1) **Develop an Employee Management System that manages employee records, their salary details, and allows adding, updating, deleting employees and saving these records to a file.**

Instructions:

a) Define a class Employee with the following attributes:
   a. id: Unique employee ID.
   b. name: Name of the employee.
   c. salary: Salary of the employee.
   d. department: Department in which the employee works.

b) The Employee class should have the following methods:
   a. increase_salary(percentage): Increases the employee's salary by the given percentage.
   b. update_department(new_department): Updates the employee's department.
   c. display_employee_details(): Displays all the details of the employee.

c) Define a class EmployeeDatabase which manages the employees:
   a. add_employee(employee): Adds a new employee to the system.
   b. remove_employee(employee_id): Removes an employee from the system by their unique id.
   c. update_employee_details(employee_id, new_details): Updates employee information.
   d. display_all_employees(): Displays the details of all employees.
   e. save_to_file(filename): Saves employee records to a text file.
   f. load_from_file(filename): Loads employee records from a file into the system.

d) Implement a simple **menu-driven** system that allows the user to:
   a. Add new employees.
   b. Remove employees.
   c. Update employee details (salary, department).
   d. Display all employees.
   e. Save and load records from a file.

2) **Build a Social Media Post Scheduler system where users can schedule and manage their posts. The system should allow adding posts, checking scheduled posts, editing, and deleting posts. Posts will be saved to a file.**

Instructions:

a) Create a class Post with the following attributes:
   a. content: The content of the post (text).
   b. scheduled_time: The time when the post is scheduled to be made.
   c. platform: The social media platform (e.g., Facebook, Instagram, Twitter).

b) The Post class should have the following methods:
   a. edit_content(new_content): Edits the content of the post.
   b. change_scheduled_time(new_time): Changes the scheduled time of the post.
   c. display_post_details(): Displays the post details.

c) Create a class Scheduler which manages posts:
   a. add_post(post): Adds a new post to the schedule.
   b. remove_post(post_id): Removes a post from the schedule.
   c. view_all_posts(): Displays all the scheduled posts.
   d. save_posts_to_file(filename): Saves scheduled posts to a text file.
   e. load_posts_from_file(filename): Loads scheduled posts from a file.

d) Implement a **menu system** that allows the user to:
   a. Add new posts.
   b. Edit post content and scheduled time.
   c. Remove posts.
   d. View all scheduled posts.
   e. Save and load posts to/from a file.

   Example:

   Add a post with content "Hello World" scheduled for 3:00 PM on Twitter.

   Edit the content to "Hello Universe" and reschedule it for 4:00 PM.

3) **Design a more advanced Banking System that allows users to deposit money, withdraw money, view transaction history, and calculate loan repayments with interest.**
Instructions:

a) Define a class BankAccount with the following attributes:
   a. account_number: Unique identifier for the bank account.
   b. account_holder: Name of the account holder.
   c. balance: Current balance in the account.
   d. transaction_history: List that stores all transactions (deposit/withdrawal).
   e. loan_balance: The balance of any ongoing loan.

b) The BankAccount class should have the following methods:
   a. deposit(amount): Adds the specified amount to the account balance and records the transaction.
   b. withdraw(amount): Deducts the specified amount if sufficient funds are available and records the transaction.
   c. view_balance(): Displays the current balance.
   d. view_transaction_history(): Displays the list of all past transactions.
   e. apply_for_loan(amount, interest_rate, term_years): Calculates and adds the loan balance (principal + interest) to the account. The loan is calculated using the formula: Loan Amount=Principal×(1+Interest Rate)Term in Years\text{Loan Amount} = \text{Principal} \times (1 + \text{Interest Rate})^{\text{Term in Years}}Loan Amount=Principal×(1+Interest Rate)Term in Years
   f. view_loan_balance(): Displays the loan balance.

c) Implement a **Banking System** class that manages multiple accounts:
   a. create_account(account_holder): Creates a new account for a user.
   b. find_account(account_number): Finds an account by account number.
   c. save_to_file(filename): Saves account and transaction data to a file.
   d. load_from_file(filename): Loads account data from a file.

d) Create a **menu system** where the user can:
   a. Deposit and withdraw money.
   b. Apply for loans.
   c. View account balance and transaction history.

d. Save and load account data from files.

4) **Design an Automated Grocery Inventory System that helps manage inventory by tracking product quantities, adding new products, updating quantities, and generating sales reports.**

Instructions:

a) Define a class Product with the following attributes:
   a. product_id: Unique identifier for each product.
   b. name: Name of the product.
   c. quantity: Available stock of the product.
   d. price: Price per unit of the product.

b) The Product class should have the following methods:
   a. update_quantity(new_quantity): Updates the stock quantity of the product.
   b. get_total_value(): Returns the total value of the stock (quantity * price).
   c. display_product_details(): Displays product details (ID, name, price, quantity).

c) Define a class InventorySystem which manages the inventory:
   a. add_product(product): Adds a new product to the inventory.
   b. update_product_quantity(product_id, new_quantity): Updates the stock quantity of an existing product.
   c. remove_product(product_id): Removes a product from the inventory.
   d. view_inventory(): Displays all products and their details.
   e. generate_sales_report(filename): Generates a report on sales, including product names and quantities sold, and saves it to a file.
   f. load_inventory_from_file(filename): Loads the inventory from a file.

d) Implement a **menu-driven system** where the user can:
   a. Add products to the inventory.
   b. Update quantities.
   c. Generate sales reports.
   d. Save and load inventory data to/from files.

5) **Create a CSV file named sales_data.csv that contains the following columns:**

   - Date: Date of the sale (in YYYY-MM-DD format).
   - Product: Product name (e.g., "Laptop", "Mobile", "Headphones").
   - Units Sold: The number of units sold.
   - Revenue: The total revenue from the sale.
   - Category: The category of the product (e.g., "Electronics", "Accessories").

- Use **Pandas** to load the CSV file into a **DataFrame** and inspect the first few rows.
- Check for missing values in the dataset and handle them appropriately (either drop or fill).
- Ensure that the Date column is in the correct datetime format.
- Remove any rows where Units Sold or Revenue is negative (if any).

Data Analysis

- Calculate the **total revenue** generated from each product using **groupby**.
- Calculate the **average units sold** per product.
- Identify the product with the **highest revenue** and **highest units sold**.

Data Visualization

- Using **Matplotlib**, plot the following visualizations:
  - A **bar chart** showing the total revenue by product.
  - A **line plot** showing the trend of Revenue over time (based on the Date column).
  - A **pie chart** showing the distribution of revenue across different categories.

Bonus Calculation

- Calculate the **growth rate** of revenue for each product by comparing the total revenue in the first month of data and the last month of data.
- Display the product(s) with the **highest growth rate**.