Oolong Fox

Inventory Database

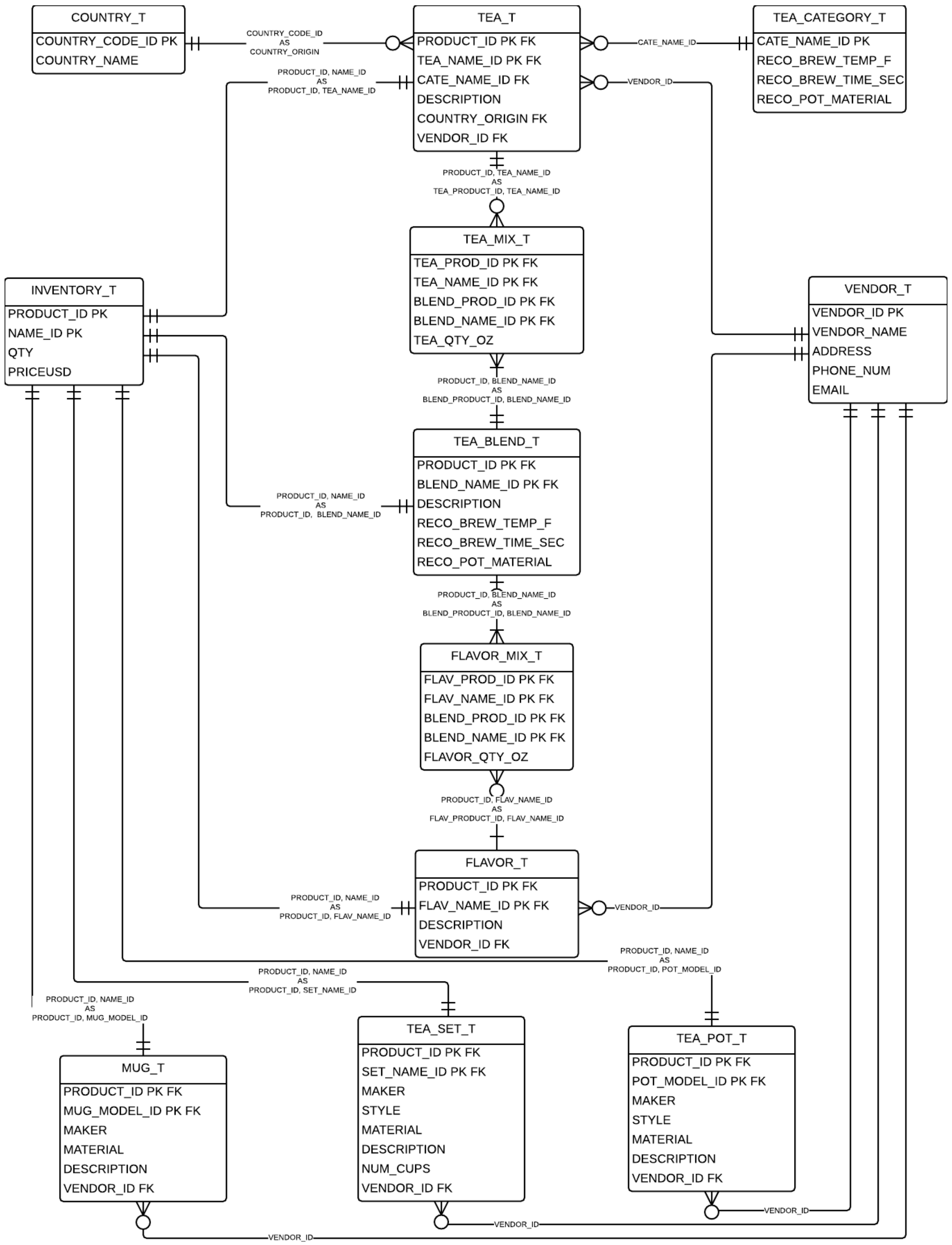
Steven Luke Bacdayan
11-30-2014

Table of Contents

<u>Executive Summary</u>

Tea is the second most consumed beverage in the world, behind water. In the United States, over three billion gallons of tea is consumed annually, with black tea making up over 50%. This Oolong Fox is a new small business that hopes to capitalize on those statistics. The Oolong Fox is a tea shop that specializes in selling tea products and tea related accessories.

As such, this small business needs a system to keep track of its inventory, product and vendor information, and prices. This document outlines a database with its tables and relationships that are needed, along with views, stored procedures, triggers, and security. Examples with test data will be shown throughout the document, to give an idea of the results that this database will provide. This database was created to be used with PostgresSQL.


COUNTRY_T
COUNTRY_CODE_ID PK
COUNTRY_NAME
COUNTRY_CODE_ID AS COUNTRY_ORIGIN
TEA_T
PRODUCT_ID PK FK
TEA_NAME_ID PK FK
CATE_NAME_ID FK
DESCRIPTION
COUNTRY_ORIGIN FK
VENDOR_ID FK
CATE_NAME_ID
TEA_CATEGORY_T
CATE_NAME_ID PK
RECO_BREW_TEMP_F
RECO_BREW_TIME_SEC
RECO_POT_MATERIAL
PRODUCT_ID, NAME_ID AS PRODUCT_ID, TEA_NAME_ID
VENDOR_ID
PRODUCT_ID, TEA_NAME_ID AS TEA_PRODUCT_ID, TEA_NAME_ID
TEA_MIX_T
TEA_PROD_ID PK FK
TEA_NAME_ID PK FK
BLEND_PROD_ID PK FK
BLEND_NAME_ID PK FK
TEA_QTY_OZ
INVENTORY_T
PRODUCT_ID PK
NAME_ID PK
QTY
PRICEUSD
VENDOR_T
VENDOR_ID PK
VENDOR_NAME
ADDRESS
PHONE_NUM
EMAIL
PRODUCT_ID, BLEND_NAME_ID AS BLEND_PRODUCT_ID, BLEND_NAME_ID
TEA_BLEND_T
PRODUCT_ID PK FK
BLEND_NAME_ID PK FK
DESCRIPTION
RECO_BREW_TEMP_F
RECO_BREW_TIME_SEC
RECO_POT_MATERIAL
PRODUCT_ID, NAME_ID AS PRODUCT_ID, BLEND_NAME_ID
PRODUCT_ID, BLEND_NAME_ID AS BLEND_PRODUCT_ID, BLEND_NAME_ID
FLAVOR_MIX_T
FLAV_PROD_ID PK FK
FLAV_NAME_ID PK FK
BLEND_PROD_ID PK FK
BLEND_NAME_ID PK FK
FLAVOR_QTY_OZ
PRODUCT_ID, FLAV_NAME_ID AS FLAV_PRODUCT_ID, FLAV_NAME_ID
FLAVOR_T
PRODUCT_ID PK FK
FLAV_NAME_ID PK FK
DESCRIPTION
VENDOR_ID FK
PRODUCT_ID, NAME_ID AS PRODUCT_ID, FLAV_NAME_ID
VENDOR_ID
PRODUCT_ID, NAME_ID AS PRODUCT_ID, POT_MODEL_ID
PRODUCT_ID, NAME_ID AS PRODUCT_ID, SET_NAME_ID
PRODUCT_ID, NAME_ID AS PRODUCT_ID, MUG_MODEL_ID
MUG_T
PRODUCT_ID PK FK
MUG_MODEL_ID PK FK
MAKER
MATERIAL
DESCRIPTION
VENDOR_ID FK
TEA_SET_T
PRODUCT_ID PK FK
SET_NAME_ID PK FK
MAKER
STYLE
MATERIAL
DESCRIPTION
NUM_CUPS
VENDOR_ID FK
TEA_POT_T
PRODUCT_ID PK FK
POT_MODEL_ID PK FK
MAKER
STYLE
MATERIAL
DESCRIPTION
VENDOR_ID FK
VENDOR_ID
VENDOR_ID
VENDOR_ID

