Association Descriptions

A manager works one and only one workCrew.
A workCrew is worked by one and only one manager.

A headChef works one and only one workCrew.
A workCrew is worked by one and only one headChef.

A workCrew is worked by two to nine lineCooks.
A lineCook works one and only one workCrew.

A maitreD works one and only one workCrew.
A workCrew is worked by one and only one maitreD.

A workCrew is worked by one to four sousChefs.
A sousChef works one and only one workCrew.

A workCrew contains one to many shiftInstances.
A shiftInstance contains one and only one work crew.

A workCrew is worked by 3 to 7 waitstaff.
A waitstaff works one and only one work crew.

A shiftInstance contains one and only one shift.
A shift contains one to many shiftInstances.

A sousChef teaches one to many mentorships.
A mentorship is taught by one and only one sousChef.

A sous chef learns one to many mentorships.
A mentorship is learned by one and only one sousChef.

A mentorship teachers one and only one menuItem.
A menuItem is taught by zero to many mentorships.

A menu lists one to many itemDetails
An itemdetail is listed in one and only one menu.

A itemDetails is listed on one and only one menuItem.
A menuItem lists one to many itemDetails.

A menuItem contains one to many itemOrders.
An itemOrder is contained in one and only one menuItems

An order contains one to many itemOrders.
A itemOrder is contained in one and only one order.

A customerOrder is ordered in one and only one order.
An order orders one to many customerOrders.

A customer places zero to many customerOrders.
A customerOrder is placed by one and only one customer.

A table is contained in one to many dineIns.
A dineIn contains at one and only one table.

Waitstaff attends one to many tables.
A table is attended to by one and only one waitstaff.

A dishwasher works one and only one workCrew.
A workCrew is worked by one and only one dishwasher.

Attributes Definition

corporateCustomer:
Office Address: The location of the corporation the customer works for in words and numbers
Contact Information: Facts of Various methods used to reach the customer, usually as to speak with them

Customer:
Name: The words used to address a certain customer
Mimi's money: The form of payment a customer uses to pay for their dine-in

dineIn:
Party Size: The total number of people eating together as one group at the restaurant

Employee:
Name: usually 2 words used to identify a person, the restaurant uses names to know and keep track of their employees.

Recipes:
Recipe: new dishes created by the headChef of the restaurant, that are sold at the restaurant.

itemDetails:
Price: The price of the individual article of food
Size: The amount of food served to the customer, categorized in 4 types (children's, large, small,medium)
Volume: The amount of soup served to the customer in ounces

itemOrder:
Quantity: The number of servings given to a customer of a specific food

Stations:
Station: position worked by an employee

Mentorship:
Start Date: the defined beginning moment of time of the guidance of the mentorship for the employee
EndOfMentorship: the defined specific moment of time when the guidance of the mentorship for the employee ends

menuItem:
itemName: the name of the article of food listed on the defined restaurants menu
Spiciness: the level of "spice" contained in the individual article of food
Categories:
CategoryType: The specific section on the menu that the article of food belongs to

Menu:

menuType: The specific menu used, served to customers at specific times during the day
Price: The cost associated with the items listed on the menu and served

Order:
Order Type: A description as to where and how the customer consumed the meal (dinein, takeout)
Payment: The form of payment provided by the customer to pay for the meal
dateOrder: The date the customer placed an order for food at the restaurant

Part Time Employee:
Hourly Rate: A certain amount of pay given to an employee in a work environment per the number of hours he/she works

privateCustomer:
Email Address: The email provided by the customer to be used at the restaurant
Snail-mail Address: The postal address provided by the customer to the restaurant

SalaryEmployee:
salaryAmount: amount that a full time employee gets paid for performing a task at a job.

