Data Mining Lab-4(FP-Growth Algo)

Dataset Used:

Grocery Store Dataset used

Link: https://drive.google.com/file/d/1Wd9Q2xTd6AYN3v5Ao4fOje6VrRHC_7oN/view?usp=drive_link

Source Code:

```
!pip install mlxtend --quiet
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori,association_rules
te=TransactionEncoder()
te_ary=te.fit(grocery).transform(grocery)
df2=pd.DataFrame(te_ary,columns=te.columns_)
display(df2)
from mlxtend.frequent_patterns import fpgrowth

fpgrowth(df2, min_support=0.02)
freq_data=fpgrowth(df2,min_support=0.020,use_colnames=True)
freq_data.head(10)
from collections import defaultdict
import graphviz

class FPNode:
    def __init__(self, item, count, parent):
        self.item = item
        self.count = count
        self.parent = parent
        self.children = {}

class FPTree:
    def __init__(self, transactions, min_support, top_n=None):
        self.min_support = min_support
        self.top_n = top_n
        self.headers = {}
        self.root = self.build_tree(transactions)

    def build_tree(self, transactions):
        item_counts = defaultdict(int)
        for t in transactions:
            for item in t:
                item_counts[item] += 1
        item_counts = {k: v for k, v in item_counts.items() if v >= self.min_support}


        if self.top_n:
            top_items = sorted(item_counts.items(), key=lambda x: x[1], reverse=True)[:self.top_n]
            allowed = set([i for i, _ in top_items])
            item_counts = {k: v for k, v in item_counts.items() if k in allowed}

            transactions = [[i for i in t if i in allowed] for t in transactions]


        root = FPNode("null", 1, None)
        for t in transactions:
            t = [i for i in t if i in item_counts]
            t.sort(key=lambda i: item_counts[i], reverse=True)
            self.insert_tree(t, root)
        return root

    def insert_tree(self, items, node):
        if not items: return
        first = items[0]
        if first in node.children:
            node.children[first].count += 1
        else:
            node.children[first] = FPNode(first, 1, node)
        self.insert_tree(items[1:], node.children[first])


def draw_tree(node, graph, parent_name=None):
    node_name = f"{id(node)}_{node.item}({node.count})"

    if node.item != "null":
        graph.node(node_name, label=f"{node.item}\n({node.count})")

    if parent_name:
        graph.edge(parent_name, node_name)

    for child_item, child_node in node.children.items():
        draw_tree(child_node, graph, node_name)
```

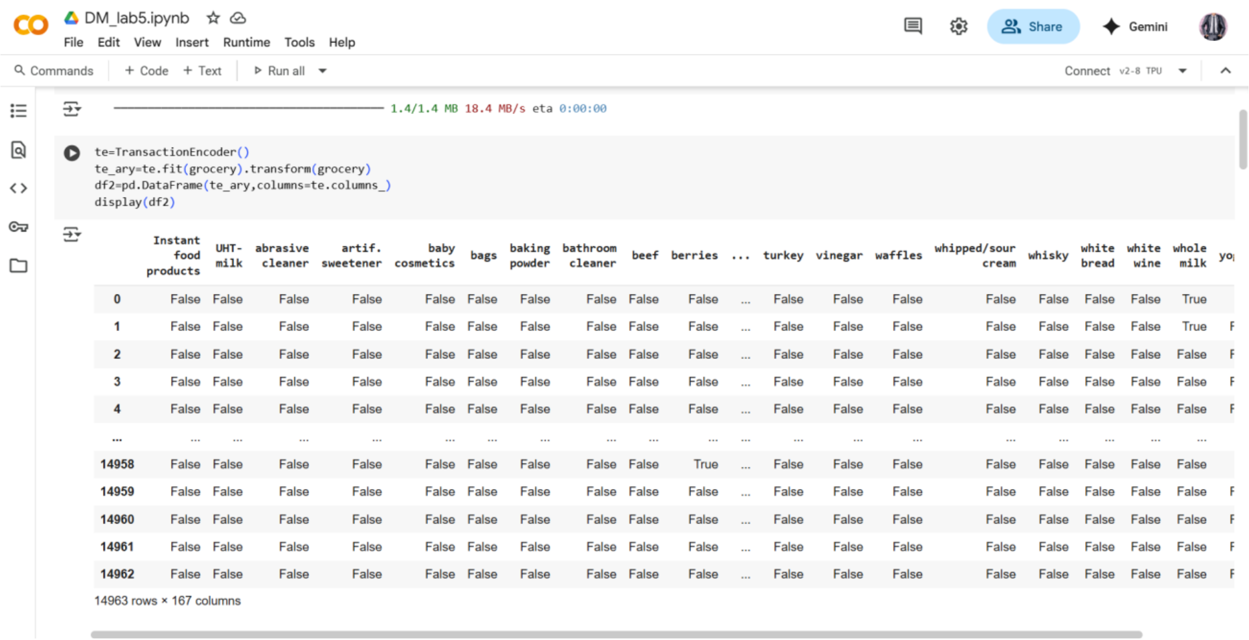
```
fp_tree = FPTree(grocery, min_support=1500)


dot = graphviz.Digraph(comment='FP-Tree')


draw_tree(fp_tree.root, dot)

display(dot)
```

Screenshots:

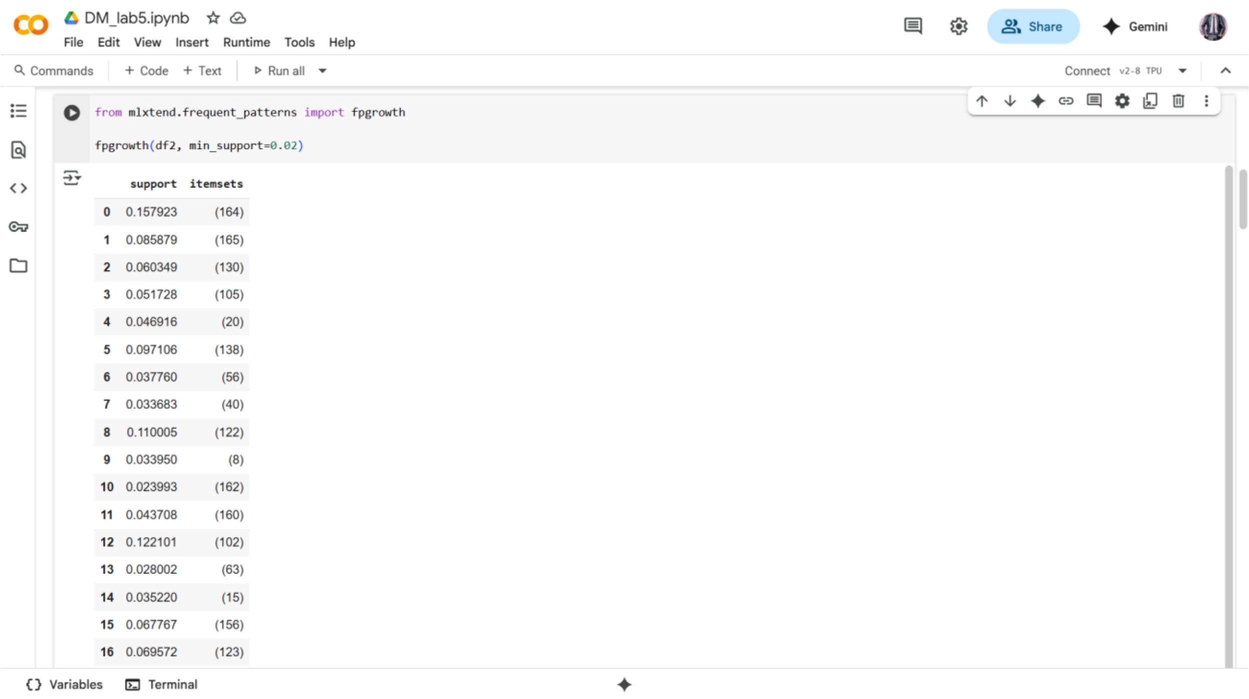


```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 18.4 MB/s eta 0:00:00
```

```
te=TransactionEncoder()
te_ary=te.fit(grocery).transform(grocery)
df2=pd.DataFrame(te_ary,columns=te.columns_)
display(df2)
```

	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	bags	baking powder	bathroom cleaner	beef	berries	...	turkey	vinegar	waffles	whipped/sour cream	whisky	white bread	white wine	whole milk	yo
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	True	
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	True	F
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
14958	False	False	False	False	False	False	False	False	False	True	...	False	False	False	False	False	False	False	False	
14959	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	F
14960	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	F
14961	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	F
14962	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	F

