MASTER OF COMPUTER APPLICATION

Inventory Management

ON

Project Report

Subject:PL/SQL LAB (24CAP-602)

Submitted To: Paras Sen.

Submitted By: Rajan Kumar

UID: 24MCA20296

Section/Group:24MCA-5(A)

University Institute of Computing
Chandigarh University,Gharuan,Mohali

Overview of Inventory Management System Project

Introduction

The Inventory Management System (IMS) is designed to effectively control and oversee stock operations for businesses of various sizes. This project incorporates three distinct user levels:

- **Billing Counter Level**
- **Manager Level**
- **Owner Level**

Key Features

The IMS includes several key features to enhance inventory management and improve operational efficiency:

1. **Sales Forecasting**
   Analyzes historical sales data to predict future demand, allowing for proactive inventory planning.
2. **Reorder Notifications**
   Alerts users when stock levels fall below predetermined thresholds, ensuring timely replenishment of inventory.
3. **Fraud Mitigation**
   Implements safeguards to minimize the risk of fraudulent activities by staff in managing stock.
4. **Customer Information Management**
   Facilitates access and maintenance of customer information to enhance service quality and tracking.
5. **Automated Invoice Generation**
   Streamlines billing processes by automatically generating invoices for sales transactions.
6. **Transaction Tracking**
   Monitors payments received through various methods, providing clear financial oversight.

Objectives

The IMS aims to establish a robust framework for inventory management that enhances operational efficiency, accuracy, and reliability. Key objectives include:

- **Streamlined Operations**
  Organize and simplify inventory management workflows.
- **Data Accuracy**
  Maintain high levels of accuracy in inventory data and transactions.
- **Reliability**
  Develop a dependable system for inventory tracking and reporting.
- **Reduction of Redundancy**
  Eliminate duplicate data entries and improve data integrity.

- **Efficient Data Access**
  Facilitate quick retrieval of information for users at all levels.
- **User-Friendly Interface**
  Design an intuitive and cost-effective application for all users.

Background

Historically, inventory management relied heavily on manual processes, which often led to inaccuracies and inefficiencies. The IMS addresses these challenges by providing an automated solution that enhances the management of inventory levels, invoicing, and customer records.

Introduction

An Inventory Management System (IMS) is a sophisticated integration of technology—encompassing both hardware and software—along with well-defined methods and strategies that facilitate the efficient monitoring and management of stocked goods. These goods may include organizational assets, raw materials, components, and finished products destined for suppliers or end customers. The versatility of an IMS makes it suitable for a wide range of environments, from small local stores and department shops to large multinational corporations (MNCs), all of which rely on precise inventory tracking to optimize their operations.

Core Functionality

One of the core functionalities of the IMS is its ability to maintain comprehensive records of various stakeholders, including manager and customer information, thereby fostering a more organized business environment. By integrating these data points, the system enhances communication and transparency within the organization, enabling informed decision-making at all levels.

Inventory Threshold Control

A key feature of the IMS is its inventory threshold control, which allows users to establish minimum inventory levels for each item. When stock levels fall below these predetermined thresholds, the system automatically triggers reorder alerts, ensuring that organizations can proactively manage their inventory. This feature not only minimizes the risk of stockouts but also enhances sales performance by ensuring that products are consistently available for customers.

Analytical Insights

Moreover, the IMS provides valuable analytical insights by analyzing historical sales data, enabling businesses to accurately forecast demand. This predictive capability helps organizations make strategic inventory decisions, reduce excess stock, and optimize storage space. Additionally, the system can facilitate various payment methods and generate detailed invoices, enhancing the overall customer experience.

Importance of an IMS

In a competitive market, having an effective inventory management system is essential for maintaining operational efficiency and maximizing profitability. By implementing an IMS, businesses can streamline their inventory processes, reduce costs, and focus on delivering exceptional value to their customers.

Scope

The Inventory Management System (IMS) is designed to streamline various aspects of inventory control and provide enhanced functionality for users at different levels. Key features and benefits of the system include:

Accurate Inventory Tracking

The IMS ensures precise counting of products in inventory, minimizing discrepancies and enabling real-time updates on inventory levels. This accuracy is vital for effective inventory management and helps maintain optimal inventory levels.

Automatic Reordering

Users can set minimum order quantities for each product, triggering automatic reorder alerts when inventory levels fall below these thresholds. This feature helps prevent stockouts, ensuring that products are consistently available to meet customer demand.

