COMP 421 Database Systems Project Deliverable#02 Group#55

Members: Jonathan Ng, Melody Zhang, Sunny Kim, Van Vai Vivien Lee

Database name: cs421g55; Current Password: cs421h4H4

1. (0 Points) Please attach a copy of your modified relational schema of Project #1 according to the feedback given on it. The new design will not be graded but will be used as a basis for the future assignments.

Entities:

- **Customer(**cust.id, full.name, birth.date, address, phone.number, passport.number, email, driver.license, insurance no., booking.id)
- **Insurance(**insurance.number, company.name, coverage, price)
- **Booking**(booking.id, date.travel , dept.location, arrv.location,price, status, booking.type, disclaimer, no.of.customers, hotel.id, t.id)
- **Employee(**e.id, full.name)
- **Hotel(**hotel.id, name, location, checkin.date, checkout.date, price, roomtype)
- **Attraction(**name, location, price)
- **CarRent(**license.plate, location, model)
- **Transportation(**transport.id, price, dep.date, re.date, dep.time, re.time)
- **Bus**(t.id, dep.route, arrv.route)
- **Airplane**(t.id, dep.fl, arrv.fl)
- **Train**(t.id, dep.tr, arrv.tr)

Weak entities:

-**Package**(booking.id, type, price, description)

Relationships:

- **Consult(**cust.id, eid session.id, date, time)
- **Rent(**cust.id, license.plate, location, rental confirmation number, start.date, end.date)
- **Booking.include.attraction**(booking.id, a.id)
- **Booking.include.transportation**(booking.id,t.id)
- **Booking.include.hotel**(booking.id,h.id)

2. (25 Points) Write a SQL database schema for the relational schema you have designed using the CREATE TABLE command and enter it in the database. Choose suitable data types for your attributes. Indicate primary keys, foreign keys or any other integrity constraints that you can express with the commands learnt. Indicate the constraints you cannot express. The Online Information contains detailed information about data types, and the CREATE TABLE statement.

Turn in your CREATE TABLE statements.Furthermore, (i) for DB2 use command line processor command DESCRIBE tablename (ii) for PostgreSQL use \d table name (prints the description of the relation on the screen) for each of your relations, print the result and turn it in.

Entities

CREATE TABLE Customer(custID VARCHAR(30) PRIMARY KEY, fullName VARCHAR(30), birthday DATE, address VARCHAR(30), phoneNumber VARCHAR(30), passportNum VARCHAR(30), email VARCHAR(30), driverLicense BOOLEAN)

```
cs421=> \d customer
             Table "cs421g55.customer"
    Column     |         Type          | Modifiers
---------------+-----------------------+-----------
 custid        | character varying(30) | not null
 fullname      | character varying(30) |
 birthday      | date                  |
 address       | character varying(30) |
 phonenumber   | character varying(30) |
 passportnum   | character varying(30) |
 email         | character varying(30) |
 driverlicense | boolean               |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (custid)
    "constraint_passportnum" UNIQUE CONSTRAINT, btree (passportnum)
Referenced by:
    TABLE "associates" CONSTRAINT "associates_custid_fkey" FOREIGN KEY (custid) REFERENCES customer(custid)
    TABLE "buys" CONSTRAINT "buys_customerid_fkey" FOREIGN KEY (customerid) REFERENCES customer(custid)
    TABLE "consults" CONSTRAINT "consults_customerid_fkey" FOREIGN KEY (customerid) REFERENCES customer(custid)
    TABLE "rents" CONSTRAINT "rents_customerid_fkey" FOREIGN KEY (customerid) REFERENCES customer(custid)
```

CREATE TABLE Booking(bookingID VARCHAR(30) PRIMARY KEY, travelDate DATE, deptLocation VARCHAR(30), arrvLocation VARCHAR(30), price FLOAT, status VARCHAR(10), bookingType VARCHAR(30), disclaimer VARCHAR(100), noOfCustomers INTEGER)

```
cs421=> \d booking
             Table "cs421g55.booking"
    Column     |          Type           | Modifiers
---------------+-------------------------+-----------
 bookingid     | character varying(30)   | not null
 traveldate    | date                    |
 deptlocation  | character varying(30)   |
 arrvlocation  | character varying(30)   |
 price         | double precision        |
 status        | character varying(10)   |
 bookingtype   | character varying(30)   |
 disclaimer    | character varying(5000) |
 noofcustomers | integer                 |
Indexes:
    "booking_pkey" PRIMARY KEY, btree (bookingid)
Referenced by:
    TABLE "associates" CONSTRAINT "associates_bookingid_fkey" FOREIGN KEY (bookingid) REFERENCES booking(bookingid)
    TABLE "includehotel" CONSTRAINT "includes_bookingid_fkey" FOREIGN KEY (bookingid) REFERENCES booking(bookingid)
    TABLE "includetransport" CONSTRAINT "includetransport_bookingid_fkey" FOREIGN KEY (bookingid) REFERENCES booking(bookingid)
    TABLE "package" CONSTRAINT "package_bookingid_fkey" FOREIGN KEY (bookingid) REFERENCES booking(bookingid)
    TABLE "purchases" CONSTRAINT "purchases_bookingid_fkey" FOREIGN KEY (bookingid) REFERENCES booking(bookingid)
```

CREATE TABLE Employee(eID VARCHAR(30) PRIMARY KEY, fullName VARCHAR(30))

```
cs421=> \d employee
          Table "cs421g55.employee"
  Column   |         Type          | Modifiers
-----------+-----------------------+-----------
 eid       | character varying(30) | not null
 fullname  | character varying(30) |
Indexes:
    "employee_pkey" PRIMARY KEY, btree (eid)
Referenced by:
    TABLE "consults" CONSTRAINT "consults_eid_fkey" FOREIGN KEY (eid) REFERENCES employee(eid)
```

CREATE TABLE Insurance(insuranceNo VARCHAR(30) PRIMARY KEY, companyName VARCHAR(30), coverage FLOAT, price FLOAT)

```
cs421=> \d insurance
          Table "cs421g55.insurance"
   Column    |         Type          | Modifiers
-------------+-----------------------+-----------
 insuranceno | character varying(30) | not null
 companyname | character varying(30) |
 coverage    | double precision      |
 price       | double precision      |
Indexes:
    "insurance_pkey" PRIMARY KEY, btree (insuranceno)
Referenced by:
    TABLE "buys" CONSTRAINT "buys_insuranceid_fkey" FOREIGN KEY (insuranceid) REFERENCES insurance(insuranceno)
```

CREATE TABLE CarRental(licensePlate VARCHAR(10), location VARCHAR(30), model VARCHAR(30), PRIMARY KEY (licensePlate, location))

```
cs421=> \d carrental
          Table "cs421g55.carrental"
    Column    |         Type          | Modifiers
--------------+-----------------------+-----------
 licenseplate | character varying(10) | not null
 location     | character varying(30) | not null
 model        | character varying(30) |
Indexes:
    "carrental_pkey" PRIMARY KEY, btree (licenseplate, location)
Referenced by:
    TABLE "rents" CONSTRAINT "rents_licenseplate_fkey" FOREIGN KEY (licenseplate, location) REFERENCES carrental(licenseplate, location)
```

