Pearl by Pearl
A Custom Cosmetics Company

CEO and Founder: Pearl Amin

Table of Contents

Executive Summary

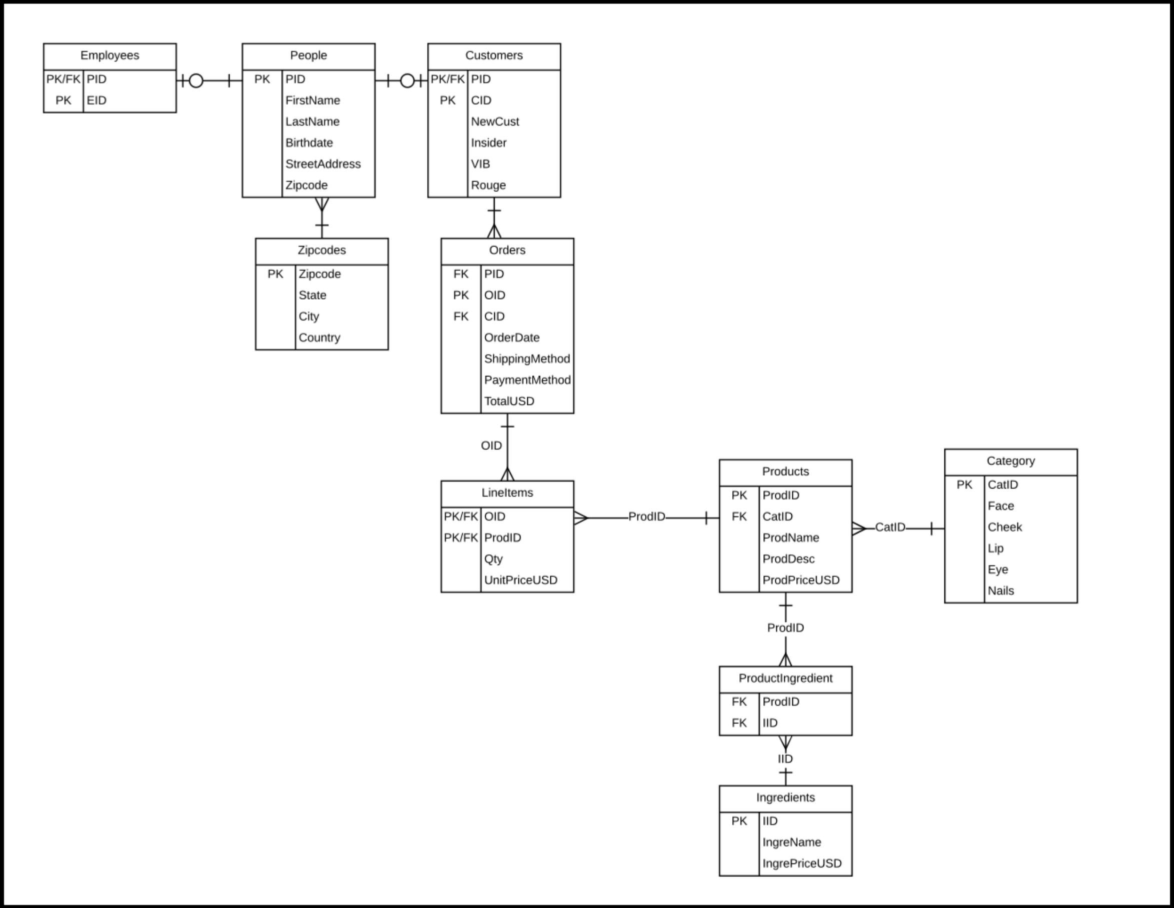
Pearl by Pearl is a 3D printing custom cosmetics company founded in 2016 that is inclusive of all people from all walks of life that allows users of all backgrounds to customize cosmetics to fit their own unique identity. The goal of Pearl by Pearl is to overthrow the monstrous, exclusive monopoly that is the cosmetics industry, which forces beautiful and exceptional people from diverse backgrounds feel as if they have to fit in the idealistic, small-minded, box that big name cosmetics companies coerce.

The Pearl by Pearl database was created in 2017 by its CEO and Founder, Pearl Amin. This database has been created to keep track of all aspects from the people to the raw materials involved in the operations of the company Pearl By Pearl.

This paper provides an insight into the the information regarding the database as well as its uses and what that indicates. The entity-relationship diagram is presented first, followed by create statements for tables, and then the sample data used. Following these are queries, views, stored procedures, reports, and triggers that are designed and tested for optimal results. The function of this database is to provide accurate, relevant, undiluted information from all aspects of the company to provide a unique and necessary insight to maximum profitability.

Entity-Relationship Diagram

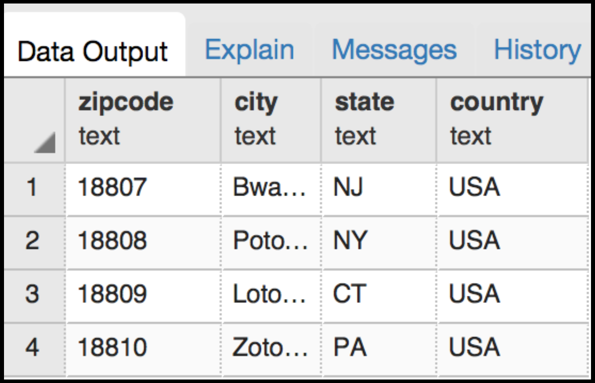
Pearl by Pearl



Zipcode Table

This table contains all zip codes for all stores and people, whether they are customers or employees. There is a city, state, and country for each unique zipcode.

```
DROP TABLE IF EXISTS Zipcode;
        CREATE TABLE Zipcode (
        Zipcode text not null,
        City text,
        State text,
        Country text,
        primary key (Zipcode) );
```



Data Output | Explain | Messages | History

	zipcode text	city text	state text	country text
1	18807	Bwa…	NJ	USA
2	18808	Poto…	NY	USA
3	18809	Loto…	CT	USA
4	18810	Zoto…	PA	USA

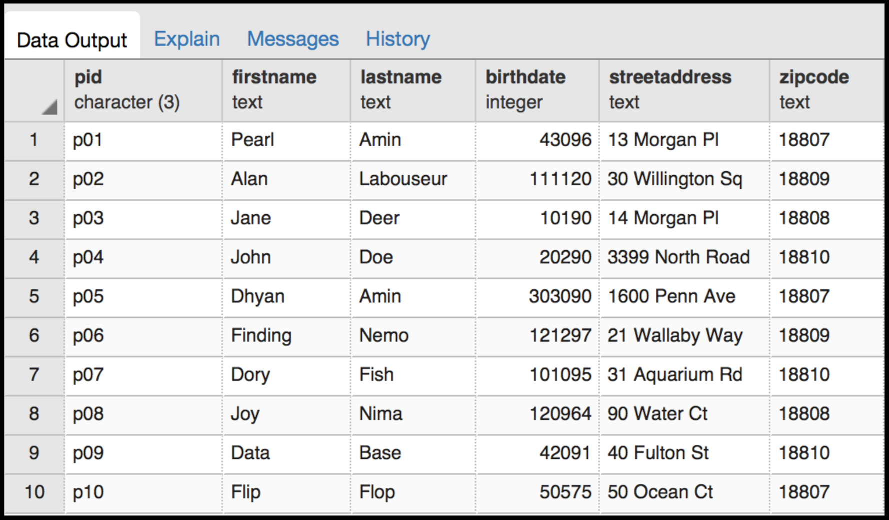
Functional dependencies:
Zipcode $\rightarrow$ city, state, country

People Table

This table contains all of the information on any customer or employee. This information includes the person's first name, last name, birthdate, street address, and zip code based on their unique identification.

```
DROP TABLE IF EXISTS People;
        CREATE TABLE People (
        PID char(3) not null,
        FirstName text,
        LastName text,
        Birthdate int,
        StreetAddress text,
        Zipcode text not null references Zipcode(Zipcode),
        primary key (PID));
```

Functional Dependencies:
PID → FirstName, LastName, Birthdate, StreetAddress, Zipcode



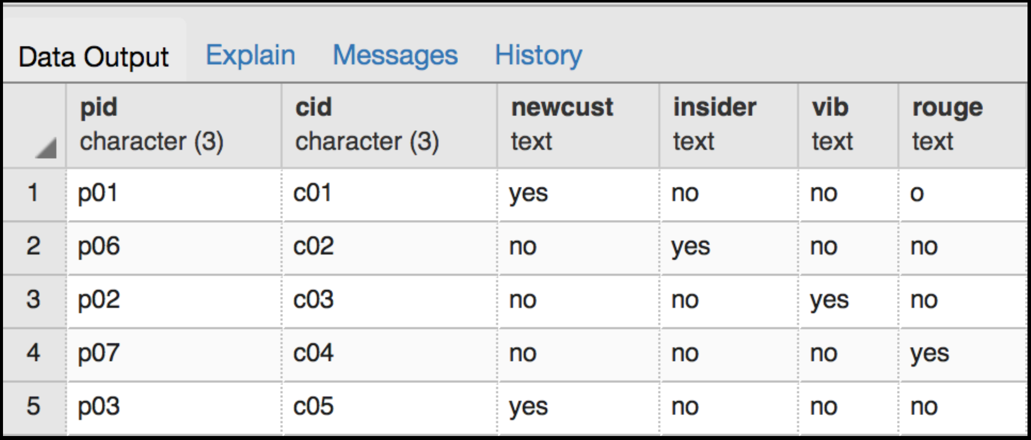
Data Output | Explain | Messages | History

	pid character (3)	firstname text	lastname text	birthdate integer	streetaddress text	zipcode text
1	p01	Pearl	Amin	43096	13 Morgan Pl	18807
2	p02	Alan	Labouseur	111120	30 Willington Sq	18809
3	p03	Jane	Deer	10190	14 Morgan Pl	18808
4	p04	John	Doe	20290	3399 North Road	18810
5	p05	Dhyan	Amin	303090	1600 Penn Ave	18807
6	p06	Finding	Nemo	121297	21 Wallaby Way	18809
7	p07	Dory	Fish	101095	31 Aquarium Rd	18810
8	p08	Joy	Nima	120964	90 Water Ct	18808
9	p09	Data	Base	42091	40 Fulton St	18810
10	p10	Flip	Flop	50575	50 Ocean Ct	18807

Customers Table

The Customers table contains all of the customers that Pearl by Pearl has, whether they are new members or returning customers who fall under the three perks categories: insider, VIB, or rouge. If the information on whether a customer is a perks member holder or a new customer is not given, then they will be default be a new customer.

```
DROP TABLE IF EXISTS Customers;
        CREATE TABLE Customers (
        PID char(3) not null references people(PID),
        CID char(3) not null,
        NewCust text default 'Yes',
        Insider text,
        VIB text,
        Rouge text,
        CONSTRAINT CHK_Memeber CHECK
        (NewCust='yes' or NewCust='no'),
        primary key(CID));
```



	pid character (3)	cid character (3)	newcust text	insider text	vib text	rouge text
1	p01	c01	yes	no	no	o
2	p06	c02	no	yes	no	no
3	p02	c03	no	no	yes	no
4	p07	c04	no	no	no	yes
5	p03	c05	yes	no	no	no

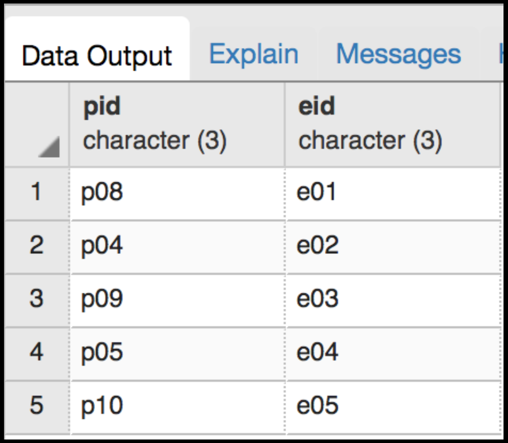
Functional Dependencies: PID, CID → NewCust, Insider, VIB, Rouge

Employees Table

The employees table contains the unique identification number of each of the employees at Pearl by Pearl.
A person cannot be added to the employee table unless they are already added to the people table.

```
DROP TABLE IF EXISTS Employees;
        CREATE TABLE Employees (
        PID char(3) not null references
        People(PID),
        EID char(3) not null,
        primary key (EID) );
```

Functional Dependencies: PID, EID →

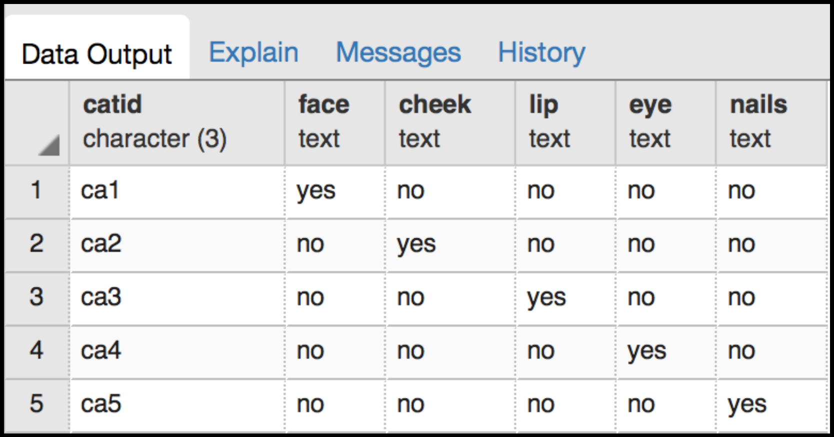


	pid character (3)	eid character (3)
1	p08	e01
2	p04	e02
3	p09	e03
4	p05	e04
5	p10	e05

Category Table

The category table contains the unique category identification that each product will be placed into. Every product sold by Pearl by Pearl will belong one of the five categories: face, cheek, lip, eye or nails.

```
DROP TABLE IF EXISTS Category;
CREATE TABLE Category (
            CatID char(3) not null,
            Face text,
            Cheek text,
            Lip text,
            Eye text,
            Nails text,
            primary key(CatID));
```


Data Output | Explain | Messages | History

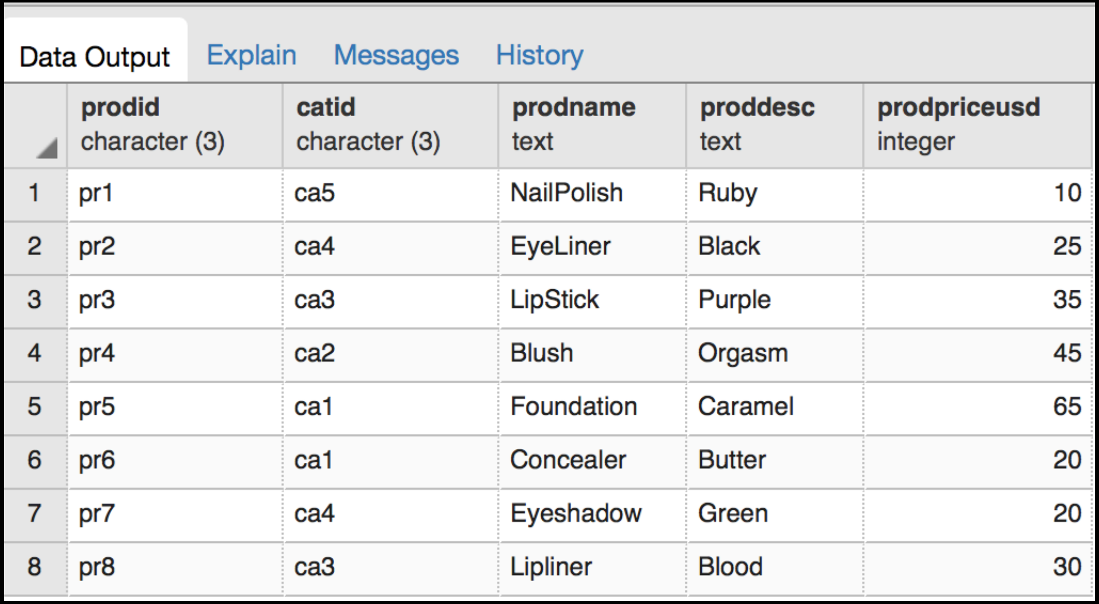
	catid character (3)	face text	cheek text	lip text	eye text	nails text
1	ca1	yes	no	no	no	no
2	ca2	no	yes	no	no	no
3	ca3	no	no	yes	no	no
4	ca4	no	no	no	yes	no
5	ca5	no	no	no	no	yes

Functional Dependencies: CatID → face, cheek, lip, eye, nails

Products Table

The products table contains the unique product identification number assigned to each product, the category identification number, the products name, the products description, and the price per produce in USD.

```
DROP TABLE IF EXISTS Products;
CREATE TABLE Products (
        ProdID char(3) not null,
        CatID char(3) not null references
        Category(CatID),
        ProdName text,
        ProdDesc text,
        ProdPriceUSD int,
        primary key(ProdID));
```



Data Output | Explain | Messages | History

	prodid character (3)	catid character (3)	prodname text	proddesc text	prodpriceusd integer
1	pr1	ca5	NailPolish	Ruby	10
2	pr2	ca4	EyeLiner	Black	25
3	pr3	ca3	LipStick	Purple	35
4	pr4	ca2	Blush	Orgasm	45
5	pr5	ca1	Foundation	Caramel	65
6	pr6	ca1	Concealer	Butter	20
7	pr7	ca4	Eyeshadow	Green	20
8	pr8	ca3	Lipliner	Blood	30

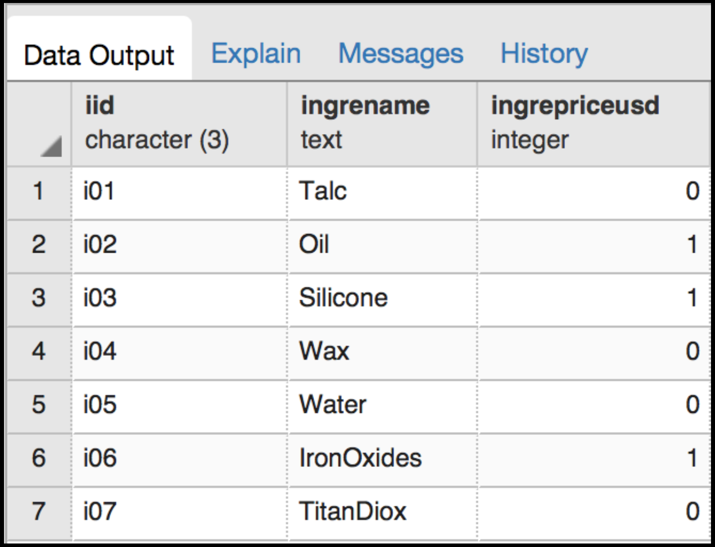
Functional Dependencies: ProdID → CatID, prodname, proddesc, prodepriceusd

Ingredients Table

The ingredients table displays all of the ingredients names, along with a unique ingredient identifier, as well as the prices, used to create the makeup sold by the company Pearl by Pearl.

```
DROP TABLE IF EXISTS Ingredients;
CREATE TABLE Ingredients (
        IID char(3) not null,
        IngreName text,
        IngrePriceUSD int,
        primary key(IID)     );
```

Functional Dependencies: IID → IngreName, IngrePriceUSD



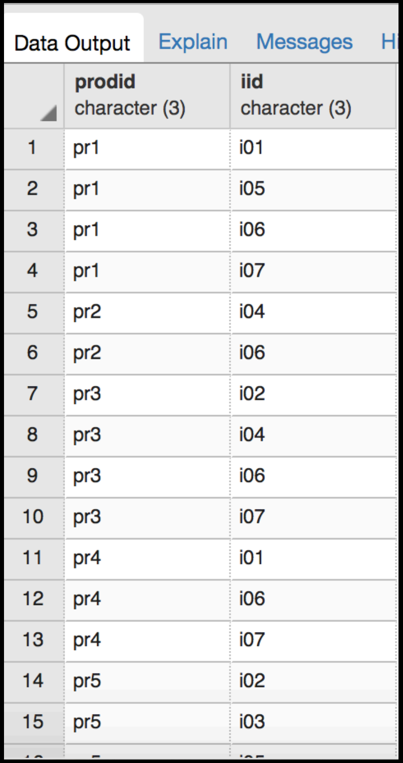
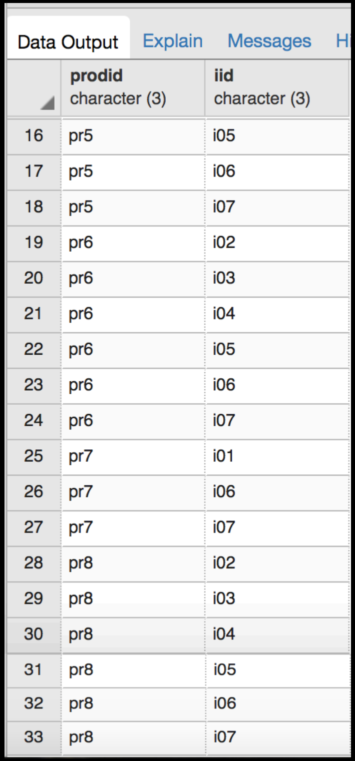
	iid character (3)	ingrename text	ingrepriceusd integer
1	i01	Talc	0
2	i02	Oil	1
3	i03	Silicone	1
4	i04	Wax	0
5	i05	Water	0
6	i06	IronOxides	1
7	i07	TitanDiox	0

ProductIngredients Table

This table displays which ingredients are used in which products by utilizing the unique identifiers for both products and ingredients

```
DROP TABLE IF EXISTS ProductIngredient;
CREATE TABLE ProductIngredient(
            ProdID char(3) not null references
            Products(ProdID),
            IID char(3) not null references Ingredients(IID)      );
```

