In [ ]:

```
import pandas as pd

prior_order_products = pd.read_csv('order_products__prior.csv')
orders_df = pd.read_csv('orders.csv')
products_df = pd.read_csv('products.csv')



order_product_prior = pd.merge(prior_order_products,
                                    products_df, how='left', on='produc
t_id')

userorder_product_prior = pd.merge(order_product_prior,
                                    orders_df, how='left', on='order_i
d')

userorder_product_prior = userorder_product_prior[:10000000]

products = userorder_product_prior['product_name']
product_nospace = []
for product in products:
    product = str(product)
    product = product.replace(" ", "_")
    product_nospace.append(product)

userorder_product_prior.drop(['product_name'], axis=1)
userorder_product_prior['product_name'] = product_nospace


list_of_names = []
for p_name in userorder_product_prior.groupby('order_id')['product_nam
e']:
    list_of_names.append(' '.join(p_name[1]))

order_id = userorder_product_prior.groupby('order_id')['product_name'].
agg('count').index
```

```
order_products = pd.DataFrame({'order_id':order_id, 'products':list_of_
names})

dataframe_list = []
index = 0
for row in order_products['products']:
    productsName = row.split(' ')
    tup = (index, productsName)
    dataframe_list.append(tup)
    index += 1

"""import random
import numpy
random.shuffle(dataframe_list)

training_data = dataframe_list[:2250411]
testing_data = dataframe_list[2250411:]
"""


import random
import numpy
random.shuffle(dataframe_list)

training_data = dataframe_list[:70001]
testing_data = dataframe_list[70001:]

from pyspark.sql import SparkSession
from pyspark.ml.feature import Word2Vec

spark = SparkSession.builder.appName("Bigram").getOrCreate()


N = len(training_data)//10000
mod = len(training_data) % 10000
train_dataframe = spark.createDataFrame(dataframe_list[0:10000], ['id',
"product_name"])

for i in range(1,N):
```

```
    train_dataframe_sub = spark.createDataFrame(training_data[10000*i:1
0000*(i+1)], ['id',"product_name"])
    traintDF = train_dataframe.union(train_dataframe_sub)

train_dataframe_sub = spark.createDataFrame(training_data[10000*N:len(t
raining_data)], ['id',"product_name"])
train_dataframe = train_dataframe.union(train_dataframe_sub)

from pyspark.ml.feature import NGram

ngram = NGram(n=2, inputCol="product_name", outputCol="bigrams")
ngram_dataframe = ngram.transform(train_dataframe)


from pyspark.ml.feature import NGram


bigrams = ngram_dataframe.toPandas()['bigrams']
tables = {}
total = len(bigrams)
complete = 0
for bigram in bigrams:
    for combination in bigram:
        components = combination.split(' ')
        key = components[0]
        ValueKey = components[1]
        if key in tables:
            Values_Dictionary = tables[key]
            if ValueKey in Values_Dictionary:
                Values_Dictionary[ValueKey] = Values_Dictionary[ValueKe
y] + 1
            else:
                Values_Dictionary[ValueKey] = 1
        else:

            Values_Dictionary =  {ValueKey: 1}
            tables[key] = Values_Dictionary
    complete += 1
```

```
def getting_puredata(Product_Name):


    if Product_Name not in tables:
        return []
    Original_SortedList = sorted(tables[Product_Name].items(), key=lamb
da x: x[1], reverse=True)
    data = {}
    for tp in Original_SortedList:
        product = tp[0]
        num = tp[1]
        if num in data:
            List_of_products = data[num]
            List_of_products.append(product)
        else:
            List_of_products = [product]
        data[num] = List_of_products
    pureData = data.values()
    return list(pureData)

def pickrecommended_products(pureData, numOfRecommend):


    recommended_products = []
    for prods in pureData:
        if len(prods) <= numOfRecommend:
            recommended_products += prods
            numOfRecommend -= len(prods)
        else:
            recommended_products += random.sample(prods, numOfRecommend
)
            numOfRecommend = 0

        if numOfRecommend == 0:
            break

    return recommended_products
```

```
def getRecommend(name, numOfRecommend):



    recommendProducts = []
    Name_of_product = name
    index = 0

    while (numOfRecommend):
        data = getting_puredata(Name_of_product)
        intermediate = pickrecommended_products(data, numOfRecommend)
        recommendProducts += intermediate
        if len(intermediate) == 0 and index == len(recommendProducts):
            break
        numOfRecommend -= len(intermediate)
        if numOfRecommend > 0:
            Name_of_product = recommendProducts[index]
            index += 1


    return recommendProducts


print(getRecommend("Cucumber_Kirby", 5))

def TestScore(testing_data):

    score = []

    for order_info in testing_data:
        thisorder = order_info[1]
        len_order = len(thisorder)
        i = 0
        this_score = 0

        while (i < len_order):
            if thisorder[i] in tables:
                recommendss = getRecommend(thisorder[i], len_order)
                laterProds = thisorder[i:]
```

```
                for prod in laterProds:
                    if prod in recommendss:
                        this_score += 1
                i += 1
            else:
                i += 1
                len_order -= 1

        if not len_order == 0:
            score.append(this_score/len_order)

    return(score)

score = TestScore(testing_data)
print("Mean Test score: ", numpy.mean(score))
```