Nisharg Gosai (002273353)
IDMP Assignment 1

Q1)1)

Assumptions,

1) A doctor must have at least one expertise
2) A doctor has only one schedule with the number of days and time when they are required to come in for clinic hours.
3) A patient can have multiple doctors on different visits.

Reducing ERD to relational schemas,

Doctor(doc_id, name, address, phone_no, license_no)

Schedule(doc_id, present_days, in_time, out_time)

Expertise(doc_id, expertise_name, experience_years)

Patient(patient_id, name, address, phone_no, last_checkup_date, next_checkup_date, disease_name, treatment_name)

consults(doc_id, patient_id, consult_date)

Medicine(med_name, med_dose, no_days)

Labtest(test_id, test_name, description, location, latest_date)

takes(patient_id, med_name)

requires(patient_id, test_id, test_name)

Q2)2)

Assumptions,

1) Airport has multiple flights and each flight can land on multiple airports.
2) Customer can make multiple bookings but each booking is unique for a customer.
3) Each flight can have only one flight leg information and each flight leg information is unique for a flight i.e no two flights have same (time + airports).
4) Each flight has a unique leg instance and each leg instance is unique for a flight.

Reducing ERD to relational schemas,

Airport(airport_code, name, state, city)

Flight(flight_no, flight_airline, days_scheduled)

landson(airport_code, flight_no)

Customer(customer_id, name, phone_no, (FK)booking_flight_no)

Booking(booking_flight_no, booking_date, seat_no)

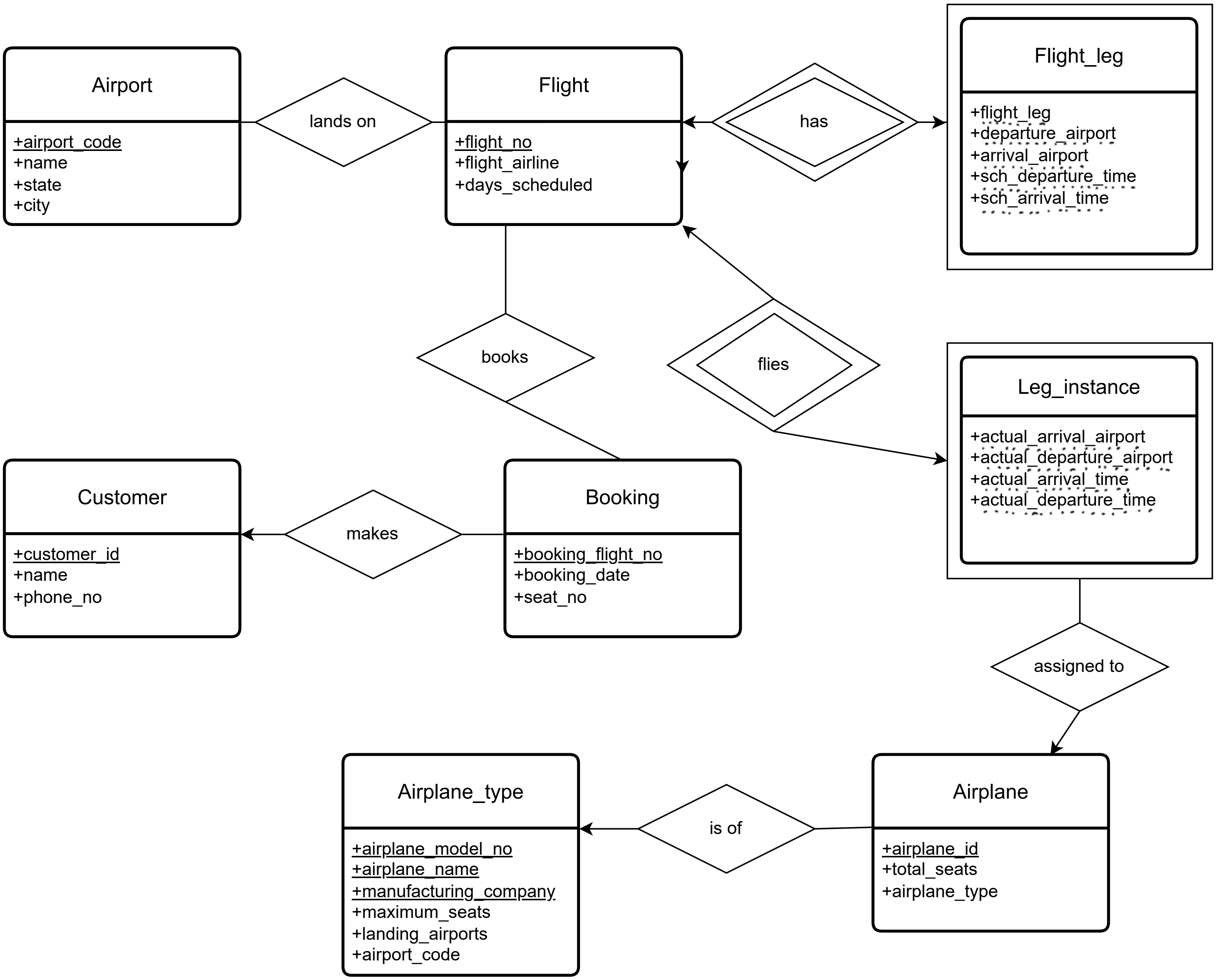
books(flight_no, booking_flight_no)

