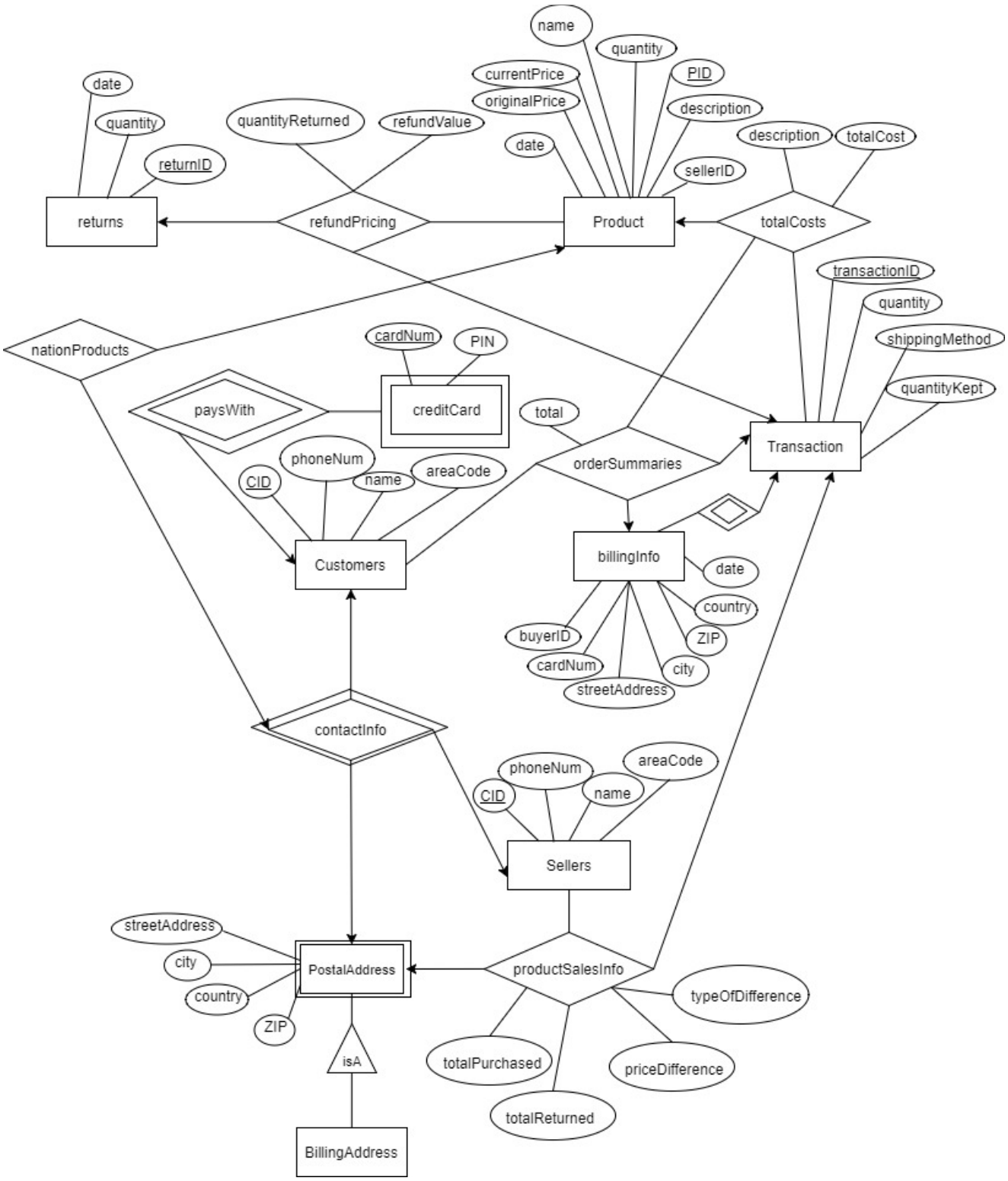
CS 410 : Introduction to Databases
Fall 2017 Term Project

Submitted by : Justin Anthony Timberlake
BlazerID : justony7@uab.edu

Due : December 15th, 2017

A : E-R Diagram


name
quantity
currentPrice
PID
originalPrice
description
date
quantity
quantityReturned
refundValue
description
totalCost
returnID
date
sellerID
returns
refundPricing
Product
totalCosts
transactionID
quantity
nationProducts
cardNum
PIN
shippingMethod
paysWith
creditCard
total
quantityKept
phoneNum
Transaction
orderSummaries
CID
name
areaCode
Customers
billingInfo
date
country
buyerID
ZIP
cardNum
city
streetAddress
contactInfo
phoneNum
areaCode
CID
name
Sellers
streetAddress
city
PostalAddress
productSalesInfo
typeOfDifference
country
ZIP
isA
totalPurchased
priceDifference
totalReturned
BillingAddress

