Assignment2

1) The frequent itemsets you obtain on Gene dataset (L1, L2, L3).

L1:

{gene_1}: sup = 0.83
{gene_12}: sup = 0.54
{gene_17}: sup = 0.55
{gene_21}: sup = 0.62
{gene_22}: sup = 0.55
{gene_23}: sup = 0.54
{gene_3}: sup = 0.71
{gene_39}: sup = 0.51
{gene_4}: sup = 0.5
{gene_45}: sup = 0.58
{gene_48}: sup = 0.57
{gene_55}: sup = 0.55
{gene_59}: sup = 0.76
{gene_6}: sup = 0.66
{gene_63}: sup = 0.5
{gene_66}: sup = 0.59
{gene_72}: sup = 0.74
{gene_77}: sup = 0.58
{gene_83}: sup = 0.5
{gene_84}: sup = 0.54
{gene_93}: sup = 0.53
{gene_99}: sup = 0.56
{gene_14}: sup = 0.52
{gene_26}: sup = 0.52
{gene_27}: sup = 0.51
{gene_36}: sup = 0.61
{gene_37}: sup = 0.56
{gene_47}: sup = 0.66

{gene_5}: sup = 0.73
{gene_50}: sup = 0.5
{gene_54}: sup = 0.67
{gene_56}: sup = 0.51
{gene_60}: sup = 0.54
{gene_75}: sup = 0.57
{gene_78}: sup = 0.59
{gene_81}: sup = 0.58
{gene_87}: sup = 0.67
{gene_89}: sup = 0.59
{gene_25}: sup = 0.57
{gene_43}: sup = 0.5
{gene_53}: sup = 0.5
{gene_71}: sup = 0.58
{gene_8}: sup = 0.66
{gene_9}: sup = 0.5
{gene_90}: sup = 0.52
{gene_91}: sup = 0.65
{gene_98}: sup = 0.51
{gene_31}: sup = 0.51
{gene_94}: sup = 0.62
{gene_64}: sup = 0.5
{gene_67}: sup = 0.62

L2:

{gene_21, gene_1}: sup = 0.53
{gene_1, gene_3}: sup = 0.63
{gene_59, gene_1}: sup = 0.62
{gene_1, gene_6}: sup = 0.59
{gene_1, gene_72}: sup = 0.61
{gene_84, gene_1}: sup = 0.5
{gene_59, gene_3}: sup = 0.56
{gene_3, gene_72}: sup = 0.53
{gene_59, gene_6}: sup = 0.51
{gene_59, gene_72}: sup = 0.62

{gene_47, gene_1}： sup = 0.59

{gene_5, gene_1}： sup = 0.65

{gene_54, gene_1}： sup = 0.58

{gene_81, gene_1}： sup = 0.51

{gene_87, gene_1}： sup = 0.56

{gene_89, gene_1}： sup = 0.52

{gene_47, gene_3}： sup = 0.5

{gene_5, gene_3}： sup = 0.59

{gene_59, gene_5}： sup = 0.51

{gene_59, gene_87}： sup = 0.51

{gene_5, gene_72}： sup = 0.51

{gene_5, gene_47}： sup = 0.53

{gene_5, gene_87}： sup = 0.51

{gene_1, gene_8}： sup = 0.53

{gene_91, gene_1}： sup = 0.55

{gene_5, gene_6}： sup = 0.52

{gene_91, gene_5}： sup = 0.5

{gene_94, gene_1}： sup = 0.54

{gene_67, gene_1}： sup = 0.55

L3:

{gene_59, gene_1, gene_72}： sup = 0.5

{gene_5, gene_1, gene_3}： sup = 0.52

2) The length-3 candidate itemsets generated during Apriori (C3) on Gene dataset.

{gene_1, gene_3, gene_59}： sup = 0.48

{gene_1, gene_3, gene_72}： sup = 0.46

{gene_1, gene_6, gene_59}： sup = 0.45

{gene_1, gene_72, gene_59}： sup = 0.5

{gene_3, gene_72, gene_59}： sup = 0.47

{gene_1, gene_3, gene_47}： sup = 0.45

{gene_1, gene_3, gene_5}： sup = 0.52

{gene_1, gene_59, gene_5}： sup = 0.44

{gene_1, gene_59, gene_87}： sup = 0.41

{gene_1, gene_72, gene_5}: sup = 0.45
{gene_3, gene_59, gene_5}: sup = 0.44
{gene_3, gene_72, gene_5}: sup = 0.42
{gene_1, gene_5, gene_47}: sup = 0.48
{gene_1, gene_5, gene_87}: sup = 0.45
{gene_3, gene_5, gene_47}: sup = 0.42
{gene_59, gene_72, gene_5}: sup = 0.4
{gene_59, gene_5, gene_87}: sup = 0.35
{gene_1, gene_6, gene_5}: sup = 0.48
{gene_1, gene_5, gene_91}: sup = 0.45
{gene_59, gene_6, gene_5}: sup = 0.37

3) The codes of the two functions: *apriori_gen* and *get_freq*.

