مكتب التكوين المهني و إنعاش الشغل
Office de la formation professionnelle et de la promotion du travail

*Direction Régionale : Casablanca – Settat* *Cité des métiers et des compétences Nouaceur*

**Contrôle continu N° 1 : Algorithmes distribués (M204)**
**Année 2025/2026**

**Filière :** Intelligence Artificielle
**Niveau :** TS
**Groupe :** DIA_IAOBD_TS_201
**Nom et Prénom :**.....

**Durée :** 2H
**Variante :** A
**Barème :** .../20

# Partie Pratique : (20 points)

**Données doit etre disponibles dans HDFS sur le dossier :** `input_Nom_Stagiaire/movielens`
`movies.dat`, `ratings.dat`, `tags.dat`

## Structure du Dataset

— **movies.dat** : `MovieID::Title::Genres`

```
1::Toy Story (1995)::Animation|Children's|Comedy

```

— **ratings.dat** : `UserID::MovieID::Rating::Timestamp`

```
1::122::5.0::838985046

```

— **tags.dat** : `UserID::MovieID::Tag::Timestamp`

```
15::4973::excellent!::1215184630

```

# 1 Exercice 1 : MapReduce avec Hadoop (8 points)

Développer des programmes MapReduce en Python pour analyser le dataset MovieLens. Tous les scripts doivent être exécutés via Hadoop Streaming.

### Q1.1 - Comptage des films par année (1 point)

Compter le nombre de films sortis chaque année.

a) Écrire `mapper_year.py` qui émet `(année, 1)`
b) Écrire `reducer_year.py` qui agrège les résultats
c) Exécuter avec Hadoop Streaming

**Sortie attendue :** Liste `année` `nombre_films`

```
#!/usr/bin/env python
import sys
import re
for line in sys.stdin:
    line = line.strip()
    fields = line.split('::')

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------
```

Code à compléter - mapper_year.py :

```
#!/usr/bin/env python
import sys

current_year = None
count = 0
for line in sys.stdin:
    line = line.strip()
    year, value = line.split('\t')

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------
```

Code à compléter - reducer_year.py :

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
    -input /user/cloudera/movielens/movies.dat \
    -output ______ \
    -mapper ______ \
    -reducer ______ \
    -file mapper_year.py \
    -file reducer_year.py
```

Commande d'exécution :

### Q1.2 - Nombre d'évaluations par utilisateur (1,5 points)

Calculer le nombre d'évaluations faites par chaque utilisateur.

a) Écrire `mapper_user_ratings.py` qui lit `ratings.dat`

b) Émettre `(UserID, 1)` pour chaque évaluation

c) Écrire `reducer_user_ratings.py` qui fait la somme

d) Trier les résultats par nombre décroissant

**Sortie attendue :** Top 10 des utilisateurs les plus actifs

```
#!/usr/bin/env python
import sys

for line in sys.stdin:
    line = line.strip()
    fields = line.split('::')
     ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------
```

Code à compléter - mapper_user_ratings.py :

### Q1.3 - Note moyenne par film (2 points)

Calculer la note moyenne de chaque film.

a) Écrire `mapper_movie_avg.py` qui émet `(MovieID, (rating, 1))`

b) Écrire `reducer_movie_avg.py` qui calcule la moyenne

c) Filtrer les films avec au moins 100 évaluations

**Indice :** Utiliser un format de sortie `MovieID  note_moyenne  nombre_notes`

```
#!/usr/bin/env python
import sys

for line in sys.stdin:
    line = line.strip()
    fields = line.split('::')

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------

    ------------------------------------------------------------------------
```

Code à compléter - mapper_movie_avg.py :

### Q1.4 - Films par genre (1,5 points)

Compter le nombre de films pour chaque genre.

a) Écrire `mapper_genre.py` qui éclate les genres multiples

b) Pour un film avec genres "Action|Comedy|Drama", émettre 3 paires

c) Écrire `reducer_genre.py` qui agrège les résultats

**Sortie attendue :** Liste des genres avec leur nombre de films

```python
#!/usr/bin/env python
import sys

for line in sys.stdin:
    line = line.strip()
    fields = line.split('::')

    # ----------------------------------------------------------------

    # ----------------------------------------------------------------

    # ----------------------------------------------------------------

    # ----------------------------------------------------------------

    # ----------------------------------------------------------------

    # ----------------------------------------------------------------

    # ----------------------------------------------------------------
```

Code à compléter - mapper_genre.py :

### Q1.5 - Top 10 des meilleurs films (2 points)

Trouver les 10 films avec la meilleure note moyenne (min 500 évaluations).

a) Utiliser le résultat de Q1.3

b) Écrire un second MapReduce qui trie les films par note moyenne

c) Filtrer les films avec plus de 500 évaluations

d) Produire un top 10

**Sortie attendue :** `MovieID  Titre  Note_moyenne  Nombre_notes`

```python
#!/usr/bin/env python
import sys

for line in sys.stdin:
    line = line.strip()
    # Format: MovieID   avg_rating   count
    movie_id, avg_rating, count = line.split('\t')
    # ----------------------------------------------------------------

    # ----------------------------------------------------------------

    # ----------------------------------------------------------------

    # ----------------------------------------------------------------

    # ----------------------------------------------------------------
```

Code à compléter - mapper_top10.py :

# 2 Exercice 2 : Apache Pig (6 points)

Vous devez écrire des scripts Pig Latin pour analyser les données. Tous les résultats doivent être stockés dans HDFS.

### Q2.1 - Chargement et inspection (1 point)

Charger les trois fichiers et afficher leur schéma.
Écrire un script Pig qui :

a) Charge `movies.dat`, `ratings.dat`, `tags.dat` avec le bon séparateur

b) Affiche le schéma de chaque relation

c) Affiche 5 tuples de chaque

```
-- Chargement des films
    ------------------------------------------------------------------

    ------------------------------------------------------------------

    ------------------------------------------------------------------

    ------------------------------------------------------------------

-- Chargement des notes
    ------------------------------------------------------------------

    ------------------------------------------------------------------

    ------------------------------------------------------------------

    ------------------------------------------------------------------
-- Chargement des tags
    ------------------------------------------------------------------

    ------------------------------------------------------------------

    ------------------------------------------------------------------

    ------------------------------------------------------------------

-- Afficher le schéma (DESCRIBE, DUMP, ...)
    ------------------------------------------------------------------

    ------------------------------------------------------------------

    ------------------------------------------------------------------

    ------------------------------------------------------------------
```

Script Pig à compléter :

### Q2.2 - Films les plus populaires (1 point)

Trouver les 20 films avec le plus d'évaluations.
Écrire un script Pig qui :

a) Joint `movies` et `ratings` par `MovieID`

b) Calcule le nombre d'évaluations par film

c) Trie par nombre décroissant

d) Stocke le résultat dans HDFS

```
-- Jointure films-notes
movies_ratings = ____________________________________________________________;


-- Regrouper par film
____________________________________________________________;

-- Calculer le nombre d'évaluations
movies_count = FOREACH movies_group GENERATE
    group AS movieId,
    COUNT(movies_ratings) AS rating_count;

-- Trier et limiter (ORDER movies_count BY rating_count DESC) 20
____________________________________________________________

____________________________________________________________;

-- Stocker dans HDFS
STORE top_movies INTO '______'_____________________________________________

_____________________________________;
```

Script Pig à compléter :

### Q2.3 - Analyse des tags (2 points)

Analyser la fréquence des tags.
Écrire un script Pig qui :

a) Charge `tags.dat`
b) Convertit tous les tags en minuscules
c) Compte la fréquence de chaque tag
d) Stocke les 50 tags les plus fréquents dans HDFS

```
-- Chargement des tags
tags = LOAD '_____________________________________________________________;'

    USING PigStorage('::')

    AS (userId:int, movieId:int, tag:chararray, timestamp:chararray);

-- Convertir en minuscules et compter (tags_lower)

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________

____________________________________________________________;

-- Top 50 (LIMIT (ORDER tags_count BY frequency DESC) 50)
____________________________________________________________

____________________________________________________________;

STORE ____________________________________________________________;
```

Script Pig à compléter :

### Q2.4 - Utilisateurs les plus actifs par année (2 points)

Identifier les utilisateurs les plus actifs chaque année.
Écrire un script Pig qui :

a) Extrait l'année du timestamp des évaluations

b) Compte le nombre d'évaluations par utilisateur et par année

c) Pour chaque année, garde le top 3 des utilisateurs

d) Stocke le résultat dans HDFS

```
-- TODO Extraire l'année du timestamp
ratings_with_year = FOREACH ratings GENERATE
----------------------------------------------------------------

----------------------------------------------------------------------

----------------------------------------------------------------

----------------------------------------------------------------;

-- TODO Compter par utilisateur et année
ratings_group = GROUP ratings_with_year BY (userId, year);
user_year_stats = FOREACH ratings_group GENERATE
----------------------------------------------------------------

----------------------------------------------------------------------;

-- Trier et prendre le top 3 par année
user_year_sorted = ORDER user_year_stats BY year, rating_count DESC;
-- TODO: Implémenter le top 3 par année
----------------------------------------------------------------

