Hult International Business School

Data Management & SQL - DAT-5305

A2: Project in Pairs

**eCommerce Database**

**Team #2**

Rodrigo Ramos Landeros

M Raafi Jahangir

Flow of System

Start

Order delivered

End

Notifies customer that their product has been delivered

Ships product

Order experiences

issues

Order ready for shipping

Send tracking information to customer

Reach out to customer to let them know there has been an error and their item is delayed

Order experiences

issues

Supplier check product stock

Processes order and gets it ready for shipping

Reach out to customer to let them know and asks them if they want a refund or order another product

Out of stock

Display order confirmation

Order details sent out to supplier

Accepted

Customer places an order

Payment is being verified

Rejected

Display payment error and allows customer to re-enter payment details or use another payment method

Database Structure

We have created seven tables in our database.

1. The first table created was *customers* which records all of the customers information including, customer id, first name, last name, gender, customer since, email address, home address, and country. This table has a primary key, which his *customer\_id.*
2. The second table created was *orders* which includes all necessary order information including order id, order date, order status and order value. This table has a primary key which is *order\_id* and foreign key from the *customers* table which is *customer\_id*.
3. The third table created was *payments* which includes all payment information including payment type, payment date and payment status. This table has a foreign key from the *orders* table which is *order\_id*.
4. The fourth table created was *order\_details* which includes all extra order information including unit price, quantity, discount and item status. This table has two foreign keys. The first one is from the *orders* table, which is *order\_id*. The second one is from the next table, *products,* which is *product\_id.*
5. The fifth table created was *products* which records all of the product information including, product name, unit price, quantity, item stock, and discontinued flag. This table has a primary key, which his *product\_id.* The table also has two foreign keys. The first one is from the next table, *category*, which is *category\_id*. The second one is from the table after, *suppliers,* which is *supplier\_id.*
6. The sixth table created was *category* which includes the product category information including category name and category description. This table has a primary key, which is *category\_id*.
7. The seventh and final table created was *suppliers* which includes all information regarding the supplier for the product including company name, company contact details, company address, and company country. This table has a primary key, which is *supplier\_id.*

Entity-Relationship Diagram (ERD)

Diagram

Description automatically generated

10 SQL Queries

**Note:**

For the JOINS we decided to use the following nomenclature for the tables:

category AS pc (for “product category”)

customers AS c

order\_details AS od

orders AS o

payments AS m (for “money”)

products AS p

suppliers AS s

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Query 1:**

**We want to find out the name and price of the most expensive item we sell.**

SELECT

product\_name, unit\_price

FROM

products

ORDER BY unit\_price DESC

LIMIT 1;

**Output:**

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Query 2:**

**We want to find out the number of customers per state ordered from highest to lowest.**

SELECT

state, COUNT(1) AS no\_of\_customers

FROM

customers

GROUP BY state

ORDER BY no\_of\_customers DESC;

**Output:**

Table

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Query 3:**

**We want to find out the name, unit price, unit stock and discontinued flag for the products that weren’t ordered by our customers.**

SELECT

p.product\_name,

p.unit\_price,

p.unit\_stock,

p.discontinued\_flag

FROM

products AS p

LEFT JOIN

order\_details AS od ON p.product\_id = od.product\_id

WHERE

od.product\_id IS NULL;

**Output:**

A screenshot of a computer

Description automatically generated with low confidence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Query 4:**

**We want to find out the name of our top supplier in terms of total revenue.**

SELECT

s.company\_name, SUM(o.order\_value) AS total\_revenue

FROM

suppliers AS s

JOIN

products AS p ON s.supplier\_id = p.supplier\_id

JOIN

order\_details AS od ON p.product\_id = od.product\_id

JOIN

orders AS o ON od.order\_id = o.order\_id

GROUP BY s.company\_name

ORDER BY total\_revenue DESC

LIMIT 1;

