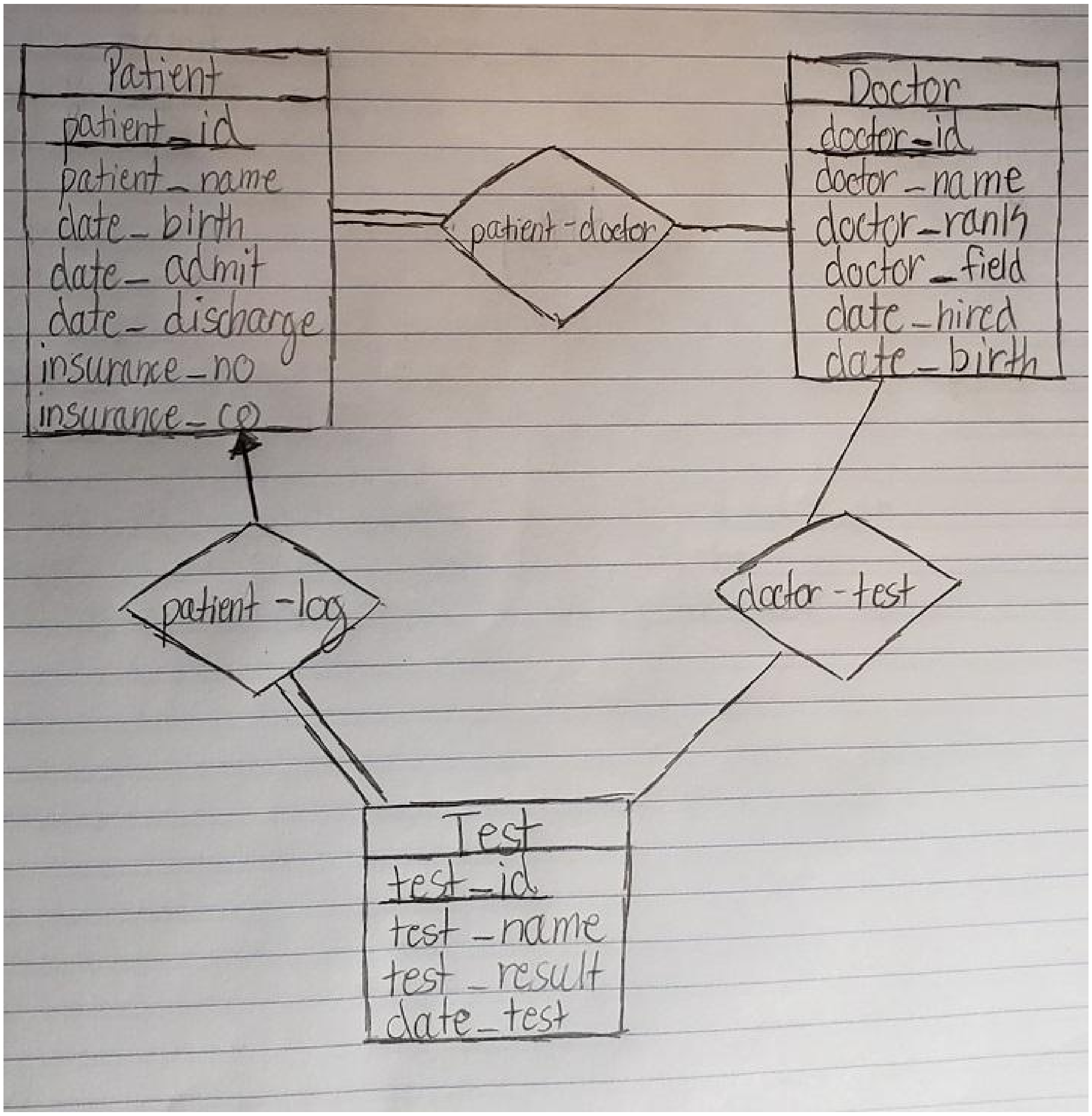
Jackie Diep

6.15)



(Not sure if this is entirely visible, but I had trouble finding a way to make a diagram online)

6.18)

a) Change the name of the variable to be a different, more identifiable description

b)

If X is the primary key for A but not B, make the primary key of A "X" into a foreign key in B.

If X is the primary key for both A and B, combine the two entity sets into one entity set.

Jackie Diep

If X is not the primary key for neither A nor B, make one into the foreign key of another. This is only possible if the one in A is given a unique constraint and made into a foreign key of B or vice-versa.

6.20)

Relation Schema (foreign keys for relations in italics)

a) 6.1)
customer(customer_id, customer_name, date_birth, customer_address)
car(car_id, year, manufacturer, model)
accident(accident_id, date_accident, address_accident, other_insurance)
policy(policy_id, car_count)
premium(premium_id, premium_cost, premium_date, premium_received)
car-owner(*customer_id*, *car_id*)
accident-car(*car_id*, *accident_id*)
policy-holder(*policy_id*, *customer_id*)
policy-car(*policy_id*, *car_id*)
policy-premium(*policy_id*, *premium_id*)

b) 6.2)
student(student_id, name, address, date_birth, date_enroll)
exam(exam_id, exam_name, exam_date)
section(section_id, section_semester, section_capacity)
course(course_id, course_name)
department(department_id)
enrolled(*student_id*, *section_id*)
course-section(*course_id*, *section_id*)
dept-course(*department_id*, *course_id*)
marks(*student_id*, *exam_id*, *course_id*, *section_id*, grade)

c) 6.3)
team(team_id, team_name, team_colors, team_capacity)
match(match_id, match_name, match_date, match_location, team_score,
enemy_score)
player(player_id, player_name, date_birth, player_position)
teammates(*team_id*, *player_id*)
match-teams(*match_id*, *team_id*, enemy_id)
match-player(*match_id*, *player_id*, individual_score)

d) 6.15)
patient(patient_id, patient_name, date_birth, date_admit, date_discharge, insurance_no,
insurance_co)
doctor(doctor_id, doctor_name, doctor_rank, doctor_field, date_hired, date_birth)
test(test_id, test_name, test_result, date_test)
patient-doctor(*patient_id*, *doctor_id*)

