Project Milestone 2

Comp 421 - Winter 2020

Ben Ruddock, Binyuan Sun, Sarah Clusiau, Eric Sun

For parts of this milestone where queries and commands were required, we have attached the relevant files for a more readable format outside of this document. Our implementation dialect was Postgresql.

Question 1

See attached.

Please note: the only changes that we made to our project from the feedback was to avoid making tables for the initiates, supplypayment, reservationpayment, located and reserves entities. This was because they had a key and participation constraint.

Question 2

Listed below are all of the create table commands for our milestone, not modeled are the participation constraints between:

- Supplier and the relation supply payment
- Dish and the relation ingredients
- Menu and the relation contains
- Event and the relation has and the relation staffed by

```
create table customer
(
   acctcreated  date,
   lastorder    date,
   emailaddress varchar(30) not null
      constraint customer_pkey
         primary key,
   phonenumber  bigint,
   name         varchar(20)
);

alter table customer
   owner to cs421g84;
```

```
create table supplier
(
   delivers    boolean,
   companyname varchar(20) not null
      constraint supplier_pkey
         primary key,
   address     varchar(20)
);

alter table supplier
   owner to cs421g84;
```

```
create table dish
(
   name              varchar(20) not null
      constraint dish_pkey
         primary key,
   preptime          time,
   type              varchar(20),
   dietaryrestriction varchar(20),
   frozen            boolean
);

alter table dish
   owner to cs421g84;

create table menu
(
   numcourses integer,
   menuid     integer not null
      constraint menu_pkey
         primary key,
   seasonal   varchar(20)
);

alter table menu
   owner to cs421g84;


create table venue
(
   address   varchar(20) not null
      constraint venue_pkey
         primary key,
   kitchen   boolean,
   tableware boolean,
   rented    boolean
);

alter table venue
   owner to cs421g84;

create table staff
(
   role          varchar(20),
   employedsince date,
   name          varchar(20),
   employeeid    integer not null
      constraint staff_pkey
         primary key
);

alter table staff
   owner to cs421g84;

create table account
(
   acctid integer not null
      constraint account_pkey
         primary key
);

alter table account
   owner to cs421g84;

create table accountreceivable
(
   revenue integer,
   acctid  integer not null
      constraint accountreceivable_pkey
         primary key
      constraint
accountreceivable_acctid_fkey
         references account
);

alter table accountreceivable
   owner to cs421g84;

create table accountpayable
(
   budget integer,
   acctid integer not null
      constraint accountpayable_pkey
         primary key
      constraint accountpayable_acctid_fkey
         references account
);
```

```
alter table accountpayable
   owner to cs421g84;

create table runs
(
   employeeid integer not null
      constraint runs_employeeid_fkey
         references staff,
   acctid     integer not null
      constraint runs_acctid_fkey
         references account,
   constraint runs_pkey
      primary key (employeeid, acctid)
);

alter table runs
   owner to cs421g84;

create table prepares
(
   menuid     integer not null
      constraint prepares_menuid_fkey
         references menu,
   employeeid integer not null
      constraint prepares_employeeid_fkey
         references staff,
   constraint prepares_pkey
      primary key (menuid, employeeid)
);

alter table prepares
   owner to cs421g84;

create table contains
(
   menuid integer     not null
      constraint contains_menuid_fkey
         references menu,
   name   varchar(20) not null
      constraint contains_name_fkey
         references dish,
   constraint contains_pkey
      primary key (menuid, name)
);

alter table contains
   owner to cs421g84;



create table ingredients
(
   companyname    varchar(20) not null
      constraint
ingredients_companyname_fkey
         references supplier,
   name           varchar(20) not null
      constraint ingredients_name_fkey
         references dish,
   ordered        date,
   received       date,
   quantity       integer,
   ingredientname varchar(35) not null,
   constraint ingredients_pk
      primary key (companyname, name,
ingredientname)
);

alter table ingredients
   owner to cs421g84;

create table drink
(
   name   varchar(20) not null
      constraint drink_pkey
         primary key
      constraint drink_name_fkey
         references dish,
   mature boolean
);

alter table drink
   owner to cs421g84;
```

```
create table invoice
(
   date                date,
   amount              numeric,
   descriptionofservices varchar(100),
   invoiceid           integer not null
      constraint invoice_pkey
         primary key,
   status              varchar(20),
   clientemail         varchar(30)
      constraint invoice_clientemail_fkey
         references customer,
   suppliername        varchar(20)
      constraint invoice_suppliername_fkey
         references supplier,
   acctid              integer
      constraint invoice_acctid_fkey
         references account
);

alter table invoice
   owner to cs421g84;

create table reservation
(
   rdate       date,
   rtime       time,
   reservationid integer not null
      constraint reservation_pkey
         primary key,
   clientemail   varchar(30)
      constraint reservation_clientemail_fkey
         references customer
);

alter table reservation
   owner to cs421g84;

create table event
(
   corporate       boolean,
   staffamount     integer,
   etime           time    not null,
   mature          boolean,
   eventdescription varchar(100),
   edate           date    not null,
   attendees       integer,
   address         varchar(20)
      constraint event_address_fkey
         references venue,
   reservationid    integer not null
      constraint event_reservationid_fkey
         references reservation,
   constraint event_pkey
      primary key (edate, etime,
reservationid)
);

alter table event
   owner to cs421g84;

create table has
(
   menuid       integer not null
      constraint has_menuid_fkey
         references menu,
   edate        date    not null,
   etime        time    not null,
   reservationid integer not null,
   constraint has_pkey
      primary key (menuid, etime, edate,
reservationid),
   constraint has_edate_fkey
      foreign key (edate, etime,
reservationid) references event
);

alter table has
   owner to cs421g84;

create table salary
(
   acctid     integer not null
      constraint salary_acctid_fkey
         references accountpayable,
```

```
    employeeid integer not null
        constraint salary_employeeid_fkey
            references staff,
    number     integer,
    constraint salary_pkey
        primary key (employeeid, acctid)
);

alter table salary
    owner to cs421g84;

create table staffby
(
    edate         date    not null,
    etime         time    not null,
    reservationid integer not null,
    emid1         integer
        constraint staffby_emid1_fkey
            references staff,
    emid2         integer
        constraint staffby_emid2_fkey
            references staff,
    emid3         integer
        constraint staffby_emid3_fkey
            references staff,
    emid4         integer
        constraint staffby_emid4_fkey
            references staff,
    emid5         integer
        constraint staffby_emid5_fkey
            references staff,
    emid6         integer
        constraint staffby_emid6_fkey
            references staff,
    emid7         integer
        constraint staffby_emid7_fkey
            references staff,
    emid8         integer
        constraint staffby_emid8_fkey
            references staff,
    emid9         integer
        constraint staffby_emid9_fkey
            references staff,
    emid10        integer
        constraint staffby_emid10_fkey
            references staff,
    constraint staffby_pkey
        primary key (edate, etime,
reservationid),
    constraint staffby_edate_fkey
        foreign key (edate, etime,
reservationid) references event
);

alter table staffby
    owner to cs421g84;
```

