Examen — Projet Final
Système de Recommandation E-Commerce en Temps Réel

Ingénierie des Données Avancée et Streaming
Master

Février 2026

1 Contexte et Objectifs

L'objectif de cet examen est de concevoir et d'implémenter un **pipeline de traitement de données en streaming** pour un cas d'usage e-commerce.

Le système doit :
— ingérer des événements utilisateurs via Kafka,
— traiter ces événements en **Event Time** avec Apache Flink,
— détecter des anomalies de comportement,
— être entièrement reproductible.

Important : aucune démonstration live n'est requise pour cet examen.

2 Objectifs Techniques Attendus

Le projet doit démontrer :
— une ingestion correcte d'événements JSON,
— l'utilisation explicite de timestamps et watermarks,
— un traitement par fenêtres temporelles,
— une détection d'anomalies justifiée,
— une architecture cohérente et expliquée.

3 Format des Données

Les événements sont publiés dans le topic Kafka `ecommerce-events`.

Listing 1 – Exemple d'événement JSON

```
{
  "user_id": 12,
  "event": "click",
  "product_id": 42,
  "timestamp": 1700000000.123
}
```

Contraintes :
— `timestamp` est exprimé en secondes UNIX (float),
— la conversion en millisecondes est requise dans Flink,
— des événements peuvent arriver en retard.

Question associée — Qualité et temporalité des données

- Pourquoi est-il essentiel de disposer d'un champ `timestamp` explicite ?
- Quelle est la différence entre **Event Time** et **Processing Time** ?
- Quels problèmes peuvent apparaître si les timestamps sont absents ou incorrects ?

4 Détection d'Anomalies

Une anomalie est détectée lorsque, pour un utilisateur donné, le nombre de clics dans une fenêtre temporelle d'une minute dépasse un seuil $\mathbf{T = 50}$.

La méthode de détection peut être :

- un comptage exact par fenêtre,
- ou un algorithme approché (ex. Count-Min Sketch).

Question associée — Logique de détection d'anomalies

- Quelle définition d'anomalie avez-vous retenue ?
- Pourquoi cette définition est-elle pertinente pour un système e-commerce ?
- Quel rôle jouent les fenêtres temporelles dans cette détection ?

5 Architecture de Référence

Question associée — Architecture et tolérance aux pannes

- Quel est le rôle de Kafka dans cette architecture ?
- Pourquoi Flink est-il adapté à ce type de traitement ?
- Que se passe-t-il si Kafka ou Flink devient temporairement indisponible ?

6 Utilisation des Scripts Fournis

Les scripts fournis permettent de reproduire l'exécution du pipeline.

`producer.py`

Génère en continu des événements e-commerce vers Kafka.

`anomaly_simulator.py`

Simule un comportement anormal (rafale de clics pour un utilisateur).

`pipeline.py`

Job Flink réalisant le traitement Event Time, l'agrégation par fenêtres et la détection d'anomalies.

Question associée — Reproductibilité

- Quel est l'ordre correct d'exécution des scripts ?
- Comment vérifiez-vous que le pipeline fonctionne correctement ?
- Quelles erreurs courantes peuvent apparaître lors de l'exécution ?

7 Livrables et Évaluation

Livrables attendus

- Code source (ZIP ou dépôt Git),
- Rapport PDF (10 pages maximum),
- Preuves d'exécution (logs, captures d'écran).

Question associée — Analyse critique

- Quelles sont les limites actuelles de votre pipeline ?
- Quelles améliorations proposeriez-vous (performance, scalabilité, précision) ?
- Que faudrait-il ajouter pour un déploiement en production ?

Barème (sur 20)

- Pipeline fonctionnel et reproductible : 8 pts
- Event Time, watermarks et fenêtres : 5 pts
- Détection d'anomalies : 3 pts
- Qualité du code et des tests : 2 pts
- Rapport et argumentation : 2 pts