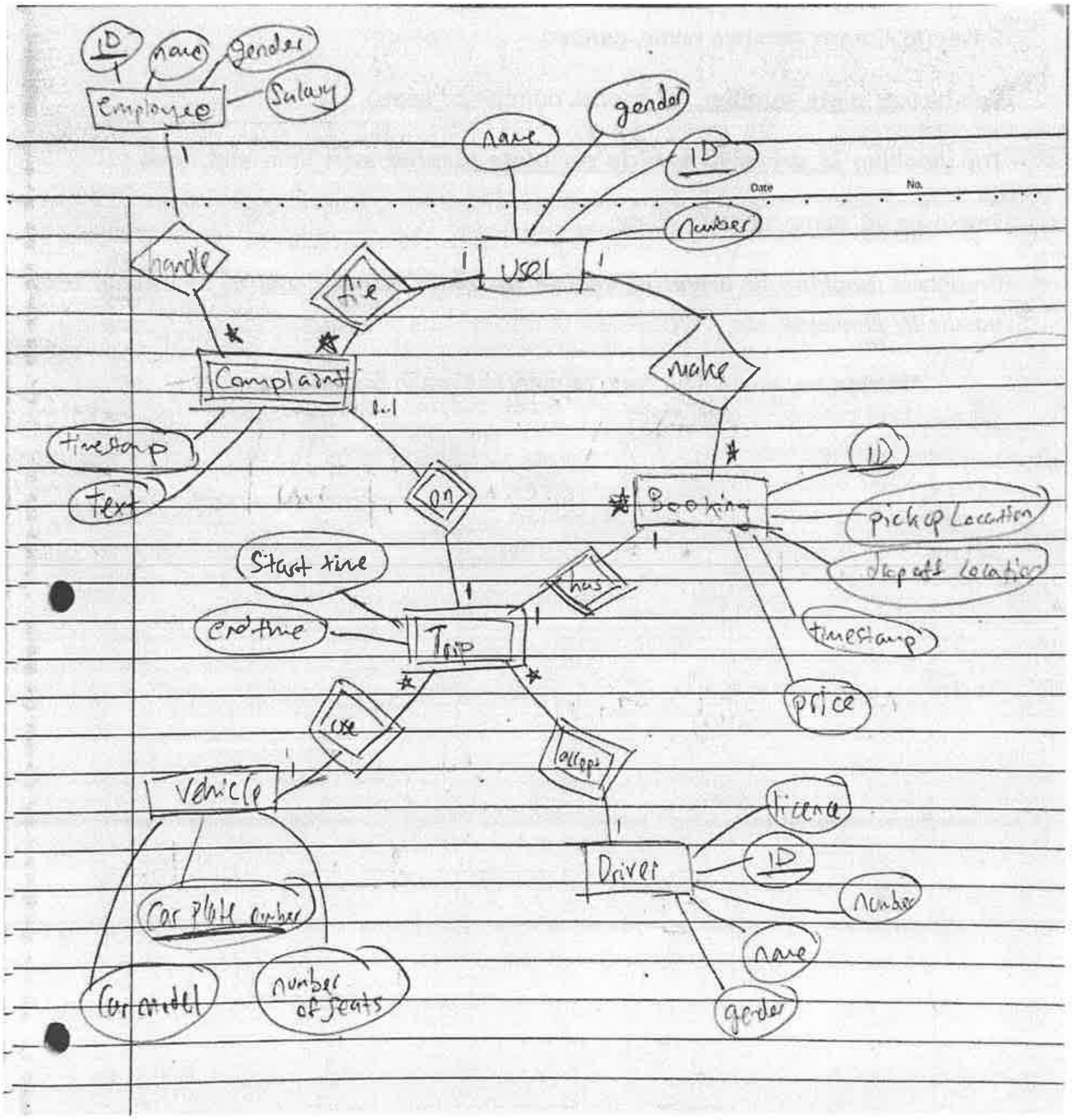
Solver: Shao Jie

Email Address: yews0012@e.ntu.edu.sg

1. (a) (i)



(ii).

