CSC343: Assignment 1

October 15th, 2020

Richard Yang (1004908870)
Katarina Chiam (1004908996)

Part 1: Queries

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:

--Step 1: instance ID and advertised price for all trips that include transportation by bus

$$TripBus(instID, price) \coloneqq \Pi_{instID,price}(\sigma_{transportation='bus'}(TripInstance \bowtie Itinerary))$$

--Step 2: instance ID, advertised price and price paid by customers for Booked trips that include transportation by bus

$$BookedBusTrip(instID, price, pricePaid) \coloneqq \Pi_{instID,price,pricePaid}(TripBus \bowtie Booking)$$

--Step 3a: instance ID, advertised price and all of the prices that are not the maximum price paid by customers for each trip instance

$$NotMax(instID, price, pricePaid) \coloneqq \pi_{instID,price,pricePaid}(\sigma_{B1.pricePaid<B2.pricePaid}(\rho_{B1}BookedBusTrip \times \rho_{B2}BookedBusTrip))$$

--Step 3b: instance ID, advertised price and maximum price paid by customer for each trip instance

$$Max(instID, price, pricePaid) \coloneqq BookedBusTrip - NotMax$$

--Step 4a: instance ID, advertised price and all of the prices that are not the minimum price paid by customers for each trip instance

$$NotMin(instID, price, pricePaid) \coloneqq \pi_{instID,price,pricePaid}(\sigma_{B1.pricePaid>B2.pricePaid}(\rho_{B1}BookedBusTrip \times \rho_{B2}BookedBusTrip))$$

--Step 4b: instance ID, advertised price and minimum price paid by customer for each trip instance

$$Min(instID, price, pricePaid) \coloneqq BookedBusTrip - NotMin$$

--Step 5: instance ID, advertised price, highest price paid, lowest price paid by any traveller for each trip instance

$$PriceSummary(instID, price, pricePaid) \coloneqq BookedBusTrip \bowtie Max \bowtie Min$$

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:

--Step 1: trip ID of all trips offered that start in Paris

$$ParisStart(tripID) \coloneqq \Pi_{tripID}(\ \sigma_{startCity='Paris'}(Trip))$$

--Step 2: trip ID of all trips that start in Paris in the Itinerary (so we know that they were booked rather than just offered)

$$ParisItinerary(tripID) \coloneqq \sigma_{PS.tripID=Itinerary.tripID}(\rho_{PS}ParisStart\ \times Itinerary)$$

--Step 3: trip ID of all trips that start in Paris in the Itinerary that include at least 3 distinct cities

$$\begin{gathered}BigParisTrip(tripID) \\ \coloneqq \Pi_{tripID}(\sigma_{\substack{T1.tripID=T2.tripID \\ \wedge \\ T2.tripID=T3.tripID \\ \wedge \\ T3.tripID=T1.tripID \\ \wedge \\ T1.cityID\neq T2.cityID \\ \wedge \\ T2.cityID\neq T3.cityID \\ \wedge \\ T3.cityID\neq T1.cityID}}\ (\rho_{T1}ParisItinerary\ \times \rho_{T2}ParisItinerary \times \rho_{T3}ParisItinerary))\end{gathered}$$

--Step 4: instance ID of all trips that start in Paris in the Itinerary that include 3 distinct cities

$$BigParisTrip2(instID) \coloneqq \Pi_{instID}(TripInstance \bowtie BigParisTrip)$$

--Step 5: traveller ID of all travellers who booked any instance of trips that start in Paris that include at least 3 distinct cities

$$ParisBook(travID) \coloneqq \Pi_{travID}(BigParisTrip2 \bowtie Booking)$$

--Step 6: first name, last name and email of all travellers who booked any instance of trips that start in Paris include at least 3 distinct cities

$$ParisPromo(fname, lname, email) \coloneqq \Pi_{fname,lname,email}(ParisBook \bowtie Traveller)$$

4. Context: We are looking for travellers who like a lot of perks. 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.

   Methodology: First we'll determine for every trip manager their earliest startdate of a trip instance. Following this, we'll determine was the manager for every trip instance and get their employment date. Lastly we'll determine pairs based on each managers trips managed via tripID while making sure the instIDs were different and arranging such that the staff1 has senority (earlier employdate)

   Solution:
   – All tuples in TripInstance where startdate is not a manager's earliest startdate.

   $$ManagerOmitFirstStartDate(instId, tripID, tripManager, startDate, price) :=$$
   $$\Pi_{\substack{T1.instID\\T1.tripID\\T1.tripManager\\T1.startDate\\T1.price}} [\sigma_{\substack{T1.tripManager=T2.tripManager\\ \wedge\\ T1.startdate>T2.startdate}}[(\rho_{T1}TripInstance)\times(\rho_{T2}TripInstance)]]$$

   – All tuples in TripInstance where a statedate was a manager's earliest startdate.

   $$ManagersAndDate(staffID, instID, tripID, startdate, employdate) :=$$
   $$[[[\Pi_{\substack{staffID\\instID\\tripID}}[\sigma_{staffID=tripManager}(TripInstanceTeam \bowtie TripInstance)]]$$
   $$\bowtie ManagersStartDates] \bowtie (\Pi_{\substack{staffID\\employDate}} Staff)]$$

   – Pairs of managers which have been managers for the same trip

   $$AllPairData() :=$$
   $$\sigma_{\substack{M1.staffID\neg M2.staffID\\ \wedge\\ M1.instID\neg M2.instID\\ \wedge\\ M1.tripID=M2.tripID\\ \wedge\\ M1.employdate\leq m2.employdate}} (\rho_{M1}ManagersAndDate \times \rho_{M2}ManagersAndDate)$$

   – Final solution of pairs with the requested attributes

   $$Pairs(staff1, start1, staff2, start2) :=$$
   $$\Pi_{M1.staffID,M1.startdate,M2.staffID,M2.startdate} AllPairData$$

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.

   Methodology: First we determine all travellers who have booked more than one trip instance. We'll also relate each trip instance to a tripID afterwards. Next we'll see all the addons these travellers have booked. Next we can construct a relation that contains the all the traveler's booked trips and all possible associated addons for that trip. Finally, we'll go through this table to determine tuples in which the traveler did not book all addons for their trip and get their email via their traveler ID and the traveler table.

   Solution:

   – All travIDs who have made at least two bookings .

   $RepeatTravs(instId, travID) :=$

   $$\Pi_{B1.instID, B1.travID}[\sigma_{\substack{B1.travID = B2.travID \\ \wedge \\ B1.instID \neg B2.instID}}([(\rho_{B1} Booking) \times (\rho_{B2} Booking)]]$$

   – All travIDs who have made at least two bookings plus the tripID for each booked instID.

   $RepeatTravsTrips(instId, travID, tripID) :=$

   $$[[\Pi_{instID, travID, tripID}[(RepeatTravs \bowtie TripInstance)]]$$

   – All addons travellers in RepeatTravs have booked

   $BookedAddOns(instID, travID, addID) := RepeatTravs \bowtie AddOnBooking$

   – tuples of travelers in repeatTravs with any possible addon for their respective trips

   $RepeatTravsAddOns(instID, travID, addID) :=$

   $$\Pi_{\substack{A1.tripID \\ A1.instID \\ A2.addID}}[\sigma_{\substack{A1.tripID = A2.tripID \\ \wedge \\ A1.instID = A2.instID}}(\rho_{A1} RepeatTravsTrips \times \rho_{A2} AddOnActivity)]$$

   – All repeat travellers who did not book all addons for their trip instance, in otherwords, all non-adventurous travellers

