In [2]:

```python
import csv
import random
import string

def generate_random_title():
    words = ['The', 'A', 'An', 'In', 'Of', 'And', 'To', 'With', 'Without', 'On', 'Over', 'Under', 'Beyond']
    title = random.choice(words) + " " + ''.join(random.choice(string.ascii_uppercase) for _ in range(random.randint(3, 7)))
    return title

def generate_random_author():
    first_names = ['John', 'Jane', 'Michael', 'Emily', 'David', 'Sarah', 'Robert', 'Jennifer']
    last_names = ['Smith', 'Johnson', 'Williams', 'Jones', 'Brown', 'Davis', 'Miller', 'Wilson']
    return random.choice(first_names) + " " + random.choice(last_names)

def generate_random_year():
    return random.randint(1900, 2022)

def generate_random_genre():
    genres = ['Fiction', 'Non-Fiction', 'Mystery', 'Science Fiction', 'Fantasy', 'Romance', 'Thriller', 'Biography', 'History']
    return random.choice(genres)

num_books = 1000
book_data = []
for _ in range(num_books):
    title = generate_random_title()
    author = generate_random_author()
    year = generate_random_year()
    genre = generate_random_genre()
    book_data.append([title, author, year, genre])

csv_file = 'books_data.csv'
with open(csv_file, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Title', 'Author', 'Publication Year', 'Genre'])
    writer.writerows(book_data)

print(f'{num_books} books data have been saved to {csv_file}.')
```

```
1000 books data have been saved to books_data.csv.
```

In [2]:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

def load_data(file_path):
    return pd.read_csv(file_path)

def get_author_recommendations(search_author, book_data):
    books_by_author = book_data[book_data['Author'].str.contains(search_author, case=False)]

    if books_by_author.empty:
        return None

    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform(books_by_author['Author'])

    cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

    sim_scores = list(enumerate(cosine_sim[-1]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:11]

    book_indices = [i[0] for i in sim_scores]
    recommendations = books_by_author.iloc[book_indices]
    return recommendations

if __name__ == "__main__":
    csv_file = 'books_data.csv'
    book_data = load_data(csv_file)

    search_author = input("Enter the author name to get book recommendations: ")

    recommendations = get_author_recommendations(search_author, book_data)

    if recommendations is not None:
        print(f"\nTop 10 book recommendations for {search_author}:")
        print(recommendations.to_string(index=False))
    else:
        print(f"No books found for author: {search_author}.")
```

```
Enter the author name to get book recommendations: Kyvalya
No books found for author: Kyvalya.
```

In [2]:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

def load_data(file_path):
    return pd.read_csv(file_path)

def get_author_recommendations(search_author, book_data):
    books_by_author = book_data[book_data['Author'].str.contains(search_author, case=False)]

    if books_by_author.empty:
        return None

    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform(books_by_author['Author'])

    cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

    sim_scores = list(enumerate(cosine_sim[-1]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:11]
    book_indices = [i[0] for i in sim_scores]
    recommendations = books_by_author.iloc[book_indices]
    return recommendations

if __name__ == "__main__":
    csv_file = 'books_data.csv'
    book_data = load_data(csv_file)

    search_author = input("Enter the author name to get book recommendations: ")

    recommendations = get_author_recommendations(search_author, book_data)

    if recommendations is not None:
        print(f"\nTop 10 book recommendations for {search_author}:")
        print(recommendations.to_string(index=False))
    else:
        print(f"No books found for author: {search_author}.")
```

```
Enter the author name to get book recommendations:

Top 10 book recommendations for :
```

Title	Author	Publication Year	Genre
In OQPM	Michael Davis	2018	Biography
Without FFS	Michael Davis	1944	Non-Fiction
A TBRLCB	Michael Davis	1994	Romance
On HHNAET	Michael Davis	1990	Romance
Without JPNDYI	Michael Davis	1939	Fantasy
And BFHU	Michael Davis	2014	Fiction
An EZZOSMG	Michael Davis	1907	Mystery
Of KVNA	Michael Davis	2009	Non-Fiction
And AFN	Michael Davis	1958	Fiction
The SRACE	Michael Davis	1947	Science Fiction

In [ ]: