1 Part 1

id	name	country
1	Toronto	Canada
2	Rome	Italy
3	Frankfurt	Germany

Table 1: City

code	name
1	Air Canada
2	Lufthansa

Table 2: Airline

id	firstName	surName
1	Sadia	Li

Table 3: Passenger

tailNumber	model	airline
1	Boeing 777	1
2	Boeing 777	2
3	Boeing 777	2

Table 4: Plane

id	plane	row	letter	class
1	1	1	A	Economy
2	2	2	A	Business
3	3	3	A	Business

Table 5: Seat

code	name	city
YYZ	Toronto Pearson International Airport	1
FCO	Leonardo da Vinci International Airport	2
FRA	Frankfurt Airport	3

Table 6: Airport

flightNumber	airline	source	destination
AC890	1	YYZ	FCO
LH231	2	FCO	FRA
LH470	2	FRA	YYZ

Table 7: Route

id	route	plane	schedDeparture	schedArrival
1	AC890	1	2025-05-01 23:40	2025-05-02 8:05
2	LH231	2	2025-05-22 8:00	2025-05-22 10:00
3	LH470	3	2025-05-22 11:55	2025-05-22 20:20

Table 8: Flight

flight	dateTime
1	2025-05-01 23:45
2	2025-05-22 8:00
2	2025-05-22 11:55

Table 9: Departure

flight	dateTime
1	2025-05-02 8:15
2	2025-05-22 10:00
2	2025-05-22 20:20

Table 10: Arrival

flight	class	price
1	Economy	600
2	Business	950
3	Business	1200

Table 11: FlightPrice

id	passenger	seat	flight	price	dateTime
1	1	1	1	600	2024-12-06 21:00
1	2	2	2	950	2025-05-09 9:00
1	3	3	3	1200	2025-05-09 9:00

Table 12: Booking

2 Part 2

The result of the given query is:

id	passenger	seat	flight
1	1	1	1
2	1	2	2
3	1	3	3

Table 13: Part 2 answer

3 Part 3

id	name	country
1	Toronto	Canada
2	Rome	Italy
3	Frankfurt	Germany

Table 14: City

code	name
1	Air Canada
2	Lufthansa

Table 15: Airline

id	firstName	surName
1	Sadia	Li
2	Sadia	Li
3	Sadia	Li

Table 16: Passenger

tailNumber	model	airline
1	Boeing 777	1
2	Boeing 777	2
3	Boeing 777	2

Table 17: Plane

id	plane	row	letter	class
1	1	1	A	Economy
4	1	1	B	Economy
2	2	2	A	Business
3	3	3	A	First

Table 18: Seat

code	name	city
YYZ	Toronto Pearson International Airport	1
FCO	Leonardo da Vinci International Airport	2
FRA	Frankfurt Airport	3

Table 19: Airport

flightNumber	airline	source	destination
AC890	1	YYZ	FCO
LH231	2	FCO	FRA
LH470	2	FRA	YYZ

Table 20: Route

id	route	plane	schedDeparture	schedArrival
1	AC890	1	2025-05-01 23:45	2025-05-02 8:15
2	LH231	2	2025-05-22 8:00	2025-05-22 10:00
3	LH470	3	2025-05-22 11:55	2025-05-22 20:20

Table 21: Flight

flight	dateTime
1	2025-05-01 23:50
2	2025-05-22 8:00
2	2025-05-22 11:55

Table 22: Departure

flight	dateTime
1	2025-05-02 8:25
2	2025-05-22 10:00
2	2025-05-22 20:20

Table 23: Arrival

flight	class	price
1	Economy	600
2	Business	950
3	First	1200

Table 24: FlightPrice

id	passenger	seat	flight	price	dateTime
1	1	1	1	600	2024-12-06 21:00
4	1	4	1	600	2024-12-06 21:00
2	2	2	2	950	2025-05-09 9:00
3	3	3	3	1200	2025-05-09 9:00

Table 25: Booking

4 Part 4

1. Not expressable

2. - *Rename Flight table for natural join.*

   FlightRename(flight, tailNumber, schedDeparture, flightNumber) := $\Pi_{id,plane,schedDeparture,route}(\textbf{Flight})$

