

Online Shopping System

Team Name: Data Driven Shoppers

Mounika Pasupuleti
Engineering Science Data Science
University at Buffalo
UBITName: mpasupul
mpasupul@buffalo.edu

Sahithya Arveti Nagaraju
Engineering Science Data Science
University at Buffalo
UBITName: sarvetin
sarvetin@buffalo.edu

Sushmitha Manjunatha
Engineering Science Data Science
University at Buffalo
UBITName: smanjuna
smanjuna@buffalo.edu

Abstract—This project focuses on developing a complete database system for online shopping that discharges the complexity of contemporary electronic commerce. With the ever-growing need for easy to access and effortless shopping experience, online shopping has become very popular. But, whilst online marketplaces have become dependent on vast amounts of data for user and product maintenance, and transactions and ‘data junkies’ exist within, most companies utilize a simple database that does not fulfill the volume of data. The goal of this project, therefore, is to solve the problems of online traders through the development of a safe, flexible and equally effective database management system that ensures proper performance, maximization of customer satisfaction, and achievement of business goals. To this project is assigned the dataset that simulates the online retail market, which has tables about users, products, shopping carts, orders and reviews.

I. INTRODUCTION

Need of database instead of excel: A database is more suitable than Excel for managing an online shopping platform due to its scalability, ability to handle large datasets, and ensure data integrity. It supports real-time multi-user access, automates key processes like stock management and order tracking, and offers enhanced security features such as encryption. Databases also provide efficient data querying, reporting, and error handling, making them ideal for managing complex e-commerce operations compared to Excel’s limitations.

Shopping from marketplaces has transformed the way retail operates, with customers able to easily view and purchase items without leaving their homes. However, the growth of an online shopping site will largely depend on the database management system responsible for handling diverse information ranging from product inventory to related transactions. Without a structured database system, businesses are bound to encounter problems such as mismanagement of stock, inefficient order processing, and lack of security of company and client data, which will negatively affect customer relation and revenue.

Database issues can cause problems in managing user data, tracking orders, and controlling stock levels, while weak security increases the risk of data loss. A specialized, secure database is needed for efficient e-commerce operations.

In this project, we utilized the Faker Python module [1] to generate data for our tables. This allowed us to create realistic

datasets for user accounts, items for sale, order fulfillment, and payments, ensuring a comprehensive simulation of an online shopping experience.

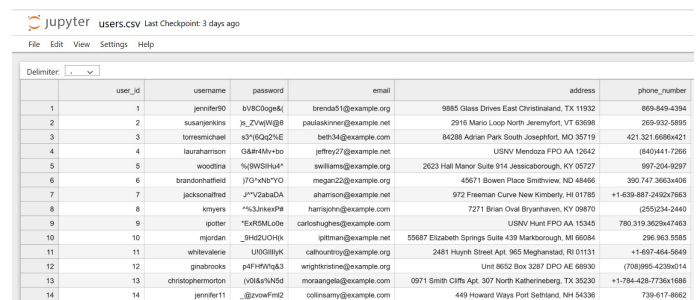
```
# Initialize Faker
fake = Faker()

# Step 1: Generate Users
num_users = 3500
users = []

for user_id in range(1, num_users + 1):
    username = fake.user_name()
    password = fake.password()
    email = fake.email()
    address = fake.address()
    phone_number = fake.phone_number()
    users.append({
        'user_id': user_id,
        'username': username,
        'password': password,
        'email': email,
        'address': address,
        'phone_number': phone_number
    })

users_df = pd.DataFrame(users)
users_df.to_csv("users.csv", index=False)
```

