오픈소스SW개론

- Project1 –

**분반 : CSE2113-001**

**학번 : 12193266**

**학과 : 중국학과**

**이름 : 박민정**

**[ 목차 ]**

1. **프로젝트 개요**
2. **구현 코드**
3. **각 기능 구현에 대한 상세 설명**

0) 기본적인 틀, 정의

1) Get the data of the movie identified by a specific 'movie id' from 'u.item'

2) Get the data of ‘action’ genre movies from 'u.item’

3) Get the average 'rating’ of the movie identified by specific 'movie id' from 'u.data’

4) Delete the ‘IMDb URL’ from ‘u.item’

5) Get the data about users from 'u.user’

6) Modify the format of 'release date' in 'u.item’

7) Get the data of movies rated by a specific 'user id' from 'u.data'

8) Get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as 'programmer'

9) Exit

1. **결과 출력**
2. **GitHub link**

**[ 프로젝트 개요 : 영화 데이터 분석 프로그램]**

**• 목표 :** 주어진 영화 데이터 파일들을 읽고, 특정 조건에 따라 정보를 출력하는 Bash 스크립트 프로그램을 작성한다.

**• 프로그래밍 언어 및 도구 :** Bash스크립트 사용

**• 입력 파일** : u.item, u.data, u.user

- u.item : movie id | movie title | release date | video release date |IMDb URL | unknown | Action | Adventure | Animation | Children's | Comedy | Crime | Documentary | Drama | Fantasy | Film-Noir | Horror | Musical | Mystery | Romance | Sci-Fi | Thriller | War | Western|

(1682 rows)

- u.data : user id | movie id | rating | timestamp (100000 rows)

- u.user : user id | age | gender | occupation | zip code (943 rows)

**• 과제 요구 사항** : 프로그램 실행 후

- 1 입력 시 : 'u.item'에서 특정 '영화 ID'로 식별된 영화 데이터 가져오기

- 2 입력 시 : 'u.item'에서 '액션' 장르 영화 데이터 가져오기

- 3 입력 시 : 'u.data'에서 특정 '영화 ID'로 식별된 영화의 평균 '평점' 가져오기

- 4 입력 시 : 'u.item'에서 'IMDb URL' 삭제하기

- 5 입력 시 : 'u.user'에서 사용자 데이터 가져오기

- 6 입력 시 : 'u.item'에서 ‘release date' 형식 수정하기

- 7 입력 시 : 'u.data'에서 특정 '사용자 ID'로 평가된 영화 데이터 가져오기

- 8 입력 시 : '나이'가 20에서 29세이고 '직업'이 '프로그래머'인 사용자가 평가한 영화의 평균 '평점' 가져오기

- 9 입력 시 : 종료 ( 9 입력 전까지는 프로그램이 계속 실행될 수 있도록 작성)

**[ 구현 코드 ]**

#!/bin/bash

item\_file=$1

data\_file=$2

user\_file=$3

echo "-------------------------------------------------------"

echo "User Name: Park Min Jung"

echo "Student Number: 12193266"

echo "[ Menu ]"

echo "1. Get the data of the movie identified by a specific 'movie id' from 'u.item'"

echo "2. Get the data of action genre movies from 'u.item'"

echo "3. Get the average 'rating' of the movie identified by specific 'movie id' from 'u.data'"

echo "4. Delete the 'IMDb URL' from 'u.item'"

echo "5. Get the data about users from 'u.user'"

echo "6. Modify the format of 'release date' in 'u.item'"

echo "7. Get the data of movies rated by a specific 'user id' from 'u.data'"

echo "8. Get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as 'programmer'"

echo "9. Exit"

echo "-------------------------------------------------------"

stop="N"

until [ $stop = "Y" ]

do

read -p "Enter your choice [ 1-9 ] " choice

case $choice in

1)

echo ""

read -p "Please enter 'movie id' (1~1682) : " movie\_id

echo ""

cat $item\_file | awk -F\| -v a=$movie\_id 'a==$1{print } '

echo "" ;;

