**Online Retail Management System**

This report describes in detail every phase of designing and implementing a database system for an online retail store and includes several SQL queries to show its functionality.

**Team Members:**

Aviv Farag - [af3228@drexel.edu](mailto:af3228@drexel.edu)

Kerwin Trim - [kt822@drexel.edu](mailto:kt822@drexel.edu)

Ashley Wheeler - [anz27@drexel.edu](mailto:anz27@drexel.edu)

Joshua McNulty - [jdm455@drexel.edu](mailto:jdm455@drexel.edu)

Warren Webb - [wcw34@drexel.edu](mailto:wcw34@drexel.edu)

Drexel University

INFO 605: Introduction to Database Management

March 14, 2022

**TABLE OF CONTENTS**

[1. Summary 4](#_Toc98153890)

[2. Project Statement: 4](#_Toc98153891)

[2.1 Overall Goals of the System 4](#_Toc98153892)

[2.2 Context and Importance of the System 4](#_Toc98153893)

[2.3 Scope of the Project 5](#_Toc98153894)

[2.3.1 IN-Scope 5](#_Toc98153895)

[2.3.2 Out-Scope 5](#_Toc98153896)

[2.4 Related systems and any open-source tools 5](#_Toc98153897)

[3. Requirements: 6](#_Toc98153898)

[3.1 Data Requirements 6](#_Toc98153899)

[3.2 Business Rules and Logic 6](#_Toc98153900)

[3.3 Sample Output 7](#_Toc98153901)

[3.4 Other Assumptions 7](#_Toc98153902)

[4. Conceptual Design 8](#_Toc98153903)

[4.1 ERD 8](#_Toc98153904)

[4.2 Explanations 9](#_Toc98153905)

[5. Translating ERD 10](#_Toc98153906)

[5.1 Relational Schema 10](#_Toc98153907)

[5.2 Referential Integrity 12](#_Toc98153908)

[6. Data Dictionary 13](#_Toc98153909)

[6.1 Customer 13](#_Toc98153910)

[6.2 Payment 13](#_Toc98153911)

[6.3 Product 13](#_Toc98153912)

[6.4 Orders 14](#_Toc98153913)

[6.5 Shipment 14](#_Toc98153914)

[6.6 LineItem 14](#_Toc98153915)

[6.7 Returns 14](#_Toc98153916)

[6.8 Cart 15](#_Toc98153917)

[6.9 Shoes 15](#_Toc98153918)

[6.10 Top 15](#_Toc98153919)

[6.11 Bottom 15](#_Toc98153920)

[7. Database Implementation – Create Commands 16](#_Toc98153921)

[8. Data 22](#_Toc98153922)

[8.1 Insertion Commands 22](#_Toc98153923)

[8.2 Select Commands 29](#_Toc98153924)

[9. Data Queries 35](#_Toc98153925)

[9.1 Queries by Aviv Farag 35](#_Toc98153926)

[9.2 Queries by Joshua McNulty 37](#_Toc98153927)

[9.3 Queries by Warren Webb 39](#_Toc98153928)

[9.4 Queries by Trim Kerwin 41](#_Toc98153929)

[9.5 Queries by Ashley Wheeler 43](#_Toc98153930)

[10. Data Manipulation (Update and Delete) 45](#_Toc98153931)

[10.1 Queries by Aviv Farag 45](#_Toc98153932)

[10.2 Queries by Joshua McNulty 47](#_Toc98153933)

[10.3 Queries by Warren Webb 49](#_Toc98153934)

[10.4 Queries by Trim Kerwin 51](#_Toc98153935)

[10.5 Queries by Ashley Wheeler 53](#_Toc98153936)

[11. Summary 55](#_Toc98153937)

[11.1 Summary by Aviv Farag 55](#_Toc98153938)

[11.2 Summary by Joshua McNulty 55](#_Toc98153939)

[11.3 Summary by Warren Webb 56](#_Toc98153940)

[11.4 Summary by Trim Kerwin 56](#_Toc98153941)

[11.5 Summary by Ashley Wheeler 57](#_Toc98153942)

[11. Appendix – Division of work 57](#_Toc98153943)

[References 57](#_Toc98153944)

## 1. Summary

In this project, we developed and implemented a database system for an online retail store. We focused on products related to sports, such as Top (e.g., T-shirts), Bottoms (e.g., tights) and shoes (e.g., running shoes), but it can easily be modified to include other types of products. The first phase was writing all of the requirements and translating them into an Entity Relationship Diagram (ERD) model. Next, we converted the conceptual model into a relational schema as well as a data dictionary to define the datatypes of each attribute. Based on this step, we implemented the database by writing SQL create commands, and inserted data using insert commands. Here, we found a few flaws in our initial design as well as in several datatypes' declarations. Every error we encountered was solved by going back to the previous steps and modifying the design. Finally, we tested the database by querying, deleting, and updating the database. All those steps are outlined in this paper.

## 2. Project Statement:

### 2.1 Overall Goals of the System

In recent years, retail stores have found value in selling their products online through an e-commerce platform, such as Amazon, or by developing a website. The rise in online retail stores inspired our project that focuses on an online retail management system. We aim to develop an efficient system that allows storing and manipulating current inventory, filtering products according to users’ needs, managing customers, which is an integral part of a business, and storing cart items for every customer to better experience shopping online. These details will allow us to create a successful online retail business with scalability and necessary features for future growth.

We would like to take advantage of the rapidly growing online shopping in order to increase our customer base. Each customer has a first name, last name, email address and shipping address which must be valid in order to purchase products. A customer may create an order that contains multiple products. Payment must be either a credit card or a debit card since it is an online shopping experience. There are different types of products that we sell: shoes, tops (e.g., shirts) and bottoms (e.g., pants). Each product has a name, quantity, price per unit, description, color, section (e.g., men) and category (e.g. basketball). A customer can also add products to the cart and create an order whenever they would like to. However, products might be sold out, so the customer will be notified if one or more products in the cart are no longer available.

## 2.2 Context and Importance of the System

According to the most recent statistical data published by the United States Census Bureau, sales from e-commerce for U.S. retailers rose 14.3% from $506.1 billion in 2018 to $578.5 billion in 2019. The data also shows an increase of 3% in the total sales between 2018 and 2019 [1]. Online shopping has likely increased because it is a convenient way to shop. One can immediately go through many products by utilizing filtering features. Online shopping allows the customers to save time; not only does it save time looking for a specific product, but it also saves the time that the customer would have spent on the road driving to a physical store.

With the recent shift in shopping ideology, partially impacted by the pandemic, the world is at an all-time high for ordering all necessary goods remotely or through the internet; the same applies to clothing and attire items. The online sales industry has become a competitive market and will not cease to operate. Over the coming years, there is nothing, but exceptional growth projected for this market.

2.3 Scope of the Project

2.3.1 IN-Scope

1. Customers
2. Product (Shoes, tops, bottoms)
3. Order
4. LineItem
5. Cart
6. Payment
7. Shipping
8. Returns

### 2.3.2 Out-Scope

1. Membership
2. Gift cards
3. Customer service requests
4. Alternative types of payment (Cash,Cheque,Cryptocurrency,etc)
5. Exchange of items (trading an item for another item)
6. Insurance/liability for items that may be damaged or lost.
7. Insurance/liability for payment methods malfunctioning.
8. Vouchers for returned items.
9. Multiple Language Settings.
10. Social Media.
11. Security(verification/MFA)
12. Webpage compatibility/mobile webpage.
13. Images correlated with product

## 2.4 Related systems and any open-source tools

* **GitHub (Store files) -** [**https://github.com/avivfaraj/INFO605-Project**](https://github.com/avivfaraj/INFO605-Project)
* **Draw.io (ERD)**

# 3. Requirements:

## 3.1 Data Requirements

* Each customer has a customer ID, first name, middle name, last name, cellphone number, email, and shipping address (including street, city, state, and zip).
* A customer can save zero or more products in a cart, for each product must specify the amount. Each product can be saved in zero or more carts.
* An Order has order ID, total price, tax, date, time. Every order can have one or more items and one customer. Each customer can have zero or more orders. Every order also has one payment method, and every payment has one order.
* Payment has payment ID, total, type (debit, credit, etc.), credit card 4 digits, credit card company (VISA, AMEX, etc.).
* Every product has product ID, name, description, quantity, unit price, section (Men, Women, Kids), color, collection, and category (basketball, football, baseball, etc.). Every product can appear one or more in an order, and every order has one or more products.
* Shoe **is a** product. Every pair of shoes has US size, EU size.
* Top (e.g. shirts, jackets, etc.) **is a** product. Each top has fit type (regular, tall, athletic), size (S, M, L), type (t-shirt, jacket, sleeveless), and material (Cotton, polyester).
* Bottom (e.g., pants, shorts, sweatpants, etc.) **is a** product. Each bottom has fit type (compression, regular), size (S, M,L) and material (Cotton, polyester).
* A product must belong to one of the following: shoes, tops, bottoms.
* Every order can be shipped in one or more shipments, but every shipment must be associated with one order. Shipment includes tracking number and address.
* Items can be returned by a customer. There could be one or more items that are returned by exactly one customer.

## 3.2 Business Rules and Logic

* Tax percentage will be fixed based on the location of the warehouse.
* When a customer creates an order, an email will be sent to the customer with the receipt.
* Accepted payments are limited to either credit cards or debit cards.
* Payment information includes 4 digits and company name. The ID will be generated based on both (first digit is for company and 4 other digits are the card’s last digits)
* Generation of Customer ID – Holds personal information of customer. Auto-generates prefix and number of characters in the ID
* Generation of Order ID - Holds information of the order (price before and after tax, the tax, time, date). Auto-generates a reference number of the sale.
* Each order must be created by one customer.
* Each customer must have a valid address for shipping the products. If the address is not valid, the customer cannot create an order.
* If a product’s quantity in the inventory is low, an email will be sent to all customers that have this product in their cart to notify them that it might be sold out soon.
* Each order can have only one method of payment
* Transaction Triggers - May be suspicious in nature. Transactions over a certain amount could require an inquiry from a representative. Transactions that have an international address or address other than the one found in the customer ID may be denied.

