Northeastern University, Khoury College of Computer Science

CS 6220 Data Mining | Assignment 1

Due: September 16, 2024 (100 points)

Xinyue Han

han.xinyue@northeastern.edu

https://github.com/aiC0ld/CS6220-DataMining

Coding Review

In subsequent lectures, you'll learn about frequent item sets, where relationships between items are learned by observing how often they co-occur in a set of data. This information is useful for making recommendations in a rule based manner. Before looking at frequent item sets, it is worth understanding the space of all possible sets and get a sense for how quickly the number of sets with unique items grows.

Suppose that we've received only a hundred records of items bought by customers at a market. Take a look at data: https://course.ccs.neu.edu/cs6220/fall2024/homework-1/basket data.csv. Each line in the file represents the items an individual customer bought, i.e. their basket. For example, consider the following rows.

ham, cheese,
bread dates,
bananas
celery, chocolate bars

Customer 1 has a basket of ham, cheese, and bread. Customer 2 has a basket of dates and bananas. Customer 3 has a basket of celery and chocolate bars. Each of these records is the receipt of a given customer, identifying what they bought.

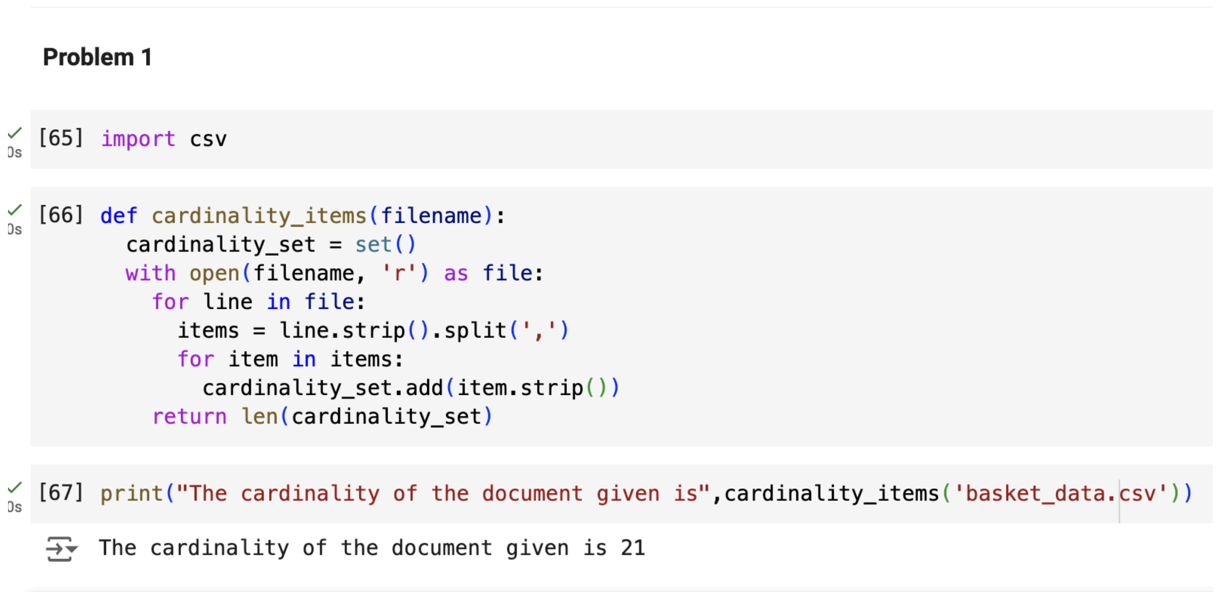
Please answer the following:

1. The *cardinality* of a set or collection of items is the number of unique items in that collection. Write a function called cardinality_items that takes a .csv text string file as input, where the format is as the above, and calculates the cardinality of the set of all the grocery items in any given dataset.

```
def cardinality_items(
      filename ): ’’’
Takes a filename "*.csv" and returns an
integer
’’’
< YOUR CODE HERE >
return 0
```

What is the cardinality in “basket data.csv”?

Answer:



Problem 1

```
[65] import csv
```

```
[66] def cardinality_items(filename):
       cardinality_set = set()
       with open(filename, 'r') as file:
         for line in file:
           items = line.strip().split(',')
           for item in items:
             cardinality_set.add(item.strip())
         return len(cardinality_set)
```

```
[67] print("The cardinality of the document given is",cardinality_items('basket_data.csv'))
```

```
The cardinality of the document given is 21
```

So the cardinality of the document given is 21.

2. Write a function called all_itemsets that takes a list of unique items and an integer k as input, and the output is a list of all possible unique itemsets with non-repeating k items. That is, the output is L = [S1, S2, · · · SN], a list of all possible sets, where each Si has k items.

For example,

all_itemsets(["ham", "cheese", "bread"], 2)

should result in:

[["ham", "cheese"], ["ham", "bread"], [’cheese", "bread"]]

Please don’t use any library functions as you will not need them. Your function signature should look like this:

```
def all_itemsets(
      unique_items ): ’’’
Enumerate out all itemsets. Note that for each itemset, order
does not matter.
```

Input:

- unique_items: a list of items (should be unique)

Output:

- all_itemsets: a list of lists: represents all possible baskets of items.

’’’

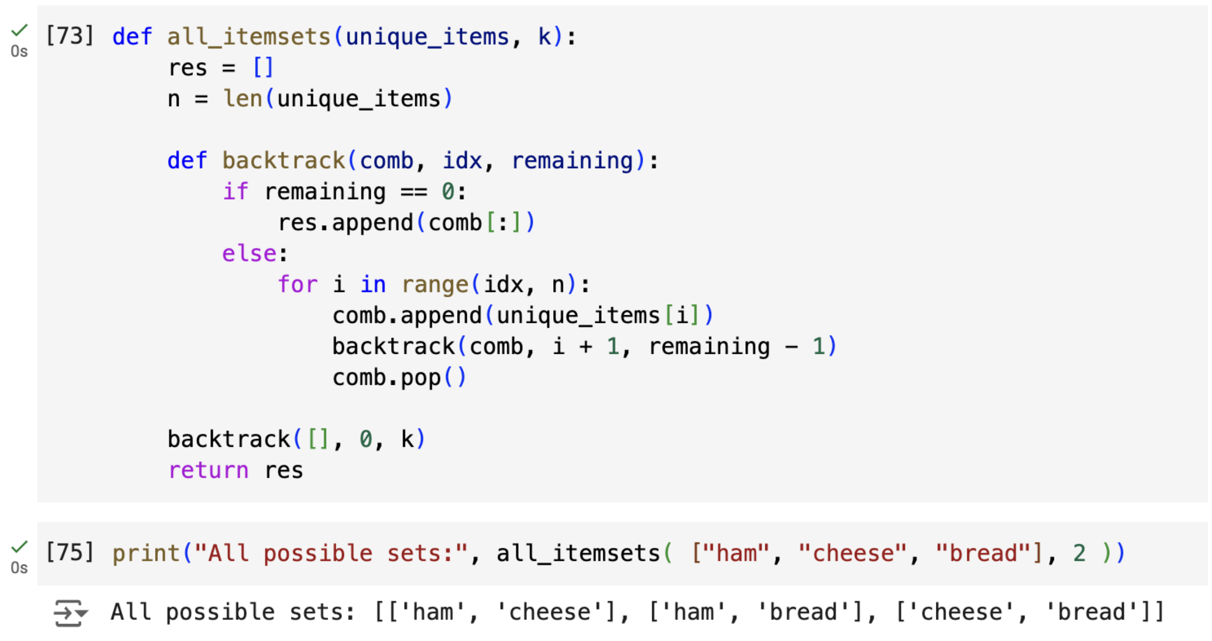
< YOUR CODE HERE >

return 0

Output your data into output.txt (which will be uploaded via Gradescope).

Answer:

Problem 2



```
[73] def all_itemsets(unique_items, k):
         res = []
         n = len(unique_items)

         def backtrack(comb, idx, remaining):
             if remaining == 0:
                 res.append(comb[:])
             else:
                 for i in range(idx, n):
                     comb.append(unique_items[i])
                     backtrack(comb, i + 1, remaining - 1)
                     comb.pop()

         backtrack([], 0, k)
         return res
```

```
[75] print("All possible sets:", all_itemsets( ["ham", "cheese", "bread"], 2 ))
```

```
All possible sets: [['ham', 'cheese'], ['ham', 'bread'], ['cheese', 'bread']]
```

So all possible sets: [['ham', 'cheese'], ['ham', 'bread'], ['cheese', 'bread']]

Examining Our First Dataset

One of the most famous challenges in data science and machine learning is Netflix’s Grand Prize Challenge, where Netflix held an open competition for the best algorithm to predict user ratings for films. The grand prize was $1,000,000 and was won by BellKor’s Pragmatic Chaos team. The original dataset that was used in that competition was downloaded from Kaggle (netflix- inc/netflix-prize-data), and you can download it here:

- https://course.ccs.neu.edu/cs6220/fall2024/homework-1/netflix-data/

In this exercise, we're going to do a bit of exploring in the Netflix Data. Start by downloading the data. Data integrity tends to be a problem in large scale processing, especially if there is little to no support. Therefore, it's important to verify the quality of the file download. If all worked out well, you should have the following files:

- combined data 1.txt
- combined data 2.txt
- combined data 3.txt
- combined data 4.txt
- movie titles.csv
- probe.txt
- qualifying.txt
- README

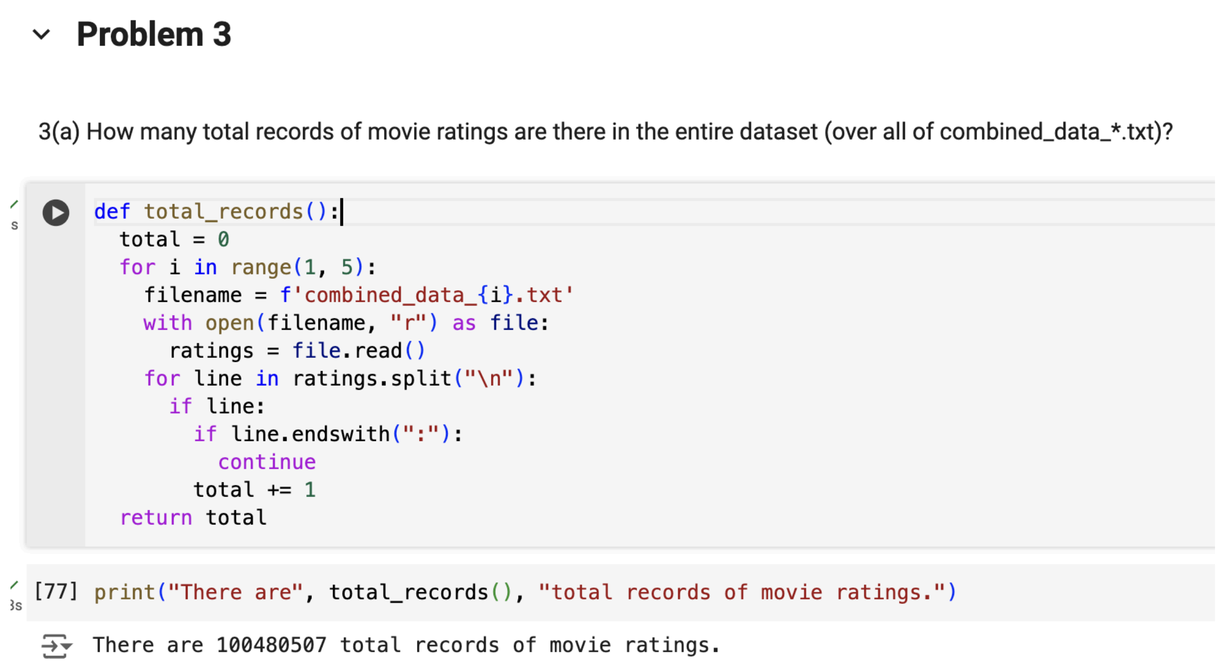
Data Verification and Analysis

A large part of machine learning and data science is about getting data in the right format and ensuring that there is no data corruption. Verify that the schema is the same as the Kaggle Dataset's description, read in the data (hint: some movies have commas in them), and then answer the following questions.

Please answer the following:

3. Let's review combined_data_*.txt.

   a) How many total records of movie ratings are there in the entire dataset (over all of combined_data_*.txt)?



Problem 3

3(a) How many total records of movie ratings are there in the entire dataset (over all of combined_data_*.txt)?

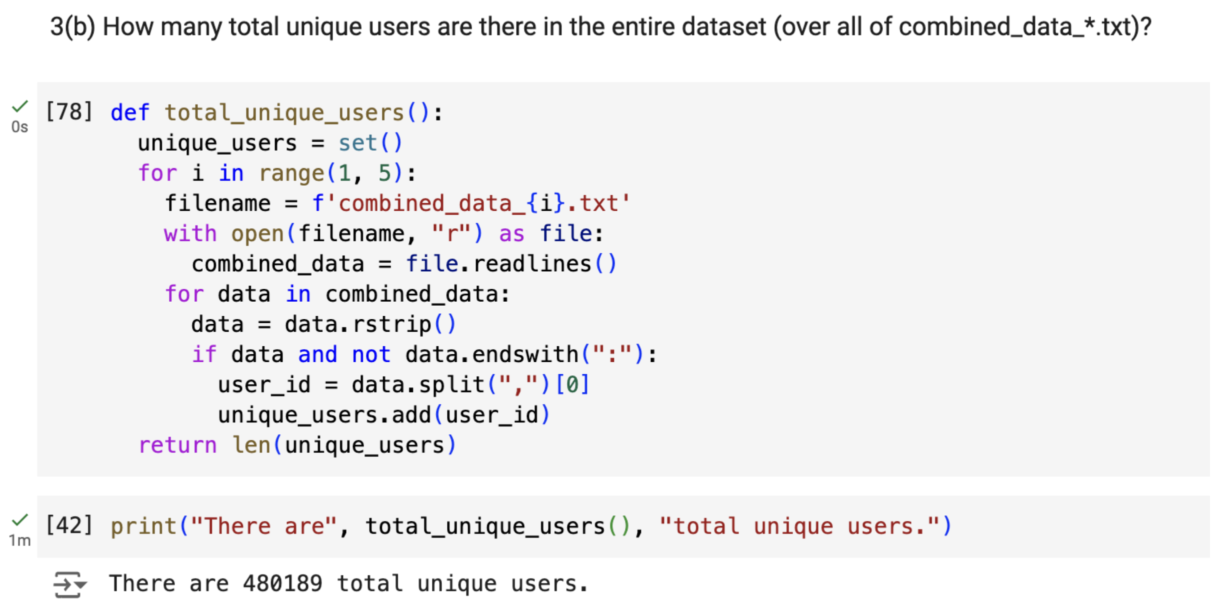
```
def total_records():
  total = 0
  for i in range(1, 5):
    filename = f'combined_data_{i}.txt'
    with open(filename, "r") as file:
      ratings = file.read()
    for line in ratings.split("\n"):
      if line:
        if line.endswith(":"):
          continue
        total += 1
  return total
```

```
[77] print("There are", total_records(), "total records of movie ratings.")
```

```
There are 100480507 total records of movie ratings.
```

There are 100480507 total records of movie ratings.

b) How many total unique users are there in the entire dataset (over all of combined_data_*.txt)?



3(b) How many total unique users are there in the entire dataset (over all of combined_data_*.txt)?

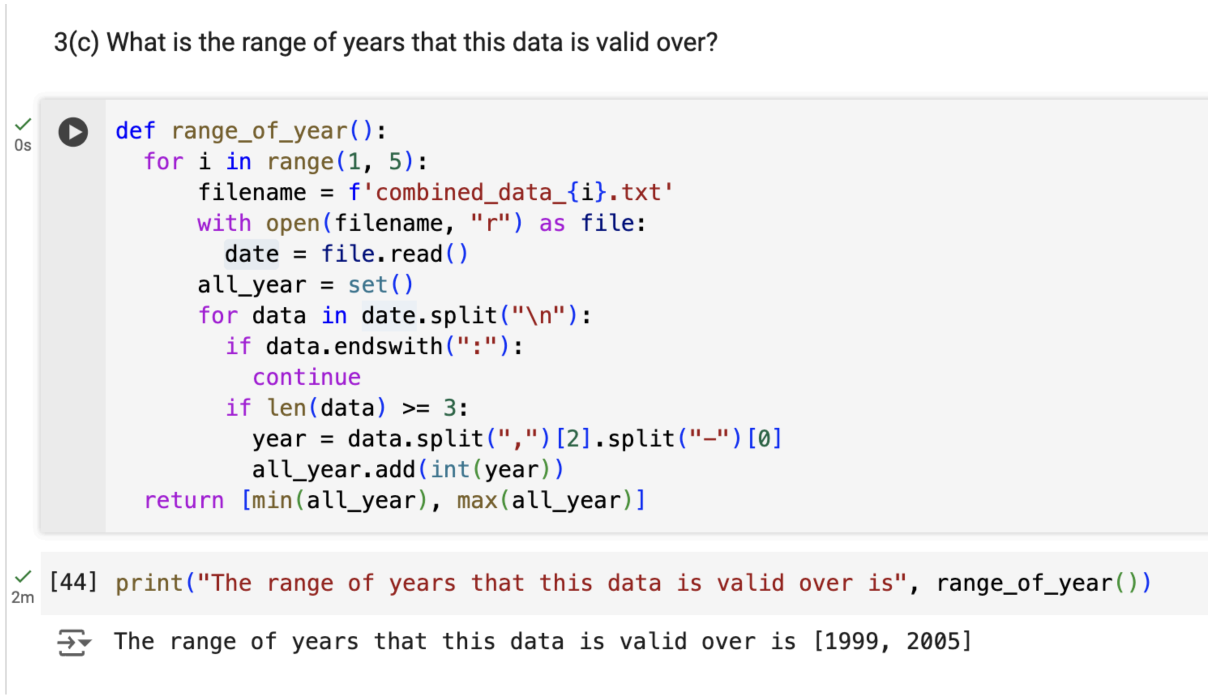
```
[78] def total_unique_users():
       unique_users = set()
       for i in range(1, 5):
         filename = f'combined_data_{i}.txt'
         with open(filename, "r") as file:
           combined_data = file.readlines()
         for data in combined_data:
           data = data.rstrip()
           if data and not data.endswith(":"):
             user_id = data.split(",")[0]
             unique_users.add(user_id)
       return len(unique_users)
```

```
[42] print("There are", total_unique_users(), "total unique users.")
```

```
There are 480189 total unique users.
```

There are 480189 total unique users.

c) What is the range of years that this data is valid over?



3(c) What is the range of years that this data is valid over?

```
def range_of_year():
  for i in range(1, 5):
      filename = f'combined_data_{i}.txt'
      with open(filename, "r") as file:
        date = file.read()
      all_year = set()
      for data in date.split("\n"):
        if data.endswith(":"):
          continue
        if len(data) >= 3:
          year = data.split(",")[2].split("-")[0]
          all_year.add(int(year))
  return [min(all_year), max(all_year)]
```

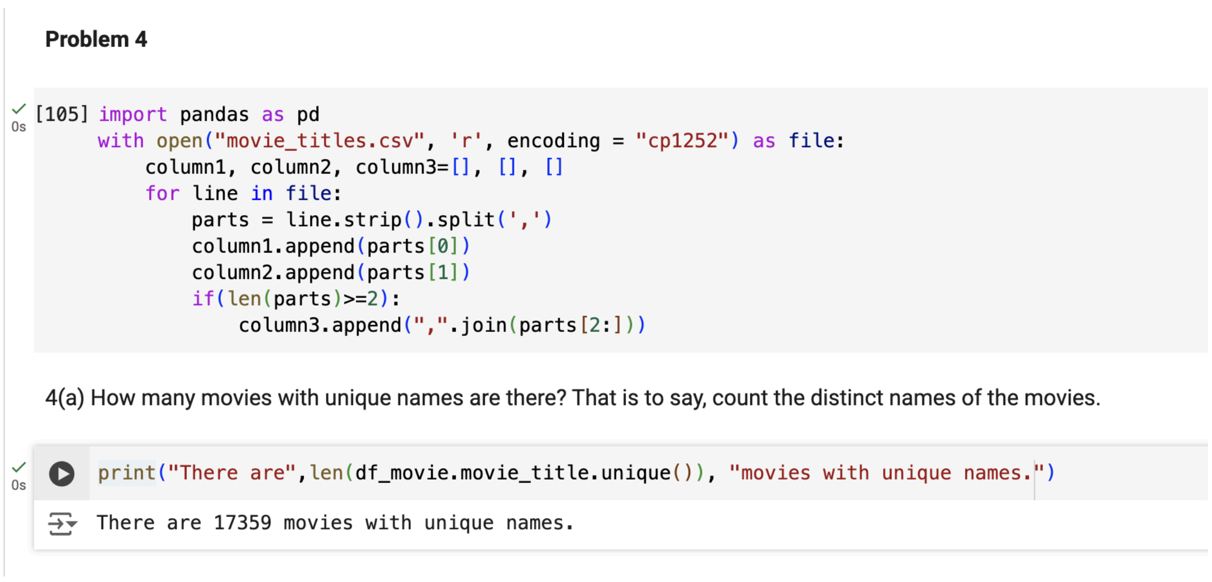
```
[44] print("The range of years that this data is valid over is", range_of_year())
```

```
The range of years that this data is valid over is [1999, 2005]
```

The range of years that this data is valid over is [1999, 2005]

4. Let's review movie_titles.csv.

a) How many movies with unique names are there? That is to say, count the distinct names of the movies.



Problem 4

```
[105] import pandas as pd
      with open("movie_titles.csv", 'r', encoding = "cp1252") as file:
          column1, column2, column3=[], [], []
          for line in file:
              parts = line.strip().split(',')
              column1.append(parts[0])
              column2.append(parts[1])
              if(len(parts)>=2):
                  column3.append(",".join(parts[2:]))
```

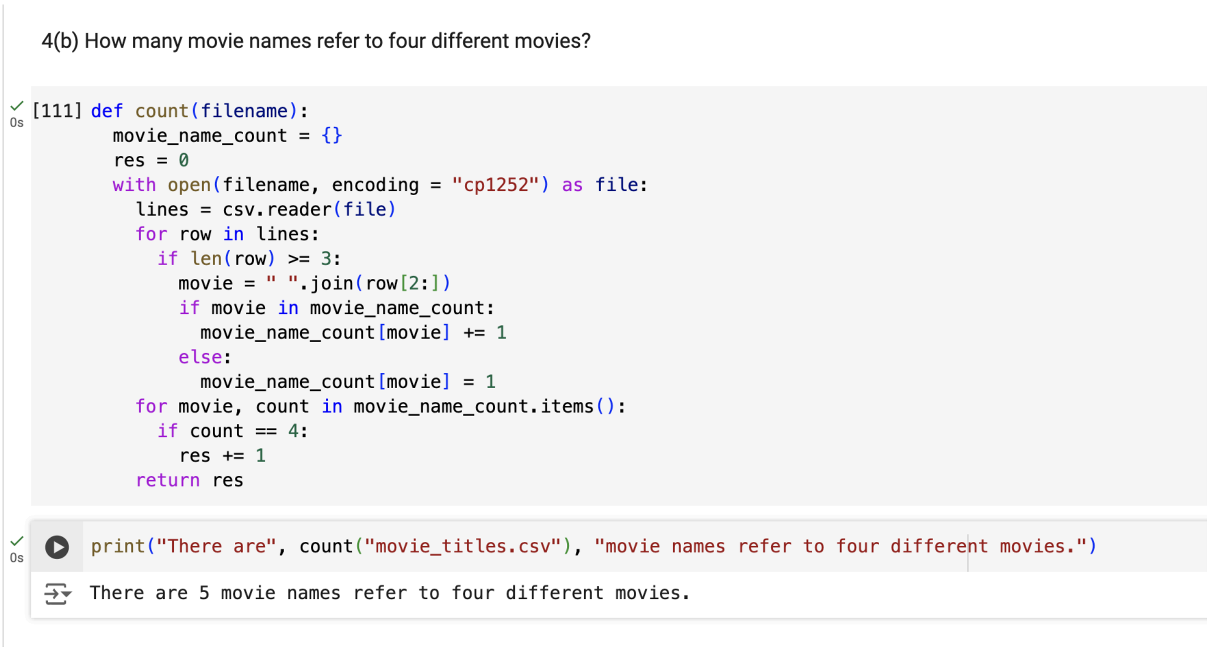
4(a) How many movies with unique names are there? That is to say, count the distinct names of the movies.

```
print("There are",len(df_movie.movie_title.unique()), "movies with unique names.")
```

```
There are 17359 movies with unique names.
```

There are 17359 movies with unique names.

b) How many movie names refer to four different movies?



4(b) How many movie names refer to four different movies?

```
[111] def count(filename):
        movie_name_count = {}
        res = 0
        with open(filename, encoding = "cp1252") as file:
          lines = csv.reader(file)
          for row in lines:
            if len(row) >= 3:
              movie = " ".join(row[2:])
              if movie in movie_name_count:
                movie_name_count[movie] += 1
              else:
                movie_name_count[movie] = 1
          for movie, count in movie_name_count.items():
            if count == 4:
              res += 1
          return res
```

