🔷 1. Rebalance storm (rééquilibrage fréquent)

📖 Le problème :

Quand un consumer rejoint ou quitte un consumer group, Kafka déclenche un rebalance : il redistribue les partitions entre les consumers.
C'est normal… mais si ça arrive trop souvent, c'est catastrophique :

- Les consumers arrêtent temporairement de lire.
- Les offsets peuvent être commités en retard ou perdus.
- Le lag explose.

💡 Causes typiques :

- Consumer qui crashe souvent ou met trop de temps à répondre au heartbeat.
- Mauvaise config de `session.timeout.ms` et `max.poll.interval.ms`.

✅ Solution :

- Ajuster ces timeouts.
- S'assurer que le traitement d'un batch est rapide.
- Utiliser le cooperative rebalancing (introduit dans Kafka 2.4+) pour des transitions plus douces.

🔷 2. Message Duplication (doublons de messages)

📖 Le problème :

Un message peut être traité deux fois si le consumer lit un message mais crashe avant le commit d'offset.

💡 Conséquence :

Ton application traite le même événement deux fois (paiement, email, etc.).
C'est critique.

✅ Solutions :

- Implémenter une logique idempotente côté consommateur (ne pas réappliquer deux fois le même message).
- Activer le mode exactly-once semantics (EOS) si ton broker et client le supportent (`enable.idempotence=true` + transactions Kafka).

🔷 3. Data Skew (déséquilibre de partitions)

📖 Le problème :

Une seule partition reçoit la majorité des messages, les autres restent quasi vides.
Résultat :

- Un consumer travaille beaucoup plus que les autres.
- Perte de parallélisme, lag artificiel.

💡 Causes :

- Mauvaise clé de partitionnement dans le producer (`key` trop statique ou vide).

✅ Solution :

- Choisir une clé de partitionnement équilibrée (hash équitable).
- Ou répartir aléatoirement si l'ordre n'a pas d'importance.

🔷 4. Message Loss (perte de messages)

📖 Le problème :

Peu fréquent, mais grave.
Arrive si :

- Le producer n'a pas activé les acks corrects.
- Un broker crash avant la réplication complète.
- Un offset est commit avant la fin du traitement.

✅ Solution :

- Configurer `acks=all` côté producer.
- Utiliser une réplication ≥ 2.
- Committer les offsets après traitement réussi (jamais avant).

🔷 5. Out-of-order delivery (désordre des messages)

📖 Le problème :

Les messages arrivent dans un ordre différent de celui de production.
Kafka garantit l'ordre dans une partition, pas à l'échelle du topic.
Si un message d'un même client se retrouve sur deux partitions différentes, l'ordre n'est plus garanti.

✅ Solution :

•Utiliser la même clé de partition pour tous les messages d’un même contexte (client, commande, etc.).