2)

echo ""

read -p "Do you want to get the data of 'action' genre from 'u.item'? (y/n) :" YN

echo ""

if [ $YN = "y" ]

then

cat $item\_file | awk -F\| '$7=="1"{print $1,$2}' | sort -t\| -k1,1n | head -n 10

fi

echo""

;;

3)

echo ""

read -p "Please enter the 'movie id' (1~1682) : " movie\_id

echo ""

cat $data\_file | awk -v a=$movie\_id 'a==$2 {sum+=$3; count+=1 } END {if (count >0) average\_rating=sum/count} END {if (count > 0) printf ("average rating of %d : %.5f \n",a,average\_rating) } '

echo ""

;;

4)

echo ""

read -p "Do you want to delete the 'IMDb URL' from 'u.item'? (y/n) :" YN

echo ""

if [ $YN = "y" ]

then

cat $item\_file | sed 's/||[^|]\*|/|||/g' | head -n 10

fi

echo ""

;;

5)

echo ""

read -p "Do you want to get the data about users from 'u.user'? (y/n) :" YN

echo ""

if [ $YN = "y" ]

then

cat $user\_file | awk -F\| '{if ($3=="F") {gender="female"}else {gender="male"}} {printf ("user %d is %d years old %s %s \n",$1,$2,gender,$4) }' | head -n 10

fi

echo ""

;;

6)

echo ""

read -p "Do you want to Modify the format of 'release data' in 'u.item'? (y/n) " YN

echo ""

if [ $YN = "y" ]

then

cat $item\_file | sed 's/\([0-9]\{2\}\)-\(Jan\)-\([0-9]\{4\}\)/\301\1/g' | tail -n 10 | sed 's/\([0-9]\{2\}\)-\(Feb\)-\([0-9]\{4\}\)/\302\1/g' | sed 's/\([0-9]\{2\}\)-\(Mar\)-\([0-9]\{4\}\)/\303\1/g' | sed 's/\([0-9]\{2\}\)-\(Apr\)-\([0-9]\{4\}\)/\304\1/g' | sed 's/\([0-9]\{2\}\)-\(May\)-\([0-9]\{4\}\)/\305\1/g' | sed 's/\([0-9]\{2\}\)-\(Jun\)-\([0-9]\{4\}\)/\306\1/g' | sed 's/\([0-9]\{2\}\)-\(Jul\)-\([0-9]\{4\}\)/\307\1/g' | sed 's/\([0-9]\{2\}\)-\(Aug\)-\([0-9]\{4\}\)/\308\1/g' | sed 's/\([0-9]\{2\}\)-\(Sep\)-\([0-9]\{4\}\)/\309\1/g' | sed 's/\([0-9]\{2\}\)-\(Oct\)-\([0-9]\{4\}\)/\310\1/g' | sed 's/\([0-9]\{2\}\)-\(Nov\)-\([0-9]\{4\}\)/\311\1/g' | sed 's/\([0-9]\{2\}\)-\(Dec\)-\([0-9]\{4\}\)/\312\1/g' > temp.txt

cat temp.txt

rm temp.txt

fi

echo ""

7)

echo ""

read -p "Please enter the 'user id' (1~943) :" user\_id

echo ""

cat $data\_file | sort -k 2,2n | awk -v user=$user\_id '$1==user {printf ("%d|",$2)}'

echo ""

echo ""

awk -v user=$user\_id '$1==user {print $2}' $data\_file| sort -k 1,1n > user\_rated\_movies.txt

awk -F\| 'FNR==NR {a[$1]++; next} $1 in a {print $1"|"$2}' user\_rated\_movies.txt $item\_file | head -n 10

echo ""

rm user\_rated\_movies.txt

;;