Fig. 1. Tables creation using faker



	user_id	username	password	email	address	phone_number
1	1	jenifer10	WRC3qgk4	lbrend51@example.org	9885 Glass Drive East Christland, TX 11932	889-849-4394
2	2	tsawyers	h_2V4p1g8E	paalakoner@example.net	2916 Mario Loop North Jermyfort, VT 63698	269-932-5895
3	3	torresricha	s5Y5Q2uLE	beth34@example.com	84288 Adrian Park South Josephfort, MO 35719	421-321-6966x421
4	4	lauraharrison	G8H4M+bo	jeffrey27@example.net	USNV Mendoca FPO AA 12642	(840)441-7266
5	5	woodrina	%BWSH4u*	swilliams@example.org	2623 Hall Manor Suite 914 Jessicaborough, KY 05727	997-204-9297
6	6	brandonathfield	J7Q*4N2*YO	megan22@example.com	45671 Bowen Place Smithview, ND 48466	390-747-3663x406
7	7	jacksonathed	J**V2abaDA	aharrison@example.net	972 Freeman Curve New Kimberly, HI 01785	+1-639-887-2492x7663
8	8	kmeyers	*%3UnxasPe	hamisjohn@example.com	7271 Brian Oval Bryanhaven, KY 09870	(255)234-2440
9	9	jpotter	*EXR5MLde	carltohughes@example.com	USNV Hunt FPO AA 15345	780-319-3629x47483
10	10	mjordan	_SH4ZUChH	ottman@example.net	55687 Elizabeth Springs Suite 439 Markborough, MI 66084	296-963-5585
11	11	whitewiers	URQ5MkK	celiaouring@example.org	2481 Haysle Street Apt. 965 Hightmanland, RE 01131	+1-697-464-5649
12	12	gratebrooks	p4FV5h9a3	enightmings@example.org	Unit 8052 Box 5287 DPO AE 68930	(708)995-4259x014
13	13	christophermorton	v0d4sUNd5	mosaangels@example.com	0971 Smith Cliffy Apt. 307 North Kathleenberg, TX 35230	+1-784-426-7736x1886
14	14	jenifer11	@zwofm2	collinsamy@example.com	449 Howard Ways Port Setthland, NH 54336	739-617-8662

Fig. 2. Output

II. BACKGROUND

This project also aims to reinforce the need for having an organized and accessible database particularly in the area of e-shops that are continually growing. e commerce is on the rise largely due to the proliferation of internet and mobile phone hence the need for greater backend systems has deepened. Merchants who operate online must maintain product information, manage customer interaction and transactions, as well as offer their customers user satisfactions.

With an increase in the number of products and the volume of the transactions, the e-business cannot do without a formidable database management system. The aim of the project is to develop a uniform and distributed architecture which is able to operate with different kinds of e-commerce information such as product specifications, orders, customer reviews, as well as payment details. This will include keeping record of the products, order management, customer reviews business, and payments management. Due to the elimination of the inefficiency factor from the use of the dataset by implementing a relational database system in this case, the project seeks to achieve the goal from the structure that will be adaptable all round as far as E-commerce is concerned.

Significance of the Problem: The lack of a well-defined database system in businesses leads to inefficiency in electronic transactions for the business, high chances of data duplication, and lots of security risks. The limitation on the amount of data support becomes an issue and response time and system performance on the other hand suffer and users aggravate. The construction of a comprehensive database management system assists in the maintenance of proper data integrity, increased storage space, and data safety which are some of the most crucial elements of an e-commerce system.

III. SCOPE

The project goals in terms of the design and the development of the “Online Shopping Cart System” database cover the three levels of a database - a multi user relational database suitable for online shopping centers. The main purpose of the project is to create a database that can smoothly resolve various e-commerce challenges such as data integrity, scalability, security, performance, etc.

The database will include important elements such as Users, Products, Categories, Shopping Carts, Orders, Payments, and Reviews taking from the data set. There will be prospects of user account management, tracking the availability of the products, shoppers’ cart actions, and order management among others. Besides, the database is going to be developed in such a way that complex features such as how to make payments online to prevent fraud, real-time information, or working with other companies’ applications is possible. It will ensure that optimization works on the target format so that there is very little duplication in the database design, there is efficient retrieval of information from the database and also there is provision for other functions e.g. payment, review and order tracking penetration. The database of the project will be designed to ensure that the growing number of users and constant change of the business will be adequately supported by the development of appropriate database management systems. This is important because it is the main part supporting a good and easy to use electronically based marketing.

IV. TARGET USER

The Shopper’s database system is intended to serve the shoppers who go to the online shopping web site to seek products, buy them and check the status of their orders.

These customers use the database to create and enforce their profiles, provide true product information, and handle the payment settlements. Also, Managers and Administrators will have access to the database for management purposes to keep track of activities, customer activities, stock control and compile sales analysis reports using the system. System will be used for catalog marketing, catalog control, price updates and promotion management.

On this type of database architecture, an online vendor such as Amazon or Flipkart would implement in order to handle millions of users and products. The online store platform has a hierarchical structure, based on the database. Online customers can view products, order goods, and give feedback, whilst offline users maintain the sales database and analyze the sales data to improve stock levels. This project is envisaged to enhance the very base of the business systems that will help manage the databases to ensure quality, customer satisfaction, and the business operating in a viable arena.

V. E/R DIAGRAM

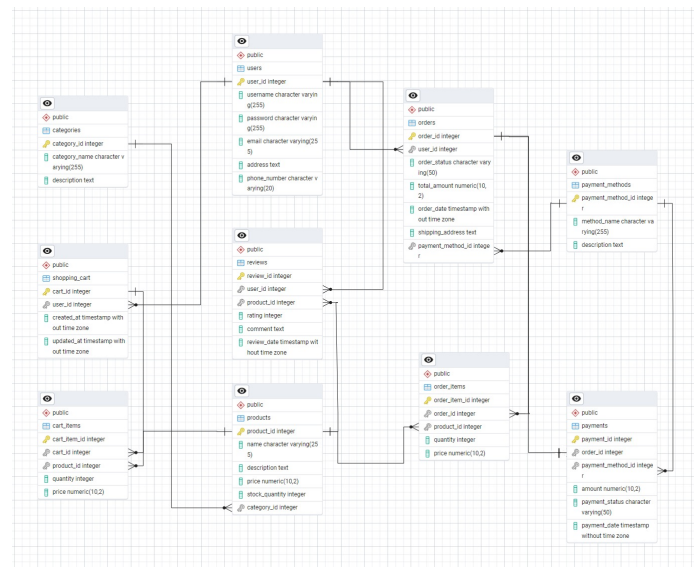


Fig. 3. ER diagram for Online Shopping System

1) Users to Shopping Cart:

- **One-to-Many:** One user can have multiple shopping carts, but each shopping cart belongs to only one user.
- The user_id in the shopping_cart table references user_id in the users table.

2) Users to Orders:

- **One-to-Many:** A user can place multiple orders (one-to-many), and an order is placed by one user (many-to-one).
- The user_id in the orders table references user_id in the users table.

3) Users to Reviews:

- One-to-Many: A user can write multiple reviews.
- The `user_id` in the reviews table references `user_id` in the users table.

4) Products to Categories:

- Many-to-One: Each product belongs to a single category, but a category can have multiple products.
- The `category_id` in the products table references `category_id` in the categories table.

5) Shopping Cart to Cart Items:

- One-to-Many: A shopping cart can contain multiple cart items.
- The `cart_id` in the cart_items table references `cart_id` in the shopping_cart table.

6) Cart Items to Products:

- Many-to-One: Each cart item is associated with a single product, but a product can appear in multiple cart items.
- The `product_id` in the cart_items table references `product_id` in the products table.

7) Orders to Order Items:

- One-to-Many: An order can have multiple order items.
- The `order_id` in the order_items table references `order_id` in the orders table.

8) Order Items to Products:

- Many-to-One: Each order item is associated with one product, but a product can appear in multiple order items.
- The `product_id` in the order_items table references `product_id` in the products table.

9) Orders to Payments:

- One-to-One and One-to-Many: Typically one-to-one in many cases. Once an order is placed, it is associated with a single payment.
- There may be few cases when an order can be paid in installments.

10) Payments to Payment Methods:

- Many-to-One: A payment is associated with one payment method, but a payment method can be used for multiple payments.
- The `payment_method_id` in the payments table references `payment_method_id` in the payment_methods table.

11) Products to Reviews:

- One-to-Many: A product can have multiple reviews.
- The `product_id` in the reviews table references `product_id` in the products table.