Tables

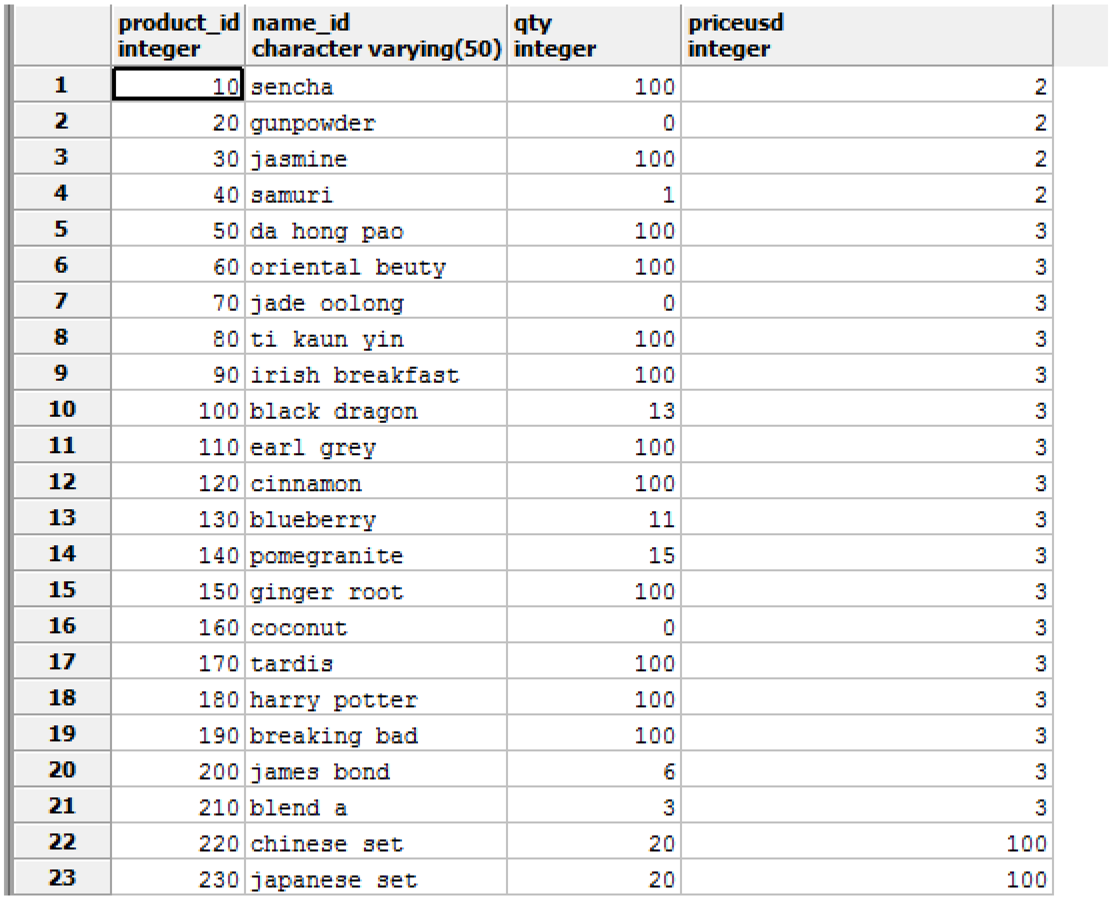
Please keep in mind that the data is purely for testing and displaying results. Accuracy of the data itself, such as country's and their codes, may vary.

INVENTORY_T

The INVENTORY_T table holds every single item that the store sells. The PRODUCT_ID and NAME_ID attributes are inherited by the different product tables.

Functional dependencies: PRODUCT_ID, NAME_ID > QTY, PRICEUSD

```
CREATE TABLE INVENTORY_T(
        PRODUCT_ID INT NOT NULL,
        NAME_ID VARCHAR(50) NOT NULL,
        QTY INT NOT NULL,
        PRICEUSD INT NOT NULL,
        PRIMARY KEY (PRODUCT_ID, NAME_ID)
);
```



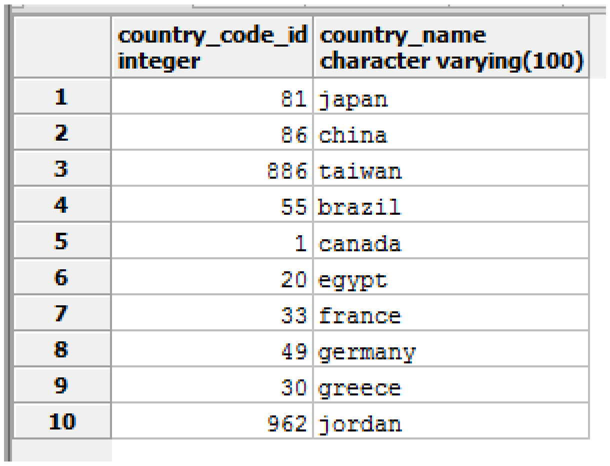
	product_id integer	name_id character varying(50)	qty integer	priceusd integer
1	10	sencha	100	2
2	20	gunpowder	0	2
3	30	jasmine	100	2
4	40	samuri	1	2
5	50	da hong pao	100	3
6	60	oriental beuty	100	3
7	70	jade oolong	0	3
8	80	ti kaun yin	100	3
9	90	irish breakfast	100	3
10	100	black dragon	13	3
11	110	earl grey	100	3
12	120	cinnamon	100	3
13	130	blueberry	11	3
14	140	pomegranite	15	3
15	150	ginger root	100	3
16	160	coconut	0	3
17	170	tardis	100	3
18	180	harry potter	100	3
19	190	breaking bad	100	3
20	200	james bond	6	3
21	210	blend a	3	3
22	220	chinese set	20	100
23	230	japanese set	20	100

COUNTRY_T

The COUNTRY_T table holds the different countries that produce tea. This is inherited by the TEA_T table, to give descriptive information by providing a teas country of origin.

Functional dependencies: COUNTRY_CODE_ID > COUNTRY_NAME

```
CREATE TABLE IF NOT EXISTS COUNTRY_T(
	COUNTRY_CODE_ID INT NOT NULL,
	COUNTRY_NAME VARCHAR(100),
	PRIMARY KEY (COUNTRY_CODE_ID)
);
```



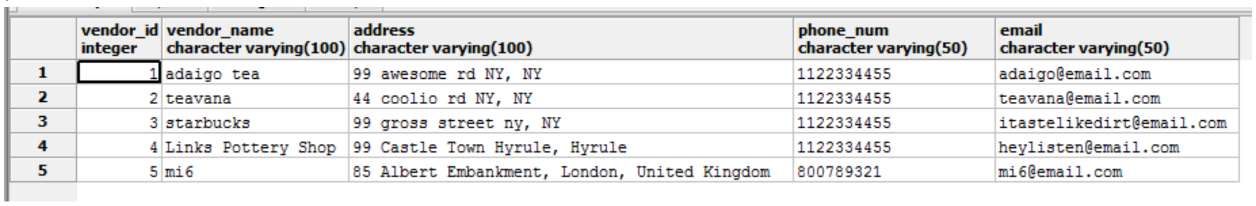
	country_code_id integer	country_name character varying(100)
1	81	japan
2	86	china
3	886	taiwan
4	55	brazil
5	1	canada
6	20	egypt
7	33	france
8	49	germany
9	30	greece
10	962	jordan

VENDOR_T

The VENDOR_T table is used to keep track of the different suppliers and their contact information. The VENDOR_ID is inherited by the all the products tables, except for TEA_BLENDS_T, since blended teas are supposed be made in the store.

Functional dependencies VENDOR_ID > VENDOR_NAME, ADDRESS, PHONE_NUM, EMAIL

```
CREATE TABLE IF NOT EXISTS VENDOR_T(
	VENDOR_ID INT NOT NULL,
	VENDOR_NAME VARCHAR(100) NOT NULL,
	ADDRESS VARCHAR(100) NOT NULL,
	PHONE_NUM VARCHAR(50) NOT NULL,
	EMAIL VARCHAR(50) NOT NULL,
	PRIMARY KEY (VENDOR_ID)
);
```



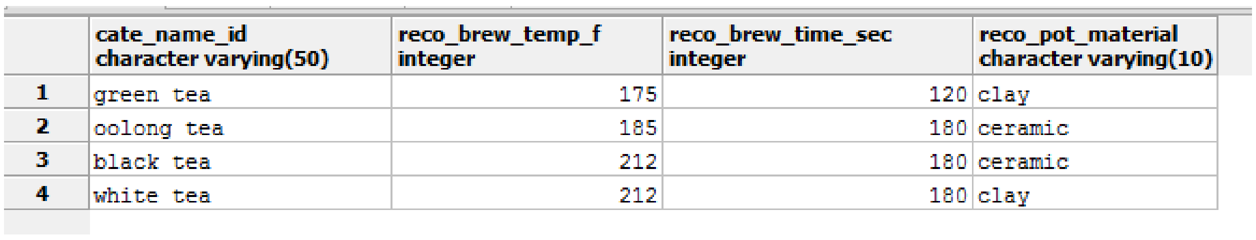
	vendor_id integer	vendor_name character varying(100)	address character varying(100)	phone_num character varying(50)	email character varying(50)
1	1	adaigo tea	99 awesome rd NY, NY	1122334455	adaigo@email.com
2	2	teavana	44 coolio rd NY, NY	1122334455	teavana@email.com
3	3	starbucks	99 gross street ny, NY	1122334455	itastelikedirt@email.com
4	4	Links Pottery Shop	99 Castle Town Hyrule, Hyrule	1122334455	heylisten@email.com
5	5	mi6	85 Albert Embankment, London, United Kingdom	800789321	mi6@email.com

TEA_CATEGORY_T

The TEA_CATEGORY_T keeps track of the different types of teas. Each category of tea has a recommended steep time and water temperature to optimize taste. The CATE_NAME_ID is inherited by the TEA_T table, since all teas, besides blended ones, can fall into one of these categories.

Functional dependencies CATE_NAME_ID > RECO_BREW_TEMP_F, RECO_BREW_TIME_SEC, RECO_POT_MATERIAL

```
CREATE TABLE IF NOT EXISTS TEA_CATEGORY_T (
        CATE_NAME_ID VARCHAR(50) NOT NULL,
        RECO_BREW_TEMP_F INT NOT NULL,
        RECO_BREW_TIME_SEC INT NOT NULL,
        RECO_POT_MATERIAL VARCHAR (10),
        PRIMARY KEY (CATE_NAME_ID)
);
```



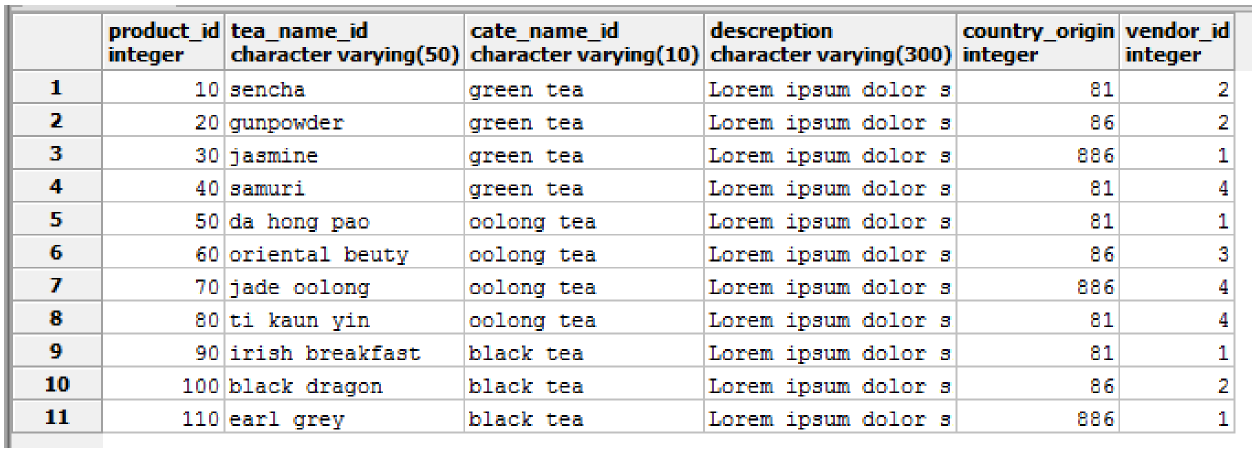
	cate_name_id character varying(50)	reco_brew_temp_f integer	reco_brew_time_sec integer	reco_pot_material character varying(10)
1	green tea	175	120	clay
2	oolong tea	185	180	ceramic
3	black tea	212	180	ceramic
4	white tea	212	180	clay

TEA_T

The TEA_T table holds all the basic tea products. PRODUCT_T and TEA_NAME_T is inherited by the TEA_MIX_T table to show what teas are in what blends.

Functional dependencies PRODUCT_ID, TEA_NAME_ID > DESCRIPTION

```
CREATE TABLE IF NOT EXISTS TEA_T(
	PRODUCT_ID INT NOT NULL,
	TEA_NAME_ID VARCHAR(50) NOT NULL UNIQUE,
	CATE_NAME_ID VARCHAR(10) NOT NULL,
	DESCRIPTION VARCHAR(300),
	COUNTRY_ORIGIN INT NOT NULL,
	VENDOR_ID INT NOT NULL ,
	FOREIGN KEY (PRODUCT_ID, TEA_NAME_ID) REFERENCES INVENTORY_T(PRODUCT_ID,
NAME_ID),
	FOREIGN KEY (CATE_NAME_ID) REFERENCES TEA_CATEGORY_T(CATE_NAME_ID),
	FOREIGN KEY (COUNTRY_ORIGIN) REFERENCES COUNTRY_T(COUNTRY_CODE_ID),
	FOREIGN KEY (VENDOR_ID) REFERENCES VENDOR_T(VENDOR_ID),
	PRIMARY KEY(PRODUCT_ID, TEA_NAME_ID)
);
```



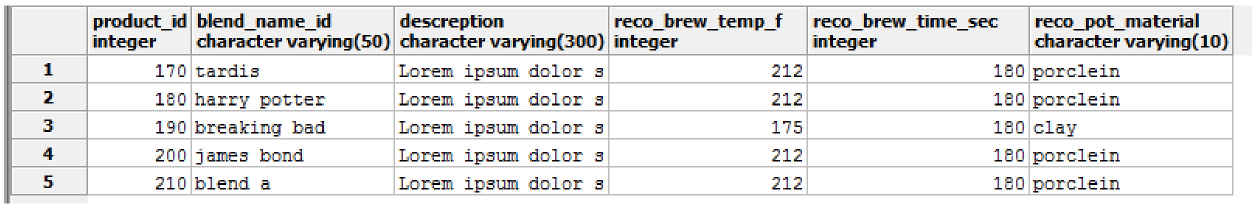
	product_id integer	tea_name_id character varying(50)	cate_name_id character varying(10)	descreption character varying(300)	country_origin integer	vendor_id integer
1	10	sencha	green tea	Lorem ipsum dolor s	81	2
2	20	gunpowder	green tea	Lorem ipsum dolor s	86	2
3	30	jasmine	green tea	Lorem ipsum dolor s	886	1
4	40	samuri	green tea	Lorem ipsum dolor s	81	4
5	50	da hong pao	oolong tea	Lorem ipsum dolor s	81	1
6	60	oriental beuty	oolong tea	Lorem ipsum dolor s	86	3
7	70	jade oolong	oolong tea	Lorem ipsum dolor s	886	4
8	80	ti kaun yin	oolong tea	Lorem ipsum dolor s	81	4
9	90	irish breakfast	black tea	Lorem ipsum dolor s	81	1
10	100	black dragon	black tea	Lorem ipsum dolor s	86	2
11	110	earl grey	black tea	Lorem ipsum dolor s	886	1

