University of Toronto
CSC343, Fall 2020

Assignment 1 – Sample Solutions

Note that there are multiple correct answers to all of these questions. Please post to Piazza if you think you've found an error in these solutions.

1. Context: Bus trips have been falling in popularity. We are reviewing our pricing for these, as well as frequent traveller discounts, and arrangements with resellers like TravelZoo and Groupon.

   Query: For each instance of a trip that includes transportation by bus, find the highest price paid by any traveller for that instance, and find the lowest price paid by any traveller for that instance. Report the trip instance ID and advertised price, the highest price paid, and the lowest price paid.

   Solution:

   — This trip instance includes transportation by bus.
   $BusTripInstance(instID, price) :=$
   $\pi_{instID,price}\sigma_{transportationType='bus'}\,(TripInstance \bowtie Trip \bowtie Itinerary)$

   — This trip instance, which included transportation by bus, was booked by this traveller for this price.
   $TargetBooking(instID, travID, pricePaid) :=$
   $\pi_{instID,travID,pricePaid}\,(BusTripInstance \bowtie Booking)$

   — This traveller didn't pay the highest price for this trip instance.
   — Note: we don't have to check that the travIDs are different, since there can't be two
   — prices for the same trip instance and traveller (due to the key of Booking).
   $NotHighestPrice(instID, travID, pricePaid) :=$
   $\pi_{\substack{T1.instID,\\ T1.travID,\\ T1.pricePaid}}\ \sigma_{\substack{(T1.instID=T2.instID)\\ \wedge\\ (T1.pricePaid<T2.pricePaid)}}\ (\rho_{T1}TargetBooking \times \rho_{T2}TargetBooking)$

   — This price is the highest price recorded for this trip instance.
   $HighestPrice(instID, highestPrice) := \pi_{instId,pricePaid}(TargetBooking - NotHighestPrice)$

   — This traveller didn't pay the lowest price for this trip instance.
   $NotLowestPrice(instID, travID, pricePaid) :=$
   $\pi_{\substack{T1.instID,\\ T1.travID,\\ T1.pricePaid}}\ \sigma_{\substack{(T1.instID=T2.instID)\\ \wedge\\ (T1.pricePaid>T2.pricePaid)}}\ (\rho_{T1}TargetBooking \times \rho_{T2}TargetBooking)$

   — This price is the highest price recorded for this trip instance.
   $LowestPrice(instID, lowestPrice) := \pi_{instId,pricePaid}(TargetBooking - NotLowestPrice)$

   — Answer.
   $Answer(instID, price, highestPrice, lowestPrice) :=$
   $BusTripInstance \bowtie HighestPrice \bowtie LowestPrice$

2. Context: We are reviewing trips that haven't been offered recently, with a plan to rethink whether they might be more appealing if they visited more cities.

   Query: For each trip that has had at least 2 instances but none in 2019 or since, report the trip id and name, and the number of stops on its itinerary.

   Solution:
   Cannot be expressed.

3. Context: The company is planning a special promotion for people who have taken extensive trips to Paris.

   Query: Find all travellers who have booked at least one instance of each trip that starts in Paris and includes 3 or more cities (the start city counts towards this total). Report the traveller's first and last name, and email address.

   Solution:

   — This trip instance starts in Paris
   $StartsInParis(tripID) := \pi_{tripID}\sigma_{startCity=cityID \wedge name='Paris'}\,(Trip \times City)$

   — This trip starts in this city.
   $StartCity(tripID, cityID) := \pi_{tripID,startCity}Trip$

   — This trip visits this city.
   $TripCity(tripID, cityID) := StartCity \bigcup (\pi_{tripID,cityID}Itinerary)$

   — This trip visits at least 3 different cities.
   — Corner case: our schema doesn't prevent repeat cities i.e. a trip can have 2 rows
   — in itinerary with different day values for the same city. Similarly, the start city may
   — be the same as one of the cities in the itinerary. Consequently, using the Itinerary
   — table only and looking for at least 2 cities or comparing the values of days in the
   — Itinerary table will be incorrect.
   $AtLeastThree(tripID) :=$

   $$\pi_{T1.tripID}\sigma_{\substack{T1.tripID=T2.tripID \\ \wedge \\ T2.tripID=T3.tripID \\ \wedge \\ T1.cityID<T2.cityID \\ \wedge \\ T2.cityID<T3.cityID}}\,(\rho_{T1}TripCity \times \rho_{T2}TripCity \times \rho_{T3}TripCity)$$

   — This trip starts in Paris and visits at least 3 cities.
   $TargetTrip(tripID) := StartsInParis \bigcap AtLeastThree$

   — This traveller has booked at least one instance of this trip.
   $TripBooking(travID, tripID) := \pi_{travID,tripID}\,(Booking \bowtie TripInstance)$