   $NonAdvenTravs(travID) :=$

   $$\Pi_{travID}(RepeatTravsAddOns - BookedAddOns)$$

   – All repeat travellers who booked all addons for their trip instances

   $AdvenTravs(travID) :=$

   $$\Pi_{travID} BookedAddOns - (\Pi_{travID} BookedAddOns \cap NonAdvenTravs)$$

   – All emails of adventurous travellers

   $AdvenTravsEmails(email) :=$

   $$\Pi_{email}(AdvenTravs \bowtie Traveller)$$

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:

--Step 1: trip ID, traveller ID, stars, text, email of all travellers who wrote at least 1 review with at most 2 stars or just the word "terrible" for a trip they went on

$$BadReviews(tripID, travID, stars, text, email) \\ \coloneqq \Pi_{tripID,R.travID,stars,text,email}(\sigma_{\substack{stars \leq 2 \\ \vee \\ text='terrible'}} (Review \bowtie Travellers))$$

--Step 2: trip ID, traveller ID, stars, text, email of all travellers who wrote a review that does not contain the lowest number of stars or the text is not just the word "terrible" for that trip

$$NotLowest(tripID, travID, stars, text, email) \\ \coloneqq \Pi_{BR1.tripID,BR1.travID,BR1.stars,BR1.text,BR1.email}(\sigma_{\substack{BR1.star > BR2.star \\ \wedge \\ BR1.text \neq 'terrible'}} (\rho_{BR1} BadReviews \times \rho_{BR2} BadReviews))$$

--Step 3: trip ID, traveller ID, stars, text, email of all travellers who wrote a review that has the lowest rating for that trip and the text is just the world "terrible" for that trip

$$LowReview(tripID, travID, stars, text, email) \coloneqq BadReviews - NotLowest$$

--Step 4: : trip ID, traveller ID, stars, text, email of all travellers who wrote a review that has the lowest rating for that trip and the text is not just the world "terrible" for that trip but the rating given is the same as another rating given for the same trip

$$NotUnique(tripID, travID, stars, text, email) \\ \coloneqq \Pi_{L1.tripID,L1.travID,L1.stars,L1.text,L1.email}(\sigma_{\substack{L1.tripID = L2.tripID \\ \wedge \\ L1.stars = L2.stars \\ \wedge \\ L1.text \neq 'terrible'}} (\rho_{L1} LowReview \times \rho_{L2} LowReview))$$

--Step 5: email of all travellers who wrote a review that has the lowest rating for that trip, the text is just the world "terrible" for that trip and the rating given for the same trip is unique

$$Lowball(email) \coloneqq \Pi_{email}(LowReview - NotUnique)$$

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.

   Methodology: In order to solve this question we would need to determine the average rating for a trip. To do this would entail the summation of all the reviews in terms of stars as well as the number of reviews present. However RA does not have a proper method to count or sum values within a column without prior knowledge of the max number present. Thus this question cannot be solved.

   Solution: This query is impossible and has no solution within the scope of RA.

Part 2: Integrity Constraints

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:

--Step 1: accommodation ID and room size of each accommodation available

$$RoomAvailable(accID, roomSize) \coloneqq \Pi_{Room.accID, roomSize}(Accomodation \bowtie Room)$$

--Step 2: accommodation ID, room size, instance ID of each accommodation available for each particular trip instance

$$RoomAvailInst(acc, roomSize, instID) \coloneqq \Pi_{RA.accID, roomSize, instID}(\sigma_{RA.accID=TA.accID}(\rho_{RA}RoomAvailable \times \rho_{TA}TripAccomodation)$$

--Step 3: Violators exist if the room size requested by the customer is larger than the room size available at each stop of the trip

$$\sigma_{RAI.roomsize<Booking.roomType}(\rho_{RAI}RoomAvailInst \bowtie Booking) = \emptyset$$

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.)

   Methodology: Determine the booked trips and the travelers who booked them. Using this, we can cross with Reviews to get all the reviewers who made a review for a corresponding trip they booked. Next we'll subtract this set of travelers form the set of all travelers who left reviews and the associated trip from that review and based on this constraint should be left with table with no data.

