SUPER **DUPER** MICRO BREWERY

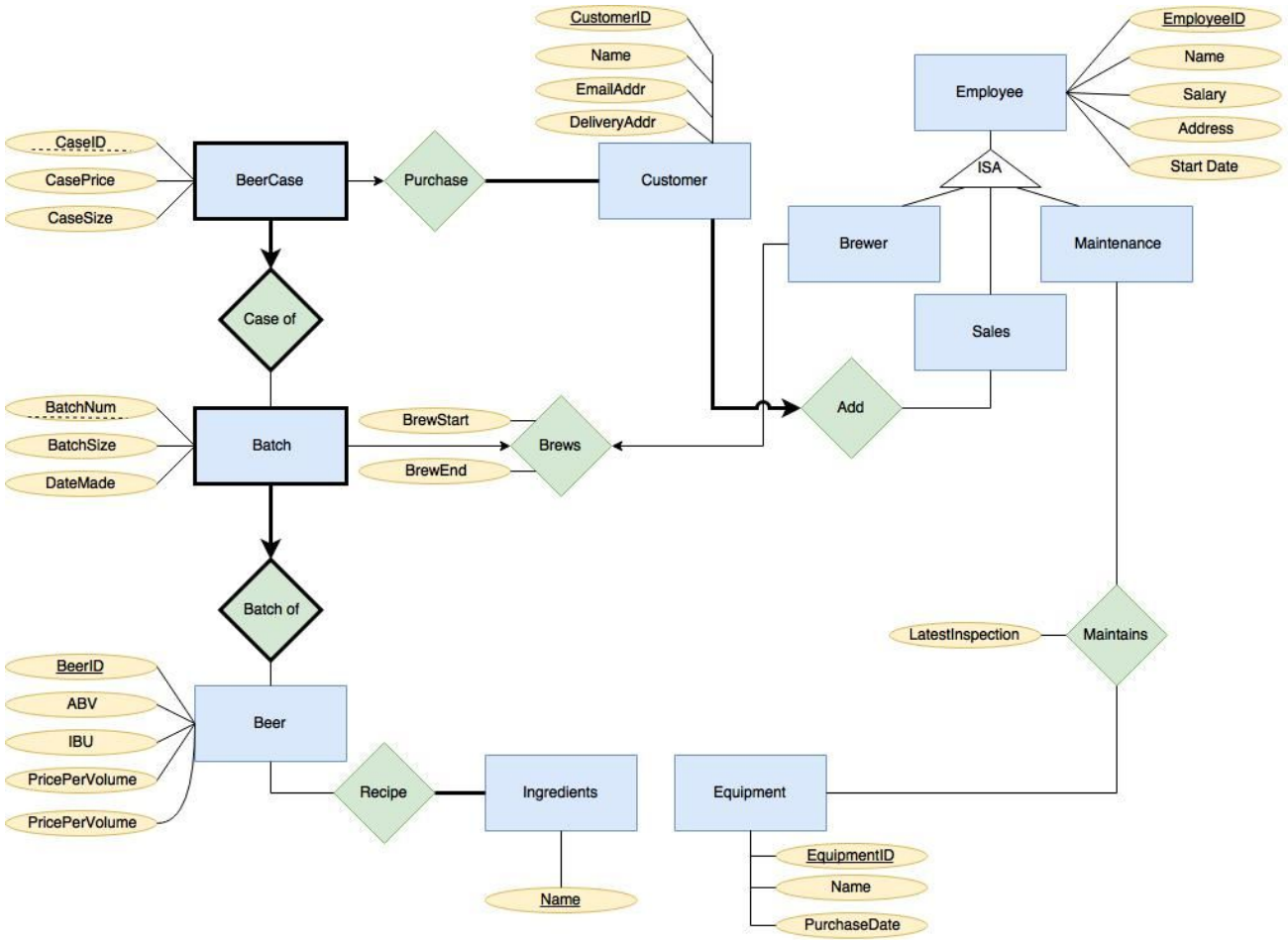
Group 22
Tynan Davis (260557673)
Simon Hsu (260610820)
Alex Wong (260602944)
Andrew Tran (260582911)

Account information:

We changed our password to !brewbeer22

Question 1.

