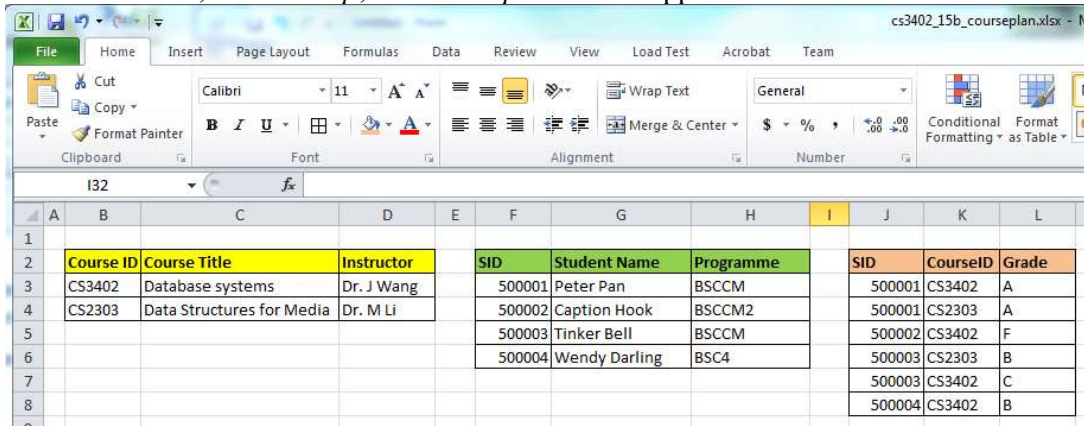


CS3402 Practice 1 (Introduction and ER Model):

- Below are some sample data stored in an Excel file. Identify *entity*, *entity set*, *attribute*, *relationship*, *relationship set* in this application.



The screenshot shows an Excel spreadsheet with the following data:

Course ID	Course Title	Instructor	SID	Student Name	Programme	SID	CourseID	Grade
CS3402	Database systems	Dr. J Wang	500001	Peter Pan	BSCCM	500001	CS3402	A
CS2303	Data Structures for Media	Dr. M Li	500002	Caption Hook	BSCCM2	500001	CS2303	A
			500003	Tinker Bell	BSCCM	500002	CS3402	F
			500004	Wendy Darling	BSC4	500003	CS2303	B
						500003	CS3402	C
						500004	CS3402	B

- Construct an ER diagram for a car insurance company with a set of customers, each of whom owns a number of cars. Each car has a number of recorded accidents associated with it.
- Construct an ER diagram for a hospital with a set of patients and a set of medical doctors. A log of the various conducted tests and results is associated with each patient.

Note the questions (2 & 3) do not contain sufficient information for building the two E/R diagrams. So you can have your own assumptions when drawing your ER diagrams.

CS3402 Practice 1:

1. Answer:

Entity: every single course, each individual student, each instructor

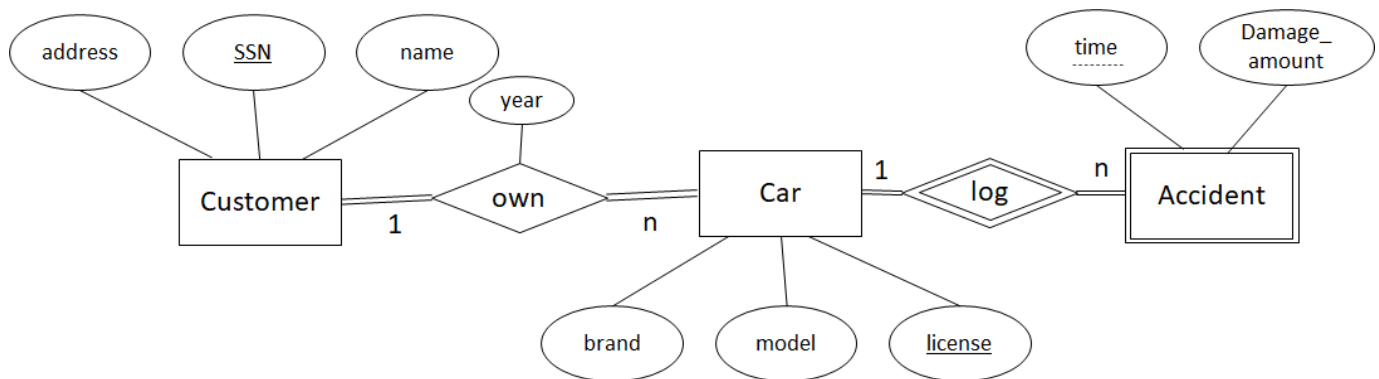
Entity set: the set of students, the set of courses, and the set of instructors

Attributes: CourseID, Course Title, Student ID, Student Name, Student Programme, Instructor Name, Grade (an attribute of a relationship).

