```
#Inventory Replenishment System: Create a Python script that monitors
#inventory levels of products in a shop and generates replenishment orders when
stock levels
#fall below predefined thresholds. The script should analyze sales data to predict
future demand
#and adjust reorder points accordingly. Implement functionality to send automated
restock
#alerts to suppliers and track the status of replenishment orders.

# PYYTHON PROJECT
# NAME : ALEENA SUSAN PRAKASH
# ROLL NO. :09
# GROUP 9
# DATE : 17-03-2025

import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart


#Initial data
product_id=[101,102,103,104,105]
items=["Mobile phone", "Laptop", "Earphones" ,"Power bank" ,"Wireless kit"]
currentStock=[20, 20, 10, 10, 45]
minimumRequired=[15, 15, 10, 15, 30]

# [ Sales Last 30 Days, Sales Last 60 Days, Sales Last 90 Days]
sales_data = [
    [ 101, 50, 120, 180],
    [ 102,120, 250, 350],
    [ 103,30, 80, 120],
    [ 104,50,40,60],
    [ 105,10,50,70]
]


#Display current stock levels

def display_stock():
    print("\n Current Stock Levels : ")
    for i in range (len(items)):
        print(f"{items[i] }: {currentStock[i] }")

#Add stock to an existing item

def add_to_existing_stock():
    display_stock()
    item_name=input("\nEnter the item name to add stock :")
    if item_name in items:
        quantity=int(input(f"Enter the quantity to add for {item_name} : "))
        index=items.index(item_name)
        currentStock[index]+=quantity #Update stock
        print(f"Added {quantity} to {item_name} . New stock :
{currentStock[index]}")
    else:
        print("Invalid item name.")

#Delete stock from an existing item

def delete_from_existing_stock():
    display_stock()
    item_name = input("Enter the item name to delete stock : ")
    if item_name in items:
        quantity = int(input(f"Enter the quantity to delete from  {item_name} : "))
        index=items.index(item_name)
        if quantity <=currentStock[index]:
            currentStock[index]-=quantity #Deduct stock
            print(f"Deleted {quantity} from {item_name}. New stock :
{currentStock[index]}")
        else:
            print(f"Cannot delete {quantity} from {item_name}. Current stock is
only {currentStock[index]} ")
    else:
        print("Invalid item name.")

#Check if restocking is required

def check_restocking():
    print("\n Checking for restocking needs...")
    restocking_needed=False
    for i in range (len(items)):
        if currentStock[i] < minimumRequired[i]:
            print(f"Restocking needed for {items[i]} . (Current : {currentStock[i]}
, Minimum Required : {minimumRequired[i]})")
            restocking_needed=True
            sender_email = "your-email@example.com"
            receiver_email = "supplier-email@example.com"
            password = "your-email-password"

            # Create email
            subject = f"Restock Alert: {items[i]} needs replenishment"
            body = f"Dear Supplier,\n\nPlease note that the stock level for
{items[i]} is below the reorder point.Current stock: {currentStock[i]}\nReorder
point: {minimumRequired[i]}\n\nPlease process the replenishment order."


            msg = MIMEMultipart()
            msg['From'] = sender_email
            msg['To'] = receiver_email
            msg['Subject'] = subject
            msg.attach(MIMEText(body,'plain'))

            # Sending email
            try:
                with smtplib.SMTP('smtp.gmail.com', 587) as server:
                    server.starttls()
                    server.login(sender_email, password)
                    text = msg.as_string()
                    server.sendmail(sender_email, receiver_email, text)
                    print(f"Restock alert sent for {items[i]}.")
            except Exception as e:
                print(f"Failed to send email: {e}")
    if not restocking_needed:
        print("All stock levels are sufficient.")


# Function to predict future demand (using simple moving average)
def predict_demand():
    product_id=int(input("Enter the product id (101-105) :"))
    # Find the product in sales data
    for sales in sales_data:
        if sales[0] == product_id:
            sales_30 = sales[1]
            sales_60 = sales[2]
            sales_90 = sales[3]
            # Simple moving average of last 30, 60, and 90 days
            predicted_demand = (sales_30 + sales_60 + sales_90) / 3
            print(f"The demand for the product with id {product_id}")
            return predicted_demand
    return 0

#Menu-driven interface

while True:
    print("\nMenu")
    print("1. Display Stock.")
    print("2. Add to existing stock.")
    print("3. Delete from existing stock.")
    print("4. Check Restocking Needs .")
    print("5. Predict future demand .")
    print("6. Exit")
    choice=int(input("Enter your choice (1-6) : "))
    if choice==1:
        display_stock()
    elif choice==2:
        add_to_existing_stock()
    elif choice==3:
        delete_from_existing_stock()
    elif choice==4:
        check_restocking()
```

```
elif choice==5:
    for i in range(len(items)):
        print(f"{items[i]} : {product_id[i]}")
    demand_for_product=predict_demand()
    print(demand_for_product)
elif choice==6:
    print("Exiting....")
    break
else:
    print("Invalid choice . Please Try again.")
```