Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Факультет інформаційних технологій
(факультет)

Кафедра системного аналізу та управління
(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

**Студента** Кормич Данило Глібович

**академічної групи** 124-21ск-1

**спеціальності** 124 Системний аналіз

**на тему**: «Системний аналіз та оптимізація діяльності організації роздрібної торгівлі»

| **Керівники** | **Прізвище, ініціали** | **Оцінка за шкалою** | | **Підпис** |
|---|---|---|---|---|
| | | **рейтинговою** | **Інституційною** | |
| **кваліфікаційної роботи** | Гаранжа Д.М., ст. викладач | | | |
| розділів: | | | | |
| Інформаційно-аналітичний | Гаранжа Д.М., ст. викладач | | | |
| Спеціальний розділ | Гаранжа Д.М., ст. викладач | | | |
| Рецензент | | | | |
| Нормоконтролер | доц. Хом'як Т.В. | | | |

Дніпро
2024

ЗАТВЕРДЖЕНО
**завідувач кафедри**
*Системного аналізу та управління*
(повна назва)

_____ *к.т.н., доц. Желдак Т.А.*
(підпис) (прізвище, ініціали)

«____»_____20___року

**ЗАВДАННЯ**
**на кваліфікаційну роботу**
**ступеня** *бакалавра*

**студенту** *Кормич Д.Г.* **академічної групи** *124-21ск-1*
**спеціальності:** *124 Системний аналіз*
**на тему**: *«Системний аналіз та оптимізація діяльності організації роздрібної торгівлі»*
затверджену наказом ректора НТУ «Дніпровська політехніка»
від *29.04.2024р. №375-с*

| Розділ | Зміст | Терміни виконання |
|---|---|---|
| 1. Інформаційно-аналітичний розділ | Проаналізувати дані транзакцій покупців. Провести аналіз існуючих методів формування асоціативних правил аналізу даних. | 11.09.2023-28.01.2024 |
| 2. Спеціальний розділ | Розробити програмне забезпечення в якому реалізовані типові алгоритми формування асоціативних правил. Зробити висновки щодо перспективності, переваг та недоліків запропонованих алгоритмів. | 29.01.2024-05.06.2024 |

Завдання _____ Гаранжа Д.М., ст. викладач
(підпис) (прізвище, ініціали)

Дата видачі: _____04.09.2023_____

Дата подання до екзаменаційної комісії: _____

Прийнято до виконання _____ Кормич Д.Г.
(підпис студента) (прізвище, ініціали)

# РЕФЕРАТ

Пояснювальна записка: 60с., 33 рис., 1 табл., 2 додатки, 17 джерел

*Об'єктом дослідження* є діяльність організації роздрібної торгівлі. Це охоплює всі аспекти роботи торговельної компанії, включаючи управління постачаннями, запасами, аналіз продажів, взаємодію з клієнтами та стратегічне планування.

*Предметом дослідження* є процеси, процедури та методи, що застосовуються в роздрібній торгівлі з метою їх аналізу та оптимізації. Це включає в себе управління запасами, ланцюгами поставок, асортиментом товарів, ціноутворенням, а також взаємодію з клієнтами.

*Метою* системного аналізу та оптимізації діяльності організації роздрібної торгівлі є підвищення ефективності та прибутковості бізнесу шляхом оптимізації процесів, зменшення витрат, покращення обслуговування клієнтів та адаптації до змін у ринкових умовах.

*Методи дослідження:* У роботі використано алгоритми Apriori, FP–growth, ECLAT для виявлення асоціативних зв'язків між різними товарами також моделювання бізнес–процесів з врахуванням результатів.

В *інформаційно–аналітичному розділі* наведено аналіз об'єкту дослідження та ключових проблем на ньому. Поставлені задачі дослідження та обрано концепції їх розв'язання.

У *спеціальному розділі* розроблено програмне забезпечення в якому реалізовані алгоритми Apriori, FP–growth, ECLAT для виявлення асоціативних зв'язків. Проаналізовані отримані результати та сформульовані висновки щодо можливості використання запропонованих алгоритмів.

Практична цінність: результати можуть бути використані для механізму рекомендацій товарів з метою збільшення продажів.

*Ключові слова*: СИСТЕМНИЙ АНАЛІЗ, APRIORI, FP–GROWTH, ECLAT, ТОРГІВЛЯ.

# ABSTRACT

Explanatory note: 60 pages, 33 figures, 1 table, 2 appendices, 17 sources

The object of the study is the activity of a retail trade organization. It covers all aspects of a trading company's operations, including supply management, inventory, sales analysis, customer interaction and strategic planning.

The subject of research is the processes, procedures and methods used in retail trade for the purpose of their analysis and optimization. This includes inventory management, supply chains, product mix, pricing, and customer interaction.

The purpose of systematic analysis and optimization of the activities of the retail trade organization is to increase the efficiency and profitability of the business by optimizing processes, reducing costs, improving customer service and adapting to changes in market conditions.

Research methods: The work uses Apriori, FP-growth, ECLAT algorithms to identify associative relationships between different products, as well as modeling business processes taking into account the results.

The informational and analytical section provides an analysis of the research object and its key problems. Research tasks are set and concepts for their solution are chosen.

In a special section, the logic and algorithm of the neural network for the analysis of the given task were formed, and a software application was written to solve the existing problem.

The practical value of the obtained results lies in increasing the efficiency of record management and reducing losses through the optimization and improvement of marketing and sales strategies.

Keywords: SYSTEM ANALYSIS, APRIORI, FP–GROWTH, ECLAT, TORIVLYA.

# ЗМІСТ

# ВСТУП

В сучасному світі, де конкуренція на ринку набирає обертів, і динаміка споживацьких вподобань постійно змінюється, системний аналіз та оптимізація діяльності організацій роздрібної торгівлі стають надзвичайно актуальними. Роздрібна торгівля відіграє ключову роль у задоволенні потреб споживачів, а також у формуванні їхнього споживчого досвіду та уподобань.

Однак, щоб ефективно конкурувати на цьому ринку та забезпечити стійкий успіх, компаніям необхідно вдосконалювати свої процеси та стратегії. В цьому контексті системний аналіз, що дозволяє ретельно досліджувати та оцінювати всі аспекти діяльності організації, а також оптимізація цих процесів для досягнення максимальної ефективності, стають невід'ємною частиною стратегічного управління.

У цій кваліфікаційній роботі ми розглянемо основні аспекти системного аналізу та оптимізації діяльності організацій роздрібної торгівлі. Ми дослідимо сучасні методи та інструменти аналізу, які допоможуть нам зрозуміти потреби ринку, побачити тенденції споживацьких вподобань та розробити ефективні стратегії відповідно до цих вимог. Також ми дослідимо ключові аспекти оптимізації, включаючи управління запасами, розміщення товарів, оптимізацію цін та маркетингові стратегії.

Наша мета полягає в тому, щоб використати цей аналіз для розробки конкретних рекомендацій та стратегій, які допоможуть підприємствам роздрібної торгівлі підтримувати конкурентні переваги, забезпечуючи якісний обслуговування клієнтів та збільшуючи свою прибутковість.

# ІНФОРМАЦІЙНО–АНАЛІТИЧНИЙ РОЗДІЛ

## 1.1 Актуальність теми

Ринок роздрібної торгівлі стає все більш насиченим, що змушує компанії шукати нові підходи для збереження та збільшення своєї частки ринку. Системний аналіз дозволяє виявити сильні та слабкі сторони діяльності компанії, а також зрозуміти можливості для розвитку.

Також з розвитком цифрових технологій та електронної комерції, роздрібні компанії мають адаптуватися до нових умов. Оптимізація бізнес–процесів через впровадження новітніх технологій, таких як великі дані, штучний інтелект та автоматизація, є необхідною для забезпечення конкурентоспроможності.

Споживачі стають більш вимогливими та поінформованими. Вони очікують високого рівня обслуговування, персоналізованих пропозицій та зручних умов для здійснення покупок. Системний аналіз допоможе компаніям краще розуміти потреби клієнтів та відповідати на них.

Для забезпечення сталого розвитку, роздрібним компаніям необхідно оптимізувати використання ресурсів: людських, фінансових та матеріальних. Системний аналіз допоможе ідентифікувати неефективні ділянки та знайти способи для їх удосконалення.

Сучасне законодавство та вимоги до сталого розвитку висувають нові виклики для роздрібних компаній. Використання системного аналізу допомагає організаціям відповідати на ці виклики, забезпечуючи відповідність нормативним вимогам та впроваджуючи принципи соціальної відповідальності та екологічної стійкості.

Таким чином, системний аналіз та оптимізація діяльності роздрібної торгівлі є критично важливими для сучасної компаній, що прагнуть залишатися

конкурентоспроможними, ефективними та адаптивними до швидкоплинних змін ринкового середовища.

Перед початком розробки додатку надзвичайно важливо провести ретельний аналіз наявних даних. Зокрема, необхідно детально ознайомитися з атрибутами, які входять до кожної транзакції. Це дозволить чітко визначити структуру даних і виокремити ключові характеристики, які мають критичне значення для подальшої обробки та успішного функціонування додатку. На рисунку 1.1 представлена інформація, що деталізує склад кожної транзакції, допомагаючи визначити, які саме дані є необхідними для досягнення цілей проекту.

| | id | DocId | Guid | ProductID | ProductName | Unit | Qty | Price | Sum |
|---|---|---|---|---|---|---|---|---|---|
| | Филь... | Филь... | Фильтр | Фильтр | Фильтр | Фи... | Фильтр | Фил... | Фильтр |
| 1 | 481267 | 204201 | B74C9CD1-C17F-41AD-B330-57B94B3C83C4 | 151087 | Кава мелена Himmel Хімел 500гр . | 3552 | 1000.0 | 125 | 125000 |
| 2 | 481268 | 204204 | 9DAB20D4-6B1B-429B-B2C5-72AFD640B46D | 149942 | Кава Lavazza TiERRA SELECTION коричневій 1кг | 3552 | 1000.0 | 560 | 560000 |
| 3 | 481269 | 204204 | 9DAB20D4-6B1B-429B-B2C5-72AFD640B46D | 132210 | Кава Tchibo Exclusive розчинна скло 100г/6шт | 3553 | 1000.0 | 155 | 155000 |
| 4 | 481367 | 204234 | 29F19E75-495E-477B-A605-09CBF4D020C5 | 154235 | Родзинки малойер світло-коричневий | 3553 | 180.0 | 100 | 18000 |
| 5 | 481368 | 204234 | 29F19E75-495E-477B-A605-09CBF4D020C5 | 132227 | Кешью сирий ww 320 (22.68 кг) | 3553 | 216.0 | 420 | 90720 |
| 6 | 481369 | 204234 | 29F19E75-495E-477B-A605-09CBF4D020C5 | 132200 | Мигдаль сирий золотий США (22.68 кг) | 3553 | 196.0 | 360 | 70560 |
| 7 | 481370 | 204234 | 29F19E75-495E-477B-A605-09CBF4D020C5 | 176025 | Фундук смажений Туреччина | 3553 | 106.0 | 450 | 47700 |
| 8 | 481371 | 204234 | 29F19E75-495E-477B-A605-09CBF4D020C5 | 179526 | Манго King 250 | 3552 | 1000.0 | 109.92 | 109920 |
| 9 | 481372 | 204235 | 8841524B-C7AA-4E8B-8B7E-D9CB28EA9D4C | 183654 | Цукерки Брауні з кешю | 3552 | 440.0 | 185 | 81400 |
| 10 | 481375 | 204238 | 115D78AB-9D9F-48A6-93C1-499738FC890C | 149996 | Драже Клім Хрусткий Ласунчик у молоч. 200г | 3552 | 1000.0 | 40 | 40000 |
| 11 | 481448 | 204245 | C084AF43-F570-4C70-B927-C9B55E8AF54D | 133401 | Родзинки довгий (В) ABC, Індія | 3553 | 530.0 | 85 | 45050 |
| 12 | 481449 | 204245 | C084AF43-F570-4C70-B927-C9B55E8AF54D | 133644 | Фиик ASAL Каб-Каб шоколадний | 3553 | 700.0 | 90 | 63000 |
| 13 | 481452 | 204249 | 962CE921-09E0-4047-9580-0A86CE1C2487 | 154235 | Родзинки малойер світло-коричневий | 3553 | 422.0 | 100 | 42200 |
| 14 | 481462 | 204252 | 6E9FA383-7578-43D9-8BA4-1995ACD8CB58 | 183654 | Цукерки Брауні з кешю | 3552 | 1648.0 | 185 | 304880 |
| 15 | 481576 | 204293 | E529242F-7311-4038-A319-F62CDD4EDEBD | 183654 | Цукерки Брауні з кешю | 3552 | 1698.0 | 185 | 314130 |
| 16 | 481577 | 204293 | E529242F-7311-4038-A319-F62CDD4EDEBD | 167727 | До мяса | 3553 | 226.0 | 250 | 56500 |
| 17 | 481578 | 204293 | E529242F-7311-4038-A319-F62CDD4EDEBD | 167837 | Перець мелений В/С | 3553 | 246.0 | 400 | 98400 |

Рисунок 1.1 – Приклад транзакцій в базі даних

На рисунку 1.2 представлена інформація про найбільш популярні товари. Цей графік дає змогу оцінити, які продукти мають найбільший попит серед клієнтів. Така візуалізація дозволяє визначити, які товари варто постійно мати в асортименті, а також які з них слід активно рекламувати. Аналіз популярності товарів є важливим для підтримки конкурентоспроможності на ринку, оскільки він допомагає швидко реагувати на зміну споживчих уподобань та попиту.

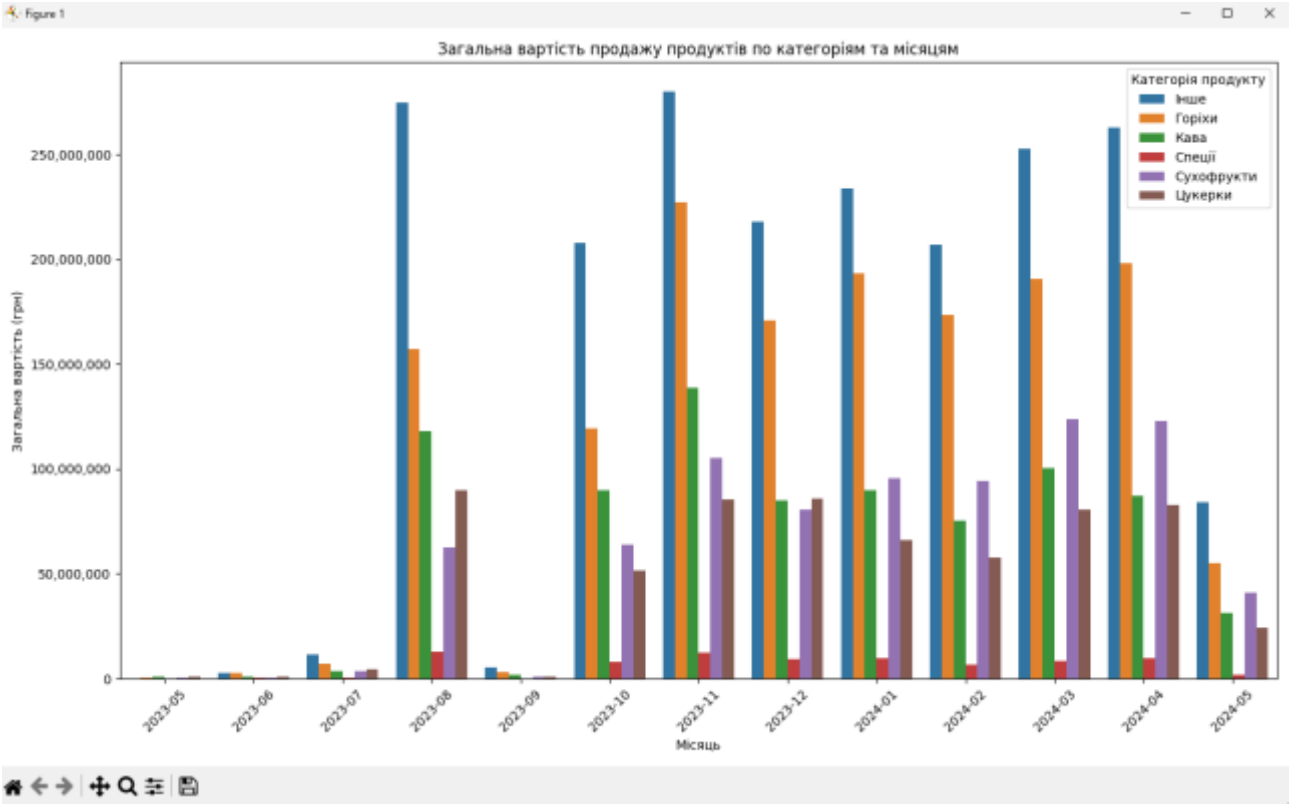



Рисунок 1.2 – Загальна вартість продажу продуктів по категоріям та місяцям

В умовах зростаючої насиченості ринку роздрібної торгівлі, компанії змушені шукати нові підходи для збереження та розширення своєї частки на ринку. Системний аналіз відіграє ключову роль у виявленні сильних і слабких сторін діяльності компанії, а також у визначенні можливостей для подальшого розвитку. Адаптація до нових умов, зумовлених розвитком цифрових технологій і електронної комерції, стає критичною для забезпечення конкурентоспроможності. Оптимізація бізнес-процесів через впровадження сучасних технологій, таких як великі дані, штучний інтелект і автоматизація, дозволяє компаніям ефективніше використовувати ресурси і задовольняти потреби споживачів.