Reduction of Duplicate Entries

By implementing a centralized database, the IMS minimizes the risk of duplicate entries and enhances data integrity. This ensures that all transactions are accurately recorded, improving reporting and analysis.

Enhanced Reporting and Analytics

The system can generate various reports that provide insights into sales trends, inventory turnover, and overall business performance. This data is invaluable for strategic decision-making and can inform future purchasing and sales strategies.

Working

The IMS operates with distinct user interfaces tailored to different roles within the organization, ensuring that each user has access to the functionalities they need while maintaining security and data integrity:

Billing Counter User

The billing counter personnel will have a streamlined interface that allows them to efficiently manage customer transactions. They can adjust product quantities in real-time, generate invoices for sold products, and create return notes for items returned by customers. This ensures a smooth and accurate checkout process, enhancing customer satisfaction.

Manager Level Access

Managers will have enhanced capabilities within the system. They can adjust pricing for dynamic inventory items, allowing for flexibility in response to market conditions. Managers also have access to detailed sales reports and inventory data to monitor performance and make informed decisions regarding inventory replenishment and promotions.

Owner Level Access

The business owner will have comprehensive access to the system, enabling them to generate summary reports of daily sales, total sales per counter, and performance by individual salespeople. This high-level overview provides the owner with critical insights into business operations, allowing for strategic planning and resource allocation.

User Role Management

The system will incorporate role-based access controls, ensuring that users can only access the functionalities pertinent to their roles. This helps safeguard sensitive information and maintain the integrity of the inventory system.

Integration with Other Systems

To enhance operational efficiency, the IMS can be integrated with other business systems, including accounting software and customer relationship management (CRM) tools. This integration allows for seamless data sharing and improved overall business intelligence.

By focusing on these components, the IMS not only simplifies inventory management but also empowers users at all levels to contribute effectively to the organization's success.

Purpose

The purpose of an effective Inventory Management System (IMS) is to seamlessly integrate the following objectives to ensure continuity across functions:

Company's Strategic Goals

The IMS aims to align inventory management with overarching business strategies, driving growth and enhancing operational performance. By synchronizing inventory practices with the company's objectives, the system ensures that inventory decisions support overall business goals.

Sales Forecasting

Utilizing historical sales data, the IMS forecasts future demand, helping maintain optimal inventory levels and preventing both overstock and stockouts. This predictive capability enables businesses to make informed purchasing decisions and align their inventory with customer demand.

Sales Performance

The IMS is designed to enhance sales performance by ensuring that products are readily available for customers. By optimizing inventory levels and providing timely information on stock status, the system helps facilitate smoother transactions and improved customer satisfaction.

Goals of the Proposed System

The proposed Inventory Management System (IMS) aims to achieve the following key goals:

Planned Approach

The organization will benefit from a well-established workflow, with data systematically stored in designated repositories. This approach facilitates efficient data retrieval and secure storage, ensuring that information is easily accessible when needed.

Accuracy

The proposed system aims for a higher level of accuracy, ensuring that all operations are conducted efficiently and that the data generated from the central system is reliable and precise. This commitment to accuracy is essential for effective inventory management.

Reliability

Enhanced reliability is a key goal, resulting from improved data storage and management practices. This focus on reliability will foster confidence in the system's outputs, allowing users to trust the information provided.

Elimination of Redundancy

The system will be designed to prevent duplicate data storage, ensuring efficient use of storage space and maintaining data consistency across all levels of operation. This elimination of redundancy will contribute to better data integrity.

Immediate Information Retrieval

A primary objective is to enable quick and efficient access to information, streamlining decision-making processes within the organization. Timely access to data is crucial for responsive inventory management.

Immediate Data Storage

Unlike manual systems, the proposed solution will efficiently handle large volumes of data, ensuring timely and effective data storage. This capability will enhance the overall operational efficiency of the organization.

User-Friendly Operation

The system will be designed for ease of use, ensuring straightforward operation. It will be developed within an affordable timeframe and budget, making it accessible to all users, regardless of their technical proficiency.

Background

In today's competitive business landscape, effective inventory management is crucial for companies, retailers, and department stores aiming to optimize their stock levels and enhance profitability. Historically, inventory tracking relied on cumbersome paper-and-pen methods, which were not only labor-intensive but also prone to inaccuracies and errors. Such outdated practices hindered operational efficiency and decision-making processes.

