FANATIC FASHION

Database Design Specification

April 15, 2016

Tashi Palden

Table of Contents

Executive Summary

Overview

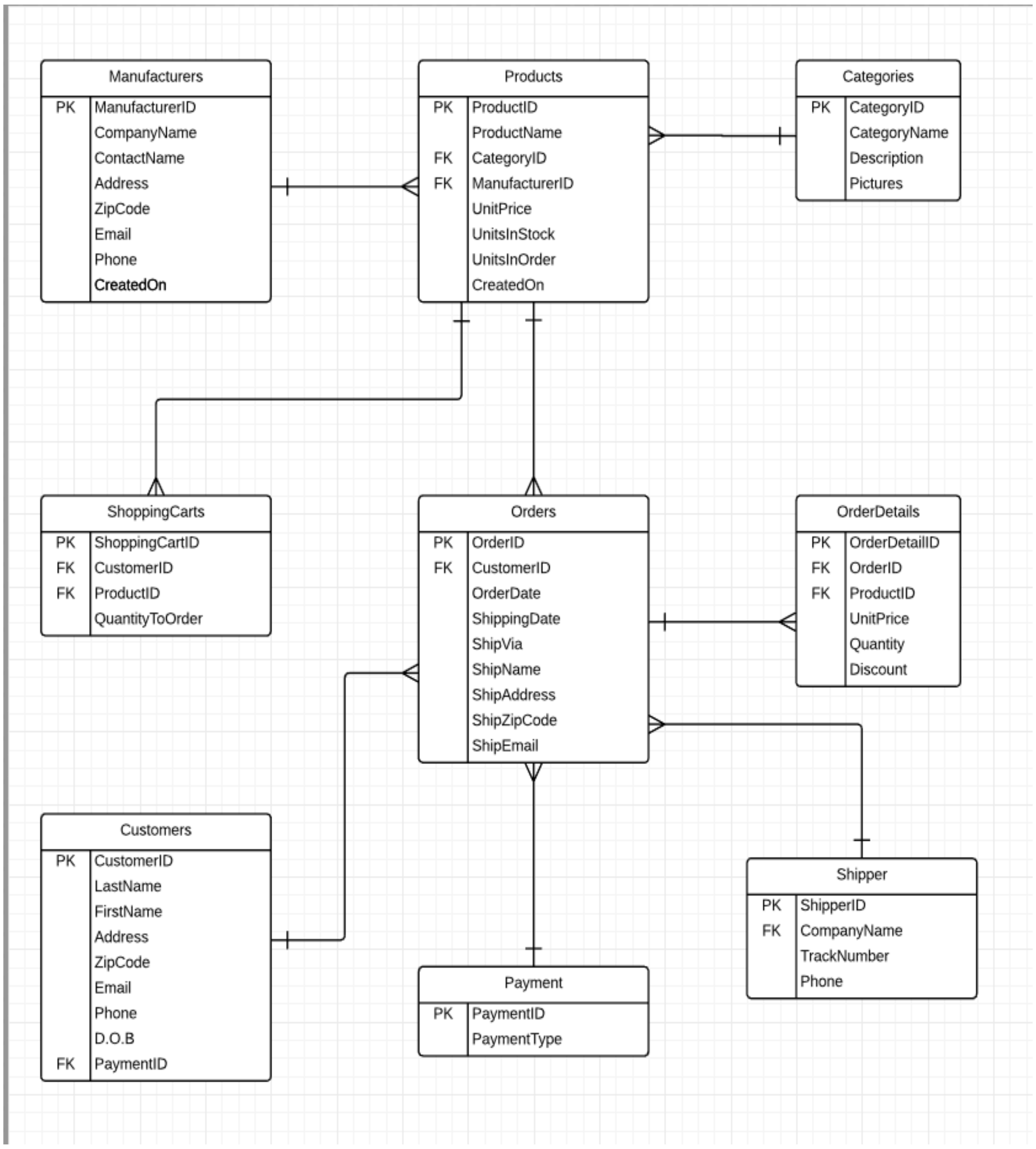
As the U.S. economy continues to grow and evolve, one of the many bright spots of our new creative economy is the fashion industry. Fashion is a $1.2 trillion global industry, with more than $250 billion spent annually on fashion in the United States, according to industry analysts. As the technology advances, people became lazier and they started shopping online instead of shopping in store. Mintel's online shopping US 2015 report reveals that over two thirds (69 percent) of US adults shop online.

Since the Fashion industry is a huge market, and the number of people started shopping online rose, our company (Fanatic Fashion) has good chances of making an impact. To do so, we have to come up with better and more reliable database system.

Objectives

The purpose of this document is to outline a database system to record the customer's information, manufacturer's information, order information, product information, and payment information. This allows the administrator to keep track of all the sales records, and transaction records. This document will provide an overview of the database, along with technical and implementation details including functional dependencies, views, reports, stored procedures, triggers, and security. This system was designed and tested on PostrgreSQL 9.2.5.

ENTITY RELATIONSHIP DIAGRAM



MANUFACTURERS

Purpose

This table holds the information of all the manufacturing companies such as company name, address, email and phone number.

