“FAST FASHION” Clothing Store SQL Database Proposal

Team Members: Yaatu Adem, Aislinn Richardson, Wenping Wang

Professor: Ersan Cam

Course: COMP122, Winter 2023

Due Date: April 14, 2023

Problem and Solution Statement:

A clothing store called FAST FASHION has managed to achieve significant success by offering a large range of clothes in various sizes, designs, and colors. However, the store has recognized a significant problem with their business: they lack a reliable system for tracking essential company information, such as sales, inventory management, and customer data. Without a centralized sales data tracking system, the store is unable to make informed decisions about what products to carry and in what quantities. Additionally, the clothing store wishes to provide personalized recommendations based on customers' previous purchases, but they currently lack a mechanism to gather and analyze customer data to make those recommendations.

Designing and putting in place a SQL database system is the answer to these problems. Information regarding customers, products, and transactions is stored and managed centrally via the database system. The clothing store can quickly obtain and analyze data related to customer needs, purchase history, and inventory levels using SQL queries. Knowing which products are popular with customers allows the store to make well-informed choices about inventory management, product promotions, and pricing.

The SQL database will need to integrate with other potential elements that make up the overall system architecture used by FAST FASHION, including point-of-sale (POS), inventory management, and customer relationship management (CRM) systems to give an all-encompassing solution. Effective inventory control is made possible by the POS system, which keeps track of retail transactions and alters inventory data in the database. The database system and the inventory management system are integrated to enable the clothing store to track inventory levels, and schedule orders as necessary while keeping an eye on the availability of merchandise. In addition, the CRM system makes use of information about customers from the SQL database to offer personalized recommendations and promotions, maximize client satisfaction, and encourage long-term client retention so the company can continue to grow.

Use Case Scenario:

John walks into the clothing store and browses the selection of shirts. He finds a blue shirt in his size and decides to purchase it. The sales associate enters the transaction into the store's point-of-sale (POS) system, which automatically updates the SQL database system.

The SQL database system records the transaction details, including the date and time of the purchase, the product ID of the shirt, the price, and John's customer information (such as name and email address). The system also updates the inventory level for the blue shirt in John's size, reducing it by one.

Later that day, the store manager uses the SQL database system to generate a sales report for the day. The report shows the total sales revenue for the day, broken down by product category, as well as the top-selling products and the total number of units sold. The report also shows the inventory levels for each product, allowing the manager to identify any products that are running low and need to be restocked.

The next day, John receives an email from the store with personalized product recommendations based on his purchase history. The email recommends a pair of pants that would match well with the blue shirt he purchased the day before. The personalized recommendation was generated by the SQL database system, which analyzed John's purchase history and identified a pattern in his clothing preferences.

In this scenario, the SQL database system was used to track the sale of a shirt, generate a sales report, and provide a personalized product recommendation to a customer. These are just a few examples of the many ways the SQL database system can be used to improve the efficiency and effectiveness of the clothing store's operations.

Business Requirements for this Project:

- Customers should be able to browse and search for products by category, size, and color.
- Customers should be able to add products to their shopping cart and check out.
- The system should calculate the total cost of an order.
- The system should validate that the customer's payment method is valid and has sufficient funds before processing the order.
- The system should send an order confirmation email to the customer upon successful order placement.
- Customers should be able to submit product reviews and ratings.
- The system should display the average rating for each product based on customer reviews.
- Customers should be able to view their order history and track the status of their current orders.
- The system should send email notifications to customers regarding changes in their order status (e.g. shipped, delayed, delivered).
- The system should keep track of inventory levels and alert staff when a product is running low or out of stock.
- Staff and customers should be able to view and update customer information.
- Staff should be able to manage supplier information and update product cost based on changes in supplier pricing.

Changes Made Throughout the Design Process:

- Narrowed down the scope of the project to only include the sales process without employee information or store location table information.
- Added a payment status column to the order table to facilitate the possibility of the store having unpaid product pre-orders being placed for future products.
- Created a rating system of 1-5 stars and added a reviews table to the database.
- Included supplier information in the product table so profit can be calculated.
- Removed the restraint on who can leave a review so people who received clothes as a gift can still leave a review.
- Most ID's should be generated by the database in sequential order to minimize insertion anomalies

ERD Diagram Design Requirements

- Many products are supplied by one supplier
- Many products are sorted by categories, sizes, and colors
- Customers can place one or many orders, which include one or many different order items
- Each order items have information for each product purchased and quantity purchased
- Each customer can leave one review for each product

CLOTHING STORE ERD DIAGRAM

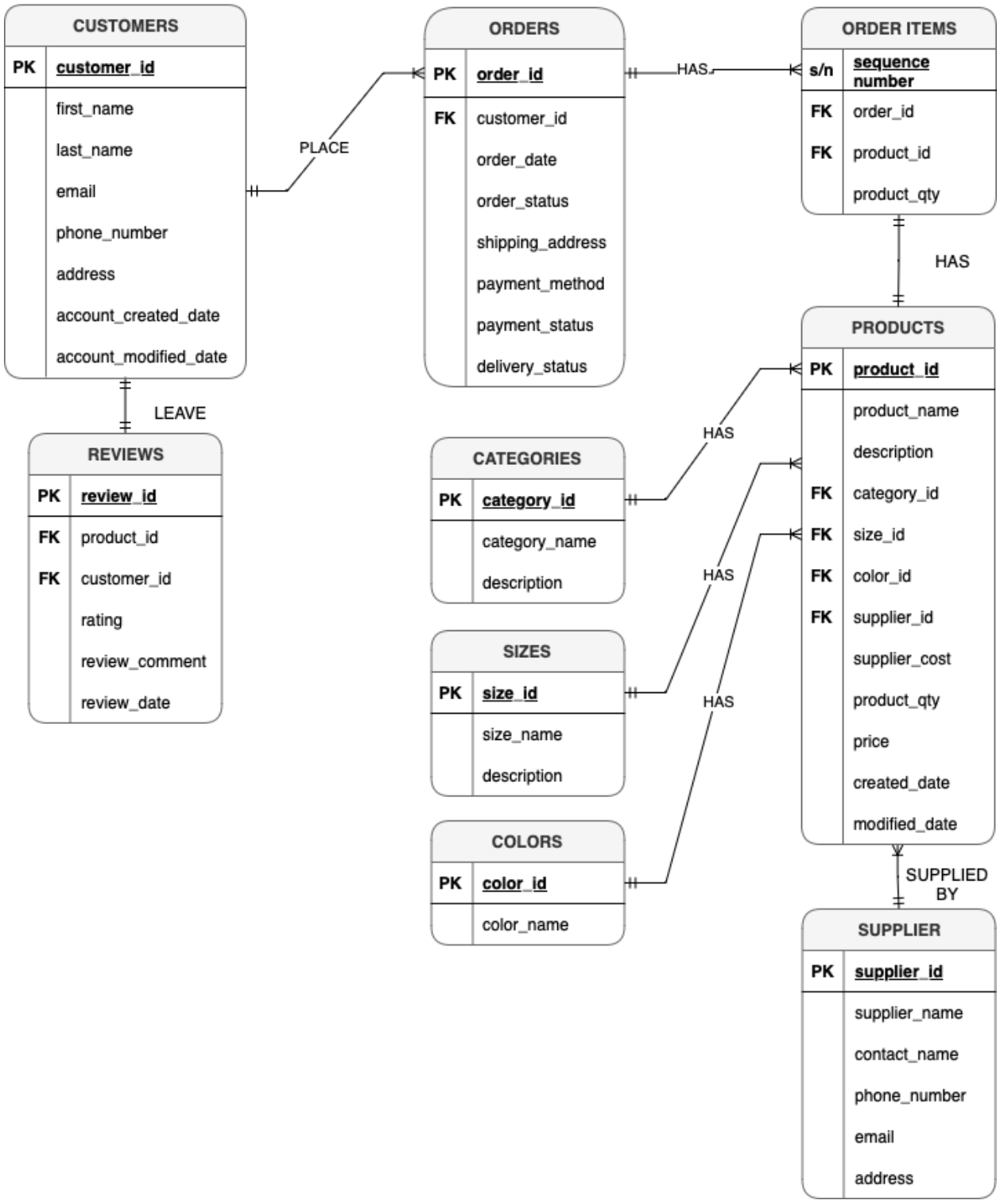

Table Creation Requirements:

1. **Products**
   - Stores individual product information, including size, price, and quantity
   - PRIMARY KEY: product_id
   - FOREIGN KEYS:
     - category_id (categories table)
     - size_id (sizes table)
     - color_id (colors table)
     - supplier_id (suppliers table)
   - Constraints:
     - product_qty >= 0
     - Most fields cannot be NULL
2. **Categories, Sizes, Colors**
   - Stores information to categorize and group together similar items
   - PRIMARY KEY is the respective ID for each table
   - Each ID also has an easily readable name, with optional description
   - Constraints:
     - Names cannot be NULL
3. **Suppliers**
   - Supplier information gives staff the information needed to facilitate restocks of products
   - PRIMARY KEY: supplier_id
   - Constraints:
     - Fields cannot be NULL so staff can have as much contact information as possible
4. **Orders**
   - Stores order information for each customer order placed to the store
   - PRIMARY KEY: order_id
   - FOREIGN KEYS:
     - customer_id (customer table)

- Constraints:
    - Most fields cannot be NULL
    - Order and delivery status, payment method, and payment status have strict options for what they are allowed to be entered as
    - Orders can only start being fulfilled when payment status is paid

5. **Order_Items**
    - Stores product-specific information about each order, including quantity purchased
    - PRIMARY KEY: item_id
    - FOREIGN KEYS:
        - order_id (orders table)
        - product_id (products table)
    - Constraints:
        - Most fields cannot be NULL
        - product_qty >= 1
6. **Customers**
    - Stores basic customer information
    - PRIMARY KEY: customer_id
    - Constraints:
        - Most fields cannot be NULL
        - Email is a unique field, so each email address provided is associated to only one customer account
7. **Reviews**

- Stores information for reviews left by customers
- PRIMARY KEY: review_id
- FOREIGN KEYS:
    - customer_id (customers table)
    - product_id (products table)
- Constraints:
    - Most fields cannot be NULL
    - Rating is a value between 1-5

Table Creation Script:

```
-- Create the "suppliers" table
CREATE TABLE suppliers (
   supplier_id NUMBER(10) GENERATED ALWAYS AS IDENTITY PRIMARY
KEY,
   supplier_name VARCHAR2(255) NOT NULL,
   contact_name VARCHAR2(255) NOT NULL,
   phone_number VARCHAR2(20) NOT NULL,
   email VARCHAR2(255) NOT NULL,
   address VARCHAR2(500) NOT NULL
);

-- Create the "categories" table
CREATE TABLE categories (
   category_id NUMBER(10) GENERATED ALWAYS AS IDENTITY PRIMARY
KEY,
   category_name VARCHAR2(255) NOT NULL UNIQUE,
   description VARCHAR2(500)
);

-- Create the "sizes" table
CREATE TABLE sizes (
   size_id NUMBER(10) GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
   size_name VARCHAR2(255) NOT NULL,
   description VARCHAR2(500)
);

-- Create the "colors" table
CREATE TABLE colors (
```

```
   color_id NUMBER(10) GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
   color_name VARCHAR2(255) NOT NULL UNIQUE
);

-- Create the "products" table
CREATE TABLE products (
   product_id NUMBER(10) GENERATED ALWAYS AS IDENTITY PRIMARY
KEY,
   product_name VARCHAR2(255) NOT NULL,
   description VARCHAR2(500),
   category_id NUMBER(10) NOT NULL,
   size_id NUMBER(10) NOT NULL,
   color_id NUMBER(10) NOT NULL,
   supplier_id NUMBER(10) NOT NULL,
   supplier_cost NUMBER(10,2) NOT NULL,
   product_qty NUMBER(10) NOT NULL CHECK (product_qty >= 0),
   price NUMBER(10,2) NOT NULL,
   created_date TIMESTAMP(6) DEFAULT CURRENT_TIMESTAMP NOT NULL,
   modified_date TIMESTAMP(6),
   CONSTRAINT fk_product_category_id FOREIGN KEY (category_id)
REFERENCES categories (category_id),
   CONSTRAINT fk_product_size_id FOREIGN KEY (size_id)
REFERENCES sizes (size_id),
   CONSTRAINT fk_product_color_id FOREIGN KEY (color_id)
REFERENCES colors (color_id),
   CONSTRAINT fk_product_supplier_id FOREIGN KEY (supplier_id)
REFERENCES suppliers (supplier_id)
);

-- Create the "customers" table
CREATE TABLE customers (
```

```
   customer_id NUMBER(10) GENERATED ALWAYS AS IDENTITY PRIMARY
KEY,
   first_name VARCHAR2(255) NOT NULL,
   last_name VARCHAR2(255) NOT NULL,
   email VARCHAR2(255) NOT NULL UNIQUE,
   phone_number VARCHAR2(20) NOT NULL,
   address VARCHAR2(500) NOT NULL,
   account_created_date TIMESTAMP(6) DEFAULT CURRENT_TIMESTAMP
NOT NULL,
   account_modified_date TIMESTAMP(6)
);

-- Create the "reviews" table
CREATE TABLE reviews (
   review_id NUMBER(10) GENERATED ALWAYS AS IDENTITY PRIMARY
KEY,
   product_id NUMBER(10) NOT NULL,
   customer_id NUMBER(10) NOT NULL,
   rating NUMBER(2) NOT NULL CHECK (rating BETWEEN 1 AND 5),
   review_comment VARCHAR2(1000),
   review_date TIMESTAMP(6) DEFAULT CURRENT_TIMESTAMP NOT NULL,
   CONSTRAINT fk_review_product_id FOREIGN KEY (product_id)
REFERENCES products (product_id),
   CONSTRAINT fk_review_customer_id FOREIGN KEY (customer_id)
REFERENCES customers (customer_id)
);

-- Create the "orders" table
CREATE TABLE orders (
   order_id NUMBER(10) GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
   customer_id NUMBER(10) NOT NULL,
   order_date TIMESTAMP(6) DEFAULT CURRENT_TIMESTAMP NOT NULL,
```

```
   order_status VARCHAR2(50) NOT NULL,
   shipping_address VARCHAR2(500) NOT NULL,
   payment_method VARCHAR2(50) NOT NULL,
   payment_status VARCHAR2(50) NOT NULL,
   delivery_status VARCHAR2(50) NOT NULL,
   CONSTRAINT fk_order_customer_id FOREIGN KEY (customer_id)
REFERENCES customers (customer_id),
   CONSTRAINT chk_order_status CHECK (order_status IN ('Order
Placed', 'In Progress', 'Cancelled', 'Order Fulfilled')),
   CONSTRAINT chk_payment_method CHECK (payment_method IN
('MasterCard', 'Visa', 'AMEX', 'PayPal', 'Debit')),
   CONSTRAINT chk_payment_status CHECK (payment_status IN
('Paid', 'Unpaid')),
   CONSTRAINT chk_delivery_status CHECK (delivery_status IN
('Order Received', 'Order Shipped', 'Order Delayed',
'Delivered')),
   CONSTRAINT chk_delivery_paid CHECK ((payment_status = 'Paid'
AND (delivery_status = 'Order Shipped' OR delivery_status =
'Delivered' OR delivery_status = 'Order Delayed')) OR
(payment_status = 'Unpaid' AND delivery_status = 'Order
Received')),
   CONSTRAINT chk_order_status_paid CHECK ((payment_status =
'Paid' AND (order_status = 'In Progress' OR order_status =
'Cancelled' OR order_status = 'Order Fulfilled')) OR
(payment_status = 'Unpaid' AND (order_status = 'Order Placed' OR
order_status = 'Cancelled')))
);

-- Create the "order_items" table
CREATE TABLE order_items (
   item_id NUMBER(10) GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
   order_id NUMBER(10) NOT NULL,
```

```
   product_id NUMBER(10) NOT NULL,
   product_qty NUMBER(10) NOT NULL CHECK (product_qty >= 1),
   CONSTRAINT fk_items_orders FOREIGN KEY (order_id) REFERENCES
orders (order_id),
   CONSTRAINT fk_items_products FOREIGN KEY (product_id)
REFERENCES products (product_id)
);
```

Data Insertion Script:

```
-- Creates data for "suppliers" table
INSERT INTO suppliers (supplier_name, contact_name,
phone_number, email, address)
VALUES ('ABC Inc.', 'John Smith', '555-555-1234',
'john@abcinc.com', '123 Main St.');
INSERT INTO suppliers (supplier_name, contact_name,
phone_number, email, address)
VALUES ('XYZ Corp.', 'Jane Doe', '555-555-5678',
'jane@xyzcorp.com', '456 Elm St.');
INSERT INTO suppliers (supplier_name, contact_name,
phone_number, email, address)
VALUES ('Acme Co.', 'Bob Johnson', '555-555-9876',
'bob@acmeco.com', '789 Oak St.');
INSERT INTO suppliers (supplier_name, contact_name,
phone_number, email, address)
VALUES ('Smith and Sons', 'Tom Smith', '555-555-5555',
'tom@smithandsons.com', '321 Maple Ave.');
INSERT INTO suppliers (supplier_name, contact_name,
phone_number, email, address)
VALUES ('Jones Inc.', 'Sally Jones', '555-555-2222',
'sally@jonesinc.com', '555 Pine St.');

-- Creates data for "categories" table
INSERT INTO categories (category_name, description)
VALUES ('Shirts', 'Various styles and sizes of shirts');
INSERT INTO categories (category_name, description)
VALUES ('Pants', 'Various styles and sizes of pants');
INSERT INTO categories (category_name, description)
VALUES ('Accessories', 'Belts, hats, scarves, and other
accessories to complement your outfit');
```

```
INSERT INTO categories (category_name, description)
VALUES ('Dresses', 'Various styles and sizes of dresses for any
occasion');
INSERT INTO categories (category_name, description)
VALUES ('Outerwear', 'Jackets, coats, and other outerwear to
keep you warm and stylish');
INSERT INTO categories (category_name, description)
VALUES ('Footwear', 'Shoes and boots for any occasion and
style');

-- Creates data for the "sizes" table
INSERT INTO sizes (size_name, description) VALUES ('Small',
'Fits chest sizes 34-36 inches');
INSERT INTO sizes (size_name, description) VALUES ('Medium',
'Fits chest sizes 38-40 inches');
INSERT INTO sizes (size_name, description) VALUES ('Large',
'Fits chest sizes 42-44 inches');
INSERT INTO sizes (size_name, description) VALUES ('X-Large',
'Fits chest sizes 46-48 inches');

INSERT INTO sizes (size_name, description) VALUES ('28x30',
'Suitable for pants with waist size 28 and length 30');
INSERT INTO sizes (size_name, description) VALUES ('30x32',
'Suitable for pants with waist size 30 and length 32');
INSERT INTO sizes (size_name, description) VALUES ('32x34',
'Suitable for pants with waist size 32 and length 34');
INSERT INTO sizes (size_name, description) VALUES ('34x36',
'Suitable for pants with waist size 34 and length 36');

INSERT INTO sizes (size_name, description) VALUES ('S', 'Small
(fits size 2-4), Dresses');
```

```
INSERT INTO sizes (size_name, description) VALUES ('M', 'Medium
(fits size 6-8), Dresses');
INSERT INTO sizes (size_name, description) VALUES ('L', 'Large
(fits size 10-12), Dresses');
INSERT INTO sizes (size_name, description) VALUES ('XL', 'Extra
Large (fits size 14-16), Dresses');

INSERT INTO sizes (size_name, description) VALUES ('7',
'Suitable for size 7 shoes');
INSERT INTO sizes (size_name, description) VALUES ('8',
'Suitable for size 8 shoes');
INSERT INTO sizes (size_name, description) VALUES ('9',
'Suitable for size 9 shoes');
INSERT INTO sizes (size_name, description) VALUES ('10',
'Suitable for size 10 shoes');

INSERT INTO sizes (size_name, description) VALUES ('O/S',
'Universal size for most accessories; One-size');

-- Creates data for the "colors" table
INSERT INTO colors (color_name) VALUES ('Red');
INSERT INTO colors (color_name) VALUES ('Blue');
