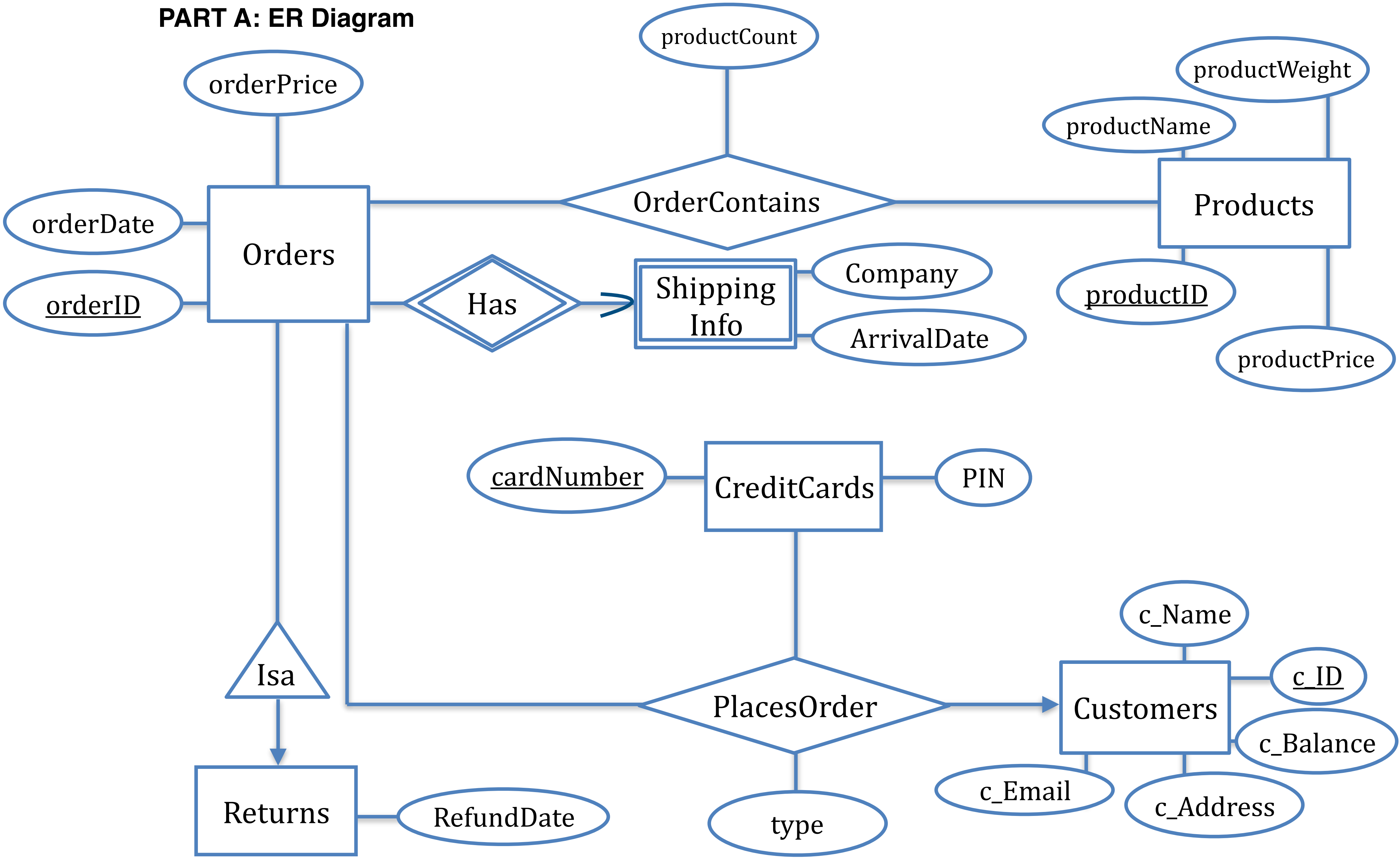
PART A: ER Diagram



UAB CS 410 Semester Project:
Online Selling System based on Databases
By Jonathan Luetze

Orders

//each order appears once, orderID is the primary key. Returns are also orders and will therefore also appear in this table
// orderPrice is the price for the entire order. It is updated via a trigger every time a product is added to the order
//CONSTRAINT: orderDate NOT NULL

Returns

orderID is the foreign key to returns
Trigger will insert into returns when placesOrder's attribute type is set to 'R' or 'FR' and set the RefundDate

Shipping Info

//every order can only have one shipping information sheet because all parts of an order are shipped together
//trigger2 after insert on placesOrder entity-> if more than 10 items, ship with UPS, arrival date in 2 days, else with USPS, arrival date 4 days; however, the arrival date can change when an order is being returned in Full - but the Company will not change
//orderID is a foreign key