Revised ER:



Reviced Relation:

Entities:

Beer(<u>BeerID,</u> beerName, ABV, IBU, pricePerVolume)

Ingredients(IngredientName)
Equipment(EquipmentID, EquipmentName, PurchaseDate)
Employee(EmployeeID, EmployeeName, Salary, Address, StartDate)
Brewer(EmployeeID) (EmployeeID ref Employee)
Sales(EmployeeID) (EmployeeID ref Employee)
Maintenance(EmployeeID) (EmployeeID ref Employee)
Customer(CustomerID, CustomerName, EmailAddress, DeliveryAddress, EmployeeID) (EmployeeID ref Sales)

Weak entities:
Batch(BatchNum,BatchSize,DateMade, BeerID) (BeerID ref Beer)
BeerCase(CaseID, CasePrice,CaseSize, BatchNum) (BatchNum ref Batch)

Relationships:
Maintains(LaterstInspection, EquipmentID, EmployeeID) (EquipmentID ref Equipment) (EmployeeID ref Employee)
Brews(BrewStart, BrewEnd, EmployeeID, BatchNum, BeerID) (EmployeeID ref Employee) (BatchNum ref Batch) (BeerID ref Beer)
Purchase(CustomerID, CaseID, BeerID) (CustomerID ref Customer) (CaseID ref Case) (BeerID ref eBeer)
Recipe(BeerID, IngredientName) (BeerID ref Beer) (IngredientName ref Ingredients)

Question 2.

Our SQL database schema.
To run our code:
$psql cs421
cs421g22@comp421:~$ \i createtables.sql
cs421g22@comp421:~$ \i insert.sql

We first DROP TABLES

```
DROP TABLE beer CASCADE;
DROP TABLE ingredients CASCADE;
DROP TABLE equipment CASCADE;
DROP TABLE employee CASCADE;
DROP TABLE brewer CASCADE;
DROP TABLE sales CASCADE;
DROP TABLE maintenance CASCADE;
DROP TABLE customer CASCADE;
DROP TABLE batch CASCADE;
DROP TABLE beercase CASCADE;
```

```
DROP TABLE maintains CASCADE;
DROP TABLE brews CASCADE;
DROP TABLE purchase CASCADE;
DROP TABLE recipe CASCADE;
```

Then we create ENTITIES

```
CREATE TABLE ingredients
(
	ingredientname VARCHAR(40) NOT NULL UNIQUE
	, PRIMARY KEY(ingredientname)
);

CREATE TABLE beer
(
	beerid INTEGER NOT NULL UNIQUE
	, beername VARCHAR(60)
	, abv FLOAT NOT NULL
	, ibu INTEGER NOT NULL
	, pricepervolume FLOAT NOT NULL
	, ingredientname VARCHAR(40) UNIQUE
	, PRIMARY KEY(beerid)
);

CREATE TABLE equipment
(
	equipmentid INTEGER NOT NULL UNIQUE
	,equipmentname VARCHAR(60) NOT NULL
	,purchasedate DATE NOT NULL
	,PRIMARY KEY (equipmentID)
);

CREATE TABLE employee
(
	employeeid INTEGER NOT NULL UNIQUE
	, employeename VARCHAR(60) NOT NULL
	, salary FLOAT NOT NULL
	, address VARCHAR(200) NOT NULL
	, startdate DATE NOT NULL
	, PRIMARY KEY(employeeid)
);

CREATE TABLE brewer
```

```
(
        employeeid INTEGER NOT NULL UNIQUE
        ,PRIMARY KEY(employeeid)
        ,FOREIGN KEY(employeeid) REFERENCES employee(employeeid)
);

CREATE TABLE sales
(
        employeeid INTEGER NOT NULL UNIQUE
        ,PRIMARY KEY(employeeid)
        ,FOREIGN KEY(employeeid) REFERENCES employee(employeeid)

);

CREATE TABLE maintenance
(
        employeeid INTEGER NOT NULL UNIQUE
        ,PRIMARY KEY(employeeid)
        ,FOREIGN KEY(employeeid) REFERENCES employee(employeeid)

);

CREATE TABLE customer
(
        customerid INTEGER NOT NULL UNIQUE
        ,customername VARCHAR(60) NOT NULL
        ,emailaddress VARCHAR(100) NOT NULL
        ,deliveraddress VARCHAR(100) NOT NULL
   ,employeeid INTEGER NOT NULL
        ,PRIMARY KEY(customerid)
   ,FOREIGN KEY(employeeid) references sales(employeeid)
);
```

