SneakerDrop (Phase One)

By Min Thiha Myo and Omer Mir

Abstract:

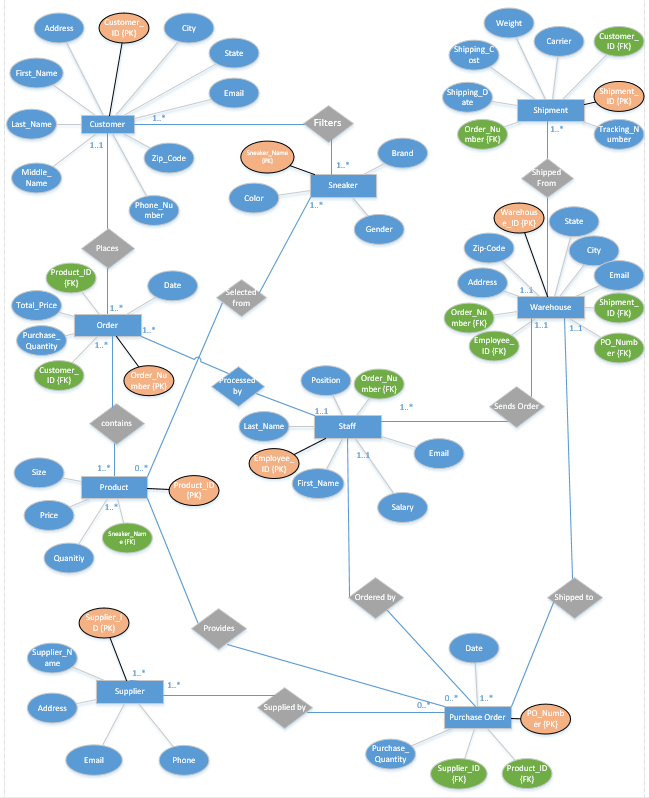
The following is a project which contains the database for an e-commerce company named SneakerDrop, which specializes in the retail of sneakers. This database/site will display the Entity/Relation diagram, the relational model displaying relations to the database design, the database functionality and user interface. This will allow the administration to data-analyze various purchase patterns and maximize sales. We will create the interactions with the database using SQL and we will create the website using HTML/Javascript.

Mission Statement:

The purpose of this database is to generate and maintain data for both the administration and the customers. This database would allow administrators to track and fulfill orders, maintain inventory, updating stock, and create shipments to be delivered. The database will also allow the administration to expedite processing orders from customers and ensure efficient shipping timing and stock availability. The database would allow the customers to search and filter sneakers by a specific size, brand, gender, color, etc. Customers can also view purchasing options and shipping methods effectively.

Mission Objectives

* Maintain data on inventory.
* Allow customers to successfully search inventory and place online orders.
* Allow customers to filter products for better purchase experience.\
* Query customers for shipping options and payment options
* Expedite purchasing experience by sending order details/confirmation by email
* Enable and provide staff with information to update stock and re-stock orders.
* Allow staff to determine which unsold products need to be on promotion/discounted.
* Allow administration to track customer data and buying patterns
* Allow staff to ship orders and track shipments.
* Allow staff to handle the returns and any order disputations



E/R DIAGRAM {Keys: Green (PK) = Primary Key ; Orange (FK) = Foreign Key}

Major user Views

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data | Access Type | Customer | Manager | Staff |
| Staff | Maintain |  | x |  |
| Query |  | x |  |
| Report |  | x |  |
| Supplier | Maintain |  | x |  |
| Query |  | x | x |
| Report |  | x | x |
| Order | Maintain |  |  | x |
| Query | x | x | x |
| Report |  |  | x |
| Purchase Order | Maintain |  | x |  |
| Query |  | x | x |
| Report |  | x | x |
| Product | Maintain |  | x | x |
| Query | x | x |  |
| Report | x |  | x |
| Shipment | Maintain |  |  | x |
| Query | x |  | x |
| Report | x |  | x |

**Use cases:** The two main actors for our database for SneakerDrop are Customer and Staff. The description below lists the main use cases and a short description of each task.