8)

echo ""

read -p "Do you want to get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as 'programmer'? (y/n) :" YN

echo ""

if [ $YN = "y" ]

then

awk -F\| '$2>=20 && $2<=29 && $4=="programmer" {print $1}' $user\_file > programmers\_20s.txt

awk 'FNR==NR {programmers[$1]++; next} $1 in programmers {sum[$2]+=$3; count[$2]++;} END {for (movie\_id in sum) {avg=sum[movie\_id]/count[movie\_id]; printf("%d %.5f\n", movie\_id, avg)}}' programmers\_20s.txt $data\_file| sort -k 1,1n > average\_ratings.txt

cat average\_ratings.txt

echo ""

rm average\_ratings.txt

rm programmers\_20s.txt

fi

echo ""

;;

9)

echo "Bye!"

stop="Y"

;;

esac

done

**[ 각 기능 구현에 대한 상세 설명 ]**

**0) 기본적인 틀, 변수 정의**

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* 각 케이스에 대한 작업 수행 코드 -



- 입력이 u.item, u.data, u.user순으로 들어옵니다. $1, $2, $3는 스크립트를 실행할 때 전달된 인자입니다. 나중에 이 파일들을 편리하게 사용하기 위하여 위의 코드에서 tem\_file, data\_file, user\_file에 각각 'u.item', 'u.data', 'u.user' 파일의 경로를 저장하였습니다.

- echo를 통해 간단한 사용자 인터페이스를 구현하였습니다. 화면에 구분선 및 사용자 정부를 출력하였으며, 다양한 메뉴 옵션들을 열거하고, 각 옵션에 대한 설명을 제공하였습니다. 사용자는 각 메뉴 옵션 입력을 통해 작업을 수행할 수 있습니다.

- stop이라는 변수를 통해 루프를 통제할 수 있게 하였습니다. 그렇기 때문에 맨 처음 시작에서 stop을 “N”으로 초기화하였습니다. ( until [ $stop = "Y" ] , stop 변수가 "Y"가 될 때까지 반복합니다.)

- read -p "Enter your choice [ 1-9 ] " choice 부분에서 사용자에게 1에서 9까지의 숫자 중 하나를 입력하라는 메시지를 출력하고, 입력값을 choice 변수에 저장합니다. 이후, choice 변수 값을 기반으로 각각의 경우(case)를 검사하고, 각 경우에 대한 작업을 수행합니다.

**Case 1) Get the data of the movie identified by a specific 'movie id' from 'u.item'**

▪ Please enter the 'movie id’(1~1682)

▪ Print the data of the movie identified by the input 'movie id’.

▪ The data can be found in ‘u.item'

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

- 기능 : 특정 ‘영화ID’를 입력하면 해당 ‘영화 ID’에 대한 정보를 u.item에서 찾아 해당 라인을 출력합니다.

- read -p "Please enter 'movie id' (1~1682) : " movie\_id 부분을 통해 과제에서 요구한 문장 출력 후, 사용자로부터 ‘영화 ID’를 입력 받습니다. read 명령어를 통해 입력 받을 수 있게 구현하였고, -p옵션을 통해 사용자에게 특정 메시지 출력하게 구현하였습니다.

- cat $item\_file | awk -F\| -v a=$movie\_id 'a==$1{print }' 을 통해 'u.item' 파일을 읽어와서 조건에 따라 출력하였습니다. '|' (파이프)를 통해 구분된 각 라인을 처리하고 이를 다음으로 넘겨주었으며, awk 명령어의 -F\| 옵션을 통해 '|'를 필드 구분자로 사용하였습니다. -v옵션을 통해 Awk 변수 a에 Bash 변수 movie\_id의 값을 넘겨줍니다.

Awk에서 $1은 라인의 첫 번째 필드, 즉 '영화 ID'를 나타냅니다. 따라서 a==$1은 Awk에서 '영화 ID'를 Bash 변수 movie\_id와 비교합니다. 조건에 맞는 라인은 print 명령어를 사용하여 출력됩니다.

**Case 2) Get the data of ‘action’ genre movies from 'u.item’**

▪ Do you want to get the data of ‘action’ genre movies from 'u.item’?(y/n)

▪ Prints 10 data of action genre movies from 'u.item’

▪ Sort ascending by ‘movie id’

▪ Print format

▪ ‘movie id’ ‘movie title’

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

- 기능 : 사용자가 ‘y’를 입력하면 ‘u.item’파일에서 ‘action’장르의 영화 정보를 출력합니다.

**-** read -p "Do you want to get the data of 'action' genre from 'u.item'? (y/n) :" YN 부분을 통해 사용자에게 'action' 장르의 영화 데이터를 보고 싶은지 물어봅니다. 사용자 입력값은 YN 변수에 저장됩니다.

- if [ $YN = "y" ] … fi 를 통해 사용자가 “y”를 입력한 경우에만 실행하도록 하였습니다. “n”을 누를 경우는 따로 지시되어 있지 않아 아무것도 하지 않도록 작성하였습니다.

- cat $item\_file | awk -F\| '$7=="1"{print $1,$2}' | sort -t\| -k1,1n | head -n 10: 'u.item' 를 통해 u.item파일을 읽어와서 조건에 따라 출력하였습니다. '|' (파이프)를 통해 구분된 각 라인을 처리하고 이를 다음으로 넘겨주었으며, awk 명령어의 -F\| 옵션을 통해 '|'를 필드 구분자로 사용하였습니다. 따라서 $7는 장르 중 action 필드입니다.

action 필드가 ‘1’인 라인을 선택하여, sort 명령어를 사용하여 ‘영화ID’를 기준으로 정렬하고, head 명령어를 통해 상위 10개의 결과만을 출력하였습니다. sort -t\| -k1,1n에서 -t\| 옵션을 구분자로 ‘|’를 사용함을 나타내고, -k1,1n은 정렬을 수행할 키를 첫번째 필드를 기준으로 함을 나타냅니다. n은 숫자 형식으로 정렬하는 옵션입니다.

print $1,$2 에서 $1은 '영화 ID', $2는 '영화 제목'입니다.

**Case 3) Get the average 'rating’ of the movie identified by specific 'movie id' from 'u.data’**

▪ Please enter the 'movie id’(1~1682)

▪ Print the average ‘rating’ of the movie identified by input 'movie id’.

▪ The data for 'rating' can be found in 'u.user’

▪ Round the ‘average rating’ to six decimal places and print it with up to five decimal places

▪ Ex) 3.878318 -> 3.87832

▪ Print format

▪ average rating of ‘movie id’: ‘average rating’

**스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명**

**-** 기능 : 특정 ‘영화ID’를 입력하면 해당 영화에 대한 평균 평점을 계산하여 출력합니다.

- read -p "Please enter the 'movie id' (1~1682) : " movie\_id 를 통해 사용자로부터 '영화 ID'를 입력받습니다. read 명령어를 통해 입력 받을 수 있게 구현하였고, -p옵션을 통해 사용자에게 특정 메시지 출력하게 구현하였습니다.

- cat $data\_file | awk -v a=$movie\_id 'a==$2 {sum+=$3; count+=1 } END {if (count >0) average\_rating=sum/count} END {if (count > 0) printf ("average rating of %d : %.5f \n",a,average\_rating) }' 을 통해 'u.item' 파일을 읽어와서 조건에 따라 출력하였습니다. '|' (파이프)를 통해 구분된 각 라인을 처리하고 이를 다음으로 넘겨주었으며, awk 명령어의 -v옵션을 통해 Awk 변수 a에 Bash 변수 movie\_id의 값을 넘겨주었습니다.

awk 명령어를 사용하여 영화 ID 필드($2)가 입력받은 movie\_id와 일치하는 라인을 선택합니다. 그런 다음, sum의 변수에 선택된 라인의 평점($3)을 더하고, 해당 영화에 대한 평점의 개수 count를 증가시킵니다. END를 통해 모든 라인을 처리한 후 평균 평점을 계산하고, printf 명령어를 사용하여 해당 영화의 ID와 평균 평점을 출력합니다. 다만 영화의 개수가 0개일 경우 나눗셈이 불가능하므로 if문을 통해 그런 경우가 오지 않도록 제어하였습니다.