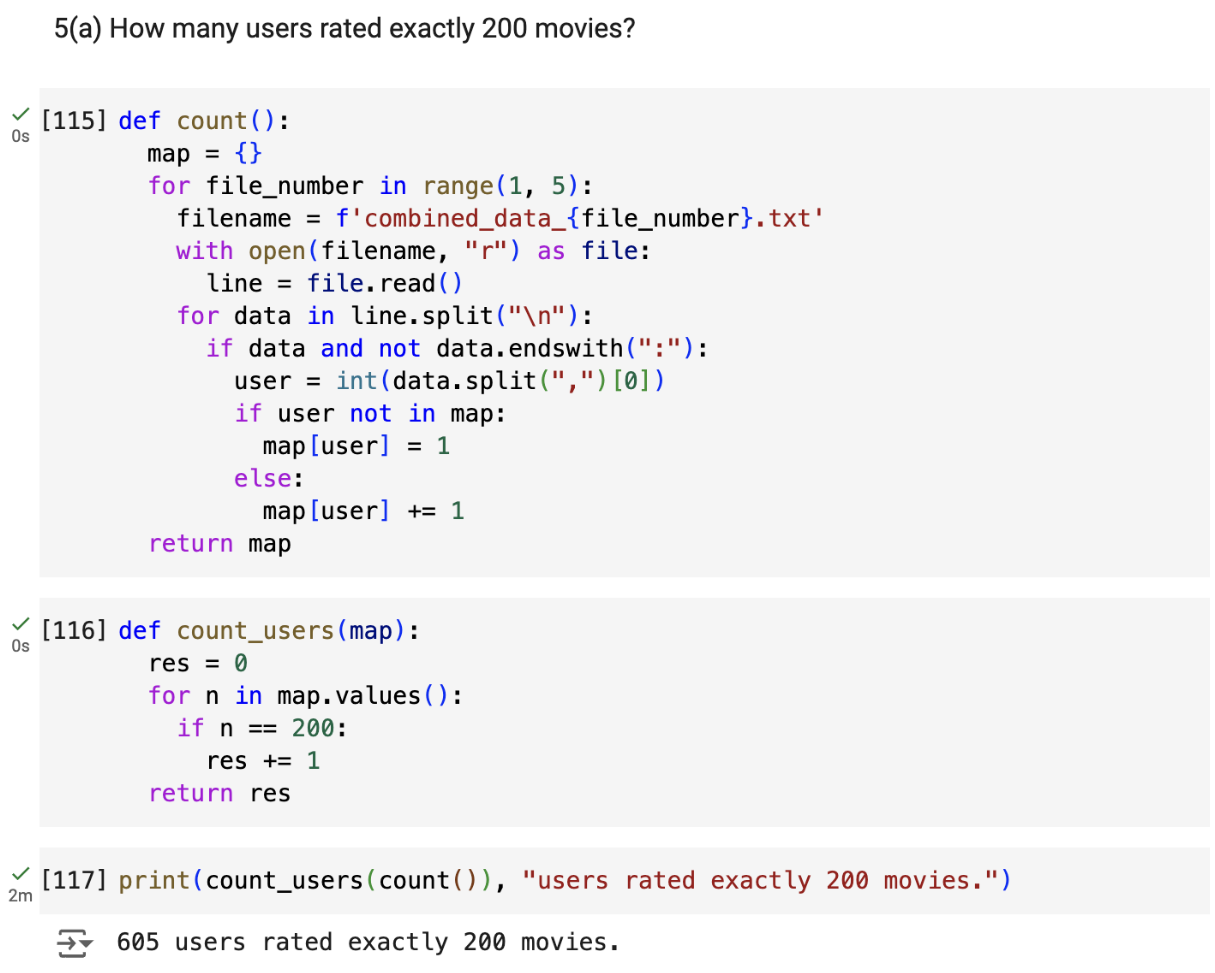
```
print("There are", count("movie_titles.csv"), "movie names refer to four different movies.")
```

```
There are 5 movie names refer to four different movies.
```

There are 5 movie names refer to four different movies.

5. Let’s review both.

a) How many users rated exactly 200 movies?


5(a) How many users rated exactly 200 movies?

```
[115] def count():
        map = {}
        for file_number in range(1, 5):
          filename = f'combined_data_{file_number}.txt'
          with open(filename, "r") as file:
            line = file.read()
          for data in line.split("\n"):
            if data and not data.endswith(":"):
              user = int(data.split(",")[0])
              if user not in map:
                map[user] = 1
              else:
                map[user] += 1
        return map
```

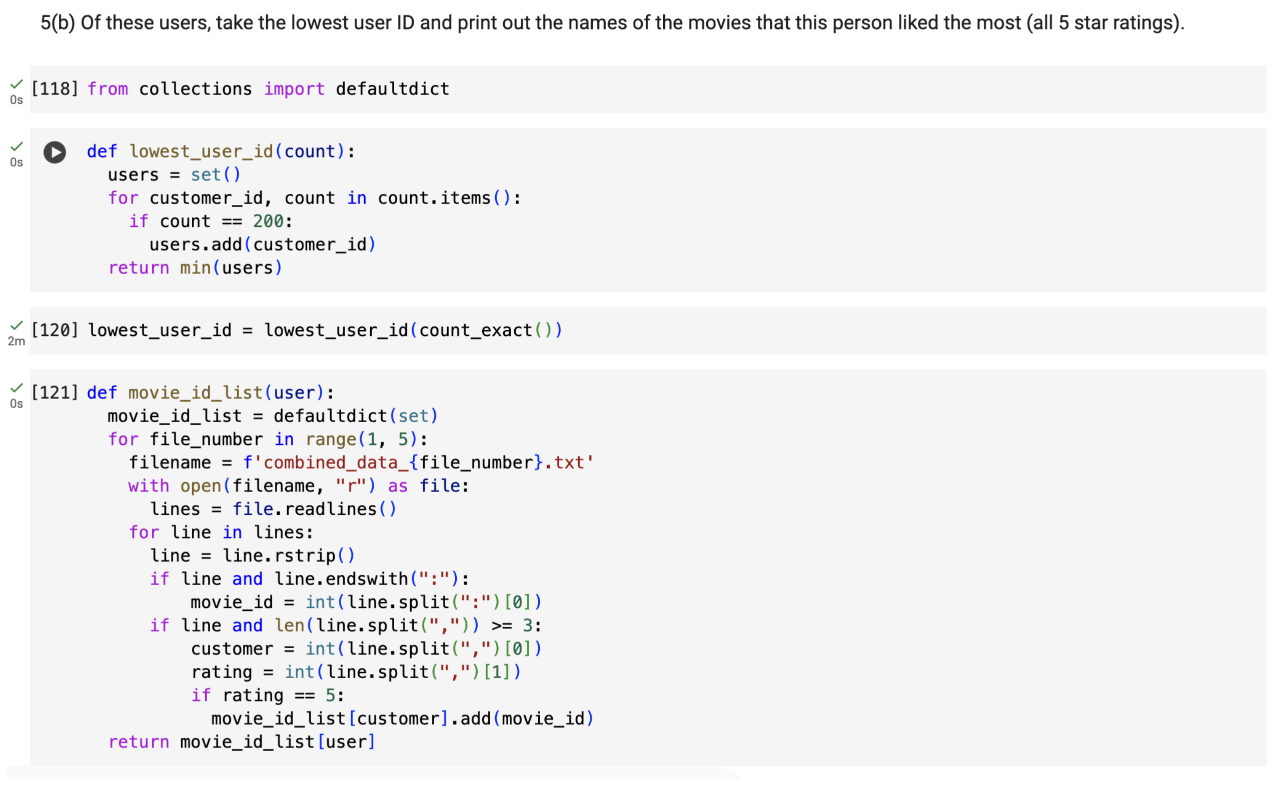
```
[116] def count_users(map):
        res = 0
        for n in map.values():
          if n == 200:
            res += 1
        return res
```

```
[117] print(count_users(count()), "users rated exactly 200 movies.")
```

```
605 users rated exactly 200 movies.
```

