tutta la merce

NYC

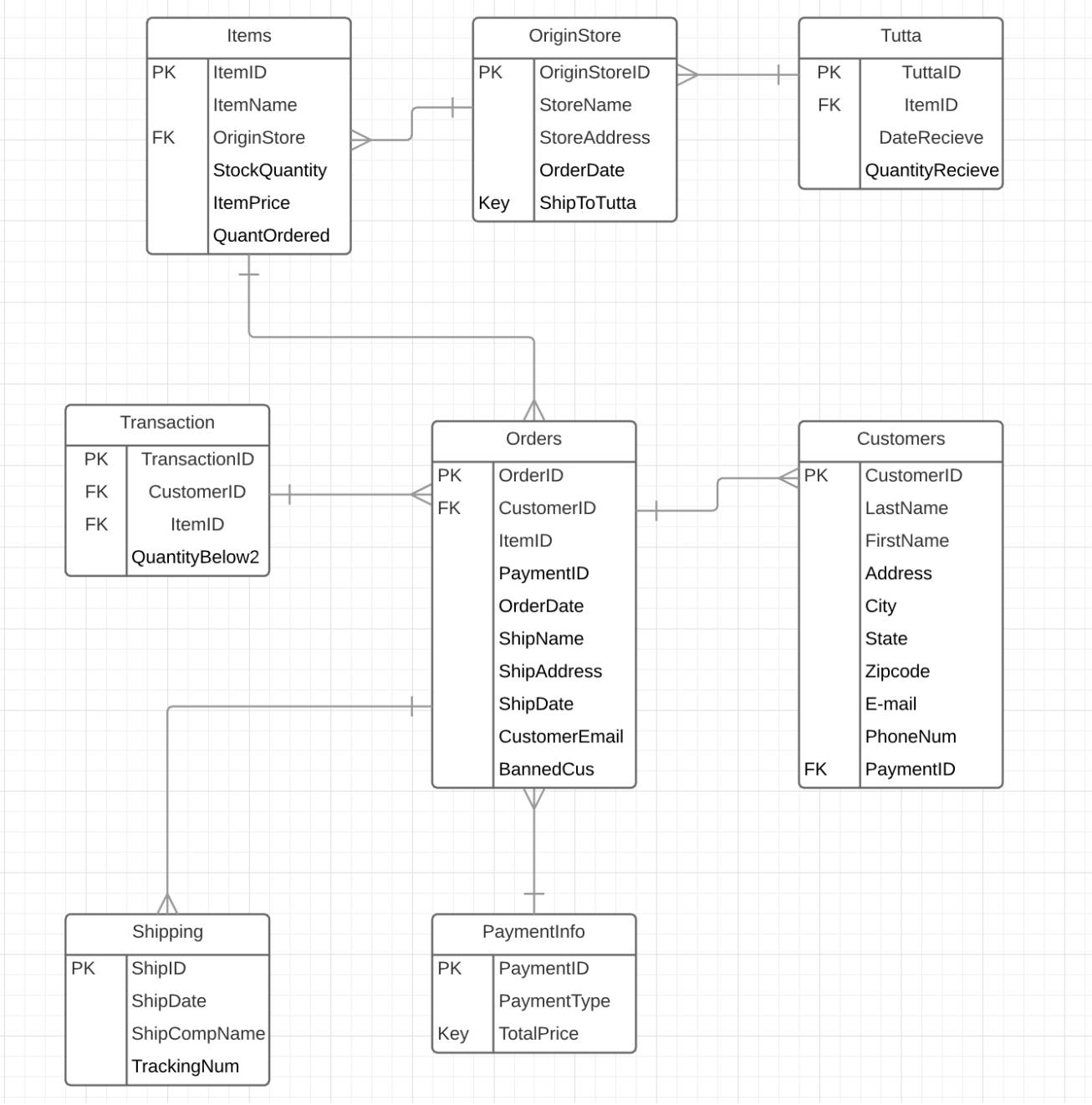
DESIGNED BY: DELROY MATHIESON

Table of Contents

Executive Summary

Tutta la merce, based in New York City is my personal online merchandise reselling business. In 2013 as a sophomore in high school I became fascinated with the business in the sneaker industry. Many sneaker fans are often left empty-handed because of the limited stock quantity from retailers and don't have time spend countless hours waiting on virtual lines only to not get their favorite sneaker. I developed my company around the customers who want exclusive limited released sneakers but don't have the time. I am responsible for all bookkeeping, management, shipping, purchasing, marketing and customer service. Over the last five years I have developed a cliental from eBay while maintaining 100% feedback rating and moved my business to my own online retail chain. Tutta la merce offers excellent service to help customers who desire limited and exclusive sneakers. Tutta la merce is one of the most effective service for my clients to be guaranteed when securing any sneaker, they desire relying heavily on values, principles and exceptional customer service.

