

# NORTH ROAD FLORALS

## Relational Database

Designed by Kathy Coomes

For

Database Management – Alan Labouseur

Spring 2017 – due 05/02/2017

## Table of Contents

Summary	4
Entity Relationship Diagram	5
Types: phone, prod, pos	6
Tables	
People table	7
Vendors table	8
VendorContacts table	9
Staff table	10
Customers table	11
Addresses table	12
Products table	13
ProductHistory table	14
Arrangements table	15
ArrangementItems	16
ArrangementItemsList	17
Inventory table	18
Orders table	19
Stored Procedures (also see Triggers)	
GetArrListofItems	20
GetProductOrderList	21
AddPeople	22 - 24
AddVendors (also see Reports)	25
AddProducts (also see Reports)	26 – 27
FindAlan (created by Daniel Verite – shared on Stack Overflow)	28 - 29
Triggers	
ValidPeopleInput   with Stored Procedure   ValidatePeopleInput()	30
ValidVendorInput   with Stored Procedure   ValidateVendorInput()	31

## Table of Contents

### Views – (each with their own reports)

VendorInfo	32
Select all	
Find vendor name by entering something LIKE the name	
StaffInfo	33 - 34
Select all	
Find an employee by entering something LIKE the first name	
List all current Staff of North Road Florals	
List any staff that has left North Road Florals	
CustomerInfo	35
Select all	
Find all Customers by entering something LIKE a city	
ProductInfo	36 - 37
Select all	
List current and history costs of Products bought	

### Reports

To list everything included in every table	38
Arrangement Item List	38
Vendors with no Vendor Contact	38
Vendors with no Products	39
Vendor from which we order the most	40
Find a Vendor city	40
Examples for reports from Stored Procedure AddVendors	41
Examples for reports from Stored Procedure AddProducts	41

Roles and Security	42
--------------------	----

Implementation Notes	43
----------------------	----

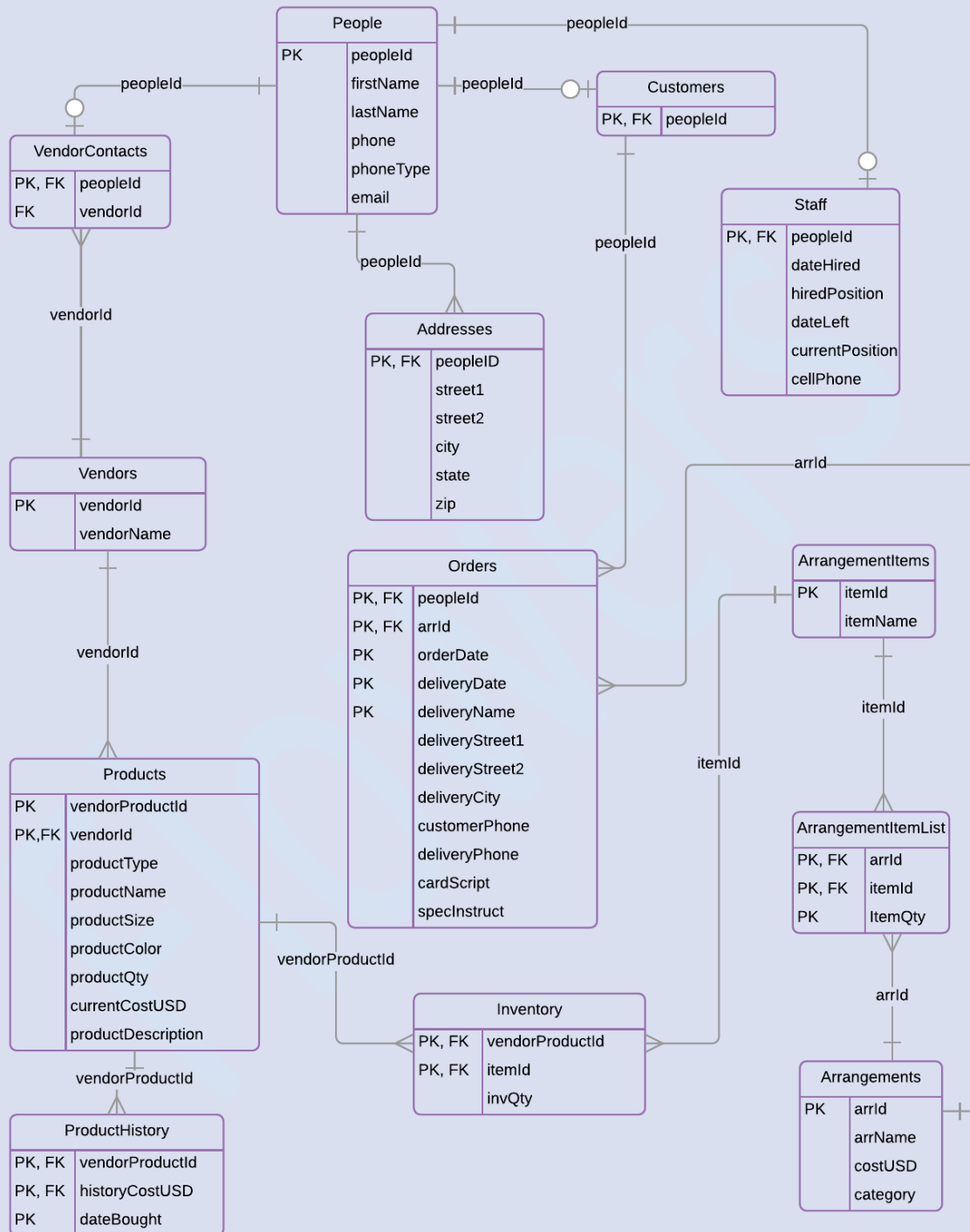
Known Problems and Future Enhancements	44
--	----

## Summary

North Road Florals is a large floral business that has been steadily growing. The company has found that using spreadsheets, as they have, is no longer the most accurate, fastest and easiest way to handle all the information it needs to track about the people involved in or with the company – (Staff, Customers, and Vendor Contacts along with all their information). I was contacted to create a relational database for the company. They also wanted to track the history of products bought - including the current and past history of prices. Having spoken with the owner, arrangers, manager, and sales clerks many times, I have designed such a database in PostgreSQL.

While speaking with the staff of the company, I found that they also wanted the ability when an arrangement was ordered to check the list of items needed for that arrangement against inventory giving them a list of products needed to be ordered. I was able to create a stored procedure for them where all they need do is to enter the arrangement ID and they will receive a list of products to order.

## Entity Relationship Diagram for North Road Florals



## Types

**phone Type** - the phone type is used in the People table

```
CREATE TYPE phone as ENUM ('work', 'home', 'cell');
```

-----

**product Type** - used in the Product Table

```
CREATE TYPE prod as ENUM ('flower', 'container', 'greenery', 'balloon', 'other');
```

-----

**position TYPE** - used in the Staff Table for hiredPosition and position

```
CREATE TYPE pos as ENUM ('owner', 'arranger', 'manager', 'salesclerk');
```

## Tables

### People

This table keeps track of the basic information that VendorContacts, Customers and Staff have in common.

```
CREATE TABLE People(
    peopleId    text    not null    unique,
    firstName   text    not null,
    lastName    text    not null,
    phone       text,
    phoneType   text,
    email       text,
    primary key(peopleId)
);
```

### Functional Dependencies:

peopleId → firstName, lastName, phone, phoneType, email

### Sample Data:

	peopleId text	firstName text	lastName text	phone text	phonetype text	email text	
<input type="checkbox"/>	people001	John	Dolan	800-827-3665	work	floralsupply.com	
<input type="checkbox"/>	people002	Michael	Growski	800-773-2554	work	directfloral.com	
<input type="checkbox"/>	people003	Jessica	Murray	877-701-7673 - ext 5465	work	globalrose.com	
<input type="checkbox"/>	people004	Jon	Sitzer	845-555-5555	home	jon.sitzer34@yahoo.com	
<input type="checkbox"/>	people005	Marisa	Sumter	845-666-6666	cell	[null]	
<input type="checkbox"/>	people006	Mandy	Mishra	845-777-7777	home	mmishra456@gmail.com	
<input type="checkbox"/>	people011	Gary	Carney	845-222-2222	home	gary.carney@net10.com	
<input type="checkbox"/>	people008	Janice	Jones	845-111-1111	home	janicejones58@yahoo.com	
<input type="checkbox"/>	people009	Jane	Doe	845-555-2323	cell	Jd2456@gmail.com	
<input type="checkbox"/>	people010	Mary	Marist	914-555-1212	cell	mmarist6486@marist.edu	
<input type="checkbox"/>	people007	Alan	Labouse...	845-440-1102	work	alan@labouseur.com	
<input type="checkbox"/>	people012	Candie	Kane	846-555-4545	home	candie.kane22@outlook.com	

## Tables

### Vendors

This table tracks Vendors from which the company buys its products.