apriori_gen:

```
def apriori_gen(freq_sets, k):
    """Generates candidate itemsets (via the F_k-1 x F_k-1 method).

    This part generates new candidate k-itemsets based on the frequent
    (k-1)-itemsets found in the previous iteration.

    The apriori_gen function performs two operations:
    (1) Generate length k candidate itemsets from length k-1 frequent itemsets
    (2) Prune candidate itemsets containing subsets of length k-1 that are infrequent

    Parameters
    ----------
    freq_sets : list
        The list of frequent (k-1)-itemsets.

    k : integer
        The cardinality of the current itemsets being evaluated.
    Returns
    -------
    candidate_list : list
        The list of candidate itemsets.
    """
    # TODO
    candidate_list = []
    # print(freq_sets) ## [frozenset({'Key-chain'}), frozenset({'Mango'}), frozenset({'Yo-yo'}), frozenset({'Eggs'}), frozenset({'Onion'})]

    # Generate length k candidate itemsets from length k-1 frequent itemsets (F_k-1 x F_k-1)
    for i in range(len(freq_sets)):
        for j in range(i+1, len(freq_sets)):
            first = list(freq_sets[i])
            second = list(freq_sets[j])
            # print(first)
            # print(second)

            # sort two list to ensure the order of items in the list is the same
            first.sort()
            second.sort()

            # if the first k-2 items are the same
            if first[:k-2] == second[:k-2]:
                first.append(second[-1])
                candidate_list.append(first)
            #    print(first)
            # print("-----\n")

    # print("candidate_list:")
    # print(candidate_list) ## [['Key-chain', 'Mango'], ['Key-chain', 'Yo-yo'], ['Key-chain', 'Eggs'], ['Key-chain', 'Onion'], ['Mango', 'Yo-yo'], ['Mango', 'Eggs'], ['Mango', 'Onion'], ['Yo-yo', 'Eggs'], ['Yo-yo', 'Onion'], ['Eggs', 'Onion']]
    # Prune candidate itemsets containing subsets of length k-1 that are infrequent
    pruned_candidate_list = []
    freq_sets_list = []
    for s in freq_sets:
        s=list(s)
        s.sort()
        freq_sets_list.append(s)
    # print(freq_sets_list)

    for candidate in candidate_list:
        all_possible_candidate_subsets = combinations(candidate, k-1)

        passed = True

        # check if all the subsets of the candidate are in freq_sets_list
        for subset in all_possible_candidate_subsets:

            sorted_subset = list(subset)
            sorted_subset.sort()
            if (sorted_subset in freq_sets_list) and (sorted_subset not in pruned_candidate_list):
                continue
            else:
                passed = False
                break
        if passed:
            pruned_candidate_list.append(candidate)



    # turn list of frozensets
    pruned_candidate_list = list(map(frozenset, pruned_candidate_list))
    # print("pruned_candidate_list:")
    # print(pruned_candidate_list, "\n\n") ##
    return pruned_candidate_list
```

get_freq:

```
def get_freq(dataset, candidates, min_support, verbose=False):
    """

    This function separates the candidates itemsets into frequent itemset and in frequent itemsets based on the min_support,
    and returns all candidate itemsets that meet a minimum support threshold.

    Parameters
    ----------
    dataset : list
        The dataset (a list of transactions) from which to generate candidate
        itemsets.

    candidates : frozenset
        The list of candidate itemsets.

    min_support : float
        The minimum support threshold.

    Returns
    -------
    freq_list : list
        The list of frequent itemsets.

    support_data : dict
        The support data for all candidate itemsets.
    """

    # TODO
    freq_list = []
    support_data = {}

    # get support count for each candidate
    for transaction in dataset:
        for candidate in candidates:
            if candidate.issubset(transaction):
                if candidate in support_data:
                    support_data[candidate] += 1
                else:
                    # use update method to add new candidate to support_data
                    support_data.update({candidate: 1})

    # After getting the support count, calculate the support and filter out the infrequent itemsets
    for key in support_data:
        # support = (support count) / (total number of transactions)
        support_data[key] /= len(dataset)
        if support_data[key] >= min_support:
            freq_list.append(key)

    # print("support_data:")
    # print(support_data) ## {frozenset({'Corn'}): 0.4, frozenset({'Ice-cream'}): 0.4, frozenset({'Key-chain'}): 0.8, frozenset({'Mango'}): 0.8, frozenset({'Umbrella'}): 0.4, frozenset({'Yo-yo'}): 0.6, frozenset({'Doll'}): 0.4, frozenset({'Eggs'}): 0.6, frozenset({'Onion'}): 0.8, frozenset({'Apple'}): 0.2, frozenset({'Nintendo'}): 0.2}
    # print("frequent itemsets:")
    # print(freq_list) ## [frozenset({'Key-chain'}), frozenset({'Mango'}), frozenset({'Yo-yo'}), frozenset({'Eggs'}), frozenset({'Onion'})]


    return freq_list, support_data
```

4) AI usage disclosure statement:

Did you use any AI tools (such as ChatGPT,Microsoft CoPilot, or similar) in completing this assignment?

No, I tried to finish this on my own.