User (**id**, name, gender, number)

Booking (**id**, pickup_location, dropoff_location, timestamp, price, *user_id*)

Driver (**id**, license, number, name, gender)

Vehicle (**car_plate_number**, car_model, number_of_seats)

Trip (***booking_id, driver_id, vehicle_car_plate_number***, start_time, end_time)

Employee (**id**, name, gender, salary)

Complaint (***booking_id, driver_id, vehicle_car_plate_number***, *user_id*, timestamp, text, *handle_by_employee_id*)

*foreign key are in italic font, primary key are in bolded font.

(b)

(b)(i)

$R1 := \gamma_{count(CID) \rightarrow count,\ CID}\ MATCH$

$R2 := \pi_{CID}\ \sigma_{count \geq 100}\ R1$

(ii)

$R1 := \gamma_{count(P1) \rightarrow count1,\ P1}\ \sigma_{P1Wins = True}\ MATCH$

$R2 := \gamma_{count(P2) \rightarrow count2,\ P2}\ \sigma_{P1Wins = False}\ MATCH$

$R3 := PLAYER \bowtie_{PID = P1} R1 \bowtie_{PID = P2} R2$

$R4 := \pi_{PID,\ (count1 + count2) \rightarrow MatchWon}\ R3$

(iii)

$R1 := \pi_{CID}\ \sigma_{Type = 'Grass'}\ COURT$

$R2 := MATCH \bowtie R1$

$R3 := \delta(\pi_{P1}\ \sigma_{P1Wins = False}\ R2)$

$R4 := \delta(\pi_{P2}\ \sigma_{P1Wins = TRUE}\ R2)$

$R5 := PLAYER \overset{o}{\bowtie}_{L\ PID = P1} R3$

$R6 := R5 \overset{o}{\bowtie}_{L\ PID = P2} R4$

$R7 := \pi_{PID, Name, Gender, DOB}\ \sigma_{P1\ is\ NULL\ and\ P2\ is\ NULL}$

(iv)

$R1 := \gamma_{count(P1) \rightarrow count,\ P1}\ \sigma_{P1Wins = True}\ MATCH$

$R2 := \gamma_{count(P2) \rightarrow count,\ P2}\ \sigma_{P1Wins = False}\ MATCH$

$R3 := PLAYER \bowtie_{PID = P1} R1 \bowtie_{PID = P2} R2$

$R4 := \pi_{PID,\ (count1 + count2) \rightarrow MatchWon}\ R3$

$R5 := \pi_{matchWon \rightarrow matchWon2}\ R4$

$R6 := R4 \bowtie_{matchWon <= matchWon2} R5$

$R7 := \gamma_{count(PID) \rightarrow count,\ PID}\ R6$

$R8 := \pi_{PID}\ \sigma_{count = 100}\ R7$

2. (a)

R (A, B, C, D, E)
Given input functional dependencies: E → C,D; A,B → C; A,D → B; B,D → C; B,D → E; B,E → D.
Calculating attribute closures:
{A}+ = {A}
{B}+ = {B}
{C}+ = {C}
{D}+ = {D}
{E}+ = {C, D, E}
{A, B}+ = {A, B, C}
{A, C}+ = {A, C}
{B, C}+ = {B, C}
{A, D}+ = {A, B, C, D, E} <--- Composite minimum candidate key
{B, D}+ = {B, C, D, E}
{C, D}+ = {C, D}
{A, E}+ = {A, B, C, D, E} <--- Composite minimum candidate key
{B, E}+ = {B, C, D, E}
{C, E}+ = {C, D, E}
{D, E}+ = {C, D, E}
{A, B, C}+ = {A, B, C}
{A, B, D}+ = {A, B, C, D, E} <--- Superkey
{A, C, D}+ = {A, B, C, D, E} <--- Superkey
{B, C, D}+ = {B, C, D, E}
{A, B, E}+ = {A, B, C, D, E} <--- Superkey
{A, C, E}+ = {A, B, C, D, E} <--- Superkey
{B, C, E}+ = {B, C, D, E}
{A, D, E}+ = {A, B, C, D, E} <--- Superkey
{B, D, E}+ = {B, C, D, E}
{C, D, E}+ = {C, D, E}
{A, B, C, D}+ = {A, B, C, D, E} <--- Superkey
{A, B, C, E}+ = {A, B, C, D, E} <--- Superkey
{A, B, D, E}+ = {A, B, C, D, E} <--- Superkey
{A, C, D, E}+ = {A, B, C, D, E} <--- Superkey
{B, C, D, E}+ = {B, C, D, E}
{A, B, C, D, E}+ = {A, B, C, D, E} <--- Superkey