It creates a many to many relationship between VendorContacts and Products.

```
CREATE TABLE Vendors(
    vendorId          text    not null    unique,
    vendorName        text    not null,
    primary key(vendorId)
);
```

### Functional Dependencies:

vendorId → vendorName

### Sample Data:

<input type="checkbox"/>	vendorid text	vendorname text
<input type="checkbox"/>	vendor001	Floral Supply Wholesale
<input type="checkbox"/>	vendor002	DirectFloral.com
<input type="checkbox"/>	vendor003	GlobalRose.com
<input type="checkbox"/>	vendor004	Flowers by Alan



## Tables

### VendorContacts

This table is an extension of the People table for Vendor Contact Information.

```
CREATE TABLE VendorContacts (
    peopleId    text    not null    references People(peopleId),
    vendorId    text    not null    references Vendors(vendorId),
    primary key(peopleId, vendorId)
);
```

### Functional Dependencies:

peopleId → vendorId

### Sample Data:

<input type="checkbox"/>	peopleId text	vendorId text	
<input type="checkbox"/>	people001	vendor001	
<input type="checkbox"/>	people002	vendor002	
<input type="checkbox"/>	people003	vendor003	

## Tables

### Staff

This table is an extension of the People table for Staff Information.

```
CREATE TABLE Staff (
    peopleId      text    not null    references People(peopleId),
    dateHired     text    not null,
    hiredPosition pos    not null,
    dateLeft      text    DEFAULT NULL,
    position      pos    not null,
    cellPhone     text,
    primary key(peopleId)
);
```

### Functional Dependencies:

peopleId → dateHired, hiredPosition, dateLeft, position, cellphone

### Sample Data:

	peopleId text	dateHired date	hiredPosition text	dateLeft date	position text	cellphone text	
<input type="checkbox"/>	people004	2007-05-12	sales clerk	[null]	manager	914-555-5555	
<input type="checkbox"/>	people005	2015-02-23	sales clerk	[null]	sales clerk	914-666-6666	
<input type="checkbox"/>	people006	2016-01-01	sales clerk	[null]	arranger	914-777-7777	
<input type="checkbox"/>	people011	2007-05-12	owner	[null]	owner-arranger	914-222-2222	
<input type="checkbox"/>	people008	2007-05-12	sales clerk	2007-07-30	sales clerk	914-111-1111	

## Tables

---

### Customers

This table is an extension of the People Table for Customer Information.

---

```
CREATE TABLE Customers (
    peopleid text not null references People(peopleid),
    primary key(peopleid)
);
```

### Functional Dependencies:

peopleid

### Sample Data:

<input type="checkbox"/>	peopleid text	
<input type="checkbox"/>	people009	
<input type="checkbox"/>	people010	
<input type="checkbox"/>	people007	
<input type="checkbox"/>	people012	

## Tables

### Addresses

This table keeps track of the addresses of all People.

```
CREATE TABLE Addresses (
    peopleId    text    not null    references People(peopleId),
    street1     text,
    street2     text,
    zip         text    not null    references Zips(zip),
    primary key(peopleId)
);
```

### Functional Dependencies:

peopleId → street1, street2, zip

### Sample Data:

Data Output		Explain	Messages	History	
<input type="checkbox"/>	peopleid text	street1 text	street2 text	zip text	
<input type="checkbox"/>	people001	[null]	15 Applewood Drive	84037	
<input type="checkbox"/>	people002	[null]	760 Killian Road	44319	
<input type="checkbox"/>	people003	[null]	7225 NW 25th Street	33122	
<input type="checkbox"/>	people004	[null]	456 Mantle Circle	12601	
<input type="checkbox"/>	people005	[null]	23 Alamo Road	12572-1234	
<input type="checkbox"/>	people006	Royal Crest Apartments	22B Royal Crest Place	12538-2256	
<input type="checkbox"/>	people007	Park Manor Apartments	4568 Springwood Drive	12538-4568	
<input type="checkbox"/>	people008	[null]	45 Zachary Way	12602-4545	
<input type="checkbox"/>	people009	Cherry Condominiums	21 Cherry Hill Road	12571	
<input type="checkbox"/>	people010	[null]	62 Hilltop Road	12573	
<input type="checkbox"/>	people011	[null]	86B Church Street	12602	
<input type="checkbox"/>	people012	[null]	91-28B Main Street	12508-1928	

## Tables

### Products

This table tracks current product information.

```
CREATE TABLE Products (
    vendorProductId    text    not null    unique,
    vendorId           text    not null    references Vendors(vendorId),
    productType        prod,
    productName        text,
    productSize        text,
    productColor       text,
    productQty         int    CHECK (productQty >= 0),
    -- # of items in currentCostUSD, might be sold singly, or by the dozen
    currentCostUSD     numeric(6,2) CHECK (currentCostUSD > 0),
    productDesc        text,
    primary key(vendorProductId, vendorId)
);
```

### Functional Dependencies:

vendorProductId, vendorID → productType, productName, productSize, productColor,  
productQty, currentCostUSD, productDesc

### Sample Data:

	vendorprod... text	vendorid text	productty... text	productname text	productsize text	productc... text	productq... integer	currentco... numeric (...)	productdesc text	
<input type="checkbox"/>	G10026B2	vendor003	flower	rose	long stem	lavender	12	7.00	[null]	
<input type="checkbox"/>	C56736	vendor001	container	country metal pail	6 inches tall	off white	5	12.00	[null]	
<input type="checkbox"/>	N674DDA	vendor002	flower	alstroemeria	[null]	white	6	4.50	[null]	
<input type="checkbox"/>	N673GST	vendor002	flower	statice	[null]	white	6	3.00	[null]	
<input type="checkbox"/>	N675GHF	vendor002	flower	daisy	[null]	yellow	12	12.00	[null]	
<input type="checkbox"/>	S97854	vendor001	other	floral foam	9in long x 4.5in wide x 3in...	green	48	25.00	dry brick	
<input type="checkbox"/>	S97657	vendor001	other	floral tape	110yds long x 1in wide	green	6	6.75	rolls	
<input type="checkbox"/>	C56765	vendor001	container	its a boy wagon	9in long x 4in wide x 4in h...	red	1	9.99	good for plant or arrangem...	
<input type="checkbox"/>	C56766	vendor001	container	its a girl wagon	9in long x 4in wide x 4in h...	red	1	9.99	good for plant or arrangem...	
<input type="checkbox"/>	G10027C3	vendor003	flower	rose	spray	yellow	8	15.00	3 - 5 roses on each spray	
<input type="checkbox"/>	N67454C3	vendor003	flower	gerbera	mini	red	12	6.00	[null]	
<input type="checkbox"/>	N67454G5	vendor003	flower	aster	[null]	red	6	9.00	[null]	
<input type="checkbox"/>	N675GHG	vendor002	flower	daisy	[null]	white	12	12.00	[null]	
<input type="checkbox"/>	G87463	vendor001	greenery	leather leaf	[null]	green	100	76.00	stems	
<input type="checkbox"/>	G87465	vendor001	greenery	huckleberry	[null]	green	1	15.75	bunch	

## Tables

### ProductHistory

This table tracks the history of products: ID, cost when bought, date bought.

```
CREATE TABLE ProductHistory (
    vendorProductId text references Products(vendorProductId),
    historyCostUSD numeric(6,2) not null CHECK (historyCostUSD > 0),
    dateBought text not null,
    primary key(vendorProductId, historyCostUSD, dateBought)
);
```

