CAKES BY MARY BELL

Claudia Rojas

DATABASE SYSTEMS

CMPT 308

Fall 2016

Table of Contents

Executive Summary

Overview

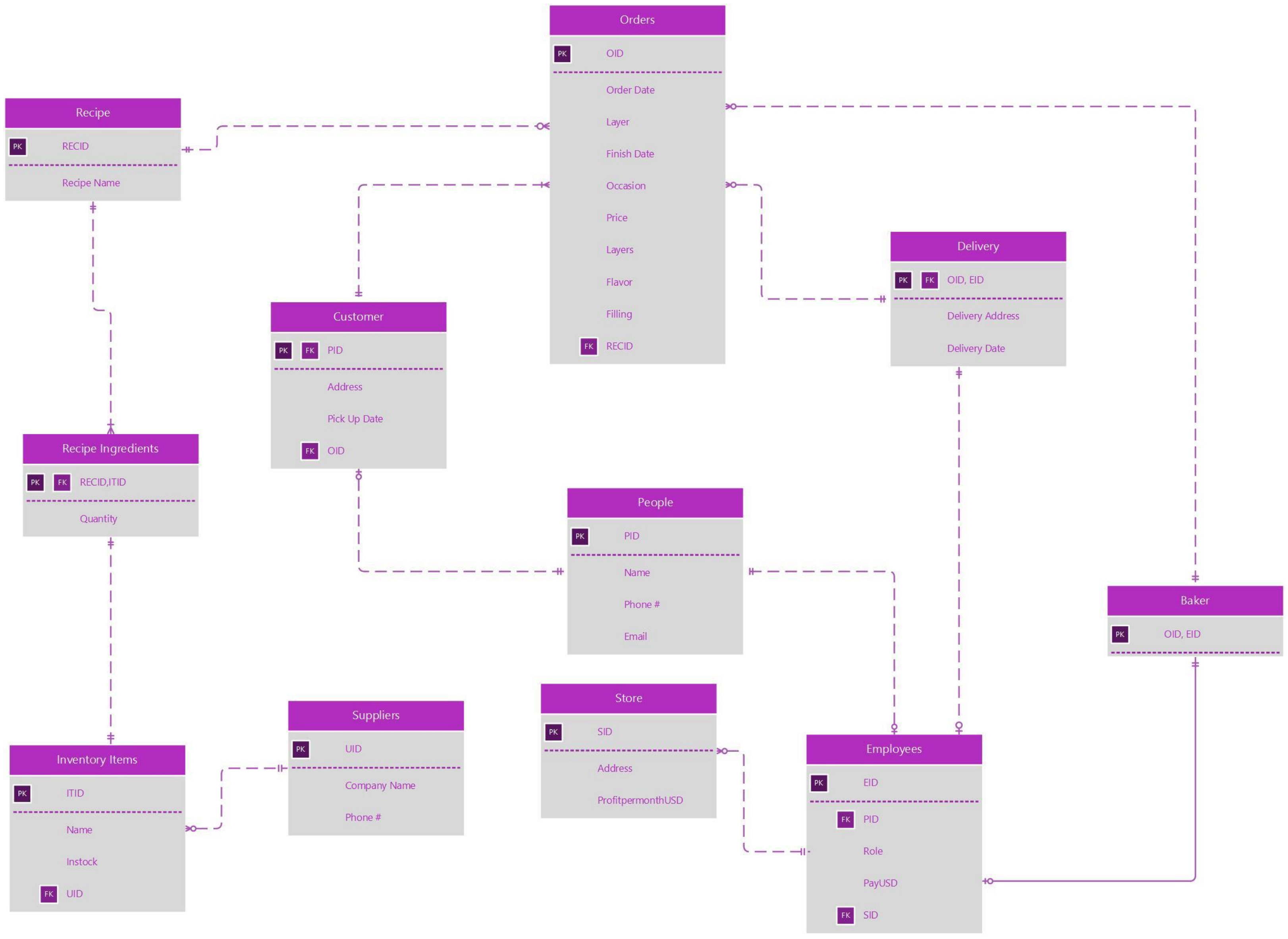
The customer, Cakes by Mary Bell, is an up-and-coming family business that is ready to open in the next few months. They have requested to create a database for their business to keep track of their employees, customers, and orders. The customer has also stated that they would like to keep an "open mind" to the database to allow for expansion. They would like the database to allow for the implementation of many stores in the future

Objectives

This document describes the database system that was created and filled with data to mimic how the database for the store will look. The purpose of the database is to show how the store will be run according to the database system. The database stores all the information of customers, employees, orders, and an inventory of items currently within one store.

The document will provide a detail look into each section within the database and will describe the implementation of the database. The document will include: tables and their functions, views, triggers, stored procedures, and security.

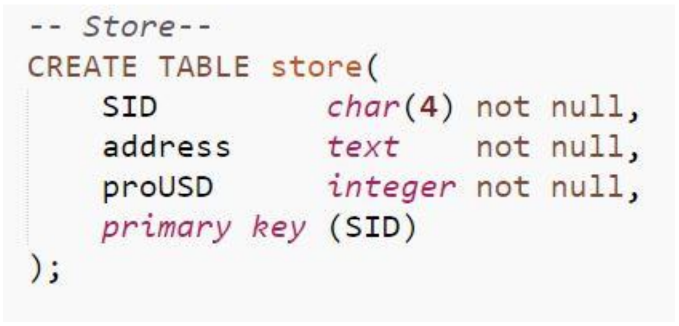
Entity Relationship Diagram



Tables

Store

This table stores the information of different stores and allows for the inclusion of different stores in the database. The store includes information about its location and the profits it earns per month.

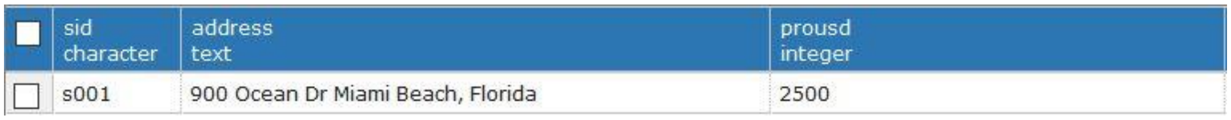


```
-- Store--
CREATE TABLE store(
    SID          char(4) not null,
    address      text    not null,
    proUSD       integer not null,
    primary key (SID)
);
```

Functional Dependencies

SID $\longrightarrow$ address, ProUSD

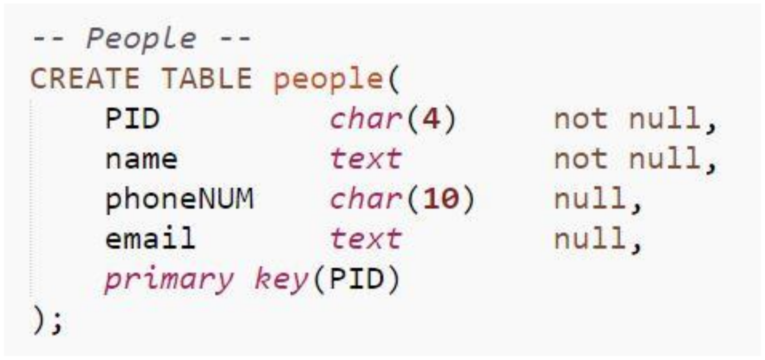
Sample Data



sid character	address text	prousd integer
s001	900 Ocean Dr Miami Beach, Florida	2500

People

This table keeps track of the people who are either customers of the store or are employees working in the store.

