University of Toronto
CSC343, Fall 2020

Assignment 1: Changyan Xu (1004802181) Jiaming Yang (1006458575)

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.

   Answer:

$$TripIncludeBus(instID) :=$$
$$\Pi_{TripInstance.instID}\Big(\sigma_{Itinerary.TransportationType='bus'}(TripInstance \bowtie Itinerary)\Big)$$

$$InstPricePaid(instID, pricePaid) := TripIncludeBus \bowtie Booking$$

– highest price.

$$NotHighestPrice(instID, pricePaid) :=$$
$$\Pi_{T1.instID,T2.pricePaid}\Big(\sigma_{\substack{T1.instID=T2.instID \\ \wedge \\ T1.pricePaid>T2.pricePaid}}(\rho_{T1}InstPricePaid \times \rho_{T2}InstPricePaid)\Big)$$

$$HighestPrice(instID, pricePaid) := InstPricePaid - NotHighestPrice$$

– lowest price.

$$NotlowestPrice(instID, pricePaid) :=$$
$$\Pi_{T1.instID,T1.pricePaid}\Big(\sigma_{\substack{T1.instID=T2.instID \\ \wedge \\ T1.pricePaid>T2.pricePaid}}(\rho_{T1}InstPricePaid \times \rho_{T2}InstPricePaid)\Big)$$

$$LowestPrice(instID, pricePaid) := InstPricePaid - NotlowestPrice$$

$$HighestAndLowest(instID, highestPricePaid, lowestPricePaid) :=$$
$$\Pi_{h.instID,h.pricePaid,l.pricePaid}\Big(\sigma_{h.instID=l.instID}(\rho_h HighestPrice \times \rho_l LowestPrice)\Big)$$

$$Q1Ans(instID, advertisedPrice, highestPricePaid, lowestPricePaid) :=$$
$$\Pi_{instID,price,highestPricePaid,lowestPricePaid}(HighestAndLowest \bowtie TripInstance)$$

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.

Answer:

Cannot be expressed.
Because wee need to use aggregate function, `count()` for adding conditions of counting TripInstances. However, a `count` function is not available in relational algebra

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.

Answer:

$$ParisID(startCity) := \Pi_{cityID}\Big(\sigma_{\substack{name='Paris' \\ \wedge \\ country='France'}}(City)\Big)$$

$$StartParis(tripID) := \Pi_{tripID}(Trip \bowtie ParisID)$$

$$Constructed(tripID, cityID) := (StartParis \times ParisID)$$

– This updated Itinerary does not contain the cityID of Paris

$$UpdatedItinerary(tripID, cityID) := \Pi_{tripID,cityID}(StartParis \bowtie Itinerary) - Constructed(tripID, cityID)$$

– have two more distinct cities other than the start city (Paris)

$$DesiredTrip(tripID) := \Pi_{u1.tripID}\Big(\sigma_{\substack{u1.tripID=u2.tripID \\ \wedge \\ u1.cityID\neq u2.cityID}}(\rho_{u1}UpdatedItinerary \times \rho_{u2}UpdatedItinerary)\Big)$$

$$DesiredTripInst(instID, tripID) := \Pi_{instID,tripID}(DesiredTrip \bowtie TripInstance)$$

$$Bought(travID, tripID) := \Pi_{travID,tripID}(DesiredTripInst \bowtie Booking)$$

$$PeopleBought(travID) := \Pi_{travID}(Bought)$$

$$ConstructedBought(travID, tripID) := (PeopleBought \times DesiredTrip)$$

$$Failures(travID) := \Pi_{travID}(ConstructedBought - Bought)$$

$$Q3Ans(fname, lname, email) :=$$
$$\Pi_{fname,lname,email}\big[(PeopleBought - Failures) \bowtie Traveller\big]$$

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.

   Answer:

   Cannot be expressed.
   Because wee need to use aggregate function, `count()` for counting the base activities for a given tripID. However, since the `count` function is not supported and unable to be constructed with limited operators.

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.