VI. BCNF

To ensure the accuracy and efficiency of our database structure, it's crucial to transform the schema into Boyce-Codd Normal Form (BCNF). This process involves analyzing the functional dependencies within each relation and decomposing them where necessary. The primary goal is to eliminate anomalies and redundancies while preserving data integrity. We'll begin by identifying the dependencies in each table and then modify them to comply with the rules of BCNF.

1) Users Table:

- Functional Dependencies:

– `user_id` (Candidate key) \rightarrow `username`, `password`, `email`, `address`, `phone_number`

Using `user_id` we can uniquely determine all the other attributes and there are no partial dependencies or transitive dependencies.

Conclusion: This table is already in BCNF.

2) Categories Table:

- Functional Dependencies:

– `category_id` (Candidate key) \rightarrow `category_name`, `description`

Using `category_id`, we can uniquely determine all the other attributes and there are no partial dependencies or transitive dependencies.

Conclusion: This table is already in BCNF.

3) Products Table:

- Functional Dependencies:

– `product_id` (Candidate key) \rightarrow `name`, `description`, `price`, `stock_quantity`, `category_id`

`product_id` is the Candidate key, and it determines all other attributes. Each product has only one category and there are no partial dependencies or transitive dependencies.

Conclusion: This table is already in BCNF.

4) Reviews Table:

- Functional Dependencies:

– `review_id` (Candidate key) \rightarrow `user_id`, `product_id`, `rating`, `comment`, `review_date`

`review_id` is the Candidate key and it determines all other attributes. One `review_id` can have only one rating or comment for one product by the user and there are no partial dependencies or transitive dependencies.

Conclusion: This table is already in BCNF.

5) Shopping Cart Table:

- Functional Dependencies:

- $\text{cart_id (Candidate key)} \rightarrow \text{user_id, created_at, updated_at}$

cart_id is the Candidate key and it determines all other attributes, which means each cart belongs to one user and has timestamps and there are no partial dependencies or transitive dependencies.

Conclusion: This table is already in BCNF.

6) Cart Items Table:

- Functional Dependencies:

- $\text{cart_item_id (Candidate key)} \rightarrow \{\text{cart_id, product_id, quantity, price}\}$

Using cart_item_id we can determine all other attributes and there are no partial dependencies or transitive dependencies..

Conclusion: This table is already in BCNF.

7) Orders Table:

- Functional Dependencies:

- $\text{order_id (Candidate key)} \rightarrow \{\text{user_id, order_status, total_amount, order_date, shipping_address, payment_method_id}\}$

Using order_id as the Candidate key we can determine all other attributes. Each order belongs to one user and there are no partial dependencies or transitive dependencies.

Conclusion: This table is already in BCNF.

8) Order Items Table:

- Functional Dependencies:

- $\text{order_item_id (Candidate key)} \rightarrow \{\text{order_id, product_id, quantity, price}\}$

order_item_id is the Candidate key, and it determines all other attributes. Each Order Item ID has only one Price and there are no partial dependencies or transitive dependencies.

Conclusion: This table is already in BCNF.

9) Payments Table:

- Functional Dependencies:

- $\text{payment_id (Candidate key)} \rightarrow \{\text{order_id, payment_method_id, amount, payment_status, payment_date}\}$

payment_id is the Candidate key, and it determines all other attributes. Each payment_id can have only one payment_method and there are no partial dependencies or transitive dependencies.

Conclusion: This table is already in BCNF.

10) Payment Methods Table:

- Functional Dependencies:

- $\text{payment_method_id (Candidate key)} \rightarrow \{\text{method_name, description}\}$

payment_method_id is the Candidate key and determines all other attributes and there are no partial dependencies or transitive dependencies.

Conclusion: This table is already in BCNF.

VII. TABLE RELATIONS

1) USERS

- Primary Key: user_id

- Foreign Keys: None

- Attributes:

- user_id (integer, not NULL): The unique identifier for each user. Serves as the primary key.
- username (varchar, not NULL): User's display name for account purposes.
- email (varchar, not NULL): Email address used for communication.
- password (varchar, not NULL): Password for authentication.
- address (varchar, NULL): User's physical address for deliveries.
- phone_number (varchar, NULL): User's contact number for communication.