CREATE TABLE Transportation (transportID VARCHAR(30) PRIMARY KEY, price FLOAT, depDate DATE, returnDate DATE, depTime TIME, returnTime TIME)

```
cs421=> \d transportation
        Table "cs421g55.transportation"
   Column    |          Type          | Modifiers
-------------+------------------------+-----------
 transportid | character varying(30)  | not null
 price       | double precision       |
 depdate     | date                   |
 returndate  | date                   |
 deptime     | time without time zone |
 returntime  | time without time zone |
Indexes:
    "transportation_pkey" PRIMARY KEY, btree (transportid)
Referenced by:
    TABLE "bus" CONSTRAINT "bus_transportid_fkey" FOREIGN KEY (transportid) REFERENCES transportation(transportid)
    TABLE "flight" CONSTRAINT "flight_transportid_fkey" FOREIGN KEY (transportid) REFERENCES transportation(transportid)
    TABLE "includetransport" CONSTRAINT "includetransport_transportid_fkey" FOREIGN KEY (transportid) REFERENCES transportation(transportid)
    TABLE "train" CONSTRAINT "train_transportid_fkey" FOREIGN KEY (transportid) REFERENCES transportation(transportid)
```

CREATE TABLE Hotel(hotelID VARCHAR(30) PRIMARY KEY, name VARCHAR(30), location VARCHAR(30), checkinDate DATE, checkoutDate DATE, price FLOAT, roomType VARCHAR(30))

```
cs421=> \d hotel
            Table "cs421g55.hotel"
    Column    |         Type          | Modifiers
--------------+-----------------------+-----------
 hotelid      | character varying(30) | not null
 name         | character varying(30) |
 location     | character varying(30) |
 checkindate  | date                  |
 checkoutdate | date                  |
 price        | double precision      |
 roomtype     | character varying(30) |
Indexes:
    "hotel_pkey" PRIMARY KEY, btree (hotelid)
Referenced by:
    TABLE "includehotel" CONSTRAINT "includes_hotelid_fkey" FOREIGN KEY (hotelid) REFERENCES hotel(hotelid)
```

CREATE TABLE Package(bookingID VARCHAR(30), type VARCHAR(30), price FLOAT, description VARCHAR(100), PRIMARY KEY (bookingID, type), FOREIGN KEY (bookingID) REFERENCES Booking)

```
cs421=> \d package
             Table "cs421g55.package"
   Column    |          Type          | Modifiers
-------------+------------------------+-----------
 bookingid   | character varying(30)  | not null
 type        | character varying(30)  | not null
 price       | double precision       |
 description | character varying(100) |
Indexes:
    "package_pkey" PRIMARY KEY, btree (bookingid, type)
Foreign-key constraints:
    "package_bookingid_fkey" FOREIGN KEY (bookingid) REFERENCES booking(bookingid)
```

CREATE TABLE Train(transportID VARCHAR(30) PRIMARY KEY, trainDeparture VARCHAR(30), trainArrival VARCHAR(30), FOREIGN KEY (transportID) REFERENCES Transportation)

```
cs421=> \d train
              Table "cs421g55.train"
     Column     |         Type          | Modifiers
----------------+-----------------------+-----------
 transportid    | character varying(30) | not null
 traindeparture | character varying(30) |
 trainarrival   | character varying(30) |
Indexes:
    "train_pkey" PRIMARY KEY, btree (transportid)
Foreign-key constraints:
    "train_transportid_fkey" FOREIGN KEY (transportid) REFERENCES transportation(transportid)
```

CREATE TABLE Bus(transportID VARCHAR(30) PRIMARY KEY, routeDeparture VARCHAR(30), routeArrival VARCHAR(30), FOREIGN KEY (transportID) REFERENCES Transportation)

```
cs421=> \d bus
               Table "cs421g55.bus"
     Column     |         Type          | Modifiers
----------------+-----------------------+-----------
 transportid    | character varying(30) | not null
 routedeparture | character varying(30) |
 routearrival   | character varying(30) |
Indexes:
    "bus_pkey" PRIMARY KEY, btree (transportid)
Foreign-key constraints:
    "bus_transportid_fkey" FOREIGN KEY (transportid) REFERENCES transportation(transportid)
```

CREATE TABLE Flight(transportID VARCHAR(30) PRIMARY KEY, flightDeparture VARCHAR(30), flightArrival VARCHAR(30), FOREIGN KEY (transportID) REFERENCES Transportation)

```
cs421=> \d flight
              Table "cs421g55.flight"
     Column      |         Type          | Modifiers
-----------------+-----------------------+-----------
 transportid     | character varying(30) | not null
 flightdeparture | character varying(30) |
 flightarrival   | character varying(30) |
Indexes:
    "flight_pkey" PRIMARY KEY, btree (transportid)
Foreign-key constraints:
    "flight_transportid_fkey" FOREIGN KEY (transportid) REFERENCES transportation(transportid)
```

CREATE TABLE Attraction(name VARCHAR(30), location VARCHAR(30), price FLOAT, PRIMARY KEY(name, location))

```
cs421=> \d attraction
        Table "cs421g55.attraction"
  Column  |         Type          | Modifiers
----------+-----------------------+-----------
 name     | character varying(30) | not null
 location | character varying(30) | not null
 price    | double precision      |
Indexes:
    "attraction_pkey" PRIMARY KEY, btree (name, location)
Referenced by:
    TABLE "purchases" CONSTRAINT "purchases_name_fkey" FOREIGN KEY (name, location) REFERENCES attra
ction(name, location)
```

Relations

- For the ‘associates’ relationship between Customer and Booking:

CREATE TABLE Associates (custID VARCHAR(30), bookingID VARCHAR(30), PRIMARY KEY(customerID, bookingID), FOREIGN KEY (customerID) REFERENCES Customer, FOREIGN KEY (bookingID) REFERENCES Booking)

```
cs421=> \d Associates
         Table "cs421g55.associates"
  Column   |         Type          | Modifiers
-----------+-----------------------+-----------
 custid    | character varying(30) | not null
 bookingid | character varying(30) | not null
Indexes:
    "associates_pkey" PRIMARY KEY, btree (custid, bookingid)
Foreign-key constraints:
    "associates_bookingid_fkey" FOREIGN KEY (bookingid) REFERENCES booking(bookingid)
    "associates_custid_fkey" FOREIGN KEY (custid) REFERENCES customer(custid)
```

- For the 'buys' relationship between Customer and Insurance:

CREATE TABLE Buys (customerID VARCHAR(30), insuranceID VARCHAR(30), PRIMARY KEY(customerID, insuranceID), FOREIGN KEY (customerID) REFERENCES Customer, FOREIGN KEY (insuranceID) REFERENCES Insurance)

```
cs421=> \d buys
             Table "cs421g55.buys"
   Column    |         Type          | Modifiers
-------------+-----------------------+-----------
 customerid  | character varying(30) | not null
 insuranceid | character varying(30) | not null
Indexes:
    "buys_pkey" PRIMARY KEY, btree (customerid, insuranceid)
Foreign-key constraints:
    "buys_customerid_fkey" FOREIGN KEY (customerid) REFERENCES customer(custid)
    "buys_insuranceid_fkey" FOREIGN KEY (insuranceid) REFERENCES insurance(insuranceno)
```

- For the 'consults' relationship between Customer and Employee:

CREATE TABLE Consults (customerID VARCHAR(30), eID VARCHAR(30), sessionID VARCHAR(30), sessionDate DATE, sessionTime TIME, PRIMARY KEY(customerID, eID), FOREIGN KEY (customerID) REFERENCES Customer, FOREIGN KEY (eID) REFERENCES Employee)

```
cs421=> \d consults
            Table "cs421g55.consults"
   Column    |          Type          | Modifiers
-------------+------------------------+-----------
 customerid  | character varying(30)  | not null
 eid         | character varying(30)  | not null
 sessionid   | character varying(30)  |
 sessiondate | date                   |
 sessiontime | time without time zone |
Indexes:
    "consults_pkey" PRIMARY KEY, btree (customerid, eid)
Foreign-key constraints:
    "consults_customerid_fkey" FOREIGN KEY (customerid) REFERENCES customer(custid)
    "consults_eid_fkey" FOREIGN KEY (eid) REFERENCES employee(eid)
```

- For the 'rents' relationship between Customer and Car-Rental:

CREATE TABLE Rents (customerID VARCHAR(30), licensePlate VARCHAR(10), location VARCHAR(30), rentID VARCHAR(30), startTime DATE, endTime DATE, PRIMARY KEY(licensePlate, location, customerID),

FOREIGN KEY (customerID) REFERENCES Customer, FOREIGN KEY (licensePlate, location) REFERENCES CarRental(licensePlate, location))

```
cs421=> \d rents
            Table "cs421g55.rents"
    Column    |         Type          | Modifiers
--------------+-----------------------+-----------
 customerid   | character varying(30) | not null
 licenseplate | character varying(10) | not null
 location     | character varying(30) | not null
 rentid       | character varying(30) |
 starttime    | date                  |
 endtime      | date                  |
Indexes:
    "rents_pkey" PRIMARY KEY, btree (licenseplate, location, customerid)
Foreign-key constraints:
    "rents_customerid_fkey" FOREIGN KEY (customerid) REFERENCES customer(custid)
    "rents_licenseplate_fkey" FOREIGN KEY (licenseplate, location) REFERENCES carrental(licenseplate, location)
```

- For the ‘includes’ relationship between Booking and Hotel:

CREATE TABLE Includehotel (bookingID VARCHAR(30), hotelID VARCHAR(30), PRIMARY KEY(bookingID, hotelID), FOREIGN KEY (bookingID) REFERENCES Booking, FOREIGN KEY (hotelID) REFERENCES Hotel)

```
cs421=> \d includehotel
         Table "cs421g55.includehotel"
  Column   |         Type          | Modifiers
-----------+-----------------------+-----------
 bookingid | character varying(30) | not null
 hotelid   | character varying(30) | not null
Indexes:
    "includes_pkey" PRIMARY KEY, btree (bookingid, hotelid)
Foreign-key constraints:
    "includes_bookingid_fkey" FOREIGN KEY (bookingid) REFERENCES booking(bookingid)
    "includes_hotelid_fkey" FOREIGN KEY (hotelid) REFERENCES hotel(hotelid)
```

- For the ‘includes’ relationship between Booking and Transportation:

CREATE TABLE Includetransport (bookingID VARCHAR(30), transportID VARCHAR(30), PRIMARY KEY(bookingID, transportID), FOREIGN KEY (bookingID) REFERENCES Booking, FOREIGN KEY (transportID) REFERENCES Transportation)

```
cs421=> \d includetransport
        Table "cs421g55.includetransport"
   Column    |         Type          | Modifiers
-------------+-----------------------+-----------
 bookingid   | character varying(30) | not null
 transportid | character varying(30) | not null
Indexes:
    "includetransport_pkey" PRIMARY KEY, btree (bookingid, transportid)
Foreign-key constraints:
    "includetransport_bookingid_fkey" FOREIGN KEY (bookingid) REFERENCES booking(bookingid)
    "includetransport_transportid_fkey" FOREIGN KEY (transportid) REFERENCES transportation(transportid)
```

- For the ‘purchases’ relationship between Attraction and Booking:

CREATE TABLE Purchases (bookingID VARCHAR(30), name VARCHAR(30), location VARCHAR(30), PRIMARY KEY (bookingID, name, location), FOREIGN KEY (bookingID) REFERENCES Booking, FOREIGN KEY (name, location) REFERENCES Attraction (name, location))

```
cs421=> \d purchase
Did not find any relation named "purchase".
cs421=> \d purchases
          Table "cs421g55.purchases"
  Column   |         Type          | Modifiers
-----------+-----------------------+-----------
 bookingid | character varying(30) | not null
 name      | character varying(30) | not null
 location  | character varying(30) | not null
Indexes:
    "purchases_pkey" PRIMARY KEY, btree (bookingid, name, location)
Foreign-key constraints:
    "purchases_bookingid_fkey" FOREIGN KEY (bookingid) REFERENCES booking(bookingid)
    "purchases_name_fkey" FOREIGN KEY (name, location) REFERENCES attraction(name, location)
```

- Limitation to the SQL Schema: Buys, Purchases and Includes have Participation constraints that can’t be represented/enforced through an SQL TAble

3. (5 Points) Execute five INSERT commands to insert tuples into one of your relations. Turn in your INSERT statements. Furthermore, print and turn in the response of CPL/psql when you type the INSERT commands. Print and turn in the result when you issue a SELECT * FROM relationname command.

Insertion of the first five records into Booking is shown below:

INSERT INTO Booking VALUES ('b370','2020-04-05', 'Ottawa, Canada', 'Calgary, Canada', 2500.0, 'Successful', TRUE, 'none', 3);

INSERT INTO Booking VALUES ('b400','2020-03-29', 'Montreal, Canada', 'Amsterdam, Netherlands', 2700.0, 'Pending', TRUE, 'none', 2);

INSERT INTO Booking VALUES ('b201','2020-03-01', 'Toronto, Canada', 'Beijing, China', 3700.0, 'Incomplete', FALSE, 'none', 2);

INSERT INTO Booking VALUES ('b173','2020-04-29', 'New York, United States', 'Rome, Italy', 3300.0, 'Successful', TRUE, 'none', 3);

INSERT INTO Booking VALUES ('b437','2020-03-15', 'Seoul, South Korea', 'Toronto, Canada', 3550.0, 'Successful', FALSE, 'none', 1);

```
cs421=> SELECT * FROM Booking;
 bookingid | traveldate | deptlocation | arrvlocation | price | status | bookingtype | disclaimer | noofcustomers
-----------+------------+--------------+--------------+-------+--------+-------------+------------+---------------
(0 rows)

cs421=> INSERT INTO Booking VALUES ('b370','2020-04-05', 'Ottawa, Canada', 'Calgary, Canada', 2500.0, 'Successful', TRUE, 'none', 3);
INSERT 0 1
cs421=> INSERT INTO Booking VALUES ('b400','2020-03-29', 'Montreal, Canada', 'Amsterdam, Netherlands', 2700.0, 'Pending', TRUE, 'none', 2);
INSERT 0 1
cs421=> INSERT INTO Booking VALUES ('b201','2020-03-01', 'Toronto, Canada', 'Beijing, China', 3700.0, 'Incomplete', FALSE, 'none', 2);
INSERT 0 1
cs421=> INSERT INTO Booking VALUES ('b173','2020-04-29', 'New York, United States', 'Rome, Italy', 3300.0, 'Successful', TRUE, 'none', 3);
INSERT 0 1
cs421=> INSERT INTO Booking VALUES ('b437','2020-03-15', 'Seoul, South Korea', 'Toronto, Canada', 3550.0, 'Successful', FALSE, 'none', 1);
INSERT 0 1
cs421=> SELECT * FROM Booking;
 bookingid | traveldate |      deptlocation       |      arrvlocation      | price |   status   | bookingtype | disclaimer | noofcustomers
-----------+------------+-------------------------+------------------------+-------+------------+-------------+------------+---------------
 b370      | 2020-04-05 | Ottawa, Canada          | Calgary, Canada        |  2500 | Successful | true        | none       |             3
 b400      | 2020-03-29 | Montreal, Canada        | Amsterdam, Netherlands |  2700 | Pending    | true        | none       |             2
 b201      | 2020-03-01 | Toronto, Canada         | Beijing, China         |  3700 | Incomplete | false       | none       |             2
 b173      | 2020-04-29 | New York, United States | Rome, Italy            |  3300 | Successful | true        | none       |             3
 b437      | 2020-03-15 | Seoul, South Korea      | Toronto, Canada        |  3550 | Successful | false       | none       |             1
(5 rows)
```