Jackie Diep

patient-log(*patient_id*, *test_id*)
doctor-test(*doctor_id*, *test_id*)

Jackie Diep

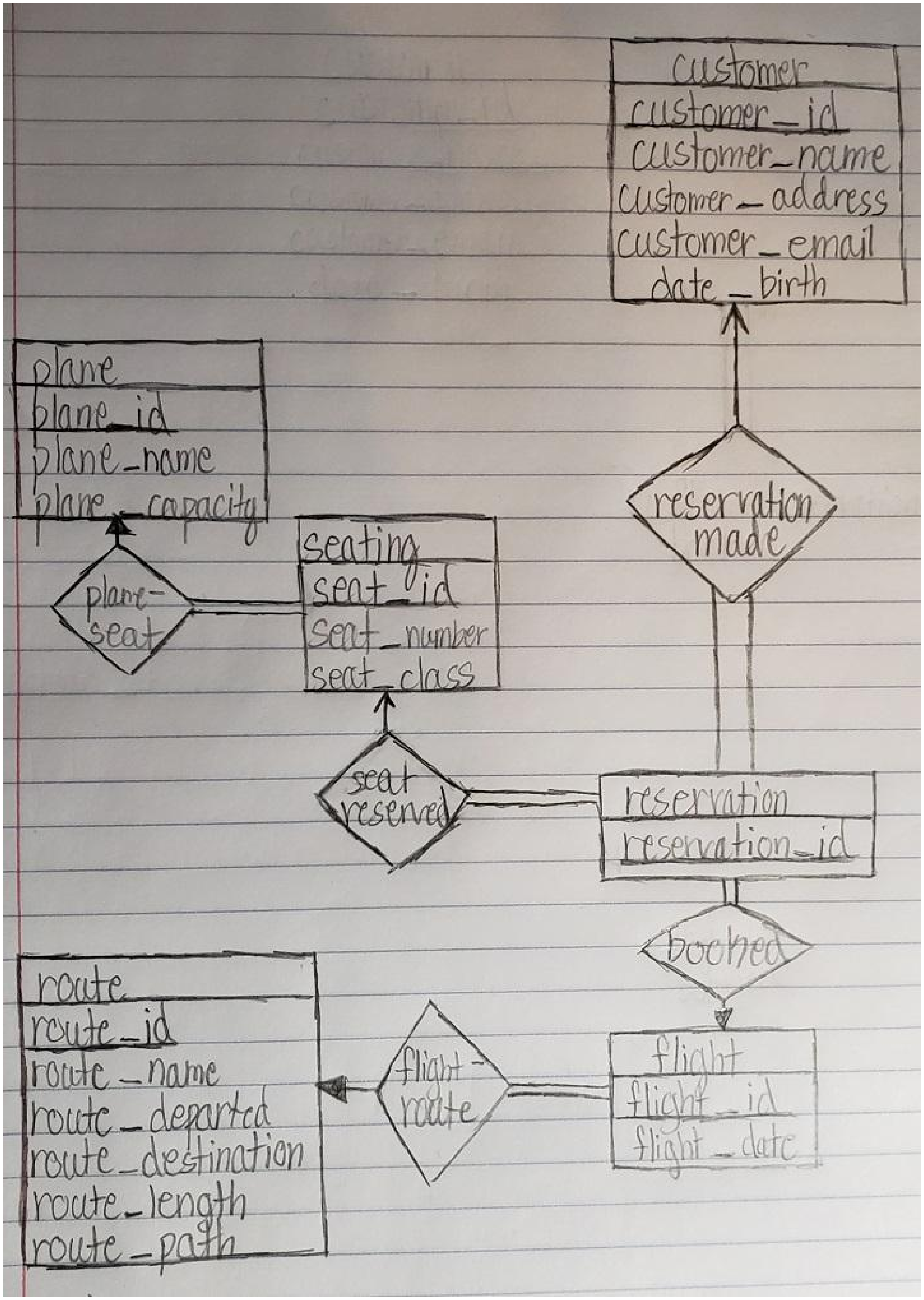
6.24)

customer(<u>customer_id</u>, customer_name, customer_address, date_birth, customer_email)
plane(<u>plane_id</u>, plane_name, plane_capacity)
route(<u>route_id</u>, route_name, route_departed, route_destination, route_length, route_path)
seating(<u>seat_id</u>, *plane_id*, seat_number, seat_class)
flight(<u>flight_id</u>, *plane_id*, *route_id*, flight_date)
reservation(<u>reservation_id</u>, *customer_id*, *flight_id, seat_id*)

Constraints:
Every reservation has exactly one customer
Every reservation has exactly one seat
Every reservation has exactly one flight
Every seat has exactly one plane
Every flight has exactly one route

My database considers each departure as one flight. For example, a flight to Asia and a flight back to America are considered two separate flights that require two separate reservations. Also, each reserved seat is considered a separate reservation with exactly one customer.

Jackie Diep


customer
customer_id
customer_name
customer_address
customer_email
date_birth
plane
plane_id
plane_name
plane_capacity
plane-seat
seating
seat_id
seat_number
seat_class
reservation made
seat reserved
reservation
reservation_id
booked
route
route_id
route_name
route_departed
route_destination
route_length
route_path
flight-route
flight
flight_id
flight_date