ER Diagram


Items
PK ItemID
ItemName
FK OriginStore
StockQuantity
ItemPrice
QuantOrdered
OriginStore
PK OriginStoreID
StoreName
StoreAddress
OrderDate
Key ShipToTutta
Tutta
PK TuttaID
FK ItemID
DateRecieve
QuantityRecieve
Transaction
PK TransactionID
FK CustomerID
FK ItemID
QuantityBelow2
Orders
PK OrderID
FK CustomerID
ItemID
PaymentID
OrderDate
ShipName
ShipAddress
ShipDate
CustomerEmail
BannedCus
Customers
PK CustomerID
LastName
FirstName
Address
City
State
Zipcode
E-mail
PhoneNum
FK PaymentID
Shipping
PK ShipID
ShipDate
ShipCompName
TrackingNum
PaymentInfo
PK PaymentID
PaymentType
Key TotalPrice

Customer Table

- The customer table shows the list of customer information.

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

Functional Dependences
CustomerID → LastName, FirstName, Address, City, State, Zipcode, Email, PhoneNum, PaymentNum, PaymentID

	customerid integer	lastname text	firstname text	address text	city text	state text	zipcode text	email text	phonenum text	paymentid text
1	1	Mathieson	Delroy	3399 North Rd	Poughkeepsie	NY	12601	d.math@yahoo.com	646-764-9284	1
2	2	Smith	Bob	25 South Side	SunnySide	CT	2551	bobby.smith@gmail.com	745-823-8721	2
3	3	James	Rick	956 Beach Ave	Richmond	VA	12601	rickyj@hotmail.com	843-842-4927	3
4	4	Labouseur	Alan	33 Yellow Brick Rd	HollyWood	CA	91210	Alan.Labouseur@marist.edu	345-984-0739	3
5	5	Peters	Mary	123 Park Ave	Bronx	NY	10453	mary.peters@gmail.com	929-937-4810	1
6	6	Brown	Johnny	563 Complex Rd	Tampa	FL	75632	johnnybrown@yahoo.com	819-526-5319	2
7	7	Henry	Mark	45 HighTop	YellowStone	MT	14560	mark.henry@aol.com	435-376-3567	1

Order Table

- The order table displays information about the customers' order.

CREATE TABLE Orders**(**
OrderID **text not null,**
CustomersID **int not null,**
ItemID **text not null,**
PaymentID **int not null,**
OrderDate **text not null,**
ShipName **text not null,**
ShipAddress **text not null,**
ShipDate **text not null,**
CustomerEmail **text not null,**
BannedCus **text not null,**
PRIMARY KEY (OrderID**)**
);

Functional Dependences
OrderID → CustomersID, ItemID, PaymentID, OrderDate, ShipName, ShipAddress, ShipDate, CustomerEmail, BannedCus

	orderid text	customersid integer	itemid text	paymentid text	orderdate text	shipname text	shipaddress text	shipdate text	customeremail text	bannedcus text
1	001	1	A003	1	03-15-13	Delroy Mathieson	3399 North Rd, Poughkeepsie, NY, 12601	03-17-13	d.math@yahoo.com	No
2	002	2	B035	2	01-23-14	Bob Smith	25 South Side, SunnySide,CT, 02551	01-25-14	bobby.smith@gmail.com	Yes
3	003	3	A003	3	09-16-16	Rick James	956 Beach Ave, Richmond, VA, 12601	09-17-16	rickyj@hotmail.com	No
4	004	4	C453	3	07-04-15	Akan Labouseur	33 Yellow Brick Rd, HollyWood, CA, 91210	07-06-15	alan.labouseur@marist.edu	No
5	005	5	C973	1	12-30-15	Mary Peters	123 Park Ave, Bronx, NY, 10453	01-02-16	mary.peters@gmail.com	Yes
6	006	6	B007	2	09-16-16	Johnny Brown	563 Complex Rd, Tampa, NY, 75632	09-16-16	johnnybrown@yahoo.com	No
7	007	7	C555	1	04-19-14	Mark Henry	45 HighTop, YellowStone, MT, 14560	04-22-14	mark.henry@aol.com	No

Shipping Table

- The shipping table has information on the shipping identification number, shipping date, shipping company and tracking number.

