Database Documentation

This document contains clear overview of database schema, with each table’s role, its columns, and how they link together.

**Table - categories**

**Purpose**  
Holds high-level classifications for products (for example Electronics, Clothing, Home, Fashion).

**Columns**

* **id** (PK, int) — unique identifier
* **name** (varchar(50), unique, not null) — category label

**Relationships**

* One category → many products (products.category\_id → categories.id)

**Table - products**

**Purpose**  
Stores each item available for sale, along with its base price and category.

**Columns**

* **id** (PK, int) — unique product identifier
* **name** (varchar(100), not null) — product title
* **category\_id** (FK → categories.id) — links to its category
* **price** (float, not null) — retail price per unit

**Relationships**

* Many products → one category
* One product → one inventory record (inventory.product\_id → products.id)
* One product → many sales records (sales.product\_id → products.id)

**Table - inventory**

**Purpose**  
Tracks current stock levels for each product, plus when that level was last updated.

**Columns**

* **id** (PK, int) — unique identifier
* **product\_id** (FK unique → products.id) — each product appears here once
* **quantity** (int, default 0) — units on hand
* **last\_updated** (datetime, not null) — timestamp of most recent change

**Relationships**

* One inventory record → one product
* Back-reference: product.inventory

**Table - sales**

**Purpose**  
Records every transaction, including how many units sold, total price, and sale date.

**Columns**

* **id** (PK, int) — unique sale identifier
* **product\_id** (FK → products.id) — which product was sold
* **quantity** (int, not null) — units sold in this transaction
* **total\_price** (float, not null) — sum paid (quantity × unit price at sale)
* **sold\_at** (datetime, not null, indexed) — when the sale occurred

**Relationships**

* Many sales → one product
* Back-reference: product.sales

**How they fit together**

1. **categories** define product groupings
2. **products** reference categories, they each have a single inventory row and many sales rows
3. **inventory** gives real-time stock for each product
4. **sales** logs every purchase event

All foreign keys are indexed for fast joins, and sales.sold\_at is indexed to speed up date-range queries (daily, weekly, monthly or annual reports). Tables are fully normalized (no redundant fields), ensuring consistency when you update price or category in one place only.

**Database schema**

| **Table** | **Columns** | **Notes** |
| --- | --- | --- |
| categories | id PK, name (unique) | Product classification |
| products | id PK, name, category\_id FK, price | Each item for sale |
| inventory | id PK, product\_id FK unique, quantity, last\_updated | Tracks stock levels |
| sales | id PK, product\_id FK, quantity, total\_price, sold\_at | Recorded transactions |