TEA_BLEND_T

The TEA_BLEND_T table holds the names of blends which are teas that are mixed together with different flavors. The PRODUCT_ID and BLEND_NAME_ID attributes are inherited by TEA_MIX_T and FLAVOR_MIX_T

Functional dependencies: PROUDUCT_ID, BLEND_NAME_ID >RECO_BREW_TEMP_F, RECO_BREW_TIME_SEC, RECO_POT_MATERIAL, DESCRIPTION

```
CREATE TABLE IF NOT EXISTS TEA_BLEND_T(
        PRODUCT_ID INT NOT NULL,
        BLEND_NAME_ID VARCHAR(50) NOT NULL UNIQUE,
        DESCRIPTION VARCHAR(300),
        RECO_BREW_TEMP_F INT NOT NULL,
        RECO_BREW_TIME_SEC INT NOT NULL,
        RECO_POT_MATERIAL VARCHAR (10),
        FOREIGN KEY (PRODUCT_ID, BLEND_NAME_ID) REFERENCES
INVENTORY_T(PRODUCT_ID, NAME_ID),
        PRIMARY KEY(PRODUCT_ID, BLEND_NAME_ID)
);
```



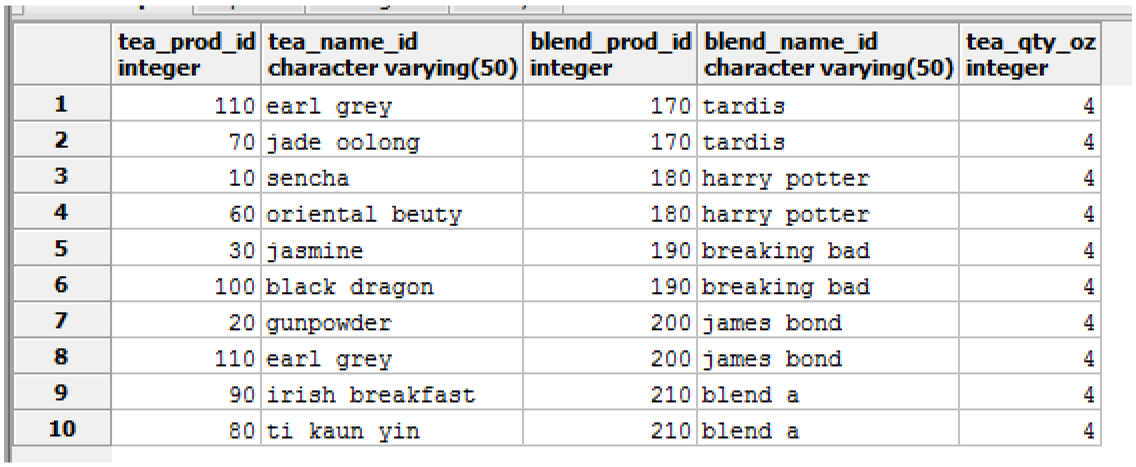
	product_id integer	blend_name_id character varying(50)	descreption character varying(300)	reco_brew_temp_f integer	reco_brew_time_sec integer	reco_pot_material character varying(10)
1	170	tardis	Lorem ipsum dolor s	212	180	porclein
2	180	harry potter	Lorem ipsum dolor s	212	180	porclein
3	190	breaking bad	Lorem ipsum dolor s	175	180	clay
4	200	james bond	Lorem ipsum dolor s	212	180	porclein
5	210	blend a	Lorem ipsum dolor s	212	180	porclein

TEA_MIX_T

The TEA_MIX_T tables contains teas and blends. Used for figuring out what teas are in a blend. This is helpful in finding recipes for different blends.

Functional Dependencies TEA_PROD_ID, TEA_NAME_ID, BLEND_PROD_ID, BLEND_NAME_ID > TEA_QTY_OZ

```
CREATE TABLE IF NOT EXISTS TEA_MIX_T(
        TEA_PROD_ID INT NOT NULL,
        TEA_NAME_ID VARCHAR(50) NOT NULL,
        BLEND_PROD_ID INT NOT NULL,
        BLEND_NAME_ID VARCHAR(50) NOT NULL,
        TEA_QTY_OZ INT NOT NULL,
        FOREIGN KEY (TEA_PROD_ID, TEA_NAME_ID) REFERENCES TEA_T(PRODUCT_ID,
TEA_NAME_ID),
        FOREIGN KEY (BLEND_PROD_ID, BLEND_NAME_ID) REFERENCES
TEA_BLEND_T(PRODUCT_ID, BLEND_NAME_ID),
        PRIMARY KEY (TEA_PROD_ID, TEA_NAME_ID, BLEND_PROD_ID, BLEND_NAME_ID)
);
```



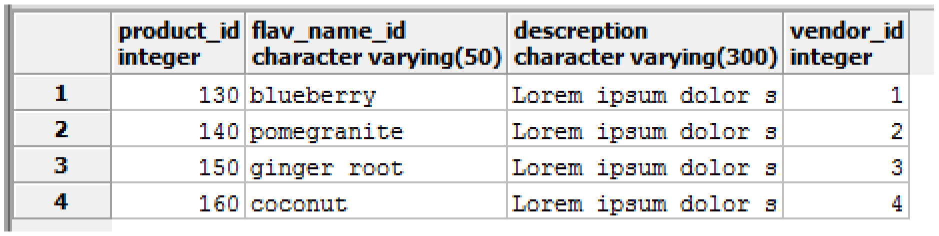
	tea_prod_id integer	tea_name_id character varying(50)	blend_prod_id integer	blend_name_id character varying(50)	tea_qty_oz integer
1	110	earl grey	170	tardis	4
2	70	jade oolong	170	tardis	4
3	10	sencha	180	harry potter	4
4	60	oriental beuty	180	harry potter	4
5	30	jasmine	190	breaking bad	4
6	100	black dragon	190	breaking bad	4
7	20	gunpowder	200	james bond	4
8	110	earl grey	200	james bond	4
9	90	irish breakfast	210	blend a	4
10	80	ti kaun yin	210	blend a	4

FLAVOR_T

FLAVOR_T contains the different flavor products. These will mostly be used in making tea blends in the store, but can be sold to customers who want to make their own blends.

Functional dependencies: PRODUCT_ID, FLAV_NAME_ID > DESCRIPTION

```
CREATE TABLE IF NOT EXISTS FLAVOR_T(
	PRODUCT_ID INT NOT NULL,
	FLAV_NAME_ID VARCHAR(50),
	DESCRIPTION VARCHAR(300),
	VENDOR_ID INT NOT NULL,
	FOREIGN KEY (PRODUCT_ID, FLAV_NAME_ID) REFERENCES INVENTORY_T(PRODUCT_ID,
NAME_ID),
	FOREIGN KEY (VENDOR_ID) REFERENCES VENDOR_T(VENDOR_ID),
	PRIMARY KEY (PRODUCT_ID, FLAV_NAME_ID)
);
```



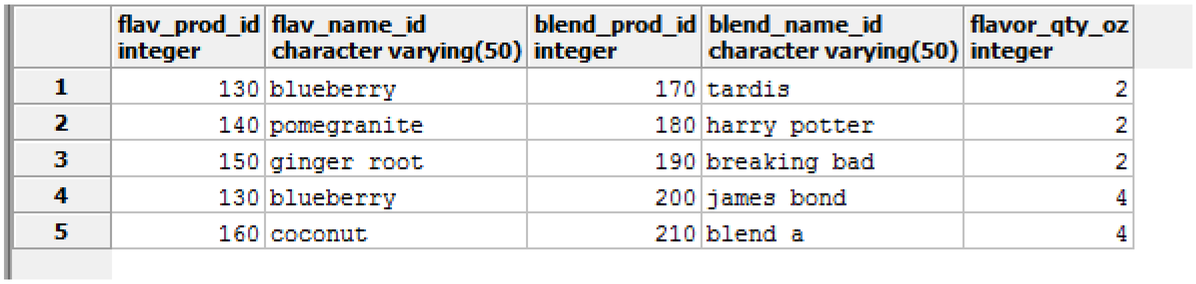
	product_id integer	flav_name_id character varying(50)	descreption character varying(300)	vendor_id integer
1	130	blueberry	Lorem ipsum dolor s	1
2	140	pomegranite	Lorem ipsum dolor s	2
3	150	ginger root	Lorem ipsum dolor s	3
4	160	coconut	Lorem ipsum dolor s	4

FLAVOR_MIX_T

FLAVOR_MIX_T contains what flavors are in what tea blends. This is helpful in finding recipes for different blends.