   - *natural join Flight and Departure to get the delayed flights in 2024.*

   DelayedFlight(flight, tailNumber, schedDeparture,dateTime, flightNumber) :=

   $$\sigma_{\substack{dateTime-schedDeparture\geq 1hour \\ \bigcap \\ schedDeparture.year=2024}}\left(\textbf{FlightRename} \bowtie \textbf{Departure}\right)$$

   - *Natural join delayed flights and plane to get the airline of the flight.*

   DelayedFinal(flight, tailNumber, schedDeparture,dateTime,airline, flightNumber) :=
   $\Pi_{flight,tailNumber,schedDeparture,dateTime,airline,flightNumber}(\textbf{DelayedFlight} \bowtie \textbf{Plane})$

   - *Get the the airlines which have at least three different delayed flights in 2024*

   $$\textbf{AtLeastThree}(\text{airline}) := \Pi_{airline}\sigma_{\substack{T1.airline=T2.airline \\ \bigcap \\ T2.airline=T3.airline \\ \bigcap \\ T1.flight\neq T2.flight \\ \bigcap \\ T2.flight\neq T3.flight \\ \bigcap \\ T1.flight\neq T3.flight}}\left(\rho_{T_1}\textbf{DelayedFinal}\times\rho_{T_2}\textbf{DelayedFinal}\times\rho_{T_3}\textbf{DelayedFinal}\right)$$

   - *Natural join the above airlines with those delayed flights*

   AtLeastThreeAllFlights(flight,airline, scheDeparture, flightNumber):=
   $\Pi_{flight,airline,schedDeparture,flightNumber}(\textbf{AtLeastThree} \bowtie \textbf{DelayedFinal})$

   - *Exclude the most recent flight.*

   NotMostRecentFlight(flight, airline, scheDeparture, flightNumber) :=

   $$\Pi_{T4.flight,airline,T4.schedDeparture,T4.flightNumber}\left(\sigma_{\substack{T4.airline=T5.airline \\ \bigcap \\ T4.schedDeparture<T5.schedDeparture}}\left(\rho_{T_4}(AtLeastThreeAllFlights)\times\rho_{T_5}(\textbf{AtLeastThreeAllFlights})\right)\right)$$

   - *Get the most recent flights by subtracting flights that are not most recent*

   AtLeastThreeMostRecentFlights(flight,code,flightNumber) :=
   $\Pi_{flight,airline,flightNumber}(\textbf{AtLeastThreeAllFlights} - \textbf{NotMostRecentFlight})$

   - *Report final answer*

   $\Pi_{code,name,flightNumber}(\textbf{AtLeastThreeMostRecentFlights} \bowtie \textbf{Airline})$

3. ▪ *All cities in the airport relation.*

$\textbf{CityWithAirport}_{(city)} := \Pi_{city}(\textbf{Airport})$

▪ *All cities associated with airport that is destination for at least once.*

$\textbf{CityWithRouteEnding}_{(city)} := \Pi_{city}(\textbf{Airport} \bowtie Airport.code = Route.destination \textbf{Route})$

▪ *Subtraction gives cities that has never been a end of a route.*
Joining with City relation and projecting their name gives the correct answer.

$\textbf{CityWithoutRouteEnding}_{(city)} := \textbf{CityWithAirport} - \textbf{CityWithRouteEnding}$

$\textbf{Answer} := \Pi_{name,city}(\textbf{CityWithoutRouteEnding} \bowtie_{CityWithoutRouteEnding.city=city.id} \textbf{City})$

4. ▪ *Join three relations, Flight, Route, and FlightPrice.*

$\textbf{FlightRoutePrice} := \textbf{Flight} \bowtie_{Flight.route=Route.flightNumber} \textbf{Route} \bowtie_{Flight.id=FlightPrice.flight} \textbf{FlightPrice}$

▪ *Filtering out the exact condition given.*

$$\textbf{SameRoutePair} := \sigma_{\substack{R1.id \neq R2.id \\ R1.airline \neq R2.airline \\ R1.source \neq R2.source \\ R1.destination \neq R2.destination \\ (R1.price < R2.price) \vee (R1.price = R2.price \wedge R1.id < R2.id)}} \left(\rho_{R1}(\textbf{FlightRoutePrice}) \times \rho_{R2}(\textbf{FlightRoutePrice}) \right)$$

$\textbf{Answer} := \Pi_{R1.id, R1.flightNumber, R2.id, R2.flightNumber}(\textbf{SameRoutePair})$

5. ▪ *Join Flight and Route relation and project out only the necessary information.*

$\textbf{F}_{(id,source,destination,schedDeparture,schedArrival)}$
$:= \pi_{id,source,destination,schedDeparture,schedArrival}(\textbf{Flight} \bowtie_{Flight.route=Route.flightNumber} \textbf{Route})$

▪ *Filter out those whose scheduled arrival time is after June 17th.*

$\textbf{AfterJune17}_{(id,source,destination,schedDeparture,schedArrival)} := \sigma_{schedArrival > 2025-06-17} \textbf{F}$

▪ *Direct flights are just those whose source and destination matches directly. For layover, use the select condition below.*

$\textbf{Direct}_{(schedDeparture,schedArrival)} := \pi_{schedDeparture,schedArrival} \left(\sigma_{\substack{source='YYZ' \\ \wedge destination='LIS'}} \textbf{AfterJune17} \right)$

$\textbf{Layover}_{(schedDeparture,schedArrival)}$

$$:= \pi_{\substack{F1.schedDeaparture, \\ F2.schedArrival}} \left(\sigma_{\substack{F1.source='YYZ' \\ F1.destination=F2.source \\ F2.destination='LIS' \\ 1hour \leq F2.schedDeparture - F1.schedArrival \leq 24hours}} \left(\rho_{F1}(AfterJune17) \times \rho_{F2}(AfterJune17) \right) \right)$$

▪ *Union of Direct and Layover gives the desired answer.*

$\textbf{YYZToLIS}_{(schedDeaparture,schedArrival)} := Direct \cup Layover$

6. ▪ *Rename booking for the following natural join*
BookingRename(id,flight) := $\Pi_{passenger,flight}$(**Booking**).

▪ *Natural join delayed flights and plane to get the airline of the flight.*
FlightRename(flight, schedDeparture) := $\Pi_{id,schedDeaparture}$(**Flight**).

▪ *Get all pairs of passengers who booked different flights or the pair contain one person twice.*
PairWithDiffTirp(id1,flight1,schedDeparture1,id2,flight2,schedDeparture2):=

$$\Pi_{T1.id,T1.flight,T1.schedDeparture1,T2.id,T2.flight,T2.schedDeparture1}(\sigma_{\substack{T1.flight\neq T2.flight\\ \bigcup\\ T1.id\geq T2.id}}\Big(\rho_{T_1}(\textbf{FlightRename} \bowtie \textbf{BookingRename})\times \rho_{T_2}(\textbf{FlightRename} \bowtie \textbf{BookingRename})\Big))$$

▪ *Project the ids of these pairs*
InvalidPairs(id1,id2):=$\Pi_{id1,id2}$(**PairWithDiffTirp**)

▪ *Get all possible pairs.*
AllPairs(id1,id2):=$\Pi_{id1,id2}(\rho_{T_1}(\textbf{FlightRename} \bowtie \textbf{BookingRename})\times\rho_{T_2}(\textbf{FlightRename} \bowtie \textbf{BookingRename})$

▪ *Get valid pairs by subtracting invalid pairs from all pairs.*
ValidPairs(id1,id2):=$\Pi_{id1,id2}(AllPairs - InvalidPairs)$

▪ *Get all pairs of passengers who have same surname.*

PairsOfSameName(id1,id2):=$\Pi_{id1,id2}\sigma_{\substack{T3.surName=T4.surname\\ \bigcap\\ T3.id<T4.id}}\Big(\rho_{T_3}\textbf{Passenger}\times\rho_{T_4}\textbf{Passenger}\Big)$

▪ *The intersection of pair of passengers who always book the same flight and pair of passengers who have the same surname is what we want*
FinalPair(id1,id2) := **ValidPairs** $\bigcap$ **PairsOfSameName**