CREATE TABLE Shipping**(**
ShipID **text not null,**
ShipDate **text not null,**
ShipCompName **text not null,**
TrackingNum **text not null,**
PRIMARY KEY (ShipID**)**
);

Functional Dependences
ShipID → ShipDate**,** ShipCompName**,** TrackingNum

	shipid text	shipdate text	shipcompname text	trackingnum text
1	U001	03-17-13	USPS	9405809699937119815434
2	B002	01-25-14	UPS	1Z06E18A6836392739
3	F003	09-17-16	FEDEX	624893691092
4	F004	07-06-15	FEDEX	624898264023
5	U002	01-02-16	USPS	9405806854858119815434
6	U003	09-16-16	USPS	9405868658075367815434
7	B003	04-22-14	UPS	1Z06E18A6847193932

Payment Table

- The payment table has payment information, such as the payment ID, the type of payment and total price paid by the customer.

CREATE TABLE PaymentInfo**(**
PaymentID **text not null,**
PaymentType **text not null,**
TotalPrice **text not null,**
PRIMARY KEY (PaymentID)
);

Functional Dependences
PaymentID → PaymentType**,** TotalPrice

	paymentid text	paymenttype text	totalprice text
1	P001	Paypal	$2100
2	V002	Visa	$1600
3	C005	Bitcoin	$10240
4	C006	Bitcoin	$12000
5	P002	Paypal	$3600
6	V003	Visa	$785
7	C007	Bitcoin	$9000

Origin Store Table

- The origin store table has information about which companies I purchased the sneakers from for Tutta.

CREATE TABLE OriginStore**(**
OriginStoreID **text not null,**
StoreName **text not null,**
StoreAddress **text not null,**
OrderDate **text not null,**
ShipToTutta **text not null,**
PRIMARY KEY (OriginStoreID**)**
);

Functional Dependences
OriginStoreID → StoreName, StoreAddress, OrderDate, ShipToTutta

	originstoreid text	storename text	storeaddress text	orderdate text	shiptotutta text
1	AD01	Adidas	2101 E Via Arado Rancho Dominguez, CA 90220	03-12-13	03-13-13
2	N010	Nike	511 NikeTown Rd, TN 32102	01-21-14	01-22-14
3	YE01	YeezySupply	11790 Southampton Ct, Los Angeles, CA 90077	09-15-16	09-15-16
4	AD02	Adidas	2101 E Via Arado Rancho Dominguez, CA 90220	07-04-15	07-04-15
5	AD03	Adidas	2101 E Via Arado Rancho Dominguez, CA 90220	12-27-15	12-28-15
6	N011	Nike	511 NikeTown Rd, TN 32102	09-14-16	09-13-16
7	YE02	YeezySupply	11790 Southampton Ct, Los Angeles, CA 90077	04-17-14	04-16-14

Tutta Table

- The Tutta table as information on the items, date the items were received and the quantity.

CREATE TABLE Tutta**(**
TuttaID **text not null,**
ItemID **text not null,**
DateRecieve **text not null,**
QuantityRecieve **text not null,**
PRIMARY KEY (TuttaID)
);

Functional Dependences
TuttaID → ItemID, DateRecieve, QuantityRecieve

	tuttaid text	itemid text	daterecieve text	quantityrecieve text
1	T001	A003	03-14-13	1000
2	T002	B035	01-23-14	6000
3	T003	A003	09-15-16	300
4	T004	C453	07-05-15	2500
5	T005	C973	12-28-15	3500
6	T006	B007	09-14-16	70
7	T007	C555	04-16-14	604

Items Table

- The items table has information on the items ID, item name, origin store, quantity in stock, listed price and quantity ordered.