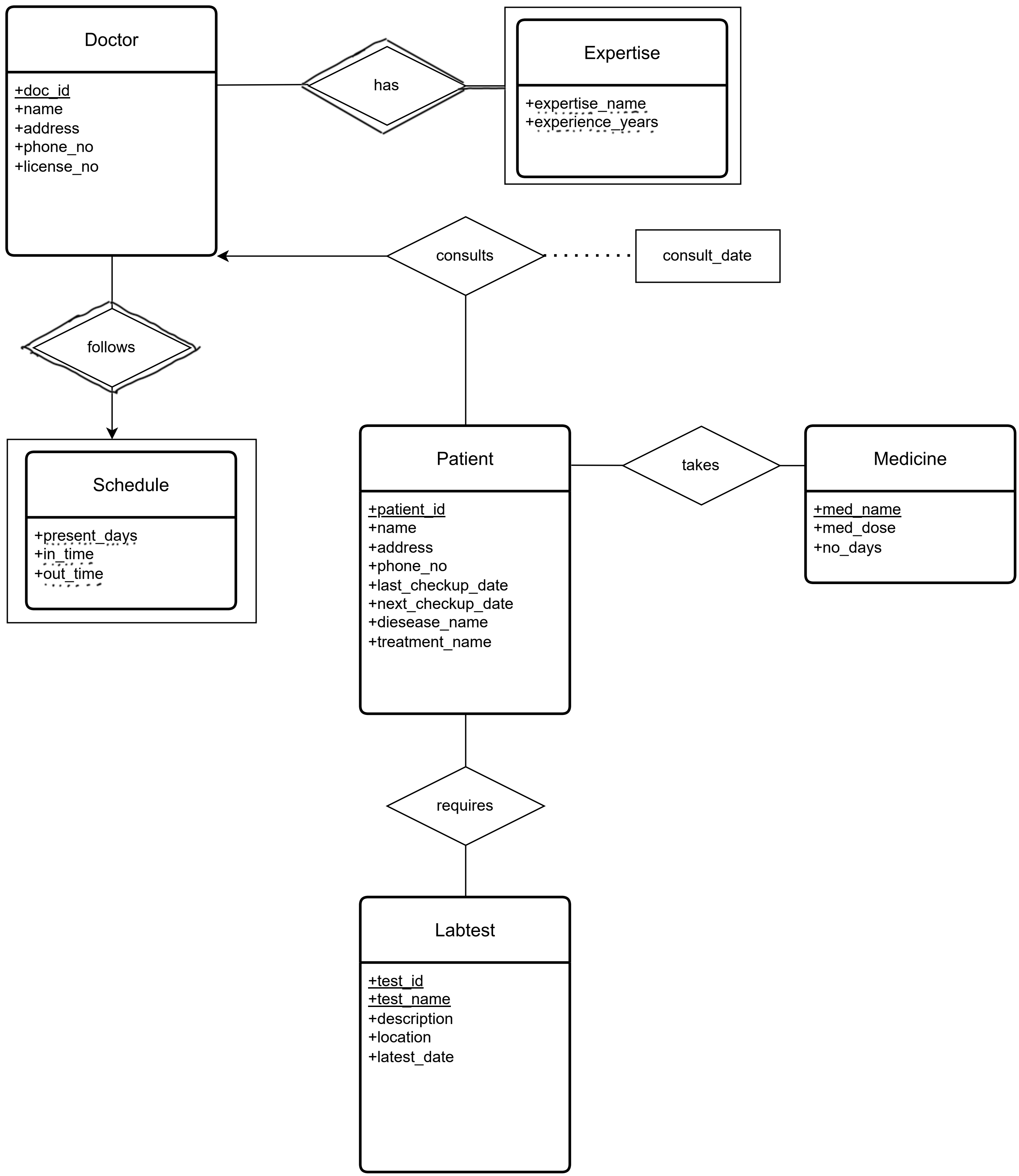
Airplane_type(airplane_model_no, airplane_name, manufacturing_company, maximum_seats, landing_airports, airport_code, (FK)airplane_id)