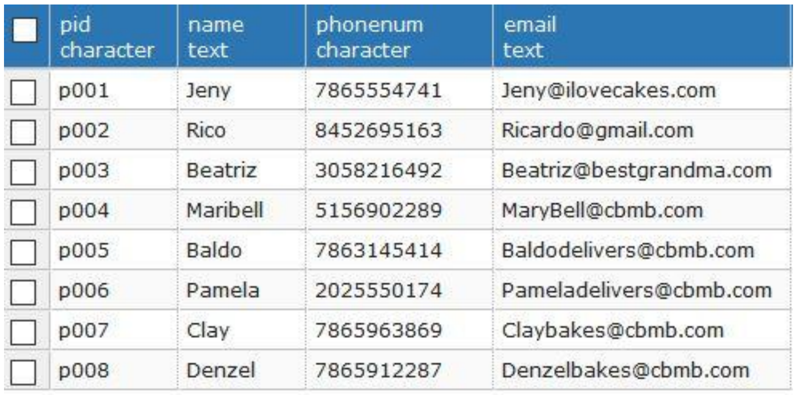


```
-- People --
CREATE TABLE people(
    PID           char(4)      not null,
    name          text         not null,
    phoneNUM      char(10)     null,
    email         text         null,
    primary key(PID)
);
```

Functional Dependencies

PID $\longrightarrow$ name, phoneNUM, email

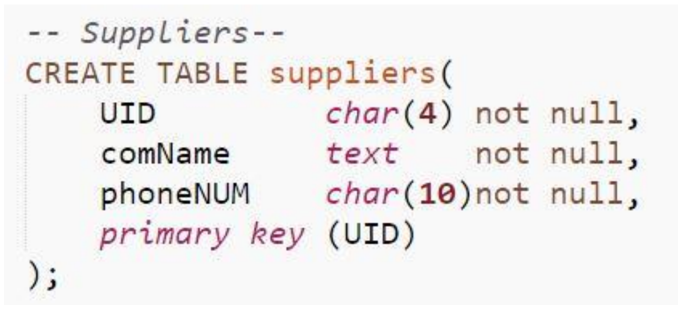
Sample Data



pid character	name text	phonenum character	email text
p001	Jeny	7865554741	Jeny@ilovecakes.com
p002	Rico	8452695163	Ricardo@gmail.com
p003	Beatriz	3058216492	Beatriz@bestgrandma.com
p004	Maribell	5156902289	MaryBell@cbmb.com
p005	Baldo	7863145414	Baldodelivers@cbmb.com
p006	Pamela	2025550174	Pameladelivers@cbmb.com
p007	Clay	7865963869	Claybakes@cbmb.com
p008	Denzel	7865912287	Denzelbakes@cbmb.com

Suppliers

This table contains the information of the different suppliers that supply the store with inventory items to keep it up and running.

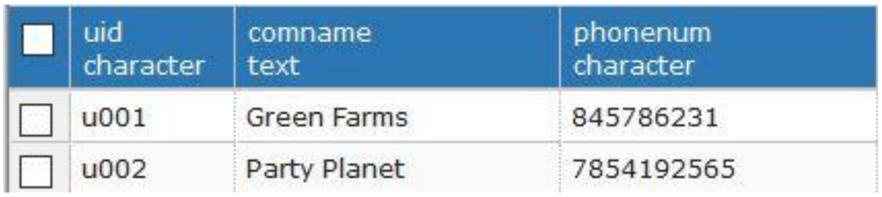


```
-- Suppliers--
CREATE TABLE suppliers(
    UID          char(4) not null,
    comName      text    not null,
    phoneNUM     char(10)not null,
    primary key (UID)
);
```

Functional Dependencies

UID $\longrightarrow$ comName, PhoneNUM

Sample Data



uid character	comname text	phonenum character
u001	Green Farms	845786231
u002	Party Planet	7854192565

Inventory

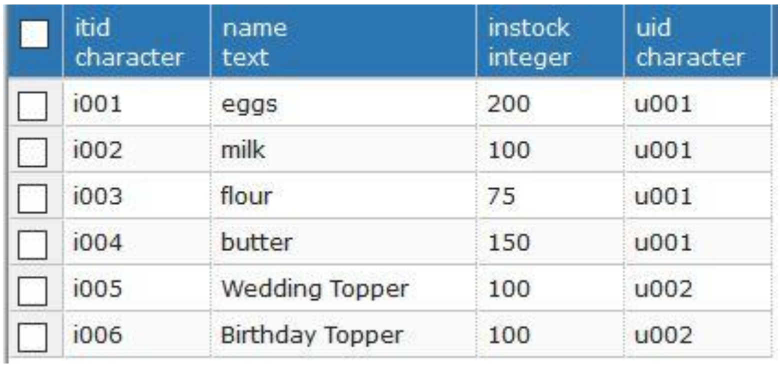
This table contains the inventory information for the items that are used in certain recipes for different cake orders.

```
-- Inventory Items--
CREATE TABLE inventory(
    ITID        char(4) not null,
    name        text    not null,
    instock     integer not null,
    UID         char(4) not null references suppliers(UID),
    primary key (ITID)
);
```

Functional Dependencies

ITID $\longrightarrow$ name, instock, UID

Sample Data



itid character	name text	instock integer	uid character
i001	eggs	200	u001
i002	milk	100	u001
i003	flour	75	u001
i004	butter	150	u001
i005	Wedding Topper	100	u002
i006	Birthday Topper	100	u002