▪ *Exclude the most recent trip for each pair*
GetNotMostRecentTrip(id1,id2,flight) :=

$$\Pi_{T5.id1,T5.id2,T5.flight}\ (\sigma_{\substack{T5.schedDeparture<T6.schedDeparture\\ \bigcap\\ T5.id1=T6.id1\\ \bigcap\\ T5.id2=T6.id2}}\Big((\textbf{FinalPair} \bowtie_{id1=id} (\textbf{BookingRename} \bowtie \textbf{Flight}))\bowtie_{\substack{T5.id1=T6.id1\\ \bigcap\\ T5.id2=T6.id2}} (\textbf{FinalPair} \bowtie_{id1=id} (\textbf{BookingRename} \bowtie \textbf{Flight})))\Big)$$

▪ *Get all trips for each pair*
AllTrip(id1,id2,flight):=$\Pi_{T5.id1,T5.id2,flight}(\sigma_{\substack{T5.id1=T6.id1\\ \bigcap\\ T5.id2=T6.id2}}(\textbf{FinalPair} \bowtie_{id1=id} (\textbf{BookingRename} \bowtie \textbf{Flight})))$

▪ *Most recent trip is all trip minus trips excluded the most recent trip*
MostRecentTrip(id1,id2,flight) := **AllTrip - GetNotMostRecentTrip**

▪ *report final answer*

$$\Pi_{P_1.firstName,P_1.surName,P_2.firstName,P_2,surName,flight}(\sigma_{\substack{P_1.id=id1\\ \bigcap\\ P_2.id=id2}}(\textbf{MostRecentTrip}\times\rho_{P_1}\textbf{Passenger}\times\rho_{P_2}\textbf{Passenger}))$$

7. ■ *Get tailNumber of planes that are used at least four times*

$$\textbf{AtLeastFourTimes}(\text{tailNumber}) := \Pi_{plane}(\sigma_{\substack{T_1.plane=T_2.plane=T_3.plane=T_4.plane \\ \bigcap \\ T_1.id \neq T_2.id \neq T_3.id \neq T_4.id}}\left(\rho_{T_1}\textbf{Flight} \times \rho_{T_2}\textbf{Flight} \times \rho_{T_3}\textbf{Flight} \times \rho_{T_4}\textbf{Flight}\right))$$

■ *Get tailNumber of all planes* $\textbf{AllPlanes}(\text{tailNumber}) := \Pi_{tailNumber}(\textbf{Plane})$

■ *All planes minus planes used for at least four times gives planes used less than four times*

PlanesLessThanFour(tailNumber) := **AllPlanes - AtLeastFourTimes**

■ *Report final answer.*

$$\Pi_{tailNumber,airline}(\textbf{PlanesLessThanFour} \bowtie \textbf{Plane})$$

8. ■ *Get passenger who ever booked at least two seats in one flight*

$$\textbf{PassengerEverBookAtLeastTwo}(\text{id}) := \Pi_{T_1.passenger}\sigma_{\substack{T_1.passenger=T_2.passenger \\ \bigcap \\ T_1.flight=T_2.flight \\ \bigcap \\ T_1.seat \neq T_2.seat}}(\rho_{T_1}\textbf{Booking} \times \rho_{T_2}\textbf{Booking})$$

■ *Get passenger who ever booked at least three seats in one flight*

$$\textbf{PassengerEverBookAtLeastThree}(\text{id}) := \Pi_{T_3.passenger}\sigma_{\substack{T_3.passenger=T_4.passenger=T_5.passenger \\ \bigcap \\ T_3.flight=T_4.flight=T_5.flight \\ \bigcap \\ T_3.seat \neq T_4.seat \neq T_5.seat}}(\rho_{T_3}\textbf{Booking} \times \rho_{T_4}\textbf{Booking} \times \rho_{T_5}\textbf{Booking})$$

■ *Get passenger who ever booked two seats in a flight but never booked more than two seats in a flight.*

PassengerEverBookAtMostTwo(id):=$\Pi_{passenger}$ **PassengerEverBookAtLeastTwo - PassengerEverBookAtLeast-Three**

■ *Get (passenger,flight) where a passenger booked more than seat in a flight.*

$$\textbf{PassengerAndFlightWithAtLeastTwoSeats}(\text{id,flight}) := \Pi_{T_1.passenger,T_1.flight}\sigma_{\substack{T_1.passenger=T_2.passenger \\ \bigcap \\ T_1.flight=T_2.flight \\ \bigcap \\ T_1.seat \neq T_2.seat}}(\rho_{T_1}\textbf{Booking} \times \rho_{T_2}\textbf{Booking})$$