Таким чином, системний аналіз і оптимізація діяльності є критично важливими для сучасних компаній, що прагнуть залишатися конкурентоспроможними, ефективними та адаптивними до швидкоплинних змін ринкового середовища.

## 1.2 Вхідний набір даних

У якості вхідних даних у кваліфікаційній роботі використовується набір реальних деталізованих транзакцій (чеків) покупців. Набір містить більш ніж 60000 транзакцій та описується 9–ма атрибутами:

- Id – ідентифікаційний номер транзакції;
- DocId – ідентифікаційний номер користувача;
- ProductId – ідентифікаційний номер товару;
- Date – дата транзакції;
- ProductName – назва товару;
- Unit – кількість товару на складі;
- Qty – кількість проданого товару;
- Price – ціна за одну одиницю товару;
- Sum – загальна вартість проданого товару.

Дані представлено на рисунку 1.3.

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Id | DocId | ProductId | Date | ProductName | Unit | Qty | Price | Sum |
| 2 | 481267 | 204201 | 151087 | 31.05.2023 | Кава мелена Himmel Хімел 500гр . | 3552 | 1000,000 | 125,00 | 125000,00 |
| 3 | 481268 | 204204 | 149942 | 31.05.2023 | Кава Lavazza TiERRA SELECTION коричневій 1кг | 3552 | 1000,000 | 560,00 | 560000,00 |
| 4 | 481269 | 204204 | 132210 | 31.05.2023 | Кава Tchibo Exclusive розчинна скло 100г/6шт | 3553 | 1000,000 | 155,00 | 155000,00 |
| 5 | 481367 | 204234 | 154235 | 31.05.2023 | Родзинки малойер світло-коричневий | 3553 | 180,000 | 100,00 | 18000,00 |
| 6 | 481368 | 204234 | 132227 | 31.05.2023 | Кешью сирий ww 320 (22.68 кг) | 3553 | 216,000 | 420,00 | 90720,00 |
| 7 | 481369 | 204234 | 132200 | 31.05.2023 | Мигдаль сирий золотий США (22.68 кг) | 3553 | 196,000 | 360,00 | 70560,00 |
| 8 | 481370 | 204234 | 176025 | 31.05.2023 | Фундук смажений Туреччина | 3553 | 106,000 | 450,00 | 47700,00 |
| 9 | 481371 | 204234 | 179526 | 31.05.2023 | Манго King 250 | 3552 | 1000,000 | 109,92 | 109920,00 |
| 10 | 481372 | 204235 | 183654 | 31.05.2023 | Цукерки Брауні з кешю | 3552 | 440,000 | 185,00 | 81400,00 |
| 11 | 481375 | 204238 | 149996 | 31.05.2023 | Драже Клім Хрусткий Ласунчик у молоч. 200г | 3552 | 1000,000 | 40,00 | 40000,00 |
| 12 | 481448 | 204245 | 133401 | 31.05.2023 | Родзинки довгий (В) ABC, Індія | 3553 | 530,000 | 85,00 | 45050,00 |
| 13 | 481449 | 204245 | 133644 | 31.05.2023 | Фиик ASAL Каб-Каб шоколадний | 3553 | 700,000 | 90,00 | 63000,00 |
| 14 | 481452 | 204249 | 154235 | 31.05.2023 | Родзинки малойер світло-коричневий | 3553 | 422,000 | 100,00 | 42200,00 |
| 15 | 481462 | 204252 | 183654 | 31.05.2023 | Цукерки Брауні з кешю | 3552 | 1648,000 | 185,00 | 304880,00 |
| 16 | 481576 | 204293 | 183654 | 31.05.2023 | Цукерки Брауні з кешю | 3552 | 1698,000 | 185,00 | 314130,00 |
| 17 | 481577 | 204293 | 167727 | 31.05.2023 | До мяса | 3553 | 226,000 | 250,00 | 56500,00 |
| 18 | 481578 | 204293 | 167837 | 31.05.2023 | Перець мелений В/С | 3553 | 246,000 | 400,00 | 98400,00 |
| 19 | 481579 | 204293 | 167818 | 31.05.2023 | 10-Овочів | 3553 | 346,000 | 320,00 | 110720,00 |
| 20 | 481580 | 204293 | 167870 | 31.05.2023 | Корица мелена | 3553 | 214,000 | 250,00 | 53500,00 |
| 21 | 481581 | 204293 | 167728 | 31.05.2023 | До курки | 3553 | 214,000 | 250,00 | 53500,00 |
| 22 | 481595 | 204298 | 167878 | 31.05.2023 | Гвоздика целая | 3553 | 28,000 | 778,57 | 21800,00 |
| 23 | 481642 | 204322 | 144880 | 31.05.2023 | Фундук сирий | 3553 | 500,000 | 360,00 | 180000,00 |
| 24 | 481643 | 204322 | 132200 | 31.05.2023 | Мигдаль сирий золотий США (22.68 кг) | 3553 | 318,000 | 360,00 | 114480,00 |
| 25 | 481699 | 204343 | 132237 | 31.05.2023 | Кешью смажений | 3553 | 254,000 | 450,00 | 114300,00 |
| 26 | 481700 | 204343 | 154479 | 31.05.2023 | Манго натураль №-1 | 3553 | 230,000 | 370,00 | 85100,00 |
| 27 | 535258 | 225534 | 133424 | 23.06.2023 | Чорнослив Венгерка Узбекистан | 3553 | 740,000 | 130,00 | 96200,00 |
| 28 | 535288 | 225563 | 132235 | 23.06.2023 | Суміш асорті горіхов №-1 | 3553 | 406,000 | 360,00 | 146160,00 |
| 29 | 535289 | 225563 | 190258 | 23.06.2023 | КУРАГА ДЖАМБО + Туреччина | 3553 | 242,000 | 300,00 | 72600,00 |
| 30 | 535290 | 225563 | 179395 | 23.06.2023 | Чорнослив вялений Молдова | 3552 | 258,000 | 180,00 | 46440,00 |
| 31 | 535291 | 225563 | 131466 | 23.06.2023 | Шок Cachet чорн. 70% 300 г | 3552 | 1000,000 | 135,00 | 135000,00 |
| 32 | 535292 | 225563 | 172911 | 23.06.2023 | Шоколад Toblerone Dark 100гр | 3552 | 1000,000 | 60,00 | 60000,00 |
| 33 | 535329 | 225591 | 132200 | 23.06.2023 | Мигдаль сирий золотий США (22.68 кг) | 3553 | 110,000 | 350,00 | 38500,00 |
| 34 | 535330 | 225591 | 144880 | 23.06.2023 | Фундук сирий | 3553 | 188,000 | 360,00 | 67680,00 |
| 35 | 535338 | 225608 | 133646 | 23.06.2023 | Цукерки Асорті Плюс в глазурі 1кг | 3553 | 192,000 | 185,00 | 35520,00 |
| 36 | 535339 | 225608 | 152963 | 23.06.2023 | Цукерки Асорті Плюс в глазурі 700г | 3552 | 1000,000 | 130,00 | 130000,00 |
| 37 | 535340 | 225608 | 133174 | 23.06.2023 | Кава Jacobs Monarch 120г м | 3552 | 1000,000 | 80,00 | 80000,00 |

Рисунок 1.3 – Набір даних

## 1.3 Алгоритми аналізу баз даних транзакцій

Навчання правилам асоціації — це заснований на правилах метод машинного навчання виявлення цікавих зв'язків між змінними у великих базах даних. Він призначений для виявлення сильних правил, виявлених у базах даних з використанням певних показників цікавості. У будь-якій транзакції з різними елементами правила асоціації призначені для виявлення закономірностей, які визначають, як і чому пов'язані певні елементи [1].

Зазвичай аналіз ринкового кошика проводиться з урахуванням великої таблиці фактів чи його частини.

Така таблиця містить як мінімум два стовпці: стовпець ідентифікатора транзакції або стовпець ідентифікатора часу та стовпець ідентифікатора елемента. Стовпці містять кожну транзакцію, тобто кожне замовлення на купівлю з придбаними товарами. Щоб заощадити використання диска, обидва стовпці часто є числовими. Крім того, зв'язок між ідентифікатором товару та назвою чи описом придбаного товару може зберігатися в окремій таблиці.

Шляхом пошуку в такій базовій таблиці аналіз ринкового кошика може виявити закономірності, що часто зустрічаються, наприклад, групи товарів, які купуються разом у багатьох транзакціях. Кількість транзакцій, що містять шаблон, називається підтримкою цього шаблону.

При аналізі таблиці використовується мінімальна підтримка. Мінімальна підтримка виключає шаблони, підтримка яких нижче вказаної мінімальної підтримки.

Можуть виникнути такі умови щодо мінімальної підтримки:

- Кількість виявлених частих патернів обернено пропорційно при мінімальній підтримці.
- Кількість виявлених частих шаблонів призводить до комбінаторного вибуху, якщо мінімальна підтримка встановлена надто низькою.
- Правила визначаються з урахуванням частих шаблонів.

Наприклад, з набору (ABC) випливають такі правила:

(AB)=>(C)

(AC) => (Б)

(BC) => (A)

Правило має таку ж підтримку, як і часті шаблони, у тому числі воно випливає. Правило також характеризується своєю достовірністю. Достовірність означає ймовірність того, що транзакції, що містять елементи у лівій частині правила, також міститимуть елементи у правій частині. Щоб запобігти

створенню та збереженню занадто великої кількості правил, рекомендується вказувати мінімальну достовірність[2].

Одними із найпопулярніших алгоритмів для побудування асоціативних правил виділяють три алгоритми, а саме Apriori, FP–Growth та ECLAT. Розглянемо детально ці алгоритми.

### 1.3.1 Алгоритм Apriori

Apriori - це алгоритм для частого аналізу наборів елементів та вивчення правил асоціації в реляційних базах даних [3]. Далі він ідентифікує як часто зустрічаються окремі елементи в базі даних і розширює їх до більших і більших наборів елементів, поки ці набори елементів з'являються в базі даних досить часто. Набори елементів, що часто зустрічаються, визначені за допомогою Apriori, можна використовувати для визначення правил асоціації, які підкреслюють загальні тенденції: це знаходить застосування в таких областях, як аналіз споживчого кошика та розробка систем рекомендацій\підказок для стимулювання продаж.

Розглянемо наступний набір даних, і ми знайдемо набори елементів, що часто зустрічаються, і створимо для них правила асоціації, представлено на рисунку 1.4.

| TID | items |
|---|---|
| T1 | I1, I2 , I5 |
| T2 | I2,I4 |
| T3 | I2,I3 |
| T4 | I1,I2,I4 |
| T5 | I1,I3 |
| T6 | I2,I3 |
| T7 | I1,I3 |
| T8 | I1,I2,I3,I5 |
| T9 | I1,I2,I3 |

Рисунок 1.4 – Приклад набору даних

***Крок 1:*** K = 1

Створимо таблицю, яка містить кількість підтримки кожного елемента, що є у наборі даних – називається C1 (набір кандидатів), приклад представлено на рисунку 1.5.

| Itemset | sup_count |
|---|---|
| I1 | 6 |
| I2 | 7 |
| I3 | 6 |
| I4 | 2 |
| I5 | 2 |

Рисунок 1.5 – Таблиця кількості підтримки кожного елементу

Потім порівнюємо кількість підтримки елементів набору кандидатів з мінімальною кількістю підтримки (тут *min_support=2*, якщо кількість підтримки елементів набору кандидатів менша за *min_support*, тоді видаліть ці елементи). Це дає нам набір елементів L1.

***Крок 2:*** K = 2

Генеруємо набір кандидатів C2, використовуючи L1 (це називається кроком з'єднання). Умовою сполуки Lk-1 та Lk-1 є наявність (K=2) спільних елементів.

Перевірте, чи всі підмножини набору елементів є частими чи ні, і якщо вони не часті, видаліть цей набір елементів. (Приклад підмножини {I1, I2} – це {I1}, {I2} – вони часті. Перевірте кожен набір елементів).

Тепер знаходимо кількість підтримки цих наборів елементів, виконавши пошук у наборі даних, приклад представлено на рисунку 1.6.

| Itemset | sup_count |
|---|---|
| I1,I2 | 4 |
| I1,I3 | 4 |
| I1,I4 | 1 |
| I1,I5 | 2 |
| I2,I3 | 4 |
| I2,I4 | 2 |
| I2,I5 | 2 |
| I3,I4 | 0 |
| I3,I5 | 1 |
| I4,I5 | 0 |

Рисунок 1.6 – Таблиця кількості підтримки

Потім порівнюємо кількість підтримки кандидатів (C2) з мінімальною кількістю підтримки (тут *min_support=2*, якщо *support_count* елемента набору кандидатів менше, ніж *min_support*, тоді видаліть ці елементи), це дає нам набір елементів L2, представлено на рисунку 1.7.

| Itemset | sup_count |
|---|---|
| I1,I2 | 4 |
| I1,I3 | 4 |
| I1,I5 | 2 |
| I2,I3 | 4 |
| I2,I4 | 2 |
| I2,I5 | 2 |
| I2,I5 | 2 |

Рисунок 1.7 – Таблиця кількості даних після порівняння

***Крок 3***: Генеруємо набір кандидатів C3, використовуючи L2 (крок об'єднання). Умовою сполуки Lk-1 та Lk-1 є наявність (K-2) загальних елементів. Отже, для L2 перший елемент має збігатися.

Таким чином, набір елементів, згенерований при об'єднанні L2, дорівнює {I1, I2, I3}{I1, I2, I5}{I1, I3, i5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}

Перевіримо, чи всі підмножини цих наборів елементів часті чи ні, і якщо ні, видаліть цей набір елементів. (Тут підмножиною {I1, I2, I3} є {I1, I2}, {I2, I3}, {I1, I3}, які для {I2, I3, I4} підмножина {I3, I4} зустрічається нечасто, тому видаліть його аналогічно.

Знаходимо кількість підтримки цих наборів елементів, що залишилися, виконавши пошук у наборі даних, приклад представлено на рисунку 1.8.

| Itemset | sup_count |
|---|---|
| I1,I2,I3 | 2 |
| I1,I2,I5 | 2 |

Рисунок 1.8 – Таблиця кількості підтримки після пошуку

Порівнюємо кількість підтримки кандидатів (C3) з мінімальною кількістю підтримки (тут *min_support=2*, якщо *support_count* елемента набору кандидатів менше, ніж *min_support*, тоді видаліть ці елементи), це дає нам набір елементів L3.

Продовжуємо до тих пір, поки не будуть виявлені повторювані набори елементів. Іншими словами, процес триває до моменту, коли ми знайдемо комбінації елементів, які вже зустрічалися раніше. Після виявлення таких наборів аналіз може бути завершений [4].

Нехай ми маємо якийсь датасет (або колекція) D, такий, що,

$$D = d_0..d_j \tag{1.1}$$

де d — унікальна транзакція-*itemset* (наприклад, касовий чек). Всередині кожної d представлений набір *items* (i - item), причому в ідеальному випадку він представлений у бінарному вигляді: *d1 = [{{Хліб: 1}, {Вода: 0}, {Кола: 1}, {...}],*

*d2 = [{Хліб: 0}, {Вода: 1}, {Кола: 1}, {...}]*. Прийнято кожен *itemset* описувати через кількість ненульових значень (*k-itemset*), наприклад, [{Хліб: 1}, {Вода: 0}, {Кола: 1}] є *2-itemset*.

Якщо спочатку датасет у бінарному вигляді не представлений, можна за бажання руками його перетворити (One Hot Encoding та pd.get_dummies вам на допомогу).

Таким чином, датасет є розрідженою матрицею зі значеннями {1,0}. Це буде бінарний датасет. Існують і інші види запису — вертикальний датасет (показує для кожного окремого item вектор транзакцій, де він є) і транзакційний датасет (приблизно як у касовому чеку).