INSERT INTO colors (color_name) VALUES ('Green');
INSERT INTO colors (color_name) VALUES ('Yellow');
INSERT INTO colors (color_name) VALUES ('Pink');
INSERT INTO colors (color_name) VALUES ('Purple');
INSERT INTO colors (color_name) VALUES ('Black');
INSERT INTO colors (color_name) VALUES ('White');
INSERT INTO colors (color_name) VALUES ('Gray');
INSERT INTO colors (color_name) VALUES ('Brown');

-- Creates data for the "products" table
```

```
-- Shirts
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Men Classic White Shirt', 'A classic white shirt for
men', 1, 1, 1, 1, 15.99, 2019, 29.99);
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Womens Striped Shirt', 'A striped shirt for women', 1,
2, 3, 2, 12.99, 909, 24.99);
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Mens Button-Down Shirt', 'A casual button-down shirt
for men', 1, 3, 6, 3, 18.99, 2180, 34.99);

-- Pants
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Slim Fit Chino Pants', 'Cotton blend chino pants with
slim fit', 2, 5, 6, 2, 25.00, 1369, 45.99);
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Straight Leg Jeans', 'Straight leg jeans made with
durable denim', 2, 6, 2, 5, 30.00, 334, 59.99);

-- Accesories
```

```
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Wool Hat', 'Warm wool hat', 3, 17, 8, 2, 8.50, 774,
20.00);
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Cashmere Scarf', 'Soft cashmere scarf', 3, 17, 3, 5,
12.00, 1919, 35.00);

-- Dresses
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Summer Floral Dress', 'A beautiful summer dress with a
floral pattern', 4, 9, 3, 2, 25.99, 2635, 49.99);

-- Outerwear
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Bomber Jacket', 'A stylish and warm bomber jacket made
from high-quality materials', 5, 2, 4, 3, 50.00, 632, 89.99);
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Trench Coat', 'A classic trench coat that is perfect
for the fall and winter seasons', 5, 1, 9, 1, 75.00, 1099,
139.99);

-- Shoes
```

```
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Womens Running Shoes', 'Breathable mesh running shoes
with cushioned soles', 6, 13, 3, 2, 35.00, 724, 69.99);
INSERT INTO products (product_name, description, category_id,
size_id, color_id, supplier_id, supplier_cost, product_qty,
price)
VALUES ('Mens Dress Shoes', 'Genuine leather dress shoes with
polished finish', 6, 15, 7, 4, 45.00, 833, 89.99);

-- Creates data for the "customers" table
INSERT INTO customers (first_name, last_name, email,
phone_number, address)
VALUES ('Jordan', 'Smith', 'jordansmith@email.com', '555-123-
4567', '987 Birch Ln.');
INSERT INTO customers (first_name, last_name, email,
phone_number, address)
VALUES ('Taylor', 'Jones', 'taylorjones@email.com', '555-987-
6543', '234 Walnut St.');
INSERT INTO customers (first_name, last_name, email,
phone_number, address)
VALUES ('Alex', 'Johnson', 'alexjohnson@email.com', '555-111-
2222', '555 Cedar Rd.');
INSERT INTO customers (first_name, last_name, email,
phone_number, address)
VALUES ('Chris', 'Martin', 'chrismartin@email.com', '555-777-
8888', '1111 Maple Dr.');
INSERT INTO customers (first_name, last_name, email,
phone_number, address)
VALUES ('Casey', 'Garcia', 'caseygarcia@email.com', '555-444-
3333', '888 Oak St.');
```

```
-- Creates data for the "orders" table"
INSERT INTO orders (customer_id, order_status, shipping_address,
payment_method, payment_status, delivery_status)
VALUES (1, 'Order Fulfilled', '987 Birch Ln.', 'PayPal', 'Paid',
'Delivered');
INSERT INTO orders (customer_id, order_status, shipping_address,
payment_method, payment_status, delivery_status)
VALUES (2, 'Order Fulfilled', '234 Walnut St.', 'Visa', 'Paid',
'Order Shipped');
INSERT INTO orders (customer_id, order_status, shipping_address,
payment_method, payment_status, delivery_status)
VALUES (3, 'Order Placed', '555 Cedar Rd.', 'MasterCard',
'Unpaid', 'Order Received');
INSERT INTO orders (customer_id, order_status, shipping_address,
payment_method, payment_status, delivery_status)
VALUES (4, 'Order Fulfilled', '1111 Maple Dr.', 'AMEX', 'Paid',
'Delivered');
INSERT INTO orders (customer_id, order_status, shipping_address,
payment_method, payment_status, delivery_status)
VALUES (5, 'Order Fulfilled', '888 Oak St.', 'Debit', 'Paid',
'Delivered');
INSERT INTO orders (customer_id, order_status, shipping_address,
payment_method, payment_status, delivery_status)
VALUES (3, 'Order Fulfilled', '555 Cedar Rd.', 'Visa', 'Paid',
'Order Delayed');
INSERT INTO orders (customer_id, order_status, shipping_address,
payment_method, payment_status, delivery_status)
VALUES (4, 'Order Fulfilled', '1111 Maple Dr.', 'Debit', 'Paid',
'Delivered');

-- Create data for "order_items" table
```

```
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (1, 1, 2);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (1, 5, 1);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (1, 9, 3);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (2, 2, 1);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (2, 6, 3);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (3, 3, 2);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (3, 7, 4);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (4, 4, 3);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (5, 5, 2);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (5, 9, 1);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (5, 12, 4);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (6, 6, 1);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (6, 10, 2);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (7, 7, 3);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (7, 11, 1);
INSERT INTO order_items (order_id, product_id, product_qty)
VALUES (7, 12, 1);
```

```
-- -- Create data for "reviews" table
INSERT INTO reviews (product_id, customer_id, rating,
review_comment)
VALUES (1, 1, 4, 'I love this classic white shirt! It fits
perfectly and looks great with jeans or dress pants.');
INSERT INTO reviews (product_id, customer_id, rating,
review_comment)
VALUES (5, 1, 4, 'These jeans fit really well and are very
comfortable. The color is a bit lighter than in the picture, but
it still looks great. Overall, very happy with my purchase!');
INSERT INTO reviews (product_id, customer_id, rating,
review_comment)
VALUES (9, 1, 2, 'Disappointed with the jacket. The zipper broke
after only a few wears.');
INSERT INTO reviews (product_id, customer_id, rating,
review_comment)
VALUES (4, 4, 3, 'Good quality pants, but a bit snug.');
INSERT INTO reviews (product_id, customer_id, rating,
review_comment)
VALUES (5, 5, 5, 'These are my new favorite jeans! They fit
perfectly and are very comfortable.');
INSERT INTO reviews (product_id, customer_id, rating,
review_comment)
VALUES (11, 4, 4, 'I bought these shoes for running and they
have been very comfortable so far. The size was true to fit and
the quality of the shoe is good for the price.');
INSERT INTO reviews (product_id, customer_id, rating,
review_comment)
VALUES (8, 4, 4, 'I love this dress! The fit is perfect and the
floral pattern is so cute.');
```

SELECT Reporting Creation Script With Screenshots:

```
-- SELECT Query #1: Displays a table that shows the total price
and total profit made by the store for each order
SELECT order_items.ORDER_ID, SUM(products.PRICE *
order_items.PRODUCT_QTY) AS TOTAL_PRICE, SUM((products.PRICE -
products.supplier_cost) * order_items.PRODUCT_QTY) AS
TOTAL_PROFIT
FROM order_items
JOIN products ON order_items.PRODUCT_ID = products.PRODUCT_ID
GROUP BY order_items.ORDER_ID
ORDER BY order_items.ORDER_ID;
```

Screenshot of Result:

ORDER_ID	TOTAL_PRICE	TOTAL_PROFIT
1	389.94	177.96
2	84.99	46.5
3	209.98	124
4	137.97	62.97
5	569.93	279.93
6	299.98	141.48
7	264.98	148.98

```
-- SELECT Query #2: Display all items that have an average
review that is 3 stars or less
SELECT categories.CATEGORY_NAME, products.PRODUCT_ID,
products.PRODUCT_NAME, AVG(reviews.RATING) AS AVERAGE_RATING
FROM categories
JOIN products ON categories.CATEGORY_ID = products.CATEGORY_ID
JOIN reviews ON products.PRODUCT_ID = reviews.PRODUCT_ID
WHERE reviews.RATING <= 3
```

```
GROUP BY categories.CATEGORY_NAME, products.PRODUCT_ID,
products.PRODUCT_NAME;
```

Screenshot of Result:

CATEGORY_NAME	PRODUCT_ID	PRODUCT_NAME	AVERAGE_RATING
Outerwear	9	Bomber Jacket	2
Pants	4	Slim Fit Chino Pants	3

```
-- SELECT Query #3: List all products by most to least sold,
including their sizes
SELECT sizes.SIZE_NAME, products.PRODUCT_ID,
products.PRODUCT_NAME, SUM(order_items.PRODUCT_QTY) AS
TOTAL_SOLD
FROM sizes
JOIN products ON sizes.SIZE_ID = products.SIZE_ID
JOIN order_items ON products.PRODUCT_ID = order_items.PRODUCT_ID
GROUP BY sizes.SIZE_NAME, products.PRODUCT_NAME,
products.PRODUCT_ID
ORDER BY SUM(order_items.PRODUCT_QTY) DESC;
```

Screenshot of Result:

	SIZE_NAME	PRODUCT_ID	PRODUCT_NAME	TOTAL_SOLD
1	O/S	7	Cashmere Scarf	7
2	9	12	Mens Dress Shoes	5
3	O/S	6	Wool Hat	4
4	Medium	9	Bomber Jacket	4
5	30x32	5	Straight Leg Jeans	3
6	28x30	4	Slim Fit Chino Pants	3
7	Small	10	Trench Coat	2
8	Large	3	Mens Button-Down Shirt	2
9	Small	1	Men Classic White Shirt	2
10	7	11	Womens Running Shoes	1
11	Medium	2	Womens Striped Shirt	1