   — This traveller should have been on this trip instance (to be included in the result).
   — Alternatively, we could have used the travID from the Booking table, but be careful:
   — which table you pull these from has to considered when doing the set difference later.
   $ShouldHaveBooked(travID, tripID) := (\pi_{travID}Traveller) \times TargetTrip$

   — This traveller hasn't booked all target trips.
   $HaveNotBookedAll(travID) := \pi_{travID}\,(ShouldHaveBooked - TripBooking)$

   — This is a target traveller: they booked at least one instance of each target trip.
   $TargetTraveller(travID) := (\pi_{travID}Traveller) - HaveNotBookedAll$

— Answer.
$Answer(firstName, lastName, email) := \sigma_{fname,lname,email}\,(TargetTraveller \bowtie Traveller)$

4. Context: We are looking for travellers who like a lot of perqs. We might design a special trip aimed at this market.

   Query: Let's define a plush trip as one with the most base activities. (There could be exactly one plush trip, or perhaps there are several that are tied.) Report the name and email address of all travellers who have taken one or more of these plush trips, have never taken a non-plush trip, and have never booked an add-on activity.

   Solution:
   Cannot be expressed.

5. Context: The company is planning a staff retreat. Some of the activities will be done in pairs, and we'd like to group people who have experiences in common.

   Query: Find all pairs of staff who've both been trip manager for different instances of the same trip. For each pair, report both people's email address, and the start date of the very first trip instance they were manager for. (It might not be an instance of the/a trip they have in common with the other person.) Use attribute names staff1, start1, staff2, start2 and make the person represented by the first two attributes be the one from the pair with the most seniority (that is, the one whose employment date comes first). If there is a tie, it doesn't matter whose data is represented by the first two attributes.

   Solution:

   — This trip instance is not the first trip for its trip manager.

   $NotFirstTrip(instID) :=$

   $$\pi_{T1.instID}\sigma_{\substack{T1.tripManager=T2.TripManager \\ \wedge \\ T1.instID \neq T2.instID \\ \wedge \\ T1.startDate > T2.startDate}}(\rho_{T1}TripInstance \times \rho_{T2}TripInstance)$$

   — This trip instance is the first trip for its trip manager.

   $FirstTrip(instID) := (\pi_{instID}TripInstance) - NotFirstTrip$

   — This is the start date of this trip manager's first trip.

   $StaffStartDate(staffID, start) := \pi_{tripManager,startDate}(FirstTrip \bowtie TripInstance)$

   — This staff member who has this email, went on their first trip on the start date
   — and was employed on employDate.

   $StaffInfo(staffID, email, start, employDate) :=$

   $$\pi_{staffID,email,start,employDate}(StaffStartDate \bowtie Staff)$$

   — This staff member was the trip manager for this trip instance,
   — which is an instance of this trip. NB: a natural join on TripInstance and Staff
   — is not sufficient; we need to check tripManager = staffID "by hand".

   $RelevantInfo(instID, tripID, staffID, email, start, employDate) :=$

   $$\pi_{instID,tripID,staffID,email,start,employDate}(TripInstance \bowtie_{tripManager=staffID} StaffInfo)$$

   — Answer.
   — Corner Case: Our schema doesn't prevent having 2 staff members with the same
   — employment date. As a result, we need the disjunction. The final comparison of
   — staff IDs removes repeat pairs in the answer ("pseudo duplicates").

   $Answer(staff1, start1, staff2, start2) :=$

   $$\pi_{\substack{R1.email, \\ R1.start, \\ R2.email, \\ R2.start}}\sigma_{\substack{R1.tripID=R2.tripID \\ \wedge \\ R1.instID=R2.instID \\ \wedge \\ [(R1.employDate<R2.employDate) \\ \vee \\ (R1.employDate=R2.employDate \\ \wedge \\ R1.staffID<R2.staffID)]}}(\rho_{R1}RelevantInfo \times \rho_{R2}RelevantInfo)$$

6. Context: We are designing a new trip for very adventurous travellers and are targeting it towards people who are willing to spend money on add-on activities.

   Query: Find travellers who have booked more than one trip instance and on every trip instance they've booked, they booked all of the add on activities available. Report simply the email address of each such traveller.

   Solution:

   — Note: If a trip has no add-ons, then it is trivially true that any traveller who books
   — an instance of this trip has booked all add-ons.

   — This traveller has booked at least 2 trip instances.
   $TargetTraveller(travID) := \pi_{B1.travID}\sigma_{\substack{B1.instID \neq B2.instID \\ \wedge \\ B1.travID = B2.travID}}(\rho_{B1}Booking \times \rho_{B2}Booking)$

   — This target traveller has booked this trip instance.
   $TargetBooking(travID, instID) := \pi_{travID,instID}(TargetTraveller \bowtie Booking)$

   — This target traveller has booked this add-on activity on this trip instance.
   $TargetAddOnBooking(travID, instID, addID) :=$
   $\pi_{travID,instID,addID}(TargetBooking \bowtie AddOnBooking)$

   — This target traveller should have booked this add-on on this trip instance.
   $ShouldHaveBooked(travID, instID, addID) :=$
   $\pi_{travID,instID,addID}(TargetBooking \bowtie TripInstance \bowtie AddOnActivity)$

   — This target traveller hasn't booked all add-ons they should have booked.
   $DidNotBookAll(travID) := \pi_{travID}(ShouldHaveBooked - TargetAddOnBooking)$

   — This is the email of a target traveller.
   $Answer(email) := \pi_{email}((TargetTraveller - DidNotBookAll) \bowtie Traveller)$

   — Corner cases to consider:
   — * 1 or all trips with no Add-ons.
   — * No booked add-ons at all.

7. Context: We think some of our reviews may be fake.

   Query: Let's say a low-ball rating for a trip is one that (a) is at most 2 stars, (b) is the lowest rating (lowest number of stars) for that trip, and (c) is unique for that trip: no one else gave that trip the same number of stars. Find travellers who've given at least one review and whose every review either gives a low-ball rating or has text consisting of just the word "terrible". For each such traveller, report just their email address.

   Solution:

   — This review gives at most 2 stars.
   $PoorRating(tripID, travID) := \pi_{tripID,travID}\sigma_{stars \leq 2} Review$

   — This review does not give the lowest rating for this trip.
   $NotLowest(tripID, travID) :=$
   $$\pi_{tripID,travID}\sigma_{r1.tripID=r2.tripID \overset{\wedge}{} r1.stars>r2.stars}(\rho_{r1}Review \times \rho_{r2}Review)$$

   — This review gives the lowest rating for this trip.
   $Lowest(tripID, travID) := (\Pi_{tripID,travID}Rating) - NotLowest$

   — This review is not unique for this trip: someone else gave the same number of stars.
   $NotUnique(tripID, travID) :=$
   $$\pi_{tripID,travID}\sigma_{r1.tripID=r2.tripID \overset{\wedge}{} r1.travID \neq r2.travID \overset{\wedge}{} r1.stars=r2.stars}(\rho_{r1}Review \times \rho_{r2}Review)$$

   — This review is unique for this trip.
   $Unique(tripID, travID) := (\Pi_{tripID,travID}Rating) - NotUnique$

   — This review is a low-ball rating.
   $LowBall(tripID, travID) := PoorRating \bigcap Lowest \bigcap Unique$

   — This review's text consists of just the word "terrible".
   $Terrible(tripID, travID) := \pi_{tripID,travID}\sigma_{text=`terrible'}Review$

   — This review either gives a low-ball rating or just says "terrible".
   $SuspiciousReview(tripID, travID) := LowBall \bigcup Terrible$

   — This review is not "suspicious". Let's call it "fine".
   $FineReview(tripID, travID) := (\Pi_{tripID,travID}Review) - SuspiciousReview$

— This traveller gave at least one review but never gave a fine review; their every review
— was suspicious.
$TargetTraveller(travID) :=$
$\quad (\Pi_{travID} Review) - (\Pi_{travID} FineReview)$

— The emails of the target travellers.
$\Pi_{email}(TargetTraveller \bowtie Traveller)$

8. Context: We are looking for some social media influencers to expand our sales of trips to South America, so we want to hire a young traveller who has had a positive outlook on our trips to Brazil.

   Query: Find all travellers who've rated a trip that includes Rio de Janeiro and Sao Paulo, and have given that trip a rating that is above the average rating for that trip. Report the traveller email, and first and last name. Include only travellers whose birth year is between 1995 and 2002 inclusive.

   Solution:
   Cannot be expressed.

Part 2: Additional Integrity Constraints

Express the following integrity constraints with the notation $R = \emptyset$, where R is an expression of relational algebra. You are welcome to define intermediate results with assignment. The last step for each question must be a single assertion of the form expression $= \emptyset$.

Remember that at least one of the queries and/or integrity constraints in this assignment cannot be expressed in the language that you are using. In those cases, simply write "cannot be expressed".

1. Context: We don't want to book a traveller on an instance of a trip if it's clear that they will not have their accommodation needs met.

   Constraint: A traveller cannot book an instance of a trip unless there is, in every city on its itinerary (including the start city), an accommodation for that trip instance that offers either a room of the size they requested in their booking or a larger room.

