DBMS | CSE202
PROJECT GUIDE

By Anya Hooda (2022088)

ONLINE ELECTRONICS RETAIL STORE

BANDWIDTH ELECTRONICS

A Robust Database Management System for an E-commerce Platform Specializing in Electronics. The system aims to efficiently manage information associated with an online electronics retail store, offering a diverse range of products. Key functionalities include handling customer information, managing inventory, processing sales, tracking financial data, and providing statistical insights.

PROJECT SCOPE

Users Can :

- Authenticate themselves on the application
- Search for components and filter results based on preferences
- Explore products compatible with their selected items
- View purchase history
- Provide product ratings and reviews.

Store Manager(s) Can :

- Add or remove products.

- Manage offers and discounts.
- Specify product quantity and availability in warehouses.
- Access store statistics, including inventory, registered users, and revenue.

REQUIREMENTS

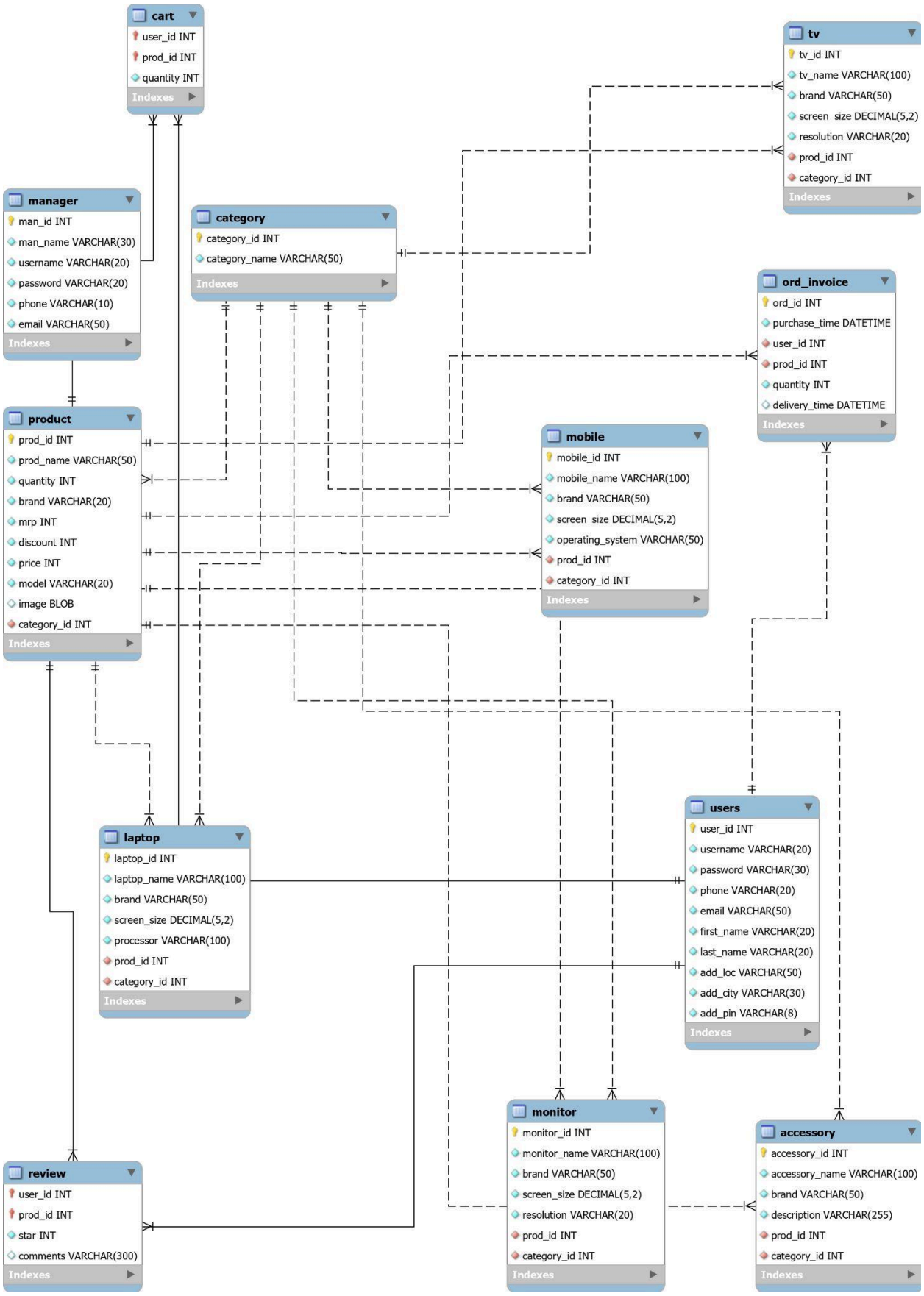
Functional :

1. User Registration and Authentication: New users can register and log in.
2. Product Searching and Filtering: Search products by name, type, brand, etc. Filter Products on the basis of prices, ratings, etc.
3. Shopping Cart Management: Users can update their cart with products. Users can also view amount , delivery time, delivery address, etc.
4. Update Inventory and Catalogue: Admin can update product availability. Add new products. Delete Products. Edit Product Details/Properties.
5. Provide Offers: Admin can update product offers and discounts. Here some constraints can be added for e.g., if user purchases using online payment methods rather than COD then they can avail additional discount of 10% , etc.
6. View Store Statistics: Admin can access information about revenue generation, inventory, product sales, etc.

Technical :

1. Web Application Interface: Intuitive and user-friendly design. Seamless signup/login process. Display highest-rated or discounted products upon authentication.
2. User Experience: Users can search and filter products easily. Explore products based on categories, ratings, and brands. Updated cart management with a smooth checkout process.
3. Database Management: Track registered users and products with defining attributes. Efficiently manage brand names, prices, features, ratings, warranty, etc. Cart management for selected items with flexible update

RELATIONAL MODEL


cart
user_id INT
prod_id INT
quantity INT
Indexes
tv
tv_id INT
tv_name VARCHAR(100)
brand VARCHAR(50)
screen_size DECIMAL(5,2)
resolution VARCHAR(20)
prod_id INT
category_id INT
Indexes
manager
man_id INT
man_name VARCHAR(30)
username VARCHAR(20)
password VARCHAR(20)
phone VARCHAR(10)
email VARCHAR(50)
Indexes
category
category_id INT
category_name VARCHAR(50)
Indexes
ord_invoice
ord_id INT
purchase_time DATETIME
user_id INT
prod_id INT
quantity INT
delivery_time DATETIME
Indexes
product
prod_id INT
prod_name VARCHAR(50)
quantity INT
brand VARCHAR(20)
mrp INT
discount INT
price INT
model VARCHAR(20)
image BLOB
category_id INT
Indexes
mobile
mobile_id INT
mobile_name VARCHAR(100)
brand VARCHAR(50)
screen_size DECIMAL(5,2)
operating_system VARCHAR(50)
prod_id INT
category_id INT
Indexes
users
user_id INT
username VARCHAR(20)
password VARCHAR(30)
phone VARCHAR(20)
email VARCHAR(50)
first_name VARCHAR(20)
last_name VARCHAR(20)
add_loc VARCHAR(50)
add_city VARCHAR(30)
add_pin VARCHAR(8)
Indexes
laptop
laptop_id INT
laptop_name VARCHAR(100)
brand VARCHAR(50)
screen_size DECIMAL(5,2)
processor VARCHAR(100)
prod_id INT
category_id INT
Indexes
monitor
monitor_id INT
monitor_name VARCHAR(100)
brand VARCHAR(50)
screen_size DECIMAL(5,2)
resolution VARCHAR(20)
prod_id INT
category_id INT
Indexes
accessory
accessory_id INT
accessory_name VARCHAR(100)
brand VARCHAR(50)
description VARCHAR(255)
prod_id INT
category_id INT
Indexes
review
user_id INT
prod_id INT
star INT
comments VARCHAR(300)
Indexes

ER DIAGRAM


email
username
password
user_ID
last_name
name
first_name
User
address
add_Hno
add_locality
add_city
add_pin
order_history
phone
Receives
Order Invoice
purchase_ID
ord_ID
delivery_ID
payment_type
user_ID
Cart
qty
user_ID
prod_ID
sumTotal
Manages
Provides
Manages
Store Manager
man_ID
username
password
name
email
phone
Review
prod_ID
stars
comments
user_ID
Product
prod_ID
qty
name
brand
mrp
discount
price()
model
image
Category
Belongs To
category_ID
category_name
tv
mobiles
laptops
monitors
accessories
screenSize
resolution
features
screenSize
storage_Cap
color
camera_res
screenSize
storage_Cap
weight
battery_Cap
screenSize
resolution
connectivity_ports
refresh_rate
type
color
compatibility

SAMPLE QUERIES

1. To select users by location and city

SELECT * FROM users WHERE add_pin = '30301' AND add_city = 'Atlanta';

RAE : σ add_pin='30301' AND add_city='Atlanta' (users)

2. Update the quantity of a specific product with a given prod_id

UPDATE product SET quantity = 55 WHERE prod_id = 3;

RAE: product{quantity} ← π quantity (σ prod_id=3 (product))

3. Retrieve the username and email of users who have not made any purchases

SELECT username, email FROM users

WHERE user_id NOT IN (SELECT DISTINCT user_id FROM ord_invoice);

RAE: π username, email (users - π user_id (ord_invoice))

4. Retrieve all products with a price greater than the average price

SELECT * FROM product

WHERE price > (SELECT AVG(price) FROM product);

RAE: σ price > avg_price (product)

5. Retrieve the names and quantities of products ordered by a specific user

SELECT p.prod_name, oi.quantity

FROM ord_invoice oi

INNER JOIN product p ON oi.prod_id = p.prod_id

WHERE oi.user_id = 3;

RAE: π prod_name, quantity ((ord_invoice ⋈ prod_id=product.prod_id) ⋈_s user_id=3)

6. List all products along with their respective categories, sorted by category name

SELECT p.prod_name, c.category_name

FROM product p

INNER JOIN category c ON p.category_id = c.category_id

ORDER BY c.category_name;

RAE: π prod_name, category_name ((product ⋈ category_id=category.category_id) ⋈_o category_name)

7. Counting the number of users in each city

SELECT add_city, COUNT(*) AS num_users FROM users

GROUP BY add_city;

RAE: ρ UsersCountByCity (π add_city, COUNT(*) AS num_users (users))_grouped

8. Selecting products with a discount percentage above 10%

SELECT * FROM product

WHERE discount > 10;

RAE: ρ DiscountedProducts (σ discount > 10 (product))

9. Selecting users who have purchased products with a specific brand

SELECT DISTINCT u.*

FROM users u

```
JOIN ord_invoice oi ON u.user_id = oi.user_id

JOIN product p ON oi.prod_id = p.prod_id

WHERE p.brand = 'Samsung';
```

RAE: ρ SamsungBuyers (π * (users ⋈ (product ⋈ ord_invoice ⋈ (σ brand = 'Samsung' (product)))))

10. Selecting users who have made purchases but have not reviewed any products.

```
SELECT *FROM users

WHERE user_id IN (

   SELECT DISTINCT user_id

   FROM ord_invoice

   WHERE user_id NOT IN (

      SELECT DISTINCT user_id

      FROM review

   ));
```

RAE : ρ UsersNoReviews (π * (users ⋈ (ord_invoice - π user_id (review))))

TRIGGERS

1. Trigger to Monitor Inventory Level: This trigger is designed to monitor the inventory level of products after the insertion of new orders into the ord_invoice table. It triggers an action when a new order is inserted and checks if the inventory level of the ordered product falls below a certain threshold.

2. Trigger for Customer Analysis: This trigger is responsible for updating the average rating of product categories after the insertion of new reviews into the review table. It calculates the average rating for each product category based

on the reviews received and updates the corresponding category's average rating.

Code Snippet:

```
DELIMITER $$
CREATE TRIGGER check_inventory_level
AFTER INSERT ON ord_invoice
FOR EACH ROW
BEGIN
    DECLARE inventory_count INT;
    DECLARE product_name VARCHAR(100);

    -- Get the current inventory count for the product
    SELECT quantity INTO inventory_count
    FROM product
    WHERE prod_id = NEW.prod_id;

    -- Get the product name for notification
    SELECT prod_name INTO product_name
    FROM product
    WHERE prod_id = NEW.prod_id;


    -- Check if the inventory level is below threshold
    IF inventory_count < 10 THEN
        -- Send notification email to the manager
        INSERT INTO notification (message)
        VALUES (CONCAT('Inventory level for product ', product_name, ' is below threshold.'));
    END IF;
END$$

DELIMITER ;

-- Trigger for customer analysis
DELIMITER $$

CREATE TRIGGER update_avg_rating
AFTER INSERT ON review
FOR EACH ROW
BEGIN
    DECLARE avg_rating DECIMAL(3,2);
    DECLARE category_id INT;

    -- Get the category ID of the reviewed product


     -- Get the category ID of the reviewed product
     SELECT category_id INTO category_id
     FROM product
     WHERE prod_id = NEW.prod_id;

     -- Calculate the average rating for the category
     SELECT AVG(star) INTO avg_rating
     FROM review
     WHERE prod_id IN (SELECT prod_id FROM product WHERE category_id = category_id);

     -- Update the average rating for the category
     UPDATE category
     SET avg_rating = avg_rating
     WHERE category_id = category_id;
END$$

DELIMITER ;
SHOW TRIGGERS;
```

