

# **E-Commerce Price & Stock Tracker Across Multiple Stores**

Anika P - PES1UG23CS073

Anish M - PES1UG23CS078

## **Description:**

This project is a streamlined e-commerce management and storefront system built using Flask and MySQL. It enables customers to register, browse categorized products across multiple stores, track price changes, create wishlists, and set store-specific price alerts. The system maintains product price history and alerts users when desired prices are reached. Administrators can manage products, categories, stores, and inventory, as well as view key analytics and run automated restocking procedures. The application uses Jinja2 templates for dynamic pages and MySQL for data storage, providing a functional end-to-end e-commerce management workflow.

## **User requirement specification:**

- **Purpose of the project:**

This project implements a small e-commerce web application using Flask (Python) and MySQL. It allows customers to browse products, manage wishlists, set price alerts, and make purchases, while administrators can manage products, inventory, stores and view analytics. The application provides price tracking (price history, alerts), inventory management, and typical e-commerce features (product listing, detail pages, customer accounts).

- **Scope of the project:**

Includes:

- 1) Customer-facing features: account registration/login, product browsing, product detail pages, wishlist management, price alerts, view purchase history, basic expense tracking.
- 2) Admin-facing features: manage products, categories, stores, inventory, users, and view analytics (counts, stock totals, alerts).
- 3) Backend: MySQL database with tables for Customers, Products, Category, Store, Inventory, Orders, PriceAlerts, PriceHistory, Wishlists, and stored procedures for some operations (app expects procedures like AddNewProduct, ComparePrices, AutoRestock).

Excludes (out of scope): Payment gateway integration, production-level security hardening (app uses a simple secret key and local DB settings), full CI/CD.

- **Detailed description:**

This is a small e-commerce management & storefront application. Customers sign up and log in to browse products grouped into categories and offered by different stores. Each store can have its own price for a product (modelled in Inventory with price\_in\_store), and the application records price history so customers can see price changes. Customers can add products to wishlists and set price alerts for specific products at specific stores; if a product reaches a target price the alert can be triggered (the app queries inventory/price to determine trigger). Administrators can add products (the app calls stored procedures to add products), update inventory levels, view analytics (counts of customers/products/stores and inventory totals), and run an auto-restock procedure. The app uses server-side templates (Jinja2) for pages and mysql-connector-python for DB access.

- **Function Requirements:**

- 1) User Registration & Login — Register new customers, store basic details and phone numbers; log in/out and maintain session.
- 2) Customer Profile Management — View and edit customer details (name, email, phones).
- 3) Product Browsing / Search — Browse product catalogue, filter by category, view paginated product lists.
- 4) Product Detail — See product details including store availability and prices (using Inventory join).
- 5) Wishlist Management — Create wishlists, add/remove products to wishlist (Wishlist, WishlistProduct).
- 6) Price Alerts — Create price alerts per product/store with target price, view, and delete alerts (table: PriceAlert).
- 7) Price History Viewing — Show historical price changes for a product (PriceHistory).
- 8) Shopping / Orders (basic) — Record a purchase/order (Orders table used for purchase history and expense tracking).
- 9) Admin Authentication & Dashboard — Admin login + admin dashboard with counts and analytics queries.
- 10) Admin Product Management — Add/edit/delete products (app calls stored procedures such as AddNewProduct).
- 11) Inventory Management — Manage inventory entries (quantity / price in store) and view total stock.
- 12) Store Management — Add/update store records (store name, possibly location).
- 13) Analytics & Reports — Dashboard queries count customers/products/stores, active alerts, inventory totals and aggregated category metrics.
- 14) Stored Procedures Interface — App calls stored procedures: e.g., ComparePrices, AutoRestock, AddNewProduct.

- 15) Notifications / Alert Triggering — Mechanism to flag triggered alerts (app queries PriceAlert with status = 'triggered').
- 16) CRUD via web forms — All entity create/update/delete operations accessible via admin/customer UI.
- 17) Input validation & Flash messages — Basic form validation with feedback via Flask flash().

### **List of Software/Tools/Programming languages used:**

#### Programming Languages

- Python 3.x (Backend logic and server-side development)
- HTML5, CSS3, JavaScript (Frontend templates and UI)