To address these challenges, the development of an Inventory Management System (IMS) provides a modern solution. This software streamlines inventory maintenance and forecasting, enabling businesses to track stock in real time and respond swiftly to market demands. The system automates the billing process by generating invoices for each transaction, thereby minimizing human error. Additionally, it maintains comprehensive records, including employee and customer details, ensuring that all relevant information is centralized in one accessible platform. Ultimately, this all-in-one system empowers organizations to enhance their operational performance, reduce costs, and drive growth.

User Characteristics

To effectively utilize the Inventory Management System, users should possess the following characteristics:

Computer Proficiency

Users need to be comfortable navigating computers and using software applications, as the system requires fundamental operational skills.

Internet Familiarity

A good understanding of internet browsers and navigation is essential, allowing users to access online resources and support.

Basic English Proficiency

Users should have a foundational understanding of English to comprehend system prompts, reports, and documentation, facilitating effective communication.

Analytical Skills

Users should be able to analyze sales data and inventory trends to make informed decisions about inventory management.

Attention to Detail

Accuracy is critical in inventory management; therefore, users should be detail-oriented to ensure that all data entries are correct and up to date.

Adaptability

Users should be open to learning new technologies and methodologies, as the IMS may be updated with new features and functionalities over time.

Technical Feasibility: Back End

In this project, we have implemented only the back end of the Inventory Management System (IMS), which is designed using SQL Plus. Utilizing this Structured Query Language (SQL) platform, we created ten tables to support the system's functionalities:

1. **Brands**
   Stores information about the different brands available in the inventory.
2. **inv_user**
   Contains details of users who interact with the system, including their roles and permissions.
3. **Categories**
   Organizes products into distinct categories for easier management and retrieval.
4. **Products**
   Holds detailed information about each product in the inventory, including descriptions and pricing.
5. **Stores**
   Records information about various store locations where products are sold.
6. **Providers**
   Maintains data on suppliers who provide the products to the inventory.
7. **Customer_cart**
   Manages temporary storage of customer selections before finalizing purchases.
8. **Select_product**
   Facilitates the selection of products for transactions, linking customers to the products they wish to purchase.
9. **Transaction**
   Records all transactions, including sales and returns, to maintain accurate financial records.

10. **Invoice**
    Generates invoices for completed transactions, providing customers with a detailed record of their purchases.

This structured back end ensures that the IMS can efficiently manage and retrieve inventory-related data, supporting the overall functionality of the system.

Advantages of Inventory Management System

Implementing an Inventory Management System (IMS) offers numerous benefits that can enhance operational efficiency and overall business performance. The key advantages include:

Inventory Balance

Effective stock control allows businesses to determine the optimal amount of inventory needed, preventing product shortages while ensuring that the right quantities are available without overstocking.

Inventory Turnover

Maintaining a high inventory turnover ratio is crucial for minimizing spoilage and obsolescence while optimizing working capital. Regular assessment of stock sales throughout the year enables better resource utilization.

Repeat Customers

Good inventory management fosters customer loyalty by ensuring that popular items are consistently in stock. This enhances the shopping experience and encourages customers to return for future purchases.

Accurate Planning

Smart inventory management enables businesses to accurately anticipate demand, ensuring that the right products are available at the right times, especially during seasonal fluctuations. This leads to improved customer satisfaction year-round.

Warehouse Organization

Understanding top-selling products and customer ordering patterns allows for optimized warehouse layouts. Strategically grouping frequently purchased items reduces picking and packing times, accelerating shipping processes.

Employee Efficiency

Training employees in inventory management tools, such as barcode scanners and software, enhances productivity and allows the organization to make better use of its human and technological resources.

Cost Reduction

By maintaining optimal inventory levels, businesses can reduce holding costs associated with excess inventory, including storage costs, insurance, and spoilage, resulting in significant cost savings.

Improved Cash Flow

Efficient inventory management ensures that capital is not tied up in unsold stock, allowing for better cash flow management and enabling investments in other areas of the business.

Enhanced Customer Satisfaction

Timely fulfillment of customer orders and fewer stockouts lead to improved customer satisfaction and brand loyalty, as customers appreciate reliable service and product availability.

Data-Driven Insights

An inventory management system provides valuable analytics and reporting features, enabling businesses to make informed decisions based on real-time data about sales trends, customer preferences, and inventory performance.

Scalability

As businesses grow, an effective inventory management system can scale with them, accommodating increasing product lines and customer bases without compromising efficiency or accuracy.