----------------------------------------------------------------------

----------------------------------------------------------------------

----------------------------------------------------------------

----------------------------------------------------------------

-- TODO STORE result INTO '_____________________________' USING PigStorage(',');

----------------------------------------------------------------------------
```

Script Pig à compléter :

# 3 Exercice 3 : Apache Hive (6 points)

Vous devez créer des tables Hive et exécuter des requêtes analytiques.

### Q3.1 - Création des tables externes (1 point)

Créer les tables Hive externes pointant vers les données HDFS.
Écrire les requêtes HiveQL pour :

a) Créer la table `movies_hive` avec séparateur `::`

b) Créer la table `ratings_hive` avec séparateur `::`

c) Créer la table `tags_hive` avec séparateur `::`

d) Vérifier le nombre d'enregistrements dans chaque table

```
-- Table movies_hive
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
-- Table ratings_hive
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
-- Table tags_hive
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
-- Vérification
```

Requête HiveQL à compléter :

### Q3.2 - Analyse temporelle (1 point)

Analyser l'évolution des évaluations dans le temps.
Écrire une requête HiveQL qui :

a) Calcule le nombre d'évaluations par mois (année-mois)

b) Calcule la note moyenne par mois

c) Affiche les résultats triés par période

d) Sauvegarde le résultat dans une table Hive `ratings_monthly_stats`

```
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
```

Requête HiveQL à compléter :

### Q3.3 - Segmentation des utilisateurs (1 point)

Segmenter les utilisateurs selon leur comportement.
Écrire une requête HiveQL qui :

a) Classe les utilisateurs en 4 catégories

b) Pour chaque segment, calcule les statistiques

c) Sauvegarde dans une table Hive `user_segments`

```
CREATE TABLE user_segments AS
WITH user_stats AS (
    SELECT
        userId,
        COUNT(*) AS rating_count,
        ROUND(AVG(rating), 3) AS avg_rating
    FROM ratings_hive
    GROUP BY userId
)
SELECT
    CASE
        WHEN rating_count >= 500 THEN 'Power Users'
        WHEN rating_count >= 100 THEN 'Active Users'
        WHEN rating_count >= 10 THEN 'Regular Users'
        ELSE 'Casual Users'
    END AS user_segment,
    COUNT(*) AS user_count,
    ROUND(AVG(rating_count), 1) AS avg_ratings_per_user,
    ROUND(AVG(avg_rating), 3) AS segment_avg_rating
FROM user_stats

GROUP BY ______________________

ORDER BY ______________________ DESC;

-------------------------------------------------------------------------

-------------------------------------------------------------------------
```

Requête HiveQL à compléter:

### Q3.4 - Analyse des genres (1 point)

Analyser la performance des différents genres.
Écrire une requête HiveQL qui :

a) Éclate les genres multiples

b) Calcule pour chaque genre les statistiques

c) Filtre les genres avec au moins 100 films

d) Sauvegarde dans une table Hive `genre_performance`

```
CREATE TABLE genre_performance AS
SELECT
    genre,
    COUNT(DISTINCT m.movieId) AS film_count,
    ROUND(AVG(r.rating), 3) AS avg_rating,
    COUNT(r.rating) AS total_ratings

-------------------------------------------------------------------------

-------------------------------------------------------------------------
```

```
11
12 ----------------------------------------------------------------------
13
14 ----------------------------------------------------------------------
15
16 ----------------------------------------------------------------------
17
18 ----------------------------------------------------------------------
19
20 ----------------------------------------------------------------------
```

Requête HiveQL à compléter :

### Q3.5 - Recommandation avancée (2 points)

Implémenter un système de recommandation collaborative simple.
Écrire une requête HiveQL qui pour un film donné (ex : MovieID=1) :

a) Trouve les utilisateurs qui ont aimé ce film

b) Trouve les autres films que ces utilisateurs ont aussi aimés

c) Calcule un score de recommandation

d) Recommande les 10 films avec le score le plus élevé

```
1  ----------------------------------------------------------------------
2
3  ----------------------------------------------------------------------
4
5  ----------------------------------------------------------------------
6
7  ----------------------------------------------------------------------
8
9  ----------------------------------------------------------------------
10
11 ----------------------------------------------------------------------
12
13 ----------------------------------------------------------------------
14
15 ----------------------------------------------------------------------
16
17 ----------------------------------------------------------------------
18
19 ----------------------------------------------------------------------
20
21 ----------------------------------------------------------------------
22
23 ----------------------------------------------------------------------
24
25 ----------------------------------------------------------------------
```

Requête HiveQL à compléter :