   Solution:

   — This trip starts in this city.
   $StartCity(tripID, cityID) := \pi_{tripID,startCity} Trip$

   — This trip visits this city.
   $TripCity(tripID, cityID) := StartCity \bigcup (\pi_{tripID,cityID} Itinerary)$

   — This traveller requires a room of at least this size in this city on this trip instance.
   $ShouldBeAvailable(travID, instID, cityID, roomType) :=$
   $\pi_{travID,instID,cityID,roomType} (Booking \bowtie TripInstance \bowtie TripCity)$

   — This room size is available on this trip instance in this city.
   $ActuallyAvailable(instID, cityID, roomSize) :=$
   $\pi_{instID,cityID,roomSize} (TripAccomodation \bowtie Accomodation \bowtie Room)$

   — This booking's room requirement is satisfied.
   $SatisfiedRequirement(travID, instID) :=$
   $\pi_{travID,instID} \sigma_{\substack{S.instID=A.instID \\ \wedge \\ S.cityID=A.cityID \\ \wedge \\ S.roomType \leq A.roomSize}} (\rho_S ShouldBeAvailable \times \rho_A ActuallyAvailable)$

   — The constraint.
   $(\pi_{travID,instID} Booking) - SatisfiedRequirement = \phi$

2. Context: The company is concerned about the legitimacy of reviews.

   Constraint: You can only review a trip if you've booked at least one instance of it. (However, nothing stops you from reviewing a trip before you go on that trip.)

   Solution:

   — This reviewer has been on an instance of this trip.
   $TookTrip(travID, tripID) := \pi_{travID,tripID}\,(Booking \bowtie TripInstance)$

   — This traveller has written a review for this trip.
   $ReviewedTrip(travID, tripID) := \pi_{travID,tripID} Review$

   — Constraint.
   $ReviewedTrip - TookTrip = \phi$

3. Context: The company needs to recoup the lost income from discounts by making money on add-on activities.

   Constraint: If a traveller does not book either the most expensive or second-most expensive add-on activity on a trip instance, they can't get a discount price for that trip instance (that is, they can't pay less than the advertised price). This constraint applies only to instances of trips that have two or more add-on activities.

   Solution:

   Corrections made: NotMostExpensive needs to include the attribute cost, because it is needed in computing NotSecondMostExpensive. (And cost must be removed before doing the set difference to find MostExpensive.)

   In addition, Discounted must only include trips with 2 or more add-ons, as the constraint only applies to those trips.

— This trip has at least 2 add-ons.
$TargetTrip(tripID, addID, cost) :=$
$\pi_{\substack{A1.addID,\\ A1.cost}} \sigma_{A1.tripID=A2.tripID \wedge A1.addID \neq A2.addID} (\rho_{A1} AddOnActivity \times \rho_{A2} AddOnActivity)$

— This add-on activity, with this cost, is not the most expensive add-on in this trip.
$NotMostExpensive(tripID, addID, cost) :=$
$\pi_{T1.tripID,T1.addID,T1.cost} \sigma_{\substack{T1.tripID=T2.tripID \wedge T1.addID \neq T2.addID \\ \wedge\ T1.cost<T2.cost}} (\rho_{T1} TargetTrip \times \rho_{T2} TargetTrip)$

— This add-on is the most expensive add-on in this trip.
$MostExpensive(tripID, addID) := (\pi_{tripID,addID} TargetTrip) - (\pi_{tripID,addID} NotMostExpensive)$

— This add-on is not the 2nd most expensive add-on in this trip.
$NotSecondMostExpensive(tripID, addID) :=$
$\pi_{N1.tripID,N1.addID} \sigma_{\substack{N1.tripID=N2.tripID \wedge N1.addID \neq N2.addID \\ \wedge\ N1.cost<N2.cost}} (\rho_{N1} NotMostExpensive \times \rho_{N2} NotMostExpensive)$

— This add-on is the 2nd most expensive add-on in this trip.
$SecondMostExpensive(tripId, addID) := NotMostExpensive - NotSecondMostExpensive$

— This add-on is the most expensive or 2nd most expensive add-on in this trip.
$EligibleAddOn(tripID, addID) := MostExpensive \bigcup SecondMostExpensive$

— This traveller is eligible for a discount on the price of this trip instance.
— Note: joined to TripInstance since the add-on could be associated with an instance
— whose trip doesn't offer this add-on.
$EligibleForDiscount(travID, instID) :=$
$\pi_{travID,instID} (AddOnBooking \bowtie TripInstance \bowtie EligibleAddOn)$

— This traveller obtained a discount on the price of this trip instance.
$Discounted(travID, instID) := \pi_{travID,instID}\sigma_{pricePaid<price}\,(Booking \bowtie TripInstance \bowtie TargetTrip$

— Constraint.
$Discounted - EligibleForDiscount = \phi$