DI31003 - Database Systems

Coursework 1 - Cover Sheet

TEAM NUMBER：Group 8

COMPANY NAME：Summit Gear & Adventures

COMPANY SLOGAN: Well-equipped, Exploring the Infinite.

TEAM MEMBERS: Tianbo Cao 2617401@dundee.ac.uk
Jianyuan Gui 2617459@dundee.ac.uk
Qihao Tang 2617287@dundee.ac.uk
Zixuan Wang 2617345@dundee.ac.uk
Boran Zhang 2617285@dundee.ac.uk
Xuan Zhu 2617341@dundee.ac.uk

Company description / Specification report Word count (900-1000): 974	Number of pages: 2-3 pages
E-R diagram	Number of pages: 4-5 pages
User interface designs	Number of pages: 14 pages
SQL CREATE statements	Number of pages: 6-7 pages

	Total pages: 29 pages

We confirm that the team members have read and understood the University policy on Academic Misconduct.

Company description / Specification report

This report claims the structure and operations of the company and how our design was implemented based on this.

Part 1. Company Profile

The company is named “Summit Gear & Adventures”. It is a professional outdoor sports equipment chain brand in the UK. Since its establishment in 2018, it has built a network of physical stores in five core cities in Scotland. The company takes "Well-equipped, Exploring the Infinite" as its core concept and slogan of striving, and provides equipment solutions for outdoor sports enthusiasts of all ages from 18 to 65. Our company supports both online platform ordering and offline store purchasing.

The organizational structure adopts a two-level management model:

The headquarters management team consists of the business manager, inventory manager and HR manager, undertaking the functions of strategic decision-making, resource coordination and standardized control. Each branch store implements standardized configuration, which is equipped with several staffs. This structure not only ensures the headquarters' centralized control over core resources but also grants branches the space to operate flexibly, thus forming a management advantage of strategic unity and execution flexibility.

Part 2. Operational Mechanism and its Impacts on Design

2.1 Organizational Structure and Data Foundation

The two-level management model of the company directly determines the structure design of the database. The database needs to maintain this hierarchical structure through entity relationships to ensure the effective implementation of data isolation and access control.

The branch information table serves as the top-level structure anchor of the database, fully documenting the static information of five physical stores, including the branch number (primary key), branch name, city of location, and specific address. This table is the basis for associating all branch-related transaction data, accurately reflecting the company's distributed operation characteristics through a "one-to-many" relationship.

The employee information table stores detailed information of all employees. A key design is to set the "branch number" as a foreign key, establishing a mandatory association of "one branch having multiple employees, and each employee belonging to only one branch". This association not only supports precise data reporting but also provides a foundation for implementing job-based permission control.

2.2 Product and Supply Chain Management

This part of the business covers the entire life cycle of products from procurement to inventory management. The database design needs to achieve integrated management throughout the entire process.

a. Procurement and Classification System:

The supplier table independently manages information of multiple suppliers. By using the "supplier number" (primary key), company name, and contact person, etc., the supplier data is separated from the product data to avoid information redundancy.

The main product table, as the core record of all goods, establishes a one-to-many relationship with the primary suppliers through the "supplier number" (foreign key).

The product classification table adopts a hierarchical design, achieving a multi-level classification structure through self-referencing of the "parent category number". This design enables enterprises to flexibly adjust

the classification hierarchy without modifying the database structure. Products are associated with specific categories through the "category number" (foreign key).

b. Procurement process management:

The procurement and replenishment process, which is uniformly managed by the headquarters' purchasing manager, is implemented through the following table:

The purchase order table, as the core of the procurement business, records the overall information of the order, including the supplier (foreign key), the employee who placed the order (foreign key), and the receiving branch (foreign key). The "order status" field tracks the progress of the procurement execution throughout the process.

The purchase detail table serves as an associative table to address the many-to-many relationship where "one purchase order contains multiple products and one product can appear in multiple purchase orders".

c. Inventory management mechanism:

Each of the five branches manages its own independent inventory.

The inventory record table, as the core of inventory management, resolves the many-to-many relationship between branches and products (one branch can store multiple products, and one product can be stored in multiple branches). This table records the actual inventory quantity of each product in a specific branch by associating the "branch number" and "product number". To ensure data accuracy, a unique constraint is set for the combination of (product number, branch number) to prevent duplicate inventory records for the same product in the same branch.

2.3 Sales and Customer Relationship

This module supports the company's core revenue generation process - selling products to customers. The design focus is on ensuring transaction integrity and the accurate traceability of historical data.

The customer information table stores all customer records. To support customer relationship management and accurately identify repeat customers, unique constraints are set on the "mobile phone number" field. This key design prevents the generation of duplicate customer records, ensuring that each customer has a unique and reliable identification marker.

a. Sales transaction process:

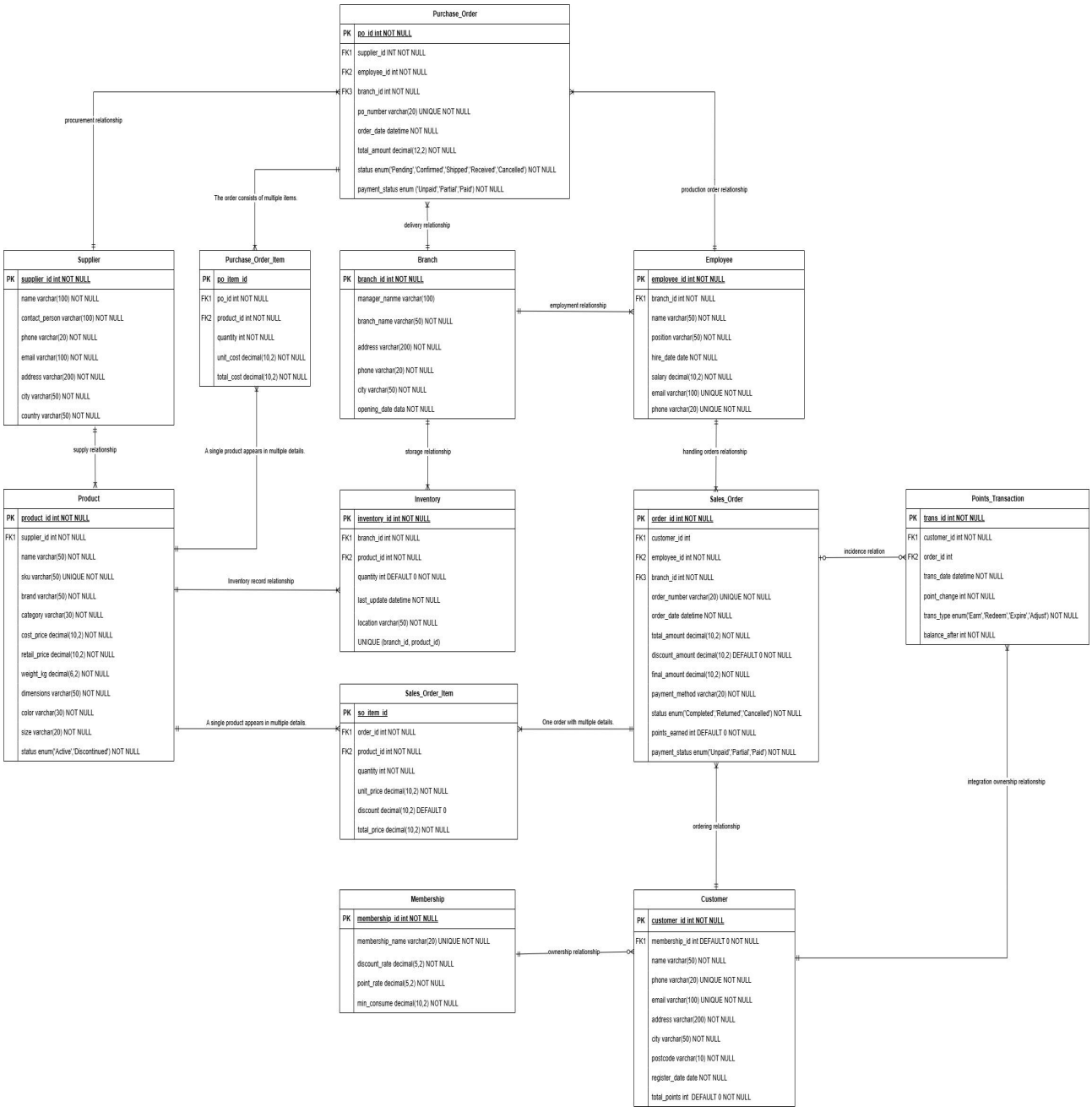
The sales order table, serving as the transaction header table, records the key information of each sale, and is associated with customer ID, employee ID, and branch ID.

The order detail table serves as an associative table to address the many-to-many relationship of "one sales order containing multiple products". This table links the "order number" and the "product number" and specifically records the "unit price of the commodity" at the time of sale. Storing the unit price in the order detail is a key design to ensure historical accuracy. Even if the retail price in the product master table changes later, the original transaction price is still fully retained, ensuring the accuracy of all historical sales reports.

b. Performance optimization design:

The sales order table particularly includes summary fields such as "total order amount", "discount amount", and "actual received amount". The discount amount is determined by the user's membership status stored in the membership level table and the usage of points based on the user's point transaction table. By pre-storing these summary values, the database can quickly generate various financial reports. This design strikes a balance between normalization principles and query performance.