Risk Management

By closely monitoring inventory levels and sales trends, businesses can proactively identify potential risks related to supply chain disruptions, demand fluctuations, or market changes, allowing for timely interventions.

Streamlined Operations

Automating inventory processes reduces manual labor and the likelihood of human error, streamlining operations and allowing employees to focus on more strategic tasks.

Better Supplier Relationships

Accurate inventory tracking enables businesses to maintain optimal inventory levels and engage in strategic ordering with suppliers, fostering stronger relationships and enhancing negotiation opportunities.

Regulatory Compliance

For industries subject to regulations (such as food and pharmaceuticals), an inventory management system ensures accurate record-keeping, traceability, and compliance with legal requirements.

Summary

In this project, we have developed a comprehensive back-end software solution designed to streamline inventory management processes. The system enables users to efficiently update and adjust inventory levels, forecast stock needs based on historical data, and generate invoices for sales transactions.

One of the core functionalities of this application is its ability to monitor inventory levels in real-time. If a particular stock item falls below a predefined threshold, notifications alert the manager or owner, facilitating timely reordering from suppliers to prevent stockouts and maintain optimal inventory levels. This proactive approach not only mitigates the risk of running out of inventory but also supports uninterrupted business operations.

Additionally, the software includes robust warehouse management capabilities, allowing users to add and manage multiple warehouse locations. This functionality enhances logistical efficiency by enabling better organization and distribution of stock across various storage facilities.

The application also maintains detailed customer profiles, aiding in tracking order histories and preferences for regular customers. By accessing comprehensive customer information, businesses can enhance their service offerings and tailor marketing strategies to meet customer needs more effectively.

Moreover, the system provides valuable insights into financial transactions by tracking payments received through various methods. This feature allows businesses to analyze cash flow patterns, optimize payment processes, and gain a better understanding of their financial standing.

The software promotes a high inventory turnover ratio, ensuring that products do not spoil or become obsolete, thus protecting working capital. By calculating how frequently inventory sells within a year, organizations can identify areas for improvement in resource utilization, ultimately leading to increased efficiency and profitability.

Overall, this Inventory Management System serves as an all-in-one solution that not only simplifies inventory control but also contributes to strategic decision-making, enhances operational performance, and fosters improved customer relationships. Its user-friendly interface and powerful back-end functionalities position it as an essential tool for businesses of all sizes seeking to optimize their inventory management practices.

SQL Code Implementation for Inventory Management System

1. Create Tables

```
sql
Copy code
-- Creating the Brands table
SQL> CREATE TABLE brands (
  bid NUMBER(5),
  bname VARCHAR(20)
);
Table created.

SQL> ALTER TABLE brands
  ADD PRIMARY KEY(bid);
Table altered.

-- Creating the Inventory User table
SQL> CREATE TABLE inv_user (
  user_id VARCHAR(20),
  name VARCHAR(20),
  password VARCHAR(20),
  last_login TIMESTAMP,
  user_type VARCHAR(10)
);
Table created.

SQL> ALTER TABLE inv_user
  ADD PRIMARY KEY(user_id);
Table altered.

-- Creating the Categories table
SQL> CREATE TABLE categories (
  cid NUMBER(5),
  category_name VARCHAR(20)
);
Table created.

SQL> ALTER TABLE categories
  ADD PRIMARY KEY(cid);
Table altered.

-- Creating the Products table
SQL> CREATE TABLE product (
  pid NUMBER(5) PRIMARY KEY,
  cid NUMBER(5) REFERENCES categories(cid),
  bid NUMBER(5) REFERENCES brands(bid),
  sid NUMBER(5),
  pname VARCHAR(20),
  p_stock NUMBER(5),
  price NUMBER(5),
  added_date DATE
);
Table created.

-- Creating the Stores table
SQL> CREATE TABLE stores (
  sid NUMBER(5),
  sname VARCHAR(20),
  address VARCHAR(20),
  mobno NUMBER(10)
);
Table created.

SQL> ALTER TABLE stores
  ADD PRIMARY KEY(sid);
Table altered.
```

```
SQL> ALTER TABLE product
  ADD FOREIGN KEY(sid) REFERENCES stores(sid);
Table altered.

-- Creating the Provides table
SQL> CREATE TABLE provides (
  bid NUMBER(5) REFERENCES brands(bid),
  sid NUMBER(5) REFERENCES stores(sid),
  discount NUMBER(5)
);
Table created.

-- Creating the Customer Cart table
SQL> CREATE TABLE customer_cart (
  cust_id NUMBER(5) PRIMARY KEY,
  name VARCHAR(20),
  mobno NUMBER(10)
);
Table created.

-- Creating the Select Product table
SQL> CREATE TABLE select_product (
  cust_id NUMBER(5) REFERENCES customer_cart(cust_id),
  pid NUMBER(5) REFERENCES product(pid),
  quantity NUMBER(4)
);
Table created.

-- Creating the Transaction table
SQL> CREATE TABLE transaction (
  id NUMBER(5) PRIMARY KEY,
  total_amount NUMBER(5),
  paid NUMBER(5),
  due NUMBER(5),
  gst NUMBER(3),
  discount NUMBER(5),
  payment_method VARCHAR(10),
  cart_id NUMBER(5) REFERENCES customer_cart(cust_id)
);
Table created.

-- Creating the Invoice table
SQL> CREATE TABLE invoice (
  item_no NUMBER(5),
  product_name VARCHAR(20),
  quantity NUMBER(5),
  net_price NUMBER(5),
  transaction_id NUMBER(5) REFERENCES transaction(id)
);
Table created.
```