Ingredients

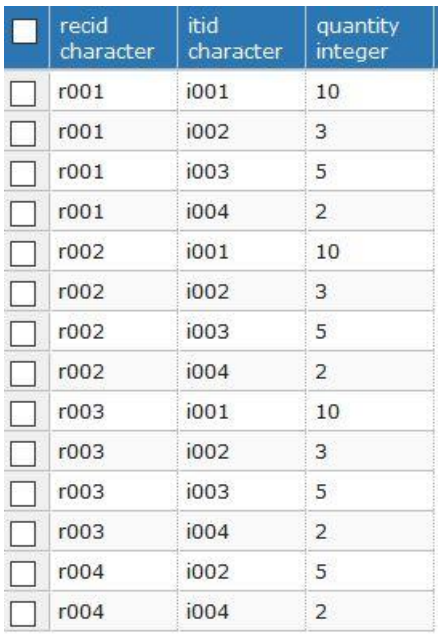
This table is a cross reference of the different ingredients needed for different recipes in an order.

```sql
-- Recipe Ingredients--
CREATE TABLE recipeIngredients(
    RECID        char(4) not null references recipe(RECID),
    ITID         char(4) not null references inventory(ITID),
    quantity     integer not null,
    primary key (RECID,ITID)
);
```

Functional Dependencies

RECID, ITID $\longrightarrow$ quantity

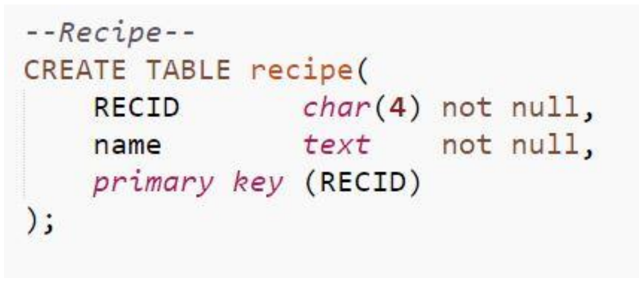
Sample Data



☐	recid character	itid character	quantity integer
☐	r001	i001	10
☐	r001	i002	3
☐	r001	i003	5
☐	r001	i004	2
☐	r002	i001	10
☐	r002	i002	3
☐	r002	i003	5
☐	r002	i004	2
☐	r003	i001	10
☐	r003	i002	3
☐	r003	i003	5
☐	r003	i004	2
☐	r004	i002	5
☐	r004	i004	2

Recipe

This table contains the name and ID of the recipes used in the cake orders.

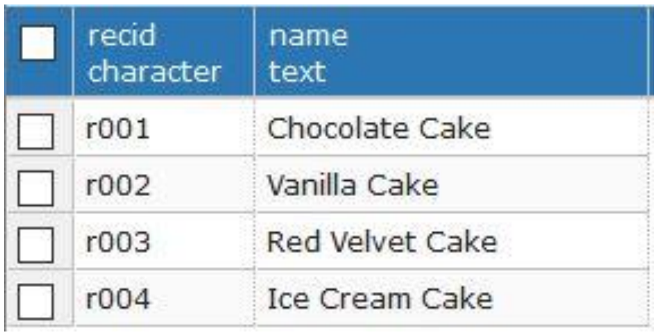


```
--Recipe--
CREATE TABLE recipe(
    RECID          char(4) not null,
    name           text    not null,
    primary key (RECID)
);
```

Functional Dependencies

RECID $\longrightarrow$ name

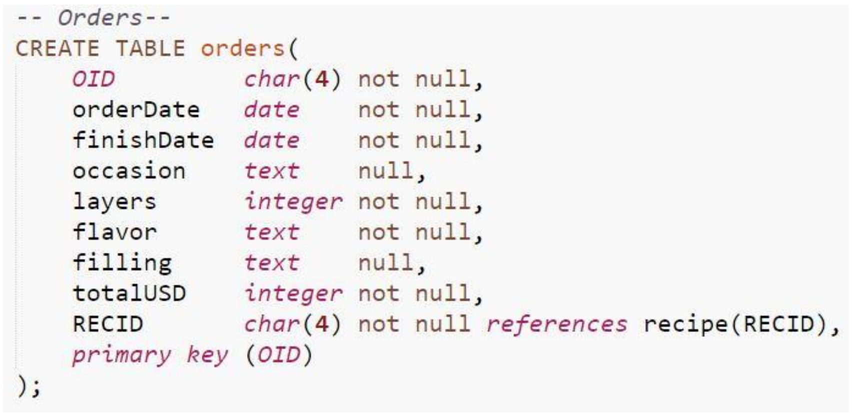
Sample Data



☐	recid character	name text
☐	r001	Chocolate Cake
☐	r002	Vanilla Cake
☐	r003	Red Velvet Cake
☐	r004	Ice Cream Cake

Orders

This table contains all the information that is needed within an order. This table is important because it is keeps tracks of all the orders within the store.

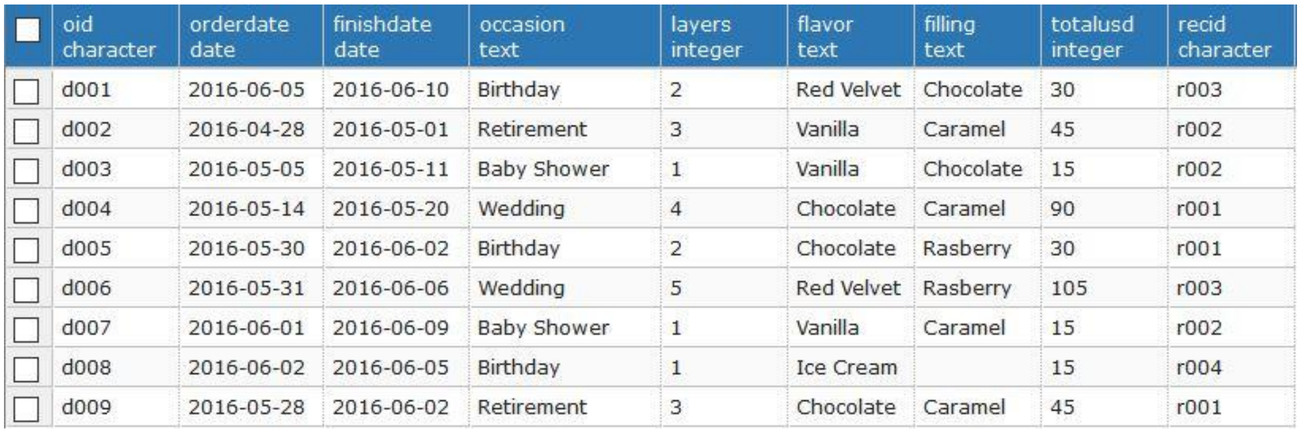


```
-- Orders--
CREATE TABLE orders(
    OID          char(4) not null,
    orderDate    date    not null,
    finishDate   date    not null,
    occasion     text    null,
    layers       integer not null,
    flavor       text    not null,
    filling      text    null,
    totalUSD     integer not null,
    RECID        char(4) not null references recipe(RECID),
    primary key (OID)
);
```

Functional Dependencies

OID $\longrightarrow$ orderDate, finishDate, occasion, layers, flavor, filling, totalUSD, RECID

Sample Data



☐	oid character	orderdate date	finishdate date	occasion text	layers integer	flavor text	filling text	totalusd integer	recid character
☐	d001	2016-06-05	2016-06-10	Birthday	2	Red Velvet	Chocolate	30	r003
☐	d002	2016-04-28	2016-05-01	Retirement	3	Vanilla	Caramel	45	r002
☐	d003	2016-05-05	2016-05-11	Baby Shower	1	Vanilla	Chocolate	15	r002
☐	d004	2016-05-14	2016-05-20	Wedding	4	Chocolate	Caramel	90	r001
☐	d005	2016-05-30	2016-06-02	Birthday	2	Chocolate	Rasberry	30	r001
☐	d006	2016-05-31	2016-06-06	Wedding	5	Red Velvet	Rasberry	105	r003
☐	d007	2016-06-01	2016-06-09	Baby Shower	1	Vanilla	Caramel	15	r002
☐	d008	2016-06-02	2016-06-05	Birthday	1	Ice Cream		15	r004
☐	d009	2016-05-28	2016-06-02	Retirement	3	Chocolate	Caramel	45	r001

Employees

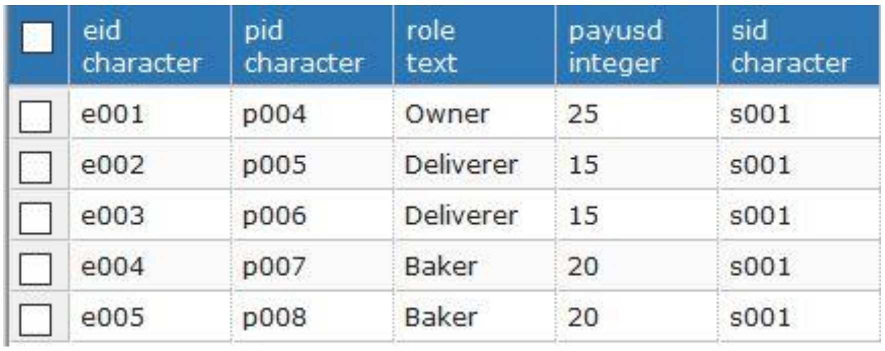
This table contains the information of each employee that works in a certain store. The employee table contains only half of the employee information because the people table contains the email and phone numbers of the employees.

```
-- Employees --
CREATE TABLE employees(
    EID         char(4) not null,
    PID         char(4) not null references people(PID),
    role        text    not null,
    payUSD      integer not null,
    SID         char(4) not null references store(SID),
    primary key (EID)
);
```

Functional Dependencies

EID $\rightarrow$ PID, role, payUSD, SID

Sample Data



☐	eid character	pid character	role text	payusd integer	sid character
☐	e001	p004	Owner	25	s001
☐	e002	p005	Deliverer	15	s001
☐	e003	p006	Deliverer	15	s001
☐	e004	p007	Baker	20	s001
☐	e005	p008	Baker	20	s001

Baker

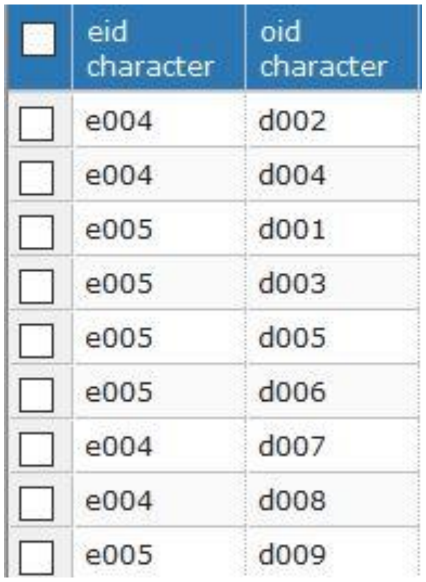
This table is a combination of the employee ID and the order ID which allows for different bakers to work on different cakes. It also allows for bakers to work on the same cakes at once.

```sql
-- Baker--
CREATE TABLE baker(
    EID          char(4) not null references employees(EID),
    OID          char(4) not null references orders(OID),
    primary key (OID, EID)
);
```

Functional Dependencies

EID, OID $\longrightarrow$

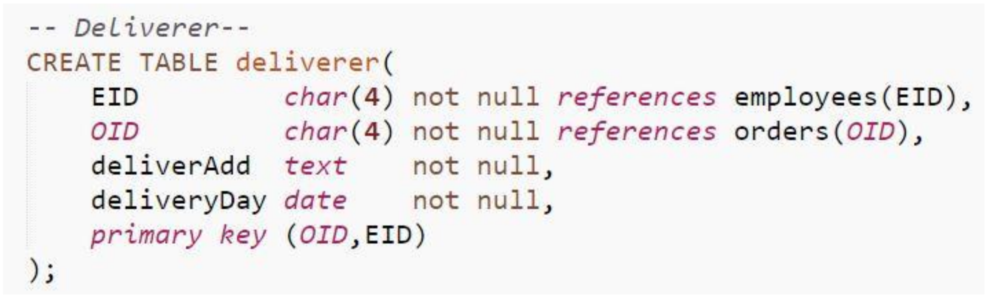
Sample Data



☐	eid character	oid character
☐	e004	d002
☐	e004	d004
☐	e005	d001
☐	e005	d003
☐	e005	d005
☐	e005	d006
☐	e004	d007
☐	e004	d008
☐	e005	d009

Deliverer

This table contains the information needed for a cake delivery. The primary key for the deliverer table is a composite of the employee ID and orders ID. Like the bakers table it also allows for multiple deliverers to go out for the same delivery if it is needed.