### Functional Dependencies:

vendorProductId → historyCostUSD, dateBought

### Sample Data:

	vendorproductid text	historycos... numeric (...)	datebought date
<input type="checkbox"/>	C56736	12.00	2016-02-15
<input type="checkbox"/>	C56736	12.00	2016-12-02
<input type="checkbox"/>	C56765	9.99	2017-04-12
<input type="checkbox"/>	C56766	9.99	2017-04-12
<input type="checkbox"/>	C78457	12.00	2016-02-02
<input type="checkbox"/>	C86397	15.00	2016-05-11
<input type="checkbox"/>	C97485	6.00	2016-02-02
<input type="checkbox"/>	F87467G5	6.00	2016-05-11
<input type="checkbox"/>	G10026B2	6.50	2016-08-11
<input type="checkbox"/>	G10026B2	6.75	2016-11-11
<input type="checkbox"/>	G10026B2	10.00	2017-04-12
<input type="checkbox"/>	G10026B2	7.00	2017-04-12
<input type="checkbox"/>	G10026BG	10.00	2016-02-02
<input type="checkbox"/>	G10027C3	13.00	2016-05-11
<input type="checkbox"/>	G10027C3	15.00	2016-09-02

## Tables

### Arrangements

This table tracks arrangements: ID, name, cost, and category.  
Category can be any holiday, birthday, birth of a child, thank you, etc.

```
CREATE TABLE Arrangements(
    arrId      text          not null    unique,
    arrName    text          not null,
    arrCostUSD numeric      (6,2) not null CHECK (arrCostUSD > 0),
    category   text          not null,
    primary key(arrId)
);
```

### Functional Dependencies:

arrId → arrName, arrCostUSD, category

### Sample Data:

<input type="checkbox"/>	arrId text	arrName text	arrCostUSD numeric (6,2)	category text	
<input type="checkbox"/>	arr001	Blooming Pail	50.00	Spring	
<input type="checkbox"/>	arr002	Wow Wagon - boy	43.00	baby boy	
<input type="checkbox"/>	arr003	Wow Wagon - girl	43.00	baby girl	
<input type="checkbox"/>	arr004	Fly Away Labouseur	53.00	birthday	
<input type="checkbox"/>	arr005	lovely Ladybug Bouquet	40.00	general	

## Tables

### ArrangementItems

This table gives a list of items that can be used in an arrangement.

```
CREATE TABLE ArrangementItems(
    itemId          text  not null    unique,
    itemName  text  not null,
    primary key(itemId)
);
```

### Functional Dependencies:

itemId → itemName

### Sample Data:

<input type="checkbox"/>	itemid text	itemname text
<input type="checkbox"/>	item001	Country Metal Pail by Alan
<input type="checkbox"/>	item002	Lavender Roses
<input type="checkbox"/>	item003	Alstroemeria
<input type="checkbox"/>	item004	Statice
<input type="checkbox"/>	item005	Yellow Daisies
<input type="checkbox"/>	item006	Floral Foam
<input type="checkbox"/>	item007	Floral tape
<input type="checkbox"/>	item008	Its a Box Red Wagon



## Tables

### ArrangementItemsList

This table brings together the arrangement and arrangementItems tables giving the number of each item to be used in each arrangement.

```
CREATE TABLE ArrangementItemsList(
    arrId      text  not null  references Arrangements(arrId),
    itemId     text  not null  references ArrangementItems(itemId),
    itemQty    int   not null  CHECK (itemQty >= 0) ,
    primary key(arrId, itemId, itemQty)
);
```

### Functional Dependencies:

arrId, itemId → itemQty

### Sample Data:

	arrId text	itemId text	itemQty integer
<input type="checkbox"/>	arr001	item001	1
<input type="checkbox"/>	arr001	item002	8
<input type="checkbox"/>	arr001	item003	5
<input type="checkbox"/>	arr001	item004	6
<input type="checkbox"/>	arr001	item005	5
<input type="checkbox"/>	arr001	item006	1
<input type="checkbox"/>	arr001	item007	1
<input type="checkbox"/>	arr001	item015	4
<input type="checkbox"/>	arr002	item005	3
<input type="checkbox"/>	arr002	item006	1
<input type="checkbox"/>	arr002	item007	1

## Tables

### Inventory

This table is a list of how many of each item the company has on hand. It creates a many to many relationship between Products and ArrangementItems.

```
CREATE TABLE Inventory (
    vendorProductId text references Products(vendorProductId),
    itemId          text references ArrangementItems(itemId),
    invQty          int CHECK (invQty >= 0) ,
    primary key(vendorProductId, itemId)
);
```

### Functional Dependencies:

vendorProductId, itemId → invQty

### Sample Data:

	vendorproductid text	itemid text	invqty integer
<input type="checkbox"/>	G10026B2	item002	10
<input type="checkbox"/>	C56736	item001	4
<input type="checkbox"/>	N674DDA	item003	0
<input type="checkbox"/>	N673GST	item004	0
<input type="checkbox"/>	N675GHF	item005	0
<input type="checkbox"/>	S97854	item006	45
<input type="checkbox"/>	S97657	item007	6
<input type="checkbox"/>	C56765	item008	1
<input type="checkbox"/>	C56766	item016	1
<input type="checkbox"/>	G10027C3	item009	6
<input type="checkbox"/>	N67454C3	item010	0
<input type="checkbox"/>	N67454G5	item011	0
<input type="checkbox"/>	N675GHG	item012	6
<input type="checkbox"/>	G87463	item015	52
<input type="checkbox"/>	G87465	item023	0

## Tables

### Orders

This table is a list of orders of arrangements from the Customer.  
It creates a many to many relationship from Customers to Arrangements.

```
CREATE TABLE Orders (
    peopleId          text    references People(peopleId),
    arrId             text    references Arrangements(arrId),
    orderDate         date    CHECK (orderDate <= deliveryDate),
    deliveryDate      date    CHECK (deliveryDate >= orderDate),
    deliveryName      text,
    deliveryStreet1   text,
    deliveryStreet2   text,
    deliveryCity      text,
    customerPhone     text,
    deliveryPhone     text,
    cardScript        text    not null,          -- what to write on card
    specInstruct      text    not null DEFAULT 'none', -- special instructions for delivery
    primary key(peopleId, arrId, orderDate, deliveryDate, deliveryName)
);
```

### Functional Dependencies:

peopleId, arrId, orderDate, deliveryDate, deliveryName → → deliveryStreet1, deliveryStreet2,  
deliveryCity, customerPhone, deliveryPhone, cardScript, specInstruct

Sample Data:

text	text	date	date	text	text	text	text	text	text
people009	arr004	2017-04-21	2017-04-22	Jodi Walker	Park Condos	78 Eastwood Drive	Hyde Park	845-632-5896	9

text	text	text
914-258-3571	Jodi - Have a Wonderful BIRTHday. We love you, Grammy and Poppy	none

## Stored Procedures

**GetArrListofItems** - Stored Procedure - enter arrId

To obtain a list of items and quantities of each for that arrangement

```
CREATE OR REPLACE FUNCTION GetArrListofItems(TEXT, REFCURSOR)
RETURNS refcursor AS
$$
DECLARE
    arrIdEntered TEXT          := $1;
    resultset      REFCURSOR   := $2;
BEGIN
    OPEN resultset FOR
    SELECT a.arrId, p.itemId, p.itemName, apl.itemQty
    FROM Arrangements a INNER JOIN ArrangementItemsList apl ON a.arrId = apl.arrId
        INNER JOIN ArrangementItems p ON apl.itemId = p.itemId
    WHERE a.arrId = arrIdEntered
    ORDER BY p.itemId ASC;
    RETURN resultset;
END;
$$
LANGUAGE plpgsql;
```

```
SELECT GetArrListofItems('arr002', 'results');
FETCH all FROM results;
```