Database Description

A Customers tuple has CID, phoneNum, name, and areaCode, which are referenced in the contactInfo relation along with their associated PostalAddress as every Customers tuple corresponds to exactly one PostalAddress tuple. The contactInfo relation also groups in all Sellers the same way, displaying the same information about them, UNION'd with the Customers information. However, a given Customer can have multiple BillingAddresses. Each BillingAddress is uniquely identified by all keys so that the foreign key constraints on billingInfo can be maintained. For every Transaction, there is an associated billingInfo tuple which gives more information about the given Transaction, including buyerID, cardNum, streetAddress, city, ZIP, country, and date. There are foreign key constraints on this table which only allow input of BillingAddresses and cards which are associated with proper buyerID's (CID's) within the creditCard and BillingAddress tables. This means that for every billingInfo tuple, one can only enter in information which is already in those tables and aligns with an intended buyerID (CID). The associated BillingAddress, creditCard, name, phoneNum, areaCode, date, and total cost of a given Transaction are displayed in the orderSummaries relation, which first relies on the totalCosts relation to calculate the total cost for each item including shipping. Each transaction is marked by the unique identifying set of transactionID and PID. This is because for every transaction, there can be multiple types of items purchased, so while there may be multiple of one transactionID, the unique entries are defined by the transactionID and the PID of the item in that portion of the given transaction. The totalCosts relation takes the transactionID, quantity, and PID from Transaction and JOINs it with the associated currentPrice and name from the Product table with its PID, then calculating the subtotal by multiplying the quantity by the item's price as well as the shippingCost, which will be 10% of the subtotal if the shippingMethod in the Transaction table for that given item is 'Normal' and 20% if it is 'Expedited'. These values are summed to form the total cost per item per transactionID. The aforementioned orderSummaries' total depends on the summation of all totalCost values for a given transactionID, so if there were multiple types of items bought within one Transaction, orderSummaries would give the total order summary per transactionID, while totalCosts generates the cost per item within a Transaction.

Using this system of Transaction parts, it is also able to handle returns, partial or whole. The returns table stores an associated transactionID to return from as well as the quantity of a given item and the PID portion of that Transaction. Each return is uniquely identified with a "returnID". There are constraints within the associated update triggers that disallow returning more items than were previously bought in the Transaction table (detailed in the Triggers section) and foreign key constraints that disallow adding anything that is not already present in the Transaction table itself. The quantity in this table will then be detracted from the quantityKept value of each associated Transaction of any number of a Product bought. Initially, the values in Transaction for quantity and quantityKept are assumed to be identical, however, every time an element is added to the returns table, the quantity in returns for a given transaction and PID will be subtracted from the quantityKept of a Transaction. Every time an item is bought or returned, the total number of available items quantity is updated in the Product table as per a trigger for updating the Product quantity when elements are inserted or updated in the Transaction table and the second trigger, which disallows anything other than inserting into the returns table, updating the quantityKept value of the Transaction table, keeping the original quantity so that information about each order can remain the same in orderSummaries and totalCosts relations. The amount of money returned to the Customer is calculated in the refundPricing relation between the returns and Products tables, which takes the quantity and item being returned and calculates the subtotal as before, but instead takes 75% of this total price calculated and returns that to the user instead of the full price, since return shipping is free. The remaining views/queries are further explained in depth in deliverable D, as well as the non-key, non-foreign-key constraints in deliverable E, Indexes in deliverable F, and finally the detailed description of triggers in deliverable G.

B : Relational Schema

Tables

Sellers(CID, phoneNum, name, areaCode)

Customer(CID, phoneNum, name, areaCode)

PostalAddress(CID, streetAddress, city, ZIP, country)

BillingAddress(CID, streetAddress, city, ZIP, country)

creditCard(cardNum, CID, PIN)

Product(PID, originalPrice, currentPrice, name, quantity, description, sellerID)

Transaction(transactionID, quantity, shippingMethod, PID, quantityKept int)

billingInfo(transactionID, buyerID, cardnum, streetaddress, city, zip, country, date)

