**Exercise 2 : E-Commerce platform search function**

**Product.java**

import java.util.\*;

public class ProductSearch {

    public static void main(String[] args) {

        ProductManager manager = new ProductManager();

        manager.addProduct(new Product("Blue Running Shoes", "Footwear", "Comfortable shoes for running", 1299));

        manager.addProduct(new Product("Adidas Sports Shoes", "Footwear", "High-quality sportswear", 2499));

        manager.addProduct(new Product("Red T-Shirt", "Apparel", "Cotton casual t-shirt", 599));

        manager.addProduct(new Product("Blue Jeans", "Apparel", "Slim-fit jeans for men", 1199));

        Scanner sc = new Scanner(System.in);

        System.out.print("Search (type something): ");

        String input = sc.nextLine();

        System.out.println("\nAutocomplete Suggestions:");

        for (String s : manager.getSuggestions(input)) {

            System.out.println(" → " + s);

        }

        System.out.println("\nSearch Results:");

        for (Product p : manager.search(input)) {

            System.out.println(" - " + p);

        }

        sc.close();

    }

}

class Product {

    String name;

    String category;

    String description;

    double price;

    public Product(String name, String category, String description, double price) {

        this.name = name;

        this.category = category;

        this.description = description;

        this.price = price;

    }

    @Override

    public String toString() {

        return name + " - ₹" + price;

    }

    public String getName() {

        return name.toLowerCase();

    }

    public String getDescription() {

        return description.toLowerCase();

    }

}

class ProductManager {

    private List<Product> products = new ArrayList<>();

    public void addProduct(Product p) {

        products.add(p);

    }

    public List<String> getSuggestions(String input) {

        input = input.toLowerCase();

        List<String> suggestions = new ArrayList<>();

        for (Product p : products) {

            if (p.getName().contains(input)) {

                suggestions.add(p.name);

            }

        }

        return suggestions;

    }

    public List<Product> search(String input) {

        input = input.toLowerCase();

        List<Product> result = new ArrayList<>();

        for (Product p : products) {

            if (p.getName().contains(input) || p.getDescription().contains(input)) {

                result.add(p);