Functional Dependencies: ProdID, IID →

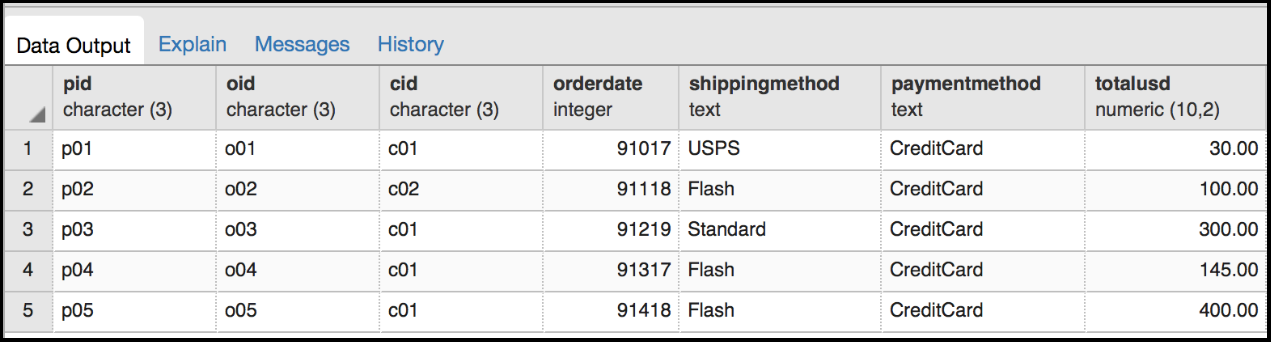
Data Output | Explain | Messages | Hi

	prodid character (3)	iid character (3)
1	pr1	i01
2	pr1	i05
3	pr1	i06
4	pr1	i07
5	pr2	i04
6	pr2	i06
7	pr3	i02
8	pr3	i04
9	pr3	i06
10	pr3	i07
11	pr4	i01
12	pr4	i06
13	pr4	i07
14	pr5	i02
15	pr5	i03
16	pr5	i05
17	pr5	i06
18	pr5	i07
19	pr6	i02
20	pr6	i03
21	pr6	i04
22	pr6	i05
23	pr6	i06
24	pr6	i07
25	pr7	i01
26	pr7	i06
27	pr7	i07
28	pr8	i02
29	pr8	i03
30	pr8	i04
31	pr8	i05
32	pr8	i06
33	pr8	i07

Orders Table

The orders table include the people, who are customers that made orders, and on which date they placed the order, the shipping method they used, the payment method they chose and the total cost in USD. A customer cannot be in the orders table unless they are also in the customers table.

```
DROP TABLE IF EXISTS Orders;
CREATE TABLE Orders (
        PID char(3) not null references People(PID),
        OID char(3) not null,
        CID char(3) not null references Customers(CID),
        OrderDate int,
        ShippingMethod text,
        PaymentMethod text,
        TotalUSD numeric(10,2),
        primary key(OID));
```



	pid character (3)	oid character (3)	cid character (3)	orderdate integer	shippingmethod text	paymentmethod text	totalusd numeric (10,2)
1	p01	o01	c01	91017	USPS	CreditCard	30.00
2	p02	o02	c02	91118	Flash	CreditCard	100.00
3	p03	o03	c01	91219	Standard	CreditCard	300.00
4	p04	o04	c01	91317	Flash	CreditCard	145.00
5	p05	o05	c01	91418	Flash	CreditCard	400.00

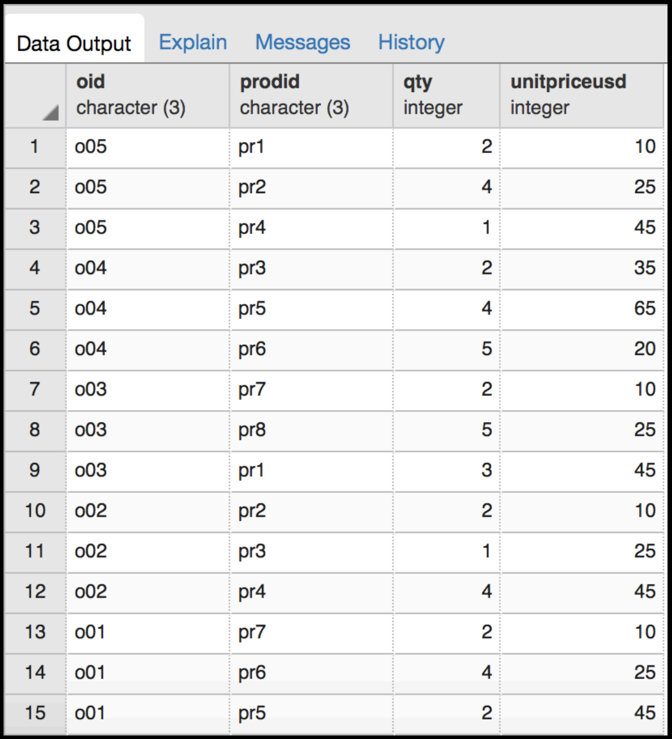
Functional Dependencies:
PID, OID, CID $\rightarrow$ Orderdate, ShippingMethod, PaymentMethod, TotalUSD

LineItems Table

The LineItems table includes the quantity of each product, with each products price per unit, in each each order placed by customers.

```
DROP TABLE IF EXISTS LineItems;
CREATE TABLE LineItems (
        OID char(3) not null references Orders(OID),
        ProdID char(3) not null references Products(ProdID),
        Qty int not null,
        UnitPriceUSD int not null,
        primary key (OID, ProdID));
```

Functional Dependencies: OID, ProdID, QTY, UnitPriceUSD →



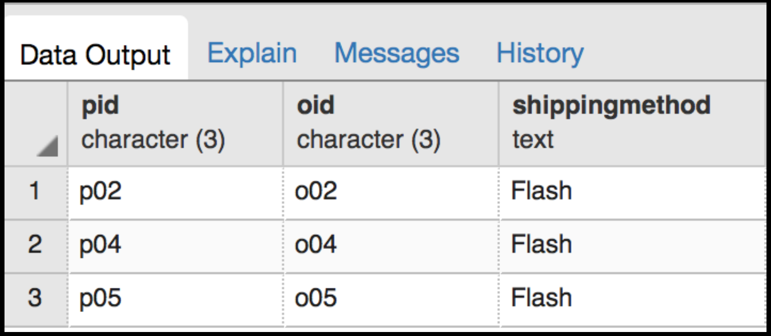
Data Output | Explain | Messages | History

	oid character (3)	prodid character (3)	qty integer	unitpriceusd integer
1	o05	pr1	2	10
2	o05	pr2	4	25
3	o05	pr4	1	45
4	o04	pr3	2	35
5	o04	pr5	4	65
6	o04	pr6	5	20
7	o03	pr7	2	10
8	o03	pr8	5	25
9	o03	pr1	3	45
10	o02	pr2	2	10
11	o02	pr3	1	25
12	o02	pr4	4	45
13	o01	pr7	2	10
14	o01	pr6	4	25
15	o01	pr5	2	45

Views

Gets the PID, OID, and Shipping Method of any customer who made an order with Flash Shipping

```
create or replace view FlashOrders as
        select distinct pid, oid, shippingmethod
        from orders
        where shippingmethod = 'Flash'  ;
```

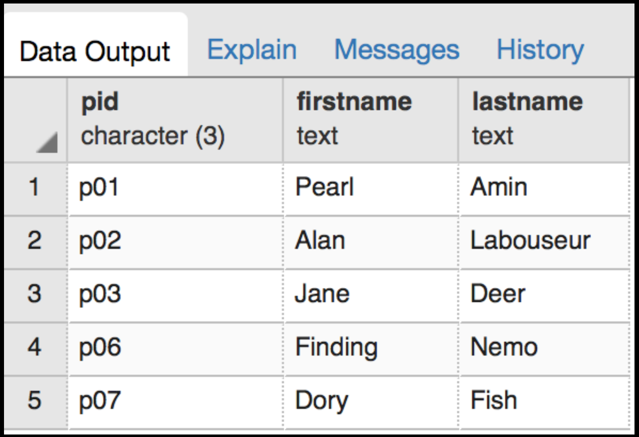


	pid character (3)	oid character (3)	shippingmethod text
1	p02	o02	Flash
2	p04	o04	Flash
3	p05	o05	Flash

Views

Retrieves the First Name, Last Name and unique identifier of people who are customers

```
create or replace view CustomersandPeople as
        select distinct p.pid, p.firstname, p.lastname
        from people p, customers c
        where c.pid in
                (select pid
                 from people
                where c.pid = p.pid
                order by p.pid ASC) ;
```

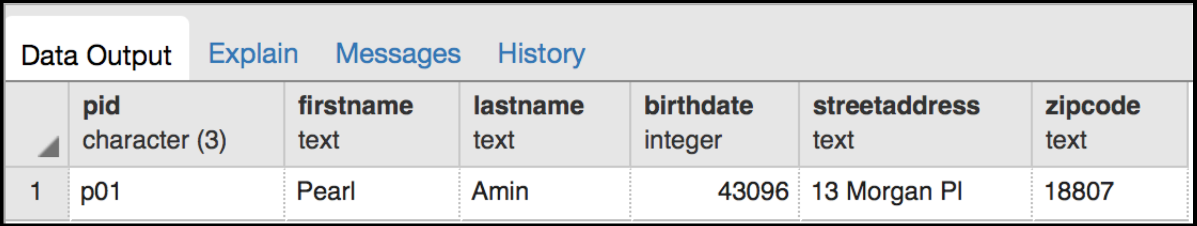


	pid character (3)	firstname text	lastname text
1	p01	Pearl	Amin
2	p02	Alan	Labouseur
3	p03	Jane	Deer
4	p06	Finding	Nemo
5	p07	Dory	Fish

Stored Procedure

This stored procedure allowed users to search for a section of a person's first name, last name, or both names. The result of the search includes a person's full first name, full last name, as well as their birthdate, street address, and zip code.

```
CREATE OR REPLACE FUNCTION searchCustomerName (text, text, REFCURSOR)
returns refcursor as
$$
Declare
        searchFirst TEXT := $1;
        searchLast TEXT := $2;
        resultSet REFCURSOR := $3;
BEGIN
        OPEN resultSet FOR
        SELECT *
        FROM People
        WHERE FirstName LIKE searchFirst
        AND LastName LIKE searchLast;
        return resultSet;
end;
$$
LANGUAGE plpgsql; -
```

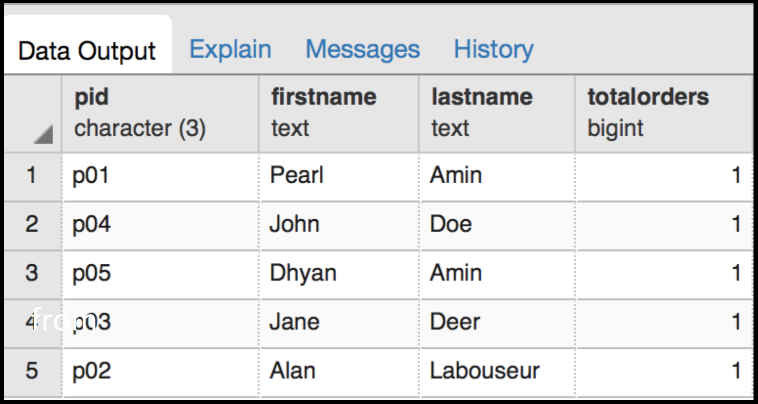


	pid character (3)	firstname text	lastname text	birthdate integer	streetaddress text	zipcode text
1	p01	Pearl	Amin	43096	13 Morgan Pl	18807

Stored Procedure

This stored procedure allows a user to search for the total number of orders placed by a customer. The result of the search is the person’s first name, last name, unique identification number, and their total number of orders.

```
Stored Procedure get totalorders per pid
CREATE OR REPLACE FUNCTION totalsalesUSD (text, text, REFCURSOR) returns refcursor as
$$
Declare
        totalsalesusd int := $1;
        pid TEXT := $2;
        oid TEXT := $3;
        cid TEXT := $4;
        resultSet REFCURSOR := $5;
 BEGIN
        OPEN resultSet FOR
        select pid, orders.oid, cid, qty*unitpriceUSD as "TotalSalesUSD"
orders inner join lineitems on orders.oid = lineitems.oid;
        return resultSet;
end;
$$
LANGUAGE plpgsql;
```

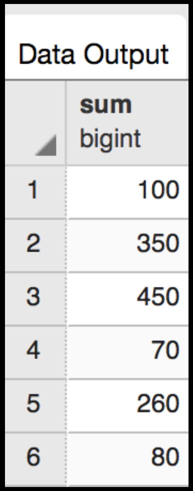


	pid character (3)	firstname text	lastname text	totalorders bigint
1	p01	Pearl	Amin	1
2	p04	John	Doe	1
3	p05	Dhyan	Amin	1
4	p03	Jane	Deer	1
5	p02	Alan	Labouseur	1