   Solution:

   – the travID of all travellers who booked a trip along with the tripID and instID associated with the booking.
   $BookedTrips(instId, travID, tripID) :=$
   $$\Pi_{B.instID, B.travID, B.tripID}[\sigma_{B.instID=T.instID}(\rho_B Booking \times \rho_T TripInstance)]$$

   – All traveller's travID and tripID who have book a instance of a trip and written a review for that instance
   $LegitReviewers(travID, tripID) :=$
   $$\Pi_{B.travID, B.tripID}[\sigma_{B.instID=R.instID \wedge B.travID=R.travID}(\rho_B BookedTrips \times \rho_T Reviews)]$$

   – travID and tripID of all travelers who wrote a review about anything $AllReviewers(travID, tripID) :=$

   $$\Pi_{travID, tripID}(Reviews)$$

   – Zero set with LegitReviewers and AllReviewers having the same set of tuples
   $$AllReviewers - LegitReviewers = \emptyset$$

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.

   Methodology: First determine all trips which have two or more addons. Next determine the top two most expensive addons. Then get all travelers who received discounts and verify that all travelers who received discounts for trips that had two or more addons purchased either the most or second most expensive addons.

   Solution:
   – determine all trips with at least two addOn activities
   $AddOnActivityMulti(addID, tripID, cost) :=$
   $$\Pi_{A1.addID,A1.tripID,A1.cost}[\sigma_{\substack{A1.tripID=A2.tripID \\ \wedge \\ A1.addID \neg A2.addID}}(\rho_{A1}AddOnActivity \times \rho_{A2}AddOnActivity)]$$

   – All addons with the most expensive addons per trip in AddONActivityMulti omitted
   $OmitMax(addID, tripID, cost) :=$
   $$\Pi_{A1.addID,A1.tripID,A1.cost}[\sigma_{\substack{A1.tripID=A2.tripID \\ \wedge \\ A1.cost<A2.cost}}(\rho_{A1}AddOnActivityMulti \times \rho_{A2}AddOnActivityMulti)$$

   – Each tuple is the most expensive addon per tripID for all tripIDs, and contains its addID, associated tripID, and cost
   $MostExpensive(addID, tripID, cost) := \Pi_{addID,tripID,cost}(AddOnActivity) - OmitMax$

   – All addons with the most expensive addons per trip in OmitMax omitted
   $OmitMaxMod(addID, tripID, cost) :=$
   $$\Pi_{M1.addID,M1.tripID,M1.cost}[\sigma_{\substack{M1.tripID=M2.tripID \\ \wedge \\ M1.cost<M2.cost}}(\rho_{M1}OmitMax \times \rho_{M2}OmitMax)]$$

   – Each tuple is the second most expensive addon per tripID for all tripIDs, and contains its addID, associated tripID, and cost
   $SecondMostExpensive(addID, tripID, cost) := OmitMax - OmitMaxMod$

   – Each tuple is the top two most expensive addon per tripID for all tripIDs, and contains its addID, associated tripID, and cost
   $ExpensiveAddOns(addID, tripID, cost) := MostExpensive \cup SecondMostExpensive$

   – Tuples of all travelers who booked either the most or second most expensive addoons and the instID associated $ExpensiveAddOnTravs(travID, instID) :=$
   $$\Pi_{A1.addID,A1.tripID,A1.cost}[\sigma_{E.addID=A.addID}(\rho_E ExpensiveAddOns \times \rho_A AddOnBooking)]$$

   – Each tuple is a traveler who got a discount when booking a trip and the associated isntID of the trip, travelers who booked a trip with less than two addons are omitted
   $DiscountTravs(travID, instID) :=$
   $$\Pi_{B.travID,B.instID}[AddOnActivityMulti \bowtie [\sigma_{\substack{B.instID=T.instId \\ \wedge \\ B.princepaid<T.price}}(\rho_B Booking \times \rho_T TripInstance)]]$$

– Zero set with ExpensiveAddOnTravs and DiscountTravs having the same set of tuples
$DiscountTravs - ExpensiveAddOnTravs = \emptyset$