1. ER Diagram

1) Assumptions

- There are multiple doctors in a clinic. Each doctor can be identified using 'doctor_id'.
- Each doctor has one or more area of expertise. There can be permanent, temporary and visiting doctor across the clinic having the same area of expertise.
- Multiple patients can book an appointment but only once a day they can visit the clinic.
- Each doctor is assigned a schedule. But, multiple doctors can have the same working hours, shifts, etc. This means that schedule is a weak entity as it doesn't have a primary key.
- When a patient makes an appointment, it is not necessary that they are attended by the same doctor each time they visit the clinic.
- The doctor confirms the appointment based on their schedule and area of expertise for each patient.
- A 'diagnosis_id' is generated for documenting the diagnosis of each patient after each visit.
- The diagnosis can be of 2 types, by giving a prescription and asking the patient to do some lab tests.
- The prescription contains 'prescription_id' which is unique & has one to one relationship with diagnosis.

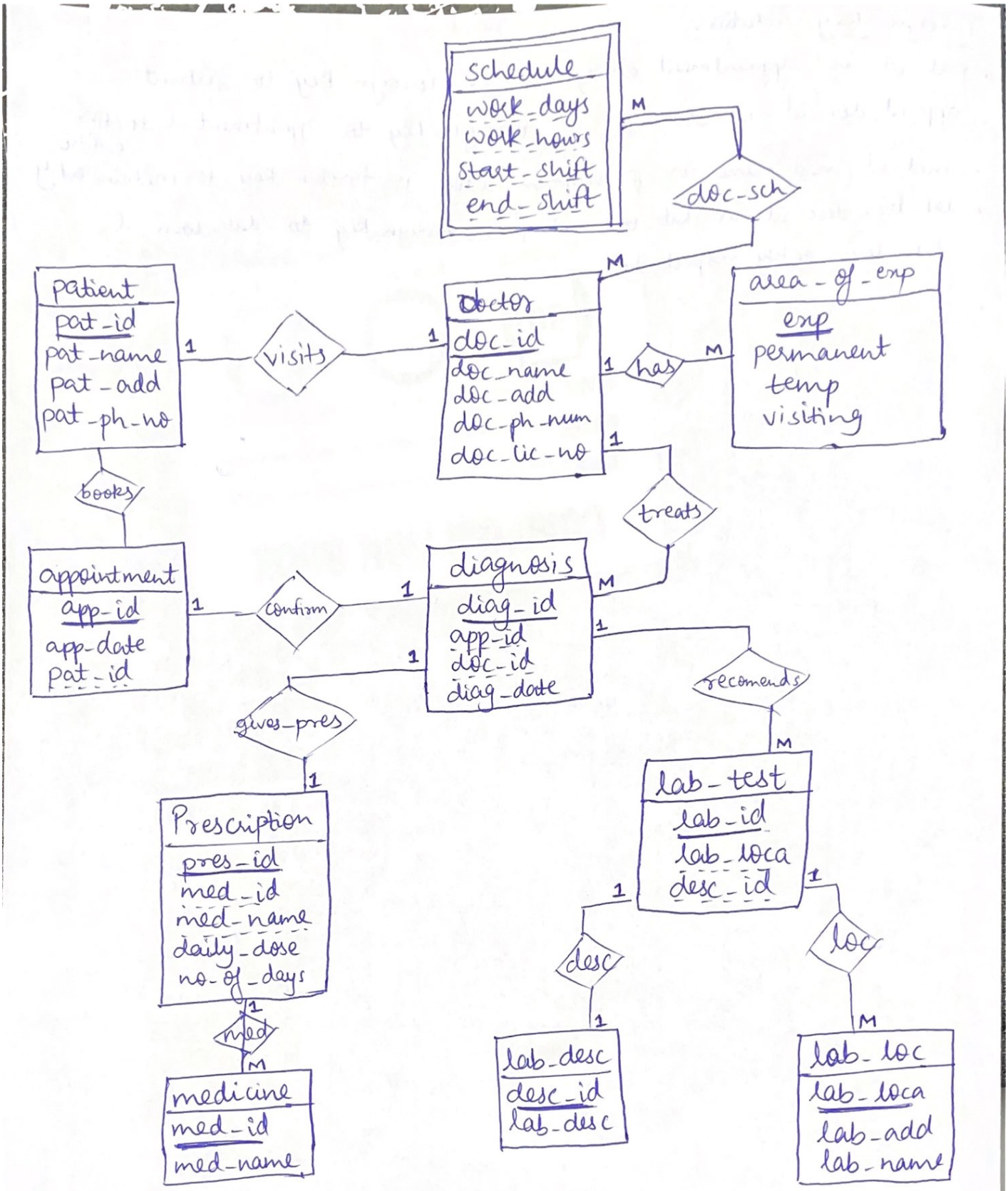
- There can be multiple lab test done at multiple location according to the earliest date available at a specific lab location.

