**ANALYSE DES DONNEES AVEC HADDOP**

***NB*** : *Étant donné que nous travaillons sur le terminal de la machine, nous ne pouvons pas fournir de lien vers les codes sources. C'est pourquoi nous avons rédigé ce document avec des captures d'écran pour illustrer nos actions et pour pouvoir revenir plus en détail sur les codes lors de la présentation.*

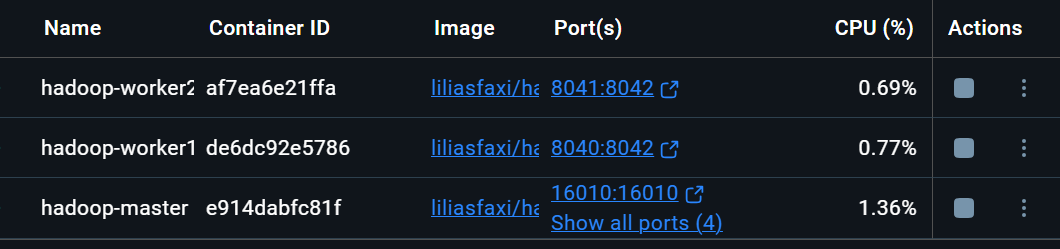
**Objectif du TP :**

La base de données **"purchases.txt"** contient plus de 4 millions de lignes et 6 variables, représentant des transactions commerciales enregistrées avec la date, l'heure, la ville, la catégorie de produit acheté, le montant de la transaction et le moyen de paiement utilisé. Chaque ligne correspond à une vente effectuée dans une ville donnée, avec des catégories variées telles que les vêtements, la musique ou encore l’électronique, et des paiements réalisés via différents moyens comme Visa, MasterCard ou en espèces. L'objectif de l'analyse est de traiter cette base avec **Hadoop Streaming**, en utilisant des scripts Python pour calculer les ventes totales par ville, ou par catégories, de calculer la moyenne des ventes etc... Il sera donc question de répondre aux cinq questions qui suivent :

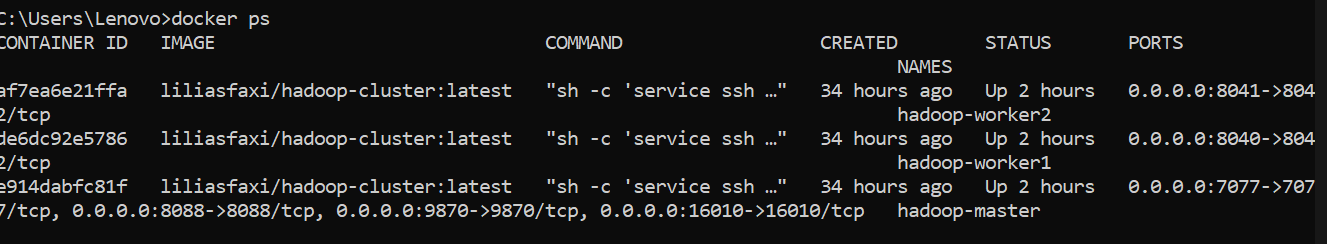
* **Total des ventes par ville** : Combien d'argent a été dépensé dans chaque ville ?
* **Total des ventes par catégorie de produit** : Quel est le total des ventes pour chaque catégorie ?
* **Total des ventes par méthode de paiement** : Quelles sont les méthodes de paiement les plus utilisées ?
* **Ventes moyennes par transaction** :Quelle est la dépense moyenne par transaction dans chaque ville et par catégorie ?
* **Analyse de la répartition temporelle** : Y a-t-il des tendances saisonnières dans les achats ?

1. **Lancement des conteneurs Hadoop dans Docker**

Avant de charger les données dans HDFS, il est nécessaire de démarrer les conteneurs Docker contenant les services Hadoop. Cette étape permet d’exécuter Hadoop dans un environnement isolé et reproductible. Pour cela, on utilise une image préconfigurée d’Hadoop et on lance un conteneur qui inclut HDFS et YARN. Une fois les conteneurs en cours d’exécution, on peut interagir avec le système de fichiers HDFS pour charger et traiter les données efficacement.

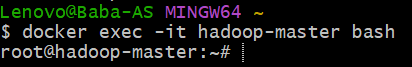
Pour le faire, on ouvre Docker et on active les conteneurs comme le montre l’image suivante suivante : 

Une fois les conteneurs lancer nous pouvons vérifier s’ils fonctionnent en exécutant : « docker ps ». Voici le résultat :



1. **Entrer dans le conteneur**

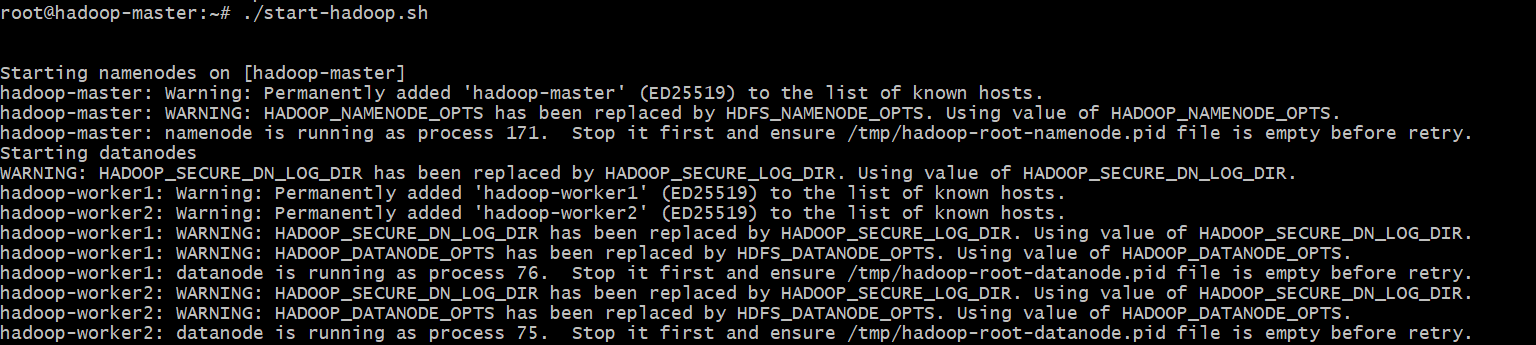
Dans ce cas pratique, nous allons entrer dans le conteneur principal (master) pour commencer à utiliser Hadoop et YARN. Pour cela, nous devons exécuter la commande suivante dans notre terminal : « **docker exec -it hadoop-master bash** »

 Une fois cette commande exécutée, nous serons connectés au conteneur, et nous verrons un prompt semblable à celui-ci :

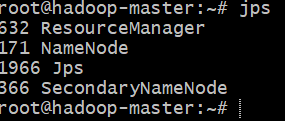
Nous nous retrouvons ainsi dans le shell du **namenode** du cluster Hadoop, ce qui nous permet de manipuler et de gérer l'ensemble de l'infrastructure Hadoop.

1. **Lancer Hadoop et YARN**

Une fois dans le conteneur, la première action que nous devons effectuer est de démarrer Hadoop et YARN. Un script d'initialisation, « **start-hadoop.sh »**, nous permet de lancer ces services de manière simple et rapide.



Ce script lancera les composants essentiels de Hadoop, y compris le **namenode** et les services **YARN** nécessaires pour la gestion des ressources du cluster. Après l'exécution du script, nous pourrons vérifier que tout fonctionne correctement en consultant l'interface web de YARN ou en analysant les logs du conteneur. Pour vérifier si les composantes sont bien lancés, nous allons utiliser la commande « **jps** » :



On voit ici que toutes les composantes sont bien lancées. Maintenant on peut manipuler des fichiers dans HDFS.

1. **Manipuler des fichiers dans HDFS : Cas Pratique**

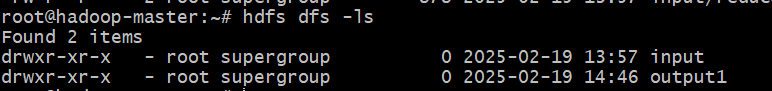
Dans ce cas pratique, nous allons interagir avec le système de fichiers HDFS en utilisant des commandes Hadoop. Nous commencerons par créer un répertoire, puis chargerons notre fichier purchages.txt pour effectuer un traitement MapReduce.