The output of \d table for all of our generated data is shown below.

```
   Table "cs421g84.account"
 Column |  Type   | Modifiers
--------+---------+-----------
 acctid | integer | not null
Indexes:
    "account_pkey" PRIMARY KEY, btree (acctid)
Referenced by:
    TABLE "accountpayable" CONSTRAINT "accountpayable_acctid_fkey" FOREIGN KEY (acctid) REFERENCES account(acctid)
    TABLE "accountreceivable" CONSTRAINT "accountreceivable_acctid_fkey" FOREIGN KEY (acctid) REFERENCES account(acctid)
    TABLE "invoice" CONSTRAINT "invoice_acctid_fkey" FOREIGN KEY (acctid) REFERENCES account(acctid)
    TABLE "runs" CONSTRAINT "runs_acctid_fkey" FOREIGN KEY (acctid) REFERENCES account(acctid)

 Index "cs421g84.account_pkey"
 Column |  Type   | Definition
--------+---------+------------
 acctid | integer | acctid
primary key, btree, for table "cs421g84.account"

Table "cs421g84.accountpayable"
 Column |  Type   | Modifiers
--------+---------+-----------
 budget | integer |
 acctid | integer | not null
Indexes:
    "accountpayable_pkey" PRIMARY KEY, btree (acctid)
Foreign-key constraints:
    "accountpayable_acctid_fkey" FOREIGN KEY (acctid) REFERENCES account(acctid)
Referenced by:
    TABLE "salary" CONSTRAINT "salary_acctid_fkey" FOREIGN KEY (acctid) REFERENCES accountpayable(acctid)

Index "cs421g84.accountpayable_pkey"
 Column |  Type   | Definition
--------+---------+------------
 acctid | integer | acctid
primary key, btree, for table "cs421g84.accountpayable"

Table "cs421g84.accountreceivable"
 Column  |  Type   | Modifiers
---------+---------+-----------
 revenue | integer |
 acctid  | integer | not null
Indexes:
    "accountreceivable_pkey" PRIMARY KEY, btree (acctid)
Foreign-key constraints:
    "accountreceivable_acctid_fkey" FOREIGN KEY (acctid) REFERENCES account(acctid)

Index "cs421g84.accountreceivable_pkey"
 Column |  Type   | Definition
--------+---------+------------
 acctid | integer | acctid
primary key, btree, for table "cs421g84.accountreceivable"

         Table "cs421g84.contains"
 Column |         Type          | Modifiers
--------+-----------------------+-----------
 menuid | integer               | not null
 name   | character varying(20) | not null
Indexes:
    "contains_pkey" PRIMARY KEY, btree (menuid, name)
Foreign-key constraints:
    "contains_menuid_fkey" FOREIGN KEY (menuid) REFERENCES menu(menuid)
    "contains_name_fkey" FOREIGN KEY (name) REFERENCES dish(name)

       Index "cs421g84.contains_pkey"
 Column |         Type          | Definition
--------+-----------------------+------------
 menuid | integer               | menuid
 name   | character varying(20) | name
primary key, btree, for table "cs421g84.contains"
```

```
           Table "cs421g84.customer"
    Column    |         Type          | Modifiers
--------------+-----------------------+-----------
 acctcreated  | date                  |
 lastorder    | date                  |
 emailaddress | character varying(30) | not null
 phonenumber  | bigint                |
 name         | character varying(20) |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (emailaddress)
Referenced by:
    TABLE "invoice" CONSTRAINT "invoice_clientemail_fkey" FOREIGN KEY (clientemail) REFERENCES customer(emailaddress)
    TABLE "reservation" CONSTRAINT "reservation_clientemail_fkey" FOREIGN KEY (clientemail) REFERENCES customer(emailaddress)

          Index "cs421g84.customer_pkey"
    Column    |         Type          |  Definition
--------------+-----------------------+--------------
 emailaddress | character varying(30) | emailaddress
primary key, btree, for table "cs421g84.customer"

                 Table "cs421g84.dish"
       Column       |          Type          | Modifiers
--------------------+------------------------+-----------
 name               | character varying(20)  | not null
 preptime           | time without time zone |
 type               | character varying(20)  |
 dietaryrestriction | character varying(20)  |
 frozen             | boolean                |
Indexes:
    "dish_pkey" PRIMARY KEY, btree (name)
Referenced by:
    TABLE "contains" CONSTRAINT "contains_name_fkey" FOREIGN KEY (name) REFERENCES dish(name)
    TABLE "drink" CONSTRAINT "drink_name_fkey" FOREIGN KEY (name) REFERENCES dish(name)
    TABLE "ingredients" CONSTRAINT "ingredients_name_fkey" FOREIGN KEY (name) REFERENCES dish(name)

        Index "cs421g84.dish_pkey"
 Column |         Type          | Definition
--------+-----------------------+------------
 name   | character varying(20) | name
primary key, btree, for table "cs421g84.dish"

          Table "cs421g84.drink"
 Column |         Type          | Modifiers
--------+-----------------------+-----------
 name   | character varying(20) | not null
 mature | boolean               |
Indexes:
    "drink_pkey" PRIMARY KEY, btree (name)
Foreign-key constraints:
    "drink_name_fkey" FOREIGN KEY (name) REFERENCES dish(name)

        Index "cs421g84.drink_pkey"
 Column |         Type          | Definition
--------+-----------------------+------------
 name   | character varying(20) | name
primary key, btree, for table "cs421g84.drink"
```

```
               Table "cs421g84.event"
      Column      |          Type          | Modifiers
------------------+------------------------+-----------
 corporate        | boolean                |
 staffamount      | integer                |
 etime            | time without time zone | not null
 mature           | boolean                |
 eventdescription | character varying(100) |
 edate            | date                   | not null
 attendees        | integer                |
 address          | character varying(20)  |
 reservationid    | integer                | not null
Indexes:
    "event_pkey" PRIMARY KEY, btree (edate, etime, reservationid)
Foreign-key constraints:
    "event_address_fkey" FOREIGN KEY (address) REFERENCES venue(address)
    "event_reservationid_fkey" FOREIGN KEY (reservationid) REFERENCES reservation(reservationid)
Referenced by:
    TABLE "has" CONSTRAINT "has_edate_fkey" FOREIGN KEY (edate, etime, reservationid) REFERENCES event(edate, etime, reservationid)
    TABLE "staffby" CONSTRAINT "staffby_edate_fkey" FOREIGN KEY (edate, etime, reservationid) REFERENCES event(edate, etime, reservationid)

             Index "cs421g84.event_pkey"
    Column     |          Type          |  Definition
---------------+------------------------+---------------
 edate         | date                   | edate
 etime         | time without time zone | etime
 reservationid | integer                | reservationid
primary key, btree, for table "cs421g84.event"

              Table "cs421g84.has"
    Column     |          Type          | Modifiers
---------------+------------------------+-----------
 menuid        | integer                | not null
 edate         | date                   | not null
 etime         | time without time zone | not null
 reservationid | integer                | not null
Indexes:
    "has_pkey" PRIMARY KEY, btree (menuid, etime, edate, reservationid)
Foreign-key constraints:
    "has_edate_fkey" FOREIGN KEY (edate, etime, reservationid) REFERENCES event(edate, etime, reservationid)
    "has_menuid_fkey" FOREIGN KEY (menuid) REFERENCES menu(menuid)

              Index "cs421g84.has_pkey"
    Column     |          Type          |  Definition
---------------+------------------------+---------------
 menuid        | integer                | menuid
 etime         | time without time zone | etime
 edate         | date                   | edate
 reservationid | integer                | reservationid
primary key, btree, for table "cs421g84.has"
```

```
            Table "cs421g84.ingredients"
     Column     |         Type          | Modifiers
----------------+-----------------------+-----------
 companyname    | character varying(20) | not null
 name           | character varying(20) | not null
 ordered        | date                  |
 received       | date                  |
 quantity       | integer               |
 ingredientname | character varying(35) | not null
Indexes:
    "ingredients_pk" PRIMARY KEY, btree (companyname, name, ingredientname)
Foreign-key constraints:
    "ingredients_companyname_fkey" FOREIGN KEY (companyname) REFERENCES supplier(companyname)
    "ingredients_name_fkey" FOREIGN KEY (name) REFERENCES dish(name)

             Index "cs421g84.ingredients_pk"
     Column     |         Type          |   Definition
----------------+-----------------------+----------------
 companyname    | character varying(20) | companyname
 name           | character varying(20) | name
 ingredientname | character varying(35) | ingredientname
primary key, btree, for table "cs421g84.ingredients"

                 Table "cs421g84.invoice"
        Column         |          Type          | Modifiers
-----------------------+------------------------+-----------
 date                  | date                   |
 amount                | numeric                |
 descriptionofservices | character varying(100) |
 invoiceid             | integer                | not null
 status                | character varying(20)  |
 clientemail           | character varying(30)  |
 suppliername          | character varying(20)  |
 acctid                | integer                |
Indexes:
    "invoice_pkey" PRIMARY KEY, btree (invoiceid)
Foreign-key constraints:
    "invoice_acctid_fkey" FOREIGN KEY (acctid) REFERENCES account(acctid)
    "invoice_clientemail_fkey" FOREIGN KEY (clientemail) REFERENCES customer(emailaddress)
    "invoice_suppliername_fkey" FOREIGN KEY (suppliername) REFERENCES supplier(companyname)

  Index "cs421g84.invoice_pkey"
  Column   |  Type   | Definition
-----------+---------+------------
 invoiceid | integer | invoiceid
primary key, btree, for table "cs421g84.invoice"

             Table "cs421g84.menu"
   Column    |         Type          | Modifiers
-------------+-----------------------+-----------
 numcourses  | integer               |
 menuid      | integer               | not null
 seasonal    | character varying(20) |
Indexes:
    "menu_pkey" PRIMARY KEY, btree (menuid)
Referenced by:
    TABLE "contains" CONSTRAINT "contains_menuid_fkey" FOREIGN KEY (menuid) REFERENCES menu(menuid)
    TABLE "has" CONSTRAINT "has_menuid_fkey" FOREIGN KEY (menuid) REFERENCES menu(menuid)
    TABLE "prepares" CONSTRAINT "prepares_menuid_fkey" FOREIGN KEY (menuid) REFERENCES menu(menuid)

  Index "cs421g84.menu_pkey"
 Column |  Type   | Definition
--------+---------+------------
 menuid | integer | menuid
primary key, btree, for table "cs421g84.menu"
```

```
    Table "cs421g84.prepares"
   Column   |  Type   | Modifiers
------------+---------+-----------
 menuid     | integer | not null
 employeeid | integer | not null
Indexes:
    "prepares_pkey" PRIMARY KEY, btree (menuid, employeeid)
Foreign-key constraints:
    "prepares_employeeid_fkey" FOREIGN KEY (employeeid) REFERENCES staff(employeeid)
    "prepares_menuid_fkey" FOREIGN KEY (menuid) REFERENCES menu(menuid)

  Index "cs421g84.prepares_pkey"
   Column   |  Type   | Definition
------------+---------+------------
 menuid     | integer | menuid
 employeeid | integer | employeeid
primary key, btree, for table "cs421g84.prepares"

           Table "cs421g84.reservation"
    Column     |          Type          | Modifiers
---------------+------------------------+-----------
 rdate         | date                   |
 rtime         | time without time zone |
 reservationid | integer                | not null
 clientemail   | character varying(30)  |
Indexes:
    "reservation_pkey" PRIMARY KEY, btree (reservationid)
Foreign-key constraints:
    "reservation_clientemail_fkey" FOREIGN KEY (clientemail) REFERENCES customer(emailaddress)
Referenced by:
    TABLE "event" CONSTRAINT "event_reservationid_fkey" FOREIGN KEY (reservationid) REFERENCES reservation(reservationid)

    Index "cs421g84.reservation_pkey"
    Column     |  Type   |  Definition
---------------+---------+--------------
 reservationid | integer | reservationid
primary key, btree, for table "cs421g84.reservation"

      Table "cs421g84.runs"
   Column   |  Type   | Modifiers
------------+---------+-----------
 employeeid | integer | not null
 acctid     | integer | not null
Indexes:
    "runs_pkey" PRIMARY KEY, btree (employeeid, acctid)
Foreign-key constraints:
    "runs_acctid_fkey" FOREIGN KEY (acctid) REFERENCES account(acctid)
    "runs_employeeid_fkey" FOREIGN KEY (employeeid) REFERENCES staff(employeeid)

    Index "cs421g84.runs_pkey"
   Column   |  Type   | Definition
------------+---------+------------
 employeeid | integer | employeeid
 acctid     | integer | acctid
primary key, btree, for table "cs421g84.runs"
```

```
     Table "cs421g84.salary"
   Column    |  Type   | Modifiers
-------------+---------+-----------
 acctid      | integer | not null
 employeeid  | integer | not null
 number      | integer |
Indexes:
    "salary_pkey" PRIMARY KEY, btree (employeeid, acctid)
Foreign-key constraints:
    "salary_acctid_fkey" FOREIGN KEY (acctid) REFERENCES accountpayable(acctid)
    "salary_employeeid_fkey" FOREIGN KEY (employeeid) REFERENCES staff(employeeid)

   Index "cs421g84.salary_pkey"
   Column    |  Type   | Definition
-------------+---------+------------
 employeeid  | integer | employeeid
 acctid      | integer | acctid
primary key, btree, for table "cs421g84.salary"

              Table "cs421g84.staff"
    Column     |         Type          | Modifiers
---------------+-----------------------+-----------
 role          | character varying(20) |
 employedsince | date                  |
 name          | character varying(20) |
 employeeid    | integer               | not null
Indexes:
    "staff_pkey" PRIMARY KEY, btree (employeeid)
Referenced by:
    TABLE "prepares" CONSTRAINT "prepares_employeeid_fkey" FOREIGN KEY (employeeid) REFERENCES staff(employeeid)
    TABLE "runs" CONSTRAINT "runs_employeeid_fkey" FOREIGN KEY (employeeid) REFERENCES staff(employeeid)
    TABLE "salary" CONSTRAINT "salary_employeeid_fkey" FOREIGN KEY (employeeid) REFERENCES staff(employeeid)
    TABLE "staffby" CONSTRAINT "staffby_emid10_fkey" FOREIGN KEY (emid10) REFERENCES staff(employeeid)
    TABLE "staffby" CONSTRAINT "staffby_emid1_fkey" FOREIGN KEY (emid1) REFERENCES staff(employeeid)
    TABLE "staffby" CONSTRAINT "staffby_emid2_fkey" FOREIGN KEY (emid2) REFERENCES staff(employeeid)
    TABLE "staffby" CONSTRAINT "staffby_emid3_fkey" FOREIGN KEY (emid3) REFERENCES staff(employeeid)
    TABLE "staffby" CONSTRAINT "staffby_emid4_fkey" FOREIGN KEY (emid4) REFERENCES staff(employeeid)
    TABLE "staffby" CONSTRAINT "staffby_emid5_fkey" FOREIGN KEY (emid5) REFERENCES staff(employeeid)
    TABLE "staffby" CONSTRAINT "staffby_emid6_fkey" FOREIGN KEY (emid6) REFERENCES staff(employeeid)
    TABLE "staffby" CONSTRAINT "staffby_emid7_fkey" FOREIGN KEY (emid7) REFERENCES staff(employeeid)
    TABLE "staffby" CONSTRAINT "staffby_emid8_fkey" FOREIGN KEY (emid8) REFERENCES staff(employeeid)
    TABLE "staffby" CONSTRAINT "staffby_emid9_fkey" FOREIGN KEY (emid9) REFERENCES staff(employeeid)

    Index "cs421g84.staff_pkey"
   Column    |  Type   | Definition
-------------+---------+------------
 employeeid  | integer | employeeid
primary key, btree, for table "cs421g84.staff"

            Index "cs421g84.staffby_pkey"
    Column     |          Type          |  Definition
---------------+------------------------+---------------
 edate         | date                   | edate
 etime         | time without time zone | etime
 reservationid | integer                | reservationid
primary key, btree, for table "cs421g84.staffby"
```

```
           Table "cs421g84.supplier"
   Column    |         Type          | Modifiers
-------------+-----------------------+-----------
 delivers    | boolean               |
 companyname | character varying(20) | not null
 address     | character varying(20) |
Indexes:
    "supplier_pkey" PRIMARY KEY, btree (companyname)
Referenced by:
    TABLE "ingredients" CONSTRAINT "ingredients_companyname_fkey" FOREIGN KEY (companyname) REFERENCES supplier(companyname)
    TABLE "invoice" CONSTRAINT "invoice_suppliername_fkey" FOREIGN KEY (suppliername) REFERENCES supplier(companyname)

         Index "cs421g84.supplier_pkey"
   Column    |         Type          | Definition
-------------+-----------------------+-------------
 companyname | character varying(20) | companyname
primary key, btree, for table "cs421g84.supplier"

           Table "cs421g84.venue"
  Column   |         Type          | Modifiers
-----------+-----------------------+-----------
 address   | character varying(20) | not null
 kitchen   | boolean               |
 tableware | boolean               |
 rented    | boolean               |
Indexes:
    "venue_pkey" PRIMARY KEY, btree (address)
Referenced by:
    TABLE "event" CONSTRAINT "event_address_fkey" FOREIGN KEY (address) REFERENCES venue(address)

        Index "cs421g84.venue_pkey"
 Column  |         Type          | Definition
---------+-----------------------+------------
 address | character varying(20) | address
primary key, btree, for table "cs421g84.venue"
```