   Answer:

$$NotFirstStartDate(startDate, staffID) :=$$
$$\Pi_{t_1.startDate,t_1.tripManager}\Big(\sigma_{t_1.tripManager=t_2.tripManager \wedge t_1.startDate>t_2.startDate}$$
$$(\rho_{t_1}TripInstance \times \rho_{t_2}TripInstance)\Big)$$

$$FirstStartDate(startDate, staffID) :=$$
$$(\Pi_{startDate,tripManager}TripInstance) - NotFirstStartDate$$

$$RenamedTripInstance(instID, tripID, staffID, startDate) :=$$
$$\Pi_{instID,tripID,tripManager,startDate}TripInstance$$

$$StaffInfo := Staff \bowtie RenamedTripInstance \bowtie FirstStartDate$$

$$Q5Ans(staff1, start1, staff2, start2) :=$$
$$\Pi_{S_1.email, S_1.startDate, S_2.email, S_2.startDate}$$
$$\Big(\sigma_{\substack{S_1.tripID=S_2.tripID \\ \wedge \\ S_1.instID \neq S_2.instID \\ \wedge \\ S_1.employDate \leqslant S_2.employDate}} (\rho_{S_1} StaffInfo \times \rho_{S_2} StaffInfo)\Big)$$

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.

   Answer:

– have two more distinct cities other than the start city (Paris)

$$MoreThanOneTrip(travID, instID) :=$$
$$\Pi_{b1.travID, b1.instID}\Big(\sigma_{\substack{b1.travID=b2.travID \\ \wedge \\ b1.instID \neq b2.instID}}(\rho_{b1} Booking \times \rho_{b2} Booking)\Big)$$

$$AddOnBooked(instID, travID, addID) :=$$
$$\Pi_{instID, travID, addID}(MoreThanOneTrip \bowtie AddOnBooking)$$

$$AddOnAvailable(instID, travID, addID) :=$$
$$\Pi_{instID, travID, addID}(TripInstance \bowtie AddOnActivity \bowtie MoreThanOneTrip)$$

$$DesiredTraveller(travID) :=$$
$$\Pi_{travID} AddOnBooked - \Pi_{travID}(AddOnAvailable - AddOnBooked)$$

$$Q6Ans(email) := \Pi_{email}(DesiredTraveller \bowtie Traveller)$$

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.

   Answer:

– For travellers who've given at least one review, we want to find the ones whose every review either has low-ball rating or only has text consisting of the word "terrible"
– Travellers who doesn't satisfy this are those who (1) gives more than two stars for a review and the text of the review is not just the word "terrible", or (2) whose review is not the lowest rating for a trip and the text of the review is not just the word "terrible" or (3) give the same number of stars as some other people in a review and the text of the review is not just the word "terrible"

$AtleastOneReview(travID) := \Pi_{travID}(Review)$

$$SameStarNotTerrible(travID) := \Pi_{r1.travID}\sigma_{\substack{r1.tripID=r2.tripID \\ \wedge \\ r1.travID\neq r2.travID \\ \wedge \\ r1.star=r2.star \\ \wedge \\ r1.text\neq' terrible'}}(\rho_{r1}Review \times \rho_{r2}Review)$$

$$NotLowestRatingNotTerrible(travID) := \Pi_{r1.travID}\sigma_{\substack{r1.tripID=r2.tripID \\ \wedge \\ e1.stars>r2.stars \\ \wedge \\ r1.text\neq' terrible'}}(\rho_{r1}Review \times \rho_{r2}Review)$$

$$MoreThan2StarsNotTerrible(travID) := \Pi_{travID}\left(\sigma_{\substack{stars>2 \\ \wedge \\ text\neq' terrible'}} Review\right)$$

$$PeopleSatisfy(travID) := AtleastOneReview - SameStarNotTerrible - NotLowestRatingNotTerrible - Morethan2StarsNotTerrible$$

$$Q7Ans := \Pi_{email}(PeopleSatisfy \bowtie Traveller)$$

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.

Answer:

Cannot be expressed.
Because we cannot find 'above the average' in relational algebra.

Part 2: Additional 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.