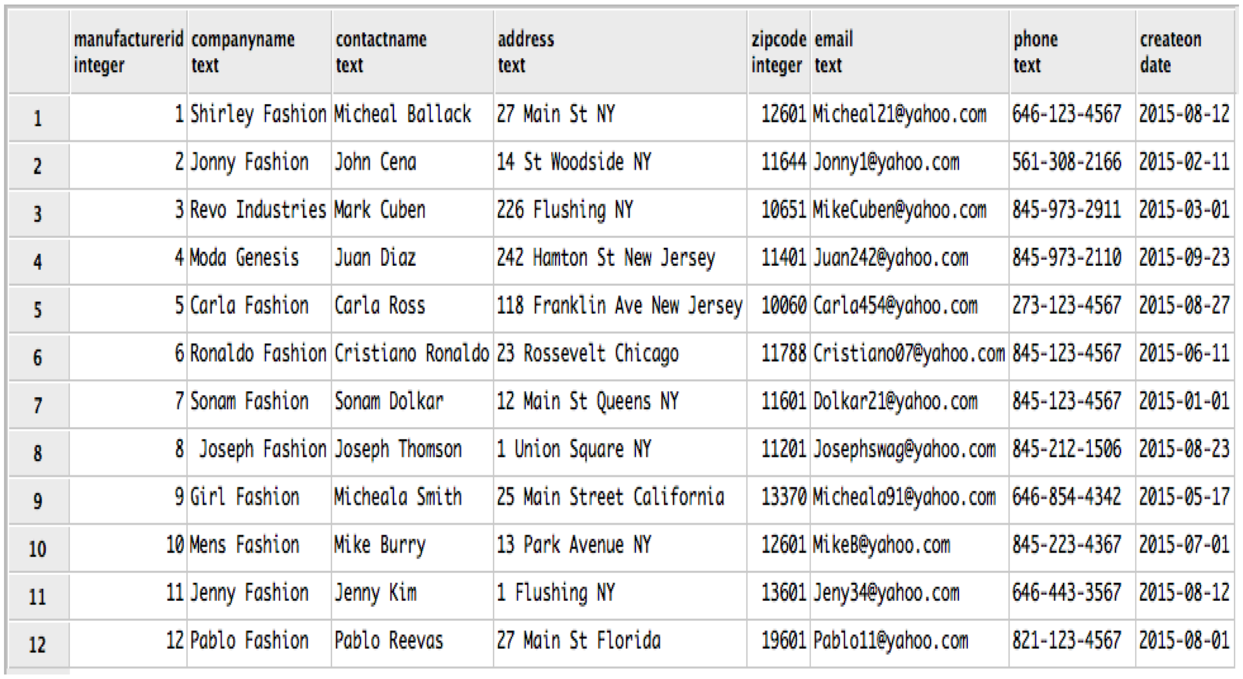
Create Statement

```
CREATE TABLE Manufacturers (
ManufacturerID int not null,
CompanyName text not null,
ContactName text not null,
Address text not null,
ZipCode int not null,
Email text not null,
Phone text not null,
CreateOn date not null,
PRIMARY KEY (ManufacturerID)
);
```

Functional Dependencies:

ManufacturerID → CompanyName, ContactName, Address, ZipCode, Email, Phone, CreateOn

Sample Data



	manufacturerid integer	companyname text	contactname text	address text	zipcode integer	email text	phone text	createon date
1	1	Shirley Fashion	Micheal Ballack	27 Main St NY	12601	Micheal21@yahoo.com	646-123-4567	2015-08-12
2	2	Jonny Fashion	John Cena	14 St Woodside NY	11644	Jonny1@yahoo.com	561-308-2166	2015-02-11
3	3	Revo Industries	Mark Cuben	226 Flushing NY	10651	MikeCuben@yahoo.com	845-973-2911	2015-03-01
4	4	Moda Genesis	Juan Diaz	242 Hamton St New Jersey	11401	Juan242@yahoo.com	845-973-2110	2015-09-23
5	5	Carla Fashion	Carla Ross	118 Franklin Ave New Jersey	10060	Carla454@yahoo.com	273-123-4567	2015-08-27
6	6	Ronaldo Fashion	Cristiano Ronaldo	23 Rossevelt Chicago	11788	Cristiano07@yahoo.com	845-123-4567	2015-06-11
7	7	Sonam Fashion	Sonam Dolkar	12 Main St Queens NY	11601	Dolkar21@yahoo.com	845-123-4567	2015-01-01
8	8	Joseph Fashion	Joseph Thomson	1 Union Square NY	11201	Josephswag@yahoo.com	845-212-1506	2015-08-23
9	9	Girl Fashion	Micheala Smith	25 Main Street California	13370	Micheala91@yahoo.com	646-854-4342	2015-05-17
10	10	Mens Fashion	Mike Burry	13 Park Avenue NY	12601	MikeB@yahoo.com	845-223-4367	2015-07-01
11	11	Jenny Fashion	Jenny Kim	1 Flushing NY	13601	Jeny34@yahoo.com	646-443-3567	2015-08-12
12	12	Pablo Fashion	Pablo Reevas	27 Main St Florida	19601	Pablo11@yahoo.com	821-123-4567	2015-08-01

PRODUCTS

Purpose

This table holds the information of all the products such as product name, unit price, unit in stock, unit in order, product created on.

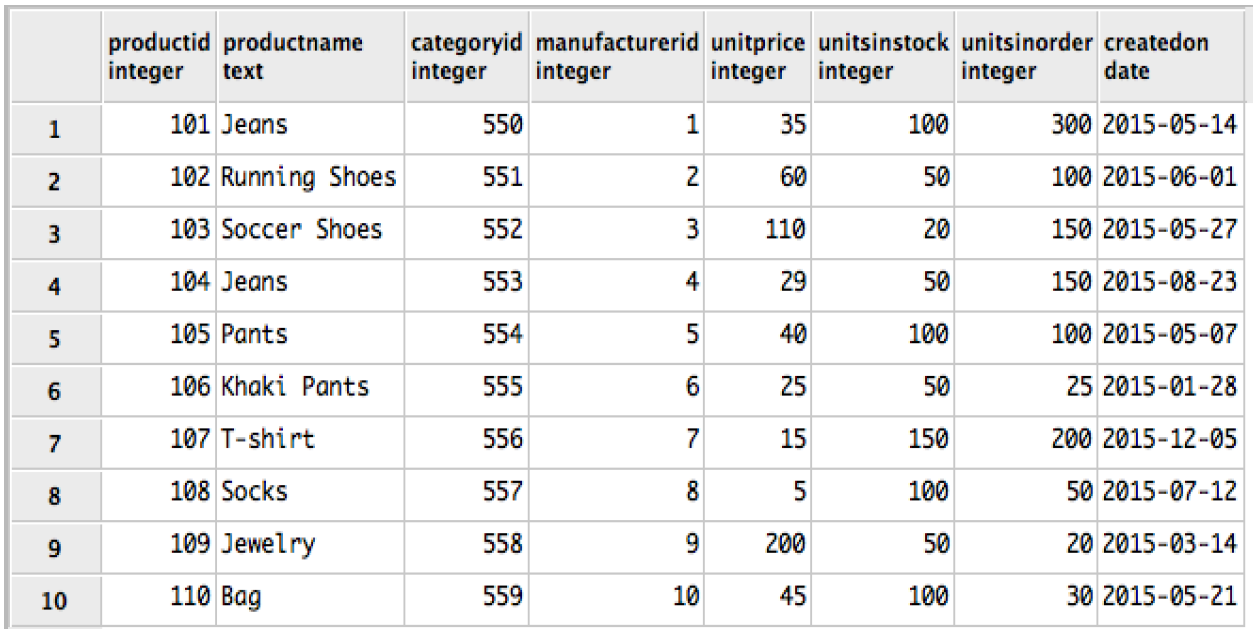
Create Statement

```
CREATE TABLE Product (
ProductID int not null,
ProductName text not null,
CategoryID int not null,
ManufacturerID int not null,
UnitPrice int not null,
UnitsInStock int not null,
UnitsInOrder int not null,
CreatedOn date not null
PRIMARY KEY (ProductID)
);
```