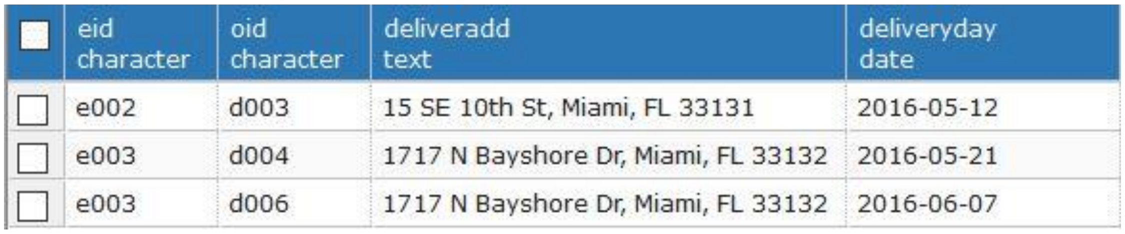


```
-- Deliverer--
CREATE TABLE deliverer(
    EID         char(4) not null references employees(EID),
    OID         char(4) not null references orders(OID),
    deliverAdd  text    not null,
    deliveryDay date    not null,
    primary key (OID,EID)
);
```

Functional Dependencies

EID, OID $\longrightarrow$ deliverAdd, deliveryDay

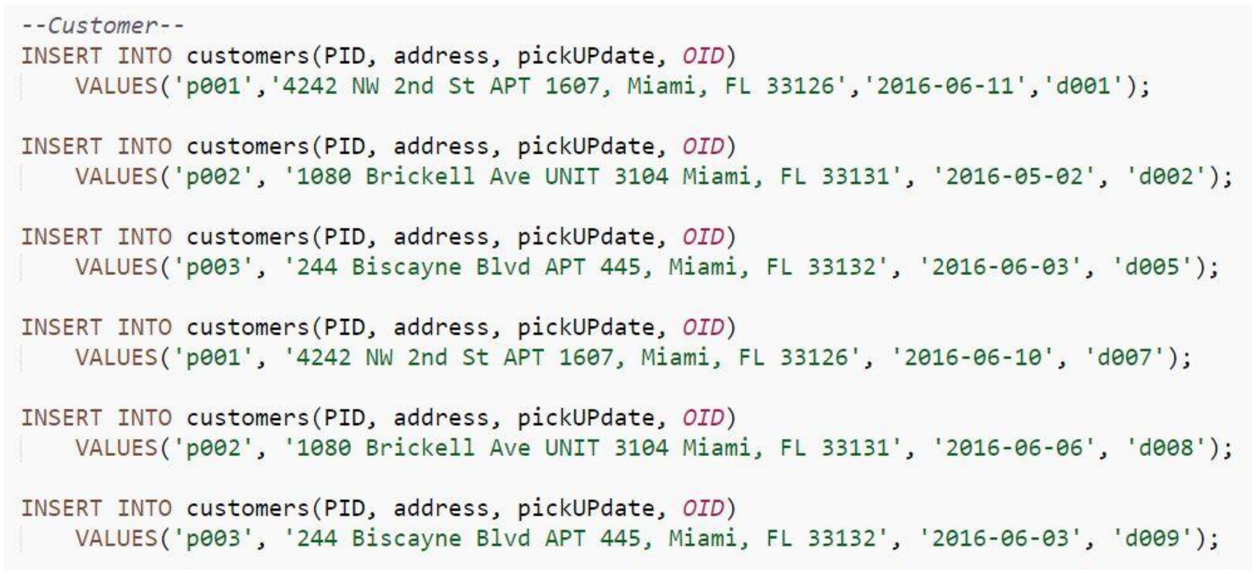
Sample Data



☐	eid character	oid character	deliveradd text	deliveryday date
☐	e002	d003	15 SE 10th St, Miami, FL 33131	2016-05-12
☐	e003	d004	1717 N Bayshore Dr, Miami, FL 33132	2016-05-21
☐	e003	d006	1717 N Bayshore Dr, Miami, FL 33132	2016-06-07

Customers

This table contains the PID and other information for the people who are customers of a store.

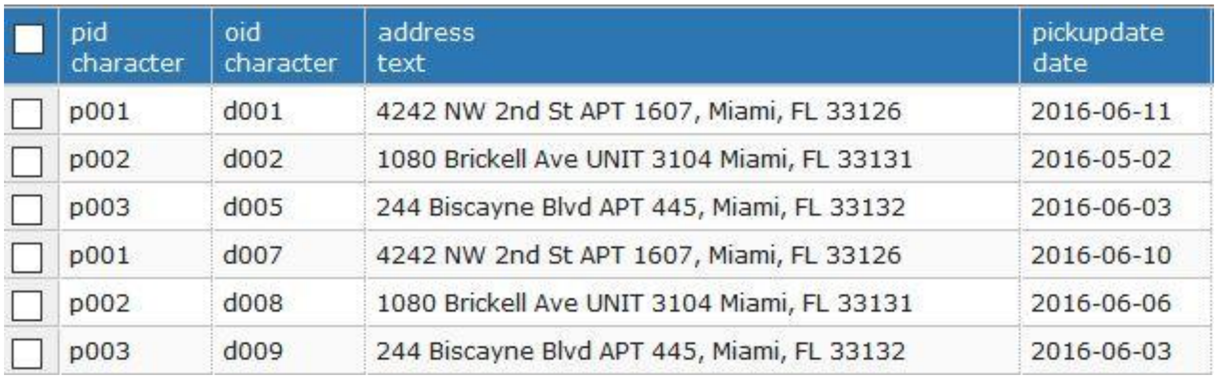


```
--Customer--
INSERT INTO customers(PID, address, pickUPdate, OID)
    VALUES('p001','4242 NW 2nd St APT 1607, Miami, FL 33126','2016-06-11','d001');

INSERT INTO customers(PID, address, pickUPdate, OID)
    VALUES('p002', '1080 Brickell Ave UNIT 3104 Miami, FL 33131', '2016-05-02', 'd002');

INSERT INTO customers(PID, address, pickUPdate, OID)
    VALUES('p003', '244 Biscayne Blvd APT 445, Miami, FL 33132', '2016-06-03', 'd005');

INSERT INTO customers(PID, address, pickUPdate, OID)
    VALUES('p001', '4242 NW 2nd St APT 1607, Miami, FL 33126', '2016-06-10', 'd007');

INSERT INTO customers(PID, address, pickUPdate, OID)
    VALUES('p002', '1080 Brickell Ave UNIT 3104 Miami, FL 33131', '2016-06-06', 'd008');

INSERT INTO customers(PID, address, pickUPdate, OID)
    VALUES('p003', '244 Biscayne Blvd APT 445, Miami, FL 33132', '2016-06-03', 'd009');
```

Functional Dependencies

PID $\rightarrow$ address, pickUPdate, OID

Sample Data



	pid character	oid character	address text	pickupdate date
☐	p001	d001	4242 NW 2nd St APT 1607, Miami, FL 33126	2016-06-11
☐	p002	d002	1080 Brickell Ave UNIT 3104 Miami, FL 33131	2016-05-02
☐	p003	d005	244 Biscayne Blvd APT 445, Miami, FL 33132	2016-06-03
☐	p001	d007	4242 NW 2nd St APT 1607, Miami, FL 33126	2016-06-10
☐	p002	d008	1080 Brickell Ave UNIT 3104 Miami, FL 33131	2016-06-06
☐	p003	d009	244 Biscayne Blvd APT 445, Miami, FL 33132	2016-06-03

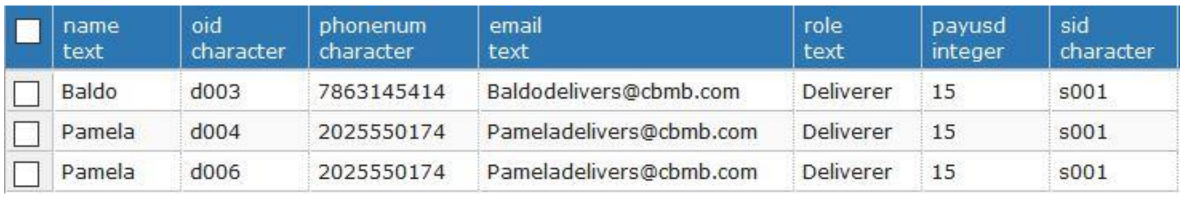
Views

DelivererJobs

This table shows the name, phone number, email address, role, pay in US dollars, and store id of a deliverer dependent on the multiple orders they have done for Cakes by Mary Bell.

```
CREATE VIEW DelivererJobs
AS
SELECT name,oid, phoneNum, email, role, payUSD, sid
FROM deliverer d, employees e, people p
WHERE d.eid = e.eid
and e.pid = p.pid
```

Sample Data



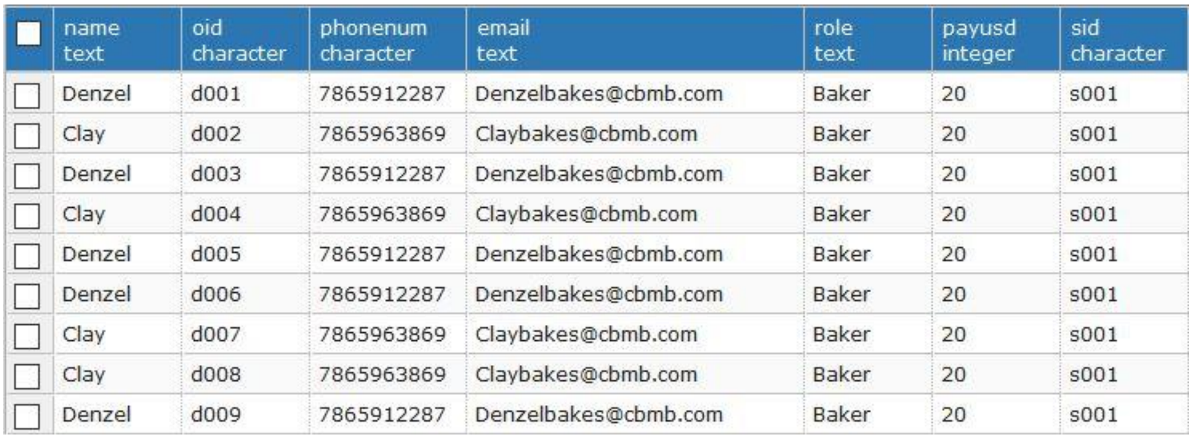
name text	oid character	phonenum character	email text	role text	payusd integer	sid character
Baldo	d003	7863145414	Baldodelivers@cbmb.com	Deliverer	15	s001
Pamela	d004	2025550174	Pameladelivers@cbmb.com	Deliverer	15	s001
Pamela	d006	2025550174	Pameladelivers@cbmb.com	Deliverer	15	s001

BakerJobs

Like DelivererJob, BakerJobs shows all the orders done by a baker including details about their phonenumber, email, role, pay in US dollars, and store ID.

```
CREATE VIEW BakerJobs
AS
SELECT name,oid, phoneNum, email, role, payUSD, sid
FROM baker b, employees e, people p
WHERE b.eid = e.eid
AND e.pid = p.pid
ORDER BY oid asc;
```

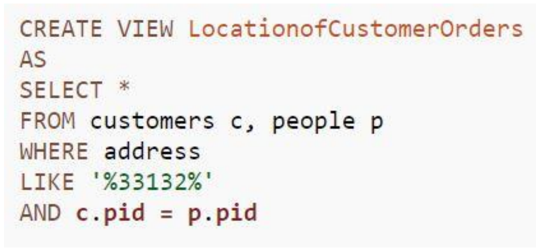
Sample Data



	name	oid	phonenum	email	role	payusd	sid
☐	text	character	character	text	text	integer	character
☐	Denzel	d001	7865912287	Denzelbakes@cbmb.com	Baker	20	s001
☐	Clay	d002	7865963869	Claybakes@cbmb.com	Baker	20	s001
☐	Denzel	d003	7865912287	Denzelbakes@cbmb.com	Baker	20	s001
☐	Clay	d004	7865963869	Claybakes@cbmb.com	Baker	20	s001
☐	Denzel	d005	7865912287	Denzelbakes@cbmb.com	Baker	20	s001
☐	Denzel	d006	7865912287	Denzelbakes@cbmb.com	Baker	20	s001
☐	Clay	d007	7865963869	Claybakes@cbmb.com	Baker	20	s001
☐	Clay	d008	7865963869	Claybakes@cbmb.com	Baker	20	s001
☐	Denzel	d009	7865912287	Denzelbakes@cbmb.com	Baker	20	s001

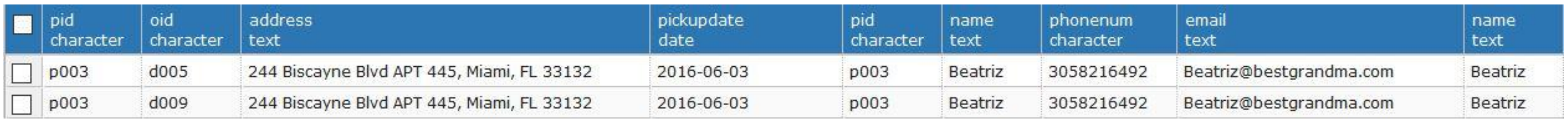
LocationofCustomerOrders

These views show the different areas customers who have purchased a cake from Cakes By Mary Bell.



```
CREATE VIEW LocationofCustomerOrders
AS
SELECT *
FROM customers c, people p
WHERE address
LIKE '%33132%'
AND c.pid = p.pid
```

