HANDS-ON

Hands-on No. : **9**

Topic : **Stream API Advanced**

Date : **12.08.2025**

Solve the following problems

Question No.	Question Detail
1	Given a list of Product objects (with fields: id, name, price, rating, category, brand), write a method to: • Sort products **first by category alphabetically** • Then by **descending rating** • And finally by **ascending price**
2	Given a list of Employee objects (with fields: name, department, salary, designation), group employees by: • Department, and • Within each department, group by designation.
3	From a list of User objects (with fields: id, name, email, isVerified), write a function to: • Find the first **unverified user whose email is null or blank**. • Return the result as an Optional<User>. • If no such user exists, return an empty Optional and print a custom message using ifPresentOrElse.
4	Given a list of Order objects (with fields: orderId, customerId, orderAmount, status, orderDate), write a function to: • Calculate the **total order amount for each customer** using Collectors.groupingBy and Collectors.summingDouble. • Identify the **customer with the highest total purchase**. • Format the result into a sorted Map in descending order of total purchase amount.
5	Given a list of Student objects (with fields: id, name, marks, className, gender), perform the following using a single stream pipeline: 1. Filter students with marks > 75 2. Group them by className 3. Within each class, calculate the **average marks by gender**

It is going to be hard but, hard does not mean impossible.

	4. Return a nested Map: Map<String, Map<String, Double>> (class → gender → avgMarks)

6	Given a list of Transaction objects (fields: id, accountId, type, amount, date), perform the following operations in a single stream:
	• Filter transactions of type "CREDIT"
	• Sort by date descending
	• Group by accountId
	• For each account, compute:
	  ○ Total credited amount
	  ○ Most recent transaction

7	Given a list of InventoryItem (fields: name, stock, price, supplier), write a method to:
	• Group items by supplier
	• For each supplier:
	  ○ Find the **item with the lowest stock**
	  ○ Use Optional to handle missing data
	  ○ Return a Map of supplier to Optional<Item>

8	Given a list of Book objects (fields: title, author, price, rating), partition the books into: • Highly rated (rating >= 4.5) • Others Then within each partition, collect titles into a **sorted set**.
9	Given a list of Flight objects (fields: flightNo, destination, duration, airline, fare), group all flights by destination, then sort the list of flights: • By ascending fare • Then by ascending duration Return: Map<String, List<Flight>> with each list sorted.
10	From a list of Product objects (fields: name, price, unitsSold), compute: • Total revenue (price * unitsSold) • Average product price • Min and max priced products • Count of products *Use:* Collectors.summarizingDouble, collectingAndThen, or custom collectors •