Printf에서의 %d는 정수를, %.5f는 소수점 다섯 자리까지의 실수를 나타냅니다.

**Case 4)** **Delete the ‘IMDb URL’ from ‘u.item’**

▪ Do you want to delete the ‘IMDb URL’ from ‘u.item’?(y/n)

▪ Get the data from 'u.item,’ delete the 'IMDb URL,' and then print the data

▪ Print 10 lines only

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

- 기능 : ‘u.item’파일에서 'IMDb URL'을 삭제할 것인지 여부를 묻고, 사용자가 'y'를 입력한 경우 'u.item' 파일에서 'IMDb URL'을 삭제한 후 상위 10개의 라인을 출력합니다.

- read -p "Do you want to delete the 'IMDb URL' from 'u.item'? (y/n) :" YN를 통해 사용자에게 'IMDb URL'을 삭제할 것인지 물어봅니다. 사용자 입력값은 YN 변수에 저장됩니다.

- if [ $YN = "y" ] … fi 를 통해 사용자가 “y”를 입력한 경우에만 실행하도록 하였습니다. “n”을 누를 경우는 따로 지시되어 있지 않아 아무것도 하지 않도록 작성하였습니다.

- cat $item\_file | sed 's/||[^|]\*|/|||/g' | head -n 10를 통해 'u.item' 파일을 읽어와서 조건에 따라 출력하였습니다.

sed 명령어를 사용하여 '||'로 시작하고 '|'로 둘러싸인 문자열을 찾아 '|||'로 대체합니다. sed 명령어 부분에서 s/는 치환을 나타내며, g는 치환할 때 모든 발생을 대상으로 함을 나타냅니다. 또한 ||[^|]\*| 에서 [^|]\*: '|' 문자가 아닌 모든 문자열이 0회 이상 반복함을 나타내므로, ||[^|]\*|는 ||로 시작하고 |가 아닌 문자가 0번 이상 반복한 후, |로 끝나는 문자입니다.

이를 통해 'IMDb URL' 부분을 삭제하게 됩니다. 그리고 head -n 10 명령어를 사용하여 상위 10개의 결과만 출력합니다.

**Case 5)** **Get the data about users from 'u.user’**

▪ Do you want to get the data about users from ‘u.user’?(y/n)

▪ Print data about users from u.user

▪ Print format

▪ user ‘user id’ is ‘age’ years old ‘gender’ ‘occupation’

▪ Print 10 lines only

스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

- 기능 : 'u.user' 파일에서 사용자 데이터를 가져올 것인지 물어보고, 사용자가 'y'를 입력한 경우 'u.user' 파일을 분석하여 상위 10개의 사용자 정보를 출력합니다.

- read -p "Do you want to get the data about users from 'u.user'? (y/n) :" YN을 통해 사용자에게 'u.user' 파일에서 사용자 데이터를 가져올 것인지 물어봅니다. 사용자 입력값은 YN 변수에 저장됩니다.

- if [ $YN = "y" ] … fi 를 통해 사용자가 “y”를 입력한 경우에만 실행하도록 하였습니다. “n”을 누를 경우는 따로 지시되어 있지 않아 아무것도 하지 않도록 작성하였습니다.

- cat $user\_file | awk -F\| '{if ($3=="F") {gender="female"}else {gender="male"}} {printf ("user %d is %d years old %s %s \n",$1,$2,gender,$4) }' | head -n 10 를 통해 'u.user' 파일을 읽어와서 조건에 따라 출력하였습니다. '|' (파이프)를 통해 구분된 각 라인을 처리하고, 이를 다음으로 넘겨주며 값을 계산합니다.

If 조건문을 통해 세 번째 필드($3)가 "F"이면 gender 변수에 "female"을, 그렇지 않으면 "male"을 할당합니다. 그 후, 사용자 ID($1), 나이($2), 성별(gender), 직업($4) 정보를 형식에 맞게 출력합니다. 단, 이때 head -n 10를 통해 상위 10개의 결과만을 출력였습니다.

