Question 1: Most Frequently Assigned Rating Score

Code Purpose:

This Python code is designed to answer Question 1 of the coursework. The objective is to determine the most frequently assigned rating score and find out how many entries in the dataset have that rating score. The code reads data from 'ratings.csv' and counts the occurrences of each rating.

Code Description:

1. **Data Processing**:
    - The code initializes a dictionary, 'rating_counts', to store the counts of each rating from the ratings file.
2. **Read and Count Ratings**:
    - It reads the data from 'ratings.csv' and counts the occurrences of each rating. For each rating, it increments the count in the rating_counts dictionary.
3. **Find Most Frequent Rating**:
    - After processing all the data, the code identifies the rating with the highest count, indicating the most frequently assigned rating score.
4. **Print Results**:
    - The code prints the most frequent rating score and the number of entries with that rating score.

Code Output:

The code outputs the most frequently assigned rating score and the number of entries in the dataset with that rating score.

```
import csv

# Create a dictionary to store the counts of each rating from the ratings file
rating_counts = {}

# Read data from ratings file and count the occurrences of each rating
with open('ratings.csv', 'r') as ratings_file:

    ratings_file_reader = csv.reader(ratings_file)

    # Skip the header row in the ratings file
    next(ratings_file_reader)

    for line in ratings_file_reader:

        rating = line[2]

        # Increment the rating count
        if rating in rating_counts:

            rating_counts[rating] += 1

        else:

            rating_counts[rating] = 1



# Find the most common rating and its count
most_common_rating = max(rating_counts, key=rating_counts.get)

most_common_rating_count = rating_counts[most_common_rating]

# Print the results
print('The most frequent rating:', most_common_rating)

print('Number of entries for this rating:', most_common_rating_count)
```

```
The most frequent rating: 4.0
Number of entries for this rating: 27742
```

Question 2: Producing Statistics of Languages for Movies Released in a Specific Year

Code Purpose:

This Python code answers Question 2 of the coursework. It aims to produce statistics of languages for movies released in a specific year. The code prompts the user to input a year, reads movie data from 'movies.json', and counts the occurrences of languages in movies released in that year.

Code Description:

1. **Data Preparation**:
    - The code initializes an empty dictionary named `language_counts` to store the counts of different languages found in movies released in the specified year.
2. **User Input (Year)**:
    - Instead of using `input()` to prompt the user for a year, this code directly assigns the year '1995' to the `year` variable. You can uncomment the `input()` line to enable user input if desired.
3. **Read and Process Movie Data**:
    - The code reads movie data from 'movies.json' and checks if each movie's release year (extracted from the 'releasedate' field) matches the specified year ('1995' in this case). It extracts the language of the movie (or assigns 'Unknown' if the language is missing) and updates the count in the `language_counts` dictionary.
4. **Sort Language Counts**:
    - After processing all the data, the code sorts the language counts in descending order, ensuring that the most common languages appear first.
5. **Print Results**:
    - Finally, it prints the statistics, including the languages and their respective counts for movies released in the specified year.

Code Output:

When executed, this code displays statistics for movies released in the year '1995' (or the specified year), including the languages used in those movies and the number of movies for each language.

```python
import json

# Create a dictionary to store language counts
language_counts = {}

# Prompt the user to input a year
year = '1995' # input('Please input a year:')

# Read movie data from 'movies.json' and count the occurrences of languages in the specified year
with open('movies.json', 'r') as movies_file:
    for line in movies_file:
        movie_data = json.loads(line)

        # Check if the movie's release year matches the specified year. The format of the year is '1995-09-09'
        if year == movie_data.get('releasedate', '')[:4]:
            language = movie_data.get('language', 'Unknown')
            language_counts[language] = language_counts.get(language, 0) + 1

# Sort the language counts in descending order
sorted_language_counts = sorted(language_counts.items(), key=lambda x: x[1], reverse=True)

# print the language and its count for specified year
if sorted_language_counts:
    print('Statistics for movies released in', year,':\n')
    for language, count in sorted_language_counts:
        print('Language:', language.upper(), 'Count:', count)
else:
    print('No movies found for the year', year)
```

```
Statistics for movies released in 1995 :

Language: EN Count: 456
Language: FR Count: 28
Language: JA Count: 21
Language: DE Count: 16
Language: ES Count: 13
Language: IT Count: 9
Language: CN Count: 8
Language: SV Count: 7
Language: NL Count: 6
Language: PT Count: 5
Language: HI Count: 5
Language: ZH Count: 3
Language: SR Count: 3
```

```
Language: NO Count: 2
Language: FI Count: 2
Language: PL Count: 2
Language: RU Count: 2
Language: TA Count: 2
Language: FA Count: 1
Language: KO Count: 1
Language: VI Count: 1
Language: EL Count: 1
Language: MR Count: 1
Language: DA Count: 1
Language: SK Count: 1
Language: IS Count: 1
Language: TE Count: 1
```

Question 3: Movie with the Highest Average Rating (Minimum 25 Reviews)

Code Purpose:

This Python code aims to find the title of the movie with at least 25 reviews that has the highest overall average rating in the dataset. The code reads movie data from 'movies.json' and rating data from 'ratings.csv', calculates the average ratings for each movie with a minimum of 25 reviews, and identifies the movie with the highest average rating.

Code Description:

1. **Data Preparation**:
   - The code initializes dictionaries: `movie_id_to_title` (to map movie IDs to titles), `movie_ratings` (to store total ratings for each movie), and `movie_ratings_count` (to store the count of ratings for each movie).
2. **Read and Process Movie Data**:
   - It reads movie data from 'movies.json' and extracts movie titles and IDs. This information is stored in the `movie_id_to_title` dictionary.
3. **Read and Process Rating Data**:
   - The code reads rating data from 'ratings.csv', calculates total ratings and counts for each movie, and stores this information in the `movie_ratings` and `movie_ratings_count` dictionaries.
4. **Find Movies with Minimum 25 Reviews**:
   - It identifies movies with at least 25 reviews and calculates their average ratings.
5. **Find Highest Rated Movie**:
   - After processing, the code identifies the movie with the highest average rating among those with a minimum of 25 reviews.
6. **Print Results**:
   - The code prints the title of the movie with the highest average rating and the average rating itself.

Code Output:

When executed, this code outputs the title of the movie with at least 25 reviews that has the highest overall average rating in the dataset, along with the average rating.

```python
import csv

import json

# Create a dictionary to map movie IDs to movie titles
movie_id_to_title = {}
# Create dictionaries to store movie ratings and their counts
movie_ratings = {}
movie_ratings_count = {}

# Read movie data from 'movies.json' and populate the dictionary
with open('movies.json', 'r') as movies_file:
    for line in movies_file:
        movie_data = json.loads(line)
        title = movie_data['title']
        movie_id = movie_data['id']
        movie_id_to_title[movie_id] = title

# Read rating data from 'ratings.csv' and calculate total ratings and counts
with open('ratings.csv', 'r') as ratings_file:
    ratings_file_reader = csv.reader(ratings_file)
    next(ratings_file_reader)  # Skip header row

    for line in ratings_file_reader:
        movie_id = line[1]
        rating = float(line[2])
        if movie_id in movie_ratings:
            movie_ratings[movie_id] += rating
            movie_ratings_count[movie_id] += 1
        else:
            movie_ratings[movie_id] = rating
            movie_ratings_count[movie_id] = 1

# Find movies with at least 25 ratings and calculate their average ratings
top_rated_movies = []

for movie_id, total_rating in movie_ratings.items():
    if movie_ratings_count[movie_id] >= 25:
        avg_rating = round(total_rating / movie_ratings_count[movie_id], 2)
        top_rated_movies.append((avg_rating, movie_id))

# Find the highest rated movie
highest_rated_movie = max(top_rated_movies)

# Unpack the values from the highest rated movie tuple
average_rating, best_movie_id = highest_rated_movie

# Get the title of the highest rated movie
best_movie_title = movie_id_to_title.get(best_movie_id, "Title not found")

# Print the results
print('Title:', best_movie_title, '\nAverage rating:', average_rating)
```

```
Title: Die Büchse der Pandora
Average rating: 4.49
```

Question 4: Movie with the Most 5.0 Ratings

Code Purpose:

The provided Python code answers Question 4 of the coursework. The goal is to find the movie title with the largest number of 5.0 ratings and determine how many 5.0 ratings it has. The code reads rating data from 'ratings.csv', identifies 5.0 ratings, and tracks the count of these ratings for each movie.

Code Description:

1. **Function Definitions:**
   - `find_movie_with_most_5_ratings()` : This function reads data from 'ratings.csv', identifies 5.0 ratings, and determines the movie with the most 5.0 ratings along with its count.
2. **Identify 5.0 Ratings:**
   - The code reads rating data from 'ratings.csv' and checks if each rating is equal to 5.0. It counts the number of 5.0 ratings for each movie and stores this information in the `total_number_of_5_ratings` dictionary.
3. **Find Movie with Most 5.0 Ratings:**
   - After processing all the ratings, the code identifies the movie(s) with the largest number of 5.0 ratings.
4. **Print Results:**
   - The code prints the title of the movie with the most 5.0 ratings and the total count of 5.0 ratings for that movie. It utilizes the `movie_id_to_title` dictionary created in a previous question to display the movie title.