**4-1. Créer un répertoire dans HDFS**

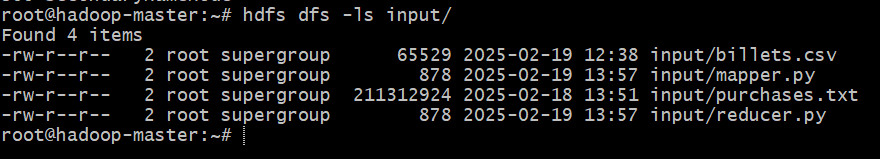
Tout d'abord, nous devons créer un répertoire appelé **input** dans HDFS ou nous allons mettre le fichier. Pour ce faire, nous utilisons la commande suivante : « hdfs dfs -mkdir -p input »

On peut vérifier que le répertoire est bien créé avec la commande : « hdfs dfs -ls/ ». Cette commande liste tous les dossiers de HDFS.

Le résultat donne ceci :



Pour mettre le fichier purchases.txt dans le dossier input, on utilise la commande suivante : « hdfs dfs -put purchases.txt input »

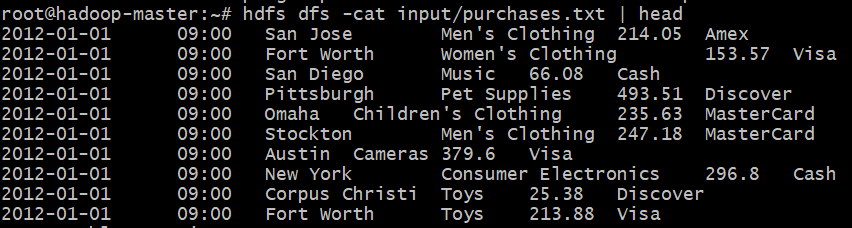


Le fichier purchases.txt est bien visible.

**4-2. Afficher les premières lignes du fichier purchases.txt**

Si nous souhaitons consulter le contenu du fichier **purchases.txt**, en particulier ses premières lignes, nous pouvons utiliser la commande suivante : « hdfs dfs -cat input/purchases.txt | head »

Le résultat donne ceci :



Une fois que notre fichier est dans HDFS, nous pouvons maintenant commencer à répondre aux questions posées un peu plus haut.

1. **Analyse des données :**

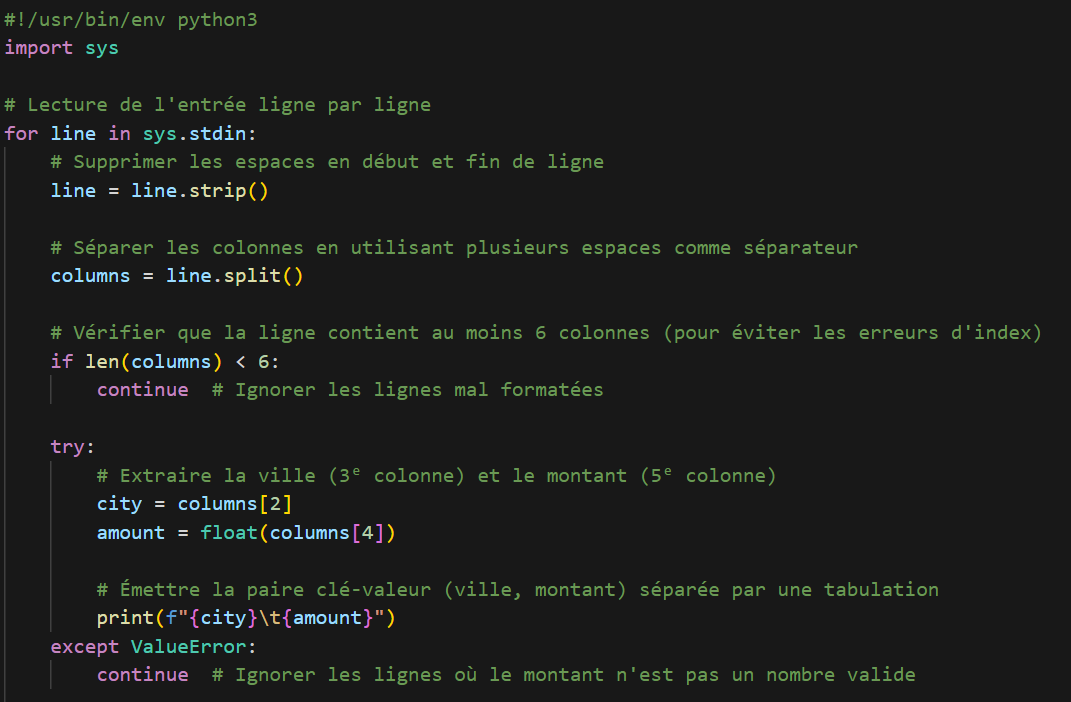
Nous allons réaliser le cas pratique pour la première question, car le procédé est identique pour les autres. Les résultats des autres questions ainsi que leurs traitements seront inclus dans les livrables finaux.