**Case 6) Modify the format of 'release date' in 'u.item’**

▪ Do you want to Modify the format of ‘release data’ in ‘u.item’?(y/n)

▪ Get the data from 'u.item,’ modify the format of u.item's video release date to YYYYMMDD format, and then print the data

▪ Ex) 01-Jan-1995 -> 19950101

▪ Print only the last 10 lines (movie id: 1673 ~ 1682)

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

- 기능 : 'u.item' 파일의 'release date' 형식을 수정할 것인지 물어보고, 사용자가 'y'를 입력한 경우 'u.item' 파일에서 'release date'를 원하는 형식으로 변경하여 출력합니다.

- read -p "Do you want to Modify the format of 'release data' in 'u.item'? (y/n) " YN를 통해 사용자에게 'release date' 형식을 수정할 것인지 물어봅니다. 사용자 입력값은 YN 변수에 저장됩니다.

- if [ $YN = "y" ] … fi 를 통해 사용자가 “y”를 입력한 경우에만 실행하도록 하였습니다. “n”을 누를 경우는 따로 지시되어 있지 않아 아무것도 하지 않도록 작성하였습니다.

- cat $item\_file | sed 's/\([0-9]\{2\}\)-\(Jan\)-\([0-9]\{4\}\)/\301\1/g' | tail -n 10 | ...는 'u.item' 파일을 읽어와서, 여러 번의 sed 명령어를 사용하여 'release date' 형식에 맞게 수정한 뒤, txt파일로 저장합니다. 각 월에 해당하는 약자를 정규 표현식으로 찾아 해당하는 숫자 코드로 대체합니다. 변경된 데이터는 temp.txt 파일에 저장됩니다. (tail을 통해 뒤에서 10개를 추출하였습니다.)

sed 's/\([0-9]\{2\}\)-\(Jan\)-\([0-9]\{4\}\)/\301\1/g' | tail -n 10에서 s/는 치환을 뜻하며, 특정 패턴을 다른 패턴으로 대체합니다.

/\([0-9]\{2\}\)부분 2자리의 연도를 뜻합니다. [0-9]는 0부터 9까지의 숫자 중 하나를 의미하며 \{2\}는 해당 패턴이 2번 반복함을 나타냅니다. ‘\(‘와 ‘\)’는 그룹을 나타냅니다. 이후 ‘–‘ 이 옵니다. \{Jan\}는 2번째 그룹으로 Jan을 나타냅니다. 이후 또 다시 ‘-‘이 오고, \([0-9]\{4\}\)을 통해 3번째 그룹, 0부터 9까지의 숫자 4개를 정의합니다. 이후 \301\1 를 통해 3번째 그룹과 1번째 그룹의 자리를 바꾸어 주고 가운데에 Jan을 뜻하는 01로 대체해줍니다. 이와 같은 형식으로 모든 달에 반복하여 바꾸어 주고, temp.txt에 저장합니다.

- cat temp.txt을 통해 변경된 'release date' 형식이 저장된 temp.txt 파일을 출력합니다.

- rm temp.txt을 통하여 임시 파일인 temp.txt를 삭제합니다.

**Case 7)** **Get the data of movies rated by a specific 'user id' from 'u.data'**

▪ Please enter the ‘user id’(1~943)

▪ Print the data for movies rated by the user identified by the input 'user id’

▪ Print format

▪ Print “movie id” of all movies rated by the user identified by the input "user id“

▪ Sort ascending by ‘movie id’

▪ ‘movie id’|’movie id’|’movie id’|’movie id’…

▪ Print “movie id” and ‘title’ of 10 movies rated by the user identified by the input "user id“

▪ Sort ascending by ‘movie id’

▪ ‘movie id’|’movie title’

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

- 기능 : 'user id'를 입력받고, 해당 사용자가 평가한 영화의 정보를 처리하고 출력합니다.

- read -p "Please enter the 'user id' (1~943) :" user\_id를 통해 사용자로부터 'user id'를 입력받습니다. 입력값은 user\_id 변수에 저장됩니다.