Usage Example:

To find the movie with the most 5.0 ratings and its count, simply call the `find_movie_with_most_5_ratings()` function. The code will display the movie title and the number of 5.0 ratings it has received.

Code Output:

When executed, the code outputs the title of the movie with the most 5.0 ratings and the total count of 5.0 ratings for that movie.

```
import csv

import json

# Create a dictionary to store the count of 5.0 ratings for each movie
total_number_of_5_ratings = {}

# Read rating data from 'ratings.csv'
```

```
with open('ratings.csv', 'r') as ratings_file:
    ratings_file_reader = csv.reader(ratings_file)
    next(ratings_file_reader)  # Skip header row

    for line in ratings_file_reader:
        rating = float(line[2])
        movie_id = line[1]

        # Check if the rating is 5.0
        if rating == 5.0:
            # Update the count of 5.0 ratings for the movie
            if movie_id not in total_number_of_5_ratings:
                total_number_of_5_ratings[movie_id] = 1
            else:
                total_number_of_5_ratings[movie_id] += 1

# Find the movie(s) with the largest number of 5.0 ratings
most_5_ratings = max(total_number_of_5_ratings, key=total_number_of_5_ratings.get)

# print the results. We use the movie_id_to_title dictionary we created in question 3.
print('The title of the movie is:', movie_id_to_title[most_5_ratings])
print('The total number of 5.0 ratings is:', total_number_of_5_ratings[most_5_ratings])
```

```
The title of the movie is: The Million Dollar Hotel
The total number of 5.0 ratings is: 424
```

Question 5: Top 5 Movies by Genre with the Highest Average Rating

Code Purpose:

The provided Python code answers Question 5 of the coursework. The goal is to create a function that takes a movie genre as an input parameter and produces a list of the top 5 movie titles within that genre, ranked by their average ratings. The code reads movie data from 'movies.json' and rating data from 'ratings.csv', filters movies by the specified genre, and calculates the average ratings for each movie in that genre.

Code Description:

1. **Function Definitions:**
   - `read_data_for_genre(genre)` : This function reads data from 'movies.json' and 'ratings.csv', filters movies by the specified genre, and returns dictionaries containing movie titles, ratings, and rating counts.
   - `get_top_movies_by_genre(genre)` : This function uses the data from `read_data_for_genre()` to calculate the average ratings for movies in the specified genre and returns the top 5 movies by average rating along with their titles.
2. **Filter Movies by Genre:**
   - The code reads movie data from 'movies.json' and filters movies based on the specified genre. It collects movie titles in the `movie_id_to_title` dictionary.
3. **Calculate Average Ratings:**
   - After filtering movies by genre, the code calculates the average ratings for each movie and stores them in the `movie_ratings` dictionary.
4. **Sort and Return Top Movies:**
   - The code sorts the list of average ratings in descending order and returns the top 5 movies along with their titles.

Usage Example:

To find the top 5 movies in the 'Action' genre with the highest average ratings, you can call the `get_top_movies_by_genre('Action')` function. The code then prints the titles and average ratings of these movies.

Code Output:

When executed, the code produces a list of the top 5 movies within the specified genre, ranked by their average ratings. It displays the movie titles and their corresponding average ratings.

```
import csv
import json

def get_top_movies_by_genre(genre):
    # Initialize lists and dictionaries to store data
    genre_movies = []
    movie_id_to_title = {}
    movie_ratings = {}
    movie_ratings_count = {}
    top_rated_movies = []

    # Collect movie IDs and titles related to the specified genre
    with open('movies.json', 'r') as movies_file:
        for line in movies_file:
            movie_data = json.loads(line)
            movie_id = movie_data['id']
            title = movie_data['title']
            genres = movie_data.get('genres', [])

            if genre in genres:
                genre_movies.append(movie_id)
                movie_id_to_title[movie_id] = title

    # Collect user ratings for movies in the specified genre and count ratings
    with open('ratings.csv', 'r') as ratings_file:
        ratings_file_reader = csv.reader(ratings_file)
        next(ratings_file_reader)  # Skip header row

        for line in ratings_file_reader:
            movie_id = line[1]
            rating = float(line[2])

            if movie_id in genre_movies:
                # Count ratings and movie ratings for movies in the specified genre
                if movie_id in movie_ratings:
                    movie_ratings[movie_id] += rating
                    movie_ratings_count[movie_id] += 1
                else:
                    movie_ratings[movie_id] = rating
                    movie_ratings_count[movie_id] = 1

    # Calculate the average rating for each movie in the genre
    for movie_id in genre_movies:
        if movie_id in movie_ratings:
            avg_rating = round(movie_ratings[movie_id] / movie_ratings_count[movie_id], 2)
            top_rated_movies.append((avg_rating, movie_id))

    # Sort the top-rated movies by average rating in descending order
    top_rated_movies.sort(reverse=True)

    # Return the top 5 movies by average rating along with their titles
    return top_rated_movies[:5], movie_id_to_title

# Example usage: Find top-rated movies in the 'Action' genre
top_movies, movie_id_to_title = get_top_movies_by_genre('Comedy')

# Print the titles and average ratings of the top-rated movies
for movie_rating, movie_id in top_movies:
    movie_title = movie_id_to_title.get(movie_id, "Unknown Title")
    print(movie_title, movie_rating)
```

```
Spice World 5.0
Beyond the Valley of the Dolls 5.0
Vixen! 5.0
Peau d'âne 5.0
Stay Hungry 5.0
```