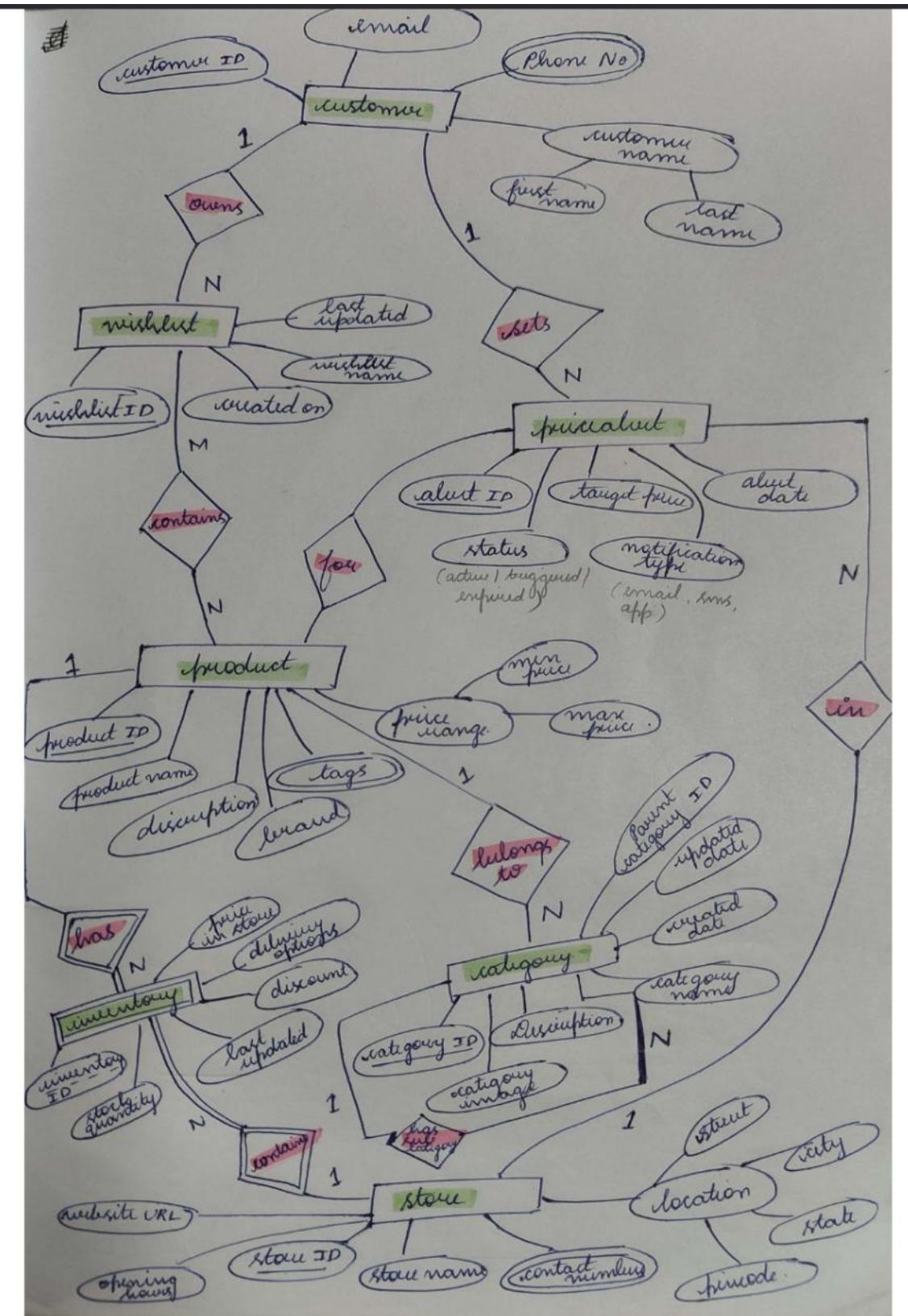
#### Frameworks & Libraries

- Flask (Web framework)
- Jinja2 (Templating engine)
- mysql-connector-python (Database connectivity)
- Bootstrap (Frontend styling, used in templates)