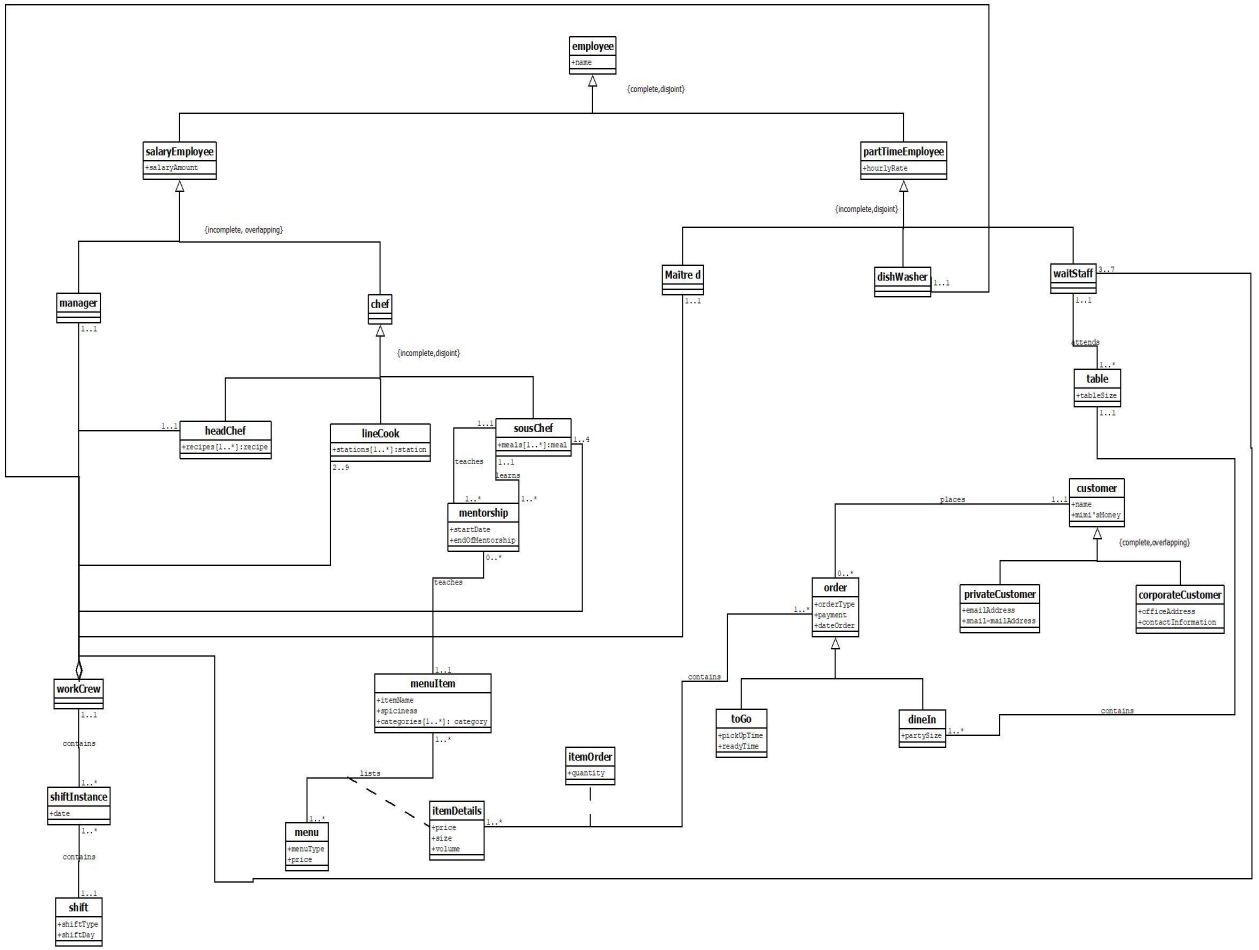
shiftInstance:
Date: certain date with month, day and year when a certain number of employees work.

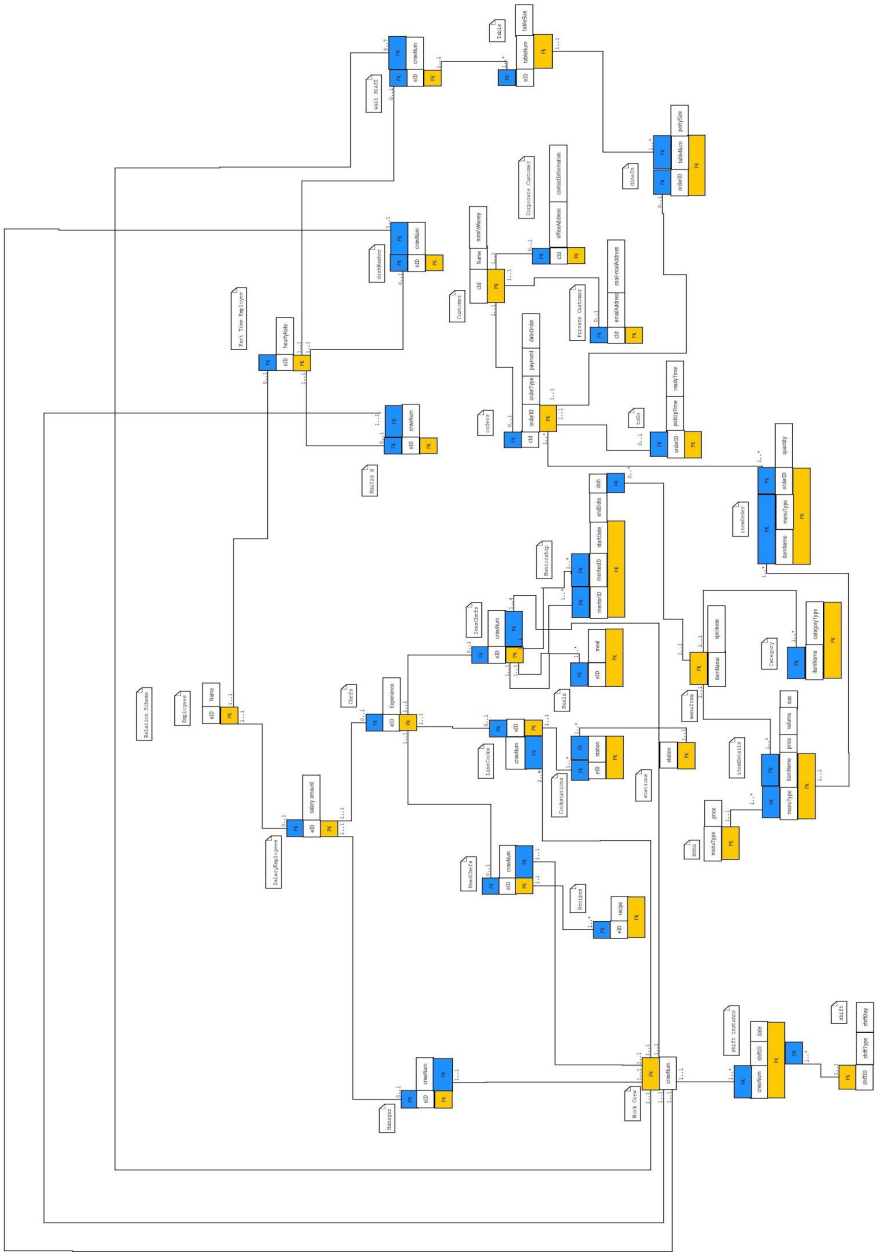
Shift:
shiftType: period when employees worked usually mornings and evenings.
shiftDay: day of the week when an employee works from Sunday to Saturday,

sousChefs:
Meals: certain dishes of food that a sous chef is expert at making, usually the person that best knows how to make them is the one in charge of preparing them for the customers.