**Output:**

A picture containing graphical user interface

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Query 5:**

**We want to find out the most popular categories where our customers are ordering from.**

SELECT

pc.category\_name, COUNT(o.order\_id) AS no\_of\_orders

FROM

category AS pc

JOIN

products AS p ON pc.category\_id = p.category\_id

JOIN

order\_details AS od ON p.product\_id = od.product\_id

JOIN

orders AS o ON od.order\_id = o.order\_id

JOIN

customers AS c ON o.customer\_id = c.customer\_id

GROUP BY pc.category\_name

ORDER BY no\_of\_orders DESC;

**Output:**

**Table

Description automatically generated**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Query 6:**

**We want to find out the top 5 customer first name, order id and the average price per order ranked by the most expensive average price per order.**

SELECT

first\_name,

od.order\_id,

ROUND(AVG(unit\_price), 2) AS avg\_price\_per\_order

FROM

order\_details AS od

JOIN

orders AS o ON od.order\_id = o.order\_id

JOIN

customers AS c ON o.customer\_id = c.customer\_id

GROUP BY od.order\_id

ORDER BY avg\_price\_per\_order DESC

LIMIT 5;

**Output:**

Table

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Query 7:**

**We want to find out the customer details who spent the most money on orders**

SELECT

c.first\_name,

c.last\_name,

c.gender,

c.customer\_since,

c.state,

c.country,

ROUND(SUM(o.order\_value), 2) AS total\_spent

FROM

orders AS o

JOIN

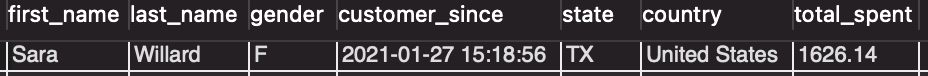
customers AS c ON o.customer\_id = c.customer\_id

GROUP BY o.customer\_id

ORDER BY total\_spent DESC

LIMIT 1;

**Output:**

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Query 8:**

**We want to find out the order id, order date, customer full name, order value, order status, payment type and payment status where the order status was “Cancelled”.**

SELECT

o.order\_id,

o.order\_date,

c.first\_name,

c.last\_name,

o.order\_value,

o.order\_status,

m.payment\_type,

m.payment\_status

FROM

orders AS o

JOIN

customers AS c ON o.customer\_id = c.customer\_id

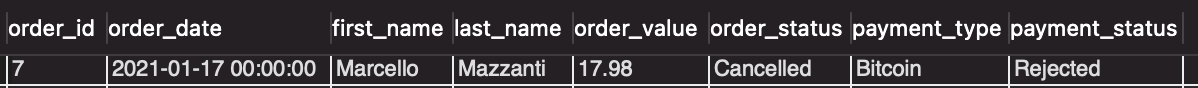
JOIN

payments AS m ON o.order\_id = m.order\_id

WHERE

o.order\_status = 'Cancelled';

**Output:**

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Query 9:**

**We want to find out the full name of customers and their amount of orders who made more than one order in the last two weeks.**

SELECT CONCAT\_WS(", ", first\_name, last\_name) as cust\_full\_name,

COUNT(o.customer\_id) as no\_of\_orders

FROM customers as c

JOIN orders as o

ON c.customer\_id = o.customer\_id

WHERE o.order\_date BETWEEN (NOW() - INTERVAL 14 DAY) AND NOW()

GROUP BY cust\_full\_name

HAVING no\_of\_orders > 1;

Output:

Table

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Query 10** *(continues to next page)***:**

**We want to find out what is the most popular category per gender and how many orders did they make from that category.**

(SELECT

c.gender AS customer\_gender,

pc.category\_name AS product\_category,

COUNT(o.order\_id) AS no\_of\_order

FROM

category AS pc

JOIN

products AS p ON pc.category\_id = p.category\_id

JOIN

order\_details AS od ON p.product\_id = od.product\_id

JOIN

orders AS o ON od.order\_id = o.order\_id

JOIN

customers AS c ON o.customer\_id = c.customer\_id

WHERE

Gender = 'M'