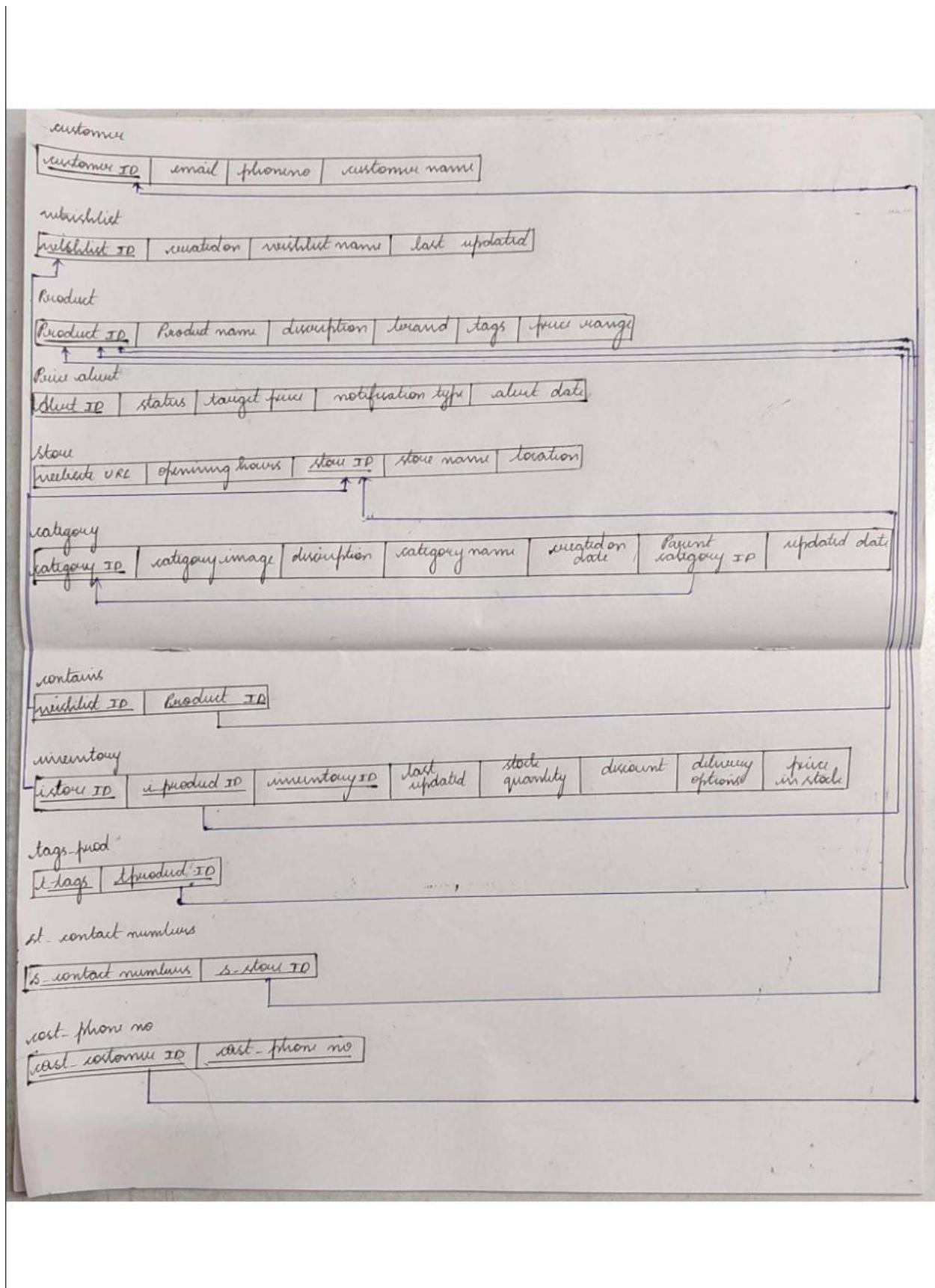
#### Database & Data Tools

- MySQL (Primary relational database)
- Development Tools
- Visual Studio Code (IDE for development)
- Git & GitHub (Version control)
- Command-line / Terminal (Running Flask server & dependencies)
- pip / virtualenv (Dependency and environment management)

## ER Diagram:



## Relational Schema:



## **DDL Commands:**

```
-- =====  
-- STRONG ENTITIES  
-- =====
```

```
CREATE TABLE Customer (  
    customer_id INT PRIMARY KEY,  
    first_name VARCHAR(50),  
    last_name VARCHAR(50),  
    email VARCHAR(100) UNIQUE  
);
```

```
-- Multivalued attribute: phone_no  
CREATE TABLE CustomerPhone (  
    customer_id INT,  
    phone_no VARCHAR(15),  
    PRIMARY KEY (customer_id, phone_no),  
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) ON DELETE CASCADE  
);
```

```
CREATE TABLE Category (  
    category_id INT PRIMARY KEY,  
    category_name VARCHAR(100),  
    description TEXT,  
    category_image VARCHAR(255),  
    parent_category_id INT,  
    FOREIGN KEY (parent_category_id) REFERENCES Category(category_id)  
    ON DELETE SET NULL -- Recursive relationship (subcategory → category)
```

);

ALTER TABLE Category

ADD COLUMN created\_date DATE,

ADD COLUMN updated\_date DATE;

ALTER TABLE Category

MODIFY created\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

MODIFY updated\_date DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP;

CREATE TABLE Store (

store\_id INT PRIMARY KEY,

store\_name VARCHAR(100),

opening\_hours VARCHAR(100),

website\_url VARCHAR(255),

street VARCHAR(100),

city VARCHAR(100),

state VARCHAR(100)

);

ALTER TABLE Store

ADD COLUMN pincode VARCHAR(10);

ALTER TABLE Store

MODIFY pincode VARCHAR(10) NOT NULL;

-- Multivalued attribute: contact number

```
CREATE TABLE StoreContact (
    store_id INT,
    contact_number VARCHAR(15),
    PRIMARY KEY (store_id, contact_number),
    FOREIGN KEY (store_id) REFERENCES Store(store_id) ON DELETE CASCADE
);
```

CREATE TABLE Product (

```
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    description TEXT,
    brand VARCHAR(50),
    min_price DECIMAL(10,2),
    max_price DECIMAL(10,2),
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES Category(category_id)
);
```

-- Multivalued attribute: tags

```
CREATE TABLE ProductTag (
    product_id INT,
    tag VARCHAR(50),
    PRIMARY KEY (product_id, tag),
    FOREIGN KEY (product_id) REFERENCES Product(product_id) ON DELETE CASCADE
);
```

```
CREATE TABLE Wishlist (
    wishlist_id INT PRIMARY KEY,
    wishlist_name VARCHAR(100),
    created_on DATE,
    last_updated DATE,
    customer_id INT,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
        ON DELETE CASCADE
);
```

-- M:N relationship between Wishlist and Product

```
CREATE TABLE WishlistProduct (
    wishlist_id INT,
    product_id INT,
    PRIMARY KEY (wishlist_id, product_id),
    FOREIGN KEY (wishlist_id) REFERENCES Wishlist(wishlist_id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES Product(product_id) ON DELETE CASCADE
);
```

```
CREATE TABLE PriceAlert (
    alert_id INT PRIMARY KEY,
    target_price DECIMAL(10,2),
    alert_date DATE,
    status VARCHAR(50),
    notification_type VARCHAR(50),
    product_id INT,
    customer_id INT,
    FOREIGN KEY (product_id) REFERENCES Product(product_id),
```

```
FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);
```

```
ALTER TABLE PriceAlert
ADD COLUMN store_id INT,
ADD CONSTRAINT fk_pricealert_store
FOREIGN KEY (store_id) REFERENCES Store(store_id)
ON DELETE CASCADE;
```

```
-- WEAK ENTITY: Inventory (depends on Product + Store)
```

```
CREATE TABLE Inventory (
    inventory_id INT,
    product_id INT,
    store_id INT,
    quantity INT,
    last_updated DATE,
    PRIMARY KEY (inventory_id, product_id, store_id),
    FOREIGN KEY (product_id) REFERENCES Product(product_id) ON DELETE CASCADE,
    FOREIGN KEY (store_id) REFERENCES Store(store_id) ON DELETE CASCADE
);
```

```
ALTER TABLE Inventory
ADD COLUMN price_in_store DECIMAL(10,2),
ADD COLUMN delivery_options VARCHAR(100),
ADD COLUMN discount DECIMAL(5,2);
```