605 users rated exactly 200 movies.

b) Of these users, take the lowest user ID and print out the names of the movies that this person liked the most (all 5 star ratings).

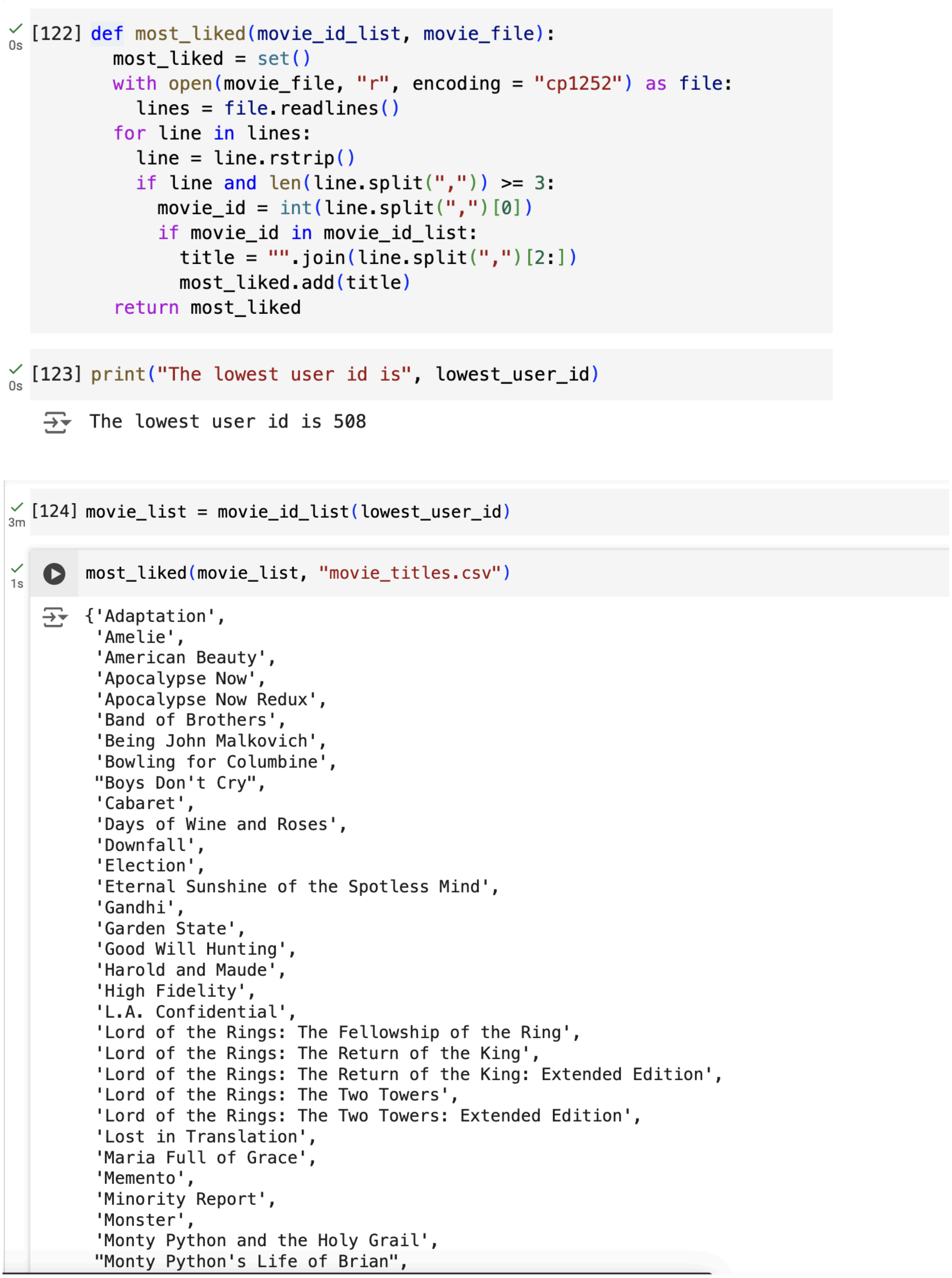

5(b) Of these users, take the lowest user ID and print out the names of the movies that this person liked the most (all 5 star ratings).

```
[118] from collections import defaultdict
```

```
def lowest_user_id(count):
  users = set()
  for customer_id, count in count.items():
    if count == 200:
      users.add(customer_id)
  return min(users)
```

```
[120] lowest_user_id = lowest_user_id(count_exact())
```

```
[121] def movie_id_list(user):
        movie_id_list = defaultdict(set)
        for file_number in range(1, 5):
          filename = f'combined_data_{file_number}.txt'
          with open(filename, "r") as file:
            lines = file.readlines()
          for line in lines:
            line = line.rstrip()
            if line and line.endswith(":"):
                movie_id = int(line.split(":")[0])
            if line and len(line.split(",")) >= 3:
                customer = int(line.split(",")[0])
                rating = int(line.split(",")[1])
                if rating == 5:
                  movie_id_list[customer].add(movie_id)
        return movie_id_list[user]
```



```
[122] def most_liked(movie_id_list, movie_file):
        most_liked = set()
        with open(movie_file, "r", encoding = "cp1252") as file:
          lines = file.readlines()
        for line in lines:
          line = line.rstrip()
          if line and len(line.split(",")) >= 3:
            movie_id = int(line.split(",")[0])
            if movie_id in movie_id_list:
              title = "".join(line.split(",")[2:])
              most_liked.add(title)
        return most_liked
```

```
[123] print("The lowest user id is", lowest_user_id)
```

```
The lowest user id is 508
```

```
[124] movie_list = movie_id_list(lowest_user_id)
```

```
most_liked(movie_list, "movie_titles.csv")
```

```
{'Adaptation',
 'Amelie',
 'American Beauty',
 'Apocalypse Now',
 'Apocalypse Now Redux',
 'Band of Brothers',
 'Being John Malkovich',
 'Bowling for Columbine',
 "Boys Don't Cry",
 'Cabaret',
 'Days of Wine and Roses',
 'Downfall',
 'Election',
 'Eternal Sunshine of the Spotless Mind',
 'Gandhi',
 'Garden State',
 'Good Will Hunting',
 'Harold and Maude',
 'High Fidelity',
 'L.A. Confidential',
 'Lord of the Rings: The Fellowship of the Ring',
 'Lord of the Rings: The Return of the King',
 'Lord of the Rings: The Return of the King: Extended Edition',
 'Lord of the Rings: The Two Towers',
