```python
import json
import os
import datetime
from collections import defaultdict

DATA_FILE = "shop_data.json"


def now_iso():
   return datetime.datetime.now().isoformat(timespec="seconds")


def load_data():
   if not os.path.exists(DATA_FILE):
       return {"products": {}, "sales": []}
   with open(DATA_FILE, "r", encoding="utf-8") as f:
       return json.load(f)


def save_data(data):
   with open(DATA_FILE, "w", encoding="utf-8") as f:
       json.dump(data, f, indent=2, ensure_ascii=False)


def generate_product_id(data):
   existing = data["products"].keys()
   i = 1
   while True:
       pid = f"P{i:04d}"
       if pid not in existing:
           return pid
       i += 1


def input_nonempty(prompt):
   while True:
       v = input(prompt).strip()
       if v:
           return v
       print("Please enter a non-empty value.")


def input_float(prompt, allow_zero=False):
   while True:
       try:
           v = float(input(prompt).strip())
           if not allow_zero and v <= 0:
               print("Enter a number greater than 0.")
               continue
           return v
       except ValueError:
           print("Invalid number. Try again.")


def input_int(prompt, allow_zero=True):
   while True:
       try:
           v = int(input(prompt).strip())
           if not allow_zero and v <= 0:
               print("Enter an integer greater than 0.")
               continue
           return v
       except ValueError:
           print("Invalid integer. Try again.")


def add_product(data):
   print("\n== Add Product ==")
   name = input_nonempty("Name: ")
   price = input_float("Selling Price (per unit): ")
   cost = input_float("Purchase Cost (per unit): ")
   qty = input_int("Quantity in stock: ", allow_zero=True)
   pid = generate_product_id(data)
   data["products"][pid] = {
       "id": pid,
       "name": name,
       "price": price,
       "cost": cost,
       "qty": qty,
       "created_at": now_iso(),
   }
   save_data(data)
   print(f"Product added with ID {pid}.")


def update_product(data):
   print("\n== Update Product ==")
   pid = input_nonempty("Product ID: ").upper()
   p = data["products"].get(pid)
   if not p:
       print("Product not found.")
       return
   print(f"Editing {pid} — {p['name']}")
   name = input(f"Name [{p['name']}]: ").strip() or p["name"]
   price = input(f"Selling Price [{p['price']}]: ").strip()
   cost = input(f"Purchase Cost [{p['cost']}]: ").strip()
   qty = input(f"Quantity in stock [{p['qty']}]: ").strip()
   try:
       price = float(price) if price else p["price"]
       cost = float(cost) if cost else p["cost"]
       qty = int(qty) if qty else p["qty"]
   except ValueError:
       print("Invalid numeric input — update cancelled.")
       return
   p.update({"name": name, "price": price, "cost": cost, "qty": qty, "updated_at":
now_iso()})
   save_data(data)
   print("Product updated.")


def delete_product(data):
   print("\n== Delete Product ==")
   pid = input_nonempty("Product ID to delete: ").upper()
   if pid not in data["products"]:
       print("Product not found.")
       return
   confirm = input(f"Are you sure you want to delete {pid}
({data['products'][pid]['name']})? [y/N]: ").lower()
   if confirm == "y":
       del data["products"][pid]
       save_data(data)
       print("Product deleted.")
   else:
       print("Cancelled.")


def list_products(data, low_stock_only=False):
   print("\n== Products ==")
   products = list(data["products"].values())
   if not products:
       print("No products in inventory.")
       return
   products.sort(key=lambda x: x["id"])
   print(f"{'ID':6} {'Name':25} {'Price':8} {'Cost':8} {'Qty':5}")
   print("-" * 60)
   for p in products:
       if low_stock_only and p["qty"] > 5:
           continue
       print(f"{p['id']:6} {p['name'][:25]:25} {p['price']:8.2f} {p['cost']:8.2f}
{p['qty']:5d}")


def create_bill(data):
   print("\n== Create Bill / Sell Items ==")
   cart = []
   while True:
       pid = input("Product ID (or 'done'): ").strip()
       if pid.lower() in ("done", "d", ""):
           break
       pid = pid.upper()
       prod = data["products"].get(pid)
       if not prod:
           print("Product not found.")
           continue
       print(f"{prod['name']} — Price {prod['price']}, Available {prod['qty']}")
       qty = input_int("Quantity to sell: ", allow_zero=False)
       if qty > prod["qty"]:
           print("Not enough stock. Try smaller quantity.")
           continue
       cart.append({"id": pid, "name": prod["name"], "price": prod["price"], "qty":
qty, "cost": prod["cost"]})
   if not cart:
       print("No items in cart. Cancelled.")