***Total des ventes par ville*** *: Combien d'argent a été dépensé dans chaque ville ?*

Pour répondre à la question "Combien d'argent a été dépensé dans chaque ville ?", nous avons utilisé Hadoop MapReduce. Le processus comprend plusieurs étapes clés : écriture du code Mapper et Reducer, exécution du job MapReduce et affichage des résultats. Voici un aperçu de ce que nous avons fait.

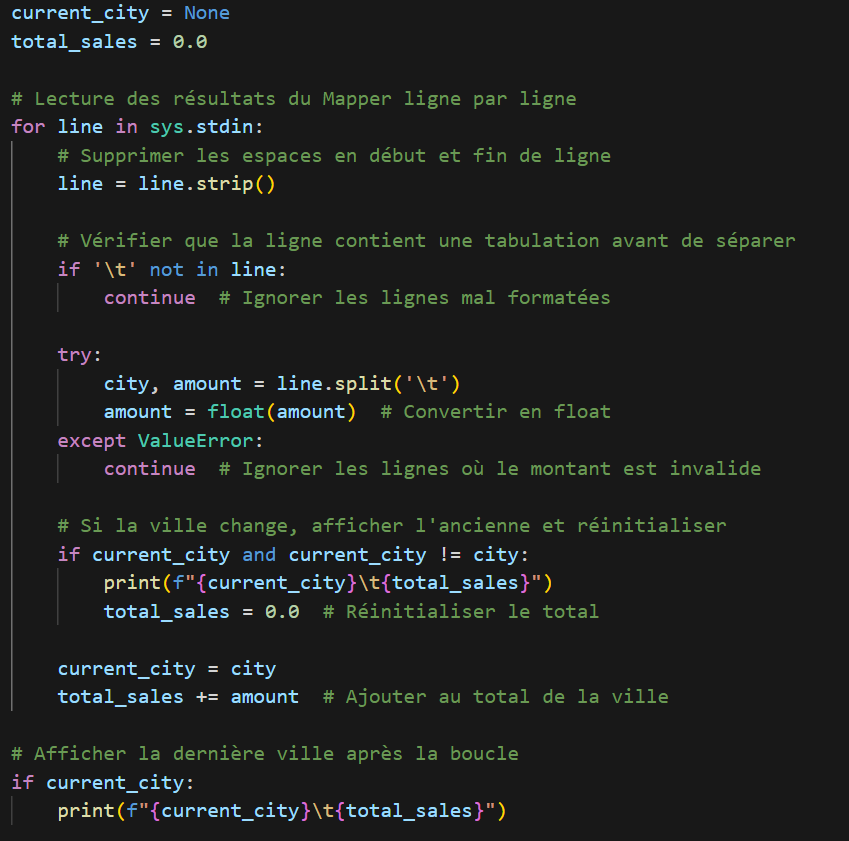
**5-1- Le code Mapper écrit en python.**

Le rôle du Mapper est de lire chaque ligne du fichier purchases.txt, d'extraire la ville et le montant, puis d'émettre une paire clé-valeur. La clé est la ville, et la valeur est le montant dépensé. Voici le code :