Then we create WEAK ENTITIES

```
CREATE TABLE batch
(
batchnum INTEGER NOT NULL UNIQUE
, batchsize INTEGER NOT NULL
, datemade DATE NOT NULL
, beerid INTEGER NOT NULL
, PRIMARY KEY(batchnum, beerid)
, FOREIGN KEY(beerid) REFERENCES beer(beerid)
);
```

```
CREATE TABLE beercase
(
caseid INTEGER NOT NULL UNIQUE
, caseprice FLOAT NOT NULL
, casesize INTEGER NOT NULL
, batchnum INTEGER NOT NULL
, PRIMARY KEY(caseid, batchnum)
, FOREIGN KEY(batchnum) REFERENCES batch(batchnum)
);
```

Then we create RELATIONSHIPS

```
CREATE TABLE maintains
(
latestinspection DATE NOT NULL
,equipmentid INTEGER NOT NULL
,employeeid INTEGER NOT NULL
,FOREIGN KEY(equipmentid) REFERENCES equipment(equipmentid)
,FOREIGN KEY(employeeid) REFERENCES employee(employeeid)
);

CREATE TABLE brews
(
        brewstart DATE NOT NULL
        , brewend DATE NOT NULL
        , employeeid INTEGER NOT NULL
        , batchnum INTEGER NOT NULL
        , beerid INTEGER NOT NULL
        , FOREIGN KEY(employeeid) REFERENCES employee(employeeid)
        , FOREIGN KEY(batchnum) REFERENCES batch(batchnum)
        , FOREIGN KEY(beerid) REFERENCES beer(beerid)
);

CREATE TABLE purchase
(
        customerid INTEGER NOT NULL
        , caseid INTEGER NOT NULL UNIQUE
        , beerid INTEGER NOT NULL
        , FOREIGN KEY(customerid) REFERENCES customer(customerid)
        , FOREIGN KEY(caseid) REFERENCES beercase(caseid)
        , FOREIGN KEY(beerid) REFERENCES beer(beerid)
);
```

```
CREATE TABLE recipe
(
        beerid INTEGER NOT NULL
        , ingredientname VARCHAR(20) NOT NULL
        , FOREIGN KEY(beerid) REFERENCES beer(beerid)
        , FOREIGN KEY(ingredientname) REFERENCES ingredients(ingredientname)
);
```

Refer to attachment: q2.log for the results

Question 3.

Execute 5 insert commands to insert tuples into one of your relations

INSERT STATEMENTS

To run our code:
$psql cs421
cs421g22@comp421:~$ \i q3.sql

```
INSERT INTO purchases VALUES(936699850,935717398,706863725);
INSERT INTO purchases VALUES(85769946,49815611,570939377);
INSERT INTO purchases VALUES(747715305,332006000,28569098);
INSERT INTO purchases VALUES(882928617,90020198,28569098);
INSERT INTO purchases VALUES(882928617,481193874,570939377);
```

Refer to attachment: q3.log for the results

Question 4.

We inserted enough meaningful information so that the queries that we create provide meaningful results using our automated data generator (java program which we created)

To run our code:
$psql cs421
cs421g22@comp421:~$ \i q4.sql

```
select * from batch limit 5;
select * from beer limit 5;
select * from beercase limit 5;
select * from brewer limit 5;
select * from brews limit 5;
select * from customer limit 5;
select * from employee limit 5;
select * from equipment limit 5;
select * from ingredients limit 5;
select * from maintains limit 5;
select * from maintenance limit 5;
select * from purchase limit 5;
select * from recipe limit 5;
select * from sales limit 5;
```

Refer to attachment: q4.log for the results

Question 5.