E-R diagram


Purchase_Order
PK po_id int NOT NULL
FK1 supplier_id INT NOT NULL
FK2 employee_id int NOT NULL
FK3 branch_id int NOT NULL
po_number varchar(20) UNIQUE NOT NULL
order_date datetime NOT NULL
total_amount decimal(12,2) NOT NULL
status enum('Pending','Confirmed','Shipped','Received','Cancelled') NOT NULL
payment_status enum ('Unpaid','Partial','Paid') NOT NULL
procurement relationship
The order consists of multiple items.
delivery relationship
production order relationship
Supplier
PK supplier_id int NOT NULL
name varchar(100) NOT NULL
contact_person varchar(100) NOT NULL
phone varchar(20) NOT NULL
email varchar(100) NOT NULL
address varchar(200) NOT NULL
city varchar(50) NOT NULL
country varchar(50) NOT NULL
Purchase_Order_Item
PK po_item_id
FK1 po_id int NOT NULL
FK2 product_id int NOT NULL
quantity int NOT NULL
unit_cost decimal(10,2) NOT NULL
total_cost decimal(10,2) NOT NULL
Branch
PK branch_id int NOT NULL
manager_nanme varchar(100)
branch_name varchar(50) NOT NULL
address varchar(200) NOT NULL
phone varchar(20) NOT NULL
city varchar(50) NOT NULL
opening_date data NOT NULL
employment relationship
Employee
PK employee_id int NOT NULL
FK1 branch_id int NOT NULL
name varchar(50) NOT NULL
position varchar(50) NOT NULL
hire_date date NOT NULL
salary decimal(10,2) NOT NULL
email varchar(100) UNIQUE NOT NULL
phone varchar(20) UNIQUE NOT NULL
supply relationship
A single product appears in multiple details.
storage relationship
handling orders relationship
Product
PK product_id int NOT NULL
FK1 supplier_id int NOT NULL
name varchar(50) NOT NULL
sku varchar(50) UNIQUE NOT NULL
brand varchar(50) NOT NULL
category varchar(30) NOT NULL
cost_price decimal(10,2) NOT NULL
retail_price decimal(10,2) NOT NULL
weight_kg decimal(6,2) NOT NULL
dimensions varchar(50) NOT NULL
color varchar(30) NOT NULL
size varchar(20) NOT NULL
status enum('Active','Discontinued') NOT NULL
Inventory
PK inventory_id int NOT NULL
FK1 branch_id int NOT NULL
FK2 product_id int NOT NULL
quantity int DEFAULT 0 NOT NULL
last_update datetime NOT NULL
location varchar(50) NOT NULL
UNIQUE (branch_id, product_id)
Inventory record relationship
Sales_Order
PK order_id int NOT NULL
FK1 customer_id int
FK2 employee_id int NOT NULL
FK3 branch_id int NOT NULL
order_number varchar(20) UNIQUE NOT NULL
order_date datetime NOT NULL
total_amount decimal(10,2) NOT NULL
discount_amount decimal(10,2) DEFAULT 0 NOT NULL
final_amount decimal(10,2) NOT NULL
payment_method varchar(20) NOT NULL
status enum('Completed','Returned','Cancelled') NOT NULL
points_earned int DEFAULT 0 NOT NULL
payment_status enum('Unpaid','Partial','Paid') NOT NULL
incidence relation
Points_Transaction
PK trans_id int NOT NULL
FK1 customer_id int NOT NULL
FK2 order_id int
trans_date datetime NOT NULL
point_change int NOT NULL
trans_type enum('Earn','Redeem','Expire','Adjust') NOT NULL
balance_after int NOT NULL
A single product appears in multiple details.
Sales_Order_Item
PK so_item_id
FK1 order_id int NOT NULL
FK2 product_id int NOT NULL
quantity int NOT NULL
unit_price decimal(10,2) NOT NULL
discount decimal(10,2) DEFAULT 0
total_price decimal(10,2) NOT NULL
One order with multiple details.
integration ownership relationship
ordering relationship
Membership
PK membership_id int NOT NULL
membership_name varchar(20) UNIQUE NOT NULL
discount_rate decimal(5,2) NOT NULL
point_rate decimal(5,2) NOT NULL
min_consume decimal(10,2) NOT NULL
ownership relationship
Customer
PK customer_id int NOT NULL
FK1 membership_id int DEFAULT 0 NOT NULL
name varchar(50) NOT NULL
phone varchar(20) UNIQUE NOT NULL
email varchar(100) UNIQUE NOT NULL
address varchar(200) NOT NULL
city varchar(50) NOT NULL
postcode varchar(10) NOT NULL
register_date date NOT NULL
total_points int DEFAULT 0 NOT NULL

Brief of E-Commerce ER-Diagram

1. Document Overview

This brief outlines the ER-Diagram's core components: entity definitions, relationship cardinalities, and constraint rules, focusing on key details to support e-commerce workflows.

2. Core Entity Definitions

2.1 Organizational Management

2.1.1 Branch (Store)

- PK: branch_id (auto-increment INT, unique store ID).
- Key Fields:
    - manager_name (VARCHAR(50), NOT NULL): Store manager's name (no FK link).
    - branch_name/phone (VARCHAR, UNIQUE + NOT NULL): Unique store name/contact.
    - address/city/opening_date (NOT NULL): Location, region, and launch date.

2.1.2 Employee (Staff)

- PK: emp_id (auto-increment INT, unique staff ID).
- Key Fields:
    - first_name/last_name/email/phone (NOT NULL; email/phone UNIQUE): Contact/identity details.
    - position (VARCHAR(50), NOT NULL): Job role (e.g., "Sales Associate").
    - branch_id (FK → Branch.branch_id, NOT NULL): Links to assigned store.
    - hire_date/salary (NOT NULL): Start date and monthly compensation.

2.1.3 Customer (User)

- PK: customer_id (auto-increment INT, unique user ID).
- Key Fields:
    - membership_id (FK → Membership.membership_id, DEFAULT 0 + NOT NULL): Ties to loyalty tier.
    - first_name/last_name/email/phone (NOT NULL; email/phone UNIQUE): Contact details.
    - address/city/postcode (NOT NULL): Shipping information.
    - register_date/total_points (NOT NULL; points DEFAULT 0): Sign-up date and loyalty points.

2.2 Product & Supply Chain

2.2.1 Supplier (Vendor)

- PK: supplier_id (auto-increment INT, unique vendor ID).

- Key Fields:
    - name (VARCHAR(100), UNIQUE + NOT NULL): Unique vendor name.
    - contact_person/phone/email/address (NOT NULL): Vendor contact/location.

2.2.2 Product (Merchandise)

- PK: product_id (auto-increment INT, unique product ID).
- Key Fields:
    - name/brand/model/category (NOT NULL): Product identification.
    - cost_price/retail_price (DECIMAL, NOT NULL): Internal cost and selling price.
    - supplier_id (FK → Supplier.supplier_id, NOT NULL): Links to primary vendor.
    - status (ENUM: Active/Discontinued, NOT NULL): Availability control.

2.2.3 Inventory (Stock)

- PK: inventory_id (auto-increment INT, unique stock ID).
- Key Fields:
    - branch_id (FK → Branch.branch_id) + product_id (FK → Product.product_id): Ties stock to store/product.
    - Composite Unique: branch_id + product_id (one stock record per product-store).
    - quantity (INT, DEFAULT 0 + NOT NULL): Current stock level.
    - last_update/unit_price (NOT NULL): Stock update time and unit cost.

2.3 Order & Transaction

2.3.1 Sales_Order (Customer Order)

- PK: order_id (auto-increment INT, unique order ID).
- Key Fields:
    - customer_id (FK → Customer.customer_id, NULLABLE): Links to customer (NULL for guests).
    - emp_id/branch_id (FK, NOT NULL): Links to staff and store.
    - order_date/total_amount (NOT NULL): Order time and total value.
    - discount_amount (DEFAULT 0) + status (ENUM: Pending/Paid/Cancelled/Completed) + payment_method (ENUM: Cash/Card/PayPal): Discount, lifecycle, and payment type.

2.3.2 Sales_Order_Item (Order Line)

- PK: so_item_id (auto-increment INT, unique line ID).
- Key Fields:

- order_id (FK → Sales_Order.order_id) + product_id (FK → Product.product_id): Ties line to order/product.
- quantity/unit_price/total_price (NOT NULL): Line item quantity, price, and total.

2.3.3 Purchase_Order (Vendor Order)

- PK: po_id (auto-increment INT, unique PO ID).
- Key Fields:
  - supplier_id/emp_id/branch_id (FK, NOT NULL): Links to vendor, staff, and receiving store.
  - po_date/total_cost (NOT NULL): PO time and total cost.
  - status (ENUM: Pending/Shipped/Received/Cancelled) + payment_method (ENUM: Cheque/Transfer/PayPal): Lifecycle and payment type.

2.3.4 Purchase_Order_Item (PO Line)

- PK: po_item_id (auto-increment INT, unique line ID).
- Key Fields:
  - po_id (FK → Purchase_Order.po_id) + product_id (FK → Product.product_id): Ties line to PO/product.
  - quantity/unit_cost/total_cost (NOT NULL): Line item quantity, cost, and total.

2.3.5 Points_Transaction (Loyalty)

- PK: trans_id (auto-increment INT, unique transaction ID).
- Key Fields:
  - customer_id (FK → Customer.customer_id, NOT NULL): Links to customer.
  - order_id (FK → Sales_Order.order_id, NULLABLE): Ties to order (NULL for manual adjustments).
  - point_change/trans_date_time/balance_after (NOT NULL): Points change, time, and post-transaction balance.

2.4 Membership (Loyalty Tier)

- PK: membership_id (auto-increment INT, unique tier ID).
- Key Fields:
  - membership_name (VARCHAR(50), UNIQUE + NOT NULL): Tier name (e.g., "Gold Member").
  - discount_rate/point_rate/min_consume (NOT NULL): Discount, points multiplier, and upgrade threshold.

3. Entity Relationships

3.1 One-to-Many (1:M)

- Branch → Employee/Inventory/Sales_Order/Purchase_Order; Supplier → Product; Customer → Sales_Order/Points_Transaction; Employee → Sales_Order/Purchase_Order; Sales_Order → Sales_Order_Item; Purchase_Order → Purchase_Order_Item; Product → Sales_Order_Item/Purchase_Order_Item/Inventory; Membership → Customer.