Keys: {A, D}, {A, E}

Input relation is not in BCNF: it is not in 3NF and not all functional dependencies satisfy at least one of the following conditions:

(1) The right-hand side is a subset of the left hand side, or

(2) the left-hand side is a super key (or minimum key) of the relation. The functional dependencies that failed are: E → C; E → D; A, B → C; B, D → E.

Decomposing input relation into BCNF relations using input relation and input functional dependencies as sources.
Initial set of decomposed BCNF relations:
R_0(C, D, E) having FD(s): E → C; E → D.
R_1(A, B, E) having FD(s): (none).
R_2(A, B, C) having FD(s): A,B → C.
R_3_0(D, E) having FD(s): E → D.
R_3_1(A, B, E) having FD(s): (none).
R_3_2_0(D, E) having FD(s): E → D.
R_3_2_1(B, E) having FD(s): (none).
R_3_3(A, B, D) having FD(s): A,D → B.
R_4_0(C, D, E) having FD(s): E → C; E → D.
R_4_1(B, E) having FD(s): (none).
R_5(A, B, D) having FD(s): A,D → B.

Final set of decomposed BCNF relations (removing duplicate and subset relations):
R_0(C, D, E) having FD(s): E → C; E → D.
R_1(A, B, E) having FD(s): (none).
R_2(A, B, C) having FD(s): A,B → C.
R_3_3(A, B, D) having FD(s): A,D → B.
The following input functional dependencies were lost: B,D → C; B,D → E; B,E → D.

(b)

Input relation is not in 3NF: it is not in 2NF and not all functional dependencies satisfy at least one of the following conditions:

(1) The right-hand side is a subset of the left hand side,

(2) the left-hand side is a super key (or minimum key) of the relation, or

(3) the right-hand side is (or is a part of) some minimum key of the relation. The functional dependencies that failed are: E → C; A, B → C.

Decomposing input relation into 3NF using canonical functional dependency cover (lossless and preserving all minimal cover set functional dependencies):
Decomposing input relation into 3NF relations using the Synthesis algorithm. For each functional dependency of the canonical cover set (merging functional dependencies having the same left-hand attribute(s)) of original relation's functional dependencies, create a relation schema with the attributes in that functional dependency (both sides). Checking if at least one key can be found in at least one newly formed 3NF relation. Since key {A, D} is present in at least one of the new 3NF relations, no new relation was created. Testing if any relation includes all of the attributes found in another relation (and deleting the duplicate or smaller one). No new relations were removed. Finished decomposing input relation into 3NF relations:

R0(C, D, E) having FD(s): E → C; E → D.
R1(A, B, C) having FD(s): A,B → C.
R2(A, B, D) having FD(s): A,D → B.
R3(B, D, E) having FD(s): E → D; B,D → E.

3. (a)
   (i) 1
   (ii) 1
   (iii) 5 (EXCEPT ALL which returns all records from the first table which are not present in the second table, leaving the duplicates as is.)

