NORTH ROAD FLORALS

Relational Database

Designed by Kathy Coomes

For

Database Management – Alan Labouseur

Spring 2017 – due 05/02/2017

Table of Contents

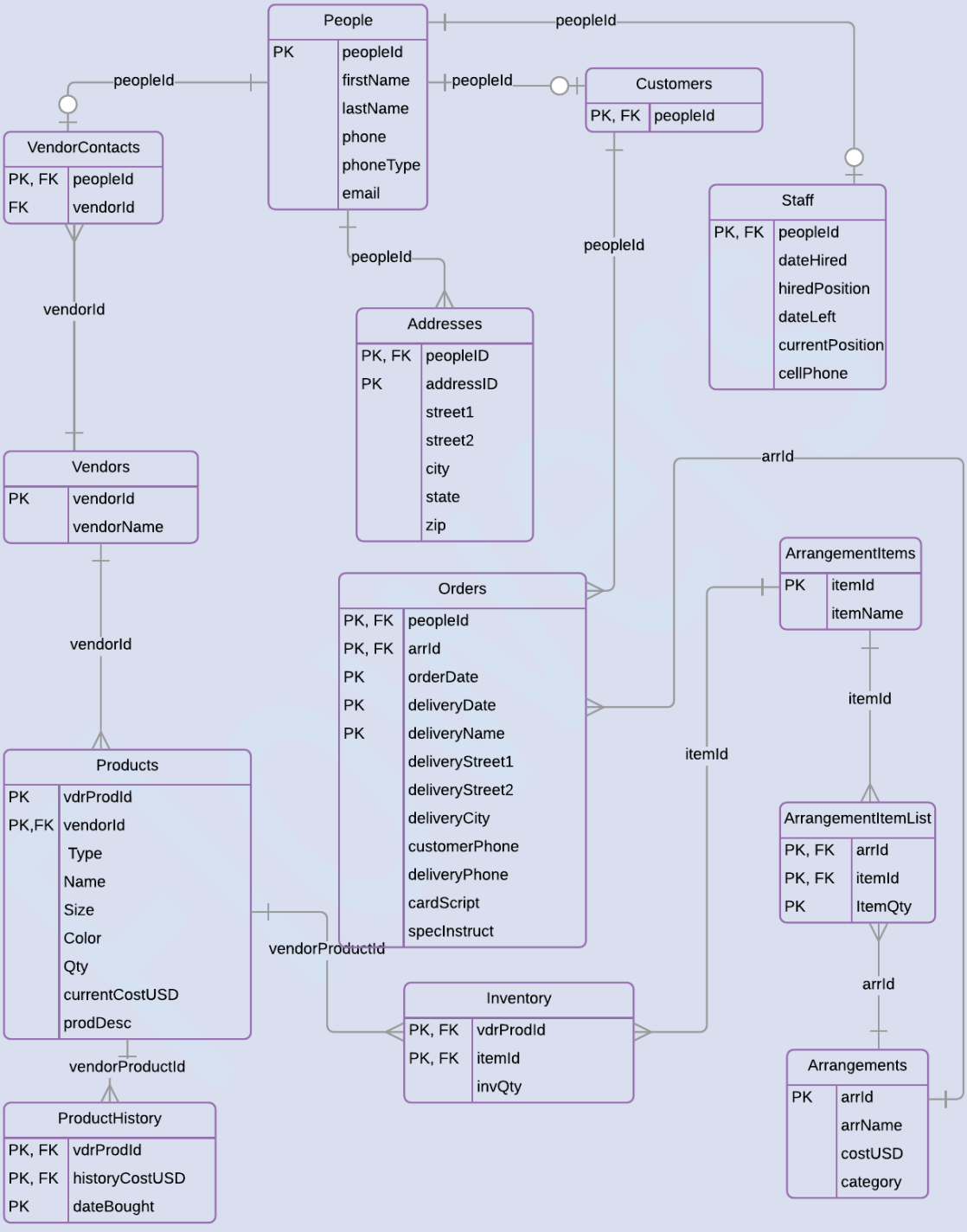
Table of Contents

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,
        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
people001	John	Dolan	800-827-3665	work	floralsupply.com
people002	Michael	Growski	800-773-2554	work	directfloral.com
people003	Jessica	Murray	877-701-7673 - ext 5465	work	globalrose.com
people004	Jon	Sitzer	845-555-5555	home	jon.sitzer34@yahoo.com
people005	Marisa	Sumter	845-666-6666	cell	[null]
people006	Mandy	Mishra	845-777-7777	home	mmishra456@gmail.com
people011	Gary	Carney	845-222-2222	home	gary.carney@net10.com
people008	Janice	Jones	845-111-1111	home	janicejones58@yahoo.com
people009	Jane	Doe	845-555-2323	cell	Jd2456@gmail.com
people010	Mary	Marist	914-555-1212	cell	mmarist6486@marist.edu
people007	Alan	Labouse...	845-440-1102	work	alan@labouseur.com
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:

vendorid text	vendorname text
vendor001	Floral Supply Wholesale
vendor002	DirectFloral.com
vendor003	GlobalRose.com
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:

peopleid text	vendorid text
people001	vendor001
people002	vendor002
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
people004	2007-05-12	sales clerk	[null]	manager	914-555-5555
people005	2015-02-23	sales clerk	[null]	sales clerk	914-666-6666
people006	2016-01-01	sales clerk	[null]	arranger	914-777-7777
people011	2007-05-12	owner	[null]	owner-arranger	914-222-2222
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:

peopleid text
people009
people010
people007
people012

Tables

-- ---

Addresses

This table keeps track of the addresses of all People.

-- ---

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

Functional Dependencies:

peopleId, addressId → street1, street2, city, state, zip

Sample Data:

peopleid text	addressid text	street1 text	street2 text	city text	state text	zip text
people001	address001	[null]	15 Applewood Drive	Fruit Heights	Utah	84037
people002	address002	[null]	760 Killian Road	Akron	Ohio	44319
people003	address003	[null]	7225 NW 25th Street	Miami	Florida	33122
people004	address004	[null]	456 Mantle Circle	Poughkeepsie	New York	12601
