CSE 3241 Project Checkpoint 03 – SQL and More SQL

You will be submitting several nicely formatted files for this checkpoint. Provide the following:

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 02. If you were instructed to change the model for Project Checkpoint 02, make sure you use the revised versions of your models

2. Given your relational schema, create a text file containing the SQL code to create your database schema. Use this SQL to create a database in SQLite. Populate this database with the data provided for the project as well as 20 sample records for each table that does not contain data provided in the original project documents

```
CREATE TABLE ACCOUNT
(Account_number INT NOT NULL,
 Username VARCHAR(15) NOT NULL,
 Type VARCHAR(6) NOT NULL,
 Address VARCHAR(50) NOT NULL,
 Karma_points INT,
 Phone_number CHAR(10) NOT NULL,
 Transaction_history VARCHAR(30) NOT NULL,
 Name VARCHAR(30) NOT NULL,
 PRIMARY KEY(Account_number));

 CREATE TABLE BUYER
 (Account_number INT NOT NULL,
  PRIMARY KEY(Account_number));

  CREATE TABLE SELLER
  (Account_number INT NOT NULL,
   PRIMARY KEY(Account_number));

CREATE TABLE VIRTUAL_STOREFRONT
    (StoreID INT NOT NULL,
     Account_number INT NOT NULL,
     Name VARCHAR(15) NOT NULL,
     PRIMARY KEY (StoreID)
    FOREIGN KEY (Account_number) REFERENCES SELLER(Account_number));

    CREATE TABLE PAYMENT_METHODS
    (Method_name VARCHAR(15) NOT NULL,
     StoreID INT NOT NULL,
     FOREIGN KEY (StoreID) REFERENCES VIRTUAL_STOREFRONT(StoreID),
     PRIMARY KEY (Method_name, StoreID));
```

```
CREATE TABLE PAYMENT
(PaymentID INT NOT NULL,
Type_of_Payment VARCHAR(10) NOT NULL,
Payment_Account_Number INT NOT NULL,
ExpDate DATE NOT NULL,
Order_Number INT NOT NULL,
Account_number INT NOT NULL,
FOREIGN KEY (Order_Number) REFERENCES ORDERS(Order_Number),
FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
PRIMARY KEY(PaymentID));

INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (1, "Credit", 123456, '2020/10/27', 1, 1);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (2, "Debit", 113456, '2020/10/27', 2, 2);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (3, "Credit", 111456, '2020/10/27', 3, 3);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (4, "Debit", 111156, '2020/10/27', 4, 4);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (5, "Credit", 111116, '2020/10/27', 5, 5);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (6, "Debit", 111111, '2020/10/27', 6, 6);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (7, "Credit", 223456, '2020/10/27', 7, 7);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (8, "Debit", 222456, '2020/10/27', 8, 8);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (9, "Credit", 222256, '2020/10/27', 9, 9);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (10, "Debit", 222226, '2020/10/27', 10,
10);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (11, "Credit", 222222, '2020/10/27', 11,
21);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (12, "Debit", 333456, '2020/10/27', 12,
22);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (13, "Credit", 333356, '2020/10/27', 13,
23);
```

```
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (14, "Debit", 333336, '2020/10/27', 14,
24);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (15, "Credit", 333333, '2020/10/27', 15,
25);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (16, "Debit", 444456, '2020/10/27', 16,
26);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (17, "Credit", 444446, '2020/10/27', 17,
27);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (18, "Debit", 444444, '2020/10/27', 18,
28);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (19, "Credit", 555556, '2020/10/27', 19,
29);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (20, "Debit", 555555, '2020/10/27', 20,
30);


CREATE TABLE ORDERS
(Order_Number INT NOT NULL,
Order_Date DATE NOT NULL,
Account_number INT NOT NULL,
CartID INT NOT NULL,
FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
FOREIGN KEY (CartID) REFERENCES SHOPPING_CART(CartID),
PRIMARY KEY(Order_Number));
INSERT INTO ORDERS VALUES (1, "2020-01-10", 1, 1), (2, "2020-01-31", 2, 2), (3,
"2020-02-01", 3, 3), (4, "2020-02-10", 4, 4), (5, "2020-02-27", 5, 5), (6, "2020-03-10", 6,6), (7,
"2020-04-01", 7, 7), (8, "2020-04-02", 8, 8), (9, "2020-04-03", 9, 9), (10, "2020-05-01", 10, 10),
(11, "2020-05-28", 21, 11), (12, "2020-05-29", 22, 12), (13, "2020-06-01", 23, 13), (14,
"2020-06-15", 24, 14), (15, "2020-06-17", 25, 15), (16, "2020-07-02", 26, 16), (17, "2020-07-08",
27, 17), (18, "2020-08-01", 28, 18), (19, "2020-08-03", 29, 19), (20, "2020-10-07", 30, 20);


CREATE TABLE SHOPPING_CART
(CartID INT NOT NULL,
Purchased BOOLEAN NOT NULL,
Account_number INT NOT NULL,
FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
```

```
PRIMARY KEY(CartID));
INSERT INTO SHOPPING_CART VALUES (1, TRUE, 1), (2, TRUE, 2), (3, TRUE, 3), (4, TRUE,
4), (5, TRUE, 5), (6, TRUE, 6), (7, TRUE, 7), (8, TRUE, 8), (9, TRUE, 9), (10, TRUE, 10), (11,
TRUE, 21), (12, TRUE, 22), (13, TRUE, 23), (14, TRUE, 24), (15, TRUE, 25), (16, TRUE, 26),
(17, TRUE, 27), (18, TRUE, 28), (19, TRUE, 29), (20, TRUE, 30), (21, FALSE, 1), (22, FALSE,
2), (23, FALSE, 3), (24, FALSE, 4), (25, FALSE, 5), (26, FALSE, 6), (27, FALSE, 7), (28, FALSE,
8), (29, FALSE, 9), (30, FALSE, 10), (31, FALSE, 21), (32, FALSE, 22), (33, FALSE, 23), (34,
FALSE, 24), (35, FALSE, 25), (36, FALSE, 26), (37, FALSE, 27), (38, FALSE, 28), (39, FALSE,
29), (40, FALSE, 30);



CREATE TABLE WISHLIST
(WishID INT NOT NULL,
NumProducts INT NOT NULL,
Account_number INT NOT NULL,
FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
PRIMARY KEY(WishID));



  CREATE TABLE PRODUCT
  (ProductID INT NOT NULL,
   StoreID INT NOT NULL,
   Name VARCHAR(15) NOT NULL,
   Buyer_feedback VARCHAR(100),
   Quantity INT,
   Availability VARCHAR(15) NOT NULL,
   Price DECIMAL(15, 2) NOT NULL CHECK(Price > 0),
  PRIMARY KEY (ProductID),
  FOREIGN KEY (StoreID) REFERENCES Virtual_storefront(StoreID));

   CREATE TABLE IMAGE
   (ImageID INT NOT NULL,
    Creation_date DATE NOT NULL,
    Link VARCHAR(60) NOT NULL,
    PRIMARY KEY (ImageID));


    CREATE TABLE WISH_PRODUCT
   (WishID INT NOT NULL,
    ProductID INT NOT NULL,
Quantity INT NOT NULL,
    FOREIGN KEY (WishID) REFERENCES WISHLIST(WishID),
```

```
    FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
    PRIMARY KEY (WishID, ProductID));
INSERT INTO WISH_PRODUCT(WishID,ProductID) VALUES (1, 1), (2, 2), (3, 3), (4, 4), (5, 5),
(6, 6), (7, 7), (8, 8), (9, 9), (10, 10), (11, 11), (12, 12), (13, 13), (14, 14), (15, 15), (16, 16), (17,
17), (18, 18), (19, 19), (20, 20);

CREATE TABLE SHOP_PRODUCT
(CartID INT NOT NULL,
ProductID INT NOT NULL,
Quantity INT NOT NULL,
FOREIGN KEY (CartID) REFERENCES SHOPPING_CART(CartID),
FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
PRIMARY KEY (CartID, ProductID));

INSERT INTO SHOP_PRODUCT(CartID, ProductID, Quantity) VALUES (1, 1, 1), (2, 2, 2), (3, 3,
3), (4, 4, 4), (5, 5, 5), (6, 6, 6), (7, 7, 1), (8, 8, 2), (9, 9, 3), (10, 10, 4), (11, 11, 1), (12, 12, 2), (13,
13, 3), (14, 14, 4), (15, 15, 5), (16, 16, 1), (17, 17, 2), (18, 18, 3), (19, 19, 4), (20, 20, 5);

CREATE TABLE PRODUCT_IMAGE
(ImageID INT NOT NULL,
ProductID INT NOT NULL,
FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
PRIMARY KEY (ProductID));
```

3. Given your relational schema, provide the SQL to perform the following queries. If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. These queries should be provided in a plain text file named “SimpleQueries.txt”:

a. Find the titles of all IP Items by a given Seller that cost less than $10 (you choose how to designate the seller)

```
SELECT Product.Name
FROM Product, Virtual_Storefront, SELLER
WHERE SELLER.Account_number=16 AND Price < 10 AND PRODUCT.StoreID =
VIRTUAL_STOREFRONT.StoreID AND SELLER.Account_number =
VIRTUAL_STOREFRONT.Account_number
```

b. Give all the titles and their dates of purchase made by given buyer (you choose how to designate the buyer)

```
SELECT product.Name, ORDERS.Order_Date
```

```
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE BUYER.Account_number = 1 AND BUYER.Account_number =
ORDERS.Account_number AND ORDERS.CartID = SHOPPING_CART.CartID AND
SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND PRODUCT.ProductID =
SHOP_PRODUCT.ProductID
```

c. Find the seller names for all sellers with less than 5 IP Items for sale

```
SELECT ACCOUNT.Name
FROM Seller, Virtual_Storefront, Product, Account
WHERE Seller.Account_number = Virtual_Storefront.Account_number AND Product.StoreID =
Virtual_Storefront.StoreID AND ACCOUNT.Account_number = SELLER.Account_number
GROUP BY Seller.Account_number
HAVING SUM(PRODUCT.Quantity) < 5;
```

d. Give all the buyers who purchased a IP Item by a given seller and the names of the IP Items they purchased

```
SELECT ACCOUNT.Name, PRODUCT.Name
FROM ACCOUNT, PRODUCT, BUYER, SELLER, VIRTUAL_STOREFRONT,
SHOPPING_CART, SHOP_PRODUCT, ORDERS
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
SHOP_PRODUCT.ProductID = PRODUCT.ProductID AND PRODUCT.StoreID =
VIRTUAL_STOREFRONT.StoreID AND VIRTUAL_STOREFRONT.Account_number =
SELLER.Account_number AND SELLER.Account_number = 11
```

e. Find the total number of IP Items purchased by a single buyer (you choose how to designate the buyer)

```
SELECT SUM(SHOP_PRODUCT.Quantity)
FROM (((BUYER NATURAL JOIN ORDERS) NATURAL JOIN SHOPPING_CART) NATURAL
JOIN SHOP_PRODUCT)
WHERE Account_Number = 1 AND SHOPPING_CART.purchased = TRUE
```

f. Find the buyer who has purchased the most IP Items and the total number of IP Items they have purchased

```
SELECT BUYER.Account_number, SUM(SHOP_PRODUCT.QUANTITY) AS Total
FROM (((Buyer NATURAL JOIN ORDERS) NATURAL JOIN Shopping_Cart) NATURAL JOIN
SHOP_PRODUCT)
```

```
WHERE SHOPPING_CART.purchased = TRUE
GROUP BY Account_number
ORDER BY Total LIMIT(1)
```

