Voici des questions que vous pouvez utiliser pour travailler plus en profondeur la programmation en map/reduce.

Exercice 0 - WordCount

Tester le programme WordCount.

Exercice 1 - WordCount + Filter

Modifier la fonction reduce du programme WordCount.java pour afficher uniquement les mots ayant un nombre d'occurrences supérieur ou égal à deux.

Exercice 2 - Group-By

Implémenter un opérateur de regroupement sur l'attribut `Customer-ID` dans GroupBy.java.
Les données sont dans `input-groupBy` et doivent calculer le total des profits (`Profit`) par client.

Exercice 3 - Group-By

Modifier le programme précédent :

1. Calculer les ventes par `Date` et `State`.
2. Calculer les ventes par `Date` et `Category`.
3. Calculer par commande :
    - Le nombre de produits distincts achetés.
    - Le nombre total d'exemplaires.

Exercice 4 - Join

Créer une classe Join.java pour joindre les informations des clients et commandes dans `input-join`.
Restituer les couples `(CUSTOMERS.name, ORDERS.comment)`.

Note : Copier les valeurs de l'itérateur dans un tableau temporaire et utiliser deux boucles imbriquées pour effectuer la jointure.

Exercice 5 - GroupBy + Join

Pour le fichier `superstore.csv`, calculer le montant total des achats faits par chaque client.
Le programme doit restituer des couples `(CUSTOMERS.name, SUM(totalprice))`.

Exercice 6 - Suppression des doublons (DISTINCT)

Donner la liste des clients (sans doublons) présents dans le dataset du répertoire `input-groupBy`.

Exercice 7 - MR <-> SQL

Donner le code SQL équivalent aux traitements Map/Reduce implémentés pour les questions 4, 5, 6 et 7.

Exercice 8 - TAM

Rendez-vous à l'adresse : offre-de-transport-tam-en-temps-reel (http://data.montpellier3m.fr/dataset/offre-de-transport-tam-en-temps-reel)
Télécharger le fichier `TAM_MMM_OffreJour.zip` contenant la prévision du service de tramway pour la journée.

Répondre aux questions suivantes :

- Donner un aperçu des trams et bus de la station `OCCITANIE`. Préciser le nombre de (bus ou trams) pour chaque heure et ligne.
  Exemple : `<Ligne 1, 17h, 30>` (à 17h, 30 trams de la ligne 1).
- Pour chaque station, donner le nombre de trams et bus par jour.
- Pour chaque station et chaque heure, afficher une information `X_tram` correspondant au trafic des trams avec des niveaux "faible", "moyen", ou "fort". Faire de même pour les bus.

Exercice 9 - Tri

Hadoop trie les clés des groupes en ordre lexicographique ascendant pendant la phase de shuffling. Modifier la méthode de tri.

1. Trier les commandes clients du fichier `superstore.csv` par date d'expédition en ordre croissant, puis décroissant.
2. Trier les clients (identifiant + nom) par profit généré.

Exercice 10 - Requêtes Top-k

Modifier la classe `TopkWordCount.java` pour répondre aux requêtes suivantes :

1. Les k premières lignes triées par profit (ordre décroissant).
2. Les k premiers clients en termes de profit réalisé (ordre décroissant).

Exercice 11 - TAM (suite question 9)

Répondre aux questions suivantes :

- Quelles sont les 10 stations les plus desservies par les trams ?
- Quelles sont les 10 stations les plus desservies par les bus ?
- Quelles sont les 10 stations les plus desservies (trams et bus) ?

Exercice 12 - Taxis New York

Rendez-vous à l'adresse : trip_record_data (http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml)

Télécharger les données des taxis jaunes pour janvier 2018. Répondre aux questions suivantes :

1. Quelles sont les heures de pointe ?
2. Quels sont les trois jours les plus chargés du mois ?
3. Quel est le moyen de paiement le plus utilisé par jour ?
4. Géolocalisation des zones les plus demandées (départ et arrivée).
5. La plupart des courses se font-elles avec deux passagers ?
6. Pourcentage du pourboire par rapport au prix des courses ?
7. Longueur moyenne des trajets ?
8. Prix moyen / médian d'une course ?
9. Nombre de disputes (payment-type=4) par jour ?

Exercice 13 - Jointure par hachage

Il s'agit d'une technique classique de jointure, qui dévient toutefois particulièrement intéressante en map/reduce lorsqu'on se retrouve avec plusieurs relations à joindre, comme dans le cas des requêtes dites "en étoile" typique des entrepôts de données.

```
DimClient(idClient, ... ) JOIN FaitVente( idClient, idProduit, ... ) JOIN DimProduit
                                    (idProduit, ...)
```

L'idée est de découper l'évaluation de la jointure dans un certain nombre de sous-tâches executables en parallèle.

```
|  tâche1    tâche2    tâche3 |

|  tâche4    tâche5    tâche6 |

|  tâche7    tâche8    tâche9 |

|  tâche10  tâche11   tâche12 |

|  tâche13  tâche14   tâche15 |
```

Chaque sous-tâche calcule une partie du résultat de la jointure. Les résultats partiels seront efin réunis.

