER NOT NULL,

section\_number INTEGER NOT NULL,

year INTEGER NOT NULL,

semester VARCHAR(255) NOT NULL,

workload DECIMAL(5,2) NOT NULL,

CONSTRAINT pk\_sht PRIMARY KEY (ta\_net\_id,course\_number, section\_number, year, semester),

CONSTRAINT chk\_sht\_course\_number CHECK (course\_number>=1000 AND course\_number<=9999),

CONSTRAINT chk\_sht\_section\_number CHECK (section\_number>=0 AND section\_number<=999),

CONSTRAINT ch\_sht\_year CHECK (year>=1000 AND year<=9999)

);

TRACK\_CORE\_COURSE (track-name, course-number)

CREATE TABLE TRACK\_CORE\_COURSE

(

track\_name VARCHAR(255) NOT NULL,

course\_number INTEGER NOT NULL,

CONSTRAINT pk\_tcc PRIMARY KEY (track\_name, course\_number),

CONSTRAINT chk\_tcc\_course\_number CHECK (course\_number>=1000 AND course\_number<=9999)

);

STUDENT\_PREREQUISITE (student-net-id, course-number)

CREATE TABLE STUDENT\_PREREQUISITE

(

student\_net\_id VARCHAR(255) NOT NULL,

course\_number INTEGER NOT NULL,

CONSTRAINT pk\_sp PRIMARY KEY (student\_net\_id, course\_number),

CONSTRAINT chk\_sp\_course\_number CHECK (course\_number>=1000 AND course\_number<=9999)

);

HIRE (dept-abbreviation, net-id)

CREATE TABLE HIRE

(

dept\_abbreviation VARCHAR(255) NOT NULL,

net\_id VARCHAR(255) NOT NULL,

CONSTRAINT pk\_hire PRIMARY KEY (dept\_abbreviation, net\_id)

);

RUN (prof-net-id, building-abbreviation, room-number)

CREATE TABLE RUN

(

prof\_net\_id VARCHAR(255) NOT NULL,

building\_abbreviation VARCHAR(255) NOT NULL,

room\_number INTEGER NOT NULL,

CONSTRAINT pk\_run PRIMARY KEY (prof\_net\_id, building\_abbreviation, room\_number),

CONSTRAINT chk\_run\_room\_number CHECK (room\_number>=1000 AND room\_number<=9999)

);

TAKE(student-net-id, course-number, section-number, year, semester, grade)

CREATE TABLE TAKE

(

student\_net\_id VARCHAR(255) NOT NULL,

course\_number INTEGER NOT NULL,

section\_number INTEGER NOT NULL,

year INTEGER NOT NULL,

semester VARCHAR(255) NOT NULL,

grade DECIMAL(3,2),

CONSTRAINT pk\_take PRIMARY KEY (student\_net\_id, course\_number, section\_number, year, semester),

CONSTRAINT chk\_take\_course\_number CHECK (course\_number>=1000 AND course\_number<=9999),

CONSTRAINT chk\_take\_section\_number CHECK (section\_number>=0 AND section\_number<=999),

CONSTRAINT chk\_take\_year CHECK (year>=1000 AND year<9999),

CONSTRAINT chk\_take\_grade CHECK (grade>=0.00 AND grade<=4.00)

);

# Use the Create View statement to create the following views:

## Department heads: List all department names with their department head's names and salaries.

CREATE VIEW [Department heads] AS

SELECT d.full\_name, p.last\_name, p.middle\_name, p.first\_name, e.salary

FROM PEOPLE p, EMPLOYEE e, DEPARTMENT d

WHERE p.net\_id = e.net\_id, e.net\_id = d.head\_prof\_net\_id;

## Students with prerequisites: List name of students who have any prerequisite course (no matter he/she had taken it or not).

CREATE VIEW [Students with prerequisites] AS

SELECT p.last\_name, p.middle\_name, p.first\_name

FROM STUDENT s, STUDENT\_PREREQUISITE sp, PEOPLE p

WHERE s.net\_id = p.net\_id AND s.net\_id = sp.student\_net\_id;

# Answer the following Queries. Feel free to use any of the views that you created in part d.

## For each department, retrieve the name and salary of employees whose salary is higher than the average salary of the department.

SELECT p.last\_name, p.middle\_name, p.first\_name, e.salary

FROM PEOPLE p, EMPLOYEE e, HIRE h

WHERE (p.net\_id = e.net\_id)

AND (e.net\_id = h.net\_id)

GROUP BY h.dept\_abbreviation

HAVING e.salary > AVG(e.salary);

## Retrieve the number of buildings which have classrooms with capacity higher than 200.

SELECT COUNT(DISTINCT building\_abbreviation)

FROM CLASSROOM

WHERE capacity>200;

## For each lecturer whose course sections have total capacity higher than 150, retrieve the lecturer's name and salary.

SELECT DISTINCT p.last\_name, p.middle\_name, p.first\_name, e.salary

FROM PEOPLE p, LECTURER l, SECTION s, EMPLOYEE e

WHERE (p.net\_id = l.ned\_id)

AND (l.net\_id = s.instructor\_net\_id)

AND (l.net\_id = [e.net](http://e.net)\_id)

AND (s.capacity > 150);

## Retrieve the name and id of students who have taken all core courses but have no advisor.

SELECT p.last\_name, p.middle\_name, p.first\_name, p.net\_id

FROM (

SELECT net\_id

FROM STUDENT

EXCEPT

SELECT DISTINCT net\_id

FROM (

SELECT s.net\_id, tic.course\_number

FROM STUDENT s, TRACK\_CORE\_COURSE tic

WHERE s.track\_name = tic.track\_name

EXCEPT

SELECT t.net\_id, t.course\_number

FROM TAKE t

WHERE t.grade IS NOT NULL

)

) AS cmpl, PEOPLE p

WHERE (cmpl.net\_id = p.net\_id)

AND (cmpl.net\_id NOT IN (

SELECT DISTINCT student\_net\_id

FROM ADVICE

))

## Retrieve the course sections which are full (enrolled student number equals capacity).

SELECT s.course\_number, s.section\_number, s.year, s.semester

FROM SECTION s, TAKE t

WHERE (s.course\_number = t.course\_number)

AND (s.section\_number = t.section\_number)

AND (s.year = t.year)

AND (s.semester = t.semester)

GROUP BY t.course\_number, t.section\_number, t.year, t.semester

HAVING s.capacity ＝ COUNT(\*);