



## Knowledge Discovery in Databases with Exercises Summer Semester 2025

# Exercise Sheet 3: Frequent Patterns

### About this Exercise Sheet

This exercise sheet focuses on the content of lecture 6. *Mining Frequent Patterns, Associations and Correlations*.

It includes both a practical data science exercise (Exercise 1) and theoretical exercises on Apriori (Exercise 2) and FP-growth (Exercise 3).

The exercise sheet is designed for a two-week period, during which the tasks can be completed flexibly (Exercise 1 is planned for the first exercise session, and Exercises 2 and 3 for the second session).

The sample solution will be published after the two weeks have elapsed.

### Preparation

Before participating in the exercise, you must prepare the following:

#### 1. Install Python and pip on your computer

- Detailed instructions can be found in 1-Introduction-Python-Pandas.pdf.

#### 2. Download provided additional files

- Download Additional-Files-Student.zip from StudOn
- Extract it to a folder of your choice.

#### 3. Install required Python packages

- Open a terminal and navigate to the folder where you extracted the files.
- Run the command `pip install -r requirements.txt` within the extracted additional files folder to install the required Python packages.

## Exercise 1: Mining Frequent Patterns

This exercise comprises practical data science tasks and thus utilizes a Jupyter Notebook:

1. Open `Mining-Frequent-Patterns.ipynb`.
2. Take a look at the tasks (blue boxes) in the notebook and try to solve them.

If you are unfamiliar with how to open a Jupyter Notebook, please refer to Exercise 1 of `1-Introduction-Python-Pandas.pdf`.

The solution to the exercise can be found in `Additional-Files-Solution.zip`.

## Exercise 2: Apriori

Given is a **transactional dataset**:

ID	Transaction
1	Apple, Banana, Cherry
2	Banana, Cherry
3	Cherry, Apple
4	Dragonfruit, Apple, Banana
5	Apple, Dragonfruit

Use **Apriori** to find all frequent itemsets for a **minimum support count** of **2**.

Write down **all** intermediate steps.

### 1. Count the occurrences of each 1-itemset:

Each item that occurs in the dataset is a 1-itemset:

- Apple: 4
- Banana: 3
- Cherry: 3
- Dragonfruit: 2

### 2. Prune non-frequent 1-itemsets:

All 1-itemsets have a support count of at least 2. Therefore, all 1-itemsets are frequent.

### 3. Generate length-2 candidate itemsets:

The candidate itemsets are generated by combining all the frequent 1-itemsets:

- Apple, Banana
- Apple, Cherry
- Apple, Dragonfruit
- Banana, Cherry
- Banana, Dragonfruit
- Cherry, Dragonfruit

4. **Count the occurrences of each length-2 candidate itemset:**

- Apple, Banana: 2
- Apple, Cherry: 2
- Apple, Dragonfruit: 2
- Banana, Cherry: 2
- Banana, Dragonfruit: 1
- Cherry, Dragonfruit: 0

5. **Prune non-frequent length-2 candidate itemsets:**

The length-2 candidate itemsets that have a support count of at least 2 are:

- Apple, Banana
- Apple, Cherry
- Apple, Dragonfruit
- Banana, Cherry

6. **Generate length-3 candidate itemsets:**

The candidate itemsets are generated by combining all the frequent length-2 itemsets:

- Apple, Banana, Cherry

This length-3 itemset contains the frequent length-2 itemsets „Apple, Banana“ and „Banana, Cherry“, and „Apple, Cherry“ and is the only valid length-3 candidate.

**Common Mistake:** A common mistake is that „Apple, Banana, Dragonfruit“, „Apple, Cherry, Dragonfruit“, and „Banana, Cherry, Dragonfruit“ are generated as length-3 candidates. These 3-itemsets each contain at least one non-frequent 2-itemset (e.g. „Apple, Banana, Dragonfruit“ contains „Banana, Dragonfruit“) and are therefore not valid length-3 candidates.

7. **Count the occurrences of the length-3 candidate itemset:**

- Apple, Banana, Cherry: 1

8. **Prune non-frequent length-3 candidate itemsets:**

„Apple, Banana, Cherry“ has a support count of 1, which is below the minimum support count of 2. Therefore, there are no frequent length-3 itemsets.

9. **Generate length-4 candidate itemsets:**

There are no frequent length-3 itemsets, so there are no valid length-4 candidates.

10. **Termination:**

The algorithm terminates because there are no length-4 candidates.

**Result:**

The frequent itemsets for a minimum support count of 2 are:

- |           |             |                  |                    |
|-----------|-------------|------------------|--------------------|
| 1. Apple  | 3. Cherry   | 5. Apple, Banana | 7. Apple, Dragonf. |
| 2. Banana | 4. Dragonf. | 6. Apple, Cherry | 8. Banana, Cherry  |

## Exercise 3: FP-growth

Given is a **transactional dataset**:

ID	Transaction
1	Apple, Banana
2	Banana, Cherry
3	Cherry, Apple
4	Apple, Banana
5	Apple, Dragonfruit

Use **FP-growth** to find all frequent itemsets for a **minimum support count** of **2**.

Write down **all** intermediate steps. This **includes** the header table for each FP-tree.

### 1. Count the occurrences of each 1-itemset:

Each item that occurs in the dataset is a 1-itemset:

- Apple: 4
- Banana: 3
- Cherry: 2
- Dragonfruit: 1

### 2. Prune non-frequent 1-itemsets:

The 1-itemsets that have a support count of at least 2 are:

- Apple: 4
- Banana: 3
- Cherry: 2

### 3. Create the f-list for our dataset:

The f-list is created by sorting the 1-itemsets in descending order of their support count:

- Apple → Banana → Cherry

### 4. Order the items in the transactions according to the f-list:

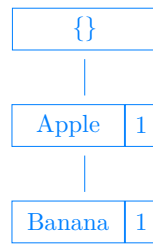
Additionally, non frequent items are removed from the transactions:

ID	Transaction
1	Apple, Banana
2	Banana, Cherry
3	Apple, Cherry
4	Apple, Banana
5	Apple

## 5. Create the initial FP-tree:

The initial FP-tree is created by inserting the items of each transaction into the tree:

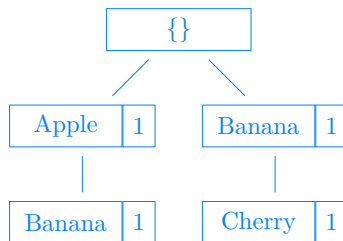
### a) Insert the first transaction (Apple, Banana):



Header table:

Item	Freq.	Nodes
Apple	1	1
Banana	1	1
Cherry	0	0

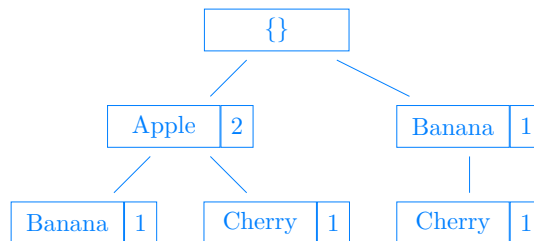
### b) Insert the second transaction (Banana, Cherry):



Header table:

Item	Freq.	Nodes
Apple	1	1
Banana	2	2
Cherry	1	1

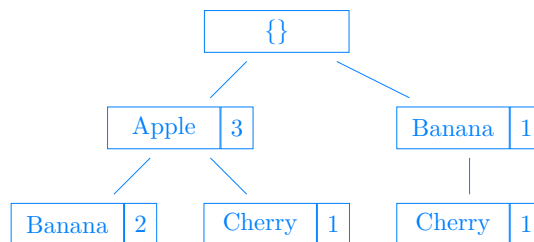
### c) Insert the third transaction (Apple, Cherry):



Header table:

Item	Freq.	Nodes
Apple	2	1
Banana	2	2
Cherry	2	2

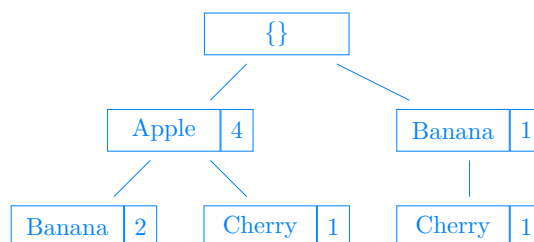
### d) Insert the fourth transaction (Apple, Banana):



Header table:

Item	Freq.	Nodes
Apple	3	1
Banana	3	2
Cherry	3	3

### e) Insert the fifth transaction (Apple):



Header table:

Item	Freq.	Nodes
Apple	4	1
Banana	3	2
Cherry	2	2

6. Determine the conditional pattern base for each frequent item in the header tables of the FP-tree:

a) **Conditional pattern base for Apple:**

Apple is the direct child of the root node, so the conditional pattern base for Apple is empty.

b) **Conditional pattern base for Banana:**

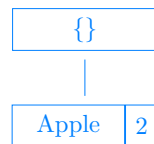
- Apple: 2

c) **Conditional pattern base for Cherry:**

- Apple: 1
- Banana: 1

7. Create the conditional FP-trees:

a) **Conditional FP-tree for Banana:**

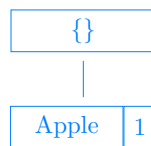


**Header table:**

Item	Freq.	Nodes
(Banana,) Apple	2	1

b) **Conditional FP-tree for Cherry:**

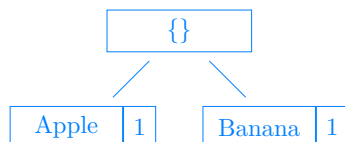
i. Insert „Apple: 1“:



**Header table:**

Item	Freq.	Nodes
(Cherry,) Apple	1	1
(Cherry,) Banana	0	0

ii. Insert „Banana: 1“:



**Header table:**

Item	Freq.	Nodes
(Cherry,) Apple	1	1
(Cherry,) Banana	1	1

8. Determine the conditional pattern base for each frequent itemset in the header tables of the conditional FP-trees:

a) **Conditional pattern base for „Banana, Apple“:**

Apple is the direct child of the root node, so the conditional pattern base is empty.

9. **Termination:**

The algorithm terminates because there are no more conditional FP-trees to create.

**Result:**

The frequent itemsets for a minimum support count of 2 are:

1. Apple
2. Banana
3. Cherry
4. Banana, Apple