Functional Dependencies:

ProductID → ProductName, CategoryID, ManufacturerID, UnitPrice, UnitsInStock, UnitsInOrder, CreateOn

Sample Data



	productid integer	productname text	categoryid integer	manufacturerid integer	unitprice integer	unitsinstock integer	unitsinorder integer	createdon date
1	101	Jeans	550	1	35	100	300	2015-05-14
2	102	Running Shoes	551	2	60	50	100	2015-06-01
3	103	Soccer Shoes	552	3	110	20	150	2015-05-27
4	104	Jeans	553	4	29	50	150	2015-08-23
5	105	Pants	554	5	40	100	100	2015-05-07
6	106	Khaki Pants	555	6	25	50	25	2015-01-28
7	107	T-shirt	556	7	15	150	200	2015-12-05
8	108	Socks	557	8	5	100	50	2015-07-12
9	109	Jewelry	558	9	200	50	20	2015-03-14
10	110	Bag	559	10	45	100	30	2015-05-21

SHOPPING CARTS

Purpose

This table holds the information of how much quantity is stored in shopping cart so that they can buy later.

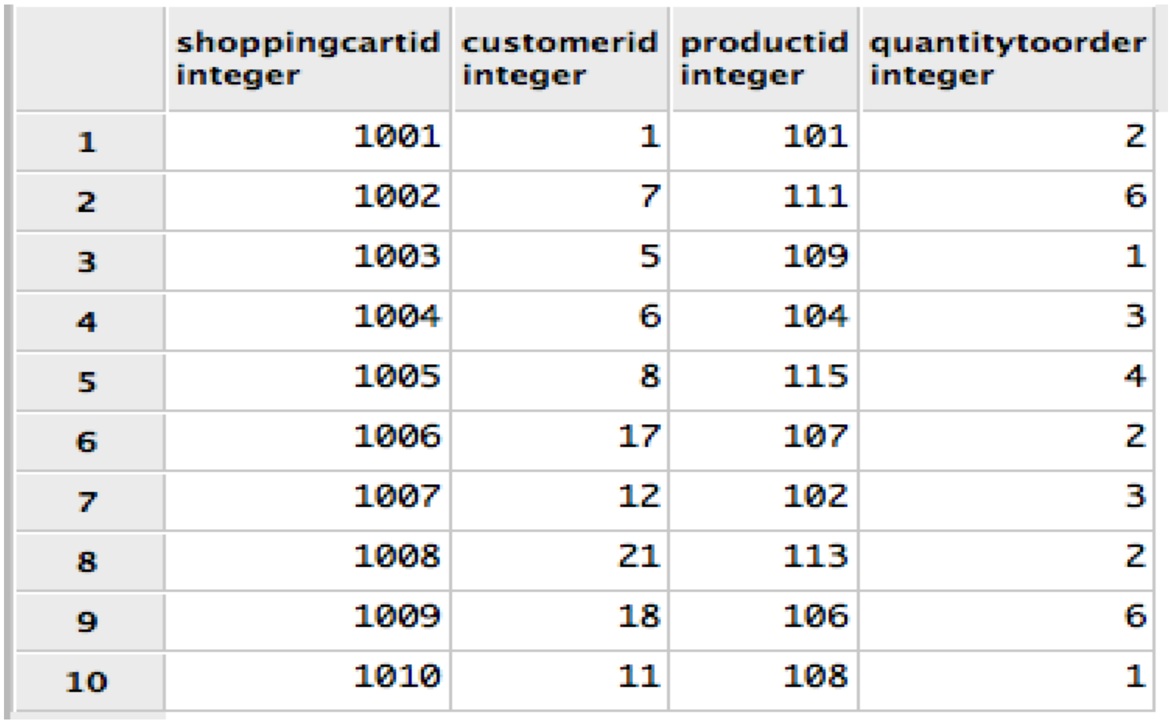
Create Statement

CREATE TABLE ShoppingCarts (
ShoppingCartID **int not null,**
CustomerID **int not null,**
ProductID **int not null,**
QuantityToOrder **int not null,**
PRIMARY KEY (ProductID)
);

Functional Dependencies:

ShoppingCartID → CustomerID, ProductID, QuantityToOrder

Sample Data



	shoppingcartid integer	customerid integer	productid integer	quantitytoorder integer
1	1001	1	101	2
2	1002	7	111	6
3	1003	5	109	1
4	1004	6	104	3
5	1005	8	115	4
6	1006	17	107	2
7	1007	12	102	3
8	1008	21	113	2
9	1009	18	106	6
10	1010	11	108	1

ORDERS

Purpose

This table holds the information of orders that customers made, such as order ID, order date, shipping date, shipping address and email.