## CRUD operation Screenshots:

create:

```
mysql> insert into customer values(8,"Ram","Panday","ram@example.com",0);
Query OK, 1 row affected (0.03 sec)

mysql> select * from customer;
+-----+-----+-----+-----+-----+
| customer_id | first_name | last_name | email | is_admin |
+-----+-----+-----+-----+-----+
| 1 | Anika | P | anika@example.com | 0 |
| 2 | Ravi | Kumar | ravi@example.com | 1 |
| 3 | Sneha | Sharma | sneha@example.com | 0 |
| 4 | Amit | Rao | amit@example.com | 0 |
| 5 | Priya | Menon | priya@example.com | 0 |
| 6 | Anish | M | anish@example.com | 0 |
| 7 | Jeevith | R | jeevith@example.com | 0 |
| 8 | Ram | Panday | ram@example.com | 0 |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

read:

```
mysql> select * from customer;
+-----+-----+-----+-----+-----+
| customer_id | first_name | last_name | email | is_admin |
+-----+-----+-----+-----+-----+
| 1 | Anika | P | anika@example.com | 0 |
| 2 | Ravi | Kumar | ravi@example.com | 1 |
| 3 | Sneha | Sharma | sneha@example.com | 0 |
| 4 | Amit | Rao | amit@example.com | 0 |
| 5 | Priya | Menon | priya@example.com | 0 |
| 6 | Anish | M | anish@example.com | 0 |
| 7 | Jeevith | R | jeevith@example.com | 0 |
+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)
```

update:

```
mysql> update customer set last_name="Kapoor" where customer_id=2;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from customer;
+-----+-----+-----+-----+-----+
| customer_id | first_name | last_name | email | is_admin |
+-----+-----+-----+-----+-----+
| 1 | Anika | P | anika@example.com | 0 |
| 2 | Ravi | Kapoor | ravi@example.com | 1 |
| 3 | Sneha | Sharma | sneha@example.com | 0 |
| 4 | Amit | Rao | amit@example.com | 0 |
| 5 | Priya | Menon | priya@example.com | 0 |
| 6 | Anish | M | anish@example.com | 0 |
| 7 | Jeevith | R | jeevith@example.com | 0 |
| 8 | Ram | Panday | ram@example.com | 0 |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

delete:

```
mysql> delete from customer where customer_id=8;
Query OK, 1 row affected (0.06 sec)
```

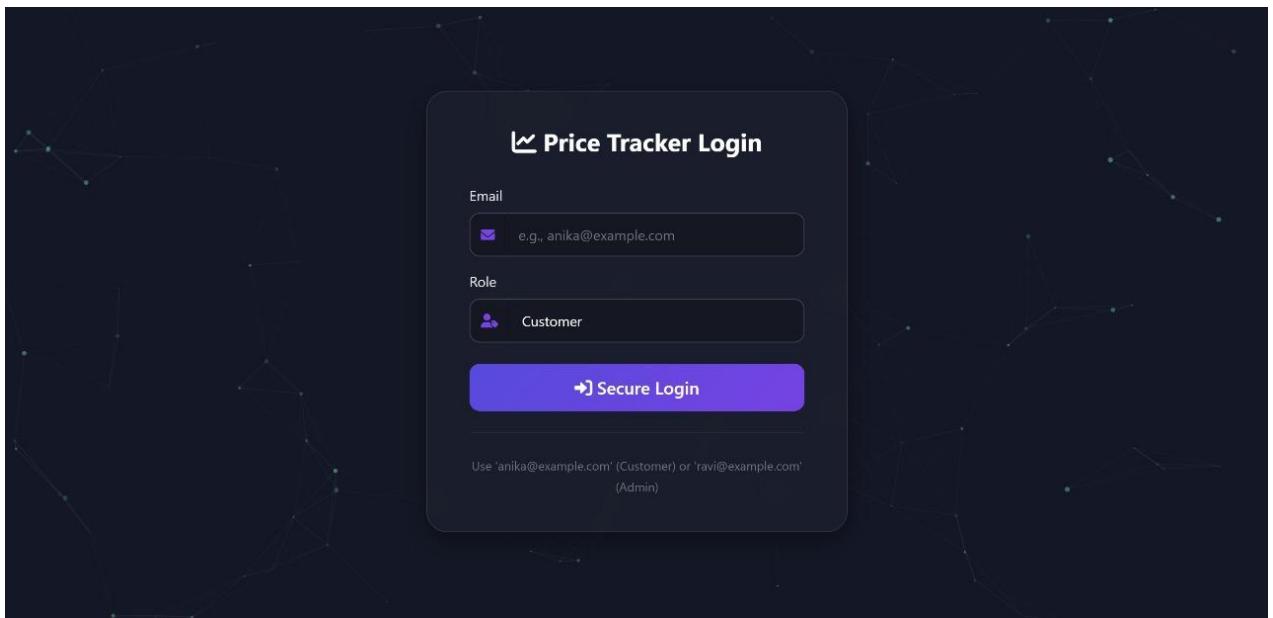
```
mysql> select * from customer;
```

customer_id	first_name	last_name	email	is_admin
1	Anika	P	anika@example.com	0
2	Ravi	Kapoor	ravi@example.com	1
3	Sneha	Sharma	sneha@example.com	0
4	Amit	Rao	amit@example.com	0
5	Priya	Menon	priya@example.com	0
6	Anish	M	anish@example.com	0
7	Jeevith	R	jeevith@example.com	0

7 rows in set (0.00 sec)

### List of functionalities/features of the application screenshots:

Customer login:



## Customer home:

Welcome Anika P!