Question 3

For additional inserts, we created our own methods in the attached java file which would generate random inserts for all of our tables. Below are 5 examples of what the inserts that were generated.

INSERT INTO staffby VALUES ('2014-06-16', '00:42:27', 1988, 290, 531, 923, 848, 768, 310, NULL, NULL, NULL, NULL);

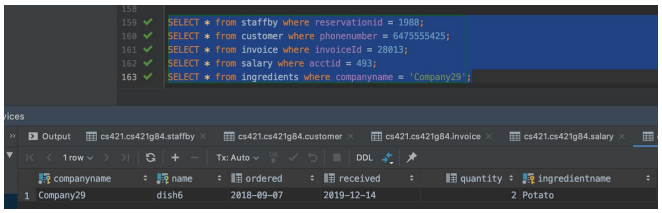
INSERT INTO invoice VALUES ('2019-02-28', 1446, 'ingredients purchase', 28013,'rejected', NULL, 'Company12', 493);

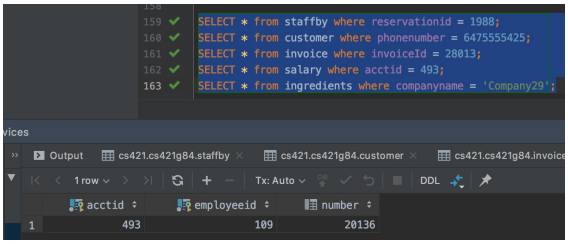
INSERT INTO salary VALUES (493, 109, 20136);

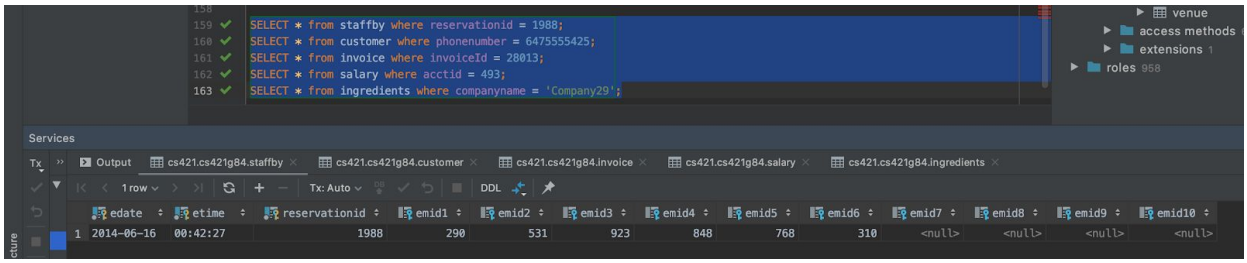
INSERT INTO customer VALUES ('2014-06-28', '2016-09-22', 'C.Smith@fakemail.com', 6475555425, 'Caroline B. Smith') ;

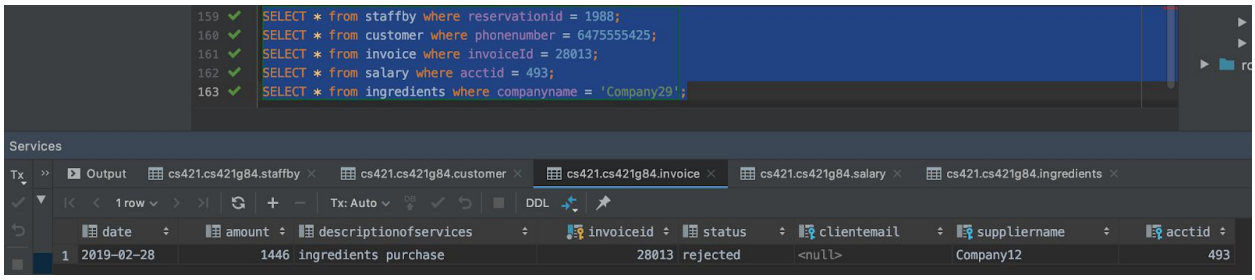
INSERT INTO ingredients VALUES ('Company29', 'dish6', '2018-09-07', '2019-12-14', 2, 'Potato');