Relationship: Dr. J Wang teaching CS3402, Dr. M Li teaching CS2303, student Peter Pan taking course CS2303, student Caption Hook taking course CS3402 ...

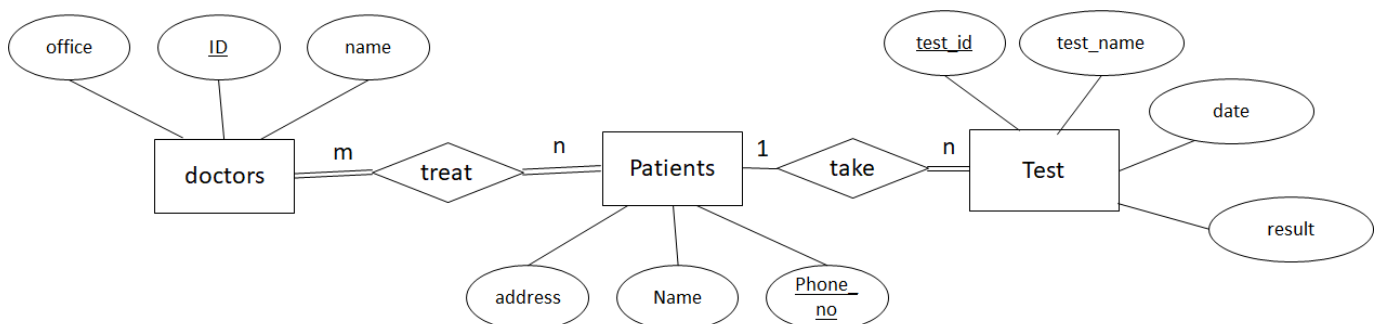
Relationship set: the set of relationships of which students taking which courses, the set of relationships of which teacher teaching which course

2. Answer:



It is assumed that one car is owned by only one customer. Each recorded accident is associated with one car.

3. Answer:



It is assumed that a doctor treats many patients and a patients may be treated by one or more doctors. One medical test only involves one patient and a patient may take no or many tests. Each test has a unique test ID.

CS3402 Practice 2:

1. Answer:

(a) Map *strong entity* type into relation

- Include simple (or atomic) attributes of the entity
- Include components of composite attributes
- Identify the primary key from the attributes
- Don't include: non-simple component of composite attributes, derived attributes, multi-valued attributes (not yet)

Assignment	<u>name</u>	weights
------------	-------------	---------

Student	<u>id</u>
---------	-----------

Discussion	<u>cn</u>	TA
------------	-----------	----

Worksheet	<u>week</u>
-----------	-------------

(b) Map *weak entity + identifying relationship* type into relation

- Include simple (or atomic) attributes
- Add the associated strong entity's primary key as attributes (also known as *foreign key* because it refers to another relation's primary key)
- Set the primary key as the combination of the *foreign* key and the partial key of the weak entity

Assignment	<u>name</u>	weights
------------	-------------	---------

Student	<u>id</u>
---------	-----------

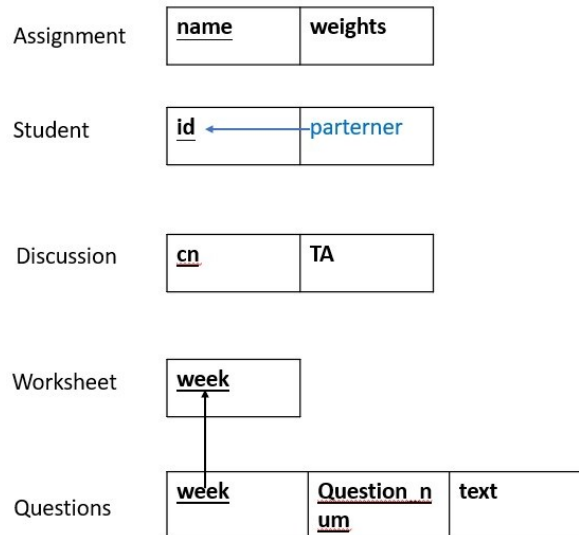
Discussion	<u>cn</u>	TA
------------	-----------	----