3.2 Resolved Many-to-Many (M:N)

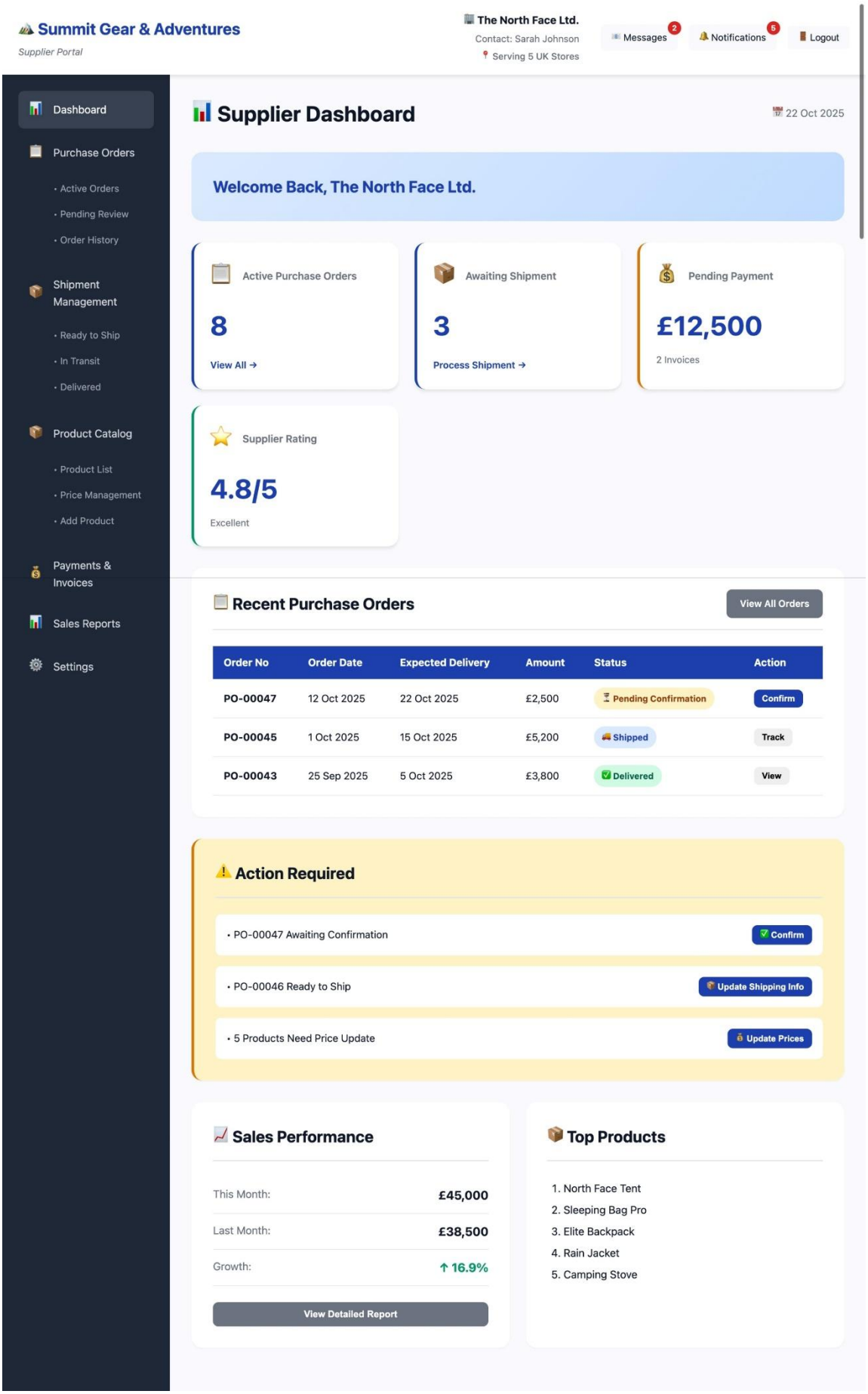
- Sales_Order ↔ Product (via Sales_Order_Item); Purchase_Order ↔ Product (via Purchase_Order_Item).

4. Constraint Rules

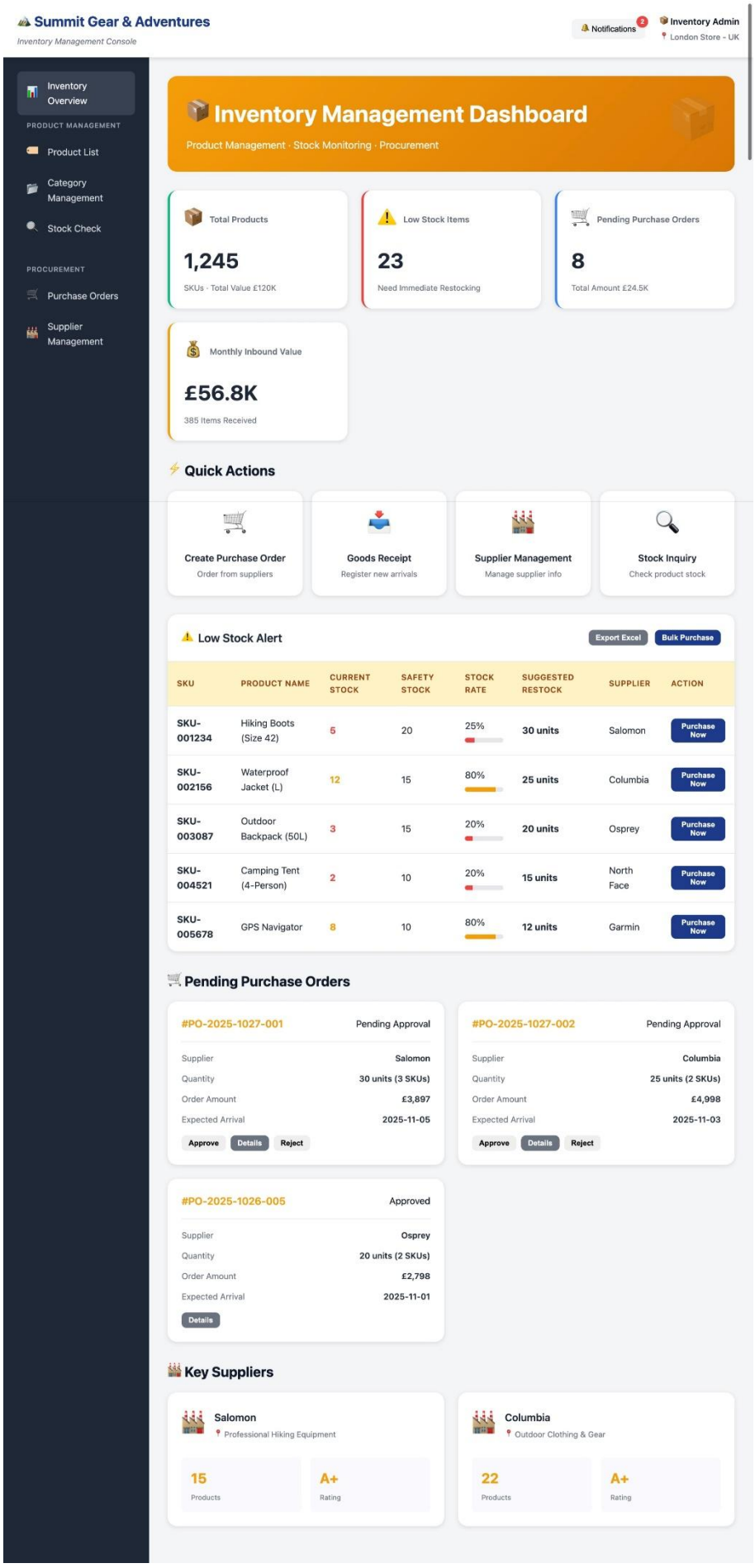
- PK: All entities have unique auto-increment INT PKs.
- FK: Enforce relationships (most non-null; exceptions: guest checkout, manual points).
- Unique: Prevent duplicates (e.g., store name, customer email; composite unique for inventory).
- Default/Non-Null: Core fields (e.g., product name, order date) are non-null; default values for points/inventory/membership.