CREATE TABLE Items(
ItemID **text not null**,
ItemName **text not null**,
OriginStore **text not null**,
StockQuantity **int not null**,
ItemPrice **text not null**,
QuantOrdered **int not null**,
PRIMARY KEY (ItemID)
);

Functional Dependences
ItemID → ItemName, OriginStore, StockQuantity, ItemPrice, QuantOrdered

	itemid text	itemname text	originstore text	stockquantity integer	itemprice text	quantordered integer
1	A003	Yeezy Boost 350	Adidas	1000	$2100	67
2	B035	Nike Air Jordan 11s	Nike	6000	$1600	53
3	A009	Yeezy Boost 350	YeezySupply	300	$10240	34
4	C453	Yeezy Boost 350	Adidas	2500	$12000	100
5	C973	Yeezy Boost 750s	Adidas	3500	$3600	23
6	B007	Nike Air Jordan 5s	Nike	70	$785	70
7	C555	Yeezy Boost 750	YeezySupply	604	$9000	64

Transaction Tables

- The transaction table has information on the transaction ID, customer ID, item ID and ensures that each transaction only has one item to ensure fairness when purchasing limited merchandise.

CREATE TABLE Transaction**(**
TransactionID **text not null,**
CustomerID **int not null,**
ItemID **int not null,**
QuantityBelow2 **int not null,**
CHECK(QuantityBelow2 < 2 AND QuantityBelow2 > 0),
PRIMARY KEY(TransactionID)
);

Functional Dependences
TransactionID → CustomerID, ItemID, QuantityBelow2

	transactionid text	customerid integer	itemid text	quantitybelow2 integer
1	3a7-D7o-cht-2ut	1	A003	1
2	9aj-X4h-cyD-2r2	2	B035	1
3	nw8-8je-8j2-12u	3	A009	1
4	93n-92n-68k-pi7	4	C453	1
5	35h-lk8-jeu-j32	5	C973	1
6	91p-33d-45f-2ew	6	B007	1
7	2uh-6ge-32e-98w	7	C555	1

Views

- This view displays the name of the sneaker and where the sneaker was bought.

```
CREATE VIEW Brands AS
SELECT OriginStore,
       ItemName
FROM   Items
ORDER BY OriginStore ASC;
```

```
SELECT *
FROM Brands;
```

	originstore text	itemname text
1	Adidas	Yeezy Boost 350
2	Adidas	Yeezy Boost 350
3	Adidas	Yeezy Boost 750s
4	Nike	Nike Air Jordan 5s
5	Nike	Nike Air Jordan 11s
6	YeezySupply	Yeezy Boost 750
7	YeezySupply	Yeezy Boost 350

- This view displays the first name, last name and origin store so I am able to know which store my clients prefer.

```
CREATE VIEW AdidasOrders AS
SELECT c.FirstName, c.LastName, i.OriginStore
FROM Customers c
INNER JOIN Orders o ON c.CustomerID = o.CustomersID
INNER JOIN Items i ON o.ItemID = i.ItemID
GROUP BY c.FirstName, c.LastName, i.OriginStore
ORDER BY i.OriginStore ASC, c.FirstName;
```

```
SELECT *
FROM AdidasOrders;
```

	firstname text	lastname text	originstore text
1	Alan	Labouseur	Adidas
2	Delroy	Mathieson	Adidas
3	Mary	Peters	Adidas
4	Rick	James	Adidas
5	Bob	Smith	Nike
6	Johnny	Brown	Nike
7	Mark	Henry	YeezySupply

- This view shows information on how my clients prefer to purchase their sneakers and how much they paid.

```
CREATE VIEW PayType AS
SELECT TotalPrice,
       PaymentType
FROM   PaymentInfo
ORDER BY TotalPrice DESC;
```

```
SELECT *
FROM PayType;
```

	totalprice text	paymenttype text
1	$9000	Bitcoin
2	$785	Visa
3	$3600	Paypal
4	$2100	Paypal
5	$1600	Visa
6	$12000	Bitcoin
7	$10240	Bitcoin

Reports

- This report runs a query that displays information on a customer name and payment type.

SELECT FirstName, LastName, PaymentType
FROM Customers, PaymentInfo
WHERE PaymentType = 'Bitcoin'
AND FirstName = 'Delroy'
AND LastName = ‘Mathieson’
LIMIT 1;

	firstname text	lastname text	paymenttype text
1	Delroy	Mathieson	Bitcoin

- This report runs a query that displays the shipping information for each order.