Worksheet	<u>week</u>
-----------	-------------

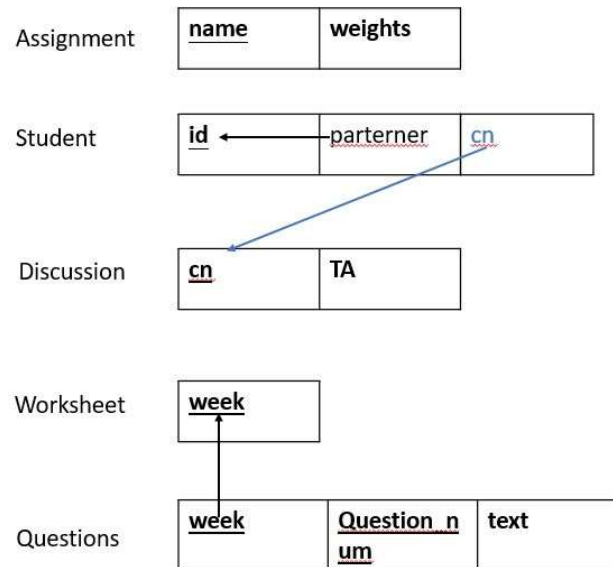
Questions	<u>week</u>	<u>Question_n um</u>	text
-----------	-------------	--------------------------	------

(c) Map binary *1:1 relationship* types into attributes

- Include the primary keys of one entity type as attributes (foreign keys) of the other entity type (*note: it is better to choose the entity in total participation to include the other entity's key as attribute*)
- Include also the simple attributes of the relationship type

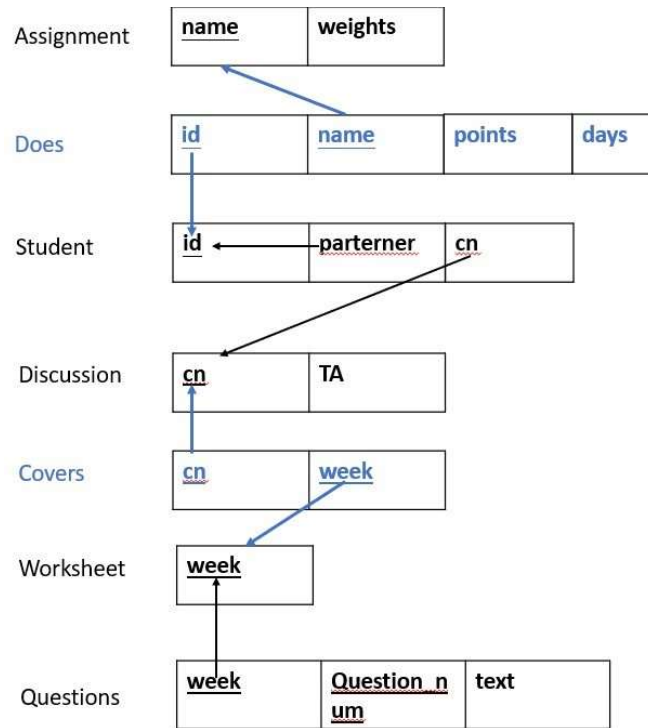
(d) Map binary *1:N Relationship* types into attributes

- In the relation representing the *N-side* entity type, add the primary keys of the *1-side* entity type as attributes (foreign key)
- Include also the simple attributes of the relationship type



(e) Map binary $M:N$ relationship type into relation

- Include the primary keys of the participating entity types as attributes (foreign key)
- Identify the primary key as the combination of the above foreign keys
- Include the simple attributes of the relationship type



