

PLASMID INNOVATION. LTD

INTERNSHIP

SQL MAJOR PROJECT

Project Title: Online Retail Application Database

Domain: E-Commerce, Database Systems

Technologies Used: MySQL

PREPARED BY: THUMATI TEJA

Online Retail Application Database

Major Project Report

Abstract

In the digital era, e-commerce has transformed traditional shopping into a dynamic and user-centric experience. The **Online Retail Application Database** project aims to design and implement a robust backend relational database that supports the core functionalities of an online retail system. This includes customer registration, product catalog management, order processing, billing, and payment handling. The project leverages SQL to create structured tables, enforce relationships through foreign keys, and ensure data integrity. Through this project, we simulate a real-world e-commerce environment where customers can seamlessly browse products, place orders, and make payments, while administrators can manage product listings and order tracking.

Online Retail Application Database

Major Project Report

1. Introduction

The rise of e-commerce and online retailing has created a massive need for efficient database systems that can manage large volumes of products, orders, and payments. The Online Retail Application Database project aims to design a relational database that supports the backend of a typical online shopping platform.

Objectives:

- Design a normalized relational database for online retail.
- Implement SQL queries for product management, customer orders, and payments.
- Ensure data integrity and efficient retrieval.

2. System Requirements

2.1 Functional Requirements:

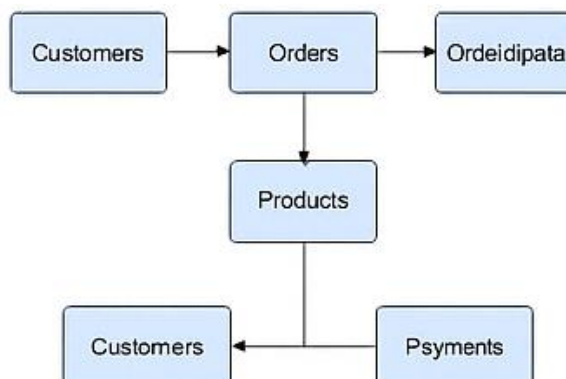
- Manage products, categories, customers, orders, and payments.
- Enable order tracking and invoice generation.

2.2 Non-Functional Requirements:

- Database normalization to at least 3NF.
- Indexing for performance improvement.
- Security measures (e.g., input validation).

3. Database Design

The following ER diagram illustrates the entity relationships:



4. Database Schema and SQL

```
CREATE TABLE Categories (  
    category_id INT PRIMARY KEY,  
    category_name VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE Products (  
    product_id INT PRIMARY KEY,  
    product_name VARCHAR(100) NOT NULL,  
    price DECIMAL(10,2) NOT NULL,  
    quantity_in_stock INT NOT NULL,  
    category_id INT,  
    FOREIGN KEY (category_id) REFERENCES Categories(category_id)  
);
```

```
CREATE TABLE Customers (  
    customer_id INT PRIMARY KEY,  
    name VARCHAR(100) NOT NULL,  
    email VARCHAR(100),  
    phone VARCHAR(15)  
);
```

```
CREATE TABLE Orders (  
    order_id INT PRIMARY KEY,  
    customer_id INT,  
    order_date DATE,  
    status VARCHAR(50),  
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)  
);
```

```
CREATE TABLE OrderDetails (  
    order_detail_id INT PRIMARY KEY,  
    order_id INT,  
    product_id INT,  
    quantity INT,  
    subtotal DECIMAL(10,2),  
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),  
    FOREIGN KEY (product_id) REFERENCES Products(product_id)  
);
```

```
CREATE TABLE Payments (  
    payment_id INT PRIMARY KEY,  
    order_id INT,  
    payment_date DATE,  
    amount_paid DECIMAL(10,2),  
    method VARCHAR(50),  
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)  
);
```

5. Sample Data

```
INSERT INTO Categories VALUES (1, 'Electronics'), (2, 'Books');  
INSERT INTO Products VALUES (101, 'Laptop', 70000, 50, 1);  
INSERT INTO Products VALUES (102, 'Smartphone', 30000, 100, 1);  
INSERT INTO Products VALUES (103, 'Fiction Book', 500, 200, 2);  
INSERT INTO Customers VALUES (1, 'Ravi Kumar', 'ravi@mail.com', '9876543210');  
INSERT INTO Orders VALUES (201, 1, '2025-06-10', 'Shipped');  
INSERT INTO OrderDetails VALUES (301, 201, 101, 1, 70000);  
INSERT INTO Payments VALUES (401, 201, '2025-06-11', 70000, 'Credit Card');
```

6. Key SQL Queries

a) List all products in stock:

```
SELECT product_name, price FROM Products WHERE quantity_in_stock > 0;
```

b) Get customer orders:

```
SELECT Orders.order_id, Customers.name, Orders.order_date  
FROM Orders JOIN Customers ON Orders.customer_id = Customers.customer_id;
```

c) Calculate total order cost:

```
SELECT order_id, SUM(subtotal) AS total_amount FROM OrderDetails GROUP BY order_id;
```

d) View full order details:

```
SELECT c.name, p.product_name, od.quantity, od.subtotal  
FROM OrderDetails od  
JOIN Orders o ON od.order_id = o.order_id  
JOIN Products p ON od.product_id = p.product_id  
JOIN Customers c ON o.customer_id = c.customer_id;
```

7. Normalization

The database is normalized to 3NF:

- No repeating groups or multivalued fields.
- Partial and transitive dependencies are removed.
- Foreign keys maintain referential integrity.

8. Future Enhancements

- Add user authentication and admin dashboard.
- Integrate delivery tracking and return/refund management.
- Analytics module for sales performance.

9. Conclusion

The project successfully demonstrates the design and implementation of a structured, relational database for an online retail platform. It highlights the importance of proper schema design, normalization, and SQL proficiency.