4. (10 Points) Insert in all your tables enough meaningful information so that the queries that you create provide meaningful results. The results of the following queries that you develop should have a reasonable number of results so that we can be convinced that your queries are correct (maybe 5-10 tuples). If you have real-world data, feel free to import it (but do not insert more than 500 records in a table). Information of how to import data into DB2 tables is provided on my courses. ***For each table show the output, truncated to the first 5-10 tuples, that are returned when you issue a SELECT * FROM relationname command. Hint:- I showed a trick in class to do exactly that****.*

The first 8 tuples of each table is pasted below: SELECT * FROM tablename LIMIT 5

Entities

1) Customer

```
cs421=> SELECT * FROM Customer LIMIT 5;
 custid |    fullname    |  birthday  |       address        | phonenumber  | passportnum |        email         | driverlicense
--------+----------------+------------+----------------------+--------------+-------------+----------------------+---------------
 c292   | Katsu Don      | 1980-06-05 | 9846 Happy Avenue    | 438-999-1234 | 67805999    | katsu.donnie@mail.ca | t
 c101   | John Snow      | 1980-10-22 | 300 CrossRoad Way    | 555-825-9034 | 40950763    | john.snow@mail.ca    | t
 c102   | Keegan Appleby | 2010-11-02 | 82 DownTown Abbey    | 555-874,1235 | 84401707    | applebybest@mail.ca  | f
 c103   | Ella Ice       | 1990-03-14 | 53 Trains Road       | 555-523-0934 | 33887199    | frozen.ella@mail.ca  | t
 c277   | Snow White     | 1965-01-01 | 300 AppleView Street | 514-995-0934 | 23810554    | snow.white@mail.ca   | t
(5 rows)
```

2) Booking

```
cs421=> SELECT * FROM booking LIMIT 5;
 bookingid | traveldate |      deptlocation       |      arrvlocation      | price |   status   | bookingtype | disclaimer | noofcustomers
-----------+------------+-------------------------+------------------------+-------+------------+-------------+------------+---------------
 b370      | 2020-04-05 | Ottawa, Canada          | Calgary, Canada        |  2500 | Successful | true        | none       |             3
 b400      | 2020-03-29 | Montreal, Canada        | Amsterdam, Netherlands |  2700 | Pending    | true        | none       |             2
 b201      | 2020-03-01 | Toronto, Canada         | Beijing, China         |  3700 | Incomplete | false       | none       |             2
 b173      | 2020-04-29 | New York, United States | Rome, Italy            |  3300 | Successful | true        | none       |             3
 b437      | 2020-03-15 | Seoul, South Korea      | Toronto, Canada        |  3550 | Successful | false       | none       |             1
(5 rows)
```

3) Employee

```
cs421=> SELECT * FROM employee LIMIT 5;
 eid  |      fullname
------+--------------------
 e159 | McLennan Roland
 e132 | Pascal Siakam
 e202 | Will Wellington
 e095 | Prakash Panangaden
 e050 | Juliet Romeo
(5 rows)
```

4) Insurance

```
cs421=> SELECT * FROM insurance LIMIT 5;
 insuranceno |      companyname      | coverage | price
-------------+-----------------------+----------+-------
 i231        | Funlife Insurance Co. |   200000 |    30
 i314        | Funlife Insurance Co. |   200000 |    30
 i845        | England Wellness      |   300000 |    45
 i696        | Dutch Duck Co.        |   300000 |    50
 i602        | England Wellness      |   250000 |    33
(5 rows)
```

5) CarRental

```
cs421=> SELECT * FROM carrental LIMIT 5;
 licenseplate |        location         |     model
--------------+-------------------------+---------------
 Z5G 3H9      | Calgary, Canada         | Nissan Micra
 V0D 0R1      | Amsterdam, Netherlands  | Nissan Maxima
 M2G 9L3      | Beijing, China          | Honda Civic
 A8X 3L2      | Rome, Italy             | Nissan Micra
 X4X 2Z0      | Toronto, Canada         | Toyota
(5 rows)
```

6) Transportation

```
cs421=> SELECT * FROM transportation LIMIT 5;
 transportid | price |  depdate   | returndate | deptime  | returntime
-------------+-------+------------+------------+----------+------------
 train001    |   500 | 2020-04-05 | 2020-04-20 | 01:35:00 | 05:45:00
 flight002   |   900 | 2020-03-29 | 2020-04-13 | 13:45:00 | 06:30:00
 flight234   |  1200 | 2020-03-01 | 2020-03-10 | 14:15:00 | 15:00:00
 flight546   |   800 | 2020-04-29 | 2020-05-10 | 18:30:00 | 17:30:00
 flight987   |  1300 | 2020-03-15 | 2020-04-01 | 21:15:00 | 20:45:00
(5 rows)
```

7) Hotel

```
cs421=> SELECT * FROM hotel LIMIT 5;
 hotelid |       name        |        location         | checkindate | checkoutdate | price |     roomtype
---------+-------------------+-------------------------+-------------+--------------+-------+-----------------
 h123    | West Bestern      | Calgary, Canada         | 2020-04-05  | 2020-04-19   |  2000 | Superior
 h234    | Holland Inns      | Amsterdam, Netherlands  | 2020-03-29  | 2020-04-11   |  1800 | Deluxe Double
 h788    | Stay Here Please  | Beijing, China          | 2020-03-01  | 2020-03-09   |  2500 | Executive Suite
 h532    | La Suite Bella    | Rome, Italy             | 2020-04-29  | 2020-05-10   |  2500 | Deluxe Single
 h677    | West Bestern      | Toronto, Canada         | 2020-03-15  | 2020-03-31   |  2250 | Superior
(5 rows)
```

8) Package

```
cs421=> SELECT * FROM package LIMIT 5;
 bookingid |  type   | price | description
-----------+---------+-------+-------------
 b279      | Couples |  2520 | none
 b449      | Seniors |  2100 | none
 b066      | Seniors |  2130 | none
 b170      | Family  |  4800 | none
 b442      | Family  |  5000 | none
(5 rows)
```

9) Train

```
cs421=> SELECT * FROM train LIMIT 5;
 transportid | traindeparture | trainarrival
-------------+----------------+--------------
 train001    | TR1234         | TR9876
(1 row)
```

10) Bus

```
cs421=> SELECT * FROM bus LIMIT 5;
 transportid | routedeparture | routearrival
-------------+----------------+--------------
 bus002      | BU1345         | KU9876
(1 row)
```

11) Flight

```
cs421=> SELECT * FROM flight LIMIT 5;
 transportid | flightdeparture | flightarrival
-------------+-----------------+---------------
 flight002   | CAC908          | CAC887
 flight234   | GUB123          | GUB110
 flight546   | POL5530         | POL4646
 flight987   | CAC1768         | CAC9908
 flight100   | TS704           | TS657
(5 rows)
```

12) Attraction

```
cs421=> SELECT * FROM attraction LIMIT 5;
       name       |         location         | price
------------------+--------------------------+-------
 Calgary Tower    | Calgary, Canada          |    20
 Van Gogh Museum  | Amsterdam, Netherlands   |    25
 Bird Nest        | Beijing, China           |    10
 Colosseum        | Vancouver, Canada        |    15
 CN Tower         | Hershey, United States   |    60
(5 rows)
```

Relations

1) Buys

```
cs421=> SELECT * FROM buys LIMIT 5;
 customerid | insuranceid
------------+-------------
 c101       | i231
 c102       | i314
 c103       | i845
 c277       | i696
 c177       | i602
(5 rows)
```

2) Consults

```
cs421=> SELECT * FROM consults LIMIT 5;
 customerid | eid  | sessionid | sessiondate | sessiontime
------------+------+-----------+-------------+-------------
 c101       | e159 | 1769      | 2020-12-12  | 14:00:00
 c102       | e132 | 1396      | 2020-02-07  | 18:30:00
 c103       | e202 | 1284      | 2020-07-19  | 19:15:00
 c277       | e095 | 1359      | 2020-07-06  | 09:30:00
 c177       | e050 | 1606      | 2020-03-03  | 15:30:00
(5 rows)
```