Airplane(airplane_id, total_seats, airplane_type)

Leg_instance(flight_no, airplane_id, flight_leg, departure_airport, arrival_airport, sch_departure_time, sch_arrival_time)

Flight_leg(flight_no, actual_arrival_airport, actual_departure_airport, actual_arrival_time, actual_departure_time)


Airport
+airport_code
+name
+state
+city
lands on
Flight
+flight_no
+flight_airline
+days_scheduled
has
Flight_leg
+flight_leg
+departure_airport
+arrival_airport
+sch_departure_time
+sch_arrival_time
books
flies
Leg_instance
+actual_arrival_airport
+actual_departure_airport
+actual_arrival_time
+actual_departure_time
Customer
+customer_id
+name
+phone_no
makes
Booking
+booking_flight_no
+booking_date
+seat_no
assigned to
Airplane_type
+airplane_model_no
+airplane_name
+manufacturing_company
+maximum_seats
+landing_airports
+airport_code
is of
Airplane
+airplane_id
+total_seats
+airplane_type



Doctor
+doc_id
+name
+address
+phone_no
+license_no
has
Expertise
+expertise_name
+experience_years
consults
consult_date
follows
Schedule
+present_days
+in_time
+out_time
Patient
+patient_id
+name
+address
+phone_no
+last_checkup_date
+next_checkup_date
+diesease_name
+treatment_name
takes
Medicine
+med_name
+med_dose
+no_days
requires
Labtest
+test_id
+test_name
+description
+location
+latest_date

NISHARG GOSAI

2. DATA NORMALIZATION

Q1) List all the nontrivial functional dependencies satisfied by the following relation R(A,B,C):

Now, to find the nontrivial functional dependencies, we need to see the definition of non-trivial functional dependencies, which is,

If X->Y is a Functional Dependency, then Y should not be a subset of X, then it is said to be a Non-trivial Functional Dependency.

A	*B*	*C*
a1	*b1*	*c1*
a1	*b1*	*c2*
a2	*b1*	*c1*
a2	*b1*	*c3*

From the above relation, let us first find all the possible combinations of functional dependencies.

A->B
A->C
B->A
B->C
C->A
C->B

A->BC
B->AC
C->AB

AB->C
BC->A
AC->B

From the above possible combinations, only A->B, C->B and AC->B are possible non-trivial function dependencies.

Q2) Consider the relational schema R(A, B, C, D, E) with the functional dependencies:
F = {C → D, CD → A, CE → B, B → ACE, E → A}

(a) Find all the candidate keys in R.

To find the candidate keys, let us start with the super key sets,
Here, (ABCDE)+ denotes the closure of (A,B,C,D,E)

(ABCDE)+ = {A,B,C,D,E) using reflexivity,

Since C->D,
(ABCE)+ = {A,B,C,E,D}

Since CE->B,
(ACE)+ = {A,C,E,B,D}

Since E->A,
(CE)+ ={C,E,A,B,D}

Let us check the subsets of (CE),
(C)+ = {C,D,A}
(E)+ = {E,A}

Therefore (CE) is a candidate key,

Further, B-> ACE, therefore B->A, B->C, B->E, using this we can write,

(B)+ = {B,A,C,E,D}

Therefore (B) is another candidate key.