Reports

This report shows the total amount of money collected in 2017 per product:

```
SELECT sum(qty * unitpriceUSD)
FROM lineitems
GROUP BY unitpriceUSD
```



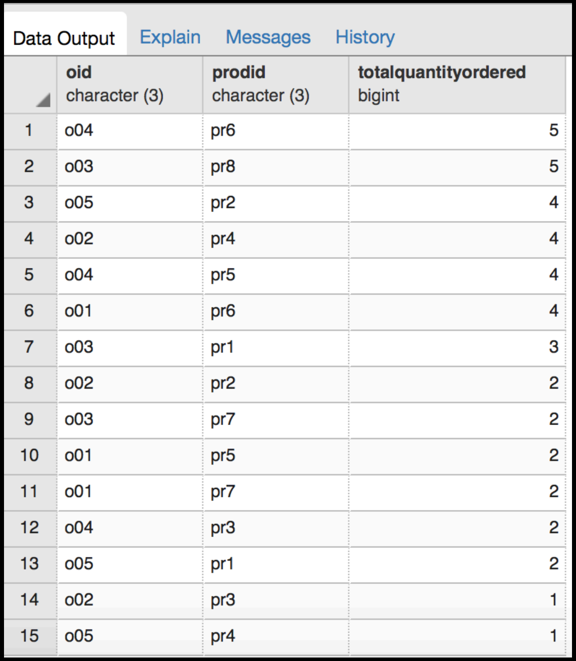
Data Output

	sum bigint
1	100
2	350
3	450
4	70
5	260
6	80

Reports

This report shows the quantity of each product ordered per order:

```
SELECT OID, ProdID, SUM(Qty) AS TotalQuantityOrdered
FROM LineItems
GROUP BY OID, ProdID
ORDER BY SUM(Qty) DESC;
```

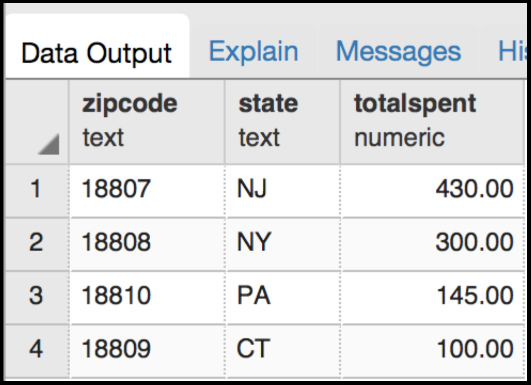


	oid character (3)	prodid character (3)	totalquantityordered bigint
1	o04	pr6	5
2	o03	pr8	5
3	o05	pr2	4
4	o02	pr4	4
5	o04	pr5	4
6	o01	pr6	4
7	o03	pr1	3
8	o02	pr2	2
9	o03	pr7	2
10	o01	pr5	2
11	o01	pr7	2
12	o04	pr3	2
13	o05	pr1	2
14	o02	pr3	1
15	o05	pr4	1

Reports

This report shows the total amount of money in USD spent by customers by each city:

```
SELECT z.ZipCode ,z.State, SUM(o.TotalUSD) AS TotalSpent
FROM Zipcode z, People p, Orders o
WHERE z.ZipCode = p.ZipCode
AND p.PID = o.PID
GROUP BY z.ZipCode, z.State
ORDER BY SUM(o.TotalUSD) DESC;
```

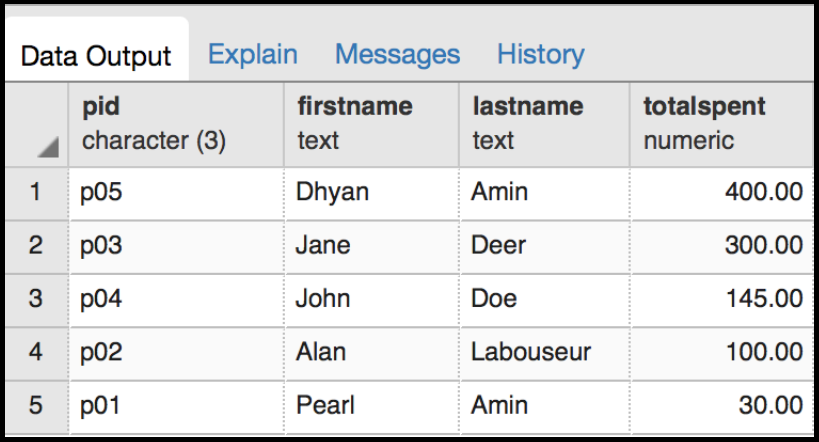


	zipcode text	state text	totalspent numeric
1	18807	NJ	430.00
2	18808	NY	300.00
3	18810	PA	145.00
4	18809	CT	100.00

Reports

This report shows the customers that spent the most money in USD at Pearl by Pearl:

```
SELECT p.PID, p.FirstName, p.LastName, sum(o.TotalUSD) AS TotalSpent
FROM People p, Orders o
WHERE p.PID = o.PID
GROUP BY p.PID, p.FirstName, p.LastName
ORDER BY sum(o.TotalUSD) DESC;
```