User interface designs

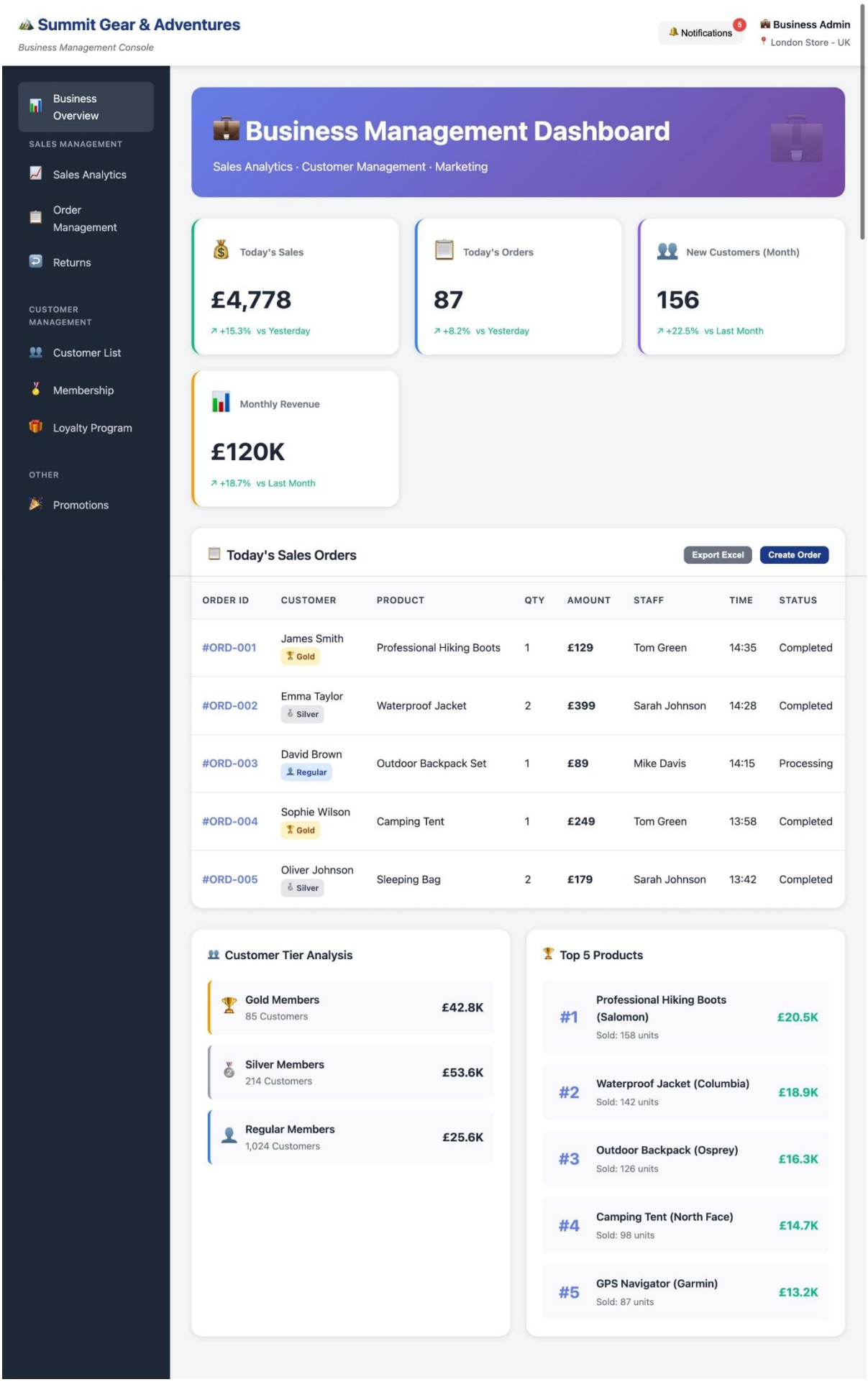
Supplier Dashboard



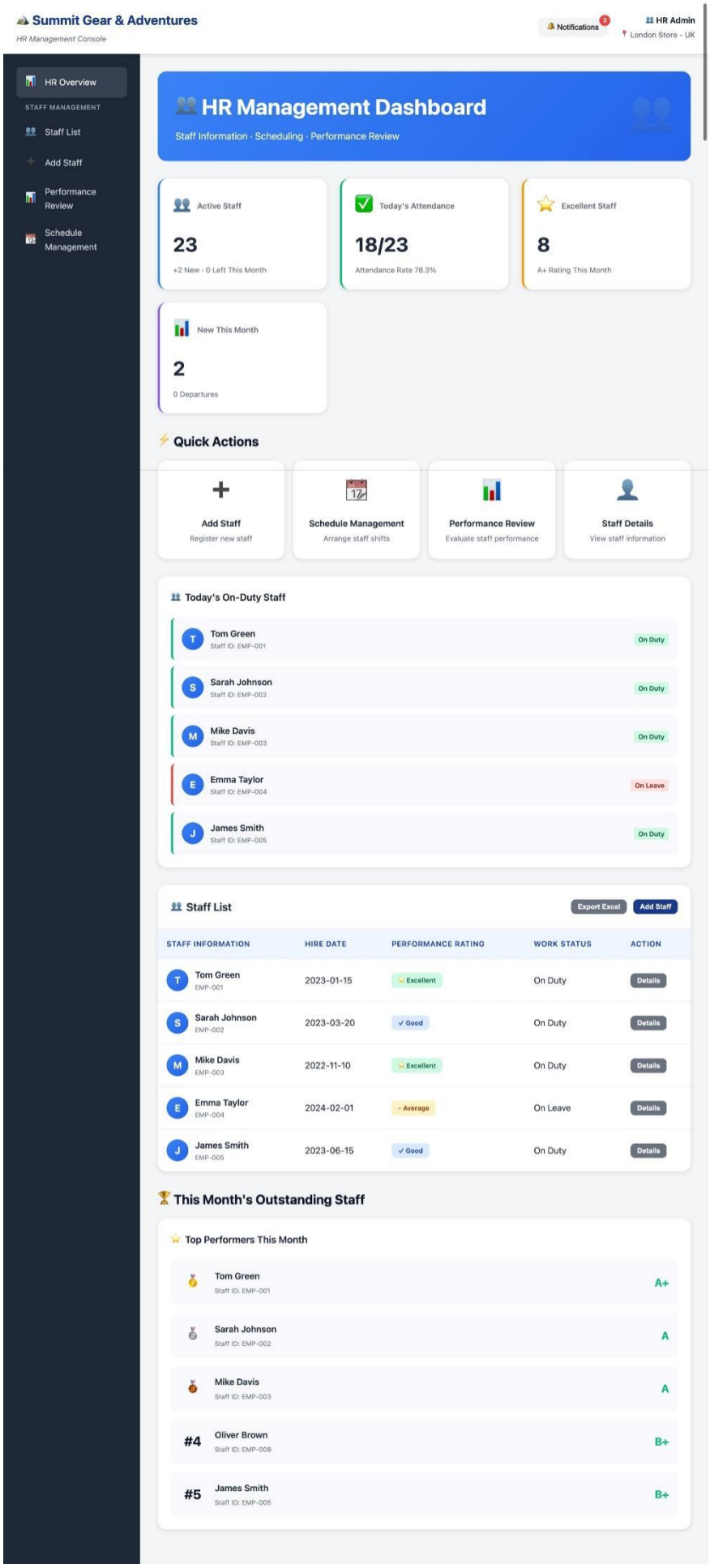
Manager Dashboard—Inventory Manager



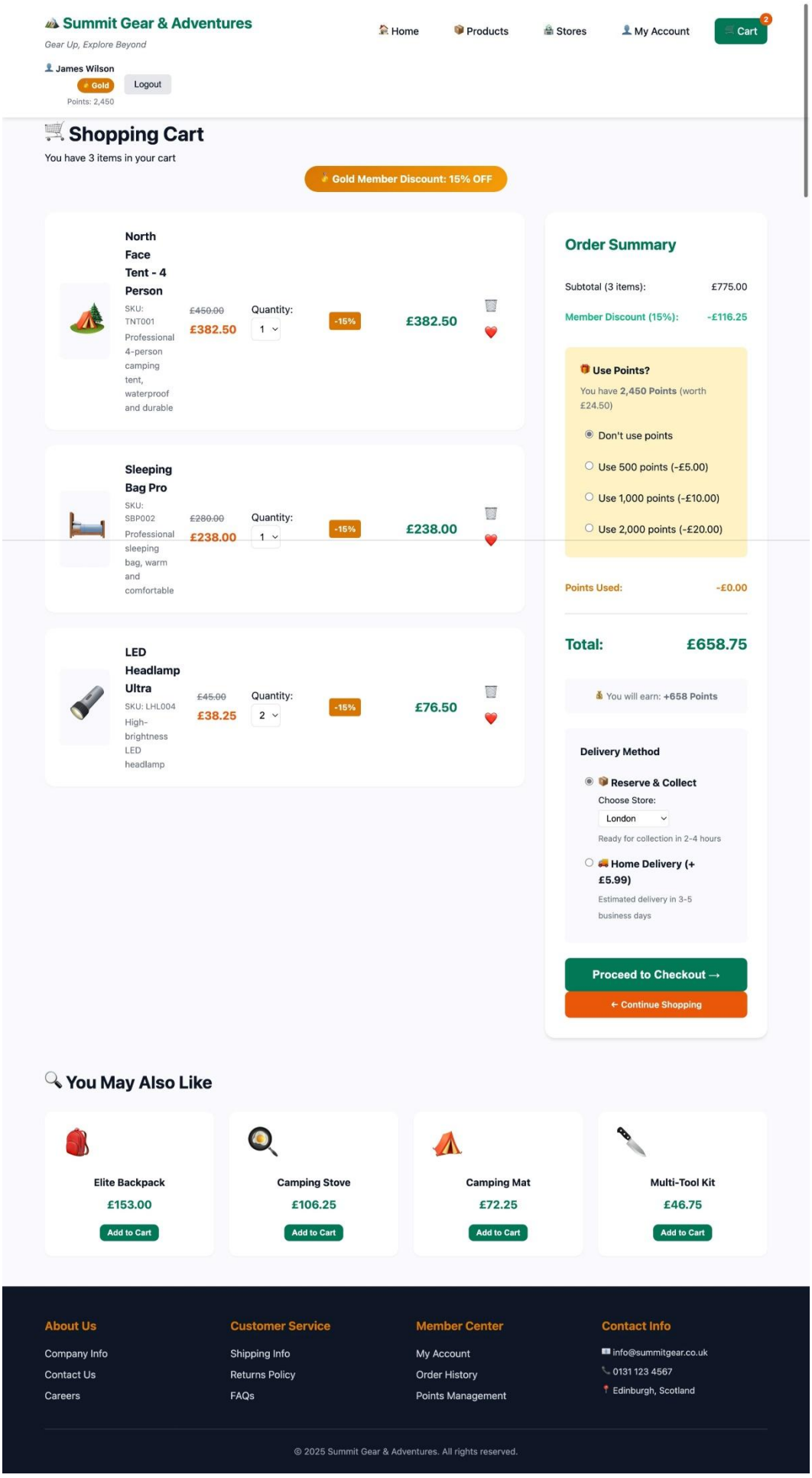
Manager Dashboard—Business Manager



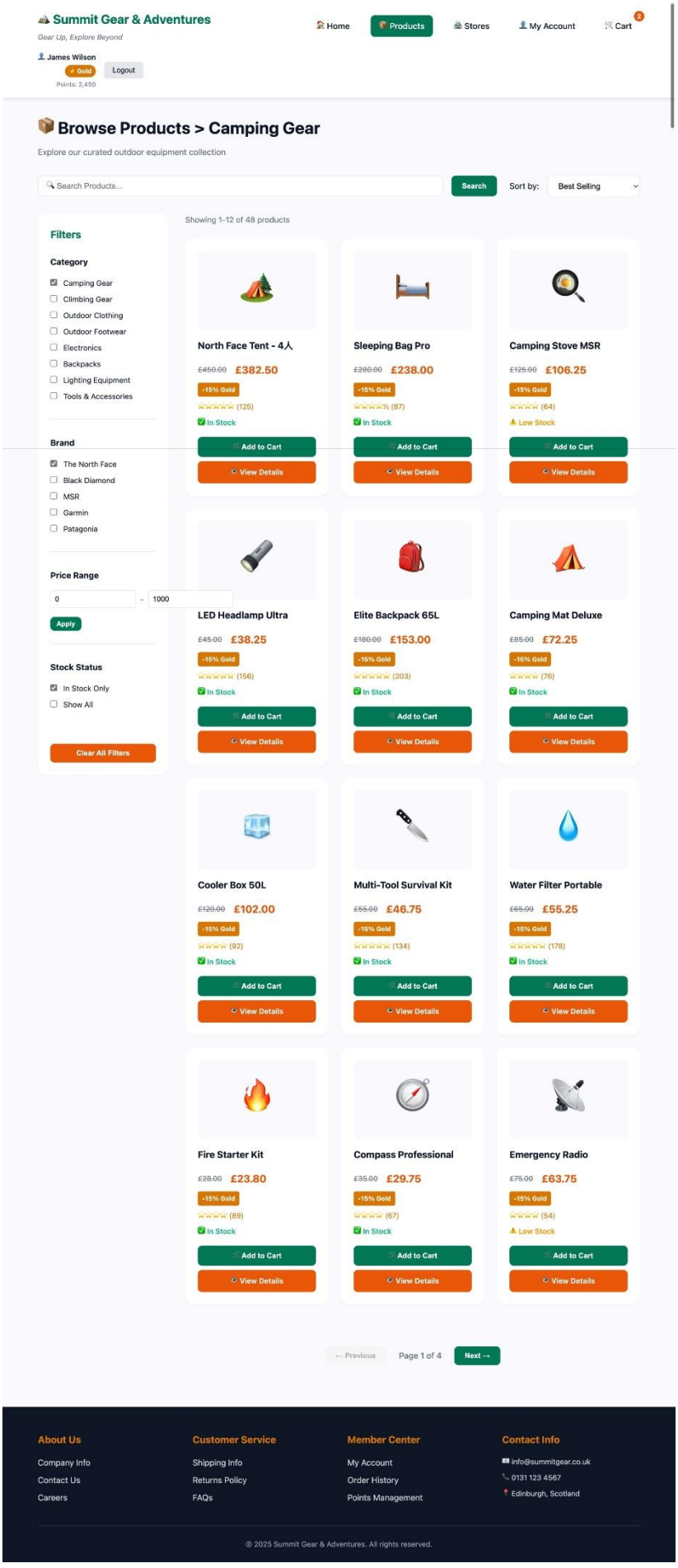
Manager Dashboard—HR Manager



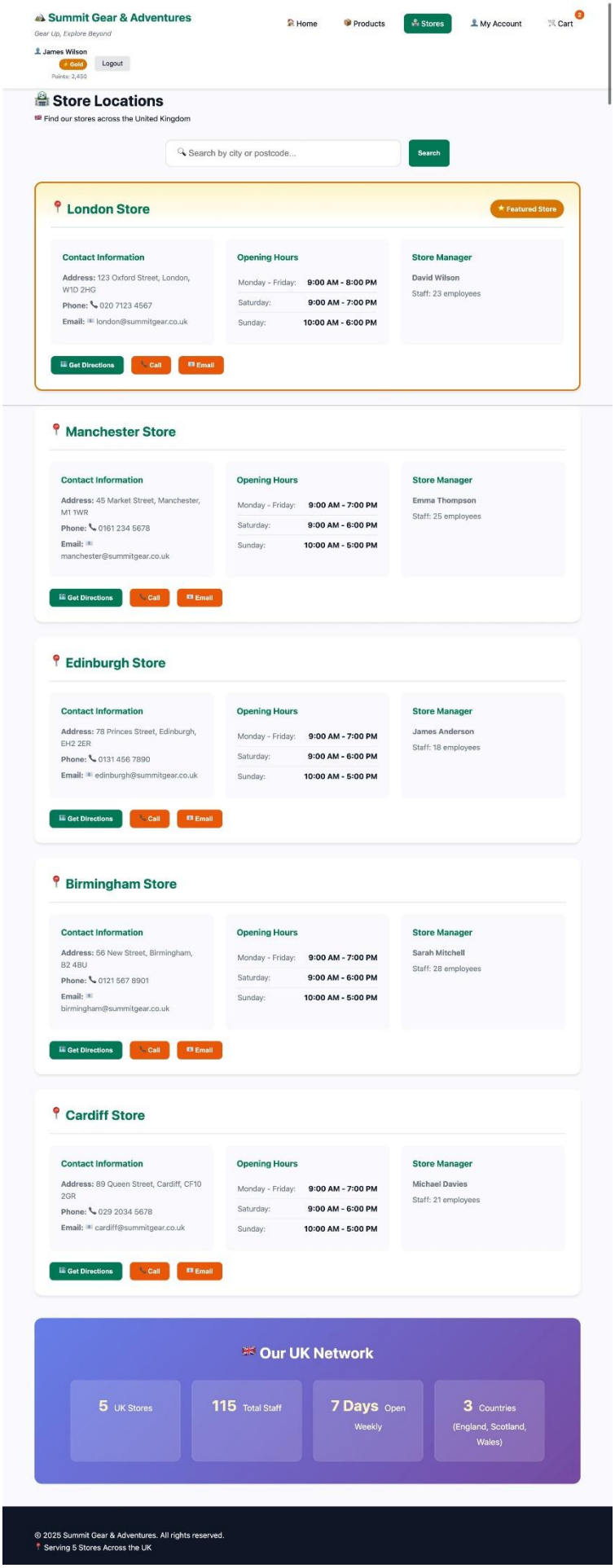
Customer Dashboard—Shopping Cart



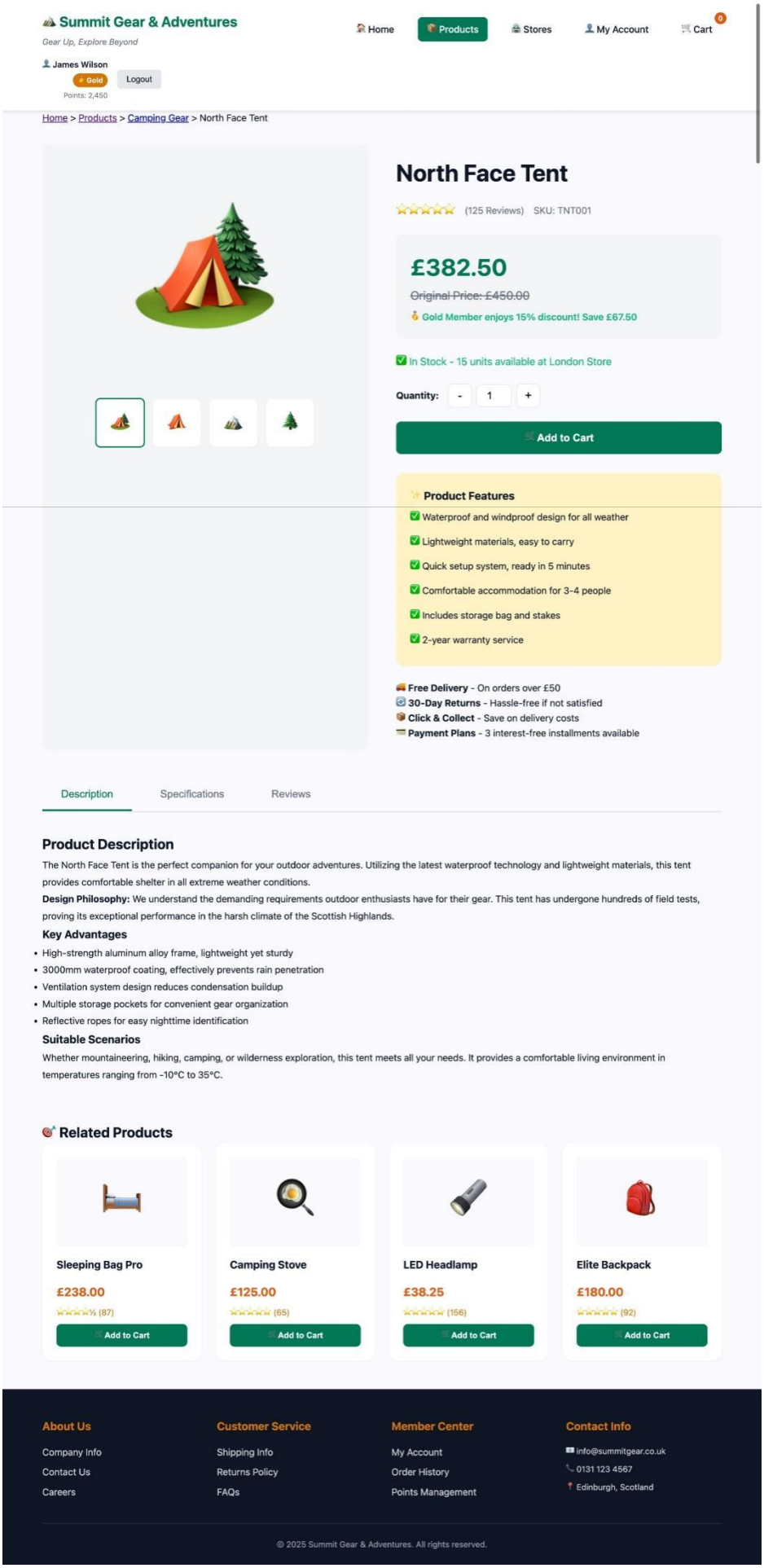
Customer Dashboard—Shopping Mall



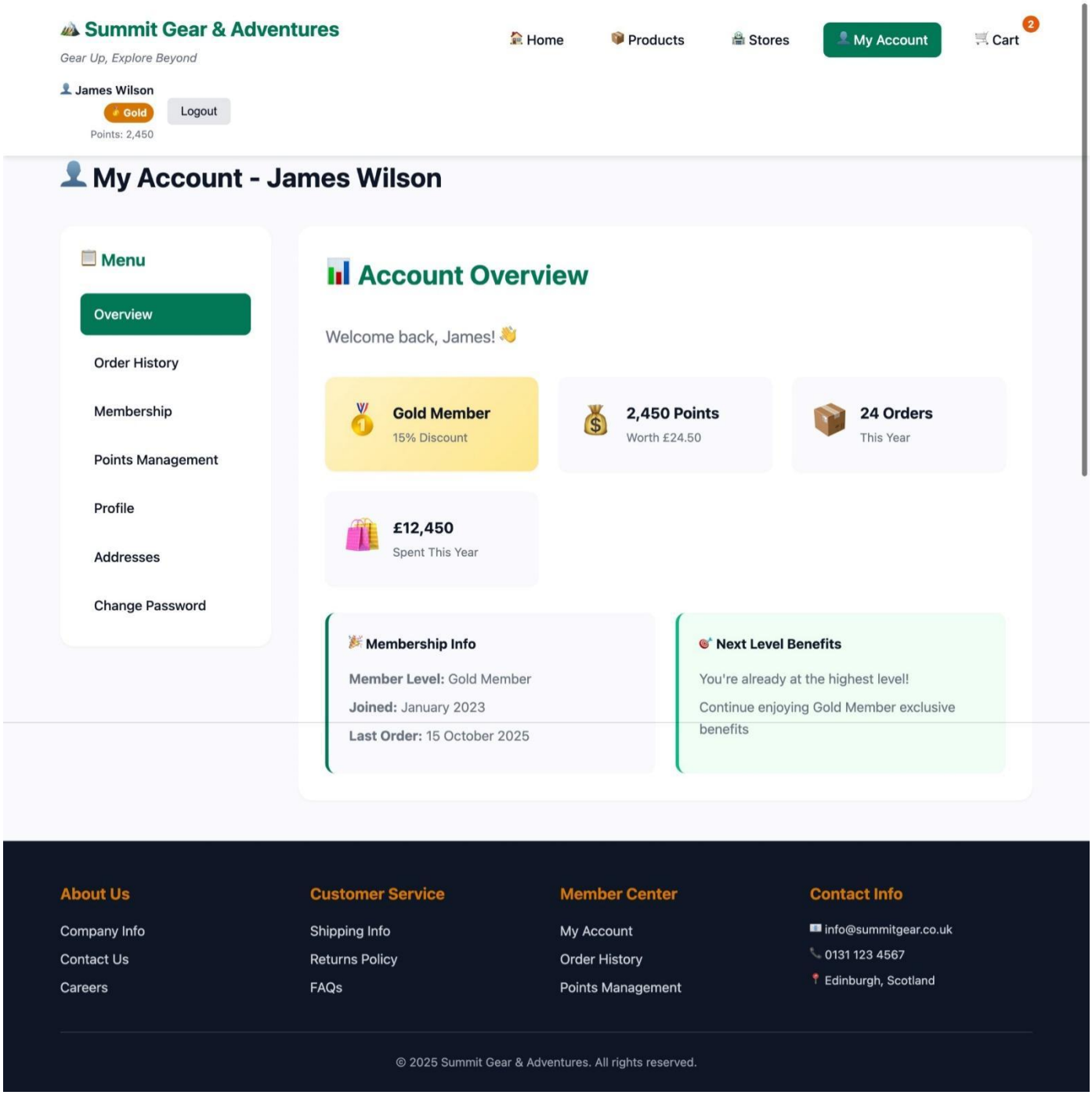
Customer Dashboard—Store Information



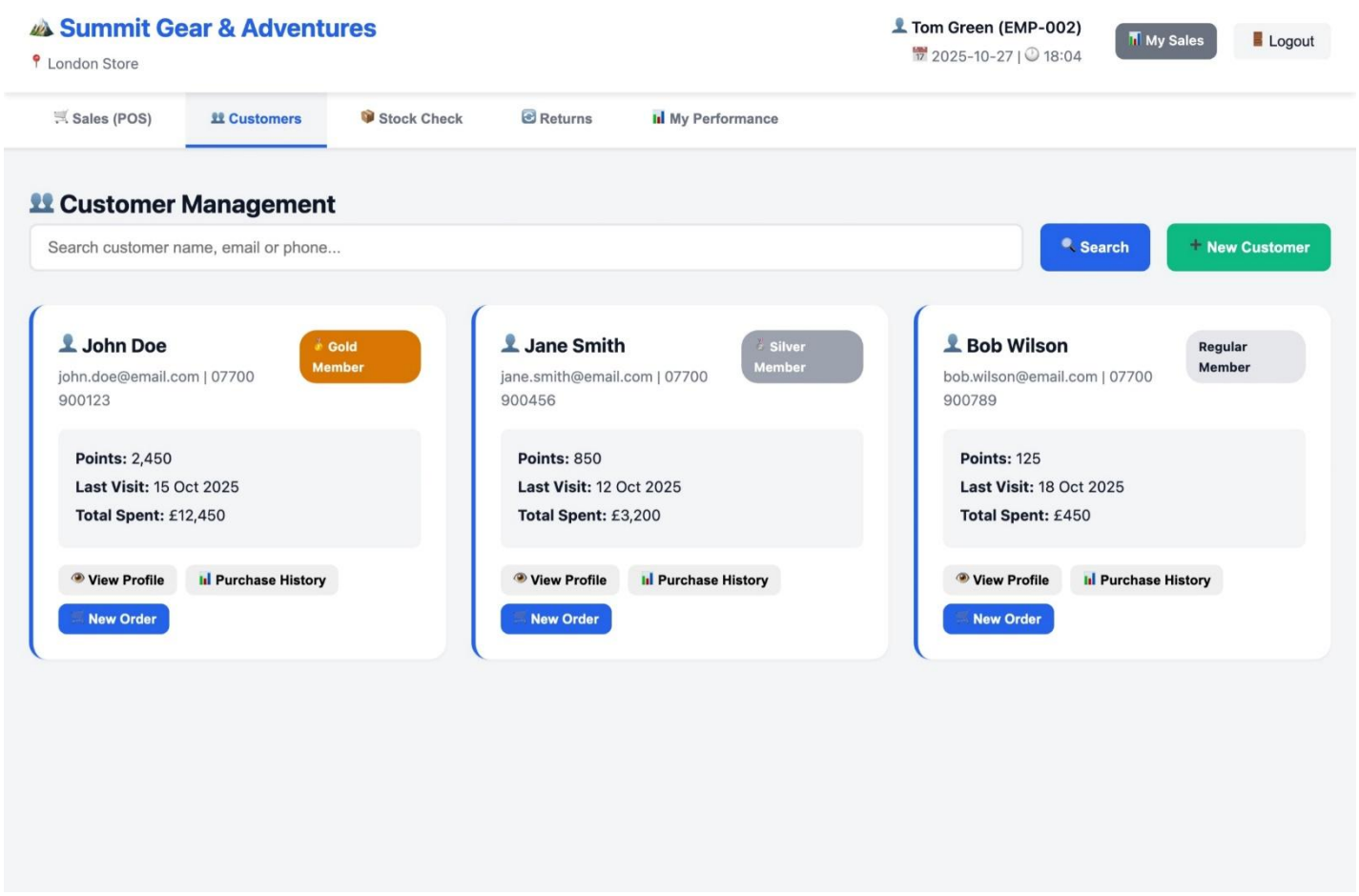
Customer Dashboard—Product Information



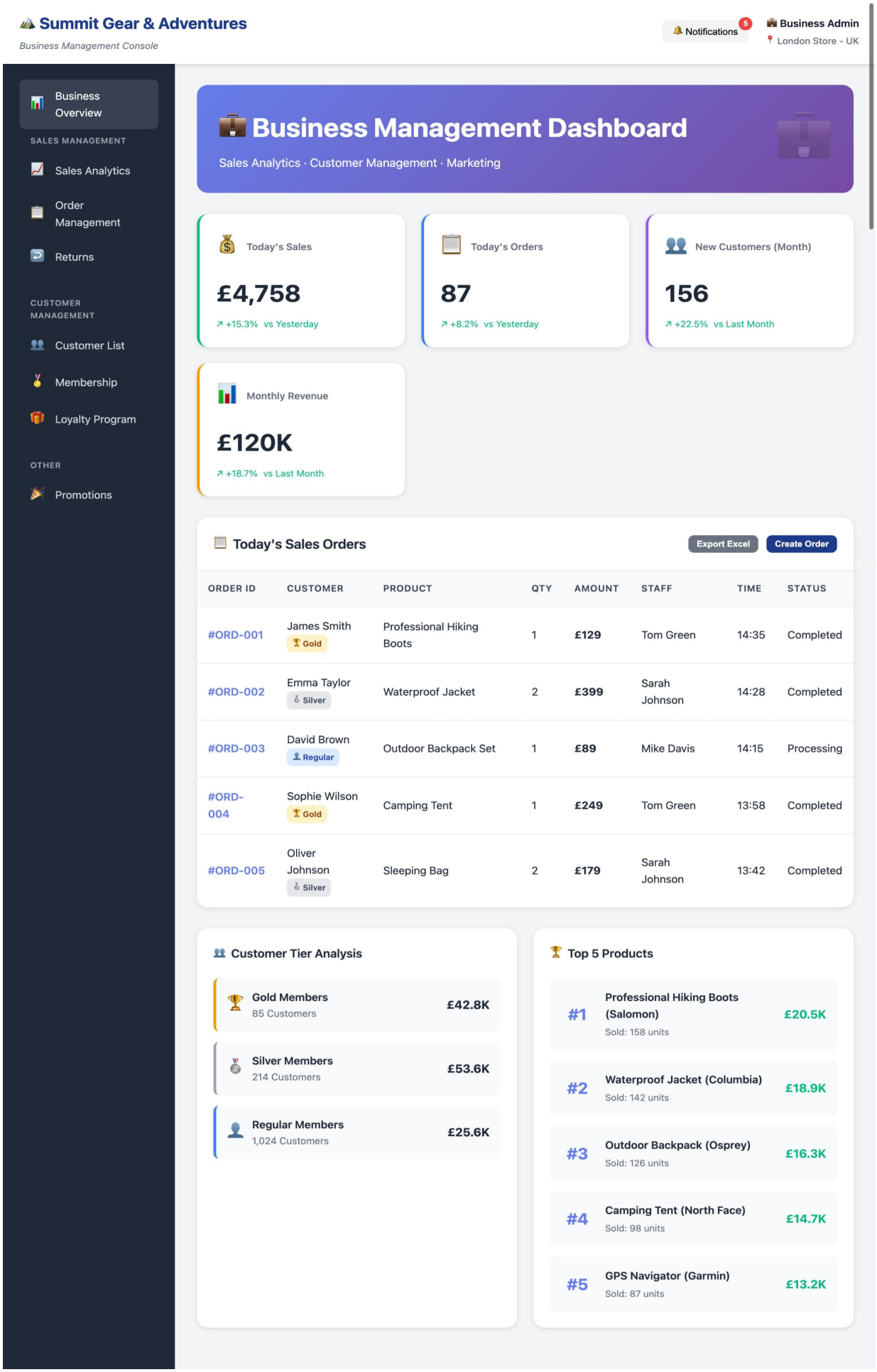
Customer Dashboard—My Account



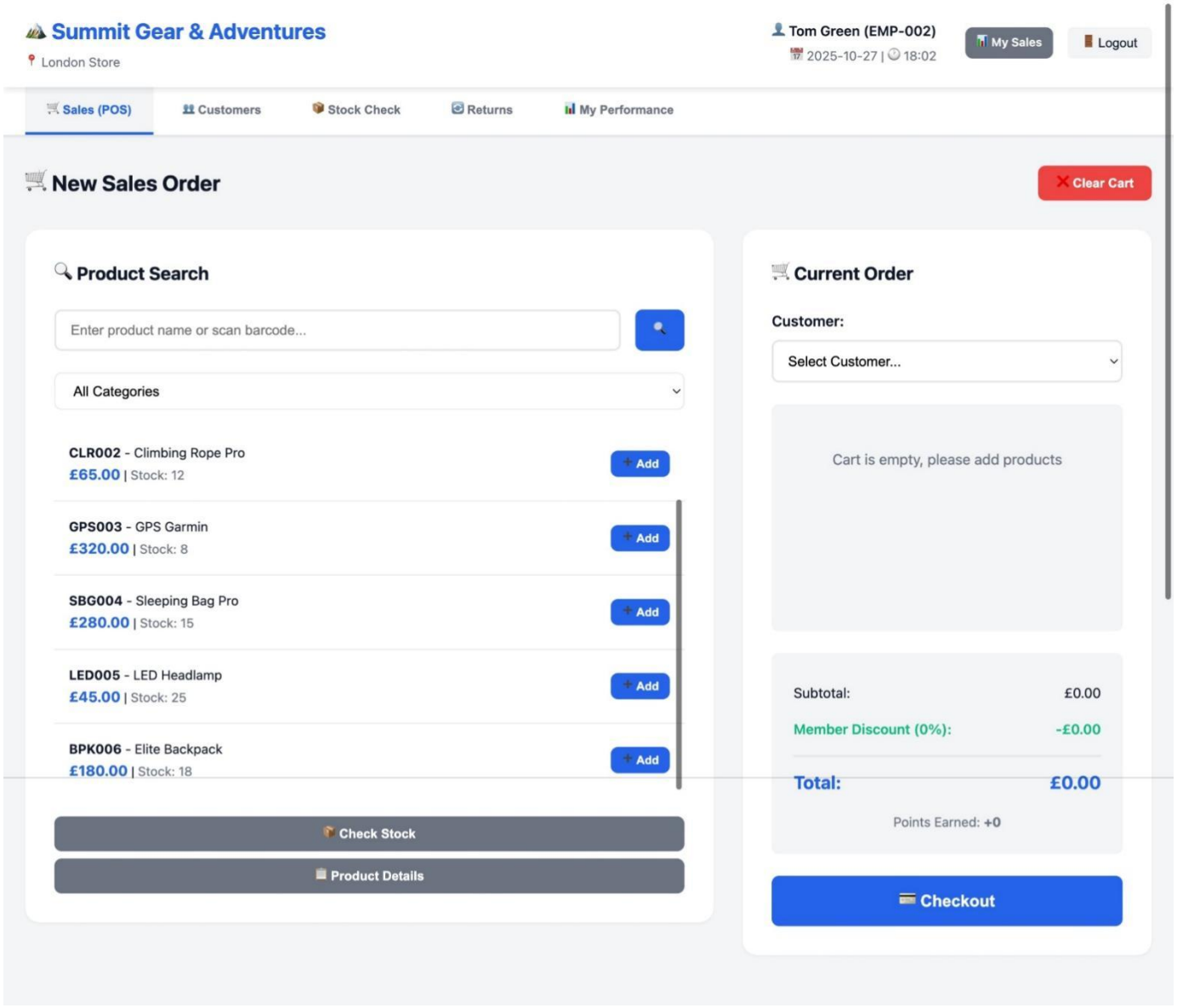
Customer Dashboard—Customer Information



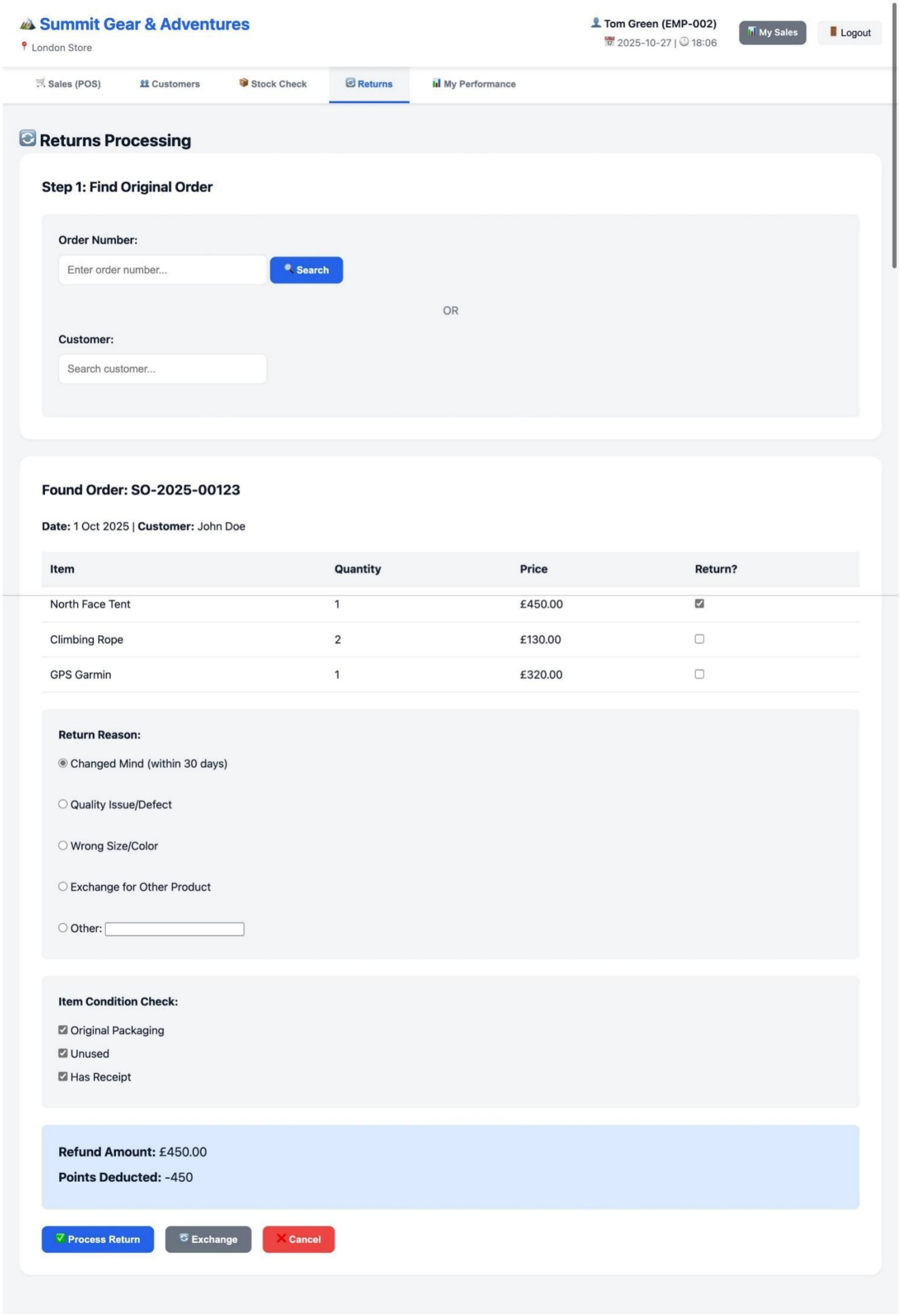
Store Staff Dashboard



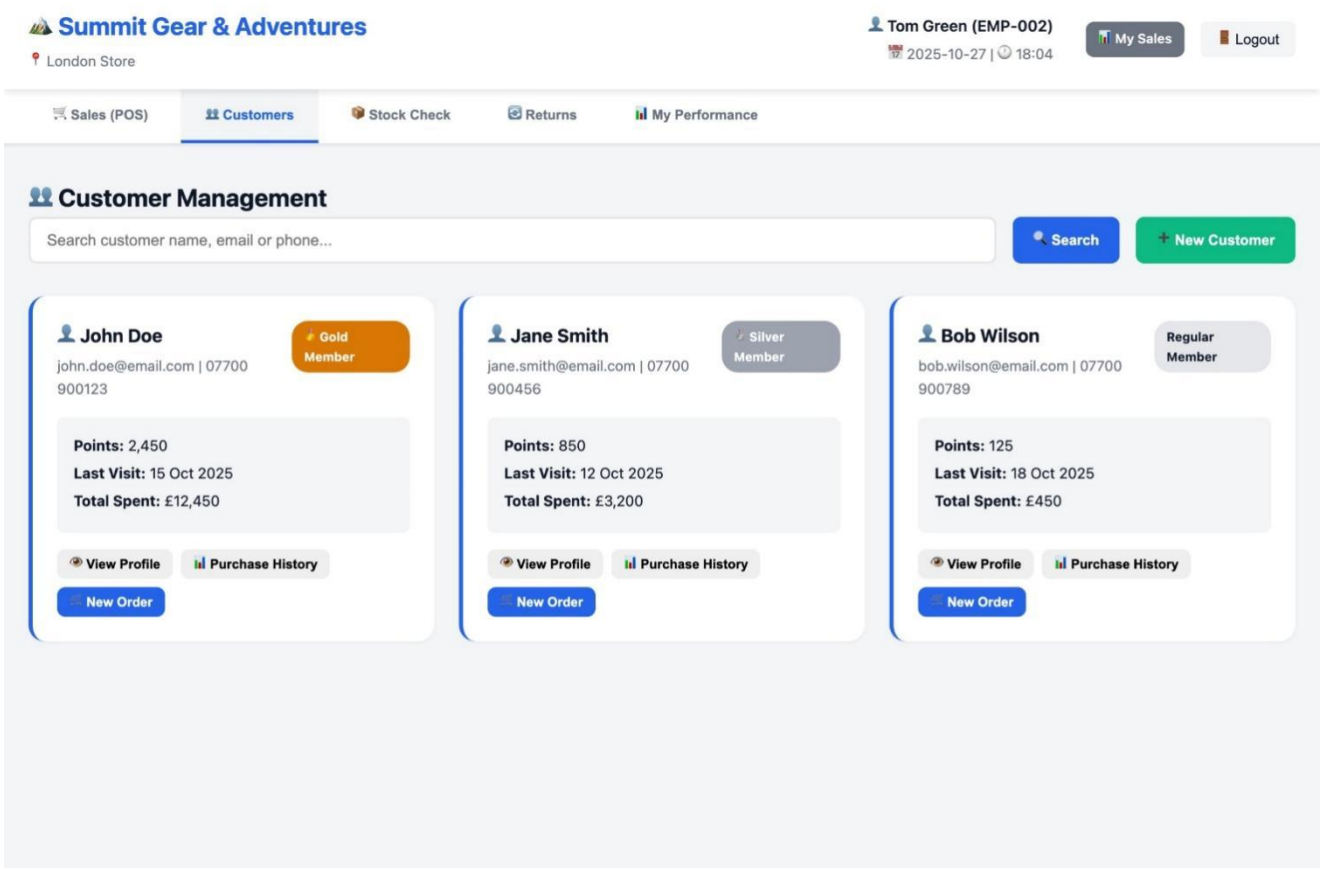
Staff Dashboard—Order Processing



Staff Dashboard—Returns Processing



Staff Dashboard—Customer Information



SQL CREATE statements

```
DROP DATABASE IF EXISTS summit_gear_db;
CREATE DATABASE summit_gear_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE summit_gear_db;

-- 1. Branch
CREATE TABLE Branch (
    branch_id INT AUTO_INCREMENT PRIMARY KEY,
    branch_name VARCHAR(100) NOT NULL,
    city VARCHAR(50) NOT NULL,
