Set Assignment

1. **Email Duplication Checker**

Problem Statement: Given two lists of email addresses from two different marketing campaigns, find out the emails that are common to both lists to avoid sending duplicate emails.

Sample Input:

```
campaign1 = {"alice@example.com", "bob@example.com", "carol@example.com"}
campaign2 = {"carol@example.com", "dave@example.com", "alice@example.com"}

# Expected Output
common_emails = {"alice@example.com", "carol@example.com"}
```

2. **Movie Genre Classification**

Problem Statement: Given a list of movies and their genres, identify movies that belong to more than one genre.

Sample Input:

```
genres = {
    "Horror": {"The Conjuring", "Get Out", "A Quiet Place"},
    "Thriller": {"Get Out", "Inception", "Gone Girl"},
    "Action": {"Inception", "Mad Max", "Gladiator"}
}

# Expected Output
multi_genre_movies = {"Get Out", "Inception"}
```

3. **Finding Unique Voters**

Problem Statement: In an election system, each district submits a list of voter IDs. Determine the set of unique voter IDs across all districts.

Sample Input:

```
district1 = {"V123", "V456", "V789"}
district2 = {"V234", "V456", "V890"}
district3 = {"V345", "V678", "V789"}

# Expected Output
unique_voters = {"V123", "V456", "V789", "V234", "V890", "V345", "V678"}
```

4. **Identifying Common Ingredients in Recipes**

Problem Statement: Given a set of ingredients for different recipes, find the common ingredients used across all recipes.

Sample Input:

```
recipe1 = {"flour", "sugar", "eggs", "butter"}
recipe2 = {"flour", "eggs", "milk"}
recipe3 = {"flour", "sugar", "eggs"}

# Expected Output
common_ingredients = {"flour", "eggs"}
```

5. **Detecting Plagiarism Between Documents**

Problem Statement: Given two sets of words representing content from two documents, find common words to detect potential plagiarism.

Sample Input:

```
document1 = {"python", "coding", "sets", "data", "structure"}
document2 = {"java", "coding", "data", "algorithms"}

# Expected Output
common_words = {"coding", "data"}
```

6. **User Interest Group Recommendations**

Problem Statement: Given a user's set of interests and multiple interest groups, find out which interest groups the user should join.

Sample Input:

```
user_interests = {"reading", "music", "traveling"}
group1 = {"reading", "writing"}
group2 = {"music", "dancing"}
group3 = {"cooking", "traveling"}

# Expected Output
recommended_groups = {"group1", "group2", "group3"}
```

7. **Inventory Management in Warehouses**

Problem Statement: Given a list of items in two warehouses, find out which items are unique to each warehouse and which items are common to both.

Sample Input:

```
warehouse1 = {"apples", "bananas", "oranges"}
warehouse2 = {"bananas", "grapes", "oranges"}

# Expected Output
common_items = {"bananas", "oranges"}
unique_to_warehouse1 = {"apples"}
unique_to_warehouse2 = {"grapes"}
```

8. **Tracking Borrowed and Returned Books**

Problem Statement: Given sets of borrowed books and returned books, identify which books are still outstanding.

Sample Input:

```
borrowed_books = {"Book1", "Book2", "Book3", "Book4"}
returned_books = {"Book2", "Book3"}

# Expected Output
outstanding_books = {"Book1", "Book4"}
```

9. **Detecting Anomalies in Transaction Records**

Problem Statement: Given a set of legitimate transactions and a set of all transactions, find transactions that might be fraudulent (i.e., not in the legitimate set).

Sample Input:

```
legitimate_transactions = {"T1", "T2", "T3", "T4"}
all_transactions = {"T1", "T2", "T3", "T4", "T5", "T6"}

# Expected Output
fraudulent_transactions = {"T5", "T6"}
```

10. **Shared Friend Recommendations**

Problem Statement: Given two users' friend lists on a social media platform, find mutual friends and suggest them as a potential connection.

Sample Input:

```
user1_friends = {"Alice", "Bob", "Charlie"}
user2_friends = {"Charlie", "David", "Alice"}
```

```
# Expected Output
mutual_friends = {"Alice", "Charlie"}
```