## 3.3 Sample Output

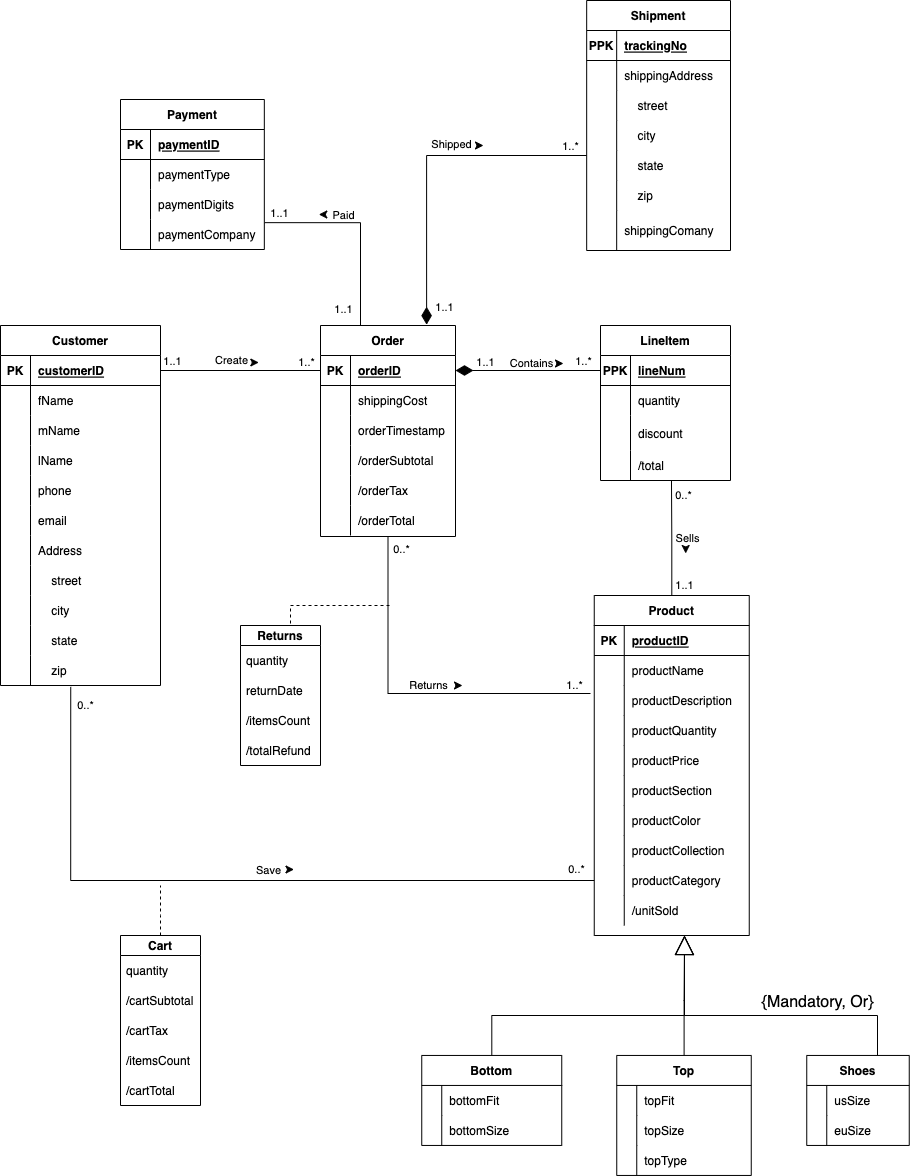
* Output of computerized bills/ receipts– After every transaction the system will produce a bill/ receipt to verify proof of purchase and so that customers also have proof for returns. This will act as confirmation of purchase and ensure the monetary transaction information is correct.
* Output of Stock and Inventory List – The system at the end of each day will produce a stock list for the store manager to review so they can cross check invoice after the day against what was sold. This will allow for accurate inventory listing and will further help avoid any confusion pertaining to product storage.
* Customer distribution among states. The system will count the customers in each state and create a report. This might be helpful for marketing strategy.
* List of the items that were sold every day. This report might indicate which items are more popular than others and can be helpful in inventory management. This can also assist with creating a daily or weekly inventory listing to help keep track of inventory management with reference to items being high in stock, low in stock, or out of stock.
* Output of items sold. This can assist the business in finding the most popular products and then further breaking down product popularity with reference to the region or location of the customers. This can allow marketing to utilize a specialized strategy with reference to advertising based on location.
* Output of revenue per location. This can be broken down by state, city, or zip code. It will allow the business to understand where the most revenue is being generated and further allow for marketing strategies to target these areas.
* Output of returning customer information. Displaying and gathering statistics about returning customers or lack thereof will allow the business to understand the strongest customer base by location. This could be a way of rewarding customers (if we decide to do discounts).

## 3.4 Other Assumptions

* Our system will be able to handle a huge volume of data since this will be used in stores at several different locations and an online store
* Our system will help us to retrieve data quickly so that we can analyze trends and make predictions for our business
* Our site will appear in related search engine searches.
* Our customer base will be productive and make multiple transactions over time.
* Our products will be at the top of the line and attractive for customers

# 4. Conceptual Design

## 4.1 ERD



## 4.2 Explanations

**Payment:** The payment entity is the first entity in the ER diagram. Its primary key is payment ID, and it has three attributes: Payment type, Payment digit and Payment company. It has a binary relationship with the Order entity where each order has one payment listed and each payment is associated with one order.

**Order:** The Order entity is next in line in the ER diagram. Its primary key is Order ID, and it has three attributes: Order Subtotal, Order Tax and Order Total. It has a binary relationship with Payment, Line Item, Customer, Shipment and Product. For Payment, Order has a relationship where every order has a payment, and every payment is associated with an order. For Line Item, every order can contain one or more line item and a line item can be associated with one order. With Customer, Customers can create one or more orders and an order will have one customer.

**Customer:** In our customer entity our PK is customer ID, and the attributes are customer First Name, customer Middle Name, customer Last Name, customer Phone, customer Email, customer Address, street, city, state and zip. Customer has a binary relationship with Order. Customer’s relationship with Order is that a customer can create one or more orders and an order will have one customer.

**Line Item:** has a PPK that is line number. Line item is a weak entity because it is dependent on Order. It has three attributes: quantity, unit price and total which is a derived attribute. Line item has a binary relationship with Order and Product. With an Order, it can contain one or more line item and one line item can be associated with one order. In the case of Product, a product can have zero or many line items and each line item must be associated with one product.

**Cart:** The cart is a result of the relationship between the customer and the product. Due to this many to many relationship, the cart will be give n the primary key of both of these entities. The cart will also be responsible for maintaining the quantity of total items within itself, and will be attributed other derived attributes such as cartSubtotal, cartTax, itemsCount, and cartTotal.

**Shipment:** The primary key is composed of both orderID and TrackingNo. The address is also recorded for every shipment. There could be one or more shipments for each order, so there is a one-to-many relationship between shipment and LineItem.

**Returns:** Due to the many to many relationship that is formed between the order and product entities, the returns table is created. The returns table will be given the primary keys of both of the strong entities. It will also be attributed the quantity of items to be returned, the date of the return, and derived attributes such as itemsCount and totalRefund.

**Product:** has a primary key represented by product ID and it has several attributes: Product name, product description, product quantity, product price, product section, product color, product collection, product category and units sold which is a derived attribute. Product has a binary relationship with Cart and Line Item. With Cart, a product may be associated with 1 cart, and a cart will have zero or many products inside it. When it comes to Product and Line Item, a product may have zero or more line items and a line item may have one product associated with it.

**Top:** has four direct attributes given to it. These are: topFit, topSize, and topType. The above attributes of the entity, Product may also be given to Top through inheritance, in a mandatory, or relationship.

**Bottom:** has three direct attributes given to it. These are: bottomFit, and bottomSize. The above attributes of the entity, Product may also be given to Bottom through inheritance, in a mandatory, or relationship.

**Shoes:** has two direct attributes given to it. These are: usSize and euSize. The above attributes of the entity, Product may also be given to Shoes through inheritance, in a mandatory, or relationship.

## 5. Translating ERD

### 5.1 Relational Schema

Customer(**customerID**, fName, mName, lName, phone, email, street, city, state, zip)

Payment(**paymentID**, paymentType, paymentDigits, paymentCompany)

Product(**productID**, productName, productDescription, productQuantity, productPrice, productSection, productColor, productCollection, productCategory)

Orders(**orderID**, shippingCost, orderTimestamp , customerID, paymentID)

FOREIGN KEY (customerID) REFERENCE Customer(customerID) ON DELETE CASCADE

FOREIGN KEY (paymentID) REFERENCE Payment(paymentID) ON DELETE CASCADE

Shipment(**trackingNo**, **orderID** , street, city, state, zip, shippingCompany)

FOREIGN KEY (orderID) REFERENCE Orders(orderID) ON DELETE CASCADE

LineItem(**line#**, **orderID** , quantity, discount, productID )

FOREIGN KEY (orderID) REFERENCE Orders(orderID) ON DELETE CASCADE

FOREIGN KEY (productID) REFERENCE Product(productID) ON DELETE CASCADE

Returns(**orderID**, **productID**, returnDate, quantity)

FOREIGN KEY (orderID) REFERENCE Orders(orderID) ON DELETE CASCADE

FOREIGN KEY (productID) REFERENCE Product(productID) ON DELETE CASCADE

Cart(**customerID**, **productID**, quantity)

FOREIGN KEY (customerID) REFERENCE Customer(customerID) ON DELETE CASCADE

FOREIGN KEY (productID) REFERENCE Product(ProductID) ON DELETE CASCADE

Shoes(**shoesID**, usSize, euSize)

FOREIGN KEY (**shoesID**) REFERENCE Product(productID) ON DELETE CASCADE

Top(**topID**, topFit, topSize, topType)

FOREIGN KEY (**topID**) REFERENCE Product(productID) ON DELETE CASCADE

Bottom(**bottomID**, bottomFit, bottomSize, bottomType)

FOREIGN KEY (**bottomID**) REFERENCE Product(productID) ON DELETE CASCADE

### 5.2 Referential Integrity



## 6. Data Dictionary

### 6.1 Customer

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| customerID | NOT NULL | NUMBER(10) | Primary Key |
| fName | NOT NULL | VARCHAR2(20) |  |
| mName |  | VARCHAR2(20) |  |
| lName | NOT NULL | VARCHAR2(20) |  |
| phone |  | NUMBER(10) | unique |
| email | NOT NULL | VARCHAR2(60) | unique |
| street | NOT NULL | VARCHAR2(50) |  |
| city | NOT NULL | VARCHAR2(20) |  |
| state | NOT NULL | CHAR(2) |  |
| zip | NOT NULL | CHAR(5) |  |

### 6.2 Payment

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| paymentID | NOT NULL | NUMBER(6) | Primary Key |
| paymentType | NOT NULL | VARCHAR2(6) | “Debit” or “Credit” |
| paymentDigits | NOT NULL | NUMBER(4) |  |
| paymentCompany | NOT NULL | VARCHAR2(16) | “Visa”, “Mastercard”  “Amex”, “Bank of America”, “Capital One” |

6.3 Product

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| productID | NOT NULL | NUMBER(10) | Primary Key |
| productName | NOT NULL | VARCHAR2(50) |  |
| productDescription |  | VARCHAR2(50) |  |
| productQuantity | NOT NULL | NUMBER(3) |  |
| productPrice | NOT NULL | NUMBER(6,2) |  |
| productSection | NOT NULL | VARCHAR(5) | “Women”, “Men” or “Kids” |
| productColor | NOT NULL | VARCHAR2(30) |  |
| productCollection |  | VARCHAR2(30) |  |
| productCategory | NOT NULL | VARCHAR2(30) |  |

### 6.4 Orders

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| orderID | NOT NULL | NUMBER(10) | Primary Key |
| shippingCost |  | NUMBER(6,2) |  |
| OrderTimestamp |  | DATE |  |
| customerID | NOT NULL | NUMBER(10) | FK(Customer) |
| paymentID |  | NUMBER(6) | FK(Payment) |

6.5 Shipment

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| trackingNo | NOT NULL | VARCHAR2(15) | Partial Primary Key |
| orderID | NOT NULL | NUMBER(10) | FK(Order) &  Partial Primary Key |
| street | NOT NULL | VARCHAR2(20) |  |
| city | NOT NULL | VARCHAR2(20) |  |
| state | NOT NULL | CHAR(2) |  |
| zip | NOT NULL | CHAR(5) |  |
| shippingCompany | NOT NULL | VARCHAR2(20) |  |

### 6.6 LineItem

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| line# | NOT NULL | NUMBER(2) | Partial Primary Key |
| orderID | NOT NULL | NUMBER(10) | FK(Order) &  Partial Primary Key |
| quantity | NOT NULL | NUMBER(1) |  |
| discount | NOT NULL | NUMBER(6,2) |  |
| productID | NOT NULL | NUMBER(10) | FK(Product) |

6.7 Returns

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| orderID | NOT NULL | NUMBER(10) | FK(Order) &  Partial Primary Key |
| productID | NOT NULL | NUMBER(10) | FK(product) &  Partial Primary Key |
| returnDate | NOT NULL | DATE |  |
| quantity | NOT NULL | NUMBER(2) |  |

### 6.8 Cart

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| customerID | NOT NULL | NUMBER(10) | PPK |
| productID | NOT NULL | NUMBER(10) | PPK |
| quantity |  | NUMBER(1) | DEFAULT 1 |

6.9 Shoes

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| shoesID | NOT NULL | NUMBER(10) | FK(Product) &  Primary Key |
| usSize | NOT NULL | NUMBER(3,1) |  |
| euSize | NOT NULL | NUMBER(3,1) |  |

6.10 Top

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| topID | NOT NULL | NUMBER(10) | FK(Product) &  Primary Key |
| topFit | NOT NULL | VARCHAR2(7) | “Regular” or “Tall” |
| topSize | NOT NULL | VARCHAR2(10) | “S”, “M”, “L”, “XL”, “XXL”, or “XXXL” |
| topType | NOT NULL | VARCHAR2(11) | “T-shirt”, “Long sleeve”, “Polo Shirt”, “Sweatshirt” or “Sleeveless” |

6.11 Bottom

|  |  |  |  |
| --- | --- | --- | --- |
| Column | NULL? | Type | Comments |
| bottomID | NOT NULL | NUMBER(10) | FK(Product) &  Primary Key |
| bottomFit | NOT NULL | VARCHAR2(7) | “Regular” or “Tall” |
| bottomSize | NOT NULL | VARCHAR2(10) | “S”, “M”, “L”, “XL”, “XXL”, or “XXXL” |
| bottomType | NOT NULL | VARCHAR2(11) | "Sweatpants”, “leggings”, “Pants”, or “Shorts”. |

## 7. Database Implementation – Create Commands

--DROP TABLE Customer CASCADE CONSTRAINTS;

CREATE TABLE Customer (

customerID NUMBER(10) NOT NULL,

fName VARCHAR2(20) NOT NULL,

mName VARCHAR2(20),

lName VARCHAR2(20) NOT NULL,

phone  NUMBER(10) UNIQUE,

email VARCHAR2(60) NOT NULL UNIQUE,

street VARCHAR2(50) NOT NULL,

city VARCHAR2(20) NOT NULL,

state CHAR(2) NOT NULL,

zip  CHAR(5) NOT NULL,

CONSTRAINT customer\_pk PRIMARY KEY (customerID));

--DROP TABLE Payment CASCADE CONSTRAINTS;

CREATE TABLE Payment(

paymentID  NUMBER(6) NOT NULL,

paymentType VARCHAR2(6) NOT NULL

CHECK (paymentType in ('Debit', 'Credit')),

paymentDigits  NUMBER(4) NOT NULL,

paymentCompany VARCHAR2(16) NOT NULL,

CHECK (paymentCompany in ('Visa','AMEX', 'Mastercard', 'Bank of America', 'Capital One')),

CONSTRAINT payment\_pk PRIMARY KEY (paymentID));

--DROP TABLE Product CASCADE CONSTRAINTS;

CREATE TABLE Product(

productID  NUMBER(10) NOT NULL,

productName VARCHAR2(50) NOT NULL,

productDescription VARCHAR2(50),

productQuantity  NUMBER(3) NOT NULL,

productPrice  NUMBER(6,2) NOT NULL,

productSection VARCHAR2(5) NOT NULL

CHECK (productSection IN ('Women','Men', 'Kids')),

productColor VARCHAR2(30) NOT NULL,

productCollection VARCHAR2(30),

productCategory VARCHAR2(30) NOT NULL,

CONSTRAINT product\_pk PRIMARY KEY (productID));

--DROP TABLE Orders CASCADE CONSTRAINTS;

CREATE TABLE Orders(

orderID NUMBER(10) NOT NULL,

shippingCost  NUMBER(6,2),

orderTimestamp DATE,

customerID  NUMBER(10) NOT NULL,

paymentID  NUMBER(6),

CONSTRAINT orders\_pk PRIMARY KEY (orderID),

CONSTRAINT orders\_fk1 FOREIGN KEY (customerID) REFERENCES Customer (customerID) ON DELETE CASCADE,

CONSTRAINT orders\_fk2 FOREIGN KEY (paymentID) REFERENCES Payment(paymentID) ON DELETE SET NULL);

--DROP TABLE Shipment CASCADE CONSTRAINTS;

CREATE TABLE Shipment(

trackingNo VARCHAR2(15) NOT NULL,

orderID  NUMBER(10) NOT NULL,

street VARCHAR2(20) NOT NULL,

city VARCHAR2(20) NOT NULL,

state  CHAR(2) NOT NULL,

zip  CHAR(5) NOT NULL,

shippingCompany VARCHAR2(20) NOT NULL,

CONSTRAINT shipment\_pk PRIMARY KEY (trackingNo, orderID),

CONSTRAINT shipmemt\_fk FOREIGN KEY (orderID) REFERENCES Orders (orderID) ON DELETE CASCADE);

--DROP TABLE LineItem CASCADE CONSTRAINTS;

CREATE TABLE LineItem (

line# NUMBER(2) NOT NULL,

orderID NUMBER(10) NOT NULL,

quantity NUMBER(1) DEFAULT 1 NOT NULL,

discount NUMBER(6,2) NOT NULL,

productID NUMBER(10) NOT NULL,

CONSTRAINT lineItem\_pk PRIMARY KEY (line#, orderID),

CONSTRAINT lineItem\_fk1 FOREIGN KEY (orderID) REFERENCES Orders(orderID) ON DELETE CASCADE,

CONSTRAINT lineItem\_fk2 FOREIGN KEY (productID) REFERENCES Product(productID) ON DELETE CASCADE);

--DROP TABLE Returns CASCADE CONSTRAINTS;

CREATE TABLE Returns(

orderID  NUMBER(10) NOT NULL,

productID NUMBER(10) NOT NULL,

returnDate DATE NOT NULL,

quantity  NUMBER(2) NOT NULL,

CONSTRAINT returns\_pk PRIMARY KEY (orderID, productID),

CONSTRAINT returns\_fk1 FOREIGN KEY (orderID) REFERENCES Orders (orderID) ON DELETE CASCADE,

CONSTRAINT returns\_fk2 FOREIGN KEY (productID) REFERENCES Product(productID) ON DELETE CASCADE);

--DROP TABLE Cart CASCADE CONSTRAINTS;

CREATE TABLE Cart (

customerID NUMBER(10) NOT NULL,

productID  NUMBER(10) NOT NULL,

quantity  NUMBER(1) DEFAULT 1,

CONSTRAINT cart\_pk PRIMARY KEY (customerID, productID),

CONSTRAINT cart\_fk1 FOREIGN KEY (customerID) REFERENCES Customer(customerID) ON DELETE CASCADE,

CONSTRAINT cart\_fk2 FOREIGN KEY (productID) REFERENCES Product(productID) ON DELETE CASCADE);

--DROP TABLE Shoes cascade CONSTRAINTS;

CREATE TABLE Shoes(

shoesID NUMBER(10) NOT NULL,

usSize  NUMBER(3,1) NOT NULL,

euSize  NUMBER(3,1) NOT NULL,

CONSTRAINT shoes\_pk PRIMARY KEY (shoesID),

CONSTRAINT shoes\_fk FOREIGN KEY (shoesID) REFERENCES Product(productID) ON DELETE CASCADE);

--DROP TABLE Top cascade CONSTRAINTS;

CREATE TABLE Top(

topID  NUMBER(10) NOT NULL,

topFit VARCHAR2(7) NOT NULL

CHECK(topFit IN ('Regular', 'Tall')),

topSize VARCHAR2(10) NOT NULL

CHECK(topSize IN ('S', 'M', 'L', 'XL', 'XXL', '3XL')),

topType VARCHAR2(11) NOT NULL

CHECK(topType IN ('T-shirt', 'Long Sleeve', 'Polo Shirt', 'Sweatshirt', 'Sleeveless')),

CONSTRAINT top\_fk FOREIGN KEY (topID) REFERENCES Product(productID) ON DELETE CASCADE,

CONSTRAINT top\_pk PRIMARY KEY (topID));

--DROP TABLE Bottom cascade CONSTRAINTS;

CREATE TABLE Bottom(

bottomID NUMBER(10) NOT NULL,

bottomFit VARCHAR2(7) NOT NULL

CHECK(bottomFit IN ('Regular', 'Tall')),

bottomSize VARCHAR2(10) NOT NULL

CHECK(bottomSize IN ('S', 'M', 'L', 'XL', 'XXL', '3XL')),

bottomType VARCHAR2(11) NOT NULL

CHECK(bottomType IN ('Sweatpants', 'Leggings', 'Pants', 'Shorts')),

CONSTRAINT bottom\_fk FOREIGN KEY (bottomID) REFERENCES Product(productID) ON DELETE CASCADE,

CONSTRAINT bottom\_pk PRIMARY KEY (bottomID));

GRANT SELECT ON Customer to PUBLIC;

GRANT SELECT ON Cart to PUBLIC;

GRANT SELECT ON Orders to PUBLIC;

GRANT SELECT ON Payment to PUBLIC;

GRANT SELECT ON Shipment to PUBLIC;

GRANT SELECT ON LineItem to PUBLIC;

GRANT SELECT ON Product to PUBLIC;

GRANT SELECT ON Shoes to PUBLIC;

GRANT SELECT ON Top to PUBLIC;

GRANT SELECT ON Bottom to PUBLIC;

GRANT SELECT ON Returns to PUBLIC;

COMMIT;

## 8. Data

### 8.1 Insertion Commands

ALTER SESSION SET nls\_date\_format='MM/DD/YYYY HH24:MI';