Functional Dependencies:
FLAV_PROD_ID, FLAV_NAME_ID, BLEND_PROD_ID, BLEND_NAME_ID > FLAV_QTY_OZ

```
CREATE TABLE IF NOT EXISTS FLAVOR_MIX_T(
        FLAV_PROD_ID INT NOT NULL,
        FLAV_NAME_ID VARCHAR(50) NOT NULL,
        BLEND_PROD_ID INT NOT NULL,
        BLEND_NAME_ID VARCHAR(50) NOT NULL,
        FLAVOR_QTY_OZ INT NOT NULL,
        FOREIGN KEY (FLAV_PROD_ID, FLAV_NAME_ID) REFERENCES  FLAVOR_T(PRODUCT_ID,
FLAV_NAME_ID),
        FOREIGN KEY (BLEND_PROD_ID, BLEND_NAME_ID) REFERENCES
TEA_BLEND_T(PRODUCT_ID, BLEND_NAME_ID),
        PRIMARY KEY (FLAV_PROD_ID, FLAV_NAME_ID, BLEND_PROD_ID, BLEND_NAME_ID)
);
```



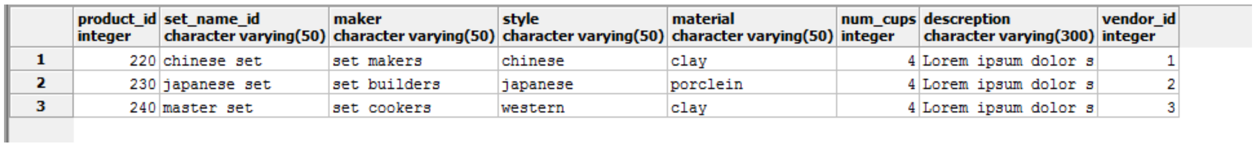
	flav_prod_id integer	flav_name_id character varying(50)	blend_prod_id integer	blend_name_id character varying(50)	flavor_qty_oz integer
1	130	blueberry	170	tardis	2
2	140	pomegranite	180	harry potter	2
3	150	ginger root	190	breaking bad	2
4	130	blueberry	200	james bond	4
5	160	coconut	210	blend a	4

TEA_SET_T

TEA_SET_T contains the different tea set products that the store sells.

PRODUCT_ID, SET_NAME_ID > MAKER, STYLE, MATERIAL, NUM_CUPS, DESCRIPTION

```
CREATE TABLE IF NOT EXISTS TEA_SET_T(
	PRODUCT_ID INT NOT NULL,
	SET_NAME_ID VARCHAR(50) NOT NULL,
	MAKER VARCHAR(50) NOT NULL,
	STYLE VARCHAR(50),
	MATERIAL VARCHAR(50),
	NUM_CUPS INT,
	DESCRIPTION VARCHAR(300),
	VENDOR_ID INT NOT NULL,
	FOREIGN KEY (PRODUCT_ID, SET_NAME_ID) REFERENCES INVENTORY_T(PRODUCT_ID,
NAME_ID),
	FOREIGN KEY (VENDOR_ID) REFERENCES VENDOR_T(VENDOR_ID),
	PRIMARY KEY (PRODUCT_ID, SET_NAME_ID)
);
```



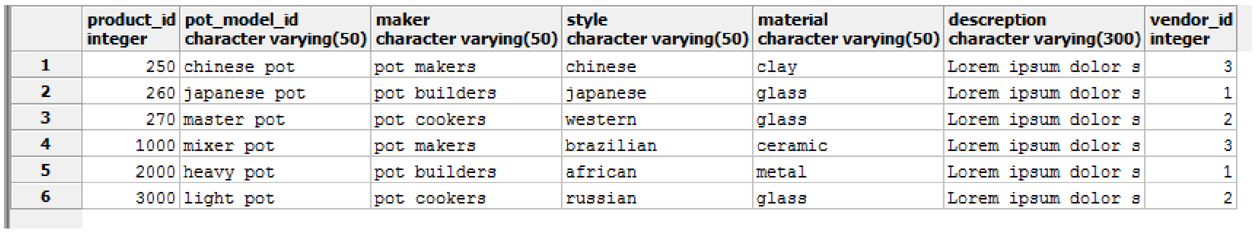
	product_id integer	set_name_id character varying(50)	maker character varying(50)	style character varying(50)	material character varying(50)	num_cups integer	descreption character varying(300)	vendor_id integer
1	220	chinese set	set makers	chinese	clay	4	Lorem ipsum dolor s	1
2	230	japanese set	set builders	japanese	porclein	4	Lorem ipsum dolor s	2
3	240	master set	set cookers	western	clay	4	Lorem ipsum dolor s	3

TEA_POT_T

TEA_POT_T contains the different tea pot products. By matching the pots material with a tea recommended pot material, deals and bundles can easily be created to sell both.

Functional dependencies: PRODUCT_ID, POT_MODEL_ID > MAKER, STYLE, MATERIAL, DESCRIPTION

```
CREATE TABLE IF NOT EXISTS TEA_POT_T(
	PRODUCT_ID INT NOT NULL,
	POT_MODEL_ID VARCHAR(50) NOT NULL,
	MAKER VARCHAR(50),
	STYLE VARCHAR(50),
	MATERIAL VARCHAR(50),
	DESCRIPTION VARCHAR(300),
	VENDOR_ID INT NOT NULL,
	FOREIGN KEY (PRODUCT_ID, POT_MODEL_ID) REFERENCES INVENTORY_T(PRODUCT_ID,
NAME_ID),
	FOREIGN KEY (VENDOR_ID) REFERENCES VENDOR_T(VENDOR_ID),
	PRIMARY KEY (PRODUCT_ID, POT_MODEL_ID)
);
```



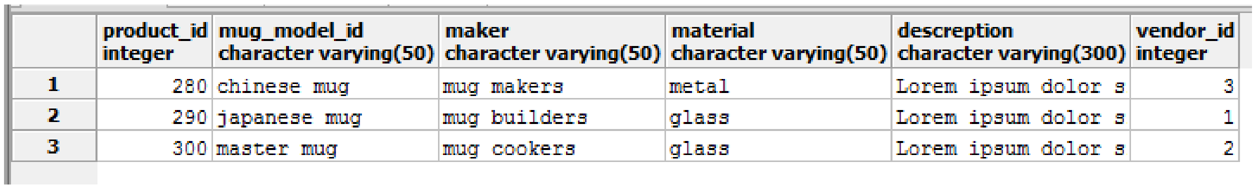
	product_id integer	pot_model_id character varying(50)	maker character varying(50)	style character varying(50)	material character varying(50)	descreption character varying(300)	vendor_id integer
1	250	chinese pot	pot makers	chinese	clay	Lorem ipsum dolor s	3
2	260	japanese pot	pot builders	japanese	glass	Lorem ipsum dolor s	1
3	270	master pot	pot cookers	western	glass	Lorem ipsum dolor s	2
4	1000	mixer pot	pot makers	brazilian	ceramic	Lorem ipsum dolor s	3
5	2000	heavy pot	pot builders	african	metal	Lorem ipsum dolor s	1
6	3000	light pot	pot cookers	russian	glass	Lorem ipsum dolor s	2

MUG_T

MUG_T contains the different mug products that the store sells.