GROUP BY customer\_gender, product\_category

ORDER BY no\_of\_order DESC

LIMIT 1)

UNION ALL

(SELECT

c.gender AS customer\_gender,

pc.category\_name AS product\_category,

COUNT(o.order\_id) AS no\_of\_order

FROM

category AS pc

JOIN

products AS p ON pc.category\_id = p.category\_id

JOIN

order\_details AS od ON p.product\_id = od.product\_id

JOIN

orders AS o ON od.order\_id = o.order\_id

JOIN

customers AS c ON o.customer\_id = c.customer\_id

WHERE

Gender = 'F'

GROUP BY customer\_gender, product\_category

ORDER BY no\_of\_order DESC

LIMIT 1)

UNION ALL

(SELECT

c.gender AS customer\_gender,

pc.category\_name AS product\_category,

COUNT(o.order\_id) AS no\_of\_order

FROM

category AS pc

JOIN

products AS p ON pc.category\_id = p.category\_id

JOIN

order\_details AS od ON p.product\_id = od.product\_id

JOIN

orders AS o ON od.order\_id = o.order\_id

JOIN

customers AS c ON o.customer\_id = c.customer\_id

WHERE

Gender = 'O'

GROUP BY customer\_gender, product\_category

ORDER BY no\_of\_order DESC

LIMIT 1);

**Output:**

Table

Description automatically generated  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 Detailed SQL Procedures

**Procedure 1:**

Outline: This procedure will allow the admin to insert a discount (in percent) to a specific product in a specific order. This would not only update the newly discounted product from a specific order, but also update the orders table with the full information on all orders.

Input:

Order id (INT),

Product id (INT),

Discount in percent (INT)

Output: None

Description:

The database administrator will have to input the order ID, product ID and discount in percent for how much the company wants to give to that specific set of item(s). Once this three information is entered, MySQL will filter that particular item in the products table and will assign the value to a variable that we will use later on. After that, the procedure is coded to update the `order\_details` table for the record that was requested and will set the new price after the discount is given. Now that we have a new product price, we also need to update the orders table because the total order `order\_value` needs to change as well, and it will recalculate the value on of the entire order based on the new information.

Procedure code:

DROP PROCEDURE IF EXISTS additional\_discount;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `additional\_discount`(

IN in\_order\_id INT,

IN in\_prod\_id INT,

IN in\_discount\_in\_percent INT

)

BEGIN

DECLARE disc\_price DECIMAL(10,2);

SELECT

p.unit\_price

INTO disc\_price

FROM

products AS p

JOIN

order\_details AS od ON p.product\_id = od.product\_id

WHERE

od.order\_id = in\_order\_id

AND od.product\_id = in\_prod\_id;

UPDATE order\_details

SET unit\_price = disc\_price \*(1-(in\_discount\_in\_percent/100)),

discount = (1-(in\_discount\_in\_percent/100))

WHERE

order\_id = in\_order\_id

AND product\_id = in\_prod\_id;

UPDATE orders AS o

JOIN  
 (SELECT

od.order\_id, SUM(od.unit\_price \* od.quantity) AS new\_order\_value

FROM

order\_details AS od

GROUP BY

od.order\_id) AS nod ON o.order\_id = nod.order\_id

SET o.order\_value = nod.new\_order\_value

WHERE

o.order\_id = nod.order\_id;

END $$

DELIMITER ;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Procedure 2:**

Outline: This procedure will allow the admin to change the status of an item. This will not only update our existing `order\_details` item updates, but also will update the master `orders` table with an order update. This is crucial because it provides our customers with the most up-to-date information on their orders.

Input:

Order id (INT),

Product id (INT),

Item Status ENUM('Pending','Shipped','Delivered','Returned','Cancelled')

Output: None

Description:

The database administrator will have to input the order ID, product ID and the updated status of the item. When the admin enters the status, the `order\_details` table will be updated and will replace the previous entry. The procedure will also update the `orders` table with the use of flags. Since we have five possible statuses, the procedure has an IF statement that goes through the updated item status and checks if the whole order from the `orders` table is satisfied. Depending on the 5 possible status, we look for each status on the order and if found, a flag will return to TRUE. Once we got through the 5 statuses, we can then proceed to update the orders table accordingly depending on the statuses that where triggered. If one item is ‘Shipped’, but the second item on the order is still ‘Pending’, it will output ‘Partly Shipped’ on the main `order\_status` on the `orders` table, however if both items are ‘Shipped’, it will output `Shipped` for the whole order on the `order\_status` on the `orders` table. This applies to all the other variables.

Procedure code:

DROP PROCEDURE IF EXISTS overall\_order\_status;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `overall\_order\_status`(

IN in\_order\_id INT,

IN in\_prod\_id INT,

IN in\_item\_status ENUM('Pending','Shipped','Delivered','Returned','Cancelled')

)

BEGIN

DECLARE status\_pending BOOL;

DECLARE status\_shipped BOOL;

DECLARE status\_delivered BOOL;

DECLARE status\_returned BOOL;

DECLARE status\_cancelled BOOL;

UPDATE order\_details

SET item\_status = CONCAT(UPPER(SUBSTRING(in\_item\_status,1,1)),LOWER(SUBSTRING(in\_item\_status,2)))

WHERE

order\_id = in\_order\_id

AND product\_id = in\_prod\_id;

SET status\_pending = FALSE,

status\_shipped = FALSE,

status\_delivered = FALSE,

status\_returned = FALSE,

status\_cancelled = FALSE;

IF (SELECT item\_status

FROM order\_details

WHERE order\_id = in\_order\_id AND item\_status = 'Pending'

GROUP BY item\_status) IS NOT NULL

THEN

SET status\_pending = TRUE;

ELSE

SET status\_pending = FALSE;

END IF;

IF (SELECT item\_status

FROM order\_details

WHERE order\_id = in\_order\_id AND item\_status = 'Shipped'

GROUP BY item\_status) IS NOT NULL

THEN

SET status\_shipped = TRUE;

ELSE

SET status\_shipped = FALSE;

END IF;

IF (SELECT item\_status

FROM order\_details

WHERE order\_id = in\_order\_id AND item\_status = 'Delivered'

GROUP BY item\_status) IS NOT NULL

THEN

SET status\_delivered = TRUE;

ELSE

SET status\_delivered = FALSE;

END IF;

IF (SELECT item\_status

FROM order\_details

WHERE order\_id = in\_order\_id AND item\_status = 'Returned'

GROUP BY item\_status) IS NOT NULL

THEN

SET status\_returned = TRUE;

ELSE

SET status\_returned = FALSE;

END IF;

IF (SELECT item\_status

FROM order\_details

WHERE order\_id = in\_order\_id AND item\_status = 'Cancelled'

GROUP BY item\_status) IS NOT NULL

THEN

SET status\_cancelled = TRUE;

ELSE

SET status\_cancelled = FALSE;

END IF;

UPDATE orders

SET order\_status = CASE

WHEN status\_cancelled = TRUE

THEN 'Cancelled'

WHEN status\_returned = TRUE

THEN 'Returned'

WHEN status\_delivered = TRUE

AND (status\_pending = TRUE OR status\_shipped = TRUE)

THEN 'Part\_Delivered'

WHEN status\_delivered = TRUE THEN 'Delivered'

WHEN status\_shipped = TRUE

AND status\_pending = TRUE

THEN 'Part\_Shipped'

WHEN status\_shipped = TRUE THEN 'Shipped'

WHEN status\_pending = TRUE THEN 'Pending'

ELSE order\_status

END

WHERE

order\_id = in\_order\_id;

END $$

DELIMITER ;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Appendix

**Table Scripts**

DROP DATABASE IF EXISTS ecommerce\_site;

CREATE DATABASE IF NOT EXISTS ecommerce\_site;

USE ecommerce\_site;

CREATE TABLE IF NOT EXISTS customers

(

customer\_id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(45) NOT NULL,

last\_name VARCHAR(45) NOT NULL,

gender ENUM('M','F','O') NOT NULL,

email\_address VARCHAR(255) NOT NULL CHECK (email\_address LIKE '%@%'),

customer\_since TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP(),

address1 VARCHAR(255) NOT NULL,

address2 VARCHAR(255),

postal\_code CHAR(5) NOT NULL,

city VARCHAR(45) NOT NULL,

state CHAR(2) NOT NULL,

country VARCHAR(45) NOT NULL,

PRIMARY KEY (customer\_id),

UNIQUE KEY `email\_address` (email\_address)

) AUTO\_INCREMENT = 1;

CREATE TABLE IF NOT EXISTS category

(

category\_id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

category\_name VARCHAR(45) NOT NULL,

PRIMARY KEY (category\_id),

UNIQUE KEY `category\_name` (category\_name)

) AUTO\_INCREMENT = 1;

CREATE TABLE IF NOT EXISTS payments

(

order\_id INT UNSIGNED NOT NULL,

payment\_type ENUM('Credit Card','Debit Card','Check','Bank Account','Bitcoin') NOT NULL,

payment\_date TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP(),

payment\_status ENUM('Successful','Pending','Rejected') NOT NULL,

CONSTRAINT `payments\_to\_order\_id` FOREIGN KEY (order\_id) REFERENCES `orders` (`order\_id`) ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS suppliers

(

supplier\_id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

company\_name VARCHAR(45) NOT NULL,

company\_contact VARCHAR(45) NOT NULL,

company\_email VARCHAR(255) NOT NULL CHECK (company\_email LIKE '%@%'),

company\_phone\_code CHAR(4) NOT NULL CHECK (company\_phone\_code LIKE '+%'),

company\_phone CHAR(10) NOT NULL,

address1 VARCHAR(255) NOT NULL,

address2 VARCHAR(255),

postal\_code CHAR(5) NOT NULL,

city VARCHAR(45) NOT NULL,

state CHAR(2) NOT NULL,

country VARCHAR(45) NOT NULL,

PRIMARY KEY (supplier\_id)

) AUTO\_INCREMENT = 1;

CREATE TABLE IF NOT EXISTS products

(

product\_id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

product\_name VARCHAR(255) NOT NULL,

unit\_price NUMERIC(10,2) UNSIGNED NOT NULL,

unit\_stock INT UNSIGNED NOT NULL DEFAULT 0,

discontinued\_flag ENUM('Y','N'),

category\_id INT UNSIGNED NOT NULL,

supplier\_id INT UNSIGNED NOT NULL,

PRIMARY KEY (product\_id),

CONSTRAINT `products\_to\_category\_id` FOREIGN KEY (category\_id) REFERENCES `category` (`category\_id`) ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT `products\_to\_supplier\_id` FOREIGN KEY (supplier\_id) REFERENCES `suppliers` (`supplier\_id`) ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS orders

(

order\_id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

order\_date TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP(),

customer\_id INT UNSIGNED,

order\_status ENUM('Pending', 'Shipped','Part\_Shipped','Delivered','Part\_Delivered','Returned','Cancelled') NOT NULL,

order\_value NUMERIC(10,2) NOT NULL,

PRIMARY KEY (order\_id),

CONSTRAINT `orders\_to\_customer\_id` FOREIGN KEY (customer\_id) REFERENCES `customers` (`customer\_id`) ON UPDATE CASCADE ON DELETE CASCADE

) AUTO\_INCREMENT = 1;

CREATE TABLE IF NOT EXISTS order\_details

(

order\_id INT UNSIGNED NOT NULL,

product\_id INT UNSIGNED NOT NULL,

unit\_price NUMERIC(10,2) UNSIGNED NOT NULL,

discount NUMERIC(10,2) UNSIGNED,

quantity INT UNSIGNED NOT NULL,

item\_status ENUM('Pending', 'Shipped','Delivered','Returned','Cancelled') NOT NULL,

CONSTRAINT `details\_to\_order\_id` FOREIGN KEY (order\_id) REFERENCES `orders` (`order\_id`) ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT `details\_to\_product\_id` FOREIGN KEY (product\_id) REFERENCES `products` (`product\_id`) ON UPDATE CASCADE ON DELETE CASCADE

);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Insert Values Scripts**

INSERT INTO customers (

first\_name, last\_name, gender, email\_address,

address1, address2, postal\_code, city, state, country

)

VALUES

('Rodrigo','Landeros','M','mail1@gmail.com','1 Education St',DEFAULT,'02141','Cambridge','MA','United States'),

('M Raafi','Jahangir','M','mail2@gmail.com','2 Education St','12','02142','North End','MA','United States'),

('Marcello','Mazzanti','M','mail@outook.com','3 Education St',DEFAULT,'02143','Houston','TX','United States'),

('Max','Lemke','M','mail@icloud.com','4 Education St','344','02144','New York','NY','United States'),

('Mike','Timberlake','M','mail@aol.com','5 Education St','2','02145','San Francisco','CA','United States'),

('Pom','Pom','F','mail@yahoo.com','6 Education St','1','02146','Cuppertino','CA','United States'),

('Sara','Willard','F','mail@gov.com','7 Education St',DEFAULT,'02147','Austin','TX','United States'),

('Katherina','Bursy','F','mail@hult.edu','8 Education St',DEFAULT,'02148','Miami','FL','United States'),

('Siri','Siri','O','mail@apple.com','9 Education St','33','02149','Brooklyn','NY','United States'),

('Alexa','Alexa','O','mail@amazon.com','10 Education St',DEFAULT,'02150','Aliston','MA','United States');

INSERT INTO category (category\_name)

VALUES

('Clothing, Shoes'),

('Home'),

('Sports'),

('Books'),

('Automotive'),

('Pet Supplies'),

('Electronics'),

('Computers'),

('Office'),

('Food & Grocery');

INSERT INTO suppliers (

company\_name, company\_contact, company\_email, company\_phone\_code, company\_phone, address1, address2, postal\_code, city, state, country

)

VALUES

('a','aa','mail1@gmail.com','+1','0000000001','1 Education St',DEFAULT,'02141','Cambridge','MA','United States'),

('b','bb','mail2@gmail.com','+1','0000000002','2 Education St','12','02142','North End','MA','United States'),

('c','cc','mail@outook.com','+1','0000000003','3 Education St',DEFAULT,'02143','Houston','TX','United States'),

('d','dd','mail@icloud.com','+1','0000000004','4 Education St','344','02144','New York','NY','United States'),

('e','ee','mail@aol.com','+1','0000000005','5 Education St','2','02145','San Francisco','CA','United States'),

('f','ff','mail@yahoo.com','+1','0000000006','6 Education St','1','02146','Cuppertino','CA','United States'),

('g','gg','mail@gov.com','+1','0000000007','7 Education St',DEFAULT,'02147','Austin','TX','United States'),

('h','hh','mail@hult.edu','+1','0000000008','8 Education St',DEFAULT,'02148','Miami','FL','United States'),

('i','ii','mail@apple.com','+1','0000000009','9 Education St','33','02149','Brooklyn','NY','United States'),

('j','jj','mail@amazon.com','+1','0000000010','10 Education St',DEFAULT,'02150','Aliston','MA','United States');

INSERT INTO products (

product\_name, unit\_price, unit\_stock, discontinued\_flag, category\_id, supplier\_id

)

VALUES

("Face Mask | Adult 2-Pack | Anti-Microbial, Reusable, Cloth, Adjustable, Breathable | The Revival Mask by coRevival (Black)",'19.99','8','N','1','1'),

("think! High Protein Bars 20g Protein, No Artificial Sweeteners, Brownie Crunch, 10 Count",'15.84','16','N','10','10'),

("RION Cycling Bibs Shorts Men's Bike Padded Tights Bicycle Pants",'38.99','2','N','3','6'),

("illy Intenso Ground Espresso Coffee, Dark Roast, Intense, Robust and Full Flavored With Notes of Deep Cocoa, 100% Arabica Coffee, No Preservatives, 8.8 Ounce (Pack of 2)",'19.6','12','N','10','10'),

("Chef Craft Can Opener with Tapper",'3.97','18','N','2','3'),

("UGREEN USB-C to USB-C Cable, USB Type C 100W Power Delivery PD Charging Cord for Apple MacBook Pro, Huawei Matebook, iPad Pro 2020, Chromebook, Pixel 3 XL, Samsung Note 10 S20 S10, Nintendo Switch 3FT",'8.99','11','N','7','7'),

("USB C to HDMI Cable, CHOETECH Type-C to HDMI Adapter 6FT 60W PD Powering Cable Thunderbolt 3(4K@60Hz) Compatible with MacBook Pro2020/iPad Pro/MacBook Air,iMac 2017,Samsung Galaxy S10/S9/S9+/Note9/S8",'19.99','16','N','7','7'),

("Orionstar Laptop Stand Portable Aluminum Laptop Riser Compatible with Apple Mac MacBook Air Pro 10 to 15.6 Inch Notebook Computer, Detachable Ergonomic Elevator Holder, Space Grey",'25.99','8','N','9','7'),

("USB C Charger RAVPower 61W PD 3.0 Wall Charger Fast Charging Type C Foldable Adapter with dual Ports Portable Charger for laptop MacBook Pro tablets iPad Pro iPhone 12 Mini Pro Max Galaxy S20 Nintendo",'25.99','18','N','7','7'),

("Gaggia Carezza De LUXE Espresso Machine, Silver",'332.41','18','N','2','4'),

("UNNI ASTM D6400 100% Compostable Trash Bags, 2.6 Gallon, 9.84 Liter, 100 Count, Extra Thick 0.71 Mils, Food Scrap Small Kitchen Trash Bags, US BPI and Europe OK Compost Home Certified, San Francisco",'11.95','2','N','2','5'),

("Superior Glass Meal Prep Containers - 6-pack (35oz) Newly Innovated Hinged BPA-free Locking lids - 100% Leak Proof Glass Food Storage Containers, Great on-the-go, Freezer to Oven Safe Lunch Containers",'29.99','20','N','2','5'),

("Milemont Memory Foam Pillow, Bamboo Charcoal Memory Foam, Cervical Pillow for Neck Pain, Orthopedic Contour Pillow Support for Back, Stomach, Side Sleepers, Pillow for Sleeping, CertiPUR-US, Standard",'32.99','13','N','2','5'),

("Instant Pot Duo Nova Pressure Cooker 7 in 1, 3 Qt, Best for Beginners",'79.99','13','N','2','4'),

("Finish Line Fiber Grip Carbon Fiber Bicycle Assembly Gel",'7','5','N','3','6'),

("Microsoft  Surface Pro 6 (Intel Core i5, 8GB RAM, 256GB)",'868.89','6','N','8','9'),

("Bell Z20 MIPS Adult Road Bike Helmet",'229.95','18','N','3','6'),

("Dell PR03X E/Port II USB 3.0 Advanced Port Replicator",'89.89','2','N','7','8'),

("Saucony Men's Multi-Pack Bolt Performance Comfort Fit No-Show Socks",'14','2','N','1','2'),