Перше поняття в *ARL - support*:

$$supp(x_1 \cup x_2) = \frac{\sigma(x_1 \cup x_2)}{|T|} \quad (1.2)$$

де $\sigma$ – кількість транзакцій, що містять $x_1$ і $x_2$

Наступне ключове поняття – *confidence*. Це показник того, як часто наше правило спрацьовує для датасету.

$$conf(x_1 \cup x_2) = \frac{supp(x_1 \cup x_2)}{supp(x_1)} \quad (1.3)$$

Наступне поняття нашого списку - *lift*. Грубо кажучи, *lift* – це відношення «залежності» items до їхньої «незалежності». *Lift* показує, наскільки items залежить один від одного. Це очевидно з формули:

$$lift(x_1 \cup x_2) = \frac{supp(x_1 \cup x_2)}{supp(x_1) * supp(x_2)} \quad (1.4)$$

У загальному вигляді *Conviction* - це "частотність помилок" нашого правила.

$$conu(x_1 \cup x_2) = \frac{1-supp(x_1)}{1-conf(x_1 \cup x_2)} \quad (1.5)$$

Що результат за формулою вище 1, то краще. Наприклад, якщо *conviction* купівлі хліба і молока разом дорівнював би 1.2, це означає, що правило «купив хліб і молоко» було б у 1.2 рази (на 20%) більш вірним, ніж якби збіг цих *items* в одній транзакції було б чисто випадковим. Трохи не інтуїтивне поняття, але й використовується не так часто, як попередні три [5].

### 1.3.2 Алгоритм FP-Growth

Часте виявлення шаблонів (або виявлення FP, аналіз FP або частий аналіз набору елементів) є частиною виявлення знань у базах даних, масового онлайн-аналізу та інтелектуального аналізу даних; він визначає завдання пошуку найчастіших і релевантних закономірностей у великих наборах даних. Ця концепція була вперше представлена для майнінгу транзакційних баз даних. Шаблони, що часто зустрічаються, визначаються як підмножини (набори елементів, підпослідовності або підструктури), які з'являються в наборі даних з частотою не нижче заданого користувачем або автоматично визначеного порога [6].

Двома основними недоліками апріорного алгоритму є:

- На кожному етапі потрібно створювати набори кандидатів.
- Щоб побудувати набори кандидатів, алгоритм має неодноразово сканувати базу даних.

Ці дві властивості неминуче уповільнюють алгоритм. Щоб подолати ці надлишкові кроки, було розроблено новий алгоритм інтелектуального аналізу правил асоціації, який отримав назву Алгоритм зростання частого шаблону. Він

долає недоліки алгоритму Apriori, зберігаючи всі транзакції у структурі даних Trie. Розглянемо такі дані, представлено на рисунку 1.9

| Transaction ID | Items |
|---|---|
| T1 | {E, K, M, N, O, Y} |
| T2 | {D, E, K, N, **O**, Y} |
| T3 | {A, E, K, M} |
| T4 | {C, K, M, U, Y} |
| T5 | {C, E, I, K, O, O} |

Рисунок 1.9 – Початкові дані

Наведені вище дані є гіпотетичний набір даних транзакцій, де кожна буква представляє елемент. Частота кожного окремого елемента розраховується, приклад представлено на рисунку 1.10.

| Item | Frequency |
|---|---|
| **A** | **1** |
| **C** | **2** |
| **D** | **1** |
| **E** | **4** |
| **I** | **1** |
| **K** | **5** |
| **M** | **3** |
| **N** | **2** |
| **O** | **4** |
| **U** | **1** |
| **Y** | **3** |

Рисунок 1.10 – Частота кожного елементу

Нехай мінімальна підтримка дорівнює 3. Створюється набір шаблонів, що часто зустрічаються, який міститиме всі елементи, частота яких більша або дорівнює мінімальній підтримці. Ці елементи зберігаються у порядку зменшення відповідних їм частот. Після додавання відповідних товарів набір L виглядає так:

*L = {K: 5, E: 4, M: 3, O: 4, Y: 3}*

Тепер для кожної з транзакцій відповідний набір замовлених товарів побудовано. Це робиться шляхом повторення набору частих шаблонів і перевірки, чи міститься поточний елемент у транзакції, що розглядається. Якщо поточний елемент міститься, він вставляється в комплект *Ordered-Item* для поточної транзакції. Для всіх транзакцій будується така таблиця, представлена на рисунку 1.11.

| Transaction ID | Items | Ordered-Item Set |
|---|---|---|
| T1 | {E, K, M, N, O, Y} | {K, E, M, O, Y} |
| T2 | {D, E, K, N, O, Y} | {K, E, O, Y} |
| T3 | {A, E, K, M} | {K, E, M} |
| T4 | {C, K, M, U, Y} | {K, M, Y} |
| T5 | {C, E, I, K, O, O} | {K, E, O} |

Рисунок 1.11 – Створення часто зустрічаючих шаблонів

Тепер усі набори замовлених товарів вставляють у структуру даних *Trie*.

а) Вставка набору *{K, E, M, O, Y}*: тут всі елементи просто зв'язуються один за одним у порядку появи в наборі та ініціалізують лічильник підтримки для кожного елемента 1, рисунок 1.12

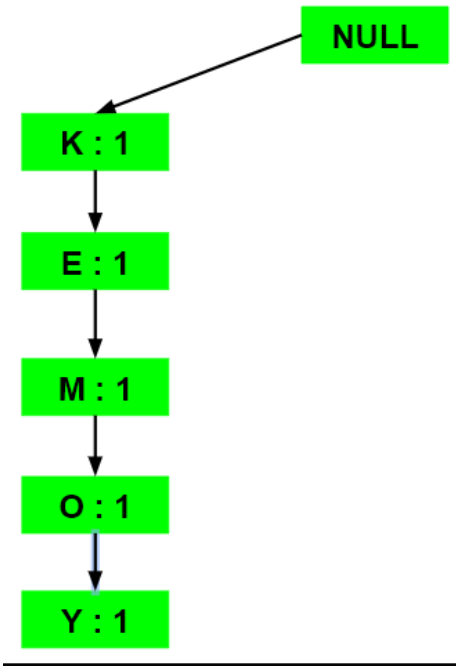


Рисунок 1.12 – Початковий набір

б) Вставка набору *{K, E, O, Y}*: до вставки елементів *K* і *E* просто кількість опор збільшується на 1. При вставці *O* бачимо, що між *E* і *O* немає прямого зв'язку. тому новий вузол для елемента *O* ініціалізується з лічильником підтримки, рівним 1, і елемент *E* пов'язаний з цим новим вузлом. При вставці Y спочатку ініціалізуємо новий вузол для елемента *Y* з лічильником підтримки, рівним 1, і зв'язуємо новий вузол *O* з новим вузлом *Y*. Вставка набору *{K, E, M}*: тут кількість опор кожного елемента збільшується на 1, рисунок 1.13

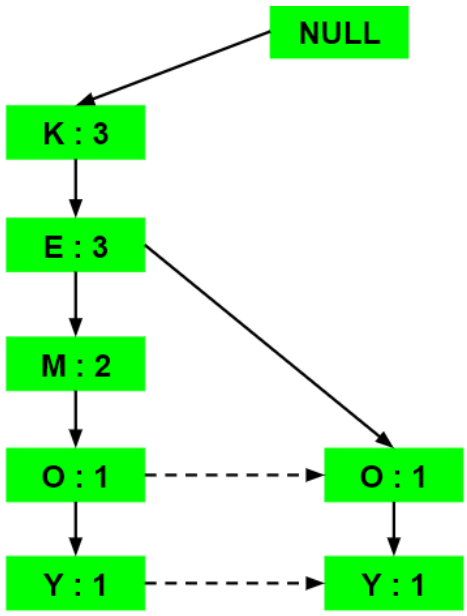



Рисунок 1.13 – Вставка нових наборів

в) Вставка набору *{K, M, Y}*: аналогічно кроку б), спочатку збільшується кількість підтримуваних *K*, потім ініціалізуються та зв'язуються нові вузли для *M* та *Y* відповідно також тут просто збільшується кількість підтримок відповідних елементів. Зверніть увагу, що кількість підтримки нового вузла елемента *O* збільшена, рисунок 1.14.

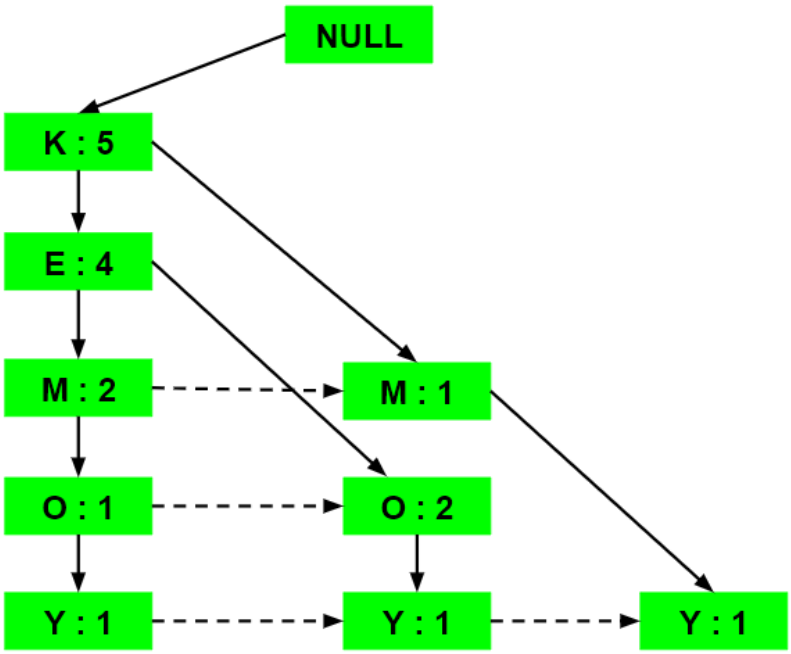



Рисунок 1.14 – Додавання нових наборів

Тепер для кожного елемента обчислюється база умовного шаблону , яка є мітками всіх шляхів, що ведуть до будь-якого вузла даного елемента в дереві шаблонів, що часто зустрічаються. Зауважте, що елементи в таблиці нижче розташовані в порядку зростання їх частоти, рисунок 1.15.

| Items | Conditional Pattern Base |
|---|---|
| Y | {{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}} |
| O | {{K,E,M : 1}, {K,E : 2}} |
| M | {{K,E : 2}, {K : 1}} |
| E | {K : 4} |
| K | |

Рисунок 1.15 – Обчислення бази умовного шаблону

Тепер для кожного елемента будується дерево умовної частої закономірності. Це робиться шляхом взяття набору елементів, який є загальним для всіх шляхів у базі умовного шаблону цього елемента, та розрахунку

кількості його підтримки шляхом підсумовування лічильників підтримки всіх шляхів у базі умовного шаблону, рисунок 1.16.

| Items | Conditional Pattern Base | Conditional Frequent Pattern Tree |
|---|---|---|
| Y | {{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}} | {K : 3} |
| O | {{K,E,M : 1}, {K,E : 2}} | {K,E : 3} |
| M | {{K,E : 2}, {K : 1}} | {K : 3} |
| E | {K : 4} | {K : 4} |
| K | | |

Рисунок 1.16. – Створення дерева умовної закономірності

З дерева умовного частого шаблону правила частого шаблону створюються шляхом об'єднання елементів дерева умовного частого шаблону, які відповідають елементу, як зазначено в таблиці нижче, рисунок 1.17.

| Items | Frequent Pattern Generated |
|---|---|
| Y | {<K,Y : 3>} |
| O | {<K,O : 3>, <E,O : 3>, <E,K,O : 3>} |
| M | {<K,M : 3>} |
| E | {<E,K : 4>} |
| K | |

Рисунок 1.17 – Дерево умовного частого шаблону

Для кожного рядка можна вивести два типи правил асоціації, наприклад, для першого рядка, що містить елемент, можна вивести правила K –> Y та Y –> K. Щоб визначити допустиме правило, розраховується достовірність обох правил і зберігається те, достовірність якого більша або дорівнює мінімального значення достовірності [7].

Нехай у нас є початковий транзакційний датасет A, на якому вже побудовано FP-дерево T(A) та відсортований за зменшенням підтримки список

ознак *H(A)*. Поточний датасет із знову отриманими даними позначимо, як *C*. Головне питання - як маючи на руках початкове дерево *T(A)* і відсортований список ознак *H(A)*, побудувати "доповнене" дерево *T(C)* так, щоб не довелося знову будувати з нуля все дерево *T(C)*. Нехай також B - нові невраховані дані, причому $B = C \setminus A$. Проблема особливо актуальна, якщо *A* - величезна, роками накопичена база даних, а *B* - статистика, що накопичилася за останній тиждень, яку теж хотілося б врахувати. Донавчанням назвемо процес отримання дерева *T(C)* з *T(A)*, використовуючи *B* і *H(A)*. Символічно донавчання позначимо так:

$$T(A) \xrightarrow[B,H(A)]{} T(C) \qquad (1.6)$$

Давайте розглянемо ще один приклад. Представимо модифікацію алгоритму, в якій враховується старіння даних та їх поступове видалення та натомість – додавання актуальних. Що ж нам, щоразу трохи зменшувати дані, трохи додавати і наново обробляти всю цю величезну транзакційну БД, витрачаючи величезну кількість часу та ресурсів заради щотижневої статистики? А якщо статистику потрібно оновлювати частіше, ніж раз на тиждень? У цьому полягає проблематика [8].

### 1.3.3 Алгоритм ECLAT

Алгоритм ECLAT означає кластеризацію класів еквівалентності та обхід ґрат знизу вгору. Це один із популярних методів майнінгу асоціативних правил. Це ефективніша і масштабована версія алгоритму Апріорі. У той час як алгоритм Apriori працює в горизонтальному напрямку, імітуючи пошук завширшки по графу, алгоритм ECLAT працює вертикально, так само, як пошук у глибину графа. Вертикальний підхід алгоритму ECLAT робить його швидшим, ніж алгоритм Apriori.

Основна ідея полягає в тому, щоб використовувати перетин наборів ідентифікаторів транзакцій (*tidsets*) для обчислення значення підтримки кандидата та уникнути створення підмножин, які не існують у дереві префіксів. При першому виклику функції використовуються всі окремі елементи разом із наборами даних. Потім функція викликається рекурсивно, і при кожному рекурсивному виклику кожна пара елемент-*tidset* перевіряється та поєднується з іншими парами елемент-*tidset*. Цей процес триває доти, доки не вдасться об'єднати жодну пару-кандидат елемент-набір елементів.

Давайте тепер розберемося у наведеному нижче прикладі, рисунок 1.18.

| Transaction Id | Bread | Butter | Milk | Coke | Jam |
|---|---|---|---|---|---|
| T1 | 1 | 1 | 0 | 0 | 1 |
| T2 | 0 | 1 | 0 | 1 | 0 |
| T3 | 0 | 1 | 1 | 0 | 0 |
| T4 | 1 | 1 | 0 | 1 | 0 |
| T5 | 1 | 0 | 1 | 0 | 0 |
| T6 | 0 | 1 | 1 | 0 | 0 |
| T7 | 1 | 0 | 1 | 0 | 0 |
| T8 | 1 | 1 | 1 | 0 | 1 |
| T9 | 1 | 1 | 1 | 0 | 0 |

Рисунок 1.18 – Приклад для аналізу

Наведені вище дані являють собою логічну матрицю, де для кожного осередку (i, j) значення позначає, включений j-й елемент і-ю транзакцію чи ні. 1 означає істину, а 0 означає брехню.

Тепер ми викликаємо функцію вперше і впорядковуємо кожен елемент з набором елементів, рисунок 1.19.

k = 1, мінімальна підтримка = 2

| Елемент | Тайдсет |
|---|---|
| Хліб | {T1, T4, T5, T7, T8, T9} |
| Масло | {T1, T2, T3, T4, T6, T8, T9} |
| Молоко | {T3, T5, T6, T7, T8, T9} |
| Кокс | {T2, T4} |
| Варення | {T1, T8} |

Рисунок 1.19 – Перше впорядкування елементів

Тепер ми рекурсивно викликаємо функцію доти, доки не перестануть об'єднуватися пари елемент-*tidset*, рисунок 1.20.

| Елемент | Тайдсет |
|---|---|
| {Хліб, Масло, Молоко, Варення} | {T8} |

Рисунок 1.20 – Останній впорядкування елементів

Ми зупиняємось на k = 4, тому що більше немає пар «елемент – набір тегів», які можна було б об'єднати.

Переваги перед алгоритмом Apriori:

1. Вимоги до пам'яті: оскільки алгоритм ECLAT використовує підхід пошуку до глибини, він використовує менше пам'яті, ніж алгоритм Apriori.

2. Швидкість: алгоритм ECLAT зазвичай швидше, ніж алгоритм Apriori.