INSERT ALL

INTO Customer VALUES(1, 'Andrew','Tesla','McDonald',7575852822,'andrewTM@gmail.com','222 Wayne St.','Philadelphia','PA','19033')

INTO Customer VALUES(2, 'Michael','Herz','Jane',7576442822,'michaelj@gmail.com','10 Jefferson Rd.','Philadelphia','PA','19332')

INTO Customer VALUES(3, 'Kim',NULL,'Harrison',7579442334,'kimharrison@yahoo.com','56 Brooks Rd.','Brooklyn','NY','11345')

INTO Customer VALUES(4, 'Michael','Dawson','Evans',5652242334,'micdawson@outlook.com','23 Dover Ln.','Philadelphia','PA','19041')

INTO Customer VALUES(5, 'John','Mitchell','Robinson',5652212244,'johnmitch@icloud.com','22 Walter Dr.','Tampa','FL','33621')

INTO Customer VALUES(6, 'Haris','Hughes','Williamson',5772214244,'harison@icloud.com','340 Kensigton Rd.','Orlando','FL','33045')

INTO Customer VALUES(7, 'Thomas','Edison','Jefferson',5772216624,'thomasj@icloud.com','410 Herkimier','Brooklyn','NY','11614')

INTO Customer VALUES(8, 'Elai','Thompson','Clay',7272212624,'elait@gmail.com','410 Aurora St.','Austin','TX','33123')

INTO Customer VALUES(9, 'Clay','Richard','Thompson',7272102633,'clayt@gmail.com','410 Miami-Dade St.','Miami','FL','33125')

INTO Customer VALUES(10, 'Thomas','Richard','Oppenheim',7232042433,'oppentom@gmail.com','423 Traders St.','Cleveland','OH','12045')

SELECT \* FROM DUAL;

/\* paymentID Generation: company, type, 4digits

Companies:

Visa - 1

Mastercard - 2

Capital One - 3

AMEX - 4

Bank of America - 5

Type: Debit - 1, Credit - 2

Example: Visa (1), Credit(2), 8222 ----> paymentID = 128222

\*/

INSERT ALL

INTO Payment VALUES (118214, 'Debit', 8214, 'Visa')

INTO Payment VALUES (218751, 'Debit', 8214, 'Mastercard')

INTO Payment VALUES (327510, 'Credit', 7510, 'Capital One')

INTO Payment VALUES (421826, 'Credit', 1826, 'AMEX')

INTO Payment VALUES (110142, 'Debit', 0142, 'Visa')

INTO Payment VALUES (526179, 'Credit', 6179, 'Bank of America')

INTO Payment VALUES (413152, 'Debit', 3152, 'AMEX')

INTO Payment VALUES (114914, 'Debit', 4914, 'Visa')

INTO Payment VALUES (217162, 'Debit', 7162, 'Mastercard')

INTO Payment VALUES (320981, 'Credit', 0981, 'Capital One')

SELECT \* FROM DUAL;

INSERT ALL

INTO Product VALUES(1, 'Cold Gear Infared Hoodie', 'Athleisure', 25, 34.60, 'Men', 'Black', NULL, 'Basketball')

INTO Product VALUES(2, 'Rush Seamless Sleeveless', 'Training', 15, 60.00, 'Men', 'Grey', 'Project Rock', 'Special Collection')

INTO Product VALUES(3, 'Rush Smart Form Long Shorts', 'Training', 29, 70.00, 'Men', 'camo green', 'Project Rock', 'Special Collection')

INTO Product VALUES(4, 'Tribase Reign 4 Pro Shoes', 'Athleisure', 58, 140.00, 'Men', 'lime green', 'Reign 4', 'Special Collection')

INTO Product VALUES(5, 'Tech V-neck', 'Training', 81, 24.99, 'Women', 'Maroon', NULL, 'Women Training')

INTO Product VALUES(6,'Meridian Crop', 'Training', 35, 65.00, 'Women', 'Rose', 'Meridian', 'Women Training')

INTO Product VALUES(7, 'Meridian Joggers', 'Training', 55, 75.00, 'Women', 'Black', 'Meridian', 'Women Training')

INTO Product VALUES(8, 'Meridian Leggings', 'Training', 70, 40.00, 'Women', 'Khaki', 'UA Meridian', 'Women Training')

INTO Product VALUES(9, 'Rush Warm Up Hoodie', 'Athleisure', 45, 44.50, 'Men', 'Grey', NULL, 'Basketball')

INTO Product VALUES(10, 'Nike Air Max 97', 'Shoes built for runners', 100, 44.50, 'Women', 'Grey', NULL, 'Running')

INTO Product VALUES(11, 'Nike Air Max 97', 'Shoes built for runners', 100, 44.50, 'Women', 'Yellow', NULL, 'Running')

INTO Product VALUES(12, 'Nike Air Max 79', 'Shoes built for runners', 100, 44.50, 'Men', 'Shiny Grey', NULL, 'Running')

SELECT \* FROM DUAL;

-- TOP

INSERT ALL

INTO Top VALUES(1,'Regular','S','Long Sleeve')

INTO Top VALUES(2,'Regular','M','Sleeveless')

INTO Top VALUES(5,'Tall','M', 'Long Sleeve')

INTO Top VALUES(6,'Tall','L', 'Long Sleeve')

INTO Top VALUES(9,'Tall','M', 'Long Sleeve')

SELECT \* FROM DUAL;

-- Bottom

INSERT ALL

INTO Bottom VALUES(3,'Regular','M','Shorts')

INTO Bottom VALUES(7,'Tall','M', 'Leggings')

INTO Bottom VALUES(8,'Regular','M', 'Leggings')

SELECT \* FROM DUAL;

-- Shoes

INSERT ALL

INTO Shoes VALUES(4,9,42)

INTO Shoes VALUES(10,8.5,41.5)

INTO Shoes VALUES(11,9,42)

INTO Shoes VALUES(12,10,43.5)

SELECT \* FROM DUAL;

INSERT ALL

INTO Cart(customerID, productID, quantity) VALUES (1, 1, 2)

INTO Cart(customerID, productID, quantity) VALUES (1, 3, 1)

INTO Cart(customerID, productID, quantity) VALUES (1, 6, 4)

INTO Cart(customerID, productID, quantity) VALUES (2, 8, 2)

INTO Cart(customerID, productID, quantity) VALUES (2, 2, 2)

INTO Cart(customerID, productID, quantity) VALUES (4, 3, 2)

INTO Cart(customerID, productID, quantity) VALUES (4, 2, 1)

INTO Cart(customerID, productID, quantity) VALUES (5, 1, 1)

SELECT \* FROM DUAL;

INSERT ALL

INTO Orders VALUES(1, 5, TO\_DATE( '12/04/2020 10:45', 'MM/DD/YYYY HH24:MI'), 1, 118214)

INTO Orders VALUES(2, 5, TO\_DATE( '01/01/2021 13:00', 'MM/DD/YYYY HH24:MI'), 2, 218751)

INTO Orders VALUES(3, 5, TO\_DATE( '01/01/2021 13:30', 'MM/DD/YYYY HH24:MI'), 3, 327510)

INTO Orders VALUES(4, 5, TO\_DATE( '01/01/2021 13:40', 'MM/DD/YYYY HH24:MI'), 4, 421826)

INTO Orders VALUES(5, 5, TO\_DATE( '01/01/2021 13:50', 'MM/DD/YYYY HH24:MI'), 5, 110142)

INTO Orders VALUES(6, 5, TO\_DATE( '02/01/2021 09:40', 'MM/DD/YYYY HH24:MI'), 3, 327510)

INTO Orders VALUES(7, 5, TO\_DATE( '02/01/2021 13:40', 'MM/DD/YYYY HH24:MI'), 2, 218751)

INTO Orders VALUES(8, 5,TO\_DATE( '03/01/2021 15:25', 'MM/DD/YYYY HH24:MI'), 8, 114914)

INTO Orders VALUES(9, 5,TO\_DATE( '03/01/2021 18:00', 'MM/DD/YYYY HH24:MI'), 7, 413152)

SELECT \* FROM DUAL;

INSERT ALL

INTO lineItem VALUES(1, 1, 2, 0,2) -- Order #1

INTO lineItem VALUES(2, 1, 1, 5,4)

INTO lineItem VALUES(1, 2, 2, 0,7) -- Order #2

INTO lineItem VALUES(2, 2, 3, 5,4)

INTO lineItem VALUES(1, 3, 1, 0,5) -- Order #3

INTO lineItem VALUES(1, 4, 6, 5,4) -- Order #4

INTO lineItem VALUES(1, 5, 2, 0,6) -- Order #5

INTO lineItem VALUES(2, 5, 1, 0,3)

INTO lineItem VALUES(1, 6, 1, 0,7) -- Order #6

INTO lineItem VALUES(2, 6, 1, 0,9)

INTO lineItem VALUES(1, 7, 1, 0,7) -- Order #7

INTO lineItem VALUES(1, 8, 2, 0,1) -- Order #8

INTO lineItem VALUES(1, 9, 1, 0,9) -- Order #9

INTO lineItem VALUES(2, 9, 1, 0,3)

INTO lineItem VALUES(3, 9, 1, 0,2)

SELECT \* FROM DUAL;

INSERT ALL

INTO Shipment VALUES('A7FV4NS0', 1, '222 Wayne St.', 'Philadelphia', 'PA', '19033', 'FedEx')

INTO Shipment VALUES('SAMNS882M', 2, '10 Jefferson Rd.', 'Philadelphia', 'PA', '19332','UPS')

INTO Shipment VALUES('LKMN892NN', 3, '56 Brooks Rd.', 'Brooklyn', 'NY', '11345','UPS')

INTO Shipment VALUES('LMKSN02', 4, '23 Dover Ln.', 'Philadelphia', 'PA', '19041','FedEx')

INTO Shipment VALUES('ALPMMWEX', 5, '22 Walter Dr.', 'Tamp', 'FL', '33621','UPS')

INTO Shipment VALUES('LPIIEMM2', 6, '56 Brooks Rd.', 'Brooklyn', 'NY', '11345','UPS')

INTO Shipment VALUES('SAMNS882M', 7, '10 Jefferson Rd.', 'Philadelphia', 'PA', '19332','UPS')

INTO Shipment VALUES('SAMNS882M', 8, '410 Aurora St.', 'Austin', 'TX', '33123','FedEx')

SELECT \* FROM DUAL;

INSERT ALL

INTO Returns VALUES(1,2, TO\_DATE( '12/10/2020 10:00', 'MM/DD/YYYY HH24:MI'), 1)

INTO Returns VALUES(2,4, TO\_DATE( '03/01/2021 10:00', 'MM/DD/YYYY HH24:MI'), 2)

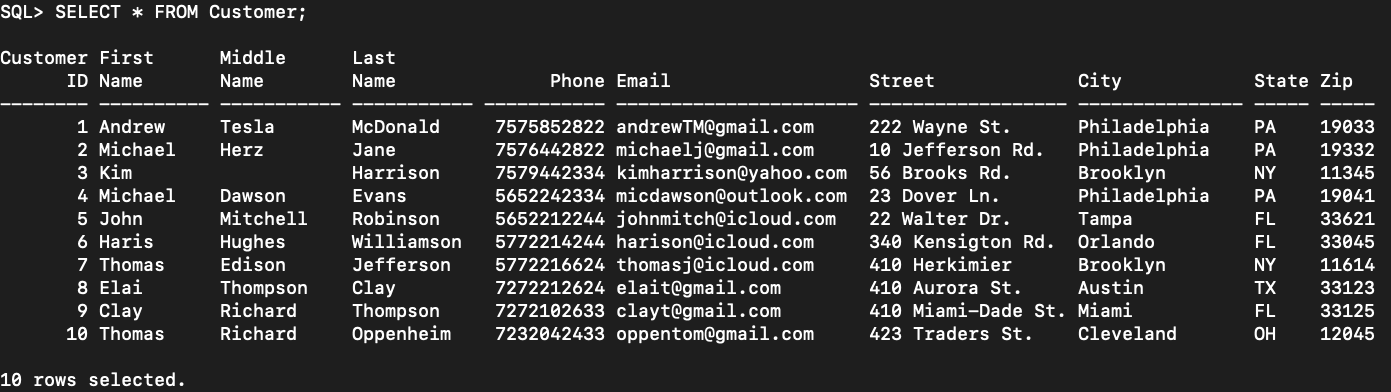
INTO Returns VALUES(8,1, TO\_DATE( '05/01/2021 10:00', 'MM/DD/YYYY HH24:MI'), 1)

SELECT \* FROM DUAL;

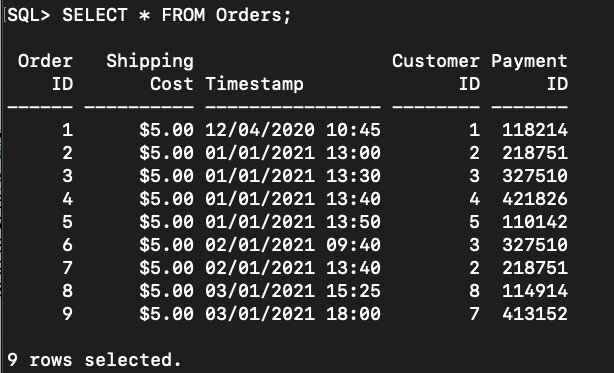
COMMIT;

### 8.2 Select Commands

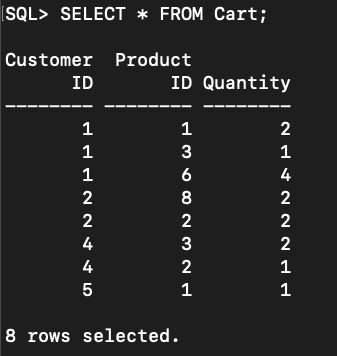
**Customer:**



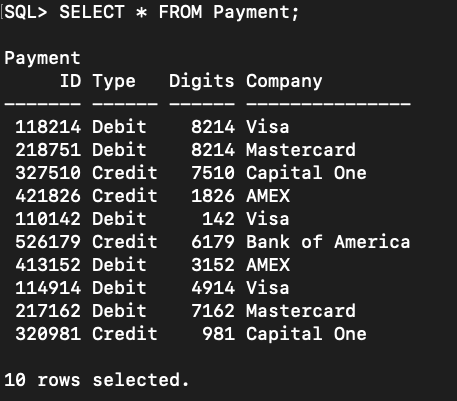
**Orders:**



**Cart:**

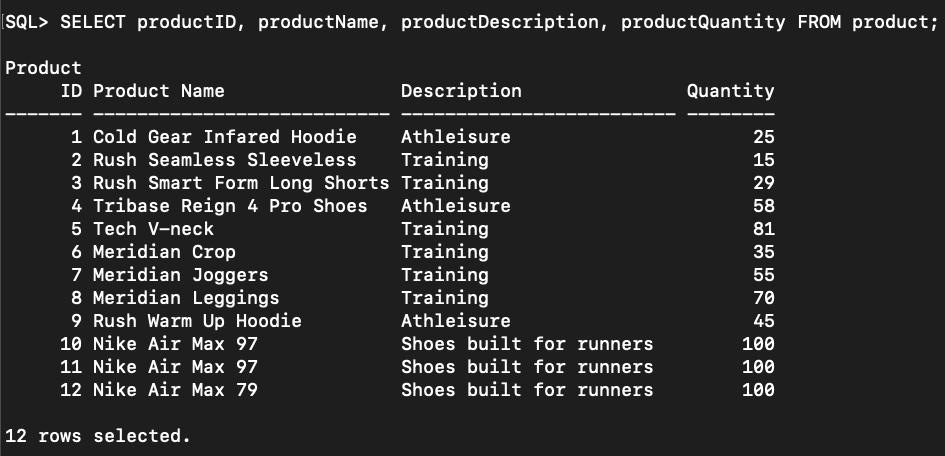


**Payment:**

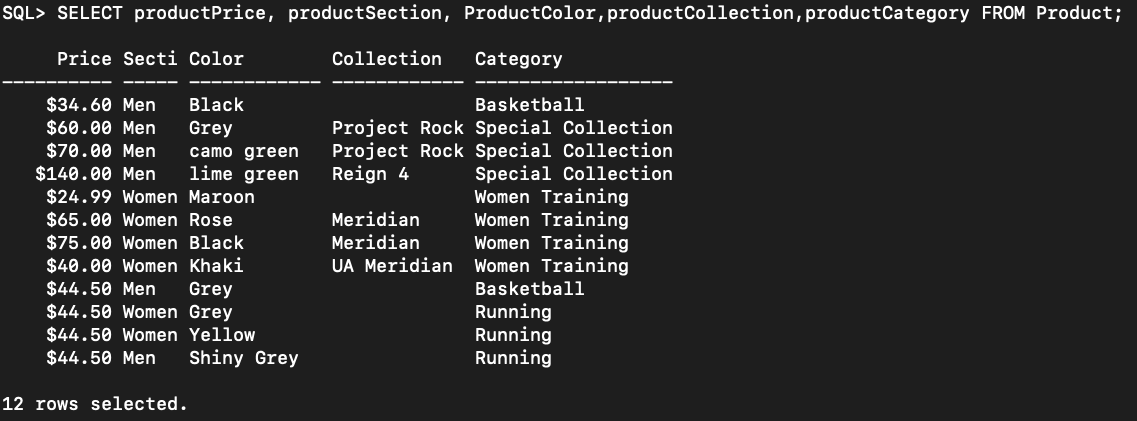


**Product:**

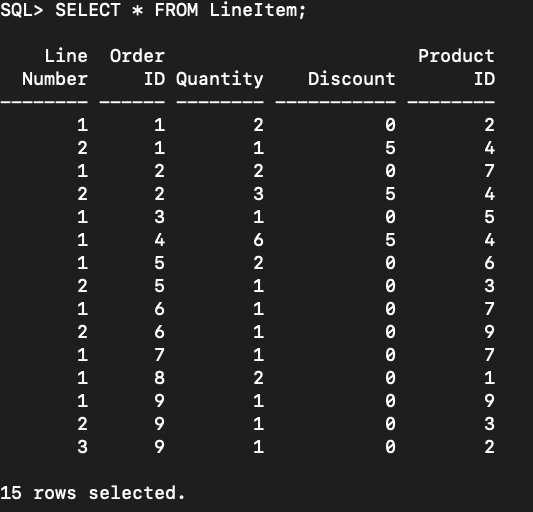
First half of Product table:



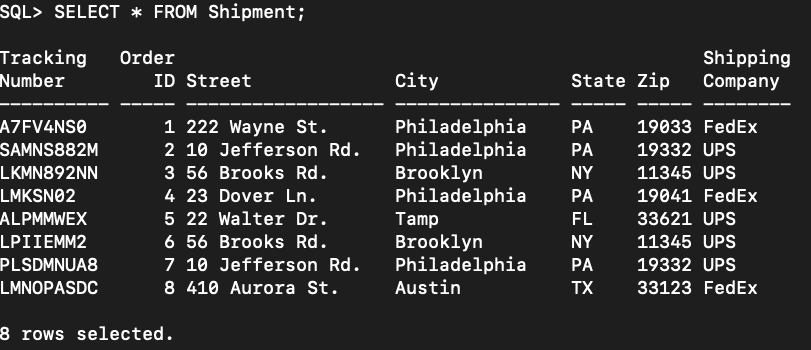
Second half of Product table:



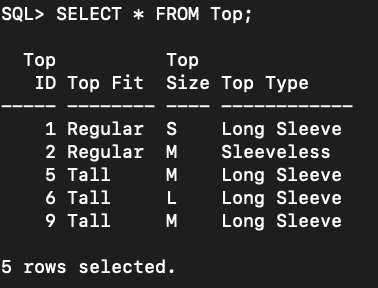
**LineItem:**



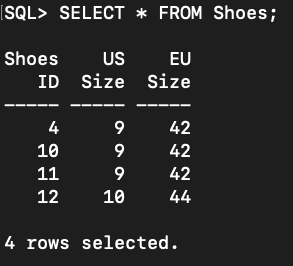
**Shipment:**



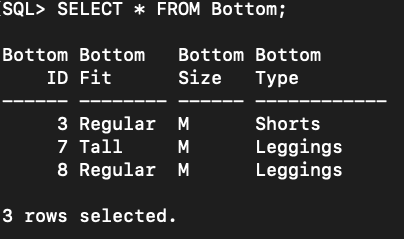
**Top:**



**Shoes:**



**Bottom:**

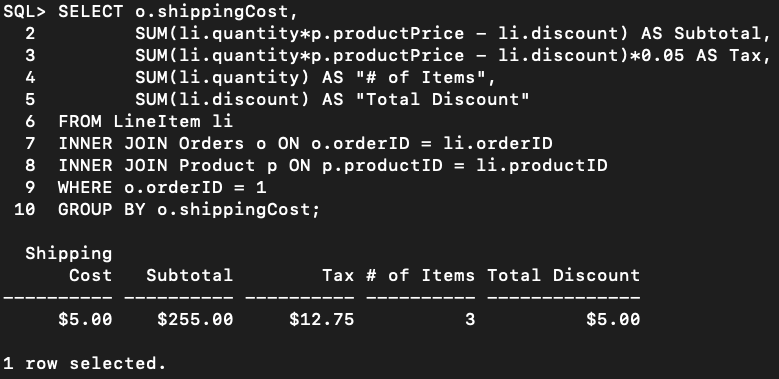


## 

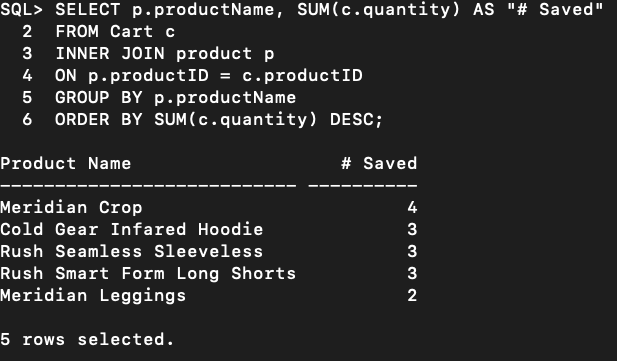
## 9. Data Queries

### 9.1 Queries by Aviv Farag

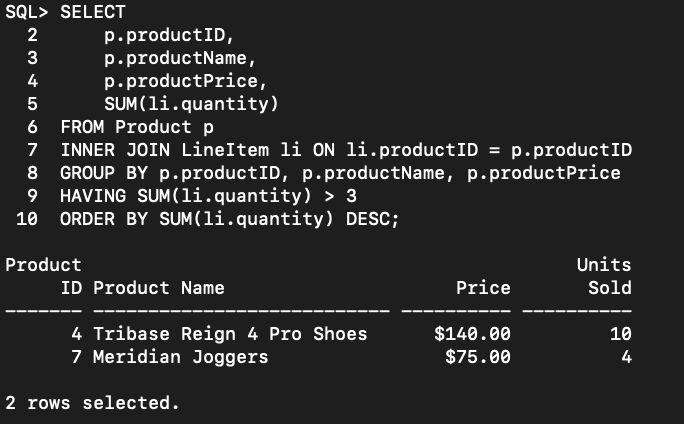
**Order #1 Summary: Print Shipping cost, subtotal, tax, the number of items ordered and total discount of order with orderID = 1:**



**Products that appear the most in all carts: Print the product name and number of times It was saved by all customers. Ordered by the number of times saved.**

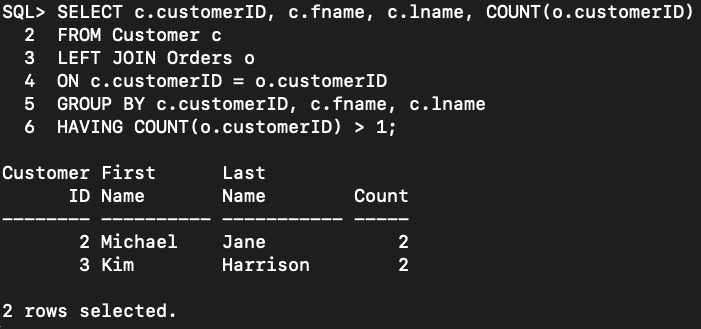


**Most sold products: Print productID, product name, price, and the number of units sold for products that were sold more than 3 times:**

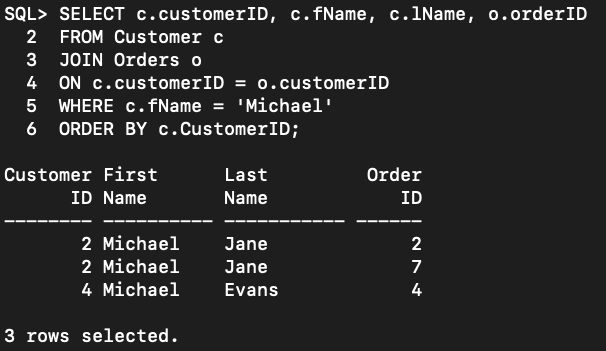


### 9.2 Queries by Joshua McNulty

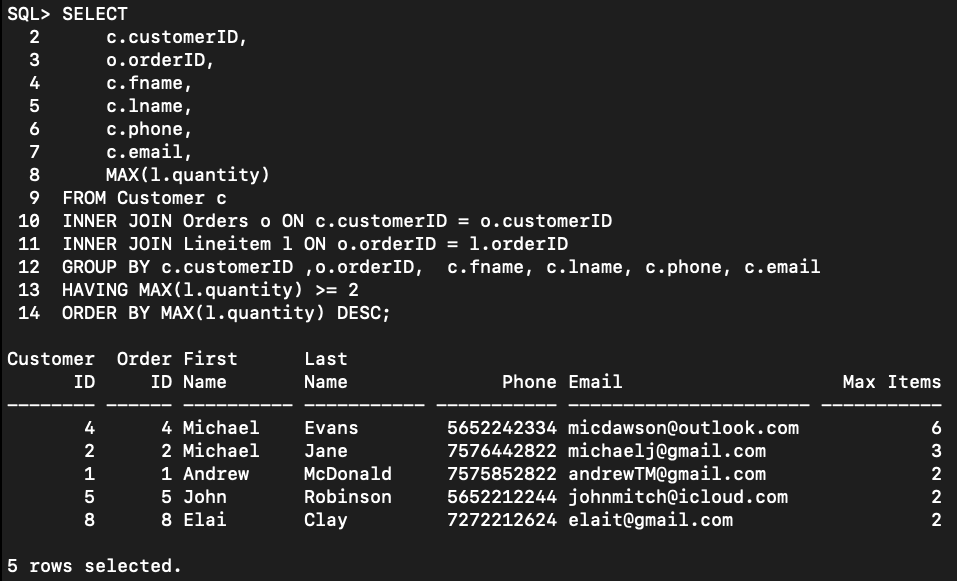
**List the number of orders for each customer with more than one order. The output values should include the count, the first name of the customer, the last name, and customerID.**



**For each customer with the first name michael, display the customerID, fname, lname, orderID, Sort by customerID**



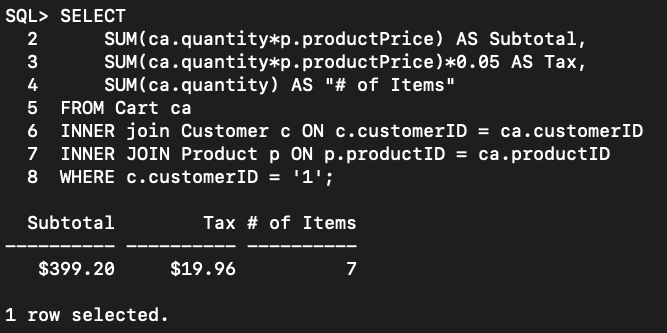
**For each customer with orders consisting of a quantity of 2 or more, output the following values: CustomerID, orderID, Customer first name, customer last name, phone, and email, Quantity**



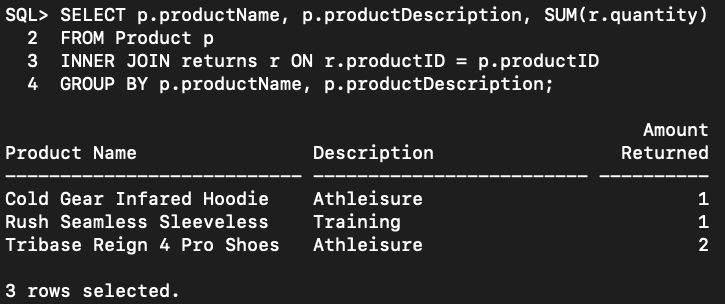
### 

### 9.3 Queries by Warren Webb

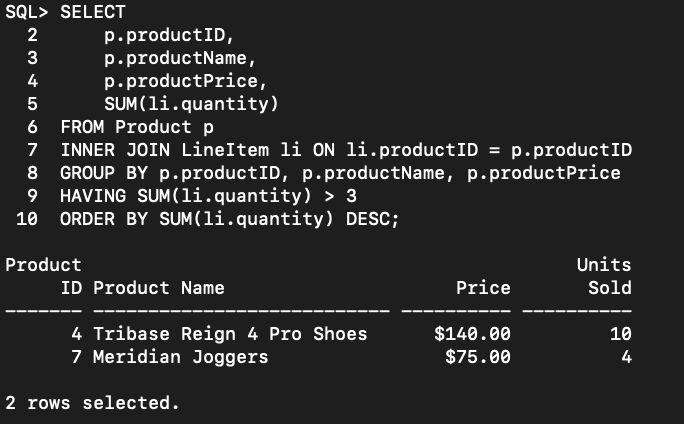
**Print subtotal, tax, and number of items in the cart of customer with customerID = 1:**



**Print product name, description, and amount returned for products that were returned:**

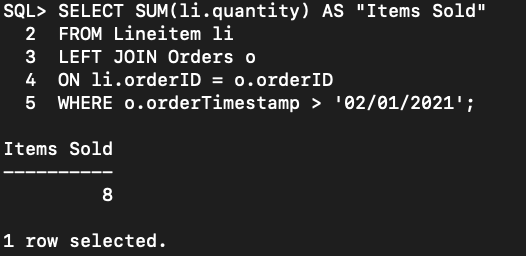


**Show the products that were sold more than 3 times: Print product ID, name, price, and number of units sold:**

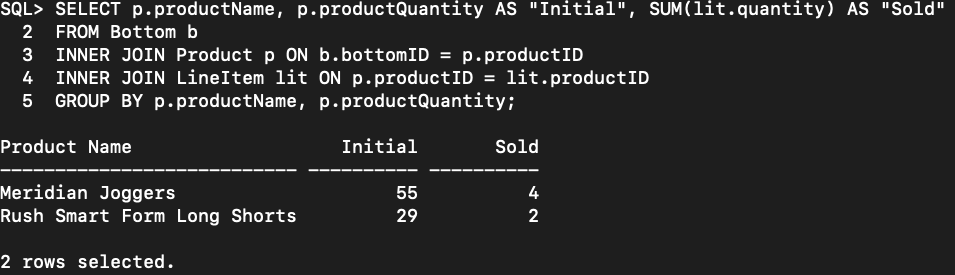


### 9.4 Queries by Trim Kerwin

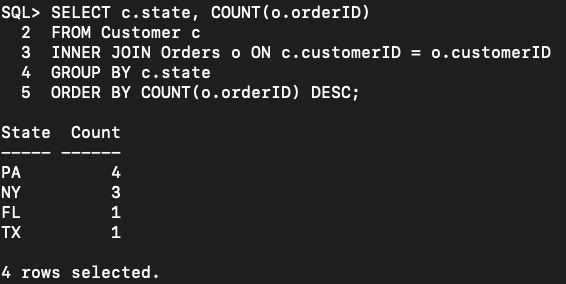
**Number of Items sold after February 1st, 2021:**



**Bottom Inventory: Print all bottoms name, the initial amount in inventory and the amount of items sold.**



**Number of orders by State: print state and the number of orders made by customers from each state. Order by the count of orders made in a descending order.**

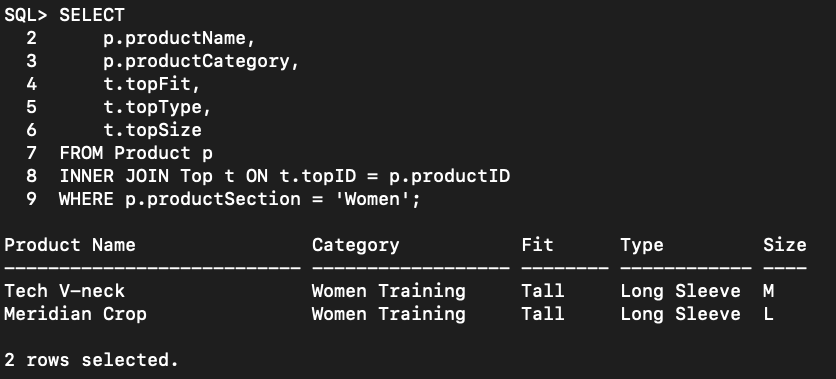


### 9.5 Queries by Ashley Wheeler

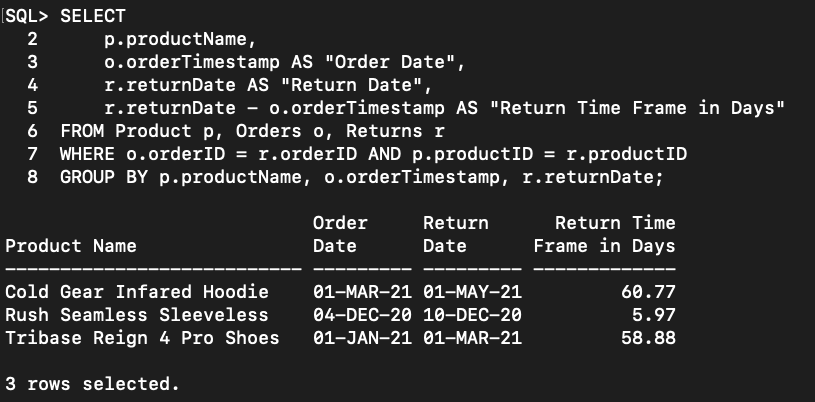
**Order details for order with orderID = 2: print product name, price, quantity, discount and total**



**Top products that are in the women section: Print product name, category, fit, type and size for every top in the women section:**



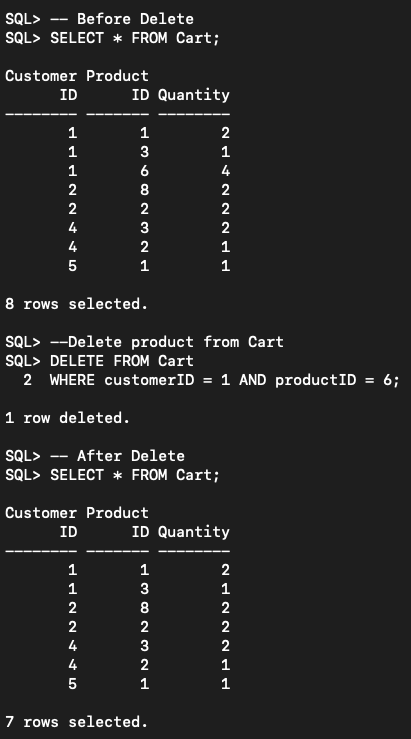
**Returned products and time till returned: Print product name , order date, return date, and the return time frame in days (return date – order date):**



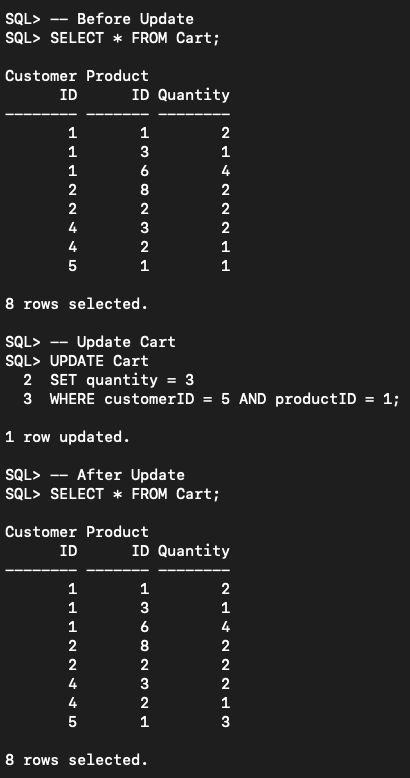
## 10. Data Manipulation (Update and Delete)