3) Rents

```
cs421=> SELECT * FROM rents LIMIT 5;
 customerid | licenseplate |        location        | rentid | starttime  |  endtime
------------+--------------+------------------------+--------+------------+------------
 c101       | Z5G 3H9      | Calgary, Canada        | 41583  | 2020-04-05 | 2020-04-20
 c102       | V0D 0R1      | Amsterdam, Netherlands | 40081  | 2020-03-29 | 2020-04-13
 c101       | M2G 9L3      | Beijing, China         | 21283  | 2020-03-01 | 2020-03-10
 c101       | A8X 3L2      | Rome, Italy            | 55592  | 2020-04-29 | 2020-05-10
 c101       | X4X 2Z0      | Toronto, Canada        | 39957  | 2020-03-15 | 2020-04-01
(5 rows)
```

4) Includehotel

```
cs421=> SELECT * FROM includehotel LIMIT 5;
 bookingid | hotelid
-----------+---------
 b279      | h111
 b449      | h647
 b066      | h093
 b170      | h543
 b442      | h948
(5 rows)
```

5) Includetransport

```
cs421=> SELECT * FROM includetransport LIMIT 5;
 bookingid | transportid
-----------+-------------
 b400      | flight002
 b201      | flight234
 b173      | flight546
 b437      | flight987
 b279      | bus002
(5 rows)
```

6) Associates

```
cs421=> SELECT * FROM Associates LIMIT 5;
 custid | bookingid
--------+-----------
 c101   | b370
 c102   | b400
 c103   | b201
 c277   | b173
 c177   | b437
(5 rows)
```

7) Purchases

```
cs421=> SELECT * FROM purchases LIMIT 5;
 bookingid |      name       |        location
-----------+-----------------+------------------------
 b370      | Calgary Tower   | Calgary, Canada
 b400      | Van Gogh Museum | Amsterdam, Netherlands
 b201      | Bird Nest       | Beijing, China
 b173      | Colosseum       | Vancouver, Canada
 b437      | CN Tower        | Hershey, United States
(5 rows)
```

5. (20 Points) Write five queries on your project database, using the select-from-where construct of SQL. The queries should be typical queries of the application domain. To receive full credit, all but perhaps one of your queries must exhibit some interesting feature of SQL: queries over more than one relation, subqueries, aggregations, grouping etc. Turn in a description of all the relations that you use in your queries (e.g., the original create statements or printouts from the SQL "describe table yourname" function), a description of what each of your queries is supposed to do, the SQL statement of each query, along with a script illustrating their execution (for example the screenshot when you execute the query). Your script should be sufficient to convince us that your commands run successfully. Please do not, however, turn in query results that are more than 50 lines long.

	Query	**Description**	**Relations Involved in the Query**
1	SELECT fullName FROM Employee WHERE eID IN (SELECT eID FROM Consults WHERE sessionDate= '2020-07-19') UNION SELECT fullName FROM Customer WHERE custID IN (SELECT customerID FROM Consults WHERE sessionDate= '2020-07-19');	Selecting Employee names and Customer Names who had a session on the 19th of July 2020	-Employee **CREATE TABLE Employee**(eID VARCHAR(30) PRIMARY KEY, fullName VARCHAR(30)) -Consults **CREATE TABLE Consults** (customerID VARCHAR(30), eID VARCHAR(30), sessionID VARCHAR(30), sessionDate DATE, sessionTime TIME, PRIMARY KEY(customerID, eID), FOREIGN KEY (customerID) REFERENCES Customer, FOREIGN KEY (eID) REFERENCES Employee) -Customer **CREATE TABLE Customer**(custID VARCHAR(30) PRIMARY KEY, fullName VARCHAR(30), birthday DATE, address VARCHAR(30), phoneNumber VARCHAR(30), passportNum VARCHAR(30), email VARCHAR(30), driverLicense BOOLEAN)

```
cs421=> SELECT fullName FROM Employee WHERE eID IN (SELECT eID FROM Consults WHERE sessionDate= '
2020-07-19') UNION SELECT fullName FROM Customer WHERE custID IN (SELECT customerID FROM Consults
 WHERE sessionDate= '2020-07-19');
    fullname
-----------------
 Ella Ice
 Will Wellington
(2 rows)
```

	Query	Description	Relations Involved in the Query
2	SELECT c.custid, a.bookingid, c.fullname FROM Customer c INNER JOIN Associates a ON c.custid = a.custid;	Matching customers with the corresponding booking id. Custid, bookingid and the customer's fullname is output.	-Customer **CREATE TABLE Customer**(custID VARCHAR(30) PRIMARY KEY, fullName VARCHAR(30), birthday DATE, address VARCHAR(30), phoneNumber VARCHAR(30), passportNum VARCHAR(30), email VARCHAR(30), driverLicense BOOLEAN) -Associates **CREATE TABLE Associates** (custID VARCHAR(30), bookingID VARCHAR(30), PRIMARY KEY(customerID, bookingID), FOREIGN KEY (customerID) REFERENCES Customer, FOREIGN KEY (bookingID) REFERENCES Booking)

```
cs421=> SELECT c.custid, a.bookingid, c.fullname FROM Customer c INNER JOIN Associates a ON c.custid =
 a.custid;
 custid | bookingid |    fullname
--------+-----------+-----------------
 c101   | b370      | John Snow
 c102   | b400      | Keegan Appleby
 c103   | b201      | Ella Ice
 c277   | b173      | Snow White
 c177   | b437      | Disney Mouse
 c292   | b279      | Katsu Don
 c218   | b449      | Cinderella Bleu
 c124   | b066      | Jojo Joestar
 c067   | b170      | Juan Nathan
 c208   | b442      | Trump McDonald
(10 rows)
```

	Query	Description	Relations Involved in the Query
3	SELECT * FROM Customer c WHERE c.phonenumber LIKE '438%';	Selecting customers with phone number that starts with area code '438' (montreal).	-Customer **CREATE TABLE Customer**(custID VARCHAR(30) PRIMARY KEY, fullName VARCHAR(30), birthday DATE, address

			VARCHAR(30), phoneNumber VARCHAR(30), passportNum VARCHAR(30), email VARCHAR(30), driverLicense BOOLEAN)

```
cs421=> SELECT * FROM Customer c WHERE c.phonenumber LIKE '438%';
 custid | fullname  |  birthday  |      address      | phonenumber  | passportnum |        email         | driverlicense
--------+-----------+------------+-------------------+--------------+-------------+----------------------+---------------
 c292   | Katsu Don | 1980-06-05 | 9846 Happy Avenue | 438-999-1234 | 67805999    | katsu.donnie@mail.ca | t
(1 row)
```

	Query	Description	Relations Involved in the Query
4	SELECT name, price FROM Hotel WHERE price < (SELECT AVG (price) FROM Hotel);	Select the Hotels with price less than the average of all hotels. Hotel name and output are output.	-Hotel **CREATE TABLE Hotel**(hotelID VARCHAR(30) PRIMARY KEY, name VARCHAR(30), location VARCHAR(30), checkinDate DATE, checkoutDate DATE, price FLOAT, roomType VARCHAR(30))

```
cs421=> SELECT name, price FROM Hotel WHERE price < (SELECT AVG (price) FROM Hotel);
     name     | price
--------------+-------
 Holland Inns |  1800
 Hotel Motel  |  1425
 Minuit Inns  |  1200
 Fishy Suites |  1600
 Hotel Motel  |  1700
 Hotel Motel  |  1840
(6 rows)
```

	Query	Description	Relations Involved in the Query
5	SELECT MIN(price) FROM Booking GROUP BY arrvLocation;	Select minimum price for each booking by location	-Booking **CREATE TABLE Booking**(bookingID VARCHAR(30) PRIMARY KEY, travelDate DATE, deptLocation VARCHAR(30), arrvLocation VARCHAR(30), price FLOAT, status VARCHAR(10), bookingType VARCHAR(30), disclaimer VARCHAR(100), noOfCustomers INTEGER)

```
cs421=> SELECT MIN(price) FROM Booking GROUP BY arrvLocation;
 min
------
  800
 3300
 2700
 3550
 1825
 2280
 2500
 2830
 2580
 2210
(10 rows)
```

6. (12 Points) Write four data modification commands for your application. Most of these commands should be "interesting," in the sense that they involve some complex feature, such as inserting the result of a query, updating several tuples at once, or deleting a set of tuples that is more than one but less than all the tuples in a relation. Turn in a description of all the relations that you use in your modifications but are not described so far. Provide a short description of what each of your statements is supposed to do, the SQL statements themselves and a script or screenshot that shows your modification commands running in a convincing fashion.

