**Case Study: The "TrendSpotter" Sales Analytics Platform**

**The Scenario:**  
You are a developer at "TrendSpotter," a fast-growing e-commerce company. The company runs a critical nightly job that processes the previous day's sales data. This job reads transaction files, enriches the data, calculates new product popularity scores, and generates a summary report.

**The Problem:**  
The job is in trouble. It's taking longer and longer to run each night. Last night, it ran for three hours before crashing with a java.lang.OutOfMemoryError. Your team lead, Sarah, has tasked you with profiling the application to find out what's wrong.

**Your Mission:**  
Use a suite of profiling tools (VisualVM, JProfiler, Eclipse MAT) to identify and document all major performance, memory, I/O, and concurrency bottlenecks in the SalesAnalyticsApplication. You are not to fix the code yet; your task is to produce a detailed report of the problems you find.

**Investigation Workflow**

1. **Compile and Run the Application:**  
   Compile the code and run it from your terminal. We need to give it a limited amount of memory to make the OutOfMemoryError happen faster.

# Run with a 256MB heap and enable heap dumps on crash. Run as a java app with the following flags added to the VM

-Xmx256m

-XX:+HeapDumpOnOutOfMemoryError

1. **Attach Your Profiler:**  
   While the application is running, find its Process ID (PID) from the console output and attach your preferred profiler (VisualVM or JProfiler or GCViewer or JMeter or MAT).

**Final Report**

Once you have completed your investigation, write a report for Sarah summarizing your findings. Structure it like this:

* + **Class & Method:** [Your finding here]
  + **Evidence:** Screenshots
  + **Explanation about the bottleneck**