■ *Get all (passenger,flight).*

$$\textbf{PassengerAndFlight}(\text{id,flight}) := \Pi_{T_1.passenger,T_1.flight}(\rho_{T_1}\textbf{Booking} \times \rho_{T_2}\textbf{Booking})$$

■ *Get passenger who ever booked only one seat in a flight*

PassengerOnlyBookOne(id):=$\Pi_{passenger}$ (**PassengerAndFlight - PassengerAndFlightWithAtLeastTwoSeats**)

■ *passenger who ever booked two seats in a flight but never booked more than two seats in a flight minus passenger who ever booked only one seat in a flight is passengers who always book two seats in every flight.*

PassengerAlwaysBookTwo(id):= **PassengerEverBookAtMostTwo - PassengerOnlyBookOne**

■ *Get passengers who ever book seats in the same row*

$$\textbf{PassengerBookSameRow}(\text{id}) := \Pi_{T_6.passenger}\sigma_{\substack{T_6.passenger=T_7.passenger \\ \bigcap \\ T_6.flight=T_7.flight \\ \bigcap \\ T_6.seat\neq T_7.seat \\ \bigcap \\ T_6.row=T_7.row}} (\rho_{T_6}\textbf{Booking} \bowtie_{Booking.seat=Seat.id} \textbf{Seat}) \times \rho_{T_7}\textbf{Booking} \bowtie_{Booking.seat=Seat.id} \textbf{Seat})$$

- *Get passengers who never book two seats in the same row*

$\textbf{PassengerNeverBookSameRow}(\text{id}) := \Pi_{passenger}\textbf{Booking - PassengerBookSameRow}$

- *Passengers who always book two seats in a flight and never book two seats in the same row is what we want*

$\textbf{PassengerNeverBookSameRow} \bigcap \textbf{PassengerAlwaysBookTwo}$

9. Not expressable

- *Get Passengers who ever pay equal or more price for a seat*

10.
- *Get Passengers who ever pay equal or more price for a seat*

PassengerPaidEqualOrMore(id):=$\Pi_{Passenger}\sigma_{Booking.price \geq FlightPrice.price}((\textbf{Booking} \bowtie_{FlightPrice.flight=Booking.flight} \textbf{FlightPrice}) \bowtie_{\substack{Booking.seat=Seat.id \\ \cap \\ FlightPrice.class=Seat.class}} \textbf{Seat})$

- *The rest is passengers who always pay stricly less.*

$\Pi_{passenger}$**Booking - PassengerPaidEqualOrMore**

11.
- *Below five expressions are just the renaming of the "vanila version" relation, for easy join and to handle attribute name conflict.*

$\textbf{B}_{(bookId,seat,flight,paidPrice)} := \rho_{B_{(bookid,seat,flight,paidPrice)}}\left(\pi_{id,seat,flight,price}\textbf{Booking}\right)$

$\textbf{F}_{(flightId,route)} := \rho_{F_{(flightId,route)}}\left(\pi_{id,seat,flight,price}\textbf{Flight}\right)$

$\textbf{R}_{(routeId,airline)} := \rho_{R_{(routeId,airline)}}\left(\pi_{flightNumber,airline}\textbf{Route}\right)$

$\textbf{S}_{(seatId,class)} := \rho_{S_{(seatId,class)}}\left(\pi_{id,class}\textbf{Seat}\right)$

$\textbf{P}_{(flight,class,price)} := \rho_{S_{(flight,class,price)}}\textbf{FlightPrice}$

- *Now join all five using relevant attribute and project out only the necessary information. Note that all the route appear in this relation is associated with at least one flight.*

$\textbf{BookDetail}_{(bookId,routeId,price,paidPrice,airline)}$

$$:= \pi_{\substack{B.bookId, \\ R.routeId, \\ P.price, \\ B.paidPrice, \\ R.airline}}\left(\textbf{B} \bowtie_{\substack{B.flight \\ =F.flightId}} \textbf{F} \bowtie_{\substack{F.route \\ =R.routeId}} \textbf{R} \bowtie_{\substack{B.seat \\ =S.seatId}} \textbf{S} \bowtie_{\substack{F.flightId=P.flight \\ \wedge S.class=P.class}} \textbf{P}\right)$$