            }

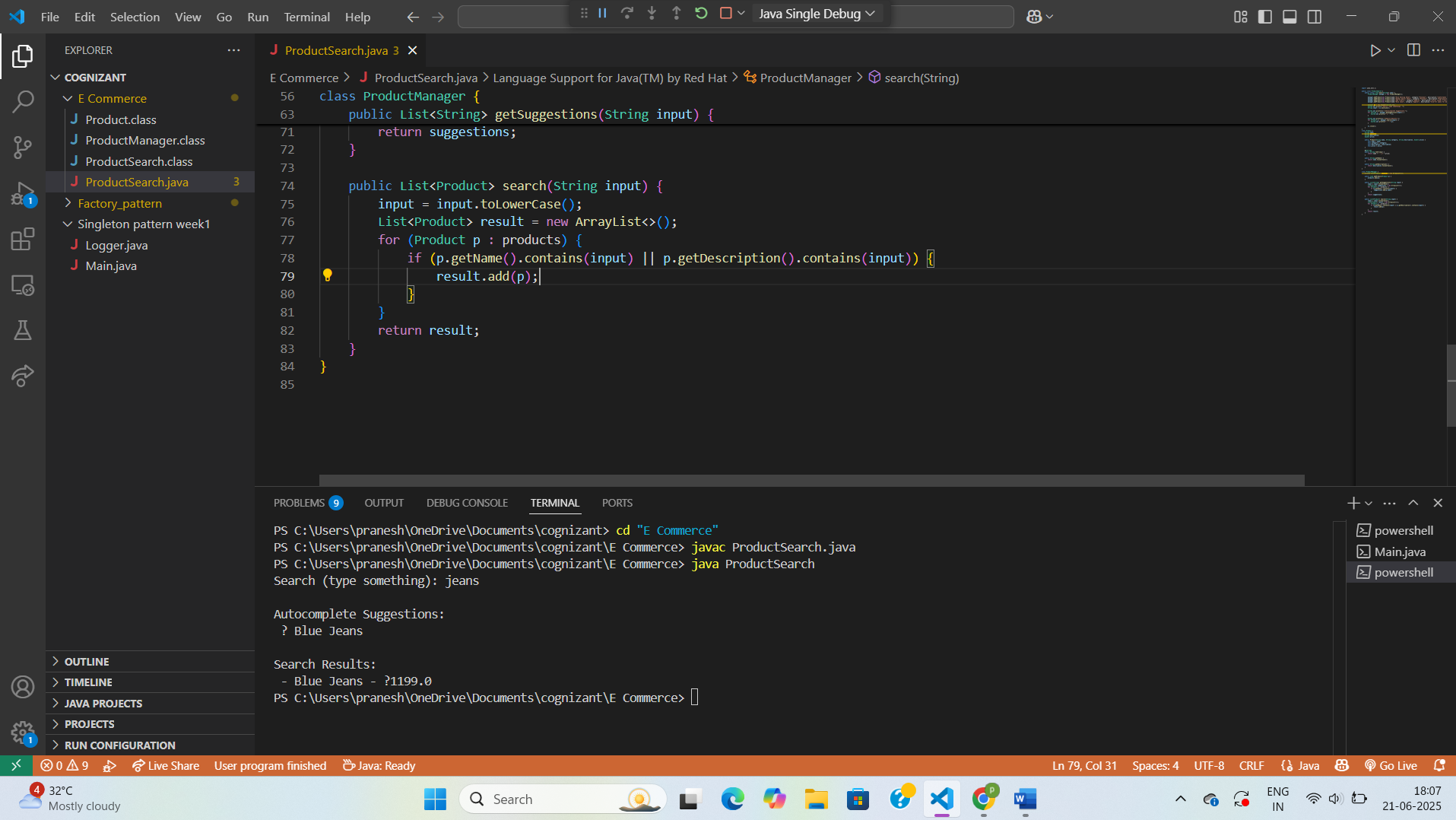
        }

        return result;

    }

}

**OUTPUT:**

****

**Exersice 7:Financial Forecasting**

**MovingArea.java**

import java.util.\*;

public class MovingAverage {

    public static List<Double> calculate(int[] data, int k) {

        List<Double> result = new ArrayList<>();

        double sum = 0;

        for (int i = 0; i < data.length; i++) {

            sum += data[i];

            if (i >= k) {

                sum -= data[i - k];

            }

            if (i >= k - 1) {

                result.add(Math.round((sum / k) \* 100.0) / 100.0);

            }

        }

        return result;

    }

}

class LinearTrendPredictor {

    public static double predictNext(int[] revenue) {

        int n = revenue.length;

        double sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0;

        for (int i = 0; i < n; i++) {

            sumX += i;

            sumY += revenue[i];

            sumXY += i \* revenue[i];

            sumX2 += i \* i;

        }

        double slope = (n \* sumXY - sumX \* sumY) / (n \* sumX2 - sumX \* sumX);

        double intercept = (sumY - slope \* sumX) / n;

        return Math.round((slope \* n + intercept) \* 100.0) / 100.0;

    }

}

**Main.java**

import java.util.\*;

public class Main {

    public static void main(String[] args) {

        int[] revenue = {1200, 1300, 1250, 1400, 1500, 1600, 1550, 1650};

        System.out.println("Past Revenue (Last 8 Days):");

        System.out.println(Arrays.toString(revenue));

        System.out.println("\n 3-Day Moving Averages:");

        List<Double> movingAvg = MovingAverage.calculate(revenue, 3);

        for (double avg : movingAvg) {

            System.out.println(" → " + avg);