    address VARCHAR(200) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    manager_name VARCHAR(100),
    opening_date DATE NOT NULL,
    INDEX idx_city (city)
) ENGINE=InnoDB;

-- 2. Employee
CREATE TABLE Employee (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone VARCHAR(20) NOT NULL,
    position VARCHAR(50) NOT NULL,
    salary DECIMAL(10,2) NOT NULL,
    hire_date DATE NOT NULL,
    branch_id INT NOT NULL,
    FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB;
```

```
-- 3. Membership
CREATE TABLE Membership (
    membership_id INT AUTO_INCREMENT PRIMARY KEY,
    membership_name VARCHAR(20) NOT NULL UNIQUE,
    discount_rate DECIMAL(5,2) NOT NULL,
    point_rate DECIMAL(5,2) NOT NULL,
    min_consume DECIMAL(10,2) NOT NULL
) ENGINE=InnoDB;

-- 4. Customer
CREATE TABLE Customer (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone VARCHAR(20) NOT NULL UNIQUE,
    address VARCHAR(200),
    city VARCHAR(50),
    postcode VARCHAR(10),
    membership_id INT DEFAULT NULL,
    total_points INT NOT NULL DEFAULT 0,
    registration_date DATE NOT NULL,
    FOREIGN KEY (membership_id) REFERENCES Membership(membership_id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB;

-- 5. Supplier
CREATE TABLE Supplier (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    contact_person VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
```

```