- *Typical way of getting all popular routes. AllRoutes represents all the routes associated with at least one flight and their airline. UnpopularRoutes represents all the unpopular routes.*

$\textbf{AllRoutes}_{(routeId,airline)} := \pi_{routeId,airline}\textbf{BookDetail}$

$\textbf{UnpopularRoutes}_{(routeId,airline)} := \pi_{routeId,airline}\left(\sigma_{paidPrice \leq price/2}\textbf{BookDetail}\right)$

$\textbf{Unpopular}_{(routeId)} := \pi_{routeId}\textbf{UnpopularRoutes}$

- *Find all airline that has not operated at least one unpopular route.*
This is done by creating a cartesian product of all airlines and route and
subtracting all the cases where the airline operated a particular unpopular route.
At the end, Filter contains all the airlines that has not operated at least one unpopular route.
(operation is assumed to be at least one booking of the flight.)

$\textbf{Filter}_{(airline,routeId)} := \pi_{airline}\textbf{BookDetail} \times \textbf{Unpopular}$

$$\textbf{Filter}_{(airline)} := \pi_{airline}\textbf{Filter}-\left(\pi_{Filter.airline}\left(\sigma_{\substack{Filter.airline=UnpopularRoutes.airline \\ \wedge Filter.route=UnpopularRoutes.route}}\left(\textbf{Filter}\times\textbf{UnpopularRoutes}\right)\right)\right)$$

- *Subtract Filter from airlines that has operated at least one flight, (thus at least one route).*
Then we get airlines that has operated on all popular route.

$\textbf{AirlinesOperatedPopularRoute} := \pi_{airlines}\textbf{RoutesDetail} - \pi_{airlines}\textbf{PopularRoutes}$

Part 5

1.

Cannot be expressed.

2.

■ *In all cases, we expand the target relation (Booking and Departure) using theta join and filter out the unwanted cases. Then we set it empty.*

$\sigma_{\text{Booking.dateTime}\geq\text{Flight.schedDeparture}}\left(\textbf{Booking} \bowtie_{\text{Booking.flight = Flight.id}} \textbf{Flight}\right) = \varnothing$

$\sigma_{\text{Departure.dateTime}<\text{Flight.schedDeparture}}\left(\textbf{Departure} \bowtie_{\text{Departure.flight = Flight.id}} \textbf{Flight}\right) = \varnothing$

$\sigma_{\text{Arrival.dateTime}\leq\text{Departure.dateTime}}\left(\textbf{Departure} \bowtie_{\text{Departure.flight = Arrival.flight}} \textbf{Arrival}\right) = \varnothing$

3.

Cannot be expressed.

4.

Cannot be expressed.

5.

■ *Join Flight and Seat relation by associated plane and project out all the class for each flight.*

$\textbf{AllClassForAllFlight}_{(flight,class)} := \pi_{Flight.id,Seat.class}\left(\sigma_{Flight.plane=Seat.plnae}\left(\textbf{Flight} \times \textbf{Seat}\right)\right)$

■ *All the class for each flight in flight class.*

$\textbf{AllFlightClassInFlightPrice}_{(flight,class)} := \pi_{flight,class}\textbf{FlightPrice}$

■ *Expressing equality using set difference and emptyset.*

$\textbf{AllClassForAllFlight} - \textbf{AllFlightClassInFlightPrice} = \varnothing$

$\textbf{AllFlightClassInFlightPrice} - \textbf{AllClassForAllFlight} = \varnothing$

6.