toGo:
Pickup time: A specific moment of time when the customer acquired their order
Ready Time: The specific moment of time when the customer's order is available to be picked up

Table:
Table size: The physical size of the table as to how many seats exist at the tables.


employee
+name
{complete,disjoint}
salaryEmployee
+salaryAmount
partTimeEmployee
+hourlyRate
{incomplete, overlapping}
{incomplete,disjoint}
manager
chef
Maitre d
dishWasher
waitStaff
{incomplete,disjoint}
headChef
+recipes[1..*]:recipe
lineCook
+stations[1..*]:station
sousChef
+meals[1..*]:meal
teaches
learns
mentorship
+startDate
+endOfMentorship
attends
table
+tableSize
customer
+name
+wimi'sMoney
places
{complete,overlapping}
order
+orderType
+payment
+dateOrder
privateCustomer
+emailAddress
+mail=mailAddress
corporateCustomer
+officeAddress
+contactInformation
toGo
+pickUpTime
+readyTime
dineIn
+partySize
contains
workCrew
contains
shiftInstance
+date
contains
shift
+shiftType
+shiftDay
menuItem
+itemName
+spiciness
+categories[1..*]: category
lists
menu
+menuType
+price
itemDetails
+price
+size
+volume
itemOrder
+quantity



DDL

```
create table employees(
   eID int NOT NULL,
   eName VARCHAR(40) not null,
   PRIMARY KEY(eID)
);
```

```
create table salaryEmployees(
    eID int NOT NULL,
    salary int NOT NULL,
    primary key(eID),
    CONSTRAINT salaryEmployee_fk FOREIGN KEY (eID) REFERENCES employees (eID)
);

create table partTimeEmployees(
    eID int not null,
    hourlyRate int not null,
    primary key(eID),
   constraint partTime_fk foreign key(eID) references employees(eID)
);

create table managers(
    eID int not null,
    creNum int not null,
    primary key(eID),
    constraint manager_fk foreign key(eID) references salaryEmployees(eID),
    constraint wk_manager foreign key(creNum) references workCrews(crewNum)
);

create table chefs(
    eID int not null,
    experience VARCHAR(35) not null,
    primary key(eID),
    constraint chefs_fk foreign key(eID) references salaryEmployees(eID)
);

create table headChefs(
    eID INT NOT NULL,
    crewNum int not null,
    primary Key(eID),
    constraint headChef_chef_fk foreign key(eID) references chefs(eID),
    constraint headChef_workcrew_fk foreign key(crewNum) references workCrews(crewNum)
);

create table lineCooks(
    eID INT NOT NULL,
    crewNum int not null,
    primary Key(eID),
    constraint lineCooks_chef_fk foreign key(eID) references chefs(eID),
    constraint lineCooks_workcrew_fk foreign key(crewNum) references workCrews(crewNum)
```

```
);

create table sousChefs(
    eID INT NOT NULL,
    crewNum int not null,
    primary Key(eID),
    constraint sousChef_chef_fk foreign key(eID) references chefs(eID),
    constraint sousChef_workcrew_fk foreign key(crewNum) references workCrews(crewNum)

);

create table recipes(
    eID INT NOT NULL,
    recipe varchar(35) not null,
    primary key(eID,recipe),
    constraint recipes_headChef_fk foreign key(eID) references headChefs(eID)
);

create table stations(
    station varchar(35) not null,
    primary key(station)
);

create table lineCookStation(
    eID INT NOT NULL,
    station varchar(35) not null,
    primary key(eID, station),
    constraint linecook_linecook_fk foreign key(eID) REFERENCES lineCooks(eID),
    constraint lineCook_station_fk foreign key(station) references stations(station)
);


create table meals(
    eID INT NOT NULL,
    meal varchar(35) not null,
    primary key(eID, meal),
    constraint sousChef_meal_fk foreign key(eID) REFERENCES sousChefs(eID)
);

create table mentorship(
    mentorID int not null,
    menteeID int not null,
    dish varchar(40) not null,
```

```
    startDate DATE NOT NULL,
    endDate DATE not null,
    primary key(mentorID, menteeID, dish, startDate),
    constraint sousChef_mentorship1_fk foreign key(mentorID) REFERENCES sousChefs(eID),
    constraint sousChef_mentorship2_fk foreign key(menteeID) REFERENCES sousChefs(eID),
    constraint mentorship_dish_fk foreign key(dish) references menuItems(itemName)
);

create table maitreD(
    eID int not null,
    crewNum int not null,
    primary key(eID),
    CONSTRAINT maitreD_partime_fk FOREIGN KEY(eID) references
partTimeEmployees(eID),
    CONSTRAINT maitreD_workshift_fk foreign key(crewNum) references workCrews(crewNum)
);

create table dishWashers(
    eID int not null,
    crewNum int not null,
    primary key(eID),
    CONSTRAINT dishwasher_partime_fk FOREIGN KEY(eID) references
partTimeEmployees(eID),
    CONSTRAINT dishwasher_workshift_fk foreign key(crewNum) references
workCrews(crewNum)
);

create table waitStaffs(
    eID int not null,
    crewNum int not null,
    primary key(eID),
    CONSTRAINT waitStaff_partime_fk FOREIGN KEY(eID) references
partTimeEmployees(eID),
    CONSTRAINT waitStaff_workshift_fk foreign key(crewNum) references