returns(transactionID, date, quantity, PID, returnID)

```
CREATE TABLE Sellers(CID int PRIMARY KEY, phoneNum int, name VARCHAR(30), areaCode
    int);
CREATE TABLE Customers(CID int PRIMARY KEY, phoneNum int, name VARCHAR(30),
    areaCode int);
CREATE TABLE PostalAddress(CID int PRIMARY KEY, streetAddress VARCHAR(40), city
    VARCHAR(30), ZIP VARCHAR(8), country VARCHAR(30));
CREATE TABLE BillingAddress(CID int, streetAddress VARCHAR(40), city VARCHAR(30), ZIP
    VARCHAR(8),country VARCHAR(30), PRIMARY KEY (CID, streetAddress, city, ZIP,
    country));
CREATE TABLE creditCard(cardNum VARCHAR(16), CID int, PIN int, PRIMARY KEY (CID,
    cardNum));
CREATE TABLE Product(PID int PRIMARY KEY, originalPrice NUMERIC, currentPrice
    NUMERIC, name VARCHAR(30), quantity int, description VARCHAR(80), sellerID int);
CREATE TABLE Transaction(transactionID int , quantity int, shippingMethod VARCHAR(20), PID
    int, quantityKept int, PRIMARY KEY (transactionID, PID));
CREATE TABLE billingInfo(transactionID int PRIMARY KEY, buyerID int,  cardnum
VARCHAR(16), streetAddress VARCHAR(40), city VARCHAR(30), ZIP VARCHAR(8), country
    VARCHAR(30), date date);
CREATE TABLE returns(transactionID int, date date, quantity int, PID int, returnID int PRIMARY
    KEY);
```

C : Sample Data

Sellers

```
justony7=> SELECT * FROM Sellers;
 cid | phonenum |  name   | areacode
-----+----------+---------+----------
 100 |  1111111 | Jenny   |      334
 101 |  2222222 | Craig   |      334
 102 |  3333333 | Joe     |      205
 103 |  4444444 | Jeffrey |      205
 104 |  5555555 | Ethan   |      205
 105 |  6666666 | Cassidy |      817
 106 |  7777777 | Julie   |      817
 107 |  8888888 | Chris   |      617
 108 |  9999999 | Garrett |      617
 109 |  1234567 | Jose    |      956
(10 rows)
```

Customers

```
justony7=> SELECT * FROM Customers ORDER BY CID;
 cid | phonenum |  name   | areacode
-----+----------+--------+----------
 200 |  1111110 | Ulga   |      719
 201 |  2222220 | Rudolf |      817
 202 |  3333330 | Kyle   |      251
 203 |  4444440 | Greg   |      334
 204 |  5555550 | Rosa   |      334
 205 |  6666660 | Kasey  |      934
 206 |  7777770 | Maria  |      197
 207 |  8888880 | Annie  |      369
 208 |  9999990 | Rolf   |      334
 209 |  1234067 | Ulrick |      447
(10 rows)
```

Product

```
justony7=> SELECT * FROM Product ORDER BY pid;
 pid | originalprice | currentprice |            name             | quantity |                      description                       | sellerid
-----+---------------+--------------+-----------------------------+----------+--------------------------------------------------------+----------
 900 |             9 |         10.2 | Void Spray                  |       20 | Just stop existing.                                    |      100
 901 |          8.02 |         8.03 | Diet Water                  |       40 | Water, but somehow healthier?                          |      101
 902 |            16 |        15.05 | DVD Rewinder                |       30 | Rewind any DVD!                                        |      102
 903 |            11 |         9.95 | Two Person Sweatshirt       |       10 | Because sharing is caring                              |      103
 904 |         10.95 |        12.02 | Shoe Umbrellas              |       25 | Pretend to protect your feet from water.               |      104
 905 |          5.99 |         5.99 | Solar Powered Cigarette     |       28 | Light a cigarette with a magnifying glass.             |      105
 906 |          9.99 |        10.99 | Head Mounted Toilet Paper Roll |    28 | For those absolute emergencies                         |      106
 907 |         15.99 |        14.99 | Steering Wheel Tray         |       33 | Eat safely while you drive!                            |      107
 908 |          8.99 |         7.99 | Plastic Snowball Maker      |       50 | For those who struggle making their own                |      108
 909 |          9.99 |         9.99 | Revolving Ice Cream Cone    |       20 | Eat ice cream as symmetrically as possible.            |      109
 910 |         39.99 |        39.99 | Air Conditioned Shoes       |       22 | Prevent sweaty feet, but avoid walking near water.     |      105
 911 |         29.99 |        28.99 | Walking Sleeping Back       |       13 | Completely cover your body, for immediate naps.        |      104
 912 |         19.99 |        19.99 | Remote Headband             |       25 | Store remote controls on your head for immediate access. |    102
 913 |         19.99 |        15.95 | Screen Privacy Hood         |       12 | Connect your face directly to your screen.             |      104
 914 |         10.23 |        21.21 | Upside Down Umbrella        |        9 | Because regular umbrellas are too mainstream           |      108
 915 |          5.95 |         4.95 | Banana Slicer               |       18 | Slice your bananas perfectly every time.               |      107
(16 rows)
```

PostalAddress

```
justony7=> SELECT * FROM PostalAddress ORDER BY cid;
 cid | streetaddress |     city      |   zip   |   country
-----+---------------+---------------+---------+-------------
 100 | 2020 I Ave    | Arlington     | 76001   | USA
 101 | 3030 J Str    | Plano         | 75023   | USA
 102 | 4040 K Cir    | New Delhi     | 110 029 | India
 103 | 5050 L Ln     | Milwaukee     | 53201   | USA
 104 | 6060 M Rd     | Bangkok       | 10200   | China
 105 | 7070 N Pl S   | Seoul         | 100-092 | South Korea
 106 | 8080 O St     | Kiel          | 24111   | Germany
 107 | 9990 P Rd     | Sydney        | 2006    | Australia
 108 | 9019 Q Cir    | Bremen        | 28195   | Germany
 109 | 9871 R Pl     | Montgomery    | 36114   | USA
 200 | 1111 A Ave    | Arlington     | 76001   | USA
 201 | 2222 B Str    | Fort Worth    | 76248   | USA
 202 | 3333 C Cir    | Hazel Green   | 35874   | USA
 203 | 4444 D Ln     | Stuttgart     | 70173   | Germany
 204 | 5555 E Rd     | Tokyo         | 100-001 | Japan
 205 | 6666 F Pl S   | Birmingham    | B20     | England
 206 | 7777 G St     | Sydney        | 2027    | Australia
 207 | 8888 F Rd     | Kiev          | 03134   | Ukraine
 208 | 9999 G Cir    | Moscow        | 105094  | Russia
 209 | 1010 H Pl     | Montes Claros | 39400   | Brazil
(20 rows)
```

BillingAddress

```
justony7=> SELECT * FROM BillingAddress ORDER BY cid;
 cid | streetaddress |      city      |   zip    |  country
-----+---------------+----------------+----------+-----------
 200 | 1111 A Ave    | Arlington      | 76001    | USA
 201 | 2222 B Str    | Fort Worth     | 76248    | USA
 202 | 3333 C Str    | Hazel Green    | 35874    | USA
 203 | 4321 S Ln     | Vienna         | 1150     | Austria
 203 | 4444 D Ln     | Stuttgart      | 70173    | Germany
 203 | 1234 T Str    | Hamburg        | 20535    | Germany
 204 | 5555 E Rd     | Tokyo          | 100-001  | Japan
 204 | 7890 U Pl N   | Kyoto          | 600-8012 | Japan
 205 | 6666 F Pl S   | Birmingham     | B20      | England
 206 | 7777 G St     | Sydney         | 2027     | Australia
 207 | 8888 F Rd     | Kiev           | 03134    | Ukraine
 208 | 9999 G Cir    | Moscow         | 105094   | Russia
 209 | 1010 H Pl     | Montes Claros  | 39400    | Brazil
 209 | 2020 V Ln     | Belo Horizonte | 30000    | Brazil
 209 | 3030 W Rd     | Belo Horizonte | 30000    | Brazil
(15 rows)
```

creditCard

```
justony7=> SELECT * FROM creditCard ORDER BY cid;
     cardnum      | cid | pin
------------------+-----+------
 3827658362865277 | 200 | 4113
 6253975733960101 | 201 | 4943
 1790494863782286 | 202 | 7389
 3365564977234213 | 202 | 2869
 3380308032635450 | 203 | 6579
 8057834922353851 | 203 | 7325
 9808318043267544 | 204 | 5695
 2177683257213386 | 205 | 9633
 2929243420375594 | 205 | 1045
 0036975955121126 | 206 | 3219
 7225527318310115 | 207 | 9047
 5922837950301104 | 208 | 7021
 6875456693404434 | 209 | 3894
(13 rows)
```

Transaction

```
justony7=> SELECT * from Transaction ORDER BY TRANSACTIONID, PID;
 transactionid | quantity | shippingmethod | pid | quantitykept
---------------+----------+----------------+-----+--------------
           500 |        5 | Expedited      | 900 |            5
           500 |        2 | Normal         | 901 |            2
           501 |        3 | Expedited      | 908 |            3
           501 |        4 | Expedited      | 914 |            4
           502 |       10 | Normal         | 909 |           10
           503 |       15 | Normal         | 912 |           15
           504 |        3 | Normal         | 903 |            3
           504 |        7 | Normal         | 913 |            7
           504 |        9 | Expedited      | 915 |            9
           505 |        6 | Expedited      | 904 |            6
           506 |       15 | Normal         | 902 |           15
           507 |        8 | Normal         | 915 |            8
           508 |       20 | Expedited      | 908 |           20
           509 |       13 | Expedited      | 906 |           13
           509 |       12 | Normal         | 907 |           12
(15 rows)
```

BillingInfo

```
justony7=> SELECT * FROM BILLINGINFO ORDER BY transactionID;
 transactionid | buyerid |     cardnum      | streetaddress |      city      |   zip   |  country  |    date
---------------+---------+------------------+---------------+----------------+---------+-----------+------------
           500 |     207 | 7225527318310115 | 8888 F Rd     | Kiev           | 03134   | Ukraine   | 2017-12-09
           501 |     203 | 3380308032635450 | 4321 S Ln     | Vienna         | 1150    | Austria   | 2017-12-10
           502 |     201 | 6253975733960101 | 2222 B Str    | Fort Worth     | 76248   | USA       | 2017-12-11
           503 |     202 | 1790494863782286 | 3333 C Str    | Hazel Green    | 35874   | USA       | 2017-12-13
           504 |     204 | 9808318043267544 | 5555 E Rd     | Tokyo          | 100-001 | Japan     | 2017-12-14
           505 |     205 | 2929243420375594 | 6666 F Pl S   | Birmingham     | B20     | England   | 2017-12-14
           506 |     209 | 6875456693404434 | 1010 H Pl     | Montes Claros  | 39400   | Brazil    | 2017-12-16
           507 |     209 | 6875456693404434 | 3030 W Rd     | Belo Horizonte | 30000   | Brazil    | 2017-12-21
           508 |     208 | 5922837950301104 | 9999 G Cir    | Moscow         | 105094  | Russia    | 2017-12-22
           509 |     206 | 0036975955121126 | 7777 G St     | Sydney         | 2027    | Australia | 2012-12-01
(10 rows)
```

Returns

```
justony7=> SELECT * FROM Returns ORDER BY returnID;
 transactionid |    date    | quantity | pid | returnid
---------------+------------+----------+-----+----------
           500 | 2017-12-10 |        2 | 901 |      700
           501 | 2017-12-17 |        3 | 908 |      701
           501 | 2017-12-18 |        1 | 914 |      702
           503 | 2017-12-20 |       14 | 912 |      703
           504 | 2017-12-17 |        7 | 915 |      704
           504 | 2017-12-19 |        3 | 903 |      705
           505 | 2017-12-21 |        4 | 904 |      706
           506 | 2017-12-18 |       15 | 902 |      707
           508 | 2017-12-31 |       18 | 908 |      708
           509 | 2017-12-31 |       12 | 906 |      709
(10 rows)
```

D : Views

1. totalCosts – This view generates the shipping and total costs per transaction listed in the transaction table.

```
CREATE OR REPLACE VIEW totalCosts AS
SELECT t.transactionID, t.quantity, ROUND(p.currentprice, 2) AS itemCost,
ROUND(t.quantity*p.currentprice, 2) AS subtotal, CASE WHEN t.shippingMethod = 'Expedited'
THEN  ROUND(.2*t.quantity*p.currentprice, 2) WHEN t.shippingMethod = 'Normal' THEN
ROUND(.1*t.quantity*p.currentprice, 2) END as shippingCost, CASE WHEN t.shippingMethod =
'Expedited' THEN ROUND(1.2*t.quantity*p.currentprice, 2) WHEN t.shippingMethod = 'Normal'
THEN  ROUND(1.1*t.quantity*p.currentprice, 2) END as totalCost, p.pid, p.name AS item
FROM product p JOIN transaction t ON p.pid = t.pid
ORDER BY transactionID, PID;
```

2. orderSummaries – This view takes the total costs for each item within each transactionID and sums them together, displaying relevant credit card, billing address, and customer information

```
CREATE OR REPLACE VIEW orderSummaries AS
SELECT t.transactionid, c.name, t.buyerid, c.areaCode, c.phonenum, t.date,
ROUND(SUM(tc.totalcost),2) AS total, t.cardNum AS creditCard, t.streetAddress, t.city, t.zip,
t.country
FROM (SELECT DISTINCT t.TransactionID, b.buyerid, b.streetAddress, b.city, b.zip, b.country,
b.cardnum, b.date FROM Transaction t JOIN billingInfo b ON t.transactionID = b.transactionID) t
JOIN totalCosts tc ON tc.transactionID = t.transactionID
JOIN Customers c ON t.buyerid = c.cid
GROUP BY c.name, c.areaCode, c.phonenum, t.buyerid, t.transactionid, t.date, t.buyerid, t.cardNum,
t.streetAddress, t.city, t.zip, t.country
ORDER BY transactionID;
```

3. contactInfo – This view displays all contact information of all users in the database, both buyers and sellers as well as their primary postal addresses (not billing addresses)

```
CREATE VIEW ContactInfo AS
SELECT s.cid, s.name, s.areaCode, s.phonenum, p.streetaddress, p.city, p.zip, p.country
FROM Sellers s JOIN PostalAddress p ON s.cid = p.cid
WHERE p.cid < 200
UNION ALL
SELECT c.cid, c.name, c.areaCode, c.phonenum, p.streetaddress, p.city, p.zip, p.country
FROM Customers c JOIN PostalAddress p ON c.cid = p.cid
WHERE p.cid >= 200
ORDER BY cid;
```

4. refundPricing – This view generates the amount customers will receive back when returning items (75% of market value per item returned, free shipping)

```
CREATE VIEW refundPricing AS
SELECT r.returnid, r.pid AS returnedItemID, r.quantity AS quantityReturned, p.currentPrice AS
marketPrice, ROUND(.75*r.quantity*p.currentprice, 2) AS refundValue
FROM Returns r JOIN Product p ON r.pid = p.pid
ORDER BY returnID;
```

5. productSalesInfo – This view shows total price increase or decrease as well as number purchased versus returned of each item in the Product table (sorted by PID) along with the name and ID of who sells each product along with the country of origin for that product (based on the PostalAddress entry for the seller).

```
CREATE OR REPLACE VIEW productSalesInfo AS
WITH CTE_Info AS
(SELECT p.pid, p.name, SUM(t.quantity) AS totalpurchased, SUM(t.quantity - t.quantityKept) AS
totalReturned, ABS(ROUND(p.currentPrice - p.originalPrice,2)) AS priceDifference, CASE WHEN
(p.currentPrice - p.originalPrice) < 0 THEN 'Discounted' WHEN (p.currentPrice - p.originalPrice) > 0
THEN 'Increased' WHEN (p.currentPrice - p.originalPrice) = 0 THEN 'Static' ELSE 'N/A' END AS
typeOfDifference, s.cid AS sellerID, s.name AS sellerName, pa.country AS productOrigin
FROM Transaction t RIGHT OUTER JOIN Product
 p ON p.pid = t.pid JOIN Sellers s ON p.sellerid = s.cid JOIN PostalAddress pa ON p.sellerid = pa.cid
WHERE t.quantity IS NOT NULL
GROUP BY p.pid, p.name, s.cid, s.name, pa.country
UNION ALL
SELECT p.pid, p.name, 0 AS totalpurchased, 0 AS totalReturned, ROUND(0, 2) AS priceDifference,
'N/A' AS typeOfDifference, s.cid AS sellerID, s.name AS sellerName, pa.country AS productOrigin
FROM Transaction t RIGHT OUTER JOIN Product
 p ON p.pid = t.pid JOIN Sellers s ON p.sellerid = s.cid JOIN PostalAddress pa ON p.sellerid = pa.cid
WHERE t.quantity IS NULL)
SELECT * FROM CTE_Info ORDER BY PID;
```

6. nationProducts – Displays sellers and the products they sell by each country

```
CREATE OR REPLACE VIEW NationProducts AS
SELECT c.country, c.name AS Seller, c.cid AS SellerID, p.name AS item, p.pid AS itemID,
p.currentPrice AS itemCost
FROM contactInfo c JOIN Product p ON c.cid = p.sellerID
ORDER BY country;
```

7. paysWith – Displays all creditCards for a given Customer

```
CREATE OR REPLACE VIEW paysWith AS
SELECT c.cid, c.name, cc.cardnum, cc.pin
FROM Customers c JOIN creditCard cc ON c.cid = cc.cid
ORDER BY c.cid;
```

E : Indexes

1. CREATE INDEX ON Product (pid);
> PID is used whenever JOINing any table with Product so it is referenced many time in my views.

2. CREATE INDEX ON Transaction (transactionID, PID);
> Every time a specific item has been purchased, both attributes need to be referenced as they are in my views as well as my triggers.

3. CREATE INDEX ON PostalAddress (CID);
> CID is used whenever JOINing any table with PostalAddress so it is referenced many time in my views.

F : Constraints

1. Check for legal values for areaCode and phonenum in the Sellers table :

```
ALTER TABLE Sellers ADD CHECK (areaCode < 1000 AND areaCode > 100 AND phoneNum <
10000000 AND phoneNum > 1000000);
```

2. Check for legal values for areaCode and phonenum in the Customers table :

```
ALTER TABLE Customers ADD CHECK (areaCode < 1000 AND areaCode > 100 AND phoneNum <
10000000 AND phoneNum > 1000000);
```

3. Make sure that the shipping method in the Transaction table is either ‘Expedited’ or ‘Normal’ :

```
ALTER TABLE Transaction ADD CHECK (shippingMethod = 'Expedited' OR shippingMethod =
'Normal');
```

G : Triggers

1. updateProduct on Transaction : will reflect the total quantity of items available in the Product table, based on modifications to the Transaction table

```
CREATE OR REPLACE FUNCTION updateproductfunc()
RETURNS TRIGGER AS $updateproductfunc$
BEGIN
	IF (TG_OP = 'INSERT') THEN
		IF (SELECT (CASE WHEN NEW.quantity > p.quantity THEN 1 ELSE 0 END) AS
			greaterThan
		FROM Product p
		WHERE p.pid = NEW.pid) THEN
			RAISE NOTICE 'Not enough in stock!';
		ELSE
			UPDATE Product
			SET quantity = quantity - NEW.quantity
			WHERE pid = NEW.pid;
			RETURN NEW;
		END IF;
		ELSEIF (TG_OP = 'UPDATE') THEN
			UPDATE Product
			SET quantity = quantity + OLD.quantityKept - NEW.quantityKept
			WHERE pid = NEW.pid;
			RETURN NEW;
		ELSEIF (TG_OP = 'DELETE') THEN
			UPDATE Product
			SET quantity = quantity + OLD.quantityKept
			WHERE pid = OLD.pid;
			RETURN OLD;
	END IF;
	RETURN NULL;
END;
$updateproductfunc$ LANGUAGE plpgsql;

CREATE TRIGGER UpdateProduct
BEFORE INSERT OR DELETE OR UPDATE ON Transaction
FOR EACH ROW
EXECUTE PROCEDURE updateproductfunc();
```

After populating Transaction table, the Product table looks like this :

```
justony7=> SELECT * FROM PRODUCT ORDER BY PID;
 pid | originalprice | currentprice |              name              | quantity |                        description                         | sellerid
-----+---------------+--------------+--------------------------------+----------+------------------------------------------------------------+----------
 900 |             9 |         10.2 | Void Spray                     |       15 | Just stop existing.                                        |      100
 901 |          8.02 |         8.03 | Diet Water                     |       38 | Water, but somehow healthier?                              |      101
 902 |            16 |        15.05 | DVD Rewinder                   |       15 | Rewind any DVD!                                            |      102
 903 |            11 |         9.95 | Two Person Sweatshirt          |        7 | Because sharing is caring                                  |      103
 904 |         10.95 |        12.02 | Shoe Umbrellas                 |       19 | Pretend to protect your feet from water.                   |      104
 905 |          5.99 |         5.99 | Solar Powered Cigarette        |       28 | Light a cigarette with a magnifying glass.                 |      105
 906 |          9.99 |        10.99 | Head Mounted Toilet Paper Roll |       15 | For those absolute emergencies                             |      106
 907 |         15.99 |        14.99 | Steering Wheel Tray            |       21 | Eat safely while you drive!                                |      107
 908 |          8.99 |         7.99 | Plastic Snowball Maker         |       27 | For those who struggle making their own                    |      108
 909 |          9.99 |         9.99 | Revolving Ice Cream Cone       |       10 | Eat ice cream as symmetrically as possible.                |      109
 910 |         39.99 |        39.99 | Air Conditioned Shoes          |       22 | Prevent sweaty feet, but avoid walking near water.         |      105
 911 |         29.99 |        28.99 | Walking Sleeping Back          |       13 | Completely cover your body, for immediate naps.            |      104
 912 |         19.99 |        19.99 | Remote Headband                |       10 | Store remote controls on your head for immediate access.   |      102
 913 |         19.99 |        15.95 | Screen Privacy Hood            |        5 | Connect your face directly to your screen.                 |      104
 914 |         10.23 |        21.21 | Upside Down Umbrella           |        5 | Because regular umbrellas are too mainstream               |      108
 915 |          5.95 |         4.95 | Banana Slicer                  |        1 | Slice your bananas perfectly every time.                   |      107
(16 rows)
```

