I apologize for not formatting the warm ups. This homework caused a lot of stress for me. The graded problems are formatted nicely and legibly.

PROBLEM 1

A)

T | T in Tree and
exists T1 in Tree(
for all T2 in tree (
T2[years_to_maturity] ≤ T1[year_to_maturity]))
and T[species name] = T1[species name]

B)

N | N in Nursery and
Exists T in Tree(
T[height > 9] and
Exists S in Sells (
S[t_id] = T[t_id] and
Exists N1 in Nursery(N1[n_id] = S[n_id])))
and N[n_name] = N1[n_name]

C)

N | N in Nursery and
exists T in Tree(
T[height > 9] and
Exists S in Sells(
S[t_id] = T[t_id]) and
Exists N1 in Nursery(
N1[n_id] = S[n_id]) and
N[n_id] = N1[n_id] and
N1[state] = California and
S[price > 20]

D)

N_id_pairs | exists S in Sells (
exists S1 in Sells(
S[t_id] = S1[t_id] and S[price] < S1[price])) and
N_id_pairs[first_n_id] = S[n_id] and N_id_pairs[second_n_id] = S1[n_id]

E)

a) all tree id and specie names available at the nursery with name “Johnny Appleseed”
b) all names and ids of nurseries that only have “best-priced” items, that is there is no tree they sell that you can get cheaper at another nursery.

PROBLEM 2

A) A query is safe if it has a finite response. This entails a procedure that constrains to tuples that exist. Because if it is unconstrained, and there is no set of answers, there are infinite answers! And the computer can't do that.
B) 1. Safe, this will return all t in shrub, or none if there are none, the first part does nothing
2. Unsafe, the first half is infinite so this returns infinite response… and more!

PROBLEM 3
A
1. Person(id, name(first, last), age, address(street_address, state, zip_code, country), {phone})
2. Gardener(Person(…), year_of_experience, {cultivates})
3. Cultivates(Gardner(…), tree(...))
4. Tree(species_type, maximum_height, years_to_maturity)
5. Phone(person(...), number)
6. Middle_name(Person(...), middle_name)

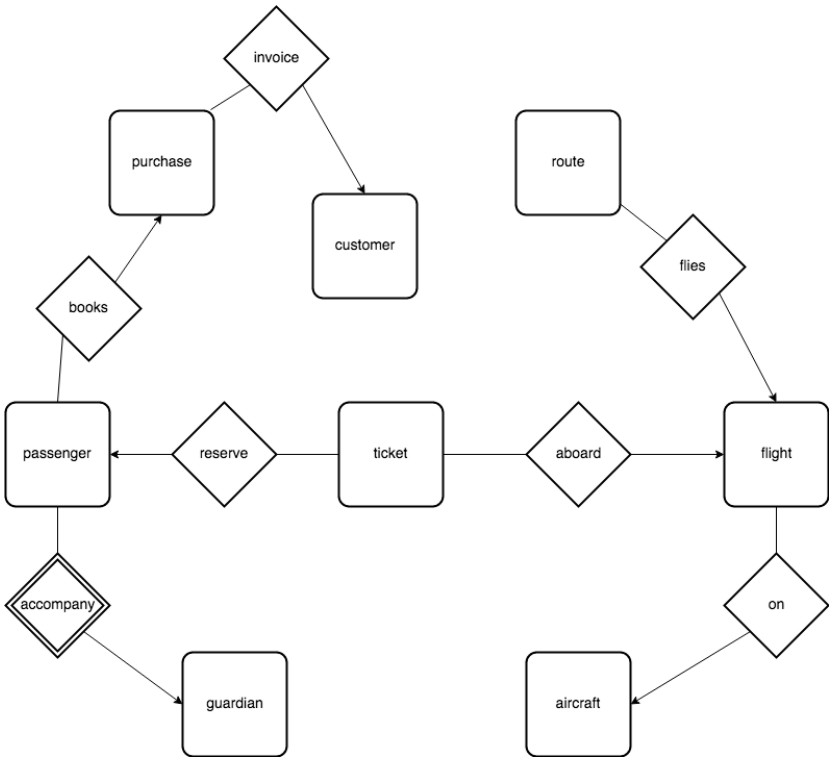
A
1. Person(id, name(first, middle, last), age, address(street_address, state, zip_code, country), phone(primary_number, alternate_number))
2. Gardener(id, years_of_experience)
3. Tree(species_type, maximum_height, years_to_maturity)
B
1. Cultivates relates gardeners to the trees they cultivate
2. Is relates people and gardener
C - pictoral
D
1. Person(ID, Name(first, middle, last), Age, Adress(Street_address, state, zip, country), phone(primary, secondary))
2. Gardener(years experience)
3. Cultivates(id, species type)
4. Tree(species_type, max height, years to maturity)