Write 5 queries
To run our code:
$psql cs421
cs421g22@comp421:~$ \i q5.sql

```
-- Find all the beer details of distinct batches of beers brewed by brewers named 'Alex'.

SELECT *
FROM beer
JOIN
        (SELECT beerid, batchnum
```

```
        FROM brews
        WHERE employeeid
        IN
                (SELECT *
                FROM brewer
                WHERE employeeid
                IN
                        (SELECT employeeid
                        FROM employee
                        WHERE employeename
                        LIKE 'Alex%')))
                        AS beerinfo
ON beer.beerid=beerinfo.beerid
;

-- List all beer names and quantity of cases purchased by customers named ‘Morty’

SELECT beername, COUNT(customerid)
AS number_of_cases
FROM beer
JOIN purchase
ON beer.beerid=purchase.beerid
WHERE beer.beerid
IN
        (SELECT beerid
        FROM purchase
        WHERE customerid
        IN
                (SELECT customerid
                FROM customer
                WHERE customername
                LIKE 'Morty%'))
GROUP BY beername
;


-- List all beer names, batchnum, abv, and their brewer with abv above 8.0% that were
brewed by either 'Tynan' or 'Alex'. Order by employeename.

SELECT beername, batchnum, abv, employeename AS brewer
FROM beer
JOIN brews
ON beer.beerid=brews.beerid
```

```
JOIN employee
ON brews.employeeid=employee.employeeid
WHERE beer.abv > 8.0
AND brews.employeeid
IN
        (SELECT *
        FROM brewer
        WHERE employeeid
        IN
                (SELECT employeeid
                FROM employee
                WHERE employeename
                LIKE 'Alex%'
                OR employeename
                LIKE 'Tynan%'))
ORDER BY employeename
;

-- Find all ingredients in beer 'SaltySeaWater' ORDER BY ingredient.

SELECT ingredientname
FROM recipe
WHERE beerid
IN
        (SELECT beerid
        FROM beer
        WHERE beername = 'SaltySeaWater')
ORDER BY ingredientname
;

-- Find the average abv of all beers except for the strongest beer.

SELECT AVG(abv)
AS avg_abv
FROM beer
WHERE beerid
NOT IN
        (SELECT beerid
        FROM beer
        WHERE abv
        IN
                (SELECT MAX(abv)
                FROM beer))
```

;

Refer to attachment: q5.log for the results

Question 6.

Write 4 data modification commands (insert/delete/update)
To run our code:
$psql cs421
cs421g22@comp421:~$ \i q6.sql

-- Add new beers to repertoire.
Before INSERT:

```
cs421=> SELECT * FROM beer
cs421-> ;
  beerid   |            beername             |  abv  | ibu | pricepervolume
-----------+---------------------------------+-------+-----+----------------
 693734615 | TearsOfAVirgin                  |   7.2 |  64 |           3.15
 344260827 | SaltySeaWater                   |  7.29 |  38 |           4.95
  92636195 | GaseousVolcano                  | 10.73 | 109 |            3.4
 733993151 | LiquidIceberg                   | 11.55 |  69 |            3.1
 415254748 | LukeSkyWalkersGreenNippleJuice  | 10.33 |  80 |           3.21
 224571418 | ToiletWater                     |  9.83 |   5 |            5.2
(6 rows)
```

INSERT INTO beer (beerid, beername, abv, ibu, pricepervolume) VALUES (114571418, 'DoubleDouble', 4.9, 50, 2.99);
INSERT INTO beer (beerid, beername, abv, ibu, pricepervolume) VALUES (366260827, 'DarkerBetter', 5.5, 10, 0.50);
After INSERT:

```
cs421-> ;
  beerid   |            beername             |  abv  | ibu | pricepervolume
-----------+---------------------------------+-------+-----+----------------
 693734615 | TearsOfAVirgin                  |   7.2 |  64 |           3.15
 344260827 | SaltySeaWater                   |  7.29 |  38 |           4.95
  92636195 | GaseousVolcano                  | 10.73 | 109 |            3.4
 733993151 | LiquidIceberg                   | 11.55 |  69 |            3.1
 415254748 | LukeSkyWalkersGreenNippleJuice  | 10.33 |  80 |           3.21
 224571418 | ToiletWater                     |  9.83 |   5 |            5.2
 114571418 | DoubleDouble                    |   4.9 |  50 |           2.99
 366260827 | DarkerBetter                    |   5.5 |  10 |            0.5
(8 rows)
```

-- Maintain List of Active Customers. (Insert the result of a query)

```
DROP TABLE activecustomers;
CREATE TABLE activecustomers
(
        customerid INTEGER NOT NULL
```

```
        ,customername VARCHAR(60) NOT NULL
);
```

Before insert the result of a query:

```
cs421=> SELECT * FROM activecustomers;
 customerid | customername
------------+--------------
(0 rows)
```

```
INSERT INTO activecustomers (
   SELECT customer.customerid, customer.customername
   FROM customer, purchase
   WHERE customer.customerid = purchase.customerid);
```

After insert the result of query:

```
cs421=> SELECT * FROM activecustomers;
 customerid |    customername
------------+--------------------
   70564146 | Morty
  942136591 | Mr. Poopy Butthole
   26493178 | Bev
  340733760 | Rick
  942136591 | Mr. Poopy Butthole
   26493178 | Bev
   70564146 | Morty
  596793719 | Summer
   26493178 | Bev
  596793719 | Summer
  596793719 | Summer
   70564146 | Morty
  596793719 | Summer
  942136591 | Mr. Poopy Butthole
  340733760 | Rick
  340733760 | Rick
  942136591 | Mr. Poopy Butthole
  958685071 | Jerry
   26493178 | Bev
  340733760 | Rick
(20 rows)
```