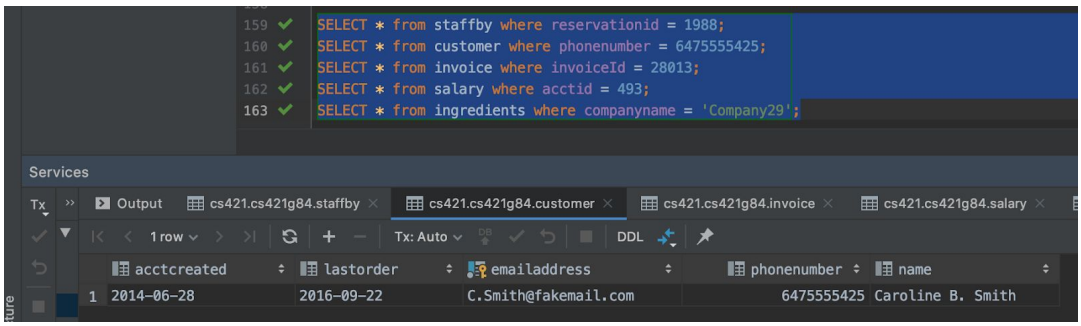
```
cs421.cs421g84> INSERT INTO staffby VALUES ('2014-06-16', '00:42:27', 1988, 290, 531, 923, 848, 768, 310, NULL, NULL, NULL, NULL)
[2020-02-26 17:07:17] 1 row affected in 11 ms
cs421.cs421g84> INSERT INTO invoice VALUES ('2019-02-28', 1446, 'ingredients purchase', 28013,'rejected', NULL, 'Company12', 493)
[2020-02-26 17:07:17] 1 row affected in 12 ms
cs421.cs421g84> INSERT INTO salary VALUES (493, 109, 20136)
[2020-02-26 17:07:17] 1 row affected in 10 ms
cs421.cs421g84> INSERT INTO customer VALUES ('2014-06-28', '2016-09-22', 'C.Smith@fakemail.com', 6475555425, 'Caroline B. Smith')
[2020-02-26 17:07:18] 1 row affected in 10 ms
cs421.cs421g84> INSERT INTO ingredients VALUES ('Company29', 'dish6', '2018-09-07', '2019-12-14', 2, 'Potato')
[2020-02-26 17:07:18] 1 row affected in 8 ms
```











Question 4

Below are our results from select * from table_name limit 10. Please note that accountreceivable and accountpayable only have 8 and 2 entries respectively. This was a design choice.

```
cs421.cs421g84> select * from has limit 10
[2020-02-26 17:16:11] 10 rows retrieved starting from 1 in 126 ms (execution: 13 ms, fetching: 113 ms)
```

cs421.cs421g84.has

10 rows | Tx: Auto | DDL

	menuid	edate	etime	reservationid
1	475	2016-12-04	02:33:17	1093
2	840	2014-06-16	00:42:27	1988
3	434	2018-09-20	04:04:15	1933
4	475	2015-11-24	03:12:27	1937
5	738	2017-10-07	00:22:51	1293
6	789	2019-04-14	02:04:25	1653
7	49	2016-07-13	04:28:10	1242
8	469	2014-01-12	03:00:46	1365
9	720	2019-09-19	04:19:12	1601
10	76	2016-11-04	00:25:16	1764

```
cs421.cs421g84> select * from ingredients limit 10
[2020-02-26 17:16:18] 10 rows retrieved starting from 1 in 71 ms (execution: 20 ms, fetching: 51 ms)
```

cs421.cs421g84.ingredients

10 rows | Tx: Auto | DDL

	companyname	name	ordered	received	quantity	ingredientname
1	Company17	dish17	2017-09-11	2015-07-15	54	Eggs
2	Company10	dish7	2018-03-16	2016-06-05	67	Nuts
3	Company9	dish30	2016-01-20	2019-11-27	99	Cider
4	Company7	dish8	2018-11-20	2019-08-25	54	Chicken
5	Company1	dish9	2017-03-13	2016-03-20	11	Olive Oil
6	Company24	dish35	2014-07-14	2016-07-27	80	Beef
7	Company18	dish21	2013-11-03	2017-04-17	63	Fruits
8	Company26	dish9	2019-05-13	2017-08-13	71	Leafy Greens
9	Company12	dish22	2013-07-16	2013-01-23	16	Spices
10	Company26	dish26	2017-01-19	2019-01-13	87	Wine

```
cs421.cs421g84> select * from supplier limit 10
[2020-02-26 17:17:30] 10 rows retrieved starting from 1 in 63 ms (execution: 13 ms, fetching: 50 ms)
```

cs421.cs421g84.supplier

10 rows | Tx: Auto | DDL

	delivers	companyname	address
1	☐	Company0	0 street av
2	☑	Company1	1 street av
3	☐	Company2	2 street av
4	☐	Company3	3 street av
5	☑	Company4	4 street av
6	☐	Company5	5 street av
7	☑	Company6	6 street av
8	☐	Company7	7 street av
9	☑	Company8	8 street av
10	☑	Company9	9 street av

```
cs421.cs421g84> select * from staffby limit 10
[2020-02-26 17:17:20] 10 rows retrieved starting from 1 in 123 ms (execution: 14 ms, fetching: 109 ms)
```

cs421.cs421g84.staffby

10 rows | Tx: Auto | DDL | Tab-se…d (TSV)

	edate	etime	reservationid	emid1	emid2	emid3	emid4	emid5	emid6	emid7	emid8	emid9	emid10
1	2014-06-16	00:42:27	1988	290	531	923	848	768	310	<null>	<null>	<null>	<null>
2	2016-12-04	02:33:17	1093	132	<null>	<null>	<null>	<null>	<null>	<null>	<null>	<null>	<null>
3	2018-09-20	04:04:15	1933	109	694	863	336	<null>	<null>	<null>	<null>	<null>	<null>
4	2015-11-24	03:12:27	1937	548	<null>	<null>	<null>	<null>	<null>	<null>	<null>	<null>	<null>
5	2017-10-07	00:22:51	1293	461	761	336	740	213	464	400	289	384	535
6	2019-04-14	02:04:25	1653	453	846	225	604	801	19	403	<null>	<null>	<null>
7	2016-07-13	04:28:10	1242	17	964	517	424	12	595	<null>	<null>	<null>	<null>
8	2014-01-12	03:00:46	1365	506	566	336	582	424	768	<null>	<null>	<null>	<null>
9	2019-09-19	04:19:12	1601	848	863	517	400	244	420	752	964	397	21
10	2016-11-04	00:25:16	1764	310	360	395	132	<null>	<null>	<null>	<null>	<null>	<null>

```
cs421.cs421g84> select * from invoice limit 10
[2020-02-26 17:17:38] 10 rows retrieved starting from 1 in 66 ms (execution: 15 ms, fetching: 51 ms)
```

cs421.cs421g84.invoice

10 rows | Tx: Auto | DDL

	date	amount	descriptionofservices	invoiceid	status	clientemail	suppliername	acctid
1	2019-02-28	1446	ingredients purchase	28013	rejected	<null>	Company12	493
2	2013-06-13	9540	ingredients purchase	27024	under review	<null>	Company22	668
3	2016-12-21	2930	ingredients purchase	26887	under review	<null>	Company6	493
4	2013-09-26	1649	ingredients purchase	21281	rejected	<null>	Company6	668
5	2017-04-20	9547	reservation payment	29721	pending	C6@fakemail.com	<null>	304
6	2018-10-25	8850	reservation payment	24509	rejected	C7@fakemail.com	<null>	69
7	2017-10-10	2657	ingredients purchase	29265	rejected	<null>	Company0	699
8	2018-05-17	6195	ingredients purchase	21405	rejected	<null>	Company21	837
9	2017-11-03	6428	reservation payment	23770	rejected	C33@fakemail.com	<null>	69
10	2016-09-24	3848	reservation payment	23697	rejected	C3@fakemail.com	<null>	304

```
cs421.cs421g84> select * from invoice limit 10
[2020-02-26 17:16:26] 10 rows retrieved starting from 1 in 94 ms (execution: 13 ms, fetching: 81 ms)
```

cs421.cs421g84.invoice

10 rows | Tx: Auto | DDL

	date	amount	descriptionofservices	invoiceid	status	clientemail	suppliername	acctid
1	2019-02-28	1446	ingredients purchase	28013	rejected	<null>	Company12	493
2	2013-06-13	9540	ingredients purchase	27024	under review	<null>	Company22	668
3	2016-12-21	2930	ingredients purchase	26887	under review	<null>	Company6	493
4	2013-09-26	1649	ingredients purchase	21281	rejected	<null>	Company6	668
5	2017-04-20	9547	reservation payment	29721	pending	C6@fakemail.com	<null>	304
6	2018-10-25	8850	reservation payment	24509	rejected	C7@fakemail.com	<null>	69
7	2017-10-10	2657	ingredients purchase	29265	rejected	<null>	Company0	699
8	2018-05-17	6195	ingredients purchase	21405	rejected	<null>	Company21	837
9	2017-11-03	6428	reservation payment	23770	rejected	C33@fakemail.com	<null>	69
10	2016-09-24	3848	reservation payment	23697	rejected	C3@fakemail.com	<null>	304

```
cs421.cs421g84> select * from prepares limit 10
[2020-02-26 17:16:46] 10 rows retrieved starting from 1 in 96 ms (execution: 26 ms, fetching: 70 ms)
```