**1. Use Case Name:** Filter the Shoe Specifications

**Actor/User: Customer**

Steps:

1. User drops down the filter box at sneakerdrop.com
2. Filter Options are displayed such as Gender, Size, Color, Material, etc.
3. User chooses an option.
4. Corresponding results will display.

This use case shows a relation between the customer and sneaker and enables customer to choose from various attributes of the entity sneakers. The foreign/primary key involving this user case is sneaker\_name, where the entity customer receives the sneaker\_name as the outcome of the filter.

SQL statement:

* Filtering All “Adidas” brand shoes

**SELECT \***

**FROM product**

**WHERE product\_name =**

**(SELECT product\_name**

**FROM sneakers**

**WHERE brand = ‘Adidas’);**

Joint Query SQL statement:

Entities: (Sneaker and Product)

* Left join query to display all sneakers

**SELECT Sneaker.Sneaker\_Name,**

**Sneaker.Gender,**

**Sneaker.Color,**

**Sneaker.Material,**

**Sneaker.Size,**

**Sneaker.Brand,**

**Sneaker.Style**

**FROM Sneaker**

**LEFT JOIN Product**

**ON Sneaker.Sneaker\_Name = Product.Sneaker\_Name**

**2. Use Case Name:** Sign up / Sign in

**Actor/User: Customer**

Steps:

1. User sign-ups and creates a user ID and password
2. New account is created storing customer’s information such as name, address, email, and phone number.
3. Customer can use his ID to sign-in and check order status and history.
4. Customer can place order by signing in to their account and order efficiently by using the saved customer information.
5. Customer can view their account and update it as needed.

This use case shows a relation between the customer and order. Customer will use their accounts to place the order and check the status as well history of their orders. The entity ORDER can also get the customer information (address, email, phone, etc.) easily from the account.

SQL statement:

* when signing up, such statement occurs:

**INSERT INTO customer**

**VALUES ( ‘4131’, ‘Alex’, ‘Shaw’, ‘Rue’, ‘121 Lane st’, ‘Houston’, ‘Texas’, ‘77000’, ‘7134956498’, ‘**[**akr@gmail.com**](mailto:asr@gmail.com)**’ );**

Joint Query SQL statement:

Entities: (Customer and Order)

* Inner join query when customer checks order history.

**SELECT Customer.First\_Name,**

**Customer.Last\_Name**

**FROM Customer**

**INNER JOIN Order**

**ON Customer.Customer\_ID = Order.Customer\_ID**

**WHERE Customer.Customer\_ID = ‘5’;**

**3. Use Case Name:** Placing an Order

**Actor/User: Customer**

Steps:

1. Used selects the desired item.
2. User checks if the Color/Size is in stock. Price will also be displayed.
3. User places the item in cart.
4. Either user signs in to his account or enters his Shipping Address.
5. An order number is generated.
6. User submits the order.

This use case shows a relation between the customer and the order. Customer can sign-in to their account and place the order. This creates the primary key, order\_number, for the entity Order, and a number is assigned for the order\_number. The order contains product\_id which defines the specific items in the order and is a primary key of Product and a foreign key of Order.

SQL statement:

* New order added.

**INSERT INTO order**

**VALUES ( ‘11231’, ‘10152018’, ‘512’, ‘39.99’, ‘4131’ );**

Joint Query SQL statement:

Entities: (Customer and Order)

* Right join query to check the orders which contains products priced over $500.

**SELECT Order.Order\_Number,**

**Order.Price,**

**Order.Date**

**FROM Order**

**RIGHT JOIN Product**

**ON Order.Product\_ID = Product.Product\_ID**

**WHERE Product.Price > ‘500’;**

**4. Use Case Name:** Processing Order

**Actor/User: Staff**

Steps:

1. Staff receives order from the customer with the order number.
2. Staff processes the payment from the customer.
3. Staff sends order to warehouse to be packaged and shipped.

This use case shows a relation between the staff and order. Staff receives the order\_number and processes it along with customer information. Then staff sends order\_number to warehouse for packaging and shipping to the customer. Order\_number is a foreign key for staff and is retrieved from the entity Order, where it is a primary key.

SQL statement:

* New warehouse ID

**INSERT INTO Warehouse ( Warehouse\_ID, Address, City, State, Zip\_Code, Phone\_Number, Email)**

**VALUES ( ‘WI5632’, ‘123 Street’. ‘Houston’, ‘Texas’ , ‘77000’, ‘8326597845”, ‘xyz@tcsc.com’ );**

Joint Query SQL statement:

Entities: (Customer and Order); Right Join Query

* Query to check which staff members are authorized to send orders to the warehouse.

**SELECT Staff.Employee\_ID,**

**Staff.First\_Name,**

**Staff.LastName**

**FROM Staff**

**INNER JOIN Warehouse**

**ON Staff.Employee\_ID = Warehouse.Employee\_ID**

**5. Use Case Name:** Shipping Order

**Actor/User: Staff**

Steps:

1. Warehouse receives the order from staff.
2. Warehouse packages the order to be shipped
3. Warehouse chooses appropriate carrier, enters weight and address, and creates shipping label.
4. Warehouse ships the order to the address that customer provides.
5. Customer receives the order.

This use case shows a relation between the warehouse and shipment. After receiving order\_number (Primary key in Order) from the staff, warehouse selects the products/items using order\_number, package it and sends order for shipment. Shipment generates a shipment\_ID (primary key in Shipment) and tracking number and ships the order to the customer using the appropriate carrier.

SQL statement:

* If the product weighs more than 20 pounds, staff uses UPS as a carrier.

**SELECT \***

**FROM Shipment**

**WHERE Shipment.Weight >= 20;**

Joint Query SQL statement:

Entities: (Customer and Order)

* Full join query to check the order\_Number shipped by the Carrier service used by the Shipment.

**SELECT Warehouse.Order\_Number,**

**Warehouse.Shipment\_ID**

**FROM Warehouse**

**FULL JOIN Shipment**

**ON Warehouse.Shipment\_ID = Shipment.Shipment\_ID**

**WHERE Carrier = ‘UPS’;**

**6. Use Case Name:** Maintain Inventory

**Actor/User: Staff**

Steps:

1. Staff keeps track on the products in stock.
2. Staff analyzes database for the amount of items left in a particular product.
3. If amount of item is less than 3, staff flags it.
4. Staff prepares an inventory list of items to be restocked.

This use case shows a relation between the staff and product. Staff checks the inventory list (quantity) of items in product to be restocked to later preparesa Purchase\_order that will have Product\_Id as foreign keys. Staff also provides Purchase\_order with product quantity to maintain the inventory.

SQL statement:

* To show the products that are in quantity less than 3.

**SELECT Product.Product\_ID,**

**Product.Quantity,**

**Product.Sneaker\_Name**

**FROM Product**

**WHERE Quantity < 3;**

Joint Query SQL statement:

Entities: (Customer and Order)

* Inner join query to show the product (sneaker name) with the most number of purchase orders.

**SELECT Product.Product\_Id**

**Product.Sneaker\_Name**

**FROM Product**

**INNER JOIN Purchase\_Order**

**ON Product.Product\_ID = Purchase\_Order. Product\_ID**

**WHERE Product.Sneaker\_Name = ‘Adidas NMD C2’;**

**7. Use Case Name :** Create purchase order

**Actor/User: Staff**

Steps:

1. Staff checks the list and items to be restocked.
2. Staff creates the purchase order.
3. Staff sends it to the supplier.
4. The supplier fulfills the order and sends to the warehouse.

This use case shows a relation between the staff and warehouse. Staff creates a Puchase\_order and assigns a PO number. Warehouse will have the PO number as a foreign key that will display the product\_Id, Supplier\_Id, and product quantity.

SQL statement:

* Sort the staff by the distinct first name.

**SELECT DISTINCT (First\_Name)**

**FROM Staff;**

Joint Query SQL statement:

Entities: (Customer and Order)

* Left join query to rank the staff who reorders the most quantity of sneakers.

**SELECT Staff.Employee\_ID,**

**Staff.First\_Name,**

**Purchase\_Order.Purchase\_Quantity**

**FROM Staff**

**LEFT JOIN Purchase.Order**

**ON Staff.PO\_Number = Purchase\_Order.PO\_Number**

**ORDER BY Purchase\_Quantity DESC;**

**8. Use Case Name :** Fulfilling Purchase Order

**Actor/User: Supplier/Staff**

Steps:

1. Supplier receives the purchase order from staff.
2. Supplier analyzes the product\_ID on the PO.
3. Supplier packages the PO and creates shipping label to warehouse
4. Supplier sends the PO to the warehouse.

This use case show a relation between the supplier and the staff. The supplier has the PO number as a foreign key.

SQL statement:

* Count the number of Suppliers the company has.

**SELECT Count(Supplier\_ID)**

**FROM Supplier;**

Joint Query SQL statement:

Entities: (Customer and Order); Inner Join Query

* To rank the staff who reorders the most quantity of sneakers.

**SELECT Supplier.Supplier\_ID,**

**Supplier.Supplier\_Name,**

**Purchase\_Order.PO\_Number**

**FROM Supplier, Purchase\_Order**

**INNER JOIN Purchase.Order**

**ON Supplier.Supplier\_ID = Purchase\_Order.Supplier\_ID**

**WHERE Supplier\_Name = ‘Nike Official’;**

**INSERT, DELETE, UPDATE, AND AGGREGATE QUERIES**

**1. CUSTOMER:**

* **INSERT:**

**INSERT INTO Customer (Customer\_ID, First\_Name, Middle\_Name, Last\_Name, Address, City, State, Zip\_Code, Phone\_Number, Email)**

**VALUES ( ‘4131’, ‘Alex’, ‘Shaw’, ‘Rue’, ‘121 Lane st’, ‘Houston’, ‘Texas’, ‘77000’, ‘7134956498’, ‘**[**asr@gmail.com**](mailto:asr@gmail.com)**’ );**

* **DELETE:**

**DELETE**

**FROM Customer**

**WHERE Customer\_ID = ‘0000’;**

* **UPDATE:**

**UPDATE Customer**

**SET email = ‘**[**abc@yahoo.com**](mailto:abc@yahoo.com)**’**

**WHERE customer\_ID = ‘0001’;**

* **AGGREGATE: (Displaying total number of customers in a company)**

**SELECT COUNT(\*) AS Total\_Customers**

**FROM Customer;**

**2. STAFF:**

* **INSERT:**

**INSERT INTO Staff ( Employee\_ID, First\_Name, Last\_Name, Position, Salary, Email)**

**VALUES ( ‘B22’, ‘Phil’, ‘Grande’, ‘Manager’, ‘36000’, ‘**[**philgran@gmail.com**](mailto:philgran@gmail.com)**’);**

* **DELETE:**

**DELETE**

**FROM Staff**

**WHERE Staff\_ID = ‘B21’;**

* **UPDATE:**

**UPDATE Staff**

**SET Salary = ‘39000’**

**WHERE Employee\_ID = ‘B21’;**

* **AGGREGATE: (Displaying minimum, maximum, and average staff salary)**

**SELECT MIN (Salary) AS Min\_Salary,**

**MAN (Salary) AS Max\_Salary,**

**AVG (Salary) AS Avg\_Salary**

**FROM Staff;**

**3. SNEAKER:**

* **INSERT:**

**INSERT INTO Sneaker (Sneaker\_Name, Color, Gender, Material, Style, Brand, Size)**

**VALUES ( ‘Adidas NMD CS2’, ‘Black’, ‘M’, ‘Prime Knit’, ‘Running’, ‘Adidas’, ‘9.5’);**

* **DELETE:**

**DELETE**

**FROM Sneaker**

**WHERE Sneaker\_Name = ‘Common Projects Achilles’;**

* **UPDATE:**

**UPDATE Sneaker**

**SET Color = ‘light blue’**

**WHERE sneaker\_name = ‘Nike Air Force 1’;**

* **AGGREGATE: (Displaying total number of men shoes only)**

**SELECT COUNT (\*) AS TOTAL\_MEN\_SHOES**

**FROM Sneaker**

**WHERE Gender = ‘M’;**

**4. ORDER:**

* **INSERT:**

**INSERT INTO ORDER ( Order\_Number, Date, Price )**

**VALUES ( ‘1001’, ‘10202018’, ‘49’ );**

* **DELETE:**

**DELETE**

**FROM Order**

**WHERE Order\_Number = ‘1002’;**

* **UPDATE:**

**UPDATE Order**

**SET Price = ‘65’**

**WHERE Order\_Number = ‘1001’;**

* **AGGREGATE: (Displaying total number of orders placed in Oct 2018)**

**SELECT COUNT (\*) AS Total\_Orders\_Oct**

**FROM Order**

**Where Date BETWEEN ‘1-Oct-18’ AND ‘31-Oct-18’;**

**5. PRODUCT:**

* **INSERT:**

**INSERT INTO Product ( Product\_ID, Quantity, Price )**

**VALUES ( ‘P86’, ‘150’, ‘60’ );**

* **DELETE:**

**DELETE**

**FROM Product**

**WHERE Product\_ID = ‘P95’;**

* **UPDATE:**

**UPDATE Product**

**SET Price = ‘65’**

**WHERE Product\_ID = ‘P86’;**

* **AGGREGATE: (Displaying the highest price product)**

**SELECT Max (Price) AS Highest\_Price**

**FROM Product;**

**6. SUPPLIER:**

* **INSERT:**

**INSERT INTO Supplier ( Supplier\_ID, Supplier\_Name, Address, Email, Phone )**

**VALUES ( ‘S8905’, ‘Adidas’, ‘123 Street’, ‘**[**123@five.com**](mailto:123@five.com)**’, ‘7135698956’ );**

* **DELETE:**

**DELETE**

**FROM Supplier**

**WHERE Supplier\_ID = ‘S5645’;**

* **UPDATE:**

**UPDATE Supplier**

**SET Email = ‘**[**456@eight.com**](mailto:456@eight.com)**’**

**WHERE Supplier\_ID = ‘S5640’;**

* **AGGREGATE: (Displaying the total number of suppliers)**

**SELECT COUNT (\*) AS Total\_Suppliers**

**FROM Supplier;**

**7. PURCHASE ORDER:**

* **INSERT:**

**INSERT INTO Purchase\_Order ( PO\_Number, Purchase\_Quantity )**

**VALUES ( ‘PO255’, ‘20’ );**

* **DELETE:**

**DELETE**

**FROM Purchase\_Order**

**WHERE PO\_Number = ‘PO921’;**

* **UPDATE:**

**UPDATE Purchase\_Order**

**SET Purchase\_Quantity = ‘50’**

**WHERE PO\_Number = ‘PO255’;**

* **AGGREGATE: (finding how many purchase orders created in Oct ‘18)**

**SELECT SUM (Purchase\_Quantity) As Total\_Shoe\_Purchase**

**FROM Purchase\_Order**

**Where Date BETWEEN ‘1-Oct-18’ AND ‘7-Oct-18’;**

**8. WAREHOUSE:**

* **INSERT:**

**INSERT INTO Warehouse ( Warehouse\_ID, Address, City, State, Zip\_Code, Phone\_Number, Email)**

**VALUES ( ‘WI5632’, ‘123 Street’. ‘Houston’, ‘Texas’ , ‘77000’, ‘8326597845”, ‘xyz@tcsc.com’ );**

* **DELETE:**