        }

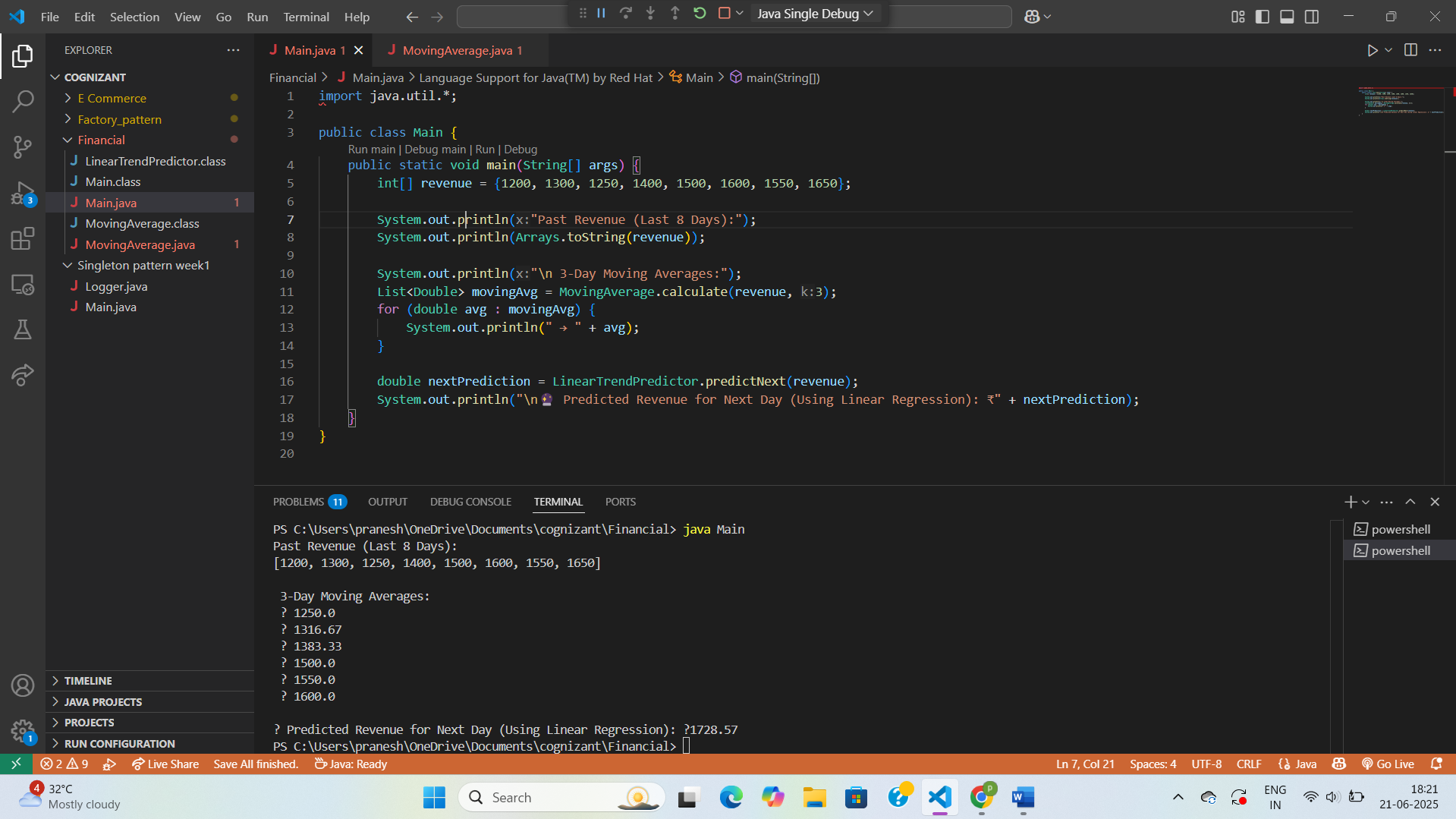
        double nextPrediction = LinearTrendPredictor.predictNext(revenue);

        System.out.println("\n Predicted Revenue for Next Day (Using Linear Regression): ₹" + nextPrediction);

    }

}

**Output:**

****