Csc343

Assign 1

Xian Zhang 1000564766

Qili Lu 1000354537

Unary operators on relations:

Πx, y, z(R)

σ*condition* (R)

ρNew (R)

ρNew(a, b, c) (R)

Binary operators on relations:

R × S

R ⋈ S

R ⋈*condition* S

R ∪ S

R ∩ S

R − S

Logical operators: ¬∧∨

ssignment:

New(a, b, c) := R

1. Find all concerts in Toronto in 2016 that have one or more unsold seats costing under $25. Report the event ID.

–Unsold ticket

Unsold(eID, sID) := ΠeID,sID (Ticket) - ΠeID,sID (Purchase)

*–Concert that under 25 in 2016*

CheapConcert2016(eID,vID) := ΠeID,vID (σ*type=’concert’ and price<25 and when.year=2016* (Event ⋈ Unsold ⋈*Event.venue=Venue.vID* Venue

) );

*-that in Toronto*

Answer := ΠeID (σ*city=’Toronto’* (CheapConcert2016⋈ Venue) );

2. Find all users who have paid at least $200 for some ticket, but have never bought a ticket to a musical. You might call these people “big spenders” who hate musicals. For each of them, find all the tickets they’ve bought for over $200. Report the username, ticket price, event ID, event date and time, and event name.

*-spender who spent more than 200*

bigSpender(eID,username) := ΠeID,username (σ*price>200* (Purchase ⋈ Ticket) );

*-who spent in musical events*

musicalSpender(username) := Πusername(σ*type=musical* (bigSpender ⋈ Event) );

NotMusical(username) := ΠusernamebigSpender – musicalSpender

Answer :=Πusername, price, eID, when, name  bigSpender ⋈ Event ⋈ NotMusical

3. Find all users who, in two consecutive years, have bought multiple tickets for a single event. Report their user names and email addresses.

MultipleBuyer (username,when) := Πusername,event.when (σ*p1.eID=p2.eID and (not (p1.sID = p2.sID)) and p1.username=p2.username* (ρp1(Purchase) × ρp2(Purchase) ⋈ Event) ;

ConsecutiveYear(username):= Πusername σ*m1.username=m2.username and m2.when.year-m1.when,year=1* ((ρm1(MultipleBuyer) × ρm2(MultipleBuyer));

Answer := Πusername,email (ConsecutiveYear ⋈ User);

4. Find all events in 2015 or earlier for which none of the seats at the top price were sold, but every seat at a lower price was sold. Report the event ID and event name.

*-all events before 2015*

Event2015(eID,name) := ΠeID,name σ *when.year<=2015* Event;

NotMax(eID,sID) := ΠeID,sID σ *t1.eID=t2.eID and t1.sID=t2.sID and t1.price<t2.price* ((ρt1(Ticket) × ρt2(Ticket));

MaxPrice(eID,sID) := ΠeID,sIDTicket – NotMax

InvalidMax(eID):= ΠeID (MaxPrice ∩ ΠeID,sID Purchase)

SoldLower(eID,sID):=ΠeID,sID (Purchase ⋈ NotMax)

InvalidLower(eID):=ΠeID (NotMax - SoldLower)

ValidEvent(eID) := ΠeID Event2015- InvalidMax - InvalidNotMax

Answer := ΠeID,name (ValidEvent ⋈ Event);

5. For each venue in New York, find the least expensive and the most expensive ticket price for a seat in that venue (for any event) in 2015. Report the venue ID, venue name, lowest price and highest price.

*-Venue in Newyork and in 2015*

NewYorkVenue (vID,name,eID) := ΠvID,Venue.name,eID σ *city=’New York’ Event.when.year=2015* (Venue ⋈ Event)

*-find the price for desired venue*

VenuePrice (vID,name,eID,price) := ΠvID,Venue.name,eID,price σ *city=’New York’* (NewYorkVenue ⋈ Ticket)

*-try to find the highest price, find those not highest first*

NotMax (vID,name,eID,price):= σ *t1.vId=t2.vID and t1.price<t2.price* ((ρt1(VenuePrice) × ρt2(VenuePrice));

*-try to find the lowest price , find those not lowest first*

NotMin (vID,name,eID,price):= σ *t1.vId=t2.vID and t1.price>t2.price* ((ρt1(VenuePrice) × ρt2(VenuePrice));

HighestPrice (vID,name,eID,highestPrice) := VenuePrice-NotMax

LowestPrice (vID,name,eID,lowestPrice):= VenuePrice-NotMin

Answer := (HighestPrice⋈ LowestPrice)

6. Find the venue with the greatest number of accessible seats. Report the venue name and city.

Cannot be expressed.