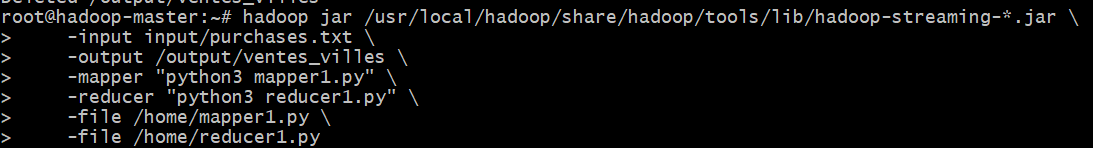
**5-2 - Le code Reducer écrit en Python**

Le rôle du Reducer est de recevoir les paires clé-valeur émises par le Mapper, de regrouper les montants par ville, puis de calculer le total des ventes pour chaque ville. Voici le code :



**5-3- Exécution du Job.**

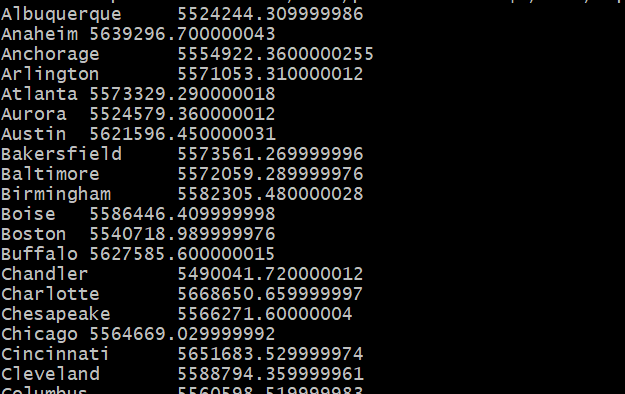
Après la rédaction des programmes mapper et reducer, nous allons lancé le job mapreduce en utilisant les codes suivants :



Ce code exécute un job Hadoop MapReduce en utilisant le package hadoop-streaming pour traiter un fichier purchases.txt avec un mapper et un reducer écrits en Python. Voici un détail de chaque partie de la commande :

* ***hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-\*.jar*** : Cette commande lance le job Hadoop en utilisant le fichier JAR hadoop-streaming, qui permet d'utiliser des scripts externes (comme Python) pour le traitement MapReduce dans Hadoop.
* ***input input/purchases.txt*** : Cette option spécifie le chemin du fichier d'entrée dans HDFS, purchases.txt, qui contient les données à traiter.
* ***output /output/ventes\_villes*** : Cette option définit le répertoire de sortie dans HDFS où les résultats seront sauvegardés. Si ce répertoire existe déjà, une erreur sera générée.
* ***mapper "python3 mapper3.py"*** : Cette option indique que le fichier mapper3.py (un script Python) sera utilisé pour la phase de map du job. Le mapper lit chaque ligne du fichier d'entrée, extrait les informations pertinentes (par exemple, la ville et le montant), et les émet sous forme de paires clé-valeur.
* ***reducer "python3 reducer3.py"*** : Cette option indique que le fichier reducer3.py (un autre script Python) sera utilisé pour la phase de reduce du job. Le reducer reçoit les paires clé-valeur émises par le mapper, les regroupe par clé (ville) et agrège les valeurs associées (par exemple, en faisant la somme des montants des ventes pour chaque ville).
* ***file /home/mapper3.py et -file /home/reducer3.py*** : Ces options spécifient les fichiers Python mapper3.py et reducer3.py, qui sont envoyés au cluster Hadoop pour être utilisés dans les phases mapper et reducer respectivement. Cela garantit que ces scripts Python sont accessibles pendant l'exécution du job.

**5-4- Résultats :**

Après l’exécution du Job, voici un aperçu des résultats obtenus :

Ces résultats fournissent, pour chaque ville, le montant total des ventes. Nous pouvons exporter ces données et effectuer des analyses plus approfondies avec Python, par exemple. En utilisant Hadoop, nous avons pu traiter efficacement une base de données contenant plusieurs millions de lignes, en la réduisant à un ensemble beaucoup plus gérable, composé de moins de 30 lignes, correspondant à une ligne par ville. Grâce à la scalabilité et au traitement parallèle de Hadoop, nous avons pu gérer de grandes quantités de données de manière distribuée et rapide, ce qui rend l'analyse de données massives plus accessible et efficace.