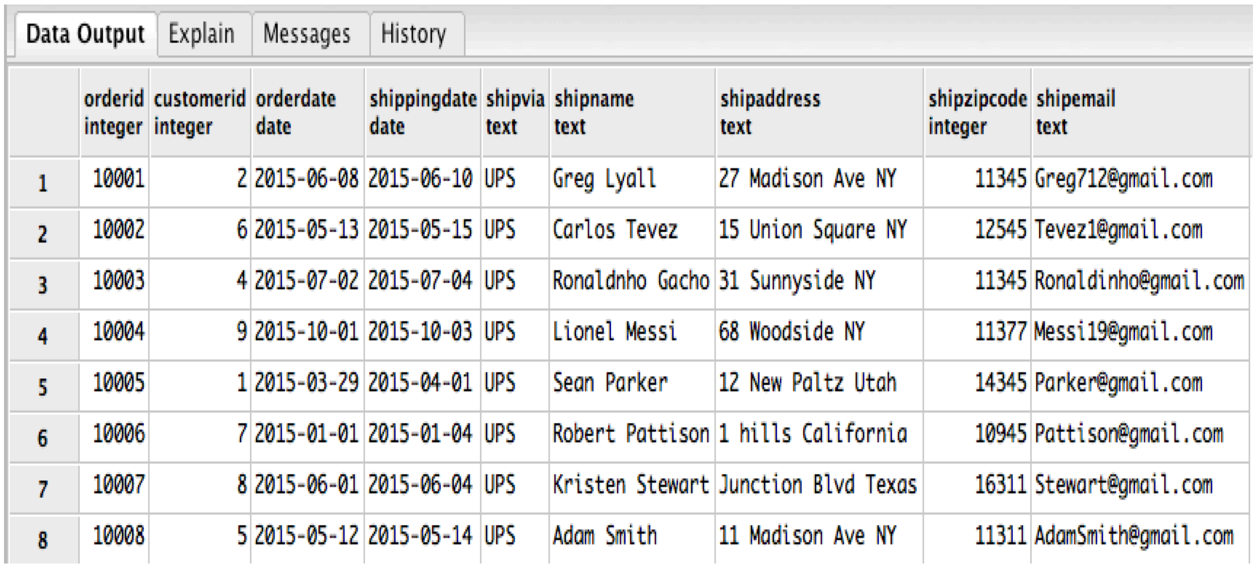
Create Statement

CREATE TABLE Orders **(**
OrderID INT **not null,**
CustomerID **int not null,**
OrderDate **date not null,**
ShippingDate **Date not null,**
ShipVia **text not null,**
ShipName **text not null,**
ShipAddress **text not null,**
ShipZipCode **int not null,**
ShipEmail **text not null,**
PRIMARY KEY (OrderID)
);

Functional Dependencies:

OrderID → CustomerID, EmployeeID, OrderDate, OrderDate, ShippingDate, ShipVia, ShipName, ShipAddress, ShipZipCode, ShipEmail

Sample Data



Data Output | Explain | Messages | History

	orderid integer	customerid integer	orderdate date	shippingdate date	shipvia text	shipname text	shipaddress text	shipzipcode integer	shipemail text
1	10001	2	2015-06-08	2015-06-10	UPS	Greg Lyall	27 Madison Ave NY	11345	Greg712@gmail.com
2	10002	6	2015-05-13	2015-05-15	UPS	Carlos Tevez	15 Union Square NY	12545	Tevez1@gmail.com
3	10003	4	2015-07-02	2015-07-04	UPS	Ronaldnho Gacho	31 Sunnyside NY	11345	Ronaldinho@gmail.com
4	10004	9	2015-10-01	2015-10-03	UPS	Lionel Messi	68 Woodside NY	11377	Messi19@gmail.com
5	10005	1	2015-03-29	2015-04-01	UPS	Sean Parker	12 New Paltz Utah	14345	Parker@gmail.com
6	10006	7	2015-01-01	2015-01-04	UPS	Robert Pattison	1 hills California	10945	Pattison@gmail.com
7	10007	8	2015-06-01	2015-06-04	UPS	Kristen Stewart	Junction Blvd Texas	16311	Stewart@gmail.com
8	10008	5	2015-05-12	2015-05-14	UPS	Adam Smith	11 Madison Ave NY	11311	AdamSmith@gmail.com

ORDER DETAILS

Purpose

This table holds the information of all the order details of customers such as unit price, quantity, and discount.