- Purpose: The "users" table holds the data related to users. It stores unique data for each user like user_id , username , email , password , and optional fields like address and phone number for communication.

- Constraints: user_id , username , email , and password cannot be NULL as they are essential for identifying and communicating with the user.

- Actions on Foreign Keys: No action, as no foreign key references this table.

2) CATEGORIES

- Primary Key: category_id

- Foreign Keys: None

- Attributes:

- category_id (integer, not NULL): Unique identifier for each category.
- category_name (varchar, not NULL): Name of the product category.
- description (text, NULL): Description of the category.

- Purpose: The "categories" table classifies the types of products available in the system.

- Constraints: category_id and category_name cannot be NULL as every product must belong to a category.

- Actions on Foreign Keys: No action, as no foreign key references this table.

3) PRODUCTS

- Primary Key: product_id

- Foreign Keys: category_id (references $\text{Categories.category_id}$)

- Attributes:
 - product_id (integer, not NULL): Unique identifier for each product.
 - name (varchar, not NULL): Name of the product.
 - description (text, NULL): Detailed description of the product.
 - price (decimal, not NULL): The selling price of the product.
 - stock_quantity (integer, not NULL): Available stock for the product.
 - category_id (integer, not NULL): Refers to the category to which the product belongs.
- Purpose: The "products" table holds information about the items available for purchase. Each product has a unique identifier and attributes like price, stock, and category.
- Constraints: product_id, price, stock_quantity, and category_id cannot be NULL because they are essential for product sales and categorization.
- Actions on Foreign Keys: If category_id (referenced category) is deleted, associated products can be deleted (ON DELETE CASCADE) or set to NULL.

4) REVIEWS:

- Primary Key: review_id
 - user_id (references Users.user_id)
 - product_id (references Products.product_id)
- Foreign Keys: None
- Attributes:
 - review_id (integer, not NULL): Unique identifier for each review.
 - user_id (integer, not NULL): Refers to the user writing the review.
 - product_id (integer, not NULL): Refers to the product being reviewed.
 - rating (integer, not NULL): Rating given to the product (1 to 5).
 - comment (text, NULL): Optional comment about the product.
 - review_date (timestamp, not NULL): Date the review was submitted.
- Purpose: The "reviews" table stores product reviews from users, including the rating and optional comment.
- Constraints: review_id, user_id, product_id, and rating cannot be NULL.
- Actions on Foreign Keys:
 - If user_id is deleted, the reviews may either be deleted or anonymized.
 - If product_id is deleted, the reviews may either be deleted or retained without product reference.

5) SHOPPING CART:

- Primary Key: cart_id

- Foreign Keys: user_id (references Users.user_id)
- Attributes:
 - cart_id (integer, not NULL): Unique identifier for each cart.
 - user_id (integer, not NULL): Refers to the user who owns the cart.
 - created_at (timestamp, not NULL): Timestamp when the cart was created.
 - updated_at (timestamp, NULL): Timestamp when the cart was last updated.
- Purpose: The "shopping_cart" table stores details about each user's cart, including the user and time-related data.
- Constraints: cart_id and user_id cannot be NULL, as they are needed to track the cart's owner and lifecycle.
- Actions on Foreign Keys: When user_id is deleted, the related cart will be deleted (ON DELETE CASCADE).

6) CART ITEMS:

- Primary Key: cart_item_id
- Foreign Keys: user_id (references Shopping Cart.cart_id)
 - product_id (references Products.product_id)
- Attributes:
 - cart_item_id (integer, not NULL): Unique identifier for each item in a cart.
 - cart_id (integer, not NULL): Refers to the cart this item belongs to.
 - product_id (integer, not NULL): Refers to the product added to the cart.
 - quantity (integer, not NULL): Quantity of the product in the cart.
 - price (decimal, not NULL): Price of the product at the time of adding to the cart.
- Purpose: The "cart_items" table stores information about products added to the shopping cart, including product details and quantity.
- Constraints: cart_id, product_id, quantity, and price cannot be NULL since each item must be linked to a product in a cart.
- Actions on Foreign Keys:
 - If cart_id is deleted, the associated items will also be deleted (ON DELETE CASCADE).
 - If product_id is deleted, the cart items may either be deleted or set to NULL.