Products

//each product appears once in this table; ProductID is primary key
//CONSTRAINTS: productPrice, productWeight, productName - NOT NULL

OrderContains

//relation; shows which products (with the count) are assigned to which orderID;
//CONSTRAINT: Cannot add more than 1000 units of the same items

Customers

//each customer appears once; primary key is c_ID because only one c_ID can be assigned to a customer
//CONSTRAINT: c_Email, c_Address, c_Balance - NOT NULL

CreditCards

// a customer can have multiple credit cards, but only one customer can be assigned to a credit card;
//CONSTRAINT: credit card number has to be 16 digits long
//(cardNumber > 999999999999999 AND cardNumber < 10000000000000000)
//CONSTRAINT: PIN NOT NULL

PlacesOrder

//connects customers to the orders placed and the credit card which is used and sets the type for the order
//orderID, cardNumber and c_ID are all foreign keys - must be in their respective relations first
//type indicates normal Order or Return or Full Return (Constraint: 'O' OR 'R' OR 'FR')
//CONSTRAINTS: orderID, cardNumber, c_ID, type - NOT NULL

Additional Assumptions:

1. No orders/products/customers/creditCards will be deleted (in short: nothing will be deleted)
2. We have infinite stock
3. The orderID is only used once to place an order, and 0 or 1 time to place a full return.
4. To return parts of an order (not full return, but option ‘R’), a new order number has to be created with the items that the customer wants to return
5. Returns ship back with the same shipping company that they were ordered with
6. User cannot move items out of an order after they have been added - they have to be ordered and then optionally returned
7. Credit card balance does not have a limit
8. Every Customer pays with his own credit card (since it’s an online service, the service cannot know who sits in front of the computer), and the credit card number is sufficient after the credit card number has been added with the PIN
9. Full returns are only placed after the exact order has been placed before
10. All prices are whole dollar values and do not exceed $10000 to $-10000
11. The customer’s balance cannot exceed the boundaries of $10000 to $-10000
12. The Process below is followed to create an order:

Process:

1. Add **customers** -(customer has to exist before it can be associated with a creditCard)
2. Add **Credit cards** - Every customer needs a credit card to place orders
3. Add **products** -(product has to be in table before OrderContains can add it)
4. **OrderContains** -> adds products to order -
    1. product must be in products table
    2. trigger to add orderID into the Orders table with current date (ONLY THE FIRST TIME)
    3. OrderPrice is SUM calculated by trigger before insert of OrderContains (because order has to be in orders table first, because it’s a primary key of the Orders table)
5. **PlacesOrder**
    1. after insert on PlacesOrder, calculate shipping
    2. For RETURNS, trigger2 enters orderID into Returns table with refundDate

PART B

Orders (orderID, orderDate, orderPrice)

```
CREATE TABLE Orders(
    orderID integer PRIMARY KEY,
    orderDate Date NOT NULL,
    orderPrice integer);
```

Returns (orderID)

```
CREATE TABLE Returns(
    orderID integer PRIMARY KEY,
    RefundDate Date NOT NULL,
    FOREIGN KEY(orderID) REFERENCES Orders(orderID));
```

Shipping Info (orderID, Company, ArrivalDate)

```
CREATE TABLE ShippingInfo(
    orderID integer PRIMARY KEY,
    company varchar,
    arrivalDate Date,
    FOREIGN KEY (orderID) REFERENCES Orders(orderID));
```

Products (productID, productName, productPrice, productWeight)

```
CREATE TABLE Products(
    productID integer PRIMARY KEY,
    productName varchar NOT NULL,
    productPrice integer NOT NULL,
    productWeight integer NOT NULL);
```

Customers(c_ID, c_Name, c_Email, c_Address, c_Balance)

```
CREATE TABLE Customers(
    c_ID integer PRIMARY KEY,
    c_Name varchar,
    c_Email varchar NOT NULL,
    c_Address varchar NOT NULL,
    c_Balance integer NOT NULL);
```

CreditCards(cardNumber, PIN)

```
CREATE TABLE CreditCards(
    cardNumber bigInt PRIMARY KEY,
    PIN integer NOT NULL,
```

```
ALTER TABLE creditCards ADD CONSTRAINT check_digits CHECK (cardNumber >
999999999999999 AND cardNumber < 10000000000000000);
```

OrderContains(orderID, productID, productCount)

```
CREATE TABLE OrderContains(
    orderID integer NOT NULL,
    productID integer NOT NULL,
    productCount integer NOT NULL,
    FOREIGN KEY (orderID) REFERENCES Orders(orderID),
    FOREIGN KEY (productID) REFERENCES Products(productID));
```

```
ALTER TABLE OrderContains ADD CONSTRAINT check_productCount
CHECK (productCount <= 1000);
```

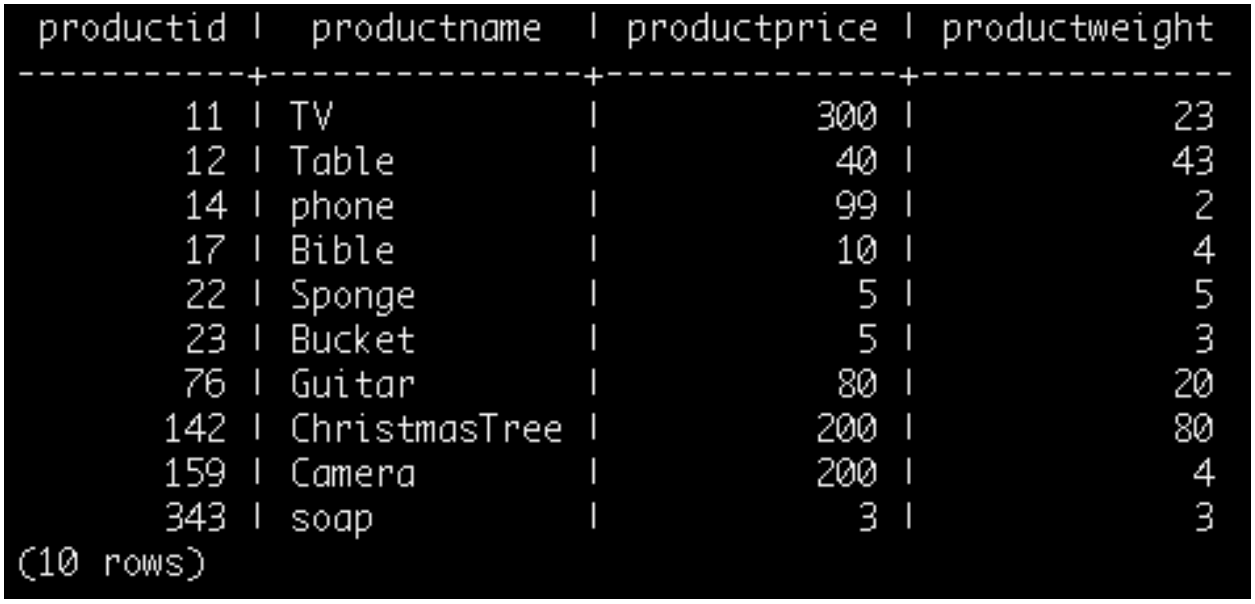
PlacesOrder(orderID, cardNumber, c_ID, type)

```
CREATE TABLE PlacesOrder(
    orderID integer NOT NULL,
    cardNumber bigInt NOT NULL,
    c_ID integer NOT NULL,
    type varchar NOT NULL,
    FOREIGN KEY (orderID) REFERENCES Orders(orderID),
    FOREIGN KEY (cardNumber) REFERENCES CreditCards(cardNumber),
    FOREIGN KEY (c_ID) REFERENCES Customers(c_ID));
```

```
ALTER TABLE PlacesOrder ADD CONSTRAINT check_type CHECK
(type = 'R' OR type = ‘O’ OR type = ‘FR’);
```