3. Кількість обчислень: Алгоритм ECLAT передбачає багаторазового сканування даних до розрахунку окремих значень опори [9].

У сфері інтелектуального аналізу даних, алгоритми Apriori, FP-growth та ECLAT широко використовуються для виявлення асоціативних правил. Кожен з них володіє унікальними характеристиками, що робить їх придатними для різних завдань та типів даних.

Apriori ґрунтується на багаторазовому скануванні бази даних, генеруючи та обрізаючи кандидати наборів правил. Цей метод, хоча й простий та поширений, має певні недоліки, пов'язані з масштабованістю та використанням пам'яті, особливо при роботі з великими обсягами даних. Проте, він залишається ефективним для аналізу даних середнього розміру.

FP-growth пропонує значне покращення продуктивності завдяки побудові компактної структури під назвою FP-дерево. Ця структура усуває потребу в генеруванні кандидатів, роблячи алгоритм економним з точки зору пам'яті. FP-growth особливо ефективний для наборів даних з великою кількістю транзакцій та невеликою кількістю часто повторюваних елементів. Він ідеально підходить для аналізу великих обсягів даних.

На відміну від Apriori та FP-growth, ECLAT використовує інший підхід, виявляючи часті набори елементів за допомогою ефективних операцій перетину підмножин та стратегії пошуку в глибину. Цей алгоритм масштабований та може ефективно працювати як з щільними, так і з розрідженими наборами даних. Проте, його реалізація може бути складнішою порівняно з Apriori та FP-growth.

Оптимальний алгоритм для виявлення асоціативних правил залежить від декількох факторів, таких як:

– Розмір набору даних: Apriori може бути кращим вибором для менших наборів даних завдяки простоті та доступності.

– Масштабованість: FP-growth та ECLAT краще підходять для роботи з великими обсягами даних.

– Використання пам'яті: FP-growth економить пам'ять за рахунок використання FP-дерева.

– Розрідженість даних: ECLAT ефективно працює з розрідженими даними.

– Конкретні вимоги: Деякі алгоритми можуть мати додаткові функції, що відповідають певним потребам.

Розробка додатку, який базується на аналізі транзакційних даних та популярності товарів, вимагає виконання ряду ключових завдань, спрямованих на забезпечення його ефективності та корисності для користувачів. Постановка задачі охоплює наступні аспекти:

- Аналіз транзакційних даних;
- Оцінка популярності товарів;
- Розробка додатку;
- Оптимізація асортименту та маркетингових стратегій;
- Забезпечення конкурентоспроможності.

Таким чином, розробка додатку передбачає комплексний підхід до аналізу транзакційних даних і популярності товарів, з метою створення інструменту, який допоможе підвищити ефективність бізнесу та задовольнити потреби клієнтів.

Було проаналізовано основні аспекти, що мають значення для ефективної розробки додатку на основі транзакційних даних і аналізу популярності товарів. Важливу роль у цьому процесі відіграють методи виявлення асоціативних правил, такі як Apriori, ECLAT та FP-growth. Ці методи дозволяють виявляти часті набори товарів і асоціативні правила, які допомагають зрозуміти поведінку споживачів і оптимізувати асортимент продукції.

Метод Apriori використовується для знаходження частих наборів товарів через ітеративний підхід, що дозволяє виявити товари, які часто купуються разом. Це дає можливість визначити ключові товари для промоційних кампаній та оптимізації запасів. ECLAT дозволяє більш ефективно аналізувати великі обсяги транзакцій, використовуючи перехресні таблиці для швидкого виявлення частих наборів товарів. Метод FP-growth використовує підхід побудови дерева частих елементів, що значно зменшує обчислювальні витрати і час аналізу, що є критичним для роботи з великими обсягами даних.

Застосування цих методів дозволяє виявити приховані закономірності у транзакційних даних, визначити товари, які мають найбільший попит серед клієнтів, і забезпечити ефективну стратегію управління асортиментом. Це допоможе бізнесу швидко реагувати на зміни попиту та залишатися конкурентоспроможним на ринку.

# СПЕЦІАЛЬНИЙ РОЗДІЛ

## 2.1 Вибір програмного забезпечення

В дані кваліфікаційній роботі для створення додатку було використано мову програмування Python.

Python – це мультипарадигмальна мова програмування. Об'єктно-орієнтоване програмування та структурне програмування повністю підтримуються, і багато з них підтримують функціональне програмування та аспектно-орієнтоване програмування (включаючи метапрограмування і метаоб'єкти). Багато інших парадигми підтримуються за допомогою розширень, включаючи проектування за контрактом та логічне програмування.

Python використовує динамічну типізацію та комбінацію підрахунку посилань та збирача сміття з виявленням циклів для управління пам'яттю. Він використовує динамічний дозвіл імен (пізнє зв'язування), яке пов'язує імена методів і змінних під час виконання програми.

Велика стандартна бібліотека Python надає інструменти, що підходять для багатьох завдань, і її зазвичай називають однією з найсильніших сторін. Для інтернет-додатків підтримуються багато стандартних форматів і протоколів, таких як MIME і HTTP . Він включає в себе модулі для створення графічних інтерфейсів користувача, підключення до реляційних баз даних, генерації псевдовипадкових чисел, арифметики з десятковими дробами довільної точності, маніпулювання регулярними виразами і модульного тестування. [10].

Python — це мова програмування, широко відома як доброзичлива до новачків. Однією з основних причин, через яку Python вважається простим у освоєнні, є його простий синтаксис. Код Python легко читати і розуміти, що полегшує новачкам написання та налагодження коду.

Ще одна причина, через яку ця мова зручна для початківців, — це її універсальність. Python можна використовувати для широкого спектру програм: від веб-розробки Python до аналізу даних та машинного навчання. Це означає, що новачки можуть вибрати область, що їх цікавить, і почати вивчати Python у контексті, що відповідає їх інтересам.

Python також має безліч ресурсів, таких як онлайн-посібники, відеокурси та інтерактивні платформи кодування. Ці ресурси містять покрокові інструкції та допомагають структуровано розвивати свої навички.

Python має велику і активну спільноту розробників і відомий своєю дружелюбністю, гостинністю та підтримкою. Ви можете знайти підтримку в групах соціальних мереж і онлайн-форумах, незалежно від того, якою мовою ви говорите.

Спільнота Python також бере активну участь у проектах з відкритим вихідним кодом. На Python є безліч бібліотек і фреймворків з відкритим вихідним кодом, які підтримуються спільнотою.

Коли ви вивчаєте нову мову, дуже важливо мати спільноту, де ви можете ставити запитання більш досвідченим професіоналам та отримувати від них відгуки.

Python - це мова високого рівня, що означає, що її легко читати і писати, при цьому основна увага приділяється абстрагування низькорівневих деталей та забезпечення більш високого рівня абстракції. Однак це також потужна мова, яку можна використовувати для складних проектів.

Однією з основних причин такої гнучкості Python є його велика бібліотека модулів та пакетів. Ці бібліотеки надають попередньо написаний код, який можна легко інтегрувати в проект, заощаджуючи час та зусилля розробників. Крім того, Python можна використовувати для веб-розробки, аналізу даних, машинного навчання та наукових обчислень, а також для інших програм.

Гнучкість Python також обумовлена його здатністю працювати з іншими мовами. Його можна легко інтегрувати з мовами, як C++ і Java, що дозволяє розробникам використовувати Python для конкретних завдань поряд з іншими мовами для інших частин проекту.

Одним з основних недоліків Python є те, що він повільніший, ніж компілювані мови, такі як C + + або Java. Це пов'язано з тим, що Python є мовою, що інтерпретується, а це означає, що кожен рядок коду виконується інтерпретатором по одному. Навпаки, скомпільовані мови перетворюються на машинний код перед виконанням, що робить їх швидше.

Ця різниця у швидкості може бути особливо помітна під час роботи з великими наборами даних або виконання складних обчислень. У цих випадках Python може бути не найкращим вибором для програм, критичних до продуктивності. Однак варто відзначити, що існують способи оптимізації коду Python та підвищення його продуктивності, наприклад використання NumPy для числових операцій або Cython для компіляції Python коду в C.

Незважаючи на обмеження продуктивності, Python залишається популярною мовою для прототипування та експериментування завдяки простоті використання та широкій бібліотеці модулів. Розробникам, яким необхідно оптимізувати свій код для програм, критичних до продуктивності, можливо, доведеться розглянути інші мови або інструменти, але для багатьох програм сильні сторони Python переважують його слабкості[11].

Також було використано деякі бібліотеки а саме:

1. Pandas (в стилі pandas) - це програмна бібліотека, написана для мови програмування Python для маніпулювання та аналізу даних. Зокрема, він пропонує структури даних та операції для маніпулювання числовими таблицями та часовими рядами [12];

2. SQLite – це механізм бази даних, написаний мовою програмування C. Це не окремий додаток; швидше, це бібліотека, яку розробники програмного забезпечення вбудовують у свої додатки [13];

3. Matplotlib – це бібліотека побудови графіків для мови програмування Python та його розширення числової математики NumPy. Він надає об'єктно-орієнтований API для вбудовування графіків у додатки з використанням наборів інструментів графічного інтерфейсу користувача загального призначення [14];

4. Apriori – бібліотека для підключення алгоритму Apriori [15];

5. Fpgrowth – бібліотека для підключення алгоритму Fpgrowth [16];

6. ECLAT – бібліотека для підключення алгоритму ECLAT [17].

## 2.2 Попередній аналіз даних та візуалізація

Перед тим як приступити до розробки додатка, необхідно провести детальний аналіз даних. Важливо зрозуміти, які саме атрибути включені в кожну транзакцію. Це дозволить чітко визначити структуру даних та визначити ключові характеристики, які будуть необхідні для подальшої обробки. На рисунку 2.1 представлена інформація, яка деталізує склад кожної транзакції, що допоможе виявити, які дані мають критичне значення для функціональності додатка.

| | id | DocId | Guid | ProductID | ProductName | Unit | Qty | Price | Sum |
|---|---|---|---|---|---|---|---|---|---|
| | Филь... | Филь... | Фильтр | Фильтр | Фильтр | Фи... | Фильтр | Фил... | Фильтр |
| 1 | 481267 | 204201 | B74C9CD1-C17F-41AD-B330-57B94B3C83C4 | 151087 | Кава мелена Himmel Хімел 500гр . | 3552 | 1000.0 | 125 | 125000 |
| 2 | 481268 | 204204 | 9DAB20D4-6B1B-429B-B2C5-72AFD640B46D | 149942 | Кава Lavazza TiERRA SELECTION коричневій 1кг | 3552 | 1000.0 | 560 | 560000 |
| 3 | 481269 | 204204 | 9DAB20D4-6B1B-429B-B2C5-72AFD640B46D | 132210 | Кава Tchibo Exclusive розчинна скло 100г/6шт | 3553 | 1000.0 | 155 | 155000 |
| 4 | 481367 | 204234 | 29F19E75-495E-477B-A605-09CBF4D020C5 | 154235 | Родзинки малойер світло-коричневий | 3553 | 180.0 | 100 | 18000 |
| 5 | 481368 | 204234 | 29F19E75-495E-477B-A605-09CBF4D020C5 | 132227 | Кешью сирий ww 320 (22.68 кг) | 3553 | 216.0 | 420 | 90720 |
| 6 | 481369 | 204234 | 29F19E75-495E-477B-A605-09CBF4D020C5 | 132200 | Мигдаль сирий золотий США (22.68 кг) | 3553 | 196.0 | 360 | 70560 |
| 7 | 481370 | 204234 | 29F19E75-495E-477B-A605-09CBF4D020C5 | 176025 | Фундук смажений Туреччина | 3553 | 106.0 | 450 | 47700 |
| 8 | 481371 | 204234 | 29F19E75-495E-477B-A605-09CBF4D020C5 | 179526 | Манго King 250 | 3552 | 1000.0 | 109.92 | 109920 |
| 9 | 481372 | 204235 | 8841524B-C7AA-4E8B-8B7E-D9CB28EA9D4C | 183654 | Цукерки Брауні з кешю | 3552 | 440.0 | 185 | 81400 |
| 10 | 481375 | 204238 | 115D78AB-9D9F-48A6-93C1-499738FC890C | 149996 | Драже Клім Хрусткий Ласунчик у молоч. 200г | 3552 | 1000.0 | 40 | 40000 |
| 11 | 481448 | 204245 | C084AF43-F570-4C70-B927-C9B55E8AF54D | 133401 | Родзинки довгий (B) ABC, Індія | 3553 | 530.0 | 85 | 45050 |
| 12 | 481449 | 204245 | C084AF43-F570-4C70-B927-C9B55E8AF54D | 133644 | Финик ASAL Каб-Каб шоколадний | 3553 | 700.0 | 90 | 63000 |
| 13 | 481452 | 204249 | 962CE921-09E0-4047-9580-0A86CE1C2487 | 154235 | Родзинки малойер світло-коричневий | 3553 | 422.0 | 100 | 42200 |
| 14 | 481462 | 204252 | 6E9FA383-7578-43D9-8BA4-1995ACD8CB58 | 183654 | Цукерки Брауні з кешю | 3552 | 1648.0 | 185 | 304880 |
| 15 | 481576 | 204293 | E529242F-7311-4038-A319-F62CDD4EDEBD | 183654 | Цукерки Брауні з кешю | 3552 | 1698.0 | 185 | 314130 |
| 16 | 481577 | 204293 | E529242F-7311-4038-A319-F62CDD4EDEBD | 167727 | До мяса | 3553 | 226.0 | 250 | 56500 |
| 17 | 481578 | 204293 | E529242F-7311-4038-A319-F62CDD4EDEBD | 167837 | Перець мелений В/С | 3553 | 246.0 | 400 | 98400 |

Рисунок 2.1 – Приклад транзакцій в базі даних

Для аналізу розподілу замовлень за загальним обсягом продажів, необхідно провести їх візуалізацію. Це дозволить більш наочно зрозуміти, як різні продукти розподіляються за обсягами продажів. На рисунку 2.2 зображено розподіл загального обсягу продажів продуктів, що надає можливість побачити тенденції та визначити, які саме товари генерують найбільше доходу.

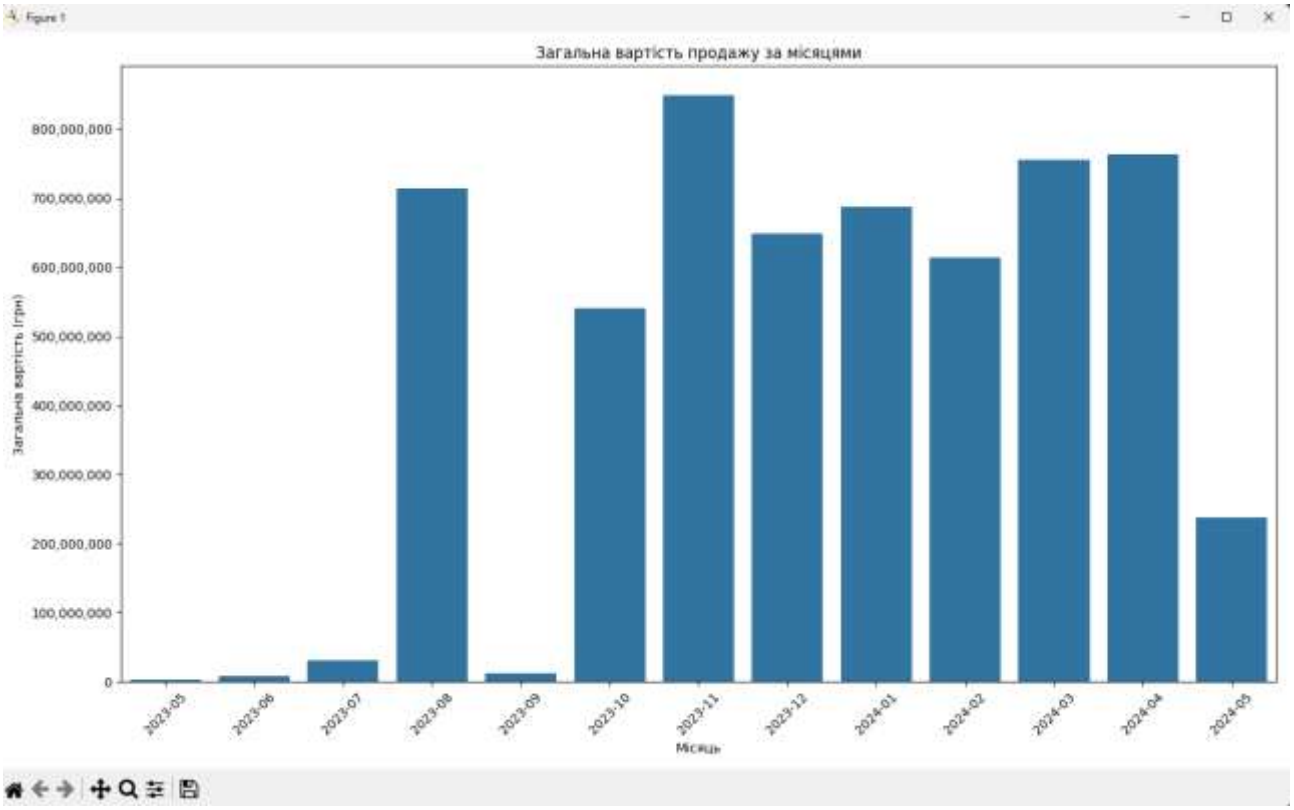



Рисунок 2.2 – Розподіл загального обсягу продажів продуктів

Далі, на рисунку 2.3, представлені найбільш популярні товари. Цей графік дозволяє оцінити, які продукти мають найбільший попит серед клієнтів. Така візуалізація допомагає зрозуміти, які товари варто тримати в постійному асортименті, а також які з них слід підкреслювати у рекламних кампаніях. Аналіз популярності товарів є важливим елементом для підтримки конкурентоспроможності на ринку, оскільки дає можливість ефективніше реагувати на зміну споживчих запитів та попиту.

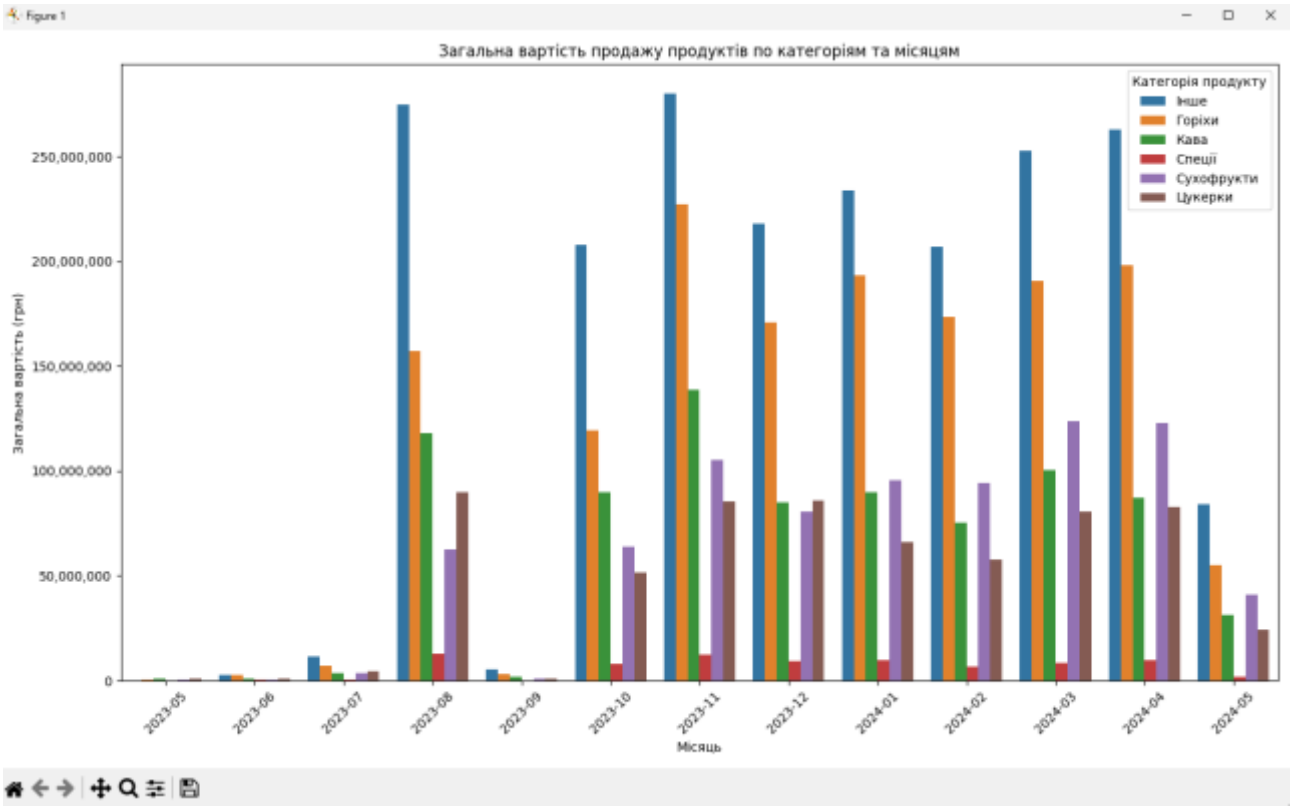


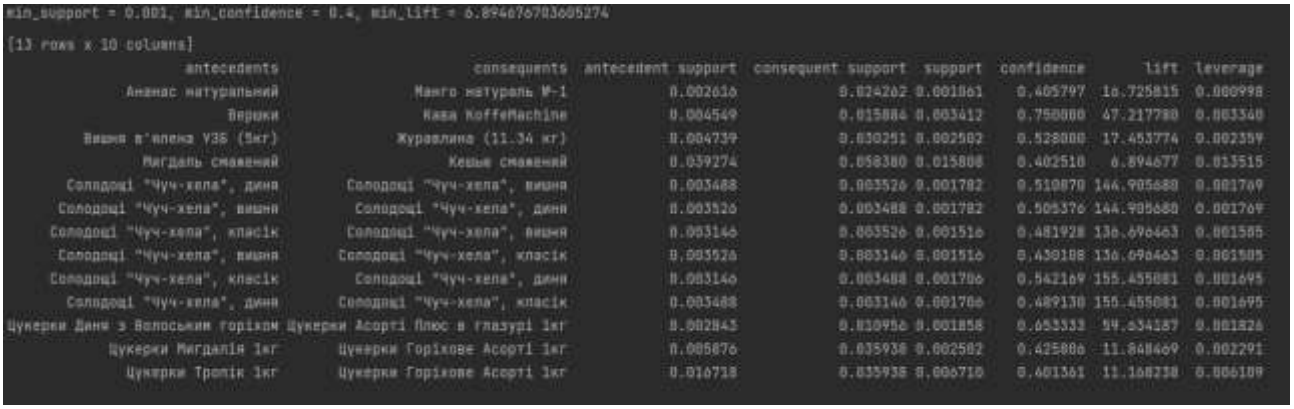

Рисунок 2.3 – Загальна вартість продажу продуктів по категоріям та місяцям

На графіку відображено, що споживачі часто придбають товари з категорії "Інше", такі як напої, масла, олії та інші продукти. Однак варто звернути увагу, що також спостерігається значна кількість продажів товарів з категорії "Горіхи". Це свідчить про те, що обидві ці категорії товарів є популярними серед покупців і мають стабільний попит.

## 2.3 Розробка алгоритмів Apriori, FP–Growth, ECLAT та їх порівняння

Щоб побудувати асоціативні правила, ми використовували алгоритми Apriori та FP-Growth, застосовуючи вбудовані модулі бібліотеки mlxtend.frequent_patterns та mlxtend.preprocessing. На початку ми обрали алгоритм Apriori, задавши мінімальне значення підтримки (support) на рівні 0.001. Залишаючи цей параметр незмінним, ми будемо аналізувати правила за допомогою метрики довіри (confidence). Спочатку будемо використовувати

поріг мінімальної довіри на рівні 0.4, а потім зменшимо цей поріг до 0.1. В усіх випадках ми виводимо лише ті правила, де значення метрики підсилення (lift) перевищує 1, представлено на рисунку 2.4

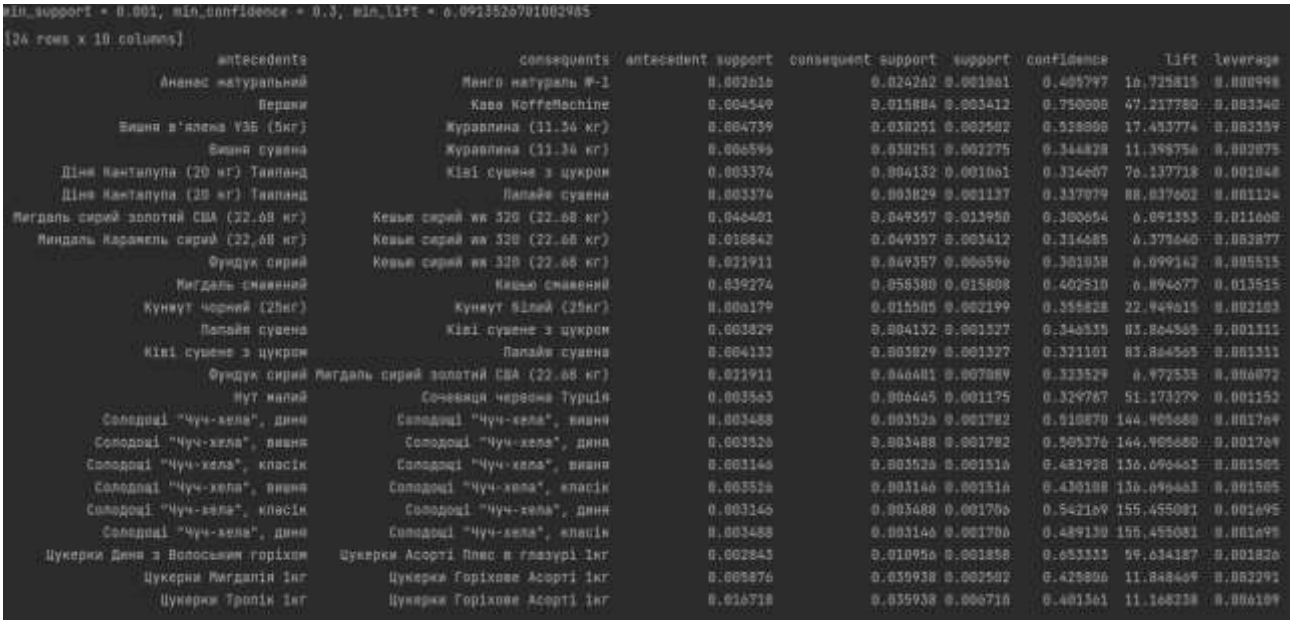


```
min_support = 0.001, min_confidence = 0.4, min_lift = 6.894676703605274
[13 rows x 10 columns]
```

| antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage |
|---|---|---|---|---|---|---|---|
| Ананас натуральний | Манго натураль М-1 | 0.002616 | 0.024202 | 0.001061 | 0.405797 | 16.725815 | 0.000998 |
| Вершки | Кава KoffeMachine | 0.004549 | 0.015884 | 0.003412 | 0.750000 | 47.217780 | 0.003340 |
| Вишня в'ялена У3Б (5кг) | Журавлина (11.34 кг) | 0.004739 | 0.030251 | 0.002502 | 0.528000 | 17.453774 | 0.002359 |
| Мигдаль смажений | Кешью смажений | 0.039274 | 0.058380 | 0.015808 | 0.402510 | 6.894677 | 0.013515 |
| Солодощі "Чуч-хела", диня | Солодощі "Чуч-хела", вишня | 0.003488 | 0.003526 | 0.001782 | 0.510870 | 144.905680 | 0.001769 |
| Солодощі "Чуч-хела", вишня | Солодощі "Чуч-хела", диня | 0.003526 | 0.003488 | 0.001782 | 0.505376 | 144.905680 | 0.001769 |
| Солодощі "Чуч-хела", класік | Солодощі "Чуч-хела", вишня | 0.003146 | 0.003526 | 0.001516 | 0.481928 | 136.696463 | 0.001505 |
| Солодощі "Чуч-хела", вишня | Солодощі "Чуч-хела", класік | 0.003526 | 0.003146 | 0.001516 | 0.430108 | 136.696463 | 0.001505 |
| Солодощі "Чуч-хела", класік | Солодощі "Чуч-хела", диня | 0.003146 | 0.003488 | 0.001706 | 0.542169 | 155.455081 | 0.001695 |
| Солодощі "Чуч-хела", диня | Солодощі "Чуч-хела", класік | 0.003488 | 0.003146 | 0.001706 | 0.489130 | 155.455081 | 0.001695 |
| Цукерки Диня з Волоським горіхом | Цукерки Асорті Плюс в глазурі 1кг | 0.002843 | 0.010956 | 0.001858 | 0.653333 | 59.634187 | 0.001826 |
| Цукерки Мигдалія 1кг | Цукерки Горіхове Асорті 1кг | 0.005876 | 0.035938 | 0.002502 | 0.425806 | 11.848469 | 0.002291 |
| Цукерки Тропік 1кг | Цукерки Горіхове Асорті 1кг | 0.010718 | 0.035938 | 0.006710 | 0.401561 | 11.168238 | 0.006109 |

Рисунок 2.4 – Правила з використанням метрики confidence і порогом 0.4 для алгоритму Apriori

На другому етапі ми знизили поріг довіри до 0.3. Це дозволяє врахувати правила, які мають нижчу ймовірність спільної появи, але все ще можуть бути корисними для аналізу. Знову ж таки, ми враховували тільки ті правила, де значення підсилення більше 1, що вказує на наявність значущої залежності між елементами, представлено на рисунку 2.5.

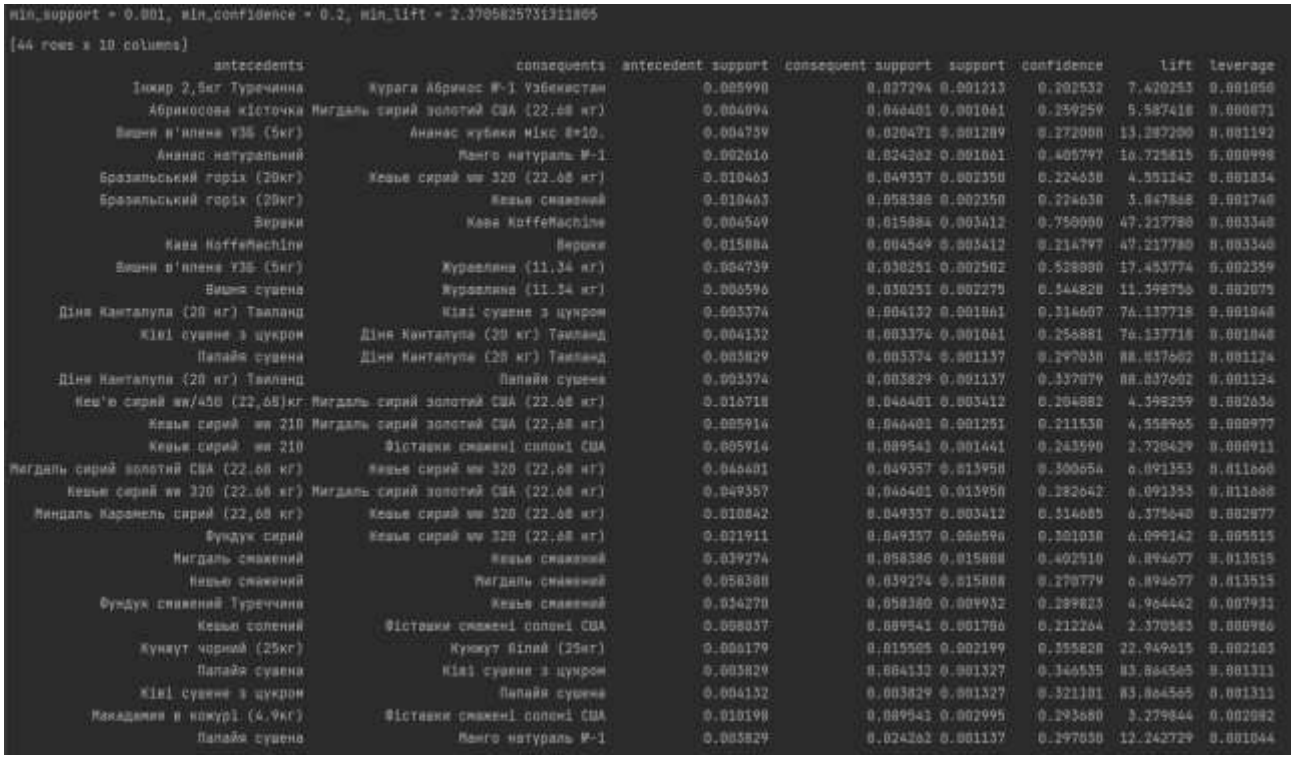


```
min_support = 0.001, min_confidence = 0.3, min_lift = 6.091352670108298
[24 rows x 10 columns]
```

| antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage |
|---|---|---|---|---|---|---|---|
| Ананас натуральний | Манго натураль М-1 | 0.002616 | 0.024202 | 0.001061 | 0.405797 | 16.725815 | 0.000998 |
| Вершки | Кава KoffeMachine | 0.004549 | 0.015884 | 0.003412 | 0.750000 | 47.217780 | 0.003340 |
| Вишня в'ялена У3Б (5кг) | Журавлина (11.34 кг) | 0.004739 | 0.030251 | 0.002502 | 0.528000 | 17.453774 | 0.002359 |
| Вишня сушена | Журавлина (11.34 кг) | 0.006596 | 0.030251 | 0.002275 | 0.344828 | 11.398756 | 0.002075 |
| Диня Канталупа (20 кг) Таїланд | Ківі сушене з цукром | 0.003374 | 0.004132 | 0.001061 | 0.314607 | 76.137718 | 0.001048 |
| Диня Канталупа (20 кг) Таїланд | Папайя сушена | 0.003374 | 0.003829 | 0.001137 | 0.337079 | 88.037602 | 0.001124 |
| Мигдаль сирий золотий США (22.68 кг) | Кешью сирий ww 320 (22.68 кг) | 0.046401 | 0.049357 | 0.013950 | 0.300654 | 6.091353 | 0.011660 |
| Мигдаль Карамель сирий (22,68 кг) | Кешью сирий ww 320 (22.68 кг) | 0.016842 | 0.049357 | 0.003412 | 0.314685 | 6.375040 | 0.002877 |
| Фундук сирий | Кешью сирий ww 320 (22.68 кг) | 0.021911 | 0.049357 | 0.006596 | 0.301038 | 6.099162 | 0.005515 |
| Мигдаль смажений | Кешью смажений | 0.039274 | 0.058380 | 0.015808 | 0.402510 | 6.894677 | 0.013515 |
| Кунжут чорний (25кг) | Кунжут білий (25кг) | 0.006179 | 0.015505 | 0.002199 | 0.355828 | 22.949615 | 0.002103 |
| Папайя сушена | Ківі сушене з цукром | 0.003829 | 0.004132 | 0.001327 | 0.346535 | 83.864565 | 0.001311 |
| Ківі сушене з цукром | Папайя сушена | 0.004132 | 0.003829 | 0.001327 | 0.321101 | 83.864565 | 0.001311 |
| Фундук сирий | Мигдаль сирий золотий США (22.68 кг) | 0.021911 | 0.046401 | 0.007089 | 0.323529 | 6.972535 | 0.006072 |
| Нут малий | Сочевиця червона Turpla | 0.003563 | 0.006445 | 0.001175 | 0.329787 | 51.173279 | 0.001152 |
| Солодощі "Чуч-хела", диня | Солодощі "Чуч-хела", вишня | 0.003488 | 0.003526 | 0.001782 | 0.510870 | 144.905680 | 0.001769 |
| Солодощі "Чуч-хела", вишня | Солодощі "Чуч-хела", диня | 0.003526 | 0.003488 | 0.001782 | 0.505376 | 144.905680 | 0.001769 |
| Солодощі "Чуч-хела", класік | Солодощі "Чуч-хела", вишня | 0.003146 | 0.003526 | 0.001516 | 0.481928 | 136.696463 | 0.001505 |
| Солодощі "Чуч-хела", вишня | Солодощі "Чуч-хела", класік | 0.003526 | 0.003146 | 0.001516 | 0.430108 | 136.696463 | 0.001505 |
| Солодощі "Чуч-хела", класік | Солодощі "Чуч-хела", диня | 0.003146 | 0.003488 | 0.001706 | 0.542169 | 155.455081 | 0.001695 |
| Солодощі "Чуч-хела", диня | Солодощі "Чуч-хела", класік | 0.003488 | 0.003146 | 0.001706 | 0.489130 | 155.455081 | 0.001695 |
| Цукерки Диня з Волоським горіхом | Цукерки Асорті Плюс в глазурі 1кг | 0.002843 | 0.010956 | 0.001858 | 0.653333 | 59.634187 | 0.001826 |
| Цукерки Мигдалія 1кг | Цукерки Горіхове Асорті 1кг | 0.005876 | 0.035938 | 0.002502 | 0.425806 | 11.848469 | 0.002291 |
| Цукерки Тропік 1кг | Цукерки Горіхове Асорті 1кг | 0.010718 | 0.035938 | 0.006710 | 0.401561 | 11.168238 | 0.006109 |

Рисунок 2.5 – Правила з використанням метрики confidence і порогом 0.3 для алгоритму Apriori

Також було проведено аналіз з використанням метрики confidence і порогом 0.2 і задавши мінімальне значення підтримки (support) на рівні 0.001, представлено на рисунка 2.6



min_support = 0.001, min_confidence = 0.2, min_lift = 2.3705825731311805

[44 rows x 10 columns]

| antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage |
|---|---|---|---|---|---|---|---|
| Інжир 2,5кг Туреччина | Курага Абрикос М-1 Узбекистан | 0.005990 | 0.037294 | 0.001213 | 0.202532 | 7.420253 | 0.001050 |
| Абрикосова кісточка | Мигдаль сирий золотий США (22.68 кг) | 0.004094 | 0.046401 | 0.001061 | 0.259259 | 5.587418 | 0.000871 |
| Вишня в'ялена Y3G (5кг) | Ананас кубики мікс 8*10. | 0.004739 | 0.020471 | 0.001289 | 0.272000 | 13.287200 | 0.001192 |
| Ананас натуральний | Манго натураль М-1 | 0.002610 | 0.024262 | 0.001061 | 0.406797 | 16.725815 | 0.000998 |
| Бразильський горіх (20кг) | Кешью сирий ww 320 (22.68 кг) | 0.010463 | 0.049357 | 0.002350 | 0.224638 | 4.551142 | 0.001834 |
| Бразильський горіх (20кг) | Кешью смажений | 0.010463 | 0.058380 | 0.002350 | 0.224638 | 3.847868 | 0.001746 |
| Вершки | Кава KoffeMachine | 0.004549 | 0.015884 | 0.003412 | 0.750000 | 47.217780 | 0.003340 |
| Кава KoffeMachine | Вершки | 0.015884 | 0.004549 | 0.003412 | 0.214797 | 47.217780 | 0.003340 |
| Вишня в'ялена Y3G (5кг) | Журавлина (11.34 кг) | 0.004739 | 0.030251 | 0.002502 | 0.528000 | 17.453774 | 0.002359 |
| Вишня сушена | Журавлина (11.34 кг) | 0.006596 | 0.030251 | 0.002275 | 0.344828 | 11.398756 | 0.002075 |
| Діня Канталупа (20 кг) Таиланд | Ківі сушене з цукром | 0.003374 | 0.004132 | 0.001061 | 0.314607 | 76.137718 | 0.001048 |
| Ківі сушене з цукром | Діня Канталупа (20 кг) Таиланд | 0.004132 | 0.003374 | 0.001061 | 0.256881 | 76.137718 | 0.001048 |
| Папайя сушена | Діня Канталупа (20 кг) Таиланд | 0.003829 | 0.003374 | 0.001137 | 0.297030 | 88.037602 | 0.001124 |
| Діня Канталупа (20 кг) Таиланд | Папайя сушена | 0.003374 | 0.003829 | 0.001137 | 0.337079 | 88.037602 | 0.001124 |
| Кеш'ю сирий ww/450 (22,68)кг | Мигдаль сирий золотий США (22.68 кг) | 0.016718 | 0.046401 | 0.003412 | 0.204082 | 4.398259 | 0.002636 |
| Кешью сирий ww 210 | Мигдаль сирий золотий США (22.68 кг) | 0.005914 | 0.046401 | 0.001251 | 0.211538 | 4.558965 | 0.000977 |
| Кешью сирий ww 210 | Фісташки смажені солоні США | 0.005914 | 0.009541 | 0.001441 | 0.243590 | 2.720429 | 0.000911 |
| Мигдаль сирий золотий США (22.68 кг) | Кешью сирий ww 320 (22.68 кг) | 0.046401 | 0.049357 | 0.013950 | 0.300654 | 6.091353 | 0.011668 |
| Кешью сирий ww 320 (22.68 кг) | Мигдаль сирий золотий США (22.68 кг) | 0.049357 | 0.046401 | 0.013950 | 0.282642 | 6.091353 | 0.011668 |
| Мигдаль Карамель сирий (22,68 кг) | Кешью сирий ww 320 (22.68 кг) | 0.010842 | 0.049357 | 0.003412 | 0.314685 | 6.375040 | 0.002877 |
| Фундук сирий | Кешью сирий ww 320 (22.68 кг) | 0.021911 | 0.049357 | 0.006596 | 0.301038 | 6.099142 | 0.005515 |
| Мигдаль смажений | Кешью смажений | 0.039274 | 0.058380 | 0.015808 | 0.402510 | 6.894677 | 0.013515 |
| Кешью смажений | Мигдаль смажений | 0.058380 | 0.039274 | 0.015808 | 0.270779 | 6.894677 | 0.013515 |
| Фундук смажений Туреччина | Кешью смажений | 0.034270 | 0.058380 | 0.009932 | 0.289823 | 4.964462 | 0.007931 |
| Кешью солений | Фісташки смажені солоні США | 0.008857 | 0.009541 | 0.001706 | 0.212264 | 2.370585 | 0.000986 |
| Кунжут чорний (25кг) | Кунжут білий (25кг) | 0.008179 | 0.015505 | 0.002199 | 0.355828 | 22.949615 | 0.002103 |
| Папайя сушена | Ківі сушене з цукром | 0.003829 | 0.004132 | 0.001327 | 0.346535 | 83.864565 | 0.001311 |
| Ківі сушене з цукром | Папайя сушена | 0.004132 | 0.003829 | 0.001327 | 0.321101 | 83.864565 | 0.001311 |
| Макадамія в кожурі (4.9кг) | Фісташки смажені солоні США | 0.010198 | 0.009541 | 0.002995 | 0.293680 | 3.279844 | 0.002082 |
| Папайя сушена | Манго натураль М-1 | 0.003829 | 0.024262 | 0.001137 | 0.297030 | 12.242729 | 0.001044 |

Рисунок 2.6 – Правила з використанням метрики confidence і порогом 0.2 для алгоритму Apriori

Також було проведено аналіз з використанням метрики confidence і порогом 0.1, представлено на рисунка 2.7

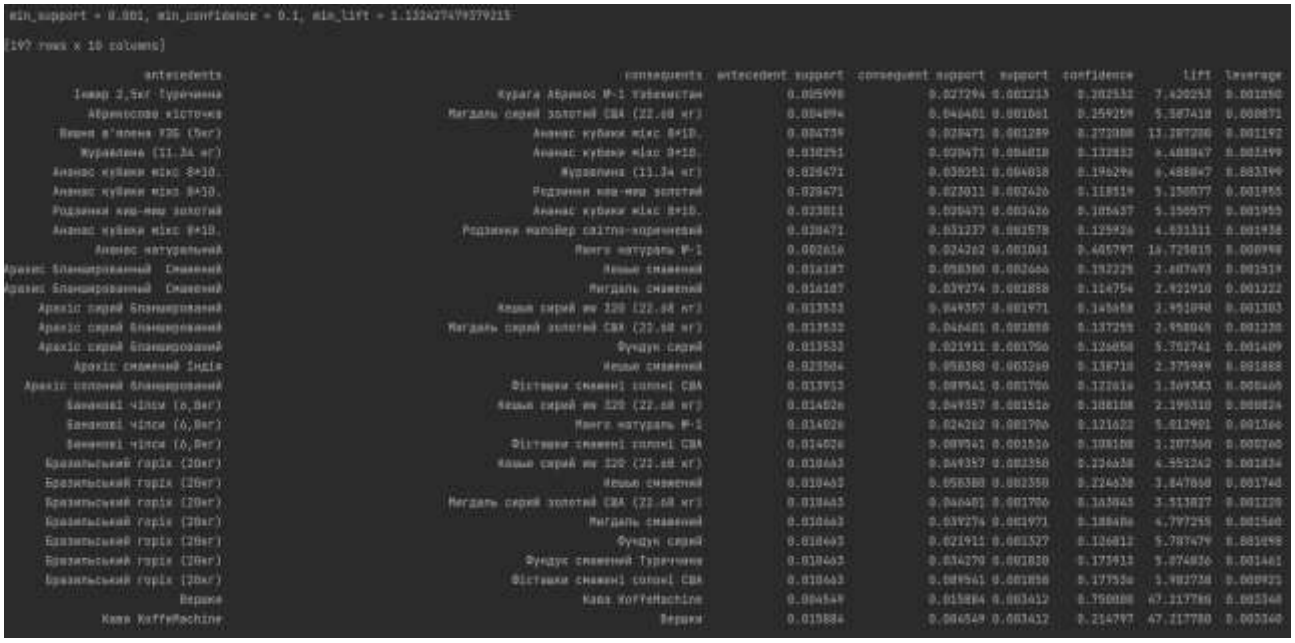



Рисунок 2.7 – Правила з використанням метрики confidence і порогом 0.1 для алгоритму Apriori

За алгоритмом FP-Growth було проаналізовано дані де ми задали мінімальне значення підтримки на рівні 0.01 та за допомогою метрики довіри (confidence). Спочатку будемо використовувати поріг мінімальної довіри на рівні 0.4, а потім зменшимо це поріг до 0.1, аналіз з порогом 0.4 та значення підтримки представлено на рисунку 2.8.

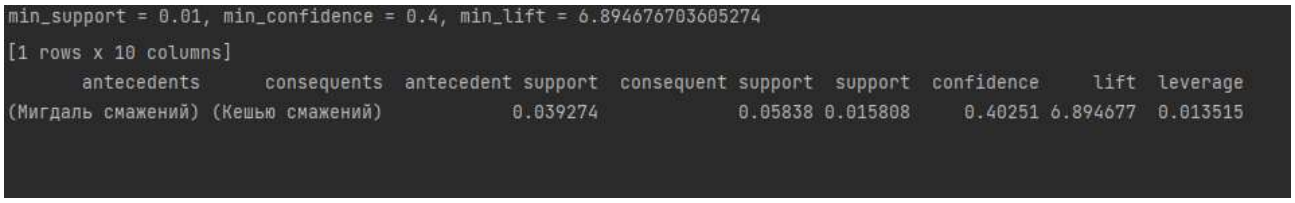



Рисунок 2.8 – Правила з використанням метрики confidence і порогом 0.4 для алгоритму FP-Growth

Далі ми змінимо поріг до 0.3, представлено на рисунку 2.9.