Question 6: Identifying Top-Rated Movies by Passionate Users

Code Purpose:

The provided Python code answers Question 6 of the coursework. The objective is to identify users passionate about a particular genre and find the top 5 highest-rated movies (on average) among those that received at least 3 ratings from these passionate users. The code reads data from 'movies.json' and 'ratings.csv', filters users passionate about the specified genre, and calculates the average ratings for movies they've rated.

Code Description:

1. **Function Definition**:
    - `passion_project(genre)` : This function takes a genre as input, identifies passionate users and top-rated movies within that genre.
2. **Filter Movies by Genre**:
    - The code reads movie data from 'movies.json' and filters movies based on the specified genre. It collects movie titles in the `movie_id_to_title` dictionary.

3. **Identify Passionate Users**:
   - The code reads rating data from 'ratings.csv' and identifies users who have rated at least 10 movies in the specified genre.
4. **Calculate Average Ratings**:
   - After identifying passionate users, the code calculates the average ratings for movies within the specified genre and stores them in the `movie_ratings` dictionary.
5. **Sort and Return Top Movies**:
   - The code sorts the list of average ratings in descending order and returns the top 5 movies along with their titles.

Usage Example:

To find the top 5 movies in the 'Action' genre with the highest average ratings among passionate users, call the `passion_project('Action')` function. The code then prints the titles and average ratings of these movies.

Code Output:

When executed, the code produces a list of the top 5 movies within the specified genre, ranked by their average ratings among passionate users.

```
import csv
import json

def passion_project(genre):
    # Initialize dictionaries to store data
    user_id_count = {}
    genre_movies = []
    movie_id_to_title = {}
    movie_ratings = {}
    movie_ratings_count = {}
    top_rated_movies = []

    # Identify users who have 10 or more ratings on movies in the specified genre
    with open('movies.json', 'r') as movies_file:
        for line in movies_file:
            movie_data = json.loads(line)
            movie_id = movie_data['id']
            title = movie_data['title']
            genres = movie_data.get('genres', [])

            if genre in genres:
                genre_movies.append(movie_id)
                movie_id_to_title[movie_id] = title

    with open('ratings.csv', 'r') as ratings_file:
        ratings_file_reader = csv.reader(ratings_file)
        next(ratings_file_reader)

        for line in ratings_file_reader:
            user_id = line[0]
            movie_id = line[1]
            rating = float(line[2])

            if movie_id in genre_movies:
                # Count ratings and movie ratings for movies in the specified genre
                if movie_id in movie_ratings:
                    movie_ratings[movie_id] += rating
                    movie_ratings_count[movie_id] += 1
                else:
                    movie_ratings[movie_id] = rating
                    movie_ratings_count[movie_id] = 1
                # Count the number of ratings each user gives for movies in the genre
                if user_id in user_id_count:
                    user_id_count[user_id] += 1
                else:
                    user_id_count[user_id] = 1

    # Find the top 5 highest rated movies among those with at least 3 ratings
    for movie_id in genre_movies:
        if movie_id in movie_ratings and movie_ratings_count[movie_id] >= 3:
            avg_rating = round(movie_ratings[movie_id] / movie_ratings_count[movie_id], 2)
            top_rated_movies.append((avg_rating, movie_id))

    top_rated_movies.sort(reverse=True)

    return top_rated_movies[:5], movie_id_to_title

# Example usage: Find top-rated movies in the 'Comedy' genre
```

```
top_movies, movie_id_to_title = passion_project('Comedy')

for movie_rating, movie_id in top_movies:
    movie_title = movie_id_to_title.get(movie_id, "Unknown Title")
    print(movie_title, movie_rating)
```

```
Stalag 17 4.9
Le couple témoin 4.67
CQ 4.62
Vampire's Kiss 4.6
The General 4.5
```