- cat $data\_file | sort -k 2,2n | awk -v user=$user\_id '$1==user {printf ("%d|",$2)}'는 'u.data' 파일을 읽은 후 형식에 맞게 출력합니다.

sort -k 2,2n 는 ‘u.data’를 'user id'를 기준으로 정렬합니다. -t\| 옵션을 통해 구분자로 ‘|’를 사용하였고, -k2,2n를 통해 정렬을 수행할 키를 두번째 필드를 기준으로,n옵션을 통해 숫자 형식으로 정렬하였습니다.

awk 명령어의 -v옵션을 통해 Awk 변수 user에 Bash 변수 user\_id의 값을 넘겨주었습니다. 이후 'user id'가 입력받은 값과 일치하는 행에서 두 번째 필드인 'movie id'를 '|'로 구분하여 출력하였습니다.

- awk -v user=$user\_id '$1==user {print $2}' $data\_file| sort -k 1,1n > user\_rated\_movies.txt: 'u.data'는 위와 비슷합니다. 출력 형식만 다릅니다. 이를 이후에 사용하기 위해 user\_rated\_movies.txt에 저장해 주었습니다.

- awk -F\| 'FNR==NR {a[$1]++; next} $1 in a {print $1"|"$2}' user\_rated\_movies.txt $item\_file | head -n 10: user\_rated\_movies.txt는 'awk' 명령어를 사용하여 두 개의 파일을 처리합니다. 'user\_rated\_ movies.txt' 파일에서 읽은 영화 ID를 기반으로 'u.item' 파일에서 영화 정보를 추출하고, 상위 10개의 행만 출력합니다.

FNR==NR {a[$1]++; next}는 처음 파일('user\_rated\_movies.txt')을 처리하며 각 행의 첫 번째 필드(영화 ID)를 키로 하는 배열 'a'를 생성합니다. 'FNR==NR'은 첫 번째 파일에 대한 조건이며, 'next'는 다음 레코드로 넘어가도록 합니다

$1 in a {print $1"|"$2}는 두 번째 파일('u.item')을 처리하며, 각 행의 첫 번째 필드(영화 ID)가 'a' 배열에 있는지 확인하고, 해당하는 행의 첫 번째 필드(영화 ID)와 두 번째 필드(영화 제목)을 '|'로 구분하여 출력합니다.

head -n 10 를 통하여 앞에서 출력된 결과 중 상위 10개의 행만 출력하였습니다.

- rm user\_rated\_movies.txt를 통해 임시 파일인 user\_rated\_movies.txt를 삭제하였습니다.

**Case 8) Get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as 'programmer'**

▪ Do you want to get the average 'rating' of movies rated by users with 'age' between

20 and 29 and 'occupation' as 'programmer'?(y/n)

▪ Print all average ratings of movies rated by programmers in their 20s(age: 20~29)

▪ Calculate the average 'rating' based only on ratings given by programmers in their

20s(age: 20~29)

▪ Round the ‘average rating’ to six decimal places and print it with up to five decimal

places

▪ Print in this format

▪ Print all in ascending order based on ‘movie id’

▪ ‘movie id’ ‘average rating’

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

- 기능 : 'age'가 20에서 29세이고 'occupation'이 'programmer'인 사용자들이 평가한 영화의 평균 평점을 계산하고 출력합니다.

- read -p "Do you want to get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as 'programmer'? (y/n) :" YN는 사용자로부터 질문을 출력하고 입력을 받습니다. 입력값은 YN 변수에 저장됩니다.

- if [ $YN = "y" ] … fi 를 통해 사용자가 “y”를 입력한 경우에만 실행하도록 하였습니다. “n”을 누를 경우는 따로 지시되어 있지 않아 아무것도 하지 않도록 작성하였습니다.

- awk -F\| '$2>=20 && $2<=29 && $4=="programmer" {print $1}' $user\_file > programmers\_20s.txt 는 u.user' 파일에서 'age'가 20에서 29세이고 'occupation'이 'programmer'인 사용자의 'user id'를 추출하여 'programmers\_20s.txt' 파일에 저장합니다.