Functional dependencies PRODUCT_ID, MUG_MODEL_ID > MAKER, MATERIAL, DESCRIPTION

```
CREATE TABLE IF NOT EXISTS MUG_T(
	PRODUCT_ID INT NOT NULL,
	MUG_MODEL_ID VARCHAR(50) NOT NULL,
	MAKER VARCHAR(50),
	MATERIAL VARCHAR(50),
	DESCRIPTION VARCHAR(300),
	VENDOR_ID INT NOT NULL,
	FOREIGN KEY (PRODUCT_ID, MUG_MODEL_ID) REFERENCES
INVENTORY_T(PRODUCT_ID, NAME_ID),
	FOREIGN KEY (VENDOR_ID) REFERENCES VENDOR_T(VENDOR_ID),
	PRIMARY KEY (PRODUCT_ID, MUG_MODEL_ID)
);
```

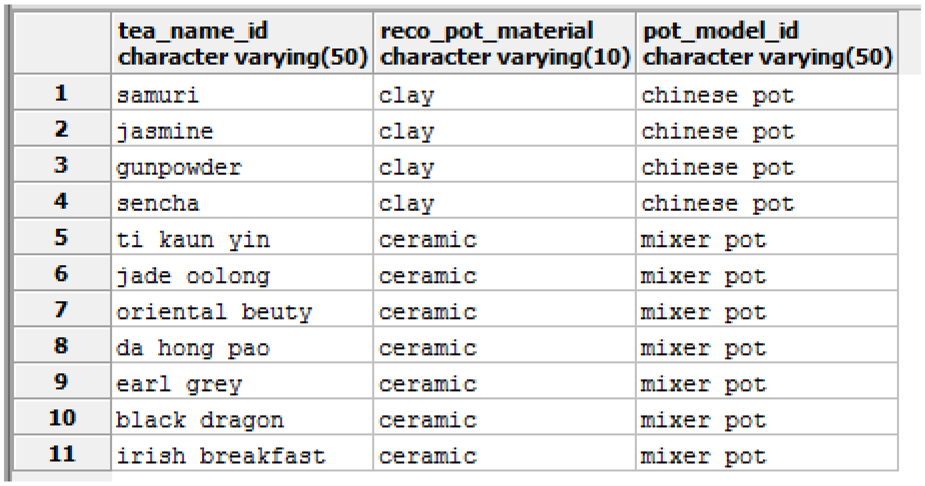


	product_id integer	mug_model_id character varying(50)	maker character varying(50)	material character varying(50)	descreption character varying(300)	vendor_id integer
1	280	chinese mug	mug makers	metal	Lorem ipsum dolor s	3
2	290	japanese mug	mug builders	glass	Lorem ipsum dolor s	1
3	300	master mug	mug cookers	glass	Lorem ipsum dolor s	2

Reports and Their Results

Get all the recommend materials for all the teas, and match them with a pot that made out of that recommend material. Useful for trying to sell addition products with an order, or making gift bundles

```
SELECT TEA_T.TEA_NAME_ID, TEA_CATEGORY_T.RECO_POT_MATERIAL,
TEA_POT_T.POT_MODEL_ID
FROM TEA_T LEFT JOIN TEA_CATEGORY_T ON
(TEA_T.CATE_NAME_ID=TEA_CATEGORY_T.CATE_NAME_ID)
                    LEFT JOIN TEA_POT_T ON
(TEA_POT_T.MATERIAL=TEA_CATEGORY_T.RECO_POT_MATERIAL);
```



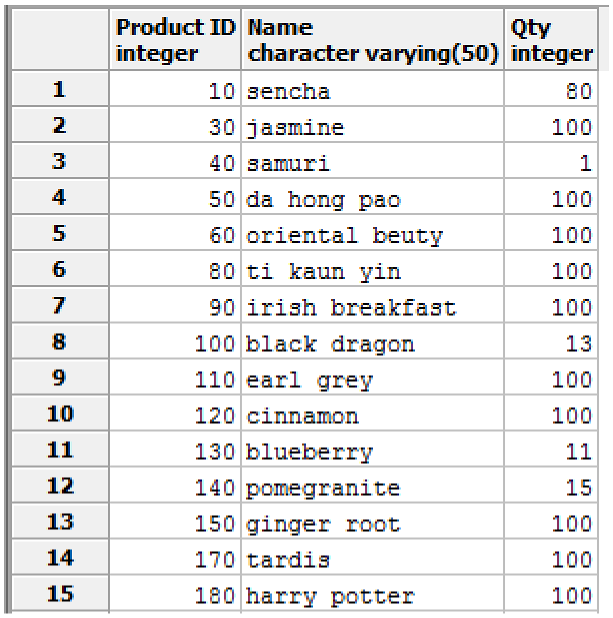
	tea_name_id character varying(50)	reco_pot_material character varying(10)	pot_model_id character varying(50)
1	samuri	clay	chinese pot
2	jasmine	clay	chinese pot
3	gunpowder	clay	chinese pot
4	sencha	clay	chinese pot
5	ti kaun yin	ceramic	mixer pot
6	jade oolong	ceramic	mixer pot
7	oriental beuty	ceramic	mixer pot
8	da hong pao	ceramic	mixer pot
9	earl grey	ceramic	mixer pot
10	black dragon	ceramic	mixer pot
11	irish breakfast	ceramic	mixer pot

Views

view_in_stock

See all the items that are in stock, by seeing which items have a QTY not equal to zero

```
CREATE VIEW view_in_stock
AS
SELECT PRODUCT_ID AS "Product ID",
                    NAME_ID AS "Name",
                    QTY AS "Qty"
FROM INVENTORY_T
WHERE QTY != 0
ORDER BY PRODUCT_ID ASC;
```

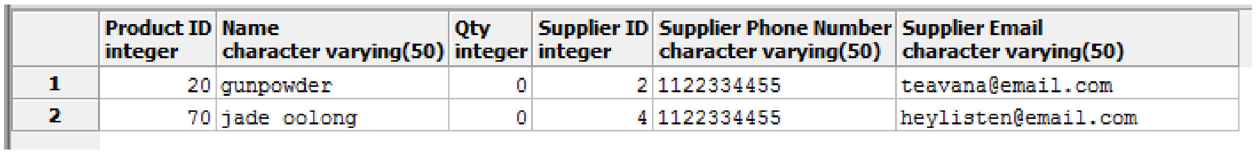


	Product ID integer	Name character varying(50)	Qty integer
1	10	sencha	80
2	30	jasmine	100
3	40	samuri	1
4	50	da hong pao	100
5	60	oriental beuty	100
6	80	ti kaun yin	100
7	90	irish breakfast	100
8	100	black dragon	13
9	110	earl grey	100
10	120	cinnamon	100
11	130	blueberry	11
12	140	pomegranite	15
13	150	ginger root	100
14	170	tardis	100
15	180	harry potter	100

view_tea_out_stock

View teas that not in stock and also display the suppliers contact information

```
CREATE VIEW view_tea_out_stock
AS
SELECT TEA_T.PRODUCT_ID AS "Product ID",
                    TEA_T.TEA_NAME_ID AS "Name",
                    INVENTORY_T.QTY AS "Qty",
                    TEA_T.VENDOR_ID AS "Supplier ID",
                    VENDOR_T.PHONE_NUM AS "Supplier Phone Number",
                    VENDOR_T.Email AS "Supplier Email"
FROM TEA_T
                    LEFT JOIN  INVENTORY_T ON (INVENTORY_T.PRODUCT_ID=TEA_T.PRODUCT_ID)
                    LEFT JOIN VENDOR_T ON (TEA_T.VENDOR_ID=VENDOR_T.VENDOR_ID)
WHERE INVENTORY_T.QTY=0
ORDER BY TEA_T.PRODUCT_ID ASC;
```

