readme

이 문서는 `“proj1_12181656_leeseungmin.sh”` 파일을 설명합니다.

먼저 학번과 번호 별 실행하는 명령을 출력하고 choice를 입력받는 부분은 다음과 같이 구현하였습니다.

```
if [ $# -ne 3 ]; then
  echo "$0: args should be 3"
  exit 1
else
  echo "----------------------"
  echo User Name: Lee Seung Min
  echo Student Number: 12181656
  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 "----------------------"
  while true
  do
    read -p "Enter your choice [1-9] " choice
    case $choice in
```

먼저 파일에 3개의 파일이 인자로 전달되었는지 확인하기 위해 $#이 3인지 확인했습니다. 출력을 위해서는 echo 명령을, choice에 따라 나누기 위해서는 case문을 사용했습니다.

```
1)
        read -p "Please enter 'movie id (1~1682): " movie_id
        awk -v movie_id="$movie_id" -F'|' '$1 == movie_id { print $0}' "./$1"
        ;;
```

movie_id를 변수로 사용해, movie_id과 u.item의 첫번째 인자와 같으면 출력하도록 구성했습니다.

```
2)
        read -p "Do you want to get the data of ‘action’ genre movies from 'u.item’?(y/n) : " yes
        if [ "$yes" = "y" ]
        then
          awk -v cnt=0 -F'|' '$7 == 1 && cnt<10 { print $1 " " $2; cnt++}' "./$1"
        fi
        ;;
```

yes 변수 값이 y와 동일할 때만 실행하도록 처리했습니다. $7(action 장르)가 1로 set되어 있는지 확인하고 cnt 변수가 10이하일 때만 출력하도록 처리했습니다.

```
3)
        read -p "Please enter 'movie id (1~1682): " movie_id
        awk -v movie_id="$movie_id" -v cnt=0 -v sum=0 '$2 == movie_id { cnt++; sum+=$3 }  END { printf("%.6g\n", sum / cnt)}' "./$2"
        ;;
```

여기서는 2와 달리 쓰인 개념은 END입니다. 파일의 끝까지 돌고 난 후에 END 뒤에 있는 명령이 실행되게 됩니다. 이때 “%.6g”으로 “Round the ‘average rating’ to six decimal places and print it with up to five decimal places” 조건을 만족했습니다.

```
4)
        read -p "Do you want to delete the ‘IMDb URL’ from ‘u.item’?(y/n) : " yes
```

```
        if [ "$yes" = "y" ]
        then
          sed -i "s/http[^|]*//g" u.item
          awk -v cnt=0 -F'|' 'cnt<10 { print $0; cnt++}' "./$1"
        fi
        ;;
```

sed 명령어에 i 옵션을 붙여 http과 붙어 있는 부분을 제거하는 식으로 `IMDb URL` 을 제거했습니다.

```
5)
        read -p "Do you want to get the data about users from ‘u.user’?(y/n) : " yes
        if [ "$yes" = "y" ]
        then
          sed -n -E -e "s/M/male/g; s/F/female/g; s/([^|]*)\|([^|]*)\|([^|]*)\|([^|]*)\|([^|]*)/user \1 is \2 years old \3 \4/p; 10q" '
        fi
        ;;
```

먼저 M은 male로, F는 female로 변경을 해주었습니다. 이후 |가 나오기 전까지 나누어 5가지 부분으로 나눈 후 양식에 맞춰 `“user \1 is \2 years old \3 \4/p”` 출력하였습니다. -n 옵션과 10q로 10번째 줄까지만 출력했습니다.

```
6)
        read -p "Do you want to Modify the format of ‘release data’ in ‘u.item’?(y/n) : " yes
        if [ "$yes" = "y" ]
        then
          sed -E -i "s/([0-9]{2})-([A-Za-z]{3})-([0-9]{4})/\3\2\1/g" "./$1"
          sed -E -i "s/Jan/01/g; s/Feb/02/g; s/Mar/03/g; s/Apr/04/g; s/May/05/g; s/Jun/06/g; s/Jul/07/g; s/Aug/08/g; s/Sep/09/g; s/Oct/
          sed -n "1673,1682p" "./$1"
        fi
        ;;
```

5번과 마찬가지로 먼저 3가지 부분으로 나눈 후 순서를 재배열하였습니다. 이후 달을 영어에서 숫자로 표현되도록 바꾼 후 1673번째 줄에서 1682번째 줄까지 출력했습니다.

```
7)
        read -p "Please enter the 'user id' (1~943): " user_id
        awk -v user_id="$user_id"  '$1 == user_id {print $2}' "./$2" > seven.data
        sort -n <seven.data > seven_sort.data
        awk -v cnt=0 'cnt==1 {ORS=""; print "|" $0} cnt==0 {ORS=""; print $0; cnt++} END {ORS="\n\n";print "";}' seven_sort.data
        awk -F'|' -v cnt=0 'NR == FNR { values[$1] = 1; next } $1 in values&&cnt<10 {print $1 "|" $2; cnt++}' seven_sort.data "./$1"
        ;;
```

먼저 awk 함수로 첫번째 인자가 입력받은 user_id와 같은 정보에서 $2(movie_id)만 seven.data에 넣습니다. 그 다음 sort 명령을 통해 seven_sort.data에 정렬시켜 다시 저장합니다. 이때 숫자순으로 정렬하기 위해 -n 옵션을 사용했습니다. 다음 awk 명령어를 수행하는데 여기서 ORS라는 내장 변수를 사용했습니다. ORS는 출력 레코드 구분자로 기본은 “\n”로 설정되지만 수정하여 사용했습니다. 다음 줄에서도 NR과 FNR 내장 변수를 사용했습니다. NR은 전체 입력의 레코드 번호를, FNR은 구분된 입력의 레코드 번호를 의미합니다. 따라서 NR과 FNR이 같다는 것은 첫번째 파일의 데이터라는 것을 의미합니다. 즉 seven_sort.data에 값이 존재하면 u.item에서 첫번째 인자와 두번째 인자를 출력한다는 것입니다.

```
8)
        read -p "Do you want to get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as 'pro
        if [ "$yes" = "y" ]
        then
        awk -F'|' -v programmer="programmer" '$2>19 && $2<30 && $4==programmer {print $1}' "./$3" > eight.user
        awk 'NR == FNR { values[$1] = 1; next} $1 in values {print $0}' eight.user "./$2" > eight.data

        awk '{
          cnt[$2]++
          sum[$2] += $3
        }
        END {
          for (arg2 in sum) {
            if (cnt[arg2] > 0) {
              printf("%s %.06g\n", arg2, sum[arg2] / cnt[arg2])
```

```
            }
          }
        }' eight.data

      fi
      ;;
```

4번째 줄에서 awk 명령어로 programmer이면서 나이가 20대인 user id를 eight.user 파일에 저장합니다. 7에서와 같은 방법으로 eight.user에 있는 user id와 같은 $1은 가진 줄들을 eight.data에 저장합니다. 다음 cnt 변수와 sum 변수를 이용해 평균을 계산해 출력합니다.

```
9)
        echo Bye!
        exit
```

9를 누르면 exit을 수행합니다.