7) ORDERS:

- Primary Key: order_id
- Foreign Keys: user_id (references Users.user_id)
 - payment_method_id (references Payment Methods.payment_method_id)
- Attributes:

- order_id (integer, not NULL): Unique identifier for each order.
- user_id (integer, not NULL): Refers to the user placing the order.
- order_status (varchar, not NULL): Status of the order (e.g., pending, shipped, delivered).
- total_amount (decimal, not NULL): Total amount for the order.
- order_date (timestamp, not NULL): Date the order was placed.
- shipping_address (varchar, not NULL): Address for delivering the order.
- payment_method_id (integer, not NULL): Refers to the method used for payment.
- Purpose: The "orders" table stores details of user purchases, including payment and shipping information.
- Constraints: order_id, user_id, payment_method_id, and total_amount cannot be NULL.
- Actions on Foreign Keys:
 - If user_id is deleted, the order may either be deleted or set to NULL.
 - If payment_method_id is deleted, the order can either reference another method or be set to NULL.

8) ORDER ITEMS:

- Primary Key: order_item_id
- Foreign Keys: order_id (references Orders.order_id)
 - product_id (references Products.product_id)
- Attributes:
 - order_item_id (integer, not NULL): Unique identifier for each item in the order.
 - order_id (integer, not NULL): Refers to the order this item belongs to.
 - product_id (integer, not NULL): Refers to the product in the order.
 - quantity (integer, not NULL): Quantity of the product ordered.
 - price (decimal, not NULL): Price of the product at the time of the order.
- Purpose: The "order_items" table stores information about the products in an order, including quantity and price.
- Constraints: order_item_id, order_id, product_id, and quantity cannot be NULL.
- Actions on Foreign Keys:
 - If order_id is deleted, the corresponding order items are deleted (ON DELETE CASCADE).
 - If product_id is deleted, the corresponding order items may either be deleted or set to NULL.

9) PAYMENTS:

- Primary Key: payment_id

- Foreign Keys:
 - order_id (references Orders.order_id)
 - payment_method_id (references Payment Methods.payment_method_id)
- Attributes:
 - payment_id (integer, not NULL): Unique identifier for each payment.
 - order_id (integer, not NULL): Refers to the associated order.
 - payment_method_id (integer, not NULL): Refers to the method used for payment.
 - amount (decimal, not NULL): Amount paid.
 - payment_status (varchar, not NULL): Status of the payment (e.g., pending, completed).
 - payment_date (timestamp, not NULL): Date the payment was made.
- Purpose: The "payments" table stores information about payments made for orders, including the amount and payment method used.
- Constraints: payment_id, order_id, payment_method_id, and amount cannot be NULL.
- Actions on Foreign Keys:
 - If order_id is deleted, the payment may be deleted or set to NULL.
 - If payment_method_id is deleted, the payment may reference another method or be set to NULL.

10) PAYMENT METHODS:

- Primary Key: payment_method_id
- Foreign Keys: None
- Attributes:
 - payment_method_id (integer, not NULL): Unique identifier for each payment method.
 - method_name (varchar, not NULL): Name of the payment method (e.g., credit card, debit card).
 - description (text, NULL): Description of the payment method.
- Purpose: The "payment_methods" table stores available payment options, such as credit cards, debit cards, etc.
- Constraints: payment_method_id and method_name cannot be NULL.
- Actions on Foreign Keys:
 - No action, as no foreign key references this table.

REFERENCES

- [1] Faker Documentation <https://faker.readthedocs.io/en/master/>

TABLE I
GROUP EVALUATION TABLE

Evaluation Criteria	Mounika Pasupuleti	Sahithya Arveti Nagaraju	Sushmitha Manjunatha
	Team Member 1	Team Member 2	Team Member 3
How effectively did your group mate work with you?	5	5	5
Contribution in writing the report	5	5	5
Demonstrates a cooperative and supportive attitude	5	5	5
Contributes significantly to the success of the project	5	5	5
Total	20	20	20