7. Find every event for which one user bought every ticket for an accessible seat. Report the event name, date and city, and username of the person who bought all the accessible seats.

*-Find all accessible seat*

AccessibleSeat(eID,sID,username) := ΠeID,sID,username σ *accessible*  Seat ⋈ Ticket;

-asscessible seats that is unsold

UnsoldSeat(eID,sID) := ΠeID,sID AccessibleSeat – ΠeID,sID Purchase

*-events that all accessible seats were bought*

DesiredEvent (eID, sID, username) := (ΠeID AccessibleSeat –ΠeIDUnsoldSeat) ⋈ AccessibleSeat

*-those events that is purchased by more than one among desired events*

NotPurchasedbyOne(eID,sID,username) := Π d1.eID,d1.sID,d1.username σ *d1.eID=d2.eID and (not (d1.username =d2.username))* (ρd1 (DesiredEvent)× ρd2 (DesiredEvent))

PurchasedbyOne (eID,sID,username):= DesiredEvent- NotPurchasedbyOne

Answer := ΠEvent.name,Event.when,city,username (PurchasedbyOne⋈ Event) ⋈*Event.venue=Venue.vID* Venue

8. Find the events in Toronto in 2015 at which at least half of the seats were unsold. Report the event ID, name and date.

Cannot be expressed.

9. Find all users who have bought a ticket to at least one event, but have never bought two or more tickets to one event. Report the username, last name and first name.

*-users who bought more thant 2 tickets*

Morethan2 (username): Πusername σ *p1.username=p2.username and not p1.eID=p2.eID*  (ρp1 (Purchase) × ρp2 (Purchase))

Morethan1 (username) := Πusername  Purchase

Answer := Πusername,lastName,firstName ((Morethan1-Morethan2) ⋈ User)

10. Find all users who have bought a ticket for each concert that the Rolling Stones have played in Toronto in 2000 or since. Report the usernames.

*- concert that the Rolling Stones have played in Toronto in 2000 or since*

DesiredEvent (eID) := Π eID σ *when.year>=2000 and city= ‘Toronto’ and type=’concert’ and name = ‘Rolling Stones’* (Event ⋈*Event.venue=Venue.vID* Venue)

*-find all possible desired result*

Checklist(username,eID) := Πusername,eID (User × DesiredEvent)

*-results that are not satisfactory*

Missed (username,eID) := Checklist - Πusername,eID Purchase

Answer := (Πusername User - Πusername Missed)

11. Find all venues at which the Rolling Stones have played a sold out concert (i.e. the event is a concert and its name is “Rolling Stones”). For each of these venues, report the name of the owner of the venue.

DesiredEvent (eID):= Π eID σ*type=’concert’ and name = ‘Rolling Stones’* Event

Checklist (eID,sID) := Π eID,sID (DesiredEvent ⋈Ticket)

InvalidEvent(eID,sID) := Checklist - Π eID,sID Purchase

Answer := Π owner ((DesiredEvent- Π eID,InvalidEvent) ⋈Event ⋈*Event.venue=Venue.vID* Venue)

12. Find all users who bought a ticket for either the first talk or the second talk in Toronto in 2016. Report their email addresses. Note: There may be only one talk in Toronto in 2016, in which case, the only people in the answer will have bought a ticket to that first (and only) talk. If there is no talk in Toronto in 2016, the answer should be an empty relation.

*-all talk type events that are in Toronto in 2016*

DesiredEvent(eID,when) := Π eID,when σ*type=’talk’ and when.year = ‘2016’ and city=’Toronto’* (Event ⋈*Event.venue=Venue.vID* Venue)

NotFirstTalk(eid,when) := Π d1.eID,d1.when σ *d1.eID=d2.eID and d1.when > d2.when* (ρd1 (DesiredEvent)× ρd2 (DesiredEvent))

FirstTalk(eid,when) := DesiredEvent - NotFirstTalk

NotSecondTalk(eid,when) := Π d1.eID,d1.when σ *d1.eID=d2.eID and d1.when > d2.when* (ρd1 (NotFirstTalk)× ρd2 (NotFirstTalk))

SecondTalk(eid,when) := NotFirstTalk - NotSecondTalk

Answer := Π email (User ⋈ Purchase ⋈ (FirstTalk ∪ SecondTalk ))

Part2.

1. A ticket for an event must be for a seat in the same venue as the event venue.

Π venue (Ticket ⋈ Seat) - Π venue (Ticket ⋈ Event) =∅

2. A ticket for an event cannot be purchased after the event.

ΠeID,sIDσPurchase.when>Event.when (Purchase ⋈*Purchase.eID=Event.eID* Event)=∅