4. For Project Checkpoint 02, you were asked to come up with three additional interesting queries that your database can provide. Provide the SQL to perform those queries. Your queries should include at least one of these:

a. outer joins
Find the buyers who don't have a wishlist.

```
SELECT Buyer.Account_Number
FROM (Buyer LEFT JOIN Wishlist ON BUYER.Account_number =
WISHLIST.Account_Number)
WHERE WishID IS NULL
```

b. aggregate function (min, max, average, etc)
Total amount of money paid by a given buyer.

```
SELECT SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) AS total
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE BUYER.Account_number = 1 AND BUYER.Account_number =
ORDERS.Account_number AND ORDERS.CartID = SHOPPING_CART.CartID AND
SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND PRODUCT.ProductID =
SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased = TRUE
GROUP BY BUYER.Account_Number
```

c. "extra" entities from CP01
Find the buyer who has the most IP products in the wishlist.

```
SELECT BUYER.Account_number, SUM(WISH_PRODUCT.QUANTITY) AS Total
FROM ((Buyer NATURAL JOIN WishList) NATURAL JOIN WISH_PRODUCT)
GROUP BY Account_number
ORDER BY Total DESC LIMIT(1)
```

If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. These queries should be provided in a plain text file named "ExtraQueries.txt":

5. Given your relational schema, provide the SQL for the following more advanced queries. These queries may require you to use techniques such as nesting, aggregation using having clauses, and other SQL techniques .

If your database schema does not contain the information to answer these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries.

Note that if your database does contain the information but in non-aggregated form, you should NOT revise your model but instead figure out how to aggregate it for the query!

These queries should be provided in a plain text file named “AdvancedQueries.txt”.
a. Provide a list of buyer names, along with the total dollar amount each buyer has spent.

```
SELECT ACCOUNT.Name, SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) AS totalCost
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY BUYER.Account_Number
```

b. Provide a list of buyer names and e-mail addresses for buyers who have spent more than the average buyer.

```
SELECT ACCOUNT.Name, ACCOUNT.Username
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY BUYER.Account_Number
HAVING SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) > (SELECT AVG(totalCost)
FROM (SELECT SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) AS totalCost
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY BUYER.Account_Number))
```

c. Provide a list of the IP Item names and associated total copies sold to all buyers, sorted from the IP Item that has sold the most individual copies to the IP Item that has sold the least.

```
SELECT PRODUCT.Name, SUM(SHOP_PRODUCT.QUANTITY) AS Total
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY PRODUCT.Name
ORDER BY Total DESC
```

d. Provide a list of the IP Item names and associated dollar totals for copies sold to all buyers, sorted from the IP Item that has sold the highest dollar amount to the IP Item that has sold the smallest.

```
SELECT PRODUCT.Name, SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) AS Total
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY PRODUCT.Name
ORDER BY Total DESC
```

e. Find the most popular Seller (i.e. the one who has sold the most IP Items)

```
SELECT SELLER.Account_number, SUM(SHOP_PRODUCT.QUANTITY) AS Total
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT, SELLER,
VIRTUAL_STOREFRONT
WHERE PRODUCT.StoreID = VIRTUAL_STOREFRONT.StoreID AND
SELLER.Account_number = VIRTUAL_STOREFRONT.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY SELLER.Account_number
ORDER BY Total DESC LIMIT(1)
```

f. Find the most profitable seller (i.e. the one who has brought in the most money)

```
 SELECT SELLER.Account_number, SUM(SHOP_PRODUCT.QUANTITY * PRODUCT.Price)
AS Total
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT, SELLER,
VIRTUAL_STOREFRONT
WHERE PRODUCT.StoreID = VIRTUAL_STOREFRONT.StoreID AND
SELLER.Account_number = VIRTUAL_STOREFRONT.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY SELLER.Account_number
ORDER BY Total DESC LIMIT(1)
```

g. Provide a list of buyer names for buyers who purchased anything listed by the most profitable Seller.

```
SELECT ACCOUNT.Name
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT,
VIRTUAL_STOREFRONT, SELLER
WHERE PRODUCT.StoreID = VIRTUAL_STOREFRONT.StoreID AND
SELLER.Account_number = VIRTUAL_STOREFRONT.Account_number AND
ACCOUNT.Account_number = BUYER.Account_number AND BUYER.Account_number =
ORDERS.Account_number
AND ORDERS.CartID = SHOPPING_CART.CartID AND SHOPPING_CART.CartID =
SHOP_PRODUCT.CartID AND PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND
SHOPPING_CART.purchased = TRUE
AND SELLER.Account_number = ( SELECT SELLER.Account_number
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT, SELLER,
VIRTUAL_STOREFRONT
WHERE PRODUCT.StoreID = VIRTUAL_STOREFRONT.StoreID AND
SELLER.Account_number = VIRTUAL_STOREFRONT.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY SELLER.Account_number
ORDER BY SUM(SHOP_PRODUCT.QUANTITY * PRODUCT.Price) DESC LIMIT(1))
```

h. Provide the list of sellers who listed the IP Items purchased by the buyers who have spent more than the average buyer.

```
SELECT SELLER.Account_number
FROM SELLER, SHOP_PRODUCT, SHOPPING_CART, VIRTUAL_STOREFRONT, PRODUCT
```

```
WHERE PRODUCT.StoreID = VIRTUAL_STOREFRONT.StoreID AND
SELLER.Account_number = VIRTUAL_STOREFRONT.Account_number AND
SHOP_PRODUCT.CartID = SHOPPING_CART.CartID AND
SHOPPING_CART.Account_number = (
SELECT ACCOUNT.Account_number
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY BUYER.Account_Number
HAVING SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) > (SELECT AVG(totalCost)
FROM (SELECT SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) AS totalCost
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY BUYER.Account_Number)))
GROUP BY SELLER.Account_number
```

6. Once you have completed all of the questions for Part Two, create a ZIP archive containing the binary SQLite file and the three text files and submit this to the Carmen Dropbox.

Make sure your queries work against your database and provide your expected output before you submit them!

```
--
-- File generated with SQLiteStudio v3.2.1 on Wed Oct 28 15:54:16 2020
--
-- Text encoding used: System
--
PRAGMA foreign_keys = off;
BEGIN TRANSACTION;

-- Table: ACCOUNT
CREATE TABLE ACCOUNT (Account_number INT NOT NULL, Username VARCHAR (15)
NOT NULL, Type VARCHAR (6) NOT NULL, Address VARCHAR (50) NOT NULL,
Karma_points INT, Phone_number CHAR (10) NOT NULL, Name VARCHAR (30) NOT NULL,
PRIMARY KEY (Account_number));
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (1, 'kadiewheatley@gmail.com', 'Buyer', '40 Fremont Street
Vicksburg, MS 39180', 18, '939-261-5642', 'Kadie Wheatley');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (2, 'fredschmitt@gmail.com', 'Buyer', '7676 Halifax St. Lake In
The Hills, IL 60156', 98, '760-397-7044', 'Fred Schmitt');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (3, 'eilishnava@gmail.com', 'Buyer', '9120 Glenridge Lane
Petersburg, VA 23803', 13, '773-320-6658', 'Eilish Nava');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (4, 'mikeyhsingleton@gmail.com', 'Buyer', '2 Heritage Street
Saugus, MA 01906', 78, '419-689-3490', 'Mikeyh Singleton');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (5, 'lesliecalvert@gmail.com', 'Buyer', '337 Colonial Rd.
Norwich, CT 06360', 90, '620-247-9090', 'Leslie Calvert');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (6, 'rhiannehines@gmail.com', 'Buyer', '9463 University Lane
Savage, MN 55378', 14, '601-799-2240', 'Rhianne Hines');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (7, 'izabelsosa@gmail.com', 'Buyer', '66 Front St.
Oconomowoc, WI 53066', 4, '347-985-5162', 'Izabel Sosa');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (8, 'belindasmall@gmail.com', 'Buyer', '7146 Pine St. Long
Branch, NJ 07740', 93, '336-855-0401', 'Belinda Small');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (9, 'salahuddingillespie@gmail.com', 'Buyer', '8190 Eagle
Road Auburndale, FL 33823', 68, '323-557-7860', 'Salahuddin Gillespie');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (10, 'tannerwells@gmail.com', 'Seller', '501 Redwood St.
Cumberland, RI 02864', 64, '951-708-8529', 'Tanner Wells');
```

INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (11, 'arissafitzpatrick@gmail.com', 'Seller', '43 N. St Paul Street North Olmsted, OH 44070', 98, '432-398-8324', 'Arissa Fitzpatrick');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (12, 'jaydnneal@gmail.com', 'Seller', '74 Marshall Lane Brick, NJ 08723', 59, '631-842-2409', 'Jaydn Neal');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (13, 'rachaelfarley@gmail.com', 'Seller', '7046 Second St. Scarsdale, NY 10583', 38, '770-231-7288', 'Rachael Farley');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (14, 'kristopherterrell@gmail.com', 'Seller', '31 High Road Carmel, NY 10512', 3, '517-568-5259', 'Kristopher Terrell');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (15, 'felicityashley@gmail.com', 'Seller', '809 Bow Ridge Street Marion, NC 28752', 54, '610-580-5234', 'Felicity Ashley');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (16, 'frayabritt@gmail.com', 'Seller', '21 Gonzales Dr. Waxhaw, NC 28173', 72, '970-818-1322', 'Fraya Britt');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (17, 'augustholman@gmail.com', 'Seller', '7875 SW. Miles Street Mount Holly, NJ 08060', 83, '914-787-1998', 'August Holman');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (18, 'rivkaredmond@gmail.com', 'Seller', '99 Fulton St. Apt 7 Woonsocket, RI 02895', 13, '561-947-1042', 'Rivka Redmond');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (19, 'johncobb@gmail.com', 'Seller', '53 New Ave. Memphis, TN 38106', 24, '415-998-6437', 'John Cobb');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (20, 'alexcastaneda@gmail.com', 'Seller', '6 High Noon Dr. Bountiful, UT 84010', 81, '858-251-0363', 'Alex Castaneda');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (21, 'waqasclayton@gmail.com', 'Buyer', '29 N. Broad Drive Dacula, GA 30019', 18, '682-967-2033', 'Waqas Clayton');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (22, 'cinarmcdonald@gmail.com', 'Buyer', '642 Princess St. Hope Mills, NC 28348', 98, '560-583-7219', 'Cinar Mcdonald');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (23, 'ailaburton@gmail.com', 'Buyer', '92 Oakwood St. Staunton, VA 24401', 63, '129-955-7386', 'Aila Burton');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (24, 'breannamathis@gmail.com', 'Buyer', '920 Evergreen Avenue Mahopac, NY 10541', 35, '838-921-4748', 'Breanna Mathis');

INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (25, 'rhiannfinch@gmail.com', 'Buyer', '429 S. Academy Street East Brunswick, NJ 08816', 4, '765-654-1150', 'Rhiann Finch');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (26, 'stephaniemaddox@gmail.com', 'Buyer', '856 North San Pablo St. Shelton, CT 06484', 68, '327-882-8174', 'Stephanie Maddox');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (27, 'roberthamilton@gmail.com', 'Buyer', '9 Kingston Drive Nottingham, MD 21236', 30, '793-324-8530', 'Robert Hamilton');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (28, 'marilynharrison@gmail.com', 'Buyer', '13 Mayfair Lane Lakeland, FL 33801', 72, '757-202-4045', 'Marilyn Harrison');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (29, 'hazelsilva@gmail.com', 'Buyer', '9420 Littleton Dr. Long Branch, NJ 07740', 93, '828-612-7915', 'Hazel Silva');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (30, 'leiladawson@gmail.com', 'Seller', '964 Somerset St. Chicago, IL 60621', 64, '966-617-7444', 'Leila Dawson');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (31, 'maegankennedy@gmail.com', 'Seller', '655 Hillcrest Rd. Summerfield, FL 34491', 54, '780-277-7014', 'Maegan Kennedy');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (32, 'siennaparry@gmail.com', 'Seller', '3 Fordham Ave. Cornelius, NC 28031', 71, '819-894-7139', 'Sienna Parry');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (33, 'abusaunders@gmail.com', 'Seller', '9037 Riverside Ave. Fairfax, VA 22030', 12, '549-779-9330', 'Abu Saunders');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (34, 'zackbrennan@gmail.com', 'Seller', '9 Evergreen Dr. Colorado Springs, CO 80911', 14, '897-267-9535', 'Zack Brennan');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (35, 'jemsheppard@gmail.com', 'Seller', '7554 Amherst Dr. Butler, PA 16001', 51, '479-601-1484', 'Jem Sheppard');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (36, 'macaulayfranklin@gmail.com', 'Seller', '7668 Cambridge Street Garland, TX 75043', 61, '144-415-7448', 'Macaulay Franklin');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (37, 'thaliabeasley@gmail.com', 'Seller', '30 Honey Creek Road Beloit, WI 53511', 83, '274-272-9228', 'Thalia Beasley');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (38, 'teriplummer@gmail.com', 'Seller', '246 Sunset Street Montgomery Village, MD 20886', 30, '112-889-4268', 'Teri Plummer');

```
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (39, 'emerbryan@gmail.com', 'Seller', '572 Newcastle Drive
Parkville, MD 21234', 70, '980-661-4803', 'Emer Bryan');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (40, 'yasmeenfrank@gmail.com', 'Seller', '9562 Lexington
Lane Whitehall, PA 18052', 52, '626-452-6788', 'Yasmeen Frank');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (41, 'blakec@gmail.com', 'Buyer', '31 High Noon Dr. Memphis,
TN 38106', 500, '123-456-7899', 'Blake Charlton');

-- Table: BUYER
CREATE TABLE BUYER
 (Account_number INT NOT NULL,
  PRIMARY KEY(Account_number));
INSERT INTO BUYER (Account_number) VALUES (1);
INSERT INTO BUYER (Account_number) VALUES (2);
INSERT INTO BUYER (Account_number) VALUES (3);
INSERT INTO BUYER (Account_number) VALUES (4);
INSERT INTO BUYER (Account_number) VALUES (5);
INSERT INTO BUYER (Account_number) VALUES (6);
INSERT INTO BUYER (Account_number) VALUES (7);
INSERT INTO BUYER (Account_number) VALUES (8);
INSERT INTO BUYER (Account_number) VALUES (9);
INSERT INTO BUYER (Account_number) VALUES (10);
INSERT INTO BUYER (Account_number) VALUES (21);
INSERT INTO BUYER (Account_number) VALUES (22);
INSERT INTO BUYER (Account_number) VALUES (23);
INSERT INTO BUYER (Account_number) VALUES (24);
INSERT INTO BUYER (Account_number) VALUES (25);
INSERT INTO BUYER (Account_number) VALUES (26);
INSERT INTO BUYER (Account_number) VALUES (27);
INSERT INTO BUYER (Account_number) VALUES (28);
INSERT INTO BUYER (Account_number) VALUES (29);
INSERT INTO BUYER (Account_number) VALUES (30);
INSERT INTO BUYER (Account_number) VALUES (41);

-- Table: IMAGE
CREATE TABLE IMAGE
   (ImageID INT NOT NULL,
    Creation_date DATE NOT NULL,
    Link VARCHAR(60) NOT NULL,
    PRIMARY KEY (ImageID));
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (1, '2020/10/27',
'eszycidpyo.jpg');
```

```
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (2, '2020/10/27',
'pumzgdpamn.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (3, '2020/10/27',
'tyyawoixzh.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (4, '2020/10/27',
'sdkaaauram.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (5, '2020/10/27',
'vgnxaqhyop.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (6, '2020/10/27',
'rhlhvhyoja.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (7, '2020/10/27',
'nrudfuxjdx.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (8, '2020/10/27',
'kxwqnqvgjj.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (9, '2020/10/27',
'spqmsbphxz.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (10, '2020/10/27',
'mnvflrwyvx.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (11, '2020/10/27',
'lcovqdyfqm.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (12, '2020/10/27',
'lpxapbjwts.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (13, '2020/10/27',
'smuffqhayg.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (14, '2020/10/27',
'rrhmqlsloi.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (15, '2020/10/27',
'vrtxamzxqz.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (16, '2020/10/27',
'eqyrgnbpls.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (17, '2020/10/27',
'rgqnplnlar.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (18, '2020/10/27',
'rtztkotazh.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (19, '2020/10/27',
'ufrsfczrzi.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (20, '2020/10/27',
'bvccaoayyi.jpg');

-- Table: ORDERS
CREATE TABLE ORDERS (Order_Number INT NOT NULL, Order_Date DATE NOT NULL,
Account_number INT NOT NULL, CartID INT NOT NULL, FOREIGN KEY (Account_Number)
REFERENCES Buyer (Account_number), FOREIGN KEY (CartID) REFERENCES
SHOPPING_CART (CartID), PRIMARY KEY (Order_Number));
```

```
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (1, '2020-01-10', 1, 1);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (2, '2020-01-31', 2, 2);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (3, '2020-02-01', 3, 3);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (4, '2020-02-10', 4, 4);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (5, '2020-02-27', 5, 5);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (6, '2020-03-10', 6, 6);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (7, '2020-04-01', 7, 7);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (8, '2020-04-02', 8, 8);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (9, '2020-04-03', 9, 9);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (10, '2020-05-01', 10, 10);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (11, '2020-05-28', 21, 11);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (12, '2020-05-29', 22, 12);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (13, '2020-06-01', 23, 13);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (14, '2020-06-15', 24, 14);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (15, '2020-06-17', 25, 15);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (16, '2020-07-02', 26, 16);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (17, '2020-07-08', 27, 17);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (18, '2020-08-01', 28, 18);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (19, '2020-08-03', 29, 19);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (20, '2020-10-07', 30, 20);

-- Table: PAYMENT
CREATE TABLE PAYMENT
(PaymentID INT NOT NULL,
```

```
 Type_of_Payment VARCHAR(10) NOT NULL,
 Payment_Account_Number INT NOT NULL,
 ExpDate DATE NOT NULL,
 Order_Number INT NOT NULL,
Account_number INT NOT NULL,
 FOREIGN KEY (Order_Number) REFERENCES ORDERS(Order_Number),
 FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
 PRIMARY KEY(PaymentID));
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (1, 'Credit', 123456, '2020/10/27', 1, 1);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (2, 'Debit', 113456, '2020/10/27', 2, 2);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (3, 'Credit', 111456, '2020/10/27', 3, 3);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (4, 'Debit', 111156, '2020/10/27', 4, 4);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (5, 'Credit', 111116, '2020/10/27', 5, 5);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (6, 'Debit', 111111, '2020/10/27', 6, 6);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (7, 'Credit', 223456, '2020/10/27', 7, 7);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (8, 'Debit', 222456, '2020/10/27', 8, 8);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (9, 'Credit', 222256, '2020/10/27', 9, 9);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (10, 'Debit', 222226, '2020/10/27', 10, 10);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (11, 'Credit', 222222, '2020/10/27', 11,
21);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (12, 'Debit', 333456, '2020/10/27', 12, 22);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (13, 'Credit', 333356, '2020/10/27', 13,
23);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (14, 'Debit', 333336, '2020/10/27', 14, 24);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (15, 'Credit', 333333, '2020/10/27', 15,
25);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (16, 'Debit', 444456, '2020/10/27', 16, 26);
```

```
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (17, 'Credit', 444446, '2020/10/27', 17,
27);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (18, 'Debit', 444444, '2020/10/27', 18, 28);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (19, 'Credit', 555556, '2020/10/27', 19,
29);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (20, 'Debit', 555555, '2020/10/27', 20, 30);

-- Table: PAYMENT_METHODS
CREATE TABLE PAYMENT_METHODS
   (Method_name VARCHAR(15) NOT NULL,
    StoreID INT NOT NULL,
    FOREIGN KEY (StoreID) REFERENCES VIRTUAL_STOREFRONT(StoreID),
    PRIMARY KEY (Method_name, StoreID));
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 2);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 2);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 3);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 3);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 4);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 4);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 5);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 5);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 6);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 6);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 7);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 7);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 8);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 8);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 9);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 9);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 10);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 10);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 11);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 11);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 12);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 12);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 13);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 13);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 14);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 14);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 15);
```

```
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 15);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 16);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 16);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 17);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 17);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 18);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 18);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 19);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 19);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 20);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 20);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Karma', 1);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 1);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 1);

-- Table: PRODUCT
CREATE TABLE PRODUCT
  (ProductID INT NOT NULL,
   StoreID INT NOT NULL,
   Name VARCHAR(15) NOT NULL,
   Buyer_feedback VARCHAR(100),
   Quantity INT,
   Availability VARCHAR(15) NOT NULL,
   Price DECIMAL(15, 2) NOT NULL CHECK(Price > 0),
  PRIMARY KEY (ProductID),
  FOREIGN KEY (StoreID) REFERENCES Virtual_storefront(StoreID));
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (1, 5, 'szycidpy',
'pumzgdpamntyyawoixzhsdkaaauramvgnxaqhyoprhlhvhyojanrudfuxjdxkxwqnqv', 2, 'Available',
5);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (2, 10, 'spqmsbph',
'nvflrwyvxlcovqdyfqmlpxapbjwtssmuffqhaygrrhmqlsloivrtxamzxqzeq', 5, 'Available', 6);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (3, 14, 'bplsrgqn',
'lnlarrtztkotazhufrsfczrzibvccaoayyihidztfljcffiqfviuwjowkppdajmknzgidixq', 2, 'Available', 160);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (4, 14, 'ahamebxf',
'wqvnrhuzwqohquamvszkvunbxjegbjccjjxfnsiearbsgsofywtqbmgldgsvnsgpdvm', 3, 'Available',
134);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (5, 16, 'aktmjafg', 'zszekngivdmrlvrpyrhcxbceffrgiyktqilkkdjhtywpesrydkb', 4,
'Available', 23);
```

INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (6, 13, 'zeekdtsz', 'csrhsciljsrdoidzbjatvacndzbghzsnfdofvhfxdnmzrjriwpkdgukbaa', 3, 'Available', 3);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (7, 20, 'komkmcck', 'odigztyrwpvlifrgjghlcicyocusukhmjbkfkzsjhkdrtsztchhazhmcircxcauajyzlppedqyzkcqvffyeekj', 1, 'Available', 186);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (8, 17, 'tjegerxb', 'tzvrxwgfjnrfbwvhiycvoznriroroamkfipazunsabwlseseei', 3, 'Available', 106);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (9, 19, 'mftchpaf', 'kquovuxhhkpvphwnkrtxuiuhbcyqulfqyzgjjwjrlfwwxotcdtqsmfeingsxyzbpvmwulmqfrxbqc', 3, 'Available', 165);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (10, 4, 'ixceytvv', 'cohmznmfkoetpgdntrndvjihmxragqosaauthigfjergijsyivozzfrlpndygsmgjzdzadsxarjvyxuecqlszjnqvlyqkadowol', 3, 'Available', 143);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (11, 13, 'kzxvspdu', 'mgraiutxxxqgotqnxwjwfotvqglqavmsnmktsxwxcpxhuujuanxueuymzi', 2, 'Available', 2);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (12, 20, 'alizwnvr', 'eoipfoqbiqdxsnclcvoafqwfwcmuwitjgqghkiccwqvloqrx', 1, 'Available', 48);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (13, 10, 'uxwriltx', 'mrmfpzitkwhitwhvatmknyhzigcuxfsosxetioq', 2, 'Available', 5);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (14, 5, 'woljymhd', 'wvjcdhmkpdfbbztaygvbpwqxtokvidtwfdhm', 2, 'Available', 131);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (15, 15, 'myfhhjor', 'mgowikpsdgcbazapkmsjgmfyuezaamevrbsmiecoujabrbqebiydncgapuexivgvomkuiiuuhhbszsflntwr', 5, 'Available', 20);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (16, 12, 'rnrgwrnv', 'wixtxycifdebgnbbuc', 5, 'Available', 125);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (17, 17, 'ldkberbo', 'emywoaxqicizkcjbmbxikxeizmzdvjdnhqrgkkqzmspdeuoqrxswqrajxfglmqkdnlescbjzurknjklikxxqqaqdekxkzks', 1, 'Available', 120);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (18, 9, 'polxmcsz', 'ebqpsizhwsxklzulm', 3, 'Available', 123);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (19, 20, 'krqfaeiv', 'sedfynxtbzdrviwdgicusqucczgufqnaslpwzj', 2, 'Available', 56);

```
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (20, 20, 'phnovlrg',
'xcingaxrymqpcmtqzssnbloagjwwuardjqxkyrusrjqnrqntusjojeqoseryfiuanxvsbln', 4, 'Available',
77);

-- Table: PRODUCT_IMAGE
CREATE TABLE PRODUCT_IMAGE
  (ImageID INT NOT NULL,
   ProductID INT NOT NULL,
   FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
   PRIMARY KEY (ProductID));
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (1, 1);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (2, 2);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (3, 3);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (4, 4);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (5, 5);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (6, 6);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (7, 7);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (8, 8);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (9, 9);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (10, 10);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (11, 11);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (12, 12);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (13, 13);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (14, 14);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (15, 15);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (16, 16);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (17, 17);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (18, 18);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (19, 19);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (20, 20);

-- Table: SELLER
CREATE TABLE SELLER
  (Account_number INT NOT NULL,
   PRIMARY KEY(Account_number));
INSERT INTO SELLER (Account_number) VALUES (10);
INSERT INTO SELLER (Account_number) VALUES (11);
INSERT INTO SELLER (Account_number) VALUES (12);
INSERT INTO SELLER (Account_number) VALUES (13);
INSERT INTO SELLER (Account_number) VALUES (14);
INSERT INTO SELLER (Account_number) VALUES (15);
INSERT INTO SELLER (Account_number) VALUES (16);
INSERT INTO SELLER (Account_number) VALUES (17);
```

```
INSERT INTO SELLER (Account_number) VALUES (18);
INSERT INTO SELLER (Account_number) VALUES (19);
INSERT INTO SELLER (Account_number) VALUES (20);
INSERT INTO SELLER (Account_number) VALUES (30);
INSERT INTO SELLER (Account_number) VALUES (31);
INSERT INTO SELLER (Account_number) VALUES (32);
INSERT INTO SELLER (Account_number) VALUES (33);
INSERT INTO SELLER (Account_number) VALUES (34);
INSERT INTO SELLER (Account_number) VALUES (35);
INSERT INTO SELLER (Account_number) VALUES (36);
INSERT INTO SELLER (Account_number) VALUES (37);
INSERT INTO SELLER (Account_number) VALUES (38);
INSERT INTO SELLER (Account_number) VALUES (39);
INSERT INTO SELLER (Account_number) VALUES (40);

-- Table: SHOP_PRODUCT
CREATE TABLE SHOP_PRODUCT
(CartID INT NOT NULL,
ProductID INT NOT NULL,
Quantity INT NOT NULL,
FOREIGN KEY (CartID) REFERENCES SHOPPING_CART(CartID),
FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
PRIMARY KEY (CartID, ProductID));
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (1, 1, 4);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (2, 2, 2);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (3, 3, 3);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (4, 4, 4);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (5, 5, 5);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (6, 6, 7);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (7, 7, 1);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (8, 8, 2);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (9, 9, 3);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (10, 10, 4);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (11, 11, 1);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (12, 12, 2);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (13, 13, 3);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (14, 14, 4);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (15, 15, 5);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (16, 16, 1);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (17, 17, 2);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (18, 18, 3);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (19, 19, 4);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (20, 20, 1);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (1, 2, 300);
```

```
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (2, 3, 1);

-- Table: SHOPPING_CART
CREATE TABLE SHOPPING_CART (CartID INT NOT NULL, Purchased BOOLEAN NOT
NULL, Account_number INT NOT NULL, FOREIGN KEY (Account_Number) REFERENCES
Buyer (Account_number), PRIMARY KEY (CartID));
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (1, 1, 1);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (2, 1, 2);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (3, 1, 3);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (4, 1, 4);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (5, 1, 5);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (6, 1, 6);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (7, 1, 7);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (8, 1, 8);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (9, 1, 9);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (10, 1, 10);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (11, 1, 21);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (12, 1, 22);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (13, 1, 23);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (14, 1, 24);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (15, 1, 25);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (16, 1, 26);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (17, 1, 27);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (18, 1, 28);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (19, 1, 29);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (20, 1, 30);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (21, 0, 1);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (22, 0, 2);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (23, 0, 3);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (24, 0, 4);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (25, 0, 5);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (26, 0, 6);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (27, 0, 7);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (28, 0, 8);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (29, 0, 9);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (30, 0, 10);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (31, 0, 21);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (32, 0, 22);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (33, 0, 23);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (34, 0, 24);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (35, 0, 25);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (36, 0, 26);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (37, 0, 27);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (38, 0, 28);
```

```
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (39, 0, 29);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (40, 0, 30);

-- Table: VIRTUAL_STOREFRONT
CREATE TABLE VIRTUAL_STOREFRONT
    (StoreID INT NOT NULL,
     Account_number INT NOT NULL,
     Name VARCHAR(15) NOT NULL,
     PRIMARY KEY (StoreID)
    FOREIGN KEY (Account_number) REFERENCES SELLER(Account_number));
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (1, 12,
'Jaydn Neal');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (2, 19,
'John Cobb');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (3, 11,
'Arissa Fitzpatrick');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (4, 14,
'Kristopher Terrell');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (5, 11,
'Arissa Fitzpatrick');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (6, 17,
'August Holman');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (7, 17,
'August Holman');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (8, 17,
'August Holman');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (9, 20,
'Alex Castaneda');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (10, 16,
'Fraya Britt');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (11, 13,
'Rachael Farley');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (12, 11,
'Arissa Fitzpatrick');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (13, 17,
'August Holman');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (14, 10,
'Tanner Wells');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (15, 16,
'Fraya Britt');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (16, 16,
'Fraya Britt');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (17, 19,
'John Cobb');
```

```
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (18, 10,
'Tanner Wells');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (19, 17,
'August Holman');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (20, 14,
'Kristopher Terrell');

-- Table: WISH_PRODUCT
CREATE TABLE WISH_PRODUCT
  (WishID INT NOT NULL,
   ProductID INT NOT NULL,
Quantity INT NOT NULL,
   FOREIGN KEY (WishID) REFERENCES WISHLIST(WishID),
   FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
   PRIMARY KEY (WishID, ProductID));
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (2, 2, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (3, 3, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (4, 4, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (5, 5, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (6, 6, 3);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (7, 7, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (8, 8, 3);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (9, 9, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (8, 9, 3);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (11, 11, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (1, 1, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (1, 2, 3);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (1, 3, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (1, 7, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (2, 3, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (3, 4, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (4, 5, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (5, 6, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (6, 7, 3);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (7, 8, 1);

-- Table: WISHLIST
CREATE TABLE "WISHLIST"
(WishID INT NOT NULL,
NumProducts INT NOT NULL,
Account_number INT NOT NULL,
FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
PRIMARY KEY(WishID));
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (16, 8, 26);
```

```
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (8, 8, 8);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (20, 9, 30);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (4, 9, 4);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (19, 5, 29);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (11, 5, 21);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (10, 9, 10);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (1, 4, 1);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (17, 5, 27);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (5, 8, 5);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (3, 3, 3);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (6, 3, 6);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (9, 2, 9);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (14, 2, 24);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (18, 2, 28);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (12, 3, 22);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (2, 3, 2);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (7, 2, 7);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (15, 10, 25);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (13, 2, 23);

COMMIT TRANSACTION;
PRAGMA foreign_keys = on;
```

```
--
-- File generated with SQLiteStudio v3.2.1 on Thu Nov 12 13:54:44 2020
--
-- Text encoding used: System
--
PRAGMA foreign_keys = off;
BEGIN TRANSACTION;

-- Table: ACCOUNT
CREATE TABLE ACCOUNT (Account_number INT NOT NULL, Username VARCHAR (15)
NOT NULL, Type VARCHAR (6) NOT NULL, Address VARCHAR (50) NOT NULL,
Karma_points INT, Phone_number CHAR (10) NOT NULL, Name VARCHAR (30) NOT NULL,
PRIMARY KEY (Account_number));
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (1, 'kadiewheatley@gmail.com', 'Buyer', '40 Fremont Street
Vicksburg, MS 39180', 18, '939-261-5642', 'Kadie Wheatley');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (2, 'fredschmitt@gmail.com', 'Buyer', '7676 Halifax St. Lake In
The Hills, IL 60156', 98, '760-397-7044', 'Fred Schmitt');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (3, 'eilishnava@gmail.com', 'Buyer', '9120 Glenridge Lane
Petersburg, VA 23803', 13, '773-320-6658', 'Eilish Nava');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (4, 'mikeyhsingleton@gmail.com', 'Buyer', '2 Heritage Street
Saugus, MA 01906', 78, '419-689-3490', 'Mikeyh Singleton');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (5, 'lesliecalvert@gmail.com', 'Buyer', '337 Colonial Rd.
Norwich, CT 06360', 90, '620-247-9090', 'Leslie Calvert');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (6, 'rhiannehines@gmail.com', 'Buyer', '9463 University Lane
Savage, MN 55378', 14, '601-799-2240', 'Rhianne Hines');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (7, 'izabelsosa@gmail.com', 'Buyer', '66 Front St.
Oconomowoc, WI 53066', 4, '347-985-5162', 'Izabel Sosa');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (8, 'belindasmall@gmail.com', 'Buyer', '7146 Pine St. Long
Branch, NJ 07740', 93, '336-855-0401', 'Belinda Small');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (9, 'salahuddingillespie@gmail.com', 'Buyer', '8190 Eagle
Road Auburndale, FL 33823', 68, '323-557-7860', 'Salahuddin Gillespie');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (10, 'tannerwells@gmail.com', 'Seller', '501 Redwood St.
Cumberland, RI 02864', 64, '951-708-8529', 'Tanner Wells');
```

INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (11, 'arissafitzpatrick@gmail.com', 'Seller', '43 N. St Paul Street North Olmsted, OH 44070', 98, '432-398-8324', 'Arissa Fitzpatrick');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (12, 'jaydnneal@gmail.com', 'Seller', '74 Marshall Lane Brick, NJ 08723', 59, '631-842-2409', 'Jaydn Neal');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (13, 'rachaelfarley@gmail.com', 'Seller', '7046 Second St. Scarsdale, NY 10583', 38, '770-231-7288', 'Rachael Farley');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (14, 'kristopherterrell@gmail.com', 'Seller', '31 High Road Carmel, NY 10512', 3, '517-568-5259', 'Kristopher Terrell');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (15, 'felicityashley@gmail.com', 'Seller', '809 Bow Ridge Street Marion, NC 28752', 54, '610-580-5234', 'Felicity Ashley');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (16, 'frayabritt@gmail.com', 'Seller', '21 Gonzales Dr. Waxhaw, NC 28173', 72, '970-818-1322', 'Fraya Britt');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (17, 'augustholman@gmail.com', 'Seller', '7875 SW. Miles Street Mount Holly, NJ 08060', 83, '914-787-1998', 'August Holman');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (18, 'rivkaredmond@gmail.com', 'Seller', '99 Fulton St. Apt 7 Woonsocket, RI 02895', 13, '561-947-1042', 'Rivka Redmond');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (19, 'johncobb@gmail.com', 'Seller', '53 New Ave. Memphis, TN 38106', 24, '415-998-6437', 'John Cobb');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (20, 'alexcastaneda@gmail.com', 'Seller', '6 High Noon Dr. Bountiful, UT 84010', 81, '858-251-0363', 'Alex Castaneda');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (21, 'waqasclayton@gmail.com', 'Buyer', '29 N. Broad Drive Dacula, GA 30019', 18, '682-967-2033', 'Waqas Clayton');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (22, 'cinarmcdonald@gmail.com', 'Buyer', '642 Princess St. Hope Mills, NC 28348', 98, '560-583-7219', 'Cinar Mcdonald');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (23, 'ailaburton@gmail.com', 'Buyer', '92 Oakwood St. Staunton, VA 24401', 63, '129-955-7386', 'Aila Burton');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (24, 'breannamathis@gmail.com', 'Buyer', '920 Evergreen Avenue Mahopac, NY 10541', 35, '838-921-4748', 'Breanna Mathis');

INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (25, 'rhiannfinch@gmail.com', 'Buyer', '429 S. Academy Street East Brunswick, NJ 08816', 4, '765-654-1150', 'Rhiann Finch');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (26, 'stephaniemaddox@gmail.com', 'Buyer', '856 North San Pablo St. Shelton, CT 06484', 68, '327-882-8174', 'Stephanie Maddox');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (27, 'roberthamilton@gmail.com', 'Buyer', '9 Kingston Drive Nottingham, MD 21236', 30, '793-324-8530', 'Robert Hamilton');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (28, 'marilynharrison@gmail.com', 'Buyer', '13 Mayfair Lane Lakeland, FL 33801', 72, '757-202-4045', 'Marilyn Harrison');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (29, 'hazelsilva@gmail.com', 'Buyer', '9420 Littleton Dr. Long Branch, NJ 07740', 93, '828-612-7915', 'Hazel Silva');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (30, 'leiladawson@gmail.com', 'Seller', '964 Somerset St. Chicago, IL 60621', 64, '966-617-7444', 'Leila Dawson');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (31, 'maegankennedy@gmail.com', 'Seller', '655 Hillcrest Rd. Summerfield, FL 34491', 54, '780-277-7014', 'Maegan Kennedy');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (32, 'siennaparry@gmail.com', 'Seller', '3 Fordham Ave. Cornelius, NC 28031', 71, '819-894-7139', 'Sienna Parry');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (33, 'abusaunders@gmail.com', 'Seller', '9037 Riverside Ave. Fairfax, VA 22030', 12, '549-779-9330', 'Abu Saunders');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (34, 'zackbrennan@gmail.com', 'Seller', '9 Evergreen Dr. Colorado Springs, CO 80911', 14, '897-267-9535', 'Zack Brennan');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (35, 'jemsheppard@gmail.com', 'Seller', '7554 Amherst Dr. Butler, PA 16001', 51, '479-601-1484', 'Jem Sheppard');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (36, 'macaulayfranklin@gmail.com', 'Seller', '7668 Cambridge Street Garland, TX 75043', 61, '144-415-7448', 'Macaulay Franklin');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (37, 'thaliabeasley@gmail.com', 'Seller', '30 Honey Creek Road Beloit, WI 53511', 83, '274-272-9228', 'Thalia Beasley');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points, Phone_number, Name) VALUES (38, 'teriplummer@gmail.com', 'Seller', '246 Sunset Street Montgomery Village, MD 20886', 30, '112-889-4268', 'Teri Plummer');

```
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (39, 'emerbryan@gmail.com', 'Seller', '572 Newcastle Drive
Parkville, MD 21234', 70, '980-661-4803', 'Emer Bryan');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (40, 'yasmeenfrank@gmail.com', 'Seller', '9562 Lexington
Lane Whitehall, PA 18052', 52, '626-452-6788', 'Yasmeen Frank');
INSERT INTO ACCOUNT (Account_number, Username, Type, Address, Karma_points,
Phone_number, Name) VALUES (41, 'blakec@gmail.com', 'Buyer', '31 High Noon Dr. Memphis,
TN 38106', 500, '123-456-7899', 'Blake Charlton');

-- Table: BUYER
CREATE TABLE BUYER
 (Account_number INT NOT NULL,
  PRIMARY KEY(Account_number));
INSERT INTO BUYER (Account_number) VALUES (1);
INSERT INTO BUYER (Account_number) VALUES (2);
INSERT INTO BUYER (Account_number) VALUES (3);
INSERT INTO BUYER (Account_number) VALUES (4);
INSERT INTO BUYER (Account_number) VALUES (5);
INSERT INTO BUYER (Account_number) VALUES (6);
INSERT INTO BUYER (Account_number) VALUES (7);
INSERT INTO BUYER (Account_number) VALUES (8);
INSERT INTO BUYER (Account_number) VALUES (9);
INSERT INTO BUYER (Account_number) VALUES (10);
INSERT INTO BUYER (Account_number) VALUES (21);
INSERT INTO BUYER (Account_number) VALUES (22);
INSERT INTO BUYER (Account_number) VALUES (23);
INSERT INTO BUYER (Account_number) VALUES (24);
INSERT INTO BUYER (Account_number) VALUES (25);
INSERT INTO BUYER (Account_number) VALUES (26);
INSERT INTO BUYER (Account_number) VALUES (27);
INSERT INTO BUYER (Account_number) VALUES (28);
INSERT INTO BUYER (Account_number) VALUES (29);
INSERT INTO BUYER (Account_number) VALUES (30);
INSERT INTO BUYER (Account_number) VALUES (41);

-- Table: IMAGE
CREATE TABLE IMAGE
   (ImageID INT NOT NULL,
    Creation_date DATE NOT NULL,
    Link VARCHAR(60) NOT NULL,
    PRIMARY KEY (ImageID));
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (1, '2020/10/27',
'eszycidpyo.jpg');
```

```
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (2, '2020/10/27',
'pumzgdpamn.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (3, '2020/10/27',
'tyyawoixzh.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (4, '2020/10/27',
'sdkaaauram.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (5, '2020/10/27',
'vgnxaqhyop.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (6, '2020/10/27',
'rhlhvhyoja.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (7, '2020/10/27',
'nrudfuxjdx.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (8, '2020/10/27',
'kxwqnqvgjj.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (9, '2020/10/27',
'spqmsbphxz.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (10, '2020/10/27',
'mnvflrwyvx.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (11, '2020/10/27',
'lcovqdyfqm.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (12, '2020/10/27',
'lpxapbjwts.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (13, '2020/10/27',
'smuffqhayg.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (14, '2020/10/27',
'rrhmqlsloi.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (15, '2020/10/27',
'vrtxamzxqz.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (16, '2020/10/27',
'eqyrgnbpls.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (17, '2020/10/27',
'rgqnplnlar.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (18, '2020/10/27',
'rtztkotazh.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (19, '2020/10/27',
'ufrsfczrzi.jpg');
INSERT INTO IMAGE (ImageID, Creation_date, Link) VALUES (20, '2020/10/27',
'bvccaoayyi.jpg');

-- Table: ORDERS
CREATE TABLE ORDERS (Order_Number INT NOT NULL, Order_Date DATE NOT NULL,
Account_number INT NOT NULL, CartID INT NOT NULL, FOREIGN KEY (Account_Number)
REFERENCES Buyer (Account_number), FOREIGN KEY (CartID) REFERENCES
SHOPPING_CART (CartID), PRIMARY KEY (Order_Number));
```

```
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (1, '2020-01-10', 1, 1);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (2, '2020-01-31', 2, 2);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (3, '2020-02-01', 3, 3);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (4, '2020-02-10', 4, 4);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (5, '2020-02-27', 5, 5);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (6, '2020-03-10', 6, 6);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (7, '2020-04-01', 7, 7);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (8, '2020-04-02', 8, 8);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (9, '2020-04-03', 9, 9);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (10, '2020-05-01', 10, 10);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (11, '2020-05-28', 21, 11);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (12, '2020-05-29', 22, 12);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (13, '2020-06-01', 23, 13);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (14, '2020-06-15', 24, 14);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (15, '2020-06-17', 25, 15);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (16, '2020-07-02', 26, 16);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (17, '2020-07-08', 27, 17);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (18, '2020-08-01', 28, 18);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (19, '2020-08-03', 29, 19);
INSERT INTO ORDERS (Order_Number, Order_Date, Account_number, CartID) VALUES (20, '2020-10-07', 30, 20);

-- Table: PAYMENT
CREATE TABLE PAYMENT
(PaymentID INT NOT NULL,
```

```
 Type_of_Payment VARCHAR(10) NOT NULL,
 Payment_Account_Number INT NOT NULL,
 ExpDate DATE NOT NULL,
 Order_Number INT NOT NULL,
Account_number INT NOT NULL,
 FOREIGN KEY (Order_Number) REFERENCES ORDERS(Order_Number),
 FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
 PRIMARY KEY(PaymentID));
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (1, 'Credit', 123456, '2020/10/27', 1, 1);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (2, 'Debit', 113456, '2020/10/27', 2, 2);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (3, 'Credit', 111456, '2020/10/27', 3, 3);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (4, 'Debit', 111156, '2020/10/27', 4, 4);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (5, 'Credit', 111116, '2020/10/27', 5, 5);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (6, 'Debit', 111111, '2020/10/27', 6, 6);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (7, 'Credit', 223456, '2020/10/27', 7, 7);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (8, 'Debit', 222456, '2020/10/27', 8, 8);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (9, 'Credit', 222256, '2020/10/27', 9, 9);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (10, 'Debit', 222226, '2020/10/27', 10, 10);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (11, 'Credit', 222222, '2020/10/27', 11,
21);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (12, 'Debit', 333456, '2020/10/27', 12, 22);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (13, 'Credit', 333356, '2020/10/27', 13,
23);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (14, 'Debit', 333336, '2020/10/27', 14, 24);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (15, 'Credit', 333333, '2020/10/27', 15,
25);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (16, 'Debit', 444456, '2020/10/27', 16, 26);
```

```
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (17, 'Credit', 444446, '2020/10/27', 17,
27);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (18, 'Debit', 444444, '2020/10/27', 18, 28);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (19, 'Credit', 555556, '2020/10/27', 19,
29);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (20, 'Debit', 555555, '2020/10/27', 20, 30);

-- Table: PAYMENT_METHODS
CREATE TABLE PAYMENT_METHODS
   (Method_name VARCHAR(15) NOT NULL,
    StoreID INT NOT NULL,
    FOREIGN KEY (StoreID) REFERENCES VIRTUAL_STOREFRONT(StoreID),
    PRIMARY KEY (Method_name, StoreID));
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 2);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 2);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 3);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 3);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 4);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 4);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 5);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 5);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 6);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 6);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 7);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 7);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 8);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 8);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 9);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 9);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 10);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 10);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 11);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 11);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 12);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 12);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 13);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 13);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 14);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 14);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 15);
```

```
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 15);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 16);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 16);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 17);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 17);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 18);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 18);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 19);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 19);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 20);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 20);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Karma', 1);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Debit', 1);
INSERT INTO PAYMENT_METHODS (Method_name, StoreID) VALUES ('Credit', 1);

-- Table: PRODUCT
CREATE TABLE PRODUCT
  (ProductID INT NOT NULL,
   StoreID INT NOT NULL,
   Name VARCHAR(15) NOT NULL,
   Buyer_feedback VARCHAR(100),
   Quantity INT,
   Availability VARCHAR(15) NOT NULL,
   Price DECIMAL(15, 2) NOT NULL CHECK(Price > 0),
  PRIMARY KEY (ProductID),
  FOREIGN KEY (StoreID) REFERENCES Virtual_storefront(StoreID));
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (1, 5, 'szycidpy',
'pumzgdpamntyyawoixzhsdkaaauramvgnxaqhyoprhlhvhyojanrudfuxjdxkxwqnqv', 2, 'Available',
5);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (2, 10, 'spqmsbph',
'nvflrwyvxlcovqdyfqmlpxapbjwtssmuffqhaygrrhmqlsloivrtxamzxqzeq', 5, 'Available', 6);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (3, 14, 'bplsrgqn',
'lnlarrtztkotazhufrsfczrzibvccaoayyihidztfljcffiqfviuwjowkppdajmknzgidixq', 2, 'Available', 160);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (4, 14, 'ahamebxf',
'wqvnrhuzwqohquamvszkvunbxjegbjccjjxfnsiearbsgsofywtqbmgldgsvnsgpdvm', 3, 'Available',
134);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (5, 16, 'aktmjafg', 'zszekngivdmrlvrpyrhcxbceffrgiyktqilkkdjhtywpesrydkb', 4,
'Available', 23);
```

INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (6, 13, 'zeekdtsz', 'csrhsciljsrdoidzbjatvacndzbghzsnfdofvhfxdnmzrjriwpkdgukbaa', 3, 'Available', 3);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (7, 20, 'komkmcck', 'odigztyrwpvlifrgjghlcicyocusukhmjbkfkzsjhkdrtsztchhazhmcircxcauajyzlppedqyzkcqvffyeekj', 1, 'Available', 186);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (8, 17, 'tjegerxb', 'tzvrxwgfjnrfbwvhiycvoznriroroamkfipazunsabwlseseei', 3, 'Available', 106);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (9, 19, 'mftchpaf', 'kquovuxhhkpvphwnkrtxuiuhbcyqulfqyzgjjwjrlfwwxotcdtqsmfeingsxyzbpvmwulmqfrxbqc', 3, 'Available', 165);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (10, 4, 'ixceytvv', 'cohmznmfkoetpgdntrndvjihmxragqosaauthigfjergijsyivozzfrlpndygsmgjzdzadsxarjvyxuecqlszjnqvlyqkadowol', 3, 'Available', 143);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (11, 13, 'kzxvspdu', 'mgraiutxxxqgotqnxwjwfotvqglqavmsnmktsxwxcpxhuujuanxueuymzi', 2, 'Available', 2);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (12, 20, 'alizwnvr', 'eoipfoqbiqdxsnclcvoafqwfwcmuwitjgqghkiccwqvloqrx', 1, 'Available', 48);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (13, 10, 'uxwriltx', 'mrmfpzitkwhitwhvatmknyhzigcuxfsosxetioq', 2, 'Available', 5);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (14, 5, 'woljymhd', 'wvjcdhmkpdfbbztaygvbpwqxtokvidtwfdhm', 2, 'Available', 131);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (15, 15, 'myfhhjor', 'mgowikpsdgcbazapkmsjgmfyuezaamevrbsmiecoujabrbqebiydncgapuexivgvomkuiiuuhhbszsflntwr', 5, 'Available', 20);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (16, 12, 'rnrgwrnv', 'wixtxycifdebgnbbuc', 5, 'Available', 125);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (17, 17, 'ldkberbo', 'emywoaxqicizkcjbmbxikxeizmzdvjdnhqrgkkqzmspdeuoqrxswqrajxfglmqkdnlescbjzurknjklikxxqqaqdekxkzks', 1, 'Available', 120);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (18, 9, 'polxmcsz', 'ebqpsizhwsxklzulm', 3, 'Available', 123);
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability, Price) VALUES (19, 20, 'krqfaeiv', 'sedfynxtbzdrviwdgicusqucczgufqnaslpwzj', 2, 'Available', 56);

```
INSERT INTO PRODUCT (ProductID, StoreID, Name, Buyer_feedback, Quantity, Availability,
Price) VALUES (20, 20, 'phnovlrg',
'xcingaxrymqpcmtqzssnbloagjwwuardjqxkyrusrjqnrqntusjojeqoseryfiuanxvsbln', 4, 'Available',
77);

-- Table: PRODUCT_IMAGE
CREATE TABLE PRODUCT_IMAGE
  (ImageID INT NOT NULL,
   ProductID INT NOT NULL,
   FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
   PRIMARY KEY (ProductID));
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (1, 1);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (2, 2);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (3, 3);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (4, 4);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (5, 5);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (6, 6);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (7, 7);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (8, 8);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (9, 9);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (10, 10);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (11, 11);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (12, 12);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (13, 13);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (14, 14);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (15, 15);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (16, 16);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (17, 17);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (18, 18);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (19, 19);
INSERT INTO PRODUCT_IMAGE (ImageID, ProductID) VALUES (20, 20);

-- Table: SELLER
CREATE TABLE SELLER
  (Account_number INT NOT NULL,
   PRIMARY KEY(Account_number));
INSERT INTO SELLER (Account_number) VALUES (10);
INSERT INTO SELLER (Account_number) VALUES (11);
INSERT INTO SELLER (Account_number) VALUES (12);
INSERT INTO SELLER (Account_number) VALUES (13);
INSERT INTO SELLER (Account_number) VALUES (14);
INSERT INTO SELLER (Account_number) VALUES (15);
INSERT INTO SELLER (Account_number) VALUES (16);
INSERT INTO SELLER (Account_number) VALUES (17);
```

```
INSERT INTO SELLER (Account_number) VALUES (18);
INSERT INTO SELLER (Account_number) VALUES (19);
INSERT INTO SELLER (Account_number) VALUES (20);
INSERT INTO SELLER (Account_number) VALUES (30);
INSERT INTO SELLER (Account_number) VALUES (31);
INSERT INTO SELLER (Account_number) VALUES (32);
INSERT INTO SELLER (Account_number) VALUES (33);
INSERT INTO SELLER (Account_number) VALUES (34);
INSERT INTO SELLER (Account_number) VALUES (35);
INSERT INTO SELLER (Account_number) VALUES (36);
INSERT INTO SELLER (Account_number) VALUES (37);
INSERT INTO SELLER (Account_number) VALUES (38);
INSERT INTO SELLER (Account_number) VALUES (39);
INSERT INTO SELLER (Account_number) VALUES (40);

-- Table: SHOP_PRODUCT
CREATE TABLE SHOP_PRODUCT
(CartID INT NOT NULL,
ProductID INT NOT NULL,
Quantity INT NOT NULL,
FOREIGN KEY (CartID) REFERENCES SHOPPING_CART(CartID),
FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
PRIMARY KEY (CartID, ProductID));
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (1, 1, 4);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (2, 2, 2);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (3, 3, 3);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (4, 4, 4);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (5, 5, 5);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (6, 6, 7);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (7, 7, 1);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (8, 8, 2);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (9, 9, 3);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (10, 10, 4);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (11, 11, 1);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (12, 12, 2);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (13, 13, 3);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (14, 14, 4);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (15, 15, 5);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (16, 16, 1);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (17, 17, 2);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (18, 18, 3);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (19, 19, 4);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (20, 20, 1);
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (1, 2, 300);
```

```
INSERT INTO SHOP_PRODUCT (CartID, ProductID, Quantity) VALUES (2, 3, 1);

-- Table: SHOPPING_CART
CREATE TABLE SHOPPING_CART (CartID INT NOT NULL, Purchased BOOLEAN NOT
NULL, Account_number INT NOT NULL, FOREIGN KEY (Account_Number) REFERENCES
Buyer (Account_number), PRIMARY KEY (CartID));
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (1, 1, 1);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (2, 1, 2);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (3, 1, 3);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (4, 1, 4);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (5, 1, 5);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (6, 1, 6);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (7, 1, 7);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (8, 1, 8);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (9, 1, 9);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (10, 1, 10);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (11, 1, 21);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (12, 1, 22);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (13, 1, 23);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (14, 1, 24);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (15, 1, 25);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (16, 1, 26);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (17, 1, 27);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (18, 1, 28);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (19, 1, 29);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (20, 1, 30);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (21, 0, 1);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (22, 0, 2);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (23, 0, 3);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (24, 0, 4);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (25, 0, 5);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (26, 0, 6);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (27, 0, 7);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (28, 0, 8);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (29, 0, 9);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (30, 0, 10);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (31, 0, 21);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (32, 0, 22);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (33, 0, 23);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (34, 0, 24);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (35, 0, 25);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (36, 0, 26);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (37, 0, 27);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (38, 0, 28);
```

```
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (39, 0, 29);
INSERT INTO SHOPPING_CART (CartID, Purchased, Account_number) VALUES (40, 0, 30);

-- Table: VIRTUAL_STOREFRONT
CREATE TABLE VIRTUAL_STOREFRONT
    (StoreID INT NOT NULL,
     Account_number INT NOT NULL,
     Name VARCHAR(15) NOT NULL,
     PRIMARY KEY (StoreID)
    FOREIGN KEY (Account_number) REFERENCES SELLER(Account_number));
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (1, 12,
'Jaydn Neal');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (2, 19,
'John Cobb');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (3, 11,
'Arissa Fitzpatrick');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (4, 14,
'Kristopher Terrell');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (5, 11,
'Arissa Fitzpatrick');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (6, 17,
'August Holman');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (7, 17,
'August Holman');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (8, 17,
'August Holman');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (9, 20,
'Alex Castaneda');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (10, 16,
'Fraya Britt');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (11, 13,
'Rachael Farley');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (12, 11,
'Arissa Fitzpatrick');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (13, 17,
'August Holman');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (14, 10,
'Tanner Wells');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (15, 16,
'Fraya Britt');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (16, 16,
'Fraya Britt');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (17, 19,
'John Cobb');
```

```
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (18, 10,
'Tanner Wells');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (19, 17,
'August Holman');
INSERT INTO VIRTUAL_STOREFRONT (StoreID, Account_number, Name) VALUES (20, 14,
'Kristopher Terrell');

-- Table: WISH_PRODUCT
CREATE TABLE WISH_PRODUCT
  (WishID INT NOT NULL,
   ProductID INT NOT NULL,
Quantity INT NOT NULL,
   FOREIGN KEY (WishID) REFERENCES WISHLIST(WishID),
   FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
   PRIMARY KEY (WishID, ProductID));
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (2, 2, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (3, 3, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (4, 4, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (5, 5, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (6, 6, 3);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (7, 7, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (8, 8, 3);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (9, 9, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (8, 9, 3);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (11, 11, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (1, 1, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (1, 2, 3);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (1, 3, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (1, 7, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (2, 3, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (3, 4, 1);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (4, 5, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (5, 6, 2);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (6, 7, 3);
INSERT INTO WISH_PRODUCT (WishID, ProductID, Quantity) VALUES (7, 8, 1);

-- Table: WISHLIST
CREATE TABLE "WISHLIST"
(WishID INT NOT NULL,
NumProducts INT NOT NULL,
Account_number INT NOT NULL,
FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
PRIMARY KEY(WishID));
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (16, 8, 26);
```

```
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (8, 8, 8);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (20, 9, 30);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (4, 9, 4);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (19, 5, 29);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (11, 5, 21);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (10, 9, 10);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (1, 4, 1);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (17, 5, 27);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (5, 8, 5);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (3, 3, 3);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (6, 3, 6);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (9, 2, 9);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (14, 2, 24);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (18, 2, 28);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (12, 3, 22);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (2, 3, 2);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (7, 2, 7);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (15, 10, 25);
INSERT INTO WISHLIST (WishID, NumProducts, Account_number) VALUES (13, 2, 23);

COMMIT TRANSACTION;
PRAGMA foreign_keys = on;
```

CSE 3241 Project Checkpoint 03 – SQL and More SQL

You will be submitting several nicely formatted files for this checkpoint. Provide the following:

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 02. If you were instructed to change the model for Project Checkpoint 02, make sure you use the revised versions of your models

2. Given your relational schema, create a text file containing the SQL code to create your database schema. Use this SQL to create a database in SQLite. Populate this database with the data provided for the project as well as 20 sample records for each table that does not contain data provided in the original project documents

```
CREATE TABLE ACCOUNT
(Account_number INT NOT NULL,
 Username VARCHAR(15) NOT NULL,
 Type VARCHAR(6) NOT NULL,
 Address VARCHAR(50) NOT NULL,
 Karma_points INT,
 Phone_number CHAR(10) NOT NULL,
 Transaction_history VARCHAR(30) NOT NULL,
 Name VARCHAR(30) NOT NULL,
 PRIMARY KEY(Account_number));

 CREATE TABLE BUYER
 (Account_number INT NOT NULL,
  PRIMARY KEY(Account_number));

  CREATE TABLE SELLER
  (Account_number INT NOT NULL,
   PRIMARY KEY(Account_number));

CREATE TABLE VIRTUAL_STOREFRONT
    (StoreID INT NOT NULL,
     Account_number INT NOT NULL,
     Name VARCHAR(15) NOT NULL,
     PRIMARY KEY (StoreID)
    FOREIGN KEY (Account_number) REFERENCES SELLER(Account_number));

    CREATE TABLE PAYMENT_METHODS
    (Method_name VARCHAR(15) NOT NULL,
     StoreID INT NOT NULL,
     FOREIGN KEY (StoreID) REFERENCES VIRTUAL_STOREFRONT(StoreID),
```

```
    PRIMARY KEY (Method_name, StoreID));


CREATE TABLE PAYMENT
(PaymentID INT NOT NULL,
Type_of_Payment VARCHAR(10) NOT NULL,
Payment_Account_Number INT NOT NULL,
ExpDate DATE NOT NULL,
Order_Number INT NOT NULL,
Account_number INT NOT NULL,
FOREIGN KEY (Order_Number) REFERENCES ORDERS(Order_Number),
FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
PRIMARY KEY(PaymentID));

INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (1, "Credit", 123456, '2020/10/27', 1, 1);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (2, "Debit", 113456, '2020/10/27', 2, 2);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (3, "Credit", 111456, '2020/10/27', 3, 3);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (4, "Debit", 111156, '2020/10/27', 4, 4);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (5, "Credit", 111116, '2020/10/27', 5, 5);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (6, "Debit", 111111, '2020/10/27', 6, 6);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (7, "Credit", 223456, '2020/10/27', 7, 7);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (8, "Debit", 222456, '2020/10/27', 8, 8);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (9, "Credit", 222256, '2020/10/27', 9, 9);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (10, "Debit", 222226, '2020/10/27', 10,
10);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (11, "Credit", 222222, '2020/10/27', 11,
21);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (12, "Debit", 333456, '2020/10/27', 12,
22);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (13, "Credit", 333356, '2020/10/27', 13,
23);
```

```
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (14, "Debit", 333336, '2020/10/27', 14,
24);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (15, "Credit", 333333, '2020/10/27', 15,
25);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (16, "Debit", 444456, '2020/10/27', 16,
26);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (17, "Credit", 444446, '2020/10/27', 17,
27);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (18, "Debit", 444444, '2020/10/27', 18,
28);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (19, "Credit", 555556, '2020/10/27', 19,
29);
INSERT INTO PAYMENT (PaymentID, Type_of_Payment, Payment_Account_Number,
ExpDate, Order_Number, Account_number) VALUES (20, "Debit", 555555, '2020/10/27', 20,
30);


CREATE TABLE ORDERS
(Order_Number INT NOT NULL,
Order_Date DATE NOT NULL,
Account_number INT NOT NULL,
CartID INT NOT NULL,
FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
FOREIGN KEY (CartID) REFERENCES SHOPPING_CART(CartID),
PRIMARY KEY(Order_Number));
INSERT INTO ORDERS VALUES (1, "2020-01-10", 1, 1), (2, "2020-01-31", 2, 2), (3,
"2020-02-01", 3, 3), (4, "2020-02-10", 4, 4), (5, "2020-02-27", 5, 5), (6, "2020-03-10", 6,6), (7,
"2020-04-01", 7, 7), (8, "2020-04-02", 8, 8), (9, "2020-04-03", 9, 9), (10, "2020-05-01", 10, 10),
(11, "2020-05-28", 21, 11), (12, "2020-05-29", 22, 12), (13, "2020-06-01", 23, 13), (14,
"2020-06-15", 24, 14), (15, "2020-06-17", 25, 15), (16, "2020-07-02", 26, 16), (17, "2020-07-08",
27, 17), (18, "2020-08-01", 28, 18), (19, "2020-08-03", 29, 19), (20, "2020-10-07", 30, 20);


CREATE TABLE SHOPPING_CART
(CartID INT NOT NULL,
Purchased BOOLEAN NOT NULL,
Account_number INT NOT NULL,
FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
```

```
PRIMARY KEY(CartID));
INSERT INTO SHOPPING_CART VALUES (1, TRUE, 1), (2, TRUE, 2), (3, TRUE, 3), (4, TRUE,
4), (5, TRUE, 5), (6, TRUE, 6), (7, TRUE, 7), (8, TRUE, 8), (9, TRUE, 9), (10, TRUE, 10), (11,
TRUE, 21), (12, TRUE, 22), (13, TRUE, 23), (14, TRUE, 24), (15, TRUE, 25), (16, TRUE, 26),
(17, TRUE, 27), (18, TRUE, 28), (19, TRUE, 29), (20, TRUE, 30), (21, FALSE, 1), (22, FALSE,
2), (23, FALSE, 3), (24, FALSE, 4), (25, FALSE, 5), (26, FALSE, 6), (27, FALSE, 7), (28, FALSE,
8), (29, FALSE, 9), (30, FALSE, 10), (31, FALSE, 21), (32, FALSE, 22), (33, FALSE, 23), (34,
FALSE, 24), (35, FALSE, 25), (36, FALSE, 26), (37, FALSE, 27), (38, FALSE, 28), (39, FALSE,
29), (40, FALSE, 30);



CREATE TABLE WISHLIST
(WishID INT NOT NULL,
NumProducts INT NOT NULL,
Account_number INT NOT NULL,
FOREIGN KEY (Account_Number) REFERENCES Buyer(Account_number),
PRIMARY KEY(WishID));



  CREATE TABLE PRODUCT
  (ProductID INT NOT NULL,
   StoreID INT NOT NULL,
   Name VARCHAR(15) NOT NULL,
   Buyer_feedback VARCHAR(100),
   Quantity INT,
   Availability VARCHAR(15) NOT NULL,
   Price DECIMAL(15, 2) NOT NULL CHECK(Price > 0),
  PRIMARY KEY (ProductID),
  FOREIGN KEY (StoreID) REFERENCES Virtual_storefront(StoreID));

   CREATE TABLE IMAGE
   (ImageID INT NOT NULL,
    Creation_date DATE NOT NULL,
    Link VARCHAR(60) NOT NULL,
    PRIMARY KEY (ImageID));


    CREATE TABLE WISH_PRODUCT
   (WishID INT NOT NULL,
    ProductID INT NOT NULL,
Quantity INT NOT NULL,
    FOREIGN KEY (WishID) REFERENCES WISHLIST(WishID),
```

```
    FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
    PRIMARY KEY (WishID, ProductID));
INSERT INTO WISH_PRODUCT(WishID,ProductID) VALUES (1, 1), (2, 2), (3, 3), (4, 4), (5, 5),
(6, 6), (7, 7), (8, 8), (9, 9), (10, 10), (11, 11), (12, 12), (13, 13), (14, 14), (15, 15), (16, 16), (17,
17), (18, 18), (19, 19), (20, 20);

CREATE TABLE SHOP_PRODUCT
(CartID INT NOT NULL,
ProductID INT NOT NULL,
Quantity INT NOT NULL,
FOREIGN KEY (CartID) REFERENCES SHOPPING_CART(CartID),
FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
PRIMARY KEY (CartID, ProductID));

INSERT INTO SHOP_PRODUCT(CartID, ProductID, Quantity) VALUES (1, 1, 1), (2, 2, 2), (3, 3,
3), (4, 4, 4), (5, 5, 5), (6, 6, 6), (7, 7, 1), (8, 8, 2), (9, 9, 3), (10, 10, 4), (11, 11, 1), (12, 12, 2), (13,
13, 3), (14, 14, 4), (15, 15, 5), (16, 16, 1), (17, 17, 2), (18, 18, 3), (19, 19, 4), (20, 20, 5);

CREATE TABLE PRODUCT_IMAGE
(ImageID INT NOT NULL,
ProductID INT NOT NULL,
FOREIGN KEY (ProductID) REFERENCES PRODUCT(ProductID),
PRIMARY KEY (ProductID));
```

3. Given your relational schema, provide the SQL to perform the following queries. If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. These queries should be provided in a plain text file named “SimpleQueries.txt”:

a. Find the titles of all IP Items by a given Seller that cost less than $10 (you choose how to designate the seller)

```
SELECT Product.Name
FROM Product, Virtual_Storefront, SELLER
WHERE SELLER.Account_number=16 AND Price < 10 AND PRODUCT.StoreID =
VIRTUAL_STOREFRONT.StoreID AND SELLER.Account_number =
VIRTUAL_STOREFRONT.Account_number
```

b. Give all the titles and their dates of purchase made by given buyer (you choose how to designate the buyer)

```
SELECT product.Name, ORDERS.Order_Date
```

```
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE BUYER.Account_number = 1 AND BUYER.Account_number =
ORDERS.Account_number AND ORDERS.CartID = SHOPPING_CART.CartID AND
SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND PRODUCT.ProductID =
SHOP_PRODUCT.ProductID
```

c. Find the seller names for all sellers with less than 5 IP Items for sale

```
SELECT ACCOUNT.Name
FROM Seller, Virtual_Storefront, Product, Account
WHERE Seller.Account_number = Virtual_Storefront.Account_number AND Product.StoreID =
Virtual_Storefront.StoreID AND ACCOUNT.Account_number = SELLER.Account_number
GROUP BY Seller.Account_number
HAVING SUM(PRODUCT.Quantity) < 5;
```

d. Give all the buyers who purchased a IP Item by a given seller and the names of the IP Items they purchased

```
SELECT ACCOUNT.Name, PRODUCT.Name
FROM ACCOUNT, PRODUCT, BUYER, SELLER, VIRTUAL_STOREFRONT,
SHOPPING_CART, SHOP_PRODUCT, ORDERS
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
SHOP_PRODUCT.ProductID = PRODUCT.ProductID AND PRODUCT.StoreID =
VIRTUAL_STOREFRONT.StoreID AND VIRTUAL_STOREFRONT.Account_number =
SELLER.Account_number AND SELLER.Account_number = 11
```

e. Find the total number of IP Items purchased by a single buyer (you choose how to designate the buyer)

```
SELECT SUM(SHOP_PRODUCT.Quantity)
FROM (((BUYER NATURAL JOIN ORDERS) NATURAL JOIN SHOPPING_CART) NATURAL
JOIN SHOP_PRODUCT)
WHERE Account_Number = 1 AND SHOPPING_CART.purchased = TRUE
```

f. Find the buyer who has purchased the most IP Items and the total number of IP Items they have purchased

```
SELECT BUYER.Account_number, SUM(SHOP_PRODUCT.QUANTITY) AS Total
FROM (((Buyer NATURAL JOIN ORDERS) NATURAL JOIN Shopping_Cart) NATURAL JOIN
SHOP_PRODUCT)
```

```
WHERE SHOPPING_CART.purchased = TRUE
GROUP BY Account_number
ORDER BY Total LIMIT(1)
```

4. For Project Checkpoint 02, you were asked to come up with three additional interesting queries that your database can provide. Provide the SQL to perform those queries. Your queries should include at least one of these:

a. outer joins
Find the buyers who don't have a wishlist.

```
SELECT Buyer.Account_Number
FROM (Buyer LEFT JOIN Wishlist ON BUYER.Account_number =
WISHLIST.Account_Number)
WHERE WishID IS NULL
```

b. aggregate function (min, max, average, etc)
Total amount of money paid by a given buyer.

```
SELECT SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) AS total
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE BUYER.Account_number = 1 AND BUYER.Account_number =
ORDERS.Account_number AND ORDERS.CartID = SHOPPING_CART.CartID AND
SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND PRODUCT.ProductID =
SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased = TRUE
GROUP BY BUYER.Account_Number
```

c. "extra" entities from CP01
Find the buyer who has the most IP products in the wishlist.

```
SELECT BUYER.Account_number, SUM(WISH_PRODUCT.QUANTITY) AS Total
FROM ((Buyer NATURAL JOIN WishList) NATURAL JOIN SHOP_PRODUCT)
GROUP BY Account_number
ORDER BY Total DESC LIMIT(1)
```

If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. These queries should be provided in a plain text file named "ExtraQueries.txt":

5. Given your relational schema, provide the SQL for the following more advanced queries. These queries may require you to use techniques such as nesting, aggregation using having clauses, and other SQL techniques .

If your database schema does not contain the information to answer these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries.

Note that if your database does contain the information but in non-aggregated form, you should NOT revise your model but instead figure out how to aggregate it for the query!

These queries should be provided in a plain text file named “AdvancedQueries.txt”.
a. Provide a list of buyer names, along with the total dollar amount each buyer has spent.

```
SELECT ACCOUNT.Name, SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) AS totalCost
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY BUYER.Account_Number
```

b. Provide a list of buyer names and e-mail addresses for buyers who have spent more than the average buyer.

```
SELECT ACCOUNT.Name, ACCOUNT.Username
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY BUYER.Account_Number
HAVING SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) > (SELECT AVG(totalCost)
FROM (SELECT SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) AS totalCost
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY BUYER.Account_Number))
```

c. Provide a list of the IP Item names and associated total copies sold to all buyers, sorted from the IP Item that has sold the most individual copies to the IP Item that has sold the least.

```
SELECT PRODUCT.Name, SUM(SHOP_PRODUCT.QUANTITY) AS Total
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY PRODUCT.Name
ORDER BY Total DESC
```

d. Provide a list of the IP Item names and associated dollar totals for copies sold to all buyers, sorted from the IP Item that has sold the highest dollar amount to the IP Item that has sold the smallest.

```
SELECT PRODUCT.Name, SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) AS Total
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY PRODUCT.Name
ORDER BY Total DESC
```

e. Find the most popular Seller (i.e. the one who has sold the most IP Items)

```
SELECT SELLER.Account_number, SUM(SHOP_PRODUCT.QUANTITY) AS Total
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT, SELLER,
VIRTUAL_STOREFRONT
WHERE PRODUCT.StoreID = VIRTUAL_STOREFRONT.StoreID AND
SELLER.Account_number = VIRTUAL_STOREFRONT.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY SELLER.Account_number
ORDER BY Total DESC LIMIT(1)
```

f. Find the most profitable seller (i.e. the one who has brought in the most money)

```
SELECT SELLER.Account_number, SUM(SHOP_PRODUCT.QUANTITY * PRODUCT.Price)
AS Total
```

```
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT, SELLER,
VIRTUAL_STOREFRONT
WHERE PRODUCT.StoreID = VIRTUAL_STOREFRONT.StoreID AND
SELLER.Account_number = VIRTUAL_STOREFRONT.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY SELLER.Account_number
ORDER BY Total DESC LIMIT(1)
```

g. Provide a list of buyer names for buyers who purchased anything listed by the most profitable Seller.

```
SELECT ACCOUNT.Name
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT,
VIRTUAL_STOREFRONT, SELLER
WHERE PRODUCT.StoreID = VIRTUAL_STOREFRONT.StoreID AND
SELLER.Account_number = VIRTUAL_STOREFRONT.Account_number AND
ACCOUNT.Account_number = BUYER.Account_number AND BUYER.Account_number =
ORDERS.Account_number
AND ORDERS.CartID = SHOPPING_CART.CartID AND SHOPPING_CART.CartID =
SHOP_PRODUCT.CartID AND PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND
SHOPPING_CART.purchased = TRUE
AND SELLER.Account_number = ( SELECT SELLER.Account_number
FROM BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT, SELLER,
VIRTUAL_STOREFRONT
WHERE PRODUCT.StoreID = VIRTUAL_STOREFRONT.StoreID AND
SELLER.Account_number = VIRTUAL_STOREFRONT.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY SELLER.Account_number
ORDER BY SUM(SHOP_PRODUCT.QUANTITY * PRODUCT.Price) DESC LIMIT(1))
```

h. Provide the list of sellers who listed the IP Items purchased by the buyers who have spent more than the average buyer.

```
SELECT SELLER.Account_number
FROM SELLER, SHOP_PRODUCT, SHOPPING_CART, VIRTUAL_STOREFRONT, PRODUCT
WHERE PRODUCT.StoreID = VIRTUAL_STOREFRONT.StoreID AND
SELLER.Account_number = VIRTUAL_STOREFRONT.Account_number AND
```

```
SHOP_PRODUCT.CartID = SHOPPING_CART.CartID AND
SHOPPING_CART.Account_number = (
SELECT ACCOUNT.Account_number
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY BUYER.Account_Number
HAVING SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) > (SELECT AVG(totalCost)
FROM (SELECT SUM(PRODUCT.price * SHOP_PRODUCT.Quantity) AS totalCost
FROM ACCOUNT, BUYER, ORDERS, SHOPPING_CART, SHOP_PRODUCT, PRODUCT
WHERE ACCOUNT.Account_number = BUYER.Account_number AND
BUYER.Account_number = ORDERS.Account_number AND ORDERS.CartID =
SHOPPING_CART.CartID AND SHOPPING_CART.CartID = SHOP_PRODUCT.CartID AND
PRODUCT.ProductID = SHOP_PRODUCT.ProductID AND SHOPPING_CART.purchased =
TRUE
GROUP BY BUYER.Account_Number)))
GROUP BY SELLER.Account_number
```

6. Once you have completed all of the questions for Part Two, create a ZIP archive containing the binary SQLite file and the three text files and submit this to the Carmen Dropbox.

Make sure your queries work against your database and provide your expected output before you submit them!