workCrews(crewNum)
);

create table tables(
    eID int not null,
    tableNum int not null,
    tableSize int not null,
    primary key(tableNum),
    constraint tables_employees_fk foreign key(eID) REFERENCES waitStaffs(eID);
);
```

```
create table customers(
    cID int NOT NULL,
    cName VARCHAR(40) not null,
    cMoney int,
    primary Key(cID)
);

create table privateCustomers(
    cID int not null,
    email varchar(40) not null,
    snail_mailAddress varchar(40) not null,
    primary key(cID),
    constraint private_customer_fk foreign key(cID) REFERENCES customers(cID)
);

create table corporateCustomers(
    cID int not null,
    oficceAddress varchar(40) not null,
    contactInfo varchar(40) not null,
    primary key(cID),
    constraint corporate_customer_fk foreign key(cID) REFERENCES customers(cID)
);


create table orders(
    cID int not null,
    orderID int not null,
    orderType varchar(40) not null,
    payment varchar(40) not null,
    dateOrder date not null,
    primary key(orderID),
    constraint orders_customer_fk foreign key(cID) REFERENCES customers(cID)
);

create table dineIn(
    orderID int not null,
    tableNum int not null,
    partSize int not null,
    primary key(orderID, tableNum),
    constraint dineIn_orders_fk foreign key(orderID) references orders(orderID),
    constraint dineIn_table_fk foreign key(tableNum) references tables(tableNum)
);
```

```
create table toGo(
    orderID int not null,
    pickUpTime time not null,
    readyTime time not null,
    primary key(orderID),
    constraint toGo_orders_fk foreign key(orderID) references orders(orderID)
);

create table menus(
    menuType VARCHAR(40) NOT NULL,
    price int,
    primary key(menuType)
);

create table itemDetails(
    menuType varchar(40) not null,
    itemName varchar(40) not null,
    price int,
    volume varchar(40),
    iSize varchar(40),
    primary key(menuType, itemName),
    constraint itemDetail_menu_fk foreign key(menuType) references menus(menuType),
    constraint itemDetail_menuItem_fk foreign key(itemName) references menuItems(itemName)
);

create table menuItems(
    itemName varchar(40) not null,
    spiciness varchar(40) not null,
    primary key(itemName)
);

create table categories(
    itemName varchar(40) not null,
    categoryType varchar(40) not null,
    primary key(itemName, categoryType),
    constraint categories_menuItem_fk foreign key(itemName) references menuItems(itemName)
);

create table itemOrder(
    itemName varchar(40) not null,
    menuType varchar(40) not null,
    orderID int not null,
    quantity int,
```

```
    primary key(itemName, menuType, orderID),
    constraint itemOrder_menuItem_fk foreign key(itemName, menuType) references
itemDetails(itemName, menuType),
    constraint itemOrder_orders_fk foreign key(orderID) references orders(orderID)
);


create table workCrews(
    crewNum int not null,
    primary key(crewNum)
);

create table shiftInstance(
    crewNum int not null,
    shiftID int not null,
    sDate DATE NOT NULL,
    PRIMARY KEY(crewNUm, shiftID,sDate),
    constraint shift_instance_workcrew_fk foreign key(crewNum) references
workCrews(crewNum),
    constraint shift_instance_shifts_fk foreign key(shiftID) references shifts(shiftID)
);

CREATE TABLE shifts(
    shiftID int not null,
    shiftType VARCHAR(40) NOT NULL,
    shiftDay VARCHAR(40) not null,
    primary key(shiftID)
);
```

Views

a. MenuItem_v – For each menu item, give it's spiciness, and all of the different costs for that item. If a given item is not on a particular menu, then report "N/A" for that particular item for that particular menu. Also, if an item only appears as a single serving portion, put in "N/A" into the report for the gallon, … prices.

```
create view MenuItem_v as
    select distinct spiciness, price, itemName, menuType
    from menuItems natural join itemOrder
    natural join itemDetails
    order by itemName;
```

spiciness	price	itemName	menuType
Hot	7	BBQ Pork	Children
Hot	11	Beef Chop Suey	Evening
Hot	9	Beef Chop Suey	Lunch
Hot	0	Beef Chop Suey	Sunday Brunch Buffet
Mild	10	Beef Chow Mein	Evening
Mild	8	Beef Chow Mein	Lunch
Mild	5	Bok Choy	Children
Mild	13	Broccoli Beef	Evening

7 19:48:38 select * from MenuItem_v LIMIT 0, 1000 — 67 row(s) returned

b. Customer_addresses_v – for each customer, indicate whether they are an individual or a corporate account, and display all of the information that we are managing for that customer.

```
Create view Customer_addresses_v As
select 'Private' as "Category", c.cID , cName as Customer
from customers c
inner join privateCustomers pc on c.cID = pc.cID
UNION
select 'Corporate' as "Category", c.cID as Customer, oficceAddress as Address
from customers c
inner join corporateCustomers cc on c.cID = cc.cID;
```