**Sample Data:**

	arrid	itemid	itemname	itemqty
	text	text	text	integer
<input type="checkbox"/>	arr002	item005	Yellow Daisies	3
<input type="checkbox"/>	arr002	item006	Floral Foam	1
<input type="checkbox"/>	arr002	item007	Floral tape	1
<input type="checkbox"/>	arr002	item016	Its a Girl Red Wag...	1
<input type="checkbox"/>	arr002	item009	Yellow Spray Roses	2
<input type="checkbox"/>	arr002	item010	Red Mini Gerbera	3
<input type="checkbox"/>	arr002	item011	Asters	2
<input type="checkbox"/>	arr002	item012	White Daisies	3
<input type="checkbox"/>	arr002	item013	White Button Mums	2
<input type="checkbox"/>	arr002	item014	Soldago	4
<input type="checkbox"/>	arr002	item015	Greenery	4

## Stored Procedures

### GetProductOrderList

A customer has ordered an arrangement - Enter the arrId and this stored procedure compares the items needed for the arrangement against what is in Inventory and gives a list of items that need to be ordered.

```
CREATE OR REPLACE FUNCTION GetProductOrderList(TEXT, REFCURSOR) RETURNS refcursor AS
$$
```

```
DECLARE
```

```
    arrIdEntered TEXT          := $1;
```

```
    resultset      REFCURSOR   := $2;
```

```
BEGIN
```

```
    OPEN resultset FOR
```

```
    SELECT i.itemId, i.vendorProductId, v.vendorName, pe.firstName, pe.lastName,
           pe.phone
```

```
    FROM Inventory i INNER JOIN ArrangementItems ai ON i.itemId = ai.itemId
```

```
           INNER JOIN ArrangementItemsList ail ON ail.itemId = ai.itemId
```

```
           INNER JOIN Arrangements a ON ail.arrId = a.arrId
```

```
           INNER JOIN Products pr ON i.vendorProductId = pr.vendorProductId
```

```
           INNER JOIN Vendors v ON pr.vendorId = v.vendorId
```

```
           INNER JOIN VendorContacts vc ON v.vendorId = vc.vendorId
```

```
           INNER JOIN People pe ON vc.peopleId = pe.peopleId
```

```
    WHERE a.arrId = arrIdEntered and i.invQty <= ail.itemQty
```

```
    ORDER BY p.itemId ASC;
```

```
    RETURN resultset;
```

```
END;
```

```
$$
```

```
LANGUAGE plpgsql;
```

```
SELECT GetProductOrderList('arr002', 'results2');
```

```
FETCH all FROM results2;
```

### Sample Data:

	itemid text	itemqty integer	invqty integer	vendorproductid text	vendorname text	firstname text	lastname text	phone text
<input type="checkbox"/>	item005	3	0	N675GHF	DirectFloral.com	Michael	Growski	800-773-2554
<input type="checkbox"/>	item010	3	0	N67454C3	GlobalRose.com	Jessica	Murray	877-701-7673 - ext 5465
<input type="checkbox"/>	item011	2	0	N67454G5	GlobalRose.com	Jessica	Murray	877-701-7673 - ext 5465
<input type="checkbox"/>	item013	2	0	N697BHG	DirectFloral.com	Michael	Growski	800-773-2554
<input type="checkbox"/>	item014	4	0	N986GKB	DirectFloral.com	Michael	Growski	800-773-2554
<input type="checkbox"/>	item016	1	1	C56766	Floral Supply Wholesale	John	Dolan	800-827-3665

## Stored Procedures

### ----- AddPeople -

select addPeople(enter info with NULL for data not needed);  
 procedure then adds people and address information, and uses IF statements to decide  
 whether to enter Staff or Customer or VendorContact information

### ----- CREATE OR REPLACE FUNCTION AddPeople

(peopleType text, peopleId text, firstName text, lastName text, phone text,  
 phoneType phone, email text, street1 text, street2 text, city text, state text,  
 zip text, dateHired date, hiredPosition pos, dateLeft date, currentPosition pos,  
 cellPhone text, vendorId text)

RETURNS void AS

\$\$

BEGIN

INSERT INTO People

VALUES (peopleId, firstName, lastName, phone, phoneType, email);

INSERT INTO addresses

VALUES(peopleId, street1, street2, city, state, zip);

IF peopleType = 'S' THEN

INSERT INTO Staff

VALUES(peopleId, dateHired, hiredPosition, dateLeft, currentPosition,  
 cellPhone);

END IF;

IF peopleType = 'C' THEN

INSERT INTO Customers

VALUES(peopleId);

END IF;

IF peopleType = 'V' THEN

INSERT INTO VendorContacts

VALUES(peopleId, vendorId);

END IF;

END

\$\$

LANGUAGE plpgsql;

-- TEST DATA for AddPeople - Staff:

Select AddPeople('S','people013', 'Kathy', 'Coomes', '555-555-5555', 'cell',  
 'kathy.coomes@marist.edu', NULL, '19 Church Street', 'Red Hook', 'New York', '12571',  
 '04/30/2017', 'arranger', NULL, 'arranger', '555-555-5555', NULL);  
 select \* from StaffInfo;

## Stored Procedures

	peopleid text	datehired date	hireposition pos	dateleft date	currentposition pos	cellphone text	firstname text	lastname text	phone text	phonetype phone	e
<input type="checkbox"/>	people006	2016-01-01	salesclerk	[null]	arranger	914-777-7777	Mandy	Mishra	845-777-7777	home	r
<input type="checkbox"/>	people008	2007-05-12	salesclerk	2007-07-30	salesclerk	914-111-1111	Janice	Jones	845-111-1111	home	j
<input type="checkbox"/>	people011	2007-05-12	owner	[null]	owner	914-222-2222	Gary	Carney	845-222-2222	home	g
<input type="checkbox"/>	people013	2017-04-30	arranger	[null]	arranger	555-555-5555	Kathy	Coomes	555-555-5555	cell	k

email text	street1 text	street2 text	city text	state text	zip text	
mmishra456@gmail.com	Royal Crest Apartments	22B Royal Crest Place	Hyde Park	New York	12538-2256	▲
janicejones58@yahoo.com	[null]	45 Zachary Way	Poughkeepsie	New York	12602-4545	
gary.carney@net10.com	[null]	86B Church Street	Poughkeepsie	New York	12602	
kathy.coomes@marist.edu	[null]	19 Church Street	Red Hook	New York	12571	

-- TEST DATA for AddPeople - Customers:

```
Select AddPeople('C','people014', 'Jennie', 'Masters', '555-555-5555', 'home',
                'jennie3456@gmail.com',    NULL, '100 Main Street', 'Rhinecliff', 'New York', '12573',
                NULL, NULL, NULL, NULL, NULL, NULL);
```

```
SELECT * FROM CustomerInfo;
```

Data Output   Explain   Messages   History

	peopleid text	firstna... text	lastname text	phone text	phonety... phone	email text	street1 text	s
<input type="checkbox"/>	people007	Alan	Labouseur	845-440-1102	work	alan@labouseur.com	Park Manor Apartments	4
<input type="checkbox"/>	people009	Jane	Doe	845-555-2323	cell	Jd2456@gmail.com	Cherry Condominiums	2
<input type="checkbox"/>	people010	Mary	Marist	914-555-1212	cell	mmarist6486@marist.edu	[null]	6
<input type="checkbox"/>	people012	Candie	Kane	846-555-4545	home	candie.kane22@outlook.com	[null]	9
<input type="checkbox"/>	people014	Jennie	Masters	555-555-5555	home	jennie3456@gmail.com	[null]	1

street2 text	city text	state text	zip text	
4568 Springwood Drive	Hyde P...	New York	12538-4568	▲
21 Cherry Hill Road	Red Hook	New York	12571	
62 Hilltop Road	Rhinecliff	New York	12573	
91-28B Main Street	Beacon	New York	12508-1928	
100 Main Street	Rhinecliff	New York	12573	▼


## Stored Procedures

-- TEST DATA for AddPeople - VendorContacts:

```
Select AddPeople('V', 'people025', 'George', 'Baker', '555-555-5555', 'work',
                'george@directfloral.com', NULL, '3657 Floral Drive', 'Pleasant Valley', 'New York',
                '12569', NULL, NULL, NULL, NULL, NULL, 'vendor002');
SELECT * FROM VendorInfo;
```

Data Output

	vendorid text	vendorname text	peopleid text	firstname text	lastname text	phone text	phonetype phone	e t
<input type="checkbox"/>	vendor001	Floral Supply Wholes...	people001	John	Dolan	800-827-3665	work	fl
<input type="checkbox"/>	vendor002	DirectFloral.com	people002	Michael	Growski	800-773-2554	work	d
<input type="checkbox"/>	vendor002	DirectFloral.com	people025	George	Baker	555-555-5555	work	g
<input type="checkbox"/>	vendor003	GlobalRose.com	people003	Jessica	Murray	877-701-7673 - ext 5465	work	g



email text	street1 text	street2 text	city text	state text	zip text	
floralsupply.com	[null]	15 Applewood Drive	Fruit Heights	Utah	84037	
directfloral.com	[null]	760 Killian Road	Akron	Ohio	44319	
george@directfloral.c...	[null]	3657 Floral Drive	Pleasant Val...	New York	12569	
globalrose.com	[null]	7225 NW 25th St...	Miami	Florida	33122	



## Stored Procedures

### AddVendor -

enter select AddVendor(enter spots with NULL for data not needed)

```
CREATE OR REPLACE FUNCTION AddVendor
    (vendorId text, vendorName text)
RETURNS void AS
$$
BEGIN
    INSERT INTO Vendors
        VALUES (vendorId, vendorName);
END
$$
LANGUAGE plpgsql;
```

-- Test Data:

```
Select AddVendor('vendor005', 'Florist ExtraOrdinary');
Select * FROM Vendors;
```

	vendorid text	vendorname text
<input type="checkbox"/>	vendor001	Floral Supply Wholesale
<input type="checkbox"/>	vendor002	DirectFloral.com
<input type="checkbox"/>	vendor003	GlobalRose.com
<input type="checkbox"/>	vendor004	Flowers by Alan
<input type="checkbox"/>	vendor005	Florist ExtraOrdinary

## Stored Procedures

### ----- **AddProducts -**

adds information to Products, ProductHistory and Inventory  
once a Product order has been received -- updates or inserts new

```
-----
CREATE OR REPLACE FUNCTION AddProducts
    (productNew text, newVendorProductId text, vendorId text, newProductType text,
    newProductName text, newProductSize text, newProductColor text, newProductQty int,
    newCurrentCostUSD numeric, newProductDesc text, historyCostUSD numeric,
    dateBought date, newItemId text, NewItemName text, newInvQty int)
RETURNS void AS $$
BEGIN
    IF productNew = 'Y' THEN
        INSERT INTO Products
            VALUES (newVendorProductId, vendorId, newProductType,
            newProductName, newProductSize, newProductColor, newProductQty,
            newCurrentCostUSD, newProductDesc);
        INSERT INTO ArrangementItems
            VALUES (newItemId, newItemName);
        INSERT INTO Inventory
            VALUES(newVendorProductId, newItemId, newInvQty);
        INSERT INTO ProductHistory
            VALUES(newVendorProductId, historyCostUSD, dateBought);
    END IF;
    IF productNew = 'N' THEN
        UPDATE Products SET
            productType = newProductType,
            productName = newProductName,
            productSize = newProductSize,
            productColor = newProductColor,
            productQty = newProductQty,
            currentCostUSD = newCurrentCostUSD,
            productDesc = newProductDesc
            WHERE vendorProductId = newVendorProductId;
        INSERT INTO ProductHistory
            VALUES(newVendorProductId, historyCostUSD, dateBought);
        UPDATE Inventory SET invQty = invQty + newInvQty
            WHERE vendorproductId = newVendorProductId and itemId =
            newItemId;
    END IF;
END
$$ LANGUAGE plpgsql;
```

## Stored Procedures

-- TEST DATA – Add new product:

```
SELECT AddProducts('Y', 'G10026BH', 'vendor003', 'flower', 'rose', 'long stem',
    'red', 12, 15, NULL, 15, '04/30/2017', 'item034', 'red roses', 12);
SELECT * FROM Products;
```

	vendorproductid text	vendorid text	producttype text	productname text	productsizes text	productcolor text	productqty integer	currentcost numeric (6,2)	productdesc text
<input type="checkbox"/>	C97465	vendor001	balloon	balloon	mylar	silver and mixed	25	12.50	colorful happy birthday
<input type="checkbox"/>	F87467G5	vendor003	flower	carnation	[null]	red	12	6.00	[null]
<input type="checkbox"/>	N986GKB	vendor002	flower	soldago	[null]	white	1	3.00	bunch
<input type="checkbox"/>	C86397	vendor001	container	vase	5in tall	clear with lady bugs	5	15.00	[null]
<input type="checkbox"/>	G10026BH	vendor003	flower	rose	long stem	red	12	15.00	[null]

SELECT \* FROM ProductHistory; SELECT \* from Inventory; SELECT \* FROM ArrangementItems;

a Output Explain Messages History

vendorproductid text	historycost numeric (6,2)	datebought date
C56766	9.99	2017-04-12
C56766	9.99	2017-04-12
S97657	5.75	2017-04-12
S97657	5.75	2017-04-12
S97854	25.00	2017-04-12
S97854	25.00	2017-04-12
G10026BH	15.00	2017-04-30
G10026BH	15.00	2017-04-30

vendorproductid text	itemid text	invqty integer
C56766	item020	21
F87467G5	item020	0
N986GKB	item014	0
C86397	item021	3
G10026BH	item034	12

itemid text	itemname text
item020	Red Carnations
item021	Clear vase with lad...
item022	Green button mums
item023	Huckleberry greens
item034	red roses

-- TEST DATA – Update a product:

```
SELECT AddProducts('N', 'G10026B2', 'vendor003', 'flower', 'rose', 'long stem',
    'lavender', 12, 15, NULL, 15, '04/30/2017', 'item002', 'lavender roses', 12);
SELECT * FROM Products;
```

vendorproductid text	vendorid text	producttype text	productname text	productsizes text	productcolor text	productqty integer	currentcost numeric (6,2)	productdesc text
F87467G5	vendor003	flower	carnation	[null]	red	12	6.00	[null]
N986GKB	vendor002	flower	soldago	[null]	white	1	3.00	bunch
C86397	vendor001	container	vase	5in tall	clear with lady bugs	5	15.00	[null]
G10026BH	vendor003	flower	rose	long stem	red	12	15.00	[null]
G10026B2	vendor003	flower	rose	long stem	lavender	12	15.00	[null]

SELECT \* FROM ProductHistory;

SELECT \* from inventory;

vendorproductid text	historycost numeric (6,2)	datebought date
C56766	9.99	2017-04-12
S97657	5.75	2017-04-12
S97854	25.00	2017-04-12
G10026BH	15.00	2017-04-30
G10026B2	15.00	2017-04-30

Data Output Explain Messages History

vendorproductid text	itemid text	invqty integer
F87467G5	item020	0
N986GKB	item014	0
C86397	item021	3
G10026BH	item034	12
G10026B2	item002	22

## Stored Procedures

-----  
**Find Alan Report** - Finds Alan or Labouseur anywhere in the database

**Created by Daniel Verite and shared on 3 Stack Overflow on 4/12/2014 under the title of 'How to search a specific value in all tables (PostgreSQL)?'** works in version 9.1 or newer

He stated the following:

"Here is a pl/pgsql function that locates records where any column contains a specific value. It takes as arguments the value to search in text format, an array of table names to search into (defaults to all tables) and an array of schema names (public by default. It returns a table structure with schema, name of table, name of column and pseudo-column ctid (non-durable physical location of the row in the table."