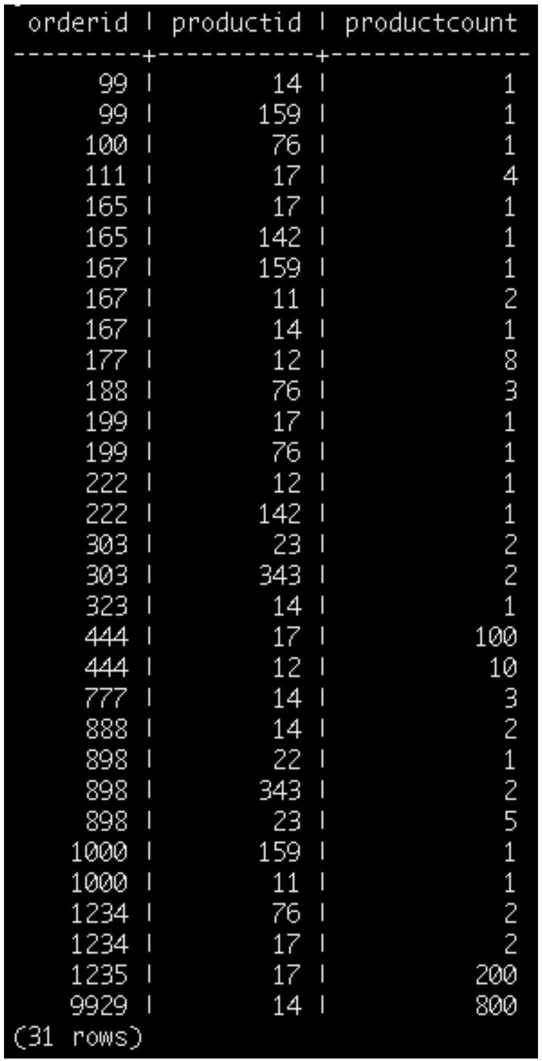
PART C - SAMPLE DATA

Relation: Products



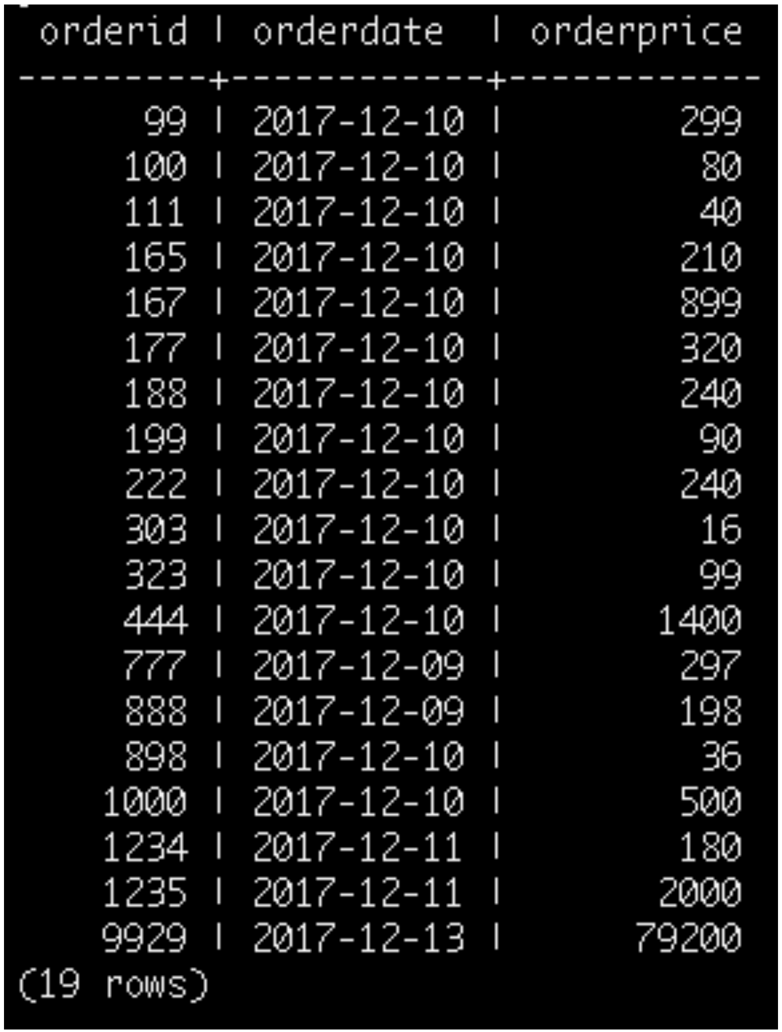
```
 productid |  productname   | productprice | productweight
-----------+----------------+--------------+---------------
        11 | TV             |          300 |            23
        12 | Table          |           40 |            43
        14 | phone          |           99 |             2
        17 | Bible          |           10 |             4
        22 | Sponge         |            5 |             5
        23 | Bucket         |            5 |             3
        76 | Guitar         |           80 |            20
       142 | ChristmasTree  |          200 |            80
       159 | Camera         |          200 |             4
       343 | soap           |            3 |             3
(10 rows)
```

Relation: OrderContains



```
 orderid | productid | productcount
---------+-----------+--------------
      99 |        14 |            1
      99 |       159 |            1
     100 |        76 |            1
     111 |        17 |            4
     165 |        17 |            1
     165 |       142 |            1
     167 |       159 |            1
     167 |        11 |            2
     167 |        14 |            1
     177 |        12 |            8
     188 |        76 |            3
     199 |        17 |            1
     199 |        76 |            1
     222 |        12 |            1
     222 |       142 |            1
     303 |        23 |            2
     303 |       343 |            2
     323 |        14 |            1
     444 |        17 |          100
     444 |        12 |           10
     777 |        14 |            3
     888 |        14 |            2
     898 |        22 |            1
     898 |       343 |            2
     898 |        23 |            5
    1000 |       159 |            1
    1000 |        11 |            1
    1234 |        76 |            2
    1234 |        17 |            2
    1235 |        17 |          200
    9929 |        14 |          800
(31 rows)
```