2. Insert Data into Tables

Inserting into Brands

sql
Copy code

```
SQL> INSERT INTO brands VALUES (1, 'Apple');
1 row created.

SQL> INSERT INTO brands VALUES (2, 'Samsung');
1 row created.

SQL> INSERT INTO brands VALUES (3, 'Nike');
1 row created.

SQL> INSERT INTO brands VALUES (4, 'Fortune');
1 row created.
```

Inserting into Inventory User

sql
Copy code
```
SQL> INSERT INTO inv_user VALUES (
  'vidit@gmail.com', 'vidit', '1234', '31-oct-18 12:40', 'admin'
);
1 row created.

SQL> INSERT INTO inv_user VALUES (
  'harsh@gmail.com', 'Harsh Khanelwal', '1111', '30-oct-18 10:20', 'Manager'
);
1 row created.

SQL> INSERT INTO inv_user VALUES (
  'prashant@gmail.com', 'Prashant', '0011', '29-oct-18 10:20', 'Accountant'
);
1 row created.
```

Inserting into Categories

sql
Copy code
```
SQL> INSERT INTO categories VALUES (1, 'Electronics');
1 row created.

SQL> INSERT INTO categories VALUES (2, 'Clothing');
1 row created.

SQL> INSERT INTO categories VALUES (3, 'Grocery');
1 row created.
```

Inserting into Stores

sql
Copy code
```
SQL> INSERT INTO stores VALUES (1, 'Ram Kumar', 'Katpadi Vellore', 9999999999);
1 row created.

SQL> INSERT INTO stores VALUES (2, 'Rakesh Kumar', 'Chennai', 8888555541);
1 row created.

SQL> INSERT INTO stores VALUES (3, 'Suraj', 'Haryana', 7777555541);
1 row created.
```

Inserting into Products

sql
Copy code
```
SQL INTO product VALUES (1, 1, 1, 1, 'IPHONE', 4, 45000, '31-oct-18');
```

```
1 row created.
> INSERT

SQL> INSERT INTO product VALUES (2, 1, 1, 1, 'Airpods', 3, 19000, '27-oct-18');
1 row created.
QL> INSERT INTO product VALUES (3, 1, 1, 1, 'Smart Watch', 3, 19000, '27-oct-18');
1 row created.

SQL> INSERT INTO product VALUES (4, 2, 3, 2, 'Air Max', 6, 7000, '27-oct-18');
1 row created.

S
SQL> INSERT INTO product VALUES (5, 3, 4, 3, 'REFINED OIL', 6, 750, '25-oct-18');
1 row created.
```

Inserting into Provides

sql
Copy code

```
SQL> INSERT INTO provides VALUES (1, 1, 12);
1 row created.

SQL> INSERT INTO provides VALUES (2, 2, 7);
1 row created.

SQL> INSERT INTO provides VALUES (3, 3, 15);
1 row created.

SQL> INSERT INTO provides VALUES (1, 2, 7);
1 row created.

SQL> INSERT INTO provides VALUES (4, 2, 19);
1 row created.

SQL> INSERT INTO provides VALUES (4, 3, 20);
1 row created.
```

Inserting into Customer Cart

sql
Copy code

```
SQL> INSERT INTO customer_cart VALUES (1, 'Ram', 9876543210);
1 row created.

SQL> INSERT INTO customer_cart VALUES (2, 'Shyam', 7777777777);
1 row created.

SQL> INSERT INTO customer_cart VALUES (3, 'Mohan', 7777777775);
1 row created.
```

Inserting into Select Product

sql
Copy code

```
SQL> INSERT INTO select_product VALUES (1, 2, 2);
1 row created.

SQL> INSERT INTO select_product VALUES (1, 3, 1);
1 row created.
```

SQL> INSERT INTO select_product VALUES (2, 3, 3);
1 row created.