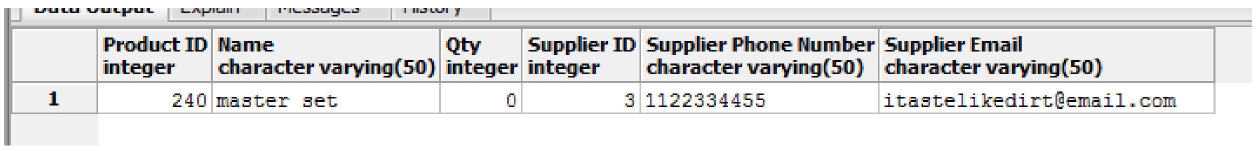


	Product ID integer	Name character varying(50)	Qty integer	Supplier ID integer	Supplier Phone Number character varying(50)	Supplier Email character varying(50)
1	20	gunpowder	0	2	1122334455	teavana@email.com
2	70	jade oolong	0	4	1122334455	heylisten@email.com

view_tea_set_out_stock

View tea sets that not in stock and also display the suppliers contact information

```
CREATE VIEW view_tea_set_out_stock
AS
SELECT TEA_SET_T.PRODUCT_ID AS "Product ID",
            TEA_SET_T.SET_NAME_ID AS "Name",
            INVENTORY_T.QTY AS "Qty",
            TEA_SET_T.VENDOR_ID AS "Supplier ID",
            VENDOR_T.PHONE_NUM AS "Supplier Phone Number",
            VENDOR_T.Email AS "Supplier Email"
FROM TEA_SET_T
            LEFT JOIN  INVENTORY_T ON
(INVENTORY_T.PRODUCT_ID=TEA_SET_T.PRODUCT_ID)
            LEFT JOIN VENDOR_T ON (TEA_SET_T.VENDOR_ID=VENDOR_T.VENDOR_ID)
WHERE INVENTORY_T.QTY=0
ORDER BY TEA_SET_T.PRODUCT_ID ASC;
```

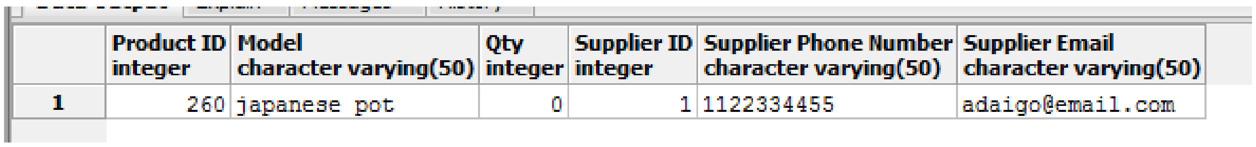


	Product ID integer	Name character varying(50)	Qty integer	Supplier ID integer	Supplier Phone Number character varying(50)	Supplier Email character varying(50)
1	240	master set	0	3	1122334455	itastelikedirt@email.com

view_tea_pot_out_stock

View tea pots that not in stock and also display the suppliers contact information

```
CREATE VIEW view_tea_pot_out_stock
AS
SELECT TEA_POT_T.PRODUCT_ID AS "Product ID",
            TEA_POT_T.POT_MODEL_ID AS "Model",
            INVENTORY_T.QTY AS "Qty",
            TEA_POT_T.VENDOR_ID AS "Supplier ID",
            VENDOR_T.PHONE_NUM AS "Supplier Phone Number",
            VENDOR_T.Email AS "Supplier Email"
FROM TEA_POT_T
            LEFT JOIN  INVENTORY_T ON
(INVENTORY_T.PRODUCT_ID=TEA_POT_T.PRODUCT_ID)
            LEFT JOIN VENDOR_T ON (TEA_POT_T.VENDOR_ID=VENDOR_T.VENDOR_ID)
WHERE INVENTORY_T.QTY=0
ORDER BY TEA_POT_T.PRODUCT_ID ASC;
```

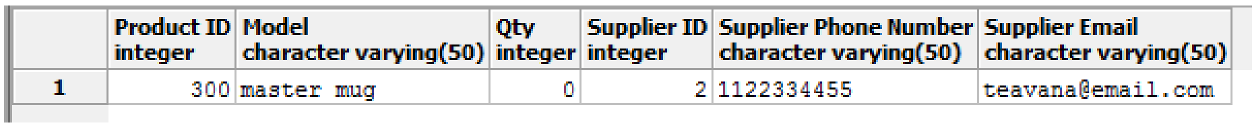


	Product ID integer	Model character varying(50)	Qty integer	Supplier ID integer	Supplier Phone Number character varying(50)	Supplier Email character varying(50)
1	260	japanese pot	0	1	1122334455	adaigo@email.com

view_mug_out_stock

View mugs that not in stock and also display the suppliers contact information

```
CREATE VIEW view_mug_out_stock
AS
SELECT MUG_T.PRODUCT_ID AS "Product ID",
            MUG_T.MUG_MODEL_ID AS "Model",
            INVENTORY_T.QTY AS "Qty",
            MUG_T.VENDOR_ID AS "Supplier ID",
            VENDOR_T.PHONE_NUM AS "Supplier Phone Number",
            VENDOR_T.Email AS "Supplier Email"
FROM MUG_T
            LEFT JOIN  INVENTORY_T ON
(INVENTORY_T.PRODUCT_ID=MUG_T.PRODUCT_ID)
            LEFT JOIN VENDOR_T ON (MUG_T.VENDOR_ID=VENDOR_T.VENDOR_ID)
WHERE INVENTORY_T.QTY=0
ORDER BY MUG_T.PRODUCT_ID ASC;
```

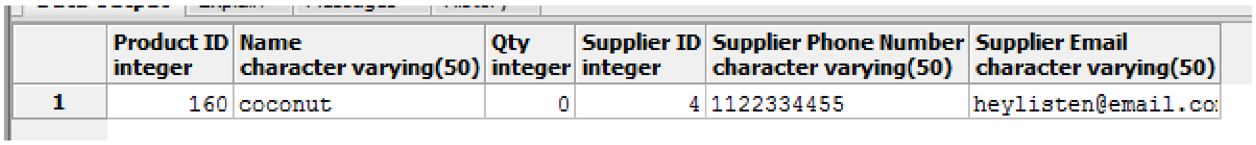


	Product ID integer	Model character varying(50)	Qty integer	Supplier ID integer	Supplier Phone Number character varying(50)	Supplier Email character varying(50)
1	300	master mug	0	2	1122334455	teavana@email.com

view_flavor_out_stock

View flavors that not in stock and also display the suppliers contact information

```
CREATE VIEW view_flavor_out_stock
AS
SELECT FLAVOR_T.PRODUCT_ID AS "Product ID",
		FLAVOR_T.FLAV_NAME_ID AS "Name",
		INVENTORY_T.QTY AS "Qty",
		FLAVOR_T.VENDOR_ID AS "Supplier ID",
		VENDOR_T.PHONE_NUM AS "Supplier Phone Number",
		VENDOR_T.Email AS "Supplier Email"
FROM FLAVOR_T
		LEFT JOIN  INVENTORY_T ON
(INVENTORY_T.PRODUCT_ID=FLAVOR_T.PRODUCT_ID)
		LEFT JOIN VENDOR_T ON (FLAVOR_T.VENDOR_ID=VENDOR_T.VENDOR_ID)
WHERE INVENTORY_T.QTY=0
ORDER BY FLAVOR_T.PRODUCT_ID ASC;
```

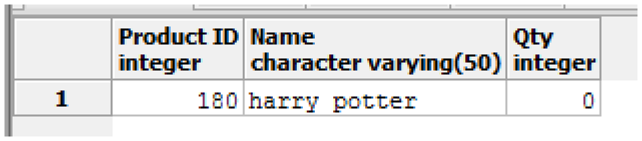


	Product ID integer	Name character varying(50)	Qty integer	Supplier ID integer	Supplier Phone Number character varying(50)	Supplier Email character varying(50)
1	160	coconut	0	4	1122334455	heylisten@email.co

View_tea_blend_out_stock

View tea blends that not in stock. This doesn't show contact info, because tea blends are supposed to be custom made in the store its self.

```
CREATE VIEW view_tea_blend_out_stock
AS
SELECT TEA_BLEND_T.PRODUCT_ID AS "Product ID",
			TEA_BLEND_T.BLEND_NAME_ID AS "Name",
			INVENTORY_T.QTY AS "Qty"
FROM TEA_BLEND_T
			LEFT JOIN  INVENTORY_T ON
(INVENTORY_T.PRODUCT_ID=TEA_BLEND_T.PRODUCT_ID)
WHERE INVENTORY_T.QTY=0
ORDER BY TEA_BLEND_T.PRODUCT_ID ASC;
```