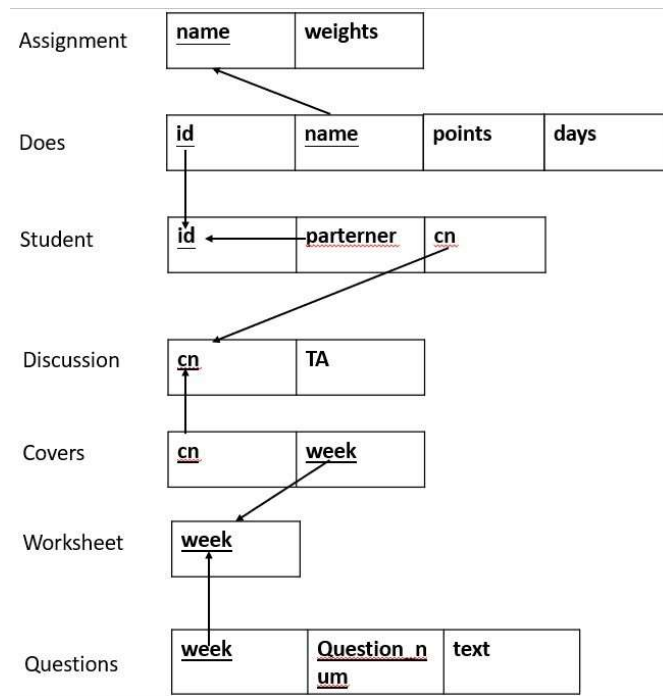
(f) Map N -ary relationship type into relation

- Similar to binary $M:N$ relationship type

(g) Map *multi-valued* attribute into relation

- Include the given attribute
- Include the primary attributes of the entity/relationship type owning the multivalued attribute
- Set the primary key to be the combination of foreign key and its original attribute

To summarize, the ER model will be translated into the following relational tables:



2 Answer:

2.1

ER Model (X:Y)	Relational Schema
M:N	$A(a) \ B(b) \ rel(a,b)$
1:N	$A(a) \ B(b,a)$
N:1	$A(a,b) \ B(b)$
1:1	$A(a) \ B(b,a) \text{ or } A(a,b) \ B(b)$

2.2

$\sqrt{(a1, b1)}$

$\sqrt{(a1, b2)}$

$\sqrt{(a2, b1)}$

$\sqrt{(a2, b2)}$

2.3 How about the 1:N case?

$\sqrt{(a1, b1)}$

$\sqrt{(a1, b2)}$

$(a2, b1)$

$(a2, b2)$

OR

$(a1, b1)$

$(a1, b2)$

$\sqrt{(a2, b1)}$

$\sqrt{(a2, b2)}$

OR

$\sqrt{(a1, b1)}$

$(a1, b2)$

$(a2, b1)$

$\sqrt{(a2, b2)}$

OR

$(a1, b1)$

$\sqrt{(a1, b2)}$

$\sqrt{(a2, b1)}$

$(a2, b2)$

2.4 How about the 1:1 case?

$\sqrt{(a1, b1)}$

$(a1, b2)$

$(a2, b1)$

$\sqrt{(a2, b2)}$

OR

$(a1, b1)$

$\sqrt{(a1, b2)}$

$\sqrt{(a2, b1)}$

$(a2, b2)$

CS3402 Practice 3:

1. Suppose each of the following Update operations is applied directly to the database below. Discuss *all* integrity constraints violated by each operation, if any, and the different ways of enforcing these constraints:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

(a) Insert < 'ProductA', 4, 'Bellaire', 2 > into PROJECT.

(b) Insert < 'Production', 4, '943775543', '1988-10-01' > into DEPARTMENT.

(c) Insert < '677678989', null, '40.0' > into WORKS_ON.

(d) Delete the EMPLOYEE tuple with SSN= '987654321'.

(e) Delete the PROJECT tuple with PNAME= 'ProductX'.

(f) Modify the SUPERSSN attribute of the EMPLOYEE tuple with SSN= '999887777' to '943775543'.

Answers:

(a) Violates **referential integrity** because DNUM=2 and there is no tuple in the DEPARTMENT relation with DNUMBER=2. We may enforce the constraint by: (i) rejecting the insertion of the new PROJECT tuple, (ii) changing the value of DNUM in the new PROJECT tuple to an existing DNUMBER value in the DEPARTMENT relation, or (iii) inserting a new DEPARTMENT tuple with DNUMBER=2.

(b) Violates both the **key constraint and referential integrity**.