Category	cID	Customer
Private	1	Frankie English
Private	2	Gonzalo Duran
Private	3	Julio Mancia
Private	4	Alexis Jaime
Private	9	Matthew Maestro
Private	10	Johnny Flores
Private	11	Jacob David Rodriguez
Private	12	Kristin Contreras

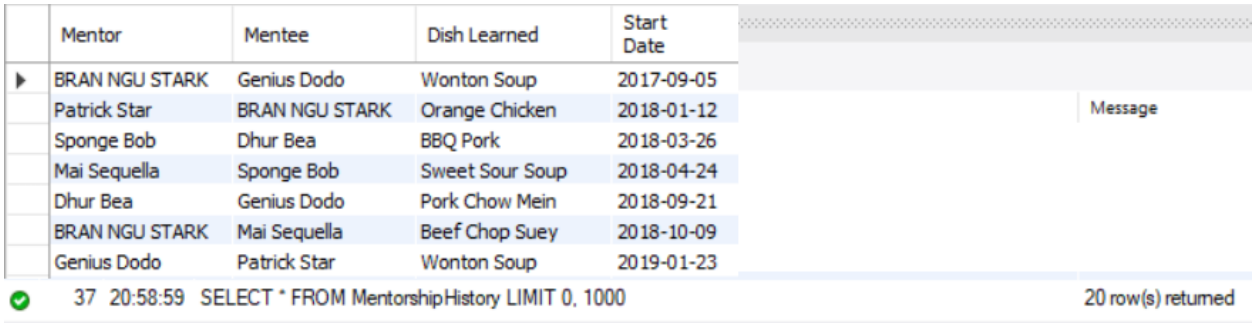
Message

10 19:51:49 Select * from Customer_addresses_v LIMIT 0, 1000 — 33 row(s) returned

c. Sous_mentor_v – reports all of the mentor/mentee relationships at Miming's, sorted by the name of the mentor, then the name of the mentee. Show the skill that the mentorship passes, as well as the start date.

```
CREATE
   ALGORITHM = UNDEFINED
   DEFINER = CURRENT_USER
   SQL SECURITY DEFINER
VIEW cecs323sec07og7.MentorshipHistory AS
   SELECT
      E.eName AS Mentor,
      F.eName AS Mentee,
      SCD.dish AS 'Dish Learned',
      SCD.startDate AS 'Start Date'
```

```
FROM
   ((cecs323sec07og7.mentorship SCD
   JOIN cecs323sec07og7.employees E ON ((E.eID = SCD.mentorID)))
   LEFT JOIN cecs323sec07og7.employees F ON ((F.eID = SCD.menteeID)))
ORDER BY SCD.startDate;
```



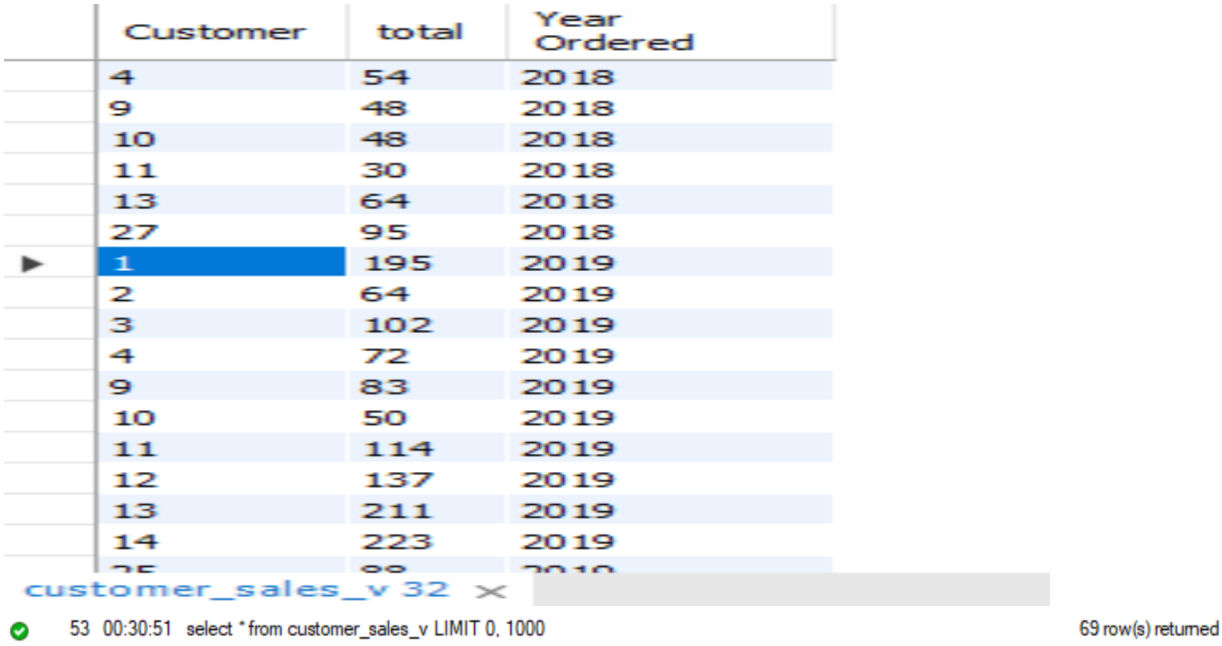
Mentor	Mentee	Dish Learned	Start Date
BRAN NGU STARK	Genius Dodo	Wonton Soup	2017-09-05
Patrick Star	BRAN NGU STARK	Orange Chicken	2018-01-12
Sponge Bob	Dhur Bea	BBQ Pork	2018-03-26
Mai Sequella	Sponge Bob	Sweet Sour Soup	2018-04-24
Dhur Bea	Genius Dodo	Pork Chow Mein	2018-09-21
BRAN NGU STARK	Mai Sequella	Beef Chop Suey	2018-10-09
Genius Dodo	Patrick Star	Wonton Soup	2019-01-23

Message

37 20:58:59 SELECT * FROM MentorshipHistory LIMIT 0, 1000 — 20 row(s) returned

d. Customer_Sales_v – On a year by year basis, show how much each customer has spent at Miming's.

```
create view customer_sales_v as
SELECT  PC.cID AS Customer, SUM(ID.price * IO.quantity) AS "total", YEAR(O.dateOrder)
AS 'Year Ordered'
FROM privateCustomers PC natural join orders O natural join itemOrder IO natural join
itemDetails ID
GROUP BY Year(O.dateOrder), PC.cID
UNION
SELECT CC.cID AS Customer, SUM(ID.price * IO.quantity) AS 'Amount
Spent',YEAR(O.dateOrder) AS 'Year Ordered'
FROM corporateCustomers CC natural join orders O natural join itemOrder IO natural
join itemDetails ID
GROUP BY Year(O.dateOrder), CC.cID;
```



Customer	total	Year Ordered
4	54	2018
9	48	2018
10	48	2018
11	30	2018
13	64	2018
27	95	2018
1	195	2019
2	64	2019
3	102	2019
4	72	2019
9	83	2019
10	50	2019
11	114	2019
12	137	2019
13	211	2019
14	223	2019

e. Customer_Value_v – List each customer and the total $ amount of their orders for the past year, in order of the value of customer orders, from highest to the lowest.

```
Create view Customer_Value_V as
SELECT PC.cID AS 'Customer', SUM(ID.price *IO.quantity) AS 'Total',
YEAR(O.dateOrder) AS 'Year Ordered'
FROM privateCustomers PC natural join orders O natural join itemOrder IO natural join
itemDetails ID
WHERE O.dateOrder >= NOW() - interval 1 year
GROUP BY PC.cID
UNION
SELECT CC.cID AS 'Customer', SUM(ID.price * IO.quantity) AS
'Total',YEAR(O.dateOrder) AS 'Year Ordered'
FROM corporateCustomers CC natural join orders O natural join itemOrder IO natural
join itemDetails ID
WHERE O.dateOrder >= now() - interval 1 year
GROUP BY CC.cID
ORDER by Total desc;
```

	Customer	Total	Year Ordered
▶	14	223	2019
	13	211	2019
	1	195	2019
	23	161	2019
	6	156	2019
	12	137	2019
	11	114	2019
	16	106	2019
	3	102	2019
	17	101	2019
	24	100	2019
	21	100	2019
	7	97	2019
	18	95	2019
	25	88	2019
	9	83	2019
	5	76	2019
	4	72	2019
	19	69	2019
	28	69	2019
	15	68	2019
	29	65	2019

Customer_Value_V 22 ×

Queries

a. List the customers. For each customer, indicate which category he or she fall into, and his or her contact information. If you have more than one independent categorization of customers, please indicate which category the customer falls into for all the categorizations.

```
select 'private customer' as "category", c.cID, c.cName as "name" , p.email
as "contact info",  p.snail_mailAddress as "address"
from customers c inner join privateCustomers p on c.cID = p.cID
union
select 'corporate customer' as "category",c.cID, c.cName, cp.contactInfo,
cp.oficceAddress
from customers c inner join corporateCustomers cp on c.cID = cp.cID;
```

#	category	cID	name	contact info	address
1	private customer	1	Frankie English	fenglish@gmail.com	8010 E Tarma st.
2	private customer	2	Gonzalo Duran	gduran@yahoo.com	408 high dr.
3	private customer	3	Julio Mancia	jmancia@yahoo.com	225 michelle st.
4	private customer	4	Alexis Jaime	ajaime@gmail.com	999 dahlia st.
5	corporate customer	5	Whole Foods	888-777-4444	2345 raddiator st
6	corporate customer	6	Telenova Films	951-555-3434	2345 guatemala ave
7	corporate customer	7	Cal State Long Beach	theBeach@yahoo.com	1243 university road
8	corporate customer	8	Williamson LLC	964-321-6743	2356 chair st
9	private customer	9	Matthew Maestro	mmaestro@gmail.com	4004 manager dr.
10	private customer	10	Johnny Flores	jflores@gmail.com	8873 jill st.
11	private customer	11	Jacob David Rodriguez	babyjacob@gmail.com	9334 temecula st.
12	private customer	12	Kristin Contreras	kcontreras@gmail.com	5263 death dr.
13	private customer	13	Adam Heath	harkins4life@gmail.com	4444 harkins st.
14	private customer	14	David Lepore	actorgamermodel@gmail.com	4467 terror ave.
15	corporate customer	15	Tarantino Films	themaster@gmail.com	435 violence lane
16	corporate customer	16	Depeche Mode Studios	951-833-6054	888 dm dr
17	corporate customer	17	Raytheon	johnong@gmail.com	6548 pepper st.
18	corporate customer	18	Northrop Grunman	515-506-4122	7894 Garo st.
19	corporate customer	19	Boeing	631-680-6725	4684 Neptune ave.
20	corporate customer	20	Amazon	866-155-7820	7894 Saturn st.
21	corporate customer	21	Harkins Theatres	aviles@gmail.com	600 Los Cerritos Center
22	corporate customer	22	Gonzalo Gains	555-555-5555	5562 earnthatmoney dr
23	corporate customer	23	Maestro Accounting	machinegun@gmail.com	678 machinegun lane
24	corporate customer	24	Microsoft	playstationsucks@yahoo.com	1234 xbox dr
25	private customer	25	Emely Torres	mylove@gamil.com	1256 OneDay ave.
26	private customer	26	Ryan Harrison	ryanharrison0424@gmail.com	17233 Balfern ave.
27	private customer	27	Dale Earnhardt Jr	letsgoracing@yahoo.com	88 car st.
28	private customer	28	Martin Gore	goreyDM@aol.com	1515 thethingsyousaid st.
29	private customer	29	Dahliha Jaime	theblackdahlia@aol.com	783 alexis dr.
30	corporate customer	30	Wes Craven Productions	scream4urlife@gmail.com	1236 horror ave
31	corporate customer	31	Pauleth Aviles	aviles@gmail.com	456 Lakewood Center

b. List the top three customers in terms of their net spending for the past **two** years, and the total that they have spent in that period.

```
select c.cID as "customer id", c.cName as "name", o.orderID, o.dateOrder,
sum(io.quantity * id.price) as "total"
from customers c natural join orders o natural join itemOrder io natural join
itemDetails id
where o.dateOrder >= now() - interval 1 year
group  by c.cID
order by total desc
LIMIT 3;
```

	customer id	name	orderID	dateOrder	total
▶	14	David Lepore	14	2019-02-17	223
	13	Adam Heath	13	2019-02-16	211
	1	Frankie English	1	2019-01-12	195

c. Find all the sous chefs who have three or more menu items that they can prepare. For each sous chef, list their name, the number of menu items that they can prepare, and each of the menu items. You can use group_concat to get all of a given sous chef's data on one row or print out one row per sous chef per menu item.

```
select e.eName as "sous chef", count(m.meal) "number of meals",
 group_concat(m.meal) as "meals"
from employees e
natural join sousChefs s
natural join meals m
group by (e.eID)
having count(m.meal) >= 3;
```

	sous chef	number of meals	meals
	Genius Dodo	9	Pork Chow Mein,Fried Rice,Spicy Snails,Nut Chic...
	Mai Sequella	8	Seafood Chow Mein,Chef Special Chop Suey,S...
	Dhur Bea	8	Secret Eggplant,Chicken Chop Suey,Steamed D...
▶	BRAN NGU STARK	8	Orange Chicken,Chef Special Chop Suey,Shark ...
	Sponge Bob	7	Sweet Sour Soup,Chicken Chop Suey,Youtiao,S...
	Patrick Star	9	Shrimp Dish,Miso Soup,Vegetable Chow Mein,Po...

d. Find all the sous chefs who have three or more menu items in common.
   a. Please give the name of each of the two sous chefs sharing three or more menu items.
   b. Please make sure that any given pair of sous chefs only shows up once.

```
select a.eName as "employee 1 name",a.eID as "employee 1 ID",
b.eName AS "employee 2 name",b.eID as "employee 2 ID", count(b.eID) as
"number of menu Items"
from (select * from meals m1 natural join employees e1)  as a
inner join
(select * from meals m2 natural join employees e2) as b
```

```
on a.meal = b.meal
where a.eID < b.eID
group by a.eID, b.eID
having count(b.eId);
```

	employee 1 name	employee 1 ID	employee 2 name	employee 2 ID	number of menu Items
▶	Genius Dodo	6	Mai Sequella	9	3
	Genius Dodo	6	Dhur Bea	10	3
	Genius Dodo	6	BRAN NGU STARK	26	2
	Genius Dodo	6	Sponge Bob	27	1
	Mai Sequella	9	Dhur Bea	10	1
	Mai Sequella	9	BRAN NGU STARK	26	3
	Mai Sequella	9	Sponge Bob	27	1
	Dhur Bea	10	Sponge Bob	27	2
	Dhur Bea	10	Patrick Star	28	1
	BRAN NGU STARK	26	Sponge Bob	27	2
	BRAN NGU STARK	26	Patrick Star	28	1

E. Find the three menu items most often ordered from the Children's menu and order them from most frequently ordered to least frequently ordered.

```
select ml.itemName, count(ml.itemName) as "times ordered"
from menuItems ml
natural join itemDetails id
natural join itemOrder io
where io.menuType = 'Children'
group by(ml.itemName)
order by "times ordered" desc
limit 3;
```

	itemName	times ordered
▶	BBQ Pork	3
	Bok Choy	2
	Scallion Pancake	2

6f. List all the menu items, the shift in which the menu item was ordered, and the sous chef on duty at the time, when the sous chef was not an expert in that menu item.

```
select distinct itemName,shiftDay,shiftType,eName as sousChef
 from employees
natural join meals
natural join menuItems
natural join itemOrder
natural join shiftInstance
natural join shifts
natural join mentorship
where shiftDay < endDate
order by shiftDay;
```