Les deux questions importantes sont les suivantes :

- comment choisir le nombre de sous-tâches ?
- comment distribuer les données dans les sous-tâches ? (on verra que la redondance est bienvenue)

#1 (conte-intuitif, mais) l'ingrédient de départ est un partitionnement des valeurs des colonnes impliquées dans la jointure (si besoin, relisez plusieurs fois cette phrase)

Par exemple, les valeurs de la colonne `idClient` sont partitionnés en 5 ensembles, lorsque les valeurs dans la colonne `idProduit` en 3 ensembles.

```
colonne idClient       colonne idProduit

clients_partition_1      produits_partition_1

clients_partition_2      produits_partition_2

clients_partition_3      produits_partition_3

clients_partition_4

clients_partition_5
```

Une fonction auxiliaire de hachage (ou coloration) `hA` associe chaque valeur de la colonne `A` à sa partition (dont le nom de cette technique).

Par exemple, `hidProduit(P132H67) = 2`.

Lire : l'identifiant de produit `P132H67` de la colonne `idProduit` est associé à la partition 2.

#2 C'est presque fini. En effet, le partitionnement nous donne un tableau définissant les sous-tâches de la jointure.

```
|  clients_p_1,produits_p_1   clients_p_1,produits_p_2   clients_p_1,produits_p_3  |

|  clients_p_2,produits_p_1   clients_p_2,produits_p_2   clients_p_2,produits_p_3  |

|  clients_p_3,produits_p_1   clients_p_3,produits_p_2   clients_p_3,produits_p_3  |

|  clients_p_4,produits_p_1   clients_p_4,produits_p_2   clients_p_4,produits_p_3  |

|  clients_p_5,produits_p_1   clients_p_5,produits_p_2   clients_p_5,produits_p_3  |
```

Chaque sous-tâche travaille sur des combinaisons de valeurs `(v1,v2,...,vk)` pour l'ensemble des colonnes `(A1,A2,...Ak)` impliquées dans la jointure.

Ici le tableau est à 2 dimensions car on considère 2 attributs de jointures.

La taille du tableau correspond au nombre total de sous–tâches (clés en map reduce).

#3 Voyons maintenant les fonctions map et reduce.

- **Reduce** (la plus simple des deux) : reçoit une sous partie des tables `DimClient`, `FaitVente`, et `DimProduit` et calcule la jointure.

- **Map** : doit se comporter différemment en fonction de chaque table.

Reprenons notre exemple et faisons une analyse des cas.

`DimClient(idClient, ... ) JOIN FaitVente( idClient, idProduit, ... ) JOIN DimProduit (idProduit, ...)`

- Une ligne de la table `FaitVente (idC,idP,...)` est associée à la sous-tâche `<hidClient(idC),hidProduit(idP)>`

par exemple la ligne `(C1123,P132H67, ...)` est associée à la sous tâche `<4,2>`

```
|  clients_p_1,produits_p_1   clients_p_1,produits_p_2   clients_p_1,produits_p_3  |

|  clients_p_2,produits_p_1   clients_p_2,produits_p_2   clients_p_2,produits_p_3  |

|  clients_p_3,produits_p_1   clients_p_3,produits_p_2   clients_p_3,produits_p_3  |

|  clients_p_4,produits_p_1   clients_p_4,produits_p_2   clients_p_4,produits_p_3  |

|  clients_p_5,produits_p_1   clients_p_5,produits_p_2   clients_p_5,produits_p_3  |
```

- Une ligne de la table `DimClient (idC,...)` est par contre associée aux sous-tâches

```
<hidClient(idC),1>

<hidClient(idC),2>

<hidClient(idC),3>
```

car la colonne `idProduit` n'apparait pas dans la table `DimClient`

Par exemple la ligne `(C1123, ...)` est associée aux sous tâches `<4,1> <4,2> <4,3>`

```
|  clients_p_1,produits_p_1   clients_p_1,produits_p_2   clients_p_1,produits_p_3  |

|  clients_p_2,produits_p_1   clients_p_2,produits_p_2   clients_p_2,produits_p_3  |

|  clients_p_3,produits_p_1   clients_p_3,produits_p_2   clients_p_3,produits_p_3  |

|  clients_p_4,produits_p_1   clients_p_4,produits_p_2   clients_p_4,produits_p_3  |

|  clients_p_5,produits_p_1   clients_p_5,produits_p_2   clients_p_5,produits_p_3  |
```

- De même, une ligne de la table `DimProduit (idP,...)` est associée aux sous tâches

```
<hidProduit(idP),1>

<hidProduit(idP),2>

<hidProduit(idP),3>

<hidProduit(idP),4>

<hidProduit(idP),5>
```

car la colonne `idClient` n'apparait pas dans la table `DimProduit`

Par exemple la ligne `(P132H67, ...)` est associée aux sous tâches `<1,2> <2,2> <3,2> <4,2> <5,2>`

```
|  clients_p_1,produits_p_1   clients_p_1,produits_p_2   clients_p_1,produits_p_3  |

|  clients_p_2,produits_p_1   clients_p_2,produits_p_2   clients_p_2,produits_p_3  |

|  clients_p_3,produits_p_1   clients_p_3,produits_p_2   clients_p_3,produits_p_3  |

|  clients_p_4,produits_p_1   clients_p_4,produits_p_2   clients_p_4,produits_p_3  |

|  clients_p_5,produits_p_1   clients_p_5,produits_p_2   clients_p_5,produits_p_3  |
```

Très important : ces valeurs sont à choisir en amont de l'évaluation de la requête. On considérera qu'il s'agit d'une entrée pour le programme map/reduce, comme l'on a fait pour la valeur `k` dans le `TopK`.