Рисунок 2.9 – Правила з використанням метрики confidence і порогом 0.3 для алгоритму FP-Growth

Після цього ми змінимо поріг до 0.2 та зменшимо мінімальне значення підтримки до 0.0001, приклад представлено на рисунку 2.10

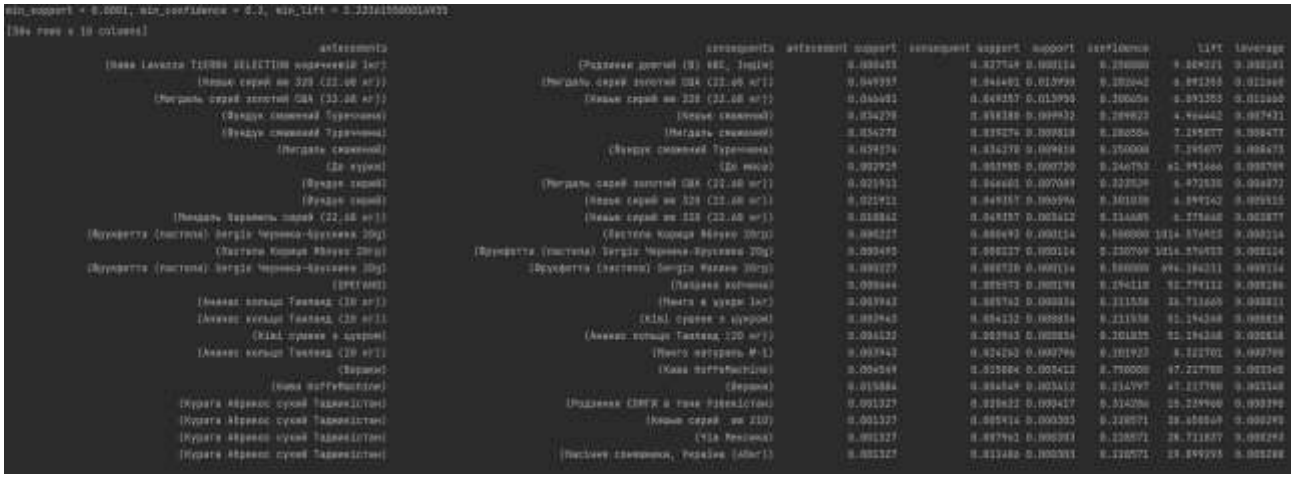


Рисунок 2.10 – Правила з використанням метрики confidence і порогом 0.2 для алгоритму FP-Growth

Останнім зміненням ми змінімо поріг до 0.1 та збільшимо мінімальне значення підтримки до 0.001, приклад представлено на рисунку 2.11

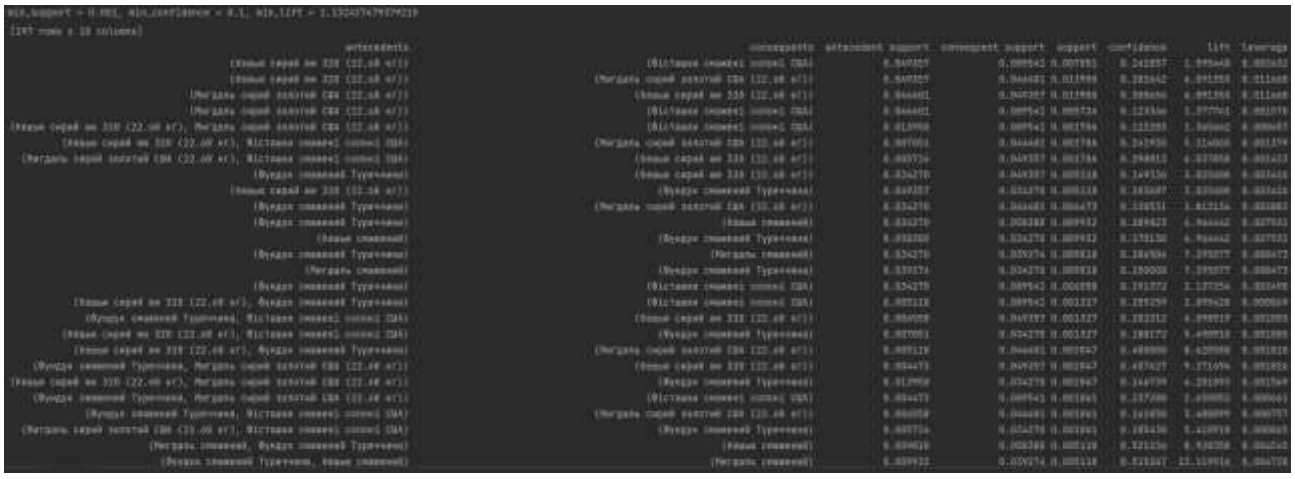


Рисунок 2.11 – Правила з використанням метрики confidence і порогом 0.1 для алгоритму FP-Growth

На рисунку 2.12 продемонстровано, як змінювалася кількість знайдених асоціативних правил для двох алгоритмів — FP-Growth та Apriori — залежно від мінімального порогу метрики "довіра" (confidence). Для кожного алгоритму було досліджено декілька значень порогу довіри, що дозволило оцінити, як цей параметр впливає на кількість виявлених правил. Графік чітко показує, що зі зменшенням мінімального порогу довіри, кількість знайдених правил

збільшується для обох алгоритмів, проте з різною інтенсивністю. Це дозволяє зробити висновки про ефективність кожного з алгоритмів у контексті аналізу асоціативних правил та вибрати найбільш підходящі параметри для їх побудови у конкретних умовах.

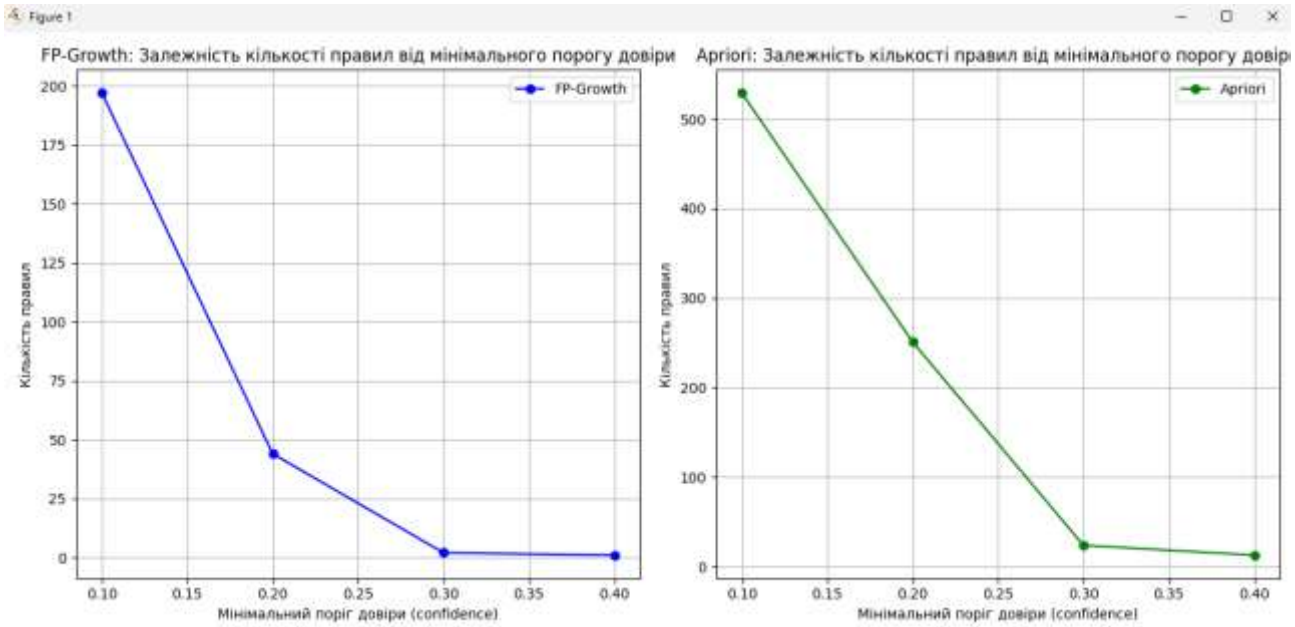



Рисунок 2.12 – Порівняння роботи алгоритмів FP-Growth та Apriori

Для реалізації алгоритму ECLAT було використано бібліотеку pyECLAT, яка забезпечує зручний інтерфейс для роботи з цим методом. У ході аналізу було встановлено мінімальну підтримку на рівні 0,01, що означає, що нас цікавили всі набори елементів, які зустрічалися у щонайменше 1% випадків у досліджуваному наборі даних. Цей поріг обрано для забезпечення балансу між точністю аналізу та обсягом оброблюваних даних, що дозволяє виявити як основні, так і менш очевидні зв’язки між елементами.

Результати розрахунку було детально представлено на рисунку 2.13, де наочно показано частоту зустрічальності виявлених наборів елементів, та у таблиці 2.1, яка містить конкретні значення підтримки для кожного з них. Ці результати дозволяють краще зрозуміти структуру даних і виявити ключові залежності між різними елементами, що може бути корисним для оптимізації діяльності організації та прийняття стратегічних рішень.

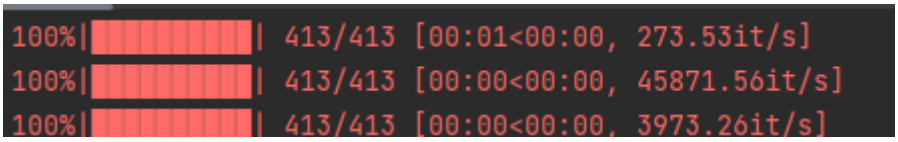



Рисунок 2.13 – Аналіз даних за допомогою алгоритму ECLAT

Таблиця 2.1

**Результат аналізу даних за допомогою алгоритму ECLAT**

| Назва товару | Support |
|---|---|
| Цукерки ASAL Фундучок 1кг & Цукерки Асорті Плюс в глазурі 1кг | 0.018604651162790697 |
| Кешью смажений & Фундук смажений Туреччина | 0.027906976744186046 |
| Кешью смажений & Мигдаль смажений | 0.01627906976744186 |
| Фундук смажений Туреччина & Мигдаль сирий золотий США (22.68 кг) | 0.013953488372093023 |
| Фундук смажений Туреччина & Мигдаль смажений | 0.018604651162790697 |
| Мигдаль сирий золотий США (22.68 кг) & Кешью сирий ww 320 (22.68 кг) | 0.027906976744186046 |
| Цукерки Асорті Плюс в глазурі 1кг & Цукерки Диня з Волоським горіхом | 0.011627906976744186 |

Алгоритм ECLAT відрізняється від інших методів аналізу асоціативних зв'язків, таких як алгоритми Apriori або FP-Growth, оскільки він спирається виключно на метрику мінімальної підтримки для виявлення частих наборів елементів. Цей підхід має свої особливості і обмеження, які роблять його незрівнянним з попередніми методами.

В рамках розробки цього програмного продукту було створено модель, яка за допомогою аналізу транзакційних баз даних і використання асоціативних правил виконує основну функцію рекомендаційної системи. Модель генерує найбільш релевантні продукти для користувача на основі його попередніх виборів. У аналізованому наборі даних для створення рекомендаційної системи містяться часті набори, які демонструють прийнятні значення за різними метриками, зокрема підтримка, де максимальне значення перевищує 0.4. Це дає змогу моделі рекомендаційної системи надавати високо релевантні результати.

Для побудови моделі рекомендаційної системи був обраний алгоритм Apriori. Його основними перевагами є простота реалізації, зручність у підтримці та легкість у розумінні, що робить його більш доступним і надійним для використання порівняно з іншими алгоритмами.

Було проведено глибокий аналіз трьох алгоритмів виявлення асоціативних правил: FP-Growth, ECLAT та Apriori. Кожен з них має свої особливості і застосування, що дозволяє оптимізувати процес аналізу транзакційних даних та виявлення важливих закономірностей.

1. FP-Growth:
2. ECLAT:
3. Apriori:

У результаті проведеного аналізу можна зробити висновок, що кожен з розглянутих алгоритмів має свої переваги і недоліки, що обумовлює їхнє специфічне застосування в різних контекстах аналізу даних. FP-Growth є оптимальним для обробки великих обсягів даних завдяки своїй ефективності у використанні пам'яті і швидкості обчислень. ECLAT підходить для швидкого і точного аналізу у випадках великої кількості транзакцій з обмеженою кількістю елементів. Apriori є корисним для глибокого аналізу менших наборів даних, забезпечуючи прозорість і надійність результатів.

Проведений аналіз алгоритмів дозволяє виділити ключові аспекти, які впливають на вибір методу для виявлення асоціативних правил у різних бізнес-

контекстах. FP-Growth ідеально підходить для обробки великих транзакційних баз, забезпечуючи високу швидкість і ефективність обчислень. ECLAT демонструє високу продуктивність у випадках з великою кількістю транзакцій, де потрібно швидко і точно визначити часті набори товарів. Apriori є найбільш ефективним для початкового аналізу даних і розуміння основних закономірностей у випадках з невеликим обсягом даних.

Використання цих алгоритмів дозволяє компаніям глибше розуміти структуру і динаміку транзакційних даних, визначати важливі закономірності та створювати ефективні бізнес-стратегії, які сприятимуть підвищенню конкурентоспроможності і успішному розвитку на сучасному ринку.

# ВИСНОВОК

У сучасному динамічному та конкурентному середовищі роздрібна торгівля потребує ефективних підходів до управління, які включають аналіз великих обсягів даних, оптимізацію бізнес-процесів та впровадження інноваційних технологій.

На основі результатів аналізу історії транзакцій та аналізу отриманих результатів було розроблено модель рекомендаційної системи, яка дозволяє генерувати релевантні рекомендації для користувачів на основі аналізу асоціативних правил у базі даних транзакцій. Ця модель може продемонструвати свою ефективність у поліпшенні споживчого досвіду та підвищенні рівня задоволеності клієнтів, що є важливим фактором успіху на конкурентному ринку.

Порівняльний аналіз алгоритмів Apriori, ECLAT та FP-Growth показав, що кожен з них має свої переваги і недоліки. Алгоритм Apriori, хоча і є найповільнішим, виявився найбільш зручним для реалізації та підтримки. ECLAT виявився більш швидким завдяки ефективному представленню даних, а FP-Growth продемонстрував найвищу продуктивність завдяки використанню структури FP-дерева. Незважаючи на це, для розробки моделі було обрано алгоритм Apriori, враховуючи його простоту та доступність.

Впровадження механізму виявлення асоціативних правил в якості системи рекомендацій здатне з одного боку підвищити ступінь задоволеності клієнтів, а з іншого допомагає формувати перспективні товарні набори, які матимуть успішний попит з боку клієнтів.

Таким чином, проведений системний аналіз та розробка рекомендаційної системи на основі асоціативних правил дозволили значно покращити розуміння діяльності організацій роздрібної торгівлі та запропонувати конкретні шляхи для їх подальшого розвитку і підвищення конкурентоспроможності на ринку. Важливість такого підходу полягає у можливості швидко адаптуватися до змін

на ринку, ефективно реагувати на потреби споживачів і, як результат, забезпечувати сталий розвиток і успішне функціонування роздрібної організації в умовах сучасної економіки.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Association rule learning. URL: https://en.wikipedia.org/wiki/Association_rule_learning (дата звернення: 02.05.2024)

2. Usage of association rules URL: https://www.ibm.com/docs/en/ias?topic=rules-usage (дата звернення: 02.05.2024)

3.Алгоритм Apriori URL: https://en.wikipedia.org/wiki/Apriori_algorithm (дата звернення: 14.05.2024)

4. Алгоритм Apriori URL: https://www.geeksforgeeks.org/apriori-algorithm/ (дата звернення: 13.05.2024)

5. Асоціативні правила URL: https://habr.com/ru/companies/ods/articles/353502/ (дата звернення: 12.05.2024)

6. Алгоритм FP-Growth URL: https://en.wikipedia.org/wiki/Frequent_pattern_discovery (дата звернення: 03.05.2024)

7. Алгоритм FP-Growth URL: https://www.geeksforgeeks.org/frequent-pattern-growth-algorithm/ (дата звернення: 15.05.2024)

8. Модифікація алгоритму FP Growth або як правильно доглядати свої дерева URL: https://habr.com/ru/articles/741024/ (дата звернення: 12.05.2024)

9. Алгоритм ECLAT URL: https://www.geeksforgeeks.org/ml-eclat-algorithm/ (дата звернення: 14.05.2024)

10. Python (programming language) URL: https://en.wikipedia.org/wiki/Python_(programming_language) (дата звернення: 01.05.2024)

11. Pros and Cons of Python Programming Language URL: https://serokell.io/blog/python-pros-and-cons (дата звернення: 10.05.2024)

12. Pandas (software) URL: https://en.wikipedia.org/wiki/Pandas_(software) (дата звернення: 10.05.2024)

13. SQLite URL: https://en.wikipedia.org/wiki/SQLite (дата звернення: 11.05.2024)

14. Matplotlib URL: https://en.wikipedia.org/wiki/Matplotlib (дата звернення: 11.05.2024)

15. Apriori URL: https://pypi.org/project/efficient-apriori/ (дата звернення: 11.05.2024)

16. FP Growth URL: https://github.com/RohiBaner/FPTree (дата звернення: 12.05.2024)

17. ECLAT URL: https://coursehunter.net/course/mashinnoe-obuchenie-ot-a-do-ya-prakticheskiy-python-i-r-v-data-science (дата звернення: 12.05.2024)

# ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

| № з/п | Позначення | Найменування | Кількість аркушів | Примітки |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | Документація | | |
| 3 | | | | |
| 4 | САУ.КР.24.03.ПЗ | Пояснювальна записка | | Формат А4 |
| 5 | | | | |
| 6 | | Демонстраційний матеріал | N2 | Презентація на CD-R |
| 7 | | | | |
| 8 | | Копія роботи | 1 | Диск CD-R |
| 9 | | | | |
| 10 | | | | |
| 11 | | | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |
| 18 | | | | |

| | | | | | САУ.КР.24.05.ДА.ПЗ. | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Змін. | Аркуш | № докум. | Підпис | Дата | | | | |
| Розроб. | | Кормич Д.Г. | | | **Матеріали кваліфікаційної роботи** | Літ. | Аркуш | Аркушів |
| К. розд. | | Гаранжа Д.М. | | | | | | |
| Керівн. | | Гаранжа Д.М., | | | | НТУ «ДП», 12; 124-21ск-1 | | |
| Н.контр. | | Хом’як Т.В. | | | | | | |
| Зав. каф. | | Желдак Т.А. | | | | | | |

