```python
# -*- coding: utf-8 -*-
"""
Description:
Apriori Implementation on retail and movie datasets

@title: Apriori
@author: Adwan Salahuddin Syed, 100525709
@email: adwan.syed@uoit.net
"""

# import libraries
import time
import itertools

def apriori(baskets, tHold, start):
    """
    apriori (baskets, tHold, start)

    Implementation of Apriori. Generates candidates and adds count.
    Tests count against threshold to generate frequent item sets

    @arg baskets   Basket with sample of dataset
    @arg tHold    Minimum support threshold
    @arg start    Timestamp used to calculate runtimes

    @return    Generate and output frequent item sets with runtimes
    """

    # Generate candidate itemset C1 of size k
    C1 = {}
    for transaction in baskets:
        for item in transaction.split():
            if item not in C1:
                C1[item] = 1
            else:
                C1[item] += 1

    # Generate frequent itemset L1 from C1
    L1 = {}
    for key in C1:
        if C1[key] > (len(baskets) * tHold):
            L1[key] = C1[key]

    # Display L1
    print("L1")
    print(L1)
    print(" ")

    L = [L1] # to store frequent itemsets
    k = 0 # counter for rounds of algorithm

    # Loop generating candidates and frequent itemsets of Ck+1 and Lk+1
    while (len(L[k]) > 0):

        # Generate candidates of frequent items
        Ck ={}
        unique = [] # Store unique items
        for key in L[k]:
            for item in key.split(','):
                if item not in unique:
                    unique.append(item)
        # All possible combinations in sorted order
        for key in itertools.combinations(unique, k+2):
```

```
            if key not in Ck:
                Ck[','.join(key)] = 0 # Join items to create candidate set

        # Display timestamp
        print (" ")
        print(time.time()-start, ': Generated candidates of size', k+2)

        # Counting candidates and adding frequency
        for transactions in baskets:
            # All possible orderings with no repeated elements
            for subset in set(itertools.permutations(transactions.split(),(k+2))):
                candidate = ','.join(subset) # tuple to string
                for keys in Ck:
                    # Comparing two strings, ex: '4,2' == '2,4'
                    if candidate == keys:
                        Ck[keys] += 1 # Add to count

        # Display timestamp
        print (" ")
        print(time.time()-start, ': Counted candidates of size', k+2)

        # Create frequent itemset from candidate set by checking min. support
        Lk = {}
        for key in Ck:
            if Ck[key] > (len(baskets) * tHold):
                Lk[key] = Ck[key]

        # Display timestamp and frequent item sets
        print (" ")
        print(time.time()-start, ': Frequent itemsets of size', k+2)
        print ("L%s with count"%(k+2))
        print (Lk)
        print (" ")

        L.append(Lk) # append to list storing all item sets

        k += 1 # update counter

#------------------------------------------------------------------------------

def runRetail(tHold):
    """
    runRetail(tHold)

    This function runs apriori algorithm on a movie dataset to find frequent item sets
    Transactions: 1382
    Baskets: 9 (10k increments)

    @arg tHold      Minimum support threshold

    @return Timestamp for completion of each basket
    """

    # dataset in use
    dataset_file = 'retail.dat'

    # declaration of baskets
    basket1 = []
    basket2 = []
    basket3 = []
    basket4 = []
    basket5 = []
    basket6 = []
```

```
    basket7 = []
    basket8 = []
    basket9 = []

    # appending transactions in 10k increments to baskets
    count = 0 # Counter
    with open(dataset_file) as file:
        for line in file:
            basket1.append(line)
            if count > 10000:
                basket2.append(line)
            if count > 20000:
                basket3.append(line)
            if count > 30000:
                basket4.append(line)
            if count > 40000:
                basket5.append(line)
            if count > 50000:
                basket6.append(line)
            if count > 60000:
                basket7.append(line)
            if count > 70000:
                basket8.append(line)
            if count > 80000:
                basket9.append(line)
            if count > 88162:
                break
            count += 1

    #------------------------------------------------------------------------
    # Running Apriori algorithm  on each basket and returning timestamps

    # Start the clock
    start = time.time()
    apriori(basket1, tHold, start)
    end = time.time()
    print('Time taken in seconds for basket 1:', end - start)

    # Start the clock
    start = time.time()
    apriori(basket2, tHold, start)
    end = time.time()
    print('Time taken in seconds for basket 2:', end - start)

    # Start the clock
    start = time.time()
    apriori(basket3, tHold, start)
    end = time.time()
    print('Time taken in seconds for basket 3:', end - start)

    # Start the clock
    start = time.time()
    apriori(basket4, tHold, start)
    end = time.time()
    print('Time taken in seconds for basket 4:', end - start)

    # Start the clock
    start = time.time()
    apriori(basket5, tHold, start)
    end = time.time()
    print('Time taken in seconds for basket 5:', end - start)

    # Start the clock
```

```
    start = time.time()
    apriori(basket6, tHold, start)
    end = time.time()
    print('Time taken in seconds for basket 6:', end - start)

    # Start the clock
    start = time.time()
    apriori(basket7, tHold, start)
    end = time.time()
    print('Time taken in seconds for basket 7:', end - start)

    # Start the clock
    start = time.time()
    apriori(basket8, tHold, start)
    end = time.time()
    print('Time taken in seconds for basket 8:', end - start)

    # Start the clock
    start = time.time()
    apriori(basket9, tHold, start)
    end = time.time()
    print('Time taken in seconds for basket 9:', end - start)

#------------------------------------------------------------------------------

def runMovie(tHold):
    """
    runMovie(tHold)

    This function runs apriori algorithm on a movie dataset to find frequent item sets
    Transactions: 1382
    Baskets: 1

    @arg tHold      Minimum support threshold

    @return Timestamp for apriori algorithm mining movies dataset
    """
    dataset_file = 'movies.dat'
    basket = []

    count = 0
    with open(dataset_file) as file:
        for line in file:
            basket.append(line)
            if count > 1382:
                break
            count += 1

    # Start the clock
    start = time.time()
    apriori(basket, tHold, start)
    end = time.time()
    print('Time taken in seconds for basket:', end - start)

"""
INSTRUCTIONS: Uncomment either runRetail(tHold) OR runMovie(tHold) one at a time
and insert minimum support threshold you wish to test for input parameter tHold.
"""

#runRetail(0.13)
runMovie(0.03)
```