    phone VARCHAR(20) NOT NULL UNIQUE,
    address VARCHAR(200),
    city VARCHAR(50),
    country VARCHAR(50) NOT NULL DEFAULT 'UK'
) ENGINE=InnoDB;

-- 6. Product
CREATE TABLE Product (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    sku VARCHAR(50) NOT NULL UNIQUE,
    name VARCHAR(50) NOT NULL,
    brand VARCHAR(50) NOT NULL,
    category VARCHAR(30) NOT NULL,
    cost_price DECIMAL(10,2) NOT NULL,
    retail_price DECIMAL(10,2) NOT NULL,
    weight_kg DECIMAL(6,2),
    dimensions VARCHAR(50),
    color VARCHAR(30),
    size VARCHAR(20),
    supplier_id INT NOT NULL,
    status ENUM('Active','Discontinued') NOT NULL DEFAULT 'Active',
    FOREIGN KEY (supplier_id) REFERENCES Supplier(supplier_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    INDEX idx_brand (brand)
) ENGINE=InnoDB;

-- 7. Inventory
CREATE TABLE Inventory (
    inventory_id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT NOT NULL,
    branch_id INT NOT NULL,
```

```
    quantity INT NOT NULL DEFAULT 0,
    last_updated DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
        ON UPDATE CURRENT_TIMESTAMP,
    location VARCHAR(50),
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    UNIQUE KEY uk_product_branch (product_id, branch_id)
) ENGINE=InnoDB;

-- 8. Sales_Order
CREATE TABLE Sales_Order (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    order_number VARCHAR(20) NOT NULL UNIQUE,