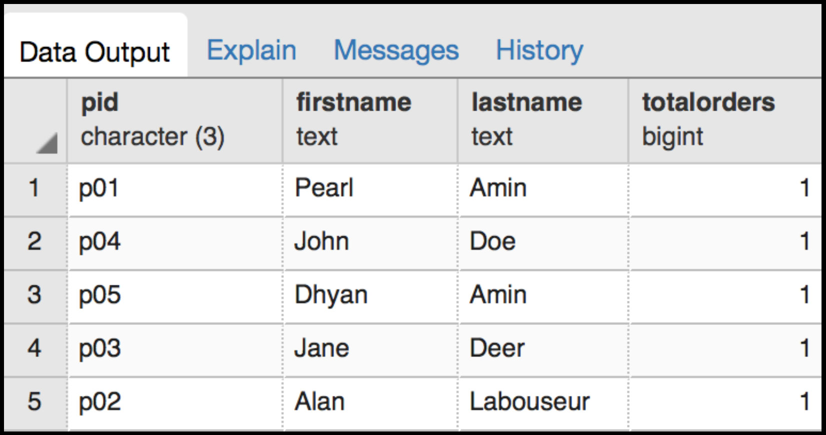


	pid character (3)	firstname text	lastname text	totalspent numeric
1	p05	Dhyan	Amin	400.00
2	p03	Jane	Deer	300.00
3	p04	John	Doe	145.00
4	p02	Alan	Labouseur	100.00
5	p01	Pearl	Amin	30.00

Reports

This report shows the customers that made the most orders at Pearl by Pearl:

```
SELECT p.PID, p.FirstName, p.LastName, count(o.OID) AS TotalOrders
FROM People p, Orders o
WHERE p.PID = o.PID
GROUP BY p.PID, p.FirstName, p.LastName
ORDER BY count(o.OID) DESC;
```



Data Output | Explain | Messages | History

	pid character (3)	firstname text	lastname text	totalorders bigint
1	p01	Pearl	Amin	1
2	p04	John	Doe	1
3	p05	Dhyan	Amin	1
4	p03	Jane	Deer	1
5	p02	Alan	Labouseur	1

Trigger

This trigger is called when there is an update in the LineItems table. This trigger ensures that any updates are immediately shown in the results so that Pearl by Pearl has the most accurate information available on items at all times.

```
CREATE OR REPLACE FUNCTION totalsalesUSD () returns trigger as
$$
DECLARE
        newTotal int;
        cursor   refcursor;
BEGIN
        open cursor for
        select sum(qty*unitpriceUSD) as "TotalSalesUSD"
        from orders inner join lineitems on orders.oid = lineitems.oid
        where orders.oid = old.oid;
        fetch cursor into newTotal;
        update orders
         set new.totalSalesUSD = newTotal
        where orders.oid = old.oid;
end;
$$
LANGUAGE plpgsql;
```

```
CREATE TRIGGER check_totalsales
AFTER UPDATE on LineItems
FOR EACH ROW
EXECUTE PROCEDURE totalsalesUSD();
```

User Roles

There are three user roles that have access to this database: the admin, the CEO, and the manager, all of which happen to be one person: Pearl Amin.

When she decides to hire people for the role of the admin and the role of the manager, the following will be the user roles:

User Roles

The Admin has administrative power over all of the Pearl by Pearl database.

```
grant all on all tables in schema public to admin;
```

User Roles

The Manager has complete power over the employees. The manager cannot delete people and customers. The Manager can only select people, customers, lineitems, products, productingredients, ingredients, category, and orders.

```
grant SELECT, INSERT, UPDATE, DELETE on employees to managers;
grant SELECT, INSERT, UPDATE on people to managers;
grant SELECT, INSERT, UPDATE on customers to managers;
grant SELECT, INSERT, UPDATE lineitems to managers;
grant SELECT, INSERT, UPDATE products to managers;
grant SELECT, INSERT, UPDATE productingredients to managers;
grant SELECT, INSERT, UPDATE ingredients to managers;
grant SELECT, INSERT, UPDATE category to managers;
grant SELECT on orders to managers;
```

User Roles

The CEO, Pearl Amin, has the ability to make any select, insert, update, delete* or view all of the tables.
*However, the CEO cannot delete customers.

```
grant SELECT, INSERT, UPDATE, DELETE on managers to CEO;
grant SELECT, INSERT, UPDATE, DELETE on employees to CEO;
grant SELECT, INSERT, UPDATE, DELETE on people to CEO;
grant SELECT, INSERT, UPDATE, on customers to CEO;
grant SELECT, INSERT, UPDATE, DELETE on orders to CEO;
grant SELECT, INSERT, UPDATE, DELETE on lineitems to CEO;
grant SELECT, INSERT, UPDATE, DELETE on zipcode to CEO;
grant SELECT, INSERT, UPDATE, DELETE products to CEO;
grant SELECT, INSERT, UPDATE, DELETE productingredients to CEO;
grant SELECT, INSERT, UPDATE, DELETE ingredients to CEO;
grant SELECT, INSERT, UPDATE, DELETE category to CEO;
```

Company Exit

When Pearl Amin decides to exit her company, the following will occur:

```
revoke all on all tables in schema public from CEO;
```

Known Problems/ Future Enhancements

- Currently only every user has only placed one order. Our sales and marketing team are working on utilizing social media in order to increase sales, our outreach, and retain customers without having to create a marketing budget.
- Our profits are so high for a few reasons: our raw materials are so cheap that they are practically free, we buy in bulk to reduce costs, and our accounts payables are timely so we receive special discounts on top all the other price minimizers.
- We have no need for more triggers at this time because they are complicated and at Pearl by Pearl we believe in simplicity, with only the best and most necessary items so they we can provide the purest standards in all things we produce and handle.
- We are currently working on creating more intricate and elaborate queries so that we can obtain even more meaningful information what what we have now, diving further into the small details such as what ingredients can be replaced since they are only used in a small amount of products that do not sell as well as our best sellers
- With our expanding customer base, this database to continue to expand and evolve as Pearl by Pearl does, for example, by adding more customer demanded products
- Since Pearl by Pearl is a 3D printing custom cosmetics company, inventory is more difficult to keep track of since it depends on the raw materials and the amounts used per products. If time and resources allowed, it would be a great addition to the database to add quality and value.
- Due to the immense amount of SQL users that could not stand NJ and CT having zip codes that started with a 0, NJ and CT decided to change their zip codes to now start with 1, although, do not fret, they are still listed as text in the create statements of this database.