Shopify Data Ingestion & Dashboard

1. Assumptions

- Each Shopify store = one tenant in our system. Multi-tenancy is handled with tenant_id foreign key in all core tables.
- Neon PostgreSQL is used as the primary RDBMS, ensuring SQL-compliant queries and scalable cloud hosting.
- Webhooks are the primary ingestion method (customers/create, orders/create, products/update, carts/update).
- Webhook payload integrity is verified using Shopify's HMAC header and SHOPIFY_SECRET.
- Express.js backend with JWT-based authentication secures API routes for tenants.
- React frontend (localhost:5173 during dev) interacts only with the backend's REST APIs.
- Email verification for new users is implemented using Nodemailer and SMTP.
- Retries of failed webhooks are not yet implemented (would be required in production).

2. High-Level Architecture

The system architecture connects Shopify webhooks to an Express backend, which stores normalized data into Neon PostgreSQL. A React frontend consumes backend APIs for metrics and dashboards.

[Diagram Placeholder: React Frontend → Express Backend → PostgreSQL, with Shopify Webhooks feeding into Backend]

3. Data Models

- Tenants: id, store_name, domain, shopify_token
- Users: id, email, password_hash, is_verified, tenant_id
- Customers: id (local), shopify_id (BIGINT), tenant_id, email, name
- Products: id (local), shopify_id (BIGINT), tenant_id, title, sku, price_cents
- Orders: id (local), shopify_id (BIGINT), tenant_id, order_number, customer_id (FK → customers.id)
- Events: Raw webhook payloads stored for audit trail

4. APIs

- Authentication: POST /api/auth/register, POST /api/auth/login
- Webhooks: POST /api/webhook → verifies HMAC and dispatches to upsert services
- Metrics: GET /api/metrics/customers, /api/metrics/orders, /api/metrics/products
- Tenants: POST /api/tenants, GET /api/tenants

5. Production Readiness / Next Steps

- Add retry queue for failed webhooks (e.g., Redis/BullMQ).
- Add more indexes on (tenant_id, shopify_id) for faster upserts.
- Enforce HTTPS, rotate JWT & SMTP secrets regularly.
- Add logging and request tracing with tools like Winston or Datadog.
- Deploy backend to Render/Railway and expose via HTTPS domain.