people005	address005	[null]	23 Alamo Road	Rhinebeck	New York	12572-1234
people006	address006	Royal ...	22B Royal Crest Place	Hyde Park	New York	12538-2256
people007	address007	Park ...	4568 Springwood Dri...	Hyde Park	New York	12538-4568
people008	address008	[null]	45 Zachary Way	Poughkeepsie	New York	12602-4545

Tables

-- ---

Products

This table tracks current product information.

-- ---

```
CREATE TABLE Products (
        vdrProdId           text    not null      unique,
        vendorId            text    not null      references Vendors(vendorId),
        Type                prod,
        Name                 text,
        Size                text,
        Color               text,
        Qty                 int   CHECK (productQty >= 0),
        -- # of items in currentCostUSD, might be sold singly, or by the dozen
        currentCostUSD              numeric(6,2)  CHECK (currentCostUSD > 0),
        prodDesc            text,
  primary key(vdrProdId, vendorId)
);
```

Functional Dependencies:

vdrProdId, vendorID → Type, Name, Size, Color, Qty, currentCostUSD, prodDesc

Sample Data:

vdrprodid text	vendorid text	type text	name text	size text	color text	qty integer	currentcostusd numeric (6,2)	proddesc text
G10026B2	vendor003	flower	rose	long stem	lavender	12	7.00	[null]
C56736	vendor001	container	country metal pail	6 inches tall	off white	5	12.00	[null]
N674DDA	vendor002	flower	alstroemeria	[null]	white	6	4.50	[null]
N673GST	vendor002	flower	statice	[null]	white	6	3.00	[null]
N675GHF	vendor002	flower	daisy	[null]	yellow	12	12.00	[null]
S97854	vendor001	other	floral foam	9in long x 4.5in wide x 3in h...	green	48	25.00	dry brick
S97657	vendor001	other	floral tape	110yds long x 1in wide	green	6	6.75	rolls
C56765	vendor001	container	its a boy wagon	9in long x 4in wide x 4in high	red	1	9.99	good for plant or arr

Tables

-- ---

ProductHistory

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

-- ---

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

Functional Dependencies:

vdrProdId → historyCostUSD, dateBought

Sample Data:

vdrprodid text	historycostusd numeric (6,2)	datebought date
G10026BG	10.00	2016-02-02
C78457	12.00	2016-02-02
N67454C3	5.00	2016-02-02
C97485	6.00	2016-02-02
G87465	13.50	2016-02-02
S97657	5.75	2016-02-02
N674DDA	6.00	2016-02-02
N67454G5	8.25	2016-02-15