### 10.1 Queries by Aviv Farag

**Delete an item from Cart:**

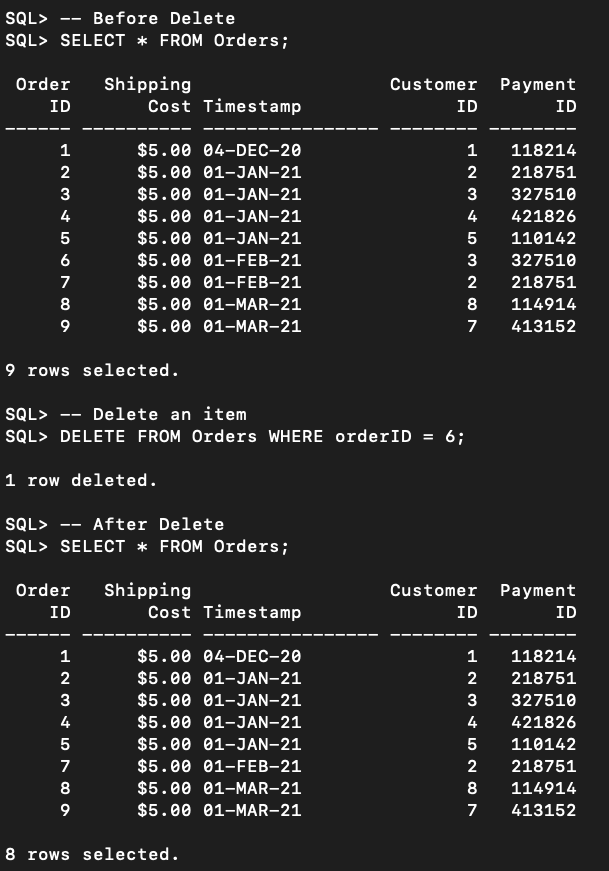


**Update quantity for a product in Cart:**

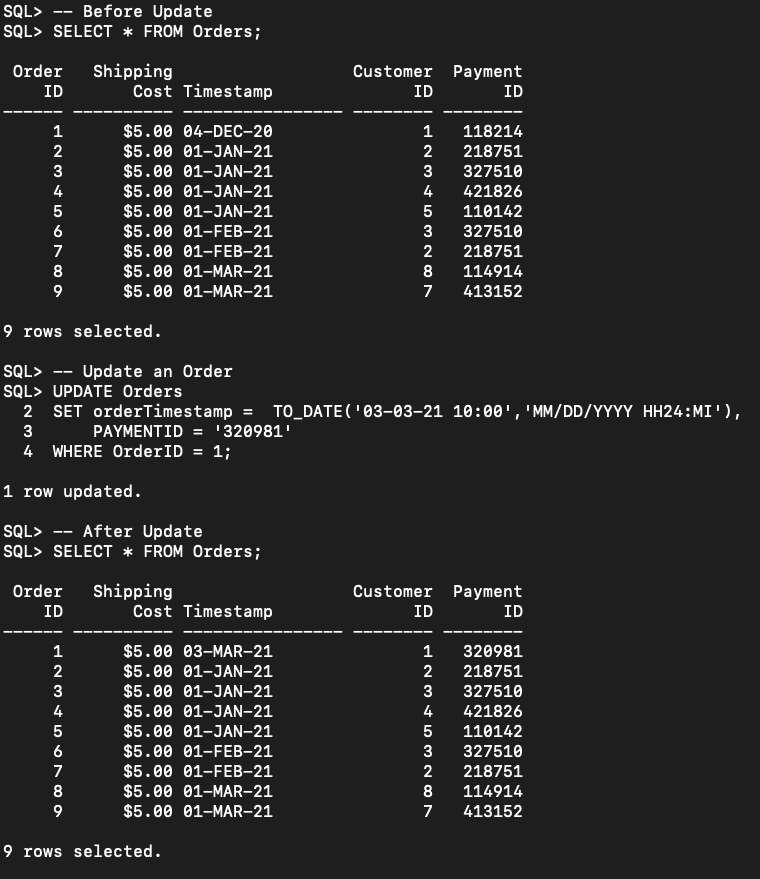


### 10.2 Queries by Joshua McNulty

**Delete an Order:**

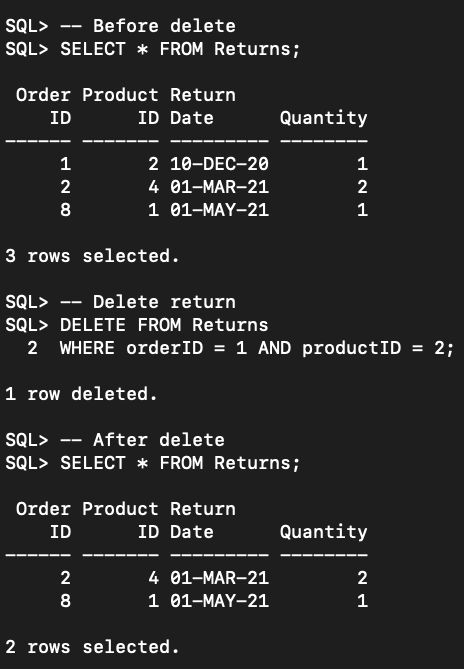


**Update an Order:**

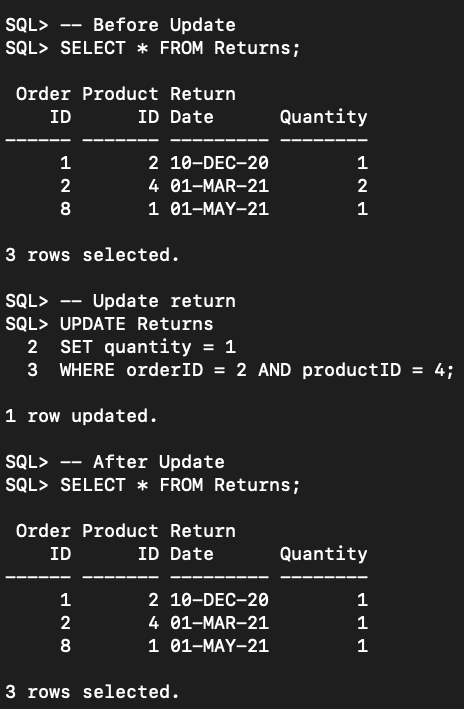


### 10.3 Queries by Warren Webb

**Delete a return:**

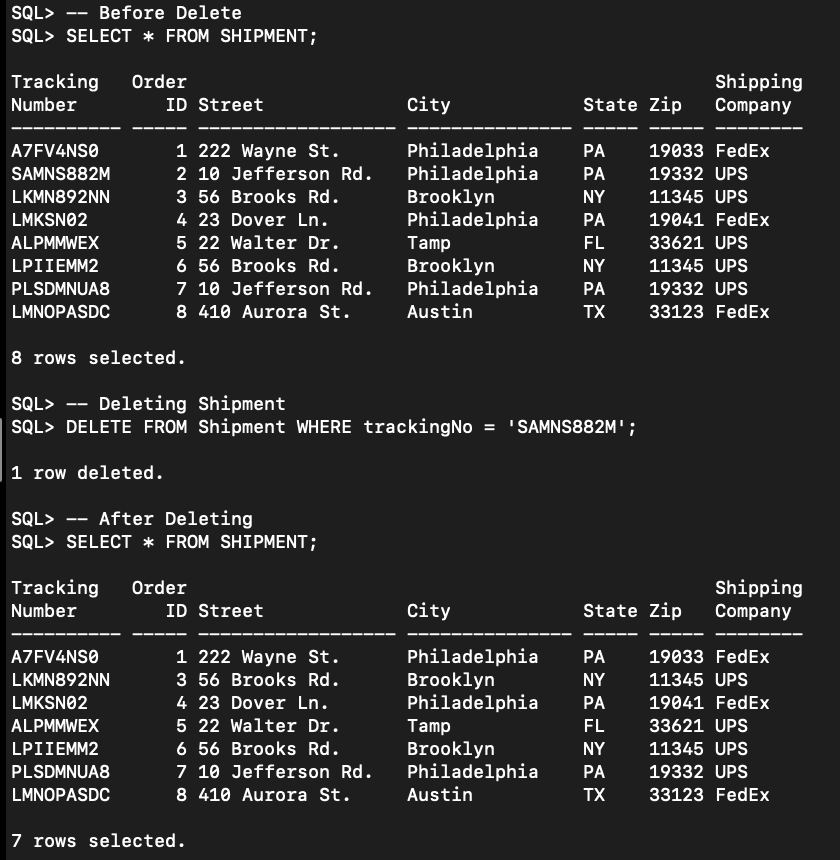


**Update returned quantity:**

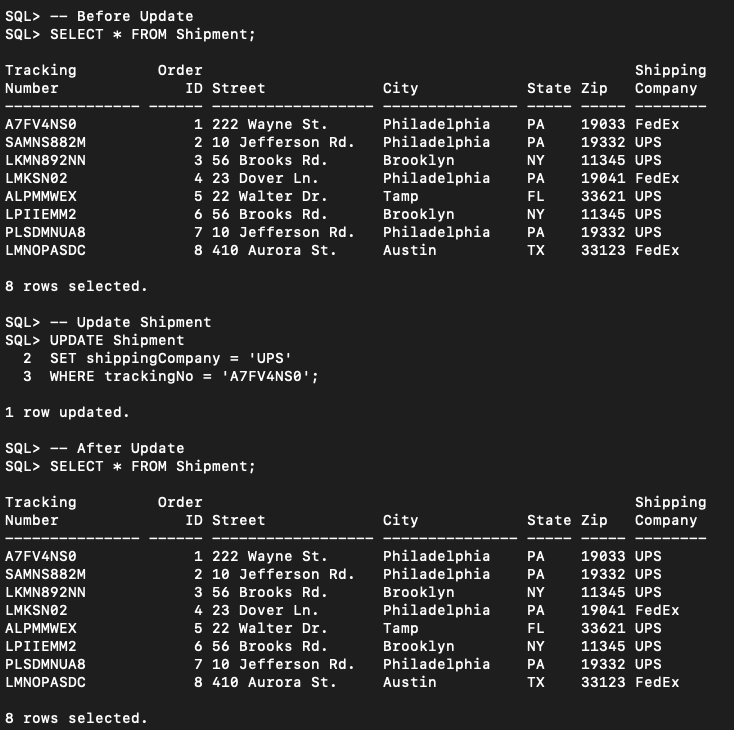


### 10.4 Queries by Trim Kerwin

**Delete a Shipment:**

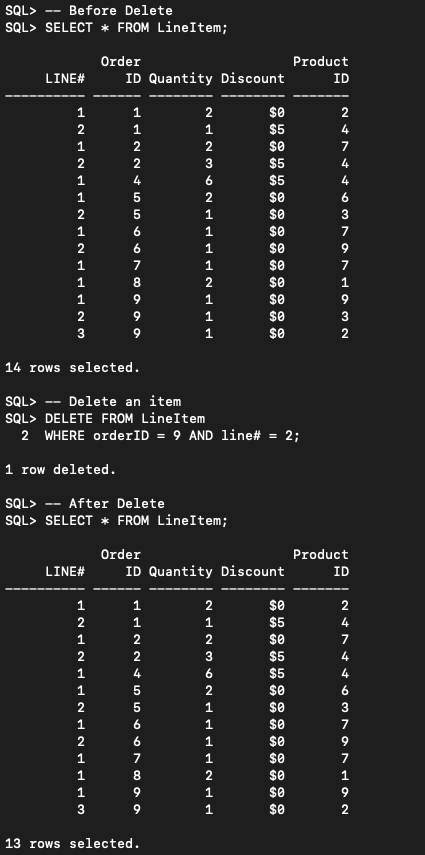


**Update a Shipment:**

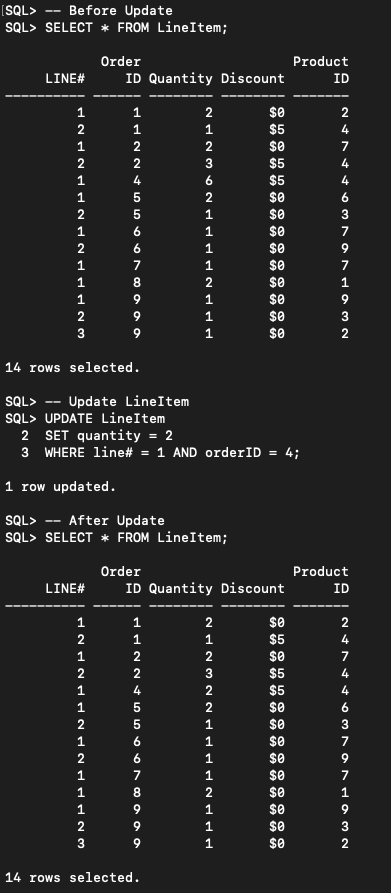


### 10.5 Queries by Ashley Wheeler

**Delete an item from LineItem:**



**Update quantity of a product in LineItem:**



## 11. Summary

### 11.1 Summary by Aviv Farag

In this project we designed a database for online retail store. We started with writing all requirements for such system and translating them into a conceptual model (ERD) using draw.io and GitHub repository to save changes. The ERD was a crucial step because the next steps are dependent on it. With that in mind, one lesson learned is to keep the ERD as simplified as possible, and the less entities the better. Translating ERD to relational schema and data dictionary is a straightforward process, and from there it is easily converted into SQL (CREATE & INSERT commands) which is the physical design. However, in this step we found many mistakes such as wrong data types, PKs, and FKs that limit the way we insert data. Those errors were all solved, but we had to go back to previous steps to refine them which reminded me of the “Waterfall process” taught at week 2. One major change we made was replacing two entities (cart & returns) with relationships in the ERD. We noticed that querying data is much easier that way, and the results are the same. The lesson learned from that experience is that not every concept must be an entity since some can be represented by a relationship which will result in a more efficient design.

### 

### 11.2 Summary by Joshua McNulty

This database project revolves around the idea of a created online retail store that sells products related to tops, bottoms, and shoes. The products all have unique identifiers and other attributes to describe their appearance. The products are chosen by the customer, and this is initiated through the relationship of the cart that is used for passive storage until the checkout process in which this would then be listed by the order after completion. The order will contain important information specific to the customer and also the items. For instance, information such as payment would be directly correlated to both the order and also the customer. There is a needed line-item entry in order to account for the output of the order consisting of attributes such as the applied discount and quantity of items within an order. Due to the nature of the store and that it is online, there is a needed shipping component compared to an on-location store with a physical shopping presence. As this business aims to please, there will be a system for returning items if there should be any cause for concern, or if the items do not please the customer. The design of this business coupled by a strongly built SQL database will result in success barring out of scope factors.

### 11.3 Summary by Warren Webb

Our project looked at making a functional database system for an online sport equipment store. The first step we looked at was writing the requirements for the project which consisted of mapping in scope items, out of scope items and future stretch goals. From there we started to build out our artifacts such as the ERD, Data dictionary and the Relational Diagram which all determines the flow and relationships of the entities that we would be creating in our database. From there we started to create the tables within SQL as well as inserting the necessary data into them. Throughout the project I have been exposed to several different software and programs such as draw.io and GitHub where we were able to collaborate and share our idea as a team. Using the ERD and other artifacts to create the database was also another great learning experience for the entire team. We faced a few challenges when it came to selecting the scope of our project in the beginning since we couldn’t decide which entities were the most important to achieving our project. After several discussion we realized that having too many entities would have made the project scope too wide and difficult to complete because our ERD would have too many entities and relationships. We eventually scaled our entity scope down to be more manageable and it saved us a lot of time going forward. In the future we can add the additional entities that we had considered such as loyalty points, discounts, etc.

### 11.4 Summary by Trim Kerwin

The goal of this project was to develop a functional system that grants users an intuitive online shopping experience. To achieve this, we created a database which encompasses the fundamental aspects of an e-commerce store. This database included customer information, order information, payment information and product information in the categories of shoes, top and bottom. With more time, this project’s shipping and return option would have been expanded for the customer’s convenience to include locations geared towards this service. The selection of products would have also be expanded to include hats and other items. With each passing week, I have learned a great deal about the fundamentals of database management as well as improving my knowledge of other platforms such as [draw.io](http://draw.io/) and GitHub for collaborative purposes. Due to the model of database development process, my team and I were able to design an ER model of the project after researching the requirements of the users. This ER model was then later translated to a relational model and then finally converted to a well-built SQL database.

### 11.5 Summary by Ashley Wheeler

This project consisted of the creation of a sports retail management store. Our goal was to design a database that could be used across multiple online stores since it can easily be updated to contain different products. Before being able to create the database we designed requirements that the database needed to obey to perform in the manner we expected. When designing requirements, a powerful tool that we learned about in class was the use of creating Entity-Relationship Diagrams (ERD) Model. We used draw.io to design our diagram, this is very helpful for when we are developing the database. When creating the physical database, we ran into some walls that allowed us to learn from experience how to update our design so that we can then implement it in our design. One of the bigger “walls” we had to deal with was deciding what should be entities versus what should be relationships both at the point of developing the ERD and when developing the database. Running into these errors when establishing the physical design allowed us to have a conversation as a group and make decisions on how to move forward. This project has not only taught me individual skills, but also personal skills when relating to others and creating a database. Future work on this project could include the addition of more items, adding discounts, creating images that show the product being sold, and payment security.

## 11. Appendix – Division of work

This group has worked cohesively on this project from the initial steps of the design and ideas behind the online retail store to the completion of the relational schema, data dictionary, and SQL commands. The individuals within the group each completed their own rendition of queries, data manipulation, and summaries separate from the others, but all other work is to be divisible in equal parts. Overall, the work was competently spread throughout the group members and communication was frequent in order to achieve completion.

## References

1. *The United States Census Bureau,* [*2019 E-Stats Report: Measuring the Electronic Economy*](https://www.census.gov/newsroom/press-releases/2021/e-estats-report-electronic-economy.html)*, August 05, 2021.*