**Project Title**:Ultimate E-commerce DB

**Project Summary:**

"Ultimate E-commerce DB" is a seller-focused e-commerce management platform that helps online merchants manage their product listings, track inventory levels, monitor storage usage, and analyze sales performance. It is built on a real-world dataset to demonstrate relational database design, complex queries, and data-driven analytics.

**Project Description:**

Our project focuses on simulating the workflow of a seller's e-commerce dashboard. The system is designed to help sellers maintain product catalogs, monitor their orders and inventory, and review their overall sales performance.

The application will allow sellers to manage product information by adding, updating, or removing product listings. They will also be able to view orders placed by customers, track order statuses, and identify best-selling items. Additionally, the system will provide straightforward performance summaries, such as sales trends over time and category-based sales distribution.

Unlike more ambitious dashboards that include features such as interactive mapping or natural-language sentiment analysis, our focus at this stage is to keep the project realistic, clear, and achievable. The emphasis is on relational database design, structured querying, and practical seller-oriented use cases.

**Clarification of TA Approval:**

Our topic is not in the category that needs TA approval.

**Usefulness:**

This project is useful because it provides a unified view of the data that sellers need to make decisions. The platform integrates catalog data and transactional data, so sellers can both manage their products and evaluate sales performance in one place. Unlike commercial dashboards (Shopify, Amazon Seller Central), this application is open, educational, and built to demonstrate best practices in database normalization and SQL query design.

By keeping the scope focused on essential seller operations (product management, order monitoring, and performance summaries), we ensure that the application is both practically useful and academically rigorous.

**Realness:**

Our project uses two complementary real-world datasets:

1. **Olist E-Commerce Dataset (SQLite format, ~100k orders, ~100k customers, ~3k sellers, ~30 categories)**
    - Captures the transactional side of e-commerce: customer information, seller information, product IDs, orders, order items, payments, and reviews.
    - Provides order-level and relationship-level data that reflects actual operations of a marketplace.
2. **BigBasket Entire Product List (~28k datapoints, CSV format, Kaggle)**
    - Captures the product catalog side: product names, categories, prices, and detailed descriptions from an online grocery retailer.
    - Complements Olist by providing a broader and more descriptive set of product attributes than Olist's limited product table.

**Functionality:**

**Product Management**

- Browse a **combined catalog** (Olist products enriched with BigBasket attributes); paginate, sort, and **filter by category/price/availability**.
- **CRUD**: create/edit/deactivate products; validate required fields; prevent edits that would break referential integrity (e.g., editing a product ID used by orders).
- Maintain **versioned product attributes** (e.g., price changes recorded; past orders keep historical prices).

**Order Tracking**

- **Search & filter** orders by ID, date range, status (created, paid, shipped, delivered, canceled), and payment method.

- **Order detail view**: items, unit prices at time of sale, quantities, payment records, and current status.
- **Lifecycle actions**: update status (e.g., created → paid → shipped), add/remove items (pre-ship), and cancel/refund with clear rules.