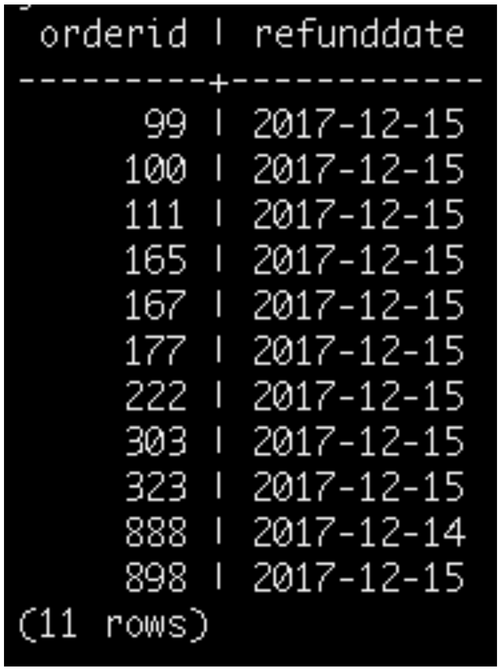
Relation: Orders



```
 orderid | orderdate  | orderprice
---------+------------+------------
      99 | 2017-12-10 |        299
     100 | 2017-12-10 |         80
     111 | 2017-12-10 |         40
     165 | 2017-12-10 |        210
     167 | 2017-12-10 |        899
     177 | 2017-12-10 |        320
     188 | 2017-12-10 |        240
     199 | 2017-12-10 |         90
     222 | 2017-12-10 |        240
     303 | 2017-12-10 |         16
     323 | 2017-12-10 |         99
     444 | 2017-12-10 |       1400
     777 | 2017-12-09 |        297
     888 | 2017-12-09 |        198
     898 | 2017-12-10 |         36
    1000 | 2017-12-10 |        500
    1234 | 2017-12-11 |        180
    1235 | 2017-12-11 |       2000
    9929 | 2017-12-13 |      79200
(19 rows)
```

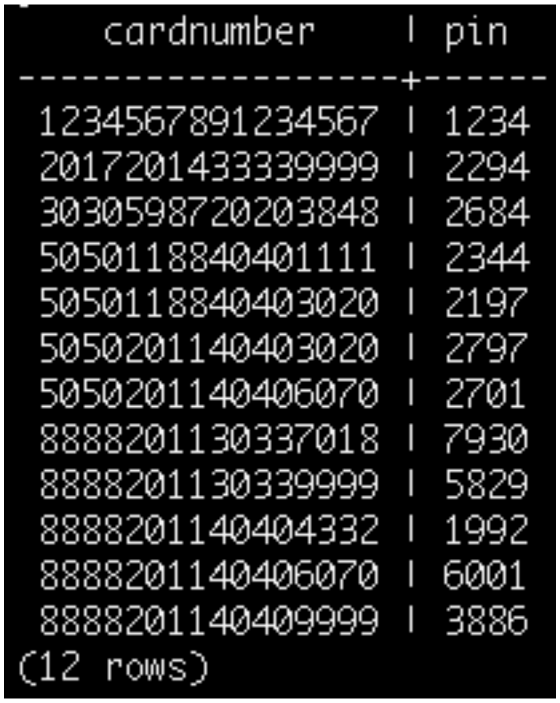
PART C - SAMPLE DATA

Relation: Returns
(Refund date only the same because entries were made the same day)



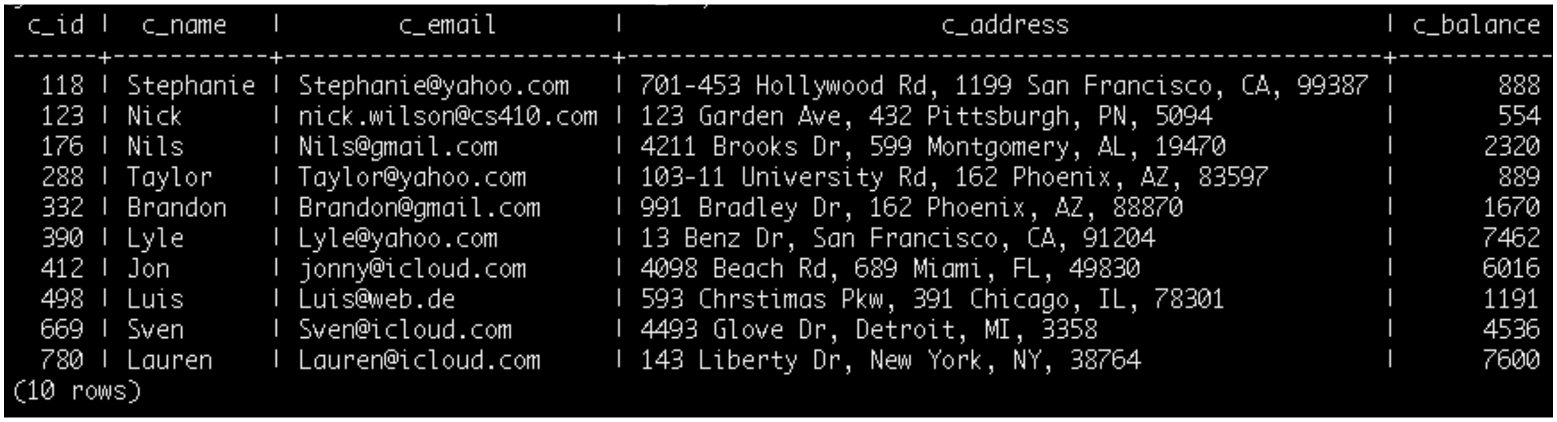
```
 orderid | refunddate
---------+------------
      99 | 2017-12-15
     100 | 2017-12-15
     111 | 2017-12-15
     165 | 2017-12-15
     167 | 2017-12-15
     177 | 2017-12-15
     222 | 2017-12-15
     303 | 2017-12-15
     323 | 2017-12-15
     888 | 2017-12-14
     898 | 2017-12-15
(11 rows)
```