ДОДАТОК Б Відгук

# ДОДАТОК В ЛІСТІНГ ПРОГРАМНОГО ПРОДУКТУ

Лістинг Б.1 – Код програми для аналізу алгоритму Apriori

```
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt
import seaborn as sns
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

# Підключення до бази даних
conn = sqlite3.connect('database.db')

# Завантаження даних з бази даних у DataFrame, включаючи
поле Date
df = pd.read_sql_query("SELECT * FROM store", conn)

# Перетворення поля Date у формат datetime
df['Date'] = pd.to_datetime(df['Date'])

# Додавання поля для місяця та року
df['Month'] = df['Date'].dt.to_period('M')

# Огляд даних
print(df.head())

# Визначення популярних продуктів (наприклад, топ-10 за
кількістю продажів)
top_products                                          =
df.groupby('ProductName')['Qty'].sum().nlargest(10).reset_ind
ex()
print(top_products)

# Фільтрування даних для популярних продуктів
popular_products_df                                   =
df[df['ProductName'].isin(top_products['ProductName'])]

# Аналіз даних: загальний обсяг продажів по популярних
продуктах
sales_by_popular_product                              =
popular_products_df.groupby('ProductName')['Sum'].sum().reset
_index()
print(sales_by_popular_product)

# Побудова графіка продажів по популярних продуктах
plt.figure(figsize=(10, 6))
sns.barplot(x='Sum',                    y='ProductName',
data=sales_by_popular_product.sort_values('Sum',
ascending=False))
```

```
    plt.title('Загальний обсяг продажів по популярних
продуктах')
    plt.xlabel('Сума продажів')
    plt.ylabel('Назва продукту')
    plt.show()

    # Аналіз даних: загальний обсяг продажів по місяцях для
популярних продуктів
    sales_by_month_product =
popular_products_df.groupby(['Month',
'ProductName'])['Sum'].sum().reset_index()

    # Перетворення 'Month' на формат 'datetime' збереженим
як рік та місяць
    sales_by_month_product['Month'] =
sales_by_month_product['Month'].dt.to_timestamp()

    # Pivot таблиця для створення стекової діаграми
    sales_pivot =
sales_by_month_product.pivot(index='Month',
columns='ProductName', values='Sum').fillna(0)

    # Побудова стекової діаграми
    plt.figure(figsize=(14, 8))
    sales_pivot.plot(kind='bar', stacked=True, figsize=(14,
8), colormap='tab20')
    plt.title('Обсяги продажів по місяцях для популярних
продуктів')
    plt.xlabel('Місяць')
    plt.ylabel('Сума продажів')
    plt.xticks(rotation=45)
    plt.legend(title='Назва продукту', bbox_to_anchor=(1.05,
1), loc='upper left')
    plt.tight_layout()
    plt.show()

    # Аналіз даних: загальний обсяг продажів по DocId для
популярних продуктів
    sales_by_docid_popular =
popular_products_df.groupby('DocId')['Sum'].sum().reset_index
()
    print(sales_by_docid_popular)

    # Побудова графіка продажів по DocId для популярних
продуктів
    plt.figure(figsize=(10, 6))
    sns.lineplot(x='DocId', y='Sum',
data=sales_by_docid_popular)
    plt.title('Загальний обсяг продажів по DocId для
популярних продуктів')
    plt.xlabel('DocId')
```

```
    plt.ylabel('Сума продажів')
    plt.show()

    # Завантаження даних з бази даних у DataFrame для
Apriori
    query = "SELECT DocId, ProductName FROM store"
    df = pd.read_sql_query(query, conn)

    # Підготовка даних для алгоритму Apriori
    # Створення таблиці транзакцій
    basket = (df.groupby(['DocId',
'ProductName'])['ProductName']
              .count().unstack().reset_index().fillna(0)
              .set_index('DocId'))

    # Перетворення значень на булеві
    def encode_units(x):
        return 0 if x <= 0 else 1

    basket = basket.applymap(encode_units).astype(bool)

    # Виконання алгоритму Apriori для різних параметрів
    def run_apriori(basket, min_support, min_confidence,
max_len=2):
        frequent_itemsets = apriori(basket,
min_support=min_support, use_colnames=True, max_len=max_len)
        if frequent_itemsets.empty:
            print(f'No frequent itemsets found for
min_support = {min_support}')
            return frequent_itemsets, pd.DataFrame()

        rules = association_rules(frequent_itemsets,
metric='confidence', min_threshold=min_confidence)
        if rules.empty:
            print(f'No rules found for min_support =
{min_support} and min_confidence = {min_confidence}')
            return frequent_itemsets, rules

        minLift = rules['lift'].min()
        print(f'min_support = {min_support}, min_confidence
= {min_confidence}, min_lift = {minLift}')
        filtered_rules = rules[rules['lift'] > 1]
        return frequent_itemsets, filtered_rules

    # Параметри для запуску
    parameters = [
        (0.001, 0.4, 2),
        (0.001, 0.3, 2),
        (0.001, 0.2, 2),
        (0.001, 0.1, 10),
    ]
```

```
    # Запуск для кожної пари параметрів
    for min_support, min_confidence, max_len in parameters:
        frequent_itemsets,   rules   =   run_apriori(basket,
min_support, min_confidence, max_len)
        if not frequent_itemsets.empty:
            print(frequent_itemsets)
        if not rules.empty:
            print(rules)
            # Вивід у вигляді таблиці
            rules_display = rules[
                ['antecedents',  'consequents',  'antecedent
support',  'consequent  support',  'support',  'confidence',
'lift',
                 'leverage']]
            #  Конвертуємо  sets  до  стрічок  для  кращого
вигляду в таблиці
            rules_display['antecedents']                    =
rules_display['antecedents'].apply(lambda        x:        ',
'.join(list(x)))
            rules_display['consequents']                    =
rules_display['consequents'].apply(lambda        x:        ',
'.join(list(x)))
            print(rules_display.to_string(index=False))

    # Закриття підключення до бази даних
conn.close()
```

Лістинг Б.2 – Код програми для аналізу алгоритму FP-Growth

```
    import sqlite3
    import pandas as pd
    from    mlxtend.frequent_patterns    import    fpgrowth,
association_rules

    # Підключення до бази даних SQLite
    conn = sqlite3.connect('database.db')
    cursor = conn.cursor()

    # Завантаження даних з бази даних у DataFrame
    query = "SELECT DocId, ProductName FROM store"
    df = pd.read_sql_query(query, conn)

    # Підготовка даних для алгоритму FP-Growth
    # Створення таблиці транзакцій
    basket              =              (df.groupby(['DocId',
'ProductName'])['ProductName']
              .count().unstack().reset_index().fillna(0)
              .set_index('DocId'))
```

```
    # Перетворення значень на булеві
    def encode_units(x):
        return 0 if x <= 0 else 1


    basket = basket.applymap(encode_units).astype(bool)


    # Виконання алгоритму FP-Growth для різних параметрів
    def run_fpgrowth(basket, min_support, min_confidence,
max_len=2):
        frequent_itemsets = fpgrowth(basket,
min_support=min_support, use_colnames=True, max_len=max_len)
        if frequent_itemsets.empty:
            print(f'No frequent itemsets found for
min_support = {min_support}')
            return frequent_itemsets, pd.DataFrame()

        rules = association_rules(frequent_itemsets,
metric='confidence', min_threshold=min_confidence)
        if rules.empty:
            print(f'No rules found for min_support =
{min_support} and min_confidence = {min_confidence}')
            return frequent_itemsets, rules

        minLift = rules['lift'].min()
        print(f'min_support = {min_support}, min_confidence
= {min_confidence}, min_lift = {minLift}')
        filtered_rules = rules[rules['lift'] > 1]
        return frequent_itemsets, filtered_rules


    # Параметри для запуску
    parameters = [
        (0.01, 0.4, 2),
        (0.01, 0.3, 10),
        (0.0001, 0.2, 2),
        (0.001, 0.1, 10),
    ]

    # Запуск для кожної пари параметрів
    for min_support, min_confidence, max_len in parameters:
        frequent_itemsets, rules = run_fpgrowth(basket,
min_support, min_confidence, max_len)
        if not frequent_itemsets.empty:
            print(frequent_itemsets)
        if not rules.empty:
            print(rules)
            # Вивід у вигляді таблиці
            rules_display = rules[
```

```
                ['antecedents', 'consequents', 'antecedent
support', 'consequent support', 'support', 'confidence',
'lift',
                 'leverage']]
            print(rules_display.to_string(index=False))

    # Закриття підключення до бази даних
conn.close()
```

Лістинг Б.3 – Код програми для аналізу алгоритму ECLAT

```
    import sqlite3
    import pandas as pd
    from pyECLAT import ECLAT
    import time

    # Підключення до бази даних SQLite
    conn = sqlite3.connect('database.db')
    cursor = conn.cursor()

    # Завантаження даних з бази даних у DataFrame
    # Вибираємо лише перші 1000 записів
    query = "SELECT DocId, ProductName FROM store LIMIT
1000"
    df = pd.read_sql_query(query, conn)

    # Закриття підключення до бази даних
    conn.close()

    # Перетворення даних у формат списку списків
    transactions =
df.groupby('DocId')['ProductName'].apply(list).values.tolist(
)

    # Конвертуємо списки транзакцій у DataFrame для pyECLAT
    # У кожному рядку DataFrame буде одна транзакція
    data = pd.DataFrame(transactions)

    # Визначення мінімальної кількості продуктів у наборі та
мінімальної підтримки
    min_n_products = 2
    min_support = 0.01
    max_length = max([len(x) for x in transactions])  #
Максимальна довжина транзакції

    # Створення об'єкта ECLAT
    my_eclat = ECLAT(data=data, verbose=True)

    # Виконання ECLAT
    start_time = time.time()
```

```
    rule_indices, rule_supports = my_eclat.fit(
        min_support=min_support,
        min_combination=min_n_products,
        max_combination=min_n_products      #   Змінено   для
покращення продуктивності
    )

    end_time = time.time()

    # Виведення результатів
    print("Індекси правил:", rule_indices)
    print("Підтримка правил:", rule_supports)
    print(f"Час виконання: {end_time - start_time} секунд")

    # Додатковий аналіз і виведення правил
    rules = my_eclat.rules_as_dataframe
    print(rules)

    # Приклад додаткового фільтрування правил за підтримкою
    filtered_rules = rules[rules['support'] >= 0.02]
print(filtered_rules)
```

Лістинг Б.4 – Код програми для підключення бази даних

```
    import sqlite3
    import pandas as pd

    excel_file = 'Store.xlsx'
    df = pd.read_excel(excel_file)

    # Выводим имена столбцов, чтобы убедиться, что они
правильные
    print(df.columns)

    # Подключение к базе данных SQLite
    conn = sqlite3.connect('database.db')
    cursor = conn.cursor()

    # Создание таблицы, если она не существует
    cursor.execute('''CREATE TABLE IF NOT EXISTS store (
        id INTEGER,
        DocId INTEGER,
        ProductID INTEGER,
        Date VARCHAR(255),
        ProductName VARCHAR(255),
        Unit INTEGER,
        Qty REAL,
        Price INTEGER,
        Sum INTEGER)''')
    conn.commit()
```

```
    # Заполнение таблицы данными из DataFrame
    df.to_sql('store',       conn,       if_exists='append',
index=False)
conn.close()
```

Лістинг Б.5 – Код програми для аналізу всіх алгоритмів

```
    import sqlite3
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from mlxtend.frequent_patterns import fpgrowth, apriori,
association_rules

    # Підключення до бази даних SQLite
    conn = sqlite3.connect('database.db')
    cursor = conn.cursor()

    # Завантаження даних з бази даних у DataFrame
    df = pd.read_sql_query("SELECT DocId, ProductName, Qty,
Sum FROM store", conn)

    # Огляд даних
    print(df.head())

    # Визначення популярних продуктів (наприклад, топ-10 за
кількістю продажів)
    top_products                                          =
df.groupby('ProductName')['Qty'].sum().nlargest(10).reset_ind
ex()
    print(top_products)

    # Фільтрування даних для популярних продуктів
    popular_products_df                                   =
df[df['ProductName'].isin(top_products['ProductName'])]

    # Аналіз даних: загальний обсяг продажів по популярних
продуктах
    sales_by_popular_product                              =
popular_products_df.groupby('ProductName')['Sum'].sum().reset
_index()
    print(sales_by_popular_product)

    # Побудова графіка продажів по популярних продуктах
    plt.figure(figsize=(10, 6))
    sns.barplot(x='Sum',                    y='ProductName',
data=sales_by_popular_product.sort_values('Sum',
ascending=False))
    plt.title('Загальний   обсяг   продажів   по   популярних
продуктах')
    plt.xlabel('Сума продажів')
```

```
    plt.ylabel('Назва продукту')
    plt.show()

    # Аналіз даних: загальний обсяг продажів по DocId для популярних продуктів
    sales_by_docid_popular = popular_products_df.groupby('DocId')['Sum'].sum().reset_index()
    print(sales_by_docid_popular)

    # Побудова графіка продажів по DocId для популярних продуктів
    plt.figure(figsize=(10, 6))
    sns.lineplot(x='DocId', y='Sum', data=sales_by_docid_popular)
    plt.title('Загальний обсяг продажів по DocId для популярних продуктів')
    plt.xlabel('DocId')
    plt.ylabel('Сума продажів')
    plt.show()

    # Підготовка даних для алгоритмів
    basket = (df.groupby(['DocId', 'ProductName'])['ProductName']
              .count().unstack().reset_index().fillna(0)
              .set_index('DocId'))


    # Перетворення значень на булеві
    def encode_units(x):
        return 0 if x <= 0 else 1


    basket = basket.applymap(encode_units).astype(bool)


    # Функція для виконання FP-Growth
    def run_fpgrowth(basket, min_support, min_confidence, max_len=2):
        frequent_itemsets = fpgrowth(basket, min_support=min_support, use_colnames=True, max_len=max_len)
        if frequent_itemsets.empty:
            print(f'No frequent itemsets found for min_support = {min_support}')
            return frequent_itemsets, pd.DataFrame()

        rules = association_rules(frequent_itemsets, metric='confidence', min_threshold=min_confidence)
        if rules.empty:
            print(f'No rules found for min_support = {min_support} and min_confidence = {min_confidence}')
```

```
            return frequent_itemsets, rules

        filtered_rules = rules[rules['lift'] > 1]
        return frequent_itemsets, filtered_rules


    # Функція для виконання Apriori
    def run_apriori(basket, min_support, min_confidence,
max_len=2):
        frequent_itemsets = apriori(basket,
min_support=min_support, use_colnames=True, max_len=max_len)
        if frequent_itemsets.empty:
            print(f'No frequent itemsets found for
min_support = {min_support}')
            return frequent_itemsets, pd.DataFrame()

        rules = association_rules(frequent_itemsets,
metric='confidence', min_threshold=min_confidence)
        if rules.empty:
            print(f'No rules found for min_support =
{min_support} and min_confidence = {min_confidence}')
            return frequent_itemsets, rules

        filtered_rules = rules[rules['lift'] > 1]
        return frequent_itemsets, filtered_rules


    # Параметри для FP-Growth
    fpgrowth_parameters = [
        (0.01, 0.4, 2),
        (0.01, 0.3, 5),
        (0.001, 0.2, 2),
        (0.001, 0.1, 10),
    ]

    # Параметри для Apriori
    apriori_parameters = [
        (0.001, 0.4, 2),
        (0.001, 0.3, 2),
        (0.0005, 0.2, 3),
        (0.0005, 0.1, 5),
    ]

    # Списки для збереження результатів
    fpgrowth_rules_count = []
    apriori_rules_count = []
    fpgrowth_confidence_levels = []
    apriori_confidence_levels = []

    # Запуск FP-Growth для кожної пари параметрів
```

```
    for min_support, min_confidence, max_len in
fpgrowth_parameters:
        fpgrowth_confidence_levels.append(min_confidence)

        _, fpgrowth_rules = run_fpgrowth(basket,
min_support, min_confidence, max_len)
        fpgrowth_rules_count.append(len(fpgrowth_rules))

    # Запуск Apriori для кожної пари параметрів
    for min_support, min_confidence, max_len in
apriori_parameters:
        apriori_confidence_levels.append(min_confidence)

        _, apriori_rules = run_apriori(basket, min_support,
min_confidence, max_len)
        apriori_rules_count.append(len(apriori_rules))

    # Побудова графіка для FP-Growth
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.plot(fpgrowth_confidence_levels,
fpgrowth_rules_count, label='FP-Growth', marker='o',
color='b')
    plt.title('FP-Growth: Залежність кількості правил від
мінімального порогу довіри')
    plt.xlabel('Мінімальний поріг довіри (confidence)')
    plt.ylabel('Кількість правил')
    plt.legend()
    plt.grid(True)

    # Побудова графіка для Apriori
    plt.subplot(1, 2, 2)
    plt.plot(apriori_confidence_levels, apriori_rules_count,
label='Apriori', marker='o', color='g')
    plt.title('Apriori: Залежність кількості правил від
мінімального порогу довіри')
    plt.xlabel('Мінімальний поріг довіри (confidence)')
    plt.ylabel('Кількість правил')
    plt.legend()
    plt.grid(True)

    # Показати графіки
    plt.tight_layout()
    plt.show()

    # Закриття підключення до бази даних
conn.close()
```