cs421.cs421g84.prepares

10 rows | Tx: Auto | DDL

	menuid	employeeid
1	974	109
2	109	768
3	403	340
4	616	572
5	32	923
6	719	19
7	774	638
8	789	846
9	289	752
10	434	229

```
cs421.cs421g84> select * from menu limit 10
[2020-02-26 17:16:38] 10 rows retrieved starting from 1 in 82 ms (execution: 14 ms, fetching: 68 ms)
```

cs421.cs421g84.menu

10 rows | Tx: Auto | DDL

	numcourses	menuid	seasonal
1	4	974	yes
2	0	109	yes
3	0	403	yes
4	4	616	no
5	0	32	no
6	2	719	no
7	3	774	no
8	0	789	yes
9	3	289	yes
10	3	434	no

```
cs421.cs421g84> select * from reservation limit 10
[2020-02-26 17:16:52] 10 rows retrieved starting from 1 in 73 ms (execution: 14 ms, fetching: 59 ms)
```

cs421.cs421g84.reservation

10 rows | Tx: Auto | DDL

	rdate	rtime	reservationid	clientemail
1	2016-04-27	08:58:14	1093	C86@fakemail.com
2	2018-12-04	05:52:47	1988	A.Boyle@fakemail.com
3	2014-02-11	02:16:30	1933	C52@fakemail.com
4	2018-08-25	05:45:24	1937	C63@fakemail.com
5	2018-04-28	13:14:06	1293	C41@fakemail.com
6	2018-03-09	12:04:22	1653	C40@fakemail.com
7	2015-01-02	12:12:43	1242	C68@fakemail.com
8	2014-06-07	20:55:55	1365	C46@fakemail.com
9	2014-12-21	18:56:34	1601	C38@fakemail.com
10	2019-01-06	01:59:09	1764	C16@fakemail.com

```
cs421.cs421g84> select * from event limit 10
[2020-02-26 17:16:03] 10 rows retrieved starting from 1 in 113 ms (execution: 13 ms, fetching: 100 ms)
```

cs421.cs421g84.event

10 rows | Tx: Auto | DDL

	corporate	staffamount	etime	mature	eventdescription	edate	attendees	address	reservationid
1	☑	97	04:00:46	☐	party	2013-09-24	50	32 street av	1093
2	☐	1	02:33:17	☐	party	2016-12-04	54	2 street av	1093
3	☐	6	00:42:27	☐	wedding	2014-06-16	16	32 street av	1988
4	☐	4	04:04:15	☑	party	2018-09-20	12	35 street av	1933
5	☑	1	03:12:27	☑	wedding	2015-11-24	36	10 street av	1937
6	☑	10	00:22:51	☑	conferences	2017-10-07	45	34 street av	1293
7	☐	7	02:04:25	☑	wedding	2019-04-14	86	38 street av	1653
8	☑	6	04:28:10	☑	conferences	2016-07-13	51	33 street av	1242
9	☑	6	03:00:46	☑	wedding	2014-01-12	86	3 street av	1365
10	☑	10	04:19:12	☑	party	2019-09-19	6	35 street av	1601

```
cs421.cs421g84> select * from drink limit 10
[2020-02-26 17:15:55] 10 rows retrieved starting from 1 in 62 ms (execution: 19 ms, fetching: 43 ms)
```

cs421.cs421g84.drink

10 rows | Tx: Auto | DDL

	name	mature
1	dish4	☐
2	dish9	☐
3	dish10	☐
4	dish11	☐
5	dish13	☑
6	dish19	☐
7	dish23	☐
8	dish25	☐
9	dish33	☑
10	dish34	☐

```
cs421.cs421g84> select * from customer limit 10
[2020-02-26 17:15:44] 10 rows retrieved starting from 1 in 53 ms (execution: 18 ms, fetching: 35 ms)
```

cs421.cs421g84.customer

10 rows | Tx: Auto | DDL

	acctcreated	lastorder	emailaddress	phonenumber	name
1	2020-02-16	2020-02-16	M.Mcsally@fakemail.com	6475550001	Martha Mcsally
2	2013-05-15	2018-06-08	C.Krebs@fakemail.com	6475550311	Charlene Krebs
3	2013-06-20	2015-09-06	W.Dell@fakemail.com	6475553351	William L. Dell
4	2013-07-02	2016-09-22	M.Shim@fakemail.com	6475552351	Melanie J. Shim
5	2014-04-08	2016-10-28	A.Laughlin@fakemail.com	6475553561	Alfredo R. Laughlin
6	2015-06-04	2017-07-11	B.Mendez@fakemail.com	6475553452	Brandy M. Mendez
7	2016-09-22	2018-06-08	F.Williams@fakemail.com	6475552456	Florence R. Williams
8	2018-06-08	2019-06-24	H.Jones@fakemail.com	6475553466	Harriet D. Jones
9	2019-04-10	2019-07-24	K.Bunton@fakemail.com	6475553563	Kevin M. Bunton
10	2019-06-24	2019-06-24	A.Boyle@fakemail.com	6475555621	Austin L. Boyle

```
cs421.cs421g84> select * from accountreceivable limit 10
[2020-02-26 17:15:25] 2 rows retrieved starting from 1 in 70 ms (execution: 12 ms, fetching: 58 ms)
```

cs421.cs421g84.accountreceivable

2 rows | Tx: Auto | DDL

	revenue	acctid
1	6695	304
2	4885	69

```
cs421.cs421g84> select * from staff limit 10
[2020-02-26 17:17:14] 10 rows retrieved starting from 1 in 65 ms (execution: 17 ms, fetching: 48 ms)
```

cs421.cs421g84.staff

10 rows | Tx: Auto | DDL

	role	employedsince	name	employeeid
1	manager	2014-03-04	staff0	109
2	manager	2017-05-17	staff1	768
3	manager	2013-04-26	staff2	340
4	accountant	2019-12-13	staff3	572
5	accountant	2013-07-20	staff4	923
6	chef	2013-06-22	staff5	19
7	waitstaff	2016-07-05	staff6	638
8	chef	2016-09-20	staff7	846
9	manager	2016-02-09	staff8	752
10	accountant	2016-07-10	staff9	229

```
cs421.cs421g84> select * from salary limit 10
[2020-02-26 17:17:07] 10 rows retrieved starting from 1 in 86 ms (execution: 15 ms, fetching: 71 ms)
```

cs421.cs421g84.salary

10 rows | Tx: Auto | DDL

	acctid	employeeid	number
1	685	768	24327
2	685	340	26222
3	824	572	34569
4	699	923	33106
5	699	19	26044
6	863	638	21283
7	699	846	26880
8	668	752	24332
9	777	229	24522
10	863	548	28758

```
cs421.cs421g84> select * from runs limit 10
[2020-02-26 17:16:59] 10 rows retrieved starting from 1 in 114 ms
```

cs421.cs421g84.runs

10 rows | Tx: Auto

	employeeid	acctid
1	109	699
2	768	777
3	340	824
4	572	685
5	923	304
6	19	493
7	638	863
8	846	837
9	752	69
10	229	668

```
cs421.cs421g84> select * from contains limit 10
[2020-02-26 17:15:15] 10 rows retrieved starting from 1 in 69 ms
```

cs421.cs421g84.contains

10 rows | Tx: Auto

	menuid	name
1	974	dish0
2	109	dish1
3	403	dish2
4	616	dish3
5	32	dish4
6	719	dish5
7	774	dish6
8	789	dish7
9	289	dish8
10	434	dish9

```
cs421.cs421g84> select * from accountpayable limit 10
[2020-02-26 17:14:58] 8 rows retrieved starting from 1 in 63 ms (execution: 12 ms, fetching: 51 ms)
```

cs421.cs421g84.accountpayable

8 rows | Tx: Auto | DDL

	budget	acctid
1	1708	699
2	3669	777
3	365	824
4	6848	685
5	6185	493
6	6573	863
7	1415	837
8	4158	668

```
cs421.cs421g84> select * from account limit 10
[2020-02-26 17:13:29] 10 rows retrieved starting from 1 in 79 ms (execution: 22 ms, fetching: 57 ms)
```

cs421.cs421g84.account

10 rows | Tx: Auto | DDL

	acctid
1	699
2	777
3	824
4	685
5	304
6	493
7	863
8	837
9	69
10	668

Question 5

1. Query returns the employee ID, role and name of the staff that are in charge of the accounting tied to a certain account.

Tables used: staff and runs

```
create table staff
(
   role         varchar(20),
   employedsince date,
   name         varchar(20),
   employeeid    integer not null
      constraint staff_pkey
         primary key
);
create table runs
(
   employeeid integer not null
      constraint runs_employeeid_fkey
         references staff,
   acctid     integer not null
      constraint runs_acctid_fkey
         references account,
   constraint runs_pkey
      primary key (employeeid, acctid)
);
```

Query:

```
cs421.cs421g84> select staff.employeeid, role, name
                from staff join runs on staff.employeeid = runs.employeeid
                where runs.acctid = 304
[2020-02-26 17:04:53] 3 rows retrieved starting from 1 in 197 ms (execution: 133 ms, fetching: 64 ms)
```

	employeeid	role	name
1	923	accountant	staff4
2	548	accountant	staff10
3	996	waitstaff	staff20

2. Query returns the name and email address of all customers who have unpaid (pending) invoices for events that they have made reservations for. Query additionally requires that the response have an email address (i.e. only private customers and no companies).