 'Lord of the Rings: The Two Towers: Extended Edition',
 'Lost in Translation',
 'Maria Full of Grace',
 'Memento',
 'Minority Report',
 'Monster',
 'Monty Python and the Holy Grail',
 "Monty Python's Life of Brian",
        "Monty Python's The Meaning of Life: Special Edition",
        'Raging Bull',
        'Raising Arizona',
        'Roger & Me',
        "Schindler's List",
        'Shakespeare in Love',
        'Sideways',
        'Super Size Me',
        'Taxi Driver',
        'The Accused',
        'The Lord of the Rings: The Fellowship of the Ring: Extended Edition',
        'The Manchurian Candidate',
        'The Pianist',
        'The Royal Tenenbaums',
        'The Shawshank Redemption: Special Edition',
        'The Silence of the Lambs',
        'The Usual Suspects',
        'This Is Spinal Tap',
        'Three Kings',
        'To Be and To Have',
        'To Die For',
        'Touching the Void',
        'Unforgiven',
        'Vietnam: A Television History',
        'Whale Rider'}
```

The lowest user id is 508

Name of the movies:

{'Adaptation',

'Amelie',

'American Beauty',

'Apocalypse Now',

'Apocalypse Now Redux',

'Band of Brothers',

'Being John Malkovich',

'Bowling for Columbine',

"Boys Don't Cry",

'Cabaret',

'Days of Wine and Roses',

'Downfall',

'Election',

'Eternal Sunshine of the Spotless Mind',

'Gandhi',

'Garden State',

'Good Will Hunting',

'Harold and Maude',

'High Fidelity',

'L.A. Confidential',

'Lord of the Rings: The Fellowship of the Ring',

'Lord of the Rings: The Return of the King',

'Lord of the Rings: The Return of the King: Extended Edition',

'Lord of the Rings: The Two Towers',

```
'Lord of the Rings: The Two Towers: Extended Edition',
'Lost in Translation',
'Maria Full of Grace',
'Memento',
'Minority Report',
'Monster',
'Monty Python and the Holy Grail',
"Monty Python's Life of Brian",
"Monty Python's The Meaning of Life: Special Edition",
'Raging Bull',
'Raising Arizona',
'Roger & Me',
"Schindler's List",
'Shakespeare in Love',
'Sideways',
'Super Size Me',
'Taxi Driver',
'The Accused',
'The Lord of the Rings: The Fellowship of the Ring: Extended Edition',
'The Manchurian Candidate',
'The Pianist',
'The Royal Tenenbaums',
'The Shawshank Redemption: Special Edition',
'The Silence of the Lambs',
'The Usual Suspects',
'This Is Spinal Tap',
'Three Kings',
'To Be and To Have',
'To Die For',
'Touching the Void',
'Unforgiven',
'Vietnam: A Television History',
'Whale Rider'}
```

Submission Instructions

Add all functions to homework1.py. Submit via Gradescope. There, you will upload output.txt, PDF, and Python code.