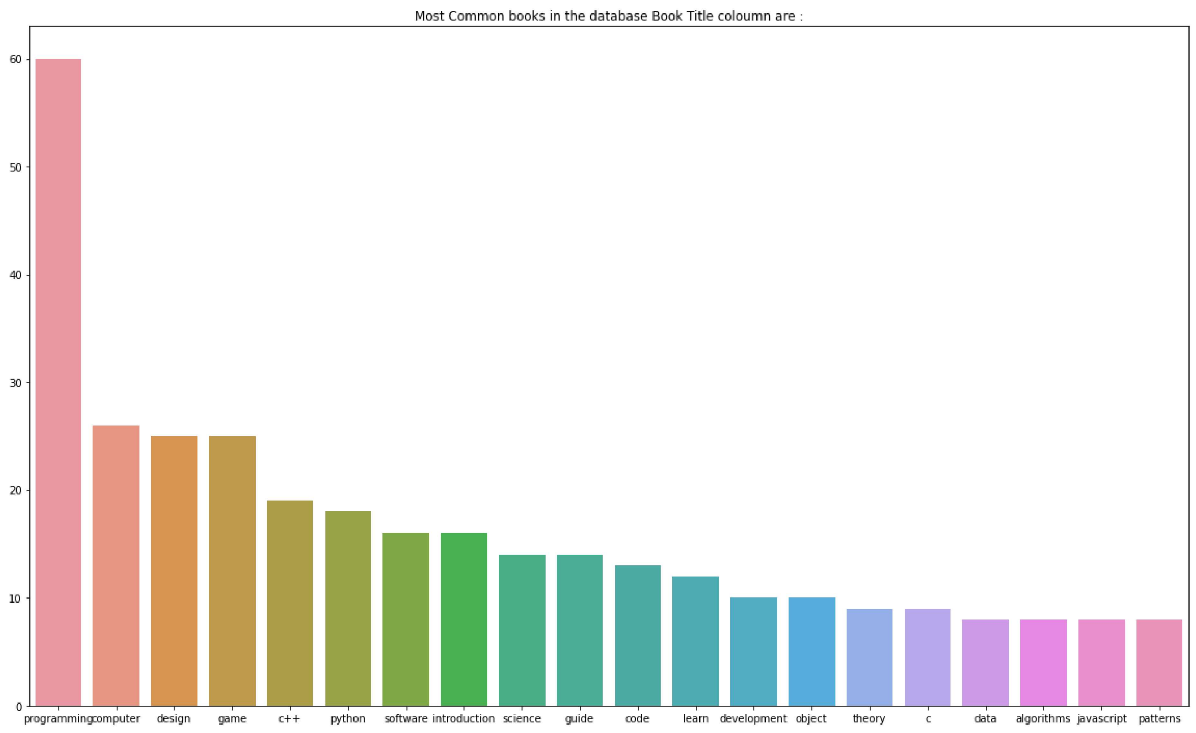
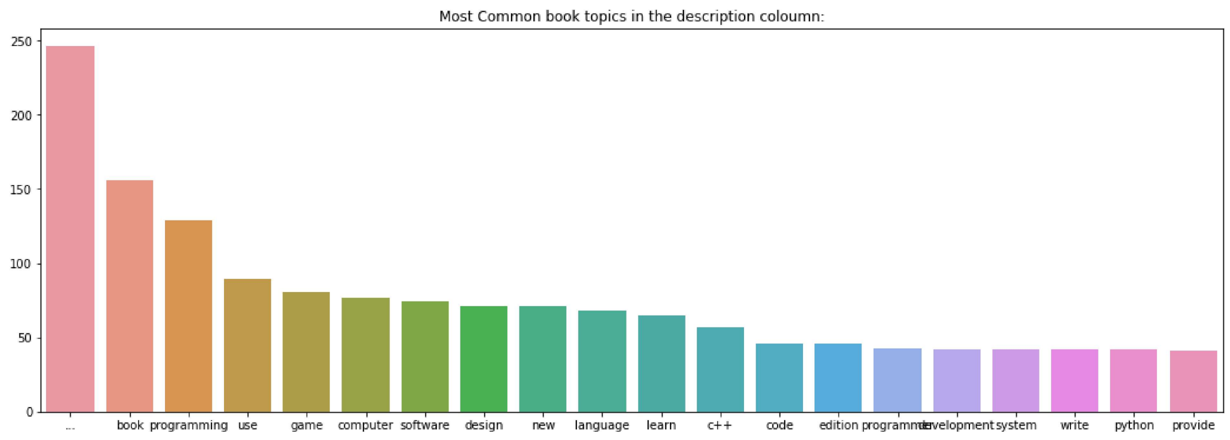
In [8]:
```
import numpy as np
import pandas as pd
import re
import time
from datasketch import MinHash, MinHashLSHForest

# Preprocess will split a string of text into individual tokens/shingles based or
def preprepare(text):
    #text = re.sub(r'[^\w+( \w+)*$]','', text) #bigram
    text = re.sub(r'[^\w\s]','',text) #unigram

    tokens = text.lower()
    tokens = tokens.split()
    return tokens

text = 'My name is RAM'
print('The shingles (tokens) are:', preprepare(text))

#Number of Permutations
permutations = 256


#Number of Recommendations to return
#num_recommendations = 1

def get_forest(data, perms):
    start_time = time.time()

    minhash = []

    for text in data['text']:
        tokens = preprepare(text)
        m = MinHash(num_perm=perms)
        for s in tokens:
            m.update(s.encode('utf8'))
        minhash.append(m)

    forest = MinHashLSHForest(num_perm=perms)

    for i,m in enumerate(minhash):
        forest.add(i,m)

    forest.index()

    print('It took %s seconds to build forest.' %(time.time()-start_time))
    return forest

def predict(text, database, perms, num_results, forest):
    start_time = time.time()

    tokens = preprepare(text)
    m = MinHash(num_perm=perms)
    for s in tokens:
        m.update(s.encode('utf8'))

    idx_array = np.array(forest.query(m, num_results))
    if len(idx_array) == 0:
        return None # if your query is empty, return none

    result = database.iloc[idx_array]['Book_title']

    print('It took %s seconds to query forest.' %(time.time()-start_time))

    return result

db = pd.read_csv(r'C:\Users\Ami\Desktop\Entity Resolution\prog_book.csv')
db['text'] = db['Book_title'] + ' ' + db['Description']
forest = get_forest(db, permutations)


num_recommendations = 20
Book_title = 'Java '           #java art   #Google
result = predict(Book_title, db, permutations, num_recommendations, forest)
print('\n Top Entity Match(es) is(are) \n',  result)
```

```
The shingles (tokens) are: ['my', 'name', 'is', 'ram']
It took 1.3221898078918457 seconds to build forest.
It took 0.009995222091674805 seconds to query forest.

 Top Entity Match(es) is(are)
 98     Release It!: Design and Deploy Production-Read...
71     Learn Java the Easy Way : A Hands-On Introduct...
169                                      Reviewing Java
171    Refactoring: Improving the Design of Existing ...
86                                     Beginning Java 2
87     Learn You a Haskell for Great Good!: A Beginne...
88                    Learn You a Haskell for Great Good!
58          The Principles of Object-Oriented JavaScript
156                                    The Joy of Clojure
Name: Book_title, dtype: object
```