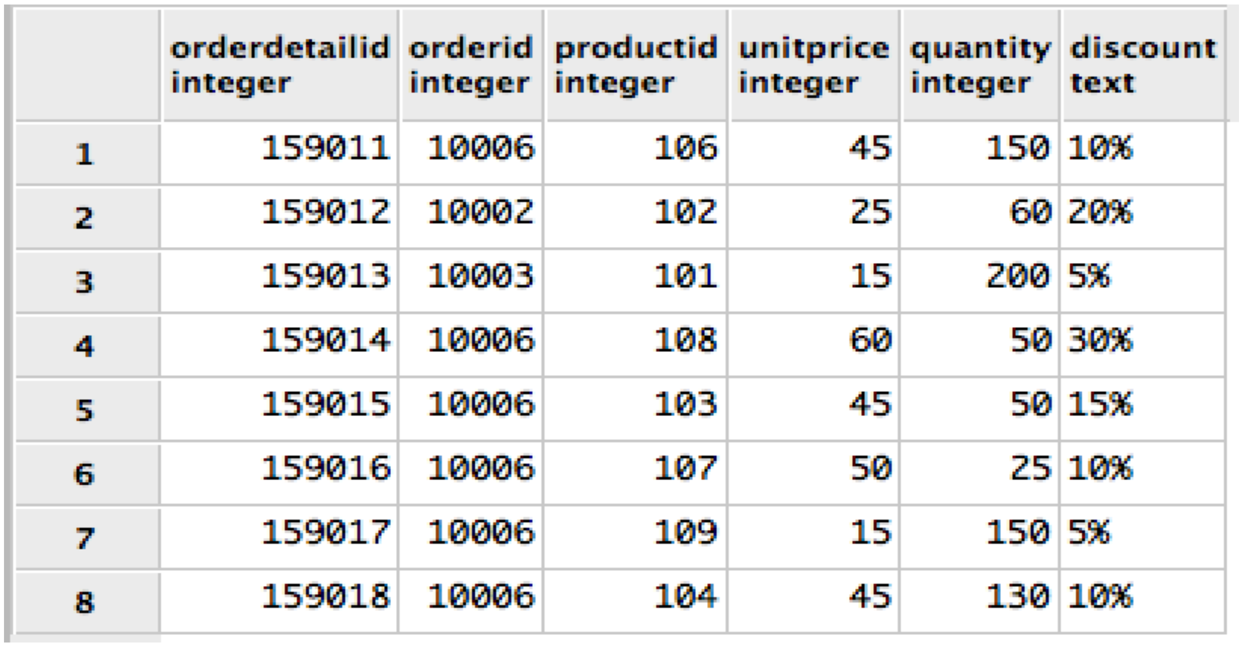
Create Statement

CREATE TABLE OrderDetails **(**
OrderDetailID **int not null,**
OrderID **int not null,**
ProductID **int not null,**
UnitPrice **int not null,**
Quantity **int not null,**
Discount **text not null,**
PRIMARY KEY (OrderDetailID)
);

Functional Dependencies:

OrderDetailID → OrderID, ProductID, UnitPrice, Quantity, Discount

Sample Data



	orderdetailid integer	orderid integer	productid integer	unitprice integer	quantity integer	discount text
1	159011	10006	106	45	150	10%
2	159012	10002	102	25	60	20%
3	159013	10003	101	15	200	5%
4	159014	10006	108	60	50	30%
5	159015	10006	103	45	50	15%
6	159016	10006	107	50	25	10%
7	159017	10006	109	15	150	5%
8	159018	10006	104	45	130	10%

CUSTOMERS

Purpose

This table holds the information of all customers such as their first name, last name, address, email, date of birth, phone number.

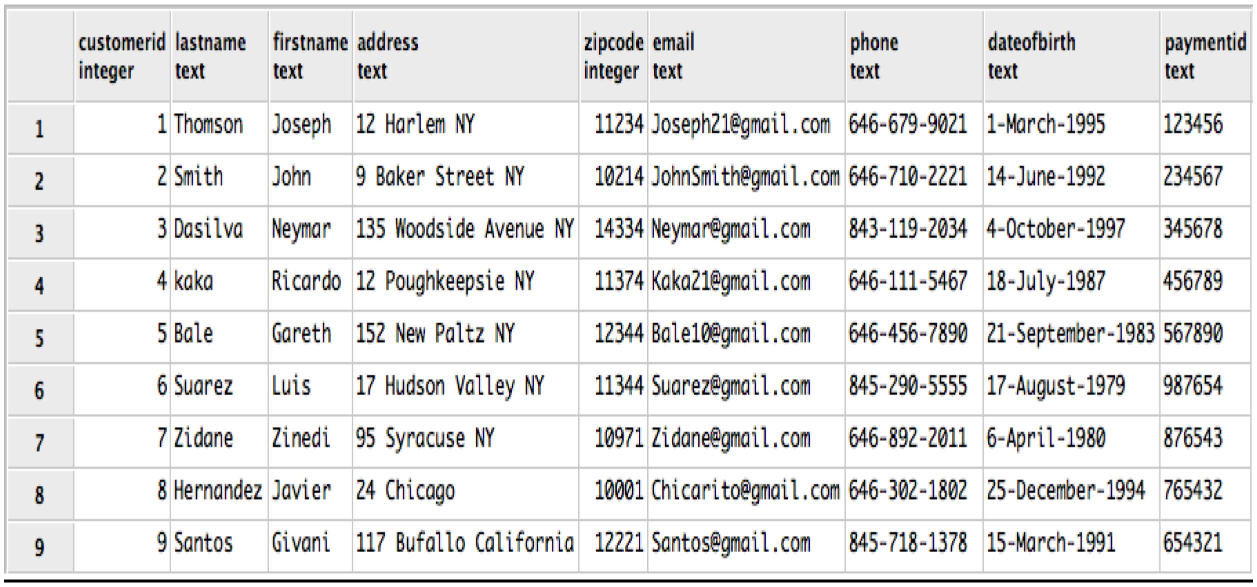
Create Statement

CREATE TABLE Customers (
CustomerID **int not null,**
LastName **text not null,**
FirstName **text not null,**
Address **text not null,**
ZipCode **int not null,**
Email **text not null,**
Phone **text not null,**
DateOfBirth **text not null,**
PaymentID **text not null,**
PRIMARY KEY (CustomerID)
);

Functional Dependencies:

CustomerID → LastName, FirstName, Address, ZipCode, Email, Phone, DateOfBirth, Payment

Sample Data



	customerid integer	lastname text	firstname text	address text	zipcode integer	email text	phone text	dateofbirth text	paymentid text
1	1	Thomson	Joseph	12 Harlem NY	11234	Joseph21@gmail.com	646-679-9021	1-March-1995	123456
2	2	Smith	John	9 Baker Street NY	10214	JohnSmith@gmail.com	646-710-2221	14-June-1992	234567
3	3	Dasilva	Neymar	135 Woodside Avenue NY	14334	Neymar@gmail.com	843-119-2034	4-October-1997	345678
4	4	kaka	Ricardo	12 Poughkeepsie NY	11374	Kaka21@gmail.com	646-111-5467	18-July-1987	456789
5	5	Bale	Gareth	152 New Paltz NY	12344	Bale10@gmail.com	646-456-7890	21-September-1983	567890
6	6	Suarez	Luis	17 Hudson Valley NY	11344	Suarez@gmail.com	845-290-5555	17-August-1979	987654
7	7	Zidane	Zinedi	95 Syracuse NY	10971	Zidane@gmail.com	646-892-2011	6-April-1980	876543
8	8	Hernandez	Javier	24 Chicago	10001	Chicarito@gmail.com	646-302-1802	25-December-1994	765432
9	9	Santos	Givani	117 Bufallo California	12221	Santos@gmail.com	845-718-1378	15-March-1991	654321

SHIPPERS

Purpose

This table holds the information of the company that ships our company products such as company name, phone number and track number.