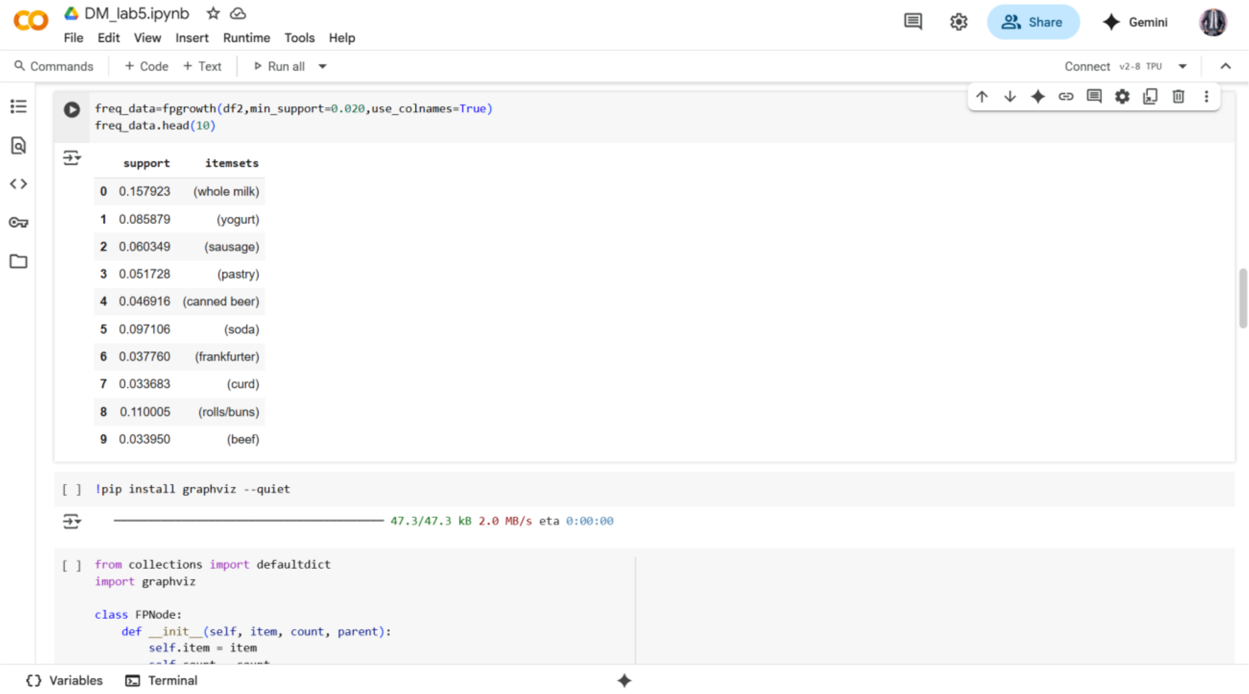
14963 rows × 167 columns



```
from mlxtend.frequent_patterns import fpgrowth

fpgrowth(df2, min_support=0.02)
```

	support	itemsets
0	0.157923	(164)
1	0.085879	(165)
2	0.060349	(130)
3	0.051728	(105)
4	0.046916	(20)
5	0.097106	(138)
6	0.037760	(56)
7	0.033683	(40)
8	0.110005	(122)
9	0.033950	(8)
10	0.023993	(162)
11	0.043708	(160)
12	0.122101	(102)
13	0.028002	(63)
14	0.035220	(15)
15	0.067767	(156)
16	0.069572	(123)



```
freq_data=fpgrowth(df2,min_support=0.020,use_colnames=True)
freq_data.head(10)
```

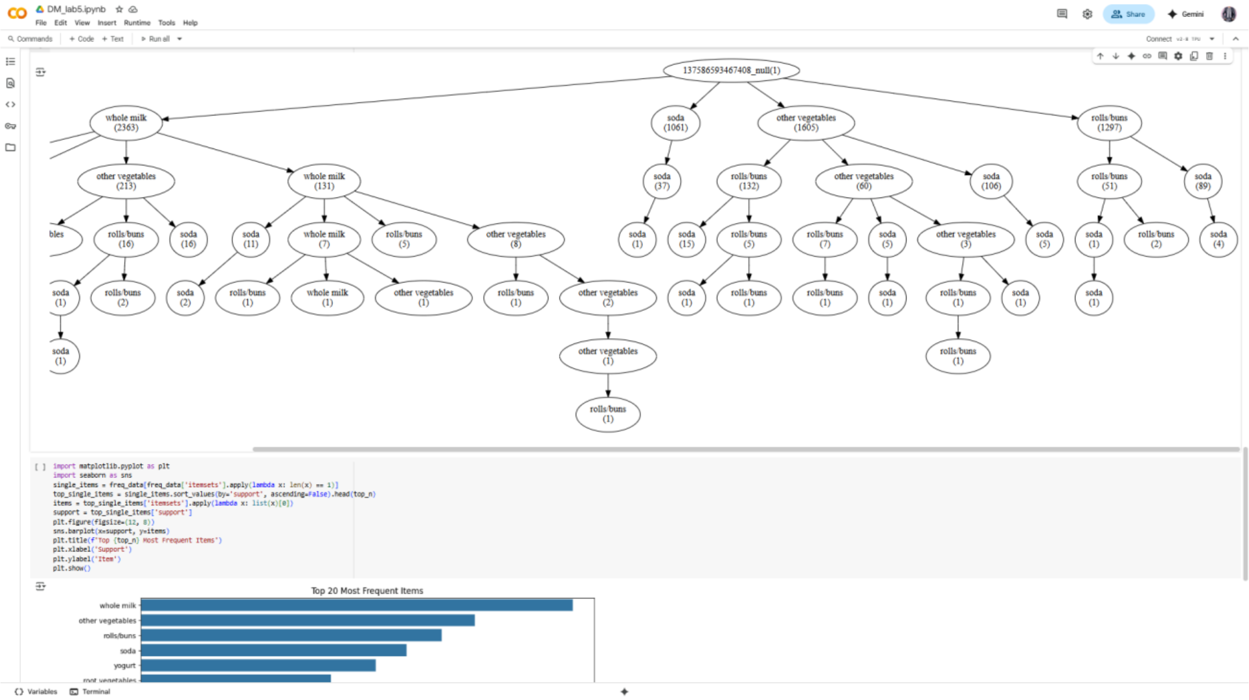
	support	itemsets
0	0.157923	(whole milk)
1	0.085879	(yogurt)
2	0.060349	(sausage)
3	0.051728	(pastry)
4	0.046916	(canned beer)
5	0.097106	(soda)
6	0.037760	(frankfurter)
7	0.033683	(curd)
8	0.110005	(rolls/buns)
9	0.033950	(beef)

```
[ ] !pip install graphviz --quiet
```

```
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.3/47.3 kB 2.0 MB/s eta 0:00:00
```

```
[ ] from collections import defaultdict
    import graphviz

    class FPNode:
        def __init__(self, item, count, parent):
            self.item = item
```



```
[ ] import matplotlib.pyplot as plt
    import seaborn as sns
    single_items = freq_data[freq_data['itemsets'].apply(lambda x: len(x) == 1)]
    top_single_items = single_items.sort_values(by='support', ascending=False).head(top_n)
    items = top_single_items['itemsets'].apply(lambda x: list(x)[0])
    support = top_single_items['support']
    plt.figure(figsize=(12, 8))
    sns.barplot(x=support, y=items)
    plt.title(f'Top {top_n} Most Frequent Items')
    plt.xlabel('Support')
    plt.ylabel('Item')
    plt.show()
```

Observations:

- **Most Frequent Items**

The top-level branches of the FP-Tree confirm the most commonly purchased products (e.g., *whole milk, bread, rolls*).

- **Common Co-occurrences**

Child branches indicate strong product pairings.

For example: If the tree shows `milk → yogurt → rolls`, it means these three items often occur together in baskets.

- **Compact Representation**

The FP-Tree reduces thousands of transactions into a compressed structure while still preserving item relationships.

- **Differences from Apriori**

Unlike Apriori, which scans the dataset multiple times, FP-Growth uses the FP-Tree to mine itemsets more efficiently.

This means it works better with large datasets like groceries.

- **Business Insight**

The branches in the FP-Tree can directly suggest **bundling opportunities**.

For example, if *milk* → *bread* → *butter* is a frequent path, stores could place these items close together or offer combo discounts