Categories: All Products, Electronics, Fashion

Our Products

Sort By: Name (A-Z)

Product Type	Product Name	Brand	Description	Best Price
Shirt	Casual Shirt	FabWear	Cotton shirt for men	₹350.00
Laptop	Laptop Pro	Comptech	High performance business laptop	₹50000.00
Shoes	PUMA GRIP-X	PUMA	Light weight, flexible	₹10000.00
Shoes	PUMA Light	PUMA	Best Price: ₹10500.00	₹10500.00
Shoes	Running Shoes	Speedo	Comfortable sports running shoes	₹4000.00
Shoes	Running Shoes	Speedo	Comfortable sports running shoes	₹4000.00

E-Commerce Tracker

Home My Wishlist My Alerts My Expenses

Anika P Logout

### Casual Shirt

FabWear

Category: Fashion  
Cotton shirt for men

Availability: Available (Result from IsProductAvailable() function)

Add to Wishlist

Store Price Comparison

Store	Price	Discount	Final Price	Set Alert	Buy
MegaMart	₹350.00	1%	₹346.50	<input checked="" type="checkbox"/>	<button>Buy Now</button>
UrbanStyle	₹899.00	1%	₹890.01	<input checked="" type="checkbox"/>	<button>Buy Now</button>

Price History

From	To	Change
2025-11-11	₹500.00 → ₹350.00	↓ ₹150.00
2025-11-08	₹950.00 → ₹899.00	↓ ₹51.00

**E-Commerce Tracker** [Home](#) [My Wishlist](#) [My Alerts](#) [My Expenses](#) [Logout](#)

Anika P

Product added to wishlist! X

### Casual Shirt

FabWear

**Category:** Fashion  
Cotton shirt for men

**Availability:** Available

(Result from `IsProductAvailable()` function)

[Add to Wishlist](#)

**Store Price Comparison**  
Result from `GetBestPrice()` procedure

Store	Price	Discount	Final Price	Set Alert	Buy
UrbanStyle	₹899.00	1%	₹890.01	<span style="color: yellow;">Bell</span>	<div style="display: flex; align-items: center;"> <span style="border: 1px solid #ccc; padding: 2px 5px;">Home Delivery</span> <span style="margin-left: 10px;"><a href="#">Buy Now</a></span> </div>
MegaMart	₹350.00	1%	₹346.50	<span style="color: yellow;">Bell</span>	<div style="display: flex; align-items: center;"> <span style="border: 1px solid #ccc; padding: 2px 5px;">Home Delivery</span> <span style="margin-left: 10px;"><a href="#">Buy Now</a></span> </div>

**Price History**  
From `GetPriceHistory()` table (via Trigger)

2025-11-11	₹500.00 → ₹350.00	↓ ₹150.00
2025-11-08	₹950.00 → ₹899.00	↓ ₹51.00

**E-Commerce Tracker** [Home](#) [My Wishlist](#) [My Alerts](#) [My Expenses](#) [Logout](#)

Anika P

**My Wishlist**

**Join Query Demo:** This page joins `Wishlist`, `WishlistProduct`, `Product`, and `Inventory` tables.

Wishlist Name	Product	Brand	Best Price	Action
Anika Favorites	<a href="#">Casual Shirt</a>	FabWear	₹350.00	<a href="#">Remove</a>
Anika Favorites	<a href="#">PUMA Light</a>	PUMA	₹10500.00	<a href="#">Remove</a>

**E-Commerce Tracker** [Home](#) [My Wishlist](#) [My Alerts](#) [My Expenses](#) [Logout](#)

Anika P

**My Expense Tracker**

Total Tracked Spending

**₹51194.02**

Based on your simulated purchases

**Purchase History**

Date	Product	Store	Price Paid
2025-11-12 09:50	Casual Shirt	UrbanStyle	₹890.01
2025-11-11 10:56	Wireless Earbuds	GadgetHub	₹2464.00
2025-11-08 23:44	Casual Shirt	UrbanStyle	₹890.01
2025-11-07 08:16	Laptop Pro	TechWorld	₹44100.00
2025-11-07 08:13	Casual Shirt	UrbanStyle	₹950.00
2025-11-06 18:17	Casual Shirt	UrbanStyle	₹950.00
2025-11-06 18:16	Casual Shirt	UrbanStyle	₹950.00

The screenshot shows a user interface for an e-commerce tracker. At the top, there's a navigation bar with links for Home, My Wishlist, My Alerts, and My Expenses, along with a user profile for Anika P and a Logout button. The main content area displays a product page for "Smartphone X" by BrandX. Below the product details, there's a "Store Price Comparison" section comparing prices from TechWorld, showing an original price of ₹10000.00, a discount of 1%, and a final price of ₹9900.00. A "Set Price Alert" modal is open, prompting the user to set a price drop alert. The user has entered a target price of ₹9000. The modal also allows selecting a notification type (Email) and includes "Close" and "Set Alert" buttons.