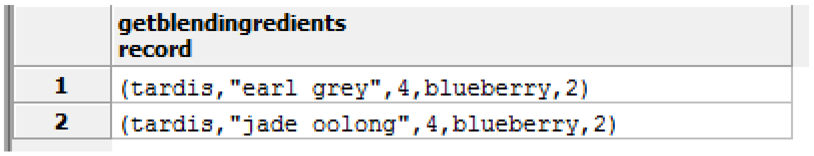
	Product ID integer	Name character varying(50)	Qty integer
1	180	harry potter	0

Stored Procedures and Triggers

getblendingredients

Get the ingredients for a tea blend where the parameter is the name of the tea blend. That way when a blend has run out, the ingredients for making more can easily be found.

```
CREATE FUNCTION getblendingredients(blendname TEXT)
returns table ("Blend Name" varchar(10), "Tea" varchar(10), "Tea QTY" int, "Flavoring"
varchar(10), "Flavor QTY" int) AS $$
BEGIN
RETURN QUERY SELECT TEA_BLEND_T.BLEND_NAME_ID AS "Blend Name",
                    TEA_MIX_T.TEA_NAME_ID AS "Tea",
                    TEA_MIX_T.TEA_QTY_OZ AS "Tea QTY",
                    FLAVOR_MIX_T.FLAV_NAME_ID AS "Flavoring",
                    FLAVOR_MIX_T.FLAVOR_QTY_OZ AS "Flavoring QTY"
          FROM TEA_BLEND_T LEFT JOIN TEA_MIX_T ON
(TEA_BLEND_T.BLEND_NAME_ID=TEA_MIX_T.BLEND_NAME_ID)
                              LEFT JOIN FLAVOR_MIX_T ON
(TEA_MIX_T.BLEND_NAME_ID=FLAVOR_MIX_T.BLEND_NAME_ID)
          WHERE TEA_BLEND_T.BLEND_NAME_ID=blendname;
END;
$$ LANGUAGE plpgsql;
```

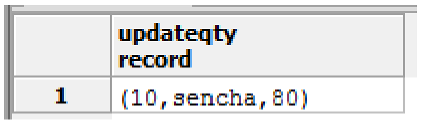


	getblendingredients record
1	(tardis,"earl grey",4,blueberry,2)
2	(tardis,"jade oolong",4,blueberry,2)

updateQTY

Update the qty of an item in the INVENTORY_T. Parameter is the number of units sold. This keeps track of an items stock.

```
CREATE FUNCTION updateQTY(productID INT, sold INT)
returns table ("Product ID" int, "Product Name" varchar(20),"New Amount" int) AS $$
BEGIN
IF sold < 0 then
        RAISE NOTICE 'Number Sold may NOT be negative';
ELSE
UPDATE INVENTORY_T
                        SET QTY= QTY - sold
                        WHERE INVENTORY_T.PRODUCT_ID= productID;
RETURN QUERY SELECT INVENTORY_T.PRODUCT_ID AS "Product ID",
                                        INVENTORY_T.NAME_ID AS "Product Name",
                                        INVENTORY_T.QTY AS "New Amount"
                                FROM INVENTORY_T
                                WHERE INVENTORY_T.PRODUCT_ID=productID;
END IF;
END;
$$ LANGUAGE plpgsql;
```

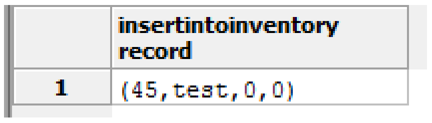


	updateqty record
1	(10,sencha,80)

insertIntoInventory

Adds a new record to the INVENTORY_T table. It makes adding new items easier. Returns the values that were inputted.

```
CREATE FUNCTION insertIntoInventory(productID INT, productName VARCHAR(20), amount int, price int)
returns table ("Product ID" int, "Product Name" varchar(20),"QTY" int, "Price" int) AS $$
BEGIN
INSERT INTO INVENTORY_T(PRODUCT_ID, NAME_ID, QTY, PRICEUSD)
                                    VALUES (productID, productName, amount, price);
RETURN QUERY SELECT INVENTORY_T.PRODUCT_ID AS "Product ID",
                                             INVENTORY_T.NAME_ID AS "Product Name",
                                             INVENTORY_T.QTY AS "QTY",
                                             INVENTORY_T.PTICEUSD AS "PriceUSD"
                                    FROM INVENTORY_T
                                    WHERE INVENTORY_T.PRODUCT_ID=productID;
END;
$$ LANGUAGE plpgsql;
```



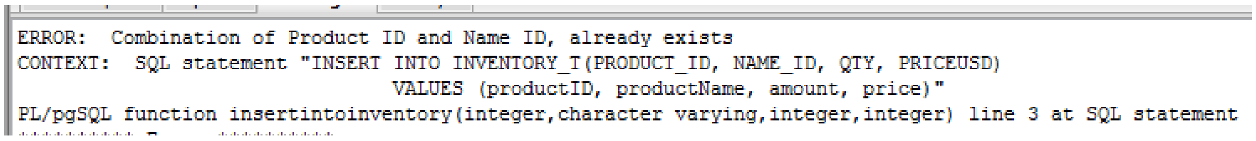
	insertintoinventory record
1	(45,test,0,0)

message and check_insert

Checks to see if then combination of Product ID and Name ID, already exists, before it is added to the INVENTORY_T. If it does, it displays and error message.

```
CREATE OR REPLACE FUNCTION message()
returns trigger AS $message$
BEGIN
	IF EXISTS (SELECT PRODUCT_ID, NAME_ID FROM INVENTORY_T WHERE
PRODUCT_ID=NEW.PRODUCT_ID AND NAME_ID=NEW.NAME_ID) THEN
	RAISE EXCEPTION 'Combination of Product ID and Name ID, already exists';
	END IF;
RETURN NEW;
END;
$message$
 LANGUAGE plpgsql;

--Execute Function 'message()' before a new item is added to the INVENTORY_T table.
CREATE TRIGGER check_insert BEFORE INSERT
ON INVENTORY_T
 FOR EACH ROW  EXECUTE PROCEDURE message();
```



```
ERROR:  Combination of Product ID and Name ID, already exists
CONTEXT:  SQL statement "INSERT INTO INVENTORY_T(PRODUCT_ID, NAME_ID, QTY, PRICEUSD)
                              VALUES (productID, productName, amount, price)"
PL/pgSQL function insertintoinventory(integer,character varying,integer,integer) line 3 at SQL statement
```

Security

Create three different roles; admin, employee, and user. Admin has the power to do anything to the database, including deleting tables. Employees have the ability to select, insert, and update tables. And user only has the ability to do select queries.

```
CREATE ROLE admin WITH LOGIN PASSWORD 'alpaca';
GRANT SELECT, INSERT, UPDATE, DELETE
ON ALL TABLES IN SCHEMA PUBLIC
TO admin;

CREATE ROLE employee WITH LOGIN PASSWORD 'password';
GRANT SELECT, INSERT, UPDATE
ON ALL TABLES IN SCHEMA PUBLIC
TO employee;

CREATE ROLE guest LOGIN;
GRANT SELECT
ON  TEA_T, FLAVOR_T, TEA_BLEND_T, TEA_SET_T, TEA_POT_T, MUG_T
TO guest;
```

Known Problems and Future Enhancements

One problem that quickly becomes apparent, is that, due to the way the database structured, you have to add new items to the inventory fist, before you can add that product to its respective table. This may be counter-intuitive to some employees. Another problem are delete anomalies. If an item is deleted in the INVENTORY_T table, then it is deleted in every table that it is mentioned in.

In term of enhancements, I'd like to implement a way that can figure out all possible combinations of teas and flavors to make new tea blends to sell. Another enhancement would be the ability to track what teas are popular during which seasons, thereby, allowing the store to better manage their orders.