       return
   # Calculate totals
   subtotal = sum(item["price"] * item["qty"] for item in cart)
   tax = round(subtotal * 0.12, 2)  # example 12% tax
   discount = 0.0
   # optional discount based on subtotal
   if subtotal > 1000:
       discount = round(subtotal * 0.05, 2)
   total = round(subtotal + tax - discount, 2)

   # Confirm and save sale
   print("\n--- Bill Preview ---")
   for it in cart:
       print(f"{it['name'][:30]:30} x{it['qty']:3d} @ {it['price']:7.2f} =
{it['price']*it['qty']:8.2f}")
   print(f"{'Subtotal':40} {subtotal:8.2f}")
   print(f"{'Tax (12%)':40} {tax:8.2f}")
   print(f"{'Discount':40} {-discount:8.2f}")
   print(f"{'Total':40} {total:8.2f}")
   confirm = input("Confirm sale and reduce stock? [y/N]: ").lower()
   if confirm != "y":
       print("Sale cancelled.")
       return

   # Reduce stock and record sale
   for it in cart:
       data["products"][it["id"]]["qty"] -= it["qty"]
   sale = {
       "id": f"S{len(data['sales'])+1:05d}",
       "datetime": now_iso(),
       "items": cart,
       "subtotal": subtotal,
       "tax": tax,
       "discount": discount,
       "total": total,
   }
   data["sales"].append(sale)
   save_data(data)
   print(f"Sale recorded with ID {sale['id']}. Total: {total:.2f}")


def view_sales(data):
   print("\n== Sales History ==")
   if not data["sales"]:
       print("No sales yet.")
       return
   for s in data["sales"][-20:]:  # show last 20 by default
       print(f"{s['id']} {s['datetime']}  total={s['total']:.2f}
items={len(s['items'])}")


def report(data):
   print("\n== Report ==")
   total_revenue = sum(s["total"] for s in data["sales"])
   total_profit = 0.0
   product_sold = defaultdict(int)
   for s in data["sales"]:
       for item in s["items"]:
           product_sold[item["id"]] += item["qty"]
           total_profit += (item["price"] - item["cost"]) * item["qty"]
   best_selling = sorted(product_sold.items(), key=lambda x: x[1], reverse=True)
   print(f"Total Revenue: {total_revenue:.2f}")
   print(f"Estimated Profit: {total_profit:.2f}")
   if best_selling:
       pid, qty = best_selling[0]
       pname = data["products"].get(pid, {}).get("name", "(deleted)")
       print(f"Best selling product: {pid} {pname} — {qty} units sold")
   low_stock = [p for p in data["products"].values() if p["qty"] <= 5]
   print(f"Products low in stock (<=5): {len(low_stock)}")
   if low_stock:
       for p in low_stock:
           print(f" - {p['id']} {p['name']} qty={p['qty']}")


def seed_demo_data(data):
   # only seed if no products exist
   if data["products"]:
       return
   sample = [
       ("P1001", {"id": "P1001", "name": "Notebook", "price": 45.0, "cost": 30.0,
"qty": 50, "created_at": now_iso()}),
       ("P1002", {"id": "P1002", "name": "Pen", "price": 10.0, "cost": 4.0, "qty":
200, "created_at": now_iso()}),
       ("P1003", {"id": "P1003", "name": "Bottle", "price": 120.0, "cost": 80.0,
"qty": 20, "created_at": now_iso()}),
   ]
   for pid, obj in sample:
       data["products"][pid] = obj
   save_data(data)


def main_menu():
   data = load_data()
   seed_demo_data(data)
   MENU = {
       "1": ("Add product", add_product),
       "2": ("Update product", update_product),
       "3": ("Delete product", delete_product),
       "4": ("List products", lambda d: list_products(d, low_stock_only=False)),
       "5": ("List low-stock products", lambda d: list_products(d,
low_stock_only=True)),
       "6": ("Create bill / Sell items", create_bill),
       "7": ("View sales history", view_sales),
       "8": ("Report (revenue/profit/best-selling)", report),
       "9": ("Save & Exit", None),
   }
```

```python
    while True:
        print("\n=== Shop Management ===")
        for k, (title, _) in MENU.items():
            print(f"{k}. {title}")
        choice = input("Choose an option: ").strip()
        if choice == "9":
            save_data(data)
            print("Data saved. Goodbye.")
            break
        action = MENU.get(choice)
        if action:
            _, func = action
            try:
                func(data)
            except Exception as e:
                print("Error during operation:", e)
        else:
            print("Invalid choice. Try again.")


if __name__ == "__main__":
    main_menu()
```