Schemas (—— = primary key ; ----- = foreign key)

doctor(doc_id, doc-name, doc-add, doc-ph-num, doc-lic-
patient(pat-id, pat-name, pat-add, pat-ph-no)
appointment(app-id, app-date, pat-id)
schedule(work-day, work-hours, start-shift, end-shift)
diagnosis(diag-id, app-id, doc-id, diag-date)
presciption(pres-id, med-id, med-name, daily-dose, no-of-days)
medicine(med-id, med-name)
lab-test(lab-id, lab-loca, desc-id)
lab-desc(desc-id, lab-desc)
lab-loc(lab-loca, lab-add, lab-name)
area-of-exp(exp, permanent, temp, visiting)
visits(pat-id, doc-id)
confirm(app-id, diag-id, diag-date)
doc-sch(doc-id, work-hours, work-days, start-shift, end-shift)
has(doc-id, exp)
treats(doc-id, diag-id, diag-date)
gives-pres(diag-id, pres-id, med-id)
med(pres-id, med-id)
recommends(diag-id, lab-id, desc-id, lab-loca)
desc(lab-id, desc-id)
loc(lab-id, lab-loca)

Foreign Key relation

- pat_id in appointment entity is the foreign key to patient
- app-id, doc_id in diagnosis is foreign key to appointment & doctor entities.
- med-id, med_name in prescription entity is foreign key to medicine entity.
- lab_loca, desc-id in lab_test entity is foreign key to lab_loca & lab_desc entity respectively.


schedule
work_days
work_hours
start_shift
end_shift
M
doc-sch
M
patient
pat_id
pat_name
pat_add
pat_ph_no
1
visits
1
doctor
doc_id
doc_name
doc_add
doc_ph_num
doc_lic_no
1
has
M
area_of_exp
exp
permanent
temp
visiting
1
books
treats
appointment
app_id
app_date
pat_id
1
confirm
1
diagnosis
diag_id
app_id
doc_id
diag_date
M
1
1
recomends
M
gives_pres
1
Prescription
pres_id
med_id
med_name
daily_dose
no_of_days
1
med
M
medicine
med_id
med_name
lab_test
lab_id
lab_loca
desc_id
1
1
desc
loc
1
M
lab_desc
desc_id
lab_desc
lab_loc
lab_loca
lab_add
lab_name

2) Assumptions

- The customer enters the date when they want to travel as well as the arrival & departure airport.
- Multiple customers can book tickets & each customer has its own 'cust_id'.
- Similarly, each airport has it's own code making it the primary key.
- There can be multiple flight companies which have the same flight date & airport.
- Each flight company have multiple airplanes making it a many to many relationship.
- ~~Different~~ Each flight companies will have multiple flight legs reserved across airports. Each flight has its own id.
- There are various flight instances for a flight leg which stores the actual date & time of the flight
- The flight leg instance is assigned to available airport which checks the flight availability from the airport entity.
- Now, from the available airplane we have to check whether that airplane type can land at a specific airport. It becomes a many to many relationship
- At the end, the customer can select a seat of their choice based on the availability from each flight leg instance.
- Assume that the flight opperates each day of the week except sunday.

Schemas (____ = primary key, ------ = foreign key)

customer(cust_id, cust_name, cust_ph_no)

flight (flight_num, flg_comp, day_of_week)

airport(air_code, air_name, air_state, air_city)

airplane(airp_id, airp_type, no_of_seats)

flight_leg(flg_id, air_arr_code, air_dept_code, day_of_week, sch_day, sch_time, flight_num)

flight_leg_inst(air_arr_code, air_dept_code, act_arr_time, act_dept_time, act_arr_day, act_dept_day, flight_num)

airplane_type(type_id, max_seats, mfg_comp, air_code, airp_type)

books(cust_id, flight_num, date, place)

slot(airp_id, type_id, time, day)

air_dept(flight_num, airp_id)

flg_dot(air_code, flight_num, flg_id)