■ *The below five expressions are just a sequence of theta joins to ultimately build the final relation F. F is a table of flights that contains the plane, the source country, the destination country, and scheduled departure time associated with each flight in the Flight relation.*

$$\mathbf{FR}\begin{pmatrix} id, \\ plane, \\ source, \\ destination, \\ schedDeparture \end{pmatrix} := \pi_{\begin{pmatrix} Flight.id, \\ Flight.plane, \\ Route.source, \\ Route.destination, \\ Flight.schedDeparture \end{pmatrix}} \mathbf{Flight} \bowtie_{\substack{Flight.route \\ =Route.flightNumber}} \mathbf{Route}$$

$$\mathbf{FRA1}\begin{pmatrix} id, \\ plane, \\ source, \\ destination, \\ schedDeparture \\ srcCity \end{pmatrix} := \pi_{\begin{pmatrix} FR.id, \\ FR.plane, \\ FR.source, \\ FR.destination, \\ FR.schedDeparture \\ ASource.city \end{pmatrix}} \mathbf{FR} \bowtie_{\substack{FR.source \\ =ASource.code}} \left(\rho_{ASource}\mathbf{Airport}\right)$$

$$\mathbf{FRA2}\begin{pmatrix} id, \\ plane, \\ source, \\ destination, \\ schedDeparture \\ srcCity \\ destCity \end{pmatrix} := \pi_{\begin{pmatrix} FRA1.id, \\ FRA1.plane \\ FRA1.source, \\ FRA1.destination, \\ FRA1.schedDeparture \\ FRA1.src \\ ADest.city \end{pmatrix}} \mathbf{FRA1} \bowtie_{\substack{FRA1.destination \\ =ADest.code}} \left(\rho_{ADest}\mathbf{Airport}\right)$$

$$\mathbf{FRC1}\begin{pmatrix} id, \\ plane, \\ source, \\ destination, \\ schedDeparture \\ srcCity \\ destCity \\ srcCountry \end{pmatrix} := \pi_{\begin{pmatrix} FRA2.id, \\ FRA2.plane, \\ FRA2.source, \\ FRA2.destination, \\ FRA2.schedDeparture, \\ FRA2.srcCity, \\ FRA2.destCity, \\ CSource.country \end{pmatrix}} \mathbf{FRA1} \bowtie_{\substack{FRA2.srcCity \\ =CSource.id}} \left(\rho_{CSource}\mathbf{City}\right)$$

$$\mathbf{FRC2}\begin{pmatrix} id, \\ plane \\ source, \\ destination, \\ schedDeparture, \\ srcCity, \\ destCity, \\ srcCountry, \\ destCountry \end{pmatrix} := \pi_{\begin{pmatrix} FRC1.id, \\ FRC1.plane, \\ FRC1.source, \\ FRC1.destination, \\ FRC1.schedDeparture, \\ FRC1.srcCity, \\ FRC1.destCity, \\ FRC1.srcCountry \\ CDest.country \end{pmatrix}} \mathbf{FRC1} \bowtie_{\substack{FRC1.destCity \\ =CDest.id}} \left(\rho_{CDest}\mathbf{City}\right)$$

■ *We have successfully built the desired F.*

$$\mathbf{F}_{(id, plane, src, dest, depart)} := \pi_{(id, plane, srcCountry, destCountry, schedDeparture)}\mathbf{FRC2}$$

■ *Flights that are not domestic is the one where the source and destination country is not equal. By subtracting all the planes associated with at least non-domestic flight, we get domestic only flights.*

$$\mathbf{NotDomesticFlights}_{(id, plane, src, dest, depart)} :=:= \sigma_{src \neq dest} F$$
$$\mathbf{DomesticOnlyPlanes}_{(plane)} := \pi_{plane}\mathbf{F} - \pi_{plane}\mathbf{NotDomesticFlights}$$

■ *D is relation of all flights of all domestic planes.*

$$\mathbf{D}_{(id, plane, src, dest, depart)} = \mathbf{F} \bowtie \mathbf{DomesticOnlyPlanes}$$

■ *No domestic planes have four or more domestic flights*

$$\mathbf{FourOrMore}\begin{pmatrix} plane \end{pmatrix}$$
$$:= \pi_{D1.plane}\left(\sigma_{\begin{pmatrix} \forall i,j \in \{1,\ldots,4\}, month(D_i.depart) = month(D_j.depart) \\ \forall i,j \in \{1,\ldots,4\}, D_i.plane = D_j.plane \\ \forall i,j \in \{1,\ldots,4\}, D_i.id \neq D_j.id \end{pmatrix}}\left(\rho_{D_1}(D) \times \rho_{D_2}(D) \times \rho_{D_3}(D) \times \rho_{D_4}(D) \times\right)\right)$$

$$\mathbf{FourOrMore} = \emptyset$$