Describe the relations that would be produced by the following relational algebra operations:

1. This will produce a relation with a single attribute (hotelNo) giving the number of those hotels with a room price greater than 50.
2. This will produce a join of the Hotel and Room relations containing all the attributes of both Hotel and Room (there will be two copies of the hotelNo attribute). Essentially this will produce a relation containing all rooms of the hotels.
3. This will produce a join of Hotel and those tuples of Room with a price greater than 50. Essentially this will produce a relation containing all hotel names with a room price above 50.
4. This will produce a (left outer) join of Guest and those tuples of Booking with an end to dare (dateTo) greater than or equal to 1-Jan-2002. All guests who dont have a booking with such a date will still be included in the join. Essentioally this will produce a relation containing wall guests and show the details of any booking they have beyond 1-Jan-2002.
5. This will produce a (semi) join of Hotel and those tuples of Room with a price greater than 50. Only those Hotel attributes will be listed. Essentially this will produce a relation containing all the details of all hotels with a room price above 50.
6. This will produce a relation containing the names of all guests who have booked all hotels in London.

Describe the relations that would be produced by the following tuple relational calculus expressions:

1. This will produce a relation containing the names of all hotels in London.
2. This will produce a relation containing the names of all hotels that have a room price above 50.
3. This will produce a relation containing the names of all hotels that have a booking for a guest called John Smith.

Generate the relational algebra, tuple relational calculus, and domain relational calculus expressions for the following queries:

1. RA: Hotel

TRC: {H | Hotel(H)}

DRC: {hotelNo, hotelName, city | Hotel (hotelNo, hotelName, city)}

2. RA: σtype=‘S’ ∧ price < 20(Room)

TRC: {R | Room(R) Ù R.type = ‘S’ Ù R.price < 20}

DRC: {roomNo, hotelNo, type, price | (Room(roomNo, hotelNo, type, price) Ù

type = ‘S’ Ù price < 20)}

3. RA: ΠguestName, guestAddress(Guest)

TRC: {G.guestName, G.guestAddress | Guest(G)}

DRC: {guestName, guestAddress | (∃guestNo) (Guest(guestNo, guestName, guestAddress))}

4. RA: Πprice, type(Room 3 hotelNo (σhotelName = ‘Grosvenor Hotel’(Hotel)))

TRC: {R.price, R.type | Room(R) ∧ (∃H) (Hotel(H) ∧ (R.hotelNo = H.hotelNo) Ù

(H.hotelName = ‘Grosvenor Hotel’))}

DRC: {price, type | (∃roomNo, hotelNo, hotelNo1, hotelName, city)

(Room(roomNo, hotelNo, type, price) ∧ Hotel(hotelNo1, hotelName, city) Ù

(hotelNo = hotelNo1) ∧ (hotelName = ‘Grosvenor Hotel’))}

5.

RA:Guest 3 guestNo (OdateFrom<= '01-01-2002'^dateTo>='01-01-2002'(Booking 3 hotelNo(OhotelName='Grosvenor Hotel'(Hotel)))

TRC: {G |Guest(G) Ù ((∃B) ( ∃H) (Booking(B) ∧ Hotel(H) ∧ (B.dateFrom ≤ ‘01-01-02’) Ù

(B.dateTo ≥ ‘01-01-02’) ∧ (B.guestNo = G.guestNo) Ù

(B.hotelNo = H.hotelNo) ∧ (H.hotelName = ‘Grosvenor Hotel’)))}

DRC: {guestNo, guestName, guestAddress | (∃hotelNo, guestNo1, dateFrom, dateTo,

hotelNo1, hotelName, city)

(Guest(guestNo, guestName, guestAddress) ∧

Booking(hotelNo, guestNo1, dateFrom, dateTo) ∧

Hotel(hotelNo1, hotelName, city) Ù (guestNo = guestNo1) ∧

(dateFrom ≤ ‘01-01-02’ Ù dateTo ≥ ‘01-01-02’) ∧

(hotelNo = hotelNo1) Ù (hotelName = ‘Grosvenor Hotel’))}

6.

RA: (Room 3 hotelNo (σhotelName = ‘Grosvenor Hotel’(Hotel))) 5 // Outer Join

ΠguestName, hotelNo, roomNo(

(Guest 3 guestNo (σdateFrom ≤ ‘01-01-02’ ∧ dateTo ≥ ‘01-01-02’ (

Booking 3 hotelNo (σhotelName=‘Grosvenor Hotel’(Hotel))))

(substitute ‘01-01-02’ for today’s date).

TRC: {R, G.guestName | (Room(R) Ù ($H)(Hotel(H) ∧

(R.hotelNo = H.hotelNo) ∧ (H.hotelName = ‘Grosvenor Hotel’))) ∨

(Guest(G) ∧ (($B)($H) (Booking(B) ∧ Hotel(H) ∧

(G.guestNo = B.guestNo) ∧ (B.hotelNo = H.hotelNo) ∧

(H.hotelName = ‘Grosvenor Hotel’) ∧

(B.dateFrom ≤ ‘01-01-02’ Ù B.dateTo ≥ ‘01-01-02’)))}

DRC: {roomNo, hotelNo, type, price, guestName |

($hNo, hName, city, hNo1, gNo1, dFrom, dTo, rNo)

(Room(roomNo, hotelNo, type, price) Ù Hotel(hNo1, hName, city) Ù

(hotelNo = hNo1) Ù (hName = ‘Grosvenor Hotel’) ) ∨

(Guest(guestNo, guestName, guestAddress) Ù Hotel(hNo, hName, city) Ù

Booking(hNo1, gNo1, dFrom, dTo, rNo) Ù

(guestNo = gNo1) Ù (hNo1 = hNo) Ù (hName = ‘Grosvenor Hotel’) Ù

(dFrom ≤ ‘01-01-02’ Ù dTo ≥ ‘01-01-02’)))}

7.

RA: ΠguestNo, guestName, guestAddress(Guest 3 guestNo (σdateFrom ≤ ‘01-01-02’ ∧ dateTo ≥ ‘01-01-02’ (

Booking 3 hotelNo (σhotelName=‘Grosvenor Hotel’(Hotel)))))

(substitute ‘01-01-02’ for today’s date).

TRC: {G | Guest(G) Ù (($B) ($H) (Booking(B) Ù Hotel(H) Ù (B.guestNo = G.guestNo) Ù

(B.hotelNo = H.hotelNo) Ù (H.hotelName = ‘Grosvenor Hotel’) Ù

(B.dateFrom ≤ ‘01-01-02’ Ù B.dateTo ≥ ‘01-01-02’) ))}

DRC: {guestNo, guestName, guestAddress |

(($hNo, gNo, dFrom, dTo, rNo, hNo1, hName, city)

(Guest (guestNo, guestName, guestAddress) Ù

Booking (hNo, gNo, dFrom, dTo, rNo) Ù Hotel (hNo1, hName, city) Ù

(guestNo = gNo) Ù (hNo = hNo1) Ù (hName = ‘Grosvenor Hotel’) Ù

(dFrom ≤ ‘01-01-02’ Ù dTo ≥ ‘01-01-02’) ))}