-----  
 CREATE OR REPLACE FUNCTION search\_columns(  
     needle text,  
     haystack\_tables name[] default '{}',  
     haystack\_schema name[] default '{public}'  
 )  
 RETURNS table(schemaname text, tablename text, columnname text, rowctid text)  
 AS \$\$  
 begin  
   FOR schemaname,tablename,columnname IN  
     SELECT c.table\_schema,c.table\_name,c.column\_name  
     FROM information\_schema.columns c  
     JOIN information\_schema.tables t ON  
       (t.table\_name=c.table\_name AND t.table\_schema=c.table\_schema)  
     WHERE (c.table\_name=ANY(haystack\_tables) OR haystack\_tables='{}')  
     AND c.table\_schema=ANY(haystack\_schema)  
     AND t.table\_type='BASE TABLE'  
   LOOP  
     EXECUTE format('SELECT ctid FROM %I.%I WHERE cast(%I as text)=%L',  
       schemaname,  
       tablename,  
       columnname,  
       needle  
     ) INTO rowctid;  
     IF rowctid is not null THEN  
       RETURN NEXT;  
     END IF;  
   END LOOP;  
 END;  
 \$\$ language plpgsql;

## Stored Procedures

```
SELECT * FROM search_columns('Alan');
```

<input type="checkbox"/>	schemaname text	tablename text	columnname text	rowctid text	
<input type="checkbox"/>	public	people	firstname	(0,11)	
<input type="checkbox"/>	public	vendors	vendorname	(0,4)	
<input type="checkbox"/>	public	arrangementitems	itemname	(0,1)	

```
SELECT * FROM search_columns('Labouseur');
```

<input type="checkbox"/>	schemaname text	tablename text	columnname text	rowctid text	
<input type="checkbox"/>	public	people	lastname	(0,11)	
<input type="checkbox"/>	public	arrangements	arrname	(0,4)	

## Triggers

### Trigger – **validPeopleInput**

Stored Procedure - **ValidatePeopleInput()**

Triggers on a new entry or update in the People table

```
-----
CREATE OR REPLACE FUNCTION ValidatePeopleInput()
RETURNS TRIGGER AS
$$
BEGIN
    IF NEW.firstName IS NULL THEN
        RAISE EXCEPTION 'firstName may not be NULL';
    END IF;
    IF NEW.lastName IS NULL THEN
        RAISE EXCEPTION 'lastName may not be NULL';
    END IF;
    RETURN NEW;
END
$$
LANGUAGE plpgsql;
-----
```

```
-----
CREATE TRIGGER validPeopleInput
BEFORE INSERT OR UPDATE ON People
FOR EACH ROW
EXECUTE PROCEDURE ValidatePeopleInput();
-----
```

### Test Data

```
INSERT INTO People (peopleId, firstName, lastName)
VALUES('people013', NULL, NULL);
INSERT INTO People (peopleId, firstName, lastName)
VALUES('people013' Joyce, NULL);
```

### Results:

ERROR: firstName may not be NULL  
 CONTEXT: PL/pgSQL function validatepeopleinput() line 4 at RAISE

ERROR: lastName may not be NULL  
 CONTEXT: PL/pgSQL function validatepeopleinput() line 7 at RAISE

## Triggers

-----  
 Trigger - **ValidVendorInput**

Stored Procedure - **ValidateVendorInput**

Triggers on insert or update of the Vendor table  
 -----

```
CREATE OR REPLACE FUNCTION ValidateVendorInput()
RETURNS TRIGGER AS
$$
BEGIN
    IF NEW.vendorId IS NULL THEN
        RAISE EXCEPTION 'vendorId may not be NULL';
    END IF;
    IF NEW.vendorName IS NULL THEN
        RAISE EXCEPTION 'vendorName may not be NULL';
    END IF;
    RETURN NEW;
END
$$
LANGUAGE plpgsql;
```

-----  
 CREATE TRIGGER ValidVendorInput  
 BEFORE INSERT OR UPDATE ON Vendors  
 FOR EACH ROW  
 EXECUTE PROCEDURE ValidateVendorInput();  
 -----

### Test Data

```
INSERT INTO Vendors (vendorId, vendorName)
VALUES(NULL, 'LabouseursPlace');
INSERT INTO Vendors (vendorId, vendorName)
VALUES('vendor234', NULL);
```

### Results:

```
ERROR: vendorId may not be NULL
CONTEXT: PL/pgSQL function validatevendorinput() line 4 at RAISE
ERROR: vendorName may not be NULL
CONTEXT: PL/pgSQL function validatevendorinput() line 7 at RAISE
```

## Views

### VendorInfo – A view with 2 reports to obtain Vendor Information

from People, Addresses, Zips, Vendors, VendorContacts

```
CREATE VIEW VendorInfo
```

```
AS
```

```
SELECT v.vendorId, v.vendorName, vc.peopleId, p.firstName, p.lastName, p.phone,
       p.phoneType, p.email, a.street1, a.street2, a.zip, z.city, z.state
```

```
FROM Vendors v,
```

```
     VendorContacts vc,
```

```
     People p,
```

```
     Addresses a,
```

```
     Zips z
```

```
WHERE v.vendorId = vc.vendorId and
```

```
      vc.peopleId = p.peopleId and
```

```
      p.peopleId = a.peopleId and
```

```
      a.zip = z.zip
```

```
ORDER BY v.vendorId ASC;
```

#### -- -----query 1 - all Vendor Information -----

```
SELECT * FROM VendorInfo;
```

#### Sample Data:

vendorId text	vendorname text	peopleId text	firstna... text	lastna... text	phone text	phon... phone	email text	street1 text	street2 text	city text	state text	zip text
vendor001	Floral Supply ...	people001	John	Dolan	800-827...	work	floralsupply.com	[null]	15 Applewood ...	Fruit Heig...	Utah	84037
vendor002	DirectFloral.com	people002	Michael	Growski	800-773...	work	directfloral.com	[null]	760 Killian Road	Akron	Ohio	44319
vendor002	DirectFloral.com	people025	George	Baker	555-555...	work	george@directfl...	[null]	3657 Floral Drive	Pleasant V...	New York	12569
vendor003	GlobalRose.com	people003	Jessica	Murray	877-701...	work	globalrose.com	[null]	7225 NW 25th ...	Miami	Florida	33122

#### -- -----query 2 - search for an unknown name -----

```
SELECT vendorId, vendorName, firstName, LastName, phone, email
```

```
FROM VendorInfo
```

```
WHERE vendorName Like '%Rose%';
```

#### Sample Data:

vendorId text	vendorname text	firstName text	lastName text	phone text	email text
vendor003	GlobalRose.com	Jessica	Murray	877-701-7673 - ext 5465	globalrose.com



## Views

### StaffInfo – A view with 4 reports to obtain Vendor Information

from People, Addresses, Zips, Staff

```

CREATE VIEW StaffInfo
AS
SELECT s.peopleId, s.dateHired, s.hiredPosition, s.dateLeft, s.position, s.cellPhone,
       p.firstName, p.lastName, p.phone, p.phoneType, p.email, a.street1, a.street2,
       a.zip, z.city, z.state
FROM Staff s,
     People p,
     Addresses a,
     Zips z
WHERE s.peopleId = p.peopleId and
      p.peopleId = a.peopleId and
      a.zip = z.zip
ORDER BY s.peopleId ASC;

```

-- ---- query 1 - all Staff Information -----

```
SELECT * FROM StaffInfo;
```

### Sample Data:

peopleId text	dateHired date	hiredPosition text	dateLeft date	position text	cellphone text	firstName text	lastName text	phone text	phonetype text	e t
people004	2007-05-12	sales clerk	[null]	manager	914-555-5555	Jon	Sitzer	845-555-5555	home	j
people005	2015-02-23	sales clerk	[null]	sales clerk	914-666-6666	Marisa	Sumter	845-666-6666	cell	[
people006	2016-01-01	sales clerk	[null]	arranger	914-777-7777	Mandy	Mishra	845-777-7777	home	n
people008	2007-05-12	sales clerk	2007-07-30	sales clerk	914-111-1111	Janice	Jones	845-111-1111	home	j
people011	2007-05-12	owner-arran...	[null]	owner-a...	914-222-2222	Gary	Carney	845-222-2222	home	g

email text	street1 text	street2 text	zip text	city text	state text
jon.sitzer34@yahoo.com	[null]	456 Mantle Circle	12601	Poughkeepsie	New York
[null]	[null]	23 Alamo Road	12572-1234	Rhinebeck	New York
mmishra456@gmail.com	Royal Crest Apart...	22B Royal Crest Place	12538-2256	Hyde Park	New York
janicejones58@yahoo.c...	[null]	45 Zachary Way	12602-4545	Poughkeepsie	New York
gary.carney@net10.com	[null]	86B Church Street	12602	Poughkeepsie	New York

