University of Toronto
CSC343, Summer 2015

Assignment 1 – Solutions

Note that there are multiple correct answers to all of these questions.

Part 1: Queries

1. Report the names of the drivers who have never driven for more than 50 kms each in two different travels.

   $LongDistanceDrivers(travelID, driverID) :=$
   $\Pi_{travelID,driverID}\sigma_{distance>50}(Driver \bowtie Travel)$

   $LongDistanceDriversTwice(driverID) :=$
   $\Pi_{L1.driverID}\sigma_{L1.driverID=L2.driverID \wedge L1.travelID \neq L2.travelID}(\rho_{L1}LongDistanceDrivers \times$
   $\rho_{L2}LongDistanceDrivers)$

   $ShortDistanceDrivers(driverID) := \Pi_{driverID}Driver - LongDistanceDriversTwice$

   $\textbf{Answer}(\textbf{name}) := \Pi_{name}(ShortDistanceDrivers \bowtie Driver)$

2. Report the zipcode of the location at which the most number of passengers were dropped in a single travel. Assume that all passengers travelling in a cab are dropped together at the same location.

   $DropLocationsAndPassengers(locationID, noOfPassengers) :=$
   $\Pi_{locationID,noOfPassengers}\sigma_{status='completed'}(Travel \bowtie Booking)$

   $NotMaxPassengers(locationID, noOfPassengers) :=$
   $\Pi_{D1.locationID,D1.noOfPassengers}\sigma_{D1.noOfPassengers<D2.noOfPassengers}$
   $(\rho_{D1}DropLocationsAndPassengers \times \rho_{D2}DropLocationsAndPassengers)$

   $MaxPassengers(locationID) :=$
   $\Pi_{locationID}(DropLocationsAndPassengers - NotMaxPassengers)$

   $\textbf{Answer}(\textbf{zipcode}) := \Pi_{zipcode}(MaxPassengers \bowtie Location)$

3. Report the names of the customers who have been picked up from at least two different locations in travels that were 'completed'.

   $CompletedPickupLocations(customerID, locationID) :=$
   $\Pi_{customerID,pickupLocationID}\sigma_{Travel.status='completed'}(Booking \bowtie Travel)$

   $TwoDifferentLocations(customerID) :=$
   $\Pi_{C1.customerID}\sigma_{C1.customerID=C2.customerID \wedge C1.pickupLocationID \neq C2.pickupLocationID}$
   $(\rho_{C1}CompletedPickupLocations \times \rho_{C2}CompletedPickupLocations)$

   $\textbf{Answer}(\textbf{name}) := \Pi_{name}(Customer \bowtie TwoDifferentLocations)$

4. Report the manufacturer and model names of cabs which were always driven by the same driver. Also report the name of the driver.

$CabsWithDifferentDrivers(cabID) :=$
$\Pi_{T1.cabID}\sigma_{T1.cabID=T2.cabID \wedge T1.driverID \neq T2.driverID}(\rho_{T1}Travel \times \rho_{T2}Travel)$

$CabsWithSameDrivers(cabID) := \Pi_{cabID}Cab - CabsWithDifferentDrivers$

$\mathbf{Answer}(\mathbf{manufacturer}, \mathbf{model}, \mathbf{driverName}) :=$
$\Pi_{manufacturer,model,name}(CabsWithSameDrivers \bowtie Cab \bowtie Travel \bowtie Driver)$

5. Report the type of cab which was filled (with passengers) to capacity exactly twice. Also report its capacity.

$CabPassengersFilled(cabID, capacity, travelID) :=$
$\Pi_{cabID,capacity}\sigma_{capacity=noOfPassengers}(Travel \bowtie Cab \bowtie Booking)$

$CabFilledAtleastTwice(cabID) :=$
$\Pi_{C1.cabID}\sigma_{C1.cabID=C2.cabID \wedge C1.travelID \neq C2.travelID}$
$(\rho_{C1}CabPassengersFilled \times \rho_{C2}CabPassengersFilled)$

$CabFilledAtleastThrice(cabID) :=$
$\Pi_{C1.cabID}\sigma_{C1.cabID=C2.cabID \wedge C2.cabID=C3.cabID \wedge C1.travelID \neq C2.travelID \wedge C2.travel \neq C3.travelID}$
$(\rho_{C1}CabPassengersFilled \times \rho_{C2}CabPassengersFilled \times \rho_{C3}CabPassengersFilled)$

$CabFilledExactlyTwice(cabID) :=$
$\Pi_{C1.cabID}(CabFilledAtleastTwice - CabFilledAtleastThrice)$

$\mathbf{Answer}(\mathbf{type}, \mathbf{capacity}) := \Pi_{type,capacity}(CabFilledExactlyTwice \bowtie Cab)$

6. Report the name and age of all drivers who are youngest and have driven a cab with the oldest date of deployment.

$NotOldestCab(cabID) := \Pi_{C1.cabID}\sigma_{C1.deploymentDate>C2.deploymentDate}(\rho_{C1}Cab \times \rho_{C2}Cab)$

$OldestCab(cabID) := \Pi_{cabID}(Cab - NotOldestCab)$

$DriversOfOldestCab(driverID) := \Pi_{driverID}(OldestCab \bowtie Travel)$

$NotYoungestDriver(driverID) :=$
$\Pi_{D1.driverID}\sigma_{D1.age>D2.age}(\rho_{D1}Driver \times \rho_{D2}Driver)$

$YoungestDriver(driverID) := Driver - NotYoungestDriver$

$OldestCabYoungestDriver(driverID) := DriversOfOldestCab \cap YoungestDriver$

$\mathbf{Answer}(\mathbf{name}, \mathbf{age}) := \Pi_{name,age}(OldestCabYoungestDriver \bowtie Driver)$

7. Report the ID of the customer who took the longest time to book a cab after registering with the company. You may use comparison operators to compare dates and timestamps with each other.