   Answer:

$$AccOffered(instID, cityID, roomSize) :=$$
$$\Pi_{instID,cityID,roomSize}(Booking \bowtie TripAccommodation \bowtie Accommodation \bowtie Room)$$

$$NotBiggestAcc(instID, cityID, roomSize) :=$$
$$\Pi_{\substack{a2.instID,\\ a2.cityID,\\ a2.roomSize}} \left(\sigma_{\substack{(a1.instID=a2.instID) \\ \wedge \\ (a1.cityID=a2.cityID) \\ \wedge \\ (a1.roomSize>a2.roomSize)}} (\rho_{a1} AccOffered \times \rho_{a2} AccOffered)\right)$$

– The biggest accommodation offered by the instances in every city

$$BiggestAcc(instID, cityID, roomSize) := AccOffered - NotBiggestAcc$$

– The travellers needed room type in the start city of the trip instance

$$NeededRoomStartCity(instID, travID, roomType, cityID) :=$$
$$\Pi_{instID,travID,roomType,startCity}(Booking \bowtie TripInstance \bowtie Trip)$$

$$NeededRoomItinerary(instID, travID, roomType, cityID) :=$$
$$\Pi_{instID,travID,roomType,cityID}(Booking \bowtie TripInstance \bowtie Itinerary)$$

$$NeededRoom(instID, travID, roomType, cityID) :=$$
$$NeededRoomStartCity \cup NeededRoomItinerary$$

$$\sigma_{roomSize<roomType}(BiggestAcc \bowtie NeededRoom) = \emptyset$$

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.)

   Answer:

$$\Pi_{tripID,travID} Review - \Pi_{tripID,travID}(Booking \bowtie TripInstance) = \emptyset$$

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.

Answer on the next page:

$$TwoOrMoreAddOns(tripID) :=$$
$$\Pi_{tripID}\Big(\sigma_{\substack{a1.tripID=a2.tripID \\ \wedge \\ a1.addID \neq a2.addID}}(\rho_{a1}AddOnActivity \times \rho_{a2}AddOnActivity)\Big)$$

– Add-ons in trips with two or more add-on activities

$$SatisfiedAddOn(addID, tripID, cost) :=$$
$$\Pi_{addID,tripID,cost}(TwoOrMoreAddOns \bowtie AddOnActivity)$$

– Find most expensive

$$NotMostExpensive(tripID, addID) :=$$
$$\Pi_{s2.tripID,s2.addID}\sigma_{\substack{s1.tripID=s2.tripID \\ \wedge \\ s1.addID \neq s2.addID \\ \wedge \\ s1.cost > s2.cost}}(\rho_{s1}SatisfiedAddOn \times \rho_{s2}SatisfiedAddOn)$$

$$MostExpensive(tripID, addID) := \Pi_{tripID,addID}SatisfiedAddOn - NotMostExpensive$$

– Find second most expensive

$$WhatLeft(tripID, addID, cost) := NotMostExpensive \bowtie SatisfiedAddOn$$

$$NotSecondMostExpensive(tripID, addID) :=$$
$$\Pi_{s2.tripID,s2.addID}\sigma_{\substack{s1.tripID=s2.tripID \\ \wedge \\ s1.addID \neq s2.addID \\ \wedge \\ s1.cost > s2.cost}}(\rho_{s1}WhatLeft \times \rho_{s2}WhatLeft)$$

$$SecondMostExpensive(tripID, addID) :=$$
$$(\Pi_{tripID,addID}WhatLeft) - NotSecondMostExpensive$$

– Combine most expensive and the second most expensive

$$FirstSecondMostExpensive(tripID, addID) := MostExpensive \cup SecondMostExpensive$$

$$FirstSecondMostExpensiveInst(instID, addID) :=$$
$$\Pi_{instID,addID}(FirstSecondMostExpensive \bowtie TripInstance)$$

$$PremiumTraveller(instID, travID) :=$$
$$\Pi_{instID,travID}(FirstSecondMostExpensiveInst \bowtie AddOnBooking)$$

$$NotPremiumTraveller(instID, travID, pricePaid) :=$$
$$\Pi_{instID,travID,pricePaid}\big[(\Pi_{instID,travID}Booking - PremiumTraveller) \bowtie Booking)\big]$$

$$\sigma_{pricePaid<price}(NotPremiumTraveller \overset{8}{\bowtie} TripInstance) = \emptyset$$