Sample Data

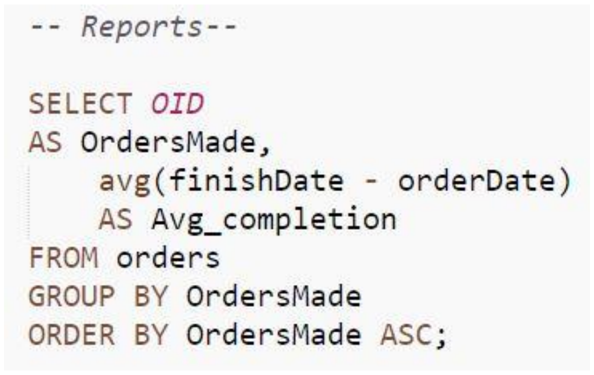


	pid character	oid character	address text	pickupdate date	pid character	name text	phonenum character	email text	name text
☐	p003	d005	244 Biscayne Blvd APT 445, Miami, FL 33132	2016-06-03	p003	Beatriz	3058216492	Beatriz@bestgrandma.com	Beatriz
☐	p003	d009	244 Biscayne Blvd APT 445, Miami, FL 33132	2016-06-03	p003	Beatriz	3058216492	Beatriz@bestgrandma.com	Beatriz

Reports

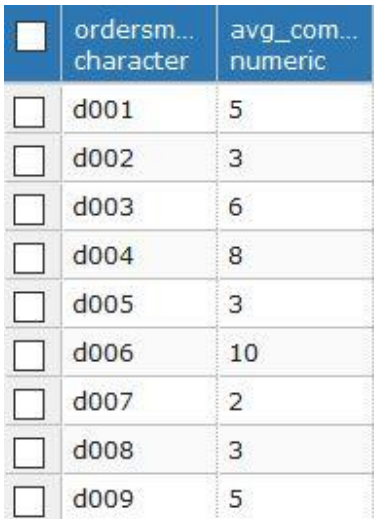
Average Completion of Orders

It is important for the business to know when each order is getting completed and how long it takes to complete them to ensure that the business is running and taking in orders and making sales.



```
-- Reports--

SELECT OID
AS OrdersMade,
    avg(finishDate - orderDate)
    AS Avg_completion
FROM orders
GROUP BY OrdersMade
ORDER BY OrdersMade ASC;
```

Sample Data



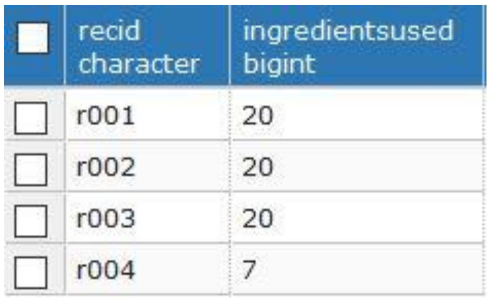
	ordersm... character	avg_com... numeric
☐	d001	5
☐	d002	3
☐	d003	6
☐	d004	8
☐	d005	3
☐	d006	10
☐	d007	2
☐	d008	3
☐	d009	5

Average Ingredients in a Recipe

It is essential to know how many ingredients are used in a recipe to be able to see when new ingredients need to be ordered.

```
SELECT RECID, SUM(quantity)
    AS IngredientsUsed
FROM recipeingredients
GROUP BY RECID
ORDER BY RECID ASC
```

Sample Data

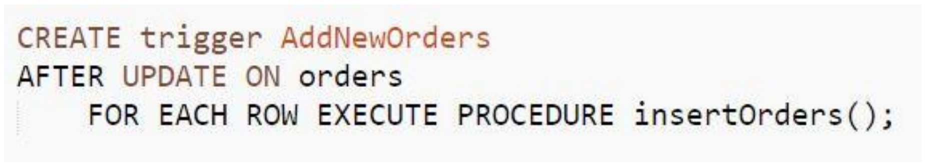


recid character	ingredientsused bigint
r001	20
r002	20
r003	20
r004	7

Triggers

AddNewOrders

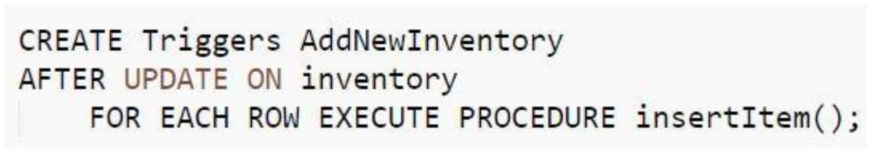
When a new order is being inputted then the Orders table must be updated, which also updates the customers table and baker table.



```
CREATE trigger AddNewOrders
AFTER UPDATE ON orders
    FOR EACH ROW EXECUTE PROCEDURE insertOrders();
```

AddNewInventory

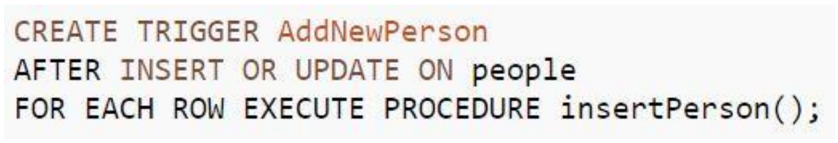
When a new item comes into the store it must be checked into by an employee and inputted into the inventory table.



```
CREATE Triggers AddNewInventory
AFTER UPDATE ON inventory
    FOR EACH ROW EXECUTE PROCEDURE insertItem();
```

AddNewPerson

Whenever a new customer or employee comes into the store they should be added to the people table in order to be able to keep track of the people that are associated with the shop



```
CREATE TRIGGER AddNewPerson
AFTER INSERT OR UPDATE ON people
FOR EACH ROW EXECUTE PROCEDURE insertPerson();
```

AddNewEmployee

When a new employee is hired they should be given a role, pay in US dollars, and ID. They should also provide their information in order to be added into the people table.

```
CREATE Triggers AddNewEmployee
AFTER UPDATE ON employees
    FOR EACH ROW EXECUTE PROCEDURE hireEmployee();
```