Violates **the key constraint** because there already exists a DEPARTMENT tuple with DNUMBER=4. We may enforce this constraint by: (i) rejecting the insertion, or (ii) changing the value of DNUMBER in the new DEPARTMENT tuple to a value that does not violate the key constraint.

Violates **referential integrity** because MGRSSN='943775543' and there is no tuple in the EMPLOYEE relation with SSN='943775543'. We may enforce the constraint by: (i) rejecting the insertion, (ii) changing the value of MGRSSN to an existing SSN value in EMPLOYEE, or (iii) inserting a new EMPLOYEE tuple with SSN='943775543'.

(c) Violates both the **entity integrity and referential integrity**.

Violates **entity integrity** because PNO, which is part of the primary key of WORKS_ON, is null. We may enforce this constraint by: (i) rejecting the insertion, or (ii) changing the value of PNO in the new WORKS_ON tuple to a value of PNUMBER that exists in the PROJECT relation.

Violates **referential integrity** because ESSN='677678989' and there is no tuple in the EMPLOYEE relation with SSN='677678989'. We may enforce the constraint by: (i) rejecting the insertion, (ii) changing the value of ESSN to an existing SSN value in EMPLOYEE, or (iii) inserting a new EMPLOYEE tuple with SSN='677678989'.

(d) Violates **referential integrity** because several tuples exist in the WORKS_ON, DEPENDENT, DEPARTMENT, and EMPLOYEE relations that reference the tuple being deleted from EMPLOYEE. We may enforce the constraint by: (i) rejecting the deletion, or (ii) deleting all tuples in the WORKS_ON, DEPENDENT, DEPARTMENT, and EMPLOYEE relations whose values for ESSN, ESSN, MGRSSN, and SUPERSSN, respectively, is equal to '987654321'.

(e) Violates **referential integrity** because two tuples exist in the WORKS_ON relations that reference the tuple being deleted from PROJECT. We may enforce the constraint by: (i) rejecting the deletion, or (ii) deleting the tuples in the WORKS_ON relation whose value for PNO=1 (the value for the primary key PNUMBER for the tuple being deleted from PROJECT).

(f) Violates **referential integrity** because the new value of SUPERSSN='943775543' and there is no tuple in the EMPLOYEE relation with SSN='943775543'. We may enforce the constraint by: (i) rejecting the modification, (ii) changing the value of SUPERSSN to an

existing SSN value in EMPLOYEE, or (iii) inserting a new EMPLOYEE tuple with SSN='943775543'.

2 Consider the relation REFRIG(MODEL#, YEAR, PRICE, MANUF_PLANT, COLOR), which is abbreviated as REFRIG(M, Y, P, U, C), and the following set of F of functional dependencies: $F = \{M \rightarrow U, \{M, Y\} \rightarrow P, U \rightarrow C\}$. Evaluate each of the following as a candidate key for REFRIG, giving reasons why it can or cannot be a key: $\{M\}$, $\{M, Y\}$, $\{M, C\}$

Answers:

- $\{M\}$ IS NOT a candidate key since it does not functionally determine attributes Y or P.
 $\{M\}^+ = \{M, U, C\}$

- $\{M, Y\}$ IS a super key since it functionally determines the remaining attributes P, U, and C.
Also $\{M\}$ and $\{Y\}$ are not the superkey, so $\{M, Y\}$ IS a candidate key.
i.e.

We have $\{M, Y\} \rightarrow P$, and $M \rightarrow U$, by augmentation $\{M, Y\} \rightarrow U$

Since $U \rightarrow C$, by transitivity $M \rightarrow U$, $U \rightarrow C$, gives $M \rightarrow C$; By augmentation $\{M, Y\} \rightarrow C$

Thus $\{M, Y\}^+ = \{M, Y, P, U, C\}$ and $\{M, Y\}$ can be a super key.

$\{M\}^+ = \{M, U, C\}$, $\{M\}$ is not super key.

$\{Y\}^+ = \{Y\}$, $\{Y\}$ is not super key.

So $\{M, Y\}$ is the candidate key.

- $\{M, C\}$ IS NOT a candidate key since it does not functionally determine attributes Y or P.
 $\{M, C\}^+ = \{M, C, U\}$.