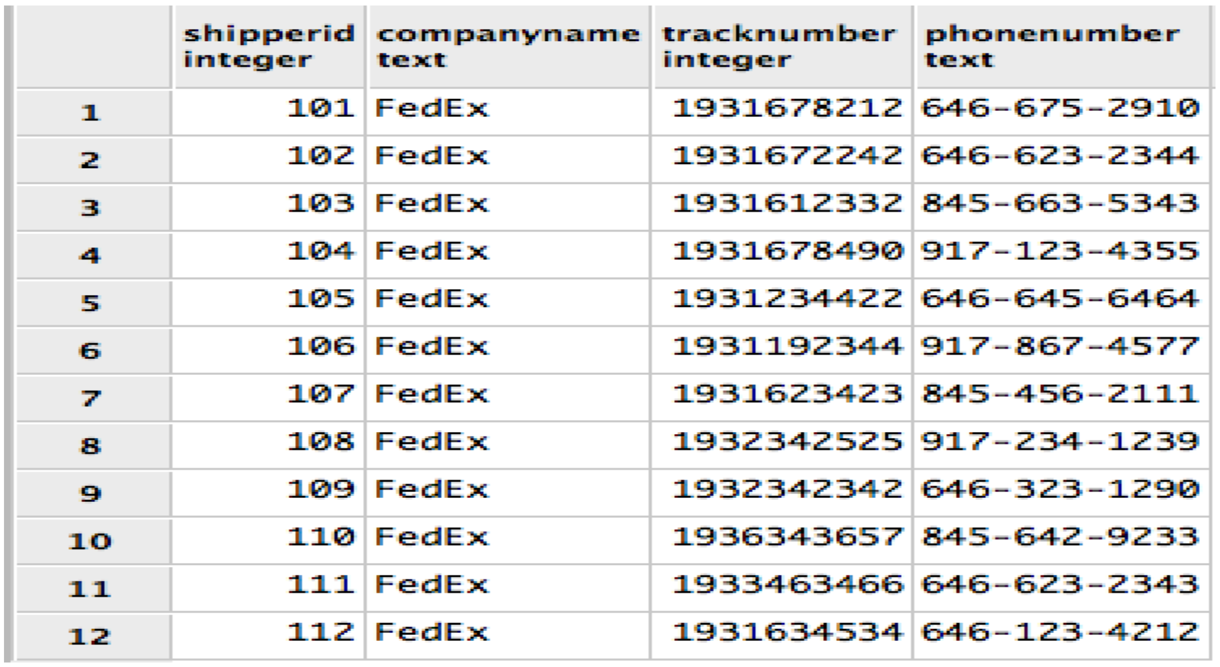
Create Statement

CREATE TABLE shippers **(**
ShipperID **int not null,**
CompanyName **text not null,**
TrackNumber **int not null,**
PhoneNumber **text not null**,
PRIMARY KEY (ShipperID)
);

Functional Dependencies:

ShipperID → CompanyName, TrackNumber, PhoneNumber

Sample Data



	shipperid integer	companyname text	tracknumber integer	phonenumber text
1	101	FedEx	1931678212	646-675-2910
2	102	FedEx	1931672242	646-623-2344
3	103	FedEx	1931612332	845-663-5343
4	104	FedEx	1931678490	917-123-4355
5	105	FedEx	1931234422	646-645-6464
6	106	FedEx	1931192344	917-867-4577
7	107	FedEx	1931623423	845-456-2111
8	108	FedEx	1932342525	917-234-1239
9	109	FedEx	1932342342	646-323-1290
10	110	FedEx	1936343657	845-642-9233
11	111	FedEx	1933463466	646-623-2343
12	112	FedEx	1931634534	646-123-4212

CATEGORIES

Purpose

This table holds the information of all the categories of product such as categories name, description and pictures.