	Modification	**Description**
1	UPDATE Booking SET price = price *0.8, status = 'Successful';	The booking price is reduced by 20% for each customer because of the travel agency's Christmas giveaway. Also every booking has now been verified and have a 'successful' status.

```
cs421=> SELECT * FROM Booking;
 bookingid | traveldate |      deptlocation       |       arrvlocation       | price |   status   | bookingtype | disclaimer | noofcustomers
-----------+------------+-------------------------+--------------------------+-------+------------+-------------+------------+---------------
 b370      | 2020-04-05 | Ottawa, Canada          | Calgary, Canada          |  2500 | Successful | true        | none       |             3
 b173      | 2020-04-29 | New York, United States | Rome, Italy              |  3300 | Successful | true        | none       |             3
 b437      | 2020-03-15 | Seoul, South Korea      | Toronto, Canada          |  3550 | Successful | false       | none       |             1
 b279      | 2020-04-29 | Mexico City, Mexico     | Austin, United States    |  1825 | Successful | true        | none       |             2
 b170      | 2020-04-17 | Berlin, Germany         | Chicago, United States   |  2580 | Successful | false       | none       |             3
 b442      | 2020-03-29 | Kyoto, Japan            | Las Vegas, United States |  2830 | Successful | true        | none       |             1
 b222      | 2020-03-29 | Kyoto, Japan            | Beijing, China           |   800 | Successful | true        | none       |             1
 b201      | 2020-03-01 | Toronto, Canada         | Beijing, China           |  3700 | Incomplete | false       | none       |             2
 b066      | 2020-04-08 | London, England         | Reykjavik, Iceland       |  2280 | Incomplete | true        | none       |             1
 b400      | 2020-03-29 | Montreal, Canada        | Amsterdam, Netherlands   |  2700 | Pending    | true        | none       |             2
 b449      | 2020-04-25 | Orlando, United States  | Paris, France            |  2210 | Pending    | false       | none       |             2
(11 rows)

cs421=> UPDATE Booking SET price = price *0.8, status = 'Successful';
UPDATE 11
cs421=> SELECT * FROM Booking;
 bookingid | traveldate |      deptlocation       |       arrvlocation       | price |   status   | bookingtype | disclaimer | noofcustomers
-----------+------------+-------------------------+--------------------------+-------+------------+-------------+------------+---------------
 b370      | 2020-04-05 | Ottawa, Canada          | Calgary, Canada          |  2000 | Successful | true        | none       |             3
 b173      | 2020-04-29 | New York, United States | Rome, Italy              |  2640 | Successful | true        | none       |             3
 b437      | 2020-03-15 | Seoul, South Korea      | Toronto, Canada          |  2840 | Successful | false       | none       |             1
 b279      | 2020-04-29 | Mexico City, Mexico     | Austin, United States    |  1460 | Successful | true        | none       |             2
 b170      | 2020-04-17 | Berlin, Germany         | Chicago, United States   |  2064 | Successful | false       | none       |             3
 b442      | 2020-03-29 | Kyoto, Japan            | Las Vegas, United States |  2264 | Successful | true        | none       |             1
 b222      | 2020-03-29 | Kyoto, Japan            | Beijing, China           |   640 | Successful | true        | none       |             1
 b201      | 2020-03-01 | Toronto, Canada         | Beijing, China           |  2960 | Successful | false       | none       |             2
 b066      | 2020-04-08 | London, England         | Reykjavik, Iceland       |  1824 | Successful | true        | none       |             1
 b400      | 2020-03-29 | Montreal, Canada        | Amsterdam, Netherlands   |  2160 | Successful | true        | none       |             2
 b449      | 2020-04-25 | Orlando, United States  | Paris, France            |  1768 | Successful | false       | none       |             2
(11 rows)
```

	Modification	**Description**
2	INSERT INTO Customer VALUES ('c987', 'Jane	Adding a new customer to our

	Doe','1998-08-20', '332 Beach Road', '333-987-0603', '34567847', 'jane.doe@mail.coml', ‘true’);	database

```
cs421=> SELECT * FROM Customer;
 custid |    fullname     |  birthday  |          address          | phonenumber | passportnum |        email         | driverlicense
--------+-----------------+------------+---------------------------+-------------+-------------+----------------------+---------------
 c292   | Katsu Don       | 1980-06-05 | 9846 Happy Avenue         | 438-999-1234 | 67805999    | katsu.donnie@mail.ca | t
 c101   | John Snow       | 1980-10-22 | 300 CrossRoad Way         | 555-825-9034 | 40950763    | john.snow@mail.ca    | t
 c102   | Keegan Appleby  | 2010-11-02 | 82 DownTown Abbey         | 555-874,1235 | 84401707    | applebybest@mail.ca  | f
 c103   | Ella Ice        | 1990-03-14 | 53 Trains Road            | 555-523-0934 | 33887199    | frozen.ella@mail.ca  | t
 c277   | Snow White      | 1965-01-01 | 300 AppleView Street      | 514-995-0934 | 23810554    | snow.white@mail.ca   | t
 c218   | Cinderella Bleu | 1995-07-01 | 1111 Egg Valley Road      | 888-123-3210 | 23132095    | cinder.bleu@mail.ca  | t
 c124   | Jojo Joestar    | 1997-01-02 | 6969 Nice Avenue          | 069-069-6969 | 10145679    | Jojstar@mail.ca      | t
 c067   | Juan Nathan     | 2004-02-04 | 383 Prospect Street       | 345-876-0918 | 70024256    | JuanNathan18@gmail.ca | f
 c208   | Trump McDonald  | 2008-10-26 | 22 Royal Avneue           | 111-345-4646 | 16233819    | McDondon@mail.ca     | t
 c177   | Disney Mouse    | 2000-10-31 | 3989 Saint Viateur Street | 777-835-0334 | 82087451    | disney.mouse@mail.ca | t
(10 rows)

cs421=> INSERT INTO Customer VALUES ('c987', 'Jane Doe','1998-08-20', '332 Beach Road', '333-987-0603', '34567847', 'jane.doe@mail.coml', 'true');
INSERT 0 1
cs421=> SELECT * FROM Customer;
 custid |    fullname     |  birthday  |          address          | phonenumber | passportnum |        email         | driverlicense
--------+-----------------+------------+---------------------------+-------------+-------------+----------------------+---------------
 c292   | Katsu Don       | 1980-06-05 | 9846 Happy Avenue         | 438-999-1234 | 67805999    | katsu.donnie@mail.ca | t
 c101   | John Snow       | 1980-10-22 | 300 CrossRoad Way         | 555-825-9034 | 40950763    | john.snow@mail.ca    | t
 c102   | Keegan Appleby  | 2010-11-02 | 82 DownTown Abbey         | 555-874,1235 | 84401707    | applebybest@mail.ca  | f
 c103   | Ella Ice        | 1990-03-14 | 53 Trains Road            | 555-523-0934 | 33887199    | frozen.ella@mail.ca  | t
 c277   | Snow White      | 1965-01-01 | 300 AppleView Street      | 514-995-0934 | 23810554    | snow.white@mail.ca   | t
 c218   | Cinderella Bleu | 1995-07-01 | 1111 Egg Valley Road      | 888-123-3210 | 23132095    | cinder.bleu@mail.ca  | t
 c124   | Jojo Joestar    | 1997-01-02 | 6969 Nice Avenue          | 069-069-6969 | 10145679    | Jojstar@mail.ca      | t
 c067   | Juan Nathan     | 2004-02-04 | 383 Prospect Street       | 345-876-0918 | 70024256    | JuanNathan18@gmail.ca | f
 c208   | Trump McDonald  | 2008-10-26 | 22 Royal Avneue           | 111-345-4646 | 16233819    | McDondon@mail.ca     | t
 c177   | Disney Mouse    | 2000-10-31 | 3989 Saint Viateur Street | 777-835-0334 | 82087451    | disney.mouse@mail.ca | t
 c987   | Jane Doe        | 1998-08-20 | 332 Beach Road            | 333-987-0603 | 34567847    | jane.doe@mail.coml   | t
(11 rows)
```