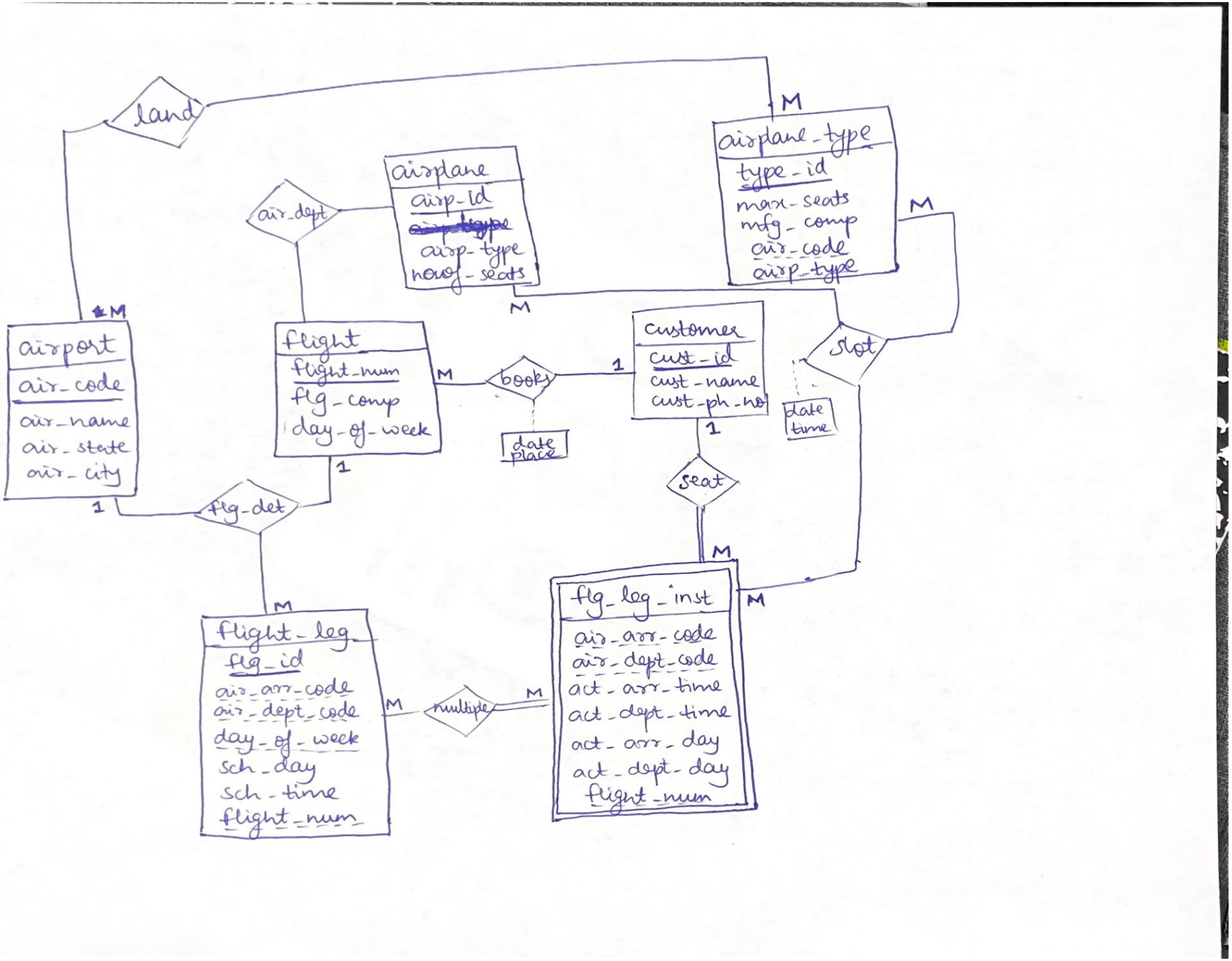
land(type_id, air_code, airp_type, air_code)

multiple(flg_id, air_arr_code, air_dept_code, act_arr_time, act_arr_day, act_dept_day, flight_num)

Foreign key relation

air_arr_code & air_dept_code

- day_of_week & flight_num is foreign key of flight_leg entity to flight entity & airport entity
- air_arr_code, air_dept_code & flight_num are foreign keys of flight_leg_inst to flight_leg entity.
- air_code & airp_type in airplane_type entity are foreign key to airport & airplane entity.


land
M
airplane_type
type_id
max_seats
mfg_comp
air_code
airp_type
M
airplane
airp_id
airp_type
nseats
air_dept
M
airport
air_code
air_name
air_state
air_city
M
1
flight
flight_num
flg_comp
day_of_week
M
book
1
customer
cust_id
cust_name
cust_ph_no
slot
date time
date place
1
1
flg_dest
seat
M
M
flight_leg
flg_id
air_arr_code
air_dept_code
day_of_week
sch_day
sch_time
flight_num
M
multiple
M
flg_leg_inst
air_arr_code
air_dept_code
act_arr_time
act_dept_time
act_arr_day
act_dept_day
flight_num
M

2. Data Normalization

1) If $X \to Y$ is a functional dependency & $X \cap Y \neq \phi$ then it is nontrivial functional dependency

$A \to B$
$C \to B$
$AC \to B$

In a relation, if attribute B is not a subset of attribute A, then it is a non-trivial dependency.