Tables used: customer and invoice

```
create table customer
(
   acctcreated  date,
   lastorder    date,
   emailaddress varchar(30) not null
      constraint customer_pkey
         primary key,
   phonenumber  bigint,
   name         varchar(20)
);
create table invoice
(
   date                 date,
   amount               numeric,
   descriptionofservices varchar(100),
   invoiceid            integer not null
      constraint invoice_pkey
         primary key,
   status               varchar(20),
   clientemail          varchar(30)
      constraint invoice_clientemail_fkey
         references customer,
   suppliername         varchar(20)
      constraint invoice_suppliername_fkey
         references supplier,
   acctid               integer
      constraint invoice_acctid_fkey
         references account );
```

Query:

```
cs421.cs421g84> select name, emailaddress
                from customer
                where emailaddress in (select clientemail
                                       from invoice
                                       where descriptionofservices = 'reservation payment' and status = 'pending')
                  and emailaddress IS NOT NULL
[2020-02-26 17:06:23] 49 rows retrieved starting from 1 in 90 ms (execution: 18 ms, fetching: 72 ms)
```

	name	emailaddress
1	Melanie J. Shim	M.Shim@fakemail.com
2	Alfredo R. Laughlin	A.Laughlin@fakemail.com
3	Brandy M. Mendez	B.Mendez@fakemail.com
4	Harriet D. Jones	H.Jones@fakemail.com
5	Kevin M. Bunton	K.Bunton@fakemail.com
6	Austin L. Boyle	A.Boyle@fakemail.com
7	Angela W. Leon	A.Leon@fakemail.com
8	Robert J. Belanger	R.Belanger@fakemail.com
9	Customer1	C1@fakemail.com
10	Customer3	C3@fakemail.com
11	Customer4	C4@fakemail.com
12	Customer5	C5@fakemail.com
13	Customer6	C6@fakemail.com
14	Customer7	C7@fakemail.com
15	Customer9	C9@fakemail.com
16	Customer11	C11@fakemail.com
17	Customer12	C12@fakemail.com
18	Customer16	C16@fakemail.com

3. Query selects all invoices from April of 2019, utilizing the extract function to retrieve the date and month.

Tables used: invoice

```
create table invoice
(
    date                date,
    amount              numeric,
    descriptionofservices varchar(100),
    invoiceid           integer not null
        constraint invoice_pkey
            primary key,
    status              varchar(20),
    clientemail         varchar(30)
        constraint invoice_clientemail_fkey
            references customer,
    suppliername        varchar(20)
        constraint invoice_suppliername_fkey
```

```
        references supplier,
    acctid              integer
        constraint invoice_acctid_fkey
            references account
);
```

Query:

```
cs421.cs421g84> select invoiceid, date, amount from invoice
                where extract(year from date) = 2019 and extract(month from date) = 04
[2020-02-26 17:08:17] 6 rows retrieved starting from 1 in 79 ms (execution: 15 ms, fetching: 64 ms)
```

	invoiceid	date	amount
1	26619	2019-04-08	2988
2	25734	2019-04-08	8683
3	15802	2019-04-07	9069
4	15854	2019-04-02	2675
5	14679	2019-04-28	2230
6	15020	2019-04-19	7505

4. Query selects all valid menus (non-zero courses) and returns all those that also contain at least one dish that contains a vegetarian option.

Tables used: menu and dish

```
 create table menu
(
    numcourses integer,
    menuid     integer not null
        constraint menu_pkey
            primary key,
    seasonal   varchar(20)
);
create table dish
(
    name              varchar(20) not null
        constraint dish_pkey
            primary key,
    preptime          time,
    type              varchar(20),
    dietaryrestriction varchar(20),
    frozen            boolean
);
```

Query:

```
cs421.cs421g84> select menuid from menu
                where numcourses > 0
                intersect
                select menuid from contains
                where name in (select name
                               from dish
                               where type = 'vegetarian')
[2020-02-26 17:10:00] 18 rows retrieved starting from 1 in 63 ms (execution: 17 ms, fetching: 46 ms)
```

	menuid
1	677
2	774
3	138
4	458
5	616
6	434
7	720
8	840
9	424
10	289
11	779
12	268
13	354
14	65
15	719
16	738
17	475
18	76

5. Query selects all venues that do not have a kitchen and will be supporting a large number of attendees (> 25).

Tables used: venue

```
create table venue
(
   address   varchar(20) not null
      constraint venue_pkey
         primary key,
   kitchen   boolean,
   tableware boolean,
   rented    boolean
);
```

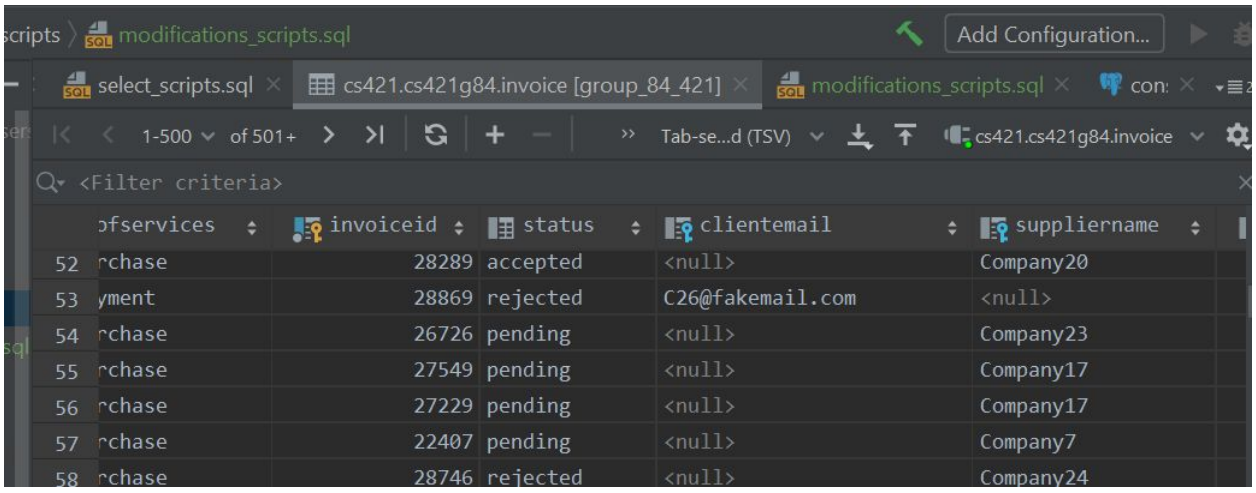
Query:

```
cs421.cs421g84> select address from venue
                where kitchen = false
                  and exists (select address
                              from event
                              where attendees > 25)
[2020-02-26 17:11:14] 30 rows retrieved starting from 1 in 138 ms (execution: 13 ms, fetching: 125 ms)
```

	address
Select All	av
2	4 street av
3	5 street av
4	7 street av
5	10 street av
6	11 street av
7	12 street av
8	13 street av
9	14 street av
10	15 street av
11	16 street av
12	18 street av
13	19 street av
14	21 street av
15	25 street av
16	26 street av
17	27 street av
18	30 street av
19	31 street av
20	32 street av

Question 6

1. Completing payment release for pending invoices for a specific company.
   Initial invoice statuses:



	ofservices	invoiceid	status	clientemail	suppliername
52	rchase	28289	accepted	<null>	Company20
53	yment	28869	rejected	C26@fakemail.com	<null>
54	rchase	26726	pending	<null>	Company23
55	rchase	27549	pending	<null>	Company17
56	rchase	27229	pending	<null>	Company17
57	rchase	22407	pending	<null>	Company7
58	rchase	28746	rejected	<null>	Company24

Change status of pending to accepted (Accountant accepts to pay pending invoices from Company17):

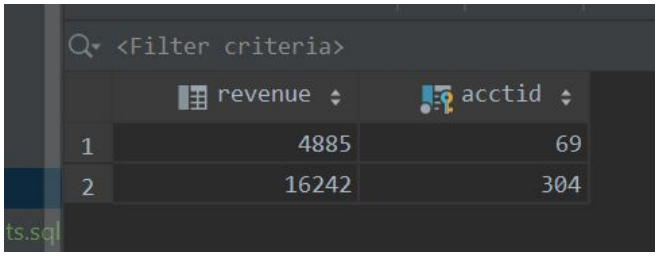
```
cs421.cs421g84> UPDATE invoice SET status = 'accepted' WHERE status = 'pending' AND suppliername = 'Company17'
[2020-02-26 18:14:47] 3 rows affected in 23 ms
```

Updated invoice statuses:

	purchase	27549	accepted	<null>	Company17	685
155	payment	28263	under review	C1@fakemail.com	<null>	69
156	purchase	25629	accepted	<null>	Company17	837
157	payment	22718	rejected	C49@fakemail.com	<null>	69
	purchase	22407	accepted	<null>	Company7	837
	purchase	27229	accepted	<null>	Company17	863
	purchase	21443	accepted	<null>	Company7	668

2. Accept payment from customers and update account balances.

Initial revenue in the different accounts payable accounts:



<Filter criteria>

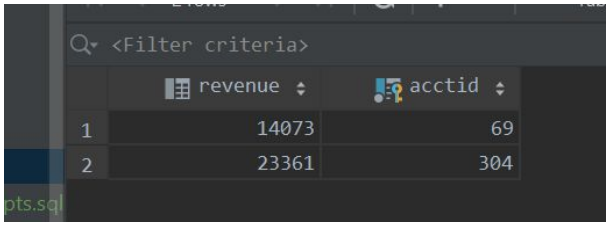
	revenue	acctid
1	4885	69
2	16242	304

ts.sql

Add all incoming pending payments to the respective accounts' revenues:

```
cs421.cs421g84> UPDATE accountreceivable
                    SET revenue = revenue + invoice.amount
                FROM invoice
                WHERE accountreceivable.acctid = invoice.acctid AND status = 'pending' AND clientemail is not nul
[2020-02-26 19:10:29] 2 rows affected in 6 ms
```

Updated account revenues:



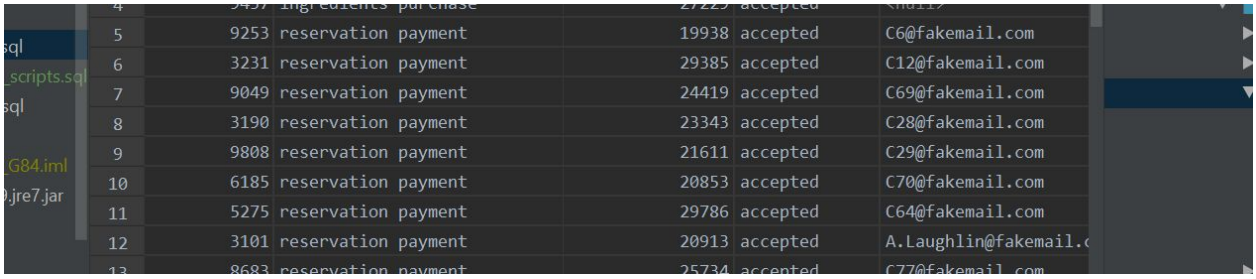
<Filter criteria>

	revenue	acctid
1	14073	69
2	23361	304

pts.sql

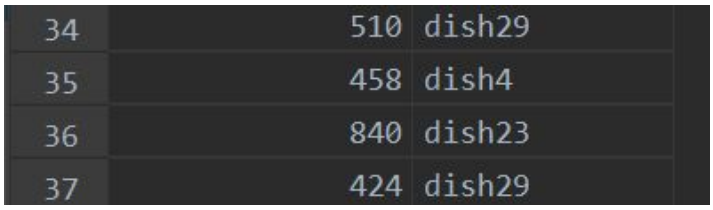
Update all incoming pending invoices to accepted:

```
cs421.cs421g84> UPDATE invoice
                SET status = 'accepted'
                WHERE status = 'pending' AND clientemail is not null
[2020-02-26 19:11:51] 62 rows affected in 11 ms
```



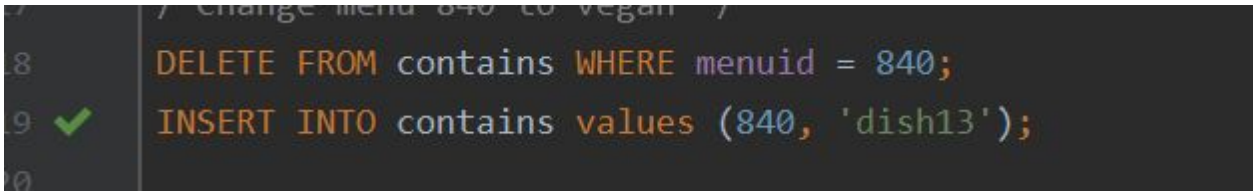
5	9253	reservation payment	19938	accepted	C6@fakemail.com
6	3231	reservation payment	29385	accepted	C12@fakemail.com
7	9049	reservation payment	24419	accepted	C69@fakemail.com
8	3190	reservation payment	23343	accepted	C28@fakemail.com
9	9808	reservation payment	21611	accepted	C29@fakemail.com
10	6185	reservation payment	20853	accepted	C70@fakemail.com
11	5275	reservation payment	29786	accepted	C64@fakemail.com
12	3101	reservation payment	20913	accepted	A.Laughlin@fakemail.c

3. Updating a menu to be vegan (changing the menu with ID 840)

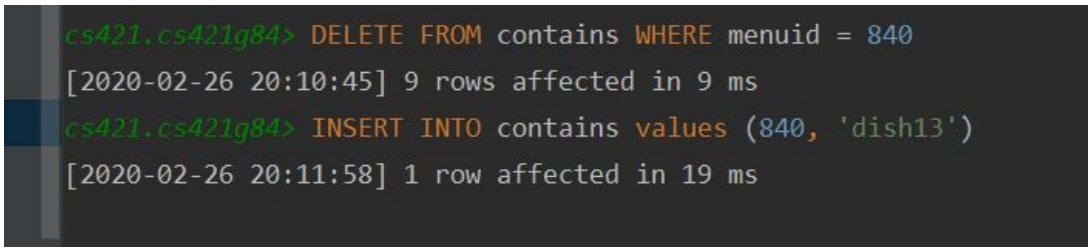


34	510	dish29
35	458	dish4
36	840	dish23
37	424	dish29

Delete initial relationship and then insert new contains relationship:



```
DELETE FROM contains WHERE menuid = 840;
INSERT INTO contains values (840, 'dish13');
```



```
cs421.cs421g84> DELETE FROM contains WHERE menuid = 840
[2020-02-26 20:10:45] 9 rows affected in 9 ms
cs421.cs421g84> INSERT INTO contains values (840, 'dish13')
[2020-02-26 20:11:58] 1 row affected in 19 ms
```

4. This Updates the list of ingredients such that no single dishname is listed as ordering multiples of an ingredient. This script creates a new temporary table (ingredients_temp) that matches the constraints of the table ingredients. It then copies all of the distinct entries from the old table into this new one. It drops the old table (ingredients) and alters the name of the new table to match that of the old. Please note that this was tested on a table of more than 500 entries to ensure a better test. The table was latter trimmed to be smaller than 500 entries for submission purposes.

```
CREATE TABLE ingredients_temp (LIKE ingredients);
INSERT INTO ingredients_temp (companyname, name , ordered, received, quantity, ingredientname)
SELECT
    DISTINCT ON (name, ingredientname) companyname, name,
                        ordered, received, quantity, ingredientname
FROM ingredients;
DROP TABLE ingredients;
ALTER TABLE ingredients_temp
    RENAME TO ingredients;
```

```
cs421.cs421g84> INSERT INTO ingredients_temp (companyname, name , ordered, received, quantity, ingredientname)
                SELECT
                    DISTINCT ON (name, ingredientname) companyname, name,
                                        ordered, received, quantity, ingredientname
                FROM ingredients
[2020-02-27 13:19:13] 656 rows affected in 30 ms
```

Pre: This is a script that prints duplicate entries in the ingredients table that have the same name and ingredient name. 204 rows of dish, ingredient pairings have duplicates.

```
SELECT name, ingredientname
FROM
    (SELECT name, ingredientname,
            ROW_NUMBER() OVER( PARTITION BY name, ingredientname
                ORDER BY  ingredientname ) AS row_num
     FROM ingredients ) t
WHERE t.row_num > 1;
```

Output | cs421.cs421g84.ingredients

204 rows | Tx: Auto | DDL

	name	ingredientname
1	dish0	Beef
2	dish0	Chicken
3	dish0	Legumes
4	dish0	Legumes
5	dish0	Olive Oil
6	dish0	Shellfish
7	dish0	Tofu
8	dish0	Tofu
9	dish1	Potato
10	dish1	Produce
11	dish10	Pork

Post: This was the same script as before that selects all duplicate entries. As you can see, there are no longer any duplicate rows in ingredients.

```
SELECT name, ingredientname
FROM
    (SELECT name, ingredientname,
            ROW_NUMBER() OVER( PARTITION BY name, ingredientname
                ORDER BY  ingredientname ) AS row_num
     FROM ingredients ) t
WHERE t.row_num > 1;

CREATE TABLE ingredients_temp (LIKE ingredients);
```

Output | cs421.cs421g84.ingredients

0 rows | Tx: Auto

name	ingredientname

Question 7

View of all customer invoices: This view is helpful to our company because it allows us to quickly parse all monetary transactions that are involving our clients. It is useful to see the name and email address of our clients alongside the important invoice information. This view also allows us to quickly query how many accepted, pending… etc invoices that a single client may have with the company and email them about payment. This also allows us to check all pending invoices and email the clients accordingly. Therefore, we thought that this was an extremely practical view for our company to have access to. This will make the lives of our accounting department easier.

```
CREATE VIEW CustomerInvoices AS SELECT
customer.emailaddress,
invoice."date",
invoice.amount,
invoice.invoiceid,
invoice.status,
invoice.acctid
FROM
   reservation
   JOIN customer ON reservation.clientemail = customer.emailaddress
   LEFT JOIN invoice ON invoice.clientemail = customer.emailaddress
```

emailaddress	date	amount	invoiceid	status	acctid
C86@fakemail.com	2014-11-13	4040	11360	accepted	69
C86@fakemail.com	2017-10-09	1559	24697	under review	69
A.Boyle@fakemail.con	2014-10-22	263	13855	rejected	304
A.Boyle@fakemail.con	2015-01-25	3751	27161	accepted	304
A.Boyle@fakemail.con	2014-11-06	7245	10670	accepted	69
C52@fakemail.com	2013-06-14	1907	11662	accepted	304
C52@fakemail.com	2019-12-05	3739	14834	rejected	69
C52@fakemail.com	2019-03-05	2352	27001	rejected	304
C63@fakemail.com	2017-10-20	8715	10425	rejected	304
C63@fakemail.com	2016-05-09	9990	28140	rejected	304
C63@fakemail.com	2017-05-22	7897	25790	under review	304
C63@fakemail.com	2013-05-01	3348	26445	rejected	69
C41@fakemail.com	2014-01-08	221	14554	under review	304
C41@fakemail.com	2014-12-07	7840	17596	rejected	304
C41@fakemail.com	2013-03-14	6835	22311	rejected	69

```
CREATE VIEW CustomerInvoices AS SELECT
customer.emailaddress,
invoice."date",
invoice.amount,
invoice.invoiceid,
invoice.status,
invoice.acctid
FROM
    reservation
    JOIN customer ON reservation.clientemail = customer.emailaddress
    LEFT JOIN invoice ON invoice.clientemail = customer.emailaddress
> OK
> Time: 0.033s
```