In [7]:
```
import spacy
import pandas as pd
import numpy as np
import seaborn as sns
import base64
import matplotlib.pyplot as plt
import string
from collections import Counter

nlp = spacy.load('en_core_web_sm')
punctuations = string.punctuation
spacy_stopwords = spacy.lang.en.stop_words.STOP_WORDS


def cleanup_text(docs, logging=False):
    texts = []
    counter = 1
    for doc in docs:
        if counter % 1000 == 0 and logging:
            print("Processed %d out of %d documents." % (counter, len(docs)))
        counter += 1
        doc = nlp(doc, disable=['parser', 'ner'])
        tokens = [tok.lemma_.lower().strip() for tok in doc if tok.lemma_ != '-PR
        tokens = [tok for tok in tokens if tok not in spacy_stopwords and tok not
        tokens = ' '.join(tokens)
        texts.append(tokens)
    return pd.Series(texts)


#INFO_text = [text for text in train[train['Conference'] == 'INFOCOM']['Title']]
#IS_text = [text for text in train[train['Conference'] == 'ISCAS']['Title']]

db = pd.read_csv(r'C:\Users\Ami\Desktop\Entity Resolution\prog_book.csv')
db['text'] = db['Book_title'] #+ ' ' + db['Description']
INFO_text =  [text for text in db['text']]
IS_text =  [text for text in db['Description']]

INFO_clean = cleanup_text(INFO_text)
INFO_clean = ' '.join(INFO_clean).split()
IS_clean = cleanup_text(IS_text)
IS_clean = ' '.join(IS_clean).split()
INFO_counts = Counter(INFO_clean)
IS_counts = Counter(IS_clean)
INFO_common_words = [word[0] for word in INFO_counts.most_common(20)]
INFO_common_counts = [word[1] for word in INFO_counts.most_common(20)]

fig = plt.figure(figsize=(20,12))
sns.barplot(x=INFO_common_words, y=INFO_common_counts)
plt.title('Most Common books in the database Book Title coloumn are :')
plt.show()
```



In [6]:
```
IS_common_words = [word[0] for word in IS_counts.most_common(20)]
IS_common_counts = [word[1] for word in IS_counts.most_common(20)]

fig = plt.figure(figsize=(18,6))
sns.barplot(x=IS_common_words, y=IS_common_counts)
plt.title('Most Common book topics in the description coloumn:')
plt.show()
```



In [ ]: