IT 450 Course Project: E-Commerce Database

Destiny Hale
Old Dominion University

IT 450

Professor Zheng

4/20/25

First, I created three tables: Products, Customers, and Orders for the "E-commerce" database.

1. **Products**
   The query shows the creation of the 'Products' table within the database 'E-Commerce. The four fields include: *product_id, product_name, product_price,* and *quantity_in_stock.* The *product_id* represents the unique code associated with the specific product. This is put in **INT** format, and will also be the primary key of the table. The *product_name* identifies the name of the product, this is put in the **CHAR** format. The *product_price* gives the price of the product, this will be put in **NUMERIC** format. Lastly, *quantity_in_stock* describes how many products are left in stock. This is put in **INT** format. This table displays what products are in the store and how many are left.

```
CREATE TABLE Products (
product_id INT PRIMARY KEY,
product_name VARCHAR(50) NOT NULL,
product_price NUMERIC(8,2),
quantity_in_stock INT);
```

2. **Customers**
   The query shows the 'Customers' table creation within the 'E-Commerce' database. The seven fields include: *customer_id, customer_name, customer_address, customer_state, customer_zipcode,* and *phone_number.* The *customer_id* is the unique code given to the customer used to identify them. This is put in the **INT** format and will be the primary key for the table. The *customer_name* field gives the full name of the customer, which **VARCHAR** is used to fit the names of each customer. The fields *customer_address, customer_state,* and *customer_zipcode* give where the customer lives, which **VARCHAR and CHAR are used**. Lastly, *phone_number* is used to contact the customer. This table displays the subscribed customers, their form of contact, and their location.

```
1  CREATE TABLE CUSTOMERS (
2  customer_id INT NOT NULL PRIMARY KEY,
3  customer_name VARCHAR(50) NOT NULL,
4  customer_address VARCHAR(30) NOT NULL,
5  customer_city VARCHAR(30) NOT NULL,
6  customer_state CHAR(2) NOT NULL,
7  customer_zipcode CHAR(5) NOT NULL,
8  phone_number CHAR(14) NOT NULL);
```

3. **Orders**
   The query shows the creation of the Orders table with the following fields: *order_id, customer_id, product_id, quantity, and order_date. Order_id* represents the unique code associated with the specific order made by the customer. This will be the primary key for the table and put **INT** format. Followed by the *order_id* is the *customer_id* that will correspond to the order that they have made. Also corresponding to both fields, the *product_id* will associate with the order number. Both *product_id* and *customer_id* are the foreign keys for this table and are in **INT** format. The *quantity* field tells how many of a specific product the customer is purchasing. Lastly, the *order_date* tells when the customer made the order. This uses the **DATE** datatype. This table displays the order numbers of each customer with their ID numbers followed by the products and the date they purchased them

```
CREATE TABLE Orders (
order_id INT PRIMARY KEY,
customer_id INT NOT NULL,
product_id INT NOT NULL,
quantity INT NOT NULL,
order_date DATE NOT NULL,
FOREIGN KEY (customer_id) REFERENCES Customers(customers_id),
FOREIGN KEY (product_id) REFERENCES Products(product_id));
```

Shown here are the 10 sample examples and the results from each table:

1. **Products**

Using the following input:

```
INSERT INTO Products VALUES (123, 'Laptops', 1500.50, 180);
```

product_id	product_name	product_price	quantity_in_stock
101	Wired Headphones	35.99	30
102	Cameras	650.99	140
103	Mechanical Keyboard	95.30	86
123	Laptops	1,500.50	180
234	Bluetooth Headphones	179.00	90
345	Smartphones	810.00	160
456	Tablets	1,000.80	70
678	Desktop Computers	1,300.10	100
789	Wireless Mouse	85.99	50
910	USBs	65.43	20

2. **Customers**

Using the following input:

```
INSERT INTO Customers VALUES (18, 'Zenith Ryan', '723 Pinecone Rd.', 'Portsmouth', 'VA', '23701', '757-970-1234');
```

customer_id	customer_name	customer_address	customer_city	customer_state	customer_zipcode	phone_number
18	Zenith Ryan	723 Pinecone Rd.	Portsmouth	VA	23701	757-970-1234
29	Jaden Skyler	1345 Main St.	Ansonville	NC	28007	908-555-0101
30	Phoenix Blaze	1674 Treestump Ln.	Fulton County	GA	31131	404-672-5555
41	Angela Brown	505 Walter St.	Atlanta	GA	30381	943-123-5678
52	Nexus Williams	1000 Isabelle Ln.	Virginia Beach	VA	23456	757-010-9010
63	Skai Galaxy	5680 St. Maria Blvd.	Jacksonville	FL	32225	904-541-2233
74	Chris Arson	4020 Winderbrown Dr.	Portsmouth	VA	23703	757-746-4770
85	Alicia Borealis	1012 St. Lawrence Blvd.	Charleston	SC	29413	843-654-2746
96	Paige Lockette	345 Sneed Rd.	Putnam County	FL	32007	727-555-8955
98	Lizzy Fox	123 Seasame St.	Clarendon County	SC	29001	843-891-1011

3. **Orders**

Using the following input:

```
INSERT INTO Orders VALUES (5172, 18, 123, 1, '2023-07-22');
```

order_id	customer_id	product_id	quantity	order_date
2,113	74	910	4	2023-04-11
3,456	85	101	2	2022-07-12
4,567	52	678	1	2025-02-14
5,172	18	123	1	2023-07-22
6,114	29	234	2	2024-10-22
7,910	96	102	2	2025-09-11
8,113	63	789	2	2024-04-17
8,779	41	456	1	2025-01-31
8,990	98	103	1	2025-04-18
9,172	30	345	1	2025-08-14

These tables are considered 3NF because they follow both requirements of 1NF and 2NF. For tables to be considered 3NF, they first need to go through the requirement of 1NF. 1NF requires that all key attributes must be defined, in each table specific fields such as *order_id, customer_id, product_id,* etc. are uniquely identified and defined. These also included primary and foreign keys such as *product_id (primary and foreign), customer_id (both primary and foreign),* and *order_id (primary)*. There are also no repeating groups in each table. Since the tables have met the requirements of 1NF, this automatically transitions to 2NF because there are no partial dependencies. Partial dependency exists when there is a functional dependency, the determinant is only part of the primary key. In each table, the primary keys determine the quantity, order dates, addresses, customer names, etc. Each non-key attribute relies on the primary key, therefore there are no partial dependencies. Lastly, since the tables meet the requirements of both 1NF and 2NF, the tables are now in 3NF. 3NF tables do not have transparent dependencies. In transparent dependency, attributes are dependent on each other and aren't a part of the primary key. However, in these tables, all non-key attributes are dependent on their primary keys.

Secondly, I created three query screens for each table.

The first query screen shows the products that are under $100.

```sql
SELECT product_id, product_name, product_price
#Table: Products
FROM products
#Table: Products
WHERE product_price < 100
# '<' is the indicator for 'less than', in this case it's gathering products less than $100.
```

products (4r × 3c)

product_id	product_name	product_price
101	Wired Headphones	35.99
103	Mechanical Keyboard	95.30
789	Wireless Mouse	85.99
910	USBs	65.43

The second query screen shows the customer id, address, and zip code from the state that they live in. For example, this query screen shows the customers who live in Virginia.

```sql
SELECT customers.customer_id, customers.customer_name, customers.customer_address, customers.customer_zipcode
    #Table: Customers, fields: customers_id, customers_name, customers_address, customers_zipcode
FROM customers
    #Table: Customers
WHERE customers.customer_state= 'VA';
#This grabs the customers that lives in the state of Virginia
```

customers (3r × 4c)

customer_id	customer_name	customer_address	customer_zipcode
18	Zenith Ryan	723 Pinecone Rd.	23701
52	Nexus Williams	1000 Isabelle Ln.	23456
74	Chris Arson	4020 Winderbrown Dr.	23703

The third query screen shows the customer's address information (which would be the shipping address) and the product they ordered by using the order id.

```
SELECT orders.order_id, products.product_name, orders.quantity, customers.customer_name, customers.customer_address, customers.customer_state,
Orders.order_date
#Tables: Orders, Customers, and Products
#Fields: order_id, product_name, quantity, customer_name, customer_address,customer_state,customer_zipcode, and order_date.
FROM  customers
#Table: Customers to gather the name associated with the order number
JOIN Orders ON Customers.customer_id = Orders.customer_id
#The join of Orders and Customers
JOIN Products ON Orders.product_id = Products.product_id
#The join of Product and Orders
WHERE Orders.order_id = 5172;
#The entry for the order id
```

customers (1r × 8c)

order_id	product_name	quantity	customer_name	customer_address	customer_state	customer_zipcode	order_date
5,172	Laptops	1	Zenith Ryan	723 Pinecone Rd.	VA	23701	2023-07-22

This application's main idea is to allow companies to manage products and track orders made by customers in their e-commerce technology stores. Companies can access a variety of functions including the retrieval of detailed information about their customers from the ‘Customers’ table, product information from the ‘Products’ table, and order information from the ‘Orders’ table. The query screens will make searches and navigation easier by retrieving products that are under or over a specific price range, customers that live in a specific state, and order ids associated with the customer, along with the quantity of a specific product they ordered and the purchase date. The first query will help companies in separating and finding low-priced products from high-priced products or vice versa. The second query would provide enhanced shipping logistics and regional marketing. Lastly, the third query will help track the purchase history of orders made by customers, improving marketing configuration. This app will ensure efficiency in data retrieval, making business and shipping operations fast and efficient.