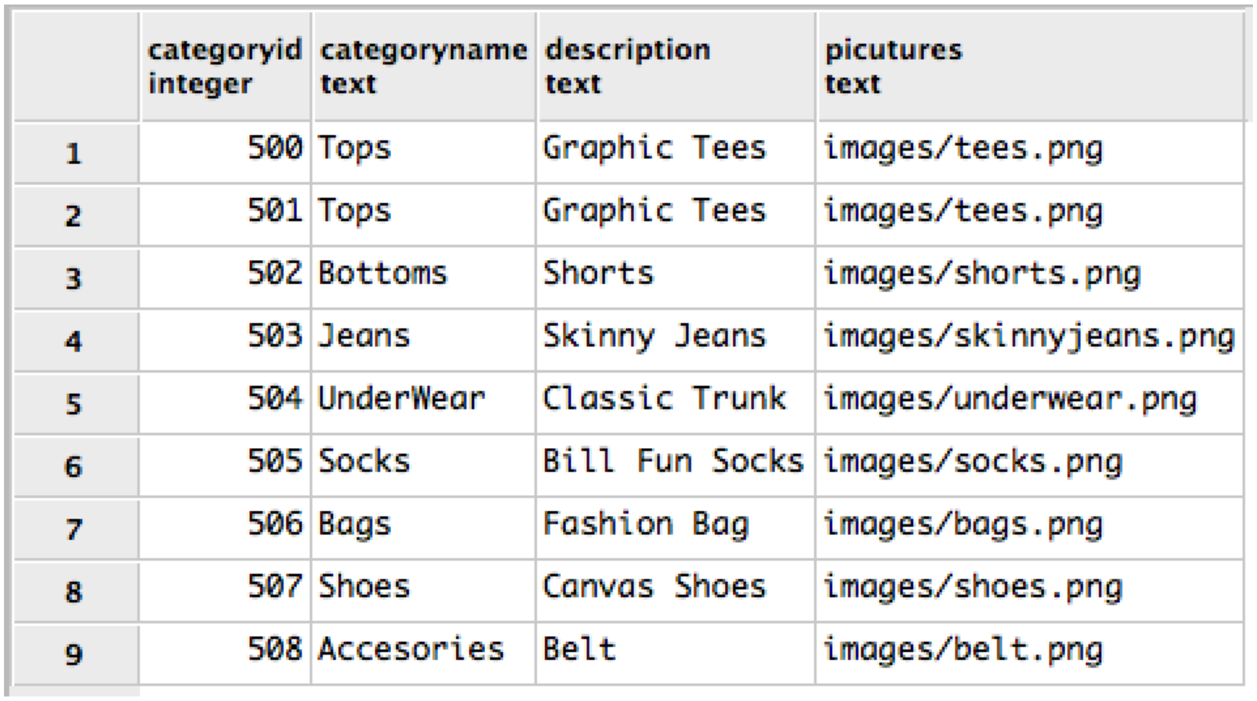
Create Statement

CREATE TABLE Categories **(**
CategoryID **int not null,**
CategoryName **text not null,**
Description **text not null,**
Pictures **text not null,**
PRIMARY KEY (CategoryID)
);

Functional Dependencies:

CategoryID → CategoryName, Description, Pictures

Sample Data



	categoryid integer	categoryname text	description text	picutures text
1	500	Tops	Graphic Tees	images/tees.png
2	501	Tops	Graphic Tees	images/tees.png
3	502	Bottoms	Shorts	images/shorts.png
4	503	Jeans	Skinny Jeans	images/skinnyjeans.png
5	504	UnderWear	Classic Trunk	images/underwear.png
6	505	Socks	Bill Fun Socks	images/socks.png
7	506	Bags	Fashion Bag	images/bags.png
8	507	Shoes	Canvas Shoes	images/shoes.png
9	508	Accesories	Belt	images/belt.png

PAYMENT

Purpose

This table holds the information of payment type.

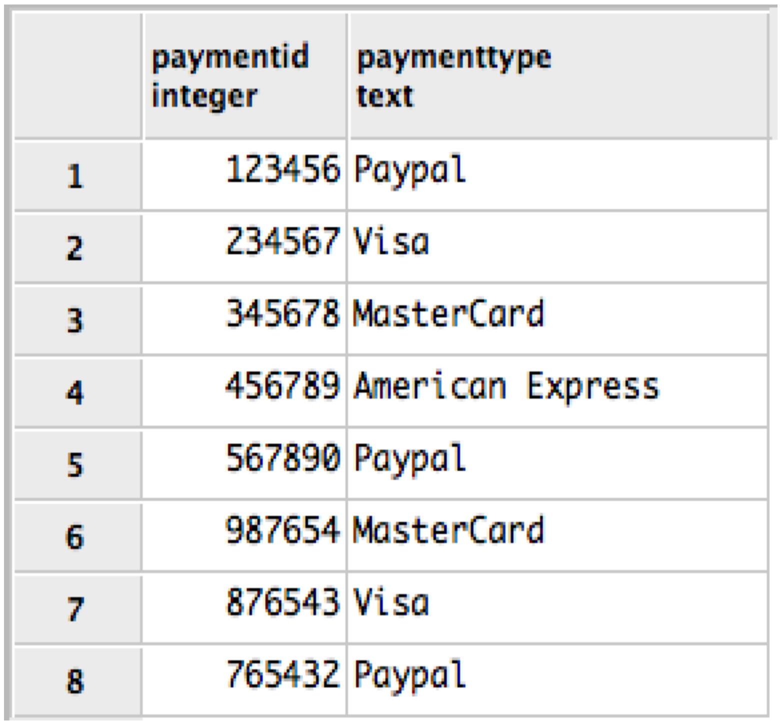
Create Statement

CREATE TABLE Payment **(**
PaymentID **int not null,**
PaymentType **text not null,**
PRIMARY KEY (PaymentID)
);

Functional Dependencies:

PaymentID → PaymentType

Sample Data



	paymentid integer	paymenttype text
1	123456	Paypal
2	234567	Visa
3	345678	MasterCard
4	456789	American Express
5	567890	Paypal
6	987654	MasterCard
7	876543	Visa
8	765432	Paypal

VIEWS

CheapestProduct

Purpose

Most of the people doesn’t have enough money to buy products such as clothes, shoes, and bags etc. They always look for the cheapest price. This view displays the cheapest product to expensive product in ascending order.

Create Statement

```
CREATE VIEW CheapestProduct AS
SELECT   ProductID,
         ProductName,
         UnitPrice
FROM     Products
ORDER BY  UnitPrice ASC;
```

ExpensiveProduct

Purpose

Some rich people doesn’t want to buy cheap clothes so they always look for expensive product. This view displays the most expensive product to cheap product in descending order.

Create Statement

```
CREATE VIEW ExpensiveProduct AS
SELECT   ProductID,
         ProductName,
         UnitPrice
FROM     Products
ORDER BY  UnitPrice DESC;
```

Tops

Purpose

Some people just want to check out tops. They don't want to go through all the products. This views provides a list of different tops such as Tees, Graphic Tees, Shirts, Sweaters, Hoodies and Jacket.

Create Statement

```
CREATE VIEW Tops AS
SELECT   CategoryName,
         Description,
         Pictures
FROM     Categories
WHERE    CategoryName = 'Tops';
```

Shoes

Purpose

Some people just want to check out shoes. This views provides a list of different shoes such as lace-up sneaker, boat shoes, boots, and sandals.

Create Statement

```
CREATE VIEW Shoes AS
SELECT   CategoryName,
         Description,
         Pictures
FROM     Categories
WHERE    CategoryName = 'Shoes';
```

MoreThanTwoProduct

Purpose

This views provides a list of orders that has 2 or more products.

Create Statement

```
CREATE VIEW MoreThanTwoProduct AS
SELECT   OrderID,
         Quantity,
         UnitPrice
FROM     OrderDetails
WHERE    Quantity > 2;
```

HundredsDollarsOrMore

Purpose

This views provides a list of orders that purchased 100 dollars products or more.

Create Statement

```
CREATE VIEW FiftyDollarsOrMore AS
SELECT   OrderID,
         ProductID,
         UnitPrice,
         Quantity
FROM     OrderDetails
WHERE    UnitPrice > 50;
```

REPORTS

Units In Stock

Purpose

It is really important to know how much product is left in stock so that the manager can get an idea about whether to order more or not. This report shows how much product in left in stock.