("Logitech Keyboard Folio for iPad mini - Mystic Blue",'79.99','9','Y','7','8');

INSERT INTO orders (

order\_date, customer\_id, order\_status, order\_value

)

VALUES

('2021-01-11',2,'Pending',39.2),

('2021-01-12',4,'Shipped',56),

('2021-01-13',6,'Part\_Shipped',668.79),

('2021-01-14',2,'Delivered',59.97),

('2021-01-15',1,'Part\_Delivered',1048.83),

('2021-01-16',9,'Returned',60.99),

('2021-01-17',3,'Cancelled',17.98),

('2021-01-18',1,'Shipped',35),

('2021-01-19',4,'Delivered',31.68),

('2021-01-20',8,'Delivered',98),

('2021-01-21',4,'Delivered',79.2),

('2021-01-22',7,'Shipped',1496.19),

('2021-01-23',8,'Delivered',319.96),

('2021-01-24',7,'Delivered',129.95),

('2021-01-25',3,'Delivered',1448.62);

INSERT INTO order\_details (

order\_id,product\_id,unit\_price,discount,quantity,item\_status

)

VALUES

(1,4,19.6,DEFAULT,2,'Pending'),

(2,19,14,DEFAULT,4,'Shipped'),

(3,10,332.41,DEFAULT,2,'Pending'),

(3,5,3.97,DEFAULT,1,'Shipped'),

(4,7,19.99,DEFAULT,3,'Delivered'),

(5,19,14,DEFAULT,1,'Shipped'),

(5,17,229.95,DEFAULT,4,'Pending'),

(5,1,19.99,0.98,4,'Shipped'),

(5,3,38.99,DEFAULT,1,'Delivered'),

(6,19,14,DEFAULT,2,'Returned'),

(6,13,32.99,DEFAULT,1,'Returned'),

(7,6,8.99,DEFAULT,2,'Cancelled'),

(8,15,7,DEFAULT,5,'Shipped'),

(9,2,15.84,DEFAULT,2,'Delivered'),

(10,4,19.6,DEFAULT,5,'Delivered'),

(11,2,15.84,DEFAULT,5,'Delivered'),

(12,13,32.99,DEFAULT,1,'Delivered'),

(12,1,19.99,0.98,4,'Delivered'),

(12,10,332.41,DEFAULT,3,'Delivered'),

(12,17,229.95,DEFAULT,1,'Delivered'),

(12,14,79.99,DEFAULT,2,'Shipped'),

(13,14,79.99,DEFAULT,4,'Delivered'),

(14,8,25.99,DEFAULT,5,'Delivered'),

(15,3,38.99,DEFAULT,1,'Delivered'),

(15,14,79.99,DEFAULT,1,'Delivered'),

(15,10,332.41,DEFAULT,4,'Delivered');

INSERT INTO payments (

order\_id, payment\_type, payment\_date, payment\_status

)

VALUES

(1,'Credit Card','2021-01-25 15:35:49','Pending'),

(2,'Debit Card',DEFAULT,'Successful'),

(3,'Credit Card',DEFAULT,'Successful'),

(4,'Credit Card',DEFAULT,'Successful'),

(5,'Credit Card',DEFAULT,'Successful'),

(6,'Check',DEFAULT,'Successful'),

(7,'Bitcoin','2021-01-22 07:11:45','Rejected'),

(8,'Credit Card',DEFAULT,'Successful'),

(9,'Debit Card',DEFAULT,'Successful'),

(10,'Debit Card',DEFAULT,'Successful'),

(11,'Bank Account',DEFAULT,'Successful'),

(12,'Credit Card',DEFAULT,'Successful'),

(13,'Credit Card',DEFAULT,'Successful'),

(14,'Credit Card',DEFAULT,'Successful'),

(15,'Credit Card',DEFAULT,'Successful');

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_