SQL> INSERT INTO select_product VALUES (3, 2, 1);
1 row created.

Inserting into Transactions

sql
Copy code
```
SQL> INSERT INTO transaction VALUES (1, 57000, 20000, 5000, 350, 350, 'card', 1);
1 row created.

SQL> INSERT INTO transaction VALUES (2, 57000, 57000, 0, 570, 570, 'cash', 2);
1 row created.

SQL> INSERT INTO transaction VALUES (3, 19000, 17000, 2000, 190, 190, 'cash', 3);
1 row created.
```

3. PL/SQL Functions

Function to Get Cart Details

sql
Copy code
```
SQL> DECLARE
  due1 NUMBER(7);
  cart_id1 NUMBER(7);

  FUNCTION get_cart(c_id NUMBER) RETURN NUMBER IS
  BEGIN
    RETURN (c_id);
  END;

BEGIN
  cart_id1 := get_cart('&c_id');
  SELECT due INTO due1 FROM transaction WHERE cart_id = cart_id1;
  DBMS_OUTPUT.PUT_LINE(due1);
END;
/
Enter value for c_id: 1
5000
PL/SQL procedure successfully completed.
```

4. Cursors

Cursor to Display Products

sql
Copy code
```
SQL> DECLARE
  p_id product.pid%TYPE;
  p_name product.pname%TYPE;
  p_stock product.p_stock%TYPE;
```

```
 CURSOR p_product IS
   SELECT pid, pname, p_stock FROM product;



BEGIN
 OPEN p_product;
 LOOP
   FETCH p_product INTO p_id, p_name, p_stock;
   EXIT WHEN p_product%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(p_id || ' ' || p_name || ' ' || p_stock);
 END LOOP;
 CLOSE p_product;
END;
/
```

Procedure to Check Stock

sql
Copy code

```
SQL> DECLARE
 a NUMBER;
 b NUMBER;

 PROCEDURE check_stock(x IN NUMBER) IS
 BEGIN
   IF x < 2 THEN
     DBMS_OUTPUT.PUT_LINE('Stock is Less');
   ELSE
     DBMS_OUTPUT.PUT_LINE('Enough Stock');
   END IF;
 END;

BEGIN
 b := '&b';
 SELECT p_stock INTO a FROM product WHERE pid = b;
 check_stock(a);
END;
/
Enter value for b: 2
Enough Stock
PL/SQL procedure successfully completed.
```

Summary of Expected Outputs and Results

1. **Table Creation and Alteration**
   The SQL statements for creating and altering tables will yield confirmation messages indicating success, such as:
   - "Table created."
   - "Table altered."
2. **Data Insertion**
   Executing the INSERT statements will generate messages confirming that rows have been created. For example:
   - "1 row created." (for each successful insert into the brands, inv_user, categories, stores, product, provides, customer_cart, select_product, and transaction tables).

3. **PL/SQL Function Output**
   Within the PL/SQL block for the get_cart function, the following output is expected when the input c_id = 1:
   - The output will display:

     ```
     yaml
     Copy code
     5000
     ```

4. This value is retrieved from the transaction table where cart_id is 1.
5. **Cursor Output**
   The cursor section will generate output for each product fetched from the product table, displaying:

   ```
   mathematica
   Copy code
   1  IPHONE      4
   2  Airpods     3
   3  Smart Watch   3
   4  Air Max     6
   5  REFINED OIL  6
   ```

   This output lists the pid, pname, and p_stock for each product.

6. **Procedure Output**
   The output from the check_stock procedure depends on the input for b (the pid). For example, when the input is b = 2, the output will be:
   - "Enough Stock"

Overall Summary

The code successfully establishes a simple inventory management system that integrates various components to manage brands, users, categories, products, stores, customer carts, and transactions. Feedback is provided through output messages and console logs, allowing users to track the system's operations effectively.