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:

arid →arrName, arrCostUSD, category

Sample Data:

arrid text	arrname text	arrcostusd numeric (6,2)	category text
arr001	Blooming Pail	50.00	Spring
arr002	Wow Wagon - boy	43.00	baby boy
arr003	Wow Wagon - girl	43.00	baby girl
arr004	Fly Away Labouseur	53.00	birthday
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:

itemid text	itemname text
item001	Country Metal Pail by Alan
item002	Lavender Roses
item003	Alstroemeria
item004	Statice
item005	Yellow Daisies
item006	Floral Foam
item007	Floral tape
item008	Its a Boy 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
arr001	item001	1
arr001	item002	8
arr001	item003	5
arr001	item004	6
arr001	item005	5
arr001	item006	1
arr001	item007	1
arr001	item015	4
arr002	item005	3
arr002	item006	1
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 (
        vdrProdId    text    references Products(vdrProdId),
        itemId               text    references ArrangementItems(itemId),
        invQty               int    CHECK (invQty >= 0) ,
  primary key(vdrProdId, itemId)
);
```

Functional Dependencies:

vdrProdId, itemId → invQty

Sample Data:

vdrprodid text	itemid text	invqty integer
G10026B2	item002	10
C56736	item001	4
N674DDA	item003	0
N673GST	item004	0
N675GHF	item005	10
S97854	item006	45
S97657	item007	6
C56765	item008	1

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	t
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 text	itemid text	itemname text	itemqty integer
arr002	item005	Yellow Daisies	3
arr002	item006	Floral Foam	1
arr002	item007	Floral tape	1
arr002	item016	Its a Girl Red Wag...	1
arr002	item009	Yellow Spray Roses	2
arr002	item010	Red Mini Gerbera	3
arr002	item011	Asters	2
arr002	item012	White Daisies	3
arr002	item013	White Button Mums	2
arr002	item014	Soldago	4
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
$$
DECLARE
        arrIdEntered TEXT               := $1;
        resultset       REFCURSOR   := $2;
BEGIN
        OPEN resultset FOR
        SELECT i.itemId,  i.vdrProdID, v.vendorName, pe.firstName, pe.lastName,
                pe.phone
        FROM Inventory i INNER JOIN ArrangementItems ai ON i.itemId = ai.itemId
                        INNER JOIN ArrangementItemsList ail ON ail.itemId = ai.itemId
                        INNER JOIN Arrangements a ON ail.arrId = a.arrId
                        INNER JOIN Products pr ON i.vdrProdId = pr.vdrProdId
                        INNER JOIN Vendors v ON pr.vendorID = v.vendorId
                        INNER JOIN VendorContacts vc ON v.vendorId = vc.vendorId
                        INNER JOIN People pe ON vc.peopleId = pe.peopleId
        WHERE a.arrId = arrIdEntered   and   i.invQty <= ail.itemQty
        ORDER BY p.itemId ASC;
        RETURN resultset;
END;
$$
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
item005	3	0	N675GHF	DirectFloral.com	Michael	Growski	800-773-2554
item010	3	0	N67454C3	GlobalRose.com	Jessica	Murray	877-701-7673 - ext 5465
item011	2	0	N67454G5	GlobalRose.com	Jessica	Murray	877-701-7673 - ext 5465
item013	2	0	N697BHG	DirectFloral.com	Michael	Growski	800-773-2554
item014	4	0	N986GKB	DirectFloral.com	Michael	Growski	800-773-2554
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, addressID 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, addressId, 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, 'address013', '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	hiredposition pos	dateleft date	currentposition pos	cellphone text	firstname text	lastname text	phone text	phonetype phone
people006	2016-01-01	salesclerk	[null]	arranger	914-777-7777	Mandy	Mishra	845-777-7777	home
people008	2007-05-12	salesclerk	2007-07-30	salesclerk	914-111-1111	Janice	Jones	845-111-1111	home
people011	2007-05-12	owner	[null]	owner	914-222-2222	Gary	Carney	845-222-2222	home
people013	2017-04-30	arranger	[null]	arranger	555-555-5555	Kathy	Coomes	555-555-5555	cell

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, 'address014', '100 Main Street', 'Rhinecliff', 'New York', '12573', NULL, NULL, NULL, NULL, NULL, NULL);
SELECT * FROM CustomerInfo;

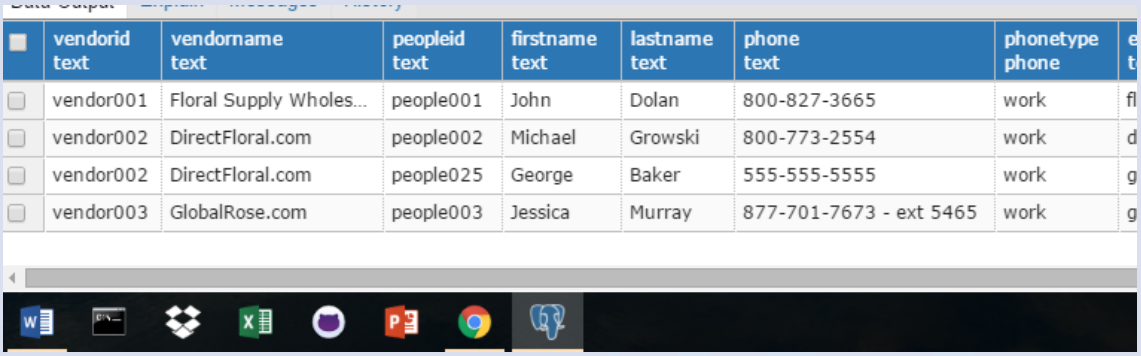
Data Output Explain Messages History

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

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, 'address025', '3657 Floral Drive', 'Pleasant Valley',
'New York',
'12569', NULL, NULL, NULL, NULL, NULL, 'vendor002');
SELECT * FROM VendorInfo;



vendorid text	vendorname text	peopleid text	firstname text	lastname text	phone text	phonetype phone	e t
vendor001	Floral Supply Wholes...	people001	John	Dolan	800-827-3665	work	fl
vendor002	DirectFloral.com	people002	Michael	Growski	800-773-2554	work	d
vendor002	DirectFloral.com	people025	George	Baker	555-555-5555	work	g
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
vendor001	Floral Supply Wholesale
vendor002	DirectFloral.com
vendor003	GlobalRose.com
vendor004	Flowers by Alan
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, newVdrProdId text, vendorId text, newType text,
        newName text, newSize text, newColor text, newQty int,
        newCurrentCostUSD numeric, newProdDesc text, historyCostUSD numeric,
        dateBought date, newItemId text, NewItemName text, newInvQty int)
RETURNS void AS $$
BEGIN
        IF productNew = 'Y' THEN
                INSERT INTO Products
                        VALUES (newVdrProdId, vendorId, newType,  newName,  newSiize,
                newColor, newQty, newCurrentCostUSD,  newProdDesc);
                INSERT INTO ArrangementItems
                        VALUES (newItemId, newItemName);
                INSERT INTO Inventory
                        VALUES(newVdrProdId, newItemId, newInvQty);
                INSERT INTO ProductHistory
                        VALUES(newVdrProdId, historyCostUSD, dateBought);
        END IF;
        IF productNew = 'N' THEN
                UPDATE Products SET
                        productType = newProductType,
                        productName = newProductName,
                        productSize = newProductSize,
                        productColor = newProductColor,
                        productQty = newProductQty,
                        currentCostUSD = newCurrentCostUSD,
                        productDesc = newProductDesc
                        WHERE vendorProductId = newVendorProductId;
                INSERT INTO ProductHistory
                        VALUES(newVendorProductId, historyCostUSD, dateBought);
                UPDATE Inventory SET invQty = invQty + newInvQty
                        WHERE vendorproductId = newVendorProductId and itemId =
                        newItemId;
        END IF;
END
$$ 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	productsize text	productcolor text	productqty integer	currentco... numeric (...	productdesc text
C97483	vendor001	balloon	balloon	mylar	silver and mixed	25	12.50	colorful HappyBirthday
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]

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

a Output | Explain | Messages | History

vendorproductid text	historycos... numeric (...	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

vendorproductid text	itemid text	invqty integer
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	productsize text	productcolor text	productqty integer	currentcostusd 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	historycos... numeric (...	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

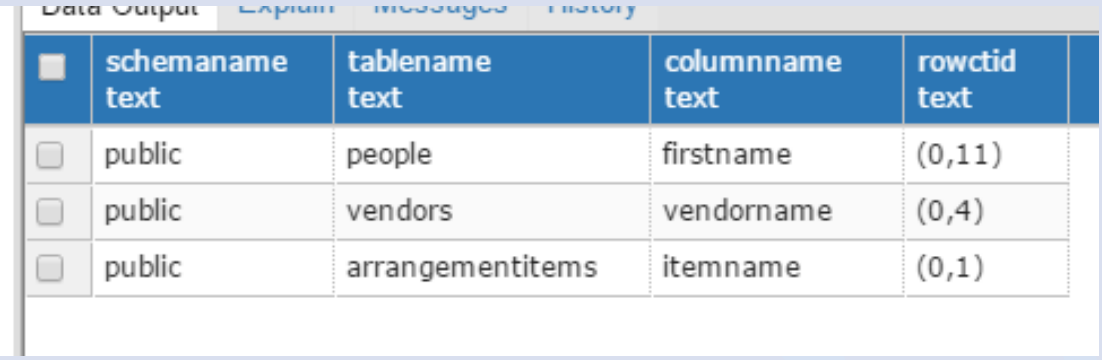
Find Alan Report - Finds Alan or Labouseur anywhere in the database

Created by Daniel Verite and shared on3 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/pgsqlfunction 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;
```

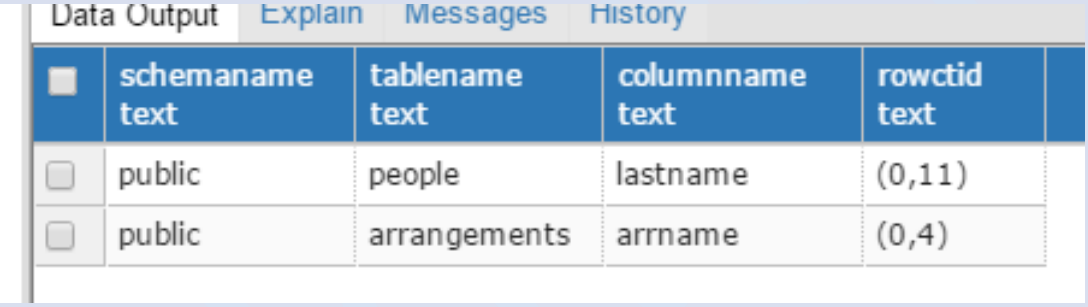
Stored Procedures

SELECT * FROM search_columns('Alan');



schemaname text	tablename text	columnname text	rowctid text
public	people	firstname	(0,11)
public	vendors	vendorname	(0,4)
public	arrangementitems	itemname	(0,1)

SELECT * FROM search_columns('Labouseur');



Data Output | Explain | Messages | History

schemaname text	tablename text	columnname text	rowctid text
public	people	lastname	(0,11)
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.addressId, 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	phonety... phone	email text	addressid text	street1 text	street2 text	city text	state text	zip text
vendor001	Floral Supp...	people001	John	Dolan	800-827...	work	floralsupply....	address001	[null]	15 Applewood ...	Fruit Hei...	Utah	84037
vendor002	DirectFlora...	people002	Michael	Growski	800-773...	work	directfloral.c...	address002	[null]	760 Killian Road	Akron	Ohio	44319
vendor003	GlobalRose...	people003	Jessica	Murray	877-701...	work	globalrose.c...	address003	[null]	7225 NW 25t...	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.address, 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:

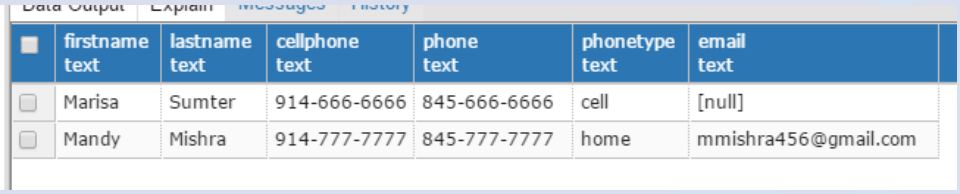
peopleid text	datehired date	hiredposition pos	dateleft date	currentposi... pos	cellphone text	firstna... text	lastna... text	phone text	phonety... phone
people004	2007-05-12	salesclerk	[null]	manager	914-555-55...	Jon	Sitzer	845-555-...	home
people005	2015-02-23	salesclerk	[null]	salesclerk	914-666-66...	Marisa	Sumter	845-666-...	cell
people006	2016-01-01	salesclerk	[null]	arranger	914-777-77...	Mandy	Mishra	845-777-...	home
people008	2007-05-12	salesclerk	2007-07-30	salesclerk	914-111-11...	Janice	Jones	845-111-...	home
people011	2007-05-12	owner	[null]	owner	914-222-22...	Gary	Carney	845-222-...	home

email text	addressid text	street1 text	street2 text	city text	state text	zip text
jon.sitzer3...	address004	[null]	456 Mantle Circle	Poughkeepsie	New York	12601
[null]	address005	[null]	23 Alamo Road	Rhinebeck	New York	12572-1234
mmishra45...	address006	Royal Crest Apart...	22B Royal Crest Pl...	Hyde Park	New York	12538-2256
janicejones...	address008	[null]	45 Zachary Way	Poughkeepsie	New York	12602-4545
gary.carne...	address011	[null]	86B Church Street	Poughkeepsie	New York	12602

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;

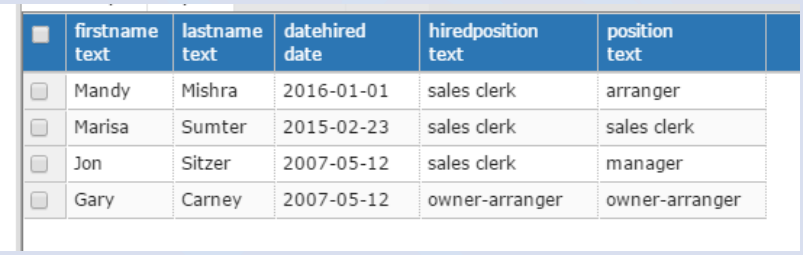
Sample Data:



firstname text	lastname text	cellphone text	phone text	phonetype text	email text
Marisa	Sumter	914-666-6666	845-666-6666	cell	[null]
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
Mandy	Mishra	2016-01-01	sales clerk	arranger
Marisa	Sumter	2015-02-23	sales clerk	sales clerk
Jon	Sitzer	2007-05-12	sales clerk	manager
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:

Data Output | Explain | Messages | History

firstname text	lastname text	datehired date	hiredposition text	position text
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.addressId,  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... phone	email text	addressid text	street1 text	street2 text	city text	state text	zip text
people007	Alan	Labouseur	845-440-1...	work	alan@labouseu...	address007	Park Manor Apa...	4568 Spring...	Hyde Park	New York	12538-45...
people009	Jane	Doe	845-555-2...	cell	Jd2456@gmail....	address009	Cherry Condomi...	21 Cherry Hill...	Red Hook	New York	12571
people010	Mary	Marist	914-555-1...	cell	mmarist6486...	address010	[null]	62 Hilltop Road	Rhinecliff	New York	12573
people012	Candie	Kane	846-555-4...	home	candie.kane22...	address012	[null]	91-28B Main ...	Beacon	New York	12508-19...

-- ---- 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
Jane	Doe	845-555-23...	cell	Jd2456@gmail.com	Red Hook
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	vdrprodid text	type text	name text	size text
vendor003	GlobalRose.com	G10026BG	flower	rose	[null]
vendor001	Floral Supply Whol...	C78457	container	vase	7in tall
vendor003	GlobalRose.com	N67454C3	flower	gerbera	mini
vendor001	Floral Supply Whol...	C97485	balloon	balloon	mylar
vendor001	Floral Supply Whol...	G87465	greenery	huckleberry	[null]
vendor001	Floral Supply Whol...	S97657	other	floral tape	110yds long x 1in wide
vendor002	DirectFloral.com	N674DDA	flower	alstroemeria	[null]

color text	qty integer	currentco... numeric (...	proddesc text	datebought date	historycostusd numeric (6,2)
orange	4	10.00	[null]	2016-02-02	10.00
cobalt blue	4	12.00	[null]	2016-02-02	12.00
red	12	6.00	[null]	2016-02-02	5.00
silver and mixed	25	12.50	colorful HappyBirthday	2016-02-02	6.00
green	1	15.75	bunch	2016-02-02	13.50
green	6	6.75	rolls	2016-02-02	5.75
white	6	4.50	[null]	2016-02-02	6.00

