NYU Tandon School of Engineering
Computer Science and Engineering
CS 6083, Spring 2019

# Problem Set # 1 Sample Solution

Problem 1

a) (Car.carMake, Car.carModel, Car.carModelNum) is a foreign key reference to (CarType.carMake, CarType.carModel, Car.carModelNum)
Rental.cID is a foreign key reference to Customer.cID
Rental.carID is a foreign key reference to Car.carID
Rental.pickBid is a foreign key reference to Branch.bid
Rental.returnBid is a foreign key reference to Branch.bid

b) (i)
```
Select distinct Customer.cID, cName
From Customer Natural Join Rental Natural Join Car Natural Join CarType
Where CarType.carModel = 'Toyota' and carColor = 'blue' and carSeats = 6 and
year(pickupTD) = 2017
```

(ii)
```
(Select distinct cID From Rental Natural Join Car Where carModel = 'Toyota')
Except
(Select distinct cID From Rental Natural Join Car Where carModel = 'Audi)
```

(iii)
```
Select carID, count(rID) as Times
From Rental
Group by carID
```

(iv)
```
Create temporary table t1
Select count(distinct bState) as totalNumofStates
From Branch
Create temporary table t2
Select cID, count(distinct bState) as numofStates
From Rental Natural Join Branch
Group by cID
Select cID,
From t1, t2
Where numofStates = totalNumofStates
Drop t1, t2
```

(v)

```
Create temporary table t1
Select cID, sum(cost) as totalCost,
From Rental
Where year(pickupTD) = 2017
Group by cID
Create temporary table t2
Select max(totalCost) as maxCost
From t1
Select Customer.cID, cName
From t1, t2, Customer
Where Customer.cID = t1.cID and totalCost = maxCost
Drop table t1, t2
```

(vi)

```
Create temporary table t1
Select max(cost) as maxCost
From Rental
Where year(pickupTD) = 2017
Select Customer.cID, cName
From t1, Rental, Customer
Where Customer.cID = Rental.cID and cost = maxCost
Drop t1
```

(c)

(i) $\Pi_{\text{customer.CID, cName}}(\sigma_{\text{carType.carModel = 'Toyota' } \wedge \text{ carColor = 'blue' } \wedge \text{ CarSeats = 6 } \wedge \text{ year(pickupTD)=2017}}$
(Customer ⋈ Car ⋈ CarType ⋈ Rental)

(ii) $\Pi_{\text{distinct CID}}(\sigma_{\text{carModel = 'Toyota'}}(\text{Rental} \bowtie \text{Car})) - \Pi_{\text{distinct CID}}(\sigma_{\text{carModel='Audi'}}(\text{Rental} \bowtie \text{Car}))$

(iii) $_{\text{carID}}\mathcal{G}_{\text{count(rID) as times}}(\text{Rental})$

(iv) $\left(\Pi_{\text{cid, bstate}}\ \text{Customer} \bowtie \text{Rental} \bowtie_{\text{brid = pickupbid}} \text{Branch}\right) \div \left(\Pi_{\text{bstate}}\ \text{Branch}\right)$

(v) $t_1 \leftarrow {}_{\text{CID}}\mathcal{G}_{\text{sum(cost) as totalCost}}(\sigma_{\text{year(pickTD)=2017}}(\text{Rental}))$
$t_2 \leftarrow \mathcal{G}_{\text{max(totalCost) as maxCost}}(t_1)$
$\Pi_{\text{customer.CID, cName}}(\sigma_{\text{customer.CID = } t_1\text{.CID } \wedge \text{ totalCost = maxCost}}(t_1 \times t_2 \times \text{Customer}))$

(vi) $T_1 \leftarrow \mathcal{G}_{\text{max(cost) as maxCost}}(\sigma_{\text{year(pickTD)=2017}}(\text{Rental}))$
$\Pi_{\text{customer.CID, cName}}(\sigma_{\text{customer.CID = Rental.cID } \wedge \text{ cost = maxCost}}(T_1 \times \text{Rental} \times \text{Customer}))$

**(d)**

I.