itemName	shiftDay	shiftType	sousChef
BBQ Pork	friday	morning	Sponge Bob
Chef Special Chop Suey	friday	morning	Genius Dodo
Tomato beef	friday	morning	Sponge Bob
Vegetables Chop Suey	friday	morning	Mai Sequella
Beef Chop Suey	friday	morning	Mai Sequella
Shark Fin Soup	friday	morning	Patrick Star
Shrimp Dish	friday	morning	Dhur Bea
Scallion Pancake	friday	morning	BRAN NGU STARK
Mongolian Chicken	friday	morning	Sponge Bob
Seafood Chop Suey	friday	morning	Genius Dodo
Chicken Egg Foo Young	friday	morning	Patrick Star
Egg Drop Soup	friday	morning	Dhur Bea
Cha Siu Bao	friday	morning	BRAN NGU STARK
Beef Chow Mein	friday	night	Sponge Bob
Chef Special Egg Foo Y...	friday	night	Genius Dodo
Vegetables Chow Mein	friday	night	Sponge Bob
Vegetables Egg Foo Yo...	friday	night	Mai Sequella
Bok Choy	friday	night	Mai Sequella
Spicy Snails	friday	night	Patrick Star
Steamed Dumplings	friday	night	Dhur Bea
Seafood Chow Mein	friday	night	BRAN NGU STARK

7g. List the customers, sorted by the amount of Miming's Money that they have, from largest to smallest.

```
select cName as Customer, cMoney as MimisMoney from privateCustomers
natural join customers
group by customer
UNION
select cName as Customer, cMoney as MimisMoney from corporateCustomers
```

natural join customers
group by Customer
order by MimisMoney desc;

Customer	MimisMoney
Emely Torres	1500
Gonzalo Gains	999
Harkins Theatres	750
Wes Craven Productions	666
Matt Harrison	555
Frankie Maestro	464
Williamson LLC	407
Tarantino Films	389
Depeche Mode Studios	342
Amazon	190
Johnny Flores	123
Matthew Maestro	90
Northrop Grunman	89
Dale Earnhardt Jr	88
Telenova Films	83
Pauleth Aviles	78
Frankie English	75
Adam Heath	67
Gonzalo Duran	54
Whole Foods	54
Boeing	45
Maestro Accounting	43
Jacob David Rodriguez	34
Alexis Jaime	30

8h. List the customers and the total that they have spent at Miming's ever, in descending order by the amount that they have spent.

select cName as Customer, sum(id.price * quantity)as Total from
privateCustomers
natural join customers
natural join orders
natural join itemOrder
natural join itemDetails id
group by Customer
UNION
select cName as Customer, sum(id.price * quantity) as Total from
corporateCustomers
natural join customers
natural join orders
natural join itemOrder
natural join itemDetails id

```
group by Customer
order by Total desc;
```

Wes Craven Productions	611
Johnny Flores	538
Telenova Films	463
Gonzalo Duran	347
Matthew Maestro	342
Microsoft	331
Gonzalo Gains	307
Adam Heath	275
David Lepore	266
Maestro Accounting	248
Dahliha Jaime	245
Kristin Contreras	231
Frankie English	195
Alexis Jaime	182
Julio Mancia	175
Cal State Long Beach	147

9i. Report on the customers at Miming's by the number of times that they come in by month and order the report from most frequent to the least frequent.

```
select cName as Customer, month(o.dateOrder) as "Month", year(o.dateOrder)
as "Year", count(month(o.dateOrder)) as Visits
from privateCustomers
natural join customers
natural join orders o
group by cName, month(o.dateOrder), year(o.dateOrder)
UNION
select cName as Customer, month(o.dateOrder) as "Month", year(o.dateOrder)
as "Year", count(month(o.dateOrder)) as Visits
from corporateCustomers
natural join customers
natural join orders o
group by cName, month(o.dateOrder), year(o.dateOrder)
order by Visits desc;
```

▶	Frankie English	4	2019	2
	Boeing	2	2019	1
	Telenova Films	6	2018	1
	Jacob David Rodriguez	2	2019	1
	Matthew Maestro	10	2018	1
	Tarantino Films	4	2018	1
	David Lepore	11	2017	1
	Martin Gore	3	2019	1
	Microsoft	8	2016	1
	Dahliha Jaime	3	2019	1
	Gonzalo Gains	10	2016	1
	Williamson LLC	4	2019	1
	Adam Heath	2	2019	1
	Julio Mancia	3	2019	1
	Cal State Long Beach	4	2019	1
	Wes Craven Productions	5	2018	1
	Johnny Flores	3	2016	1
	Amazon	2	2019	1

10j. List the three customers who have spent the most at Miming's over the past year. Order by the amount that they spent, from largest to smallest.

```
select cName as "Customer", sum(ID.price * IO.quantity) as "Amount Spent Over
Past Year" from privateCustomers PC
natural join customers
inner join orders O on PC.cID = O.cID
inner join itemOrder IO on IO.orderID = O.orderID
inner join itemDetails ID on ID.itemName = IO.itemName
group by cName
where (year(O.dateOrder) >= year(current_timestamp) -1)
UNION select cName as "Customer", sum(ID.price * IO.quantity) as "Amount Spent
Over Past Year" from corporateCustomers CC
natural join customers
inner join orders O on CC.cID = O.cID
inner join itemOrder IO on IO.orderID = O.orderID
inner join itemDetails ID on ID.itemName = IO.itemName
where (year(O.dateOrder) >= year(current_timestamp) -1)
group by cName
order by "Amount Spent Over Past Year" desc limit 3;
```

	Customer	Amount Spent Over Past Year
▶	Adam Heath	602
	Alexis Jaime	245
	Dahliha Jaime	149