When price meets user requirements:

This screenshot shows the same e-commerce tracker interface after a price drop. A green notification bubble titled "Price Drop!" appears in the center of the screen, stating: "Good news! One or more of your price alerts have been triggered: Smartphone X at TechWorld. Your target was ₹9000.00, and the price is now ₹8500.00!". The user can click a "Great, thanks!" button. In the background, the product listing for "Smartphone X" is visible with its new price of ₹8500.00. The rest of the interface, including categories like Electronics and Fashion, and a "Shoes" section, remains largely the same.

**E-Commerce Tracker** Home My Wishlist My Alerts My Expenses

**My Price Alerts**

**Trigger Demo:** The 'Status' of these alerts is updated automatically by the trigger\_price\_alert trigger when inventory prices change.

Product	Store	Target Price	Current Price	Status	Date Set	Action
Smartphone X BrandX	TechWorld	₹9000.00	₹10000.00	Active	2025-11-16	

## Alerts triggered:

**E-Commerce Tracker** Home My Wishlist My Alerts My Expenses

**My Price Alerts**

**Trigger Demo:** The 'Status' of these alerts is updated automatically by the trigger\_price\_alert trigger when inventory prices change.

Product	Store	Target Price	Current Price	Status	Date Set	Action
Smartphone X BrandX	TechWorld	₹9000.00	₹8500.00	viewed	2025-11-16	

## Price history, discount, final price seen in customer page after update by admin

**E-Commerce Tracker** Home My Wishlist My Alerts My Expenses

**Smartphone X**  
BrandX

**Category:** Mobiles  
Latest 5G Android phone

**Availability:** Available  
(Result from IsProductAvailable() function)

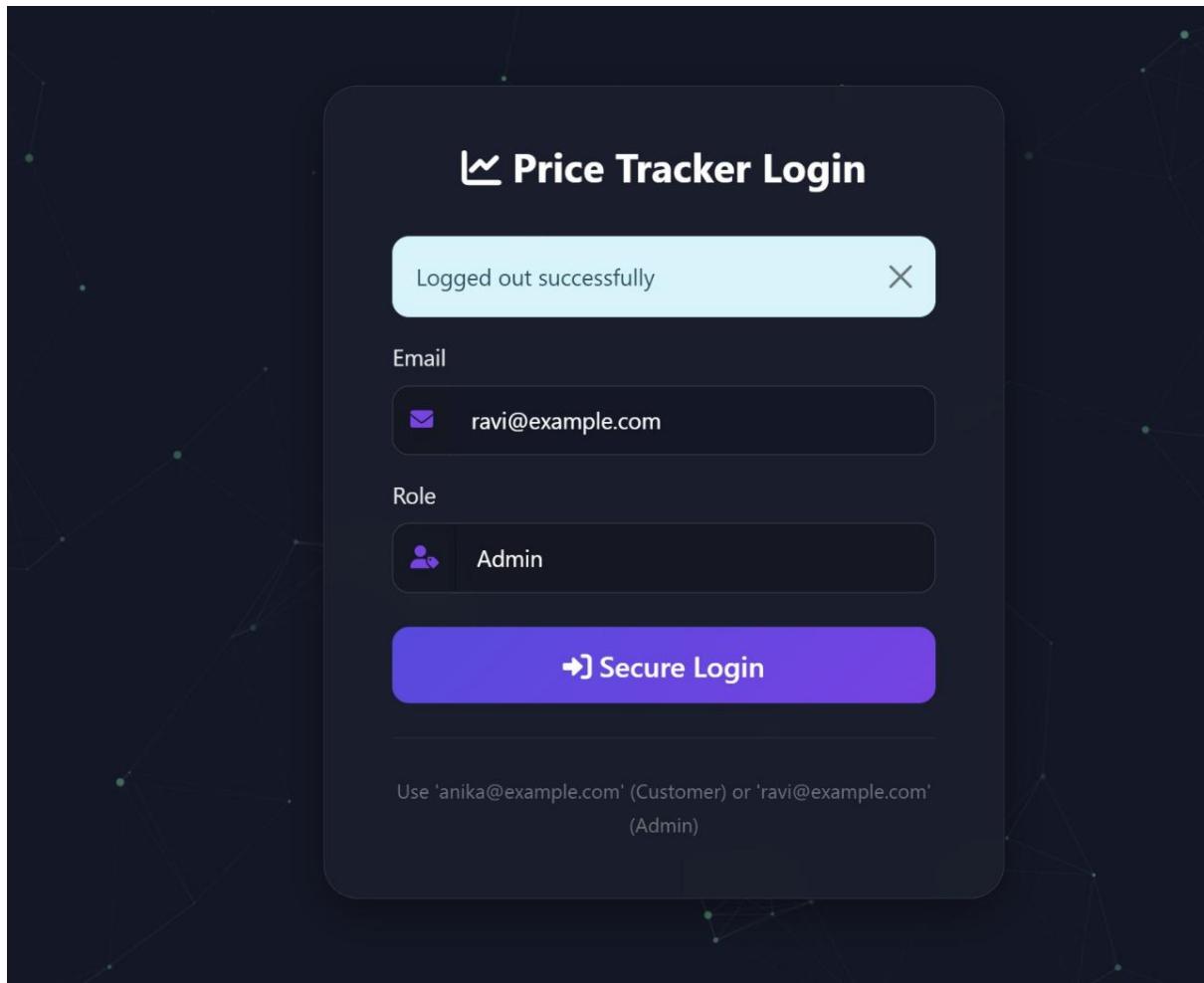
**Store Price Comparison**  
Result from procedure