GRADED PROBLEMS

FOUR

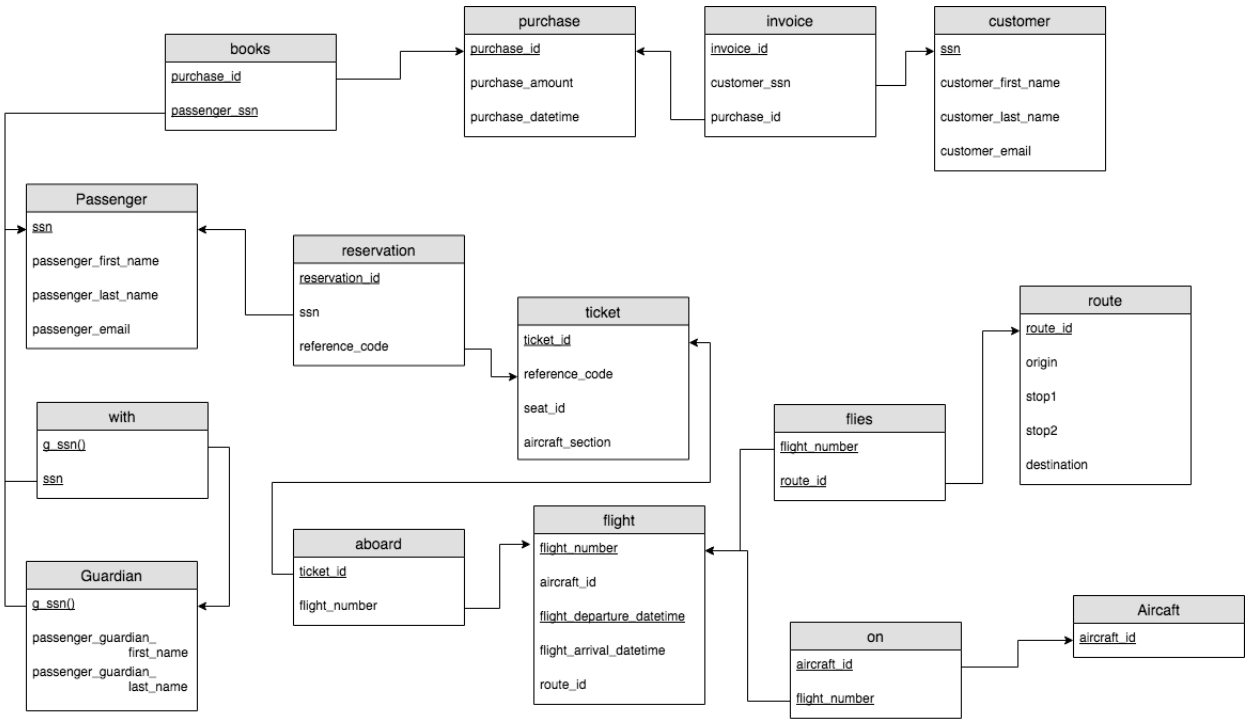
1. customer(ssn, cutsomer_first_name, customer_last_name, customer_email)
   passenger(ssn, passenger_first_name, passenger_last_name, passenger_email, passenger_minor)
   guardian(g_ssn(), passenger_guardian_first_name, passenger_guardian_last_name)
   purchase(customer_email, purchase_datetime, purchase_amount)
   seat(reference_code, aircraft_id, seat_id, aircraft_section)
   ticket(reference_code, flight_number, flight_departure_datetime, flight_arrival_datetime, flight_route)
   route(route_id, origin, stop1, stop2, destination)

   I had originally considered including a plane, but it seems that the plane itself isn't important as much as the seats on it. I included an SSN attribute as a primary key to make cross referencing easier. I also separated the guardian entity since it seems to be an edge (in the sense that the majority of passengers don't needs one).