	Modification	**Description**
3	DELETE FROM Booking WHERE arrvlocation = 'Paris, France' AND status = 'Successful';	Deleting successful booking(s) to Paris because the government has limited number of entries [hypothetical]

Note: Tuples from other relations that are dependent on Booking (use Booking as a foreign key) were deleted alongside this particular tuple (‘Paris, France’).

```
cs421=> SELECT * FROM Booking;
 bookingid | traveldate |      deptlocation       |       arrvlocation        | price |   status   | bookingtype | disclaimer | noofcustomers
-----------+------------+-------------------------+---------------------------+-------+------------+-------------+------------+---------------
 b370      | 2020-04-05 | Ottawa, Canada          | Calgary, Canada           |  2000 | Successful | true        | none       |             3
 b173      | 2020-04-29 | New York, United States | Rome, Italy               |  2640 | Successful | true        | none       |             3
 b437      | 2020-03-15 | Seoul, South Korea      | Toronto, Canada           |  2840 | Successful | false       | none       |             1
 b279      | 2020-04-29 | Mexico City, Mexico     | Austin, United States     |  1460 | Successful | true        | none       |             2
 b170      | 2020-04-17 | Berlin, Germany         | Chicago, United States    |  2064 | Successful | false       | none       |             3
 b442      | 2020-03-29 | Kyoto, Japan            | Las Vegas, United States  |  2264 | Successful | true        | none       |             1
 b222      | 2020-03-29 | Kyoto, Japan            | Beijing, China            |   640 | Successful | true        | none       |             1
 b201      | 2020-03-01 | Toronto, Canada         | Beijing, China            |  2960 | Successful | false       | none       |             2
 b066      | 2020-04-08 | London, England         | Reykjavik, Iceland        |  1824 | Successful | true        | none       |             1
 b400      | 2020-03-29 | Montreal, Canada        | Amsterdam, Netherlands    |  2160 | Successful | true        | none       |             2
 b449      | 2020-04-25 | Orlando, United States  | Paris, France             |  1768 | Successful | false       | none       |             2
(11 rows)

cs421=> DELETE FROM Booking WHERE arrvlocation = 'Paris, France' AND status = 'Successful';
DELETE 1
cs421=> SELECT * FROM Booking;
 bookingid | traveldate |      deptlocation       |       arrvlocation        | price |   status   | bookingtype | disclaimer | noofcustomers
-----------+------------+-------------------------+---------------------------+-------+------------+-------------+------------+---------------
 b370      | 2020-04-05 | Ottawa, Canada          | Calgary, Canada           |  2000 | Successful | true        | none       |             3
 b173      | 2020-04-29 | New York, United States | Rome, Italy               |  2640 | Successful | true        | none       |             3
 b437      | 2020-03-15 | Seoul, South Korea      | Toronto, Canada           |  2840 | Successful | false       | none       |             1
 b279      | 2020-04-29 | Mexico City, Mexico     | Austin, United States     |  1460 | Successful | true        | none       |             2
 b170      | 2020-04-17 | Berlin, Germany         | Chicago, United States    |  2064 | Successful | false       | none       |             3
 b442      | 2020-03-29 | Kyoto, Japan            | Las Vegas, United States  |  2264 | Successful | true        | none       |             1
 b222      | 2020-03-29 | Kyoto, Japan            | Beijing, China            |   640 | Successful | true        | none       |             1
 b201      | 2020-03-01 | Toronto, Canada         | Beijing, China            |  2960 | Successful | false       | none       |             2
 b066      | 2020-04-08 | London, England         | Reykjavik, Iceland        |  1824 | Successful | true        | none       |             1
 b400      | 2020-03-29 | Montreal, Canada        | Amsterdam, Netherlands    |  2160 | Successful | true        | none       |             2
(10 rows)
```

	Modification	Description
4	DELETE FROM Hotel WHERE price > (SELECT MIN(price) FROM Hotel);	*Delete all hotels except the cheapest one*

```
cs421=> SELECT * FROM Hotel;
 hotelid |       name       |          location          | checkindate | checkoutdate | price |     roomtype
---------+------------------+----------------------------+-------------+--------------+-------+------------------
 h123    | West Bestern     | Calgary, Canada            | 2020-04-05  | 2020-04-19   |  2000 | Superior
 h234    | Holland Inns     | Amsterdam, Netherlands     | 2020-03-29  | 2020-04-11   |  1800 | Deluxe Double
 h788    | Stay Here Please | Beijing, China             | 2020-03-01  | 2020-03-09   |  2500 | Executive Suite
 h532    | La Suite Bella   | Rome, Italy                | 2020-04-29  | 2020-05-10   |  2500 | Deluxe Single
 h677    | West Bestern     | Toronto, Canada            | 2020-03-15  | 2020-03-31   |  2250 | Superior
 h111    | Hotel Motel      | Austin, United States      | 2020-04-29  | 2020-05-13   |  1425 | Regular
 h647    | Minuit Inns      | Paris, Frnace              | 2020-04-25  | 2020-05-12   |  1200 | Comfort Plus
 h093    | Fishy Suites     | Reykjavik, Iceland         | 2020-04-08  | 2020-04-19   |  1600 | Comfort
 h543    | Hotel Motel      | Chicago, United States     | 2020-04-17  | 2020-04-27   |  1700 | Garden view
 h948    | Hotel Motel      | Las Vegas, United States   | 2020-03-29  | 2020-04-12   |  1840 | Ocean view
(10 rows)

cs421=> DELETE FROM Hotel WHERE price > (SELECT MIN(price) FROM Hotel);
DELETE 9
cs421=> SELECT * FROM Hotel;
 hotelid |    name     |   location    | checkindate | checkoutdate | price |   roomtype
---------+-------------+---------------+-------------+--------------+-------+--------------
 h647    | Minuit Inns | Paris, Frnace | 2020-04-25  | 2020-05-12   |  1200 | Comfort Plus
(1 row)
```

7. (10 Points) Create two views on top of your database schema. Turn in an informal description what data each of the views represents, show your CREATE VIEW statements and the response of the system. Also, show a query involving each view and the system response (but truncate the response if there are more than a few tuples produced). Finally, show a script of what happens when you try run an SQL UPDATE statement on each of your views. Are either of your views updatable (that is, the database system will automatically translate the update into an update on the base table(s). Explain why or why not. Summarize the conditions that must hold so that DB2/PostgreSQL allows updating a view.