Store	Price	Discount	Final Price	Set Alert	Buy
TechWorld	₹8500.00	1%	₹8415.00		

**Price History**  
From table (via Trigger)

2025-11-16 ₹10000.00 → ₹8500.00		₹1500.00
2025-11-10 ₹19000.00 → ₹10000.00		₹9000.00

Admin login:



Admin Dashboard:

The screenshot shows the 'E-Commerce Tracker' Admin Dashboard. The top navigation bar includes the logo, the user name 'Ravi Kapoor' with an 'Admin' badge, and a 'Logout' link. The dashboard features a sidebar on the left titled 'Admin Panel' with links for 'Dashboard', 'Users', 'Products', 'Inventory', and 'Analytics'. The main content area is titled 'Admin Dashboard' and includes a note about aggregate queries. It displays four summary cards: 'Total Customers' (7), 'Total Products' (7), 'Total Stores' (5), and 'Active Alerts' (2). Below these are two larger sections: 'Total Inventory' (144 Units) and 'Recent Price Changes' (No recent price changes). The bottom right corner shows a log entry: 'Logged by track\_price\_change Trigger'.

## Users visible to the admin:

The screenshot shows the Admin Panel of the E-Commerce Tracker. The left sidebar has links for Dashboard, Users (selected), Products, Inventory, and Analytics. The main area is titled "User Management". It has a "Add New User (CREATE)" form with fields for First Name, Last Name, Email, and Phone (Optional), followed by a "Add User" button. To the right is a table titled "Existing Users (READ / DELETE)" with columns for ID, First Name, Last Name, Email, and Action (with delete icons). The users listed are Anika, Ravi, Sneha, Amit, Priya, Anish, and Jeevith.

ID	First Name	Last Name	Email	Action
1	Anika	P	anika@example.com	
2	Ravi	Kapoor	ravi@example.com	
3	Sneha	Sharma	sneha@example.com	
4	Amit	Rao	amit@example.com	
5	Priya	Menon	priya@example.com	
6	Anish	M	anish@example.com	
7	Jeevith	R	jeevith@example.com	

## Products visible to the admin:

The screenshot shows the Admin Panel of the E-Commerce Tracker. The left sidebar has links for Dashboard, Users, Products (selected), Inventory, and Analytics. The main area is titled "Product Management". It has a "Add New Product" form with fields for Product Name, Brand, Description, Category (with a dropdown menu showing "-- Select Category --"), Initial Inventory (with fields for Store ID and Quantity/Price), and a "Add Product" button. To the right is a table titled "Existing Products" with columns for ID, Name, Brand, and Category. The products listed are Smartphone X, Laptop Pro, Casual Shirt, Running Shoes, Wireless Earbuds, PUMA GRIP-X, and PUMA Light.

ID	Name	Brand	Category
101	Smartphone X	BrandX	Mobiles
102	Laptop Pro	CompTech	Laptops
103	Casual Shirt	FabWear	Fashion
104	Running Shoes	Speedo	Shoes
105	Wireless Earbuds	SoundMax	Mobiles
106	PUMA GRIP-X	PUMA	Shoes
107	PUMA Light	PUMA	Shoes

## Inventory:

Trigger Demo: Updating price or quantity will fire `track_price_change`, `trigger_price_alert`, and `update_inventory_timestamp` triggers.