CS3402 Practice 4:

1. Examine the table shown below.

Branch

Branch No	BranchAddress	TelNo
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618, 503-555-2727, 503-555-6534
B002	City Center Plaza, Seattle, WA 98122	206-555-6756, 206-555-8836
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131, 206-555-4112

- (a) Why this table is not in 1NF?
 (b) Describe and illustrate the process of normalizing the data shown in this table to third normal form (3NF).

Answer:

- (a) *TelNo* is not an attribute with atomic values, but with multi-values. So, the table is NOT in 1NF.
 (b) Create another relation specifically for *TelNo* with *BranchNo* as a foreign key

Branch

<u>BranchNo</u>	BranchAddress
B001	8 Jefferson Way, Portland, OR 97201
B002	City Center Plaza, Seattle, WA 98122
B003	14 – 8th Avenue, New York, NY 10012
B004	16 – 14th Avenue, Seattle, WA 98128

BranchTel

<u>BranchNo</u>	<u>TelNo</u>
B001	503-555-3618
B001	503-555-2727
B001	503-555-6534
B002	206-555-6756
B002	206-555-8836
B003	212-371-3000
B004	206-555-3131
B004	206-555-4112

2. Examine the table shown below.

StaffBranchAllocation

StaffNo	BranchNo	BranchAddress	Name	Position	HoursPer
---------	----------	---------------	------	----------	----------

					Week
S4555	B002	City Center Plaza, Seattle, WA 98122	Ellen Layman	Assistant	16
S4555	B004	16 – 14th Avenue, Seattle, WA 98128	Ellen Layman	Assistant	9
S4612	B002	City Center Plaza, Seattle, WA 98122	Dave Sinclair	Assistant	14
S4612	B004	16 – 14th Avenue, Seattle, WA 98128	Dave Sinclair	Assistant	10

<StaffNo, BranchNo> is the primary key.

<StaffNo> -> <Name, Position>; <BranchNo> -> <BranchAddress>

(a) Why this table is not in 2NF?

(b) Describe and illustrate the process of normalizing the data shown in this table to third normal form (3NF).

Answer:

(a) The primary key of StaffBranchAllocation table is <StaffNo, BranchNo>. There exist the partial functional dependencies: *StaffNo* → *Name, Position* and *BranchNo* → *BranchAddress*. The non-key attributes are not fully dependent on the key. So, the table is NOT in 2NF.

(b) Remove *BranchAddress, Name, Position* from StaffBranchAllocation relation to capture the partial functional dependencies separately.

Branch

<u>BranchNo</u>	BranchAddress
B002	City Center Plaza, Seattle, WA 98122
B004	16 – 14th Avenue, Seattle, WA 98128

Staff

<u>StaffNo</u>	Name	Position
S4555	Ellen Layman	Assistant
S4612	Dave Sinclair	Assistant

StaffBranchAllocation

<u>StaffNo</u>	<u>BranchNo</u>	HoursPerWeek
S4555	B002	16
S4555	B004	9
S4612	B002	14
S4612	B004	10

3. Examine the table shown below.

BranchManager

BranchNo	BranchAddress	TelNo	MgrStaffNo	MgrName
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618	S1500	Tom Daniels
B002	City Center Plaza, Seattle, WA 98122	206-555-6756	S0010	Mary Martinez
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000	S0145	Art Peters
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131	S2250	Sally Stern

<BranchNo> is the primary key; <MgrStaffNo> -> <MgrName>

(a) Why this table is not in 3NF?

(b) Describe and illustrate the process of normalizing the data shown in this table to third normal form (3NF).

1. **Answer:**

(a) There exists a non-key attribute transitively dependent on the key, i.e.,

MgrName depends on *MgrStaffNo* and *MgrStaffNo* depends on *BranchNo*.