-- Fire two active customers. (delete a set of tuples that is more than one)

```
DELETE FROM activecustomers WHERE customerid IN (340733760, 596793719);
```

```
cs421=> DELETE FROM activecustomers WHERE customerid IN (340733760, 596793719);
DELETE 8
cs421=> SELECT * FROM activecustomers;
 customerid |    customername
------------+--------------------
   70564146 | Morty
  942136591 | Mr. Poopy Butthole
   26493178 | Bev
  942136591 | Mr. Poopy Butthole
   26493178 | Bev
   70564146 | Morty
   26493178 | Bev
   70564146 | Morty
  942136591 | Mr. Poopy Butthole
  942136591 | Mr. Poopy Butthole
  958685071 | Jerry
   26493178 | Bev
(12 rows)
```

-- Update customer personal details. (update several tuples at once)

Before UPDATE:

```
 customerid |    customername    |         emailaddress          |   deliveraddress   |
employeeid
------------+--------------------+-------------------------------+--------------------+-
-----------
  942136591 | Mr. Poopy Butthole | 420blazeit@hotmail.com        | 1 Ave WALAO        |
 232716130
   26493178 | Bev                | wakaka@gmail.com              | 6 Hong Gan Plz     |
 232716130
  958685071 | Jerry              | youdontknowme@yahoo.com       | Area 51            |
 232716130
  340733760 | Rick               | bigcowstrongchicken@gmail.com | The Jupiter System |
 232716130
   70564146 | Morty              | dropitlikeitscold@hotmail.com | Hoth Rebel Base    |
 232716130
  596793719 | Summer             | zipzap@yahoo.com              | 221b Baker Street  |
 232716130
(6 rows)
```

UPDATE customer
SET emailaddress = 'beerlover42@gmail.com', deliveraddress = '2 Wallaby Lane'
WHERE customername LIKE 'Rick%';

After UPDATE:

```
 customerid |    customername    |         emailaddress          |  deliveraddress   | e
mployeeid
------------+--------------------+-------------------------------+-------------------+--
----------
  942136591 | Mr. Poopy Butthole | 420blazeit@hotmail.com        | 1 Ave WALAO       |
232716130
   26493178 | Bev                | wakaka@gmail.com              | 6 Hong Gan Plz    |
232716130
  958685071 | Jerry              | youdontknowme@yahoo.com       | Area 51           |
232716130
   70564146 | Morty              | dropitlikeitscold@hotmail.com | Hoth Rebel Base   |
232716130
  596793719 | Summer             | zipzap@yahoo.com              | 221b Baker Street |
232716130
  340733760 | Rick               | beerlover42@gmail.com         | 2 Wallaby Lane    |
232716130
(6 rows)
```

Question 7.

To run our code:
$psql cs421
cs421g22@comp421:~$ \i q7.sql

Create 2 views:

CREATE VIEW [Light Beer List] AS
SELECT beername
FROM beer
WHERE abv < 5.0

CREATE VIEW [All Employees] AS
SELECT employeename
FROM employees

//Updating Views
CREATE OR REPLACE [Light Beer List] AS
SELECT beername, abv
FROM beer
WHERE abv < 5.0

CREATE OR REPLACE [All Employees] AS
SELECT employeename, employeeid
FROM employees

Refer to attachment: q7.log for the results

Question 8.

