```python
from itertools import chain, combinations
from collections import defaultdict

def load_dataset():
    dataset = [
        {'A', 'B', 'D'},
        {'B', 'C', 'E'},
        {'A', 'B', 'D', 'E'},
        {'A', 'E'},
        {'B', 'D', 'E'}
    ]
    return dataset

def generate_candidates(itemsets, k):
    return set(i.union(j) for i in itemsets for j in itemsets if len(i.union(j))

def get_frequent_itemsets(dataset, min_support):
    itemsets = [frozenset([item]) for item in set(chain.from_iterable(dataset))]

    def get_support_counts(candidate_itemsets):
        support_counts = defaultdict(int)
        for transaction in dataset:
            for itemset in candidate_itemsets:
                if itemset.issubset(transaction):
                    support_counts[itemset] += 1
        return {itemset: support / len(dataset) for itemset, support in support_

    frequent_itemsets = []
    k = 1

    while itemsets:
        candidate_itemsets = generate_candidates(itemsets, k)
        support_counts = get_support_counts(candidate_itemsets)
        frequent_itemsets.extend((itemset, support) for itemset, support in supp
        itemsets = generate_candidates(set(itemset for itemset, _ in support_cou
        k += 1

    return frequent_itemsets

if __name__ == "__main__":
    dataset = load_dataset()
    min_support = 0.4

    frequent_itemsets = get_frequent_itemsets(dataset, min_support)

    print("Frequent Itemsets:")
    for itemset, support in frequent_itemsets:
        print(f"{itemset}: {support}")
```

```
Frequent Itemsets:
frozenset({'A'}): 0.6
frozenset({'D'}): 0.6
frozenset({'B'}): 0.8
frozenset({'E'}): 0.8
frozenset({'D', 'B'}): 0.6
frozenset({'B', 'A'}): 0.4
frozenset({'D', 'A'}): 0.4
frozenset({'B', 'E'}): 0.6
frozenset({'E', 'A'}): 0.4
frozenset({'D', 'E'}): 0.4
frozenset({'D', 'A', 'B'}): 0.4
frozenset({'D', 'E', 'B'}): 0.4
```