1) **Creating a view of each customer's customer id and associated booking number as well as their birth date**

CREATE VIEW CustomerBooking (custID, fullName, bookingid)
AS SELECT t.custid, t.fullname, t.bookingid, t.birthday
FROM (SELECT c.custid, a.bookingid, c.fullname, c.birthday FROM Customer c INNER JOIN Associates a ON c.custid = a.custid) t

The view is created successfully,

```
cs421=> CREATE VIEW CustomerBooking (custID, fullName, bookingid)
cs421-> AS SELECT t.custid, t.fullname, t.bookingid, t.birthday
cs421-> FROM (SELECT c.custid, a.bookingid, c.fullname, c.birthday FROM Customer c INNER JOIN Associates a ON c.custid = a.custid ) t
cs421-> ;
CREATE VIEW
```

SELECT * query is run to see what's in view customerbooking,

```
cs421=> SELECT * FROM CustomerBooking
;
 custid |    fullname     | bookingid |  birthday
--------+-----------------+-----------+------------
 c101   | John Snow       | b370      | 1980-10-22
 c102   | Keegan Appleby  | b400      | 2010-11-02
 c103   | Ella Ice        | b201      | 1990-03-14
 c277   | Snow White      | b173      | 1965-01-01
 c177   | Disney Mouse    | b437      | 2000-10-31
 c292   | Katsu Don       | b279      | 1980-06-05
 c218   | Cinderella Bleu | b449      | 1995-07-01
 c124   | Jojo Joestar    | b066      | 1997-01-02
 c067   | Juan Nathan     | b170      | 2004-02-04
 c208   | Trump McDonald  | b442      | 2008-10-26
(10 rows)
```

From the screenshot below, we can see **view CustomerBooking** is not updatable.

```
cs421=> UPDATE CustomerBooking SET fullname = 'JJ Snow' WHERE fullname = 'John Snow';
ERROR:  cannot update view "customerbooking"
DETAIL:  Views that do not select from a single table or view are not automatically updatable.
HINT:  To enable updating the view, provide an INSTEAD OF UPDATE trigger or an unconditional ON UPDATE DO INSTEAD rule.
```

2) Creating a view of customers who could possibly drive: customer id and fullname are included

CREATE VIEW PossibleDrivers(custID, fullName)
AS SELECT custID, fullName
FROM Customer
WHERE customer.driverLicense='true';

SELECT * query is run to see what's in view PossibleDrivers,

```
cs421=> SELECT * FROM possibledrivers;
 custid |    fullname
--------+-----------------
 c101   | John Snow
 c124   | Jojo Joestar
 c103   | Ella Ice
 c292   | Katsu Don
 c218   | Cinderella Bleu
 c208   | Trump McDonald
 c277   | Snow White
 c177   | Disney Mouse
(8 rows)
```

We successfully update the view this time,

```
cs421=> UPDATE possibleDrivers SET fullname='Disney Mouse' WHERE fullname='Minne
y Mouse';
UPDATE 1
cs421=> \d
cs421=> UPDATE possibleDrivers SET fullname='Minney Mouse' WHERE fullname='Disne
y Mouse';
UPDATE 1
cs421=> SELECT * FROM possibleDrivers;
 custid |    fullname
--------+----------------
 c292   | Katsu Don
 c101   | John Snow
 c103   | Ella Ice
 c277   | Snow White
 c218   | Cinderella Bleu
 c124   | Jojo Joestar
 c208   | Trump McDonald
 c987   | Jane Doe
 c177   | Minney Mouse
(9 rows)
```

Below is the screenshot of new records in Customer, Disney Mouse now has the name of Minney Mouse

```
cs421=> SELECT * FROM Customer;
 custid |    fullname     |  birthday
--------+-----------------+------------
 c292   | Katsu Don       | 1980-06-05
 c101   | John Snow       | 1980-10-22
 c102   | Keegan Appleby  | 2010-11-02
 c103   | Ella Ice        | 1990-03-14
 c277   | Snow White      | 1965-01-01
 c218   | Cinderella Bleu | 1995-07-01
 c124   | Jojo Joestar    | 1997-01-02
 c067   | Juan Nathan     | 2004-02-04
 c208   | Trump McDonald  | 2008-10-26
 c987   | Jane Doe        | 1998-08-20
 c177   | Minney Mouse    | 2000-10-31
(11 rows)
```

Overall, we discovered that views are updatable when the view is created from only one table. The first view was not updatable due to the fact that it was created from two tables-Customer and Booking. Updating such a view could be problematic, especially if the tables have foreign keys. In that case, the attributes in those other tables have to be updated as well, which could be undesirable. It is easy to use a view created with only one table to update it, since there are no other tables to worry about.

8. (8 Points) Add two CHECK constraints to relations of your database schema. Turn in the revised schema, its successful declaration, and the response of database to modifications (insert/update) that violate the constraints.

1) First CHECK

CREATE TABLE Customer
custID VARCHAR(30) PRIMARY KEY NOT NULL,
fullName VARCHAR(30),
birthday DATE,
address VARCHAR(30),
phoneNumber VARCHAR(30),
passportNum VARCHAR(30),
email VARCHAR(30),
driverLicense BOOLEAN,
CHECK (birthday < 2020-02-28)

We added a CHECK to constrain the birth date of each customer to be earlier than 2020-02-28 (submission date)

```
cs421=> ALTER TABLE Customer ADD CONSTRAINT birthCheck CHECK (birthday < '2020-0
2-28');
ALTER TABLE
```

We can see the CHECK constraint has been successfully added because a new record (i.e. customer) that is born after 2020-02-28 cannot be added.

```
cs421=> INSERT INTO Customer VALUES ('c888', 'Mike Wazowski', '2021-09-12', '
444 Monster Way', '777-777-7777', '54354666', 'mike.w@mail.ca',false);
ERROR:  new row for relation "customer" violates check constraint "birthcheck
"
DETAIL:  Failing row contains (c888, Mike Wazowski, 2021-09-12, 444 Monster W
ay, 777-777-7777, 54354666, mike.w@mail.ca, f).
```

2) Second CHECK

CREATE TABLE Hotel
hotelID VARCHAR(30) PRIMARY KEY NOT NULL,
Name VARCHAR(30),
Location VARCHAR(30),
checkinDate DATE,
checkoutDate DATE,
Price FLOAT,
roomType VARCHAR(30),
CHECK(price < 3000);

We added another CHECK to constrain the overall cost of hotel to be lower than 3000

```
cs421=> ALTER TABLE Hotel ADD CONSTRAINT priceCheck CHECK (price < 3000);
ALTER TABLE
```

Here we can see we are unable to add a hotel with overall cost greater than 3000

```
cs421=> UPDATE Hotel SET price=3100 WHERE name = 'Minuit Inns';
ERROR:  new row for relation "hotel" violates check constraint "pricecheck"
DETAIL:  Failing row contains (h647, Minuit Inns, Paris, Frnace, 2020-04-25,
2020-05-12, 3100, Comfort Plus).
```

9. (10) Points. Creativity

In addition to the creativity of our relations & tuples, we tried to develop a business question,

Complex Analytical Queries:
Query: Identify all customers between the ages of 0-16 who have been listed as ‘true’ in the database for possessing a drivers’ license.

Application: This is important as it would indicate that there is an error, either in the database system or due to inattentive answering by the user.

SELECT * FROM Customer WHERE birthday < ‘2020-02-28’ AND birthday > ‘2004-01-01’ AND driverLicense = ‘true’;

```
custid |    fullname    |  birthday  |     address     | phonenumber  | passportnum |      email       | driverlicense
-------+----------------+------------+-----------------+--------------+-------------+------------------+---------------
c208   | Trump McDonald | 2008-10-26 | 22 Royal Avneue | 111-345-4646 | 16233819    | McDondon@mail.ca | t
(1 row)
```