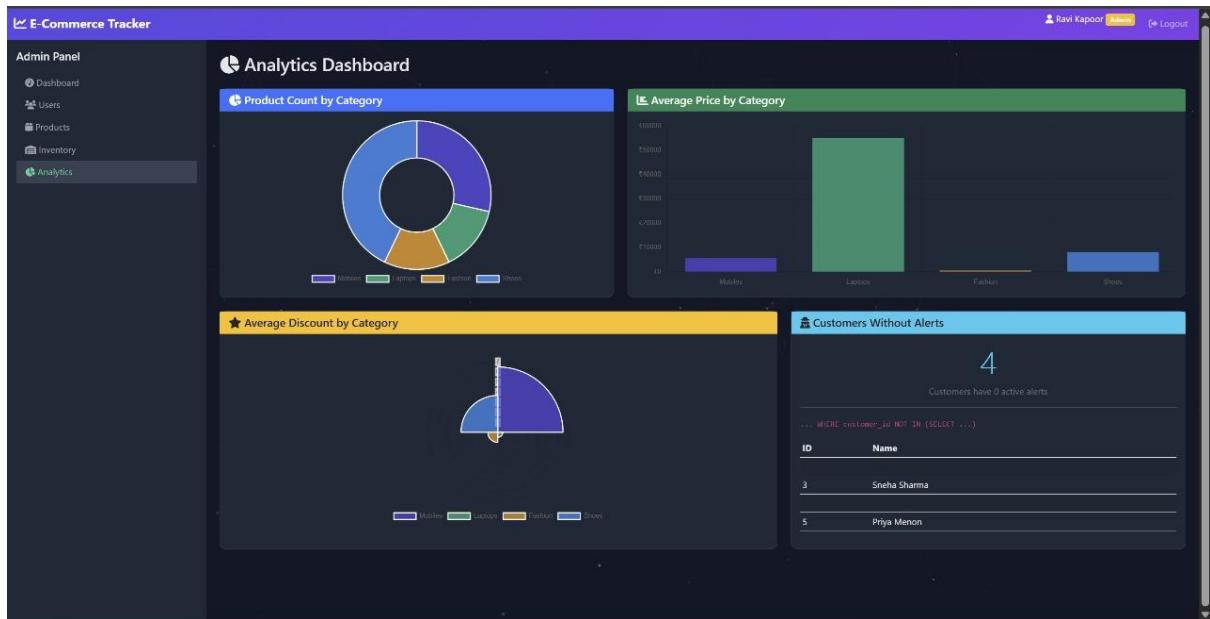
Inv. ID	Product ID	Store ID	Product	Store	Quantity	Price	Discount	Last Updated
3	103	4	Casual Shirt	UrbanStyle	27	₹899.00	1.0%	2025-11-12
5	105	2	Wireless Earbuds	GadgetHub	39	₹2800.00	12.0%	2025-11-11
6	106	2	PUMA GRIP-X	GadgetHub	9	₹10000.00	0.0%	2025-11-11
8	102	2	Laptop Pro	GadgetHub	6	₹5999.00	0.0%	2025-11-11
9	103	3	Casual Shirt	MegaMart	3	₹350.00	1.0%	2025-11-11
1	101	1	Smartphone X	TechWorld	10	₹10000.00	1.0%	2025-11-10
2	102	1	Laptop Pro	TechWorld	11	₹5000.00	1.0%	2025-11-10
7	107	2	PUMA Light	GadgetHub	15	₹10500.00	3.0%	2025-11-09
4	104	5	Running Shoes	ShoePlanet	24	₹4000.00	8.0%	2025-11-08

## Updated smartphone for alert verification

Inventory updated! Triggers executed.

Inv. ID	Product ID	Store ID	Product	Store	Quantity	Price	Discount	Last Updated
1	101	1	Smartphone X	TechWorld	12	₹8500.00	1.0%	2025-11-16
3	103	4	Casual Shirt	UrbanStyle	27	₹899.00	1.0%	2025-11-12
5	105	2	Wireless Earbuds	GadgetHub	39	₹2800.00	12.0%	2025-11-11
6	106	2	PUMA GRIP-X	GadgetHub	9	₹10000.00	0.0%	2025-11-11
8	102	2	Laptop Pro	GadgetHub	6	₹5999.00	0.0%	2025-11-11
9	103	3	Casual Shirt	MegaMart	3	₹350.00	1.0%	2025-11-11
2	102	1	Laptop Pro	TechWorld	11	₹5000.00	1.0%	2025-11-10
7	107	2	PUMA Light	GadgetHub	15	₹10500.00	3.0%	2025-11-09
4	104	5	Running Shoes	ShoePlanet	24	₹4000.00	8.0%	2025-11-08

## Analytics:



## Triggers, Procedures/Functions, Nested query, Join, Aggregate queries:

Attached in .sql file

## Code snippets for invoking the Procedures/Functions/Trigger:

Line 245:

```
228
229     @app.route('/customer/product/<int:product_id>')
230     @login_required
231     def customer_product_detail(product_id):
232         conn = get_db_connection()
233         cursor = conn.cursor(dictionary=True)
234
235         # Get product details
236         cursor.execute("""
237             SELECT p.*, c.category_name
238             FROM Product p
239             JOIN Category c ON p.category_id = c.category_id
240             WHERE p.product_id = %s
241             """, (product_id,))
242         product = cursor.fetchone()
243
244         # Call ComparePrices procedure
245         cursor.callproc('ComparePrices', [product_id])
```

Line 786:

```
778     @app.route('/admin/auto_restock')
779     @admin_required
780     def admin_auto_restock():
781         conn = get_db_connection()
782         cursor = conn.cursor()
783
784     try:
785         # Call the AutoRestock procedure
786         cursor.callproc('AutoRestock')
787         conn.commit()
788         flash('Auto-restock procedure executed!', 'success')
789     except Error as e:
790         flash(f'Error running auto-restock: {e}', 'danger')
791
792     cursor.close()
793     conn.close()
794
795     return redirect(url_for('admin_inventory'))
```

Line 875:

```
859     @app.route('/admin/add_product', methods=['POST'])
860     @admin_required
861     def admin_add_product():
862         product_name = request.form.get('product_name')
863         description = request.form.get('description')
864         brand = request.form.get('brand')
865         category_id = request.form.get('category_id')
866         store_id = request.form.get('store_id')
867         quantity = request.form.get('quantity')
868         price = request.form.get('price')
869
870         conn = get_db_connection()
871         cursor = conn.cursor()
872
873     try:
874         # Call AddNewProduct procedure
875         cursor.callproc('AddNewProduct',
876                         [product_name, description, brand, category_id, store_id, quantity, price])
877
878         conn.commit()
879         flash('Product added using stored procedure!', 'success')
880     except Error as e:
881         flash(f'Error adding product: {e}', 'danger')
882
883     cursor.close()
884     conn.close()
```

[Github repo link:](#)

<https://github.com/anikap2801/E-Commerce-Price-Stock-Tracker-Across-Multiple-Stores>