Relation: CreditCards



```
    cardnumber    | pin
------------------+------
 1234567891234567 | 1234
 2017201433339999 | 2294
 3030598720203848 | 2684
 5050118840401111 | 2344
 5050118840403020 | 2197
 5050201140403020 | 2797
 5050201140406070 | 2701
 8888201130337018 | 7930
 8888201130339999 | 5829
 8888201140404332 | 1992
 8888201140406070 | 6001
 8888201140409999 | 3886
(12 rows)
```

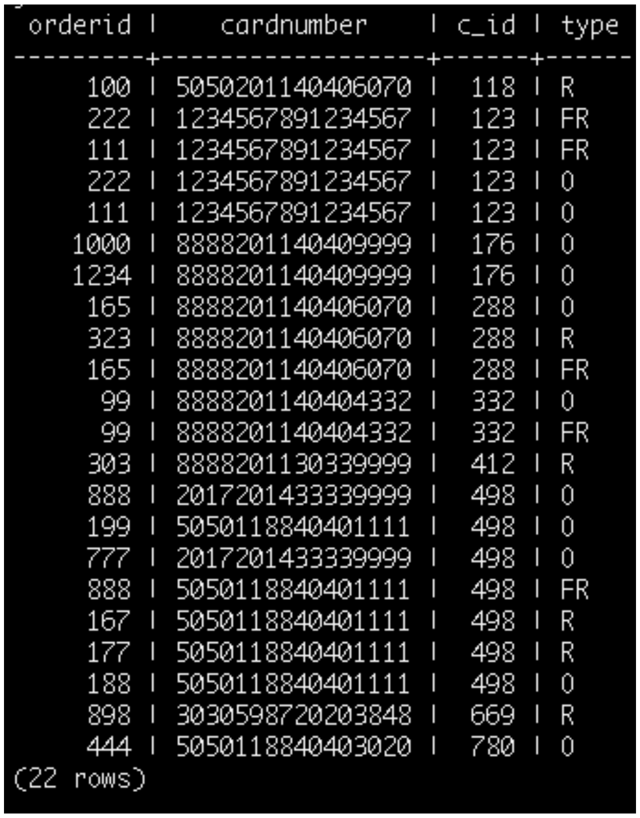
Relation: Customers



```
 c_id |  c_name   |       c_email        |                       c_address                       | c_balance
------+-----------+----------------------+-------------------------------------------------------+-----------
  118 | Stephanie | Stephanie@yahoo.com  | 701-453 Hollywood Rd, 1199 San Francisco, CA, 99387 |       888
  123 | Nick      | nick.wilson@cs410.com | 123 Garden Ave, 432 Pittsburgh, PN, 5094            |       554
  176 | Nils      | Nils@gmail.com       | 4211 Brooks Dr, 599 Montgomery, AL, 19470           |      2320
  288 | Taylor    | Taylor@yahoo.com     | 103-11 University Rd, 162 Phoenix, AZ, 83597        |       889
  332 | Brandon   | Brandon@gmail.com    | 991 Bradley Dr, 162 Phoenix, AZ, 88870              |      1670
  390 | Lyle      | Lyle@yahoo.com       | 13 Benz Dr, San Francisco, CA, 91204                |      7462
  412 | Jon       | jonny@icloud.com     | 4098 Beach Rd, 689 Miami, FL, 49830                 |      6016
  498 | Luis      | Luis@web.de          | 593 Chrstimas Pkw, 391 Chicago, IL, 78301           |      1191
  669 | Sven      | Sven@icloud.com      | 4493 Glove Dr, Detroit, MI, 3358                    |      4536
  780 | Lauren    | Lauren@icloud.com    | 143 Liberty Dr, New York, NY, 38764                 |      7600
(10 rows)
```