SELECT ShipDate,
ShipCompName,
TrackingNum
FROM Shipping
ORDER BY (ShipCompName) **ASC**;

	shipdate text	shipcompname text	trackingnum text
1	07-06-15	FEDEX	624898264023
2	09-17-16	FEDEX	624893691092
3	04-22-14	UPS	1Z06E18A6847193932
4	01-25-14	UPS	1Z06E18A6836392739
5	03-17-13	USPS	9405809699937119815434
6	01-02-16	USPS	940580685485811981543 4
7	09-16-16	USPS	9405868658075367815434

Stored Procedures

- This stored procedure is used to locate a customer.

```
CREATE OR REPLACE FUNCTION locateCust(TEXT)
RETURNS TABLE(lastName TEXT, firstName TEXT, Address TEXT, City TEXT, State
TEXT, ZipCode TEXT) AS
$$
DECLARE
        findCust TEXT := $1;
BEGIN
        RETURN QUERY
        SELECT Customers.lastName, Customers.firstName, Customers.Address,
Customers.City, Customers.State, Customers.ZipCode
        FROM Customers
        WHERE Customers.lastName = findCust
        ORDER BY lastName ASC, firstName ASC;
END;
$$ LANGUAGE plpgsql

SELECT locateCust('Mathieson');
```

	locatecust record
1	(Mathieson,Delroy,"3399 North Rd",Poughkeepsie,NY,12601)

- This stored procedure is used to check the product quantity.

```
CREATE OR REPLACE FUNCTION checkProductQuantity(TEXT)
RETURNS TABLE(ItemID TEXT, ItemName TEXT, OriginStore TEXT, StockQuantity INT)
AS
$$
DECLARE
        productName TEXT := $1;
BEGIN
        RETURN QUERY
        SELECT Items.ItemID, Items.ItemName, Items.OriginStore, Items.StockQuantity
        FROM Items
        WHERE Items.ItemName = productName;
END
$$ LANGUAGE plpgsql

SELECT checkProductQuantity('Yeezy Boost 350');
```

	checkproductquantity record
1	(A003,"Yeezy Boost 350",Adidas,1000)
2	(A009,"Yeezy Boost 350",YeezySupply,300)
3	(C453,"Yeezy Boost 350",Adidas,2500)

Trigger

- The purpose for this trigger is to reduce the stock quantity when a customer purchases an item.

```
CREATE OR REPLACE FUNCTION negativeQuantity()
RETURNS TRIGGER AS
$$
BEGIN
        IF (NEW.StockQuantity <= 0) THEN
        DELETE FROM Items WHERE Items.ItemID = NEW.ItemID;
        DELETE FROM Items WHERE Items.StockQuantity = NEW.StockQuantity;
        END IF;
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER negativeQuantity
AFTER INSERT ON Items
FOR EACH ROW
EXECUTE PROCEDURE negativeQuantity();



SELECT * FROM Items;
INSERT INTO Items VALUES
('D070', 'Dell Maths 70s', 'Reebok', -100, '$70', 70);
```

Security

- Security is an important function for a database.

Administration Role: Represents myself as System Administrator who has full access to the database.

---SECUIRTY---

CREATE ROLE ADMIN;
GRANT ALL
ON ALL TABLES IN SCHEMA PUBLIC
TO ADMIN;

Employee Role: In the future if I were to have an employee they would need to make changes to the database.

---EMPLOYEE---

EMPLOYEE **SELECT**, **UPDATE**, **AND DELETE**
CREATE ROLE EMPLOYEE;
GRANT SELECT, **UPDATE**, **DELETE**
ON Orders, Customers
TO EMPLOYEE;

Customer Role: Customer can alter their personal information in the database.

---CUSTOMERS---

CREATE ROLE Customers;
GRANT SELECT
ON ALL TABLES **IN** SCHEMA PUBLIC
TO CUSTOMERS;
GRANT SELECT, **UPDATE**, **DELETE**
REVOKE CustomerID
ON ALL TABLES IN SCHEMA PUBLIC
FROM CUSTOMERS;

Implementation Notes

- Currently the system is implemented to handle a small database that can function for a few clients. Customer information plays an important role when creating a database for a company.

Known Problems

- Customers aren’t able to purchase more than one item the same day due to the check constraint.
- Multiply customers can’t necessarily buy items with the same identification number.

Future Enhancements

- In the future, I can add a table for future employees, date & time for order date and overall improvements on customers being able to purchase different items at the same time.