2. The reserve relation is many to one because people can have many tickets and a ticket is only for one person. The aboard relation is many to one because there are multiple

tickets on the same flight and a ticket can't be for multiple flights. Flights occur on aircrafts, which is represented with the on relation; it is many to one because multiple flights may use the same aircraft and no one flight occurs on multiple aircrafts. Flies is a many to one relation because one flight can have multiple route available on it, and routes cannot occur over multiple flights. Back on the person side of the ticket, there is a weak entity many to one relation titled "with" between passenger and guardian. This is because one guardian may be a chaperone to multiple minors. The books relation is many to one since group purchases can be made. Invoice is a many to one relation because one customer can make multiple purchase, and every time a purchase is made only one customer is responsible for it.



3. The invoice relation has a primary key called invoice_id, this is because I think it is worth being able to reference the invoice with a single key as opposed to multi attribute key since it is a relation that may be referenced on its own; the relation represents a customer's purchase. The books relation has a multi attribute foreign key from purchase and passenger. The reservation relation represents a person's ticket reservation; I made a reservation id as a key to account for the fact that someone can have multiple tickets (would have implied a multi attribute key). The with relation represent the presence of a guardian when necessary. I made the multivariate SSN attribute to account for the fact that a guardian can be chaperoning multiple children. The aboard relation uses ticket_id

as a primary key. The on relation describes which flights are being made on an aircraft. The flies relation connects a flight number and the routes it covers. I gave all of these multi value primary keys. Additionally, the reference code attribute from ticket is only assigned once a reservation is made. If the ticket is unreserved the reference code is null.

FIVE:

1. $T1 \leftarrow \sigma_{9/23/19 < flight\_departure\_datetime}(\sigma_{9/24/19 > flight\_departure\_datetime}(flight)) \bowtie flies \bowtie (\sigma_{destination=LAX}(\sigma_{origin=PVD}(\sigma_{stop2=null}(route))))$
   $T2 \leftarrow \sigma_{aircraft\_section=business}(ticket) \bowtie aboard$
   $A \leftarrow \pi_{flight\_number}(T1 \bowtie T2)$

2. $T1 \leftarrow \pi_{flight\_number}(\sigma_{flight\_departure\_datetime > current\_time}(flight) \bowtie flies \bowtie \sigma_{destination=passenger\text{-}destination}(\sigma_{origin=current\text{-}location}(route)))$
   $T2 \leftarrow ticket \bowtie aboard \bowtie T1$
   $A \leftarrow \pi_{flight\_number}(\sigma_{tickets\_available>2}(flight\_number\ g\ count(reference\_code=null)\ as\ tickets\_available(T2)))$

1. $F \mid F \in flight(flight\_departure\_datetime > 9/23 \wedge flight\_departure\_datetime < 9/24)$
   $\wedge \exists\ t \in ticket(aircraft\_section = business)$
   $\wedge \exists\ a \in aboard(ticket\_id = t.ticket\_id \wedge flight\_number = f.flight\_number)$
   $\wedge \exists\ r \in route(destination = LAX \wedge origin = PVD \wedge stop2 = null)$
   $\wedge \exists\ f \in flies(flight\_number = a.flight\_number \wedge route\_id = r.route\_id)$
   $\wedge F.flight\_number = f.flight\_number$

2. $F \mid F \in flight(flight\_departure\_datetime > current\_time)$
   $\wedge \exists\ r \in route(destination = passenger\text{-}destination \wedge origin = current\text{-}location)$
   $\wedge \exists\ f \in flies(route\_id = r.route\_id)$
   $\wedge \exists\ a1, a2, a3 \in aboard(flight\_number = f.flight\_number)$
   $\wedge a1.ticket\_id \neq a2.ticket\_id$
   $\wedge a2.ticket\_id \neq a3.ticket\_id$
   $\wedge a3.ticket\_id \neq a1.ticket\_id$
   $\wedge \exists\ t1, t2, t3 \in ticket(reference\_code = null)$
   $\wedge t1.ticket\_id = a1.ticket\_id$
   $\wedge t2.ticket\_id = a2.ticket\_id$
   $\wedge t3.ticket\_id = a3.ticket\_id$
   $\wedge\ F.flight\_number = f.flight\_number$