2) $F = \{C \to D, CD \to A, CE \to B, B \to ACE, E \to A\}$

a) $\{B\}^+ \to \{BACED\}$

$\{CE\}^+ \to \{CEBAD\}$

$\{B\}^+$ & $\{CE\}^+$ are the candidate key in R.

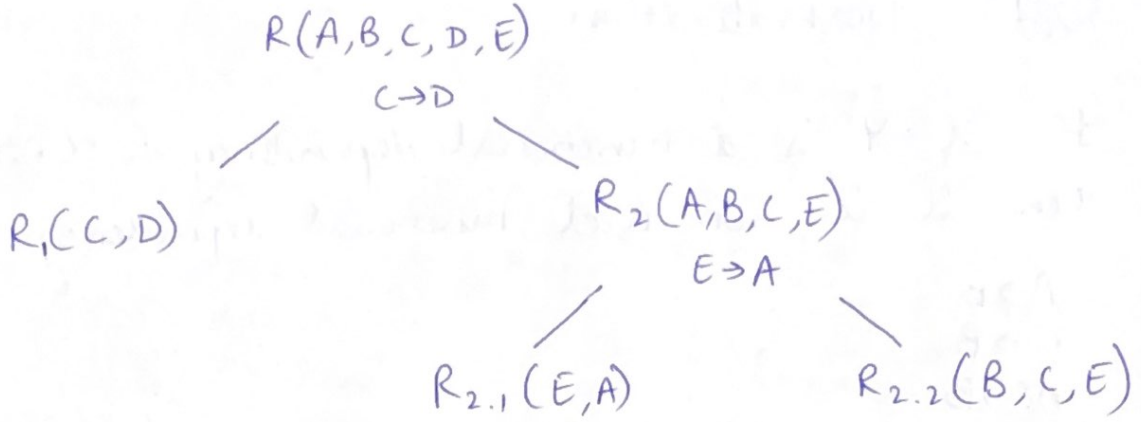
b) $R_1 = \{A, B, C\}$ & $R_2 = \{C, D, E\}$

$R_1 \cap R_2 = \{C\}^+ \to \{CDA\}$

Here, $\{C\}^+$ is neither in R_1 nor in R_2
So it is not a lossless-join.

c) $E \to A$, $C \to D$, $CD \to A$ are the BCNF violations in R.

d)



$R_1(C,D)$; $R_{2.1}(E,A)$; $R_{2.2}(B,C,E)$ are now in BCNF.

e) $CD \rightarrow A$ and $B \rightarrow ACE$ are not preserved by BCNF decomposition.

3) employee(office_num, SSN, phone, manager_name, dept_name)

office_num → phone
SSN → office_num, dept_num
dept_num → manager_name

a) $\{SSN\}^+ \rightarrow \{$ office_num, dept_num $\}$

Here, $\{SSN\}^+$ is the candidate key & can also be the super key as we can get phone & manager_name using it.

b) No, the employee relation is not in BCNF.

E(office-num, phone, SSN, dept-num, manager-num)

office-num → phone

E_1(office-num, phone)

E_2(SSN, office-num, manager-num, dept-num)

dept-num → manager-num

$E_{2.1}$(dept-num, manager-num)

$E_{2.2}$(SSN, office-num, dept-num)

E_1(office-num, phone) ; $E_{2.1}$(dept-num, manager-num) ; $E_{2.2}$(SSN, office-num, dept-num) are now in BCNF.

Primary key :-

E_1(office-num, phone) = office-num

$E_{2.1}$(dept-num, manager-num) = dept-num

$E_{2.2}$(SSN, office-num, dept-num) = SSN

Foreign key :-

~~office-num of E_1 is foreign key to $E_{2.2}$~~

~~dept-num of $E_{2.1}$ is foreign key to $E_{2.2}$~~

office-num of $E_{2.2}$ is foreign key to E_1

dept-num of $E_{2.2}$ is foreign key to $E_{2.1}$