## Views

-- ---- **query 2. - search for Staff (when you know part of the first name)**

-- list staff info for first names that have the letters 'ma' in it

```
SELECT firstName, LastName, cellPhone, phone, phoneType, email
```

```
FROM StaffInfo
```

```
WHERE firstName Like 'Ma%'
```

```
ORDER BY firstName DESC;
```

### Sample Data:

	firstname text	lastname text	cellphone text	phone text	phonetype text	email text
<input type="checkbox"/>	Marisa	Sumter	914-666-6666	845-666-6666	cell	[null]
<input type="checkbox"/>	Mandy	Mishra	914-777-7777	845-777-7777	home	mmishra456@gmail.com

-- ---- **query 3. - search for current Staff**

-- list current staff

```
SELECT firstName, LastName, dateHired, hiredPosition, position
```

```
FROM StaffInfo
```

```
WHERE dateLeft is Null
```

```
ORDER BY dateHired DESC;
```

### Sample Data:

	firstname text	lastname text	datehired date	hiredposition text	position text
<input type="checkbox"/>	Mandy	Mishra	2016-01-01	sales clerk	arranger
<input type="checkbox"/>	Marisa	Sumter	2015-02-23	sales clerk	sales clerk
<input type="checkbox"/>	Jon	Sitzer	2007-05-12	sales clerk	manager
<input type="checkbox"/>	Gary	Carney	2007-05-12	owner-arranger	owner-arranger

-- ---- **query 4. - search for Staff who have left**

-- list staff who are no longer with the company

```
SELECT firstName, LastName, dateHired, hiredPosition, position
```

```
FROM StaffInfo
```

```
WHERE dateLeft is not Null;
```

### Sample Data:

	firstname text	lastname text	datehired date	hiredposition text	position text
<input type="checkbox"/>	Janice	Jones	2007-05-12	sales clerk	sales clerk

## Views

### CustomerInfo – A view with 4 reports to obtain Customer Information

from People, Addresses, Zips, Customers

```
CREATE VIEW CustomerInfo
```

```
AS
```

```
SELECT c.peopleId, p.firstName, p.lastName, p.phone, p.phoneType, p.email,  
       a.street1, a.street2, a.zip, z.city, z.state
```

```
FROM Customers c,
```

```
     People p,
```

```
     Addresses a,
```

```
     Zips z
```

```
WHERE c.peopleId = p.peopleId and
```

```
      p.peopleId = a.peopleId and
```

```
      a.zip = z.zip
```

```
ORDER BY c.peopleId ASC;
```

-- ---- **query 1 - all Customer Information** -----

```
SELECT * FROM CustomerInfo;
```

#### Sample Data:

	peopleId text	firstName text	lastName text	phone text	phonety... text	email text	street1 text	street2 text	zip text	city text	state text
<input type="checkbox"/>	people007	Alan	Labouseur	845-440-1102	work	alan@labouseur.com	Park Manor Apart...	4568 Springwoo...	12538-4568	Hyde Park	New York
<input type="checkbox"/>	people009	Jane	Doe	845-555-2323	cell	Jd2456@gmail.com	Cherry Condomin...	21 Cherry Hill R...	12571	Red Hook	New York
<input type="checkbox"/>	people010	Mary	Marist	914-555-1212	cell	mmarist6486@mari...	[null]	Park Manor Apartments Apartment 100 Road	12573	Rhinecliff	New York
<input type="checkbox"/>	people012	Candie	Kane	846-555-4545	home	candie.kane22@outl...	[null]	91-28B Main Str...	12508-1928	Beacon	New York

-- ---- **query 2 - By city** -----

-- list customer info for cities that start with 'R'

```
SELECT firstName, LastName, phone, phoneType, email, city
```

```
FROM CUSTOMERINFO
```

```
where city Like 'R%';
```

#### Sample Data:

	firstName text	lastName text	phone text	phonetype text	email text	city text
<input type="checkbox"/>	Jane	Doe	845-555-23...	cell	Jd2456@gmail.com	Red Hook
<input type="checkbox"/>	Mary	Marist	914-555-12...	cell	mmarist6486@marist.edu	Rhinecliff

## Views

### ProductInfo – A view with 2 reports to obtain Product Information

from Vendors, Products, ProductHistory

```
CREATE VIEW ProductInfo
```

```
AS
```

```
SELECT c.vendorId, c.vendorName, h.vendorProductId, p.productType, p.productName,
       p.productSize, p.productColor, p.productQty, p.currentCostUSD, p.productDesc,
       h.dateBought, h.historyCostUSD
```

```
FROM Vendors c,
     Products p,
     ProductHistory h
```

```
WHERE c.vendorId = p.vendorId and
       p.vendorProductId = h.vendorProductId;
```

-- ---- query 1 - all Product Information -----

```
SELECT * from ProductInfo;
```

### Sample Data:

	vendorid text	vendorname text	vendorproductid text	producttype text	productname text	productsizesize text	productcolor text
<input type="checkbox"/>	vendor003	GlobalRose.com	G10026BG	flower	rose	[null]	orange
<input type="checkbox"/>	vendor001	Floral Supply Whole...	C78457	container	vase	7in tall	cobalt blue
<input type="checkbox"/>	vendor003	GlobalRose.com	N67454C3	flower	gerbera	mini	red
<input type="checkbox"/>	vendor001	Floral Supply Whole...	C97485	balloon	balloon	mylar	silver and mixed
<input type="checkbox"/>	vendor001	Floral Supply Whole...	G87465	greenery	huckleberry	[null]	green

	productqty integer	currentcostusd numeric (6,2)	productdesc text	datebought date	historyc... numeri...
	4	10.00	[null]	2016-02-02	10.00
	4	12.00	[null]	2016-02-02	12.00
	12	6.00	[null]	2016-02-02	5.00
	25	12.50	colorful HappyBirthday	2016-02-02	6.00
	1	15.75	bunch	2016-02-02	13.50

## Views

-- ---- query 2 - List of History of cost and current cost for each product -----

```
SELECT vendorProductId, productName, productSize, productQty, currentCostUSD, dateBought,
historyCostUSD
FROM ProductInfo
ORDER BY vendorProductId, dateBought;
```

### Sample Data:

Data Output		Explain	Messages	History			
<input type="checkbox"/>	vendorproductid text	productname text	productsizes text	productqty integer	currentcostusd numeric (6,2)	datebought date	historycostusd numeric (6,2)
<input type="checkbox"/>	C56736	country metal pail	6 inches tall	5	12.00	2016-02-15	12.00
<input type="checkbox"/>	C56736	country metal pail	6 inches tall	5	12.00	2016-12-02	12.00
<input type="checkbox"/>	C56765	its a boy wagon	9in long x 4in wide x 4in high	1	9.99	2017-04-12	9.99
<input type="checkbox"/>	C56766	its a girl wagon	9in long x 4in wide x 4in high	1	9.99	2017-04-12	9.99
<input type="checkbox"/>	C78457	vase	7in tall	4	12.00	2016-02-02	12.00
<input type="checkbox"/>	C86397	vase	5in tall	5	15.00	2016-05-11	15.00
<input type="checkbox"/>	C97485	balloon	mylar	25	12.50	2016-02-02	6.00
<input type="checkbox"/>	F87467G5	carnation	[null]	12	6.00	2016-05-11	6.00
<input type="checkbox"/>	G10026B2	rose	long stem	12	7.00	2016-08-11	6.50
<input type="checkbox"/>	G10026B2	rose	long stem	12	7.00	2016-11-11	6.75
<input type="checkbox"/>	G10026B2	rose	long stem	12	7.00	2017-04-12	10.00

## Reports

Reports that **list all** from each of the tables.