$\{ res \mid \exists u \in Customer (res[cID] = u[cID] \wedge res[cName] = u[cName] \wedge r \in Rental(r[cID] = u[cID] \wedge r[pickupTD].year = 2017 \wedge \exists c \in Car (c[carID] = r[carID] \wedge c[carMake] = Toyota \wedge \exists t \in CarType(c[carMake] = t[carMake] \wedge c[carModel] = t[carModel] \wedge c[carModelNum] = t[carModelNum] \wedge t[carSeats] = 7 )))) \}$

II.

$\{ res \mid \exists tr \in Rental (res[cID] = tr[cID] \wedge \exists t \in Car(t[carMake] = Toyota \wedge tr[carID] = t[carID] )) \wedge \neg\exists ar \in Rental (res[cID] = ar[cID] \wedge \exists a \in Car(a[carMake] = Audi \wedge ar[carID] = a[carID] )) \}$

III.

As TRC/DRC cannot express aggregation function, it cannot be expressed.

IV.

$\{ c \mid \forall b' \in Branch (\exists r \in Rental (c[cID] = r[cID] \wedge \exists b \in Branch (b'[bState] = b[bState] \wedge r[pickupBid] = b[bid]))) \}$

V.

As TRC/DRC cannot express aggregation function, it cannot be expressed.

VI.

$\{ res \mid \forall r' \in Rental (r'[pickupTD].year = 2017 \wedge (\exists r \in Rental (res[cID] = r[cID] \wedge r[pickupTD].year = 2017 \wedge r[cost] \geqslant r'[cost] \wedge \exists c \in Customer (r[cID] = c[cID] \wedge res[cName] = c[cName]))) \}$

**Problem 2**

**(a)**

Customer (<u>cID</u>, cName, cPhone, cCard, ccID)
Location (<u>lID</u>, name, street, city, zip, state, phone)
Shipment (<u>sID</u>, sDateTime, weight, sourceID, destinationID, cost, senderID, payerID)
Track (<u>tID</u>, sID, tDateTime, tLongitude, tLatitude, tCity, tZip, tDescription)
ZipCategory (<u>zip</u>, category)
Price (minWeight, <u>maxWeight, sourceCategory, destinationCategory, isInSameState</u>, pPrice)
CustomerClass (<u>ccID</u>, className, discount)

Foreign Key:
Customer (ccID) -> CustomerClass (ccID)
Location (zip) -> ZipCategory (zip)

Shipment (sourceID) -> Location (lID)
Shipment (destinationID) -> Location (lID)
Shipment (senderID) -> Customer (cID)
Shipment (payerID) -> Customer (cID)
Track (sID) -> Shipment (sID)

**(b)**

I.

```
select cID, cName, sum(cost)
from Customer join Shipment on cID = payerID
where getYear(sDateTime) = 2017
group by cID, cName
```

II.

```
with TrackLastTime as
        (select sID, tDateTime as tDateTimeLast
        from Track
        where tDescription = “Arrival scan, Miami airport” and now() – tDateTime > 5 * 24 *
3600);

select sID
from TrackLastTime
where sID not in (
        select sID
        from TrackLastTime natural join Track
        where tDateTime > tDateTimeLast
)
```

III.

```
with AllPackages as
        (select sID, cost
        from Shipment join Customer on payerID = cID
        where year(sDateTime) = 2017 and month(sDateTime) = 10 and cName = ‘ACME
Global’);

with UserDiscount as
        (select discount
        from Customer natural join CustomerClass
        where cName = ‘ACME Global’);
```

```
with ExpectedPrice as
        (select ship.sID as sID, p.pPrice as price
        from Price p, Shipment ship, Location ls, Location ld, ZipCategory s, ZipCategory d
        where ship.sourceID = ls.lID and ship.destinationID = ld.lID and ls.zip = s.zip and ld.zip
        = d.zip and s.category = p.souceCategory and d.category = p.destinationCategory and
        ship.weight <= p.maxWeight and ship.weight > p.minWeight and (ls.state = ld.state) = p.
        isInSameState
        group by sID);

select sID
from AllPackages, ExpectedPrice, UserDiscount
where AllPackages.sID = ExpectedPrice.sID and AllPackages.cost > ExpectedPrice.price *
UserDiscount.discount
```