View of the count of different types of dish in different types of events: We decided to implement this view because it would allow us to start researching our menus and what specifics our clients desire for the types of events that they are putting on. For example, vegetarian menus are the most commonly ordered menu, probably has something to do with an eco conscious world. We can also see that parties are less likely to order a vegan menu over any of the other options. Our company can take this data and market the menus that do well in certain areas to the customer that frequently use our catering for events of the discovered type. This will generate more revenue.

```
CREATE VIEW FoodTypeInEvents AS SELECT
event.eventdescription,
dish."type",
COUNT ( dish."type" )
FROM
    event
    JOIN has ON event.reservationid = has.reservationid
    JOIN menu ON menu.menuid = has.menuid
    JOIN "contains" ON "contains".menuid = menu.menuid
    RIGHT JOIN dish ON dish."name" = "contains"."name"
GROUP BY
    event.eventdescription,
    dish."type"
ORDER BY
    COUNT ( menu.menuid ) DESC;
```

eventdescription	type	count
party	vegetarian	184
wedding	vegetarian	181
wake	vegetarian	178
conferences	vegetarian	153
party	none	112
wake	none	94
wedding	none	92
conferences	none	88
wake	gluten free	87
party	gluten free	86
conferences	gluten free	85
wedding	gluten free	84
conferences	vegan	73
wake	vegan	64
wedding	vegan	61
party	vegan	58

```
CREATE VIEW FoodTypeInEvents AS SELECT
event.eventdescription,
dish."type",
COUNT ( dish."type" )
FROM
    event
    JOIN has ON event.reservationid = has.reservationid
    JOIN menu ON menu.menuid = has.menuid
    JOIN "contains" ON "contains".menuid = menu.menuid
    RIGHT JOIN dish ON dish."name" = "contains"."name"
GROUP BY
    event.eventdescription,
    dish."type"
ORDER BY
    COUNT ( menu.menuid ) DESC
> OK
> Time: 0.031s
```

Update on the view

While attempting to update the first view, the following message was outputted.

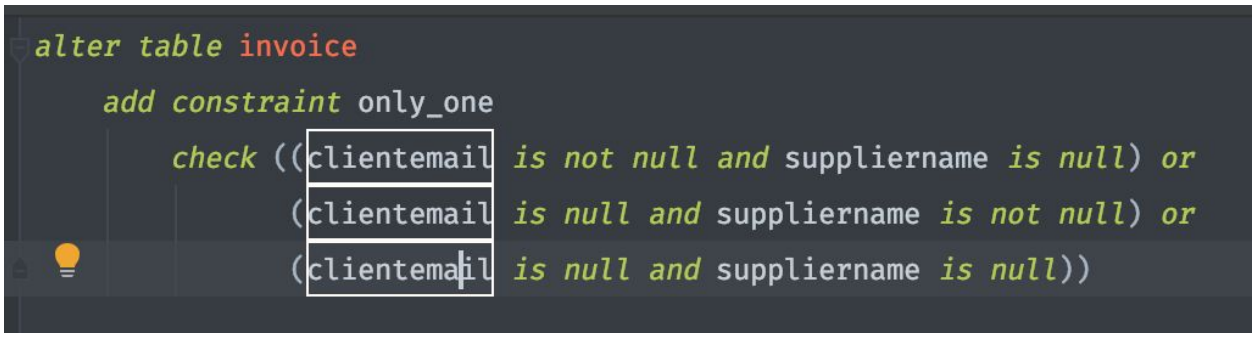
```
UPDATE "customerinvoices" SET amount=1000 WHERE "customerinvoices".invoiceid=11360
> ERROR:  cannot update view "customerinvoices"
DETAIL:  Views that do not select from a single table or view are not automatically updatable.
HINT:  To enable updating the view, provide an INSTEAD OF UPDATE trigger or an unconditional ON UPDATE DO INSTEAD rule.

> Time: 0.069s
```

This makes sense because the view contains a join operation in which information from different tables is being coalesced into a single table. This makes the database unsure of how to update the column since there may be more than a one-to-one relationship so multiple entries in separate tables relate to a single one in our view. At other times, two joined tables might not have any relationship between them and will make deletions very complicated. For this reason, the database insteads defines semantics to specify the way to update a view when such an operation is executed.

Similarly, the operation would not work on the second view since not only was it a join between many tables, it also grouped and aggregated various columns together making the update even harder.

Question 8

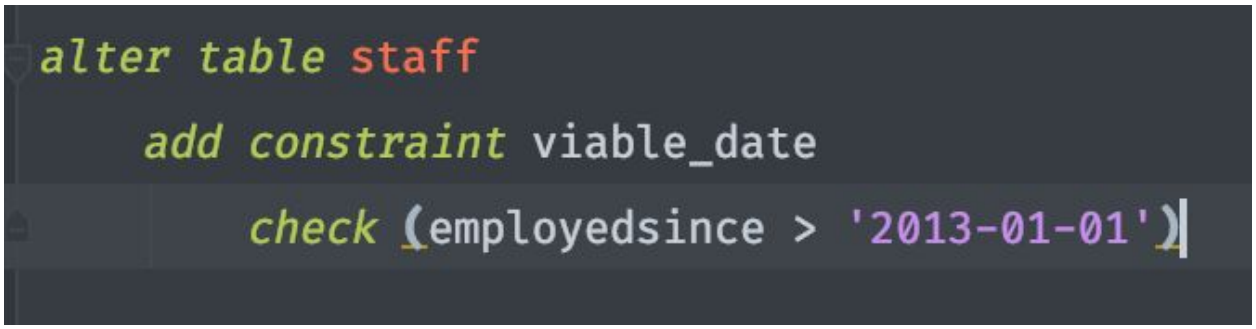
1. Added constraint to invoice, allowing only one of client email or supplier name to be filled.

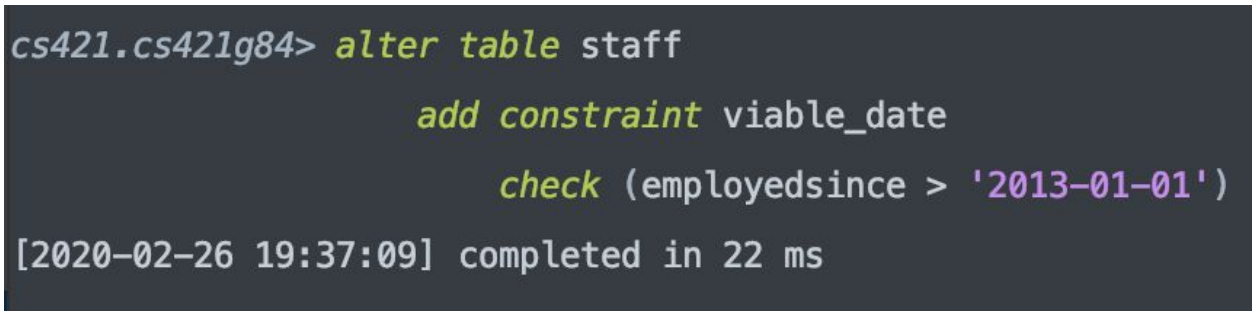



```
cs421.cs421g84> alter table invoice add constraint only_one check ((clientemail is not null and suppliername is null) or (client
[2020-02-26 18:07:57] completed in 30 ms
```

```
cs421.cs421g84> insert into invoice values ('2021-12-21', 9999, 'wake', 12345, 'pending', 'b.saget@fakemail.com', 'bob saget inc', 54321)
[2020-02-26 19:55:52] [23514] ERROR: new row for relation "invoice" violates check constraint "only_one"
[2020-02-26 19:55:52] Detail: Failing row contains (2021-12-21, 9999, wake, 12345, pending, b.saget@fakemail.com, bob saget inc, 54321).
```

2. Added constraint to staff employment dates, disallowing inputs from before our fictional service was founded (1/1/2013).






```
cs421.cs421g84> insert into staff Values ('chef', '2011-01-01', 'Bob Saget', 12345)
[2020-02-26 19:53:59] [23514] ERROR: new row for relation "staff" violates check constraint "viable_date"
[2020-02-26 19:53:59] Detail: Failing row contains (chef, 2011-01-01, Bob Saget, 12345).
```

Question 9

Generating real random data.

For creativity we wrote automated generation scripts for all of the data in our database system. See the attached java code file named tableGenerator.java. Inside of which are a set of methods that generate random data for each table that is contained in our database. Any numeric value or option selected from a list of possibilities is completely random. For example all id's are unique and random in a single table. In tables like our ingredients table. We randomly selected from the list pork, chicken, tofu… etc. to give accurate real data for our generated values. This occured for all similar data across all tables, including booleans, descriptions and labels. The only non-real data we had were names and addresses which we created with an iterator.