So, the two candidate keys are (CE) and (B).

(b) Is the decomposition of R into R1 = (A,B,C), R2 = (C,D,E) lossless-join?

R1 = (A,B,C)
R2 = (C,D,E)

For the decomposition of R into R1 and R2 to be a lossless-join the following property must be satisfied,

R1 ∩ R2 = R1 or R1 ∩ R2 = R2

Here, R1 ∩ R2 = C, so we check closure of C,
(C)+ = {C,D,A}

Since, C cannot determine either R1 or R2; therefore it is not a lossless-join decomposition.

(c) Indicate all BCNF violations in R.

The candidate keys of the given relation are B and CE,
Therefore any FD with the super keys not on the left side violates BCNF,

C->D (violates BCNF)
CD->A (violates BCNF)
CE->B
B->ACE
E->A (violates BCNF)

(d) Decompose R into a set of relations that are in BCNF.

Since C->D violates BCNF we decompose on this fd,

R(ABCDE) will decompose into R1(CD) and R2(ABCE)

R1(CD), will have functional dependency C->D
R2(ABCE), will have functional dependencies CE->B, B->ACE, E->A

Since CD-> A is lost and E->A is a BCNF violation so we decompose on E->A,

R2(ABCE) will decompose into R3(EA) and R4(BCE)

R3(EA) will have functional dependency E->A
R4(BCE) will have functional dependencies B->C, B->E, CE->B

Therefore, after decompositions, we will have three relations R1(CD), R3(EA) and R4(BCE) which will be in BCNF.

(e) Indicate which dependencies, if any, are not preserved by the BCNF decomposition.

After BCNF decomposition, the functional dependencies CD->A and B->A are not preserved.

Q3) A store maintains the following relation for its employee records:

employee (office_num, SSN, phone, manager_name, dept_num)

Given the following functional dependencies:

office_num → phone
SSN → office_num, dept_num
dept_num → manager_name

(a) List the candidate keys in the employee relation.

To find candidate keys, let us start with finding the closure of the superkeys and reducing the attributes using the functional dependencies as we move forward,

Let us take,

office_num = A
SSN = B
phone = C
manager_name = D
dept_num = E

Therefore, the Functional Dependencies will be,

A->C, B->AE, E->D

(ABCDE)+ = {A,B,C,D,E} (The plus sign denotes the closure),

Given, A->C,
(ABDE)+ = {A,B,D,E,C}

Given, B->AE we can write it as, B->A and B->E,
(BD)+ = {B,D,A,E,C}

Given, B->E and E->D,
(B)+ = {B,A,E,D,C}

Therefore, (B) is a candidate key.

Since, B is not present on R.H.S of any functional dependencies, No attribute will determine B and therefore B will always be a part of the set of candidate key and since (B)+ is a candidate key there will be no other candidate key.

Therefore, B is a candidate key and B= SSN, SSN is the only candidate key.

(b) Is the employee relation in BCNF? If so, write "Yes" below. Otherwise, decompose it into BCNF and specify all the primary and foreign keys in the final relations.

Let us take,

office_num = A
SSN = B
phone = C
manager_name = D
dept_num = E

Therefore, the Functional Dependencies will be,

A->C, B->AE, E->D

A->C and E->D are in BCNF violations.

Decomposing on A->C, we get two relations R1(AC) and R2(ABDE), with R1 having functional dependency A->C and R2 having functional dependencies B->AE and E->D.

Since R2(ABDE) is still in BCNF violation, we decompose further on E->D

Decomposing, we get R3(ED) and R4(ABE), with R3 having functional dependency E->D and R4 having functional dependency B->AE

Since there are no more BCNF violations, we are left with three relations, R1(AC), R3(ED) and R4(ABE)

So the final relations with primary and foreign keys are:

R1(<u>office_num(FK), phone</u>)
{R1.office_num is a foreign key which references R4.office_num }
{office_num,phone} is the composite primary key in R1

R3(<u>dept_num</u>(FK), manager_name)
{R3.dept_num is a foreign key which references R4.dept_num}
{dept_num} is the primary key in R3

R4(office_num, <u>SSN</u>, dept_num)
{SSN} is the primary key in R4