RFM = **Recency, Frequency, Monetary**

**Quais UFs concentram os clientes mais valiosos (High)?**

**1) Job 1 — “Por cliente” → classifica e atribui UF**

**1.1 Mapper (RfmClientMapper)**

• Lê cada linha do CSV, ignora cabeçalho e **parseia** com split seguro para aspas.

• Extrai campos úteis: client\_id, date, amount, merchant\_city, merchant\_state, etc.

• Emite **uma chave por cliente**:  
**KEY:** client\_id  
**VALUE:** um TransactionRfmWritable com:

◦ timestamp (para calcular **recency** do cliente),

◦ amount em centavos (para **monetary**),

◦ city/state (para saber **território predominante**),

◦ (outros, se você incluiu: mcc, flags, etc.).

Por que assim? O reducer receberá “todas as transações de um mesmo cliente” para agregar tudo de uma vez.

**1.2 Reducer (RfmClientReducer)**

Para cada client\_id:

**1 Agrega eventos** do cliente:

**◦ F (frequency):** tx = número de transações.

**◦ M (monetary):** soma e média por transação (avgCents), e **pico** (maxCents).

**◦ R (recency):** diferença (em dias) entre rfm.reference.date e a **última** compra do cliente.

**◦ Território predominante:** topState = argmax(merchant\_state) e topCity = argmax(merchant\_city).

**2 Classifica** o cliente:

◦ Usa thresholds do Configuration (vindos do Driver ou da linha de comando).

◦ Compara recency, freq, avgCents e, se você manteve, maxCents para decidir **LOW/MED/HIGH**.

◦ A ideia: **HIGH** são clientes “muito ativos e/ou muito valiosos” (freq/valor alto e/ou recência muito recente).

**3 Emite 1 linha por cliente**, chaveada na **UF predominante**:

**◦ KEY:** topState

**◦ VALUE (texto):** "1:low:med:high|CITY=1"

▪ 1 é o total de **1 cliente**.

▪ Exatamente **um** entre low:med:high será 1; os outros 0.

▪ Após | vai **CITY=1** **somente** se o cliente for **HIGH** (para contagem de hotspots).

**Exemplos de valores do Stage 1:**

SP 1:0:1:0|

RJ 1:0:0:1|RIO DE JANEIRO=1

MG 1:1:0:0|

Observação: emitir **texto legível** facilita depurar com cat e alimenta o Job 2 sem conversão adicional.

**2) Job 2 — “Por UF” → consolida distribuição e hotspots**

**2.1 Mapper (StateAggMapper)**

• Lê as linhas do Job 1, cada uma no formato:  
STATE \t total:low:med:high|cidade1=n1,cidade2=n2,...

**• Parseia** o texto e monta um StateClientAggWritable:

**◦ Contadores somáveis:** total, low, med, high.

**◦ Mapa somável:** cityHighMap (cidade → nº de clientes **HIGH**).

**2.2 Combiner (StateAggCombiner)**

**• Soma localmente** os contadores e **faz merge** dos mapas de cidades antes de mandar ao reducer.  
Isso reduz tráfego (escala melhor).

• O merge é associativo/comutativo (seguro para combiner).

**2.3 Reducer (StateAggReducer)**

**• Consolida por UF**: soma todos os parciais (contadores e mapa de cidades).

• Calcula **percentuais**: Low%, Med%, High% sobre o total de clientes da UF.

• Ordena o mapa de cidades por número de **clientes HIGH** e pega o **Top N** (parâmetro top.cities).

**• Emite o resumo final legível**:

**◦ KEY:** STATE

**◦ VALUE (texto):**  
Clients: T | Low: L (p%) | Med: M (p%) | High: H (p%) | Top Cities (High Risk): C1:n1 | C2:n2 | ...

**Exemplo de saída final:**

SP Clients: 12345 | Low: 6543 (53.00%) | Med: 4012 (32.50%) | High: 1790 (14.50%) | Top Cities (High Risk): SAO PAULO: 950 | CAMPINAS: 210 | SANTOS: 130

(Opcional) Regras de estabilidade como min.uf.clients podem suprimir UFs com poucos clientes para evitar leituras enganosas.

**O que dá para concluir**

**1 Onde estão os clientes mais valiosos (agora):**

◦ UFs com **maior % de HIGH** (e/ou maior **número absoluto** de HIGH) são seus “**Estados premium**”.

◦ As **Top Cidades (High Risk / High Value)** apontam **hotspots** para **campanhas, vendas, pricing e suporte**.

**2 Qual é o “perfil” de cada UF:**

◦ Uma UF com **muito MED** pode ter boa **base ativa** mas sem “altíssimo valor” — ótimo alvo para **upsell** (elevar avg ticket/frequência).

◦ Uma UF com **muito LOW** pode ser **mercado pouco engajado** ou com **recência ruim** — alvo para **reativação**.

**3 Concentração vs. dispersão:**

◦ Se o Top Cidades é muito concentrado (ex.: 70% dos HIGH só em 2 cidades), você tem **poucos polos** dominantes.

◦ Se há muitas cidades com counts relevantes, a **oportunidade é mais distribuída** (estratégia de capilaridade).

**4 Risco/Valor ao mesmo tempo:**

◦ (Se você usa o pipeline de **Comportamento Online/Erros** em paralelo) dá para cruzar:

▪ UFs/cidades **valiosas** (RFM) com **alto onlineRate/erroRate** (comportamento) → priorizar **fraude/risco** vs **crescimento**.