2. updateTransaction on Returnds – will reflect the total number of items able to be returned from the transaction table by updating each transaction as an element is added to the Returns table. Also accounts for invalid dates (before a given transaction) and when users try to return more than the number of items which was bought. There are already foreign key constraints on this table so that transactions only within the Transaction table can be returned partially or fully. With the use of the prior trigger, these updates to Transaction will also be reflected in the total number of items added back to the product table. This trigger also prevents updates and deletes on the Returns table, as all returns are final.

```
CREATE OR REPLACE FUNCTION updatetransactionfunc()
RETURNS TRIGGER AS $updatetransactionfunc$
BEGIN
	IF (TG_OP = 'INSERT') THEN
		IF (SELECT (CASE WHEN NEW.date < t.date THEN 1 ELSE 0 END) AS before
		FROM (SELECT t.pid, t.transactionid, b.date
			FROM Transaction t JOIN billingInfo b ON t.transactionid = b.transactionid ) t
		WHERE t.pid = NEW.pid AND t.transactionID = NEW.transactionID) THEN
			RAISE NOTICE 'A return cannot occur before a purchase!';
			RETURN NULL;
			END IF;
		IF (SELECT (CASE WHEN NEW.quantity > t.quantityKept THEN 1 ELSE 0 END) AS
				greaterThan
		FROM Transaction t
		WHERE t.pid = NEW.pid AND t.transactionID = NEW.transactionID) THEN
			RAISE NOTICE 'Cannot return more than owned!';
			ELSE
				UPDATE Transaction
				SET quantityKept = quantityKept - NEW.quantity
				WHERE pid = NEW.pid AND transactionID = NEW.transactionID;
				RETURN NEW;
		END IF;
		ELSEIF (TG_OP = 'UPDATE' OR TG_OP = 'DELETE') THEN
			RAISE NOTICE 'All returns final! No modifications allowed!';
	END IF;
	RETURN NULL;
END;
$updatetransactionfunc$ LANGUAGE plpgsql;

CREATE TRIGGER UpdateTransaction
BEFORE INSERT OR DELETE OR UPDATE ON Returns
FOR EACH ROW
EXECUTE PROCEDURE updatetransactionfunc();
```

After populating the Returns table, the Transaction table looks like this :

```
justony7=> SELECT * FROM TRANSACTION ORDER BY TRANSACTIONID, PID;
 transactionid | quantity | shippingmethod | pid | quantitykept
---------------+----------+----------------+-----+--------------
           500 |        5 | Expedited      | 900 |            5
           500 |        2 | Normal         | 901 |            0
           501 |        3 | Expedited      | 908 |            0
           501 |        4 | Expedited      | 914 |            3
           502 |       10 | Normal         | 909 |           10
           503 |       15 | Normal         | 912 |            1
           504 |        3 | Normal         | 903 |            0
           504 |        7 | Normal         | 913 |            7
           504 |        9 | Expedited      | 915 |            2
           505 |        6 | Expedited      | 904 |            2
           506 |       15 | Normal         | 902 |            0
           507 |        8 | Normal         | 915 |            8
           508 |       20 | Expedited      | 908 |            2
           509 |       13 | Expedited      | 906 |            1
           509 |       12 | Normal         | 907 |           12
(15 rows)
```

As the Transaction table is updated accordingly, the Product table will then reflect the changes as well as such :

```
justony7=> SELECT * FROM Product ORDER BY pid;
 pid | originalprice | currentprice |             name              | quantity |                        description                         | sellerid
-----+---------------+--------------+-------------------------------+----------+------------------------------------------------------------+----------
 900 |             9 |         10.2 | Void Spray                    |       15 | Just stop existing.                                        |      100
 901 |          8.02 |         8.03 | Diet Water                    |       40 | Water, but somehow healthier?                              |      101
 902 |            16 |        15.05 | DVD Rewinder                  |       30 | Rewind any DVD!                                            |      102
 903 |            11 |         9.95 | Two Person Sweatshirt         |       10 | Because sharing is caring                                  |      103
 904 |         10.95 |        12.02 | Shoe Umbrellas                |       23 | Pretend to protect your feet from water.                   |      104
 905 |          5.99 |         5.99 | Solar Powered Cigarette       |       28 | Light a cigarette with a magnifying glass.                 |      105
 906 |          9.99 |        10.99 | Head Mounted Toilet Paper Roll |       27 | For those absolute emergencies                             |      106
 907 |         15.99 |        14.99 | Steering Wheel Tray           |       21 | Eat safely while you drive!                                |      107
 908 |          8.99 |         7.99 | Plastic Snowball Maker        |       48 | For those who struggle making their own                    |      108
 909 |          9.99 |         9.99 | Revolving Ice Cream Cone      |       10 | Eat ice cream as symmetrically as possible.                |      109
 910 |         39.99 |        39.99 | Air Conditioned Shoes         |       22 | Prevent sweaty feet, but avoid walking near water.         |      105
 911 |         29.99 |        28.99 | Walking Sleeping Back         |       13 | Completely cover your body, for immediate naps.            |      104
 912 |         19.99 |        19.99 | Remote Headband               |       24 | Store remote controls on your head for immediate access.   |      102
 913 |         19.99 |        15.95 | Screen Privacy Hood           |        5 | Connect your face directly to your screen.                 |      104
 914 |         10.23 |        21.21 | Upside Down Umbrella          |        6 | Because regular umbrellas are too mainstream               |      108
 915 |          5.95 |         4.95 | Banana Slicer                 |        8 | Slice your bananas perfectly every time.                   |      107
(16 rows)
```