Stored Procedures

InsertOrders

For a new order to be inserted into the Order table it must be filled out on paper while in the store or done online.

```
CREATE OR REPLACE FUNCTION insertOrders()
RETURNS trigger AS $$
    BEGIN
        IF NEW.OID = true THEN
            INSERT INTO Orders
            VALUES(orderDate, layers, finishDate, occasion,
                price, layers, flavor, filling, RECID);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

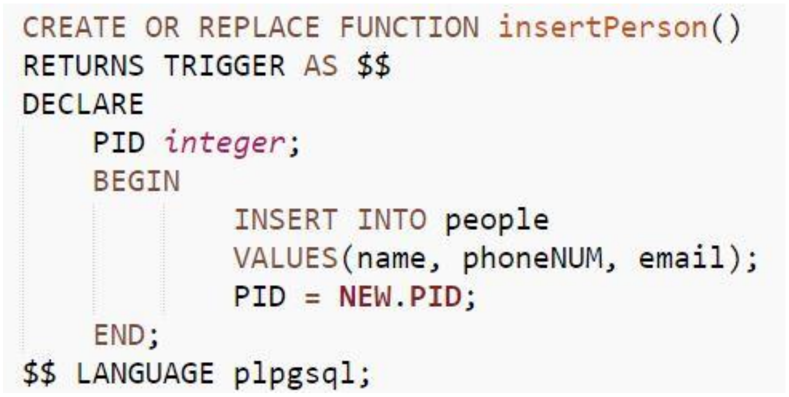
InsertItem

Whenever a new item is ordered and delivered it must be entered in the inventory table with the correct quantity.

```
CREATE OR REPLACE FUNCTION insertItem()
RETURNS TRIGGER AS $$
    BEGIN
         IF NEW.ITID = true THEN
            INSERT INTO inventory
            VALUES(name, instock, UID);
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

InsertPerson

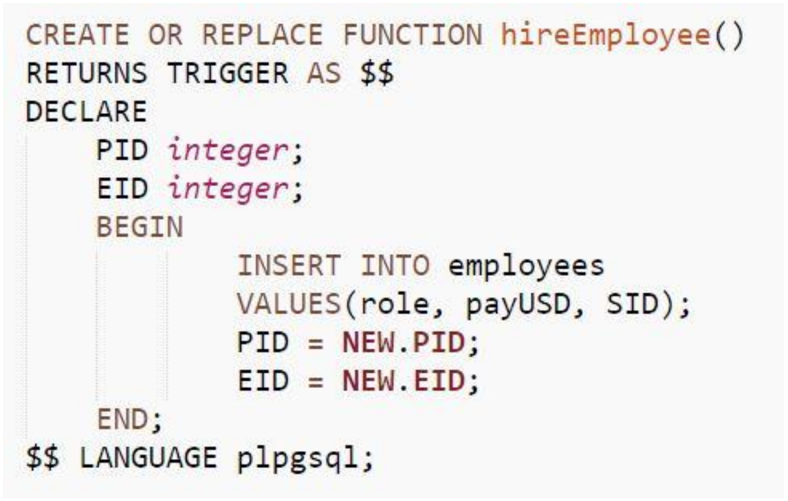
In this case, in order for a new employee to be inputted into the employees table they must first be inserted into the people table.



```
CREATE OR REPLACE FUNCTION insertPerson()
RETURNS TRIGGER AS $$
DECLARE
    PID integer;
    BEGIN
            INSERT INTO people
            VALUES(name, phoneNUM, email);
            PID = NEW.PID;
    END;
$$ LANGUAGE plpgsql;
```

InsertEmployees

Once they have been inserted into the people table the employee can now be inserted into the employees table



```
CREATE OR REPLACE FUNCTION hireEmployee()
RETURNS TRIGGER AS $$
DECLARE
    PID integer;
    EID integer;
    BEGIN
            INSERT INTO employees
            VALUES(role, payUSD, SID);
            PID = NEW.PID;
            EID = NEW.EID;
    END;
$$ LANGUAGE plpgsql;
```

Security

Administrator

The administrator should have the highest privileges because they are can change and update the table in whatever manner that benefits the store.

```
GRANT ALL PRIVLIGES ON ALL TABLE IN SCHEMA public to administrator;
```

Employees

The employees should be able to see their individual employee data but should not be able to make changes on them.

```
GRANT SELECT ON employees to employees;
```

Deliverers

The employees who do deliveries should be able to see and add a new delivery they are doing. They should not be able to update it as that may create conflict.

```
GRANT SELECT, INSERT ON deliverer to deliverer;
```

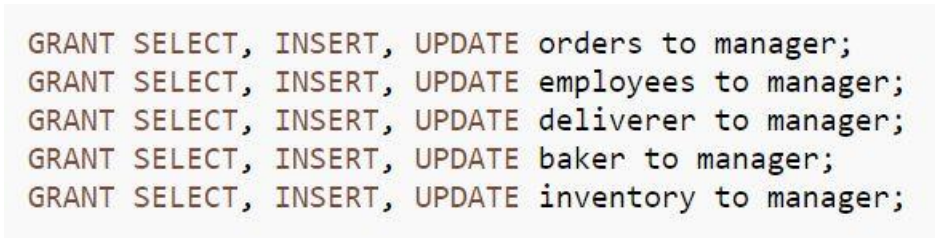
Bakers

The bakers should also be given the chance to insert and see the cakes they have worked on and the cakes other bakers are working on. Bakers should also be able to check the order specifications to be able to make a cake.

```
GRANT SELECT, INSERT ON baker to baker;
GRANT SELECT, ON orders to baker
```

Manager

Above the employees is the manager who should be able to select, insert, and update in the orders, employees, deliverer, bakers, and inventory tables.



```
GRANT SELECT, INSERT, UPDATE orders to manager;
GRANT SELECT, INSERT, UPDATE employees to manager;
GRANT SELECT, INSERT, UPDATE deliverer to manager;
GRANT SELECT, INSERT, UPDATE baker to manager;
GRANT SELECT, INSERT, UPDATE inventory to manager;
```

Implementation Notes

- The orders must be fully filled out before they are submitted into the database. The finish date can be left empty until the cake has been finished.
- Once a customer has picked up and paid for a cake the date which they came must be entered into the database.
- If a new store is created, then there would be different SID for that store but the suppliers would still be the same. The new store would have to change its name to “Branch One” while the original store would have to change its name to “Main Store”.
- The administrator has the highest amount of access because they can select, insert, or update a supplier and the people table.
- Customers are automatically entered into the database but they cannot be removed or changed by the manager.

Known Problems

- The bakers table needs further information as to what sets it apart from the employees table.
- The store table needs more information about a certain store. In the future it may include a column that states the amount of employees working in a specific store or he customers visiting the store per day.
- For a customer to change their order they must call the manager in order to update it. If they wanted to change their order they should be able to do so if they go online and submit a form.
- Only customers who have made an order appear on the customers table. Should a customer still be allowed into the table if they made an order but it was cancelled?

Future Enhancements

- Once a new store opens then the store tables should be updated.
- There should be a shift table in order to allow shifts for employees.