Views

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

```
SELECT vdrProdId, Name, Size, Qty, currentCostUSD, dateBought, historyCostUSD
FROM ProductInfo
ORDER BY vdrProdId, dateBought;
```

Sample Data:

vdrprodid text	name text	size text	qty integer	currentcostusd numeric (6,2)	datebought date	historycostusd numeric (6,2)
C56736	country metal pail	6 inches tall	5	12.00	2016-02-15	12.00
C56736	country metal pail	6 inches tall	5	12.00	2016-12-02	12.00
C56765	its a boy wagon	9in long x 4in wide x 4in high	1	9.99	2017-04-12	9.99
C56766	its a girl wagon	9in long x 4in wide x 4in high	1	9.99	2017-04-12	9.99
C78457	vase	7in tall	4	12.00	2016-02-02	12.00
C86397	vase	5in tall	5	15.00	2016-05-11	15.00
C97485	balloon	mylar	25	12.50	2016-02-02	6.00
F87467G5	carnation	[null]	12	6.00	2016-05-11	6.00

Reports

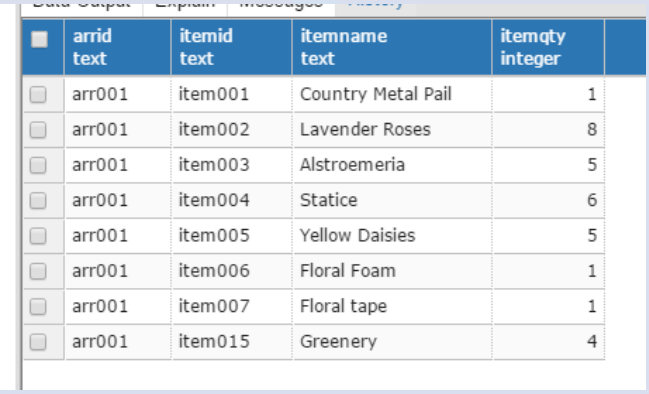
-- ---
Reports that **list all** from each of the tables.
Results were listed with each table create:
-- ---

```
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.addId = 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
arr001	item001	Country Metal Pail	1
arr001	item002	Lavender Roses	8
arr001	item003	Alstroemeria	5
arr001	item004	Statice	6
arr001	item005	Yellow Daisies	5
arr001	item006	Floral Foam	1
arr001	item007	Floral tape	1
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:

vendorid text	vendorname text
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:

vendorid text	vendorname text
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.vdrProdId = ph.vdrProdId
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', 'newVdrProdId', 'vendorId', 'newType',
-- 'newName', 'newSize', 'newColor', 'newQty', 'newCurrentCostUSD', 'newDesc',
-- '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,
        Orders CASCADE;

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

DROP FUNCTION IF EXISTS GetArrListofItems(text, refcursor) CASCADE;
DROP FUNCTION IF EXISTS GetProductOrderList(text, refcursor) CASCADE;
DROP FUNCTION IF EXISTS AddPeople(peopleType text, peopleId text, firstName text,
                lastName text, phone text,
                phoneType phone, email text, addressId text, street1 text, street2 text,
                city text, state text, zip text, dateHired date, hiredPosition pos,
                dateLeft date, currentPosition pos, cellPhone text, vendorId text) CASCADE;
DROP FUNCTION IF EXISTS AddVendor(vendorId text, vendorName text) CASCADE;
DROP FUNCTION IF EXISTS     AddProducts(productNew text, newVdrProdId text, vendorId text,
        newType text, newName text, newSize text, newColor text, newQty int,
        newCurrentCostUSD numeric, newprodDesc text, historyCostUSD numeric,
        dateBought date, newItemId text, NewItemName text, newInvQty int) CASCADE;

DROP TYPE IF EXISTS phone, prod, pos CASCADE;

DROP TRIGGER IF EXISTS validPeopleInput ON People CASCADE;
DROP TRIGGER IF EXISTS validVendorInput ON Vendors 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 ArrangementIitems. 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.