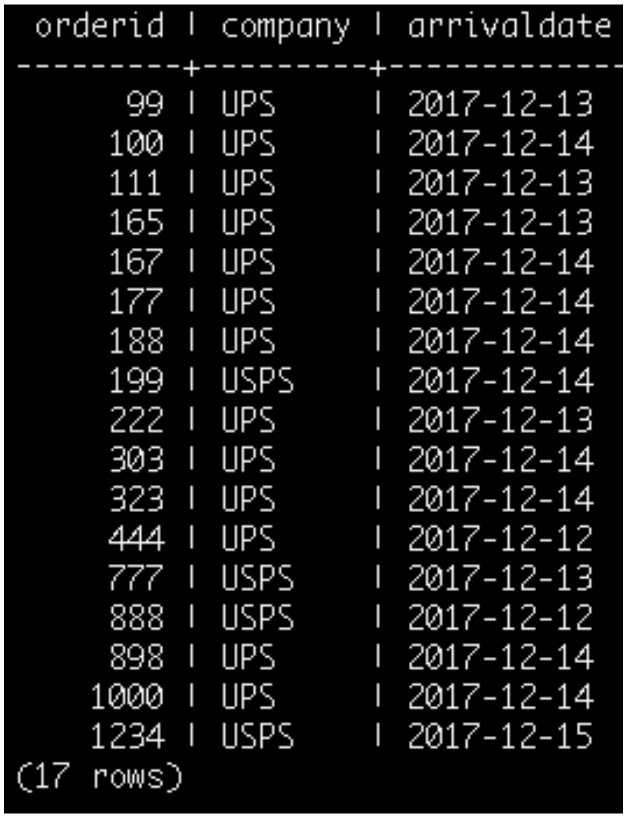
PART C - SAMPLE DATA

Relation: PlacesOrder



```
 orderid |    cardnumber     | c_id | type
---------+------------------+------+------
     100 | 5050201140406070 |  118 | R
     222 | 1234567891234567 |  123 | FR
     111 | 1234567891234567 |  123 | FR
     222 | 1234567891234567 |  123 | 0
     111 | 1234567891234567 |  123 | 0
    1000 | 8888201140409999 |  176 | 0
    1234 | 8888201140409999 |  176 | 0
     165 | 8888201140406070 |  288 | 0
     323 | 8888201140406070 |  288 | R
     165 | 8888201140406070 |  288 | FR
      99 | 8888201140404332 |  332 | 0
      99 | 8888201140404332 |  332 | FR
     303 | 8888201130339999 |  412 | R
     888 | 2017201433339999 |  498 | 0
     199 | 5050118840401111 |  498 | 0
     777 | 2017201433339999 |  498 | 0
     888 | 5050118840401111 |  498 | FR
     167 | 5050118840401111 |  498 | R
     177 | 5050118840401111 |  498 | R
     188 | 5050118840401111 |  498 | 0
     898 | 3030598720203848 |  669 | R
     444 | 5050118840403020 |  780 | 0
(22 rows)
```

Relation: ShippingInfo



```
 orderid | company | arrivaldate
---------+---------+------------
      99 | UPS     | 2017-12-13
     100 | UPS     | 2017-12-14
     111 | UPS     | 2017-12-13
     165 | UPS     | 2017-12-13
     167 | UPS     | 2017-12-14
     177 | UPS     | 2017-12-14
     188 | UPS     | 2017-12-14
     199 | USPS    | 2017-12-14
     222 | UPS     | 2017-12-13
     303 | UPS     | 2017-12-14
     323 | UPS     | 2017-12-14
     444 | UPS     | 2017-12-12
     777 | USPS    | 2017-12-13
     888 | USPS    | 2017-12-12
     898 | UPS     | 2017-12-14
    1000 | UPS     | 2017-12-14
    1234 | USPS    | 2017-12-15
(17 rows)
```

PART D - VIEWS

VIEW1: FOR ORDER SHIPMENT

```
CREATE VIEW shippingView AS
        SELECT OrderID, orderDate, Company, ArrivalDate
        FROM Orders NATURAL JOIN shippingInfo;
```

VIEW2: FOR RETURN REFUND VIEW

```
CREATE VIEW returnView AS
        SELECT OrderID, orderPrice, RefundDate
        FROM Returns NATURAL JOIN Orders;
```

VIEW3: ORDER DETAILS

```
CREATE VIEW orderDetailsView AS
        SELECT OrderID, orderDate, productID, productName, SUM(productCount) AS
productCount, productPrice
        FROM OrderContains NATURAL JOIN Orders NATURAL JOIN Products
        GROUP BY OrderID, orderDate, productID, productName, productPrice
        ORDER BY OrderID;
```

VIEW4: **RETURN DETAILS**

```
CREATE VIEW returnDetailsView AS
        SELECT orderID, RefundDate, productID, productName, SUM(productCount) AS
productCount, productPrice
        FROM Returns NATURAL JOIN OrderContains NATURAL JOIN Products
        GROUP BY RefundDate, productID, productName, productPrice, OrderID
        ORDER BY OrderID;
```