Our revised SQL database schema.
To run our code:
$psql cs421
cs421g22@comp421:~$ \i createcheckedtables.sql
cs421g22@comp421:~$ \i insert.sql
Add 2 CHECK constraints to relations(We did CHECK on batch and beercase)

```
DROP TABLE beer CASCADE;
DROP TABLE ingredients CASCADE;
DROP TABLE equipment CASCADE;
DROP TABLE employee CASCADE;
DROP TABLE brewer CASCADE;
DROP TABLE sales CASCADE;
DROP TABLE maintenance CASCADE;
DROP TABLE customer CASCADE;
DROP TABLE batch CASCADE;
DROP TABLE beercase CASCADE;
DROP TABLE maintains CASCADE;
DROP TABLE brews CASCADE;
DROP TABLE purchase CASCADE;
DROP TABLE recipe CASCADE;

CREATE TABLE ingredients
(
        ingredientname VARCHAR(40) NOT NULL UNIQUE
        , PRIMARY KEY(ingredientname)
);

CREATE TABLE beer
(
        beerid INTEGER NOT NULL UNIQUE
        , beername VARCHAR(60)
        , abv FLOAT NOT NULL
        , ibu INTEGER NOT NULL
        , pricepervolume FLOAT NOT NULL
        , ingredientname VARCHAR(40) UNIQUE
        , PRIMARY KEY(beerid)
);

CREATE TABLE equipment
```

```
(
        equipmentid INTEGER NOT NULL UNIQUE
        ,equipmentname VARCHAR(60) NOT NULL
        ,purchasedate DATE NOT NULL
        ,PRIMARY KEY (equipmentID)
);

CREATE TABLE employee
(
        employeeid INTEGER NOT NULL UNIQUE
        , employeename VARCHAR(60) NOT NULL
        , salary FLOAT NOT NULL
        , address VARCHAR(200) NOT NULL
        , startdate DATE NOT NULL
        , PRIMARY KEY(employeeid)
);

CREATE TABLE brewer
(
        employeeid INTEGER NOT NULL UNIQUE
        ,PRIMARY KEY(employeeid)
        ,FOREIGN KEY(employeeid) REFERENCES employee(employeeid)
);

CREATE TABLE sales
(
        employeeid INTEGER NOT NULL UNIQUE
        ,PRIMARY KEY(employeeid)
        ,FOREIGN KEY(employeeid) REFERENCES employee(employeeid)

);

CREATE TABLE maintenance
(
        employeeid INTEGER NOT NULL UNIQUE
        ,PRIMARY KEY(employeeid)
        ,FOREIGN KEY(employeeid) REFERENCES employee(employeeid)

);

CREATE TABLE customer
(
        customerid INTEGER NOT NULL UNIQUE
```

```
        ,customername VARCHAR(60) NOT NULL
        ,emailaddress VARCHAR(100) NOT NULL
        ,deliveraddress VARCHAR(100) NOT NULL
    ,employeeid INTEGER NOT NULL
        ,PRIMARY KEY(customerid)
    ,FOREIGN KEY(employeeid) references sales(employeeid)
);

CREATE TABLE batch
(
batchnum INTEGER NOT NULL UNIQUE
, batchsize INTEGER NOT NULL CHECK (batchsize > 0)
, datemade DATE NOT NULL
, beerid INTEGER NOT NULL
, PRIMARY KEY(batchnum, beerid)
, FOREIGN KEY(beerid) REFERENCES beer(beerid)
);

CREATE TABLE beercase
(
caseid INTEGER NOT NULL UNIQUE
, caseprice FLOAT NOT NULL CHECK (caseprice > 0)
, casesize INTEGER NOT NULL CHECK (casesize > 0)
, batchnum INTEGER NOT NULL CHECK (batchnum > 0)
, PRIMARY KEY(caseid, batchnum)
, FOREIGN KEY(batchnum) REFERENCES batch(batchnum)
);


CREATE TABLE maintains
(
latestinspection DATE NOT NULL
,equipmentid INTEGER NOT NULL
,employeeid INTEGER NOT NULL
,FOREIGN KEY(equipmentid) REFERENCES equipment(equipmentid)
,FOREIGN KEY(employeeid) REFERENCES employee(employeeid)
);

CREATE TABLE brews
(
        brewstart DATE NOT NULL
        , brewend DATE NOT NULL
        , employeeid INTEGER NOT NULL
```

```
        , batchnum INTEGER NOT NULL
        , beerid INTEGER NOT NULL
        , FOREIGN KEY(employeeid) REFERENCES employee(employeeid)
        , FOREIGN KEY(batchnum) REFERENCES batch(batchnum)
        , FOREIGN KEY(beerid) REFERENCES beer(beerid)
);

CREATE TABLE purchase
(
        customerid INTEGER NOT NULL
        , caseid INTEGER NOT NULL UNIQUE
        , beerid INTEGER NOT NULL
        , FOREIGN KEY(customerid) REFERENCES customer(customerid)
        , FOREIGN KEY(caseid) REFERENCES beercase(caseid)
        , FOREIGN KEY(beerid) REFERENCES beer(beerid)
);

CREATE TABLE recipe
(
        beerid INTEGER NOT NULL
        , ingredientname VARCHAR(20) NOT NULL
        , FOREIGN KEY(beerid) REFERENCES beer(beerid)
        , FOREIGN KEY(ingredientname) REFERENCES ingredients(ingredientname)
);
```

Refer to attachment: q8.log for the results

Question 9.

Creativity points

- **Super duper automated data generation written in java**