Query

```
SELECT    ProductID,
          ProductName,
          UnitPrice,
          UnitsInStock
FROM      Product
ORDER BY UnitsInStock  DESC
```

Discount Report

Purpose

It is important to know the discount rate of the product. This report shows the discount rate of the product.

Query

```
SELECT    OrderID,
          ProductID,
          UnitPrice,
          Quantity,
          Discount
FROM      OrderDetails
ORDER BY Discount DESC
```

Product Sold

Purpose

It is important to know how much product quantity is sold. This report shows the number of product quantity sold and their unit price.

Query

```
SELECT    OrderID,
          ProductID,
          UnitPrice,
          Quantity,
FROM      OrderDetails
ORDER BY Quantity DESC
```

Shipping Report

Purpose

Keeping track of order date and shipping date is important. This report shows order date and shipping date of product.

Query

```
SELECT    OrderID,
          CustomerID,
          OrderDate,
          ShippingDate
FROM      Orders
ORDER BY OrderDate by ASC
```

STORED PROCEDURES

Add_ShoppingCarts

Purpose

It is important to have add to shopping cart. People usually add to cart first before they purchase it. Add to shopping cart temporarily creates virtual shopping cart and you only need to pay when you go to checkout and confirm your purchase. Add to shopping cart makes it easy for customer to purchase a lot of things together.

Query

```
CREATE OR REPLACE FUNCTION add_shoppingCarts
RETURNS trigger AS
$BODY$
   BEGIN
       IF NEW.ProductID is NULL THEN
       RAISE EXCEPTION ‘Invalid ProductID provided’:
       END IF;
       INSERT INTO ShoppingCarts (ProductID, QuantityToOrder)
                             VALUES (NEW. ProductID, ‘3’);
       RETURN NEW;
    END;
$$ LANGUAGE plpgsql;
```

Add_OrderDateAndShippingDate

Purpose

Adding order date and shipping date is really important. Keeping track of order date and shipping date will let you know when the company got order and when they shifted it. It also helps you keep track of the number of order received.

Query

```
CREATE OR REPLACE FUNCTION add_OrderDateAndShippingDate
RETURNS trigger AS
$BODY$
   BEGIN
       IF NEW.OrderID is NULL THEN
        RAISE EXCEPTION ‘Invalid OrderID provided’:
       END IF;
       INSERT INTO Orders (OrderID, CustomerID, OrderDate, ShippingDate)
                             VALUES (NEW. OrderID, 8, ‘8-March-2015’, ’10-March-2015’);
       RETURN NEW;
    END;
$$ LANGUAGE plpgsql;
```

TRIGGERS

FixUnitPriceTrigger

Purpose

The prices of product always change in quick succession. The company have to update the price when it happens. This trigger updates the unit price of the products.

Query

```
CREATE TRIGGER FixUnitPriceTrigger
BEFORE INSERT ON UnitPrice
REFERENCING
        NEW ROW AS NewRow
        NEW TABLE AS NewStuff
FOR EACH ROW
WHEN NewRow.UnitPrice IS NULL
UPDATE NewStuff SET UnitPrice = 30;
```

AddQuantity

Purpose

Add_Quantity on an insert to OrderDetails, this calls the stored procedure add_Quantity() which updates the quantity column of the new insert into the OrderDetails table.

Query

```
CREATE TRIGGER add_Quantity
AFTER INSERT ON OrderDetails
FOR EACH ROW
EXECUTE PROCEDURE add_Quantity();
```

SECURITY

There are four primary users of the database: customers, manufacturers, fanatic fashion administrators, database administrators. For each user role, the user is revoked of all privileges on all tables before being assigned any applicable privileges. These revoke statements are not included for the sake of brevity.

Customers

Customers interact with the database directly by entering into database.

GRANT INSERT ON Customers **TO** FanaticFashionAdministrator;

Fanatic Fashion Administrator

The fanatic fashion administrator is the person responsible for overseeing all store operation including keeping tracks of orders, payment system, and modifying details of customers, manufacturer, order details information.

GRANT SELECT, INSERT, DELETE ON Manufacturer **TO** FanaticFashionAdministrator;
GRANT SELECT, INSERT, DELETE ON Products **TO** FanaticFashionAdministrator;
GRANT SELECT, INSERT, DELETE ON Orders **TO** FanaticFashionAdministrator;
GRANT SELECT, INSERT, DELETE ON Customers **TO** FanaticFashionAdministrator;
GRANT SELECT, INSERT, DELETE ON OrderDetails **TO** FanaticFashionAdministrator;
GRANT SELECT, INSERT, DELETE ON ShoppingCarts **TO** FanaticFashionAdministrator;
GRANT SELECT, INSERT, DELETE ON Payment **TO** FanaticFashionAdministrator;

Manufacturers

Manufacturers interact with database directly by entering into database.

GRANT INSERT ON Manufacturers **TO** FanaticFashionAdministrator;

Database Administrator Role

The database administrator has god-like powers.

GRANT ALL PRIVILEDGES ON ALL TABLES IN SCHEMA public **TO** bdAdministrator;

Implementation Notes

The following are suggestions and/or requirements for implementation:

- The customers must know the ProductName and ProductID to place an order. The customer also has to know the payment type to purchase product.
- The manufacturer must know the ManufacturerID in order to do business with Fanatic Fashion. Knowing ManufacturerID will make it easier to do business. The Fanatic Fashion employees can just look at ManufacturerID and can easily figure out the manufacturer company.

Known Problems

There are few known problems in database. Some of them are:

- If the customer wants to return a product, that will be a problem. Since it is not a real company database, the database doesn't have reshipping command.
- The database doesn't have employee's information because employee information is not so important, compare to order, and customer information.

Future Enhancements

Some features and functionalities that might be desirable in the future:

- Allowing more details about shipment.
- Creating employees table.
- More information about discounts to attract more customer.
- More details about order details.