$BookingTimes(customerID, bookingTime) :=$
$\Pi_{customerID,bookingTime}(Travel \bowtie Customer \bowtie Booking)$

$NotEarliestBookingTimes(customerID, bookingTime) :=$
$\Pi_{customerID,bookingTime}\sigma_{B1.customerID=B2.customerID \wedge B1.bookingTime>B2.bookingTime}$
$(\rho_{B1}BookingTimes \times \rho_{B2}BookingTimes)$

$EarliestBookingTimes(customerID, bookingTime) :=$
$BookingTimes - NotEarliestBookingTimes$

$RegistrationAndBookingTimes(customerID, bookingTime, registrationDate) :=$
$\Pi_{customerID,bookingTime,registrationDate}(EarliestBookingTimes \bowtie Customer)$

$NotLongestRegistrationTimes(customerID) :=$
$\Pi_{R1.customerID}\sigma_{(R1.bookingTime-R1.registrationDate)<(R2.bookingTime-R2.registrationDate)}$
$(\rho_{R1}RegistrationAndBookingTimes \times \rho_{R2}RegistrationAndBookingTimes)$

$LongestRegistrationTimes(customerID) :=$
$\Pi_{customerID}Customer - NotLongestRegistrationTimes$

$\mathbf{Answer(customerID)} := LongestRegistrationTimes$

Note: Submissions which did not consider earliest booking times will not be penalized since the question did not explicitly mention the longest time to book the "first" cab.

8. Report the name and salary of the driver who is paid the second highest salary among all female drivers.

$FemaleDrivers(driverID) := \Pi_{driverID}\sigma_{gender='female'}Driver$

$NotHighestSalary(driverID) :=$
$\Pi_{F1.driverID}\sigma_{F1.salary<F2.salary}(\rho_{F1}FemaleDrivers \times \rho_{F2}FemaleDrivers)$

$NotSecondHighestSalary(driverID) :=$
$\Pi_{N1.driverID}\sigma_{N1.salary<N2.salary}(\rho_{N1}NotHighestSalary \times \rho_{N2}NotHighestSalary)$

$SecondHighestSalary(driverID) := NotHighestSalary - NotSecondHighestSalary$

$\mathbf{Answer(name, salary)} := \Pi_{name,salary}(SecondHighestSalary \bowtie Driver)$

9. Report the cost of the shortest distance travel for which the booking was 'confirmed' or the travel was 'terminated'.

$ConfirmedBookingTravels(travelID) :=$
$\Pi_{travelID}\sigma_{B.status='confirmed'}(\rho_{B}Booking \bowtie \rho_{T}Travel)$

$TerminatedTravels(travelID) := \Pi_{travelID}\sigma_{status='terminated'}Travel$

$ConfirmedOrTerminated(travelID) := ConfirmedBookingTravels \cup TerminatedTravels$

$NotShortestTravels(travelID) := \Pi_{T1.travelID}\sigma_{T1.distance>T2.distance}(\rho_{T1}Travel \times \rho_{T2}Travel)$

$ShortestTravels(travelID) := \Pi_{travelID}Travel - NotShortestTravels$

$\mathbf{Answer}(\mathbf{cost}) := \Pi_{cost} ShortestTravels \bowtie Travel$

10. A driver is considered "flexible" if he/she has driven every *type* of cab. Report the names of all "flexible" drivers.

$AllDriverAndCabTypes(driverID, type) = \Pi_{driverID,type}(Driver \times Cab)$

$DrivenCabTypes(driverID, type) := \Pi_{driverID,type}(Travel \bowtie Cab)$

$NotFlexibleDrivers(driverID) := \Pi_{driverID}(AllDriverAndCabTypes - DrivenCabTypes)$

$FlexibleDrivers(driverID) := \Pi_{driverID} Driver - NotFlexibleDrivers$

$\mathbf{Answer}(\mathbf{name}) := \Pi_{name}(FlexibleDrivers \bowtie Driver)$

Part 2: Additional Integrity Constraints

Express the following integrity constraints with the notation $R = or \neq \emptyset$, where R is an expression of relational algebra. If a constraint cannot be expressed using this notation, simply write "cannot be expressed".

You are welcome to define intermediate results with assignment and then use them in an integrity constraint.

1. No driver can have a monthly salary less than $2000.
   $\sigma_{salary<2000} Driver = \emptyset$
2. A customer can not make a booking before his/her registration date.
   $\sigma_{bookingTime<registrationDate}(Customer \bowtie Booking) = \emptyset$
3. Every cab must be of type 'sedan', 'limousine' or 'bus'.
   $\sigma_{type \neq 'sedan' \wedge type \neq 'limousine' \wedge type \neq 'bus'} Cab = \emptyset$
4. There exist drivers who have never driven a 'limousine' as part of some travel.
   $\Pi_{driverID} Driver - \Pi_{driverID} \sigma_{type='limousine'}(Travel \bowtie Cab) \neq \emptyset$
5. A driver can pick up customers only after at least an hour has passed since another driver dropped customers using the same cab.
   $CabDrivers(cabID, driverID, pickupTime, dropTime) :=$
   $\Pi_{cabID,driverID}(Travel \bowtie Cab \bowtie Booking)$

   $\sigma_{C1.cabID=C2.cabID \wedge C1.driverID \neq C2.driverID \wedge C1.pickupTime > C2.dropTime \wedge C1.pickupTime - C2.dropTime \leq 1hour}$
   $(\rho_{C1} CabDrivers \times \rho_{C2} CabDrivers) = \emptyset$