    customer_id INT NOT NULL,
    employee_id INT NOT NULL,
    branch_id INT NOT NULL,
    order_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    total_amount DECIMAL(10,2) NOT NULL,
    discount_amount DECIMAL(10,2) NOT NULL DEFAULT 0,
    final_amount DECIMAL(10,2) NOT NULL,
    payment_method VARCHAR(20) NOT NULL,
    status ENUM('Completed','Returned','Cancelled') NOT NULL DEFAULT 'Completed',
    points_earned INT NOT NULL DEFAULT 0,
    payment_status ENUM('Unpaid','Partial','Paid') NOT NULL DEFAULT 'Unpaid',
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (employee_id) REFERENCES Employee(employee_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
```

```
    ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB;

-- 9. Sales_Order_Item
CREATE TABLE Sales_Order_Item (
    so_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    discount DECIMAL(10,2) NOT NULL DEFAULT 0,
    total_price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES Sales_Order(order_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB;

-- 10. Purchase_Order
CREATE TABLE Purchase_Order (
    po_id INT AUTO_INCREMENT PRIMARY KEY,
    po_number VARCHAR(20) NOT NULL UNIQUE,
    supplier_id INT NOT NULL,
    employee_id INT NOT NULL,
    branch_id INT NOT NULL,
    order_date DATE NOT NULL,
    total_amount DECIMAL(12,2) NOT NULL,
    status ENUM('Pending','Confirmed','Shipped','Received','Cancelled') NOT NULL DEFAULT 'Pending',
    payment_status ENUM('Unpaid','Partial','Paid') NOT NULL DEFAULT 'Unpaid',
    FOREIGN KEY (supplier_id) REFERENCES Supplier(supplier_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
```

```
    FOREIGN KEY (employee_id) REFERENCES Employee(employee_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB;

-- 11. Purchase_Order_Item
CREATE TABLE Purchase_Order_Item (
    po_item_id INT AUTO_INCREMENT PRIMARY KEY,
    po_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    total_price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (po_id) REFERENCES Purchase_Order(po_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
        ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB;

-- 12. Points_Transaction
CREATE TABLE Points_Transaction (
    trans_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    order_id INT,
    point_change INT NOT NULL,
    trans_type ENUM('Earn','Redeem','Expire','Adjust') NOT NULL,
    trans_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    balance_after INT NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
```

```
    FOREIGN KEY (order_id) REFERENCES Sales_Order(order_id)
        ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB;
```