Results were listed with each table create:

```
SELECT * FROM People;
SELECT * FROM Vendors;
SELECT * FROM VendorContacts;
SELECT * FROM Staff;
SELECT * FROM Customers;
SELECT * FROM Zips;
SELECT * FROM Addresses;
SELECT * FROM Products;
SELECT * FROM ProductHistory;
SELECT * FROM Arrangements;
SELECT * FROM ArrangementItems;
SELECT * FROM ArrangementItemsList;
SELECT * FROM Inventory;
```

### List Items used in an Arrangement and their quantity

Report to obtain a list of items needed for an arrangement.

```
SELECT a.arrId, p.itemId, p.itemName, apl.itemQty
FROM Arrangements a INNER JOIN ArrangementItemsList apl ON a.arrId = apl.arrId
INNER JOIN ArrangementItems p ON apl.itemId = p.itemId
WHERE a.arrId = 'arr001'
ORDER BY p.itemId ASC;
```

### Sample Data:

	arrid text	itemid text	itemname text	itemqty integer
<input type="checkbox"/>	arr001	item001	Country Metal Pail	1
<input type="checkbox"/>	arr001	item002	Lavender Roses	8
<input type="checkbox"/>	arr001	item003	Alstroemeria	5
<input type="checkbox"/>	arr001	item004	Statice	6
<input type="checkbox"/>	arr001	item005	Yellow Daisies	5
<input type="checkbox"/>	arr001	item006	Floral Foam	1
<input type="checkbox"/>	arr001	item007	Floral tape	1
<input type="checkbox"/>	arr001	item015	Greenery	4

## Reports

### Vendors with no contact person

Report finds Vendors who have no Vendor Contacts

```
SELECT *
FROM Vendors
WHERE vendorId not in
      (SELECT      vendorId
       FROM VendorContacts
       )
ORDER BY vendorName;
```

Sample Data:

<input type="checkbox"/>	vendorid text	vendorname text
<input type="checkbox"/>	vendor004	Flowers by Alan

### Vendors with no products

Report finds Vendors who we have not ordered from

```
SELECT *
FROM Vendors
WHERE vendorId not in
      (SELECT      vendorId
       FROM Prooducts
       )
ORDER BY vendorName;
```

Sample Data:

<input type="checkbox"/>	vendorid text	vendorname text
<input type="checkbox"/>	vendor004	Flowers by Alan

## Reports

### Vendor Most Used

Report finds Vendor who we order the most from

```
SELECT v.vendorId as "Vendor ID",
       v.vendorName as "Vendor Name",
       count(v.vendorId) as "Product Orders Numbers"
FROM Vendors v INNER JOIN Products p ON v.vendorId = p.vendorId
              INNER JOIN ProductHistory ph ON p.vendorProductId = ph.vendorProductId
GROUP BY v.vendorId
ORDER BY count(v.vendorId) DESC
limit 1;
```

Vendor ID text	Vendor Name text	Product O... bigint
vendor001	Floral Supply Wholesale	20

### Find Vendor City

Finds Vendors in a particular City

```
SELECT v.vendorName as "Vendor Name"
FROM Vendors v
WHERE v.vendorId in
      (SELECT distinct vc.vendorId
       FROM VendorContacts vc
       WHERE vc.peopleId in
            (SELECT p.peopleId
             FROM People p
             WHERE p.peopleId in
                  (SELECT a.peopleId
                   FROM Addresses a
                   WHERE a.city in ('Miami'))
            )
      )
ORDER BY v.vendorName ASC;
```

Vendor Name text
GlobalRose.com



## Reports

-----  
**AddVendors** - Calls the Stored Procedure AddVendors (test data is there)  
 -----

-- example:

-- Select AddVendor('vendor004', 'WonderlandFlorist.com');

-- Use this below - remove the dashes first, replace the information

-- SELECT AddPeople('vendorId', 'vendorName');

-----  
**AddProducts** - Calls the Stored Procedure AddProducts (test data there)  
 -----

--example1:

--SELECT AddProducts('Y', 'G10026BH', 'vendor003', 'flower', 'rose', 'long stem',  
                   'red', 12, 15, NULL, 15, '04/30/2017', 'item034', 'red roses', 12);

--SELECT AddProducts('N', 'G10026B2', 'vendor003', 'flower', 'rose', 'long stem',  
                   'lavender', 12, 15, NULL, 15, '04/30/2017', 'item002', 'lavendsr roses', 12);

-- Use this below - remove the dashes first, replace the information

-- SELECT AddProducts('productNew', 'newVendorProductId', 'vendorId', 'newProductType',  
 -- 'newProductName', 'newProductSize', 'newProductColor', 'newProductQty',  
 -- 'newCurrentCostUSD', 'newProductDesc', 'historyCostUSD', 'dateBought', 'newItemId',  
 -- 'newItemName', 'newInvQty');

-----  
 -- There are more reports attached to all the views and stored procedures  
 -----

## Roles and Security

```

-----
-- Security restart for all roles, tables, views, and grants --
-----

DROP ROLE IF EXISTS admin;
DROP ROLE IF EXISTS owner;
DROP ROLE IF EXISTS manager;
DROP ROLE IF EXISTS salesclerk;

DROP TABLE IF EXISTS People, VendorContacts, Staff, Customers, Vendors, Addresses, Products,
    Producthistory, Inventory, Arrangements, ArrangementItems, ArrangementItemsList
    CASCADE;

DROP VIEW IF EXISTS CompanyInfo, StaffInfo, CustomerInfo, ProductInfo CASCADE;

DROP TYPE IF EXISTS phone, prod, pos CASCADE;

DROP TRIGGER IF EXISTS validPeopleInput ON People CASCADE;
-----
-- Administrator Role - has full access --
-----

CREATE ROLE admin;
GRANT ALL ON ALL TABLES IN SCHEMA public TO admin;
-----
-- Owner Role - Owns the business, so has full access --
-----

CREATE ROLE owner;
GRANT ALL ON ALL TABLES IN SCHEMA public TO owner;
-----
-- Manager Role - the manager has access to insert or update in all tables --
-----

CREATE ROLE manager;
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM manager;
GRANT ALL ON ALL TABLES IN SCHEMA public TO manager;
-----
-- Salesclerk Role - has ability to select all --
-----

CREATE ROLE salesclerk;
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM salesclerk;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO salesclerk;

```

## Implementation Notes

With a much larger data sample (which would take more than the time given and would make this project extremely unwieldy), there would be more chances to create more complex queries.

Also (to keep the project down to size), I have only included validation for input or update into two of the tables: People and Vendors as it would only be repetition.

I have included AddPeople, AddVendors, and AddProducts, but again due to size and time, I have not included AddArrangements (which would add to Arrangements, ArrangementItems, ArrangementItemsList, Inventory, Products and possibly Vendors, VendorContacts, People, and Addresses. When adding, it is required that the information for each be entered in its entirety. This leaves less chance for missing information.

I would like to thank Daniel Verite for the Stored Procedure from Stack Overflow (see my documentation in the sql) that I used to find “Alan” or “Labouseur” anywhere in the database.

Each arrangement uses particular items. As an arrangement is ordered a report can be run to check inventory to obtain a list of products that are need to be ordered. Once products are received, if they are new, they are added to Products, ProductHistory, Inventory, and ArrangementItems. If they are products that we have previously ordered, they are added to ProductHistory, the quantity is added to the Inventory quantity, and items in Products that may have changed are updated. As an arrangement is made, the items used will be subtracted from the Inventory quantity. \*NOTE\* Information in Products is from the last time they were ordered. Information in product history is a list of every time we order the product.

If anyone utilizing this database finds that a report that they would like to have is not available, please contact the administrator who will endeavor to write such a report in a timely fashion.

## Known Problems and Future Enhancements

There are a few things missing and others that should be added, such as:

More validation of all input - When adding information there is no validation to ensure that information fits the criteria for each column in each table. Set up more Types for anything that doesn't have a long list of possibilities.

AddArrangements – a stored procedure for adding arrangements should also be added.

ArrangementInfo - a view for all arrangement information

Financial Information - as the company is currently using QuickBooks to keep its accounts, there was no financial information included in this database other than the cost of an arrangement, and the cost of a product. I would like to add the financials to the database, but it is, of course, up the company.

Missing information – another procedure should be added that finds anything that is NULL in each column in each table.

Update missing, changed or incorrect information – another procedure that changes the information that was missing or incorrect from each column of each table (once that information has been found).

Subtract from Inventory – once an arrangement has been made the items used need to be subtracted from Inventory.

The Orders table, added at the last minute, should also have an OrdersHistory table, an OrdersInfo view and reports.

As this was already much too long, I added the Order table because it was a must.

Florals