(b) Create another relation which specifically captures the dependency

MgrStaffNo → *MgrName*

Branch

BranchNo	BranchAddress	TelNo	MgrStaffNo
B001	8 Jefferson Way, Portland, OR 97201	503-555-3618	S1500
B002	City Center Plaza, Seattle, WA 98122	206-555-6756	S0010
B003	14 – 8th Avenue, New York, NY 10012	212-371-3000	S0145
B004	16 – 14th Avenue, Seattle, WA 98128	206-555-3131	S2250

ManagerStaff

MgrStaffNo	MgrName
S1500	Tom Daniels
S0010	Mary Martinez
S0145	Art Peters
S2250	Sally Stern

4. Examine the table shown below and the set of functional dependency on its attributes:

CourseRmAlloc (CourseId, CourseName, Year, Lecturer, Enrollment, RoomId, RoomCapacity, Day, Time)

FD = { *CourseId* -> *CourseName*, *CourseName* -> *CourseId*,
 CourseId, *Year* -> *Lecturer*, *CourseId*, *Year* -> *Enrollment*,
 RoomId -> *RoomCapacity*, *RoomId*, *Year*, *Day*, *Time* -> *CourseId*,

CourseId, Year, Day, Time -> RoomId }

- (a) Find all candidate keys of this table.
- (b) Decompose this table into a design into BCNF.

Answer:

- (a) There are three candidate keys in this table:

(Year, Day, Time, CourseId)

(Year, Day, Time, CourseName)

(Year, Day, Time, RoomId)

- (b) This table can be decomposed into the following in BCNF (so also in 3NF):

CourseTeaching (CourseId, Year, Lecturer, Enrollment)

Room (RoomId, RoomCapacity)

CourseRoomAlloc (CourseId, Year, Day, Time, RoomId)

Course (CourseId, CourseName)

CS3402 Practice 5:

Given 4 tables shown below.

Student

s_id	s_name	s_birth	s_sex
01	ZHAO Lei	2000-01-01	Male
02	QIAN Dian	2000-12-21	Female
03	SUN Feng	2000-05-20	Male
04	LI Yun	2000-08-06	Male
05	ZHOU Mei	2001-12-01	Female
06	WU Lan	2002-03-01	Female
07	ZHENG Zhu	NULL	Female
08	WANG Jv	2001-01-20	Female
09	QIAN Dian	2001-7-15	Female

Course

c_id	c_name	t_id
01	Chinese	02
02	Math	01
03	English	03

Teacher

t_id	t_name
01	ZHANG San
02	LI Si
03	WANG Wu

Sc

s_id	c_id	s_Sc
01	01	80
01	02	90
01	03	99
02	01	70
02	02	60
02	03	80
03	01	80
03	02	80
03	03	90
04	01	50
04	02	30

04	03	20
05	01	76
05	02	87
06	01	31
06	03	34
07	02	89
07	03	98

1. Query male student's name.

Answer:

```
SELECT
    s_name
FROM
    Student
WHERE
    s_sex = 'Male';
```

2. Query the name and score of each course that 'ZHAO Lei' took.

Answer:

```
SELECT
    c_name, s_Sc
FROM
    Sc,
    Student,
    Course
WHERE
    Student.s_name = 'ZHAO Lei'
    AND Sc.s_id = Student.s_id
    AND Course.c_id = Sc.c_id;
```

3. Query name of students with higher grades in the '02' course than in the '03' course.

Answer:

```
SELECT
    s_name
FROM
    Student a,
    Sc b,
    Sc c
WHERE
    a.s_id = b.s_id
    AND a.s_id = c.s_id
    AND b.c_id = '02'
    AND c.c_id = '03'
    AND b.s_Sc > c.s_Sc;
```

4. Query the information of students whose birth date is NULL.

Answer:

```
SELECT
    *
FROM
    Student
WHERE
    Student.s_birth IS NULL;
```

5. Query the id of teachers surnamed 'LI'.

Answer:

```
SELECT
    t_id
FROM
    Teacher
WHERE
    t_name LIKE 'LI %';
```

6. Query information of students who born after '2001-01-01'.

Answer:

```
SELECT
    *
FROM
    Student
WHERE
    s_birth > Date('2001-01-01');
```

7. Retrieve name and '01' course score of students whose score of '01' course is less than 60, sorted in descending order of the score.

Answer:

```
SELECT
    a.s_name,
    b.s_Sc
FROM
    Student a,
    Sc b
WHERE
    a.s_id = b.s_id
    AND b.c_id = '01'
    AND b.s_Sc < 60
ORDER BY
    b.s_Sc DESC;
```

8. Retrieve information about students who have studied '01' course and '02' course, sorted in ascending order of the sum score of two courses.

Answer:

```
SELECT
    a.*,
    b.s_Sc + c.s_Sc
FROM
    Student a,
```

```

        Sc b,
        Sc c
WHERE
    a.s_id = b.s_id
    AND a.s_id = c.s_id
    AND b.c_id = '01'
    AND c.c_id = '02'
ORDER BY
    b.s_Sc + c.s_Sc ASC;

```

9. Retrieve the distinct names of students.

Answer:

```

SELECT
    DISTINCT s_name
FROM
    Student;

```

10. Query id of the courses that 'ZHENG Zhu' and 'SUN Feng' both took.

Answer:

```

SELECT
    c_id
FROM
    Sc,
    Student
WHERE
    Student.s_name = 'ZHENG Zhu'
    AND Sc.s_id = Student.s_id
INTERSECT
SELECT
    c_id
FROM
    Sc,
    Student
WHERE
    Student.s_name = 'SUN Feng'
    AND Sc.s_id = Student.s_id;

```

CS3402 Practice 6:

1. Query the name of students who have studied 'English' course.

Answer:

```
SELECT
    s_name
FROM
    Student
WHERE
    s_id IN (
        SELECT
            s_id
        FROM
            Sc S, Course C
        WHERE
            S.c_id = C.c_id and C.c_name = 'English'
    );
```

2. Query the information of students who have studied the course with id '01' but have not studied the course with id '02'.

Answer:

```
SELECT
    *
FROM
    Student
WHERE
    s_id IN (
        SELECT
            s_id
        FROM
            Sc
        WHERE
            c_id = '01'
    )
    AND s_id NOT IN (
        SELECT
            s_id
        FROM
            Sc
        WHERE
            c_id = '02'
    );
```

3. Query the names of students who have not studied any course taught by teacher 'ZHANG San'.

Answer:

```
SELECT S.s_name
```

```

FROM Student S
WHERE
    NOT EXISTS (
        SELECT c_id
        FROM Sc
        WHERE Sc.s_id=S.s_id

        INTERSECT

        SELECT c_id
        FROM Course C, Teacher T
        WHERE C.t_id = T.t_id AND t_name = 'ZHANG San'
    );

```

4. Query course ids and the number of students enrolling for each course.

Answer:

```

SELECT
    c_id, COUNT(s_id)
FROM
    Sc
GROUP BY
    c_id;

```

5. Query the average score of each course, and sort the results in descending order of the average score.

Answer:

```

SELECT
    c_id, AVG(s_Sc)
FROM
    Sc
GROUP BY
    c_id
ORDER BY
    AVG(s_Sc) DESC;

```

6. Query id and name of students who enroll only 2 courses.

Answer:

```

SELECT
    s_id, s_name
FROM
    Student
WHERE
    s_id IN (
        SELECT
            s_id
        FROM
            Sc
        GROUP BY
            s_id
    );

```

```

HAVING
    COUNT(c_id) = 2
);

```

7. Query information of students who have taken all courses.

Answer:

```

SELECT
    *
FROM
    Student
WHERE
    s_id IN (
        SELECT
            s_id
        FROM
            Sc
        GROUP BY
            s_id
        HAVING
            COUNT(*) = (
                SELECT
                    COUNT(*)
                FROM
                    Course
            )
    );

```

8. Display student id and average grades for each student in descending order of average grades.

Answer:

```

SELECT
    Sc.s_id, AVG(s_Sc)
FROM
    Sc
GROUP BY
    Sc.s_id
ORDER BY
    AVG(s_Sc) DESC;

```

9. Query the information of other students who took at least one same course as the student with id '01'.

Answer:

```

SELECT
    DISTINCT Student.*
FROM
    Student, SC
WHERE
    Student.s_id=SC.s_id
    AND SC.c_id IN (
        SELECT

```

```

        a.c_id
FROM
    Sc a
WHERE
    a.s_id = '01'
    )
AND Student.s_id != '01';

```

10. Query the information of the students who got highest score in the 'Math' course.

Answer:

```

SELECT
    *
FROM
    Student
WHERE
    Student.s_id IN (
        SELECT s_id
        FROM Sc
        WHERE (Sc.c_id, Sc.s_sc) = (
            SELECT Sc.c_id, MAX(Sc.s_sc)
            FROM Sc, Course
            WHERE Sc.c_id = Course.c_id
            AND Course.c_name = 'Math'
            Group BY Sc.c_id
        )
    );

```