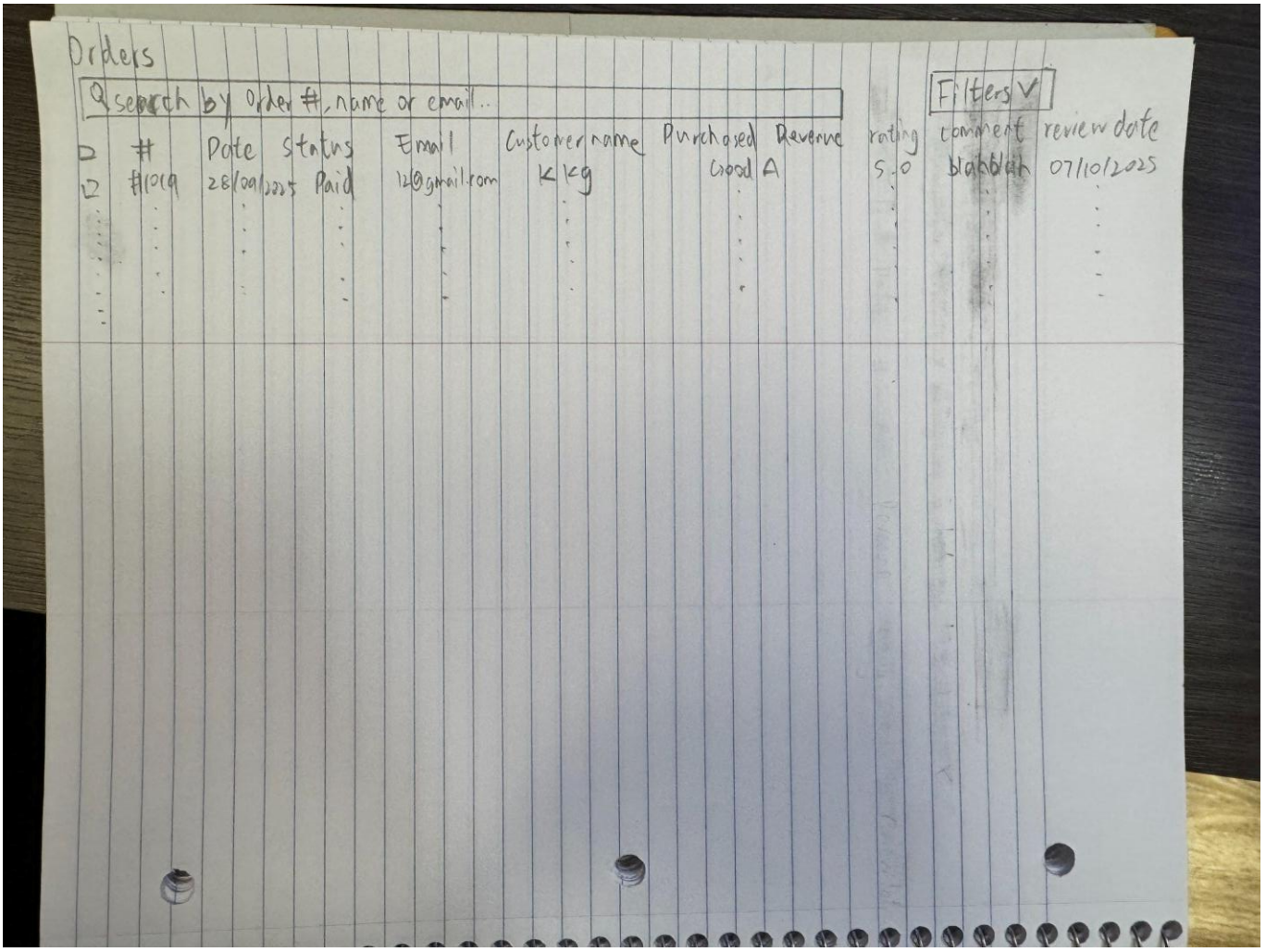
**Sales Performance Summaries**

- Metrics: **total revenue**, **orders count**, **top-selling products**, **category sales distribution**, and **period-over-period comparisons**.
- **Lightweight charts** (bar/line) and **CSV export** for reports; no heavy ML/geo features.

**Dynamic Updates & Consistency**

- **ACID transactions** for order/item changes; **inventory adjusted atomically** with item add/remove/cancel to prevent overselling.
- **Optimistic locking**: each mutable row carries a version; updates include the expected version—mismatches return **409 Conflict** with the latest row to retry/merge.
- **Idempotent endpoints** (e.g., POST /orders with Idempotency-Key) to avoid duplicates on client retries.
- **Incremental rollups**: update sales_daily (orders, qty, revenue) **at status transitions** (e.g., when order becomes paid); nightly reconciliation corrects edge cases.
- **Audit logging**: record who changed what and when (entity, action, timestamp, minimal diff), enabling traceability.
- **UI freshness**: dashboard **polls for deltas** (?since=<timestamp>) every few seconds to reflect new orders/edits without reloading entire pages.

**Low-fidelity UI mockup:**




**Project Work Distribution:**

**Backend & Database (Member A):** Responsible for exploring the dataset schema, designing optimized SQL queries, ensuring normalization, and implementing business logic such as discounts or performance aggregations.

**Backend Integration (Member B):** Develops RESTful API endpoints (using Flask/Django) to connect the relational database with the frontend. This member also ensures that CRUD operations for products and orders are fully functional.

**Frontend Development (Member C):** Designs and implements the web interface for product management and order tracking, ensuring usability and responsiveness. This includes forms for adding/editing products and pages for order filtering.

**Analytics & Creative Component (Member D):** Develops the analytics dashboard with advanced visualization libraries (e.g., D3.js, Plotly) and integrates the geospatial mapping feature. Also implements the profitability simulator as the creative extension.

.