**DELETE**

**FROM Warehouse**

**WHERE Warehouse\_ID = “WI456’;**

* **UPDATE:**

**UPDATE Warehouse**

**SET email = ‘123@uci.com’**

**WHERE Warehouse\_ID = ‘WI4563’;**

* **AGGREGATE: (Displaying total number of purchase orders)**

**SELECT COUNT (\*) Purchase\_Order**

**FROM Warehouse**

**WHERE Warehouse\_ID = ‘WI562’;**

**9. SHIPMENT:**

* **INSERT:**

**INSERT INTO Shipment ( Shipment\_ID, Shipping\_Date, Shipping\_Cost, Weight, Carrier, Tracking\_Number )**

**VALUES ( ‘SI8956’, ‘10252018’, ‘9.99’, ‘5’, ‘UPS’, ‘1Z45659798564’ );**

* **DELETE:**

**DELETE**

**FROM Shipment**

**WHERE Shipment\_ID = ‘SIO921’;**

* **UPDATE:**

**UPDATE Shipment**

**SET Tracking\_Number = ‘1Z45698633631’**

**WHERE Shipping\_ID = ‘SI8956’;**

* **AGGREGATE: (displaying the total shipments)**

**SELECT SUM ( Shipping\_Cost ) AS Total\_Shipments**

**FROM Shipment;**

**Database Prototype**

**In Phase Two, we have accomplished our conceptual data model and have analyzed our requirements for the data model. We have used Visio 2019 to populate the E/R diagrams: it displays our nine entities with its Primary Keys (PK) and Foreign Keys (FK). In addition, it displays the relationships between the each entities along with its multiplicities. Furthermore, our relational model is also populated used Visio 2019. The Primary Keys (PK) and Foreign Keys (FK) are titled on each table and its relationships are connected along with its multiplicities. Visio 2019 has been a great help since it allows the diagrams to be manipulated the way that is both visually distinct and accessible. In terms of physical data, for this project we will be fabricating our data and will manually populate our data into the data into the database system. We will be using Microsoft SQL Server for the physical implementation of data as it readily available and very efficient. We will be creating the Databases and its subsequents tables using the SQL Server and will be manually inserting the data into the table.**

**SneakerDrop: Project Timetable**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SneakerDrop Formation & General Tasks (08/20/2018)** | | | | |
| **TASKS** | | | | |
| **Member** | **Selecting application of data** | **Formation of Company and Naming** | **In-depth description of database structure** | **Review of the company and database structure** |
| **Min Thiha Myo** |  |  |  |  |
| **Omer Mir** |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Phase One ( 08/25/2018 - 09/16/2018)** | | | | | | | |
| **TASKS** | | | | | | | |
| **Member** | **Abstract** | **Mission Statement** | **Mission Objectives** | **Major User Views** | **Use Cases** | **ER Diagram** | **Review** |
| **Min Thiha Myo** |  |  |  |  |  |  |  |
| **Omer Mir** |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Phase Two ( 09/17/2018 - 10/21/2018)** | | | | | | | | | | |
| **TASKS** | | | | | | | | | | |
| **Member** | **Complete E/R Diagram** | **Relational Model** | **Use Cases (Update)** | **Use Cases Statements** | **Insert, Delete, Update** | **Joint Queries** | **Use Case Realization** | **Database Prototype** | **Project Timetable** | **Review** |
| **Min Thiha Myo** |  |  |  |  |  |  |  |  |  |  |
| **Omer Mir** |  |  |  |  |  |  |  |  |  |  |

* To show the staff ranked by the highest salary. (FOR LATER)

**SELECT Staff.Employee\_ID,**

**Staff.First\_Name,**

**Staff.Salary,**

**Purchase\_Order.PO\_Number**

**FROM Staff, Purchase\_Order**

**LEFT JOIN Purchase.Order**

**ON Staff.Employee\_ID = Purchase\_Order.Employee\_ID,**

**ORDER BY salary DESC;**