(b)

```
SELECT DISTINCT t.name
FROM TEAMS t
INNER JOIN PLAYERS p ON t.teamid = p.teamid
WHERE p.numgoals>5;
```

(c)

```
CREATE VIEW PromotionSummary AS
SELECT category, MIN(price) as minprice, MAX(price) as maxprice
FROM BOOKS
WHERE promoted = True
```

(d)

(i)

```
UPDATE EMP
SET salary = salary/2
WHERE salary = (SELECT MAX(salary) FROM EMP);
```

(ii)

```
SELECT e.name
FROM EMP e
WHERE NOT EXIST (SELECT *
                 FROM DEPENDENT d
                 WHERE d.essn = e.ssn)
```

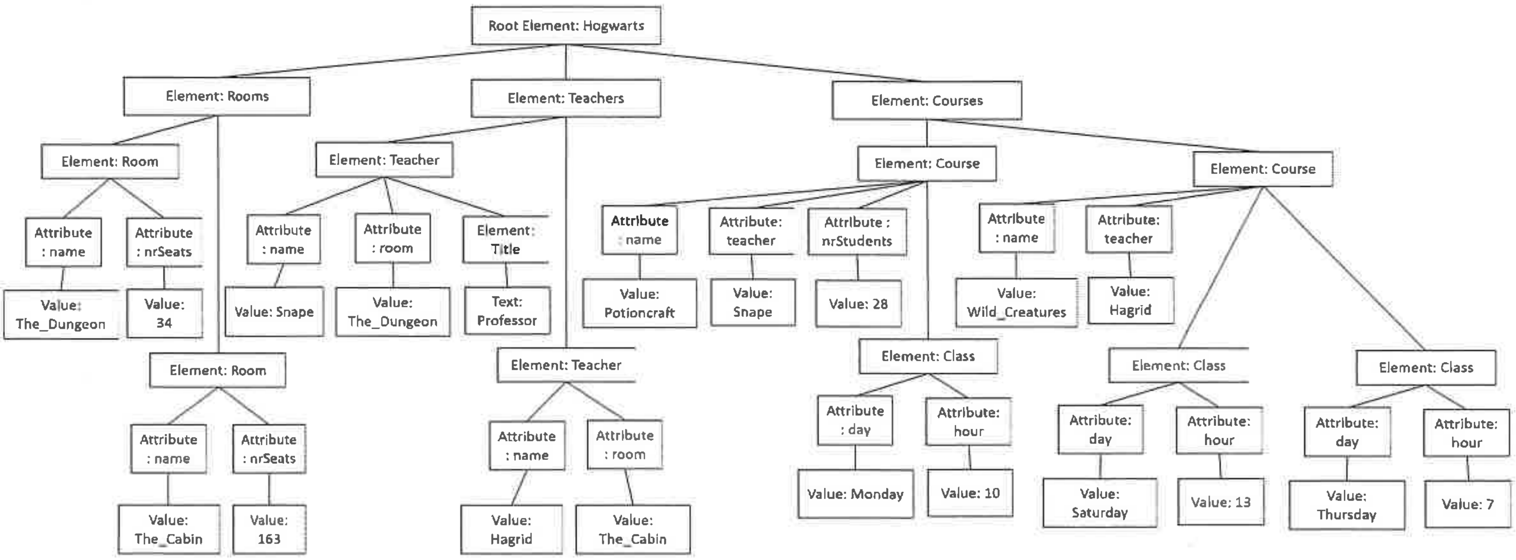
4. (a)

```
CHECK (
        NOT EXIST(
                SELECT *
                FROM PC p
                WHERE p.model NOT IN (SELECT pd.model FROM PRODUCT pd)
        )
)
```

(b)

A	B
1	8
100	0
100	0

(c)



(d)

(i)

//Course[Class/@day='Monday']

(ii)

//Room[@name=(//Teacher[@name=(//Course[Class/@day='Monday']/@teacher)]/@room)]