**Inventory Database Management**

**Problem Statement:** Design and develop a comprehensive inventory management system as a project. The system should include functionalities for tracking product details, managing suppliers, processing customer orders, and generating sales reports.

Your project should demonstrate effective database design principles and provide a user-friendly interface for easy interaction. Showcase your skills in entity-relationship modeling, data normalization, and SQL queries to efficiently retrieve and manipulate inventory data. Additionally, consider implementing security measures to protect sensitive inventory information.

**Solution**

**-- Create the table for products**

CREATE TABLE products (

product\_id NUMBER PRIMARY KEY,

product\_name VARCHAR2(100) NOT NULL,

quantity\_in\_stock NUMBER,

unit\_price NUMBER,

last\_updated DATE

);

**-- Create the table for suppliers**

CREATE TABLE suppliers (

supplier\_id NUMBER PRIMARY KEY,

supplier\_name VARCHAR2(100) NOT NULL,

contact\_name VARCHAR2(100),

address VARCHAR2(200),

phone\_number VARCHAR2(20),

email VARCHAR2(100)

);

**-- Create the table for orders**

CREATE TABLE orders (

order\_id NUMBER PRIMARY KEY,

order\_date DATE,

product\_id NUMBER,

quantity NUMBER,

total\_price NUMBER,

CONSTRAINT fk\_product\_id FOREIGN KEY (product\_id) REFERENCES products (product\_id)

);

**-- Create the table for customers**

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100) NOT NULL,

contact\_name VARCHAR2(100),

address VARCHAR2(200),

phone\_number VARCHAR2(20),

email VARCHAR2(100)

);

**-- Create the table for sales**

CREATE TABLE sales (

sale\_id NUMBER PRIMARY KEY,

sale\_date DATE,

customer\_id NUMBER,

product\_id NUMBER,

quantity NUMBER,

total\_price NUMBER,

CONSTRAINT fk\_customer\_id FOREIGN KEY (customer\_id) REFERENCES customers (customer\_id),

CONSTRAINT fk\_product\_id1 FOREIGN KEY (product\_id) REFERENCES products (product\_id)

);

**-- Create the table for employees**

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

employee\_name VARCHAR2(100) NOT NULL,

hire\_date DATE,

salary NUMBER,

manager\_id NUMBER,

CONSTRAINT fk\_manager\_id FOREIGN KEY (manager\_id) REFERENCES employees (employee\_id)

);

**Question 1**: Retrieve all products and their quantities in stock.

**Solution 1**

SELECT product\_name, quantity\_in\_stock FROM products;

**Question 2**: Get the details of all suppliers including their contact name and phone number.

**Solution 2:**

SELECT supplier\_name, contact\_name, phone\_number

FROM suppliers;

**Question 3**: Find all orders placed on a specific date along with the corresponding product and quantity.

**Solution 3:**

SELECT o.order\_id, o.order\_date, p.product\_name, o.quantity

FROM orders o

JOIN products p ON o.product\_id = p.product\_id

WHERE o.order\_date = DATE '2023-06-25';

**Question 4**: List all customers along with their addresses and emails.

**Solution 4:**

SELECT customer\_name, address, email

FROM customers;

**Question 5**: Retrieve the sales records with the sale date, customer name, and total price.

**Solution 5:**

SELECT s.sale\_date, c.customer\_name, s.total\_price

FROM sales s

JOIN customers c ON s.customer\_id = c.customer\_id;

**Question 6:** Write a cursor to retrieve the product names and quantities in stock for all products that have a quantity below 10.

**Solution 6**

DECLARE

CURSOR product\_cursor IS

SELECT product\_name, quantity\_in\_stock

FROM products

WHERE quantity\_in\_stock < 10;

BEGIN

FOR product\_rec IN product\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE('Product: ' || product\_rec.product\_name || ', Quantity in Stock: ' || product\_rec.quantity\_in\_stock);

END LOOP;

END;

/

**Question 7**: Create a trigger that updates the last\_updated column in the products table to the current date and time whenever a product's quantity\_in\_stock is updated.

**Solution 7**

CREATE OR REPLACE TRIGGER update\_last\_updated

BEFORE UPDATE OF quantity\_in\_stock ON products

FOR EACH ROW

BEGIN

:NEW.last\_updated := SYSDATE;

END;

/

**Question 8**: Write a cursor to retrieve the customer names and the total number of orders placed by each customer.

**Solution 8:**

DECLARE

CURSOR customer\_cursor IS

SELECT c.customer\_name, COUNT(\*) AS total\_orders

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_name;

BEGIN

FOR customer\_rec IN customer\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE('Customer: ' || customer\_rec.customer\_name || ', Total Orders: ' || customer\_rec.total\_orders);

END LOOP;

END;

/

**Question 9:** Create a trigger that calculates and updates the total\_price column in the orders table whenever a new order is inserted, based on the quantity and unit\_price of the product.

**Solution 9**

CREATE OR REPLACE TRIGGER calculate\_total\_price

BEFORE INSERT ON orders

FOR EACH ROW

BEGIN

:NEW.total\_price := :NEW.quantity \* (SELECT unit\_price FROM products WHERE product\_id = :NEW.product\_id);

END;

/

**Question 10:** Write a cursor to retrieve the employee names and the names of their respective managers (if applicable).

**Solution 10:**

DECLARE

CURSOR employee\_cursor IS

SELECT e.employee\_name, m.employee\_name AS manager\_name

FROM employees e

LEFT JOIN employees m ON e.manager\_id = m.employee\_id;

BEGIN

FOR employee\_rec IN employee\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || employee\_rec.employee\_name || ', Manager: ' || employee\_rec.manager\_name);

END LOOP;

END;

/