VIEW 5: CUSTOMER DETAILS

```
CREATE VIEW customerDetailsView AS
        SELECT c_ID, c_Name, c_Email, c_Address, c_Balance, COUNT (DISTINCT
cardNumber) AS Credit_Cards, COUNT(orderID) AS Orders
        FROM Customers NATURAL JOIN PlacesOrder
        GROUP BY c_ID, c_Name, c_Email, c_Address, c_Balance
        ORDER BY c_ID;
```

VIEW6: ORDER BILLING

```
CREATE VIEW billingDetailsView AS
        SELECT c_ID, orderID, cardNumber, orderPrice AS Price_OR_Refund, type
        FROM Orders NATURAL JOIN PlacesOrder
        ORDER BY c_ID;
```

PART E - INDEXES

ON CUSTOMERS:

CREATE INDEX customerIndex ON Customers(c_ID);

This Index will aid in making the Trigger update of the balance quicker (after insert into PlacesOrder) and helps the foreign key to PlacesOrder.

———————————————————————————————————————

ON ORDERS:

CREATE INDEX orderIDIndex ON Orders(orderID);
- To improve joins of tables for the Return Details and Return views

CREATE INDEX orderID_AND_DATE_Index ON Orders(orderID, orderDate);
- To improve joins of tables for the orderShipment and orderDetails views

———————————————————————————————————————

ON PRODUCTS:

CREATE INDEX productID_AND_NAME_Index ON Products(productID, productName);
- improves joins on views OrderContains and ReturnContains

PART F - CONSTRAINTS

```
ALTER TABLE OrderContains ADD CONSTRAINT check_productCount 
CHECK (productCount <= 1000);
```

```
ALTER TABLE PlacesOrder ADD CONSTRAINT check_type CHECK 
(type = 'R' OR type = 'O' OR type = 'FR');
```

```
ALTER TABLE creditCards ADD CONSTRAINT check_digits CHECK 
(cardNumber > 999999999999999 AND cardNumber < 
10000000000000000);
```

PART G: TRIGGERS

TRIGGER 1:

```
CREATE TRIGGER insertOrdersTrigger
        BEFORE INSERT ON OrderContains
        FOR EACH ROW
        EXECUTE PROCEDURE insertOrder();

CREATE FUNCTION insertOrder()
        RETURNS trigger AS $BODY$
DECLARE
n int;
inTable int;
BEGIN
        n:= (SELECT productPrice FROM Products WHERE productID =
NEW.productID);
         n:= n*NEW.productCount;
        inTable := (SELECT count(orderID) FROM Orders Where orderID =
NEW.orderID);
        IF (inTable = 0) THEN
                INSERT INTO Orders (orderID, orderDate, orderPrice)
                VALUES(NEW.orderID, current_date, n);
        END IF;
        IF (inTable = 1) THEN
                UPDATE Orders SET orderPrice = orderPrice + n WHERE orderID =
NEW.orderID;
        END IF;
        RETURN NEW;
END;
$BODY$ LANGUAGE plpgsql;
```

TRIGGER 2:

```
CREATE TRIGGER insertReturnTrigger
	AFTER INSERT ON PlacesOrder
	FOR EACH ROW
	EXECUTE PROCEDURE insertReturn();

CREATE FUNCTION insertReturn()
	RETURNS trigger AS $BODY$
DECLARE
n int;
updateBalance int;
BEGIN
	updateBalance:= (SELECT orderPrice FROM Orders WHERE orderID = New.orderID);
	IF (NEW.type = 'R' OR NEW.type = 'FR') THEN
		INSERT INTO Returns (orderID, RefundDate)
		VALUES(NEW.orderID, current_date + integer '5');
		UPDATE Customers SET c_Balance = c_Balance + updateBalance WHERE
NEW.c_ID = c_ID;
	END IF;
	IF (NEW.type = 'O') THEN
		UPDATE Customers SET c_Balance = c_Balance - updateBalance WHERE
NEW.c_ID = c_ID;
	END IF;
	IF(NEW.type = 'FR' IS TRUE) THEN
		UPDATE ShippingInfo SET ArrivalDate = current_date + integer '3';
	ELSE
		n:= (SELECT SUM(productCount) FROM orderContains WHERE New.orderID =
orderID);
		IF(n > 10) THEN
			INSERT INTO ShippingInfo (orderID, Company, ArrivalDate)
				VALUES(NEW.orderID, 'UPS', current_date + integer '2');
		END IF;
		IF (n <= 10) THEN
			INSERT INTO ShippingInfo (orderID, Company, ArrivalDate)
				VALUES(NEW.orderID, 'USPS', current_date + integer '4');
		END IF;
	END IF;
	RETURN NEW;
END;
$BODY$ LANGUAGE plpgsql;
```