TRANSACTIONS

Non-Conflicting Transactions:

1. View All Products and View Products Category Wise
2. Add and delete Products from the Cart
3. Display the Cart
4. Place Order
5. Display a Particular Product's Details
6. View Inventory
7. Add and Delete Products From the Inventory
8. Modify Product Details
9. View Orders
10. View Customers

Conflicting Transactions:

1. Concurrent Product Modification Transaction:

   Two managers simultaneously attempt to modify the same product's details. One manager's changes should overwrite the other's.

2. Concurrent Purchase Transaction:

   Two users simultaneously attempt to purchase the same product. Ensure that the inventory is updated correctly and only one user successfully completes the purchase.

USER'S GUIDE

Table of Contents

1. Introduction

This user guide provides comprehensive instructions on interacting with the MySQL database and Python program for managing an e-commerce system of electronic store.

2. MySQL Database Setup

2.1 Database Schema

The MySQL database schema consists of several tables:

1. users: Stores user information.
2. manager: Stores manager information.
3. category: Stores product categories.
4. product: Stores product details.
5. ord_invoice: Stores order details.
6. cart: Stores shopping cart items.
7. review: Stores product reviews.
8. tv, mobile, laptop, monitor, accessory: Stores specific product details.

2.2 Data Population

1. The database is pre-populated with sample data for demonstration purposes.
2. Tables like users, manager, category, product, etc., contain sample entries.
3. Sample data includes users, managers, products, categories, etc.

3. Python Program Overview

1. The Python program provides functionalities for both managers and users.

2. Managers can perform operations like viewing inventory, adding new products, modifying product details, deleting products, viewing orders, and viewing customers.
3. Users can view products, add products to the cart, remove products from the cart, view the cart, and place orders.
4. The program interacts with the MySQL database using MySQL python connector to perform these operations.

4. Manager Operations

4.1 Manager Login

1. Managers can log in using their username and password.
2. Upon successful login, managers gain access to various management functionalities.

4.2 Manager Actions

Managers can perform the following actions:

1. View Inventory: View all products in the inventory.
2. Add New Product: Add a new product to the inventory.
3. Modify Product Details: Modify details of existing products.
4. Delete Product: Delete a product from the inventory.
5. View Orders: View all orders placed by users.
6. View Customers: View all registered customers.
7. Logout as Manager: Logout from the manager interface.

```
Manager Actions:
1. View Inventory
2. Add New Product
3. Modify Product Details
4. Delete Product
5. View Orders
6. View Customers
7. Logout as Manager
Enter your choice:
```

5. User Operations

5.1 User Login

1. Users can log in using their username and password.
2. Upon successful login, users gain access to various user functionalities.

```
Select user type:
1. Login as User
2. Login as Manager
3. Exit
Enter your choice: 1
Enter username: mzott222
Enter password: letmein
Login successful.
```

5.2 User Actions

Users can perform the following actions:

1. View All Products: View all products available in the inventory.
2. View Products Category-Wise: View products categorized by type (e.g., TVs, Mobiles, Laptops, etc.).
3. View Cart: View the items added to the cart along with the total price.
4. Add Product to Cart: Add a product to the cart for purchase.
5. Remove Product from Cart: Remove a product from the cart.
6. Place Order: Place an order for the products in the cart.

```
Select an option:
1. View All Products
2. View Products Category-wise
3. Display Cart
4. Add to Cart
5. Remove from Cart
6. Place Order
7. Display Product Details
8. Logout
Enter your choice: 8
Logging out.
```