-F\| 옵션을 통해 'u.user' 파일의 각 행을 '|' 기호로 분리하였습니다. 그 후, $2>=20 && $2<=29 && $4=="programmer" 을 통해 두 번째 필드(나이)가 20에서 29 사이이며 네 번째 필드(직업)이 'programmer'인 행을 찾았으며, {print $1}> programmers\_20s.txt 첫 번째 필드(사용자 ID)를 추출하여 txt파일에 저장하였습니다.

- awk 'FNR==NR {programmers[$1]++; next} $1 in programmers {sum[$2]+=$3; count[$2]++;} END {for (movie\_id in sum) {avg=sum[movie\_id]/count[movie\_id]; printf("%d %.5f\n", movie\_id, avg)}}' programmers\_20s.txt $data\_file| sort -k 1,1n > average\_ratings.txt는 'programmers\_20s.txt' 파일과 'u.data' 파일을 조인하여 각 영화의 평균 평점을 계산합니다. 계산된 평균 평점은 'average\_ratings.txt' 파일에 저장됩니다.

FNR==NR {programmers[$1]++; next}는 처음 파일 'programmers\_20s.txt'을 처리하며 각 행의 첫 번째 필드(영화 ID)를 키로 하는 배열 'programmers'를 생성합니다. 이 부분은 'programmers\_20s.txt' 파일을 처리하는 부분입니다.

$1 in programmers {sum[$2]+=$3; count[$2]++;}는 두 번째 파일 'u.data'를 처리하며, 'programmers' 배열에 있는 'user id'와 일치하는 행들의 두 번째 필드(영화 ID)와 세 번째 필드(평점)을 합산합니다. 또한, 해당 영화 ID에 대한 평점 개수를 세는 'count' 배열도 업데이트합니다.

END {for (movie\_id in sum) {avg=sum[movie\_id]/count[movie\_id]; printf("%d %.5f\n", movie\_id, avg)}}'는 파일 처리가 끝난 후, 'sum' 배열(영화별 총 평점)과 'count' 배열(영화별 평점 개수)을 기반으로 각 영화의 평균 평점을 계산하고, 영화 ID와 평균 평점을 'average\_ratings.txt' 파일에 출력합니다.

| sort -k 1,1n > average\_ratings.txt: 앞서 출력한 결과를 영화 ID를 기준으로 오름차순 정렬하고, 그 결과를 'average\_ratings.txt' 파일에 저장합니다.

- cat average\_ratings.txt는 계산된 영화 평균 평점을 출력합니다.

- rm average\_ratings.txt는 앞서 출력을 완료한 임시 파일, 'average\_ratings.txt'를 삭제합니다.

- rm programmers\_20s.txt는 임시 파일인 'programmers\_20s.txt'를 삭제합니다.

**Case 9) Exit**

▪ The menu selection should continue to repeat until option 9 is chosen

▪ When option 9, 'Exit,' is selected, it will print 'Bye' and then exit the Program

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

- 기능 : “Bye!”라는 메시지를 출력하고, 프로그램을 종료합니다.

- stop변수에 “Y”를 할당합니다. 이 변수를 통해 루프를 더 이상 진행하지 않습고, 프로그램이 종료됩니다.

**[ 결과 출력 ]**

**1) Get the data of the movie identified by a specific 'movie id' from 'u.item'**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**2) Get the data of ‘action’ genre movies from 'u.item’**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**3) Get the average 'rating’ of the movie identified by specific 'movie id' from 'u.data’**

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

**4) Delete the ‘IMDb URL’ from ‘u.item’**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**5) Get the data about users from 'u.user’**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**6) Modify the format of 'release date' in 'u.item’**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**7) Get the data of movies rated by a specific 'user id' from 'u.data'**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**8) Get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as 'programmer'**

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명… (길어서 생략했습니다)

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

**9) Exit**

스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

**[ GitHub Link ]**